```
/*********************************************************************

    Name    : Test Rig - Series Elastic Actuator
    Author  : Jonathan Jenkins
    Created : May 147, 2022
    Last Modified: May 147, 2022
    Version : 1.0

Key Information:
  - Revision 1.0
  - Board : Arduino uno
  - Screen : Monochrome OLEDs based on SSD1306
  - Screen : 128x32 size display using I2C


 Notes   : This code is for use with an Arduino Uno and LCD/button shield. The
              intent is for anyone to use this program to give them a starting
              program with a fully functional menu with minimal modifications
              required by the user.
*********************************************************************/



 // **************** INCLUDE NECESSARY LIBRARIES *****************

#include <SPI.h>
#include <Wire.h>

// **************** GLOBAL CONSTANT VARIABLES *****************       //
variables will not change:

// PIN VARIABLES:
  const int potpin                 = A2; //  the FSR and 10K pulldown
  const int led_ring_pin            = 2;    //  DIGITAL 2
  const int limitpin               = 3;
  const int key_pad_pin             = 7;    // ANALOG A7
  const int motor_enable_pin        = 8;
  const int motor_direction_pin     = 9;
  const int motor_on_pin            = 10;

 // ****************  GLOBAL VARIABLES (GLOBAL)****************        //
variables will change:

// PRIMARY PIN VARIABLES:
  int SerialSensorMore = false;         // PUSH BUTTON 1 --- ARM
  int task = false;
int position_hold_tollerance =0;
int task2 = false;
int task2limit= false;
int starttime = 0;
  int straingage = 445.77; // strain gage value
  int resistance = 10; // strain gage value
  int straingage2 = 445.77; // strain gage value
  int hallReading = 445.77; // strain gage value
  int newState = 0; // strain gage value
  int oldState = 0; // strain gage value
   int  sea_f1_measured = 100; // strain gage value
  int  sea_f2_measured = 100; // strain gage value

 // SECONDARY PIN VARIEABLES PER PEROJECT:
    //THRUST STAND
   /*
  int forceMeasured = 100; // PUSH BUTTON 1 --- ARM
  int tempmeasured = 100; // PUSH BUTTON 1 --- ARM
  int force2measured= 100; // PUSH BUTTON 1 --- ARM
  int force1measured = 100; // PUSH BUTTON 1 --- ARM
  int pushbutton1 = digitalRead(armpin); // PUSH BUTTON 1 --- ARM
  int pushbutton2 = digitalRead(testpin); // PUSH BUTTON 2 --- TEST
  int pushbutton3 = digitalRead(zeropin); // PUSH BUTTON 3 --- ZERO
  int armfunction = LOW; // initiate arm change
  int testfunction = LOW; // initiate Static Thrust Test
  int zerofunction = LOW; // initiate Zeroing of gage
  int armlimit = LOW; //  arm button limit
  int testlimit = LOW; // test button limit
  int zerolimit = LOW; // zero button limit

  int armstate = 0; // arm control
  int systemstate = 0; // system control
  int armtime = 0; // arm delay timer
  int testtime = 0; // test program delay timer
        unsigned long int now = 0;
        unsigned long int starttime = now;
        unsigned long int laptime = now-starttime;
  */
// **************** POTENTIOMETER *****************

int angle=0;
// **************** OLED LIBRARY *****************
//Required Libraries
```

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// OLED SCREEN  SETUP
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
// The pins for I2C are defined by the Wire-library.
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
// The pins for I2C are defined by the Wire-library.
// On an arduino UNO:       A4(SDA), A5(SCL)
// On an arduino MEGA 2560: 20(SDA), 21(SCL)
// On an arduino LEONARDO:   2(SDA),  3(SCL), ...

// this is the Width and Height of Display which is 128 xy 32
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels
#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C
for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
#define XPOS 0
#define YPOS 1
#define DELTAY 2
double count=0;


// **************** NEOPIXEL LIBRARY *****************
// Simple demonstration on using an input device to trigger changes on your
// NeoPixels. Wire a momentary push button to connect from ground to a
// digital IO pin. When the button is pressed it will change to a new pixel
// animation. Initial state has all pixels off -- press the button once to
// start the first animation. As written, the button does not interrupt an
// animation in-progress, it works only when idle.
//Required Library
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

// Digital IO pin connected to the button. This will be driven with a
// pull-up resistor so the switch pulls the pin to ground momentarily.

#define PIXEL_PIN    led_ring_pin  // Digital IO pin connected to the NeoPixels.
#define PIXEL_COUNT 35  // Number of NeoPixels

// Declare our NeoPixel strip object:
```

```
Adafruit_NeoPixel strip(PIXEL_COUNT, PIXEL_PIN, NEO_GRB + NEO_KHZ800);
// Argument 1 = Number of pixels in NeoPixel strip
// Argument 2 = Arduino pin number (most are valid)
// Argument 3 = Pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
//   NEO_RGBW    Pixels are wired for RGBW bitstream (NeoPixel RGBW products)
  unsigned long pixel_t = 0;        // pixel timer for animation
  long firstPixelHue = 0;
  int pixelLT = 0;         //DESIRED PIXEL
  int pixelRT = 0 ;        //Desired Pixel
  int pixeli=0;       //bit to index led for a full ring animation
  int pixelj=0;       //bit to index led for a full ring animation
  int pixelji=0;       //bit to index led for a full ring animation
  int pixelk=0;       //bit to index led for a full ring animation
  int pixelki=0;       //bit to index led for a full ring animation




// **************** HX711 SCALE LIBRARY *****************

#include <HX711_ADC.h>
#if defined(ESP8266)|| defined(ESP32) || defined(AVR)
#include <EEPROM.h>
#endif

//pins:
const int HX711_dout = 4; //mcu > HX711 dout pin
const int HX711_sck = 5; //mcu > HX711 sck pin
//HX711 constructor:
HX711_ADC LoadCell(HX711_dout, HX711_sck);
const int calVal_calVal_eepromAdress = 0;
unsigned long t = 0;


// ****************Menu *****************


// You can have up to 10 menu items in the menuItems[] array below without having
to change the base programming at all. Name them however you'd like. Beyond 10
items, you will have to add additional "cases" in the switch/case
// section of the operateMainMenu() function below. You will also have to add
```

```
additional void functions (i.e. menuItem11, menuItem12, etc.) to the program.
String menuItems[] = {" ","Hold","Oscillate"};
// Navigation button variables
int readKey;
// Menu control variables
int menuPage = 1;
int maxMenuPages = 2 ; // amount f menu items -1
int activepage=0;
int activeButton;
int button=0;
int buttonA=0;

int buttonlimit;
int lcdbutton=1000;

// ****************potentiometer *****************

float floatMap(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
int ANGLE_MAX = 270;







// ****************SETUP *****************
void setup() {
  Serial.begin(9600);
  Serial.println();
  Serial.println("Starting...");


// ****************  OLED SETUP *****************
  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  // Clear the buffer
  display.clearDisplay();
  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();

// ****************  NEOPIXEL SETUP *****************
 // These lines are specifically to support the Adafruit Trinket 5V 16 MHz.
  // Any other board, you can remove this part (but no harm leaving it):
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif
  // END of Trinket-specific code.

  strip.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)
  strip.show();            // Turn OFF all pixels ASAP
  strip.setBrightness(50); // Set BRIGHTNESS to about 1/5 (max = 255)









     pinMode(motor_enable_pin, OUTPUT);
     pinMode(motor_direction_pin, OUTPUT);
     pinMode(motor_on_pin, OUTPUT);





// ****************  Scale 1 SETUP *****************
float calibrationValue; // calibration value
  calibrationValue = 95564.68; // uncomment this if you want to set this value in
the sketch
#if defined(ESP8266) || defined(ESP32)
  //EEPROM.begin(512); // uncomment this if you use ESP8266 and want to fetch
this value from eeprom
#endif
  //EEPROM.get(calVal_eepromAdress, calibrationValue); // uncomment this if you
want to fetch this value from eeprom

  LoadCell.begin();
  unsigned long stabilizingtime = 2000; // tare preciscion can be improved by
adding a few seconds of stabilizing time
  boolean _tare = true; //set this to false if you don't want tare to be
performed in the next step
  LoadCell.start(stabilizingtime, _tare);
```

```
   if (LoadCell.getTareTimeoutFlag()) {
  }
  else {
    LoadCell.setCalFactor(calibrationValue); // set calibration factor (float)
  }
  while (!LoadCell.update());


// ****************  IO SETUP *****************
   /*



   // initialize Digital Input and Outputs

   //THRUST STAND

     pinMode(mosfetpin, OUTPUT);
     pinMode(armpin,INPUT_PULLUP);
     pinMode(testpin,INPUT_PULLUP);
     pinMode(zeropin,INPUT_PULLUP);
     digitalWrite(mosfetpin,LOW);
     systemstate=1;
  */








  Serial.println("READY");

  delay(100);
}
// ****************End SETUP ***********************





// ****************  Main Loop *****************

void loop() {
    scale();   // function that controls gage zeroing
    poteniometer();   // function that controls gage zeroing
    menu();         // Function that controls Oled Screen
    Serial.println(straingage);

}

// **************** End Main Loop *****************
```