MEC ENG 102B
Mechatronics Design
Fall 2022 – Project
Final Report

Project Title: Continuously Variable Differential Transmission Pulley System
Team 9: Toh Ming Xian, Brian Chuan Jie Sim, Jovon Jun Wei Lim, Megan Choo Ming Hui

1 Introduction and Overview

At construction sites, pulleys have difficulty lifting heavier loads with the generator typically producing a lot of smoke, showing that power drain increases when heavier loads are lifted. We saw this opportunity with the problem statement: "How can we create a power-saving pulley system that consumes constant power regardless of torque loads on the pulley?"

2 Component Specification and High Level Strategy

We created a **Continuously Variable Differential Gear Transmission Pulley System**. The system comprises a differential gear transmission, and is powered by 2 AndyMark NeveRest Classic 60 DC Gearmotors connected via 5V wall plug adaptors. It contains 2 digital motor encoders, an on/off toggle button, is controlled via the ESP32, and is programmed with a state machine using the Arduino IDE in C++. Appendix A shows the motor specifications. We wanted to prove that the concept was feasible and the pulley system performed completely to our expectations. The system was able to lift a load of 1.92 kg using a constant 0.5W of power, lower than the 1.7W of power needed in a pulley system with a single motor.

2.1 Operating Conditions

To keep total power P_{total} = 0.5W constant, current I and total voltage V_{total} were kept constant at 0.114A and 4.4V respectively for the entire duration of system operation. However, individual voltages supplied to M_A and M_B vary based on the required torque to lift the load. *Appendix A* contains the motor curve, from which we obtained power and torque values for the motor.

2.2 Mechanical: Function critical calculations

We define the primary motor, Motor A, as M_A. The gear ratio of the primary system is 1:1. We define the secondary motor, Motor B, as M_B. The gear ratio of the secondary system is 4:1. The relationship between the output shaft torque τ_{out} and the torques of input shaft A, τ_A, and input shaft B, τ_B is given by the general equation of the differential gear transmission system shown below:

$$\tau_{out} = 2\,[(\tau_A) + (\tau_B)] \quad \textbf{[1]}$$

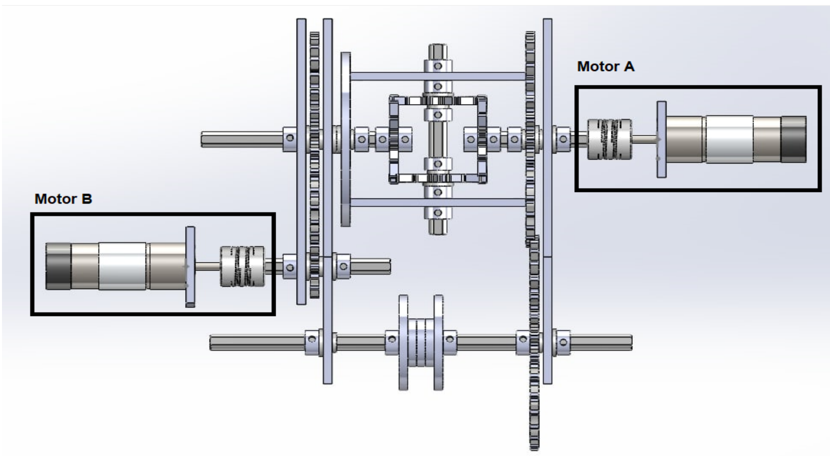


Figure 1: Top-view of system assembly

Case 1 (No Load):
Total power P_{total} = 0.5W is supplied only to M_A. From the motor curve, torque τ_{A1} produced was τ_{A1} = 0.002 x 60 = 0.12Nm *(see Appendix A*). M_B is off. Under no load, τ_{out} = 2 [(τ_{A1}) + (0)] = 2(0.12) = 0.24Nm.

Case 2 (Applied Load of m_{load} = 1.92 kg):
The radius of our pulley wheel is r_{pulley} = 1.5" = 0.0381m, hence the torque required to pull the load up is given by τ_{load} = (g)(r_{pulley})(m_{load}) = (9.81)(0.0381)(1.92) = 0.72Nm, which is more than what was originally provided by M_A in Case 1, where τ_{out} = 0.24Nm. Power P_{total} = 0.5W is now redirected to both M_A and M_B instead of only M_A.

As M_A is running at a 1:1 gear ratio, and M_B is running at a 4:1 gear ratio, power will be redirected to M_A and M_B in such a way that total torque produced τ_{out} will be equal to the torque required τ_{load}. In other words, τ_{out} = τ_{load}.

Hence, the total torque produced by M_A and M_B is:

$$\tau_{out} = 2\,[(\tau_{A2}) + (\tau_{B2})\,] = \tau_{load}$$

At I = 0.114A, the system supplies M_A with V_{A2} = 68 / 85 PWM x 4.4V = 3.5V as seen from the serial monitor. Hence, Power P_{A2} supplied to M_A is P_{A2} = IV_{A2} = 0.114 x 3.5 = 0.4W. Taking into account the 60:1 gear motor, and referring to the motor curve, torque τ_{A2} produced will be τ_{A2} = 0.002 x 60 = 0.12Nm *(see Appendix B)*.

The system supplies M_B with V_{B2} = 17 / 85 PWM x 4.4V = 0.9V as seen from the serial monitor. Hence, Power P_{B2} supplied to M_B is P_{B2} = IV_{B2} = 0.114 x 0.9 = 0.1W. Taking into account the 60:1 gear motor and differential gear ratio, and referring to the motor curve, torque τ_{B2} produced will be τ_{B2} = 0.001 x 60 x 4 = 0.24Nm *(see Appendix C)*.

V_{A2} + V_{B2} = V_{total} = 3.5 + 0.9 = 4.4V is kept constant as per operating conditions, keeping total power P_{total} = IV_{A2} + IV_{B2} = IV_{total} = 0.114 x 4.4 = 0.5W constant. Total torque produced is τ_{out} = 2 [(τ_{A2}) + (τ_{B2})] = 2[0.12 + 0.24] = 0.72 = τ_{load}. Since τ_{out} = τ_{load}, the load can now be lifted by the pulley system.

2.3 Proof of Concept by Comparison

A pulley system using a single NeveRest Classic 60 motor requires torque $\tau_A = \frac{0.72}{60}$ = 0.012Nm to lift the load of 1.92kg. From the motor curve, Power P required to produce torque τ_A = 0.012Nm = 1.7W, more than 3 times higher than P_{total} = 0.5W with our design, thus providing the proof of concept that we sought to achieve. *(see Appendix D)*

2.4 Electronics: Circuitry, state transition diagram and explanation of motor code

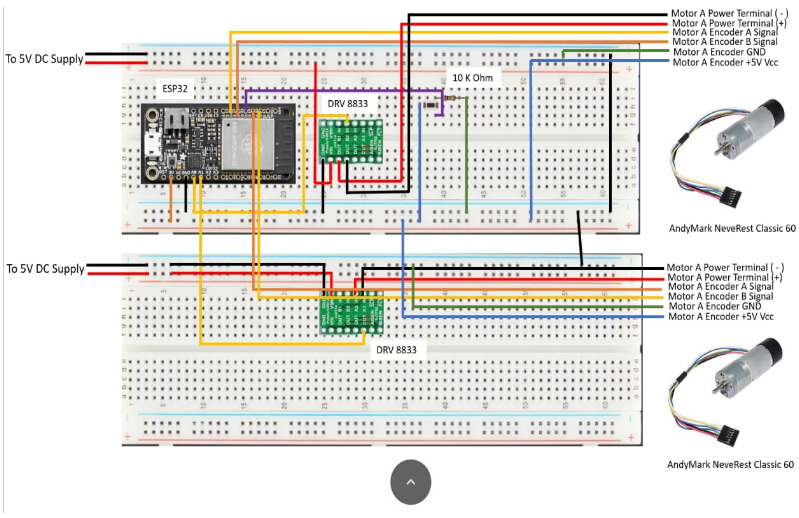


Figure 2: Circuit diagram

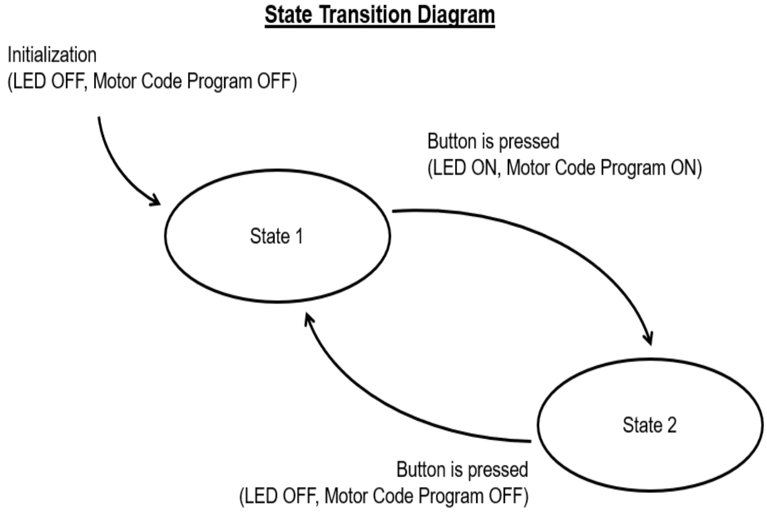


Figure 3: state transition diagram (right)

While our state machine is simple with only 2 states, we used a relatively complex Proportional Speed Control Loop as our motor code, which works in State 2. A constant total Pulse Width Modulation voltage (PWM voltage) of 850 is used here to maintain constant power P_{total}.

Under no load conditions in **Case 1**, power is solely supplied to M_A with current I = 0.114A and PWM voltage PWM_{total} = 850, which translates to 4.4V. Under an applied load in **Case 2**, M_A rotates slower due to the increased torque τ_{load} required to lift the steady load, as per $P_{total} = \tau\omega$.

When the encoder of M_A detects an error in the difference between the desired threshold angular speed $\omega_{threshold}$ and the actual speed ω_A in M_A, it will produce a control effort proportional to the error difference and supply the proportional gain to M_B. We define the PWM supplied to M_B as B, which follows the equation $B = K_p\ (\omega_{threshold} - \omega_A)$. Our chosen K_p value was 55. M_B now starts moving and supplies torque τ_{B2} to the differential. M_A receives the remainder of PWM_{total}, which we define as A, from M_B. The total PWM voltage is kept constant at A + B = 850. M_A now supplies a torque of τ_{A2} to the differential.

Depending on the τ_{load} required to lift the steady load, the code will continuously proportion varying amounts of PWM voltage to M_B and hence M_A to produce an output torque τ_{out} that is equal to torque required τ_{load} in the following manner: $\tau_{out} = 2\ [(\tau_{A2}) + (\tau_{B2})\ ] = \tau_{load}$.

3 Reflections

In order to keep costs affordable, we 3D printed heat-set collars for the parts of our transmission that were not load bearing. Our 3D-printed collars effectively kept our transmission shafts in place at a much lower cost than purchasing over 20 metal collars online.

Similarly to cutting costs, we opted to laser cut spur gears out of plywood for our differential instead of purchasing metal gears. We were limited by the inaccuracy that comes with laser cutting and had to design gears with rather large teeth gaps. This caused a slightly noticeable backlash in our differential transmission. With a bigger budget, we could CNC mill our differential gears out of aluminium or even purchase bevel gears online. The gears would have more precise teeth and hence better meshing, resulting in smoother operation and negligible backlash.

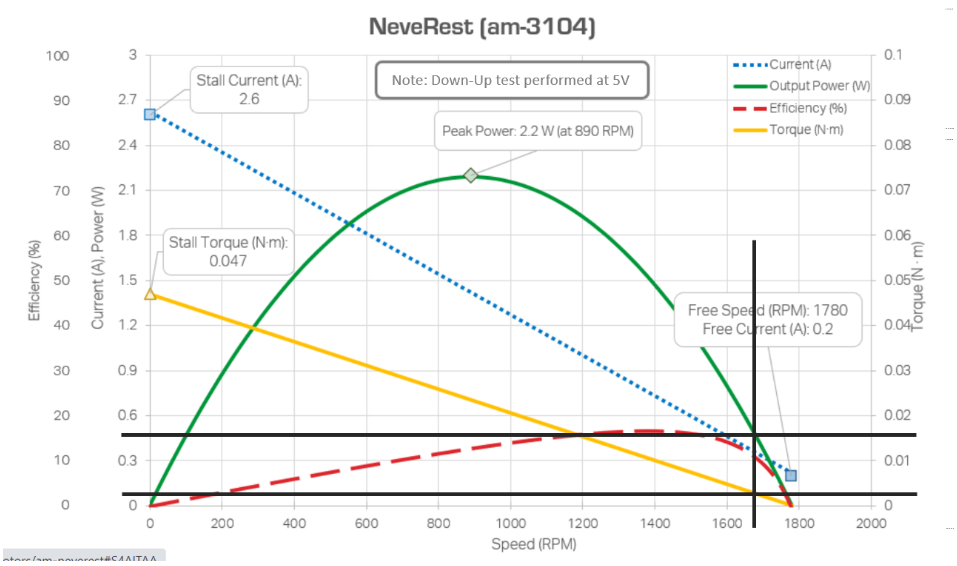
Appendix

Motor Specification Curve

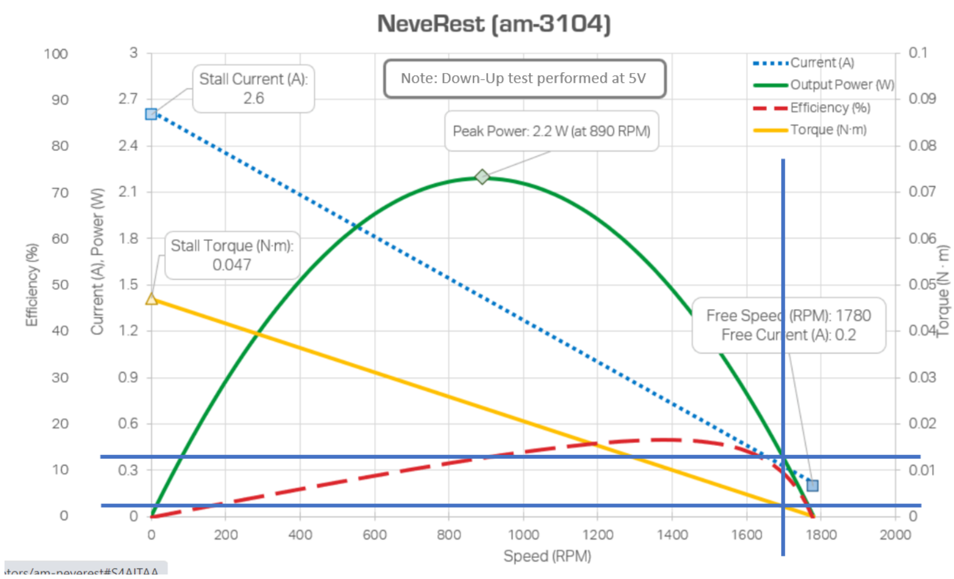
As the motor curve of the AndyMark NeveRest Classic 60 Gearmotor am-3103 was not available (we emailed the manufacturer to ask), we used the AndyMark NeveRest Classic Gearmotor am-3104 Gearmotor curve instead, which is an AndyMark NeveRest Classic 60 Gearmotor without the external 60:1 gear reducer. Hence, torque calculated in our report was always multiplied by 60, in order to reinclude the effect of the 60:1 gear reducer that the am-3103 has.

To calculate motor torque, start off with the power provided P to the motor specified, and draw a horizontal line equal to the power provided until the horizontal line intersects with the power curve. Then, draw a vertical line through the power curve and the torque curve. At the intersection of the vertical line and the torque curve, read off the torque value of the motor by drawing another horizontal line that cuts through the intersection of the vertical line and the torque curve.

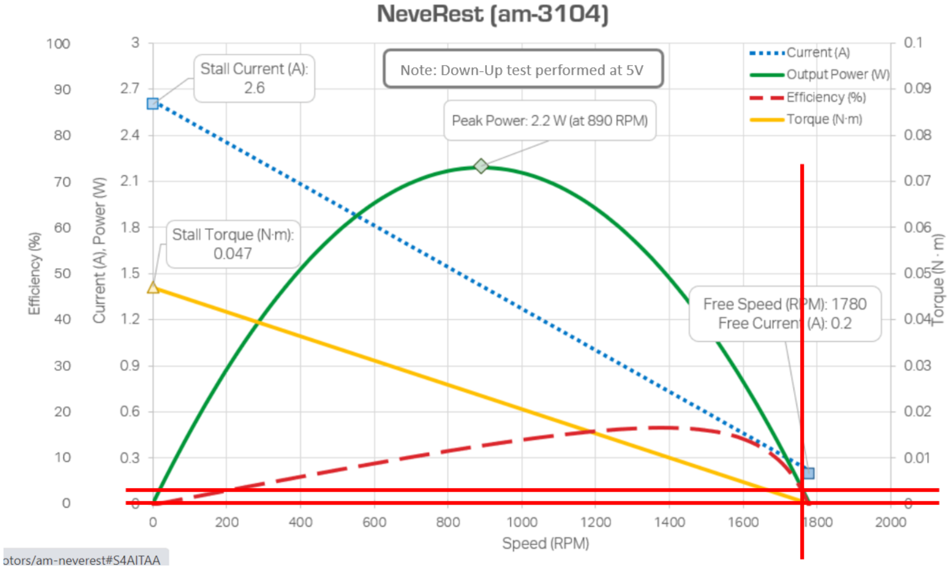
Appendix A: M_A producing 0.002 x 60 = 0.12Nm with 0.5W of power (Case 1)



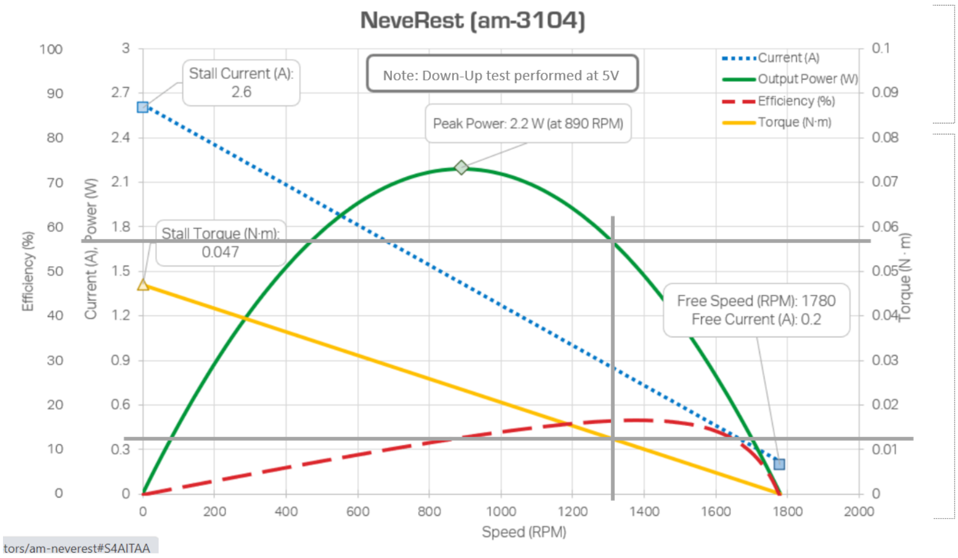
Appendix B: M_A producing 0.002 x 60 = 0.12Nm with 0.4W of power (Case 2)



Appendix C: M_B producing 0.001 x 60 x 4 = 0.24Nm with 0.1W of power (Case 2)



Appendix D: Single motor requiring 1.7W of power to obtain required torque of 0.012Nm x 60 = 0.72Nm



Appendix E: Bill of Materials

Item No.	Description	Part Number	Purchase Link	Quantity	Unit Cost	Total Cost	Remarks
1	Plywood 1/4" thickness	NA	https://store.jacobshall.org/products/plywood-1-4-x-12-x-24?variant=19550516543584	3	1.67	5.01	
2	NeveRest Classic 60 Gearmotor	am-3103b	https://www.andymark.com/products/neverest-classic-60-gearmotor?Power%20Connector=JST-VH-2%20(am-3103b)&quantity=1	1	16	16	
3	Encoder	am-2992	https://www.andymark.com/products/hall-effect-encoder-cable-with-4-pin-connector	2	1.5	3	

5	Shim	97022A380	https://www.mcmaster.com/97022A380/	3	5.46	16.38	Pack of 10
6	D Profile Rotary Shaft	8632T7	https://www.mcmaster.com/8632T7/	4	12.02	48.08	
7	Set Screw Shaft Collar (Hex Socket)	9414T11	https://www.mcmaster.com/collars/set-screw-shaft-collars-8/?SrchEntryWebPart_InpBox=ball+bearing	2	2.12	4.24	
8	Ultra-Low-Friction Oil-Embedded Sleeve Bearing	1677K8	https://www.mcmaster.com/catalog/128/1336	12	2	24	
12	Zinc Plated Steel Nut M3	90591A121	https://www.mcmaster.com/nuts/metric-low-strength-steel-hex-nuts/thread-size~m3/	1	1.9	1.9	Pack of 100
13	Screw M3 20mm	91274A109	https://www.mcmaster.com/91274A109/	1	7.65	7.65	Pack of 50
14	Screw M3 5mm	91292A110	https://www.mcmaster.com/91274A109/length~5-mm/socket-head-screws-4/	1	6	6	Pack of 100
15	M3 Split Lock Washer	92148A150	https://www.mcmaster.com/spring-washers/for-screw-size~m3/	1	1.92	1.92	Pack of 100, Standard not Curved
16	M3 General Purpose Washer	98689A112	https://www.mcmaster.com/washers/washers-3/for-screw-size~m3/metric-general-purpose-washers/	1	3.42	3.42	Pack of 100
17	Wood glue	7476A11	https://www.mcmaster.com/glue/wood-glue-6/	1	3.86	3.86	
4	1/4" x 1/2" Flexible Shaft Coupling	NA	https://www.amazon.com/12-7mm-Flexible-Shaft-Coupling-Coupler/dp/B07K3X3X1D	2	14.99	29.98	
11	12V Adapter	NA	https://www.amazon.com/Adapter-100-240V-Transformer-Charger-Security/dp/B091XSVV1Y/ref=psdc_10967101_t3_B00Q2E5IXW	2	6.29	12.58	
9	Heat Set Inserts	NA	https://www.amazon.com/cSeao-120pcs-Inserted-Knurled-Embedded/dp/B07D683Q26/ref=sr_1_10?crid=117GO4QBX1LL8&keywords=heat+set+insert+m3&qid=1667723186&sprefix=heat+set+insert+m%2Caps%2C141&sr=8-10	1	7.99	7.99	

10	Right Angle Bracket	NA	https://www.amazon.com/uxcell-Fastener-Brackets-Protector-Furniture/dp/B07RRS1P91/ref=sr_1_3?crid=2QQ9YATTAUPC8&keywords=m3+screw+right+angle+bracket&qid=1667768754&sprefix=m3+screw+right+angle+bracket%2Caps%2C145&sr=8-3	1	11.49	11.49	Pack of 80, M3, Sharp tip
18	Amazon Basics Hex Key Allen Wrench Set with Ball End - Set of 26	NA	https://www.amazon.com/AmazonBasics-Hex-Allen-Wrench-Ball/dp/B0776C2D6H/ref=sr_1_7?keywords=metric%2Ballen%2Bkey&qid=1667958094&s=hi&sprefix=metric%2Bellen%2Bke%2Ctools%2C158&sr=1-7&th=1	1	13.68	13.68	
19	Wooden Dowel	#52161	https://www.acehardware.com/departments/building-supplies/lumber-and-trim/dowels/52161	1	9.91	9.91	1" dia, 48" length
20	Screws	NA	https://www.amazon.com/dp/B00P8E1KCK?psc=1&ref=ppx_yo2ov_dt_b_product_details	1	4.73	4.73	
					Total Cost	**207.81**	

Appendix F: NeveRest Classic 60 Gear Motor Specifications

Back Drive Force	19.200 oz-in
Free Current	0.5 Amperes
Gear Material	Steel
Material	Steel with Plastic Encoder Housing
Maximum Diameter	1.500 in.
Maximum Power	14 Watts
No Load RPM	105RPM
Output Shaft Size	6 mm D Shaft
Ratio	60:1
Stall Current	11.5 Amperes
Stall Torque	525 oz-in
Voltage Requirement	12 Volts DC

Appendix G: CAD

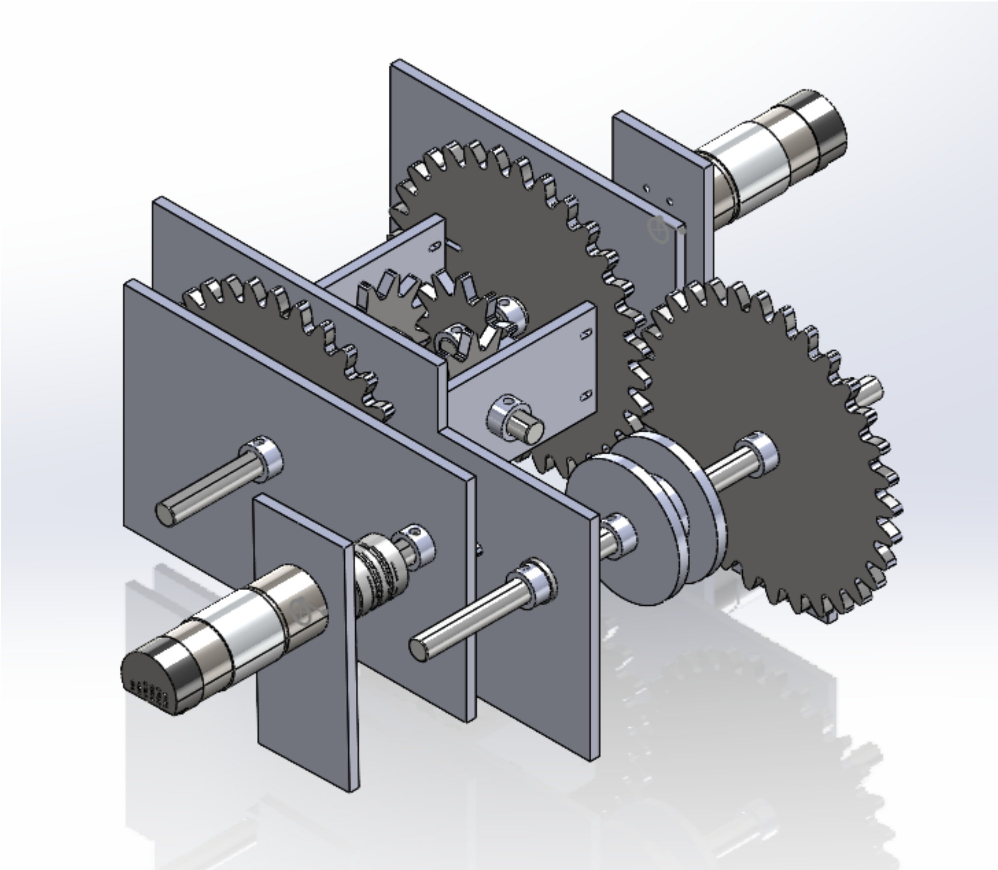


Figure 1: Isometric View

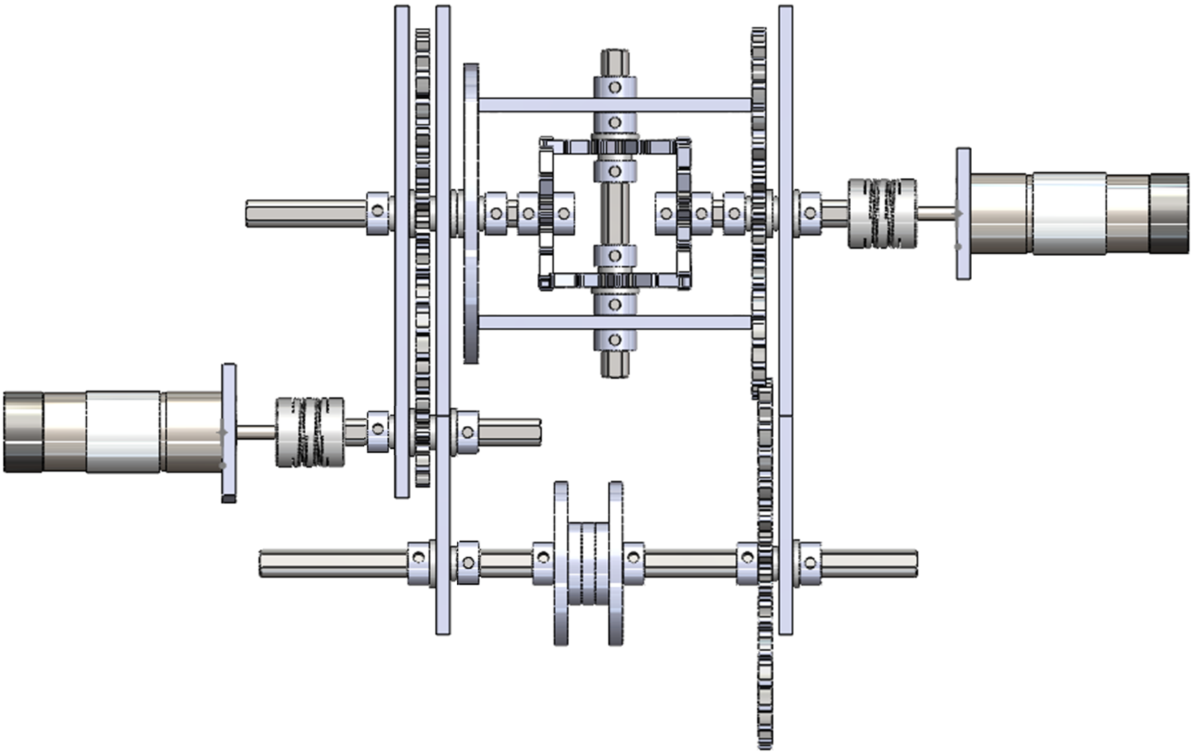


Figure 2: Top View

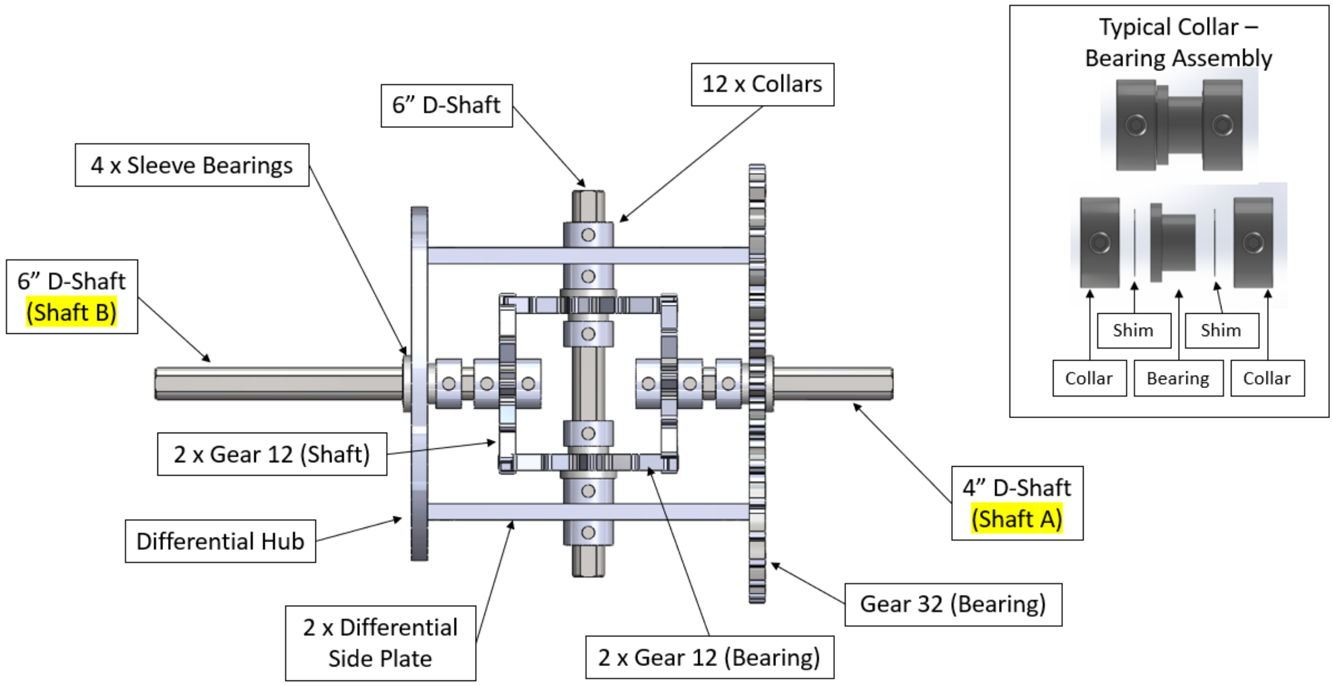


Figure 3: Differential Subassembly

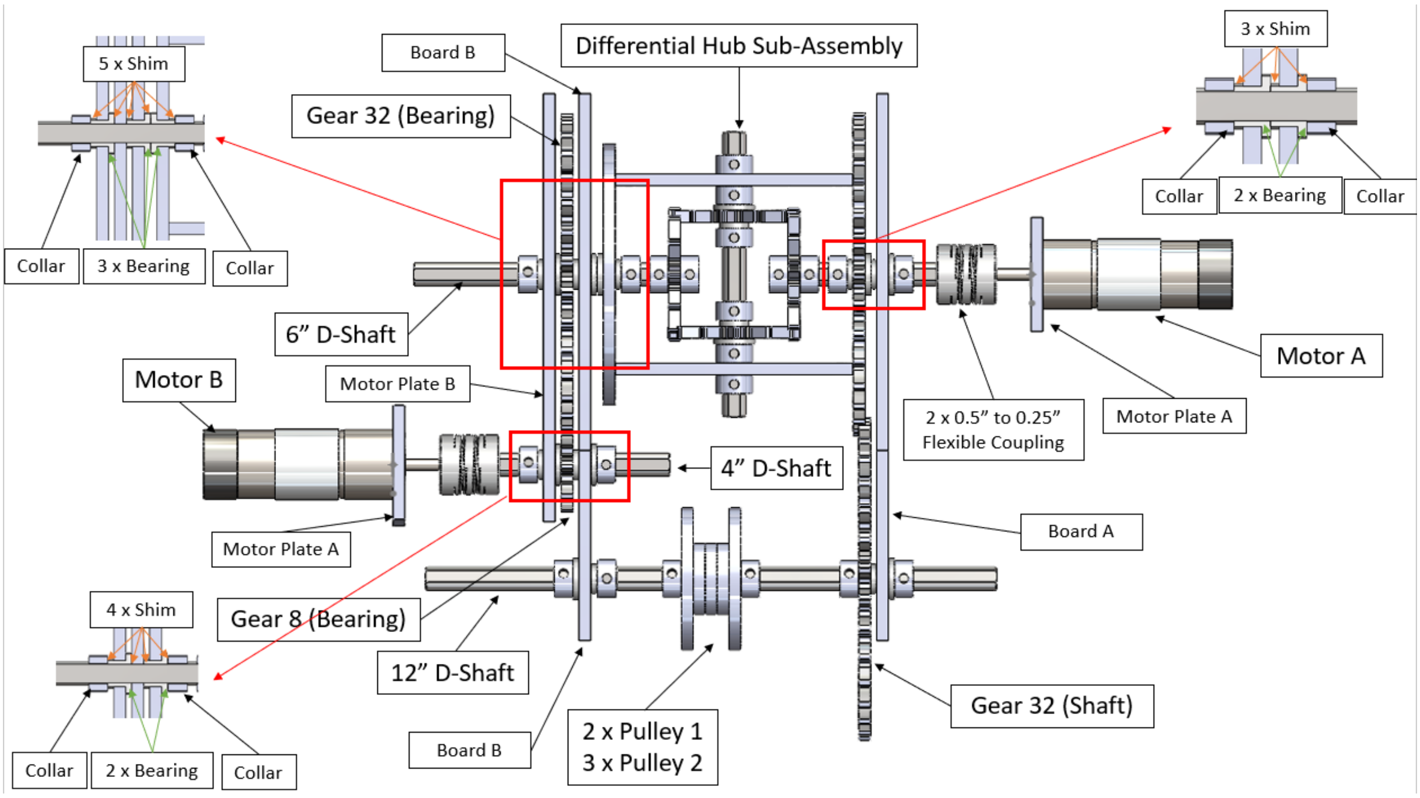


Figure 4: Top View Full Assembly

Appendix H: Code

```
#include <ESP32Encoder.h>
#define BIN_2 26 //Motor A
#define BIN_1 25 //Motor B
#define LED_PIN 13 // declare the builtin LED pin number
#define BTN 15  // declare the button ED pin number

ESP32Encoder encoderA; //encoder A
ESP32Encoder encoderB; // encoder B

int omegaSpeedA = 0;
int omegaSpeedB = 0;
int omegaThreshold = 6;
int A = 0;
int B = 0;
int motorstate = 0;

volatile bool buttonIsPressed = false;
int state = 1;

int Kp = 55;   // TUNE THESE VALUES TO CHANGE CONTROLLER PERFORMANCE
int omegaTransition = 0;


//Setup interrupt variables ----------------------------
volatile int countA = 0; // encoder count A
volatile int countB = 0; //encoder count B
volatile bool interruptCounter = false;    // check timer interrupt 1
volatile bool deltaT = false;     // check timer interrupt 2
int totalInterrupts = 0;   // counts the number of triggering of the alarm
hw_timer_t * timer0 = NULL;
hw_timer_t * timer1 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;


// setting PWM properties ----------------------------
const int freq = 5000;
const int pwmChannel = 0;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 800;

//Initialization ------------------------------------
void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  interruptCounter = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux0);
}
```

```
void IRAM_ATTR isr() {  // the function to be called when interrupt is triggered
  buttonIsPressed = true; //sets condition of buttonIsPressed to true
  interruptCounter = false; //resets condition of interruptCounter back to false
  timerStart(timer0); //starts timer

}

void IRAM_ATTR onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
  countA = encoderA.getCount( ); //encoder A
  countB = encoderB.getCount( ); //encoder B
  encoderA.clearCount ( );
  encoderB.clearCount ( );
  deltaT = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux1);
}

 void setup() {
   // put your setup code here, to run once:
   pinMode(LED_PIN, OUTPUT);
   digitalWrite(LED_PIN, LOW); // sets the initial state of LED as turned-off
   pinMode(BTN, INPUT);  // configures the specified pin to behave either as an input or an output
   attachInterrupt(BTN, isr, RISING);

   ledcSetup(pwmChannel, freq, resolution);


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

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

   encoderB.attachHalfQuad(32, 14); // Attache pins for use as encoder pins  //encoder B
   encoderB.setCount(0);  // set starting count value after attaching

// configure LED PWM functionalitites
ledcSetup(ledChannel_1, freq, resolution);
ledcSetup(ledChannel_2, freq, resolution);

// attach the channel to the GPIO to be controlled
ledcAttachPin(BIN_1, ledChannel_1); //Motor B
ledcAttachPin(BIN_2, ledChannel_2); //Motor A

// initilize timer
timer0 = timerBegin(0, 80, true);  // timer 0, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 ->
timerAttachInterrupt(timer0, &onTime0, true); // edge (not level) triggered
timerAlarmWrite(timer0, 100000, true); // 100000 * 1 us = 100 ms, autoreload true, adds debounce with 100ms delay

timer1 = timerBegin(1, 80, true);  // timer 1, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 ->
timerAttachInterrupt(timer1, &onTime1, true); // edge (not level) triggered
timerAlarmWrite(timer1, 10000, true); // 10000 * 1 us = 10 ms, autoreload true
```

```
  // at least enable the timer alarms
  timerAlarmEnable(timer0); // enable
  timerAlarmEnable(timer1); // enable


}



void loop() {


switch (state) {


  case 1:
    motor();
    if (CheckForButtonPress() == true) { //Service function: turns on led,changes motorstate to 1,changes to state 2
      Serial.println("Transmission is now on");
      led_on();
      motorstate = 1;
      state = 2;
      Serial.println(motorstate);

    }
    break;

  case 2:

    motor();
    if (CheckForButtonPress() == true) { //Service function: turns off led,changes motorstate to 0,changes to state 1
      Serial.println("Transmission is now off");
      led_off();
      motorstate = 0;
      state = 1;
      Serial.println(motorstate);

    }
    break;

  |
}
}

//Motor Function
void motor() {

if(motorstate == 1){

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


    omegaSpeedA = countA;
    omegaSpeedB = countB;|

    B = Kp*(omegaThreshold - omegaSpeedA); //Proportional Control Loop Code, change omegaThreshold, Kp to adjust sensitivity

    omegaTransition = (omegaThreshold - omegaSpeedA);
```

```
  if (B > MAX_PWM_VOLTAGE) {
      B = MAX_PWM_VOLTAGE;
  }
  else if (B < -120) {
      B = -120;
  }


  //Map the D value to motor directionality
  //FLIP ENCODER PINS SO SPEED AND D HAVE SAME SIGN
  if (B > 0) {
      ledcWrite(ledChannel_1, B);

  }

  else {
      ledcWrite(ledChannel_1, LOW);
      ledcWrite(ledChannel_2, LOW);
```

```
 A = 850 - B; //Governing Equation to direct remaining voltage from motor B to motor A, where PWM = 850 is total PWM voltage of system



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


  //Map the D value to motor directionality
  //FLIP ENCODER PINS SO SPEED AND D HAVE SAME SIGN
  if (A > 0) {

      ledcWrite(ledChannel_2, A);
  }
  else if (A < 0) {
      ledcWrite(ledChannel_1, -A);

  }
  else {
      ledcWrite(ledChannel_1, LOW);
      ledcWrite(ledChannel_2, LOW);
  }


  plotControlData();

else{

  ledcWrite(ledChannel_1, 0);
  ledcWrite(ledChannel_2, 0);

}


}

bool CheckForButtonPress() { //Event Checker, checks for buttonIsPressed == true and interruptCounter == true,starts timer
```

```
  if((buttonIsPressed == true) & (interruptCounter == true)){
    void timerRestart(hw_timer_t *timer);
    buttonIsPressed = false;
    return true;
  }
  else {
    return false;
  }
}

void led_on() {
  digitalWrite(LED_PIN, HIGH);
}

void led_off() {
  digitalWrite(LED_PIN, LOW);
}

void plotControlData() {
  Serial.print(omegaThreshold);
  Serial.print(" ");
  Serial.print(omegaTransition);
  Serial.print(" ");
  Serial.print(omegaSpeedA);
  Serial.print(" ");
  Serial.print(omegaSpeedB);
  Serial.print(" ");
  Serial.print(A/10);  //PWM is scaled by 1/10 to get more intelligible graph
  Serial.print(" ");
  Serial.print(B/10);
  Serial.print(" ");
  Serial.println(abs(A/10) + abs(B/10));

}
```

References

[1] *2.972 How A Differential Works*. (n.d.). https://web.mit.edu/2.972/www/reports/differential/differential.html