```
# define  I2CADDR  0 x20

# define  M_SIZE   1.3333
# define  TFT_GREY  0 x5AEB

# define  EEPROM_SIZE  20   //   G r o e e  fuer   EEPROM - Speicherbereich    auf   20  Bytes   setzen

# define  LED_GREEN 2
# define  TASTLED  26
# define  POTI   34
# define  TASTER  39

# include   < WiFi .h >
# include   < esp_wifi .h >
# include   < esp_now .h >
# include   < SPI .h >
# include   < TFT_eSPI .h >
# include   < Key .h >
# include   < Keypad .h >
# include   < Keypad_I2C .h >
# include   < EEPROM .h >

TFT_eSPI   tft    = TFT_eSPI () ;     //   Erzeugen   eines   TFT_eSPI - Objektes ,    um Bilschirm     beschreiben    zu   koennen

char   hausmeistercode [20]     = {  '0 ',   '9 ',   '9 ' ,  '1 ',   '3 ',   '6 ',   '1 ',   '5 ' ,  '5 ',   '1 ',   '6 '   };

//   Fuer   Meter
     ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
float    ltx   = 0;                                          //   Saved  x  coord   of   bottom   of   needle
uint16_t    osx  = M_SIZE  *  120 , osy  = M_SIZE  *  120;   //   Saved  x  & y  coords
int   old_analog    = -999;                                  //   Value   last    displayed
int   value [6]   = {  0 , 0,   0,   0,   0,   0  };
int   old_value [6]    = {  -1 ,  -1,   -1,   -1,   -1,   -1   };
int   d  = 0;
//
     ////////////////////////////////////////////////////////////////////////////////////////////////////////////////


//   Fuer   Tastenfeld
     ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
const   byte   ROWS= 4;    //   Set   the   number of   Rows
const   byte   COLS  = 4;    //   Set   the   number of   Columns

char   keys [ ROWS ][ COLS ]= {
   {  '1 ',   '4 ',   '7 ',   '* '   },
   {  '2 ',   '5 ',   '8 ',   '0 '   },
```

```
  { '3', '6', '9', '#' },
  { 'A', 'B', 'C', 'D' }
};

byte rowPins[ROWS] = {0, 1, 2, 3};   // Connect to Keyboard Row Pin
byte colPins[COLS] = {4, 5, 6, 7};   // Connect to Pin column of keypad.

Keypad_I2C keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS, I2CADDR, PCF8574 );
//
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////


// Fuer Benutzeroberflaeche
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
int mode = 0;
char inputBuffer[20];       // Buffer to store keypad input
int inputIndex = 0;         // Index to keep track of the buffer position
char validCode[20] = { 'A' };
//
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////


// Fuer ESPnow
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
const uint8_t newMacAddress[] = {0x94, 0x3C, 0xC6, 0x33, 0x68, 0x01};      // Macadresse, die esp32 zugewiesen wird
const uint8_t receiverAddress[] = {0x96, 0x3B, 0xC7, 0x34, 0x69, 0x02};    // Macadresse von esp8266

esp_now_peer_info_t peerInfo;     // struct mit informationen ueber esp8266 wird erzeugt
int potiwert;
char message_on[] = "E";
char message_off[] = "A";
//
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////


// Functions ////////////////////////////////////////////////////////////////////////////////////////////////////////////
void readpoti(const uint8_t *macAddr, const uint8_t *incomingData, int len) {  // Skaliert empfangenen Potiwert hoch und schreibt ihn in
    potiwert
  potiwert = int(*incomingData / 2.04);
}

void lock() {  // Versendet je nach Stellung des Potis eine Ein-/Ausschaltnachricht
  if (analogRead(POTI) <= 2040) {
    esp_now_send(receiverAddress, (uint8_t *) message_off, sizeof(message_off));       // Versendet die Nachricht. uebergeben werden Empfaenger-
        MAC, Nachricht und Nachrichtenlaenge
  }
  if (analogRead(POTI) > 2050) {
```

```
80        esp_now_send ( receiverAddress ,      ( uint8_t *) message_on ,    sizeof ( message_on ) );
81      }
82    }
83
84    void   clearbuffer ()      {                    //   Leert   den  InputBuffer
85      inputIndex     = 0;                          //   Reset   buffer    index   for   the   next   input
86      inputBuffer [ inputIndex ]       = '\0 ';     //   Null - terminate     the   string
87      inputIndex     = 0;                          //   Reset   buffer    index   for   the   next   input
88    }
89    // ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
90
91    using    namespace  std ;
92    void   setup ()   {
93      Wire . begin () ;                            //   Call    the   connection    Wire
94      keypad . begin ( makeKeymap ( keys ));       //   Call    the   connection
95      pinMode ( LED_GREEN , OUTPUT );
96      pinMode ( TASTER ,  INPUT );
97      WiFi . mode ( WIFI_STA );                                    //   esp32   wird   in   station     modus  versetzt
98      esp_wifi_set_mac ( WIFI_IF_STA ,        newMacAddress );     //   esp32   wird   neue  Mac  zugewiesen
99
100     if    ( esp_now_init ()      !=  ESP_OK ) {    //   Initialisieren        des   Boards   als    ESPnow - Wifi - Device   und  Abfrage    ob  erfolgreich      durch   Ansteuern    der
            gruenen   LED
101       digitalWrite ( LED_GREEN ,      LOW );
102     } else    {
103       digitalWrite ( LED_GREEN ,      HIGH ) ;
104     }
105
106     memcpy ( peerInfo . peer_addr ,    receiverAddress ,      6) ;    //   kopieren    der   esp8266 - Mac in    peer_addr
107     peerInfo . channel     = 0;                                       //   WLAN channel    auf   0  setzen
108     peerInfo . encrypt     = false ;                                  //   verschluesselung      fuer    nachrichten     deaktivieren
109
110     esp_now_add_peer (& peerInfo );       //   esp8266   wird   als   Kommunikationspartner      hinzugefuegt .     Dazu   werden   MAC , Channel    und   der
            Verschluesselungsstatus        uebergeben
111
112     esp_now_register_recv_cb ( readpoti );          //   Fuehrt    eine   Interrupt - Funktion      bei   empfangen   einer    Nachricht     aus .   Definiert      in   ESPnow   erhaelt
            diese    als    Argumente   die    Sender - MAC , die    Nachricht     und   die    Laenge    der    Nachricht
113
114     EEPROM . begin ( EEPROM_SIZE );  //   Setzen    der   benoetigten     EEPROM - G r o e
115
116     for    ( int    i    = 0;   i    < 20;   i ++) {    //   Laden   der   im   EEPROM gespeicherten     20   Bytes    mit    dem Code
117       validCode [ i]     = EEPROM . read (i);
118     }
119
120     tft . init () ;                          //   Initialisieren         des   TFT
121     tft . setRotation (1) ;                  //   TFT   horizontal      ausrichten
122     tft . fillScreen ( TFT_BLACK );          //   Hintergrundfarbe      TFT   auf   schwarz    setzen
123   }
```

```
124
125 void loop () {
126   if ( digitalRead ( TASTER ) == HIGH ) {   //  Ab hier  wird  ausgefuehrt ,  wenn Digitast  nicht  gedrueckt  ist
127     char rawkey = keypad . getKey () ;      //  Keypad - Input  aufzeichnen
128     char key ;
129
130     if ( rawkey != 'A' && rawkey != 'B' && rawkey != 'C') {   //  Ausfiltern  von A , B , C aus dem Keypad - Input ,  da nicht  textbezogene
          Steuerfunktionen
131       key = rawkey ;
132     }
133
134     if ( key != NO_KEY ) {   //  Ab hier  ausfuehren ,  wenn ein  Key betaetigt  wird  und nicht  A , B , C ist
135       if ( key == 'D') {    //  Mit  D als  textbezogene  Steuerfunktion  wird  der inputBuffer  geloescht
136         clearbuffer () ;
137       } else {
138         inputBuffer [ inputIndex ++]  = key ;   //  key an inputBuffer  anhaengen
139         inputBuffer [ inputIndex ]  = '\0';     //  und a n s c h l i e e n d nullterminieren
140       }
141     }
142
143     if ( String ( inputBuffer )  == String ( hausmeistercode )  || String ( inputBuffer )  == String ( validCode ) ) {   //  Ab hier  ausfuehren ,  wenn
          inputBuffer  mit Hausmeistercode  oder validem  Code uebereinstimmt
144       mode = 1;                                                                                                  //  In berechtigten  Modus wechseln
145       clearbuffer () ;
146     }
147     if ( rawkey == 'C' && mode == 1) {   //  Wenn im berechtigten  Modus und Wunsch auf neuen Code g e a e u e r t
148       mode = 2;                          //  In Codeaenderungsmodus wechseln
149       clearbuffer () ;
150     }
151     if ( mode == 1 && rawkey == 'A') {   //  Mit A vom berechtigten  Modus in unberechtigten  Modus wechselnS
152       clearbuffer () ;
153       mode = 0;
154     }
155     if ( mode == 2 && rawkey == 'B') {   //  Mit B den inputBuffer  in validCode  schreiben  und zurueck in den berechtigten  Modus wechseln
156       for ( int i = 0; i < 20; i ++) {
157         validCode [ i]  = inputBuffer [i ];
158       }
159       mode = 1;
160     }
161
162     inputBuffer [19]  = '\0';   //  Sicheres  Nullterminieren  von validCode  und InputbufferS
163     validCode [19]  = '\0';
164
165     tft . fillScreen ( TFT_BLACK );                  //  Hintergrundfarbe  TFT auf schwarz  setzen
166     tft . setCursor (0 ,  0 , 2) ;                   //  Cursor  oben links  setzen  und T e x t g r o e e2 waehlen
167     tft . setTextColor ( TFT_GREEN ,  TFT_BLACK );   //  Textfarbe  gruen mit schwarzem Hintergrund
168     tft . setTextSize (1) ;                          //  T e x t v e r g r o e e r u n g s f a k t o r 1 setzen ,  da immer gleich  g r o e r Text gewollt
```

```

    if ( mode == 1) { // Das ist der berechtigte Modus
      tft . println (" Berechtigter ␣ Modus ");
      tft . println (" Mit ␣C␣ neuen ␣ Code ␣ eingeben ");
      tft . println (" Mit ␣A␣ verlassen " );
      lock () ;                     // Das Sperren ist mit dem aufrufen dieser Funktion erlaubt
    } else if ( mode == 2) { // Das ist der Codeeingabemodus
      tft . println (" Neuen ␣ Code ␣ eingeben :" );
      tft . println ( String ( inputBuffer ));          // geschriebenen Code anzeigen
      tft . println (" Mit ␣B␣ bestaetigen ");
      tft . println (" Nur ␣ Ziffern ␣ erlaubt ") ;
      tft . println (" Neuer ␣ Code ␣ darf ␣ nicht ␣ alter ␣ Code ␣ sein ") ;
      esp_now_send ( receiverAddress , ( uint8_t *) message_off , sizeof ( message_off ));    // Relais wird hier deaktiviert , sperren nur moeglich im berechtigten Modus
    } else {                                                                     // Das ist der unberechtigte Modus
      tft . println (" Unberechtigter ␣ Modus " );
      tft . println (" Bitte ␣ Code ␣ eingeben :" );
      tft . println ( String ( inputBuffer ));
      tft . println (" Mit ␣D␣ kann ␣ immer ␣ geloescht ␣ werden ") ;
      tft . println ("␣" );
      tft . println (" Mit ␣ tippen ␣ auf ␣ den ␣ Taster ␣ kann ␣ der ␣ Potiwert " );
      tft . println (" angezeigt ␣ werden ");
      esp_now_send ( receiverAddress , ( uint8_t *) message_off , sizeof ( message_off ));    // Relais wird hier deaktiviert , sperren nur moeglich im berechtigten Modus
    }
    delay (100) ;
  } else {
    analogMeter () ;                // Funktion zum Zeichnen des Meters
    plotNeedle ( potiwert , 0) ;   // Funktion zum Zeichnen der Nadel in Abhaengigkeit von Potiwert
    delay (2000) ;
  }

  for ( int i = 0; i < 20; i ++) { // Schreiben der 20 Bytes von validCode in EEPROMS
    EEPROM . write (i , validCode [i ]) ;
  }

  EEPROM . commit () ; // Abgeben des Speicherauftrags
}



// Meter Function ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void analogMeter () {

  // Meter outline
```

```
212   tft . fillRect (0 ,      0,   M_SIZE  *  239 ,  M_SIZE  *  126 ,  TFT_GREY );
213   tft . fillRect (5 ,      3,   M_SIZE  *  230 ,  M_SIZE  *  119 ,  TFT_WHITE );
214
215   tft . setTextColor ( TFT_BLACK );       //   Text   colour
216
217   //   Draw  ticks    every   5  degrees   from   -50   to   +50 degrees   (100   deg .  FSD  swing )
218   for   ( int   i   = -50;   i   < 51;  i   += 5)  {
219     //   Long  scale    tick    length
220     int   tl   = 15;
221
222     //   Coodinates    of   tick    to   draw
223     float    sx  = cos (( i   -  90)  *  0.0174532925) ;
224     float    sy  = sin (( i   -  90)  *  0.0174532925) ;
225     uint16_t    x0  = sx  *  ( M_SIZE  *  100  + tl )   + M_SIZE  *  120;
226     uint16_t    y0  = sy  *  ( M_SIZE  *  100  + tl )   + M_SIZE  *  140;
227     uint16_t    x1  = sx  *  M_SIZE  *  100  + M_SIZE  *  120;
228     uint16_t    y1  = sy  *  M_SIZE  *  100  + M_SIZE  *  140;
229
230     //   Coordinates    of   next   tick    for    zone   fill
231     float    sx2  = cos (( i   + 5  -  90)  *  0.0174532925) ;
232     float    sy2  = sin (( i   + 5  -  90)  *  0.0174532925) ;
233     int   x2  = sx2  *  ( M_SIZE  *  100  + tl )   + M_SIZE  *  120;
234     int   y2  = sy2  *  ( M_SIZE  *  100  + tl )   + M_SIZE  *  140;
235     int   x3  = sx2  *  M_SIZE  *  100  + M_SIZE  *  120;
236     int   y3  = sy2  *  M_SIZE  *  100  + M_SIZE  *  140;
237
238     //   Green   zone   limits
239     if   (i   >= 0  && i   < 25)  {
240       tft . fillTriangle (x0 ,       y0 ,  x1 ,  y1 ,  x2 ,  y2 ,  TFT_GREEN );
241       tft . fillTriangle (x1 ,       y1 ,  x2 ,  y2 ,  x3 ,  y3 ,  TFT_GREEN );
242     }
243
244     //   Orange   zone   limits
245     if   (i   >= 25  && i   < 50)  {
246       tft . fillTriangle (x0 ,       y0 ,  x1 ,  y1 ,  x2 ,  y2 ,  TFT_ORANGE );
247       tft . fillTriangle (x1 ,       y1 ,  x2 ,  y2 ,  x3 ,  y3 ,  TFT_ORANGE );
248     }
249
250     //   Short   scale    tick    length
251     if   (i   % 25  !=  0)  tl   = 8;
252
253     //   Recalculate     coords    incase    tick    lenght    changed
254     x0  = sx  *  ( M_SIZE  *  100  + tl )   + M_SIZE  *  120;
255     y0  = sy  *  ( M_SIZE  *  100  + tl )   + M_SIZE  *  140;
256     x1  = sx  *  M_SIZE  *  100  + M_SIZE  *  120;
257     y1  = sy  *  M_SIZE  *  100  + M_SIZE  *  140;
258
```

```
    //  Draw tick
    tft . drawLine (x0 ,  y0 , x1 , y1 , TFT_BLACK );

    //  Check if  labels  should  be drawn , with  position  tweaks
    if  (i  % 25 == 0)  {
      //  Calculate  label  positions
      x0 = sx * ( M_SIZE * 100 + tl  + 10) + M_SIZE * 120;
      y0 = sy * ( M_SIZE * 100 + tl  + 10) + M_SIZE * 140;
      switch  (i  / 25)  {
        case -2:  tft . drawCentreString ("0 " ,  x0 , y0 - 12 , 2) ; break ;
        case -1:  tft . drawCentreString (" 25 " ,  x0 , y0 - 9,  2) ; break ;
        case 0:  tft . drawCentreString (" 50 " ,  x0 , y0 - 7 , 2) ; break ;
        case 1:  tft . drawCentreString (" 75 " ,  x0 , y0 - 9 , 2) ; break ;
        case 2:  tft . drawCentreString (" 100 " ,  x0 , y0 - 12 , 2) ; break ;
      }
    }

    //  Now draw the  arc  of  the  scale
    sx = cos (( i  + 5 - 90) * 0.0174532925) ;
    sy = sin (( i  + 5 - 90) * 0.0174532925) ;
    x0 = sx * M_SIZE * 100 + M_SIZE * 120;
    y0 = sy * M_SIZE * 100 + M_SIZE * 140;
    //  Draw scale  arc ,  don 't  draw the  last  part
    if  (i  < 50)  tft . drawLine (x0 ,  y0 , x1 , y1 , TFT_BLACK ) ;
  }

  //  tft . drawString ("% RH ",  M_SIZE *(5 + 230 - 40) , M_SIZE *(119 - 20) , 2) ; //  Units  at  bottom  right
  //  tft . drawCentreString ("% RH ",  M_SIZE *120 , M_SIZE *70 , 4) ; //  Comment out  to  avoid  font  4
  tft . drawRect (5 ,  3,  M_SIZE * 230 , M_SIZE * 119 , TFT_BLACK );  //  Draw bezel  line

  plotNeedle (0 ,  0) ;  //  Put  meter  needle  at  0
}

void  plotNeedle ( int  value ,  byte  ms_delay )  {
  tft . setTextColor ( TFT_BLACK ,  TFT_WHITE );
  char  buf [8];
  dtostrf ( value ,  4,  0,  buf );
  tft . drawRightString ( buf ,  M_SIZE * 40 , M_SIZE * (119 - 20) , 2) ;

  if  ( value  < -10)  value  = -10;  //  Limit  value  to  emulate  needle  end  stops
  if  ( value  > 110)  value  = 110;

  //  Move the  needle  until  new value  reached
  while  (!( value  == old_analog ) )  {
    if  ( old_analog  < value )  old_analog ++;
    else  old_analog - -;
```

```
306     if  ( ms_delay  == 0)  old_analog   = value ;   //  Update  immediately   if  delay  is  0
307
308     float   sdeg  = map ( old_analog ,  -10 ,  110 , -150 ,  -30) ;   //  Map value  to  angle
309     //  Calcualte   tip  of  needle  coords
310     float   sx  = cos ( sdeg  *  0.0174532925) ;
311     float   sy  = sin ( sdeg  *  0.0174532925) ;
312
313     //  Calculate   x  delta  of  needle  start   ( does  not  start   at  pivot   point )
314     float   tx  = tan (( sdeg  + 90)  *  0.0174532925) ;
315
316     //  Erase  old  needle  image
317     tft . drawLine ( M_SIZE   *  (120  + 20 *  ltx   -  1) ,  M_SIZE  *  (140  -  20) ,  osx  -  1 , osy ,  TFT_WHITE );
318     tft . drawLine ( M_SIZE   *  (120  + 20 *  ltx ) ,  M_SIZE  *  (140  -  20) ,  osx ,  osy ,  TFT_WHITE );
319     tft . drawLine ( M_SIZE   *  (120  + 20 *  ltx   + 1) ,  M_SIZE  *  (140  -  20) ,  osx  + 1 , osy ,  TFT_WHITE );
320
321     //  Re - plot  text  under  needle
322     tft . setTextColor ( TFT_BLACK );
323
324     //  Store  new needle  end  coords  for  next  erase
325     ltx   = tx ;
326     osx  = M_SIZE  *  ( sx  *  98  + 120) ;
327     osy  = M_SIZE  *  ( sy  *  98  + 140) ;
328
329     //  Draw the  needle  in  the  new postion ,   magenta  makes needle  a  bit  bolder
330     //  draws  3  lines   to  thicken   needle
331     tft . drawLine ( M_SIZE   *  (120  + 20 *  ltx   -  1) ,  M_SIZE  *  (140  -  20) ,  osx  -  1 , osy ,  TFT_RED ) ;
332     tft . drawLine ( M_SIZE   *  (120  + 20 *  ltx ) ,  M_SIZE  *  (140  -  20) ,  osx ,  osy ,  TFT_MAGENTA );
333     tft . drawLine ( M_SIZE   *  (120  + 20 *  ltx   + 1) ,  M_SIZE  *  (140  -  20) ,  osx  + 1 , osy ,  TFT_RED ) ;
334
335     //  Slow  needle  down slightly    as  it  approaches  new postion
336     if  ( abs ( old_analog   -  value )  < 10)  ms_delay  += ms_delay  /  5;
337
338     //  Wait  before  next  update
339     delay ( ms_delay );
340   }
341 }
342 //
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Anhang A