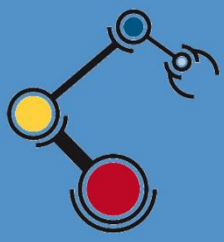
Escuela Politécnica Superior

Estudio del control mioeléctrico en 2D a partir de métodos de regresión lineal



Máster Universitario en Automática y Robótica

Trabajo Fin de Máster

Autor:
Cristina Antelo del Río

Tutor/es:
Andrés Úbeda Castellanos
Santiago Timoteo Puente Méndez

Universitat d'Alacant
Universidad de Alicante

Diciembre 2018

Resumen

Uno de los métodos más recientes de control mioeléctrico es el basado en regresión lineal. Este proyecto propone estudiar su uso para evaluar el control en un espacio bidimensional. Para ello, en este proyecto se utiliza la actividad de varios músculos del brazo medida a través de electromiografía superficial (EMG) para obtener comandos de control en dos dimensiones (x e y). Como procesamiento de esas señales se aplica el método de la regresión lineal.

Una vez realizado el procesamiento y generación de los valores calculados de las trayectorias, se evalúa la eficiencia del algoritmo comparando las trayectorias decodificadas con las trayectorias realizadas por los sujetos. Los métodos de predicción de errores utilizados son: el Coeficiente de Correlación (CC), la Raíz Normalizada del Error Cuadrático Medio (NRMSE), el Ratio de la Señal de Ruido (SNR), el Coeficiente de Determinación (COD), la Desviación Global (GD) y mezclas entre ellos. Se han realizado dos estudios de las señales: uno a partir de los 4 canales de las señales electromiográficas y otro seleccionando únicamente los canales pertenecientes a una dimensión.

Las conclusiones finales son que no presenta una mejora al separar los canales para el estudio, ya que da lugar a pérdida de información. Además, el mejor método para comprobar la efectividad del algoritmo es el CC+SNR ya que muestra los errores de ruido y de escalado.

Contenido

Índice de Figuras:

Índice de Tablas:

1 INTRODUCCIÓN:

Este proyecto se basa en el estudio de los movimientos de la mano, que se pueden detectar a partir de variaciones en los músculos del antebrazo. Las trayectorias seguidas por la mano son analizadas mediante el método de regresión lineal realizado a partir de las señales electromiográficas (EMG) recogidas de 4 pares de electrodos (Ag/AgCl) ubicados en el antebrazo. Para una localización más precisa de los puntos de la trayectoria seguida, se ha empleado también un sensor GY-521, con el que se pueden calcular los ángulos de giro.

En este apartado se enumeran los objetivos cumplidos y se muestran los antecedentes más significativos del estudio. En el apartado 2 se explican qué son las señales electromiográficas, los diferentes métodos de recogida de esas señales, y otras características necesarias para tener una idea clara de qué significan los datos medidos. En el apartado 3 se describen los elementos de hardware necesarios para la realización del estudio: una placa Arduino y el equipo Noraxon MiniDTS. Para mejorar la precisión en las medidas se ha utilizado también el Módulo GY-521 explicado en el apartado 4. Una vez definidos todos los elementos necesarios se detalla la metodología seguida en el apartado 5. Este capítulo engloba el procedimiento de adquisición de datos y la programación del clasificador. En el apartado 6 se describe la interfaz hombre-máquina (HMI) programada en MATLAB que controla el calibrado inicial del sistema, la recogida de datos, y su posterior análisis offline. En el apartado 7, se muestran los resultados obtenidos gráfica y analíticamente, y en el apartado 8 las conclusiones y trabajos futuros.

1.1 Objetivos

- Definición de movimientos y adquisición de la señal EMG mediante el equipo NORAXON MiniDTS.
- Obtención de los parámetros necesarios del acelerómetro-giroscopio GY-521 para la determinación de la posición a partir del Arduino UNO.
- Recogida de datos EMG y del sensor simultáneamente.
- Implementación del algoritmo del clasificador de regresión lineal.
- Creación de una HMI para la adquisición de datos y empleo del clasificador con los datos offline.

1.2 Estado del arte

En 1783, Luigi Galvani descubrió el resultado eléctrico de las contracciones musculares al diseccionar una rana en una mesa donde habían experimentado con electricidad estática. Este resultado eléctrico se denominó posteriormente "electricidad animal". En 1846, Carlo Matteucci inventó el "kymograph" (Figura 1.1). Se trataba de un dispositivo mecánico capaz de registrar las contracciones musculares mediante un movimiento físico [1].

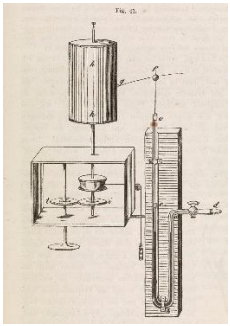


Figura 1.1 Primer "kymograph" [1]

Desde mediados del siglo XX se ha investigado el uso de señales EMG como elemento biotecnológico. El primer documento que hace referencia a la electromiografía data de 1666 [2], en el que Francesco Redi (1626-1697) estudia los defectos nerviosos del pez torpedo.

En el 1950, la empresa DISA A/S (Dinamarca) en colaboración con Fritz Buchthal, introdujo el primer equipo comercial de electromiografía (Figura 1.2), basado en circuitos electrónicos analógicos [2]. Esta máquina era capaz de medir 3 señales EMG independientes y mostrar las formas de onda de cada canal, que se grababan en una película de papel. Para poder procesar en profundidad los datos de cada forma de onda había que extraer la película y analizarla en un cuarto oscuro.

Figura 1.2 Equipo EMG DISA 13A67 [2]

Una década más tarde, los equipos empezaron a estar compuestos de placas de circuitos impresos y la película de papel dio paso a las instantáneas de una cámara polaroid. A esta

[1] https://www.agenciacyta.org.ar/2014/01/%E2%80%9Cla-bioingenieria-se-presenta-llena-de-posibilidades%E2%80%9D/
[2] Jee Hong Quach [1]

mejora también hay que añadirle el paso a la era digital en los años 70, siendo el primer equipo completamente digital el DISA 1500 (Figura 1.3).

Figura 1.3 Equipo EMG DISA 1500 [3]

Actualmente, los equipos se componen de microprocesadores capaces de adquirir señales y procesarlas. Este último avance ha supuesto un gran aumento en la fiabilidad de las medidas [3]. A su vez, se ha optado por hacer un sistema EMG basado en PC, que da como resultado un procesamiento de señal elevado, una representación gráfica variable y mayores capacidades de almacenamiento. También se han desarrollado dispositivos EMG portátiles en los que los datos se guardan en una memoria interna y que se pueden volcar posteriormente a un ordenador.

Figura 1.4 Equipo EMG portátil MIOQUICK Matrix Line de MICROMED [4]

Las señales EMG se utilizan principalmente en terapia o en entrenamiento, pero tienen su principal campo de acción en la medicina. Algunos de los ejemplos que se controlan a partir de las señales mioeléctricas (MES) son: prótesis, exoesqueletos, teleoperación de robots y realidad virtual. Una de las primeras aplicaciones fue el control de prótesis simples. Estas prótesis tienen dos principios de funcionamiento, de manera neumática accionándola con el cuerpo, o de forma eléctrica. El primer grupo obtiene energía de los músculos, pero a veces, dependiendo del usuario, resulta complicada su implementación. Su principal ventaja es el bajo coste. En cambio, el segundo grupo utiliza materiales más caros y más pesados y

[3] http://www.neurologyindia.com/viewimage.asp?img=ni_2018_66_2_459_227272_f18.jpg
[4] http://tps.co.rs/oprema%20za%20neurofiziologiju.html

está formado por baterías. Se activa a partir de señales de presión, interruptores, medidores de tensión, y desde hace unos años a partir de señales EMG.

La primera prótesis basada en las señales mioeléctricas la diseñó, en 1948, Reiter R. [4]. El método elegido de entrenamiento era por medio de discriminación Bayesiana, logrando clasificar seis clases de movimientos distintos con un 99% de éxito. El inconveniente era la cantidad de horas necesarias para desarrollar el entrenamiento requerido. El objetivo era lograr controlar el dispositivo de la prótesis de mano a partir de la intención del usuario. Más adelante se empezó también a controlar la variación en la cantidad de fuerza dependiendo de la amplitud de la onda.

La primera prótesis de mano comercial se fabricó en el 1957 en el Instituto Central de Investigación de Prótesis (Moscú) a partir de motores paso a paso. Posteriormente se introdujo el motor de imán permanente y los relés electromagnéticos.

Actualmente se está desarrollando otro tipo de prótesis controlada también a partir de señales MES (Figura 1.5). Son las denominadas manos artificiales. Son dispositivos ligeros, con poco mantenimiento y una gran capacidad de agarre. La mano mioeléctrica recibe las señales EMG que realiza el sujeto al activar un músculo determinado. Su principal inconveniente es el control a partir de señales electromiográficas superficiales, ya que no las detecta correctamente. Se está estudiando la opción de cambiar el modo de adquirir las señales, realizando un control de la mano a partir de los nervios del brazo mediante una cirugía de reinervación muscular dirigida (TMR), implantando un sensor mioeléctrico en el que se conectan nervios a diferentes músculos para poder medir las señales desde la superficie.

Figura 1.5 Ejemplo de mano mioeléctrica [5]

En el trabajo de Pinzón, J. et Al. [5] se ha simulado un brazo robótico por medio de las señales EMG recogidas en la totalidad del brazo. El conjunto está controlado por medio de una placa Arduino y por elementos analógicos como amplificadores operacionales, diferenciales, resistencias, potenciómetros y condensadores. A su vez ha utilizado como

[5] http://fissioterapia.blogspot.com/2013/02/las-protesis-mioelectricas.html

elementos de medición un osciloscopio y un multímetro. El esquema analógico ha sido el siguiente:

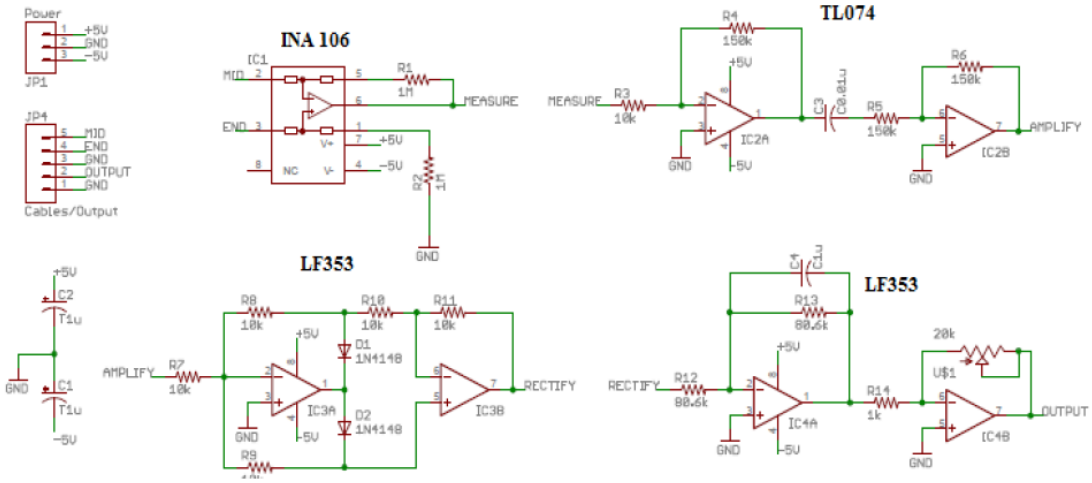


Figura 1.6 Esquema electrónico de un brazo robótico [6]

Para el análisis de las señales EMG no se suele probar sólo un método, ya que varía mucho el modo de la adquisición de datos y su pre-procesamiento. En el estudio realizado por Sekiya et al. [6] comparan el método de la regresión lineal con el de las redes neuronales artificiales para la clasificación de las señales EMG. En él se explica que el método de la regresión lineal es uno de los métodos más comunes en estadística, y que se han realizado muchos estudios con él en el pasado. Los estudios tenían como objetivo aumentar las variables de entrada y encontrando formas de realizar la mejor correlación de las variables de salida, lo que supuso una complicación del modelo inicial. Crearon un método más fiable de regresión denominado regresión lineal logística, que convertía una función no lineal en un modelo lineal.

En el artículo de Hahne et al [7] compara las técnicas de regresión lineal y no lineal para el control de una muñeca con 2 DoF (Grados de Libertad) a partir de señales EMGs. Estas técnicas son: regresión lineal, mezcla de expertos lineales (ME) y regresión de la cresta de del núcleo (KRR). La conclusión es que, si se pudiese linealizar el problema con algún método, la regresión lineal tendría un rendimiento similar al KRR, reduciendo el costo computacional.

Smith et al. [8] estudió un procedimiento para realizar el análisis de regresión lineal a las señales electromiográficas intramusculares en las prótesis para amputados transradiales con varios grados de libertad a partir del aislamiento del estudio individualizado de cada grado de libertad.

[6] Pinzón et al. [5]

2 SEÑALES ELECTROMIOGRÁFICAS

2.1 Definición

Las señales EMG analizan la suma de la actividad eléctrica o de las variaciones de potencial eléctrico en todas las fibras musculares (MUAPs), al producirse un movimiento de contracción y relajación. Por esta razón, el sistema es fuertemente dependiente del correcto posicionamiento de los electrodos en las regiones musculares deseadas. En la figura siguiente se puede observar perfectamente las etapas de contracción y relajación, presentando las primeras los valores más elevados de amplitud.

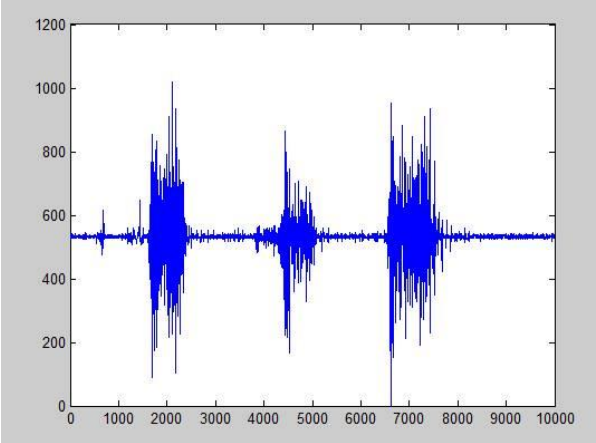


Figura 2.1 Ejemplo de señal EMG [7]

Esas contracciones de los músculos son debidas a impulsos nerviosos que transmiten las unidades motoras o motoneuronas, que son capaces de responder a un estímulo con la fabricación de unas enzimas que se van propagando por el nervio central desde el cerebro hasta el músculo que se desea mover [5].

Las características que se estudian a partir de los datos recogidos en las gráficas son: el valor del pico máximo, la duración y los cambios de fase. La amplitud y la calidad de las señales EMG dependen de varios factores, entre ellos:

- Tiempo e intensidad de la contracción muscular.
- Distancia entre electrodo y zona de actividad muscular.
- Propiedades de la piel, del electrodo y del amplificador.
- Fatiga muscular y cantidad de oxígeno disponible en el metabolismo.

[7] https://sean441.wordpress.com/2015/09/22/segundo-dia/

2.2 Clasificación

Dependiendo de la forma de adquisición de los datos existen dos tipos de señales EMG: invasivas y no invasivas.

2.2.1 Método no invasivo

Señales electromiográficas de superficie (EMGs) en las que los electrodos se localizan en la superficie de los músculos activos, es decir, adheridos a la piel. Es el método más común de medida y el más aconsejado por la SENIAM (Grupo de Estudio sobre EMGs para una evaluación no invasiva de los músculos). Estos electrodos son diferenciales, es decir, miden la diferencia de potencial existente entre los dos valores. De esta manera se puede reducir el ruido en el resultado. Normalmente, por cada canal, se colocan dos electrodos de medida y uno de referencia. Este último se sitúa en una zona de baja actividad muscular, como pueden ser los huesos. La dirección de propagación de la señal entre los electrodos diferenciales va de 2 a 6 m/s. A continuación, se muestra un esquema del proceso:

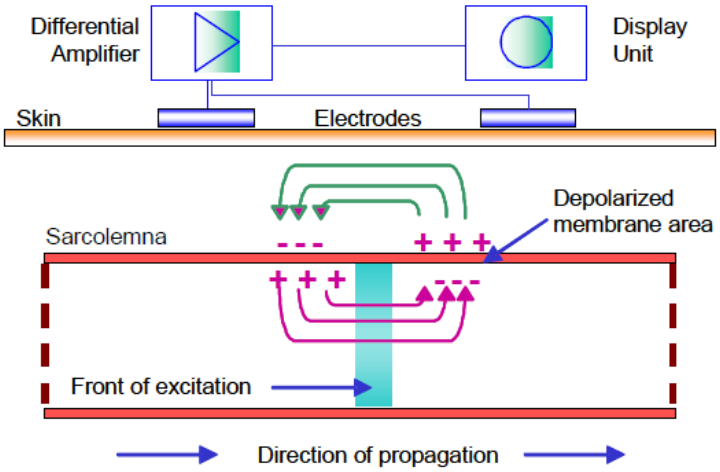


Figura 2.2 Principio de funcionamiento de los electrodos diferenciales [8]

El resultado es la creación de una onda que viaja a lo largo de la superficie de la fibra muscular (Figura 2.3). En ella se muestra una onda propagándose en el tiempo. En el eje de abscisas se localiza la diferencia de potencial entre los electrodos diferenciales y en el eje de ordenadas, el tiempo de propagación. Se puede observar que en el tiempo T1 los electrodos aún no han sido cargados eléctricamente, por lo que la diferencia de potencial es 0. Ésta diferencia se pone igual a 0 cuando la señal eléctrica está a la misma distancia entre un electrodo y otro [9].

[8] Konrad [9]

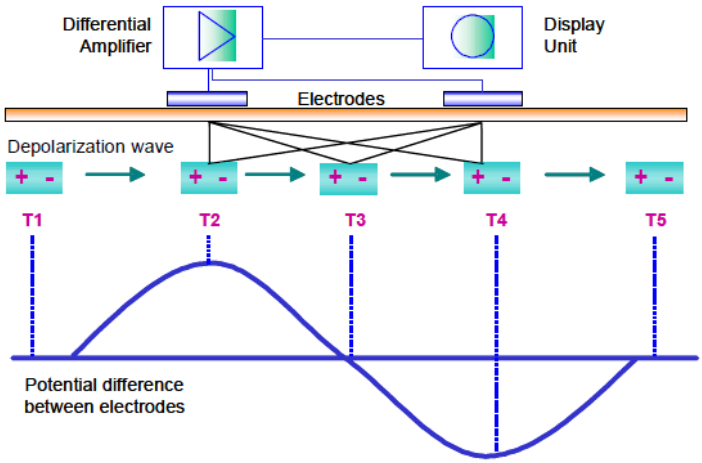


Figura 2.3 Diferencias de potencial de una señal EMG [9]

Al ser una diferencia de potencial, es una resta, por ese motivo la onda es sinusoidal, es decir, tiene parte positiva y parte negativa dependiendo de dónde se localice la señal eléctrica en cada momento.

Los electrodos pueden ser de diferentes formas (circular, rectangular y cuadrada) y tamaños. Las señales registradas varían considerablemente dependiendo del tamaño del electrodo, ya que, a mayor dimensión, menor será la cantidad de señales obtenidas a alta frecuencia [10]. También se diferencian en el material: los electrodos más típicos son los de disco de platino, los electrodos Ag/AgCl pregelificados, y los electrodos de gel húmedo. La diferencia entre ellos es que los de disco de platino son reutilizables completamente. Los otros se pueden reutilizar, pero van perdiendo fiabilidad en la medida. En la figura siguiente se muestran los tres tipos (de disco, Ag/AgCl y de gel húmedo):

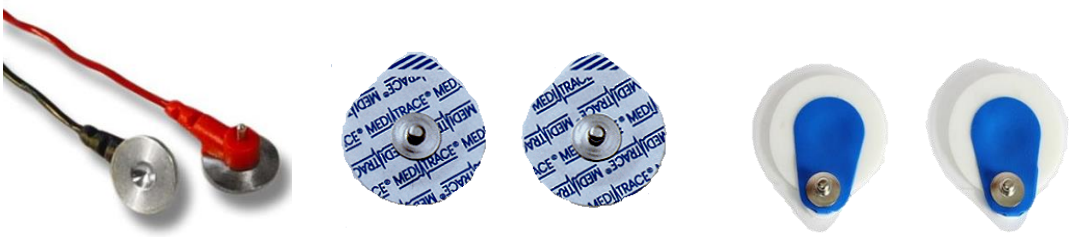


Figura 2.4 Ejemplos de electrodos de superficie [10] [11] [12]

[9] Konrad [9]
[10] https://www.gvb-spes.es/accesorios-de-emg/electrodos-emg/stimulation/247/electrodos-de-disco-estimulacion
[11] http://neuroline.es/?page_id=316
[12] https://spanish.alibaba.com/product-detail/Cheap-Price-Disposable-Medical-Blue-Round-60717751217.html

2.2.2 Método invasivo

Señales electromiográficas intramusculares, en las que el electrodo se localiza en el interior de los músculos. Se utiliza una aguja hipodérmica que se inserta hasta el tejido muscular.

Para este método es necesario la presencia de un profesional entrenado que domine perfectamente el vientre de cada músculo que se vaya a estudiar. De esta manera no se dañará el tejido y las medidas serán completamente fiables.

El proceso a seguir es el siguiente: una vez introducido el electrodo, se procede a su retirada unos mm para luego volver a insertarlo en una posición muy cercana. Cada medida del electrodo muestra resultados de una zona muy limitada, por lo que al ir retirando e insertando se van obteniendo distintos puntos cercanos.

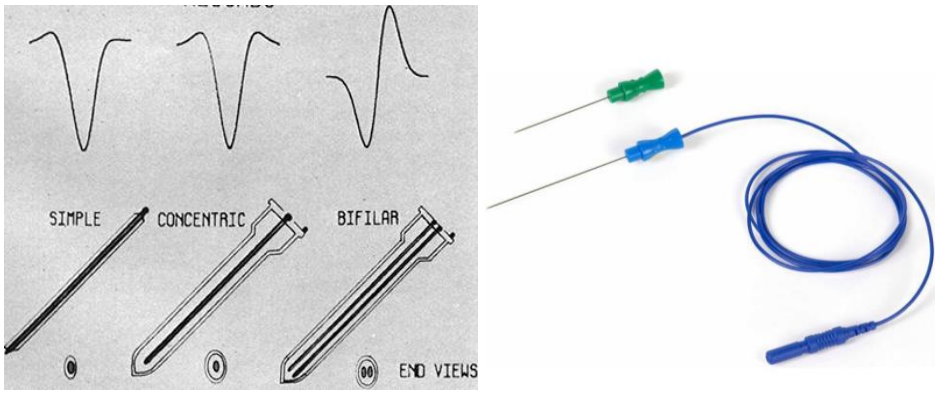


Figura 2.5 Agujas método EMG intramuscular [13]

2.2.3 Ventajas e inconvenientes de las EMGs

Ventajas:

- Reutilización de una persona a otra
- Facilidad para recolectar muestras
- Mayor campo de estudio, ya que en el método intramuscular sólo interviene un número pequeño de fibras.
- Los músculos no sufren daño, por lo que el paciente no queda adolorido después del procedimiento.

[13] https://interferenciales.com.mx/products/agujas-monopolares-desechables-pre-cableadas

Inconvenientes:

- Muy sensibles al ruido debido a la presencia de campos electromagnéticos cercanos.
- Es necesario la aplicación de un gel para una mejor conductividad y adherencia. En sí hay algunos electrodos que ya vienen con el gel, por lo que en la mayoría de los casos no supone un inconveniente.
- Las señales recogidas son más débiles, por lo que hace falta un procesamiento previo de filtraje y amplificación.
- El método EMG de superficie no sirve cuando el músculo a medir es pequeño, está superpuesto o no se detecta en la piel.

En resumen, con respecto a un estudio realizado por Englehart et al.[4], comparando seis patrones de movimientos mediante los dos modelos de obtención de datos, se ha podido comprobar que no existe diferencia significativa en la precisión de la clasificación. Por lo que el método EMGs es el más utilizado, ya que no presenta ningún riesgo al paciente.

3 HARDWARE EMPLEADO

3.1 Arduino

Es tanto un hardware libre como un software libre. Se denomina hardware a las partes físicas tangibles de un sistema informático, es decir, sus componentes eléctricos, electrónicos, mecánicos y electromecánicos. Según la Real Academia Española de la Lengua (RAE), un software se define como "*un conjunto de programas, instrucciones y reglas informáticas que sirven para ejecutar ciertas tareas en un ordenador*". A su vez, al ser libres, sus especificaciones y diagramas esquemáticos son de acceso público, ya sea mediante incentivo de dinero o no.

El software de Arduino se trata de una plataforma de código abierto. En internet hay disponibles cursos, tutoriales, herramientas de consulta… esto es debido a que es una comunidad gratuita en dónde las personas pueden subir sus trabajos, resolver dudas, comentar nuevas ideas, elaborar librerías para facilitar su uso, etc.

Con respecto a su hardware, puede presentar mejores características a bajo coste incorporando alguna Shield. Una Shield o escudo es una placa apilable con las mismas dimensiones que el Arduino original, que se encaja en este último para mejorar sus capacidades. Por ejemplo, tomando como referencia el Arduino Uno, podría ser una Wifi Shield o una Motor Shield. Tiene las conexiones de tal forma que cuadran perfectamente con cada pin y con la distancia entre pines. Se pueden comunicar con el Arduino por alguno de los pines o por algún bus de datos.

Existen multitud de modelos Arduino, por ello hay que saber elegir qué tipo de microcontrolador y qué comunicación se quiere con el ordenador. Algunos de ellos son: Arduino Yun, Arduino Due, Arduino Mega, Arduino Ethernet, Arduino Leonardo, Arduino Mini o Arduino Nano.

El Arduino UNO, que es el que se ha utilizado, presenta las siguientes características:

[14]

Microcontrolador	ATmega328P
Tensión trabajo	5 V
Tensión de entrada	7-12 V
Pines de E/S digital	14 (6 pueden ser salidas PWM)
Pines de E/S analógica	6
Corriente DC por pin E/S	20 mA
Memoria Flash	32 kB
SRAM	2 kB
EEPROM	1 kB
Frecuencia	16 MHz
Wifi	-
Ethernet	-

Tabla 3.1 Características Arduino UNO

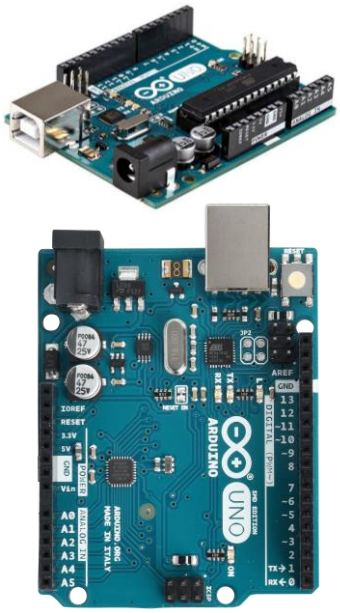


Figura 3.1 Arduino UNO

Viendo los datos de la Tabla 2.1, se diferencian 3 memorias. Una memoria SRAM es aquella dónde el Arduino crea y manipula las variables cuando se ejecuta. Una memoria EEPROM es una memoria no volátil para guardar los datos por si se apaga el dispositivo o se resetea. Una memoria flash es la memoria del programa [11].

3.2 Noraxon MiniDTS

El "Mini Direct Transmission System" (DTS) o Mini Sistema de Transmisión Directa es un equipo de medida de EMGs portátil con un tamaño y un peso reducido, y una fiabilidad similar a la de los sistemas convencionales. Un inconveniente es la cantidad de canales a medir, ya que presentan un número menor que los grandes equipos. La ventaja es que su coste es la mitad, por lo que está especialmente recomendado para proyectos de investigación o para la enseñanza.

Está compuesto por 3 elementos principales (Figura 3.2): el módulo de recepción de datos, el electrodo de referencia (transmisor), y la estación de recarga de los sensores EMG. Además, es necesario un el cable de envío de datos de los electrodos Ag/AgCl al transmisor.

[14] Fuente propia

Figura 3.2 Elementos sistema miniDTS [15]

Los módulos transmisores contienen un sistema de adquisición de datos con un preamplificador de las señales EMG. Cada módulo está alimentado con una batería de 190 mAh. Presenta un disco de metal en su parte inferior que tiene que estar en contacto con la piel del paciente mediante una cinta de doble cara. Este disco de metal se considera la referencia entre las dos señales diferenciales extraídas de los electrodos Ag/AgCl.

Los datos de las señales EMG se envían por medio de señales de radio que llegan a la antena del módulo de recepción. Este módulo transfiere los datos al ordenador por medio de un cable USB. Por último, éstos se procesan en el software de análisis biomecánico de Noraxon (MR3 myoMUSCLE), del que se hablará en el apartado 6. En las figuras de abajo se muestra un esquema de los componentes principales del sensor EMG y del receptor, extraídos del manual de instrucciones [12]:

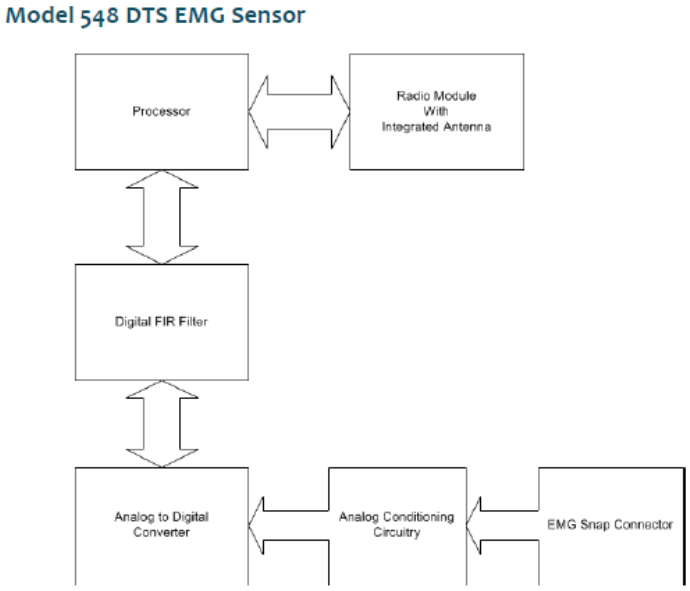


Figura 3.3 Esquema del sensor EMG modelo 548 de Noraxon [16]

[15] https://www.noraxon.com/our-products/mini-dts/
[16] Noraxon [12]

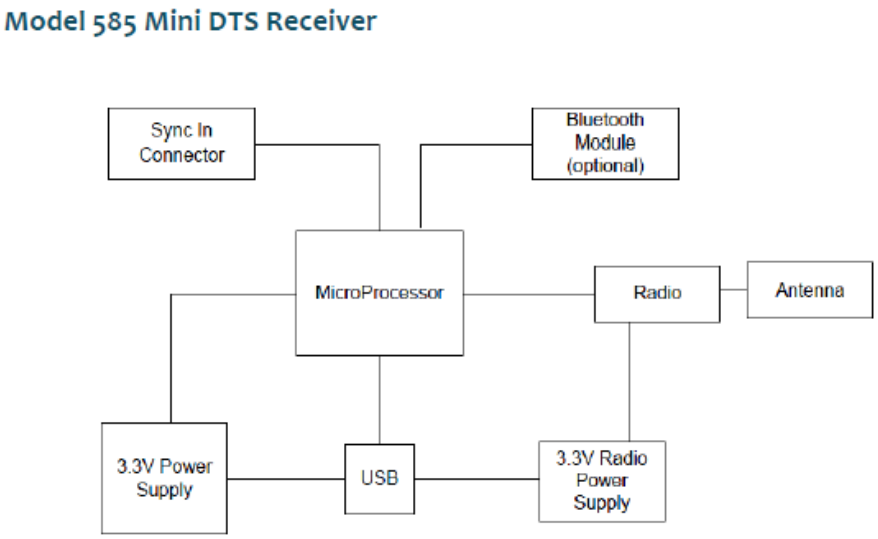


Figura 3.4 Esquema del receptor mini DTS 585 de Noraxon [17]

Los datos técnicos que se muestran a continuación son de la página oficial de Noraxon [13]:

Inalámbrico	Sí	**CMRR**	>100dB
Frecuencia de muestreo	1500, 3000	**Impedancia de entrada**	>100MΩ
Resolución	16 bits	**Ruido base**	<1 µVRMS
Canales	4	**Tamaño sensor EMG**	3.4 x 2.4 x 1.4 (cm)
Ganancia	500	**Tamaño del receptor**	7.66 x 10.18 x 3.55

Tabla 3.2 Datos técnicos Noraxon miniDTS [18]

[17] Noraxon [12]
[18] Noraxon web [13]

4 MÓDULO GY-521

4.1 Definición

Shield diseñado por InvenSense especialmente para Arduino, que se puede acoplar perfectamente a cualquier modelo. Consiste en una unidad de medición inercial (IMU). Contiene el sensor MPU-6050, un acelerómetro-giroscopio con 6 grados de libertad (6 DOF). Este módulo se utiliza principalmente en el control de drones o como detector de caídas o sensor de distancia y de velocidad.

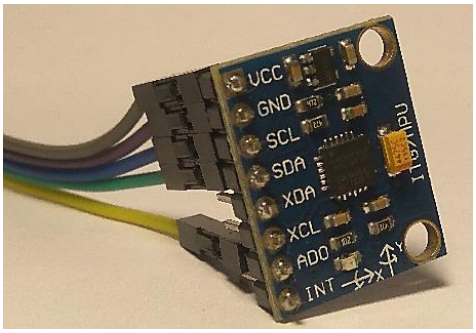


Figura 4.1 Shield GY-521 [19]

Tensión de entrada	3.3 - 5 V
Comunicación	Bus I2C (MPU-6050)
Gama giroscopio	±250 ±500 ±1000 ±2000 º/seg
Rango de aceleración	±2 ±4 ±8 ±16 G
Dimensiones	2.1 x 1.6 x 0.3 cm
Consumo máximo	3.9 mA

Tabla 4.1 Especificaciones técnicas GY-521[20]

El acelerómetro evalúa la aceleración o inclinación a la que se ha sometido el módulo. Esta última propiedad se utiliza para la detección de vibraciones. El sensor MPU-6050 está diseñado a partir de la tecnología MEMS (Sistemas Microelectromecánicos), en la que todos sus componentes son del orden de micras. En la Figura 4.2 se muestra una foto de la zona relativa al acelerómetro, realizada a partir de una cámara con un objetivo de aumento. En la fotografía también se puede observar el procedimiento de medición del acelerómetro.

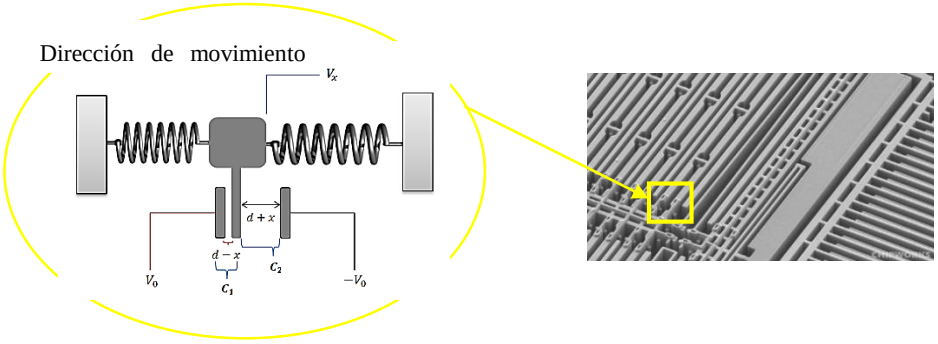


Figura 4.2 Acelerómetro tipo MEMS [21]

[19] Fuente propia
[20] Datasheet GY-521
[21] https://www.fayerwayer.com/2009/06/el-boom-de-los-acelerometros-w-guia/

Teniendo en cuenta la 1ª Ley de Newton:

$$F = m * a$$

Existen dos placas fijas y una móvil. Al permanecer con movimiento constante o en reposo, la placa móvil vuelve a la posición inicial. Al variar la aceleración de un objeto a la derecha, la masa del objeto empuja el muelle y, en la placa móvil se ejerce una fuerza contraria, moviendo la plaquita hacia la izquierda. Se mide la variación de las distancias entre las placas fijas y la móvil y se determina el valor y la orientación [14].

El giroscopio es el encargado de medir los ángulos de rotación en los ejes X, Y y Z. El sentido de giro positivo de cada eje se muestra en la Figura 4.3. El sensor tiene un giroscopio integrado del tipo vibratorio de efecto Coriolis (CVG), que tienen la capacidad de vibrar en el mismo plano a pesar de la rotación. El sensor presenta partes móviles que se someten a vibración deformando la estructura debido al efecto Coriolis, lo que supone una variación en la distancia entre dos placas localizadas a ambos lados [15]. El principio es similar al del acelerómetro, ya que mide la variación de la capacitancia o de la distancia entre las placas. En la Figura 4.4 se puede apreciar la fotografía del giroscopio con una escala de 100µm.

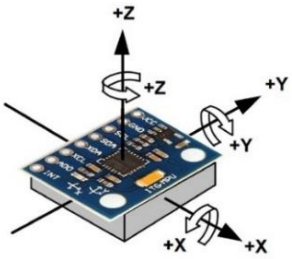


Figura 4.3 Sentidos de giro de los ejes XYZ en el módulo GY-521 [22]

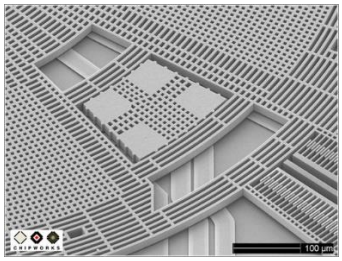


Figura 4.4 Giroscopio integrado MPU-6050 [23]

Normalmente se suelen acoplar, dentro de un mismo sensor, un acelerómetro y un giroscopio. Esto es debido a que el giroscopio presenta una mayor precisión en tiempos cortos y responde muy bien a cambios bruscos. Sin embargo, presenta una acumulación de errores de medición y ruido, al ser necesaria una integral para calcular el ángulo de giro. El acelerómetro funciona al revés: en tiempos cortos tienen demasiado ruido por lo que son poco fiables, pero en tiempos medios o largos, al hacer siempre un promedio entre sus valores, la precisión aumenta.

[22]https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html
[23] https://www.microsiervos.com/archivo/tecnologia/giroscopio-digital-microscopio.html

4.2 Comunicación bus I2C

Para empezar a hablar de una comunicación entre dos elementos, es necesario ver las conexiones de la placa Arduino Uno con el módulo GY-521 para determinar quién es el maestro y quién el esclavo, y otros datos relevantes como la dirección del puerto de comunicación. Los pines de conexión con el Arduino se muestran en la siguiente tabla:

GY-521	Arduino UNO	Color Cable
VCC	3.3V	Gris
GND	GND	Morado
SCL	Pin Analógico 5	Azul
SDA	Pin Analógico 4	Verde
AD0	GND	Amarillo

Tabla 4.2 Conexión GY-521 con Arduino UNO [24]

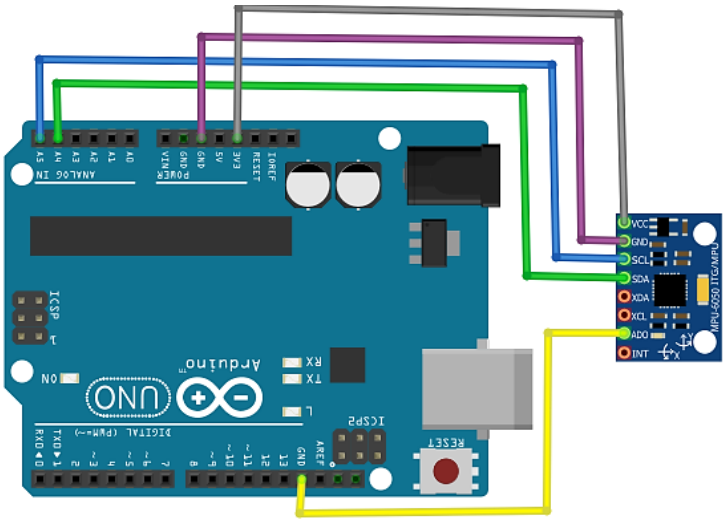


Figura 4.5 Esquema de conexionado GY-521 con Arduino Uno con Fritzing [25]

Como se puede observar en el esquema de la Figura 4.5, el módulo GY-521 actúa como esclavo, ya que se conecta mediante los pines SDA y SLC a los pines A4 y A5 que es dónde se localizan los pines SDA y SLC en el Arduino Uno. El Arduino es el maestro, es decir, el que envía una señal para que el otro realice una tarea determinada. En este proyecto, el Arduino le demanda los datos de medida, guardados en la memoria flash del MPU-6050, en un tiempo determinado a través del bus I2C. La comunicación del bus I2C se puede dar a partir de la dirección 0x68 (LOW) o 0x69 (HIGH). En este caso el bit de comunicación es 0x68, ya que se conecta el pin AD0 al GND del Arduino (cable amarillo).

[24] Fuente propia
[25] Fuente propia

El bus I2C (Iter-Integrated Circuit) está compuesto por dos cables: el del reloj (SCL) y el de los datos (SDA) [16]. En la Figura 4.6 se muestra el conexionado interno que conecta el maestro (Master) con cuatro esclavos (Slave). En la comunicación I2C pueden existir un número ilimitado de esclavos, pero normalmente sólo hay un maestro. Cuanto mayor sea el número de esclavos, más compleja resulta la programación. El maestro es el encargado de controlar las transferencias. En el caso de este proyecto existen un maestro y un esclavo, tal como se ha mencionado arriba.

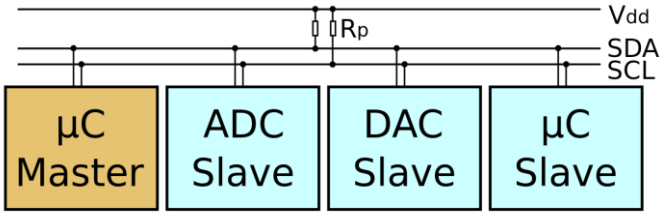


Figura 4.6 Comunicación bus I2C [26]

La trama de datos que se envía a través de la comunicación I2C sigue siempre la secuencia que se muestra en la Figura 4.7. El proceso a seguir es el siguiente:

1º) El maestro comienza la comunicación enviando una condición de START, poniendo los esclavos a la espera.

2º) El maestro envía 1 byte. Los primeros 7 bits (A7-A1) se corresponden con la dirección de memoria a la que se quieren enviar los datos. Luego, el bit siguiente (A0) indica 1 si es de lectura (R) o 0 si es de escritura (W) y un bit de validación (ACK).

3º) El esclavo comprueba que la dirección enviada por el maestro coincide con la suya. Este paso es muy importante cuando hay más de un esclavo, para que sólo reciba la información el esclavo al que va dirigido. Así se consigue reducir la velocidad de transmisión.

4º) El maestro envía los datos agrupados en bytes, siendo el último bit el de validación.

5º) El envío de datos sigue hasta que el maestro le transmita al esclavo la condición de STOP.

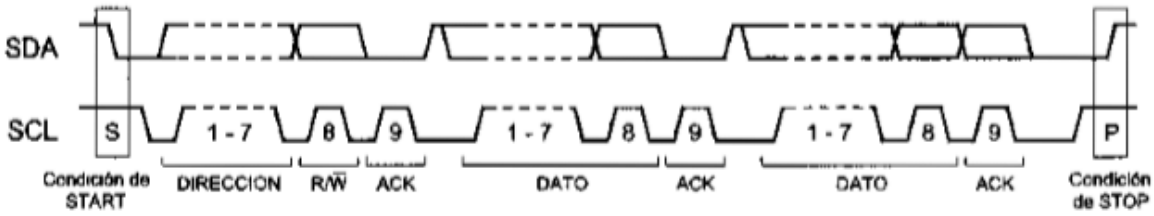


Figura 4.7 Secuencia de tramas comunicación I2C [27]

[26] https://es.wikipedia.org/wiki/I%C2%B2C
[27] https://www.i-ciencias.com/pregunta/36253/como-mostrar-i2c-direccion-en-hexadecimal

Uno de los principales inconvenientes de este tipo de comunicación bidireccional es su reducida velocidad en el envío y recepción de datos. La velocidad estándar de transmisión es de 100 MHz, con un modo de alta velocidad de 400 MHz cuya velocidad de transferencia es de 50kbytes/sec. Otro inconveniente es que no se sabe con certeza si los datos recibidos son los que se han enviado. Al enviar el bit de validación se sabe si el byte es el mismo (el mensaje), pero no especifica el valor de los datos., ya que no pasa por ningún método de testeo.

5 METODOLOGÍA

- En primer lugar, se deberán escoger los movimientos de brazo la mano que se van a clasificar y determinar los músculos que intervienen.

- En segundo lugar, se realizarán registros de los distintos movimientos mediante las señales EMG y se medirá la trayectoria seguida a partir del sensor GY-521.

- En tercer lugar, y se implementará un algoritmo de decodificación basado en el método de regresión lineal a partir de los resultados extraídos en el paso anterior.

- Por último, se analizarán los resultados obtenidos para evaluar la efectividad de los algoritmos de adquisición, procesamiento y decodificación. Este último paso se describe en el apartado 6.

A continuación, se muestra un esquema general del proceso, desde el estudio del movimiento a realizar hasta la evaluación de los resultados:

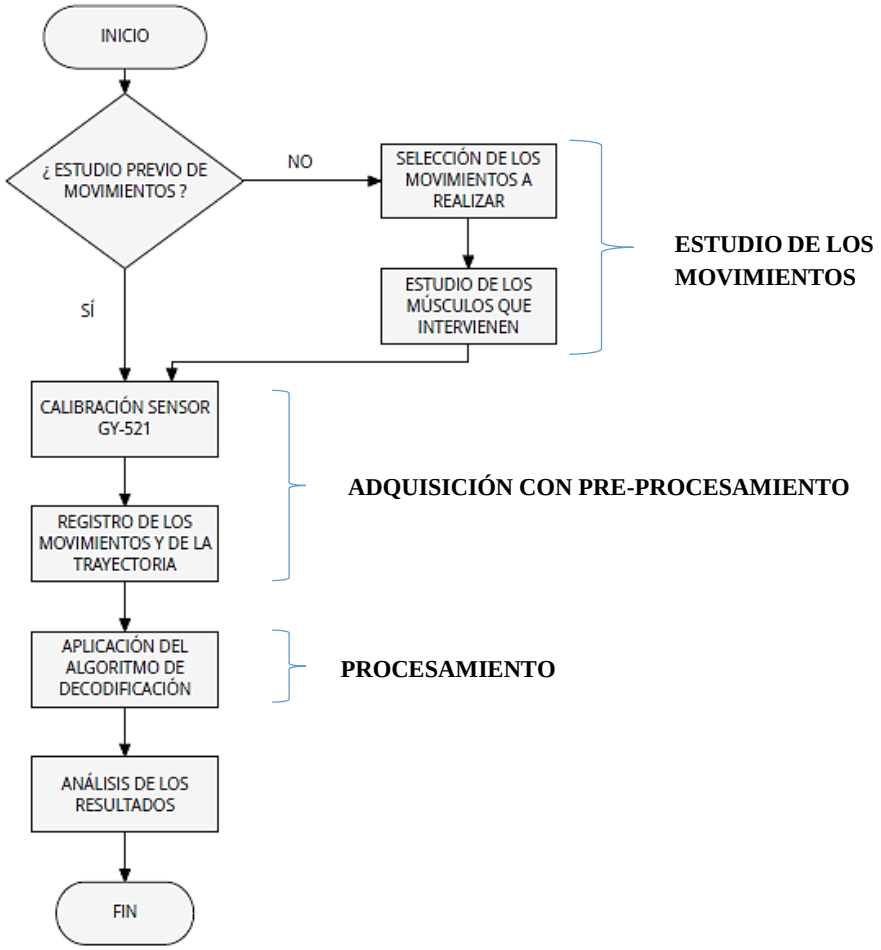


Figura 5.1 Esquema del proceso [28]

[28] Fuente propia

5.1 Estudio de los movimientos

En este proyecto se han estudiado los gestos que se muestran en la Figura 5.2, que se corresponden con los movimientos de flexión/extensión y de abducción/aducción:

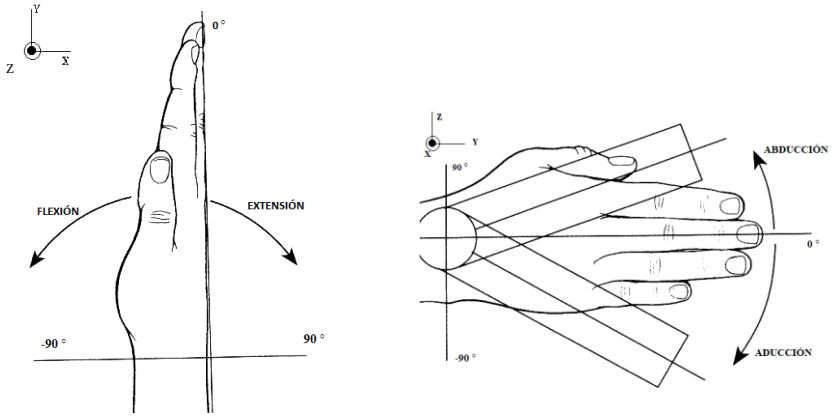


Figura 5.2 Visualización de los movimientos realizados [29]

Los músculos necesarios para realizar los movimientos seleccionados se muestran con unos recuadros en las imágenes siguientes, extraídas del Atlas de Anatomía Sobotta[17]:

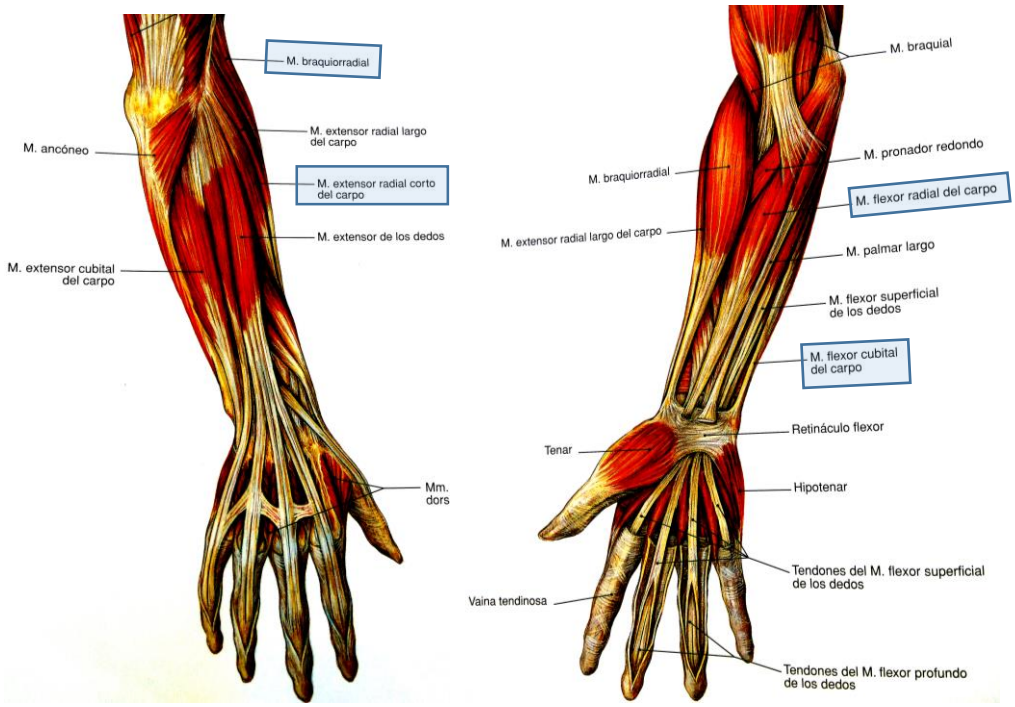


Figura 5.3 Músculos del antebrazo [30]

29 http://www.traumazaragoza.com/traumazaragoza.com/Documentacion_files/Biomeca%CC%81nica%20de%20la%20Mun%CC%83eca.pdf

30 [17]

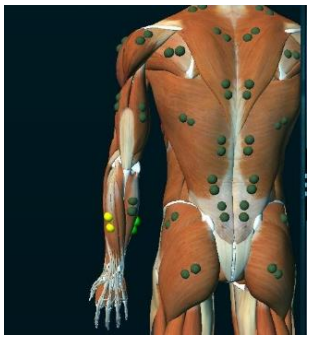
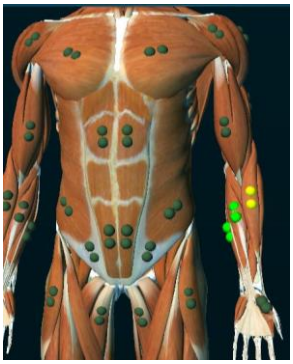
Músculos y movimientos	Localización de los pares de electrodos
El "Extensor Carpi Radialis o **Extensor Radial del Carpo**" es el músculo que presenta una mayor actividad al realizar el movimiento de "Extensión". En la figura de la derecha se muestra la colocación óptima del par de electrodos con dos puntos amarillos.	
El "Brachioradialis o **Braquiorradial**" realiza el movimiento de "Abducción o Desviación Radial". En este caso, el músculo también puede dar valores elevados al realizar una extensión.	
El músculo "Flexor Carpum Radialis o **Flexor Radial del Carpo**" alcanza sus valores máximos al realizar un movimiento de "Flexión".	
El "Flexor Carpum Ulnaris, **Flexor Cubital del Carpo** o Flexor Ulnar del Carpo" se encarga de realizar el movimiento de "Aducción". Al tener una distancia muy pequeña con el Flexor Radial del Carpo, también se registran valores elevados al hacer una flexión.	

Tabla 5.1 Posicionamiento de los pares de electrodos [31]

[31] Fuente propia

5.2 Adquisición de datos con pre-procesamiento

En los dos primeros subapartados se explican los pasos previos necesarios para la adecuada recogida de las señales a analizar en el apartado de procesamiento. Esos pasos son: la calibración del sensor giroscopio-acelerómetro, y el filtrado de las señales del sensor y electromiográficas para eliminar el ruido existente.

El material necesario para medir las señales EMG se muestran a continuación:

Figura 5.4 Instrumentación necesaria para la medición de señales EMG [32]

Elemento	Unidades
Pares de electrodos	4
Alcohol	1
Equipo Noraxon miniDTS	1
Etiquetas doble cara transmisor	4
Arduino UNO	1
Módulo GY-521	1
Ordenador	1

Tabla 5.2 Material necesario TFM [33]

5.2.1 Calibración GY-521

El primer paso para recoger los valores del sensor acelerómetro-giroscopio es la calibración, ya que necesita un primer valor para poder situarse en el espacio.

El código (realizado en el IDE de Arduino) funciona de la siguiente forma: primero resetea los offsets de otra medida anterior y luego recoge 8 medidas y hace la media aritmética. Con este método se consigue reducir las variaciones en el pulso que puede tener una persona. Ese código es el siguiente:

```
void calibracion()
{
  // Inicializa las variables
int j;
int suma_ax=0;
int suma_ay=0;
int suma_az=0;
```

[32] Fuente propia
[33] Fuente propia

```
int suma_gx=0;
int suma_gy=0;
int suma_gz=0;

  while (j<8){
      // coge una medida
      meansensors();
      // se divide entre 8 porque mean_ax>>3 es igual a mean_ax/(2^3)
      ax_offset=-mean_ax/8;
      ay_offset=-mean_ay/8;
      az_offset=(16384-mean_az)/8;
      gx_offset=-mean_gx/4;
      gy_offset=-mean_gy/4;
      gz_offset=-mean_gz/4;

      suma_ax=suma_ax+ax_offset;
      suma_ay=suma_ay+ay_offset;
      suma_az=suma_az+az_offset;
      suma_gx=suma_gx+gx_offset;
      suma_gy=suma_gy+gy_offset;
      suma_gz=suma_gz+gz_offset;
      j=j+1;
  }

  // Realiza la media aritmética
  ax_offset=suma_ax/8;
  ay_offset=suma_ay/8;
  az_offset=suma_az/8;
  gx_offset=suma_gx/8;
  gy_offset=suma_gy/8;
  gz_offset=suma_gz/8;

  // Copia los valores como valores iniciales dentro del dispositivo mpu
  mpu.setXAccelOffset(ax_offset);
  mpu.setYAccelOffset(ay_offset);
  mpu.setZAccelOffset(az_offset);
  mpu.setXGyroOffset(gx_offset);
  mpu.setYGyroOffset(gy_offset);
  mpu.setZGyroOffset(gz_offset);

  // Muestra los valores por el monitor
  Serial.print(ax_offset);
  Serial.print("\t");
  Serial.print(ay_offset);
  Serial.print("\t");
  Serial.print(az_offset);
  Serial.print("\t");
  Serial.print(gx_offset);
  Serial.print("\t");
  Serial.print(gy_offset);
  Serial.print("\t");
  Serial.println(gz_offset);

  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetGravity(&gravity, &q);
  mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

  // Se pasan los radianes a grados
  ypr_offset1=ypr[0]* 180 / M_PI;
 }
```

La función que se encarga de realizar cada medida es **meansensors**():

```
void meansensors(){
  long i=0,buff_ax=0,buff_ay=0,buff_az=0,buff_gx=0,buff_gy=0,buff_gz=0;
  while (i<(buffersize+101)){
    // lee los valores en crudo del acel/giro desde el sensor
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    if (i>100 && i<=(buffersize+100)){ //Las primeras 100 medidas se
desechan
      buff_ax=buff_ax+ax;
      buff_ay=buff_ay+ay;
      buff_az=buff_az+az;
      buff_gx=buff_gx+gx;
      buff_gy=buff_gy+gy;
      buff_gz=buff_gz+gz;
    }
    if (i==(buffersize+100)){
      mean_ax=buff_ax/buffersize;
      mean_ay=buff_ay/buffersize;
      mean_az=buff_az/buffersize;
      mean_gx=buff_gx/buffersize;
      mean_gy=buff_gy/buffersize;
      mean_gz=buff_gz/buffersize;
    }
    i++;
    delay(2); // Necesario para no tomar mediciones repetidas
  }
}
```

5.2.2 Filtrado de las señales

Consiste en la modificación de los datos para extraer la información deseada, amplificando una onda aumentando el valor de la ganancia o implementando los conjuntos de filtros necesarios para una mejor definición de la señal resultante. Al ser dos formas distintas de captación de datos, los filtros también varían:

SEÑALES EMG

Las señales obtenidas directamente de los electrodos son demasiado débiles (amplitud típica 0-6 mV), por lo que el programa Noraxon las amplifica. Un inconveniente de estas señales es su elevada presencia de ruido, por lo que hace falta eliminar las componentes de alta frecuencia y las debidas a la influencia de la corriente alterna de la red eléctrica (50 Hz). Por lo tanto, el rango típico de las señales EMGs es de 10 a 500 Hz, con un filtro de rechazo de banda o tipo “notch” en 50 Hz. He implementado un filtro digital de paso bajo (FPB) de 500 Hz a través del programa, en la configuración de la señal, tal como se muestra en la Figura 5.5:

Figura 5.5 Características dispositivo Mini DTS [34]

En la figura de arriba también se puede observar que se ha seleccionado una frecuencia de muestreo (Sample Rate) de 1500 Hz. Según el Teorema de Nyquist-Shannon, la frecuencia de muestreo debe ser al menos el doble de la componente frecuencial más alta que se pretende registrar.

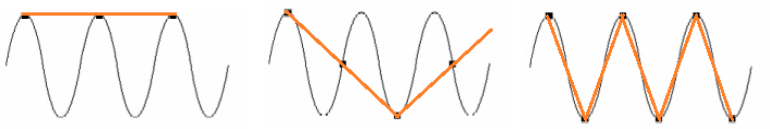


Figura 5.6 Variación en la frecuencia de muestreo [35]

En el ejemplo superior se muestra una onda con diferentes frecuencias de muestreo: con la frecuencia máxima de la señal de entrada, 4/3 de la frecuencia máxima, y el doble de dicha frecuencia. Como se puede observar, una onda pierde información si la frecuencia de muestreo es menor que el doble de la frecuencia máxima medida. También se puede producir "aliasing" o solapamiento, lo que supone una pérdida de información al producirse interferencias. Por tanto, para poder registrar componentes de frecuencia de 500 Hz, la frecuencia tiene que ser mayor que el doble (filtro antialiasing). En este caso ponemos 1500 Hz porque es la que está más próxima a 1000 Hz.

$$f_{muestreo} > 2 * f_{máxima\ entrada}$$

Se podría coger una frecuencia de muestreo de 3000 Hz porque cumple el teorema, pero al aumentar la frecuencia aumentan también la cantidad de puntos, por tanto, de datos. Esto requiere una mayor capacidad de computación y supone un incremento en el tiempo de procesamiento.

[34] Fuente propia
[35] Fuente propia

Una vez adquiridas las señales, se han implementado dos métodos de pre-procesado para reducir el proceso de cálculo: rectificación y suavizado (smoothing). Sus características se muestran en la Figura 5.7:

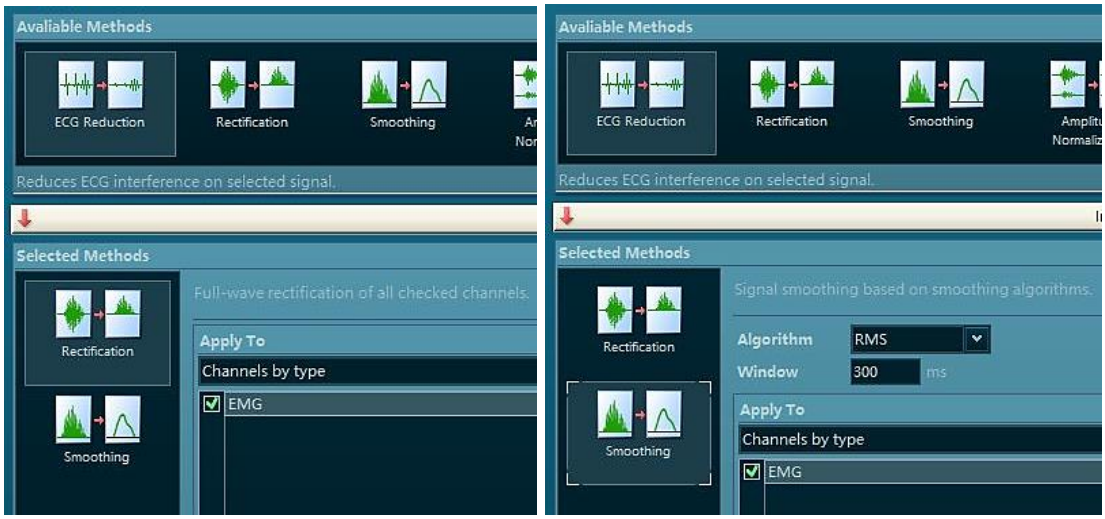


Figura 5.7 Pre-procesado de las señales EMGs [36]

La rectificación de onda completa es un proceso en el que los valores negativos pasan a ser valores positivos. Es decir, la onda resultante es el valor absoluto de la onda inicial sin eliminar ningún valor. A continuación, se muestra un ejemplo de una rectificación de onda completa.

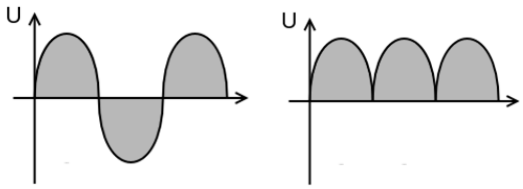


Figura 5.8 Rectificación de onda completa [37]

Con respecto al suavizado, he utilizado un pre-procesado de RMS con una ventana móvil de 300 ms. El método de las ventanas sirve para suavizar las discontinuidades que existen entre el principio y final de cada muestra. Las ventanas van variando de posición hasta recorrer toda la onda. Los valores finales son el resultado de aplicar el método RMS a cada ventana.

$$\mathrm{RMS} = \sqrt{\frac{1}{\mathrm{N}}\sum_{\mathrm{n}=1}^{\mathrm{N}} \mathrm{X_n}^2}$$

[36] Fuente propia
[37] https://es.wikipedia.org/wiki/Rectificador

Englehart et Al.[18] mencionaron en su artículo que el tiempo de respuesta máximo de un sistema de control mioeléctrico online debe ser inferior a 300 ms, con el fin de que el usuario no perciba una sensación de retraso.

MÓDULO GY-521

Es necesario realizar un filtrado de la señal a la vez que se produce la adquisición, ya que el sensor siempre presenta un error de medición acumulativo. El giroscopio tiene un offset (drift) debido a la integral que se utiliza para extraer el ángulo de giro a partir de la velocidad angular. Con respecto al acelerómetro, éste es muy sensible a los golpes y presenta ruido a alta frecuencia. Todo ello se resuelve con un filtro complementario [19], que es una simplificación del filtro de Kalman. La ecuación a seguir es la siguiente:

$$\theta = A * (\theta_{prev} + \theta_{giro}) + B * \theta_{acel}$$

El filtro complementario está implementado dentro de la función **almacenarValores**() del código de Arduino:

```
// muestra los ángulos de Euler en grados
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
double x=(ypr[0] * 180 / M_PI)*0.89+0.11*ypr_offset1;
double z=ypr[2] * 180 / M_PI;
valoresNuevos=String(x)+ ' ' + String(z);
```

Hay que tener en cuenta que los parámetros A y B siempre tienen que sumar 1. En este caso A=0.89 y B=0.11, por tanto, A+B=1. Se cumple la condición.

5.2.3 Recogida de datos

El primer paso antes de proceder a medir las señales EMG procedentes de los electrodos es la preparación de la piel para la colocación de estos últimos:

- Un día antes hay que evitar el uso de lociones o cremas en la zona del ensayo.
- Limpiar la piel con alcohol antiséptico (Alcohol 70-96º).
- En numerosos artículos se cita la necesidad de aplicar un gel que mejore la conductividad y adherencia, pero los electrodos utilizados ya presentan ese gel integrado.

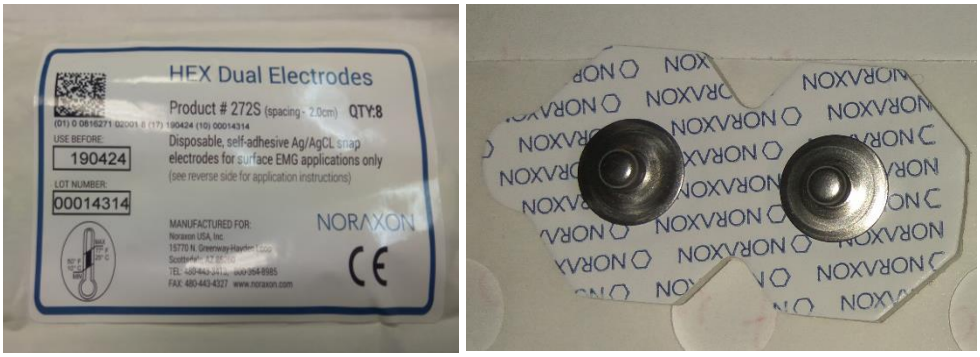


Figura 5.9 Electrodos Ag/AgCl [38]

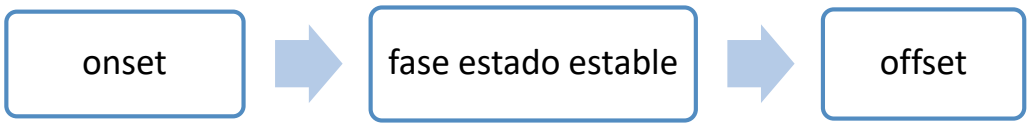
Para una correcta colocación de los electrodos es necesario comprobar en cada usuario los movimientos a realizar y así detectar mejor el vientre del músculo y su dirección de propagación. En este proyecto se van a emplear 8 electrodos, agrupados de 2 en 2, dando lugar a 4 señales electromiográficas. Se utiliza el método de electrodos diferenciales, explicado en el apartado 2. Estos electrodos tienen la propiedad de que, al estar muy próximos, las perturbaciones que afectan por igual a ambos electrodos se anulan [2]. Cuando el número de electrodos es mayor de 6 canales es necesario la necesidad de un experto, ya que aumenta la complejidad en su colocación [18]. En el caso de este proyecto, los músculos del antebrazo son muy delgados, por lo que pueden presentar zonas de inervación y crosstalk. La última es el conjunto de interferencias que presenta un músculo en la señal de un músculo vecino.

Hay que tener en cuenta que al aumentar el número de electrodos la precisión de la medida de la señal aumenta, pero también se incrementan las interferencias que se pueden dar entre los canales, y también el tiempo de respuesta [4]. El estudio se ha realizado con 5 usuarios diestros (4 varones y 1 mujer) de entre 20 y 30 años. Se les ha pedido que se sentasen en una silla con reposabrazos y mantuviesen la misma posición del antebrazo fija y relajada durante el experimento (Figura 5.10). Las muestras han sido recogidas cada 5 segundos, con un tiempo de descanso de 3 segundos entre una muestra y otra para evitar fatiga muscular. Ésta es debida a la contínua activación de las fibras musculares y supone cambios fisiológicos a nivel celular, tal como lo demuestra M.Rojas [20].

Figura 5.10 Ejemplo de colocación de los sensores en la adquisición de las señales [39]

[38] Fuente propia
[39] Fuente propia

A medida que aumenta el nivel de contracción, los músculos producen potenciales de acción mayores, ya que entran en funcionamiento nuevas fibras musculares. Cada contracción presenta las siguientes fases:



- **Onset**: inicio de la contracción. Se activan las primeras unidades motoras.

- **Fase estado estable:** en el instante en el que la señal llega a su máximo. Es la señal que se suele analizar, ya que al aumentar el nivel de fuerza se produce un aumento en la amplitud y de la frecuencia de disparo de las motoneuronas [3].

- **Offset:** fin de la contracción.

A continuación, para comprobar que la colocación de los electrodos es correcta, se procede a visualizar la actividad de los cuatro músculos (Extensor Radial del Carpo, Braquiorradial, Flexor Radial del Carpo Radial y Flexor Ulnar o Cubital del Carpo) realizando los 4 movimientos que se van a estudiar (Flexión, Extensión, Abducción y Aducción). Como se puede observar en la Figura 5.11 existen algunos instantes en los que, al realizar un movimiento, aparecen picos en dos o más músculos. Esto es debido al crosstalk entre músculos vecinos.

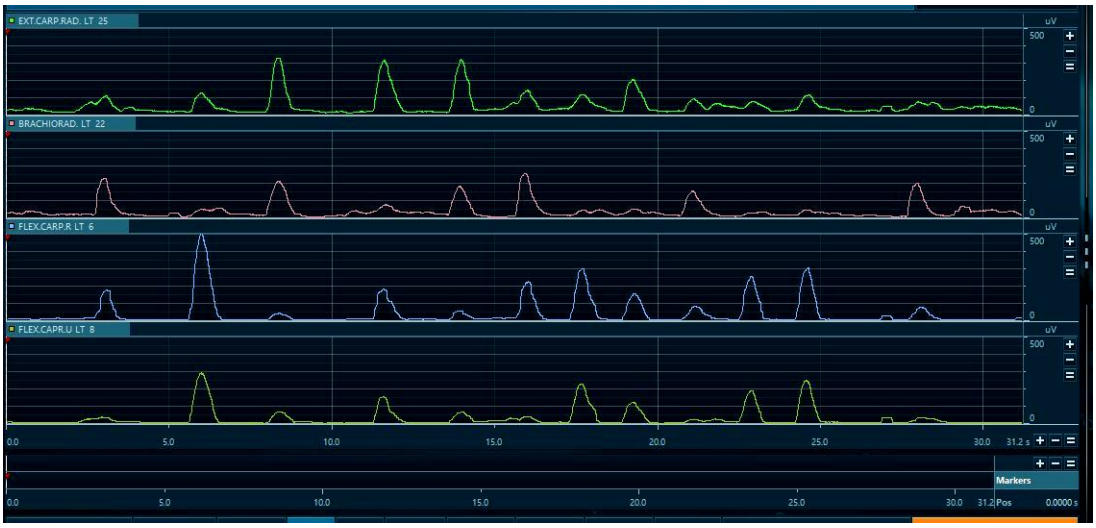


Figura 5.11 Ejemplo de las señales EMG en los músculos del estudio [40]

[40] Fuente propia

La recogida de datos de las señales EMG se hace a partir de MATLAB. Software multidisciplinar que combina en su totalidad los cálculos necesarios en el proyecto, la comunicación con Arduino y Noraxon, y el procesamiento de las señales.

El código de la adquisición de las señales EMG está localizado en el archivo "**noraxon_stream_collect.m**". Este archivo es una modificación del archivo original de Noraxon. En él recoge a su vez las señales mostradas por el Arduino en el mismo instante de tiempo que las señales EMG. El resultado es una estructura 1x6 con dos campos (info y samples o muestras), tal como se muestra a continuación:

Fields	info	samples
1	*1x1 struct*	*7500x1 dou...*
2	*1x1 struct*	*7500x1 dou...*
3	*1x1 struct*	*7500x1 dou...*
4	*1x1 struct*	*7500x1 dou...*
5	'EjeX'	*7499x1 dou...*
6	'EjeZ'	*7499x1 dou...*

Figura 5.12 Estructura general de un movimiento [41]

Los 4 primeros campos de la estructura general corresponden con los 4 canales EMG, mientras que los dos últimos campos pertenecen a los valores medidos por el sensor GY-521. Internamente, los datos se dividen en info y samples (Figura 5.12). La columna de datos engloba todos los datos medidos en los 5 segundos de las señales EMG y del sensor acelerómetro-giroscopio simultáneamente. Con respecto al campo de info, las señales EMG contienen información necesaria para su procesamiento futuro.

data(1).samples

	1
1	24.1200
2	24.1400
3	24.1300
4	24.2100
5	24.4200

name	'EXT.CARP.RAD. LT'
type	'emg'
full_type	'real.emg'
sample_rate	1500
units	'uV'
index	0

Figura 5.13 Muestras e información del campo 1 (Canal 1 de las señales EMG) [42]

```
% Función que devuelve un objeto que contiene toda la información y sus valores

function data = noraxon_stream_collect(stream_config,seconds, serial, posicion)
a = exist('webread');

if a == 0
    error('webread() missing. (Se necesita Matlab R2014b+.)');
end

if nargin < 1
```

[41] Fuente propia
[42] Fuente propia

```
    error('A stream_object (returned from noraxon_stream_init) is required');
end

if nargin < 2
    seconds = 10;
end

data = {};
info = stream_config.channelinfo;

% almacena las cabeceras de info y samples
for n=1:length(info)
   data(n).info = info(n);
   data(n).samples = [];
   % Determina el número de muestras necesario
   samples_remaining(n) = info(n).sample_rate * seconds;
end

last_data_timer = tic;
while (max(samples_remaining) > 0)

    try
        new_data = webread(strcat(stream_config.server_url,'/samples'),
              weboptions('ContentType','json'));
    catch
        new_data = [];
    end

    % Se envía el valor 3 por el puerto serie a Arduino para que comience a
    % medir
    fprintf(serial,'%s','3');
    valor=fscanf(serial)
    matrizValores=sscanf(valor,'%f')
    vector(:,posicion+1)=matrizValores
    posicion=posicion+1;

    if isstruct(new_data)
        % actualiza el último valor de tiempo
        last_data_timer = tic;

        % itera sobre los datos recibidos
        for n=1:length(new_data.channels)
            % coge el índice del canal actual
            index = new_data.channels(n).index;

            % encuentra el canal en el objeto data
            for i = 1:length(data)

                if data(i).info.index == index
                        % guarda los datos, basados en el tipo
                        if strcmp(data(i).info.full_type, 'real.') % cuaternios,
                          % necesarios para calcular los ángulos de Euler.

                            to_copy = min(length(new_data.channels(n).samples)/3,
                                samples_remaining(i));
                            samples = [];

                            for x = 1:to_copy
                               tmpx = new_data.channels(n).samples(((x-1)*3)+1);
                               tmpy = new_data.channels(n).samples(((x-1)*3)+2);
```

```
                                tmpz = new_data.channels(n).samples(((x-1)*3)+3);

                                samples(x).q0 = sqrt(max(0, 1 - ((tmpx * tmpx) +
                                       (tmpy * tmpy) + (tmpz * tmpz))));
                                samples(x).q1 = tmpx;
                                samples(x).q2 = tmpy;
                                samples(x).q3 = tmpz;

                            end
                            data(i).samples = [data(i).samples, samples];
                            samples_remaining(i) = samples_remaining(i) - to_copy;

                        elseif ~isempty(strfind(data(i).info.full_type,'vector3'))

                            to_copy = min(length(new_data.channels(n).samples)/3,
                                      samples_remaining(i));
                            samples = [];
                            for x = 1:to_copy
                                samples(x).x = new_data.channels(n).
                                             samples(((x-1)*3)+1);
                                samples(x).y = new_data.channels(n).
                                             samples(((x-1)*3)+2);
                                samples(x).z = new_data.channels(n).
                                             samples(((x-1)*3)+3);
                            end
                            data(i).samples = [data(i).samples, samples];
                            samples_remaining(i) = samples_remaining(i) - to_copy;

                        else % canal analógico

                            to_copy = min(length(new_data.channels(n).samples),
                                samples_remaining(i));
                            data(i).samples = [data(i).samples;
                                new_data.channels(n).samples(1:to_copy)];
                            samples_remaining(i) = samples_remaining(i) - to_copy;

                        end
                    break
                end
            end

        end


    else
        % webread error
        if toc(last_data_timer) > 5
            error('MR3 stream timeout.  Check to be sure HTTP Streaming is enabled
              and a measurement is currently running.');
            break;
        end

    end

end
% se envía el valor 5 para que Arduino deje de almacenar valores
fprintf(serial,'%s','5');
valor=fscanf(serial)
% seguir almacenando hasta recibir una 'f' por el puerto serie.
while(valor~='f')
```

```
    matrizValores=sscanf(valor,'%f')
    vector(:,posicion+1)=matrizValores;
    posicion=posicion+1;
    valor=fscanf(serial)
end

%% Ponemos el vector con las mismas dimensiones que los vectores EMG
% con una interpolación
[a,b]=size(vector);
xq = 1:b/7521:b;
for i=1:a
    valoresGYInterpolacion(i,:) = interp1(vector(i,:),xq);
end

%% Ponemos los valores de giro dados por el giroscopio en la estructura
% de los datos EMG
data(5)=struct('info','EjeX','samples',valoresGYInterpolacion(1,:)');
data(6)=struct('info','EjeZ','samples',valoresGYInterpolacion(2,:)');
```

Módulo GY-521

El Arduino recibe los valores en crudo (raw) y, a partir de la librería MPU6050 [21] englobada dentro de la librería "MPU6050_6Axis_MotionApps20.h" implementada en el código, se extraen los valores de los ángulos de Euler, del vector de gravedad y de los cuaternios. Hay que tener en cuenta que la sensibilidad del giroscopio y del acelerómetro depende del rango configurado. En este proyecto se ha elegido el rango seleccionado en la siguiente tabla:

Rango De Escala Completa Giroscopio	Sensibilidad del Giroscopio	Rango De Escala Completa Acelerómetro	Sensibilidad del Acelerómetro
±250	131	±2	16384
±500	65.5	±4	8192
±1000	32.8	±8	4096
±2000	16.4	±16	2048

Tabla 5.3 Rangos de escala acelerómetro-giroscopio [43]

La resolución del circuito integrado es de 16 bits, por lo que el rango de valores es:

$$2^{16} - 1 = 65535 \rightarrow {}^{65535}/_{2} \rightarrow \boxed{\text{Rango de medición} = [-32768, 32767]}$$

[43] https://hetpro-store.com/TUTORIALES/modulo-acelerometro-y-giroscopio-mpu6050-i2c-twi/

Un aumento en la velocidad de la comunicación serial (baudios) permite generar más datos, pero imprimir en pantalla es más lento, por lo que se puede llegar a saturar el buffer de memoria. Se ha agrandado el tamaño del buffer a 1000 para poder poner una velocidad de 115200 baudios. El buffer se va vaciando cada vez que el programa de Matlab lee los datos del sensor, cuando se reinicia el sistema o cuando se inicializa al calibrar.

El código completo de la adquisición y calibración de señales de Arduino está en el archivo "**GY-521_Cristina_Antelo.ino**", y se muestra a continuación:

```
// Basada en la librería I2Cdev y en el trabajo realizado por Jeff Rowberg
//jeff@rowberg.net con la librería MPU6050

// Librerías necesarias
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"

// ================================================================
// ===                      CONFIGURACION                       ===
// ================================================================
int buffersize=1000;

// Creación de un objeto mpu de la clase MPU6050
MPU6050 mpu;

// Yaw/pitch/roll ángulos (en grados) calculados a partir de los
//cuaternios del FIFO. También necesita cálculos del vector de gravedad.
#define OUTPUT_READABLE_YAWPITCHROLL

// MPU control
bool dmpReady = false; // pone en true si la inicializión del DMP ha sido
correcta
uint8_t mpuIntStatus; // mantiene el byte de la interrupción de estados
del MPU
uint8_t devStatus; // devuelve el estado después de cada operación (0 =
correcto, !0 = error)
uint16_t packetSize; // tamaño esperado del paquete DMP (por defecto es 42
bytes)
uint16_t fifoCount; // suma de todos los bytes actuales en el FIFO
uint8_t fifoBuffer[64]; // buffer de almacenamiento del FIFO

Quaternion q;           // [w, x, y, z]         vector de cuaternios
VectorFloat gravity;    // [x, y, z]            vector de gravedad
float euler[3];         // [psi, theta, phi]    vector de ángulos de Euler
float ypr[3];           // [yaw, pitch, roll]   vector yaw/pitch/roll y de
gravedad

int ax, ay, az,valorMonitor,state,dt,tiempo_prev;
int gx, gy, gz;
int mean_ax,mean_ay,mean_az,mean_gx,mean_gy,mean_gz;
int k;
String valoresNuevos;
String valoresString;
float girosc_ang_x, girosc_ang_y,girosc_ang_z;
int ax_offset, ay_offset, az_offset, gx_offset, gy_offset, gz_offset;
```

```
float ypr_offset1;

// ================================================================
// ===                      SETUP                               ===
// ================================================================
void setup() {
 // comienza I2C bus (la librería I2Cdev no lo hace automáticamente)
  Wire.begin();
  TWBR = 24; // 400kHz reloj I2C (200kHz si CPU es 8MHz).

  // inicializa la comunicación serial
  Serial.begin(115200);

  // inicializa el dispositivo
  mpu.initialize();
  while (Serial.available() && Serial.read()); // buffer vacío de nuevo

  devStatus = mpu.dmpInitialize();
  mpu.setDMPEnabled(true);
  packetSize = mpu.dmpGetFIFOPacketSize();

  // resetea offsets
  mpu.setXAccelOffset(0);
  mpu.setYAccelOffset(0);
  mpu.setZAccelOffset(0);
  mpu.setXGyroOffset(0);
  mpu.setYGyroOffset(0);
  mpu.setZGyroOffset(0);

  k=0;
}

// ================================================================
// ===                          LOOP                            ===
// ================================================================
void loop() {
    // Para obtener el número recibido por el Serial Monitor, se resta el
    // valor '0', ya que el dato recibido es un valor en código ASCII
    valorMonitor=Serial.read();
    state=valorMonitor-'0';

    // Para saber si queremos inicializar los valores o no
  switch(state){
    case 0:
    // reset offsets
    mpu.setXAccelOffset(0);
    mpu.setYAccelOffset(0);
    mpu.setZAccelOffset(0);
    mpu.setXGyroOffset(0);
    mpu.setYGyroOffset(0);
    mpu.setZGyroOffset(0);
      // Falta obtener la aceleración inicial
      calibracion();

      break;

    case 1: // Recoger valores rangos
        mpu.resetFIFO();
        updateGiro();
        delay(1000);
```

```
      break;

    case 3: // Recoger valores entrenamiento

        while(1){
          mpu.resetFIFO();
          valoresString=almacenarValores();
          Serial.println(valoresString);
          if (Serial.available()> 0){
            state=Serial.read()-'0';
          }
          if (state==5){
            valoresString.remove(0);
            Serial.println('f');
            break;
          }
        }

    default:
      break;
  }
  delay(5);
}


// ================================================================
// ===                      FUNCIONES                           ===
// ================================================================
void updateGiro()
{
 fifoCount = mpu.getFIFOCount();
    if (fifoCount == 2048){
      Serial.println("FIFO overflow");
    }else{
      while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
      // leer un paquete de FIFO
      mpu.getFIFOBytes(fifoBuffer, packetSize);
      fifoCount -= packetSize;
      #ifdef OUTPUT_READABLE_YAWPITCHROLL
        // muestra los ángulos de Euler en grados
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
        Serial.print(1);
        Serial.print(" ");
        Serial.print(((ypr[0] * 180 / M_PI))*0.89+0.11*ypr_offset1);
        Serial.print(" ");
        Serial.print(ypr[1] * 180 / M_PI);
        Serial.print(" ");
        Serial.println(ypr[2] * 180 / M_PI);
      #endif
    }
}

String almacenarValores()
{
 fifoCount = mpu.getFIFOCount();
    if (fifoCount == 2048){
      Serial.println("FIFO overflow");
    }else{
```

```
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
        mpu.getFIFOBytes(fifoBuffer, packetSize);
        fifoCount -= packetSize;

        #ifdef OUTPUT_READABLE_YAWPITCHROLL
          // muestra los ángulos de Euler en grados
          mpu.dmpGetQuaternion(&q, fifoBuffer);
          mpu.dmpGetGravity(&gravity, &q);
          mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
          double x=(ypr[0] * 180 / M_PI)*0.89+0.11*ypr_offset1;
          double z=ypr[2] * 180 / M_PI;
          valoresNuevos=String(x)+ ' ' + String(z);
          return valoresNuevos;
        #endif
      }
      delay(20);
}

void calibracion()
{
  // Inicializa las variables
int j;
int suma_ax=0;
int suma_ay=0;
int suma_az=0;
int suma_gx=0;
int suma_gy=0;
int suma_gz=0;

while (j<8){
  // coge una medida
  meansensors();
  ax_offset=-mean_ax/8;
  ay_offset=-mean_ay/8;
  az_offset=(16384-mean_az)/8;
  gx_offset=-mean_gx/4;
  gy_offset=-mean_gy/4;
  gz_offset=-mean_gz/4;

  suma_ax=suma_ax+ax_offset;
  suma_ay=suma_ay+ay_offset;
  suma_az=suma_az+az_offset;
  suma_gx=suma_gx+gx_offset;
  suma_gy=suma_gy+gy_offset;
  suma_gz=suma_gz+gz_offset;
  j=j+1;
  }

  // Realiza la media aritmética
  ax_offset=suma_ax/8;
  ay_offset=suma_ay/8;
  az_offset=suma_az/8;
  gx_offset=suma_gx/8;
  gy_offset=suma_gy/8;
  gz_offset=suma_gz/8;

  // Copia los valores como valores iniciales dentro del dispositivo mpu
  mpu.setXAccelOffset(ax_offset);
  mpu.setYAccelOffset(ay_offset);
  mpu.setZAccelOffset(az_offset);
```

```
  mpu.setXGyroOffset(gx_offset);
  mpu.setYGyroOffset(gy_offset);
  mpu.setZGyroOffset(gz_offset);

  // Muestra los valores por el monitor
  Serial.print(ax_offset);
  Serial.print("\t");
  Serial.print(ay_offset);
  Serial.print("\t");
  Serial.print(az_offset);
  Serial.print("\t");
  Serial.print(gx_offset);
  Serial.print("\t");
  Serial.print(gy_offset);
  Serial.print("\t");
  Serial.println(gz_offset);

  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetGravity(&gravity, &q);
  mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
  ypr_offset1=ypr[0]* 180 / M_PI;
 }

 void meansensors(){
  long i=0,buff_ax=0,buff_ay=0,buff_az=0,buff_gx=0,buff_gy=0,buff_gz=0;
  while (i<(buffersize+101)){
    // lee los valores en crudo del acel/giro desde el sensor
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    if (i>100 && i<=(buffersize+100)){ //Las primeras 100 medidas se
desechan
      buff_ax=buff_ax+ax;
      buff_ay=buff_ay+ay;
      buff_az=buff_az+az;
      buff_gx=buff_gx+gx;
      buff_gy=buff_gy+gy;
      buff_gz=buff_gz+gz;
    }
    if (i==(buffersize+100)){
      mean_ax=buff_ax/buffersize;
      mean_ay=buff_ay/buffersize;
      mean_az=buff_az/buffersize;
      mean_gx=buff_gx/buffersize;
      mean_gy=buff_gy/buffersize;
      mean_gz=buff_gz/buffersize;
    }
    i++;
    delay(2); // Necesario para no medir mediciones repetidas
  }
}
```

A continuación, se muestra un ejemplo de visualización de las señales EMG y del sensor GY-521 al realizar un movimiento de aducción:

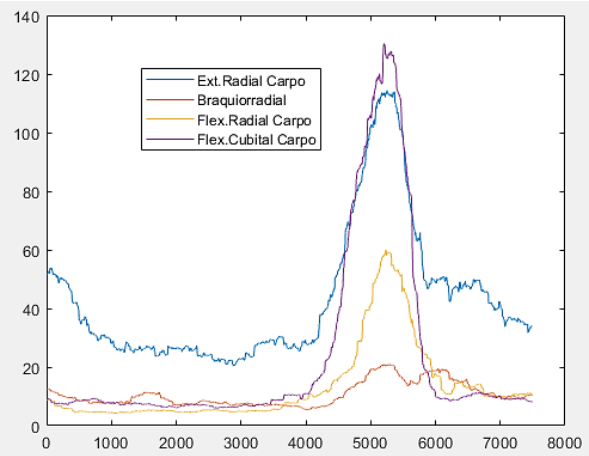


Figura 5.14 Mediciones de las señales EMG al realizar un movimiento de aducción [44]

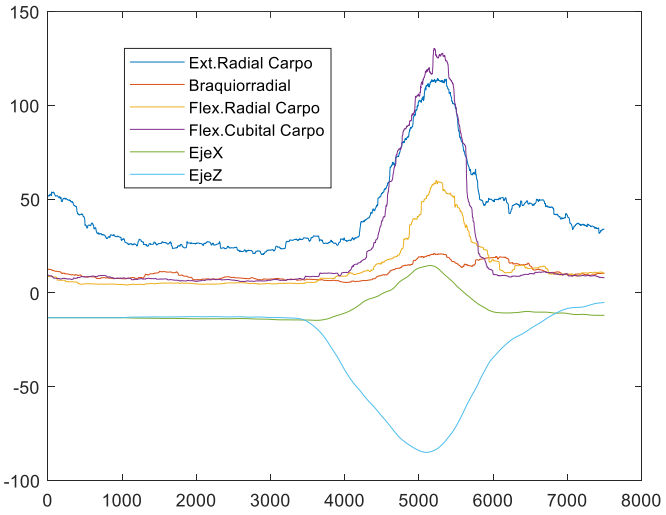


Figura 5.15 Mediciones de las señales EMG y del sensor GY-521 al realizar un movimiento de aducción [45]

En el apartado de resultados se muestran las gráficas correspondientes a los cuatro movimientos de cada individuo. Además, se explican las similitudes y diferencias entre cada una.

[44] Fuente propia
[45] Fuente propia.

5.3 Procesamiento

Una vez obtenidos los datos, se procede a su estudio para llegar a un fin determinado. En este proyecto se quiere detectar la trayectoria que sigue el usuario a partir únicamente de las señales electromiográficas. Para el análisis de las señales neuronales existen dos áreas dependiendo de la respuesta que se desea obtener. Por un lado, están los métodos de clasificación, cuya respuesta es binaria, es decir, una salida discreta. Y, por el otro lado están los métodos de regresión, que tienen una respuesta continua.

Algunos de los métodos de control de las señales electromiográficas son:

- Control proporcional: la velocidad varía en función del nivel de contracción de las señales EMG, debido a que la tensión aplicada presenta un rango de valores.
- Control ON/OFF: se determina la activación/desactivación de los motores mediante la comparación entre la amplitud generada a partir del método RMS (Raíz Cuadrática Media) o del método MAV (Valor Absoluto Medio) con un umbral establecido previamente. Inconvenientes: sólo admite 2 DoF (Grados de Libertad) y presenta siempre una velocidad constante, independientemente de la amplitud de la onda.
- Control de máquina de estados finitos: se definen previamente unos estados y unas transiciones. Este método es adecuado si hay un número fijo de posturas.
- Control basado en reconocimiento de patrones: extracción de características de la señal y clasificación dependiendo de su comportamiento. Se extraen principalmente muestras en el tiempo y en la frecuencia. Últimamente se han utilizado Redes Neuronales (NN) para este método. La ventaja es que aprenden características lineales y no lineales.
- Esquemas de control de regresión: en este método se engloban el control proporcional y el simultáneo. El objetivo es obtener señales de control que se transformen en ángulos de las articulaciones.

A partir de esa distinción, en este estudio se emplea la regresión, un método de predicción basado en el aprendizaje automático: el clasificador de regresión lineal. Este clasificador se basa en los datos extraídos del acelerómetro-giroscopio para calcular unos coeficientes y crear un modelo para luego calcular unos puntos X y Z a partir de otras señales EMG.

5.3.1 Introducción

Los métodos de regresión dependen del método de generación de datos y de la forma en la que se relacionan las variables. Existen dos grupos principales: regresión lineal y no lineal. Dentro de la lineal tenemos regresión lineal simple y regresión lineal múltiple.

En la regresión no lineal, la más común es la regresión segmentada o regresión por pedazos. En ella se divide en grupos o particiones las variables independientes ajustando los valores mediante una curva de datos. Dentro de cada segmento las relaciones entre las variables dependientes e independientes vienen dadas por una regresión lineal. En la unión de dichos segmentos se puede producir una desviación del origen o un cambio brusco en la pendiente (puntos de quiebra).

El primer método de regresión utilizado fue el denominado "Método de mínimos cuadrados". En él se pretende minimizar la suma de las distancias en el eje de ordenadas entre las respuestas calculadas y las reales [22].

5.3.2 Regresión lineal

Una regresión lineal o ajuste lineal es una técnica de modelización estadística que relaciona una o varias variables independientes (X) dando lugar a una recta de aproximación (Y_t), calculada mediante unos coeficientes calculados a partir de valores reales. Una recta de regresión divide una nube de puntos en dos partes iguales. Las técnicas de regresión lineal tienen su campo de aplicación principalmente en la economía, la biología y la ingeniería.

Existen varios tipos de regresión lineal: simple, múltiple y multivariante. La diferencia entre las dos primeras es la cantidad de parámetros independientes: en la simple sólo se maneja una variable independiente, por lo que sólo se calculan dos coeficientes β_0 y β_1. En una regresión lineal múltiple o regresión múltiple existen n variables independientes y un parámetro β_0 que es el término constante en la recta, es decir, el parámetro independiente de una recta. La ecuación de una recta múltiple es la siguiente:

$$Y_t = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

La ecuación de la regresión lineal tiene la siguiente forma:

$$Y_t = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_n X_n + \hat{\varepsilon}$$

En este tipo de regresiones múltiples, el número de coeficientes $\hat{\beta}$ es de *n+1* y son los calculados a partir de los valores de X y de Y, y se denominan coeficientes de correlación. El parámetro $\hat{\varepsilon}$ es el error entre los valores calculados y los reales, contenidos en un vector de n elementos. La notación matricial de la regresión lineal múltiple es:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Y la fórmula para la obtención de los coeficientes $\hat{\beta}$:

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

En el artículo de Scott H. Brown [23], se muestra un modo de realizar el análisis de regresión lineal multiplicando directamente las matrices con la fórmula de arriba.

La línea resultante se denomina recta de regresión, y es siempre una línea recta, pero su pendiente no tiene por qué ser siempre positiva. Su forma dependerá de la distribución seguida por la nube de puntos que se quiere analizar. En la figura de abajo se muestran distintos tipos de líneas creadas a partir de una regresión lineal en la que la pendiente puede ser positiva (b y c), negativa (a) o nula (d):

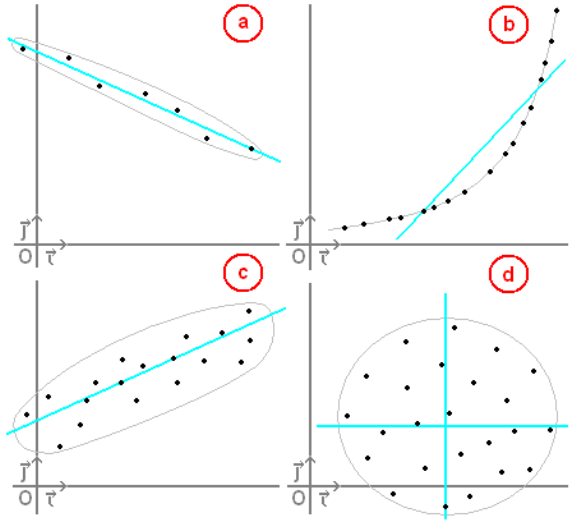


Figura 5.16 Ejemplos de regresión lineal [46]

Para que una regresión lineal genere correctamente los coeficientes de correlación tiene que cumplir los siguientes requisitos [24]:

- La relación entre las variables tiene que ser lineal.
- Los errores en la medición de las variables ($\hat{\varepsilon}$) tienen que ser independientes entre sí.
- Los errores tienen que tener varianza constante (Homocedasticidad).
- Los errores tienen que tener una esperanza matemática igual a 0.
- El error total tiene que ser la suma de todos los errores.

[46] http://enciclopedia.us.es/index.php/Regresi%C3%B3n_lineal

5.3.3 Clasificador de la regresión lineal

El clasificador se realiza a partir de la técnica de **validación cruzada** o **k-fold cross validation**. En ella se divide un conjunto en varias particiones, dependiendo del número de iteraciones a realizar. El grupo de entrenamiento tiene que ser mucho mayor que el de testeo para que los resultados sean favorables. Los pasos seguidos son:

1º) Una partición se selecciona como test o prueba y el resto como entrenamiento. Un dato a tener en cuenta es que, al realizar el clasificador, los datos del entrenamiento no pasan a ser test, ya que habría una sobreclasificación y el resultado sería erróneo.

2º) Se determinan los coeficientes de correlación a partir de los datos de entrenamiento con la función **regress()** de Matlab.

3º) Se calculan los valores de X e Y con los valores de los coeficientes y los datos de test como datos de entrada.

4º) Se procede al análisis de los resultados (apartado 5.1)

5º) El bucle vuelve a empezar, pero esta vez la partición de testeo se mueve una casilla a la derecha, de forma que quede la misma cantidad de datos de test y de entrenamiento que la iteración anterior, pero con valores diferentes.

6º) Una vez que se ha realizado la última iteración, el programa concluye y se extraen nuevos resultados totales, calculando la correlación media y la desviación estándar de la correlación.

En la figura siguiente se muestra un ejemplo de la variación de posición del grupo de datos de prueba. Como puede observarse, el número de datos a procesar no varía, lo único que se modifican son los valores con los que se entrena y se testea.

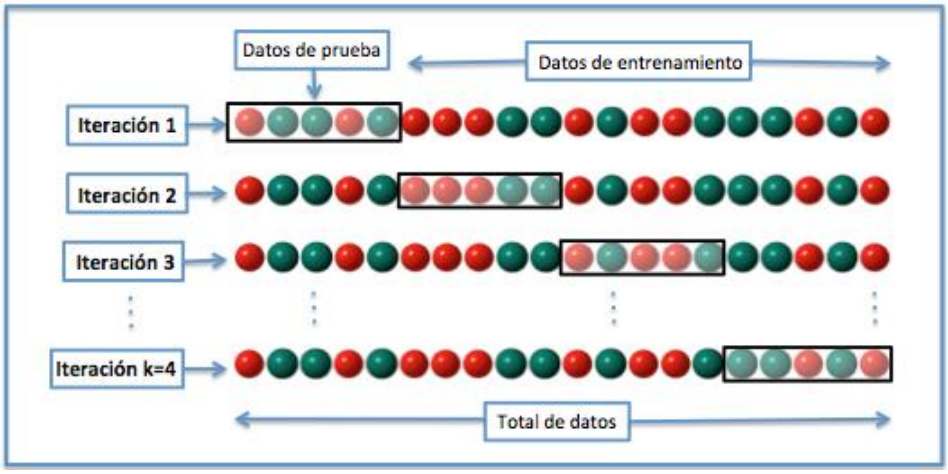


Figura 5.17 Ejemplo de validación cruzada [47]

[47] https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada

Debido a su complejidad, se han realizado dos regresiones lineales múltiples, una para calcular los valores del Eje X, y la otra para los valores del Eje Z.

$$EjeX = \beta_0 + \beta_1 EMG_1 + \beta_2 EMG_2 + \beta_3 EMG_3 + \beta_4 EMG_4$$

$$EjeZ = \beta_0 + \beta_1 EMG_1 + \beta_2 EMG_2 + \beta_3 EMG_3 + \beta_4 EMG_4$$

Para realizar el cálculo en Matlab ha sido necesario implementar la **"Curve Fitting Toolbox"**. A continuación, se muestra el código explicado de la función **regresion()**, localizada en el archivo **regresion.m**

```
%% Seleccionamos los valores que queremos clasificar:
valores=get_file_names;

%% Características clasificador:
format long
iteraciones=5;

n_datos=valores.n_files;
porc_test=iteraciones/n_datos;
file_numbers=zeros(n_datos,6);
trayectoria={n_datos};

%% Pre-procesamiento de las señales
for i=1:n_datos
    trayectoria(i)=struct2cell(load(string(valores.filename(i))));

    % Creación de la matriz
    for canales=1:6
        matrizCanalesTotales(canales,i)={trayectoria{1,i}(canales).samples()};
    end

    % Ponemos las longitudes de los datos de EMG y del sensor iguales
    % (pre-procesamiento)
    while length(matrizCanalesTotales{1,i})~= length(matrizCanalesTotales{5,i})
        matrizCanalesTotales{1,i}(end)=[];
        matrizCanalesTotales{2,i}(end)=[];
        matrizCanalesTotales{3,i}(end)=[];
        matrizCanalesTotales{4,i}(end)=[];
    end
end

%% Variables procesamiento de la señal
% iteracion=fix(1/porc_test);
tam_test=fix(1/porc_test);
accuracy1=zeros(iteraciones,1);

%% Validación cruzada
for turns=1:iteraciones

    %% Recorrido de los datos para test segun el K-folds
    data_test=matrizCanalesTotales(:,1+(turns-1)*tam_test:tam_test+(turns-
1)*tam_test);
    data_train=matrizCanalesTotales;
    data_train(:,1+(turns-1)*tam_test:tam_test+(turns-1)*tam_test)=[];

    %% Pasamos todos los datos de test y entrenamiento a una matriz de
n_datos(40) x canales(6)
```

```
    for i=1:size(data_test,2)
        if i==1
            datos_test = [data_test{:,1}];
        else
            datos_test = [datos_test ; data_test{:,i}];
        end
    end
    for i=1:size(data_train,2)
        if i==1
            datos_train = [data_train{:,1}];
        else
            datos_train = [datos_train ; data_train{:,i}];
        end
    end

    %% Separamos los valores de las señales EMG y las del sensor
    puntosEMG=datos_train(:,1:4);
    puntosSensor=datos_train(:,5:6);
    puntosEMG_test=datos_test(:,1:4);
    puntosSensor_test=datos_test(:,5:6);
    puntosSensor(:)=smooth(puntosSensor(:),200,'lowess');
    puntosEMG(:)=smooth(puntosEMG(:),200,'lowess');
    puntosSensor_test(:)=smooth(puntosSensor_test(:),200,'lowess');
    puntosEMG_test(:)=smooth(puntosEMG_test(:),200,'lowess');

    % Se añade una columna a la derecha de puntos EMG para el valor
    % independiente
    puntosEMG=[puntosEMG ones(size(puntosEMG,1),1)];

    nombre_metodo='Regresión Lineal Múltiple';
    %% Funciones para la regresión lineal
    [beta_x]=regress(puntosSensor(:,1),puntosEMG);
    [beta_y]=regress(puntosSensor(:,2),puntosEMG);

    % Calculamos los puntos del sensor a partir de las señales EMG y de los
    % coeficientes determinados a partir de la regresión

puntosSensor_test_calc(:,1)=puntosEMG_test(:,1)*beta_x(1)+puntosEMG_test(:,2)*b
eta_x(2)...
        +puntosEMG_test(:,3)*beta_x(3)+puntosEMG_test(:,4)*beta_x(4)+beta_x(5);

puntosSensor_test_calc(:,2)=puntosEMG_test(:,1)*beta_y(1)+puntosEMG_test(:,2)*b
eta_y(2)...
        +puntosEMG_test(:,3)*beta_y(3)+puntosEMG_test(:,4)*beta_y(4)+beta_y(5);

    %% Graficamos los resultados
    figure(3*turns-2)
    plot(puntosSensor_test(:,1)); % Datos originales
    hold on
    plot(puntosSensor_test_calc(:,1),'r'); % Recta de regresión
    title(sprintf('Iteración %d eje X', turns));
    hold off

    figure(3*turns-1)
    plot(puntosSensor_test(:,2)); % Datos originales
    hold on
    plot(puntosSensor_test_calc(:,2),'r'); % Recta de regresión
    title(sprintf('Iteración %d eje Z', turns));
    hold off
```

```
    % Trayectorias:
    figure(3*turns)
    plot(puntosSensor_test(:,1),puntosSensor_test(:,2))
    hold on
    plot(puntosSensor_test_calc(:,1),puntosSensor_test_calc(:,2),'r')
    hold off
    %% Calculamos la correlación existente entre los datos extraídos del sensor
y los calculados (apartado 5.1)
 …

 clear puntosSensor_test_calc
 clear puntosEMG_test
 …
end

%% Calculamos la media y la desviación estándar de la correlación
corr_x_media=mean(corr_x_med);
corr_y_media=mean(corr_y_med);
corr_x_std=std(corr_x_med);
corr_y_std=std(corr_y_med);
```

El resultado de aplicar la función de arriba son 2 gráficas por iteración o vuelta, de cada eje que relacionan los valores de X y de Z calculados con los reales. La línea en azul corresponde a los valores reales del sensor, mientras que la roja son los valores calculados a partir de los coeficientes de la regresión lineal. Los valores se analizarán en el Apartado 7 en profundidad.

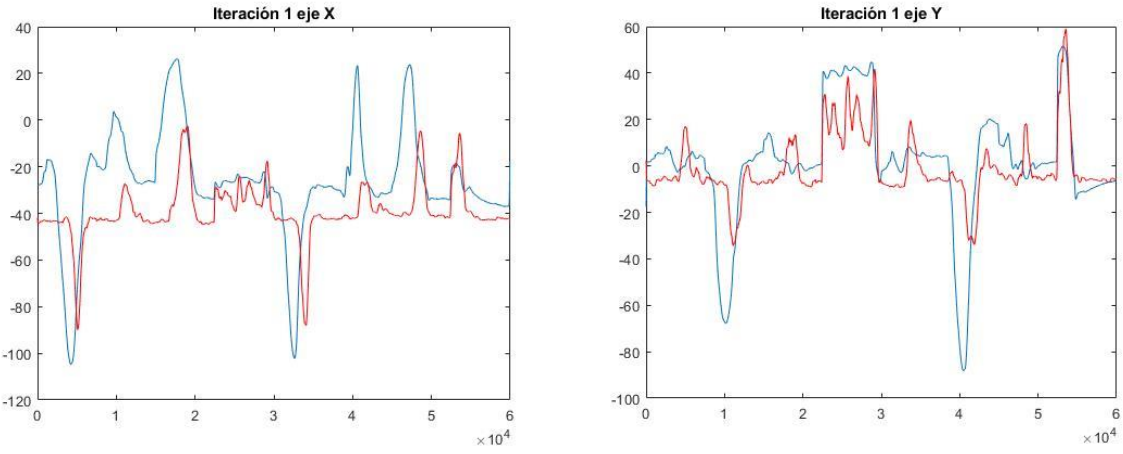


Figura 5.18 Resultado de la regresión entre los valores reales y los calculados [48]

Se ha probado también a hacer una regresión lineal solamente en un eje, para que las señales electromiográficas correspondientes con los otros movimientos no interfieran en el resultado final. De esta manera se pretendía determinar la influencia entre los músculos que no correspondían a un movimiento determinado.

[48] Fuente propia

El código es muy similar al anterior, sólo varía el número de coeficientes de correlación, ya que ahora la ecuación sería:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \end{bmatrix} \cdot \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Por ejemplo, si quisiésemos observar los movimientos abducción/aducción, los canales de EMG serían 2 y 4, por tanto, la ecuación del eje Z quedaría:

$$EjeZ = \beta_0 + \beta_1 EMG_2 + \beta_2 EMG_4$$

Este código se encuentra en el archivo **regresion_2channels.m**. Lo único que lo diferencia del código anterior es el tamaño de la matriz puntosEMG y el apartado de la regresión lineal:

```
%% Separamos los valores de las señales EMG y las del sensor
    puntosEMG=[datos_train(:,2) datos_train(:,4)];
    puntosSensor=datos_train(:,6);
    puntosEMG_test=[datos_test(:,2) datos_test(:,4)];
    puntosSensor_test=datos_test(:,6);
    puntosSensor(:)=smooth(puntosSensor(:),200,'lowess');
    puntosEMG(:)=smooth(puntosEMG(:),200,'lowess');
    puntosSensor_test(:)=smooth(puntosSensor_test(:),200,'lowess');
    puntosEMG_test(:)=smooth(puntosEMG_test(:),200,'lowess');

    % Se añade una columna a la derecha de puntos EMG para el valor
    % independiente
    puntosEMG=[puntosEMG ones(size(puntosEMG,1),1)];

    nombre_metodo='Regresión Lineal Múltiple';
    %% Funciones para la regresión lineal
    %beta_x]=regress(puntosSensor(:,1),puntosEMG);
    [beta_y]=regress(puntosSensor(:),puntosEMG);

    % Calculamos los puntos del sensor a partir de las señales EMG y de
    % los coeficientes determinados a partir de la regresión

puntosSensor_test_calc(:,1)=puntosEMG_test(:,1)*beta_y(1)+puntosEMG_test(:
,2)*beta_y(2)...
        +beta_y(3);
    %% Graficamos los resultados
    figure(2*turns-1)
    plot(puntosSensor_test); % Datos originales
    hold on
    plot(puntosSensor_test_calc,'r'); % Recta de regresión
    title(sprintf('Iteración %d eje Z', turns));
    hold off
```

5.4 Métodos de análisis de la regresión lineal

Para determinar la robustez del algoritmo de regresión lineal, se ha optado primeramente por el cálculo de la correlación lineal existente entre la recta real y la calculada a partir de los coeficientes de la regresión lineal múltiple extrayendo el valor de la mediana de la función **normxcorr2** de Matlab. Dicho método del cálculo de error se ha llevado a cabo porque, a partir de la visualización de unos resultados en unas gráficas, se podía percibir un ligero desplazamiento horizontal entre la gráfica de los valores calculados y los reales. Además, es el método más común para calcular la efectividad en señales neuronales ya que tienen poca influencia las señales de ruido. Si las variaciones entre una señal y otra son debidas al ruido, la correlación consigue eliminar esa variación, logrando una diferencia entre ellas del casi 100% [25].

En el código de **regresion()**:

```
…
 corr_x=normxcorr2(puntosSensor_test(:,1),puntosSensor_test_calc(:,1));
 corr_y=normxcorr2(puntosSensor_test(:,2),puntosSensor_test_calc(:,2));
 corr_x_med(turns,1)=median(corr_x);
 corr_y_med(turns,1)=median(corr_y);
…
 clear corr_x
 clear corr_y
end
```

Al presentar un coeficiente de correlación tan reducido, se ha optado por implementar otros métodos de predicción de errores que se suelen utilizar en el análisis de las señales neuronales, seleccionados a partir del estudio realizado por M. Spuler et al. [25]. Siendo y la trayectoria real, e $\hat{y}$ la trayectoria calculada:

NRMSE (Raíz Normalizada del Error Cuadrático Medio):

$$NRMSE(y,\hat{y}) = \frac{\sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_t - y_t)^2}{n}}}{(y_{max} - y_{min})}$$

SNR (Relación Señal-Ruido):

$$\mathrm{SNR}(y,\hat{y}) = \frac{\mathrm{var}(y - \hat{y})}{\mathrm{var}(\hat{y})}$$

COD (Coeficiente de Determinación):

$$COD(y,\hat{y}) = \frac{\sum_{i=1}^{n}(\hat{y}_t - y_t)^2}{\sum_{i=1}^{n}(\hat{y}_t - mean(\hat{y}))^2}$$

GD (Desviación Global):

$$GD(y,\hat{y}) = \left(\frac{\sum_{i=1}^{n}(\hat{y}_t - y_t)^2}{n}\right)^2$$

Se crea una función que englobe todos los métodos dentro de una misma estructura. Un segmento del código está extraído de la Toolbox de Matlab PSNR MSE R RMSE NRMSE MAPE Calculating, de Abbas Manthiri. Además de esas funciones, se han calculado también SNR, COD, GD y sus combinaciones. La función se denomina **calculoErrores( )**:

```
function Resultado = calculoErrores(real,test)
% Result-struct
% 1.NRMSE (Normalized Root-mean-square deviation)
% 2.CC (Correlation Coeficient)
% 3.SNR (Signal-noise ratio)
% 4.COD (Coefficient of determination)
% 5.GD (Global deviation)
% 6.CC-NRMSE
% 7.CC/NRMSE
% 8.CC+SNR

%% geting size and condition checking
[row_R,col_R,dim_R]=size(real);
[row_T,col_T,dim_T]=size(test);
if row_R~=row_T || col_R~=col_T || dim_R~=dim_T
    error('Input must have same dimentions')
end
%% Common function for matrix
% Mean for Matrix
meanmat=@(a)(mean(mean(a)));
% Sum for Matrix
summat=@(a)(sum(sum(a)));
% Min  for Matrix
minmat=@(a)(min(min(a)));
% Max  for Matix
maxmat=@(a)(max(max(a)));
%% RMSE Root-mean-square deviation
RMSE=abs( sqrt( meanmat((test-real).^2) ) );
%% Normalized RMSE Normalized Root-mean-square deviation
Resultado.NRMSE=RMSE/(maxmat(real)-minmat(real));
%% Método_1 Coeficientes de correlación (CC):
    %Calculamos la correlación existente entre los datos extraídos del
sensor y los calculados
```

```
    corr_calc=normxcorr2(real,test);
    corr_med=median(corr_calc);
    Resultado.CC=corr_med;

 %% Método_3 Relación señal-ruido (SNR)
    Resultado.SNR=(var(real - test))/var(test);

 %% Método_4 Coeficiente de determinación (COD)
    Resultado.COD= (summat((test-real).^2))/ (summat((test-
mean(real)).^2));

  %% Método_5 Desviación global (GD)
    Resultado.GD=((summat(test-real))/length(real))^2;
  %% Combinaciones
    Resultado.CC_Menos_NRMSE = Resultado.CC - Resultado.NRMSE;
    Resultado.CC_Entre_NRMSE = Resultado.CC / Resultado.NRMSE;
    Resultado.CC_Mas_SNR = Resultado.CC + Resultado.SNR;
end
```

En la Figura 5.19 se puede observar la estructura de la variable *Resultado_error_x* que está compuesta por los métodos de análisis explicados anteriormente. En la imagen de la derecha se muestran los valores que han tomado algunos de los métodos, que están almacenados dentro de esa estructura. Los formatos son los mismos para los valores del eje *x* y del eje *z*.

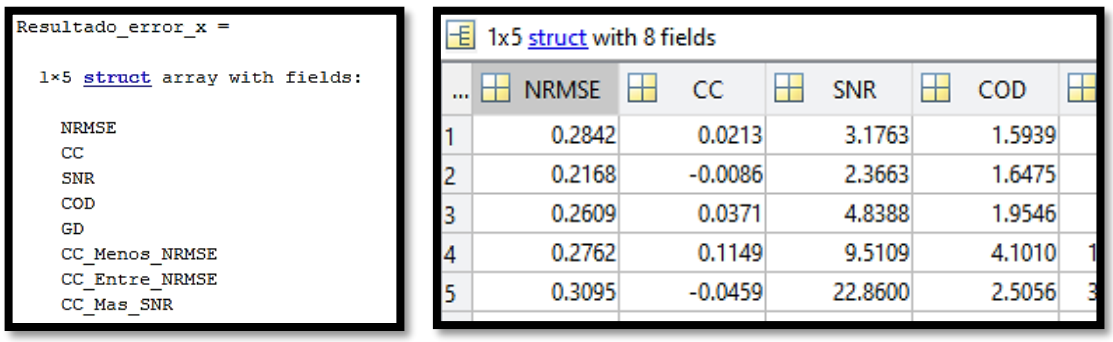


Figura 5.19 Esquema métodos y ejemplo de valores en 5 iteraciones [49]

Por último, se realiza una media para poder determinar el comportamiento general de los métodos en cada individuo. Esta media se realiza en el programa principal. El resultado completo se muestra en una tabla, siendo el último valor de cada método la media.

```
%% Media de cada error:
Resultado_error_x(turns+1).NRMSE=mean([Resultado_error_x.NRMSE]);
Resultado_error_x(turns+1).CC=mean([Resultado_error_x.CC]);
Resultado_error_x(turns+1).SNR=mean([Resultado_error_x.SNR]);
Resultado_error_x(turns+1).COD=mean([Resultado_error_x.COD]);
Resultado_error_x(turns+1).GD=mean([Resultado_error_x.GD]);
```

[49] Fuente propia

```
Resultado_error_x(turns+1).CC_Menos_NRMSE=mean([Resultado_error_x.CC_Menos
_NRMSE]);
Resultado_error_x(turns+1).CC_Entre_NRMSE=mean([Resultado_error_x.CC_Entre
_NRMSE]);
Resultado_error_x(turns+1).CC_Mas_SNR=mean([Resultado_error_x.CC_Mas_SNR])
;

Resultado_error_z(turns+1).NRMSE=mean([Resultado_error_z.NRMSE]);
Resultado_error_z(turns+1).CC=mean([Resultado_error_z.CC]);
Resultado_error_z(turns+1).SNR=mean([Resultado_error_z.SNR]);
Resultado_error_z(turns+1).COD=mean([Resultado_error_z.COD]);
Resultado_error_z(turns+1).GD=mean([Resultado_error_z.GD]);
Resultado_error_z(turns+1).CC_Menos_NRMSE=mean([Resultado_error_z.CC_Menos
_NRMSE]);
Resultado_error_z(turns+1).CC_Entre_NRMSE=mean([Resultado_error_z.CC_Entre
_NRMSE]);
Resultado_error_z(turns+1).CC_Mas_SNR=mean([Resultado_error_z.CC_Mas_SNR])
;

%% Tabla de resultados
Resultado_error_x_tabla=struct2table(Resultado_error_x)
Resultado_error_z_tabla=struct2table(Resultado_error_z)
```

La estructura final de la variable Resultado_error_x se muestra a continuación, junto con un ejemplo de algunos métodos. A diferencia de la anterior figura se puede observar que se ha añadido una fila. Ésta es la media de cada método. De esta forma se puede conocer con mejor precisión la variación entre un método y otro y se pueden evaluar mejor las diferencias que existentes entre un individuo y otro.

```
Resultado_error_x =

  1×6 struct array with fields:

    NRMSE
    CC
    SNR
    COD
    GD
    CC_Menos_NRMSE
    CC_Entre_NRMSE
    CC_Mas_SNR
```

1x6 struct with 8 fields

...	NRMSE	CC	SNR	COD	
1	0.2842	0.0213	3.1763	1.5939	
2	0.2168	-0.0086	2.3663	1.6475	
3	0.2609	0.0371	4.8388	1.9546	
4	0.2762	0.1149	9.5109	4.1010	1
5	0.3095	-0.0459	22.8600	2.5056	3
6	0.2695	0.0238	8.5505	2.3605	1

Figura 5.20 Estructura completa del estudio del error y ejemplos de valores [50]

[50] Fuente propia

6 SISTEMA DE INTERACCIÓN HOMBRE-MÁQUINA (HMI)

Este proyecto está gestionado por medio de una interfaz gráfica (guide) diseñada en Matlab. Las principales tareas de la interfaz son:

- Conexión/desconexión del Arduino o del equipo Noraxon miniDTS.
- Calibración del sensor GY-521.
- Llamada a la función de adquisición de datos.
- Aplicación del clasificador de regresión lineal en modo offline.
- Gestión de la simultaneidad de las medidas entre los dos tipos de señales.

6.1 Introducción

La interfaz está compuesta por dos interfaces: la ventana principal y la ventana de calibración y entrenamiento. En la primera interfaz es en la que se pone en marcha la aplicación y se testea el clasificador de regresión lineal seleccionando la opción de modo offline. La segunda interfaz es la que inicia las conexiones entre el Arduino Uno y el software Noraxon con el ordenador. También realiza la fase de calibración y de captación simultánea de datos y su posterior almacenamiento en archivos .mat. Cada archivo presenta un único movimiento.

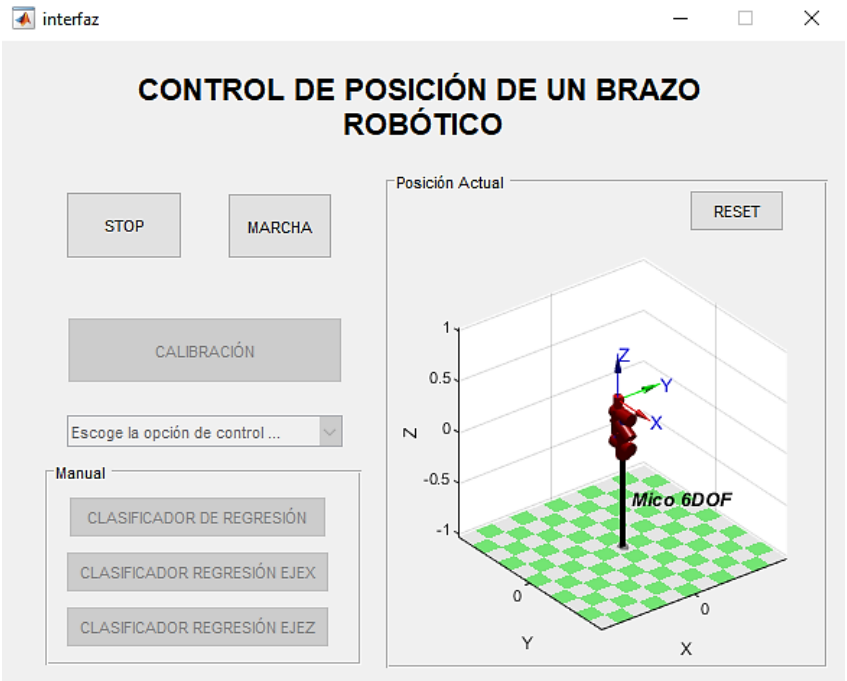


Figura 6.1 Interfaz principal sin pulsar ningún botón [51]

[51] Fuente propia

Figura 6.2 Interfaz de calibración [52]

En la figura de abajo se muestra un esquema del procedimiento seguido en las dos ventanas (interfaz e interfaz_calibración):

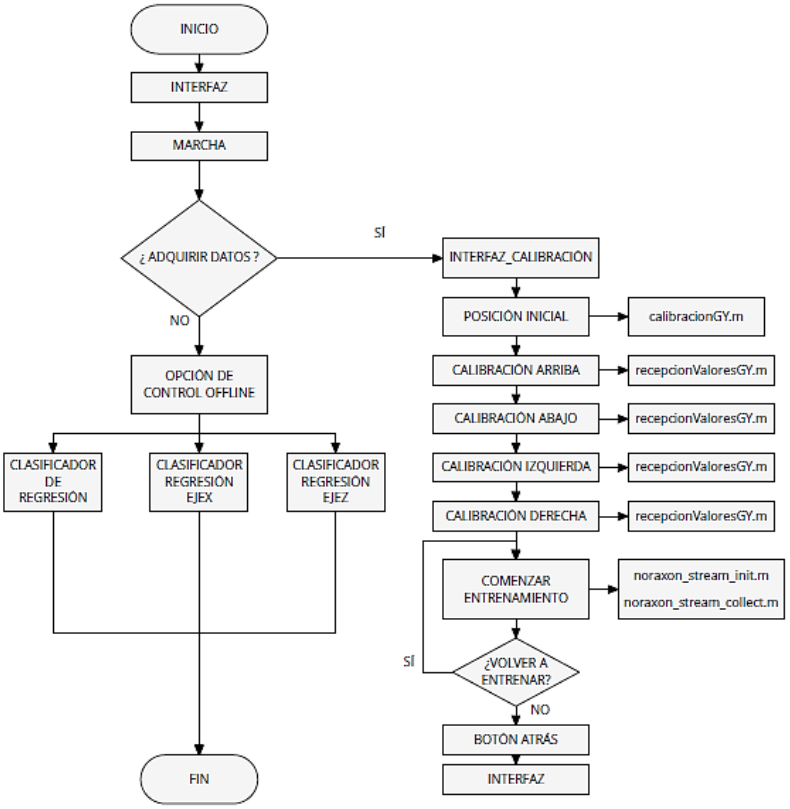


Figura 6.3 Esquema procedimiento interfaz [53]

[52] Fuente propia
[53] Fuente propia

6.2 Modo de calibración y entrenamiento

6.2.1 Procedimiento

Para entrar en la otra interfaz hay que darle primero al botón MARCHA. El robot simulado en la aplicación cambia de posición yendo a su punto HOME. En cuanto llega a esa posición se activa el botón de Calibración para poder pulsarlo.

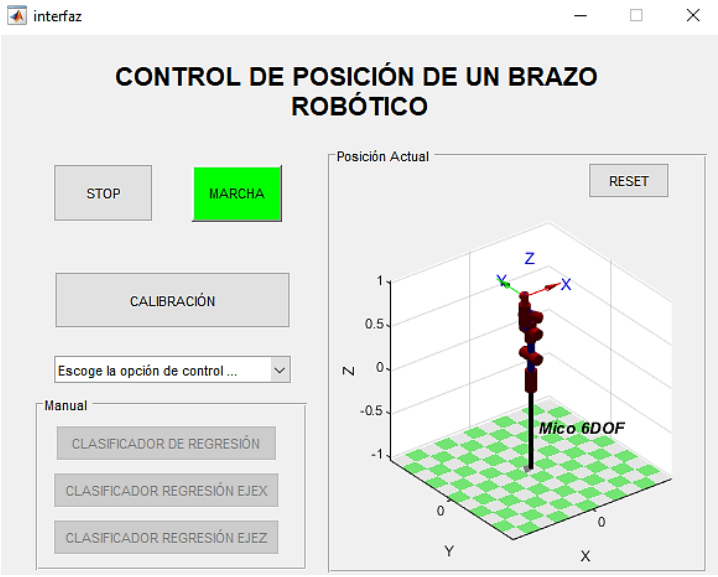


Figura 6.4 Interfaz principal con MARCHA pulsado [54]

Al accionarlo nos abre la otra interfaz, en la que engloba la calibración y la recogida de datos.

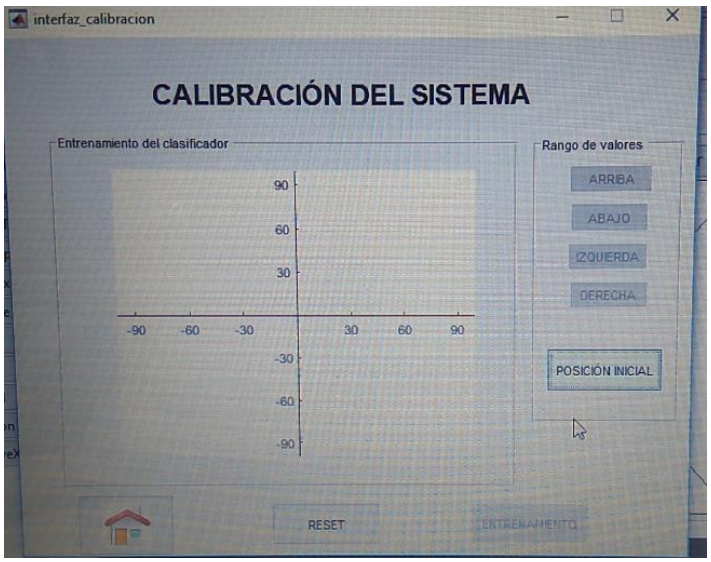


Figura 6.5 Paso 1 interfaz calibración [55]

[54] Fuente propia
[55] Fuente propia

Para proceder a la calibración del sensor GY-521 se pulsa el botón POSICIÓN INICIAL, que llamará a la función calibracionGY.m. Mientras el proceso de calibración esté en marcha aparecerá el mensaje de la Figura 6.6 para verificar que se están recogiendo los 8 valores.

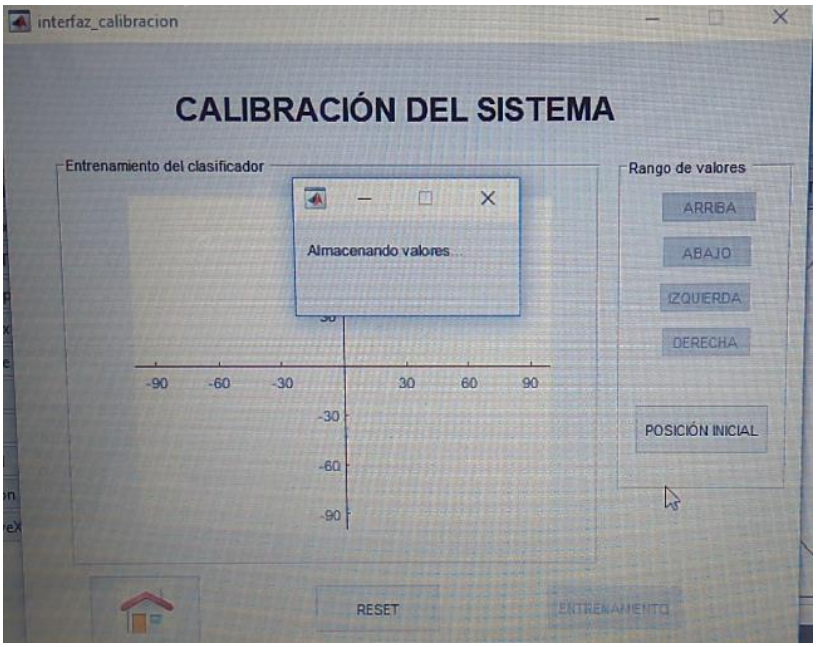


Figura 6.6 Paso 2 interfaz calibración [56]

Una vez realizada la media de los 8 valores necesarios para determinar una magnitud con el menor ruido posible, se muestra el mensaje "La posición inicial se ha calibrado correctamente".

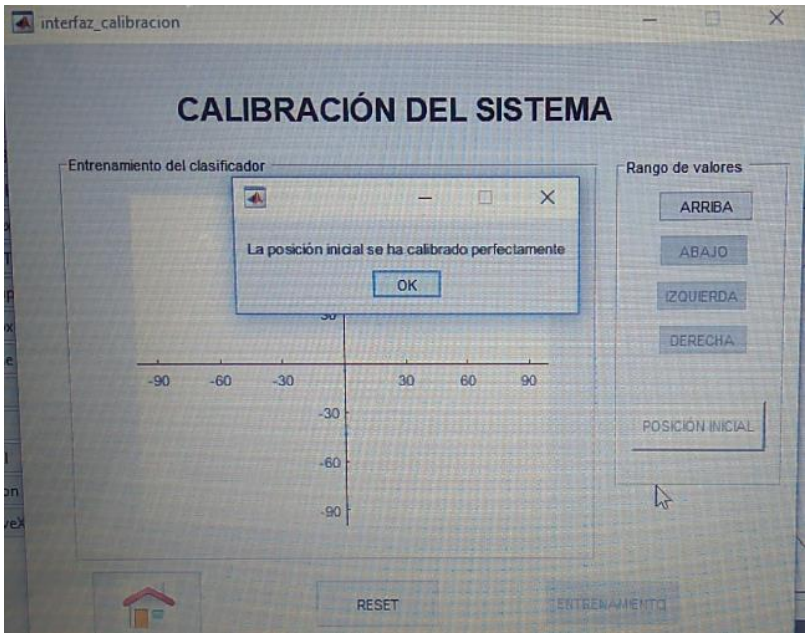


Figura 6.7 Paso 3 interfaz calibración [57]

[56] Fuente propia
[57] Fuente propia

A partir de ese momento es muy importante no modificar la posición del antebrazo o del cuerpo para que no varíe la posición de reposo. El siguiente paso consiste en adecuar la gráfica con los valores máximos del usuario. Por tanto se selecciona uno de los movimientos a realizar (Arriba, Abajo, Izquierda, Derecha) y una vez en esa posición, se le da al botón correspondiente. Una vez capturado el valor máximo aparecerá en la gráfica como un punto negro. Este paso hay que realizarlo para los 4 movimientos.

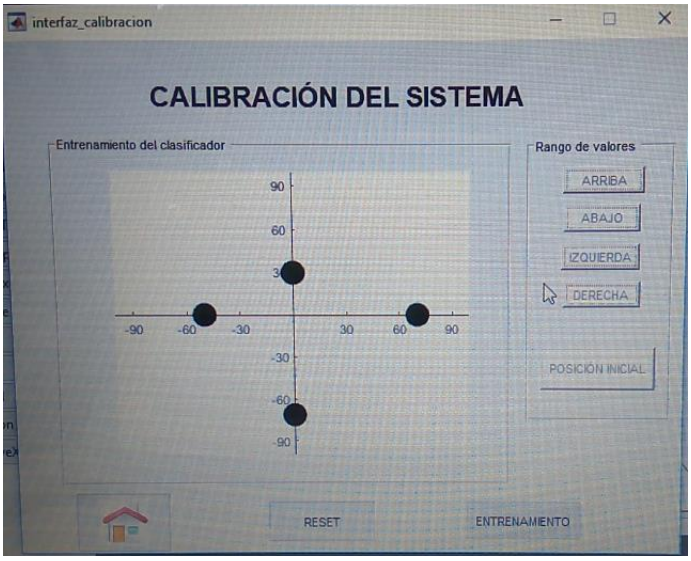


Figura 6.8 Paso 4 interfaz calibración [58]

Una vez obtenidos los máximos valores a los que puede llegar el usuario, la gráfica se redimensiona con esos valores, dándole un rango de ±10º. La adquisición de datos comienza al pulsar el botón de ENTRENAMIENTO. Existen dos colores de puntos: rojos (datos por recoger) y verdes (datos recogidos). El punto rojo va indicándole al usuario el movimiento que tiene que realizar.

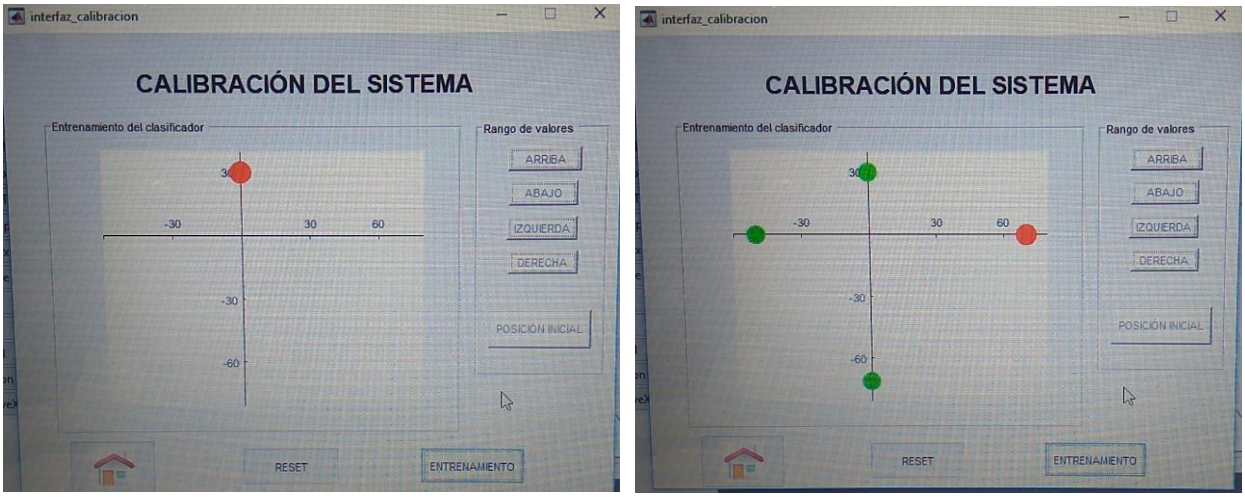


Figura 6.9 Paso 5 interfaz calibración. Obtención de los valores [59]

[58] Fuente propia
[59] Fuente propia

Una vez realizada la adquisición de los 4 movimientos se pregunta al usuario si quiere entrenar de nuevo. De ser así la gráfica volvería a ponerse en blanco y aparecería un punto rojo en la posición de abducción. Si el usuario no quiere seguir entrenando se selecciona No y no sucede nada hasta que no le dé a ATRÁS para volver a la pantalla principal.

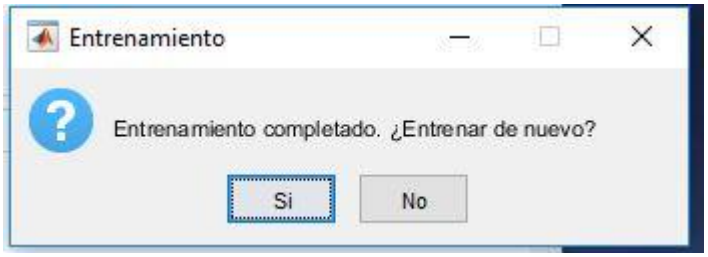


Figura 6.10 Paso 6 interfaz calibración [60]

A continuación, se muestra el código implementado:

```
function varargout = interfaz_calibracion(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @interfaz_calibracion_OpeningFcn, ...
                   'gui_OutputFcn',  @interfaz_calibracion_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before interfaz_calibracion is made visible.
function interfaz_calibracion_OpeningFcn(hObject, eventdata, handles,
varargin)
%Colocación de la interfaz en el centro de la pantalla
scrsz = get(0,'ScreenSize');
pos_act = get(gcf,'Position');
xr = scrsz(3) - pos_act(3);
xp = round(xr/2);
yr = scrsz(4) - pos_act(4);
yp = round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

%Poner imagen de ir a para atrás
[a,map]=imread('home.png');
[r,c,d]=size(a);
x=ceil(r/63);
```

[60] Fuente propia

```
y=ceil(c/63);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.atras,'CData',g);

global zMax; global zMin; global xMax; global xMin;
zMax=100; zMin=-100; xMax=100; xMin=-100;

set(handles.calibracionArriba,'enable','off');
set(handles.calibracionAbajo,'enable','off');
set(handles.calibracionIzquierda,'enable','off');
set(handles.calibracionDerecha,'enable','off');
set(handles.comenzarEntrenamiento,'enable','off');
handles.marcha = varargin{2};
handles.comunicacionArduino = varargin{1};
% Choose default command line output for interfaz_calibracion
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = interfaz_calibracion_OutputFcn(hObject, eventdata,
handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

function botonPosicionInicial_Callback(hObject, eventdata, handles)
valoresPosicionInicial = calibracionGY(handles.comunicacionArduino)
while isempty(valoresPosicionInicial)==1
    valoresPosicionInicial = calibracionGY(handles.comunicacionArduino)
end
set(gcbo,'Backgroundcolor','[0.97 0.97 0.97]');
set(gcbo,'enable','off');
set(handles.calibracionArriba,'enable','on');


function calibracionArriba_Callback(hObject, eventdata, handles)
global zMax;
datosArduino=recepcionValoresGY(handles.comunicacionArduino)
zMax = datosArduino(4)
%Dibujamos el punto obtenido
axes(handles.figuraEjes);
plot(0,zMax,'ko','linewidth',10);
set(gcbo,'Backgroundcolor','[0.97 0.97 0.97]');
set(gcbo,'enable','off');
set(handles.calibracionAbajo,'enable','on');


function calibracionAbajo_Callback(hObject, eventdata, handles)
global zMin;
datosArduino=recepcionValoresGY(handles.comunicacionArduino)
zMin=datosArduino(4)
%Dibujamos el punto obtenido
axes(handles.figuraEjes);
plot(0,zMin,'ko','linewidth',10);
set(gcbo,'Backgroundcolor','[0.97 0.97 0.97]');
set(gcbo,'enable','off');
set(handles.calibracionIzquierda,'enable','on');
```

```
function calibracionIzquierda_Callback(hObject, eventdata, handles)
global xMin;
datosArduino=recepcionValoresGY(handles.comunicacionArduino)
xMin=datosArduino(2)
%Dibujamos el punto obtenido
axes(handles.figuraEjes);
plot(xMin,0,'ko','linewidth',10);
set(gcbo,'Backgroundcolor','[0.97 0.97 0.97]');
set(gcbo,'enable','off');
set(handles.calibracionDerecha,'enable','on');


function calibracionDerecha_Callback(hObject, eventdata, handles)
global xMax;
datosArduino=recepcionValoresGY(handles.comunicacionArduino)
xMax=datosArduino(2)
%Dibujamos el punto obtenido
axes(handles.figuraEjes);
plot(xMax,0,'ko','linewidth',10);
set(gcbo,'Backgroundcolor','[0.97 0.97 0.97]');
set(gcbo,'enable','off');
set(handles.comenzarEntrenamiento,'enable','on');


function botonReset_Callback(hObject, eventdata, handles)
opcion=questdlg('¿Está seguro de reiniciar los
valores?','Reinicio','Sí','No','Sí');
global xMin; global xMax; global zMin; global zMax;
switch opción

    case 'Sí'
        punto=0;
        fclose(handles.comunicacionArduino);

        set(handles.calibracionIzquierda,'enable','off');
        set(handles.calibracionDerecha,'enable','off');
        set(handles.calibracionArriba,'enable','off');
        set(handles.calibracionAbajo,'enable','off');
        set(handles.botonPosicionInicial,'enable','on');
        set(handles.calibracionIzquierda,'Backgroundcolor','[0.94 0.94
0.94]');
        set(handles.calibracionDerecha,'Backgroundcolor','[0.94 0.94
0.94]');
        set(handles.calibracionArriba,'Backgroundcolor','[0.94 0.94
0.94]');
        set(handles.calibracionAbajo,'Backgroundcolor','[0.94 0.94
0.94]');
        set(handles.botonPosicionInicial,'Backgroundcolor','[0.94 0.94
0.94]');
        % Borrar datos gráfica
        cla
        xMin=-100; xMax=100; zMin=-100; zMax=100;
        axes(handles.figuraEjes);
        axis([xMin xMax zMin zMax]);
        pause(2);

    case 'No'
        return;
end
```

```
% Volvemos a la pantalla principal (Home).
function atras_Callback(hObject, eventdata, handles)
set(handles.marcha,'value',1);
interfaz(handles.marcha);

function figuraEjes_CreateFcn(hObject, eventdata, handles)
global zMax; global zMin; global xMax; global xMin;
axis([xMin xMax zMin zMax]);
axis on;
hold on;


function comenzarEntrenamiento_Callback(hObject, eventdata, handles)
stream_config = noraxon_stream_init('127.0.0.1', 9220);
punto=0;
global zMax; global zMin; global xMax; global xMin;
cla
axis([xMin-10 xMax+10 zMin-10 zMax+10]);
entrenamiento=1;
% Secuencia para ir midiendo cada punto
while(punto<4)
    switch punto

        case 0
            plot(0,zMax,'ro','linewidth',10);
            pause(1);
            i=0;
            data = noraxon_stream_collect(stream_config,
5,handles.comunicacionArduino,i);
            fname =
sprintf('./Adquisicion_Señales/Individuo/Entrenamiento/%d_Punto.mat',
entrenamiento);
            save(fname,'data');
            entrenamiento=entrenamiento+1;
            punto=1;


        case 1
            cla
            axis([xMin-10 xMax+10 zMin-10 zMax+10]);
            plot(0,zMax,'go','linewidth',8);
            plot(xMin,0,'ro','linewidth',10);
            pause(1);
            i=0;
            data = noraxon_stream_collect(stream_config,
5,handles.comunicacionArduino,i);
            fname =
sprintf('./Adquisicion_Señales/Individuo/Entrenamiento/%d_Punto.mat',
entrenamiento);
            save(fname,'data');
            entrenamiento=entrenamiento+1;
            punto=2;


        case 2
            cla
            axis([xMin-10 xMax+10 zMin-10 zMax+10]);
            plot(0,zMax,'go','linewidth',8);
            plot(xMin,0,'go','linewidth',8);
            plot(0,zMin,'ro','linewidth',10);
```

```
            pause(1);
            i=0;
            data = noraxon_stream_collect(stream_config,
5,handles.comunicacionArduino,i);
            fname =
sprintf('./Adquisicion_Señales/Individuo/Entrenamiento/%d_Punto.mat',
entrenamiento);
            save(fname,'data');
            entrenamiento=entrenamiento+1;
            punto=3;


        case 3
            axes(handles.figuraEjes);
            cla
            axis([xMin-10 xMax+10 zMin-10 zMax+10]);
            plot(0,zMax,'go','linewidth',8);
            plot(xMin,0,'go','linewidth',8);
            plot(0,zMin,'go','linewidth',8);
            plot(xMax,0,'ro','linewidth',10);
            pause(1);
            i=0;
            data = noraxon_stream_collect(stream_config,
5,handles.comunicacionArduino,i);
            fname =
sprintf('./Adquisicion_Señales/Individuo/Entrenamiento/%d_Punto.mat',
entrenamiento);
            save(fname,'data');
            pause(1);
            opcionEntrenamiento=questdlg('Entrenamiento completado.
¿Entrenar de nuevo?','Entrenamiento','Si','No','Si');
            switch opcionEntrenamiento
                case 'Si'
                    punto=0;
                    cla
                    entrenamiento=entrenamiento+1;
                case 'No'
                    punto=4;
                    entrenamiento=0;
                    fclose(handles.comunicacionArduino);
            end
    end
end
```

6.2.2 Recogida de valores simultánea

Uno de los problemas con los que ha habido que lidiar ha sido la recogida de valores de las señales electromiográficas y del acelerómetro-giroscopio simultáneamente. Se estudiaron posibles métodos para resolverlo, entre ellos destacaron:

- Utilizar dos programas de Matlab para que trabajen en paralelo.
- Mandar una señal de trigger por el Jack al software Noraxon para controlar el momento del inicio y del fin.
- Enviar unos determinados comandos al Arduino a través de la interfaz de Matlab.

Finalmente se optó por la última opción. La primera opción requería un ordenador con un procesador adecuado, y la segunda no se pudo realizar por las condiciones del jack de comunicación.

El envío de los comandos se realiza de la siguiente manera:

1º) Al pulsar el botón de CALIBRACIÓN, el programa calibracionGY.m de Matlab envía al Arduino un "0" por medio de la función **fprintf()**. El Arduino realiza la etapa de calibración del sensor GY-521.

2º) En los botones de ARRIBA, ABAJO, IZQUIERDA y DERECHA, el programa recepcionValoresGY.m envía un "1". El Arduino recoge un valor para que funcione como valor máximo o mínimo.

3º) Una vez pulsado el botón COMENZAR ENTRENAMIENTO, se inicia la adquisición de valores durante un cierto intervalo de tiempo. Para iniciar el almacenamiento, a partir del programa noraxon_stream_collect.m se le envía al Arduino el valor "3". Una vez terminado el tiempo, Matlab le envía el valor "5" para que deje de almacenar valores. La velocidad de envío de datos del Arduino es reducida, por lo que Matlab, después de pasarle el valor 5 le escribe un valor "f" para que le pase todos los valores hasta que llegue a ese char. De esta forma se consigue vaciar el buffer con información que aún no se había recibido pero que eran datos medidos en ese intervalo de tiempo.

6.3 Modo offline

Al activar el modo offline en la interfaz principal, ésta permite realizar el estudio de tres tipos de clasificadores de regresión:

- Clasificador de regresión: en el que se analizan los 4 canales de los músculos EMG.
- Clasificador de regresión en el eje X: en el que sólo intervienen los movimientos respecto al eje X, es decir, flexión/extensión.
- Clasificador de regresión en el eje Z: clasificador similar al anterior, pero en este caso los movimientos que se estudian son los que están relacionados con el eje Z (abducción/aducción).

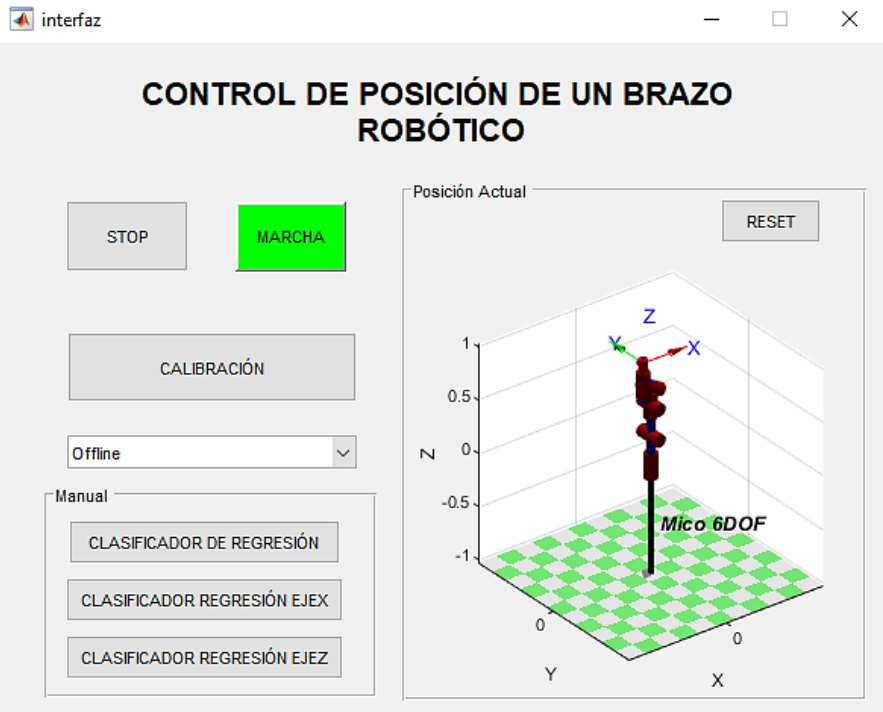


Figura 6.11 Modo offline interfaz [61]

Para simular la figura del robot ha sido necesaria la librería de Peter Corke. El código se localiza en el archivo **interfaz.m**:

```
function varargout = interfaz(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @interfaz_OpeningFcn, ...
                   'gui_OutputFcn',  @interfaz_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Se ejecuta antes de que la interfaz se haga visible.
function interfaz_OpeningFcn(hObject, eventdata, handles, varargin)
%Colocación de la interfaz en el centro de la pantalla
scrsz = get(0,'ScreenSize');
pos_act = get(gcf,'Position');
```

[61] Fuente propia

```
xr = scrsz(3) - pos_act(3);
xp = round(xr/2);
yr = scrsz(4) - pos_act(4);
yp = round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

%% Añade las carpetas y subcarpetas necesarias
addpath(genpath('./SimuladorMico2/robot-10.2'));
addpath(genpath('Adquisicion_Señales'));

% Deshabilita los botones del proceso manual hasta que no se seleccione el
% modo manual en la lista si estamos en el inicio para que no se resetee
% al cambiar de interfaz
funcionando=get(handles.marcha,'value');
if funcionando==0
    set(handles.clasificadorRegresionX,'enable','off');
    set(handles.clasificadorRegresionZ,'enable','off');
    set(handles.manualAuto,'enable','off');
    set(handles.calibracion,'enable','off');
    set(handles.clasificadorRegresion,'enable','off');
end
calibrado=0;
handles.calibrado=calibrado;

%% Arrancar simulador robot
% Inicio librería Peter Corke
startup_rvc;
% Cargar robot Kinova Mico 6gdl
mdl_mico;
mico.plot([0 0 0 0 0 0]);
% Creación de una variable propia en la interfaz del robot Mico para poder
% utilizarla en otra función
handles.RobotMico=mico;
mico
%mico.plot([0 -210 180 0 0 0]);

% Choose default command line output for interfaz
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = interfaz_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

% Crea la conexión con el Arduino.
function marcha_Callback(hObject, eventdata, handles)
clc;
%% Cambio del color del botón a verde
set(gcbo,'Backgroundcolor','g');
set(gcbo,'enable','inactive');
set(handles.STOP,'enable','on');
set(handles.STOP,'Backgroundcolor',[0.94 0.94 0.94]);
%% Arrancar simulador robot
q0=[0 0 0 0 0 0];
q1=[pi pi pi pi/2 0 0];
% Definir las trayectorias
```

```
t=0:0.05:2;
tray1=jtraj(q0,q1,t);
handles.home=q1;
% Mostrar las trayectorias
parado=get(handles.STOP,'Value');
handles.STOP
if parado~=1
    handles.RobotMico.plot(tray1);
end
parado=get(handles.STOP,'Value')
% Habilita los botones necesarios
set(handles.calibracion,'enable','on');
set(handles.manualAuto,'enable','on');
guidata(gcbo,handles);

function STOP_Callback(hObject, eventdata, handles)
set(gcbo,'Backgroundcolor','r');
set(gcbo,'enable','off');
set(handles.marcha,'enable','on');
set(handles.marcha,'Backgroundcolor',[0.94 0.94 0.94]);
if handles.calibrado==1
fclose(handles.comunicacionArduino);
calibrado=0;
handles.comunicacionArduino % Para ver lo que hay dentro de la variable
end

guidata(gcbo,handles);
% Deshabilita todos los botones
set(handles.clasificadorRegresionX,'enable','off');
set(handles.clasificadorRegresionZ,'enable','off');
set(handles.calibracion,'enable','off');
set(handles.manualAuto,'enable','off');
set(handles.manualAuto,'Value',1);
set(handles.clasificadorRegresion,'enable','off');

% --- Executes on button press in calibracion.
function calibracion_Callback(hObject, eventdata, handles)
    try %% Inicializar puerto (borrar conexiones anteriores)
        delete(instrfind({'Port'},{'COM5'}));
        %% Comprobamos la conexion entre el Arduino Uno y Matlab
        s=serial('COM5','Baudrate',115200);
        %Informe de errores
        warning('off','MATLAB:serial:fscanf:unsuccessfulRead');

        %% Inicia la comunicación con Noraxon (llama al archivo de
inicialización)
         stream_config = noraxon_stream_init('127.0.0.1', 9220);

         %% Guardar los objetos de comunicación
        handles.comunicacionArduino=s;
        handles.comunicacionNoraxon=stream_config;
        calibrado=1;
        handles.calibrado=calibrado;
        guidata(gcbo,handles);

        interfaz_calibracion(handles.comunicacionArduino, handles.marcha);
    catch
        f=errordlg('Algún dispositivo no está conectado o se ha
desconectado.','Error');
    end
```

```
% Programa que le da al usuario la opción de elegir modo.
function manualAuto_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
str = get(hObject,'String');
switch str{val}

    case 'Offline' % El usuario ha seleccionado modo manual
        set(handles.clasificadorRegresionX,'enable','on');
        set(handles.clasificadorRegresionZ,'enable','on');
        set(handles.clasificadorRegresion,'enable','on');

    case 'Online' % El usuario ha seleccionado modo automático
        a=msgbox('Disponible próximamente');
        set(handles.clasificadorRegresionX,'enable','off');
        set(handles.clasificadorRegresionZ,'enable','off');
        set(handles.clasificadorRegresion,'enable','off');
end

function manualAuto_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function clasificadorRegresion_Callback(hObject, eventdata, handles)
% Llamada a la función regresion para realizar la regresión lineal a todo
% el conjunto
regression;

function clasificadorRegresionX_Callback(hObject, eventdata, handles)
regresion_2channels_x;

function clasificadorRegresionZ_Callback(hObject, eventdata, handles)
regresion_2channels_z;


function graficaXZ_CreateFcn(hObject, eventdata, handles)


function resetGrafica_Callback(hObject, eventdata, handles)
q1=[pi pi pi pi/2 0 0];
handles.RobotMico.plot(q1);

function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

7 RESULTADOS

En los subapartados siguientes se muestran los datos de las señales adquiridas y los resultados que ha obtenido cada usuario, así como los valores extraídos a partir de los diferentes métodos de análisis de comparación entre las señales calculadas de la trayectoria y las que realmente ha medido el acelerómetro-giroscopio.

Las primeras gráficas corresponden con las señales EMG y del sensor medidas al realizar determinados movimientos (de izquierda a derecha y de arriba abajo: abducción, flexión, aducción y extensión). Posteriormente se muestran los resultados gráfica y analíticamente del análisis de los 4 canales de las señales electromiográficas. Finalmente se muestran los resultados a partir del estudio de cada eje por separado, es decir, el eje x se estudia con los movimientos de flexión/extensión; y el eje z con abducción/aducción [62].

7.1 Individuo 1

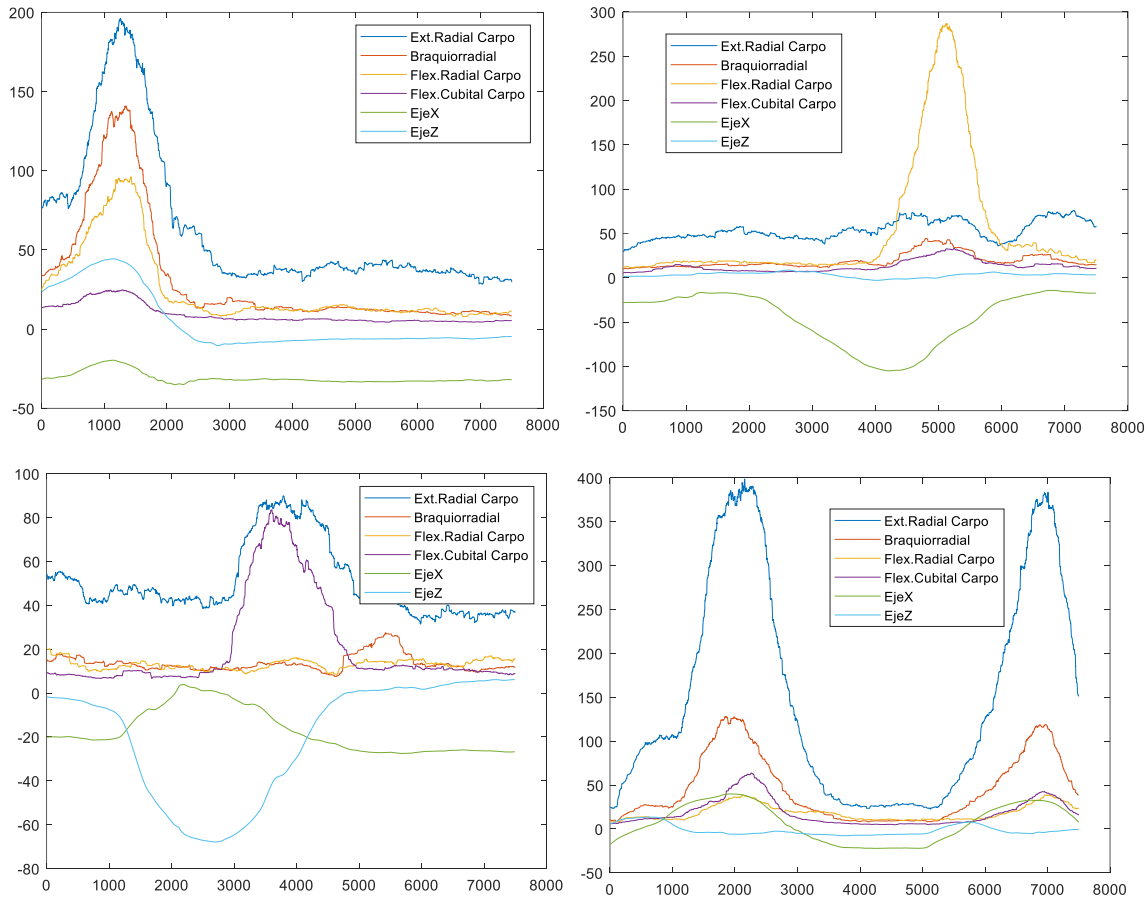


Figura 7.1 Señales de los movimientos medidos del Individuo 1.

[62] Todas las gráficas y tablas de este apartado son de Fuente propia.

ESTUDIO CON LOS 4 CANALES EMG:

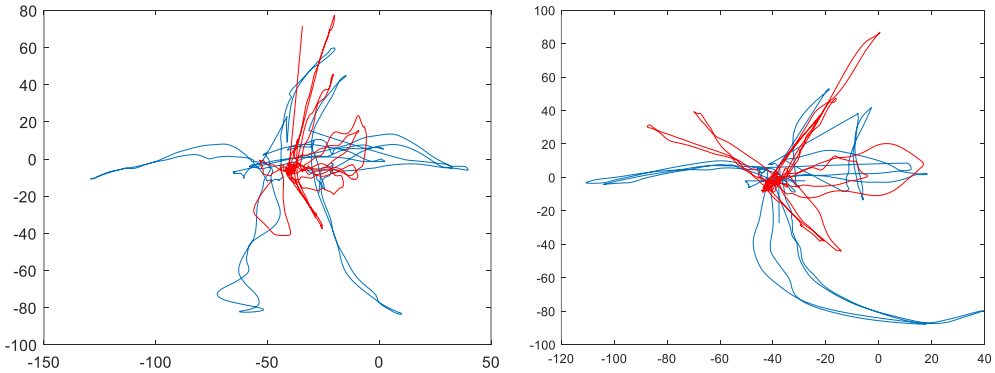
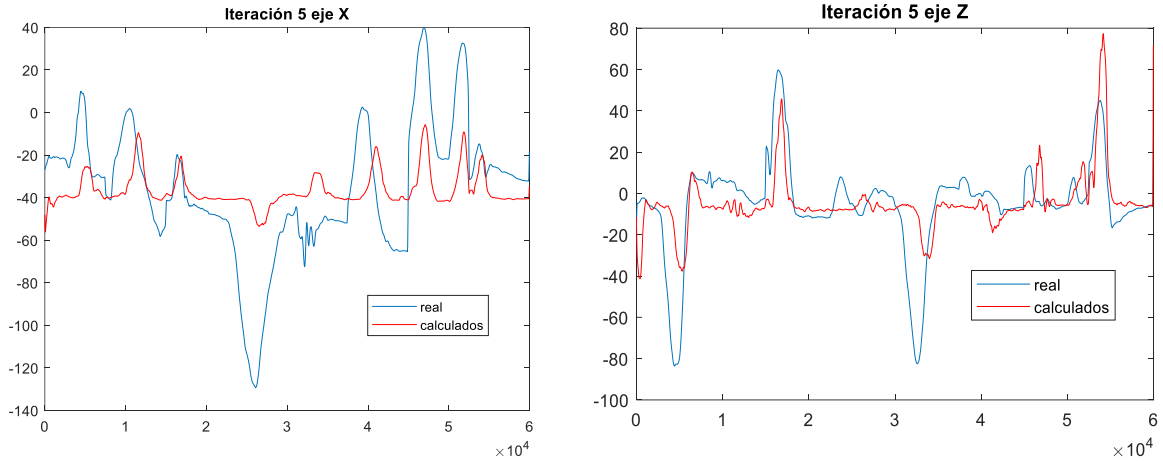
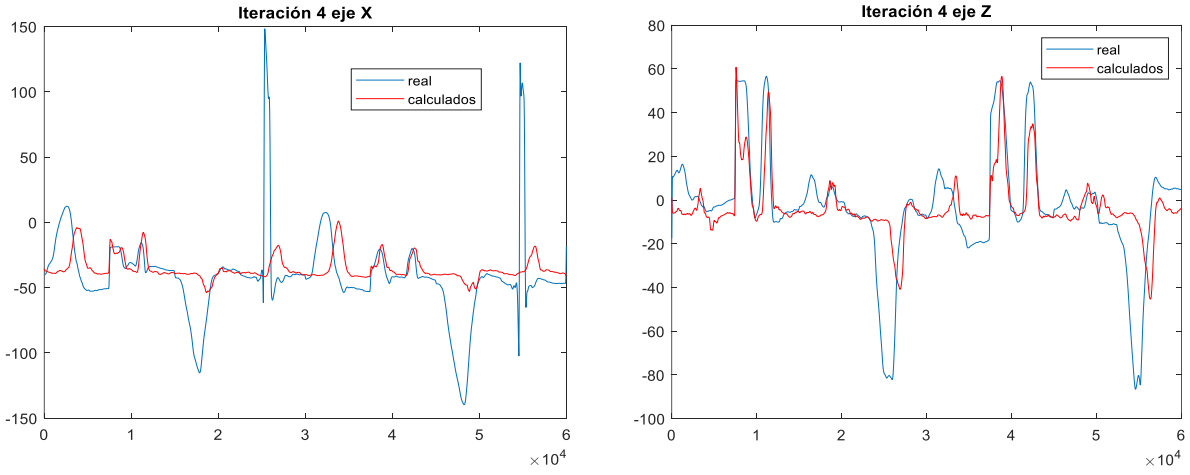


Figura 7.2 Ejemplos de trayectoria real (azul) vs. trayectoria calculada del Individuo 1.





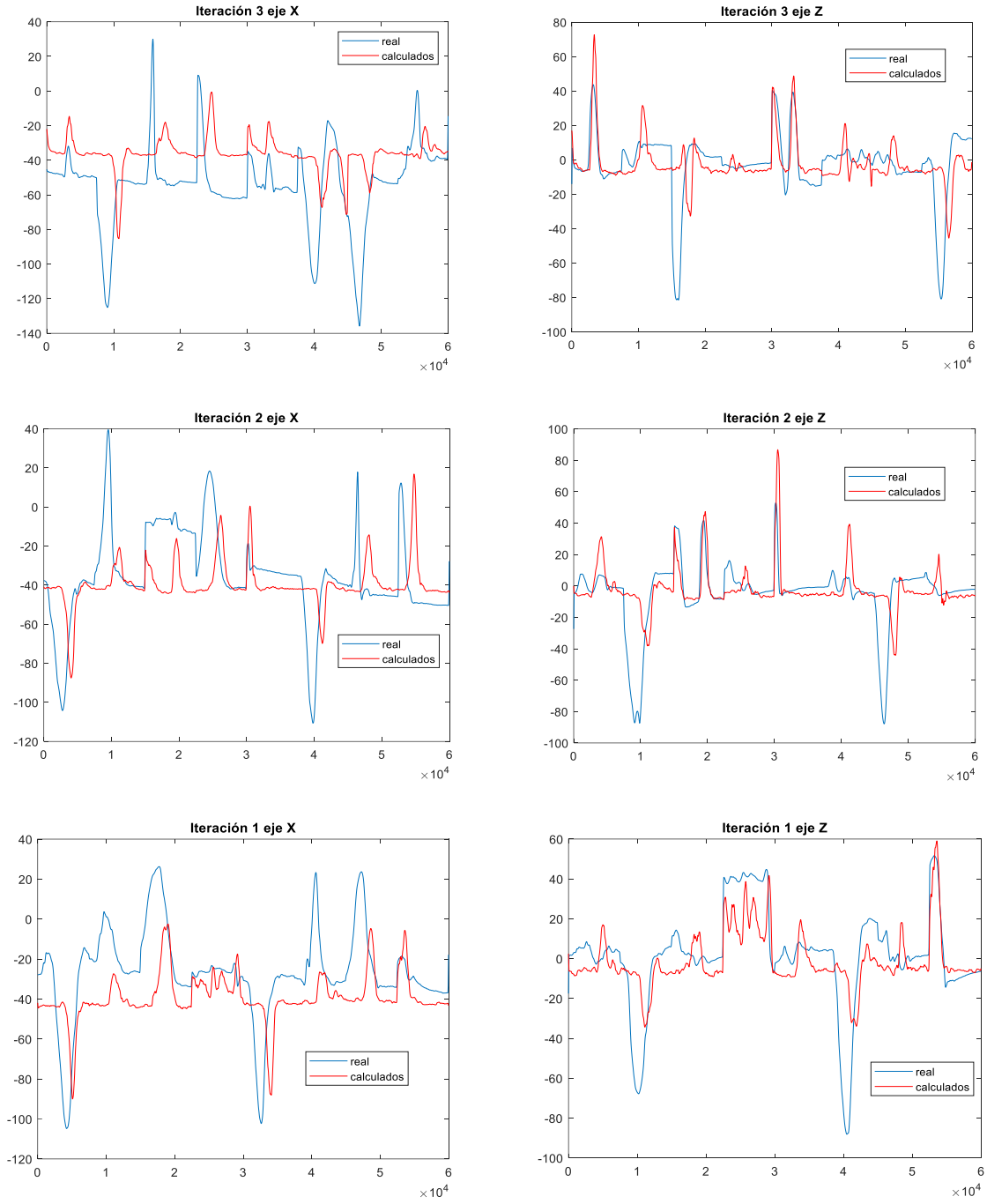


Figura 7.3 Gráfica de los valores de x y de z reales vs. valores calculados para cada iteración del Individuo 1.

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1887	-0.0135	3.3291	2.0834	156.9783	-0.2022	-0.0714	3.3157
2	0.1658	-0.0112	4.8078	4.3108	18.7757	-0.1770	-0.0672	4.7966
3	0.1781	0.0051	6.3812	2.2196	303.8697	-0.1730	0.0285	6.3863
4	0.1127	0.0192	12.7088	9.7592	27.1798	-0.0935	0.1706	12.7281
5	0.1596	0.0709	11.2835	11.2223	0.3864	-0.0888	0.4440	11.3543
5	0.1610	0.0141	7.7021	5.9191	101.4380	-0.1469	0.1009	7.7162

Tabla 7.1 Valores de los métodos de predicción de errores del eje x del Individuo 1.

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1345	0.0095	1.8024	1.7741	7.0437	-0.1251	0.0704	1.8119
2	0.1468	0.0148	2.1989	2.1664	5.3388	-0.1319	0.1012	2.2138
3	0.1433	0.0136	2.0714	2.0710	0.0539	-0.1297	0.0948	2.0850
4	0.1399	0.0396	2.2769	2.2767	0.0241	-0.1003	0.2831	2.3165
5	0.1285	0.0066	1.7902	1.7902	0.0163	-0.1219	0.0512	1.7968
6	0.1386	0.0168	2.0280	2.0157	2.4954	-0.1218	0.1201	2.0448

Tabla 7.2 Valores de los métodos de predicción de errores del eje z del Individuo 1.

ESTUDIO CON 2 CANALES EMG:

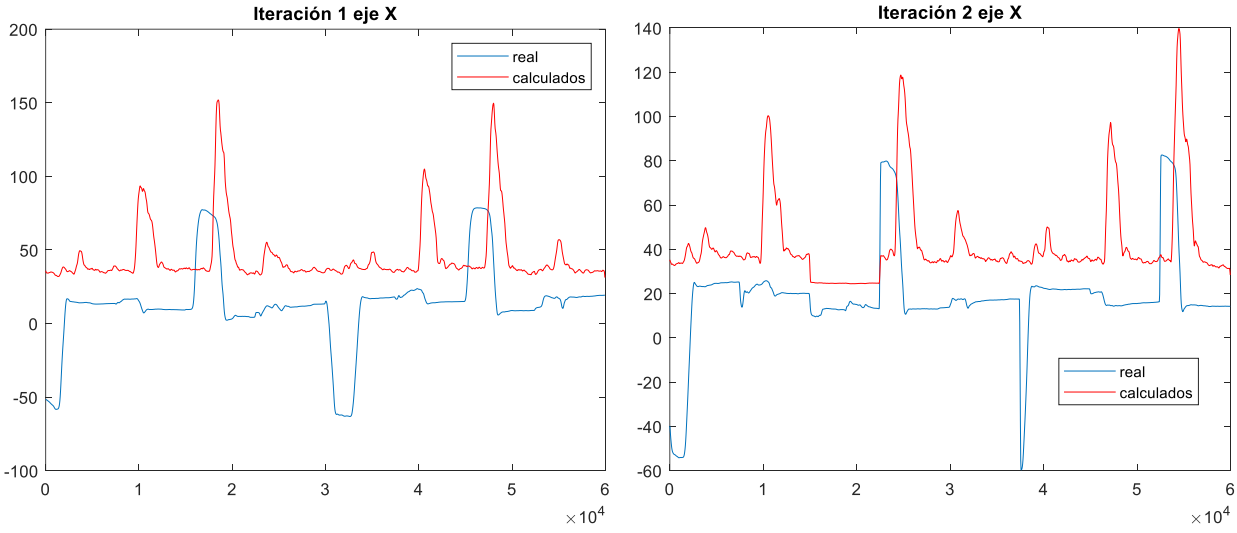


Figura 7.4 Gráfica de los valores de x reales vs. valores calculados para cada iteración del Individuo 1 (2 canales).

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.3115	0.0138	1.9910	1.3058	1.0322e+03	-0.2977	0.0443	2.0048
2	0.2499	-0.0070	1.7913	1.3301	553.8949	-0.2570	-0.0281	1.7843
3	0.2701	0.0426	4.6997	1.6486	724.5666	-0.2276	0.1576	4.7423
4	0.2750	0.0134	29.9665	4.2102	1.5961e+03	-0.2616	0.0486	29.9799
5	0.3195	-0.0859	91.6057	2.5043	3.9855e+03	-0.4054	-0.2689	91.5198
6	0.2852	-0.0046	26.0108	2.1998	1.5784e+03	-0.2898	-0.0093	26.0062

Tabla 7.3 Valores de los métodos de predicción de errores del eje x del Individuo 1 (2 canales).

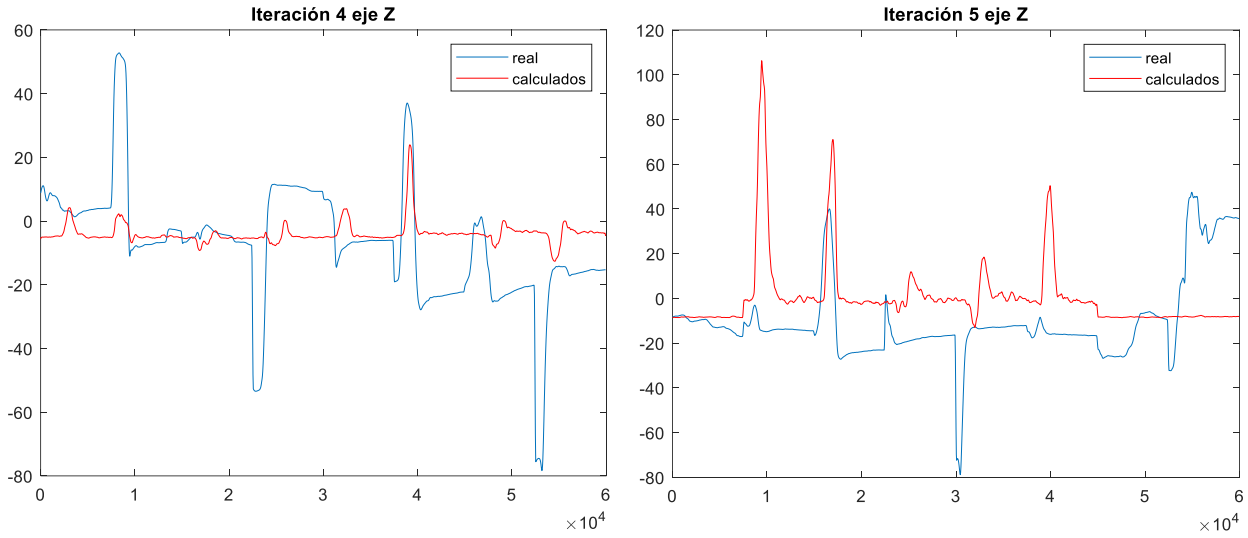


Figura 7.5 Gráfica de los valores de z reales vs. valores calculados para cada iteración del Individuo 1 (2 canales).

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1770	0.0291	90.4193	7.1350	49.8870	-0.1479	0.1646	90.4484
2	0.1637	4.7524e-04	55.7879	5.2512	47.9022	-0.1632	0.0029	55.7884
3	0.1487	-6.1107e-04	43.4396	3.9180	53.1308	-0.1493	-0.0041	43.4390
4	0.1380	0.0295	27.9496	16.0085	9.0686	-0.1085	0.2138	27.9791
5	0.2119	-0.0181	2.3352	1.9259	114.0230	-0.2300	-0.0853	2.3172
6	0.1679	0.0081	43.9863	6.8477	54.8023	-0.1598	0.0584	43.9944

Tabla 7.4 Valores de los métodos de predicción de errores del eje z del Individuo 1 (2 canales).

7.2 Individuo 2

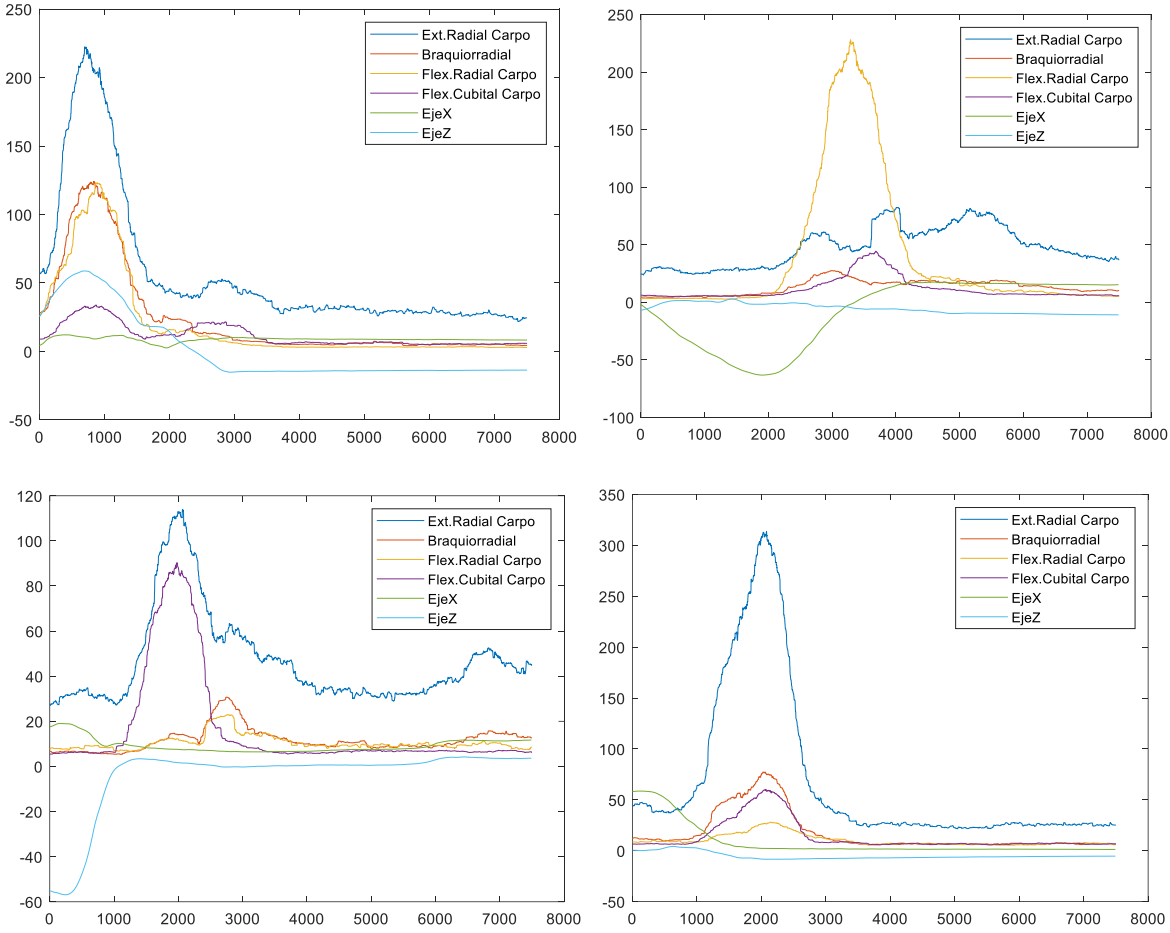


Figura 7.6 Señales de los movimientos medidos del Individuo 2.

ESTUDIO CON LOS 4 CANALES EMG:

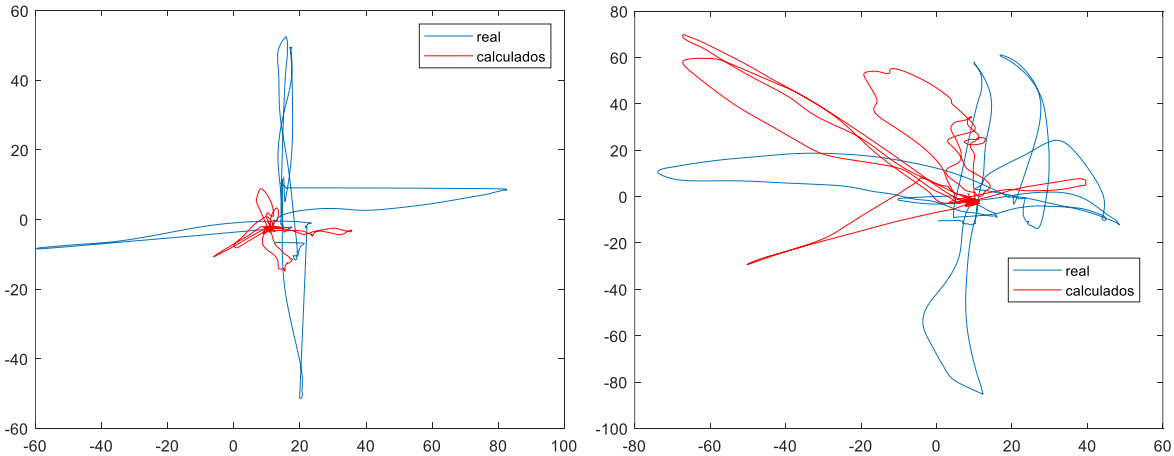
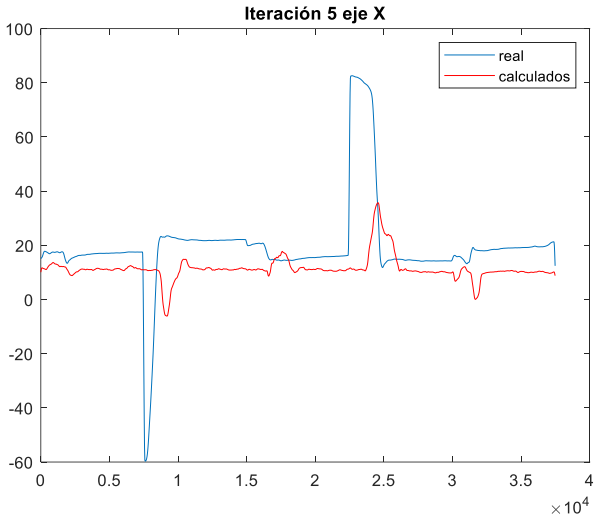
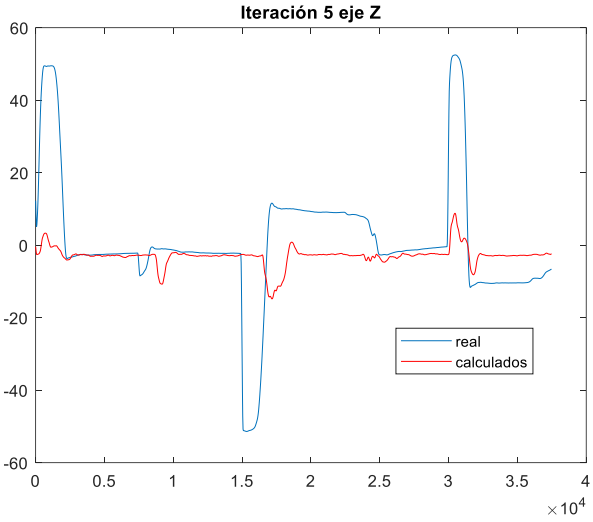
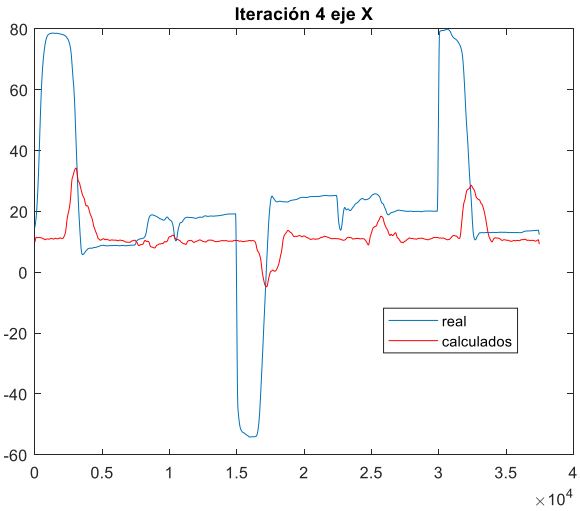
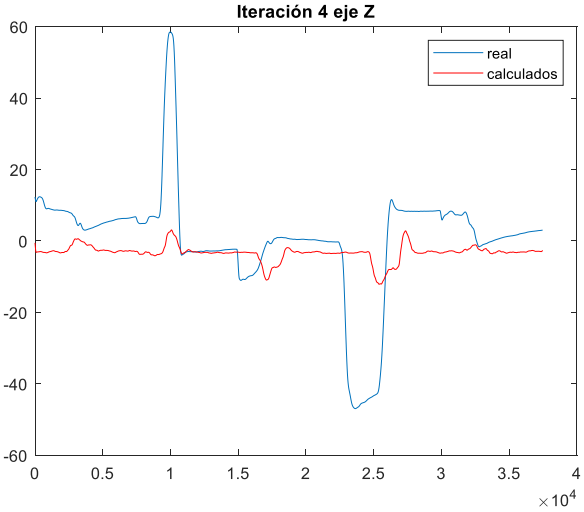


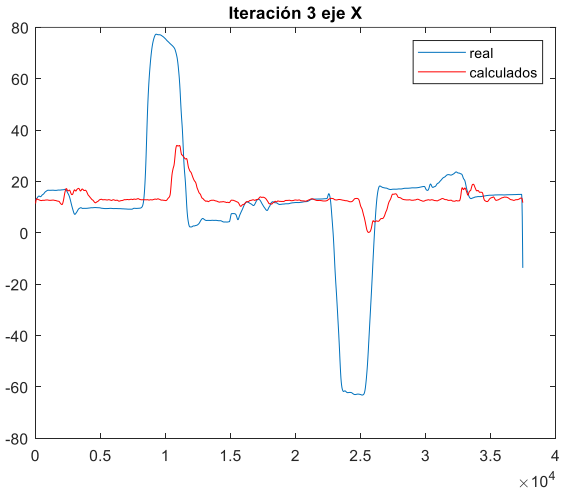
Figura 7.7 Ejemplos de trayectoria real (azul) vs. trayectoria calculada del Individuo 2.

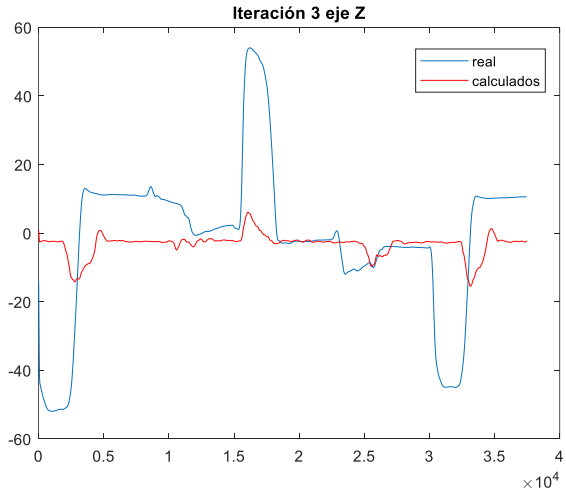

Iteración 5 eje X
real
calculados
×10^4


Iteración 5 eje Z
real
calculados
×10^4


Iteración 4 eje X
real
calculados
×10^4


Iteración 4 eje Z
real
calculados
×10^4


Iteración 3 eje X
real
calculados
×10^4


Iteración 3 eje Z
real
calculados
×10^4

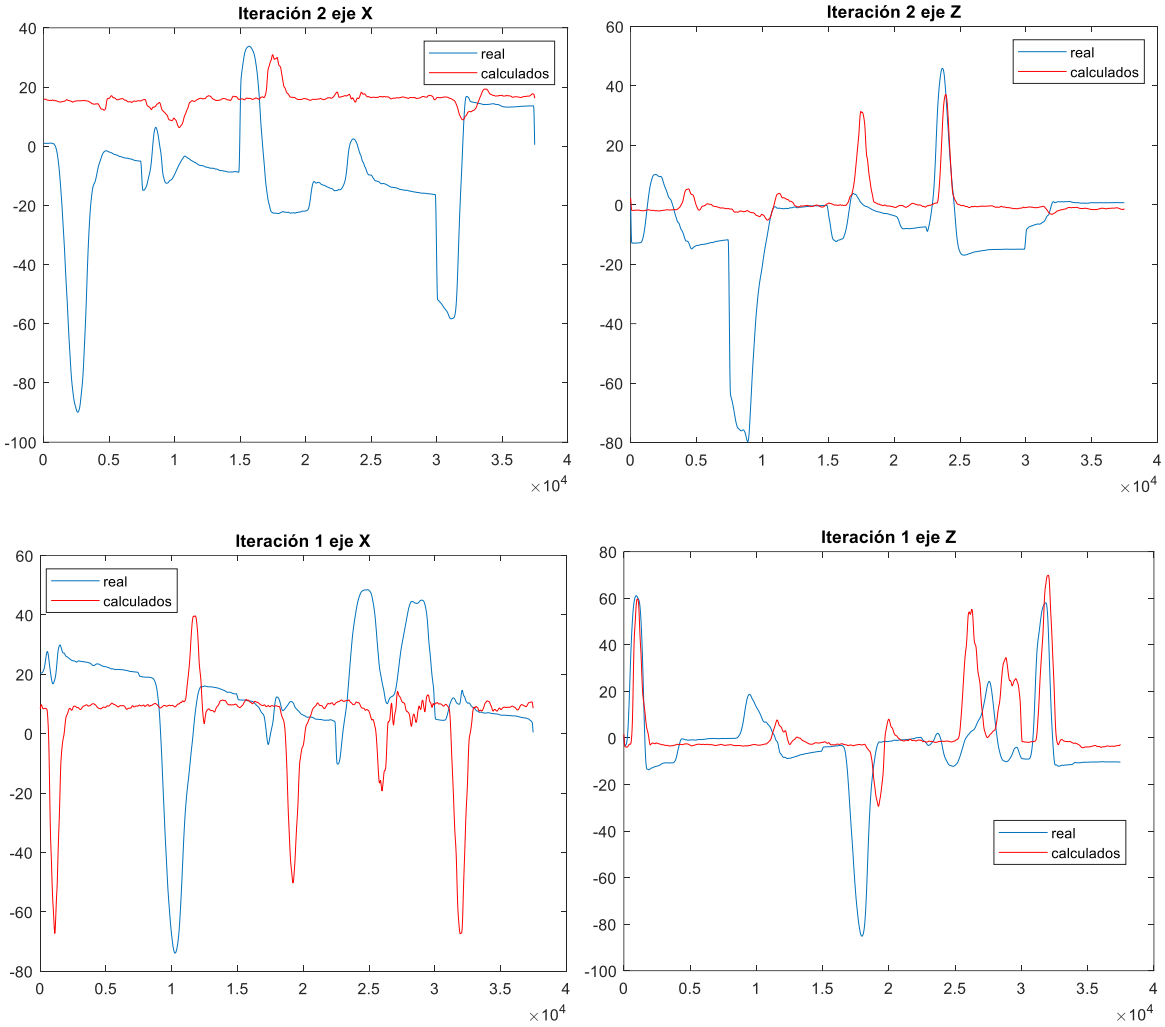


Figura 7.8 Gráfica de los valores de x y de z reales vs. valores calculados para cada iteración del Individuo 2.

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.2207	-0.0193	2.7661	2.4002	62.9753	-0.2400	-0.0874	2.7468
2	0.2759	0.0346	52.0006	1.6904	678.9551	-0.2413	0.1253	52.0351
3	0.1703	0.0218	33.1798	28.5082	2.9164	-0.1486	0.1279	33.2016
4	0.2022	-0.0079	23.6410	6.3848	87.7175	-0.2101	-0.0390	23.6331
5	0.1301	-0.0188	13.6165	3.9526	66.3423	-0.1488	-0.1442	13.5977
6	0.1998	0.0021	25.0408	8.5872	179.7813	-0.1977	-0.0035	25.0429

Tabla 7.5 Valores de los métodos de predicción de errores del eje x del Individuo 2.

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1324	0.0147	1.4714	1.4048	37.7074	-0.1177	0.1110	1.4861
2	0.1487	0.0405	7.5312	3.1250	75.3073	-0.1083	0.2721	7.5717
3	0.2013	-0.0077	44.3525	34.7279	2.9091	-0.2090	-0.0385	44.3448
4	0.1441	0.0285	43.6511	11.4997	15.1207	-0.1156	0.1975	43.6796
5	0.1635	-0.0071	38.3581	12.8968	15.2395	-0.1706	-0.0436	38.3510
6	0.1580	0.0137	27.0729	12.7308	29.2568	-0.1442	0.0997	27.0866

Tabla 7.6 Valores de los métodos de predicción de errores del eje z del Individuo 2.

ESTUDIO CON 2 CANALES EMG:

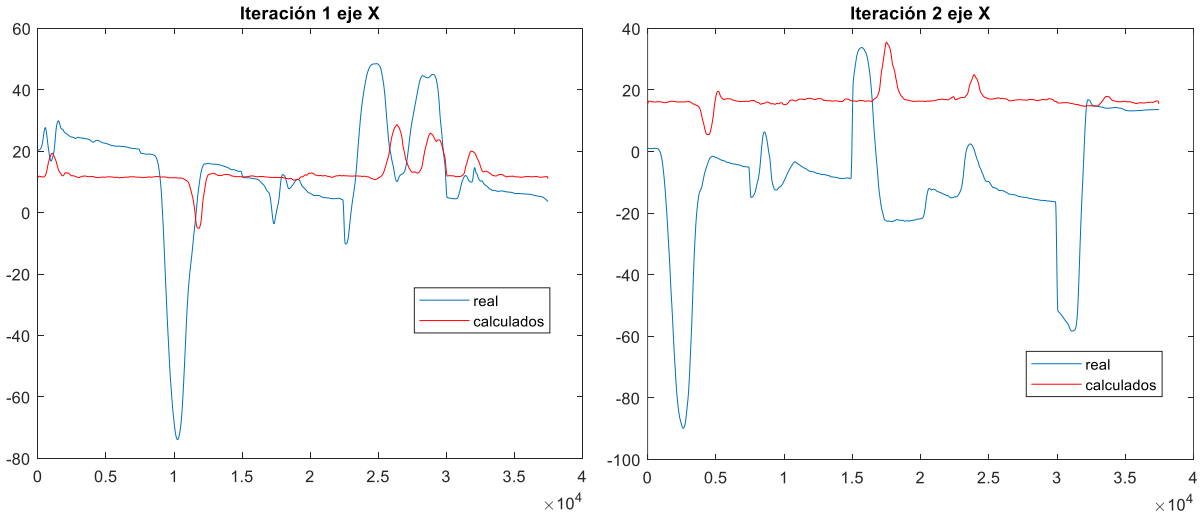


Figura 7.9 Gráfica de los valores de x reales vs. valores calculados para cada iteración del Individuo 2 (2 canales).

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1546	0.0397	20.8294	20.7752	0.0471	-0.1149	0.2570	20.8691
2	0.2817	0.0563	49.0885	1.6426	728.7406	-0.2254	0.1999	49.1448
3	0.1745	0.0236	110.7992	100.0925	0.5856	-0.1509	0.1350	110.8228
4	0.2075	-0.0092	84.5030	6.3989	113.1528	-0.2167	-0.0445	84.4938
5	0.1326	-0.0168	35.6497	3.8192	85.5896	-0.1494	-0.1264	35.6329
6	0.1902	0.0187	60.1740	26.5457	185.6232	-0.1715	0.0842	60.1927

Tabla 7.7 Valores de los métodos de predicción de errores del eje x del Individuo 2 (2 canales).

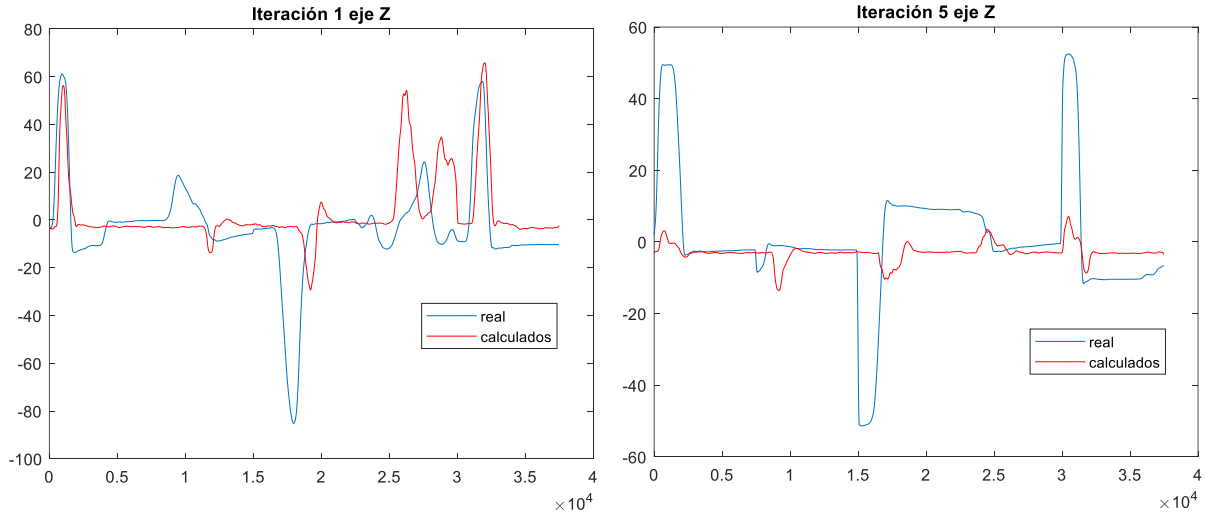


Figura 7.10 Gráfica de los valores de z reales vs. valores calculados para cada iteración del Individuo 2 (2 canales).

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1319	0.0161	1.5597	1.4870	32.5184	-0.1158	0.1222	1.5758
2	0.1474	0.0475	7.5213	3.2849	67.7279	-0.0999	0.3222	7.5688
3	0.2008	-0.0091	64.7470	53.0381	1.5693	-0.2099	-0.0453	64.7379
4	0.1450	0.0310	35.5085	12.3860	12.6293	-0.1140	0.2139	35.5395
5	0.1628	-0.0101	44.9329	13.7913	14.6897	-0.1729	-0.0621	44.9228
6	0.1576	0.0151	30.8539	16.7974	25.8269	-0.1425	0.1102	30.8690

Tabla 7.8 Valores de los métodos de predicción de errores del eje z del Individuo 2 (2 canales).

7.3 Individuo 3

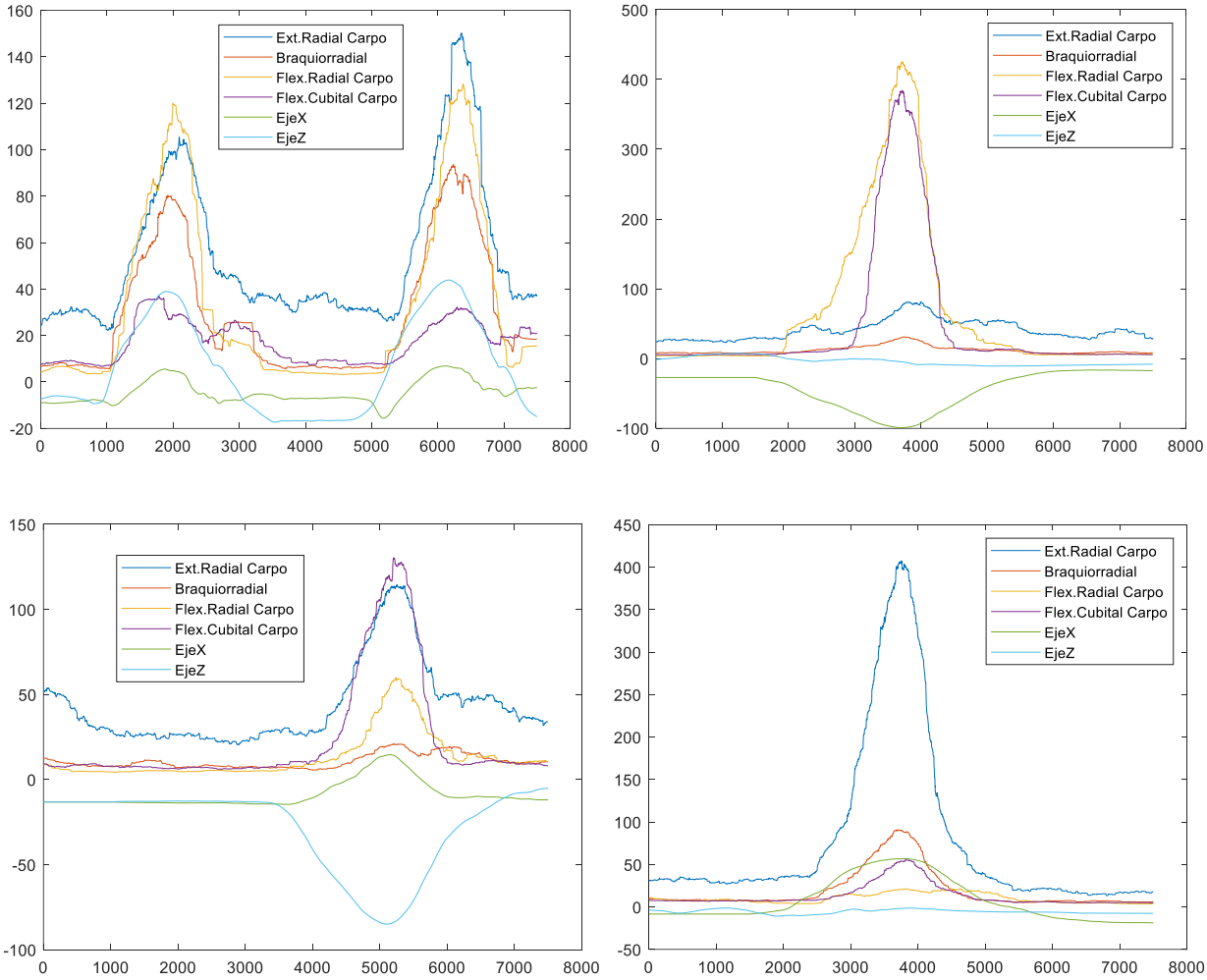


Figura 7.11 Señales de los movimientos medidos del Individuo 3.

ESTUDIO CON LOS 4 CANALES EMG:

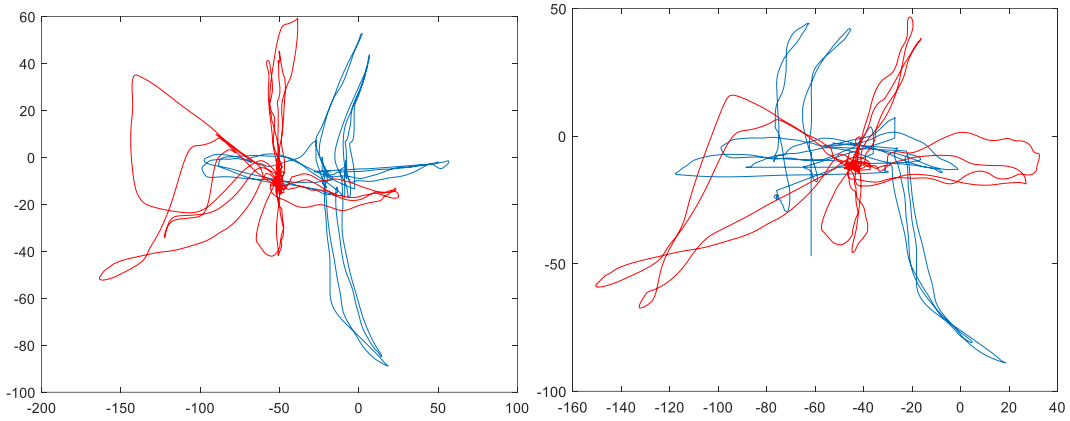
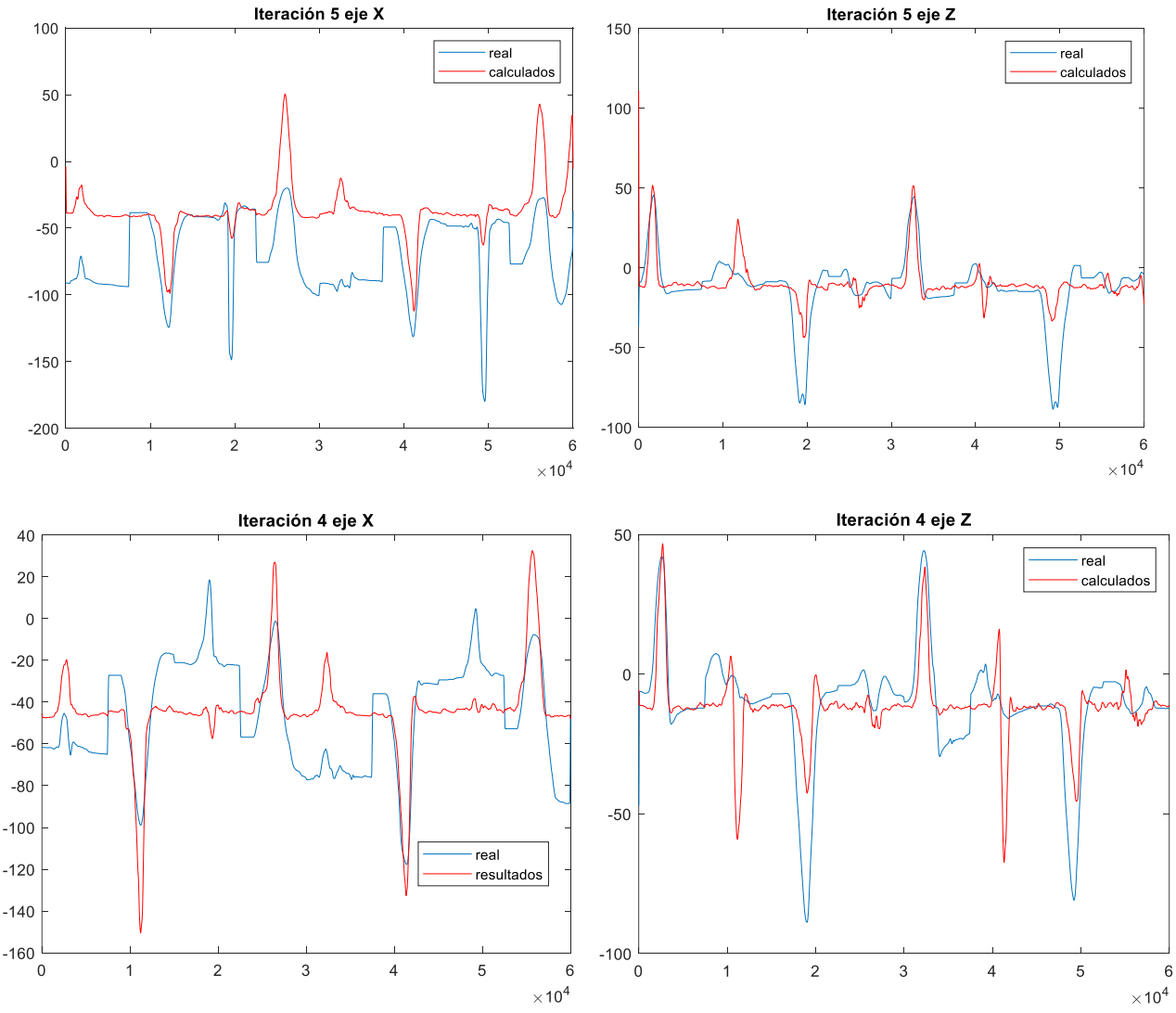


Figura 7.12 Ejemplos de trayectoria real (azul) vs. trayectoria calculada del Individuo 3.



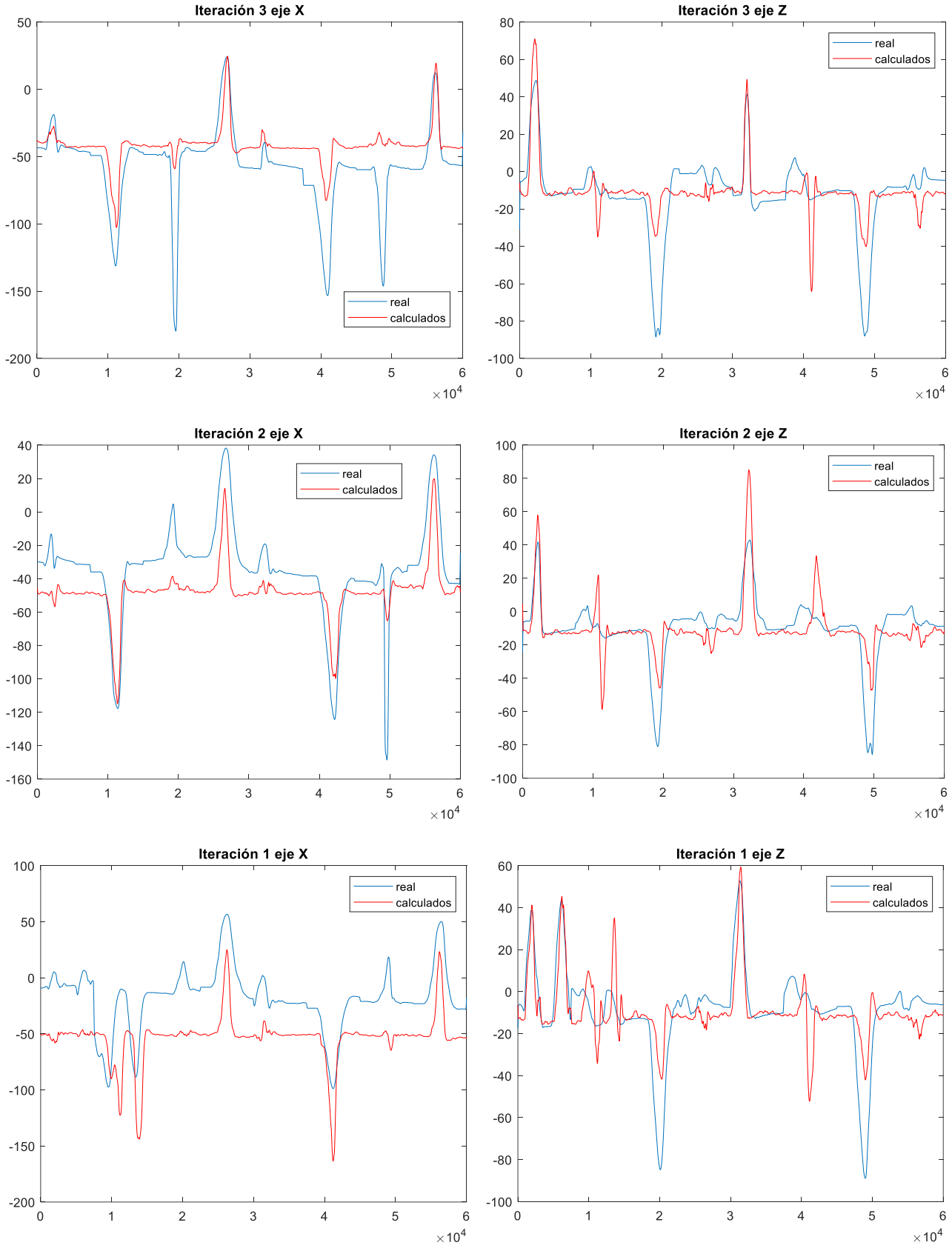


Figura 7.13 Gráfica de los valores de x y de z reales vs. valores calculados para cada iteración del Individuo 3.

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.2655	0.0132	0.8889	0.9704	1.2894e+03	-0.2523	0.0497	0.9021
2	0.1143	-0.0114	1.2538	1.1398	179.5007	-0.1257	-0.0999	1.2424
3	0.1268	-0.0123	2.6569	1.7007	226.2638	-0.1390	-0.0968	2.6446
4	0.1714	-0.0226	1.3163	1.3140	2.9566	-0.1940	-0.1316	1.2937
5	0.2584	0.0015	1.6297	1.1747	1.0480e+03	-0.2569	0.0057	1.6312
6	0.1873	-0.0063	1.5491	1.2599	549.2194	-0.1936	-0.0546	1.5428

Tabla 7.9 Valores de los métodos de predicción de errores del eje x del Individuo 3.

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1042	0.0269	1.1249	1.1242	1.0712	-0.0773	0.2581	1.1518
2	0.1123	0.0203	0.8420	0.8424	0.5158	-0.0920	0.1811	0.8624
3	0.1118	0.0308	1.3424	1.3411	0.6765	-0.0810	0.2754	1.3732
4	0.1121	0.0059	1.5045	1.5010	1.0286	-0.1062	0.0528	1.5104
5	0.1138	0.0290	1.7123	1.6901	4.2839	-0.0848	0.2547	1.7413
6	0.1109	0.0226	1.3052	1.2998	1.5152	-0.0883	0.2044	1.3278

Tabla 7.10 Valores de los métodos de predicción de errores del eje z del Individuo 3.

ESTUDIO CON 2 CANALES EMG:

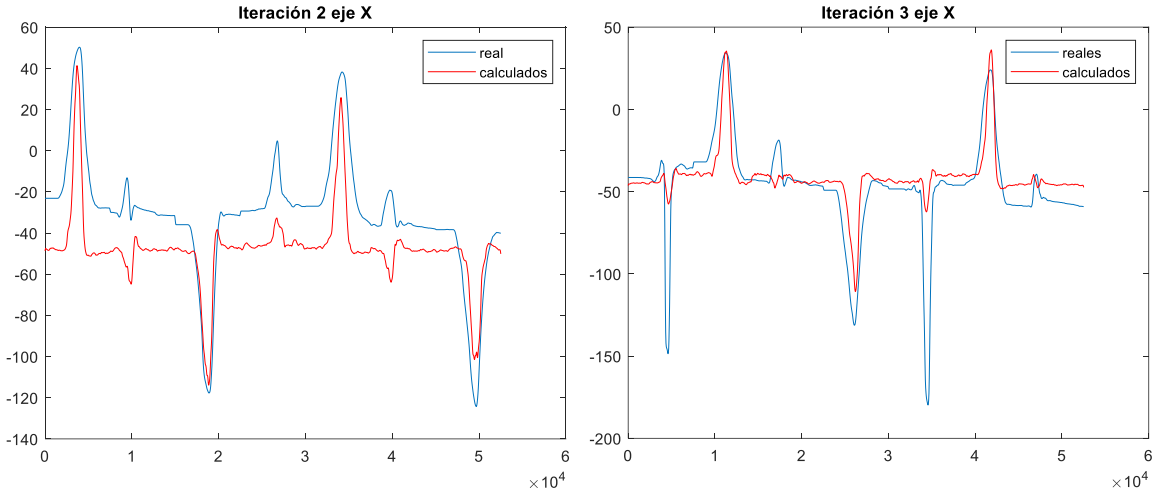


Figura 7.14 Gráfica de los valores de x reales vs. valores calculados para cada iteración del Individuo 3 (2 canales).

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.2580	0.0274	0.8780	0.9623	1.1568e+03	-0.2306	0.1062	0.9054
2	0.1299	0.0586	0.7412	0.8604	275.2992	-0.0713	0.4514	0.7998
3	0.0944	0.0054	1.5441	1.4978	23.1154	-0.0890	0.0572	1.5495
4	0.1475	0.0085	1.6962	1.5462	89.5452	-0.1389	0.0580	1.7048
5	0.1848	0.0029	1.8117	1.4691	230.7564	-0.1819	0.0158	1.8147
6	0.1629	0.0206	1.3342	1.2672	355.1122	-0.1423	0.1377	1.3548

Tabla 7.11 Valores de los métodos de predicción de errores del eje x del Individuo 3 (2 canales).

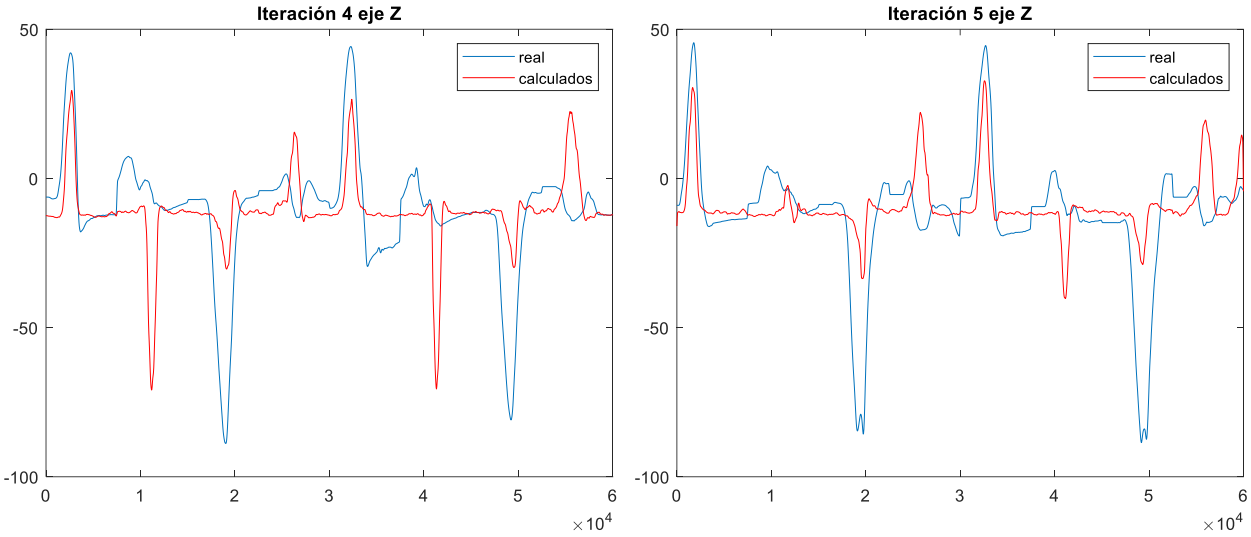
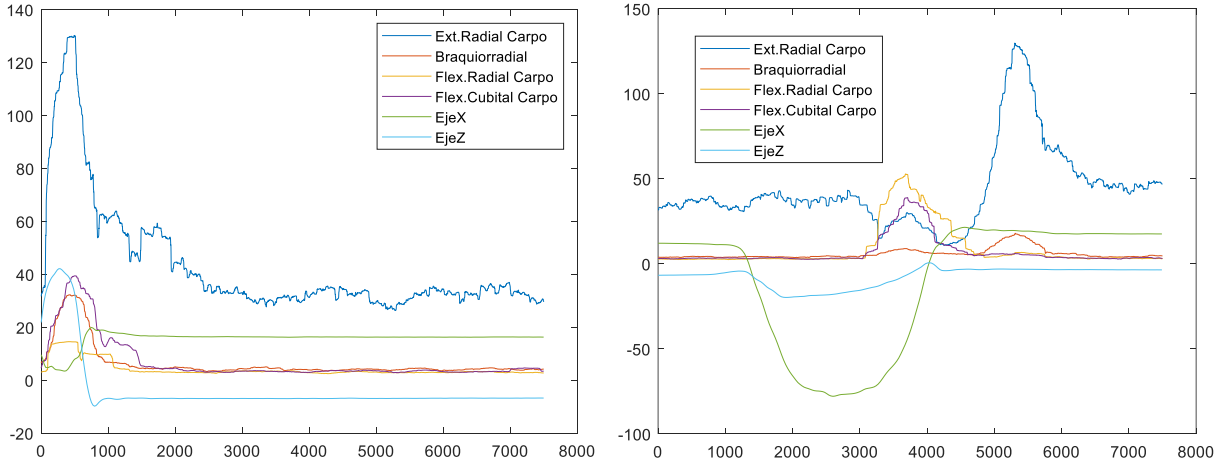


Figura 7.15 Gráfica de los valores de z reales vs. valores calculados para cada iteración del Individuo 3 (2 canales).

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1230	0.0102	2.4370	2.3853	4.5765	-0.1128	0.0827	2.4472
2	0.1196	0.0208	2.4284	2.4114	1.1701	-0.0988	0.1736	2.4492
3	0.1229	0.0215	2.8720	2.8622	0.5219	-0.1014	0.1748	2.8935
4	0.1343	-0.0038	2.3181	2.3170	0.1199	-0.1381	-0.0281	2.3143
5	0.1290	0.0269	3.3022	3.1315	7.0799	-0.1021	0.2089	3.3292
6	0.1258	0.0151	2.6715	2.6215	2.6937	-0.1106	0.1224	2.6867

Tabla 7.12 Valores de los métodos de predicción de errores del eje z del Individuo 3 (2 canales).

7.4 Individuo 4



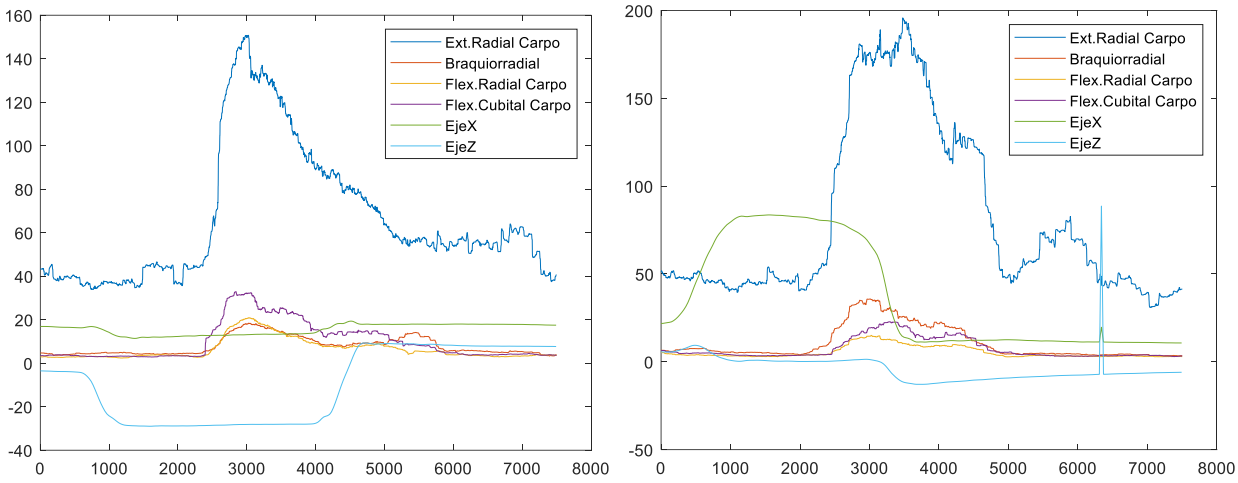


Figura 7.16 Señales de los movimientos medidos del Individuo 4.

ESTUDIO CON LOS 4 CANALES EMG:

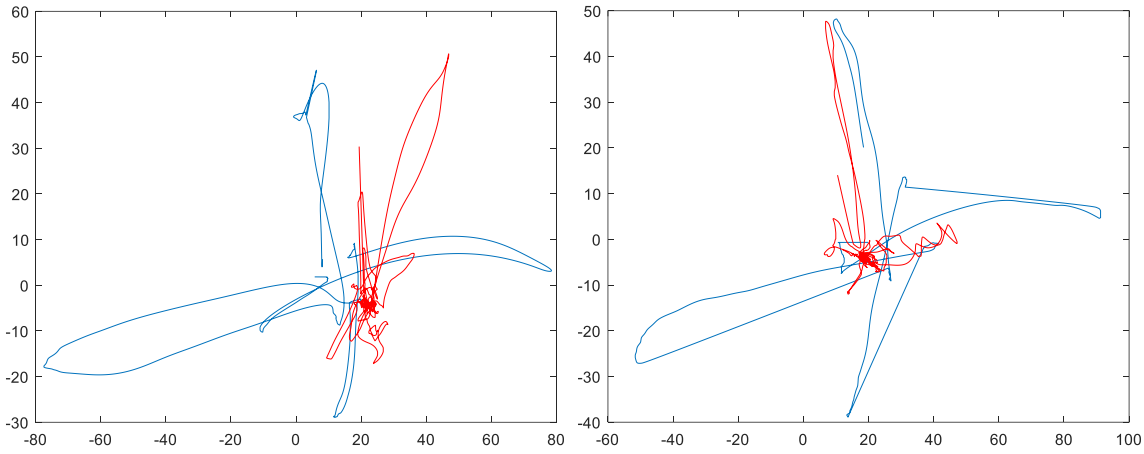
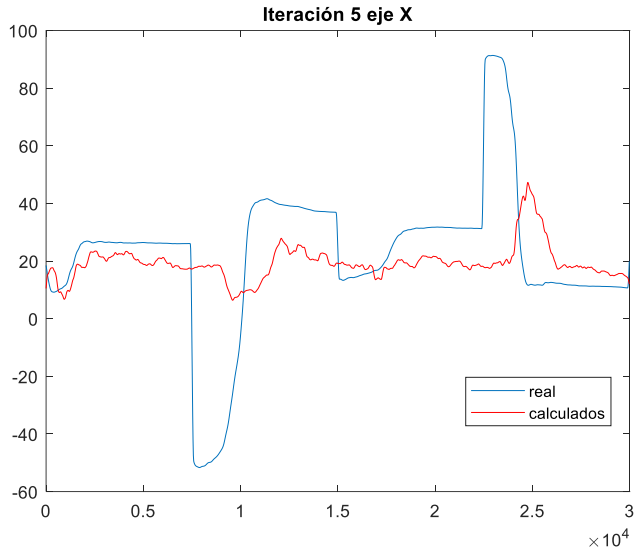
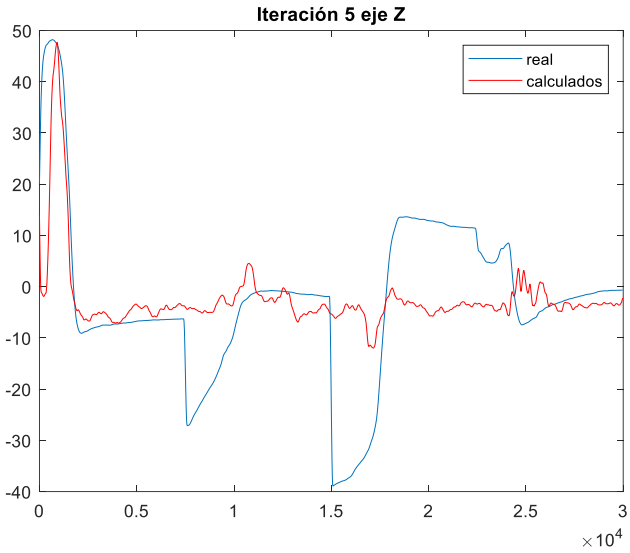
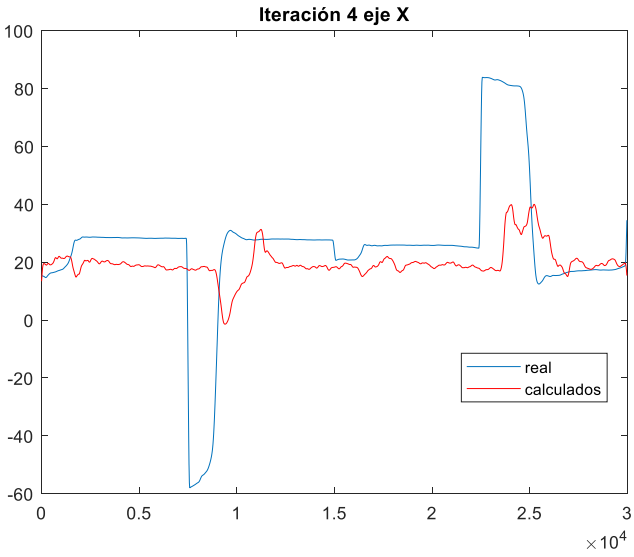
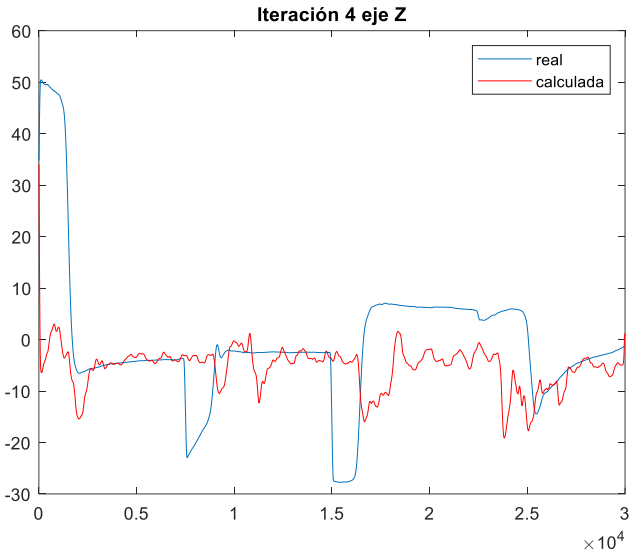


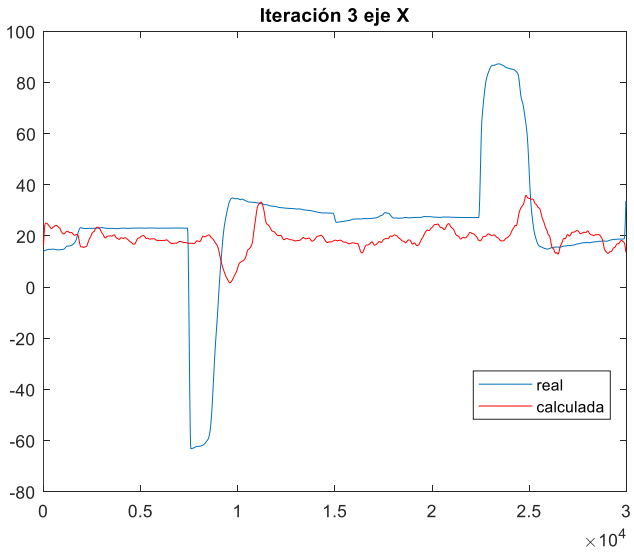
Figura 7.17 Ejemplos de trayectoria real (azul) vs. trayectoria calculada del Individuo 4.

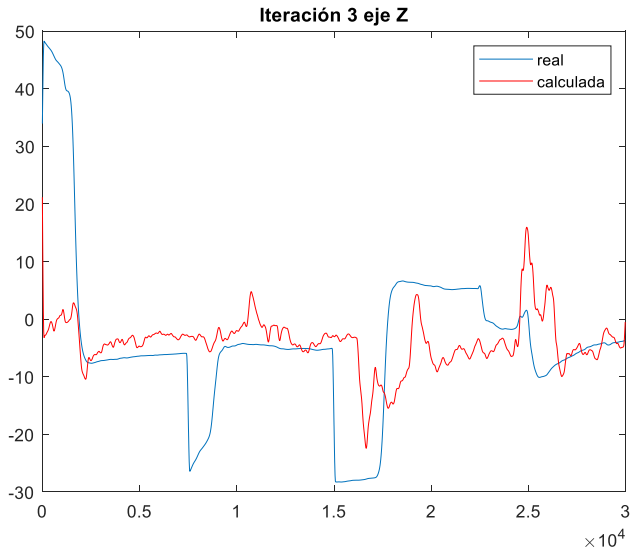

Iteración 5 eje X
real
calculados
×10^4


Iteración 5 eje Z
real
calculados
×10^4


Iteración 4 eje X
real
calculados
×10^4


Iteración 4 eje Z
real
calculada
×10^4


Iteración 3 eje X
real
calculada
×10^4


Iteración 3 eje Z
real
calculada
×10^4

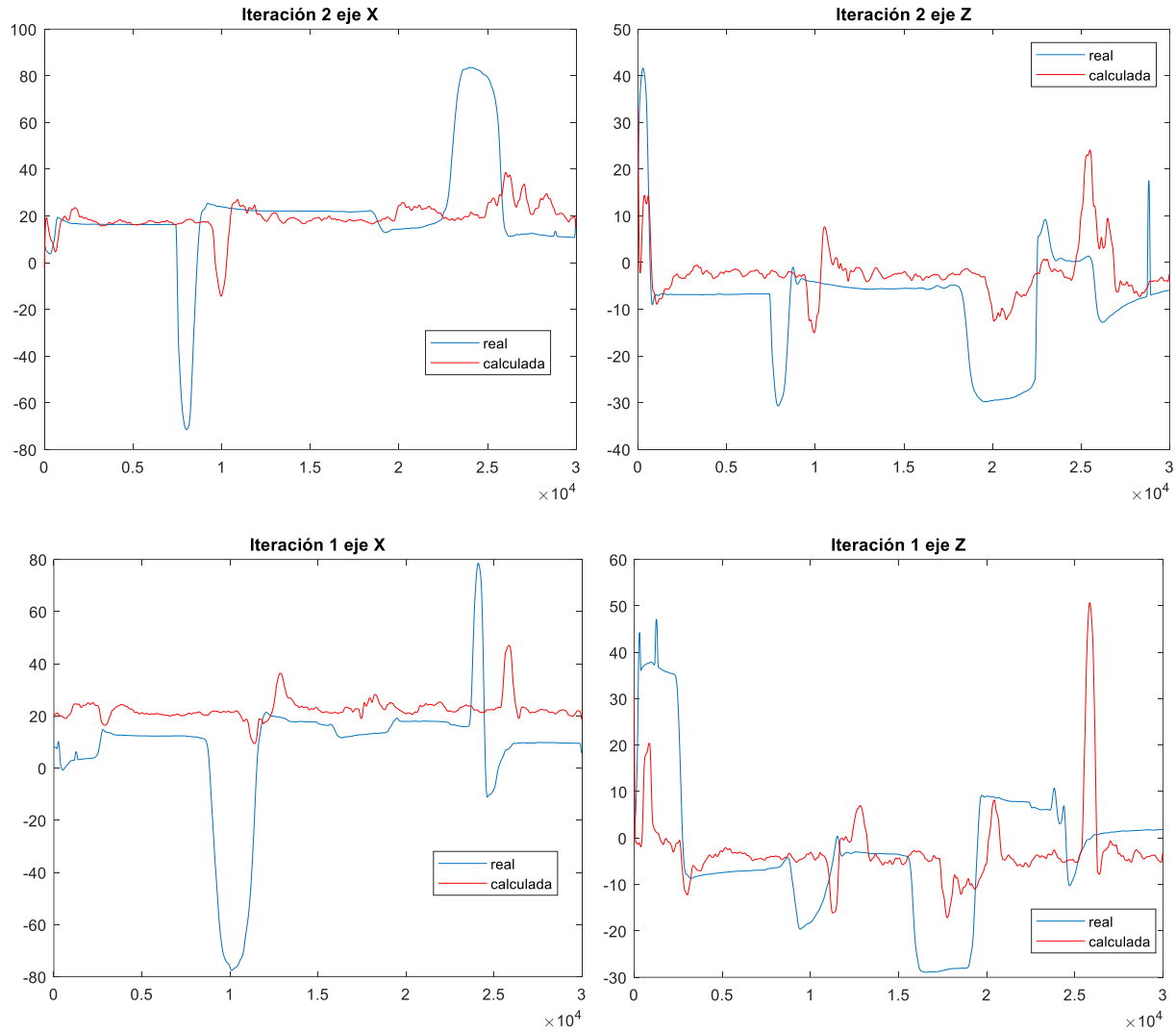


Figura 7.18 Gráfica de los valores de x y de z reales vs. valores calculados para cada iteración del Individuo 4.

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1759	0.0188	30.7086	3.0036	234.2893	-0.1571	0.1070	30.7274
2	0.1500	-0.0020	14.1038	13.5742	1.6103	-0.1520	-0.0134	14.1018
3	0.1666	-0.0230	25.7708	11.2354	32.8328	-0.1896	-0.1379	25.7478
4	0.1662	-0.0232	17.0433	8.8582	32.0133	-0.1894	-0.1393	17.0201
5	0.1795	-0.0364	18.0799	14.9626	8.0371	-0.2160	-0.2029	18.0435
6	0.1677	-0.0132	21.1413	10.3268	61.7566	-0.1808	-0.0773	21.1281

Tabla 7.13 Valores de los métodos de predicción de errores del eje x del Individuo 4.

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1949	0.0285	3.4976	3.4789	0.4731	-0.1664	0.1461	3.5261
2	0.1571	0.0170	3.2759	2.0505	33.9361	-0.1400	0.1084	3.2929
3	0.1820	-0.0075	9.1254	9.0109	0.3033	-0.1895	-0.0409	9.1179
4	0.1825	-0.0073	11.4022	5.1719	23.5646	-0.1898	-0.0401	11.3948
5	0.1516	0.0021	3.2232	3.2230	0.0042	-0.1494	0.0142	3.2253
6	0.1736	0.0066	6.1049	4.5871	11.6563	-0.1670	0.0375	6.1114

Tabla 7.14 Valores de los métodos de predicción de errores del eje z del Individuo 4.

ESTUDIO CON 2 CANALES EMG:

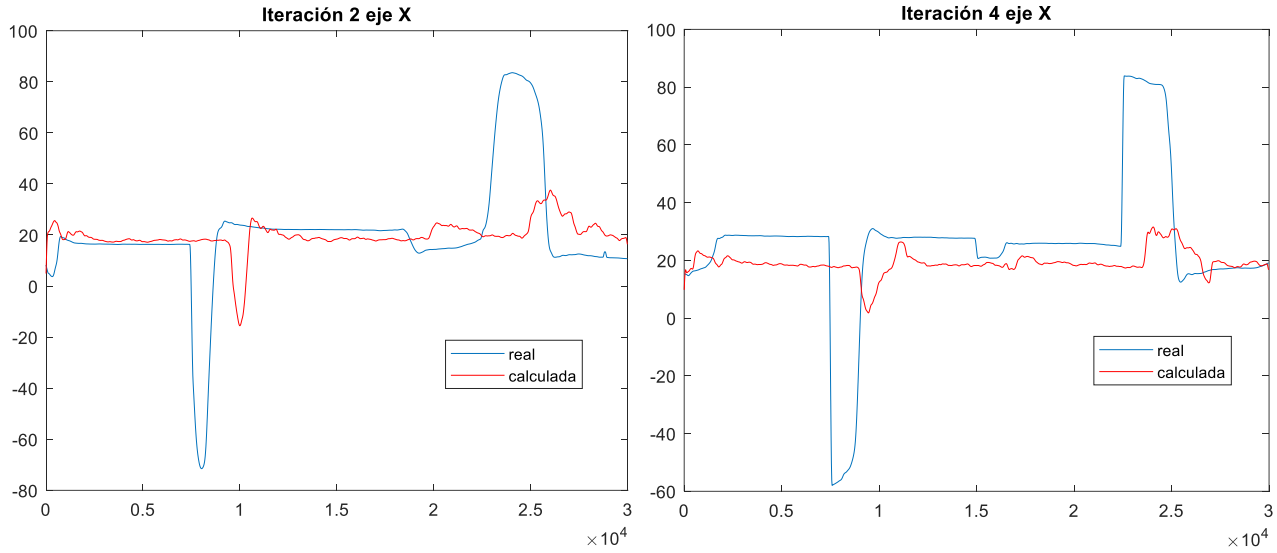


Figura 7.19 Gráfica de los valores de x reales vs. valores calculados para cada iteración del Individuo 4 (2 canales).

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1757	0.0224	37.6448	2.9411	242.3774	-0.1534	0.1273	37.6672
2	0.1483	0.0132	14.7206	14.4255	0.7890	-0.1352	0.0887	14.7338
3	0.1681	-0.0213	23.9964	10.2039	37.6034	-0.1894	-0.1268	23.9751
4	0.1678	-0.0155	33.8961	10.5098	38.3329	-0.1833	-0.0922	33.8806
5	0.1803	-0.0294	23.6288	19.3798	6.4426	-0.2098	-0.1631	23.5993
6	0.1681	-0.0061	26.7773	11.4920	65.1091	-0.1742	-0.0332	26.7712

Tabla 7.15 Valores de los métodos de predicción de errores del eje x del Individuo 4 (2 canales).

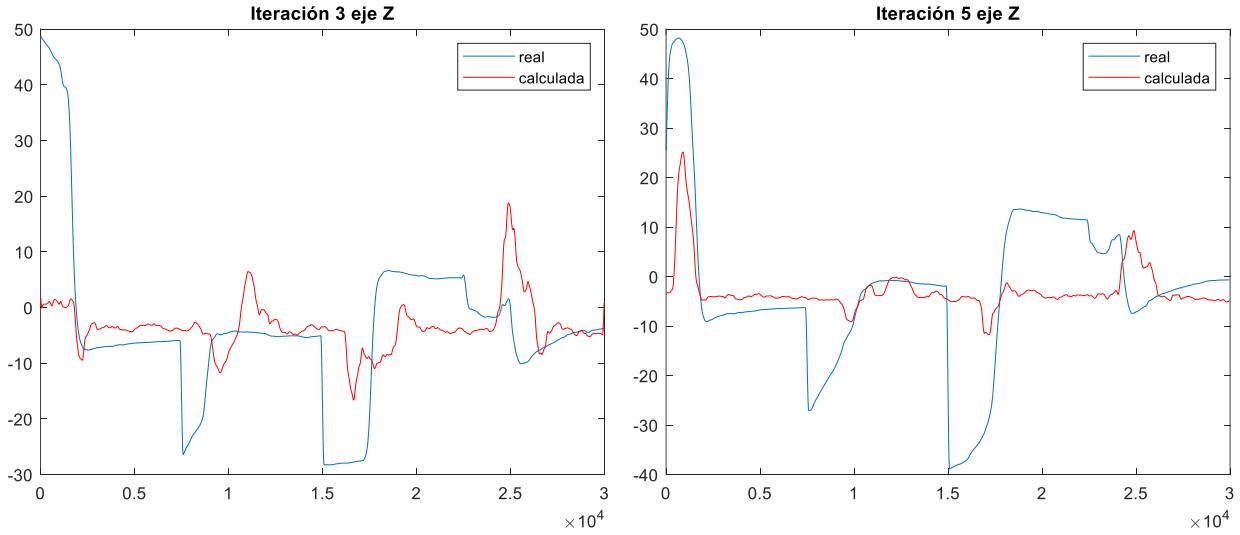
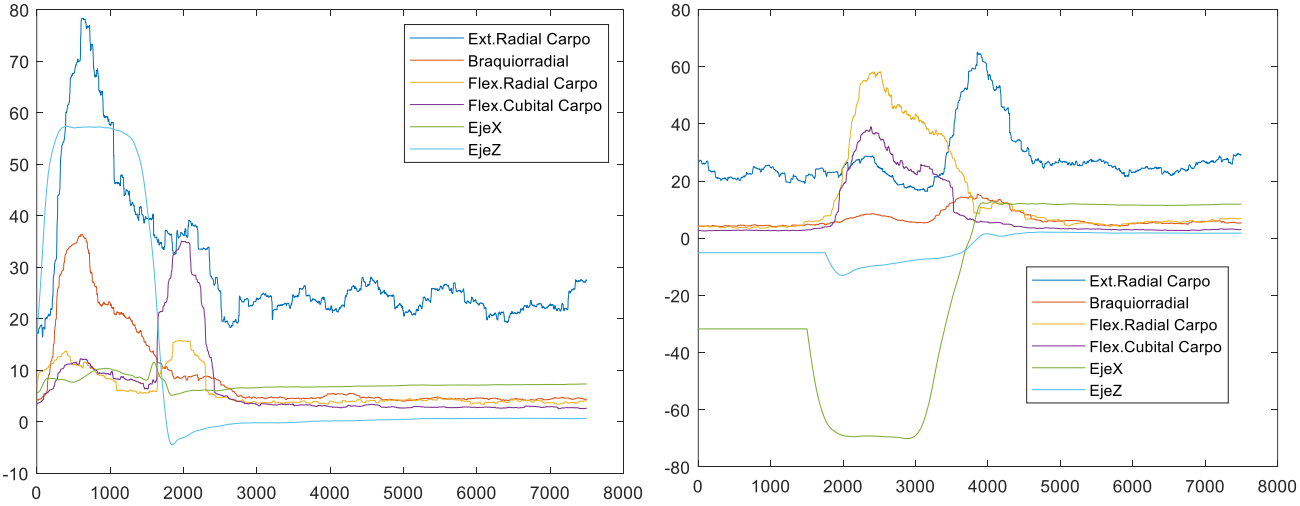


Figura 7.20 Gráfica de los valores de z reales vs. valores calculados para cada iteración del Individuo 4 (2 canales).

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1985	0.0298	4.6263	4.4993	1.7741	-0.1687	0.1500	4.6561
2	0.1678	0.0257	4.0685	2.4876	30.5531	-0.1421	0.1530	4.0942
3	0.1780	-0.0097	9.2780	9.2779	4.2607e-04	-0.1877	-0.0546	9.2683
4	0.1751	-0.0211	16.3965	7.4830	14.4920	-0.1962	-0.1207	16.3754
5	0.1619	0.0011	8.9574	8.8634	0.2650	-0.1608	0.0066	8.9585
6	0.1762	0.0051	8.6654	6.5222	9.4169	-0.1711	0.0269	8.6705

Tabla 7.16 Valores de los métodos de predicción de errores del eje z del Individuo 4 (2 canales).

7.5 Individuo 5



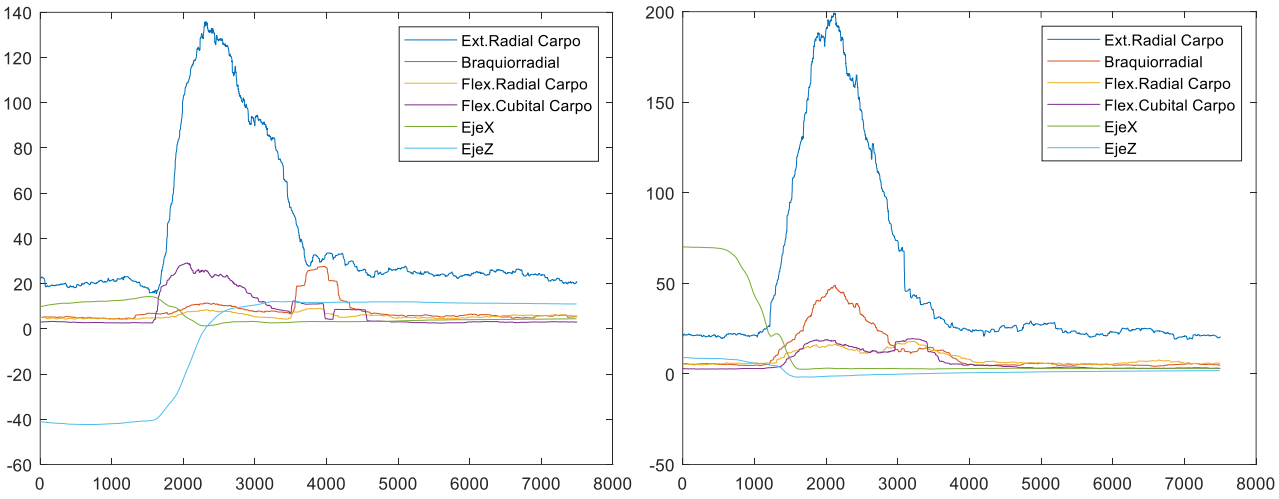


Figura 7.21 Señales de los movimientos medidos del Individuo 5.

ESTUDIO CON LOS 4 CANALES EMG:

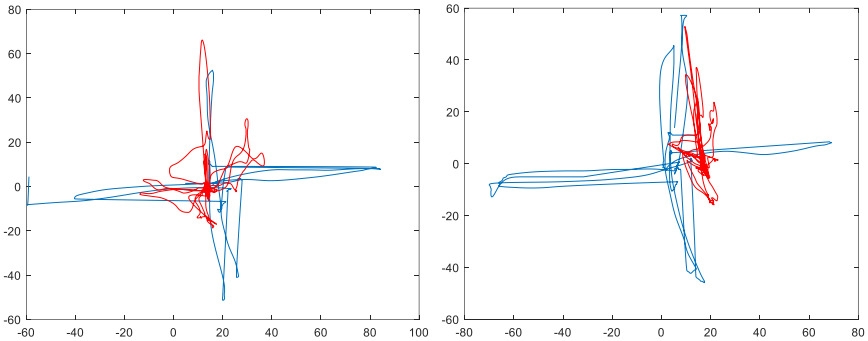
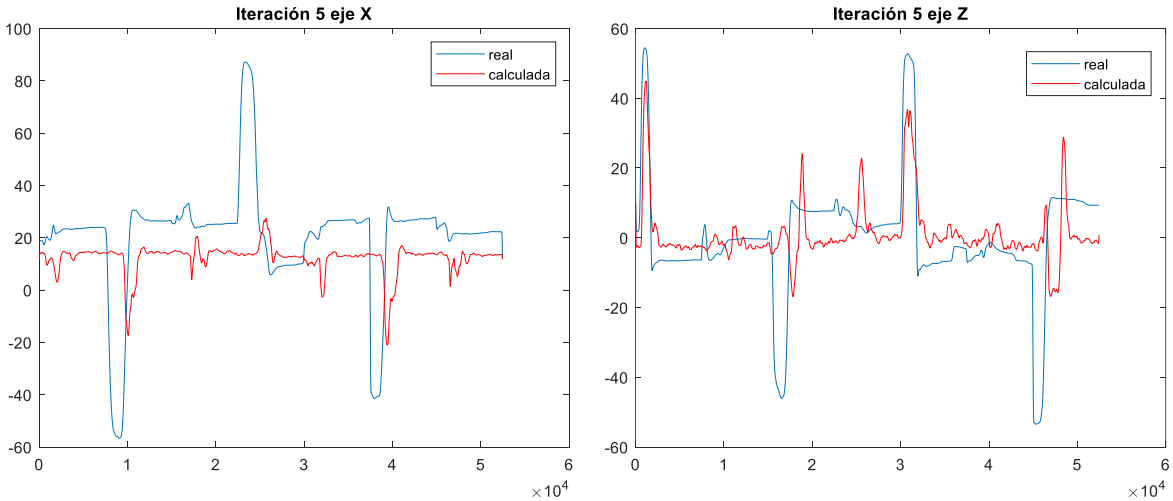
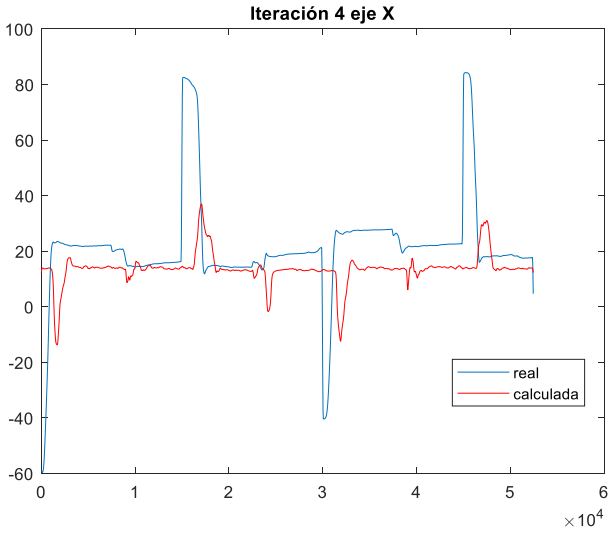
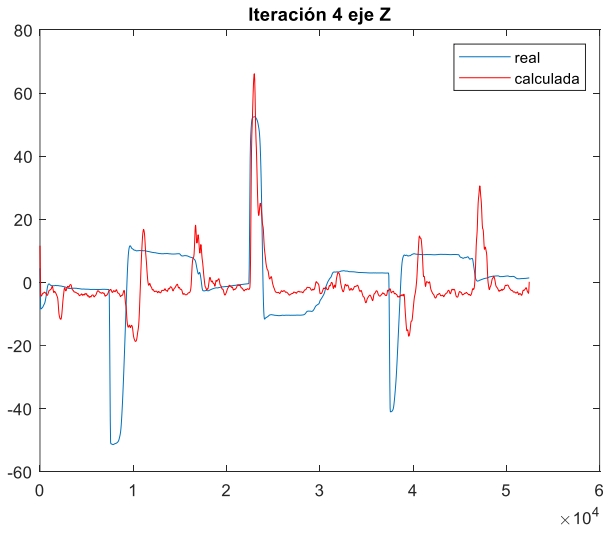
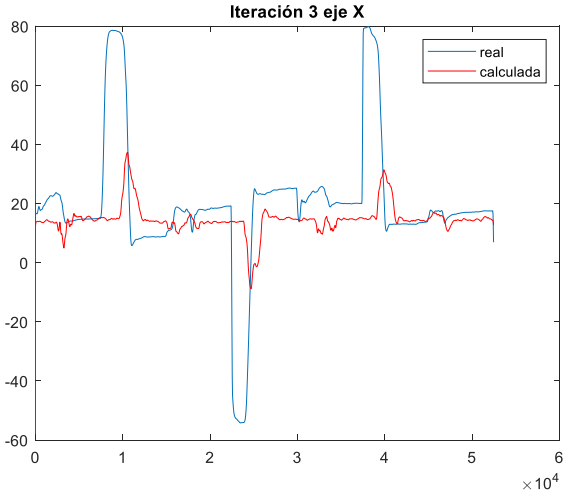


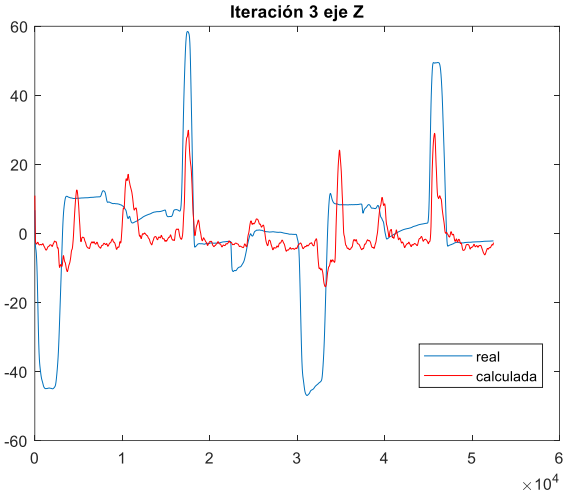
Figura 7.22 Ejemplos de trayectoria real (azul) vs. trayectoria calculada del Individuo 5.

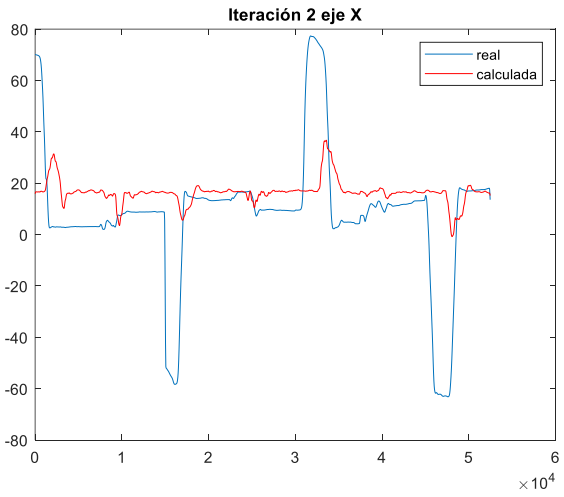


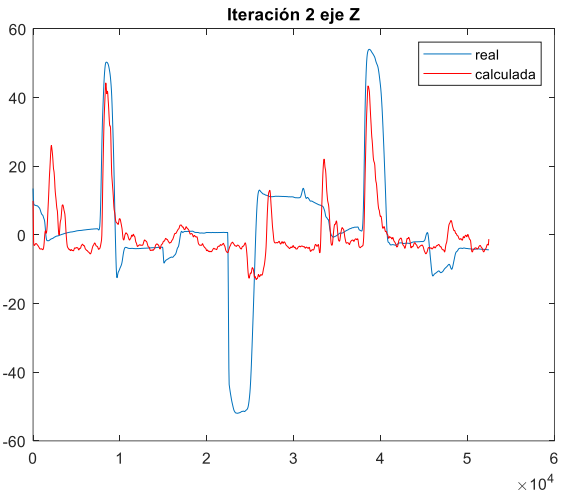

Iteración 4 eje X
real
calculada
×10^4


Iteración 4 eje Z
real
calculada
×10^4


Iteración 3 eje X
real
calculada
×10^4


Iteración 3 eje Z
real
calculada
×10^4


Iteración 2 eje X
real
calculada
×10^4


Iteración 2 eje Z
real
calculada
×10^4

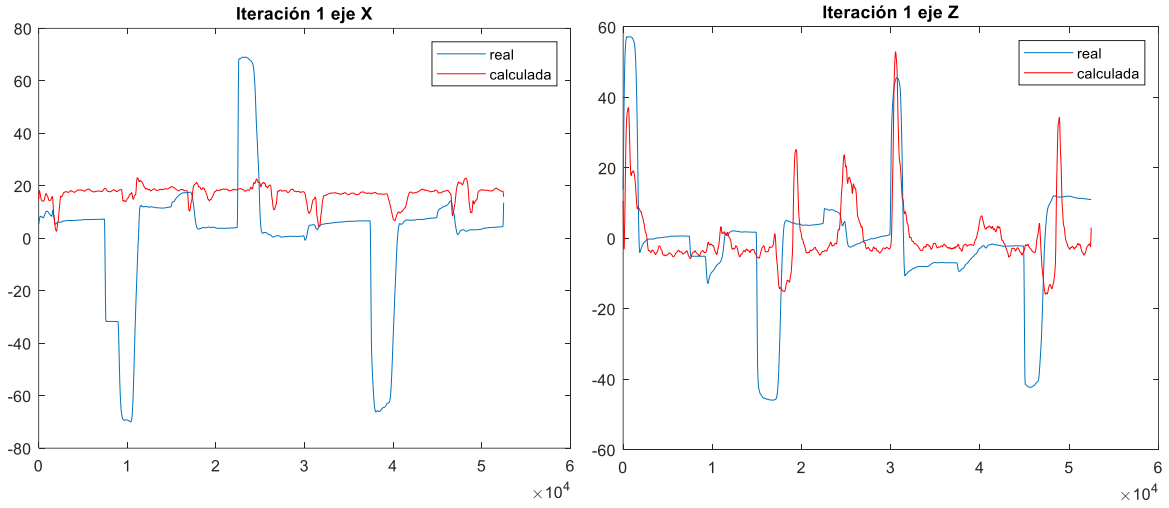


Figura 7.23 Gráfica de los valores de x y de z reales vs. valores calculados para cada iteración del Individuo 5.

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.2038	0.0229	60.6457	3.3394	230.8551	-0.1809	0.1124	60.6686
2	0.1841	0.0288	29.7274	7.8631	64.7190	-0.1553	0.1565	29.7562
3	0.1661	-0.0168	19.1036	9.2908	28.9261	-0.1828	-0.1010	19.0868
4	0.1432	0.0087	11.5846	4.4760	63.7323	-0.1345	0.0605	11.5932
5	0.1614	0.0063	14.3400	5.1811	71.3985	-0.1552	0.0389	14.3463
6	0.1717	0.0100	27.0803	6.0301	91.9262	-0.1617	0.0535	27.0902

Tabla 7.17 Valores de los métodos de predicción de errores del eje x del Individuo 5.

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1509	0.0074	2.7980	2.7576	1.9753	-0.1435	0.0491	2.8054
2	0.1358	0.0145	2.7063	2.6926	0.6170	-0.1213	0.1070	2.7209
3	0.1497	-3.4392e-04	5.9105	5.7579	1.3433	-0.1500	-0.0023	5.9101
4	0.1245	0.0092	2.1868	2.1573	1.9253	-0.1153	0.0739	2.1960
5	0.1312	5.1544e-04	2.6783	2.6602	0.8123	-0.1307	0.0039	2.6788
6	0.1384	0.0063	3.2560	3.2051	1.3346	-0.1321	0.0463	3.2622

Tabla 7.18 Valores de los métodos de predicción de errores del eje z del Individuo 5.

ESTUDIO CON 2 CANALES EMG:

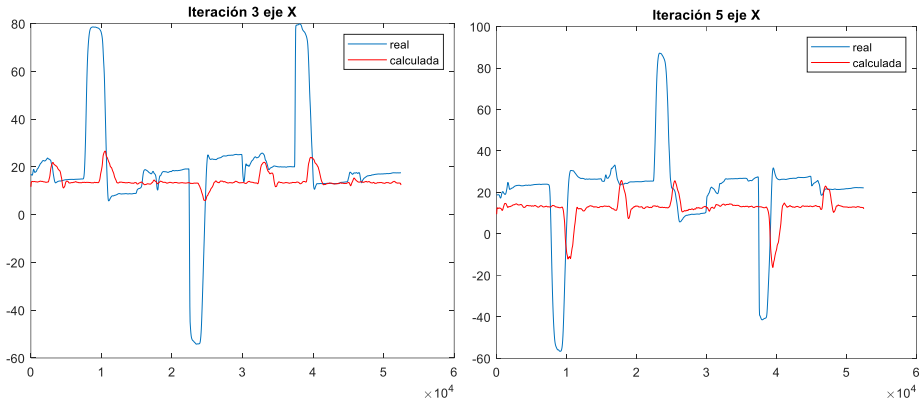


Figura 7.24 Gráfica de los valores de x reales vs. valores calculados para cada iteración del Individuo 5 (2 canales).

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.2067	0.0210	237.1623	3.1710	258.1078	-0.1857	0.1016	237.1833
2	0.1865	0.0385	86.6168	9.2043	67.4318	-0.1480	0.2065	86.6553
3	0.1679	-0.0123	56.5207	12.2206	33.0949	-0.1802	-0.0731	56.5084
4	0.1468	0.0061	13.2342	4.2495	77.2057	-0.1408	0.0413	13.2403
5	0.1612	0.0040	14.9977	5.3341	69.5073	-0.1571	0.0251	15.0018
6	0.1738	0.0115	81.7064	6.8359	101.0695	-0.1624	0.0603	81.7178

Tabla 7.19 Valores de los métodos de predicción de errores del eje x del Individuo 5 (2 canales).

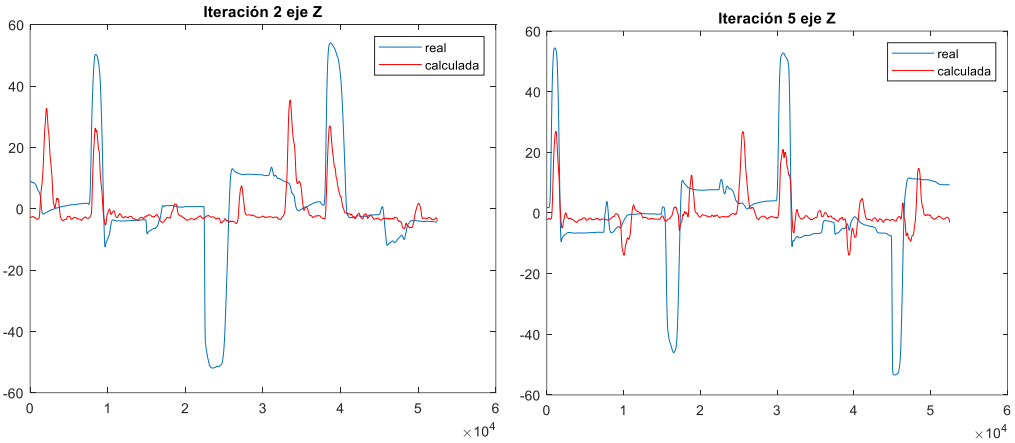


Figura 7.25 Gráfica de los valores de z reales vs. valores calculados para cada iteración del Individuo 5 (2 canales).

	1 NRMSE	2 CC	3 SNR	4 COD	5 GD	6 CC_Menos_NRMSE	7 CC_Entre_NRMSE	8 CC_Mas_SNR
1	0.1520	0.0129	7.9070	7.8407	0.3011	-0.1392	0.0847	7.9198
2	0.1482	0.0064	4.3769	4.3254	0.8695	-0.1419	0.0429	4.3833
3	0.1594	0.0181	6.0667	6.0666	4.6912e-04	-0.1414	0.1135	6.0848
4	0.1279	0.0168	2.9894	2.9750	0.4300	-0.1111	0.1313	3.0062
5	0.1346	-0.0019	6.0407	6.0265	0.0986	-0.1366	-0.0144	6.0387
6	0.1445	0.0104	5.4761	5.4468	0.3399	-0.1340	0.0716	5.4866

Tabla 7.20 Valores de los métodos de predicción de errores del eje z del Individuo 5 (2 canales).

8 CONCLUSIONES

8.1 Discusión de los resultados

En las gráficas de las señales adquiridas de los movimientos realizados por el Individuo 1 y por el Individuo 2 se puede percibir un desfase entre las medidas de las señales EMG y las del acelerómetro-giroscopio, especialmente en los movimientos de Flexión (figura de arriba a la derecha) y Aducción (abajo a la izquierda). Con respecto a las señales electromiográficas, se pude observar que en casi todas las gráficas el músculo Extensor Radial del Carpo presenta medidas de mayor magnitud que las de otros músculos. Cuando se realiza el movimiento propio que activa ese músculo (Extensión) el valor es el doble, por lo que no supone ningún problema. Además, el eje x del Individuo 1 no se ha calibrado correctamente, ya que toma su valor de origen en -20.

El Individuo 3 presenta unas gráficas con todas las señales en fase, con los picos y los valles de las señales electromiográficas y las del acelerómetro-giroscopio coincidentes. Se observa que la calibración inicial no se ha realizado bien, ya que los dos ejes tienen su valor inicial en -20. El Individuo 4 presenta un desfase en todos los movimientos y un calibrado del eje x distinto de 0, por lo que la adquisición de datos ha sido bastante mala. Además, en el movimiento de Extensión (figura de abajo a la derecha) se puede percibir una interferencia en la señal del eje z que influirá en los resultados. Por último, las señales EMG y las recogidas por el sensor de los movimientos de Aducción (figura de abajo a la izquierda) y de Extensión (abajo a la derecha) del Individuo 5 están desfasados.

Las trayectorias calculadas (rojo) resultantes de la aplicación de la regresión lineal a los datos del Individuo 1 y del 2 en 5 iteraciones presentan un error considerable con respecto a la trayectoria real (azul). A su vez, en las gráficas, se puede ver que el eje x medido por el sensor GY-521 muestra valores que no se corresponden con la amplitud normal del movimiento de la muñeca [26], tal como se muestra en la Tabla 8.1. Esto puede ser debido al cambio de posición del antebrazo o de la mano, y puede dar lugar a errores en el cálculo de los coeficientes de la regresión.

	Boone y Azen	**Ryu et al.**
Flexión	75 ± 6.6º	79º
Extensión	74 ± 6.6º	59º
Desviación radial o Abducción	21 ± 4.0º	21º
Desviación cubital o Aducción	35 ± 3.8º	38º

Tabla 8.1 Amplitud normal de movimiento de la muñeca en adultos sanos [63]

[63] Dr. Emilio Juan García [26]

En las gráficas de las trayectorias del Individuo 3 y del Individuo 5 se puede observar una similitud entre las trayectorias calculadas y las reales, aunque dependiendo de cada iteración se puede ver un desplazamiento en el eje vertical. En el Individuo 4 la trayectoria calculada y la real no se asemeja prácticamente en nada, y existe una gran presencia de ruido en la trayectoria calculada, debido a la mala adquisición de los valores.

Si la correlación presenta valores cercanos a 1 significa que los coeficientes calculados han sido bastante acertados, y el error es mínimo, logrando un modelo fiable. En cambio, si se aproxima más a 0 significa que el vector de coeficientes no se ha calculado correctamente o los valores varían considerablemente entre unos puntos y otros, por lo que es complicado hallar unos coeficientes determinados con un error reducido. En este último las predicciones no son fiables, el modelo no es representativo de la realidad, por lo que se debería emplear otro método. El coeficiente de correlación medio en el estudio está en el rango de medidas entre 0.0021 y 0.0226. La correlación depende del desfase entre las ondas, cuanto menor sea el desfase, mayor es el coeficiente. Debido a su reducido valor se realizaron los estudios a partir de otros métodos. De las tablas de los valores de los métodos de predicción de errores se pueden extraer, observando la media, es decir, la última fila, las siguientes conclusiones:

- El método NRMSE muestra un coeficiente de similitud entre las dos gráficas mayor que el método CC, con un rango que varía entre 0.1109 y 0.1998.
- El método SNR que muestra el ruido existente en la señal presenta valores fuera de su rango, ya que el resultado debería estar entre 0 y 1 y el intervalo va de 1.3052 a 27.0803. Esto significa que existe una presencia de ruido importante.
- En el artículo de Spuler et al. [25] pone que el mejor método para determinar el error entre los valores calculados y los reales cuando existen errores debido al escalado es el que suma los valores de los coeficientes de correlación y de las señales de ruido (CC+SNR). El rango en el escalado está entre 1.3278 y 27.0902.

Al medir las señales electromiográficas en un antebrazo, los músculos están muy próximos entre sí, por lo que es muy probable que las interferencias entre unas señales y otras supongan un error considerable en la aplicación del método de regresión lineal. Por ello se han estudiado los movimientos en eje x (Ext. Radial del Carpo/Flexor Radial del Carpo) y en eje z (Braquiorradial/Flexor Cubital del Carpo) por separado. Comparando las dos tablas de valores se puede decir que:

- Los coeficientes de correlación CC en el Individuo 1 y del Individuo 3 presentan una disminución de un 10-50% con respecto al análisis de los 4 canales. Sin embargo, los resultados en el Individuo 2 y en el Individuo 5 son mejores en el

análisis de un solo eje. La variación de los valores en el Individuo 4 es casi inapreciable.

- Los valores del método NRMSE son siempre mayores, por lo que las señales se asemejan más.
- El método SNR muestra una variación de ruido de un 200-600%. Esto puede ser debido a errores debido a fatiga muscular.
- Con respecto a los valores del método CC+SNR también han sufrido un aumento considerable.

Lo que se puede deducir de los resultados de los métodos de predicción de errores es que, a pesar de existir interferencia entre las señales de los músculos, éstas no influyen en los resultados finales. El mejor resultado se ha obtenido para el Individuo 3. Los posibles motivos en el error pueden ser:

- Calibración del sensor incorrecta.
- Tiempos entre señales EMG y sensor no son simultáneos.
- Crosstalk entre músculos o incorrecta colocación de los electrodos.
- Errores debidos a la fatiga del usuario.
- Colocación errónea de los electrodos.

8.2 Trabajos futuros

Debido al tiempo limitado del proyecto, han quedado algunas funciones de mejora de los resultados de las regresiones lineales sin testear. Estas funciones de Matlab son las siguientes:

- fitrsvm: aplicación del método SVM a las regresiones lineales.
- fitrlinear: modelo de regresión lineal para datos multidimensionales.
- predict: una vez obtenidos los modelos, utilizar esta función para predecir la respuesta de una regresión lineal.
- feval: similar a predict, pero se pueden añadir predictores nuevos como elementos de entrada a la función.

Una vez que la respuesta de la señal calculada presente un error de menos de un 20% con respecto a la trayectoria real, se puede proceder a la simulación del robot de la interfaz gráfica. Esto no se ha podido llevar a cabo por la diferencia considerable entre las dos trayectorias. El siguiente paso podría ser la simulación de un robot real de la trayectoria simulada en el robot de la interfaz.

Finalmente, la interfaz presenta un desplegable con las opciones offline y online. Si se selecciona online aparece un mensaje de que “Disponible próximamente”, por lo que una vez que el modo offline da un resultado correcto se podría ir calculando la trayectoria en tiempo real.

9 BIBLIOGRAFÍA

[1] J. H. Quach, «Surface Electromyography: Use, Design & Technological Overview», p. 34.

[2] L. Gila, A. Malanda, I. Rodríguez Carreño, J. Rodríguez Falces, y J. Navallas, «Métodos de procesamiento y análisis de señales electromiográficas», *An. Sist. Sanit. Navar.*, vol. 32, pp. 27-43, 2009.

[3] C. M. Durán Acevedo y A. L. Jaimes Mogollón, «Optimización y clasificación de señales EMG a través de métodos de reconocimiento de patrones», *ITECKNE*, vol. 10, n.º 1, jun. 2013.

[4] H. A. Romo, Realpe, J., y Jojoa, P., «Análisis de Señales EMG Superficiales y su Aplicación en Control de Prótesis de Mano», *Rev. Av. En Sist. E Informática*, vol. 4, n.º 1, pp. 127-136, 2007.

[5] J. V. Pinzón, R. P. Mayorga, y G. C. Hurtado, «Brazo robótico controlado por electromiografía», *Sci. Tech.*, vol. 1, n.º 52, pp. 165-173, dic. 2012.

[6] M. Sekiya y T. Tsuji, «Inverse estimation of multiple muscle activations based on linear logistic regression», en *2017 International Conference on Rehabilitation Robotics (ICORR)*, London, 2017, pp. 935-940.

[7] J. M. Hahne *et al.*, «Linear and Nonlinear Regression Techniques for Simultaneous and Proportional Myoelectric Control», *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, n.º 2, pp. 269-279, mar. 2014.

[8] L. H. Smith, T. A. Kuiken, y L. J. Hargrove, «Use of probabilistic weights to enhance linear regression myoelectric control», *J. Neural Eng.*, vol. 12, n.º 6, p. 066030, dic. 2015.

[9] P. Konrad, *The ABC of EMG: a practical introduction to kinesiological electromyography*, Version 1.4. Noraxon USA, Inc, 2006.

[10] M. A. Villarroya Aparicio, «Electromiografía cinesiológica», *Rehabilitación*, vol. 39, n.º 6, pp. 255-264, ene. 2005.

[11] Mario, «Comparativa Arduino: Arduino vs. el resto», *NeoTeo*, 04-abr-2009. [En línea]. Disponible en: http://www.neoteo.com/comparativa-arduino-arduino-vs-el-resto-15399/. [Accedido: 05-dic-2018].

[12] Noraxon U.S.A., Inc., «TeleMyo miniDTS User Manual». ago-2015.

[13] «Mini DTS», *Noraxon USA*. [En línea]. Disponible en: https://www.noraxon.com/our-products/mini-dts/. [Accedido: 05-dic-2018].

[14] «Cómo usar un acelerómetro en nuestros proyectos de Arduino», *Luis Llamas*. [En línea]. Disponible en: https://www.luisllamas.es/como-usar-un-acelerometro-arduino/. [Accedido: 07-dic-2018].

[15] «Cómo usar un giroscopio en nuestros proyectos de Arduino», *Luis Llamas*. [En línea]. Disponible en: https://www.luisllamas.es/como-usar-un-giroscopio-arduino/. [Accedido: 07-dic-2018].

[16] «El bus I2C en Arduino», *Luis Llamas*. [En línea]. Disponible en: https://www.luisllamas.es/arduino-i2c/. [Accedido: 07-dic-2018].

[17] F. Paulsen y J. Waschke, *Sobotta. Atlas de anatomía humana vol 1 (24ª ed.): Anatomía general y aparato locomotor*, Edición: 24. Barcelona: Elsevier, 2018.

[18] J. J. Villarejo Mayor, R. M. Costa, A. Frizera-Neto, y T. F. Bastos, «Decodificación de Movimientos Individuales de los Dedos y Agarre a Partir de Señales Mioeléctricas de Baja Densidad», *Rev. Iberoam. Automática E Informática Ind. RIAI*, vol. 14, n.º 2, pp. 184-192, abr. 2017.

[19] «Medir la inclinación con IMU, Arduino y filtro complementario», *Luis Llamas*. [En línea]. Disponible en: https://www.luisllamas.es/medir-la-inclinacion-imu-arduino-filtro-complementario/. [Accedido: 12-dic-2018].

[20] M. Rojas Martínez y M. Á. Mañanas Villanueva, «Electromiografía de superficie multicanal como herramienta no invasiva en la rehabilitación neuromuscular», 2012.

[21] *MPU6050 arduino processing. Contribute to HeTpro/MPU6050_arduino_processing development by creating an account on GitHub*. HeTpro, 2018.

[22] «Mínimos cuadrados ordinarios», *Wikipedia, la enciclopedia libre*, 20-abr-2018. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=M%C3%ADnimos_cuadrados_ordinarios&oldid=107161698. [Accedido: 13-dic-2018].

[23] S. H. Brown, «Multiple Linear Regression Analysis: A Matrix Approach with», p. 3.

[24] «Regresión lineal», *Wikipedia, la enciclopedia libre*, 08-oct-2018. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Regresi%C3%B3n_lineal&oldid=111139867. [Accedido: 13-dic-2018].

[25] M. Spuler, A. Sarasola-Sanz, N. Birbaumer, W. Rosenstiel, y A. Ramos-Murguialday, «Comparing metrics to evaluate performance of regression methods for decoding of neural signals», en *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Milan, 2015, pp. 1083-1086.

[26] DR. EMILIO L. JUAN GARCÍA, «Biomecánica de la Muñeca.pdf». .

[27] «MPU6050 Arduino, Acelerómetro y Giroscopio», *HETPRO/TUTORIALES*, 25-abr-2014. [En línea]. Disponible en: https://hetpro-store.com/TUTORIALES/modulo-acelerometro-y-giroscopio-mpu6050-i2c-twi/. [Accedido: 07-dic-2018].

[28] «Determinar la orientación con Arduino y el IMU MPU-6050», *Luis Llamas*. [En línea]. Disponible en: https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/. [Accedido: 07-dic-2018].

[29] «Arduino Playground - MPU-6050». [En línea]. Disponible en: https://playground.arduino.cc/Main/MPU-6050. [Accedido: 07-dic-2018].

[30] «Tutorial MPU6050, Acelerómetro y Giroscopio». [En línea]. Disponible en: https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html. [Accedido: 07-dic-2018].

[31] D. O. B. Guerrero, «MANUAL DE INTERFAZ GRÁFICA DE USUARIO EN MATLAB». .

[32] «Corcuera - Creación de interfaces de usuario con MATLAB.pdf». .

[33] P. Corcuera, «Creación de interfaces de usuario con MATLAB». .

[34] «Change background color in pushbutton [Matlab-GUI] - MATLAB Answers - MATLAB Central». [En línea]. Disponible en: https://es.mathworks.com/matlabcentral/answers/157445-change-background-color-in-pushbutton-matlab-gui. [Accedido: 16-dic-2018].