Captive Portal mit dem ESP 32

Die positive Resonanz und das allgemeine Interesse zu dem Vorgängerblog über ein Captive Portal, hat mich dazu veranlasst, einen weiteren Sonderblog zu dem Thema schreiben und auf einige Punkte und Wünsche von euch im Einzelnen einzugehen. Unter vielen anderen Detailfragen, auf die ich später noch eingehe, bestand der Wunsch, das Captive Portal nicht nur auf einem ESP8266, sondern auch auf dem ESP32 laufen zu lassen. Dies ist nativ nicht möglich. Dazu müssen einige Aänderungen, speziell der genutzten Libarys, vorgenommen werden. Der heutige Code ist daher NUR auf dem ESP32 lauffähig. Im Detail sind neben den Libaryanpassungen auch einige Optimierungen speziell für den ESP32 hinzugekommen.

Bei einigen Lesern des Blogs ist es darüber hinaus wohl vorgekommen, dass der ESP sich nicht mehr richtig mit einem Access Point reconnecten konnte, nachdem er rebootet wurde. Diesen Bug konnte ich in nach einer längeren Suche nachstellen. Dieser trifft wohl im Zusammenspiel mit bestimmten Access Points auf, bei der ESP trotz richtigen WLAN Zugangsdaten bei der Anmeldung an den Access Point svond diesem abgewiesen wird und folgerichtig vom Station Mode in den eigenen Access Point Mode wechselt. Bei diesem Vorgang werden die vorherigen WLAN Zugangsdaten aus dem EEPROM gelöscht. (Dies ist auch so gewollt)

Die Lösung für diesen systemseitigen Bug ist, die Anmeldung mehrere Male trotz angeblich „falschen“ Zugangsdaten zu versuchen, bevor wieder in den Access Point Mode gewechselt wird.

As weitere kleine Verbesserung wird die aktuelle IP Adresse, egal in welchem Mode sich der ESP32 befindet, auf der seriellen Schnittstelle ausgegeben. Wer also diese IP Adresse gerne auf seinem Display ausgeben möchte, muss nur noch Ausgabe an der entsprechenden Stelle im Code abändern.

Um möglichst mit vielen ESP Boards kompatibel zu sein, verzichte ich weiterhin auf die Ausgabe der Statis auf externe Ports des ESP‘s.

Auch in diesem Teil baut der ESP32 mit unserem nachfolgendem Captive Portal Code ein Captive-Portal auf. Das WLAN hat den Namen „ESP_Config“ und das Passwort „12345678“. Mit diesem können wir uns mit unserem Handy verbinden, und werden dann von dem Handy automatisch auf die Captive Portal Webseite geleitet. Diese entspricht im Design und Funktionsweise im wesentlichem dem vorherigen Teil.

Auf dieser Webseite können wir nun dem System-Link „WiFi Einstellungen“ klicken, und gelangen nun auf eine umfangreiche WLAN Konfigurationsseite, mit der wir nun sowohl ein Netz, mit dem sich der ESP32 verbinden soll, auswählen können:

Das hier ausgewählte Funknetz und das eingegebene Passwort werden im EEPROM gespeichert. Beim nächsten Boot versucht sich der ESP32 mit diesem Netzwerk zu verbinden. Scheitert dies, nach mehrmaligen Versuchen, weil das Netzwerk z.B nicht mehr erreichbar ist, oder das Passwort geändert wurde, schaltet der ESP zurück in den Access Point Modus und wartet auf eine Neukonfiguration.

Der angepasste Code für das Captive Portal für den ESP32 lautet:

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <DNSServer.h>
#include <EEPROM.h>

#define GPIO_OUT_W1TS_REG (DR_REG_GPIO_BASE + 0x0008)
#define GPIO_OUT_W1TC_REG (DR_REG_GPIO_BASE + 0x000c)

static const byte WiFiPwdLen = 25;
static const byte APSTANameLen = 20;

struct WiFiEEPromData
  {
    bool APSTA = true; // Access Point or Station Mode - true AP Mode
    bool PwDReq = false; // PasswordRequired
    bool CapPortal = true ; //CaptivePortal on in AP Mode
    char APSTAName[APSTANameLen]; // STATION /AP Point Name TO
cONNECT, if definded
    char WiFiPwd[WiFiPwdLen]; // WiFiPAssword, if definded
    char ConfigValid[3]; //If Config is Vaild, Tag "TK" is required"
  };

/* hostname for mDNS. Should work at least on windows. Try http://esp8266.local
*/
const char *ESPHostname = "ESP32";

// DNS server
const byte DNS_PORT = 53;
DNSServer dnsServer;

//Conmmon Paramenters
bool SoftAccOK  = false;

// Web server
WebServer server(80);

/* Soft AP network parameters */
IPAddress apIP(172, 20, 0, 1);
IPAddress netMsk(255, 255, 255, 0);
```

```
unsigned long currentMillis = 0;
unsigned long startMillis;

/** Current WLAN status */
short status = WL_IDLE_STATUS;

WiFiEEPromData MyWiFiConfig;
String temp ="";


void setup()
{
  REG_WRITE(GPIO_OUT_W1TS_REG, BIT(GPIO_NUM_16));     // Guru
Meditation Error Remediation set
  delay(1);
  REG_WRITE(GPIO_OUT_W1TC_REG, BIT(GPIO_NUM_16));     // Guru
Meditation Error Remediation clear
  bool ConnectSuccess = false;
  bool CreateSoftAPSucc  = false;
  bool CInitFSSystem  = false;
  bool CInitHTTPServer  = false;
  byte len;
  Serial.begin(9600);
   while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB
  }
  Serial.println(F("Serial Interface initalized at 9600 Baud."));
  WiFi.setAutoReconnect (false);
  WiFi.persistent(false);
  WiFi.disconnect();
  WiFi.setHostname(ESPHostname); // Set the DHCP hostname assigned to ESP
station.
  if (loadCredentials()) // Load WLAN credentials for WiFi Settings
  {
     Serial.println(F("Valid Credentials found."));
     if (MyWiFiConfig.APSTA == true)  // AP Mode
      {
        Serial.println(F("Access Point Mode selected."));
        Serial.println(MyWiFiConfig.APSTA);
        len = strlen(MyWiFiConfig.APSTAName);
        MyWiFiConfig.APSTAName[len+1] = '\0';
        len = strlen(MyWiFiConfig.WiFiPwd);
        MyWiFiConfig.WiFiPwd[len+1] = '\0';
        CreateSoftAPSucc = CreateWifiSoftAP();
      } else
      {
        Serial.println(F("Station Mode selected."));
        len = strlen(MyWiFiConfig.APSTAName);
        MyWiFiConfig.APSTAName[len+1] = '\0';
        len = strlen(MyWiFiConfig.WiFiPwd);
        MyWiFiConfig.WiFiPwd[len+1] = '\0';
      len = ConnectWifiAP();
      if ( len == 3 ) { ConnectSuccess = true; } else { ConnectSuccess = false; }
     }
  } else
  { //Set default Config - Create AP
    Serial.println(F("NO Valid Credentials found."));
    SetDefaultWiFiConfig ();
    CreateSoftAPSucc = CreateWifiSoftAP();
    saveCredentials();
    delay(500);
  }
  if ((ConnectSuccess or CreateSoftAPSucc))
    {
     Serial.print (F("IP Address: "));
     if (CreateSoftAPSucc) { Serial.println(WiFi.softAPIP());}
     if (ConnectSuccess) { Serial.println(WiFi.localIP());}
     InitalizeHTTPServer();
    }
    else
    {
     Serial.setDebugOutput(true); //Debug Output for WLAN on Serial Interface.
     Serial.println(F("Error: Cannot connect to WLAN. Set DEFAULT
Configuration."));
     SetDefaultWiFiConfig();
     CreateSoftAPSucc = CreateWifiSoftAP();
     InitalizeHTTPServer();
     SetDefaultWiFiConfig();
     saveCredentials();
    }
}

void InitalizeHTTPServer()
 {
  bool initok = false;
  /* Setup web pages: root, wifi config pages, SO captive portal detectors and not
found. */
  server.on("/", handleRoot);
  server.on("/wifi", handleWifi);
  if (MyWiFiConfig.CapPortal) { server.on("/generate_204", handleRoot); } //Android
captive portal. Maybe not needed. Might be handled by notFound handler.
  if (MyWiFiConfig.CapPortal) { server.on("/favicon.ico", handleRoot); }   //Another
Android captive portal. Maybe not needed. Might be handled by notFound handler.
Checked on Sony Handy
  if (MyWiFiConfig.CapPortal) { server.on("/fwlink", handleRoot); }  //Microsoft
captive portal. Maybe not needed. Might be handled by notFound handler.
  //server.on("/generate_204", handleRoot);  //Android captive portal. Maybe not
needed. Might be handled by notFound handler.
  //server.on("/favicon.ico", handleRoot);    //Another Android captive portal. Maybe
not needed. Might be handled by notFound handler. Checked on Sony Handy
  //server.on("/fwlink", handleRoot);   //Microsoft captive portal. Maybe not needed.
Might be handled by notFound handler.
```

```
  server.onNotFound ( handleNotFound );
  // Speicherung Header-Elemente anfordern
  // server.collectHeaders(Headers, sizeof(Headers)/ sizeof(Headers[0]));
  server.begin(); // Web server start
 }

boolean CreateWifiSoftAP()
{
  WiFi.disconnect();
  Serial.print(F("Initalize SoftAP "));
  if (MyWiFiConfig.PwDReq)
    {
      SoftAccOK  =  WiFi.softAP(MyWiFiConfig.APSTAName,
MyWiFiConfig.WiFiPwd); // Passwortlänge mindestens 8 Zeichen !
    } else
    {
      SoftAccOK  =  WiFi.softAP(MyWiFiConfig.APSTAName); // Access Point
WITHOUT Password
      // Overload Function:; WiFi.softAP(ssid, password, channel, hidden)
    }
  delay(2000); // Without delay I've seen the IP address blank
  WiFi.softAPConfig(apIP, apIP, netMsk);
  if (SoftAccOK)
  {
  /* Setup the DNS server redirecting all the domains to the apIP */
  dnsServer.setErrorReplyCode(DNSReplyCode::NoError);
  dnsServer.start(DNS_PORT, "*", apIP);
  Serial.println(F("successful."));
  // Serial.setDebugOutput(true); // Debug Output for WLAN on Serial Interface.
  } else
  {
  Serial.println(F("Soft AP Error."));
  Serial.println(MyWiFiConfig.APSTAName);
  Serial.println(MyWiFiConfig.WiFiPwd);
  }
  return SoftAccOK;
}

byte ConnectWifiAP()
{
  Serial.println(F("Initalizing Wifi Client."));
  byte connRes = 0;
  byte i = 0;
  WiFi.disconnect();
  WiFi.softAPdisconnect(true); // Function will set currently configured SSID and
password of the soft-AP to null values. The parameter  is optional. If set to true it
will switch the soft-AP mode off.
  delay(500);
  WiFi.begin(MyWiFiConfig.APSTAName, MyWiFiConfig.WiFiPwd);
  connRes  = WiFi.waitForConnectResult();
  while (( connRes == 0 ) and (i != 10))  //if connRes == 0  "IDLE_STATUS - change
Statius"
    {
      connRes  = WiFi.waitForConnectResult();
      delay(2000);
      i++;
      Serial.print(F("."));
      // statement(s)
    }
  while (( connRes == 1 ) and (i != 10))  //if connRes == 1  NO_SSID_AVAILin -
SSID cannot be reached
    {
      connRes  = WiFi.waitForConnectResult();
      delay(2000);
      i++;
      Serial.print(F("."));
      // statement(s)
    }
  if (connRes == 3 ) {
                    WiFi.setAutoReconnect(true); // Set whether module will attempt to
reconnect to an access point in case it is disconnected.
                    // Setup MDNS responder
                        if (!MDNS.begin(ESPHostname)) {
                            Serial.println(F("Error: MDNS"));
                            } else { MDNS.addService("http", "tcp", 80); }
                   }
  while (( connRes == 4 ) and (i != 10))  //if connRes == 4  Bad Password.
Sometimes happens this with corrct PWD
    {
      WiFi.begin(MyWiFiConfig.APSTAName, MyWiFiConfig.WiFiPwd);
      connRes = WiFi.waitForConnectResult();
      delay(3000);
      i++;
      Serial.print(F("."));
    }
  if (connRes == 4 ) {
                    Serial.println(F("STA Pwd Err"));
                    Serial.println(MyWiFiConfig.APSTAName);
                    Serial.println(MyWiFiConfig.WiFiPwd);
                    WiFi.disconnect();
                   }
  // if (connRes == 6 ) { Serial.println("DISCONNECTED - Not in station mode"); }
  // WiFi.printDiag(Serial);
Serial.println("");
return connRes;
}

#define SD_BUFFER_PIXELS 20
/** Load WLAN credentials from EEPROM */
bool loadCredentials()
{
 bool RetValue;
 EEPROM.begin(512);
 EEPROM.get(0, MyWiFiConfig);
 EEPROM.end();
 if (String(MyWiFiConfig.ConfigValid) = String("TK"))
  {
    RetValue = true;
  } else
  {
    RetValue = false; // WLAN Settings not found.
  }
  return RetValue;
}

/** Store WLAN credentials to EEPROM */

bool saveCredentials()
{
bool RetValue;
// Check logical Errors
RetValue = true;
if  (MyWiFiConfig.APSTA == true ) //AP Mode
  {
   if (MyWiFiConfig.PwDReq and (sizeof(String(MyWiFiConfig.WiFiPwd)) < 8))
    {
      RetValue = false;  // Invalid Config
    }
   if (sizeof(String(MyWiFiConfig.APSTAName)) < 1)
    {
      RetValue = false;  // Invalid Config
    }
  }
if (RetValue)
  {
  EEPROM.begin(512);
  for (int i = 0 ; i < sizeof(MyWiFiConfig) ; i++)
     {
      EEPROM.write(i, 0);
     }
  strncpy( MyWiFiConfig.ConfigValid , "TK", sizeof(MyWiFiConfig.ConfigValid) );
  EEPROM.put(0, MyWiFiConfig);
  EEPROM.commit();
  EEPROM.end();
  }
  return RetValue;
}

void SetDefaultWiFiConfig()
{
   byte len;
   MyWiFiConfig.APSTA = true;
   MyWiFiConfig.PwDReq = true;  // default PW required
   MyWiFiConfig.CapPortal = true;
   strncpy( MyWiFiConfig.APSTAName, "ESP_Config",
sizeof(MyWiFiConfig.APSTAName) );
   len = strlen(MyWiFiConfig.APSTAName);
   MyWiFiConfig.APSTAName[len+1] = '\0';
   strncpy( MyWiFiConfig.WiFiPwd, "12345678", sizeof(MyWiFiConfig.WiFiPwd) );
   len = strlen(MyWiFiConfig.WiFiPwd);
   MyWiFiConfig.WiFiPwd[len+1] = '\0';
   strncpy( MyWiFiConfig.ConfigValid, "TK", sizeof(MyWiFiConfig.ConfigValid) );
   len = strlen(MyWiFiConfig.ConfigValid);
   MyWiFiConfig.ConfigValid[len+1] = '\0';
   Serial.println(F("Reset WiFi Credentials."));
}

void handleRoot() {
//  Main Page:
 temp = "";
 byte PicCount = 0;
 byte ServArgs = 0;
  // HTML Header
  server.sendHeader("Cache-Control", "no-cache, no-store, must-revalidate");
  server.sendHeader("Pragma", "no-cache");
  server.sendHeader("Expires", "-1");
  server.setContentLength(CONTENT_LENGTH_UNKNOWN);
// HTML Content
  server.send ( 200, "text/html", temp );   // Speichersparen - Schon mal dem Cleint
senden
  temp = "";
  temp += "<!DOCTYPE HTML><html lang='de'><head><meta charset='UTF-
8'><meta name= viewport content='width=device-width, initial-scale=1.0,'>";
  server.sendContent(temp);
  temp = "";
  temp += "<style type='text/css'><!-- DIV.container { min-height: 10em; display:
table-cell; vertical-align: middle }.button {height:35px; width:90px; font-size:16px}";
  server.sendContent(temp);
  temp = "";
  temp += "body {background-color: powderblue;}</style>";
  temp += "<head><title>Hauptseite</title></head>";
  temp += "<h2>Hauptseite</h2>";
  temp += "<body>";
  server.sendContent(temp);
  temp = "";
// Processing User Request
  temp = "";
  temp += "<table border=2 bgcolor = white width = 500 cellpadding =5
><caption><p><h3>Systemlinks:</h2></p></caption>";
  temp += "<tr><th><br>";
  temp += "<a href='/wifi'>WIFI Einstellungen</a><br><br>";
  temp += "</th></tr></table><br><br>";
  temp += "<footer><p>Programmed and designed by: Tobias
Kuch</p><p>Contact information: <a
href='mailto:tobias.kuch@googlemail.com'>tobias.kuch@googlemail.com</a>.</p>
</footer>";
  temp += "</body></html>";
  server.sendContent(temp);
  temp = "";
  server.client().stop(); // Stop is needed because we sent no content length
}

void handleNotFound() {
   if (captivePortal())
     { // If caprive portal redirect instead of displaying the error page.
       return;
     }
   temp = "";
   // HTML Header
   server.sendHeader("Cache-Control", "no-cache, no-store, must-revalidate");
   server.sendHeader("Pragma", "no-cache");
   server.sendHeader("Expires", "-1");
   server.setContentLength(CONTENT_LENGTH_UNKNOWN);
   // HTML Content
   temp += "<!DOCTYPE HTML><html lang='de'><head><meta charset='UTF-
8'><meta name= viewport content='width=device-width, initial-scale=1.0,'>";
   temp += "<style type='text/css'><!-- DIV.container { min-height: 10em; display:
table-cell; vertical-align: middle }.button {height:35px; width:90px; font-size:16px}";
   temp += "body {background-color: powderblue;}</style>";
   temp += "<head><title>File not found</title></head>";
   temp += "<h2> 404 File Not Found</h2><br>";
   temp += "<h4>Debug Information:</h4><br>";
   temp += "<body>";
   temp += "URI: ";
   temp += server.uri();
   temp += "\nMethod: ";
   temp+= ( server.method() == HTTP_GET ) ? "GET" : "POST";
   temp += "<br>Arguments: ";
   temp += server.args();
   temp += "\n";
     for ( uint8_t i = 0; i < server.args(); i++ ) {
       temp += " " + server.argName ( i ) + ": " + server.arg ( i ) + "\n";
       }
   temp += "<br>Server Hostheader: "+ server.hostHeader();
   for ( uint8_t i = 0; i < server.headers(); i++ ) {
       temp += " " + server.headerName ( i ) + ": " + server.header ( i ) + "\n<br>";
       }
   temp += "</table></form><br><br><table border=2 bgcolor = white width = 500
cellpadding =5 ><caption><p><h2>You may want to browse
to:</h2></p></caption>";
   temp += "<tr><th>";
   temp += "<a href='/'>Main Page</a><br>";
   temp += "<a href='/wifi'>WIFI Settings</a><br>";
```

```
    temp += "</th></tr></table><br><br>";
    temp += "<footer><p>Programmed by: Tobias Kuch</p><p>Contact information:
<a
href='mailto:tobias.kuch@googlemail.com'>tobias.kuch@googlemail.com</a>.</p>
</footer>";
    temp += "</body></html>";
    server.send ( 404, "", temp );
    server.client().stop(); // Stop is needed because we sent no content length
    temp = "";

}

/** Redirect to captive portal if we got a request for another domain. Return true in
that case so the page handler do not try to handle the request again. */
boolean captivePortal() {
  if (!isIp(server.hostHeader()) && server.hostHeader() !=
(String(ESPHostname)+".local")) {
    // Serial.println("Request redirected to captive portal");
    server.sendHeader("Location", String("http://") +
toStringIp(server.client().localIP()), true);
    server.send ( 302, "text/plain", ""); // Empty content inhibits Content-length
header so we have to close the socket ourselves.
    server.client().stop(); // Stop is needed because we sent no content length
    return true;
  }
  return false;
}

/** Wifi config page handler */
void handleWifi()
 {
  //  Page: /wifi
  byte i;
  byte len ;
  temp = "";
  // Check for Site Parameters
      if (server.hasArg("Reboot") )  // Reboot System
       {
         temp = "Rebooting System in 5 Seconds..";
         server.send ( 200, "text/html", temp );
         delay(5000);
         server.client().stop();
         WiFi.disconnect();
         delay(1000);
       }

      if (server.hasArg("WiFiMode") and (server.arg("WiFiMode") == "1")  )  // STA
Station Mode Connect to another WIFI Station
       {
        startMillis = millis(); // Reset Time Up Counter to avoid Idle Mode whiole
operating
```

```
      // Connect to existing STATION
      if ( sizeof(server.arg("WiFi_Network")) > 0  )
       {
         Serial.println("STA Mode");
         MyWiFiConfig.APSTA = false; // Access Point or Station Mode - false
Station Mode
         temp = "";
         for ( i = 0; i < APSTANameLen;i++) { MyWiFiConfig.APSTAName[i] =  0; }
         temp = server.arg("WiFi_Network");
         len =  temp.length();
         for ( i = 0; i < len;i++)
         {
              MyWiFiConfig.APSTAName[i] =  temp[i];
         }
         temp = "";

         for ( i = 0; i < WiFiPwdLen;i++)  { MyWiFiConfig.WiFiPwd[i] =  0; }
         temp = server.arg("STAWLanPW");
         len =  temp.length();
         for ( i = 0; i < len;i++)
           {
             if (temp[i] > 32) //Steuerzeichen raus
               {
                MyWiFiConfig.WiFiPwd[i] =  temp[i];
               }
           }
         temp = "WiFi Connect to AP: -";
         temp += MyWiFiConfig.APSTAName;
         temp += "-<br>WiFi PW: -";
         temp += MyWiFiConfig.WiFiPwd;
         temp += "-<br>";
         temp += "Connecting to STA Mode in 2 Seconds..<br>";
         server.send ( 200, "text/html", temp );
         server.sendContent(temp);
         delay(2000);
         server.client().stop();
         server.stop();
         temp = "";
         WiFi.disconnect();
         WiFi.softAPdisconnect(true);
         delay(500);
        // ConnectWifiAP
        bool SaveOk = saveCredentials();
         i = ConnectWifiAP();
         delay(700);
         if (i != 3) // 4: WL_CONNECT_FAILED - Password is incorrect 1:
WL_NO_SSID_AVAILin - Configured SSID cannot be reached
           {
             Serial.print(F("Cannot Connect to specified Network. Reason: "));
             Serial.println(i);
             server.client().stop();
```

```
              delay(100);
              WiFi.setAutoReconnect (false);
              delay(100);
              WiFi.disconnect();
              delay(1000);
              SetDefaultWiFiConfig();
              CreateWifiSoftAP();
              return;
            } else
            {
              // Safe Config
              bool SaveOk = saveCredentials();
              InitalizeHTTPServer();
              return;
            }
         }
       }

      if (server.hasArg("WiFiMode") and (server.arg("WiFiMode") == "2")  )  //
Change AP Mode
       {
        startMillis = millis(); // Reset Time Up Counter to avoid Idle Mode whiole
operating
        // Configure Access Point
        temp = server.arg("APPointName");
        len =  temp.length();
        temp =server.arg("APPW");
        if (server.hasArg("PasswordReq"))
           {
            i =  temp.length();
           } else { i = 8; }

        if (  ( len > 1 ) and (server.arg("APPW") == server.arg("APPWRepeat")) and ( i
> 7)         )
          {
           temp = "";
           Serial.println(F("APMode"));
           MyWiFiConfig.APSTA = true; // Access Point or Sation Mode - true AP
Mode

           if (server.hasArg("CaptivePortal"))
           {
             MyWiFiConfig.CapPortal = true ; //CaptivePortal on in AP Mode
           } else { MyWiFiConfig.CapPortal = false ; }

           if (server.hasArg("PasswordReq"))
           {
             MyWiFiConfig.PwDReq = true ; //Password Required in AP Mode
           } else { MyWiFiConfig.PwDReq = false ; }

           for ( i = 0; i < APSTANameLen;i++) { MyWiFiConfig.APSTAName[i] =  0; }
           temp = server.arg("APPointName");
           len =  temp.length();
           for ( i = 0; i < len;i++) { MyWiFiConfig.APSTAName[i] =  temp[i]; }
           MyWiFiConfig.APSTAName[len+1] = '\0';
           temp = "";
           for ( i = 0; i < WiFiPwdLen;i++)  {  MyWiFiConfig.WiFiPwd[i] =  0; }
           temp = server.arg("APPW");
           len =  temp.length();
           for ( i = 0; i < len;i++)  { MyWiFiConfig.WiFiPwd[i] =  temp[i];  }
           MyWiFiConfig.WiFiPwd[len+1] = '\0';
           temp = "";
           if (saveCredentials()) // Save AP ConfigCongfig
             {
                     temp = "Daten des AP Modes erfolgreich gespeichert. Reboot
notwendig.";
             } else  { temp = "Daten des AP Modes fehlerhaft.";  }
         } else if (server.arg("APPW") != server.arg("APPWRepeat"))
              {
                temp = "";
                temp = "WLAN Passwort nicht gleich. Abgebrochen.";
              } else
              {
                temp = "";
                temp = "WLAN Passwort oder AP Name zu kurz. Abgebrochen.";
              }
       }

   // HTML Header
   server.sendHeader("Cache-Control", "no-cache, no-store, must-revalidate");
   server.sendHeader("Pragma", "no-cache");
   server.sendHeader("Expires", "-1");
   server.setContentLength(CONTENT_LENGTH_UNKNOWN);
// HTML Content
   temp += "<!DOCTYPE HTML><html lang='de'><head><meta charset='UTF-
8'><meta name= viewport content='width=device-width, initial-scale=1.0,'>";
   server.send ( 200, "text/html", temp );
   temp = "";
   temp += "<style type='text/css'><!-- DIV.container { min-height: 10em; display:
table-cell; vertical-align: middle }.button {height:35px; width:90px; font-size:16px}";
   temp += "body {background-color: powderblue;}</style><head><title>Smartes
Tuerschild - WiFi Settings</title></head>";
   server.sendContent(temp);
   temp = "";
   temp += "<h2>WiFi Einstellungen</h2><body><left>";
   temp += "<table border=2 bgcolor = white width = 500 ><td><h4>Current WiFi
Settings: </h4>";
   if (server.client().localIP() == apIP) {
      temp += "Mode : Soft Access Point (AP)<br>";
      temp += "SSID : " + String (MyWiFiConfig.APSTAName) + "<br><br>";
   } else {
      temp += "Mode : Station (STA) <br>";
```

```
    temp += "SSID  :  "+ String (MyWiFiConfig.APSTAName) + "<br>";
    temp += "BSSID :  " + WiFi.BSSIDstr()+ "<br><br>";
  }
  temp += "</td></table><br>";
  server.sendContent(temp);
  temp = "";
  temp += "<form action='/wifi' method='post'>";
  temp += "<table border=2 bgcolor = white width = 500><tr><th><br>";
  if (MyWiFiConfig.APSTA == 1)
   {
     temp += "<input type='radio' value='1' name='WiFiMode' > WiFi Station
Mode<br>";
   } else
   {
     temp += "<input type='radio' value='1' name='WiFiMode' checked > WiFi
Station Mode<br>";
   }
  temp += "Available WiFi Networks:<table border=2 bgcolor = white
></tr></th><td>Number </td><td>SSID  </td><td>Encryption </td><td>WiFi
Strength </td>";
  server.sendContent(temp);
  temp = "";
  WiFi.scanDelete();
  int n = WiFi.scanNetworks(false, false); //WiFi.scanNetworks(async,
show_hidden)
  if (n > 0) {
    for (int i = 0; i < n; i++) {
    temp += "</tr></th>";
    String Nrb = String(i);
    temp += "<td>" + Nrb + "</td>";
    temp += "<td>" + WiFi.SSID(i) +"</td>";

    Nrb = GetEncryptionType(WiFi.encryptionType(i));
    temp += "<td>"+ Nrb + "</td>";
    temp += "<td>" + String(WiFi.RSSI(i)) + "</td>";
    }
  } else {
    temp += "</tr></th>";
    temp += "<td>1 </td>";
    temp += "<td>No WLAN found</td>";
    temp += "<td> --- </td>";
    temp += "<td> --- </td>";
  }
  temp += "</table><table border=2 bgcolor = white ></tr></th><td>Connect to
WiFi SSID: </td><td><select name='WiFi_Network' >";
if (n > 0) {
    for (int i = 0; i < n; i++) {
    temp += "<option value='" + WiFi.SSID(i) +"'>" + WiFi.SSID(i) +"</option>";
    }
  } else {
    temp += "<option value='No_WiFi_Network'>No WiFiNetwork found !/option>";
  }
  server.sendContent(temp);
  temp = "";
  temp += "</select></td></tr></th></tr></th><td>WiFi Password: </td><td>";
  temp += "<input type='text' name='STAWLanPW' maxlength='40' size='40'>";
  temp += "</td></tr></th><br></th></tr></table></table><table border=2 bgcolor =
white width = 500 ><tr><th><br>";
  server.sendContent(temp);
  temp = "";
  if (MyWiFiConfig.APSTA == true)
    {
      temp += "<input type='radio' name='WiFiMode' value='2' checked> WiFi
Access Point Mode <br>";
    } else
    {
      temp += "<input type='radio' name='WiFiMode' value='2' > WiFi Access Point
Mode <br>";
    }
  temp += "<table border=2 bgcolor = white ></tr></th> <td>WiFi Access Point
Name: </td><td>";
  server.sendContent(temp);
  temp = "";
  if (MyWiFiConfig.APSTA == true)
    {
      temp += "<input type='text' name='APPointName'
maxlength='"+String(APSTANameLen-1)+"' size='30' value='" +
String(MyWiFiConfig.APSTAName) + "'></td>";
    } else
    {
      temp += "<input type='text' name='APPointName'
maxlength='"+String(APSTANameLen-1)+"' size='30' ></td>";
    }
  server.sendContent(temp);
  temp = "";
  if (MyWiFiConfig.APSTA == true)
    {
      temp += "</tr></th><td>WiFi Password: </td><td>";
      temp += "<input type='password' name='APPW'
maxlength='"+String(WiFiPwdLen-1)+"' size='30' value='" +
String(MyWiFiConfig.WiFiPwd) + "'> </td>";
      temp += "</tr></th><td>Repeat WiFi Password: </td>";
      temp += "<td><input type='password' name='APPWRepeat'
maxlength='"+String(WiFiPwdLen-1)+"' size='30' value='" +
String(MyWiFiConfig.WiFiPwd) + "'> </td>";
    } else
    {
      temp += "</tr></th><td>WiFi Password: </td><td>";
      temp += "<input type='password' name='APPW'
maxlength='"+String(WiFiPwdLen-1)+"' size='30'> </td>";
      temp += "</tr></th><td>Repeat WiFi Password: </td>";
      temp += "<td><input type='password' name='APPWRepeat'
maxlength='"+String(WiFiPwdLen-1)+"' size='30'> </td>";
    }
      temp += "</table>";
  server.sendContent(temp);
  temp = "";
  if (MyWiFiConfig.PwDReq)
    {
      temp += "<input type='checkbox' name='PasswordReq' checked> Password for
Login required. ";
    } else
    {
      temp += "<input type='checkbox' name='PasswordReq' > Password for Login
required. ";
    }
  server.sendContent(temp);
  temp = "";
  if (MyWiFiConfig.CapPortal)
    {
      temp += "<input type='checkbox' name='CaptivePortal' checked> Activate
Captive Portal";
    } else
    {
      temp += "<input type='checkbox' name='CaptivePortal' > Activate Captive
Portal";
    }
  server.sendContent(temp);
  temp = "";
  temp += "<br></tr></th></table><br> <button type='submit' name='Settings'
value='1' style='height: 50px; width: 140px' autofocus>Set WiFi Settings</button>";
  temp += "<button type='submit' name='Reboot' value='1' style='height: 50px;
width: 200px' >Reboot System</button>";
  server.sendContent(temp);
  temp = "";
  temp += "<button type='reset' name='action' value='1' style='height: 50px; width:
100px' >Reset</button></form>";
  temp += "<table border=2 bgcolor = white width = 500 cellpadding =5
><caption><p><h3>Systemlinks:</h2></p></caption><tr><th><br>";
  server.sendContent(temp);
  temp = "";
  temp += "<a href='/'>Main Page</a><br><br></th></tr></table><br><br>";
  temp += "<footer><p>Programmed and designed by: Tobias
Kuch</p><p>Contact information: <a
href='mailto:tobias.kuch@googlemail.com'>tobias.kuch@googlemail.com</a>.</p>
</footer>";
  temp += "</body></html>";
  server.sendContent(temp);
  server.client().stop(); // Stop is needed because we sent no content length
  temp = "";
}
```

```
/** Is this an IP? */
boolean isIp(String str) {
  for (int i = 0; i < str.length(); i++) {
    int c = str.charAt(i);
    if (c != '.' && (c < '0' || c > '9')) {
      return false;
    }
  }
  return true;
}

String GetEncryptionType(byte thisType) {
  String Output = "";
   // read the encryption type and print out the name:
   switch (thisType) {
     case 5:
       Output = "WEP";
       return Output;
       break;
     case 2:
       Output = "WPA";
       return Output;
       break;
     case 4:
       Output = "WPA2";
       return Output;
       break;
     case 7:
       Output = "None";
       return Output;
       break;
     case 8:
       Output = "Auto";
       return Output;
      break;
   }
}

/** IP to String? */
String toStringIp(IPAddress ip) {
  String res = "";
  for (int i = 0; i < 3; i++) {
    res += String((ip >> (8 * i)) & 0xFF) + ".";
  }
  res += String(((ip >> 8 * 3)) & 0xFF);
  return res;
}

String formatBytes(size_t bytes) {            // lesbare Anzeige der Speichergrößen
   if (bytes < 1024) {
    return String(bytes) + " Byte";
  } else if (bytes < (1024 * 1024)) {
    return String(bytes / 1024.0) + " KB";
  } else if (bytes < (1024 * 1024 * 1024)) {
    return String(bytes / 1024.0 / 1024.0) + " MB";
  }
 }

void loop()
 {
  if (SoftAccOK)
  {
   dnsServer.processNextRequest(); //DNS
  }
  //HTTP
  server.handleClient();
  }
```

Im nächsten Teil wollen wir uns einmal die praktische Verwendung unseres Codes anschauen, und bauen uns, basierend auf diesem Code einen kleinen Fileserver auf.

Ich wünsche viel Spaß mit dem Captive Portal und bei der Implementierung in eigene ESP32 Projekte.