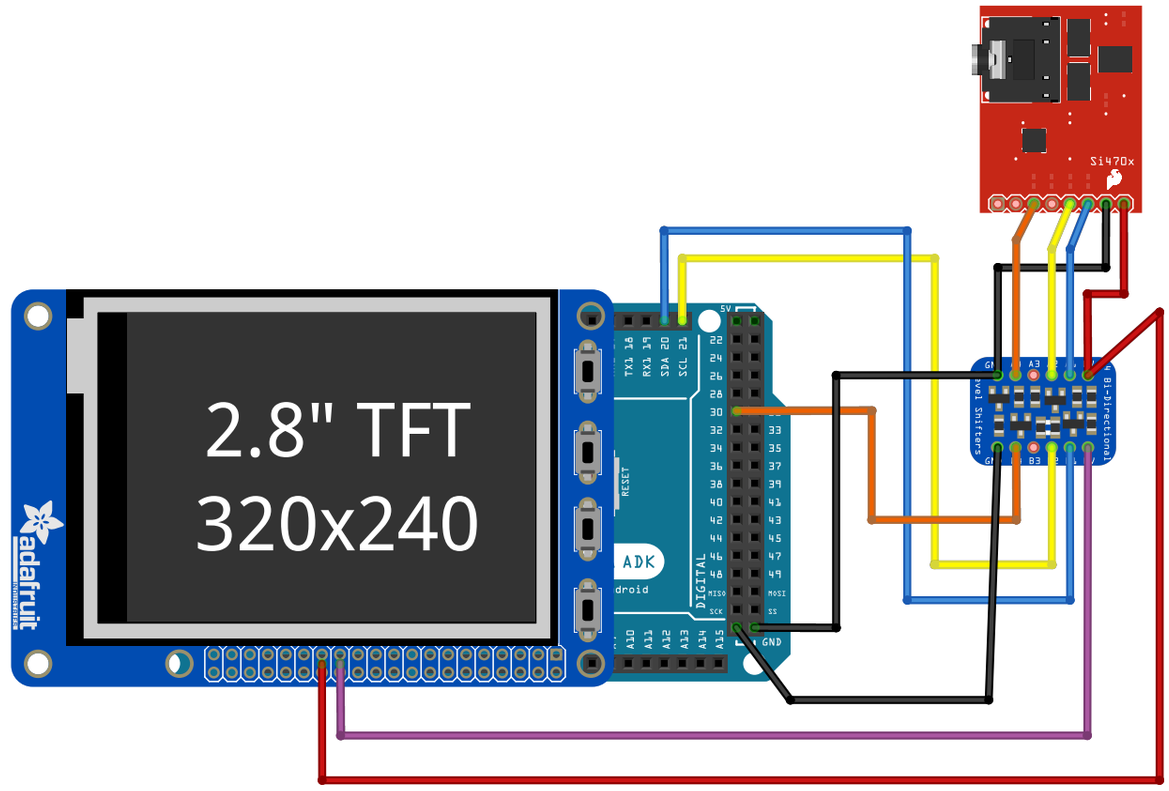
Digital FM Radio With RDS Message Display, Stereo Indication, and Received Signal Strength.

Also Has Touch Screen Operation For Volume and Channel Seek



Hiletgo Si4703 Board
- Red Wire = 3.3V
- Black Wire = Gnd
- Orange Wire = Reset
- Yellow Wire = SCLK
- Blue Wire = SDIO

5 Volt to 3.3 Volt
Logic Converter:

Low Side is 3.3V I/O described above:

High Side is 5 I/O described below:
- Purple Wire = 5V
- Black Wire = Gnd
- Orange Wire = Digital (30)
- Yellow Wire = SCL
- Blue Wire = SDA

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.

```
/*
 This is a full function radio implementation that uses a TFT display to show the current station information
 and status  It can be used with various chips after adjusting the radio object definition.
 You can also open the Serial console with 57600 baud to see current radio information and change various settings.

 Wiring:
 ------
 Arduino port | SI4703 signal | RDA5807M signal
 :----------: | :-----------: | :-------------:
 GND (black)  | GND           | GND
 3.3V (red)   | VCC           | VCC
 5V (red)     | -             | -
 21 (yellow)  | SCLK          | SCLK
 20 (blue)    | SDIO          | SDIO
 30           | RST           | -

 Notes:
 -----
 1) The original library was  modified to use an Arduino Mega (changed reset and SDA to above
 2) The TFT screen was from Amazon Elegoo EL-SM-004 R3 2.8 Inches TFT Touch Screen
 3) A logic level converter (5V-3.3v) is required to interface Arduino Mega to Si4703 for (SCL,SCA,RST)
 */
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_TFTLCD.h> // Hardware-specific library
#include <TouchScreen.h>     // Touchscreen library
#include <Wire.h>
#include <radio.h>
#include <RDA5807M.h>
#include <SI4703.h>
#include <SI4705.h>
#include <TEA5767.h>
#include <RDSParser.h>

// 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.
#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

// 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

```

```
65  /****************** UI details */
66  #define BUTTON_X 70
67  #define BUTTON_Y 200
68  #define BUTTON_W 50
69  #define BUTTON_H 30
70  #define BUTTON_SPACING_X 50
71  #define BUTTON_SPACING_Y 50
72  #define BUTTON_TEXTSIZE 2
73
74  // Text box where the Radio Status Goes
75  #define TEXT_X 10
76  #define TEXT_Y 35 //50
77  #define TEXT_W 220
78  #define TEXT_H 105 // 90, 110 was last
79
80  #define YP A3  // must be an analog pin, use "An" notation!
81  #define XM A2  // must be an analog pin, use "An" notation!
82  #define YM 9   // can be a digital pin
83  #define XP 8   // can be a digital pin
84
85  //Touch Screen Min & Max Parameters For New ILI9341 TP
86  #define TS_MINX 120
87  #define TS_MAXX 900
88
89  #define TS_MINY 70
90  #define TS_MAXY 920
91
92  //#define BUT1 39 // Arduino Output-1 Pin
93  //#define BUT2 45  // Arduino Outout-2 Pin
94
95  // Define some stations available at your locations here:
96  // 89.40 MHz as 8940
97  RADIO_FREQ preset[] = {
98    9330,
99    9450,
100   9510,
101   9790,
102   10070
103 };
104
105 int    i_sidx = 2;        ///< Start at Station with index = 2
106
107 // The radio object has to be defined by using the class corresponding to the used chip.
108 // by uncommenting the right radio object definition.
109
110 // RADIO radio;    // Create an instance of a non functional radio.
111 // RDA5807M radio;    // Create an instance of a RDA5807 chip radio
112 SI4703   radio;    // Create an instance of a SI4703 chip radio.
113 // TEA5767  radio;    // Create an instance of a TEA5767 chip radio.
114
115
116 // Get a RDS parser
117 // ----------------
118 RDSParser rds;
119
120 /// State definition for this radio implementation.
121 enum RADIO_STATE {
122   STATE_PARSECOMMAND, ///< waiting for a new command character.
123   STATE_PARSEINT,     ///< waiting for digits for the parameter.
124   STATE_EXEC,          ///< executing the command.
125   //STATE_FREQ,
126   STATE_RDS,         // New, Roy added
127 };
128
129 RADIO_STATE state; ///< The state variable is used for parsing input characters.
130 //RADIO_STATE rot_state; // Not needed, for an encoder
131
132 Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
133 TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);
```

```
134
135  Adafruit_GFX_Button buttons[4];
136  /* create 4 buttons, in classic candybar phone style */
137  char buttonlabels[4][4] = {"UP", "DWN", "UP", "DWN"};
138  uint16_t buttoncolors[4] = {GREEN, RED, GREEN, RED};
139  int v = 0;  // Global Initial Volume level
140  int newv = 0; // Global stored volume level
141  int fr; // Global Radio Frequency
142  int newrsi = 0;  // Global RSSI stored value
143  bool newster = 0; // Global Stereo stored value
144  bool newrd = 0; // Global RDS stored value
145  char rdsServiceName[8+2]; // String with actual RDS Service Name
146  char rdsText[64+2]; // String with actual RDS Text
147  char rdsTime[6]; // String with actual RDS Time
148
149
150  // Setup Program Stuff:
151  // ====================
152  void setup() {
153    Serial.begin(57600);
154    Serial.println("TFT Display Test...");
155    delay(500); // 0.5 second delay
156    //pinMode(BUT1, OUTPUT);  // Set the pin as an output
157    //pinMode(BUT2, OUTPUT);  // Set the pin as an output
158    //digitalWrite(BUT1, HIGH); // Turn off filterpump (negative logic)
159    //digitalWrite(BUT2, HIGH); // Turn off filterpump (negative logic)
160
161   // TFT Setup Stuff:
162   // ----------------
163  #ifdef USE_ADAFRUIT_SHIELD_PINOUT
164    Serial.println(F("Using Adafruit 2.8\" TFT Arduino Shield Pinout"));
165  #else
166    Serial.println(F("Using Adafruit 2.8\" TFT Breakout Board Pinout"));
167  #endif
168
169    tft.reset(); // Reset TFT Display
170    tft.begin(0x9341); // Start TFT Display
171    tft.setRotation(2);  // Rotate screen 90 degrees
172    tft.fillScreen(BLACK); // TFT background color
173    tft.setCursor(0, 0);   // Initial position
174    tft.setTextColor(RED); // TFT Color
175    tft.setTextSize(2); // Fontsize
176    tft.println("Digital FM Radio"); //Text to Display
177    tft.println(); // Space
178    tft.println(); // Space
179    tft.setTextColor(GREEN);  // TFT Color
180    tft.setTextSize(3); // Font size
181    tft.println("Si4703"); //Text to Display
182    tft.println(); // Space
183    tft.println("  RDS Display"); //Text to Display
184    tft.println(); // Space
185    tft.println(); // Space
186    tft.setTextColor(BLUE); // TFT Color
187    tft.setTextSize(2); // Fontsize
188    tft.println("By Roy H. Guerra Jr."); //Text to Display
189    tft.println(); // Space
190    tft.println(); // Space
191    delay(4000); // 4 second delay
192    tft.fillScreen(BLACK);  // Clear to a new screen
193    tft.setCursor(2, 5); // Start in home position
194    tft.setTextColor(WHITE);
195    tft.setTextSize(3);
196    tft.println("Digital Radio");
197
198    tft.drawRect(TEXT_X, TEXT_Y, TEXT_W, TEXT_H, WHITE);  // create a blank 'text field'
199
200    tft.setCursor(50, 160); // Start in this position
201    tft.setTextColor(WHITE);
202    tft.setTextSize(2);
```

```
203     tft.println("Station Seek");
204     tft.setCursor(50, 240); // Start in this position
205     tft.setTextColor(WHITE);
206     tft.setTextSize(2);
207     tft.println("Radio Volume");
208
209     for (uint8_t row=0; row<2; row++) { // create display buttons
210       for (uint8_t col=0; col<2; col++) {
211         buttons[col + row*2].initButton(&tft, BUTTON_X+col*(BUTTON_W+BUTTON_SPACING_X),
212                     BUTTON_Y+row*(BUTTON_H+BUTTON_SPACING_Y),    // x, y, w, h, outline,
                    fill, text
213                      BUTTON_W, BUTTON_H, WHITE, buttoncolors[col+row*2], WHITE,
214                      buttonlabels[col + row*2], BUTTON_TEXTSIZE);
215         buttons[col + row*2].drawButton();
216       }
217     }
218      // Radio Setup Stuff
219      // -----------------
220      radio.init(); // Initialize the Radio
221      radio.debugEnable(); // Enable information to the Serial port
222      radio.setBandFrequency(RADIO_BAND_FM, preset[i_sidx]); // Preset's.
223      //radio.setFrequency(979);  // Initial Station
224      radio.setVolume(v);  // Set to Initial Value "0"
225      radio.setMono(false);  // Keep in Stereo Mode
226      radio.setMute(false);  // Turn off Mute
227      Serial.write('>'); // Write command prompt
228      // radio.debugRegisters();  // Turn on to debug
229      state = STATE_PARSECOMMAND;
230      radio.attachReceiveRDS(RDS_process); // setup the information chain for RDS data.
231      rds.attachServicenNameCallback(DisplayServiceName);  // Callback to the selected
         function
232      rds.attachTimeCallback(DisplayTime); // Callback to the selected function
233      rds.attachTextCallback(DisplayText); // Callback to the selected function
234      runSerialCommand('?', 0);
235      v = radio.getVolume();  // Get Radio Volume
236      DisplayVolume(v); // Goto Volume Function
237   }
238
239   #define MINPRESSURE 10  // Set touch screen limits
240   #define MAXPRESSURE 1000
241
242   // Update the Frequency on the TFT display.
243   // ----------------------------------------
244   void DisplayFrequency(RADIO_FREQ f){
245     char s[12];
246     radio.formatFrequency(s, sizeof(s));
247     Serial.print("FREQ:"); Serial.println(s);
248     tft.setCursor(20, 42); // Start in this position
249     tft.fillRect(20, 42, 205, 15, BLACK);  // This clears last position by turning the
        line "black"
250     tft.setTextColor(YELLOW);
251     tft.setTextSize(2);
252     tft.println("Freq: " + String(s));
253   }
254   // Update the ServiceName text on the LCD display when in RDS mode.
255   // ----------------------------------------------------------------
256   void DisplayServiceName(char *name){
257     Serial.print("RDS:");
258     Serial.println(name);
259      //size_t len = strlen(name);
260     //lcd.setCursor(0, 1);
261     //lcd.print(name);
262     //while (len < 8) {
263       //lcd.print(' ');
264      // len++;
265     //}
266   }
267   // Update RDS Display Time:
268   // ------------------------
```

```
269   void DisplayTime(uint8_t hour, uint8_t minute) {
270     Serial.print("RDS-Time:");
271     if (hour < 10) Serial.print('0');
272     Serial.print(hour);
273     Serial.print(':');
274     if (minute < 10) Serial.print('0');
275     Serial.print(minute);
276   }
277   // Display the Current Volume:
278   // ---------------------------
279   void DisplayVolume(int v){
280     Serial.print("VOL: "); Serial.println(v);
281     tft.setCursor(20, 62); // Start in this position
282     tft.fillRect(20, 62, 205, 15, BLACK); // This clears last position by turning the
      line "black"
283     tft.setTextColor(BLUE);
284     tft.setTextSize(2);
285     tft.println("Volume: " + String(v) + " of 15");
286   }
287   // Update the RDS Display Text:
288   // ----------------------------
289   void DisplayText(char *name){ // RDS Info that Scrolls
290     Serial.print("RDS:");
291     Serial.println(name);
292     tft.setTextWrap(false);  // Do not allow test to wrap
293     String text = String(name); // RDS Callback Text as a string
294     const int width = 32;  // width of the marquee display (in characters)
295     for (int offset = 0; offset < text.length(); offset++) { //Index the entire string
296       String t ="";  // Null
297       for (int i = 0; i < width; i++) // Condense string to width
298       //for (int i = 0; i < text.length(); i++)
299       t += text.charAt((offset + i) % text.length()); // Print text from (Offfset +
        Index)
300       tft.setCursor(28, 122);   // Initial position (45)
301       tft.setTextColor(CYAN); // TFT Color
302       tft.setTextSize(1); // Fontsize
303       tft.print(t); // Print
304       delay(200); // Delay Scroll as to not move fast
305       tft.fillRect(20, 122, 205, 15, BLACK); // This clears last text
306     }
307   }
308   // Display the Audio Parameters:
309   // -----------------------------
310   void Radio_Param(int rsi, bool ster, bool rd){
311     if ((rsi != newrsi)||(ster != newster)||(rd != newrd)){
312        String a = "  ";
313        String b = "  ";
314        tft.setCursor(20, 82); // This is position
315        tft.fillRect(20, 82, 205, 15, BLACK); // This clears last position by turning the
        line "black"
316        tft.setTextColor(ORANGE);
317        tft.setTextSize(2);
318        tft.println("RSSI: " + String(rsi) + " dBuV");
319
320       if (ster == 1) {
321         a = "Stereo";
322       } else {
323         a = "Mono";
324       }
325       if (rd == 1) {
326         b = "Off";
327       } else {
328         b = "On";
329       }
330     tft.setCursor(20, 102); // This is text
331     tft.fillRect(20, 102, 205, 15, BLACK); // This clears last position by turning the
      line "black"
332     tft.setTextColor(PINK);
333     tft.setTextSize(2);
```

```
    tft.println("RDS: " + b + "    " + a);

    newrsi = rsi;
    newster = ster;
    newrd = rd;
    }
  }
  // Display the Mono Switch:
  // ------------------------
  void DisplayMono(uint8_t y){
    Serial.print("MONO: "); Serial.println(y);
    //lcd.setCursor(0, 1);
    //lcd.print("MONO: "); lcd.print(y);
  }
  // Display the Soft Mute Switch:
  // -----------------------------
  void DisplaySoftMute(uint8_t z){
    Serial.print("SMUTE: "); Serial.println(z);
    //lcd.setCursor(0, 1);
    //lcd.print("SMUTE: "); lcd.print(z);
  }
  // RDS Signal Information:
  // -----------------------
  void RDS_process(uint16_t block1, uint16_t block2, uint16_t block3, uint16_t block4) {
    rds.processData(block1, block2, block3, block4);
  }
  // Execute a command identified by a character and an optional number.
  // See the "?" command for available commands.
  // \param cmd The command character.  \param value An optional parameter for the command.
  // ---------------------------------------------------------------------------------------
  void runSerialCommand(char cmd, int16_t value){
    if (cmd == '?') {
      Serial.println();
      Serial.println("? Help");
      Serial.println("+ increase volume");
      Serial.println("- decrease volume");
      Serial.println("> next preset");
      Serial.println("< previous preset");
      Serial.println(". scan up   : scan up to next sender");
      Serial.println(", scan down ; scan down to next sender");
      Serial.println("fnnnnn: direct frequency input");
      Serial.println("i station status");
      Serial.println("s mono/stereo mode");
      Serial.println("b bass boost");
      Serial.println("m mute/unmute");
      Serial.println("u soft mute/unmute");
    }
    // ----- control the volume and audio output -----
    else if (cmd == '+') {
      // increase volume
      int v = radio.getVolume();
      if (v < 15) radio.setVolume(++v);
    }
    else if (cmd == '-') {
      // decrease volume
      int v = radio.getVolume();
      if (v > 0) radio.setVolume(--v);
    }
    else if (cmd == 'm') {
      // toggle mute mode
      radio.setMute(! radio.getMute());
    }
    else if (cmd == 'u') {
      // toggle soft mute mode
      radio.setSoftMute(! radio.getSoftMute());
    }
    // toggle stereo mode
    else if (cmd == 's') { radio.setMono(! radio.getMono()); }
    // toggle bass boost
```

```
403     else if (cmd == 'b') { radio.setBassBoost(! radio.getBassBoost()); }
404     // ----- control the frequency -----
405     else if (cmd == '>') {
406       // next preset
407       if (i_sidx < (sizeof(preset) / sizeof(RADIO_FREQ))-1) {
408         i_sidx++; radio.setFrequency(preset[i_sidx]);
409       } // if
410     }
411     else if (cmd == '<') {
412       // previous preset
413       if (i_sidx > 0) {
414         i_sidx--;
415         radio.setFrequency(preset[i_sidx]);
416       } // if
417     }
418     else if (cmd == 'f') { radio.setFrequency(value); }
419     else if (cmd == '.') { radio.seekUp(false); }
420     else if (cmd == ':') { radio.seekUp(true); }
421     else if (cmd == ',') { radio.seekDown(false); }
422     else if (cmd == ';') { radio.seekDown(true); }
423     // not in help:
424     else if (cmd == '!') {
425       if (value == 0) radio.term();
426       if (value == 1) radio.init();
427     }
428     else if (cmd == 'i') {
429       char s[12];
430       radio.formatFrequency(s, sizeof(s));
431       Serial.print("Station:"); Serial.println(s);
432       Serial.print("Radio:"); radio.debugRadioInfo();
433       Serial.print("Audio:"); radio.debugAudioInfo();
434       //Serial.print("  RSSI: ");
435       //Serial.print(info.rssi);
436   //
437   //     for (uint8_t i = 0; i < info.rssi - 15; i+=2) { Serial.write('*'); } //
      Empfangspegel ab 15. Zeichen
438   //     Serial.println();
439     } // info
440   //  else if (cmd == 'n') { radio.debugScan(); }
441     else if (cmd == 'x') { radio.debugStatus(); }
442   }
443
444   // Main Program Stuff:
445   // ===================
446   void loop() {
447     // Radio Stuff:
448     // ------------
449     int newPos;
450     unsigned long now = millis();
451     static unsigned long nextFreqTime = 0;
452     static unsigned long nextRadioInfoTime = 0;
453     // some internal static values for parsing the input
454     static char command;
455     static int16_t value;
456     static RADIO_FREQ lastf = 0;
457     RADIO_FREQ f = 0;
458     char c;
459
460    if (v != newv){
461       v = radio.getVolume();
462       DisplayVolume(v); // Goto Function
463       newv = v;
464    }
465     // Serial Commands:
466     if (Serial.available() > 0) {
467       // read the next char from input.
468       c = Serial.peek();
469       if ((state == STATE_PARSECOMMAND) && (c < 0x20)) {
470         // ignore unprintable chars
```

```
        Serial.read();
      }
      else if (state == STATE_PARSECOMMAND) {
        // read a command.
        command = Serial.read();
        state = STATE_PARSEINT;
      }
      else if (state == STATE_PARSEINT) {
        if ((c >= '0') && (c <= '9')) {
          // build up the value.
          c = Serial.read();
          value = (value * 10) + (c - '0');
        }
        else {
          // not a value -> execute
          runSerialCommand(command, value);
          command = ' ';
          state = STATE_PARSECOMMAND;
          value = 0;
        }
      }
    }
    radio.checkRDS();  // check for RDS data
    // update the display from time to time
    if (now > nextFreqTime) {
      //DisplayText(char *name);  // Goto Function
      f = radio.getFrequency();
      if (f != lastf) {
        // don't display a Service Name while frequency is not stable.
        //DisplayServiceName("        ");
        DisplayFrequency(f);
        lastf = f;
      }
      nextFreqTime = now + 400;
    }
    if (now > nextRadioInfoTime) {
      RADIO_INFO info;  // Read Structure tied to "info"
      radio.getRadioInfo(&info); // Get parameters below
      Radio_Param(info.rssi, info.stereo, info.rds);  // Send to Function
      //DisplayText();  // Goto Function
      nextRadioInfoTime = now + 2000;
    }


    TSPoint p = ts.getPoint();  // Get X&Y values from touch screen, used for button
    calibration
    //Serial.println("Point is: ");
    //Serial.print("X = "); Serial.print(p.x);
    //Serial.print("/tY = "); Serial.print(p.y);
    //Serial.print("\tPressure = "); Serial.println(p.z);

    pinMode(XM, OUTPUT);  // Set the pin as an output
    pinMode(YP, OUTPUT);  // Set the pin as an output

    if (p.z > MINPRESSURE && p.z < MAXPRESSURE) { // scale from 0->1023 to tft.width
      p.x = map(p.x, TS_MINX, TS_MAXX, tft.width(), 0);
      p.y = (tft.height()-map(p.y, TS_MINY, TS_MAXY, tft.height(), 0));
     }

    for (uint8_t b=0; b<4; b++) { // Go thru all the buttons, checking if they were pressed
      if (buttons[b].contains(p.x, p.y)) {
        //Serial.print("Pressing: "); Serial.println(b);
        buttons[b].press(true);  // Tell the button it is pressed
      } else {
        buttons[b].press(false);  // Tell the button it is NOT pressed
      }
    }

    for (uint8_t b=0; b<4; b++) { // See if the buttons state has changed
```

```
539       if (buttons[b].justReleased()) {
540         //Serial.print("Released: "); Serial.println(b);
541         buttons[b].drawButton();  // draw normal
542       }
543 
544       if (buttons[b].justPressed()) {
545           buttons[b].drawButton(true);  // Draw invert so we can tell if button was
              pressed
546 
547       switch (b) {  // Perform actions when button is pressed
548       case 0:  // Seek Up
549          radio.seekUp(true); // Set Logic
550          fr = radio.getFrequency();  // Get new frequency
551          DisplayFrequency(fr); // Goto frequency display function
552         break;
553       case 1:  // Seek Down
554          radio.seekDown(true); // Set Logic
555          //radio.seekUp(false); // Set Logic
556          fr = radio.getFrequency();  // Get new frequency
557          DisplayFrequency(fr); // Goto frequency display function
558         break;
559       case 2:  // Increase Volume
560          v = radio.getVolume();
561          if (v < 15) radio.setVolume(++v);
562          DisplayVolume(v); // Goto Function
563         break;
564       case 3: // Decrease Volume
565          v = radio.getVolume();
566          if (v > 0) radio.setVolume(--v);
567          DisplayVolume(v); // Goto Function
568         break;
569       default:
570         break;
571     }
572     delay(100);  // button debounce
573       }
574     }
575   }
```