Lymphedema Rehabilitation

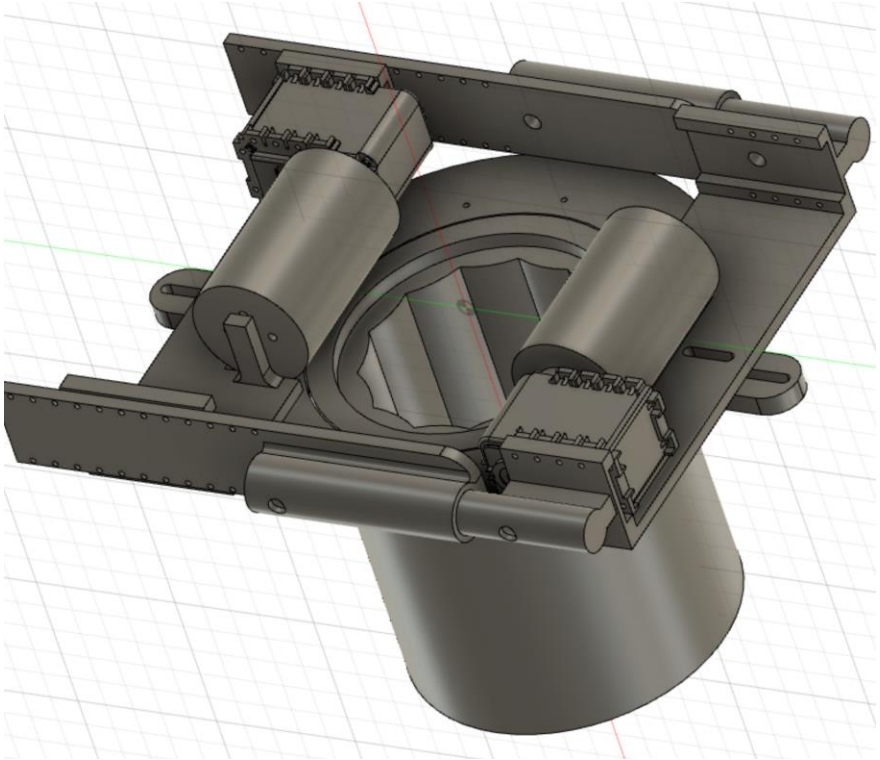
Report 3

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.

2) **Design changes according to Static analysis:**
   Changes are made to the design based on results from the static analysis.



3) **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_presen
t_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)
```

2) **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