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.

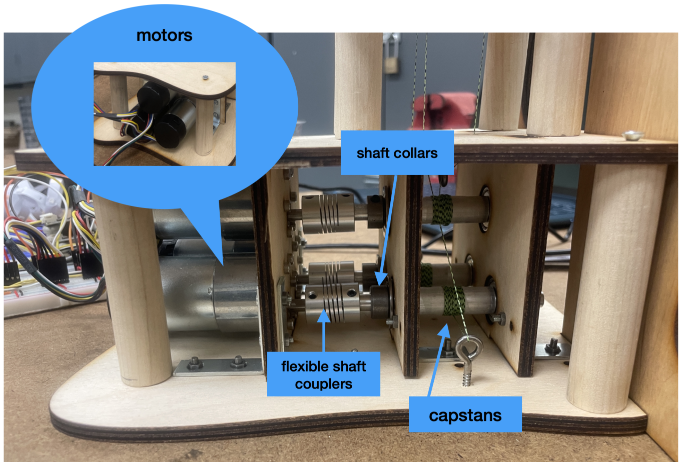
High Level Strategy:

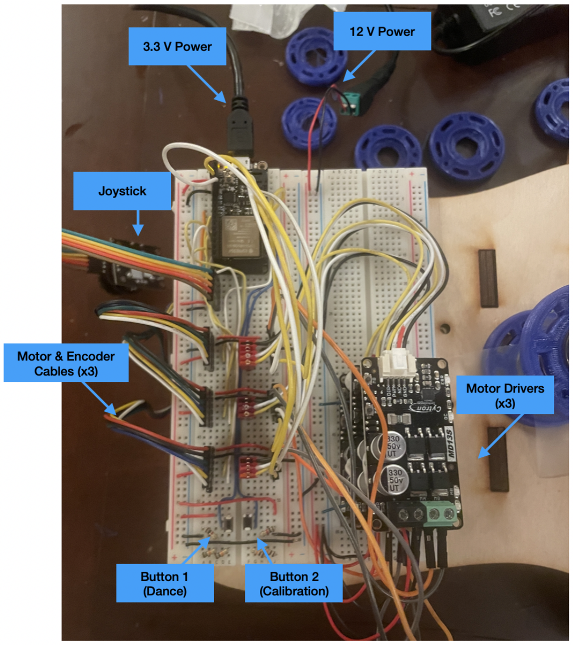
The initial desired functionality was to have an octopus robot that flexed and/or moved its tentacles in response to hand motions as the external controller. We were quite ambitious and expected to have multiple tentacle arms and utilize various sensors to capture and transmit the motions of a hand and 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.

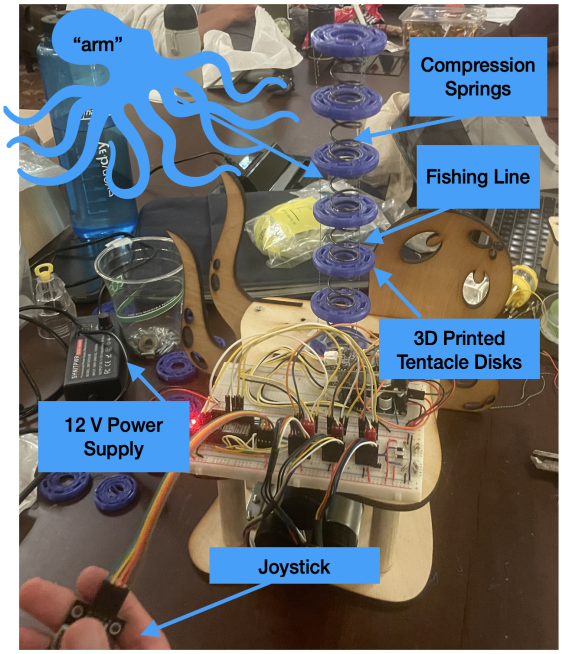
Our high level strategy then became to utilize three degrees of actuation controlled by three "tendons". The string lengths would vary by motor-capstan rotation, thereby creating the fluid motion of the tentacle. The rigidity of the tentacle arm would be maintained by disks and springs between them. 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:

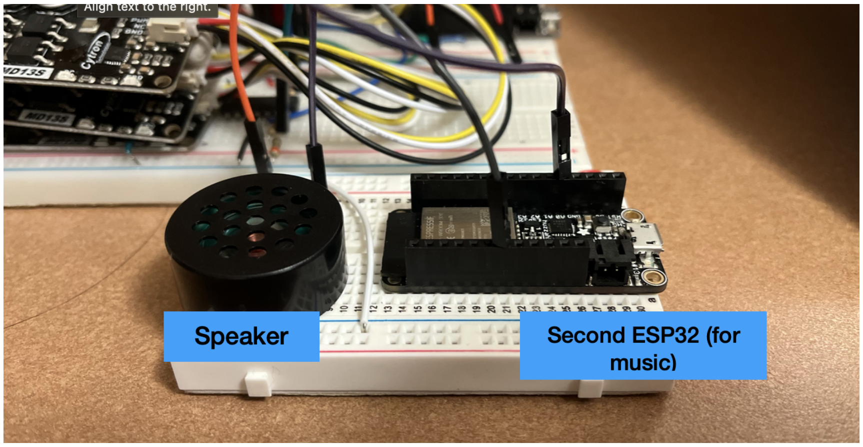
Below, we show our motor assembly and housing, circuitry, and final assembly.







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

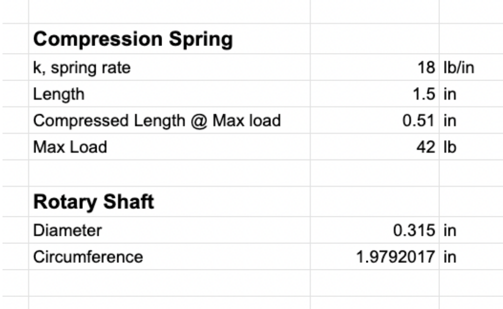


Critical Design Decisions and Calculations:

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:

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

The free length of the tentacle is:

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

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

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

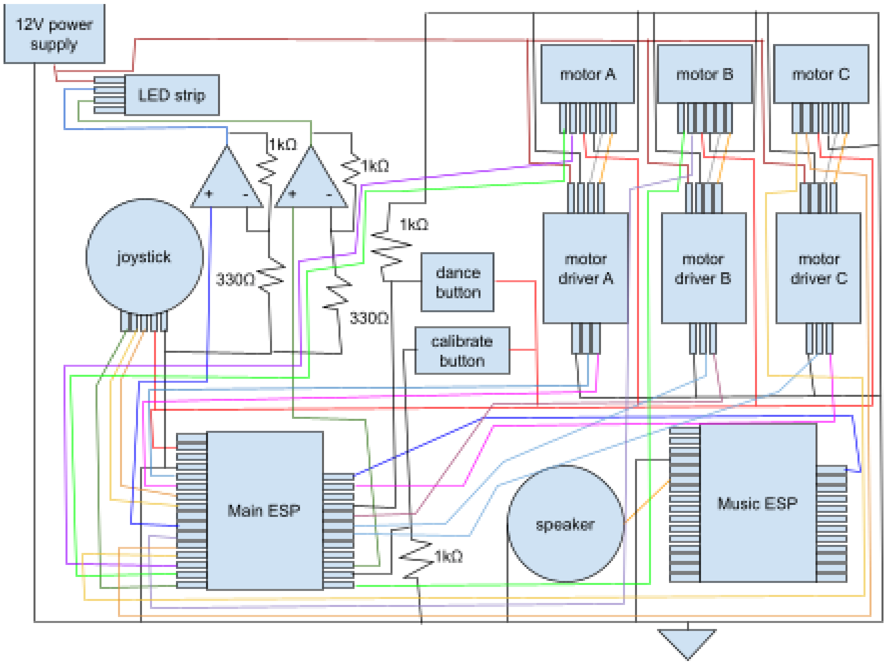
At worst, this would all be held by one cable, with a tension of 9 lb. We used this maximum cable tension to determine which fishing line to buy. However, we ultimately opted for overkill since we didn't want the line to stretch and there was no obvious downside to doing so. We also used the tension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

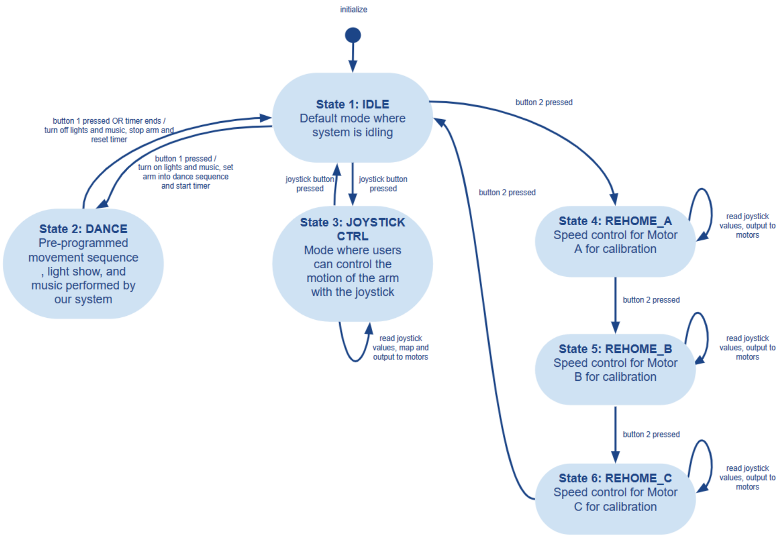
The associated motor torque would be:

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

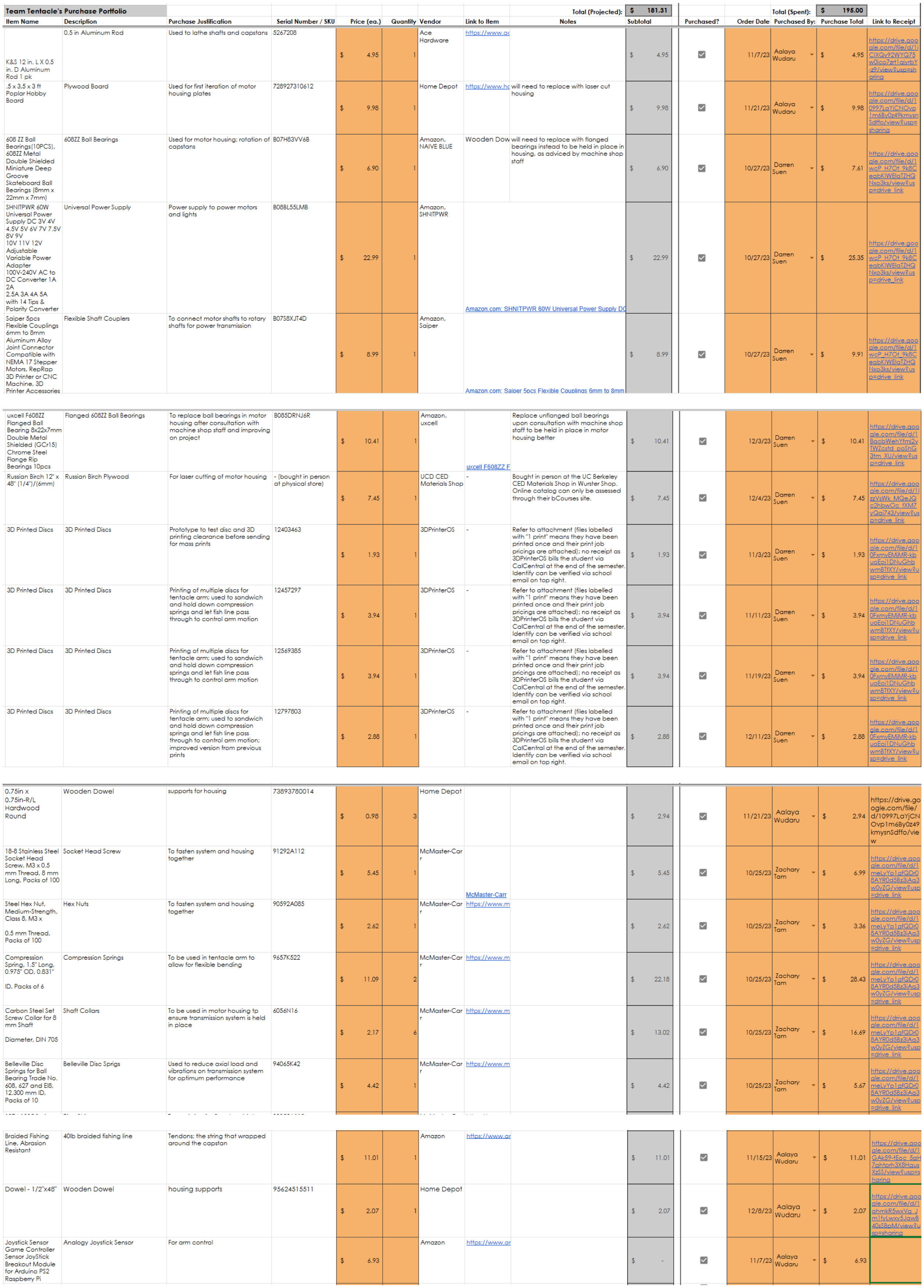



initialize
State 1: IDLE
Default mode where system is idling
button 1 pressed OR timer ends / turn off lights and music, stop arm and reset timer
button 1 pressed / turn on lights and music, set arm into dance sequence and start timer
State 2: DANCE
Pre-programmed movement sequence , light show, and music performed by our system
joystick button pressed
joystick button pressed
State 3: JOYSTICK CTRL
Mode where users can control the motion of the arm with the joystick
read joystick values, map and output to motors
button 2 pressed
button 2 pressed
State 4: REHOME_A
Speed control for Motor A for calibration
read joystick values, output to motors
button 2 pressed
State 5: REHOME_B
Speed control for Motor B for calibration
read joystick values, output to motors
button 2 pressed
State 6: REHOME_C
Speed control for Motor C for calibration
read joystick values, output to motors

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

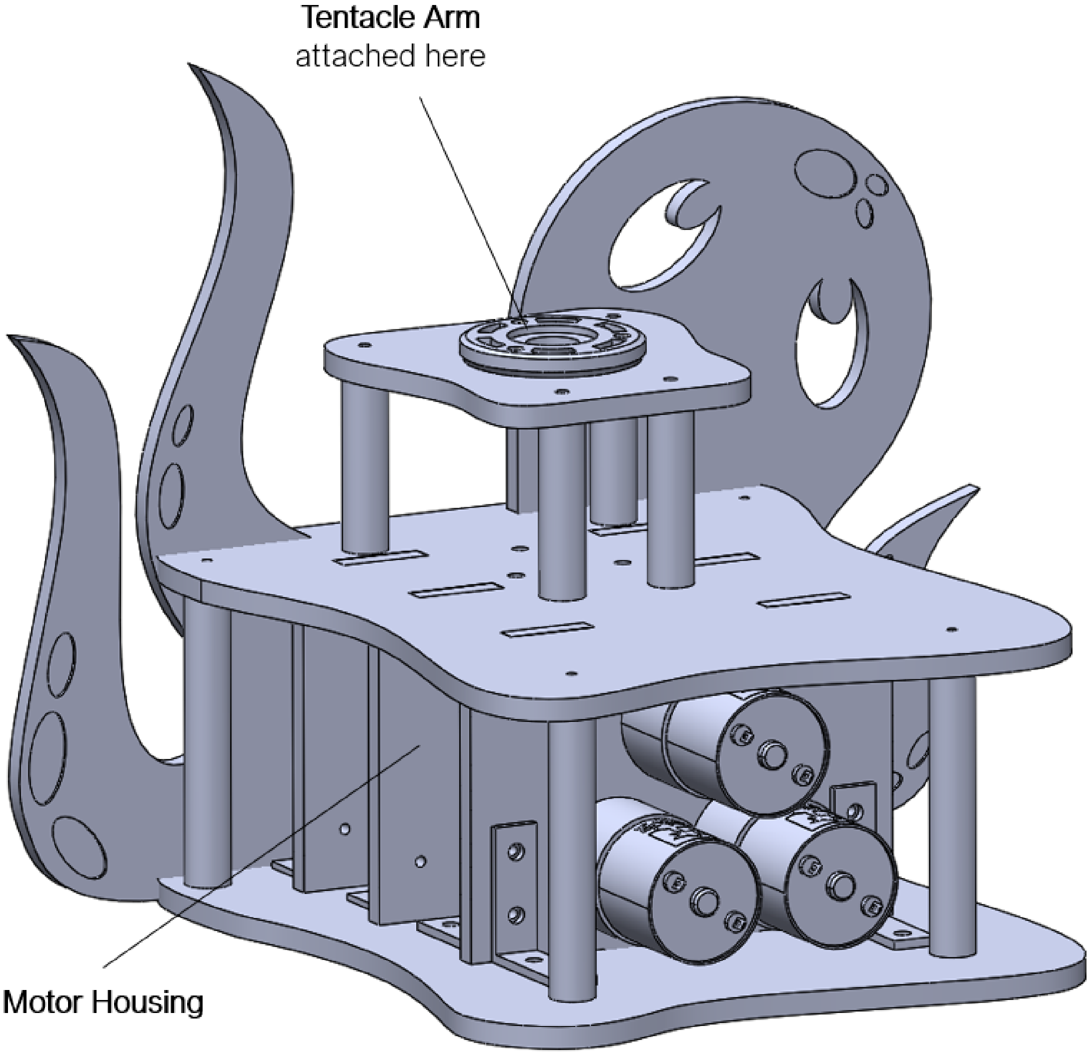


Team Tentacle's Purchase Portfolio								Total (Projected):	$ 181.31			Total (Spent):	$ 195.00	
Item Name	**Description**	**Purchase Justification**	**Serial Number / SKU**	**Price (ea.)**	**Quantity**	**Vendor**	**Link to Item**	**Notes**	**Subtotal**	**Purchased?**	**Order Date**	**Purchased By:**	**Purchase Total**	**Link to Receipt**
K&S 12 in. L X 0.5 in. D Aluminum Rod 1 pk	0.5 in Aluminum Rod	Used to lathe shafts and capstans	5267208	$ 4.95	1	Ace Hardware	https://www.ac		$ 4.95	☑	11/7/23	Aalaya Wudaru	$ 4.95	https://drive.google.com/file/d/1iCIXQy92WYQ75w0ico7zrt1qjvrbY-z9/view?usp=sharing
.5 x 3.5 x 3 ft Poplar Hobby Board	Plywood Board	Used for first iteration of motor housing plates	728927310612	$ 9.98	1	Home Depot	https://www.hc	will need to replace with laser cut housing	$ 9.98	☑	11/21/23	Aalaya Wudaru	$ 9.98	https://drive.google.com/file/d/10997LaYjCNOvp1m6By0z49kmysnSdffo/view?usp=sharing
608 ZZ Ball Bearings(10PCS), 608ZZ Metal Double Shielded Miniature Deep Groove Skateboard Ball Bearings (8mm x 22mm x 7mm)	608ZZ Ball Bearings	Used for motor housing; rotation of capstans	B07H83VV6B	$ 6.90	1	Amazon, NAIVE BLUE	Wooden Dow	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	$ 7.61	https://drive.google.com/file/d/1wcP_H7Ot_9k8CeabKIWElaTZHQNxo3ks/view?usp=drive_link
SHNITPWR 60W Universal Power 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 2.5A 3A 4A 5A with 14 Tips & Polarity Converter	Universal Power Supply	Power supply to power motors and lights	B08BL55LMB	$ 22.99	1	Amazon, SHNITPWR	Amazon.com: SHNITPWR 60W Universal Power Supply DC		$ 22.99	☑	10/27/23	Darren Suen	$ 25.35	https://drive.google.com/file/d/1wcP_H7Ot_9k8CeabKIWElaTZHQNxo3ks/view?usp=drive_link
Saiper 5pcs Flexible Couplings 6mm to 8mm Aluminum Alloy Joint Connector Compatible with NEMA 17 Stepper Motors, RepRap 3D Printer or CNC Machine, 3D Printer Accessories	Flexible Shaft Couplers	To connect motor shafts to rotary shafts for power transmission	B07S8XJT4D	$ 8.99	1	Amazon, Saiper	Amazon.com: Saiper 5pcs Flexible Couplings 6mm to 8mm		$ 8.99	☑	10/27/23	Darren Suen	$ 9.91	https://drive.google.com/file/d/1wcP_H7Ot_9k8CeabKIWElaTZHQNxo3ks/view?usp=drive_link
uxcell F608ZZ Flanged Ball Bearing 8x22x7mm Double Metal Shielded (GCr15) Chrome Steel Flange Rip Bearings 10pcs	Flanged 608ZZ Ball Bearings	To replace ball bearings in motor housing after consultation with machine shop staff and improving on project	B085DRNJ6R	$ 10.41	1	Amazon, uxcell	uxcell F608ZZ F	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	$ 10.41	https://drive.google.com/file/d/1BacbWehYfmj2vTWZcstd_poShG3tm_XU/view?usp=drive_link
Russian Birch 12" x 48" (1/4")/(6mm)	Russian Birch Plywood	For laser cutting of motor housing	- (bought in person at physical store)	$ 7.45	1	UCD CED Materials Shop	-	Bought in person at the UC Berkeley CED Materials Shop in Wurster Shop. Online catalog can only be assessed through their bCourses site.	$ 7.45	☑	12/4/23	Darren Suen	$ 7.45	https://drive.google.com/file/d/1IzzVsWk_MQeJQc2hbwOc_fXM7yQai743/view?usp=drive_link
3D Printed Discs	3D Printed Discs	Prototype to test disc and 3D printing clearance before sending for mass prints	12403463	$ 1.93	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 CalCentral at the end of the semester. Identify can be verified via school email on top right.	$ 1.93	☑	11/3/23	Darren Suen	$ 1.93	https://drive.google.com/file/d/10FxmyEMjMR-kbuoEoj1DNuGhbwmBTfXY/view?usp=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	$ 3.94	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 CalCentral at the end of the semester. Identify can be verified via school email on top right.	$ 3.94	☑	11/11/23	Darren Suen	$ 3.94	https://drive.google.com/file/d/10FxmyEMjMR-kbuoEoj1DNuGhbwmBTfXY/view?usp=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	12569385	$ 3.94	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 CalCentral at the end of the semester. Identify can be verified via school email on top right.	$ 3.94	☑	11/19/23	Darren Suen	$ 3.94	https://drive.google.com/file/d/10FxmyEMjMR-kbuoEoj1DNuGhbwmBTfXY/view?usp=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; improved version from previous prints	12797803	$ 2.88	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 CalCentral at the end of the semester. Identify can be verified via school email on top right.	$ 2.88	☑	12/11/23	Darren Suen	$ 2.88	https://drive.google.com/file/d/10FxmyEMjMR-kbuoEoj1DNuGhbwmBTfXY/view?usp=drive_link
0.75in x 0.75in-R/L Hardwood Round	Wooden Dowel	supports for housing	73893780014	$ 0.98	3	Home Depot			$ 2.94	☑	11/21/23	Aalaya Wudaru	$ 2.94	https://drive.google.com/file/d/10997LaYjCNOvp1m6By0z49kmysnSdffo/view
18-8 Stainless Steel Socket Head Screw, M3 x 0.5 mm Thread, 8 mm Long, Packs of 100	Socket Head Screw	To fasten system and housing together	91292A112	$ 5.45	1	McMaster-Carr	McMaster-Carr		$ 5.45	☑	10/25/23	Zachary Tam	$ 6.99	https://drive.google.com/file/d/1meLyYp1gfQDr08AYR0d5Bz3iAq3w0yZG/view?usp=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	$ 2.62	1	McMaster-Carr	https://www.m		$ 2.62	☑	10/25/23	Zachary Tam	$ 3.36	https://drive.google.com/file/d/1meLyYp1gfQDr08AYR0d5Bz3iAq3w0yZG/view?usp=drive_link
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	$ 11.09	2	McMaster-Carr	https://www.m		$ 22.18	☑	10/25/23	Zachary Tam	$ 28.43	https://drive.google.com/file/d/1meLyYp1gfQDr08AYR0d5Bz3iAq3w0yZG/view?usp=drive_link
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	$ 2.17	6	McMaster-Carr	https://www.m		$ 13.02	☑	10/25/23	Zachary Tam	$ 16.69	https://drive.google.com/file/d/1meLyYp1gfQDr08AYR0d5Bz3iAq3w0yZG/view?usp=drive_link
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	$ 4.42	1	McMaster-Carr	https://www.m		$ 4.42	☑	10/25/23	Zachary Tam	$ 5.67	https://drive.google.com/file/d/1meLyYp1gfQDr08AYR0d5Bz3iAq3w0yZG/view?usp=drive_link
Braided Fishing Line, Abrasion Resistant	40lb braided fishing line	Tendons; the string that wrapped around the capstan		$ 11.01	1	Amazon	https://www.ar		$ 11.01	☑	11/15/23	Aalaya Wudaru	$ 11.01	https://drive.google.com/file/d/1GAk59-tEoc_5gH7ghtprh3X8HgusXzSS/view?usp=sharing
Dowel - 1/2"x48"	Wooden Dowel	housing supports	95624515511	$ 2.07	1	Home Depot			$ 2.07	☑	12/8/23	Aalaya Wudaru	$ 2.07	https://drive.google.com/file/d/1ahmkR5wxVg_Jm1fyLwxy5JawB40sSBpM/view?usp=sharing
Joystick Sensor Game Controller Sensor JoyStick Breakout Module for Arduino PS2 Raspberry Pi	Analogy Joystick Sensor	For arm control		$ 6.93		Amazon	https://www.ar		$ -	☑	11/7/23	Aalaya Wudaru	$ 6.93	

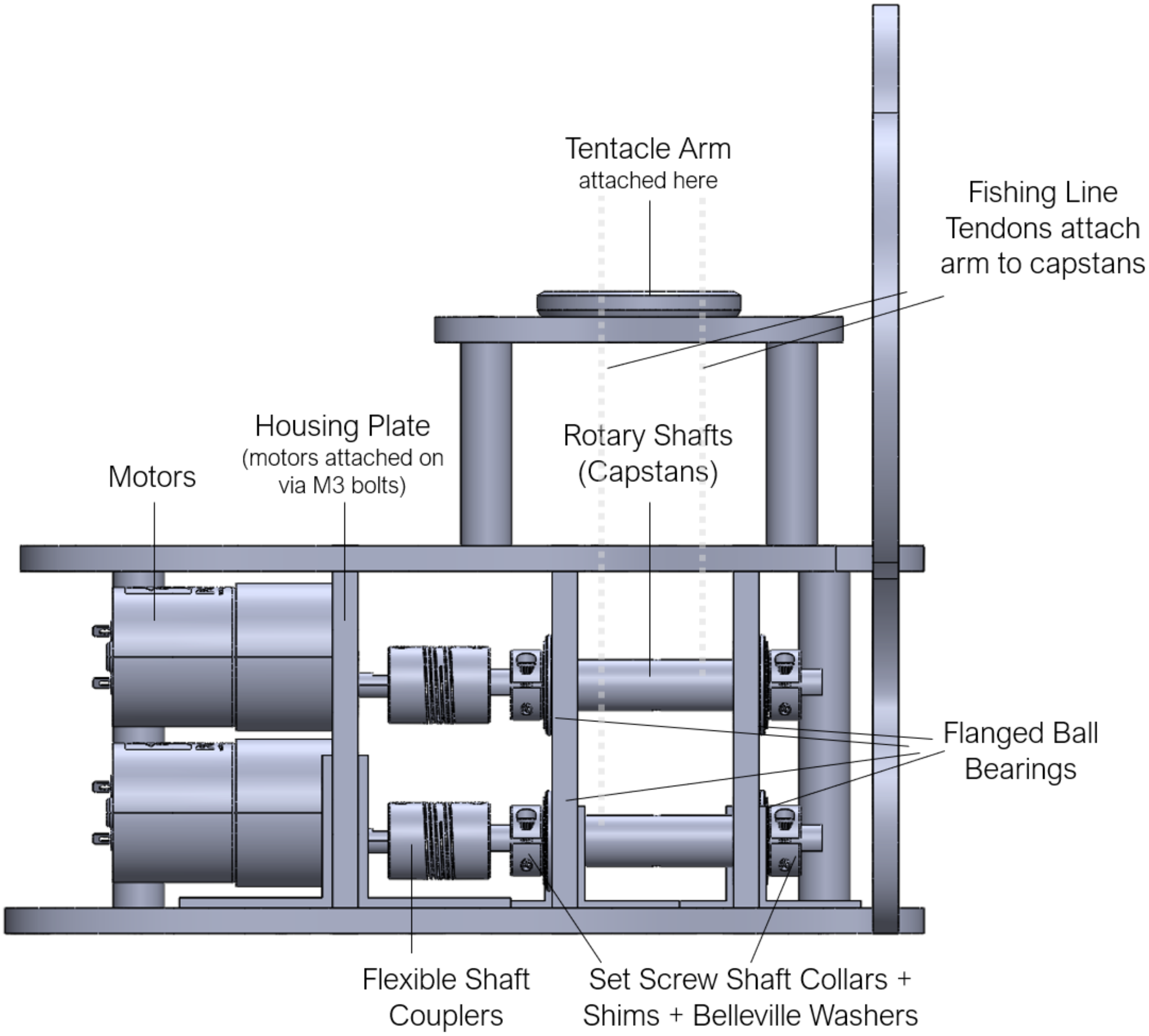
1074-1095 Spring Steel Ring Shim, 0.2mm Thick, 8mm ID, Packs of 50	Ring Shims	Transmission for 3 motors. Make sure shaft collars don't touch both the inner and outer race of the bearings.	98055A112	$ 6.90	1	McMaster-Carr	https://www.m	$ 6.90	☑	10/25/23	Zachary Tam	$ 8.84	https://drive.google.com/file/d/1meLyYp1gfQDr08AYR0d5Bz3iAq3w0yZG/view?usp=drive_link
1.5" Zinc 4pk Corner Brace	1.5" Corner Brace	for housing	30699153046	$ 3.37	2	Home Depot	https://www.h	$ 6.74	☑	11/21/23	Aalaya Wudaru	$ 6.74	https://drive.google.com/file/d/10997LaYjCNOvp1m6By0z49kmysnSdffo/view
1" Zinc 4pk Corner Brace	1" Corner Brace	for housing	30699136193	$ 2.57	2	Home Depot	https://www.h	$ 5.14	☑	11/21/23	Aalaya Wudaru	$ 5.14	https://drive.google.com/file/d/10997LaYjCNOvp1m6By0z49kmysnSdffo/view
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-Ph	$ 6.87	☑	11/21/23	Aalaya Wudaru	$ 6.87	https://drive.google.com/file/d/10997LaYjCNOvp1m6By0z49kmysnSdffo/view
Loctite Ultra Gel Super Glue	Loctite Super Glue	superglue	79340686076	$ 5.68	1	Home Depot	https://www.h	$ 5.68	☑	12/8/23	Aalaya Wudaru		https://drive.google.com/file/d/1ahmkR5wxVg_Jm1fyLwxy5JawB40sSBpM/view?usp=sharing
0.75in x 0.75in x 48in HWD RND	Wooden Dowel	housing supports	728927280038	$ 3.98	1	Home Depot		$ 3.98	☑	12/8/23	Aalaya Wudaru	$ 3.98	https://drive.google.com/file/d/1ahmkR5wxVg_Jm1fyLwxy5JawB40sSBpM/view?usp=sharing

Appendix II: CAD

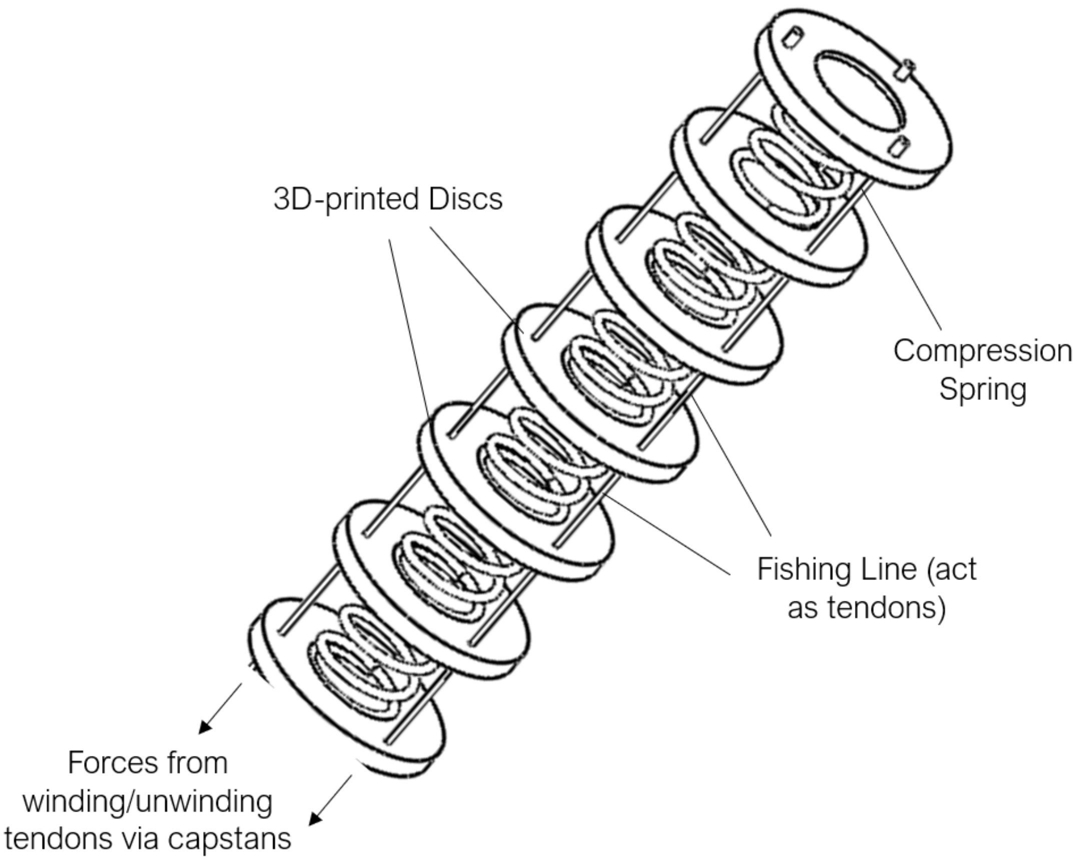
Motor Housing - Isometric View



Motor Housing - Side View

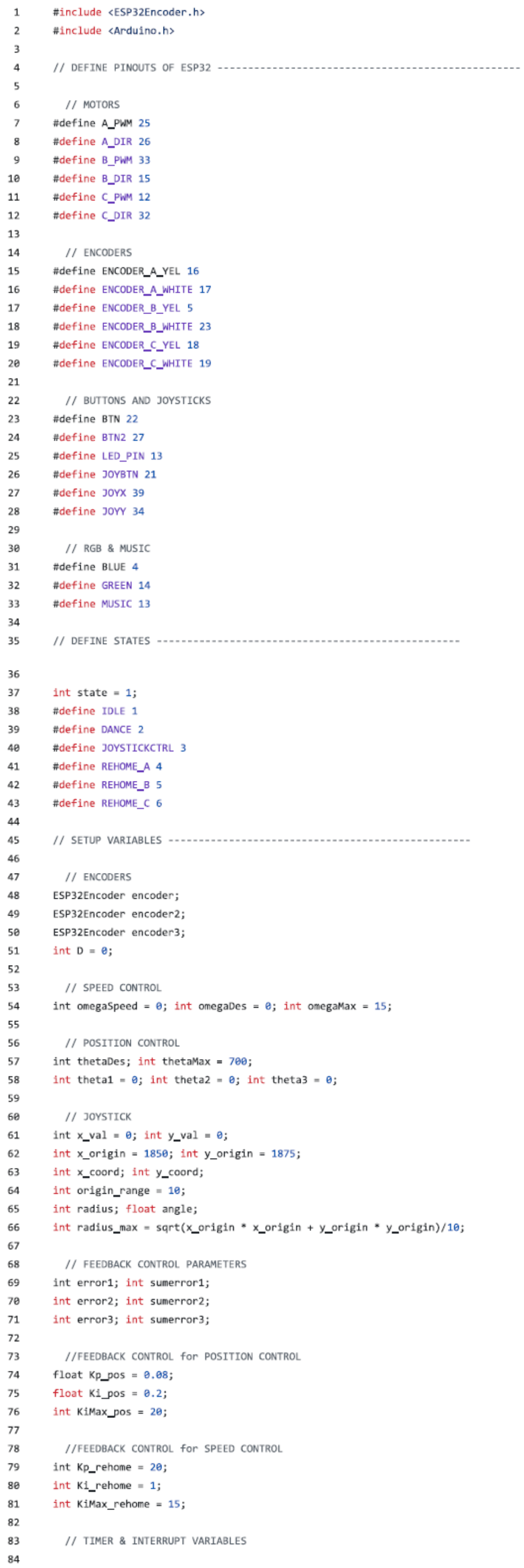


Tentacle Arm

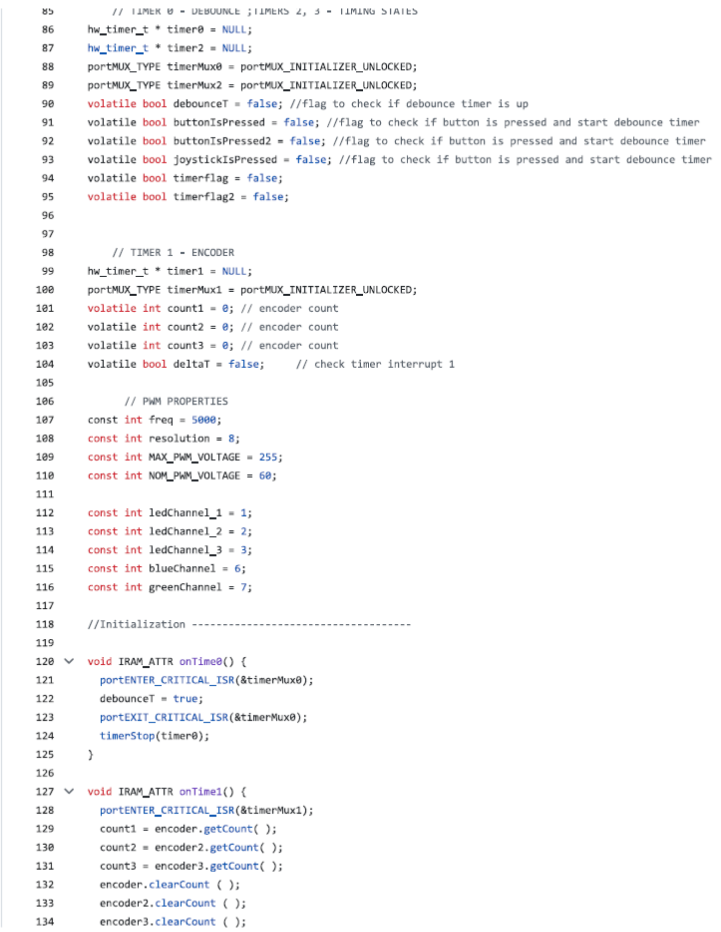

3D-printed Discs
Compression Spring
Fishing Line (act as tendons)
Forces from winding/unwinding tendons via capstans

Appendix III: Code

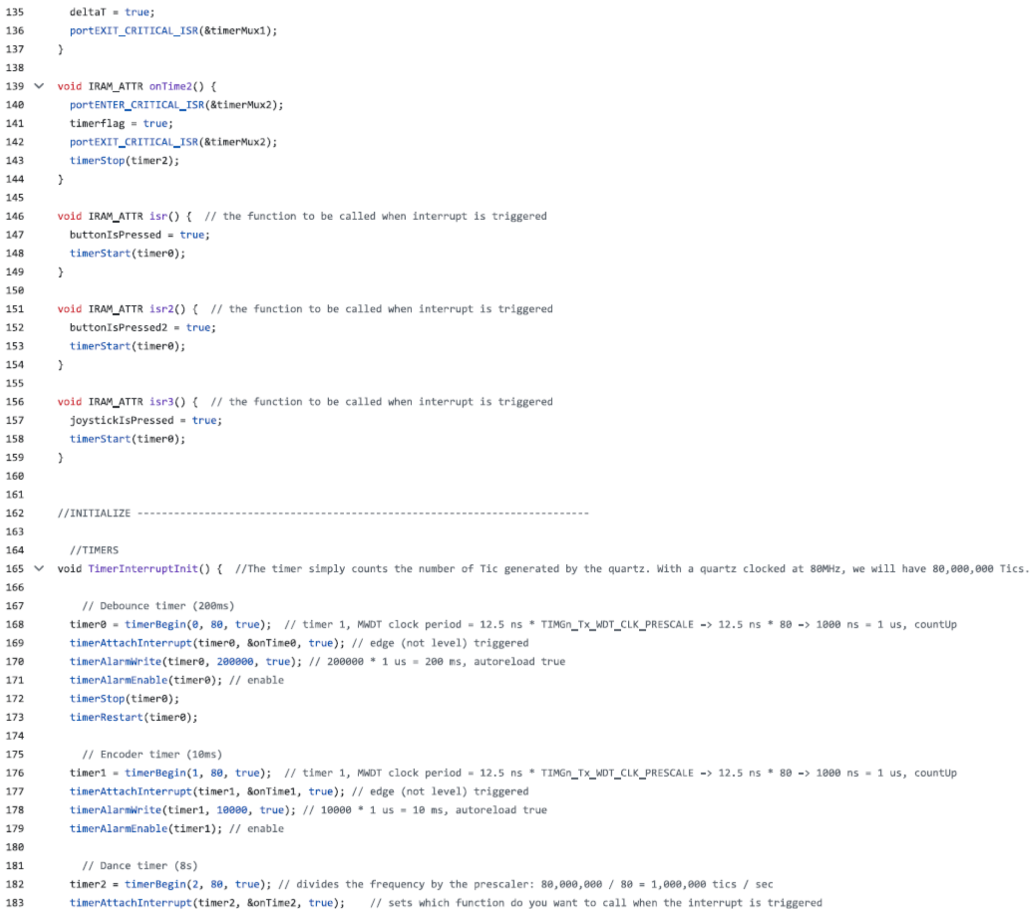
Main System



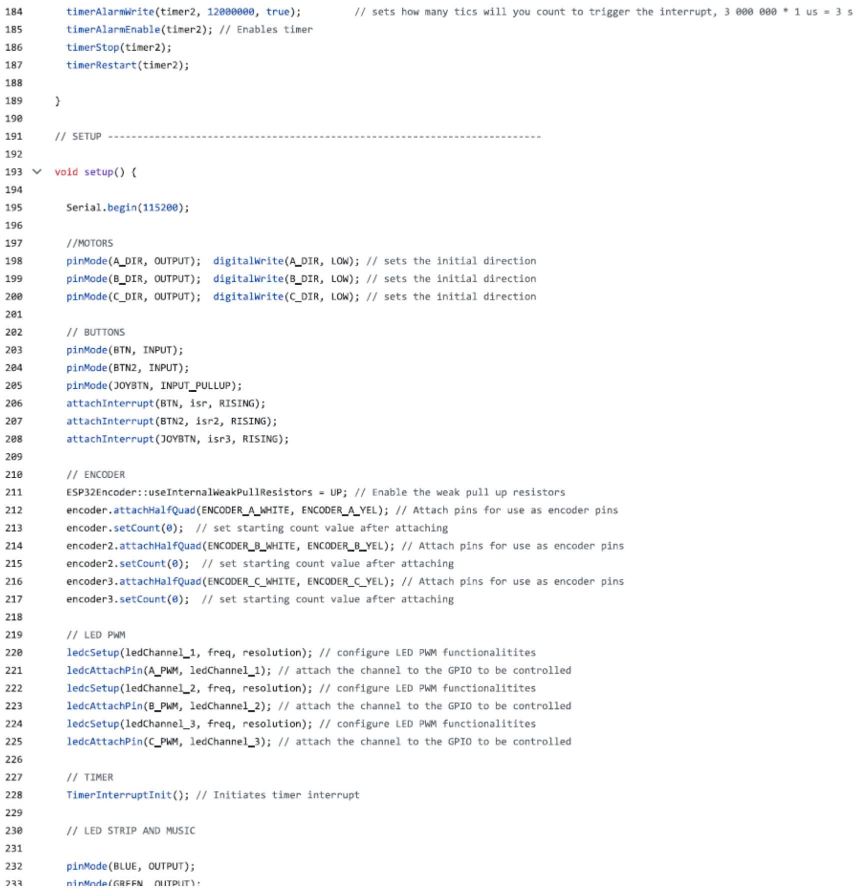
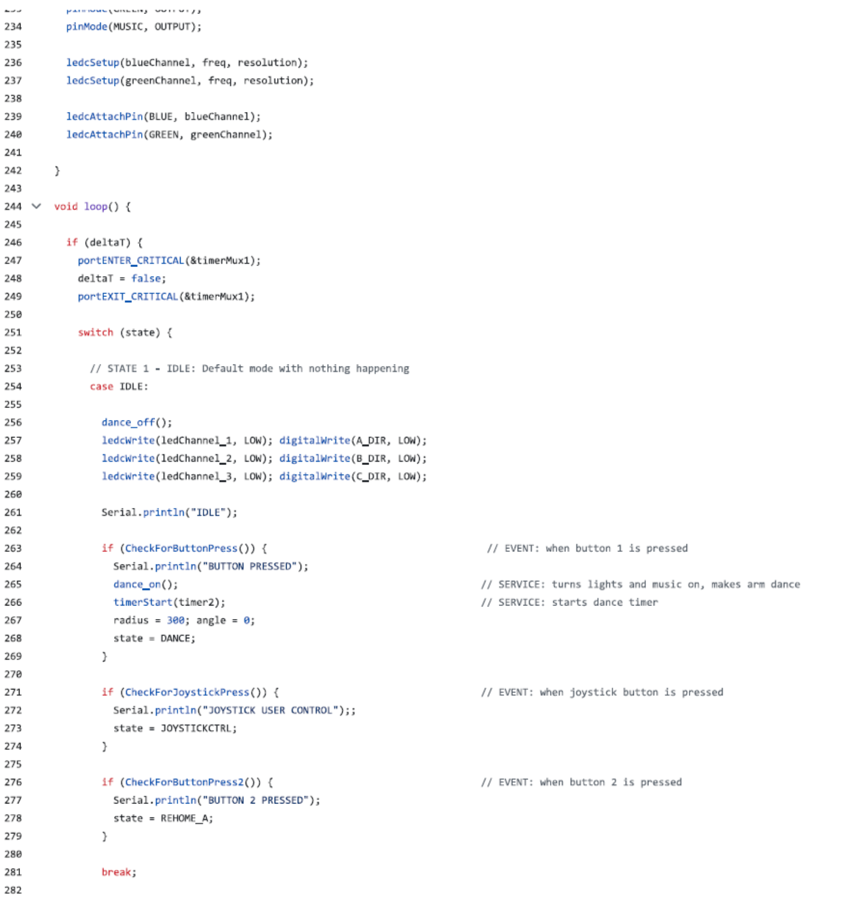
```
#include <ESP32Encoder.h>
#include <Arduino.h>

// DEFINE PINOUTS OF ESP32 --------------------------------------------------

  // MOTORS
#define A_PWM 25
#define A_DIR 26
#define B_PWM 33
#define B_DIR 15
#define C_PWM 12
#define C_DIR 32

  // ENCODERS
#define ENCODER_A_YEL 16
#define ENCODER_A_WHITE 17
#define ENCODER_B_YEL 5
#define ENCODER_B_WHITE 23
#define ENCODER_C_YEL 18
#define ENCODER_C_WHITE 19

  // BUTTONS AND JOYSTICKS
#define BTN 22
#define BTN2 27
#define LED_PIN 13
#define JOYBTN 21
#define JOYX 39
#define JOYY 34

  // RGB & MUSIC
#define BLUE 4
#define GREEN 14
#define MUSIC 13

// DEFINE STATES --------------------------------------------------

int state = 1;
#define IDLE 1
#define DANCE 2
#define JOYSTICKCTRL 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
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;
int x_origin = 1850; int y_origin = 1875;
int x_coord; int y_coord;
int origin_range = 10;
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 error2; int sumerror2;
int error3; int sumerror3;

  //FEEDBACK CONTROL for POSITION CONTROL
float Kp_pos = 0.08;
float Ki_pos = 0.2;
int KiMax_pos = 20;

  //FEEDBACK CONTROL for SPEED CONTROL
int Kp_rehome = 20;
int Ki_rehome = 1;
int KiMax_rehome = 15;

  // TIMER & INTERRUPT VARIABLES

```



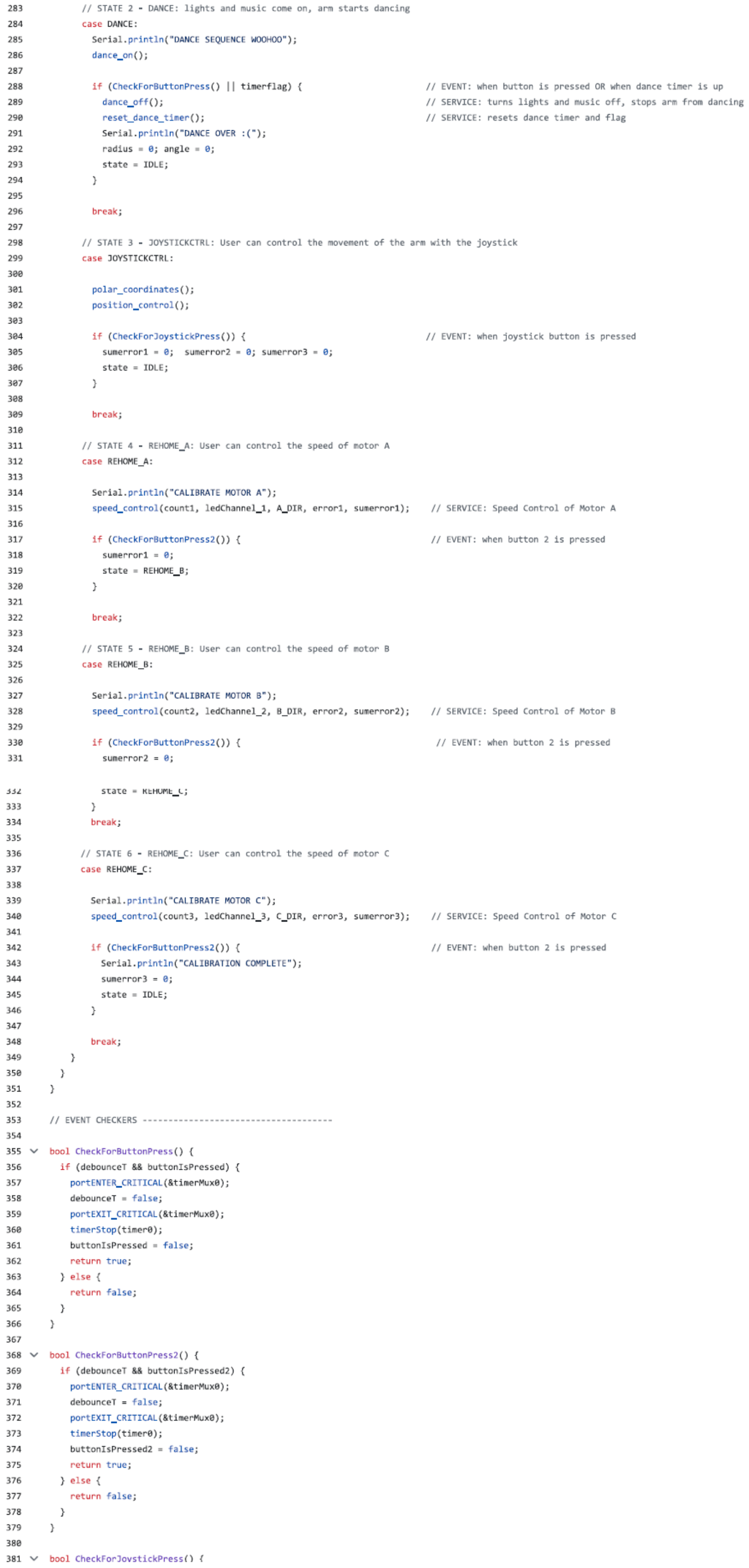
```
    // TIMER 0 - DEBOUNCE ;TIMERS 2, 3 - TIMING STATES
hw_timer_t * timer0 = NULL;
hw_timer_t * timer2 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux2 = portMUX_INITIALIZER_UNLOCKED;
volatile bool debounceT = false; //flag to check if debounce timer is up
volatile bool buttonIsPressed = false; //flag to check if button is pressed and start debounce timer
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
volatile bool timerflag = false;
volatile bool timerflag2 = false;


    // TIMER 1 - ENCODER
hw_timer_t * timer1 = NULL;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;
volatile int count1 = 0; // encoder count
volatile int count2 = 0; // encoder count
volatile int count3 = 0; // encoder count
volatile bool deltaT = false;     // check timer interrupt 1

      // PWM PROPERTIES
const int freq = 5000;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 60;

const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int ledChannel_3 = 3;
const int blueChannel = 6;
const int greenChannel = 7;

//Initialization -----------------------------------

void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  debounceT = true;
  portEXIT_CRITICAL_ISR(&timerMux0);
  timerStop(timer0);
}

void IRAM_ATTR onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
  count1 = encoder.getCount( );
  count2 = encoder2.getCount( );
  count3 = encoder3.getCount( );
  encoder.clearCount ( );
  encoder2.clearCount ( );
  encoder3.clearCount ( );
```



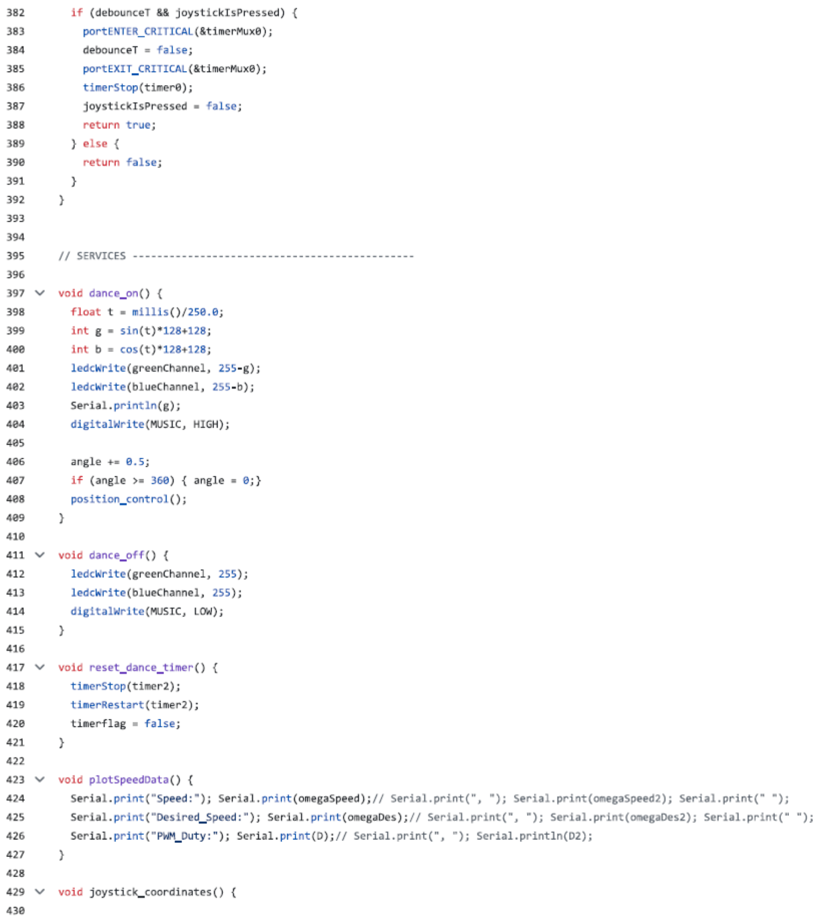
```
  deltaT = true;
  portEXIT_CRITICAL_ISR(&timerMux1);
}

void IRAM_ATTR onTime2() {
  portENTER_CRITICAL_ISR(&timerMux2);
  timerflag = true;
  portEXIT_CRITICAL_ISR(&timerMux2);
  timerStop(timer2);
}

void IRAM_ATTR isr() {  // the function to be called when interrupt is triggered
  buttonIsPressed = true;
  timerStart(timer0);
}

void IRAM_ATTR isr2() {  // the function to be called when interrupt is triggered
  buttonIsPressed2 = true;
  timerStart(timer0);
}

void IRAM_ATTR isr3() {  // the function to be called when interrupt is triggered
  joystickIsPressed = true;
  timerStart(timer0);
}


//INITIALIZE ----------------------------------------------------------------------

  //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.

    // Debounce timer (200ms)
  timer0 = timerBegin(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
  timerAttachInterrupt(timer0, &onTime0, true); // edge (not level) triggered
  timerAlarmWrite(timer0, 200000, true); // 200000 * 1 us = 200 ms, autoreload true
  timerAlarmEnable(timer0); // enable
  timerStop(timer0);
  timerRestart(timer0);

    // 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
  timerAttachInterrupt(timer1, &onTime1, true); // edge (not level) triggered
  timerAlarmWrite(timer1, 10000, true); // 10000 * 1 us = 10 ms, autoreload true
  timerAlarmEnable(timer1); // enable

    // Dance timer (8s)
  timer2 = timerBegin(2, 80, true); // divides the frequency by the prescaler: 80,000,000 / 80 = 1,000,000 tics / sec
  timerAttachInterrupt(timer2, &onTime2, true);    // sets which function do you want to call when the interrupt is triggered
```

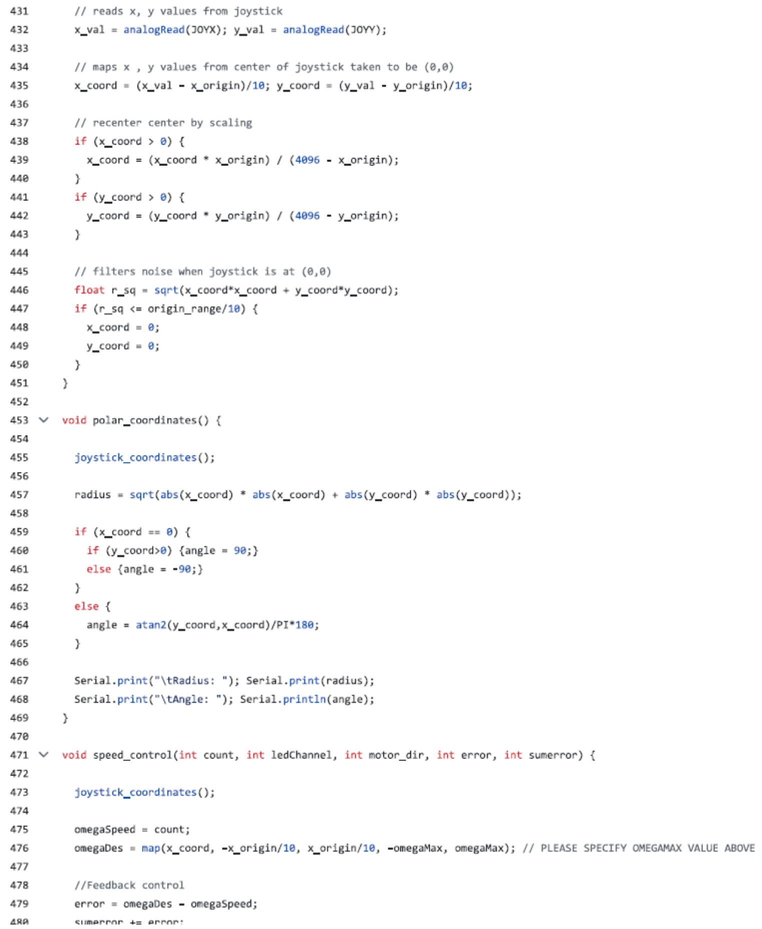
```
  timerAlarmWrite(timer2, 12000000, true);         // sets how many tics will you count to trigger the interrupt, 3 000 000 * 1 us = 3 s
  timerAlarmEnable(timer2); // Enables timer
  timerStop(timer2);
  timerRestart(timer2);

}

// SETUP -----------------------------------------------------------------------

void setup() {

  Serial.begin(115200);

  //MOTORS
  pinMode(A_DIR, OUTPUT);  digitalWrite(A_DIR, LOW); // sets the initial direction
  pinMode(B_DIR, OUTPUT);  digitalWrite(B_DIR, LOW); // sets the initial direction
  pinMode(C_DIR, OUTPUT);  digitalWrite(C_DIR, LOW); // sets the initial direction

  // BUTTONS
  pinMode(BTN, INPUT);
  pinMode(BTN2, INPUT);
  pinMode(JOYBTN, INPUT_PULLUP);
  attachInterrupt(BTN, isr, RISING);
  attachInterrupt(BTN2, isr2, RISING);
  attachInterrupt(JOYBTN, isr3, RISING);

  // ENCODER
  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
  encoder2.attachHalfQuad(ENCODER_B_WHITE, ENCODER_B_YEL); // Attach pins for use as encoder pins
  encoder2.setCount(0);  // set starting count value after attaching
  encoder3.attachHalfQuad(ENCODER_C_WHITE, ENCODER_C_YEL); // Attach pins for use as encoder pins
  encoder3.setCount(0);  // set starting count value after attaching

  // LED PWM
  ledcSetup(ledChannel_1, freq, resolution); // configure LED PWM functionalitites
  ledcAttachPin(A_PWM, ledChannel_1); // attach the channel to the GPIO to be controlled
  ledcSetup(ledChannel_2, freq, resolution); // configure LED PWM functionalitites
  ledcAttachPin(B_PWM, ledChannel_2); // attach the channel to the GPIO to be controlled
  ledcSetup(ledChannel_3, freq, resolution); // configure LED PWM functionalitites
  ledcAttachPin(C_PWM, ledChannel_3); // attach the channel to the GPIO to be controlled

  // TIMER
  TimerInterruptInit(); // Initiates timer interrupt

  // LED STRIP AND MUSIC

  pinMode(BLUE, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(MUSIC, OUTPUT);

  ledcSetup(blueChannel, freq, resolution);
  ledcSetup(greenChannel, freq, resolution);

  ledcAttachPin(BLUE, blueChannel);
  ledcAttachPin(GREEN, greenChannel);

}

void loop() {

  if (deltaT) {
    portENTER_CRITICAL(&timerMux1);
    deltaT = false;
    portEXIT_CRITICAL(&timerMux1);

    switch (state) {

      // STATE 1 - IDLE: Default mode with nothing happening
      case IDLE:

        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);

        Serial.println("IDLE");

        if (CheckForButtonPress()) {                                      // EVENT: when button 1 is pressed
          Serial.println("BUTTON PRESSED");
          dance_on();                                                    // SERVICE: turns lights and music on, makes arm dance
          timerStart(timer2);                                            // SERVICE: starts dance timer
          radius = 300; angle = 0;
          state = DANCE;
        }

        if (CheckForJoystickPress()) {                                   // EVENT: when joystick button is pressed
          Serial.println("JOYSTICK USER CONTROL");;
          state = JOYSTICKCTRL;
        }

        if (CheckForButtonPress2()) {                                    // EVENT: when button 2 is pressed
          Serial.println("BUTTON 2 PRESSED");
          state = REHOME_A;
        }

        break;

```



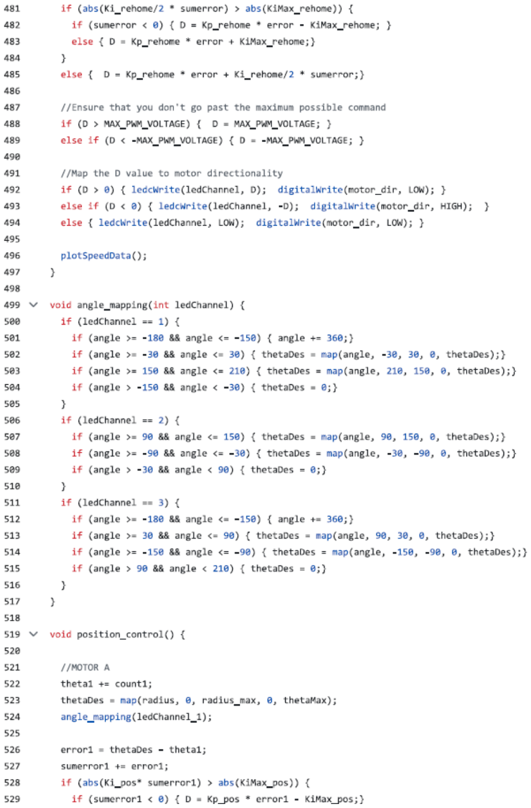
```
      // STATE 2 - DANCE: lights and music come on, arm starts dancing
      case DANCE:
        Serial.println("DANCE SEQUENCE WOOHOO");
        dance_on();

        if (CheckForButtonPress() || timerflag) {                        // EVENT: when button is pressed OR when dance timer is up
          dance_off();                                                   // SERVICE: turns lights and music off, stops arm from dancing
          reset_dance_timer();                                           // SERVICE: resets dance timer and flag
          Serial.println("DANCE OVER :(");
          radius = 0; angle = 0;
          state = IDLE;
        }

        break;

      // STATE 3 - JOYSTICKCTRL: User can control the movement of the arm with the joystick
      case JOYSTICKCTRL:

        polar_coordinates();
        position_control();

        if (CheckForJoystickPress()) {                                   // EVENT: when joystick button is pressed
          sumerror1 = 0;  sumerror2 = 0; sumerror3 = 0;
          state = IDLE;
        }

        break;

      // STATE 4 - REHOME_A: User can control the speed of motor A
      case REHOME_A:

        Serial.println("CALIBRATE MOTOR A");
        speed_control(count1, ledChannel_1, A_DIR, error1, sumerror1);    // SERVICE: Speed Control of Motor A

        if (CheckForButtonPress2()) {                                     // EVENT: when button 2 is pressed
          sumerror1 = 0;
          state = REHOME_B;
        }

        break;

      // STATE 5 - REHOME_B: User can control the speed of motor B
      case REHOME_B:

        Serial.println("CALIBRATE MOTOR B");
        speed_control(count2, ledChannel_2, B_DIR, error2, sumerror2);    // SERVICE: Speed Control of Motor B

        if (CheckForButtonPress2()) {                                      // EVENT: when button 2 is pressed
          sumerror2 = 0;

          state = REHOME_C;
        }
        break;

      // STATE 6 - REHOME_C: User can control the speed of motor C
      case REHOME_C:

        Serial.println("CALIBRATE MOTOR C");
        speed_control(count3, ledChannel_3, C_DIR, error3, sumerror3);    // SERVICE: Speed Control of Motor C

        if (CheckForButtonPress2()) {                                     // EVENT: when button 2 is pressed
          Serial.println("CALIBRATION COMPLETE");
          sumerror3 = 0;
          state = IDLE;
        }

        break;
    }
  }
}

// EVENT CHECKERS -------------------------------------

bool CheckForButtonPress() {
  if (debounceT && buttonIsPressed) {
    portENTER_CRITICAL(&timerMux0);
    debounceT = false;
    portEXIT_CRITICAL(&timerMux0);
    timerStop(timer0);
    buttonIsPressed = false;
    return true;
  } else {
    return false;
  }
}

bool CheckForButtonPress2() {
  if (debounceT && buttonIsPressed2) {
    portENTER_CRITICAL(&timerMux0);
    debounceT = false;
    portEXIT_CRITICAL(&timerMux0);
    timerStop(timer0);
    buttonIsPressed2 = false;
    return true;
  } else {
    return false;
  }
}

bool CheckForJoystickPress() {
```



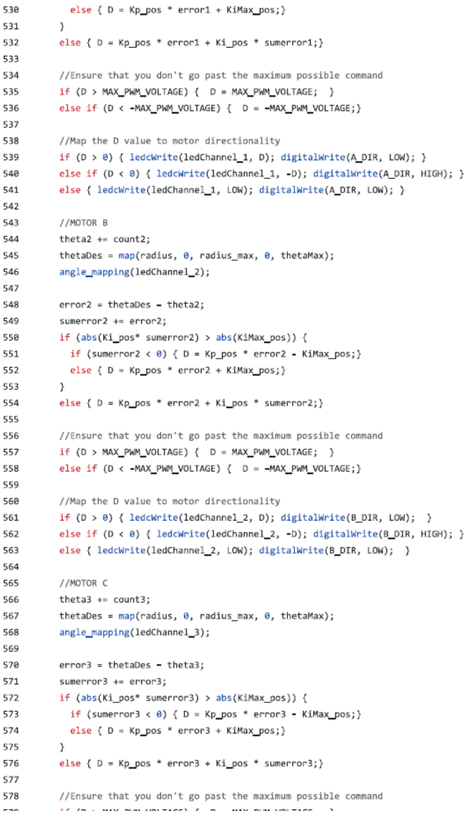
```
  if (debounceT && joystickIsPressed) {
    portENTER_CRITICAL(&timerMux0);
    debounceT = false;
    portEXIT_CRITICAL(&timerMux0);
    timerStop(timer0);
    joystickIsPressed = false;
    return true;
  } else {
    return false;
  }
}


// SERVICES ---------------------------------------------

void dance_on() {
  float t = millis()/250.0;
  int g = sin(t)*128+128;
  int b = cos(t)*128+128;
  ledcWrite(greenChannel, 255-g);
  ledcWrite(blueChannel, 255-b);
  Serial.println(g);
  digitalWrite(MUSIC, HIGH);

  angle += 0.5;
  if (angle >= 360) { angle = 0;}
  position_control();
}

void dance_off() {
  ledcWrite(greenChannel, 255);
  ledcWrite(blueChannel, 255);
  digitalWrite(MUSIC, LOW);
}

void reset_dance_timer() {
  timerStop(timer2);
  timerRestart(timer2);
  timerflag = false;
}

void plotSpeedData() {
  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(" ");
  Serial.print("PWM_Duty:"); Serial.print(D);// Serial.print(", "); Serial.println(D2);
}

void joystick_coordinates() {

```



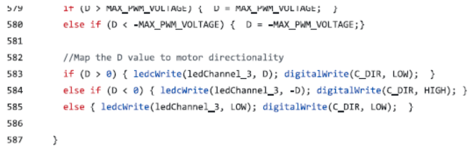
```
  // reads x, y values from joystick
  x_val = analogRead(JOYX); y_val = analogRead(JOYY);

  // maps x , y values from center of joystick taken to be (0,0)
  x_coord = (x_val - x_origin)/10; y_coord = (y_val - y_origin)/10;

  // recenter center by scaling
  if (x_coord > 0) {
    x_coord = (x_coord * x_origin) / (4096 - x_origin);
  }
  if (y_coord > 0) {
    y_coord = (y_coord * y_origin) / (4096 - y_origin);
  }

  // filters noise when joystick is at (0,0)
  float r_sq = sqrt(x_coord*x_coord + y_coord*y_coord);
  if (r_sq <= origin_range/10) {
    x_coord = 0;
    y_coord = 0;
  }
}

void polar_coordinates() {

  joystick_coordinates();

  radius = sqrt(abs(x_coord) * abs(x_coord) + abs(y_coord) * abs(y_coord));

  if (x_coord == 0) {
    if (y_coord>0) {angle = 90;}
    else {angle = -90;}
  }
  else {
    angle = atan2(y_coord,x_coord)/PI*180;
  }

  Serial.print("\tRadius: "); Serial.print(radius);
  Serial.print("\tAngle: "); Serial.println(angle);
}

void speed_control(int count, int ledChannel, int motor_dir, int error, int sumerror) {

  joystick_coordinates();

  omegaSpeed = count;
  omegaDes = map(x_coord, -x_origin/10, x_origin/10, -omegaMax, omegaMax); // PLEASE SPECIFY OMEGAMAX VALUE ABOVE

  //Feedback control
  error = omegaDes - omegaSpeed;
  sumerror += error;
```



```
  if (abs(Ki_rehome/2 * sumerror) > abs(KiMax_rehome)) {
    if (sumerror < 0) { D = Kp_rehome * error - KiMax_rehome; }
    else { D = Kp_rehome * error + KiMax_rehome;}
  }
  else {  D = Kp_rehome * error + Ki_rehome/2 * sumerror;}

  //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; }

  //Map the D value to motor directionality
  if (D > 0) { ledcWrite(ledChannel, D);  digitalWrite(motor_dir, LOW); }
  else if (D < 0) { ledcWrite(ledChannel, -D);  digitalWrite(motor_dir, HIGH);  }
  else { ledcWrite(ledChannel, LOW);  digitalWrite(motor_dir, LOW); }

  plotSpeedData();
}

void angle_mapping(int ledChannel) {
  if (ledChannel == 1) {
    if (angle >= -180 && angle <= -150) { angle += 360;}
    if (angle >= -30 && angle <= 30) { thetaDes = map(angle, -30, 30, 0, thetaDes);}
    if (angle >= 150 && angle <= 210) { thetaDes = map(angle, 210, 150, 0, thetaDes);}
    if (angle > -150 && angle < -30) { thetaDes = 0;}
  }
  if (ledChannel == 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);}
    if (angle > -30 && angle < 90) { thetaDes = 0;}
  }
  if (ledChannel == 3) {
    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);}
    if (angle > 90 && angle < 210) { thetaDes = 0;}
  }
}

void position_control() {

  //MOTOR A
  theta1 += count1;
  thetaDes = map(radius, 0, radius_max, 0, thetaMax);
  angle_mapping(ledChannel_1);

  error1 = thetaDes - theta1;
  sumerror1 += error1;
  if (abs(Ki_pos* sumerror1) > abs(KiMax_pos)) {
    if (sumerror1 < 0) { D = Kp_pos * error1 - KiMax_pos;}
```

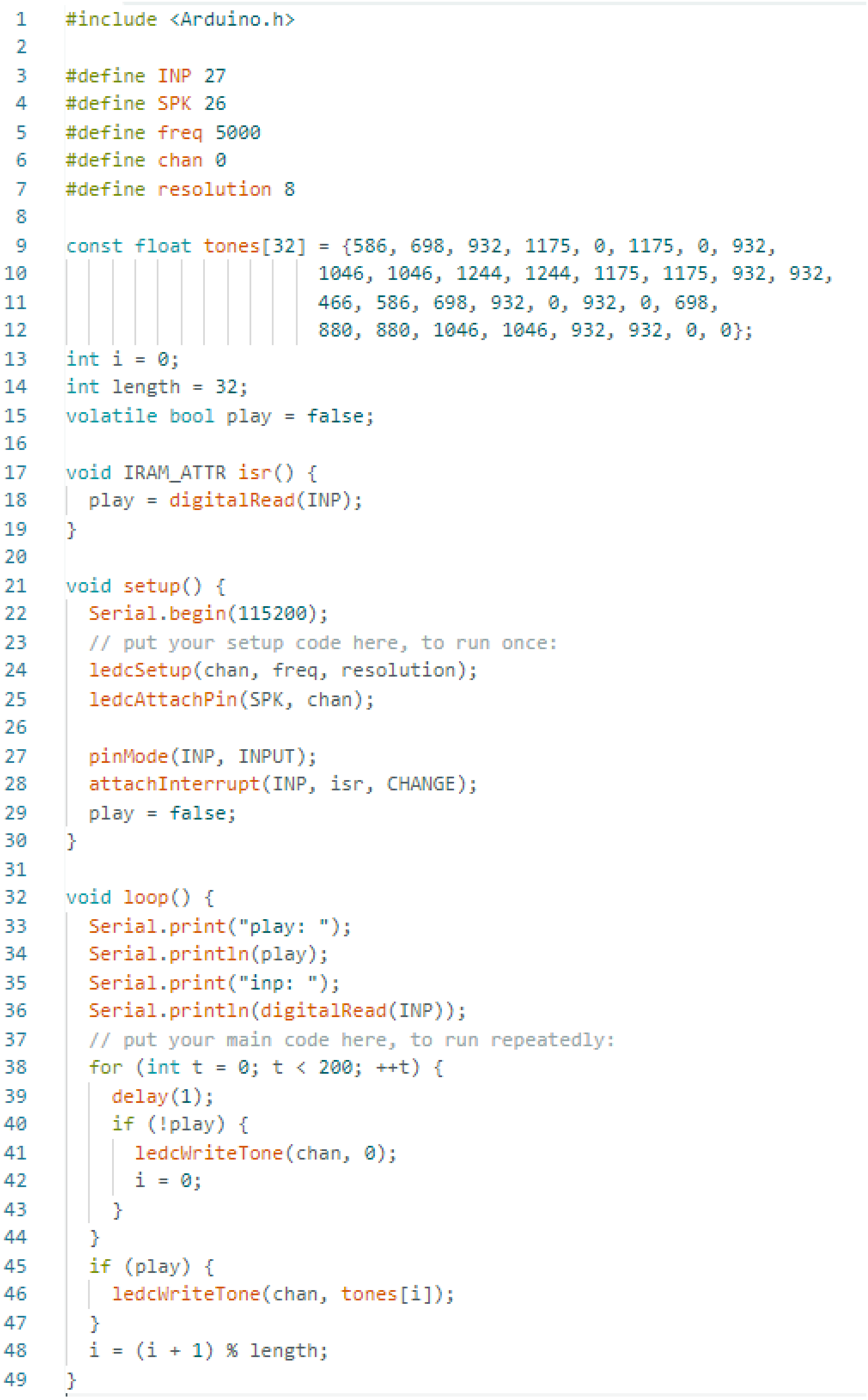


```
    else { D = Kp_pos * error1 + KiMax_pos;}
  }
  else { D = Kp_pos * error1 + Ki_pos * sumerror1;}

  //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;}

  //Map the D value to motor directionality
  if (D > 0) { ledcWrite(ledChannel_1, D); digitalWrite(A_DIR, LOW); }
  else if (D < 0) { ledcWrite(ledChannel_1, -D); digitalWrite(A_DIR, HIGH); }
  else { ledcWrite(ledChannel_1, LOW); digitalWrite(A_DIR, LOW); }

  //MOTOR B
  theta2 += count2;
  thetaDes = map(radius, 0, radius_max, 0, thetaMax);
  angle_mapping(ledChannel_2);

  error2 = thetaDes - theta2;
  sumerror2 += error2;
  if (abs(Ki_pos* sumerror2) > abs(KiMax_pos)) {
    if (sumerror2 < 0) { D = Kp_pos * error2 - KiMax_pos;}
    else { D = Kp_pos * error2 + KiMax_pos;}
  }
  else { D = Kp_pos * error2 + Ki_pos * sumerror2;}

  //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;}

  //Map the D value to motor directionality
  if (D > 0) { ledcWrite(ledChannel_2, D); digitalWrite(B_DIR, LOW);  }
  else if (D < 0) { ledcWrite(ledChannel_2, -D); digitalWrite(B_DIR, HIGH); }
  else { ledcWrite(ledChannel_2, LOW); digitalWrite(B_DIR, LOW);  }

  //MOTOR C
  theta3 += count3;
  thetaDes = map(radius, 0, radius_max, 0, thetaMax);
  angle_mapping(ledChannel_3);

  error3 = thetaDes - theta3;
  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;}
  }
  else { D = Kp_pos * error3 + Ki_pos * sumerror3;}

  //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;}

  //Map the D value to motor directionality
  if (D > 0) { ledcWrite(ledChannel_3, D); digitalWrite(C_DIR, LOW);  }
  else if (D < 0) { ledcWrite(ledChannel_3, -D); digitalWrite(C_DIR, HIGH); }
  else { ledcWrite(ledChannel_3, LOW); digitalWrite(C_DIR, LOW);  }

}
```

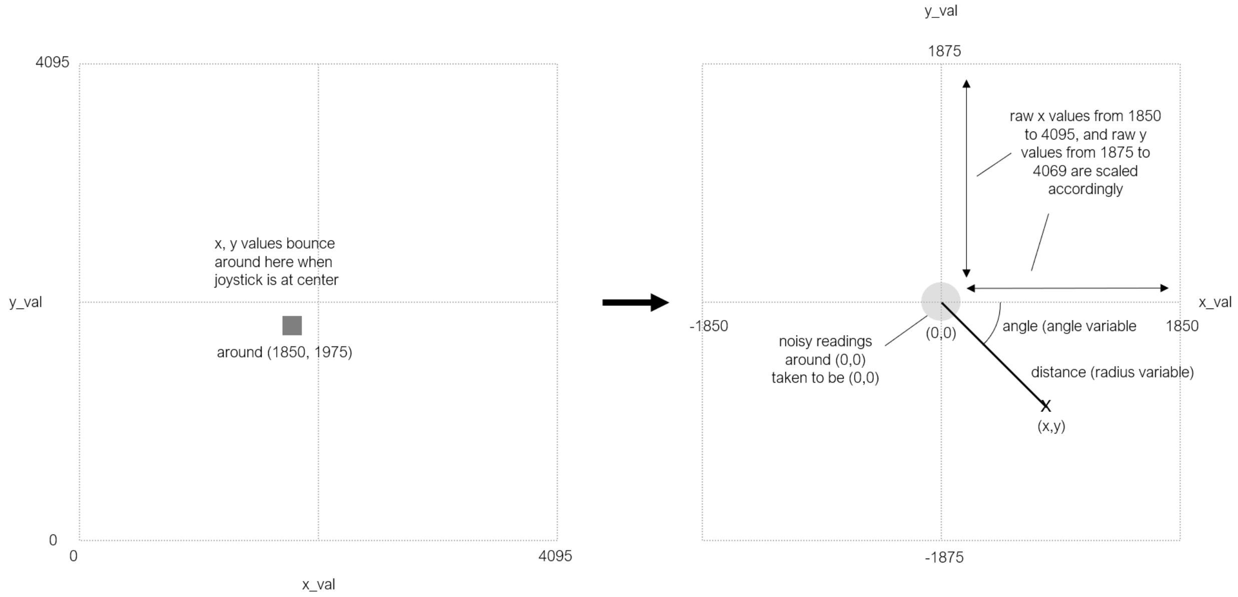
Music Peripheral



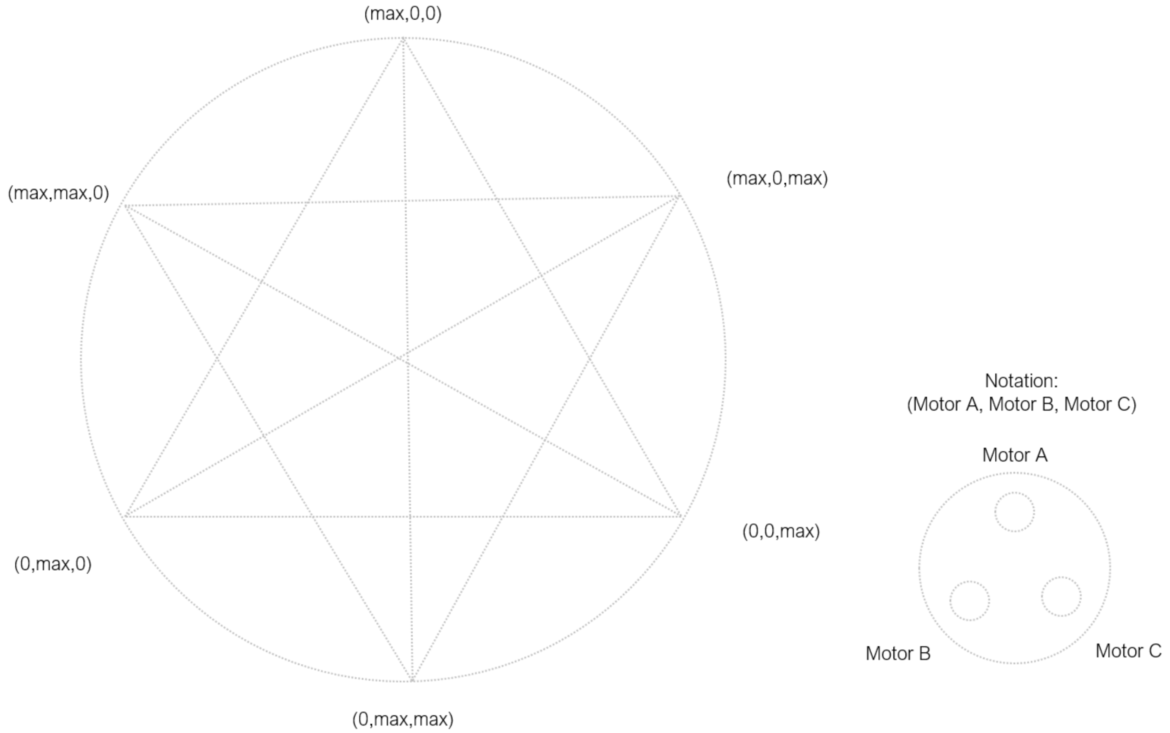
```
#include <Arduino.h>

#define INP 27
#define SPK 26
#define freq 5000
#define chan 0
#define resolution 8

const float tones[32] = {586, 698, 932, 1175, 0, 1175, 0, 932,
                      1046, 1046, 1244, 1244, 1175, 1175, 932, 932,
                      466, 586, 698, 932, 0, 932, 0, 698,
                      880, 880, 1046, 1046, 932, 932, 0, 0};
int i = 0;
int length = 32;
volatile bool play = false;

void IRAM_ATTR isr() {
  play = digitalRead(INP);
}

void setup() {
  Serial.begin(115200);
  // put your setup code here, to run once:
  ledcSetup(chan, freq, resolution);
  ledcAttachPin(SPK, chan);

  pinMode(INP, INPUT);
  attachInterrupt(INP, isr, CHANGE);
  play = false;
}

void loop() {
  Serial.print("play: ");
  Serial.println(play);
  Serial.print("inp: ");
  Serial.println(digitalRead(INP));
  // put your main code here, to run repeatedly:
  for (int t = 0; t < 200; ++t) {
    delay(1);
    if (!play) {
      ledcWriteTone(chan, 0);
      i = 0;
    }
  }
  if (play) {
    ledcWriteTone(chan, tones[i]);
  }
  i = (i + 1) % length;
}
```

Appendix IV: Mapping of motors to joystick position considerations

Preparing joystick data and converting it to polar coordinates


4095
y_val
x, y values bounce around here when joystick is at center
around (1850, 1975)
0
0
4095
x_val
y_val
1875
raw x values from 1850 to 4095, and raw y values from 1875 to 4069 are scaled accordingly
x_val
-1850
noisy readings around (0,0) taken to be (0,0)
(0,0)
angle (angle variable
1850
distance (radius variable)
(x,y)
-1875

Motor Mapping to position


(max,0,0)
(max,max,0)
(max,0,max)
(0,max,0)
(0,0,max)
(0,max,max)
Notation:
(Motor A, Motor B, Motor C)
Motor A
Motor B
Motor C