



Captive Portal Follow-up: BMP Dateien auf 8x8 LED-Dot Matrix Display anzeigen

Hallo und willkommen zu einem weiteren Teil der ESP Captive Portal Reihe. Im heutigen Teil erweitern wir unser Captive Portal mit integriertem Fileserver um eine weitere, sehr interessante Follow-UP-Anwendung: Wir verbinden ein 8x8 WS1812 Matrix mit dem ESP32 und zeigen darauf auf dem Fileserver abgelegte 8x8 Pixel große BMP Dateien an! Aktuell besteht die Limitierung, dass die BMP Datei genau 8x8 Pixel groß sein muss, um angezeigt werden zu können. Alle anderen Formate oder Größen werden nicht zur Anzeige-Auswahl angeboten. Die Auswahl, welche BMP Datei angezeigt werden soll, erfolgt über die Hauptwebseite:

LED Display

Available Pictures in SPIIFS for 8x8 Display

<input checked="" type="radio"/> Clear LED Display
<input type="checkbox"/> Image 1 /8x8_64Bit.bmp Res: 8x8px Filesize: 246 Byte
<input type="checkbox"/> Image 2 /8x8_64Bit_white.bmp Res: 8x8px Filesize: 246 Byte
<input type="checkbox"/> Image 3 /8x8_64Bit_red.bmp Res: 8x8px Filesize: 246 Byte
<input checked="" type="checkbox"/> Image 4 /8x8_1Bit.bmp Res: 8x8px Filesize: 94 Byte
<input type="checkbox"/> Image 5 /8x8_64Bit_Christmas_Tree.bmp Res: 8x8px Filesize: 246 Byte
<input type="checkbox"/> Image 6 /8x8_64Bit_Blackbmp.bmp Res: 8x8px Filesize: 246 Byte
<input type="checkbox"/> Image 7 /8x8_64Bit_Hi.bmp Res: 8x8px Filesize: 246 Byte

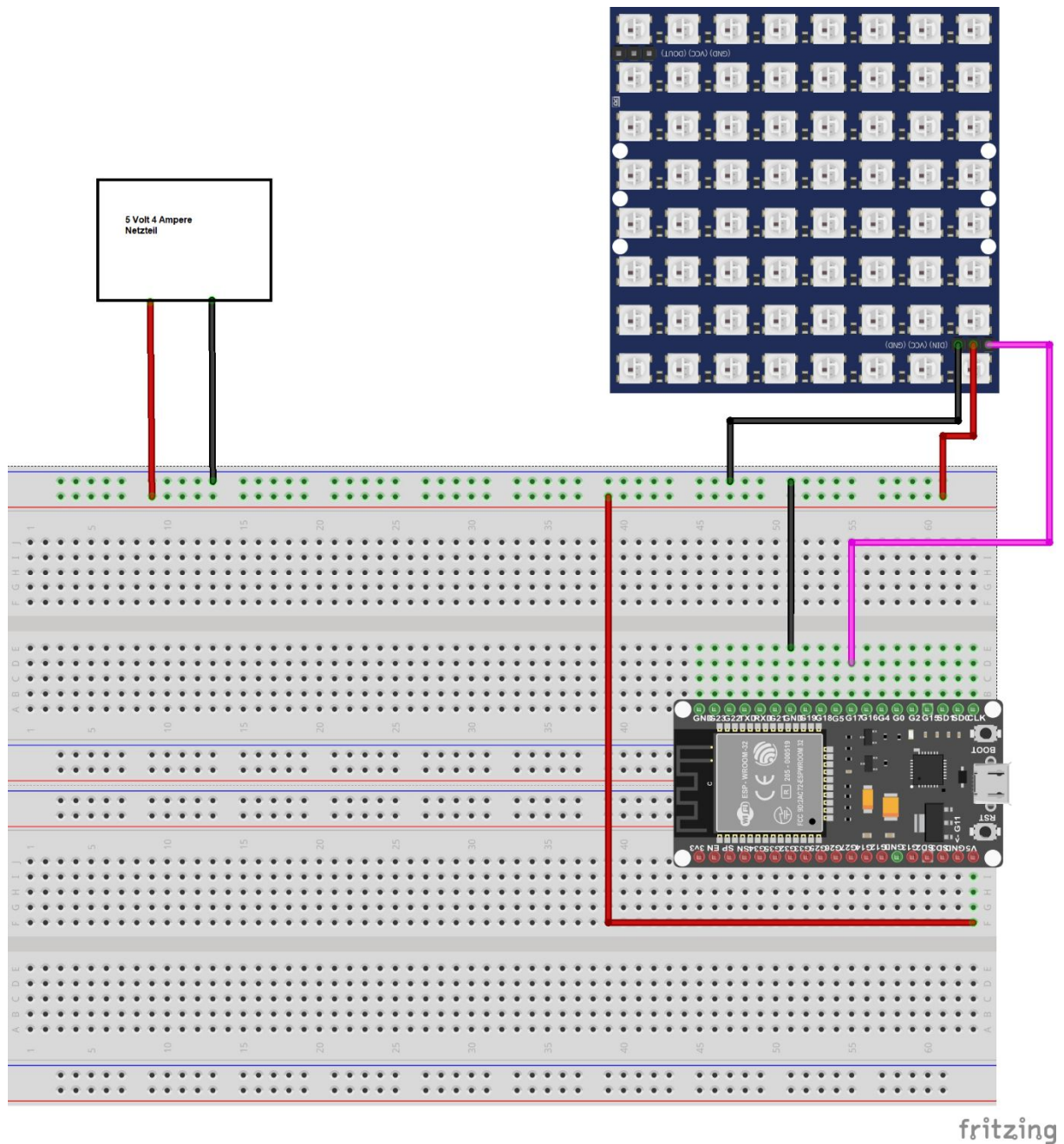
Show Image on Led Display

Systemlinks:

[WIFI Settings](#)
[Filemanager](#)

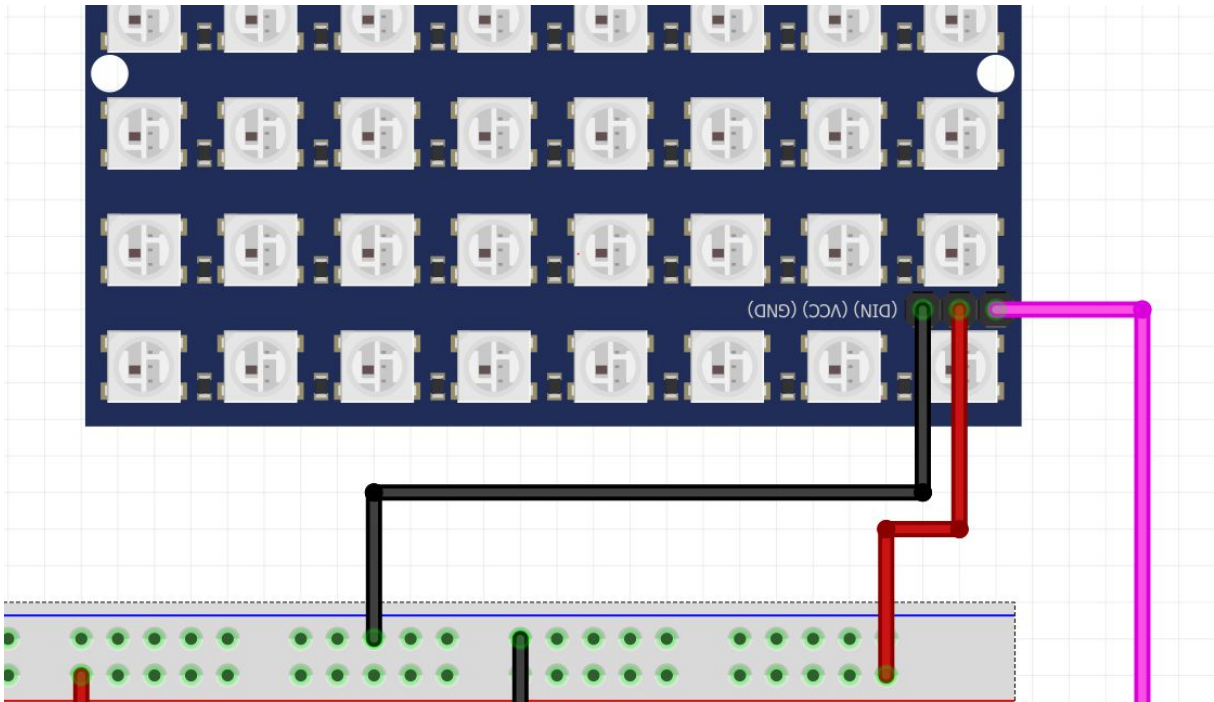
Programmed and designed by: Tobias Kuch
Contact information: tobias.kuch@gmail.com.

Zunächst jepodh müssen wir die Hardware um ein 8x8 Dot Matrix Display ergänzen, und verdrahten. Der Aufbau gestaltet sich dank des Eindraht-Busses der WS2812 LED's recht einfach:



fritzing

Zur besseren Übersicht hier nochmal die Verdrahtung des LED's Moduls im Detail:



Wir laden den erweiterten Code auf den ESP 32 hoch:

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

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

#define LED_PIN 17
#define COLOR_ORDER GRB
#define CHIPSET WS2812

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 defined
    char WiFiPwd[WiFiPwdLen]; // WiFiPassword, if defined
};
```

```

    char ConfigValid[3]; //If Config is Vaild, Tag "TK" is required"
};

struct BMPHeader // BitMapStucture
{
    uint32_t fileSize; //
    uint32_t creatorBytes; //
    uint32_t imageOffset; // Start of image data "Image Offset:
    uint32_t headerSize; //
    uint32_t width;
    uint32_t height;
    uint16_t planes;
    uint16_t depth; // bits per pixel
    uint32_t format;
};

/* 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;

//Common 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;

File fsUploadFile; // a File object to temporarily store the received file
WiFiEEPromData MyWiFiConfig;
String getContentType(String filename); // convert the file extension to the MIME type
bool handleFileRead(String path); // send the right file to the client (if it exists)
void handleFileUpload(); // upload a new file to the SPIFFS
String temp = "";

byte BRIGHTNESS = 100; // PresetBrightness
// Params for LED's
const uint8_t kMatrixWidth = 8;
const uint8_t kMatrixHeight = 8;
//const bool kMatrixSerpentineLayout = false;

#define NUM_LEDS (kMatrixWidth * kMatrixHeight)

```

```

CRGB leds_plus_safety_pixel[ NUM_LEDS + 1];
CRGB* const leds( leds_plus_safety_pixel + 1);

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 initialized at 9600 Baud."));
  FastLED.addLeds<CHIPSET, LED_PIN, COLOR_ORDER>(leds,
NUM_LEDS).setCorrection(TypicalSMD5050);
  FastLED.setBrightness(BRIGHTNESS);
  FastLED.show();
  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."));
      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 ();
  }
}

```

```

    CreateSoftAPSuccess = CreateWifiSoftAP();
    saveCredentials();
    // Blink
    delay(500);
}
// Initialize Filesystem
CInitFSSystem = InitializeFileSystem();
if (!(CInitFSSystem)) {Serial.println(F("File System not initialized ! ")); }
if ((ConnectSuccess or CreateSoftAPSuccess))
{
    Serial.print (F("IP Address: "));
    if (CreateSoftAPSuccess) { Serial.println(WiFi.softAPIP());}
    if (ConnectSuccess) { Serial.println(WiFi.localIP());}
    InitializeHTTPServer();
}
else
{
    Serial.setDebugOutput(true); //Debug Output for WLAN on Serial Interface.
    Serial.println(F("Error: Cannot connect to WLAN. Set DEFAULT Configuration."));
    SetDefaultWiFiConfig();
    CreateSoftAPSuccess = CreateWifiSoftAP();
    InitializeHTTPServer();
    SetDefaultWiFiConfig();
    saveCredentials();
}
for ( int i = 0; i < NUM_LEDS; i++) // Clear LED Display
{
    leds[i]= 0x000000;
}
FastLED.show(); // Clear Display :)
}

void InitializeHTTPServer()
{
    bool initok = false;
    /* Setup web pages: root, wifi config pages, SO captive portal detectors and not found. */
    server.on("/", handleRoot);
    server.on("/wifi", handleWifi);
    server.on("/filesystem", HTTP_GET,handleDisplayFS);
    server.on("/upload", HTTP_POST, []() {
        server.send(200, "text/plain", "");
    }, handleFileUpload);
    // 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 );
server.begin(); // Web server start
}
```

```
boolean InitalizeFileSystem() {
  bool initok = false;
  initok = SPIFFS.begin();
  delay(200);
  if (!(initok))
  {
    Serial.println(F("Format SPIFFS"));
    SPIFFS.format();
    initok = SPIFFS.begin();
  }
  return initok;
}
```

```
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."));
  } 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 Status"
{
    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 corrcr PWD
{
    WiFi.begin(MyWiFiConfig.APSTAName, MyWiFiConfig.WiFiPwd);
    connRes = WiFi.waitForConnectResult();
    delay(2000);
    i++;
    Serial.print(F("."));
}
if (connRes == 4 ) {
    Serial.println(F("STA Pwd Err"));
    Serial.println(MyWiFiConfig.APSTAName);
    Serial.println(MyWiFiConfig.WiFiPwd);
    WiFi.disconnect();
}
Serial.println(F(""));
return connRes;
}

uint16_t read16(File f)
{

```

```

// BMP data is stored little-endian, same as Arduino.
uint16_t result;
((uint8_t *)&result)[0] = f.read(); // LSB
((uint8_t *)&result)[1] = f.read(); // MSB
return result;
}

uint32_t read32(File f)
{
// BMP data is stored little-endian, same as Arduino.
uint32_t result;
((uint8_t *)&result)[0] = f.read(); // LSB
((uint8_t *)&result)[1] = f.read();
((uint8_t *)&result)[2] = f.read();
((uint8_t *)&result)[3] = f.read(); // MSB
return result;
}

BMPHeader ReadBitmapSpecs(String filename)
{
File file;
BMPHeader BMPData;
file = SPIFFS.open(filename, "r");
if (!file)
{
file.close();
return BMPData;
}
// Parse BMP header
if (read16(file) == 0x4D42) // BMP signature
{
BMPData.fileSize = read32(file);
BMPData.creatorBytes = read32(file);
BMPData.imageOffset = read32(file); // Start of image data
BMPData.headerSize = read32(file);
BMPData.width = read32(file);
BMPData.height = read32(file);
BMPData.planes = read16(file);
BMPData.depth = read16(file); // bits per pixel
BMPData.format = read32(file);
}
file.close();
return BMPData;
}

#define SD_BUFFER_PIXELS 20

void drawBitmap_SPIFFS(String filename, uint8_t x, uint8_t y)
{
File file;
uint8_t buffer[3 * SD_BUFFER_PIXELS]; // pixel buffer, size for r,g,b
bool valid = false; // valid format to be handled
bool flip = true; // bitmap is stored bottom-to-top

```

```

uint32_t pos = 0;
file = SPIFFS.open(filename, "r");
if (!file)
{
    Serial.print(F("Filesystem Error"));
    return;
}
// Parse BMP header
if (read16(file) == 0x4D42) // BMP signature
{
    uint32_t fileSize = read32(file);
    uint32_t creatorBytes = read32(file);
    uint32_t imageOffset = read32(file); // Start of image data
    uint32_t headerSize = read32(file);
    uint32_t width = read32(file);
    uint32_t height = read32(file);
    uint16_t planes = read16(file);
    uint16_t depth = read16(file); // bits per pixel
    uint32_t format = read32(file);
    if ((planes == 1) && (format == 0)) // uncompressed is handled
    {
        Serial.print(F("File size: "));
        Serial.println(fileSize);
        Serial.print(F("Image Offset: "));
        Serial.println(imageOffset);
        Serial.print(F("Header size: "));
        Serial.println(headerSize);
        Serial.print(F("Bit Depth: "));
        Serial.println(depth);
        Serial.print(F("Image size: "));
        Serial.print(width);
        Serial.print('x');
        Serial.println(height);
        uint32_t rowSize = (width * depth / 8 + 3) & ~3;
        if (height < 0)
        {
            height = -height;
            flip = false;
        }
        uint16_t w = width;
        uint16_t h = height;
        size_t buffidx = sizeof(buffer); // force buffer load
        for (uint16_t row = 0; row < h; row++) // for each line
        {
            if (flip) // Bitmap is stored bottom-to-top order (normal BMP)
                pos = imageOffset + (height - 1 - row) * rowSize;
            else // Bitmap is stored top-to-bottom
                pos = imageOffset + row * rowSize;
            if (file.position() != pos)
            { // Need seek?
                file.seek(pos, SeekSet); // if mode is SeekSet, position is set to offset bytes from the
                beginning.
                // if mode is SeekCur, current position is moved by offset bytes.
            }
        }
    }
}

```

```

        // if mode is SeekEnd, position is set to offset bytes from the end of the
        buffidx = sizeof(buffer); // force buffer reload
    }
    uint8_t bits;
    for (uint16_t col = 0; col < w; col++) // for each pixel
    {
        // Time to read more pixel data?
        if (buffidx >= sizeof(buffer))
        {
            file.read(buffer, sizeof(buffer));
            buffidx = 0; // Set index to beginning
        }
        switch (depth)
        {
            case 1: // one bit per pixel b/w format
            {
                valid = true;
                if (0 == col % 8)
                {
                    bits = buffer[buffidx++];
                }
                uint16_t bw_color = bits & 0x80;
                uint16_t PixelNum = (row*8)+col;
                leds[PixelNum].red = bw_color;
                leds[PixelNum].green = bw_color;
                leds[PixelNum].blue = bw_color;
                bits <<= 1;
            }
            break;
            case 24: // standard BMP format
            {
                valid = true;
                uint16_t b = buffer[buffidx++];
                uint16_t g = buffer[buffidx++];
                uint16_t r = buffer[buffidx++];
                uint16_t PixelNum = (row*8)+col;
                leds[PixelNum].red = r;
                leds[PixelNum].green = g;
                leds[PixelNum].blue = b;
            }
            break;
        }
    } // end pixel
} // end line
FastLED.show(); // Show results :)
}
}
file.close();
if (!(valid))
{
    Serial.println(F("Err: BMP"));
}
}

```

```

void handleFileUpload() {
  if (server.uri() != "/upload") return;
  HTTPUpload& upload = server.upload();
  if (upload.status == UPLOAD_FILE_START) {
    String filename = upload.filename;
    if (upload.filename.length() > 30) {
      upload.filename = upload.filename.substring(upload.filename.length() - 30,
upload.filename.length()); // Dateinamen auf 30 Zeichen kürzen
    }
    Serial.println("FileUpload Name: " + upload.filename);
    if (!filename.startsWith("/")) filename = "/" + filename;
    fsUploadFile = SPIFFS.open("/") + server.urlDecode(upload.filename), "w");
    filename = String();
  } else if (upload.status == UPLOAD_FILE_WRITE) {
    if (fsUploadFile)
      fsUploadFile.write(upload.buf, upload.currentSize);
  } else if (upload.status == UPLOAD_FILE_END) {
    if (fsUploadFile)
      fsUploadFile.close();
    handleDisplayFS();
  }
}

void handleDisplayFS() {          // HTML Filesystem
  // Page: /filesystem
  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
  server.send ( 200, "text/html", 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><head><title>File System
Manager</title></head>";
  temp += "<h2>Serial Peripheral Interface Flash Filesystem</h2><body><left>";
  server.sendContent(temp);
  temp = "";
  if (server.args() > 0) // Parameter wurden uebergeben
  {
    if (server.hasArg("delete"))
    {
      String FToDel = server.arg("delete");
      if (SPIFFS.exists(FToDel))

```

```

    {
        SPIFFS.remove(FToDel);
        temp += "File " + FToDel + " successfully deleted.";
    } else
    {
        temp += "File " + FToDel + " cannot be deleted.";
    }
    server.sendContent(temp);
    temp = "";
}
if (server.hasArg("format") and server.arg("on"))
{
    SPIFFS.format();
    temp += "SPI File System successfully formatted.";
    server.sendContent(temp);
    temp = "";
} // server.client().stop(); // Stop is needed because we sent no content length
}

temp += "<table border=2 bgcolor = white width = 400 ><td><h4>Current SPIFFS Status: </h4>";
temp += formatBytes(SPIFFS.usedBytes() * 1.05) + " of " + formatBytes(SPIFFS.totalBytes()) + "
used. <br>";
temp += formatBytes((SPIFFS.totalBytes() - (SPIFFS.usedBytes() * 1.05))) + " free. <br>";
temp += "</td></table><br>";
server.sendContent(temp);
temp = "";
// Check for Site Parameters
temp += "<table border=2 bgcolor = white width = 400><tr><th><br>";
temp += "<h4>Available Files on SPIFFS:</h4><table border=2 bgcolor = white
></tr></th><td>Filename</td><td>Size</td><td>Action </td></tr></th>";
server.sendContent(temp);
temp = "";
File root = SPIFFS.open("/");
File file = root.openNextFile();
while (file)
{
    temp += "<td> <a title=\"Download\" href =\"\" + String(file.name()) + \"\" download=\"\" +
String(file.name()) + \"\">\" + String(file.name()) + \"</a> <br></th>";
    temp += "<td>\" + formatBytes(file.size()) + \"</td>\";
    temp += "<td><a href =filesystem?delete=\" + String(file.name()) + \"> Delete </a></td>\";
    temp += "</tr></th>\";
    file = root.openNextFile();
}
temp += "</tr></th>\";
temp += "</td></tr></th><br></th></tr></table></table><br>";
temp += "<table border=2 bgcolor = white width = 400 ><td><h4>Upload</h4>\";
temp += "<label> Choose File: </label>\";
temp += "<form method='POST' action='/upload' enctype='multipart/form-data'
style='height:35px;'><input type='file' name='upload' style='height:35px; font-size:13px;'
required>\r\n<input type='submit' value='Upload' class='button'></form>\";
temp += "</table><br>\";
server.sendContent(temp);
temp = "";

```

```

    temp += "<td><a href =filesystem?format=on> Format SPIFFS Filesystem. (Takes up to 30
Seconds) </a></td>";
    temp += "<table border=2 bgcolor = white width = 500 cellpadding =5
><caption><p><h3>Systemlinks:</h2></p></caption><tr><th><br>";
    temp += " <a href='/'>Main Page</a><br><br></th></tr></table><br><br>";
    server.setContent(temp);
    temp = "";
    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></bo
dy></html>";
    //server.send ( 200, "", temp );
    server.setContent(temp);
    server.client().stop(); // Stop is needed because we sent no content length
    temp = "";
}

/** 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 = "";
short PicCount = 0;
byte ServArgs = 0;

//Building Page
// 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 Client senden
temp = "";
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}";
server.sendContent(temp);
temp = "";

```

```

temp += "body {background-color: powderblue;}</style>";
temp += "<head><title>Tobi's LED Display</title></head>";
temp += "<h2>LED Display</h2>";
temp += "<body>";
server.sendContent(temp);
temp = "";
// Processing User Request
if (server.args() > 0) // Parameter wurden uebergeben
{
    temp += "<br>Eingaben werden verarbeitet. Bitte warten..<br><br>";
    server.sendContent(temp);
    temp = "";
// Update Background Paper
if (server.arg("PicSelect") == "off") // Clear LED Display
{
    temp = "";
    for ( int i = 0; i < NUM_LEDS; i++)
    {
        leds[i]= 0x000000;
    }
    FastLED.show();
} else
{
    temp = server.arg("PicSelect"); // Bild gewaehlt. Display inhalt per Picselect hergstellt
    drawBitmap_SPIFFS(temp,0,0);
    temp = "";
}
}
temp += "<table border=2 bgcolor = white ><caption><p><h3>Available Pictures in SPIIFS for 8x8
Display</h2></p></caption>";
temp += "<form>";
temp += "<tr><th><input type='radio' name='PicSelect' value = 'off' checked> Clear LED
Display<br></th></tr>";
temp += "<tr><th>";
//List available BMP Files in SPIFFS
File root = SPIFFS.open("/");
File file = root.openNextFile();
PicCount = 1;
while (file)
{
    if (String(file.name()).endsWith(".bmp") or String(file.name()).endsWith(".BMP"))
    {
        BMPHeader PicData = ReadBitmapSpecs(file.name());
        if ((PicData.width < kMatrixWidth + 1) and (PicData.height < kMatrixHeight + 1 )) // Display only
        in list, when Bitmap not exceeding Display Resolution. Bigger Images are not listed.
        {
            temp += "<label for='radio1'><img src='"+ String(file.name())+"' alt='"+ String(file.name())+"'
            border='3' bordercolor=green> Image " + PicCount+"</label><input type='radio' value='"+
            String(file.name())+"' name='PicSelect' /> <br>";
            temp += String(file.name())+ " Res: " + String(PicData.width) + "x" + String(PicData.height) +
            "px Filesize: " + formatBytes(file.size()) + "</th></tr><tr><th>";
            PicCount ++;
        }
    }
}

```

```

    }
    file = root.openNextFile();
    }
    server.sendContent(temp);
    temp = "";
    temp = "<button type='submit' name='action' value='0' style='height: 50px; width: 280px'>Show
Image on Led Display</button>";
    temp += "</form>";
    temp += "<br><table border=2 bgcolor = white width = 280 cellpadding =5
><caption><p><h3>Systemlinks:</h2></p></caption>";
    temp += "<tr><th><br>";
    temp += "<a href='/wifi'>WIFI Settings</a><br><br>";
    temp += "<a href='/filesystem'>Filemanager</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 captive portal redirect instead of displaying the error page.
        return;
    }
    if (!handleFileRead(server.uri()))
    {
        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 += "<a href='/filesystem'>Filemanager</a><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.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 whole 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];
            }
            // MyWiFiConfig.APSTAName[len+1] = '\0';
            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];
                }
            }
            // MyWiFiConfig.WiFiPwd[len+1] = '\0';
            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.setContent(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 whole 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.";
}
// End WifiAP
}
// 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 Settings</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 = "";
}

void handleUploadSave()
{
    String FileData ;
    temp = "";
    for (byte i = 0; i < server.args(); i++)
    {
        temp += "Arg " + (String)i + " -> "; //Include the current iteration value
        temp += server.argName(i) + ": "; //Get the name of the parameter
        temp += server.arg(i) + "\n"; //Get the value of the parameter
    }
    // server.send(200, "text/plain", temp); //Response to the HTTP request
    FileData = server.arg("datei");
    server.sendHeader("Location", "filesystem", true);
    server.sendHeader("Cache-Control", "no-cache, no-store, must-revalidate");
    server.sendHeader("Pragma", "no-cache");
    server.sendHeader("Expires", "-1");
    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
}

/** 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";
    }
}

String getContentType(String filename) { // convert the file extension to the MIME type
    if (filename.endsWith(".htm")) return "text/html";
    else if (filename.endsWith(".css")) return "text/css";
}

```

```

else if (filename.endsWith(".js")) return "application/javascript";
else if (filename.endsWith(".ico")) return "image/x-icon";
else if (filename.endsWith(".gz")) return "application/x-gzip";
else if (filename.endsWith(".bmp")) return "image/bmp";
else if (filename.endsWith(".tif")) return "image/tiff";
else if (filename.endsWith(".pbm")) return "image/x-portable-bitmap";
else if (filename.endsWith(".jpg")) return "image/jpeg";
else if (filename.endsWith(".gif")) return "image/gif";
else if (filename.endsWith(".png")) return "image/png";
else if (filename.endsWith(".svg")) return "image/svg+xml";
else if (filename.endsWith(".html")) return "text/html";
else if (filename.endsWith(".wav")) return "audio/x-wav";
else if (filename.endsWith(".zip")) return "application/zip";
else if (filename.endsWith(".rgb")) return "image/x-rgb";
// Complete List on https://wiki.selfhtml.org/wiki/MIME-Type/Übersicht
return "text/plain";
}

bool handleFileRead(String path) { // send the right file to the client (if it exists)
  if (path.endsWith("/")) path += "index.html"; // If a folder is requested, send the index file
  String contentType = getContentType(path); // Get the MIME type
  String pathWithGz = path + ".gz";
  if (SPIFFS.exists(pathWithGz) || SPIFFS.exists(path)) { // If the file exists, either as a compressed
  archive, or normal
    if (SPIFFS.exists(pathWithGz)) // If there's a compressed version available
      path += ".gz"; // Use the compressed version
    File file = SPIFFS.open(path, "r"); // Open the file
    size_t sent = server.streamFile(file, contentType); // Send it to the client
    file.close(); // Close the file again
    return true;
  }
  return false;
}

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

```

Ich habe ein paar 8x8 BMP Grafiken in verschiedenen Farbtiefe für euch zum Testen erstellt:



Im nächsten Teil kümmern wir uns um eine höhere Auflösung unseres LED Displays. Bis dahin wünsche viel Spaß mit dem Anzeigen eigener BMP Dateien auf dem LED-Display.

