



ONSITE XMAS: PROGRAMMING GUIDE

T-Systems on site Services GmbH

Version: 1.0

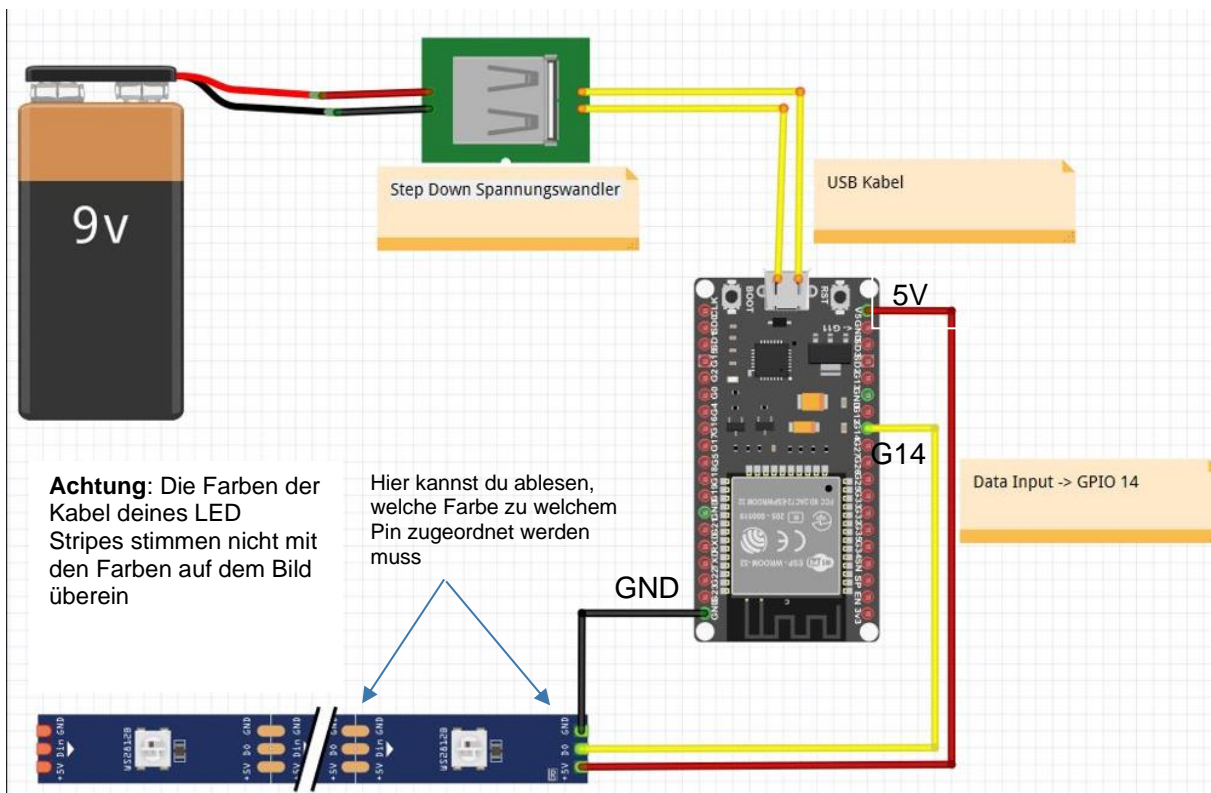
Onsite Xmas

Inhaltsverzeichnis

1	Setup.....	3
1.1	Schaltplan	3
1.2	Software	3
1.3	Hardware	4
1.4	Sag Hallo, zur Welt!.....	5
2	Der Sketch	7
2.1	WiFi-Setup und Netzwerkdetails:	7
2.2	LED-Initialisierung:.....	7
2.3	Webserver-Routing und Setup:	7
2.4	LED-Steuerung (Ein/Aus):.....	7
2.5	Farbe und Helligkeit setzen:.....	8
2.6	HTML-Webseite	8
2.7	Breathing Mode	9
3	Der erste Test.....	10
4	Die aufgabe	11
5	Bauanleitung	13
A.1	LED-Stripe: Nützliche Funktionen	15

1 SETUP

1.1 Schaltplan



ACHTUNG: Für die Entwicklung muss der ESP32 nur mit deinem LED Stripe verbunden werden, da die Stromversorgung über das Micro-USB Kabel erfolgt, welches in deinem Notebook steckt.

Erst wenn die Weihnachtsbaumkugel zusammengebaut wird, wird die Stromversorgung über die Batterie und den Spannungswandler benötigt.

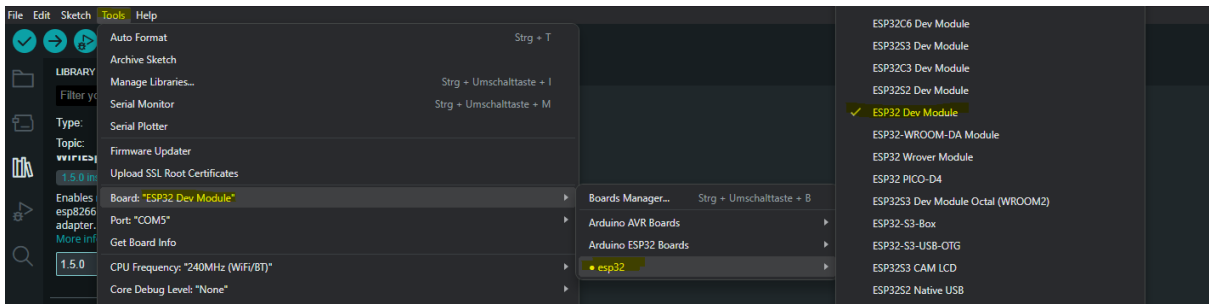
1.2 Software

Arduino IDE

Bevor du die Arduino IDE zum Programmieren des ESP32 verwenden kannst, musst du zunächst das ESP32-Board über den Arduino Board Manager installieren. Eine Anleitung findest du hier:

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

Folgendes „Borad“ muss anschließend ausgewählt werden:



Bibliotheken für LED-Steuerung und den Webserver

Über den Bibliotheks-manager müssen folgende Bibliotheken installiert werden.

- ESP Async Webserver by Me-No-Dev
- AsyncTCP by Me-No-Dev
- Adafruit Neopixel by Adafruit

1.3 Hardware

Die Wichtigsten Komponenten im Überblick:

ESP 32



Der ESP32 ist eine kostengünstige und mit geringem Leistungsbedarf ausgeführte 32-Bit-Mikrocontrollerfamilie der Firma espressif. Er kommt standardmäßig mit Wifi und BLE an Bord.

Power-supply Modul



Wird benötigt, um die 9 Volt des Akkus auf konstante 5 Volt zu bringen, so kommt es nicht zu Verbindungsabbrüchen des BLE Chips. Der Spannungswandler verfügt über einen USB Output, wodurch eine einfache Stromversorgung des ESP32 ermöglicht wird.

LED Stripe



WS2812b LED Stripe: Bei den WS2812 LEDs handelt es sich um adressierbare RGB-LEDs. Sie verfügen über einen integrierten Chip und belegen daher nur einen einzigen digitalen Output

Liste deiner Hardware:

- ESP32
- LED-Strip (WS2812 oder kompatibel).
- 9V Batterie
- Micro USB Kabel
- Christbaumkugel
- Stepdown Spannungswandler
- Magenta Einlegescheibe

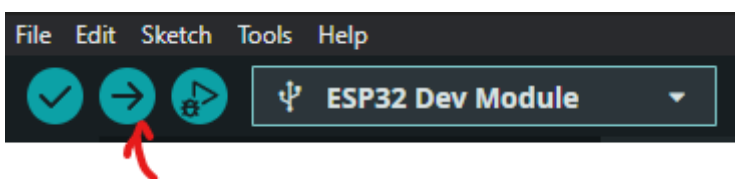
1.4 Sag Hallo, zur Welt!

Zum Testen der Konfiguration kann ein kleiner „Hello-World“ Sketch implementiert werden. Hierzu muss über „Datei -> Neu“ ein neuer Sketch angelegt werden. Mithilfe des in dargestelltem Sketch kann eine Serielle Ausgabe geschrieben werden, welche im Seriellen Monitor verfolgt werden kann. Dabei ist darauf zu achten, dass die Baudrate im Seriellen Monitor mit der im Sketch hinterlegten Baudrate (115200) übereinstimmt. (Serieller Monitor kann über „Tools -> Serieller Monitor“ gestartet werden)

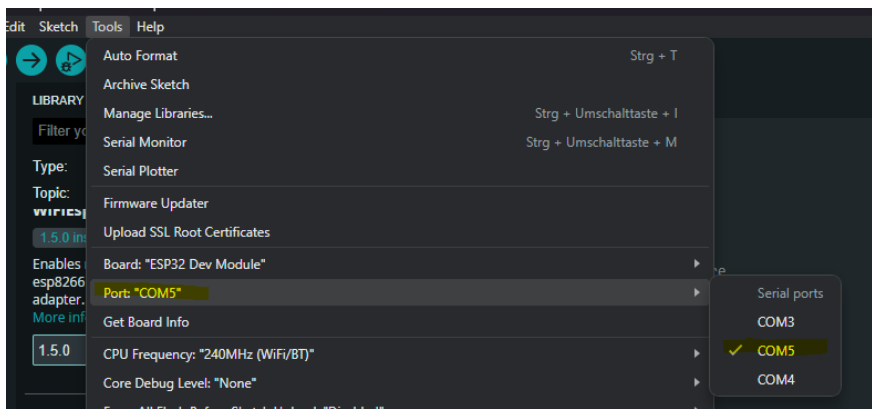
```
void setup() {
  Serial.begin(115200);
  Serial.println("Hello ESP32 World!");
}

void loop() {
  Serial.println("Hello");
  delay(500);
}
```

Will man den Sketch auf das Board flashen, klickt man auf den Pfeil „Hochladen“ (oben links in der IDE). Daraufhin wird der Sketch kompiliert und das Flashen beginnt. Will man lediglich kompilieren, klickt man auf den Haken „Überprüfen“.



Zuvor musst du jedoch das richtige Board und den richtigen Port auswählen. Welcher COM Port der richtige ist, findest du ganz einfach heraus indem du den USB Kabel wieder aussteckst, und schaust, welcher COM Port verschwindet. Sollte kein Port erkannt werden kann es sein, dass der CP210x-Treiber nachinstalliert werden muss.



Den Treiber findest du hier:

<https://www.silabs.com/developer-tools/usb-to-uart-bridge-vcp-drivers?tab=downloads>

Genauer Details findest du außerdem unter „Troubleshooting“ in diesem Artikel:

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

Während des Flashvorgangs wird am unteren Bildschirm der aktuelle Vorgang in der Konsole ausgegeben. Sobald dort „Connecting...“ steht, muss in einigen Fällen der „Boot“ Knopf des ESP32 gedrückt werden, bis die Übertragung startet. Danach kann der Knopf losgelassen werden.

```
Globale Variablen verwenden 52876 Bytes
esptool.py v2.6
Serial port COM3
Connecting...
```

Genauer Details findest du unter „Troubleshooting“ in diesem Artikel:

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

Für Linux Nutzer:

Unter Linux muss zunächst die Versionskontrollsoftware Git und das Python-Modul Serial mittels

```
sudo apt install git python-serial
```

installiert werden.

2 DER SKETCH

Um deine Weihnachtskugel zu steuern, wird eine HTML-basierte UI auf dem ESP32 bereitgestellt. Diese kann über die zugewiesene IP, welche beim Starten im Seriellen Monitor ausgegeben wird, erreicht werden. Über das WLAN-Netzwerk können verschiedene LED-Funktionen (Ein/Aus, Helligkeit, Farbe und Modi) über die UI gesteuert werden. Im Hintergrund werden verschiedene http-Endpunkte aufgerufen, welche gewisse Funktionen ausführen.

Achtung: Der Microcontroller muss sich im selben Netzwerk befinden wie dein Client, mit dem du die Weihnachtskugel steuern möchtest.

2.1 WiFi-Setup und Netzwerkdetails:

Hier wird die WLAN-SSID und das Passwort definiert, um eine Verbindung zu einem WLAN-Netzwerk herzustellen. Die IPAddress-Objekte legen die statische IP-Adresse, das Gateway und das Subnetz des ESP32 fest.

```

5  /* Put your SSID & Password */
6  const char* ssid = ""; // Enter SSID here
7  const char* password = "!"; //Enter Password here

```

2.2 LED-Initialisierung:

Die LED-Leiste wird mit Adafruit_NeoPixel initialisiert. LED_PIN gibt den GPIO-Pin an, der mit der LED-Leiste verbunden ist, und LED_COUNT gibt die Anzahl der LEDs an.

```

14 /* LED Configuration */
15 #define LED_PIN 14 // Define pin for LED strip
16 #define LED_COUNT 12 // Number of LEDs
17 Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);

```

2.3 Webserver-Routing und Setup:

Zunächst wird der Webserver gestartet und dann mehrere Routen registriert, die auf HTTP-GET-Anfragen reagieren. Jede Route ist einer Funktion zugeordnet, die eine spezifische Aktion auslöst (z.B. LEDs ein-/ausschalten, Farbe ändern).

```

42 server.on("/", HTTP_GET, handle_OnConnect);
43 server.on("/led/on", HTTP_GET, handle_led_on);
44 server.on("/led/off", HTTP_GET, handle_led_off);
45 server.on("/led/color", HTTP_GET, handle_set_color);
46 server.on("/led/brightness", HTTP_GET, handle_set_brightness);
47 server.on("/led/breathingMode/on", HTTP_GET, handle_breathingMode_on);
48 server.on("/led/breathingMode/off", HTTP_GET, handle_breathingMode_off);
49 server.on("/led/customMode/on", HTTP_GET, handle_custom_mode_on);
50 server.on("/led/customMode/off", HTTP_GET, handle_custom_mode_off);
51 server.onNotFound(handle_NotFound);

```

2.4 LED-Steuerung (Ein/Aus):

Diese Funktionen steuern das Ein- und Ausschalten der LEDs. Wenn LEDs eingeschaltet werden, werden sie auf Magenta gesetzt, und wenn sie ausgeschaltet werden, wird die Farbe auf

„Schwarz“ gesetzt, um sie auszuschalten. Nach dem setzen der Farbe, wird die neue Konfiguration mit „*show()*“ auf den LED-stripe übertragen

```

100 void handle_led_on(AsyncWebServerRequest *request){
101     ledStatus = true;
102     Serial.println("LED Status: ON");
103     strip.fill(strip.Color(255, 255, 255)); // White color
104     strip.show();
105     request->send(200, "text/html", SendHTML(ledStatus));
106 }
107
108 void handle_led_off(AsyncWebServerRequest *request) {
109     ledStatus = false;
110     Serial.println("LED Status: OFF");
111     strip.fill(strip.Color(0, 0, 0)); // Off
112     strip.show();
113     request->send(200, "text/html", SendHTML(ledStatus));
114 }

```

2.5 Farbe und Helligkeit setzen:

handle_set_color legt die Farbe der LEDs basierend auf den RGB-Werten fest, die als URL-Parameter übergeben werden. Hierbei gibt es drei URL Parameter jeweils für rot, grün und blau. *handle_set_brightness* ändert die Helligkeit und prüft, ob der Helligkeitswert als Parameter bereitgestellt wurde. Falls ja, wird die Helligkeit geändert „*strip.setBrightness(brightness)*;“ und über „*strip.show()*;“ aktiviert.

```

116 void handle_set_brightness(AsyncWebServerRequest *request){
117     if (request->hasArg("brightness")) {
118         int brightness = request->arg("brightness").toInt();
119         Serial.println("Brightness received: " + String(brightness)); // Debugging log
120         strip.setBrightness(brightness);
121         strip.show(); // Apply the brightness change to the LED strip
122         request->send(200, "text/plain", "Brightness updated");
123     } else {
124         request->send(400, "text/plain", "Brightness level not provided");
125     }
126 }
127
128 void handle_set_color(AsyncWebServerRequest *request) {
129     if(ledStatus == true){
130         // Get the RGB value from the URL
131         String r = request->arg("r");
132         String g = request->arg("g");
133         String b = request->arg("b");
134         Serial.println("RGB received: " + r + "," + g + "," + b); // Debugging log
135
136         setColor(strip.Color(r.toInt(), g.toInt(), b.toInt()));
137         request->send(200, "text/html", SendHTML(true)); // Send back the updated HTML
138     }else{
139         Serial.println("LEDs are off, first turn LED ons");
140         request->send(403, "text/plain", "Error: LED is currently off. Turn on the LED first.");
141     }
142 }

```

2.6 HTML-Webseite

Die Webseite zum Steuern der Christbaumkugel wird auf dem ESP32 definiert und gehostet. Sobald sich ein Client verbindet, wird die HTML Webseite übertragen.

```

64 void handle_OnConnect(AsyncWebServerRequest *request){
65     Serial.println("Client connected");
66     request->send(200, "text/html", SendHTML(ledStatus));
67 }

```



```

181 String SendHTML(bool ledStat) {
182     String html = "<!DOCTYPE html>\n"
183     "    <html lang='en'>\n"
184     "    <head>\n"
185     "        <meta name='viewport' content='width=device-width, initial-scale=1.0'>\n"
186     "        <title>Onsite Xmas</title>\n"
187     "        <style>\n"
188     "            /* General styling */\n"
189     "            html, body {\n"
190     "                font-family: Arial, sans-serif;\n"
191     "                display: flex;\n"
192     "                justify-content: center;\n"
193     "                align-items: center;\n"
194     "                min-height: 100vh;\n"
195     "                margin: 0;\n"
196     "                background-color: #000;\n"

```

2.7 Breathing Mode

Der **breathing_mode** wird in der `loop()`-Funktion aufgerufen, damit die LEDs den sanften Ein- und Ausblendeeffekt (das „Atmen“) flüssig und kontinuierlich darstellen können. Da dieser Effekt, im Gegensatz zum einfachen setzen einer Farbe, aus einer regelmäßigen Anpassung der Helligkeit besteht, muss er ständig aktualisiert werden, solange der Modus aktiv ist.

Ob die Animation tatsächlich ausgeführt wird, wird durch das `breathing`-Flag gesteuert. Dieses Flag wird in den Funktionsaufrufen `handle_breathingMode_on` und `handle_breathingMode_off` gesetzt (`true/false`), wenn der Breathing-Modus aktiviert (`true`) oder deaktiviert (`false`) wird:

- Wenn `breathing` auf `true` gesetzt ist (also `handle_breathingMode_on` aufgerufen wurde), wird die Animation in der `loop()` gestartet.
- Ist `breathing` auf `false`, wird die `breathing_mode`-Funktion in der `loop()` ignoriert.

```

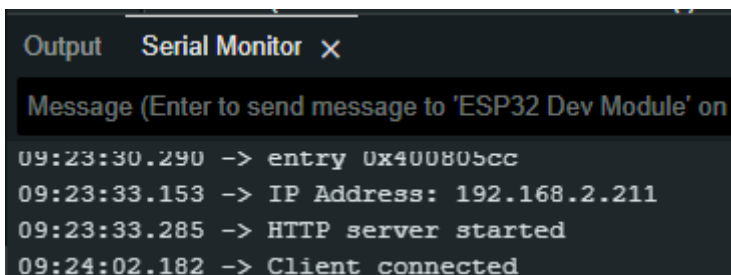
57 void loop() {
58     //server.handleClient();
59     if (breathing) {
60         breathing_mode();
61     }
62 }

```

3 DER ERSTE TEST

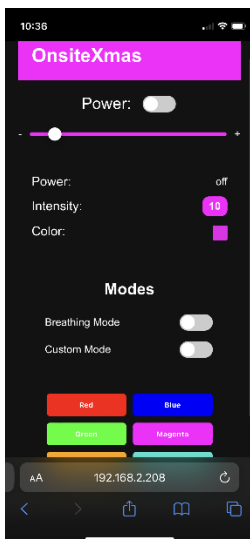
Da du dich nun mit dem vorhandenen Sketch auseinandergesetzt hast, wird es Zeit diesen zu testen. Dafür musst du zunächst den richtigen COM-Port auswählen und anschließend den Code kompilieren und Hochladen.

Anschließend kannst du auf deinem Handy oder deinem Notebook die IP-Adresse aufrufen, welche in deinem Seriellen Monitor erscheint. (Achte darauf das der Browser http und nicht https nutzt). Mit dem Reset Knopf auf deinem ESP32 (RST), kannst du diesen neu starten. Das ist hilfreich, sollte nach dem Upload Vorgang, keine IP in deinem seriellen Monitor stehen.



```
Output  Serial Monitor X
Message (Enter to send message to 'ESP32 Dev Module' on 'COM3')
09:23:30.290 -> entry 0x400805cc
09:23:33.153 -> IP Address: 192.168.2.211
09:23:33.285 -> HTTP server started
09:24:02.182 -> Client connected
```

Anschließend kannst du die Weihnachtskugel über die Webseite steuern.



4 DIE AUFGABE

In dieser Aufgabe kannst Du zwei Funktionen implementieren, die über HTTP-Endpunkte aufgerufen werden, um einen benutzerdefinierten LED-Modus zu steuern.

1. `handle_custom_mode_on`: Diese Funktion wird aufgerufen, wenn Du den benutzerdefinierten Modus aktivieren möchtest. Du kannst entweder eine einzelne Farbe für alle LEDs festlegen oder eine einfache Animation wie das "Snake"-Lichtspiel implementieren, bei dem jede LED nacheinander ein- und ausgeschaltet wird. Du hast hier die Freiheit, zu entscheiden, wie die Animation aussehen soll – Tob dich einfach aus!
2. `handle_custom_mode_off`: Diese Funktion wird aufgerufen, wenn der benutzerdefinierte Modus deaktiviert werden soll. Hier solltest Du entweder alle LEDs auf eine Standardfarbe setzen oder sie einfach ausschalten, je nachdem, wie Du es für sinnvoll hältst.

Du kannst Dich für ein einfaches Lichtspiel entscheiden oder kannst eine andere Animation wählen, die die LEDs in einem bestimmten Muster steuert. Der Kreativität sind keine Grenzen gesetzt.

```

70
71 void handle_custom_mode_on(AsyncWebServerRequest *request) {
72     // TO BE DONE
73 }
74
75
76 void handle_custom_mode_off(AsyncWebServerRequest *request) {
77     // TO BE DONE
78 }

```

Sobald der Endpunkt „/led/customMode/on“ oder „/led/customMode/off“ aufgerufen wird, wird die Funktion aufgerufen und dein Lichtspiel wird gestartet oder beendet.

```

44 server.on("/", HTTP_GET, handle_OnConnect);
45 server.on("/led/on", HTTP_GET, handle_led_on);
46 server.on("/led/off", HTTP_GET, handle_led_off);
47 server.on("/led/color", HTTP_GET, handle_set_color);
48 server.on("/led/brightness", HTTP_GET, handle_set_brightness);
49 server.on("/led/breathingMode/on", HTTP_GET, handle_breathingMode_on);
50 server.on("/led/breathingMode/off", HTTP_GET, handle_breathingMode_off);
51 server.on("/led/customMode/on", HTTP_GET, handle_custom_mode_on);
52 server.on("/led/customMode/off", HTTP_GET, handle_custom_mode_off); |
53 server.onNotFound(handle_NotFound);

```

Auf der Webseite geschieht dies, sobald der toggle „custom-mode“ betätigt wird.

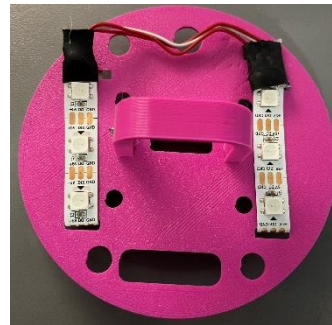
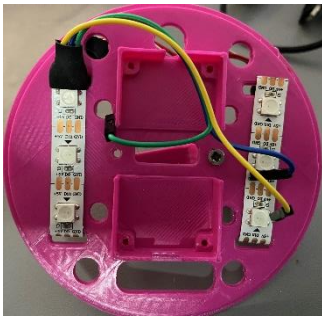
```
472     "    function toggleCustomMode() {\n"
473     "    const isChecked = document.getElementById('custom-mode').checked;\n"
474     "    fetch(`/led/customMode/${isChecked ? 'on' : 'off'}`)\n"
475     "    .then(response => {\n"
476     "    if (response.ok) {\n"
477     "    showToast('Custom Mode set successfully!');\n"
478     "    }else {\n"
479     "    showToast('Error set Custom Mode!');\n"
480     "    }\n"
481     "    }}\n"
482     "    .catch(error => showToast(`Error: ${error.message}`, true));\n"
483     "    }\n"
```

Viel Spaß beim Programmieren!

5 BAUANLEITUNG

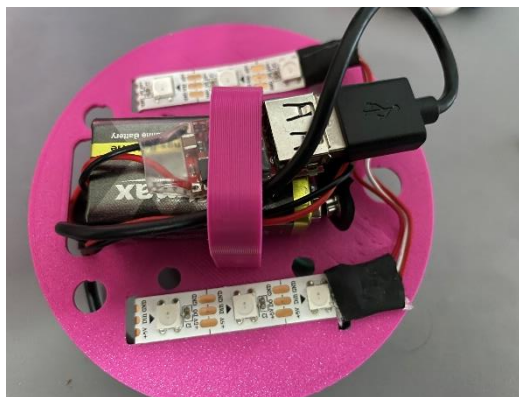
Schritt 1:

im ersten Schritt kannst Du deinen LED Stripe durch die Öffnungen ziehen, so dass sowohl hinten als auch vorne jeweils 3 LED's auf jeder Seite die Christbaumkugel beleuchten. Ziehe dabei am besten nicht an den Kabeln, da sonst die Lötstellen reißen können.



Schritt 2:

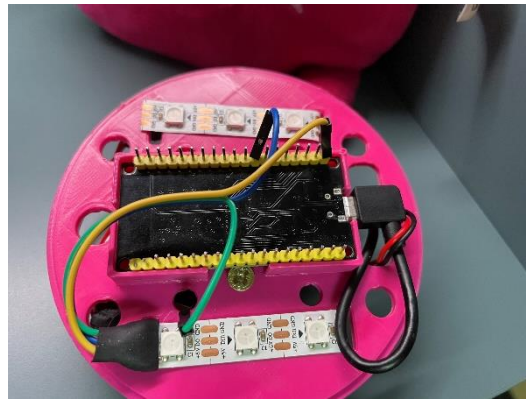
Nun kannst Du die Batterie Halterung an die „Scheibe“ schrauben. Anschließend kannst du die Batterie, sowie den Spannungswandler in der Halterung einklemmen. Der Spannungswandler kann auch mit Doppelseitigem Klebeband am 9V Block fixiert werden. Die Kabel kannst Du legen, wie du möchtest. Am besten ist es wenn man diese mit in der Halterung fixiert oder sie anderweitig z. B. mit Klebeband festklebt, damit diese später keinen Schatten werfen.



Schritt 3:

Im diesem Schritt kann der ESP32 Mikrocontroller in die Halterung geklickt werden. Dieser sollte fest genug sein und nicht rausfallen. Achte dabei darauf, dass der Micro-USB Port in Richtung der großen Öffnung zeigt.

Der Micro-USB Kabel, welcher am Spannungswandler eingesteckt wird, muss durch die Öffnung auf die andere Seite gelegt werden, um ihn dann in den ESP32 einzustecken.



A.1 LED-Stripe: Nützliche Funktionen

<code>Adafruit_NeoPixel strip(LED_ANZAHL, LED_PIN, NEO_GRB + NEO_KHZ800);</code>	Erstellt das Objekt "stripe" vom Typ "Adafruit_NeoPixel". Parameter: <ul style="list-style-type: none"> - LED_ANZAHL: Anzahl der LEDs auf dem Stripe. - LED_PIN: Date input Pin am Board. - NEO_GRB + NEO_KHZ800: Konstanter Wert.
<code>strip.begin();</code>	Startet den LED-Stripe.
<code>strip.show();</code>	Änderungen am LED-Stripe übernehmen. Z.B. bei neuer Farbe.
<code>strip.setBrightness(int HELBIGKEIT);</code>	Setzt die Helligkeit der LEDs. Wert von 0 – 255.
<code>strip.Color(RED, GREEN, BLUE);</code>	Erstellt Farbcode anhand von RGB-Code. Gibt den Farbcode als uint32_t zurück.
<code>Strip.setPixelColor(int POSITION_LED, uint32_t FARBE);</code>	Setzt LED an POSITION_LED auf Farbe FARBE.
<code>strip.fill(uint32_t FARBE, VonLED, BisLED);</code>	Setzt mehrere LEDs auf Farbe FARBE
<code>strip.clear();</code>	Schaltet alle LEDs aus. Funktioniert nur in Verbindung mit stripe.show().