ELBOW EXOSKELETON PROJECT

BY: YUSUF WONG

Project Overview

The purpose of this project is to design, and 3D print a wearable elbow exoskeleton robot using 3D printed hardware, a servo motor, and a loading cell sensor. There are 3 main components to this project: position control (point-to-point & trajectory), admittance control, and virtual wall control. This paper will demonstrate attempts to allow the elbow exoskeleton to work properly.

Background Introduction

Robots are part of our day-to-day life and are used to make everyone's life a bit easier. We can program robots to assist us in our day-to-day work, especially when working in a factory. This elbow exoskeleton project aims to discover some underlying principles of control and assisting someone trying to lift a heavy object. This type of design will allow people, such as factory and warehouse workers to have less train on their arms by providing torque assistance whenever they are carrying extremely large loads that put strain on their backs and limbs. The ultimate goal of this project is to lessen this strain and increase productivity.

Biomechanics

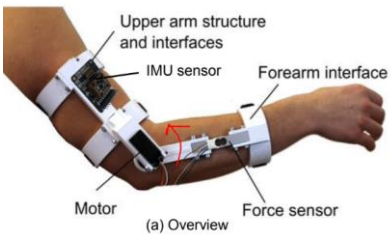

Figure 1: Biomechanics Diagram

Elbow exoskeletons work by sensing a loading or stress on a user using a loadcell which is a force sensor. When a factory worker, for example, is carrying something heavy, the loadcell senses stress and activates the servo motor to apply a counter rotation and reduce strain on the user by applying assistive torque about the elbow socket. The motor produces enough torque so that the user doesn't get hurt. Figure 1 shows an example of how most elbow exoskeletons behave and look like.

Hardware

The components of the elbow exoskeleton design are as follows: Upper Arm Cuffs (x2), Upper Cuff Attachment, Upper Arm Servo Attachment, High-Torque Servo Motor, Lower Arm Servo Attachment, a load cell, a Lower Arm Cuff, screws, wires, and an Arduino. These can be illustrated in Figures 1 and 2.

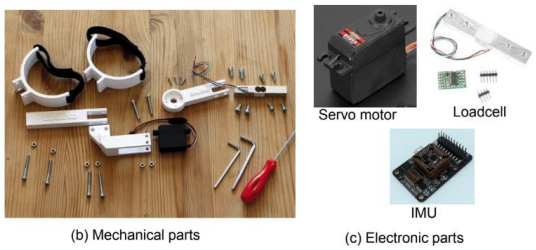

Figure 2: Hardware Components

Design

The following image is a design of my 3D assembly model of the exoskeleton:

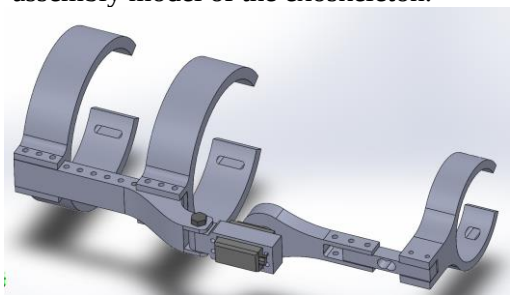

Figure 3: 3D Model Design Assembly

This design has two degrees of freedom (DOF), one of which is applied by the servo motor. The other DOF is about the bolt near the Upper Arm Servo Attachment. This will allow a user to have free later and medial rotation during operation, while still having flexion and extension assistance from the motor that supplies assistive torque when the user experiences strain on his or her forearms. Thew following image demonstrates the 2 DOFs:

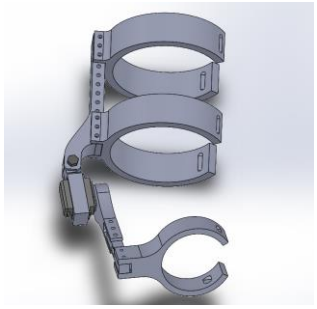

Figure 4: Top View of Design, Showing 2 DOFs

Control System

The following is the code for the Point-to-Point Position Control:

Yusuf_Point_to_Point

```
//Created By: Yusuf Wong
#include <Servo.h>
Servo myservo;

int positionDesired;
int counter = 0;
int reps = 6;
int delayTime = 1000;
int servoPin = 7;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  myservo.attach(servoPin);
}

void loop() {
  // put your main code here, to run repeatedly:
  positionDesired = 0;
  myservo.write(positionDesired);
  delay(delayTime);

  positionDesired = 90;
  myservo.write(positionDesired);
  delay(delayTime);
  counter++;
  if(counter == reps){
    exit(0);
  }
}
```

Figure 5: Point-to-Point Position Control Code

This is the code for Replaying Trajectory Position Control:

Yusuf_Replay_Trajectory

```
//Created By: Yusuf Wong
#include <Servo.h>
#include <EEPROM.h>
#define SAMPLE_DELAY 25
#define SAMPLES 200
Servo myservo;
int servoPin = 7;
int servoAnalogInPin = A0;
int delayTime = 3000;
int posIs;
int posIsServo;
void setup() {
  Serial.begin(115200);
}
void loop() {
    reset();
    Serial.println("Trajectory recording starts in 4 seconds");
    delay(4000);
    recordTrajectory();
    myservo.write(EEPROM.read(0));
    delay(1000);
    replayTrajectory();
    delay(1000);
}
```

Yusuf_Replay_Trajectory

```
void recordTrajectory(){
  Serial.println("Recording");
  for(int addr=0; addr<SAMPLES; addr++){
    posIs = analogRead(servoAnalogInPin);
    posIsServo = map(posIs,616,350, 0,90); //616 352
    if(posIsServo>90){
      posIsServo = 90;
    }
    else {if(posIsServo<0){
      posIsServo = 0;    }
    }
    EEPROM.write(addr,posIsServo);
    delay(SAMPLE_DELAY);
  }
  Serial.println("Done recording");
}
void replayTrajectory(){
  myservo.attach(servoPin);
  Serial.println("Playing");
    for(int addr=0; addr<SAMPLES; addr++){
    byte positionDesired = EEPROM.read(addr);
    myservo.write(positionDesired);
    //Serial.print("Address: "); Serial.print(addr);
    //Serial.print("    Position: "); Serial.print(positionDesired); Serial.print("\n");
    delay(SAMPLE_DELAY);
  }
  Serial.println("Done replaying");
  myservo.detach();
}
void reset(){
    Serial.println("Resetting");
    myservo.write(0);
    delay(2000);
}
```

Figure 6: Replay Trajectory Position Control (NOTE: an enhanced version can be shown in the PowerPoint)

Here are the results of calibrating the loading cell:

Weight (grams)	Amplified Voltage (V)
0	13
104	168
180	289
260	417
341	542
412	657
500	798
597	949
665	1038
709	1121
765	1206
816	1295
850	1355
903	1432
950	1499
1003	1583

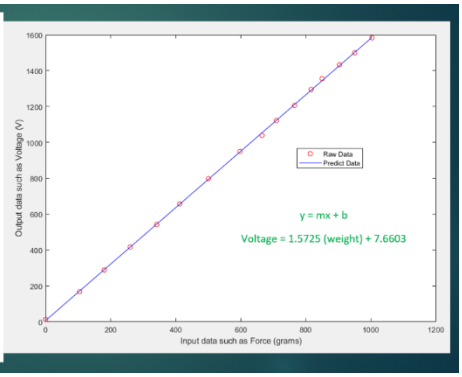

Figure 7: Load Cell Calibration

The following is the code for Admittance Control:

Yusuf_Admittance_Control

```
#include <Servo.h>
Servo myservo;

#include "HX711.h"
#define calibration_factor -7050.0 //This value is obtained usi
#define DOUT  3
#define CLK  2
HX711 scale(DOUT, CLK);
int data=0;
uint8_t dataout[8];
int i=0;
double forceAnalog;
double m = 1.5725;
double b = 7.6603;
int servoAnalogInPin = A0;
void setup() {
  Serial.begin(115200);
  myservo.attach(7);
  myservo.write(60);
  scale.set_scale(calibration_factor); //This value is obtained
  scale.tare();  //Assuming there is no weight on the scale at
  delay(3000);
}
```

```
const int forceOffset = 20; //207
int weight;
int forceDesired = 0;

int positionDesired = 70; //40;
float gain = 0.2;
int minAngle = 30;
int maxAngle = 80;
int servoAnalog;
int currentAngle;
void loop() {
  // y = grams = 1.5725*x + 7.6603 //x = amplified voltage reading from
  data=(int)100*scale.get_units();
  forceAnalog = (data -b)/m;  // # of grams reported from loadcell
  weight = forceAnalog - forceOffset; //actual weight/grams experienced

  Serial.print(weight); Serial.print("g\t\t");

  servoAnalog = analogRead(servoAnalogInPin); //reading analog values fr
  currentAngle = map(servoAnalog,630,350, 0,90); //get angular servo pos
  Serial.print(servoAnalog); Serial.print("\t"); Serial.print(currentAng

  delay(20);

  positionDesired -= gain*(weight-forceDesired);
  //VirtualWall_SafetyCheck(positionDesired);
  //VirtualWall_SafetyCheck(currentAngle);
  myservo.write(positionDesired);
  delay(1000);
}
```

Figure 8: Admittance Control Code

The following screenshot shows the implementation of using a Virtual Wall function with the Admittance Control code:

```
void loop() {
  // y = grams = 1.5725*x + 7.6603 //x = amplified voltage reading from arduino
  data=(int)100*scale.get_units();
  forceAnalog = (data -b)/m;  // # of grams reported from loadcell
  weight = forceAnalog - forceOffset; //actual weight/grams experienced

  Serial.print(weight); Serial.print("g\t\t");

  servoAnalog = analogRead(servoAnalogInPin); //reading analog values from servo
  currentAngle = map(servoAnalog,630,350, 0,90); //get angular servo position reading
  Serial.print(servoAnalog); Serial.print("\t"); Serial.print(currentAngle); Serial.p

  delay(20);

  positionDesired -= gain*(weight-forceDesired);
  VirtualWall_SafetyCheck(positionDesired);
  myservo.write(positionDesired);
  delay(1000);
}

void VirtualWall_SafetyCheck(int angle){
  if(angle > maxAngle){
    angle = maxAngle;
  }
  else if(angle < minAngle){
    angle = minAngle;
  }
```

Figure 9: Virtual Wall Code

NOTE: The Virtual Wall control is just a few extra lines of code added to the admittance control, so that the arm doesn't exceed the customized min and max angles, as shown in the Virtual Wall function above.

Testing

The following list of links are the videos demonstrating each task (NOTE: These can also be found in my PowerPoint!):

Point-to-Point Position Control:
https://www.youtube.com/watch?v=V5tBGYdt8eQ

Replaying Trajectory Position Control:
https://www.youtube.com/watch?v=So4-_zbF5u0

Admittance Control:
https://www.youtube.com/watch?v=qkjsa1iEiG0

Virtual Wall:
https://www.youtube.com/watch?v=1KnaGHWw0e0

Both Position Control Tasks were pretty straight forward, easy to code, and worked very well. The admittance and virtual wall control were a bit difficult because the loading cell reports constantly fluctuating values every time the wires were touched (since we're amplifying very, very small values of voltage). However, by tuning the ForceOffset so that these values don't go exceedingly high, I was able to get the arm to respond to a loading, as well as using the gain control function to return to a desired position. I then used the virtual wall function to make sure the arm doesn't exceed the bounds of the restricted rotation.

Some Problems & Troubleshooting:

While testing my Code with the Load Cell, and constantly trying to apply a load on the arm, the hinge of the lower arm servo attachment broke off at the Motor Connection Interface. I used tape and screw the screws at the bottom of the load cell instead:

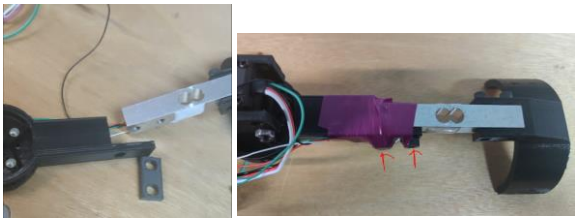

Figure 10: Broken Hinge & Compromise

Final Thoughts

In my opinion, the loadcell was very difficult to use for this project. Whenever the wires were slightly touched or moved, it would give very odd values for the amplified voltage and corresponding measurement for the way. For future designs, I would look into finding either a better loadcell, thicker wires, or another way to determine the stress or loading. Also, the Lower arm Servo Attachment has stress concentration factors which caused a piece to break apart from it. 3D printing a larger density would prevent this from happening, as well as creating fillets in the 3D printed design.

ACKNOWLEDGEMENT

There were a lot of difficulties with this project because it required a lot of testing with Arduino and calibrating the loadcell. I want to thank Professor Hao Su and doctoral teaching assistants Selena and Bowen. They've been such a great help, especially during this horrible pandemic. Thanks guys. 🙂