YILDIZ TECHNICAL UNIVERSITY
FACULTY OF MECHANICAL ENGINEERING

AN EXOSKELETON LEG MECHANISM CONTROLLABLE VIA ELECTROMYOGRAPHY SIGNALS

1506A601 Erdoğan Yıldız

1406A019 Erdem Işık

SENIOR DESIGN PROJECT REPORT

PREPARED BY MECHATRONICS ENGINEERING DEPARTMENT

Project Consultant: Assoc. Prof. Cüneyt Yılmaz

İSTANBUL, 2019

YILDIZ TECHNICAL UNIVERSITY
FACULTY OF MECHANICAL ENGINEERING

AN EXOSKELETON LEG MECHANISM CONTROLLABLE VIA ELECTROMYOGRAPHY SIGNALS

1506A601 Erdoğan Yıldız

1406A019 Erdem Işık

SENIOR DESIGN PROJECT REPORT

PREPARED BY MECHATRONICS ENGINEERING DEPARTMENT

Project Consultant: Assoc. Prof. Cüneyt Yılmaz

İSTANBUL, 2019

CONTENTS

REVISION HISTORY

Date	Rev. No	Definition	Author(s)
15/05/2019	1.0	Initial document	Erdoğan Yıldız Erdem Işık
25/05/2019	1.1	Revised document	Erdoğan Yıldız Erdem Işık
01/06/2019	1.2	Revised document	Erdoğan Yıldız Erdem Işık

LIST OF SYMBOLS

g	*Gravitational Acceleration S*
m	*Mass*
r	*Position of Center of Mass*
l	*Length*
θ	*Joint Angle*
b	*Damping Coefficient*
K	*Kinetic Energy*
P	*Potential Energy*
q	*Vector of Generalized Coordinates*
Y	*Generalized Forces*
L	*Lagrangian*
M	*Inertia Matrix*
V	*Coriolis/Centripetal Vector*
G	*Gravity Vector*
τ	*Torque Vector*
Kp	*Proportional Term*
Kv	*Derivative Term*
Ki	*Integral Term*
u	*Auxiliary Control Signal*

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Firstly, we would like to express our sincere gratitude to our advisor Assoc. Prof. Cüneyt Yılmaz for his continuous support of our project, for his patience, motivation, and immense knowledge. His guidance helped us in all the time of research and writing of this thesis. We could not have imagined having a better advisor and mentor for our thesis study.

Besides our advisor, we would like to thank the Research Assistant Mehmet İşcan for his insightful comments, encouragement, and guidance.

Also, we are grateful to Tubitak for believing in our research and financially supporting the project through the Tubitak 2209-B Industry Oriented Thesis Support Program.

ABSTRACT

In this senior design project, design and implementation of an assistive lower-extremity exoskeleton leg mechanism has been developed. The leg mechanism is primarily used to support people with walking difficulties due to orthopedic or neurological disorders, trauma or old age. The thesis consists of a mechanical design of leg mechanism, mathematical derivation of equations of motion, dynamic modelling, MATLAB modelling, controller design, design verifications, simulations, artificial neural network application, system implementation and integration.

Within the scope of the thesis work, the mechanical design of the exoskeleton is performed using Solidworks software, the stress analysis of the critical positions where the mechanism is subjected to maximum loading is also carried out using Solidworks software. Equations of motions for a rigid double pendulum model that represents the leg mechanism has been derived and a MATLAB Simulink model has been built based on the governing equations. Also, another MATLAB model has been built by transferring a solid model created on Solidworks software to the MATLAB Simulink environment. A PID computed torque controller is designed to control both mathematical and simscape models. Both MATLAB models are fed by the desired walking trajectory and design verifications and validations are done by examining different kind of outputs such as tracking error and required torque values. A real-time MATLAB graphical user interface is created in order to acquire data from myo armband. Later, an artificial neural network algorithm is used in order to classify the EMG signals taken from the user's arm via myo armband. After implementation and system integration, results are obtained and compared with simulation results.

Keywords: Exoskeleton,Lower Extremity Exoskeleton, Wearable Exoskeleton, Walking Assistive Exoskeleton, Computed Torque Control, PID Control,MATLAB Simulation, Muscle-Signal Based Control, Artificial Neural Networks.

1. INTRODUCTION

Biomechatronics, which is a new field as one of the working areas of mechatronics engineering, locates the biology science on its basis and main purpose in addition to the mechanics, electronics, computer and control disciplines to partially and/or fully support the problematic body functions as Assoc. Prof. Erhan Akdoğan said in *Otomasyon Dergisi journal* [1].This area includes a wide range of working groups, ranging from intelligent rehabilitation technologies to artificial cardiac pacemakers, from intelligent insulin pumps to biological sensors, from analysis of the human motion to assisted exoskeleton robots. Perhaps one of the most common application outcomes of biomechatronics is the exoskeleton robots. Exoskeleton robots are devices that are intended to be used for movement and/or rehabilitation. Rehabilitation purposed exoskeleton devices are designed for the physiotherapy of patients who cannot use their limbs due to cerebrovascular and neuromuscular diseases (Rosen J et al. 2007)[20].Other types of auxiliary exoskeleton systems are designed for the workers to be safer and more confortable on industrial production lines (Rosen J. et al. 2005) [21] or for disabled users to improve their living standards or even for military purposes (HEXAR, Oct. 2014) [15].

The aim of this project is to build a stable system that is able to understand the intention of the user by processing the electromyography (EMG) signals taken from the user's arm and help the movement being performed. The functional relationship between the EMG signals and the exoskeleton leg mechanism is intended to contribute to the user movement. Basically, the natural movement of the human leg is to be imitated in a functional way.

The assistive exoskeleton leg mechanism is designed and implemented in a way that can be adjustable for individuals of different body sizes, the mechanism is designed to be fixed with safety parts from the waist and leg regions.

Literature research has been done for assistive or rehabilitation purposed exoskeleton leg mechanisms. By benefiting this research carried out, an assistive exoskeleton leg mechanism that can understand the intention of the user and assist the ongoing movement has been implemented.

1.1 Goal of the Project

The main goal of this project is to develop an exoskeleton leg mechanism that will help the elderly and/or disabled people to perform movements such as walking, climbing stairs, sitting and getting up; and increasing the performance of heavy load workers while minimizing the occupational disease and deformations at the same time.

In this project, it is aimed to run tests and simulations of the application of artificial intelligence and control algorithms by extracting the dynamic equations of a lower limb robot with 2 limbs. To do this, the following methods have been performed:

- Solidworks modeling of the mechanism
- Running simulations in order to obtain required parameters to select actuators by transferring the Solidworks model into MATLAB-Simulink environment
- Selecting material and producing the parts according to the dynamic equations and simulations
- Using an artificial neural network model to classify EMG signals
- Using computed torque control as a control algorithm
- Writing a graphical user interface

By following the above steps, it is aimed to present a stable system which can interpret EMG signals taken from the user's arm, understand the movement intention of the user by classifying different hand gestures (flexion, extension, fist, and relaxation) and drive the actuators of the exoskeleton mechanism to assist the ongoing movement of the user.

1.2 Motivation

According to recent research Chen, Bing, Journal of Orthopaedic Translation 5, 15.3% of the world population and 16.4% of the Europe population consists of individuals who have middle and severe lower limb-related disabilities according to Chen, Bing, et al. (2016) [5]. In

addition, more than 30% of the Europe population consists of people over 60 years old. According to TURKSTAT 2017 data, the number of people over the age of 65 in Turkey is close to 7 million. One of the biggest problems of the elderly population is the interruption of daily physical activities due to significant muscle strength and muscle mass loss. These individuals account for 11.5% of the world human population as it has been said in Chan, Margaret, and R. B. Zoelick. "World Report On Disability." [4].

The primary motivation for the project process has been to increase the living standards of elderly people and/or people who have lower limb-related disabilities. Besides, the exoskeleton mechanism can be used to improve the working conditions of industrial production line workers and to minimize occupational deformations and diseases.

Also, the global exoskeleton market size, which was $ 25.4 million in 2015, is expected to reach $ 3.5 billion by 2025. This exponential increase in market size is a great source of motivation and opportunity for us. Fig. 1.1 below shows the exoskeleton market size and growth prospects from Exoskeleton Market Size, Analysis & Research | Industry Report [9].

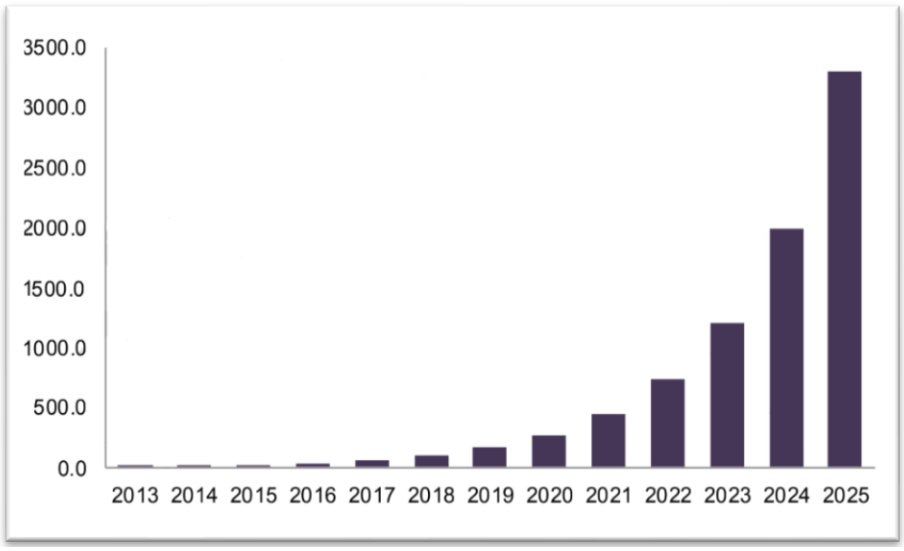


Figure 1.1. Market Size and Growth Prospects 2013-2025 (USD Million)

1.3 Literature Research

Initial studies on the exoskeletons date back about 55 years. In Fig. 1.2, HARDIMAN (Human Augmentation Research and Development) project, developed by the General Electric company is shown from Makinson, B J. et al. (1971) [13]. It was one of the first noteworthy studies about exoskeletons. HARDIMAN was a master-slave controlled full-body

exoskeleton, that is planned to carry 680 kgs load. However, the experiments with the developed exoskeleton resulted in uncontrolled movements that may harm the user. After many unsuccessful

attempts, they decided to focus on only a single-arm prototype that can carry 340 kgs load. However, only the developed arm itself was about 750 kgs, which caused the project to be terminated in 1970.

Figure 1.2 HARDIMAN Exoskeleton

Although studies on exoskeletons were interrupted after many unsuccessful attempts, studies on the subject gained momentum at the beginning of the 2000s in parallel with the developing technology.

In Fig. 1.3, a full body exoskeleton system called Wearable Power Assist Suit developed by the Kanagawa Institute of Technology is shown in Naruse et al. (2005) [17]. The primary purpose of the project was to enable nurses to carry patients in their arms. Total weight of the device was 18kgs and in order to move the knee, elbow and waist joints of the device, 5 pneumatic actuators fed by micropumps were used. Control of the device is performed by EMG signals measured from the specific muscle groups of the user. As a result of the experiments, a 64 kgs-weighted nurse was able to carry a patient weighing 70 kgs.

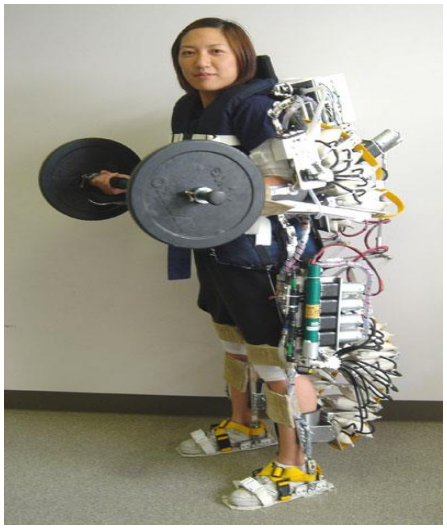


Figure 1.3 Wearable Power Assist Suit

The use of exoskeletons in order to support the heavy load-bearing infantry soldiers has drawn the attention of the US government. Defense Advanced Research Projects Agency (DARPA) has started to provide significant support to many projects related to the subject in different organizations.

In Fig. 1.4, a load-bearing exoskeleton system called BLEEX (Berkeley Lower Extremity Exoskeleton) developed by Berkeley University is shown [6]. BLEEX was the first load-bearing autonomous exoskeleton of the World. The hip, knee and ankle joints of the device were actuated with electrohydraulic cylinders, and more than 40 sensors connected to each other, used in order to control the device. Despite its lack of maneuverability due to its cumbersome structure and the inability of its actuators to provide sufficient power, BLEEX made a significant contribution to exoskeleton studies. In the BLEEX project, many types of actuators and control algorithms that can be used in the exoskeleton have been tried and compared with each other.

Figure 1.4 BLEEX

All three exoskeleton studies above were full body exoskeletons, but all three of them have a special significance on the development process of the exoskeleton industry. That makes them worth mentioning. But now, lower extremities exoskeleton study examples are going to be given, which are more related to the subject of this thesis.

In Fig. 1.5, a lower extremities exoskeleton system called REX developed by a New Zealand company Rex Bionics is shown in Rex Bionics Ltd. (2018) [19]. It helps wheelchair users to regain their abilities such as walking, stairway-climbing, sitting-up. The REX weighs approximately 38 kgs and has a portable power unit. Electric motors placed on the hips and knees are used as actuators. The movement control of the device is provided by a joystick connected to the waist.

Figure 1.5 REX

In Fig. 1.6, another lower-limb exoskeleton called ReWalk designed by an Israeli company Argo Medical is shown in Esquenazi et al. (2012) [8]. The primary purpose of the project was to help wheelchair users to regain their ability to walk. The device has a portable power unit and it has electric motors placed on the hips and knees as an actuator. ReWalk can perform predefined movements such as walking, stair-climbing, standing up, which can be selected from a control unit connected to the user's arm.

Figure 1.6 ReWalk

In Fig. 1.7, the Vanderbilt Exoskeleton is shown from Vanderbilt University (November 2018) [23]. It is developed to provide gait assistance to the stroke and spinal cord injured (SCI) population. Clinical evaluations have validated its ability to restore legged mobility to individuals with impaired mobility, including the following functionalities: walking, standing, sit-to-stand, stand-to-sit, stair ascent and descent, and functional electrical stimulation (FES). The exoskeleton weighs 12.3 kgs and has been tested on users up to 90 kgs in weight. It uses onboard embedded microprocessors and sensors to determine the user's current state and intentions and it provides joint torques at the hips and knees to generate motion.

Figure 1.7 Vanderbilt Exoskeleton

2. REQUIREMENT SPECIFICATIONS

2.1 Market Requirements

Market requirements are listed below:

1. User’s movement must be perceived truly
2. The mechanism must be adjustable for different body sizes
3. User’s safety must be provided
4. The mechanism must be compatible with both legs
5. The mechanism must increase the user’s movement ability
6. The mechanism must not disturb the balance of the user
7. Easy usability
8. Affordable price

Table 2.1. Requirements Specifications Table

Market Expectations	Technical Requirements	Explanation
1,6	1. EMG sensor must be able to acquire signals between 0-500 Hz frequency.	The usable energy of the muscle signal is limited to the 0 - 500 Hz frequency range.
2,7,8	2. Adjustability in system height for users 1.75 ± 0.10 m tall.	Decided based on the anthropometric values of average human height.
2,4,5,7,8	3. 4 DOF; 2 hip joint, 1 knee joints and 1 ankle joints.	Decided by looking at similar products in the market.
5,6,8	4. Disability rate coverage: 50%	Decided by looking at similar products in the market and also by considering the budget.
2,8	5. As a boundary user, 100 kg adult individual	Decided based on average adult human weight (62kg). Approximately %50 tolerance.
5,6	6. Default speed: 126 RPM	126 RPM is the normal gait speed for a 1.75m user.
3,5,6	7. Operational range: -120 to 60 degree for hip joint, 120 to 0 degree for knee joint (extension/flexion)	Decided based on the average human leg's normal extension/flexion angles
3,5,6	8. Abduction/Adduction allowance between -30 to 30 degrees	Decided based on the average human leg's normal abduction/Adduction angles.
5,6,7,8	9. Mechanism weight: less than 6 kg	Decided by looking at similar products in the market.

2.2 Technical Requirements

Technical requirements are examined in 4 sub-topics as; Actuators, Emg Sensor, Controller, Drivetrain/Frame of the Mechanism and the Material to be Used.

2.2.1 Actuators

The required articulation torque and joint velocities are very important for the movement of a lower limb exoskeleton. In order to select the actuator that meets the required torque and speed values, as a first step, the 3D model of the mechanism has been done and the material

selection has been applied in SolidWorks environment. After the modelling is done, the Solidworks model has been transferred into the MATLAB-Simulink environment. In accordance with the anthropometric values and boundary user selection, different MATLAB simulations have been done and the required torque values have been obtained as can be seen in Figures 5.12 and 5.21. Simulation results showed that required torque value varies between -4 and +5 N.m. After market research considering these required torque value, default speed value, price, size, easy controllability, etc. model of the actuator that will be used in this project has been selected as Dynamixel MX-106T. The technical specifications of the Dynamixel MX-106T have given in Table 2.2.

Table 2.2. Technical specifications of the Dynamixel MX-106T

Model Name		Dynamixel MX-106T
MCU		Cortex-M3 (72 MHz, 32 bit)
Dimensions (WxHxD) [mm]		40.2 X 65.1 X 46.0
Weight [g]		153.00
Input Voltage	**Min. [V]**	10.0
	Recommended [V]	12.0
	Max. [V]	14.8
Performance Characteristics	**Voltage [V]**	12.0
	Stall Torque [N.m]	8.40
	Stall Current [A]	5.2
	No Load Speed [rpm]	45.0
	No Load Current [A]	0.17
Resolution	**Resolution [deg/pulse]**	0.0879
	Step [pulse]	4,096

2.2.2 EMG Sensor

Usable energy of the EMG signals is limited to the 0 - 500 Hz frequency, with the dominant energy being in the 50 – 150 Hz range according to Mozhanova et al. (2017) [16]. After the technical requirements have been obtained, the market research has been done and it has been decided to use the myo armband as the EMG sensor which can be seen in Fig. 2.1. Myo armband has 8 stainless steel EMG sensors on it. Myo armband sends the EMG signals with 8-bit resolution with 200 Hz sampling frequency via Bluetooth. In addition to these, myo armband has 3 axis gyroscope and 3 axis accelerometer which are working with 50 Hz sampling frequencies. It is important to have a good resolution on the EMG sensor, especially while classifying the signals by artificial neural networks.

Figure 2.1 Myo Armband

2.2.3 Controller

As the controller, a personal computer is used for the demo version. After the mobilization it is planned to use Raspberry Pi 3. It has Quad Core Arm Cortex-A53 CPU, 1GB LPDDR2 (900 MHz) RAM, Bluetooth 4.1 classic and microSD storage port on it. As it is accessible and powerful enough to run A.I. algorithms and classify the signals, it is a good choice.

2.2.4 Drivetrain / Frame of the Mechanism and the Material to Be Used

According to the actuator selection and required speed/torque values, it has been decided to use a drivetrain with the module of 3. In order to select the material that is used in the frame of the mechanism, Solidworks simulations is used. Stress, displacement and strain tests are applied with 150% nominal stress. As a result, the material that is used in the frame is selected as Aluminum LM24.

3. THEORETICAL BACKGROUND INFORMATION

3.1 Control Method

The top 4 most known control methods for exoskeletons are; user commanded control, predefined motion control, direct force feedback control and the method which is planned to be used in this exoskeleton mechanism: muscle signal-based control. In this control method, the EMG signals taken from the user's arm with the help of myo armband are be used to control the actuators.

As motion control method, the computed torque control method is used. Computed torque control method combines the benefits of a good dynamic model and the stabilization of the PID controller.

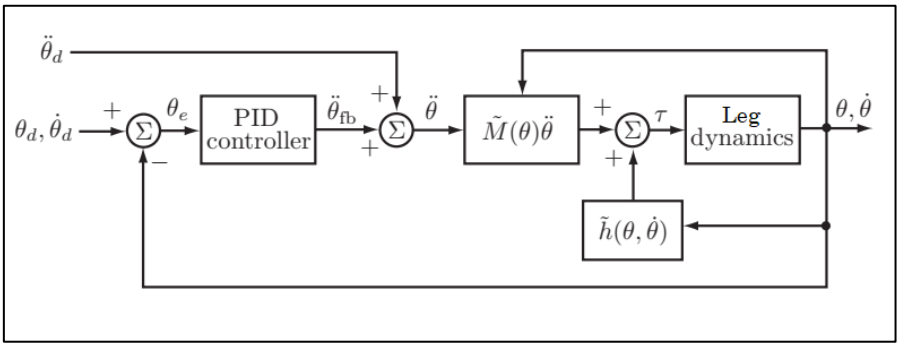


Figure 3.1 Computed Torque Control Method

Figure 3.1 shows a block diagram of the computed torque control method. The measured position (θ) and the velocity ($\dot{\theta}$) of the mechanism, inertia matrix (M-tilde) and h-tilde vector can be seen in Fig. 3.1. This controller provides good performance when the dynamic model of the mechanism is reasonably accurate. If the dynamic model is poor, then using this model as the controller could hurt the performance according to the Lynch, K. (2017) [12].

$$\tau = \widetilde{M}\left(\ddot{\theta}_d + K_p\theta_e + K_i\int \theta_e.dt + K_d\dot{\theta}_e\right) + \tilde{h}(\theta,\dot{\theta}) \qquad (3.1)$$

The first term of Equation 3.1 represents M-tilde times acceleration which is the feedforward acceleration at this time instant, plus an acceleration generated by the PID controller. M-tilde

matrix turns the feedforward plus feedback acceleration into a joint torque. Second term h-tilde is estimated to balance friction and gravity at the current state. If the error is always zero, this control law reduces to the feedforward control (M is the scalar inertia of the link about the revolute joint, H is the gravitational and frictional terms lumped together).

3.2 EMG Signals

EMG signal is a biomedical outcome that measures electrical currents generated in muscles during its contraction representing neuromuscular activities. The nervous system always controls muscle activity (contraction/relaxation). Hence, EMG signal is a complicated signal, which is controlled by the nervous system and it is dependent on the anatomical and physiological properties of muscles. EMG signal acquires noise while traveling through different tissues. Moreover, the EMG detector, particularly if it is at the surface of the skin, collects signals from different motor units at a time which may generate interaction of different signals. Detection of EMG signals with powerful and advance methodologies is becoming a very important requirement according to Raez M.B.I. et al. (2006) [18]. Unlike the other control methods for exoskeletons, using EMG signals maximizes the utility of the exoskeleton and makes it easier for the end user.

3.3 Machine Learning With Artificial Neural Network

To be able to help the ongoing movement, the system has to understand the intention of the user. In order to achieve this, EMG signals have to be classified. Signals coming from the user's arm are classified for 4 different hand gestures such as flexion, extension, fist and relaxation. EMG signals are too complicated to classify with the human eye and there is no mathematical model or explanation for it. In order to classify the EMG signals, supervised machine learning methods mentioned in İşcan M. et al. (2018) are used [10]. Sample EMG signals are taken while the main 4 movements are being performed (flexion, extension, fist and relaxation). These samples are used as training data for the algorithm. Optimum hidden neuron and layer numbers are decided by tests. After obtaining network parameters, the neural network became ready to classify real-time signals.

3.4 Generating Desired Path

Besides evaluating the mathematical model and the dynamic analysis of the system, another important problem is to generate the desired path for the exoskeleton to follow.

In robot manipulators, a path and/or trajectory is needed to provide the capability to move the manipulator and its end effector or the mobile robot from the initial posture to the final posture. Although the terms path and trajectory are often confused and used as synonyms in informal discussions, path consists of ordered location of points in the space (either the joint space or the operational space), which the robot should follow. On the other hand, a trajectory is a path plus velocities and accelerations in its every point according to the Hlaváč, Václav. "Robot trajectory generation." [11].

For planning a robot's motion one can use one of many techniques. One of the most frequently used methods in the literature is to define path either the joint space or the operational space. Since the motor does not understand the operational space coordinates, the coordinates must be converted to the joint space parameters via inverse kinematics. In both cases, there are advantages and disadvantages such as computational expensive due to inverse kinematics, tough obstacle avoidance obtention, and singularity problems. Another method is to create a path and/or trajectory based on a pre-determined dataset, especially in biomedical and bio robotics fields. One of the enormous advantages of this method is obtaining a more accurate result in the motions that are hard to compute or expressed via functions.

For the demo system, the desired path is based on the research and analysis of the gait cycle from Bovi et al. (2011) [2]. The reason to choose a database as the base reference is to obtain real-like movement of the limb in order to provide rather smooth locomotion. Note that, data acquisition interface allows us to record and mimic any user path. This feature can be used for future stages of the project.

A vast portion of the lower extremity locomotion consists of walking. Therefore, the standardized gait cycle term has been created. A gait cycle is the period or sequence of events or movements during locomotion in which one-foot contacts the ground to when that same foot again contacts the ground and involves forward propulsion of the center of gravity. A single gait cycle is also known as a stride (Whittle et al. 1993) [24].

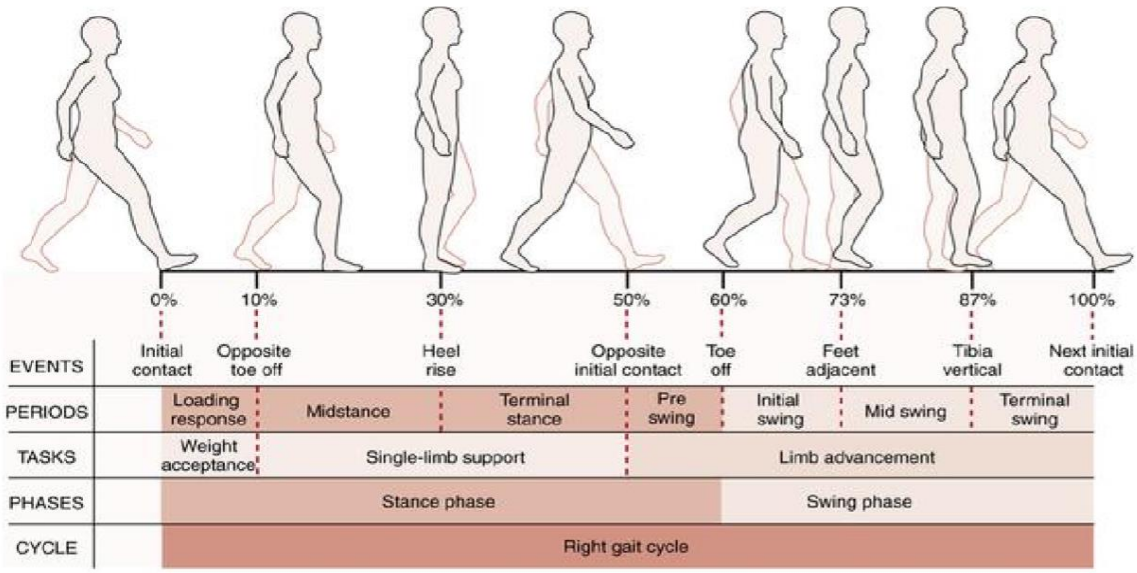


Figure 3.2 Single Gait Cycle, Willson et al. (2011) [25]

In the Bovi et al. [3], many features of the human leg were measured on 40 healthy subjects divided into two segments as young and adult regarding their age spawn when walking with different speeds, and stair ascending/descending. Among the findings of this research, joint rotations were the based parameters of this study in terms of the desired path, which will be used in simulation and control of the mechanism. The needed hip and knee joint angles on a non-gradient, obstacle-free, plain road at natural walking speed during a single gait cycle were represented at Fig. 3.3 and Fig. 3.4, respectively.

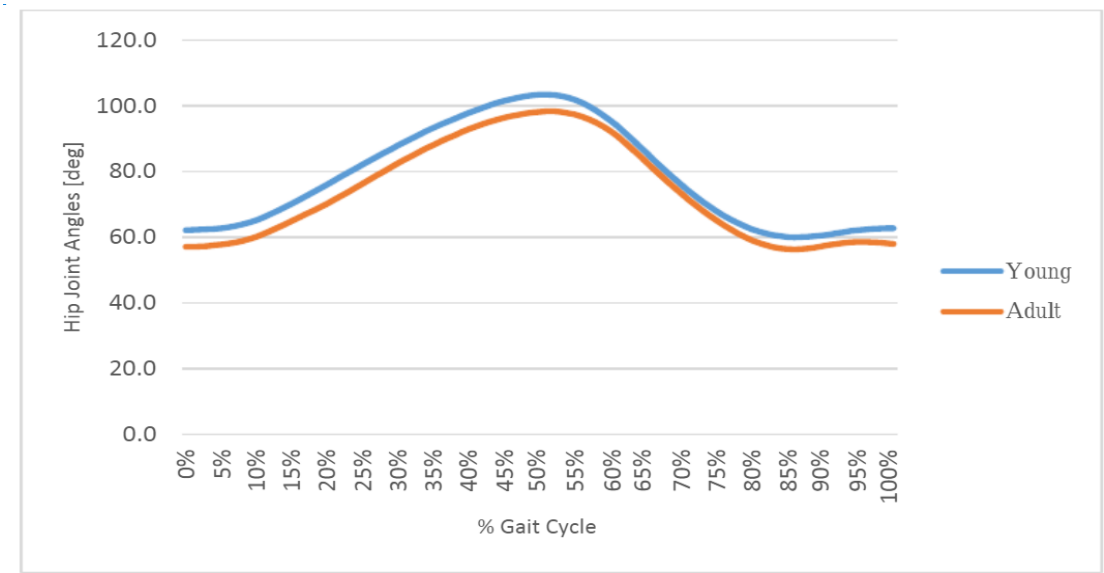


Figure 3.3 Hip joint angles on a non-gradient, obstacle-free, plain road at a natural walking speed during a asingle gait cycle

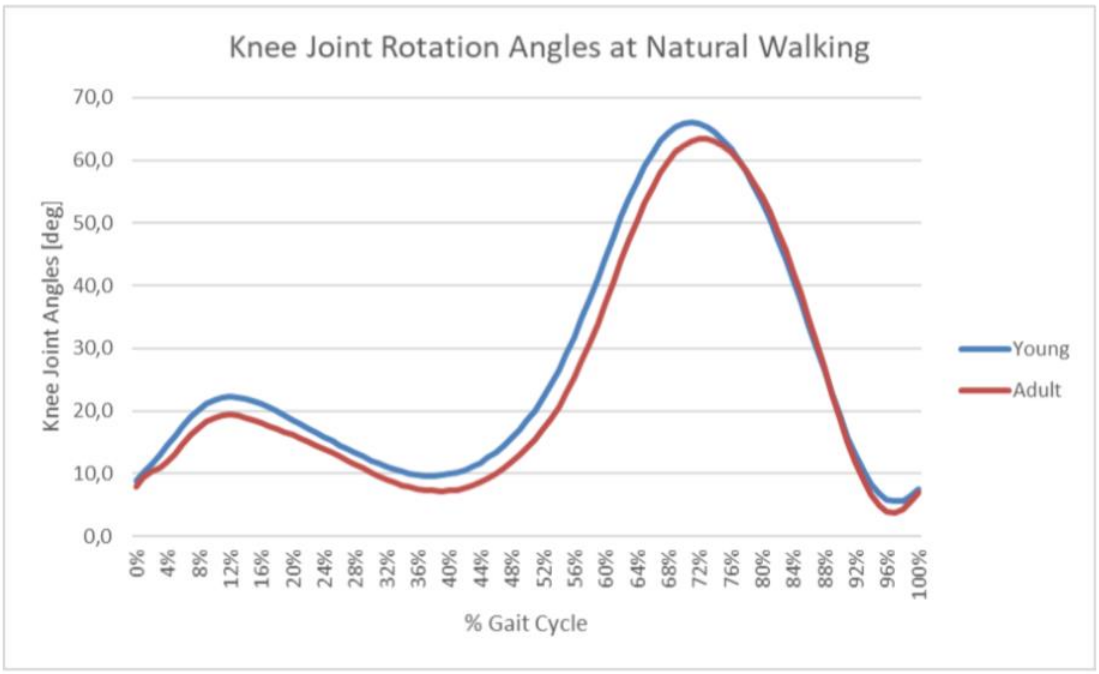


Figure 3.4 Knee joint angles on a non-gradient, obstacle-free, plain road at a natural walking speed during a asingle gait cycle

After the mechanism was fed with these aforementioned reference values, the sole of the foot was expected to follow a path similar to a water droplet that was converted ninety degrees counter-clockwise. This path can be observed in the Cartesian Space, as in Fig. 3.5.

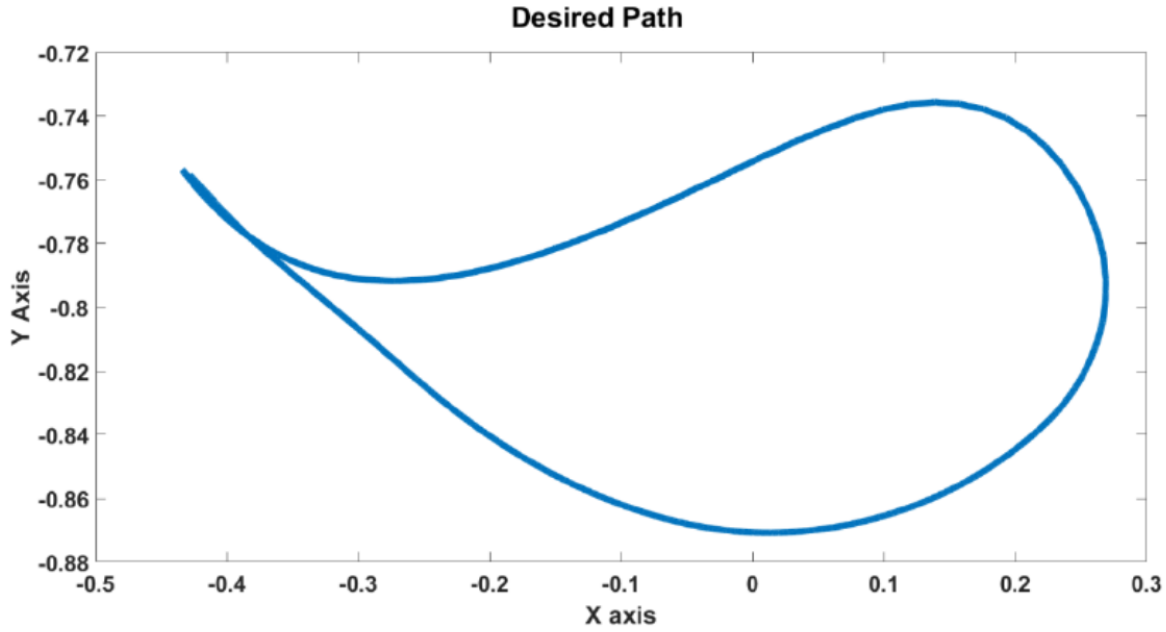


Figure 3.5 Desired path of the sole of the exoskeleton foot in the Cartesian Space

4. DESIGN

4.1 Mechanical Design

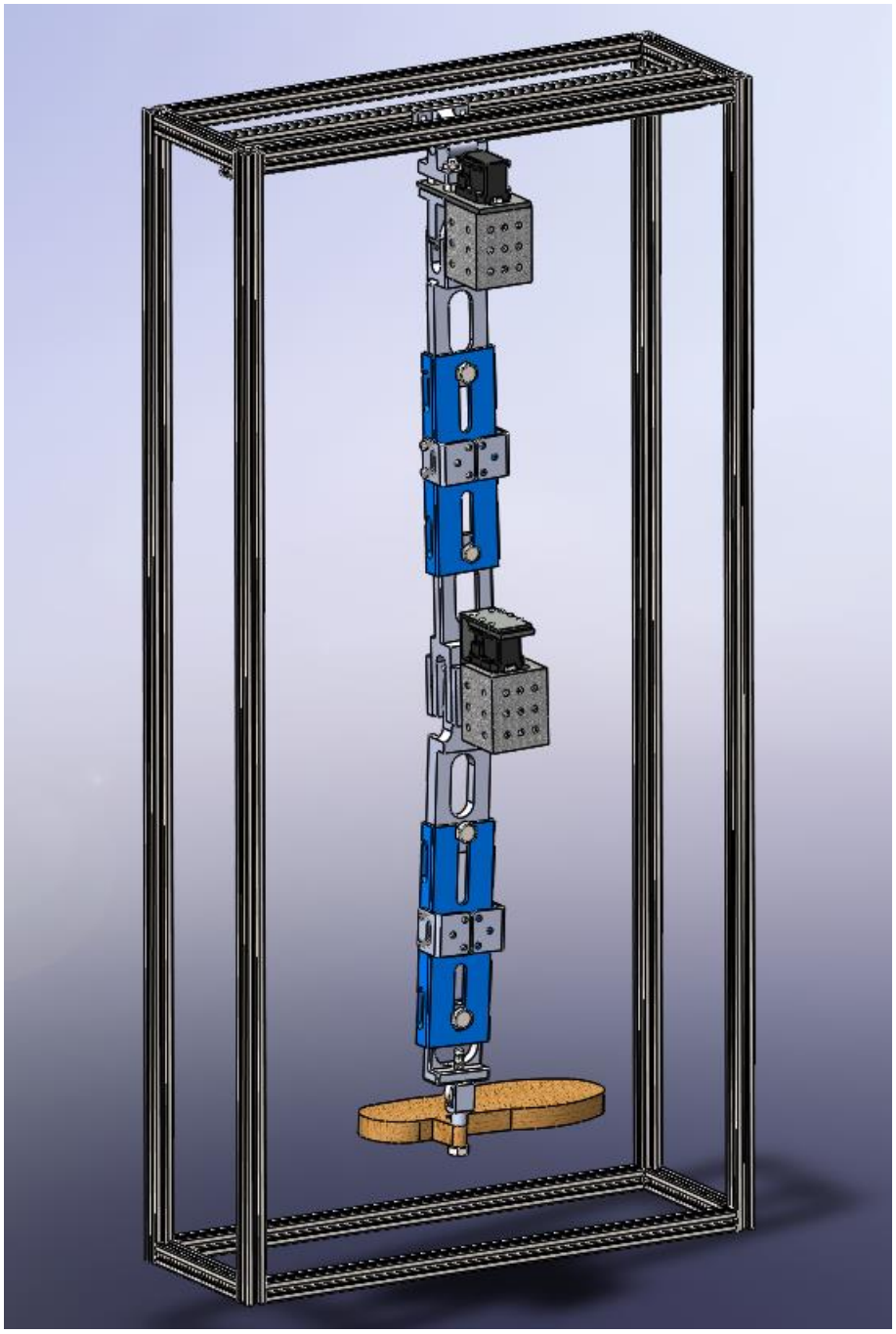


Figure 4.1 Whole View Of The Mechanical Design

Figure 4.2 Front View Of The Mechanism

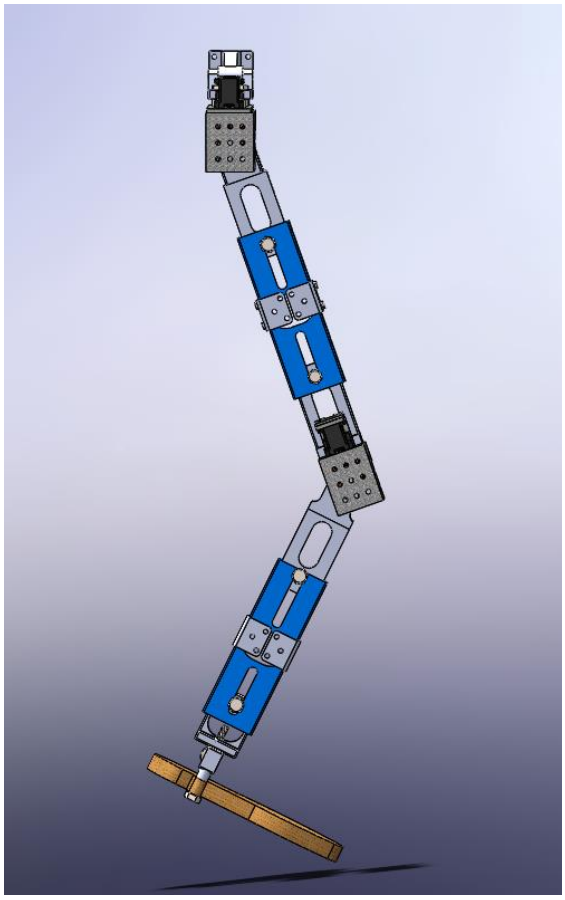


Figure 4.3 Initial Stepping Move

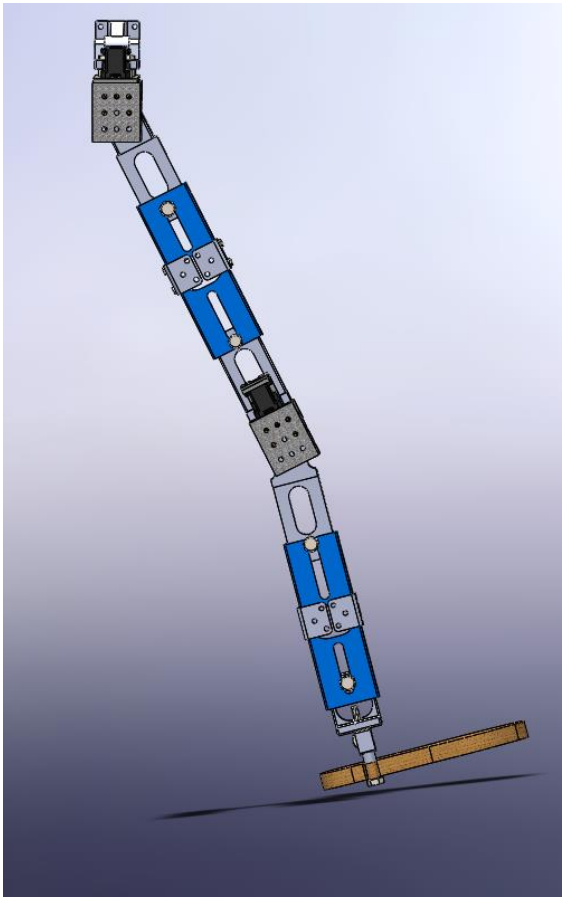


Figure 4.4 Final Stepping Move

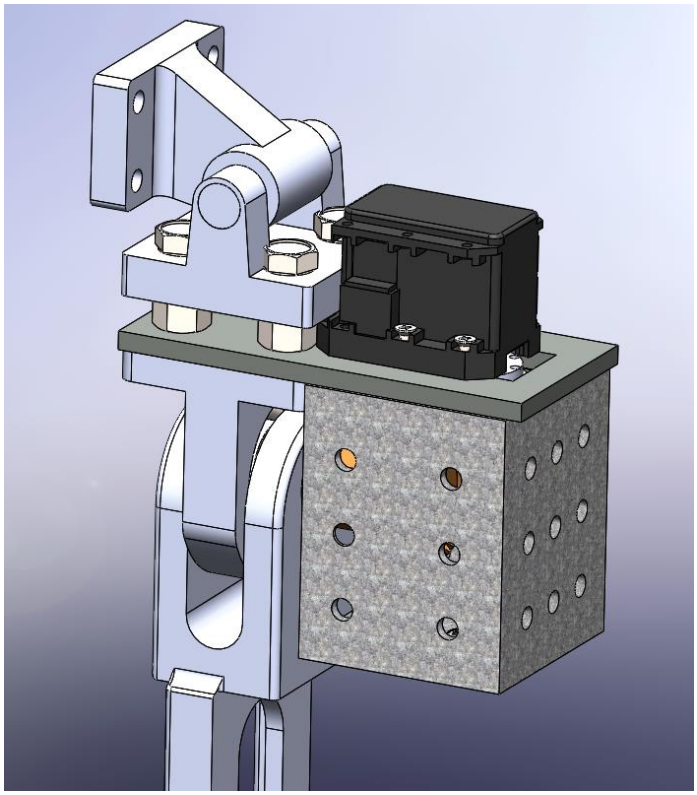


Figure 4.5 Cross View Of The Actuator Hip Joint Connection

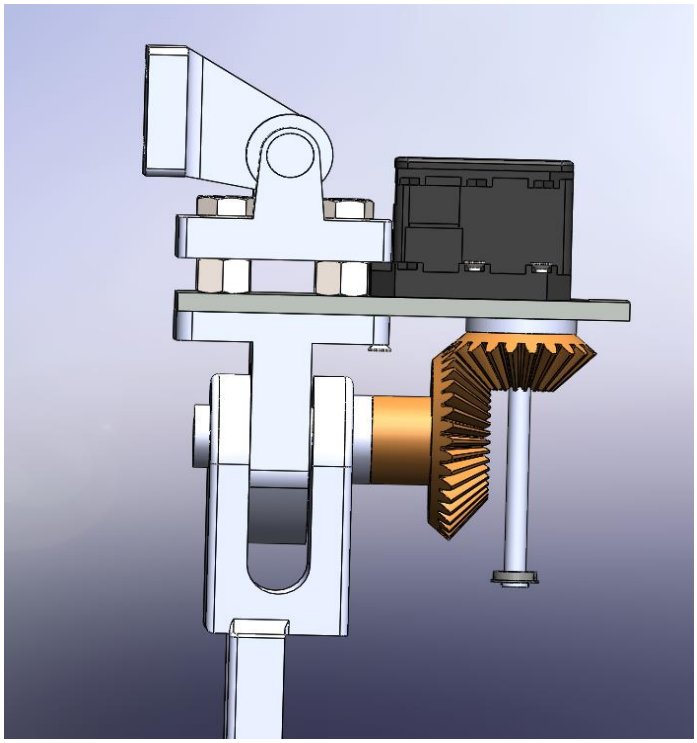


Figure 4.5 Side View Of The Actuator Hip Joint Connection Without Cover Part

Figure 4.6 Front View Of The User Mounting Strap

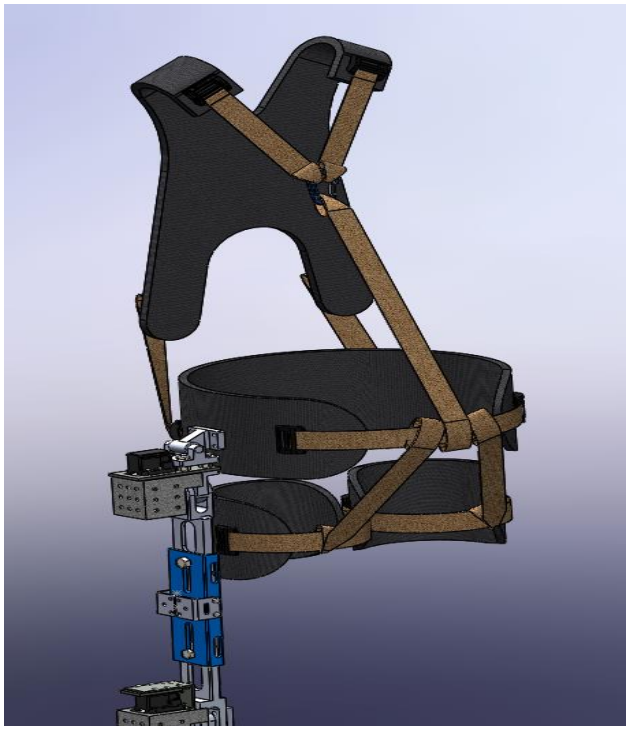


Figure 4.7 Cross View Of The User Mounting Strap

4.2 System Block Diagrams

In Fig. 4.8, the basic flow chart of the system is shown. EMG signals are taken from the user's arm and the movement intention of the user is understood by benefiting the artificial neural network, finally, the leg mechanism is controlled so that it can assist the ongoing movement of the user.

Figure 4.8 Basic Flow Chart

In Fig. 4.9, detailed working principle of the system is shown.

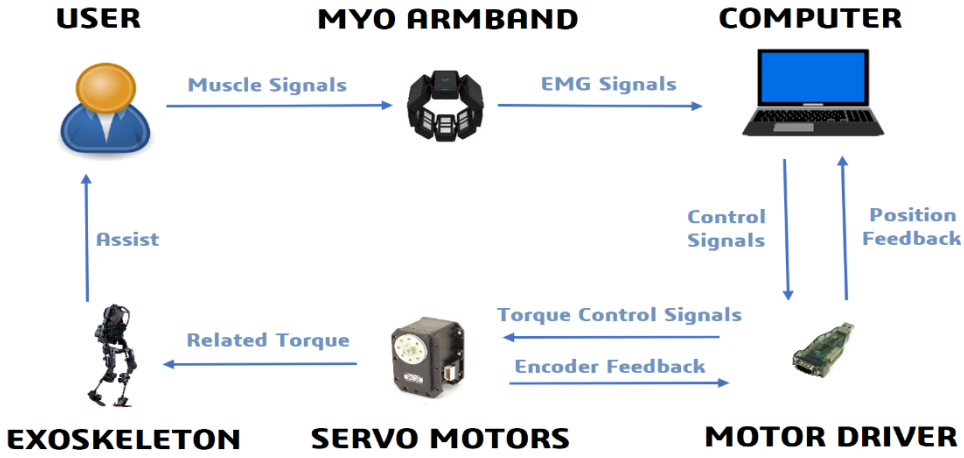


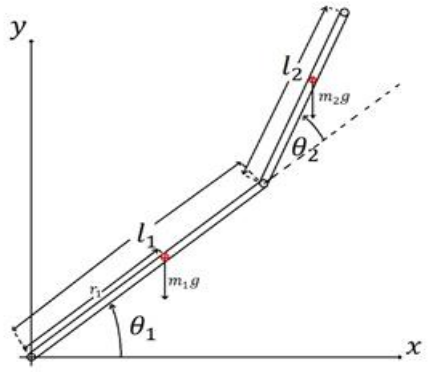
Figure 4.9 Overall System

Muscle signals are taken from the user's arm by myo armband, then they are sent to a computer where an artificial neural network and controller algorithm has been embedded. At this point, the system classifies (flexion, extension, fist and relaxation movements) the EMG signals taken and understands the intention of the user. Then related actuation signals are sent to the motor driver circuit which will drive the servo motors located in hip and knee joints. By the feedback mechanism, the rotary encoder provides instant angle-position data of the leg mechanism, so that leg mechanism can be controlled precisely. Finally, the leg mechanism

driven by motors helps the ongoing movement of the user by providing the required torque values.

4.3 Mathematical Modelling

In this section, firstly, the mathematical derivation of dynamic equations of the system will be explained. Note that, the rigid double pendulum model shown in Fig. 4.10 will be used to represent the leg mechanism. Later, the mathematical model which shows the behavior of the system will be developed. Then the MATLAB-Simulink model for the derived dynamic equations will be shown at the end of this part. After all, more realistic Simulink model will be given, which is created by transforming the CAD file of the system into the Simulink model by using Simscape libraries.



The closed-form equations of motion of a double pendulum are going to be derived using Lagrange's equations of motion [14]. Each link of the pendulum is treated as a rigid body, defined by its mass and the position of the center of mass with respect to its corresponding joint; for example, r_1 is the distance of the center of mass of link 1 with respect to joint 1.

Figure 4.10 Rigid Double Pendulum

First, the Cartesian coordinates (x_i, y_i) of each center of mass are calculated:

$$x_1 = r_1 \cos\theta_1 \tag{4.1}$$

$$x_2 = l_1 \cos\theta_1 + r_2 \cos(\theta_1 + \theta_2) \tag{4.2}$$

$$y_1 = r_1 \sin\theta_1 \tag{4.3}$$

$$y_2 = l_1 \sin\theta_1 + r_2 \sin(\theta_1 + \theta_2) \tag{4.4}$$

Then by taking the time derivative of the equations 4.1-4.4 the corresponding velocities of each center of mass with respect to the cartesian coordinate system can be determined:

$$\dot{x_1} = -r_1 . \dot{\theta_1} sin\theta_1 \tag{4.5}$$

$$\dot{x_2} = -l_1 . \dot{\theta_1} . sin\theta_1 - r_2 . (\dot{\theta_1} + \dot{\theta_2}) . sin(\dot{\theta_1} + \dot{\theta_2}) \tag{4.6}$$

$$\dot{y_1} = r_1 . \dot{\theta_1} . cos\theta_1 \tag{4.7}$$

$$\dot{y_2} = l_1.\dot{\theta_1}.cos\theta_1 + r_2.(\dot{\theta_1} + \dot{\theta_2}).cos(\dot{\theta_1} + \dot{\theta_2}) \quad (4.8)$$

Now squares of the center of mass velocities for each link will be calculated as shown below:

$$v_1{}^2 = \dot{x_1}{}^2 + \dot{y_1}{}^2 = r_1{}^2 + \dot{\theta_1}{}^2 \quad (4.9)$$

$$v_2{}^2 = \dot{x_2}{}^2 + \dot{y_2}{}^2 = l_1{}^2 + \dot{\theta_1}{}^2 + r_2{}^2 + (\dot{\theta_1} + \dot{\theta_2})^2 + 2.l_1.r_2\left(\dot{\theta_1}{}^2 + \dot{\theta_1}.\dot{\theta_2}\right).cos\theta_2 \quad (4.10)$$

Now let's calculate kinetic and potential energies of each link. Note that each link modeled as a homogeneous cylindrical bar and I_i represents the moment of inertia of link i about the center of mass, which can be calculated as (1/12)*m*l^2.

$$K_1 = \frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}m_1v_1^2 \quad (4.11)$$

$$K_2 = \frac{1}{2}I_2(\dot{\theta_1} + \dot{\theta_2})^2 + \frac{1}{2}m_2v_2^2 \quad (4.12)$$

$$P_1 = m_1gr_1sin\theta_1 \quad (4.13)$$

$$P_2 = m_2g(l_1sin\theta_1 + r_2\sin(\theta_1 + \theta_2)) \quad (4.14)$$

The Lagrangian, L can be calculated as shown below:

$$L = K - P = (K_1 + K_2) - (P_1 + P_2) \quad (4.15)$$

$$L = \frac{1}{2}(I_1\dot{\theta}_1^2 + m_1r_1^2\dot{\theta}_1^2) + \frac{1}{2}\left[I_2(\dot{\theta_1} + \dot{\theta_2})^2 + m_2\left(l_1^2\dot{\theta}_1^2 + r_2^2(\dot{\theta_1} + \dot{\theta_2})^2 + 2l_1r_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2)cos\theta_2\right)\right] - m_1gr_1sin\theta_1 - m_2g(l_1sin\theta_1 + r_2\,sin(\theta_1 + \theta_2)) \quad (4.16)$$

The terms needed for the Lagrange's Equation of Motion:

$$\frac{\partial L}{\partial\dot{\theta_1}} = I_1\dot{\theta}_1 + m_1r_1^2\dot{\theta}_1 + I_2(\dot{\theta_1} + \dot{\theta_2}) + m_2\left[l_1^2\dot{\theta}_1 + l_1r_2cos\theta_2(2\dot{\theta_1} + \dot{\theta_2}) + r_2^2(\dot{\theta_1} + \dot{\theta_2})\right] \quad (4.17)$$

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\theta_1}} = I_1\ddot{\theta}_1 + m_1r_1^2\ddot{\theta}_1 + I_2(\ddot{\theta_1} + \ddot{\theta_2}) + m_2l_1^2\ddot{\theta}_1 + m_2r_2^2(\ddot{\theta_1} + \ddot{\theta_2}) + m_2l_1r_2\left[(2\ddot{\theta}_1 + \ddot{\theta}_2)cos\theta_2 - \sin(\theta_2)\dot{\theta}_2(2\dot{\theta_1} + \dot{\theta_2})\right] \quad (4.18)$$

$$\frac{\partial L}{\partial\theta_1} = -m_1gr_1cos\theta_1 - m_2gl_1\,cos\theta_1 - m_2gr_2\,cos(\theta_1 + \theta_2)) \quad (4.19)$$

$$\frac{\partial L}{\partial\dot{\theta_2}} = I_2(\dot{\theta_1} + \dot{\theta_2}) + m_2r_2^2(\dot{\theta_1} + \dot{\theta_2}) + m_2l_1r_2cos(\theta_2)\dot{\theta_1} \quad (4.20)$$

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\theta_2}} = I_2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2r_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2l_1r_2\cos(\theta_2)\ddot{\theta}_1 - m_2l_1r_2sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 \quad (4.21)$$

$$\frac{\partial L}{\partial\theta_2} = -m_2l_1r_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2)sin\theta_2 - m_2gr_2\cos(\theta_1 + \theta_2) \quad (4.22)$$

Now Lagrange's Equation of Motion will be obtained. Note that, q is an n-vector of

generalized coordinates q_i, L is the Lagrangian, Yi equals to the generalized forces applied on each joint. Since the joints of the double pendulum are rotational the generalized forces refer to torques. Additionally, a damping term can be applied (bi the damping coefficient of joint i):

$$Y_i = \tau_i - b_i\dot{\theta}_i \tag{4.23}$$

$$Y_1 = [\, I_1 + I_2 + m_1r_1^2 + m_2l_1^2 + m_2r_2^2 + 2m_2l_1r_2cos\theta_2]\ \ddot{\theta}_1 + [I_2 + m_2r_2^2 + m_2l_1r_2cos\theta_2]\ \ddot{\theta}_2 - m_2l_1r_2\sin(\theta_2)\,\dot{\theta}_2\left(2\dot{\theta}_1 + \dot{\theta}_2\right) + m_1gr_1cos\theta_1 + m_2gl_1\,cos\theta_1 + m_2gr_2\,cos(\theta_1 + \theta_2) \tag{4.24}$$

$$Y_2 = [\, I_2 + m_2r_2^2 + m_2l_1r_2\cos(\theta_2)\,]\ \ddot{\theta}_1 + [I_2 + m_2r_2^2]\ \ddot{\theta}_2 - m_2l_1r_2sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 + m_2l_1r_2\left(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2\right)sin\theta_2 + m_2gr_2\cos(\theta_1 + \theta_2) \tag{4.25}$$

Now let's obtain the equations of motion in the following vector form. Note that, M(q) is inertia matrix, V(q,q̇) is Coriolis/centripetal vector, and G(q) is gravity vector.

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \mathrm{M(q)}\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix} + \mathrm{V(q,\dot{q})} + \mathrm{G(q)} \tag{4.26}$$

$$\mathrm{M(q)} = \begin{bmatrix} I_1 + I_2 + m_1r_1^2 + m_2l_1^2 + m_2r_2^2 + 2m_2l_1r_2cos\theta_2 & I_2 + m_2r_2^2 + m_2l_1r_2cos\theta_2 \\ I_2 + m_2r_2^2 + m_2l_1r_2\cos(\theta_2) & I_2 + m_2r_2^2 \end{bmatrix} \tag{4.27}$$

$$\mathrm{V(q,\dot{q})} = \begin{bmatrix} -m_2l_1r_2\sin(\theta_2)\,\dot{\theta}_2\left(2\dot{\theta}_1 + \dot{\theta}_2\right) + b_1\dot{\theta}_1 \\ m_2l_1r_2sin\theta_2\dot{\theta}_1^2 + b_2\dot{\theta}_2 \end{bmatrix} \tag{4.28}$$

$$\mathrm{G(q)} = \begin{bmatrix} m_1gr_1cos\theta_1 + m_2gl_1\,cos\theta_1 + m_2gr_2\,cos(\theta_1 + \theta_2) \\ m_2gr_2\cos(\theta_1 + \theta_2) \end{bmatrix} \tag{4.29}$$

The same equation in a simplified form can be obtained. To do that, let's say:

$$\alpha = I_1 + I_2 + m_1r_1^2 + m_2l_1^2 + m_2r_2^2 \tag{4.30}$$

$$\beta = m_2l_1r_2 \tag{4.31}$$

$$\delta = I_2 + m_2r_2^2 \tag{4.32}$$

$$c_i = cos\theta_i,\ \ s_i = sin\theta_i,\ \ c_{ij} = \cos(\theta_i+\theta_j) \tag{4.33}$$

Then,

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix}\alpha + 2\beta c_2 & \delta + \beta c_2\\ \delta + \beta c_2 & \delta\end{bmatrix}\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix} + \begin{bmatrix}-\beta s_2\dot{\theta}_2 + b_1 & -\beta s_2(\dot{\theta}_1 + \dot{\theta}_2)\\ \beta s_2\dot{\theta}_1 & b_2\end{bmatrix}\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{bmatrix} + \begin{bmatrix} m_1gr_1c_1 + m_2g(l_1c_1 + r_2c_{12}) \\ m_2gr_2c_{12}\end{bmatrix} \tag{4.34}$$

4.4 MATLAB Mechanism Model (Based on Equations of Motions)

In Fig. 4.11, the MATLAB-Simulink model based on the equations of motion derived under the previous title is shown:

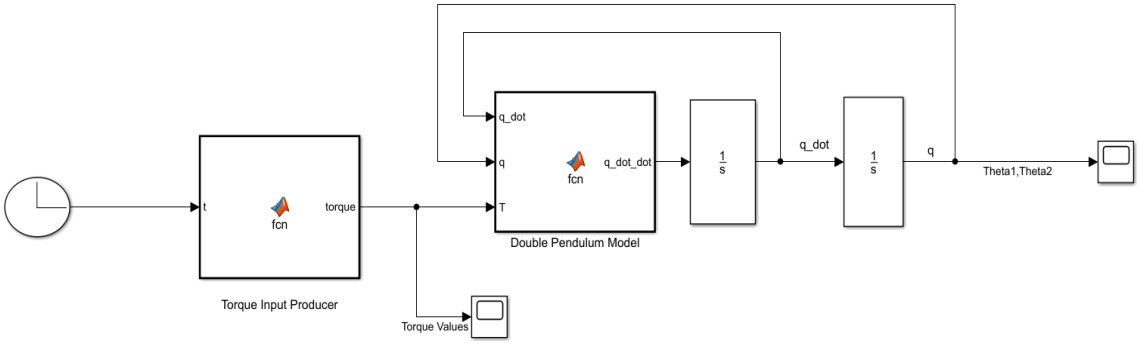


Figure 4.11 MATLAB Model Based On Equations

The equations of motion have been embedded inside the MATLAB function block called Double Pendulum Model. Script code inside the Double Pendulum Model can be seen below:

```
function q_dot_dot = fcn(q_dot,q,T)

g=9,80665;          % m/s²
m1=1.39355;         % kg
m2= 0.96819;        %kg
r1=0.23678;         %m                  COM
r2=0.19350;         %m                  COM
l1=0.48373;         %m                  Total length
l2= 0.38927;        %m                  Total length
I1=0.07540282163;   %kg.m^2             Inertia
I2= 0.0294251582;   %kg.m^2             Inertia

Q1=q(1);            %Radians            Hip joint angle
Q2=q(2);            %Radians            Knee joint angle
Q1_DOT=q_dot(1);    %Radians/s          Angular velocity
Q2_DOT=q_dot(2);    %Radians/s          Angular velocity
b1=0.01;            %N.m.s/radian       Damping coefficient
b2=0.01;            %N.m.s/radian       Damping coefficient


alfa=I1+I2+m1*r1^2+ m2*(l1^2+r2^2);
beta=m2*l1*r2;
theta=I2+m2*r2^2;

M=[alfa+2*beta*cos(Q2),  theta+beta*cos(Q2);
```

```
    theta+beta*cos(Q2),                theta;]

V=[(-beta*sin(Q2)*Q2_DOT+b1)*Q1_DOT -
(beta*sin(Q2)*(Q1_DOT+Q2_DOT))*Q2_DOT;
    (beta*sin(Q2)*Q1_DOT*Q1_DOT+b2*Q2_DOT)]

G=[m1*g*r1*cos(Q1)+m2*g*(l1*cos(Q1)+r2*cos(Q1+Q2));
   m2*g*r2*cos(Q1+Q2)]

q_dot_dot = inv(M)*(T-V-G);

end
```

The Torque Input Producer shown in Fig. 4.11 is another MATLAB function which takes time as an input and provides a torque vector into the system. System behavior can be seen under different torque input conditions. It will be useful to test if a meaningful system has been obtained or is there any calculation errors. For example, 0 torque input can be provided to the system which means θ_1 value should go -$\pi/2$ radians and θ_2 value should go 0 radians considering Fig. 4.10.

4.5 Controller Design

In Fig. 4.12 a MATLAB function block that represents controller built by benefiting computed torque method is shown. The theory behind the computed torque control has been already discussed in section 3.1.

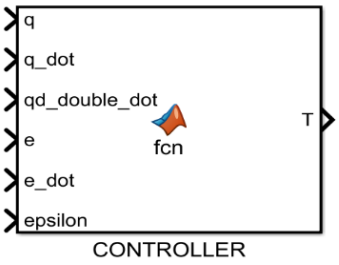


Figure 4.12 MATLAB Controller Block

Figure 4.13 will help to understand how this controller block works:

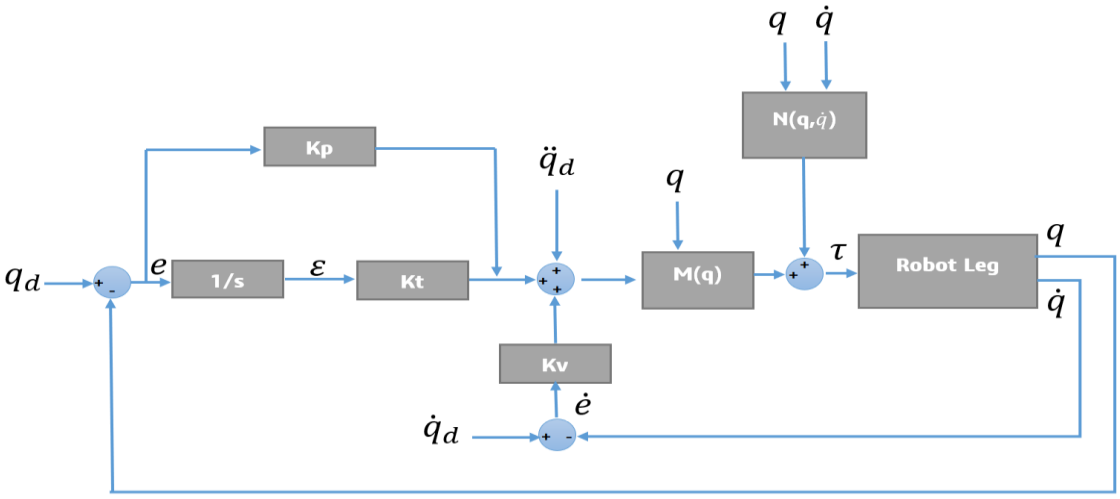


Figure 4.13 PID Computed Torque Controller

The auxiliary control signal u(t) has been selected as:

$$u = -K_v\dot{e} - K_p e - K_i\varepsilon_i \tag{4.40}$$

$$\dot{\varepsilon} = e \tag{4.41}$$

Which yields the leg control input:

$$\tau = M(q)(\ddot{q}_d + u) + N(q,\dot{q}) \tag{4.42}$$

Equation 4.42 is going to be used in order to build the MATLAB controller model. Here is the script code has been used inside the controller block diagram:

```
g=9,80665;          % m/s²
m1=1.39355;         % kg
m2= 0.96819;        %kg
r1=0.23678;         %m                  COM
r2=0.19350;         %m                  COM
l1=0.48373;         %m                  Total length
l2= 0.38927;        %m                  Total length
I1=0.07540282163;   %kg.m^2             Inertia
I2= 0.0294251582;   %kg.m^2             Inertia

Q1=q(1);            %Radians            Hip joint angle
Q2=q(2);            %Radians            Knee joint angle
Q1_DOT=q_dot(1);    %Radians/s          Angular velocity
Q2_DOT=q_dot(2);    %Radians/s          Angular velocity
b1=0;                % Damping coefficient
b2=0;               % Damping coefficient
```

```
Kv = 100;
Kp = 100;
Ki = 1;

alfa=I1+I2+m1*r1^2+ m2*(l1^2+r2^2);
beta=m2*l1*r2;
theta=I2+m2*r2^2;

M=[alfa+2*beta*cos(Q2),  theta+beta*cos(Q2);
    theta+beta*cos(Q2),               theta;]

V=[(-beta*sin(Q2)*Q2_DOT+b1)*Q1_DOT -
(beta*sin(Q2)*(Q1_DOT+Q2_DOT))*Q2_DOT;
    (beta*sin(Q2)*Q1_DOT*Q1_DOT+b2*Q2_DOT)]

G=[m1*g*r1*cos(Q1)+m2*g*(l1*cos(Q1)+r2*cos(Q1+Q2));
   m2*g*r2*cos(Q1+Q2)]

N=V+G;

s= qd_double_dot + Kv.*e_dot + Kp.*e + Ki.*epsilon;

T =M*s+N;     % Torque vector  provided by the controller

end
```

The PID coefficients have been selected so that, error results can be minimized, and sufficient speed of response can be obtained for the system. The values of Kv, Kd and Kt have been optimized to obtain better system behavior. Note that, a very important point about computed torque control is that, the system has to be modelled very precisely, otherwise it is not going to work.

4.6 Entire System Model (Both Mathematical and Simscape)

The system which is shown in Fig. 4.14 consists of the controller and the leg mechanism. The desired trajectory for the leg mechanism has been provided by using an actuation block. Note that, this system in Fig. 4.14 built by using the equations of motions derived before. On the other hand, another entire system model in Fig. 4.15 has been built by transforming CAD design into a Simscape model. Both mathematical and Simscape simulation results are examined in the next sections.

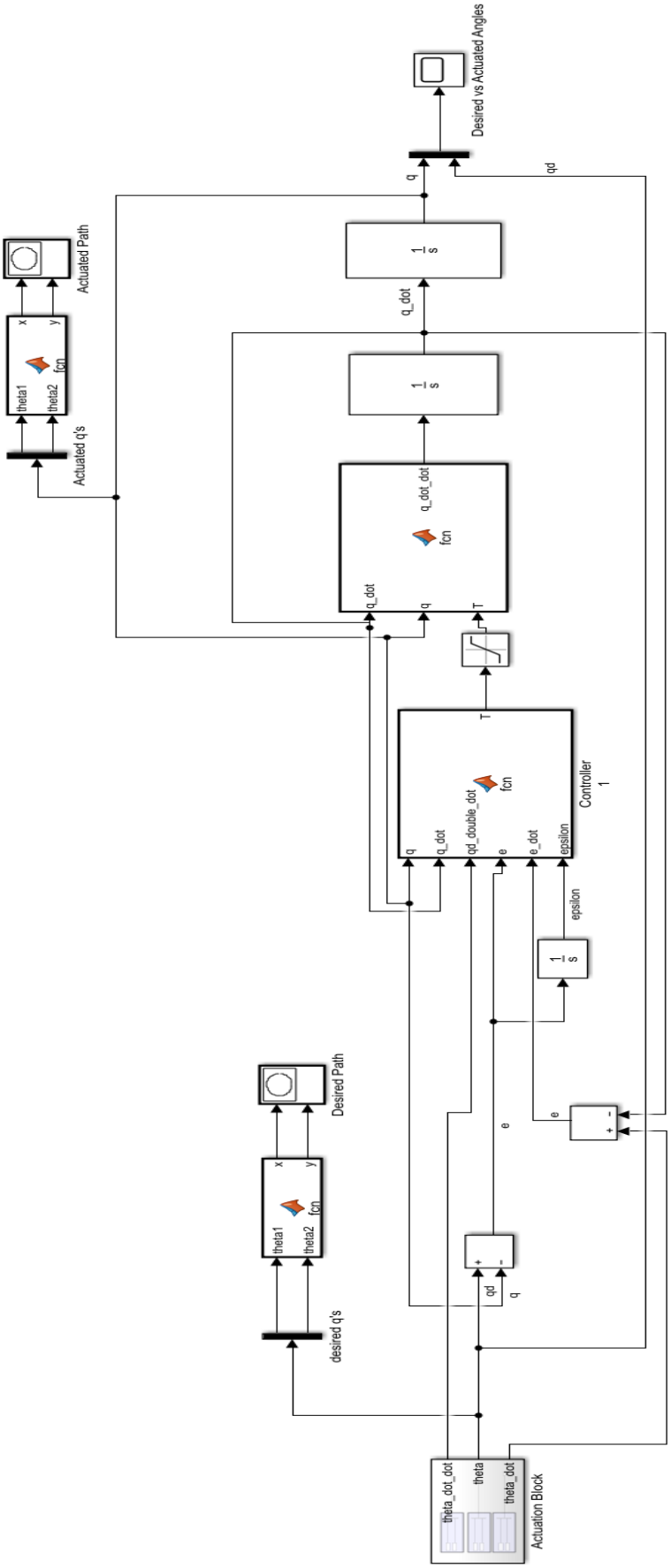


Figure 4.14 Entire Mathematical System Model

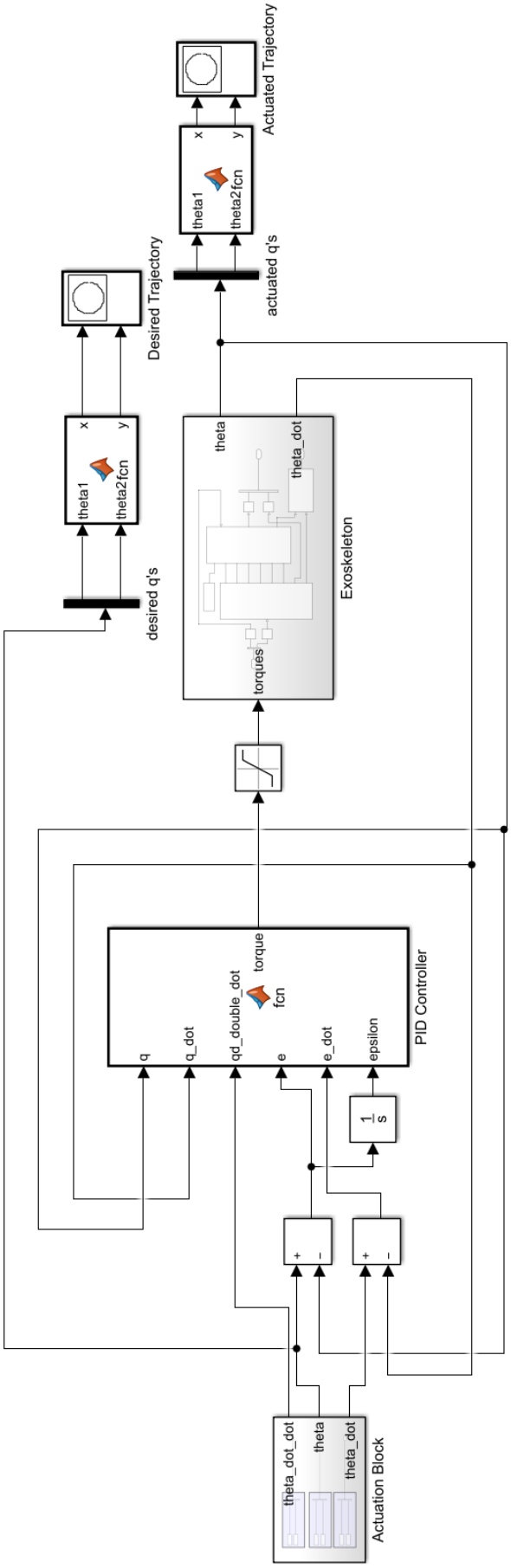


Figure 4.15 Entire Simscape Model

5. DESIGN VERIFICATIONS

5.1 CAD Design Verifications

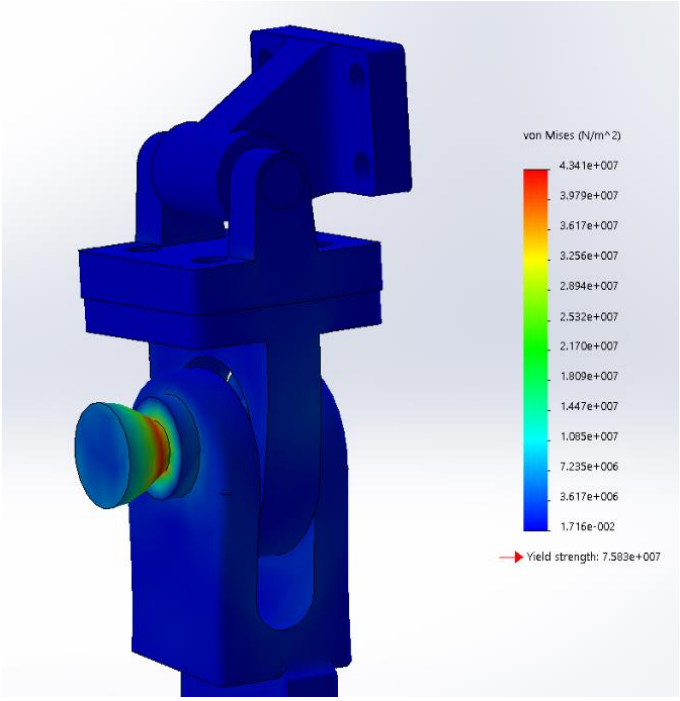


Figure 5.1 Stress Analysis of the Hip

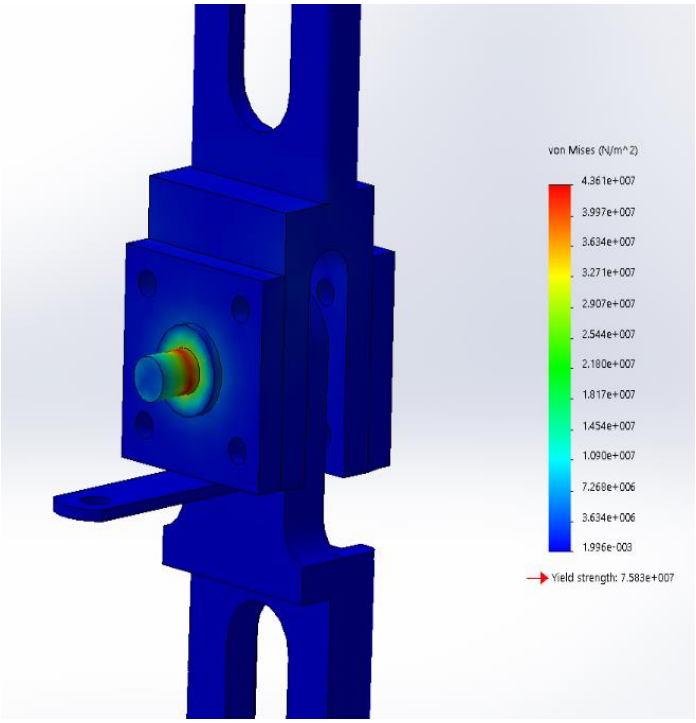


Figure 5.2 Stress Analysis of the Knee

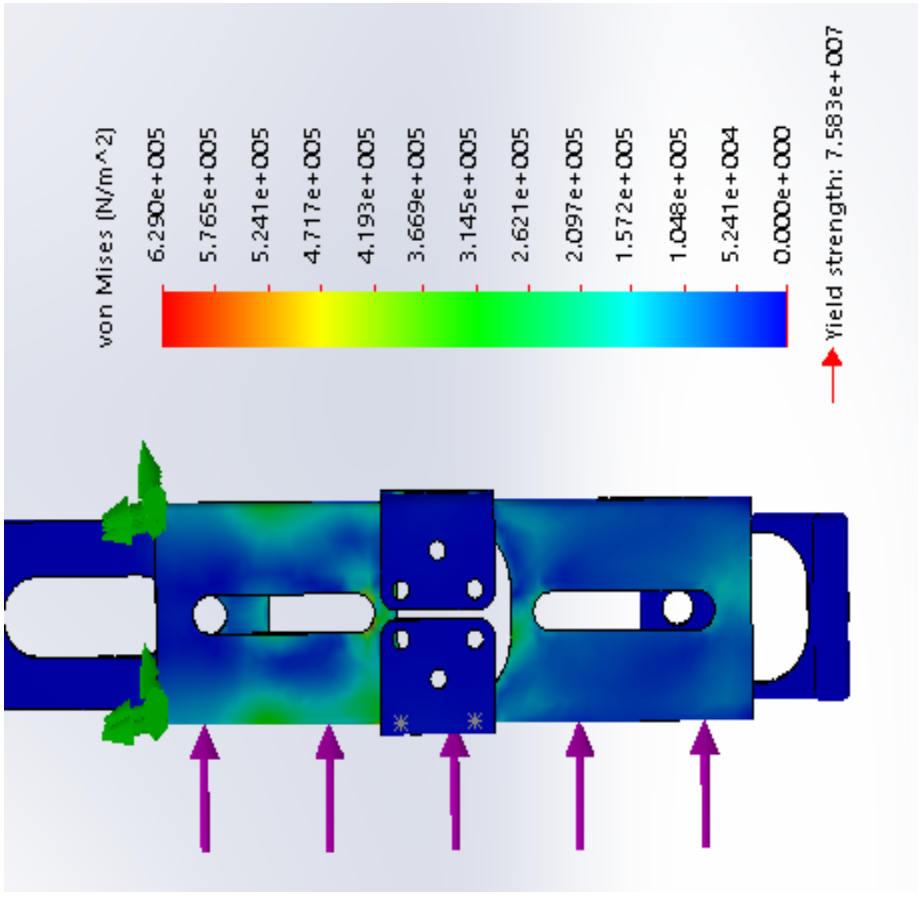


Figure 5.3 Stress Analysis of the Outer Link

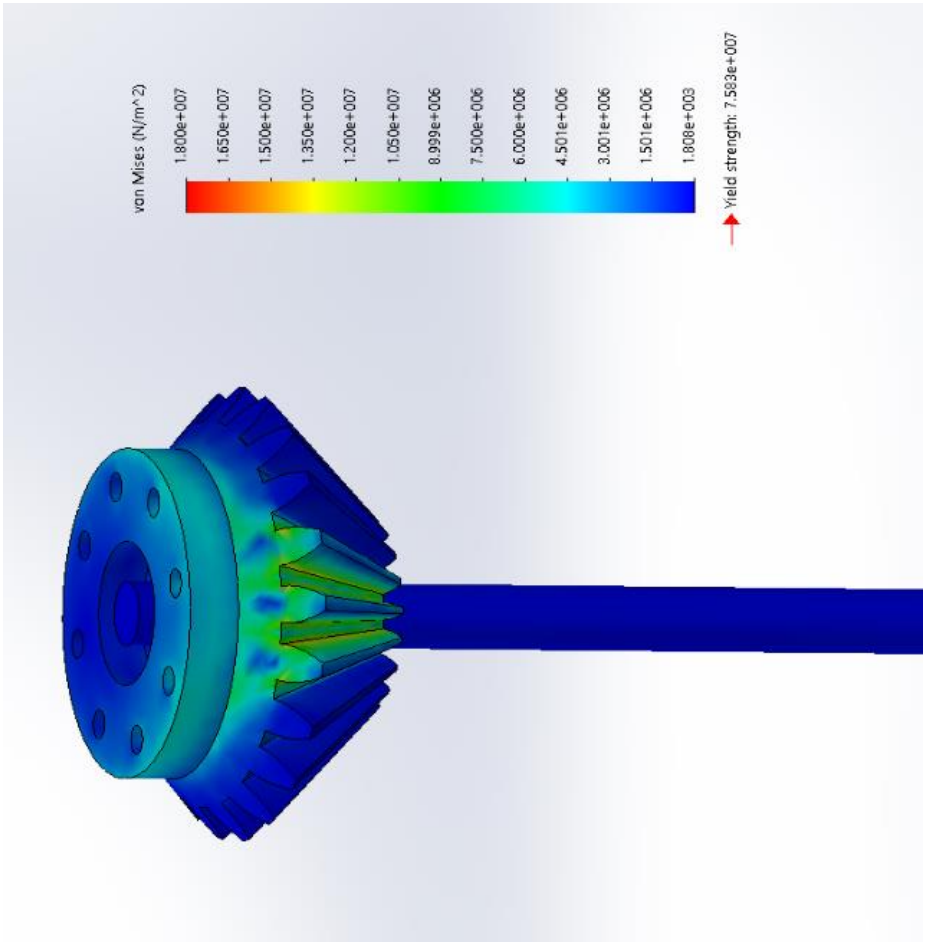


Figure 5.4 Stress Analysis Reductor Gear

5.2 Mathematical System Model Verifications

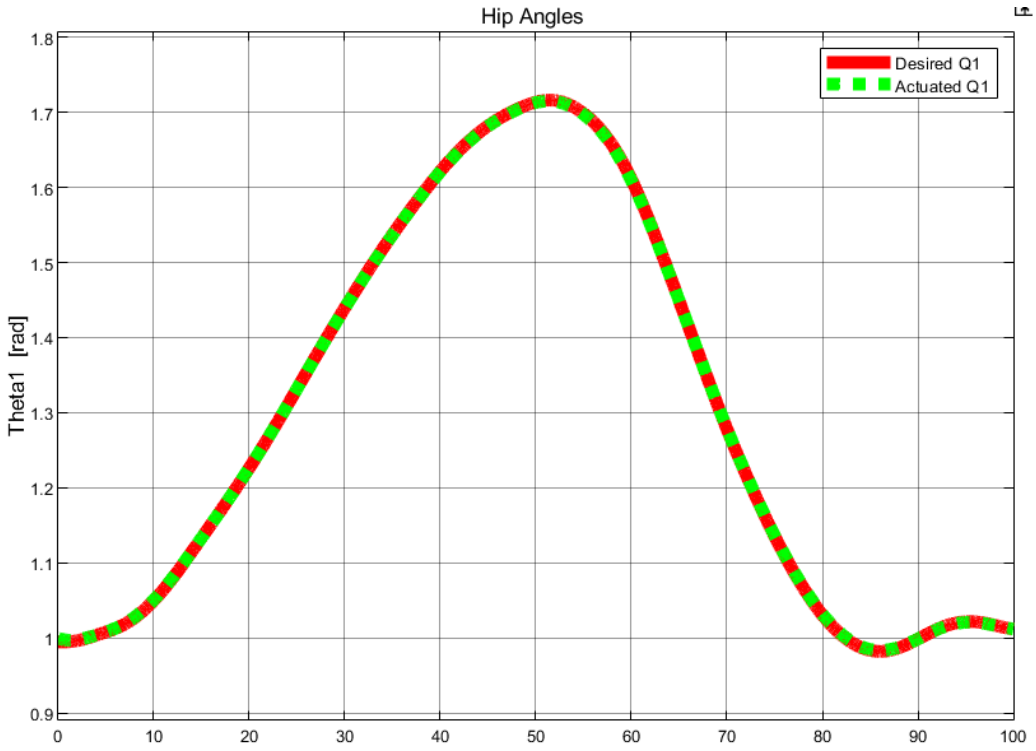


Figure 5.4 Desired vs Actuated Hip Joint Angles

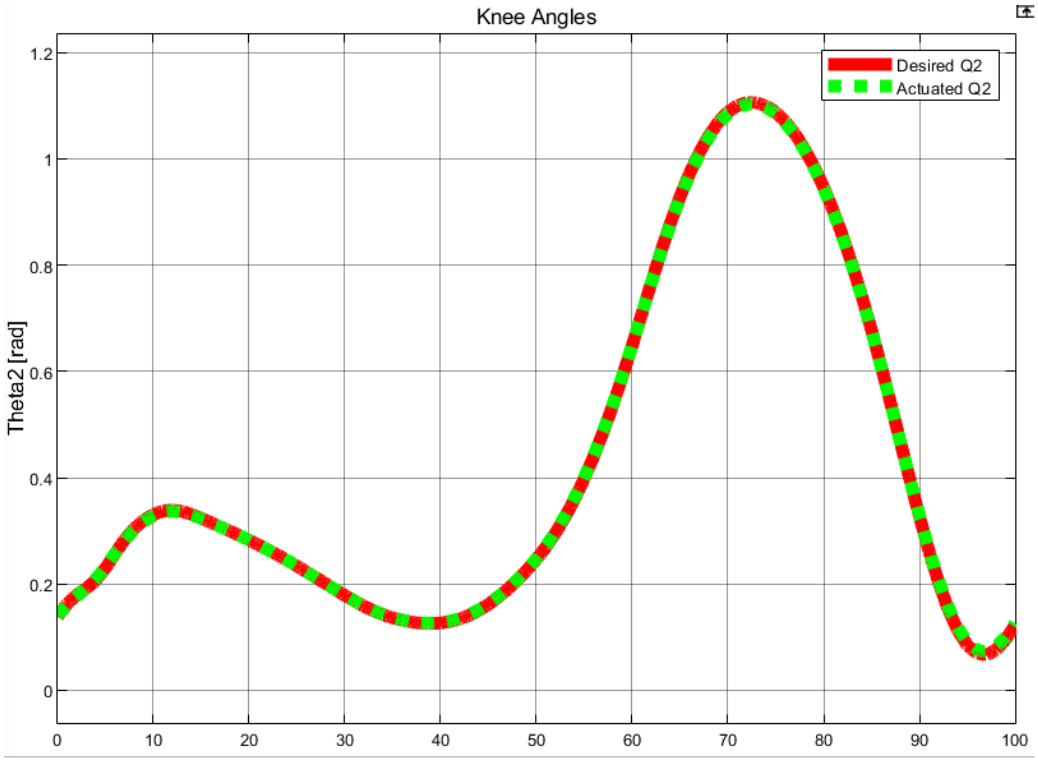


Figure 5.5 Desired vs Actuated Knee Joint Angles

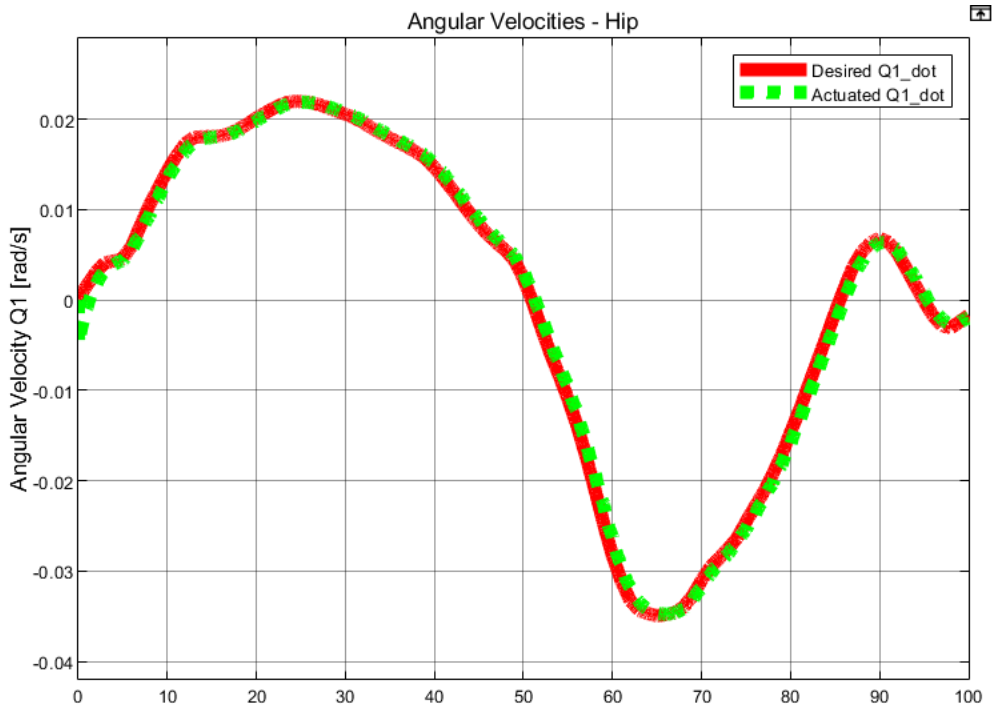


Figure 5.6 Desired vs Actuated Hip Joint Angular Velocities

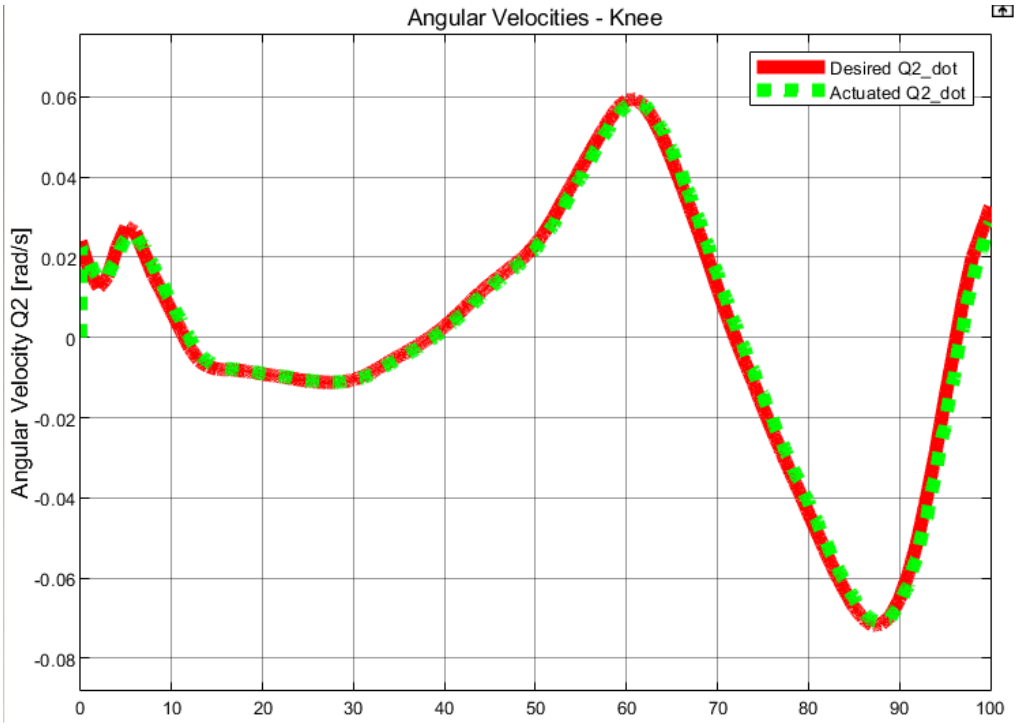


Figure 5.7 Desired vs Actuated Knee Joint Angular Velocities

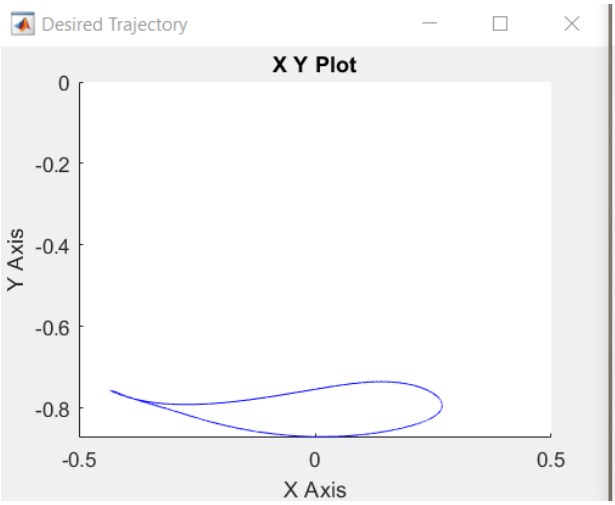


Figure 5.8 Desired Path

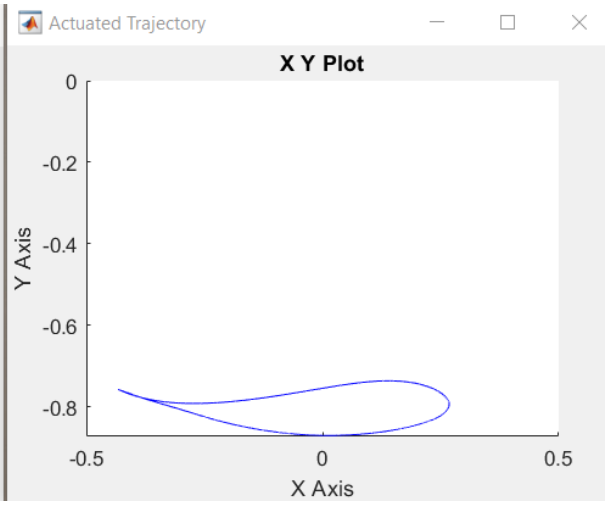


Figure 5.9 Actuated Path

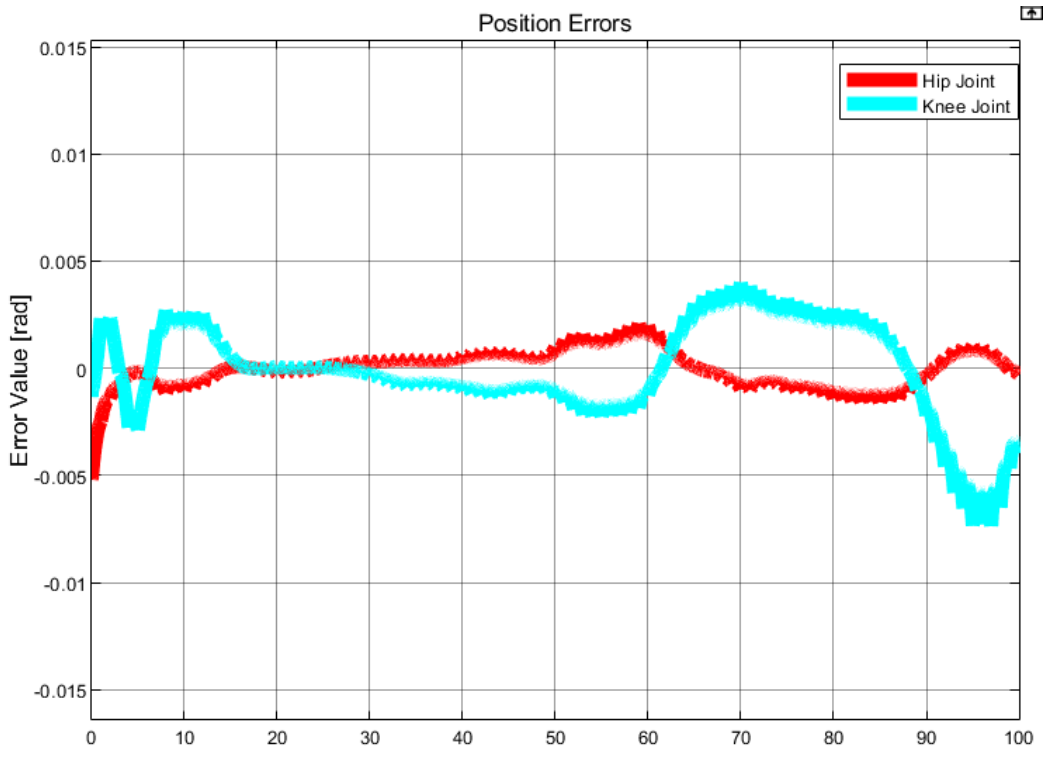


Figure 5.10 Position Tracking Errors

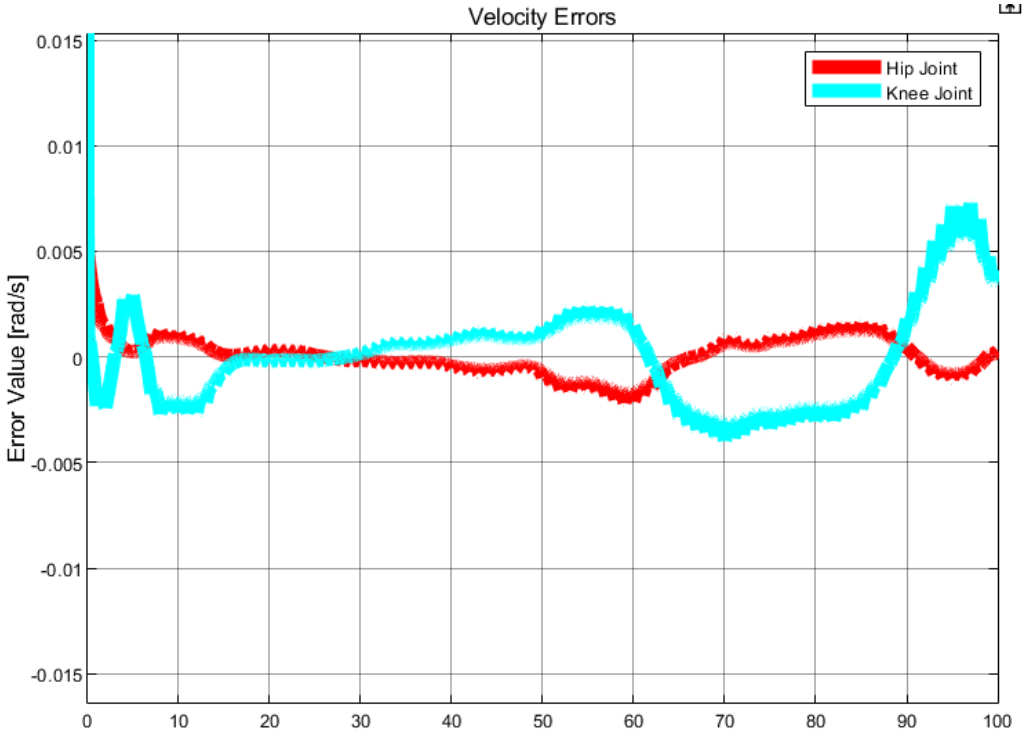


Figure 5.11 Velocity Tracking Errors

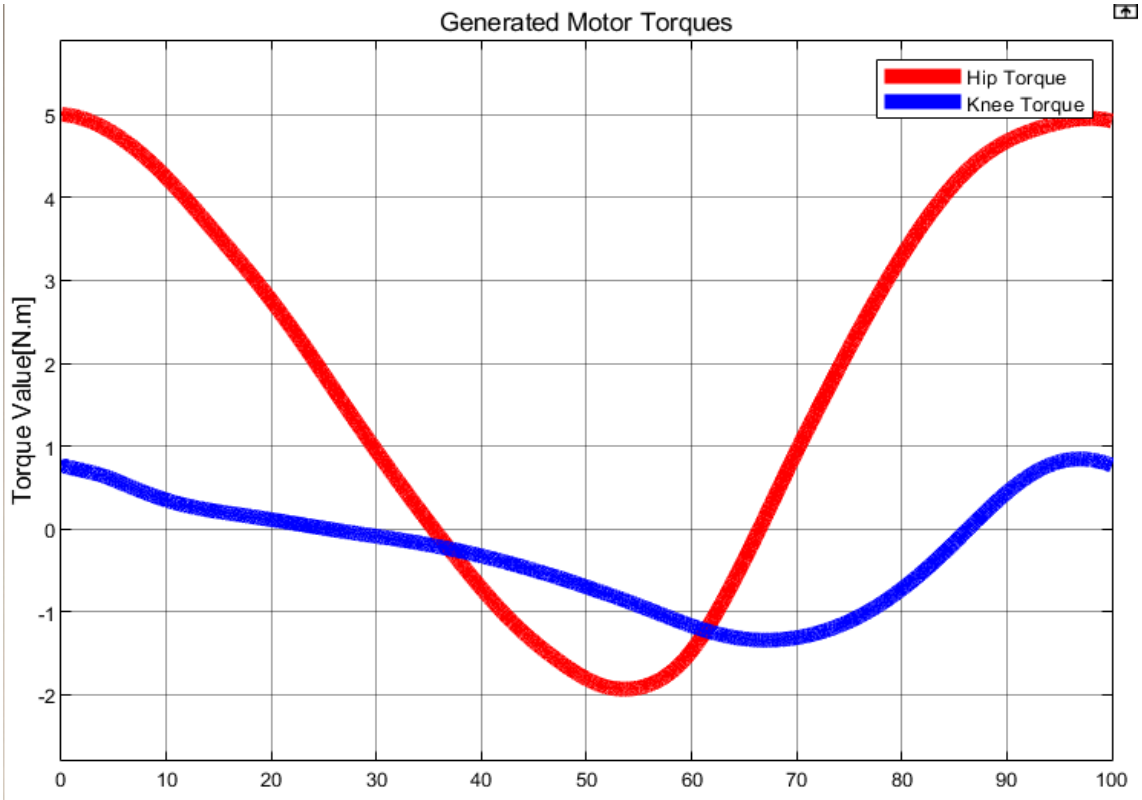


Figure 5.12 Generated Torque Values

5.3 Simscape System Model Verifications

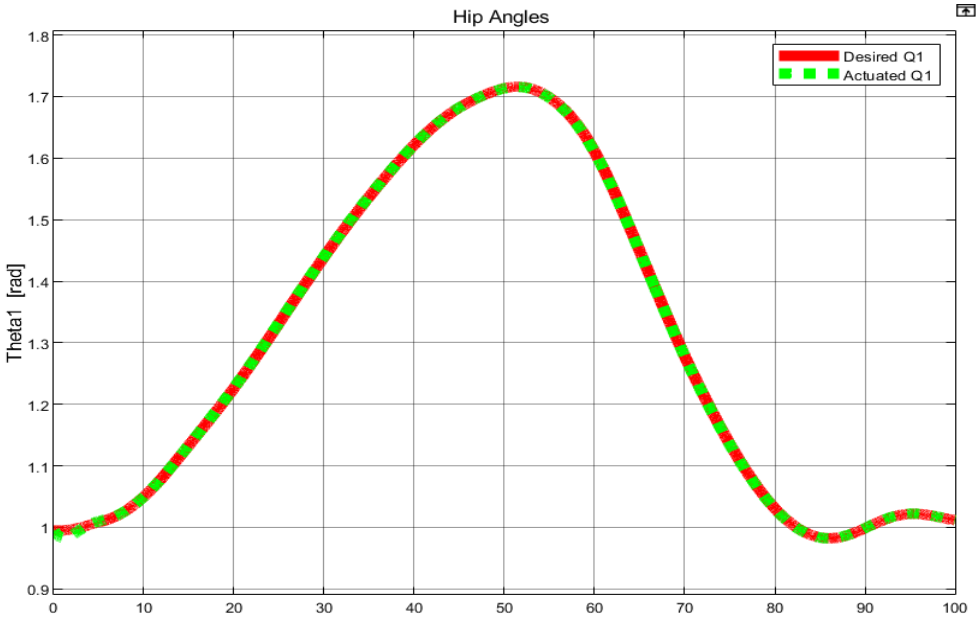


Figure 5.13 Desired vs Actuated Hip Joint Angles

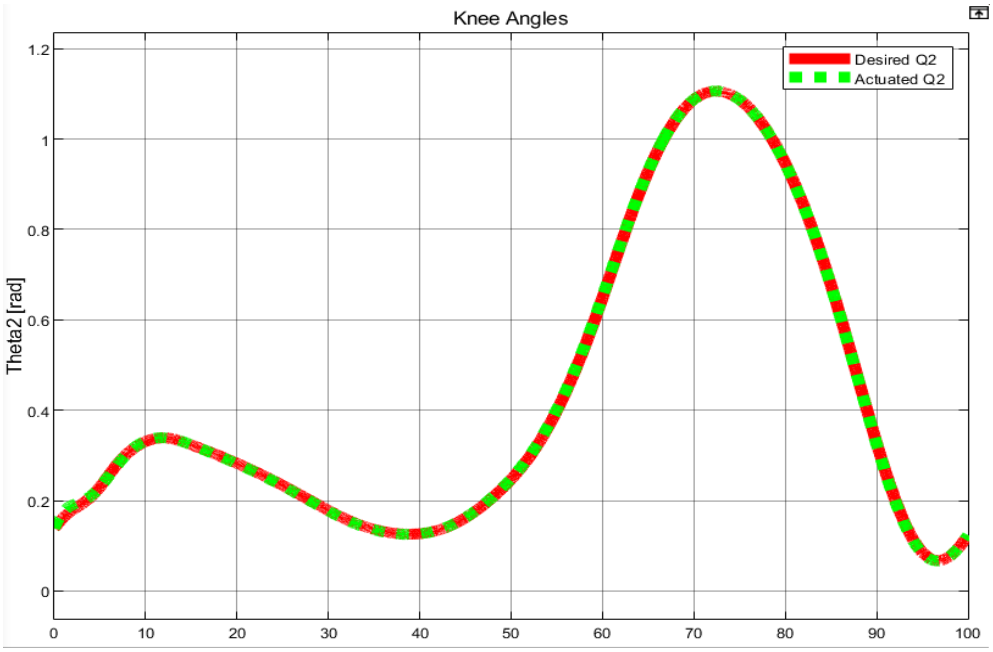


Figure 5.14 Desired vs Actuated Knee Joint Angles

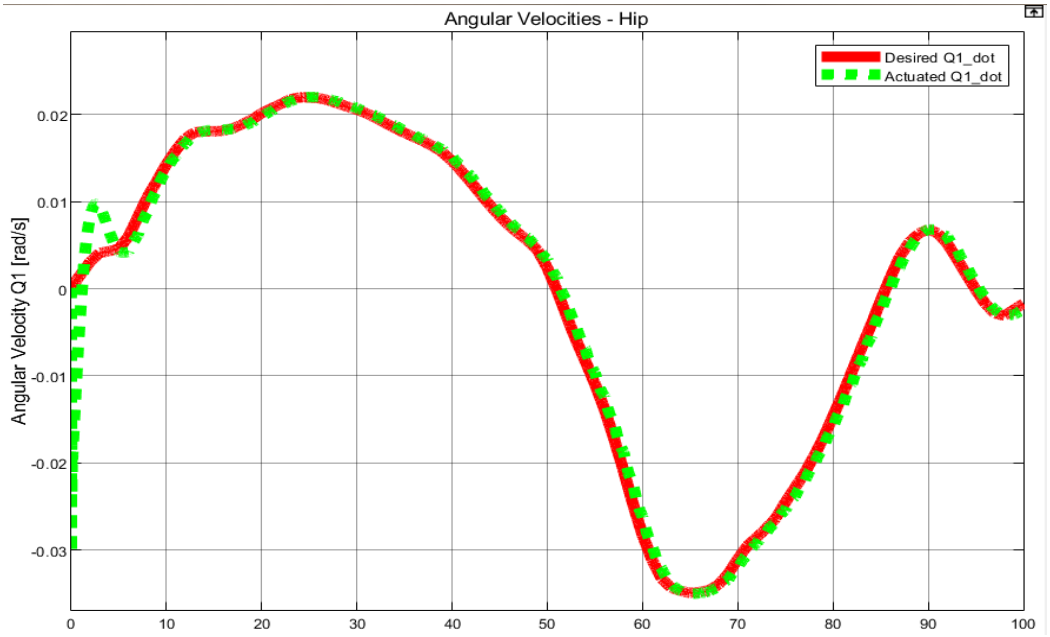


Figure 5.15 Desired vs Actuated Hip Joint Angular Velocities

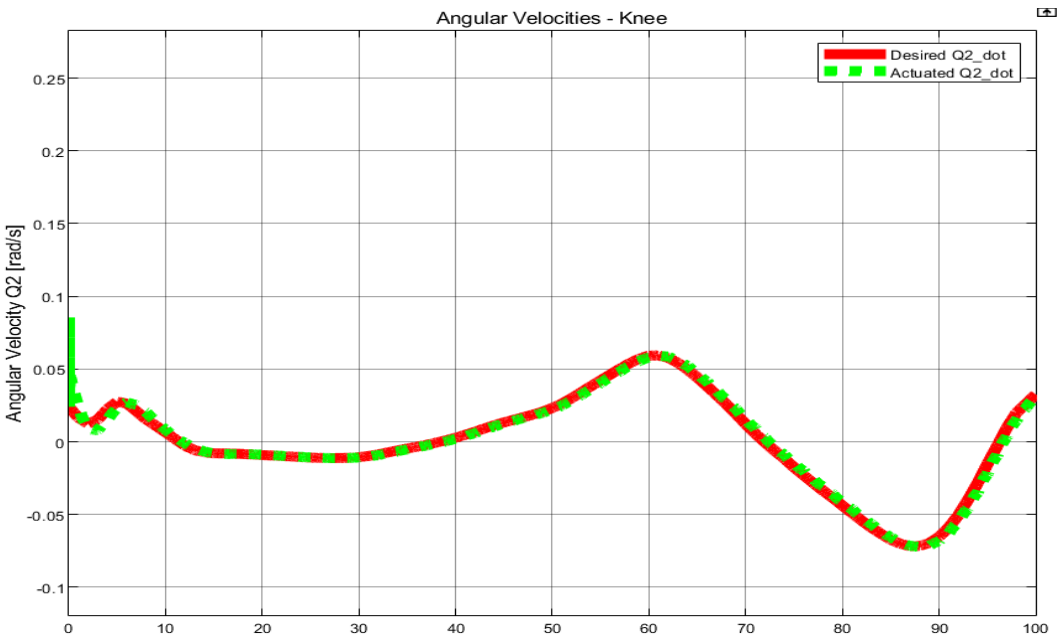


Figure 5.16 Desired vs Actuated Knee Joint Angular Velocities

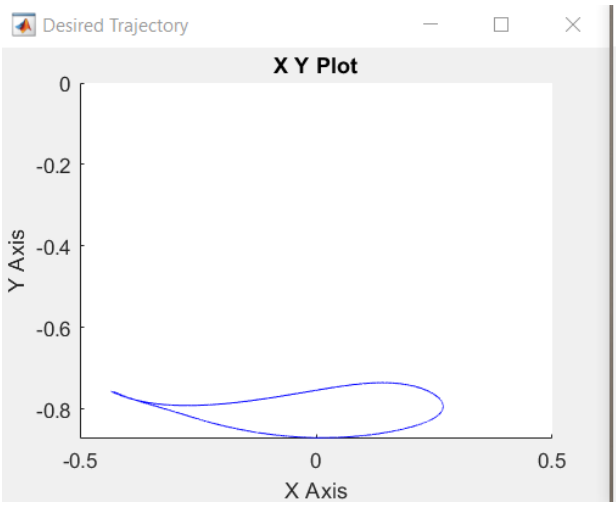


Figure 5.17 Desired Path

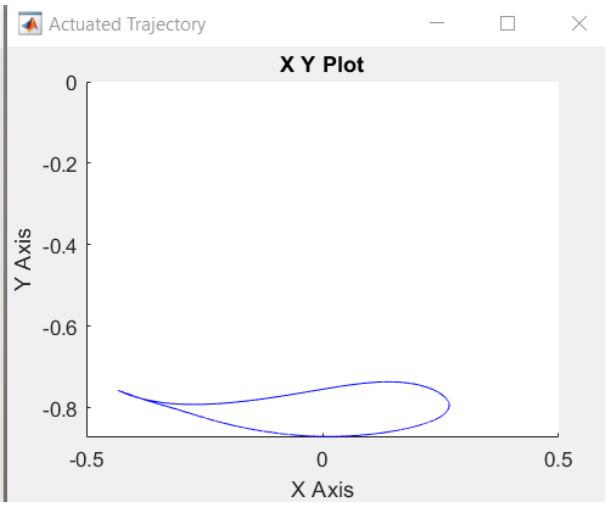


Figure 5.18 Actuated Path

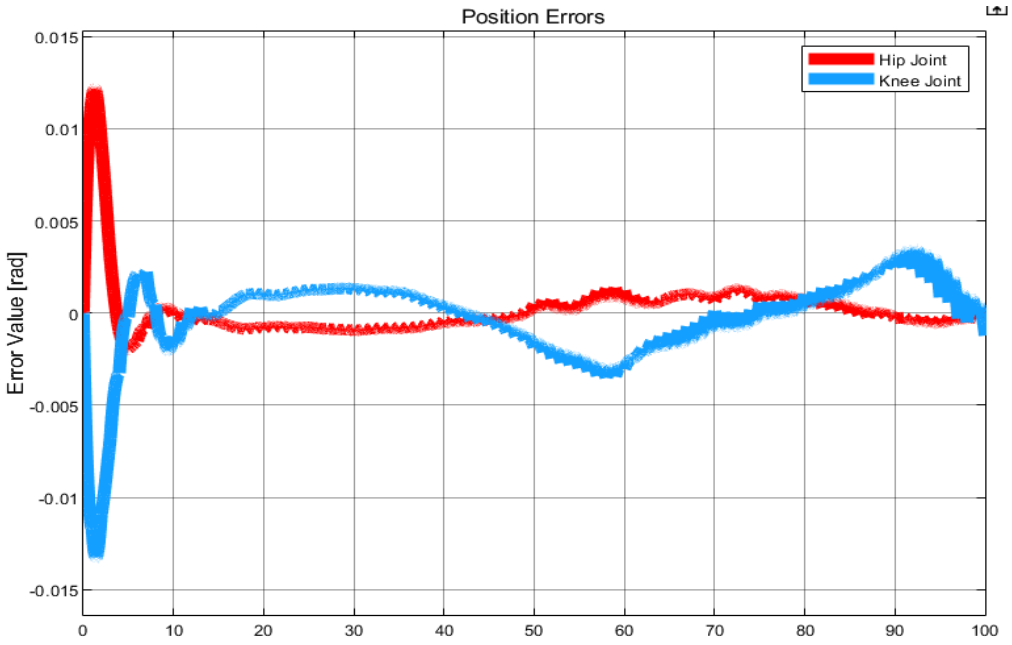


Figure 5.19 Position Tracking Errors

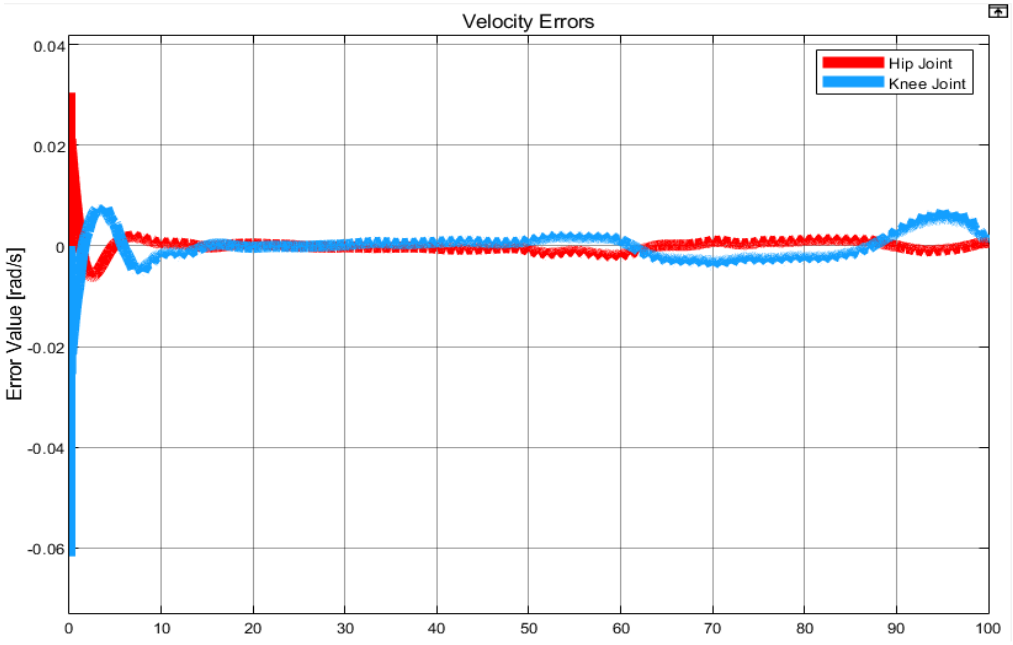


Figure 5.20 Velocity Tracking Errors

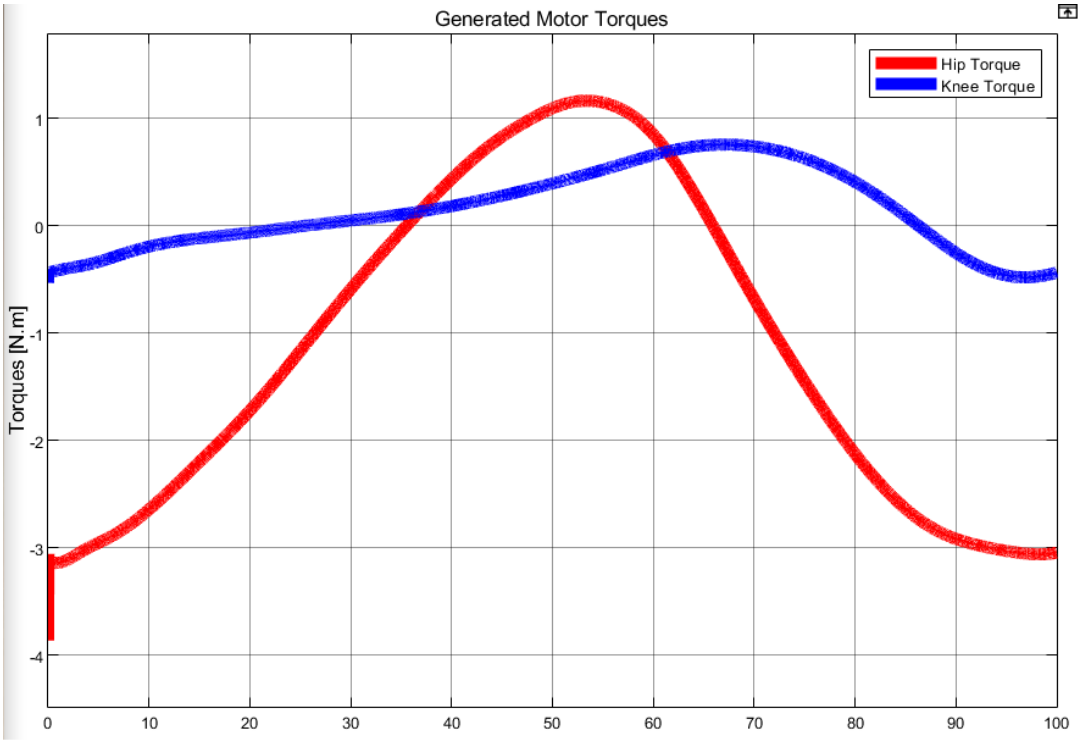


Figure 5.21 Generated Torque Values

6. DESIGN IMPLEMENTATION

6.1 Mechanical Implementation

The body material has been cast from ETIAL 160 as blocks in the shape of the rectangular prism. Then they have been reformed via CNC milling and drilling. Manufactured body parts weighted more than the expected. Thus, CNC milling and drilling operations are applied once more the lighten the parts with an engineering approach. Finally, the total system's weight decreased by 100% to 2.5 kg. The entire mechanical system is shown in Fig. 6.1 and Fig. 6.2.

Figure 6.1-a Entire Exoskeleton System Demo 1

Figure 6.1-b Entire Exoskeleton System Demo 2

Figure 6.2 Close-up Photo of the gear set

6.2 Electronic Implementation

Electronic circuitries are the essence of the system in order to move the mechanical parts according to the given input. Electronic components play a key role in actuating the system in combination with the software. Under this title, firstly, basic electronic components of the system will be introduced, then the whole electronic system will be illustrated with the connections between them.

6.2.1 Myo Armband

Myo armband has 8 stainless steel EMG sensors on it. Myo armband sends the EMG signals with 8-bit resolution with 200 Hz sampling frequency via Bluetooth. In addition to these, myo armband has 3 axis gyroscope and 3 axis accelerometer which are working with 50 Hz sampling frequencies. The part members of the Myo Armband are shown in Fig 6.3.

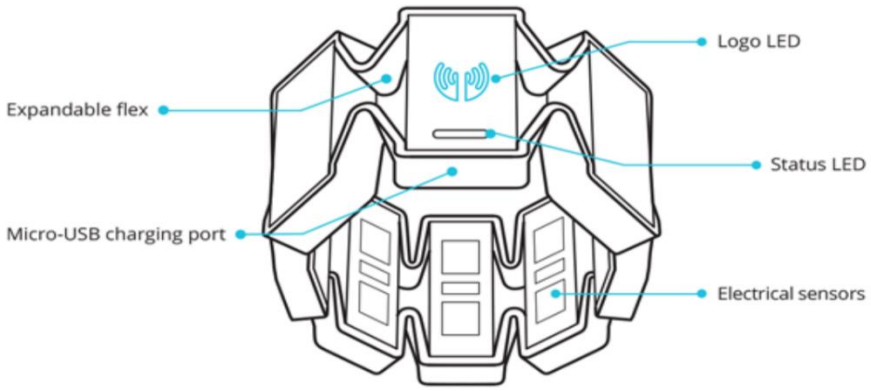


Figure 6.3 Myo Armband Parts

6.2.2 Dynamixel MX106-T Servo Motors

Choosing the system's motors might be the key factor in the Electronic Design of the project. As it is the part that gives the actuation to the system, it will not be wrong to say that it is the central component of the entire mechanism.

And within the motor selection, the main criteria are the torque values, later on, followed by weight, dimensions, actuation of the motor and the communication of the motor, respectively. The motors are chosen according to torque analysis explained in the Simulation Sections. Robotis Dynamixel MX-106T Smart Servo Motor shown in Fig 6.4 is agreed upon as it covers the needed torque of the system along with other advantages. Basic system parameters for the motor is shown in Table 6.1 below.

Figure 6.4 Dynamixel MX106-T Servo Motor

Model Name		Dynamixel MX-106T
MCU		Cortex-M3 (72 MHz, 32 bit)
Dimensions (WxHxD) [mm]		40.2 X 65.1 X 46.0
Weight [g]		153.00
Input Voltage	**Min. [V]**	10.0
	Recommended [V]	12.0
	Max. [V]	14.8
Performance Characteristics	**Voltage [V]**	12.0
	Stall Torque [N.m]	8.40
	Stall Current [A]	5.2
	No Load Speed [rpm]	45.0
	No Load Current [A]	0.17
Resolution	**Resolution [deg/pulse]**	0.0879
	Step [pulse]	4,096

Table 6.1 Dynamixel MX106-T Parameters

6.2.3 Usb2Dynamixel Motor Controller

USB2Dynamixel is a device used to operate Dynamixel directly from PC. USB2Dynamixel is connected to the USB port of PC, and 3P and 4P connectors have been installed so that various Dynamixels can be connected. Also, USB2Dynamixel can be used to change from USB port to Serial port on the PC without serial port such as in notebook computer, etc.

Figure 6.5 Usb2Dynamixel Motor Controller

6.2.4 Power Cell

The powering of the system has been handled with a 12V, 4A adaptor, which converts AC to DC. Note that for future works, the mechanism will be mobilized and will require mobile power components. These power components determined as Li-Po Batteries to benefit from their advantages. Li-Po Batteries has 4 times energy density, more resistance to physical effects, lightweight and flexible than normal batteries. Also, they can be produced in any shape or size, which grants the Li-Po batteries the wide range of application areas.

Figure 6.6 Power Units

6.2.5 Computer

Neural network algorithm and controller algorithm have been embedded into the computer. Signals coming from EMG sensors and actuators have been interpreted inside the computer and related output signals have been produced based on algorithms. In short, the computer plays the role of the brain inside the system. Later the mechanism will be mobilized and instead of a personal computer, a Raspberry Pi 2 will be used.

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not include peripherals (such as keyboards, mice, and cases). However, some accessories have been included in several official and unofficial bundles [22].

6.2.6 Entire Electronic System

All electronic components have been discussed in previous sections. In this section connections between them are illustrated and explained. Fig. 6.7 below shows the connections:

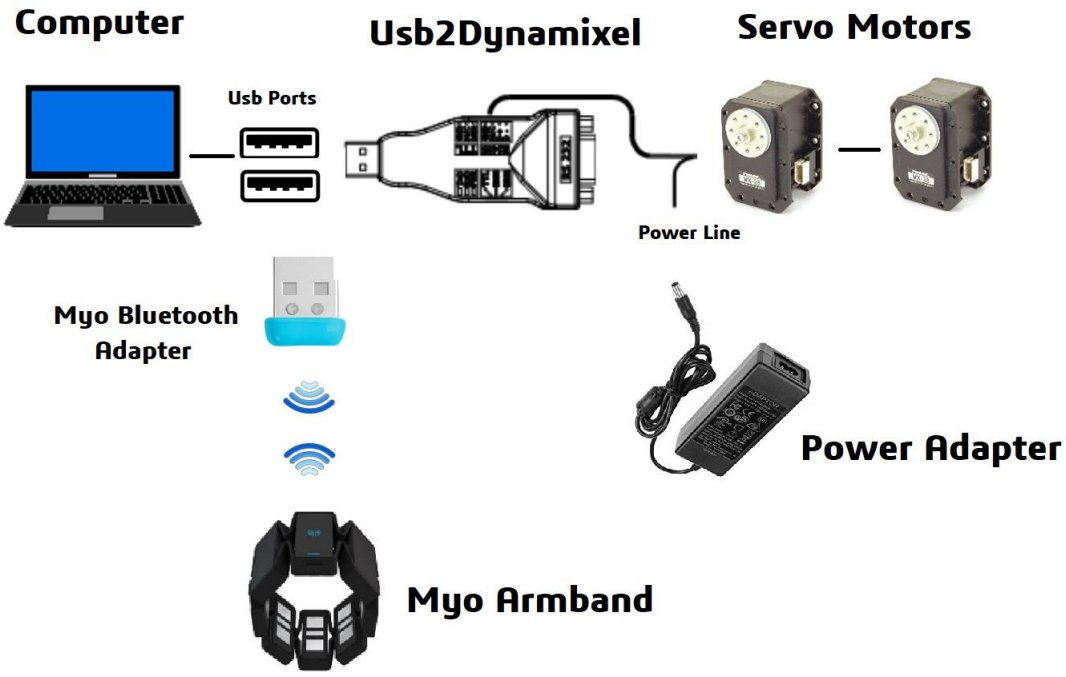


Figure 6.7 Entire Electronic System

The connections between each motor, motors and the Usb2Dynamixel, motors and power adapter have been handled with the special 3-pin connectors (TTL communication).

6.3 Software Implementation

Solidworks and MATLAB-Simulink software have been used to obtain mechanical design, controller design and system modelling, even for simulations, but these topics have been discussed in sections 4.

Content of section 6.3 can be divided into 4 subtitles: Data Acquisition from Myo Armband, Signals Processing, Neural Network Application and Motor Driving. For all these three subtitles MATLAB has been used as a software. Detailed information will be given in the below sections.

6.3.1 Data Acquisition from Myo Armband

In order to apply a neural network a training data set is needed. To obtain this training data set, a graphical user interface that is able to acquire real-time data from myo armband has been created using MATLAB. In Fig 6.8 a screenshot of the MATLAB-GUI is shown.

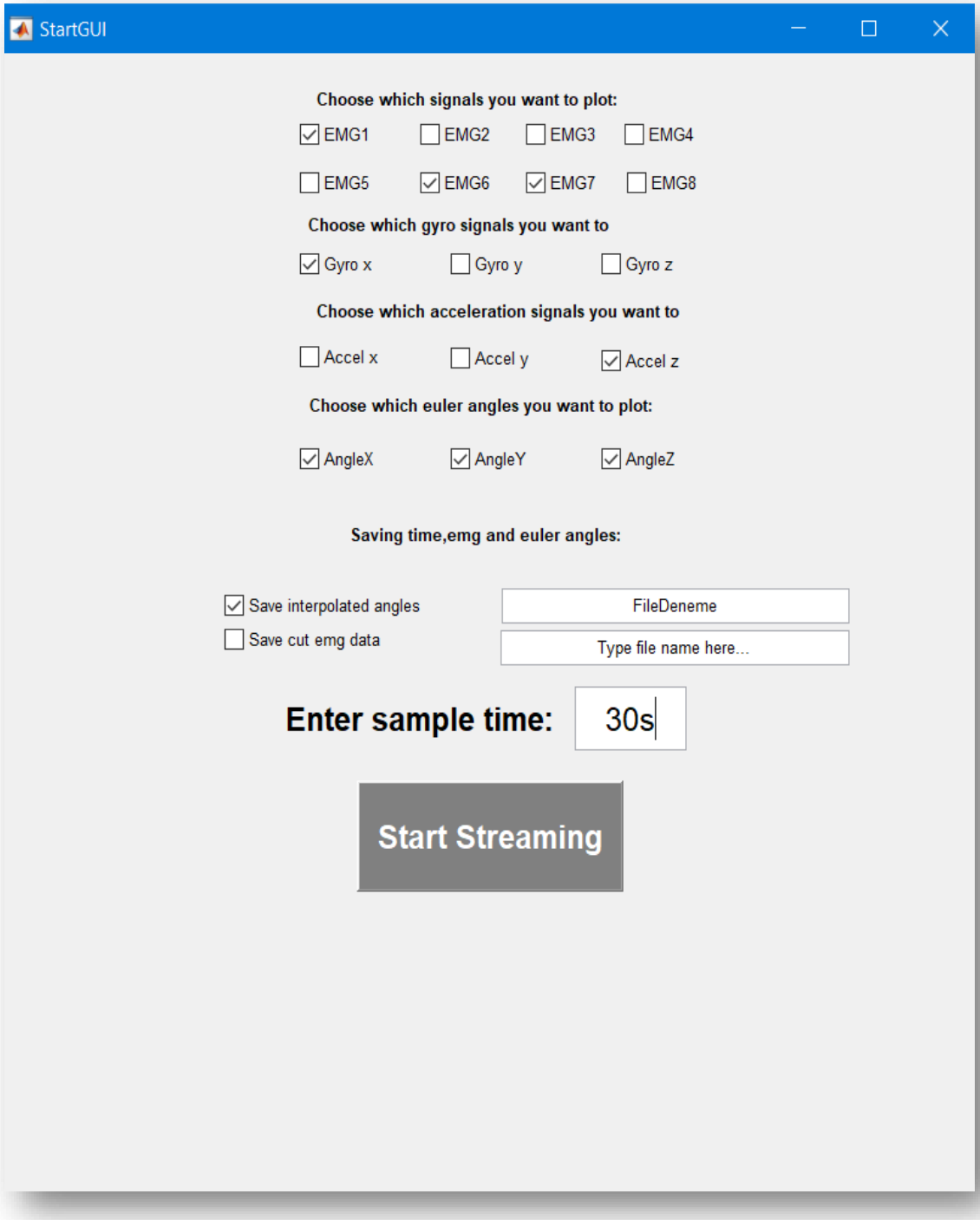


Figure 6.8 Data Acquisition GUI

User can choose which data he/she wants to plot just by clicking the related checkboxes . There are 8 different EMG electrodes inside myo armband, the user can plot and observe all of them separately. Also, acceleration, gyroscope and Euler angle data can be observed. In order to save data to an excel file user can simply click the 'save interpolated angles' checkbox and enter the file name. After entering the sample time value to the related dynamic text element, user can simply click start streaming button and he/she will observe real-time data plots. In Fig. 6.9 a screenshot of the interface while real-time data is streaming is shown.

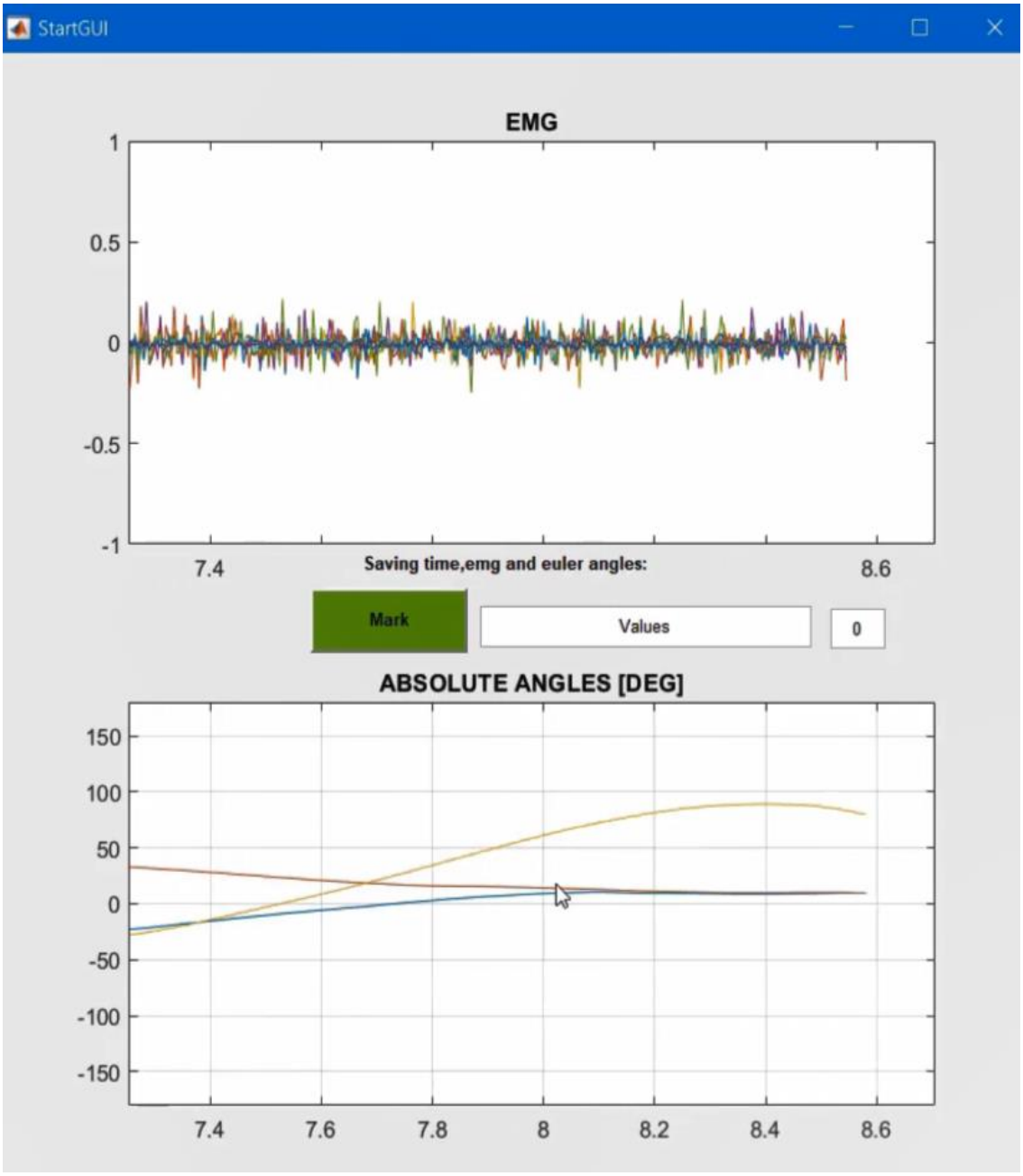


Figure 6.9 Data Acquisition GUI Streaming

The green mark button has been used to mark specific data points, hereby signals processing will be easier after acquiring and save the data. 8 different real-time EMG data and 3 different real-time Euler angle data is shown on the interface.

In short, the data acquisition interface has been used to obtain data needed to train a neural network. Also, by obtaining Euler angles, movement trajectories of any user can be constituted. Later, obtained movement trajectories can be stored in the device memory and mimic by the leg mechanism to assist the user.

Related MATLAB code for data acquisition GUI is shown in Appendix B.

6.3.2 Signal Processing

Acquired EMG and position data are stored as an Excel file and later it has been processed to be ready for the artificial neural network application. At the beginning of the project, the objective of the artificial neural network application was to determine instantaneous joint angles just by observing the EMG data. Therefore, some signal processing steps have been implemented to the acquired data and the Fig.6.10 shown below is obtained.

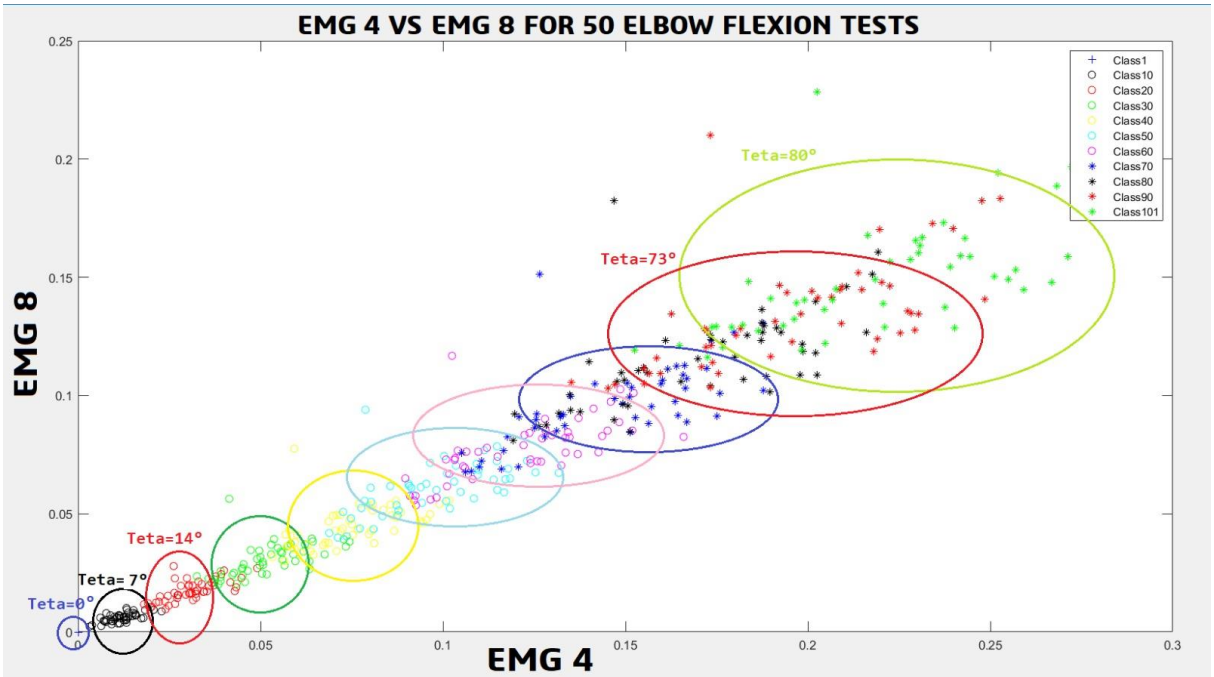


Figure 6.10 Signal Processing 1

50 elbow extension movements have been realized and related data is stored. Later, stored data is processed. The MATLAB code for signal processing is shown in Appendix C. The graph above represents processed EMG signal levels (channel 4 vs channel 5) for different elbow angles. Every different class corresponds to different joint angles. As can be seen, for small elbow angles like Theta = 0° or Theta = 7° and so on, it is possible to make a classification. Despite that, while Theta is getting bigger, the classification process gets harder because EMG data is nested and overlapped for bigger elbow angles.

As a result of this nested and overlapped data, it has been decided to change the objective of

the neural network. Instead of determining instant angles, the neural network has been used to understand the user's intention by classifying different hand gestures such as flexion, extension, fist and do-nothing cases. New signal processing codes are shown in Appendix D. As a result of this signal processing step, the Fig.6.11 shown below is obtained.

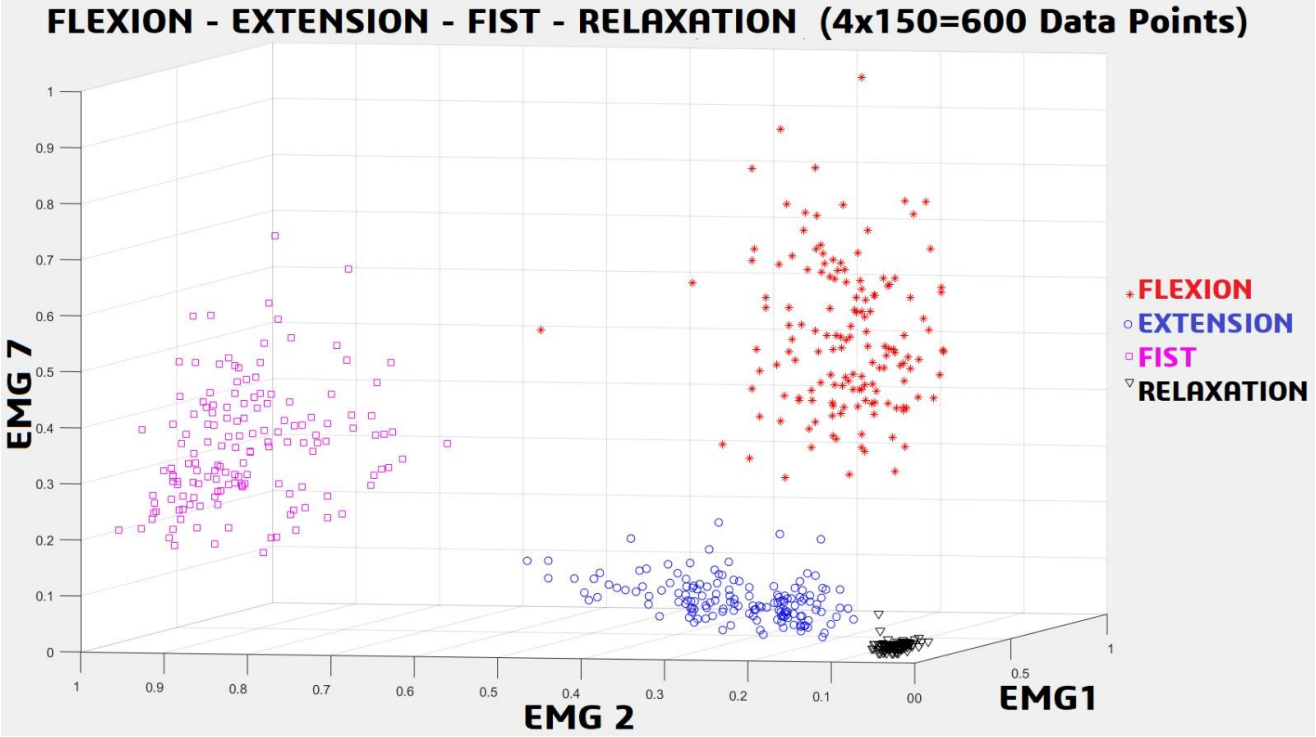


Figure 6.11 Signal Processing 2

In the figure above, processed EMG data (channel 1 - channel 2 - channel 7) is plotted. Every different group of data represents different hand gestures and it can be seen that every data set is clearly separated from each other.

Therefore, by training a neural network with this separated data, it can be accomplished to classify different hand gestures and understand the user's intention.

6.3.3 Artificial Neural Network Application

Artificial neural networks are one of the main tools used in machine learning. As the "neural" part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

In order to classify 4 different hand gestures (flexion, extension, fist and relaxation) an artificial neural algorithm has been used. The general structure of the used neural network is shown in Fig.6.12 below:

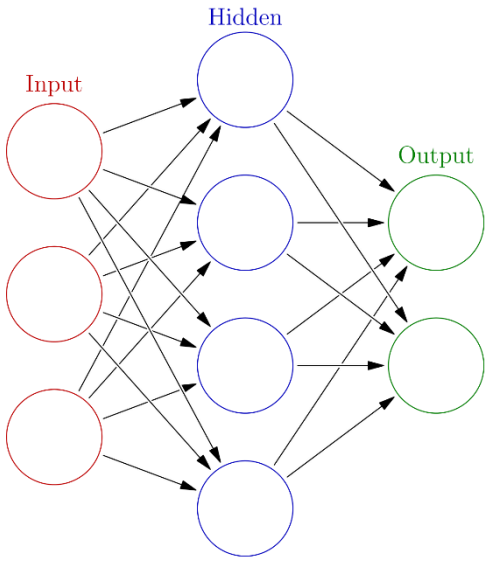


Figure 6.12 Artificial Neural Network Structure

Each unit is a logistic regression unit and inputs are bias signal 1, 2nd EMG channel of the armband and 7th EMG channel of armband respectively. There are 4 different output possibility: 00, 01, 10, 11 each one of these outputs represents different hand gestures. As a result, by training the neural network by using the code in Appendix E it has been accomplished to obtain neural network coefficients that provide the classification of 4 different hand gestures. Training data for the neural network are the data plotted in Fig. 6.11.

6.3.4 Motor Driving

To drive motors and actuate the system. Robotis had launched a software development kit named Dynamixel SDK to write needed codes in various environments and languages.

The ROBOTIS Dynamixel SDK is a software development library that provides Dynamixel control functions for packet communication. The API is designed for Dynamixel actuators and Dynamixel-based platforms. It assumes that you are familiar with C/C++ programming. This e-Manual provides comprehensive information about ROBOTIS products and applications [7].

Dynamixel SDK also contains example codes available for each programming language and operating system. For this particular project, it is decided to use MATLAB and the example code of 'bulk_read.m' file is modified. Related motor driving codes are shown in Appendix F. Keep in mind that, as an input, hip and knee trajectories obtained in section 3.4 is used. The code is embedded into a MATLAB GUI, thus it makes the motor driving process easier.

6.4 System Integration

After the mechanical, electronical hardware along with the software implementation of the project tested to eliminate possible malfunctions, the systems are integrated to mechanism's current stage called the demo mode.

A 20 x 20 sigma chassis is produced to hold the mechanism and the mechanical body is hung on to it. Electronic components that contained needed software are placed in their places, and due to their physical locations on the exoskeleton, needed extension cables are done by manipulating the jumper and 3-pin cables. All of the electronic components except motors are placed on a wooden board attached to the top of the chassis for the ease.

Later on, when the mechanism becomes a mobile industrial product, instead of a computer a Raspberry Pi 2 will be used and components will be placed inside of the manufactured waist mechanism. Also, it should not be forgotten that in the mobilized version of the exoskeleton, the power cells will be replaced with the Li-Po batteries with corresponding voltage and current values.

Figure 6.13-a Integrated Entire System With User

Figure 6.13-a Integrated Entire System Without User

7. ASSUMPTIONS AND ANALYSIS OF POSSIBLE ERRORS

7.1 Assumptions

In this design of the mechanism, following acceptances and assumptions are made:

• In contrast to the other exoskeleton mechanism designs for helping the fully disabled patients, this mechanism is designed to assist the partially disabled people.
• The exoskeleton is designed and manufactured in the consideration of its adult users.
• From exoskeleton motions, the adduction and abduction movements are disabled for demo works in order to avoid its effect on flexion/extension movements, which is the main task of the exoskeleton system. Moreover, the remnant effect of adduction/abduction is neglected.
• The system is assumed to be working on a non-gradient, obstacle-free, plain road.
• For the demo work, the system is immobilized. Also, the waist mechanism had been not manufactured. Instead, a frame is designed to stabilize the mechanism.
• Maximum limit of user's weight is determined as 100 kilograms and height is 1.75 m. Additionally, the upper limit for the disability rate of the user is 50%.
• The torques and/or powers of the disabled individual's leg is accepted as proportional with his/her disability rate, directly. (i.e. for the disability rate of 50%, it is considered as half of the normal/healthy torque/power)

7.2 Analysis of Possible Errors

In the gearbox located at the hip area, there is a minor axis shift of 1 mm that may cause harm to the motor at higher torques. This error can be eliminated by replacing the gear shaft holder. And also, the PID gains calculated according to Computed-Torque Control Algorithm and these are not working perfectly when they have embedded to the motor PID gains. So, slight overshoots occur from time to time. In order to solve this issue, another PID gain calculation must be handled. The weight of the system can be reduced by using different materials. Thus, easier position control results can be provided. Neural network learning has been made by using EMG signals from one person only, by increasing the person number a better result in the learning algorithm can be achieved.

Moreover, as it is taken in granting in the acceptances and assumption, the gait that the exoskeleton helps to perform should be on a non-gradient and collision-free path. Thus, a path with a slope also has the possibility of the motors to overload.

8. WORKING PLAN

8.1 Gantt Chart

The working plan can be seen in Fig. 8.1.

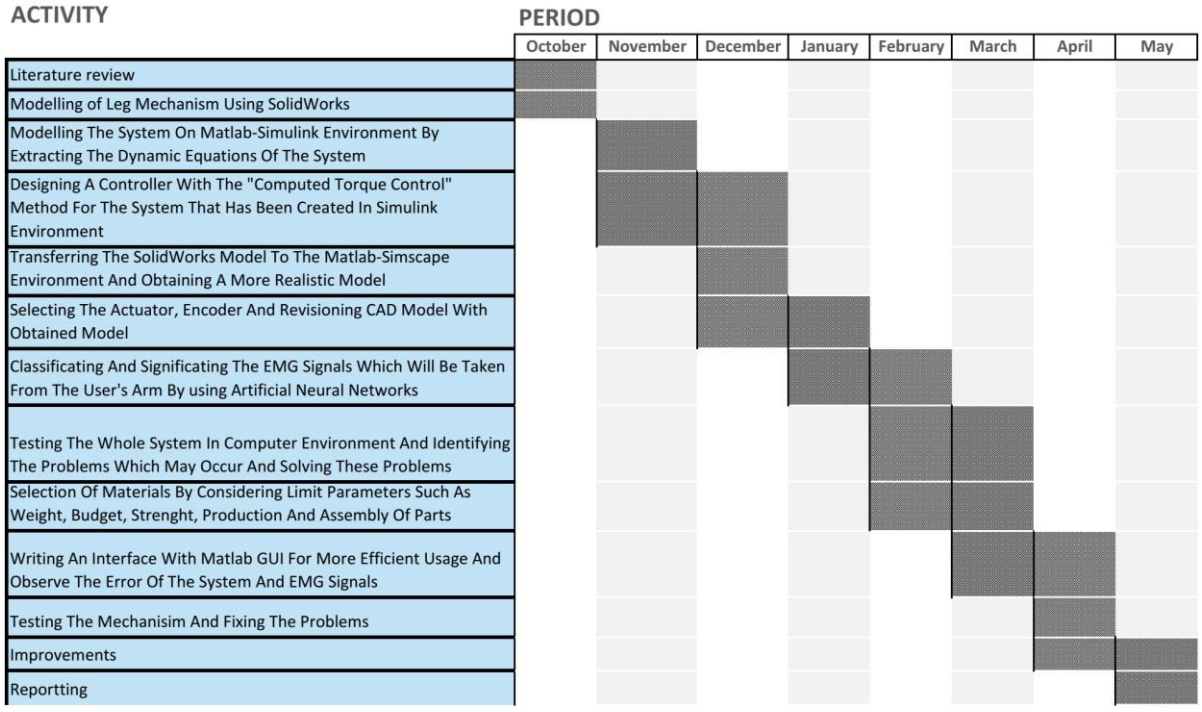


Figure 8.1 Gantt Chart Of The Working Plan

8.2 Task Distribution

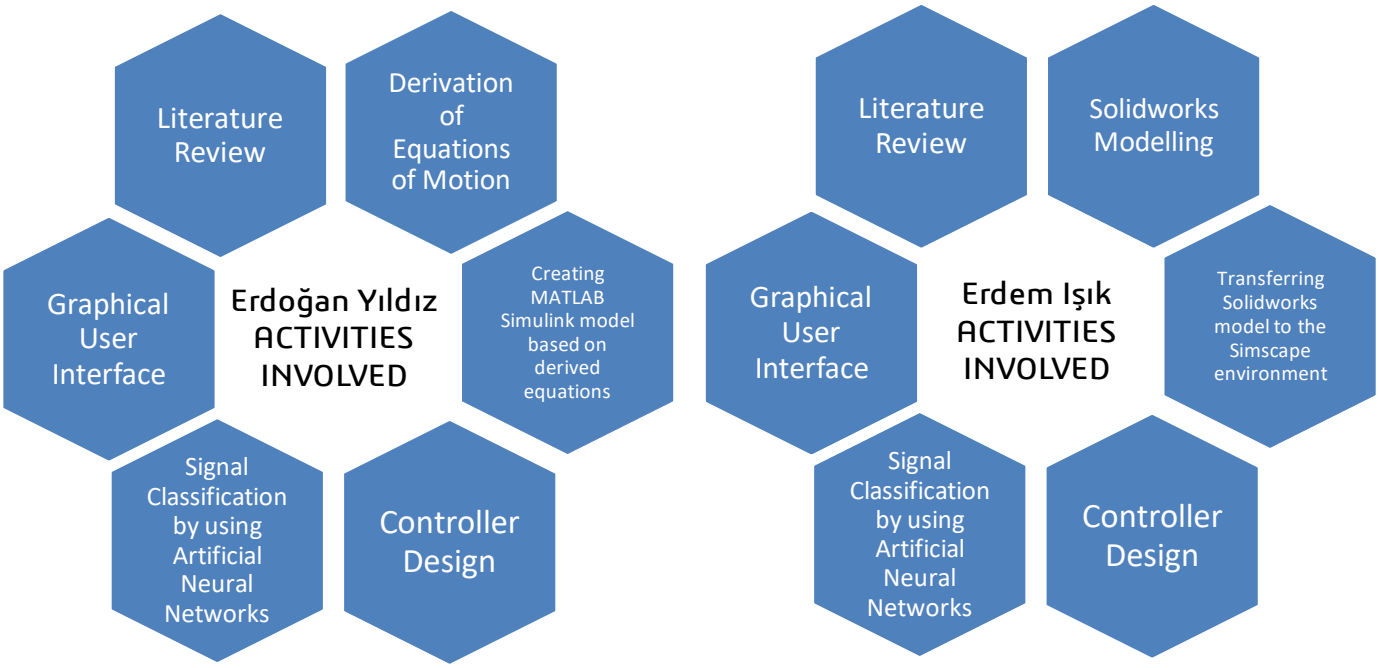


Figure 8.2 Task Distributions

In addition to Fig. 8.2, processes such as report preparation, presentation preparations, system tests, material selections have been decided by both team members involved. As can be seen in Fig. 8.2, most of the works are being done together by both team members. The reason behind this is to understand how to properly use theoretical information obtained during the education period. Since this is a nonprofit project, the main objective of both team members were to gain experience on different subjects.

9. BUDGET

9.1 List of Materials

Table 9. 1 List of Materials

Name of the Material	Piece	Total Price
Robotis Dynamixel MX-106T	2	5.000 ₺
Robotis USB2Dynamixel	1	250 ₺
Raspberry Pi 3	1	250 ₺
Myo Armband	2	3500 ₺
Mechanical: Materials	-	1000 ₺
Mechanical: Labor Cost	-	1500 ₺
Frame	1	200 ₺
Other Expenses	-	250 ₺
Total Cost		16.950 ₺

9.2 External Financial Support Applications

For the purpose of finding external financial support, Tubitak 2209-B Industry Oriented Thesis Support Program application has been done for the 4th period and denied. Later, for the 5th period, another application has been done and accepted. The project has won 3500 TL financial support from Tubitak 2209-B Industry Oriented Thesis Support Program.

10. RESULTS, DISCUSSION AND FUTURE WORKS

10.1 Results and Discussion

The resultant product of this project is a demo version of an assistive lower extremity exoskeleton. As it is mentioned, in the demo version of the system, it is immobilized. Therefore, a frame is manufactured instead of the waist part to hang the exoskeleton because it is not tested on humans. Also, the actuation of the motors and other electronic components achieved by a 12 DC, 4A AC to DC converter, so that the power can be taken directly from a plug.

Dynamixel MX series motors have it is own position controller as a PID Controller. The control of the motors is handled in that way providing the desired data point in joint space and defining the PID gains to the motor. According to data taken from encoders of the motors, the paths of the upper link, lower link and sole path can be observed in Fig. 10.1, Fig. 10.2 and Fig. 10.3

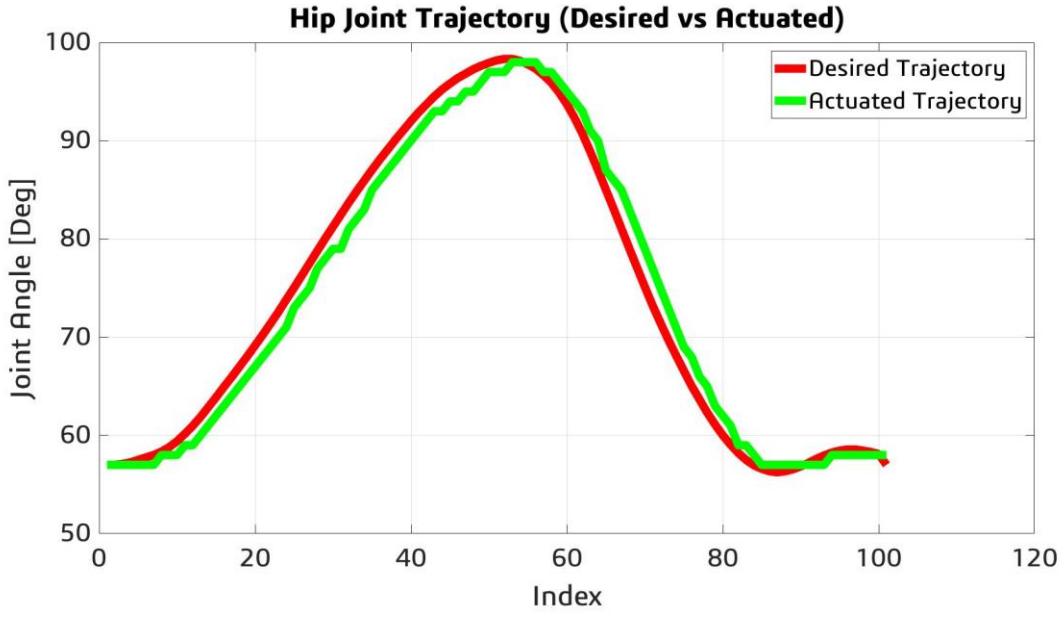


Figure 10.1 Real-life Test Hip Joint

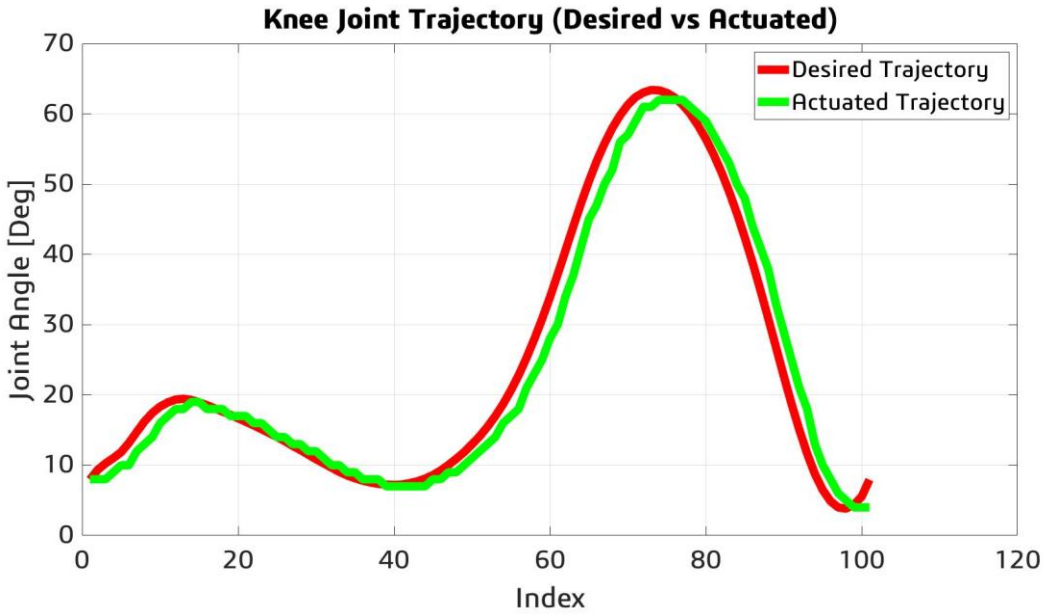


Figure 10.2 Real-life Test Knee Joint

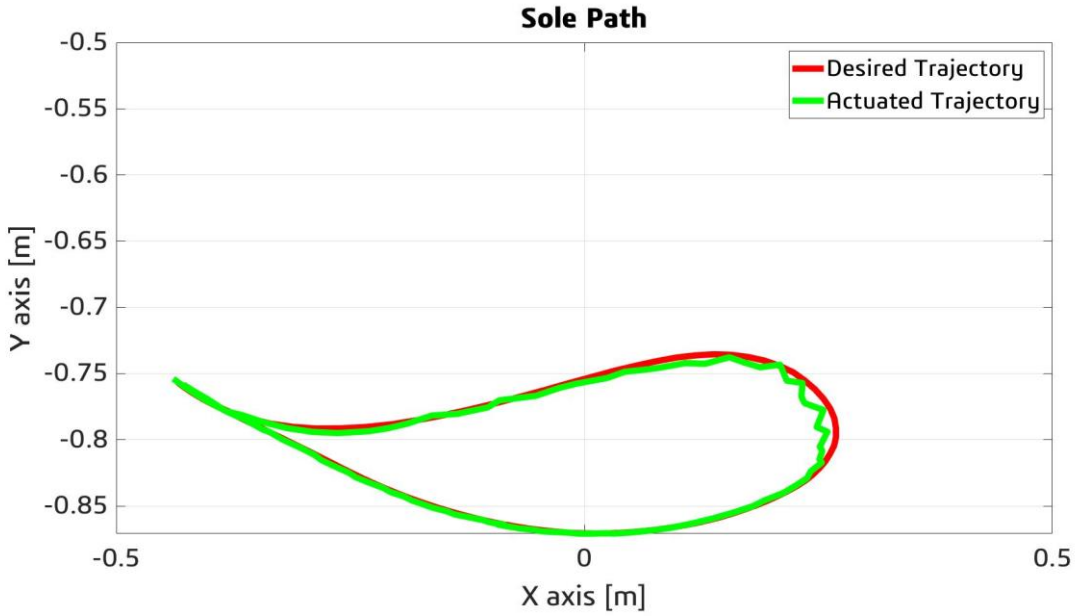


Figure 10.3 Real-life Test Sole Path

The deviations of the motor joint positions can be reduced by isolating and/or reducing the effects of hip and knee motors to each other and re-selecting the PID gains of the motor as it is mentioned in Section 7.2 Analysis of Possible Errors.

In Table 10.1, maximum error values for mathematical model, simscape model and real-life model is shown.

Table 10. 1 Maximum Tracking Errors

Maximum Tracking Error Values [Deg.]		
	Hip Angle	Knee Angle
Mathematical Model	0.12 °	0.23 °
Simscape Model	0,77 °	0.82 °
Real-life Model	3.9 °	7.1 °

As it can be seen that the position errors in the Mathematical Model are really small by the virtue of it is the based model for the PID Controller as well. When it comes to the Simscape Model, the error values are almost 5 times as high as 0,82°, because of the nonlinearities such as frictions, included in the Simscape Model. But the error values still remain within around 1.0~1.5% of the range. Finally, the error of the real-life motors is almost 5 times bigger as high as 3,9° for the hip and almost 9 times bigger for the knee as high as 7,1°. The reason behind this error increasing is the difference between real-life manufactured model and the CAD Design. Moreover, the deviation of error is bigger in the knee than the hip. That is because unlike in the simulations, in the model knee rotations effect from the hip rotations.

In short, the increase in the tracking error values between three models was expected as a consequence of assumptions and simplifications for the mathematical and simscape model. Although the resultant product is a demo version, real-life tracking error values of the joints are acceptable but must be reduced in order to obtain a better performance. There are three major solutions to reduce real-life tracking errors. Firstly, the small gap between gears which may cause some errors can be eliminated. Secondly, reducing the weight of the device will lead to reduced tracking errors. It may be handled by 3-D printing or by choosing some lightweight materials such as carbon fiber. Finally, using a better performance controller will lead to better tracking results. A better controller implementation is in our future plans.

Generated motor torque values for both mathematical and simscape models varies between -4 and +5 N.m. These results have also significant importance for the project since the motor selection process has been made mostly based on these values. Note that, smooth characteristics of generated motor torques are indicators for the controller and system to work

as desired, because as can be seen, there are no sudden jumps or sharp changes that are hard or maybe impossible to produce, in torque characteristics. Little jump at t=0 means, there will be a small error for beginning, and after a small amount of time, the behavior of the exoskeleton will be as expected. Thus, this little jump at the beginning can be ignored. Real-life results show that the motor selections were correct.

10.2 Summary, Conclusion & Future Works

In this thesis, an assistive lower-extremity exoskeleton leg mechanism is designed and implemented. The main purpose of the leg mechanism is to support people with walking difficulties due to orthopedic or neurological disorders, trauma or old age. In order to validate the design, variety of verifications have been done. “Computed Torque Control” method has been used in order to control the leg mechanism accurately so that it can assist the ongoing movement of the user.

The thesis consists of the mechanical design of leg mechanism, mathematical derivation of equations of motion, dynamic modelling, MATLAB modelling, controller design, simulations, neural network application and mechanical, electronic and software implementations.

Within the scope of thesis work, the mechanical design of the exoskeleton was performed using Solidworks software, the stress analysis of the critical parts where the mechanism is subjected to maximum loading was also carried out using Solidworks software by assuming 10 N.m motor torque is being applied to pins and reduction gear. Equations of motion for a rigid double pendulum model that represents the leg mechanism has been derived and a MATLAB Simulink model has been built based on the governing equations. Also, another MATLAB model has been built by transferring a solid model created on Solidworks software to the MATLAB Simulink / Simmechanics environment. A PID computed torque controller has been designed in order to control both mathematical and solid design-based models. Both MATLAB models are fed by the desired walking trajectory and design verifications and validations have been done by examining different kinds of outputs such as tracking error and required torques. An artificial neural network is applied into the system that can classify the EMG signals taken from the user’s arm and understands the intention by separating 4 different hand gestures such as flexion, extension, fist and relaxation.

The designed assistive leg mechanism can be used by people who have different body sizes (1.65 - 1.75 m long and up to 100 kg weight). Material selection has been made to ensure that the developed exoskeleton is light and durable. Required CAD design verifications have been made in part 5.1, as can be seen from Fig.s 5.1 – 5.4 it was confirmed that mechanism elements are resistant to maximum loads. Note that the tests in section 5.1 have been made for 150% nominal stress values.

The actuators' selection has been made according to torque values obtained from the simulation results in part 5.2 and 5.3 also other products and studies in the literature is used in order to decide motor speed values. As a result, Dynamixel MX-106T servo motors with parameters: 8.4 N.m stall torque and 45 rpm no load speed and 1:3 reducer are used in order to sufficiently provide required torque and rpm values.

Dynamic modeling of the lower extremity exoskeleton was performed using the Lagrange formulation. The obtained mathematical model contains a certain degree of error and uncertainty due to assumptions such as uniform rod body parts and the mass concentrated in the center of the mass. In addition to the mathematical model, another MATLAB model is constructed in order to reduce the errors and uncertainties caused by previous assumptions. The second model has been constructed by following these steps: the solid model has been obtained using Solidworks software, then the model has been transferred to the MATLAB Simulink / Simmechanics environment using an interface program. Then a dynamic model, that is more suitable for reality in terms of geometry, mass and inertia properties are obtained.

A PID computed torque controller is designed in order to control leg mechanism in a way that it can follow the desired path with acceptable error values. Detailed design and structure of the controller are shown in part 4.5.

By benefiting the controller designed, two different MATLAB system models were created as can be seen in part 4.6.

Both MATLAB models were tested for a generated joint angle trajectory vector, that represents walking. Detailed simulation inputs and outputs can be seen in parts 5.2 and 5.3.

The simulation results in parts 5.2 and 5.3 have significant importance for project verification and validation.

In parts 5.2 and 5.3 it can be seen that, for both models, the leg mechanism successfully follows the desired path. Position and velocity error graphs are also provided in parts 5.2 and 5.3 and both error values are in acceptable range considering the functionality of the project.

In part 6, detailed information about mechanical, electronic and software implementations is given and integration of the system is explained. In this context, in part 7 , assumptions and possible errors are discussed. Work plan and budget information are given in parts 8 and 9 respectively. Finally, in section 10.1 real-life test results are given and discussed.

In conclusion, a stable and durable assistive leg mechanism that can follow the given desired joint angle trajectory with acceptable tracking position and velocity errors have been designed and modeled. The system is capable of classifying 4 different hand gestures (flexion, extension, fist and relaxation) with almost no errors. After understanding the intention of the user, the system executes related trajectory and assist the ongoing movement of the user. The resultant product of this project can be considered as a demo version of an assistive lower extremity exoskeleton it is immobilized and not tested on a human. The tracking errors difference between simulation results and real-life tests were expected and can be reduced by implementing a better controller, or improving system bugs and reducing the system weight.

The real-life test and simulation results for both models verify the design and give confidence for the future works. Planned future works for the project as follows: Firstly, instead of a position control by using motor controllers, a current based controller is going to be applied. By applying this controller it is expected to significantly reduce the tracking error values. In addition, the system is going to be mobilized, in order to do that, all of the software implementations are going to be transferred to a Raspberry Pi 2 and instead of a Myo Armband, a faster and convenient EMG sensor will be used. Li-Po batteries are going to be used instead of a power adapter. All electronic equipment are going be placed to the waist mechanism designed. Finally, mechanical improvements such as reducing the mechanism weight or producing better gearbox might increase the system performance and reduce the errors.

REFERENCES

[1] Akdoğan, E. "Biyomekatronik Sistemler", *Otomasyon Dergisi*, p.288, 2012

[2] Bovi, Gabriele, et al. "A multiple-task gait analysis approach: kinematic, kinetic and EMG reference data for healthy young and adult subjects." Gait & posture 33.1 (2011): 6-13.

[3] Bovi, G., Rabuffetti, M., Mazzoleni, P., Ferrarin, M. (2011). A multiple-task gait analysis approach: kinematic, kinetic and EMG reference data for healthy young and adult subjects. Gait & posture, 33(1), 6-13.

[4] Chan, Margaret, and R. B. Zoelick. "World Report On Disability." WHO & The World Bank (2004), s.208

[5] Chen, Bing, et al. "Recent developments and challenges of lower extremity exoskeletons." Journal of Orthopaedic Translation 5 (2016), s.27

[6] Chu, A., Kazerooni, H., Zoss, A.B. 2005. On the biomimetic design of the Berkeley lower extremity exoskeleton (BLEEX). Proceedings of the IEEE Int. Conf. on Robotics and Automation ICRA, Barcelona, Spain, April 18-22.

[7] Dynamixel MX-106. Robotis E-Manuel. (2018). Retrieved June 5 2018 from http://emanual.robotis.com/docs/en/dxl/mx/mx-106/#connector-information

[8] Esquenazi, A.,Talaty M., Packel A.," The ReWalk Powered Exoskeleton to Restore Ambulatory Function to Individuals with Thoracic-Level Motor-Complete Spinal Cord Injury", American Journal of Physical Medicine & Rehabilitation, Volume 91, 911-921 , November 2012.

[9] "Exoskeleton Market Size, Analysis & Research | Industry Report, 2025." Exoskeleton Market Size, Analysis & Research | Industry Report, 2025, www.grandviewresearch.com/industry-analysis/exoskeleton-market.

[10] İşcan M., Yesildirek A., Emec C." Hand gesture movement classification based on dynamically structured neural network", April 2018

[11] Hlaváč, Václav. "Robot trajectory generation." Czech Technical University, Prague.

[12] Lynch, K., Park, F., "Modern Robotics: Mechanics, Planning, And Control", Page 403, May 3 2017

[13] Makinson, B J.," Research and Development Prototype for Machine Augmentation of Human Strength and Endurance. Hardiman I Project ", Defense Technical Information Center, AD0724797, 1-22, May 1971.

[14] Marion, J. B. (2013). Classical dynamics of particles and systems. Academic Press.

[15] Mechanical design of the Hanyang Exoskeleton Assistive Robot(HEXAR) Oct. 2014 Wansoo Kim, Heedon Lee, Donghwan Kim, Jungsoo Han, Department of Mechanical System Engineering, Hansung University, Seoul, South Korea

[16]Mozhanova, Marzhan. "Design of a High–Resolution Surface Electromyogram (EMG) Conditioning Circuit ." N.p., 4 Jan. 2012. Web. 20 May 2017.

[17] Naruse, K., Kawai, S., Kukichi, T. 2005. Three-dimensional lifting-up motion analysis for wearable power assist device of lower back support. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Edmonton, Canada, August 2-6.

[18] Raez M.B.I., Hussain M.S., Mohd-Yasin F., ”Techniques of EMG signal analysis: detection, processing, classification and applications“, US National Library of Medicine National Institutes of Health, 23 Mar 2006

[19] Rex Bionics Ltd. “REX is a hands-free robotic device for rehabilitation”. https://www.rexbionics.com/product-information/, 21 November 2018.

[20] Rosen J. ,Perry J.C. , Manning N. , Burns S. , Hannaford B. , ” Upper Limb Powered Exoskeleton”, University of Washington, Seattle, Washington, USA , 8 February 2007

[21] Rosen J. and Perry J.C. ,” The Human Arm Kinematics and Dynamics During Daily Activities – Toward a 7 DOF Upper Limb Powered Exoskeleton”, University of Washington, Box 352500, Seattle, Washington, USA , 15 July 2005

[22] "Ten millionth Raspberry Pi, and a new kit – Raspberry Pi". 8 September 2016. Retrieved 9 September 2016. we've beaten our wildest dreams by three orders of magnitude

[23] Vanderbilt University Center for Intelligent Mechatronics “Powered Lower-Limb Exoskeleton”. http://research.vuse.vanderbilt.edu/cim/research_orthosis.html, 21 November 2018.

[24] Whittle, Michael W. "Gait analysis." The Soft Tissues. 1993. 187-199.

[25] Willson, John D., et al. "Gluteal muscle activation during running in females with and without patellofemoral pain syndrome." Clinical Biomechanics 26.7 (2011): 735-740.

APPENDICES

Appendix A - Technical Drawings

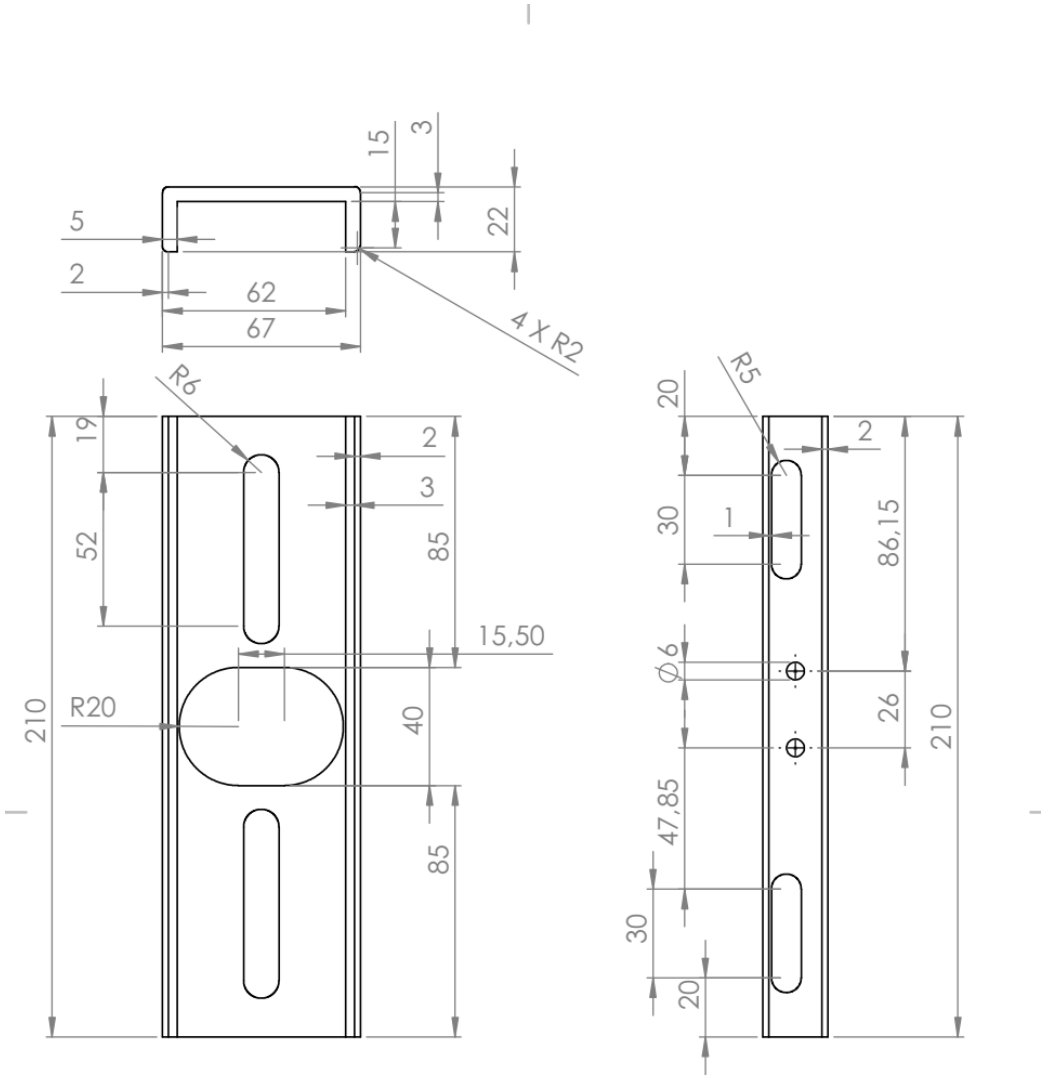


Figure 1 Outer Link

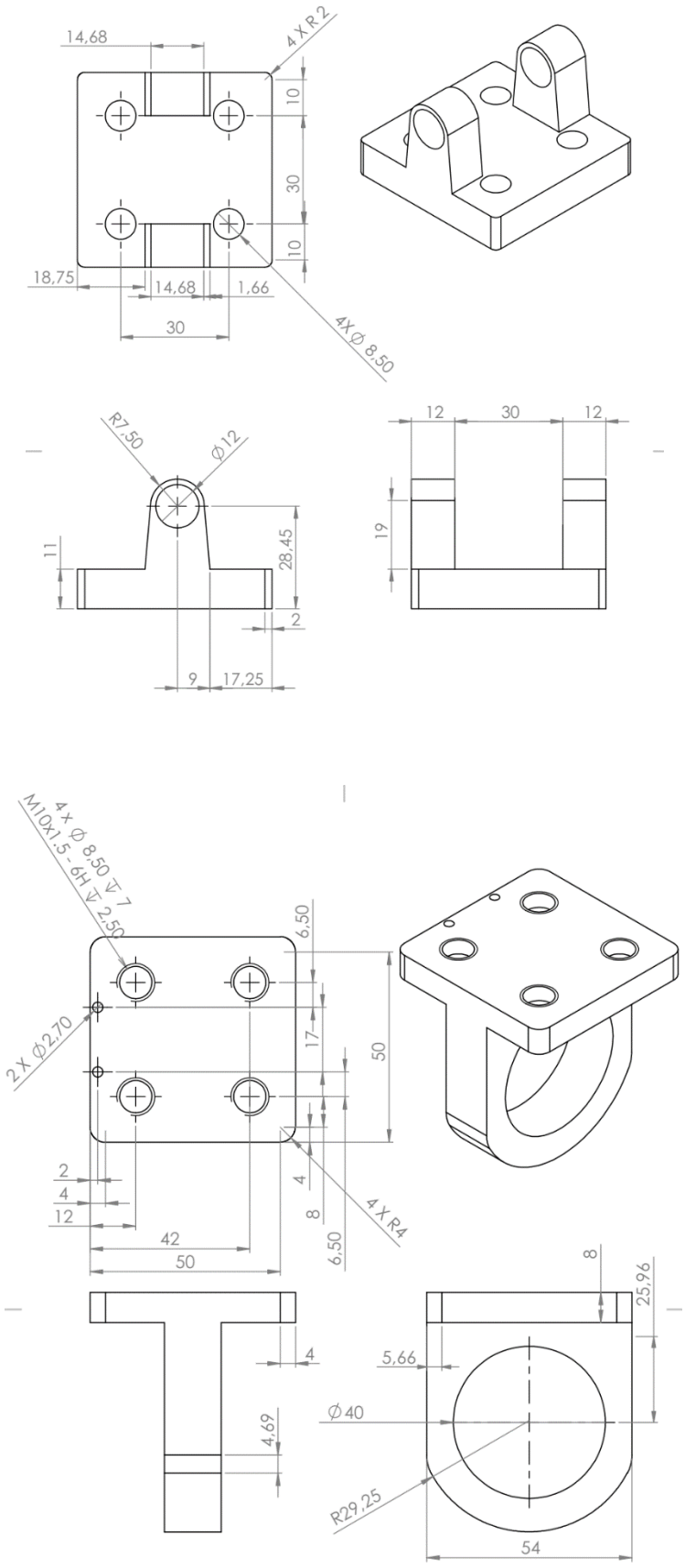


Figure 3 Top Bearing Base

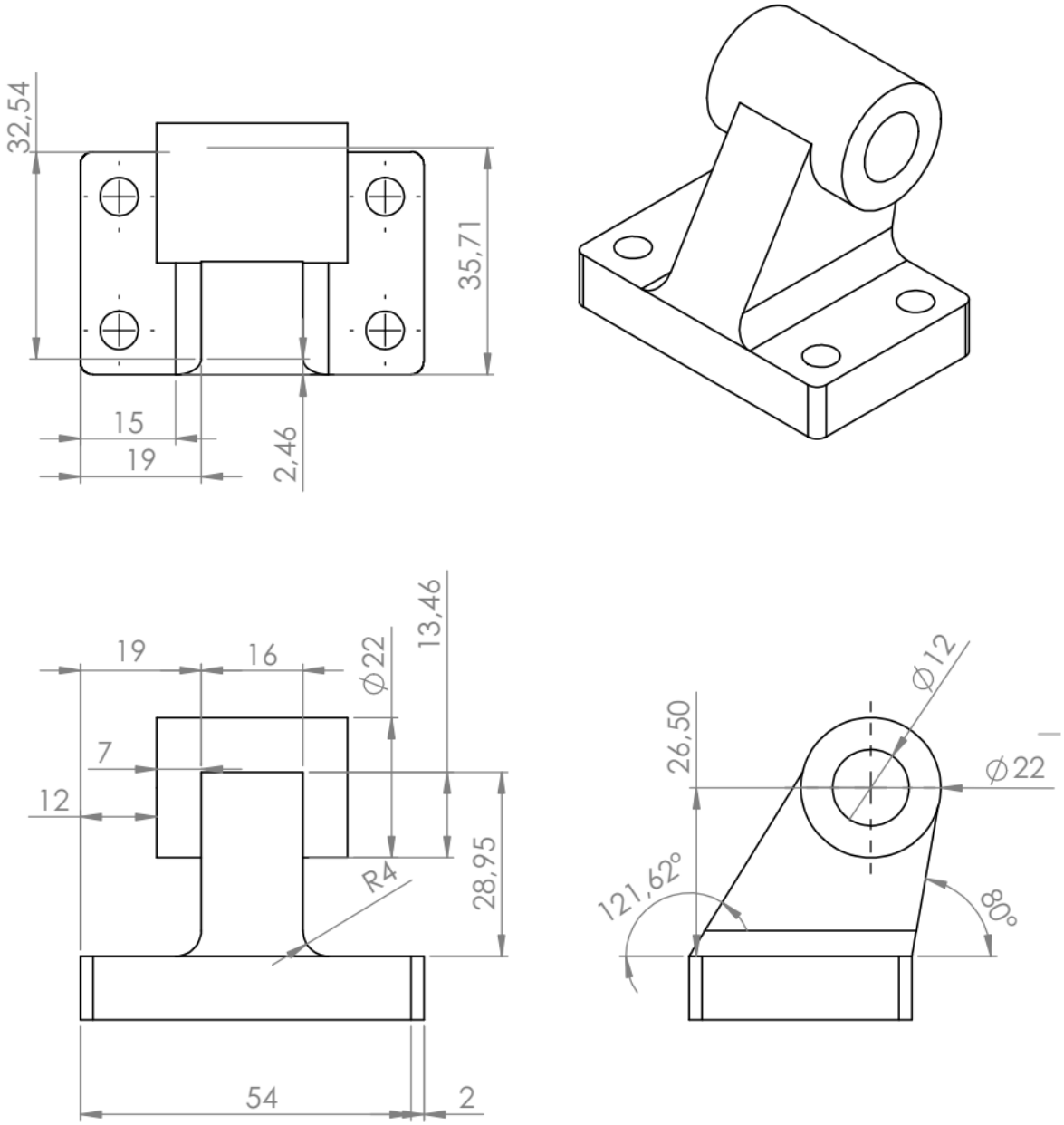


Figure 4 Top Shaft Base

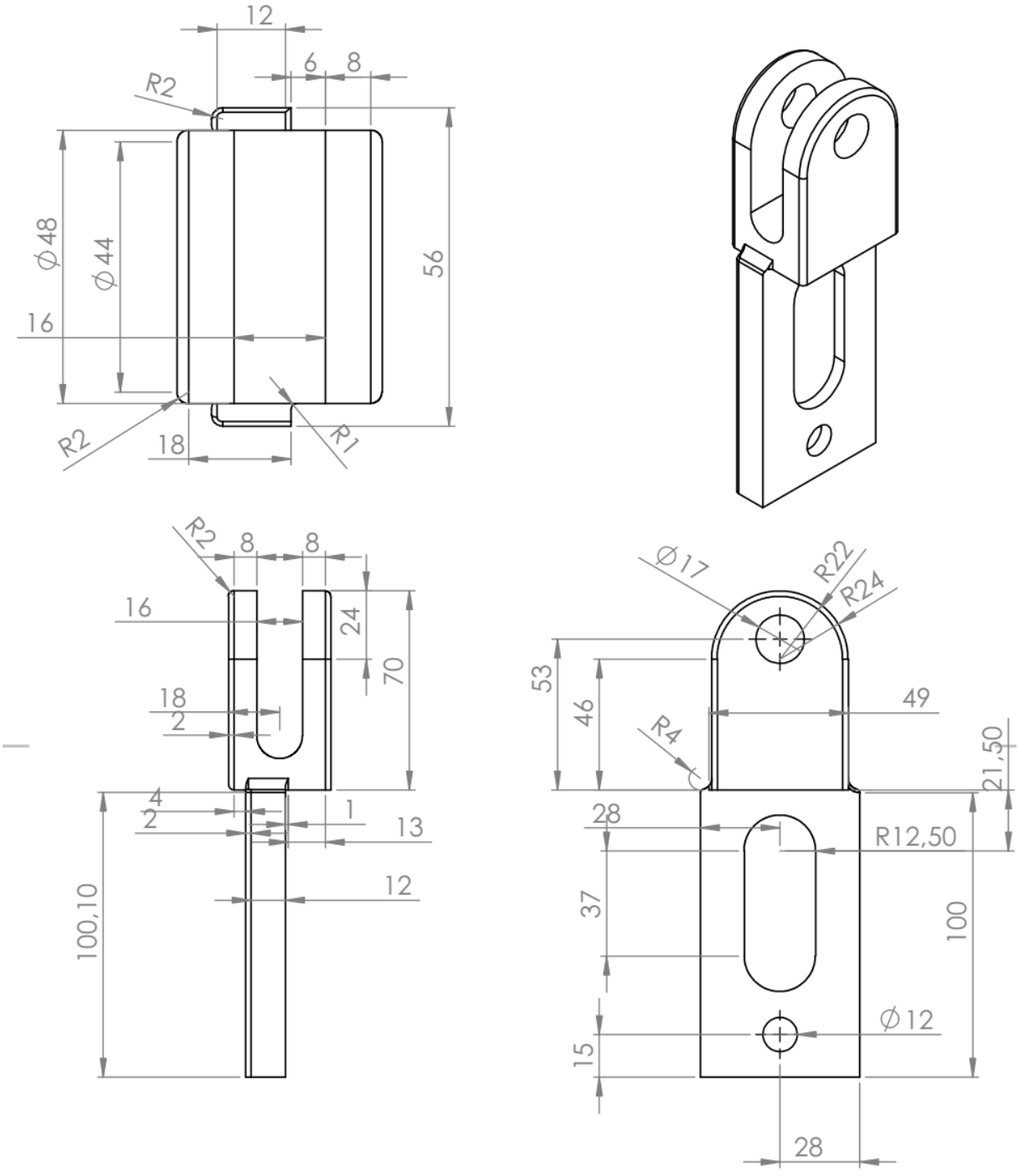


Figure 5 Upper Leg Inner Link Top

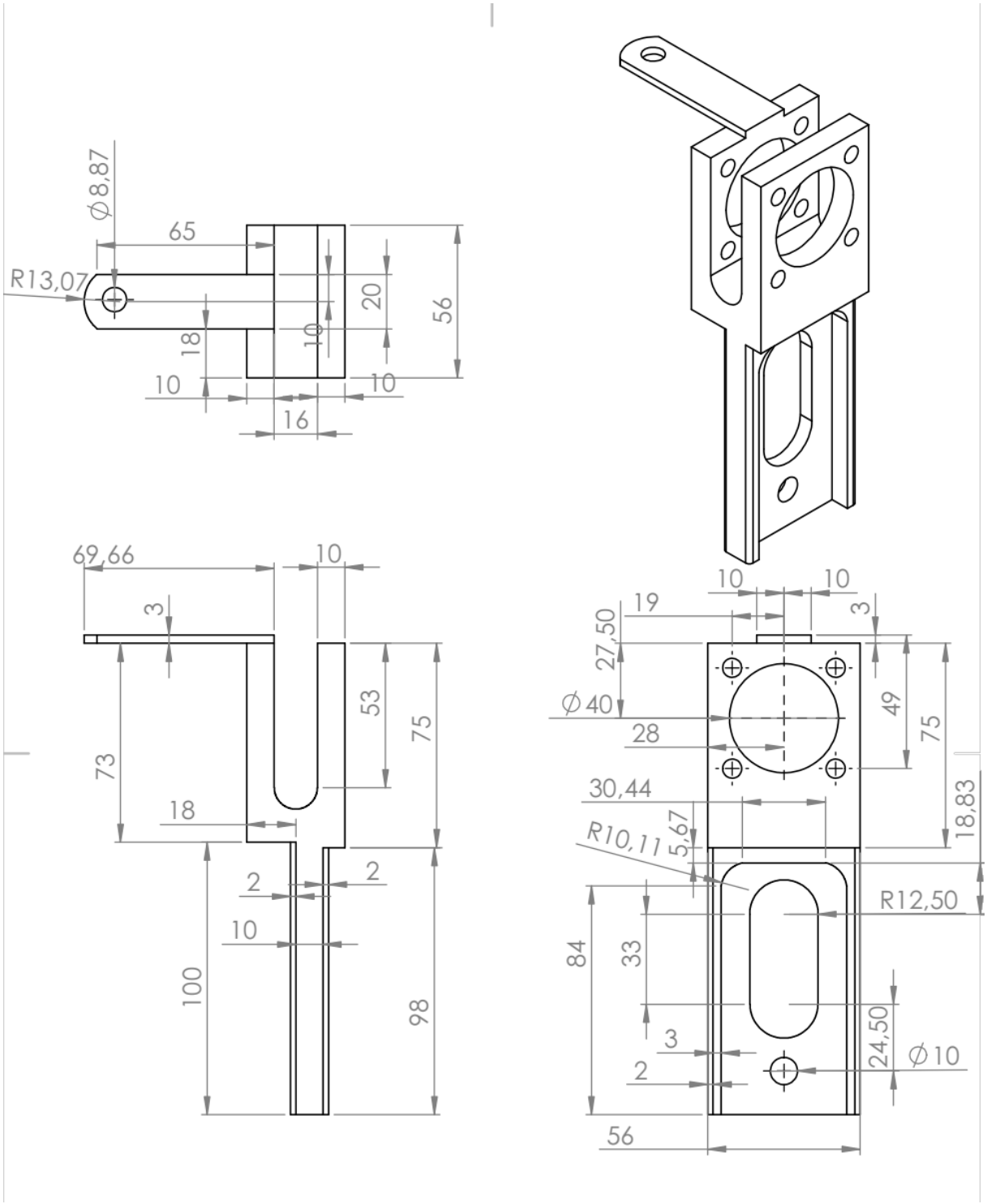


Figure 6 Upper Leg Inner Link Bottom

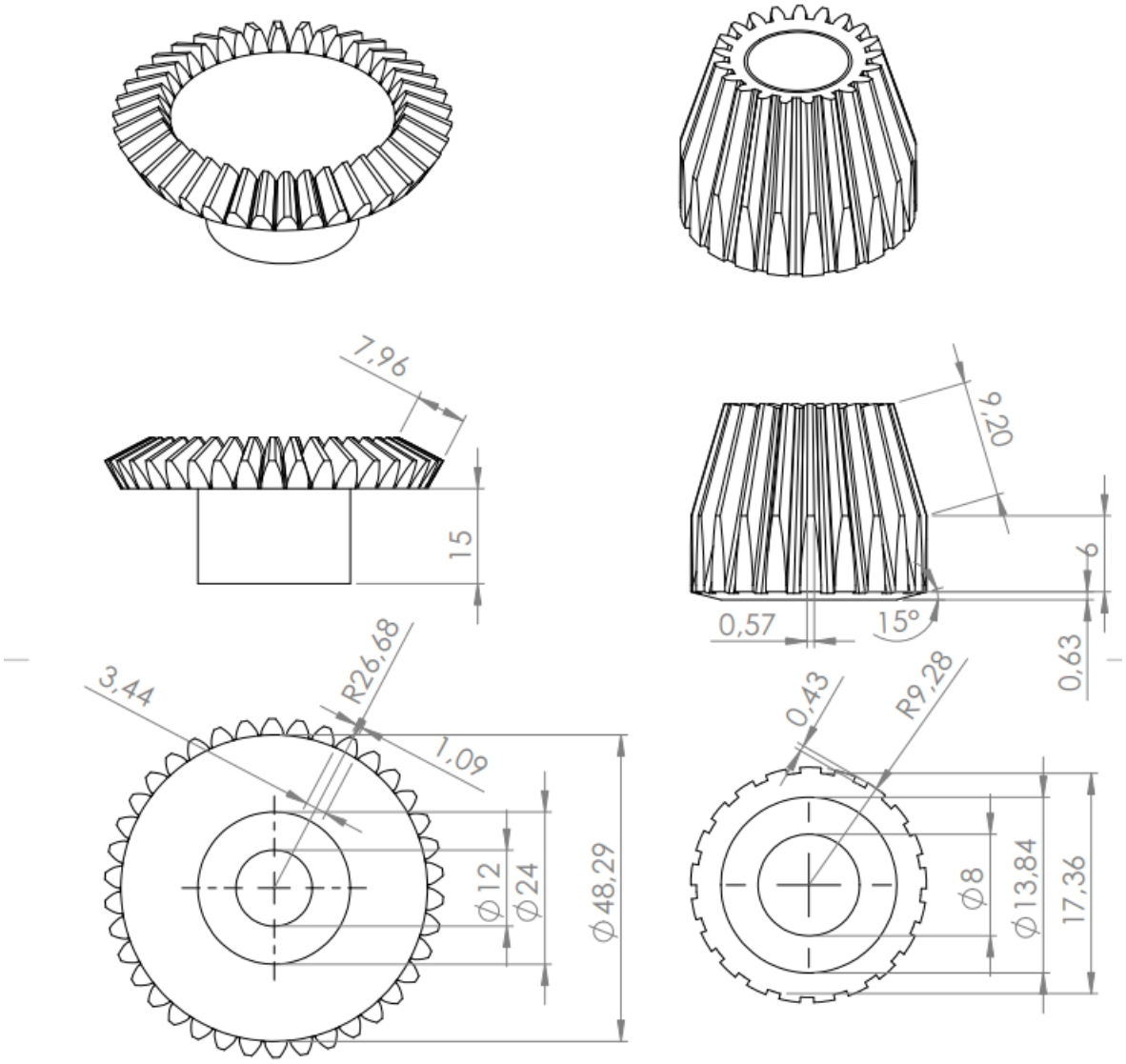


Figure 7 Gear Mechanism

Appendix B - Data Acquisition Codes

```
function varargout = StartGUI(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @StartGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @StartGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% --- Executes just before StartGUI is made visible.
function StartGUI_OpeningFcn(hObject, eventdata, handles, varargin)
clc
countMyos = 1;
handles.mm = MyoMex(countMyos);
handles.m1 = handles.mm.myoData(1);
handles.marksemgtime=-99*ones(100,1);

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

% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = StartGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

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


function edit1_Callback(hObject, eventdata, handles)

function edit1_CreateFcn(hObject, eventdata, handles)

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


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

T = str2num(get (handles.edit1,'String')); % seconds
handles.m1.clearLogs()
handles.m1.startStreaming();
```

```
%Close visibilities
set(handles.text2,'visible', 'off');
set(handles.text3,'visible', 'off');
set(handles.text4,'visible', 'off');
set(handles.text5,'visible', 'off');
set(handles.text6,'visible', 'off');
set(handles.text7,'visible', 'off');

set(handles.checkbox1,'visible', 'off');
set(handles.checkbox2,'visible', 'off');
set(handles.checkbox3,'visible', 'off');
set(handles.checkbox4,'visible', 'off');
set(handles.checkbox5,'visible', 'off');
set(handles.checkbox6,'visible', 'off');
set(handles.checkbox7,'visible', 'off');
set(handles.checkbox8,'visible', 'off');
set(handles.checkbox9,'visible', 'off');
set(handles.checkbox10,'visible', 'off');
set(handles.checkbox11,'visible', 'off');
set(handles.checkbox12,'visible', 'off');
set(handles.checkbox13,'visible', 'off');
set(handles.checkbox14,'visible', 'off');
set(handles.checkbox15,'visible', 'off');
set(handles.checkbox16,'visible', 'off');
set(handles.checkbox17,'visible', 'off');
set(handles.checkbox20,'visible', 'off');
set(handles.checkbox21,'visible', 'off');
set(handles.pushbutton1,'visible', 'off');

set(handles.edit1,'visible', 'off');
set(handles.edit5,'visible', 'off');
set(handles.edit7,'visible', 'off');

set(handles.pushbutton2,'visible', 'on');
set(handles.MarkText,'visible', 'on');
set(handles.MTV,'visible', 'on');
set(handles.text10,'visible', 'on');

set(handles.axes7,'visible', 'on');
set(handles.axes8,'visible', 'on');


%Plot EMG Data real-time

FX=0;
FS=0;
EX=0;
RX=0;
t = 0 ;
x = 0 ;
startSpot = 0;
tstart=99;



while ( t < T )
      if(length(handles.m1.timeEMG_log)>300 && length(handles.m1.timeIMU_log)>75)

            a=handles.m1.timeEMG_log;
            b=ones(length(a),1);
            timeemg = a-a(1).*b;

            emg = handles.m1.emg_log(1:length(timeemg),:);

            c=handles.m1.timeIMU_log;
            d=ones(length(c),1);
```

```
timevectorimu=c-c(1).*d;

gyromatrix=handles.m1.gyro_log();


%% NN Application
 maxEMG2=max(handles.m1.emg_log(end-20:end,2));
 maxEMG7=max(handles.m1.emg_log(end-20:end,7));

 input= [1; maxEMG2; maxEMG7];

W =      [ 0.8836   -7.0693    6.6697
           1.5403  -14.4231  -13.9882
          -9.7691   10.6543   27.0664
          -0.1365   16.2684  -13.0495];


V =      [3.3316   10.5119  -27.4685   12.1715
         -10.7406  -19.9309  -15.7503   25.9366];

z=zeros(4,1);

z(1) = 1/(1 + exp(-W(1,:)*input));
z(2) = 1/(1 + exp(-W(2,:)*input));
z(3) = 1/(1 + exp(-W(3,:)*input));
z(4) = 1/(1 + exp(-W(4,:)*input));


y(1) = 1/(1 + exp(-V(1,:)*z));
y(2) = 1/(1 + exp(-V(2,:)*z));

result=2*y(1) + 1*y(2);


 if(abs(result-0)<0.0001) %FLEXION
    FX=FX+1;
    if(FX>5)
        set (handles.text10,'String','FLEXION');
    end
    FS=0;
    RX=0;
    EX=0;
 end

 if(abs(result-1)<0.0001) %FIST
    FS=FS+1;
    if(FS>5)
        set (handles.text10,'String','FIST');
    end
    FX=0;
    RX=0;
    EX=0;
 end

 if(abs(result-2)<0.0001) %DO NOTHING
    RX=RX+1;
    if(RX>5)
        set (handles.text10,'String','RELAX');
    end
    FX=0;
    FS=0;
    EX=0;
 end

 if(abs(result-3)<0.0001) %EXTENSION
    EX=EX+1;
    if(EX>5)
        set (handles.text10,'String','EXTENSION');
    end
    FX=0;
```

```
            FS=0;
            RX=0;
        end


        if(length(timevectorimu)== length(gyromatrix))
                tvi=timevectorimu;
                gyro1=gyromatrix(:,1);
                gyro2=gyromatrix(:,2);
                gyro3=gyromatrix(:,3);
                anglex=cumtrapz(tvi,gyro1);
                angley=cumtrapz(tvi,gyro2);
                anglez=cumtrapz(tvi,gyro3);
        else
                continue;
        end

        anglematrixgyros=[anglex,angley,anglez];

        timeemg=timeemg(end-299:end);
        emg=emg(end-299:end,:);
        timevectorimu=timevectorimu(end-74:end);
        anglematrixgyros=anglematrixgyros(end-74:end,:);

else

        if(length(handles.m1.timeEMG_log)>0 && length(handles.m1.timeIMU_log)>0)

                a=handles.m1.timeEMG_log;
                b=ones(length(a),1);
                timeemg = a-a(1).*b;

                emg = handles.m1.emg_log(1:length(timeemg),:);

                c=handles.m1.timeIMU_log;
                d=ones(length(c),1);
                timevectorimu=c-c(1).*d;

                gyromatrix=handles.m1.gyro_log();


                if(length(timevectorimu)== length(gyromatrix))
                                tvi=timevectorimu;
                                gyro1=gyromatrix(:,1);
                                gyro2=gyromatrix(:,2);
                                gyro3=gyromatrix(:,3);
                                anglex=cumtrapz(tvi,gyro1);
                                angley=cumtrapz(tvi,gyro2);
                                anglez=cumtrapz(tvi,gyro3);
                else
                                continue;
                end


                anglematrixgyros=[anglex,angley,anglez];

        else
         continue;
        end
end


p1=plot(handles.axes7,timeemg,emg); title('EMG');
p2=plot(handles.axes8,timevectorimu,anglematrixgyros);
title('ABSOLUTE ANGLES [DEG]');

        if (t < 1.2)
                startSpot = 0;
        else
                startSpot = t-1.2;
```

```
            end

       xlim(handles.axes7,[startSpot (t+0.25)]);
       ylim(handles.axes7,[-1 1]);

        xlim(handles.axes8,[startSpot (t+0.25)]);
        ylim(handles.axes8,[-180 180]);

        grid
        t = timeemg(end);
        drawnow;
        pause(0.001)


end


handles.m1.stopStreaming();
set(handles.pushbutton2,'visible', 'off'); %Mark button invisible!

timevectoremg=handles.m1.timeEMG_log -
handles.m1.timeEMG_log(1)*ones(length(handles.m1.timeEMG_log),1);
timevectorimu=handles.m1.timeIMU_log -
handles.m1.timeIMU_log(1)*ones(length(handles.m1.timeIMU_log),1);

checkEMGs=ones(1,8);
checkEMGs(1)= get(handles.checkbox1,'Value');
checkEMGs(2)= get(handles.checkbox2,'Value');
checkEMGs(3)= get(handles.checkbox3,'Value');
checkEMGs(4)= get(handles.checkbox4,'Value');
checkEMGs(5)= get(handles.checkbox5,'Value');
checkEMGs(6)= get(handles.checkbox6,'Value');
checkEMGs(7)= get(handles.checkbox7,'Value');
checkEMGs(8)= get(handles.checkbox8,'Value');
numberofchecksemg=sum(checkEMGs(:) == 1);

checkGYROs=ones(1,3);
checkGYROs(1)= get(handles.checkbox9,'Value');
checkGYROs(2)= get(handles.checkbox10,'Value');
checkGYROs(3)= get(handles.checkbox11,'Value');
numberofchecksgyro=sum(checkGYROs(:) == 1);

checkACCELs=ones(1,3);
checkACCELs(1)= get(handles.checkbox12,'Value');
checkACCELs(2)= get(handles.checkbox13,'Value');
checkACCELs(3)= get(handles.checkbox14,'Value');
numberofchecksaccel=sum(checkACCELs(:) == 1);

checkANGLEs=ones(1,3);
checkANGLEs(1)= get(handles.checkbox15,'Value');
checkANGLEs(2)= get(handles.checkbox16,'Value');
checkANGLEs(3)= get(handles.checkbox17,'Value');
numberofchecksangle=sum(checkANGLEs(:) == 1);



if(numberofchecksemg>0) %%Eğer hiç emg checkbox seçilmediyse skip!
    plotGUI

    k=1;

    for i=1:1:8
        if checkEMGs(i)
             subplot(numberofchecksemg,1,k);
             plot(timevectoremg,handles.m1.emg_log(:,i));
             title(strcat('EMG Sensor ',num2str(i)));
             k=k+1;
        end

        xlabel('time [s]')
```

```
        ylabel('%')

    end
end


if(numberofchecksgyro>0) %%Skip if no gyro checkboxes are selected!
    plotGYRO

    l=1;
    for m=1:1:3
        if checkGYROs(m)
            subplot(numberofchecksgyro,1,l);
plot(timevectorimu,handles.m1.gyro_log(:,m));   title(strcat('GYRO  ',num2str(m)));
            l=l+1;
            xlabel('time [s]')
            ylabel('*/s')
        end
    end
end


if(numberofchecksaccel>0) %%Skip if no acceleration checkboxes are selected!
    plotACCEL

    n=1;
    for j=1:1:3
        if checkACCELs(j)
             subplot(numberofchecksaccel,1,n);
             plot(timevectorimu,handles.m1.accel_log(:,j));
             title(strcat('ACCEL  ',num2str(j)));
             n=n+1;
             xlabel('time [s]')
             ylabel('g')

        end
    end
end


if(numberofchecksangle>0)
    plotANGLEs

    gyromatrix=handles.m1.gyro_log();

    anglex=cumtrapz(timevectorimu,gyromatrix(:,1));
    angley=cumtrapz(timevectorimu,gyromatrix(:,2));
    anglez=cumtrapz(timevectorimu,gyromatrix(:,3));

    anglematrixgyros=[anglex,angley,anglez];

    q=1;

    for u=1:1:3
        if checkANGLEs(u)
             subplot(numberofchecksangle,1,q);
             plot(timevectorimu,anglematrixgyros(:,u));
             title(strcat('Euler Angle  ',num2str(u)));
             q=q+1;
             xlabel('time [s]')
             ylabel('Angle[Degrees]')

        end
    end
end
```

```
%% Interpolate and Save to file:


if(get(handles.checkbox20,'Value')==1) %Interpolated

    %%Marking
    handles.marksemgtime=getappdata(0,'marks');
    guidata(hObject, handles);

    cutindexes=find(handles.marksemgtime==-99);
    handles.marksemgtime(cutindexes)=[];
    guidata(hObject, handles);

    markindex=(-1*ones(1,length(handles.marksemgtime)))';

    a=handles.m1.timeEMG_log;
    b=ones(length(a),1);
    timeemg = a-a(1).*b;

    if(handles.marksemgtime(end)~=-99)
        for o=1:1:length(markindex)
            markindex(o) = find(timeemg==handles.marksemgtime(o)) ;
        end
    end

    timeemglength=length(timeemg);
    markvector=zeros(1,timeemglength)';.

    if(handles.marksemgtime(end)~=-99) %
    markvector(markindex)=[1];
    end

    h1=handles.m1.timeEMG_log;
    h2=handles.m1.timeEMG_log(1)*ones(length(handles.m1.timeEMG_log),1);
    timevectoremg=h1-h2;

    h1=handles.m1.timeIMU_log;
    h2= handles.m1.timeIMU_log(1)*ones(length(handles.m1.timeIMU_log),1);
    timevectorimu=h1-h2;

    emgmatrix=handles.m1.emg_log; %Store emg data

    gyromatrix=handles.m1.gyro_log();

    anglex=cumtrapz(timevectorimu,gyromatrix(:,1));
    angley=cumtrapz(timevectorimu,gyromatrix(:,2));
    anglez=cumtrapz(timevectorimu,gyromatrix(:,3));

    anglematrixgyros=[anglex,angley,anglez];

    %% Expand anglematrix via interpolation. ( 50Hz ---> 200Hz )

    %Define the query points to be a finer sampling over the range of x.
    desiredtimeinterval = timevectoremg;
    %Interpolate the function at the query points and plot the result.
    imutimeinterp = interp1(timevectorimu,timevectorimu,desiredtimeinterval);

 anglesxinterp = interp1(timevectorimu, anglematrixgyros(:,1),desiredtimeinterval);
 anglesyinterp = interp1(timevectorimu, anglematrixgyros(:,2),desiredtimeinterval);
 angleszinterp = interp1(timevectorimu, anglematrixgyros(:,3),desiredtimeinterval);

    anglematrixdegrelativeinterp=[anglesxinterp, anglesyinterp, angleszinterp];

    % Save results to a file
```

```
    T =
table(timevectoremg,emgmatrix,imutimeinterp,anglematrixdegrelativeinterp,markvector
,'VariableNames',{'TimeEmg','Emg','TimeIMUInterpolated','InterpolatedAngles','IsMar
ked'});
    filename = strcat(get(handles.edit5,'String'),'.xlsx');
    % Write table to file
    writetable(T,filename)
    % Print confirmation to command line

    fprintf('Results table with %g emg measurements saved to file
%s\n',length(timevectoremg),filename)

end



%% Cut and Save to file:

if(get(handles.checkbox21,'Value')==1)

    timevectoremg=handles.m1.timeEMG_log -
handles.m1.timeEMG_log(1)*ones(length(handles.m1.timeEMG_log),1);
    timevectorimu=handles.m1.timeIMU_log -
handles.m1.timeIMU_log(1)*ones(length(handles.m1.timeIMU_log),1);

    emgmatrix=handles.m1.emg_log;
    emg1vector=emgmatrix(:,1);
    gyromatrix=handles.m1.gyro_log();
    accelmatrix=handles.m1.accel_log();


    anglematrixdeg=rad2deg(quat2eul(handles.m1.quat_log));
    difmatrix1s=ones(length(anglematrixdeg),3);
    difmatrix1s(:,1)=difmatrix1s(:,1)*anglematrixdeg(1,1);
    difmatrix1s(:,2)=difmatrix1s(:,2)*anglematrixdeg(1,2);
    difmatrix1s(:,3)=difmatrix1s(:,3)*anglematrixdeg(1,3);
    anglematrixdegrelative=anglematrixdeg-difmatrix1s;


    % CUT and save EMGTİME AND EMGDATA in a matrix 50Hz instead of 200Hz

    i=1;
    k=1;

    emgtimevcut=-99.*ones(length(timevectorimu),1);
    emgmcut=-99.*ones(length(timevectorimu),8);


    while (i<=length(timevectorimu) && k<=length(timevectoremg))

        if (abs(timevectorimu(i)-timevectoremg(k))<0.000001)
            emgtimevcut(i)=timevectorimu(i);
            emgmcut(i,:)=emgmatrix(k,:); %
        else
            k=k+1;
            continue;
        end

        i=i+1;
    end


    if(emgtimevcut(end)==-99)

    indexes=find(emgtimevcut==-99)


    emgtimevcut(indexes,:)=[];
    emgmcut(indexes,:)=[];
```

```
    timevectorimu(indexes,:)=[];

    gyromatrix(indexes,:)=[];


    anglex=cumtrapz(timevectorimu,gyromatrix(:,1));
    angley=cumtrapz(timevectorimu,gyromatrix(:,2));
    anglez=cumtrapz(timevectorimu,gyromatrix(:,3));
    anglematrixgyros=[anglex,angley,anglez];

    end



    % Save results to a file

    T =
table(emgtimevcut,emgmcut,timevectorimu,anglematrixgyros,'VariableNames',{'TimeEmgC
ut','EmgCut','TimeIMU','Angles_Gyro'});
    filename = strcat(get(handles.edit7,'String'),'.xlsx');
    % Write table to file
    writetable(T,filename)
    % Print confirmation to command line
    fprintf('Results table with %g emg measurements saved to file
%s\n',length(timevectorimu),filename)

end

    handles.mm.delete();
    clear handles.mm;


% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)

% --- Executes on button press in checkbox3.
function checkbox3_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox4.
function checkbox4_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox5.
function checkbox5_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox6.
function checkbox6_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox7.
function checkbox7_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox8.
function checkbox8_Callback(hObject, eventdata, handles)


function edit2_Callback(hObject, eventdata, handles)

function edit2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end


% --- Executes on button press in checkbox9.
function checkbox9_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox10.
function checkbox10_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox11.
function checkbox11_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox12.
function checkbox12_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox13.
function checkbox13_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox14.
function checkbox14_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox15.
function checkbox15_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox16.
function checkbox16_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox17.
function checkbox17_Callback(hObject, eventdata, handles)



function edit5_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)

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


% --- Executes on button press in checkbox20.
function checkbox20_Callback(hObject, eventdata, handles)


% --- Executes on button press in checkbox21.
function checkbox21_Callback(hObject, eventdata, handles)



function edit7_Callback(hObject, eventdata, handles)



% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
```

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


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

       numberofmarks=str2num(get (handles.MarkText,'String'));

       a=handles.m1.timeEMG_log;
       b=ones(length(a),1);
       timeemg = a-a(1).*b;

       handles.marksemgtime(numberofmarks+1)=timeemg(end);
       guidata(hObject, handles);

       stringcounter=num2str(numberofmarks+1);
       set (handles.MarkText,'String',stringcounter);
       transfervector=handles.marksemgtime;
       setappdata(0,'marks',transfervector);


       set(handles.MTV,'String',strcat(get(handles.MTV,'String'),strcat(':',num2str(
       timeemg(end)))));



function MarkText_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function MarkText_CreateFcn(hObject, eventdata, handles)

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


function MTV_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function MTV_CreateFcn(hObject, eventdata, handles)

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

Appendix C - Signal Processing Code 1

```
clear all;
close all;
clc;

data = xlsread('21ALT.xlsx','A1:N10021');
dataUK =xlsread('99ust.xlsx','A1:N40021');

isMarkedColumn=data(:,14);
MarkedIndexes=find(isMarkedColumn==1);

isMarkedColumnUK=dataUK(:,14);.
MarkedIndexesUK=find(isMarkedColumnUK==1);

datalength=MarkedIndexes(2)-MarkedIndexes(1)+1;

for i=1:1:(length(MarkedIndexes)-1)

    a = MarkedIndexes(i+1)-MarkedIndexes(i)+1;

    if a>datalength
        datalength = a;
    end

end

datalengthUK=MarkedIndexesUK(2)-MarkedIndexesUK(1)+1;

for i=1:1:(length(MarkedIndexesUK)-1)

    a = MarkedIndexesUK(i+1)-MarkedIndexesUK(i)+1;
    if a>datalengthUK
       datalengthUK = a;
    end
end


MovementData = -99*ones(datalength,length(MarkedIndexes)-1);
EMGDataUK =  -99*ones(datalengthUK,8*(length(MarkedIndexesUK)-1));


for i=1:1:(length(MarkedIndexes)-1)
    SingleMovement= data (MarkedIndexes(i):MarkedIndexes(i+1),12);
    SingleMovementEMG = data (MarkedIndexes(i):MarkedIndexes(i+1),2:9);
    MovementData(1:length(SingleMovement),i) = SingleMovement;
end



for i=1:1:(length(MarkedIndexesUK)-1)

    SingleMovementEMGUK = dataUK (MarkedIndexesUK(i):MarkedIndexesUK(i+1),2:9);
    if(i==1)
       startEMGindexUK=i;
    end

EMGDataUK(1:length(SingleMovementEMGUK),startEMGindexUK:startEMGindexUK+7) =
SingleMovementEMGUK;
startEMGindexUK=startEMGindexUK+8;

end
```

```
UpMovesData = MovementData(:,1:2:end);
DownMovesData = MovementData(:,2:2:end);



        [mUK,nUK] = size(EMGDataUK);

        oUK=1;
        jUK=1;
        kUK=1;

        for i=1:8:nUK

            if (rem(oUK,2)==1)
            UpMovesEMGUK(:,jUK:jUK+7) = EMGDataUK(:,i:i+7);
            jUK=jUK+8;
            else
            DownMovesEMGUK(:,kUK:kUK+7) = EMGDataUK(:,i:i+7);
            kUK=kUK+8;
            end
            oUK=oUK+1;

        end

%TAKE ABSOLUTE OF EMG DATA

        UpMovesEMGUK=abs(UpMovesEMGUK);
        DownMovesEMGUK=abs(DownMovesEMGUK);


UpMovesData;    %datalength x n  matrix  n=movement cycle  15
DownMovesData;  %datalength x n  matrix  n=movement cycle 15

UpMovesEMGUK;     %datalength x 8n  matrix  n=movement cycle 15
DownMovesEMGUK;   %datalength x 8n  matrix  n=movement cycle 15


%% Resize with interpolation. Different sample time: 15ms,20ms,10ms

[rowsu,columnsu] = size(UpMovesData);
[rowsd,columnsd] = size(UpMovesData);

[rowsueUK,columnsueUK] = size(UpMovesEMGUK);
[rowsdeUK,columnsdeUK] = size(DownMovesEMGUK);

NewUpMovesData=-99*ones(101,columnsu);
NewDownMovesData=-99*ones(101,columnsd);

NewUpMovesEMGUK = -99*ones(101,columnsueUK);
NewDownMovesEMGUK = -99*ones(101,columnsdeUK);

for i=1:1:columnsu

    indexes99=find(UpMovesData(:,i)==-99);
    UpMove=UpMovesData(:,i);
    UpMove(indexes99)=[];
    y=UpMove;

    lastdatatime=0.005*(length(UpMove)-1);
    t=0:0.005:lastdatatime;

    newsamplingtime=lastdatatime/100;

    newt=0:newsamplingtime:lastdatatime;
    intUpMove1 = interp1(t, y, newt);
    ynew=intUpMove1';

    NewUpMovesData(:,i)=ynew;
```

```
end



for i=1:1:columnsd

    indexes99=find(DownMovesData(:,i)==-99);
    DownMove=DownMovesData(:,i);
    DownMove(indexes99)=[];
    y=DownMove;

    lastdatatime=0.005*(length(DownMove)-1);
    t=0:0.005:lastdatatime;

    newsamplingtime=lastdatatime/100

    newt=0:newsamplingtime:lastdatatime;
    intDownMove1 = interp1(t, y, newt);
    ynew=intDownMove1';

    NewDownMovesData(:,i)=ynew;


end



for i=1:1:columnsueUK

     indexes99=find(UpMovesEMGUK(:,i)==99);
     UpMoveEMGUK=UpMovesEMGUK(:,i);
     UpMoveEMGUK(indexes99)=[];
     y=UpMoveEMGUK;
     t=0:0.005:0.005*(length(UpMoveEMGUK)-1);

     lastdatatime=0.005*(length(UpMoveEMGUK)-1);
     newsamplingtime=lastdatatime/100;

     newt=0:newsamplingtime:lastdatatime;
     intUpMoveEMGUK = interp1(t, y, newt);
     ynew=intUpMoveEMGUK;
     NewUpMovesEMGUK(:,i)=ynew;
end



 for i=1:1:columnsdeUK
     indexes99=find(DownMovesEMGUK(:,i)==99);
     DownMoveEMGUK=DownMovesEMGUK(:,i);
     DownMoveEMGUK(indexes99)=[];
     y=DownMoveEMGUK;
     t=0:0.005:0.005*(length(DownMoveEMGUK)-1);

     lastdatatime=0.005*(length(DownMoveEMGUK)-1);
     newsamplingtime=lastdatatime/100;

     newt=0:newsamplingtime:lastdatatime;
     intDownMoveEMGUK = interp1(t, y, newt);
     ynew=intDownMoveEMGUK;
     NewDownMovesEMGUK(:,i)=ynew;

 end

NewUpMovesEMGUK;
NewDownMovesEMGUK;
```

```
[m,n]=size(NewUpMovesEMGUK);
integratedEMGUK = zeros(m,n);
t=0:(1.5/100):1.5;

for i=1:1:
    windowSize = 5;
    b = (1/windowSize)*ones(1,windowSize);
    a = 1;

    y = filter(b,a,NewUpMovesEMGUK(:,i)); %1. filtering

    windowSize = 30;
    b = (1/windowSize)*ones(1,windowSize);
    a = 1;

    integratedEMGUK(:,i)=cumtrapz(t,y);
    y = integratedEMGUK(:,i);

    Interpolated_ten_rows_NewUpMovesEMGUK(:,i) = y
end


figure
tnew=1:1:101;
plot(tnew,NewUpMovesEMGUK(:,1),tnew,Interpolated_ten_rows_NewUpMovesEMGUK(:,1),'*')
title('Absolute EMG - Max veya Mean alınmış 10 datalık emg - Interpolated 101 data
emg')

Interpolated_ten_rows_NewUpMovesEMGUK;
[rows,columns]=size(NewUpMovesData);


internal_emg_1 = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_2 = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_3 = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_4 = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_5 = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_6 = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_7 = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_8 = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);

figure

for j = 1 : 1: rows  %101

    for i = 1 : 1 : length(Interpolated_ten_rows_NewUpMovesEMGUK(1,:))/8 %49


        internal_emg_1(i,j) = Interpolated_ten_rows_NewUpMovesEMGUK(j,1+(i-1)*8);
        internal_emg_2(i,j) = Interpolated_ten_rows_NewUpMovesEMGUK(j,2+(i-1)*8);

        internal_emg_3(i,j) = Interpolated_ten_rows_NewUpMovesEMGUK(j,3+(i-1)*8);
        internal_emg_4(i,j) = Interpolated_ten_rows_NewUpMovesEMGUK(j,4+(i-1)*8);
        internal_emg_5(i,j) = Interpolated_ten_rows_NewUpMovesEMGUK(j,5+(i-1)*8);
        internal_emg_6(i,j) = Interpolated_ten_rows_NewUpMovesEMGUK(j,6+(i-1)*8);
        internal_emg_7(i,j) = Interpolated_ten_rows_NewUpMovesEMGUK(j,7+(i-1)*8);
        internal_emg_8(i,j) = Interpolated_ten_rows_NewUpMovesEMGUK(j,8+(i-1)*8);

    end

    hold on
```

```
    if j==1
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'b+')
    end
    if j==10
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'ko')
    end
    if j==20
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'ro')
    end
    if j==30
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'go')
    end
    if j==40
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'yo')
    end
    if j==50
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'co')
    end
    if j==60
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'mo')
    end
    if j==70
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'b*')
    end
    if j==80
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'k*')
    end
    if j==90
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'r*')
    end
    if j==101
    plot3(internal_emg_4(:,j),internal_emg_8(:,j),internal_emg_7(:,j),'g*')
    end
    title('EMG4 VS EMG8 FOR 50 UPMOVE TESTS')
    xlabel('EMG4 DATA')
    ylabel('EMG8 DATA')
    zlabel('EMG7 DATA')

end


figure
p=plot(Interpolated_ten_rows_NewUpMovesEMGUK(:,1:8));
title('UpMove1 EMG Data')
xlabel('TIME [s]')
ylabel('EMG')
legend('EMG1','EMG2','EMG3','EMG4','EMG5','EMG6','EMG7','EMG8')
p1.LineWidth = 3;
figure
plot(Interpolated_ten_rows_NewUpMovesEMGUK(:,9:16))
title('UpMove2 EMG Data')
legend
figure
plot(Interpolated_ten_rows_NewUpMovesEMGUK(:,17:24))
title('UpMove3 EMG Data')
legend
figure
plot(Interpolated_ten_rows_NewUpMovesEMGUK(:,25:32))
title('UpMove4 EMG Data')
legend




[m,n]=size(internal_emg_1);
c1to101_emg1to8_tr=ones(m,8*101);
```

```
j=1;
for i=1:1:101

        c1to101_emg1to8_tr(:,j)=internal_emg_1(:,i); % emg1
        c1to101_emg1to8_tr(:,j+1)=internal_emg_2(:,i); %emg2
        c1to101_emg1to8_tr(:,j+2)=internal_emg_3(:,i); %emg3
        c1to101_emg1to8_tr(:,j+3)=internal_emg_4(:,i)'; %emg4
        c1to101_emg1to8_tr(:,j+4)=internal_emg_5(:,i); %emg5
        c1to101_emg1to8_tr(:,j+5)=internal_emg_6(:,i); %emg6
        c1to101_emg1to8_tr(:,j+6)=internal_emg_7(:,i); %emg7
        c1to101_emg1to8_tr(:,j+7)=internal_emg_8(:,i); %emg8

        j=j+8;

end


c1_x1_tr=c1to101_emg1to8_tr(:,4); %% emg 4 corresponding to class 1
c1_x2_tr=c1to101_emg1to8_tr(:,5); %% emg 5 corresponding to class 1

c2_x1_tr=c1to101_emg1to8_tr(:,76); %% emg 4 corresponding to class 10
c2_x2_tr=c1to101_emg1to8_tr(:,77); %% emg 5 corresponding to class 10

c3_x1_tr=c1to101_emg1to8_tr(:,156);
c3_x2_tr=c1to101_emg1to8_tr(:,157);

c4_x1_tr=c1to101_emg1to8_tr(:,236);
c4_x2_tr=c1to101_emg1to8_tr(:,237);

c5_x1_tr=c1to101_emg1to8_tr(:,316);
c5_x2_tr=c1to101_emg1to8_tr(:,317);

c6_x1_tr=c1to101_emg1to8_tr(:,396);
c6_x2_tr=c1to101_emg1to8_tr(:,397);

c7_x1_tr=c1to101_emg1to8_tr(:,476);
c7_x2_tr=c1to101_emg1to8_tr(:,477);

c8_x1_tr=c1to101_emg1to8_tr(:,556);
c8_x2_tr=c1to101_emg1to8_tr(:,557);

c9_x1_tr=c1to101_emg1to8_tr(:,636);
c9_x2_tr=c1to101_emg1to8_tr(:,637);

c10_x1_tr=c1to101_emg1to8_tr(:,716);
c10_x2_tr=c1to101_emg1to8_tr(:,717);

c11_x1_tr=c1to101_emg1to8_tr(:,804); %% emg 4 corresponding to class 101
c11_x2_tr=c1to101_emg1to8_tr(:,805); %% emg 5 corresponding to class 101

figure

plot(c1_x1_tr,c1_x2_tr,'b+');
hold on
plot(c2_x1_tr,c2_x2_tr,'ko');
plot(c3_x1_tr,c3_x2_tr,'ro');
plot(c4_x1_tr,c4_x2_tr,'go');
plot(c5_x1_tr,c5_x2_tr,'yo');
plot(c6_x1_tr,c6_x2_tr,'co');
plot(c7_x1_tr,c7_x2_tr,'mo');
plot(c8_x1_tr,c8_x2_tr,'b*');
plot(c9_x1_tr,c9_x2_tr,'k*');
plot(c10_x1_tr,c10_x2_tr,'r*');
plot(c11_x1_tr,c11_x2_tr,'g*');
legend('Class1','Class10','Class20','Class30','Class40','Class50','Class60','Class7
0','Class80','Class90','Class101')
xlabel('EMG4')
ylabel('EMG5')
title('EMG4 VS EMG5 FOR 50 UPMOVE TESTS')
```

Appendix D - Signal Processing Code 2

```
clear all;
close all;
clc;


dataUK =xlsread('Flexion21.xlsx','A1:N14013'); %%Flexion!

dataEX =xlsread('Extension21.xlsx','A1:N16033'); %%Extension!

dataRX =xlsread('Relax21.xlsx','A1:N14029'); %%Relax!

dataFS =xlsread('Fist21.xlsx','A1:N14017'); %%Relax!


isMarkedColumnUK=dataUK(:,14); %
MarkedIndexesUK=find(isMarkedColumnUK==1);

isMarkedColumnEX=dataEX(:,14);
MarkedIndexesEX=find(isMarkedColumnEX==1);

isMarkedColumnRX=dataRX(:,14);
MarkedIndexesRX=find(isMarkedColumnRX==1);

isMarkedColumnFS=dataFS(:,14);
MarkedIndexesFS=find(isMarkedColumnFS==1);


datalengthUK=MarkedIndexesUK(2)-MarkedIndexesUK(1)+1;

for i=1:1:(length(MarkedIndexesUK)-1
    a = MarkedIndexesUK(i+1)-MarkedIndexesUK(i)+1;
    if a>datalengthUK
        datalengthUK = a;
    end
end

datalengthEX=MarkedIndexesEX(2)-MarkedIndexesEX(1)+1;
for i=1:1:(length(MarkedIndexesEX)-1)
    a = MarkedIndexesEX(i+1)-MarkedIndexesEX(i)+1;
    if a>datalengthEX
        datalengthEX = a;
    end
end

datalengthRX=MarkedIndexesRX(2)-MarkedIndexesRX(1)+1;
for i=1:1:(length(MarkedIndexesRX)-1)
    a = MarkedIndexesRX(i+1)-MarkedIndexesRX(i)+1;
    if a>datalengthRX
        datalengthRX = a;
    end
end

datalengthFS=MarkedIndexesFS(2)-MarkedIndexesFS(1)+1;
for i=1:1:(length(MarkedIndexesFS)-1

    a = MarkedIndexesFS(i+1)-MarkedIndexesFS(i)+1;
    if a>datalengthFS
        datalengthFS = a;
    end
end
```

```
EMGDataUK =  -99*ones(datalengthUK,8*(length(MarkedIndexesUK)-1));
EMGDataEX =  -99*ones(datalengthEX,8*(length(MarkedIndexesEX)-1));

EMGDataRX =  -99*ones(datalengthRX,8*(length(MarkedIndexesRX)-1));
EMGDataFS =  -99*ones(datalengthFS,8*(length(MarkedIndexesFS)-1));


for i=1:1:(length(MarkedIndexesUK)-1

    SingleMovementEMGUK = dataUK (MarkedIndexesUK(i):MarkedIndexesUK(i+1),2:9);

    if(i==1)
        startEMGindexUK=i;
    end

    EMGDataUK(1:length(SingleMovementEMGUK),startEMGindexUK:startEMGindexUK+7) =
    SingleMovementEMGUK;

    startEMGindexUK=startEMGindexUK+8;

 end

 for i=1:1:(length(MarkedIndexesEX)-1)
     SingleMovementEMGEX = dataEX (MarkedIndexesEX(i):MarkedIndexesEX(i+1),2:9);
     if(i==1)
        startEMGindexEX=i;
     end

     EMGDataEX(1:length(SingleMovementEMGEX),startEMGindexEX:startEMGindexEX+7) =
     SingleMovementEMGEX;
     startEMGindexEX=startEMGindexEX+8;

 end

 for i=1:1:(length(MarkedIndexesRX)-1)

     SingleMovementEMGRX = dataRX (MarkedIndexesRX(i):MarkedIndexesRX(i+1),2:9);
     if(i==1)
        startEMGindexRX=i;
     end

     EMGDataRX(1:length(SingleMovementEMGRX),startEMGindexRX:startEMGindexRX+7) =
     SingleMovementEMGRX;
     startEMGindexRX=startEMGindexRX+8;

 end

 for i=1:1:(length(MarkedIndexesFS)-1)

     SingleMovementEMGFS = dataFS (MarkedIndexesFS(i):MarkedIndexesFS(i+1),2:9);
     if(i==1)
        startEMGindexFS=i;
     end

     EMGDataFS(1:length(SingleMovementEMGFS),startEMGindexFS:startEMGindexFS+7) =
     SingleMovementEMGFS;
     startEMGindexFS=startEMGindexFS+8;

 end


 [mUK,nUK] = size(EMGDataUK);

 oUK=1;
 jUK=1;
 kUK=1;
```

```
 [mEX,nEX] = size(EMGDataEX);
 oEX=1;
 jEX=1;
 kEX=1;

 [mRX,nRX] = size(EMGDataRX);

oRX=1;
jRX=1;
kRX=1;

[mFS,nFS] = size(EMGDataFS);

oFS=1;
jFS=1;
kFS=1;

for i=1:8:nUK

    if (rem(oUK,2)==1)
        UpMovesEMGUK(:,jUK:jUK+7) = EMGDataUK(:,i:i+7);
        jUK=jUK+8;
    else
        DownMovesEMGUK(:,kUK:kUK+7) = EMGDataUK(:,i:i+7);
        kUK=kUK+8;
    end
        oUK=oUK+1;

end

for i=1:8:nEX

    if (rem(oEX,2)==1)
        UpMovesEMGEX(:,jEX:jEX+7) = EMGDataEX(:,i:i+7);
        jEX=jEX+8;
    else
        DownMovesEMGEX(:,kEX:kEX+7) = EMGDataEX(:,i:i+7);
        kEX=kEX+8;
    end
        oEX=oEX+1;

end

for i=1:8:nRX

    if (rem(oRX,2)==1)
        UpMovesEMGRX(:,jRX:jRX+7) = EMGDataRX(:,i:i+7);
        jRX=jRX+8;
    else
        DownMovesEMGRX(:,kRX:kRX+7) = EMGDataRX(:,i:i+7);
        kRX=kRX+8;
    end
        oRX=oRX+1;

end


for i=1:8:nFS

    if (rem(oFS,2)==1)
        UpMovesEMGFS(:,jFS:jFS+7) = EMGDataFS(:,i:i+7);
        jFS=jFS+8;
    else
        DownMovesEMGFS(:,kFS:kFS+7) = EMGDataFS(:,i:i+7);
        kFS=kFS+8;
    end
        oFS=oFS+1;

end
```

```
 %TAKE ABSOLUTE OF EMG DATA

UpMovesEMGUK=abs(UpMovesEMGUK);
DownMovesEMGUK=abs(DownMovesEMGUK);

 UpMovesEMGEX=abs(UpMovesEMGEX
 DownMovesEMGEX=abs(DownMovesEMGEX);

 UpMovesEMGRX=abs(UpMovesEMGRX);
 DownMovesEMGRX=abs(DownMovesEMGRX);

 UpMovesEMGFS=abs(UpMovesEMGFS);
 DownMovesEMGFS=abs(DownMovesEMGFS);


 UpMovesEMGUK;      %datalength x 8n  matris  n=movement cycle 10
 UpMovesEMGEX;
 UpMovesEMGRX;
 UpMovesEMGFS;


 [rowsueUK,columnsueUK] = size(UpMovesEMGUK);
 NewUpMovesEMGUK = -99*ones(15,columnsueUK);

 [rowsueEX,columnsueEX] = size(UpMovesEMGEX);
 NewUpMovesEMGEX = -99*ones(15,columnsueEX);

 [rowsueRX,columnsueRX] = size(UpMovesEMGRX);
 NewUpMovesEMGRX = -99*ones(15,columnsueRX);

 [rowsueFS,columnsueFS] = size(UpMovesEMGFS);
 NewUpMovesEMGFS = -99*ones(15,columnsueFS);

 for i=1:1:columnsueUK

     indexes99=find(UpMovesEMGUK(:,i)==99);
     UpMoveEMGUK=UpMovesEMGUK(:,i);
     UpMoveEMGUK(indexes99)=[];
     y=UpMoveEMGUK;

     NewUpMovesEMGUK(1,i)=max(y(1:20))
     NewUpMovesEMGUK(2,i)=max(y(21:40))
     NewUpMovesEMGUK(3,i)=max(y(41:60))
     NewUpMovesEMGUK(4,i)=max(y(61:80));
     NewUpMovesEMGUK(5,i)=max(y(81:100));
     NewUpMovesEMGUK(6,i)=max(y(101:120));
     NewUpMovesEMGUK(7,i)=max(y(121:140));
     NewUpMovesEMGUK(8,i)=max(y(141:160));
     NewUpMovesEMGUK(9,i)=max(y(161:180));
     NewUpMovesEMGUK(10,i)=max(y(181:200));
     NewUpMovesEMGUK(11,i)=max(y(201:220));
     NewUpMovesEMGUK(12,i)=max(y(221:240));
     NewUpMovesEMGUK(13,i)=max(y(241:260));
     NewUpMovesEMGUK(14,i)=max(y(261:280));
     NewUpMovesEMGUK(15,i)=max(y(281:300));

 end


  for i=1:1:columnsueEX

     indexes99=find(UpMovesEMGEX(:,i)==99);
     UpMoveEMGEX=UpMovesEMGEX(:,i);
     UpMoveEMGEX(indexes99)=[];
     y=UpMoveEMGEX;

     NewUpMovesEMGEX(1,i)=max(y(1:20));
     NewUpMovesEMGEX(2,i)=max(y(21:40));
     NewUpMovesEMGEX(3,i)=max(y(41:60));
```

```
      NewUpMovesEMGEX(4,i)=max(y(61:80));
      NewUpMovesEMGEX(5,i)=max(y(81:100));
      NewUpMovesEMGEX(6,i)=max(y(101:120));
      NewUpMovesEMGEX(7,i)=max(y(121:140));
      NewUpMovesEMGEX(8,i)=max(y(141:160));
      NewUpMovesEMGEX(9,i)=max(y(161:180));
    NewUpMovesEMGEX(10,i)=max(y(181:200));
    NewUpMovesEMGEX(11,i)=max(y(201:220));
    NewUpMovesEMGEX(12,i)=max(y(221:240));
    NewUpMovesEMGEX(13,i)=max(y(241:260));
    NewUpMovesEMGEX(14,i)=max(y(261:280));
    NewUpMovesEMGEX(15,i)=max(y(281:300));

 end


for i=1:1:columnsueRX

    indexes99=find(UpMovesEMGRX(:,i)==99);
    UpMoveEMGRX=UpMovesEMGRX(:,i);
    UpMoveEMGRX(indexes99)=[];
    y=UpMoveEMGRX;

    NewUpMovesEMGRX(1,i)=max(y(1:20));
    NewUpMovesEMGRX(2,i)=max(y(21:40));
    NewUpMovesEMGRX(3,i)=max(y(41:60));
    NewUpMovesEMGRX(4,i)=max(y(61:80));
    NewUpMovesEMGRX(5,i)=max(y(81:100));
    NewUpMovesEMGRX(6,i)=max(y(101:120));
    NewUpMovesEMGRX(7,i)=max(y(121:140));
    NewUpMovesEMGRX(8,i)=max(y(141:160));
    NewUpMovesEMGRX(9,i)=max(y(161:180));
    NewUpMovesEMGRX(10,i)=max(y(181:200));
    NewUpMovesEMGRX(11,i)=max(y(201:220));
    NewUpMovesEMGRX(12,i)=max(y(221:240));
    NewUpMovesEMGRX(13,i)=max(y(241:260));
    NewUpMovesEMGRX(14,i)=max(y(261:280));
    NewUpMovesEMGRX(15,i)=max(y(281:300));

end



for i=1:1:columnsueFS

    indexes99=find(UpMovesEMGFS(:,i)==99);
    UpMoveEMGFS=UpMovesEMGFS(:,i);
    UpMoveEMGFS(indexes99)=[];
    y=UpMoveEMGFS;

    NewUpMovesEMGFS(1,i)=max(y(1:20))
    NewUpMovesEMGFS(2,i)=max(y(21:40))
    NewUpMovesEMGFS(3,i)=max(y(41:60))
    NewUpMovesEMGFS(4,i)=max(y(61:80));
    NewUpMovesEMGFS(5,i)=max(y(81:100));
    NewUpMovesEMGFS(6,i)=max(y(101:120));
    NewUpMovesEMGFS(7,i)=max(y(121:140));
    NewUpMovesEMGFS(8,i)=max(y(141:160));
    NewUpMovesEMGFS(9,i)=max(y(161:180));
    NewUpMovesEMGFS(10,i)=max(y(181:200));
    NewUpMovesEMGFS(11,i)=max(y(201:220));
    NewUpMovesEMGFS(12,i)=max(y(221:240));
    NewUpMovesEMGFS(13,i)=max(y(241:260));
    NewUpMovesEMGFS(14,i)=max(y(261:280));
    NewUpMovesEMGFS(15,i)=max(y(281:300));

end
```

```
NewUpMovesEMGUK;
NewUpMovesEMGEX;
NewUpMovesEMGRX;
NewUpMovesEMGFS;




[rows,columns]=size(NewUpMovesEMGUK); %rows=15

internal_emg_1FX = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_2FX = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_3FX = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_4FX = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_5FX = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_6FX = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_7FX = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);
internal_emg_8FX = zeros(length(NewUpMovesEMGUK(1,:))/8,rows);


for j = 1 : 1: rows  %15

    for i = 1 : 1 : length(NewUpMovesEMGUK(1,:))/8 %49

        internal_emg_1FX(i,j) = NewUpMovesEMGUK(j,1+(i-1)*8);
        internal_emg_2FX(i,j) = NewUpMovesEMGUK(j,2+(i-1)*8);
        internal_emg_3FX(i,j) = NewUpMovesEMGUK(j,3+(i-1)*8);
        internal_emg_4FX(i,j) = NewUpMovesEMGUK(j,4+(i-1)*8);
        internal_emg_5FX(i,j) = NewUpMovesEMGUK(j,5+(i-1)*8);
        internal_emg_6FX(i,j) = NewUpMovesEMGUK(j,6+(i-1)*8);
        internal_emg_7FX(i,j) = NewUpMovesEMGUK(j,7+(i-1)*8);
        internal_emg_8FX(i,j) = NewUpMovesEMGUK(j,8+(i-1)*8);

    end
end


internal_emg_1EX = zeros(length(NewUpMovesEMGEX(1,:))/8,rows);
internal_emg_2EX = zeros(length(NewUpMovesEMGEX(1,:))/8,rows);
internal_emg_3EX = zeros(length(NewUpMovesEMGEX(1,:))/8,rows);
internal_emg_4EX = zeros(length(NewUpMovesEMGEX(1,:))/8,rows);
internal_emg_5EX = zeros(length(NewUpMovesEMGEX(1,:))/8,rows);
internal_emg_6EX = zeros(length(NewUpMovesEMGEX(1,:))/8,rows);
internal_emg_7EX = zeros(length(NewUpMovesEMGEX(1,:))/8,rows);
internal_emg_8EX = zeros(length(NewUpMovesEMGEX(1,:))/8,rows);


for j = 1 : 1: rows  %15

    for i = 1 : 1 : length(NewUpMovesEMGUK(1,:))/8 %49

        internal_emg_1EX(i,j) = NewUpMovesEMGEX(j,1+(i-1)*8);
        internal_emg_2EX(i,j) = NewUpMovesEMGEX(j,2+(i-1)*8);
        internal_emg_3EX(i,j) = NewUpMovesEMGEX(j,3+(i-1)*8);
        internal_emg_4EX(i,j) = NewUpMovesEMGEX(j,4+(i-1)*8);
        internal_emg_5EX(i,j) = NewUpMovesEMGEX(j,5+(i-1)*8);
        internal_emg_6EX(i,j) = NewUpMovesEMGEX(j,6+(i-1)*8);
        internal_emg_7EX(i,j) = NewUpMovesEMGEX(j,7+(i-1)*8);
        internal_emg_8EX(i,j) = NewUpMovesEMGEX(j,8+(i-1)*8);

    end
end


internal_emg_1RX = zeros(length(NewUpMovesEMGRX(1,:))/8,rows);
internal_emg_2RX = zeros(length(NewUpMovesEMGRX(1,:))/8,rows);
internal_emg_3RX = zeros(length(NewUpMovesEMGRX(1,:))/8,rows);
```

```
internal_emg_4RX = zeros(length(NewUpMovesEMGRX(1,:))/8,rows);
internal_emg_5RX = zeros(length(NewUpMovesEMGRX(1,:))/8,rows);
internal_emg_6RX = zeros(length(NewUpMovesEMGRX(1,:))/8,rows);
internal_emg_7RX = zeros(length(NewUpMovesEMGRX(1,:))/8,rows);
internal_emg_8RX = zeros(length(NewUpMovesEMGRX(1,:))/8,rows);



 for j = 1 : 1: rows  %15

     for i = 1 : 1 : length(NewUpMovesEMGRX(1,:))/8 %49

         internal_emg_1RX(i,j) = NewUpMovesEMGRX(j,1+(i-1)*8);
         internal_emg_2RX(i,j) = NewUpMovesEMGRX(j,2+(i-1)*8);
         internal_emg_3RX(i,j) = NewUpMovesEMGRX(j,3+(i-1)*8);
         internal_emg_4RX(i,j) = NewUpMovesEMGRX(j,4+(i-1)*8);
         internal_emg_5RX(i,j) = NewUpMovesEMGRX(j,5+(i-1)*8);
         internal_emg_6RX(i,j) = NewUpMovesEMGRX(j,6+(i-1)*8);
         internal_emg_7RX(i,j) = NewUpMovesEMGRX(j,7+(i-1)*8);
         internal_emg_8RX(i,j) = NewUpMovesEMGRX(j,8+(i-1)*8);

     end
 end



 internal_emg_1FS = zeros(length(NewUpMovesEMGFS(1,:))/8,rows);
 internal_emg_2FS = zeros(length(NewUpMovesEMGFS(1,:))/8,rows);
 internal_emg_3FS = zeros(length(NewUpMovesEMGFS(1,:))/8,rows);
 internal_emg_4FS = zeros(length(NewUpMovesEMGFS(1,:))/8,rows);
 internal_emg_5FS = zeros(length(NewUpMovesEMGFS(1,:))/8,rows);
 internal_emg_6FS = zeros(length(NewUpMovesEMGFS(1,:))/8,rows);
 internal_emg_7FS = zeros(length(NewUpMovesEMGFS(1,:))/8,rows);
 internal_emg_8FS = zeros(length(NewUpMovesEMGFS(1,:))/8,rows);



 for j = 1 : 1: rows  %15

     for i = 1 : 1 : length(NewUpMovesEMGFS(1,:))/8 %49

         internal_emg_1FS(i,j) = NewUpMovesEMGFS(j,1+(i-1)*8);
         internal_emg_2FS(i,j) = NewUpMovesEMGFS(j,2+(i-1)*8);
         internal_emg_3FS(i,j) = NewUpMovesEMGFS(j,3+(i-1)*8);
         internal_emg_4FS(i,j) = NewUpMovesEMGFS(j,4+(i-1)*8);
         internal_emg_5FS(i,j) = NewUpMovesEMGFS(j,5+(i-1)*8);
         internal_emg_6FS(i,j) = NewUpMovesEMGFS(j,6+(i-1)*8);
         internal_emg_7FS(i,j) = NewUpMovesEMGFS(j,7+(i-1)*8);
         internal_emg_8FS(i,j) = NewUpMovesEMGFS(j,8+(i-1)*8);

     end
 end

 Figure

 for i=1:1:10
 plot(internal_emg_2FX(i,:),internal_emg_7FX(i,:),'*r')
 hold on
 plot(internal_emg_2EX(i,:),internal_emg_7EX(i,:),'ob')
 plot(internal_emg_2FS(i,:),internal_emg_7FS(i,:),'sm')
 plot(internal_emg_2RX(i,:),internal_emg_7RX(i,:),'vk')
 end

 title('FLEXION EXTENSION AND DO NOTHING (150x3=450 data)')
 xlabel('EMG 2');
 ylabel('EMG 7');

 legend('FLEXION','EXTENSION','FIST','DO NOTHING');
```

```
grid on


classfxx1=internal_emg_2FX';
classfxx2=internal_emg_7FX';

classexx1=internal_emg_2EX';
classexx2=internal_emg_7EX';

classfsx1=internal_emg_2FS';
classfsx2=internal_emg_7FS';

classrxx1=internal_emg_2RX';
classrxx2=internal_emg_7RX';


classFXEMG2=classfxx1(:); %column vector
classFXEMG7=classfxx2(:);

classEXEMG2=classexx1(:); %column vector
classEXEMG7=classexx2(:);

classFSEMG2=classfsx1(:); %column vector
classFSEMG7=classfsx2(:);

classRXEMG2=classrxx1(:); %column vector
classRXEMG7=classrxx2(:);


a=[classFXEMG2;classFSEMG2;classRXEMG2;classEXEMG2];
b=[classFXEMG7;classFSEMG7;classRXEMG7;classEXEMG7];

Y = ones(600,2);
Y(1:150,1:2)=0;
Y(151:300,1)=0;
Y(301:450,2)=0;


trainingdata=[a,b,Y];

figure
plot(trainingdata(1:150,1),trainingdata(1:150,2),'*r')
hold on
plot(trainingdata(151:300,1),trainingdata(151:300,2),'*b')
plot(trainingdata(301:450,1),trainingdata(301:450,2),'*k')
plot(trainingdata(451:600,1),trainingdata(451:600,2),'*m')
legend('FLEXION','FIST','RELAX','EXTENSION')
```

Appendix E - Neural Network Training Algorithm

```
%% User input
number_of_hidden_layer_node = 2;
number_of_output_layer_node = 2;
H = 4;
K = 2;
I = 3;
%%% CLASSIFICATION Problem
%% Backpropagation Algorithm

%% Writing the dataset on the graph
class3_tr=trainingdata;
variable = class3_tr;
```

```
length_of_class = length(variable(:,1))/4;

figure
plot(variable(1:length_of_class,1),variable(1:length_of_class,2),'+')
hold on
plot(variable(length_of_class+1:2*length_of_class,1),variable(length_of_class+1:2*l
ength_of_class,2),'o')
hold on
plot(variable(2*length_of_class+1:3*length_of_class,1),variable(2*length_of_class+1
:3*length_of_class,2),'*')
hold on
plot(variable(3*length_of_class+1:4*length_of_class,1),variable(3*length_of_class+1
:4*length_of_class,2),'x')

% Denotes Number of Training Examples
%N is for the FOR cycle
N = length(class3_tr);

% We define the error as N elemented series
error = 1;
error_func = ones(N,1);
error = sum(error_func);

%% Input values are arranged.
X0 = ones(4*length_of_class,1);                 % for bias input
X1 = variable(:,1);                             % for first input
X2 = variable(:,2);                             % for second input

Y_general = variable(:,3:4)';                    % target values

X_general_unnormalized = [X0 X1 X2];
X_general = [X0 mat2gray(X1) mat2gray(X2)]';     % normalization in order to
provide better convergence

%parameters are arranged before calculations
%Weights for our system will be defined through using randi

epsilon = 0.00001;
learning_rate = 1;
alfa = 0.5;

error_previous = 1;

% coefficient initialization

%% (1) define parametric coefficients, the reason is that user can change the
dimensions of the layers,
%% (2) paving the way for creating for loops
W = randi([5 10],H,I)./100;
W_previous = randi([5 10],H,I)./100;
V = randi([5 10],K,H)./100;
V_previous = randi([5 10],K,H)./100;

%% (1) delta (d_W, d_V) and its real values (W,V) must be the same dimesions to
each other
d_W = zeros(H,I);
d_V = zeros(K,H);

% temp variables including model values

%% (1) sigmoidal function and the total output must be arranged properly with the
outputs of them:
%% z -> sigmoid function, so H output are produced. N - training set
z = zeros(H,N);
%% y -> total function, so K output are produced. N - training set
y = zeros(K,N);

figure
iteration = 0;      % iteration number obtained
ax1 = subplot(3,1,1);
```

```
h1 = animatedline;   % iteration graph is produced with animation feature
ax2 = subplot(3,1,2);
h2 = animatedline;
ax3 = subplot(3,1,3);
h3 = animatedline;

%% comparing with the total error
while error > epsilon
    iteration = iteration + 1;
    %% initialize all delta values after the training session
    d_W = zeros(H,I);
    d_V = zeros(K,H);

    % training session is started
    for i = 1 : 1 : N

        % z value calculation

        for h = 1 : 1 : H

            z(h,i) = 1/(1 + exp(-W(h,:)*X_general(:,i)));
        end

        % y value calculation

        for k = 1 : 1 : K

             y(k,i) = 1/(1 + exp(-V(k,:)*z(:,i)));
        end

        % delta v coefficient calculation

        %% d_V() = zeros(K,H);
        %% first k, second h loop are established
        for k = 1 : 1 : K

            for h = 1 : 1 : H

                d_V(k,h) = d_V(k,h) + learning_rate*(Y_general(k,i) -
y(k,i))*z(h,i);

            end

        end

        % delta w coefficient calculation

        %% d_W() = zeros(H,I);
        %% first h, second in loop are established

        for h = 1 : 1 : H

            for in = 1 : 1 : I

                d_W(h,in) = d_W(h,in) + learning_rate*(Y_general(:,i) -
y(:,i))'*V(:,h)*z(h,i)*(1 - z(h,i))*X_general(in,i);

            end

        end

        error_func(i,1) = sum(1/2*(Y_general(:,i) - y(:,i)).^2);

    end

    % we have to update the real weights(V,W) after the whole training process


    error = sum(error_func)/N
```

```
    V = V + d_V/N;
    W = W + d_W/N;

    W_previous = d_W;
    V_previous = d_V;



end

figure
Y_plot = zeros(1,330);

Y_plot(1:150) = 0;
Y_plot(151:300) = 1;
Y_plot(301:450) = 2;
Y_plot(451:600) = 3;

plot(Y_plot)

y_model_plot = 2*y(1,:) + 1*y(2,:);

hold on
plot(y_model_plot)
```

Appendix F - Motor Driving Codes

```
function varargout = MotorDriveGUI(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MotorDriveGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @MotorDriveGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end


% --- Executes just before MotorDriveGUI is made visible.
function MotorDriveGUI_OpeningFcn(hObject, eventdata, handles, varargin)
clc
countMyos = 1;
handles.mm = MyoMex(countMyos);
handles.m1 = handles.mm.myoData(1);


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

% Update handles structure
guidata(hObject, handles);


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

% Get default command line output from handles structure
```

```
varargout{1} = handles.output;


function edit1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)

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


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

% Load Libraries
if ~libisloaded('dxl_x64_c')
    [notfound, warnings] = loadlibrary('dxl_x64_c', 'dynamixel_sdk.h', 'addheader',
'port_handler.h', 'addheader', 'packet_handler.h', 'addheader',
'group_bulk_read.h', 'addheader', 'group_bulk_write.h');
end


hip = [33.0020599365234
32.9961166381835
32.9022865295410
32.7251701354980
32.4885025024414
32.2425765991210
31.9790401458740
31.6379680633544
31.1666221618652
30.5649471282958
29.8566493988037
29.0381107330322
28.1263275146484
27.1368141174316
26.1073455810546
25.0718631744384
24.0372943878173
22.9979515075683
21.9433460235595
20.8605537414550
19.7469654083251
18.5935440063476
17.4081439971923
16.1928977966308
14.9482555389404
13.6897535324096
12.4322538375854
11.1794776916503
9.9406013488770
8.7219266891479
7.5251364707947
6.3478674888611
5.1944661140442
4.0714416503906
2.9829537868500
1.9266539812088
0.9017928242683
-0.0949064493179
-1.0637298822403
-2.0000596046448
-2.8958532810211
-3.7348558902740
-4.5061912536621
-5.2052955627441
```

```
-5.8269972801208
-6.3732309341431
-6.8485932350159
-7.2627344131470
-7.6247291564941
-7.9376230239868
-8.1883201599121
-8.3246707916260
-8.3151283264160
-8.1381597518921
-7.8028392791748
-7.3256626129150
-6.6989431381226
-5.9068794250488
-4.9218425750732
-3.7272205352783
-2.3101551532745
-0.6890018582344
1.0933736562729
3.0001778602600
4.9658031463623
6.9535708427429
8.9540643692017
10.9481077194213
12.9302864074707
14.8752241134643
16.7498912811279
18.5277938842773
20.2187404632568
21.8537902832031
23.4282608032226
24.9365539550781
26.3409023284912
27.6500053405761
28.8645896911621
29.9674835205078
30.9468708038330
31.7855224609375
32.4818572998046
33.0246810913085
33.4131469726562
33.6506233215332
33.7299385070800
33.6605300903320
33.4512252807617
33.1463737487792
32.7834167480468
32.3931007385253
32.0288085937500
31.7320175170898
31.5344944000244
31.4407844543457
31.4482364654541
31.5487995147705
31.7119159698486
31.8889904022216
33.0020599365234];



knee = [7.94416189193725
9.32002353668212
10.26949787139890
10.98186588287350
11.85558986663810
13.14454650878900
14.72700881958000
16.23527526855460
17.43891143798820
```

18.32238578796380
18.95714569091790
19.32979011535640
19.43752479553220
19.28828430175780
18.95792388916010
18.54344940185540
18.09063529968260
17.63248443603510
17.16906547546380
16.69008636474600
16.19159889221190
15.66807746887200
15.12835025787350
14.57038879394530
13.99144840240470
13.39033794403070
12.76707172393790
12.13257217407220
11.49122142791740
10.85100078582760
10.22210502624510
9.61800193786621
9.05775928497314
8.56165218353271
8.14449882507324
7.80518579483032
7.54060602188110
7.34648084640502
7.22678184509277
7.19422483444213
7.25126838684082
7.41063356399536
7.68543815612792
8.08269500732421
8.61239624023437
9.26546382904052
10.02891159057610
10.89607238769530
11.86620330810540
12.93995189666740
14.12257194519040
15.45643901824950
16.96233558654780
18.68184471130370
20.63736152648920
22.82624435424800
25.24896430969230
27.89923286437980
30.77828598022460
33.87284469604490
37.15571212768550
40.54917144775390
43.95245742797850
47.25880813598630
50.36808776855460
53.21256256103510
55.74335098266600
57.93608856201170
59.77521896362300
61.25550079345700
62.35399627685540
63.05057907104490
63.37553024291990
63.34664916992180
62.99328994750970
62.32610702514640
61.31919479370110
59.99690628051750
58.36628341674800

```
56.44129562377920
54.23117065429680
51.71706008911130
48.91371536254880
45.81033325195310
42.43138122558590
38.80767440795890
34.95063018798820
30.93660163879390
26.82167816162100
22.72513198852530
18.74131774902340
15.00180816650390
11.63293838500970
8.74474143981933
6.46722602844238
4.84831571578979
3.97896361351013
3.80848383903503
4.36116695404052
5.50598049163818
7.94416189193725];


hip=round(hip*2);
knee=round(knee*2);

hip = hip/(360/4096);
knee = knee/(360/4096);


plot(handles.axes1,hip); title('Hip Joint Trajectory [Deg]');
xlabel(handles.axes1,'Instruction step');
ylabel(handles.axes1,'Joint Agnle [Deg]');
xlim(handles.axes1,[0 101]);


plot(handles.axes2,knee); title('Knee Joint Trajectory [Deg]');
xlabel(handles.axes2,'Instruction step');
ylabel(handles.axes2,'Joint Agnle [Deg]');
xlim(handles.axes2,[0 101]);



% Control table address
ADDR_PRO_TORQUE_ENABLE          = 64;
ADDR_PRO_LED_RED                = 65;
ADDR_PRO_GOAL_POSITION          = 116;
ADDR_PRO_PRESENT_POSITION       = 132;
ADDR_PRO_OPERATING_MODE         = 11;

% Data Byte Length
LEN_PRO_LED_RED                 = 1;
LEN_PRO_GOAL_POSITION           = 4;
LEN_PRO_PRESENT_POSITION        = 4;

% Protocol version
PROTOCOL_VERSION                = 2.0

% Default setting
DXL1_ID                         = 2;  %Hip           % Dynamixel#1 ID: 1
DXL2_ID                         = 1;  %Knee          % Dynamixel#2 ID: 2
BAUDRATE                        = 56700;
DEVICENAME                      = 'COM5';

TORQUE_ENABLE                   = 1;            % Value for enabling the torque
TORQUE_DISABLE                  = 0;            % Value for disabling the torque
DXL_MINIMUM_POSITION_VALUE      = 0;            % Dynamixel will rotate between
this value
DXL_MAXIMUM_POSITION_VALUE      = 4000;
```

```
DXL_MINIMUM_POSITION_VALUE_2    = 2000;
DXL_MAXIMUM_POSITION_VALUE_2    = 3000;

DXL_MOVING_STATUS_THRESHOLD     = 20;           % Dynamixel moving status threshold

COMM_SUCCESS                    = 0;            % Communication Success result
value
COMM_TX_FAIL                    = -1001;        % Communication Tx Failed

% Initialize PortHandler Structs
% Set the port path
% Get methods and members of PortHandlerLinux or PortHandlerWindows
port_num = portHandler(DEVICENAME);

% Initialize PacketHandler Structs
packetHandler();

% Initialize groupBulkWrite Struct
groupwrite_num = groupBulkWrite(port_num, PROTOCOL_VERSION);

% Initialize Groupbulkread Structs
groupread_num = groupBulkRead(port_num, PROTOCOL_VERSION);

index = 1;
index_2 = 1;
dxl_comm_result = COMM_TX_FAIL;                 % Communication result
dxl_addparam_result = false;                    % AddParam result
dxl_getdata_result = false;                     % GetParam result

dxl_goal_position = [DXL_MINIMUM_POSITION_VALUE DXL_MAXIMUM_POSITION_VALUE];
dxl_goal_position_2 = [DXL_MINIMUM_POSITION_VALUE_2 DXL_MAXIMUM_POSITION_VALUE_2];

dxl_error = 0;
dxl_led_value = [0 1];
dxl1_present_position = 0;
dxl2_present_position = 0;
dxl2_led_value_read = 0;

% Open port
if (openPort(port_num))
    fprintf('Succeeded to open the port!\n');
else
    unloadlibrary('dxl_x64_c');
    fprintf('Failed to open the port!\n');
    input('Press any key to terminate...\n');
    return;
end


% Set port baudrate
if (setBaudRate(port_num, BAUDRATE))
    fprintf('Succeeded to change the baudrate!\n');
else
    unloadlibrary('dxl_x86_c');
    fprintf('Failed to change the baudrate!\n');
    input('Press any key to terminate...\n');
    return;
end


% Enable Dynamixel#1 Torque
write1ByteTxRx(port_num, PROTOCOL_VERSION, DXL1_ID, ADDR_PRO_TORQUE_ENABLE,
TORQUE_ENABLE);
if getLastTxRxResult(port_num, PROTOCOL_VERSION) ~= COMM_SUCCESS
    printTxRxResult(PROTOCOL_VERSION, getLastTxRxResult(port_num,
PROTOCOL_VERSION));
elseif getLastRxPacketError(port_num, PROTOCOL_VERSION) ~= 0
    printRxPacketError(PROTOCOL_VERSION, getLastRxPacketError(port_num,
PROTOCOL_VERSION));
else
```

```
    fprintf('Dynamixel has been successfully connected \n');
end

% Enable Dynamixel#2 Torque
write1ByteTxRx(port_num, PROTOCOL_VERSION, DXL2_ID, ADDR_PRO_TORQUE_ENABLE,
TORQUE_ENABLE);
if getLastTxRxResult(port_num, PROTOCOL_VERSION) ~= COMM_SUCCESS
    printTxRxResult(PROTOCOL_VERSION, getLastTxRxResult(port_num,
PROTOCOL_VERSION));
elseif getLastRxPacketError(port_num, PROTOCOL_VERSION) ~= 0
    printRxPacketError(PROTOCOL_VERSION, getLastRxPacketError(port_num,
PROTOCOL_VERSION));
else
    fprintf('Dynamixel has been successfully connected \n');
end


% Add parameter storage for Dynamixel#1 present position value
dxl_addparam_result = groupBulkReadAddParam(groupread_num, DXL1_ID,
ADDR_PRO_PRESENT_POSITION, LEN_PRO_PRESENT_POSITION);
if dxl_addparam_result ~= true
    fprintf('[ID:%03d] groupBulkRead addparam failed', DXL1_ID);
    return;
end

% Add parameter storage for Dynamixel#2 present position value
dxl_addparam_result = groupBulkReadAddParam(groupread_num, DXL2_ID,
ADDR_PRO_PRESENT_POSITION, LEN_PRO_PRESENT_POSITION);
if dxl_addparam_result ~= true
    fprintf('[ID:%03d] groupBulkRead addparam failed', DXL2_ID);
    return;
end


% Bulkread present positions
groupBulkReadTxRxPacket(groupread_num);
if getLastTxRxResult(port_num, PROTOCOL_VERSION) ~= COMM_SUCCESS
   printTxRxResult(PROTOCOL_VERSION, getLastTxRxResult(port_num,
PROTOCOL_VERSION));
end


% Get Dynamixel#1 hip present position value
dxl1_present_position = groupBulkReadGetData(groupread_num, DXL1_ID,
ADDR_PRO_PRESENT_POSITION, LEN_PRO_PRESENT_POSITION) %HİP AÇISI

% Get Dynamixel#2 knee present position value
dxl2_present_position = groupBulkReadGetData(groupread_num, DXL2_ID,
ADDR_PRO_PRESENT_POSITION, LEN_PRO_PRESENT_POSITION) %KNEE AÇISI

%Present position ile ilk hedef pozisyonu arasındaki farkı bulalım


hip = hip + dxl1_present_position;
knee= knee + dxl2_present_position;


hip = round(hip);
knee = round(knee);

T = str2num(get (handles.edit2,'String')); % seconds
handles.m1.clearLogs()
handles.m1.startStreaming();
pause(1);


FX=0;
FS=0;
EX=0;
RX=0;
```

```
t = 0 ;
x = 0 ;
tstart=99;
figurecount=0;

fullmoveindex=1;
op_mode=-99;
op_mode_counter=0;

while ( t < T )


            maxEMG2=max(handles.m1.emg_log(end-20:end,2));
            maxEMG7=max(handles.m1.emg_log(end-20:end,7));

             input1= [1; maxEMG2; maxEMG7];

             W =     [ 0.8836   -7.0693    6.6697
                       1.5403  -14.4231  -13.9882
                      -9.7691   10.6543   27.0664
                      -0.1365   16.2684  -13.0495];


             V =     [3.3316   10.5119  -27.4685   12.1715
                     -10.7406  -19.9309  -15.7503   25.9366];

             z=zeros(4,1);

             z(1) = 1/(1 + exp(-W(1,:)*input1));
             z(2) = 1/(1 + exp(-W(2,:)*input1));
             z(3) = 1/(1 + exp(-W(3,:)*input1));
             z(4) = 1/(1 + exp(-W(4,:)*input1));


              y(1) = 1/(1 + exp(-V(1,:)*z));
              y(2) = 1/(1 + exp(-V(2,:)*z));

              result=2*y(1) + 1*y(2);


              if(abs(result-0)<0.0001) %FLEXION
                 FX=FX+1;
                 if(FX>5)

                     tic;
                     set (handles.text1,'String','FLEXION');



                  if (op_mode_counter==0)
                      op_mode = read1ByteTxRx(port_num, PROTOCOL_VERSION, DXL1_ID,
ADDR_PRO_OPERATING_MODE)
                      op_mode = read1ByteTxRx(port_num, PROTOCOL_VERSION, DXL2_ID,
ADDR_PRO_OPERATING_MODE)
                      op_mode_counter=1;
                  end

                  while(1)


                  % Add parameter storage for Dynamixel#1 goal position
                  dxl_addparam_result = groupBulkWriteAddParam(groupwrite_num,
DXL1_ID, ADDR_PRO_GOAL_POSITION, LEN_PRO_GOAL_POSITION, typecast(int32(hip(index)),
'uint32'), LEN_PRO_GOAL_POSITION);
                  if dxl_addparam_result ~= true
                     fprintf(stderr, '[ID:%03d] groupBulkWrite addparam failed',
DXL1_ID);
                      return;
                  end
```

```
                    % Add parameter storage for Dynamixel#2 LED value
                     dxl_addparam_result = groupBulkWriteAddParam(groupwrite_num,
DXL2_ID, ADDR_PRO_GOAL_POSITION, LEN_PRO_GOAL_POSITION,
typecast(int32(knee(index)), 'uint32'), LEN_PRO_GOAL_POSITION);
                    if dxl_addparam_result ~= true
                       fprintf(stderr, '[ID:%03d] groupBulkWrite addparam failed',
DXL2_ID);
                        return;
                    end



                    groupBulkWriteTxPacket(groupwrite_num);

                    if getLastTxRxResult(port_num, PROTOCOL_VERSION) ~=
COMM_SUCCESS
                         printTxRxResult(PROTOCOL_VERSION,
getLastTxRxResult(port_num, PROTOCOL_VERSION));
                    end

                    % Clear bulkwrite parameter storage
                    groupBulkWriteClearParam(groupwrite_num);


                    %OBTAIN INSTAN POSITION

                    % Bulkread present positions
                    groupBulkReadTxRxPacket(groupread_num);
                    if getLastTxRxResult(port_num, PROTOCOL_VERSION) ~=
COMM_SUCCESS
                       printTxRxResult(PROTOCOL_VERSION,
getLastTxRxResult(port_num, PROTOCOL_VERSION));
                    end



                    % Check if groupbulkread data of Dynamixel#1 is available
                                        dxl_getdata_result =
groupBulkReadIsAvailable(groupread_num, DXL1_ID, ADDR_PRO_PRESENT_POSITION,
LEN_PRO_PRESENT_POSITION);
                                        if dxl_getdata_result ~= true
                                            fprintf('[ID:%03d] groupBulkRead getdata
failed', DXL1_ID);
                                            return;
                                        end

                                        % Check if groupbulkread data of Dynamixel#2 is
available
                                        dxl_getdata_result =
groupBulkReadIsAvailable(groupread_num, DXL2_ID, ADDR_PRO_PRESENT_POSITION,
LEN_PRO_PRESENT_POSITION);
                                        if dxl_getdata_result ~= true
                                            fprintf('[ID:%03d] groupBulkRead getdata
failed', DXL2_ID);
                                            return;
                                        end

                                        % Get Dynamixel#1 present position value
                                        dxl1_present_position =
groupBulkReadGetData(groupread_num, DXL1_ID, ADDR_PRO_PRESENT_POSITION,
LEN_PRO_PRESENT_POSITION);

                                        % Get Dynamixel#2 present position value
                                        dxl2_present_position =
groupBulkReadGetData(groupread_num, DXL2_ID, ADDR_PRO_PRESENT_POSITION,
LEN_PRO_PRESENT_POSITION);

                                        pos1(index) =
typecast(uint32(dxl1_present_position), 'int32');
```

```
                                    pos2(index) =
typecast(uint32(dxl2_present_position), 'int32');


                                    pos1full(fullmoveindex) =
typecast(uint32(dxl1_present_position), 'int32');
                                    pos2full(fullmoveindex) =
typecast(uint32(dxl2_present_position), 'int32');

                                    timervalue=toc;
                                    t1(index)=timervalue;

                                    index = index + 1 ;

                                    a=handles.m1.timeEMG_log;
                                    b=ones(length(a),1);
                                    timeemg = a-a(1).*b;
                                    t = timeemg(end)-1;

                                   set(handles.edit2,'String',num2str(t));
                                               pause(0.0001);

                                    %Record full time
                                    t1full(fullmoveindex)=t;

                                    fullmoveindex=fullmoveindex+1;

                                    if index == 102
                                        index = 1;
                                        index_2 = index_2 + 1;
                                    end

                                    if index_2 == 2
                                        index_2=1;
                                        break
                                    end


                end


        end

                FS=0;
                RX=0;
                EX=0;

end

if(abs(result-1)<0.0001) %FIST
   FS=FS+1;

    if(FS>5)
       set (handles.text1,'String','FIST');

       plot(handles.axes5,pos1); title('Hip Joint Trajectory [Deg]');
       xlabel(handles.axes5,'Instruction step');
       ylabel(handles.axes5,'Joint Angle [Deg]');
       xlim(handles.axes5,[0 101]);

       plot(handles.axes6,pos2); title('Knee Joint Trajectory [Deg]');
       xlabel(handles.axes6,'Instruction step');
       ylabel(handles.axes6,'Joint Angle [Deg]');
       xlim(handles.axes6,[0 101]);


       if(figurecount==0)

          figure
          plot(pos1)
```

```
            hold on
            plot(hip)


           figure
           plot(pos2)
            hold on
            plot(knee)


           figure
           plot(t1,pos1)
           hold on
           plot(t1,hip)


           figure
           plot(t1,pos2)
           hold on
           plot(t1,knee)


           figure
           plot(pos1full)
           hold on
           plot(pos2full)


           figure
           plot(t1full,pos1full)
           hold on
           plot(t1full,pos2full)

           figurecount=1;
       end

   end
           FX=0;
           RX=0;
           EX=0
end


if(abs(result-2)<0.0001) %DO NOTHING
    RX=RX+1;
    if(RX>5)
        set (handles.text1,'String','RELAX');

    end
    FX=0;
    FS=0;
    EX=0;
End

if(abs(result-3)<0.0001) %EXTENSION
    EX=EX+1;
    if(EX>5)
        set (handles.text1,'String','EXTENSION');

    end
    FX=0;
    FS=0;
    RX=0;
end


a=handles.m1.timeEMG_log;
b=ones(length(a),1);
timeemg = a-a(1).*b;
t = timeemg(end)-1;
```

```
set(handles.edit2,'String',num2str(t));
pause(0.0001);


end
    handles.m1.stopStreaming();
    handles.m1.timeEMG_log(end)
    handles.mm.delete();

    clear handles.mm;

    % Disable Dynamixel#1 Torque
write1ByteTxRx(port_num, PROTOCOL_VERSION, DXL1_ID, ADDR_PRO_TORQUE_ENABLE,
TORQUE_DISABLE);
if getLastTxRxResult(port_num, PROTOCOL_VERSION) ~= COMM_SUCCESS
    printTxRxResult(PROTOCOL_VERSION, getLastTxRxResult(port_num,
PROTOCOL_VERSION));
elseif getLastRxPacketError(port_num, PROTOCOL_VERSION) ~= 0
    printRxPacketError(PROTOCOL_VERSION, getLastRxPacketError(port_num,
PROTOCOL_VERSION));
end

% Disable Dynamixel#2 Torque
write1ByteTxRx(port_num, PROTOCOL_VERSION, DXL2_ID, ADDR_PRO_TORQUE_ENABLE,
TORQUE_DISABLE);
if getLastTxRxResult(port_num, PROTOCOL_VERSION) ~= COMM_SUCCESS
    printTxRxResult(PROTOCOL_VERSION, getLastTxRxResult(port_num,
PROTOCOL_VERSION));
elseif getLastRxPacketError(port_num, PROTOCOL_VERSION) ~= 0
    printRxPacketError(PROTOCOL_VERSION, getLastRxPacketError(port_num,
PROTOCOL_VERSION));
end

% Close port
closePort(port_num);

% Unload Library
unloadlibrary('dxl_x64_c');

 set (handles.text1,'String','ENDED');



function edit2_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)

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

RESUMES

Erdoğan Yıldız
E-Mail: yildizcontact@gmail.com

EDUCATION

2009-2013
Antalya Anatolian Highschool
GPA: [90.5/100]

2015-2018
Yıldız Technical University (100% English) Department of Mechatronics Engineering
GPA: [3.83/4]

2017-2018 Spring Semester (Erasmus)
Budapest University of Technology and Economics Mechatronics Engineering

INTERNSHIPS

2017 – Orker Asansör
2018 – ASELSAN
2019 - TUSAŞ

LANGUAGE

ENGLISH: Good
Level: C1 (According to the Common European Framework of Reference for Languages(CEFR))

COMPUTER

Java, C#, Matlab, Simulink, Assembly, Solidworks, PTC Creo, Multisim, Proteus,Html,Css,Javascript

Erdem IŞIK
E-Mail : kngngstr@gmail.com

EDUCATION

2010-2014
Hasan Polatkan Anatolian Highschool
GPA: [78.8/100]

2014-2018
Yıldız Technical University (100% English) Department of Mechatronics Engineering
GPA: [2.49/4]

2017-2018 Spring Semester (Erasmus)
Budapest University of Technology and Economics Mechatronics Engineering

INTERNSHIPS

2017 – Pressan
2018 – Bimsa

LANGUAGE

ENGLISH: Good
Level: C1 (According to the Common European Framework of Reference for Languages(CEFR))

COMPUTER

Java, C#, Python, Linux, Matlab, Simulink, Solidworks, Proteus, Microchip PIC