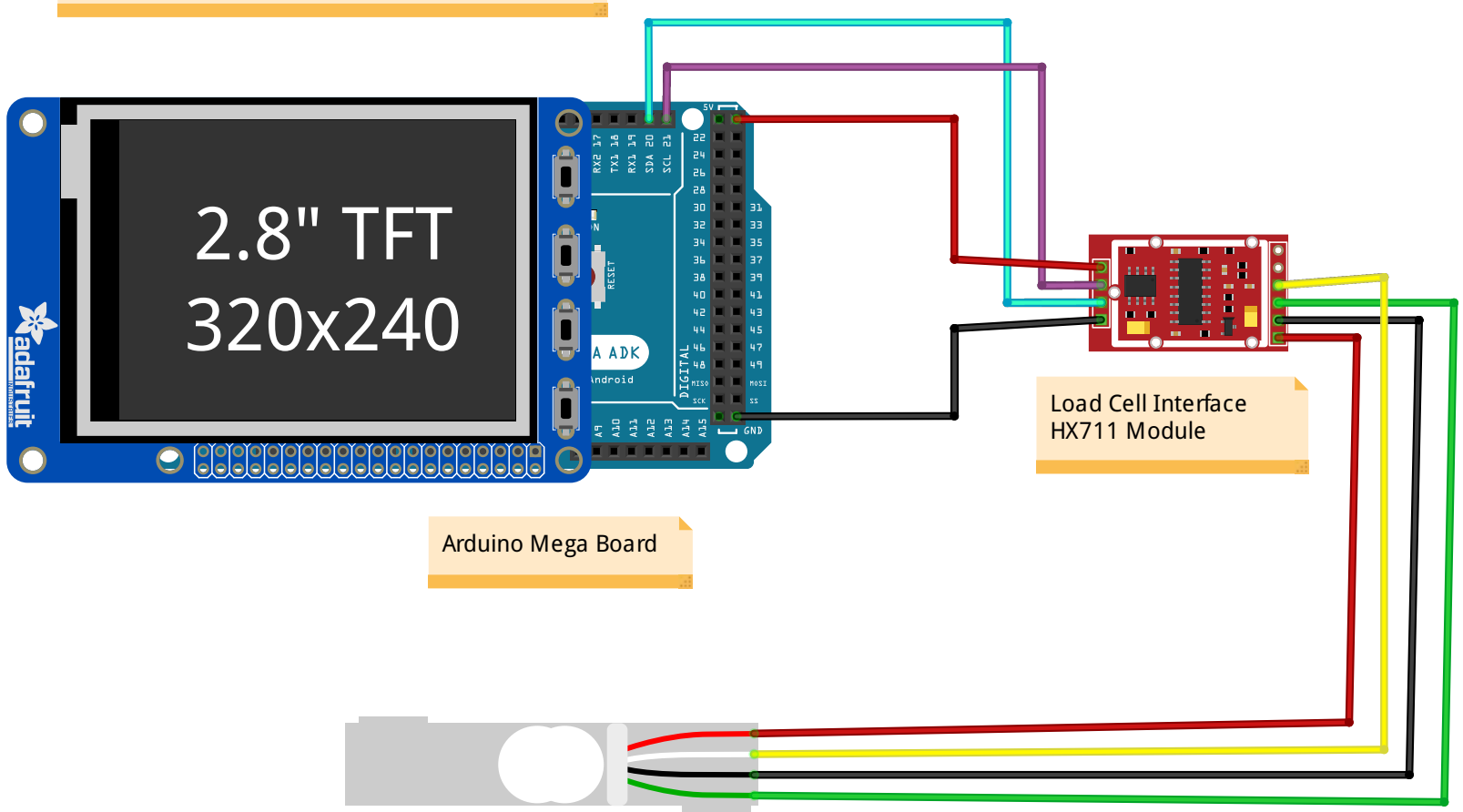
Propane Tank Scale:
- This circuit reads propane tank level in %
- This circuit also read the total scale weight
- Incorporates a self zero which is automatic
- Also has a secondary Menu for setting the Tare Weight (Zero) and calibrating the sclae (span)

The TFT screen was from Amazon Elegoo EL-SM-004 R3 2.8 Inches TFT Touch Screen. It Goes on Top of an Arduino Mega Board, and has the folowing Connections (A2,A3 swap functions in code, hardware connection is the same):
Display-
LCD_CS A3 // Chip Select goes to Analog 3
LCD_CD A2 // Command/Data goes to Analog 2
LCD_WR A1 // LCD Write goes to Analog 1
LCD_RD A0 // LCD Read goes to Analog 0
LCD RST // LCD Reset goes to Analog 4
Data Transfer-
LCD D0 // Digital 8
LCD D1 // Digital 9
LCD D2 // Digital 2
LCD D3 // Digital 3
LCD D4 // Digital 4
LCD D5 // Digital 5
LCD D6 // Digital 6
LCD D7 // Digital 7
Touch Screen-
YP A3 // Analog 3
XM A2 // Analog 2
YM 9 // Digital 9
XP 8 // Digital 8
Power-
5 Volts
3.3 Volts
Gnd

Note - Some LCD Data Pins may not be used if utilizing the Adafruit Library to Drive the TFT Display.



0-20 Kg Load Cell

Amazon Part Number:
Bolsen 20Kg Sensor with HX711 Interface Module

```
/*
 TFT Load Cell Program to determine Propane Tank Level
 using an Arduino Mega Board and offers the following features:
 1) A Adjustable zero that stores an empty tank level (Tare Weight), and places in EEPROM
 2) A Span Adj to set an accurate weight and % Tank level, and places in EEPROM
 2) Displays Tank Level
 3) Connects to a 20KG Load cell, and uses IIC interface, and converts to Lbs
 4) Utilizes an interrupt service routine (ISR) to provide an accurate 1 second timebase
 5) Displays any communication errors with sensor (if they happen)
 */

// Libraries to be Included
// ------------------------
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_TFTLCD.h> // Hardware-specific library
#include <TouchScreen.h>     // Touchscreen library
#include <Wire.h>  // Arduino hardware library, required for touch screen
#include <HX711.h> // Load cell signal conditioning library
#include <EEPROM.h> // EEPROM Library

// The control pins for the LCD can be assigned to any digital or
// analog pins...but we'll use the analog pins as this allows us to
// double up the pins with the touch screen (see the TFT paint example)
// ---------------------------------------------------------------------
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0
#define LCD_RESET A4 // Can alternately just connect to Arduino's reset pin
#define YP A3  // must be an analog pin, use "An" notation!
#define XM A2  // must be an analog pin, use "An" notation!
#define YM 9   // can be a digital pin
#define XP 8   // can be a digital pin

// Touch Screen Min & Max Parameters For New ILI9341 TP
// ----------------------------------------------------
#define TS_MINX 120
#define TS_MAXX 900
#define TS_MINY 70
#define TS_MAXY 920

// When using the TFT BREAKOUT BOARD only, use these 8 data lines to the LCD:
// For the Arduino Uno, Duemilanove, Diecimila, etc.:
//   D0 connects to digital pin 8  (Notice these are
//   D1 connects to digital pin 9   NOT in order!)
//   D2 connects to digital pin 2
//   D3 connects to digital pin 3
//   D4 connects to digital pin 4
//   D5 connects to digital pin 5
//   D6 connects to digital pin 6
//   D7 connects to digital pin 7
// For the Arduino Mega, use digital pins 22 through 29
// (on the 2-row header at the end of the board).

// Assign readable names to some common 16-bit color values:
// ---------------------------------------------------------
#define BLACK       0x0000
#define BLUE        0x001F
#define RED         0xF800
#define GREEN       0x07E0
#define CYAN        0x07FF
#define MAGENTA     0xF81F
#define YELLOW      0xFFE0
#define WHITE       0xFFFF
#define NAVY        0x000F
#define DARKGREEN   0x03E0
#define DARKCYAN    0x03EF
#define MAROON      0x7800
#define PURPLE      0x780F
```

```
#define OLIVE       0x7BE0
#define LIGHTGREY   0xC618
#define DARKGREY    0x7BEF
#define ORANGE      0xFD20
#define GREENYELLOW 0xAFE5
#define PINK        0xF81F

// SET Menu Box (1rst menu)
// ------------------------
#define BOXSIZE 40
#define MENU_BUTTON_X 180
#define MENU_BUTTON_Y 0
#define MENU_WIDTH 50
#define MENU_HEIGHT 40

//Button UI details (2nd menu)
// ---------------------------
#define BUTTON_X 70
#define BUTTON_Y 120
#define BUTTON_W 60
#define BUTTON_H 30
#define BUTTON_SPACING_X 50
#define BUTTON_SPACING_Y 30
#define BUTTON_TEXTSIZE 2

// Text box where the Second Menu Goes
// -----------------------------------
#define TEXT_X 20
#define TEXT_Y 10
#define TEXT_W 100
#define TEXT_H 30

// Touch Screen Pressure Limits
// ----------------------------
#define MINPRESSURE 10
#define MAXPRESSURE 1000

// Global Variables
// ----------------
unsigned long period1 = 5000;  // 5 second update
unsigned long period2 = 200;  // 200mS update
unsigned long time_now1 = 0, time_now2 = 0;
boolean
  EraseFlag = false;
int w,h,x,y,r; // Coordinates
float zero,span;  // Global Calibration factors
volatile word count=0; // Any variables inside interrupt need to be declared as volatile
volatile int INT_flag = 0; // Set interrupt flag to "0"
int menu_flag = 0; // Set time menu flag
int pos_x; // "X" Touchscreen position
int pos_y; // "Y" Touchscreen position
String z,s; // String variable for zero and span TFT Display
const int LOADCELL_DOUT_PIN = 20;  // Load cell Board IIC Interface  data pin
const int LOADCELL_SCK_PIN = 21;  // Load cell Board IIC Interface clock pin
float Kg;  // Load cell output
const float lbs = 2.205;  // Kg to Lbs conversion
float weight;  // Scale output
float weight1;  // Constrained output for % Full Calculation
float zero1;    // Constrained scale tare weight (offset) % Full Calculation
float percent_full; // Tank Level Calculated based on load cell weight and tank tare
weight

// Set up Class Objects
// --------------------
TouchScreen  ts = TouchScreen(XP, YP, XM, YM, 300);  // Touchscreen
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);  // Display
HX711 scale; // Scale load cell Interface

// Create 4 Buttons
```

```
138  // ----------------
139  /* create 4 buttons, in classic candybar phone style */
140  char buttonlabels[6][6] = {"UP", "DWN", "UP", "DWN", "CLR", "ENT"};
141  uint16_t buttoncolors[6] = {GREEN, RED, GREEN, RED, MAGENTA, BLUE};
142  Adafruit_GFX_Button buttons[6];
143
144  // Setup Function
145  // --------------
146  void setup() {
147    Serial.begin(9600);
148   // TFT Setup Stuff:
149  #ifdef USE_ADAFRUIT_SHIELD_PINOUT
150    Serial.println(F("Using Adafruit 2.8\" TFT Arduino Shield Pinout"));
151  #else
152    Serial.println(F("Using Adafruit 2.8\" TFT Breakout Board Pinout"));
153  #endif
154    tft.reset();
155    tft.begin(0x9341); // Start TFT Display, and address
156    tft.setRotation(2);  // Rotate Screen 90 Degrees
157    tft.fillScreen(BLACK); // TFT background color Erase
158    PrintText(0, 10, 3, RED, " Smart Scale"); // Goto Function
159    tft.println(); // Space
160    tft.println(); // Space
161    tft.setTextColor(BLUE); // TFT Color
162    tft.setTextSize(2); // Fontsize
163    tft.println("Used to Weigh Tanks"); //Text to Display
164    tft.println(); // Space
165    tft.println(); // Space
166    tft.setTextColor(GREEN);  // TFT Color
167    tft.setTextSize(2); // Font size
168    tft.println(" Includes an Offset"); //Text to Display
169    tft.println(); // Space
170    tft.println(" To Subtract From"); //Text to Display
171    tft.println(); // Space
172    tft.println(); // Space
173    tft.println(); // Space
174    tft.setTextColor(MAGENTA);  // TFT Color
175    tft.println(" By, Roy H Guerra Jr"); //Text to Display
176    delay(3000); // 3 second delay
177    tft.fillScreen(BLACK); // TFT background color Erase
178    PrintText(2, 10, 3, RED, " Scale Zero"); // Goto Function
179    tft.println(); // Space
180    tft.println(); // Space
181    tft.setTextColor(BLUE); // TFT Color
182    tft.setTextSize(2); // Fontsize
183    tft.println(" This will Begin in"); //Text to Display
184    tft.println(" 10 seconds:"); //Text to Display
185    tft.println(); // Space
186    tft.println("Note - The Zero Adj."); //Text to Display
187    tft.println("Is the Tare Weight"); //Text to Display
188    tft.println(); // Space
189    tft.println("Note - The Span Adj."); //Text to Display
190    tft.println("Is for Weight Cal."); //Text to Display
191    tft.println(); // Space
192    tft.setTextColor(GREEN);  // TFT Color
193    tft.setTextSize(2); // Font size
194    tft.println(" Be sure to Remove"); //Text to Display
195    tft.println(); // Space
196    tft.println(" All Weight Now"); //Text to Display
197    delay(10000); // 10 second delay
198    Serial.println("Initializing the scale");
199    scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
200    scale.set_scale(2280.f); // 2280; this value is obtained by calibrating the scale
       with known weights; see the README for details
201    scale.tare();               // reset the scale to 0, with no weight
202    tft.fillScreen(BLACK); // TFT background color Erase
203    PrintText(2, 10, 3, RED, " Scale Zero"); // Goto Function
204    tft.println(); // Space
205    tft.println(" Is Complete"); //Text to Display
```

```
206     delay(2000); // 2 second delay
207     // obtain measure of screen
208     w = tft.width();
209     Serial.println("TFT Width = " + String(w));
210     h = tft.height();
211     Serial.println("TFT Height = " + String(h));
212     // radius and center of the clock
213     r = (min(w,h)/2) - 1;
214     Serial.println("Radius = " + String(r));
215     x = w/2,
216     Serial.println("X = " + String(x));
217     y = h - x;
218     Serial.println("Y = " + String(y));
219     DrawDial(); // Goto function
220     //tft.drawRect(TEXT_X, TEXT_Y, TEXT_W, TEXT_H, WHITE);  // create a blank 'text
        field'
221     pinMode(13, OUTPUT);  // Set on Board LED to Output ISR status
222     INT_Init(); // Goto Interrupt Setup Function
223     zero = EEPROM.get(0, zero); // Get zero value in EEPROM
224     span = EEPROM.get(8, span); // Get span value in EEPROM
225     Serial.println("Getting zero and span floats from EEPROM!");
226     Serial.println(zero); // Debug
227     Serial.println(span); // Debug
228     if (isnan((zero) || (span))) { // Store fake values in EEPROM if blank, to aviod
        errors
229         EEPROM.put(0, 10.34f); // Store fake zero value in EEPROM
230         EEPROM.put(8, 1.01f); // Store fake span value in EEPROM
231         Serial.println("Wrote zero and span floats to EEPROM!"); // Debug
232     }
233   }
234
235   // Function to Draw Scale Dial
236   // ---------------------------
237   void DrawDial()
238   {
239     tft.fillScreen(BLACK);
240     //draw ten circles as the outside of the clock with the center at the radius
241     for (int i=0; i<10; i++){
242     tft.drawCircle(x, y, r-i,(CYAN));
243     //tft.fillCircle(x, y, r-i,(CYAN));
244     }
245      // Draw Menu Box
246      tft.drawRect(MENU_BUTTON_X, MENU_BUTTON_Y, MENU_WIDTH, MENU_HEIGHT, WHITE);  // Draw
        Menu Box
247      tft.setTextColor(BLUE);
248      tft.setTextSize(2);
249      tft.setCursor(MENU_BUTTON_X + 8, MENU_BUTTON_Y + 14);
250      tft.print("CAL");
251      PrintText(0, 50, 3, RED, " Smart Scale"); // Goto Function
252      PrintText(70, 120, 3, GREEN, "% Full"); // Goto Function
253      tft.drawRect(MENU_BUTTON_X-145, MENU_BUTTON_Y+155, MENU_WIDTH+120, MENU_HEIGHT+15,
        WHITE);  // Draw Display Box
254      //PrintText(60, 170, 4, GREEN, "50.78"); // Goto Function, used to Debug! Comment
        out after position.
255      PrintText(58, 223, 2, YELLOW, "Weight(lbs)"); // Goto Function
256      tft.drawRect(MENU_BUTTON_X-105, MENU_BUTTON_Y+250, MENU_WIDTH+45, MENU_HEIGHT-10,
        WHITE);  // Draw Display Box
257      //PrintText(95, 257, 2, YELLOW, "42.18"); // Goto Function, used to Debug! Comment
        out after position.
258   }
259   // Function to Draw 2nd menu & Buttons
260   // -----------------------------------
261   void Cal_Menu() { // 2nd menu
262     tft.fillScreen(BLACK);
263     zero = EEPROM.get(0, zero); // Get zero value in EEPROM
264     span = EEPROM.get(8, span); // Get span value in EEPROM
265     Serial.println("Getting zero and span floats from EEPROM!");
266     Serial.println(zero, 3);
267     Serial.println(span, 3);
```

```
268     z = String(zero);
269     PrintText(41, 53, 2, MAGENTA, z); // Goto Function
270     s = String(span);
271     PrintText(153, 53, 2, CYAN, s); // Goto Function
272     PrintText(2, 5, 3, YELLOW, "Calibration"); // Goto Function
273     PrintText(7, 51, 2, MAGENTA, "Z="); // Goto Function
274     tft.drawRect(TEXT_X+15, TEXT_Y+35, TEXT_W-30, TEXT_H, WHITE); // Draw Display Box
275     PrintText(118, 51, 2, CYAN, "S="); // Goto Function
276     tft.drawRect(TEXT_X+125, TEXT_Y+35, TEXT_W-30, TEXT_H, WHITE); // Draw Display Box
277     PrintText(59, 141, 2, MAGENTA, "Zero Adj.(Z)"); // Goto Function
278     PrintText(59, 201, 2, CYAN, "Span Adj.(S)"); // Goto Function
279     PrintText(74, 261, 2, ORANGE, "Functions"); // Goto Function
280     // Create buttons for Display
281     for (uint8_t row=0; row<3; row++) { // 3 rows
282       for (uint8_t col=0; col<2; col++) { // 2 columns
283         buttons[col + row*2].initButton(&tft, BUTTON_X+col*(BUTTON_W+BUTTON_SPACING_X),
284                     BUTTON_Y+row*(BUTTON_H+BUTTON_SPACING_Y),    // x, y, w, h, outline,
                    fill, text, row * 2 columns
285                      BUTTON_W, BUTTON_H, WHITE, buttoncolors[col+row*2], WHITE,
286                      buttonlabels[col + row*2], BUTTON_TEXTSIZE);
287         buttons[col + row*2].drawButton();
288       }
289     }
290   }
291
292   // Function to Read the Scale Weight
293   // ---------------------------------
294   void ReadScale(){
295     if (scale.is_ready()) {
296       tft.fillRect(10,48,210,28,BLACK);  // Used to erase display field
297       PrintText(0, 50, 3, RED, " Smart Scale"); // Goto Function
298       Kg = abs(scale.get_units(5)); // print the average of 5 readings from the ADC minus
      tare weight, divided by the SCALE parameter set with set_scale
299       Serial.println("Kg = " + String(Kg));
300       weight = abs(Kg*lbs*span); // KG to lbs conversion
301       weight = constrain(weight, 0.0, 50.00);  // Constrain load cell readings between 0
      and 44.1 lbs
302       zero1 = constrain(zero, 0.00, zero);  // Constrain zero to get accurate tank level
      (no negative)
303       weight1 = constrain(weight, 0.00, (2*zero1));  // Constrain weight to liquid
      measure of tank
304       percent_full = ((weight1/zero1)*100)/2; // Tank Calculation for % Full
305       Serial.println("Zero1 = " + String(zero1));
306       Serial.println("Weight1 = " + String(weight1));
307       Serial.println("Lbs = " + String(weight));
308       Serial.println("% Full = " + String(percent_full));
309
310
311       tft.fillRect(MENU_BUTTON_X-143, MENU_BUTTON_Y+159, MENU_WIDTH+115, MENU_HEIGHT+5,
      BLACK);  // Draw Display Box
312       PrintText(60, 170, 4, GREEN, String(percent_full)); // Goto Function, used to
      Debug! Comment out after position.
313       tft.fillRect(MENU_BUTTON_X-103, MENU_BUTTON_Y+254, MENU_WIDTH+40, MENU_HEIGHT-20,
      BLACK);
314       if (weight > 44.1) {
315         PrintText(95, 257, 2, YELLOW, "OVER"); // Goto Function, used to Debug! Comment
          out after position.
316       }
317       else{
318         PrintText(95, 257, 2, YELLOW, String(weight)); // Goto Function, used to Debug!
          Comment out after position.
319       }
320     }
321      else {
322       Serial.println("HX711 not found.");
323       tft.fillRect(10,48,210,28,BLACK);  // Used to erase display field
324       PrintText(0, 50, 3, RED, " COMM ERROR!"); // Goto Function
325     }
326   }
```

```
327
328   // Function to Read Touch Screen
329   // ----------------------------
330   void readTouchScreen(){
331       if (menu_flag != 1){
332         TSPoint p = ts.getPoint();
333         // if sharing pins, you'll need to fix the directions of the touchscreen pins
334         //pinMode(XP, OUTPUT);
335         pinMode(XM, OUTPUT);
336         pinMode(YP, OUTPUT);
337         //pinMode(YM, OUTPUT);
338         // Get measured values
339         if (p.z > MINPRESSURE && p.z < MAXPRESSURE) { // scale from 0->1023 to tft.width
340            p.x = map(p.x, TS_MINX, TS_MAXX, tft.width(), 0);
341            p.y = (tft.height()-map(p.y, TS_MINY, TS_MAXY, tft.height(), 0));
342            Serial.print("p.x="); Serial.println(p.x);
343            Serial.print("p.y="); Serial.println(p.y);
344            pos_x = p.x; // Return global variable for "X" position
345            pos_y = p.y; // Return global variable for "Y" position
346          }
347       }
348   }
349   // Function to Format Data
350   // -----------------------
351   void PrintText(int x, int y, int textSize, int color, String(text)){
352      tft.setCursor(x, y);
353      tft.setTextColor(color);
354      tft.setTextSize(textSize);
355      tft.println(text);
356   }
357   // Main Program Loop
358   // -----------------
359   void loop() {
360     if ((INT_flag == 1) && (menu_flag == 0)){ // Check for 1 second interrupt
361       INT_flag = 0; // Set interrupt flag to "0"
362     }
363     if ((pos_x > 177 && pos_x < 240) && (pos_y > 1 && pos_y < 46) && (menu_flag == 0)) {
364       pos_x = -1; // Default return position
365       pos_y = -1; // Default return position
366       menu_flag = 1;  // Set menu flag to "1"
367       tft.fillScreen(BLACK); // Erase Screen
368       Cal_Menu(); // Goto menu Function
369       do { // Locks the menu in Cal mode
370         //Cal_Menu(); // Goto menu Function
371         TSPoint p = ts.getPoint();  // Get X&Y values from touch screen
372         Serial.println("Point is: ");
373         Serial.print("X = "); Serial.print(p.x);
374         Serial.print("Y = "); Serial.print(p.y);
375         Serial.print("Pressure = "); Serial.println(p.z);
376
377         pinMode(XM, OUTPUT);  // Set the pin as an output
378         pinMode(YP, OUTPUT);  // Set the pin as an output
379
380         if (p.z > MINPRESSURE && p.z < MAXPRESSURE) { // scale from 0->1023 to
          tft.width
381           p.x = map(p.x, TS_MINX, TS_MAXX, tft.width(), 0);
382           p.y = (tft.height()-map(p.y, TS_MINY, TS_MAXY, tft.height(), 0));
383          }
384
385         for (uint8_t b=0; b<6; b++) { // go thru all the buttons, checking if pressed
386         if (buttons[b].contains(p.x, p.y)) {
387         Serial.print("Pressing: "); Serial.println(b);
388         buttons[b].press(true);  // tell the button it is pressed
389         } else {
390         buttons[b].press(false);  // tell the button it is NOT pressed
391         }
392        }
393        // now we can ask the buttons if their state has changed
394        for (uint8_t b=0; b<6; b++) {
```

```
395         if (buttons[b].justReleased()) {
396          Serial.print("Released: "); Serial.println(b);
397          buttons[b].drawButton();  // draw normal buttons (no change)
398         }
399          if (buttons[b].justPressed()) {
400            buttons[b].drawButton(true);  // draw invert to show button change
401            switch (b) {  // Perform actions when button is pressed
402            case 0:
403               if (zero < 25.0) {  // Do not go past 25
404                tft.fillRect(TEXT_X+17, TEXT_Y+39, TEXT_W-35, TEXT_H-10, BLACK);  // This
                   clears last position by turning the section "black"
405                zero += 0.1; // Increase count
406                z = String(zero);
407                PrintText(41, 53, 2, MAGENTA, z); // Goto Function
408                Serial.println("Zero = " + String(zero));
409               }
410              break;
411            case 1:
412              if (zero > 0.1) {  // Do not go past 0
413                tft.fillRect(TEXT_X+17, TEXT_Y+39, TEXT_W-35, TEXT_H-10, BLACK);  // This
                   clears last position by turning the section "black"
414                zero -= 0.1; // Decrease count
415                z = String(zero);
416                PrintText(41, 53, 2, MAGENTA, z); // Goto Function
417                Serial.println("Zero = " + String(zero));
418               }
419              break;
420             case 2:
421              if (span < 1.50) {  // Do not go past 5
422                tft.fillRect(TEXT_X+127, TEXT_Y+39, TEXT_W-35, TEXT_H-10, BLACK);  //
                   This clears last position by turning the section "black"
423                span += 0.01; // Increase count
424                s = String(span);
425                PrintText(153, 53, 2, CYAN, s); // Goto Function
426                Serial.println("Span = " + String(span));
427               }
428              break;
429             case 3:
430              if (span > 0.01) {  // Do not go past 0
431                tft.fillRect(TEXT_X+127, TEXT_Y+39, TEXT_W-35, TEXT_H-10, BLACK);  //
                   This clears last position by turning the section "black"
432                span -= 0.01; // Decrease count
433                s = String(span);
434                PrintText(153, 53, 2, CYAN, s); // Goto Function
435                Serial.println("Span = " + String(span));
436               }
437              break;
438            case 4:
439                zero = 0.0;
440                span = 0.0;
441                tft.fillRect(TEXT_X+17, TEXT_Y+39, TEXT_W-35, TEXT_H-10, BLACK);  // This
                   clears last position by turning the section "black"
442                z = String(zero);
443                PrintText(41, 53, 2, MAGENTA, z); // Goto Function
444                tft.fillRect(TEXT_X+127, TEXT_Y+39, TEXT_W-35, TEXT_H-10, BLACK);  //
                   This clears last position by turning the section "black"
445                s = String(span);
446                PrintText(153, 53, 2, CYAN, s); // Goto Function
447              break;
448            case 5:
449              EEPROM.put(0, zero); // Store in zero value in EEPROM
450              EEPROM.put(8, span); // Store span value in EEPROM
451              Serial.println("Wrote zero and span floats to EEPROM!");
452              menu_flag = 0; // Reset menu flag to display clock screen
453              DrawDial(); // Goto function
454              EraseFlag == true; // Set Erase Flag
455              break;
456            default:
457              break;
```

```
458             delay(100); // button debouncing
459             }
460           }
461         }
462       } while (menu_flag != 0); // Do routine while not in mian menu
463     }
464     if ((millis() - time_now1 > period1) && (menu_flag == 0)){  // If time passed and in
        main menu
465       ReadScale(); // Goto Function to read scale
466       time_now1 = millis();
467     }
468     if (millis() - time_now2 > period2){  // Enable touchscreen every 200mS to set time
469       readTouchScreen(); // Goto Function
470       time_now2 = millis();
471     }
472   }
473   // Interrupt Setup Function
474   // ------------------------
475   void INT_Init() {  // Set up Timer Interrupt for 1S
476     cli();           // Disable global interrupts
477     TCCR1A = 0;      // Set entire TCCR1A register to 0
478     TCCR1B = 0;      // Same for TCCR1B
479     TCNT1 = 0;       // Set counter variable to 0
480     // set compare match register to set sample time 1s
481     // Note - change OCR1A if time is Fast / Slow (1S = 15624) if time is 1.5 min slow
        per day;
482     // 1.5 min = 90 seconds,so error is 90 sec/day, and there are 86,400 sec/day. of slow
        error is (-)
483     // -error = (90/86400)*100 = 0.104%
484     // New (OCR1A) = [15624 - (15624*(0.104/100)) = 15607
485     OCR1A = 15607;  // preset = [16E6 / (prescale * (Hz))] - 1
486     TCCR1B |= (1 << WGM12);  // turn on CTC mode
487     TCCR1B |= (1 << CS12) | (1 << CS10) ; // Set CS12 and CS10 bits for prescaling by
        1024
488     TIMSK1 |= (1 << OCIE1A); // enable timer compare interrupt
489     sei();           // enable global interrupts
490   }
491   // Timer1 Interrupt Routine
492   // ------------------------
493   ISR(TIMER1_COMPA_vect) { // The ISR will be called every 1 second to keep accurate time
494     digitalWrite(13, !digitalRead(13));
495     INT_flag = 1; // Set interrupt flag to "1"
496     count++;  // Increase counter by one
497     if(count == 300) {  // 300 counts is 5 minutes
498       // Place function here
499       count = 0; // Reset counting variable
500     }
501   }
502
503
504
```