ME 102B: Morphing Airfoil Manual

Group 11: Richard Tee, James Yoo

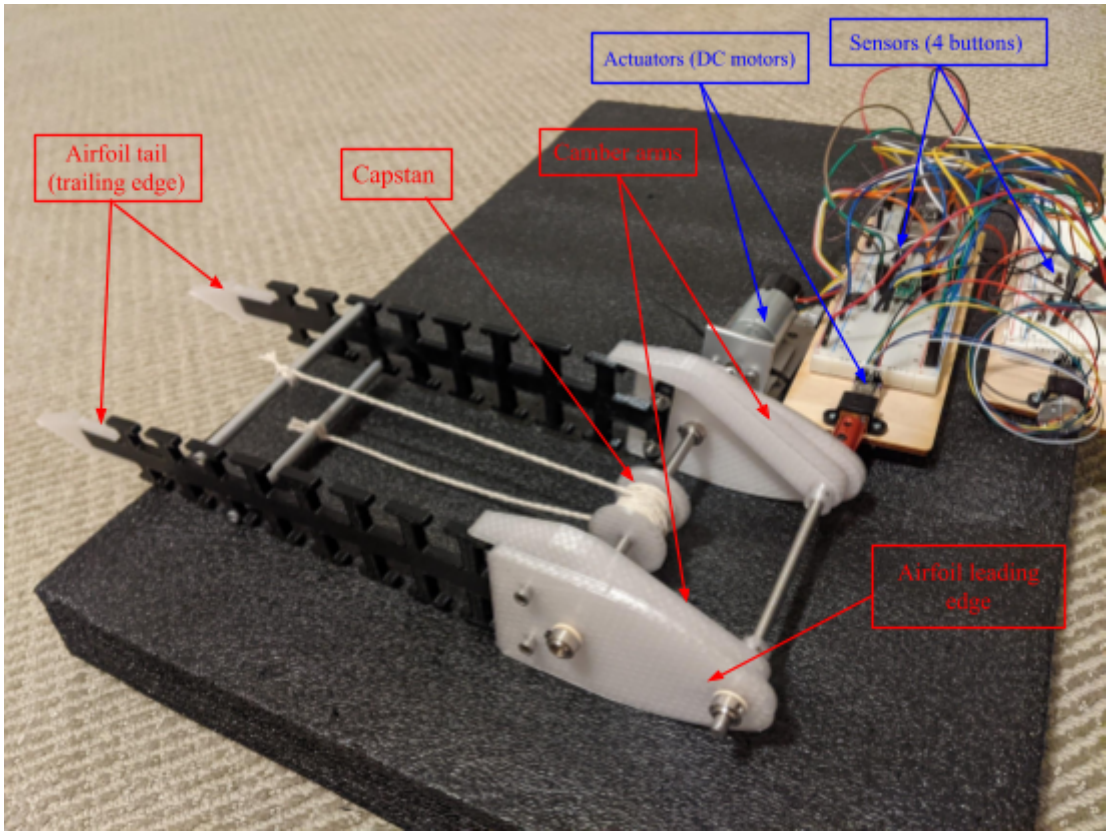
I. Opportunity

We want to improve the efficiency and control that pilots have during airplane flight by allowing for fine tuned control of airfoil parameters, such as camber thickness, camber-line, and deflection of the trailing edge.

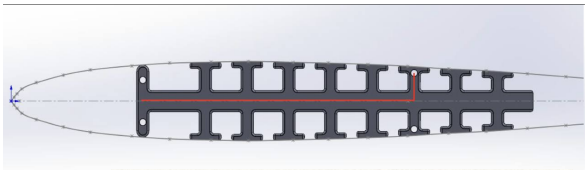
II. High Level Strategy

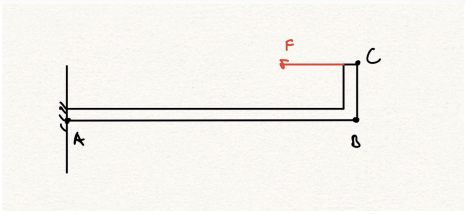
We wanted the ability to change the geometric shape of an airfoil. We were able to implement controls that would alter the camber of the airfoil and bend the tail of the airfoil upwards or downwards to desired positions. The airfoil would be able to switch between being symmetric or asymmetric and create more lift or drag depending on these parameters. We seeked to implement camber arms in the airfoil lifting up, making the upper camber of the airfoil thicker than the bottom and a capstan with rope to deflect the airfoil tail upwards and downwards by approximately 20° from the horizontal symmetric axis. We achieved the camber arms lifting up and down to any angle we desired (due to how light the arms weighed), which alters the thickness of the upper camber profile. In addition, the airfoil tail is able to deflect ~20° downwards, but the upwards component proved to be more difficult and we weren't able to achieve the same amount of deflection in the upward direction. We found that the motor isn't quite strong enough to lift the tail upwards, therefore we only have good deflection in the downwards direction. Despite this, it is acceptable in our design as downwards deflection proves to be most effective for generating desired lift. In the future, we want to implement a flexible outer shell for the airfoil that acts as the outer structure/shell of the wing. It would further show off the wing characteristics changing.

III. Full Assembly of System



IV. Function critical decisions and calculations





Tail deflection calculation

Material: 95A TPU

Elastic Modulus: 26.0 MPa = 3770.98118 psi

Arm 1 attachment Length: 6.5 in

Arm 2 attachment Length: 0.6657 in

Total tail length: 9.5 in

B = 0.2 in

H = 0.5 in

Moment of inertia: $\frac{BH^3}{12} = \frac{(0.2)(0.5^3)}{12} = .002083\ in^4$

Assumptions:

1. Fishtail acts as "L-shaped" fixed cantilever beam with force applied at point C.
2. We treat the deflection of BC as a moment Mz with magnitude FL_{CB}
3. We treat the deflection of AB as a sum of the reaction force Fat B and a reaction moment $M_z = FL_{CB}$ at point B (in the absence of BC)

(1) $M_z = FL_2$

(2) $\frac{\partial M_z}{\partial F} = L_2$

Calculation is based on an assumption of a pulling force on an L shaped cantilever beam as shown below. Castigliano's theorem was applied to approximate the deflection of the beam. We begin by stating the total deflection of the beam is the superposition of the partial derivatives of strain energies of beam AB and BC with respect to the force applied P (in this case, F):

$$\delta_{tot} = \frac{\partial U}{\partial P} = \frac{\partial U_{AB}}{\partial P} + \frac{\partial U_{BC}}{\partial P}$$

Splitting the strain energies of each beam into their respective components, we obtain:

$$\delta_{tot} = \frac{\partial}{\partial F}\left[\int_0^{L_1} \frac{F^2}{2EA}dx + \int_0^{L_1} \frac{M_z}{2EI}dx\right] + \left[\frac{\partial}{\partial F}\int_0^{L_2} \frac{M_z}{2EI}dy\right]$$

Simplifying,

$$\delta_{tot} = \frac{1}{2EA}\int_0^{L_1} 2F(\frac{\partial F}{\partial F})dx + \frac{1}{2EI}\int_0^{L_1} 2M_z(\frac{\partial M_z}{\partial F})dx + \frac{1}{2EI}\int_0^{L_2} 2M_z(\frac{\partial M_z}{\partial F})dy$$

Substitute (2):

$$\delta_{tot} = \frac{1}{2EA}\int_0^{L_1} 2Fdx + \frac{1}{2EI}\int_0^{L_1} 2M_zL_2dx + \frac{1}{2EI}\int_0^{L_2} 2M_zL_2dy$$

Substitute (1):

$$\delta_{tot} = \frac{1}{2EA}(2FL_1) + \frac{1}{2EI}[2(FL_2)(L_2)(L_1)] + \frac{1}{2EI}[2(FL_2)(L_2)(L_2)]$$

Finally,

$$\delta_{tot} = \frac{FL_1}{EA} + \frac{FL_2^2 L_1}{EI} + \frac{FL_2^3}{EI}$$

For total deflection of .5 in from point B, we calculate the force on the beam to be (through MATLAB):
P = 1.1864 lb

From this value, we can calculate the torque that the pulley must provide as

$$T = PL_2 = 1.1864\,lb * .6657\,in = 0.7898\,lb - in$$

Converting to kg-cm,

$$0.7898\,[lb - in] * \frac{1}{.868}\left[\frac{kg-cm}{lb-in}\right] = 0.910\,kg - cm$$

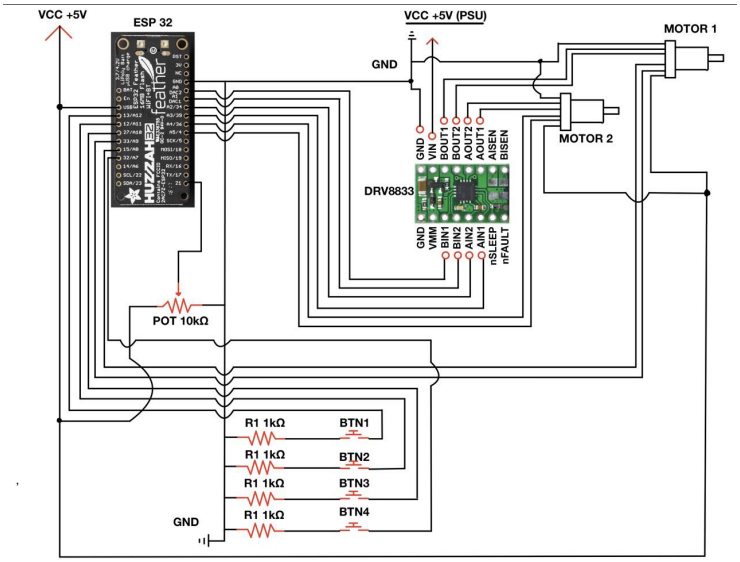
Thus, we found our motor to be the 12V Gearmotor with nominal torque of 1.2 kg-cm.
MATLAB Code:

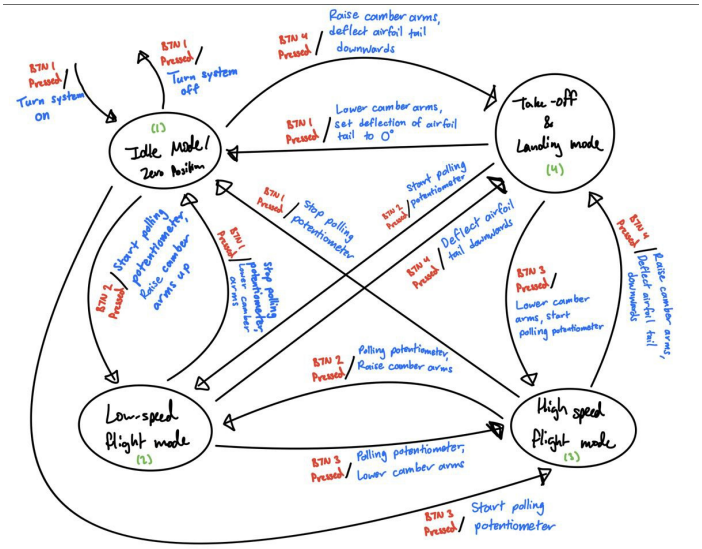
```
delta = 0.5;
b = 0.2;
h = 0.5;
I = (1/12)*b*h^3;
E = 3770.98118;
A = b*h;
L1 = 6.5;
L2 = 0.6657;


syms F
eqn = delta == (F*L1)/(E*A)+(F*L2^2*L1)/(E*I)+(F*L2^3)/(E*I);
P = double(solve(eqn,F)) %force in lb
T = P*L2 %torque in lb-in
```

```
P = 1.1864
T = 0.7898
```

V. Circuit Diagram/State Diagram





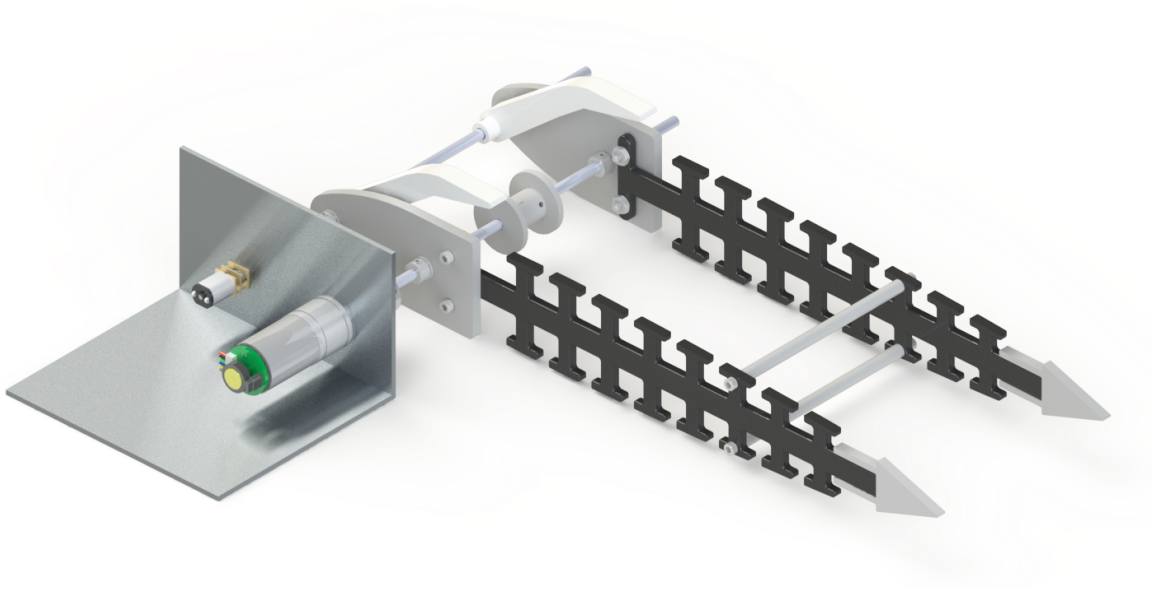
VI. Reflection

For future students, it would be extremely prudent to have a clear idea of what you want to build when you start - and when you start, do it early. What worked for our group was to keep up good communication, take tasks in chunks, and have a clear understanding of responsibilities. One thing we wished we did was to finalize a design early and start building super early so we had a lot of time to fix any errors in design. Also, double check hardware when purchasing online and make sure to get it right the first time - check tolerances, put them in CAD, and ask for help to see if there are better options.

Appendix A: Bill of Materials of Final Product

ESP32 Feather	Microcontroller	Microcontroller to run the entire system		N/a	1	
Pushbutton	Buttons	Buttons for control system and signal inputs		N/a	4	
Potentiometer	Potentiometer dial	Potentiometer for analog control of airfoil tail		N/a	1	
Micro Metal Gearmotor	75:1 Micro Metal Gearmotor HP 6V	Motor for camber arm system	2215	$ 16.95	1	Pololu
Magnetic Encoder Pair Kit	Magnetic Encoder Pair Kit with Side	Encoder for micromotor	4761	$ 9.95	1	Pololu
Dual Motor Driver	DRV8833 Dual Motor Driver Carrier	Motor driver to drive two motors wit	2130	$ 6.95	1	Pololu
12V Gear Motor	Metal Gearmotor 12V DC High Spe	Motor for pulley system	B07GNDG2NC	$ 15.99	1	Amazon
3mm String Spool	3mm Black Satin Cord Rattail Silk C	String for pulley system for morphing	B07TZW1DBW	$ 10.99	1	Amazon
5mm Rotary Shaft	5mm x 150mm 304 Stainless Steel Sc	Shaft for rotary camber arm	B082ZNS27M	$ 11.99	1	Amazon
4mm to 5mm shaft coupling	4mm to 5mm Bore Rigid Coupling S	Coupling for motor 1 and rotary sh	B07P95G6ZK	$ 7.49	1	Amazon
3mm to 5mm shaft coupling	3mm to 5mm Bore Rigid Coupling S	Coupling for motor 2 and rotary sh	B07PBBDS3Z	$ 5.99	1	Amazon
5mm shaft collar	10PCS RC Airplane Plane Landing C	Shaft collar for rotary shaft	B07TXCM64H	$ 9.49	1	Amazon
5mm Belleville Disc Spring	Belleville Disc Spring for 5 mm Shaft Diameter, 5.2 mm ID	Disc springs to preload bearings an	96445K204	$ 3.31	1	McMaster Carr
18-8 Stainless Steel Hex Nut	18-8 Stainless Steel Hex Nut 6-32 Thread Size	Hex nuts for fastener screws for asse	91841A007	$ 3.18	1	McMaster Carr
Stainless Steel Socket Head Screw	Stainless Steel Socket Head Screw 6-32 Thread Size, 5/8" Long	Fasteners to attach parts together	92185A149	$ 2.65	1	McMaster Carr
Female Threaded Round Standoff	Female Threaded Round Standoff,	Needed light standoffs for the caps	93330A562	$ 2.48	3	McMaster Carr
Dry-Running Sleeve Bearing	Light Duty Dry-Running Flanged Sle	Need bushings to prevent any fricti	2705T117	$ 1.20	6	McMaster Carr
TPU Filament	OVERTURE TPU Filament 1.75mm Fle	TPU filament to 3D print the flexible	B07VDP2S3P	$ 21.59	1	Amazon
PLA Filament	Atomic CLEAR / NATURAL PLA FILA	PLA filament to 3D print the airfoil structures (leading edg		$ 29.99	1	Atomic

Appendix B: CAD of Final Product



Appendix C: Arduino Code used for Final Product

ProjectCodeME102b

```
#include <ESP32Encoder.h>
#define BIN_1 25
#define BIN_2 26
#define AIN_1 21
#define AIN_2 4
#define LED_PIN 13
#define BTN_PWR 12         //power motor and zero
#define BTN_LowSpeed 36    //low speed button
#define BTN_HighSpeed 15   //high speed button
#define BTN_TOAL 32          // take off and landing button
#define POT 14

//initialize two encoders
ESP32Encoder encoder1;
ESP32Encoder encoder2;


//variables for motor 1 control
int omegaSpeed = 0;
int omegaDes = 0;
int posDes = 0;
int pos = 0; //position of motor 1
int posErr = 0;
int D = 0;
int err = 0;
int sumErr = 0;

//variables for motor 2 control

int omegaSpeed2 = 0;
int omegaDes2 = 0;
int posDes2 = 0;
int pos2 = 0; //position of motor 2
int posErr2 = 0;
int D2 = 0;
int err2 = 0;
int sumErr2 = 0;
```

```
int KpTail = 15;
int KiTail = 1;
int KpCam = 10;
int KiCam = 1;


//Setup interrupt variables ----------------------------
volatile int count = 0; // encoder1 count
volatile int count2 = 0; //encoder2 count

volatile bool interruptCounter = false;    // check timer interrupt 1
volatile bool deltaT = false;     // check timer interrupt 2
volatile bool deltaT2 = false; //check timer interrupt 3

int totalInterrupts = 0;   // counts the number of triggering of the alarm
hw_timer_t * timer0 = NULL;
hw_timer_t * timer1 = NULL;
hw_timer_t * timer2 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux2 = portMUX_INITIALIZER_UNLOCKED;


volatile bool button1IsPressed = false;
volatile bool button2IsPressed = false;
volatile bool button3IsPressed = false;
volatile bool button4IsPressed = false;


// setting PWM properties ---------------------------
const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int ledChannel_3 = 3; //look into this
const int ledChannel_4 = 4;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 150;
```

```
//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 onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
  count = encoder1.getCount( );
  pos += count; //position control
  encoder1.clearCount ( );
  deltaT = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux1);
}
void IRAM_ATTR onTime2() {
  portENTER_CRITICAL_ISR(&timerMux2);
  count2 = encoder2.getCount( );
  pos2 += count2; //position control
  encoder2.clearCount ( );
  deltaT2 = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux2);
}


void IRAM_ATTR isr1() {  // the function to be called when interrupt is triggered
  button1IsPressed = true;
}
void IRAM_ATTR isr2() {
  button2IsPressed = true;
}
void IRAM_ATTR isr3() {
  button3IsPressed = true;
}
void IRAM_ATTR isr4() {
  button4IsPressed = true;
}
```

```
void setup() {
  // put your setup code here, to run once:
  pinMode(BTN_PWR, INPUT); // configures the specified pin to behave either as an input or an output
  pinMode(BTN_LowSpeed, INPUT);
  pinMode(BTN_HighSpeed, INPUT);
  pinMode(BTN_TOAL, INPUT);
  pinMode(POT, INPUT);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW); // sets the initial state of LED as turned-off
  attachInterrupt(BTN_PWR, isr1, RISING);
  attachInterrupt(BTN_LowSpeed, isr2, RISING);
  attachInterrupt(BTN_HighSpeed, isr3, RISING);
  attachInterrupt(BTN_TOAL, isr4, RISING);

  Serial.begin(115200);
  ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up resistors
  encoder1.attachHalfQuad(33, 27); // Attache pins for use as encoder pins
  encoder1.setCount(0);  // set starting count value after attaching
  encoder2.attachHalfQuad(34, 39); //attach encoder pins for second motor
  encoder2.setCount(0);

  // configure LED PWM functionalitites
  ledcSetup(ledChannel_1, freq, resolution);
  ledcSetup(ledChannel_2, freq, resolution);
  ledcSetup(ledChannel_3, freq, resolution);
  ledcSetup(ledChannel_4, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(BIN_1, ledChannel_1);
  ledcAttachPin(BIN_2, ledChannel_2);
  ledcAttachPin(AIN_1, ledChannel_3);
  ledcAttachPin(AIN_2, ledChannel_4);

  // initilize timer
  timer0 = timerBegin(0, 80, true);  // timer 0, 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, 2000000, true); // 2000000 * 1 us = 2 s, autoreload true

  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

  timer2 = timerBegin(2, 80, true);  // timer 2, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 -> 1000 ns = 1 us, countUp
  timerAttachInterrupt(timer2, &onTime2, true); // edge (not level) triggered
  timerAlarmWrite(timer2, 10000, true); // 10000 * 1 us = 10 ms, autoreload true

  // at least enable the timer alarms
  timerAlarmEnable(timer0); // enable
  timerAlarmEnable(timer1); // enable
  timerAlarmEnable(timer2); // enable
}


void loop() {
  // put your main code here, to run repeatedly:
  switch (state) {

    case 0 : // motor is stopped, default off state.
      if (buttonPressPower()) {
        //turns on motor, sets to idle position
        state = 1;
        startMotorResponse();

      }
      break;
```

```
case 1: //idle position



  //zero position
  zeroPos();

  //check for buttons that want to switch to HS,LS,TOAL, and power.

  if (buttonPressPower()) {
    state = 0;
    stopMotorResponse();

  }
  if (buttonPressTOAL()) {
    state = 2;
  }
  if (buttonPressLS()) {
    state = 3;
  }
  if (buttonPressHS()) {
    state = 4;
  }



  //}
  //=====================================================================
  break;


 case 2: //take off and land case: disable tail control and set to specified position. set camber angle up.


   //set tail control to hard limit of 25.
   TOALpos();

   //set camber angle to 25.
   raiseCamberArm();



   //check for buttons that want to switch to high speed or low speed flight or turn off.
   if (buttonPressPower()) {
     state = 0;

     stopMotorResponse();
   }
   if (buttonPressLS()) {
     state = 3;
   }
   if (buttonPressHS()) {
     state = 4;
   }


   break;
```

```
    case 3: //low speed case: allow for tail control, set camber angle up.

      //initiate tail position control
      manualTailControl();

      //set camber angle up.
      raiseCamberArm();
      //camberOn();

      //check for buttons that want to switch to takeoff/land or high speed or turn off.
      if (buttonPressPower()) {
        state = 0;

        stopMotorResponse();
      }
      if (buttonPressTOAL()) {
        state = 2;
      }
      if (buttonPressHS()) {
        state = 4;
      }

      break;


    case 4: //high speed case: allow for tail control, set camber angle down.

      //initiate tail position control
      manualTailControl();

      //set camber angle down to 0.
      lowerCamberArm();
      //camberOn();


      //check for buttons that want to switch to takeoff/land or low speed or turn off.
      if (buttonPressPower()) {
        state = 0;
        stopMotorResponse();
      }
      if (buttonPressTOAL()) {
        state = 2;
      }
      if (buttonPressLS()) {
        state = 3;
      }
      break;




    default: // should not happen
      Serial.println("SM_ERROR");
      break;
  }
}
```

```
//Event Checkers
bool buttonPressPower() { //turns on and off, zeroes position at start and at shutoff.  First press of the button sets in free mode.

  if (button1IsPressed == true) {
    Serial.println("PRESS");
    button1IsPressed = false;
    return true;
  }
  else {
    return false;
  }
}
bool buttonPressLS() { //low speed flight
  if (button2IsPressed == true) {
    button2IsPressed = false;
    return true;
  }
  else {
    return false;
  }
}
bool buttonPressHS() { //high speed flight
  if (button3IsPressed == true) {
    button3IsPressed = false;
    return true;
  }
  else {
    return false;
  }
}
bool buttonPressTOAL() { //takeoff and land mode
  if (button4IsPressed == true) {
    button4IsPressed = false;
    return true;
  }
  else {
    return false;
  }
}
```

```
//Event Service Responses
void stopMotorResponse() {//stop both motor responses.
  ledcWrite(ledChannel_2, LOW);
  ledcWrite(ledChannel_1, LOW);
  ledcWrite(ledChannel_4, LOW);
  ledcWrite(ledChannel_3, LOW);
  //ADD ledcWrite for the other motor
  digitalWrite(LED_PIN, LOW);
}

void startMotorResponse() {//start both motor responses.
  ledcWrite(ledChannel_1, LOW);
  ledcWrite(ledChannel_2, D);
  ledcWrite(ledChannel_3, LOW);
  ledcWrite(ledChannel_4, D2);
  //ADD ledcWrite for other motor
  digitalWrite(LED_PIN, HIGH);
}
void manualTailControl() {
  posDes = map(analogRead(POT), 0, 4095, -100, 100);
  tailPositionControl(posDes);
}
void zeroPos() {
  //sets both camber arm and tail to home position.
  lowerCamberArm();
  posDes = 0;
  tailPositionControl(posDes);
}
void raiseCamberArm() {
  //raise camber arm to predetermined angle.
  posDes2 = 50;
  camberPositionControl(posDes2);
}
void lowerCamberArm() {
  //zero camber arm angle.
  posDes2 = 0;
  camberPositionControl(posDes2);

}
```

```
void TOALpos() {
  //set tail angle to TOAL position of 25 degrees.
  posDes = -25;
  tailPositionControl(posDes);
}
```

```
void tailPositionControl(int posDes) { //controls position of the tail fishbone
  if (deltaT) {
    portENTER_CRITICAL(&timerMux1);
    deltaT = false;
    portEXIT_CRITICAL(&timerMux1);

    //posDes = map(analogRead(POT), 0, 4095, -1000, 1000); //map potentiometer values to output position of 200/8
    posErr = posDes - pos;


    //=========================================================  POSITION CONTROL SECTION
    if (abs(posErr) < 2) {
      D = 0;
    }
    else {
      omegaDes = posErr;
      if (abs(omegaDes) > 20) {
        omegaDes = omegaDes / abs(omegaDes) * 20;
      }

      omegaSpeed = count;
      err = omegaDes - omegaSpeed;
      sumErr = sumErr + err;


      D = KpTail * (err) + KiTail * (sumErr)/10;

    }

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


      //Map the D value to motor directionality
      //Assumes encoder direction is same as GSI
      if (D > 0) {
        ledcWrite(ledChannel_1, LOW);
        ledcWrite(ledChannel_2, D);
      }
      else if (D < 0) {
        ledcWrite(ledChannel_2, LOW);
        ledcWrite(ledChannel_1, -D);
      }
      else {
        ledcWrite(ledChannel_2, LOW);
        ledcWrite(ledChannel_1, LOW);
      }

      //plotControlData();

    }
}
```

```
void camberPositionControl(int posDes2) { //controls position of the camber arm angle.

  if (deltaT2) {
    portENTER_CRITICAL(&timerMux2);
    deltaT2 = false;
    portEXIT_CRITICAL(&timerMux2);


    posErr2 = posDes2 - pos2;


    //=========================================================  POSITION CONTROL SECTION
    if (abs(posErr2) < 2) {
      D2 = 0;
    }
    else {
      omegaDes2 = posErr2;
      if (abs(omegaDes2) > 20) {
        omegaDes2 = omegaDes2 / abs(omegaDes2) * 20;
      }

      omegaSpeed2 = count2;
      err2 = omegaDes2 - omegaSpeed2;
      sumErr2 = sumErr2 + err2;


      D2 = KpCam * (err2) + KiCam * (sumErr2)/10;

    }

    //Ensure that you don't go past the maximum possible command
    if (D2 > MAX_PWM_VOLTAGE) {
      D2 = MAX_PWM_VOLTAGE;
      sumErr2 -= err2; //anti-windup
    }
    else if (D2 < -MAX_PWM_VOLTAGE) {
      D2 = -MAX_PWM_VOLTAGE;
      sumErr2 -= err2; //anti-windup
    }
```

```
      D2 = KpCam * (err2) + KiCam * (sumErr2)/10;

    }

    //Ensure that you don't go past the maximum possible command
    if (D2 > MAX_PWM_VOLTAGE) {
      D2 = MAX_PWM_VOLTAGE;
      sumErr2 -= err2; //anti-windup
    }
    else if (D2 < -MAX_PWM_VOLTAGE) {
      D2 = -MAX_PWM_VOLTAGE;
      sumErr2 -= err2; //anti-windup
    }

    //Map the D value to motor directionality
    //Assumes encoder direction is same as GSI
    //CHANGE FOR LED CHANNEL 3/4
    if (D2 > 0) {
      ledcWrite(ledChannel_3, LOW);
      ledcWrite(ledChannel_4, D2);
    }
    else if (D2 < 0) {
      ledcWrite(ledChannel_4, LOW);
      ledcWrite(ledChannel_3, -D2);
    }
    else {
      ledcWrite(ledChannel_4, LOW);
      ledcWrite(ledChannel_3, LOW);
    }


    plotControlData2();

  }

}
```