EMOTATOR 1105MSX

Umbau der Antennen-Richtungsanzeige

DK5BD

Der Emotator Rotor hat viele Jahre ohne Probleme gearbeitet. Die Richtungsanzeige wird über einen Motor angetrieben, der über ein vom Richtungspoti im Rotorgehäuse gelieferte Gleichspannung gesteuert wird. Irgendwann fingen die Anzeigeprobleme an. Der Zeiger blieb stehen oder ruckte, obwohl der Rotor in die richtige Richtung lief. Einen offensichtlichen Defekt konnte ich relativ schnell beheben: Der Antriebsgummiriemen war mit der Zeit verschlissen. Aber nach einiger Zeit gab es wieder Probleme: Die Anzeige ruckte erneut oder blieb auch komplett stehen. Diesmal hatte ich ein defektes Poti im Steuergehäuse festgestellt, dass die Zeigerposition an einen Operationsverstärker lieferte. Da das Poti in den Originalwerten nicht beschaffbar war und ich mich mittlerweile in die Arduino-Programmierung eingearbeitet hatte, entschloss ich mich, die Anzeige komplett gegen eine Arduino Grafik-Lösung auszutauschen.

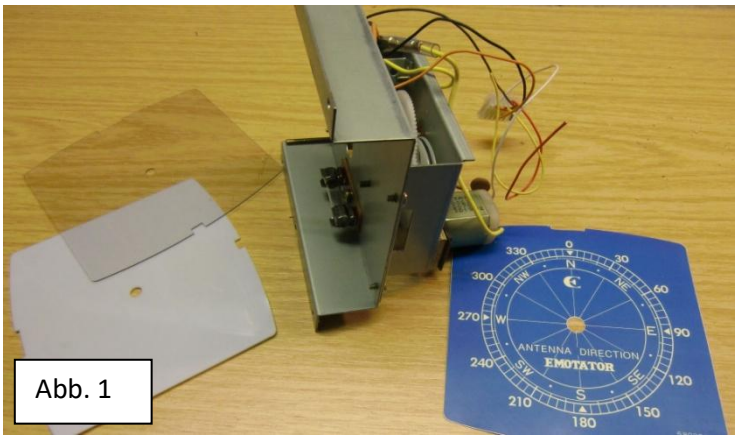

Abb. 1

Zunächst baute ich die gesamte Anzeigemechanik inklusive der Elektronikplatine aus.

In den nun freigewordenen Platz mußte ich nun den Arduino samt Grafik-Shield einbauen.

Als Arduino kam jetzt der Mega in Frage, da die vorgesehene Grafik praktisch alle Ports des Arduino Uno belegt. Einen Port brauchte ich aber für die Abfrage des Gleichspannungswertes des Rotor-Potentiometers. Das liefert eine Spannung von 0V-5V, die an den AD-Port des Arduino gehen.

Es stellte sich jetzt heraus, daß der Trafo nicht die Versorgungsspannung für den Arduino samt Grafik (5V, 200mA) liefern konnte, so daß ich einfach ein kleines Steckernetzteil vorsehen mußte. Dieses liefert 12V bei max. 500mA. Dem Arduino wollte ich aber nur 8V spendieren und habe deshalb auf einer kleinen Lochplatine einen 8V Längsregler eingebaut.

Nun mußte ich eine Trägerplatte herstellen, auf die der Arduino samt Grafik montiert werden konnte. Diese wurde dann mit 4 M3-Gewindestäben an der Frontplatte montiert. Ich konnte diese in die vorhandenen Löcher schrauben, die vorher die Blechschrauben der Mechanik aufgenommen hatten.

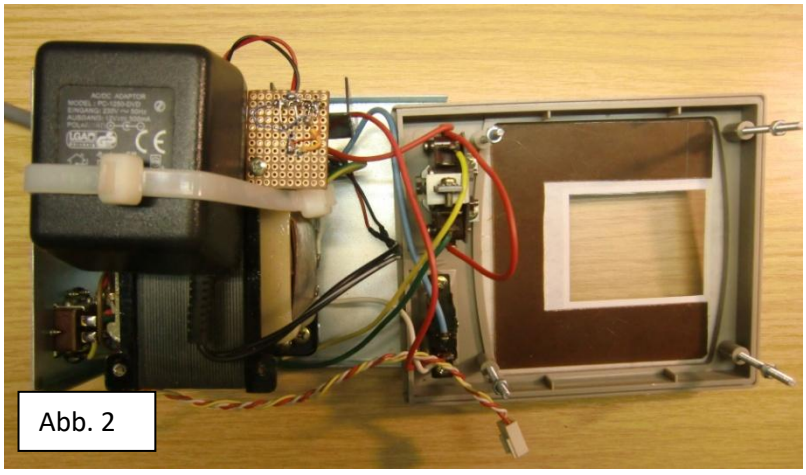
Abb. 2

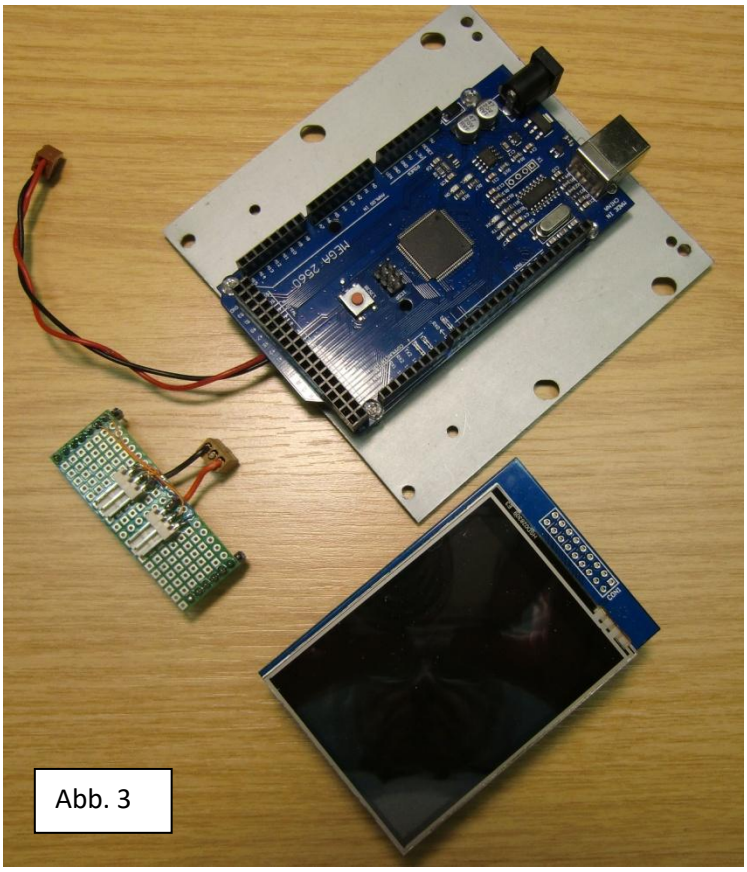

Abb. 3

Der Arduino Mega ist hier auf der Platte montiert. Die Platte stammt von einem ausgedienten PC-Netzteil und wurde auf Maß geschnitten.

Für die Montage des Poti-Steckers mußte ich eine kleine Lochplatine herstellen.

Das Kabel auf dem Bild dient der Stromversorgung des Arduinos.

Und so sieht das Ganze zusammengesteckt aus.

In Abb. 2 kann bereits der Ausschnitt in der Frontplatte gesehen werden. Die Frontplatte habe ich mal wieder mit dem Designer Programm entworfen, ausgedruckt und dann laminiert.

Da die abzudeckende Fläche zu offen war, habe ich sie noch mit einer Pertinaxplatte hinterfüttert. Das Ganze mit doppelseitigem Klebeband fixiert.

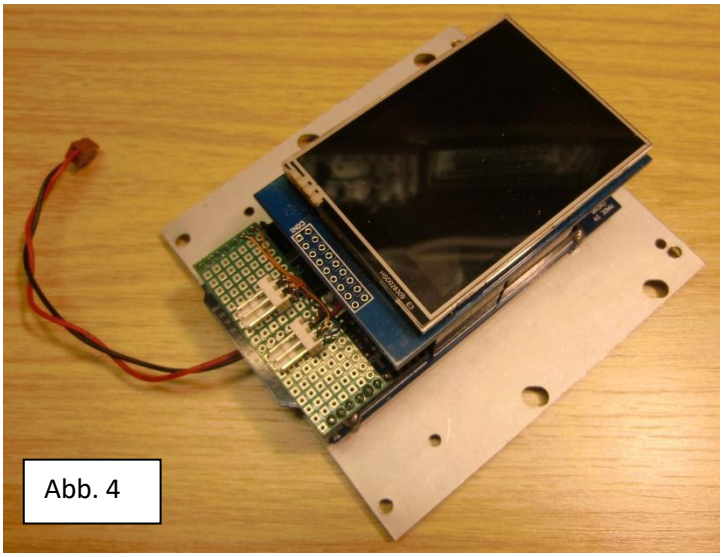
Abb. 4

Hier ist die neue Anzeige montiert und mit der neuen Betriebsspannung verbunden.

Das 3-polige Kabel für das Potentiometer geht direkt an die Buchse für das Steuerkabel, das zum Rotor führt.

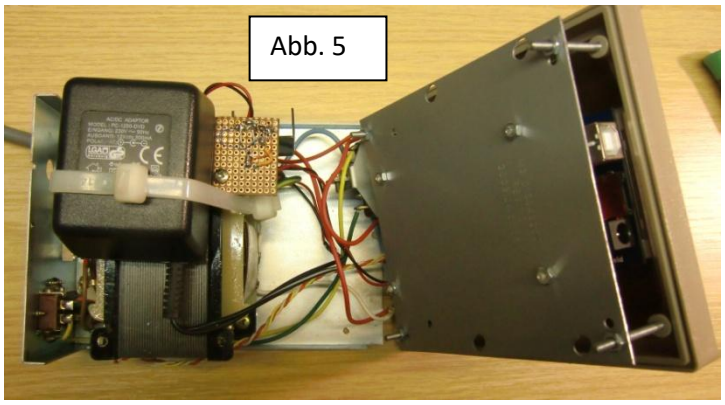
Abb. 5

Abb. 6

Und so sieht jetzt die neue Front aus. Als zusätzliches Feature habe ich eine digitale Richtungsausgabe vorgesehen.

Der Arduino kann jetzt auch zu jeder Zeit mit neuer Software geladen werden.

Ich denke daran, daß heutige Logbuchprogramme nicht nur den Transceiver steuern, sondern auch die Antennenrichtung vorgeben können, so daß der Rotor dann automatisch die Antenne in die richtige Richtung dreht.

Eine automatsche Antennenumschaltung habe ich bereits realisiert.

Abb. 7

Der automatische Antennenumschalter ist mit dem IC-706 verbunden und wertet die Band-abhängige Steuerspannung aus.

Der Schalter kann aber auch manuell bedient werden, da auch mein Drake TR7 verbunden werden soll und dieser natürlich eine solche Steuerspannung nicht zur Verfügung stellt.

Das war jetzt einmal ein nützliches Projekt während der Quarantänezeit wegen der Corona Pandemie.

Das Programm

```
// Rotor Display für Emotator Rotor von DK5BD
// In Betrieb genommen am 05.05.2020
#include <Elegoo_GFX.h>    // Core graphics library
#include <Elegoo_TFTLCD.h> // Hardware-specific library
#include <TouchScreen.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 (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
// 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
// Color definitions
#define ILI9341_BLACK       0x0000      /*   0,   0,   0 */
#define ILI9341_NAVY        0x000F      /*   0,   0, 128 */
#define ILI9341_DARKGREEN   0x03E0      /*   0, 128,   0 */
#define ILI9341_DARKCYAN    0x03EF      /*   0, 128, 128 */
#define ILI9341_MAROON      0x7800      /* 128,   0,   0 */
#define ILI9341_PURPLE      0x780F      /* 128,   0, 128 */
#define ILI9341_OLIVE       0x7BE0      /* 128, 128,   0 */
#define ILI9341_LIGHTGREY   0xC618      /* 192, 192, 192 */
#define ILI9341_DARKGREY    0x7BEF      /* 128, 128, 128 */
#define ILI9341_BLUE        0x001F      /*   0,   0, 255 */
#define ILI9341_GREEN       0x07E0      /*   0, 255,   0 */
#define ILI9341_CYAN        0x07FF      /*   0, 255, 255 */
#define ILI9341_RED         0xF800      /* 255,   0,   0 */
#define ILI9341_MAGENTA     0xF81F      /* 255,   0, 255 */
#define ILI9341_YELLOW      0xFFE0      /* 255, 255,   0 */
#define ILI9341_WHITE       0xFFFF      /* 255, 255, 255 */
#define ILI9341_ORANGE      0xFD20      /* 255, 165,   0 */
#define ILI9341_GREENYELLOW 0xAFE5      /* 173, 255,  47 */
#define ILI9341_PINK        0xF81F
/****************** UI details */
#define BUTTON_X 40
#define BUTTON_Y 100
#define BUTTON_W 60
#define BUTTON_H 30
#define BUTTON_SPACING_X 20
#define BUTTON_SPACING_Y 20
#define BUTTON_TEXTSIZE 2
// text box where numbers go
#define TEXT_X 10
#define TEXT_Y 10
#define TEXT_W 220
#define TEXT_H 50
#define TEXT_TSIZE 3
#define TEXT_TCOLOR ILI9341_MAGENTA
// the data (phone #) we store in the textfield
#define TEXT_LEN 12
char textfield[TEXT_LEN+1] = "";
uint8_t textfield_i=0;

#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
//*********************************************************************Rotor Variablen
int XX = 1;
int YY = 1;
float Richtung = 0;
int Smeter = 0;
float RichtungAlt=0;
float Altrichtung=0;
int richtungPin = A11;
int smeterPin = A12;
float Direction = 0;

Elegoo_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
//TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);
// If using the shield, all control and data lines are fixed, and
```

```
// a simpler declaration can optionally be used:
// Elegoo_TFTLCD tft;

void setup(void) {
  Serial.begin(9600);
  Serial.println(F("TFT LCD test"));

#ifdef USE_Elegoo_SHIELD_PINOUT
  Serial.println(F("Using Elegoo 2.8\" TFT Arduino Shield Pinout"));
#else
  Serial.println(F("Using Elegoo 2.8\" TFT Breakout Board Pinout"));
#endif

  Serial.print("TFT size is "); Serial.print(tft.width()); Serial.print("x");
Serial.println(tft.height());
  tft.reset();
  uint16_t identifier = tft.readID();
  if(identifier == 0x9325) {
    Serial.println(F("Found ILI9325 LCD driver"));
  } else if(identifier == 0x9328) {
    Serial.println(F("Found ILI9328 LCD driver"));
  } else if(identifier == 0x4535) {
    Serial.println(F("Found LGDP4535 LCD driver"));
  }else if(identifier == 0x7575) {
    Serial.println(F("Found HX8347G LCD driver"));
  } else if(identifier == 0x9341) {
    Serial.println(F("Found ILI9341 LCD driver"));
  } else if(identifier == 0x8357) {
    Serial.println(F("Found HX8357D LCD driver"));
  } else if(identifier==0x0101)
  {
      identifier=0x9341;
       Serial.println(F("Found 0x9341 LCD driver"));
  }else {
    Serial.print(F("Unknown LCD driver chip: "));
    Serial.println(identifier, HEX);
    Serial.println(F("If using the Elegoo 2.8\" TFT Arduino shield, the line:"));
    Serial.println(F("  #define USE_Elegoo_SHIELD_PINOUT"));
    Serial.println(F("should appear in the library header (Elegoo_TFT.h)."));
    Serial.println(F("If using the breakout board, it should NOT be #defined!"));
    Serial.println(F("Also if using the breakout, double-check that all wiring"));
    Serial.println(F("matches the tutorial."));
    identifier=0x9341;
  }
  tft.begin(identifier);
  tft.setRotation(2);
  tft.fillScreen(BLACK);
   //**********************************************************************Kompass Rose zeichnen
  tft.drawCircle(120,210,90,ILI9341_RED);
  tft.drawCircle(120,210,91,ILI9341_RED);
  tft.drawCircle(120,210,92,ILI9341_RED);
  tft.drawCircle(120,210,60,ILI9341_RED);
  tft.drawCircle(120,210,30,ILI9341_RED);
  tft.drawLine(120,120,120,300,ILI9341_RED);
  tft.drawLine(30,210,210,210,ILI9341_RED);
  tft.setTextSize(2);
  tft.setTextColor(ILI9341_WHITE);
  tft.setCursor(115,95);
  tft.print("N");
  tft.setCursor(220,200);
  tft.print("E");
  tft.setCursor(115,304);
  tft.print("S");
  tft.setCursor(10,200);
  tft.print("W");
    // ************************************************************Feld für Richtungsausgabe
  tft.drawRect(60, 10, 120, 55, ILI9341_RED);
  tft.fillRect(61, 11, 118, 53, ILI9341_YELLOW);
  tft.setTextColor(ILI9341_BLACK);
    // ********************************************************************Smeter Balken Feld
    //tft.drawRect(20, 70, 200, 15, ILI9341_RED);  nur bei S-Meterausgabe
    tft.setTextColor(ILI9341_WHITE);
  tft.setTextSize(1);
  tft.setCursor(9,295);
  tft.print("DK5BD");
  tft.setCursor(175,295);
  tft.print("06.05.2020");
 }
```

```
 //*************************************************************************Schleife
void loop(void) {
  tft.drawLine(120,120,120,300,ILI9341_RED);                        //Kreuz zeichnen
  tft.drawLine(30,210,210,210,ILI9341_RED);
  tft.setTextColor(ILI9341_WHITE);
  tft.drawCircle(120,210,60,ILI9341_RED);
  tft.drawCircle(120,210,30,ILI9341_RED);
  // ***********************************************************S-Meterbalken
    /*tft.fillRect(21, 71, 190, 13, ILI9341_BLACK);
    Smeter=analogRead(smeterPin);
    Smeter=Smeter/(1023/160);
    tft.fillRect(21, 71, Smeter, 12, ILI9341_YELLOW);*/
  // ***********************************************************Richtungspoti auslesen
  Richtung=analogRead(richtungPin);
  if(Richtung!=Altrichtung){
    Altrichtung=Richtung;
    Serial.print(Richtung,0);                                       //AD Ergebnis (0-1023)
    Serial.print("  ");

    // ***************************************************1023 bei 5V. Hier Emotatoanpassung
    Richtung = Richtung / 872;
    Richtung = Richtung * 360;
    Richtung = Richtung -90;
    if(Richtung+270>=360){
      Richtung=Richtung-360;
    }
    // ***********************************************************Richtungsstrich löschen
    YY=120-cos(RichtungAlt*3.14/180)*90;
    XX=210-sin(RichtungAlt*3.14/180)*90;
    tft.drawLine(120,210,YY,XX,ILI9341_BLACK);
    // ***********************************************************neuen Richtungsstrich ziehen
    YY=120-cos(Richtung*3.14/180)*90;
    XX=210-sin(Richtung*3.14/180)*90;
    tft.drawLine(120,210,YY,XX,ILI9341_YELLOW);
    RichtungAlt=Richtung;

    // ***********************************************************Richtung drucken
    tft.drawRect(60, 10, 120, 55, ILI9341_RED);
    tft.fillRect(61, 11, 118, 53, ILI9341_YELLOW);
    tft.setTextColor(ILI9341_BLACK);
    tft.setCursor(80,20);
    tft.setTextSize(5);
    if(Richtung + 270<=100){
      tft.print("0");
    }
    if(Richtung + 270<=10){
      tft.print("0");
    }
    tft.print(Richtung+270,0);


  }
}
```