**Lymphedema Rehabilitation**

Report 3

**Progress made till date:**

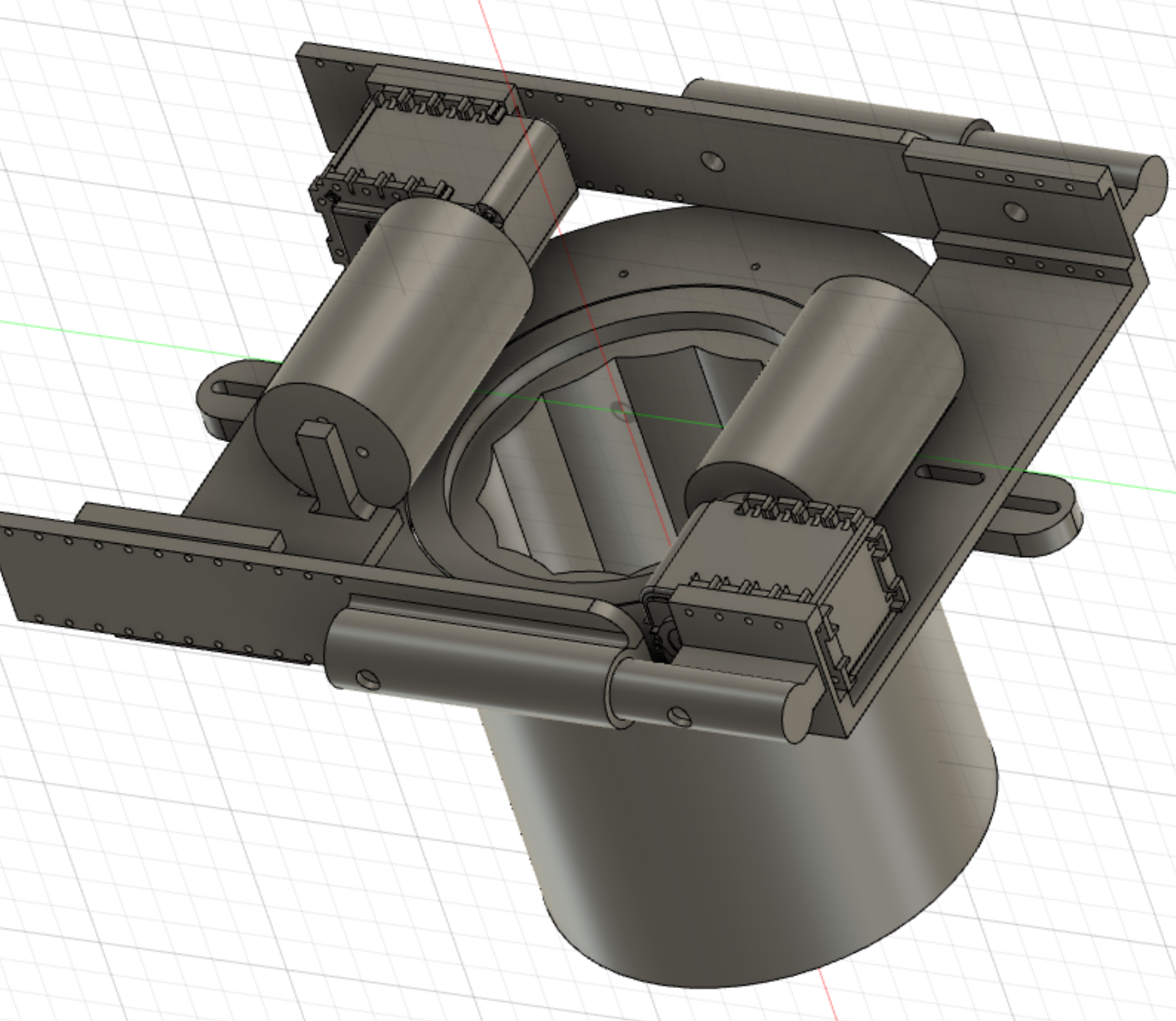
**A) Mechanical**

1. **Static Analysis of the design:**

The static analysis of the robot is done in Fusion 360 Simulation platform. The simulation is based on static stress analysis. Forces for the analysis are considered based on normal reaction due to the two acting springs. Simulation provided the suggestions on design changes to be made.

1. **Design changes according to Static analysis:**

Changes are made to the design based on results from the static analysis.



1. **First Dynamic Analysis in Inventor:**

Dynamic analysis of the mechanism is done on the preliminary level, where the robot only moves but fails to provide the climbing simulation. The progress is being made on the climbing simulation in Inventor.

**B) Control:**

1. **Implementation of Dynamixel Motor Control**

Raspberry Pi is setup and is programmed to control two Dynamixel Servo motors. Control includes position and velocity of the motors.

import os

import time

from dynamixel\_sdk import \*

ADDR\_MX\_TORQUE\_ENABLE = 24 # Control table address is different in Dynamixel model

ADDR\_MX\_GOAL\_POSITION = 30

ADDR\_MX\_PRESENT\_POSITION = 36

LEN\_MX\_PRESENT\_POSITION = 2

#minimum\_value\_ax12 =

#maximum\_value\_ax12 =

# Protocol version

PROTOCOL\_VERSION = 1.0 # See which protocol version is used in the Dynamixel

# Default setting

# Dynamixel ID : 1

BAUDRATE = 1000000 # Dynamixel default baudrate : 57600

DEVICENAME = '/dev/ttyUSB0' # Check which port is being used on your controller # ex) Windows: "COM1" Linux: "/dev/ttyUSB0" Mac: "/dev/tty.usbserial-\*"

TORQUE\_ENABLE = 1 # Value for enabling the torque

TORQUE\_DISABLE = 0 # Value for disabling the torque

DXL\_MOVING\_STATUS\_THRESHOLD = 20 # Dynamixel moving status threshold

port\_num = PortHandler(DEVICENAME)

packetHandler = PacketHandler(PROTOCOL\_VERSION)

COM\_SUCCESS = 0

COM\_TX\_FAIL = -1001

sampling\_rate = 0.22 #seconds

Setup()

def Setup():

# "Opening the port and setting the baudrate"

# if port\_num.openPort():

# print("Succeeded to open the port")

# else:

# print("Failed to open the port")

# print("Press any key to teAinate...")

# quit()

# # Set port baudrate

if port\_num.setBaudRate(BAUDRATE):

print("Succeeded to change the baudrate")

else:

print("Failed to change the baudrate")

print("Press any key to teAinate...")

quit()

"just to check the connection"

DXL\_ID = [1,2 ]

"enabling the torque and checking if dynamixel is connected"

dxl\_coM\_result, dxl\_error = packetHandler.write1ByteTxRx(port\_num, DXL\_ID[1], ADDR\_MX\_TORQUE\_ENABLE, 0) # TORQUE\_DISABLE

if dxl\_coM\_result != COM\_SUCCESS:

print("%s" % packetHandler.getTxRxResult(dxl\_coM\_result))

elif dxl\_error != 0:

print("%s" % packetHandler.getRxPacketError(dxl\_error))

else:

print("Dynamixel has been successfully connected")

def move\_motor(goal\_Pos, speed=20):

DXL\_ID = int(1)

dxl\_comm\_result, dxl\_error = packetHandler.write2ByteTxRx(port\_num, DXL\_ID, ADDR\_MX\_GOAL\_POSITION, goal\_Pos)

if dxl\_comm\_result != COMM\_SUCCESS:

print("THIS IS THE PROBLEM 1")

print("%s" % packetHandler.getTxRxResult(dxl\_comm\_result))

elif dxl\_error != 0:

print("%s" % packetHandler.getRxPacketError(dxl\_error))

print("THIS IS THE PROBLEM 2")

else:

print("THIS IS OK!")

time.sleep(0.2)

"reading the present position and stopping if threshold is lower than 20"

dxl\_present\_position, dxl\_comm\_result, dxl\_error = packetHandler.read2ByteTxRx(port\_num, DXL\_ID, ADDR\_MX\_PRESENT\_POSITION)

if dxl\_comm\_result != COMM\_SUCCESS:

print("%s" % packetHandler.getTxRxResult(dxl\_comm\_result))

elif dxl\_error != 0:

print("%s" % packetHandler.getRxPacketError(dxl\_error))

print("[ID:%03d] GoalPos:%03d PresPos:%03d" % (DXL\_ID, goal\_Pos, dxl\_present\_position))

if not abs(goal\_Pos - dxl\_present\_position) > DXL\_MOVING\_STATUS\_THRESHOLD:

return

else:

print("[ID:%03d] GoalPos:%03d PresPos:%03d" % (DXL\_ID, goal\_Pos, dxl\_present\_position))

global dxl\_load

dxl\_load, dxl\_comm\_result, dxl\_error = packetHandler.read2ByteTxRx(port\_num, DXL\_ID, 40)

if dxl\_comm\_result != COMM\_SUCCESS:

print("%s" % packetHandler.getTxRxResult(dxl\_comm\_result))

elif dxl\_error != 0:

print("%s" % packetHandler.getRxPacketError(dxl\_error))

else:

print("[ID:%03d] GoalPos:%03d Load:%03d" % (DXL\_ID, goal\_Pos, dxl\_load))

return dxl\_load

def read\_ax(ID,ID1,ID2):

count = 0

open(filename, 'w').close()

dxl\_comm\_result, dxl\_error = packetHandler.write1ByteTxRx(port\_num, ID, ADDR\_MX\_TORQUE\_ENABLE, 0)

dxl\_comm\_result1, dxl\_error1 = packetHandler.write1ByteTxRx(port\_num, ID1, ADDR\_MX\_TORQUE\_ENABLE, 0)

dxl\_comm\_result2, dxl\_error2 = packetHandler.write1ByteTxRx(port\_num, ID2, ADDR\_MX\_TORQUE\_ENABLE, 0)

if dxl\_comm\_result != COMM\_SUCCESS:

print("%s" % packetHandler.getTxRxResult(dxl\_comm\_result))

elif dxl\_error != 0:

print("%s" % packetHandler.getRxPacketError(dxl\_error))

else:

print("Dynamixel#%d has been successfully connected" % ID)

while 1:

time1 = time.time()

dxl\_present\_position, dxl\_comm\_result, dxl\_error = packetHandler.read2ByteTxRx(port\_num, ID, ADDR\_MX\_PRESENT\_POSITION)

dxl\_present\_position1, dxl\_comm\_result1, dxl\_error1 = packetHandler.read2ByteTxRx(port\_num, ID1, ADDR\_MX\_PRESENT\_POSITION)

dxl\_present\_position2, dxl\_comm\_result2, dxl\_error2 = packetHandler.read2ByteTxRx(port\_num, ID2, ADDR\_MX\_PRESENT\_POSITION)

if dxl\_comm\_result != COMM\_SUCCESS:

print("%s" % packetHandler.getTxRxResult(dxl\_comm\_result))

elif dxl\_error != 0:

print("%s" % packetHandler.getRxPacketError(dxl\_error))

else:

#print("[ID:%03d] PresPos:%03d" % (ID, dxl\_present\_position))

#return dxl\_present\_position

#print("[ID:%03d] PresPos:%03d" % (ID1, dxl\_present\_position1))

if writefile(ID,dxl\_present\_position,ID1,dxl\_present\_position1,ID2,dxl\_present\_position2) == 1:

#print("this has been broken")

break

delaytime = time.time() - time1

time.sleep(sampling\_rate - delaytime)

print("time delay while storing",delaytime)

#print("time is",time.time()-time1)

def readspeed(ID,ID1, ID2):

dxl\_speed, dxl\_comm\_result, dxl\_error = packetHandler.read2ByteTxRx(port\_num, ID, 38)

dxl\_speed1, dxl\_comm\_result1, dxl\_error1 = packetHandler.read2ByteTxRx(port\_num, ID1, 38)

dxl\_speed2, dxl\_comm\_result2, dxl\_error2 = packetHandler.read2ByteTxRx(port\_num, ID2, 38)

if dxl\_comm\_result != COMM\_SUCCESS:

print("%s" % packetHandler.getTxRxResult(dxl\_comm\_result))

return 6000,6000, 6000

elif dxl\_error != 0:

print("%s" % packetHandler.getRxPacketError(dxl\_error))

return 6000,6000, 6000

else:

#print("[ID:%03d] Speed:%03d" % (ID, dxl\_speed))

#print("[ID:%03d] Speed:%03d" % (ID1, dxl\_speed1))

return dxl\_speed,dxl\_speed1, dxl\_speed2

def ChangeSpeed\_AX(DXL\_ID,speed):

dxl\_comm\_result, dxl\_error = packetHandler.write2ByteTxRx(port\_num,DXL\_ID, 32, speed)

print("Speed is set")

dxl\_present\_speed, dxl\_comm\_result, dxl\_error = packetHandler.read2ByteTxRx(port\_num, DXL\_ID, 32)

time.sleep(0.2)

print(dxl\_present\_speed)

def ChangeSpeed\_MX(DXL\_ID,speed):

dxl\_comm\_result, dxl\_error = packetHandler.write2ByteTxRx(port\_num,DXL\_ID, 32, speed)

print("Speed is set")

dxl\_present\_speed, dxl\_comm\_result, dxl\_error = packetHandler.read2ByteTxRx(port\_num, DXL\_ID, 32)

time.sleep(0.2)

print(dxl\_present\_speed)

1. **Test Motors:**

Motors have been tested using the program. The successfully controls the position and the speed of the motors.

**Ongoing Progress:**

A) Mechanical:

1. Dynamic Analysis in Inventor

B) Control:

1. Integrating Flex Sensor

**Next Tasks:**

A) Mechanical:

1. Print Parts
2. Assemble Climbing Mechanism
3. Integrate pressurizing cuff mechanism

B) Control:

1. Integrate pressure sensor and pump
2. Test overall program
3. Test climbing mechanism
4. Test pressurizing mechanism
5. Test both mechanisms together