Kalenderwoche 9 (ca. 3 Stunden)

Das primäre Ziel in Kalenderwoche neun war die Erarbeitung eines Konzepts und die Niederschrift dessen zu einem Exposé. Das genaue Konzept kann dem Exposé entnommen werden.

Kalenderwoche 14 (ca. 2 Stunden)

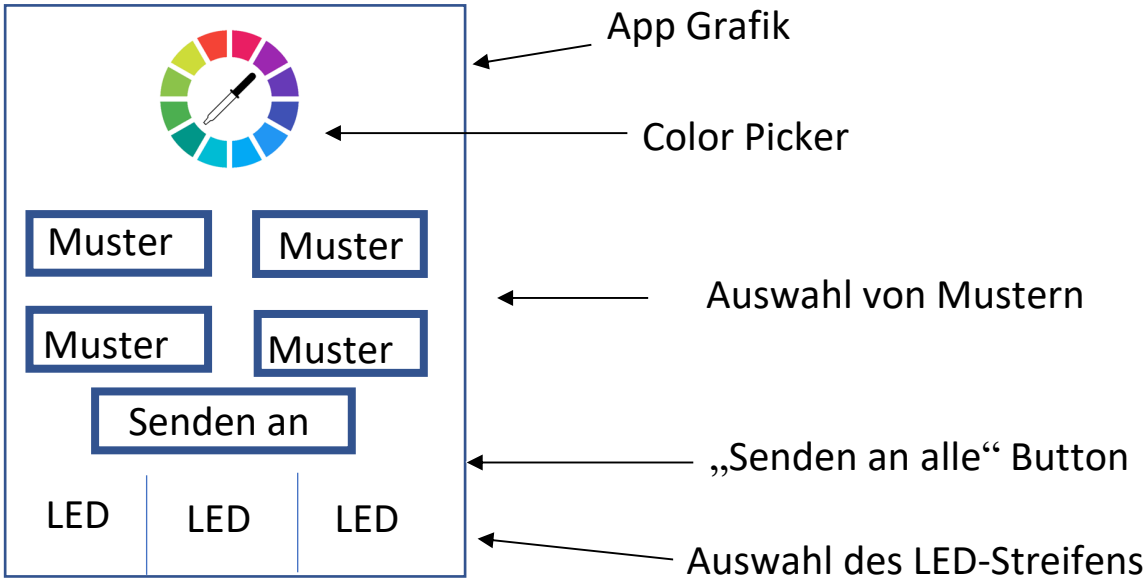
In Kalenderwoche 14 habe ich mir einen Überblick über den schriftlichen Teil meines Projektes gemacht. Außerdem fand ein Beratungstermin statt, in dem die Einzelheiten des schriftlichen Teiles erörtert wurden. Dieser umfasst die Langzeitdokumentation, die Werkanalyse, die Darstellung der das Projekt betreffenden aktuellen und historisch technischen Entwicklungen, die Übersicht verwendeter Materialien, ein Promo-Video sowie die selbstkritische Analyse.

Kalenderwoche 17 (ca. 2 Stunden)

Da mein Projekt aus mehreren „Baustellen“ besteht, habe ich beschlossen, diese zu gliedern und der Reihe nach abzuarbeiten. Das sind im Einzelnen:

1. die App Programmierung,
2. die Hardware (darunter fallen die LED-Streifen, inklusive dem Mini Computer, Netzteil etc.) und
3. den bereits in Kalenderwoche 14 angesprochenen schriftlichen Teil. Was genau darunter fällt, kann dort entnommen werden.

In Kalenderwoche 17 habe ich mit dem ersten Teil begonnen, der App Programmierung. Dazu habe ich mir zuerst Skizzen gemacht, wie genau ich mir den grafischen Aufbau der App vorstelle und wie die Funktionsweise sein soll (Wie werden die LED-Streifen, die man ansteuern will, ausgewählt? Mit einem Button? Einem Drop-Down Menu? Wie soll die Farbe geändert werden? Mit einem Farbkreis, auf dem man die gewünschte Farbe auswählt? Mit einem Button für jede Farbe? etc.). All diese Fragen habe ich mir dabei gestellt. Die Ergebnisse können der Skizze entnommen werden. Die Skizze wird auf der nächsten Seite erläutert.



Die Skizze stellt die App Oberfläche da, allerdings rein schematisch. Die Skizze spiegelt nicht das Aussehen der App wider.

Das Ziel des Projektes ist, am Ende mehrere LED-Streifen gleichzeitig oder einzeln anzusteuern. Daher wird zuerst in der App der gewünschte LED-Streifen gewählt, dies geschieht in der unteren Hälfte. Alternativ kann auch die Checkbox „Senden an alle“ aktiviert werden, ist diese aktiviert, werden alle verbundenen LED-Streifen gleichzeitig gesteuert, anstatt nur die eine unten ausgewählte LED (LED ist hier gleichbedeutend mit LED-Streifen). Diese Option kann jeder Zeit aktiviert oder deaktiviert werden.

Hat man nun einen LED-Streifen ausgewählt (oder alle) kann man von diesem mit dem „Color Picker“ die Farbe ändern (durch die Auswahl auf einem Color Picker, wie genau dieser dann aussieht, rund, eckig, wie ein Zebra etc. entscheide ich zu einem späteren Zeitpunkt). Außerdem können bereits vorprogrammierte Muster aktiviert werden, durch drücken einer der Muster-Button. Sollte man ein Muster auswählen wird natürlich das Muster auf dem LED-Streifen erscheinen, auch wenn vorher eine Farbe gewählt wurde. Ein Befehl überschreibt den vorherigen. Dies gilt natürlich auch andersrum: Ist auf der LED-Leiste ein Muster eingestellt und man klickt auf eine Farbe im „Color-Picker“ wird diese Farbe angezeigt auf dem Streifen, statt dem Muster. Einen Helligkeitsregler gibt es auch, der ist allerdings selbsterklärend. Wichtig: Egal ob Helligkeit, Muster oder Farbe, alle Änderungen finden nur auf dem davor ausgewählten LED-Streifen statt (oder auf allen, wenn diese Funktion aktiviert wurde).

Soweit zumindest die Theorie und meine Vorstellung. Für diese Art der Darstellung (Aufbau der App etc.) habe ich mich entschieden, da man auf einer Oberfläche die zu steuernde LED auswählen kann und diese auch zugleich steuern kann, es gibt nicht mehrere Instanzen („Seiten“) in der App, sondern nur diese eine, wo übersichtlich alle Funktionen aufgeführt sind.

Kalenderwoche 18 (ca. 7 Stunden)

Anschließend habe ich mich mit der Programmier-Umgebung beschäftigt (weiterhin „Baustelle 1“). Dabei habe ich mir erstmal alles Nötige Installiert, genutzt wird Android Studio, da dies der etablierte Standard ist zur Android App Programmierung. Android Studio ermöglicht das Editieren sowie die Verwaltung des programmierten Quellcodes, und das Erstellen der Layouts für die Bedienelemente, sowie die Übersetzung des in Java erstellten Programmes in den maschinenlesbaren Code.

Um sich in die Programmierumgebung und die Programmiersprache einzuarbeiten habe ich mir viele Anleitungen auf YouTube angeschaut sowie Artikel im Internet dazu gelesen. Das begann mit einfachen Buttons, Countern bis hin zur Programmierung von kleinen Taschenrechnern.

Kalenderwoche 19 (ca. 6 Stunden)

Nachdem ich mich allgemein mit dem Programmieren befasst habe, habe ich nun damit begonnen mich mit dem Thema des Sendens eines Befehles zu beschäftigen. Denn die App muss dem LED-Streifen mitteilen was er anzeigen soll. Das Handy soll hierbei nur eine Zahlenfolge an den LED-Streifen schicken. Der LED-Streifen wiederum (besser gesagt der Mini Computer, der an dem Streifen angeschlossen ist) weiß dann, wie der Zahlencode interpretiert werden muss (also z.B. ob er rot leuchten muss oder gelb oder die Helligkeit reduzieren soll). Für dieses System mit den Zahlencodes habe ich mich entschieden, da dies mir nach Recherchen als der einfachste und beste Weg erschien, um möglichst schnell und ohne große Datenmengen zu übertragen, alle Funktionen steuern zu können. Alles andere wäre technisch komplizierter und ineffizienter.

Doch wie sende ich von dem Handy, wenn ich einen Button drücke (z.B. den „Muster 1“ Button) einen Zahlencode (=Zahlenfolge) an einen LED-Streifen?

Dazu habe ich mich eingelesen und bin letztendlich auf zwei gängige Standards gestoßen. Das ist einmal der UDP und der TCP Übertragungs-Standard. Beide ermöglichen es meinen Zahlencode vom Handy an den Led-Streifen (bzw. den Mini Computer an ihm) zu senden. Da UDP technisch leichter umzusetzen ist, habe ich mich für diesen entschieden.

Also habe ich mich im Rest der Woche damit beschäftigt wie ich Daten von einer Android App per UDP senden kann. Als Empfänger diente hier zu Testzwecken ein Windows Computer, in Zukunft wird der Empfänger der Zahlenfolge natürlich der Mini Computer des LED-Streifens sein, den gibt es nur noch nicht. Bis ich allerdings herausgefunden hatte, wie ich Befehle von meiner App per UDP sende, verging eine Woche, da erstmal der richtige Code recherchiert werden musste, dabei erlitt ich viele Rückschläge, bis ich endlich den richtigen Programmcode erstellt hatte.

Hier ein Screenshot des Codes:

```
private class AsyncTaskRunner extends AsyncTask<String, String, String> {

    private String resp;
    public String Message="Wert=";
    DatagramSocket ds = null;
    DatagramPacket dp;

    @Override
    protected String doInBackground(String... params) {
        try {
            Message=Message+params[0];
            resp = "gesendet:"+Message;
            ds = new DatagramSocket();
            dp = new DatagramPacket(Message.getBytes(), Message.length(), InetAddress.getByName("192.168.4.99"), port: 1234);
            ds.setBroadcast(true);
            ds.send(dp);
        } catch (Exception e) {
            e.printStackTrace();
            resp = e.getMessage();
        }
        finally
        {
            if (ds != null)
            {
                ds.close();
            }
        }
        return resp;
    }
```

Kalenderwoche 21 (ca. 2 Stunden)

Nachdem es mir gelungen ist Zahlenfolgen von der App über WLAN (UDP) an einen Computer zu schicken (wie oben beschrieben) habe ich mich damit beschäftigt, wie diese Zahlenfolgen aussehen könnten, um Befehle für Muster bzw. Farbinformationen zu senden. Dazu habe ich mir dann folgendes System überlegt:

Die erste Zahl in der Zahlenfolge gibt die Art des Befehls an, der ausgeführt werden soll. So steht der Zahlenwert „1“ für die Helligkeit der LEDs, „2“ für die Farbe des LED-Streifens und die „3“ für das Muster (Animation), das dargestellt werden soll.

Die Helligkeit wird mit einem Zahlenwert zwischen 000 und 255 definiert. Also 000 ist dunkel (also ausgeschaltet) und 255 ist volle Helligkeit, Abstufungen dazwischen sind prozentual mit der Helligkeit gekoppelt. Würde ich jetzt z.B. die Helligkeit auf 220 (=86% Helligkeit) stellen wollen, würde das Handy den Wert 001 220 schicken. Die 1 als Befehl für die Helligkeit und schließlich die 220 als Wert für die Helligkeit.

Bei der Farbe ist es etwas komplizierter, da eine Farbe aus drei Helligkeitswerten besteht. Jede Farbe wird bei LEDs aus Rot, Grün und Blau zusammengesetzt. In meinem Zahlensystem werden die Helligkeitswerte der einzelnen Farben aneinander gereiht in der Reihenfolge Rot, Grün, Blau auch jeweils von 000 bis 255 angegeben. Um die Farbe Rot einzustellen würde die Zahlenfolge 002 255 000 000 gesendet werden. Die 002 am Anfang um zu signalisieren, dass die Farbe verändert werden soll, und die 255 danach für Rot gleich volle Helligkeit und Grün und Blau jeweils 000. Blau wäre z.B. 002 000 000 255. Oder Lila wäre 002 132 000 255, also eine Mischung aus Rot (Helligkeit 132) und Blau (Helligkeit 255). Der Helligkeitsregler funktioniert genauso, nur dass er alle Farben prozentual gleich Heller oder Dunkler macht, um die Farbe an sich nicht zu verändern, sondern nur dessen Helligkeit.

Durch dieses System weiß die App genau, wann sie welche Zahlenfolge schicken muss und der LED-mini Computer weiß genau bei welcher Zahlenfolge er was verändern muss (z.B. ob er die Farbe Rot setzen soll, Blau oder die Helligkeit verändern soll). Natürlich muss man das erst im Mini-Computer des LED-Streifens programmieren (ihm die korrekte Interpretation der Zahlenfolge „beibringen“), aber das kommt erst danach. Die genaue Programmierung, wann die App welchen Befehl schicken muss, erfolgt später.

Diese oben beschriebenen Zahlenfolgen werden nur zur Kommunikation zwischen dem Handy und dem Mini Computer des LED-Streifens benutzt.

Kalenderwoche 24

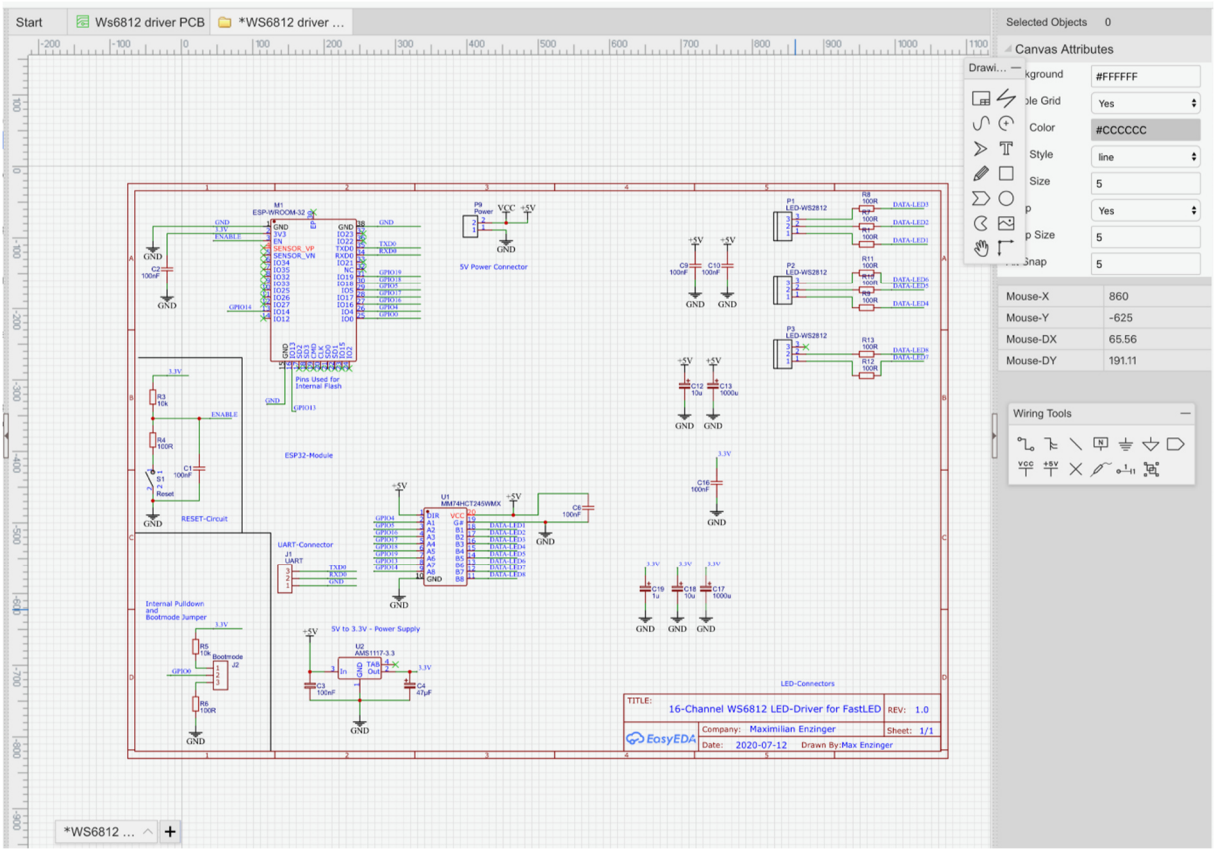
Nun eine kurze Bestandsaufnahme was ich habe und was mir noch fehlt:

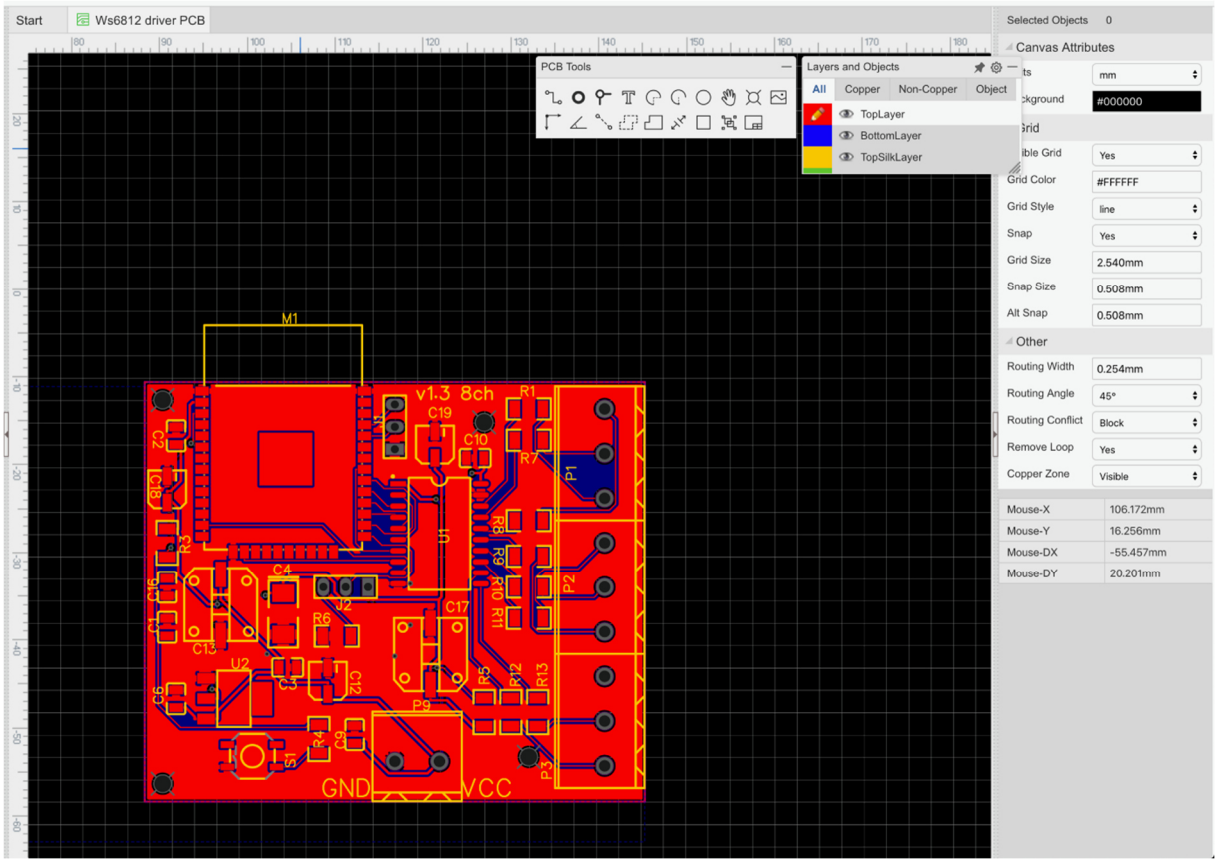
Der komplette Programmcode fehlte natürlich noch, also all das was man als Nutzer nicht direkt sieht. Beim Code habe ich bis jetzt nur die Vorlage von Kalenderwoche 19, aber den muss ich nun anpassen auf meine App, sodass immer der richtige Befehl zu den LEDs gesendet wird. In Kalenderwoche 19 hatte ich mich nur damit beschäftigt, wie ich überhaupt irgendwas über UDP schicke. Wie in Kalenderwoche 21 erklärt, muss allerdings ständig ein anderer Befehlscode gesendet werden, je nachdem welche Helligkeit, welche Farbe oder welches Muster gewählt wird. Dies muss natürlich alles noch Programmiert werden. Aktuell habe ich somit nur die Idee meines Designs und eine Code Vorlage für das allgemeine Senden eines Befehls, wie oben beschrieben. Es fehlt also noch der Code hinter der Grafik (ebenfalls oben beschrieben), die Umsetzung der Grafik, also das Designen der App und als Letztes die 2. „Baustelle“, wie anfangs definiert, also der physische Bau der LED-Streifen.

Kalenderwoche 25 (ca. 12 Stunden)

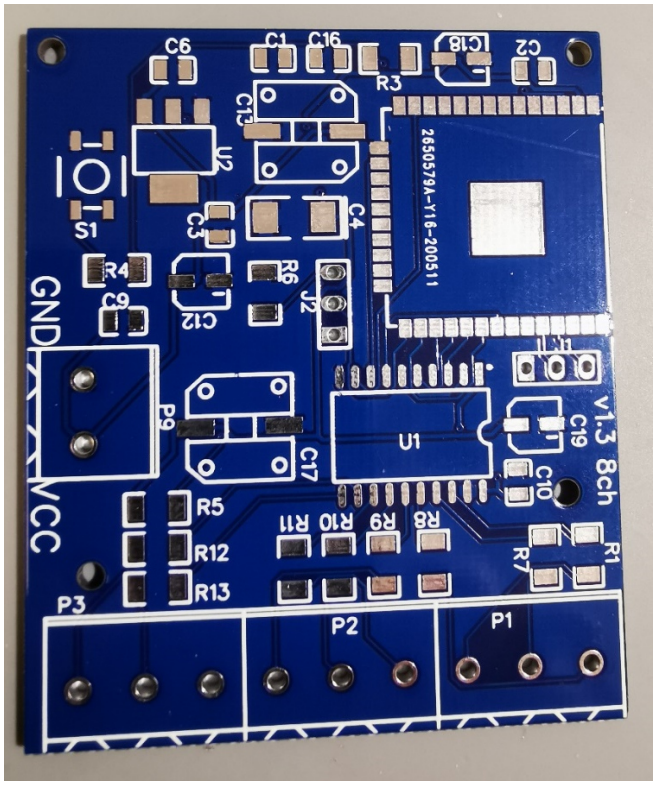
Was in meinem Projekt noch fehlt, habe ich bereits in Kalenderwoche 24 beschrieben, bevor ich den Code allerdings schreibe, werde ich nun erstmal „Baustelle“ 2 abarbeiten, also den Bau der LED-Streifen. Denn wenn ich jetzt schon den Code programmieren würde, könnte ich diesen nicht testen,

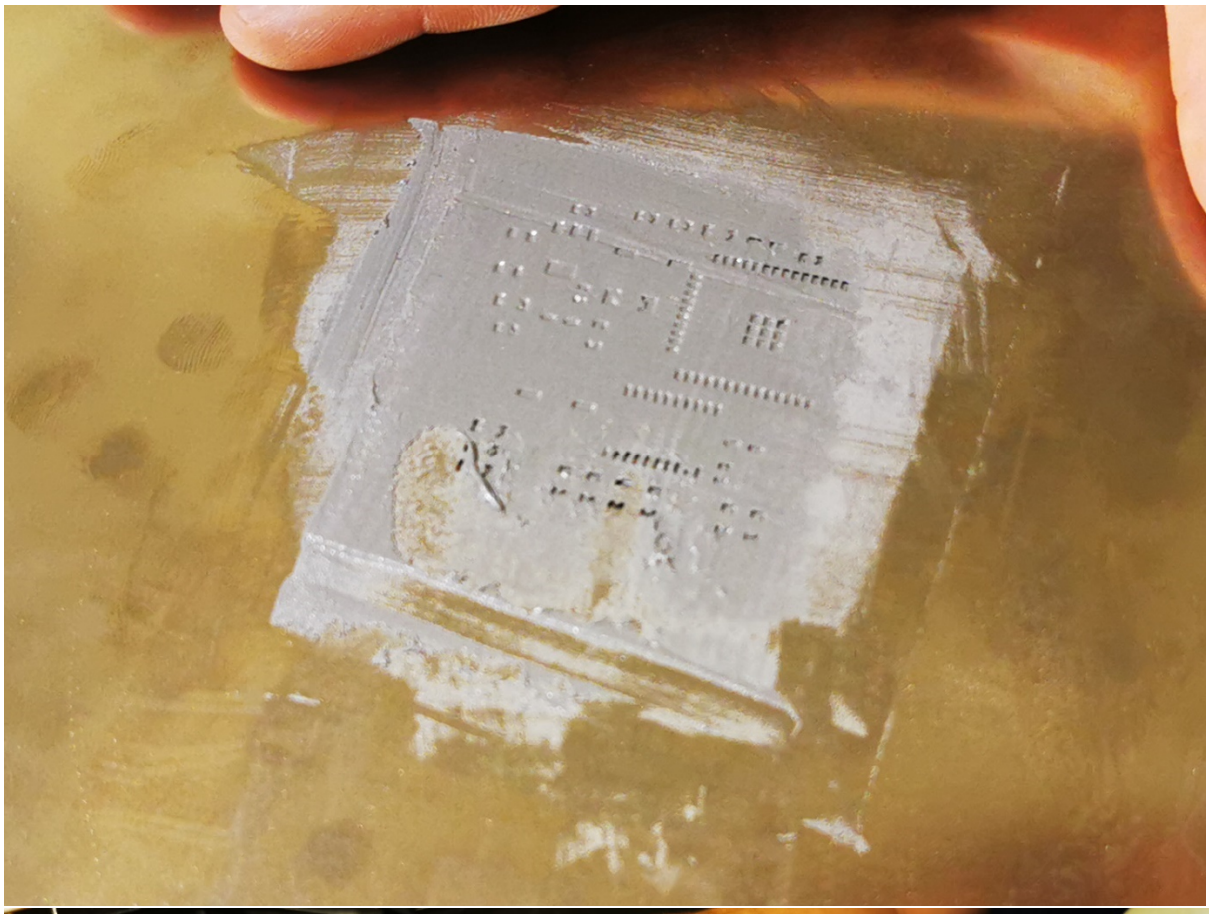
da der zu steuernde LED-Streifen noch gar nicht existiert (außer über einen PC als Empfänger, aber das ist unnötig umständlich). Auch hier habe ich erstmal lange recherchiert um zu wissen was ich benötige. Die Bauteile bestellte ich in China, da das Projekt rein mit deutschen Bauteilen zu teuer wäre um es selbst realisieren zu können. Neben dem LED-Streifen brauchte ich noch den Mini Computer, Modell ist der ESP-32, außerdem ein paar kleinere Bauteile, die ich bereits besaß (beim Bau werden diese alle einzeln gezeigt) und zuletzt noch eine Platine, auf die all das vereinfacht gesagt montiert werden muss. Da es solch eine Platine nicht direkt in Deutschland oder auch nicht in China genau wie ich sie brauche zu kaufen gibt, musste ich mir diese selbst entwickeln. Also erstellte ich die Platine in einem extra dafür existierenden Programm und ließ diese anschließend in China herstellen und mir zusenden. Wie dieses Programm aussieht und wie ich diese Platine erstellte, ist in den folgenden Bildern zu sehen. Die Informationen wie eine solche Platine erstellt wird habe ich mir durch Recherchen angeeignet, also was alles auf der Platine benötigt wird. Die Anfertigung habe ich allerdings alleine durchgeführt, weswegen diese Platine auch ein Unikat darstellt, der Zeitaufwand hierfür war sehr groß.



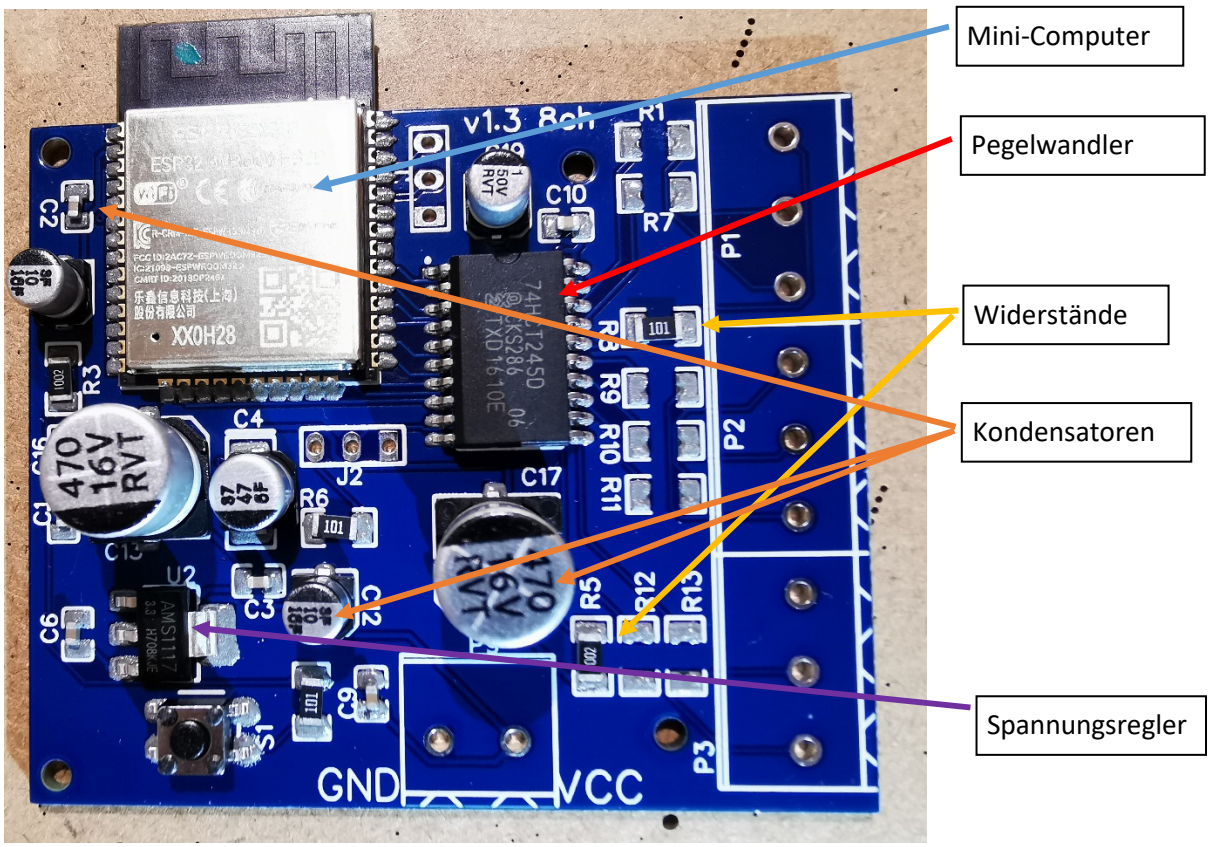


Nachdem ich also alle Teile zusammen hatte, begann der Bau der Hardware. Dazu nahm ich erst meine Platine als Grundlage und bestückte diese mit Lötzinn-Paste. Dazu benutzte ich eine Schablone, unter der sich die Platine befand. Anschließend strich ich mit Lötpaste darüber.





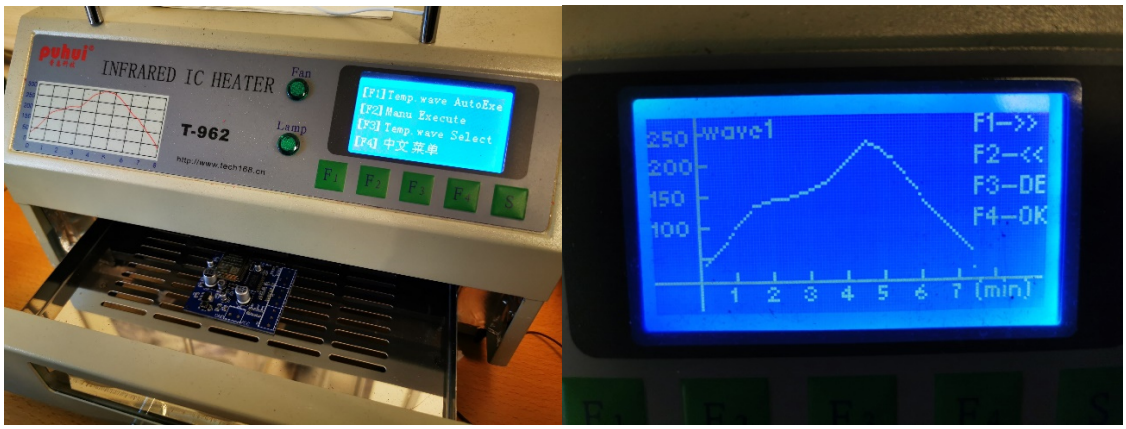
Anschließend bestückte ich die Leiterplatine wie folgt:



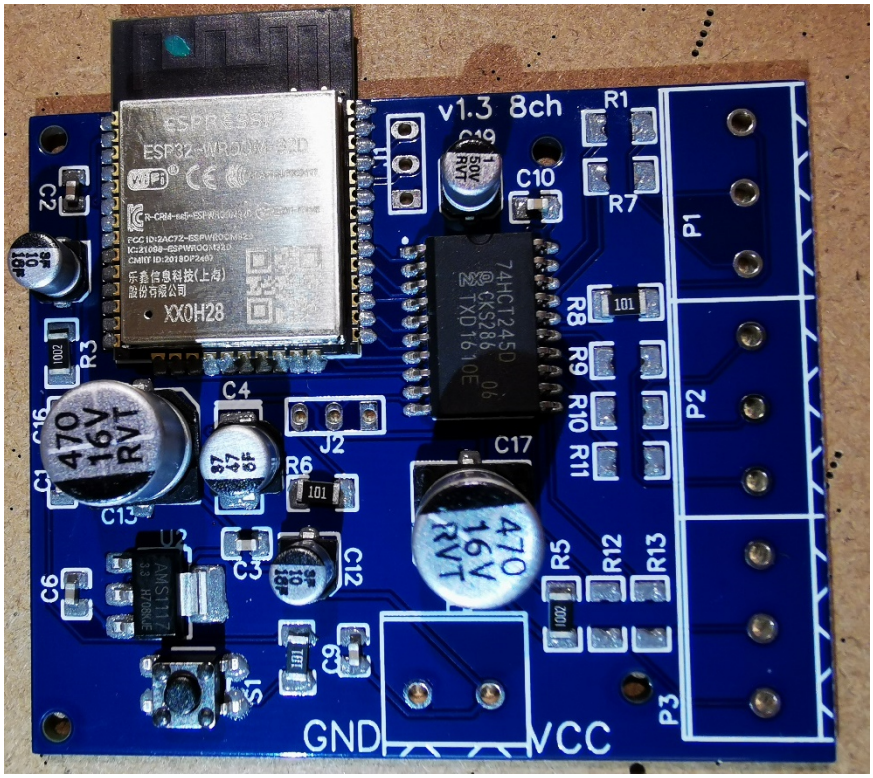
Erklärung der Bauteile:

- Der Mini-Computer besitzt Wlan-Funktion und ist für den Empfang und die Verarbeitung der Befehle zuständig.
- Der Spannungsregler regelt die Spannung von 5V auf 3,3V, da dies die Spannung ist, die der Mini-PC benötigt.
- Die Kondensatoren dienen dem Ausgleich von Spannungsschwankungen bzw. dem Glätten der Versorgungsspannung.
- Der Pegelwandler dient dazu die Spannung des Datensignals wieder auf 5V zu wandeln, da die LEDs 5VPegel benötigen, der Mikrocontroller bzw. Mini-Computer aber nur 3,3V ausgibt.

Danach wurde die bestückte Platine in einen Löt-Ofen gestellt, um die Teile mit der Platine zu verlöten.



Die fertige Platine sah dann so aus:



Kalenderwoche 26 (ca. 4 Stunden)

In Kalenderwoche 26 habe ich mich dann mit dem Programmieren der LEDs beschäftigt. Dazu benutzte ich das Programm Visual Studio von Microsoft. Da die App noch nicht fertig programmiert ist, habe ich den nachfolgenden Code mit einem Computerprogramm namens Packet Sender auf einem Windows PC getestet. Mit diesem Programm habe ich die einzelnen Befehle gesendet, die normalerweise die App senden würde (natürlich ist das nicht grafisch, ich musste den jeweiligen Zahlenbefehl wie in Kalenderwoche 21 beschrieben erst manuell im Kopf bilden und eintippen, das Programm sendete ihn dann per UDP Telegramm an den LED Streifen).

Als LED-Streifen habe ich den Typ WS2812 ausgewählt. Dieser hat den Vorteil, dass sich mehrere hundert LEDs, die sich an einem Streifen befinden, jeweils einzeln in Farbe und Helligkeit steuern

lassen. Jede der LEDs auf dem Streifen besitzt auf kleinstem Raum drei einzelne integrierte LEDs in den Farben Rot, Grün und Blau, so dass sich jede Farbe durch entsprechende Kombination der Farben darstellen lässt. Dies geschieht über nur eine einzige Datenleitung, über die die Farb- und Helligkeitswerte aller LEDs in sehr schneller Abfolge mehrmals pro Sekunde zu allen LEDs übertragen werden. Die Übersetzung der gewünschten Farbwerte der LEDs in die Steuerbefehle auf der Datenleitung übernimmt dabei die Software-Bibliothek „Fastled“, die ich in meinem Projekt verwende.

So ging ich beim Programmieren des LED-Streifens vor:

Zuerst legte ich die grundsätzlichen Werte fest: Anzahl der LED am Streifen, Wlan in das sich der Streifen zum Empfangen der Befehle einloggen soll, IP-Adresse unter der dieser erreichbar sein soll, Helligkeit am Anfang bei Start etc., all das ist im unteren Bild zu sehen. Die Starthelligkeit wählte ich deshalb auf nur 40 von 255 um im Falle einer zu schwachen Stromversorgung nicht sofort das Programm bzw. den Mini Computer abstürzen zu lassen. Sobald das Netzteil jedoch verbaut ist (aktuell wird der Streifen mit einer Powerbank zum komfortableren Testen und Programmieren mit Strom versorgt) wird die Starthelligkeit auf Maximum gestellt.

```
4   #include <Arduino.h>
5
6   #include <WiFi.h>
7   #include <ArduinoOTA.h>
8   #include <FastLED.h>
9   #include "AsyncUDP.h"
10  int localPort = 1234;
11  AsyncUDP udp; //erstellen eines UDP Objektes
12
13
14
15  #define LED_PIN     16
16  #define NUM_LEDS    240
17  #define BRIGHTNESS  40
18  #define LED_TYPE    WS2811
19  #define COLOR_ORDER GRB
20  #define UPDATES_PER_SECOND 50
21  #define ZOOMING_BEATS_PER_MINUTE 122
22
23  CRGBPalette16 currentPalette;
24  TBlendType    currentBlending;
25
26  extern CRGBPalette16 myRedWhiteBluePalette;
27  extern const TProgmemPalette16 myRedWhiteBluePalette_p PROGMEM;
28
29
30  CRGB leds[NUM_LEDS];
31
32
33  const char* ssid     = "SmartLED";
34  const char* password = "SmartLED";
35
36  byte UDPEmpfangen = 0;
37  byte OTA=0;
38  char buf[20];
39  byte brightness;
40  byte rot;
41  byte gruen;
42  byte blau;
43  byte muster=0;
44  byte ranr;
45  byte rang;
46  byte ranb;
47  byte ran2;
48
49
50
51  IPAddress local_IP(192, 168, 6, 201);
52  IPAddress gateway(192, 168, 6, 1);
53  IPAddress subnet(255, 255, 255, 0);
```

Anschließend kommt die Setup() Funktion, die nach Anlegen der Versorgungspannung nur einmal zu Beginn aufgerufen wird. Am Anfang der Setup Funktion wird der LED-Streifen auf Rot gestellt, heißt

der komplette LED-Streifen leuchtet rot um zu signalisieren, dass er zwar Strom erhält, aber noch nicht bereit ist, da er z.B. noch nicht im Wlan verbunden ist. Am Ende des Setups (nach ein paar Sekunden) ändert der LED-Streifen seine Farbe auf Grün, ab diesem Zeitpunkt ist der Streifen bereit und kann angesteuert werden. Dies soll dem späteren Benutzer genau signalisieren ab wann der Streifen mit der App gesteuert werden kann. Der Code dazu in folgenden Bildern:

```
306  void setup()
307  {
308
309    FastLED.addLeds<LED_TYPE, LED_PIN, COLOR_ORDER>(leds, NUM_LEDS).setCorrection( TypicalLEDStrip );
310    FastLED.setBrightness(  BRIGHTNESS );
311    fill_solid( leds, NUM_LEDS, CRGB(255, 0,0));
312    FastLED.show();
313
314    Serial.begin(115200);
315
316    if (!WiFi.config(local_IP, gateway, subnet)) {
317      Serial.println("STA Failed to configure");
318    }
319
320    Serial.print("Connecting to ");
321    Serial.println(ssid);
322
323    WiFi.begin(ssid, password);
324
325    while (WiFi.status() != WL_CONNECTED) {
326      delay(500);
327      Serial.print(".");
328    }
329
330    Serial.println("");
331    Serial.println("WiFi connected!");
332    Serial.print("IP address: ");
333    Serial.println(WiFi.localIP());
334    Serial.print("ESP Mac Address: ");
335    Serial.println(WiFi.macAddress());
336    Serial.print("Subnet Mask: ");
337    Serial.println(WiFi.subnetMask());
338    Serial.print("Gateway IP: ");
339    Serial.println(WiFi.gatewayIP());
340    Serial.print("DNS: ");
341    Serial.println(WiFi.dnsIP());
342
343
344    ArduinoOTA
345      .onStart([]() {
346        String type;
347        OTA=1;
348        if (ArduinoOTA.getCommand() == U_FLASH)
349          type = "sketch";
350        else
351          type = "filesystem";
352
353
354        Serial.println("Start updating " + type);
355      })
356      .onEnd([]() {
357        Serial.println("\nEnd");
358      })
359      .onProgress([](unsigned int progress, unsigned int total) {
360        Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
361      })
362      .onError([](ota_error_t error) {
363        Serial.printf("Error[%u]: ", error);
364        if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
365        else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
366        else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
367        else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
368        else if (error == OTA_END_ERROR) Serial.println("End Failed");
369      });
```

```
344   ArduinoOTA
345     .onStart([]() {
346       String type;
347       OTA=1;
348       if (ArduinoOTA.getCommand() == U_FLASH)
349         type = "sketch";
350       else
351         type = "filesystem";
352
353
354       Serial.println("Start updating " + type);
355     })
356     .onEnd([]() {
357       Serial.println("\nEnd");
358     })
359     .onProgress([](unsigned int progress, unsigned int total) {
360       Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
361     })
362     .onError([](ota_error_t error) {
363       Serial.printf("Error[%u]: ", error);
364       if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
365       else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
366       else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
367       else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
368       else if (error == OTA_END_ERROR) Serial.println("End Failed");
369     });
370
371   ArduinoOTA.begin();
372
373   long safetyOTAtimer=millis();
374   while ((millis()-safetyOTAtimer<30000) || OTA)
375       ArduinoOTA.handle();
376
377
378   currentPalette = RainbowColors_p;
379   currentBlending = LINEARBLEND;
380
381    if (udp.listen(localPort)) {
382         delay(100);
383         udp.onPacket([](AsyncUDPPacket packet) {
384           memcpy(buf, packet.data(), packet.length());
385           UDPEmpfangen=1;
386         });
387     }
388
389   fill_solid( leds, NUM_LEDS, CRGB(0, 255,0));
390   FastLED.show();
391 }
```

Nachdem also das Setup durchlaufen ist und der Streifen grün leuchtet, beginnt der Streifen in die sogenannte loop() Funktion zu springen. Diese wird nun immer wieder durchlaufen solange der Mini Computer mit Strom versorgt ist.

Hier habe ich also programmiert, dass wenn der erste Wert im Datentelegramm, also der Befehlscode (UDP-Buffer an der Stelle 0) eins ist, wird die Helligkeit auf den Wert gesetzt, der im darauffolgenden Zahlenwert übermittelt wurde.

Wenn der erste Wert (Befehlscode) die Zahl 2 ist, wird rot auf den zweiten Zahlenwert gesetzt, grün auf den dritten und Blau auf den vierten Wert.

Und wenn der erste Wert (Befehlscode) eine drei ist, dann steht der nachfolgende Wert für das anzuwendende Muster.

Übersicht der Codierung der einzelnen Befehle, die per UDP übertragen werden:

1.Byte /Befehlscode	2.Byte	3.Byte	4.Byte	Bedeutung
001	Gesamthelligkeit 000…255	—-	—-	Helligkeit Verändern
002	Helligkeit Farbe rot 000…255	Helligkeit Farbe grün 000…255	Helligkeit Farbe blau 000…255	Farbe Verändern
003	Muster / Animation 000…255	—-	—-	Animation starten

```
393  void loop()
394  {
395
396      if (UDPEmpfangen){
397          UDPEmpfangen=0;
398
399        if (buf[0]==1){
400          brightness=buf[1];
401          FastLED.setBrightness(  brightness );
402          FastLED.show();
403        }
404
405        if (buf[0]==2){
406          rot=buf[1];
407          gruen=buf[2];
408          blau=buf[3];
409          fill_solid( leds, NUM_LEDS, CRGB(rot, gruen,blau));
410          FastLED.show();
411          muster=0;
412        }
413
414        if (buf[0]==3){
415          muster=buf[1];
416        }
417
418      }
419
420      switch(muster){
421        case 1:
422          rainbow_beat();
423          FastLED.show();
424
425        break;
426
427        case 2:
428          currentPalette = CloudColors_p;
429          currentBlending = LINEARBLEND;
430          static int startIndex_c7 = 0;
431          startIndex_c7 = startIndex_c7 + 1;
432          FillLEDsFromPaletteColors( startIndex_c7);
433          FastLED.show();
434          FastLED.delay(1000/ UPDATES_PER_SECOND);
435        break;
436
437        case 3:
438         SetupPurpleAndGreenPalette();
439         currentBlending = LINEARBLEND;
440         static int startIndex_c10 = 0;
441         startIndex_c10 = startIndex_c10 + 1;
442         FillLEDsFromPaletteColors( startIndex_c10);
443         FastLED.show();
444         FastLED.delay(1000/ UPDATES_PER_SECOND);
445        break;
```

```
447         case 4:
448           ranr = rand() %255;
449           rang = rand() %255;
450           ranb = rand() &255;
451           ran2 = rand() %600 + 100;
452           fill_solid( leds, NUM_LEDS, CRGB(ranr, rang, ranb));
453
454           FastLED.setBrightness(0);
455           if ((brightness>0)) {
456             FastLED.setBrightness(60);
457           }
458           FastLED.show();
459           delay(40+ran2);
460
461           fill_solid( leds, NUM_LEDS, CRGB(255, 255, 255));
462
463           FastLED.setBrightness(0);
464           if (brightness>0) {
465             FastLED.setBrightness(255);
466           }
467           FastLED.show();
468           delay(10);
469         break;
470
471         case 5:
472           discostrobe();
473           FastLED.setBrightness(0);
474           if (brightness>0) {
475                   FastLED.setBrightness(255);
476           }
477           FastLED.show();
478           delayToSyncFrameRate( UPDATES_PER_SECOND);
479           FastLED.setBrightness(0);
480           if (brightness>0) {
481             FastLED.setBrightness(255);
482           }
483           break;
484
485         case 6:
486           Serial.println(LEDS.getFPS());
487           sawtooth();
488           FastLED.show();
489          break;
490
491       }
492
493
494
495   ArduinoOTA.handle();
496
497   }
```

Kalenderwoche 29 (ca. 3 Stunden)

Nachdem der erste LED-Streifen also gebaut und programmiert wurde, baute ich nach selbigem Prinzip wie in KW 25 einen zweiten, da am Ende mein Projekt zwei LED-Streifen a vier Meter lang umfassen sollte. Bauaufwand war ungefähr derselbe, das Programm allerdings konnte ich 1 zu 1 übernehmen, nur die IP-Adresse musste natürlich eine andere sein. Das Designen der Platine muss natürlich auch nicht nochmal gemacht werden, daher die zeitliche Differenz zur KW 25. Die LED-Streifen, also Baustelle zwei wie ich es am Anfang nannte, ist damit zum Teil abgeschlossen. Nun kommen noch die Muster und deren Methoden. Das Muster 1-5 ist Open-Source Code aus dem Internet (mit den aufgerufenen Methoden), keine Nennung notwendig. Muster 5 ist selbst programmiert. Der komplette restliche Code ist selbst programmiert, nur alle Muster nochmal selbst zu programmieren, wäre eine Seminararbeit für sich. Damit stellen die Muster auch das einzige Fremdmaterial im kompletten Projekt dar.

```
104  const TProgmemPalette16 myRedWhiteBluePalette_p PROGMEM =
105  {
106      CRGB::Red,
107      CRGB::Gray,
108      CRGB::Blue,
109      CRGB::Black,
110
111      CRGB::Red,
112      CRGB::Gray,
113      CRGB::Blue,
114      CRGB::Black,
115
116      CRGB::Red,
117      CRGB::Red,
118      CRGB::Gray,
119      CRGB::Gray,
120      CRGB::Blue,
121      CRGB::Blue,
122      CRGB::Black,
123      CRGB::Black
124  };
125
126
127
128  void ChangePalettePeriodically()
129  {
130      uint8_t secondHand = (millis() / 1000) % 60;
131      static uint8_t lastSecond = 99;
132
133      if( lastSecond != secondHand) {
134          lastSecond = secondHand;
135          if( secondHand ==  0)  { currentPalette = RainbowColors_p;         currentBlending = LINEARBLEND; }
136          if( secondHand == 10)  { currentPalette = RainbowStripeColors_p;   currentBlending = NOBLEND;  }
137          if( secondHand == 15)  { currentPalette = RainbowStripeColors_p;   currentBlending = LINEARBLEND; }
138          if( secondHand == 20)  { SetupPurpleAndGreenPalette();             currentBlending = LINEARBLEND; }
139          if( secondHand == 25)  { SetupTotallyRandomPalette();              currentBlending = LINEARBLEND; }
140          if( secondHand == 30)  { SetupBlackAndWhiteStripedPalette();       currentBlending = NOBLEND; }
141          if( secondHand == 35)  { SetupBlackAndWhiteStripedPalette();       currentBlending = LINEARBLEND; }
142          if( secondHand == 40)  { currentPalette = CloudColors_p;           currentBlending = LINEARBLEND; }
143          if( secondHand == 45)  { currentPalette = PartyColors_p;           currentBlending = LINEARBLEND; }
144          if( secondHand == 50)  { currentPalette = myRedWhiteBluePalette_p; currentBlending = NOBLEND;  }
145          if( secondHand == 55)  { currentPalette = myRedWhiteBluePalette_p; currentBlending = LINEARBLEND; }
146      }
147  }
148
149  void rainbow_beat() {
150
151    uint8_t beatA = beatsin8(17, 0, 255);                        // Start
152    uint8_t beatB = beatsin8(13, 0, 255);
153    fill_rainbow(leds, NUM_LEDS, (beatA+beatB)/2, 8);            // Starte Regenbogen
154
155  } // rainbow()
156
```

```
57  void FillLEDsFromPaletteColors( uint8_t colorIndex)
58  {
59      uint8_t brightness = 255;
60
61      for( int i = 0; i < NUM_LEDS; i++) {
62          leds[i] = ColorFromPalette( currentPalette, colorIndex, brightness, currentBlending);
63          colorIndex += 3;
64      }
65  }
66
67
68
69  void SetupTotallyRandomPalette()
70  {
71      for( int i = 0; i < 16; i++) {
72          currentPalette[i] = CHSV( random8(), 255, random8());
73      }
74  }
75
76
77  void SetupBlackAndWhiteStripedPalette()
78  {
79
80      fill_solid( currentPalette, 16, CRGB::Black);
81      currentPalette[0] = CRGB::White;
82      currentPalette[4] = CRGB::White;
83      currentPalette[8] = CRGB::White;
84      currentPalette[12] = CRGB::White;
85
86  }
87
88
89  void SetupPurpleAndGreenPalette()
90  {
91      CRGB purple = CHSV( HUE_PURPLE, 255, 255);
92      CRGB green  = CHSV( HUE_GREEN, 255, 255);
93      CRGB black  = CRGB::Black;
94
95      currentPalette = CRGBPalette16(
96                                      green,  green,  black,  black,
97                                      purple, purple, black,  black,
98                                      green,  green,  black,  black,
99                                      purple, purple, black,  black );
100 }
101
102
157  static void drawRainbowDashes(
158    uint8_t startpos, uint16_t lastpos, uint8_t period, uint8_t width,
159    uint8_t huestart, uint8_t huedelta, uint8_t saturation, uint8_t value)
160  {
161    uint8_t hue = huestart;
162    for( uint16_t i = startpos; i <= lastpos; i += period) {
163      CRGB color = CHSV( hue, saturation, value);
164
165      uint16_t pos = i;
166      for( uint8_t w = 0; w < width; w++) {
167        leds[ pos ] = color;
168        pos++;
169        if( pos >= NUM_LEDS) {
170          break;
171        }
172      }
173
174      hue += huedelta;
175    }
176  }
177
178
179
180  CRGB wheel(int WheelPos, int dim) {
181    CRGB color;
182    if (85 > WheelPos) {
183     color.r=0;
184     color.g=WheelPos * 3/dim;
185     color.b=(255 - WheelPos * 3)/dim;;
186    }
187    else if (170 > WheelPos) {
188     color.r=WheelPos * 3/dim;
189     color.g=(255 - WheelPos * 3)/dim;
190     color.b=0;
191    }
192    else {
193     color.r=(255 - WheelPos * 3)/dim;
194     color.g=0;
195     color.b=WheelPos * 3/dim;
196    }
197    return color;
198  }
199
```

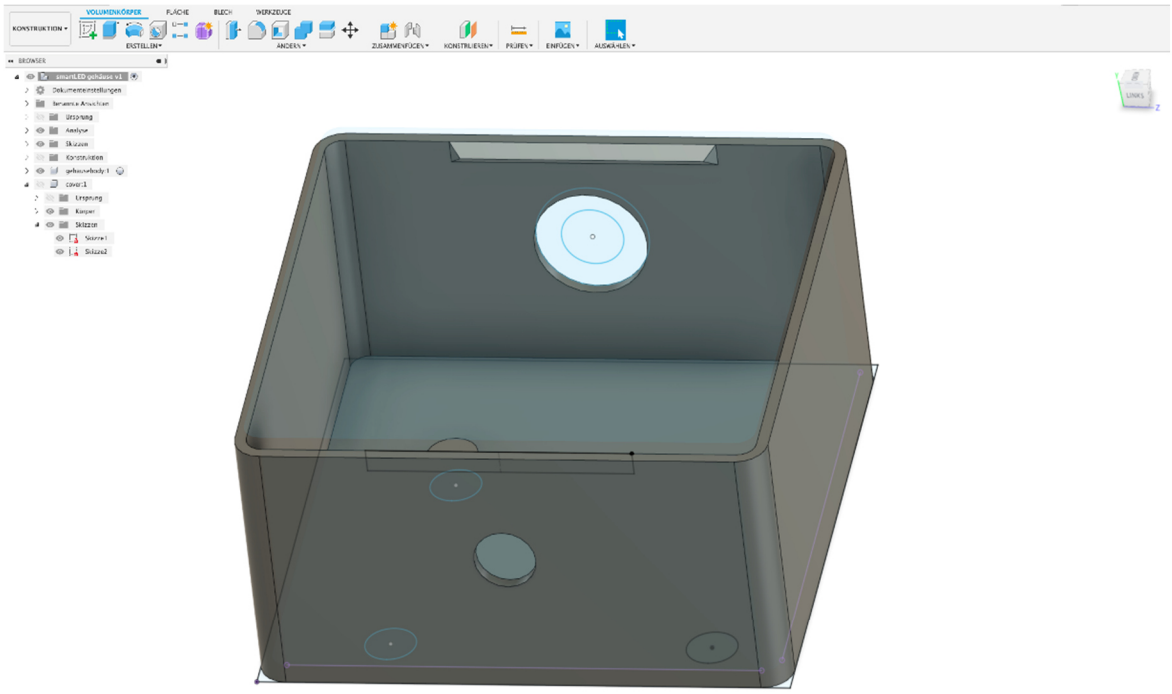
```
200   void discoWorker(
201       uint8_t dashperiod, uint8_t dashwidth, int8_t  dashmotionspeed,
202       uint8_t stroberepeats,
203       uint8_t huedelta)
204   {
205     static uint8_t sRepeatCounter = 0;
206     static int8_t sStartPosition = 0;
207     static uint8_t sStartHue = 0;
208
209     sStartHue += 1;
210
211     sRepeatCounter = sRepeatCounter + 1;
212     if( sRepeatCounter>= stroberepeats) {
213       sRepeatCounter = 0;
214
215       sStartPosition = sStartPosition + dashmotionspeed;
216
217       if( sStartPosition >= dashperiod ) {
218         while( sStartPosition >= dashperiod) { sStartPosition -= dashperiod; }
219         sStartHue  -= huedelta;
220       } else if( sStartPosition < 0) {
221         while( sStartPosition < 0) { sStartPosition += dashperiod; }
222         sStartHue  += huedelta;
223       }
224     }
225
226     const uint8_t kSaturation = 208;
227     const uint8_t kValue = 255;
228
229     drawRainbowDashes( sStartPosition, NUM_LEDS-1,
230                        dashperiod, dashwidth,
231                        sStartHue, huedelta,
232                        kSaturation, kValue);
233   }
234
235   void discostrobe()
236   {
237
238     fill_solid( leds, NUM_LEDS, CRGB::Black);
239
240
241     const uint8_t kStrobeCycleLength = 3;
242     static uint8_t sStrobePhase = 0;
243     sStrobePhase = sStrobePhase + 1;
244     if( sStrobePhase >= kStrobeCycleLength ) {
245       sStrobePhase = 0;
246     }
247
248     if( sStrobePhase == 0 ) {
249
250
251       uint8_t dashperiod= beatsin8( 8, 4,10);
252
253       uint8_t dashwidth = (dashperiod / 4) + 1;
254
255       uint8_t zoomBPM = ZOOMING_BEATS_PER_MINUTE;
256       int8_t  dashmotionspeed = beatsin8( (zoomBPM /2), 1,dashperiod);
257
258       if( dashmotionspeed >= (dashperiod/2)) {
259         dashmotionspeed = 0 - (dashperiod - dashmotionspeed );
260       }
261
262
263       uint8_t cycle = beat8(2); // two cycles per minute
264       uint8_t easedcycle = ease8InOutCubic( ease8InOutCubic( cycle));
265       uint8_t wavecycle = cubicwave8( easedcycle);
266       uint8_t hueShift = scale8( wavecycle,130);
267
268
269       uint8_t strobesPerPosition = 1; // try 1..4
270
271
272       discoWorker( dashperiod, dashwidth, dashmotionspeed, strobesPerPosition, hueShift);
273     }
274   }
275
276   static void delayToSyncFrameRate( uint8_t framesPerSecond)
277   {
278     static uint32_t msprev = 0;
279     uint32_t mscur = millis();
280     uint16_t msdelta = mscur - msprev;
281     uint16_t mstargetdelta = 1000 / framesPerSecond;
282     if( msdelta < mstargetdelta) {
283       delay( mstargetdelta - msdelta);
284     }
285     msprev = mscur;
286
287   }
```

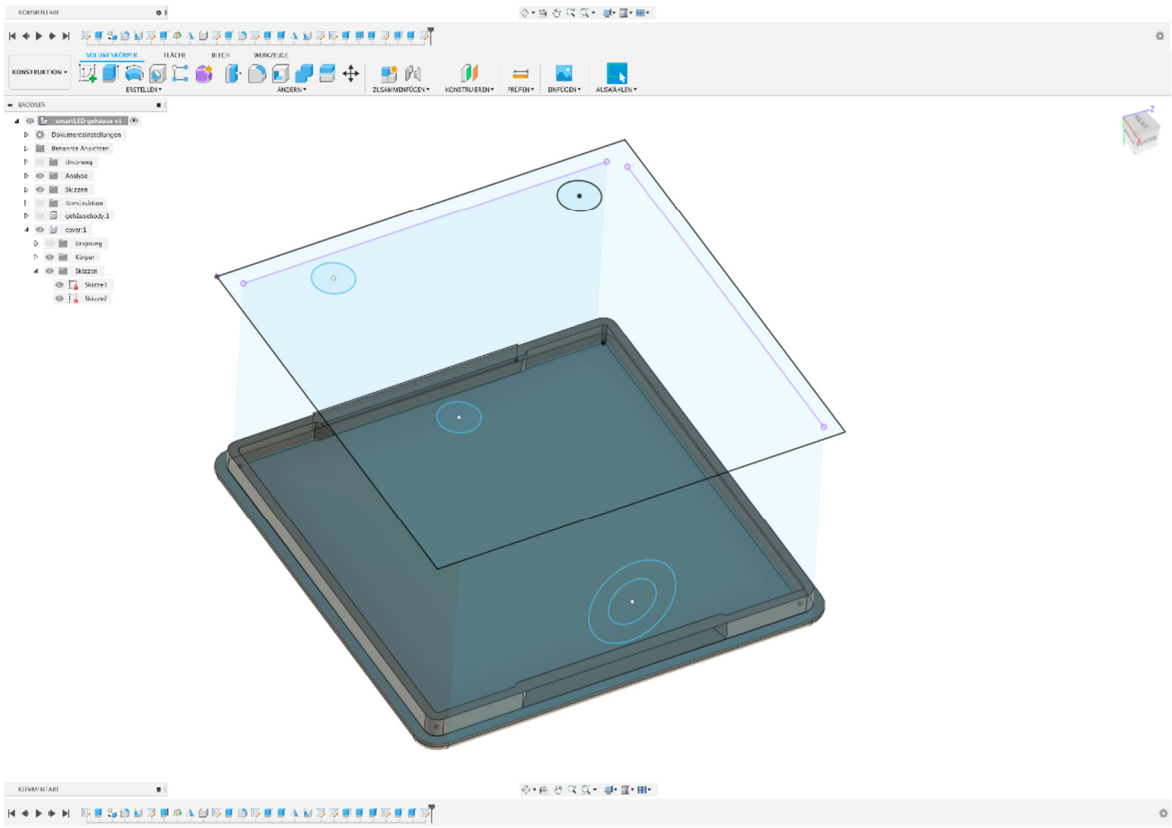
```
289  void sawtooth() {
290
291    int bpm = 20;
292    int ms_per_beat = 60000/bpm;
293    int ms_per_led = 60000/bpm/NUM_LEDS;
294
295    int cur_led = ((millis() % ms_per_beat) / ms_per_led)%(NUM_LEDS);
296
297    if (cur_led == 0)
298     fill_solid(leds, NUM_LEDS, CRGB::Black);
299    else
300      leds[cur_led] = CRGB(0, 0, 255);
301
302  }
```

Kalenderwoche 30 (ca. 12 Stunden)

Nun habe ich den Bau der LED-Streifen vollendet. Das Gehäuse habe ich mit Fusion360 designt, einem 3D Grafik Programm, um dies dann mit dem 3D-Drucker Ultimaker 2 zu drucken. Besonders schwierig war hierbei die Maße alle korrekt abzumessen, zum einen für das Gehäuse selbst, zum anderen für die drei Säulen im Gehäuse. Die drei Säulen, ca. 5mm hoch, baute ich ein um auf diesen die Platine festzuschrauben. Bevor ich in das Gehäuse etwas einbaute, schliff ich es zweimal mit zuerst gröberem Schleifpapier und anschließend mit feinerem, um eine glatte, gleichmäßige und angenehme Oberfläche zu gewährleisten. Zuletzt erfolgte dann noch die Einbauarbeit mit den richtigen Zwischensteckern, die ich ebenfalls im Internet bestellte. Diese werden in die Wand des Gehäuses eingebaut und führen das Kabel aus bzw. in das Gehäuseinnere, sodass diese nicht einfach lose aus einem Loch im Gehäuse hängen, sondern durch massive Stecker am Gehäuse modular angeschlossen werden können. Alle außerhalb des Gehäuses, sowie teils innenliegende Lötstellen und Anschlüsse wurden mit einem sogenannten Schrumpfschlauch überzogen, um Kurzschlüsse, herausstehende Drähte und Beschädigungen zu vermeiden.

All das ist auf folgenden Bildern Dokumentiert:





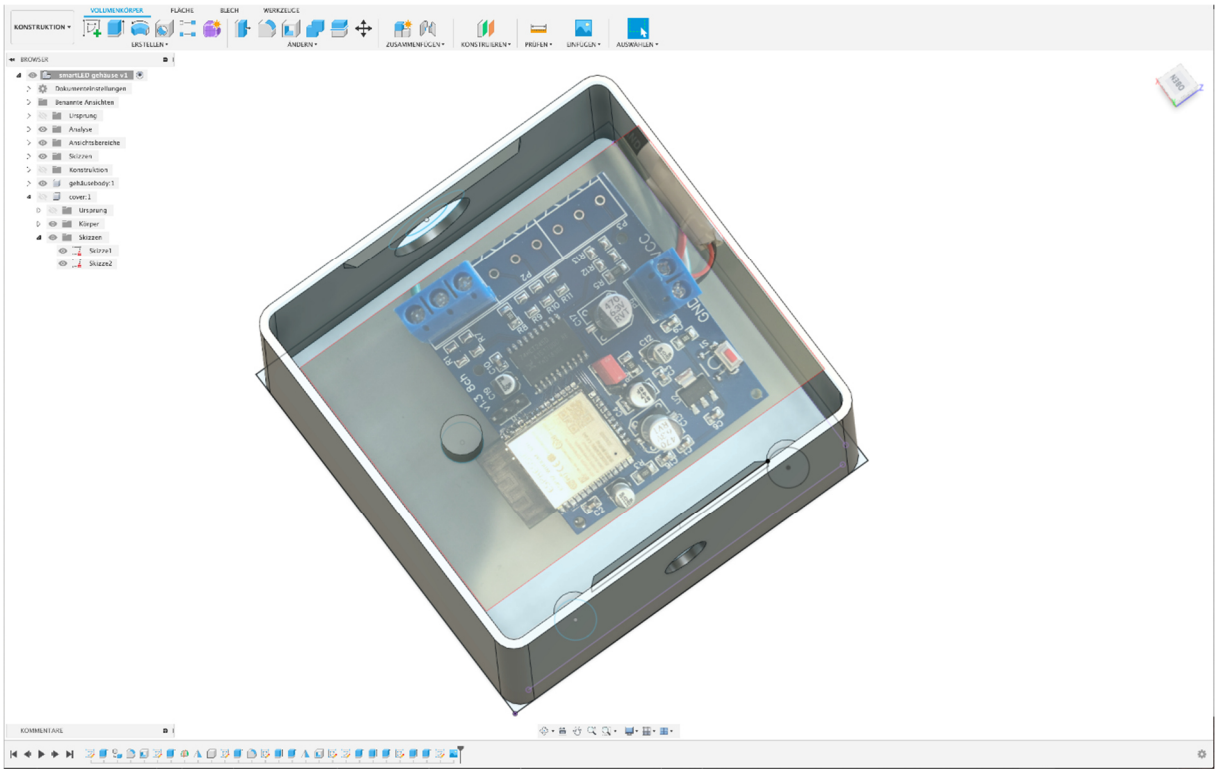
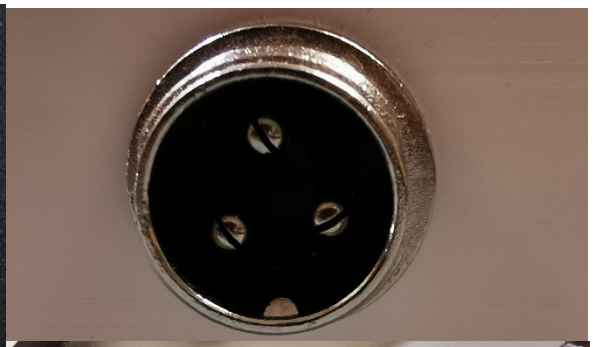
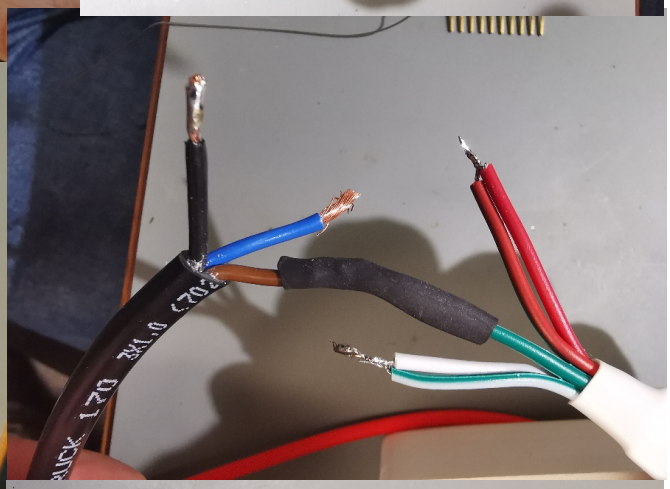
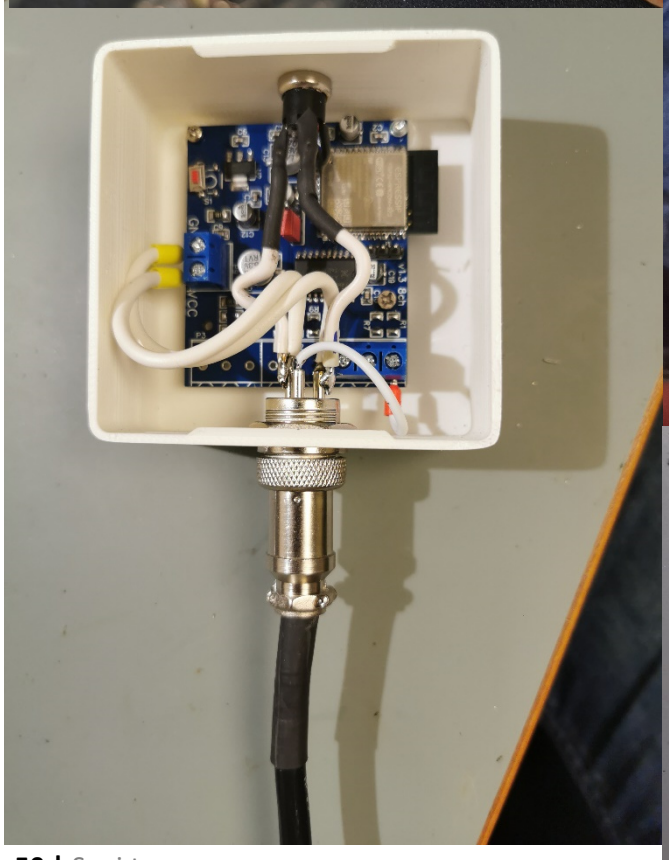


Bild der Platine wurde importiert um Maße und Abstände nachzuprüfen.

Anschließend überlegte ich mir noch eine Art der Verpackung. Dazu wählte ich einen kleinen Koffer wie auf folgenden Bildern zu sehen. Im Koffer selbst schnitt ich mir das Innere so zurecht, dass die einzelnen Teile gut darin verstaut werden können, dokumentiert auf folgenden Bildern. Die Rollen sind übrigens die LED-Streifen, die weißen Boxen die Elektronik mit Wlan und Mini Computer und auf den LED-Streifen selbst folgen schließlich noch 2 Netzteile (zum Zeitpunkt des Fotos war nur eines zur Hand). Eine Anleitung wird zum Verständnis der Funktionsweise auch noch beigelegt werden.










Zuletzt lackierte ich die Gehäuse schließlich noch mit einer weißen Farbe. Der Deckel kann übrigens abgenommen werden. Falls in das Gehäuse gesehen werden will, kann der Deckel mit etwas Kraft (am besten Fingernagel an einer Gehäuseecke zwischen Deckel und Gehäuse schieben) entfernt und später wieder verschlossen werden.

Kalenderwoche 30 (ca. 9 Stunden)

In Kalenderwoche 30 habe ich mich dann mit dem endgültigen Programmieren der App beschäftigt. Also zu hinterlegen was genau bei welchem Button passieren soll, vereinfacht ausgedrückt. Schwerpunkte bei der Programmierung waren die Buttons, sodass bei Betätigung immer der richtige Befehl an den LED-Streifen gesendet wird, sowie der Helligkeitsregler, der On/ Off switch sowie der Color Picker, welcher implementiert wurde. Auf die genauen einzelnen programmiertechnischen Schritte sowie Einzelheiten einzugehen ist denke ich unnötig, das Programmierte lässt sich schließlich anhand des Programmtextes einsehen. Die genaue technische Umsetzung und Funktionsweise wurde schon mehrmals erläutert, es ist nur die Anwendung dieser.

Anschließend der Programmtext, den ich für die App geschrieben habe.

Der Komplette Code aus Android Studio ist am Ende des Dokuments zu finden.

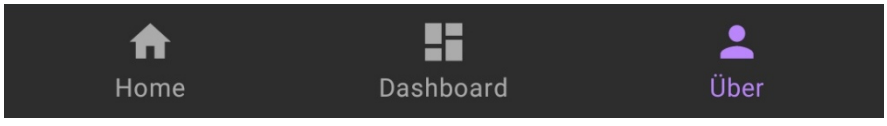
Kalenderwoche 31 (ca. 12 Stunden)

In Kalenderwoche 31 habe ich mich dann mit dem Design der App beschäftigt. Da die 2. Baustelle, also die LED-Streifen fertig sind, sowie die Programmierung der App, ist das Design das Letzte, was mir fehlt in meinem Projekt, ausgenommen die Schreibarbeit (3. Baustelle).

Als erstes habe ich mich auf eine Menü Führung/ Struktur festgelegt. Bei dem Design gibt es nun drei verschiedene „Seiten“ bzw. Menüpunkte, die sich unten in einer Navigationsleiste wechseln lassen. Somit ist die Menüführung wie auch das Design und der grafische Aufbau anders als noch in Kalenderwoche 17 geplant, die Skizze aus Kalenderwoche 17 ist damit nicht mehr gültig. Bei meiner neu ausgedachten Menüführung gibt es insgesamt drei Menüpunkte, die sich mit einem Klick auf das jeweilige Symbol wechseln lassen. Der erste Menüpunkt ist die Geräteauswahl, auf dieser „Seite“ wird nur ausgewählt, welches Gerät gesteuert werden soll. Sobald das zu steuernde Gerät ausgewählt ist, muss man auf den zweiten Menüpunkt in der Navigationsleiste tippen und gelangt somit in die Steuerung des LED-Streifens, den man zuvor ausgewählt hat. Mit einem Tippen auf den ersten Menüpunkt gelangt man wieder in die Geräteauswahl und kann eine andere LED-Leiste (-Streifen) auswählen, und dann wieder mit Klick auf den zweiten Menüpunkt steuern. Der dritte Menüpunkt dient nur einer FAQ oder „Über mich“ Seite.

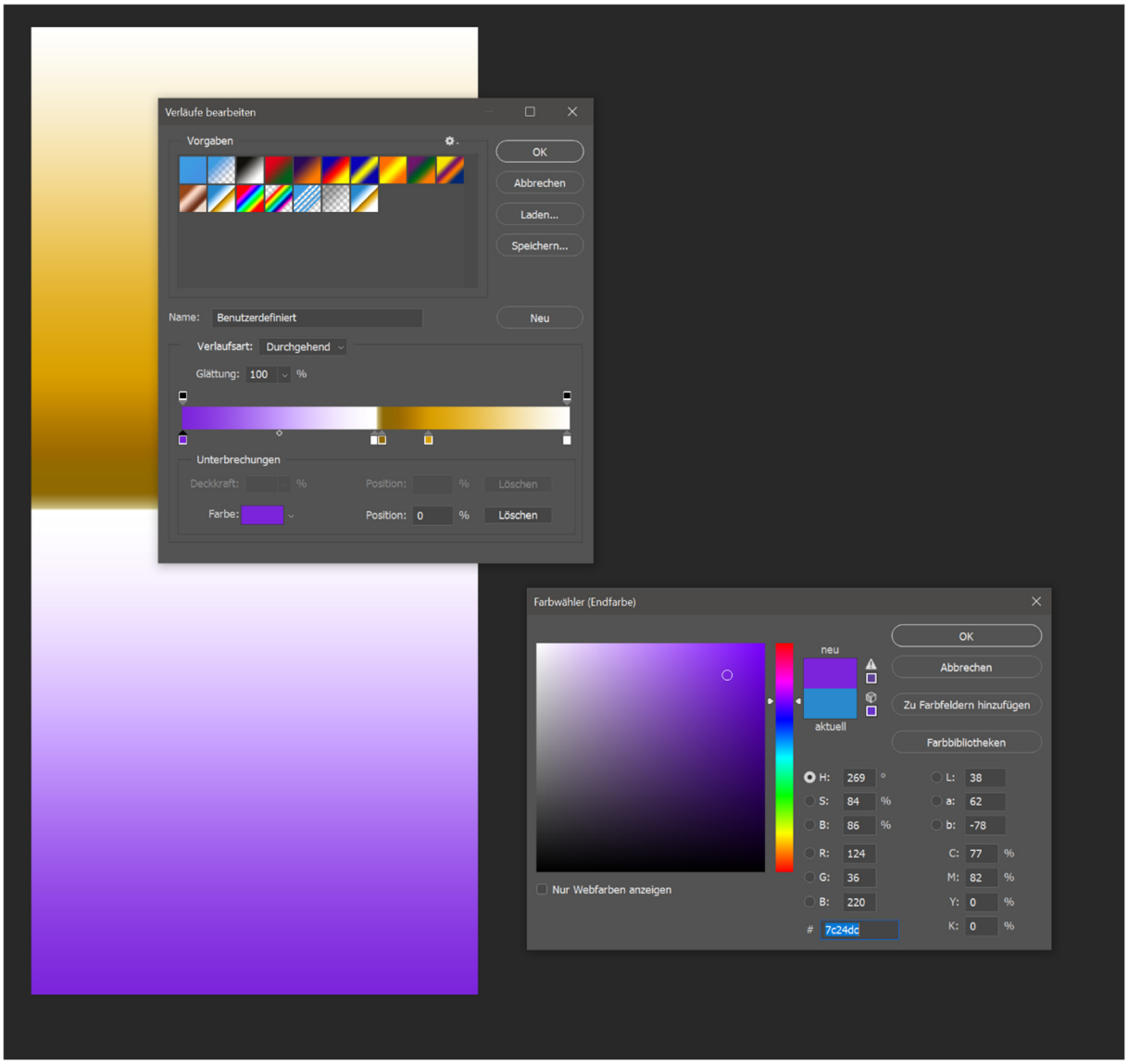
Zusammengefasst: Es gibt drei in einer unteren Menüleiste anklickbare Menüpunkte, bei Ersterem (der auch bei Start der App angezeigt wird) wird das zu steuernde Gerät (LED-Streifen) ausgewählt, im zweiten Menüpunkt wird das zuvor ausgewählte Gerät schließlich gesteuert. Zwischen den Menüpunkten lässt sich jederzeit hin und her wechseln. Die Funktionalität bleibt so wie in Kalenderwoche 17 bereits erklärt, nur die Auswahl der zu steuernden Geräte ist auf eine extra Seite ausgelagert worden. Die App hat somit drei Seiten, statt wie vorher geplant eine. Das schafft mehr Überblick. Es muss nicht so viel auf eine Seite gequetscht werden, das lässt unter anderem mehr Platz für Erklärungen in Textform, führt aber auch zu einer besseren Kompatibilität mit kleineren Handys, da weniger auf einer Seite angezeigt werden muss, denn wie bereits erwähnt, wird der zu steuernde LED-Streifen auf einer extra „Seite“ ausgewählt und nicht auf der gleichen „Seite“/ Menüpunkt, wo er gesteuert wird.

Bevor ich mit dem Design der „Seiten“ begonnen habe, habe ich noch im Navigationsmenü unten die Symbole und Texte angepasst. Die erste Seite der App habe ich „Home“ genannt und in der Navigationsleiste ein Haus als Symbol hinterlegt. Die zweite Seite habe ich „Dashboard“ genannt und ein passendes schematisches Symbol hinzugefügt. Außerdem habe ich noch die letzte Seite mit „Über“ beschriftet und eine Person als Symbol hinterlegt. Zu sehen auf folgendem Screenshot („Über“ ist Rosa gefärbt, da zum Zeitpunkt des Screenshots diese Seite geöffnet bzw. angewählt war.).

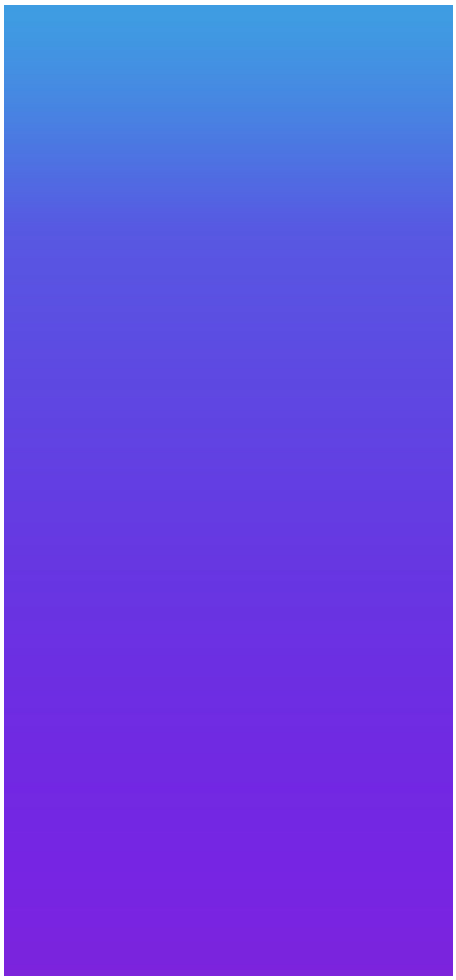


Nachdem das Menü fertig war, habe ich mit dem Design der ersten Seite begonnen. Für das dunkle Menü und die dunkle Leiste am oberen Bildschirmrand (standartmäßig wäre diese blau und das Menü weiß & blau) habe ich mich entschieden, da ich das zum einen angenehmer für die Augen finde und zum anderen da es zum blauen Hintergrundverlaufsbild, das ich als Designgrundlage selbst gestaltet habe, einen schönen Kontrast bildet und dadurch kein zu kitschiges bzw. farbenblendendes Design entsteht. Das Blau der oberen Leiste hätte sich viel zu sehr „gebissen“ mit dem der blauen Farbe des Hintergrundbildes. Nach häufigem testen kam ich zu dem Schluss, dass das dunkle Design, das sich in der oberen Leiste unter dem Bildschirmrand und in der Navigationsleiste äußert, die bessere Wahl ist.

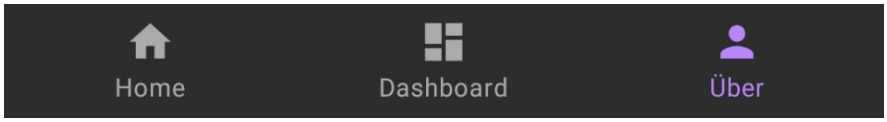
Noch dazu kommt, dass es dem aktuellen Trend entspricht, Apps in einem sogenannten „Night Mode“ auszuliefern. Ich gehe damit einen Zwischenweg. Die App soll zum einen zwar freundlich und farbenfroh sein, jedoch nicht abschreckend wirken durch übertriebene, sich beißende „Farben“. Die untere und obere Leiste ist zwar eher dunkel, der App Inhalt dazwischen ist mit dem blauen Hintergrundverlaufsbild aber farbenfroh. Damit wollte ich die genau passende Mitte zwischen überzogenen, kitschigen Farben und langweiliger Farbgebung finden. Bevor ich das Seitendesign mache, hier ein Bild zu Hintergrundbild, dass ich in Photoshop designte.



Das Ergebnis:



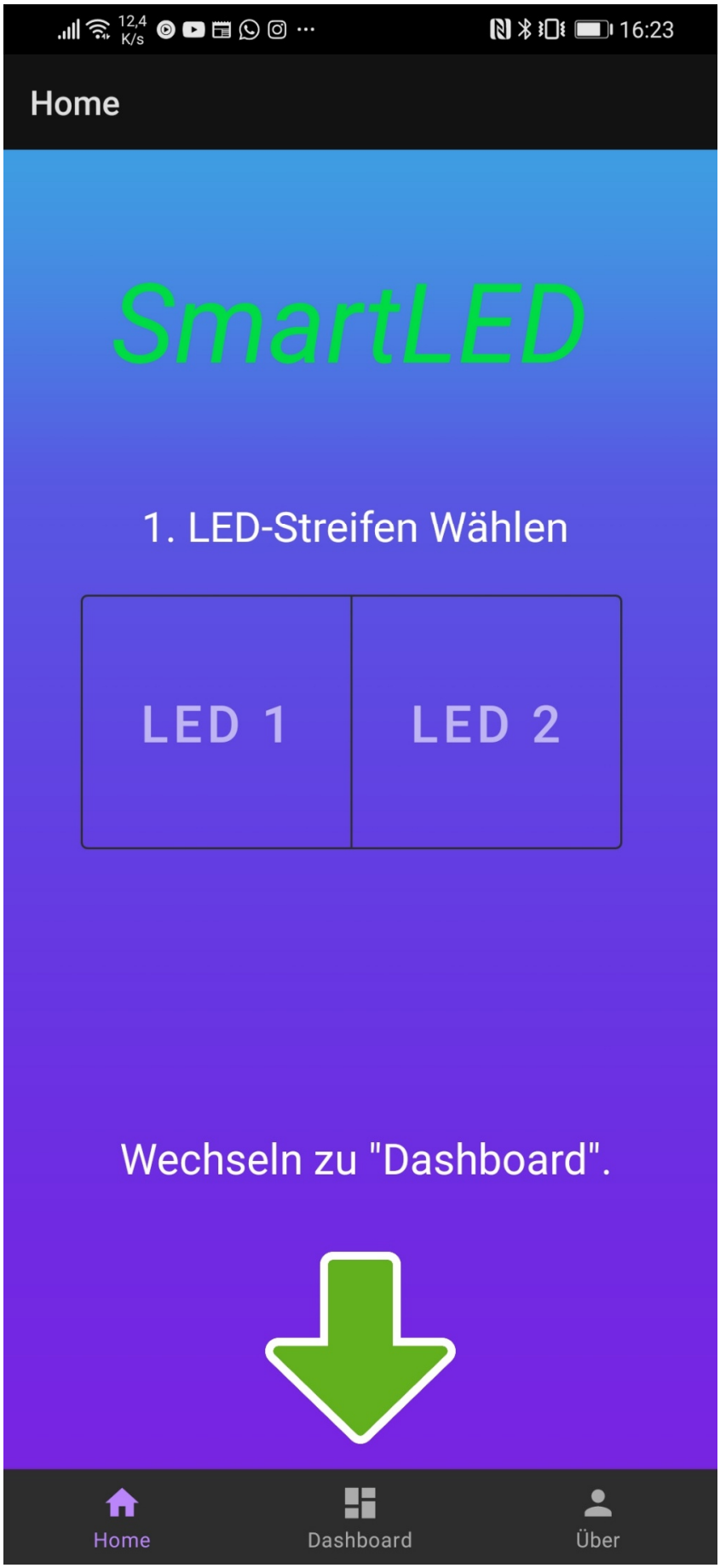
Wie bereits bei dem Screenshot der Menüführung zu sehen, habe ich mich außerdem für eine Rosa Akzentfarbe entschieden. Hier nochmal das Menü:

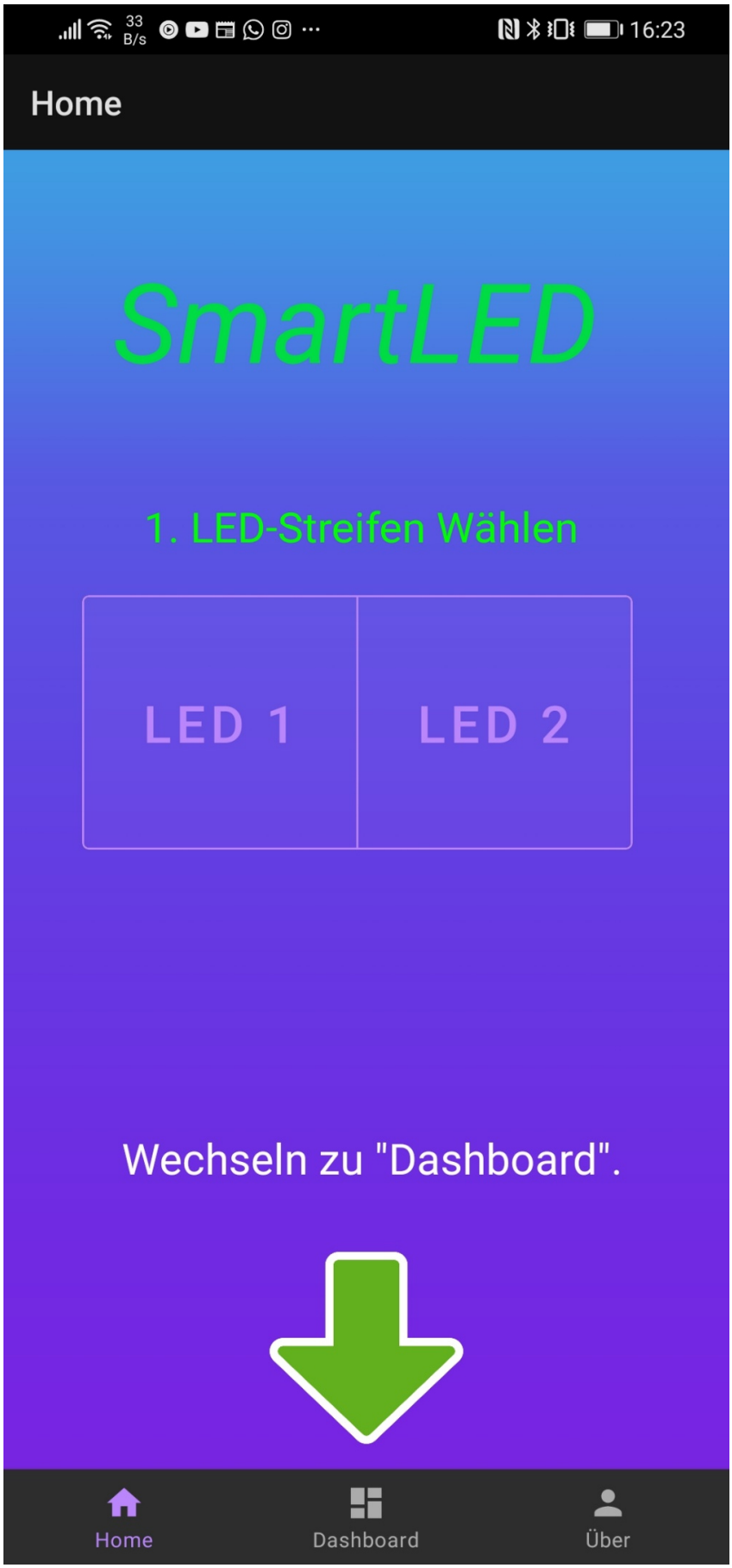


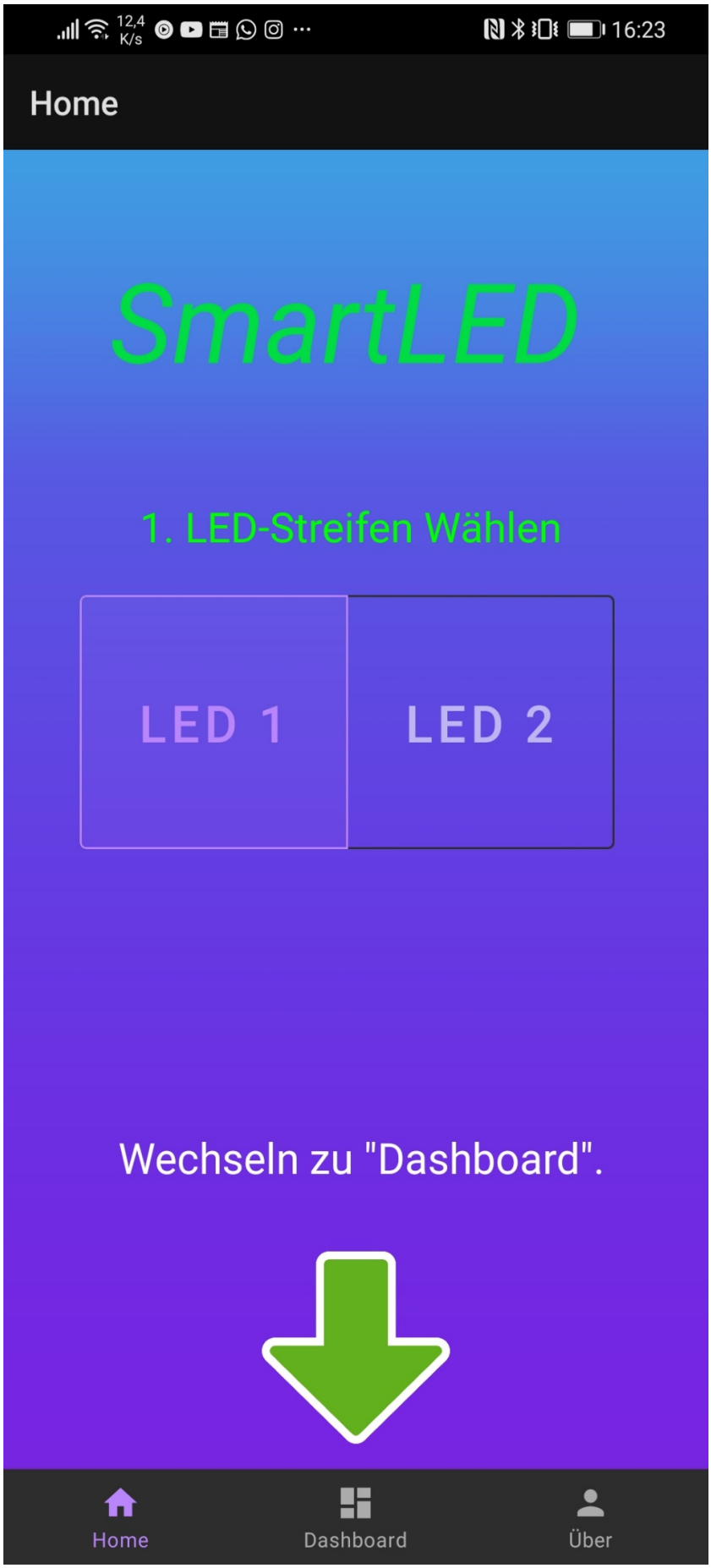
Zu sehen ist die Akzentfarbe unter anderem (wie auf dem Bild) in der Menüleiste bei immer dem Tab, der gerade ausgewählt ist. Zum Zeitpunkt des Screenshots war also die „Über“ Seite ausgewählt bzw. geöffnet. Außerdem ist die Akzentfarbe noch bei den Buttons für die Muster zu sehen, auf dessen Design ich auch gleich eingehe. Gewählt habe ich diese Farbe, da Blau sowie Lila aufgrund des Hintergrundes vergeben waren und da eine Akzentfarbe gleich der Hintergrundfarbe keinen Sinn macht. Es würde sich beißen, deswegen kamen diese beiden Farben nicht in Frage. Grün war außerdem bereits an die „Über“ mich Seite vergeben (das Design für die „Über“ mich Seite existiert zu diesem Zeitpunkt nur in meinem Kopf, beschrieben wird es später), Rot ist zu grell und würde hier nicht reinpassen. Schwarz wäre dem Hintergrund des Menüs zu ähnlich. Hatte ich getestet, mit der Konsequenz, dass weder Symbol noch Text der ausgewählten Seite lesbar waren. Dann kam ich auf Rosa und die Farbe biss sich mit nichts, überschnitt sich mit nichts und gliederte sich zusätzlich noch gut in Hintergrund und die allgemeine Farbgebung ein.

Bis jetzt habe ich rein das Design für die untere Menüführung sowie das Hintergrundbild. Jetzt fehlt noch das Design der drei verschiedenen Seiten: „Home“, „Dashboard“, „Über“.

Nun habe ich mich mit dem Design der ersten App Seite beschäftigt, also der „Home“ Seite auf der das zu steuernde Gerät ausgewählt wird. Bei der Überschrift wählte ich den Schriftzug „SmartLED“, damit jeder sofort erkennt, in welcher App er sich befindet. Darunter folgt der Schriftzug „1. LED-Streifen wählen“. Dieser soll darauf hinweisen, dass auf dem darunter zu findenden toggle-Button der zu steuernde LED-Streifen ausgewählt werden soll. Dieser Schriftzug wird grün, sobald ein LED-Streifen ausgewählt wurde. Die grüne Färbung signalisiert, dass jetzt auf das Dashboard gewechselt werden kann um den LED-Streifen zu steuern. Ist der Schriftzug weiß, ist kein Gerät gewählt, würde jetzt auf das Dashboard gewechselt werden, würde somit nichts passieren, da kein zu steuernder LED-Streifen gewählt wurde. Der Schriftzug ist somit immer grün oder weiß. Beim Start der App erstmal weiß, da zu diesem Zeitpunkt noch kein LED-Streifen gewählt ist. Die Überschrift ist übrigens immer grün, um sich von dem Hintergrund abzuheben, zudem aus selbigem Grunde fett und kursiv. Gleichzeitig beißt sich das Grün nicht mit dem Hintergrund und es bildet eine Einheit mit der „Über“ Seite, die ich auch grün machen werde. Darunter folgen dann nun die sogenannten toggle-Buttons. Es sind zwei viereckige, transparente Buttons mit Umrandung und den Schriftzügen „LED 1“ und „LED 2“. Wird einer dieser Buttons gedrückt, wird er mit der Akzentfarbe, also Rosa markiert. Somit ist deutlich zu erkennen, welcher Button gewählt ist und welcher nicht. Abwählen kann man die Buttons natürlich auch wieder. Auf unterem Screenshot (dem 3.) ist z.B. der erste LED-Streifen angewählt, der 2. nicht. Würde man also in diesem Zustand auf das „Dashboard“ wechseln, steuert man nur den ersten. Natürlich können auch beide gleichzeitig angewählt sein, somit werden in diesem Fall auch beide gleichzeitig gesteuert. Für das Design der Buttons entschied ich mich, da der Button dadurch sehr präsent wirkt. Man erkennt sofort worum es auf der Seite geht, was man machen muss und außerdem ist es einzigartig. Noch nie habe ich in einer App so designte Buttons gesehen, die App hat somit gleich schonmal einen Wiedererkennungswert. Auch hätte es keinen Sinn gemacht, die toggle-Buttons kleiner zu machen, da auf der Seite ja sowieso nicht mehr gemacht wird, für mehr ist die Seite sowieso nicht existent. Außerdem sieht der Button so auch einfach gut in meiner Wahrnehmung aus und hat mir nach langem probieren so am besten zugesagt. Der Begriff der Kachel ist als Beschreibung dessen eigentlich noch zutreffender. Unter den Kacheln bzw. Buttons folgt dann der Schriftzug in weiß „Wechseln zu „Dashboard““ gefolgt von einem Pfeil, der den Schriftzug nochmal graphisch unterstützt bzw. übersetzt. Der Aufbau dieser Seite ist also von oben nach unten, wie man es gewöhnt ist. Der Betrachter der App erblickt zuerst oben die Überschrift, liest dann, dass er einen LED-Streifen wählen soll, anschließend kommen die Buttons, die dies auch ermöglichen. Darunter folgt dann der Schriftzug, der dazu auffordert, auf das Dashboard zu wechseln gefolgt von dem Pfeil zur graphischen Unterstützung. Auch zeigt der Pfeil im unteren Menü auf das „Dashboard“, also dem richtigen Menüpunkt. Falls man durch den Schriftzug nicht wüsste, wie man jetzt auf das Dashboard wechselt, würde das einem der Pfeil zeigen. Hier drei Screenshots der Seite, auf ersterem ist kein LED-Streifen zum Steuern angewählt (Zustand beim Starten der App), auf zweitem der Erste und auf drittem beide. In der App fügte ich zu dem unteren Schriftzug noch eine „2.“ an den Anfang dazu, zur Vereinheitlichung (nachdem untere Screenshots aufgenommen wurden).

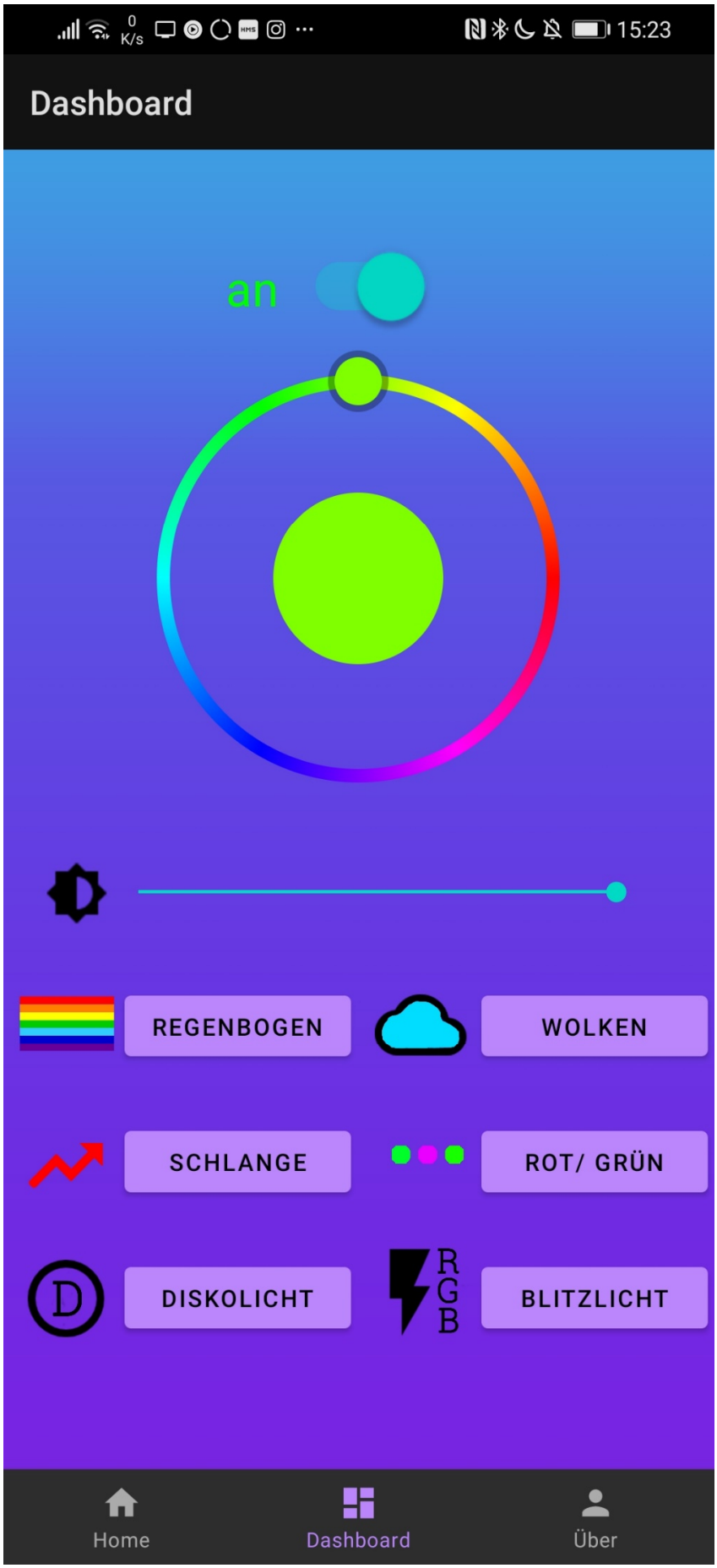

12,4 K/s
16:23
Home
SmartLED
1. LED-Streifen Wählen
LED 1
LED 2
Wechseln zu "Dashboard".
Home
Dashboard
Über


16:23
Home
SmartLED
1. LED-Streifen Wählen
LED 1
LED 2
Wechseln zu "Dashboard".
Home
Dashboard
Über


12,4 K/s
16:23
Home
SmartLED
1. LED-Streifen Wählen
LED 1
LED 2
Wechseln zu "Dashboard".
Home
Dashboard
Über

Nun habe ich mich mit dem Design der zweiten Seite beschäftigt, auf der dann schließlich der vorher ausgewählte LED-Streifen (oder beide) gesteuert werden soll. Zuerst setzte ich einen an/ aus Switch um den LED-Streifen an bzw. aus zu machen. Bei dem Anschalten geht der LED-Streifen auf den Zustand von vor dem Ausschalten zurück. Vor dem Switch ist der Schriftzug „an“ bzw. „aus“ zu finden. Beim Start der App ist der Switch an, und der Schriftzug demnach auch auf „an“. Der Schriftzug „an“ ist immer grün, der Schriftzug „aus“ immer rot. Die Schriftzüge vor dem Switch ersetzen sich jeweils gegenseitig. Darunter ist dann der Color Picker zu finden. Dieser besteht aus einem Kreis auf dem wiederum ein kleiner Kreis bewegt werden kann um die gewünschte Farbe des LED-Streifens einzustellen. In der Mitte des Kreises ist wiederum ein Kreis, gefüllt mit der aktuellen Farbe, die am äußeren Kreis eingestellt ist. Die Farbauswahl ist übrigens live. Also die Farbe ändert sich schon während man mit dem Finger über den Color Picker streicht, nicht erst wenn man loslässt. Gleiches gilt für den Farbwechsel am LED-Streifen. Unter dem Color Picker folgt der Helligkeitsregler, dafür wählte ich eine türkisene seekbar. Türkis deshalb, da Grün komisch aussieht, einfach zu kitschig, dunkles Blau sich mit dem Hintergrund beißt und Rot keinen Sinn macht als Warnfarbe für den Helligkeitsregler. Somit kam ich auf Türkis. Die Akzentfarbe Rosa wählte ich nicht, da die Buttons bereits rosa sind, die unter dem Helligkeitsregler für die Wahl der Muster kommen. Links neben dem Helligkeitsregler entschied ich mich noch für die Ergänzung eines Symboles, das zeigen soll, dass es sich hierbei um die Helligkeit handelt. Das Symbol machte ich selbst aus zwei Quadraten, die aufeinander im 45 Grad Winkel zueinander gedreht angebracht sind. Das eine ist zur Hälfte gefüllt. Die Buttons darunter wählte ich im Gegensatz zur Starseite gefüllt, nicht transparent. Das macht die Seite abwechslungsreicher und da diese Buttons wesentlich kleiner sind, sehen diese in Rosa auch nicht so unpassend aus, wie das die Großen täten. Rosa ist hierbei wieder die Akzentfarbe. Insgesamt gibt es sechs Muster, demnach auch sechs Buttons, um alle sechs Muster anzusteuern.

Hier ein Screenshot der Seite:


15:23
Dashboard
an
REGENBOGEN
WOLKEN
SCHLANGE
ROT/ GRÜN
DISKOLICHT
RGB
BLITZLICHT
Home
Dashboard
Über

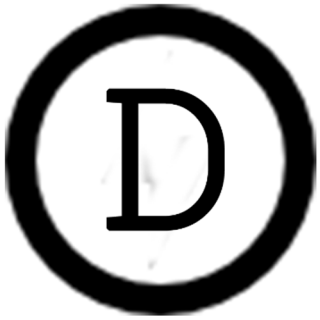
Da die Buttons ursprünglich eigentlich ohne Logo waren, habe ich beschlossenen, noch vor jeden Button ein Logo zu setzen. Zum einen werden die Muster (die Buttons sind ja im Endeffekt nur für die Auswahl der Muster zuständig) dadurch anschaulicher beschrieben, als wenn sie nur mit einem Wort im Button-Text zusammengefasst würden, und es trägt positiv zum Erscheinungsbild der App bei. Hier also meine selbst designten Logos, jedes Logo erstellte ich natürlich abgestimmt auf das Muster:

Dieses Logo ist neben dem Button für das Regenbogen Muster zu finden und symbolisiert, offensichtlich, einfach einen Regenbogen, bzw. dessen Farben, die das Muster wiedergibt.

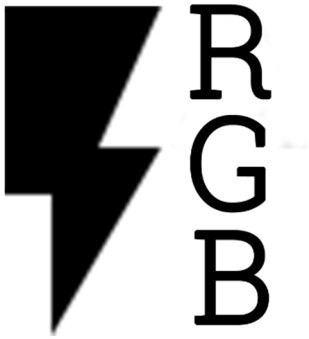
Die Wolke designte ich so, da das Muster in Abschnitten zwischen den Farben Blau und Weiß wechselt, bzw. weiterbewegt. Dieses Weiß und Blau assoziierte ich mit Himmel und Wolken, deshalb die blaue Wolke, um beides zu kombinieren, die Wolke steht für die Wolke und das Blau steht für den Himmel.

Der Pfeil soll eine Schlange symbolisieren, die sich in eine Richtung bewegt. Das Muster baut sich meist in der Farbe Rot (abhängig von dem vorherigen Muster) auf. Also wie eine Schlange.

Bei diesem Muster bewegen sich abwechselnd grüne und lila Farbsegmente über den Streifen. Zwischen ihnen ist jeweils eine Lücke. Dieses Symbol soll einfach deren Verlauf symbolisieren.



Dieses Muster erinnert stark an ein Diskolicht. Es ist eine schnelle Abfolge von verschiedenen LED-Sequenzen, die sich relativ zum LED-Streifen bewegen und dabei die Farbe wechseln. Der Kreis soll hierbei für eine Diskokugel stehen (Dreidimensionalität nicht möglich, da die Symbole in der App viel zu klein dargestellt werden für starke Individualität). Das D steht für Disko oder Diskolicht.



Dieses Muster leuchtet in einer abgedunkelten, zufälligen Farbe, anschließend folgt nach zufälliger Dauer ein weißer Blitz, der ebenfalls von zufälliger Dauer ist, gefolgt wieder von einer abgedunkelten, zufälligen Farbe von zufälliger Dauer usw. Der Blitz symbolisiert also den weißen Blitz des LED-Streifens, und das „R“, „G“ und „B“ jeweils die aus RGB-Farben zusammengesetzten Farben, die zwischen den Blitzen erscheinen. Die Farben sind übrigens im Muster abgedunkelt um den Kontrast zum weißen Blitz zu erhöhen, das lässt ihn heller wirken. Die letzten beiden Symbole sind absichtlich in Schwarz/ Weiß gehalten, da diese zusammengehören, beide sind eher so für Partys oder Diskos gedacht und durch die gleiche Farbverwendung kommt es zu einer automatischen, gemeinsamen Kategorisierung der beiden Symbole. Es wird direkt erkannt, dass diese zusammengehören.

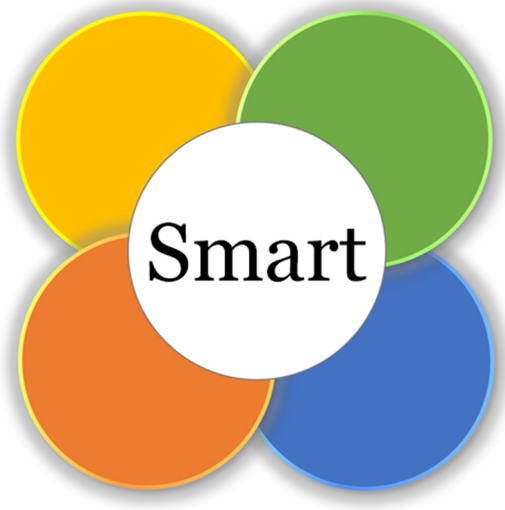
Zu berücksichtigen ist außerdem der technische Aufwand, der benötigt wird, all das im Kopf vorgestellte in einer App umzusetzen, denn es ist auch jederzeit wichtig, das Design für andere Handy-Größen (kleinere/ größere Handys) zu optimieren. So einfach wie in Word, ein paar Sachen per Drag & Drop zu schieben, ist dies also bei weitem nicht, daher auch die langen Zeiten.

Kalenderwoche 32 (ca. 2-3 Stunden)

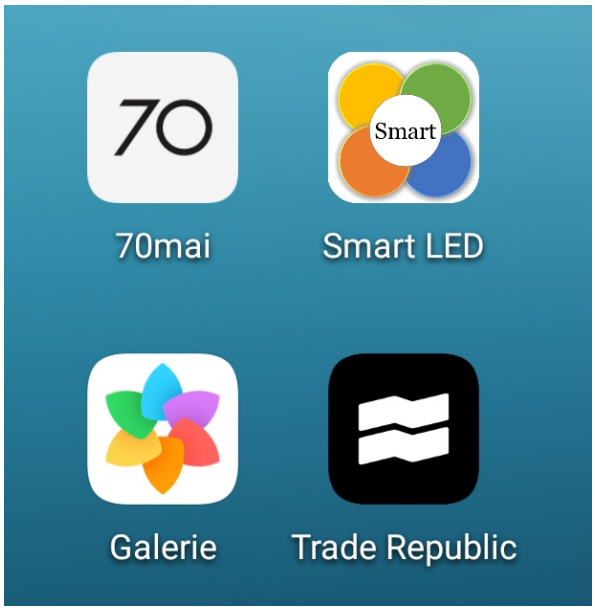
Anschließend habe ich das App-Logo designet. Dazu habe ich Photoshop genutzt. Zunächst habe ich vier Kreise erstellt und gefüllt in den RGB Farben plus Gelb, um auf vier zu kommen. Den Farbkreisen habe ich dann noch Schatten gegeben und in der Mitte einen weißen Kreis darüber gelegt in dem wiederum in weißer Schrift „Smart“ steht. Dies soll eine Referenz zu dem Projektnamen, SmartLED, darstellen. Auf dieses Design kam ich erst nach langem probieren und testen verschiedener Ideen. Gewählt habe ich dies letztendlich, weil es mich grafisch sehr anspricht, ich es noch nie bei einer anderen App gesehen habe und weil es inhaltlich zum Projekt passt. Außerdem ist es sehr symmetrisch, hat abgerundete Ecken, einen weißen Hintergrund und bindet sich durch all das gut zwischen all den anderen Apps auf dem Handy ein.

Und dann kam, wie bei allem was ich für dieses Projekt designe, noch die technische Arbeit dazu, das App-Logo in Android Studio einzubinden, in den richtigen Größen zu skalieren etc.

Hier das fertige App-Logo:



So Sieht das Logo neben anderen Apps aus:



Kalenderwoche 33 (ca. 3 Stunden)

In Kalenderwoche 33 habe ich mich dann mit dem Design meiner letzten App-Seite beschäftigt, also der „Über“ Seite, auf der in erster Linie Informationen zum Ersteller der App zu finden sein sollen, also zu mir. Der Hintergrund ist wie bei allen anderen Seiten der blaue Farbverlauf um ein einheitliches Design zu gewährleisten. Nachdem ich mich informiert hatte, welche Informationen in einem „Über“ Tab sollten, habe ich mir Gedanken zu meinem gemacht.

Zuerst habe ich mich für ein Logo entschieden, das ganz oben auf dem Bildschirm erscheinen soll. Das designte ich mir zunächst in Photoshop. Ich entschied mich für die zwei Buchstaben „me“, da diese einmal die Anfangsbuchstaben meines Vor- und Zunamen entsprechen und außerdem als Wort gelesen, im Englischen übersetzt „mich“ heißen kann, passend zu dem Thema der App-Seite, in der es ja um „mich“ geht. Diese zwei Buchstaben habe ich dann in Photoshop noch ein bisschen aufgehübscht, und mit Photoshop darunter den Rest meiner „Über“ Seite erstellt. Unter den Buchstaben habe ich als Erklärung der Buchstaben meinen Namen gesetzt und unter diesem wiederum Impressum als Überschrift gewählt, dass es sich dabei um eine Überschrift handelt, signalisiere ich durch Kursivität und Unterstreichung des Begriffs. Darunter folgen Angaben zu meinem Wohnort, also das Typische was in ein Impressum gehört. Als Nächstes folgt eine weitere Überschrift im gleichen Design der vorherigen. Diese lautet „Kontakt“ und unter ihr ist eine extra für das Projekt erstellte E-Mail-Adresse angegeben. Als Letztes folgt schließlich noch eine weitere Überschrift, „Mehr“, unter dieser eine URL für weitere Informationen, z.B. um eine Anleitung zu erhalten, angegeben ist. Für die Erstellung einer eigenen Webseite nur für das Projekt habe ich mich kurzfristig während des Erstellens dieser „Seite“ entschieden. Das werde ich aber ganz am Ende angehen, da das kein obligatorischer Teil meines Projektes ist, ich es aber wenn Zeit über ist, gerne noch verwirklichen würde. Für die Farbe Grün habe ich mich entschieden, da diese als eine von wenigen bei dem blauen Verlaufshintergrund gut lesbar ist und optisch gut passt.

Hier ein Screenshot der Seite:


9,7 K/s
100 %
13:34
Über
me
Maximilian Enzinger
Impressum
Maximilian Enzinger
Bischof-Otto-Weg 9
91086 Aurachtal
Kontakt
info@smartled.app
Mehr
www.smartled.app
Home
Dashboard
Über

Kalenderwoche 39 (ca. 2 Stunde)

In Kalenderwoche 39 beschäftigte ich mich mit der Anleitung, die ich noch mit beilegen will. Alles andere, bis auf die Webseite, ist fertig. Die App ist fertig und funktioniert, der LED-Streifen sowie die zugehörige Verpackung sind ebenfalls fertig. Die Langzeitdokumentation ist ebenfalls fertig (nach dieser KW).

Zum Erstellen der Anleitung gibt es eigentlich nicht viel zu sagen, das Ergebnis wird unten angehängt. Zuerst habe ich in Word mich für eine blaue Überschrift entschieden um das Format beizubehalten und dann eben aufgeschrieben, was chronologisch die richtige Vorgehensweise bei der Benutzung meines Projektes ist. Im oberen Teil beschrieb ich alle Inhalte des USB-Sticks und den der Verpackung, also aus was mein Projekt besteht. Darunter dann wie es zu benutzen ist. Unter dieser Aufbauanleitung habe ich dann noch eine kurze Anleitung zur App selbst gemacht. Auf der Rückseite folgt eine Fehlerbeschreibung, um bei dem unwahrscheinlichen Auftreten von Fehlern eine Lösung anbieten zu können. Hier ein Screenshot der Vorderseite:

1. Projekt-Umfang

- Im Koffer: USB-Stick, 1x Wlan-Hotspot, 2x Netzteile, 2x Aufgerollte LED-Streifen, 2x Weiße Boxen für die Elektronik.
- Auf dem USB-Stick: Langzeitdokumentation, Android App Download, Übersicht über verwendete Materialien, Trailer, Bilder.

2. Anleitung Aufbau

1. Nehmen Sie alle Teile aus dem Koffer und Rollen Sie anschließend beide LED-Streifen aus (Bei Benutzung im aufgerollten Zustand kann es zu Überhitzungen kommen).
2. Zum Einschalten des Netgear Wlan-Routers drücken Sie für 3 Sekunden auf die schwarze Taste am oberen Rand des Gerätes (Dieser Simuliert das Haus-Wlan, ist also bei einer festen Installation in einer Wohnung nicht mehr nötig). Nach der Benutzung möglichst den Wlan-Hotspot durch drei Sekunden drücken der Ausschalt-Taste und anschließendem Druck auf den „Shut Down“-Button in dem Display ausschalten. Falls das Gerät doch einmal leer sein sollte, wird im Koffer ein Kabel zum Laden beigelegt.
3. Nehmen Sie sich einen der beiden LED-Streifen und stecken Sie ihn in den einzig passenden Anschluss in einer der beiden Boxen. Stecken Sie anschließend eines der Netzteile an die andere Seite der weißen Box an. Wiederholen Sie diesen Vorgang für den 2. Led-Streifen.
4. Schließen Sie nun beide Netzteile an den Strom an. Beide LED-Streifen Leuchten nun Rot. Wenn diese mit dem Wlan verbunden sind und einsatzbereit sind werden diese zu Grün wechseln.
5. Installieren Sie sich nun die App „SmartLED“ zum Steuern der beiden LED-Streifen. Die App kann unter www.smartled.app/download.html heruntergeladen werden.
6. Nach der Installation verbinden Sie sich mit dem Wlan „SmartLED“, das Passwort ist ebenfalls „SmartLED“ (In einer festen Haus-Installation wäre dieser Schritt ebenfalls nicht nötig, da das Haus-Wlan zum Steuern verwendet werden würde, mit dem man ja sowieso verbunden ist).
7. Nun öffnen Sie die App und Sie können beide LED-Streifen steuern.

3. Anleitung App

- Die App besteht aus drei verschiedenen Seiten, die im unteren Navigationsmenü gewechselt werden können. Auf ersterer „Home“ können Sie auswählen ob einer der beiden oder beide LED-Streifen gleichzeitig gesteuert werden sollen, anschließend muss auf die zweite Seite „Dashboard“ gewechselt werden um die vorher ausgewählten LED-Streifen zu steuern.
- Um die zu steuernden LED-Streifen erneut auszuwählen, muss zurück auf die Seite „Home“ gewechselt werden, auf dieser können dann erneut die zu steuernden LED-Streifen angewählt werden.

Bitte Rückseite beachten.

Hier ein Screenshot der Rückseite:

4. Fehlerbeschreibung und Lösung

1. Der LED-Streifen leuchtet ausschließlich rot und wechselt auch 20 Sekunden nach dem Anstecken des Netzteils nicht die Farbe zu grün.
   - ➔ Ist nur einer der beiden Streifen betroffen muss das Netzteil für ca. 10 Sekunden aus der Steckdose gezogen und anschließend wieder gesteckt werden.
   - ➔ Sind beide Streifen betroffen so ist der Wlan-Hotspot Akku leer oder er ist nicht eingeschaltet.

2. Der Wlan-Hotspot schaltet sich auch nach 5-10 Sekunden drücken der Einschalttaste nicht an.
   - ➔ Dann ist vermutlich der Akku leer. Ein Kabel zum Aufladen ist beigelegt. Nach einer kurzen Ladezeit ist der Hotspot wieder einsatzbereit.

3. Einer der beiden LED-Streifen geht plötzlich aus und leuchtet anschließend nur noch rot.
   - ➔ Der LED-Streifen ist abgestürzt, das Netzteil muss für 10 Sekunden gezogen und anschließend wieder in die Steckdose eingesteckt werden.

4. Beide LED-Streifen leuchten nach dem Einstecken Grün, lassen sich aber durch die App nicht steuern.
   - ➔ Das Handy das zum Steuern genutzt werden soll befindet sich nicht in dem Wlan „SmartLED“.
   - ➔ Auf der Startseite „Home“ wurden vor dem Wechsel zu „Dashboard“ kein LED-Streifen ausgewählt.

Nachdem ich die Anleitung fertig hatte und damit alles was ich zur Abgabe beilegen muss, habe ich mich noch mit der Webseite beschäftigt. Da ich die Langzeitdokumentation an dieser Stelle abschicke, ist die Webseite nicht mehr Teil der Dokumentation. Würde mich aber freuen, wenn auf dieser vorbeigeschaut wird: www.smartled.app.

Auf den nächsten Seiten folgt schließlich noch der komplette Programmtext.

```
 1 package de.me.smartled.ui.dashboard;
 2
 3 import android.graphics.Color;
 4 import android.os.AsyncTask;
 5 import android.os.Build;
 6 import android.os.Bundle;
 7 import android.view.LayoutInflater;
 8 import android.view.View;
 9 import android.view.ViewGroup;
10 import android.widget.Button;
11 import android.widget.CompoundButton;
12 import android.widget.SeekBar;
13 import android.widget.Switch;
14 import androidx.annotation.NonNull;
15 import androidx.annotation.RequiresApi;
16 import androidx.fragment.app.Fragment;
17
18 import java.net.DatagramPacket;
19 import java.net.DatagramSocket;
20 import java.net.InetAddress;
21
22 import de.me.smartled.MainActivity;
23 import de.me.smartled.R;
24
25 public class DashboardFragment extends Fragment {
26
27     Switch simpleSwitch;
28     View view;
29     Button buttonregenbogen;
30     Button buttonwolken;
31     Button buttonlilaweiss;
32     Button buttonblitzlicht;
33     Button buttondiskolicht;
34     Button buttonschlangerot;
35     SeekBar seekbarbrightness;
36     ColorPicker picker;
37     int newBrightness;
38
39
40     public View onCreateView(@NonNull LayoutInflater inflater,
41                              ViewGroup container, Bundle savedInstanceState) {
42
43         view = inflater.inflate(R.layout.fragment_dashboard, container, false);
44         simpleSwitch = (Switch) view.findViewById(R.id.onoff);
45         buttonregenbogen = (Button) view.findViewById(R.id.button_regenbogen);
46         buttonwolken = (Button) view.findViewById(R.id.button_wolken);
47         buttonlilaweiss = (Button) view.findViewById(R.id.button_lila_weiss);
48         buttonblitzlicht = (Button) view.findViewById(R.id.button_blitzlicht);
49         buttondiskolicht = (Button) view.findViewById(R.id.button_diskolicht);
50         buttonschlangerot = (Button) view.findViewById(R.id.Schlange_Rot);
51         seekbarbrightness =(SeekBar) view.findViewById(R.id.seek_bar_brightness);
52         picker = (ColorPicker) view.findViewById(R.id.picker);
53         simpleSwitch.setText("an");
54         simpleSwitch.setChecked(true);
55         simpleSwitch.setTextColor(Color.GREEN);
56         newBrightness = 255;
57
58
59         simpleSwitch.setOnCheckedChangeListener(new CompoundButton.
   OnCheckedChangeListener() {
60             @RequiresApi(api = Build.VERSION_CODES.P)
61             @Override
62             public void onCheckedChanged(CompoundButton buttonView, boolean isChecked
   ) {
63                 if (isChecked){
64                     byte[] buf=new byte[6];
65                     buf[0]=1;
```

```
66                      buf[1]= (byte) newBrightness;
67                      AsyncTaskRunner runner = new AsyncTaskRunner();
68                      runner.execute(buf);
69                      simpleSwitch.setText("an");
70                      simpleSwitch.setTextColor(Color.GREEN);
71                  }
72                  else
73                  {
74                      byte[] buf=new byte[6];
75                      buf[0]=1;
76                      buf[1]= (byte) 0;
77                      buf[4]= (byte) 1;
78                      AsyncTaskRunner runner = new AsyncTaskRunner();
79                      runner.execute(buf);
80                      simpleSwitch.setText("aus");
81                      simpleSwitch.setTextColor(Color.RED);
82                  }
83              }
84          });
85
86          buttonregenbogen.setOnClickListener(new View.OnClickListener() {
87              public void onClick(View v) {
88                  byte[] buf = new byte[6];
89                  buf[0] = 3;
90                  buf[1] = (byte) 1;
91                  AsyncTaskRunner runner = new AsyncTaskRunner();
92                  runner.execute(buf);
93              }
94
95          });
96
97          buttonwolken.setOnClickListener(new View.OnClickListener() {
98              public void onClick(View v) {
99                  byte[] buf = new byte[6];
100                 buf[0] = 3;
101                 buf[1] = (byte) 2;
102                 AsyncTaskRunner runner = new AsyncTaskRunner();
103                 runner.execute(buf);
104             }
105
106         });
107
108         buttonlilaweiss.setOnClickListener(new View.OnClickListener() {
109             public void onClick(View v) {
110                 byte[] buf = new byte[6];
111                 buf[0] = 3;
112                 buf[1] = (byte) 3;
113                 AsyncTaskRunner runner = new AsyncTaskRunner();
114                 runner.execute(buf);
115             }
116
117         });
118
119         buttonblitzlicht.setOnClickListener(new View.OnClickListener() {
120             public void onClick(View v) {
121                 byte[] buf = new byte[6];
122                 buf[0] = 3;
123                 buf[1] = (byte) 4;
124                 AsyncTaskRunner runner = new AsyncTaskRunner();
125                 runner.execute(buf);
126             }
127
128         });
129
130         buttondiskolicht.setOnClickListener(new View.OnClickListener() {
131             public void onClick(View v) {
132                 byte[] buf = new byte[6];
```

```
133                 buf[0] = 3;
134                 buf[1] = (byte) 5;
135                 AsyncTaskRunner runner = new AsyncTaskRunner();
136                 runner.execute(buf);
137             }
138 
139         });
140 
141         buttonschlangerot.setOnClickListener(new View.OnClickListener() {
142             public void onClick(View v) {
143                 byte[] buf = new byte[6];
144                 buf[0] = 3;
145                 buf[1] = (byte) 6;
146                 AsyncTaskRunner runner = new AsyncTaskRunner();
147                 runner.execute(buf);
148             }
149 
150         });
151 
152         seekbarbrightness.setOnSeekBarChangeListener(new SeekBar.
    OnSeekBarChangeListener() {
153             int progressChangedValue = 0;
154 
155             public void onProgressChanged(SeekBar seekBar, int progress, boolean
    fromUser) {
156                 progressChangedValue = progress;
157                 byte[] buf = new byte[6];
158                 buf[0] = 1;
159                 buf[1] = (byte) progressChangedValue;
160                 newBrightness = progressChangedValue;
161                 AsyncTaskRunner runner = new AsyncTaskRunner();
162                 runner.execute(buf);
163             }
164 
165             public void onStartTrackingTouch(SeekBar seekBar) {
166             }
167             public void onStopTrackingTouch(SeekBar seekBar) {
168             }
169         });
170 
171         picker.setOnColorChangedListener(new ColorPicker.OnColorChangedListener() {
172             @Override
173             public void onColorChanged(int color) {
174                 int varb;
175                 int varg;
176                 int varr;
177                 int aktuelleFarbe;
178                 aktuelleFarbe = picker.getColor();
179                 varb = aktuelleFarbe & 0xff;
180                 varg = (aktuelleFarbe >> 8) & 0xff;
181                 varr = (aktuelleFarbe >> 16) & 0xff;
182 
183                 byte[] buf = new byte[6];
184                 buf[0] = 2;
185                 buf[1] = (byte) varr;
186                 buf[2] = (byte) varg;
187                 buf[3] = (byte) varb;
188                 AsyncTaskRunner runner = new AsyncTaskRunner();
189                 runner.execute(buf);
190             }});
191         picker.setShowOldCenterColor(false);
192         return view;
193     }
194 
195     private class AsyncTaskRunner extends AsyncTask<byte[], byte[], byte[]> {
196         DatagramSocket ds = null;
197         DatagramPacket dp;
```

```
198         String ip1 = "192.168.6.200";
199         String ip2 = "192.168.6.201";
200         String ip = "";
201
202         @Override
203
204         protected byte[] doInBackground(byte[]... params) {
205             for(int i = 0; i<=1; i++)
206             {
207                if(i==0 && MainActivity.button1checked)
208                    ip = ip1;
209                else{
210                    if(i==1 && MainActivity.button2checked)
211                        ip = ip2;
212                    else
213                        continue;
214                }
215
216
217
218                try {
219                   ds = new DatagramSocket();
220                   dp = new DatagramPacket(params[0], 6, InetAddress.getByName(ip),
    1234);
221                   ds.setBroadcast(false);
222                   ds.send(dp);
223                } catch (Exception e) {
224                    e.printStackTrace();
225                    //resp = e.getMessage();
226                } finally {
227                    if (ds != null) {
228                        ds.close();
229                    }
230                }
231             }
232             return params[0];
233         }
234     }
235 }
236
```

File - C:\Users\Max\AndroidStudioProjects\v7 2\SmartLED 2\SmartLED 2\app\src\main\res\layout\fragment_dashboard.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.
android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.dashboard.DashboardFragment">

    <Switch
        android:id="@+id/onoff"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:buttonTint="@color/colorAccent"
        android:scaleX="1.2"
        android:scaleY="1.2"
        android:text="Switch"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/guideline3" />


    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.98" />

    <Button
        android:id="@+id/button_regenbogen"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Regenbogen"
        app:layout_constraintBottom_toTopOf="@+id/Schlange_Rot"
        app:layout_constraintEnd_toStartOf="@+id/imageView8"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="@+id/guidelineVert"
        app:layout_constraintTop_toTopOf="@+id/guideline" />

    <Button
        android:id="@+id/button_diskolicht"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="1dp"
        android:text="Diskolicht"
        app:layout_constraintBottom_toTopOf="@+id/guideline4"
        app:layout_constraintEnd_toStartOf="@+id/imageView5"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="@+id/guidelineVert"
        app:layout_constraintTop_toBottomOf="@+id/Schlange_Rot" />

    <Button
        android:id="@+id/Schlange_Rot"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="1dp"
        android:text="Schlange"
        app:layout_constraintBottom_toTopOf="@+id/button_diskolicht"
        app:layout_constraintEnd_toStartOf="@+id/imageView9"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="@+id/guidelineVert"
        app:layout_constraintTop_toBottomOf="@+id/button_regenbogen" />

    <Button
        android:id="@+id/button_wolken"
```

```
67          android:layout_width="0dp"
68          android:layout_height="wrap_content"
69          android:text="Wolken"
70          app:layout_constraintBottom_toTopOf="@+id/button_lila_weiss"
71          app:layout_constraintEnd_toEndOf="parent"
72          app:layout_constraintHorizontal_bias="0.5"
73          app:layout_constraintStart_toStartOf="@+id/guidelineVert4"
74          app:layout_constraintTop_toTopOf="@+id/guideline" />
75
76      <Button
77          android:id="@+id/button_lila_weiss"
78          android:layout_width="0dp"
79          android:layout_height="wrap_content"
80          android:text="Rot/ Grün"
81          app:layout_constraintBottom_toTopOf="@+id/button_blitzlicht"
82          app:layout_constraintEnd_toEndOf="parent"
83          app:layout_constraintHorizontal_bias="0.5"
84          app:layout_constraintStart_toStartOf="@+id/guidelineVert4"
85          app:layout_constraintTop_toBottomOf="@+id/button_wolken" />
86
87      <Button
88          android:id="@+id/button_blitzlicht"
89          android:layout_width="0dp"
90          android:layout_height="wrap_content"
91          android:text="Blitzlicht"
92          app:layout_constraintBottom_toTopOf="@+id/guideline4"
93          app:layout_constraintEnd_toEndOf="parent"
94          app:layout_constraintHorizontal_bias="0.5"
95          app:layout_constraintStart_toStartOf="@+id/guidelineVert4"
96          app:layout_constraintTop_toBottomOf="@+id/button_lila_weiss" />
97
98      <SeekBar
99          android:id="@+id/seek_bar_brightness"
100         android:layout_width="317dp"
101         android:layout_height="20dp"
102         android:layout_centerVertical="true"
103         android:layout_marginBottom="6dp"
104         android:max="255"
105         android:maxHeight="6dip"
106         android:minHeight="6dip"
107         android:paddingLeft="50px"
108         android:paddingTop="10px"
109         android:paddingRight="50px"
110         android:paddingBottom="10px"
111         android:progress="255"
112         app:layout_constraintBottom_toTopOf="@+id/guideline"
113         app:layout_constraintEnd_toEndOf="parent"
114         app:layout_constraintHorizontal_bias="0.85"
115         app:layout_constraintStart_toEndOf="@+id/imageView2"
116         app:layout_constraintStart_toStartOf="parent" />
117
118     <de.me.smartled.ui.dashboard.ColorPicker
119         android:id="@+id/picker"
120         android:layout_width="wrap_content"
121         android:layout_height="0dp"
122         app:layout_constraintBottom_toTopOf="@+id/seek_bar_brightness"
123         app:layout_constraintEnd_toEndOf="parent"
124         app:layout_constraintHorizontal_bias="0.5"
125         app:layout_constraintStart_toStartOf="parent"
126         app:layout_constraintTop_toTopOf="@+id/guideline2" />
127
128     <ImageView
129         android:id="@+id/imageView2"
130         android:layout_width="41dp"
131         android:layout_height="38dp"
132         android:src="@mipmap/helligkeit"
133         app:layout_constraintBottom_toBottomOf="@+id/seek_bar_brightness"
```

```
134         app:layout_constraintEnd_toStartOf="@+id/seek_bar_brightness"
135         app:layout_constraintHorizontal_bias="0.5"
136         app:layout_constraintStart_toStartOf="parent"
137         app:layout_constraintTop_toTopOf="@+id/seek_bar_brightness" />
138
139     <ImageView
140         android:id="@+id/imageView4"
141         android:layout_width="0dp"
142         android:layout_height="55dp"
143         android:src="@mipmap/rainbow"
144         app:layout_constraintEnd_toStartOf="@+id/guidelineVert1"
145         app:layout_constraintStart_toStartOf="parent"
146         app:layout_constraintTop_toTopOf="@+id/button_regenbogen" />
147
148     <ImageView
149         android:id="@+id/imageView5"
150         android:layout_width="0dp"
151         android:layout_height="55dp"
152         android:src="@mipmap/bltzlicht"
153         app:layout_constraintEnd_toStartOf="@+id/guidelineVert3"
154         app:layout_constraintHorizontal_bias="0.5"
155         app:layout_constraintStart_toStartOf="@+id/guidelineVert2"
156         app:layout_constraintTop_toTopOf="@+id/button_blitzlicht" />
157
158     <ImageView
159         android:id="@+id/imageView6"
160         android:layout_width="0dp"
161         android:layout_height="55dp"
162         android:src="@mipmap/pfeil"
163         app:layout_constraintEnd_toStartOf="@+id/guidelineVert1"
164         app:layout_constraintHorizontal_bias="0.5"
165         app:layout_constraintStart_toStartOf="parent"
166         app:layout_constraintTop_toTopOf="@+id/Schlange_Rot" />
167
168     <ImageView
169         android:id="@+id/imageView7"
170         android:layout_width="0dp"
171         android:layout_height="55dp"
172         android:src="@mipmap/diskolicht"
173         app:layout_constraintEnd_toStartOf="@+id/guidelineVert1"
174         app:layout_constraintHorizontal_bias="0.5"
175         app:layout_constraintStart_toStartOf="parent"
176         app:layout_constraintTop_toTopOf="@+id/button_diskolicht" />
177
178     <ImageView
179         android:id="@+id/imageView8"
180         android:layout_width="0dp"
181         android:layout_height="55dp"
182         android:src="@mipmap/wolke"
183         app:layout_constraintEnd_toStartOf="@+id/guidelineVert3"
184         app:layout_constraintHorizontal_bias="0.5"
185         app:layout_constraintStart_toStartOf="@+id/guidelineVert2"
186         app:layout_constraintTop_toTopOf="@+id/button_wolken"
187         app:layout_constraintVertical_chainStyle="spread" />
188
189     <ImageView
190         android:id="@+id/imageView9"
191         android:layout_width="0dp"
192         android:layout_height="55dp"
193         android:src="@mipmap/rotgruen"
194         app:layout_constraintEnd_toStartOf="@+id/guidelineVert3"
195         app:layout_constraintHorizontal_bias="0.5"
196         app:layout_constraintStart_toStartOf="@+id/guidelineVert2"
197         app:layout_constraintTop_toTopOf="@+id/button_lila_weiss" />
198
199     <androidx.constraintlayout.widget.Guideline
200         android:id="@+id/guidelineVert1"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.10" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guidelineVert"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.11" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guidelineVert2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.50" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guidelineVert3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.60" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guidelineVert4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.61" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.65" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.09" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_begin="16dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

```java
package de.me.smartled.ui;

import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;

import de.me.smartled.MainActivity;
import de.me.smartled.R;
import de.me.smartled.ui.home.HomeViewModel;


public class HomeFragment extends Fragment {

    private HomeViewModel homeViewModel;
    Button button_1, button_2;
    View view;
    TextView textviewtest;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {

            homeViewModel =
                    ViewModelProviders.of(this).get(HomeViewModel.class);
            View root = inflater.inflate(R.layout.fragment_home, container, false);
            final TextView textView = root.findViewById(R.id.textView5);

            homeViewModel.getText().observe(getViewLifecycleOwner(), new Observer<
String>() {
                @Override
                public void onChanged(@Nullable String s) {
                    textView.setText(s);
                }
            });


        view = inflater.inflate(R.layout.fragment_home, container, false);

        button_1 = (Button) view.findViewById(R.id.button1);
        button_2 = (Button) view.findViewById(R.id.button2);
        textviewtest = (TextView) view.findViewById(R.id.textView5);
        MainActivity.button1checked= false;
        MainActivity.button2checked= false;

        button_1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (MainActivity.button1checked == true) {
                    MainActivity.button1checked = false;
                }
                else {
                    textviewtest.setTextColor(Color.GREEN);
                    MainActivity.button1checked = true;
                }
            }
        });

        button_2.setOnClickListener(new View.OnClickListener() {
```

```
67              @Override
68              public void onClick(View v) {
69                  if (MainActivity.button2checked == true) {
70                      MainActivity.button2checked = false;
71                  }
72                  else {
73                      textviewtest.setTextColor(Color.GREEN);
74                      MainActivity.button2checked = true;
75                  }
76              }
77          });
78
79          return view;
80
81
82      }
83  }
84
85
86
87
88
```

File - C:\Users\Max\AndroidStudioProjects\v7 2\SmartLED 2\SmartLED 2\app\src\main\res\layout\fragment_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.
android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.HomeFragment">


    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="150dp"
        android:layout_marginEnd="150dp"
        android:text="@string/smartled"
        android:textColor="#00dc43"
        android:textSize="65dp"
        android:textStyle="italic"
        app:layout_constraintBottom_toTopOf="@+id/textView5"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="171dp"
        android:layout_marginEnd="181dp"
        android:text="1. LED-Streifen Wählen"
        android:textColor="#FFFFFF"
        android:textSize="25dp"
        app:layout_constraintBottom_toTopOf="@+id/toggleButton"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <TextView
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:text='Wechseln zu "Dashboard".'
        android:textColor="#FFFFFF"
        android:textSize="25sp"
        app:layout_constraintBottom_toTopOf="@+id/imageView10"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toggleButton" />

    <com.google.android.material.button.MaterialButtonToggleGroup
        android:id="@+id/toggleButton"
        android:layout_width="332dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="35dp"
        android:layout_marginEnd="35dp"
        app:layout_constraintBottom_toTopOf="@+id/textView6"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView5">

```

File - C:\Users\Max\AndroidStudioProjects\v7 2\SmartLED 2\SmartLED 2\app\src\main\res\layout\fragment_home.xml

```
        <Button
            android:id="@+id/button1"
            style="?attr/materialButtonOutlinedStyle"
            android:layout_width="161dp"
            android:layout_height="161dp"
            android:text="LED 1"
            android:textSize="30dp" />


        <Button
            android:id="@+id/button2"
            style="?attr/materialButtonOutlinedStyle"
            android:layout_width="161dp"
            android:layout_height="161dp"
            android:text="LED 2"
            android:textSize="30dp" />

    </com.google.android.material.button.MaterialButtonToggleGroup>

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.80" />

    <ImageView
        android:id="@+id/imageView10"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:src="@mipmap/arrow"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/guideline" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

```java
package de.me.smartled;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;


import com.google.android.material.bottomnavigation.BottomNavigationView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;



public class MainActivity extends AppCompatActivity {
    static public boolean button1checked;
    static public boolean button2checked;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        BottomNavigationView navView = findViewById(R.id.nav_view);
        AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(
                R.id.navigation_home, R.id.navigation_dashboard, R.id.navigation_notifications)
                .build();
        NavController navController = Navigation.findNavController(this, R.id.nav_host_fragment);
        NavigationUI.setupActionBarWithNavController(this, navController, appBarConfiguration);
        NavigationUI.setupWithNavController(navView, navController);


    }


}

```

File - C:\Users\Max\AndroidStudioProjects\v7 2\SmartLED 2\SmartLED 2\app\src\main\res\navigation\mobile_navigation.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mobile_navigation"
    app:startDestination="@+id/navigation_home">

    <fragment
        android:id="@+id/navigation_home"
        android:name="de.me.smartled.ui.HomeFragment"
        android:label="@string/title_home"
        tools:layout="@layout/fragment_home" />

    <fragment
        android:id="@+id/navigation_dashboard"
        android:name="de.me.smartled.ui.dashboard.DashboardFragment"
        android:label="@string/title_dashboard"
        tools:layout="@layout/fragment_dashboard" />

    <fragment
        android:id="@+id/navigation_notifications"
        android:name="de.me.smartled.ui.notifications.NotificationsFragment"
        android:label="@string/title_about"
        tools:layout="@layout/fragment_notifications" />
</navigation>

```

File - C:\Users\Max\AndroidStudioProjects\v7 2\SmartLED 2\SmartLED 2\app\src\main\res\values\strings.xml

```xml
<resources>
    <string name="app_name">Smart LED</string>
    <string name="title_home">Home</string>
    <string name="title_dashboard">Dashboard</string>
    <string name="title_notifications">Notifications</string>
    <string name="title_about">Über</string>
    <string name="smartled">SmartLED</string>
</resources>

```

File - C:\Users\Max\AndroidStudioProjects\v7 2\SmartLED 2\SmartLED 2\app\src\main\res\values\styles.xml

```
<resources>

    <!-- Base application theme. -->
     <style name="AppTheme" parent="android:Theme.Material.Light">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>

    </style>




</resources>

```

```java
package de.me.smartled.ui.notifications;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import de.me.smartled.R;

public class NotificationsFragment extends Fragment {
    View view;

    private NotificationsViewModel notificationsViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        view = inflater.inflate(R.layout.fragment_notifications, container, false);

        return view;
    }
}

```

File - C:\Users\Max\AndroidStudioProjects\v7 2\SmartLED 2\SmartLED 2\app\src\main\res\layout\fragment_notifications.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.
android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.notifications.NotificationsFragment">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:src="@mipmap/layoutabout"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

File - C:\Users\Max\AndroidStudioProjects\v7 2\SmartLED 2\SmartLED 2\app\src\main\res\layout\activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.
android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="?attr/actionBarSize"
    android:background="@mipmap/background" >
    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/nav_view"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="0dp"
        android:layout_marginEnd="0dp"
        android:background="?android:attr/windowBackground"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:menu="@menu/bottom_nav_menu" />

    <fragment
        android:id="@+id/nav_host_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:defaultNavHost="true"
        app:layout_constraintBottom_toTopOf="@id/nav_view"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0"
        app:navGraph="@navigation/mobile_navigation" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

File - C:\Users\Max\AndroidStudioProjects\v7 2\SmartLED 2\SmartLED 2\app\src\main\AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.me.smartled">
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/icon"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MaterialComponents" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

File - C:\Users\Max\AndroidStudioProjects\v7 2\SmartLED 2\SmartLED 2\app\src\main\res\menu\bottom_nav_menu.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/navigation_home"
        android:icon="@drawable/ic_home_black_24dp"
        android:title="@string/title_home" />

    <item
        android:id="@+id/navigation_dashboard"
        android:icon="@drawable/ic_dashboard_black_24dp"
        android:title="@string/title_dashboard" />

    <item
        android:id="@+id/navigation_notifications"
        android:icon="@drawable/baseline_person_24"
        android:title="@string/title_about" />

</menu>

```