**David Cano Acedo**

# Anàlisi i ampliació d'usos d'un exoesquelet de braç d'usos educatius

**TREBALL DE FI DE GRAU**

**dirigit per Albert Oller Pujol**

**Grau d'Enginyeria Electrònica Industrial i Automàtica**

UNIVERSITAT ROVIRA I VIRGILI

**Tarragona**

**2024**

# ÍNDEX

# 1 Memòria.

## 1.1 Objecte.

L'objectiu d'aquest treball consisteix en analitzar un exoesquelet de braç i veure quines són les funcionalitats que té. Aquest exoesquelet es fa servir en les pràctiques de laboratori de l'assignatura de Robòtica Mèdica del grau en Enginyeria Biomèdica i ja s'utilitza en alguns àmbits per a la formació d'estudiants.

No obstant, a banda d'analitzar els usos que ja es donen a aquest braç robòtic, també s'ha d'ampliar els usos que pot arribar a tenir com a part de l'objectiu del projecte. A més a més, també s'ha de dissenyar una forma de fer el sistema més compacte i robust per evitar que hi hagi desperfectes.

## 1.2 Abast.

Per aconseguir els objectius plantejats a l'apartat anterior d'analitzar i ampliar els usos que pot tenir l'exoesquelet de braç, es seguiran els següents punts, que mostren tot allò que s'abasta en el treball:

- Estudi de l'exoesquelet: Primer de tot el que s'ha de fer és analitzar i estudiar el funcionament de l'exoesquelet. Per fer això s'analitzaran els seus components i es llegirà i s'estudiarà el manual de l'exoesquelet de braç i dels seus sensors i actuadors.

- Muntatge elèctric: El següent pas és fer les connexions elèctriques necessàries per aconseguir que el braç funcioni. La principals connexions elèctriques que s'han de fer són les de l'amplificador INA 125.

- Programació en Arduino: Una vegada es coneix el funcionament de l'exoesquelet i dels seus components i totes les connexions estan realitzades, el següent pas es programar el braç i fer proves amb el software d'Arduino. Aquest software és el que s'aplica normalment en els laboratoris on s'utilitza aquest exoesquelet.

- Programació en Simulink: Com a ampliació i millores a aplicar al braç robòtic el que es fa es programar l'exoesquelet a través de Simulink, una eina de MATLAB. Això permetrà fer programes més senzills i intuïtius. A més, amb aquest programa serà possible realitzar programes de control.

- Programació en Unity: Una altra millora d'aquest projecte consisteix en programar un videojoc que es connectarà al braç robòtic. Amb el moviment del braç serà possible controlar el joc i motivar als pacients en la seva rehabilitació.

- Disseny de la caixa de protecció: Finalment, s'ha de dissenyar un mètode, en aquest cas una caixa, on situar les connexions de l'amplificador i de la placa Arduino per fer tot el sistema més robust.

## 1.3 Antecedents.

En aquest apartat s'explicaran tots els conceptes que són necessaris per comprendre els apartats que es troben més endavant.

### 1.3.1 Rehabilitació.

En la robòtica de rehabilitació la interacció entre home-màquina té molta importància. Normalment, en la rehabilitació les activitats que es desenvolupen consisteixen en una sèrie de moviments repetitius fets amb ajuda d'un professional, com per exemple un fisioterapeuta. La inserció d'un robot permetria facilitar el treball del professional, que només hauria d'explicar al pacient com fer l'exercici, i ajudaria al pacient a fer els moviments repetitius que li han explicat.

L'ús de robots en el camp de la rehabilitació té molts avantatges. Per exemple, durant l'exercici de rehabilitació, el robot pot recopilar informació sobre com realitza el moviment el pacient i com és la seva evolució. També pot provocar moviments passius que simulen els moviment que realitzaria el fisioterapeuta.

### 1.3.2 Robòtica Mèdica.

Dins de la robòtica de rehabilitació hi ha diversos sistemes robòtics que permeten facilitar i millorar la rehabilitació dels pacients. Aquests sistemes es poden classificar en dos arquitectures diferents:

- Robots basats en efector final: En aquest tipus de sistemes només existeix un punt de contacte entre el robot i l'usuari. L'avantatge d'aquest sistema és la senzillesa a l'hora d'ajustar el robot a l'usuari. Un desavantatge és la posició del membre a rehabilitar, ja que no queda del tot determinada.

- Exoesquelets: En aquest tipus de sistemes el robot es troba fixat al llarg d'un membre, mantenint una estructura similar a aquest. Un avantatge d'aquest sistema és que la posició del membre queda totalment determinada. Com a desavantatge es té que els eixos del robot han d'estar totalment alineats amb els eixos anatòmics del membre que es vol rehabilitar.

En aquest treball es farà ús del segon tipus d'arquitectura, l'exoesquelet. Es treballarà amb aquest sistema i s'analitzarà el seu funcionament. A més, es pretén ampliar les seves funcionalitats i millorar la seva aplicació.

A l'hora de fer la rehabilitació amb robots hi ha diverses formes en les que el robot pot assistir al pacient en funció de quan i com s'administra la força. Hi ha 5 d'aquestes modalitats de control.

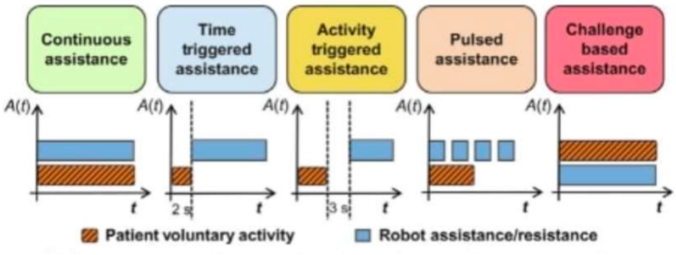



**Figura 1.** Modalitats de control per a la rehabilitació amb robots.

Ara s'explicaran les diferents modalitats de control des de l'esquerra de la imatge cap a la dreta:

- Assistència continua: En aquesta modalitat de control es subministra una força de forma continuada durant tot el temps. És de tipus gradual i en forma de funció de rampa de durada T:

$$A(t) = F(t) \cdot R(t, T) \quad (1)$$

La magnitud de la força d'assistència, expressada com a rigidesa o força màxima, s'ha d'establir amb cura d'acord a la força mínima que pot fer el pacient per iniciar el moviment. És una modalitat més simple, i dirigida a pacients més greus, que inicialment no poden iniciar els moviments de forma autònoma. No obstant, aquesta força d'assistència s'ha d'anar regulant cada sessió d'acord a l'evolució del pacient.

- Assistència amb restricció temporal: En aquesta modalitat de control a l'inici el pacient pot moure lliurement el dispositiu, però quan passa un cert temps $t_0$ la força comença a actuar.

$$A(t) = F(t) \cdot R(t - t_0, T) \quad (2)$$

- Assistència en funció de l'activitat: En aquesta modalitat de control el dispositiu només subministra força si l'usuari realitza una activitat. S'incorpora un algorisme per valorar l'esforç que fa l'usurari.

$$A(t) = k(v_H, t) \cdot F(t) \cdot R(t, T) \quad (3)$$

- Assistència polsada: En aquesta modalitat de control la força es subministra de forma intermitent i de forma freqüencial. La fórmula és la següent:

$$A(t) = \left[k + (1-k) \cdot \sum \phi_N \left(t - nT_p\right)\right] \cdot F(t) \cdot R(t) \quad (4)$$

- Assistència en funció d'un repte: En aquesta modalitat de control l'usuari ha de fer els moviments contra una força subministrada.

#### *1.3.2.1 Senyal electromiogràfica (EMG).*

Aquest tipus de senyal es rep quan les fibres musculars generen biopotencials quan aquestes són activades. En un procés de contracció muscular els potencials generats en les fibres musculars són transmesos a la superfície.

Hi ha dos formes de mesurar el senyal EMG, una forma que no es gens invasiva, amb l'ús d'elèctrodes, i una altra forma molt més invasiva, fent ús d'agulles que s'insereixen entre les fibres musculars. En aquest treball s'utilitzarà la primera forma, amb elèctrodes.

Per poder mesurar aquest tipus de senyal s'ha de fer ús d'elèctrodes que es troben enganxats en la superfície de la pell per damunt del múscul en el que es vol llegir la senyal EMG. En aquest treball es mesurarà el senyal EMG mesurada al bíceps, per tant, s'enganxaran dos elèctrodes sobre el bíceps, on es mesurarà la diferencial de potencial, i es connectarà un tercer elèctrode, que serà de referència, en la zona que es troba entre el bíceps i el tríceps. Normalment el rang de la senyal mesurada amb els sensors EMG es troba dins del rang dels 0 als 10 mV de pic a pic.

És una senyal de caràcter estocàstic i de baixa amplitud, per tant, es pot considerar que aquest tipus de senyal és molt sensible al soroll. Per eliminar aquest soroll es fa ús de filtres tant analògics com digitals.

En aquest projecte s'utilitzarà el senyal EMG mesurat al bíceps per poder controlar el moviment i l'accionament de l'exoesquelet de braç EduExo. En cas de no haver activitat elèctrica l'exoesquelet no té perquè entrar en acció, però si es detecta activitat muscular de l'usuari, que indica que vol moure el braç, el sistema s'activarà i ajudarà a l'usuari a moure el braç.

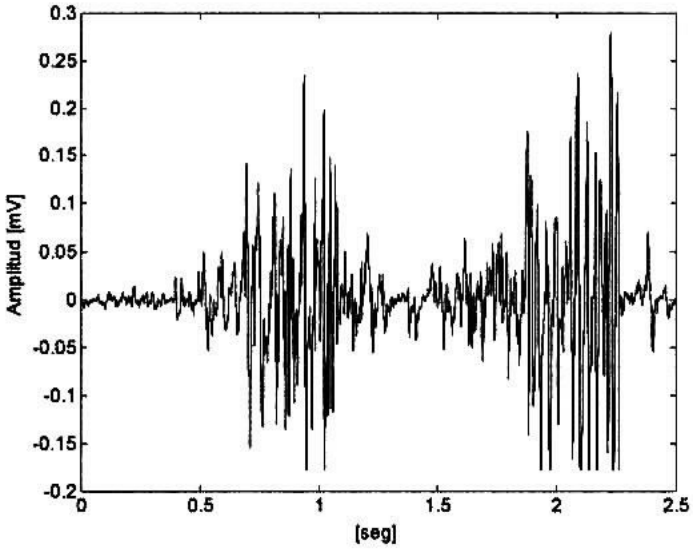



**Figura 2.** Exemple de senyal EMG.

### 1.3.3 EduExo.

EduExo és un kit educatiu que ha estat dissenyat per ensenyar un tòpic que és molt comú avui dia, la robòtica. Aquest kit inclou un exoesquelet robòtic i el suficient material didàctic per a que els estudiants puguin aprendre sobre robòtica, enginyeria i programació. A més, el kit proporciona una experiència pràctica i interactiva sobre els exoesquelets.

A banda del muntatge del braç robòtic, amb EduExo també s'explora els conceptes que comprenen el control i lectura d'un servomotor, el control de diversos sensors com el de força o el sensor EMG i la programació d'un videojoc controlat per l'exoesquelet.

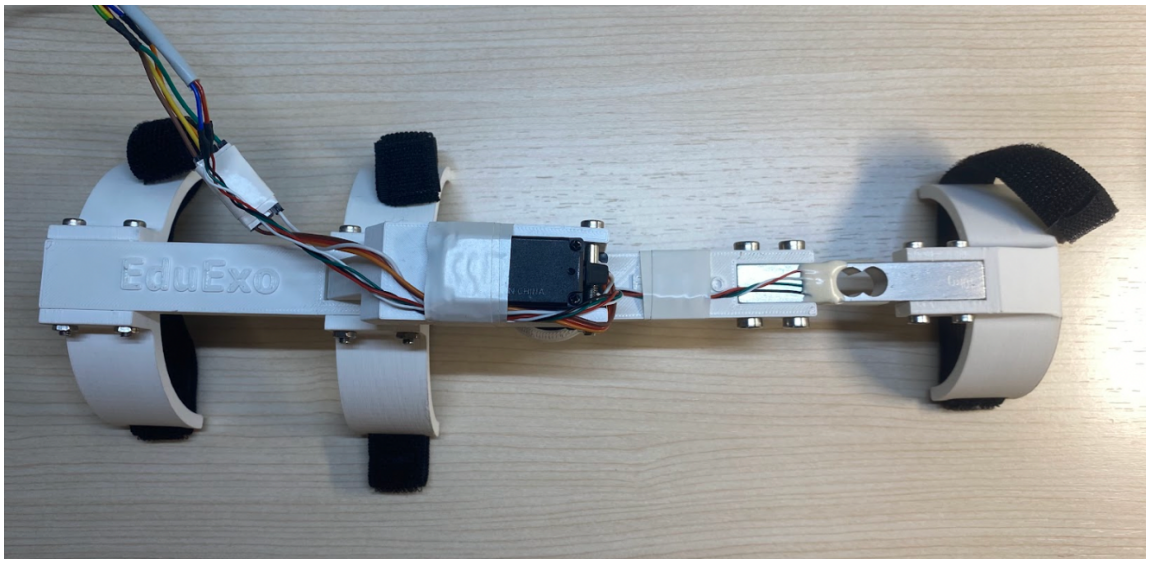



**Figura 3.** Fotografia de l'exoesquelet de braç EduExo.

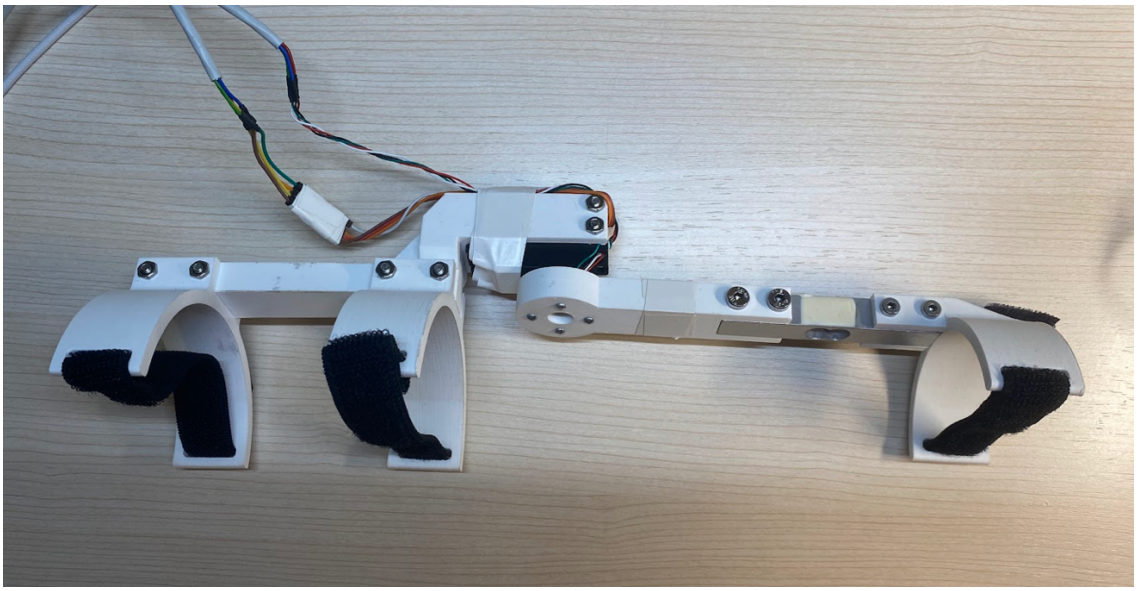


**Figura 4.** Fotografia de l'exoesquelet de braç EduExo.

### 1.3.4 Hardware-In-Loop (HIL).

La simulació Hardware-In-Loop (HIL) és una tècnica que es fa servir pel desenvolupament i comprovació de sistemes encastats en temps reals complexes. Aquest tipus de simulació és efectiva perquè inclou tota la complexitat de la planta que controla el sistema encastat. Això es realitza mitjançant models matemàtics dels sistemes dinàmics, formant la "simulació de la planta".

La simulació HIL ha d'incloure la simulació elèctrica dels sensors i actuadors, i aquestes simulacions funcionen com a interfície entre el model de la planta i el sistema integrat. El valor de cada sensor es troba controlat pel model de planta i es llegit pel sistema encastat. El sistema encastat executa el seu algorisme de control per mitjà dels senyals dels actuadors.

A més, un HIL ha de presentar una interfície de comunicació amb els sistemes dels busos de dades.

Moltes vegades la planta és més cara que un simulador que sigui fiable en temps real, per això es recomana la seva simulació. Això permet que el desenvolupament i la prova amb un simulador HIL sigui més econòmic que amb la planta real. En aquest cas es segueix la idea del HIL, on s'utilitza un sistema real però es simula en el sistema encastat amb Simulink de MATLAB.

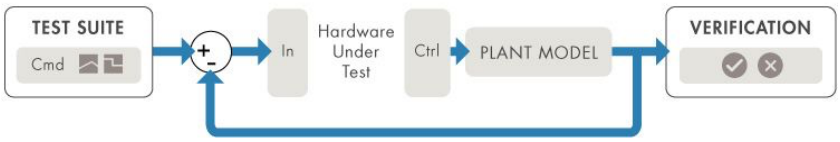



**Figura 5.** Configuració de simulació del HIL.

### 1.3.5 Arduino.

Arduino és una companyia de desenvolupament de software i de hardware lliure que dissenya plaques de desenvolupament de hardware que permeten construir dispositius que interaccionen amb el món real. Arduino es basa en una plataforma de prototipat electrònic de codi obert, que fa sevir maquinari i programari fàcils d'utilitzar. El seu objectiu principal és el de fer accessible la creació de projectes de l'àmbit electrònic en sistemes embedded per a tothom, tant si són professionals com si són principiants.

Arduino està format per diversos components, entre els que trobem el hardware i el software:

- En quant al hardware, el maquinari físic consisteix en un microcontrolador, on normalment fa servir un microcontrolador de la família AVR de Atmel. A més, també es fan servir un nombre de connectors d'entrada i de sortida que varien en funció del model de la placa que es faci servir. Tot això es troba implementat en una placa base, que normalment funciona amb un sistema molt bàsic.

- A banda de les plaques base també hi ha els shields, o plaques d'expansió. La funció d'aquestes plaques és la de complementar i ampliar la funcionalitat de la placa base, permetent que l'Arduino pugui realitzar encara més funcions.

- En quant al software, s'utilitza un Entorn de Desenvolupament Integrat (IDE, amb les seves sigles en anglès), que és el programari que s'utilitza per escriure, compilar i carregar el codi a la placa d'Arduino. El llenguatge de programació que es fa servir per controlar aquestes plaques és C/C++, i es proporciona una interfície senzilla per desenvolupar els programes. A més, en l'àmbit de la programació també es fan sevir les llibreries de d'Arduino, que són col·leccions de codi que poden proporcionar funcionalitats addicionals, com pot ser el control de motor o sensors. L'ús d'aquestes llibreries de codi permeten facilitar i simplificar el desenvolupament de projectes.

Algunes de les característiques que fa que Arduino destaqui és la facilitat en el seu ús, on fins i tot gent que no té gens d'experiència pot fer projectes d'electrònica, la seva comunitat, que comparteix projectes i ofereix suport, i la seva versatilitat, on es pot fer servir Arduino en una infinitat de projectes diferents.

#### *1.3.5.1 Arduino UNO.*

Hi ha molts tipus de plaques de microcontroladors disponibles en el catàleg d'Arduino, però la més comuna, i la que es fa servir en aquest treball és la placa Arduino UNO. Aquesta placa es basa en el microcontrolador ATmega328P, i disposa de 14 entrades/sortides de tipus digital, on 6 d'aquests pins es poden fer servir per a PWM. També es disposa de 6 entrades analògiques, un ressonador ceràmic de 16 MHz, una connexió USB, una presa de corrent i un botó de reset.

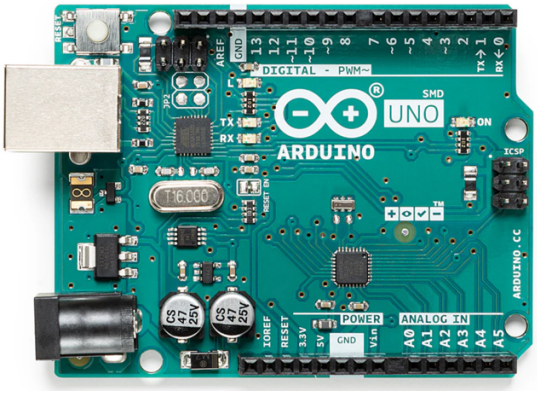



**Figura 6.** Cara anterior de la placa Arduino UNO.

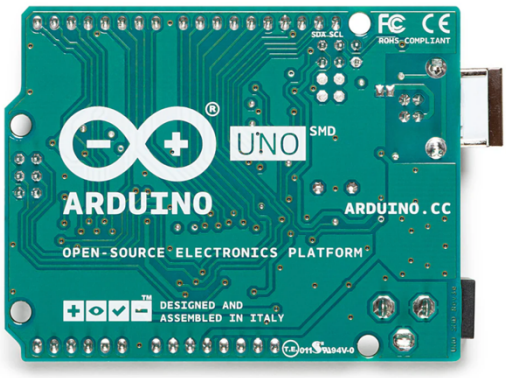



**Figura 7.** Cara posterior de la placa Arduino UNO.

En la següent taula es poden veure les especificacions l'Arduino UNO:

| | |
|---|---|
| **Microcontrolador** | ATmega328P |
| **Tensió d'operació** | 5 V |
| **Tensió d'entrada (recomanada)** | 7-12 V |
| **Tensió d'entrada (límit)** | 6-20 V |
| **Pins digitals d'entrada/sortida** | 14 (6 dels quals tenen sortida PWM) |
| **Pins digitals de PWM d'entrada/sortida** | 6 |
| **Pins d'entrada analògica** | 6 |
| **Corrent DC per pin d'entrada/sortida** | 20 mA |
| **Corrent DC per pins de 3,3 V** | 50 mA |
| **Memòria Flash** | 32 kB dels quals 0,5 kB s'utilitzen pel bootloader |
| **SRAM** | 2 kB |
| **EEPROM** | 1 kB |
| **Velocitat del rellotge** | 16 MHz |
| **LED de la placa** | 13 |
| **Longitud** | 68,6 mm |
| **Amplada** | 53,4 mm |
| **Pes** | 25 g |

**Taula 1.** Especificacions de la placa Arduino UNO.

#### *1.3.5.2 Avantatges i problemes de la programació en Arduino.*

Com s'ha comentat anteriorment, l'exoesquelet de braç d'EduExo s'utilitza en l'assignatura de Robòtica Mèdica del GEB com un element educatiu en pràctiques per ensenyar com programar-lo i llegir les dades que recopila, i per fer totes aquestes activitats l'ús de la placa Arduino i de la seva IDE és una bona opció. Amb aquesta IDE és poden realitzar programes molt senzills i entenedors per a gent que no és molt experta en la programació de línies de codi.

No obstant, a l'hora de fer programes més complexos i a la vegada aconseguir que continuïn sent entenedors, és millor fer ús d'un altre mètode de programació. El millor mètode en aquest cas és l'ús de diagrames de blocs. Aquest tipus de programació permet fer programes complexos però fàcils d'entendre, a més de poder modificar-los de forma

ràpida i senzilla. Per aplicar aquest tipus de programació és farà ús de Simulink, una extensió de MATLAB.

### 1.3.6 MATLAB.

MATLAB és un entorn de programació i càlcul numèric utilitzat en àmbits acadèmics i industrials. MATLAB és un sistema de còmput numèric que ofereix un Entorn de Desenvolupament Integrat (IDE) amb un llenguatge de programació propi, anomenat llenguatge M.

Algunes característiques d'aquest programa són:

- L'Entorn de Desenvolupament Integrat (IDE) de MATLAB inclou una finestra de comandes, un editor per escriure i editar codi, i diverses eines per a la depuració i la gestió de fitxers.

- El llenguatge de programació és de nivell alt i està dissenyat per a fer operacions amb matrius. Es fàcilment llegible i permet realitzar càlculs complexes amb codi concís.

- Hi ha una gran col·lecció de funcions matemàtiques per realitzar operacions de càlcul, estadística, àlgebra lineal, etc.

- Hi ha una amplia varietat de toolboxes, o caixes d'eines, que són col·leccions de funcions per aplicacions específiques com bé pot ser el processament de senyals o la visió per computadora.

- També es possible crear gràfiques en dos dimensions i en tres dimensions per tenir una millor visualització de les dades amb les que es treballa.

Una eina extra de la que disposa MATLAB és Simulink. Aquesta eina és molt important en l'execució d'aquest treball.

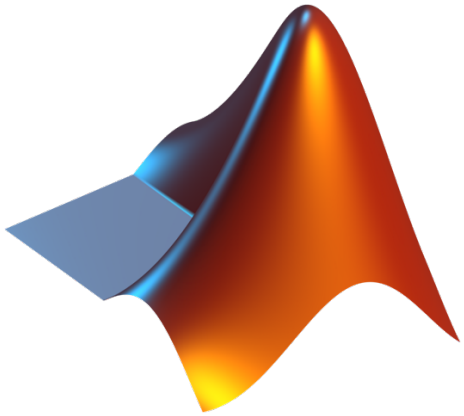


**Figura 8.** Logotip del programa informàtic MATLAB.

#### *1.3.6.1 Simulink.*

Simulink és un entorn de programació visual, desenvolupada per MathWorks (la mateixa empresa que va crear MATLAB), que funciona sobre l'entorn de MATLAB. L'entorn de programació és d'un nivell més alt que el del llenguatge M, i la programació amb aquesta eina està basada en diagrames de blocs.

Algunes característiques d'aquesta eina són les següents:

- Es proporciona una interfície gràfica on s'arrosseguen i es deixen caure els blocs per construir els models dels sistemes. Cada bloc representa un component del sistema (com per exemple les fonts dels senyals).

- Simulink permet que els models es puguin simular en temps real.

- Es disposa d'una àmplia varietat de biblioteques de blocs predefinits, però també es poden crear blocs personalitzats per l'usuari, que s'adaptin a les necessitats específiques del projecte.

- Els resultats de les simulacions de Simulink es poden visualitzar i analitzar fent ús de diverses eines.

Simulink es fa servir en múltiples camps de l'enginyeria entre els que es troben l'electrònica de potència i el processament de senyals. No obstant, un dels camps on més s'utilitza aquesta eina, i on s'aplicarà en aquest treball, és l'enginyeria de control, on es fan llaços de control i controladors que s'aplicaran al exoesquelet.

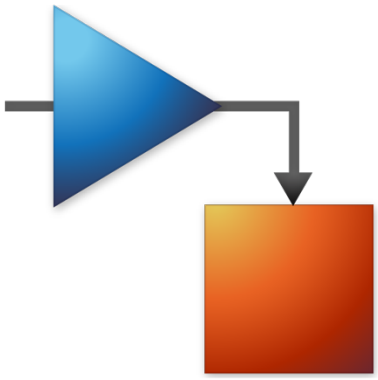


**Figura 9.** Logotip del programa informàtic Simulink.

### 1.3.7 Unity.

Una altra millora que en aquest treball s'aplicarà a l'exoesquelet és la creació de videojocs i la connexió de l'exoesquelet per poder controlar el videojoc. Per poder fer això, però, és necessari entendre la plataforma de creació que s'utilitzarà, en aquest cas Unity.

Unity és un motor de videojocs de codi obert, desenvolupat per Unity Technologies, que permet la creació i desenvolupament de videojocs de dos i de tres dimensions, entre altres experiències interactives. Aquest motor és molt utilitzat en la indústria dels videojocs per la seva potència, flexibilitat i facilitat d'ús.

El motor gràfic de Unity és un motor avançat que permet renderitzar gràfics en temps real i amb gran qualitat. Aquest motor també inclou físiques que permeten simular el comportament real dels objectes i consta d'eines d'animació per als personatges i els elements de l'escenari.

La programació en Unity es fa a través del llenguatge C#. Amb aquests scripts es pot controlar tot, des del moviment dels objectes i dels personatges fins la lògica del joc.


**Figura 10.** Logotip del programa informàtic Unity.

## 1.4 Normes i referencies.

En aquest capítol es troben les diferents normes i referencies usades per la realització del projecte.

### 1.4.1 Disposicions legals i normes aplicades.

Les normes que s'han aplicat en aquest treball són les següent:

- UNE 157001: Criteris generals per a l'elaboració formal dels documents que constitueixen un projecte tècnic.

### 1.4.2 Programes de càlcul.

Els programes informàtics que s'han utilitzat en aquest treball són el següents:

- Arduino
- MATLAB
- Simulink
- Unity
- Tinkercad

### 1.4.3 Pla de gestió de la qualitat aplicat durant la redacció del projecte.

El pla de gestió de la qualitat que s'ha portat a terme per assegurar que l'execució del projecte es correcta consisteix en revisar els codis que s'han programat i comprovar

que el seu funcionament és correcte. A més, a l'hora de redactar el projecte s'ha seguit la norma UNE 157001 per assegurar la correcta redacció del treball.

### 1.4.4 Bibliografia.

En aquest apartat es llistaran els llibres, revistes, manuals o altres textos que s'han consultat per a la realització del projecte:

EduExo, The Robotic Exoskeleton Kit Handbook and Tutorial, Beyond Robotics GmbH, Segona edició, 2019.

EduExo, The Robotic Exoskeleton Kit Muscle Control Extension Handbook and Tutorial, Beyond Robotics GmbH, Primera edició, 2017.

Consorcio OPENSURG, Robótica Médica notas prácticas para el aprendizaje de la robótica en bioingeniería, CYTED, 2013.

### 1.4.5 Altres referencies.

En aquest treball s'han consultat alguns documents disponibles a l'assignatura de Robòtica Mèdica del Grau d'Enginyeria Biomèdica. A més, també s'ha consultat una guia avançada del sensor EMG MyoWare 2.0, una versió més moderna i millorada del sensor EMG MyoWare que s'utilitza en aquest treball, però que serveix per entendre el funcionament del sensor.

## 1.5 Definicions i abreviatures.

En aquest apartat es llistaran totes les abreviatures que s'han utilitzat en aquest treball i les seves definicions:

EMG, Electromiografia.

PWM, Modulació per amplada de polsos (Pulse Width Modulation).

IDE, Entorn de Desenvolupament Integrat (Integrated Development Environment).

HIL, Hardware-In-Loop.

C/C++, Llenguatge de programació en la IDE d'Arduino.

C#, Llenguatge de programació en Unity.

EEPROM, Memòria de només lectura programable esborrable elèctricament (Electrically Erasable Programmable Read-Only Memory).

SRAM, Memòria estàtica d'accés aleatori (Static Random Access Memory).

$V_{out}$, Tensió de sortida del divisor de tensió del servomotor.

$V_{source}$, Tensió d'entrada del divisor de tensió del servomotor.

$R_G$, Resistència de guany.

SIG, Senyal processada del sensor EMG (Signal).

RAW, Senyal sense processar del sensor EMG.

A0, Entrada analògica 0 de la placa Arduino.

A3, Entrada analògica 3 de la placa Arduino.

A5, Entrada analògica 5 de la placa Arduino.

$\varphi_{colze}$, Angle del servomotor.

$a_0$, Valor en bits de la posició del servomotor a 0º.

$a_{90}$, Valor en bits de la posició del servomotor a 90º.

## 1.6 Requisits de disseny.

En aquest apartat es descriuen les dades de partida d'aquest treball, és a dir, els sensors i actuadors que s'utilitzaran.

### 1.6.1 Servomotor.

L'actuador que es fa servir en aquest kit es un servomotor elèctric amb un engranatge i electrònica de control dins de la carcassa del motor. A més, el servomotor també conté un potenciòmetre integrat que permet mesurar la posició a través de l'angle de rotació de l'eix del motor. El servomotor consta de 4 cables: dos cables d'entrada per a la font d'alimentació (els cables de color vermell i marró), un cable d'entrada per al senyal de control del servomotor (de color taronja), i un cable de sortida per a la senyal de l'angle que es troba al sensor de posició integrat (de color blanc).

Aquest servomotor té un sistema de control integrat que pot fer un control bàsic (de baix nivell). Un exemple d'aquest tipus de control és moure la posició del servomotor a un angle desitjat.

L'angle al que es desitja posicionar el servomotor es codifica a través de la modulació per amplada de polsos, és a dir, a partir d'un PWM. En la codificació per la modulació per amplada de polsos l'angle desitjat al que es vol posicionar el motor és proporcional a l'amplada d'un pols elèctric. Aquesta senyal s'envia a intervals regulars de temps per mantenir la posició del motor en l'angle desitjat. El servomotor que es fa servir en aquest treball té un període de 20 ms.

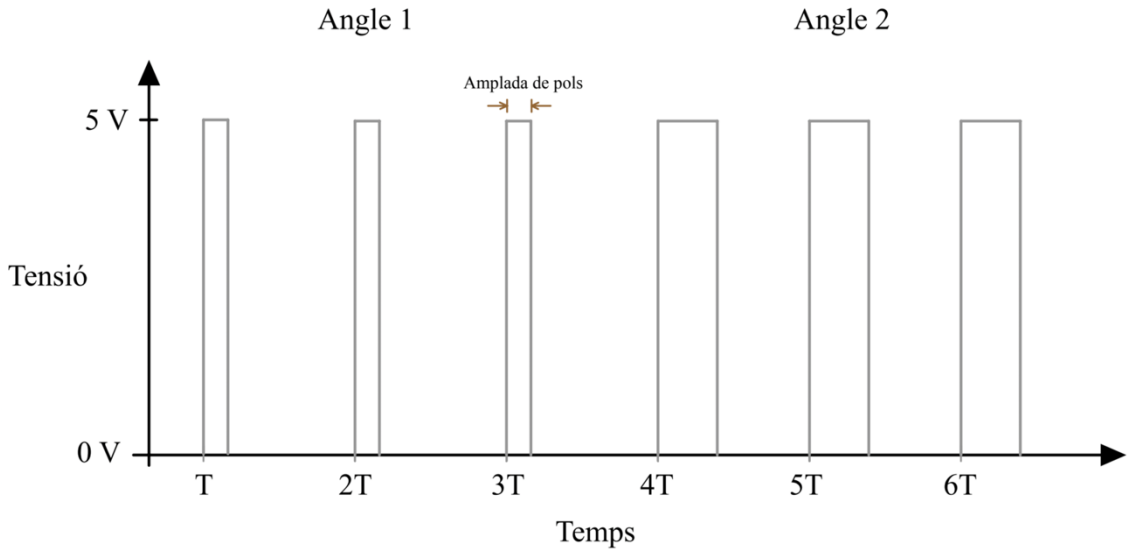



**Figura 11.** Exemple de funcionament d'un PWM en funció de l'angle del motor.

A més de poder posicionar el servomotor amb un PWM, també es pot enviar el senyal de posició del servomotor al microcontrolador, en aquest cas la placa Arduino. El sensor de posició és un potenciòmetre analògic configurat com un divisor de tensió ajustable de tres contactes, on dos contactes es troben connectats a la font d'alimentació i el tercer contacte és un contacte mòbil que rota al llarg de l'eix del motor. En la següent equació es pot veure com afecta el canvi en el tercer contacte del potenciòmetre:

$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{source} \tag{5}$$

De l'equació anterior es pot veure que les resistències $R_1$ i $R_2$ canvien en funció de la posició del tercer contacte del potenciòmetre, per tant el ratio entre les dos resistències també canvia, afectant a la tensió de sortida $V_{out}$ mesurada, que representa la senyal de sortida. Aquesta senyal de sortida serveix per mesurar la posició de l'angle del motor.

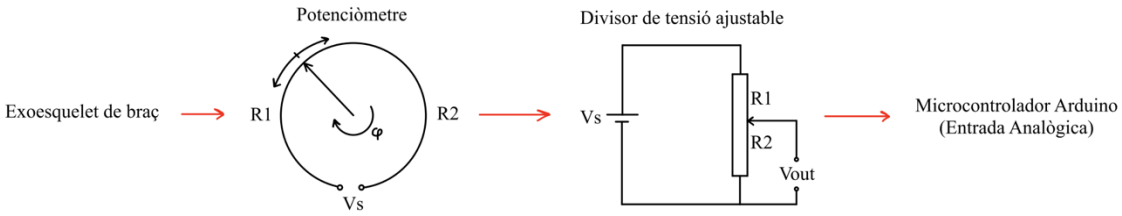



**Figura 12.** Procediment per llegir la posició del motor.

A l'hora de fer les connexions es faran d'acord al següent esquema:

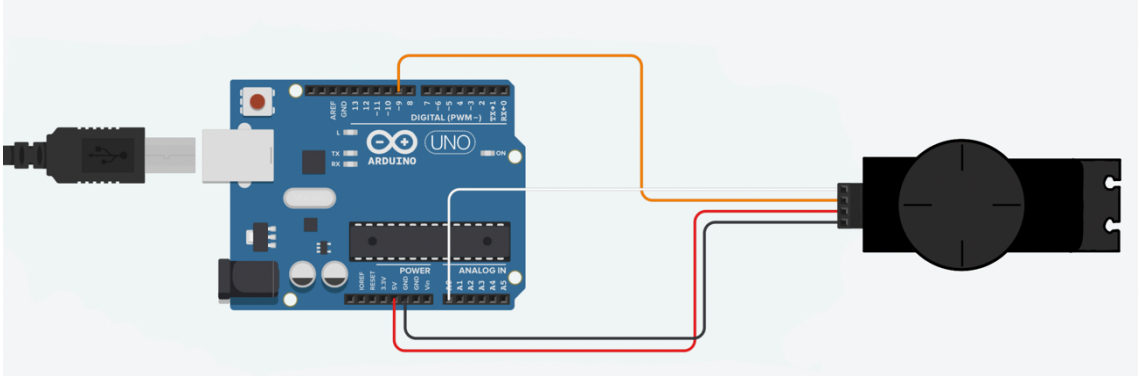



**Figura 13.** Connexions del servomotor amb la placa Arduino.

De la imatge anterior es pot observar que el cable vermell es connecta als 5 V de la placa Arduino i el cable marró es connecta al terra o GND. El cable de color taronja serveix per controlar la posició del servomotor i es connecta a un del pins digitals que proporcionen una sortida amb PWM. El cable blanc serveix per llegir l'angle del servomotor i es connecta en una de les entrades analògiques de la placa Arduino.

### 1.6.2 Sensor de força.

El sensor de força mesura la força d'interacció entre l'exoesquelet i la persona que el porta. Es pot utilitzar per registrar la interacció que hi ha hagut i per controlar l'exoesquelet.

Per tenir la capacitat de mesurar la força que s'està aplicant, primer de tot es necessari transformar la força física, que es troba mesurada en Newtons, en una senyal elèctrica, que es troba mesurada en volts. Amb aquesta transformació les dades podran ser enregistrades en el microcontrolador de l'Arduino.

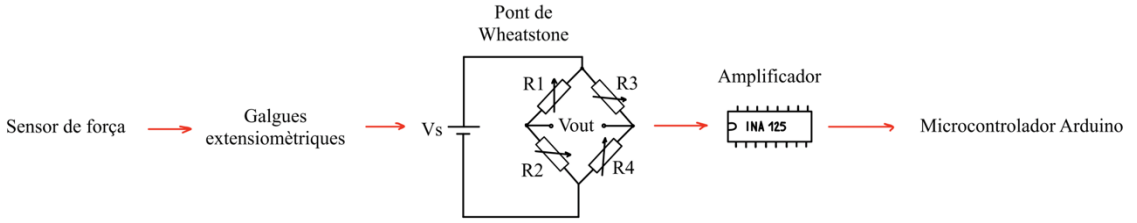



**Figura 14.** Procediment per llegir la força aplicada.

El sensor de força que es fa servir és una peça de metall que té uns resistors elèctrics especials que s'anomenen galgues extensiomètriques. Les galgues extensiomètriques canvien la seva resistència elèctrica quan hi ha esforços i petites deformacions. Quan s'aplica una força i el metall es deforma, de forma imperceptible, les galgues extensiomètriques canvien la seva longitud, el que causa un canvi en la resistència elèctrica.

Normalment, varis d'aquests resistors es troben connectats en forma de pont de Wheatstone, que permet fer mesures dels canvis de resistència elèctrica de forma precisa. Quan les resistències canvien el seu valor, la tensió de sortida del circuit també canvia, i a la vegada, aquesta tensió de sortida és la senyal del sensor de tensió, que és proporcional a la força aplicada.

No obstant, la tensió de sortida del pont de Wheatstone és massa petita i s'ha de connectar a un amplificador que amplifica el senyal de la tensió de sortida (i per tant facilita la lectura de la força aplicada) i que es llegida pel microcontrolador de l'Arduino. En aquest cas, l'amplificador del que es farà ús és l'INA 125.

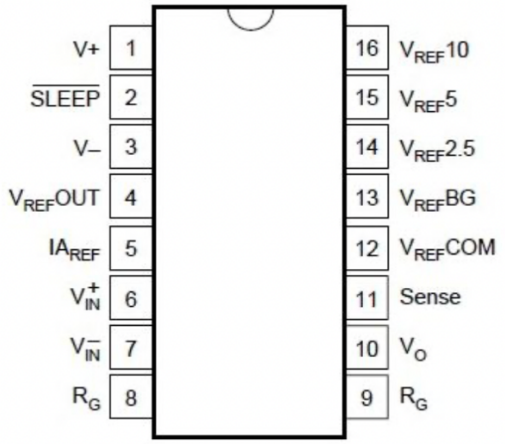



**Figura 15.** Esquema dels pins de l'amplificador INA 125

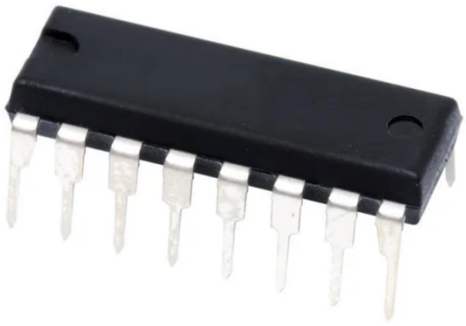


**Figura 16.** Amplificador INA 125

A la següent figura es pot observar les connexions que s'han realitzat al pont de Wheatstone i a l'amplificador.

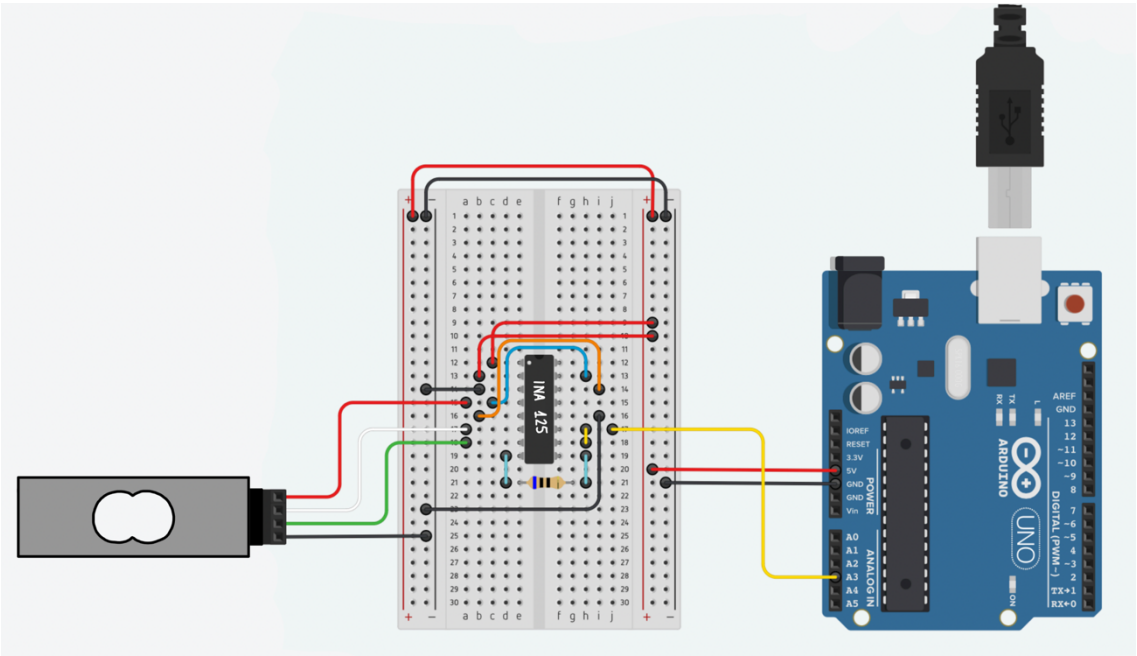



**Figura 17.** Connexions del sensor de força amb l'amplificador i la placa Arduino.

A la figura anterior es pot observar a l'esquerra el pont de Wheatstone, al centre l'amplificador i a la dreta les connexions a l'Arduino. En quant a l'amplificador a la següent taula es pot veure a que correspon cada pin:

| | |
|---|---|
| **Pin 1** | V+ |
| **Pin 2** | SLEEP |
| **Pin 3** | V- |
| **Pin 4** | $V_{REF}$OUT |
| **Pin 5** | $IA_{REF}$ |
| **Pin 6** | $V_{IN}^{+}$ |
| **Pin 7** | $V_{IN}^{-}$ |
| **Pin 8** | $R_G$ |
| **Pin 9** | $R_G$ |
| **Pin 10** | $V_O$ |
| **Pin 11** | Sense |
| **Pin 12** | $V_{REF}$COM |
| **Pin 13** | - |
| **Pin 14** | $V_{REF}$2.5 |
| **Pin 15** | $V_{REF}$5 |
| **Pin 16** | - |

**Taula 2.** Connexions dels pins de l'amplificador.

Els pins 1 i 2 de l'amplificador es connecten a la font de tensió, on el pin 1 és la font d'alimentació del chip i el pin 2 activa el chip si aquest es troba en *HIGH*, és a dir, si s'alimenta a 5 V. El pin 3 és el terra de l'amplificador, i el pin 4 és la tensió d'alimentació del sensor de força, que es connecta al pin 15 que és la tensió de referència. Això el que provoca és que hi hagi una font de tensió precisa per al pont de Wheatstone. En el pin 5 es troba la referència de l'amplificador d'instrumentació, que es connecta al pin 14 per establir la tensió de referència a 2,5 V. Els pins 6 i 7 es connecten a la sortida del pont de Wheatstone.

Els pins 8 i 9 es connecten entre ells amb un resistor intercalat anomenat $R_G$. El valor de la resistència $R_G$ s'utilitza per establir el guany de l'amplificador. La resistència

del resistor que es farà servir és de 60,4 Ω, així es posarà el guany de l'amplificador a 1000. Fer servir una resistència més petita permet tenir un guany major però també tindre un soroll més gran.

Els pins 10 i 11 es troben connectats i proporcionen la senyal de sortida amplificada, que es connectarà a una entrada analògica de la placa Arduino. La tensió de referencia del pin 12 es connecta a terra, i els pins 13 i 16 es deixen en circuit obert, sense connectar-hi res.

### 1.6.3 Sensor EMG.

El sensor electromiogràfic, o EMG, es basa en l'activitat muscular. En el cas d'aquest treball, el sensor mesura l'activitat muscular al bíceps per controlar els moviments de l'exoesquelet. El senyal del sensor EMG es llegeix a través d'una de les entrades analògiques de l'Arduino. El sensor EMG que es fa servir en aquest treball té diferents formes de donar el senyal: es pot obtenir un senyal processat (SIG), on el senyal es més suau, o també es pot obtenir el senyal directe (RAW), sense cap modificació al senyal muscular. En aquest projecte es farà ús del senyat processat SIG, que ja es troba amplificat.

El sensor que s'ha fet servir en aquest projecte és el sensor MyoWare. En les següents imatges es pot veure el sensor per la seva cara anterior i posterior:

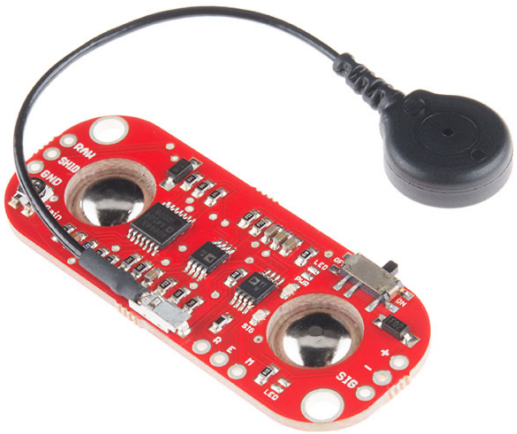


**Figura 18.** Cara anterior del sensor EMG MyoWare.


**Figura 19.** Cara posterior del sensor EMG MyoWare.

Hi ha diverses configuracions a l'hora de connectar els elèctrodes:

- Configuració Monopolar: En aquesta configuració es fan servir 2 elèctrodes. Aquí un únic elèctrode d'entrada es col·loca sobre el grup muscular que on es vol mesurar l'activitat elèctrica, i l'altre elèctrode, el de referència, és col·loca en una localització adjacent elèctricament neutre.

- Configuració Bipolar: En aquesta configuració es fan servir 3 elèctrodes. Aquí 2 elèctrodes d'entrada es col·loquen en el grup muscular on es vol

mesurar l'activitat elèctrica. El primer elèctrode es col·loca prop de la meitat del múscul, i el segon elèctrode es col·loca a una distància d'entre 1 a 3 centímetres del primer elèctrode. El tercer elèctrode, el de referència, es col·loca en una zona adjacent elèctricament neutre. En aquesta configuració la diferència de tensió entre el dos elèctrodes s'amplifica respecte a l'elèctrode de referència. A més, un avantatge d'aquesta configuració és que s'elimina el soroll comú entre els dos elèctrodes a causa de la relació de rebuig en mode comú (CMRR) de l'amplificador. UN sensor EMG bipolar produeix un senyal EMG més net i amb una relació senyal-soroll (SNR) més gran.

Si es vol consultar més informació sobre el funcionament del sensor EMG, a l'apartat *2.3 Guia avançada del sensor EMG MyoWare 2.0* dels Annexes es pot trobar informació sobre el sensor EMG MyoWare 2.0, una versió millorada del sensor que es fa servir en aquest treball, però on s'explica el funcionament, que és el mateix que el del sensor que s'utilitza en el projecte.

En aquest cas, el sensor funciona amb 3 elèctrodes (Configuració Bipolar): 2 que es col·loquen separats en el bíceps que mesuren la diferencia de potencial, i un altre que es connecta lluny del múscul (entre el bíceps i el tríceps) que fa la funció de referència.

Per connectar el sensor a l'Arduino s'han de fer 3 connexions. Dos cables són referents a l'alimentació del sensor, i el tercer cable és un senyal analògic que es connecta en una de les entrades analògiques de l'Arduino.

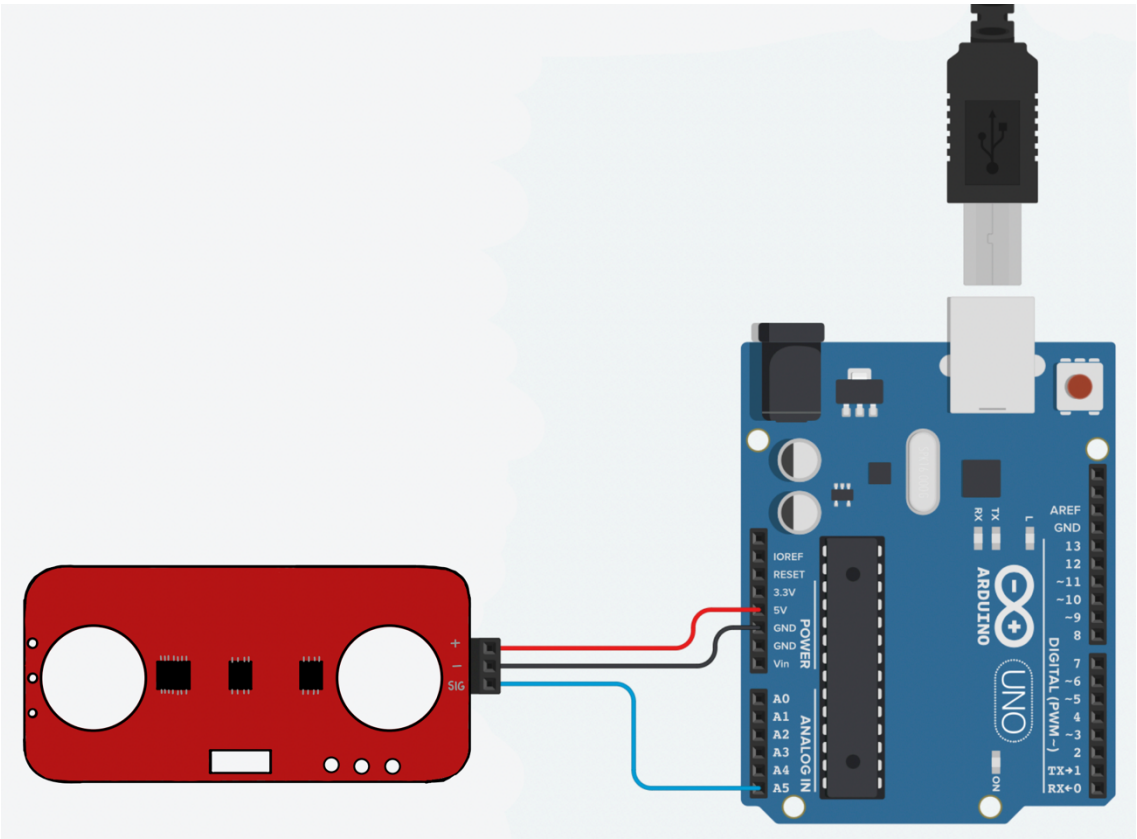



**Figura 20.** Connexions del sensor EMG amb la placa Arduino.

## 1.7 Anàlisi de solucions.

En aquest apartat s'explicaran totes les possibles solucions que s'han tingut en compte a l'hora de fer el treball i es veuran diversos programes que s'han fet per entendre el funcionament de l'exoesquelet de braç. Primer de tot es faran els programes en la IDE d'Arduino i s'explicaran. Una vegada acabat amb els programes d'Arduino es passarà a fer els programes en Simulink de MATLAB, i finalment el videojoc en Unity.

### 1.7.1 Programació en Arduino.

#### *1.7.1.1 Servomotor.*

El primer programa que s'ha de fer és un programa molt senzill per poder mesurar la posició a la que es troba el braç robòtic a partir de l'angle de rotació que hi ha aplicat al motor. Aquest programa és el que es troba en l'apartat *2.1.1 Lectura de l'angle del servomotor en bits* dels Annexes.

Amb el codi anterior el que es fa és llegir la informació de la posició que hi arriba a través de l'entrada analògica A0 de la placa Arduino. Aquesta dada es guarda en una variable anomenada "posIs" que indica la posició del servomotor, i es mostra per pantalla.

```
La posició és: 214
La posició és: 210
La posició és: 204
La posició és: 198
La posició és: 193
La posició és: 187
La posició és: 183
La posició és: 179
La posició és: 175
La posició és: 171
La posició és: 168
```

**Figura 21.** Posicions del servomotor mesurades.

El següent pas que s'ha de fer amb el servomotor és el seu calibratge. Les dades que es reben amb el programa anterior són valors de bit que es troben entre 0 i 1023. Aquest valors no són gens representatius per tant s'han de passar a una unitat que sigui molt més fàcil d'interpretar, en aquest cas els graus sexagesimals.

El primer que s'ha de fer es moure l'exoesquelet a dos posicions conegudes. En aquest cas les posicions que s'escullen són quan el braç es troba completament estirat, on l'angle seria de 0º, i quan el braç es troba formant un angle de 90º.

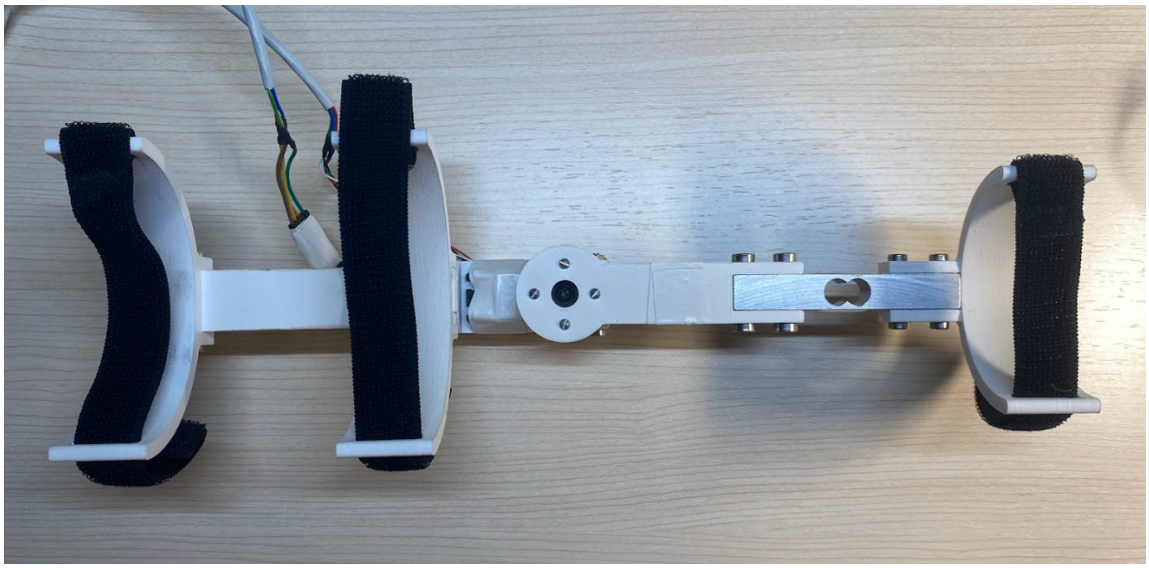


**Figura 22.** Exoesquelet en posició de 0º.

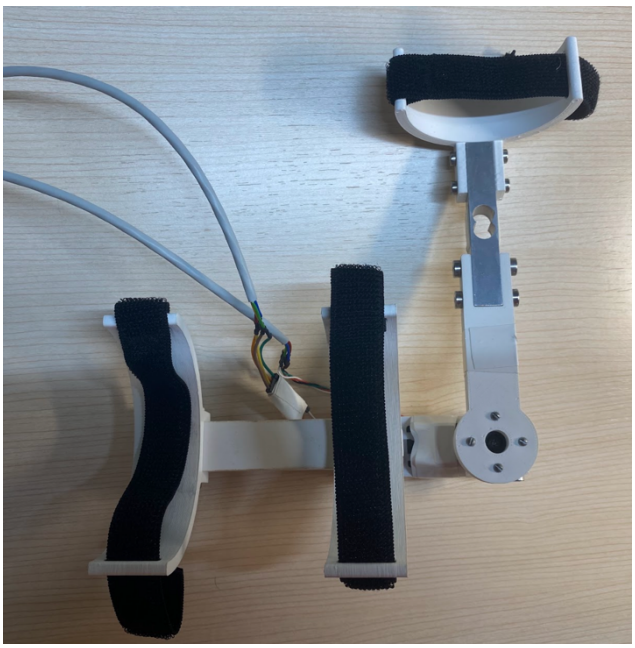


**Figura 23.** Exoesquelet en posició de 90º.

Amb l'exoesquelet col·locat en aquestes posicions es pot llegir quin és el valor en bits de cada posició amb el codi anterior. S'anota cada posició en les següents variables $a_0$ i $a_{90}$.

Ara es fa una equació que permet passar dels valors en bits als valors en graus sexagesimals, que representen l'angle de l'articulació $\varphi_{colze}$. L'equació és la següent:

$$\varphi_{colze} = \frac{90° - 0°}{a_{90} - a_0} \cdot (pos_{is} - a_0) \quad (6)$$

Els valor de $a_0$ i $a_{90}$ obtinguts són els següents d'acord al resultat de la simulació:

| | |
|---|---|
| **$a_0$** | 130 |
| **$a_{90}$** | 316 |

**Taula 3.** Variables de la posició de l'exoesquelet de braç a 0º i a 90º.

```
La posició és: 131
La posició és: 131
La posició és: 131
La posició és: 130
La posició és: 130
La posició és: 131
La posició és: 130
La posició és: 130
La posició és: 131
La posició és: 130
La posició és: 131
La posició és: 131
La posició és: 130
La posició és: 131
La posició és: 131
La posició és: 131
```

**Figura 24.** Valors mesurats amb l'exoesquelet a 0º.

```
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
La posició és: 316
```

**Figura 25.** Valors mesurats amb l'exoesquelet a 90º.

I ara si es substitueixen les variables $a_0$ i $a_{90}$ pels valors obtinguts es té la següent expressió:

$$\varphi_{colze} = \frac{90° - 0°}{316 - 130} \cdot (pos_{is} - 130) \quad (7)$$

Ara s'escriu el següent codi, molt similar a l'anterior, on s'aplica l'equació 7 per passar de valors de bit a valors en graus sexagesimals. El codi es troba a l'apartat *2.1.2 Lectura de l'angle del servomotor en graus sexagesimals* dels Annexes.

El que es fa és afegir el càlcul guardant el resultat en una variable anomenada "posIsDeg" i posteriorment es mostreja aquesta variable. Un detall important és que la variable "posIs" està declarada com un enter, ja que s'utilitzen números de bit sencers, però la variable "posIsDeg" s'ha de declarar com una variable amb decimals, ja que es poden obtenir graus que no siguin sencers.

Una vegada s'ha calibrat el motor de l'exoesquelet de braç ja es pot programar el servomotor per a que faci diferents funcions. El que es fa a continuació és un codi on es posiciona el braç a l'angle que es desitja. El codi és troba a l'apartat *2.1.3 Posicionament del servomotor* dels Annexes.

Per posicionar l'exoesquelet a la posició desitjada es fa servir la instrucció "myservo.write()".

#### *1.7.1.2 Sensor de força.*

El programa més simple que es pot fer amb el sensor de força és el de llegir els valors que mesura el sensor i mostrejar-los per pantalla. Aquest programa és molt similar al programa on es llegeix la posició del servomotor. El programa és el de l'apartat *2.1.4 Lectura del sensor de força* dels Annexes.

En aquest cas es llegeix la força que mesura el sensor a través de l'entrada analògica A3 de la placa Arduino. També és declara una variable anomenada "forceIs" on es guarda el valor de la força mesurada. Com en el cas del servomotor es mostra la força mesurada per pantalla.

El següent pas és el de calibrar el sensor de força. Per aconseguir això s'escriu el codi de l'apartat *2.1.5 Calibratge del sensor de força* dels Annexes

En aquest cas es declara una variable constant de tipus sencer anomenada "forceOffset" de valor 442. Aquest valor s'obté deixant l'exoesquelet en repòs sense aplicar cap força. D'aquesta manera s'aplica un offset a la força que permet conèixer el sentit de la força (si és força de flexió o força d'extensió) a partir del signe que té. Per aplicar aquest offset s'aplica la següent fórmula:

$$forceIs = forceIs - forceOffset \quad (8)$$

On es substitueix forceOffset pel valor mesurat de 442:

$$forceIs = forceIs - 442 \quad (9)$$

A la següent imatge es pot veure el valor de la força offset mesurat amb el sensor:

```
La força es: 441
La força es: 443
La força es: 442
La força es: 441
La força es: 442
La força es: 442
La força es: 442
La força es: 442
La força es: 442
La força es: 442
La força es: 442
La força es: 442
La força es: 442
La força es: 442
```

**Figura 26.** Força mesurada en repòs.

#### *1.7.1.3 Sensor EMG.*

Amb aquest sensor, al igual que en els casos anteriors, el primer pas és fer un programa molt senzill per llegir els valors que mesura. El codi d'aquest programa és troba a l'apartat *2.1.6 Lectura del sensor EMG* dels Annexes.

El valor del sensor EMG que es mesura es fa a través de l'entrada analògica A5 de la placa Arduino. Després de mesurar el valor del sensor es mostreja per pantalla. A partir d'aquest codi es pot establir quin és el valor de l'activitat elèctrica d'un usuari en repòs i el valor d'aquesta activitat elèctrica quan es fa un esforç. Aquests valors que s'han mesurat són els següents:

| | |
|---|---|
| **Activitat elèctrica en repòs** | 170 |
| **Activitat elèctrica amb un esforç** | 985 |

**Taula 4.** Valors de l'activitat elèctrica quan l'usuari té el braç en repòs i fent un esforç.

Tal i com es pot veure a les següents imatges:

```
senyal de l'activitat elèctrica mesurada pel sensor EMG és: 985
senyal de l'activitat elèctrica mesurada pel sensor EMG és: 985
senyal de l'activitat elèctrica mesurada pel sensor EMG és: 985
senyal de l'activitat elèctrica mesurada pel sensor EMG és: 985
senyal de l'activitat elèctrica mesurada pel sensor EMG és: 985
senyal de l'activitat elèctrica mesurada pel sensor EMG és: 984
senyal de l'activitat elèctrica mesurada pel sensor EMG és: 985
senyal de l'activitat elèctrica mesurada pel sensor EMG és: 985
senyal de l'activitat elèctrica mesurada pel sensor EMG és: 985
```

**Figura 27.** Valor mesurat aplicant un esforç.

```
El senyal de l'activitat elèctrica mesurada pel sensor EMG és: 171
El senyal de l'activitat elèctrica mesurada pel sensor EMG és: 173
El senyal de l'activitat elèctrica mesurada pel sensor EMG és: 178
El senyal de l'activitat elèctrica mesurada pel sensor EMG és: 170
El senyal de l'activitat elèctrica mesurada pel sensor EMG és: 172
El senyal de l'activitat elèctrica mesurada pel sensor EMG és: 176
El senyal de l'activitat elèctrica mesurada pel sensor EMG és: 168
El senyal de l'activitat elèctrica mesurada pel sensor EMG és: 171
```

**Figura 28.** Valor mesurat en repòs.

A partir d'aquests valors és pot establir i calcular un llindar per indicar quan activar el moviment de l'exoesquelet a partir de l'activitat elèctrica. Aquest llindar es calcularà amb el valor mig entre el valor mesurat en repòs i el valor mesurat fent un esforç de la següent forma:

$$EMG_{llindar} = \frac{Valor\ màxim + Valor\ mínim}{2} \tag{10}$$

Amb els valors mesurat de la taula anterior i amb l'expressió anterior és pot calcular el valor llindar:

$$EMG_{llindar} = \frac{985 + 170}{2} = 577{,}5 \simeq 578 \tag{11}$$

Amb aquest valor es pot fer un codi com el que es troba a l'apartat *2.1.7 Controlador de l'exoesquelet amb el sensor EMG* dels Annexes, on si es supera el valor llindar mesurat pel sensor EMG el servomotor es col·locarà en una posició establerta, i si no es supera aquest valor llindar el servomotor es desacobla del seu pin.

### 1.7.2 Simulink.

El següent esglaó en dificultat per controlar l'exoesquelet és el de programar en Simulink. Amb aquesta forma de programació es pretén fer un sistema més senzill de comprendre i de modificar, sense línies de codi per mig. A més, amb aquest programa es possible fer llaços de control, tant senzills com complexos, de forma molt més entenedora que si es fessin amb la IDE d'Arduino.

El primer que s'ha de fer per poder programar en la placa d'Arduino programes de Simulink és descarregar els Add-On d'Arduino. Els que s'han fet servir en aquest treball són els següents: *MATLAB Support Package for Arduino Hardware* i *Simulink Support Package for Arduino Hardware*.

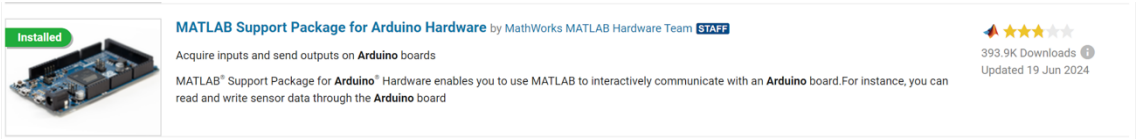



**Figura 29.** Add-On MATLAB Support Package for Arduino Hardware.

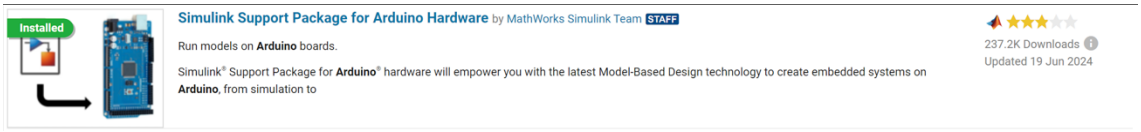



**Figura 30.** Add-On Simulink Support Package for Arduino Hardware.

Una vegada instal·lats aquests Add-On ja es poden començar a fer els programes en Simulink.

#### *1.7.2.1 Servomotor.*

El primer programa que es farà és el de llegir la posició a la que es troba el servomotor. Per fer això es programarà el següent diagrama de blocs:

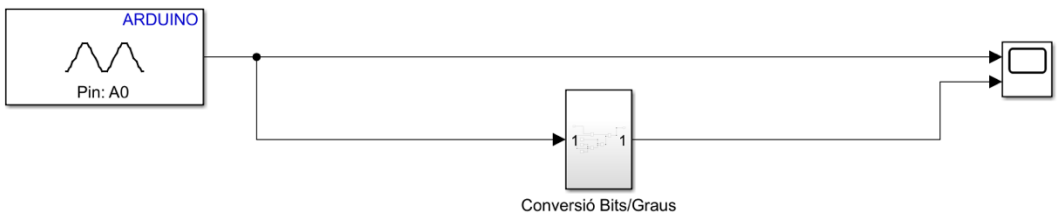



**Figura 31.** Diagrama de blocs per a la lectura de la posició del servomotor.

A l'anterior figura és poden veure diversos blocs. El primer bloc de l'esquerra és l'entrada analògica, on s'indica per quin pin arriba la informació de la posició del servomotor. El següent bloc que es pot veure és "Conversió Bits/Graus", que és un bloc propi que fa la conversió de dades de bits a graus sexagesimal. En la següent imatge es pot veure com està format aquest bloc:

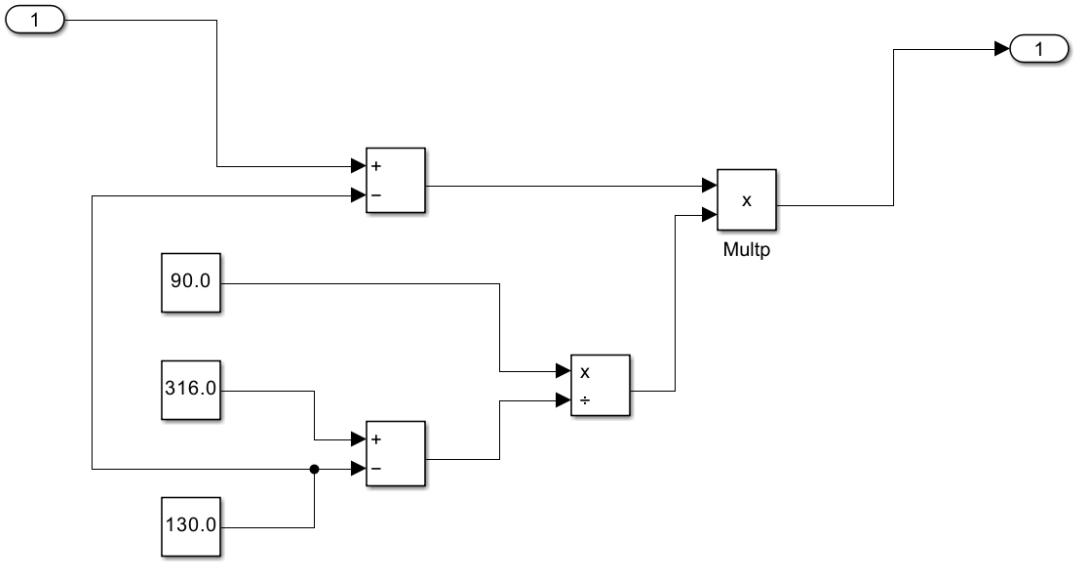



**Figura 32.** Diagrama de blocs del bloc de Conversió Bits/Graus.

Finalment, el bloc de l'esquerra és un "scope" on es pot veure, en gràfiques, els valors que assoleix la posició del motor de l'exoesquelet tant en bits com en graus.

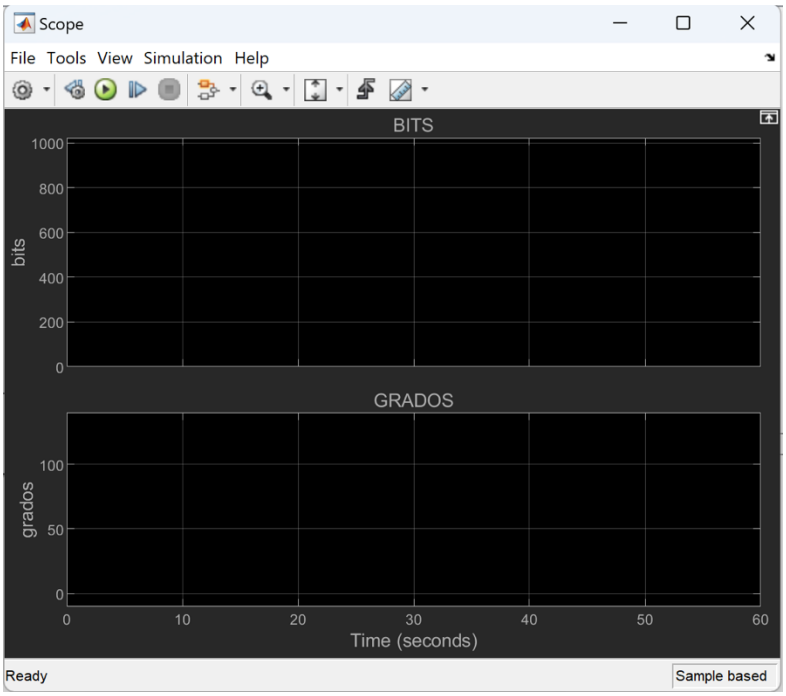



**Figura 33.** Exemple de la pantalla del scope.

Amb el diagrama de blocs anterior es pot veure l'evolució de la posició del servomotor mesurada pel seu sensor de posició en la següent imatge:

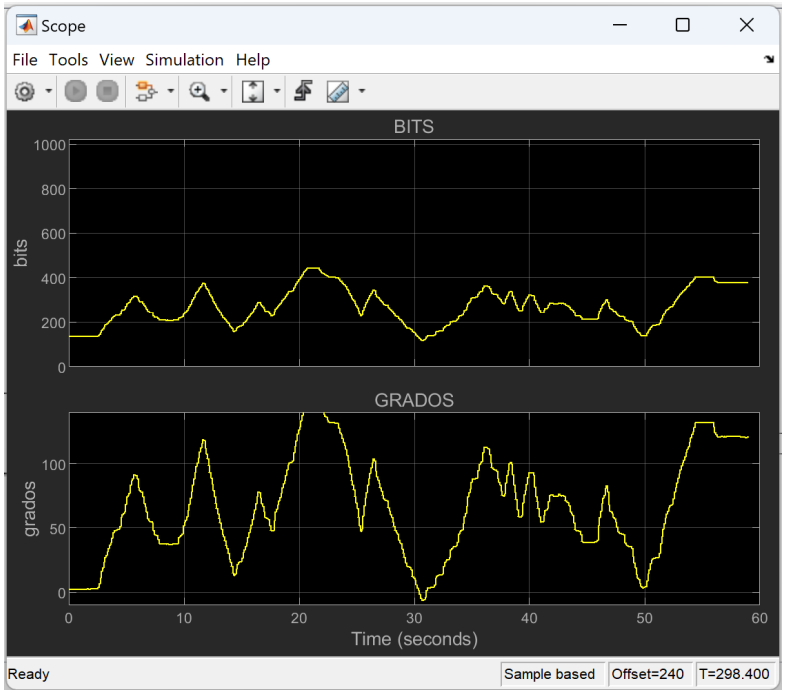



**Figura 34.** Evolució de la posició del servomotor en bits i en graus sexagesimals.

El següent programa serveix per posicionar el servomotor a la posició desitjada. Per fer això s'agafa la base del programa anterior i s'amplia amb uns blocs més. El primer bloc que s'afegeix és una constant de valor 1. A continuació es connecta un bloc de guany variable que podrà prendre valors dins del rang de 12 a 110 graus. Aquest rang de valors són els que s'han imposat en aquest treball per evitar que l'usuari prengui mal quan es modifica la posició de l'exoesquelet de braç. I finalment, és connecta un bloc de "Standard Servo Write", on s'escriu el pin on està connectat el servomotor, i que serveix per escriure la posició a la que es vol posicionar el braç robòtic.

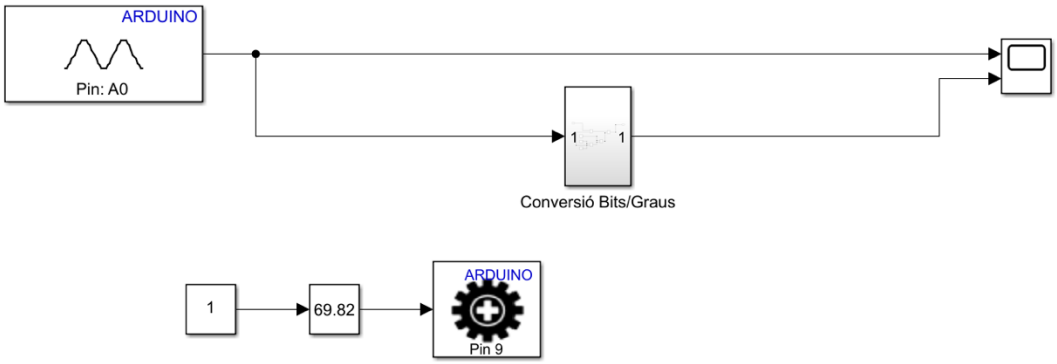



**Figura 35.** Diagrama de blocs per posicionar el servomotor a la posició (en graus) desitjada.

A continuació es pot veure la configuració que s'ha fet al bloc de guany variable:

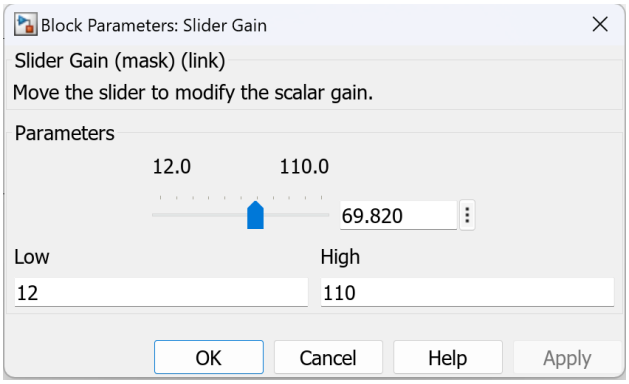



**Figura 36.** Configuració del bloc de guany variable.

A partir d'aquest programa es pot veure el següent conjunt d'imatges, on es veu una seqüència de com es modifica la posició del servomotor amb la variació del bloc de guany variable:

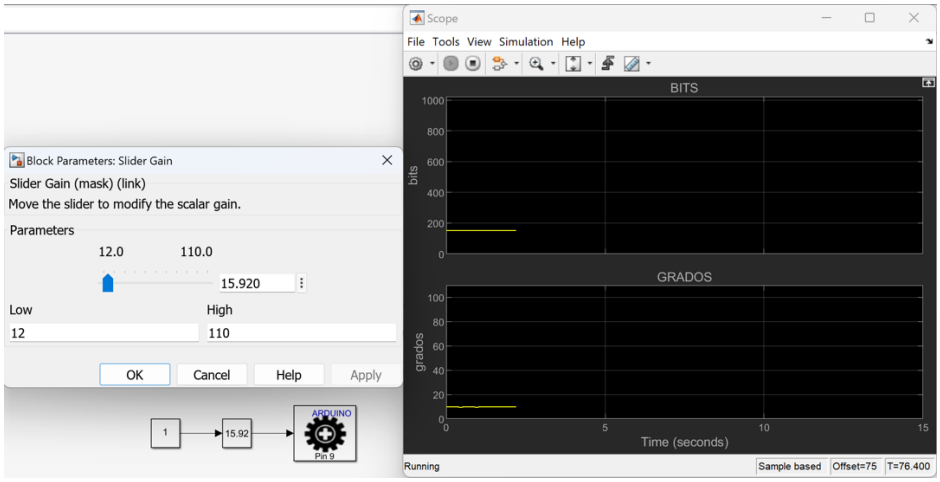



**Figura 37.** Variació de la posició del servomotor amb un guany de 15,92.

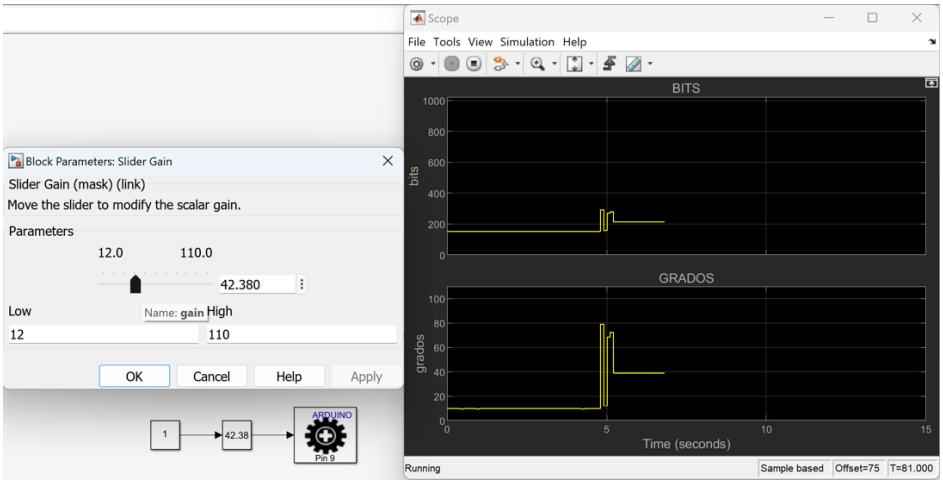



**Figura 38.** Variació de la posició del servomotor amb un guany de 42,38.

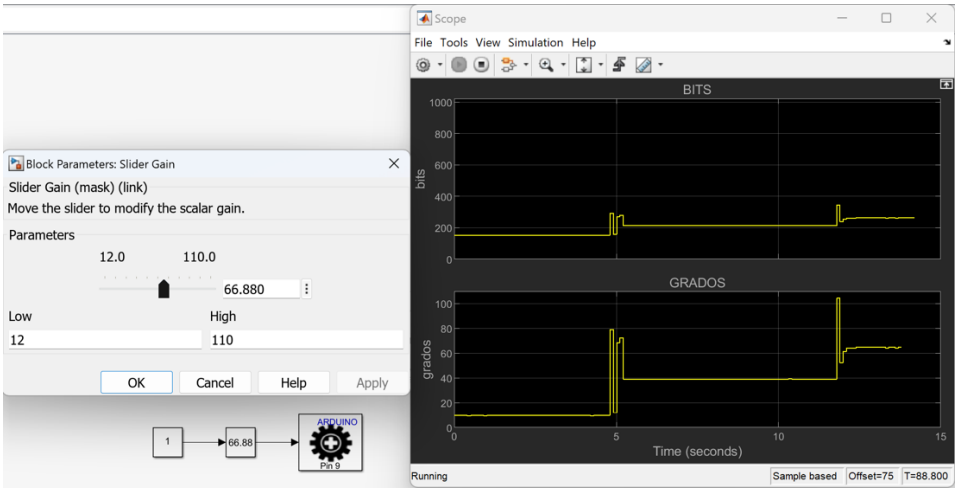



**Figura 39.** Variació de la posició del servomotor amb un guany de 66,88.

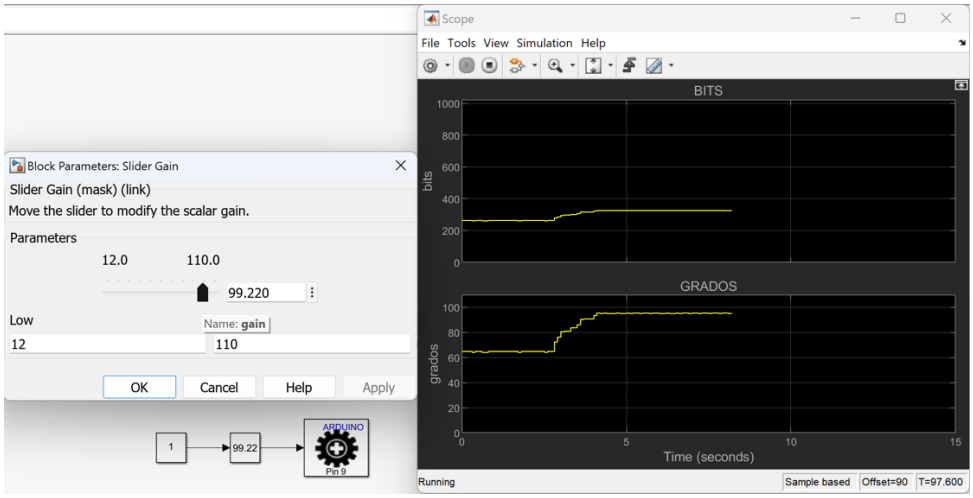



**Figura 40.** Variació de la posició del servomotor amb un guany de 99,22.

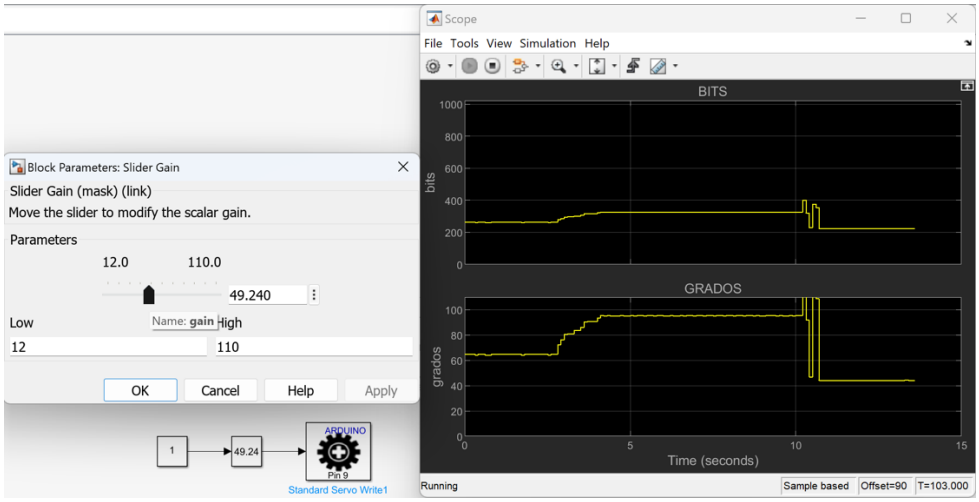



**Figura 41.** Variació de la posició del servomotor amb un guany de 49,24.

### *1.7.2.2 Sensor de força.*

El tercer programa que és fa implementa el calibratge del sensor de força i la seva lectura. Amb el bloc de "Analog Input" és llegeix l'entrada A3 a la placa d'Arduino i es resta l'offset de 442 per calibrar el sensor. El valor mesurat es mostreja per un display, on és pot veure el valor numèric que es rep i també és pot veure en forma de gràfica a través del bloc "scope".

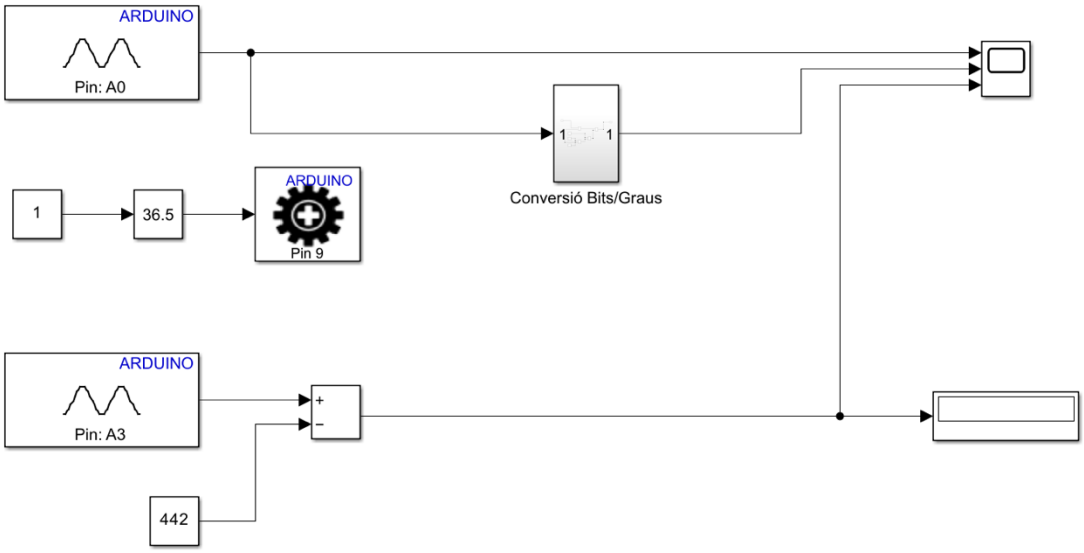



**Figura 42.** Diagrama de blocs del calibratge i lectura del sensor de força.

Amb el programa anterior es pot veure la següent imatge on s'observa la variació de la força mesurada pel sensor de força:

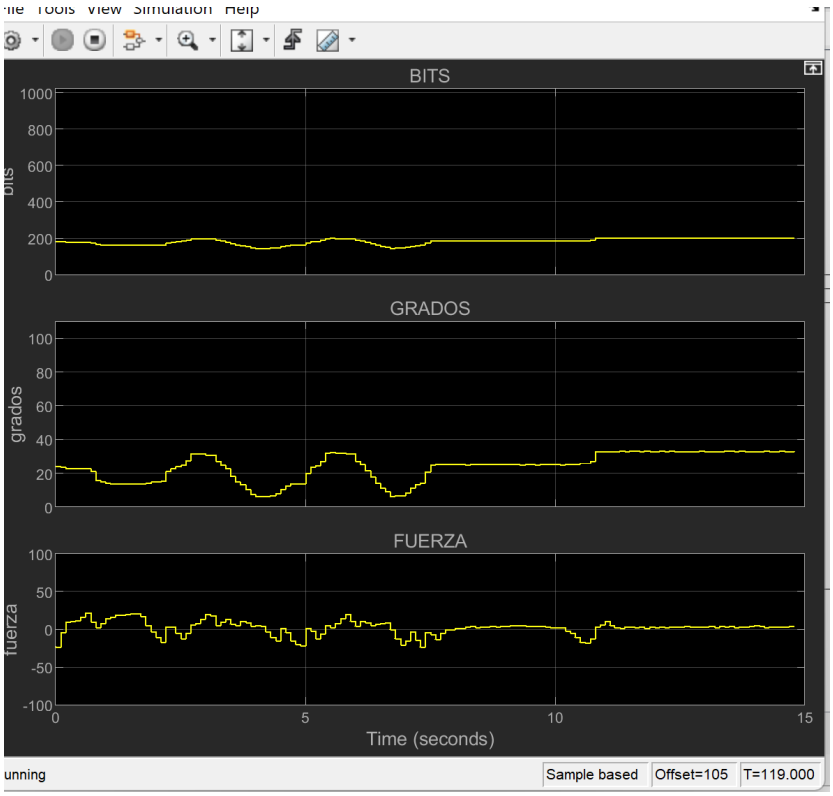



**Figura 43.** Evolució de la força mesurada pel sensor de força.

### *1.7.2.3 Sensor EMG.*

El següent programa serveix per llegir la senyal elèctrica mesurada pel sensor EMG. Amb aquest programa de Simulink es llegeix el pin d'entrada analògica A5, on es troba connectat el sensor EMG, i es mostra la lectura en un display, amb el valor exacte mesurat, i en un scope, amb la gràfica dels valors mesurats. A continuació es pot veure el circuit que s'ha realitzat:

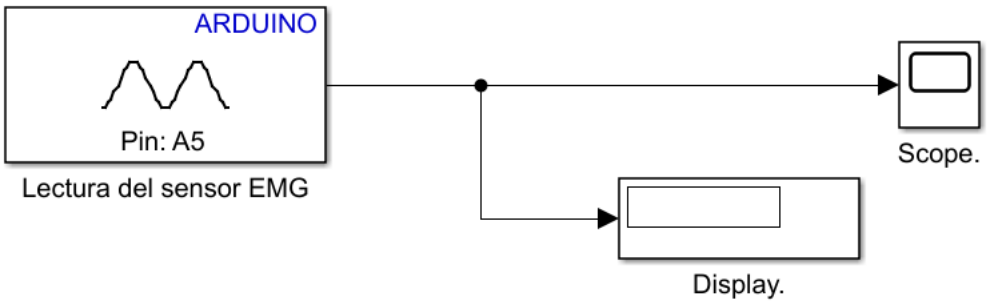



**Figura 44.** Diagrama de blocs de la lectura del sensor EMG.

A la següent imatge es pot veure l'evolució de l'activitat elèctrica mesurada pel sensor EMG:

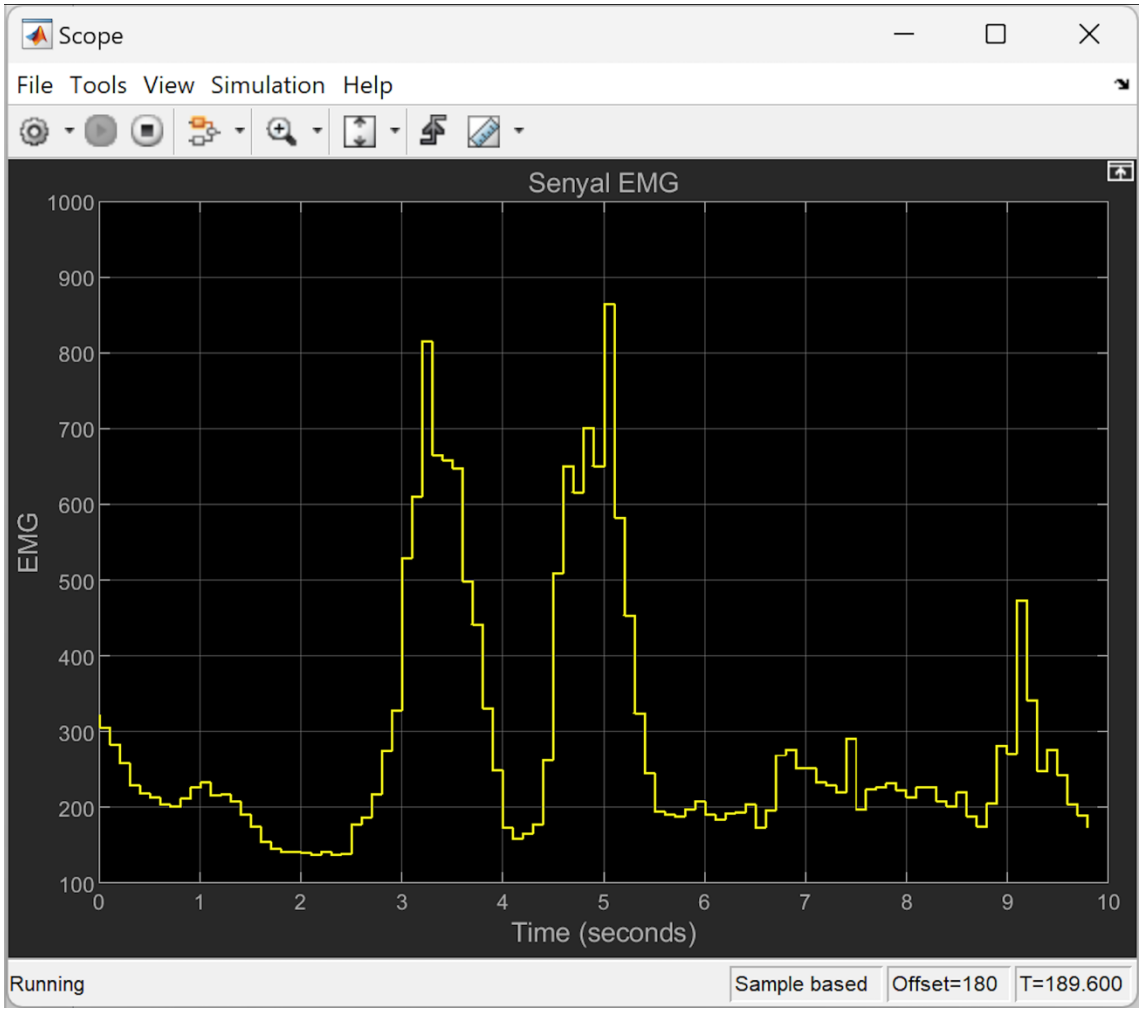



**Figura 45.** Gràfica de l'evolució de l'activitat elèctrica mesurada pel sensor EMG.

### 1.7.3 Unity.

El següent pas és fer un videojoc i connectar l'exoesquelet de braç. Això es farà amb el software Unity, que permet crear videojocs. Amb aquesta millora es pretén que els pacients que han de fer rehabilitació amb l'exoesquelet ho facin sense avorrir-se, ja que amb el joc es distrauen i fa que sigui més amè. A més, amb aquest tipus d'activitat es poden proposar reptes que motivin al pacient a millorar.

En aquest treball el videojoc que es programarà és similar al pong, on una pilota ha de rebotar en una plataforma sense que aquesta caigui. El moviment de la plataforma serà controlat a través de l'exoesquelet, on el jugador haurà de moure el braç per evitar que la pilota caigui.

#### *1.7.3.1 Creació de l'escenari.*

Primer de tot el que s'ha de fer és crear el projecte a Unity i crear els elements que hi ha a l'escenari del joc. Es crea un cub, que es modela d'acord a les següents mides i se li afegeix les següents restriccions:

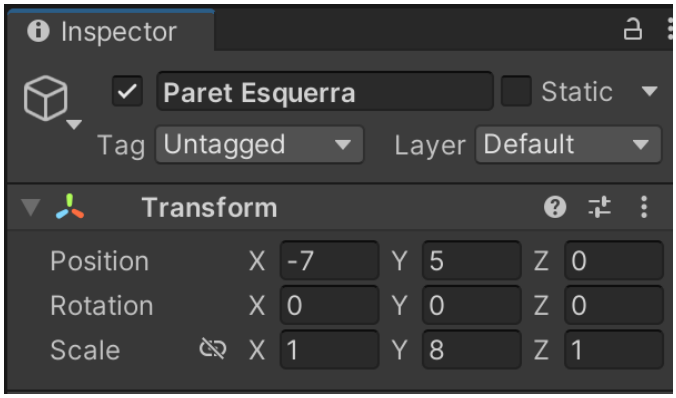



**Figura 46.** Mides i coordenades de la paret esquerra.

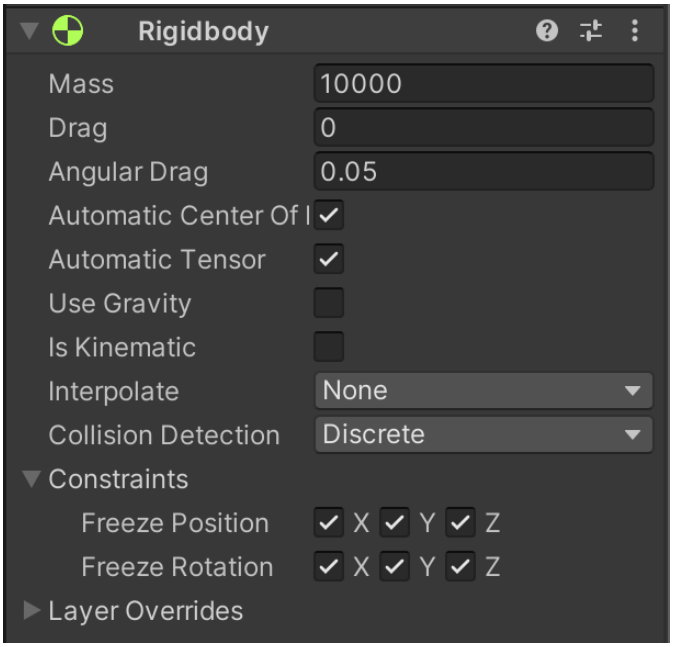



**Figura 47.** Limitacions de la paret esquerra.

Aquest cos representa la paret de l'esquerra, que es declara com un cos rígid. A aquest element se li deshabilita l'opció de fer ús de la gravetat per a que no caigui una vegada el joc estigui executant-se. També se li atribueix una massa molt gran per a que quan la pilota xoqui contra la paret aquesta no es mogui i la pilota reboti. A més, també és limita el moviment del cos fixant la seva posició i orientació. Amb tot això ja s'ha creat la paret de l'esquerra, i per crear la paret de la dreta només s'ha de duplicar el cos anterior i canviar les coordenades de l'eix x.

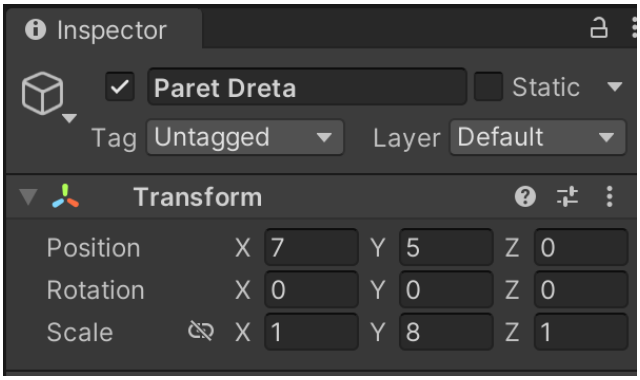



**Figura 48.** Mides i coordenades de la paret dreta.

A més, també es crearà un sostre per a que la pilota no surti per dalt de l'escenari. Les noves mides i coordenades són les següents:

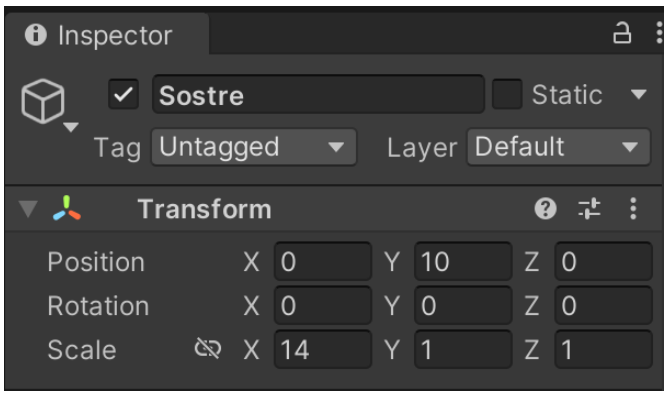



**Figura 49.** Mides i coordenades del sostre.

Ara es crearà la barra que es mourà per evitar que la pilota caigui. Per crear aquest cos es segueixen els passos anteriors, però canviant la posició i les dimensions del cos i fixant el moviment en l'eix y i z i en les tres orientacions, però permetent que es mogui en l'eix x.


**Figura 50.** Mides i coordenades de la barra.

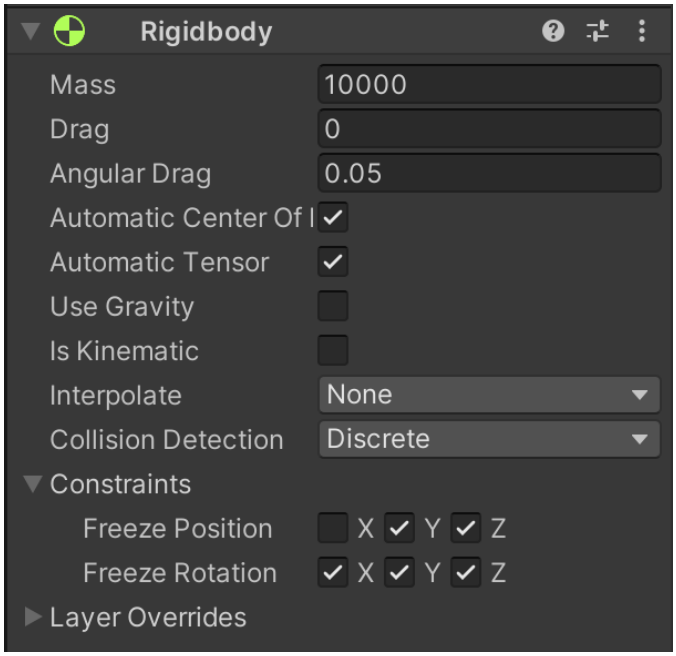



**Figura 51.** Limitacions de la barra.

Finalment, es crea l'últim cos que hi haurà a l'escenari del videojoc, la pilota. Per crear la pilota s'afegeix una esfera, es declara la seva posició i les seves mides. En aquest cas, en canvi, la massa de la pilota serà de 1, i s'activarà la gravetat marcant la seva casella.

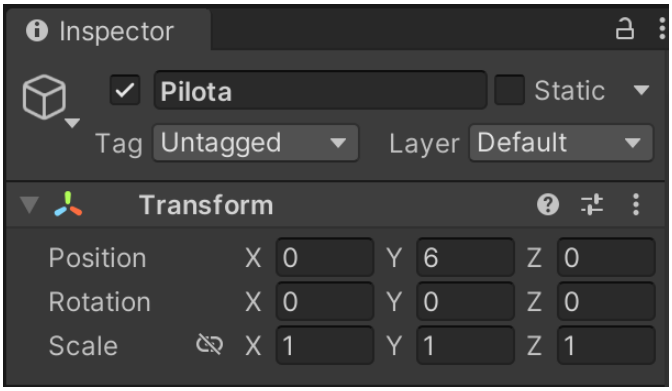



**Figura 52.** Mides i coordenades de la pilota.

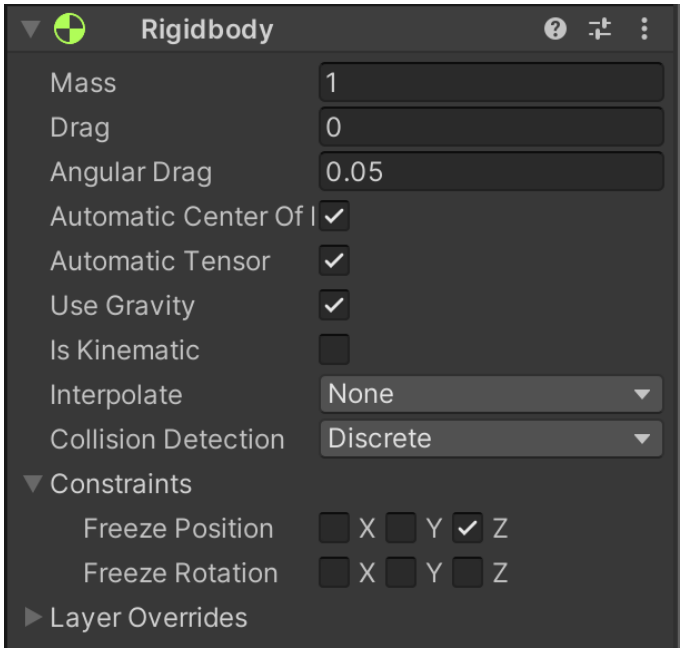



**Figura 53.** Limitacions de la pilota.

Ara s'ha d'afegir les físiques a cada element del joc. Per fer que la pilota reboti s'han de crear uns "Physical Material", que es configuren i s'apliquen a cada cos arrossegant aquest asset fins que estigui damunt de l'element. En aquest asset de "Physical Material" s'ha de configurar la fricció a 0 i el rebot a 1. Amb aquestes configuracions la pilota sempre estarà rebotant.

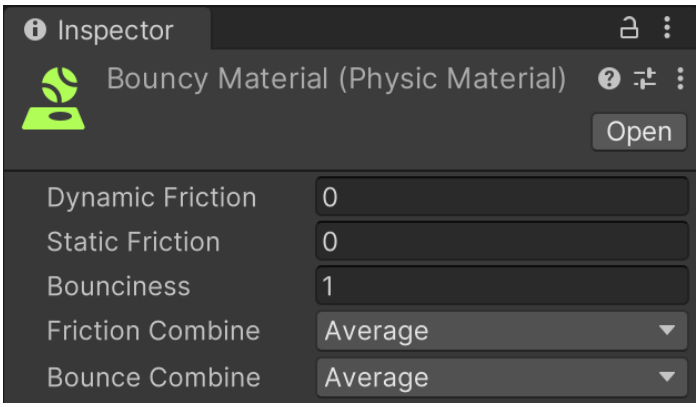



**Figura 54.** Comportament físic de la pilota.

### *1.7.3.2 Programació del joc.*

Amb tot l'escenari del joc creat toca passar a la programació del joc. Per fer això s'afegiran scripts on es farà el programa en llenguatge C# a l'apartat 'Assets', 'Create', 'C# Script'. El primer Script serà el que controla la barra.

En aquest script el que es pretén és indicar les tecles necessàries per controlar la barra i programar el moviment d'aquesta. Per fer això és crearà una variable de velocitat,

on quant menor sigui el valor més lent es mou la barra i més difícil serà el joc. Una vegada fet això, no es realitza cap inicialització en l'apartat *Start()* i en l'apartat *Update()* es redactarà tot el programa.

Aquí s'indica que si es prem la tecla 'A' o la fletxa de l'esquerra (*LeftArrow*) la barra es mourà cap a l'esquerra a partir de la multiplicació de la velocitat establerta anteriorment per l'increment de temps. En cas que es premi la tecla 'D' o la fletxa de la dreta (*RightArrow*) la barrà es mourà cap a la dreta.

Tot el codi de l'script és pot veure amb més detall a l'apartat *2.2.1 Control de la barra* dels Annexes.

El segon script que es programarà és el que estableix el moviment inicial de la pilota. Com que en aquest programa es vol descriure una acció que només ocorrerà a l'inici del programa, tot el codi ha d'anar dins de l'apartat *Start()*. Aquí s'establirà una funció que aleatoritza la velocitat de la pilota en l'eix x i en l'eix y entre un rang de valors, i posteriorment s'assigna aquestes velocitats a la variable *RigidBody* de la pilota.

El codi d'aquest script es troba a l'apartat *2.2.2 Control de la pilota* dels Annexes.

#### *1.7.3.3 Control del joc amb l'exoesquelet de braç.*

Una vegada s'ha creat, s'ha programat i es pot controlar el joc amb el teclat, el següent pas és el de programar el seu control amb l'exoesquelet de braç i jugar amb el braç connectat al joc.

Primer s'escriu amb la IDE d'Arduino un programa per mesurar la posició a la que es troba el servomotor i s'envia aquesta informació per la interfície sèrie per a que es pugui llegir per Unity. En aquest programa es mapeja la posició del servomotor dins dels límits de rotació establerts i s'obté un valor de tipus byte que es troba dins del rang de 0 a 255. Aquest valor s'envia per comunicació sèrie amb la instrucció *Serial.write()* i amb la instrucció *Serial.flush()* s'assegura que la transmissió de dades s'ha completat. Al final del programa es troba la instrucció *delay(20)*. Amb aquesta instrucció s'estableix un temps entre els paquets de dades que s'envien. Amb un valor de 20 ms es pot assegurar que no s'envien massa dades i que Unity es capaç de processar-les.

Aquest programa s'ha de pujar a la placa Arduino, i una vegada connectada la placa, a la cantonada inferior dreta de la IDE es pot veure el port de comunicació que s'utilitza. Aquest port de comunicació és el que s'utilitzarà en el següent programa.

El següent programa que s'escriu llegeix el valor mesurat a través de la comunicació sèrie i utilitza el valor rebut per posicionar la barra del pong. Aquí s'afegeixen les llibreries per habilitar la comunicació sèrie i s'indica el port de comunicació i la velocitat de comunicació. Amb els valors rebuts es mapeja la posició en el sistema de coordenades de Unity en una variable decimal. Amb la funció *catch()* s'obté un missatge d'error en cas que la comunicació sèrie no funcioni.

Finalment, abans d'executar el joc s'ha d'habilitar la comunicació sèrie en Unity en la configuració.

## 1.8 Resultats finals.

En aquest apartat es veuran els resultats finals i la conclusió del treball.

### 1.8.1 Sistemes de control amb Arduino.

Una vegada es coneix com fer els programes bàsics per controlar els diferents sensors, ja es poden fer programes una mica més complexos per realitzar sistemes de control de l'exoesquelet. El primer d'aquest sistemes és un sistema de control de posició on es fa un moviment de punt a punt PtP (point-to-point).

Amb aquest sistema el que es fa és un moviment repetitiu, on es declaren les variables anomenades "counter" i "reps", que indiquen el nombre de vegades que s'ha repetit el moviment i el nombre de vegades que s'ha de repetir, respectivament. Per indicar les posicions que es volen assolir amb l'exoesquelet es fa com amb els programes bàsics, amb la instrucció "myservo.write()". El codi d'aquest programa es troba a l'apartat *2.1.8 Moviment Punt a Punt (PtP Movement)* dels Annexes.

En el següent sistema es grava una trajectòria amb el braç i aquest l'executa. Per fer això s'ha de fer ús de les memòries que hi ha a l'Arduino. En aquest programa hi ha dos funcions, una funció que grava la trajectòria que s'està realitzant, i una funció que repeteix la trajectòria que s'ha gravat. A l'apartat *2.1.9 Trajectòria pregravada* dels Annexes es pot veure el programa que s'ha fet.

El següent sistema que es fa és un control d'admitància, on s'implementen junts el sensor de força i el servomotor. El programa es troba a l'apartat *2.1.10 Control d'admitància* dels Annexes.

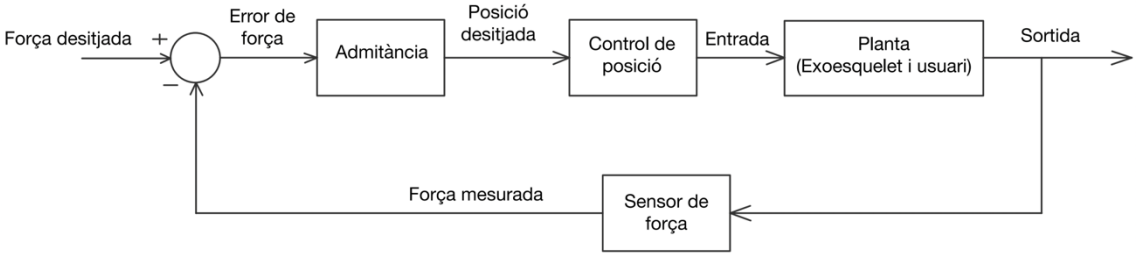



**Figura 55.** Diagrama de control d'un sistema de control d'admitància.

Del codi anterior es pot veure que s'introdueix una variable "forceDesired" on s'indica la força que es desitja imposar a l'usuari per oferir suport. També s'ha d'inicialitzar la posició del braç a un punt dins del rang de moviment del braç i s'introdueix un guany. Aquest guany fixa com de sensible serà el controlador. Quant més gran sigui aquest valor del guany, més sensible serà. Fer el control molt sensible és bo perquè fa que el sistema reaccioni molt més ràpidament, no obstant, també té la seva part negativa, ja que un guany gran implica que el sistema pugui assolir ràpidament un comportament inestable i que hi hagi més soroll.

Dins del bucle s'introdueix un retard d'un valor petit. Aquest retard afecta a la freqüència de control, això vol dir que indica cada quant s'aplica el control en el sistema. Com més petit és aquest valor més vegades entra el control en joc.

En aquest control s'aplica la següent expressió, on és varia la posició desitjada d'acord al guany, a la força aplicada, i a la força desitjada:

$$posDesired = posDesired - gain \cdot (forceIs - forceDesired) \quad (12)$$

El següent control és un on s'implementen parets virtuals, és a dir, límits per a que la posició de l'exoesquelet de braç no surti del rang de moviment que té disponible. En aquest control s'aplica el mateix codi que en el control d'admitància, però s'afegeix unes línies de condició on, si la posició desitjada que s'obté després de fer el càlcul en el codi és superior a l'angle màxim que es pot aplicar al motor del servomotor, es declara la posició desitjada com la màxima. Si la posició desitjada, en canvi, fos inferior que la de l'angle mínim del motor, en aquest cas s'aplicaria que la posició desitjada fos la de l'angle mínim. El programa d'aquest control es troba a l'apartat *2.1.11 Murs virtuals al control d'admitància* dels Annexes.

### 1.8.2 Sistemes de control amb Simulink.

Una vegada s'han fet els diversos programes per llegir els diferents sensors i per posicionar el servomotor a la posició desitjada, el següent pas és el de combinar els diferents elements per poder realitzar sistemes de control per al exoesquelet de braç.

El primer programa que es pretén fer és un sistema de control a partir del que llegeix el sensor de força. La primera idea que es va proposar a l'hora de fer aquest sistema de control era fer ús d'un llaç tancat on s'utilitzava com una entrada la posició desitjada. A continuació és pot veure el diagrama de blocs que es proposava:

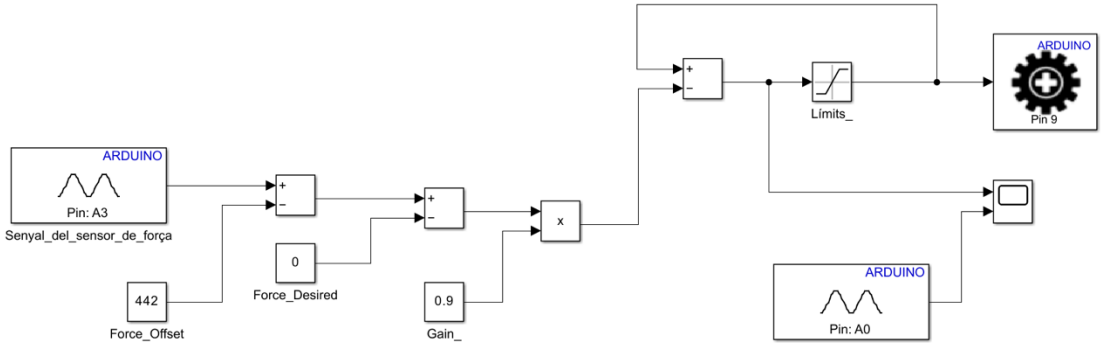



**Figura 56.** Diagrama de blocs proposat.

No obstant, a l'hora de fer llaços tancats en Simulink amb Arduino apareix un error que indica que no és possible fer llaços tancats:

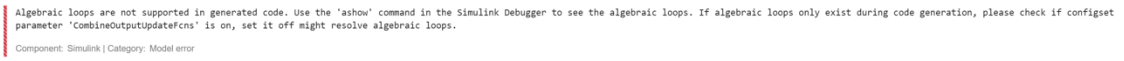



**Figura 57.** Missatge d'error sobre els llaços tancats.

Com que no és possible fer llaços tancats, s'ha decidit fer el següent diagrama de blocs:

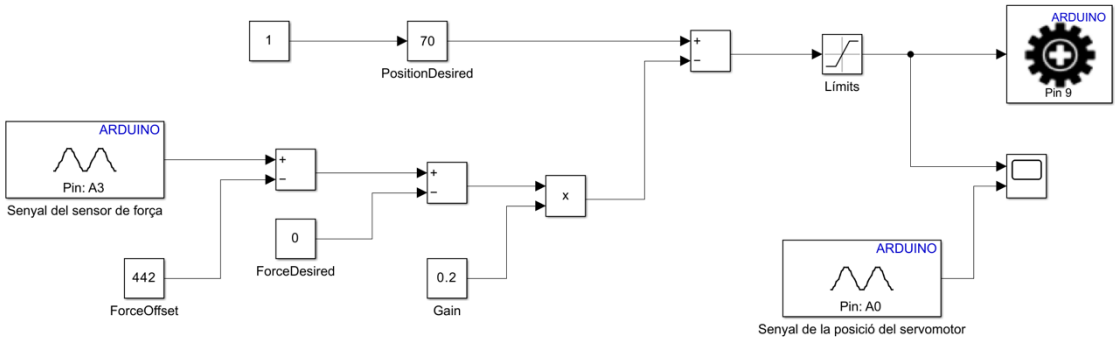



**Figura 58.** Diagrama de blocs del sistema de control a partir del sensor de força.

En aquest cas en lloc de fer un llaç tancat s'estableix quina és la posició desitjada a la que es vol mantenir el servomotor, amb un guany de 1 i un bloc de guany variable entre els límits establerts (12º i 110º). A aquesta posició desitjada se li resta la part de la força. En aquesta part es llegeix la força mesurada pel sensor i se li resta l'offset de la força. A aquest resultat se li resta la força desitjada, que és la força que l'exoesquelet imposarà a l'usuari. Aquest resultat es multiplicaria pel guany, que es pot anar variant.

A la sortida d'aquest control és connecta un bloc saturador, que imposa els límits de la posició a la que es col·locarà el servomotor entre 12º i 110º. Aquest bloc és un bloc de control per evitar que la posició del servomotor surti del seu rang de moviment permès i evitar que l'usuari prengui mal.

A continuació es pot veure com afecta a la posició tenir un valor de guany no molt gran de 0,2:

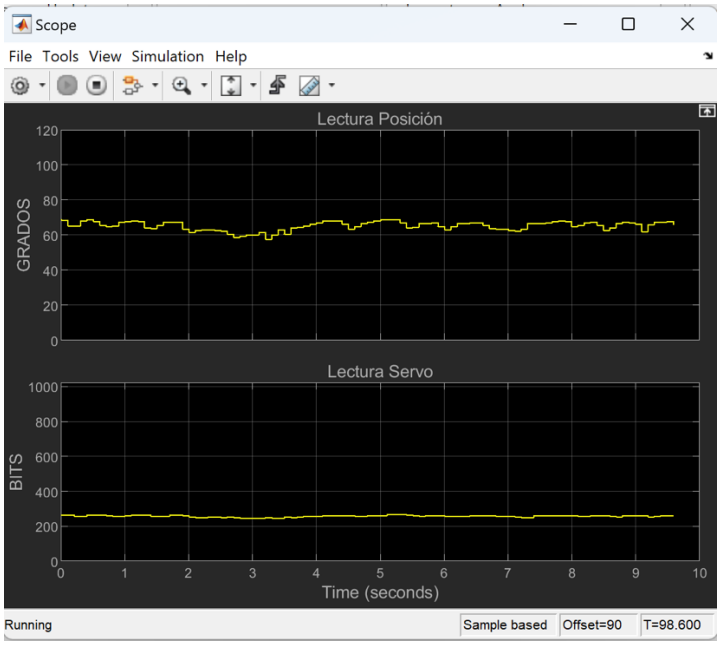



**Figura 59.** Evolució de la posició del servomotor amb aquest sistema de control amb un guany de 0,2.

Es pot observar com hi ha petites variacions en la posició del servomotor, però es manté bastant estable al voltant del valor de la posició desitjat.

Ara, si s'augmenta el valor del bloc de guany a un valor alt, com el de 0,9, es pot veure com el següent senyal:

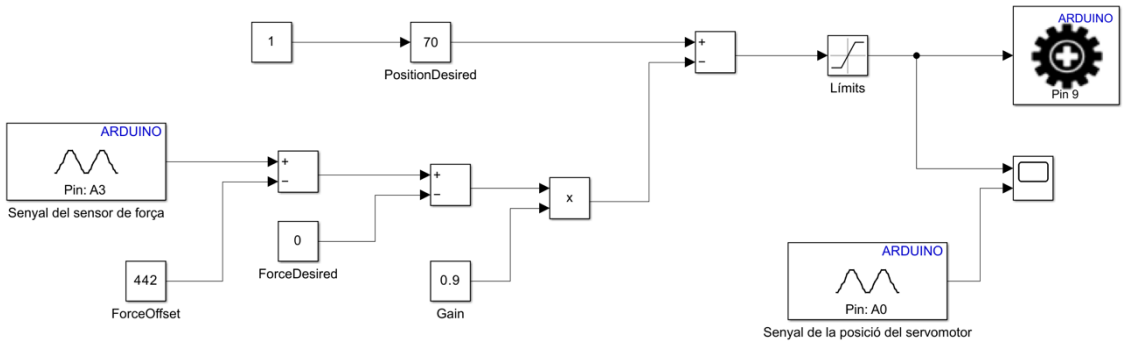



**Figura 60.** Diagrama de blocs del sistema de control a partir del sensor de força amb un guany de 0,9.

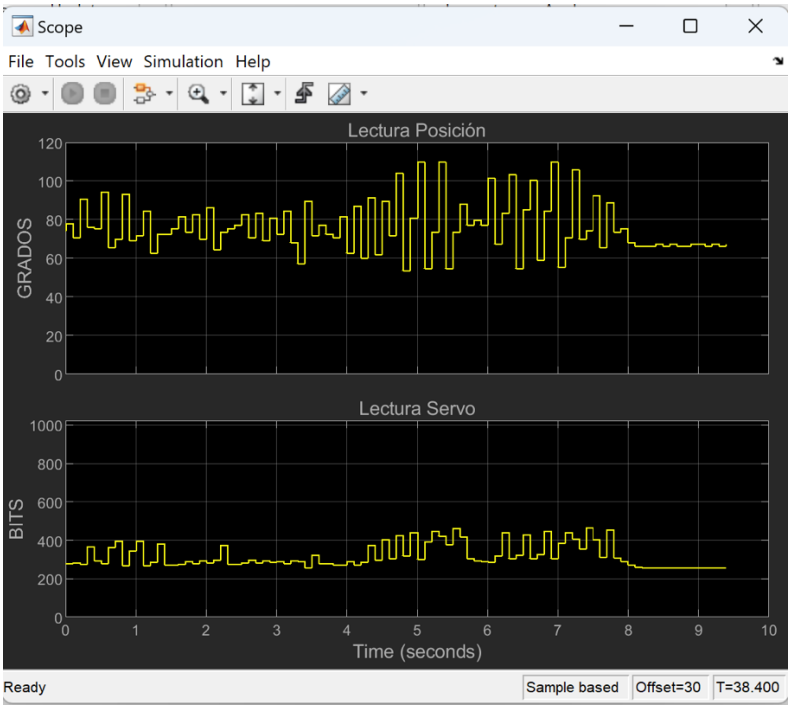



**Figura 61.** Evolució de la posició del servomotor amb aquest sistema de control amb un guany de 0,9.

Amb aquest valor del guany es pot veure com el senyal de la posició varia molt més i es pot observar més soroll. Si el guany fos massa gran el sistema podria tornar-se inestable.

El següent controlador que es construirà implementa el sensor EMG. Igual que en el controlador anterior basat en el sensor de força, la primera idea era la de fer un llaç tancat, on el control a partir del sensor de força només entraria en joc quan es mesurés un valor del sensor EMG per sobre del llindar de 577,5. Mentre no es superés aquest valor llindar el servomotor es mantindria en la mateixa posició (es retroalimentaria). Com que no s'aplicaria cap control la variació de l'angle del servomotor no suposaria cap problema ja que no hi hauria cap força oposada al moviment.

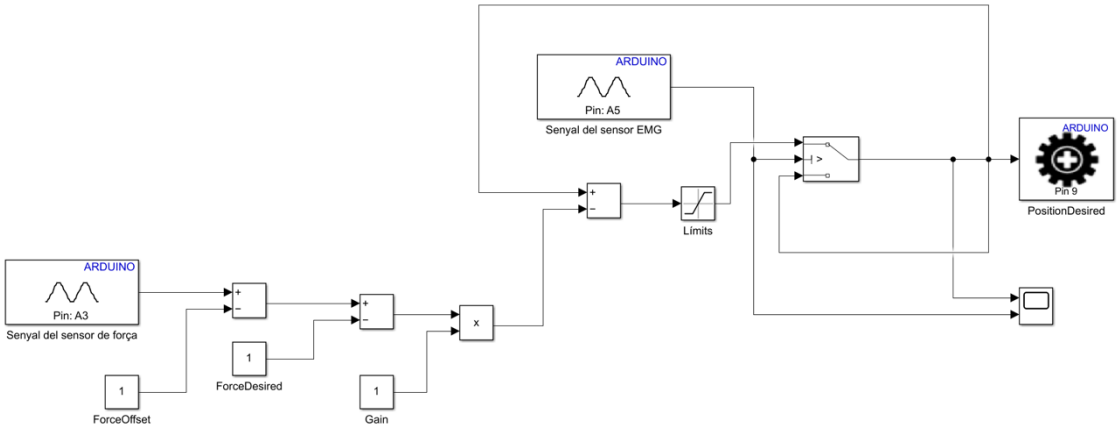



**Figura 62.** Diagrama de blocs proposat.

No obstant, com s'ha vist abans, no es possible fer llaços de control tancats amb la connexió de l'Arduino, per tant, es va decidir canviar el sistema.

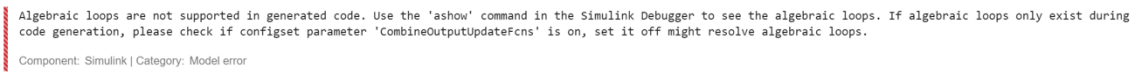



**Figura 63.** Missatge d'error sobre els llaços tancats.

Aquest nou sistema el que fa és que si es supera el valor llindar de 577,5 mesurat pel sensor EMG el braç es posiciona a una posició desitjada establerta pel bloc de guany variable (entre 12º i 110º). En aquest exemple s'ha utilitzat una posició de 70º. Si no es supera el valor llindar, es llegeix la posició actual del servomotor i es col·loca el servomotor en aquesta posició llegida. Amb això s'aconsegueix crear una mena de retroalimentació, tal i com és volia fer amb el diagrama de blocs proposat. Com que el sistema es retroalimentarà, es podrà variar la posició de l'exoesquelet de braç ja que no hi haurà cap força oposada al moviment.

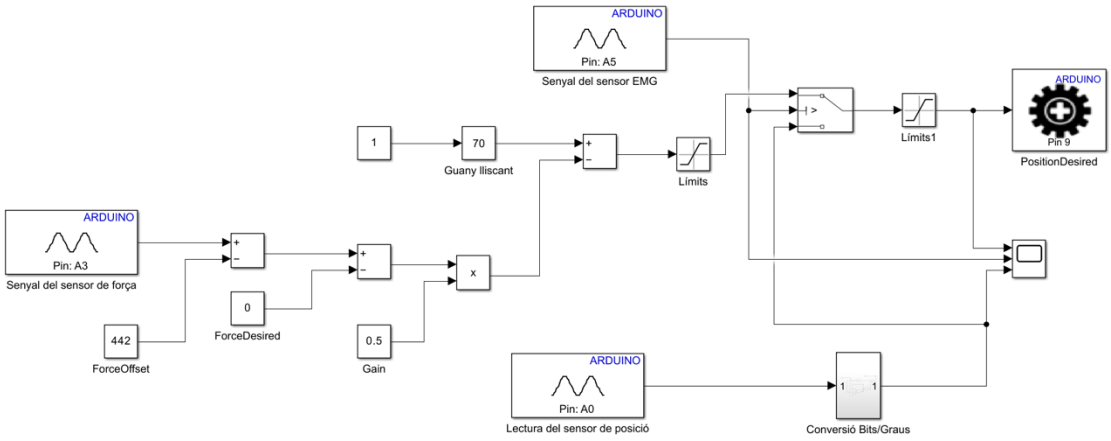



**Figura 64.** Diagrama de blocs del sistema de control a partir dels sensor de força, EMG, i de posició.

Del sistema anterior és pot veure la següent evolució de la posició del servomotor en funció dels valors llegits pel sensor EMG. Es pot veure com la posició varia sense problemes fins als, aproximadament, 6,5 segons, i a partir d'aquí es detecta activitat elèctrica amb el sensor EMG. Això fa que el servomotor vulgui anar a la posició desitjada de 70º. Als més o menys 8,5 segons l'activitat elèctrica del sensor EMG torna a baixar per sota del valor llindar, i per tant l'exoesquelet de braç pot tornar a moure's lliurement.

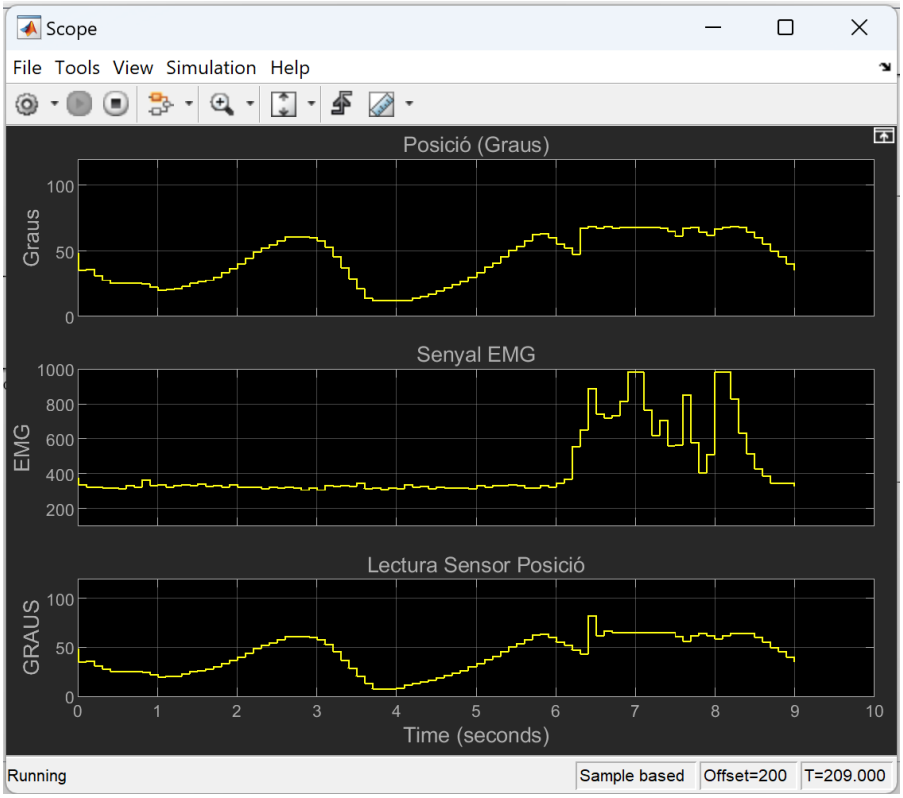



**Figura 65.** Evolució de la posició en funció de l'activitat elèctrica mesurada pel sensor EMG.

### 1.8.3 Videojoc pong.

El resultat final de l'escenari del videojoc pong que s'ha dissenyat és el que es pot veure a la següent imatge:

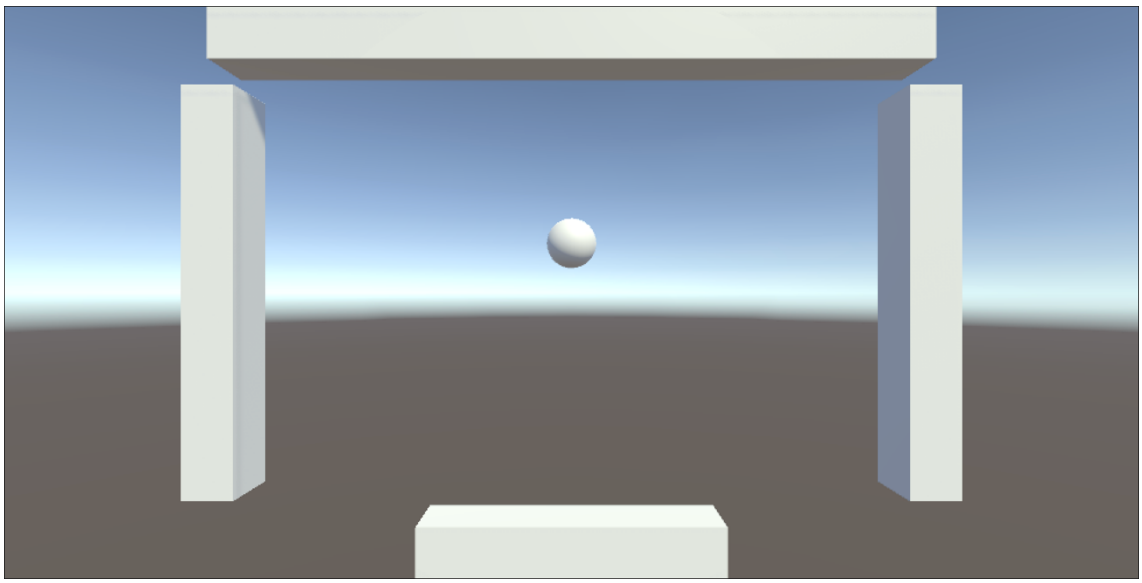


**Figura 66.** Escenari creat pel videojoc pong.

Es poden veure tots els elements que s'han creat. Es veuen les parets i el sostre, que delimiten l'escenari de joc, i la barra mòbil. També es pot observar la pilota.

A més, en els següents enllaços es poden veure els vídeos de demostració del funcionament del videojoc. El primer enllaç porta a un vídeo on es veu el joc controlat amb el teclat, i el segon enllaç porta a un vídeo on es veu el joc controlat amb l'exoesquelet de braç EduExo:

- Pong controlat amb el teclat: https://youtu.be/F9y59cJKpt0
- Pong controlat amb l'EduExo: https://youtu.be/d4NDqoqkJR8

### 1.8.4 Caixa de protecció del sistema i connexions realitzades.

Una altra millora que s'ha d'aplicar al sistema de l'exoesquelet és trobar una forma de protegir els seus elements i de fer tot el sistema més robust. Per aconseguir aquest objectiu és pretén dissenyar una caixa de protecció del sistema, on guardar tots els elements (hardware i circuiteria) de la placa de control i que només surtin els cables que siguin necessaris.

Primer de tot s'ha de mesurar les mides que tenen tant la placa Arduino com la protoboard, sense cap cable connectat. Les mides aproximades són les següents:

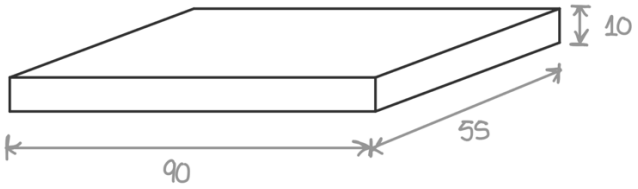



**Figura 67.** Mides de la protoboard en mil·límetres (sense cables ni components).

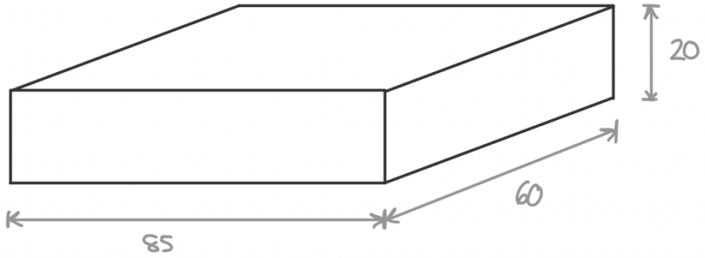



**Figura 68.** Mides de la placa Arduino en mil·límetres (sense cables).

Amb aquestes mides en ment es dissenya la caixa, on s'ha de deixar més marge d'espai per que hi càpiguen tots els cables, la resistència i l'amplificador. El disseny de la caixa que es proposa és el següent:

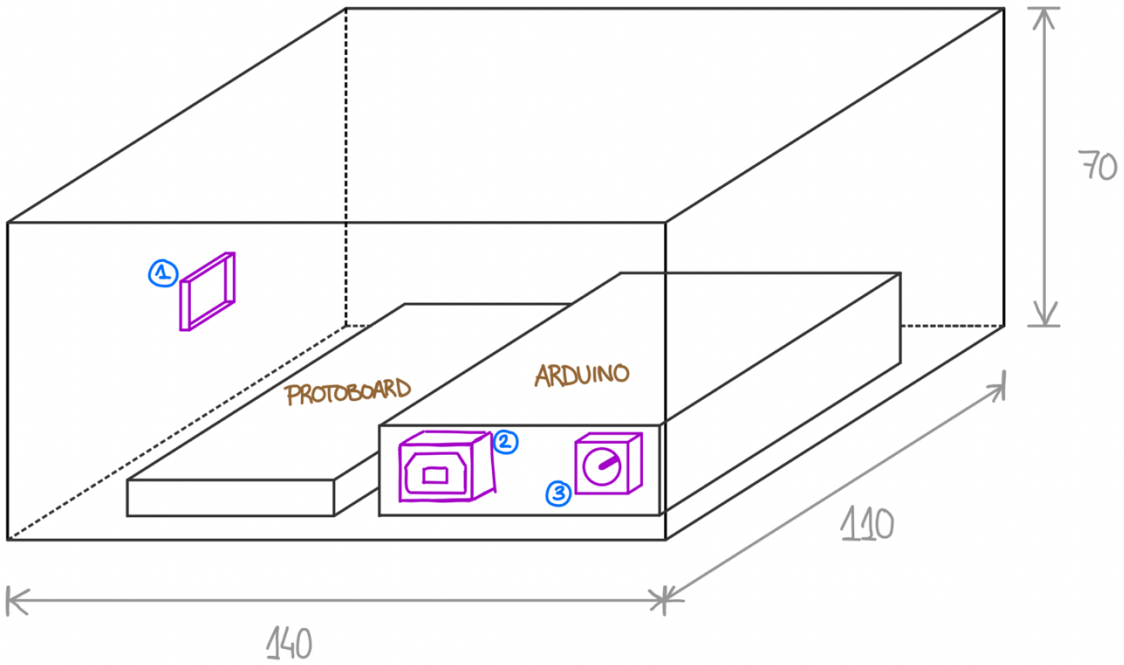



**Figura 69.** Esbós de la caixa protectora amb la protoboard i la placa Arduino, els forats del cables numerat, i les mides de la caixa en mil·límetres.

En l'esbós anterior es pot veure que la caixa tindria unes mides aproximades de 140 mm de llarg, 110 mm d'ample, i 70 mm d'alçada. Dins de la caixa és troba la placa Arduino, amb la protoboard on es troben les connexions de l'amplificador del pont de Wheatstone. Tot això es troba fixat a l'interior, sense que es pugui moure, impedint que es pugui desmuntar, o que rebi cops. De la caixa sortirà una mànega de cables, que seran els del servomotor, els del sensor de força i els del sensor EMG. Aquesta mànega de cables sortiran pel forat de la caixa marcat amb el número 1.

També sortiran dos cables més de la caixa, un d'ells el cable USB per connectar a l'ordinador en cas de voler reprogramar la placa o d'alimentar el sistema a través de l'ordinador. Aquest cable sortirà pel forat marcat amb el número 2, on es veu que té la forma de la connexió de l'USB.

L'altre cable que sortirà és per connectar una bateria, que s'utilitzarà per alimentar el sistema en cas de no voler fer servir un ordinador. Aquest cable surt pel forat número 3, on es veu la connexió del cable d'alimentació.

Les connexions del sistema seran les que es poden veure a la següent figura, on hi ha tota la circuiteria que connecta la placa Arduino amb la protoboard i l'amplificador, i amb els sensors i el servomotor:

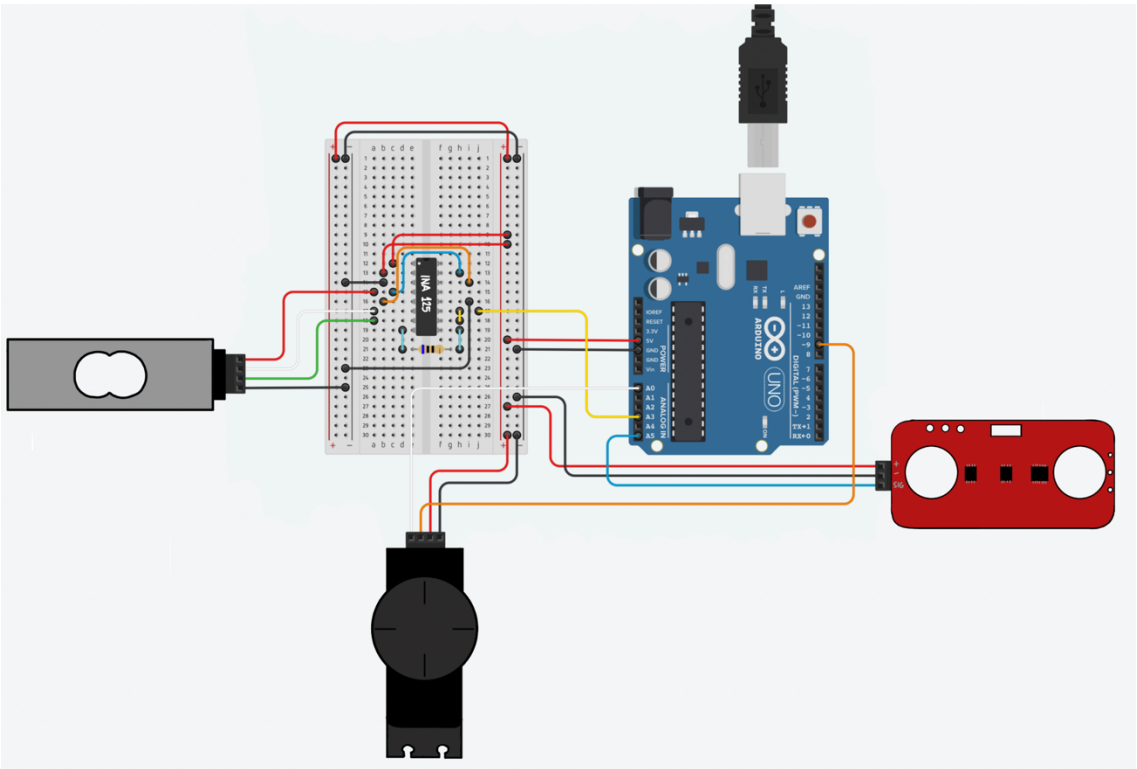



**Figura 70.** Connexions de tots els sensors i actuadors amb la placa Arduino

I a la següent taula es pot veure quins pins de la placa Arduino s'han utilitzat i que s'ha connectat en ells:

| | |
|---|---|
| **5 V** | Alimentació del positiu de la protoboard |
| **GND** | Terra de la protoboard |
| **A0** | Lectura de la posició del servomotor |
| **A3** | Lectura del sensor de força |
| **A5** | Lectura del sensor EMG |
| **~9** | Control de la posició del servomotor |

**Taula 5.** Pins utilitzats de la placa Arduino i funció que fan.

A continuació és poden veure unes imatges de com quedarien les connexions amb les que s'han treballat durant el projecte. La primera imatge mostra tot el sistema, amb l'exoesquelet de braç, el servomotor, el sensor de força, el sensor EMG, la protoboard amb l'amplificador i la placa Arduino. Les altres dos fotos mostren de més a prop les connexions de la protoboard amb l'amplificador INA 125.

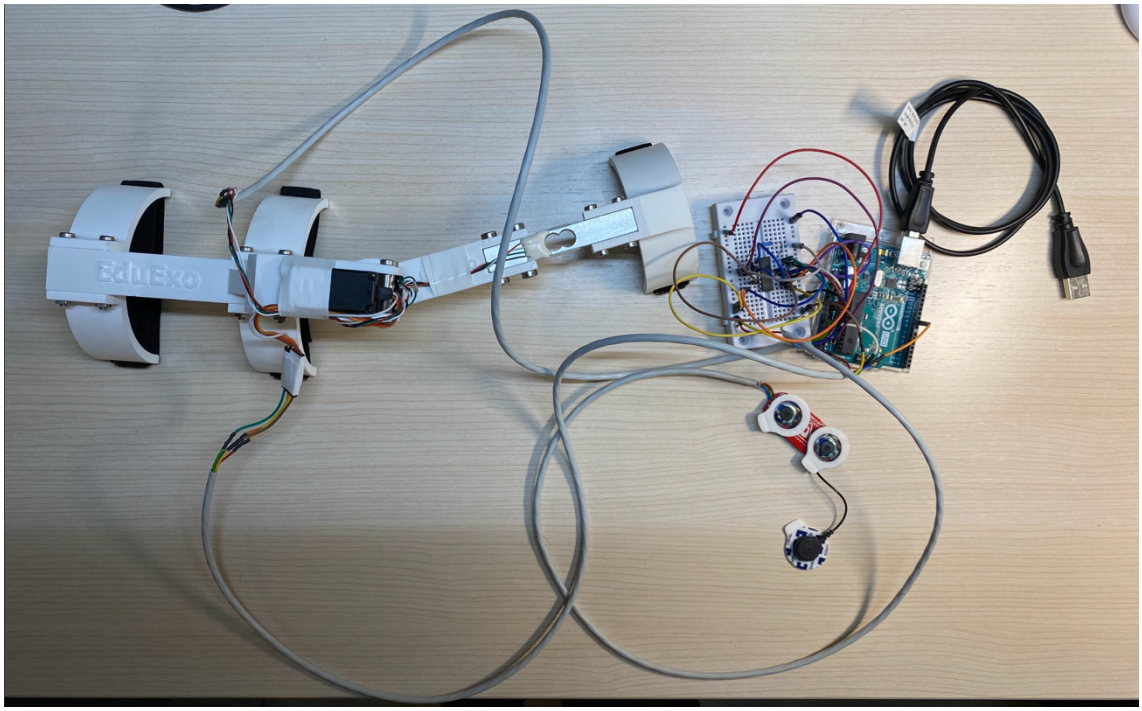



**Figura 71.** Connexions del sistema de l'exoesquelet de braç.

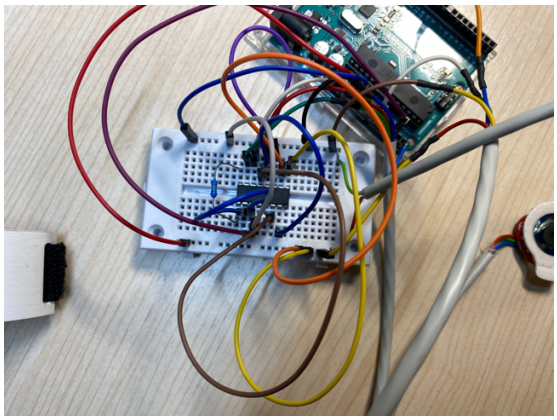


**Figura 72.** Connexions de la protoboard i l'amplificador vist des de dalt.

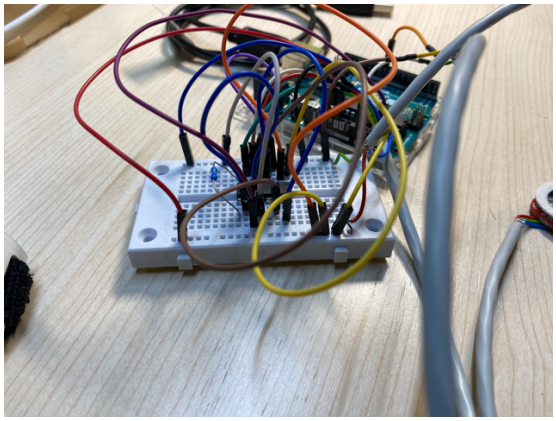


**Figura 73.** Connexions de la protoboard i l'amplificador vist des del costat.

### 1.8.5 Conclusions.

Amb aquest treball es pot veure com s'apliquen els exoesquelets per millorar la rehabilitació dels pacients, recopilant dades i ajudant al fisioterapeuta a facilitar la seva feina. A més, es mostra com, amb un videojoc, es pot motivar als pacients a millorar i fer entretinguda la rehabilitació.

També s'han vist diversos mètodes de programació de l'exoesquelet de braç EduExo que es poden aplicar en la docència (en aquest cas en l'assignatura de Robòtica Mèdica del Grau d'Enginyeria Biomèdica) i diferents sistemes de control amb els que es poden treballar.

A més, s'ha proposat un esbós d'una caixa de protecció pel sistema per evitar desperfectes. Aquesta caixa es podria fabricar, ja sigui amb impressió 3D o amb altres mètodes, i utilitzar-la en les pràctiques de laboratori per evitar que els alumnes, a causa de la seva curiositat, desconnectin o trenquin el sistema, ja que és bastant sensible.

Per tant, amb tot això es pot concloure que s'han complert els objectius establerts en aquest treball.

No obstant, cal dir que amb més temps i més recursos el sistema es podria millorar, com per exemple implementant més sensors i actuadors, millorant l'aspecte estètic del videojoc, o fins i tot fent-ne un de nou, o fer una caixa de protecció millorada o que sigui portàtil. Per tant, es podria dir que aquest sistema encara és podria evolucionar molt més i es podrien fer futurs projectes per tal de millorar aquests aspectes.

## 1.9 Planificació.

No es d'aplicació en aquest treball.

## 1.10 Ordre de prioritat entre documents.

En cas de discrepàncies, l'ordre de prioritat dels documents d'aquest treball és el següent:

1 Memòria

2 Annexes

# 2 Annexes.

En aquest apartat d'Annexes es troben els diversos programes que s'han fet amb la IDE d'Arduino i els codis del videojoc.

## 2.1 Arduino.

En aquest apartat es troben tots els codis que s'han programat per entendre el funcionament de l'exoesquelet de braç.

### 2.1.1 Lectura de l'angle del servomotor en bits.

Amb aquest codi és possible llegir l'angle en el que es troba l'exoesquelet de braç en bits a partir del sensor de posició que porta incorporat el servomotor:

```
int servoAnalogInPin = A0; /*Pin on es llegeix la posició del servomotor en bits*/
int posIs; /*Variable on es guarda la posició del servomotor en bits*/

void setup()
{
 Serial.begin(9600);
 delay(1000);
}

void loop()
{
 posIs = analogRead(servoAnalogInPin); /*Procés per obtenir i guardar la posició del motor*/
 Serial.print("La posicio es: "); /*Línies de codi per mostrar per pantalla la posició del motor*/
 Serial.println(posIs);
 delay(10);
}
```

### 2.1.2 Lectura de l'angle del servomotor en graus sexagesimals.

Amb aquest codi és possible llegir l'angle en el que es troba l'exoesquelet de braç en bits i en graus sexagesimals a partir del sensor de posició que porta incorporat el servomotor:

```
int servoAnalogInPin = A0; /*Pin on es llegeix la posició del servomotor en bits*/
int posIs; /*Variable on es guarda la posició del servomotor en bits*/
float posIsDeg; /*Variable on es guarda la posició del servomotor en graus sexagesimals*/

void setup()
```

```
{
 Serial.begin(9600);
 delay(1000);
}

void loop()
{
 posIs = analogRead(servoAnalogInPin); /*Procés per obtenir
i guardar la posició del motor*/
 posIsDeg = ((90.0 - 0.0)/(316.0 - 130.0)) * (posIs - 130.0);
/*Càlcul per obtenir la posició del motor en graus
sexagesimals a partir dels valors mesurats en el calibratge
del motor*/
 Serial.print("La posicio (en graus) es: "); /*Línies de
codi per mostrar per pantalla la posició del motor*/
 Serial.println(posIsDeg);
 delay(10);
}
```

### 2.1.3 Posicionament del servomotor.

Amb aquest codi el que es fa és posicionar l'exoesquelet de braç a la posició que es desitja indicant aquesta posició en graus:

```
#include <Servo.h> /*Afegim la llibreria del servomotor*/

Servo myservo; /*Instanciem l'objecte myservo de la classe
Servo*/

int positionDesired = 30; /*Creem una variable on s'indica
la posició desitjada (en graus) a la que es vol col·locar el
servomotor*/

void setup()
{
 Serial.begin(9600);
 myservo.attach(9); /*Indiquem quin és el pin en el que es
troba connectat el servomotor, en aquest cas el pin 9*/
}

void loop()
{
 myservo.write(positionDesired); /*Indiquem la posició en
graus en la que s'ha de col·locar el braç robòtic*/
}
```

### 2.1.4 Lectura del sensor de força.

Amb aquest codi es pretén llegir el valor de la força mesurat a través del sensor de força:

```
int forceAnalogInPin = A3; /*Pin on es llegeix la força
mesurada pel sensor de força*/
int forceIs; /*Variable on es guarda la força mesurada pel
sensor*/

void setup()
{
 Serial.begin(9600);
 delay(1000);
}

void loop()
{
 forceIs = analogRead(forceAnalogInPin); /*Procés per
obtenir i guardar el valor de la força mesurat pel sensor*/
 Serial.print("La força es: "); /*Línies de codi per mostrar
per pantalla la força mesurada pel sensor*/
 Serial.println(forceIs);
 delay(100);
}
```

### 2.1.5 Calibratge del sensor de força.

En aquest codi l'objectiu és el de calibrar el sensor de força a partir d'un valor d'offset de la força mesurat amb el codi anterior:

```
int forceAnalogInPin = A3; /*Pin on es llegeix la força
mesurada pel sensor de força*/
int forceIs; /*Variable on es guarda la força mesurada pel
sensor*/
const int forceOffset = 442; /*Constant de valor 442 que
correspon al valor de l'offset de la força*/

void setup()
{
 Serial.begin(9600);
 delay(1000);
}

void loop()
{
 forceIs = analogRead(forceAnalogInPin); /*Procés per
obtenir i guardar el valor de la força mesurat pel sensor*/
 forceIs -= forceOffset; /*Càlcul per calibrar el sensor de
força i conèixer el sentit d'aplicació de la força*/
 Serial.print("La força es: "); /*Línies de codi per mostrar
per pantalla la força mesurada pel sensor*/
 Serial.println(forceIs);
 delay(10);
}
```

### 2.1.6 Lectura del sensor EMG.

A partir d'aquest codi el que es fa és llegir l'activitat elèctrica que s'ha mesurat amb el sensor electromiogràfic (EMG):

```
int EMG_AnalogInPin = A5; /*Pin on es llegeix l'activitat elèctrica mesurada pel sensor EMG*/
int EMG_Signal; /*Variable on es guarda l'activitat elèctrica mesurada pel sensor*/

void setup()
{
 Serial.begin(9600);
}

void loop()
{
 EMG_Signal = analogRead(EMG_AnalogInPin); /*Procés per obtenir i guardar el valor del senyal elèctric mesurat pel sensor*/
 Serial.print("El senyal de l'activitat elèctrica mesurada pel sensor EMG és: "); /*Línies de codi per mostrar per pantalla la força mesurada pel sensor*/
 Serial.println(EMG_Signal);
 delay(20);
}
```

### 2.1.7 Controlador de l'exoesquelet amb el sensor EMG.

A partir d'aquest codi es dissenya un sistema on s'estableix un valor llindar del senyal mesurat pel sensor EMG. Si es supera aquest valor llindar, el servomotor de l'exoesquelet de braç posicionarà el sistema en una posició establerta, però si no es supera aquest valor llindar, el servomotor es desacoblarà del seu pin de connexió:

```
#include <Servo.h> /*Afegim la llibreria del servomotor*/

Servo myservo; /*Instanciem l'objecte myservo de la classe Servo*/

int EMG_AnalogInPin = A5; /*Pin on es llegeix l'activitat elèctrica mesurada pel sensor EMG*/
int EMG_Signal; /*Variable on es guarda l'activitat elèctrica mesurada pel sensor*/
int positionDesired = 75; /*Creem una variable on s'indica la posició desitjada (en graus) a la que es vol col·locar el servomotor*/

void setup()
{
 myservo.attach(9); /*Indiquem quin és el pin en el que es troba connectat el servomotor*/
}
```

```
void loop()
{
 EMG_Signal = analogRead(EMG_AnalogInPin); /*Procés per
obtenir i guardar el valor del senyal elèctric mesurat pel
sensor*/
 if(EMG_Signal < 578) /*Si el senyal d'activitat elèctrica
mesurat és menor que el valor llindar*/
 {
   if(myservo.attached())    myservo.detach();    /*Llavors
desacoblem el servomotor del pin*/
 }
 else /*Si el senyal de l'activitat elèctrica mesurat és
major que el valor llindar*/
 {
   if(!myservo.attached())   myservo.attach(9);   /*Llavors
s'acobla el servomotor al pin en el que es troba connectat
en cas que s'hagués desacoblat anteriorment*/
   myservo.write(positionDesired);    /*I    s'indica    al
servomotor en quina posició s'ha de col·locar*/
 }
 delay(10);
}
```

### 2.1.8 Moviment Punt a Punt (PtP Movement).

Amb aquest codi el que es pretén fer és un moviment de punt a punt (PtP), on s'indica a l'exoesquelet de braç a quines posicions ha d'arribar i quantes vegades ha de repetir el moviment:

```
#include <Servo.h> /*Afegim la llibreria del servomotor*/

Servo myservo; /*Instanciem l'objecte myservo de la classe
Servo*/
int positionDesired; /*Creem una variable on s'indica la
posició desitjada (en graus) a la que es vol col·locar el
servomotor*/
int counter = 0; /*Creem la variable counter i s'inicialitza
a 0*/
int reps = 5; /*Creem la variable reps i s'inicialitza a 5*/

void setup()
{
 Serial.begin(9600);
 myservo.attach(9); /*Indiquem quin és el pin en el que es
troba connectat el servomotor*/
}

void loop()
{
```

```
 positionDesired = 5; /*Aquí s'indica la primera posició a
la que es vol col·locar l'exoesquelet*/
 myservo.write(positionDesired); /*Col·loquem el braç a la
posició desitjada*/
 delay(1250); /* Deixem un temps de delay per a que el braç
arribi a la posició */

 positionDesired = 95; /*Aquí s'indica la segona posició a
la que es vol col·locar l'exoesquelet*/
 myservo.write(positionDesired); /* Col·loquem el braç a la
posició desitjada */
 delay(1250); /*Deixem un temps de delay per a que el braç
arribi a la posició*/

 counter++; /*Augmentem la variable counter*/

 if(counter == reps) /*En cas que les variables counter y
reps siguin iguals, es finalitza l'execució del codi*/
 {
   exit(0);
 }
}
```

### 2.1.9 Trajectòria pregravada.

En aquest codi es programa una forma de gravar trajectòries, guardant en la memòria la trajectòria que s'ha fet i repetint-la per l'exoesquelet:

```
#include <Servo.h> /*Afegim la llibreria del servomotor*/
#include <EEPROM.h> /*Afegim la llibreria de la memòria
EEPROM*/

#define SAMPLE_DELAY 25 /*Definim una constant de valor 25
que indica el retardament que hi ha entre la presa de
mostres*/
#define SAMPLES 200 /*Definim una constant de valor 200, que
indica el número de mostres que s'agafaran*/

Servo myservo; /*Instanciem l'objecte myservo de la classe
Servo*/

int servoPin = 9; /*Pin per on s'indicarà la posició del
braç*/
int servoAnalogInPin = A0; /*Pin on es llegeix la posició
del servomotor en bits*/

void setup()
{
 Serial.begin(9600);
}
```

```
void loop()
{
 Serial.println("La trajectoria es gravarà en 3");
/*Missatge que indica quan es començarà a gravar la
trajectòria*/
 delay(1000);
 Serial.println("2");
 delay(1000);
 Serial.println("1");
 delay(1000);

 recordTrajectory(); /*Crida a la funció per guardar la
trajectòria*/

 Serial.println("La trajectoria es repetirà en 3");
/*Missatge que indica quan es començarà a reproduir la
trajectòria*/
 delay(1000);
 Serial.println("2");
 delay(1000);
 Serial.println("1");
 delay(1000);

 replayTrajectory(); /*Crida a la funció per reproduir la
trajectòria*/
}

void recordTrajectory() /*Funció per guardar la
trajectòria*/
{
 Serial.println("Gravant"); /*Indiquem que s'està gravant la
trajectòria*/
 for(int addr = 0; addr < SAMPLES; addr++) /*Mentre no
s'hagin pres totes les mostres es continuarà gravant*/
 {
   int posIs = analogRead(servoAnalogInPin); /*Es llegeix la
posició del servomotor*/
   byte posIsServo = map(posIs, 114, 328, 0, 100); /*Es
mapeja la posició del servo*/
   EEPROM.write(addr, posIsServo); /*Guardem la posició del
servomotor en la memòria*/
   delay(SAMPLE_DELAY); /*Es deixa un temps de delay entre
mostres*/
  }
 Serial.println("Gravació finalitzada"); /*S'indica que la
gravació ha finalitzat*/
}

void replayTrajectory() /*Funció per reproduir la
trajectòria*/
{
```

```
 myservo.attach(servoPin); /*Acoblem el servomotor al pin*/
 Serial.println("Repetint"); /*Indiquem que s'està repetint
la trajectòria*/
 for(int addr = 0; addr < SAMPLES; addr++) /* Mentre no
s'hagin repetit totes les mostres es continuarà repetint */
 {
   byte positionDesired = EEPROM.read(addr); /*Es guarda la
posició desitjada a una variable d'acord al que s'ha gravat*/
   myservo.write(positionDesired); /*Es col·loca el braç a
la posició desitjada*/
   delay(SAMPLE_DELAY); /*Es deixa un temps de delay entre
posicionament del servomotor*/
 }
 Serial.println("Repetició finalitzada"); /*S'indica que la
repetició ha finalitzat*/
 myservo.detach(); /*Es desacobla el servomotor del pin*/
}
```

### 2.1.10 Control d'admitància.

A partir d'aquest codi es programa un control d'admitància que permet ajudar a l'usuari a realitzar moviments amb l'exoesquelet de braç d'acord a la força que es mesura amb el sensor de força:

```
#include <Servo.h> /*Afegim la llibreria del servomotor*/

Servo myservo; /*Instanciem l'objecte myservo de la classe
Servo*/
int forceAnalogInPin = A3; /*Pin on es llegeix la força
mesurada pel sensor de força*/
const int forceOffset = 442; /*Constant que indica el valor
de la força Offset mesurat durant el calibratge del sensor*/
int forceIs; /*Variable on es guarda la força mesurada pel
sensor*/
int forceDesired = 0; /*Creem una variable on s'indica la
força desitjada que es vol aplicar*/
int positionDesired = 60; /*Creem una variable on s'indica
la posició desitjada (en graus) a la que es vol col·locar el
servomotor*/
float gain = 0.2; /*Valor del guany de control*/

void setup()
{
 myservo.attach(9); /*Indiquem quin és el pin en el que es
troba connectat el servomotor*/
 delay(1000);
}

void loop()
{
```

```
 forceIs = analogRead(forceAnalogInPin) - forceOffset;
/*Valor de la força mesurada*/
 delay(10);
 positionDesired -= gain*(forceIs - forceDesired); /*Posició
desitjada del braç d'acord al càlcul del control*/
 myservo.write(positionDesired); /*Col·loquem l'exoesquelet
a la posició desitjada*/
}
```

### 2.1.11 Murs virtuals al control d'admitància.

Aquest codi és una continuació de l'anterior, on s'implementa unes limitacions virtuals (murs virtuals) per a que la posició de l'exoesquelet de braç no surti del seu rang de funcionament establert:

```
#include <Servo.h> /*Afegim la llibreria del servomotor*/

Servo myservo; /*Instanciem l'objecte myservo de la classe
Servo*/
int forceAnalogInPin = A3; /*Pin on es llegeix la força
mesurada pel sensor de força*/
const int forceOffset = 442; /*Constant que indica el valor
de la força Offset mesurat durant el calibratge del sensor*/
int forceIs; /*Variable on es guarda la força mesurada pel
sensor*/
int forceDesired = 0; /*Creem una variable on s'indica la
força desitjada que es vol aplicar*/
int positionDesired = 20; /*Creem una variable on s'indica
la posició desitjada (en graus) a la que es vol col·locar el
servomotor*/
float gain = 0.2; /*Valor del guany de control*/
int maxAngle = 95; /*Angle màxim al que pot arribar el braç
(paret superior)*/
int minAngle = 0; /*Angle mínim al que pot arribar el braç
(paret inferior)*/

void setup()
{
 Serial.begin(9600);
 myservo.attach(9); /*Indiquem quin és el pin en el que es
troba connectat el servomotor*/
 delay(1000);
}

void loop()
{
 forceIs = analogRead(forceAnalogInPin) - forceOffset;
/*Valor de la força mesurada*/
 delay(10);
 positionDesired -= gain*(forceIs - forceDesired); /*Posició
desitjada del braç d'acord al càlcul del control*/
```

```
 if(positionDesired > maxAngle) /*En cas que la posició
desitjada sigui superior a l'angle màxim del braç*/
 {
   positionDesired = maxAngle; /*La posició desitjada serà
el valor de l'angle màxim*/
 }
 else if(positionDesired < minAngle) /*En cas que la posició
desitjada sigui inferior a l'angle mínim del braç*/
 {
   positionDesired = minAngle; /*La posició desitjada serà
el valor de l'angle mínim*/
 }
 myservo.write(positionDesired); /*Es col·loca el servomotor
a la posició desitjada*/
 Serial.print("La posició desitjada és: "); /*Línies de codi
per mostrar per pantalla la posició desitjada*/
 Serial.println(positionDesired);
}
```

## 2.2 Unity.

En aquest apartat es veuran tots els scripts que s'han programat per fer que el videojoc del pong funcioni de forma correcta.

### 2.2.1 Control de la barra.

En el següent programa el que es fa és programar els controls del teclat que té el videojoc. Aquí es crearà una velocitat i s'indicarà quines tecles cal prémer per tal de moure la barra. A continuació es pot veure el codi:

```
using System.Collections; //Fem servir using per introduir
les llibreries que s'utilitzaran en aquest script.
using System.Collections.Generic;
using System.Collections.Specialized;
using System.Security.Cryptography;
using System.Threading;
using UnityEngine;

public class BarControl : MonoBehaviour //Es crea una classe
de tipus pública on es programarà el control de la barra.
{
 float speed = 5f; //Es declara una variable decimal
anomenada velocitat de valor 5.

 //Start() es crida abans de la primera actualització del
fotograma.
 void Start()
 {
   //Com que no hi ha cap inicialització en aquest programa,
aquest apartat es deixa en blanc.
```

```
 }

//Update() es crida una vegada per fotograma.
void Update()
 {

if(Input.GetKey(KeyCode.A)||Input.GetKey(KeyCode.LeftArrow)
) //En cas que es premi la tecla 'A' o la fletxa esquerra.
  {
   transform.position   +=   Vector3.left   *   speed   *
Time.deltaTime; //S'incrementa la posició cap a l'esquerra
d'acord a la multiplicació de la velocitat amb l'increment
del temps.
  }
if(Input.GetKey(KeyCode.D)||Input.GetKey(KeyCode.RightArrow
)) //En cas que es premi la tecla 'D' o la fletxa dreta.
  {
   transform.position   +=   Vector3.right   *   speed   *
Time.deltaTime; // S'incrementa la posició cap a la dreta
d'acord a la multiplicació de la velocitat amb l'increment
del temps.
  }
 }
}
```

### 2.2.2 Control de la pilota.

El següent codi serveix per aleatoritzar la velocitat inicial de la pilota cada vegada que s'inicia el joc del pong. Com que l'aleatorització de la velocitat inicial de la pilota només passa una vegada i és a l'inici, aquest codi s'escriu en l'apartat Start() i no s'escriu res a l'apartat Update(). A continuació es pot veure el codi que s'ha fet:

```
using System.Collections; //Fem servir using per introduir
les llibreries que s'utilitzaran en aquest script.
using System.Collections.Generic;
using System.Collections.Specialized;
using UnityEngine;

public class BallControl : MonoBehaviour //Es crea una classe
de tipus pública on es programarà el control de la pilota.
{

//Start() es crida abans de la primera actualització del
fotograma.

 void Start()
 {
   float speedx = Random.Range(5, 7); //Aquí es crea la
velocitat de la coordenada x de forma aleatòria dins d'un
rang entre 5 i 7.
```

```
   float speedy = Random.Range(2, 5); //Aquí es crea la
velocitat de la coordenada y de forma aleatòria dins d'un
rang entre 2 i 5.
   GetComponent<Rigidbody>().velocity = new Vector3(speedx,
speedy, 0); //Aquí s'assigna les velocitats que s'han generat
a la variable de l'asset Rigidbody de la pilota.
 }

//Update() es crida una vegada per fotograma.
 void Update()
 {
   //En aquest cas no s'ha d'escriure res aquí.
 }
}
```

### 2.2.3 Control del videojoc amb l'exoesquelet EduExo.

En aquest apartat es mostraran els programes que s'han utilitzat per controlar el videojoc del pong a través de la posició de l'exoesquelet de braç EduExo.

#### *2.2.3.1 Lectura i enviament de dades en la IDE d'Arduino.*

Primer de tot s'escriu el programa d'Arduino per llegir la posició del servomotor de l'exoesquelet de braç. Aquests valors es guarden en una variable tipus byte (de valors que van des de 0 fins a 255) i s'envien a través d'una interfície sèrie:

```
int servoAnalogInPin = A0; /*Pin on es llegeix la posició
del servomotor en bits*/
int posIs; /*Variable on es guarda la posició del servomotor
en bits*/
byte sendValue; /*Es crea una variable de tipus byte (0-255)
on es guardaran els valors de la posició mesurats*/

void setup()
{
 Serial.begin(9600);
 delay(1000);
}

void loop()
{
 posIs = analogRead(servoAnalogInPin); /*Procés per obtenir
i guardar la posició del motor*/
 sendValue = map(posIs, 130, 328, 0, 255); /*S'utilitza la
funció de mapejar per obtenir la dada de la posició, dins
del rang de moviment establert pel servomotor, amb un valor
dins del rang de 0-255*/
 Serial.write(sendValue); /*Es fa servir aquesta funció per
enviar un valor de byte binari per la interfície sèrie*/
 Serial.flush(); /*Amb aquesta funció assegurem que la
transmissió del byte ha finalitzat*/
```

```
 delay(20); /*Aquest retard de 20 ms assegura que no s'envien
massa dades per ser transmeses i processades per Unity*/
}
```

#### *2.2.3.2 Rebuda de dades y control de la barra en Unity.*

En aquest apartat es mostrarà el codi per llegir les dades que arriben per la comunicació sèrie i posicionar la barra del pong d'acord a la posició de l'exoesquelet de braç:

```
using System.Collections; //Fem servir using per introduir
les llibreries que s'utilitzaran en aquest script.
using System.Collections.Generic;
using System.Collections.Specialized;
using System.Security.Cryptography;
using System.Threading;
using UnityEngine;

using  System.IO.Ports;  //S'hi  inclouen  les  següents  3
llibreries per fer la comunicació sèrie.
using System.IO;
using System;

public class BarControl : MonoBehaviour //Es crea una classe
de tipus pública on es programarà el control de la barra.
{
 SerialPort  sp;  //Es  crea  una  variable  sp  del  tipus
SerialPort.

 //Start() es crida abans de la primera actualització del
fotograma.
 void Start()
 {
   sp = new SerialPort("COM4", 9600); //Aquí s'indica el port
de comunicació i també la velocitat de comunicació.
   sp.ReadTimeout = 10;
   sp.Open();
 }

//Update() es crida una vegada per fotograma.
void Update()
 {
   if(sp.IsOpen) //Si la comunicació sèrie està oberta.
   {
    try //Amb aquesta instrucció s'impedeix que el joc deixi
de  funcionar  en  cas  que  la  funció  de  lectura  sèrie  no
funcioni.
    {
     int value = sp.ReadByte(); //Es llegeix el byte rebut.
     float positionUnity = (10-((float)value/15)); //I amb
el  byte  rebut  es  mapeja  la  posició  en  el  sistema  de
```

```
coordenades de Unity i se li assigna a la variable decimal
positionUnity.
     transform.position = new Vector3(positionUnity,
transform.position.y, transform.position.z); //Aquí
s'assigna la posició mapejada a la barra.
    }
    catch(System.Exception e) //La declaració catch
s'activarà i mostrarà un missatge d'error quan hi hagi algun
error en la comunicació sèrie.
    {

    }
   }
 }
}
```

## 2.3 Guia avançada del sensor EMG MyoWare 2.0.

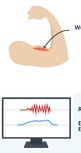
En aquest apartat és pot veure una guia avançada del sensor electromiogràfic (EMG) MyoWare 2.0. Aquest sensor és la versió millorada del sensor MyoWare utilitzat en aquest treball, no obstant, aquesta guia serveix per entendre el funcionament del sensor i algunes de les característiques que comparteixen.

MYOWARE® 2.0

# Advanced Guide

## How MyoWare works

The MyoWare measures muscle activity through the electric potential of the muscle, commonly referred to as surface electromyography *(EMG or sEMG for short)*. When your brain tells your muscle to flex, it sends an electrical signal to your muscle to start recruiting motor units *(the bundles of muscle fibers that generate the force behind your muscles)*.





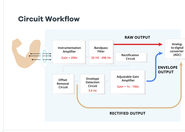




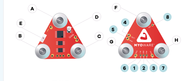
The harder you flex, the more motor units are recruited to generate greater muscle force.

The greater the number of motor units, the more the electrical activity of your muscle increases. The MyoWare will analyze this electrical activity and output an analog signal that represents how hard the muscle is being flexed. The harder you flex, the higher the MyoWare output voltage will go.

### Circuit Workflow




### Sensor Parts




**FRONT**

1. Ground (GND) snap for shields
2. Envelope (ENV) output strip for shields
3. Gain potentiometer (GAIN) knob
4. Gain potentiometer (GAIN) turn
5. Reference electrode snap socket
6. Reference electrode snap socket
7. Mid-muscle (MID) electrode snap socket
8. End-muscle (END) electrode snap socket

**BACK**

**Primary Pins**
1. Input Voltage (VIN)
2. Ground (GND)
3. Envelope (ENV) output

**Optional Pins**
4. Rectified (RECT) output
5. Raw EMG (RAW) output
6. Mid-muscle (MID) electrode
7. End-muscle (END) electrode
8. Reference (REF) electrode

## Why three electrodes?

**Mono vs. Bipolar**

EMG sensors can either have a two electrode (monopolar) or three electrode (bipolar) configuration.

### Monopolar Configuration

A single input electrode is placed over the body of the targeted muscle group. A reference electrode is placed in an adjacent electrically neutral location.


### Bipolar Configuration

Two input electrodes (e.g. MID and END) are placed on the body of the targeted muscle group. The first electrode is placed near the middle of the muscle body and the second electrode is placed 1-3 cm from the first electrode. A reference electrode is placed in an adjacent electrically neutral location.


The difference in voltage between the two electrodes is amplified with respect to the reference electrode. The advantage of this configuration is the common noise between the two electrodes is removed due to the amplifier's common mode rejection ratio (CMRR).

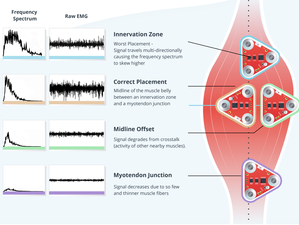
A bipolar EMG sensor, like the MyoWare, produces a much cleaner EMG signal with a much greater signal-to-noise ratio (SNR).

## Why is electrode placement important?[2]

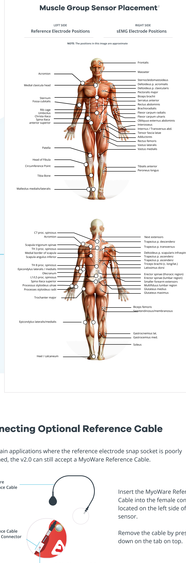
Proper electrode placement and orientation is essential to acquire consistent and quality signals with the MyoWare. For the best possible signal, place the electrodes on the belly of the target muscle between the nearest innervation zone and the myotendon junction-where the muscle fibers are most dense. Orientation-wise, the electrodes should form a line longitudinally parallel to the muscle fibers. This ensures the detecting surfaces intersect the same muscle fibers creating a better superimposed signal.[1]

### Raw EMG output

**DISCLAIMER:** Data show is for illustration purposes only; not actual data



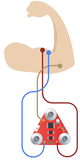

### Muscle Group Sensor Placement[3]




## Connecting Optional External Cables

The MyoWare has embedded electrode snaps right on the sensor board itself, replacing the need for a cable. However, if the on board snaps do not fit a user's specific application, an external cable can be connected to the board through three through hole pads shown above.

**PLEASE NOTE:** MyoWare 2.0 Cable Shield is recommended.



- **End** — Connect this to the cable leading to an electrode placed adjacent to the middle electrode towards the end of the muscle body
- **Middle** — Connect this pad to the cable leading to an electrode placed in the middle of the muscle body
- **Ref** — Connect this to the reference electrode. The reference electrode should be placed on an separate section of the body, such as the bony portion of the elbow or a nonadjacent muscle

## Connecting Optional Reference Cable

For certain applications where the reference electrode snap socket is poorly positioned, the v2.0 can still accept a MyoWare Reference Cable.




Insert the MyoWare Reference Cable into the female connector located on the left side of the sensor.

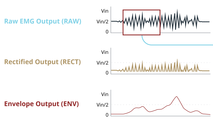
Remove the cable by pressing down on the tab on top.

The reference electrode socket remains active even with the cable inserted. Disable the socket by cutting the jumper trace on the top side of the sensor. Re-enable the socket by shorting the jumper pads with solder.


## Overview

The MyoWare is designed to be used directly with a microcontroller. It's primary output is not the raw EMG signal but rather the envelope of the amplified and rectified signal that is ideal to work with a microcontroller's analog-to-digital converter (ADC). However, MyoWare 2.0 also provides the raw and rectified signals.

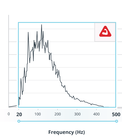
**DISCLAIMER:** Data show is for illustration purposes only; not actual data




## Power Spectrum

Surface EMG signals typically have an amplitude of 0 - 10 mV (peak to peak) and a frequency band of 10 - 500 Hz. MyoWare has a first-order passband of 20 - 500 Hz which is ideal for capturing the bulk of the power spectrum while removing unwanted signal sources such as motion artifacts.[2]

**DISCLAIMER:** Data show is for illustration purposes only; not actual data



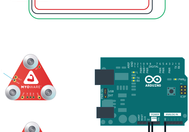

## Setting up Envelope (ENV) Output

To output the ENV signal, simply connect the ENV pin to one of your measuring device's analog input. Unlike the other two outputs, the ENV output has an additional gain stage that is adjustable via the gain potentiometer.

To adjust the gain, locate the gain potentiometer in the upper left corner of the sensor (marked as "GAIN"). Using a Phillips screwdriver, turn the potentiometer clockwise to increase the output gain; turn the potentiometer counterclockwise to reduce the gain.

**QUICK TIP**

We recommend for users to get their sensor setup working reliably prior to adjusting the gain. The default gain setting should be appropriate for most applications.




## Setting up Raw (RAW) Output

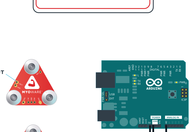
Like the previous version, MyoWare 2.0 has the ability to output an amplified raw EMG signal.

To output the raw EMG signal, simply connect the RAW pin to your measuring device instead of the ENV pin.

**QUICK TIP**

The RAW output is centered about an offset voltage of VIN/2, see above. It is important to ensure VIN is the max voltage of the MCU's analog to digital converter. This will assure that you completely see both positive and negative portions of the waveform.

The amplification for the RAW output is not adjustable via the GAIN potentiometer.



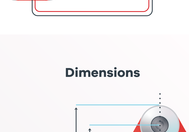

## Setting up Rectified (RECT) Output

MyoWare 2.0 now has the ability to output the amplified and full-wave rectified signal.

To output the rectified signal, simply connect the RECT pin to your measuring device instead of the ENV pin.

**QUICK TIP**

The amplification for the RECT output is not adjustable via the GAIN potentiometer.




## Technical Specifications

**Supply Voltage:**
min. = 2.7V,
typ. = 3.3V or +5V,
max. = +5.47V

**Input Bias Current:**
250 pA, max 1 nA

**Input Impedance:**
800

**Common Mode Rejection Ratio (CMRR):**
140 dB

**Ideal Gain Equation:**
Raw (RAW): G = 200
Rectified (RECT): G= 200
Envelope (ENV): G = 200 * R / 1 kOhm
R is the resistance of the gain potentiometer in kOhm

**Filters:**
High-pass Filter: Active 1st order, fc = 20.8 Hz,[2] -20dB

Low-pass Filter: Active 1st order, fc = 498.4 Hz,[2] -20dB

**Envelope Detection:**
Linear, Passive 1st order, fc = 3.6 Hz,[2] -20 dB

**Rectification Method:**
Full-wave

**Sample Rate:**
*Not applicable - MyoWare Sensor is analog. See measuring device specifications.*

## Dimensions


## References


[1] De Luca, C. J. (1997). The use of surface electromyography in Biomechanics. Journal of Applied Biomechanics, 13(2), 135-163. https://doi.org/10.1123/jab.13.2.135

[2] Hermens, H. J., Freriks, B., Merletti, R., Stegeman, D., Blok, J., Rau, G., Disselhorst-Klug, C., Hägg, G. (1999). *SENIAM 8: European Recommendations for Surface ElectroMyoGraphy* (2nd ed.). Roessingh Research and Development b.v., ISBN 90-75452-15-2.

[3] Winter, D. A. (2009). *Biomechanics and motor control of human movement* (4th ed.). Wiley. ISBN 978-0-470-39818-0.