Arduino Mega Clock
With Analog & Digital time displays, and a secondary menu to set the time..

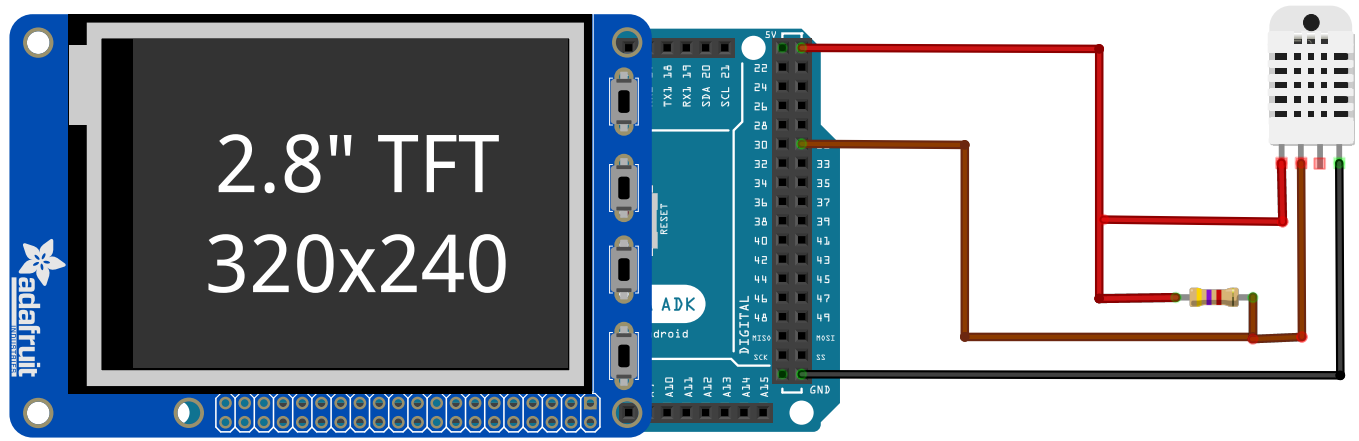
Clock also displays indoor Temperature and Humidity

DHT22 Sensor
- Red Wire = 5.0V
- Black Wire = Gnd
- Cyan Wire = Data Pin

Note - Use a 4.7K Pullup

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.



```
/*
 Clock Circuit the offers the following features:
 1) A thin film transistor (TFT) Touch display to set time, and display parameters
 2) Contains a seperate setime setup menu
 3) Displays analog time and digital time
 4) Displays indoor temperature and humidity
 5) Contains a timeout function to return to display??
 6) Utilizes an interrupt service routine (ISR) to provide an accurate 1 second timebase
 7) Temp. & Humidity updated once a minute
 */

// 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 <DHT.h>  // DHT Temperature / Humidity 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 human-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 FG_COLOR WHITE
#define BG_COLOR BLACK
#define MINUTE_COL YELLOW
#define HOUR_COL RED

// Position of time in text
```

```
// ------------------------
#define H_HOUR  10
#define H_MIN  60
#define H_SEC  110
#define SIZE_TIME 3
#define X_COL_TIME 10

// 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 for Time Display Goes
// ----------------------------------------------------
#define TEXT_X 50
#define TEXT_Y 50
#define TEXT_W 155
#define TEXT_H 40

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

// DHT Sensor Characteristics
// --------------------------
#define DHTPIN 31 //select pin
#define DHTTYPE DHT22   // DHT 22 Change this if you have a DHT11
DHT dht(DHTPIN, DHTTYPE);

// Global Variables
// ----------------
const float DegToRad = 0.0174533;  // Degreess to Radians conversion (pi/180)
float Hum; // Humidity storage Variable
float TemperatureC;  // Temperature storage variable for Deg C
float TempF;  // Temperature storage variable for Deg F
unsigned long period1 = 60000;  // 1 minute update
unsigned long period2 = 200;  // 200mS update
unsigned long time_now1 = 0, time_now2 = 0;
unsigned int
  seconds = 0, secondsOld = 0,
  minutes = 30, minutesOld = 0,
  hour = 10, hourOld = 0;
boolean
  incMinutes = false, incHour = false, EraseFlag = false;
int   w,h;
// Time Variables
int x, y, r, sr, mr, hr,
    sx = 0, sy = 0,
    mx = 0, my = 0,
    hx = 0, hy = 0;
// Scalar for Analog Hands
float a;
// Interrupt variables
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

// Set up Class Objects
// --------------------
TouchScreen  ts = TouchScreen(XP, YP, XM, YM, 300);
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);

// Create 8 buttons, in classic candybar phone style when function is called
// -------------------------------------------------------------------------
char buttonlabels[8][8] = {"Hr+", "Hr-", "Min+", "Min-", "Sec+", "Sec-", "Clr", "Ent"};
uint16_t buttoncolors[8] = {GREEN, RED, GREEN, RED, GREEN, RED, MAGENTA, BLUE};
Adafruit_GFX_Button buttons[8];

// Setup Function
// --------------
void setup() {
  Serial.begin(9600);
 // TFT Setup Stuff:
#ifdef USE_ADAFRUIT_SHIELD_PINOUT
  Serial.println(F("Using Adafruit 2.8\" TFT Arduino Shield Pinout"));
#else
  Serial.println(F("Using Adafruit 2.8\" TFT Breakout Board Pinout"));
#endif
  dht.begin(); // Initialize DHT sensor
  tft.reset();
  tft.begin(0x9341); // Start TFT Display, and address
  tft.setRotation(2);  // Rotate Screen 90 Degrees
  tft.fillScreen(BLACK); // TFT background color
  tft.setCursor(0, 10);   // Initial position
  tft.setTextColor(RED); // TFT Color
  tft.setTextSize(3); // Fontsize
  tft.println(" Smart Clock"); //Text to Display
  tft.println(); // Space
  tft.println(); // Space
  tft.setTextColor(BLUE); // TFT Color
  tft.setTextSize(2); // Fontsize
  tft.println("  Analog & Digital"); //Text to Display
  tft.println(); // Space
  tft.println(); // Space
  tft.setTextColor(GREEN);  // TFT Color
  tft.setTextSize(2); // Font size
  tft.println("  With Temperature"); //Text to Display
  tft.println(); // Space
  tft.println("  And Humidity"); //Text to Display
  delay(3000); // 3 second delay
  // obtain measure of screen
  w = tft.width();
  Serial.println("TFT Width = " + String(w));
  h = tft.height();
  Serial.println("TFT Height = " + String(h));
  // radius and center of the clock
  r = (min(w,h)/2) - 1;
  Serial.println("Radius = " + String(r));
  x = w/2,
  Serial.println("X = " + String(x));
  y = h - x;
  Serial.println("Y = " + String(y));
  // radius line for seconds, minutes and hour (subtract "10" from circle draw)
  sr = r-10;
  mr = sr-10;
  hr = mr-10;
  paintClockScreen(); // Goto function
  SensorData(); // Goto Function
  pinMode(13, OUTPUT);  // Set on Board LED to Output ISR status
  INT_Init(); // Goto Interrupt Setup Function
}
// Function to Draw Clock Face
```

```
208  // ---------------------------
209  void paintClockScreen()
210  {
211    tft.fillScreen(BG_COLOR);
212    //draw ten circles as the outside of the clock with the center at the radius
213    for (int i=0; i<10; i++){
214    tft.drawCircle(x, y, r-i,(CYAN));
215    //tft.fillCircle(x, y, r-i,(CYAN));
216    }
217    //draw numbers on clock face
218     tft.drawChar(x-10, y-r, 49, WHITE, BLACK, 2); // Print Character "1"(49)
219     tft.drawChar(x, y-r, 50, WHITE, BLACK, 2); // Print Character "2"(50), this forms 12
        with above
220     tft.drawChar(x-r, y-8, 57, WHITE, BLACK, 2); // Print Character "9"(57)
221     tft.drawChar(x+r-10, y-8, 51, WHITE, BLACK, 2); // Print Character "3"(51)
222     tft.drawChar(x-8, y+r-12, 54, WHITE, BLACK, 2); // Print Character "6"(54)
223     // Draw Semi-Colons
224     tft.drawChar(45, X_COL_TIME, 58, WHITE, BLACK, SIZE_TIME); // Print Character ":"(58)
225     tft.drawChar(94, X_COL_TIME, 58, WHITE, BLACK, SIZE_TIME); // Print Character ":"(58)
226     // Draw Menu Box
227     tft.drawRect(MENU_BUTTON_X, MENU_BUTTON_Y, MENU_WIDTH, MENU_HEIGHT, WHITE);  // Draw
        Menu Box
228     tft.setTextColor(BLUE);
229     tft.setTextSize(2);
230     tft.setCursor(MENU_BUTTON_X + 8, MENU_BUTTON_Y + 14);
231     tft.print("SET");
232  }
233  // Function to Draw 2nd menu & Buttons
234  // ----------------------------------
235  void paintsecondmenu() { // 2nd menu
236    tft.fillScreen(BG_COLOR);
237    tft.setCursor(0, 20);   // Initial position
238    tft.setTextColor(WHITE); // TFT Color
239    tft.setTextSize(2); // Fontsize
240    tft.println("  Menu to Set Time");
241    tft.drawRect(TEXT_X, TEXT_Y, TEXT_W, TEXT_H, WHITE);
242    tft.drawChar(95, 60, 58, WHITE, BLACK, 3); // Print Character ":"(58)
243    tft.drawChar(145, 60, 58, WHITE, BLACK, 3); // Print Character ":"(58)
244    printDec2(60, 60, 3, RED, hour); // Goto Fuction
245    printDec2(110, 60, 3, YELLOW, minutes); // Goto Fuction
246    printDec2(160, 60, 3, GREEN, seconds); // Goto Fuction
247    // Create buttons for Display
248    for (uint8_t row=0; row<4; row++) { // 4 rows
249      for (uint8_t col=0; col<2; col++) { // 2 columns
250        buttons[col + row*2].initButton(&tft, BUTTON_X+col*(BUTTON_W+BUTTON_SPACING_X),
251                   BUTTON_Y+row*(BUTTON_H+BUTTON_SPACING_Y),    // x, y, w, h, outline,
                   fill, text, row * 2 columns
252                    BUTTON_W, BUTTON_H, WHITE, buttoncolors[col+row*2], WHITE,
253                    buttonlabels[col + row*2], BUTTON_TEXTSIZE);
254        buttons[col + row*2].drawButton();
255      }
256    }
257  }
258  // Function to Read Touch Screen
259  // -----------------------------
260  void readTouchScreen(){
261      if (menu_flag != 1){
262        TSPoint p = ts.getPoint();
263        // if sharing pins, you'll need to fix the directions of the touchscreen pins
264        //pinMode(XP, OUTPUT);
265        pinMode(XM, OUTPUT);
266        pinMode(YP, OUTPUT);
267        //pinMode(YM, OUTPUT);
268        // Get measured values
269        if (p.z > MINPRESSURE && p.z < MAXPRESSURE) { // scale from 0->1023 to tft.width
270           p.x = map(p.x, TS_MINX, TS_MAXX, tft.width(), 0);
271           p.y = (tft.height()-map(p.y, TS_MINY, TS_MAXY, tft.height(), 0));
272           Serial.print("p.x="); Serial.println(p.x);
273           Serial.print("p.y="); Serial.println(p.y);
```

```
          pos_x = p.x; // Return global variable for "X" position
          pos_y = p.y; // Return global variable for "Y" position
        }
     }
  }
  // Function to Format Data
  // -----------------------
  void printDec2(int x, int y, int textSize, int color, int value){
     tft.setCursor(x, y);
     tft.setTextColor(color);
     tft.setTextSize(textSize);
     if(value<10){
       tft.print('0');
     }
     tft.print(value);
  }
  // Function to Draw Clock Segments
  // -------------------------------
  void drawSegment(int x, int y, int r1, int r2, float a, int col){
    a = (a * DegToRad) - 1.57; // 1.57 = 90 Degrees hand tilt
    float a1 = a-1.57,
      a2 = a+1.57,
      x1 = x + (cos(a1) * r1),
      y1 = y + (sin(a1) * r1),
      x2 = x + (cos(a2) * r1),
      y2 = y + (sin(a2) * r1),
      x3 = x + (cos(a) * r2),
      y3 = y + (sin(a) * r2);
    tft.fillTriangle(x1,y1,x2,y2,x3,y3,col);
  }
  // Function to Erase Time
  // ----------------------
  void eraseTime(){
    if(sx+sy>0){
       // erase previous line
      tft.drawLine(x, y, sx, sy, BG_COLOR);
      //drawSegment(x, y, 6, sr, 6*secondsOld, BG_COLOR);
     if (incHour){
         printDec2(H_HOUR,X_COL_TIME,SIZE_TIME,BG_COLOR,hourOld); // Goto Function to
         update digital display
         incHour=false;
       }
     if(incMinutes){
        //tft.drawLine(x, y, mx, my, BG_COLOR);
        drawSegment(x, y, 6, mr, 6*minutesOld, BG_COLOR); // Update minutes function
        drawSegment(x, y, 8, hr, ((30*hourOld) + (minutesOld*30/60)), BG_COLOR); //
        Update hours function
        printDec2(H_MIN,X_COL_TIME,SIZE_TIME,BG_COLOR,minutesOld); // Goto Function
        update every minute for digital display
        incMinutes=false;
      }
      printDec2(H_SEC,X_COL_TIME,SIZE_TIME,BG_COLOR,secondsOld); // Goto Function
    }
  }
  // Function to Draw Clock Time
  // ---------------------------
  void drawTime(){
    // Draw Analog Time
    drawSegment(x, y, 8, hr, ((30*hour) + (minutes*30/60)), HOUR_COL); // 30 = (360
    degrees / 12) for hours
    drawSegment(x, y, 6, mr, 6*minutes, MINUTE_COL); // 6 = (360 degrees / 60) for minutes
    // Next Three lines draw the analog second hand
    a = ((6 * seconds) * DegToRad) - 1.570; // 1.57 = 90 Degrees
    sx = x + (cos(a) * sr);
    sy = y + (sin(a) * sr);
    tft.drawLine(x, y, sx, sy, GREEN);
    // Draw Digital Time
    printDec2(H_HOUR,X_COL_TIME,SIZE_TIME,HOUR_COL,hour);
    printDec2(H_MIN,X_COL_TIME,SIZE_TIME,MINUTE_COL,minutes);
```

```
    printDec2(H_SEC,X_COL_TIME,SIZE_TIME,GREEN,seconds);
  }
  // Function to Increment Time Ticks
  // --------------------------------
  void incTime(){
    secondsOld = seconds;
    minutesOld = minutes;
    hourOld = hour;
    if(seconds>=59){
      seconds = 0;
      incMinutes=true;
      if(minutes>=59){
        minutes = 0;
        incHour=true;
        if(hour>=12){
          hour = 1;
        }else{
          hour++;
        }
      }else{
        minutes++;
      }
    }else{
      seconds++;
    }
  }
  // Read Sensor Function
  // --------------------
  void SensorData(){
    Hum = dht.readHumidity();  // Measure the humidity
    TemperatureC = dht.readTemperature(); // Measure the temperature
    TempF = ((TemperatureC * 9/5) + 32); // Convert temperature to degrees
    Fahrenheit
     // Compare temperature & humidity events and perform a check sum.
     if (isnan(TemperatureC) || isnan(Hum)){ // Print "0" for a bad reading
      TempF = 0;
      Hum = 0;
     }
    tft.setTextSize(2); // Fontsize
    tft.setCursor(5, 45);   // Initial position
    tft.fillRect(170, 45, 70, 16, BLACK); // Erase previous numbers (x,y,W,H,color)
    tft.setTextColor(MAGENTA); // TFT Color
    Serial.println("Temperature = " + String(TempF));
    tft.println("Temperature = " + String(TempF)); //Text to Display
    tft.setCursor(5, 65);   // Initial position
    tft.fillRect(132, 65, 80, 16, BLACK); // Erase previous numbers (x,y,W,H,color)
    tft.setTextColor(MAGENTA); // TFT Color
    Serial.println("Humidity = " + String(Hum));
    tft.println("Humidity = " + String(Hum) + "%"); //Text to Display
  }
  // Main Program Loop
  // -----------------
  void loop() {
    if ((INT_flag == 1) && (menu_flag == 0)){ // Check for 1 second interrupt
      INT_flag = 0; // Set interrupt flag to "0"
      eraseTime(); // Goto Function
      drawTime();  // Goto Function
      incTime(); // Goto Function
    }
    if ((pos_x > 177 && pos_x < 240) && (pos_y > 1 && pos_y < 46) && (menu_flag == 0)) {
      pos_x = -1; // Default return position
      pos_y = -1; // Default return position
      menu_flag = 1;  // Set menu flag to "1"
      tft.fillScreen(BLACK); // Erase Screen
      paintsecondmenu(); // Goto second menu Function
      do { // Locks the menu in time set mode
        TSPoint p = ts.getPoint();  // Get X&Y values from touch screen
        Serial.println("Point is: ");
        Serial.print("X = "); Serial.print(p.x);
```

```
407         Serial.print("Y = "); Serial.print(p.y);
408         Serial.print("Pressure = "); Serial.println(p.z);
409
410         pinMode(XM, OUTPUT);  // Set the pin as an output
411         pinMode(YP, OUTPUT);  // Set the pin as an output
412
413         if (p.z > MINPRESSURE && p.z < MAXPRESSURE) { // scale from 0->1023 to
            tft.width
414           p.x = map(p.x, TS_MINX, TS_MAXX, tft.width(), 0);
415           p.y = (tft.height()-map(p.y, TS_MINY, TS_MAXY, tft.height(), 0));
416          }
417
418         for (uint8_t b=0; b<8; b++) { // go thru all the buttons, checking if pressed
419         if (buttons[b].contains(p.x, p.y)) {
420         Serial.print("Pressing: "); Serial.println(b);
421         buttons[b].press(true);  // tell the button it is pressed
422         } else {
423         buttons[b].press(false);  // tell the button it is NOT pressed
424         }
425        }
426        // now we can ask the buttons if their state has changed
427        for (uint8_t b=0; b<8; b++) {
428          if (buttons[b].justReleased()) {
429           Serial.print("Released: "); Serial.println(b);
430           buttons[b].drawButton();  // draw normal buttons (no change)
431          }
432           if (buttons[b].justPressed()) {
433             buttons[b].drawButton(true);  // draw invert to show button change
434             switch (b) {  // Perform actions when button is pressed
435       case 0:
436          if (hour < 12) {  // Do not go past 12
437           tft.fillRect(60, 58, 40, 28, BLACK);  // This clears last position by turning
              the section "black"
438           hour++; // Increase hours
439           printDec2(60, 60, 3, RED, hour); // Goto Fuction
440          }
441         break;
442       case 1:
443         if (hour > 1) {  // Do not go past 1
444           tft.fillRect(60, 58, 40, 28, BLACK);  // This clears last position by turning
              the section "black"
445           hour--; // Decrease hours
446           printDec2(60, 60, 3, RED, hour); // Goto Fuction
447          }
448         break;
449       case 2:
450          if (minutes < 59) {  // Do not go past 59
451           tft.fillRect(110, 58, 40, 28, BLACK);  // This clears last position by turning
              the section "black"
452           minutes++; // Increase minutes
453           printDec2(110, 60, 3, YELLOW, minutes); // Goto Fuction
454          }
455         break;
456       case 3:
457          if (minutes > 0) {  // Do not go past 0
458           tft.fillRect(110, 58, 40, 28, BLACK);  // This clears last position by turning
              the section "black"
459           minutes--; // Decrease minutes
460           printDec2(110, 60, 3, YELLOW, minutes); // Goto Fuction
461          }
462         break;
463       case 4:
464          //printDec2(160, 60, 3, GREEN, seconds); // Goto Fuction
465          if (seconds < 59) {  // Do not go past 59
466           tft.fillRect(160, 58, 40, 28, BLACK);  // This clears last position by turning
              the section "black"
467           seconds++; // Increase seconds
468           printDec2(160, 60, 3, GREEN, seconds); // Goto Fuction
469          }
```

```
        break;
      case 5:
        if (seconds > 0) {  // Do not go past 0
          tft.fillRect(160, 58, 40, 28, BLACK);  // This clears last position by turning the section "black"
          seconds--; // Decrease seconds
          printDec2(160, 60, 3, GREEN, seconds); // Goto Fuction
         }
        break;
      case 6:
          seconds = 0;
          tft.fillRect(160, 58, 40, 28, BLACK);  // This clears last position by turning the section "black"
          printDec2(160, 60, 3, GREEN, seconds); // Goto Fuction
          hour = 10;
          tft.fillRect(60, 58, 40, 28, BLACK);  // This clears last position by turning the section "black"
          printDec2(60, 60, 3, RED, hour); // Goto Fuction
          minutes = 30;
          tft.fillRect(110, 58, 40, 28, BLACK);  // This clears last position by turning the section "black"
          printDec2(110, 60, 3, YELLOW, minutes); // Goto Fuction
        break;
      case 7:
        menu_flag = 0; // Reset menu flag to display clock screen
        paintClockScreen(); // Goto function
        SensorData(); // Goto Function
        EraseFlag == true; // Set Erase Flag
        break;
      default:
        break;
      delay(100); // button debouncing
        }
       }
     }
    } while (menu_flag != 0); // Do routine while not in mian menu
   }
    if ((millis() - time_now1 > period1) && (menu_flag == 0)){  // If time passed and in main menu
      SensorData(); // Goto Function
      time_now1 = millis();
    }
    if (millis() - time_now2 > period2){  // Enable touchscreen every 200mS to set time
      readTouchScreen(); // Goto Function
      time_now2 = millis();
    }
  }
  // Interrupt Setup Function
  // ------------------------
  void INT_Init() {  // Set up Timer Interrupt for 1S
    cli();          // Disable global interrupts
    TCCR1A = 0;     // Set entire TCCR1A register to 0
    TCCR1B = 0;     // Same for TCCR1B
    TCNT1 = 0;      // Set counter variable to 0
    // set compare match register to set sample time is
    // Note - change OCR1A if time is Fast / Slow (1S = 15624) if time is 1.5 min slow per day;
    // 1.5 min = 90 seconds,so error is 90 sec/day, and there are 86,400 sec/day. of slow error is (-)
    // -error = (90/86400)*100 = 0.104%
    // New (OCR1A) = [15624 - (15624*(0.104/100)) = 15607
    OCR1A = 15607;  // preset = [16E6 / (prescale * (Hz))] - 1
    TCCR1B |= (1 << WGM12);  // turn on CTC mode
    TCCR1B |= (1 << CS12) | (1 << CS10) ; // Set CS12 and CS10 bits for prescaling by 1024
    TIMSK1 |= (1 << OCIE1A); // enable timer compare interrupt
    sei();          // enable global interrupts
  }
  // Timer1 Interrupt Routine
```

```
// ------------------------
ISR(TIMER1_COMPA_vect) { // The ISR will be called every 1 second to keep accurate time
  digitalWrite(13, !digitalRead(13));
  INT_flag = 1; // Set interrupt flag to "1"
  count++;  // Increase counter by one
  if(count == 300) {  // 300 counts is 5 minutes
    // put temp sense function here or use millis????
    count = 0; // Reset counting variable
  }
}
```