Development and Control of a Shoulder Joint for Humanoid Robotics Application

by

Zhenis Otarbay

Submitted to the Department of Robotics and Mechatronics
in partial fulfillment of the requirements for the degree of
Master of Science in Robotics
at the
NAZARBAYEV UNIVERSITY
Apr 2020

Author .....
Department of Robotics and Mechatronics
Apr 29, 2020

Certified by 19/03/2021
Michele Folgheraiter (Thesis supervisor)
Associate Professor
Thesis Supervisor

Certified by .....
Berdakh Abidullaev (co-supervisor)
Assistant professor
Thesis Supervisor

Certified by .....
Mohamad Mosadeghzad (co-supervisor)
Assistant professor
Thesis Supervisor

Certified by .....
Matteo Rubagotti(external supervisor)
Associate Professor
Thesis Supervisor

Accepted by .....
Vassilios D. Tourassis
Dean, School of Engineering and Digital Sciences

Abstract

Humanoid robotics represents an important sub-field of Robotics that has many applications, for example in house assistance, surveillance, medical assistance, dangerous environments, repetitive works, and rehabilitation. Human joints are difficult to replicate in a humanoid robot. The available biological models may differ from the real-life robotic models. This work aims at modeling, building, and controlling a tendon-driven shoulder joint. Force and position sensors will be integrated for further study on the system dynamics. A Python-V-REP model of the joint will be developed to study the joint motion and to optimize its design. Also, a python interface will be used to visualize the control mechanism and to represent the force sensor and magnetic encoder values and facilitates monitoring and controlling the actuators' status. We will also explore the possibility to use EEG (Electroencephalogram) signals to control the robotic joint, this foreseen possible applications in telerobotics and as prosthesis. A parallel manipulator with three curved limbs is used in humanoid robots as a shoulder and wrist. Describing the kinematics of this type of Stewart platform is a challenging process. This paper also tries to examine two methods of finding direct and inverse kinematics of parallel manipulators. The successive screw method is applied on one limb of the manipulator with an assumption that the other two legs will follow the first one. The geometric method describes a change in height of the manipulator where pulleys are located.

Acknowledgments

At first, we would like to thank our supervisor professor Michele Folgheraiter for his guidance and patience. Furthermore, we would like to thank professor Berdakh Abibullaev, professor Matteo Rubagotti, professor Atakan Varol, professor Mohammad Mosadeghzad, professor Almas Shintemirov, professor Tohid Alizadeh, professor Anara Sandygulova, professor Do Duc Ton, professor Zhanat Kappasov, instructor Altay Zhakatayev, instructor Iliyas Tursynbek for helping us and as Professors to teach us many important subjects throughout the six years of study (4 years in bachelor and 2 years in masters) in Nazarbayev University and to my colleagues Sharafatdin Yessirkepov, Darkhan Zholtayev, Zaksylyk Kazykenov, Alfarabi Imashev, Fahim Raza Sunasara, Asset Yskakov and Timur Ishuov.

Contents

1 Introduction **9**

1.1 Introduction to the topic 9

1.2 Importance of the topic 10

1.3 Literature review (The current state of the realm and existing solutions) 10

1.4 Knowledge gap (What is missing?) 15

1.5 Problem statement 16

1.6 Potential impact 17

1.7 Terminology 17

1.8 Classification, slicing, and dicing, taxonomy 18

2 Design and implementation **19**

2.1 Mechanical design (Computer-Aided Design) 19

2.2 Electrical design 22

2.3 Manufacturing and assembly 23

2.4 GUI to manage the enarthrosis 27

2.5 Force and position control 29

3 Articulatio spheroidea model and simulation **32**

3.1 Detailed design of three degrees of freedom manipulator 32

3.2 Kinematics for the primary leg or limb: 33

3.3 Inverse Kinematics 34

3.4 Inverse Kinematics - Geometric Method (based on [7], see some related paper from our Folgheraiter's lab: [8]) 36

3.5 MATLAB simulation of PM 38

3.6 Kinematics of PM using Geometrical Method 38

3.7 3D design components of PM 40
3.8 A V-rep model which imitates the spherical joint and a workspace of an imitating v-rep model 43
3.9 EEG Control 44
3.10 availability of resources like equipment, consumables, library, and computation to unravel the matter 46
3.11 Approach and methods to resolve the matter 46
3.12 Research challenges 46

4 Testing and results **48**
4.1 Experiment results 48

5 Conclusion and future work **53**

A Tables **54**
A.1 Costs of the materials 54

B GUI software **55**
B.1 GUI software in python 55

List of Figures

1-1 Lims ambidex [7] . . . 11
1-2 Survey results . . . 13
1-3 BCI-controlled-UR-manipulator [10] . . . 14
1-4 Paraplegic woman uses her thoughts to guide a robotic arm [10] . . . 14
1-5 Folgheraiter et al EEG based BCI to predict motion [13] . . . 15
1-6 Change of L in ball-and-socket joint . . . 16
1-7 the Stewart platform with rotational joints . . . 16

2-1 Parts of low-inertia high-stiffness manipulator . . . 20
2-2 Views of articulatio humeri assembly in Solidworks a) enarthrosis assembly and simulation in Solidworks b) cotyloid joint is bent to target position. . . 21
2-3 Tendon actuation system . . . 22
2-4 Circuit schematics design in KiCAD . . . 23
2-5 PCB design in KiCAD . . . 24
2-6 Proof of concept for a 3-DOF shoulder joint (only two actuated) realized in PLA. . . . 25
2-7 Concept explanation [4] . . . 26
2-8 Motherboard which incorporates a Teensy 3.5, force sensors amplifiers, motor controllers, and rotary encoders. . . . 27
2-9 Concept explanation: interface . . . 28
2-10 Example of an interface to manage the spheroid joint (the graph shows the force sensor measurements) . . . 29
2-11 Position encoder forward 3 complete rotations . . . 30
2-12 Position encoder reverse 3 complete . . . 30
2-13 Position or angle relationship . . . 30

2-14 Force and position control 31

3-1 Architecture of Stewart platform with rotational joints 32
3-2 Screw method 33
3-3 Formulation of dependency 36
3-4 geometry of movement 37
3-5 Geometrical solution attempt for the one limb adapted from [7] 37
3-6 Geometrical solution attempt for the one limb adapted from [7] 38
3-7 MATLAB simulation of PM, the first leg is bent $\frac{\pi}{4}$ (θ angle) and therefore the moving platform is bent $\frac{\pi}{4}$ (ϕ angle) 39
3-8 MATLAB simulation of PM, the first leg is in the base position 40
3-9 3-DOF parallel manipulator with coordinates adapted from [7] 41
3-10 Low inertia high-stiffness parallel manipulator 41
3-11 V-rep simulation, a) the model which is controlled with x and y angles b) simulation of cotyloid joint 42
3-12 V-rep simulation, a) speed control of the model which is controlled with x and y angles b) speed control of the tendon actuated shoulder joint simulation c) graph which is obtained from shoulder joint simulation d) the model which is controlled with x and y angles is bent to a target location e) simulation of the tendon actuated shoulder joint where it is bent to a target location 42
3-13 Shoulder joint platform older design version 43
3-14 When the tendons in vertical orientation [4] 43
3-15 A V-rep model which imitates the spherical joint bent to 90 degrees . . . 44
3-16 Workspace of a imitating v-rep model 44
3-17 Workspace of a imitating v-rep model represented in 2d 45

4-1 Proof of concept for a 3-DOF shoulder joint (only two actuated) realized in PLA. The shoulder joint is bent to the target position. 49
4-2 Proof of concept for a 3-DOF shoulder joint with an elastic tendon (only two actuated) realized in PLA. The shoulder joint is bent to the target position. 50
4-3 Act left about 90 degrees 50

4-4 Act left about 90 degrees, 3 force sensor measurements 51
4-5 Act right about 30 degrees and Act top -45 degrees 51
4-6 Act right about 30degrees and Act top -45 degrees dynamic graph 51
4-7 Act left 75 degrees 52
4-8 Act left 75 degrees dynamic graph 52

Chapter 1

Introduction

1.1 Introduction to the topic

Humanoid robotics aims at developing robotics systems that manipulate objects and move in the environment in a similar way humans can do. Their kinematic architecture is meant to partially reproduce the mobility of the human skeleton. In most of the systems developed so far, a sequence of revolute joints is used to reproduce the mobility of complex articulations present in the human body [17]. As an example, the human shoulder represents a complex mechanism that can apply rotation and translations to the upper limb. In this thesis work, we want to develop a tendon actuated spherical joint that can reproduce the behavior of the humerus-glenoid cavity present in the human shoulder. We will also explore, with a feasibility study, the possibility to integrate EEG signals in the control architecture to predict motion [15].

Parallel manipulators can withstand huge loads. One type of parallel manipulator is the Stewart platform. The Stewart platforms can be used to substitute shoulder and wrist in humanoid robots. This work will try to solve direct and inverse kinematics of the Stewart platform that has 12 rotational joints, 4 in each limb. Also, we will examine the problem of finding the relationship between the end-effector's position and orientation. We have also tried to verify direct and inverse kinematics using MATLAB and the simulation was done in V-REP. Successful implementation of the Stewart platform with 12 rotational joints would help to make the humanoid robot's hand robust.

1.2 Importance of the topic

The problem we intend to solve is very important for future applications in the field of humanoid robotics. An actuated spherical joint will allow the robot to move more naturally and simplify its design. Since the actuation system will be installed before the joint, the upper arm will be lighter and therefore capable to move faster and displace bigger payloads.

The possibility to integrate EEG signals in the control architecture of the artificial shoulder represents an important open research question [2].

Using brain signals to assist the control of the shoulder motion may be beneficial for many people who have lost their limbs and need a more natural control of their prosthetic arm [5] [4]. The system can also be used for teleoperation purposes, facilitating the operation of an exoskeleton (master-arm). Predicting the motion of the operator's arm could be used to reduce the stiffness of the exoskeleton and allowing a more natural operation [10].

1.3 Literature review (The current state of the realm and existing solutions)

This literature review is in two sections. In the first part, we will only talk about the design and control of shoulder joints. In the second part, we will talk about the shoulder joints that are controlled by EEG signals from the brain.

Control of shoulder joints survey

In this survey different designs will be compared and opposed to each other, to make 3 DOF tendon-driven shoulder joints robust. On each paper, there is some important information in torque, force, design specialties, and the most important findings.
Kim et al (2018) [7] emphasize their high speed, safe interaction with their robot with robotic joints. This is the most important side that will be processed in this paper, how it is possible to make the robot joints move with high speed, additionally low stiffness, and safe interaction will be considered. This paper contains quaternion representation, forward and inverse kinematics. The key point is to master them and apply the knowledge to build a system. So, the mathematics of this paper is valuable. The authors brought

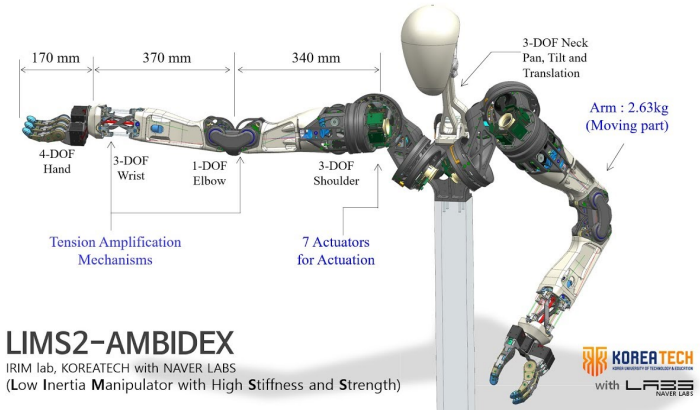

Figure 1-1: Lims ambidex [7]

their robot structure to real life is called LIMS2-AMBIDEX.
This research aims to develop our design approaching Kim et al's (2018) design level using 3d printed parts or we will design our model that would somehow over-perform. The mechanical design and the electronics part represent the core of my thesis.

The design of our shoulder joint initially was aimed to operate using tendons like [7] Kim and others humanoid arm's shoulder joint, however, we will also evaluate other actuation mechanisms.

Additionally, this paper uses the quaternion representation of the shoulder joint kinematics and implements the forward and inverse kinematics.

One of our goals is to over-perform Kim et al's (2018) design level in constructing this joint.
Ball et al (2017, August) emphasize the robotic joint's rehabilitation sides. And the design of the robotic joint is made especially safe and for rehabilitation purposes, rather than for experimentation of dangerous works, for example. This has a real-life prototype. *Ball et al (2017, September)* also emphasizes its rehabilitation side and has a robotic exoskeleton of 5 DOF and this designed mechanical joint is cable-driven. The value of these two Ball et al's papers in medical applications is relatively huge. And the design of the system is appealing. This publication probably has its real-life prototype, but in the paper, it was a bit unclear or was not noted.
Ball et al (2017) represented a planar 3 DOF cable-driven robotic joint that is to move shoulder and wrist for rehabilitation purposes.
Buongiorno et al (2018) [3] emphasize its tendon-driven differential transmission mecha-

nism for the rehabilitation side. So, it is important to know how differential transmission is used for this purpose. It is a bit intuitive, but not rare in this domain. These authors have their mechanics in real life and worn by users.
Buongiorno et al (2018) designed 3 DOF shoulder joints with the differential transmission.
Jiang et al (2019) emphasize [6] their modular design and replication human arm which is 7 DOF. So, it is more difficult to replicate a holonomic redundant system. So, it is particularly interesting to understand how this mechanism works. In the paper, it is unclear if they have real-life humanoid arms built.
Jiang et al (2019) also designed a tendon-driven 7-DOF cable-driven humanoid arm.

Most of the citations for Ball et al's (2017, August) work, whose group is at Queen's University.

All the papers above represent application papers. All of the papers as you could read theoretically proven, most of them are proven in CAD. Some papers used a tendon-driven mechanism, for example, Lims-ambidex 2 and some other scientific publications that will be compared and contrasted in this survey paper. The construction process of the design of the 3-DOF spherical joint was started and is close to the final version.

Folgheraiter et al (2019) [13] focus on Cartesian and joint space modalities of existing teach pendant. The paper is important as the propositions can be used to make the interface of our 3-DOF tendon-driven shoulder joint. This is our previous year's work and we want to add a page that controls our 3 DOF shoulder joint to the pendant. Eventually, this pendant can control the whole NU-biped robot. This paper has its theoretical proofs, CAD model, and real-life device. Automation of the device was implemented using python modules such as matplotlib, NumPy, scipy in the back end. And as a front end uses a python Tkinter module.

Summarize of the Survey part There were compared and contrasted 6 publication works in this survey paper. Ball et al [1] presented relatively robust and rehabilitation aimed exoskeletons. Tendon is comprehensive and a bit more robust for shoulder or

Hand authors	mass	Nominal size	Number of actuators	Number of tendons/cables	Number of sensors	Number of DOF
Kim et al (2018)	Shoulder: 4.17 kg	Small size (not enough information)	~5+	~12 in one shoulder or wrist joint	~6 sensors for shoulder (10-15 sensors for whole robot)	3
Ball et al (2017, August)	No information	No information	~5+	4+	No information	3
Buongiorno et al (2018)	~4.7kg (WRES + driverbox)	Standard man's wrist side hand size	~3+	3+	6+	3
Jiang et al (2019)	No information	No information	~3+	6+	No information	7
Ball et al (2017, September)	115 kg (Actuators and exoskeleton)	exoskeleton	~5+	5+	3+	5
Folgheraiter and Otarbay* (planned to send for publishing in 2020-2021)	~2.4kg	Radius ~ 140mm and the height is about 300mm	4	3	7 (3 force, 4 magnetic encoders)	3

Figure 1-2: Survey results

wrist joint substitute purposes. Like some of the literature we reviewed here, we are also going to apply the simulation techniques for the shoulder joint before we test each of the movements in the real life. But, the shoulder joint can be moved in real-life now as a preliminary work with the control mechanism program and with the python interface.

V Shoulder or robotic grippers that are controlled by EEG signals from the brain survey

The robotic arm UR5 (Universal Robots 5) is controlled in the paper of Abidullaev et al (2020, February)

In this [9] work from Abidullaev (2020) and colleagues, the robotic arm UR5 is controlled using EEG signals acquired from the brain. The accuracy of classification to control left and right or up and down movements achieved for 7 different subjects is about 75.9% on average. This is a bit lower than what is required for real-time control and also makes the subject tired, so that makes this control mechanism not so practical.

Luu et al [16] present EEG signals from the brain to make a walking avatar. For that, they get EEG signals from the cortical part of the brain. The used dataset is EEG obtained from 64 channels for 100 Hz. Useful to my research even though it is for walking, the general structure uses the EEG signal from the brain.

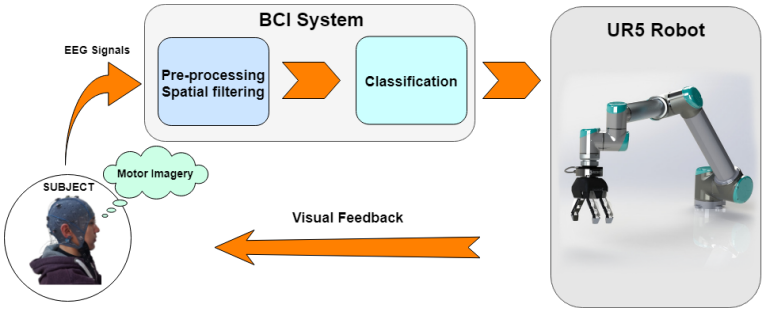

Figure 1-3: BCI-controlled-UR-manipulator [10]

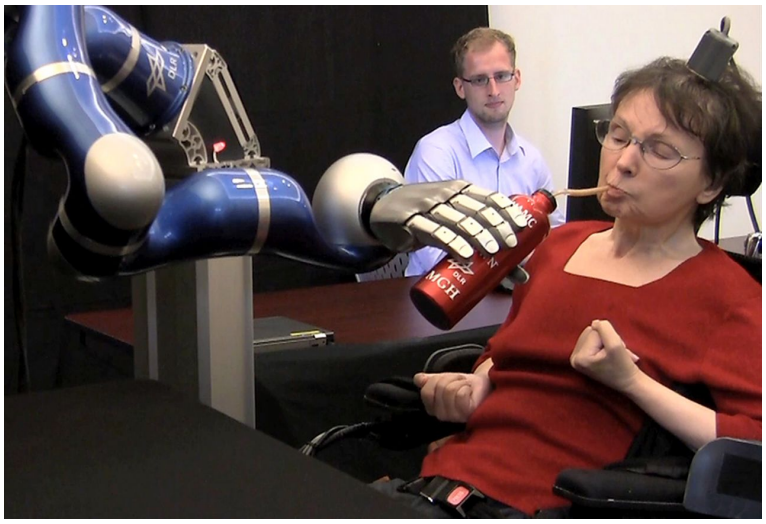

Figure 1-4: Paraplegic woman uses her thoughts to guide a robotic arm [10]

The following is an application of EEG robotic arm control using a direct brain interface. *Fu et al* [14] represents an EEG control mechanism for shoulder-elbow joint by focusing on ERD for stroke survivors to assist them. One of the important parts of this research is noise rejection which we can apply in my research. They reject noises from the scalp and facial muscles.

In [12], the authors implemented EEG integrated with 9 DOF shoulder joints. I reviewed the work of Folgheraiter et. al. implementing a 9 DOF exoskeleton that controls the system is modulated with the EEG signals. These are used to predict 300 ms in advance the intention of movement of the user and trigger the exoskeleton actuation system. This design suggests a robust electronics and safe design with its software. The voltage and the power of the hardware are also safe, which is highly unlikely to harm the human subject. This research represents teleoperation-based software-hardware parts and a movement prediction technique which is one of the important parts of the research that is related to my thesis work.

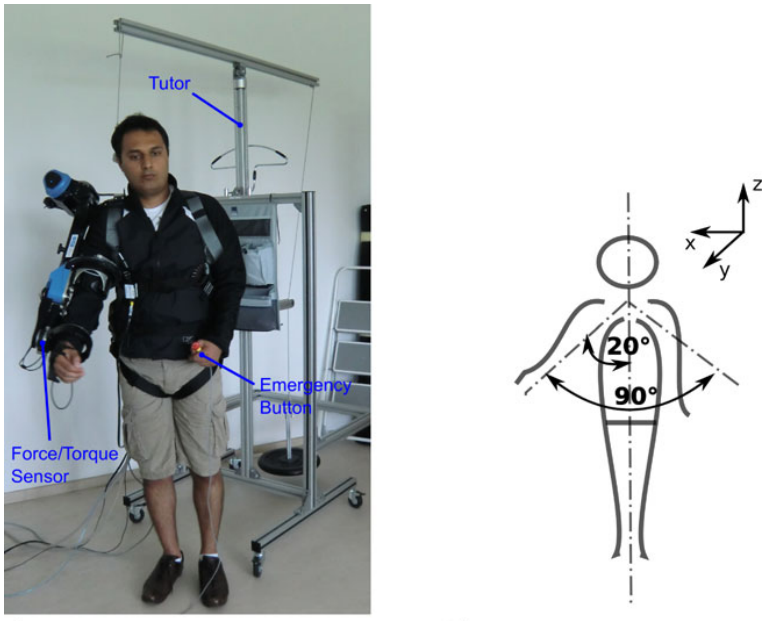

Figure 1-5: Folgheraiter et al EEG based BCI to predict motion [13]

Bhagat et al [11] show a feasibility study of an asynchronous EEG-based BMI for a chronic stroke. Although this method is slightly different from the method we are going to use, but it is more or less related to its general structure. This research uses combined EEG and EMG to assist the subjects' movements. Fixation is also used in this research to determine the subjects' intentions.

1.4 Knowledge gap (What is missing?)

Most of the researches that we listed in the literature review are not employing a spherical joint in the shoulder, but a sequence of revolute joints [17]. The payload is generally limited, especially in tendon-driven systems. Different systems are also not accurate or report just simulated models. Very few examples are available of integration of EEG signals in the control architecture.

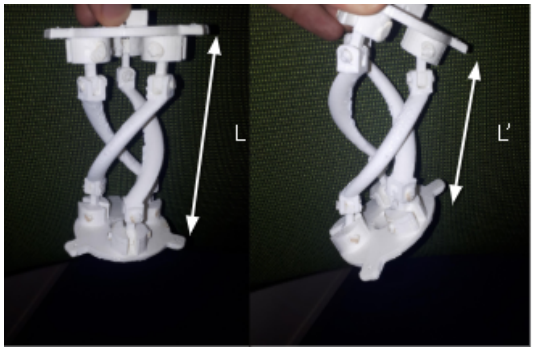

Figure 1-6: Change of L in ball-and-socket joint

1.5 Problem statement

Part of our research is focusing on measuring the shoulder activity in a dynamic interface we can measure dynamic 3 force sensor values and 4 magnetic encoder values where each of the magnetic encoders are attached to the actuators and each of the force sponsors are attached to the shoulder parts where we can measure forces. After this, we will focus on the V-rep simulation part to investigate how the shoulder joint performs in the virtual simulation so that until the end of the project the shoulder joint will not be bothered too much in real life.

Figure 1-6 represents the change of L in the shoulder joint for the older version of the model:

Direct and inverse kinematics of the Stewart platform with rotational joints and curved limbs need to be solved and verified algebraically, geometrically using MATLAB, CAD, and simulation platforms. The main problem is to identify the relationship between the end-effector's position and orientation. The challenge of solving the dynamics problem is to make sure correctness and coincidence with the theoretical calculations.

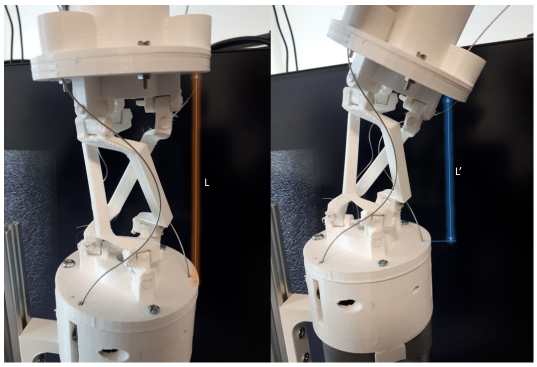

Figure 1-7: the Stewart platform with rotational joints

1.6 Potential impact

It may bring many benefits to our country in terms of science, economics, technology, and socially.

1.7 Terminology

V-rep - is a software to simulate the mechanical assembles to see how it performs and analyze its movements before implementing the movements in real-life so that the shoulder is not bothered too much before its development is finished.
DOF - degree of freedom, characteristics of the movement of mechanical systems. Quaternion (from Latin quaternion, four) - system of hyper-complex numbers, forming vector space with a dimension of four on the field of material numbers. It is used to decrease the calculation amount.
Forward kinematics - the process of determining parameters of connected flexible objects (for example, kinematic pairs or kinematic chains) for approaching necessary positions, orientations, and the location of these objects.
Inverse kinematics - the process of finding parameters of connected flexible objects (for example, kinematic pairs or kinematic chains) for approaching necessary positions, orientations, and the location of these objects. We have an end-effector in the case of a tendon-driven joint, and we aim to find the angles of the joints.
EEG - Electroencephalogram is a way of representing brain data.
UR5 - Universal robotics 5, a family of robot manipulators for research and industrial purposes.
PCB - printed circuit board.
KiCAD - software for designing models of PCB designs.
CNC machine - computer numerical control machine for engraving the models of PCB.
PWM signal - pulse width modulation signal.
DRV8835 - a motor driver to actuate and control the directions of 2 actuators.

1.8 Classification, slicing, and dicing, taxonomy

Exoskeleton rehabilitation

One of the most successful applications of robotic rehabilitation mechanisms is Exoskeleton rehabilitation. Usually, it consists of planar cable-driven 3 DOF joints. A person can sit on an exoskeleton mechanism and can move his hands.

Robotic shoulder joints to make autonomous robots

LIMS-AMBIDEX2 is one of the closest examples of shoulder joints. The shoulder and wrist joints are tendon-driven, fast, and low stiffness. This kind of design may sustain high torque and low cost.

3 DOF differential transmission planar tendon-driven robotic rehabilitation hands This design allows robust control in hand motions. The force sensors and magnetic encoders give information about torque, force, weight, and joint angle rotation. **7 DOF modular design, replicated human arm** Is aimed to make a holonomic and redundant structure like a human arm. Tendon is used to actuate the spherical joints such as the shoulder and wrist. Theoretically proved in CAD.

Chapter 2

Design and implementation

As a preliminary work, firstly we have tried to make calculations for the shoulder joint and tried to find the distance change between the fixed and moving part of the shoulder joint. Then, we have made a control mechanism where coded in teensy 3.6 and python 3 for the interface and we have now an interface that can control the shoulder joint. Also, we have worked on a simulated model for the older version of the shoulder joint. Then, we have made a KiCAD model to engrave the PCB for the shoulder joint.
There was designed a V-rep model for the older version of the hardware and now we can update it for the newer version as it is primarily working in the real life. The constraints mostly were used in the v-rep model.

2.1 Mechanical design (Computer-Aided Design)

The model was drawn, designed, and assembled first. After it was successfully assembled, it was printed. Figure **??** represents the cad model of our shoulder joint. The Solidworks was used to design and simulate. It was possible to do a detailed design that reduced iterations. The constraints such as force, motor speed, torque were used in the v-rep model of the shoulder joint. The angular constraints or the maximum and minimum angles reach of the bend are pitch - 90 degrees, roll - 60 degrees. These are the angles at which the shoulder can rotate. **??** shows the views of shoulder joint assembly in Solidworks low-inertia high-stiffness manipulator with 180 degrees curved legs $h_{leg} = 60mm$. Each leg makes 180 degrees of a curve from the base to reach the platform. The three limbs of the shoulder joint are separated 120 degrees from each other.

Parts of the SPM	Material type	Approximate dimensions in axis, mm			Quantity
		X axis	Y axis	Z axis	
Base part	PLA	140	140	8	1
Curved leg	Metal	40	10	90	3
Universal joint(plastic)	Metal	10	10	30	6
Tendon wire	Metal	1	1	150	3
Circul**ar** Platform	PLA	140	140	50	1
Manipulator hanger	PLA	65	20	110	1
Aluminum extrusion	Aluminum	300	20	20	4
Clamps	PLA	10	60	50	2
motor pack	PLA	140	140	45	1
pulley	Stainless steel	10	10	3	3
Shaft pack	PLA	25	25	30	3
Motor holder	PLA	140	140	3	1
Motor	metal	38	38	75	4

Figure 2-1: Parts of low-inertia high-stiffness manipulator

The symmetric orientation of the platform compared to the base makes 60 degrees. The 2-1 consists of shoulder joint part materials, approximate dimensions, and their quantity. The CAD model is as good as the actual 3d printed shoulder joint.

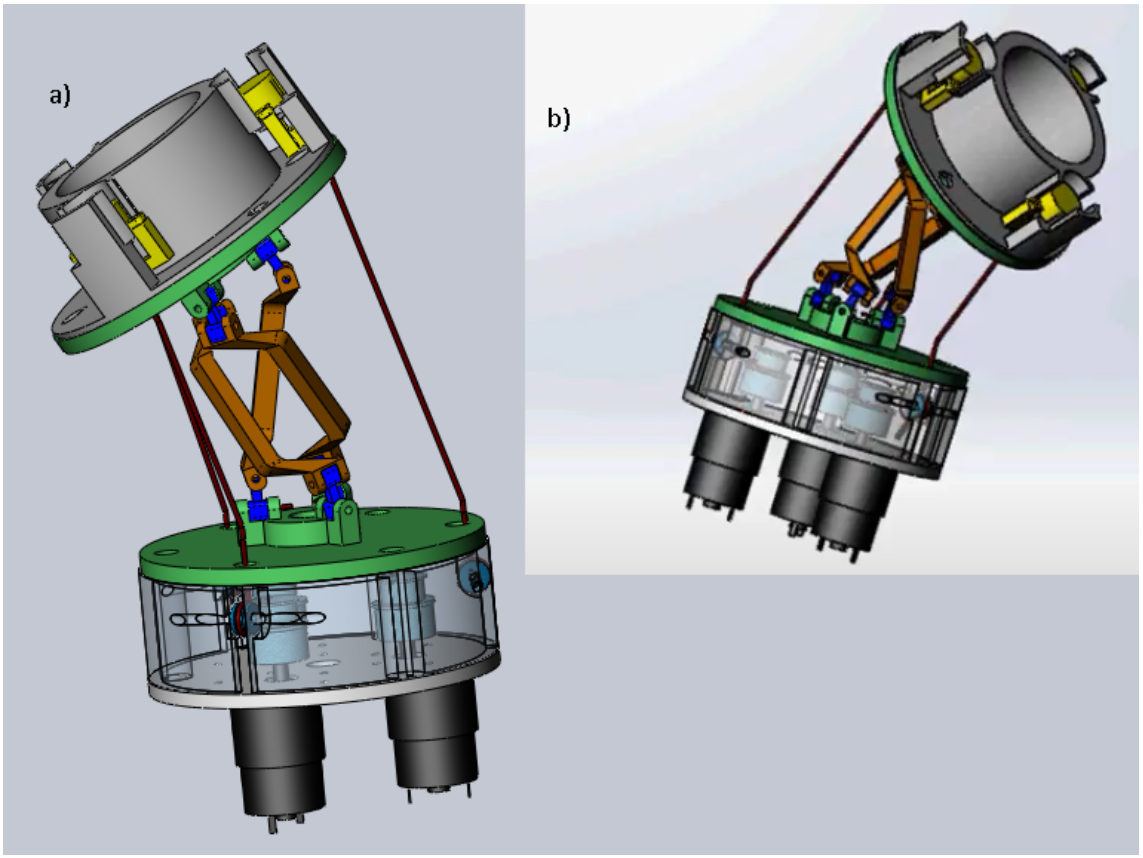

Figure 2-2: Views of articulatio humeri assembly in Solidworks a) enarthrosis assembly and simulation in Solidworks b) cotyloid joint is bent to target position.

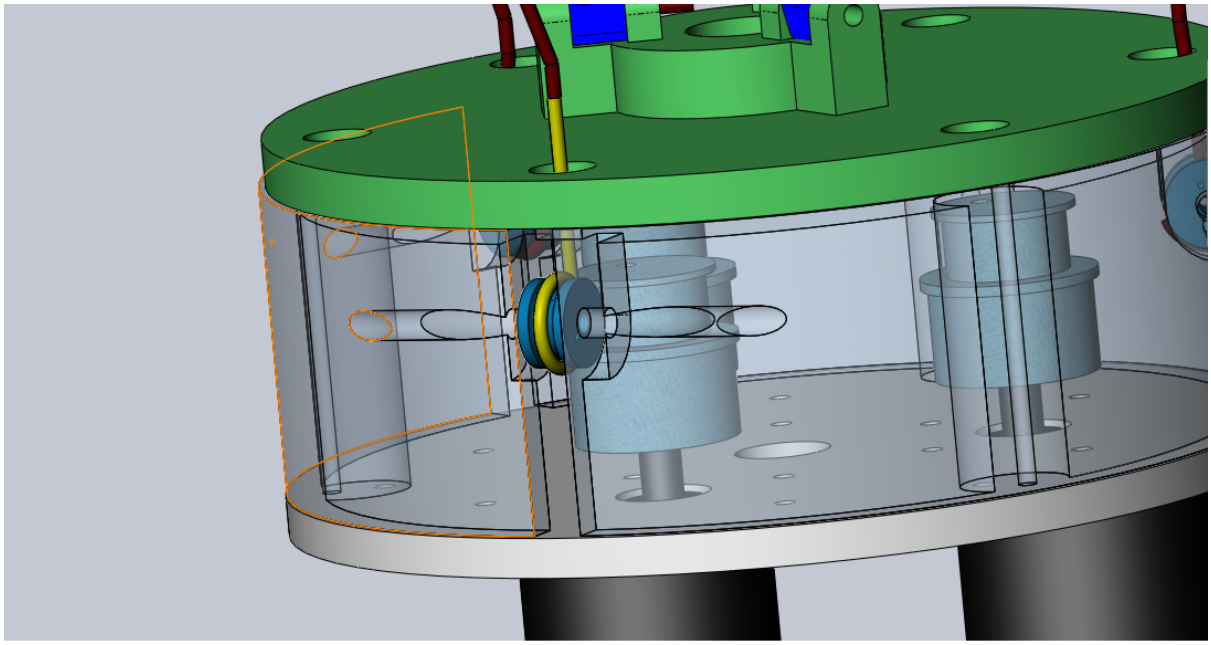

Figure 2-3: Tendon actuation system

2.2 Electrical design

Figure 2-3 demonstrates how tendon actuation in our shoulder joint works. The yellow one which is the tendon is rotated to the head of the actuator. So, when an actuator is rotated clockwise and anticlockwise, the tendon is tightened and loosened.

The following is the schematics of the circuit of the circuit in figure 2-4 The following figure 2-5 represents the KiCAD PCB design model for the shoulder control mechanism. We built this PCB to reduce the amount of wires. To avoid this we choose to make a PCB plate for the control mechanism. Almost all of the footprints and schematics were designed by the author of this document. We realized the PCB using a CNC machine and after, to avoid the presence of short circuits, we tested all the connection on the PCB with a multi-meter. The KiCAD model is flipped for all components which allows checking the teensy 3.5 pins easily by looking at its numbered side. This is one of the unusual sides of the design. However, this does not affect the work of the control mechanism. Also, the reset button of teensy 3.5 can be used in controlling PC or just can be reset when there is no power for the teensy 3.5, so that is rarely used, but anyway, the push button can be pressed even though it is looking at the PCB side.

Figure 2-5 represents the KiCAD model for the PCB of the control mechanism.

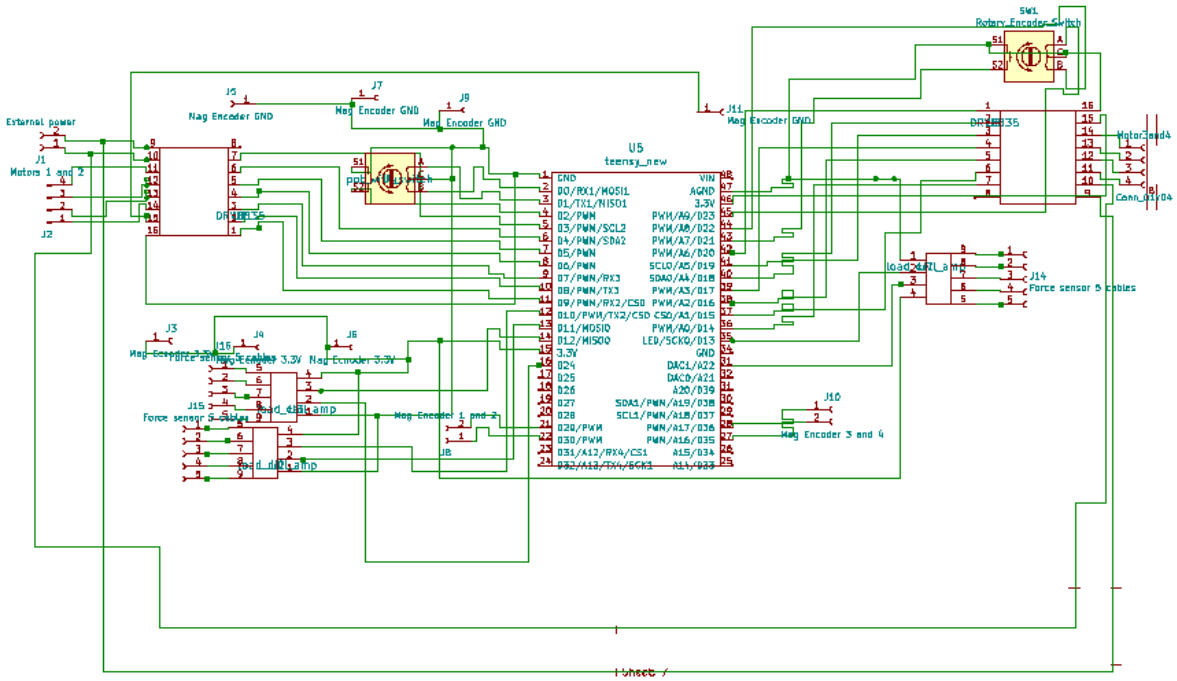

Figure 2-4: Circuit schematics design in KiCAD

2.3 Manufacturing and assembly

Figure 2-6 is the demonstration of our tendon-driven joint which we intended to replicate human's spherical shoulder joint (for now it is limited to 2 DOF as long as we are using only 3 tendons, it can be 3 DOF if we use 4 tendons). Each of the three actuators winds the tendons and thread through holes of the platforms. And the next end of the tendons is fixed to the moving platform. The angle between each tendon is 120 degrees. There are used 2 potentiometer encoders for a real-time (with teensy constraints) testing of shoulder joint motion manually. These 2 potentiometer encoders send signals to DRV8835 to control the motors in forward and inverse directions. 1 potentiometer initially was used to control motors 1 and 2. This first potentiometer can stop the motor, reverse the direction and switch from motor 1 to motor 2 when the switch button is pressed. The second potentiometer does the same thing but for motors numbers 3 and 4. There are 2 motor drivers, 1 motor driver is for motor 1 and 2 and the second motor driver is for motor 3 and 4.

Also, there we use 3 force sensors to measure the weight on each tendon. These force sensors are calibrated with a calibration factor of -340500 and this is somehow equivalent to measuring the mass of real objects with some weight limit. These force sensors are

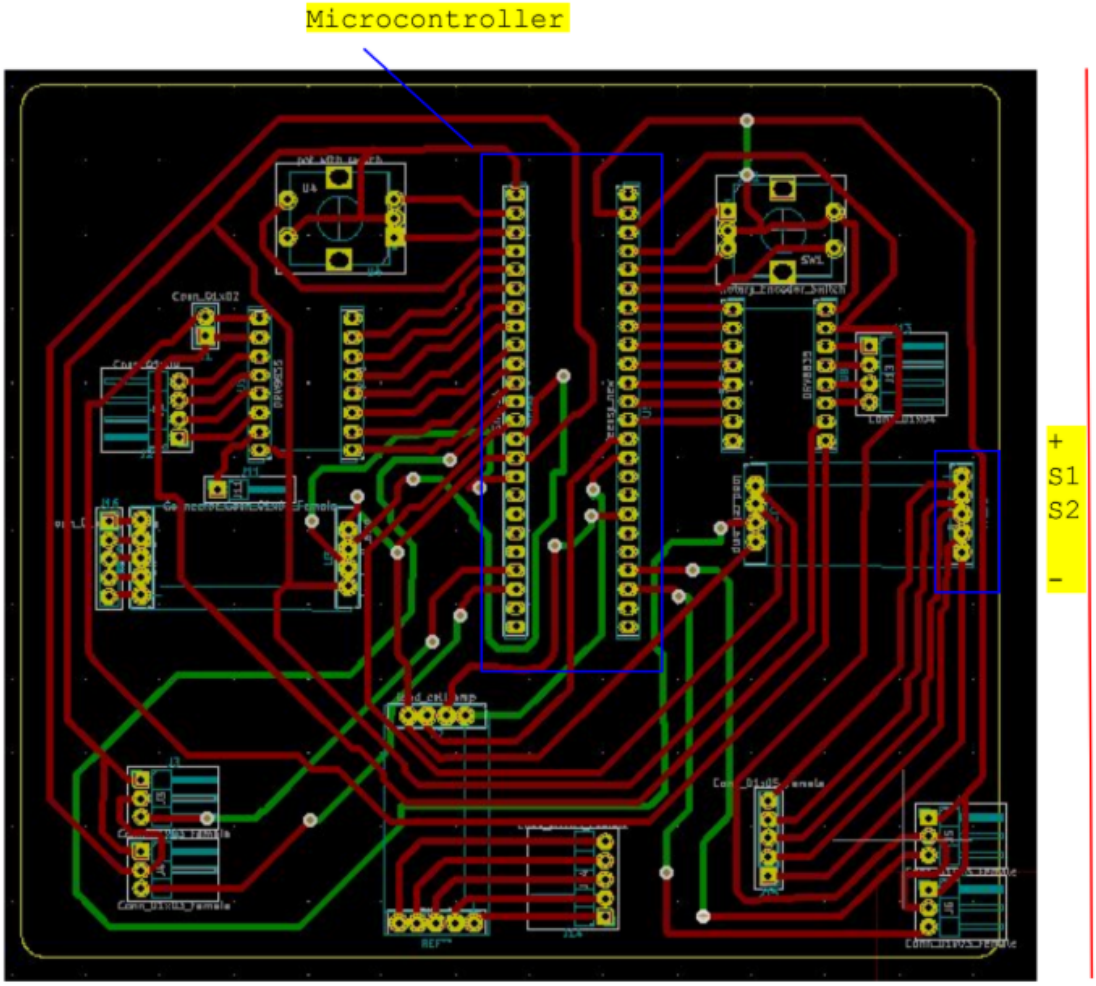

Figure 2-5: PCB design in KiCAD

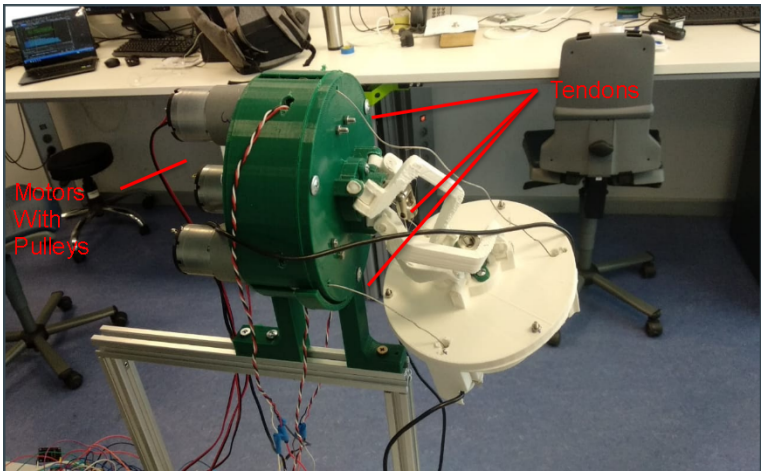

Figure 2-6: Proof of concept for a 3-DOF shoulder joint (only two actuated) realized in PLA.

working parallel with the potentiometers and motor drivers. There is a 1-2 seconds delay when the force sensor sends its value to the serial monitor, this is probably because there are 7 sensors and 2 motor drivers and 2 potentiometer encoders are working at the same time and each of them writes data on the serial monitor. But this should be a solvable problem because the system works fine with 2 potentiometer encoders, 2 motor drivers, and 3 force sensors.

Also, 4 magnetic encoders sense rotation of actuators in clockwise and anticlockwise directions and send them to Arduino as a PWM signal. These encoders count relatively precisely when not mapping their output meaning these magnetic encoders work as non-calibrated more precisely. Their signal ranges from 0 to 1000 and it works well, showing the counts when the motor rotates. But, we mapped 0 to 1000 signal to be 0 to 360, so that we could associate with a real circular motion, however, we see some jump because the actuator actuating range is too small for 360 degrees measurement, but this jump is justified, for example when a little bit actuating we see from 0 going to 22. However, this problem has been solved by removing the delay during reading the encoder. This, on the other hand, works very smoothly when we do not map the magnetic encoder signal and leave it to be 0 to 1000. Also, these magnetic encoders work simultaneously with 3 force sensors and 2 motor drivers, and 2 potentiometers. This all because of teensy's all pin interrupts capabilities.

When 3 tendons work in parallel, the force that is applied to the actuated tendon should be opposite compared to the other two tendons that have to loosen/tighten. (At

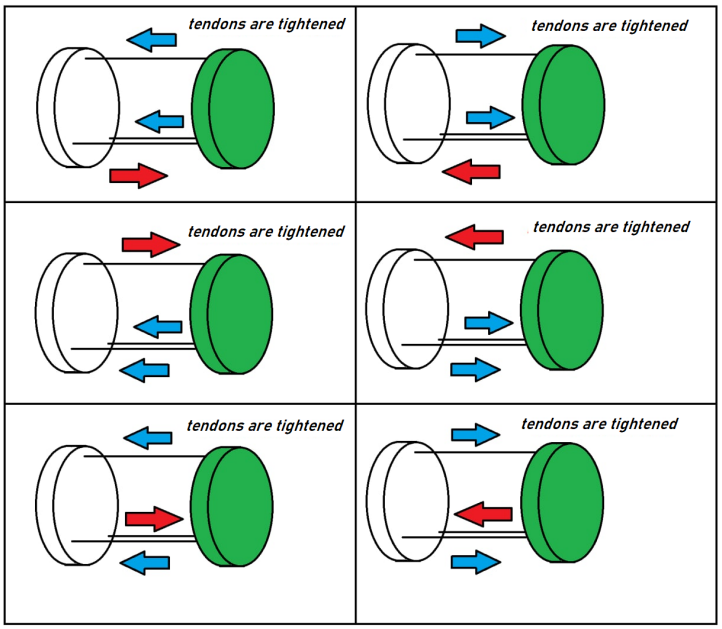

Figure 2-7: Concept explanation [4]

this point tendon simulation is not scripted, but controlled by Newton's laws). Because in V-rep, there was plotted the movement of one tendon (not in parallel). In the physical simulation, in the python program, each button was programmed to make 3 motors work at the same time. to make the best movement, it was decided to make all tendons tightened, and move tendons like that, otherwise, it created a dangerous movement when tendons were exceeding their limits or loosen too much.

The motherboard with teensy 3.5 demonstrated in Figures 2-8 includes microelectronics and wires for the force sensors, motor drivers, position encoders, actuators, and all these elements except the shoulder joint and actuators. As long as the interface works as expected and can control the shoulder joint. Now many wires can be substituted with the PCB which is manufactured by us. The strategy is that first, the PCB plate contacts have to be tested with a multimeter. It was working well, so now we can start soldering starting from the microcontroller which in this case is the teensy 3.6. After making the teensy 3.6 easily connectable and disconnectable using male & female connectors and soldering the female connectors to the PCB plate, it can be checked if it is working. After seeing it is working as expected, each component was soldered one after another and after each component was soldered it was checked if the component is working integrated with the teensy properly. The following figure depicts how the last version of the soldered PCB looks like.

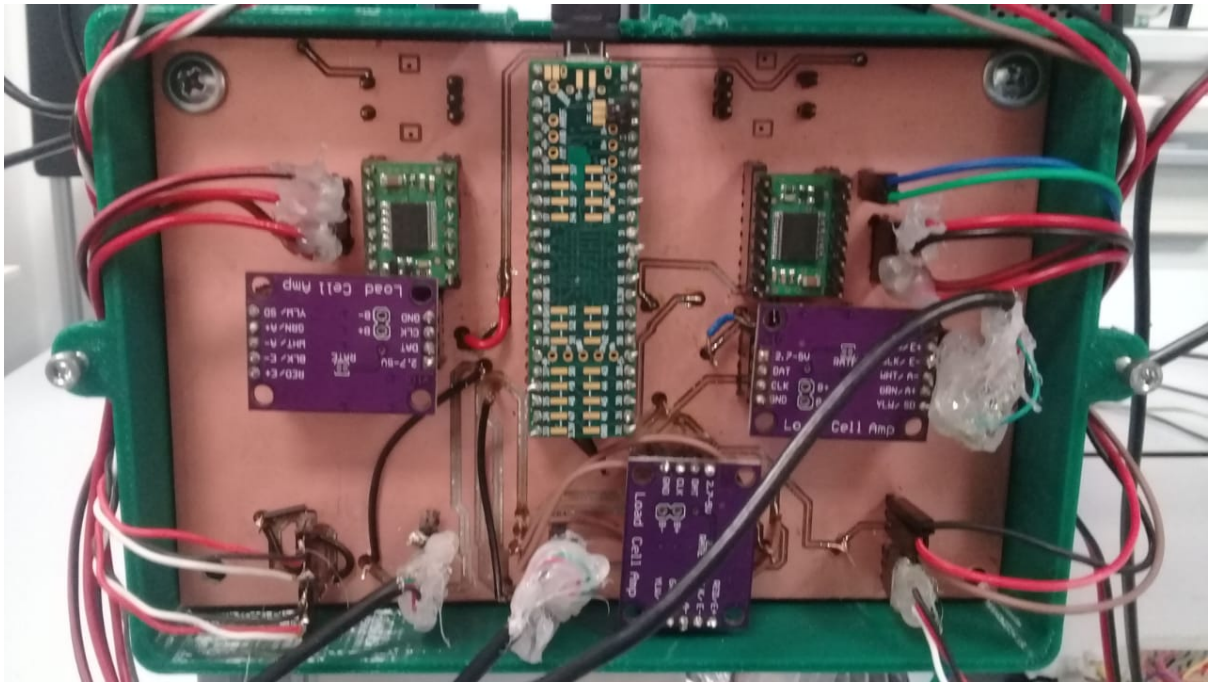

Figure 2-8: Motherboard which incorporates a Teensy 3.5, force sensors amplifiers, motor controllers, and rotary encoders.

2.4 GUI to manage the enarthrosis

When running the python interface code, an updated interface can be seen. When the gauge is rotated and the actuator is chosen; for example, the actuator 1 and then there the click button can be pressed to send value to the microcontroller. The tendon which is attached to actuator 1 gets loosened or tightened depending on which direction the gauge is rotated. This logic is the same for actuator 2 and actuator 3. Also, the indicator lamps show the color green if the corresponding actuator is actuated. Also, the interface can read the force sensor values instead of reading the position encoder values. position encoder values can be read by default. The actuators can be turned off in case of an emergency using one button, so all the actuators shut down immediately. Also, the table below shows the values of position encoders or force sensors. The dynamic graph represents the position encoder or force sensor values while they are changing dynamically depending on time. The table values also correspond to this time. 2 or 3 actuators can be activated at the same time, but it is recommended to be careful with the mechanical limit of the shoulder so that the tendons' or live PLA components' do not get shortened.

The dial in the interface is used to represent the negative and positive values from -100 to 100. The button "send the value" is used to send the values in the dial. "Turn

Figure 2-9: Concept explanation: interface

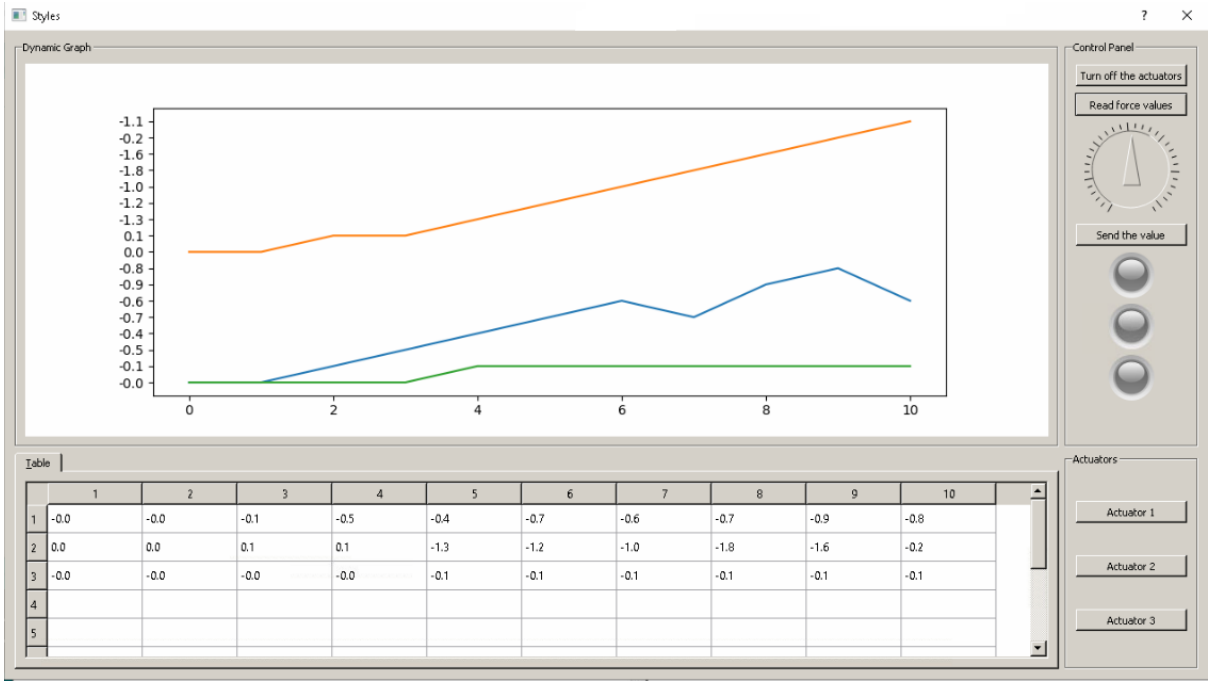

Figure 2-10: Example of an interface to manage the spheroid joint (the graph shows the force sensor measurements)

off the actuators" is used to turn off all the actuators. "Actuator 1" actuates actuator 1 and deactivates it back. "Actuator 2" actuates the actuator 2 and deactivates back and so on. And the red indicator is turned on when one of the actuators is activated. The graph for the encoder represents the corresponding position values received from each of the magnetic encoders that are installed over the head of the actuators such that the encoder can increase or decrease its values depending on the motorhead direction.

Figure 2-6 shows how our control mechanism looks like:

Also, 2-10, 2-11 are an old and new interface to control the shoulder joint:

Figure 2-12 shows the updated interface. In this interface, the "act roll" is corresponded to the motor which rotates the shoulder joint in the yaw axis. When for example we click on "act left" one motor moves on the reverse and 2 motors moves forward to make a synchronous movement.

2.5 Force and position control

The following is the tendon length and position or angle relationship. The figure below 2-14 shows position and force control scheme.

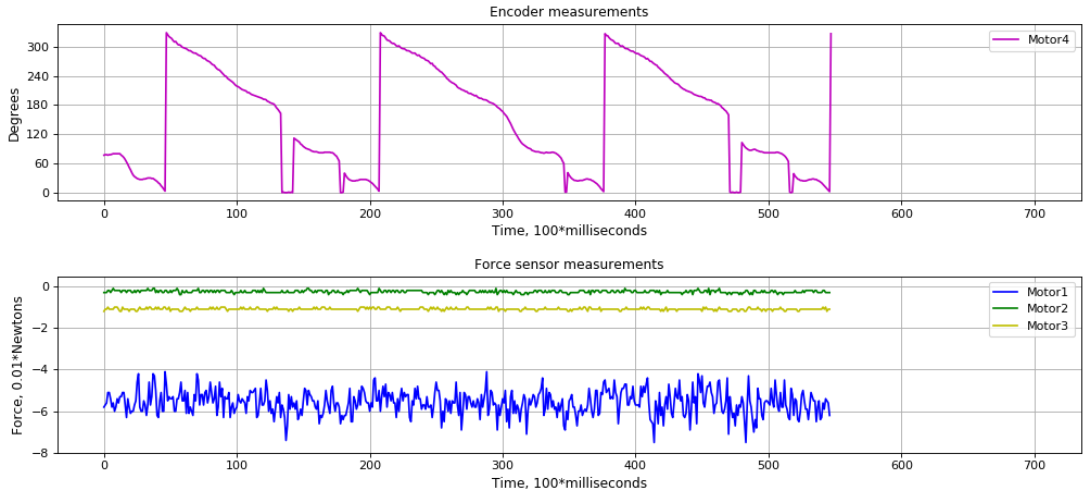

Figure 2-11: Position encoder forward 3 complete rotations

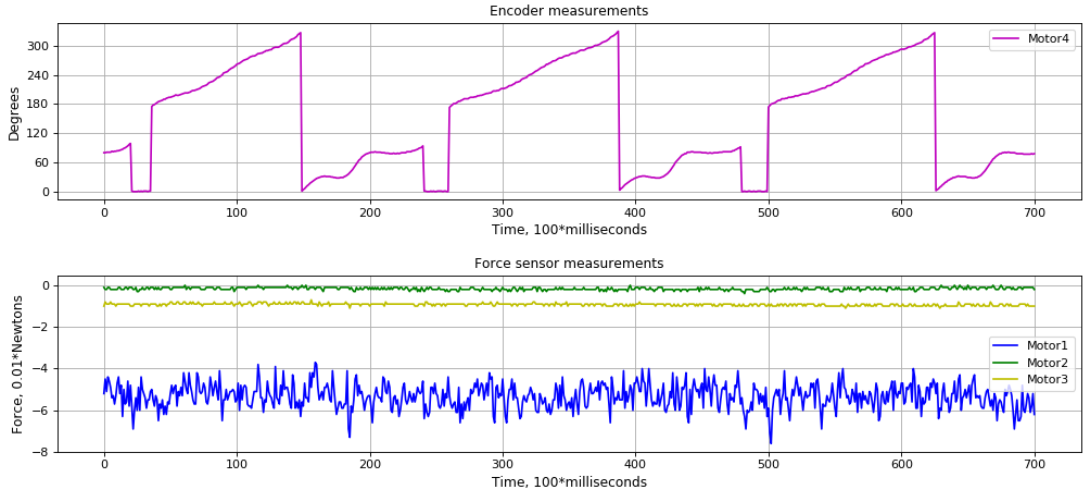

Figure 2-12: Position encoder reverse 3 complete

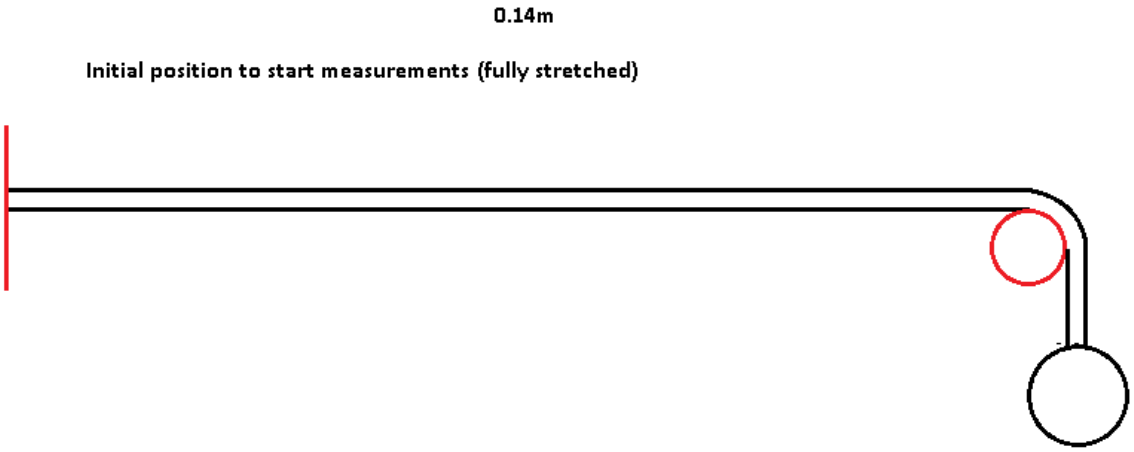

Figure 2-13: Position or angle relationship

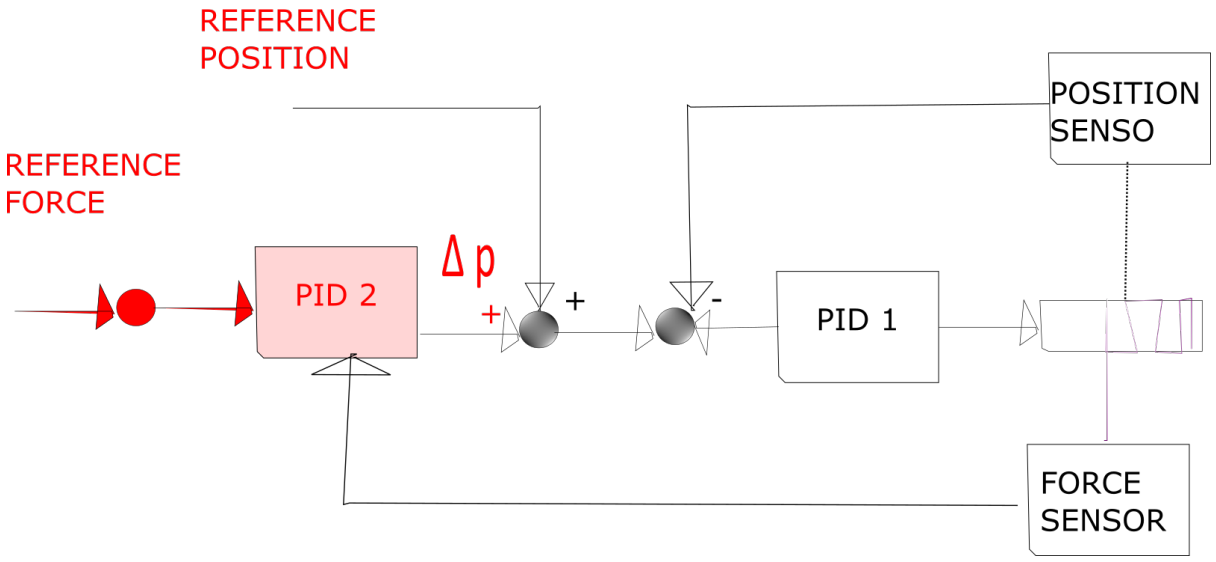

Figure 2-14: Force and position control

Chapter 3

Articulatio spheroidea model and simulation

3.1 Detailed design of three degrees of freedom manipulator

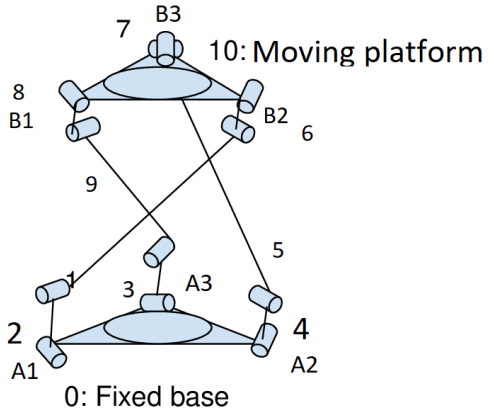

Figure 3-1: Architecture of Stewart platform with rotational joints

L is the independent loop in this closed kinematic chain and n=11 links, the number of binary joints j=12. Using Euler equation:

L=j-n+1

L=12-11+1=2

And for any parallel manipulator the number of limb:

m=L+1=2+1=3

However, the DOFs of this manipulator are 2, F=2.

3.2 Kinematics for the primary leg or limb:

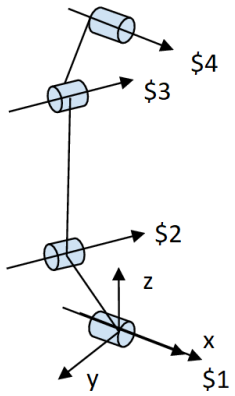

Figure 3-2: Screw method

s_i	s_{oi}
(1, 0, 0)	$(a_0, 0, 0)$
(0, 1, 0)	$(a_0, 0, a_1)$
(0, 1, 0)	$(a_0, 0, a_1 + a_2)$
(1, 0, 0)	$(a_0, 0, 2a_1 + a_2)$

Note, here we use a notation: $s\theta = s$ and $c\theta = c$

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_1 & -s_1 & 0 \\ 0 & s_1 & c_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} c_2 & 0 & s_2 & a_0(s_2 - 1) - a_1 s_2 \\ 0 & 1 & 0 & 0 \\ -s_2 & 0 & c_1 & -a_0 s_3 + (a_1 + a_2)(s_2 - 1) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} c_3 & 0 & s_3 & a_0(s_3 - 1) - (a_1 + a_2)s_3 \\ 0 & 1 & 0 & 0 \\ -s_3 & 0 & c_3 & a_0 s_3 - (a_1 + a_2)(s_3 - 1) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_4 & -s_4 & -(2a_1 + a_2)s_4 \\ 0 & s_4 & c_4 & (2a_1 + a_2)(1 - s_4) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$A_{14} = A_1A_2A_3A_4$ gives us a direct kinematics solution for 1 limb or leg.

$$A_{14} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & p_x \\ a_{21} & a_{22} & a_{23} & p_y \\ a_{31} & a_{32} & a_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$p_x = s_2(a_0s_3 - (a_1 + a_2)(s_3 - 1)) - s_2(a_1s_2 - a_0(s_3 - 1)) + a_0(s_3 - 1) - s_{2-3}(s_4 - 1)(2a_1 + a_2)$

$p_y = s_1s_2(a_1s_2 - a_0(s_3 - 1)) - s_1(a_1 + a_2 - a_0s_3) - s_1s_4(2a_1 + a_2) - (s_4 - 1)(s_1s_2s_3 - s_2s_3s_1)(2a_1 + a_2) - s_2s_1(a_0s_3 - (a_1 + a_2)(s_3 - 1))$

$p_z = s_1(a_1 + a_2 - a_0s_3) - s_1s_2(a_1s_2 - a_0(s_3 - 1)) + (s_4 - 1)(s_1s_2s_3 - s_1s_2s_3)(2a_1 + a_2) - s_4s_1(2a_1 + a_2) + s_1s_2(a_0s_3 - (a_1 + a_2)(s_3 - 1))$

3.3 Inverse Kinematics

We assume that inverse kinematics's for one limb of the shoulder joint should be the same for the other limbs.

$$p = A_1A_2A_3A_4p_0$$
$$A_2^{-1}A_1^{-1}p = A_3A_4p_0$$

$$p_0 = \begin{bmatrix} a_0 \\ 0 \\ 2a_1 + a_2 \\ 1 \end{bmatrix}$$

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

$$u_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$u = R_1 R_2 R_3 R_4 u_0$$

$$u = \begin{bmatrix} c_{2-3} \\ -s_{23} s_1 \\ s_{23} s_1 \end{bmatrix}$$

$$u_y^2 + u_z^2 = s_{23}^2$$

$$\theta_{23} = \pm\sqrt{u_y^2 + u_z^2}$$

$$\tfrac{u_y}{u_z} = -tan\theta_1$$

$$\theta_1 = -atan2(\tfrac{u_y}{u_z})$$

By equating

$$A_2^{-1} A_1^{-1} p \text{ and } A_3 A_4$$

using MATLAB results:

$$\theta_{4,1} = -\tfrac{\pi}{4} - \arcsin(\tfrac{\sqrt{\frac{1}{2}}(p_y s_1 + p_z s_1)}{2(2a_1 + a_2)})$$

using MATLAB results and

$$\theta_{4,2} = \tfrac{3\pi}{4} + \arcsin(\tfrac{\sqrt{\frac{1}{2}}(p_y s_1 + p_z s_1)}{2(2a_1 + a_2)})$$

$$u_x = c_{2-3}$$

$$s_{2-3} = \pm\sqrt{1 - u_x^2}$$

$$\theta_{2-3} = atan2(\pm\sqrt{1-u_x^2}, u_x)$$

$$\theta_{2-3} + \theta_{2+3} = atan2(\pm\sqrt{1-u_x^2}, u_x) \pm \sqrt{u_y^2+u_z^2}$$

$$2\theta_2 = atan2(\pm\sqrt{1-u_x^2}, u_x) \pm \sqrt{u_y^2+u_z^2}$$

$$\theta_{2,1} = \frac{atan2(\pm\sqrt{1-u_x^2}, u_x)+\sqrt{u_y^2+u_z^2}}{2}$$

$$\theta_{2,2} = \frac{atan2(\pm\sqrt{1-u_x^2}, u_x)-\sqrt{u_y^2+u_z^2}}{2}$$

$$\theta_{2-3} - \theta_{2+3} = atan2(\pm\sqrt{1-u_x^2}, u_x) \pm \sqrt{u_y^2+u_z^2}$$

$$-2\theta_3 = atan2(\pm\sqrt{1-u_x^2}, u_x) \pm \sqrt{u_y^2+u_z^2}$$

$$\theta_3 = \frac{atan2(\pm\sqrt{1-u_x^2}, u_x)\pm\sqrt{u_y^2+u_z^2}}{-2}$$

$$\theta_{3,1} = \frac{atan2(\pm\sqrt{1-u_x^2}, u_x)+\sqrt{u_y^2+u_z^2}}{-2}$$

$$\theta_{3,2} = \frac{atan2(\pm\sqrt{1-u_x^2}, u_x)-\sqrt{u_y^2+u_z^2}}{-2}$$

3.4 Inverse Kinematics - Geometric Method (based on [7], see some related paper from our Folgheraiter's lab: [8])

Formulation of dependency

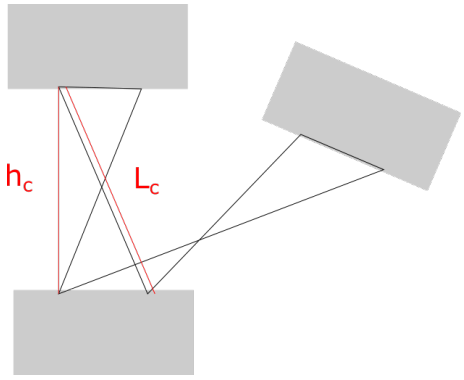

Figure 3-3: Formulation of dependency

$$tg(\psi)y_c = x_c - h_0 * tg(\psi)$$

$$tg(\psi) = \frac{x_c}{h_0 + y_c}$$

$$r = x_c * sin(\psi)$$

$$sin(\frac{\theta}{2}) = \frac{k}{\frac{w}{2}}$$

$$k = \frac{w}{2} * sin(\frac{\theta}{2})$$

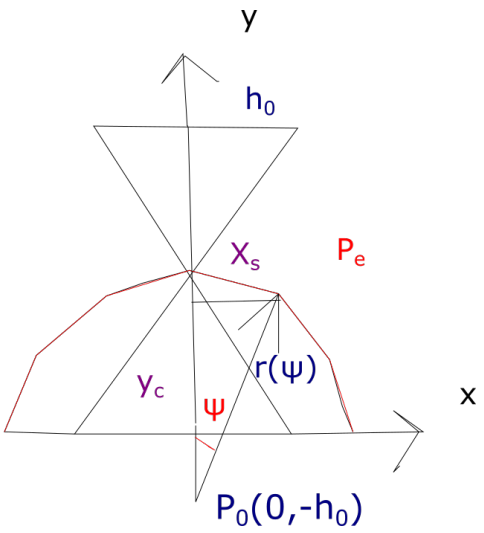

Figure 3-4: geometry of movement

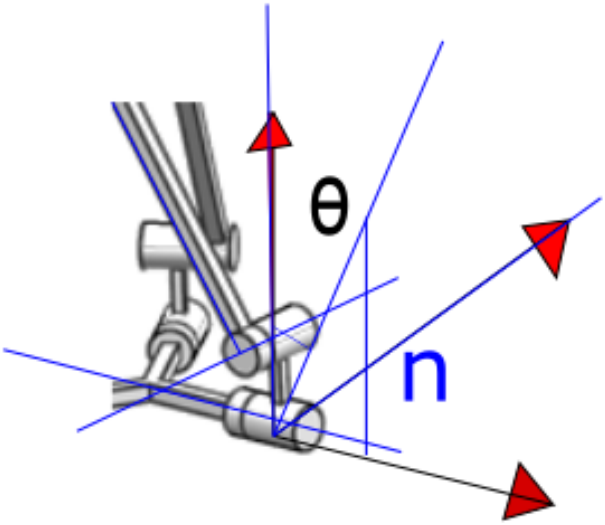

Figure 3-5: Geometrical solution attempt for the one limb adapted from [7]

$$z_0 = x_a = f + s_2 * m + s_3 n - f$$

$$x_0 = y_a = s_1 * n + s_4 * k$$

$$y_0 = z_a = c_4 * n + c_2 * m + s_3 * n + c_4 * k$$

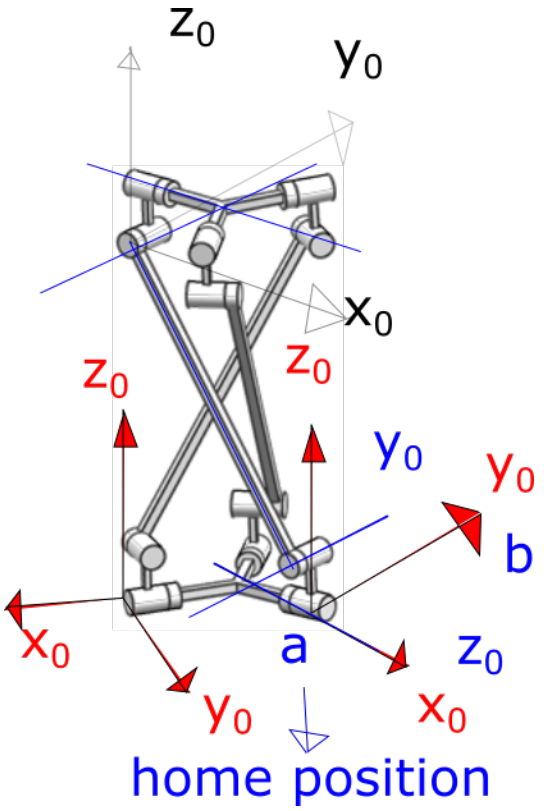

Figure 3-6: Geometrical solution attempt for the one limb adapted from [7]

3.5 MATLAB simulation of PM

We have simulated one limb of the parallel manipulator in MatLab as can be shown in figures 5 and 7. According to our thoughts, the other two legs will follow the first one. By changing the angle of joints in the first leg it is possible to change the position and orientation of the end-effector.

3.6 Kinematics of PM using Geometrical Method

Figure 6 shows a parallel manipulator that is made up of a moving platform, fixed base, and three limbs. Both the moving platform and the fixed base take the form of a tetrahedron. The moving platform is connected directly to the fixed base by an additional limb that does not affect the spatial movement of an end-effector. Stewart platform is considered spatially oriented, thus, will be analyzed as three extensible limbs connected to the moving platform at points B_i and the fixed base at points A_i. Pulleys across the platform can be used to vary the lengths L of the robe to control the motion of the moving platform. For analysis, two Cartesian coordinate systems A(x_0, y_0, z_0) and B(x_1, y_1, z_1)

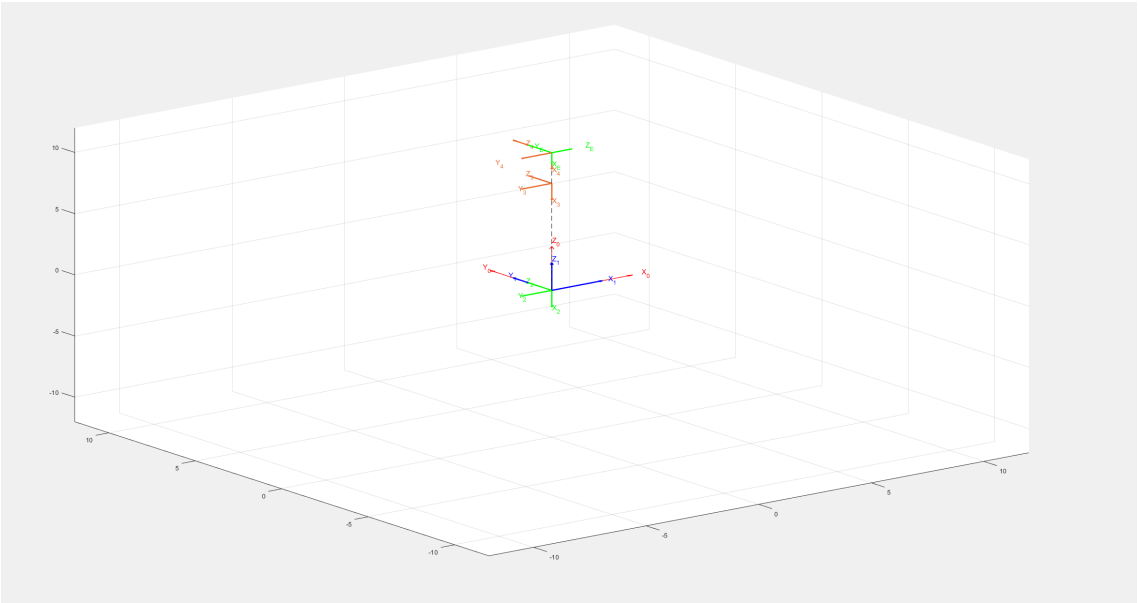

Figure 3-7: MATLAB simulation of PM, the first leg is bent $\frac{\pi}{4}$ (θ angle) and therefore the moving platform is bent $\frac{\pi}{4}$ (ϕ angle)

are attached to the fixed base and moving platform. The origin of the platform is chosen to be at the center of an additional limb. The transformation from the moving frame B to the fixed frame A is described by a 3x3 rotation matrix ${}^{A}R_{B}$. The initial location of the moving frame B coincides with the fixed frame A and the final position is obtained by rotation of ϕ about the x-axis, followed by a second rotation of θ about the y-axis. Our platform does not move about the z-axis which is fixed by an additional limb. The resulting rotation matrix is given by:

$$ {}^{A}R_{B} = R_x * R_y = \begin{bmatrix} c_\phi & 0 & -s_\phi \\ s_\phi * s_\theta & c_\theta & c_\phi * s_\theta \\ c_\theta * s_\phi & -s_\phi & c_\phi * c_\theta \end{bmatrix} $$

If we assume that the position vector a of point B_i concerning the moving frame is given by

$$ {}^{B}b_i = \begin{bmatrix} b_{i_u}, & b_{i_v}, & b_{i_w} \end{bmatrix}^T $$

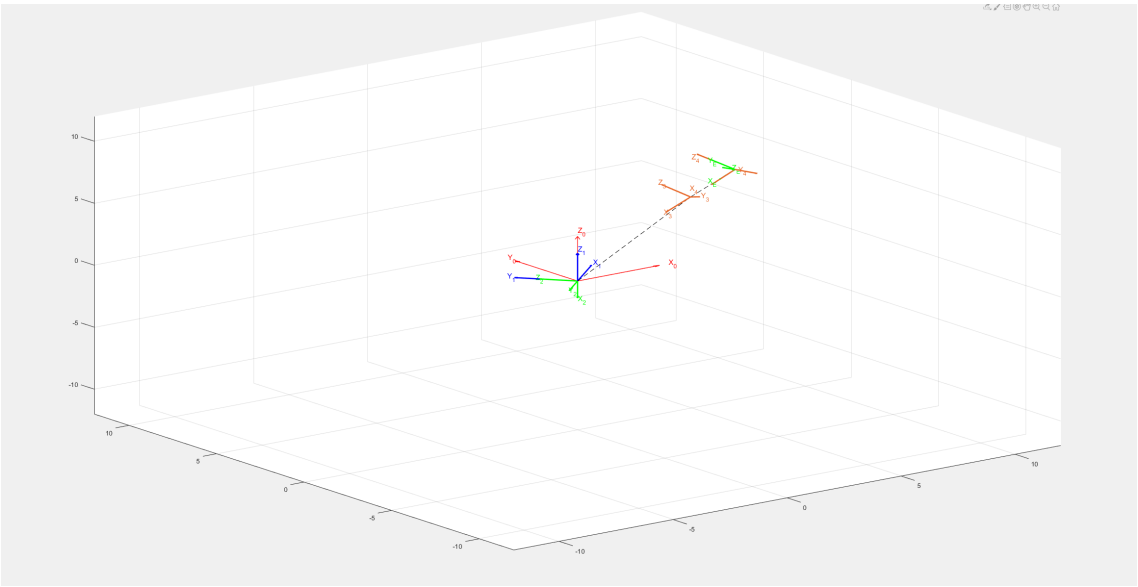

Figure 3-8: MATLAB simulation of PM, the first leg is in the base position

Then the position vector of point B_i in the fixed frame A is

$$\mathbf{b_i} = {}^A R_B * {}^B b_i = \begin{bmatrix} b_{i_u} * c_\phi - b_{i_w} * s_\phi \\ b_{i_u} * s_\phi * s_\theta + b_{i_v} * c_\theta + b_{i_w} * c_\phi * s_\theta \\ b_{i_u} * c_\theta * s_\phi - b_{i_v} * s_\phi + b_{i_w} * c_\phi * c_\theta \end{bmatrix}^T$$

Also, let the position vector of point A_i concerning the fixed frame A be given by

$$\mathbf{a_i} = \begin{bmatrix} a_{i_x}, & a_{i_y}, & a_{i_z} \end{bmatrix}^T$$

Then a loop-closure equation can be written for limb i as $\mathbf{d_i} = \mathbf{b_i} - \mathbf{a_i}$, where $\mathbf{d_i} = \overline{A_i B_i}$. Dot multiplication of this equation gives as $d_i^2 = \mathbf{a_i^2} + \mathbf{b_i^2} - \mathbf{2a_i^T b_i}$

3.7 3D design components of PM

The 3D design of the Stewart platform is shown in figure 4. The numbers in the figure represent different parts of that manipulator. 1 - fixed base, 2- base and limb connection, 3-rotational joint, 4-link, 5-rotational joint, 6-union, 7- platform and limb connection, 8-moving platform.

It was understood that tendons were not separate models, but included in the system by torques (pull forces) segmented by 120 degrees.

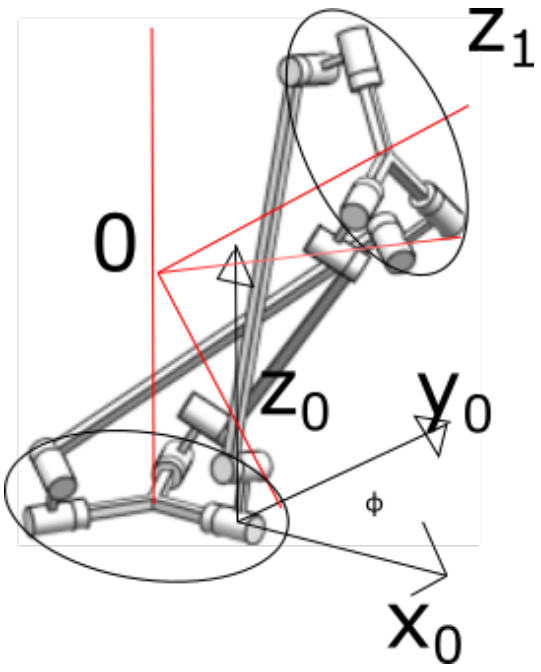

Figure 3-9: 3-DOF parallel manipulator with coordinates adapted from [7]

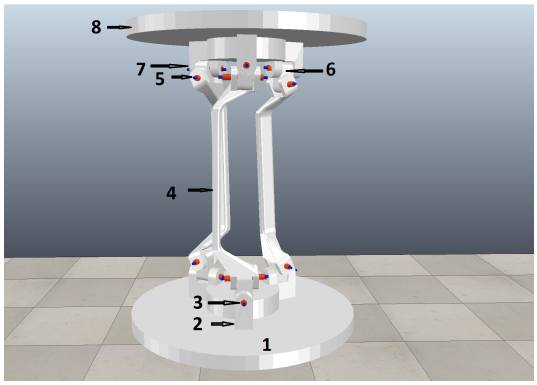

Figure 3-10: Low inertia high-stiffness parallel manipulator

Figure 3-11 shows the shoulder joint simulated model (in v-rep, older version)

The figure 3-12 shows how the shoulder joint is bent in a v-rep simulation environment. a) model is controlled with x and y angles, the second b) with tendons. **Torque and Speed formula** Torque $=\frac{Power}{speed}$

or

$\tau = \frac{P}{w}$ Where P is the power of the actuator,

τ is the torque,

w is the angular speed or velocity.

Figure 8 represents problem older version of our shoulder joint:

When in vertical orientation our shoulder model's tendons look like these shoulders

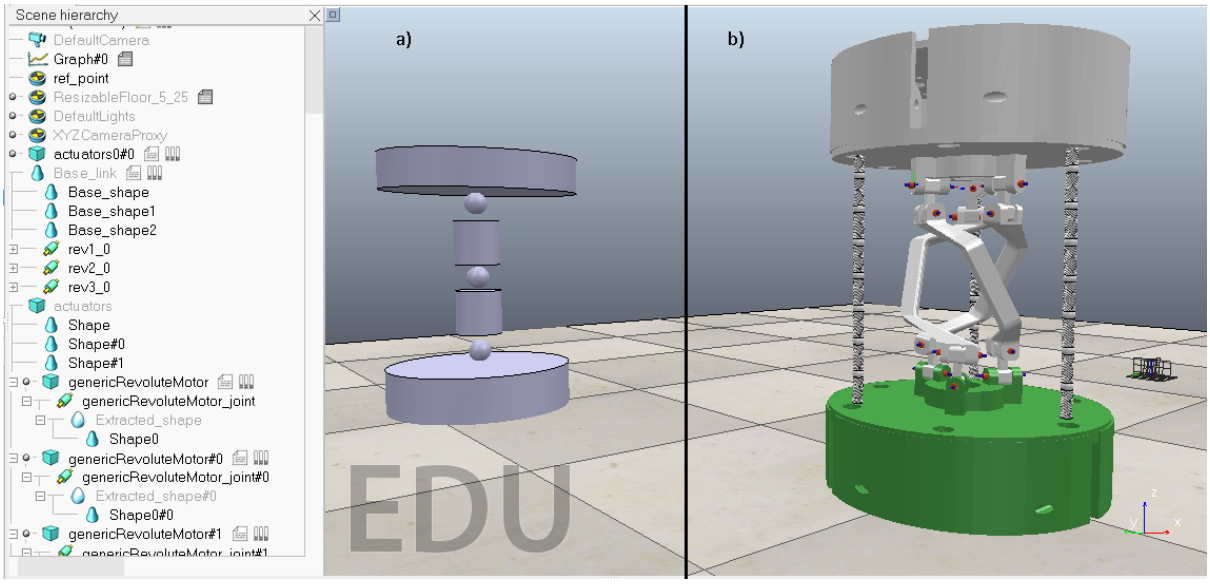

Figure 3-11: V-rep simulation, a) the model which is controlled with x and y angles b) simulation of cotyloid joint

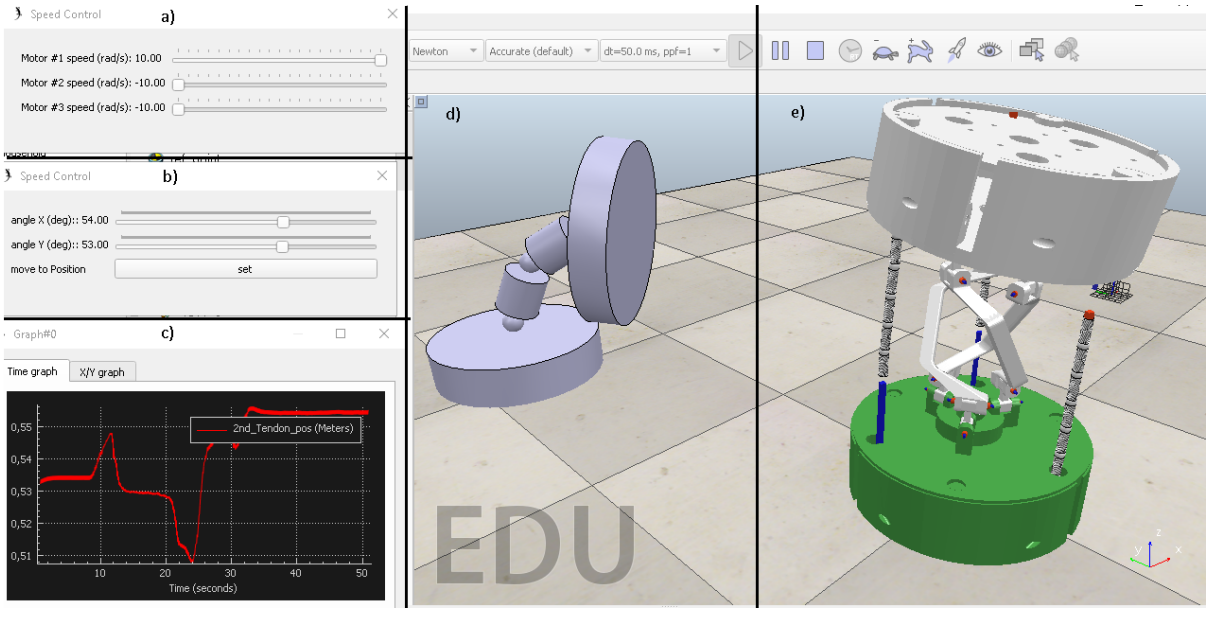

Figure 3-12: V-rep simulation, a) speed control of the model which is controlled with x and y angles b) speed control of the tendon actuated shoulder joint simulation c) graph which is obtained from shoulder joint simulation d) the model which is controlled with x and y angles is bent to a target location e) simulation of the tendon actuated shoulder joint where it is bent to a target location

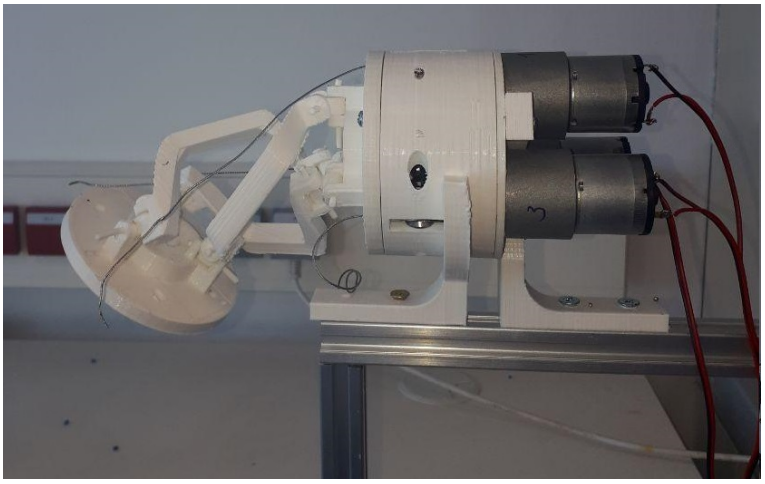

Figure 3-13: Shoulder joint platform older design version

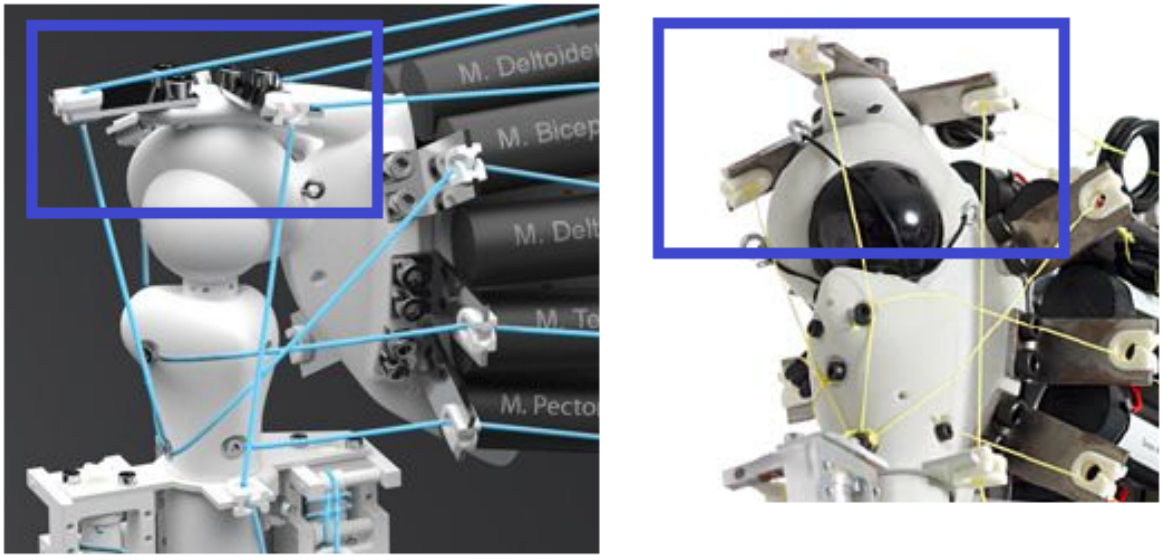

Figure 3-14: When the tendons in vertical orientation [4]

holders, but links instead of the spherical joint creates additional strength.

3.8 A V-rep model which imitates the spherical joint and a workspace of an imitating v-rep model

The figure **??** shows an approach to imitating the spherical joint. Due to the difficulty of replicating the tendon in v-rep, we have built a v-rep model which is a simulation of the real hardware. However, the workspace shows that the imitation model differs from the real hardware.

The figure **??** depicts the 3d represented workspace of the v rep model which imitates

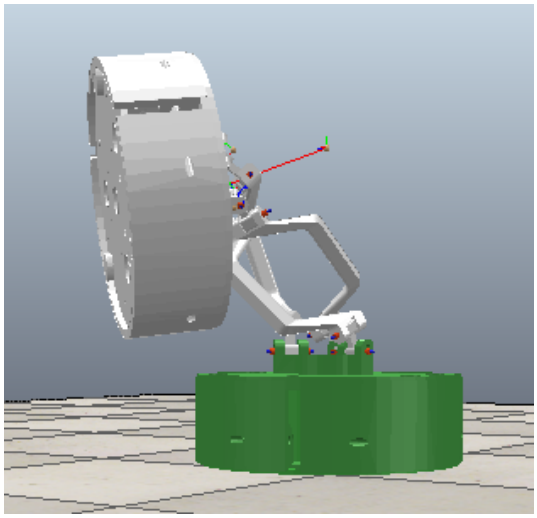

Figure 3-15: A V-rep model which imitates the spherical joint bent to 90 degrees

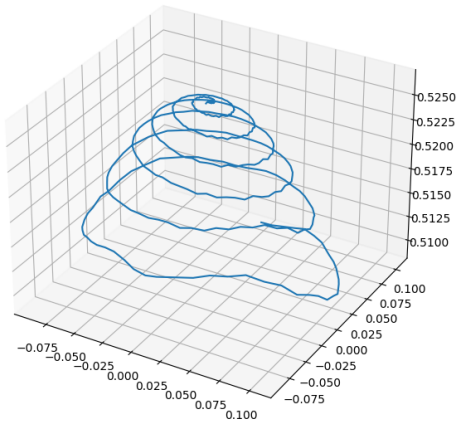

Figure 3-16: Workspace of a imitating v-rep model

the spherical joint.

The next figure **??** represents the 2d horizontal workspace of the v pep model which imitating the spherical joint.

3.9 EEG Control

So, we have one joint with 3 (2-degree freedom, because there is probably 1 more actuator needed to make it 3) degrees of freedom. In the beginning, we work in a simulation, then we start working with the real device and EEG cap.

For the EEG integration part, we are going to use a motor imagery dataset to get

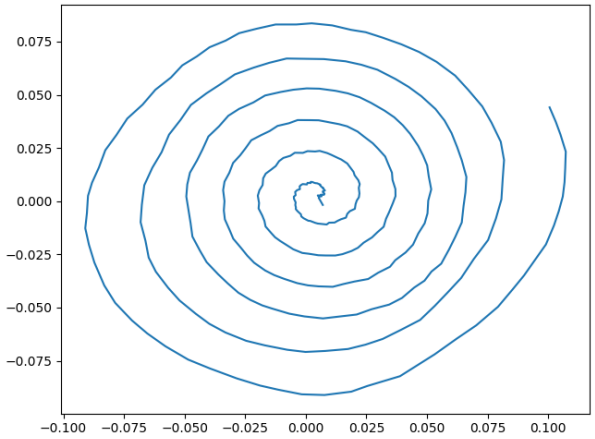

Figure 3-17: Workspace of a imitating v-rep model represented in 2d

concentration on shoulder movement duration times (in the source code they are stated as x, y z directions) and set on actuators 1, 2, and 3 correspondingly to move the shoulder joint. So, firstly we will test the system without EEG with the motor imagery dataset and when we can make it work we will use EEG to directly do the control. We will also focus on implementing prediction. In this code, for example, we will only have a com port for the shoulder teensy 3.5, because the IP address is here for the ur5. The code was my first attempt to adapt the ur5 code to our shoulder, but we forgot that we move in the reverse direction when the value is negative but not the integer value (integer value -1 -2 -3 stops the motors) type which we sent with leaving a space. So, we will try to update this code soon.

The software part using EEG data acquired from the brain from an experiment in [9] was adapted so that it controls the spherical joint that we have discussed in this thesis instead of the UR-5 manipulator. So, the signals that are useful to rotate the spherical joint in our shoulder were used for the software to send it to the teensy microcontroller to see the movement in the spherical joint. However, testing the system with EEG cap and the software is left as future work.

3.10 availability of resources like equipment, consumables, library, and computation to unravel the matter

All resources are available for the research.

3.11 Approach and methods to resolve the matter

We are working for this shoulder and have some outcome which is a robotic shoulder that can be controlled using the interface. We will also be working with V-rep to simulate the shoulder joint to be able to test all the movements before we finish the whole development (see some related paper from our university: [18]), but it now also works in the real life.

3.12 Research challenges

The first problem is that it is difficult to make a 3-DOF shoulder joint. The human shoulder joint is not easy to replicate, because as an organism we have everything that supports our joints, lubrication, sphere-like mechanisms, and everything that was evolved during human evolution time. The second problem is with the tendon-driven mechanism. If we make it tendon driven, tendons may rotate with a larger radius and the joint is rotated anticlockwise instead of clockwise.

The third problem is a collision between the objects. For example, the link should not touch the platform during the activation process of manipulator motors. In this case, it is important to build the right geometry of objects of the manipulator. The fourth problem is a real-time signal transfer problem. And measuring these systems with sensors is also difficult if you aimed to make a low-cost design. For example, in our case, the signal from force sensors comes to the teensy 3.6 with a delay of 1-2 seconds. But, this problem we guess is solvable. So, this problem probably is a real-time signal transfer problem.

The fifth problem is the integration problem. As a whole system all sensors, actuators, and signals from the controller should work smoothly. And the interface should be working synchronously as well. This is one of the challenging parts of the shoulder joint design.

The sixth problem is the design problem. As long as we make the design by ourselves,

we firstly make it tendon-driven with 3 tendons. But, that restricted the 3rd DOF, our mechanism was 2 DOF. Now we updated the mechanism to become 3-DOF, for that we used the 4th tendon. So there are 4 actuators and 4 tendons.

Chapter 4

Testing and results

Achieve full control of the shoulder joint using the interface. Allow the monitoring and control joint position, tendons forces. Develop a V-rep to simulate the shoulder joint so that to test all the movements before we finish the whole development. Feasibility test on using EEG to predict motion. Start the implementation of some parts in aluminum using CNC, All the necessary resources are available in our laboratory or at NU. The total cost of the shoulder joint materials is approximately under 500$. See the table in the appendix.

4.1 Experiment results

We try to connect all the components of the shoulder joint. In the simulation the graph of the voltage shows that the force sensor measurements in 0.01*Newton versus time in a millisecond, therefore we think that as the next step we need to further measure the force sensor values for “Act left 90 degrees” and “Act bottom or top 90 degrees” and the force sensor measurements correspond due to the tensions in the tendon.

Act left about 90 degrees (the green force sensor in the interface experiences the highest tension in the tendon, force sensor value should be multiplied by 0.01N or by 0.01 Newton). The figures 4-3 and 4-4:

Act right about 30 degrees and Act top -45 degrees (the joint rotated to the bottom, the blue experiences the most highest tendon and the orange force sensor the second highest tension (motor which rotates the joint to the right), force sensor value should be multiplied by 0.01 Newton) 4-5 and 4-6:

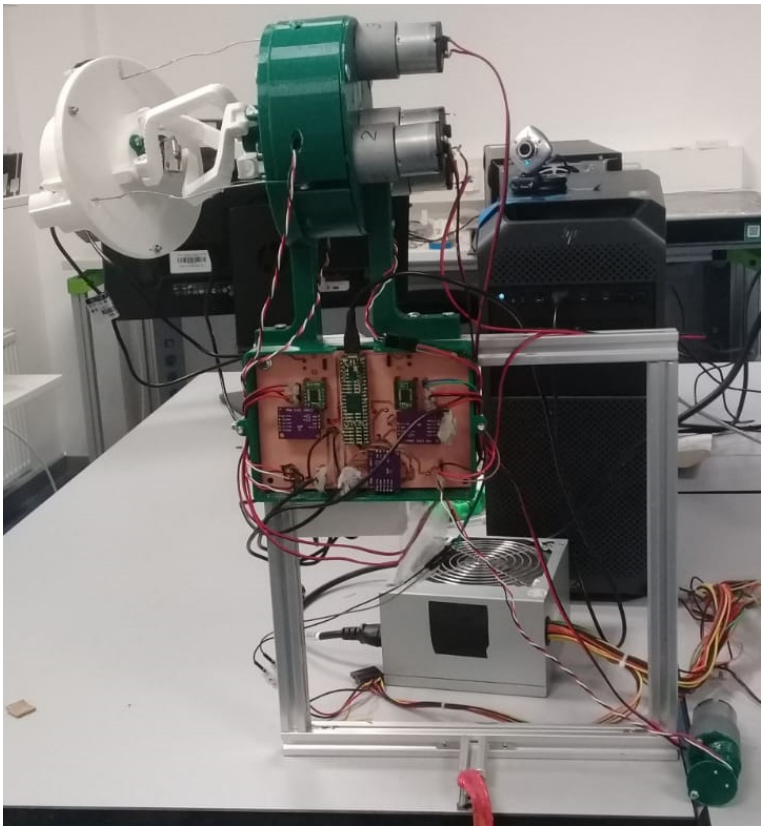

Figure 4-1: Proof of concept for a 3-DOF shoulder joint (only two actuated) realized in PLA. The shoulder joint is bent to the target position.

Act left 75 degrees (the green force sensor in the interface experiences the highest tension in the tendon, force sensor value should be multiplied by 0.01Newton) 4-7 and 4-8:

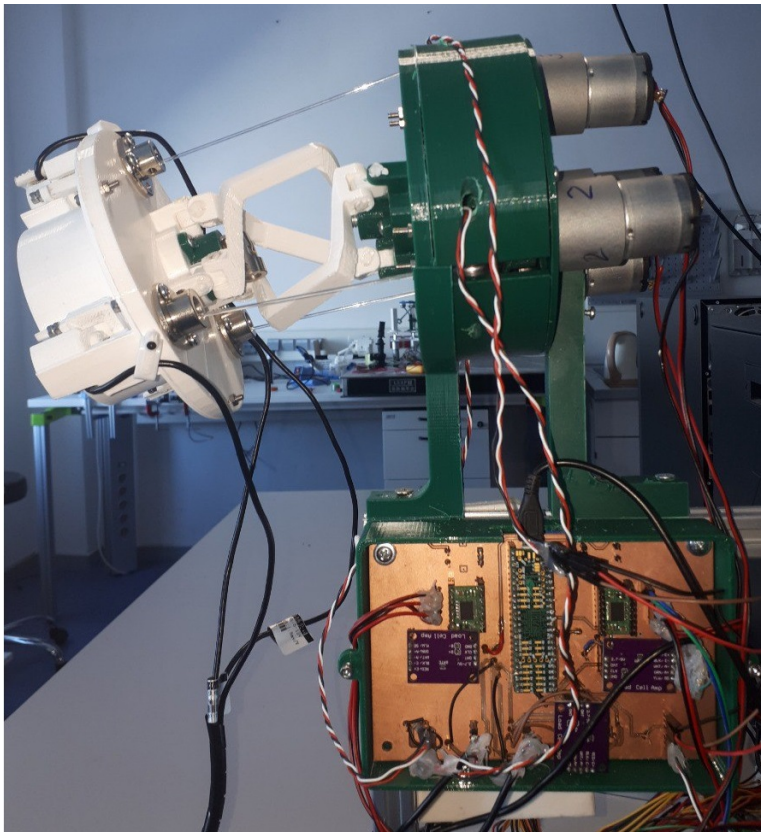

Figure 4-2: Proof of concept for a 3-DOF shoulder joint with an elastic tendon (only two actuated) realized in PLA. The shoulder joint is bent to the target position.

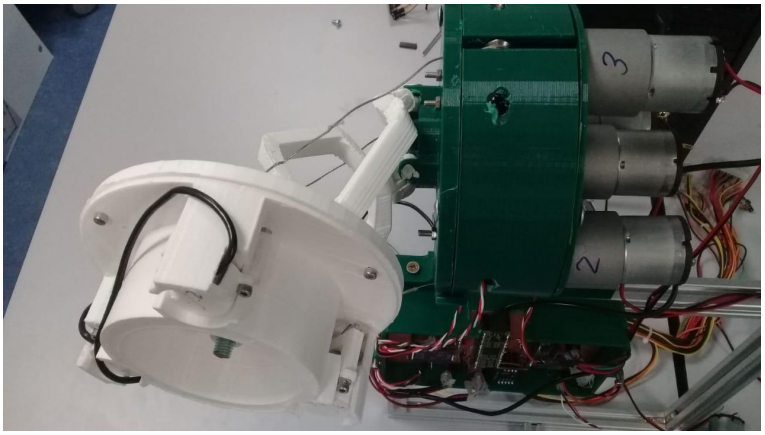

Figure 4-3: Act left about 90 degrees

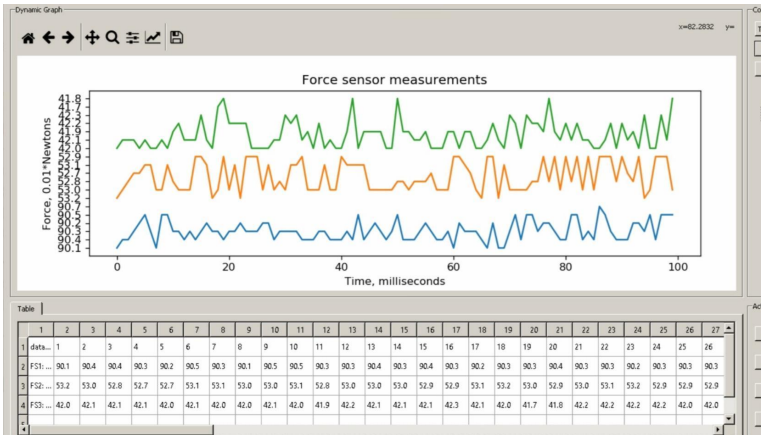

Figure 4-4: Act left about 90 degrees, 3 force sensor measurements

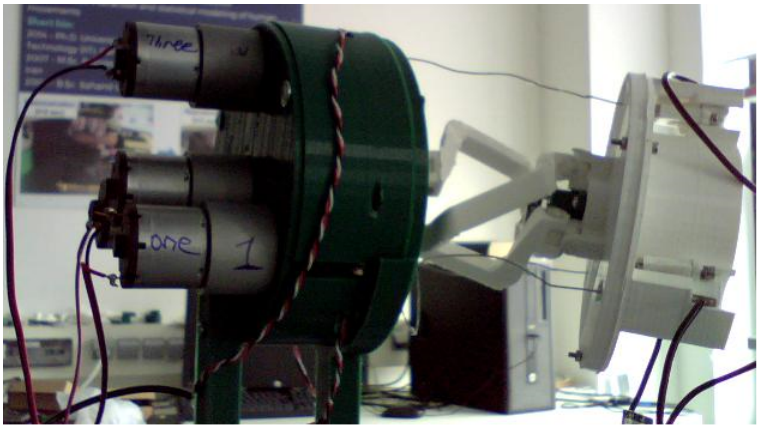

Figure 4-5: Act right about 30 degrees and Act top -45 degrees

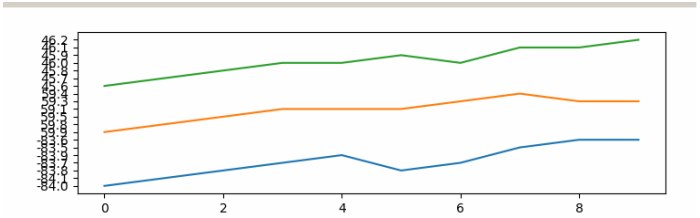

Figure 4-6: Act right about 30degrees and Act top -45 degrees dynamic graph

Figure 4-7: Act left 75 degrees

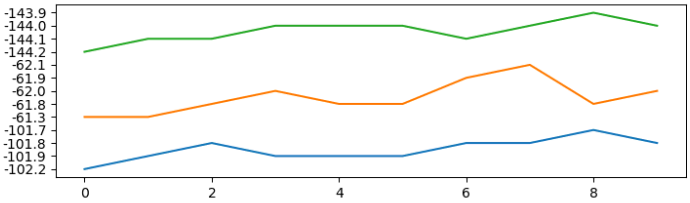

Figure 4-8: Act left 75 degrees dynamic graph

Chapter 5

Conclusion and future work

We have achieved full control of tendon actuated shoulder joint. We have learned that in the v simulation there are limitations to rotate the shoulder joint more. The first one is the successive screw method that does not describe the dynamics of all limbs fully but assumes that two other limbs will follow the first one. The geometric method finds the length d_i of the manipulator which gives us the position and orientation of an end-effector. This method requires further investigation and research in the future.

Appendix A

Tables

A.1 Costs of the materials

Materials	Cost
600 g ABS filament	∼35$
4 x chinese 12 V DC motors	∼20.25$
3 x force sensor amplifiers and force sensors	∼40$
4 x position encoders	∼36$
teensy 3.5, solder, wires, double side circuit board copper	∼35$
other materials (screws, tendon)	∼10$

Appendix B

GUI software

B.1 GUI software in python

Listing B.1: Python interface

```
from PyQt5.QtCore import QDateTime, Qt, QTimer
from PyQt5.QtWidgets import (QApplication, QCheckBox, QComboBox,
QDateTimeEdit,
        QDial, QDialog, QGridLayout, QGroupBox, QHBoxLayout,
        QLabel, QLineEdit,
        QProgressBar, QPushButton, QRadioButton, QScrollBar,
        QSizePolicy,
        QSlider, QSpinBox, QStyleFactory, QTableWidget,
        QTabWidget, QTextEdit,
        QTableWidgetItem, QVBoxLayout, QWidget)
from PyQt5.QtGui import *

import serial
from QLed import QLed

import matplotlib
matplotlib.use('Qt5Agg')
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
from matplotlib.figure import Figure
"""
arduinoData=serial.Serial('COM6',9600)

```

```
def led_on():
    arduinoData.write(b'1')
def led_off():
    arduinoData.write(b'0')
def led2_on():
    arduinoData.write(b'3')
def led2_off():
    arduinoData.write(b'2')
def led3_on():
    arduinoData.write(b'5')
def led3_off():
    arduinoData.write(b'4')
t=0
while(t<3000000):
    t+=1
led3_off()
"""
class MplCanvas(FigureCanvasQTAgg):
    def __init__(self, parent=None, width=5, height=4, dpi=100):
        fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = fig.add_subplot(111)
        super(MplCanvas, self).__init__(fig)

class WidgetGallery(QDialog):
    def __init__(self, parent=None):
        super(WidgetGallery, self).__init__(parent)
        styleComboBox = QComboBox()

        styleLabel = QLabel("Style:")

        self.useStylePaletteCheckBox =
        QCheckBox("&Use style's standard palette")
        self.useStylePaletteCheckBox.setChecked(True)
        disableWidgetsCheckBox = QCheckBox("&Disable widgets")
        self.createTopLeftGroupBox()
        self.createTopRightGroupBox()
        self.createBottomLeftTabWidget()
        self.createBottomRightGroupBox()
        self.createProgressBar()
```

```
62
63          topLayout = QHBoxLayout()
64
65          mainLayout = QGridLayout()
66          mainLayout.addWidget(self.topLeftGroupBox, 1, 0)
67          mainLayout.addWidget(self.topRightGroupBox, 1, 1)
68
69          mainLayout.addWidget(self.bottomLeftTabWidget, 2, 0)
70          mainLayout.addWidget(self.bottomRightGroupBox, 2, 1)
71          self.setLayout(mainLayout)
72          self.setWindowTitle("Styles")
73          self.changeStyle('Windows')
74
75      def changeStyle(self, styleName):
76          QApplication.setStyle(QStyleFactory.create(styleName))
77          self.changePalette()
78      def changePalette(self):
79          if (self.useStylePaletteCheckBox.isChecked()):
80              QApplication.setPalette(QApplication.style().
81              standardPalette())
82          else:
83              QApplication.setPalette(self.originalPalette)
84      def advanceProgressBar(self):
85          curVal = self.progressBar.value()
86          maxVal = self.progressBar.maximum()
87          self.progressBar.setValue(curVal + (maxVal - curVal) // 100)
88      def createTopLeftGroupBox(self):
89          self.topLeftGroupBox = QGroupBox("Dynamic Graph")
90          layout = QVBoxLayout()
91          self.sc = MplCanvas(self, width=5, height=4, dpi=100)
92          layout.addWidget(self.sc)
93          self.topLeftGroupBox.setLayout(layout)
94
95
96
97
98      def createBottomLeftTabWidget(self):
99
100         self.bottomLeftTabWidget = QTabWidget()
```

```
101
102         self.tab1 = QWidget()
103         self.tab1.resize(1200,500);
104
105         self.tableWidget = QTableWidget(10, 10)
106
107
108         self.tab1hbox = QHBoxLayout()
109
110
111         self.tab1hbox.addWidget(self.tableWidget)
112
113         self.tab1.setLayout(self.tab1hbox)
114
115
116         self.bottomLeftTabWidget.addTab(self.tab1, "&Table")
117
118
119
120
121     def dialMoved(self):
122         print(self.dial3.value())
123
124     def createBottomRightGroupBox(self):
125         self.bottomRightGroupBox = QGroupBox("Actuators")
126         self.Button_1 = QPushButton("Actuator 1")
127         self.Button_1.setCheckable(True)
128         self.Button_1.clicked.connect(self.Button_1_Handler)
129         self.Button_2 = QPushButton("Actuator 2")
130         self.Button_2.setCheckable(True)
131         self.Button_2.clicked.connect(self.Button_2_Handler)
132         self.Button_3 = QPushButton("Actuator 3")
133         self.Button_3.setCheckable(True)
134         self.Button_3.clicked.connect(self.Button_3_Handler)
135         layout2 = QVBoxLayout()
136         layout2.addWidget(self.Button_1)
137         layout2.addWidget(self.Button_2)
138         layout2.addWidget(self.Button_3)
139         self.bottomRightGroupBox.setLayout(layout2)
```

```
140
141     def createTopRightGroupBox(self):
142
143
144         self.topRightGroupBox = QGroupBox("Control Panel")
145
146         self.dial3 = QDial(self.topRightGroupBox)
147         self.dial3.setMinimum(-31)
148         self.dial3.setMaximum(31)
149         self.dial3.setNotchesVisible(True)
150         #self.dial3.valueChanged.connect(self.dialMoved)
151         print(self.dial3.value())
152
153         sendPushButton=QPushButton("Send the value")
154         sendPushButton.setDefault(True)
155         sendPushButton.clicked.connect(self.sendSerial)
156
157         readForceButton=QPushButton("Read force values")
158         readForceButton.setDefault(True)
159         readForceButton.clicked.connect(self.readForce)
160
161         flatPushButton = QPushButton("Turn off the actuators")
162
163         layout = QVBoxLayout()
164         layout.addWidget(flatPushButton)
165         layout.addWidget(readForceButton)
166         layout.addWidget(self.dial3)
167         layout.addWidget(sendPushButton)
168
169         self.red_led=QLed(self, onColour=QLed.Red, shape=QLed.Circle)
170         self.red_led.value=False
171
172         self.blue_led=QLed(self, onColour=QLed.Blue, shape=QLed.
                Circle)
173         self.blue_led.value=False
174
175         self.green_led=QLed(self, onColour=QLed.Green, shape=QLed.
                Circle)
176         self.green_led.value=False
```

```
177
178         self.light = 0
179
180         layout.addWidget(self.red_led)
181         layout.addWidget(self.blue_led)
182         layout.addWidget(self.green_led)
183         layout.addStretch(1)
184         self.topRightGroupBox.setLayout(layout)
185
186     def sendSerial(self):
187
188         if (self.light == 1):
189             self.red_led
190         #else if (self.light == 2):
191         #else if (self.light == 3):
192
193         ser=serial.Serial('COM3',9600)
194         buf = str(self.dial3.value()).encode("utf-8")
195         ser.write(b'' + buf)
196         ser.close()
197
198         ser=serial.Serial('COM3',9600, timeout=7)
199         n_cnt = 0
200         n_list = []
201         enc_list = []
202         enc_val1 = []
203         enc_val2 = []
204         enc_val3 = []
205         enc_val4 = []
206         while(n_cnt<=4):
207             enc = ser.readline()
208             enc_str = str(enc)
209             if ("enc" in enc_str):
210                 enc_list = enc_str.split(":", 5)
211                 n_list.append(n_cnt)
212                 enc_val1.append(enc_list[1])
213                 enc_val2.append(enc_list[2])
214                 enc_val3.append(enc_list[3])
215                 enc_val4.append(enc_list[4])
```

```
216                 print(enc)
217             n_cnt = n_cnt + 1
218         ser.close()
219         self.sc.axes.set_ylim(auto=True)
220         self.sc.axes.plot(n_list, enc_val1)
221         self.sc.axes.plot(n_list, enc_val2)
222         self.sc.axes.plot(n_list, enc_val3)
223         self.sc.axes.plot(n_list, enc_val4)
224         for i in range(n_cnt-1):
225             self.tableWidget.setItem(0,i, QTableWidgetItem(enc_val1[i
                ]))
226             self.tableWidget.setItem(1,i, QTableWidgetItem(enc_val2[i
                ]))
227             self.tableWidget.setItem(2,i, QTableWidgetItem(enc_val3[i
                ]))
228             self.tableWidget.setItem(3,i, QTableWidgetItem(enc_val4[i
                ]))
229 
230         self.sc.draw()
231 
232     def readForce(self):
233         ser=serial.Serial('COM3',9600)
234         ser.write(b'255')
235         ser.close()
236 
237         ser=serial.Serial('COM3',9600, timeout=7)
238         n_cnt = 0
239         n_list = []
240         enc_list = []
241         enc_val1 = []
242         enc_val2 = []
243         enc_val3 = []
244         enc_val4 = []
245         while(n_cnt<=4):
246             enc = ser.readline()
247             enc_str = str(enc)
248             if ("fs" in enc_str):
249                 enc_list = enc_str.split(":", 5)
250                 n_list.append(n_cnt)
```

```
251                 enc_val1.append(enc_list[1])
252                 enc_val2.append(enc_list[2])
253                 enc_val3.append(enc_list[3])
254                 enc_val4.append(enc_list[4])
255                 print(enc)
256             n_cnt = n_cnt + 1
257         ser.close()
258         self.sc.axes.set_ylim(auto=True)
259         self.sc.axes.plot(n_list, enc_val1)
260         self.sc.axes.plot(n_list, enc_val2)
261         self.sc.axes.plot(n_list, enc_val3)
262         for i in range(n_cnt-1):
263             self.tableWidget.setItem(0,i, QTableWidgetItem(enc_val1[i
                ]))
264             self.tableWidget.setItem(1,i, QTableWidgetItem(enc_val2[i
                ]))
265             self.tableWidget.setItem(2,i, QTableWidgetItem(enc_val3[i
                ]))
266
267         self.sc.draw()
268
269     def Button_1_Handler(self):
270         if self.Button_1.isChecked():
271             self.sendType_1()
272         else:
273             self.sendStop_1()
274     def Button_2_Handler(self):
275         if self.Button_2.isChecked():
276             self.sendType_2()
277         else:
278             self.sendStop_2()
279     def Button_3_Handler(self):
280         if self.Button_3.isChecked():
281             self.sendType_3()
282         else:
283             self.sendStop_3()
284     def sendType_1(self):
285         self.light = 1
286         print('test')
```

```
287             ser=serial.Serial('COM3',9600)
288             ser.write(b'1')
289             ser.close()
290         def sendType_2(self):
291             self.light = 2
292             ser=serial.Serial('COM3',9600)
293             ser.write(b'2')
294             ser.close()
295         def sendType_3(self):
296             self.light = 3
297             ser=serial.Serial('COM3',9600)
298             ser.write(b'3')
299             ser.close()
300         def sendStop_1(self):
301             self.light = -1
302             ser=serial.Serial('COM3',9600)
303             ser.write(b'-1')
304             ser.close()
305         def sendStop_2(self):
306             self.light = -2
307             ser=serial.Serial('COM3',9600)
308             ser.write(b'-2')
309             ser.close()
310         def sendStop_3(self):
311             self.light = -3
312             ser=serial.Serial('COM3',9600)
313             ser.write(b'-3')
314             ser.close()
315
316
317         def createProgressBar(self):
318             self.progressBar = QProgressBar()
319             self.progressBar.setRange(0, 10000)
320             self.progressBar.setValue(0)
321             timer = QTimer(self)
322             timer.timeout.connect(self.advanceProgressBar)
323             timer.start(1000)
324
325 if __name__ == '__main__':
```

```
326     import sys
327     app = QApplication(sys.argv)
328     gallery = WidgetGallery()
329     gallery.show()
330     sys.exit(app.exec_())
```

Listing B.2: Arduino spherical joint control

```
1  void loop() {
2
3    while (Serial.available() == 0) {}
4    int MaxSpeed = 120;
5
6    char input[INPUT_SIZE + 1];
7    byte size = Serial.readBytes(input, INPUT_SIZE);
8    input[size] = 0;
9    char* cmd = strtok(input, "|");
10   int n=0;
11   while (cmd != 0){
12     char* separator = strchr(cmd, ':');
13     if (separator != 0)
14     {
15         // Actually split the string in 2: replace ':' with 0
16         *separator = 0;
17         opt[n] = atoi(cmd);
18         ++separator;
19         if (val<=5) {
20           val[n] = atoi(separator);
21         } else val[n] = separator;
22     }
23     // Find the next command in input string
24     cmd = strtok(0, "|");
25     n++;
26   }
27  // try{
28 //     char last_val[6] = val[6]
29 //   }
30 //   catch(Exception e){
31 //   }
```

```
32
33    for (int i=1; i<=4; i++){
34      if (opt[i]>=-4 && opt[i]<=4){
35        if (val[i]==0){
36          if (opt[i]>=-4 && opt[i]<=-1){
37            stopMotor(pinMotors[abs(opt[i])]);
38          }
39        } else if (val[i]>0 && abs(val[i]) <= MaxSpeed){
40          motor_rotate(pinMotors[abs(opt[i])], val[i]);
41        } else if (val[i]<0 && abs(val[i]) <= MaxSpeed){
42          reverse_motor(pinMotors[abs(opt[i])], val[i]);
43        }
44      }
45    }
46    if (val[5] == 255) {
47      force_read();
48    }
49    else if (val[5] == 254) {
50      encoder_read();
51    }
52    else if (val[5] == 253) {
53      enc_force_read();
54    }
55    delay(500);
56  }
57
58  void motor_rotate(const int pinMotor[4], int encVal) {
59    digitalWrite(pinSTBY_1, HIGH);
60    digitalWrite(pinSTBY_2, HIGH);
61    digitalWrite(pinMotor[1], HIGH);
62    digitalWrite(pinMotor[2], LOW);
63    analogWrite(pinMotor[0], encVal);
64  }
65
66  void  reverse_motor(const int pinMotor[4], int encVal) {
67    digitalWrite(pinSTBY_1, HIGH);
68    digitalWrite(pinSTBY_2, HIGH);
69    digitalWrite(pinMotor[1], LOW);
70    digitalWrite(pinMotor[2], HIGH);
```

```
71    analogWrite(pinMotor[0], encVal);
72  }
73
74  void stopMotor(const int pinMotor[4]) {
75    digitalWrite(pinMotor[1], LOW);
76    digitalWrite(pinMotor[2], LOW);
77  }
78
79
80  void encoder_read() {
81    int i = 0;
82    while (i <= 10) {
83      delay(100);
84
85      Serial.print("enc:|");
86      for(int j=1;j<=4;j++){
87        pwm_value[j] = pulseIn(PWM_PINS[j], HIGH, 10000);
88        pwm_value[j] = map(pwm_value[j], 0, 918, 0, 360);
89        Serial.print(pwm_value[j]);
90        Serial.print("|");
91      }
92      Serial.println();
93      i = i + 1;
94    }
95  }
96
97  void force_init() {
98
99    long zero_factor[4];
100
101   for (int j=1;j<=3;j++){
102     scale[j].begin(DOUT[j], CLK[j]);
103     scale[j].set_scale();
104     scale[j].tare(); //Reset the scale to
105     //Get a baseline reading
106     zero_factor[j] = scale[j].read_average();
107   }
108 }
109
```

```
110 void force_read() {
111
112   int i = 0;
113   //while(true){
114   while (i <= 10) {
115     //force sensor code
116     Serial.print("fs:|");
117     for (int j=1;j<=3;j++){
118       scale[j].set_scale(calibration_factor[j]);
119       Serial.print(scale[j].get_units(), 1);
120       Serial.print("|");
121     }
122     Serial.println();
123     delay(100);
124     i = i + 1;
125   }
126 }
127
128 void enc_force_read() {
129
130   int i = 0;
131   int val = 0;
132   //while(true){
133
134   int rot[] = {0, 0, 0, 0, 0};
135   int rot_lim = 514;
136   bool exceeded = false;
137
138
139   int diff[5] = {0, 0, 0, 0 , 0};
140   int lastValue[5] = {0, 0, 0, 0 , 0};
141   int prevValue[5] = {0, 0, 0, 0 , 0};
142   int servoId[5];
143
144
145   char input[INPUT_SIZE + 1];
146   Serial.setTimeout(7000);
147   byte size = Serial.readBytes(input, INPUT_SIZE);
148   input[size] = 0;
```

```
149   char* cmd = strtok(input, "|");
150   int n=0;
151   while (cmd != 0){
152     char* separator = strchr(cmd, ':');
153     if (separator != 0)
154     {
155         // Actually split the string in 2: replace ':' with 0
156         *separator = 0;
157         servoId[i] = atoi(cmd);
158         ++separator;
159         lastValue[i] = atoi(separator);
160 
161         // Do something with servoId and position
162     }
163     // Find the next command in input string
164     cmd = strtok(0, "|");
165     n++;
166   }
167   Serial.setTimeout(1000);
168 
169 
170   while (i <= 700) {
171     delay(100);
172 
173     //force sensor code
174     Serial.print("fs:|");
175     for (int j=1;j<=3;j++){
176       scale[j].set_scale(calibration_factor[j]);
177       Serial.print(scale[j].get_units(), 1);
178       Serial.print("|");
179     }
180     Serial.println();
181 
182     //encoder code
183     Serial.print("enc:|");
184     for(int j=1;j<=4;j++){
185       pwm_value[j] = pulseIn(PWM_PINS[j], HIGH, 10000);
186       pwm_value[j] = map(pwm_value[j], 0, 918, 0, 360);
187       Serial.print(pwm_value[j]);
```

```
188       Serial.print("|");
189     }
190     for(int j=1;j<=4;j++){
191       //check if rotation exceeded limit
192       if(i == 0){
193         diff[j] = 0;
194       }else{
195         diff[j] = pwm_value[j] - prevValue[j];
196       }
197 
198       if (abs(diff[j]) >= 300) {
199         if (diff[j]>=0){
200           rot[j]++;
201         }else{
202           rot[j]--;
203         }
204       } else{
205         if (abs(diff[j])<20) val = val + abs(diff[j]); //
206         if (val >= rot_lim) {
207           stopMotor(pinMotors[j]);
208           exceeded = true;
209         }
210       }
211       Serial.print(rot[j]);
212       Serial.print("|");
213       prevValue[j] = pwm_value[j];
214     }
215 
216     if(exceeded){
217       Serial.print("off");
218       Serial.print("|");
219       Serial.print(val);
220       Serial.println();
221       break;
222     }else{
223       Serial.print("on");
224       Serial.print("|");
225       Serial.print(val);
226       Serial.println();
```

```
227       }
228       i = i + 1;
229     }
230 }
231
232
233 void stop_all() {
234   stopMotor(pinMotor1);
235   stopMotor(pinMotor2);
236   stopMotor(pinMotor3);
237   stopMotor(pinMotor4);
238 }
```

Bibliography

[1] Ball, S. J., Brown, I. E., & Scott, S. H. Medarm: a rehabilitation robot with 5dof at the shoulder complex. *IEEE/ASME international conference on Advanced intelligent mechatronics*, 2007.

[2] Bhattacharyya, S., Konar, A., & Tibarewala, D. N. . Motor imagery, p300 and error-related eeg-based robot arm movement control for rehabilitation purpose. *Medical & biological engineering & computing*, 52(12):1007–1017, 2014.

[3] Buongiorno, D., Sotgiu, E., Leonardis, D., Marcheschi, S., Solazzi, M., & Frisoli, A. Wres: a novel 3 dof wrist exoskeleton with tendon-driven differential transmission for neuro-rehabilitation and teleoperation. *IEEE Robotics and Automation Letter.*, 3(3):2152–2159, 2001.

[4] Graimann, B., Allison, B., & Pfurtscheller, G. . Brain-computer interfaces: A gentle introduction. *In Brain-computer interfaces*, pages 1–27, 2009.

[5] Hussein, M. E. . 3d printed myoelectric prosthetic arm (doctoral dissertation, thesis bachelor degree engineering (mechatronics). pages 1–87, 2014.

[6] Jiang, H., Zhang, T., Xiao, C., Li, J., & Guan, Y. Modular design of 7-dof cable-driven humanoid arms. *In International Conference on Intelligent Robotics and Applications*, pages 680–691, 2019, August.

[7] Kim, Y. J., Kim, J. I., & Jang, W. Quaternion joint: Dexterous 3-dof joint representing quaternion motion for high-speed safe interaction. *In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 935–942, 2018, October.

[8] Otarbay, Z., Assylgali, I., Yskak, A., & Folgheraiter, M. . Development of a teach pendant for humanoid robotics with cartesian and joint-space control modalities. *In 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 1(2):1–6, (2019, October).

[9] Saduanov, B., Tokmurzina, D., Kunanbayev, K., & Abibullaev, B. . Design and optimization of a real-time asynchronous bci control strategy for robotic manipulator assistance. *In 2020 8th International Winter Conference on Brain-Computer Interface (BCI)*, pages 1–5, (2020, February).

[10] Al-Quraishi, M. S., Elamvazuthi, I., Daud, S. A., Parasuraman, S., & Borboni, A. Eeg-based control for upper and lower limb exoskeletons and prostheses: A systematic review. *Sensors*, 18(10):3342, 2018.

[11] Bhagat, N. A., Venkatakrishnan, A., Abibullaev, B., Artz, E. J., Yozbatiran, N., Blank, A. A., ... & Francisco, G. E. Design and optimization of an eeg-based brain machine interface (bmi) to an upper-limb exoskeleton for stroke survivors. *Frontiers in neuroscience*, 10(122), 2016.

[12] Folgheraiter, M., Jordan, M., Straube, S., Seeland, A., Kim, S. K., & Kirchner, E. A. Measuring the improvement of the interaction comfort of a wearable exoskeleton. *International Journal of Social Robotics*, 4(3):285–302, 2012.

[13] Folgheraiter, M., Yessirkepov, S., & Yessaly, A. Modeling and simulation of spherical parallel manipulators in coppeliasim (v-rep) robot simulator software. *IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 571–576, 2019, November.

[14] Fu, M. J., Daly, J. J., & Cavusoglu, M. C. Assessment of eeg event-related desynchronization in stroke survivors performing shoulder-elbow movements. *In Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 3158–3164, 2006, May.

[15] Lew, E. Y., Chavarriaga, R., Silvoni, S., & Millán, J. D. R. . Single trial prediction of self-paced reaching directions from eeg signals. *Frontiers in neuroscience*, 8(222), 2014.

[16] Luu, T. P., Nakagome, S., He, Y., & Contreras-Vidal, J. L. Real-time eeg-based brain-computer interface to a virtual avatar enhances cortical involvement in human treadmill walking. *Scientific reports*, 7(1):1–12, 2017.

[17] Stienen, A. H., Hekman, E. E., Van Der Helm, F. C., & Van Der Kooij, H. . Self-aligning exoskeleton axes through decoupling of joint rotations and translations. *IEEE Transactions on Robotics*, 25(3):628–633, 2009.

[18] Tursynbek, I., & Shintemirov, A. . Modeling and simulation of spherical parallel manipulators in coppeliasim (v-rep) robot simulator software. *International Conference Nonlinearity, Information and Robotics (NIR)*, pages 1–6, 2020, December.