Korrigenda zum Buch „Das ESP8266-Projektbuch“ von Martin Mohr

Auf den Seiten 34, 65 und 66 beinhalten mehrere Wörter ein Leerzeichen, das an diesen Stellen aber falsch ist. Richtig lauten die Wörter wie folgt:

ESP8266WebServer

ESP8266WebServer.h

Die betreffenden Wörter sind auf den folgenden Seiten markiert.

Viele Grüße von Ihrem entwickler.press-Lektorat.

Feedback und Anregungen nehmen wir jederzeit unter lektorat@entwickler-press.de entgegen.

In der *setup();*-Funktion wird das serielle Interface als erstes initialisiert, damit wir die Möglichkeit haben, Debuginformationen ausgeben zu lassen. Danach verbinden wir uns mit dem WLAN und bereiten das Starten des Web Servers vor. Am Ende der *setup();*-Funktion wird der Web Server zu guter Letzt gestartet. In der *loop();*-Funktion müssen wir dann nur noch *server.handleClient();* aufrufen, damit der Web Server permanent läuft.

Es ist natürlich klar, dass Sie die IP-Adressen und auch die WLAN-Verbindungsdaten an Ihre Gegebenheiten anpassen müssen. Weiterhin müssen noch die Werte der *myservo.write();*-Aufrufe angepasst werden. Diese sollen eigentlich den Winkelgrad der Servoposition angeben. Leider passt das so gut wie nie, da sich die Servos verschiedener Hersteller alle leicht unterschiedlich verhalten. Die einzige Möglichkeit, die man hat, ist manuelles Nachjustieren.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <Servo.h>
const char* ssid = "<YOUR_SSID>";
const char* pass = "<YOUR_PASS>";
IPAddress ip(192,168,1,20);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);
Servo myservo;
String inString;
ESP8266WebServer server(80);
void handleRoot() {
  server.send(200, "text/plain", "ESP Flag Server");
  }
void handleFlag() {
  server.send(200, "text/plain", "Flag is up");
  myservo.attach(13);
  myservo.write(80);
  delay(1500);
  myservo.detach();
  }
void setup() {
  Serial.begin(115200);
  Serial.println("\r\n") ;
  WiFi.begin(ssid, pass);
  WiFi.config(ip, gateway, subnet);
```

Das ist der Grund, warum wir uns das Programm etappenweise ansehen werden.

Direkt am Anfang stehen, wie man es von C-Programmen her kennt, die Includes der zusätzlichen Bibliotheken. Da wir einige davon noch nicht benutzt haben, werden wir sie uns etwas genauer ansehen.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <Wire.h>
#include <time.h>
#include <EEPROM.h>
```

Die *ESP8266WiFi.h* kennen wir schon aus den vorhergehenden Projekten. Sie bringt die Grundfunktionen mit, um auf ein WLAN zuzugreifen. Die *ESP8266WebServer.h* ist die Bibliothek für alles rund um den Web Server des ESP8266. Die *Wire.h* ermöglicht uns die serielle Kommunikation. Wir brauchen sie, um mit unserem Sensor über I²C sprechen zu können. Jetzt kommen die zwei Neuzugänge: Die *time.h* bringt jede Menge Funktionen rund um die Uhrzeit mit, unter anderem auch ein Synchronisieren der internen Uhr des ESP8266 mit einen NTP Server. NTP Server sind die Zeitserver im Internet, sie liefern die Uhrzeit fast atomgenau. Und last but not least die *EEPROM.h*, mit der es möglich ist, Daten in das interne EEPROM des ESP8266 zu schreiben und daraus zu lesen. Das wird benötigt, damit wir die Parameter unserer Gartenbewässerung nicht nach einem Neustart verlieren.

Der nächste Block in unserem Programm definiert Konstanten und globale Variablen. Besonderes Augenmerk sei hier auf alle Arrays gelegt. Möchte man einen Bereich hinzufügen, muss man die Konstante *maxBereich* um 1 erhöhen. Bitte verzeihen Sie mir die Speicherplatzverschwendung, die nullte Stelle in allen Arrays ist unbenutzt. Das erste Element findet sich unter dem Index 1, nicht 0, wie bei C sonst üblich. Das Array *GPIO* gibt an, welcher GPIO-Anschluss zu welchem Bereich gehört. Hier müssen natürlich beim Erweitern der Bereiche auch zusätzliche Anschlüsse des ESP8266 hinzugefügt werden.

```
const char* ssid = "<Your SSID>";
const char* pass = "<Your pass>";
IPAddress ip(192,168,2,60);
```

```
IPAddress gateway(192,168,2,1);
IPAddress subnet(255,255,255,0);
ESP8266WebServer server(80);
const int maxBereich = 4;
String beginTime[maxBereich+1];
String endTime[maxBereich+1];
String area[maxBereich+1];
int GPIO[maxBereich+1]={0,13,12,14,16};
char timeC[5];
String timeString;
const int I2Caddress=0x20;
int hum;
int oldMinute;
```

Die nun folgenden drei Funktionen dienen dazu, die Werte für Bodenfeuchte, Helligkeit und Temperatur aus dem Sensor auszulesen. Sie steuern den Sensor über I²C-Bus an und lesen die entsprechenden Werte aus. Die Werte werden innerhalb der Funktionen so umgerechnet, dass man sie direkt anzeigen kann.

```
float getTemp(){
  float t;
  int16_t temp,high,low;
  Wire.beginTransmission(I2Caddress);
  Wire.write(5);
  Wire.endTransmission();
  Wire.beginTransmission(I2Caddress);
  Wire.requestFrom(I2Caddress, 2);
  if (Wire.available()) {
    high = Wire.read();
    low = Wire.read();
    }
  Wire.endTransmission();
  temp=low;
  temp+=high<<8;
  t=temp;
  t=t/10;
  return t;
  }

int getHum() {
  int h;
  int16_t high,low ;
```