Control mioeléctrico de una prótesis robótica mediante máquinas de estados.

Mario Amorós Rivera
Pablo Rocamora García
Miguel Baidal Marco
Fernando Augusto Marina Urriola

Índice

1. Introducción

Las prótesis han sido empleadas desde hace miles de años (ver figura 1 y 2) para cubrir las necesidades carentes por causa de una amputación. Además, las prótesis palian el trauma que genera al usuario por la pérdida del miembro al volver a dotar a las personas de la funcionalidad de la extremidad, consiguiendo así que la persona afectada pueda interactuar con el entorno de una forma más acorde a como lo haría si tuviese esa extremidad.

Las prótesis más eficaces para devolver la movilidad del miembro superior son las que funcionan a partir de control electromiográfico; a partir de las señales bioeléctricas que se crean durante la contracción de los músculos al realizar movimientos. Si bien, se sabe que las personas que sufren de una amputación ya no pueden realizar movimientos, quedan señales bioeléctricas residuales en los músculos afectados, aún funcionales, que se utilizan para realizar este control.

(Universitaria, 2004)

Un estado es la condición de una cosa en un tiempo determinado. Una máquina de estados es un modelo de comportamiento de un sistema con distintas entradas y salidas que no solo dependen de las señales de entrada actuales, sino también de las anteriores. Existen diferentes tipos de máquinas de estado como, por ejemplo: (1) Máquina de estado Moore, (2) Máquina de estado Mealy etc (ver figuras 3 y 4).

(Captación, 2019)

2. Objetivos y tareas

Objetivos

Diseñar el control mioeléctrico de una prótesis robótica siguiendo un esquema de control basado en máquina de estados. El entorno simulado permitirá realizar movimientos de flexión/extensión, pronación/supinación y apertura/cierre de la prótesis. Se diseñarán varias máquinas de estados comandadas por movimientos de flexión, extensión y co-contracción del antebrazo.

Tareas

- Generar secuencias de control a partir de las señales EMG disponibles en la práctica 2.
- Implementar distintas máquinas de estados que generen los comandos de control adecuados a partir de las secuencias anteriores.
- Implementar un entorno en el que se visualicen las acciones de control de cada máquina de estados en base a las secuencias de contracción muscular predefinidas.

3. Materiales

Se han utilizado diferentes programas, además de componentes hardware, tales como:

- Coppelia (V-REP)
- Matlab
- PC
- Brazo virtual 3D
- Autodesk Fusion 360

4. Proyecto

4.1 Creación de modelo funcional

Para la realización de este proyecto se ha descargado un prototipo de brazo robótico desde este enlace.

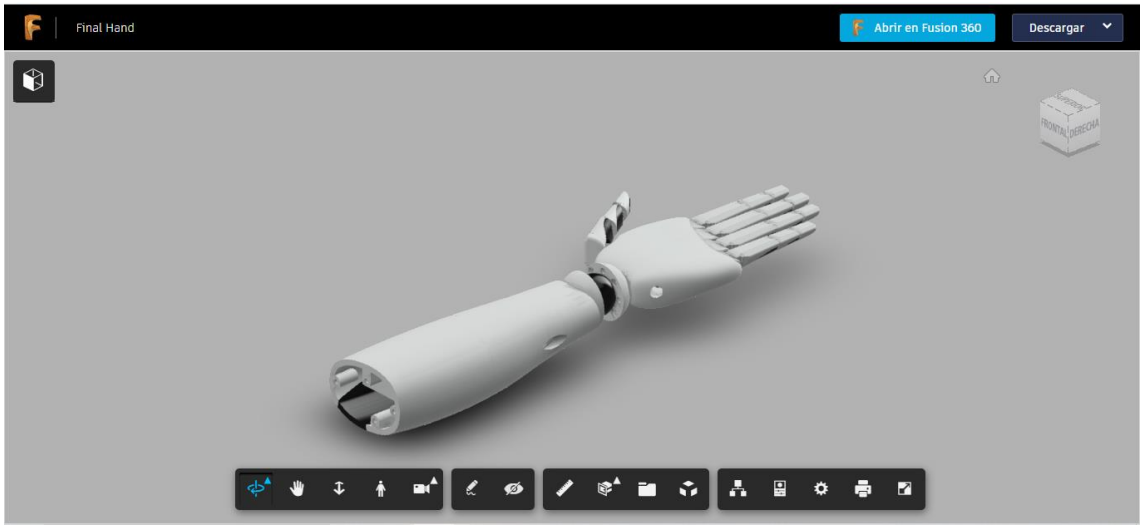


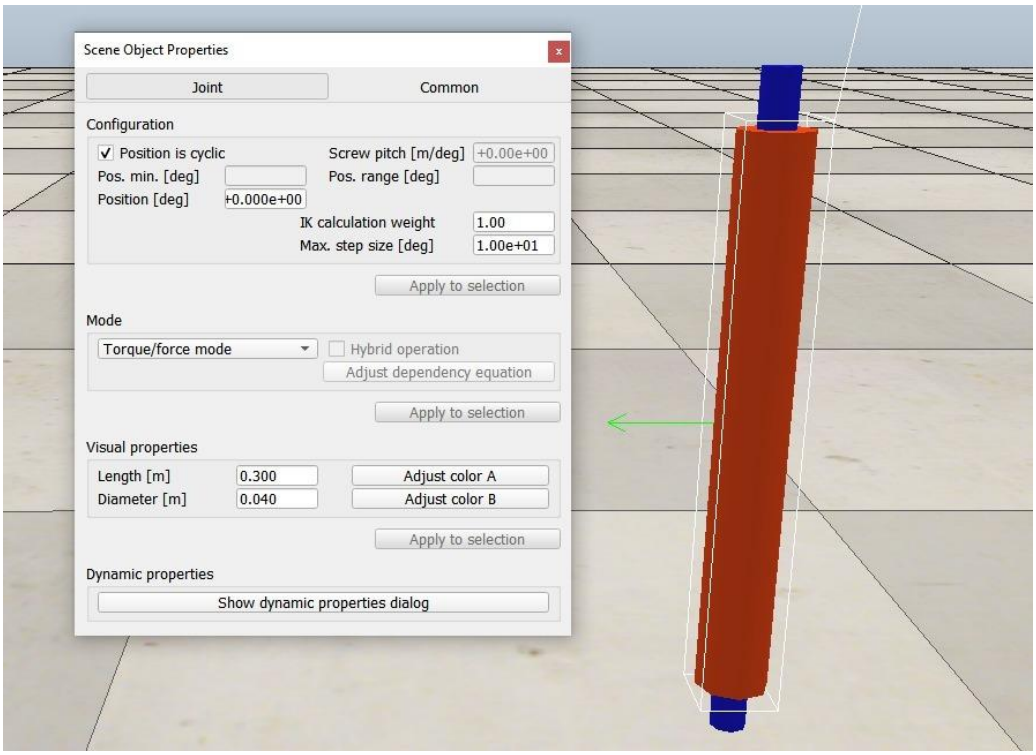
Figura 1. Brazo robótico en Autodesk Fusion 360.

Este brazo 3D ha sido modificado a través del editor 'Autodesk Fusion 360', para, posteriormente, realizar las diferentes mejoras necesarias en Coppelia. El formato de los archivos utilizados es '.obj', permitido en editores de esta índole, tales como Coppelia, Unity etc.

Una vez editado el brazo 3D, se ha procedido al montaje de este dentro de Coppelia. En este software se ha realizado la unión de las piezas mediante articulaciones que más tarde serán capaces de moverse dependiendo de la señal EMG que se ha enviado a Matlab, consiguiendo así la interoperabilidad requerida en este proyecto.

En V-REP, se ha realizado el montaje del brazo robótico, cada una de las piezas se ha puesto como pieza dinámica.

Se han realizado modificaciones a cada una de las piezas y se han creado las articulaciones necesarias (*joints*), estas son las siguientes:



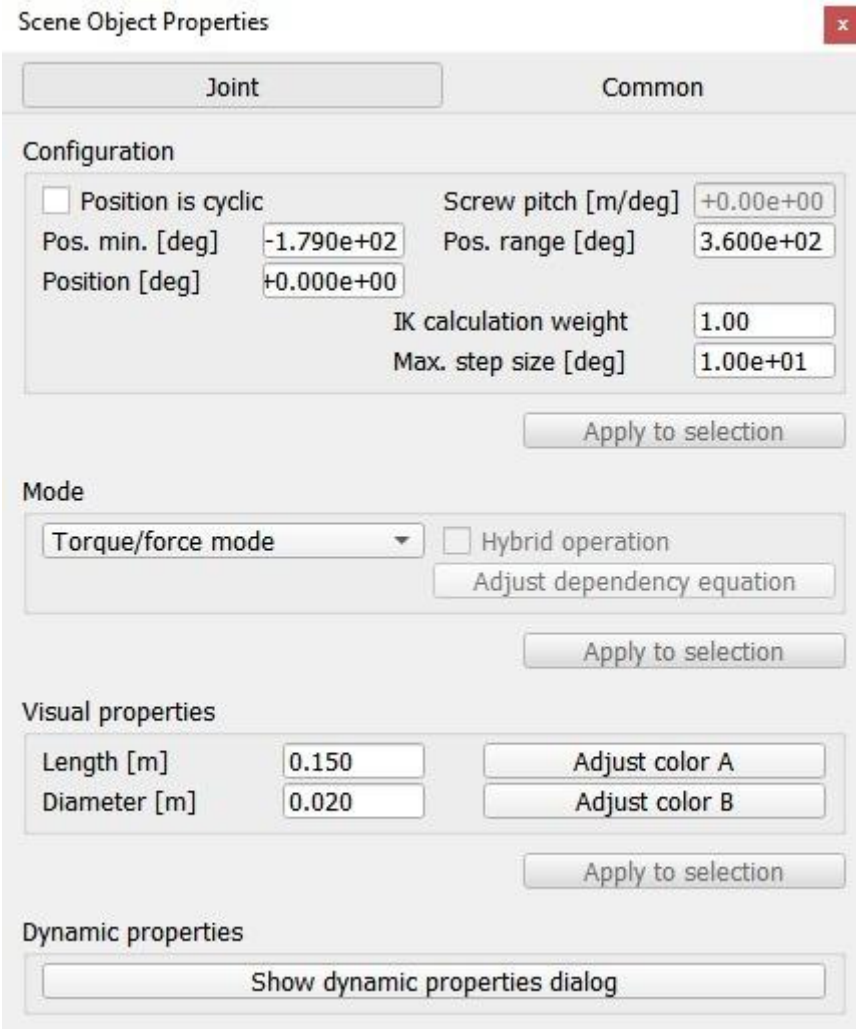


Figura 2 y 3. Ejemplo de joint (articulación)

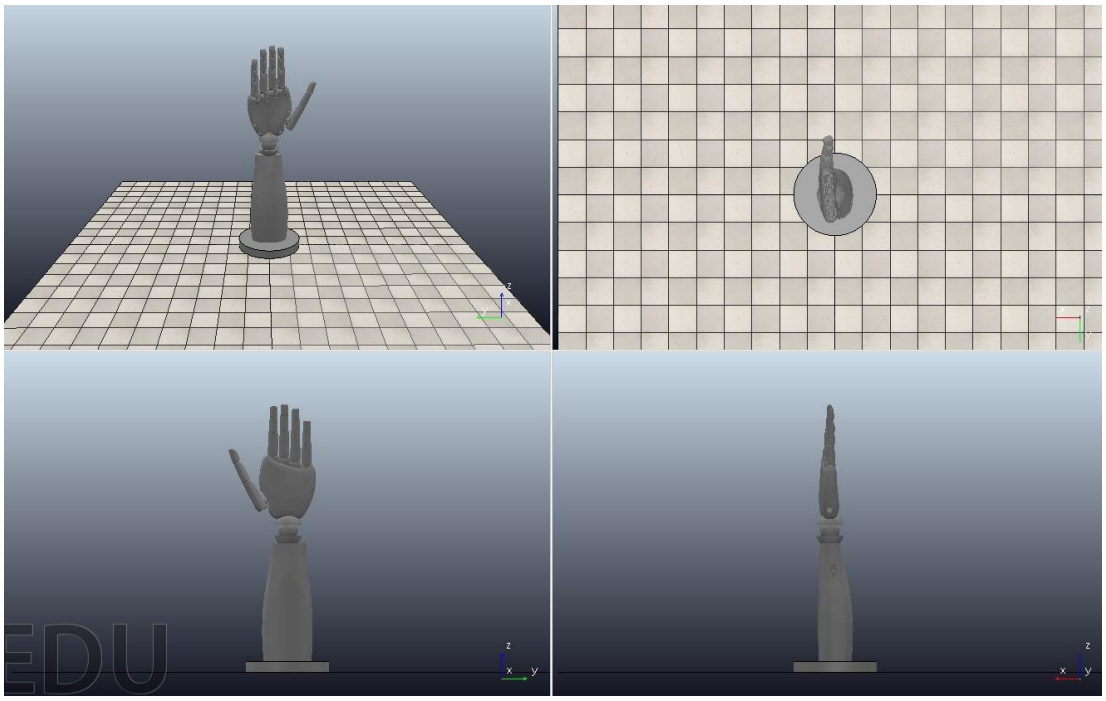


Figura 4. Brazo robótico, diferentes vistas.

A cada una de las articulaciones se les ha puesto una fuerza igual a la necesaria para aguantar el peso de la pieza que está sujeta, véase el siguiente ejemplo:

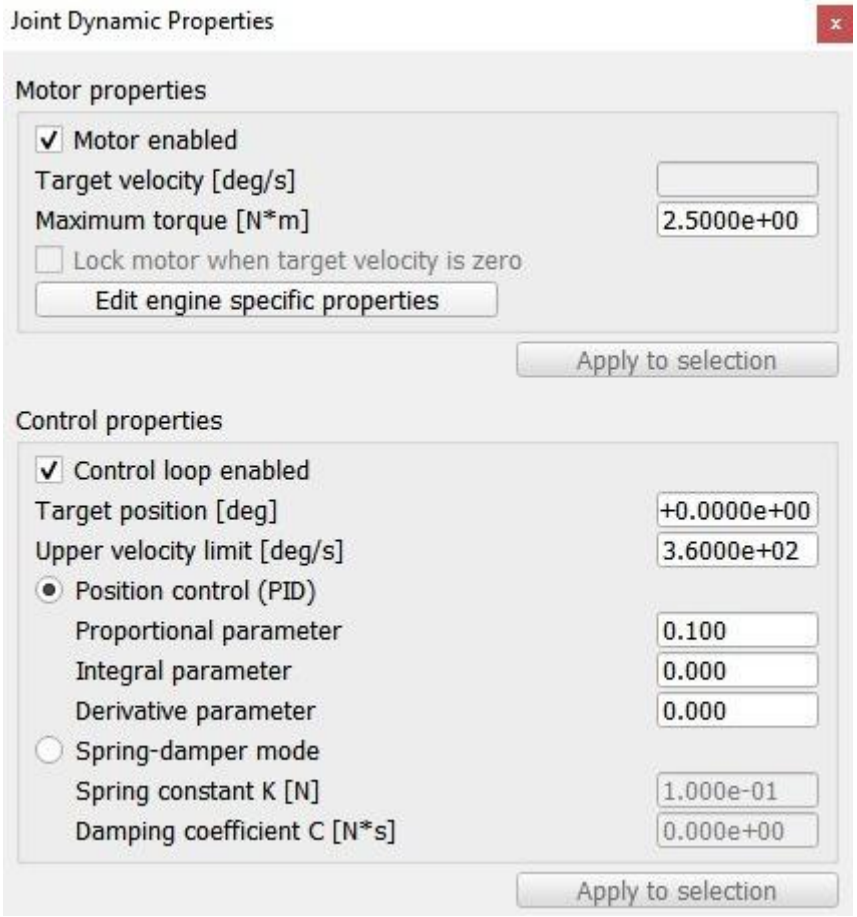


Figura 5. Ejemplo de motor y torque, variables configurables de los joints.

Después de haber realizado las modificaciones necesarias en Coppelia, se ha procedido a la creación de código necesario para conseguir la interoperabilidad entre Matlab y Coppelia.

4.2 Conexión entre Matlab y V-REP

Para manejar el robot simulado en Coppelia (V-REP) mediante código Matlab (programa externo) debe establecerse una conexión cliente-servidor mediante el uso de la API de V-REP diseñada para permitir el control remoto y de los Sockets correspondientes (uno para cada programa) para que se pueda crear el hilo que permitirá la comunicación entre ambos programas.

El servidor (V-REP) puede recibir y ejecutar las instrucciones que envía el cliente (Matlab) al igual que puede enviar información al cliente como puede ser la posición en coordenadas de un objeto.

La API remota de V-REP para Matlab (existen diversas APIs para posibilitar su manejo mediante otros lenguajes como Java, C++, etc) dispone de una gran variedad de métodos a través de los cuales se puede tanto enviar información al robot como solicitarla.

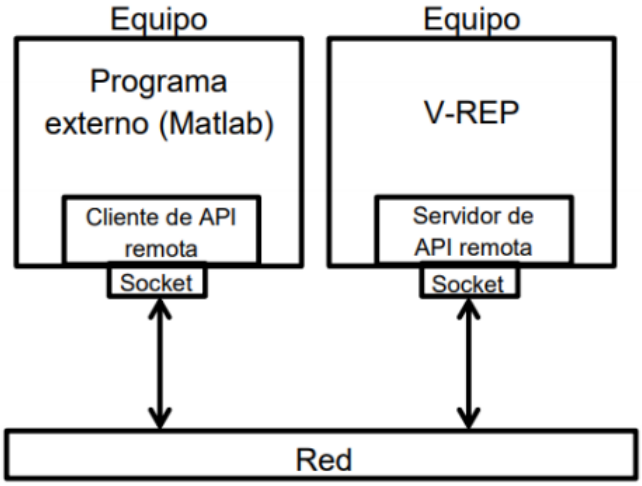


Figura 6. Esquema de interoperabilidad Matlab y V-REP

Para poder conectar correctamente nuestro IDE de MATLAB desde el que se mandan instrucciones con V-REP es necesario realizar una serie de ajustes en ambos programas:

En MATLAB se debe importar la API remota que ofrece V-REP para hacer posible la interconexión entre ambos. Para ello se deben copiar todos los archivos .m de la carpeta donde están los archivos del programa V-REP; *"…/V-Rep-/programming/remoteApiBindings/matlab/matlab"* en la carpeta de los archivos de MATLAB: *"Matlab/R2016b/bin".* También se ha de copiar en la misma carpeta el archivo "remoteApi.dll" que se encuentra en la carpeta *"V-Rep/programming/remoteApiBindings/lib/lib",* donde hay dos subcarpetas entre las cuales se ha de elegir la correspondiente al sistema operativo del dispositivo con el que se trabaja (32 o 64 bits).

En cuanto a V-REP, es necesario generar un nuevo script de tipo "child" desde la pestaña "scripts". Dentro de ese script, escoger en la pestaña "Associated Object cualquier objeto del robot por ejemplo "DefaultCamera" (la cámara que nos permite ver y moverse por la escena) que no se elimina en ningún momento. Finalmente se ha de abrir el código correspondiente del script

(lenguaje Lua) y añadir la instrucción simExtRemoteApiStart(19999) con el fin de abrir un puerto a través del que Matlab se conectará.

Por último, se ha de incluir en el script de Matlab el siguiente código:

```
% CONEXION CON COPPELIA
vrep=remApi('remoteApi'); % Objeto remoto "vrep"
vrep.simxFinish(-1); % Se cierran posibles conexiones previas
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5); % Nueva conexion con V-REP
if (clientID>-1)
```

Se crea un objeto "vrep" que representará nuestro robot, al cual se le irán implementando los métodos correspondientes para que el robot se mueva como se explica más adelante.

Se eliminan posibles conexiones previas que se encuentren abiertas y se inicia la comunicación a través del puerto que se ha indicado en el script de V-REP en el método simExtRemoteApiStart(), en este caso el 19999. El resto de los parámetros son valores estándar y vienen por defecto.

Si la conexión se ha producido correctamente la variable "clientID" tendrá valor de –1 y comenzará la simulación.

Para cerrar la conexión, se "corta" la comunicación entre los dos Sockets y se elimina el objeto "vrep" al igual que se limpia la parte de memoria RAM utilizada:

```
vrep.simxFinish(clientID); %Corta el hilo de comunicación con V-REP
vrep.delete(); %Elimina el objeto "vrep"
clear all; clc %Se limpia la memoria RAM utilizada
```

4.3 Obtención y análisis de la señal EMG

Para el desarrollo del proyecto han sido empleadas señales de electromiografía de los músculos flexor y extensor del antebrazo mediante el equipo EMG Noraxon MiniDTS. Este equipo es capaz de proporcionar una señal preprocesada y lista para ser empleada. Dado que no se ha podido obtener una señal en tiempo real, y se disponía de una única señal se ha desarrollado un código mostrado a continuación para multiplicar un fragmento de señal un numero cualquiera de veces como muestra la ilustración 7.

```
clear all
clc
load('EMG_2_channel_Wrist_Flex_Ext_Proc.mat')%Cargar el archivo

t_total=Data{1};
flexor=Data{2};
extensor=Data{3};

flexor_comp=flexor; %Segmento que contiene copia del vector flexor
inicial
extensor_comp=extensor;
incremento_t=t_total(2,:);%El segundo valor contiene el incremento de la
serie temporal
t_repeticion=t_total(end,:);%Tomamos el ultimo valor de la serie temporal
final=5; %Número de veces que se desea copiar la señal(Ultimo número se
incluye)
%Si quieres 4 repeticiones pones 3
%Si quieres 3 repeticiones pones 2...
```

```
for n=1:final
    flexor=vertcat(flexor,flexor_comp);%Se concatena la señal del flexor
    extensor=vertcat(extensor,extensor_comp);%Se concatena la señal del
extensor
    t_inicial=t_total(end,:);%Se toma el ultimo valor temporal del vector
de tiempos como principio de la siguiente repeticion
    disp("Iteracion "+n);
    disp("Tiempo de inicio: "+t_inicial);
    t_final=t_inicial+t_repeticion;%Ultimo elemento del vector
    disp("Tiempo de finalizacion del vector "+t_final);
    t_comp=t_inicial:incremento_t:t_final;%Creamos el vector
    t_comp=transpose(t_comp);%Trasponemos el vector
    t_total=vertcat(t_total,t_comp);%Concatenamos el vector de tiempos
que teníamos previamente y el nuevo
end
```

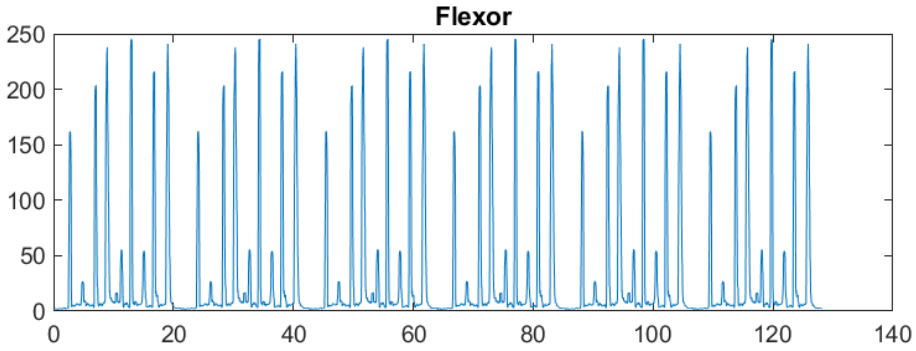


Figura 7. Señal formada por 6 repeticiones de la señal original para el flexor

Además, se ha construido una señal alternativa que presenta un aspecto similar a la señal original tal y como muestra la ilustración 8 y que ha sido reproducida con el método mencionado previamente un numero n de veces hasta obtener el valor de tiempo deseado.

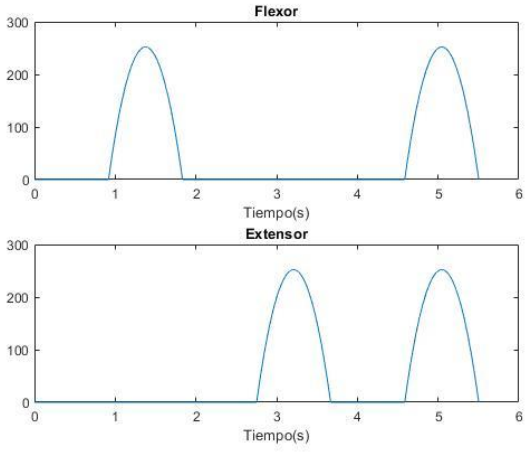


Figura 7. Señal creada a partir de una función cuadrática para los canales flexor y extensor.

4.4 Manejo del robot mediante código

Para manejar el robot se ha hecho uso de los siguientes métodos de la API:

simxAddStatusbarMessage(clientID, mensaje, modo_operación);

Su finalidad es poder incluir un mensaje de texto en la consola de Coppelia. Se ha utilizado para que aparezca un mensaje por pantalla en el programa de Coppelia confirmando que la conexión entre él y MatLab ha sido correcta.

simxGetObjectHandle(clientID, nombre_objeto, modo_operación);

Para poder trabajar con cualquier articulación u objeto del robot de Coppelia es imprescindible generar previamente un objeto "handler" o manejador a partir del cual trabajar. Este método devuelve ese objeto "handler" correspondiente al que se le introduce como segundo parámetro (tipo String).

simxSetJointTargetPosition(clientID, handler_objeto, posición, modo_operación);

En el caso de este proyecto los manejadores sólo son utilizados para programar los movimientos de la mano robótica. Este método permite indicar por parámetro la posición que se desea que tome una articulación (joint) concreta mediante su correspondiente "handler".

Como se aprecia en el código, todos los métodos devuelven como mínimo un "[returnCode]" (además de otros parámetros como en el caso del objeto manejador) de tipo numérico. Este parámetro de salida indica si la función se ha ejecutado de forma correcta. (enlace)

Los modos de operación incluidos en los métodos como último parámetro sirven para indicar el tipo de relación cliente-servidor necesitará ese método con el fin de evitar retrasos y optimizar el tiempo. Es decir, no es lo mismo un método que sólo necesita enviar una posición destino para un objeto determinado, el cual sólo necesita el envío de datos desde cliente a servidor sin respuesta por parte del servidor. Mientras tanto, otros métodos como obtener una determinada posición de un objeto sólo necesitará información desde el servidor al cliente. En la API aparece el método recomendado para cada método. (enlace) (enlace 2)

4.5 Creación de la simulación:

Para la creación de la simulación, lo hemos realizado de dos formas distintas (ambas igual de válidas). Para dichos algoritmos, se han de conocer los siguientes parámetros:

- Frecuencia de la señal (en este caso, 1500Hz)
- Tiempo de señal: este valor será el valor máximo del vector tiempo de la señal
- Tiempo de ejecución: el tiempo de ejecución que queramos ver lo observaremos gracias a las funciones tic toc, la cual nos permite conocer el tiempo de ejecución entre tic y toc donde se guardará en toc.

La primera forma, vista en el archivo PROYECTO.m donde se conocen la cantidad de iteraciones realizadas ya que la cantidad de la ventana de cada iteración será del tamaño de la muestra entre 100 por tanto, realizará 100 iteraciones, inconvenientes de este método: que conforme aumenta la señal, la ventana será más grande y la visualización en la gráfica será peor.

Una vez se conocen las iteraciones, se calcula el tiempo por iteración(tpi) que se hallará con la siguiente forma:

Tpi=tiempo/100

Una vez tenemos el tpi, lo podemos utilizar para calcular el tiempo que debe estar en pausa cada vez que termine la iteración mediante:

Tpi=tiempo de pause +tiempo de ejecución (tic toc)

Y despejando, se halla.

La segunda forma es conociendo la frecuencia de la señal, en este caso, sabemos que será

4.7 Máquina de estados

La máquina de estados se ha definido de acuerdo con las siguientes especificaciones:

En primer lugar, se han definido unos umbrales de contracción para cada músculo, de este modo podemos determinar diferentes estados de reposo o de contracción con una determinada salida tal y como demuestra la tabla 1.

Tabla 1. Criterios de activación y salida

Músculo	Criterio	Salida
Flexor	>= 35	0
	< 35	1
Extensor	>= 20	0
	< 20	1

A continuación, se ha asociado un valor entero entre el 0 y el 6 a cada movimiento realizado por la prótesis tal y como muestra la tabla 2. Estos valores identificaran el tipo de movimiento que va a ser ejecutado en cada momento. Además, se ha relacionado cada movimiento con un estado determinado siendo 1, 2 o 3. Cabe destacar que cuando no se produce activación en ninguno de los músculos no se tiene en cuenta el estado puesto que no se realiza ningún movimiento. Otro aspecto que destacar es que cuando se produce una co-contracción, es decir, una doble activación de flexor y extensor, no se realiza ningún movimiento, pero se realiza un cambio de estado.

Tabla 2. Movimientos y estados con su respectivo valor asociado.

Estado	Movimiento	Valor
Ninguno	Reposo	0
Estado 1	Flexión	1
	Extensión	2
Estado 2	Pronación	3
	Supinación	4
Estado 3	Cierre	5

	Apertura	6

Además, se ha establecido un estado inicial en el que se encuentran los movimientos de flexión y extensión (estado 1) que serán accionados ante una contracción muscular en uno de los dos canales.

Una vez clasificados los estados, los umbrales de contracción y los movimientos han sido relacionados de acuerdo con la siguiente combinación de los anteriores.

Extensor	Flexor	Estado	Salida	Movimiento
0	0	Cualquiera	0	Reposo
0	1	1	1	Flexión
1	0	1	2	Extensión
0	1	2	3	Supinación
1	0	2	4	Pronación
0	1	3	5	Cierre
1	0	3	6	Apertura
1	1	Cualquiera	Cambio de estado	

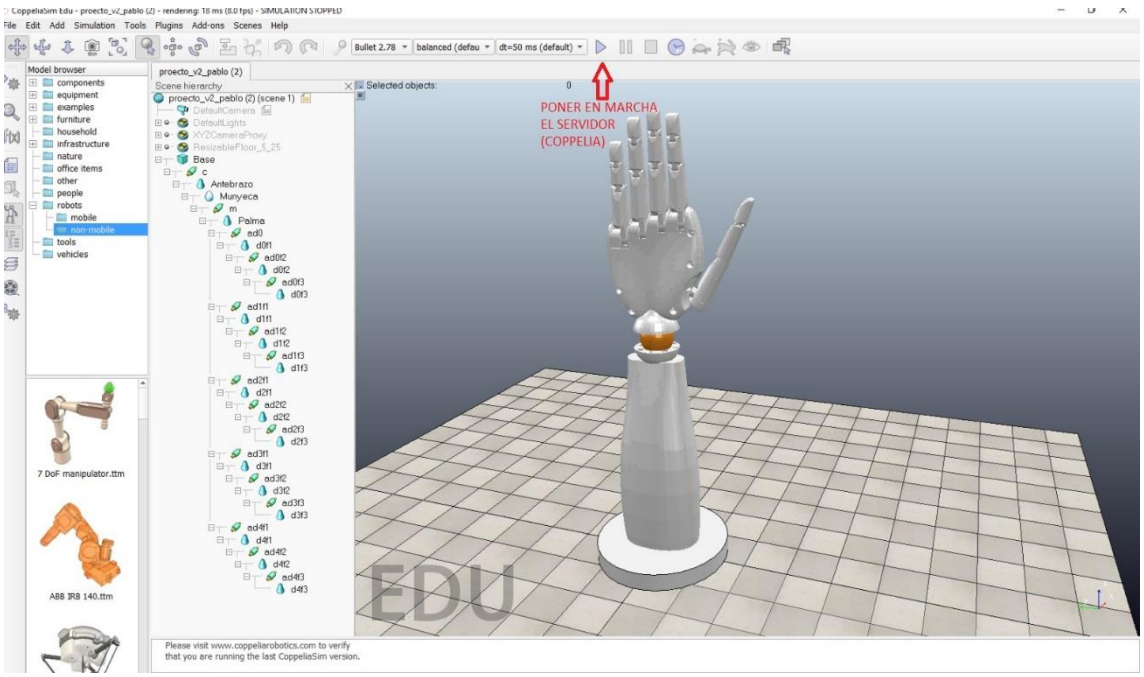
El código empleado ha sido el siguiente:

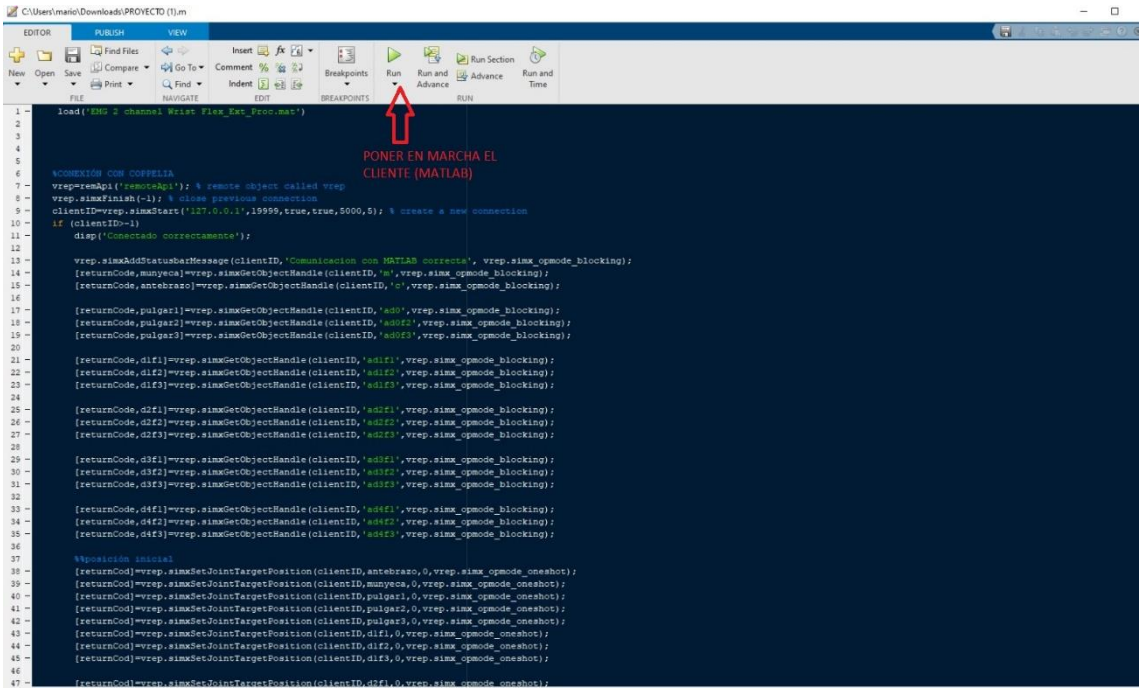
```
clear
clc
arch1="EMG 2 channel Wrist Flex_Ext_Proc_v2.mat"
load('generar_senyales_v3.mat');
time=Data{1};
flex=Data{2};
ext=Data{3};
[m,n]=size(flex);%Se obtienen las dimensiones es de la matriz
umbrales=[35,20];%Establecemos los umbrales de forma externa de modo que
sea mas sencillo modificarlos
estados_forzados=[1,3,5]%"flexion","pronacion","cierre"
estados_relax=[2,4,6]%"extension","supinacion","apertura"
estado=1;
pos=1;
while pos<=m
    if (flex(pos)>=umbrales(1) && ext(pos)>=umbrales(2))%Se comprueba si
hay co-contraccion en la posicion especificada
        if (flex(pos+1)<umbrales(1) || ext(pos+1)<umbrales(2))%Se
comprueba si en la siguiente posicion a la especificada no hay co-
contraccion
            if estado==1%Si es asi se cambia de estado
                estado=2;
            elseif estado==2
                estado=3;
            elseif estado==3
                estado=1;
            end
```

```
        else%En caso contrario se ejecuta la accion correspondiente al
estado de contraccion
            Estados(pos,1)=estados_forzados(estado);
        end
    else%En caso de no haber cocontraccion se comprueba qué accion se
debe ejecutar
        if(flex(pos)>=umbrales(1) && ext(pos)<=umbrales(2))%Si hay
contraccion en el flexor se realizará "flexion","pronacion","cierre"
segun el estado
            Estados(pos,1)=estados_forzados(estado);
        elseif(flex(pos)<umbrales(1) && ext(pos)>=umbrales(2))%Si no hay
contraccion de flexor se realizará "extension","supinacion" o "apertura"
según el estado actual
            Estados(pos,1)=estados_relax(estado);
        else
            Estados(pos,1)=0;
        end
    end
    pos=pos+1;
end
```

4.8 Manual de funcionamiento

Para poner en funcionamiento el programa, primero se deberá dar al play en coppelia y luego se debe dar play en matlab; una vez hecho esto, la conexión entre Matlab y V-Rep funciona como una conexión cliente-servidor.





5. Conclusión

La mano humana es un instrumento muy complejo por lo que no es posible plasmar todas sus funcionalidades y características en un periodo corto de tiempo como el que supone este proyecto. El proyecto realizado cumple con los requisitos y funcionalidades marcadas desde el inicio, se puede determinar que se ha realizado de manera correcta puesto que se han cumplido todos los requerimientos de manera exitosa. El modelo se ha realizado en un entorno ideal para, en un futuro y cuando el desarrollo sea completo, poder conectarlo con otros modelos de actuadores o exoesqueletos ya modelados

6.Bibliografía

Captación, M. (2019). Máquinas de estado. *MCI Captación*.

Human Robotics. (s.f.). *EVALUACIÓN DE UN ESQUEMA DE CONTROL MIOELÉCTRICO.* Sant Vicent del Raspeig: Universidad de Alicante.

Universitaria, R. D. (2004). *ROBÓTICA Y PRÓTESIS INTELIGENTES.* México .

Conexión Coppelia y Matlab

https://robologs.net/2017/07/04/interaccion-entre-v-rep-y-matlab/

https://idus.us.es/bitstream/handle/11441/94413/TFG-2683-MORALES%20MARQUEZ.pdf?sequence=1&isAllowed=y

Comandos http://oa.upm.es/54445/1/TFG_PABLO_MARTINEZ_CAMPOS.pdf

Api de matlab

https://www.coppeliarobotics.com/helpFiles/en/remoteApiConstants.htm#functionErrorCodes

Función tic toc

https://es.mathworks.com/help/matlab/ref/tic.html

7. Anexo

Figura 1. sofisticación en el antiguo Egipto: prótesis de hace 3.000 años

Figura 2. Prótesis edad media antebrazo.

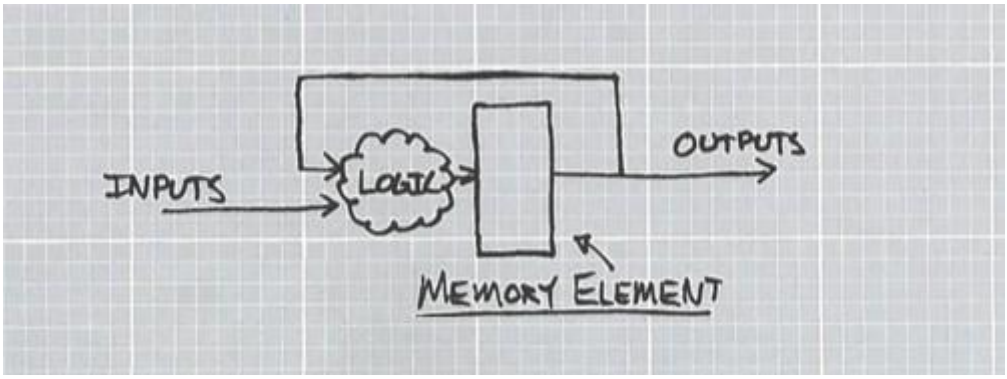


Figura 3. Máquina de estado Moore.

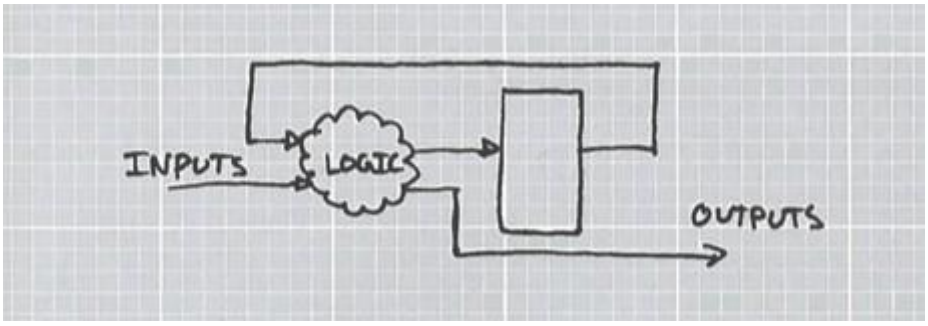


Figura 4. Máquina de estado Mealy.