GENERATORE SWEEP RF 1 MHz 150 MHz PARTE SECONDA

Il SOTWARE : versione n° 38 GEN_RF_SWEEP_ESP32_TFT_V38_DEF.ino

Scritto in C++ e compilato tramite l' IDE di Arduino vers. 2.3.2

```
/*****************************************************************************
                                NOTE INIZIALI PER PROMEMORIA
VERSIONE  uguale alla ESP32_TFT_V37_band_HF_ADS1115   ma ordinata il 16/04/2025
ultima installata sulla scheda   : GEN_RF_SWEEP_ESP32_TFT_V38_DEF.ino
* Display TFT 2.8"
        #include <TFT_eSPI.h>              per la libreria graphics and font library for ST7735  driver chip
* Modificato nella libreria TFT_eSPI  il file : User_Setup.h in  User_Setup_2.h
        con i pin corretti e impostazione RGB
* #include <si5351.h>          //fonte : Etherkit   https://github.com/etherkit/Si5351
* PIN ESP32 per TFT
*   Vcc             +5
*   GND             GND
*   CS              PIN 15
*   RESET           EN/RESET
*   A0 DC           PIN   2
*   SDA             PIN   13
*   SCL SCK         PIN   14
*   LED             21
*  NEL file User_Setup.h  modificare come segue e salvare come User_Setup_2.h
*   #define TFT_MISO 19
*   #define TFT_MOSI 23 SDA
*   #define TFT_SCLK 18   SCL-SCK
*   #define TFT_CS    17                  // Chip select control pin
*   #define TFT_DC     2                  // Data Command control pin
*   #define TFT_RST    14                 // Reset pin (could connect to RST pin)
*   #define PIN_IN1 25              // CLK ENCODER 21 e 22 non sono validi perchè utilizzati come SDA e SCL
*   #define PIN_IN2 26              // DT ENCODER
*        per Si5351 includere "Wire.h"  e inserire nel setup   Wire.begin(21, 22);          // SCL 22   SDA 21
*  PER CANCELLARE TESTO   tft.fillRect(0, 0, 280, 15, TFT_BLACK);         x,y,larghezza,altezza,colore BLACK
*  MODIFICHE HARDWARE E SOFTWARE PER UTILIZZARE IL MODULO ADC ESTERNO A ADS1115  A 16 BIT
*****************************************************************************************
                                INIZIO SKETCH          LIBRERIE INCLUSE
*****************************************************************************************/
    #include <Preferences.h>
        Preferences preferences;                                    // CLASS NAME
    #include <XPT2046_Touchscreen.h>
        SPIClass touchscreenSPI = SPIClass(VSPI);                   // CLASS NAME
    #include <FS.h>
    #include <TFT_eSPI.h>                       // Graphics and font library for ST7735 driver chip
        TFT_eSPI tft = TFT_eSPI();              // CLASS NAME
    #include <SPI.h>
    #include "Wire.h"
    #include <si5351.h>                         //Etherkit   https://github.com/etherkit/Si5351Arduino
```

```
#include <Adafruit_ADS1X15.h>        // convertitore ADC con chip ADS1115
    Adafruit_ADS1115 ads;             // CLASS NAME
#include <ESP32Encoder.h>             // https://github.com/madhephaestus/ESP32Encoder.git
/*****************************************************************************************
                                    DEFINIZIONI DI VARIABILI E PIN
*****************************************************************************************/
#define XPT2046_IRQ 36                // T_IRQ       // Touchscreen pins
#define XPT2046_MOSI 32               // T_DIN
#define XPT2046_MISO 39               // T_OUT
#define XPT2046_CLK 25                // T_CLK
#define XPT2046_CS 33                 // T_CS
#define XPT2046_BL 23                 // T_BL
    XPT2046_Touchscreen touchscreen(XPT2046_CS, XPT2046_IRQ);  // pin CS e IRQ
#define SCREEN_WIDTH 320          // inizia con 0,0 in alto a sinistra
#define SCREEN_HEIGHT 240
#define FONT_SIZE 1
/*****************************************************************************************
            IMPOSTAZIONE DELLA DIMENSIONE E POSIZIONE DEI PULSANTI VIRTUALI
                    // Button position X – Y ,dimensione W larghezza , H altezza in pixel
*****************************************************************************************/
#define FRE_PIU_X 300         // BOTTONE FREQUENZA +
#define FRE_PIU_Y 30
#define FRE_PIU_W 20
#define FRE_PIU_H 20
#define FRE_MENO_X 300   // BOTTONE FREQUENZA -
#define FRE_MENO_Y 55
#define FRE_MENO_W 20
#define FRE_MENO_H 20
#define STEP_PIU_X 300        // BOTTONE STEP +
#define STEP_PIU_Y 80
#define STEP_PIU_W 20
#define STEP_PIU_H 20
#define STEP_MENO_X 300   // BOTTONE STEP -
#define STEP_MENO_Y 105
#define STEP_MENO_W 20
#define STEP_MENO_H 20
#define BAND_PIU_X 300    // BOTTONE BANDA +
#define BAND_PIU_Y 130
#define BAND_PIU_W 20
#define BAND_PIU_H 20
#define BAND_MENO_X 300    // BOTTONE BANDA -
#define BAND_MENO_Y 155
#define BAND_MENO_W 20
#define BAND_MENO_H 20
#define START_SWEEP_X 300    // BOTTONE SET SWEEP   GIALLO
#define START_SWEEP_Y 180
#define START_SWEEP_W 20
#define START_SWEEP_H 20
```

```
    #define STOP_SWEEP_X 300    // BOTTONE START SWEEP VERDE / ROSSO
    #define STOP_SWEEP_Y 205
    #define STOP_SWEEP_W 20
    #define STOP_SWEEP_H 20
    #define CLEAR_X 300          // BOTTONE CANCELLA GRAFICO
    #define CLEAR_Y 2
    #define CLEAR_W 20
    #define CLEAR_H 20
    #define CLMKR_X 275          // BOTTONE CANCELLA MARKER
    #define CLMKR_Y 2
    #define CLMKR_W 20
    #define CLMKR_H 20
    #define MARK1_X 30           // BOTTONE MARKER
    #define MARK1_Y 226
    #define MARK1_W 10
    #define MARK1_H 10
    #define CALI_X 250          // BOTTONE CALIBRAZIONE ROSSO/VERDE
    #define CALI_Y 2
    #define CALI_W 20
    #define CALI_H 20
/*******************************************************************************************
                                COLORI E IMPOSTAZIONI PER IL DISPLAY
*******************************************************************************************/
    #define TFT_CS  15           // pin connessione modulo TFT
    #define TFT_RST  -1          //
    #define TFT_DC   2           //   definito anche A0
    #define BLACK 0x0000         // definizione dei colori necessaria per questa libreria
    #define BLUE 0x001F
    #define RED 0xF800
    #define GREEN 0x07E0
    #define CYAN 0x07FF
    #define YELLOW 0xFFE0
    #define WHITE 0xFFFF
    #define DARKGREY 0x7BEF
    #define DARKCYAN 0x03EF
    #define TFT_MOSI 13          // NON UTILIZZATI Data out per la scheda SD  MOSI pin 23
    #define TFT_SCLK 14          // Clock out per la scheda SD   -- SCK pin 18
    #define TFT_MISO 12          // per scheda SD pin 19 gli altri sono in parallelo
 /*******************************************************************************************
                                IMPOSTAZIONE PER IL MODULO Si5351
*******************************************************************************************/
  #define XT_CAL_F 153423 //Si5351 calibration factor, adjust to get exatcly 10MHz.
  Si5351   si5351(0x60);  //si5351 I2C Address 0x60
  long cal = XT_CAL_F;
/*******************************************************************************************
                                IMPOSTAZIONE PER L'ENCODER ROTATIVO
*******************************************************************************************/
  #define CLK 16               // CLK ENCODER
  #define DT 17                // DT ENCODER
```

```
  ESP32Encoder encoder;      // CLASS NAME
  long oldPosition = 0;
  long newPosition = 0;
  String direzione;
/*********************************************************************************************
PIN INPUT ANALOGICO (non usato dopo installazione del modulo ADC più presiso a 16 bit)
*********************************************************************************************/
  const int pinVin = 35;              // analog input non usato disponibile
  const int pinSwcontinuo = 19;      // sweep continuo
  const int pinResSof = 18;           // ESPRestart
/*********************************************************************************************
                                          VARIABILI USATE
*********************************************************************************************/
    int sweepcontinuo = HIGH;
    int resetsoft = HIGH;
    long freq = 1000000;
    long interfreq = 10700000;
    long spanfreq;
    long fstep = 10;
    long fstepband;
    long startfreq;
    long stopfreq;
    long markf1;
    long curfreq;
    byte ones,tens,hundreds,thousands,tenthousands,hundredthousands,millions ;
    byte onefm,tenfm,hundredfm,thousandfm,tenthousandfm,hundredthousandfm,millionfm;
    byte onefst,tenfst,hundredfst,thousandfst,tenthousandfst,hundredthousandfst,millionfst;
    long count = 1000;
    long old_count;
    int stpcount = 1;
    bool sts = 0;
    int stp = 1;
    String band = "FREE";
    float VIN;                // VIN;
    float VIN_old;
    float VIN_max;
    float Vcal = 0.00;
    float Vout;
    int dB = 0;
    int dBmax = 0;
    int dBcalmax = 0;
    int delta = 0;
    int x,y;                   //   posizione x,y del puntatore sul display
    int xp,yp;
    int yb, prev_yb;
    int markerX = 0;
    int bordo = 15;          // spazio dal bordo 15
    int xb = bordo;
    int maxxb = 0;
```

```
    int maxyb = 0;
    int mkvert_Old = 40;
    int mkvert_Cur = 40;
    int mkCount = 1;
    int curyb = 0;
    int mark1 = 0;
    float mi_Freq[300];             // array con valori della frequenza per ogni punto misura di 245
    int   mi_Xb[300];               // array con valori della posizione x  per ogni punto misura di 245
    float mi_Vinmax[300];           // array con valori massimo misurato per ogni punto
    int mi_Yb[300];                 // array con valori della posizione y  per ogni punto misura di 245
    float cor_Misura[13];           // valori di calibrazione per ogni banda
    bool calibra = true ;           // calibrazione ON/OFF
    float elenFreq[]=
{0,1000000,3500000,7000000,10100000,14000000,18068000,21000000,24890000,28000000,50000000,88000000,144
000000};
/*****************************************************************************************
  LA PARTE SEGUENTE VIENE SEMPRE ESEGUITA ( UNA SOLA VOLTA) PER PRIMA DOPO L'ACCENSIONE
*****************************************************************************************/
void setup()
{
    Wire.begin(21, 22);              // SCL 22   SDA 21  per comunicazione I2c
    Serial.begin(115200);            // velocità comunicazione per monitor seriale
    preferences.begin("my-app", false);     // nome file per memoria interna
    ads.begin();                     // inizializza il modulo ADS
    encoder.attachHalfQuad ( DT, CLK );   // inizializza pin encoder
    encoder.setCount ( 0 );                // parametro per funzione encoder
    pinMode(pinSwcontinuo, INPUT);       // pin 19 interruttore swepp continuo/singolo
    pinMode(pinResSof, INPUT);           // pin 27 avvia reset software
    touchscreenSPI.begin(XPT2046_CLK, XPT2046_MISO, XPT2046_MOSI, XPT2046_CS);
                                         // avvia SPI per il touchscreen e inizializza il touchscreen
    touchscreen.begin(touchscreenSPI);
    touchscreen.setRotation(3);      // Set the Touchscreen rotation in landscape mode 3 per allineare con display
    tft.init();
    tft.setRotation(1);
    tft.fillScreen(TFT_BLACK);
    tft.setTextColor(TFT_YELLOW, TFT_BLACK);
/*****************************************************************************************
                          AVVIO SOTTOPROGRAMMI PER PREPARARE  IL DISPLAY
*****************************************************************************************/
    drawbutton();
    displayfreq();
    axis();
/*****************************************************************************************
                                    AVVIO SETUP MODULO Si5351
*****************************************************************************************/
    si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
    si5351.set_correction(cal, SI5351_PLL_INPUT_XO);
    si5351.drive_strength(SI5351_CLK0,SI5351_DRIVE_8MA);
    si5351.output_enable(SI5351_CLK0, 1);                    //1   - Enable / 0 - Disable CLK
```

```
    si5351.output_enable(SI5351_CLK1, 1);                  //  uscita regolabile con attenuatore
    si5351.output_enable(SI5351_CLK2, 0);                  // uscita fissa
 /*********************************************************************************************
       QUESTA PARTE E' VISIBILE SOLO SUL SERIAL MONITOR E INDICA QUALE SKETCH E' CARICATO
*********************************************************************************************/
  Serial.println();
  Serial.println();
  Serial.print("avvio sketch");
  Serial.println(__FILE__);
  Serial.print("Caricato il:    ");
  Serial.println(__DATE__);
  Serial.print("Alle ore:    ");
  Serial.println(__TIME__);
  delay(500);
  preferences.putBool("cal_on_off", calibra);
}
/*********************************************************************************************
            SOTTOPROGRAMMA (VOID)  PER VISUALIZZARE GLI ASSI  CARTESIANI SUL DISPLAY
*********************************************************************************************/
void axis()
{
    tft.drawLine(bordo+15, SCREEN_HEIGHT - bordo , bordo+15, bordo+20, WHITE);
                          // punto superiore, punto inferiore   15+15,240-15   ,15+15,15+20 verticale
    tft.drawLine(bordo+15, SCREEN_HEIGHT - bordo, SCREEN_WIDTH-40, SCREEN_HEIGHT -bordo,WHITE);
                          // 15+15,240-15   320-40,240-15
    tft.setTextSize(0);
    tft.setCursor(0,40);
    tft.print(" 26dB");
    tft.drawLine(bordo+15, 40, SCREEN_WIDTH-40, 40, DARKCYAN);         // y = 40
    tft.setCursor(0,55);
    tft.print(" 20dB");
    tft.drawLine(bordo+15, 55, SCREEN_WIDTH-40, 55, DARKCYAN);         // y = 55
    tft.setCursor(0,70);
    tft.print(" 15dB");
    tft.drawLine(bordo+15, 70, SCREEN_WIDTH-40, 70, DARKCYAN);         // y = 70
    tft.setCursor(0,85);
    tft.print(" 10dB");
    tft.drawLine(bordo+15, 85, SCREEN_WIDTH-40, 85, DARKCYAN);         // y = 85
    tft.setCursor(0,100);
    tft.print("   5dB");
    tft.drawLine(bordo+15, 100, SCREEN_WIDTH-40,100, DARKCYAN);       // y = 100
    tft.setCursor(0,115);
    tft.print("   0dB");
    tft.drawLine(bordo+15, 115, SCREEN_WIDTH-40,115, DARKCYAN);       // y = 115
    tft.setCursor(0,130);
    tft.print("-5dB");
    tft.drawLine(bordo+15, 130, SCREEN_WIDTH-40,130, DARKCYAN);       // y = 130
    tft.setCursor(0,145);
    tft.print("-10dB");
```

```
    tft.drawLine(bordo+15,145, SCREEN_WIDTH-40,145, DARKCYAN);        // y = 145
    tft.setCursor(0,160);
    tft.print("-15dB");
    tft.drawLine(bordo+15, 160, SCREEN_WIDTH-40, 160, DARKCYAN);      // y = 160
    tft.setCursor(0,175);
    tft.print("-20dB");
    tft.drawLine(bordo+15,175, SCREEN_WIDTH-40,175, DARKCYAN);        // y = 175
    tft.setCursor(0,190);
    tft.print("-30dB");
    tft.drawLine(bordo+15,190, SCREEN_WIDTH-40,190, DARKCYAN);        // y = 190
    tft.setCursor(0,205);
    tft.print("-40dB");
    tft.drawLine(bordo+15,205, SCREEN_WIDTH-40,205, DARKCYAN);        // y = 205
    tft.setCursor(0,220);                                             // y = 220
    tft.print("-50dB");
}
/*********************************************************************************************
              VOID LEGGI POSIZIONE DEL PUNTATORE TOUCHSCREEN ED AZIONE SUCCESSIVA
*********************************************************************************************/
void leggi_xp_yp()
{
      if (xp >= 300 & yp >= 10 & yp <= 40)                    // CANCELLA GRAFICO
        {
           clearGraf();
        }
      if (xp >= 270 & xp < 290 & yp >= 10 & yp <= 40)         // CANCELLA GRAFICO
        {
            clearMarker();
        }
      calibra = preferences.getBool("cal_on_off", 0);         //calibrazione già fatta ? true
      if (xp >= 251 & xp < 269 & yp >= 10 & yp <= 40)         // CALIBRAZIONE verde avvia
        {
          if (calibra == true)
             {
               calibrazione();                                // void per calibrazione
             }
        }
       if ( yp>= 226 )                                  // marker posizione e valori relativi
         {
            if(xp > 290 )
             { xp = 290;}
               if(xp < 30 )
               { xp = 30;}
           tft.fillCircle(xp+5,MARK1_Y+5, 3,TFT_CYAN);                //cerchietto raggio 3 pixel   25/02/2025
           tft.drawFastVLine(xp+5,maxyb,240-maxyb-bordo,TFT_DARKGREY);          //linea grigia verticale
           tft.setTextSize(0);
           tft.setTextColor(TFT_GREEN);                       //cambio colore testo
           if (mkvert_Cur <= mkvert_Old)                      // controllo numero marker e posizione x
             {
```

```
          tft.setCursor(40,mkvert_Cur);
          tft.print("F: ");
          tft.print(mi_Freq[markerX]);                  // visualizza freq alla posizione del marker
          tft.setCursor(130,mkvert_Cur);
          dB = map(mi_Vinmax[markerX],12100,4950,27,-30);       // conversione valore in dB
          tft.print("dB: ");
          tft.print(dB);                                // visualizza valore in dB alla posizione del marker
           tft.setCursor(200,mkvert_Cur);
           delta = dB – dBmax;                          // calcolo e visualizzazione delta dB
           tft.print("Delta: ");
           tft.print(delta);
            mkvert_Cur = mkvert_Cur + 10;          //sposto scrittura su seconda riga
          tft.setTextColor(TFT_RED);
          tft.setCursor(xp-6,MARK1_Y+1);
          tft.print(mkCount);                          //numero del marker
          tft.setTextColor(TFT_YELLOW, TFT_BLACK);

         }
        mkCount = mkCount+1 ;
        if (mkCount >= 6)                               // massimo 5 marker
          {
           clearMarker();
          }
         mkvert_Old = mkvert_Cur;                // memorizza posizione verticale valori precedente marker
         delay(200);
     }

if (xp >= 300 & yp >= 30 & yp <= 50)       // aumenta/diminuisce la frequenza in base allo step
    {
      freq = freq + fstep;
    }
      if (freq >= 160000000) {freq = 160000000;}
 if (xp >= 300 & yp >= 55 & yp <= 75)
    { freq = freq - fstep;
    }
       if (fstep == 1000000 && freq <= 1000000)   // LIMITE DI FRERQUENZA INFERIORE
            { freq = 1000000; }       // la frequenza non può essere mai negativa
       if ( freq < 1000000)
           { freq = 1000000;   }
 if (xp >= 300 & yp >= 80 & yp <= 100)           // aumenta/diminuisce  valore dello step
    {
     stpcount++;
     if(stpcount > 11) stpcount = 1;
     setstep();                                         // void impostazione step
     }
if (xp >= 300 & yp >= 105 & yp <= 125)
    {
      stpcount--;
      if(stpcount < 0) stpcount = 1;
```

```
          setstep();                                  // void impostazione step
        }
    if (xp >= 300 & yp >= 130 & yp <= 150)        // aumenta/diminuisce BANDA
        {
        count++;
        if (count > 12) count = 1;
        bandpresets();                              // void impostazione banda
        }
   if (xp >= 300 & yp >= 155 & yp <= 175)
        {
          count--;
          if (count < 0) count = 1;
          bandpresets();                            // void impostazione banda
        }
    if (xp >= 300 & yp >= 180 & yp <= 200)     // set   sweep
        {
           sweeprange();                            // visualizza range di frequenza
        }

    if (xp >= 300 & yp >= 205 & yp <= 240)    // start  sweep
        {
           do_scan();                               // void scansione
        }
    tunegen();                                      // void invio frequenza a Si5351
  }
/*********************************************************************************************
          LOOP PRINCIPALE – CONTROLLA TUTTO IL CICLO DELLO SKETCH- SI AVVIA DOPO SETUP
*********************************************************************************************/
void loop()
{
   newPosition = encoder.getCount() / 2;         // controlla ciclicamente se l' encoder è stato girato
   leggi_Encoder();                              // legge e imposta freq in base alla rotazione CW 0 CCW
   if (touchscreen.tirqTouched() && touchscreen.touched())   // la parte seguente controlla ciclicamente il touchscreen
   {
    TS_Point p = touchscreen.getPoint();                 // punto sul  Touchscreen
    x = map(p.x, 200, 3700, 1, SCREEN_WIDTH);         //200,3700 mappatura entro i limiti in pixel
    y = map(p.y, 240, 3800, 1, SCREEN_HEIGHT);        //240,3800
    xp =x;
    yp = y;
    markerX = xp-30;                  //sincronizza posizione con inizio grafico
    drawbutton();                     // ridisegna i pulsanti virtuali
    axis();                           // ridisegna  assi per il grafico
    leggi_xp_yp();                    //  chiama void per azioni in base alla posizione del punto toccato
    displayfreq();                    // visualizza la frequenza generata al momento da  Si5351
   }
    tunegen();                        // void invio frequenza a Si5351
    sweepcontinuo =(digitalRead(pinSwcontinuo));      //controola se sweep singolo o continuo
    if (sweepcontinuo == LOW)
      {
```

```
          tft.fillRect(STOP_SWEEP_X, STOP_SWEEP_Y, STOP_SWEEP_W, STOP_SWEEP_H, TFT_RED);
          do_scan();    // se swepp continuo avvia scansione e dopo 2 secondi la ripete fino a che lo switch non cmbia
          delay(2000);
        }
    resetsoft = (digitalRead(pinResSof));      // reset completo via software in caso di blocco
    if (resetsoft == LOW)                      // pulsante reset premuto
       { resetsoft = HIGH;
        ESP.restart();
       }
    delay(100);
}
/*********************************************************************************************
          CANCELLA GRAFICO SOVRASCRIVENDO RETTANGOLO = COLORE SFONDO
*********************************************************************************************/
 void clearGraf()
 {
   tft.fillRect(30,21,320-40, 240-15, TFT_BLACK);    // cancella solo il grafico
   drawbutton();                    // ridisegna i pulsanti virtuali
   axis();                          // ridisegna assi per il grafico
   displayfreq();                   // visualizza la frequenza generata
   markerX = 0;                     // azzera posizione marker
   mkvert_Cur = 40;
 }
/*********************************************************************************************
          CANCELLA MARKER SOVRASCRIVENDO RETTANGOLO = COLORE SFONDO
*********************************************************************************************/
void clearMarker()
{
  tft.fillRect(35,38,320-80, 60, TFT_BLACK);          // cancella solo marker
  tft.fillRect(30, MARK1_Y, 320, 10, TFT_BLACK);  // CANCELLA LINEA MARKER
  mkvert_Cur = 40;
  mkCount = 1;
  axis();                           // ridisegna assi per il grafico
}
/*********************************************************************************************
                          VISUALIZZA LA FREQUENZA CON FORMATO 000.000.000 Hz
*********************************************************************************************/
void displayfreq()
{
    tft.setTextSize(0);               // 6x8 pixel
    tft.fillRect(0, 0, 210, 12, TFT_BLACK);    // cancella prima riga del display
    tft.setCursor(2,2);
    tft.print("Freq:");
    millions = (freq/1000000);
    hundredthousands = ((freq/100000)%10);
    tenthousands = ((freq/10000)%10);
    thousands = ((freq/1000)%10);
    hundreds = ((freq/100)%10);
    tens = ((freq/10)%10);
```

```
    ones = ((freq/1)%10);
    tft.setCursor(35,2);
    tft.print(millions);
    tft.print(".");
    tft.print(hundredthousands);
    tft.print(tenthousands);
    tft.print(thousands);
    tft.print(".");
    tft.print(hundreds);
    tft.print(tens);
    tft.print(ones);
    tft.print(" Hz    ");
    tft.print(band);
    displayStep();
}
/*****************************************************************************************
                                    VISUALIZZA LO STEP IMPOSTATO
*****************************************************************************************/
void displayStep()
{
   tft.setCursor(2,12);
   tft.print("Step:");
   tft.fillRect(30,12, 40, 10, TFT_BLACK);       / / cancella seconda riga del display
   tft.setCursor(35,12);
    if (stp == 1) tft.print("10Hz");          // stp impostato con pulsanti virtuali
    if (stp == 2) tft.print("50Hz");
    if (stp == 3) tft.print("100Hz");
    if (stp == 4) tft.print("500Hz");
    if (stp == 5) tft.print("1kHz");
    if (stp == 6) tft.print("5kHz");
    if (stp == 7) tft.print("10kHz");
    if (stp == 8) tft.print("100kHz");
    if (stp == 9) tft.print("1MHz");
    if (stp == 10) tft.print("AUTO");
}
/*****************************************************************************************
                                    CONTROLLA IL  MODULO Si5351
*****************************************************************************************/
void tunegen()
{
  si5351.set_freq(freq * 100ULL, SI5351_CLK1);    //+ (interfreq * 1000ULL)
  si5351.set_freq(freq * 100ULL, SI5351_CLK0);    //+ EVENTUALE (interfreq * 1000ULL)
}
/*****************************************************************************************
                                    SELEZIONE BANDA CON PULSANTI VIRTUALI
*****************************************************************************************/
void bandpresets()
  {
    switch (count)
```

```
    {
    case 1:
          freq = 1000000; band = "FREE";                    // frequenza inizio ;band = ("FREE");
          tunegen(); break;

     case 2:
          freq = 3500000; band = "80m"; fstepband = ((3800000-3500000)/245) ;     // 80 metri
          break;
    case 3:
          freq = 7000000; band = "40m"; fstepband   = ((7200000-7000000)/245);    // 40 metri
          break;
    case 4:
          freq = 10100000; band = "30m"; fstepband   = ((10150000-10100000)/245);       // 30 metri
          break;
    case 5:
          freq = 14000000; band = "20m"; fstepband   = ((14350000-14000000)/245);    // 20 metri
          break;
    case 6:
          freq = 18068000; band = "17m"; fstepband   = ((18168000-18068000)/245);    // 17 metri
          break;
    case 7:
          freq = 21000000; band = "15m"; fstepband   = ((21450000-21000000)/245);  // 15 metri
          break;
    case 8:
          freq = 24890000; band = "12m"; fstepband   = ((24990000-24890000)/245);  // 12 metri
          break;
    case 9:
          freq = 28000000; band = "10m"; fstepband   = ((29700000-28000000)/245);  // 10 metri
          break;
    case 10:
          freq = 50000000; band = "6m"; fstepband   = ((51000000-50000000)/245);  // 6  metri
    case 11:
          freq = 88000000; band = "WFM"; fstepband   = ((108000000-88000000)/245); // WFM
          break;
    case 12:
          freq = 144000000; band = "2m"; fstepband   = ((146000000-144000000)/245);  // 2   metri
          break;
  }
  setstep();                          // richiama void set step

          si5351.pll_reset(SI5351_PLLA);   // reset del modulo
}
/*********************************************************************************************
                    VOID SET STEP IN BASE AL VALORE IMPOSTATO CON PULSANTI VIRTUALI
*********************************************************************************************/
void setstep()
  {
  switch (stpcount)                                 // modificato dai pulsanti virtuali step + step-
  {
```

```
    case 1: stp = 1; fstep = 10; break;
    case 2: stp = 2; fstep = 50; break;
    case 3: stp = 3; fstep = 100; break;
    case 4: stp = 4; fstep = 500; break;
    case 5: stp = 5; fstep = 1000; break;
    case 6: stp = 6; fstep = 5000; break;
    case 7: stp = 7; fstep = 10000; break;
    case 8: stp = 8; fstep = 100000; break;
    case 9: stp = 9; fstep = 1000000; break;
    case 10: stp = 10; fstep = fstepband;              // nuova funzione   per banda HF
             break;
    case 11: break;
  }
   displayStep();                    // visualizza il valore dello step
 }
/*****************************************************************************************
                IMPOSTAZIONE DELLO STEP E VISUALIZZAZIONE DEL RANGE CALCOLATO
*****************************************************************************************/
void sweeprange()
{
    startfreq = freq;
    spanfreq = fstep * 245;          //numero massimo di step per lo spazio disponibile del grafico
    stopfreq =  freq + spanfreq;
    displayswstp();
}
/*****************************************************************************************
                        VISUALIZZAZIONE DELLA FREQUENZA DI  INIZIO FINE SWEEP
*****************************************************************************************/
void displayswstp()
{
    tft.setTextSize(0);
    tft.setCursor(78,12);                    // terza riga 2,22
    tft.print("SW:");
    tft.fillRect(95,12, 140, 10, TFT_BLACK);   // cancella dati Sweep sulla  terza riga
    tft.setCursor(95,12);                    // posizione testo
    tft.print(millions);
    tft.print(".");
    tft.print(hundredthousands);
    tft.print(tenthousands);
    tft.print(thousands);
    tft.print(".");
    tft.print(hundreds);
    tft.print(tens);
    tft.print(ones);
    tft.setCursor(160,12);
    tft.print("->");
    millionfst = (stopfreq/1000000);                    //     0.000.000
    hundredthousandfst = ((stopfreq/100000)%10);        //       000.000
    tenthousandfst = ((stopfreq/10000)%10);             //         00.000
```

```
    thousandfst = ((stopfreq/1000)%10);                 //          0.000
    hundredfst = ((stopfreq/100)%10);                   //           000
    tenfst = ((stopfreq/10)%10);                        //            00
    onefst = ((stopfreq/1)%10);                         //             0
    tft.setCursor(175,12);                  //   fine sweep
    tft.print(millionfst);
    tft.print(".");
    tft.print(hundredthousandfst);
    tft.print(tenthousandfst);
    tft.print(thousandfst);
    tft.print(".");
    tft.print(hundredfst);
    tft.print(tenfst);
    tft.print(onefst);
    tft.fillRect(30,22, 160, 10, TFT_BLACK);      // cancella Fmax sulla terza riga
}
/*********************************************************************************************
                              SCANSIONE – MISURA – VISUALIZZAZIONE SU GRAFICO X-Y
*********************************************************************************************/
void do_scan()
{
     prev_yb = 225 ;               // coordinate verticali inizio grafico   bordo+20 in alto    225 in basso
     xb = 30;                      // coordinate orizzontali inizio grafico   30
     for (int i=0; i<=290; i++)          //i = 290   per non superare l'indice dell'array
       {
          leggi_val();                        // esegue misura e corregge in base alla calibrazione
          yb = map(VIN, 12500, 4950, 30, 200);     // calcola pos y in base a max e min di VIN e posizione 30,200
          if (yb > 220){yb = 220;}                  // se si supera 220 ( in basso, il grafico parte da 0,0 alto sx)
          mi_Freq[i] = freq;                // registra valori per ogni lettura
          mi_Vinmax[i] = VIN;               // registra valori per ogni lettura
          mi_Xb[i] = xb;                    // registra valori per ogni lettura
          mi_Yb[i] = yb;                    // registra valori per ogni lettura
          if (VIN > VIN_old)                // cerco il massimo dei valori
             {
              VIN_max = VIN;
              markf1 = freq;          // frequenza al valore max
              maxxb =   i;            //posizione X al massimo valore
              maxyb = yb;             // posizione y al massimo valore
              mark1 = xb;             //posizione X al massimo valore
              VIN_old = VIN;
             }
          tft.drawLine(xb, prev_yb, xb+1, yb,WHITE );         // partenza grafico in alto yb = 15+20  colore WHITE
          prev_yb = yb;                                       // punto precedente del grafico
          xb++;                                               // incremento posizione asse X
          freq = freq + fstep;                                // incremento frequenza
          tunegen();                                          // richiamo funzione generatore
          if (xb >= 285 )                                     // 260   controllo fine swepp
            {
               break;                         // esce dal ciclo for
```

```
            }
        }

      dBmax = map(VIN_max,12100,4950,27,-30);          // centra valori dB tra +27  e -26
      freq = startfreq;                                // per ripartire dall'inizio e ripetere la scansione
      display_max();                                   //visualizza valore massimo e frequenza
      displayfreq();                                   // visualizza  frequenza
}


/*****************************************************************************************
       DISPLAY VALORI MASSIMI DI FRERQUENZA E AMPIEZZA – DISEGNA LINEA VERDE VERTICALE
*****************************************************************************************/
void display_max()
{
    if(mark1 > 280 )                    // elimina linea fuori grafico
          { mark1 = 280;}
          if(mark1 < 30 )
          { mark1 = 30;}
    tft.drawFastVLine(mark1,maxyb,240-maxyb-bordo,TFT_GREEN);          //linea verde verticale
    tft.setCursor(2,22);                                               // display fmax = markf1
    tft.print("Fmax: ");
    tft.fillRect(30,22, 160, 10, TFT_BLACK);                           // cancella terza riga
    millionfm = (markf1/1000000);                    // 0.000.000
    hundredthousandfm = ((markf1/100000)%10);        //    000.000
    tenthousandfm = ((markf1/10000)%10);             //     00.000
    thousandfm = ((markf1/1000)%10);                 //      0.000
    hundredfm = ((markf1/100)%10);                   //        000
    tenfm = ((markf1/10)%10);                        //         00
    onefm = ((markf1/1)%10);                         //          0
    tft.setCursor(35,22);
    tft.print(millionfm);
    tft.print(".");
    tft.print(hundredthousandfm);
    tft.print(tenthousandfm);
    tft.print(thousandfm);
    tft.print(".");
    tft.print(hundredfm);
    tft.print(tenfm);
    tft.print(onefm);
    tft.setCursor(105,22);
    tft.print("dBmax: ");
    tft.print(dBmax);
    mark1 = 0;                                  // azzeramento variabili per ciclo successivo
    markf1 = 0;
    maxxb = 0;
    VIN_old = 0;
  }
```

```
/*****************************************************************************************
      VOID PER DISEGNARE I PULSANTI VIRTUALI E I TESTI VERTICALI IN BASE ALLA LORO POSIZIONE
                        INDICATA IN        #define nome_X,nome_Y,nome_H,nome_V
*****************************************************************************************/
void drawbutton()
{
  tft.setTextColor(TFT_YELLOW);
  tft.setTextSize(2);
  tft.fillRect(FRE_PIU_X, FRE_PIU_Y, FRE_PIU_W, FRE_PIU_H, TFT_BLUE);      //esempio 300,5,20,20
  tft.drawString("+", FRE_PIU_X-5 + (FRE_PIU_W / 2), FRE_PIU_Y-5 +(FRE_PIU_H / 2));    //295+10,0+10
  tft.fillRect(FRE_MENO_X, FRE_MENO_Y, FRE_MENO_W, FRE_MENO_H, TFT_BLUE);
  tft.drawString("-", FRE_MENO_X-5 + (FRE_MENO_W / 2), FRE_MENO_Y-5 +(FRE_MENO_H / 2));
  tft.setTextSize(1);
  tft.drawString("F", FRE_PIU_X-22+ (FRE_PIU_W / 2), FRE_PIU_Y-7 +(FRE_PIU_H / 2));
  tft.drawString("r", FRE_PIU_X-22+ (FRE_PIU_W / 2), FRE_PIU_Y+2 +(FRE_PIU_H / 2));
  tft.drawString("e", FRE_PIU_X-22+ (FRE_PIU_W / 2), FRE_PIU_Y+12+FRE_PIU_H / ));
  tft.drawString("q", FRE_PIU_X-22+ (FRE_PIU_W / 2), FRE_PIU_Y+22 +(FRE_PIU_H / ));
  tft.setTextSize(2);
  tft.fillRect(STEP_PIU_X, STEP_PIU_Y, STEP_PIU_W, STEP_PIU_H, TFT_BLUE);
  tft.drawString("+", STEP_PIU_X-5 + (STEP_PIU_W / 2), STEP_PIU_Y-5 +(STEP_PIU_H / 2));
  tft.fillRect(STEP_MENO_X, STEP_MENO_Y, STEP_MENO_W, STEP_MENO_H, TFT_BLUE);
  tft.drawString("-", STEP_MENO_X-5 + (STEP_MENO_W / 2), STEP_MENO_Y-5 +(STEP_MENO_H / 2));
  tft.setTextSize(1);
  tft.drawString("S", STEP_PIU_X-22+ (STEP_PIU_W / 2), STEP_PIU_Y-7 +(STEP_PIU_H / 2));
  tft.drawString("t", STEP_PIU_X-22+ (STEP_PIU_W / 2), STEP_PIU_Y+2 +(STEP_PIU_H / 2));
  tft.drawString("e", STEP_PIU_X-22+ (STEP_PIU_W / 2), STEP_PIU_Y+12 +(STEP_PIU_H /2));
  tft.drawString("p", STEP_PIU_X-22+ (STEP_PIU_W / 2), STEP_PIU_Y+22 +(STEP_PIU_H / 2));
  tft.setTextSize(2);
  tft.fillRect(BAND_PIU_X, BAND_PIU_Y, BAND_PIU_W, BAND_PIU_H, TFT_BLUE);
  tft.drawString("+", BAND_PIU_X-5 + (BAND_PIU_W / 2), BAND_PIU_Y-5 +(BAND_PIU_H / 2));
  tft.fillRect(BAND_MENO_X, BAND_MENO_Y, BAND_MENO_W, BAND_MENO_H, TFT_BLUE);
  tft.drawString("-", BAND_MENO_X-5 + (BAND_MENO_W / 2), BAND_MENO_Y-5 +(BAND_MENO_H / 2));
  tft.setTextSize(1);
  tft.drawString("B", BAND_PIU_X-22+ (BAND_PIU_W / 2), BAND_PIU_Y-7 +(BAND_PIU_H / 2));
  tft.drawString("a", BAND_PIU_X-22+ (BAND_PIU_W / 2), BAND_PIU_Y+2 +(BAND_PIU_H / 2));
  tft.drawString("n", BAND_PIU_X-22+ (BAND_PIU_W / 2), BAND_PIU_Y+12 +(BAND_PIU_H / 2));
  tft.drawString("d", BAND_PIU_X-22+ (BAND_PIU_W / 2), BAND_PIU_Y+22 +(BAND_PIU_H / 2));
  tft.fillRect(START_SWEEP_X, START_SWEEP_Y, START_SWEEP_W, START_SWEEP_H, TFT_YELLOW);
    if (sweepcontinuo == LOW)       // cambio colore a start sweep
      {
        tft.fillRect(STOP_SWEEP_X, STOP_SWEEP_Y, STOP_SWEEP_W, STOP_SWEEP_H, TFT_RED);
      }
    else
     { tft.fillRect(STOP_SWEEP_X, STOP_SWEEP_Y, STOP_SWEEP_W, STOP_SWEEP_H, TFT_GREEN);
     }
    tft.setTextSize(1);
tft.drawString("S", START_SWEEP_X-22+ (START_SWEEP_W / 2), START_SWEEP_Y-7 (START_SWEEP_H / 2));
tft.drawString("w", START_SWEEP_X-22+ START_SWEEP_W / 2), START_SWEEP_Y+2 (START_SWEEP_H / 2));
tft.drawString("e", START_SWEEP_X-22+(START_SWEEP_W / 2), START_SWEEP_Y+12+(START_SWEEP_H /2));
```

```
tft.drawString("p", START_SWEEP_X-22+ (START_SWEEP_W / 2), START_SWEEP_Y+22 +(START_SWEEP_H / 2));
    tft.setTextSize(1);
    tft.fillRect(CLEAR_X, CLEAR_Y, CLEAR_W, CLEAR_H, TFT_RED);
    tft.fillRect(CLMKR_X, CLMKR_Y, CLMKR_W, CLMKR_H, TFT_RED);
    tft.setTextColor(TFT_WHITE, TFT_RED);
    tft.drawString("CL", CLEAR_X+3, CLEAR_Y+2 );                    //280+3,2+2
    tft.drawString("GR", CLEAR_X+3, CLEAR_Y+10 );                   //280+3,2+10
    tft.drawString("CL", CLMKR_X+3, CLMKR_Y+2 );                    //280+3,2+2
    tft.drawString("MKR", CLMKR_X+2, CLMKR_Y+10 );                  //280+3,2+2
    if (calibra == false)                   // calibrazione eseguita colore verde
      {
      tft.fillRect(CALI_X, CALI_Y, CALI_W, CALI_H, TFT_GREEN);
      tft.setTextColor(TFT_BLUE,GREEN);
      tft.drawString("CAL", CALI_X+2, CALI_Y+3 );                   //280+3,2+2
      }
    else if (calibra == true)           // calibrazione da fare colore rosso
      {
        tft.fillRect(CALI_X, CALI_Y, CALI_W, CALI_H, TFT_RED);
        tft.drawString("CAL", CALI_X+2, CALI_Y+3 );                 //280+3,2+2
      }
      tft.setTextColor(TFT_YELLOW);
}
/*************************************************************************************************
                                LEGGE I VALORI DAL CIRCUITO RF CON IL MODULO ADS1115
*************************************************************************************************/
void leggi_val()
{
   VIN = ads.readADC_SingleEnded(0);            // porta n°0 delle 4 disponibili
    if (calibra == false)                       //correzione rispetto a 21 MHz dopo la calibrazione
        {
          if (band == "FREE") { VIN = VIN + cor_Misura[1];  }
          if (band == "80m")  {  VIN = VIN + cor_Misura[2];  }
          if (band == "40m")   { VIN =  VIN  + cor_Misura[3];  }
          if (band == "30m")  {  VIN =  VIN + cor_Misura[4];  }
          if (band == "20m")  {  VIN =  VIN + cor_Misura[5];  }
          if (band == "17m") {  VIN =  VIN + cor_Misura[6];  }
          if (band == "15m") {  VIN =  VIN + cor_Misura[7];  }
          if (band == "12m") {  VIN =  VIN + cor_Misura[8];  }
          if (band == "10m") {  VIN =  VIN + cor_Misura[9];  }
          if (band == "6m") {  VIN =  VIN + cor_Misura[10];  }
          if (band == "WFM") {  VIN =  VIN + cor_Misura[11];  }
          if (band == "2m") {  VIN =  VIN + cor_Misura[12];  }
      }
}
/*************************************************************************************************
            LEGGE ENCODE - DETERMINA LA ROTAZIONE CW / CCW – E VARIA LA FREQUENZA
*************************************************************************************************/
void leggi_Encoder()
{
```

```
  newPosition = encoder.getCount() / 2;
      if (newPosition != oldPosition)
      {
        if (newPosition > oldPosition)
        {
          direzione = "CW";
          freq = freq + fstep;
        }
        if (newPosition < oldPosition)
            {
             direzione = "CCW";
             freq = freq - fstep;
             if ( freq < 1000000)
             {
               freq = 1000000;
             }
            }
          oldPosition = newPosition;
          displayfreq();
          tunegen();
      }
}
/*********************************************************************************************
                    CALIBRAZIONE E REGISTRO SU ARRAY    float cor_Misura[12];   bool calibra ;
*********************************************************************************************/
void calibrazione()
{
    freq = elenFreq[7]; band = "15m";
    tunegen();                              // imposta freq = 21000 Hz
    leggi_val();                            // legge valori senza apportare correzioni calibra == true
    Vcal = VIN;
    tft.setCursor(50,35);
    tft.print("Freq rifer cal >    ");        // visualizza freq riferimento
    tft.print(freq);
   int posvert = 50;                        // posizione verticale prima riga
    for (int w =1 ; w < 13; w++)        // esegue misura su tutte le bande e visualizza su display
      {
        freq = elenFreq[w];
        tunegen();
        leggi_val();
        cor_Misura[w] = Vcal-VIN;
        tft.setCursor(50,posvert);
        tft.print("Freq cal >    ");
        tft.print(elenFreq[w]);
        tft.setCursor(200,posvert);
        tft.print(cor_Misura[w]);
        posvert = posvert + 15;         // incrementa posizione verticale per le altre righe
      }
    delay(2000);
```

```
    calibra =! calibra;                         // calibrazione eseguita cambio valore calibra == false
    preferences.putBool("cal_on_off", calibra);
    freq = elenFreq[1]; band = "FREE";   // riparte con band == FREE  frequenza 1.000.000 Hz
    tunegen();
    clearGraf();                                // calcella misure calibrazione
}
/*****************************************************************************************/
```