Team 11 Sauce-EZ

ME 102b: Mechatronics Design

Kyle Nelson, Yen Do, Derrick Ma, Sohan Jayanth

12.14.2023
University of California, Berkeley

INTRODUCTION

OPPORTUNITY

Our project aims to help chefs be more efficient by decreasing unnecessary movement. Chefs must navigate a crowded and chaotic kitchen environment to acquire each new condiment or sauce they need, while leaving burners unattended. Our goal is to design a wearable sauce dispenser providing sauce at the push of a button, without needing to leave the stove. This novel assistive wearable device will greatly streamline the cooking process and allow chefs to cook to the fullest of their ability.

HIGH-LEVEL STRATEGY

Our product aims to provide a modular dispenser that can be strapped onto the user's body with tubes running along the user's arm. All mechanical components are housed in a single 3 in x 2 in x 7.5 in box, along with our microcontroller and motor driver. Tubes and wires are sewn into the sleeve of a chef's jacket for dispensation and control, and to stay out of the way during regular cooking actions. Based on standard oil usage during cooking, our desired functionality was an output of 1 mL of oil per second, which we were able to exceed.

DESIGN

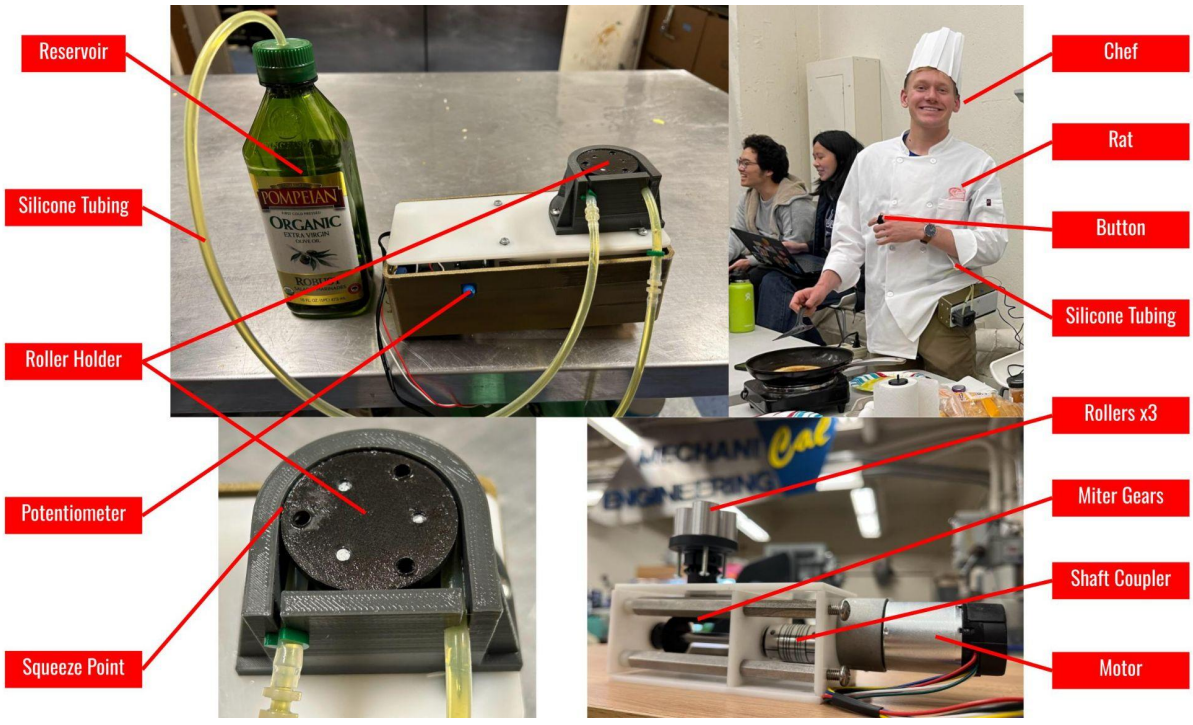


Figure 1: Labeled pictures of fully-integrated device and subsystems

HARDWARE

REQUIRED SPECS AND CALCULATIONS

Based on the 1.5 in diameter of our roller assembly, we expect a 1.5 in * π * 25.4 mm/in = **120 mm** length of tube to be emptied per revolution. The tube has a cross-sectional area of π * (1.5 mm radius)^2 ≈ **7 mm^2**. Each revolution pumps 120 mm * 7 mm^2 * 0.001 mL/mm^2 = **0.84 mL** of liquid. With our desired pumping capability of of 1 mL per second, we are left with a requirement of 1 mL/s ÷ 0.84 mL/revolution * 60 s/min ≈ **71 rpm**. Additionally, based on pumps with similar specifications, we expect a maximum torque requirement of less than 4 kg-cm. Thus, we selected a DC motor with a no-load speed of 251 rpm and a torque of 18 kg-cm, which is more than adequate.

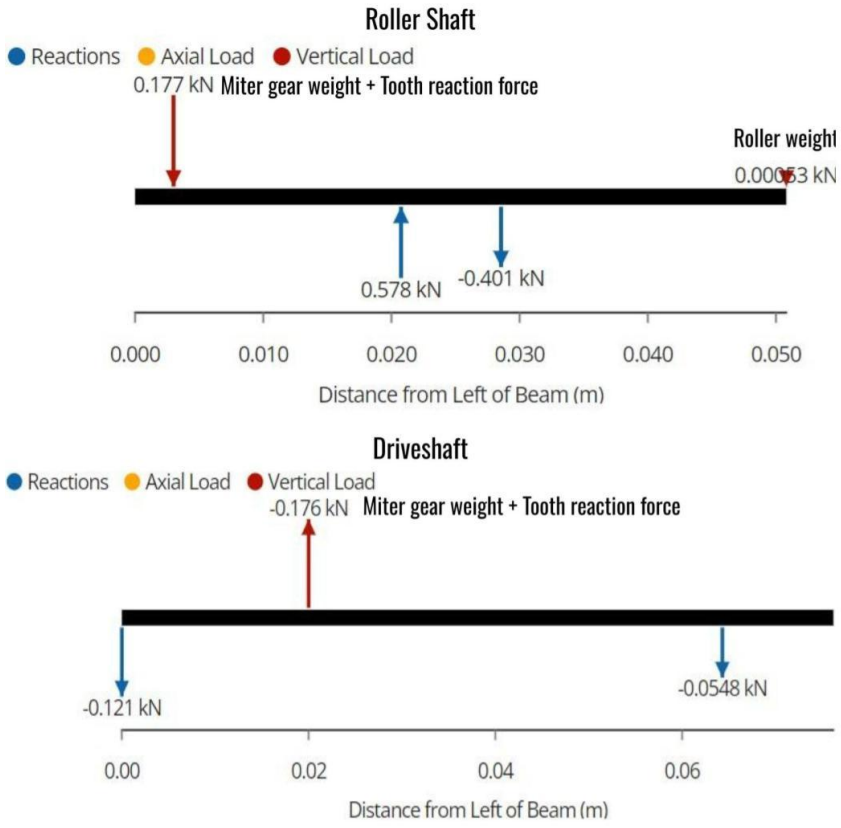


Figure 2: Support force calculations for bearings

The gear forces from the motor at maximum torque are by far the largest supported by our bearings, and even then they do not exceed 600 N or 135 lbf. Our bearings are rated for radial loads of up to 330 lbs, and we do not expect to ever need the maximum amount of torque either.

ELECTRONICS AND SOFTWARE

CIRCUIT DIAGRAM

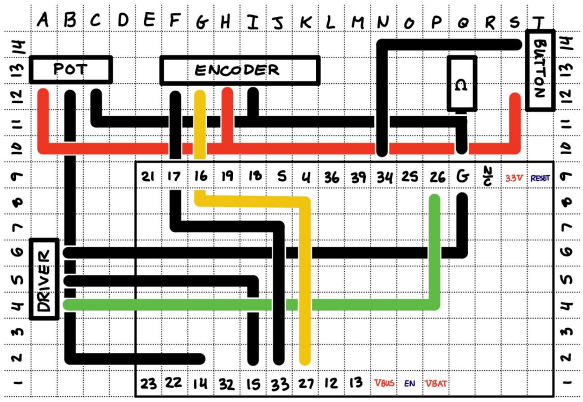


Figure 3: Circuit diagram of Sauce-EZ dispenser

STATE TRANSITION DIAGRAM

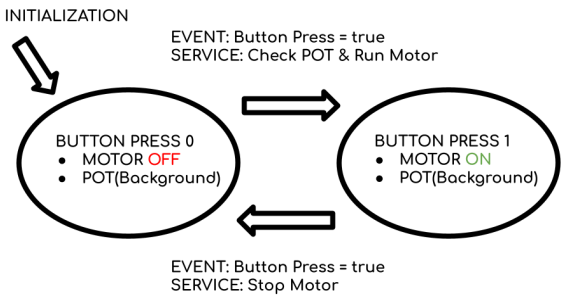


Figure 4: State transition diagram for Sauce-EZ dispenser

To adhere to our state transition diagram, a basic switch statement is used in the loop( ) body of the code: the current state is checked and the appropriate body of code is jumped to, then the program checks for any button input. If a button input is detected, a debounce timer starts to ensure that the ESP32 does not register any chatter, and the state is switched. If not already running, the input switches the motor on, and vice versa, allowing the same button to be used to activate and deactivate the pump.

P-I control is implemented such that any errors in the desired speed and measured speed are corrected by appropriately increasing or decreasing the PWM duty cycle that is being sent to the motor controller. The proportional and integral response terms were carefully tuned and selected based on the performance we were after. Once the tuned duty cycle is calculated, it is written to the PWM pin and sent to the motor controller.

CONCLUSION

Despite tight deadlines, our team developed a product that exceeded project requirements. If done again, materials would have been ordered in advance and more time would have been allocated for fabrication. Additionally, we would have performed more testing with low-fidelity prototypes. The design can be improved by shortening the driveshaft and optimizing roller size for better pumping. With a slimmer form factor and the addition of multiple sauce outputs, the Sauce-EZ could revolutionize the restaurant industry.

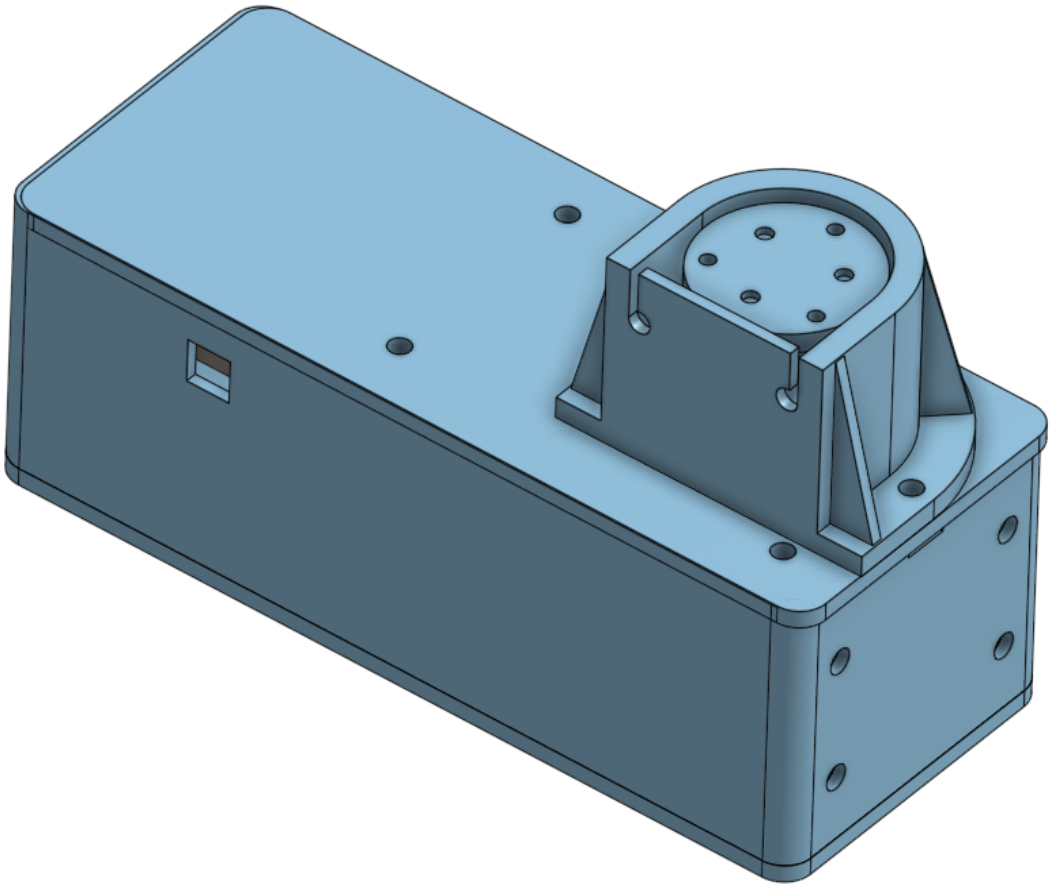
APPENDIX A: BILL OF MATERIALS

Listing Name/Part	Quantity	Cost	Vendor and Link
Chef's Jacket	1	$17.49	Amazon https://www.amazon.com/dp/B00CV3E8I4?psc=1&ref=ppx_yo2ov_dt_b_product_details
Gearmotor with Encoder	1	$29.00	DFROBOT https://www.dfrobot.com/product-634.html
Standoff: Motor - Bearing	4	26.32	McMaster https://www.mcmaster.com/91115A940/
Standoff: Bearing 1 - Bearing 2	4	29.48	McMaster https://www.mcmaster.com/91075A254/
Nylon Spacers	4	$14.33	McMaster https://www.mcmaster.com/94639A448/
Silicone Tubing, 2 x 4mm BPT Peristaltic Pump	1	$22.67	Amazon https://www.amazon.com/gp/product/B0CFDNJ56Z/ref=ewc_pr_img_1?smid=A37GDVPIINVQ03&psc=1
Clear Acrylic - 1/8" x 16" x 32"	1	$27.60	Jacobs Material Store https://store.jacobshall.org/products/clear-acrylic-1-8-x-16-x-32
PLA 3D Print Filament	N/A	$0.22	Jacobs Material Store https://jacobsinstitute.berkeley.edu/jacobs-self-service-printing/
3in Stainless Steel Shaft 1/4" Diameter	1	$8.13	McMaster https://www.mcmaster.com/1263K173/
2in Stainless Steel Shaft 1/4" Diameter	1	$20.71	McMaster https://www.mcmaster.com/1162K125/
Flexible Shaft Coupler	1	$61.67	McMaster https://www.mcmaster.com/2464K2/
Flange Mount Shaft Collar	1	$48.81	McMaster https://www.mcmaster.com/9684T27/
Flanged Ball Bearings	4	25.68	McMaster https://www.mcmaster.com/57155K305/

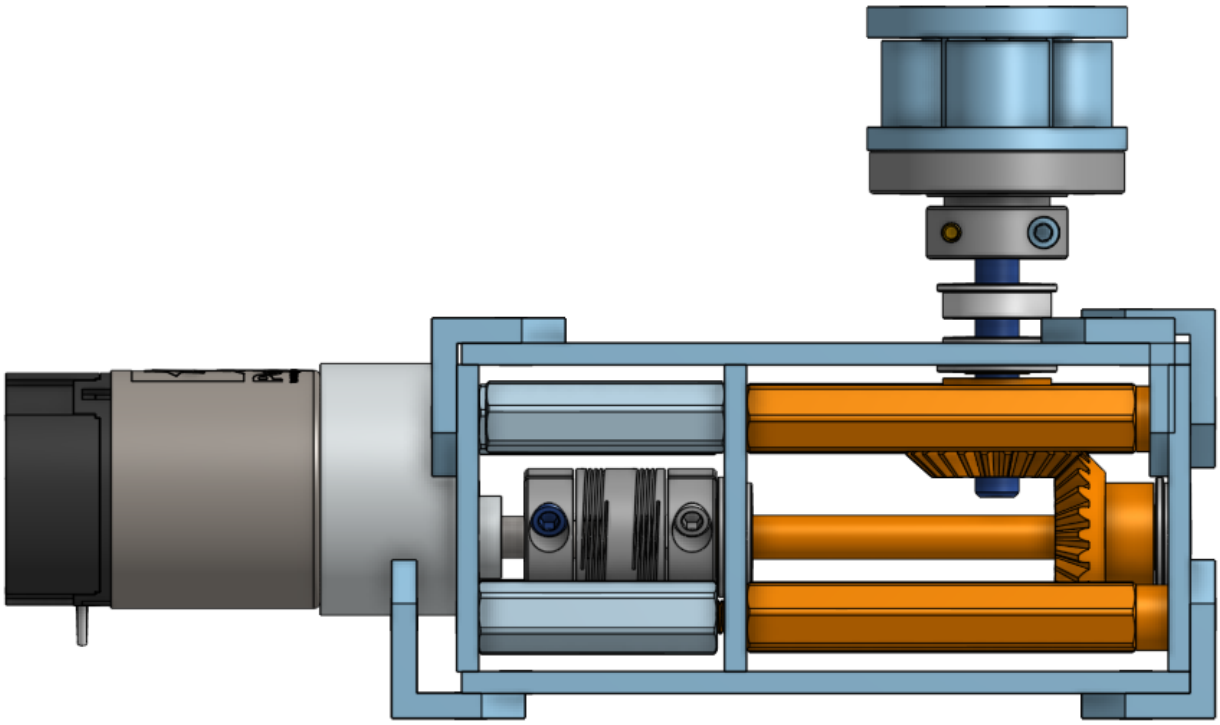
Miter Gear 90º	2	57.7	McMaster https://www.mcmaster.com/2600N1/
Belleville Disc Spring	12	$6.19	McMaster https://www.mcmaster.com/9712K61/

APPENDIX B: CAD

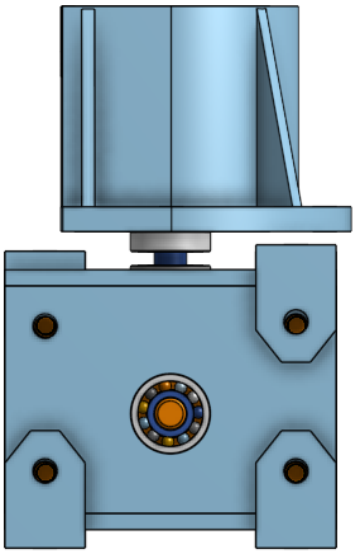
Enclosed Device



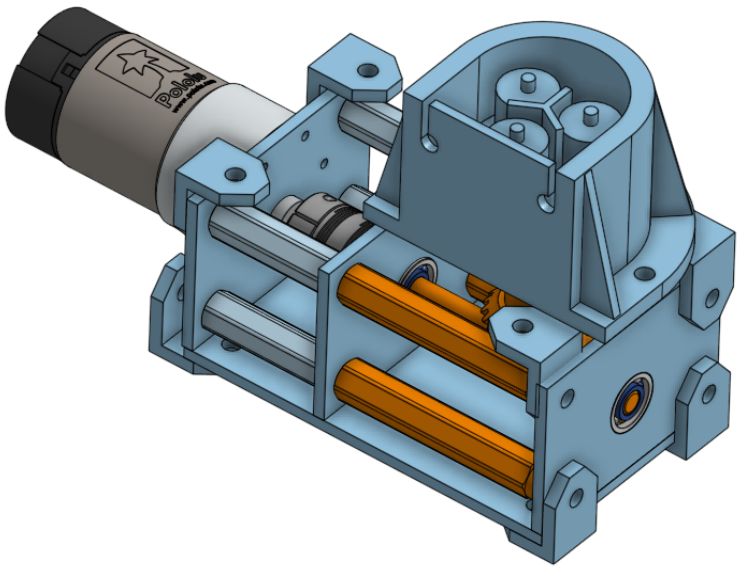
Side View



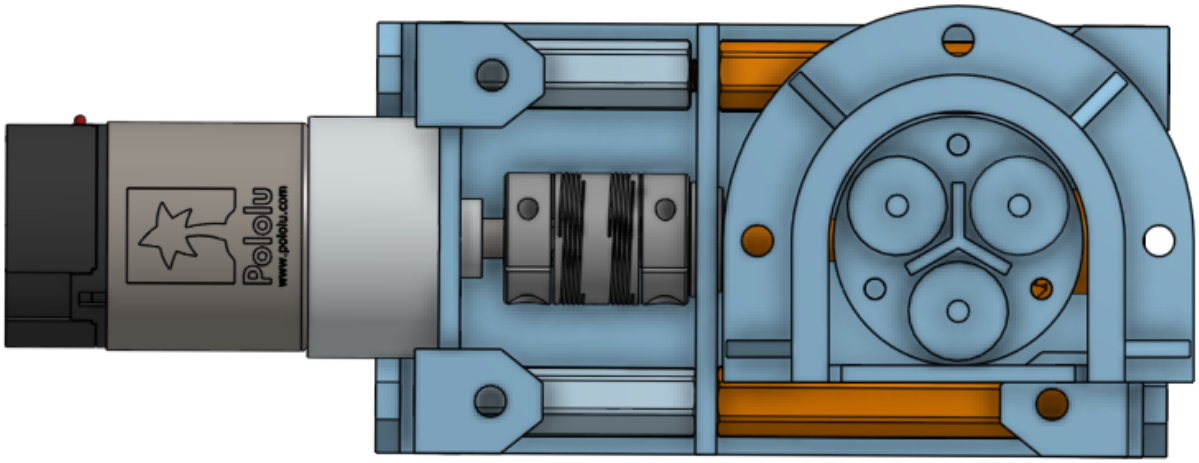
Front View



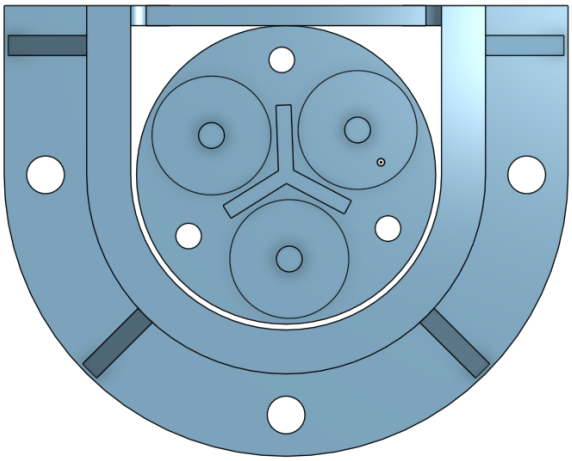
Isometric View



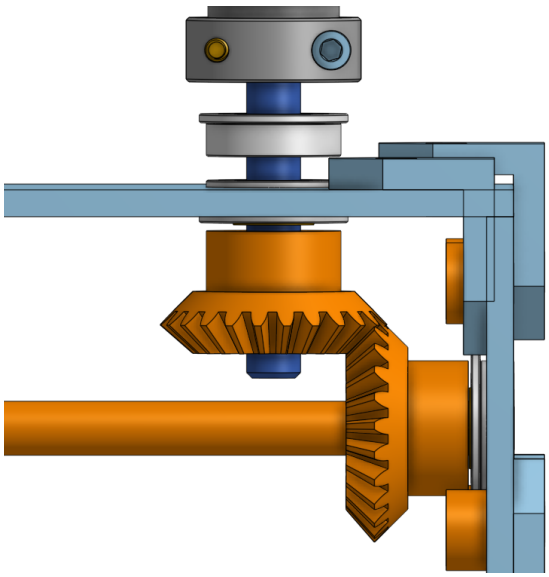
Top View



Squish Point and Rollers



Miter Gear Interaction



APPENDIX C: CODE

```
#include <ESP32Encoder.h>
#define DIR 26
#define AIN_2 25
#define PWM_A 15
#define POT 14
#define BTN 34

ESP32Encoder encoder;

int omegaSpeed = 0;
int omegaDes = 0;
int omegaMax = 100;
int D = 0;
int dir = 1;
int potReading = 0;

int Kp = 4;
float Ki = 0.4;
int IMax = 0;
int sumErr = 0;
int gErr = 0;
char antiwindup = 'Y';
int state = 1;

//Setup interrupt variables ----------------------------
volatile int count = 0; // encoder count
volatile bool deltaT = false;     // check timer interrupt 2
volatile bool buttonTriggered = false;
int totalInterrupts = 0;   // counts the number of triggering of the alarm
hw_timer_t * timer1 = NULL;
hw_timer_t * timer0 = NULL;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;

// setting PWM properties ------------------------------
const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int ledChannel_3 = 3;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 150;
const int MIN_PWM_VOLTAGE = 40;

//Initialization --------------------------------------
void IRAM_ATTR onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
  count = encoder.getCount( );
  encoder.clearCount ( );
  deltaT = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux1);
}
void IRAM_ATTR isDebounced(){
  timerStop(timer0);
}
void IRAM_ATTR isr(){
  buttonTriggered = true;
}
void setup() {
  pinMode(POT, INPUT);
  pinMode(BTN, INPUT);
  pinMode(DIR, OUTPUT);//new direction pin for new driver

  //interrupt triggered whenever signal changes(LOW to HIGH or HIGH to LOW)
  attachInterrupt(BTN, isr, CHANGE);

  Serial.begin(115200);
  ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up resistors
```

```
  encoder.attachFullQuad(33, 27); // Attache pins for use as encoder pins
  encoder.setCount(0);  // set starting count value after attaching

  // configure LED PWM functionalitites
  ledcSetup(ledChannel_3, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(PWM_A, ledChannel_3);

  // initilize timers
  timer1 = timerBegin(1, 80, true);  // timer 1, 12.5 ns * 80 -> 1000 ns = 1 us, countUp
  timerAttachInterrupt(timer1, &onTime1, true); // edge (not level) triggered
  timerAlarmWrite(timer1, 15000, true); // 15000 * 1 us = 15 ms measures counts per 15 ms

  timer0 = timerBegin(0, 80, true);
  timerAttachInterrupt(timer0, &isDebounced, true);
  timerAlarmWrite(timer0, 300000, true);

  timerAlarmEnable(timer1); // enable
  timerAlarmEnable(timer0);
  timerStop(timer0);
}

void loop() {
  switch (state){
    case 1: //pump off
      motor_off();
      if(CheckForButtonTrigger()){
        ButtonResponse();
        state = 2;
      }
      break;
    case 2: //pump on
      motor_on();
      if(CheckForButtonTrigger()){
        ButtonResponse();
        state = 1;
      }
      break;
  }

}

//SERVICE FUNCTIONS

bool CheckForButtonTrigger() {
  if (timerStarted(timer0)){//debounce button inp
    buttonTriggered = false;
    return false;
  }
  else{
    if(buttonTriggered){
      return true;
    }
    else{
      return false;
    }
  }
}
void ButtonResponse(){
  buttonTriggered = false;
  timerStart(timer0);
}
void motor_off(){
  ledcWrite(ledChannel_3, LOW);
  omegaDes = 0;
```

```
  sumErr = 0;
  omegaSpeed = 0;
  plotControlData();
}
void motor_on(){
  if (deltaT) {
    portENTER_CRITICAL(&timerMux1);
    deltaT = false;
    portEXIT_CRITICAL(&timerMux1);

    omegaSpeed = count;
    potReading = analogRead(POT);
    omegaDes = map(potReading, 0, 4095, 0, omegaMax);

    //P-I CONTROL
    int err = abs(omegaDes) - abs(omegaSpeed);
    gErr = err;
    sumErr = sumErr + err;

    D = Kp * err + Ki * sumErr;

    //Ensure that you don't go past the maximum possible command
    if (D > MAX_PWM_VOLTAGE) {
        D = MAX_PWM_VOLTAGE;
    }

    if (D > 0) {
        ledcWrite(ledChannel_3,   D);
    }
    else if (D < 0) {
        ledcWrite(ledChannel_3, -D);
    }
    else {
        ledcWrite(ledChannel_1, LOW);
        ledcWrite(ledChannel_2, LOW);
    }
    plotControlData();
  }
}
void plotControlData() {
  Serial.print("Error: ");
  Serial.print(gErr);
  Serial.print("  Sum Error: ");
  Serial.print(sumErr);
  Serial.print(" Speed: ");
  Serial.print(omegaSpeed);
  Serial.print(" Desired_Speed: ");
  Serial.print(omegaDes);
  Serial.print(" POT Reading: ");
  Serial.print(analogRead(POT));
  Serial.print(" PWM_Duty/10: ");
  Serial.print(D);  //PWM is scaled by 1/10 to get more intelligible graph
  Serial.print(" State: ");
  Serial.println(state);
}

```