```
// More information of this library on : http://www.RinkyDinkElectronics.com/
#include <UTFT.h>

// Declare which fonts we will be using
extern uint8_t SmallFont[];
extern uint8_t BigFont[];
extern uint8_t SevenSegNumFont[];
extern uint8_t UbuntuBold[];

// Define Input & ADC
#define PtPin 1       //Incident Power Port on analog A1
#define PrPin 0       //Reflected Power Port on analog A0
#define ADCref 2.56   //Voltage reference for ADC conversion at 2.56V

// Define RF Parameters
#define NoRF 0.18     //LT5581 produced 180mV when no signal on RF Input
#define Slope 0.031   //LT5581 produced 21mv per 1 dB
#define PtOffset 7.0  //Offset of Incident Power(dB)
#define PrOffset 7.0  //Offset of Reflected Power(dB)

// Assign all text positions
const unsigned int corner_line1 = 14;
const unsigned int corner_line2 = 111;
const unsigned int corner_line3 = 208;
const unsigned int corner_line4 = 305;
const unsigned int x_text1 = 10;
const unsigned int x_text2 = 250;
const unsigned int x_dbm = 185;
const unsigned int y_line1 = 47;
const unsigned int y_line2 = 144;
const unsigned int y_line3 = 241;

// Declare variables
double Pt, Pr, VSWR, S11, ReCof;
float PtDC, PrDC;

UTFT myGLCD(CTE32HR, 38, 39, 40, 41);

void setup()
{
  analogReference(INTERNAL2V56);  //Set Voltage reference for ADC conversion at 2.56V
  myGLCD.InitLCD();
  draw_layout();                  //Draw border, Rectangular Color Box
  draw_text();                    //Draw header text, footer text
}

void loop()
{
  PtDC = analogRead(PtPin);       //Read voltage from Incident Power Pin
  PrDC = analogRead(PrPin);       //Read voltage from Reflected Power Pin
  calculate();                    //Calculate for result value
  print_value();                  //Print value on TFT LCD
  delay(1000);                    //Do above every 1 second
}

// Calculate for all parameters
void calculate()
{
  PtDC = ((PtDC / 1023.0) * ADCref);  //Calculate Pt Pin Voltage
  PrDC = ((PrDC / 1023.0) * ADCref);  //Calculate Pr Pin Voltage

  Pt = ((PtDC - NoRF) / Slope) - PtOffset;  //Calculate Pt(dB)
  Pr = ((PrDC - NoRF)/ Slope) - PrOffset;   //Calculate Pr(dB)

  ReCof = sqrt(pow(10, Pr / 20) / pow(10, Pt / 20)); //Calculate Reflection Coeficient
  S11 = 20 * log10(ReCof);                           //Calculate S11
  VSWR = (1 + ReCof) / (1 - ReCof);                  //Calculate VSWR
}
```

```
// Print all value on TFT LCD
void print_value()
{
  if (VSWR > 99.99)
    VSWR = 99.99;
  myGLCD.setFont(UbuntuBold);                                  //Set Font
  myGLCD.setColor(255, 255, 255);
  myGLCD.setBackColor(0, 0, 0);

  myGLCD.printNumF(PtDC * 1000, 0, x_text1 + 40, y_line1);  //Print Pt(mV)
  myGLCD.printNumF(Pt, 2, x_text1 + 40, y_line2);           //Print Pt(dB)
  myGLCD.printNumF(S11, 2, x_text1 + 40, y_line3);          //Print S11(dB)

  myGLCD.printNumF(PrDC * 1000,  0, x_text2 + 40, y_line1); //Print Pr(mV)
  myGLCD.printNumF(Pr, 2, x_text2 + 60, y_line2);           //Print Pr(dB)
  myGLCD.printNumF(VSWR,  1, x_text2 + 40, y_line3);        //Print VSWR
}

void draw_text()
{
  myGLCD.setFont(BigFont);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.print("Pt(mV):", x_text1, corner_line1 + 10);
  myGLCD.print("Pt(dB):", x_text1, corner_line2 + 10);
  myGLCD.print("S11:", x_text1, corner_line3 + 10);

  myGLCD.print("Pr(mV):", x_text2, corner_line1 + 10);
  myGLCD.print("Pr(dB):", x_text2, corner_line2 + 10);
  myGLCD.print("VSWR:", x_text2, corner_line3 + 10);

  myGLCD.print("mV", x_dbm, corner_line2 - 25);
  myGLCD.print("dB", x_dbm, corner_line3 - 25);
  myGLCD.print("dB", x_dbm, corner_line4 - 25);
  myGLCD.print("mV", x_dbm + 255, corner_line2 - 25);
  myGLCD.print("dB", x_dbm + 255, corner_line3 - 25);
}

void draw_layout()
{
  myGLCD.clrScr();
  myGLCD.setColor(255, 0, 0);
  myGLCD.fillRect(0, 0, 479, 13);
  myGLCD.setColor(64, 64, 64);
  myGLCD.fillRect(0, 306, 479, 319);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setBackColor(255, 0, 0);

  myGLCD.setFont(SmallFont);
  myGLCD.print("5.8 GHz Digital SWR & Power Meter", CENTER, 1);
  myGLCD.setBackColor(64, 64, 64);
  myGLCD.setColor(255, 255, 0);
  myGLCD.print("Copyright 2016 Electronic Engineering#2 All rights reserved", CENTER, 307);

  myGLCD.setColor(  0,   0, 255);
  myGLCD.drawRect(  0,  14, 479, 305);
  myGLCD.drawLine(  0, 111, 479, 111);
  myGLCD.drawLine(  0, 208, 479, 208);
  myGLCD.drawLine(240,  14, 240, 305);
}

// Special Function,Can use when want to find a average of raw inputs
float average_input(char analogPin, unsigned char amount)
{
  unsigned int minValue, maxValue, temp, sum;
  minValue = 1023;
  maxValue = 0;
```

```
  sum = 0;
  for (unsigned char x = 0; x < amount ; x++)
  {
    temp = analogRead(analogPin);
    if (temp > maxValue)
      maxValue = temp;
    if (temp < minValue)
      minValue = temp;
    sum += temp;
  }
  sum -= (minValue + maxValue);
  return (sum * 1.0) / (amount * 1.0);
}
```