

Projekt Weihnachtsbeleuchtung

Projektdokumentation für das Modul Embedded
Systems I

**Florian Kropf
Gruppe H**

Inhalt

1 Idee & Vorüberlegung	2
1.1 Idee	2
1.2 Vorüberlegung	4
2 Hardware	5
2.1 LEDs-WS2812 Module	5
2.2 Microcontroller-ESP32	6
3 Software	8
3.1 MIT App Inventor	8
4 Realisierung	9
4.1 Konzept	9
4.2 Arduino-Code	12
4.3 Android-App	14
4.4 Fotos	16
5 Troubleshooting	18
Arduino-Code	19
Referenzen	26

1 Idee & Vorüberlegung

1.1 Idee

Grundlage des Projekts ist der alte Weihnachtsschmuck meiner Familie, der entsorgt werden sollte. Die Dekoration bestand aus drei "Engeln", die mit einfarbigen, weißen LEDs versehen waren. Da die Schalter zum Einschalten der Beleuchtung an der Unterseite jedes Engels angebracht sind, muss jeder Engel einzeln angehoben werden um die LEDs ein- und auszuschalten. Außerdem sind in den LED-Einheiten keine "Timer-Module" verbaut, was die Benutzung der Lichter sehr umständlich gestaltet, vor allem, wenn die "Engel" z.B. auf einem Regal positioniert sind. Das war auch der Grund, für die geplante Entsorgung der Engel.

Das Projekt soll nun die Bedienung der Engel erleichtern und wenn möglich auch noch eine bunte Beleuchtung der Engel ermöglichen.

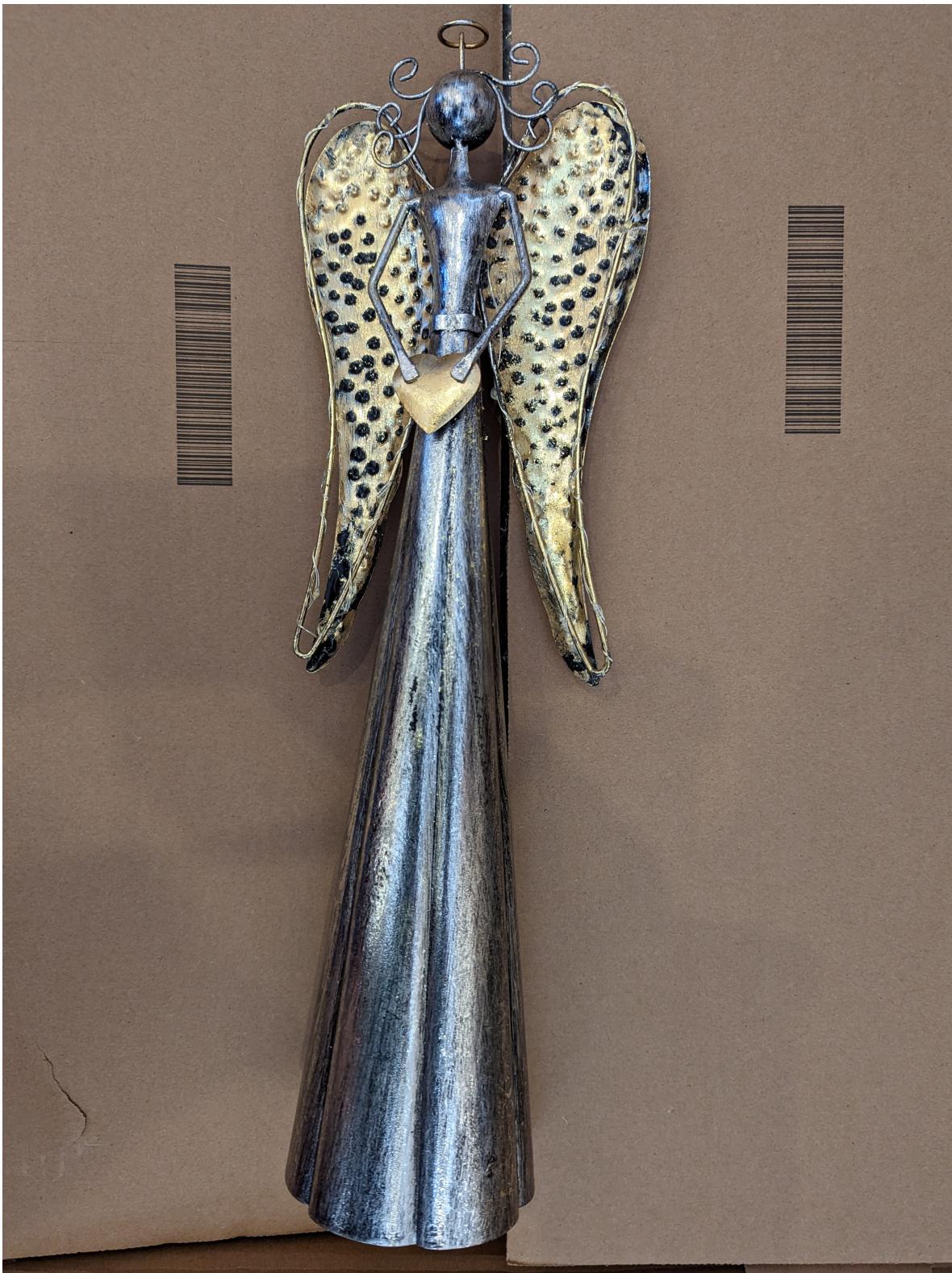


Figure 1: der Weihnachtsschmuck

1.2 Vorüberlegung

Die Bedienung der LEDs an den Engeln soll kabellos z.B. mit dem Smartphone erfolgen. Außerdem sollen die einfarbigen, weißen LEDs der Engel mit RGB-LEDs ersetzt werden, um weitere Möglichkeiten der Beleuchtung zu ermöglichen. Die Engel sollen als Einheit bestehen bleiben und von einem Mikrocontroller gesteuert werden.

Fraglich war nun, welches Kommunikationsprotokoll, zur Kommunikation zwischen Smartphone und Mikrocontroller, verwendet werden sollte. In Frage kamen WLAN sowie Bluetooth, da fast jedes Smartphone über diese Schnittstellen kommunizieren kann.

Außerdem soll die Kommunikation über eine App auf dem Smartphone gesteuert werden. Möglich wären hier auch Web-Interfaces, die direkt vom Mikrocontroller zur Verfügung gestellt werden. Die Benutzung einer App soll die Bedienung aber erleichtern und übersichtlicher gestalten.

2 Hardware

2.1 LEDs-WS2812 Module

Um die neu geplante Beleuchtung der Engel zu ermöglichen, sollen WS2812 LED-Module zum Einsatz kommen. Mit diesen Modulen habe ich bereits Erfahrungen sammeln können und sie in einigen anderen Projekten eingesetzt. (Bsp. Fahrradbeleuchtung [1]).

Der WS2812 ist ein LED-Chip, der eine RGB-LED (Rot, Grün, Blau) und einen integrierten Controller in einem Gehäuse kombiniert. Sie sind auch unter dem Namen "Neopixel" bekannt. Unter diesem Namen werden sie von der US-amerikanischen Firma "Adafruit" vertrieben.

WS2812 sind bekannt für ihre einfache Ansteuerung und die Fähigkeit, einzelne LEDs oder eine Reihe von LEDs zu steuern. Jeder WS2812-Chip verfügt über drei Kanäle (Rot, Grün und Blau), die es ermöglichen, eine breite Palette von Farben zu erzeugen. Der integrierte Controller des WS2812 ermöglicht es, die Farben und Helligkeiten der LEDs individuell anzupassen und komplexe Animationen und Effekte zu erstellen.

Ein großer Vorteil des WS2812 ist seine Daisy-Chain-Fähigkeit. Das bedeutet, dass mehrere WS2812-LEDs miteinander verbunden werden können, wobei der Datenstrom von LED zu LED weitergegeben wird. Dies ermöglicht es, eine größere Anzahl von LEDs mit nur einem einzigen Datenpin eines Mikrocontrollers zu steuern. Angesteuert werden die WS2812 LEDs über den PWM-Pin eines Mikrocontrollers. Adafruit stellt hierfür eine geeignete Arduino Bibliothek zur Verfügung, die über die Arduino-IDE bezogen und eingebunden werden kann. [2]

Angewendet werden WS2812 häufig z.B. für:

Ambientebeleuchtung: Der WS2812 kann verwendet werden, um farbige Beleuchtungseffekte zu erzeugen, z.B. in Wohnräumen, Restaurants oder Veranstaltungsorten. Durch die individuelle Ansteuerung der LEDs können dynamische und ansprechende Beleuchtungsszenarien erstellt werden.

Kunstinstallationen: Künstler nutzen den WS2812 gerne, um beeindruckende visuelle Effekte in ihren Installationen zu erzeugen. Die Möglichkeiten reichen von lebendigen Lichtkunstwerken bis hin zu interaktiven Installationen, bei denen die LEDs auf Umgebungssignale reagieren.

Wearable-Elektronik: Durch seine kompakte Größe und die Daisy-Chain-Fähigkeit eignet sich der WS2812 auch für den Einsatz in tragbaren elektronischen Geräten wie Kleidung, Schmuck oder Accessoires.

2.2 Microcontroller-ESP32

Um nun die WS2812-Module zu steuern, war ein Mikrocontroller notwendig, der den Vorgaben der Projektidee entsprach. Da die Wahl zwischen dem Einsatz von Wlan oder Bluetooth noch offen war, entschied ich mich für den "ESP32"

Der ESP32 ist ein Mikrocontroller, der für die drahtlose Kommunikation und die Anwendungssteuerung entwickelt wurde. Entwickelt wurde er von "Espressif Systems". Der ESP32 bietet eine Vielzahl von Funktionen und eignet sich für eine breite Palette von Anwendungen.

Der ESP32 verfügt über einen Dual-Core-Prozessor mit einer Taktgeschwindigkeit von bis zu 240 MHz. Darüber hinaus bietet er eine integrierte WLAN- und Bluetooth-Konnektivität. Deswegen findet er häufig Anwendung in IOT (Internet der Dinge)-Anwendungen.

Im Vergleich mit vielen anderen Mikrocontrollern verfügt der ESP32 über vergleichsweise viel Speicherplatz und eine gute Rechenleistung. Es können verschiedene Programmiersprachen wie Arduino und MicroPython verwendet werden, um den ESP32 zu programmieren.

Der ESP32 kann in einer Vielzahl von Projekten eingesetzt werden. z.B.:

1. IoT-Anwendungen: Der ESP32 kann als zentrales Steuerungselement für vernetzte Geräte verwendet werden. Sensoren und Aktoren können angeschlossen werden und Daten drahtlos übertragen, um Smart-Home-Systeme, Überwachungssysteme oder Umweltüberwachungssysteme zu steuern.
2. Wearable-Geräte: Aufgrund seiner geringen Größe und geringen Leistungsaufnahme eignet sich der ESP32 für die Entwicklung von tragbaren Geräten wie Fitness-Trackern, Smartwatches oder intelligenten Brillen.
3. Industrielle Automatisierung: Der ESP32 kann in industriellen Umgebungen eingesetzt werden, um Maschinen zu überwachen, Daten zu sammeln und Prozesse zu steuern. Er kann auch drahtlos mit anderen Geräten oder Systemen kommunizieren, um die Effizienz und Produktivität zu verbessern.

3 Software

Da es leider nicht möglich ist ungeprüfte Apps auf IOS Geräten zu installieren, muss die App zur Bedienung für ein Android Smartphone erstellt werden. Dafür wird in diesem Projekt der "MIT App Inventor" benutzt.

3.1 MIT App Inventor

Der MIT App Inventor ist eine webbasierte Entwicklungsumgebung, die es Benutzern ermöglicht, eigene mobile Anwendungen (Apps) für Android-Geräte zu erstellen, ohne umfangreiche Programmierkenntnisse zu besitzen. Es wurde vom Massachusetts Institute of Technology (MIT) entwickelt und ist eine benutzerfreundliche Plattform, die darauf abzielt, Programmierung zugänglich und einfach zu machen.[3]

Mit dem MIT App Inventor können Benutzer visuell und blockbasiert Apps entwickeln, indem sie Bausteine oder "Blöcke" verwenden, die spezifische Funktionen repräsentieren. Diese Blöcke können miteinander verbunden werden, um die Logik und das Verhalten der App festzulegen. Anstatt Code manuell zu schreiben, können Benutzer die Funktionen und Eigenschaften der App durch das Zusammensetzen von Blöcken definieren.

Die Entwicklungsumgebung des MIT App Inventor bietet eine Vielzahl von vordefinierten Blöcken, die Funktionen wie Benutzeroberflächenelemente, Datenbankzugriff, Sensordatenverarbeitung, Mediensteuerung und vieles mehr abdecken. Benutzer können auch eigene Blöcke erstellen, um spezifische Funktionalitäten zu implementieren oder komplexe Logik in ihre Apps einzufügen.

4 Realisierung

4.1 Konzept

Da die WS2812 mit 5V Versorgungsspannung betrieben werden müssen, können sie nicht direkt über den Microcontroller betrieben werden. Deswegen müssen sie mit einer externen Spannungsquelle (z.B. einer Powerbank) versorgt werden. Auch der ESP32 auf dem "NodeMCU-Entwicklerboard" kann mit 5V Versorgungsspannung betrieben werden. Ein voll ausgelasteter Neopixel (also weißes Licht und volle Leuchtkraft) benötigt etwa 60mA. Die Powerbank mit einer Maximalspannung von 2.1A kann also 35 Neopixel betreiben. Allerdings ist dieser Zustand oft nicht realistisch. Häufig wird nur etwa ein Drittel, also 20mA, an Spannung benötigt, da die Neopixel farbiges Licht in gedimmter Helligkeit ausgeben. In diesem Fall kann man sogar 105 Neopixel an der Powerbank betreiben! Genug für dieses Projekt.[4]

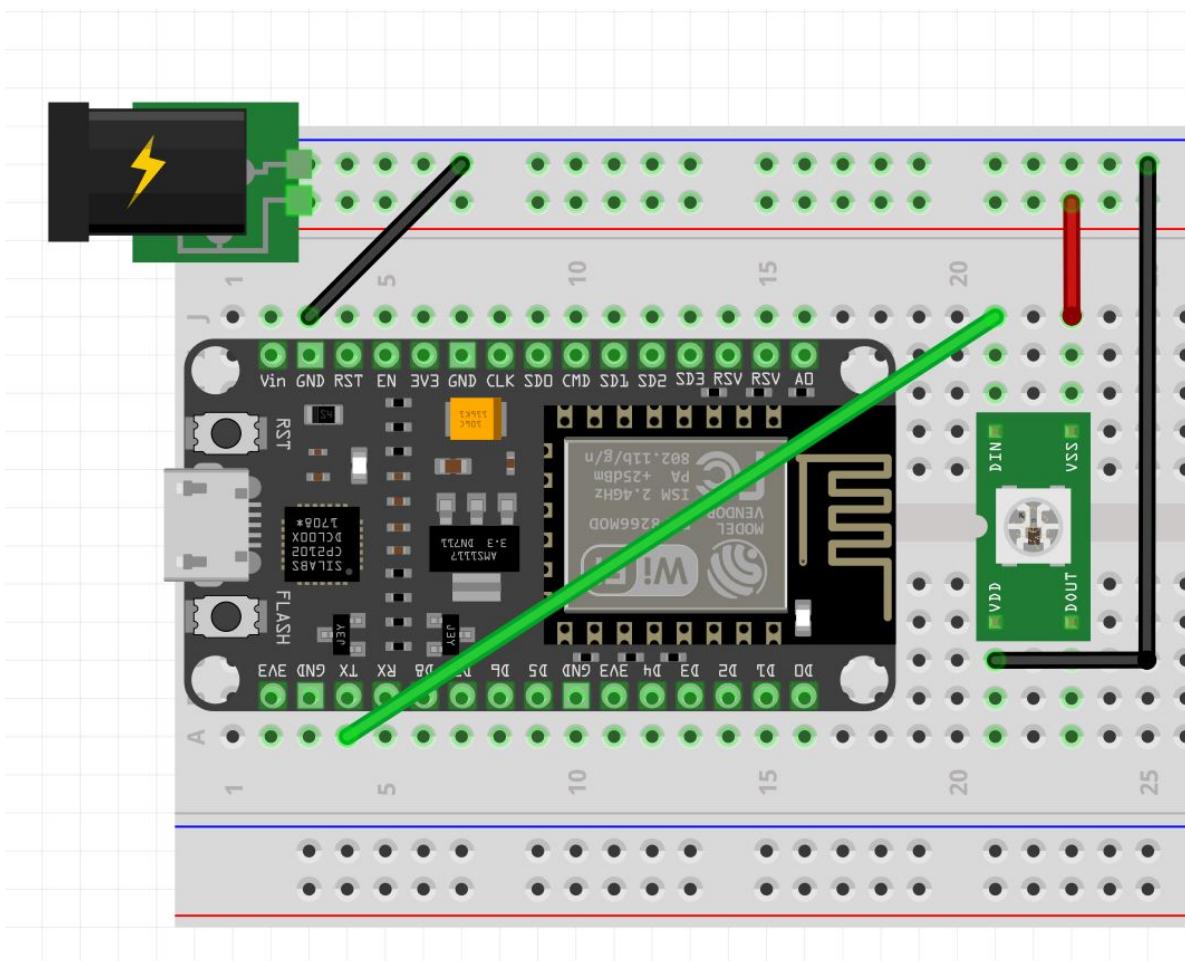


Figure 2: Schaltung mit einem WS2812 Modul

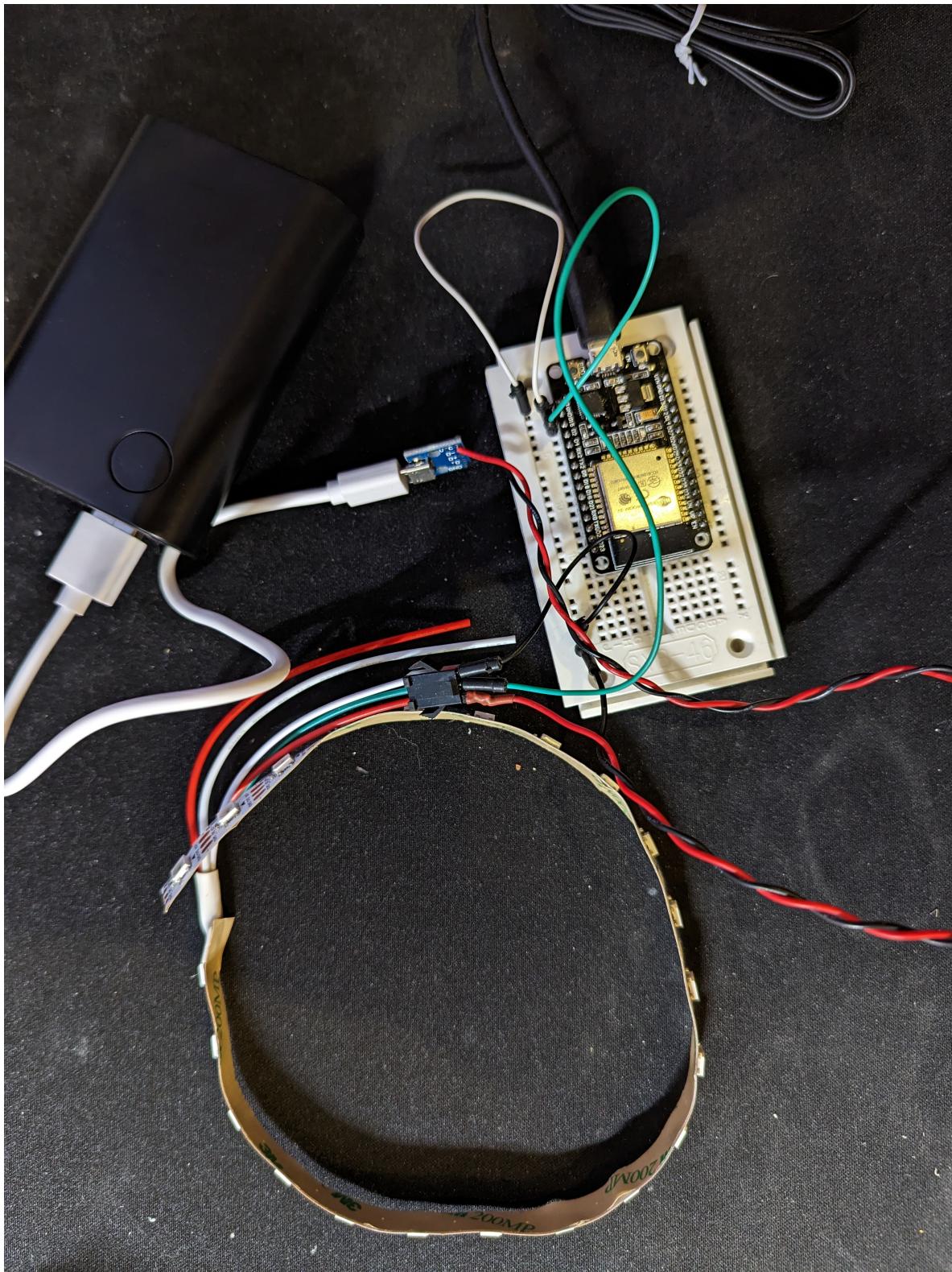


Figure 3: Versuchsaufbau der Schaltung

4.2 Arduino-Code

Im Arduino Code wird zunächst ein asynchroner Webserver im Heimnetzwerk gestartet dafür wird die "ESPAsyncWebServer" library benutzt. Im Arduino Code wird die SSID und das Passwort des Heim-Wlans eingegeben. Anschließend wird der Webserver gestartet und eine Meldung darüber über die serielle Schnittstelle übertragen. Anschließend erfolgt die Mitteilung, das der ESP32 mit dem Wlan verbunden ist, und die aktuelle IP-Adresse wird angezeigt.

```
1     Serial.begin(115200);
2     // Set pin mode
3     pinMode(LED, OUTPUT);
4     digitalWrite(LED, LOW);
5
6     WiFi.begin(ssid, password);
7
8     Serial.print("Connecting to WiFi..");
9     while (WiFi.status() != WL_CONNECTED) {
10         delay(500);
11         Serial.print(".");
12     }
13    Serial.println();
14    Serial.println("CONNECTED!");
15
16    Serial.print("IP-Adress: ");
17    Serial.print(WiFi.localIP());
```

Listing 1: Webserver Setup

Anschließend wird der http-Handler gestartet, der auf bestimmte Anfragen reagiert und entsprechende Aktionen ausführt

```
1     server.on("/hello", HTTP_GET, [] (AsyncWebRequest *request){
2         request->send(200, "text/plain", "Hello World");
3     });
```

Listing 2: Reaktion auf http-Anfrage

Im Haupteil des Programmes wird die globale Variable "op" überwacht und bei Veränderung ein entsprechendes Unterprogramm ausgeführt.

```
1  if(op == 1)  {
2      //RGB fade
3      rainbow(10);
4  }
5  else if(op == 2) {
6      //all white
7      light(strip.Color(75, 50, 50));
8  }
9
```

Listing 3: Start der Unterprogramme

Diese Unterprogramme steuern ihrerseits die Neopixel. Jedes Unterprogramm ist für eine bestimmte Beleuchtung verantwortlich. Hier wird z.B. einfach Licht in einer bestimmten Farbe an allen LEDs eingeschaltet.

```
1  void light(uint32_t color) {
2      for(int i=0; i<strip.numPixels(); i++) { // For each pixel in strip...
3          strip.setPixelColor(i, color);           // Set pixel's color (in RAM)
4      }
5      strip.show();                           // Update strip to match
6 }
```

Listing 4: "light" Unterprogramm

4.3 Android-App

Die App bietet einen Button für jeden Leuchtmodus der LEDs, einen Button zum ausschalten der LEDs und einen Test-Button um zu überprüfen, ob der ESP32 noch verbunden ist und auf Anfragen reagiert.

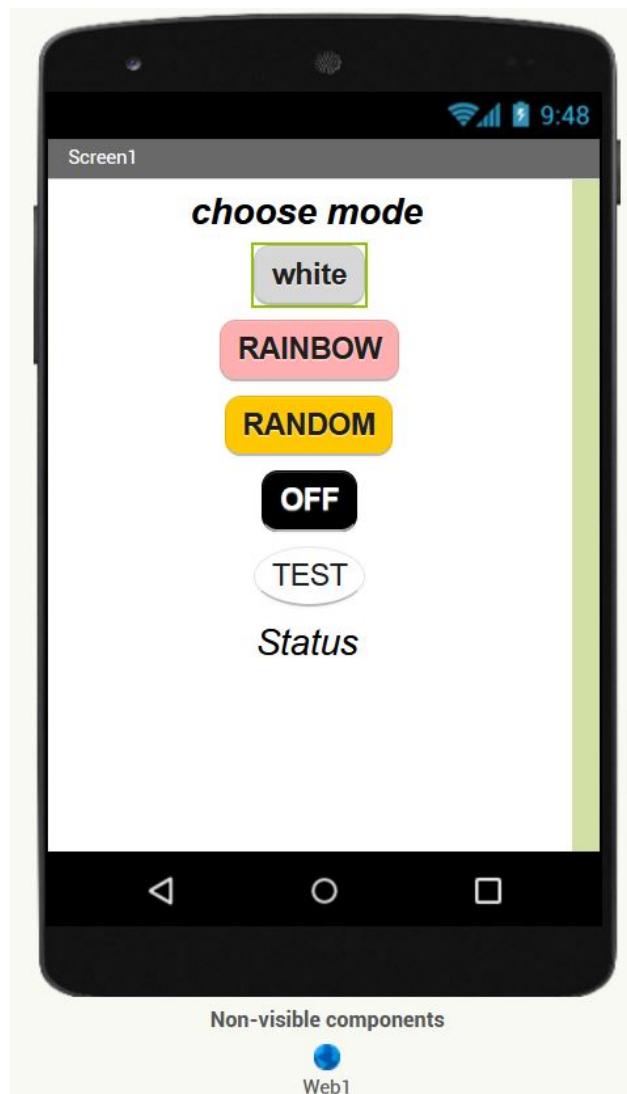


Figure 4: GUI der App



Figure 5: Codeblöcke im App Inventor

4.4 Fotos

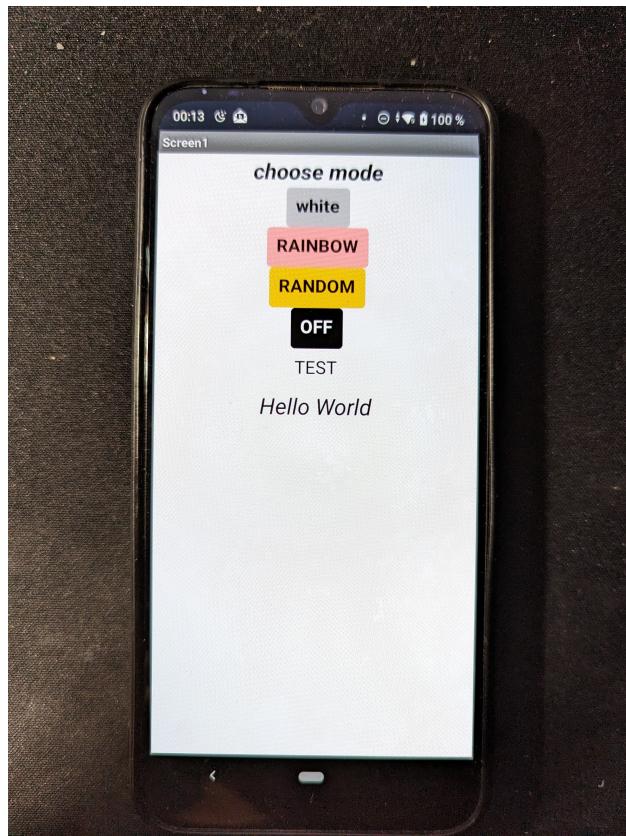


Figure 6: App auf Smartphone

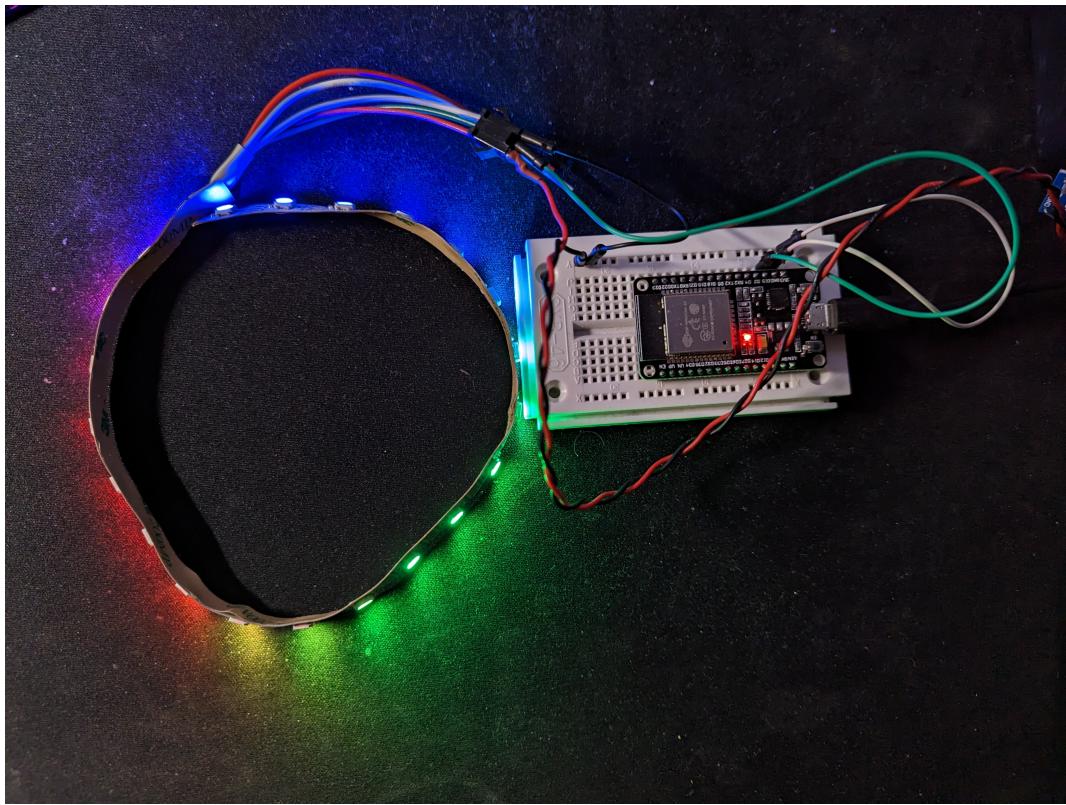


Figure 7: ESP32 mit aktivierte LEDs

5 Troubleshooting

1. ESP32 muss in Arduino IDE zusätzlich installiert werden

Lösung: Anleitung unter <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>

2. Upload aus Arduino IDE funktioniert nicht

Lösung: Boot Knopf muss bei der Aufforderung "Connecting" gedrückt werden, um ESP32 in Flash-Modus zu versetzen. [5]

3. Watchdog timer rebootet ESP32 bei laufender for Schleife

Lösung: Lösung des Problems war es, die laufende for schleife mithilfe einer globalen Variable in der loop() schleife zu starten, die vom http-handler geändert wird. Somit können die laufenden IDLE-Prozesse nicht timeouten.

4. Neopixel reagierten nicht sofort auf /stop Anfrage

Lösung: Ein return Befehl im Unterprogramm für die Erzeugung der Neopixel-Steuерdaten musste hinzugefügt werden, da sonst das Programm noch ein weiteres Mal vollständig durchlaufen wurde. Dafür wurde der Status der "op"-Variable erneut überprüft, bevor die strip.show() Anweisung erteilt wird.

Arduino-Code

```
    /*Engel_Webserver by Florian Kropf
2 * used for Embedded Systems I project
*
4 *
*
6 * changeColor based on "Christmas Tree Lights"
* by JormaSnow
8 * https://github.com/JormaSnow/NeoPixel_ChristmasLights/blob/master/
    NeoPixel_ChristmasLights.ino
*/
10
11 #include <WiFi.h>
12 #include <AsyncTCP.h>
13 #include <ESPAsyncWebServer.h>
14 #include <Adafruit_NeoPixel.h>
15 #ifdef __AVR__
16 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
17 #endif
18 #define LED 2
19 #define LED_PIN 15 // Pin for Neopixel Data Line
20 #define LED_COUNT 22
21
22 // PARAMETERS--for /random application
23 #define VMAX      150 // the maximum value that can be sent to any one
24           of the RGB channels--255 is the upper limit
# define VMIN      15 // the minimum non-zero value that can be sent to
25           any one of the RGB channels
26 #define SLOW      1500 // slows down the transition between values when
27           changing LED RGB values in microseconds
# define SWITCH_MAX 13 // the max value for the switch statement--should
28           be >7. See comment below
# define SWITCH_MIN 1 // the minimum for the switch statement--should be
29           1. See comment below
// constants for the switch statement
30 #define RED      1
# define GREEN     2
```

```
32 #define RED_AND_GREEN 3
    #define BLUE        4
34 #define RED_AND_BLUE 5
    #define GREEN_AND_BLUE 6
36 #define RGB         7

38 int op = 0;
const char* ssid = "ypor_ssid"; //replace
40 const char* password = "your_pw"; //replace

42 Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);
AsyncWebServer server(80);

44
void setup() {
46 #if defined(__AVR_ATtiny85__)
    clock_prescale_set(clock_div_1);
48#endif

50     strip.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)
    strip.show();             // Turn OFF all pixels ASAP
52     strip.setBrightness(50);

54
    Serial.begin(115200);
56 // Set pin mode
    pinMode(LED, OUTPUT);
58     digitalWrite(LED, LOW);

60 WiFi.begin(ssid, password);

62     Serial.print("Connecting to WiFi..");
    while (WiFi.status() != WL_CONNECTED) {
64         delay(500);
        Serial.print(".");
66     }
    Serial.println();
68     Serial.println("CONNECTED!");

70     Serial.print("IP-Adress: ");
```

```
Serial.print(WiFi.localIP());  
72  
    server.on("/hello", HTTP_GET, [](AsyncWebRequest *request){  
        request->send(200, "text/plain", "Hello World");  
    });  
76  
    server.on("/ON", HTTP_GET, [](AsyncWebRequest *request){  
        request->send(200, "text/plain", "LED ON");  
        digitalWrite(LED, HIGH);  
    });  
80  
  
    server.on("/OFF", HTTP_GET, [](AsyncWebRequest *request){  
        request->send(200, "text/plain", "LED OFF");  
        digitalWrite(LED, LOW);  
    });  
86  
  
    server.on("/blink", HTTP_GET, [](AsyncWebRequest *request){  
        request->send(200, "text/plain", "LED blinking");  
        op = 1;  
90  
    });  
92  
  
    server.on("/light", HTTP_GET, [](AsyncWebRequest *request){  
        request->send(200, "text/plain", "LED white");  
        op = 2;  
96  
    });  
98  
  
    server.on("/random", HTTP_GET, [](AsyncWebRequest *request){  
        request->send(200, "text/plain", "LED random");  
        op = 3;  
102  
    });  
104  
  
    server.on("/stop", HTTP_GET, [](AsyncWebRequest *request){  
        request->send(200, "text/plain", "LED stop");  
        op = 0;  
108  
        //strip.clear();  
        //strip.show();
```

```
110
111     });
112
113
114     server.begin();
115
116 }
117
118 void loop(){
119
120     if(op == 1) {
121         //RGB fade
122         rainbow(10);
123     }
124     else if(op == 2) {
125         //all white
126         light(strip.Color(75, 50, 50));
127     }
128     else if(op == 3) {
129         //random RGB effect
130         switch (random(SWITCH_MIN, SWITCH_MAX)) {
131             case RED: // sends random value to the red led and 0 to others
132                 changeColor(random(LED_COUNT), (uint8_t)random(VMIN, VMAX), 0x00, 0
133                             x00);
134                 break;
135             case GREEN:// sends random value to the green led and 0 to others
136                 changeColor(random(LED_COUNT), 0x00, (uint8_t)random(VMIN, VMAX), 0
137                             x00);
138                 break;
139             case RED_AND_GREEN: // sends random value to the red and green leds
140                 and 0 to blue
141                 changeColor(random(LED_COUNT), (uint8_t)random(VMIN, VMAX), (uint8_t)
142                             random(VMIN, VMAX), 0x00);
143                 break;
144             case BLUE: // sends random value to the blue led and 0 to others
145                 changeColor(random(LED_COUNT), 0x00, 0x00, (uint8_t)random(VMIN,
146                             VMAX));
147                 break;
148             case RED_AND_BLUE: // sends random value to the red and blue leds and
149                 0 to green
```

```
    changeColor(random(LED_COUNT), (uint8_t)random(VMIN, VMAX), 0x00, (uint8_t)random(VMIN, VMAX));
144    break;
    case GREEN_AND_BLUE: // sends random value to the green and blue leds
and 0 to red
146    changeColor(random(LED_COUNT), 0x00, (uint8_t)random(VMIN, VMAX), (uint8_t)random(VMIN, VMAX));
    break;
148    case RGB: // sends random value to the each of the leds
    changeColor(random(LED_COUNT), (uint8_t)random(VMIN, VMAX), (uint8_t)random(VMIN, VMAX), (uint8_t)random(VMIN, VMAX));
    break;
150    default:
152        changeColor(random(LED_COUNT), 0x00, 0x00, 0x00);
        break;
154    }
    }
156 else {
    strip.clear();
158    strip.show();
    }
160
}
162
void rainbow(int wait) {
164
    for(long firstPixelHue = 0; firstPixelHue < 5*65536; firstPixelHue +=
256) {
166    strip.rainbow(firstPixelHue);

    if(op == 1){
        strip.show(); // Update strip with new contents
168    delay(wait); // Pause for a moment
    }
170    else{
        return;
    }
172
    }
174
}
176}
```

```
178 void light(uint32_t color) {
179     for(int i=0; i<strip.numPixels(); i++) { // For each pixel in strip...
180         strip.setPixelColor(i, color);           // Set pixel's color (in RAM)
181     }
182     strip.show();                           // Update strip to match
183 }
184
185 void changeColor(long led, uint8_t newR, uint8_t newG, uint8_t newB) {
186     //capture the current state of the led
187     uint8_t currentR = (strip.getPixelColor(led) >> 16);
188     uint8_t currentG = (strip.getPixelColor(led) >> 8);
189     uint8_t currentB = (strip.getPixelColor(led));
190
191     boolean changing = true;
192     while (changing) {
193         changing = false;
194         if (currentR < newR) {
195             currentR++;
196             strip.setPixelColor(led, currentR, currentG, currentB);
197             changing = true;
198         } else if (currentR > newR) {
199             currentR--;
200             strip.setPixelColor(led, currentR, currentG, currentB);
201             changing = true;
202         }
203
204         if (currentG < newG) {
205             currentG++;
206             strip.setPixelColor(led, currentR, currentG, currentB);
207             changing = true;
208         } else if (currentG > newG) {
209             currentG--;
210             strip.setPixelColor(led, currentR, currentG, currentB);
211             changing = true;
212         }
213
214         if (currentB < newB) {
215             currentB++;
216         }
217     }
218 }
```

```
216     strip.setPixelColor(led, currentR, currentG, currentB);
217     changing = true;
218 } else if (currentB > newB) {
219     currentB--;
220     strip.setPixelColor(led, currentR, currentG, currentB);
221     changing = true;
222 }
223 if(op == 3){
224     strip.show();
225     delayMicroseconds(SLOW); // pauses briefly to slow the overall
226     transition
227 }
228 else{
229     return;
230 }
231 }
232 }
233
234 }
```

Listing 5: Webserver Setup

Referenzen

- [1] Florian Kropf. *Jetzt ist offiziell Weihnachtszeit!!* URL: https://www.reddit.com/r/Fahrrad/comments/za03zb/jetzt_ist_offiziell_weihnachtszeit/.
- [2] Adafruit. *Arduino Library Use.* URL: <https://learn.adafruit.com/adafruit-neopixel-uber-guide/arduino-library-use>.
- [3] MIT. *MIT App Inventor.* URL: <http://appinventor.mit.edu/about-us>.
- [4] Adfruit. *powering-neopixels.* URL: <https://learn.adafruit.com/adafruit-neopixel-uber-guide/powering-neopixels>.
- [5] Random Nerd Tutorials. *Failed to connect to ESP32: Timed out waiting for packet header.* URL: <https://randomnerdtutorials.com/solved-failed-to-connect-to-esp32-timed-out-waiting-for-packet-header/>.