Aplicaciones de la Automática en Biomedicina - G21

Informe 2 – Biomecánica del brazo humano

Fátima Zohra El Yacoubi Benabbou (100454677)

Jorge Viñuela Pérez (100429688)

Índice

Introducción

La unidad fundamental del control neuronal del proceso de contracción muscular es la unidad motora. Está compuesta por una única neurona motora, sus dendritas, las ramificaciones de su axón y las fibras musculares que inerva. La cantidad de fibras varía según el músculo, desde pocos para movimientos precisos hasta millas para grandes músculos.

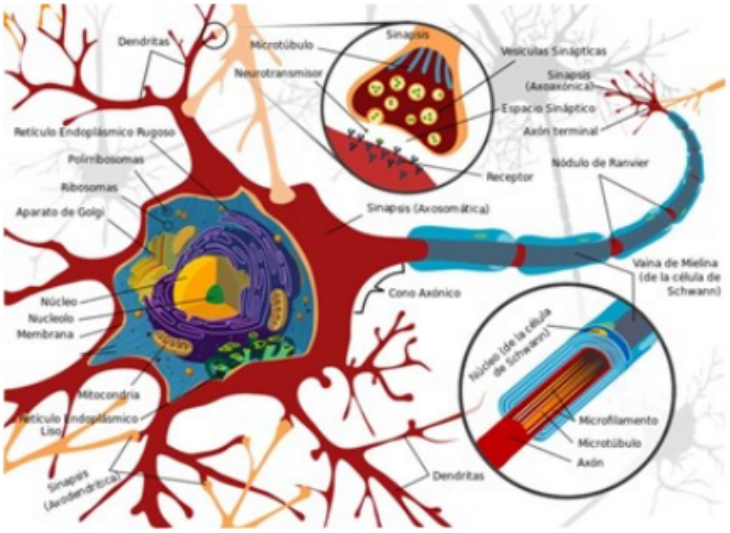

Figura 1. Diagrama esquemático de una neurona común

Las neuronas motoras transportan impulsos eléctricos desde el cerebro y la médula espinal hasta el músculo, utilizando reacciones químicas en las sinapsis neuromusculares. Cuando una neurona motora se activa, genera una pequeña señal eléctrica llamada potencial de acción de la unidad motora (MUAP), que es el componente fundamental de la señal mioeléctrica.

El MUAP es el resultado de la despolarización y repolarización de las membranas de las fibras musculares. Se expande por la fibra muscular a través de un sistema tubular, liberando iones de calcio que provocan la contracción muscular.

Para mantener la contracción, el sistema nervioso envía continuamente señales de activación a las unidades motoras, generando una secuencia de MUAP llamada tren de potenciales de acción de la unidad motora (MUAPT). La señal mioeléctrica es la superposición de todos los MUAPT de las unidades motoras bajo un electrodo.

La magnitud y densidad de la señal mioeléctrica dependen del reclutamiento de unidades motoras y de su frecuencia de activación. Estos dos mecanismos controlan la contracción muscular y la fuerza ejercida por los músculos.

La señal mioeléctrica en bruto tiene una naturaleza aleatoria, ya que el conjunto de unidades motoras reclutadas y sus tasas de activación varían constantemente. La señal tiene un rango de amplitud de -5 a 5 mV, una frecuencia de 0 a 500 Hz, y la energía dominante se encuentra entre 50 y 150 Hz.

Para medir la señal mioeléctrica se utilizan electrodos, que pueden ser invasivos o no invasivos. Los electrodos invasivos se insertan en los músculos y detectan la señal directamente del sistema nervioso, proporcionando señales de alta calidad y permitiendo detectar MUAP individuales.

Los electrodos no invasivos se colocan sobre la piel y detectan la señal mioeléctrica superficial. No permiten detectar MUAP individuales, sino la suma de todos los MUAPT en la zona debajo del electrodo.

Primera parte

Para la realización de esta práctica usaremos Matlab Simulink. Para ello primero se debe realizar el filtrado de la señal que se realiza en el programa emg_sim.mdl. El objetivo es eliminar artefactos de la señal EMG (descargada de aula global) para lo cual se utiliza un filtro paso banda que se programa con la ayuda de un bloque Digital Filter Design.

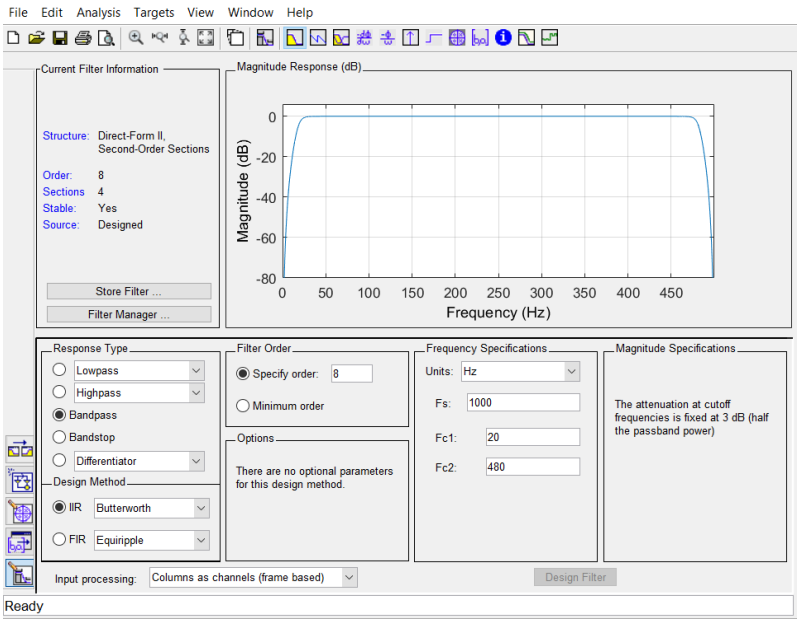

Figura 2. Filtro paso banda

Se eligió una frecuencia de muestreo de 1000 Hz para capturar adecuadamente las frecuencias de interés en la señal EMG, lo que significa que se están tomando 1000 muestras por segundo. Asimismo, se estableció una frecuencia de corte inferior (Fc1) de 20 Hz para permitir el paso de las componentes de baja frecuencia asociadas con la actividad muscular real, mientras que la frecuencia de corte superior (Fc2) se fijó en 480 Hz para permitir el paso de componentes de alta frecuencia que podrían representar artefactos de interferencia eléctrica. Estas configuraciones buscan conservar la información relevante de la señal EMG mientras se eliminan artefactos no deseados y se asegura una representación precisa de la actividad muscular.

A continuación se aplica a la señal filtrada para obtener el valor absoluto de la señal. Se puede utilizar la función de valor absoluto en Matlab/Simulink.

Después, para el suavizado se aplica un filtro Butterworth paso bajo a la señal rectificada para eliminar el ruido de alta frecuencia. La frecuencia de muestreo se mantuvo en 1000 Hz, coincidiendo con la misma frecuencia de muestreo utilizada en el primer filtro para mantener la consistencia en el proceso de filtrado. Respecto a la frecuencia de corte (Fc) del, se fijó

en 6 Hz. Esta elección se fundamenta en su capacidad para eliminar el ruido de alta frecuencia presente en la señal EMG rectificada, al tiempo que preserva las características esenciales de la señal relacionadas con la actividad muscular. Así, se logra suavizar la señal de manera efectiva, facilitando su análisis posterior con una reducción significativa del ruido.

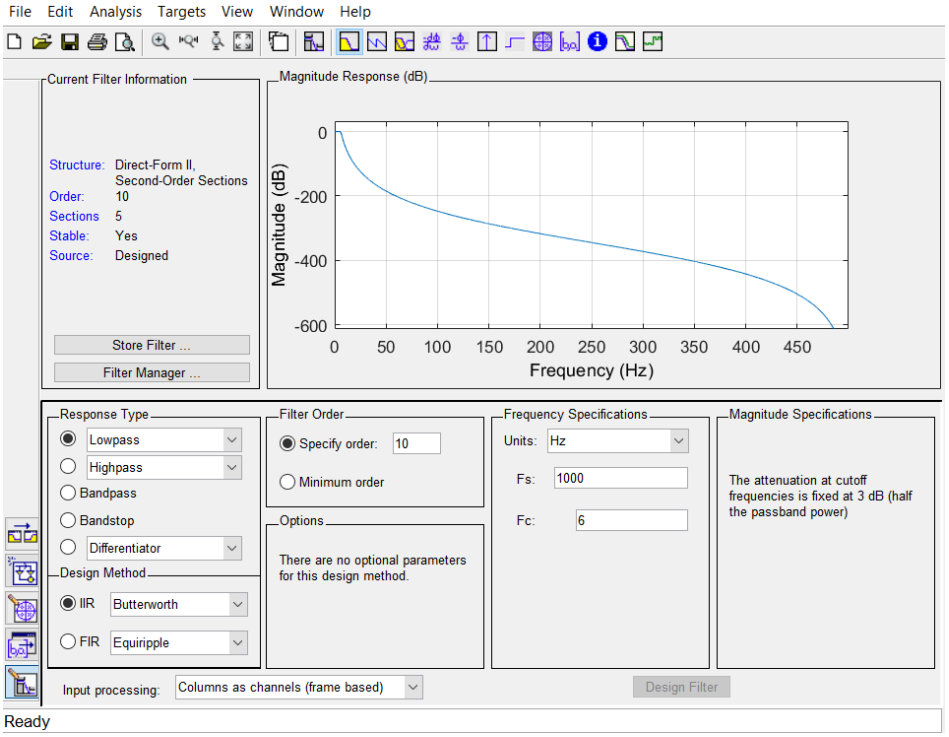

Figura 3. Filtro Butterworth

Una vez se han aplicado a nuestra señal ambos filtros, se obtiene una señal sin ruidos, de la siguiente manera.

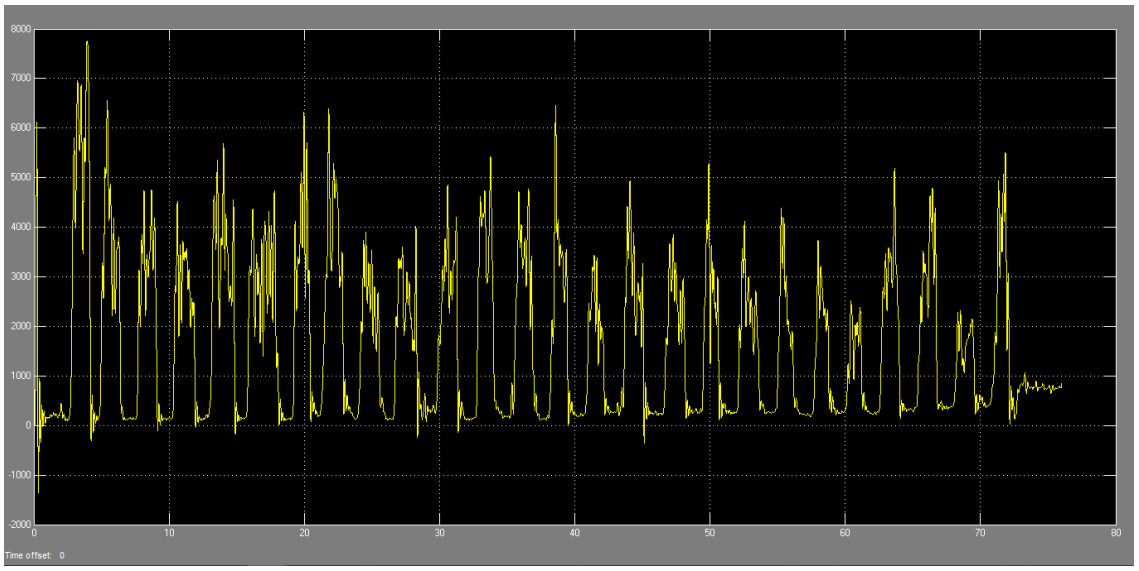

Figura 4. Señal EMG filtrada

Después de aplicar los filtros necesarios para procesar la señal EMG, se procede a normalizar la señal utilizando la ecuación proporcionada más adelante. Este proceso de normalización es fundamental para ajustar la amplitud de la señal dentro de un rango específico, lo que facilita la comparación entre diferentes señales o condiciones. En este caso, se hace referencia a un periodo inicial de 20 segundos, durante el cual se obtiene la contracción máxima del músculo (MVC).

Para calcular la normalización, se utiliza la señal EMG filtrada, o rectificada, y se determinan los valores mínimo y máximo de esta señal durante esos 20 segundos. Estos valores se emplean en la ecuación de normalización para ajustar la señal EMG actual y asegurar que esté dentro del rango deseado.

$$E_{norm} = (E_{act} - E_{min})/(E_{max} - E_{min}) \qquad (1)$$

Donde E_{act} es la señal rectificada, E_{min} es el valor mínimo de la misma en los primeros 20 segundos, E_{max} el valor máximo y E_{norm} la señal normalizada. Para implementar este proceso en Matlab, se utilizan bloques digitales de reloj para medir el tiempo y bloques de memoria para almacenar el valor anterior de la señal durante el proceso de normalización, así como un bloque de Matlab Function para introducir la expresión (1).

```
function [Enorm,Emax] = fcn(t,Emg,Emaxdt)
%#codegen
if t<20
    Enorm=0;
    if Emg>Emaxdt;
        Emax=Emg;
    else
        Emax=Emaxdt;
    end
else
    Emax=Emaxdt;
    Enorm=abs(Emg/Emaxdt);
end
```

Figura 5. Código para la normalización

De esta forma, se obtiene una señal EMG que varía entre 0 y 1, por lo que es mucho más manejable que la señal original con la que tratábamos, de manera que será más fácil utilizarla para nuestro objetivo.

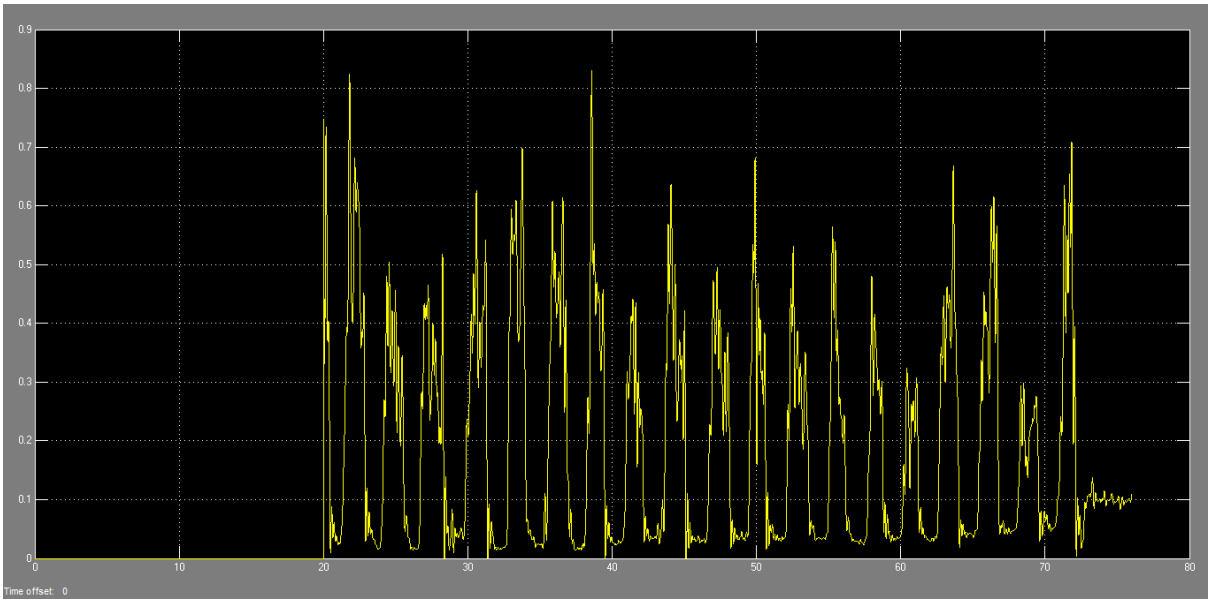

Figura 6. Señal EMG Normalizada

Una vez se ha obtenido esta señal normalizada, el próximo paso es transformarla para cumplir nuestro objetivo de mover un servo a partir de la misma. Utilizando una vez más el bloque de Matlab Function, se establecen unos umbrales para conseguir una señal cuadrada, que sea o bien su valor máximo (1) o su valor mínimo (0). Si la señal tiene su valor máximo, el servo se moverá 180º, si por el contrario tiene su valor mínimo volverá a su valor inicial.

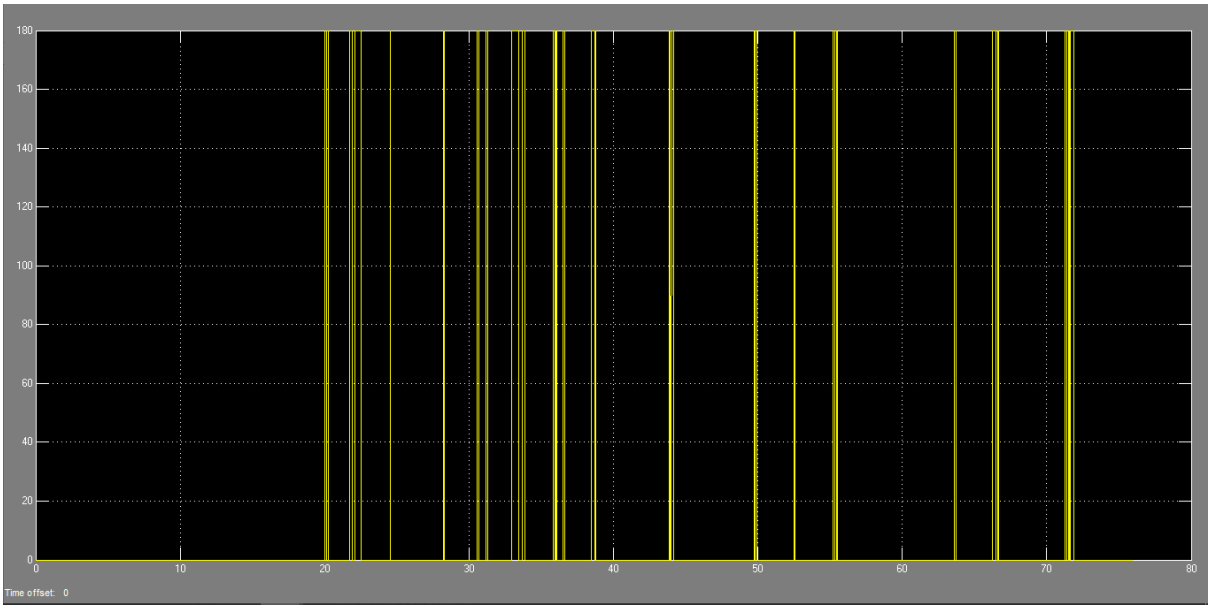

Figura 7. Señal EMG de entrada al servo

La figura muestra claramente cómo, a partir de los 20 segundos, la forma de la señal cambia a una señal cuadrada, tal como se explicó anteriormente. Se estableció un umbral en 0.5; esto significa que si la señal supera este valor en cualquier momento, su valor se

convierte en 1, lo que representa 180 grados, mientras que si no lo supera, su valor es 0 y, por lo tanto, el servo vuelve a su posición inicial.

Por último, se muestra el esquema de toda esta primera parte de la práctica.

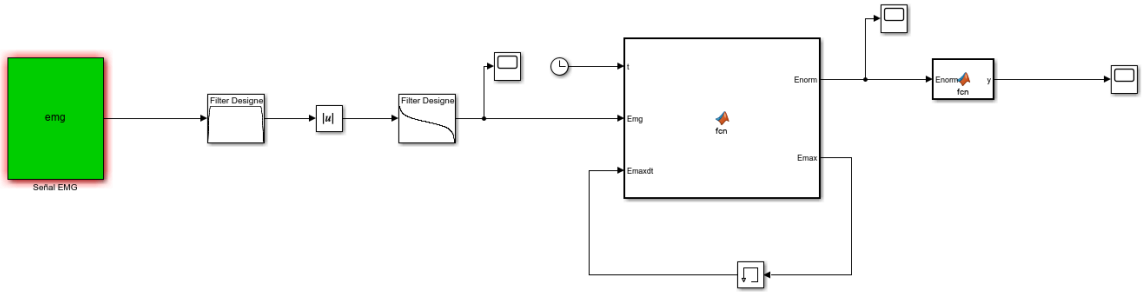

Figura 8. Esquema de procesamiento de las señales EMG

Segunda parte

En esta práctica, se lleva a cabo el proceso de programación del kit STM32F4DISCOVERY desde Matlab/Simulink con el fin de controlar un LED y un servomotor. Este kit consiste en una placa de desarrollo de microcontroladores que se basa en el microcontrolador STM32F407VGT6. Para facilitar este proceso, se emplean las librerías STM32F4 & NRF51 Basic Blockset y STM32F4 V3.0 Beyond Control. Estas librerías proporcionan una serie de bloques básicos y avanzados que permiten la interacción con los periféricos y componentes de la placa, simplificando así la programación y el control de los dispositivos conectados. El uso de Matlab/Simulink junto con estas librerías, facilita la configuración y el control de los dispositivos.

Para la programación del kit STM32F4DISCOVERY conectamos el kit al ordenador mediante los puertos USB Mini USB y Micro USB. Descargamos y extraemos los archivos de la carpeta USB test de Aula Global. Abrimos Matlab e introducimos todos los archivos en el mismo directorio, para que así el programa pueda ejecutarse correctamente.

El siguiente paso es identificar los dos programas descargados:

- **test.mdl:** programa que se compila y carga en el kit de desarrollo.
- **test_host.mdl**: programa que se ejecuta en el ordenador para la adquisición de datos.

Abrimos el programa test.mdl e identificamos los bloques que lo componen. Debemos destacar dos bloques muy importantes en este primer programa: el USB VCP Receiver STM32F4 y el USB VCP Send STM32F4. Estos bloques reciben y envían datos al microprocesador, respectivamente. Actualizamos el modelo y generamos y cargamos el código en el kit de desarrollo. Abrimos el programa test_host.mdl para verificar la transmisión de datos entre la PC y el dicho kit de desarrollo.

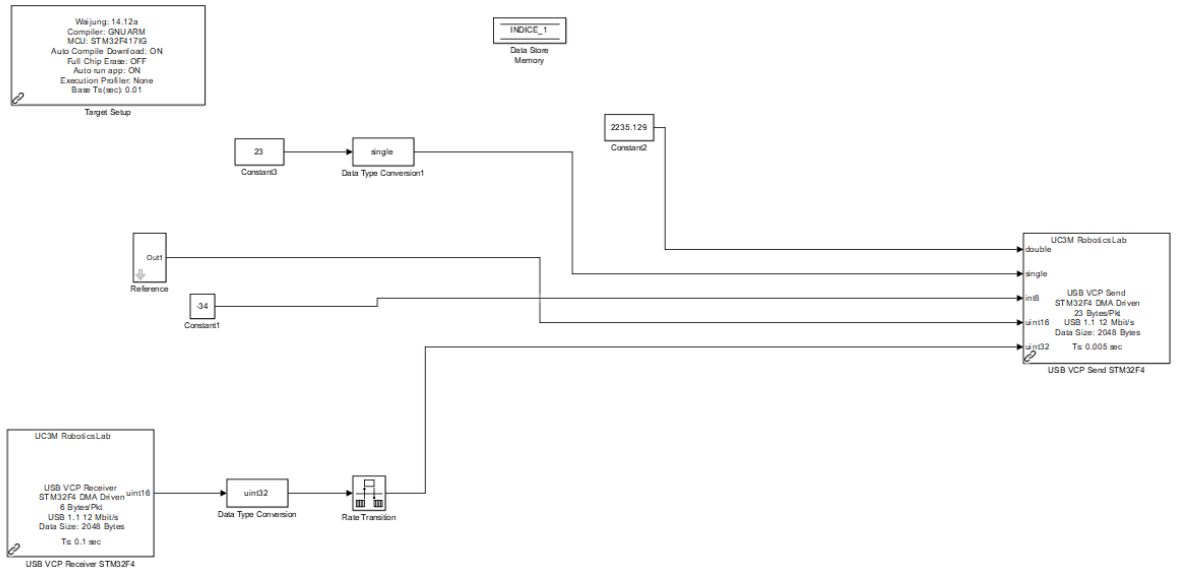

Figura 9. Test.mdl inicial

Al configurar el puerto COM en los bloques Host Serial Setup, Host Serial Rx y Host Serial Tx en función del puerto COM, que va a ser donde se detecta la tarjeta en el ordenador, tendremos que ejecutar el programa test_host.mdl para visualizar los datos.

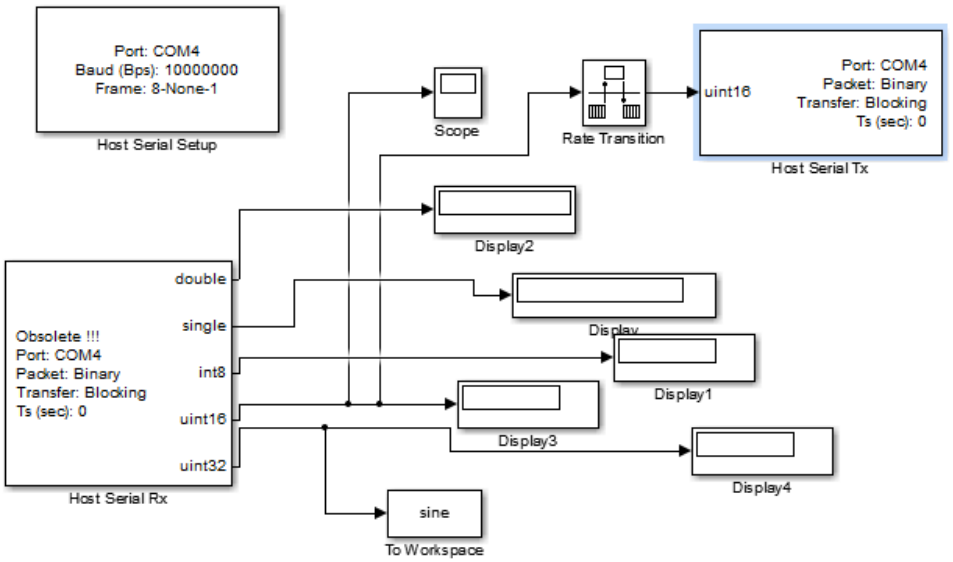

Figura 10. Test_host.mdl inicial

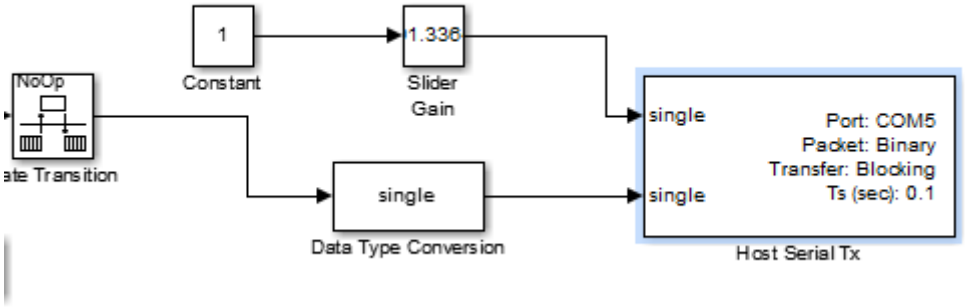

Figura 11. Test host configurando

Modificamos el programa para controlar el LED, para ello agregamos un bloque Constant al programa test.mdl en el cual establecemos un valor variable entre 0 y 255 (0% a 100% de brillo). Además, agregamos un bloque Slider Gain al programa para poder variar dicho valor de forma intuitiva. Después, conectamos el bloque Slider Gain al bloque de Host Serial Tx, el cuál envía la información al bloque de USB VCP Receiver STM32F4 explicado anteriormente. De esta forma, se envía la información al microprocesador, concretamente a un puerto PWM o analógico, consultando el archivo pin_out.pdf de Aula Global para realizar las conexiones.

Una vez que compilamos y cargamos el código modificado en el kit de desarrollo, se ejecuta el programa test_host.mdl que utiliza el deslizador del bloque Slider Gain para variar la intensidad de la luz del LED.

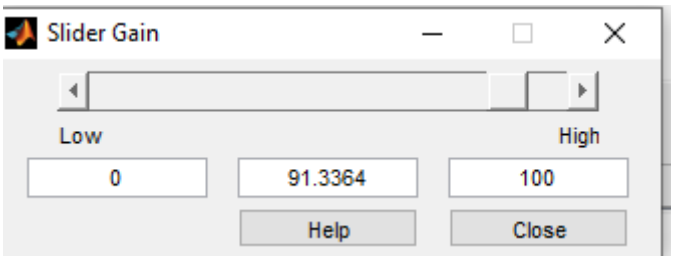

Figura 12. Slider Gain variando intensidad LED

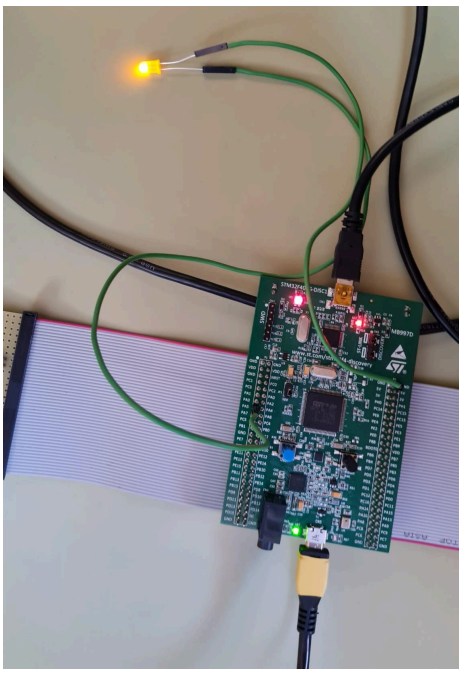

Figura 13. LED encendido

Para el control del servomotor FUTABA S3003 agregamos un bloque UC3M Basic PWM al programa test.mdl, en el que estableceremos el tiempo de muestreo de 20 ms.

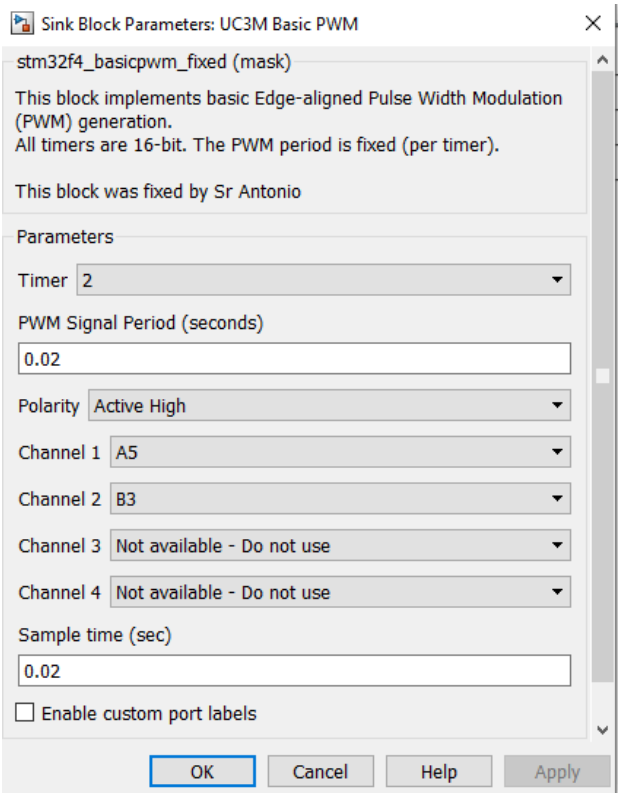

Figura 14. Puerto PWM

En la fase subsiguiente, calculamos el ciclo de trabajo en porcentaje (%) para el ángulo deseado del servomotor, para lo cual utilizaremos la siguiente ecuación:

$$y = m + \left(\frac{(p-m)}{ran \quad angulo}\right) \times u \qquad (2)$$

Y finalmente, conectamos el bloque UC3M Basic PWM a un puerto PWM del kit de desarrollo y el servomotor al mismo puerto.

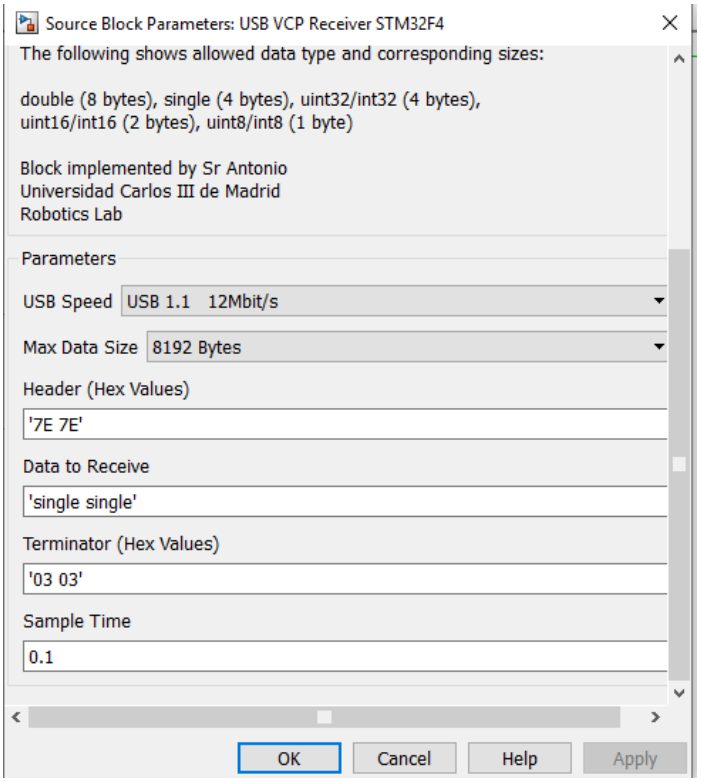

Figura 15. Receiver STM32F4

Para facilitar este proceso, se emplean las librerías STM32F4 & NRF51 Basic Blockset y STM32F4 V3.0 Beyond Control.

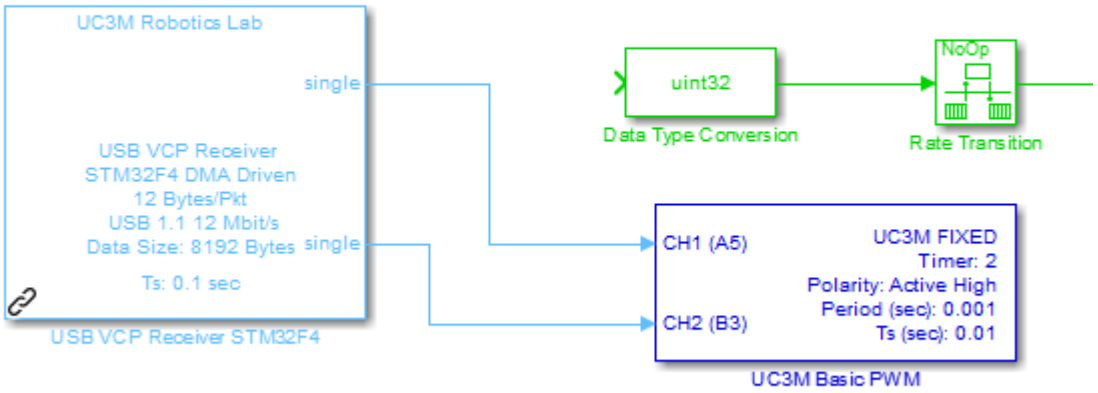

Figura 16. Envío de datos al kit de desarrollo

Una vez se haya compilado, tendremos que cargar el código modificado en el kit de desarrollo. El cual ejecutará el programa test_host.mdl, donde se introduce el ángulo deseado en el bloque Constant y observamos el movimiento del servomotor.

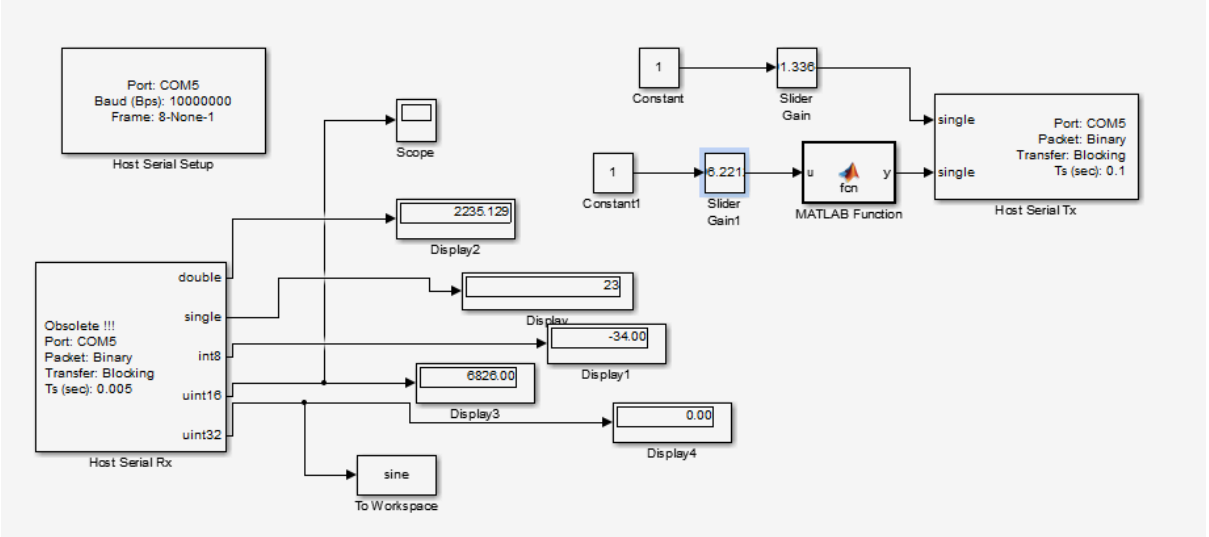

Figura 17. Test_host.mdl final

Tercera parte

Para la última parte se describe el procesamiento de señales EMG reales para controlar un servomotor y un LED. Dichas señales EMG se adquieren mediante electrodos superficiales colocados en los músculos Flexor Digitorum Superficales (FDS) y Flexor Carpi Radialis (FCR), mostrados en la siguiente figura.

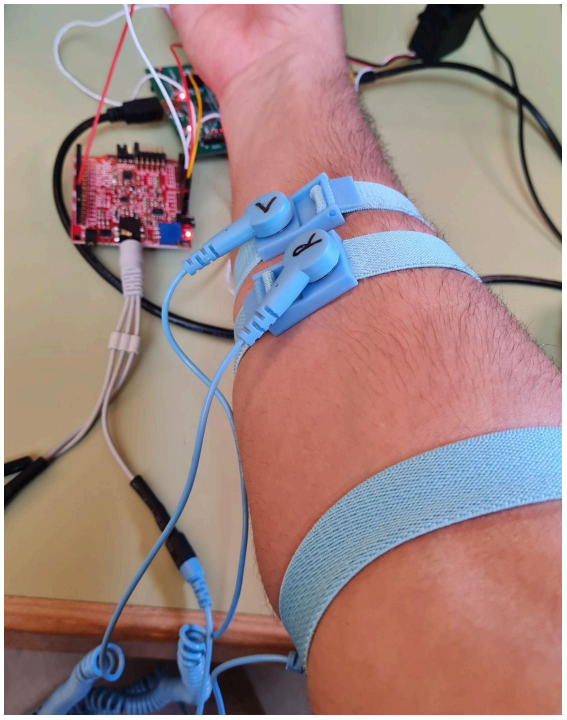

Figura 18. Conexionado de los electrodos

Al igual que en la primera parte de esta misma práctica, aunque ahora con una señal EMG real, se rectifica y normaliza la señal con el mismo objetivo: introducirla al servo. Para que el ordenador sea capaz de recibir la señal, es necesario, igual que en la segunda parte, conectar el microprocesador a los puertos miniUSB y microUSB del mismo. A su vez, para que el mismo micro sea capaz de recibir las señales desde los electrodos, hay que conectar el EMG-SHIELD-EKG-EMG al kit de desarrollo, conectando los pines de GND y de tensión de ambas placas entre ellos.

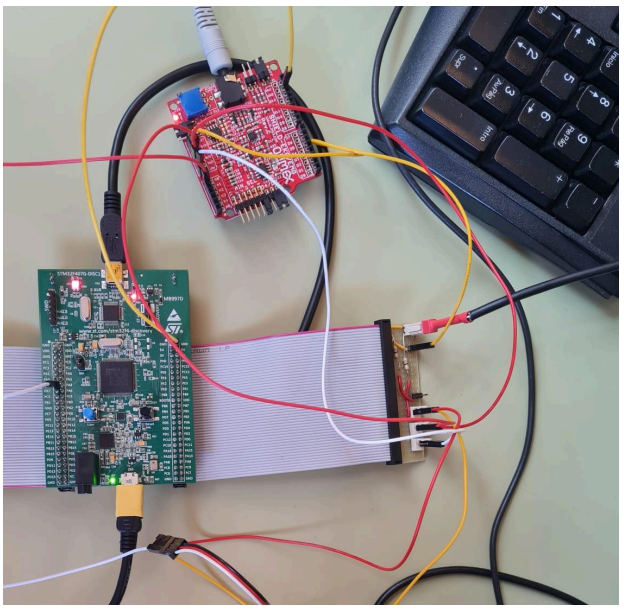

Figura 19. Conexión entre el kit de desarrollo y EMG-SHIELD-EKG-EMG

En esta tercera parte del proyecto, se modificará el programa test.mdl para adquirir datos del circuito de electromiografía y enviarlos a través del USB a la interfaz. Para ello, se modifica el bloque del conversor ADC para que sea el puerto PA4 el que reciba la señal EMG, para su posterior rectificación.

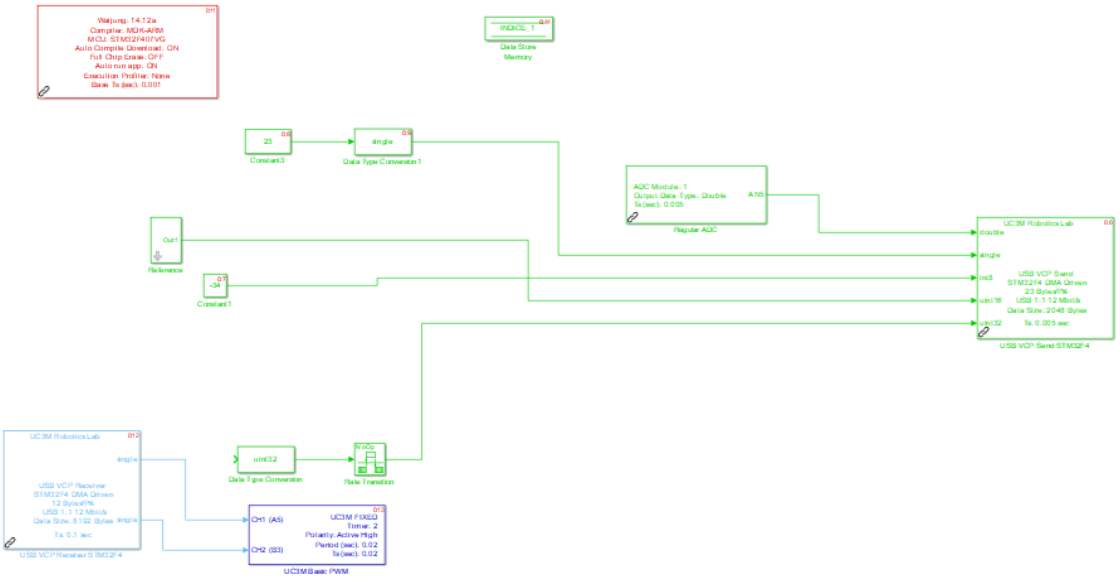

Figura 20. programa test.mdl modificado

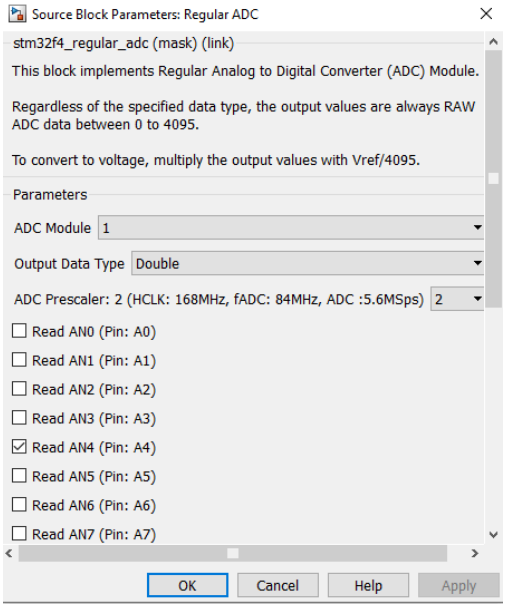

Figura 21. Parámetros del bloque Regular ADC

Una vez la señal se ha recibido en el programa test_host.mdl, se normaliza de la misma manera que en la primera parte. Hemos decidido ir observando como evoluciona a través de los distintos procesos que sufre la misma. Primero, analizamos solamente la señal cruda, accionando los músculos del brazo para observar cómo varía la señal en base a estos impulsos.

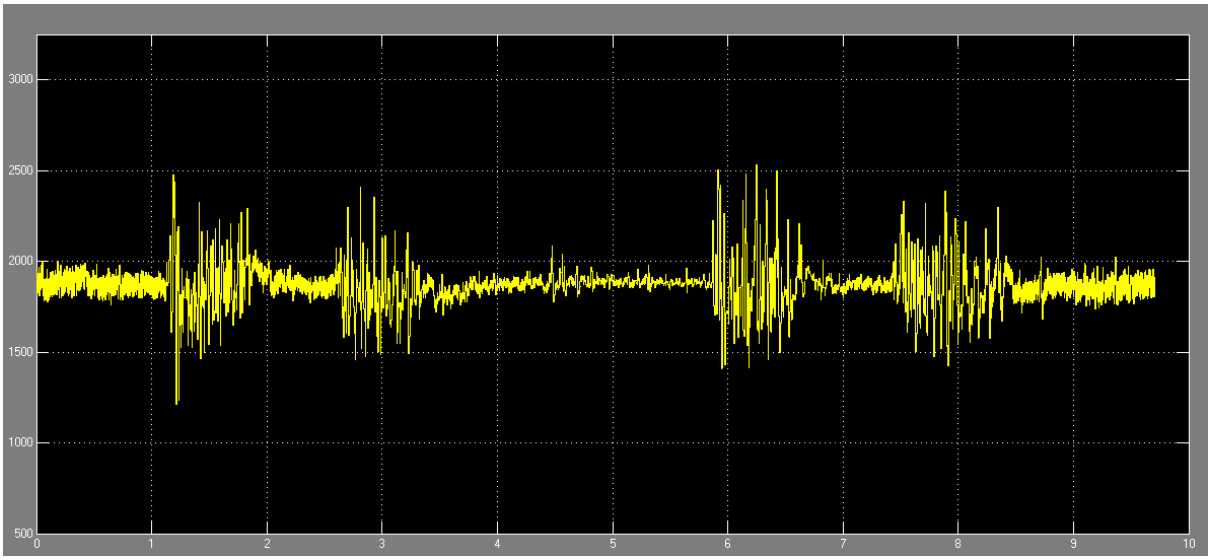

Figura 22. Señal EMG cruda

Como se observa en la figura X, al apretar el músculo se genera una variación en la señal, lo que nos lleva a pensar que efectivamente, la señal se está recibiendo de buena manera. Sin embargo, se observa bastante ruido, por lo que es necesario filtrar la señal, como se indicó en apartados anteriores.

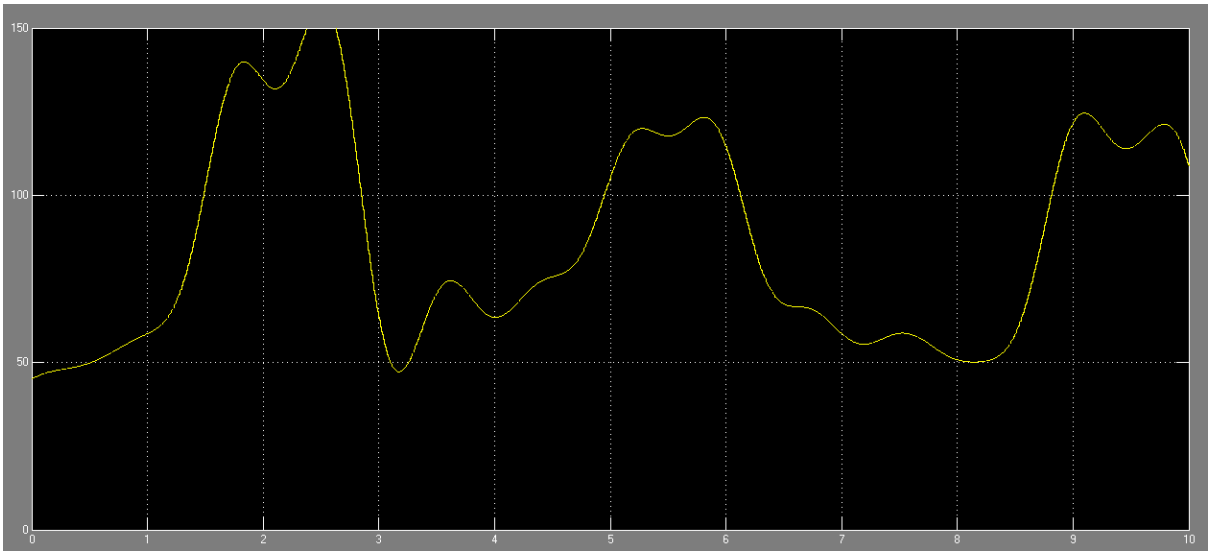

Figura 23. Señal EMG filtrada

Cabe destacar que cada gráfica se basa en impulsos eléctricos distintos, dado que para poder capturarlas era necesario ir tensando el músculo en cada “scope”. En esta figura se observa como, cuando se genera tensión en el músculo la señal aumenta su valor, dependiendo de cuánta tensión se genere en dicho músculo.

Con esto, ya se consigue lo mismo que se tenía en la primera parte de esta práctica, una señal EMG filtrada, aunque esta no se trata de una simulación. Al igual que en esa primera parte, el siguiente paso es normalizar la señal y convertirla a una señal cuadrada que varíe entre 180 y 0, con el objetivo de mover el servo conectado en el pin PA5 del microprocesador, como se muestra en la siguiente figura.

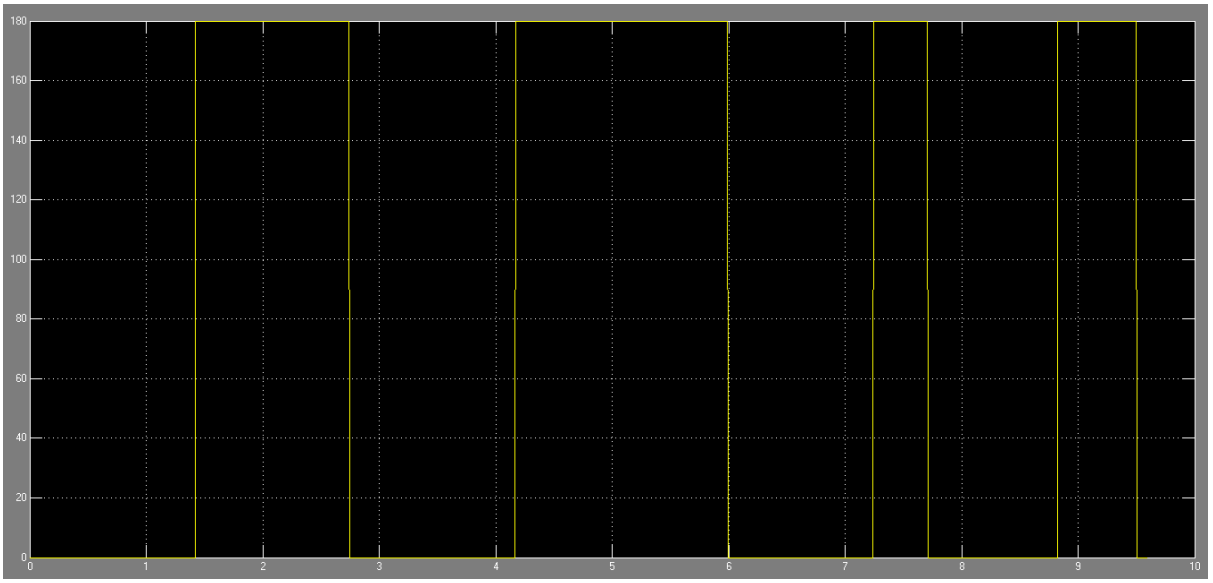

Figura 24. Señal EMG final

El último paso consiste simplemente en transformar este 180 o 0 en parámetros que el servo entiende como órdenes, para lo cual se hace uso de la misma expresión utilizada en la parte 2 de este informe. Esta información, a través del bloque Host Serial Tx, es enviada al bloque de USB VCP Receiver STM32F4 del programa test.mdl, de igual forma que en la figura 16.

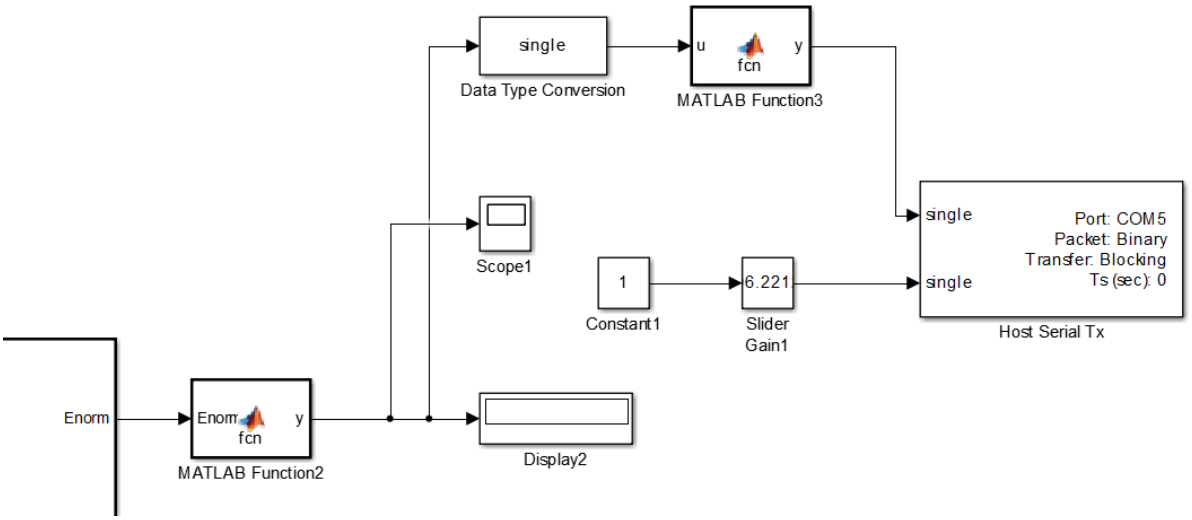

Figura 25. Salida del servo

Como conclusión, en esta práctica hemos podido observar de primera mano cómo funcionaría en un sistema real la aplicación de señales de electromiografía al diseño de algún dispositivo de asistencia médica o rehabilitación. Esta experiencia ha sido esencial para comprender cómo la teoría de control automático se puede aplicar de manera efectiva en el ámbito de la biomedicina. Desde el análisis y filtrado de señales hasta el control de dispositivos físicos como servomotores, hemos podido apreciar la relevancia y el potencial de la ingeniería automática en la mejora de la calidad de vida de las personas.