Anatomically Accurate Motorized Shoulder Model



A Major Qualifying Project submitted to the Faculty of Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Engineering.

Submitted By: Gray Rahm Sean Merone

Authorship: Gray Rahm Sean Merone Cameron Leffler

Date: March 22nd, 2024

Project Advisor: Professor Fiona Levey

This report represents the work of one or more WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its site without editorial or peer review.

Abstract

Despite its significance within the body, educational tools effectively modeling the intricacies of the shoulder joint are scarce. Seeking to address this void, our team continued the development of a life-sized model comprising the scapula, humerus, and a removable clavicle. Our model aims to more accurately replicate the nuanced scapulohumeral rhythm observed during abduction of the humerus, a process integral to understanding the shoulder joint. The scapulohumeral rhythm, orchestrating the coordinated movement between the scapula and humerus during various shoulder motions, particularly abduction, is paramount for maintaining shoulder joint stability and functionality. Our developed rig serves as an excellent foundation for future teams to supplement. A clear path forward is laid for any future teams to fully develop a valuable educational resource for students, healthcare professionals, and medical enthusiasts seeking a comprehensive understanding of the human shoulder.

Acknowledgements

Our team would like to thank several individuals for their efforts in aiding us in completing our project. We would like to thank Pradeep Radhakrishnan for providing us with a space where our team could thrive and construct the model. We would additionally like to thank Peter Hefti in providing us the tools we needed to build the final model. Finally, we'd like to thank our advisor, Fiona Levey, for her incredible guidance and support throughout this project.

Authorship

Our MQP project in creating an anatomically accurate motorized shoulder model was completed through the efforts of Gray Rahm, Cameron Leffler, and Sean Merone. Each member contributed towards the research behind the construction of the model and the materials necessary for the project. Construction of the model's cage, design, schematics, and circuitry was a team effort led by Gray Rahm. The code for the project was developed by Cameron Leffler. The final report submitted to the standards of the WPI MQP report guidelines was a team effort led by Sean Merone.

Cameron Leffler will continue to work on the project for the remainder of the academic year and will submit his MQP report at the conclusion of the academic year. He will be working on creating a 3D animation of the scapulohumeral rhythm using Blender.

Table of Contents

Abstract	0
Acknowledgements	1
Authorship	3
Table of Contents	4
Table of Figures	6
Table of Tables	8
1.0 Introduction	9
2.0 Background	10
2.1 Scapulohumeral Relationship	10
2.2 Previous Work on the Model	14
2.2.1 2021-2022 MQP: Anatomically Accurate Motorized Shoulder Model with Scape Movement	ula 15
2.2.1.1 3D Printed Bones	16
2.2.1.2 Mounting the Bones and Motors	19
2.2.1.3 2021-2022 Team's Final Product and Data Collection	21
2.2.2 2022-2023 MQP: Realistic Shoulder Model with Soft Tissue Attachments	22
3.0 Methodology	23
3.1 Design Goals and Constraints	23
3.2 Rig Design Concepts	24
3.2.1 Channel Design Concept	24
3.2.2 Cage Design Concept	25
3.3 Scapular Rotation	30
3.4 Humeral Abduction	33
3.5 Material Selection	40
3.5.1 Motor Selection	40
3.5.2 Connecting Wires	43
3.5.3 Motor Driver	46
3.6 Cage Construction and Implementing a Matlab Code	48
3.6.1 The Cage	48
3.6.2 Coding and the MATLAB Model	52
3.6.3 Circuit Design	57
3.7 Testing	61
3.7.1 Testing Relative Position	62
3.7.2 Motor Calibration	62
3.7.3 Testing Individual Motors	62
3.7.4 Troubleshooting	63
3.8 Results	63
3.8.1 Partially Completed Rig	63
3.8.2 Interactive Scapula Position Graph	64

4.0 Discussion	65
4.1 Motor Location and Function	
4.2 Materials Selection	
5.0 Broader Impacts	
5.1 Engineering Ethics	66
5.2 Social and Global Impact	67
5.3 Environmental Impact	
5.4 Economic Impact	
6.0 Conclusions and Future Work	70
7.0 Appendices	
7.1 MatLab Code	
8.0 References	

Table of Figures

Figure 1: Depiction of the Scapula and its varying regions (Teach Me Anatomy, 2022)	10
Figure 2: Depiction of the Scapula Ridge and its varying joints (Teach Me Anatomy, 2022)	. 11
Figure 3: Depiction of the Humerus and its varying regions (Teach Me Anatomy, 2022)	. 11
Figure 4: The type A pattern of scapulohumeral rhythm. (Bagg, 2016)	12
Figure 5: The type B pattern of scapulohumeral rhythm. (Bagg, 2016)	.13
Figure 6: The type C pattern of scapulohumeral rhythm. (Bagg, 2016)	13
Figure 7: Sketch of the Secondary Rod Preliminary Design (Deane et al., 2022)	.15
Figure 8: Sketches of top, Front and side view of the Scapulothoracic Joint utilized within the Secondary Rod Design (Deane et al., 2022)	. 16
Figure 9: Sketches of top, front, and side view of the unique slot feature representing the	
Scapulothoracic joint (Deane et al., 2022)	.16
Figure 10: Front and back view of model true to scale (Biomedical Modeling Inc.)	17
Figure 11: Left and right view of the model true to scale (Biomedical Modeling Inc.)	.17
Figure 12: Front and Side Views of Full Shoulder Rig Assembly without Soft Tissue (Deane e al., 2022)	t 17
Figure 13: Front and Right-side View of Rib Connections (Deane et al., 2022)	. 18
Figure 14: Corresponding digital plot (left) that was transferred to the plywood (right) for accurate bone positioning (Deane et al., 2022)	. 19
Figure 15: Clavicle and plywood attachments (Deane et al., 2022)	. 20
Figure 16: Second stationary motor mount designed to secure a single Planetary Gearbox	
Nema 17 stepper motor to the top of the t-slot beam (Deane et al., 2022)	. 20
Figure 17: Sketch of glenohumeral joint design (Deane et al., 2022)	.20
Figure 18: Sketch of glenohumeral joint design (Deane et al., 2022)	.21
Figure 19: Isolated sketch of sternoclavicular joint design (Deane et al., 2022)	. 21
Figure 20: Side, top, and front view of shoulder model (Deane et al., 2022)	. 22
Figure 21: Humeral and Scapular Motion	.23
Figure 22: Channel Design Concept	25
Figure 23: Model in the cage	. 26
Figure 24: Vertice motor design (left) and track method design (right)	27
Figure 25: Motor placements along the faces of the cage	.28
Figure 26: Constructed and Realized Cage	29
Figure 27: Potential movements of the Scapula (Physiotutors, 2023)	.30
Figure 28: A simplified view of each of the motor attachment points	32
Figure 29: Key Points of Mass Distribution within the Scapula	32
Figure 30: An example of the Rodrigues' Rotation Formula. (YouTube channel Mathoma)	33
Figure 31: All Angles and Lengths during Humeral Abduction	. 35
Figure 32: The model 17HS13-0404S-PG27 data sheet (StepperOnline, 2020)	. 42
Figure 33: The NEMA model 17HS19-1684S-PG27 motor (StepperOnline, 2020)	. 43
Figure 34: The NEMA model 17HS13-0404S-PG27 motor (StepperOnline, 2020)	. 44

Figure 35: The Young's modulus (10 ⁶ psi) of different materials similar to collagen	45
Figure 36: Braided Fishing Line	46
Figure 37: Braided Fishing Line attached to a 17HS13 motor	47
Figure 38: The figure of the driver (Pololu, 2024)	48
Figure 39: How to adjust current limit potentiometer	49
Figure 40: Posterior View of the Cage	50
Figure 41: Lateral View of the Cage	51
Figure 42: Lateral Midline View of the Cage	52
Figure 43: Top View of the Cage	52
Figure 44: Bottom View of the Cage	52
Figure 45: Supporting Bars of the Cage	53
Figure 46: A simplified view of each of the motor attachment points	54
Figure 47: Acceleration of point A over degrees Abduction	55
Figure 48: Acceleration of point B over degrees Abduction	56
Figure 49: Acceleration of point C over degrees Abduction	57
Figure 50: Acceleration of point D over degrees Abduction	57
Figure 51: Acceleration of point E over degrees Abduction	57
Figure 52: Acceleration of point F over degrees Abduction	58
Figure 53: The 2022-2023 Team's Fully Assembled Physical Circuit (McEvilly et al., 2022)	59
Figure 54: Second Iteration of the Circuitry	60
Figure 55: Updated Schematic of the Circuitry	62
Figure 56: Zoomed in Schematics of the Arduinos	63
Figure 57: The Interactive Scapula Position Graph	67

Table of Tables

Table 1: Abduction Ratio (Average Arm Rotation:Scapular Rotation) (Bagg, 2016)	.12
Table 2: Rib Connection Points (Deane et al., 2022)	18
Table 3: Total rotation of the scapula in all three directions during 30° increments of abductionunder "control" (Yabata, 2022)	31
Table 4: Total rotation of the scapula in all three directions during 30° increments of flexion under "control" (Yabata, 2022)	31
Table 5: Nomenclature Table for Humeral Abduction	34
Table 6: Solving for the Necessary Force and the Position of the Motor	36
Table 7: Rotation of the Humerus during Abduction and Flexion (Yabata, 2022)	37
Table 8: The chosen orientation of the wires from the motor to the driver	48
Table 9: Details of Aluminum Extrusion Rods	50
Table 10: Cost of Materials Used to Modify the Rig	73

1.0 Introduction

The human shoulder stands as a marvel of anatomical complexity, functioning as a pivotal joint essential for a wide range of movements. Despite its significance, educational tools that effectively model the intricacies of the shoulder joint are scarce. In response to this gap, we have continued the mechanization of a life-sized model encompassing the scapula, humerus, and clavicle. This creation aims to replicate the nuanced motion observed during abduction, a process integral to understanding the scapulohumeral rhythm.

The scapulohumeral rhythm refers to the coordinated movement between the scapula (the shoulder blade) and the humerus (the upper-arm bone) during various shoulder motions, particularly abduction. This rhythm plays a crucial role in maintaining the stability and functionality of the shoulder joint. Our mechanization of the model seeks to provide a tangible and visually instructive representation of this intricate interplay, offering a valuable educational resource for students, healthcare professionals, and anyone interested in comprehending the complexities of the human shoulder.

By combining anatomical accuracy with a focus on motion dynamics, our model offers a hands-on approach to learning about the scapulohumeral rhythm. Through its open and spacious design, users can gain a tangible understanding of the spatial relationships and articulations involved in shoulder abduction. This educational tool not only serves as a valuable asset in academic settings but also contributes to a broader appreciation of human anatomy and biomechanics.

In the subsequent sections, we will delve into the key components of our contribution to the shoulder model's development through accurate relative motion, visual representation, exploring the anatomical principles it embodies, and elucidating the educational benefits it presents. From biomechanical insights to practical applications, this life-sized shoulder model is poised to enhance the understanding and appreciation of one of the body's most intricate and essential joints.

2.0 Background

2.1 Scapulohumeral Relationship

The movement of the shoulder consists of two primary bones, those being the the scapula and the humerus, with the clavicle factoring in largely as support and a connection to the sternum in the center of the ribcage. Understanding the relationship of movement between these two bones is key towards accurately modeling and displaying how the shoulder joint functions. The joints connecting these bones are equally as integral to this system, with the two most prominent in the system being the glenohumeral joint, which serves to connect the scapula and the humerus, and the acromioclavicular joint, which connects the scapula and the clavicle.



Figure 1: Depiction of the Scapula and its varying regions (Teach Me Anatomy, 2022)



Figure 2: Depiction of the Scapula Ridge and its varying joints (Teach Me Anatomy, 2022)



Figure 3: Depiction of the Humerus and its varying regions (Teach Me Anatomy, 2022)

Upon abduction of the humerus, the scapula transitions and rotates to accommodate the movement in several stages, however, where the humerus exists and moves on a two dimensional plane assuming there is no antepulsion or retropulsion, the scapula shifts and rotates in a three dimensional manner, with each stage of movement being met with a different facet of rotation. Nikita Igoshin's 2022 Independent Study Project (ISP) focused heavily on these different stages of movement. Igoshin's ISP based its findings on the work from "A Biomedical Analysis of Scapular Rotation During Arm Abduction in the Scapular Plane" written by Stephen D. Bagg, MD, MSc and William J, Forrest MD, MSc. with their results displayed below in Table 1.

Range	Average Arm Rotation (AA) : Scapular Rotation (SR)
0-20.8°	1:0
20.8-81.8°	4.29:1
81.8-139.1°	1.71:1
139.1-170°	4.49:1
20.8-170°	2.25:1

Table 1: Abduction Ratio (Average Arm Rotation: Scapular Rotation) (Bagg, 2016)

The results displayed indicate the ratio of movement between the humerus and the scapula, with the first range being especially of note as until the humerus meets a 20.8° angle with the horizontal axis, there is negligible movement in the scapula. As abduction of the humerus continues, transitional movement of the scapula and rotational movement of the scapula differentiate. The angle of the scapula was found to follow one of three patterns in its rotational movement with respect to the angle of the humerus through the analysis of a large number of subjects. The patterns are depicted below in figures 4, 5, and 6.



Figure 4: The type A pattern of scapulohumeral rhythm. (Bagg, 2016)



Figure 5: The type B pattern of scapulohumeral rhythm. (Bagg, 2016)



Figure 6: The type C pattern of scapulohumeral rhythm. (Bagg, 2016)

The beginning position of the scapula and the point at which the scapula begins to rotate in accordance with the movement of the arm differ greatly between the three patterns, with patterns A and C beginning at a starting angle of around 85 degrees to the vertical axis with pattern B beginning at 95 degrees to the vertical axis. The rate at which the scapula rotates and shifts, while not uniform between the patterns, is largely comparable at the later stages of rotation, with an arm angle of 60 to 80 degrees being the point where the three patterns begin to line up. These results indicate that while the scapulohumeral rhythm differs between individual subjects, it can be approximated and averaged to produce tangible and uniform results. Understanding this, the team opted to use the ratios found by Nikita Igoshin for the scapulohumeral rhythm as the foundation for the project.

2.2 Previous Work on the Model

This project is a continuation of two previous MQP efforts. Two prior teams similarly conducted their own work and tackled the project with different concepts. The 2021-2022 team was largely centered around the development of the bones and structure that the model would rest on and the construction of a rig to demonstrate scapulohumeral rhythm during abduction from 0 to 60 degrees. The group of students procured 3D models of the bones of the shoulder, the scapula, the humerus, and the clavicle as well as the rib cage; to which they added blocks for mounting. The team then 3D printed the models out of PLA (Polylactic acid; a common 3D printing filament). They then mounted the rib cage on to plywood and created a simple rig focused on replicating abductional movement of the shoulder through nylon threads and motors, with partial success.

The 2022-2023 team continued construction of the model, this time focusing on the addition of ligaments and altering motor attachment points to properly replicate the components of an anatomical shoulder. Their team intended to select materials that accurately replicated the function of shoulder tendons, ligaments, and muscles, utilizing a number of different materials to replicate the movement of the humerus.

Our team similarly continued the construction of a rig to replicate movements, utilizing the models produced by the 2021-2022 team and the ideas of an extraneous point of connection from the 2022-2023 team. This was the foundation that our team worked from, building and improving upon their previous designs and accounting for facets and complications the previous teams did not have the time to consider.

2.2.1 2021-2022 MQP: Anatomically Accurate Motorized Shoulder Model with Scapula Movement

The 2021-2022 team gave a clear indication of their design process throughout their project. The shoulder joint, including the scapula, humerus, clavicle, and rib cage would be mounted on a supporting board with motors providing the force needed to conduct movement of the humerus and scapula. Their primary designs were as follows: a secondary rod design and a pulley system trailing the length of the humerus.



Figure 7: Sketch of the Secondary Rod Preliminary Design (Deane et al., 2022)

The above figure displays the shoulder joint in its entirety alongside the secondary rod system, with each part labeled within the table on the left. The Secondary Rod was intended to function as another source of support as well as a guiding hand for scapular rotation. It would achieve these goals by connecting to the scapula through a ball and joint socket in a slot displayed below in figure 8 and figure 9. The slot is curved and hollow so as to allow the scapula to rotate during translation throughout the humeral abduction process. During the rotation and translation of the scapula, the ball is free to slide through the hollow interior of the slot from the center to the edges to accurately mimic the movement of the scapula within the shoulder joint. Similarly during humeral flexion, the ball is free to return to its initial position.



Figure 8: Sketches of top, Front and side view of the Scapulothoracic Joint utilized within the Secondary Rod Design (Deane et al., 2022)



Figure 9: Sketches of top, front, and side view of the unique slot feature representing the Scapulothoracic joint (Deane et al., 2022)

The secondary rod idea was ultimately scrapped by the 2021-2022 team due to the additional parts that would be required to provide the force necessary to replicate the motion of the scapula and humerus. These additional parts physically intersected the secondary rod, preventing the motion from completion. However, this idea of another point of connection is something our team sought to continue with our project. The 2021-2022 team's second primary design was the development of a pulley system that trailed the length of the humerus and scapula as described in section 2.2.1.3.

2.2.1.1 3D Printed Bones

The 2021-2022 team purchased files of the bones that they then 3D printed (Figure 10 and Figure 11). In order to fasten the ribcage to the plywood, a 0.75in x 0.75in rectangular segment was added to each rib at varying lengths to meet at one plane. These fastenings are

shown in Figure 12 and Figure 13. The points of these connections are displayed below in Table 2 and Figure 14.



Figure 10: Front and back view of model true to scale (Biomedical Modeling Inc.)



Figure 11: Left and right view of the model true to scale (Biomedical Modeling Inc.)



Figure 12: Front and Side Views of Full Shoulder Rig Assembly without Soft Tissue (Deane et al., 2022)



Figure 13: Front and Right-side View of Rib Connections (Deane et al., 2022)

Y	Z	Y	Z
38.8411	277.6354	160.753	105.5732
15.2039	238.4864	168.1988	142.0047
5.6789	204.5192	170.3961	171.297
0	164.5711	177.8404	196.7875
0	132.4797	177.8404	223.5691
22.1709	103.6558	165.7629	248.0114
29.3163	74.9135	160.8711	268.1104
42.4981	44.6453	160.8711	290.4167
59.4609	14.6593	149.1488	312.0448
117.1742	0	130.0988	325.9589
145.7492	24.1843	126.6992	345.0089
149.1488	76.648	59.4608	322.4804

Table 2: Rib Connection Points (Deane et al., 2022)



Figure 14: Corresponding digital plot (left) that was transferred to the plywood (right) for accurate bone positioning (Deane et al., 2022)

2.2.1.2 Mounting the Bones and Motors

The bones were mounted and secured to 3D printed blocks which were then fastened to the points on the plywood above as designated in section 2.2.1.1, the clavicle and motors were mounted differently due to their differing functions. The clavicle was mounted to a plywood block through a ball and socket joint that allowed for limited rotational movement. The motors and circuits of the 2021-2022 team were mounted to the rear side of the plywood to not obstruct the movements or visual of the shoulder during its abduction, however, doing so introduced a large amount of friction between the plywood board and the nylon monofilament fishing line utilized in their pulley system as they travel over the top of the plywood. The mounts utilized are displayed below in Figures 15 through 19.



Figure 15: Clavicle and plywood attachments (Deane et al., 2022)



Figure 16: Second stationary motor mount designed to secure a single Planetary Gearbox Nema 17 stepper motor to the top of the t-slot beam (Deane et al., 2022)



Figure 17: Sketch of glenohumeral joint design (Deane et al., 2022)



Figure 18: Sketch of glenohumeral joint design (Deane et al., 2022)



Figure 19: Isolated sketch of sternoclavicular joint design (Deane et al., 2022)

2.2.1.3 2021-2022 Team's Final Product and Data Collection

The 2021-2022 team unfortunately did not have the time to flesh out a fully functioning system to induce movement in the arm, as their project focused much more heavily on creating the models of the bones and bringing them to fruition. That being said, their motorized design had numerous aspects that our team built upon. Their design utilized a motor system that trailed the length of the arm and was attached to the other side of the wooden board used to hold and support the ribcage. While the use of a motor system is the most cost-effective option available, the angle the force was delivered made vertical movement of the arm incredibly intensive, as a

vast majority of the force was pulling the arm into the shoulder socket rather than upwards to display abduction.



Figure 20: Side, top, and front view of shoulder model (Deane et al., 2022)

2.2.2 2022-2023 MQP: Realistic Shoulder Model with Soft Tissue Attachments

The 2022-2023 team focused far more heavily on the addition of materials that would replicate biological functions within the shoulder. The team 3D printed several slabs of polymer materials and attached them to the areas of important tendons and muscle groups to simulate restrictions the shoulder would experience. However, these materials had difficulty remaining attached in the intended manner and often blocked visibility of the more important movements of the model. The inefficiency and other problems introduced by these design options were heavily accounted for when forming our own designs for the rig.

3.0 Methodology

3.1 Design Goals and Constraints

Our team focused our project around numerous different smaller scale goals, each of which were focused on completing our overarching goal of improving upon the design of the two previous MQP team's models to more accurately reflect shoulder movement through abduction of the humerus. To achieve this goal, our team had to operate under a number of constraints, primarily seeking to minimize the cost of the project so as to make the final product as accessible and affordable as possible. Other constraints such as accessible materials and machines to shape these materials would additionally limit the abilities of the team to develop the final product. Further constraints including the size and the transportability of the rig similarly limited potential designs. Our overarching objective would be completed through our other goals which were similar to the previous MQP team's goals in developing an efficient rig that would provide adequate force to lift the humerus whilst rotating and transitioning the scapula appropriately in response to the humerus's position. The rig should additionally be capable of completing the motion displayed in figure 21 an indefinite number of times without requiring frequent replacement of parts.



Figure 21: Humeral and Scapular Motion

Our first goal was to calculate and replicate accurate motion of the scapula. The scapula shifts and rotates on a three dimensional plane and has a constantly shifting center of rotation alongside a complex geometric shape that can make accurately replicating its movement

difficult. Our efforts to reflect this movement are detailed in section 3.3 Scapular Rotation. Our second goal was to achieve the full desired range of motion, raising the humerus from 0° to 120° with the vertical axis. Our final goal to improve the previous design of the model was to accurately replicate relative motion between the scapula and humerus. While the movements of the scapula and humerus aren't causing relative motion between one another in the model like they would in the body, the motors in our rig would accurately replicate the relative movement of the bones associated with the scapulohumeral rhythm described in our design concepts and rig designs in section 3.2.

3.2 Rig Design Concepts

To achieve the scapulohumeral rhythm and motion desired, our team created two preliminary design concepts with a number of smaller variations.

3.2.1 Channel Design Concept

One preliminary design concept our team focused on was the idea of implementing a channel to follow the length of the humerus, mimicking the design of the previous MQP team's final implemented design. Where the previous teams had the wires and supporting materials on the outside surface of the humerus, this design would involve drilling a channel into and along the length of the humerus and a portion of the scapula, where the wires would be run through and over the collar into the plywood board supporting the model where the motors would pull from. This design is displayed below in figure 22.



Figure 22: Channel Design Concept

The channel design allows for a much cleaner final model without the clutter of the previous team's approach in attaching motors, wires, and other materials to the exterior of the humerus and allows for the most visibility of the bones as they complete their motions. However, the channel design invites a number of complications including the question of force vector angles and friction of the internal wires as they travel through the bones and their connections.

3.2.2 Cage Design Concept

Our other preliminary design focused on creating a structure surrounding the bones to avoid the complications of the motor weights on the bones and suboptimal force vector angles. We referred to this structure as the cage. The bones of the ribcage and shoulder would be surrounded by a cage as shown in Figure 23 with the dimensions 0.5 meters x 0.8 meters x 0.8 meters, with the width of the cage being the shortest dimension of the cage.



Figure 23: Model in the cage

Along the faces of the cage, motors would be placed and supported by beams with wire connections to the humerus and scapula to pull on the bones with adequate forces and angles to accurately replicate shoulder movement and scapulohumeral rhythm. The original placements of the motors were at the vertices of the cage where they would pull a final motor along the open face of the cage to guarantee the optimal angles for force vectors, however, this would restrict visibility and introduce difficulties in reinforcing the motor under its own force as well as simply being inefficient. Other placements were considered such as a moving track along the face of the cage, however, the final placements were decided to be constrained to one position and unmoving throughout the shoulder movements. These original motor placements and design concepts are shown below in figure 24.



Figure 24: Vertice motor design (left) and track method design (right)

The exact placements of these motors are displayed in Figure 25 below. The materials utilized for the wires and the models of motors are discussed in sections 3.5.2 and 3.5.1 respectively. The motors were mounted to the cage through the use of 3D printed motor mounts left by the previous team. Additional mounts were printed using the same file the previous team used.



Figure 25: Motor placements along the faces of the cage

The primary benefits of the cage design were the visibility of the model and its movements it provided due to the lack of additional items cluttering up the humerus and scapula,

which additionally made it much more efficient to design methods to move the humerus and scapula since factors such as the weight of the motors did not have to be considered when focusing on the movement of the bones.



Figure 26: Constructed and Realized Cage

3.3 Scapular Rotation

Calculating how the scapula will rotate while the humerus abducts and flexes was vital to this project. The scapula's unorthodox shape and uneven distribution of mass caused the process to be complex, with the numerous directions of movement and constant uneven shifts of the scapula discussed in section 2.1 further complicating the process.



Figure 27: Potential movements of the Scapula (Physiotutors, 2023)

The position of the scapula was found by calculating where all five motor attachment points, A, B, C, D, and E, will be relative to the initial position of motor attachment point A. The attachment points are displayed below in figure 28. The amount of scapular rotation in each direction–upward rotation, anterior tilt, and external rotation–in 30° increments of humeral rotation were taken from tables 3 and 4. The team then assumed that rotation during each 30° increment was linear in its rotation, with the positions indicated on the scapula transitioning in a consistent fashion within the 30° periods.

Humeral	Upward rotation angle (°) ^a		Anterior tilt angle (°) ^b		External rotation angle (°) ^c		ngle (°) ^c		
abdction									
angle	Control	Thorax	Thorax	Control	Thorax	Thorax	Control	Thorax	Thorax
		flexion	extention		flexion	extention		flexion	extention
0°-30°	3.39 ±	1.60 ±	$2.05 \pm$	$-6.04 \pm$	$-4.95 \pm$	-5.55 ±	1.29 ±	0.93 ±	1.19 ±
	4.23	3.23	5.38	2.55	3.11	3.28	1.65	1.50	1.72
30°-60°	$26.34 \pm$	17.89 ±	$22.38 \pm$	$-22.28 \pm$	$-24.13 \pm$	$-19.31 \pm$	5.93 ±	6.45 ±	4.51 ±
	11.94	8.98	14.78	7.55	7.66	7.57	2.90	3.66	3.63
60°–90°	41.25 ±	34.09 ±	$35.20 \pm$	$-31.45 \pm$	$-39.25 \pm$	$-26.47 \pm$	8.92 ±	8.92 ±	6.09 ±
	9.22	9.32	15.45	11.47	12.99 ^d	10.80	4.52	4.52 ^d	5.15
90°-120°	$50.85 \pm$	$45.80 \pm$	42.58 ±	33.49 ±	$-46.24 \pm$	$-26.97 \pm$	10.71 ±	15.69 ±	6.46 ±
	11.20	10.60	16.34	16.02	17.61 ^d	14.55	6.08	4.95 ^d	5.94

Difference amount of change in scapula angle during shoulder abduction

Table 3: Total rotation of the scapula in all three directions during 30° increments of abductionunder "control" (Yabata, 2022)

Humeral	Upward rotation angle (°) ^a		Upward rotation angle (°) ^a Anterior tilt angle (°) ^b		External rotation angle (°) ^c		ngle (°) ^c		
angle	Control	Thorax flexion	Thorax extention	Control	Thorax flexion	Thorax extention	Control	Thorax flexion	Thorax extention
0°-30°	1.55 ± 2.52	2.29 ± 2.55	0.83 ± 3.15	1.82 ± 2.38	1.91 ± 3.45	2.75 ± 2.68	-1.13 ± 1.52	-1.00 ± 1.76	-1.38 ± 1.07
30°-60°	9.14 ± 5.03	9.14 ± 5.33	7.91 ± 6.98	0.08 ± 6.01	0.09 ± 8.65	1.01 ± 6.38	-1.93 ± 1.76	-1.75 ± -3.09	-2.89 ± 2.97
60°–90°	15.03 ± 5.60	14.50 ± 5.62	14.85 ± 8.01	-3.80 ± 9.15	-4.65 ± 12.99	-2.71 ± 9.61	-1.95 ± 3.69	-1.95 ± 3.69	-3.71 ± 4.27
90°–120°	26.35 ± 7.02	24.84 ± 5.99	23.09 ± 8.62	-11.16 ± 13.45	-14.99 ± 16.86	-6.74 ± 13.52	0.22 ± 4.49	3.61 ± 5.64 ^d	-3.47 ± 5.88

Difference amount of change in scapula angle during shoulder flexion

Table 4: Total rotation of the scapula in all three directions during 30° increments of flexionunder "control" (Yabata, 2022)



Figure 28: A simplified view of each of the motor attachment points

The team then defined two axes of rotation, one for upward rotation and one for anterior tilt, IJ and IK respectively. These axes of rotation were formed from the vector between two points on the scapula as displayed in figure 29. External rotation of the scapula is centered around a point called the instantaneous center of rotation, or the ICR.



Figure 29: Key Points of Mass Distribution within the Scapula

The team then applied the Rodrigues' Rotation Formula, a formula for rotating a vector around a normal vector being used as an axis of rotation, for each direction of rotation in increments of 0.1° of humeral abduction or flexion up to 120°, and averaged the three outputs to find the new position of each motor attachment point. This strategy was also applied to points I, J, and K, so that the axes of rotation stayed within the scapula. An example of the Rodrigues' Rotation Formula is shown below. In this case, vector v is being rotated around the normal vector n by theta degrees to create vector v prime. The equation is the form of the Rodrigues' Rotation Formula the team used in their math.



Figure 30: An example of the Rodrigues' Rotation Formula (YouTube channel Mathoma)

3.4 Humeral Abduction

Scapulohumeral rhythm was modeled for both humeral abduction and flexion. Humeral movement was confined to a two dimensional plane in its movements while it is moving in each direction, with additional safeguards in attached motors ensuring the humerus does not stray from its path of movement. The force required to lift the humerus was calculated through trigonometry utilizing a number of variables displayed in the nomenclature table below.

Variable	Definition (unit)			
L _H	Length of the humerus (meters)			
W _H	Mass of the humerus (kilograms)			
ai	Initial angle between the humerus and the vertical axis (°)			
Θi	Initial angle between the humerus and the thread (°)			

Θ'	Angle between the humerus and the thread post humeral abduction (°)			
L _{Ti}	Initial thread length (meters)			
L _T '	Thread length following humeral abduction (meters)			
Δα	Desired change in angle alpha (°)			
t	Time (seconds)			
L _{HM}	Distance between the top of the humerus and the motor (meters)			
β _i	Initial angle between the motor wire and the humerus (°)			
β'	Angle β after abduction of the humerus (°)			
d _{Hbot}	Distance traveled by the bottom of the humerus (meters)			
V _{Hbot}	Velocity of the bottom of the humerus (meters/second)			
a _{Hbot}	Acceleration of the bottom of the humerus (meters/second ²)			
F _{gopp}	The force of gravity opposite to the direction of movement (Newtons)			
F _{Net}	Net force acting on the humerus (Newtons)			
F _{Required}	Required force to lift the humerus (Newtons)			

Table 5: Nomenclature Table for Humeral Abduction



Figure 31: All Angles and Lengths during Humeral Abduction

The steps to solve for the force necessary to lift the humerus and the ideal position of the motor are displayed below in table 6.

Step Description	Equation
Beginning with the length of the humerus, the value for L_{HM} can be found using the law of cosines:	$L_{\rm HM} = \sqrt{(L_{\rm H}^2 + L_{\rm Ti}^2 - 2L_{\rm H}L_{\rm Ti}cos\Theta_{\rm i})}$
Using other given variables, L_{Ti} , L_{H} , and Θ_i the team can similarly solve for β_i	$(\sin\beta_i)/L_{Ti} = (\sin\Theta_i)/L_H$
With the initial value for the angle found, the team can then find the angle after the abduction of the humerus, β ':	$\beta' = \beta_i - \Delta \alpha$
With these variables defined, the desired thread length following humeral abduction can be found:	$L_{\rm T}' = \sqrt{(L_{\rm H}^2 + L_{\rm HM}^2 - 2L_{\rm H}L_{\rm HM}\cos\beta')}$
With this final thread length determined, other values can be solved to determine the other kinematics needed to solve for the force required to lift the humerus.	$\Theta' = \sin^{-1}(L_{\rm HM}\sin\beta'/L_{\rm T}')$
--	--
With the new angle found, the team can then solve	$d_{\rm Hbot} = \Delta \alpha \pi L_{\rm H} / 180$
for the distance, velocity, and acceleration of the	$v_{Hbot} = d_{Hbot}/t$
humerus.	$v_c = 2v_{Hbot}$
	$a_{Hbot} = v_c/t$
New angles, E and C, must first be defined to relate	$C = 90 - (a_i + \Delta a)$
F _g and solve for the force:	$\mathbf{E} = (\Theta' + \mathbf{C}) - 90$
Now with all values known, the force can be solved	$F_g = 9.8 W_H$
for using Newton's Second Law:	$F_{gopp} = F_g cos E$
	$F_{net} = W_H a_{Hbot}$
	$F_{required} = F_{net} + F_{gopp}$

Table 6: Solving for the Necessary Force and the Position of the Motor

To track the position of the humerus, the team calculated where motor attachment point F will be relative to the initial position of motor attachment point A. This was done by setting the center of rotation at motor attachment point B and applying the Rodrigues' Rotation Formula and calculating the positions of motor attachment point F in increments of 0.1° of abduction and flexion up to 120°. In order to account for the movement of point B during this rotation, the team used geometry to "drag" the attachment point to its proper final position.

Givens:

All values are in meters and degrees. All distances are measured from the origin which is set at the bottom left corner of the cage as seen in figure 40.

Initial Wire Attachment Points:	Initial Axes Points:	Motor Placement:
A = [0.073;0.507;0.445];	I = [0.074; 0.508; 0.434];	motorA = [-0.045;.534;0.825];
B = [0.203; 0.509; 0.458];	J = [0.2; 0.508; 0.434];	motorB = [0.18;0.534;0.825];

C = [0.083; 0.525; 0.312];	K = [0.074; 0.508; 0.313];	motorCPosY = [0.44;0.852;0.54];
D = [0.154;0.522;0.383];		motorD = [0.185;0;0.5];
E = [0.077; 0.524; 0.388];		motorE = [0.19;0.852;0.366];
F = [0.11;0.035;-0.3696];		motorFZ = [0.405;0.534;0.825];
		motorFPosY = [0.57;0.852;0.54];

The variation in angle of the bones was based on a study (Yabata, 2022), where the angles are measured in cartesian coordinates relative to an origin.

motorFNegY = [0.505;0;0.48];

Angle of Abduction	Total Upward Rotation	Total Anterior Tilt	Total External Rotation
0-30°	-3.39°	6.04°	1.29°
30-60°	-22.95°	16.24°	4.46°
60-90°	-14.91°	9.17°	2.99°
90-120°	-9.60°	2.04°	1.79°
Angle of Flexion			
0-30°	-1.55°	-1.82°	-1.13°
30-60°	-7.59°	1.74°	0.80°
60-90°	-5.89°	3.88°	-0.02°
90-120°	-11.32°	7.36°	2.17°

Table 7: Rotation of the Humerus during Abduction and Flexion (Yabata, 2022)

Calculations:

- How to rotate the scapula. The example provided will be point E moving as the humerus abducts from 0° to 10°. Each step calculated by the code is 0.1°; the 10° increment is used for illustrative purposes.
 - 1.1. Find how much E externally rotates during this interval using the table and step size.
 - 1.1.1. $1.29^{\circ}/(30/10) = 0.43^{\circ}$

- 1.2. Find the normal axis of rotation, \widehat{IK} in this case
 - 1.2.1. Move the origin to I
 - 1.2.2. I = [0.074; 0.508; 0.434] [0.074; 0.508; 0.434] = [0; 0; 0] K = [0.074; 0.508; 0.313] - [0.074; 0.508; 0.434] = [0; 0; - 0.121]E = [0.077; 0.524; 0.388] - [0.074; 0.508; 0.434] = [0.003; 0.016; -0.046]

1.2.3. Find
$$\overline{IK}$$

1.2.3.1. $\overline{IK} = [0; 0; -0.121] - [0; 0; 0] = [0; 0; -0.121]$

1.2.4. Find *IK*

1.2.4.1.
$$|\overline{IK}| = \sqrt{0^2 + 0^2 + (-0.121)^2} = 0.121$$

 $\widehat{IK} = [0; 0; -0.121] / 0.121 = [0; 0; -1]$

- 1.3. Derive the Rodrigues Rotation Formula: a formula for rotating a point a desired number of degrees around any normal axis
 - 1.3.1. $\vec{E}_{parallel} = component of \vec{E}$ that is parallel to \widehat{IK} $\vec{E}_{perpendicular} = component of \vec{E}$ that is perpendicular to \widehat{IK} $\vec{E} = \vec{E}_{parallel} + \vec{E}_{perpendicular}$
 - 1.3.2. $\vec{E'}_{parallel} = \vec{E}_{parallel}$ 1.3.2.1. $\vec{E'} = \vec{E}_{parallel} + \vec{E'}_{perpendicular}$
 - 1.3.3. $\Theta = angle from 1.1$

1.3.3.1.
$$\vec{E'} = \vec{E}_{parallel} + \cos\Theta * \vec{E}_{parallel} + \sin\Theta * (\hat{IK} \times \vec{E})$$

1.3.3.2. $\vec{E'} = (1 - \cos\Theta) * \vec{E}_{parallel} + \cos\Theta * \vec{E} + \sin\Theta * (\hat{IK} \times \vec{E})$

1.3.4. Final Rodrigues Rotation Formula

1.3.4.1.
$$\vec{E'} = (1 - \cos\Theta) * (\vec{E} \cdot \hat{IK}) * \hat{IK} + \cos\Theta * \vec{E} + \sin\Theta * (\hat{IK} \times \vec{E})$$

1.4. Using the Rodrigues Rotation Formula

1.4.1.
$$\vec{E'} = (1 - \cos(0.43)) * ([0.003; 0.016; -0.046] \cdot [0; 0; -1]) * [0; 0; -1] + \cos(0.43) * [0.003; 0.016; -0.046] + \sin(0.43)*([0; 0; -1] x [0.003; 0.016; -0.046])$$

 $\vec{E'} = [0.00288; 0.01602; -0.046]$

1.5. Move the origin back

1.5.1.
$$\vec{E'_{IK}} = [0.00288; 0.01602; -0.046] + [0.074; 0.508; 0.434] = [0.07688; 0.52402; 0.388]$$

1.6. Repeat steps 1.1, 1.2, and 1.4 with the axes as IJ and the axis through the ICR orthogonal to the plane containing I, J, and K, making sure that the origin is moved to the correct point, I for IJ and the ICR.

1.6.1.
$$\vec{E'}_{IJ} = [0.077; 0.52238; 0.38747]$$

 $\vec{E'}_{ICR} = [0.07836; 0.524; 0.38929]$

1.7. Average the three E values

1.7.1.
$$\vec{E'} = [0.07741; 0.52347; 0.38825]$$

- 1.8. Repeat 1.1 1.7 for points A, B, C, D, I, J, K, and the ICR
- 1.9. Repeat 1.1 1.8 until total abduction angle is 120° while always using the most recent I, J, K and ICR values
- 1.10. Repeat 1.1 1.9 to find all points for flexion
- 2. How to find how point F rotates. The example used will also be the humerus abducting from 0° to 10° .
 - 2.1. Move the origin to B, as this is where the humerus is rotating around

2.1.1.
$$B = [0.203; 0.509; 0.458] - [0.203; 0.509; 0.458] = [0; 0; 0]$$

 $F = [0.21; 0.515; 0.13] - [0.203; 0.509; 0.458] = [0.007; 0.006; -0.328]$

2.2. Set the axis of rotation for abduction

2.2.1.
$$\hat{n} = [0; -1; 0]$$

2.3. Use the Rodrigues Rotation Formula

- 2.3.1. $\vec{F'} = (1 \cos(10)) * ([0.007; 0.006; -0.328] \cdot [0; -1; 0]) * [0; -1; 0] + \cos(10) * [0.007; 0.006; -0.328] + \sin(10)*([0; -1; 0] \times [0.007; 0.006; -0.328]) = [-0.05001; 0.006; -0.32423]$
- 2.4. Move origin back
 - 2.4.1. $\vec{F}' = [-0.05001; \ 0.006; \ -0.32423] + [0.203; \ 0.509; \ 0.458] = [0.15299;$ $0.515; \ 0.13377]$
- 2.5. Now the team must account for the movement of the humeral head in the XZ plane
 - 2.5.1. B' B = [0.2029; 0.509; 0.4589] [0.203; 0.509; 0.458] = [-0.0001; 0; 0.0009]
 - 2.5.2. F' (B' B) = [0.15299; 0.515; 0.13377] [-0.0001; 0; 0.0009] = [0.15309; 0.515; 0.13287]
- 2.6. Now the team must account for the movement of the humeral head along the y-axis while keeping F in the same XZ plane.
 - 2.6.1. Create a circle, in the same XZ plane as F, of all possible locations of F with the knowns of the length of the humerus and the location of the humeral head at B'

2.6.1.1. Center of Circle =
$$[X_{B'}; 0.515; Z_{B'}] = [0.2029; 0.515; 0.4589]$$

2.6.1.2. Radius of Circle =
$$\sqrt{Length_{Humerus}^2 - (Y_{B'} - 0.515)^2} = \sqrt{0.3683^2 - (0.509 - 0.515)^2} = 0.13561$$

- 2.6.2. Create a vector from the center of the circle to the current position of F and normalize it
 - 2.6.2.1. $\vec{R} = [0.2029; 0.515; 0.4589] [0.15309; 0.515; 0.13287] = [0.04981; 0; 0.32603]$

2.6.2.2.
$$|\vec{R}| = [0.04981; 0; 0.32603] / 0.32894 = [0.15173; 0; 0.98842]$$

- 2.6.3. Find the closest point on the circle to the current position of F. This point on the circle is F'
 - 2.6.3.1. F' = [0.15309; 0.515; 0.13287] + 0.3683 * [0.15173; 0; 0.98842] = [0.20897; 0.515; 0.49691]

- 2.7. Repeat 2.1 2.6, using the correct B and B', until the total abduction angle is 120°
- 2.8. Repeat 2.1 2.7, using [1; 0; 0] as the axis of rotation, to find the values of F for flexion

3.5 Material Selection

3.5.1 Motor Selection

The motors used by the 2021-2022 and 2022-2023 teams were the Nema 17 Stepper Motor model number 17HS19-1684S-PG27, which will be shortened to the 17HS19 model for the duration of the report. The 17HS19 model has a number of specifications that allows for efficiency, however, our team had concerns in utilizing these motors due to the amperage requirements of 1.2 Amps. To circumvent this issue, our team looked for other models of Nema 17 motors and decided on the use of the 17HS13-0404S-PG27 motor, which will similarly be shortened to the 17HS13 model. The qualities that make the 17HS13 motor the ideal are the rated current and the maximum permissible torque. The specifications for the 17HS13 motor are given in figure 32 below.



Figure 32: The model 17HS13-0404S-PG27 data sheet (StepperOnline, 2020)

The low rated current of 0.4 amps allows the motor to function on low currents, meaning the circuit will need to involve fewer pieces and draw less current and power into the system, exerting a lower strain on the equipment. The required voltage input for the motors was 12V. The max permissible torque of 3 Newton meters for each motor is capable of pulling on the humerus and scapula with adequate force to induce the rotation and transitions necessary to replicate scapulohumeral rhythm in the shoulder. The two previous teams both used five of the model 17HS19 motors to pull the humerus along their trail of wire that led along the length of the humerus.



Figure 33: The NEMA model 17HS19-1684S-PG27 motor (StepperOnline, 2020)

For our rig design, discussed in section 3.2, ten model 17HS13 motors were implemented to ensure correct positioning and stability of the humerus and scapula during humeral abduction. We attached the motors to the cage using 3D printed mounts. The locations of the motor placement within the cage are similarly discussed in section 3.2.



Figure 34: The NEMA model 17HS13-0404S-PG27 motor (StepperOnline, 2020)

3.5.2 Connecting Wires

Utilizing the correct materials for each material that the team is trying to simulate is crucial to the project's success. The first team developed a shoulder model with the bones properly assembled. For the material of the bones, they used PLA due to its rigid properties and ease of molding through 3D printing into the desired shapes. Our team will be similarly utilizing PLA for this aspect of the model.

The first team utilized nylon wires as a way of moving the bones to their desired location due to their strength and flexibility, whilst still remaining cost-effective. However, this team did not account for the issue of creep. The deformation caused by the creep changes the length of the nylon wires and thus changes the calculations that must be done to pull on the bones with the right force vectors. The second team utilized rubber threads to better operate under creep conditions, however, the friction caused between the rubber threads and other components of the rig hampered movements. To eliminate these issues, our team decided to change the material used for the connecting wires.



Figure 35: The Young's modulus (10⁶ psi) of different materials similar to collagen

To decide on the material replacing the nylon wires, our team calculated the stress and strain that the nylon is under while it is pulling on the bones and the temperature of the room the model is in. With these two variables, the team calculated the strain rate of each nylon strand using the equation $\varepsilon = A\sigma^n exp(\frac{-Q}{RT})$, where ε is the strain rate, A and n are material based constants, σ is the applied stress, Q is the activation energy of the creep mechanism, R is the gas constant, and T is the ambient temperature in Kelvin. The applied stress, σ , was calculated with the equation $\sigma = \frac{F}{A}$, where F is the applied force delivered to the wire from the motor and A is the area of the wire that the force is delivered across. The area of the wire was calculated to be 0.00126 in² or 3.243 * 10⁻⁶ m² with an applied load of 11 kg. These calculations were completed in MatLab, with a final value of 3.4 MPa for the stress. We performed our calculations based on a wire with a diameter of 0.04 inches and material constants A = 3.73 * 10⁻⁵ and n = 4.

Several options for the connective wires were discussed including a number of polymers and metals as well as the previously utilized nylon 6 wires, however, each material encountered different issues. The nylon 6 wires, as discussed above, were susceptible to creep. Silver alloys were too soft and could easily be damaged. Copper alloys would be susceptible to environmental factors. Finally, various steels were similarly proposed but were too resistant to bending, far too expensive, and had high friction coefficients, making their movements in the model more difficult than necessary. With this in mind, our team considered PTFE coated wires to mitigate friction between the wire and any surfaces contacted, however, a suitable product was not commercially available.

Ultimately, the team utilized braided nylon fishing line to achieve a balance between all sought characteristics. With the values attained, the wire would need to be capable of supporting a maximum of 13.8 MPa without experiencing creep, whilst still being capable of repeatedly bending in the process of pulling the shoulder model into position and not inhibiting the movement of the humerus and scapula all of which braided fishing line achieves.



Figure 36: Braided Fishing Line



Figure 37: Braided Fishing Line attached to a 17HS13 motor

3.5.3 Motor Driver

We used a stepper motor driver in order to control the motors. The brand name for the driver is 'Allegro's A4988 DMOS Microstepping Driver with Translator and Overcurrent Protection'. We bought our driver from Amazon which was manufactured by Shenzhenshi Yongfukang Technology Co. The A4988 has 16 pins of which our team utilized 10 pins. The Vmot and GND are connected to a 9V battery. The 2B, 2A, 1A, and 1B pins are connected to the four motor wires. Figure 39 below shows the corresponding colors to letters. The wiring of the motor to the driver can be arranged in a multitude of variations as long as the one coil on the motor is not connected to two different coil outputs on the driver.



Figure 38: The figure of the driver (Pololu, 2024)

A4988	Motor Datasheet	Color
1A	A+	Green
1B	A-	Black
2A	B+	Red
2B	B-	Blue

Table 8: The chosen orientation of the wires from the motor to the driver

In order to set up the motor driver we needed to adjust the current limit potentiometer. The following equation is used to find the voltage reference for the driver.

$$V_{ref} = 8 * I_{max} * R_{cs}$$

The I_{max} is the max current rated for the motors we were using which is 0.4A. R_{cs} is the current sense resistance of the driver motor, $R_{cs} = 0.1 \Omega$. Therefore, our V_{ref} would be 0.32V. The figure below shows one of the ways to adjust the current limit potentiometer. For the V Ω port we used an alligator clip that was connected to a screwdriver that we turned the potentiometer with.



Figure 39: How to adjust current limit potentiometer

Another important note about the A4988 motor driver is the temperatures it can operate at. The maximum junction temperature is 150°C, attaching the heat sink to the motor driver aids to lower the temperature but the drivers can still cause burns with direct contact at operational temperatures.

3.6 Cage Construction and Implementing a Matlab Code

3.6.1 The Cage

The cage surrounding the rig was constructed utilizing aluminum extrusion rods present available in the WPI MQP labs. The exact dimensions of the rods vary and are listed in table 9 below.

Part No.	Aluminum Extrusion Rod Dimensions (Length X Width X Depth)	Quantity
1010 T-Slotted Profile - Four Open T-Slots	0.912m X 1.00" X 1.00"	2

1010 T-Slotted Profile - Four Open T-Slots	2.44m X 1.00" X 1.00"	2
20-2020 T-Slotted Profile - Four Open T-Slots	0.695m X 20mm X 20mm	2
1003-s Smooth Surface T-Slotted Profile - Three Adjacent Open T-Slots	2.44m X 1.00" X 1.00"	1

Table 9: Details of Aluminum Extrusion Rods

The minimum dimension of the cage is 0.5 meters by 0.8 meters by 0.8 meters. Different views of the cage are displayed below in figures 39 through 43. The varying dimensions displayed in the figures are from the different sizes of aluminum extrusion available to the team.



Figure 40: Posterior View of the Cage



0.811m

Figure 41: Lateral View of the Cage



0.811m

Figure 42: Lateral Midline View of the Cage



Figure 43: Top View of the Cage





Figure 44: Bottom View of the Cage

Another issue we saw that was due to aging was wood rot in the original rib cage mount. We replaced the old plywood with a ³/₄ inch plywood. Additional bars were installed to further support the cage under the weight of the rig and correctly position the motors in accordance with Figure 25 in section 3.2.1 to pull on the 3D printed bones within. To minimize any clutter and ensure visibility of the rig, the wires for each motor run along the bars of the cage where they conjoin in the circuit on the backside of the board.



Figure 45: Supporting Bars of the Cage

3.6.2 Coding and the MATLAB Model

The team lead for this portion of the project was Cameron Leffler, who wrote MATLAB code to calculate the positions of each of the six wire attachment points on the scapula and humerus relative to the origin for every 0.1° of arm abduction and flexion. The following documentation explains how the code calculates all of the relative positions of the wire attachment points and wire lengths.

The Code is split into four scripts of code; Setup, Abduction, Flexion, and App code scripts, all of which are run using Arduinos connected in the circuit. The setup code defines the

relative positions of the motors and of the motor attachment points for all possible angles of arm abduction and flexion.



Figure 46: A simplified view of each of the motor attachment points

It does this in the manner stated in section 3.2 using the Rodrigues' Rotation Formula. It then calculates the distance from each motor to the motor attachment point, giving the length of thread between the attachment point and the motor and creates a global matrix of all thread lengths given a desired abduction or flexion angle. The app script graphs the movement of the rig through previous calculations and transmits that information to the abduction and flexion code. The abduction and flexion scripts take an input of a desired angle of abduction or flexion and change the thread length to the correct amount based on the desired abduction or flexion by rotating the motors. Cameron then took these relative positions and made an interactive 3D plot in MATLAB where a user can input an amount of abduction or flexion and the program will show four views of where the wire attachment points will be in 3D space, one angled view from

behind and above, one view from straight on, one view from directly above, and one view straight on from the side, as shown in Figure 56 in Section 3.8.2.

Using the MATLAB code we can determine the position of each point throughout scapular and humeral movement. In figures 46 through 51, we determined the acceleration of each point throughout abduction. The data for the graphs below is made up of over 10,000 calculated data points so the addition of variables was made to make the data more digestible in an excel spreadsheet. The duration of abduction and flexion can be altered by changing the total time and the degrees between data points can be altered to get an average of the data for each point. We wanted to portray the change in acceleration of each point to convey the complexity of the scapulohumeral rhythm.



Figure 47: Acceleration of point A over degrees Abduction



Figure 48: Acceleration of point B over degrees Abduction





Figure 49: Acceleration of point C over degrees Abduction

Figure 50: Acceleration of point D over degrees Abduction



Figure 51: Acceleration of point E over degrees Abduction



Figure 52: Acceleration of point F over degrees Abduction

We can also use the thread lengths to check our work along with measuring the distances by hand. We can measure xyz manually to make sure the ijk vectors are correct and measure the thread length to confirm the overall vector acceleration. The wire we are using is not elastic under the delivered force so we know that the delta thread length is relatively true.

3.6.3 Circuit Design

The circuitry utilized for the rig underwent a large change from the 2021-2022 team's circuit design. The 2021-2022 circuit design, displayed below in figure 54, is a simple parallel circuit designed to deliver power and current to each motor simultaneously so as to ensure each motor receives enough voltage to run, however, the equipment used to build the circuit was inadequate to serve our purpose. The breadboard utilized to build the circuit was not built to handle a current higher than 1 amp, which is almost doubled when all five motors are run through the circuit. Additionally a single 9-volt battery is incapable of delivering enough power to support five motors at once, let alone the ten our team would implement.



Figure 53: The 2022-2023 Team's Fully Assembled Physical Circuit (McEvilly et al., 2022)

To resolve these issues, our team opted to implement a larger power source, with the intention of utilizing numerous 9-volt batteries at once with plans to ultimately plug the system into a wall outlet. Further alterations to the circuit include the addition of a second Arduino board, to ensure enough pins were available for the ten microprocessor-motor systems. The next iteration of the circuit design is displayed below in Figure 55.



Figure 54: Second Iteration of the Circuitry

To circumvent the breadboard's limitations, breadboards were removed from the circuit, instead utilizing space studs, cable nail in clips, and terminal blocks to connect and complete the circuit. The team also experienced trouble operating both Arduinos. This was due to the fact that we had not adjusted the current limit potentiometer and we also had the power source of all ten motor drivers connected to the Arduinos in series. This overloaded one of the Arduinos causing it to short and turn off. To fix these issues we moved from breadboards to 10 circuit terminal blocks instead of the breadboard. We added five more 9V batteries to power each motor individually which we were hoping would improve our low voltage. To fix the shorting Arduino,

instead of all ten of the motor drivers being powered by the 5V output from the Arduino which were run in one series we split it into two series of five. Each series is powered by their own 5V Arduino output pin . The updated schematic is shown in Figure 56 below. One of our largest oversights came with the needed voltage. The original consensus was that 9V would have been enough to power the motors but that was for the previous model of motor used. The 17HS19-1684S-PG27 only needed 2.8V in order to turn, whereas the 17HS13-0404S-PG27 needed around 12V to properly run.



Figure 55: Updated Schematic of the Circuitry



Figure 56: Zoomed in Schematics of the Arduinos

On the A4988 motor driver the RESET pin needs to be HIGH so our team connected it to the SLEEP pin which pulls HIGH by default. We also used a 100uF capacitor to regulate power surges between our motor power supply and the driver. When picking a place for the capacitor our team placed it as close to the driver as possible so there is less surge throughout the wire, especially if the distance between the batteries and the motor drivers is larger.

3.7 Testing

The team successfully constructed the cage and attached the motors to the surrounding frame. We were unable to conduct testing of the fully constructed rig due to time constraints. While the team did not get all 10 motors rotating at once, the team did test to see if the motor positions were able to achieve the correct relative positions of the 3D printed bones. In doing so, all ten motors were tested for their ability to function as expected in the rig as described in the following sections.

3.7.1 Testing Relative Position

The team chose to test the relative positions at rest, then in intervals of 30 degrees of abduction and flexion, ending at 120 degrees of abduction and flexion. The team made this decision as in between these checkpoints the motion of points on the scapula is linear. The team manually set the thread lengths to their desired lengths and suspended the scapula and humerus from the motors in order to show a proof of concept.

3.7.2 Motor Calibration

Motor calibration tests were based on finding out how much the motor changed the thread length as a function of how many seconds the motor spun. The first step was finding the angular velocity of the motor. To do this, the team made a simple MatLab code that can spin the motor for an inputted number of seconds. The team then imputed one, five, and ten seconds and measured the degrees of rotation with tape and a protractor. To refine this measurement, the team made a MatLab code that inputted desired degrees of rotation and spun the motor as such. The team ran the code to ten full rotations, or 3600 degrees, and manually changed the conversion coefficient–that converts from degrees to seconds of rotation–until the motor could spin 3600 degrees within half a degree of precision. The team then calculated, based on the diameter of the spool, how much thread the motor was moving per second and updated the conversion coefficient–that converts from thread length moved to seconds of rotation–in the final code.

3.7.3 Testing Individual Motors

In order to test individual motors, the team wrote a simple MatLab code that talked to one motor at a time and was easily switched between which motor was being tested. Motors were identified to work upon the activation of the code, where the motor would begin to rotate in accordance with the written script. Motors that did not rotate or did not rotate in accordance with the script were identified as dysfunctional. Once it was determined if a motor was functioning properly or not the team troubleshooted to identify the error, further details on the troubleshooting process are provided in the following section.

3.7.4 Troubleshooting

Our team troubleshooted for errors throughout the process of constructing the rig. During the construction of the cage, an angle bar was utilized to ensure proper angles within the cage as well as physical exertion upon the bars to ensure they were properly secured and firm in their placement. Motor tests revealed a number of faulty motors that were promptly replaced or repaired through corrections within the code.

Throughout construction of the rig, circuitry for the rig proved to be the most complex to troubleshoot, as the presence of numerous components, each with their own individual ability to experience error, posed difficult to sort through. Removal of the breadboards, as described in section 3.3, was the first correction to attempt to circumvent the amperage limit that was imposed by the hardware. Further corrections to the circuitry were completed with the use of a DC power supply, allowing the circuitry to experience a number of different amperes and voltages combinations to expose further faults within the wiring by observing if the motors would rotate with varying inputs. By providing a wider range of voltages and amperes, we were able to identify what range of voltages and amperes worked with the circuit and ten 17HS13 motors. Additionally, a voltmeter was utilized to measure the voltage across circuitry components, to ensure the proper distribution of voltage across all key components. Several faulty capacitors were replaced in addition to corrections made to the Arduino boards.

3.8 Results

While relative motion of the scapular and the humerus was not achieved, the team made progress that will be valuable for the continuation of this project, by Cameron Leffler, in D-term and for future teams that continue the project.

3.8.1 Partially Completed Rig

The team constructed a nearly complete model, including the surrounding frame to support the humerus and scapula. The cage is capable of supporting all necessary forces to manipulate the bones to reproduce anatomically correct movements. The circuitry was similarly completed and is capable of delivering sufficient power and commands from the MatLab scripts to all ten motors of the rig.

3.8.2 Interactive Scapula Position Graph

The team lead for this portion of the project, Cameron Leffler, created an interactive graph within MatLab that, with two inputs between 0 and 120 degrees, one for abduction and one for flexion, would replicate the proper anatomical movements of a humerus and scapula within a confined graph. This motion is viewable from numerous varying angles. The viewer can select a degree from a bar located at the bottom of the window with intervals of 1 degree. Figure 57 shows the interactive app set at 90 degrees of humeral flexion. Due to constraints with setting the camera in the MatLab app maker, the axes in the figure are not correct, however, the relative motion is the same.



Figure 57: The Interactive Scapula Position Graph

4.0 Discussion

Through review of the results, the team was able to determine our successes and shortcomings throughout the duration of the project.

4.1 Motor Location and Function

The team was successful in relocating the motors to positions that would more easily allow freedom of movement for the rig through abduction and flexion. The surrounding motors pulling from locations external to the humerus and scapula provided far more ideal force vectors to manipulate the movement of the bones. The motors were summarily successful in their calibration with the Arduino code to accurately rotate the correct amount and retract the correct length of braided nylon fishing wire. Unfortunately, due to time constraints, a full test of the motors system was unable to be completed, leaving a lack of experimental results to supplement the theoretical calculations.

4.2 Materials Selection

The team was successful in selecting proper materials to mitigate complications during humeral abduction and flexion. The removal of the restrictive materials placed on the model by the 2022-2023 team, meant to simulate the muscles, ligaments, tendons, and other biological material within the shoulder allowed for freedom in designing the rig, with a larger focus on properly replicating the relative motion between the scapula and humerus. To minimize friction from connection points between the motors and the scapula and humerus, a selection of nylon braided fishing wire was utilized for the low friction coefficient alongside the durability presented by the material. Similar to section 4.1, time constraints limited the team's ability to experimentally test the materials selected.

4.3 MatLab Code and Interactive Position Graph

The team was successful in writing MatLab code to control 10 motors to accurately recreate the scapular humeral rhythm during humeral abduction and flexion and used MatLab to create an interactive position graph showing the relative motion of the scapula and humerus

during humeral abduction and flexion. To the team's knowledge, both of these codes did not exist before this project. The relative positions from the code will be saved and used by Cameron Leffler to create a 3D animation of the scapular humeral rhythm during D-term.

5.0 Broader Impacts

5.1 Engineering Ethics

Over the duration of our project, we sought to be in accordance with the engineering code of ethics implemented by the American Society of Mechanical Engineering. Our team strived to use our understanding of engineering for the benefits of human welfare, increase the competence and prestige of the engineering profession, and to act with honesty and impartiality when developing this project. These fundamental principles integral to the engineering guidelines were held to high standards as we completed our project. This project was completed with the goal of aiding medical students through the use of an educational device, a goal created through the intention of providing beneficial material for the betterment of others. Through this report, we have strived to be truthful to establish integrity within our work. It is our sincere hope that our efforts reflect well on the engineering field.

Our project was a continuation of two prior MQP teams' efforts on the shoulder model, with great effort made to build off the foundation they created whilst respecting their own accomplishments. Their results and data were supplemented by our own research and findings to create a sufficient project that reflects the efforts of all the teams who have dedicated their work towards this shoulder model.

Finally, we sought to complete this project to the fullest extent of our abilities and produce the highest quality model we were capable of creating, utilizing numerous reputable sources and our own engineering expertise to create as accurate a model as possible.

5.2 Social and Global Impact

The primary intention of this model is to be utilized as an educational tool for the improved understanding of the shoulder joint and the complexities that lie within it. This educational tool would go on to benefit a subset of people within society, primarily students seeking to study human anatomy to become doctors. This tool would further these student's understanding of the complexities of the human shoulder, producing more educated doctors who will go on to provide better treatment to patients suffering from injuries, conditions caused by chronic illnesses or genetic defects. Additional applications towards injury prevention are possible such as utilizing the model to explore protective gear for athletes.
5.3 Environmental Impact

The new materials used for the model and rig include braided nylon fishing line wires and aluminum extrusion to construct the cage for the model. The aluminum extrusion used to construct the cage was reused from previous projects. The braided nylon fishing line connecting the bones to the 17HS13 motors is biodegradable and environmentally friendly. motors used within the rig are composed of a number of metals and electronic components which are not environmentally friendly. Finally, the current power source for the rig, 9V batteries are not environmentally friendly, however, the rig can be modified to draw power from electrical outlets to reduce this waste. The model is projected to be utilized over a large quantity of years without requiring significant replacements to parts or significant upkeep, allowing a minimum amount of materials to be wasted over the course of the model's lifespan.

Materials utilized for our rig by the previous teams includes PLA 3D printed bones and 3D printed motor mounts which are biodegradable and will decompose in 12 weeks time when disposed of. Other materials utilized by the previous teams included KT Tape, Formlabs Elastic 50A, Formlabs Flexible 80A, and Thermoplastic Polyurethane, all of which we removed from our design and model to minimize waste and clutter of the rig, and reduce our environmental impact.

5.4 Economic Impact

The cost of our modifications to the model was relatively low, and is tabulated below. The braided fishing line and aluminum extrusion were inexpensive. The stepper motors contributed to a far larger comparative expense. The electrical components consisting of the arduino board and microprocessors along with all the other circuitry additionally contributed to the cost.

Material	Quantity	Independent Vendor	Cost
17HS13 Motors	10	Yes	~\$300
A4988 Stepper Motor Drivers	10	Yes	~\$13
Braided Nylon Fishing Line	1	Yes	~\$12
Aluminum Extrusion (Varying Sizes)	19	No	Not Applicable
Cable Nail-in Clips	12	Yes	~\$8
Terminal Blocks	12	Yes	~\$120

Table 10: Cost of Materials Used to Modify the Rig

The projected cost of the model is about \$453 USD. While initially costly, when compared to current commercially available educational models, the model becomes far more reasonable in price as most other models range from around \$300 to \$900, with varying features included within the product. Most models on the lower end do not include any joints to display movement, rather they are entirely static. Many models on the upper end of the range include very detailed depictions of muscles and ligaments, but still do not display movement.

6.0 Conclusions and Future Work

The goal of this project was to demonstrate accurate scapulohumeral rhythm by revising and improving upon the previous team's iterations. To accomplish this, the team stripped the model of the restrictive soft tissue materials and constructed a surrounding cage where motors would deliver force upon the humerus and scapula to direct its movements to replicate the scapulohumeral rhythm during abduction and flexion of the humerus. The team then wrote MatLab scripts to direct movements of the motors to produce our desired motion. Although time constraints prevented further testing of the model and developments of the system, the modifications our team made to the model present potential for a fully realized system that replicates proper anatomical movements.

With the aforementioned time constraints restricting the progress of the model, our team believes that future teams can complete the construction of the rig and upon discovery of any potential errors, can continue to develop it. Of the potential foci future teams can follow, we suggest to first continue development of the current model, finalizing its testing to provide experimental results to supplement theoretical calculations. Additionally, altering the power source for the model from batteries to outlets to avoid repeated purchases of power sources is heavily suggested. Other avenues of development involve a number of options. The development of specialized printed bones to replicate medical conditions such as arthritis would allow for easily swappable parts to bolster the versatility of the model. Adjustments to the model to account for the removed materials from the 2022-2023 team could similarly be made to allow for the biological materials that the 2022-2023 team sought to replicate in their model.

With a fully realized rig and modifications made to the power source, this model has incredible potential as an educational tool. Medical experts that have a deeper understanding of injuries of any type, are capable of treating a greater number of patients. The value in providing medical students a deeper and more thorough understanding of the human shoulder cannot be understated.

7.0 Appendix: MatLab Code

Provided below is the complete code written by our team. The team lead for this portion of the project, as previously stated, was Cameron Leffler

setup.n	abCode.m 🗙 motortest2.m 🗙 🕇	
1	global currentAbIndex;	A A
2	<pre>global currentFlexIndex;</pre>	
3	global abMatrix;	
4	<pre>global flexMatrix;</pre>	
5		
6	<pre>global xCoordsPositionMatrixAb;</pre>	
7	<pre>global yCoordsPositionMatrixAb;</pre>	
8	<pre>global zCoordsPositionMatrixAb;</pre>	
9	global FMasterXAb;	
10	global FMasterYAb;	
11	global FMasterZAb;	
12		
13	<pre>global xCoordsPositionMatrixFlex;</pre>	
14	<pre>global yCoordsPositionMatrixFlex;</pre>	
15	<pre>global zCoordsPositionMatrixFlex;</pre>	
16	global FMasterXFlex;	
17	global FMasterYFlex;	
18	global FMasterZFlex;	
19		
20	global masterPointAAb	
21	global masterPointBAb	
22	global masterPointCAb	
23	global masterPointDAb	
24	global masterPointEAb	
25	global masterPointFAb	
26		
27	global masterPointAFlex	
28	global masterPointBFlex	
29	global masterPointCFlex	
30	global masterPointDFlex	
31	global masterPointEFlex	
32	global masterPointFFlex	
33		
34	global motorA	
35	global motorB	-
36	4	•

6			_
setup.m	🗙 abCode.m 🗶 motortest2.m 🗶 🕇		
36	global motorCPosY	A	
37	global motorCNegY		
38	BIODAT MOLOLCY		
10	global motorF		
40	global motorF7		
41	global motorFPosY		
43	global motorFNegY		
44	Paanaa		
45	currentFlexIndex = 1;		
46	currentAbIndex = 1;		
47			
48	%%%Initial Positions		
49	A = [0;0;0];		
50	B = [0.11; 0.035; 0.013];		
51	C = [0.03; -0.004; -0.141];		
52	D = [-0.015;0;-0.03];		
53	E = [0.07;0;-0.03];		
54	F = [0.11; 0.035; -0.3696];		
55	b = b;		
56	I = [0.01; 0; -0.01]; $I = [0.10; 0; -0.01];$		
57	J = [0.12; 0; -0.01]; $K = [0.01; 0; -0.15];$		
50	$\kappa = [0.01, 0; -0.10];$		
60	motorA = [A(1), A(2), A(2), A(3)]		
61	motorB = [B(1); B(2); 0.38203];		
62	motorCPosY = [C(1); 0.39; C(3)];		
63	motorCNegY = [C(1); -0.39; C(3)];		
64	<pre>motorCX = [0.474; C(2); 0];</pre>		
65	<pre>motorD = [D(1); -0.39; D(3)];</pre>		
66	motorE = [E(1); 0.39; E(3)];		
67	motorFZ = [0.237; 0; 0.38203];		
68	<pre>motorFPosY = [0.237; 0.39; 0];</pre>		
69	motorFNegY = [0.237; -0.39; 0];		
			_
70		•	
70 71	Verene Andressian	• •	
70 71	eeeeeee Abduction	▼ ▶	
70 71	A abCode m X motortest2 m X +	•	
70 71	A abCode.m × motortest2.m × +	▼ ▶	
70 71 setup.m 71 72	A abCode.m × motortest2.m × +		4
70 71 setup.m 71 72 73	<pre> abCode.m × motortest2.m × + %%%%%%%%% Abduction %Other starting conditions </pre>		A
70 71 setup.m 71 72 73 74	<pre> Abduction Abduction Abduction X02000000000 X02000000 X0200000 X020000 X02000 X0200 X0</pre>	*	
70 71 setup.m 71 72 73 74 75	<pre> abCode.m × motortest2.m × + %%%%%%%%% Abduction %Other starting conditions armAngleAbduction = 0; increaseAbA = 0.1; </pre>		A
70 71 setup.m 71 72 73 74 75 76	<pre> abCode.m × motortest2.m × + %%%%%%%%%% Abduction %Other starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); </pre>		A
70 71 setup.m 71 72 73 74 75 76 77	<pre> abCode.m × motortest2.m × + xxxxxxxx Abduction %Other starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; </pre>		A
70 71 setup.m 71 72 73 74 75 76 77 78	<pre> abCode.m × motortest2.m × + %%%%%%%%%% Abduction %Other starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; </pre>		
70 71 8etup.m 71 72 73 74 75 76 77 78 79	<pre> abCode.m × motortest2.m × + %%%%%%%%%%% Abduction %Other starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [A,B,C,D,E,I,J,K]; </pre>		A
70 71 71 72 73 74 75 76 77 77 78 79 80	<pre> Abduction Abduction Abduction X abCode.m × motortest2.m × + XXXXXXXXX Abduction XOther starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [A,B,C,D,E,I,J,K]; xCoordsPositionMatrixAb = [0,0,0,0,0]; </pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81	<pre> abCode.m × motortest2.m × + %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%</pre>		4
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82	<pre> abCode.m × motortest2.m × + %%%%%%%%% Abduction %Other starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [0,0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; zCoordsPositionMatrixAb = [0,0,0,0,0]; </pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83	<pre> abCode.m × motortest2.m × + %%%%%%%%%%% Abduction %Other starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [A,B,C,D,E,I,J,K]; xCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; zCoordsPositionMatrixAb = [0,0,0,0,0]; FMasterAb = [0.11;0.035;-0.3696]; </pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83 84 82	<pre></pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 52	<pre></pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 85	<pre> Abduction Abduction X abCode.m × motortest2.m × + %%%%%%%%% Abduction %Other starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [A,B,C,D,E,I,J,K]; xCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; zCoordsPositionMatrixAb = [0,0,0,0,0]; FMasterAb = [0:11;0:035;-0:3696]; FMasterYAb = [0;0]; FMasterZAb = [0;0]; FMasterZAb = [0;0]; </pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88	<pre> Abduction Abduction X abCode.m × motortest2.m × + XXXXXXXXX Abduction XOther starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [A,B,C,D,E,I,J,K]; xCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; rdasterXab = [0:1];0:035;-0:3696]; FMasterXAb = [0;0]; FMasterZAb = [0;0]; wipelengthMatrixAb = allWipelength(motorA motorB motorCosY motorCherY </pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 85 86 87 88	<pre></pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 85 86 87 88 89 90	<pre> abCode.m × motortest2.m × + xxxxxxxxx Abduction XOther starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; rMasterAb = [0:11;0:035;-0:3696]; FMasterXAb = [0;0]; FMasterZAb = [0;0]; wireLengthMatrixAb = allWireLength(motorA,motorB,motorCPosY,motorCNegY, motorCX,motorD,motorE,motorFPosY,motorFNegY,motorFZ,A,B,C,D,E,F); thread enothMatrixAb = [0:0]: </pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83 84 82 83 84 85 86 87 88 89 90 91	<pre> abCode.m × motortest2.m × + %%%%%%%%%% Abduction %Other starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [A,B,C,D,E,I,J,K]; xCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; zCoordsPositionMatrixAb = [0,0,0,0,0]; FMasterAb = [0:11;0.035;-0.3696]; FMasterAb = [0;0]; FMasterZAb = [0;0]; wireLengthMatrixAb = allWireLength(motorA,motorB,motorCPosY,motorCNegY, motorCX,motorD,motorE,motorFPosY,motorFNegY,motorFZ,A,B,C,D,E,F); threadLengthMatrixAb = [0:0]; </pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83 84 82 83 84 85 86 87 88 89 90 91 92	<pre></pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 88 89 90 91 92 92 93	<pre> Abduction X abCode.m × motortest2.m × + %%%%%%%%% Abduction %Other starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; rCoordsPositionMatrixAb = [0,0,0,0,0]; rMasterXAb = [0;0]; FMasterYAb = [0;0]; WireLengthMatrixAb = allWireLength(motorA,motorB,motorCPosY,motorCNegY, motorCX,motorP,motorE,motorFPosY,motorFNegY,motorFZ,A,B,C,D,E,F); threadLengthBMatrixAb = [0;0]; threadLengthCPosYMatrixAb = [0;0]; threa</pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 84 85 86 87 88 89 90 91 92 93 93 94	<pre> Abduction X abCode.m × motortest2.m × + XXXXXXXXX Abduction XOther starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [A,B,C,D,E,I,J,K]; xCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; zCoordsPositionMatrixAb = [0,0,0,0,0]; rMasterXAb = [0;0]; FMasterXAb = [0;0]; FMasterXAb = [0;0]; wireLengthMatrixAb = allWireLength(motorA,motorB,motorCPosY,motorCNegY, motorCX,motorD,motorF,motorFPosY,motorFNegY,motorFZ,A,B,C,D,E,F); threadLengthBMatrixAb = [0;0]; threadLengthCNegYMatrixAb = [0;0]; threadLengthCNMatrixAb = [0;0]; threadLengthCNMatrixAb = [0;0]; threadLengthCNMatrixAb = [0;0]</pre>		
70 71 setup.m 71 72 73 74 75 76 77 78 80 81 82 83 84 85 86 87 88 85 86 87 88 89 90 91 92 93 94 95	<pre></pre>		
70 71 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 85 86 87 88 89 90 91 92 93 94 95 96	<pre> abCode.m × motortest2.m × + xxxxxxxxxx Abduction XOther starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; rMasterAb = [0:0]; FMasterAb = [0:0]; FMasterAb = [0;0]; threadLengthMatrixAb = [0;0]; threadLengthBMatrixAb = [0;0]; threadLengthCPosYMatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthCMatrixAb =</pre>		
70 71 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 85 86 87 88 89 90 91 92 93 94 92 93 94 95 96 97	<pre> abCode.m × motortest2.m × + xxxxxxxxxx Abduction XOther starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; rMasterAb = [0:11;0:035;-0:3696]; FMasterXAb = [0;0]; FMasterZAb = [0;0]; wireLengthMatrixAb = allWireLength(motorA,motorB,motorCPosY,motorCNegY, motorCX,motorD,motorE,motorFPosY,motorFNegY,motorFZ,A,B,C,D,E,F); threadLengthMatrixAb = [0;0]; threadLengthMatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthMatrixAb = [0;0]; threadLengthMatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthMatrixAb = [0;0]; threadLengthMatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthFNatrixAb = [0;0]; threadLengthFNatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; thr</pre>		
70 71 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 86 87 88 89 90 91 92 93 94 95 96 97 98	<pre> Abduction X abCode.m × motortest2.m × + %</pre>		
70 71 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99	<pre> Abduction X abCode.m × motortest2.m × + XXXXXXXXX Abduction XOther starting conditions armAngleAbduction = 0; increaseAbA = 0.1; radIncreaseAbA = deg2rad(increaseAbA); armAngleAbductionMatrix = [armAngleAbduction;0]; masterPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; yCoordsPositionMatrixAb = [0,0,0,0,0]; rCoordsPositionMatrixAb = [0,0,0,0,0]; rMasterAb = [0:11;0.035;-0.3696]; FMasterYAb = [0;0]; FMasterYAb = [0;0]; twrealengthMatrixAb = allWireLength(motorA,motorB,motorCPosY,motorCNegY, motorCX,motorD,motorE,motorFPosY,motorFNegY,motorFZ,A,B,C,D,E,F); threadLengthBMatrixAb = [0;0]; threadLengthCPosYMatrixAb = [0;0]; threadLengthCNatrixAb = [0;0]; threadLengthCMatrixAb = [0;0]; threadLengthEMatrixAb = [0;0]; threadLengthFEMatrixAb = [0;0]; threadLengthEMatrixAb = [0;0]; threadLengthEMatrixAb = [0;0]; threadLengthEXMatrixAb = [0;0]; threadLengthEXMatrixAb = [0;0]; threadLengthEXMatr</pre>		

•

counter = 4; i = 1;

while armAngleAbduction <= 120</pre>

setu	p.m 🗶 abCode.m 🗶 motortest2.m 🗶 🕂		
107	%%%Set correct angles or rotation	A	•
108	if armAngleAbduction <= 30		E
109	%SA angles for 0-30 degrees of AA		
110	upwardRotation = deg2rad(-3.39);		
111	anteriorTilt = deg2rad(6.04);		
112	externalRotation = deg2rad(1.29);		
113			
114	<pre>upwardRotationStep = upwardRotation/(30/increaseAbA);</pre>		
115	anteriorTiltStep = anteriorTilt/(30/increaseAbA);		
116	<pre>externalRotationStep = externalRotation/(30/increaseAbA);</pre>		
117			
118	ICR = A;		
119	end		
120			
121	if armAngleAbduction > 30 && armAngleAbduction <= 60		
122	%SA angles for 30-60 degrees of AA		
123	upwardRotation = deg2rad(-26.34)-deg2rad(-3.39);		
124	anteriorTilt = deg2rad(22.28)-deg2rad(6.04);		
125	externalRotation = deg2rad(5.93)-deg2rad(1.29);		
126			
127	<pre>upwardRotationStep = upwardRotation/(30/increaseAbA);</pre>		
128	<pre>anteriorTiltStep = anteriorTilt/(30/increaseAbA);</pre>		
129	externalRotationStep = externalRotation/(30/increaseAbA);		
130			
131	ICR = A;		
132	end		
133			
134	if armangleAbduction > 60 && armangleAbduction <= 90		
135	%SA angles for 60-90 degrees of AA		
136	upwardKotation = deg2rad(-41.25)-deg2rad(-26.34);		
137	anteriofilit = degzrad(51.45)-degzrad(22.26);		
138	externalRotation = degrad(8.92)-degrad(5.93);		
140	u_{B}		
140	aptonionTiltStop = aptonionTilt/(30/increaseADA);		
141	anteriorificstep = anteriorifit/(S0/IncreaseADA);		

```
abCode.m × motortest2.m × +
setup.m
          \times
142
                   externalRotationStep = externalRotation/(30/increaseAbA);
                                                                                                                              - 🔺
143
144
                  ICR = A;
145
              end
146
               if armAngleAbduction > 90 && armAngleAbduction <= 120</pre>
147
                  %SA angles for 90-120 degrees of AA
148
                   upwardRotation = deg2rad(-50.85)-deg2rad(-41.25);
149
150
                   anteriorTilt = deg2rad(33.49)-deg2rad(31.45);
151
                   externalRotation = deg2rad(10.71)-deg2rad(8.92);
152
                  upwardRotationStep = upwardRotation/(30/increaseAbA);
153
                  anteriorTiltStep = anteriorTilt/(30/increaseAbA);
154
                   externalRotationStep = externalRotation/(30/increaseAbA);
155
156
157
                   changeX = B(1) - A(1);
158
                   changeY = B(2) - A(2);
                   changeZ = B(3) - A(3);
159
160
                  ICR = [changeX/2;changeY/2;changeZ/2];
               end
161
162
              BforF = B;
163
164
165
               masterPositionMatrixAb(counter:counter+2,:) = fullRotation(A,B,C,D,E,I,...
166
                  J,K,upwardRotationStep,anteriorTiltStep,externalRotationStep,ICR);
167
               FMasterAb(counter:counter+2,1) = humeralAbduction(BforF,B,F,radIncreaseAbA);
168
              A = masterPositionMatrixAb(counter:counter+2,1);
169
              B = masterPositionMatrixAb(counter:counter+2,2);
170
              C = masterPositionMatrixAb(counter:counter+2,3);
171
172
              D = masterPositionMatrixAb(counter:counter+2,4);
173
              E = masterPositionMatrixAb(counter:counter+2,5);
174
               I = masterPositionMatrixAb(counter:counter+2,6);
               J = masterPositionMatrixAb(counter:counter+2,7);
175
               K = masterPositionMatrixAb(counter:counter+2,8);
176
                   177
```

setup.m 🗶 abCode.m 🗶 motortest2.m 🗶 🕇 F = FMasterAb(counter:counter+2,1); ^ armAngleAbductionMatrix((counter-1)/3) = armAngleAbduction; threadLengthAMatrixAb((counter-1)/3) = wireLengthMatrixAb(1,1); threadLengthBMatrixAb((counter-1)/3) = wireLengthMatrixAb(1,2); threadLengthCPosYMatrixAb((counter-1)/3) = wireLengthMatrixAb(1,3); threadLengthCNegYMatrixAb((counter-1)/3) = wireLengthMatrixAb(1,4); threadLengthCXMatrixAb((counter-1)/3) = wireLengthMatrixAb(1,5); threadLengthDMatrixAb((counter-1)/3) = wireLengthMatrixAb(1,6); threadLengthEMatrixAb((counter-1)/3) = wireLengthMatrixAb(1,7); threadLengthFPosYMatrixAb((counter-1)/3) = wireLengthMatrixAb(1,8); threadLengthFNegYMatrixAb((counter-1)/3) = wireLengthMatrixAb(1,9); threadLengthFZMatrixAb((counter-1)/3) = wireLengthMatrixAb(1,10); wireLengthMatrixAb = allWireLength(motorA,motorB,motorCPosY,motorCNegY,... motorCX,motorD,motorE,motorFPosY,motorFNegY,motorFZ,A,B,C,D,E,F); armAngleAbduction = armAngleAbduction + increaseAbA; counter = counter + 3; end while i < size(masterPositionMatrixAb,1)/3</pre> xCoordsPositionMatrixAb(i;:) = masterPositionMatrixAb(i*3-2,1:5); yCoordsPositionMatrixAb(i,:) = masterPositionMatrixAb(i*3-1,1:5); zCoordsPositionMatrixAb(i,:) = masterPositionMatrixAb(i*3,1:5); i = i+1;end i = 1; while i < size(FMasterAb,1)/3</pre> FMasterXAb(i,:) = FMasterAb(i*3-2,:); FMasterYAb(i,:) = FMasterAb(i*3-1,:);

setup.m	n 🗙 abCode.m 🗙 motortest2.m 🗶 🕂		
212	<pre>FMasterZAb(i,:) = FMasterAb(i*3,:);</pre>		Δ
213			
214	i = i+1;		
215 ^L	end		
216			
217			
218	abMatrix = [threadLengthAMatrixAb,threadLengthBMatrixAb,threadLengthCPosYMatrixAb,		
219	threadLengthCNegYMatrixAb, threadLengthDMatrixAb,threadLengthEMatrixAb,		
220	threadLengthFPosYMatrixAb,threadLengthFNegYMatrixAb,threadLengthFZMatrixAb];		
221			
222			
223	%%% Reset Initial Positions		
224	A = [0;0;0];		
225	B = [0.11; 0.035; 0.013];		
226	C = [0.03;-0.004;-0.141];		
227	D = [-0.015;0;-0.03];		
228	E = [0.07;0;-0.03];		
229	F = [0.11;0.035;-0.3696];		
230	I = [0.01; 0; -0.01];	- 10	
231	J = [0.12; 0; -0.01];		
232	K = [0.01; 0; -0.15];		
233			
234	%%%%%%%%% Flexion Time		
235			
236	wireLengthMatrixFlex = allWireLength(motorA,motorB,motorCPosY,motorCNegY,		
237	<pre>motorCX,motorD,motorF,motorFPosY,motorFNegY,motorFZ,A,B,C,D,E,F);</pre>		
238			
239	% Other starting conditions		
240	armAngleFlex = 0;		
241	increaseFlexA = 0.1;		
242	radIncreaseFlexA = deg2rad(increaseFlexA);		
243	armAngleFlexMatrix = [armAngleFlex;0];		
244			
245	<pre>masterPositionMatrixFlex = [A,B,C,D,E,I,J,K];</pre>		
246	xCoordsPositionMatrixFlex = [0,0,0,0,0];	-	
247		•	

setup.	m	🗙 abCode.m 🗶 motortest2.m 🗶 🕂		
247		<pre>yCoordsPositionMatrixFlex = [0,0,0,0,0];</pre>		Δ
248		<pre>zCoordsPositionMatrixFlex = [0,0,0,0,0];</pre>		
249		FMasterFlex = [0.11;0.035;-0.3696];		
250		<pre>FMasterXFlex = [0;0];</pre>		
251		FMasterYFlex = [0;0];		
252		FMasterZFlex = [0;0];		
253				
254		threadLengthAMatrixFlex = [0;0];		
255		<pre>threadLengthBMatrixFlex = [0;0];</pre>		
256		<pre>threadLengthCPosYMatrixFlex = [0;0];</pre>		
257		<pre>threadLengthCNegYMatrixFlex = [0;0];</pre>		
258		<pre>threadLengthCXMatrixFlex = [0;0];</pre>		
259		threadLengthDMatrixFlex = [0;0];		
260		<pre>threadLengthEMatrixFlex = [0;0];</pre>		
261		threadLengthFPosYMatrixFlex = [0;0];		
262		threadLengthFNegYMatrixFlex = [0;0];		
263		threadLengthFZMatrixFlex = [0;0];		
264				
265		counter = 4;		
266		i = 1;		
267				
268	-	while armAngleFlex <= 120		
269				
270				
271		%%%Set correct angles or rotation		
272		if armAngleFlex <= 30		
273		%SA angles for 0-30 degrees of AA		
274		upwardRotation = deg2rad(-1.55);		
275		anteriorlilt = deg2rad(-1.82);		
276		externalRotation = deg2rad(-1.13);		
277				
278		upwardKotationStep = upwardKotation/(30/increaseFlexA);		
2/9		anteriorilitytep = anteriorilit/(30/increaseFlexA);		
280		externalKotationStep = externalKotation/(30/increaseFlexA);		
281			•	
282				

setup.	m 🗙 abCode.m 🗶 motortest2.m 🗶 🕂		
282	ICR = A;	<u>ـ</u>	A
283	end		E
284			
285	if armAngleFlex > 30 && armAngleFlex <= 60		
286	%SA angles for 30-60 degrees of AA		
287	upwardRotation = deg2rad(-9.14)-deg2rad(-1.55);		
288	anteriorTilt = deg2rad(-0.08)-deg2rad(-1.82);		
289	externalRotation = deg2rad(-1.93)-deg2rad(-1.13);		
290			
291	upwardRotationStep = upwardRotation/(30/increaseFlexA);		
292	anteriorTiltStep = anteriorTilt/(30/increaseFlexA);		
293	externalRotationStep = externalRotation/(30/increaseFlexA);		
294			
295	ICR = A;		
296	end		
297			
298	if armAngleFlex > 60 && armAngleFlex <= 90		
299	%SA angles for 60-90 degrees of AA		
300	upwardRotation = deg2rad(-15.03)-deg2rad(-9.14);		
301	anteriorTilt = deg2rad(3.80)-deg2rad(-0.08);		
302	externalRotation = deg2rad(-1.95)-deg2rad(-1.93);		
303			
304	upwardRotationStep = upwardRotation/(30/increaseFlexA);		
305	anteriorTiltStep = anteriorTilt/(30/increaseFlexA);		
306	externalRotationStep = externalRotation/(30/increaseFlexA);		
307			
308	LCR = A;		
309	end		
310			
311	1f armAngleFlex > 90 && armAngleFlex <= 120		
312	%SA angles for 90-120 degrees of AA		
313	upwardKotation = degZrad(-26.35)-degZrad(-15.03);		
314	anteriorilit = deg/rad(11.16)-deg/rad(3.80);		
315	externalKotation = deg2rad(0.22)-deg2rad(-1.95);		
316		· · · · · · · · · · · · · · · · · · ·	
31/	•	•	

setup.	.m 🛪 abCode.m 🛪 motortest2.m 🛪 🕂		
317	upwardRotationStep = upwardRotation/(30/increaseFlexA);		A
318	<pre>anteriorTiltStep = anteriorTilt/(30/increaseFlexA);</pre>		
319	<pre>externalRotationStep = externalRotation/(30/increaseFlexA);</pre>		
320			
321	changeX = B(1) - A(1);		
322	changeY = B(2) - A(2);		
323	changeZ = B(3) - A(3);		
324	<pre>ICR = [changeX/2;changeY/2;changeZ/2];</pre>		
325	end		
326			
327	BforF = B;		
328			
329	<pre>masterPositionMatrixFlex(counter:counter+2,:) = fullRotation(A,B,C,D,E,</pre>		
330	I,J,K,upwardRotationStep,anteriorTiltStep,externalRotationStep,ICR);		
331	<pre>FMasterFlex(counter:counter+2,1) = humeralFlexion(BforF,B,F,radIncreaseFlexA);</pre>		
332			
333	<pre>A = masterPositionMatrixFlex(counter:counter+2,1);</pre>		
334	<pre>B = masterPositionMatrixFlex(counter:counter+2,2);</pre>		
335	<pre>C = masterPositionMatrixFlex(counter:counter+2,3);</pre>		
336	<pre>D = masterPositionMatrixFlex(counter:counter+2,4);</pre>		
337	<pre>E = masterPositionMatrixFlex(counter:counter+2,5);</pre>		
338	<pre>I = masterPositionMatrixFlex(counter:counter+2,6);</pre>		
339	<pre>J = masterPositionMatrixFlex(counter:counter+2,7);</pre>		
340	<pre>K = masterPositionMatrixFlex(counter:counter+2,8);</pre>		
341	<pre>F = FMasterFlex(counter:counter+2,1);</pre>		
342			
343	armAngleFlexMatrix((counter-1)/3) = armAngleFlex;		
344			
345	<pre>threadLengthAMatrixFlex((counter-1)/3) = wireLengthMatrixFlex(1,1);</pre>		
346	threadLengthBMatrix+lex((counter-1)/3) = wireLengthMatrixFlex(1,2);		
347	threadLengthLPosYMatrix+Lex((counter-1)/3) = wireLengthMatrix+Lex(1,3);		
348	threadLengthCNegYMatrixLex((counter-1)/3) = wireLengthMatrixFlex(1,4);		
349	<pre>threadLength(Xhatrix+lex((counter-1)/3) = wireLengthMatrix+lex(1,5);</pre>		
350	<pre>tnreadlengtnUmatrix+lex((counter-1)/3) = wireLengthWatrix+lex(1,b); then dlue to the third to the the third to the the third to the the the the the the the the the the</pre>		
351	<pre>tnreadLengtntMatrixFlex((counter-1)/3) = WireLengthMatrixFlex(1,/);</pre>	•	
352	4	•	

setu	ıp.m	🗙 abCode.m 🗙 motortest2.m 🗙 🕇		
352		threadLengthFPosYMatrixFlex((counter-1)/3) = wireLengthMatrixFlex(1,8);		Ē
353		threadLengthFNegYMatrixFlex((counter-1)/3) = wireLengthMatrixFlex(1,9);		1
354		<pre>threadLengthFZMatrixFlex((counter-1)/3) = wireLengthMatrixFlex(1,10);</pre>		i
355				i
356		wireLengthMatrixFlex = allWireLength(motorA,motorB,motorCPosY,motorCNegY,		l
357		<pre>motorCX,motorD,motorE,motorFPosY,motorFNegY,motorFZ,A,B,C,D,E,F);</pre>		f
358				
359		armAngleFlex = armAngleFlex + increaseFlexA;		
360		counter = counter + 3;		
361	L	end		
362				
363	Ę	while i < size(masterPositionMatrixFlex,1)/3		
364		<pre>xCoordsPositionMatrixFlex(i,:) = masterPositionMatrixFlex(i*3-2,1:5);</pre>		
365		<pre>yCoordsPositionMatrixFlex(i,:) = masterPositionMatrixFlex(i*3-1,1:5);</pre>		
366		<pre>zCoordsPositionMatrixFlex(i,:) = masterPositionMatrixFlex(i*3,1:5);</pre>		
367				
368		i = i+1;		
369	L	end		
370				
371		i = 1;		
372	_			
3/3	딕	while 1 < size(FMaster+lex,1)/3		
3/4		<pre>FMasterXFlax(1,:) = FMasterFlax(1*3-2,:); FMasterVFlax(1*) = FMasterFlax(1*3-2,:); FMasterVFlax(1*) = FMasterFlax(1*3-2,:); FMasterVFlax(1*3-2,:); FMasterV</pre>		
375		<pre>FMasterYFlex(1,:) = FMasterFlex(1*3-1,:); FMasterZFlav(4:) FMasterFlex(1*3-1,:);</pre>		
376		<pre>FMaster2Flex(1,:) = FMasterFlex(1*3,:);</pre>		
3//		4 - 4.4.		
3/8		1 = 1+1;		
380	_	enu		
381		fleyMatrix = [threadlengthAMatrixEley threadlengthRMatrixEley		
382		thread ength CosyMatrixFlex. thread ength CNeg/MatrixFlex. thread ength DMatrixFlex		
383		threadLengthEMatrixFlex, threadLengthFPosYMatrixFlex,threadLengthFNegYMatrixFlex,		
384		threadLengthFZMatrixFlex];		
385				
386			_	
387				
200				
388		masterPointApD = [xcoordsPositionMatrixAb(:,1),yCoordsPositionMatrixAb(:,1),zCoordsPositionMatrixAb(:,1)];		
389		masterPointBAD = [xCoordsPositionMatrixAD(:,2),yCoordsPositionMatrixAD(:,2),zCoordsPositionMatrixAD(:,2)];		
390		masterPointCAD = [xCoordsPositionMatrIXAD(:,5),yCoordsPositionMatrIXAD(:,5),zCoordsPositionMatrIXAD(:,5)];		
391		masterPointDAD = [xcoordsPositionMatrixAO(:,4),ycoordsPositionMatrixAD(:,4),zcoordsPositionMatrixAD(:,4)];		
392		masterPointEAD = [EXcoordsPositionMatriAd(:,5)], ycoordsPositionMatriAd(:,5)], 200rdsPositionMatriAd(:,5)]; masterPointEAD = [EMasterVAD(: 1) EmasterVAD(: 1)]		
204		masterrointrad = [rmasterrad(:,i),rmasterrad(:,i),rmasterrad(:,i)];		
395		masterPoint&Elev = [vCoordsPositionMatrivElev(+1) vCoordsPositionMatrivElev(+1) zCoordsPositionMatrivElev(+1)]		
396		masterPointRELay = [v(ondersitionMatrix[Eav(.)) v(onderSetionMatrix[Eav(.)) z(ondersitionMatrix[Eav(.)])		
397		masterPointClex = [xcordsPositionMatrixElex(:3), vCoordsPositionMatrixElex(:3), vCoordsPositionMatrixElex(:3))		
398		masterPointDElex = $[x \cap ondsPositionMatrixElex(.4), x \cap ondsPositionMatrixElex(.4), z \cap ondsPositionMatrixElex(.4)]$		
399		masterPointFlex = [xcordsPositionMatrixFlex(:5),vCoordsPositionMatrixFlex(:5),zCoordsPositionMatrixFlex(:5)]		
400		masterPointFFLex = [FMasterXFlex(:,1),FMasterYFlex(:,1),FMasterZFlex(:,1)]:		
401				
402			*	

📝 Edit	tor - C:\Users\aravr\OneDrive\Documents\MATLAB\map\rotation.m *	Θ×
+2	motortest2 m x allWirel enoth m x fullRotation m x humeralAbduction m x humeralElevion m x rotation m * x +	
	<pre>function [outputVector] = rotation(inputVector,thetaICR,thetaIJ.thetaIK.I.J.K.ICR)</pre>	
2 📮	%ROTATION Finds where a point will be after a rotation around all three	- 🗸
3 -	%axes	
4		
5	Mmoving orgin to I	
5	[Centereu1 = [v]v[v]]; $[Centereu1 = [v]v[v]];$ $[Centereu1 = [v]v[v]v[v]v[v]v[v]v[v]v[v]v[v]v[v]v[v]$	
8	ICenteredK = $[K(1) - I(1); K(2) - I(2); K(3) - I(3)];$	
9	<pre>ICenteredInputVector = [inputVector(1)-I(1);inputVector(2)-I(2);inputVector(3)-I(3)];</pre>	
10		
11	%Finding axis IJ and IK	
12	IJ = ICENTERED - ICENTEREDI; madI = sort((I)(1)(2)+(I)(2))(2+(I)(3))(2):	
14	normalIJ = IJ/magI);	
15		
16	IK = ICenteredI - ICenteredK;	
17	<pre>magIK = sqrt([IK(1))^2+(IK(2))^2+(IK(3))^2);</pre>	
10	normalik = ik/magik;	
20	%Rotation around IJ	
21	<pre>dotIJ = dot(ICenteredInputVector,normalIJ);</pre>	
22	<pre>crossIJ = cross(normalIJ,ICenteredInputVector);</pre>	
23		
24	(Tentered nutVector) = (1-Cos(thetal))*(crossl)).	
26	midVector1 = [[GenteredMidVector1(1)+I(1);[GenteredMidVector1(2)+I(2);[CenteredMidVector1(3)+I(3);];	
27		
28	%Rotation around JK	
29	doTIK = dot(ICenteredInputVector, normalIK);	
30	crossik = cross(normalik,icenteredinputvector);	
32	ICenteredMidVector2 = ((1-cos(thetaIK))*(dotIK)*(normalIK))+((cos(thetaIK))*	
33	(ICenteredInputVector))+((sin(thetaIK))*(crossIK));	
34	<pre>midVector2 = [ICenteredMidVector2(1)+I(1);ICenteredMidVector2(2)+I(2);ICenteredMidVector2(3)+I(3);];</pre>	
35	Maying Angin to TCP	
50		•
36	%Moving Orgin to ICR	
3/	LtRenteredJ = [J(1)-LtR(1);J(2)-LtR(2);J(3)-LtR(3)]; LtProtemedK = [V(1)-LtR(1):V(2)-LtR(2):V(3)-LtR(3)];	
39	$\frac{1}{10000000000000000000000000000000000$	100
40	<pre>ICRCenteredInputVector = [inputVector(1)-ICR(1);inputVector(2)-ICR(2);inputVector(3)-ICR(3)];</pre>	
41		
42	%Making ICR Axis	
43	ICRJ = ICRCenteredJ - ICRCenteredICR;	
44	ICAN = ICACENTEREOR - ICACENTEREOICA; avisICR = consc(ICR) ICRN).	
46	<pre>magICR = sqrt((axisICR(1))^2+(axisICR(2))^2+(axisICR(3))^2);</pre>	
47	normalICR = axisICR/magICR;	
48		
49	%Rotation around ICR	
50	<pre>dotLk = dot(ltklenteredInputVector,normalLk); crossIR = cross(normalIR)[R IFRentenedInputVector);</pre>	
52	crossient crossient mailen, ich center eainput vector /,	
53	<pre>ICRCenteredMidVector3 = ((1-cos(thetaICR))*(dotICR)*(normalICR))+((cos(thetaICR))</pre>	
54	*(ICRCenteredInputVector))+((sin(thetaICR))*(crossICR));	
55	<pre>midVector3 = [ICRCenteredMidVector3(1)+ICR(1);ICRCenteredMidVector3(2)+ICR(2);ICRCenteredMidVector3(3)+ICR(3)];</pre>	
56	% als the output	
58	outputVector = (midVector1+midVector2+midVector3)/3:	
59		
60 L	end	*

+2	abCode.m 🗶 motortest2.m 🗶 allWireLength.m 🗶 fullRotation.m 🗶 humeralAbduction.m 🗶 humeralFlexion.m * 🗶 🕂
1 📮	<pre>function [FPrime] = humeralFlexion(B,BPrime,F,armAngleChange)</pre>
2	%humeralFlexion finds the position of the bottom of the humerus
3	
4	BCenteredF = [F(1)-B(1);F(2)-B(2);F(3)-B(3)];
5	
6	normalAxisofRotation = [1;0;0];
7	
8	%Rotation around B
9	dotHumeral = dot(BCenteredF,normalAxisofRotation);
10	crossHumeral = cross(normalAxisofRotation,BCenteredF);
11	
12	BCenteredMidVector = ((1-cos(armAngleChange))*(dotHumeral)*(normalAxisofRotation))+
13	<pre>((cos(armAngleChange))*(BCenteredF))+((sin(armAngleChange))*(crossHumeral));</pre>
14	
15	
16	madvector = [0,11]; Buenteredmidvector(2)+B(2); Buenteredmidvector(3)+B(3)];
17	miavector(2) = [0.11; miavector(2) + (B(2)-bPrime(2)); miavector(3) + (B(3)-bPrime(3))];
10	singleConten = [0, 11, PDnime(2), PDnime(2)].
19	<pre>refrective = [0.11, brime(2), brime(3)]; neduc(0finel) = cont(sho((0.3633)(2), (0.80)));</pre>
20	Tadiusoferrere - sqrt(abs(((0.5003) 2)-((brrime(1)-0.11) 2)));
22	center2F = midVector2 - circleCenter:
22	machenter2F = sort(center)F(2)^2+center2F(3)^2):
24	normalCenter2F = center2F/magCenter2F:
25	<pre>EPrime = [0.11:circle(enter(2)+(radiusOfCircle*normalCenter2E(2)):circle(enter(3)+(radiusOfCircle*normalCenter2E(3))]:</pre>
26	
27 L	end

set	tup.m 🛪 abCode.m 🛪 motortest2.m 🛪 allWireLength.m 🛪 fullRotation.m 🛪 humeralAbduction.m * 🛪 🕇
1 -	<pre>function [FPrime] = humeralAbduction(B,BPrime,F,armAngleChange)</pre>
2	%humeralAbduction finds the position of the bottom of the humerus
3	
4	BCenteredF = $[F(1)-B(1);F(2)-B(2);F(3)-B(3)];$
5	
6	normalAxisofRotation = [0;-1;0];
7	
8	%Rotation around B
9	<pre>dotHumeral = dot(BCenteredF,normalAxisofRotation);</pre>
10	<pre>crossHumeral = cross(normalAxisofRotation,BCenteredF);</pre>
11	
12	BCenteredMidVector = ((1-cos(armAngleChange))*(dotHumeral)*(normalAxisofRotation))+
13	((cos(armAngleChange))*(BCenteredF))+((sin(armAngleChange))*(crossHumeral));
14	
15	
16	<pre>midVector = [BCenteredMidVector(1)+B(1);0.035;BCenteredMidVector(3)+B(3)];</pre>
17	midVector2 = [midVector(1) + (B(1)-BPrime(1)); 0.035; midVector(3) + (B(3)-BPrime(3))];
18	
19	circleCenter = [BPrime(1); 0.035; BPrime(3)];
20	radiusOfCircle = sqrt(((0.3683)^2)-((BPrime(2)-0.035)^2));
21	
22	center2F = midVector2 - circleCenter;
23	<pre>magCenter2F = sqrt(center2F(1)^2+center2F(3)^2):</pre>
24	normalCenter2F = center2F/magCenter2F;
25	FPrime = [circleCenter(1)+(radiusOfCircle*normalCenter2F(1)):0.035;circleCenter(3)+(radiusOfCircle*normalCenter2F(3))];
26	
27 L	end

se	etup.m 🗙 abCode.m 🗙 motortest2.m 🗙 allWireLength.m 🗶 fullRotation.m 🗶 🕂	
1 🗐	<pre>function [positionMatrix] = fullRotation(A,B,C,D,E,I,J,K,upwardRotation,anteriorTilt,externalRotation,ICR)</pre>	2
2	%FULLROTATION rotates all points around the three axes	٢.
3		
4	rotatedA = rotation (A,upwardRotation,anteriorTilt,externalRotation,I,J,K,ICR);	
5	rotatedB = rotation (B,upwardRotation,anteriorTilt,externalRotation,I,J,K,ICR);	
6	rotatedC = rotation (C,upwardRotation,anteriorTilt,externalRotation,I,J,K,ICR);	
7	rotatedD = rotation (D,upwardRotation,anteriorTilt,externalRotation,I,J,K,ICR);	
8	rotatedE = rotation (E,upwardRotation,anteriorTilt,externalRotation,I,J,K,ICR);	
9	rotatedK = rotation (K,upwardRotation,anteriorTilt,externalRotation,I,J,K,ICR);	
10	rotatedJ = rotation (],upwardRotation,anteriorTilt,externalRotation,I,J,K,ICR);	
11	rotatedI = rotation (I,upwardRotation,anteriorTilt,externalRotation,I,J,K,ICR);	
12		
13	positionMatrix = [rotatedA,rotatedB,rotatedL,rotatedL,rotatedL,rotatedL,rotatedL];	
14		
15 -	ena	

set	tup.m 🗶 abCode.m 🗶 motortest2.m 🗶 allWireLength.m 🗶 🕂
1 📮	function [wireLengths] = allWireLength(motorA,motorB,motorCPosY,motorCNegY,motorCX,motorD,motorF,motorFPosY,
2	motorFNegY,motorFZ,A,B,C,D,E,F)
3 🖨	%WIRELENGTH Finds the length of a wire given the motor and the attachment
4 -	%point
5	
6	threadLengthA = sqrt((motorA(1)-A(1))^2+(motorA(2)-A(2))^2+(motorA(3)-A(3))^2);
7	threadLengthB = sqrt((motorB(1)-B(1))^2+(motorB(2)-B(2))^2+(motorB(3)-B(3))^2);
8	<pre>threadLengthCPosY = sqrt((motorCPosY(1)-C(1))^2+(motorCPosY(2)-C(2))^2+(motorCPosY(3)-C(3))^2);</pre>
9	<pre>threadLengthCNegY = sqrt((motorCNegY(1)-C(1))^2+(motorCNegY(2)-C(2))^2+(motorCNegY(3)-C(3))^2);</pre>
10	threadLengthCX = sqrt((motorCX(1)-C(1))^2+(motorCX(2)-C(2))^2+(motorCX(3)-C(3))^2);
11	threadLengthD = sqrt((motorD(1)-D(1))^2+(motorD(2)-D(2))^2+(motorD(3)-D(3))^2);
12	threadLengthE = sqrt((motorE(1)-E(1))^2+(motorE(2)-E(2))^2+(motorE(3)-E(3))^2);
13	<pre>threadLengthFPosY = sqrt((motorFPosY(1)-F(1))^2+(motorFPosY(2)-F(2))^2+(motorFPosY(3)-F(3))^2);</pre>
14	<pre>threadLengthFNegY = sqrt((motorFNegY(1)-F(1))^2+(motorFNegY(2)-F(2))^2+(motorFNegY(3)-F(3))^2);</pre>
15	threadLengthFZ = sqrt((motorFZ(1)-F(1))^2+(motorFZ(2)-C(2))^2+(motorFZ(3)-F(3))^2);
16	
17	wireLengths = [threadLengthA, threadLengthB, threadLengthCPosY, threadLengthCNegY, threadLengthCX, threadLengthD, threadLengthE
18	threadLengthFPosY,threadLengthFNegY,threadLengthF2];
19 -	end

S	etu	ip.m 🛪 abCode.m * 🛪 motortest2.m 🛪 🕂		
1	7	<pre>function [currentAbIndex] = abCode(desiredIndex)</pre>		
2		global currentAhIndex:		
4		gibbai currentabindex,		
5		global currentFlexIndex		
6		global abMatrix;		
8		if currentFlexIndex > 1		
9		flexCode(1)		
10		end		
11		al = anduine('COM4', 'Une');		
13		a2 = arduino('COM3', 'Uno');		
14				
15		pinDirA = "D12"; minDirB = "D12";		
10		pinDirCPosY = "D8";		
18		pinDirCNegY = "D8";		
19		<pre>pinDirCX = "D13";</pre>		
20		pinDirD = "D12":		
22		pinDirE = "D4";		
23		<pre>pinDirFPosY = "D7";</pre>		
24		pinDirFNegY = "D7";		
25		pinbirriz = b4;		
27				
28				
29 30		pinMovA = "D10"; pinMovB = "D11";		
31		pinMovCPosY = "D9";		
32		<pre>pinMovCNegY = "D9";</pre>		
33		<pre>pinMovCX = "D11";</pre>		
34 35		pinMovD = "D10":		
36		pinMovE = "D5";		
		THER V HOCH		
	setu	up.m × abCode.m × motortest2.m × +		1
37	setu	<pre>up.m x abCode.m x motortest2.m x + pinMovFPosY = "D6";</pre>		
37 38	setu	<pre>up.m × abCode.m × motortest2.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6";</pre>	•	•
37 38 39	setu	<pre>up.m x abCode.m x motortest2.m x + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5";</pre>	^	•
37 38 39 40 41	setu	<pre>up.m X abCode.m X motortest2.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5";</pre>	^	•
37 38 39 40 41 42	setu	<pre>up.m x abCode.m x motortest2.m x + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput')</pre>	•	•
37 38 39 40 41 42 43	setu	<pre>up.m x abCode.m x motortest2.m x + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirB,'DigitalOutput')</pre>		•
37 38 39 40 41 42 43 44	setu	<pre>up.m x abCode.m x motortest2.m x + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirB,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput')</pre>		•
37 38 39 40 41 42 43 44 45 46	setu	<pre>up.m X abCode.m X motortest2.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput')</pre>		
37 38 39 40 41 42 43 44 45 46 47	setu	<pre>up.m X abCode.m X motortest2.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirB, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48	setu	<pre>up.m X abCode.m X motortest2.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a2,pinDirE, 'DigitalOutput') configurePin(a2,pinDirE, 'DigitalOutput') configurePin(a2,pinDirE,'DigitalOutput') configurePin(a2,pinDirE,'DigitalOutput') configurePin(a2,pinDirE,'DigitalOutput') configurePin(a2,pinDirE,'DigitalOutput')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 49 50	setu	<pre>up.m X abCode.m X motortest2.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirB,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a1,pinDirE,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a2,pinDirFZ,'DigitalOutput')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51	setu	<pre>up.m X abCode.m X motortest2.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirB,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a1,pinDirE,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFY,'DigitalOutput') configurePin(a1,pinDirFY,'DigitalOutput')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52	setu	<pre>up.m × abCode.m × motortest2.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirB,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFNegY,'DigitalOutput') configurePin(a1,pinDirFNegY,'DigitalOutput') configurePin(a1,pinDirFNegY,'DigitalOutput')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54	setu	<pre>up.m × abCode.m × motortest2.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirB, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a2,pinDirFPosY, 'DigitalOutput') configurePin(a1,pinDirFX, 'DigitalOutput') configurePin(a2,pinDirFPosY, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM')</pre>		•
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55	setu	<pre>up.m × abCode.m × motortest2.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirB, 'DigitalOutput') configurePin(a1,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a2,pinDirFPosY, 'DigitalOutput') configurePin(a1,pinDirFX, 'DigitalOutput') configurePin(a2,pinDirFNosY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovCPosY, 'PWM')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56	setu	<pre>up.m × abCode.m × motortest2.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirB, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovCNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 56 57 52	setu	<pre>up.m × abCode.m × motortest2.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirB, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinMovA, 'PWM') configurePin(a1,pinMovCNsY, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59	setu	<pre>up.m × abCode.m × motortest2.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNogY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirFY, 'DigitalOutput') configurePin(a2,pinDirFNogY, 'DigitalOutput') configurePin(a1,pinDirFNogY, 'DigitalOutput') configurePin(a1,pinDirFNogY, 'DigitalOutput') configurePin(a1,pinMovA, 'PWM') configurePin(a1,pinMovCN, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 60 59 60	setu	<pre>pinMovFPosY = "D6"; pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovC, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 950 51 52 53 54 55 56 57 58 59 60 61	setu	<pre>pinMovFPosY = "D6"; pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovC, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a2,pinMovFNegY, 'PWM')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 950 51 52 53 54 55 56 57 58 59 60 61 62 63	setu	<pre>pinMovFPosY = "D6"; pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovC, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovFosY, 'PWM') configurePin(a1,pinMovFosY, 'PWM') configurePin(a2,pinMovFNegY, 'PWM') configurePin(a2,pinMovFNegY, 'PWM') configurePin(a1,pinMovFosY, 'PWM') configurePin(a2,pinMovFNegY, 'PWM') configurePin(a1,pinMovFosY, 'PWM')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 950 51 52 53 54 55 56 57 58 59 60 61 62 63 64	setu	<pre>pinMovFPosY = "D6"; pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovC, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 950 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	setu	<pre>pinMovFPosY = "D6"; pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovC, 'PWM') configurePin(a1,pinMovCN, 'PWM') configurePin(a1,pinMovCN, 'PWM') configurePin(a1,pinMovCN, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM')</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 950 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 67	setu	<pre>pinMovFPosY = "D6"; pinMovFPosY = "D6"; pinMovFRegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovFosY, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovFosY, 'PWM') configurePin(a1,pinMovF, 'PWM') con</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 950 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68	setu	<pre>pinMovFPosY = "D6"; pinMovFPosY = "D6"; pinMovFRegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovCNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCN, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') config</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 950 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69	setu	<pre>pin % abCode.m % motortest2.m % + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFRegY = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a2,pinDirFS,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFX,'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovA,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovFX,'PWM') configurePin(a1,pinMovFNegY,'PWM') configurePin(a1,pinMovFNegY,'PWM') configurePin(a1,pinMovFX,'PWM') configurePin(a1,pin</pre>		
37 38 39 40 41 42 43 44 45 46 47 48 950 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70	setu	<pre>pin X abCode.m X motortest2.m X + pinMovFPosY = "D6"; pinMovFRogY = "D6"; pinMovFRogY = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCVegY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a2,pinDirFNegY,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFX,'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovA,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCX,'PWM') configurePin(a1,pinMovCX,'PWM') configurePin(a1,pinMovFNegY,'PWM') configurePin(a1,pinMovFNegY, pinDirFNegY, pinDirCN,pinDirCX]; directionPins1 = [pinDirFZ, pinDirCNegY, pinDirFNegY, pinDirFNegY, pinDirCX]; movementPins1 = [pinMovFZ, pinMovCNegY, pinMovFNegY, pinMovD, pinMovCX]; </pre>		
37 38 39 40 41 42 43 44 45 46 47 48 99 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72	setu	<pre>pin X abCode.m X motortest2.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCN,'DigitalOutput') configurePin(a1,pinDirCN,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirFNegY,'DigitalOutput') configurePin(a2,pinDirFNegY,'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovCRegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovFA,'PWM') configurePin(a1,pinMovFA,'PWM') configurePin(a1,pinMovFA,'PWM') configurePin(a1,pinMovFA,'PWM') configurePin(a1,pinMovFA,'PWM') configurePin(a1,pinMovFA,'PWM') configurePin(a1,pinMovFA,'PWM') configurePin(a1,pinMovFZ,'PWM') configurePin(a1,</pre>		

se	etup.m 🛪 abCode.m 🛪 motortest2.m 🛪 🕂			
75	i = 1;	▲ <u>A</u>		
76				
77	if currentAbIndex < desiredIndex			
78 🖨	while currentAbIndex < desiredIndex			
79				
80 🗆) while i<=5			
81	<pre>deltaLength = abs(abMatrix(currentAbIndex,i)-abMatrix(currentAbIndex</pre>	+1,i));		
82				
83	rotationDuration = deltaLength*116.886;			
84 95				
00	<pre>if admatrix(currentAdindex+i,1) > admatrix(currentAdindex,1)</pre>			
00 87	writeDigitalPin(a2_directionPins2(i)_1)			
88				
89	writeDigitalPin(al.directionPins1(i).0)			
90	writeDigitalPin(a2.directionPins2(i).0)			
91	end			
92				
93	<pre>writePWMDutyCycle(a1,movementPins1(i),0)</pre>			
94	<pre>writePWMDutyCycle(a2,movementPins2(i),0)</pre>			
95	minDutyCycle = 0.03;			
96	<pre>maxDutyCycle = 0.12;</pre>			
97	<pre>writePWMDutyCycle(a1,movementPins1(i),minDutyCycle)</pre>			
98	<pre>writePWMDutyCycle(a2,movementPins2(i),minDutyCycle)</pre>			
99				
100	startTime = tic;			
101				
102 -	while toc(startlime) < rotationDuration			
105	Adjust the duty cycle gradually to simulate Potation	tion		
104	* (maxDutyCycle - minDutyCycle + (coc(scarchime) / rocacionbura	(101)		
105	writePWMDutyCycle(a1, movementPins1(i), currentDutyCycle);			
107	writePWMDutyCycle(a2, movementPins2(i), currentDutyCycle):			
108				
109	% Add a small delay for smoother motion (adjust as needed)			
110	pause(0.01);	•		

```
abCode.m 🗙 motortest2.m
                                               +
    setup.m
111
                     end
                                                                                                                                 •
112
                                                                                                                                   113
                     % Stop the servo motor by setting the duty cycle to 0
114
                     writePWMDutyCycle(a1, movementPins1(i), 0);
                     writePWMDutyCycle(a2, movementPins2(i), 0);
115
116
                     i = i+1;
 117
                 end
                 currentAbIndex = currentAbIndex + 1;
118
119
                 i = 1;
 120
             end
 121
         else
 122 🗄
             while currentAbIndex > desiredIndex
123
 124 🗄
                 while i<=5
                     deltaLength = abs(abMatrix(currentAbIndex,i)-abMatrix(currentAbIndex-1,i));
125
126
127
128
                     rotationDuration = deltaLength*116.886;
 129
 130
 131
                     if abMatrix(currentAbIndex-1,i) > abMatrix(currentAbIndex,i)
132
                        writeDigitalPin(a1,directionPins1(i),1)
                         writeDigitalPin(a2, directionPins2(i),1)
133
134
                     else
                         writeDigitalPin(a1,directionPins1(i),0)
135
 136
                         writeDigitalPin(a2,directionPins2(i),0)
137
                     end
 138
 139
                     writePWMDutyCycle(a1,movementPins1(i),0)
 140
                     writePWMDutyCycle(a2,movementPins2(i),0)
                     minDutyCycle = 0.03;
141
 142
                     maxDutvCvcle = 0.12;
                     writePWMDutyCycle(a1,movementPins1(i),minDutyCycle)
 143
144
                     writePWMDutyCycle(a2,movementPins2(i),minDutyCycle)
 145
146
                     147
                     startTime = tic;
148
149 🗄
                     while toc(startTime) < rotationDuration</pre>
150
                         % Adjust the duty cycle gradually to simulate rotation
                         currentDutyCycle = minDutyCycle + (toc(startTime) / rotationDuration)...
151
                             * (maxDutyCycle - minDutyCycle);
152
                         writePWMDutyCycle(a1, movementPins1(i), currentDutyCycle);
153
154
                         writePWMDutyCycle(a2, movementPins2(i), currentDutyCycle);
155
 156
                         % Add a small delay for smoother motion (adjust as needed)
157
                        pause(0.01);
                     end
158
159
160
                     % Stop the servo motor by setting the duty cycle to 0
161
                     writePWMDutyCycle(a1, movementPins1(i), 0);
                     writePWMDutyCycle(a2, movementPins2(i), 0);
162
 163
                     i = i+1;
164
                 end
165
                 currentAbIndex = currentAbIndex - 1;
166
                 i = 1;
167
            end
         end
168
         clear al
169
170
         clear a2
171
         end
```

	etun m 💥 motortest? m 💥 flexCode m 💥 abCode m 💥 🕂	
1 S	function [currentFlexIndex] = flex(ode(desiredIndex))	
2	- reneered feathener revenuev1 - i revenue(nesti entinev)	Â
3	global currentFlexIndex;	=
4		
5	global currentFlexIndex	
6	global flexMatrix;	
	if currentElexIndex > 1	
9	flexCode(1)	
10	end	
11		
12	a1 = arduino('COM4', 'Uno');	
13	a2 = arduino('COM3', 'Uno');	
14	ninDinA - "D12",	
15	pinDirA = 012; pinDirB = "D13":	
17	pinDirCPosY = "D8";	
18	<pre>pinDirCNegY = "D8";</pre>	
19	<pre>pinDirCX = "D13";</pre>	
20		
21	pinuru = "D12"; pinDirE = "D4":	
22	pinDirFPosY = "D7":	
24	pinDirFNegY = "D7";	
25	<pre>pinDirFZ = "D4";</pre>	
26		
27		
28	ninMovA = "D10"	
30	pinMovA = Did ; pinMovB = "D11":	
31	pinMovCPosY = "D9";	
32	<pre>pinMovCNegY = "D9";</pre>	
33	<pre>pinMovCX = "D11";</pre>	
34		
35	pinMovD = "D10"; pinMovE = "D5";	
50	-3-MFDV "DC".	•
se	etup.m 🗙 motortest2.m 🛪 flexCode.m 🗶 abCode.m 🗶 🕇	
37	etup.m × motortest2.m × flexCode.m × abCode.m × + pinMovFPosY = "D6";	^
37 38	etup.m × motortest2.m × flexCode.m × abCode.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6";	^ <u>A</u>
37 38 39	etup.m × motortest2.m × flexCode.m × abCode.m × + pinMovFPosY = "D6"; pinMovFX = "D6"; pinMovFZ = "D5";	
37 38 39 40 41	etup.m × motortest2.m × flexCode.m × abCode.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5";	
37 38 39 40 41 42	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput')</pre>	
37 38 39 40 41 42 43	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirB,'DigitalOutput')</pre>	
37 38 39 40 41 42 43 44	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirB,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput')</pre>	
37 38 39 40 41 42 43 44 45	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirB,'DigitalOutput') configurePin(a1,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configu</pre>	
see 37 38 39 40 41 42 43 44 45 46 47 47	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCN_Y) configurePin(a1,pinDirCN_Y) configurePin(a1,pinDirC)</pre>	
se 37 38 39 40 41 42 43 44 45 46 47 48	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFNegY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a2,pinDirE, 'DigitalOutput') configurePin(a2,pinDirE, 'DigitalOutput')</pre>	
se 37 38 39 40 41 42 43 44 45 46 47 48 49	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFNegY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput')</pre>	
se 37 38 39 40 41 42 43 44 45 46 47 48 49 50	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFNegY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirFZ,'DigitalOutput') configurePin(a2,pinDirFZ,'DigitalOutput') configurePin(a2,pinDirFZ,'DigitalOutput') configurePin(a2,pinDirFZ,'DigitalOutput') configurePin(a2,pinDirFZ,'DigitalOutput')</pre>	
se 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 51	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFNegY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a1,pinDirF,'DigitalOutput') configurePin(a1,pinDirF,'DigitalOutput') configurePin(a1,pinDirF,'DigitalOutput') configurePin(a1,pinDirF,'DigitalOutput') configurePin(a1,pinDirF,'DigitalOutput') configurePin(a1,pinDirF,'DigitalOutput') configurePin(a1,pinDirF,'DigitalOutput') configurePin(a1,pinDirFNegY,'DigitalOutput')</pre>	
see 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFNegY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirB,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a1,pinDirF,'DigitalOutput') configurePin(a1,pinDirFNegY,'DigitalOutput') configurePin(a1,pinDirFNegY,'DigitalOutput') configurePin(a1,pinDirFNegY,'DigitalOutput') configurePin(a1,pinDirFNegY,'DigitalOutput')</pre>	
See 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 53	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirFZ, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirFZ, 'DigitalOutput') configurePin(a2,pinDirFZ, 'DigitalOutput') configurePin(a2,pinDirFZ, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM')</pre>	
see 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirFZ, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirFX, 'DigitalOutput') configurePin(a2,pinDirFX, 'DigitalOutput') configurePin(a2,pinDirFX, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM')</pre>	
37 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFNegY = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirF2, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF2, 'DigitalOutput') configurePin(a2,pinDirF2, 'DigitalOutput') configurePin(a2,pinDirF2, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovK,'PWM') configurePin(a2,pinMovCNegY,'PWM')</pre>	
Sec 37 38 38 39 40 41 42 43 44 45 46 47 48 99 50 551 52 53 54 55 56 56 57 77	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCN,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirFNegY,'DigitalOutput') configurePin(a2,pinDirFNegY,'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovCNegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') </pre>	
see 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCN'DigitalOutput') configurePin(a1,pinDirCN'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirFNegY,'DigitalOutput') configurePin(a2,pinDirFNegY,'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovCNegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCN,'PWM') configurePin(a1,pinMovCN,'PWM')</pre>	
see 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCK,'DigitalOutput') configurePin(a1,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirFY,'DigitalOutput') configurePin(a2,pinDirFY,'DigitalOutput') configurePin(a2,pinDirFNegY,'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovCNegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCN,'PWM') configurePin(a1,pinMovCY,'PWM') configurePin(a1,pinMovCY,'PWM') configurePin(a1,pinMovCY,'PWM') configurePin(a1,pinMovCY,'PWM')</pre>	
see 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirB,'DigitalOutput') configurePin(a1,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirFY,'DigitalOutput') configurePin(a2,pinDirFPosY,'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovCNeyY,'PWM') configurePin(a1,pinMovCNeyY,'PWM') configurePin(a1,pinMovCNeyY,'PWM') configurePin(a1,pinMovCNeyY,'PWM') configurePin(a1,pinMovCPosY,'PWM') config</pre>	
see 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFNegY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirFPosY,'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovCPosY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCFosY,'PWM') configurePin(a1,pinMovFPosY,'PWM') con</pre>	
See 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFNegY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCPosY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirC,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirFY,'DigitalOutput') configurePin(a2,pinDirFY,'DigitalOutput') configurePin(a2,pinDirFY,'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovCPosY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovCNegY,'PWM') configurePin(a1,pinMovFPosY,'PWM') configurePin(a2,pinMovF,'PWM') configurePin(a2,pinMovFY,'PWM') configurePin(a2,pinMovFY,'PWM') configurePin(a2,pinMovFY,'PWM') configurePin(a1,pinMovFPosY,'PWM') configurePin(a2,pinMovFY,'PWM') configurePin(a1,pinMovFY,'PWM') configurePin(a1,pinMovFY,'PWM'</pre>	
Sec 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFNegY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirFX, 'DigitalOutput') configurePin(a2,pinDirFX, 'DigitalOutput') configurePin(a2,pinDirFX, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovK, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM')</pre>	
Sec 37 38 39 40 41 42 43 44 45 46 47 88 99 50 51 52 53 54 55 56 57 58 59 60 61 62 63	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFNegY = "D6"; pinMovFNegY = "D6"; pinMovFXegY = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a2,pinDirFZ, 'DigitalOutput') configurePin(a2,pinDirFZ, 'DigitalOutput') configurePin(a2,pinDirFX, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovK, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovE, 'PWM') configurePin(a1,pinMovF, 'PWM')</pre>	
Sec 37 38 39 40 41 42 43 44 45 46 47 89 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66	<pre>etup.m X motortest2.m X flexCode.m X abCode.m X + pinMovFNegY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirB, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a2,pinDirFZ, 'DigitalOutput') configurePin(a2,pinDirFZ, 'DigitalOutput') configurePin(a2,pinDirFX, 'DigitalOutput') configurePin(a2,pinDirFX, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovCNegY, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovE, 'PWM') configurePin(a2,pinMovFNegY, 'PWM') configurePin(a1,pinMovFNegY, 'PWM') configurePin(a1</pre>	
second 37 38 39 40 41 42 43 44 45 46 47 48 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67	<pre>etup.m × motortest2.m × flexCode.m × abCode.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCN, 'DigitalOutput') configurePin(a1,pinDirCN, 'DigitalOutput') configurePin(a1,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovCNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a2,pinMovFNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovFNegY, 'PWM</pre>	
see 37 38 39 40 41 42 43 44 45 46 57 58 59 60 61 62 63 64 65 66 67 68	<pre>etup.m × motortest2m × flexCode.m × abCode.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCN, 'DigitalOutput') configurePin(a1,pinDirC, 'DigitalOutput') configurePin(a1,pinDirFZ, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinDirFNegY, 'DigitalOutput') configurePin(a1,pinDirFNegY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovA, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovFZ, 'PWM') configurePin(a1,pinMovFZ, 'PWM') configurePin(a1,pinMovFZ, 'PWM') configurePin(a1,pinMovFZ, 'PWM') configurePin(a1,pinMovFZ, 'PWM') configurePin(a1,pinMovFZ, 'PWM') configurePin(a2,pinMovFZ, 'PWM') configurePin(a2,pinMovFZ, 'PWM') configurePin(a1,pinMovFZ, 'PWM') configurePin(a2,pinMovFZ, 'PWM') configurePin(a1,pinMovFZ, 'PWM') configurePin(a1,pinMovFZ, 'PWM') configurePin(a2,pinMovFEZ, 'PWM') configurePin(a1,pinMovFZ, 'PWM') configurePi</pre>	
see 37 38 39 40 41 42 43 44 45 46 57 58 59 60 61 62 63 64 65 66 67 68 69 7	<pre>stup.m × motortest2.m × flexCode.m × abCode.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA,'DigitalOutput') configurePin(a2,pinDirCNegY,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a1,pinDirCX,'DigitalOutput') configurePin(a2,pinDirF,'DigitalOutput') configurePin(a2,pinDirFPosY,'DigitalOutput') configurePin(a2,pinDirFPosY,'DigitalOutput') configurePin(a2,pinDirFNegY,'DigitalOutput') configurePin(a2,pinDirFNegY,'DigitalOutput') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovA,'PWM') configurePin(a2,pinMovCy,'PWM') configurePin(a1,pinMovC,'PWM') configurePin(a1,pinMovC,'PWM') configurePin(a1,pinMovC,'PWM') configurePin(a1,pinMovC,'PWM') configurePin(a1,pinMovFz,'PWM') configurePin(BYMOYB'A,pinDirFNegY,pinDirFNeg</pre>	
see 37 38 39 40 41 42 43 44 45 46 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71	<pre>stup.m × motortest2.m × flexCode.m × abCode.m × + pinMovFPosY = "D6"; pinMovFNegY = "D6"; pinMovFZ = "D5"; configurePin(a2,pinDirA, 'DigitalOutput') configurePin(a2,pinDirCPosY, 'DigitalOutput') configurePin(a1,pinDirCNegY, 'DigitalOutput') configurePin(a1,pinDirCX, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirF, 'DigitalOutput') configurePin(a2,pinDirFPosY, 'DigitalOutput') configurePin(a2,pinDirFPosY, 'DigitalOutput') configurePin(a2,pinMovA, 'PWM') configurePin(a2,pinMovK, 'PWM') configurePin(a1,pinMovCNegY, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovCX, 'PWM') configurePin(a1,pinMovFosY, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovFosY, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') configurePin(a1,pinMovF, 'PWM') configurePin(a2,pinMovF, 'PWM') con</pre>	

75	i = 1;	-	A
76			
77	if currentFlexIndex < desiredIndex		-
78 🖨	while currentFlexIndex < desiredIndex		
79			
80 Ė	while i<=5		
81	<pre>deltaLength = abs(flexMatrix(currentFlexIndex,i)-flexMatrix(currentFlexIndex+1,i));</pre>		
82			
83	rotationDuration = deltaLength*116.886;		
84			
85	<pre>if Trekmatrix(currentFlexindex+1,1) > Trekmatrix(currentFlexindex,1) uniteDicitedDicitedDiction</pre>		
00 87	writeDigitalFin(a)_directionPins(1)_j) writeDigitalFin(a)_directionPins(2)_j 1)		
88			
89	writeDigitalPin(a1.directionPins1(i).0)		
90	writeDigitalPin(a2,directionPins2(i),0)	1.00	
91	end		
92			
93	<pre>writePWMDutyCycle(a1,movementPins1(i),0)</pre>		
94	<pre>writePWMDutyCycle(a2,movementPins2(i),0)</pre>		
95	minDutyCycle = 0.03;		
96	maxDutyCycle = 0.12;		
97	<pre>writePWMDutyCycle(a1,movementPins1(i),minDutyCycle)</pre>		
98	writePWMDutyCycle(a2,movementPins2(1),minDutyCycle)		
99			
100	startlime = tic;		
101	while tor(startTime) < rotationDuration		
102	* Adjust the diverse of a simulate rotation		
104	currentDutyCycle = minDutyCycle + (toc(startTime) / rotationDuration)		
105	<pre>* (maxDutyCycle - minDutyCycle);</pre>		
106	<pre>writePWMDutyCycle(a1, movementPins1(i), currentDutyCycle);</pre>		
107	<pre>writePWMDutyCycle(a2, movementPins2(i), currentDutyCycle);</pre>		
108			
109	% Add a small delay for smoother motion (adjust as needed)		
110	pause(0.01);	-	
se	tup.m 🗙 motortest2.m 🗶 flexCode.m 🗶 abCode.m 🗶 🕇		
111 -	end		Δ
112			
113	% Stop the servo motor by setting the duty cycle to 0		
114	<pre>writePWMDutyCycle(a1, movementPins1(i), 0);</pre>		
115	<pre>writePWMDutyCycle(a2, movementPins2(i), 0);</pre>		
116	i = i+1;		
117 -	end		
118	currentFlexIndex = currentFlexIndex + 1;		
119	1 = 1;		
120 -	ena		
121	while supportFloyIndex > designedIndex		
122	WITTE CUTERCITEXTINEX / GESTERLINEX		
124	while i<=5		
125	<pre>deltaLength = abs(flexMatrix(currentFlexIndex.i)-flexMatrix(currentFlexIndex-1.i));</pre>		
126			
127			
128	rotationDuration = deltaLength*116.886;		
128 129	<pre>rotationDuration = deltaLength*116.886;</pre>		
128 129 130	rotationDuration = deltaLength*116.886;		
128 129 130 131	<pre>rotationDuration = deltaLength*116.886; if flexMatrix(currentFlexIndex-1,i) > flexMatrix(currentFlexIndex,i)</pre>		

```
if flexMatrix(currentFlexIndex-1,i) > flexMatrix(currentFlexIndex,i)
    writeDigitalPin(a1,directionPins1(i),1)
    writeDigitalPin(a2, directionPins2(i),1)
else
    writeDigitalPin(a1,directionPins1(i),0)
    writeDigitalPin(a2,directionPins2(i),0)
end
writePWMDutyCycle(a1,movementPins1(i),0)
writePMMDutyCycle(a2,movementPins2(i),0)
```

minDutyCycle = 0.03; maxDutyCycle = 0.12; writePWMDutyCycle(a1,movementPins1(i),minDutyCycle) writePWMDutyCycle(a2,movementPins2(i),minDutyCycle)

Ŧ

	100		
147		<pre>startTime = tic;</pre>	
148			
149 E		while toc(startTime) < rotationDuration	
150		% Adjust the duty cycle gradually to simulate rotation	
151		<pre>currentDutyCycle = minDutyCycle + (toc(startTime) / rotationDuration)</pre>	
152		<pre>* (maxDutyCycle - minDutyCycle);</pre>	
153		<pre>writePWMDutyCycle(a1, movementPins1(i), currentDutyCycle);</pre>	
154		<pre>writePWMDutyCycle(a2, movementPins2(i), currentDutyCycle);</pre>	
155			
156		% Add a small delay for smoother motion (adjust as needed)	
157		pause(0.01);	
158	-	end	
159			
160		% Stop the servo motor by setting the duty cycle to 0	
161		<pre>writePWMDutyCycle(a1, movementPins1(i), 0);</pre>	
162		<pre>writePWMDutyCycle(a2, movementPins2(i), 0);</pre>	
163		i = i+1;	
164	-	end	
165		currentFlexIndex = currentFlexIndex - 1;	
166		i = 1;	
167	-	end	
168		end	
169		clear al	
170		clear a2	
171	-	end 🔹	

1 -	<pre>classdef shoulderRotationApp < matlab.apps.AppBase</pre>		^
2 4 - 5	% Properties that correspond to app components properties (Access = public) UFigure matlab.vi.Figure		5
6	ShowWiresCheckBox matlab.ui.control.CheckBox		
7	HumeralFlexionSlider matlab.ui.control.Slider		
8	HumeralFlexionSliderLabel matlab.ui.control.Label		
9	HumeralAbductionSlider matlab.ui.control.Slider		
10	HumeralAbductionSilderLabel matlab.ul.control.Label		
12	UTAxesSide matlab.ul.control.UTAxes		
13	UIAxesFront matlab.ui.control.UIAxes		
14	UIAxesAngled matlab.ui.control.UIAxes		
15 -	end		
16			
17	% Callbacks that handle component events		
18 -	methods (Access = private)		
19	W forder which any state officer and state of the		
20	S code that executes after component creation		
22	setup		
23	global xCoordsPositionMatrixAb		
24	global yCoordsPositionMatrixAb		
25	global zCoordsPositionMatrixAb		
26	global FMasterXAb		
27	global FMasterYAb		
28	global FMasterZAb		
30	nlot3(ann UTàvasànglad [vCoordePositionNatriváh(1 1) vCoordeP	ne HighNatriyab/1 2) vCoordsDasiHighNatriyab/1 5) vCoordsDasiHighNatriyab/1 3) vCoordsDasiHighNatriyab/1 4) vCoordsDasiHighNatriyab/1 1)]	
31	[vCoordsPositionMatrixéh(1.1) vCoordsPositionMatrixéh(1.2) vC	<pre>control (i) / condeparticular (i) / con</pre>	
32	[zCoordsPositionHatrixAb(1,1) zCoordsPositionHatrixAb(1,2) zC	oordsPositionNatrixAb(1,5) _CoordsPositionNatrixAb(1,3) _CoordsPositionNatrixAb(1,4) _CoordsPositionNatrixAb(1,1)], '.k-',	
33	<pre>[xCoordsPositionMatrixAb(5,2) FMasterXAb(1)], [yCoordsPosition</pre>	nMåtrixAb(5,2) FMasterYAb(1)], [zCoordsPositionNatrixAb(5,2) FMasterZAb(1)], '.k-');	
34			
35			
36	plot3(app.UlAxesFront,[xCoordsPositionMatrixAb(1,1) xCoordsPo	SationNatrixAb(1,2) XCoordsPositionNatrixAb(1,5) XCoordsPositionNatrixAb(1,3) XCoordsPositionNatrixAb(1,4) XCoordsPositionNatrixAb(1,1)],	
3.9	[ycoordsPositionmatrixad(1,1) ycoordsPositionMatrixAd(1,2) yc	oomparaiidannii taxaa(1,2) yyyteenaariikaa(1,2) yyteenaariikaa(1,4) yyteenaariikaa	
39	[xCoordsPositionMatrixAb(5,2) EMasterXAb(1)], [xCoordsPositionMatrixAb(5,2)]	<pre>doubsetsitionmati table(1,5) tool setsitionmati table(1,5) tool setsitionmati table(1,4) tool setsitionmati table(1,4);,, minimati table(1,1);,, minimati table(1,4);,, minimati table(1,4);,,,,,,,</pre>	
40	[
41	app.UIAxesFront.CameraPosition = [0.05,1,-0.075];		
42	app.UIAxesFront.CameraTarget = [0.05,0,-0.075];		
43			*

<pre>4 plot(sputkestike_(constraintionart.ukk(1,2) (sceedsmit(spatr.ukk(1,2) (sceedsmit(spatr.ukk(1,3) (sceedsmit(spatr.ukk(1,4) (sceedsmit(spatr.uk</pre>			
<pre>/// for any interval (1) (coordstationarisk(1)) (coordstationar</pre>	44	plot3(app.UIAxesSide,[xCoordsPositionMatrixAb(1,1) xCoordsPositionMatrixAb(1,2) xCoordsPositionMatrixAb(1,5) xCoordsPositionMatrixAb(1,3) xCoordsPositionMatrixAb(1,4) xCoordsPositionMatrixAb(1,1)],	▲
<pre>[[[conditionationationationationationationation</pre>	45	[yCoordsPositionMatrixAb(1,1) yCoordsPositionMatrixAb(1,2) yCoordsPositionMatrixAb(1,5) yCoordsPositionMatrixAb(1,3) yCoordsPositionMatrixAb(1,4) yCoordsPositionMatrixAb(1,1)],	
<pre>/// // // // // // // // // // // // //</pre>	46	[zCoordsPositionMatrixAb(1,1) zCoordsPositionMatrixAb(1,2) zCoordsPositionMatrixAb(1,5) zCoordsPositionMatrixAb(1,3) zCoordsPositionMatrixAb(1,4) zCoordsPositionMatrixAb(1,1)],'.k-',	=
<pre>sp.Ukaetids.ComersPosition - [-1,6,-4,073]; sp.Ukaetids.ComersPosition - [-1,6,-4,07]; sp.Ukaetids.comersPosition - [-1,6,-4,07];</pre>	47	[xCoordsPositionHatrixAb(5,2) FMasterXAb(1)], [yCoordsPositionHatrixAb(5,2) FMasterYAb(1)], [zCoordsPositionHatrixAb(5,2) FMasterZAb(1)], '.k-');	
app.UAxeside.Cameraborition = (-1,0,-0.475); app.UAxeside.Cameraboriti.camerab	48		
<pre>sep_UtkerSide.cmmr3rgt = [0,0,0.075]; plot(gep_UtkerSide.cmmr3rgt): CoordPositionMarrLub(1,1) xCoordPositionMarrLub(1,5) xCoordPositionMarrLub(1,5) xCoordPositionMarrLub(1,5), xCoordPositionMar</pre>	49	app.UIAxesSide.CameraPosition = [-1,0,-0.075];	
<pre>plot(gp.UixesTop.[scoordsbot(ionNatrixb(1,1) xcoordsbot(ionNatrixb(1,2) xcoordsbot(ionNatrixb(1,3) xcoordsbot(ionNatrixb(1,3</pre>	50	app.UIAxesSide.CameraTarget = [0,0,-0.075];	_
<pre>plation=line=line=line=line=line=line=line=lin</pre>	51		
<pre> i</pre>	52	plot3(app.UIAxesTop.[xCoordsPositionMatrixAb(1,1) xCoordsPositionMatrixAb(1,2) xCoordsPositionMatrixAb(1,5) xCoordsPositionMatrixAb(1,3) xCoordsPositionMatrixAb(1,4) xCoordsPositionMatrixAb(1,1)],	
54 [Icoordsbilledurtixb2(1,1) Icoordsbilledurtixb2(1,2) Icoordsbilledurtixb2	53	[yCoordsPositionNatrixAb(1,1) yCoordsPositionMatrixAb(1,2) yCoordsPositionMatrixAb(1,5) yCoordsPositionMatrixAb(1,3) yCoordsPositionMatrixAb(1,4) yCoordsPositionMatrixAb(1,1)],	
55 [xcordsBosilioNutrixAb(5,2) PMasterVAb(1)], [xcordsPosilioNutrixAb(5,2) PMasterVAb(1)], ',k-'); 57 app.UDAxesTop.CameraDayTel. 58 app.UDAxesTop.CameraDayTel. 59 glabal correntAbledes 61 glabal correntAbledes 62 if currentAbledes 63 if currentAbledes 64 abcdot(1) 65 iff currentAbledes 66 iff currentAbledes 67 abcdot(1) 68 abcdot(2) 69 rend 70 fileSocie(1) 61 abcdot(2) 71 K Value changed function: HomeslAbductionSilder 720 fonction HomeslAbductionSilder 73 glabal AccordsbosticonNatriab 74 glabal MusterValueChanging(ap, event) 75 glabal MusterValueChanging(ap, event) 76 glabal MusterValueChanging(ap, event) 77 glabal MusterValueChanging(ap, event) 78 glabal MusterValueChanging(ap, event) 79 glabal MusterValueChanging(ap, event) 70 glabal MusterValueChanging(ap, event) 76	54	[zCoordsPositionMatrixAb(1,1) zCoordsPositionMatrixAb(1,2) zCoordsPositionMatrixAb(1,5) zCoordsPositionMatrixAb(1,3) zCoordsPositionMatrixAb(1,4) zCoordsPositionMatrixAb(1,1)],'.k-',	_
<pre>split.strip.clameraPosition = [0.05,0.2]; split.strip.clameraPosition = [0.05,0.2]; split.strip.clameraPosition = [0.05,0.0]; split.strip.s</pre>	55	<pre>[xCoordsPositionHatrixAb(5,2) FMasterXAb(1)], [yCoordsPositionHatrixAb(5,2) FMasterYAb(1)], [zCoordsPositionHatrixAb(5,2) FMasterZAb(1)], '.k-');</pre>	
image: state of the state	56		=
<pre>sep_UlkesTop_LamerTargt = [0.05,0,0]; global_currentLisiondes global_currentLisiondes global_currentLisiondes if currentLisiondes > 1</pre>	57	app.UIAxesTop.CameraPosition = [0.05,0,2];	
<pre>global currentbindes global materpointAb global</pre>	58	app.UIAxesTop.CameraTarget = [0.05,0,0];	
00 global currentblender 11 global currentblender 12 global currentblender > 1 13 global currentblender > 1 14 abcKe(1) 15 elstif currentblender > 1 15 end 15 forstin humenalablactionilider 17 forstin humenalablactionilider 17 global vicondostiliderktinab 18 global vicondostiliderktinab 17 global vicondostiliderktinab 18 global vicondostiliderktinab 19 global vicondostiliderktinab 19 global vicondostiliderktinab 19 global vicondostiliderktinab 19 global vicondostiliderktinab 10 global vicondostiliderktinab 11 global vicondostiliderktinab 12 global vicondostiliderktinab 13 global materbineBo 14 global materbineBo 15 global materbineBo 16 global materbineBo 16 global materbineBo 16 global materbineBo 16 global materbin	59		
1 glabal currentizindes: 4	60	global currentAbIndex	
<pre>if current&lindex > 1</pre>	61	global currentFlexIndex	
1 if currentWillndex > 1 2 elsif currentVielndex > 1 2 end 2 end 2 function NumeralAdductionSlider 7 global xcondosticonterisab 8 global xcondosticonterisab 9 global xcondosti	62		
44 abcds(1) 54 abcds(1) 65 abcds(1) 66 abcds(1) 67 abcds(1) 68 abcds(1) 70 end 71 % Value changed function: HumeralAbductionSlider 72 faction HumeralAbductionSlider 73 glabal xcorddositionWariab 74 glabal xcorddositionWariab 75 glabal XcorddositionWariab 76 glabal HisterAb 77 glabal HisterAb 78 glabal MisterAib 79 glabal misterAiba 80 glabal misterAiba 81 glabal misterAiba 82 glabal misterAiba 83 glabal misterAiba 84 glabal misterAiba 85 glabal misterAiba 86 glabal misterAiba 87 glabal misterAiba 88 glabal misterAiba 89 glabal misterAiba 80 glabal misterAiba 81 glabal misterAiba 82 glabal misterAiba 83 glabal misterAiba 84 glabal misterAiba 85 glabal misterAiba	63	if currentAbIndex > 1	
65 elsif current/sinkex > 1 66 ed 78 ed 79 fibiolity 74 fibiolity 75 fibiolity 76 fibiolity 77 fibiolity 78 fibiolity 79 fibiolity 74 fibiolity 75 fibiolity 76 fibiolity 77 fibiolity 78 fibiolity 79 fibiolity 70 fibiolity 71 fibiolity 72 fibiolity 73 fibiolity 74 fibiolity 75 fibiolity 76 fibiolity 77 fibiolity 78 fibiolity 79 fibiolity 70 fibiolity 71 fibiolity 72 fibiolity 73 fibiolity 74 fibiolity 75 fibiolity fibiolity fibio	64	abCode(1)	
66 InterCode(1) 67 end 78 end 79 end 71 % Value changed function: HumeralAdoutionSlider 72 function HumeralAdoutionSlider 73 global koorddoutionstriade 74 global koorddoutionstriade 75 global functorNameralAdoutionSlider 76 global functorNameralAdoutionSlider 77 global functorNameralAdoutionSlider 78 global functorNameralAdoutionSlider 79 global functorNameralAdoutionSlider 70 global functorNameralAdoutionSlider 71 global functorNameralAdoutionSlider 72 global matterOde 73 global matterOde 74 global matterOde 75 global matterOde 76 global matterOde 77 global matterOde 78 global matterOde 79 global matterOde 70 global matterOde 71 global matterOde 72 global matterOde 73 global matterOde 74 global matterOde 75 global matterOde 76 global matterOde 77 global matterOde <th>65</th> <th>elseif currentFlexIndex > 1</th> <th></th>	65	elseif currentFlexIndex > 1	
27 end 68 end 71 K Value changed function: HumeralAbbuctionSlider 72 function: HumeralAbbuctionSlider 73 global viccordstotisticaturisab 74 global viccordstotisticaturisab 75 global Finistervab 76 global Finistervab 77 global Finistervab 78 global Finistervab 79 global Finistervab 81 global materplintAb 82 global materplintAb 83 global materplintAb 84 global materplintAb 85 global materplintAb	66	flexCode(1)	
and 1 % Value changed function: HumeralAbductionSlider 11 % Value changed function: HumeralAbductionSlider 12 function HumeralAbductionSlider 13 global xCoord/dositLooHtrixab 14 global xCoord/dositLooHtrixab 15 global xCoord/dositLooHtrixab 16 global instrefinitAb 17 global mater/bintAb 18 global mater/bintAb 19 global mater/bintAb 10 global mater/bintAb 11 global mater/bintAb 12 global mater/bintAb 13 global mater/bintAb 14 global mater/bintAb 15 global mater/bintAb 16 global mater/bintAb 17 global mater/bintAb	67	end	
edd Value changed function: HumeralAbbuctionSlider To S Value changed function: HumeralAbbuctionSlider Lobal AccordentionMetriab diabal YourdensitionMetriab diabal YourdensitionMetriab diabal FiniterAb diabal FiniterAb diabal FiniterAb diabal FiniterAb diabal FiniterAb diabal FiniterAb diabal FiniterAb diabal materPointAb diabal materPointAb diab	68		
70 K Value changed function: HumeralAbductionSlider 71 function: HumeralAbductionSlider 72 function: HumeralAbductionSlider 73 glabal, sconedFostichuntriade 74 glabal, sconedFostichuntriade 75 glabal, sconedFostichuntriade 76 glabal, functorAb 77 glabal, functorAb 78 glabal, materPointAb 81 glabal, materPointAb 82 glabal, materPointAb 83 glabal, materPointAb 84 glabal, materPointAb 85 glabal, materPointAb	69	end	
1 % Value changed function; MeeralabouctionSider 2 function MueralabouctionSider(spr.event) 3 fileBal xfoordfootsitonWtriab 4 fileBal xfoordfootsitonWtriab 7 fileBal xfoordfootsitonWtriab 7 fileBal xfoordfootsitonWtriab 8 fileBal FileBartMal 9 fileBal Mitter/Bal 8 fileBal mitter/Bal	70		
Image: Control Material Abdaction Material Abda	71	% Value changed function: HumeralAbductionSlider	
24 global According triabab 25 global According triabab 26 global FinaterAdb 27 global FinaterAdb 28 global FinaterAdb 29 global According triabab 20 global According triabab 21 global According triabab 22 global According triabab 23 global According triabab 24 global According triabab 25 global According triabab 26 global According triabab 27 global According triabab 28 global According triabab 29 global According triabab 20 global According triabab 21 global According triabab 22 global According triabab 23 global According triabab 24 global According triabab 25 global According triabab 26 global According triabab <	72 -	function HumeralAbductionSilderValueChanging(app, event)	
4 4088. yCoofdoolthWhiteAB 7 4088. YCoofdoolthWhiteAB 8 4088. YCoofdoolthWhiteAB 81 4088. mattrPointAB 82 4108. mattrPointAB 83 global mattrPointAB 84 global mattrPointAB 85 global mattrPointAB	/5	global xcords/ostronmetrixab	
7 Libbal Finister/Ab 77 Libbal Finister/Ab 78 Libbal Finister/Ab 78 Libbal Finister/Ab 80 global master/SintAdb 81 Libbal master/SintAdb 82 Libbal master/SintAdb 83 Libbal master/SintAdb 84 Libbal master/SintAdb 85 global master/SintAdb	74	global yCooraspositionmatrixab	
7 2005 73 2005 73 2005 74 2005 75 2005 75 2005 76 2005 77 2005 78 2005 79 2005 79 2005 70 2005	75	global floorastostionmatrixad	
g doba (miscerub 70 g doba (miscerub 70 g doba (miscerub) 80 g doba (miscerub) 81 g doba (miscerub) 82 g doba (miscerub) 83 g doba (miscerub) 84 g doba (miscerub) 85 g doba (miscerub)	70	Subda, ThesterAnd	
29 guidas materiau 20 gidas materiau 21 gidas materiau 22 gidas materiau 23 gidas materiau 24 gidas materiau 25 gidas materiau 26 gidas materiau	79	SLUDG THESE THE	
00 glabal matterbointAbb 01 glabal matterbointAbb 02 glabal matterbointAbb 03 glabal matterbointAbb 04 glabal matterbointAbb 05 glabal matterbointAbb	70	Saddar Historizab	
11 cjobak materbointekb 12 cjobak materbointekb 13 cjobak materbointekb 14 cjobak materbointekb 15 cjobak materbointekb	80	alabal mastarDointAAb	
12 glabal matterPointCab 13 glabal matterPointCab 14 glabal matterPointCab 15 glabal matterPointCab 16 glabal matterPointCab	81		
83 glóbaj matterPolintAbb 84 glóbaj matterPolintAb 85 glóbaj matterPolintAb	82	alobal masterPoint/Ab	
64 global masterPointEAb 85 global masterPointFAb	83	global masterPointDAb	
85 global masterPointFAb	84	global masterPointEAb	
66 V	85	global masterPointFAb	
	86		*
			• •

87	global motorA
88	global motorB
89	global motorCPosY
90	global motorCNegY
91	global motorCX
92	global motorD
93	global motorE
94	global motorFZ
95	global motorFPosY
96	global motorFNegY
97	
98	changingValue = event.Value;
99	app.HumeralFlexionSlider.Value = 0;
100	<pre>getPoint = round((round(changingValue,1)/0.1)+1);</pre>
101	
102	if app.ShowWiresCheckBox.Value 1
103	wireColor = 'r';
104	elseif app.ShowWiresCheckBox.Value 0
105	wireColor = 'w:square';
106	end
107	
108	plot3(app.UIAxesAngled,[masterPointAAb(getPoint,1) motorA(1)], [masterPointAAb(getPoint,2) motorA(2)], [masterPointAAb(getPoint,3) motorA(3)], wireColor,
109	[masterPoint8Ab(getPoint,1) motor8(1)], [masterPoint8Ab(getPoint,2) motor8(2)], [masterPoint8Ab(getPoint,3) motor8(3)], wireColor,
110	[masterPointCAb(getPoint,1) motorCPosY(1)], [masterPointCAb(getPoint,2) motorCPosY(2)], [masterPointCAb(getPoint,3) motorCPosY(3)], wireColor,
111	[masterPointCAb(getPoint,1) motorCNegY(1)], [masterPointCAb(getPoint,2) motorCNegY(2)], [masterPointCAb(getPoint,3) motorCNegY(3)], wireColor,
112	[masterPointCAb(getPoint,1) motorCX(1)], [masterPointCAb(getPoint,2) motorCX(2)], [masterPointCAb(getPoint,3) motorCX(3)], wireColor,
113	[masterPointDAb(getPoint,1) motorD(1)], [masterPointDAb(getPoint,2) motorD(2)], [masterPointDAb(getPoint,3) motorD(3)], wireColor,
114	[masterPointEAb(getPoint,1) motorE(1)], [masterPointEAb(getPoint,2) motorE(2)], [masterPointEAb(getPoint,3) motorE(3)], wireColor,
115	[masterPointFAb(getPoint,1) motorFPosY(1)], [masterPointFAb(getPoint,2) motorFPosY(2)], [masterPointFAb(getPoint,3) motorFPosY(3)], wireColor,
116	[masterPointFAb(getPoint,1) motorFNegY(1)], [masterPointFAb(getPoint,2) motorFNegY(2)], [masterPointFAb(getPoint,3) motorFNegY(3)], wireColor,
117	[masterPointFAb(getPoint,1) motorFZ(1)], [masterPointFAb(getPoint,2) motorFZ(2)], [masterPointFAb(getPoint,3) motorFZ(3)], wireColor,
118	[xCoordsPositionMatrixAb(getPoint,1) xCoordsPositionMatrixAb(getPoint,2) xCoordsPositionMatrixAb(getPoint,5) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,2) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,2) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,2) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,2) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,2) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPo
119	[yCoordsPositionMatrixAb(getPoint,1) yCoordsPositionMatrixAb(getPoint,2) yCoordsPositionMatrixAb(getPoint,5) yCoordsPositionMatrixAb(getPoint,3) yCoordsPositionMatrixAb(getPoint,2) ;
120	[zCoordsPositionMatrixAb(getPoint,1) zCoordsPositionMatrixAb(getPoint,2) zCoordsPositionMatrixAb(getPoint,5) zCoordsPositionMatrixAb(getPoint,3) zCoordsPositionMatrixAb(getPoint,2)],'.k-',
121	[xCoordsPositionMatrixAb(getPoint,2) FMasterXAb(getPoint,1)], [vCoordsPositionMatrixAb(getPoint,2) FMasterYAb(getPoint,1)], [zCoordsPositionMatrixAb(getPoint,2) FMasterZAb(getPoint,1)], '.k-')
122	
123	
124	plot3(app.UIAxesFront,[masterPointAAb(getPoint,1) motorA(1)], [masterPointAAb(getPoint,2) motorA(2)], [masterPointAAb(getPoint,3) motorA(3)], wireColor,
125	[masterPointBAb(getPoint,1) motorB(1)], [masterPointBAb(getPoint,2) motorB(2)], [masterPointBAb(getPoint,3) motorB(3)], wireColor,
126	[masterPointCAb(getPoint,1) motorCPosY(1)], [masterPointCAb(get#oint,2) motorCPosY(2)], [masterPointCAb(getPoint,3) motorCPosY(3)], wireColor,
127	[masterPointCAb(getPoint,1) motorCNegY(1)], [masterPointCAb(get@oint,2) motorCNegY(2)], [masterPointCAb(getPoint,3) motorCNegY(3)], wireColor,
128	[masterPointCAb(getPoint,1) motorCX(1)], [masterPointCAb(getPoint,2) motorCX(2)], [masterPointCAb(getPoint,3) motorCX(3)], wireColor,
129	[masterPointDAb(getPoint,1) motorD(1)], [masterPointDAb(getPoint,2) motorD(2)], [masterPointDAb(getPoint,3) motorD(3)], wireColor,

130	martarDointEAb(ratDoint 1) motorE(1)] [martarDointEAb(ratDoint 2) motorE(2)] [martarDointEAb(ratDoint 3) motorE(3)] wire(olor
131	[materiolital(active)] model(x)] [materiolital(active)] [materiolita
132	[moterPoint(h)(getOint()) moterFleeV(1)]. [moterPoint(h)(getOint()) moterFleeV(1)]. [moterPoint(h)(getOint()) moterFleeV(1)]. [insterPoint(h)(getOint()) moterFleeV(1)].
133	masterPointEd/(cerPoint 1) motorF7(1)] [materPointEd/(cerPoint 2) motorF7(2)] wire(olor
13.4	[v[anddDastfinHatrixh/gatDoint]) v[anddDastfinHatrixh/gatDoint]) v[anddDastfinHatrixh/gatDoint]) v[anddDastfinHatrixh/gatDoint]) v[anddDastfinHatrixh/gatDoint]) v[anddDastfinHatrixh/gatDoint]) v[anddDastfinHatrixh/gatDoint])
135	woords of characterial and (see cards) woords of characte
136	[scondDesitionNatrixh(ratDoint 1) scondsDesitionNatrixh(ratDoint 2) scondsDesitionNatrixh(ratDoint 3) scondsDesitionNatrixh(ratDoint 4) sconds
137	[voordoriingland voorgevand) beland voorgevand voo
138	[voor as extrained warfed ericks]. Lince warfed ericks]]). Door as extrained warfed ericks]]). [See as extrained warfed ericks]].
139	
140	ann HTAvasFront CamaraDosition = [0.05.1 .0.075].
141	ann IlläverFront CameraTarget = [0,05,0,-0,075]
142	abbient unter an unter an an Bar (area 2)
143	plot3(app.UIAxesSide.[masterPointAAb(getPoint.1) motorA(1)], [masterPointAAb(getPoint.2) motorA(2)], [masterPointAAb(getPoint.3) motorA(3)], wireColor,
144	[masterPointBab(eetPoint_1) motorB(1)]. [masterPointBab(eetPoint_2) motorB(2)]. [masterPointBab(eetPoint_3) motorB(3)]. wice(olor
145	masterPointCAb(setPoint_1) motorCPosY(1)]. [masterPointCAb(setPoint_2) motorCPosY(2)]. [masterPointCAb(setPoint_3) motorCPosY(3)]. wireColor
146	<pre>masterPointCAB(eetPoint.1) motorCHeeY(1)], masterPointCAB(eetPoint.2) motorCHeeY(2)], masterPointCAB(eetPoint.3) motorCHeeY(3)], wireColor,</pre>
147	[masterPointCAb(setPoint.1) motorCX(1)], [masterPointCAb(setPoint.2) motorCX(2)], [masterPointCAb(setPoint.3) motorCX(3)], wireColor,
148	<pre>masterPointDAb(getPoint_1) motorD(1)], [masterPointDAb(getPoint_2) motorD(2)], [masterPointDAb(getPoint_3) motorD(3)], wireColor,</pre>
149	[masterPointEAb(getPoint,1) motorE(1)], [masterPointEAb(getPoint,2) motorE(2)], [masterPointEAb(getPoint,3) motorE(3)], wireColor,
150	[masterPointFAb(getPoint,1) motorFPosY(1)], [masterPointFAb(getPoint,2) motorFPosY(2)], [masterPointFAb(getPoint,3) motorFPosY(3)], wireColor,
151	[masterPointFAb(getPoint,1) motorFNegY(1)], [masterPointFAb(getPoint,2) motorFNegY(2)], [masterPointFAb(getPoint,3) motorFNegY(3)], wireColor,
152	[masterPointFAb(getPoint,1) motorFZ(1)], [masterPointFAb(getPoint,2) motorFZ(2)], [masterPointFAb(getPoint,3) motorFZ(3)], wireColor,
153	[xCoordsPositionNatrixAb(getPoint,1) xCoordsPositionNatrixAb(getPoint,2) xCoordsPositionNatrixAb(getPoint,5) xCoordsPositionNatrixAb(getPoint,3) xCoordsPositionNatrixAb(getPoint,4) xCoordsPositionNatrixAb(getPoint,1)],
154	[yCoordsPositionNatrixAb(getPoint,1) yCoordsPositionNatrixAb(getPoint,2) yCoordsPositionNatrixAb(getPoint,5) yCoordsPositionNatrixAb(getPoint,3) yCoordsPositionNatrixAb(getPoint,4) yCoordsPositionNatrixAb(getPoint,1)],
155	[zCoordsPositionNatrixAb(getPoint,1) zCoordsPositionNatrixAb(getPoint,2) zCoordsPositionNatrixAb(getPoint,5) zCoordsPositionNatrixAb(getPoint,3) zCoordsPositionNatrixAb(getPoint,4) zCoordsPositionNatrixAb(getPoint,1)],'.k-',
156	[xCoordsPositionNatrixAb(getPoint,2) FMasterXAb(getPoint,1)], [yCoordsPositionNatrixAb(getPoint,2) FMasterYAb(getPoint,1)], [zCoordsPositionNatrixAb(getPoint,2) FMasterZAb(getPoint,1)], '.k-')
157	
158	
159	app.UIAxesSide.CameraPosition = [-1,0, 0.075];
160	app.UIAxesSide.CameraTarget = [0,0,-0.075];
161	
162	plot3(app.UIAxesTop,[masterPointAAb(getPoint,1) motorA(1)], [masterPointAAb(getPoint,2) motorA(2)], [masterPointAAb(getPoint,3) motorA(3)], wireColor,
163	[masterPointBAb(getPoint,1) motorB(1)], [masterPointBAb(getPoint,2) motorB(2)], [masterPointBAb(getPoint,3) motorB(3)], wireColor,
164	[masterPointCAb(getPoint,1) motorCPosY(1)], [masterPointCAb(getPoint,2) motorCPosY(2)], [masterPointCAb(getPoint,3) motorCPosY(3)], wireColor,
165	[masterPointCAb(getPoint,1) motorCHegV(1)], [masterPointCAb(getPoint,2) motorCHegV(2)], [masterPointCAb(getPoint,3) motorCHegV(3)], wireColor,
166	[masterPointCAb(getPoint,1) motorCX(1)], [masterPointCAb(getPoint,2) motorCX(2)], [masterPointCAb(getPoint,3) motorCX(3)], wireColor,
167	[masterPointDAb(getPoint,1) motorD(1)], [masterPointDAb(getPoint,2) motorD(2)], [masterPointDAb(getPoint,3) motorD(3)], wireColor,
168	[masterPointEAb(getPoint,1) motorE(1)], [masterPointEAb(getPoint,2) motorE(2)], [masterPointEAb(getPoint,3) motorE(3)], wireColor,
169	[masterPointFAb(getPoint,1) motorFPosY(1)], [masterPointFAb(getPoint,2) motorFPosY(2)], [masterPointFAb(getPoint,3) motorFPosY(3)], wireColor,
170	[masterPointFAb(getPoint,1) motorFNegY(1)], [masterPointFAb(getPoint,2) motorFNegY(2)], [masterPointFAb(getPoint,3) motorFNegY(3)], wireColor,
171	[masterPointFAb(getPoint,1) motorFZ(1)], [masterPointFAb(getPoint,2) motorFZ(2)], [masterPointFAb(getPoint,3) motorFZ(3)], wireColor,
172	[xCoordsPositionMatrixAb(getPoint,1) xCoordsPositionMatrixAb(getPoint,2) xCoordsPositionMatrixAb(getPoint,5) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,1), 🔻
173	A Design of the second s

173	[yCoordsPositionMatrixAb(getPoint,1) yCoordsPositionMatrixAb(get	Point,2) yCoordsPositionMatrixAb(getPoint,5) yCoordsPositionMatrixAb(getPoint,3) yCoordsPositionMatrixAb(getPoint,4) yCoordsPositionMatrixAb(getPoint,1)], 🔺
174	[zCoordsPositionMatrixAb(getPoint.1) zCoordsPositionMatrixAb(get	Point.2) /CoordsPositionMatrixAb(getPoint.5) /CoordsPositionMatrixAb(getPoint.3) /CoordsPositionMatrixAb(getPoint.4) /CoordsPositionMatrixAb(getPoint.1)].
175	[vCoordsPositionMatrixAb(gatPoint 2) EMarterVAb(gatPoint 1)] [v	CondeParitionMatrixAb(catPoint 1)] [CondeParitionMatrixAb(catPoint 1)] [
176	[Action distorter to match the (get of me, r) in a set (get of me, r)]; []	contraction and the (Bertoruch) in and the (Bertoruch) in the (Bertoruch) in a contraction (Bertoruch)). In a
177		
470		
1/8	app.UlAxesTop.CameraPosition = [0.05,0,1];	
179	app.UIAxesTop.CameraTarget = [0.05,0,0];	
180		
181		
182	global currentFlexIndex	
183		
184	if currentFlexIndex > 1	
185	flexCode(1)	
186	abCode(getPoint(1))	
187	else	
188	abCode(getPoint(1))	
189	end	
190	end	
191		
192	% Value changed function: HumanalElavionEliden	
102	function Humanillantiantialuctionsider	
100	Tone cton numeral restonsition variation (app), even()	
194	alabel accord best for the state of the	
195	global xCoordsPositionMatrixFlex	
196	global ycoordsPositionmatrixilex	
197	global zCoordsPositionMatrixFlex	
198	global FMasterXFlex	
199	global FMasterYFlex	
200	global FMasterZFlex	
201		
202		
203	global masterPointAFlex	
204	global masterPointBFlex	
205	global masterPointCFlex	
206	global masterPointDFlex	
207	global masterPointEflex	
208	global masterPointEElex	
209		
210		
211	global motorà	
212	global sotorB	
213	global sotor(PerV	
214		
214	Storet moroucueRt	
215	giobal motorca	
216	4	

216	global motorD	
217	global motorE	
218	global motorFZ	
219	global motorFPosY	
220	global motorFNegY	
221		
222	changingValue = event.Value;	
223	app.HumeralAbductionSlider.Value = 0;	
224	<pre>getPoint = round((round(changingValue,1)/0.1)+1);</pre>	
225		
226	if app.ShowWiresCheckBox.Value 1	
227	wireColor = 'r';	
228	elseif app.ShowWiresCheckBox.Value 0	
229	wireColor = 'wisquare':	
230	end	
231		
232		-
233	plot3(app.UIAxesAngled.[masterPointAFlex(getPoint,1) motorA(1)], [masterPointAFlex(getPoint,2) motorA(2)], [masterPointAFlex(getPoint,3) motorA(3)], wireColor,	
234	[masterPointBFlex(getPoint.1) motorB(1)], [masterPointBFlex(getPoint.2) motorB(2)], [masterPointBFlex(getPoint.3) motorB(3)], wireColor,	
235	[masterPoint(Flex(getPoint,1) motor(PosY(1)], [masterPoint(Flex(getPoint,2) motor(PosY(2)], [masterPoint(Flex(getPoint,3) motor(PosY(3)], wire(Color,	
236	[masterPointCFlex/getPoint,1] motorCNegY(1)], [masterPointCFlex/getPoint,2] motorCNegY(2)], [masterPointCFlex/getPoint,3] motorCNegY(3)], wireColor,	
237	[masterPointCFlex(getPoint,1) motorCX(1)], [masterPointCFlex(getPoint,2) motorCX(2)], [masterPointCFlex(getPoint,3) motorCX(3)], wireColor,	
238	[masterPointDFlex(getPoint,1) motorD(1)], [masterPointDFlex(getPoint,2) motorD(2)], [masterPointDFlex(getPoint,3) motorD(3)], wireColor,	
239	[masterPointEFlex(getPoint,1) motorE(1)], [masterPointEFlex(getPoint,2) motorE(2)], [masterPointEFlex(getPoint,3) motorE(3)], wireColor,	
240	[masterPointFflex(getPoint,1) motorFPosY(1)], [masterPointFflex(getPoint,2) motorFPosY(2)], [masterPointFflex(getPoint,3) motorFPosY(3)], wireColor,	
241	[masterPointFFlex(getPoint,1) motorFNegY(1)], [masterPointFflex(getPoint,2) motorFNegY(2)], [masterPointFFlex(getPoint,3) motorFNegY(3)], wireColor,	
242	[masterPointFFlex(getPoint,1) motorFZ(1)], [masterPointFFlex(getPoint,2) motorFZ(2)], [masterPointFFlex(getPoint,3) motorFZ(3)], wireColor,	
243	[xCoordsPositionMatrixFlex(getPoint.1) xCoordsPositionMatrixFlex(getPoint.2) xCoordsPositionMatrixFlex(getPoint.3) xCoordsPositionMatrixFlex(getPoint.4) xCoordsPositionMatrixFlex(getPoint.1)],	
244	[VCoordsPositionMatrixFlex(getPoint,1) vCoordsPositionMatrixFlex(getPoint,2) vCoordsPositionMatrixFlex(getPoint,3) vCoordsPositionMatrixFlex(getPoint,4) vCoordsPositionMatrixFlex(getPoint,1),	
245	[2CoordsPositionMatrixFlex(getPoint,1) 2CoordsPositionMatrixFlex(getPoint,2) 2CoordsPositionMatrixFlex(getPoint,3) 2CoordsPositionMatrixFlex(getPoint,2), '.+.',	
246	[xCoordsPositionHatrixflex(getPoint.2) FMasterXflex(getPoint.1)], [xCoordsPositionMatrixflex(getPoint.2) FMasterXflex(getPoint.2) FMasterXflex(getPoint.1)], [xCoordsPositionMatrixflex(getPoint.2) FMasterXflex(getPoint.2)	
247		
248		
249	plot3(app.UIAxesFront,[masterPointAFlex(getPoint,1) motorA(1)], [masterPointAFlex(getPoint,2) motorA(2)], [masterPointAFlex(getPoint,3) motorA(3)], wireColor,	
250	[masterPointBFlex(getPoint,1) motorB(1)], [masterPointBFlex(getPoint,2) motorB(2)], [masterPointBFlex(getPoint,3) motorB(3)], wireColor,	
251	[masterPointCFlex(getPoint,1) motorCPosV(1)], [masterPointCFlex(getPoint,2) motorCPosV(2)], [masterPointCFlex(getPoint,3) motorCPosV(3)], wireColor,	
252	[masterPointCFlex(getPoint,1) motorCNegY(1)], [masterPointCFlex(getPoint,2) motorCNegY(2)], [masterPointCFlex(getPoint,3) motorCNegY(3)], wireColor,	
253	[masterPointCFlex(getPoint,1) motorCX(1)], [masterPointCFlex(getPoint,2) motorCX(2)], [masterPointCFlex(getPoint,3) motorCX(3)], wireColor,	
254	[masterPointDFlex(getPoint,1) motorD(1)], [masterPointDFlex(getPoint,2) motorD(2)], [masterPointDFlex(getPoint,3) motorD(3)], wireColor,	
255	[masterPointEFlex(getPoint,1) motorE(1)], [masterPointEFlex(getPoint,2) motorE(2)], [masterPointEFlex(getPoint,3) motorE(3)], wireColor,	
256	[masterPointFFlex(getPoint,1) motorFPosY(1)], [masterPointFFlex(getPoint,2) motorFPosY(2)], [masterPointFFlex(getPoint,3) motorFPosY(3)], wireColor,	
257	[masterPointFFlex(getPoint,1) motorFNegY(1)], [masterPointFFlex(getPoint,2) motorFNegY(2)], [masterPointFFlex(getPoint,3) motorFNegY(3)], wireColor,	
258	[masterPointFFlex(getPoint,1) motorFZ(1)], [masterPointFFlex(getPoint,2) motorFZ(2)], [masterPointFFlex(getPoint,3) motorFZ(3)], wireColor,	-
259	Roningening interesting of residencies of residenci	

259 260 261 262 263	[xCordsPositionNatrixFlex(getPoint,1) xCordsPositionNatrixFlex(getPoint,2) xCordsPositionNatrixFlex(getPoint,3) xCordsPosi	-
264	app.UIAxesFront.CameraPosition = [0.05,1,-0.075];	
265 266	app.UIAxesFront.CameraTarget = [0.05,0,-0.075];	
267	plot3(app.UIAxesSide,[masterPointAFlex(getPoint,1) motorA(1)], [masterPointAFlex(getPoint,2) motorA(2)], [masterPointAFlex(getPoint,3) motorA(3)], wireColor,	
268	[masterPoint8Flex(getPoint,1) motor8(1)], [masterPoint8Flex(getPoint,2) motor8(2)], [masterPoint8Flex(getPoint,3) motor8(3)], wireColor,	
269	[masterPointCFlex(getPoint,1) motorCPosY(1)], [masterPointCFlex(getPoint,2) motorCPosY(2)], [masterPointCFlex(getPoint,3) motorCPosY(3)], wireColor,	
270	[masterPointCFlex(getPoint,1) motorCNegY(1)], [masterPointCFlex(getPoint,2) motorCNegY(2)], [masterPointCFlex(getPoint,3) motorCNegY(3)], wireColor,	
271	[masterPointCFlex(getPoint,1) motorCX(1)], [masterPointCFlex(getPoint,2) motorCX(2)], [masterPointCFlex(getPoint,3) motorCX(3)], wireColor,	
272	[masterPointDFlex(getPoint,1) motorD(1)], [masterPointDFlex(getPoint,2) motorD(2)], [masterPointDFlex(getPoint,3) motorD(3)], wireColor,	
273	[masterPointEFlex(getPoint,1) motorE(1)], [masterPointEFlex(getPoint,2) motorE(2)], [masterPointEFlex(getPoint,3) motorE(3)], wireColor,	
274	[masterPointFFlex(getPoint,1) motorFPosY(1)], [masterPointFFlex(getPoint,2) motorFPosY(2)], [masterPointFFlex(getPoint,3) motorFPosY(3)], wireColor,	
275	[masterPointFFlex(getPoint,1) motorFNegY(1)], [masterPointFFlex(getPoint,2) motorFNegY(2)], [masterPointFFlex(getPoint,3) motorFNegY(3)], wireColor,	
276	[masterPointFFlex(getPoint,1) motorFZ(1)], [masterPointFFlex(getPoint,2) motorFZ(2)], [masterPointFFlex(getPoint,3) motorFZ(3)], wireColor,	
277	[xCoordsPositionMatrixFlex(getPoint,1) xCoordsPositionMatrixFlex(getPoint,2) xCoordsPositionMatrixFlex(getPoint,3) xCoordsPositionMatrixFlex(getPoint,4) xCoordsPositionMatrixFlex(getPoint,1)],	
278	[yCoordsPositionMatrixFlex(getPoint,1) yCoordsPositionMatrixFlex(getPoint,2) yCoordsPositionMatrixFlex(getPoint,3) yCoordsPositionMatrixFlex(getPoint,4) yCoordsPositionMatrixFlex(getPoint,1)],	
279	[zCoordsPositionMatrixFlex(getPoint,1) zCoordsPositionMatrixFlex(getPoint,2) zCoordsPositionMatrixFlex(getPoint,3) zCoordsPositionMatrixFlex(getPoint,1)], '.k-',	
280	[xCoordsPositionMatrixFlex(getPoint,2) FMasterXFlex(getPoint,1)], [yCoordsPositionMatrixFlex(getPoint,2) FMasterYFlex(getPoint,1)], [xCoordsPositionMatrixFlex(getPoint,2) FMasterXFlex(getPoint,1)], '.k-')	
281		
282	app.UIAxesSide.CameraPosition = [-1,0,-0.075];	
283	<pre>app.UIAxesSide.CameraTarget = [0,0,-0.075];</pre>	
284		
285	plot3(app.UIAxesTop,[masterPointAFlex(getPoint,1) motorA(1)], [masterPointAFlex(getPoint,2) motorA(2)], [masterPointAFlex(getPoint,3) motorA(3)], wireColor,	
286	[masterPointBFlex(getPoint,1) motorB(1)], [masterPointBFlex(getPoint,2) motorB(2)], [masterPointBFlex(getPoint,3) motorB(3)], wireColor,	
287	[masterPointCFlex(getPoint,1) motorCPosY(1)], [masterPointCFlex(getPoint,2) motorCPosY(2)], [masterPointCFlex(getPoint,3) motorCPosY(3)], wireColor,	
288	[masterPointCFlex(getPoint,1) motorCNegY(1)], [masterPointCFlex(getPoint,2) motorCNegY(2)], [masterPointCFlex(getPoint,3) motorCNegY(3)], wireColor,	
289	[masterPointCFlex(getPoint,1) motorCX(1)], [masterPointCFlex(getPoint,2) motorCX(2)], [masterPointCFlex(getPoint,3) motorCX(3)], wireColor,	
290	[masterPointDFlex(getPoint,1) motorD(1)], [masterPointDFlex(getPoint,2) motorD(2)], [masterPointDFlex(getPoint,3) motorD(3)], wireColor,	
291	[masterPointEFlex(getPoint,1) motorE(1)], [masterPointEFlex(getPoint,2) motorE(2)], [masterPointEFlex(getPoint,3) motorE(3)], wireColor,	
292	[mastervointriex(getvoint,1) motorPvos(1)], [mastervointriex(getvoint,2) motorPvos(2)], wirecolor,	
293	[masterPointFilex(getPoint,1) motorNegY(1)], [masterPointFilex(getPoint,2) motorNegY(2)], [masterPointFilex(getPoint,3) motorNegY(3)], wireColor,	
294	[mattervointriix(getvoint,1)], [mattervointriik(getvoint,2)] motorr2(1)], [mattervointriik(getvoint,3)] motorr2(1)], [mattervointriik(getvoint,3)] motorr2(1)], [mattervointriik(getvoint,3)]	
295	[x.cordsrositionmetrix1ex[getVoint,3] x.cordsrositionmetrix1ex[getVoint,2] x.cordsPositionMetrix1ex[getVoint,3] x.cordsPositionMetrix1ex[getVoint,4] x.cordsrositionMetrix1ex[getVoint,3] x.cordsPositionMetrix1ex[getVoint,3] x.cordsPos	
290	[[condstoaling]] (condstoaling] (condstoaling] (condstoaling] (condstoaling] (condstoaling) (con	
297	[LCUD OF DELCOMENT AL LEAGETOINE, 1] LCUD OF DELCOMENT AL REAGETOINE, 1] LCUD OF DELCOMENT, 1] LCUD OF DELCOMENT AL REAGETOINE, 1] LCUD OF	
298	[xcoordsPositionMatrixFlex(getPoint,2) FMasterXFlex(getPoint,1)], [ycoordsPositionMatrixFlex(getPoint,2) FMasterZFlex(getPoint,1)], '.k-')	
200	and Minister Competence 10 (C. A. 1)	
301	$ap(barren by converted variable) = \{a (b, c), c \}$	
302	abhanacaubhceann an Ber - facalatat	
- 54	Y Provide the second seco	

305	global currentAbIndex		
306			
307	if currentAbIndex > 1		
308	abCode(1)		
309	flexCode(getPoint(1))		
310	else		
311	flexCode(getPoint(1))		
312	end		
313	4		
314			
315	/alue changed function: ShowWiresCheckBox		
316	<pre>httion ShowWiresCheckBoxValueChanged(app, event)</pre>		
317	global xCoordsPositionMatrixFlex		
318	global yCoordsPositionMatrixFlex		
319	global zCoordsPositionMatrixFlex		
320	global FMasterXFlex		
321	global FMasterYFlex		
322	global FMasterZFlex		
323			
324			
325	global masterPointAFlex		-
326	global masterPointBFlex		
327	global masterPointCFlex		
328	global masterPointDFlex		
329	global masterPointEFlex		
330	global masterPointFFlex		
331			
332	global xCoordsPositionMatrixAb		
333	global yCoordsPositionMatrixAb		
334	global zCoordsPositionMatrixAb		
335	global FMasterXAb		
336	global FMasterYAb		
337	global FMasterZAb		
338			
339	global masterPointAAb		
340	global masterPointBAb		
341	global masterPointCAb		
342	global masterPointDAb		
343	global masterPointEAb		
344	global masterPointFAb		
345			
346			+
347	global motorá	1	

347	global motorA	
348	global motorB	
349	global motorCPosY	
350	global motorCNegY	
351	global motorCX	
352	global motorD	
353	global motorE	
354	global motorFZ	
355	global motorFPosY	
356	global motorFNegY	
357		
358	<pre>value = app.ShowWiresCheckBox.Value;</pre>	
359	if app.HumeralAbductionSlider.Value > 0	
360	<pre>getPoint = round((round(app.HumeralAbductionSlider.Value,1)/0.1)+1);</pre>	
361		
362	if value == 1	
363	wireColor = 'r';	
364	elseif value == 0	
365	wireColor = 'w:square';	
366	end	
367		_
368	plot3(app.UIAxesAngled,[masterPointAAb(getPoint,1) motor%(1)], [masterPointAAb(getPoint,2) motorA(2)], [masterPointAAb(getPoint,3) motorA(3)], wireColor,	
369	[masterPoint8Ab(getPoint,1) motor8(1)], [masterPoint8Ab(getPoint,2) motor8(2)], [masterPoint8Ab(getPoint,3) motor8(3)], wireColor,	
370	[masterPointCAb(getPoint,1) motorCPosY(1)], [masterPointCAb(getPoint,2) motorCPosY(2)], [masterPointCAb(getPoint,3) motorCPosY(3)], wireColor,	
371	[masterPointCAb(getPoint,1) motorCNegY(1)], [masterPointCAb(getPoint,2) motorCNegY(2)], [masterPointCAb(getPoint,3) motorCNegY(3)], wireColor,	
372	[masterPointCAb(getPoint,1) motorCX(1)], [masterPointCAb(getPoint,2) motorCX(2)], [masterPointCAb(getPoint,3) motorCX(3)], wireColor,	
373	[masterPointDAb(getPoint,1) motorD(1)], [masterPointDAb(getPoint,2) motorD(2)], [masterPointDAb(getPoint,3) motorD(3)], wireColor,	
374	[masterPointEAb(getPoint,1) motorE(1)], [masterPointEAb(getPoint,2) motorE(2)], [masterPointEAb(getPoint,3) motorE(3)], wireColor,	
375	[masterPointFAb(getPoint,1) motorFPosY(1)], [masterPointFAb(getPoint,2) motorFPosY(2)], [masterPointFAb(getPoint,3) motorFPosY(3)], wireColor,	
376	[masterPointFAb(getPoint,1) motorFNegY(1)], [masterPointFAb(getPoint,2) motorFNegY(2)], [masterPointFAb(getPoint,3) motorFNegY(3)], wireColor,	
377	[masterPointFAb(getPoint,1) motorFZ(1)], [masterPointFAb(getPoint,2) motorFZ(2)], [masterPointFAb(getPoint,3) motorFZ(3)], wireColor,	
378	[xCoordsPositionNatrixAb(getPoint,1) xCoordsPositionNatrixAb(getPoint,2) xCoordsPositionNatrixAb(getPoint,5) xCoordsPositionNatrixAb(getPoint,3) xCoordsPositionNatrixAb(getPoint,4) xCoordsPositionNatrixAb(getPoint,1)],	
379	[yCoordsPositionMatrixAb(getPoint,1) yCoordsPositionMatrixAb(getPoint,2) yCoordsPositionMatrixAb(getPoint,5) yCoordsPositionMatrixAb(getPoint,4) yCoordsPositionMatrixAb(getPoint,1),	
380	[zCoordsPositionNatrixAb(getPoint,1) zCoordsPositionNatrixAb(getPoint,2) zCoordsPositionNatrixAb(getPoint,5) zCoordsPositionNatrixAb(getPoint,3) zCoordsPositionNatrixAb(getPoint,4) zCoordsPositionNatrixAb(getPoint,1)],'.k-',	
381	[xCoordsPositionMatrixAb(getPoint,2) FMasterXAb(getPoint,1)], [yCoordsPositionMatrixAb(getPoint,2) FMasterYAb(getPoint,1)], [zCoordsPositionMatrixAb(getPoint,2) FMasterZAb(getPoint,1)], '.k-')	
382		
383		
384	plot3(app.UIAxesFront,[masterPointAAb(getPoint,1) motorA(1)], [masterPointAAb(getPoint,2) motorA(2)], [masterPointAAb(getPoint,3) motorA(3)], wireColor,	
385	[masterPointBAb(getPoint,1) motorB(1)], [masterPointBAb(getPoint,2) motorB(2)], [masterPointBAb(getPoint,3) motorB(3)], wireColor,	
386	[masterPointCAb(getPoint,1) motorCPosY(1)], [masterPointCAb(getPoint,2) motorCPosY(2)], [masterPointCAb(getPoint,3) motorCPosY(3)], wireColor,	
387	[masterPointCAb(getPoint,1) motorCNegY(1)], [masterPointCAb(getPoint,2) motorCNegY(2)], [masterPointCAb(getPoint,3) motorCNegY(3)], wireColor,	
388	[masterPointCAb(getPoint,1) motorCX(1)], [masterPointCAb(getPoint,2) motorCX(2)], [masterPointCAb(getPoint,3) motorCX(3)], wireColor,	
389	[masterPointDAb(getPoint,1) motorD(1)], [masterPointDAb(getPoint,2) motorD(2)], [masterPointDAb(getPoint,3) motorD(3)], wireColor,	*
390		F.

390	[masterPointEAb(getPoint,1) motorE(1)], [masterPointEAb(getPoint,2) motorE(2)], [masterPointEAb(getPoint,3) motorE(3)], wireColor,
391	[masterPointFAb(getPoint,1) motorFPosY(1)], [masterPointFAb(getPoint,2) motorFPosY(2)], [masterPointFAb(getPoint,3) motorFPosY(3)], wireColor,
392	[masterPointFAb(getPoint,1) motorFNegY(1)], [masterPointFAb(getPoint,2) motorFNegY(2)], [masterPointFAb(getPoint,3) motorFNegY(3)], wireColor,
393	[masterPointFAb(getPoint,1) motorFZ(1)], [masterPointFAb(getPoint,2) motorFZ(2)], [masterPointFAb(getPoint,3) motorFZ(3)], wireColor,
394	[xCoordsPositionMatrixAb(getPoint,1) xCoordsPositionMatrixAb(getPoint,2) xCoordsPositionMatrixAb(getPoint,5) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,4) xCoordsPositionMatrixAb(getPoint,1)],
395	[yCoordsPositionNatrixAb(getPoint,1) yCoordsPositionNatrixAb(getPoint,2) yCoordsPositionNatrixAb(getPoint,5) yCoordsPositionNatrixAb(getPoint,4) yCoordsPositionNatrixAb(getPoint,1)],
396	[zCoordsPositionNatrixAb(getPoint,1) zCoordsPositionNatrixAb(getPoint,2) zCoordsPositionNatrixAb(getPoint,3) zCoordsPositionNatrixAb(getPoint,4) zCoordsPositionNatrixAb(getPoint,1)],'.k-',
397	[xCoordsPositionMatrixAb(getPoint,2) FMasterXAb(getPoint,1)], [yCoordsPositionMatrixAb(getPoint,2) FMasterYAb(getPoint,1)], [zCoordsPositionMatrixAb(getPoint,2) FMasterZAb(getPoint,1)], '.k-')
398	
399	
400	app.UIAxesFront.CameraPosition = [0.05,1,-0.075];
401	app.UIAxesFront.CameraTarget = [0.05,0,-0.075];
402	
403	plot3(app.UIAxesSide,[masterPointAAb(getPoint,1) motorA(1)], [masterPointAAb(getPoint,2) motorA(2)], [masterPointAAb(getPoint,3) motorA(3)], wireColor,
484	[masterPointBAb(getPoint,1) motorB(1)], [masterPointBAb(getPoint,2) motorB(2)], [masterPointBAb(getPoint,3) motorB(3)], wireColor,
405	[masterPointCAb(getPoint,1) motorCPosY(1)], [masterPointCAb(getPoint,2) motorCPosY(2)], [masterPointCAb(getPoint,3) motorCPosY(3)], wireColor,
406	[masterPointCAb(getPoint,1) motorCNegY(1)], [masterPointCAb(getPoint,2) motorCNegY(2)], [masterPointCAb(getPoint,3) motorCNegY(3)], wireColor,
407	[masterPointCAb(getPoint,1) motorCX(1)], [masterPointCAb(getPoint,2) motorCX(2)], [masterPointCAb(getPoint,3) motorCX(3)], wireColor,
408	[masterPointDAb(getPoint,1) motorD(1)], [masterPointDAb(getPoint,2) motorD(2)], [masterPointDAb(getPoint,3) motorD(3)], wireColor,
409	[masterPointEAb(getPoint,1) motorE(1)], [masterPointEAb(getPoint,2) motorE(2)], [masterPointEAb(getPoint,3) motorE(3)], wireColor,
410	[masterPointFAb(getPoint,1) motorFPosY(1)], [masterPointFAb(getPoint,2) motorFPosY(2)], [masterPointFAb(getPoint,3) motorFPosY(3)], wireColor,
411	[masterPointFAb(getPoint,1) motorFNegY(1)], [masterPointFAb(getPoint,2) motorFNegY(2)], [masterPointFAb(getPoint,3) motorFNegY(3)], wireColor,
412	[masterPointFAb(getPoint,1) motorFZ(1)], [masterPointFAb(getPoint,2) motorFZ(2)], [masterPointFAb(getPoint,3) motorFZ(3)], wireColor,
413	[xCoordsPositionMatrixAb(getPoint,1) xCoordsPositionMatrixAb(getPoint,2) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,4) xCoordsPositionMatrixAb(getPoint,1),
414	[yCoordsPositionMatrixAb(getPoint,1) yCoordsPositionMatrixAb(getPoint,2) yCoordsPositionMatrixAb(getPoint,3) yCoordsPositionMatrixAb(getPoint,4) yCoordsPositionMatrixAb(getPoint,1)],
415	[zCoordsPositionMatrixAb(getPoint,1) zCoordsPositionMatrixAb(getPoint,2) zCoordsPositionMatrixAb(getPoint,3) zCoordsPositionMatrixAb(getPoint,4) zCoordsPositionMatrixAb(getPoint,1)],'.k-',
416	[xCoordsPositionMatrixAb(getPoint,2) FMasterXAb(getPoint,1)], [yCoordsPositionMatrixAb(getPoint,2) FMasterYAb(getPoint,1)], [zCoordsPositionMatrixAb(getPoint,2) FMasterZAb(getPoint,1)], '.k-')
417	
418	
419	app.UIAxesSide.CameraPosition = [-1,0,-0.075];
420	app.UIAxesSide.CameraTarget = [0,0,-0.075];
421	
422	plot3(app.UIAxesTop,[masterPointAAb(getPoint,1) motorA(1)], [masterPointAAb(getPoint,2) motorA(2)], [masterPointAAb(getPoint,3) motorA(3)], wireColor,
423	[masterPointBAb(getPoint,1) motorB(1)], [masterPointBAb(getPoint,2) motorB(2)], [masterPointBAb(getPoint,3) motorB(3)], wireColor,
424	[masterPointCAb(getPoint,1) motorCPosY(1)], [masterPointCAb(getPoint,2) motorCPosY(2)], [masterPointCAb(getPoint,3) motorCPosY(3)], wireColor,
425	[masterPointCAb(getPoint,1) motorCNegY(1)], [masterPointCAb(getPoint,2) motorCNegY(2)], [masterPointCAb(getPoint,3) motorCNegY(3)], wireColor,
426	[masterPointCAb(getPoint,1) motorCX(1)], [masterPointCAb(getPoint,2) motorCX(2)], [masterPointCAb(getPoint,3) motorCX(3)], wireColor,
427	[masterPointDAb(getPoint,1) motorD(1)], [masterPointDAb(getPoint,2) motorD(2)], [masterPointDAb(getPoint,3) motorD(3)], wireColor,
428	[masterPointEAb(getPoint,1) motorE(1)], [masterPointEAb(getPoint,2) motorE(2)], [masterPointEAb(getPoint,3) motorE(3)], wireColor,
429	[masterPointFAb(getPoint,1) motorFPosY(1)], [masterPointFAb(getPoint,2) motorFPosY(2)], [masterPointFAb(getPoint,3) motorFPosY(3)], wireColor,
430	[masterPointFAb(getPoint,1) motorFNegY(1)], [masterPointFAb(getPoint,2) motorFNegY(2)], [masterPointFAb(getPoint,3) motorFNegY(3)], wireColor,
431	[masterPointFAb(getPoint,1) motorFZ(1)], [masterPointFAb(getPoint,2) motorFZ(2)], [masterPointFAb(getPoint,3) motorFZ(3)], wireColor,
432	[xCoordsPositionMatrixAb(getPoint,1) xCoordsPositionMatrixAb(getPoint,2) xCoordsPositionMatrixAb(getPoint,5) xCoordsPositionMatrixAb(getPoint,3) xCoordsPositionMatrixAb(getPoint,4) xCoordsPositionMatrixAb(getPoint,1)], 💌
433	

433 434 435	[/coordsPositionNetrisk/getPoint,1] /coordsPositionNetrisk/getPoint,2] /coordsPositionNetrisk/getPoint,3] /coordsPositionNetrisk/getPoint,3] /coordsPositionNetrisk/getPoint,4] /coordsPosition
436	
437	
438	<pre>app.UIAxesTop.CameraPosition = [0.05,0,1];</pre>
439	app.UIAxesTop.CameraTarget = [0.05,0,0];
440	
441	elseif app.HumeralFlexionSlider.Value > 0
442	getPoint = round((round(app.HumeralFlexionSlider.Value,1)/0.1)+1);
443	
444	if value 1
445	wireColor = 'r';
446	elseif value == 0
447	wireColor = 'w:square';
448	end
449	
450	
451	pios(app.oiAccomptot). where (i): [matching (i): [m
452	[masterPointerax[getPoint,i] motor[i]]; [masterPointer[ax[getPoint,2] motor[2]]; [masterPointerax[getPoint,3] motor[3]]; Mireolar,
455	[masterPoint(rick(getPoint,1)motorCost(1)]; [masterPoint(rick(getPoint,2)]; [masterPoint(rick(getPoint,3)motorCost(3)]; wireColor,
455	[mostervanciaex[gervancja] motorchg(a)]; [mostervanciaex[gervancja] motorchg(a); [mostervanciaex[gervancja] motorchg(a)]; matedati,
456	[materoniciteRecomparimeteronica] meteronical accession acce
457	[motor date prior backget own [2] [motor (attra [motor (attra)] motor (2)] [motor (attra)] motor (2)] [motor (attra)] motor (2)] [motor (attra)] [motor (attra
458	[masterPointFE]ex(setPoint.]) motorPosy(1)]. [masterPointFE]ex(setPoint.2) motorPosy(2)]. [masterPointFE]ex(setPoint.3) motorPosy(3)]. wireColor
459	[masterPointFF]ev(etPoint.1) motorFMeeV(1)] [masterPointFF]ev(etPoint.2) motorFMeeV(2)] [masterPointFF]ev(etPoint.3) motorFMeeV(3)], wireColor,
460	[masterPointFFlex(getPoint.]) motorFZ(1)], [masterPointFFlex(getPoint.2) motorFZ(2)], [masterPointFFlex(getPoint.3) motorFZ(3)], wireColor
461	[xCoordsPositionMatrixFlex(getPoint,1) xCoordsPositionMatrixFlex(getPoint,2) xCoordsPositionMatrixFlex(getPoint,3) xCoordsPositionMatrixFlex(getPoint,4) xCoordsPositionMatrixFlex(getPoint,1)],
462	[yCoordsPositionHatrixFlex(getPoint,1) yCoordsPositionHatrixFlex(getPoint,2) yCoordsPositionHatrixFlex(getPoint,3) yCoordsPositionHatrixFlex(getPoint,4) yCoordsPositionHatrixFlex(getPoint,1)],
463	[zCoordsPositionNatrixFlex(getPoint,1) zCoordsPositionNatrixFlex(getPoint,2) zCoordsPositionNatrixFlex(getPoint,3) zCoordsPositionNatrixFlex(getPoint,4) zCoordsPositionNatrixFlex(getPoint,1)],'.k-',
464	[xCoordsPositionHatrixFlex(getPoint,2) FHasterXFlex(getPoint,1)], [yCoordsPositionHatrixFlex(getPoint,2) FHasterYFlex(getPoint,1)], [zCoordsPositionHatrixFlex(getPoint,2) FHasterZFlex(getPoint,1)], '.k-')
465	
466	
467	plot3(app.UIAxesFront,[masterPointAFlex(getPoint,1) motorA(1)], [masterPointAFlex(getPoint,2) motorA(2)], [masterPointAFlex(getPoint,3) motorA(3)], wireColor,
468	[masterPointBFlex(getPoint,1) motorB(1)], [masterPointBFlex(getPoint,2) motorB(2)], [masterPointBFlex(getPoint,3) motorB(3)], wireColor,
469	[masterPointCFlex(getPoint,1) motorCPosY(1)], [masterPointCFlex(getPoint,2) motorCPosY(2)], [masterPointCFlex(getPoint,3) motorCPosY(3)], wireColor,
470	[masterPointCFlex(getPoint,1) motorCNegY(1)], [masterPointCFlex(getPoint,2) motorCNegY(2)], [masterPointCFlex(getPoint,3) motorCNegY(3)], wireColor,
471	[masterPointCFlex(getPoint,1) motorCX(1)], [masterPointCFlex(getPoint,2) motorCX(2)], [masterPointCFlex(getPoint,3) motorCX(3)], wireColor,
472	[masterPointDFlex(getPoint,1) motorD(1)], [masterPointDFlex(getPoint,2) motorD(2)], [masterPointDFlex(getPoint,3) motorD(3)], wireColor,
473	[masterPointEFlex(getPoint,1) motorE(1)], [masterPointEFlex(getPoint,2) motorE(2)], [masterPointEFlex(getPoint,3) motorE(3)], wireColor,
474	[masterPointFFlex(getPoint,1) motorFPosY(1)], [masterPointFFlex(getPoint,2) motorFPosY(2)], [masterPointFFlex(getPoint,3) motorFPosY(3)], wireColor,
4/5	[masterPointFFlex(getPoint,1) motorFNegY(1)], [masterPointFFlex(getPoint,2) motorFNegY(2)], [masterPointFFlex(getPoint,3) motorFNegY(3)], wireColor,
4/6	

476	[masterPointFFlex(getPoint,1) motorFZ(1)], [masterPointFFlex(getPoint,2) motorFZ(2)], [masterPointFFlex(getPoint,3) motorFZ(3)], wireColor,	
477	[xCoordsPositionMatrixFlex(getPoint,1) xCoordsPositionMatrixFlex(getPoint,2) xCoordsPositionMatrixFlex(getPoint,3) xCoordsPositionMatrixFlex(getPoint,4) xCoordsPositionMatrixFlex(getPoint,1)],	
478	[yCoordsPositionMatrixFlex(getPoint,1) yCoordsPositionMatrixFlex(getPoint,2) yCoordsPositionMatrixFlex(getPoint,3) yCoordsPositionMatrixFlex(getPoint,4) yCoordsPositionMatrixFlex(getPoint,1)],	
479	[zCoordsPositionMatrixFlex(getPoint,1) zCoordsPositionMatrixFlex(getPoint,2) zCoordsPositionMatrixFlex(getPoint,3) zCoordsPositionMatrixFlex(getPoint,4) zCoordsPositionMatrixFlex(getPoint,1)],'.k-'.	
480	[xCoordsPositionMatrixFlex(getPoint.2] FMasterXFlex(getPoint.1)], [vCoordsPositionMatrixFlex(getPoint.2] FMasterYFlex(getPoint.1)], [.*.*')	
481		
482	app.UIAxesEcont.CameraPosition = [0.05.10.075]:	
483	app_UIAxesFront_CameraTarget = [0,05.0,-0,075]:	
484		
485	nlot3(ann.UTAyesSide_[masterPoint4E]ev(getPoint_1) mater4(1)]. [masterPoint4E]ev(getPoint_2) mater4(2)]. [masterPoint4E]ev(getPoint_3) mater4(3)]. wirefolor	
486	[matterDoint#Flex(setPoint 1) motor#(1)] [matterDoint#Flex(setPoint 2) motor#(2)] [matterDoint#Flex(setPoint 3) motor#(3)] wireColor	
487	[masterPointCf]ex(getPoint.1) materPointCf)ex(getPoint.2) materPointCf)ex(getPoint.2) materPointCf)ex(getPoint.3) wige(a)or	
488	[macterPointFlev(atPoint]] motor[Nev()]] [macterPointFlev(atPoint 2) motor[Nev(2)] [macterPointFlev(atPoint 3) motor[Nev(3]] wirefolor	
489	[martarDointFlav(mtDoint]) motor(Y(1)) [martarDointFlav(mtDoint 2) motor(Y(2)] [martarDointFlav(mtDoint Flav(mtDoint Flav(
498	[macterPointDF]ev(etPoint]) motorD(1) [macterPointDF]ev(etPoint]) [materPointDF]ev(etPoint] wireColor	
491	macterpointFFLav(setPoint) motorF(1) [macterPointFLav(setPointFLav(setPointFFLav(setPointFLav(setPoint)) motorF(1)]] [macterPointFLav(setPointFLav(s	
492	[matterPointFFlex(setPoint.]) motorPoy(1)]. [matterPointFlex(setPoint.2) motorPoy(2)]. [matterPointFFlex(setPoint.3)], wiseGalor,	
493	[macterDointFF]ev(atDoint 1) motorFNev(1)] [macterDointFF]ev(atDoint 2) motorFNev(2)] [macterDointFF]ev(atDoi	
494	[mattrbointFiles(getAnis)] motorF2(11). [mattrbointFiles(getAnis)] [mattrbointFiles(getAnis)] mattrbointFiles(getAnis)]	
495	[VFoordeDecitionNatrivE]av(aetDoint 1) vFoordeDecitionNatrivE]av(aetDoint 2) vFoordeDecitionNatrivE]av(aetDoint 3) vFoordeDecitionNatrivE]av(aetDoint 1)]	
496	WoordsPositionMatrix[]ex(ertPoint_1) wCoordsPositionMatrix[]ex(ertPoint_5) wCoordsPositionMatrix[]ex(ertPoint_3) wCoordsPositionMatrix[]ex(ertPoint_1)	
497	[zCoordsPositionNatrixFlex(eetPoint.1] zCoordsPositionNatrixFlex(eetPoint.2) zCoordsPositionNatrixFlex(eetPoint.3) zCoordsPositionNatrixFlex(eetPo	
498	[xCoordsPositionNatrixE]ex(eetPoint.2) FMasterXE]ex(eetPoint.1)]. [xCoordsPositionNatrixE]ex(eetPoint.2) FMasterZE]ex(eetPoint.1)]. [.k.]	
499		
500	ann.UTAxesSide.CameraPosition = [-1,0,-0,075]:	
501	app.ULAxesSide.CameraTarset = [0,0,-0,075];	
502		
503	plot3(app.UIAxesTop.[masterPointAFlex(getPoint,1) motorA(1)], [masterPointAFlex(getPoint,2) motorA(2)], [masterPointAFlex(getPoint,3) motorA(3)], wireColor,	
504	[masterPointBFlex/getPoint.1] motorB(1)]. [masterPointBFlex/getPoint.2] motorB(2)]. [masterPointBFlex/getPoint.3] motorB(3)]. wireColor	
505	[masterPointCflex/getPoint.1] motorCPosY(1)]. [masterPointCflex(getPoint.2] motorCPosY(2)]. [masterPointCflex(getPoint.3] motorCPosY(3)], wireColor,	
506	[masterPointCFlex(getPoint,1) motorCNegY(1)], [masterPointCFlex(getPoint,2) motorCNegY(2)], [masterPointCFlex(getPoint,3) motorCNegY(3)], wireColor,	
507	[masterPointCflex(getPoint.1) motorCX(1)], [masterPointCflex(getPoint.2) motorCX(2)], [masterPointCflex(getPoint.3) motorCX(3)], wireColor,	
508	[masterPointDFlex(getPoint_1] motorD(1)], [masterPointDFlex(getPoint_2] motorD(2)], [masterPointDFlex(getPoint_3] motorD(3)], wireColor,	
509	[masterPointEFlex(getPoint,1) motorE(1)], [masterPointEFlex(getPoint,2) motorE(2)], [masterPointEFlex(getPoint,3) motorE(3)], wireColor,	
510	[masterPointFFlex(getPoint,1) motorFPosY(1)], [masterPointFFlex(getPoint,2) motorFPosY(2)], [masterPointFFlex(getPoint,3) motorFPosY(3)], wireColor,	
511	[masterPointFflex(getPoint,1) motorFMegY(1)], [masterPointFflex(getPoint,2) motorFMegY(2)], [masterPointFflex(getPoint,3) motorFMegY(3)], wireColor,	
512	[masterPointFFlex(getPoint,1) motorFZ(1)], [masterPointFFlex(getPoint,2) motorFZ(2)], [masterPointFFlex(getPoint,3) motorFZ(3)], wireColor,	
513	[xCoordsPositionMatrixFlex(getPoint,1) xCoordsPositionMatrixFlex(getPoint,2) xCoordsPositionMatrixFlex(getPoint,3) xCoordsPositionMatrixFlex(getPoint,4) xCoordsPositionMatrixFlex(getPoint,1)]	
514	[vCoordsPositionMatrixFlex(getPoint,1) vCoordsPositionMatrixFlex(getPoint,2) vCoordsPositionMatrixFlex(getPoint,3) vCoordsPositionMatrixFlex(getPoint,4) vCoordsPositionMatrixFlex(getPoint,1)]	
515	[zCoordsPositionMatrixFlex(getPoint,1) zCoordsPositionMatrixFlex(getPoint,2) zCoordsPositionMatrixFlex(getPoint,3) zCoordsPositionMatrixFlex(getPoint,4) zCoordsPositionMatrixFlex(getPo	
516	[xCoordsPositionMatrixFlex(getPoint,2) FMasterXFlex(getPoint,1)], [yCoordsPositionMatrixFlex(getPoint,2) FMasterZFlex(getPoint,1)], '.k-')	
517		
518	app.UIAxesTop.CameraPosition = [0.05,0,1];	+
519		

		_
518	app.UIAxesTop.CameraPosition = [0.05,0,1];	
519	app.UIAxesTop.CameraTarget = [0.05,0,0];	
520		
521	elseif app.HumeralFlexionSlider.Value == 0 && app.HumeralAbductionSlider.Value==0	
522	getPoint = round((round(app.HumeralFlexionSlider.Value,1)/0.1)+1);	
523		
524	if value == 1	
525	wireColor = 'r';	
526	elseif value == 0	
527	wireColor = 'w:square';	
528	end	
529		
530		
531	plot3(app.UIAxesAngled,[masterPointAFlex(getPoint,1)] motorA(1)], [masterPointAFlex(getPoint,2) motorA(2)], [masterPointAFlex(getPoint,3) motorA(3)], wireColor,	
532	[masterPointBFlex(getPoint,1) motorB(1)], [masterPointBFlex(getPoint,2) motorB(2)], [masterPointBFlex(getPoint,3) motorB(3)], wireColor,	
533	[masterPointCFlex(getPoint,1) motorCPosY(1)], [masterPointCFlex(getPoint,2) motorCPosY(2)], [masterPointCFlex(getPoint,3) motorCPosY(3)], wireColor,	
534	[masterPointCFlex(getPoint,1) motorCNegY(1)], [masterPointCFlex(getPoint,2) motorCNegY(2)], [masterPointCFlex(getPoint,3) motorCNegY(3)], wireColor,	
535	[masterPointCFlex(getPoint,1) motorCX(1)], [masterPointCFlex(getPoint,2) motorCX(2)], [masterPointCFlex(getPoint,3) motorCX(3)], wireColor,	
536	[masterPointDFlex(getPoint,1) motorD(1)], [masterPointDFlex(getPoint,2) motorD(2)], [masterPointDFlex(getPoint,3) motorD(3)], wireColor,	
537	[masterPointEFlex(getPoint,1) motorE(1)], [masterPointEFlex(getPoint,2) motorE(2)], [masterPointEFlex(getPoint,3) motorE(3)], wireColor,	
538	[masterPointFFlex(getPoint,1) motorFPosy(1)], [masterPointFflex(getPoint,2) motorFPosy(2)], [masterPointFFlex(getPoint,3) motorFPosy(3)], wireColor,	
539	[masterPointFFlex(getPoint,1) motorFNegY(1)], [masterPointFFlex(getPoint,2) motorFNegY(2)], [masterPointFFlex(getPoint,3) motorFNegY(3)], wireColor,	
540	[masterPointFFlex(getPoint,1) motorFZ(1)], [masterPointFFlex(getPoint,2) motorFZ(2)], [masterPointFFlex(getPoint,3) motorFZ(3)], wireColor,	
541	[xCoordsPositionNatrixFlex(getPoint,1) xCoordsPositionNatrixFlex(getPoint,2) xCoordsPositionNatrixFlex(getPoint,3) xCoordsPositionNatrixFlex(getPoint,2),	
542	[yCoordsPositionMatrixFlex(getPoint,1) yCoordsPositionMatrixFlex(getPoint,2) yCoordsPositionMatrixFlex(getPoint,3) yCoordsPositionMatrixFlex(getPoint,4) yCoordsPositionMatrixFlex(getPoint,1)],	÷.,
543	[zCoordsPositionNatrixFlex(getPoint,1) zCoordsPositionNatrixFlex(getPoint,2) zCoordsPositionNatrixFlex(getPoint,3) zCoordsPositionNatrixFlex(getPoint,4) zCoordsPositionNatrixFlex(getPoint,1)],'.k	
544	[xCoordsPositionNatrixFlex(getPoint,2) FMasterXFlex(getPoint,1)], [yCoordsPositionMatrixFlex(getPoint,2) FMasterYFlex(getPoint,1)], [xCoordsPositionMatrixFlex(getPoint,2) FMasterZFlex(getPoint,1)], '.k-')	
545		
546		
547	plot3(app.UIAxesFront,[masterPointAFlex(getPoint,1) motorA(1)], [masterPointAFlex(getPoint,2) motorA(2)], [masterPointAFlex(getPoint,3) motorA(3)], wireColor,	
548	[masterPointBFlex(getPoint,1) motorB(1)], [masterPointBFlex(getPoint,2) motorB(2)], [masterPointBFlex(getPoint,3) motorB(3)], wireColor,	
549	[masterPointCFlex(getPoint,1) motorCPosY(1)], [masterPointCFlex(getPoint,2) motorCPosY(2)], [masterPointCFlex(getPoint,3) motorCPosY(3)], wireColor,	
550	[masterPointCFlex(getPoint,1) motorCNegY(1)], [masterPointCFlex(getPoint,2) motorCNegY(2)], [masterPointCFlex(getPoint,3) motorCNegY(3)], wireColor,	
551	[masterPointCFlex(getPoint,1) motorCX(1)], [masterPointCFlex(getPoint,2) motorCX(2)], [masterPointCFlex(getPoint,3) motorCX(3)], wireColor,	
552	[masterPointDFlex(getPoint,1) motorD(1)], [masterPointDFlex(getPoint,2) motorD(2)], [masterPointDFlex(getPoint,3) motorD(3)], wireColor,	
553	[masterPointEFlex(getPoint,1) motorE(1)], [masterPointEFlex(getPoint,2) motorE(2)], [masterPointEFlex(getPoint,3) motorE(3)], wireColor,	
554	[masterPointFlex(getPoint,1) motorFPosY(1)], [masterPointFlex(getPoint,2) motorFPosY(2)], [masterPointFFlex(getPoint,3) motorFPosY(3)], wireColor,	
555	[masterPointFflex(getPoint,1) motorFNegY(1)], [masterPointFFlex(getPoint,2) motorFNegY(2)], [masterPointFFlex(getPoint,3) motorFNegY(3)], wireColor,	
556	[masterPointFflex(getPoint,1) motorFZ(1)], [masterPointFflex(getPoint,2) motorFZ(2)], [masterPointFflex(getPoint,3) motorFZ(3)], wireColor,	
557	[xCoordsPositionNatrixFlex(getPoint,1) xCoordsPositionNatrixFlex(getPoint,2) xCoordsPositionNatrixFlex(getPoint,5) xCoordsPositionNatrixFlex(getPoint,3) xCoordsPositionNatrixFlex(getPoint,1)],	÷.,
558	[yCoordsPositionNatrixFlex(getPoint,1) yCoordsPositionNatrixFlex(getPoint,2) yCoordsPositionNatrixFlex(getPoint,3) yCoordsPositionNatrixFlex(getPoint,4) yCoordsPositionNatrixFlex(getPoint,1)],	÷.,
559	[zCoordsPositionMatrixFlex(getPoint,1) zCoordsPositionMatrixFlex(getPoint,2) zCoordsPositionMatrixFlex(getPoint,3) zCoordsPositionMatrixFlex(getPoint,4) zCoordsPositionMatrixFlex(getPoint,1)],'.k	
560	[xCoordsPositionMatrixFlex(getPoint,2) FMasterXFlex(getPoint,1)], [yCoordsPositionMatrixFlex(getPoint,2) FMasterYFlex(getPoint,1)], [zCoordsPositionMatrixFlex(getPoint,2) FMasterZFlex(getPoint,2)], '.k-')	
561		× 11

562	app.UIAxesFront.CameraPosition = [0.05,1,-0.075];	
563	app.UIAxesFront.CameraTarget = [0.05,0,-0.075];	
564		
565	plot3(app.UIAxesSide,[masterPointAFlex(getPoint,1) motorA(1)], [masterPointAFlex(getPoint,2) motorA(2)], [masterPointAFlex(getPoint,3) motorA(3)], wireColor,	
566	[masterPointBFlex(getPoint,1) motor8(1)], [masterPointBFlex(getPoint,2) motor8(2)], [masterPointBFlex(getPoint,3) motor8(3)], wireColor,	
567	[masterPointCflex(getPoint,1) motorCPosY(1)], [masterPointCflex(getPoint,2) motorCPosY(2)], [masterPointCflex(getPoint,3) motorCPosY(3)], wireColor,	
568	[masterPointCFlex(getPoint,1) motorCNegY(1)], [masterPointCFlex(getPoint,2) motorCNegY(2)], [masterPointCFlex(getPoint,3) motorCNegY(3)], wireColor,	
569	[masterPointCflex(getPoint,1) motorCX(1)], [masterPointCflex(getPoint,2) motorCX(2)], [masterPointCflex(getPoint,3) motorCX(3)], wireColor,	
570	[masterPointDFlex(getPoint,1) motorD(1)], [masterPointDFlex(getPoint,2) motorD(2)], [masterPointDFlex(getPoint,3) motorD(3)], wireColor,	
571	[masterPointEFlex(getPoint,1) motorE(1)], [masterPointEFlex(getPoint,2) motorE(2)], [masterPointEFlex(getPoint,3) motorE(3)], wireColor,	
572	[masterPointFflex(getPoint,1) motorFPosY(1)], [masterPointFflex(getPoint,2) motorFPosY(2)], [masterPointFflex(getPoint,3) motorFPosY(3)], wireColor,	
573	[masterPointFflex(getPoint,1) motorFNegY(1)], [masterPointFflex(getPoint,2) motorFNegY(2)], [masterPointFflex(getPoint,3) motorFNegY(3)], wireColor,	
574	[masterPointFFlex(getPoint,1) motorFZ(1)], [masterPointFFlex(getPoint,2) motorFZ(2)], [masterPointFFlex(getPoint,3) motorFZ(3)], wireColor,	
575	[xCoordsPositionMatrixFlex(getPoint,1) xCoordsPositionMatrixFlex(getPoint,2) xCoordsPositionMatrixFlex(getPoint,3) xCoordsPositionMatrixFlex(getPoint,4) xCoordsPositionMatrixFlex(getPoint,3) xCoordsPositionMatrixFlex(getPoint,4) xCoordsPositionMatrixFlex(getPo	
576	[yCoordsPositionMatrixFlex(getPoint,1) yCoordsPositionMatrixFlex(getPoint,2) yCoordsPositionMatrixFlex(getPoint,3) yCoordsPositionMatrixFlex(getPoint,4) yCoordsPositionMatrixFlex(getPoint,3) yCoordsPositionMatrixFlex(getPoint,4) yCoordsPositionMatrixFlex(getPoint,2) yCoordsPositionMatrixFlex(getPoint,3) yCoordsPositionMatrixFlex(getPoint,4) yCoordsPositionMatrixFlex(getPoint,3) yCoordsPositionMatrixFlex(getPoint,4) yCoordsPositionMatrixFlex(getPo	
577	[zCoordsPositionMatrixFlex(getPoint,1) zCoordsPositionMatrixFlex(getPoint,2) zCoordsPositionMatrixFlex(getPoint,3) zCoordsPositionMatrixFlex(getPoint,1)],'.+', .	
578	[xCoordsPositionMatrixFlex(getPoint,2) FMasterXFlex(getPoint,1)], [yCoordsPositionMatrixFlex(getPoint,1)], [zCoordsPositionMatrixFlex(getPoint,2) FMasterZFlex(getPoint,1)], '.k-')	
579		
580	app.UIAxesSide.CameraPosition = [-1,0,-0.075];	
581	app.UIAxesSide.CameraTarget = [0,0,-0.075];	
582		
583	plot3(app.UIAxesTop,[masterPointAFlex(getPoint,1) motorA(1)], [masterPointAFlex(getPoint,2) motorA(2)], [masterPointAFlex(getPoint,3) motorA(3)], wireColor,	
584	[masterPoint8Flex(getPoint,1) motor8(1)], [masterPoint8Flex(getPoint,2) motor8(2)], [masterPoint8Flex(getPoint,3) motor8(3)], wireColor,	
585	[masterPointCFlex(getPoint,1) motorCPosY(1)], [masterPointCFlex(getPoint,2) motorCPosY(2)], [masterPointCFlex(getPoint,3) motorCPosY(3)], wireColor,	
586	[masterPointCFlex(getPoint,1) motorCNegV(1)], [masterPointCFlex(getPoint,2) motorCNegV(2)], [masterPointCFlex(getPoint,3) motorCNegV(3)], wireColor,	
587	[masterPointCFlex(getPoint,1) motorCX(1)], [masterPointCFlex(getPoint,2) motorCX(2)], [masterPointCFlex(getPoint,3) motorCX(3)], wireColor,	
588	[masterPointDFlex(getPoint,1) motorD(1)], [masterPointDFlex(getPoint,2) motorD(2)], [masterPointDFlex(getPoint,3) motorD(3)], wireColor,	
589	[masterPointEflex(getPoint,1) motorE(1)], [masterPointEflex(getPoint,2) motorE(2)], [masterPointEflex(getPoint,3) motorE(3)], wireColor,	
590	[masterPointFlex(getPoint,1) motorFPosY(1)], [masterPointFFlex(getPoint,2) motorFPosY(2)], [masterPointFFlex(getPoint,3) motorFPosY(3)], wireColor,	
591	[masterPointFlex(getPoint,1) motorFNegY(1)], [masterPointFFlex(getPoint,2) motorFNegY(2)], [masterPointFFlex(getPoint,3) motorFNegY(3)], wireColor,	
592	[masterPointFlax(getPoint,1) motorFZ(1)], [masterPointFlax(getPointF2(2)], [masterPointFFlax(getPoint,3) motorFZ(3)], wireColor,	
593	[X.CordsPositionmatrix+iex[getPoint,1] X.CordsPositionmatrix+iex[getPoint,2] X.CordsPositionmatrix+iex[getPoint,3] X.CordsPositionmatrix+iex[getPoint,4] X.CordsPositionmatrix+iex[getPo	
594	[ycoordsPositionMatrixiex(getPoint, 1) ycoordsPositionMatrixiex(getPoint, 2) ycoordsPositionMatrixiex(getPoint, 3) ycoordsPositionMatrixiex(getPoint, 4) ycoordsPositionMatrixiex(getPoi	1.00
595	[LoordsPositionmatrixrick(getPoint,1) zloordsPositionmatrixrick(getPoint,1)], .k-, .	
596	[X.CordsPositionMatrixFiex(getPoint,2) FMasterXFiex(getPoint,1)], [y.CordsPositionMatrixFiex(getPoint,1)], [z.CordsPositionMatrixFiex(getPoint,2) FMasterZFiex(getPoint,1)], '.x-')	- 11
597		
530	$p_{1} = p_{1} = p_{2} = p_{1} = p_{2} = p_{2$	
600	app.uixxesiop.Lameraiarget = [u.us,u,u];	
601		
602		
603		
684		-
0.04		

605	% Component initialization
606 📋	methods (Access = private)
607	
608	% Create UIFigure and components
609 📄	function createComponents(app)
610	
611	% Create UIFigure and hide until all components are created
612	app.UIFigure = uifigure('Visible', 'off');
613	app.UIFigure.Position = [100 100 786 588];
614	app.UIFigure.Name = 'MATLAB App';
615	
616	% Create UIAxesAngled
617	app.UIAxesAngled = uiaxes(app.UIFigure);
618	title(app.UIAxesAngled, 'Angled View')
619	app.UIAxesAngled.XLim = [-0.1 0.574];
620	app.UIAxesAngled.YLim = [-0.49 0.49];
621	app.UIAxesAngled.ZLim = [-0.48 0.48];
622	app.UIAxesAngled.Position = [1 326 399 239];
623	
624	% Create UIAxesFront
625	app.UIAxesFront = uiaxes(app.UIFigure):
626	title(app.UIAxesFront, 'Front View')
627	app.UIAxesFront.XLim = [-0.1 0.574];
628	app.UIAxesFront.YLim = [-0.49 0.49];
629	app.UIAxesFront.ZLim = [-0.48 0.48];
630	app.UTAxesFront.Position = [399 326 388 239]:
631	
632	% Create UTavesSide
633	ann IITAverSide = uiaver(ann IITEigure).
634	title(ann UTAvesSide, 'Side View')
635	ann IITáverSide View = [0.0]
636	ann UTAxesSide XLim = [-0.1.0.574]:
637	ann IITávesSide VI (m = [-0.49.0.49])
638	ann.UTAxesSide.71 in = [-0.48 0.48];
639	ann UTAverSide Porition = [20 107 380 220];
640	
641	S Crasta UTAverTon
642	ann IITävesTon = ulaves(ann IITE(gura))
643	title(ann UTAverTon (Ton View))
644	ann UTavesTon CameraPosition = [0.0.1]
645	app UTAxesTop CameraTarget = [0.0.0];
646	app UTAxesTop XLim = [-0, 1, 0, 574];
647	app. UTAvesTop. Vide = [-0.10.3/4];
	approximestop.term = [-0.49 0.49];

647	ann-IITAxesTon, YI im = [-0,49,0,49]:	
648	app.UIAxesTop.ZLim = [-0.48 0.48]:	^
649	ann.1174x+xTon.Postfion = [370 107 387 220]:	
650	abharweitabh agrean - Eile rei Par reit	
651	% Create HumeralAbductionSliderLabel	
652	ann HumeralabductionSliderlabel = uilabel(ann HTFigure))	
653	ann HumanalahdurtinnSilderlahal Honizontalahanset - (-abbi' -	
654	app interest about consider tables into internal generation of a constraints and the const	
655	and Human labeled consistent and the set of	
656	apprimate achoust construct = nomeral mount construction ;	
657	% Crasta NumaralAbdurtionSliden	
658	a trade numeralbductionslate:	
659	approximate debate constraints $-$ default (approximate); and human algorithm (approximate);	
660	opprover consistent identification is not a second of the second and the second of the second se	
661	app nume arbourtionsider valetions (32 or 32 3)	
662	approaler approach for a construction of the state of the	
663	% Craste NumeralElavionElidarishal	
664		
665	app.HumerarlexionsiderLader = Gladeriapp.Urrigure;;	
666	apprinted bit accompanies above for a constant against - (agins)	
667	approximation of the state of the state $-[227 + 32 + 22]$	
669	opprimit of texton site toot the texton y	
669	% Create HumanalFlavionSlider	
670	ann HumenalFlavinolider = uiclider/ann HTF(sure):	
671	ϕ_{P} model of a kereoristic (ϕ_{P} (ϕ_{P}) ($\phi_{$	
672	apprinde ariestonistie	
673	ann Human Flavinn State Change and the second and t	
674	obhunanci at teronotitaci u controli - [oo, oo oot o])	
675	% Create ShowWiresCheckBox	
676	app_ShowWiresCheckBox = uicheckbox(app_UIFigure):	
677	ann-ShowdiresCheckBox-ValueChangedFrn = createCallbackFrn(ann, @ShowdiresCheckBoxValueChanged, true):	
678	ann. ShouidiresCheckBox. Text = 'Show Wires':	
679	app.ShowWiresCheckBox.Position = [104 45 86 43]:	
680		
681	% Show the figure after all components are created	
682	app.UIFigure.Visible = 'on':	
683 -	end	
684 -	end	
685		
686	% App creation and deletion	
687 🗖	methods (Access = public)	
688		
689	% Construct app	
690 🗄		
T.		•

6	1 app.HumeralFlexionSlider.Limits = [0 120];	
6	2 app.HumeralFlexionSlider.ValueChangedFcn = createCallbackFcr	n(app, @HumeralFlexionSliderValueChanging, true);
6	app.HumeralFlexionSlider.Position = [337 54 381 3];	
6	4	
6	5 % Create ShowWiresCheckBox	
6	6 app.ShowWiresCheckBox = uicheckbox(app.UIFigure);	
6	7 app.ShowWiresCheckBox.ValueChangedFcn = createCallbackFcn(a)	pp, @ShowWiresCheckBoxValueChanged, true);
6	<pre>8 app.ShowWiresCheckBox.Text = 'Show Wires':</pre>	
6	9 app.ShowWiresCheckBox.Position = [104 45 86 43];	
6	0	
6	1 % Show the figure after all components are created	
6	2 app.UIFigure.Visible = 'on';	
6	3 - end	
6	4 - end	
6	5	
6	6 % App creation and deletion	
6	7 methods (Access = public)	
6	8	
6	9 % Construct app	
6	<pre>0 function app = shoulderRotationApp</pre>	
6	1	
6	2 % Create UIFigure and components	
6	3 createComponents(app)	
6	4	
6	5 % Register the app with App Designer	
6	6 registerApp(app, app.UIFigure)	
6	7	
6	8 % Execute the startup function	
6	9 runStartupFcn(app, @startupFcn)	
7	0	
70	<pre>if nargout == 0</pre>	
7	2 clear app	
7	3 end	
70	4 - end	
7	5	
70	6 % Code that executes before app deletion	
71	7 function delete(app)	
70	8	
7	9 % Delete UIFigure when app is deleted	
7:	0 delete(app.UIFigure)	
7	1 end	
7:	2 end	
7:	3 ^L end	
	4	

8.0 References

- *Anatomical Terminology*. Anatomical Terminology | SEER Training. (n.d.). Retrieved March 14, 2024, from <u>https://training.seer.cancer.gov/anatomy/body/terminology.html</u>
- Avin KG, Bloomfield SA, Gross TS, Warden SJ. Biomechanical aspects of the muscle-bone interaction. Curr Osteoporos Rep. 2015 Feb;13(1):1-8. Doi: 10.1007/s11914-014-0244-x. PMID: 25515697; PMCID: PMC4306629.
- Bagg, S. D., & Forrest, W. J. (2016, March 4). A biomechanical analysis of scapula rotation during arm abduction in the scapula plane. Academia.edu. Retrieved March 14, 2024, from https://www.academia.edu/22798706/A_biomechanical_analysis_of_scapula_rotation_duri ng_arm_abduction_in_the_scapula_Plane?bulkDownload=thisPaper-topRelatedsameAuthor-citingThis-citedByThis-secondOrderCitations&from=cover_page
- Bolsterlee, B., Veeger, D. H. E. J., & Chadwick, E. K. (2013). Clinical applications of musculoskeletal modelling for the shoulder and upper limb. Medical & Biological Engineering & Computing, 51(9), 953–963. <u>https://doi.org/10.1007/s11517-013-1099-5</u>
- Bones of the shoulder. jointspecialists.org. (2017, March 27). Retrieved March 14, 2024, from https://jointspecialists.org/bones-of-the-shoulder/
- Chua, Chee Kai, and Wai Yee Yeong. "Impact Toughness." *Impact Toughness an Overview* | *ScienceDirect Topics*, www.sciencedirect.com/topics/engineering/impact-toughness. Accessed 14 Mar. 2024.
- Deane, Ella, et al. Worcester, Massachusetts, 2022, Anatomically Accurate Motorized Shoulder Model with Scapula Movement.
- Engelhardt, C., Camine , M., V., Ingram D., Müllhaupt P., Farron A., Pioletti D., Terrier A. (April 4, 2014). Comparison of an EMG-based and a stress-based method to predict shoulder muscle forces, Computer Methods in Biomechanics and Biomedical Engineering. Taylor& Francis Online Journal, 1272-1279, DOI: 10.1080/10255842.2014.899587

- Garofalo, Pietro. "Clavicular Component of Scapulo-Humeral Rhythm." *ResearchGate*, www.researchgate.net/figure/Clavicular-Component-of-Scapulo-humeral-Rhythm-Third-III -top-phase-of-scapulohumeral_fig15_24019863. Accessed 14 Mar. 2024.
- Igoshin, Nikita. Worcester, Massachusetts, 2022, Producing Accurate Relative Motion in a Shoulder Model.
- Mathoma. (July 26, 2016). "3D Rotations in General: Rodrigues Rotation Formula and Quaternion Exponentials" *YouTube*, https://www.youtube.com/watch?v=q-ESzg03mQc. Accessed 12 Mar. 2024.
- McEvilly, Fiona, et al. Worcester, Massachusetts, 2023, *Realistic Shoulder Model with Soft Tissue Attachments*.
- O'Leary S, Christensen SW, Verouhis A, Pape M, Nilsen O, McPhail SM. Agreement between physiotherapists rating scapular posture in multiple planes in patients with neck pain: Reliability study. Physiotherapy. 2015 Dec;101(4):381-8. doi: 10.1016/j.physio.2015.01.005. Epub 2015 Jan 25. PMID: 25749493.

"Online Learning for Physiotherapists." Physiotutors, 21 Feb. 2024, Physiotutors.com/.

OpenStax. "University Physics Volume 1." *10.6 Torque* | *University Physics Volume 1*, 3 Aug. 2016, courses.lumenlearning.com/suny-osuniversityphysics/chapter/10-6-torque/.

Physics LibreTexts, Libretexts, 12 Mar. 2024,

phys.libretexts.org/Bookshelves/Conceptual_Physics/Introduction_to_Physics_(Park)/03% 3A_Unit_2-_Mechanics_II_- Energy_and_Momentum_Oscillations_and_Waves_Rotation _and_Fluids/06%3A_Rotation/6.03%3A_Dynamics_of_Rotational_Motion-_Rotational_In ertia.

"Pololu Robotics and Electronics." *Pololu Robotics & Electronics*, www.pololu.com/. Accessed 14 Mar. 2024.

- "Programming with Matlab." MATLAB & Simulink,
 - www.mathworks.com/products/matlab/programming-with-matlab.html. Accessed 14 Mar. 2024.
- Shelton, Stephen M., and William H. Swanger. *Fatigue Properties of Steel Wire*, nvlpubs.nist.gov/nistpubs/jres/14/jresv14n1p17_A1b.pdf. Accessed 14 Mar. 2024.
- Shojaei, Amir Mohammad. "Connect Two Arduino Boards Using I2C Communication Protocol." *Electropeak*, 28 May 2023, electropeak.com/learn/connect-two-arduino-boards-using-i2c-communication-protocol/.
- Singh, A. P. (n.d.) Muscles of Shoulder Region. Bones and Spine. https://boneandspine.com/muscles-of-shoulder/
- "StepperOnline." STEPPERONLINE, StepperOnline.com/. Accessed 14 Mar. 2024.

"Tutorials." Arduino, www.arduino.cc/en/Tutorial/HomePage. Accessed 14 Mar. 2024.

- Ultimaker Nylon: Technical Data Sheet. (2022, 20 Apr). Ultimaker. Retrieved March 14, 2024, from https://ultimaker.com/materials/nylon
- Ultimaker TPU 95A: Technical Data Sheet. (2022, 29 Apr). Ultimaker. March 14, 2024, from https://ultimaker.com/materials/tpu-95a
- "The Ultimate Resource for Healthcare Professionals & Medical Students." *TeachMeAnatomy*, teachmeanatomy.info/. Accessed 14 Mar. 2024.
- "Unleashing the Power of Programming." *An Introduction To Coding*, www.arduino.cc/education/an-introduction-to-coding/. Accessed 14 Mar. 2024.
- Yabata, Kazuyuki, and Tsutomu Fukui. "Characteristics of the Scapula Movement during Shoulder Elevation Depend on Posture." *Journal of Physical Therapy Science*, U.S. National Library of Medicine, July 2022, www.ncbi.nlm.nih.gov/pmc/articles/PMC9246406/.