Team Tentacle 🐙

ME 102B Fall 2023 Darren Suen, Zach Tam, Aalaya Wudaru

Opportunity:

Our journey began with the opportunity to craft a robotic system featuring seamless, intuitive control. We hoped to blend complex software and precision mechanical control systems to make the movement user centric and instinctual. While brainstorming applications for the robotic control system, we became increasingly interested in the applications of soft robotics such as search and rescue as well as surgery. Driven by soft robotics' origins in biomimicry we couple our interests in soft robotics and intuitive control to design and construct an intuitively controlled bio-mechanism Therefore, our project ultimately took on the fabrication of a robotic arm that behaved akin to an octopus' tentacle: fluid and versatile.

High Level Strategy:

The initial desired functionality was to have an octopus robot that flexed and/or moved its tentacles in response to hand motions as the external controller. We were quite ambitious and expected to have multiple tentacle arms and utilize various sensors to capture and transmit the motions of a hand and fingers as a signal for robotic motion. However, upon further research and consultation we scaled down our strategy to fit within financial and time constraints.

Our high level strategy then became to utilize three degrees of actuation controlled by three "tendons". The string lengths would vary by motor-capstan rotation, thereby creating the fluid motion of the tentacle. The rigidity of the tentacle arm would be maintained by disks and springs between them. While initial functionality goals included sensors picking up hand motions, we decided to go for a joystick controller for our arm and limit additional functionality to an LED light show and preprogrammed "dance sequence". The 2-degree joystick was mapped using a mathematical function in our code, and we mapped x and y coordinates of the joystick input to radial positions of our arm. We implemented position control to have soft-stops so that the tentacle did not bend beyond its fracture point. Our arm meets our soft robotics specifications and moves like an octopus tentacle while also being intuitively controlled, but it did not reach the ambitious goals of multiple arms due to budgeting and timeline.

Integrated Device:

Below, we show our motor assembly and housing, circuitry, and final assembly.





Music Board and Op Amps for Light Strips (under motor driver cables):



Critical Design Decisions and Calculations:

The table below shows the constants we had given the compression spring and rotary shaft sizes that were roughly forced on us based on price and size.

Specifically, we needed springs that were relatively short for their diameter (and not too expensive), and these were the weakest ones we could find of that type. This is because shorter and wider springs are typically stronger.

As for the rotary shafts, we picked the diameter based on the size of the cheap shaft couplers and bearings we found on Amazon.

Compression Spring		
k, spring rate	18	lb/in
Length	1.5	in
Compressed Length @ Max load	0.51	in
Max Load	42	lb
Rotary Shaft		
Diameter	0.315	in
Circumference	1.9792017	in

We then compute the equivalent spring constant for 6 springs in series:

 $k_{eq} = k / 6 = 3 \text{ lb/in}$

The free length of the tentacle is:

 $L_{\text{tot}}=6L=9 \text{ in}$

We assume that at most, the total tower would be compressed to 6 inches. In this state, the spring force would be:

 $|F_{max}| = k_{eq}^{*}(L_{tot} - 6) = 9 \text{ lb}$

At worst, this would all be held by one cable, with a tension of 9 lb. We used this maximum cable tension to determine which fishing line to buy. However, we ultimately opted for overkill since we didn't want the line to stretch and there was no obvious downside to doing so. We also used the tension estimate to determine which motors to use.

The associated motor torque would be:

 $\tau_{max} = |F_{max}|^* (d_{shaft} / 2) = 9 \text{ lb} * 0.1575 \text{ in} = 1.4175 \text{ lb}^* \text{in} = 16.33 \text{ kg}^* \text{mm}$

Initially, we looked at the Pololu metal gearmotor datasheets to determine the appropriate gearing to achieve this max torque with less than 60% PWM. This ended up being the 30:1 gear ratio model. However, these motors were very expensive. We presented these numbers to Tom, who lent us the appropriately sized motors and corresponding motor drivers that he had for free.

Circuit Diagram and State Transition Diagram:





Reflection:

One of the biggest lessons we learned from this project was the importance of proper software integration. While presenting our functionality demo, we had not yet added in a position based stop in the code for when the tentacle was bent to the point of fracture. When dealing with flexible soft robotics components, it is important that since the components aren't as rigid, the code is intuitive and accounts for this room to fail. We have now implemented a position based control system that will also rehome the tentacle after the user plays with the arm. By adding soft stops in our code we were able to maintain the simplicity and affordability of the arm build while achieving the same functionality. Additionally, we initially hand-machined the motor housing out of wood, something that proved to be a great, time consuming challenge. We highly recommend to all future groups to start on the housing design early so you can laser cut or water jet, as we eventually did to achieve a cleaner look. We learned a lot from our mistakes in this project, but each one made us better engineers!

Appendix I: Bill of Materials

Team Tentacle's	Purchase Portfolio							Total (Projected):	\$	181.31			Total (Spent):	\$	195.0	
Item Name	0.5 in Aluminum Rod	Used to lathe shafts and capstans	Serial Number / SKU 5267208	Price (e	 Quantity 	Ace	https://www.o	Notes	Subto	tal	Purchased?	Order Date	Purchased B	y: P	urchase loto	Link to Receipt
K&S 12 in. L X 0.5				\$ 4.	15 1	Hardware			\$	4.95		11/7/23	Aalaya Wudaru	- \$	4.9	https://drive.goo gle.com/file/d/li CIXQv92WYQ75 w0ico7zrt1giyrbY
Rod 1 pk .5 x 3,5 x 3 ft Poplar Hobby	Plywood Board	Used for first iteration of motor housing plates	728927310612			Home Depot	https://www.h	will need to replace with laser cut housing						+		aring
board				\$ 9.	8 1				\$	9.98		11/21/23	Aalaya Wudaru	- \$	9.9	3 0297LaYiCNOvp 1m6Bv0z49kmvsn Sdffo/view?usp= sharing
608 ZZ Ball Bearings(10PCS), 608ZZ Metal Double Shielded Miniature Deep Groove	60822 Ball Bearings	Used for motor housing: rotation of capstans	B07H83VV6B	\$ 63	10 1	Amozon, NAIVE BLUE	Wooden Dov	will need to replace with flanged bearings instead to be held in place in housing, as adviced by machine shop staff	\$	6.90		10/27/23	Darren . Suen	- \$	7.6	https://drive.goo gle.com/file/d/1 wcP_H7Qt_9k8C eqbKiWElaTZHQ
Skateboard Ball Bearings (8mm x 22mm x 7mm) SHNITPWR 60W	Universal Power Supply	Power supply to power motors	BOBBLSSLMB			Amazon,										Nxo3ks/view?us p=drive_link
Supply DC 3V 4V 4.5V 5V 6V 7V 7.5V 8V 9V 10V 11V 12V Adjustable Variable Power Adapter 100V-240V AC to DC Converter 1A 2A		unu syrns		\$ 22:	19 1	SHULFWK			\$	22.99	Ø	10/27/23	Darren Suen	- \$	25.3	https://drive.goo ale.com/file/d/1 wcP H7Ot 9k8C eabK/WEloTIHQ Nxo3ks/view?us p=drive_link
2.5A 3A 4A 5A with 14 Tips & Polarity Converter Saiper 5pcs	Flexible Shaft Couplers	To connect motor shafts to rotary	B07S8XJT4D			Amazon,	Amazon.com: S	HNITPWR 60W Universal Power Supply Do	_					+		
Hexible Couplings 6mm to 8mm Aluminum Allay Joint Connector Compatible with NEMA 17 Stepper Motors, RepRap 3D Printer or CNC Machine, 3D Printer Accessoria		shafts for power transmission		\$8.	19 1	Saiper	Amazon com: S	sing fore Elevible Countine from to Some to So	\$	8.99		10/27/23	Darren Suen	- 5	9.9	https://drive.gog gle.com/file/d/1 wcP_H7O19k8C egbKWEloTZHQ Nxo3ks/view?us p=drive_link
							Punacon.com. o									
uxcell F60822 Flanged Ball Bearing 8x22x7mm Dauble Metal Shielded (GCr15) Chrome Steel Flange Rig	Flanged 60822 Ball Bearings	To replace ball bearings in motor housing after consultation with machine shop staff and improving on project	B085DRNJ6R	\$ 10.	د ا ا	Amazon, uxcell		Replace unflanged ball bearings upon consultation with machine shop staff to be held in place in motor housing better	\$	10.41		12/3/23	Darren Suen	- \$	10.4	https://drive.gos gle.com/file/d/1 BacbWehYfmi2v TWZcstd.poShG 3tm_XU/view?us
Bearings 10pcs Russian Birch 12" v	Puerian Birch Playaad	For later cutting of mator bouring	- (hought in person			UCD CED	uxcell F608ZZ	Rought in person of the LIC Berkeley						+		p=drive_link
48" (1/4")/(6mm)	Kasan bion nywood	for laser coning of molor housing	at physical store)	\$7.	45 1	Materials Shop		CED Materials Shop in Wurster Shop. Online catalog can only be assessed through their bCourses site.	\$	7.45	y	12/4/23	Darren Suen	- \$	7.4	https://drive.goc gle.com/file/d/1 zzVsWk_MGeJQ c2hbwQc_fXM7 yQai743/view?u p=drive_link
3D Printed Discs	3D Printed Discs	Prototype to test disc and 3D printing clearance before sending for mass prints	12403463	\$ 1.	73 1	3DPrinterOS	-	Refer to attachment (files labelled with "1 print" means they have been printed ance and their print job priorings are attached); no receipt as 3DPrinterOS bills the student via CalCentral at the end of the semester Identify can be verified via school email on to right.	\$	1.93	M	11/3/23	Darren Suen	- \$	1.5	https://drive.goo gle.com/file/d/1 3 OFxmyEMjMR-kb uoEoi1DNuGhb wmBTfXY/viewSu sp=drive_link
3D Printed Discs	3D Printed Discs	Printing of multiple discs for tentacle arm; used to sandwich and hold down compression springs and let fish line pass through to control arm motion	12457297	\$ 3.	24 1	3DPrinterOS		Refer to attachment (files labelled with "1 print" means they have been printed once and their print job pricings are attached); no receipt as 3DPrinterOS bills the student via CalCentrol at the end of the semester Identify can be verified via school email on to einth	\$	3.94	M	11/11/23	Darren Suen	- \$	3.5	https://drive.gos gle.com/file/d/1 0ExmvEMiM8-kb uoEoi1DNuGhb wm81fXY/view9u sp=drive_link
3D Printed Discs	3D Printed Discs	Printing of multiple discs for tentacle arm: used to sandwich and hold down compression springs and lef fish line pass through to control arm motion	12569385	\$ 3.	74 1	3DPrinterOS	-	Refer to attachment (files labelled with "11 print" means they have been printed ance and their print jab priorings are attached); no receipt as 3DPrinterOS bills the student via CalCentral at the end of the semester Identify can be verified via school	44	3.94	M	11/19/23	Darren Suen	- \$	3.5	https://drive.goo gle.com/file/d/1 0FxmyEM/MR-kb uoEoi1DNuGhb wmB1KY/viewSu sp=drive_link
3D Printed Discs	3D Printed Discs	Printing of multiple discs for tentacle arm; used to sandwich and hold down compression springs and lef fish line pass through to control arm motion; improved version from previous prints	12797803	\$ 2.	38 1	3DPrinterOS	-	Refer to attachment (files labelled with "1 pint" means they have been printed once and their pint job prioring are attached); no receipt as 3DPinterOS bills the student via CalCentral at the end of the semester Identify can be verified via school	\$	2.88		12/11/23	Darren Suen	- \$	2.8	https://drive.goc gle.com/file/d/1 0FxmvEMiM8-kb uoEci1DNuGhb wm817X/view3, sp=drive_link
											1					
0.75in x 0.75in-R/L Hardwood	Wooden Dowel	supports for housing	73893780014	\$ 0.9		Home Depot			¢	2.94		11/21/22	Aalaya			https://drive.gr ogle.com/file/
18.8 Stainlass Staal	Saaliat Haad Sarau	To faster water and having	912924112			Holdestos Ca							Wudaru			Ovp1m6By0z49 kmysnSdffo/vie w
Socket Head Screw, M3 x 0.5 mm Thread, 8 mm Long, Packs of 100		together	712728112	\$ 5.	15 1	r	McMaster-Carr		\$	5.45		10/25/23	Zachary Tam	- \$	6.5	https://drive.go gle.com/file/d/1 melyYp1afQD/C 8AYR0d58z3iAa3 w0yZG/view?usr =drive_link
Steel Hex Nut, Medium-Strength, Class 8, M3 x 0.5 mm Thread, Packs of 100	Hex Nuts	To fasten system and housing together	90592A085	\$ 2.	52 1	McMaster-Ca r	https://www.n	2	\$	2.62		10/25/23	Zachary Tam	- 5	3.3	https://drive.go gle.com/file/d/1 meLYYp1ofQD0 8AYR0d58z3iAc2 w0yZG/view?us edrive_fink
Compression Spring, 1.5" Long, 0.975" OD, 0.831" ID, Packs of 6	Compression Springs	To be used in tentacle arm to allow for flexible bending	9657K522	\$ 11))9 2	MoMaster-Ca r	https://www.n	2	\$	22.18		10/25/23	Zachary Tam	- \$	28.4	https://drive.go ale.com/file/d/1 3 melvYp1gtQpC 8AYR0d58z3iAc w0yZG/view?us =drive_fink
Carbon Steel Set Screw Collar for 8 mm Shaft Diameter, DIN 705	Shaft Collars	To be used in motor housing tp ensure transmission system is held in place	6056N16	\$ 2.	7 6	McMaster-Ca r	https://www.n	1	\$	13.02	M	10/25/23	Zachary Tam	~ \$	16.0	https://drive.go/ gle.com/file/d/ mely/tp1gfQDr0 8AYR0d5Bz3iAq3 w0yZG/view%usr =drive_ink
Belleville Disc Springs for Ball Bearing Trade No. 608, 627 and El8, 12.300 mm ID, Packs of 10	Belleville Disc Sprigs	Used to reduce axial load and vibrations on transmission system for optimum performance	94065K42	\$ 4.	12 1	McMaster-Ca r	https://www.n	1	\$	4.42	M	10/25/23	Zachary Tam	- \$	5.4	https://drive.go/ gle.com/file/d/1 7 mel/Yp1afQD/0 8AYR0d58z3iAa3 w0/ZG/view?usr =drive_fink
											1					
Braided Fishing Line, Abrasion Resistant	401b braided fishing line	Tendons; the string that wrapped around the capstan		\$ 11.	01 1	Amazon	https://www.g	2	\$	11.01	V	11/15/23	Aalaya Wudaru		5 11.0	https://drive.go gle.com/file/d/ GAk59-tEoc.5g Zghtprh3X8Hgu XzSS/view?usp=
Dowel - 1/2"x48"	Wooden Dowel	housing supports	95624515511	\$ 2.)7 1	Home Depo			\$	2.07		12/8/23	Aalaya	~ <u>-</u>	\$ 2.0	haring https://drive.go ale.com/file/d/ ahmikR5wxVg_J
Joystick Sensor	Analogy Joystick Sensor	For arm control				Amazon	https://www.g	4					modulo			m1fvLwxv5Jaw6 40x58pM/siew3/ sp=sharing
Sensor JayStick Breakout Module for Arduino PS2 Raspberry Pi				\$ 6.	93				\$			11/7/23	Aalaya Wudaru	- 1	§ 6.	13

1074-1095 Spring Steel Ring Shim, 0.2mm Thick. 8mm ID, Packs of 50	Ring Shims	Transmission for 3 motors. Make sure shaft collars don't touch both the inner and outer race of the bearings.	98055A112	\$ 6.90	1	McMaster-Car r	https://www.m	\$	6.90	N	10/25/23	Zachary Tam	- \$	8.84	https://drive.goo ale.com/file/d/1 metyYp1afGDr0 8AYR0d5Bz3iAo3 w0yZG/view8usp adrive_link
1.5" Zinc 4pk Corner Brace	1.5° Corner Brace	for housing	30699153046	\$ 3.37	2	Home Depot	https://www.b	\$	6.74		11/21/23	Aalaya Wudaru	- \$	6.74	https://drive.go ogle.com/file/ d/10997LaYjCN Ovp1m68y0z49 kmysnSdffo/vie W
1" Zinc 4pk Corner Brace	1" Corner Brace	for housing	30699136193	\$ 2.57	2	Home Depot	https://www.k	\$	5.14	V	11/21/23	Aalaya Wudaru	~ \$	5.14	https://drive.go ogle.com/file/ d/10997LaYjCN Ovp1m6By0z49 kmysnSdffo/vie W
Wood Screw Zinc PHL FLT #6 x 3/4 100 PC	100 Pack Wood Screws	wood screws for housing	887400176255	\$ 6.87	1	Home Depot	https://www.homedepot.com/p/6-x-1-in-Zinc-Plated-P	\$ 'h	6.87		11/21/23	Aalaya Wudaru	- s	6.87	https://drive.go ogle.com/file/ d/10997LaYjCN Ovp1m6By0z49 kmysnSdffo/vie W
Loctite Ultra Gel Super Glue	Loctite Super Glue	superglue	79340686076	\$ 5.68	1	Home Depot	https://www.k	\$	5.68		12/8/23	Aalaya Wudaru	•		https://drive.goo gle.com/file/d/1 ahmkR5wxVa_J m1fvLwxv5JawB 40sSBpM/view?u sp=sharing
0.75in x 0.75in x 48in HWD RND	Wooden Dowel	housing supports	728927280038	\$ 3.98	1	Home Depot		\$	3.98	V	12/8/23	Aalaya Wudaru	- 5	3.98	http://drive.goo gle.com/file/d/1 ahmkR5wxVg_J m1fyLwxy5Jaw8 40s38pM/view?u sp=sharing

Appendix II: CAD

Motor Housing - Isometric View



Motor Housing - Side View





Appendix III: Code <u>Main System</u>

1	tinclude (ESP32Encoden b)
1	#include (corozencoder.in)
2	#include <arduino.h></arduino.h>
3	
4	// DEETNE PINOLITS OF ESP32
-	I DETARE FARONS OF ESTOR
5	
6	// MOTORS
7	#define & DWM 25
<i>′</i>	
8	#define A_DIR 26
9	#define B_PWM 33
10	#define B DTR 15
**	
11	#define C_PWM 12
12	#define C_DIR 32
13	
14	// ENCODERS
14	// ENCODERS
15	#define ENCODER_A_YEL 16
16	#define ENCODER_A_WHITE 17
17	#define ENCODER B YEL 5
18	#define ENCODER_B_WHITE 23
19	#define ENCODER_C_YEL 18
20	#define ENCODER_C_WHITE 19
21	
21	
22	// BUTTONS AND JOYSTICKS
23	#define BTN 22
24	#define BTN2 27
25	define LED DTN 13
20	TWEITING LED_FIN 15
26	#define JOYBTN 21
27	#define JOYX 39
28	#define JOYY 34
20	
29	
30	// RGB & MUSIC
31	#define BLUE 4
22	#dofine CREEN 14
33	#detine MUSIC 13
34	
35	// DEFINE STATES
36	
27	det state de
37	int state = 1;
38	#define IDLE 1
39	#define DANCE 2
40	#define JOYSTICKCTRL 3
40	#define JOYSTICKCTRL 3
40 41	<pre>#define JOYSTICKCTRL 3 #define REHOME_A 4</pre>
40 41 42	#define DOYSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5
40 41 42 43	#define DOYSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6
40 41 42 43 44	<pre>#define JOYSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6</pre>
40 41 42 43 44	<pre>#define INVSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP MEDIAN ES</pre>
40 41 42 43 44 45	<pre>#define DOYSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES</pre>
40 41 42 43 44 45 46	<pre>#define ZOVSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES</pre>
40 41 42 43 44 45 46 47	<pre>#define ZOVSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS</pre>
40 41 42 43 44 45 46 47 48	<pre>#define JOYSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder:</pre>
40 41 42 43 44 45 46 47 48 40	<pre>#define ZOVSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder;</pre>
40 41 42 43 44 45 46 47 48 49	<pre>#define DOVSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder2;</pre>
40 41 42 43 44 45 46 47 48 49 50	<pre>#define ZOVSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder2; ESP32Encoder encoder3;</pre>
40 41 42 43 44 45 46 47 48 49 50 51	<pre>#define DOWSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder2; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0;</pre>
40 41 42 43 44 45 46 47 48 49 50 51 52	<pre>#define ZONSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0;</pre>
40 41 42 43 44 45 46 47 48 49 50 51 52 52	<pre>#define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SETUP SOUTH</pre>
40 41 42 43 44 45 46 47 48 49 50 51 52 53	<pre>#define DOWSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL</pre>
40 41 42 43 44 45 46 47 48 49 50 51 52 53 53 54	<pre>#define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder;; ESP32Encoder encoder;; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15;</pre>
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55	<pre>#define ZONSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ENCODERS ESP32Encoder encoder2; ESP32Encoder encoder2; ESP32Encoder encoder2; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15;</pre>
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 55 56	<pre>#define INVSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL</pre>
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57	<pre>edefine ZONSTICKCTRL 3 #define REHOME_A 4 edefine REHOME_B 5 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSTION CONTROL int betaTup int int heretWare = 700; </pre>
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 72	<pre>#define ZONSTICKCTRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; </pre>
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 55 55 55 57 58	<pre>#define INVSTICKCTRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES SP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0;</pre>
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59	<pre>edefine ZONSTICKCTRL 3 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_B 5 #// SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES SEP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int theta1 = 0; int theta2 = 0; int theta3 = 0;</pre>
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60	<pre>#define ZONSTICKCTRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SFEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK</pre>
40 41 42 43 44 45 46 47 48 9 50 51 55 55 55 55 55 55 55 55 55 55 55 55	<pre>#define ZONSTICKCTRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int theta1 = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x val = 0; int y val = 0;</pre>
48 41 42 44 45 46 47 48 49 55 55 55 55 55 55 55 55 55 55 55 55 55	<pre>#define REHOME_A 4 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetaI = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_val = 0; int y_val = 0; int tage, int y_val = 0; </pre>
48 41 42 43 44 45 46 47 48 49 50 51 55 55 55 55 55 55 55 55 55 55 55 55	<pre>#define INVSTICKCTRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int thetaMax = 70; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_val = 0; int y_val = 0; int x_origin = 1850; int y_origin = 1875;</pre>
48 41 42 43 44 45 46 47 48 49 50 51 55 55 55 55 55 55 55 55 55 55 55 55	<pre>#define ZONSTICKCTRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetaT = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_val = 0; int y_val = 0; int x_origin = 180; int y_origin = 1875; int x_coord; int y_coord;</pre>
48 41 42 43 44 45 46 47 48 49 50 51 55 55 55 55 55 55 55 55 55 55 55 55	<pre>#define ZONSTICKCTRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int theta1 = 0; int theta2 = 0; int theta3 = 0; // JONSTICK int x_val = 0; int y_val = 0; int x_cond; int y_cond; int cncigin = 1850; int y_origin = 1875; int x_cond; int y_cond; int cncigin_page = 10;</pre>
48 41 42 43 44 45 46 47 48 49 55 55 55 55 55 57 58 59 66 1 62 63 46 56 63 64 55 55 55 57 58 59 66 1 62 63 64 50 64 50 50 50 50 50 50 50 50 50 50 50 50 50	<pre>edefine ZOVSTICKCTRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SPEED CONTROL int megaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOVSTICK int x_val = 0; int y_val = 0; int v_origin = 1850; int y_corigin = 1875; int x_coord; int y_coord; int origin_range = 10; int radius: float anele;</pre>
40 41 42 43 44 45 47 48 49 51 52 53 45 55 57 85 960 61 26 364 65 66 65 66 65 66 65 66 65 66 66 66 66	<pre>#define RENOME_A 4 #define RENOME_A 4 #define RENOME_B 5 #define RENOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSITCK int x_val = 0; int y_val = 0; int x_origin = 1850; int y_origin = 1875; int x_cord; int y_cord; int onigin_range = 10; int radius; float angle; int radius; float angle; </pre>
48 41 42 43 44 45 64 7 89 95 55 55 55 55 89 66 162 63 46 56 66 66 66 66	<pre>#define RENOME_A 4 #define RENOME_A 4 #define RENOME_B 5 #define RENOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder;; ESP32Encoder encoder;; ESP32Encoder encoder;; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int tworigin = 0; int y_val = 0; int x_origin = 1850; int y_origin = 1875; int x_origin = 1850; int y_origin = 1875; int adjus_float angle; int radius_max = sqrt(x_origin * x_origin + y_origin * y_origin)/10;</pre>
48 41 42 43 44 45 46 47 48 95 55 55 55 55 55 57 85 96 61 62 63 45 56 67	<pre>#define RENOME_A 4 #define RENOME_A 4 #define RENOME_B 5 #define RENOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // EXP32Encoder encoder; ESP32Encoder encoder2; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_val = 0; int y_val = 0; int x_origin = 1850; int y_origin = 1875; int x_corigi, = 1850; int y_origin = 1875; int radius; float angle; int radius_max = sqrt(x_origin * x_origin + y_origin * y_origin)/10;</pre>
40 41 42 43 44 56 47 89 51 52 53 55 55 57 58 59 66 16 26 34 55 66 78	<pre>#define RENOME_A 4 #define RENOME_A 4 #define RENOME_B 5 #define RENOME_C 6 // SETUP VARIABLES // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int y_val = 0; int twist = 0; int y_val = 0; int x_val = 0; int y_val = 0; int origin = 1850; int y_origin = 1875; int x_coord; int y_coord; int origin_nage = 10; int radius; float angle; int radius; float angle; int radius_max = sqrt(x_origin * x_origin + y_origin)/10; // FEEDBACK CONTROL PARAMETERS</pre>
40 41 42 43 44 45 6 47 48 95 152 55 55 55 55 55 55 60 162 63 64 56 66 67 89	<pre>#define RENOME_A 4 #define RENOME_A 4 #define RENOME_B 5 #define RENOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder2; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetales; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_corigin = 1850; int y_origin = 1875; int x_corigin = 1850; int y_origin = 1875; int radius_max = sqrt(x_origin * x_origin + y_origin * y_origin)/10; // FEEDBACK CONTROL PARAMETERS int error1; int sumerror1;</pre>
48 41 42 43 44 56 47 8 9 51 52 53 55 55 57 58 96 61 62 63 45 66 67 68 97	<pre>define INVSITEXCRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ENCODERS ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int thetaMax = 700; int thetal = 0; int thetaZ = 0; int theta3 = 0; // JOYSITCK int x_val = 0; int y_val = 0; int x_val = 0; int y_origin = 1875; int accord; int y_cord; int radius; float angle; int radius; float angle; int radius_max = sqrt(x_origin * x_origin + y_origin)/10; // FEEDBACK CONTROL PARAMETERS int error1; int summerro1; int error1; int summerro1; </pre>
48 4 4 2 4 3 4 4 5 4 6 4 7 8 4 9 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	<pre>#define RENOME_A 4 #define RENOME_A 4 #define RENOME_B 5 #define RENOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder;; int encode; int y_val = 0; int thetaMax = 700; int x_origin = 1875; int x_origin = 1850; int y_origin, int endis_float angle; int radius_max = sqrt(x_origin * x_origin + y_origin * y_origin)/10; // FEEDBACK CONTROL PARAMETERS int encod; int y_uare;; int encod; int y_uare;</pre>
40 41 42 43 44 56 47 89 55 55 55 57 89 60 61 62 63 64 65 66 67 88 69 77 1	<pre>define IONSTICKCTRL 3 define REHOME_A 4 define REHOME_A 4 define REHOME_B 5 define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int thetaA = 0; // JOYSITCK int x_val = 0; int y_val = 0; int x_contgin = 1850; int y_origin = 1875; int x_contgin = 1850; int y_origin = 1875; int andius; float angle; int radius_max = sqr(x_origin * x_origin + y_origin //10; // FEEDBACK CONTROL PARAMETERS int error1; int sumerror1; int error2; int sumerror3; </pre>
40 41 42 43 44 56 51 52 55 55 55 59 66 16 67 66 66 78 69 70 72	<pre>define INVSITICKTRL 3 #define REHOME_A 4 define REHOME_A 4 define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // SETUP variable SP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; isSP32Encoder encoder; isSP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int dretaDes; int thetaMax = 700; int thetal = 0; int y_val = 0; int x_val = 0; int y_val = 0; int x_val = 0; int y_val = 0; int adjug = 185; int y_origin = 1875; int adjug = 10; int radjus_max = sqrt(x_origin * x_origin + y_origin)/10; // FEEDBACK CONTROL PARAMETERS int error1; int sumerror1; int error3; int sumerror3;</pre>
48 4 4 2 4 3 4 4 5 6 6 7 8 9 9 5 1 5 2 3 3 4 5 5 6 7 5 8 9 6 6 1 2 6 3 6 4 6 5 6 6 7 8 9 7 7 7 7 7 7	<pre>define INVSITEXCRL 3 define REHOME_A 4 define REHOME_A 4 define REHOME_B 5 define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ENCODERS ESP32Encoder encoder2; ESP32Encoder encoder2; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_val = 0; int y_val = 0; int radius; float angle; int radius; float angle; int error1; int sumerror1; int error3; int sumerror3; // FEEDBACK CONTROL FOR FOR THE SUME A SUME A</pre>
40 41 42 43 44 56 47 89 51 52 53 55 55 55 59 66 162 63 45 56 67 89 70 71 72 73 7	<pre>#define RENOME_A 4 #define RENOME_A 4 #define RENOME_B 5 #define RENOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int theta1 = 0; int thetaMax = 700; int theta1 = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_val = 0; int y_val = 0; int x_val = 0; int y_origin = 1875; int r_coord; int y_coord; int radius; float angle; int radius; float angle; int radius; float angle; int radius; float angle; int error1; int sumerror1; int error2; int sumerror2; int error3; int sumerror3; // FEEDBACK CONTROL for POSITION CONTROL</pre>
48 4 4 2 4 3 4 4 5 6 6 7 8 9 9 5 1 5 2 5 3 5 5 5 7 8 9 68 61 62 63 64 65 66 67 88 9 70 71 72 73 74	<pre>define IONSTICKCRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ENCODERS ESP32Encoder encoder2; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_cond; int y_val = 0; int x_origin = 180; int y_origin = 1875; int ardius_max = sqrt(x_origin * x_origin + y_origin * y_origin)/10; // FEEDBACK CONTROL PARAMETERS int error1; int sumerror1; int error3; int sumerror3; //FEEDBACK CONTROL for POSITION CONTROL float Kp_pos = 0.08;</pre>
40 41 42 43 44 56 47 89 51 52 53 55 55 57 58 59 66 162 66 67 68 69 70 71 72 73 74 75	<pre>define INVSTICKCTRL 3 #define REHOME_A 4 define REHOME_A 4 define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int y_val = 0; int thetal = 0; int y_val = 0; int x_val = 0; int y_val = 0; int x_val = 0; int y_origin = 1875; int accord; int y_cord; int radius; float angle; int radius; float angle; int radius; float angle; int error1; int sumerror1; int error3; int sumerror2; int error3; int sumerror3; //FEEDBACK CONTROL FOR SITION CONTROL float Kp_DOS = 0.08; float Ki_pos = 0.2;</pre>
40 4 4 2 4 3 4 4 5 6 6 7 8 9 9 5 1 5 2 5 3 5 5 5 5 7 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76	<pre>define IONSTICKCTRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ENCODERS ESP32Encoder encoder2; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_cord; int y_val = 0; int x_origin = 1850; int y_origin = 1875; int radius; float angle; int radius; float angle; int radius; float angle; int encor1; int sumerror2; int encor1; int sumerror3; //FEEDBACK CONTROL for POSITION CONTROL float Kp_pos = 0.00; float Ki_pos = 0.00; float Ki_pos = 0.2; int KiMax_pos = 20;</pre>
48 4 4 2 4 3 4 4 5 6 6 7 8 9 9 5 1 5 2 5 3 5 4 5 5 6 7 8 9 6 6 1 6 2 6 3 6 4 6 5 6 6 7 8 6 9 7 7 1 7 2 3 7 4 7 5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	<pre>#define RENOME_A 4 #define RENOME_A 4 #define RENOME_B 5 #define RENOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // EXP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_val = 0; int y_val = 0; int x_vorigin = 1850; int y_origin = 1875; int acounty int y_coord; int radius; float angle; int radius; float angle; int radius_max = sqrt(x_origin * x_origin + y_origin * y_origin)/10; // FEEDBACK CONTROL PARAMETERS int error1; int sumerror1; int error3; int sumerror3; //FEEDBACK CONTROL for POSITION CONTROL float Kp_pos = 0.06; float Ki_pos = 20;</pre>
48 4 4 2 4 3 4 4 5 6 6 7 8 9 9 5 1 5 2 5 3 5 5 5 5 7 8 5 9 68 61 62 63 64 65 66 67 8 69 78 71 72 73 74 75 76 77 -	<pre>define INVSITEXCRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_E 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_val = 0; int y_val = 0; int x_val = 18; int y_origin = 1875; int x_coord; int y_coord; int radius_max = sqrt(x_origin * x_origin + y_origin)/10; // FEEDBACK CONTROL PARAMETERS int error3; int sumerror3; //FEEDBACK CONTROL for POSITION CONTROL float Kp_pos = 0.08; float Kimyos = 20; </pre>
40 4 4 2 4 3 4 4 5 6 6 7 8 9 9 5 1 5 2 5 3 5 4 5 5 6 7 8 9 6 6 1 6 2 6 3 6 4 6 5 6 6 7 8 9 7 7 1 7 2 7 7 4 7 5 7 7 7 8	<pre>define IONSTICKCTRL 3 define REHOME_A 4 define REHOME_A 4 define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES ESP32Encoder encoder; int D = 0; // SFEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int thetaA = 0; // JOYSITCK int x_val = 0; int y_val = 0; int x_origin = 1850; int y_origin = 1875; int x_corigin = 1859; int y_origin = 1875; int radius; float angle; int radius; float angle; int radius_max = sqrt(x_origin * x_origin + y_origin * y_origin)/10; // FEEDBACK CONTROL PARAMETERS int error3; int sumerror3; int sumerror3; int sumerror3; int Kimax_pos = 20; //FEEDBACK CONTROL for SPEED CONTROL</pre>
40 4 4 4 2 4 3 4 4 5 6 7 8 9 9 5 1 5 2 5 3 5 5 5 5 7 5 8 5 9 60 61 62 63 64 65 66 67 88 69 70 71 72 73 74 75 76 77 78 79	<pre>define CONSTICKCTRL 3 #define REHOME_A 4 define REHOME_A 4 define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder; istspace = 0; int omegaDes = 0; int omegaMax = 15; // SEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int detaDes; int thetaMax = 700; int thetal = 0; int y_val = 0; int x_val = 0; int y_val = 0; int x_val = 0; int y_val = 0; int x_val = 0; int y_origin = 1875; int ac_origin = 1850; int y_origin = 1875; int radius; float angle; int radius; float angle; int radius; float angle; int radius; float angle; int error1; int sumerror1; int error2; int sumerror2; int error3; int sumerror2; int error3; int sumerror3; //FEEDBACK CONTROL for POSITION CONTROL float Kp_DOS = 0.08; float Klpos = 0.2; int KIMax_pos = 20; //FEEDBACK CONTROL for SPEED CONTROL int Kp_rehome = 20;</pre>
40 4 4 2 4 3 4 4 5 6 7 8 9 9 5 1 5 2 5 3 5 4 5 5 6 7 8 9 60 61 62 63 64 65 66 67 88 97 71 72 73 74 75 76 77 78 79 80	<pre>define IONSTICKCTRL 3 define REHOME_A 4 define REHOME_A 4 define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ENCODERS ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_roligin = 180; int y_val = 0; int x_origin = 180; int y_origin = 1875; int radius; float angle; int radius; float angle; int error1; int sumerror1; int error3; int sumerror3; //FEEDBACK CONTROL for POSITION CONTROL float KL_pos = 0.0; float KL_pos = 20; //FEEDBACK CONTROL for SPEED CONTROL int KLMAX_DOS = 20; //FEEDBACK CONTROL for SPEED CONTROL int KL_prehome = 20; int KL_pos = 20;</pre>
40 4 4 2 4 3 4 4 5 4 6 7 8 4 9 50 51 52 53 54 55 56 75 88 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 51	<pre>define IONSTICKCTRL 3 #define REHOME_A 4 define REHOME_A 4 define REHOME_B 5 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int metaBes; int thetaMax = 700; int thetal = 0; int thetaMax = 700; int tradius; int y_val = 0; int x_val = 0; int y_val = 0; int x_val = 0; int y_val = 0; int radius; float angle; int radius; float angle; int radius; float angle; int error1; int sumerror1; int error3; int sumerror1; int error3; int sumerror3; //FEEDBACK CONTROL FOR POSITION CONTROL float Kp_pos = 0.0; float Kj_pos = 0.2; int KIMax_pos = 20; //FEEDBACK CONTROL for SPEED CONTROL int Kp_rehome = 10; int KIMax_pos = 15;</pre>
40 4 4 2 4 3 4 4 5 6 6 7 8 9 5 1 5 2 5 3 5 5 5 5 7 5 8 5 60 61 62 63 64 65 66 67 86 69 70 71 72 73 74 75 76 77 78 79 80 11 c	<pre>define IONSTICKCRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_C 6 // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; ESP32Encoder encoder; ESP32Encoder encoder2; ESP32Encoder encoder3; int D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSTICK int x_cord; int y_val = 0; int x_origin = 1850; int y_origin = 1875; int radius; float angle; int radius_max = sqrt(x_origin * x_origin + y_origin * y_origin)/10; // FEEDBACK CONTROL PARAMETERS int error3; int sumerror3; //FEEDBACK CONTROL for POSITION CONTROL float Kp_pos = 0.00; float Ki_pos = 0.2; int KiMax_rehome = 15; </pre>
40 4 4 2 4 3 4 4 5 6 6 7 8 9 9 5 1 5 2 5 3 5 4 5 5 6 5 7 8 9 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82	<pre>define IONSTICKCTRL 3 dedfine REHOME_A 4 define REHOME_A 4 define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // ENCODERS ESP32Encoder encoder; int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int thetaDes; int thetaMax = 700; int thetal = 0; int theta2 = 0; int theta3 = 0; // JOYSITCK int x_val = 0; int y_val = 0; int x_corigin = 1850; int y_origin = 1875; int x_corigin = 1859; int y_origin = 1875; int andius_int v_origin = 1875; int andius_int v_origin * x_origin + y_origin)/10; // FEEDBACK CONTROL PARAMETERS int error1; int sumerror1; int error1; int sumerror3; //FEEDBACK CONTROL for POSITION CONTROL float Kip_ops = 0.08; float Ki_pos = 20; //FEEDBACK CONTROL for SPEED CONTROL int Kimax_nehome = 1; int KiMax_nehome = 15;</pre>
40 4 4 2 4 3 4 4 5 6 6 7 8 9 9 5 1 5 2 5 3 5 5 5 5 7 8 5 9 60 61 62 63 64 65 66 67 8 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83	<pre>define ConstructRL 3 #define REHOME_A 4 #define REHOME_A 4 #define REHOME_C 6 // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // SETUP VARIABLES // ESP32Encoder encoder; ist D = 0; // SPEED CONTROL int omegaSpeed = 0; int omegaDes = 0; int omegaMax = 15; // POSITION CONTROL int dretaDes; int thetaMax = 700; int thetal = 0; int y_val = 0; int x_val = 0; int y_origin = 1875; int accord; int y_coord; int radius; float angle; int radius; float angle; int radius; float angle; int error1; int sumerror1; int error2; int sumerror2; int error3; int sumerror2; int error3; int sumerror3; //FEEDBACK CONTROL for POSITION CONTROL float Kp_DOS = 0.08; float Kipos = 20; //FEEDBACK CONTROL for SPEED CONTROL int Kip_rehome = 1; int Kifax_rehome = 1; int Kifax_rehome = 15; // THER & INTERRUPT VARIABLES</pre>

// ILMER 0 - DEBOUNCE ; ILMERS 2, 3 - ILMING STATES 85 86 hw_timer_t * timer0 = NULL; hw_timer_t * timer2 = NULL; 87 portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED; 88 89 portMUX_TYPE timerMux2 = portMUX_INITIALIZER_UNLOCKED; 90 volatile bool debounceT = false; //flag to check if debounce timer is up volatile bool buttonIsPressed = false; //flag to check if button is pressed and start debounce timer 91 92 volatile bool buttonIsPressed2 = false; //flag to check if button is pressed and start debounce timer volatile bool joystickIsPressed = false; //flag to check if button is pressed and start debounce timer 93 94 volatile bool timerflag = false; 95 volatile bool timerflag2 = false; 96 97 98 // TIMER 1 - ENCODER hw_timer_t * timer1 = NULL; 99 100 portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED; 101 volatile int count1 = 0; // encoder count volatile int count2 = 0; // encoder count 102 103 volatile int count3 = 0; // encoder count volatile bool deltaT = false; // check timer interrupt 1 104 105 106 // PWM PROPERTIES const int freq = 5000; 107 108 const int resolution = 8; 109 const int MAX_PWM_VOLTAGE = 255; const int NOM_PWM_VOLTAGE = 60; 110 111 112 const int ledChannel_1 = 1; const int ledChannel_2 = 2; 113 114 const int ledChannel_3 = 3; const int blueChannel = 6; 115 116 const int greenChannel = 7; 117 //Initialization -----118 119 120 ∨ void IRAM_ATTR onTime0() { portENTER CRITICAL ISR(&timerMux0); 121 122 debounceT = true; 123 portEXIT_CRITICAL_ISR(&timerMux0); timerStop(timer0); 124 125 } 126 127 Void IRAM_ATTR onTime1() { 128 portENTER_CRITICAL_ISR(&timerMux1); 129 count1 = encoder.getCount(); count2 = encoder2.getCount(); 130 131 count3 = encoder3.getCount(); encoder.clearCount (); 132 133 encoder2.clearCount (); 134 encoder3.clearCount (); 135 deltaT = true; 136 portEXIT_CRITICAL_ISR(&timerMux1); 137 > 138 139 V void IRAM_ATTR onTime2() { portENTER_CRITICAL_ISR(&timerMux2); 140 timerflag = true; 141 142 portEXIT_CRITICAL_ISR(&timerMux2); 143 timerStop(timer2); 144 } 145 void IRAM_ATTR isr() { // the function to be called when interrupt is triggered 146 buttonIsPressed = true; 147 148 timerStart(timer0); 149 } 150 void IRAM_ATTR isr2() { // the function to be called when interrupt is triggered 151 buttonIsPressed2 = true; 152 153 timerStart(timer0); } 154 155 156 void IRAM_ATTR isr3() { // the function to be called when interrupt is triggered 157 joystickIsPressed = true; 158 timerStart(timer0); 159 > 160 161 //INITIALIZE -----162 163 164 //TIMERS void TimerInterruptInit() { //The timer simply counts the number of Tic generated by the quartz. With a quartz clocked at 80MHz, we will have 80,000,000 Tics. 165 V 166 167 // Debounce timer (200ms) 168 timer0 = timer8egin(0, 80, true); // timer 1, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 -> 1000 ns = 1 us, countUp 169 timerAttachInterrupt(timer0, &onTime0, true); // edge (not level) triggered timerAlarmWrite(timer0, 200000, true); // 200000 * 1 us = 200 ms, autoreload true 170 timerAlarmEnable(timer0); // enable 171 172 timerStop(timer0); 173 timerRestart(timer0); 174 175 // Encoder timer (10ms) timer1 = timerBegin(1, 80, true); // timer 1, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 -> 1000 ns = 1 us, countUp 176 177 timerAttachInterrupt(timer1, &onTime1, true); // edge (not level) triggered 178 timerAlarmWrite(timer1, 10000, true); // 10000 * 1 us = 10 ms, autoreload true 179 timerAlarmEnable(timer1); // enable 180 181 // Dance timer (8s) timer2 = timerBegin(2, 80, true); // divides the frequency by the prescaler: 80,000,000 / 80 = 1,000,000 tics / sec 182 183 timerAttachInterrupt(timer2, &onTime2, true); // sets which function do you want to call when the interrupt is triggered

184 timerAlarmWrite(timer2, 12000000, true); // sets how many tics will you count to trigger the interrupt, 3 000 000 * 1 us = 3 s timerAlarmEnable(timer2); // Enables timer 185 186 timerStop(timer2); 187 timerRestart(timer2); 188 189 } 198 // SETUP -----191 192 193 V void setup() { 194 195 Serial.begin(115200); 196 197 //MOTORS pinMode(A_DIR, OUTPUT); digitalWrite(A_DIR, LOW); // sets the initial direction 198 pinMode(B_DIR, OUTPUT); digitalWrite(B_DIR, LOW); // sets the initial direction pinMode(C_DIR, OUTPUT); digitalWrite(C_DIR, LOW); // sets the initial direction 199 200 201 // BUTTONS 202 203 pinMode(BTN, INPUT); pinMode(BTN2, INPUT); 204 205 pinMode(JOYBTN, INPUT_PULLUP); attachInterrupt(BTN, isr, RISING); 206 207 attachInterrupt(BTN2, isr2, RISING); 208 attachInterrupt(JOYBTN, isr3, RISING); 209 210 // ENCODER 211 ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up resistors encoder.attachHalfQuad(ENCODER_A_WHITE, ENCODER_A_YEL); // Attach pins for use as encoder pins encoder.setCount(0); // set starting count value after attaching 212 213 encoder2.attachHalfQuad(ENCODER_B_WHITE, ENCODER_B_YEL); // Attach pins for use as encoder pins
encoder2.setCount(0); // set starting count value after attaching 214 215 216 encoder3.attachHalfQuad(ENCODER_C_WHITE, ENCODER_C_YEL); // Attach pins for use as encoder pins encoder3.setCount(0); // set starting count value after attaching 217 218 // LED PWM 219 220 ledcSetup(ledChannel_1, freq, resolution); // configure LED PWM functionalitites ledcAttachPin(A_PWM, ledChannel_1); // attach the channel to the GPIO to be controlled 221 222 ledcSetup(ledChannel_2, freq, resolution); // configure LED PWM functionalitites ledcAttachPin(B_PWM, ledChannel_2); // attach the channel to the GPIO to be controlled 223 224 ledcSetup(ledChannel_3, freq, resolution); // configure LED PWM functionalitites ledcAttachPin(C PWM, ledChannel 3); // attach the channel to the GPIO to be controlled 225 226 // TIMER 227 TimerInterruptInit(); // Initiates timer interrupt 228 229 // LED STRIP AND MUSIC 230 231 pinMode(BLUE, OUTPUT); 232 233 ninMode(GREEN OUTPUT): ~~. 234 pinMode(MUSIC, OUTPUT); 235 236 ledcSetup(blueChannel, freq, resolution); 237 ledcSetup(greenChannel, freq, resolution); 238 239 ledcAttachPin(BLUE, blueChannel); 240 ledcAttachPin(GREEN, greenChannel); 241 } 242 243 244 void loop() { 245 246 if (deltaT) { 247 portENTER_CRITICAL(&timerMux1); 248 deltaT = false; portEXIT_CRITICAL(&timerMux1); 249 250 switch (state) { 251 252 // STATE 1 - IDLE: Default mode with nothing happening 253 254 case IDLE: 255 256 dance_off(); ledcWrite(ledChannel_1, LOW); digitalWrite(A_DIR, LOW); ledcWrite(ledChannel_2, LOW); digitalWrite(B_DIR, LOW); ledcWrite(ledChannel_3, LOW); digitalWrite(C_DIR, LOW); 257 258 259 260 261 Serial.println("IDLE"); 262 263 if (CheckForButtonPress()) { // EVENT: when button 1 is pressed Serial.println("BUTTON PRESSED"); 264 // SERVICE: turns lights and music on, makes arm dance 265 dance_on(); 266 timerStart(timer2);
radius = 300; angle = 0; // SERVICE: starts dance timer 267 268 state = DANCE; } 269 270 if (CheckForJoystickPress()) { // EVENT: when iovstick button is pressed 271 Serial.println("JOYSTICK USER CONTROL");; 272 273 state = JOYSTICKCTRL; 274 } 275 276 if (CheckForButtonPress2()) { // EVENT: when button 2 is pressed 277 Serial.println("BUTTON 2 PRESSED"); 278 state = REHOME_A; 279 } 280 281 break;

282

283	// STATE 2 - DANCE: lights and music come on, arm starts dancing	
284	case DANCE:	
285	Serial.println("DANCE SEQUENCE WOOHOO");	
286	dance_on();	
288	<pre>if (CheckForButtonPress() timerflag) {</pre>	// EVENT: when button is pressed OR when dance timer is up
289	<pre>dance_off();</pre>	<pre>// SERVICE: turns lights and music off, stops arm from dancing</pre>
290	<pre>reset_dance_timer(); Sepial_println("DANCE_OVER_:(");</pre>	<pre>// SERVICE: resets dance timer and flag</pre>
291	radius = 0; angle = 0;	
293	<pre>state = IDLE;</pre>	
294	}	
295	healt	
296	break;	
298	// STATE 3 - JOYSTICKCTRL: User can control the movement of the a	rm with the joystick
299	case JOYSTICKCTRL:	
300	polar coordinates();	
302	<pre>position_control();</pre>	
303		
304	<pre>if (CheckForJoystickPress()) {</pre>	<pre>// EVENT: when joystick button is pressed</pre>
305	<pre>sumerror1 = 0; sumerror2 = 0; sumerror3 = 0; state = TDLE;</pre>	
307)	
308		
309	break;	
310	// STATE 4 - REHOME A: User can control the speed of motor A	
312	case REHOME_A:	
313		
314	<pre>Serial.println("CALIBRATE MOTOR A");</pre>	11 CEMITER, Grand Cart 3 of Maria
315 316	<pre>speed_control(count1, ledChannel_1, A_DIR, error1, sumerror1);</pre>	// SERVICE: Speed Control of Motor A
317	<pre>if (CheckForButtonPress2()) {</pre>	// EVENT: when button 2 is pressed
318	<pre>sumerror1 = 0;</pre>	
319	<pre>state = REHOME_B;</pre>	
320	3	
322	break;	
323		
324	<pre>// STATE 5 - REHOME_B: User can control the speed of motor B</pre>	
325	Case KEHCME_B:	
327	Serial.println("CALIBRATE MOTOR B");	
328	<pre>speed_control(count2, ledChannel_2, B_DIR, error2, sumerror2);</pre>	// SERVICE: Speed Control of Motor B
329	if (ChackEonButtonDesse2()) [// EVENT: when hutton 2 is pressed
331	sumerror2 = 0;	// Event. when buccon 2 is pressed
352	<pre>state = KEHUME_C;</pre>	
333	} break:	
335	U CORY	
336	<pre>// STATE 6 - REHOME_C: User can control the speed of motor C</pre>	
337	case REHOME_C:	
338	Serial.println("CALIBRATE MOTOR C");	
340	<pre>speed_control(count3, ledChannel_3, C_DIR, error3, sumerror3);</pre>	// SERVICE: Speed Control of Motor C
341		
342	if (CheckForButtonPress2()) { Social printle("CALIBRATION COMPLETE");	// EVENT: when button 2 is pressed
345	sumerror3 = 0;	
345	<pre>state = IDLE;</pre>	
346	}	
347	broak	
349	}	
350	>	
351	}	
352	// EVENT CHECKERS	
354		
355 🗸	<pre>bool CheckForButtonPress() {</pre>	
356	if (debounceT && buttonIsPressed) { portENTER_CETITCAL(&timenture);	
358	<pre>debounceT = false;</pre>	
359	<pre>portEXIT_CRITICAL(&timerMux0);</pre>	
360	<pre>timerStop(timer0);</pre>	
361	<pre>buttonIsPressed = false;</pre>	
363	} else {	
364	return false;	
365	}	
366	}	
368 ~	<pre>bool CheckForButtonPress2() {</pre>	
369	if (debounceT && buttonIsPressed2) {	
370	<pre>portENTER_CRITICAL(&timerMux0);</pre>	
371	<pre>depounce1 = false; nortEXIT_CRITICAL(&timerMux8);</pre>	
373	<pre>timerStop(timer0);</pre>	
374	<pre>buttonIsPressed2 = false;</pre>	
375	return true;	
376 377	<pre>> eise { return false:</pre>	
378	}	
379	}	
380	heal ChackEanJourtickDears() (
581 V	DOUL CHECKFOFJOVSLICKPPESS() {	

382 if (debounceT && joystickIsPressed) { 383 portENTER_CRITICAL(&timerMux0); 384 debounceT = false; 385 portEXIT_CRITICAL(&timerMux0); timerStop(timer0);
joystickIsPressed = false; 386 387 388 return true; 389 } else { 390 return false; 391 } 392 } 393 394 395 // SERVICES -----396 397 ∨ void dance_on() { float t = millis()/250.0; 398 399 int g = sin(t)*128+128; 400 int b = cos(t)*128+128; ledcWrite(greenChannel, 255-g); 401 ledcWrite(blueChannel, 255-b); 402 403 Serial.println(g); digitalWrite(MUSIC, HIGH); 404 405 angle += 0.5; if (angle >= 360) { angle = 0;} 406 407 408 position_control(); 409 } 410 411 v void dance_off() { ledcWrite(greenChannel, 255); ledcWrite(blueChannel, 255); 412 413 414 digitalWrite(MUSIC, LOW); 415 } 416 417 Void reset_dance_timer() { 418 timerStop(timer2); 419 timerRestart(timer2); 420 timerflag = false; 421 } 422 423 V void plotSpeedData() { 424 Serial.print("Speed:"); Serial.print(omegaSpeed);// Serial.print(", "); Serial.print(omegaSpeed2); Serial.print(" "); Serial.print("Desired_Speed:"); Serial.print(omegaDes);// Serial.print(", "); Serial.print(omegaDes2); Serial.print(" "); 425 426 Serial.print("PWM_Duty:"); Serial.print(D);// Serial.print(", "); Serial.println(D2); 427 } 428 429 V void joystick_coordinates() { 430 431 // reads x, y values from joystick 432 x_val = analogRead(JOYX); y_val = analogRead(JOYY); 433 434 // maps x , y values from center of joystick taken to be (0,0)435 x_coord = (x_val - x_origin)/10; y_coord = (y_val - y_origin)/10; 436 437 // recenter center by scaling 438 if (x_coord > 0) { x_coord = (x_coord * x_origin) / (4096 - x_origin); 439 440 _____y_coord = (y_coord * y_origin) / (4096 - y_origin); } 441 442 443 444 445 // filters noise when joystick is at (0,0) 446 float r_sq = sqrt(x_coord*x_coord + y_coord*y_coord); if (r_sq <= origin_range/10) {
 x_coord = 0;</pre> 447 448 y_coord = 0; 449 } 450 451 } 452 453 ∨ void polar_coordinates() { 454 joystick_coordinates(); 455 456 457 radius = sqrt(abs(x_coord) * abs(x_coord) + abs(y_coord) * abs(y_coord)); 458 459 if (x_coord == 0) { 460 if (y_coord>0) {angle = 90;} 461 else {angle = -90;} 462 } 463 else { angle = atan2(y_coord,x_coord)/PI*180; 464 465 } 466 Serial.print("\tRadius: "); Serial.print(radius); 467 468 Serial.print("\tAngle: "); Serial.println(angle); 469 } 470 471 \vee void speed_control(int count, int ledChannel, int motor_dir, int error, int sumerror) { 472 473 joystick_coordinates(); 474 475 omegaSpeed = count: 476 omegaDes = map(x_coord, -x_origin/10, x_origin/10, -omegaMax, omegaMax); // PLEASE SPECIFY OMEGAMAX VALUE ABOVE 477 478 //Feedback control 479 error = omegaDes - omegaSpeed; 480 sumerror += error:

if (abs(Ki_rehome/2 * sumerror) > abs(KiMax_rehome)) { 481 if (sumerror < 0) { D = Kp_rehome * error - KiMax_rehome; }
else { D = Kp_rehome * error + KiMax_rehome;}</pre> 482 483 484 485 else { D = Kp_rehome * error + Ki_rehome/2 * sumerror;} 486 //Ensure that you don't go past the maximum possible command 487 488 if (D > MAX_PWM_VOLTAGE) { D = MAX_PWM_VOLTAGE; }
else if (D < -MAX_PWM_VOLTAGE) { D = -MAX_PWM_VOLTAGE; }</pre> 489 490 //Nap the D value to motor directionality
if (D > 0) (ideckwite(ledchannel, 0); digitalkwite(motor_dir, LON);)
else if (D < 0) (ideckwite(ledchannel, -0); digitalkwite(motor_dir, NIGN);)
else { ledckwite(ledChannel, LON); digitalkwite(motor_dir, LON); }</pre> 491 492 493 494 495 496 plotSpeedData(); 497 } 498 499 ∨ void angle_mapping(int ledChannel) { if (ledChannel == 1) {
 if (angle >= -180 && angle <= -150) { angle += 360;}</pre> 500 501 $\label{eq:angle} if \{angle >= -30 & 8k angle <= 30 \} \ \{ thetaDes = nap(angle, -30, 30, 0, thetaDes); \} \\ if (angle >= 150 & 8k angle <= 210) \ \{ thetaDes = nap(angle, 210, 150, 0, thetaDes); \} \\ if (angle >= -150 & 8k angle <= -30) \ \{ thetaDes = 0; \} \\ \end{cases}$ 502 503 504 505 } 506 507 if (ledChappel == 2) / t (zecunameL == 2) {
 if (angle >= 90 && angle <= 150) { thetaDes = map(angle, 90, 150, 0, thetaDes);}
 if (angle >= -90 && angle <= -30) { thetaDes = map(angle, -30, -90, 0, thetaDes);}
</pre> 508 509 if (angle > -30 && angle < 90) { thetaDes = 0;} 510 if (ledChannel == 3) { 511 (rescummes ==); if (angle >= -180 && angle <= -150) { angle += 360;} if (angle >= -30 && angle <= -90) { thetaDes = map(angle, 90, 30, 0, thetaDes);} if (angle >= -150 && angle <= -90) { thetaDes = map(angle, -150, -90, 0, thetaDes);}</pre> 512 513 514 if (angle > 90 && angle < 210) { thetaDes = 0;} 515 } 516 517 > 518 519 ∨ void position_control() { 520 521 //MOTOR A thetal += count1; 522 thetaDes = map(radius, 0, radius_max, 0, thetaMax); 523 524 angle_mapping(ledChannel_1); 525 error1 = thetaDes - theta1: 526 sumerror1 += error1; if (abs(Ki_pos* sumerror1) > abs(KiMax_pos)) { 527 528 529 if (sumerror1 < 0) { D = Kp_pos * error1 - KiMax_pos;}</pre> 530 else { D = Kp_pos * error1 + KiMax_pos;} 531 else { D = Kp_pos * error1 + Ki_pos * sumerror1;} 532 533 //Ensure that you don't go past the maximum possible command if (D > MAX_PMM_VOLTAGE) { D = MAX_PMM_VOLTAGE; } else if (D < -MAX_PMM_VOLTAGE) { D = -MAX_PMM_VOLTAGE;}</pre> 534 535 536 537 //Map the D value to motor directionality 538 539 540 if (D > 0) { ledcWrite(ledChannel_1, D); digitalWrite(A_DIR, LOW); }
else if (D < 0) { ledcWrite(ledChannel_1, -D); digitalWrite(A_DIR, HIGH); }</pre> 541 else { ledcWrite(ledChannel_1, LOW); digitalWrite(A_DIR, LOW); } 542 543 //MOTOR B 544 theta2 += count2: thetaDes = map(radius, 0, radius_max, 0, thetaMax); angle_mapping(ledChannel_2); 545 546 547 548 549 error2 = thetaDes - theta2: sumerror2 += error2; if (abs(Ki_pos* sumerror2) > abs(KiMax_pos)) { 550 551 552 if (sumerror2 < 0) { D = Kp_pos * error2 - KiMax_pos;}
else { D = Kp_pos * error2 + KiMax_pos;}</pre> 553 } 554 else { D = Kp_pos * error2 + Ki_pos * sumerror2;} 555 //Ensure that you don't go past the maximum possible command if (D > MAX_PWM_VOLTAGE) { D = MAX_PWM_VOLTAGE; } else if (D < -MAX_PWM_VOLTAGE) { D = -MAX_PWM_VOLTAGE;}</pre> 556 557 558 559 560 //Map the D value to motor directionality 561 if (D > 0) { ledcWrite(ledChannel_2, D); digitalWrite(B_DIR, LOW); } 562 else if (D < 0) { ledcWrite(ledChannel_2, -D); digitalWrite(B_DIR, HIGH); }
else { ledcWrite(ledChannel_2, LOW); digitalWrite(B_DIR, LOW); }</pre> 563 564 565 //MOTOR C 566 theta3 += count3; 567 thetaDes = map(radius, 0, radius_max, 0, thetaMax); angle mapping(ledChannel 3); 568 569 570 error3 = thetaDes - theta3: 571 sumerror3 += error3; if (abs(Ki_pos* sumerror3) > abs(KiMax_pos)) {
 if (sumerror3 < 0) { D = Kp_pos * error3 - KiMax_pos;}
 else { D = Kp_pos * error3 + KiMax_pos;}</pre> 572 573 574 575 } 576 577 else { D = Kp_pos * error3 + Ki_pos * sumerror3;} 578 //Ensure that you don't go past the maximum possible command if (D > MAX_PWM_VULIAGE) { D = MAX_PWM_VULIAGE; }
else if (D < -MAX_PWM_VOLTAGE) { D = -MAX_PWM_VOLTAGE;}</pre> 579 580 581 582 //Map the D value to motor directionality 583 if (D > 0) { ledcWrite(ledChannel_3, D); digitalWrite(C_DIR, LOW); } 584 else if (D < 0) { ledcwite(ledChannel_3, -D); digitalWrite(C_DIR, HIGH); }
else { ledcwrite(ledChannel_3, LOW); digitalWrite(C_DIR, LOW); }</pre> 585 586 587 }

```
#include <Arduino.h>
1
 2
     #define INP 27
 3
 4
    #define SPK 26
   #define freg 5000
 5
 6
    #define chan 0
 7
     #define resolution 8
 8
 9
     const float tones[32] = {586, 698, 932, 1175, 0, 1175, 0, 932,
10
                          1046, 1046, 1244, 1244, 1175, 1175, 932, 932,
                           466, 586, 698, 932, 0, 932, 0, 698,
11
12
                          880, 880, 1046, 1046, 932, 932, 0, 0};
13
     int i = 0;
14
    int length = 32;
15
    volatile bool play = false;
16
17
    void IRAM_ATTR isr() {
     play = digitalRead(INP);
18
19
    }
20
21
    void setup() {
22
     Serial.begin(115200);
23
       // put your setup code here, to run once:
24
      ledcSetup(chan, freq, resolution);
25
      ledcAttachPin(SPK, chan);
26
27
      pinMode(INP, INPUT);
      attachInterrupt(INP, isr, CHANGE);
28
29
     play = false;
30
     - }
31
32
     void loop() {
33
     Serial.print("play: ");
34
      Serial.println(play);
35
      Serial.print("inp: ");
      Serial.println(digitalRead(INP));
36
37
      // put your main code here, to run repeatedly:
38
      for (int t = 0; t < 200; ++t) {</pre>
39
       delay(1);
40
        if (!play) {
          ledcWriteTone(chan, 0);
41
42
         i = 0;
43
         }
44
       3
45
      if (play) {
46
       ledcWriteTone(chan, tones[i]);
47
       - 3
48
      i = (i + 1) % length;
49
```

Appendix IV: Mapping of motors to joystick position considerations



Preparing joystick data and converting it to polar coordinates

(0,max,max)