Heterogenes Computing - Smart Home Projekt

Bericht

Narek Grigoryan
Matrikelnummer: 1547616

Inhaltsverzeichnis

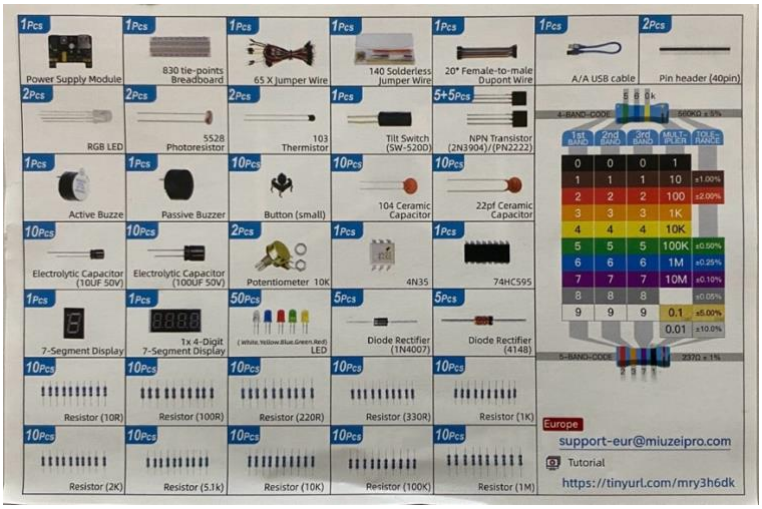
1. Einleitung, Projektidee und Motivation

Zu Beginn entschied ich mich für das Thema Smart Home, da ich bereits mehrere smarte Geräte zu Hause nutzte, wie einen Luftentfeuchter, einen Roboterstaubsauger und eine LED-Lampe. Jedes dieser Geräte erforderte eine eigene App zur Steuerung, was unpraktisch war. Daher kam mir die Idee, alle Smart-Home-Geräte miteinander zu verbinden und über einen Mikrocontroller zu steuern. Hierfür benötigte ich jedoch eine passende Hardware, die diese Funktionalität ermöglicht.
Nach einem Gespräch mit meinem Mentor, Herrn Stumm, machte er mich auf den **Arduino Nano Matter** aufmerksam, der ideal für mein Vorhaben erschien.

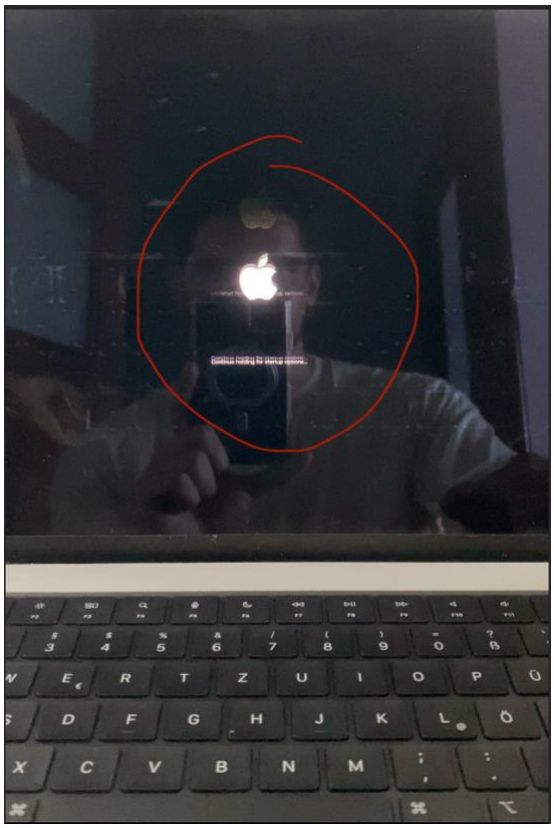
2. Entwicklungsprozess

2.1 Erste Phase des Projekts (Spanien)

Im Juli 2024 reiste ich aus familiären Gründen nach Spanien, was dazu führte, dass ich mein ursprünglich geplantes Projekt pausieren musste. Vor Ort wollte ich jedoch weiterhin etwas Nützliches entwickeln – insbesondere für meinen Großvater, der altersbedingt unter Sehschwächen leidet und Schwierigkeiten hat, im Dunkeln den Lichtschalter zu finden. Er lebt allein und verwendet bereits einen Amazon Echo Dot in seiner Wohnung, mit dem er sich sehr wohlfühlt. Für ihn ist Alexa eine Art KI-Freundin, die ihm im Alltag hilft.
Diese Situation brachte mich auf die Idee, ein smartes Lichtsteuerungssystem für meinen Großvater zu entwickeln, das er einfach per Sprachsteuerung bedienen kann.
Ich bestellte daraufhin einen Arduino Nano Matter, dessen Lieferung aus Italien etwa 5–6 Tage dauerte. Als der Arduino ankam, stellte ich jedoch fest, dass ich mit dem Arduino allein nicht weit kommen würde, da mir weitere Bauteile fehlten und ich kaum Erfahrung mit Mikrocontrollern hatte. Daher entschied ich mich, ein **Elektronik-Fun-Kit von Amazon** (Bild unten) zu kaufen, um die notwendigen Hardware-Komponenten zu erhalten und mich intensiver mit der Technik auseinanderzusetzen.



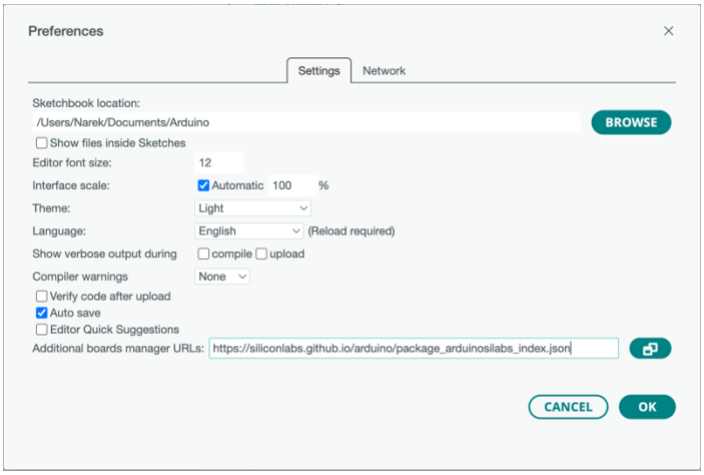
Kaum hatte ich angefangen mit dem Projekt, begann mein **M1 MacBook** mir Probleme zu bereiten.

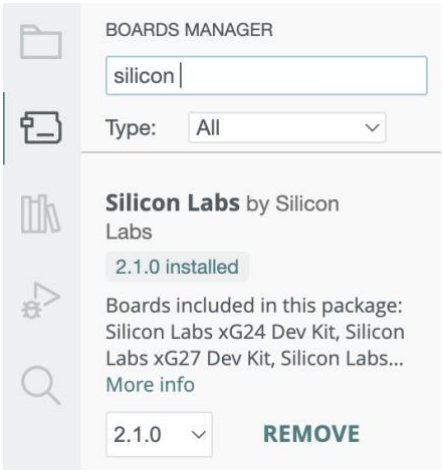


Der Bildschirm flackerte zunächst, und nach mehreren Neustarts stellte ich fest, dass der Bildschirm nach etwa 30 Sekunden Betrieb schwarz wurde. Dies machte es schließlich unmöglich, das MacBook weiter zu benutzen, da ich nichts mehr sehen konnte.
Zum Glück stellte mir mein Arbeitgeber ein **M3 MacBook** zur Verfügung, das ich jedoch persönlich in Deutschland abholen musste. Deshalb musste ich nach Deutschland zurückreisen, um es zu erhalten.
Wieder in Deutschland überlegte ich, ob ich mein ursprüngliches Projekt fortsetzen sollte. Schließlich besaß ich bereits eine Vielzahl an **smarten Geräten** die ich zentral zu steuern möchte. Dies würde mir ermöglichen, mein Zuhause noch smarter und effizienter zu gestalten.

2.2 Einrichtung der DIE und Einstiegsprojekt

Um erste Einblicke in die Welt der Mikrocontroller und des Arduino Nano Matter zu gewinnen, habe ich die folgende Dokumentation durchgesehen: Arduino Nano Matter Benutzerhandbuch. *Zum Einstieg wollte ich mit meinem Arduino Nano Matter eine einfache LED-Lampe zum Leuchten bringen.* Dafür musste ich zunächst die Arduino IDE von dieser Seite herunterladen und auf meinem Mac installieren. Zusätzlich musste in den Einstellungen der IDE der Pfad https://siliconlabs.github.io/arduino/package_arduinosilabs_index.json angegeben werden, damit ich die Silicon-Library herunterladen und installieren konnte.





Nach der Einrichtung des Boards, der Ports und der Bibliotheken habe ich mich mit der benötigten Hardware beschäftigt, um mein Einstiegsprojekt zu realisieren. Nach etwas Recherche stellte ich fest, dass ich folgende Komponenten brauchte:

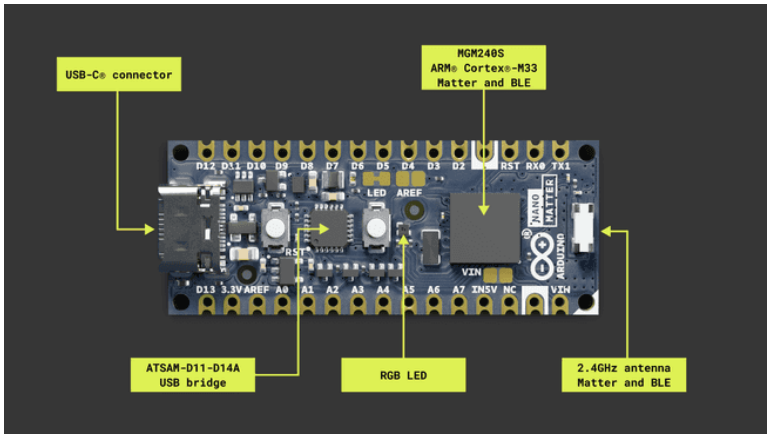
- Arduino Nano Matter
- Steckbrett
- LED-Lampe
- 330-Ohm-Widerstand
- Jumper-Kabel

Da ich nicht genau wusste, wofür die einzelnen Bauteile gedacht waren, musste ich mich intensiver mit ihnen auseinandersetzen.

Hardware-Komponenten:

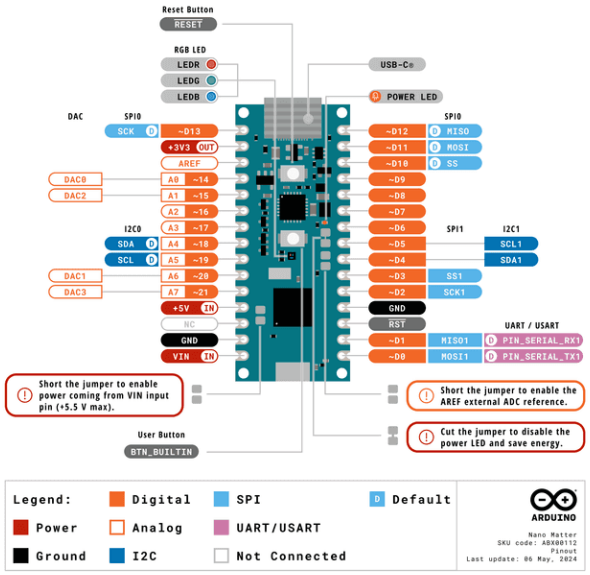
Arduino Nano Matter:

- **Technische Übersicht:**
  Das Nano Matter kombiniert den Arduino-Ansatz mit dem MGM240S von Silicon Labs in einem kompakten Format. Es unterstützt 802.15.4 (Thread®) und Bluetooth® Low Energy zur Interaktion mit Matter-kompatiblen Geräten.



Board-Architektur:

- **Mikrocontroller:** MGM240S mit 32-Bit Arm® Cortex®-M33, 78 MHz, optimiert für IoT-Geräte.
- **Konnektivität:** Multi-Protokoll-Unterstützung für Matter IoT und Bluetooth® LE zur Integration in Smart-Home-Systeme.



Steckbrett:

Ein Steckbrett ist ein Werkzeug, mit dem elektronische Schaltungen aufgebaut werden können, ohne dass gelötet werden muss. Mein 830-Tie-Point-Breadboard hat 830 Anschlusspunkte und besteht aus zwei Hauptbereichen:

1. **Zentrale Arbeitsbereiche:** Diese Bereiche haben horizontal verbundene Reihen von jeweils fünf Löchern, in denen Bauteile wie LEDs und Widerstände platziert werden.
2. **Power Rails:** Oben und unten befinden sich vertikale Versorgungsleisten, die für Plus- und Minuspol der Stromversorgung zuständig sind.

LED:

Eine LED hat zwei Anschlüsse: die Anode (positiver Pol) und die Kathode (negativer Pol). Die Anode ist länger und wird mit dem Pluspol der Stromquelle verbunden, während die Kathode an den Minuspol angeschlossen wird.

Widerstand:

Ein Widerstand begrenzt den Stromfluss zur LED, um ein Durchbrennen zu verhindern.

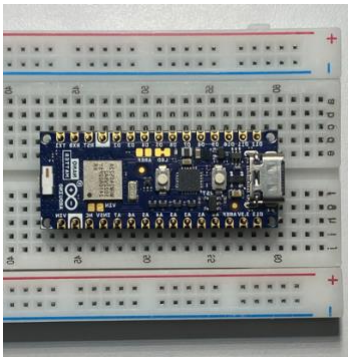
Jumper-Kabel:
Flexible Leitungen, die verwendet werden, um Verbindungen zwischen verschiedenen Punkten auf dem Steckbrett oder zu anderen Komponenten wie Mikrocontrollern herzustellen.

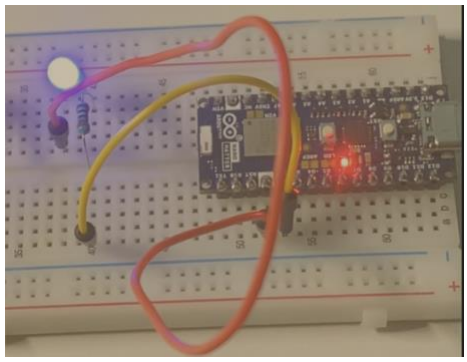
Mit diesem theoretischen Wissen war ich bereit, mein Einstiegsprojekt umzusetzen. Also, was habe ich konkret gemacht?

Aufbau der Schaltung:

1. Pin D2 des Arduino Nano Matter wurde mit einem gelben Jumper-Kabel an eine Reihe des Steckbretts angeschlossen, um die LED zu steuern.
2. Die Anode der LED wurde in dieselbe Reihe wie das Kabel von Pin D2 gesteckt.
3. Ein 330-Ohm-Widerstand wurde zwischen der LED und D2 geschaltet, um den Strom zu begrenzen.
4. GND (Masse) wurde mit einem roten Jumper-Kabel an die Kathode des LED's angeschlossen.

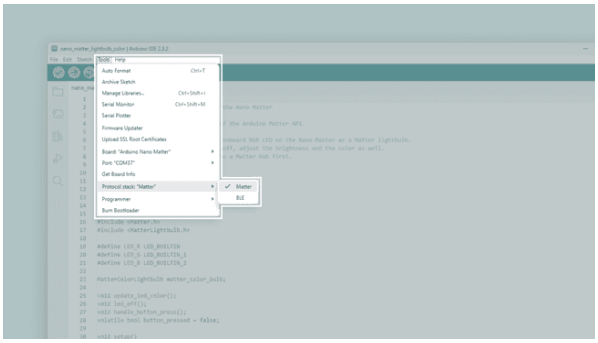
*Anfangs war mein Arduino **nicht gelötet** (Bild 1), was dazu führte, dass die LED nur **sporadisch leuchtete**, je nachdem, wie gut die Kontakte saßen. Daher habe ich eine **Lötmaschine gekauft und die Verbindungen gelötet.** Diese Erfahrung war sehr lehrreich, und nach dem Löten funktionierte die Schaltung einwandfrei.*

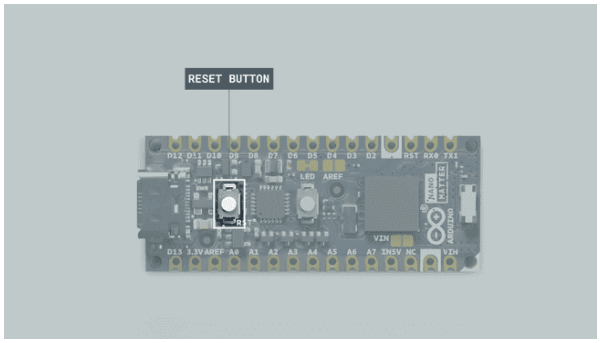




```
1   void setup() {
2     pinMode(2, OUTPUT);
3
4   }
5
6   void loop() {
7     digitalWrite(2),HIGH);
8     delay(100);
9     digitalWrite(2,LOW);
10    delay(100);
11
12  }
13
```

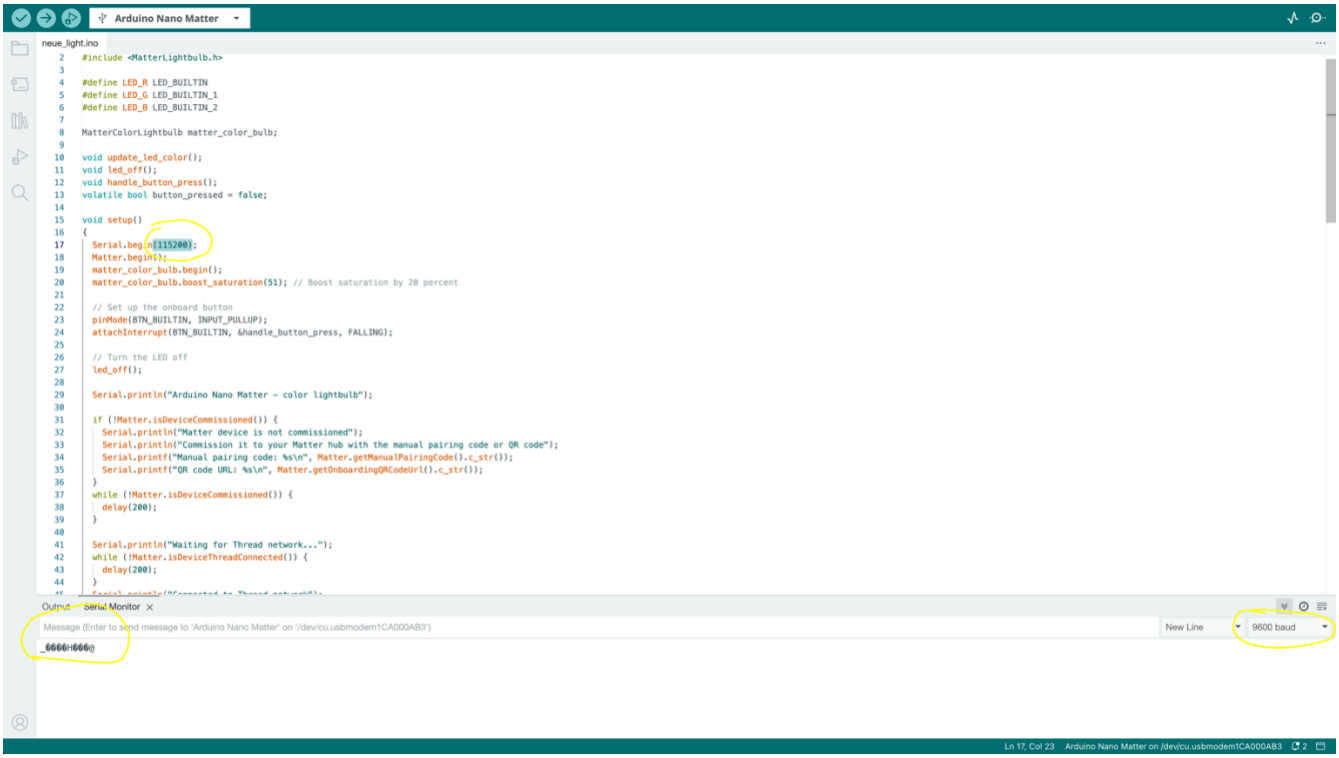
Nachdem ich die LEDs erfolgreich zum Leuchten gebracht hatte, folgte ich der offiziellen Arduino-Dokumentation weiter, um zu erfahren, wie ich mein Arduino als Matter-Gerät einrichten kann. Dazu nutzte ich das Beispiel **matter_lightbulb** aus der Dokumentation und führte es in meiner IDE aus. Zunächst musste ich das Matter-Protokoll aktivieren, was über **Tools > Protocol stack > Matter** möglich war. Nachdem das Arduino entsprechend konfiguriert war, habe ich den Code auf das Board hochgeladen.

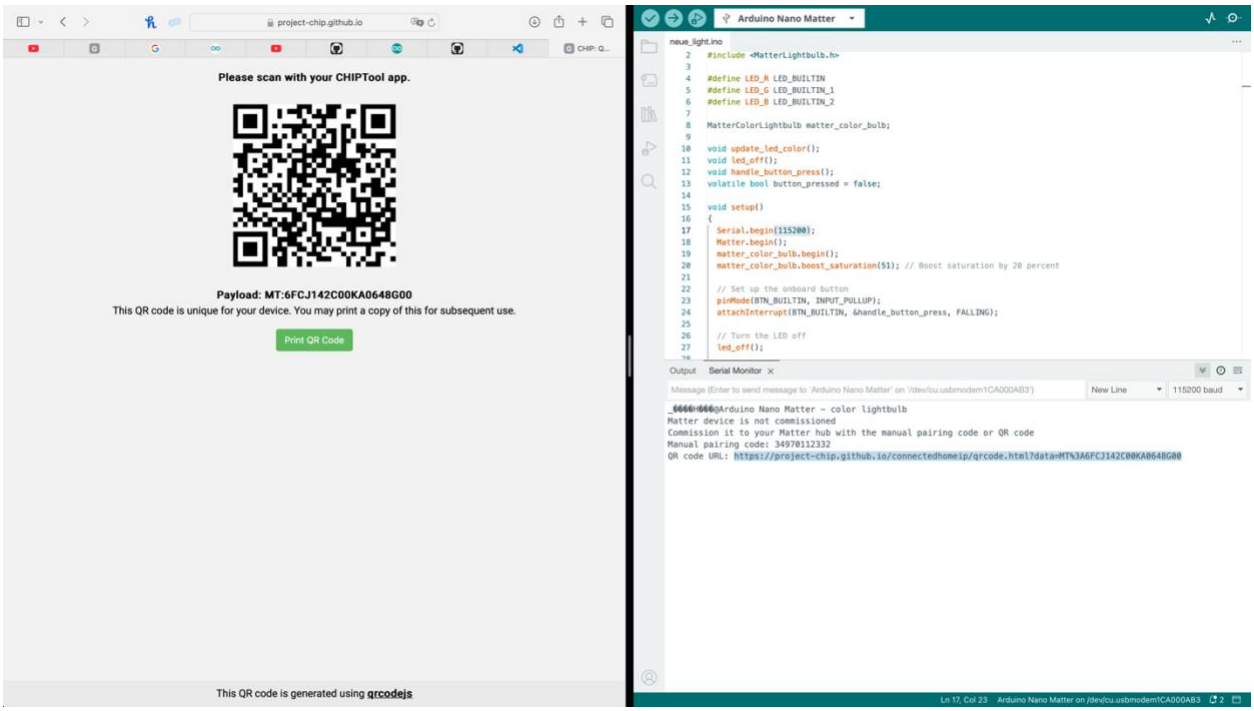




Laut der Dokumentation sollte ich nach dem Hochladen des Codes den **Serial Monitor** in der Arduino IDE öffnen und das Board durch Drücken des Reset-Knopfes zurücksetzen. Um das Matter-Gerät mit dem Netzwerk zu verbinden, benötigte ich die im Terminal angezeigten Zugangsdaten (die werden nach Reset angezeigt), einschließlich eines manuellen Pairing-Codes und eines QR-Code-Links.

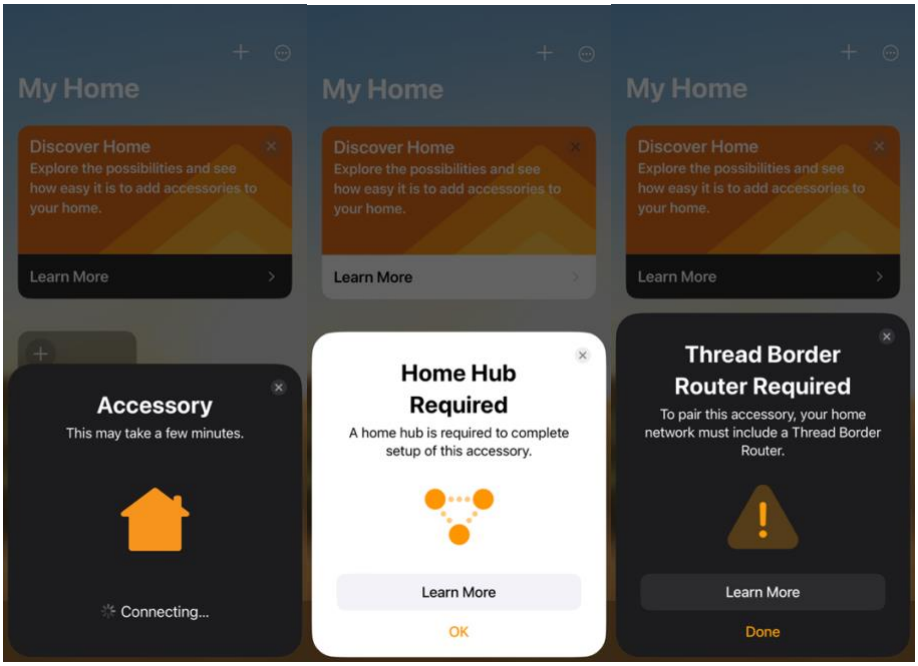
Allerdings wurde mir kein Pairing-Code angezeigt, da der Beispielcode **Serial.begin(115200)** verwendete, während meine IDE auf eine Baudrate von **9600** eingestellt war. Aus diesem Grund erhielt ich keine korrekten Ausgaben im Serial Monitor. Ich habe lange nach dem Fehler gesucht und schließlich herausgefunden, dass es an der falschen Baudrate in den IDE-Einstellungen lag.





3. Implementierung und Probleme (MacBook und Matter)

Nachdem der QR-Code endlich angezeigt wurde, habe ich ihn mit meinem iPhone in der HomeKit-App eingescannt, um den Arduino Nano Matter hinzuzufügen. Dabei erhielt ich jedoch die **Fehlermeldung, dass ein Home Hub oder ein Thread Border Router benötigt wird.**



Diese Nachricht überraschte mich, da ich zu Hause bereits einen Echo Dot verwende und damit über WLAN meine Lampe steuern konnte.

Daraufhin begann ich zu recherchieren und stellte fest, dass **die Funktionsweise von Matter und Thread sich deutlich von der herkömmlicher Smart-Home-Geräte unterscheidet**. Aber was genau steckt hinter Matter und Thread?

Matter wurde entwickelt, um einen einheitlichen Standard für Smart-Home-Geräte zu schaffen. Früher war ein großes Problem in Smart Homes die mangelnde Interaktion zwischen verschiedenen Plattformen und Herstellern. Geräte unterschiedlicher Marken konnten nicht miteinander kommunizieren, was bedeutete, dass Nutzer mehrere Apps oder verschiedene Hubs zur Steuerung benötigten. Matter wurde von der Connectivity Standards Alliance (CSA),ins Leben gerufen, um dieses Problem zu lösen. **Der zentrale Gedanke hinter Matter ist, einen Standard zu schaffen, der es Geräten ermöglicht, unabhängig vom Hersteller oder der Plattform problemlos zusammenzuarbeiten.** Mit Matter können Geräte über Plattformen wie Apple HomeKit, Google Home und Amazon Alexa gesteuert werden, ohne Kompatibilitätsprobleme. Matter unterstützt verschiedene Verbindungen wie WLAN und Ethernet, was die Nutzung bestehender Netzwerke ermöglicht und die Einrichtung erleichtert.

Thread hingegen ist ein **Funkprotokoll,** das speziell für Smart-Home-Geräte entwickelt wurde. Es nutzt ein **Mesh-Netzwerk**, in dem jedes Gerät nicht nur Daten empfängt, sondern auch Signale weiterleitet. Dadurch wird die Reichweite und Stabilität des Netzwerks deutlich verbessert, **da jedes Gerät das Signal an das nächste weitergibt**. Im Gegensatz zu traditionellen WLAN-Netzwerken oder Zigbee, die auf zentrale Hubs angewiesen sind, funktioniert Thread dezentral. Wenn ein Gerät ausfällt, kann ein anderes Gerät die Signalweiterleitung übernehmen, was das Netzwerk sehr robust macht. Ein weiterer wichtiger Vorteil von Thread ist die Energieeffizienz. Es ist für einen geringen Energieverbrauch optimiert und eignet sich daher gut für batteriebetriebene Geräte wie Sensoren oder Türschlösser. Thread zielt darauf ab, zuverlässige und sichere Verbindungen zu schaffen, ohne den Energieverbrauch zu erhöhen.

Da ich bereits WLAN-basierte Smart-Home-Geräte verwendet habe, war ich überrascht zu erfahren, dass diese Technologien eine andere Struktur verwenden. Im Gegensatz zu herkömmlichen Smart-Home-Geräten, die über WLAN oder Bluetooth verbunden sind, setzt Thread auf dieses spezialisierte Mesh-Netzwerk, das einen Thread Border Router benötigt – ein Gerät, das als Schnittstelle zwischen dem Thread-Netzwerk und dem Internet fungiert. Das erklärt, warum mein Echo Dot, der nur WLAN unterstützt, für die Verwendung mit Thread nicht ausreichte.

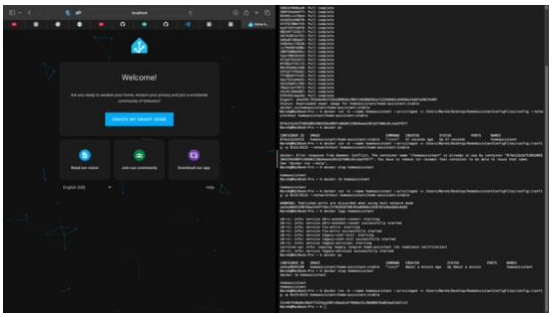
Aus diesem Grund versuchte ich, den eingebauten LED-Schalter des Arduino über **Bluetooth Low Energy (BLE)** zu steuern.

Bluetooth.ino

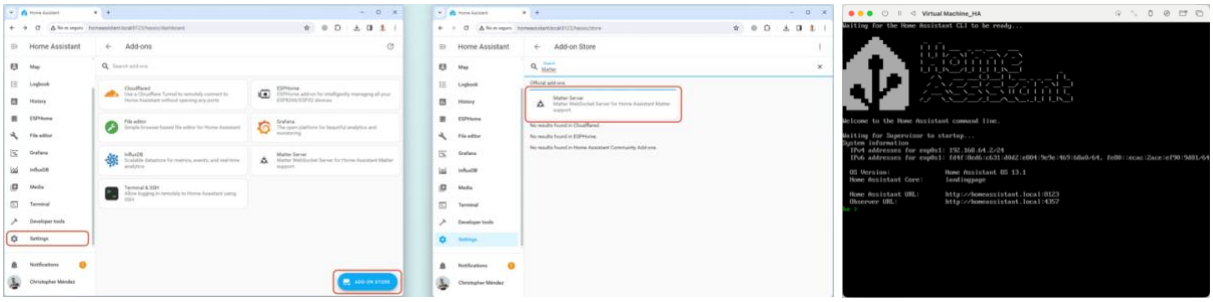
```ino
#include <ArduinoBLE.h>

BLEService ledService("19B10000-E8F2-537E-4F6C-D104768A1214");
BLEByteCharacteristic ledCharacteristic("19B10001-E8F2-537E-4F6C-D104768A1214", BLERead | BLEWrite);

const int ledPin = LED_BUILTIN;

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);

  if (!BLE.begin()) {
    Serial.println("Starting BLE failed!");
    while (1);
  }

  BLE.setLocalName("NanoBLE_LED");
  BLE.setAdvertisedService(ledService);
  ledService.addCharacteristic(ledCharacteristic);
  BLE.addService(ledService);

  ledCharacteristic.writeValue(0);
  BLE.advertise();

  Serial.println("Bluetooth device active, waiting for connections...");
}

void loop() {
  BLEDevice central = BLE.central();

  if (central) {
    Serial.print("Connected to central: ");
    Serial.println(central.address());

    while (central.connected()) {
      if (ledCharacteristic.written()) {
        int ledState = ledCharacteristic.value();
        digitalWrite(ledPin, ledState);
      }
    }

    Serial.print("Disconnected from central: ");
    Serial.println(central.address());
  }
}
```

Der Code definiert dabei einen BLE-Service, der es ermöglicht, ein zentrales Gerät (wie mein iPhone) mit dem Arduino zu verbinden. Dieser Service enthält eine **Charakteristik**, die es erlaubt, den Status der eingebauten LED (ein oder aus) zu steuern.
In der **Setup-Phase** wird der BLE-Service initialisiert, der Arduino bewirbt sich als Bluetooth-Gerät, und die LED wird zunächst ausgeschaltet. Die Hauptlogik befindet sich in der **Loop-Phase**, wo der Arduino auf eine Verbindung von einem zentralen Gerät wartet. Sobald die Verbindung hergestellt ist, überwacht der Arduino, ob der Wert der LED-Charakteristik geändert wird und steuert die LED entsprechend. Es gelang mir, das Licht ein- und auszuschalten, indem ich über mein iPhone die Werte 0 und 1 an den Arduino sendete.

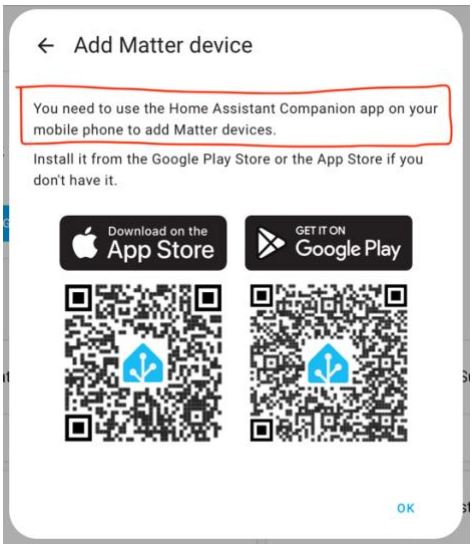
Trotz dieses Erfolgs war ich frustriert, dass es mir nicht gelungen war, die Matter-Funktionalität zu nutzen. Also begann ich mir Gedanken darüber zu machen, wie ich den Arduino mit Matter verbinden könnte. Da mein Echo Dot zu alt war und ich keine anderen kompatiblen Geräte zur Verfügung hatte, musste ich zunächst alles über mein MacBook ausprobieren.
In diesem Zuge entschied ich mich, den **Home Assistant**-Server über Docker auf meinem MacBook einzurichten. Zunächst funktionierte das auch einwandfrei, aber ich bemerkte schnell, dass der **Add-ons Store** im Home Assistant fehlte, der notwendig ist, um Matter zu integrieren. Daher entschied ich mich, Matter direkt über Docker herunterzuladen. Allerdings war das keine ideale Lösung, da das Docker-Image etwa 14 GB groß war.

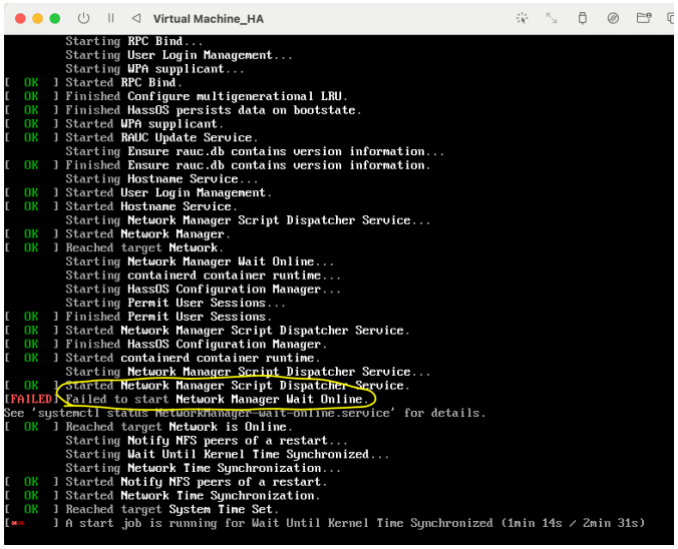


Leider gelang es mir nicht, Matter mit Home Assistant zu verbinden. Also forschte ich weiter und fand heraus, dass ich Home Assistant auf einer **virtuellen Maschine** laufen lassen musste, um Zugriff auf den Add-ons Store zu erhalten. Mit dieser Version von Home Assistant hatte ich dann den Add-ons Store zur Verfügung und konnte Matter schließlich erfolgreich integrieren.



Nun stand ich vor dem Problem, dass die **Home Assistant**-URL nur auf meinem MacBook zugänglich war und nicht auf meinem iPhone. Da der Arduino Nano Matter einen QR-Code generierte, mit dem man sich verbinden kann, benötigte ich jedoch Zugriff auf Home Assistant über mein iPhone. Ich versuchte, die Netzwerkeinstellungen der virtuellen Maschine auf **Bridged Mode** zu setzen, damit mein iPhone Zugriff auf die VM hat. Allerdings funktionierte Home Assistant dann nicht mehr.



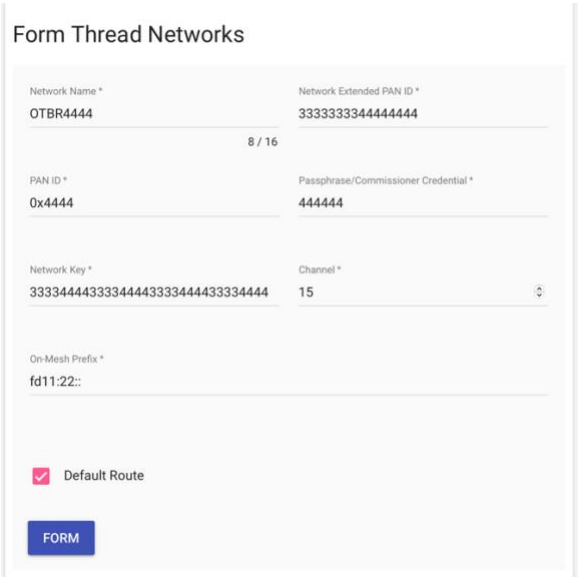


Daraufhin wandte ich mich erneut an meinen Professor, Herrn Stumm, der mir den Tipp gab, die Firewall-Einstellungen meines Macs zu überprüfen. Letztendlich stellte sich heraus, dass das Problem genau dort lag: Die **UTM-App**, die ich zur Erzeugung der virtuellen Maschine nutzte, war nicht in den Firewall-Ausnahmen meines MacBook aufgelistet. Nachdem ich UTM zur Liste der erlaubten Programme hinzugefügt hatte, konnte ich endlich den Home Assistant-Server auf meinem iPhone öffnen und den **Pairing-QR-Code** scannen.

Nachdem ich Home Assistant erfolgreich auf meinem MacBook eingerichtet hatte, benötigte ich nun einen **Thread Border Router**, um die volle Funktionalität nutzen zu können. Mein Professor gab mir den Tipp, die **Open Source-Version eines Thread Border Routers (OTBR)** auf meinem Notebook zu installieren, um die Funktionen eines Hubs zu übernehmen. Also entschied ich mich, OTBR lokal auf meinem Mac mithilfe von Docker zum Laufen zu bringen.

Zunächst lud ich das OTBR Docker-Image herunter und startete den Container. Nachdem ich die laufenden Container überprüft hatte, versuchte ich, über **localhost:8080** auf die OTBR-Weboberfläche zuzugreifen, aber das funktionierte nicht. Der Localhost konnte nicht richtig gestartet werden, da meine simulierte RCP (Radio Co-Processor) nicht erkannt wurde. Das Problem lag an den typischen Mac-Berechtigungsproblemen.

```
Narek@MacBook-Pro ~ % ls -l /dev/ttys*
crw--w----  1 root   wheel   0x4000030 Sep 22 16:55 /dev/ttys0
crw--w----  1 Narek  tty     0x10000000 Sep 23 13:27 /dev/ttys000
crw--w----  1 Narek  tty     0x10000001 Sep 23 13:27 /dev/ttys001
crw--w----  1 Narek  tty     0x10000002 Sep 23 13:16 /dev/ttys002
crw--w----  1 Narek  tty     0x10000003 Sep 23 13:27 /dev/ttys003
crw--w----  1 Narek  tty     0x10000004 Sep 23 13:23 /dev/ttys004
crw--w----  1 Narek  tty     0x10000005 Sep 23 13:28 /dev/ttys005
crw-rw-rw-  1 root   wheel   0x4000031 Sep 22 16:55 /dev/ttys1
crw-rw-rw-  1 root   wheel   0x4000032 Sep 22 16:55 /dev/ttys2
crw-rw-rw-  1 root   wheel   0x4000033 Sep 22 16:55 /dev/ttys3
crw-rw-rw-  1 root   wheel   0x4000034 Sep 22 16:55 /dev/ttys4
crw-rw-rw-  1 root   wheel   0x4000035 Sep 22 16:55 /dev/ttys5
crw-rw-rw-  1 root   wheel   0x4000036 Sep 22 16:55 /dev/ttys6
crw-rw-rw-  1 root   wheel   0x4000037 Sep 22 16:55 /dev/ttys7
crw-rw-rw-  1 root   wheel   0x4000038 Sep 22 16:55 /dev/ttys8
crw-rw-rw-  1 root   wheel   0x4000039 Sep 22 16:55 /dev/ttys9
crw-rw-rw-  1 root   wheel   0x400003a Sep 22 16:55 /dev/ttysa
crw-rw-rw-  1 root   wheel   0x400003b Sep 22 16:55 /dev/ttysb
crw-rw-rw-  1 root   wheel   0x400003c Sep 22 16:55 /dev/ttysc
crw-rw-rw-  1 root   wheel   0x400003d Sep 22 16:55 /dev/ttysd
crw-rw-rw-  1 root   wheel   0x400003e Sep 22 16:55 /dev/ttyse
crw-rw-rw-  1 root   wheel   0x400003f Sep 22 16:55 /dev/ttysf
Narek@MacBook-Pro ~ % sudo chmod 666 /dev/ttys001
sudo chmod 666 /dev/ttys002

Password:
Narek@MacBook-Pro ~ % ls -l /dev/ttys*
crw-rw-rw-  1 root   wheel   0x4000030 Sep 22 16:55 /dev/ttys0
crw--w----  1 Narek  tty     0x10000000 Sep 23 13:27 /dev/ttys000
crw-rw-rw-  1 Narek  tty     0x10000001 Sep 23 13:27 /dev/ttys001
crw-rw-rw-  1 Narek  tty     0x10000002 Sep 23 13:16 /dev/ttys002
crw--w----  1 Narek  tty     0x10000003 Sep 23 13:27 /dev/ttys003
crw--w----  1 Narek  tty     0x10000004 Sep 23 13:23 /dev/ttys004
crw--w----  1 Narek  tty     0x10000005 Sep 23 13:30 /dev/ttys005
crw-rw-rw-  1 root   wheel   0x4000031 Sep 22 16:55 /dev/ttys1
crw-rw-rw-  1 root   wheel   0x4000032 Sep 22 16:55 /dev/ttys2
crw-rw-rw-  1 root   wheel   0x4000033 Sep 22 16:55 /dev/ttys3
crw-rw-rw-  1 root   wheel   0x4000034 Sep 22 16:55 /dev/ttys4
crw-rw-rw-  1 root   wheel   0x4000035 Sep 22 16:55 /dev/ttys5
crw-rw-rw-  1 root   wheel   0x4000036 Sep 22 16:55 /dev/ttys6
crw-rw-rw-  1 root   wheel   0x4000037 Sep 22 16:55 /dev/ttys7
```



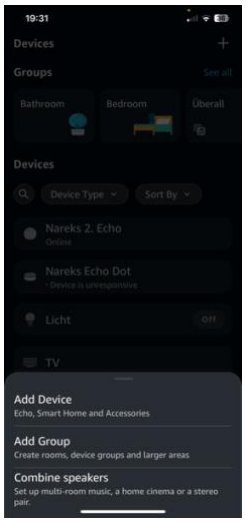
Also verbrachte ich einige Zeit im Terminal, um die Zugriffsrechte anzupassen. Nachdem ich diese Hürde überwunden hatte, lief OTBR endlich auf **localhost**. Allerdings stieß ich auf weitere Schwierigkeiten, insbesondere beim erstellen des **Thread-Netzwerks**. Auch nach stundenlangem Herumprobieren erhielt ich keine brauchbaren Ergebnisse, was den Fortschritt des Projekts erheblich verlangsamte. Der zunehmende Zeitdruck machte die Situation zusätzlich kompliziert, da nur noch wenig Zeit bis zur Abgabefrist des Projekts blieb.
In meiner Verzweiflung entschied ich mich, die Dokumentation von Arduino erneut durchzusehen. Schließlich beschloss ich, eine pragmatische Lösung zu wählen und kaufte mir einen **Amazon Echo 4**, der als Thread Border Router fungiert und gleichzeitig Matter-Integration bietet.

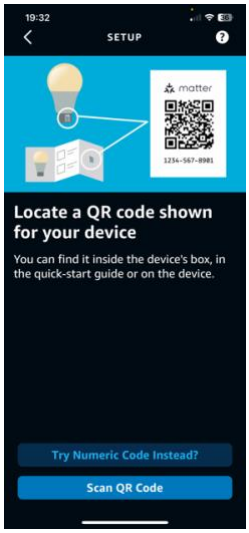
4. RGB-LED Steuerung, Matter-Integration und den Amazon Echo

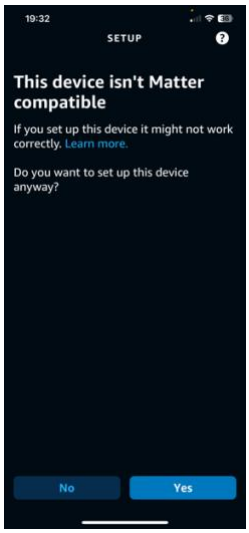
Nachdem ich den **Amazon Echo 4** eingerichtet hatte, überprüfte ich meine vorhandenen Smart-Geräte zu Hause und stellte fest, dass keines von ihnen **Matter**-Unterstützung hatte, da sie bereits veraltet waren. Daraufhin kam mir mein ursprünglicher Plan in den Sinn, eine LED-Steuerung umzusetzen. Also entschied ich mich, einen **LED-Strip** zu kaufen, ihn mit meinem Arduino zu verbinden und ihn anschließend über Alexa zu steuern.

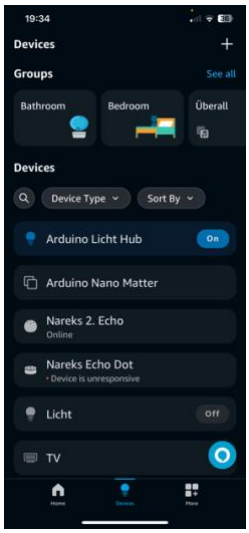
Nach dem Kauf des LED-Strips begann ich mit der Verbindung zum Arduino Nano Matter. Als Einstieg in die Welt von Matter nutzte ich das Beispielprojekt **nano_matter_lightbulb_color** aus der offiziellen Arduino-Dokumentation (über **File > Examples > Matter > nano_matter_lightbulb_color**). Mit diesem Beispiel begann ich, meinen **Arduino Nano Matter** einzurichten, um ihn mit der Alexa-App zu steuern.

Der Einrichtungsprozess war ähnlich wie zuvor: Ich lud den Matter-Code hoch, öffnete den Serial Monitor in der Arduino IDE und drückte den Reset-Knopf, um den Arduino Nano Matter in den Pairing-Modus zu versetzen. Die genaue Schritt-für-Schritt-Einrichtung ist auf den folgenden Bildern dargestellt.

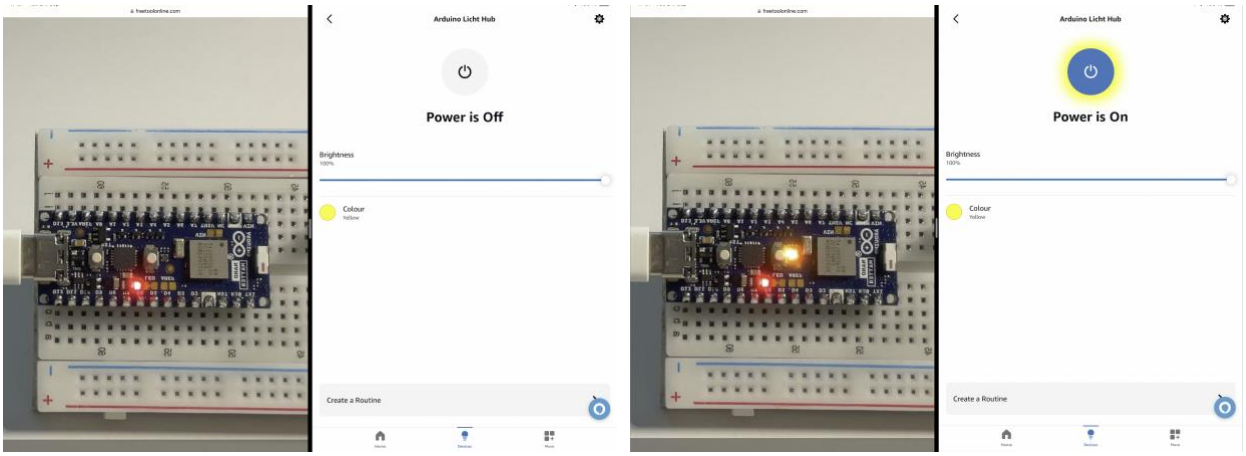








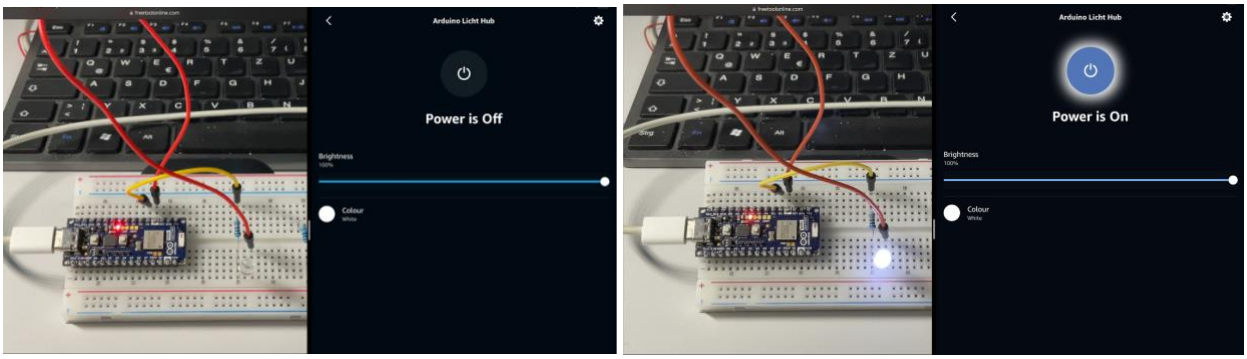
Nach der erfolgreichen Einrichtung des Arduino Nano Matter konnte ich mithilfe des Beispielcodes die eingebaute LED über die Alexa-App steuern.



Dies motivierte mich, das Projekt mit einem **WS2812B LED-Strip** weiterzuführen. Doch als ich versuchte, den LED-Strip ebenfalls mit dem Arduino zu verbinden und zu steuern, stellte ich bald fest, dass die Nano Matter-Architektur sich deutlich von den üblichen Atmel/Microchip-Chips wie dem ATmega328 unterscheidet. Daher funktionierten die bekannten Bibliotheken wie FastLED oder Adafruit NeoPixel nicht.

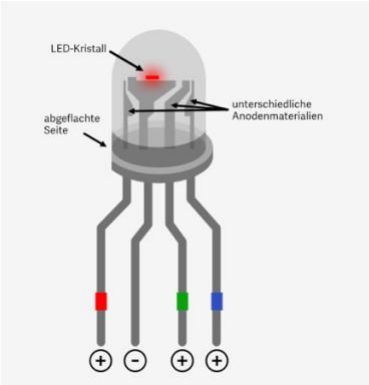
Nach längerer Recherche stieß ich auf die Bibliothek ezWS2812, die von Silicon Labs entwickelt wurde. Meine anfängliche Freude war jedoch kurz, als ich entdeckte, dass ich einen Level Shifter benötigte, um das Projekt korrekt umzusetzen. Da mein Fun Kit keinen Level Shifter enthielt, musste ich einen neuen bestellen, der jedoch erst nach einigen Tagen eintreffen würde. Angesichts der knappen Zeit bis zur Deadline entschied ich mich, stattdessen die LED-Lichter aus meinem Toolkit zu verwenden.

Zunächst nutzte ich meine bereits vorhandene Schaltung und erweiterte den Code für die Matter-Integration. Damit konnte ich über Alexa das bereits eingebaute Licht steuern, was problemlos funktionierte.



Allerdings war ich mit diesem Teilerfolg nicht zufrieden und beschloss, die Schaltung auf **RGB-LEDs** umzubauen, um die Steuerung weiter auszubauen und vielseitiger zu gestalten.

Für mein Projekt habe ich eine RGB-LED auf einem Breadboard aufgebaut, um sie mithilfe eines Arduino Nano Matter zu steuern. Die RGB-LED verfügt über vier Pins, wobei der erste Pin für Rot (R), der zweite für GND, der dritte für Grün (G) und der vierte für Blau (B) vorgesehen ist.

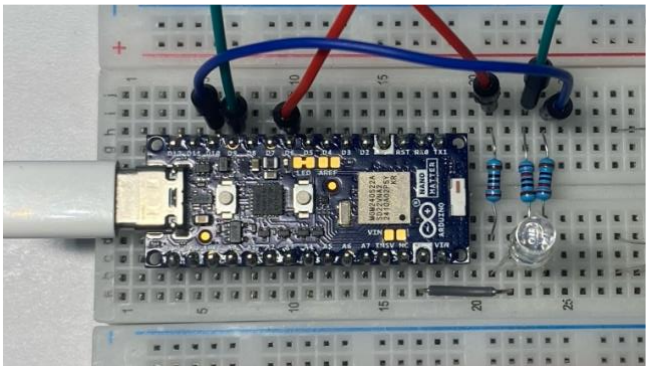


Zuerst habe ich den GND-Pin der RGB-LED mit dem GND-Pin des Arduino verbunden, um eine gemeinsame Masse für alle Farbkanäle sicherzustellen. Anschließend habe ich die Rot-, Grün- und Blau-Pins der LED jeweils über 200 Ohm Widerstände mit den Digitalpins D6, D9 und D10 des Arduino Nano Matter verbunden. Diese Widerstände sind notwendig, um den Strom zu begrenzen und die LED vor Überlastung zu schützen.

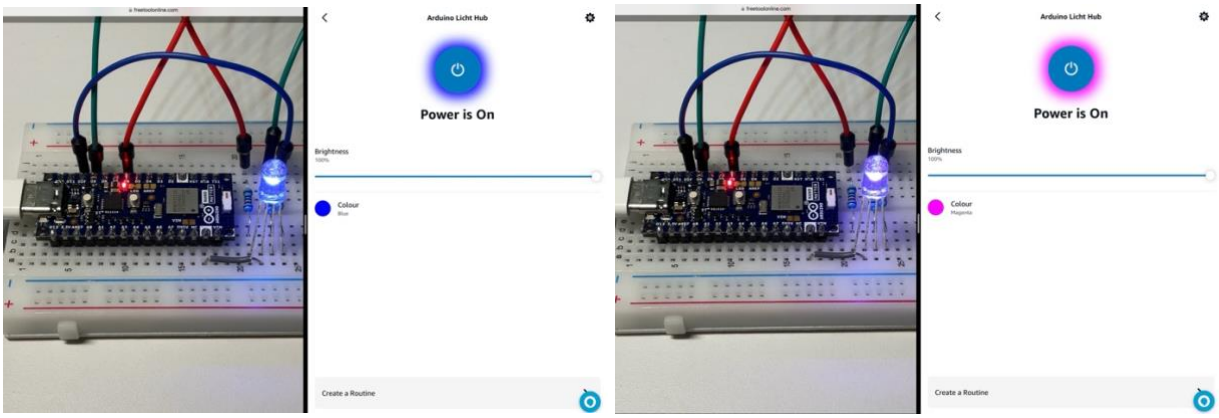
Im Detail bedeutet das:

- Der Rot-Pin (R) der RGB-LED ist mit D6 verbunden, wodurch die rote Farbe gesteuert wird.
- Der Grün-Pin (G) ist mit D9 verbunden, um den grünen Farbanteil zu kontrollieren.
- Der Blau-Pin (B) ist mit D10 verbunden, um den blauen Anteil zu steuern.

Die gesamte Schaltung wurde auf einem Breadboard realisiert, wobei ich für die Verbindung zwischen dem Arduino und den entsprechenden Pins der RGB-LED Jumper-Kabel verwendet habe. Die Spannungsversorgung und Programmierung des Arduino erfolgte über ein USB-C Kabel.



Um die RGB-LED zu steuern, habe ich die Pins D10, D9 und D6 über PWM (Pulsweitenmodulation) angesteuert. Dies ermöglicht mir, die Intensität der roten, grünen und blauen Farben unabhängig voneinander zu regulieren und dadurch verschiedene Farbmischungen zu erzeugen. Mit dieser Ansteuerung lässt sich die LED in einer Vielzahl von Farben darstellen, indem die Werte für Rot, Grün und Blau entsprechend angepasst werden.



5. Der Code

Der vorliegende Abschnitt beschreibt detailliert die Funktionsweise des Codes, der mithilfe des Arduino Nano Matter und einer RGB-LED eine Verbindung zu einem Matter-kompatiblen Smart-Home-System herstellt. Der Code ermöglicht es, Farben sowie den Betriebsstatus der LED direkt von der Matter-Plattform aus zu steuern. Zusätzlich wird ein physischer Button integriert, um das Licht manuell zu toggeln (Ein-/Ausschalten).

Detaillierte Funktionsweise des Codes:

Einbindung der Bibliotheken und Definition der Pins

Der erste Abschnitt des Codes bindet zwei Bibliotheken ein: Matter.h und MatterLightbulb.h. Diese Bibliotheken sind notwendig, um die Matter-Funktionalität und die Steuerung einer RGB-Glühbirne bereitzustellen. Anschließend werden die Pins für die roten, grünen und blauen LEDs definiert, die über die PWM-Funktion des Arduino angesteuert werden.

```
#include <Matter.h>
#include <MatterLightbulb.h>

#define RED_PIN 6    // PWM-Pin für die rote LED
#define GREEN_PIN 9  // PWM-Pin für die grüne LED
#define BLUE_PIN 10  // PWM-Pin für die blaue LED

MatterColorLightbulb matter_color_bulb;  // Definiert das Matter-Objekt für die Glühbirne
volatile bool button_pressed = false;    // Speichert den Status des Buttons (gedrückt/nicht gedrückt)
```

Initialisierung im setup()-Block

Im setup()-Abschnitt wird die serielle Kommunikation gestartet und die Matter-Umgebung initialisiert. Diese Initialisierung ermöglicht es dem Arduino, mit einem Matter-Hub zu kommunizieren und als steuerbares RGB-Licht im Smart-Home-System erkannt zu werden.

Zudem wird ein eingebauter Button konfiguriert, der als Input mit einem Pull-Up-Widerstand genutzt wird. Ein Interrupt wird auf den Button gesetzt, um auf Knopfdruck zu reagieren und das Licht an- oder auszuschalten. Dies ermöglicht eine manuelle Steuerung der RGB-LED, unabhängig von der Fernsteuerung über den Matter-Hub.

Im selben Abschnitt werden die Pins für die roten, grünen und blauen LEDs als Ausgänge definiert und die RGB-LED zu Beginn ausgeschaltet. Danach prüft der Code, ob das Gerät bereits mit einem Matter-Hub gekoppelt ist (Kommissionierung). Falls nicht, wird ein Pairing-Code und QR-Code zur manuellen Kopplung ausgegeben. Der Abschnitt endet mit einer Schleife, die wartet, bis das Gerät erfolgreich mit dem Matter-Netzwerk verbunden ist.

```
11  void setup() {
12    // Diese Funktion initialisiert die serielle Kommunikation, das Matter-Protokoll
13    // und die RGB-LED-Steuerung. Sie überprüft auch, ob das Gerät bereits mit dem Matter-Hub
14    // gekoppelt ist. Falls nicht, werden Pairing-Informationen bereitgestellt.
15
16    Serial.begin(115200);
17    Matter.begin();
18    matter_color_bulb.begin();
19
20    // Hier wird der eingebaute Button konfiguriert, und ein Interrupt wird eingerichtet,
21    // um auf Knopfdruck zu reagieren.
22    pinMode(BTN_BUILTIN, INPUT_PULLUP);
23    attachInterrupt(digitalPinToInterrupt(BTN_BUILTIN), handle_button_press, FALLING);
24
25    // Die Pins für die roten, grünen und blauen LEDs werden als Ausgang definiert,
26    // und die LEDs werden zunächst ausgeschaltet.
27    pinMode(RED_PIN, OUTPUT);
28    pinMode(GREEN_PIN, OUTPUT);
29    pinMode(BLUE_PIN, OUTPUT);
30    setColor(0, 0, 0);  // Startet mit ausgeschalteten RGB-LEDs (alle Farben aus)
31
32    // Falls das Gerät nicht mit einem Matter-Hub verbunden ist, gibt es die Pairing-Informationen
33    // aus und wartet auf die Kommissionierung und Verbindung.
34    if (!Matter.isDeviceCommissioned()) {
35      Serial.printf("Manueller Pairing-Code: %s\n", Matter.getManualPairingCode().c_str());
36      Serial.printf("QR-Code-URL: %s\n", Matter.getOnboardingQRCodeUrl().c_str());
37      waitForCommissioning();
38    }
39
40    // Wartet darauf, dass das Gerät online und im Matter-Netzwerk verfügbar ist.
41    Serial.println("Warte auf Matter-Geräteerkennung...");
42    while (!matter_color_bulb.is_online()) delay(200);
43    Serial.println("Matter-Gerät ist jetzt online");
44  }
```

Hauptlogik in der loop()-Funktion

In der loop()-Funktion wird kontinuierlich überprüft, ob der Button gedrückt wurde. Falls ja, wird der Status des Lichtes (An/Aus) umgeschaltet. Zudem wird überprüft, ob das Licht gerade eingeschaltet ist. Wenn es an ist, werden die RGB-Werte aus den Matter-Einstellungen abgerufen und die Farbe entsprechend geändert. Falls das Licht aus ist, werden die RGB-LEDs ebenfalls ausgeschaltet.

```
46  void loop() {
47    // Diese Funktion prüft kontinuierlich, ob der Button gedrückt wurde, um das Licht
48    // ein- oder auszuschalten. Außerdem werden die RGB-Werte aus dem Matter-Netzwerk abgerufen,
49    // wenn das Licht eingeschaltet ist, und auf die LEDs angewendet.
50
51    if (button_pressed) {
52      button_pressed = false;
53      matter_color_bulb.toggle();
54    }
55
56    // Überprüft, ob das Licht eingeschaltet ist und wendet entsprechende RGB-Werte an,
57    // oder schaltet die LEDs aus, wenn das Licht aus ist.
58    bool light_state = matter_color_bulb.get_onoff();
59    if (light_state) {
60      setRGBColorFromMatter();
61    } else {
62      setColor(0, 0, 0);
63    }
64  }
65
66  void setRGBColorFromMatter() {
67    // Diese Funktion ruft die aktuellen RGB-Werte aus dem Matter-Netzwerk ab
68    // und skaliert sie, bevor sie an die LEDs gesendet werden, um die Helligkeit zu steuern.
69
70    uint8_t r, g, b;
71    matter_color_bulb.get_rgb(&r, &g, &b);  // Holt die RGB-Werte von Matter
72
73    // Skaliert die Farbwerte, um eine angemessene Helligkeit zu gewährleisten
74    setColor(map(r, 0, 255, 50, 255), map(g, 0, 255, 50, 255), map(b, 0, 255, 50, 255));
75  }
76
77  void setColor(uint8_t red, uint8_t green, uint8_t blue) {
78    // Diese Funktion setzt die Farben der RGB-LED durch Anwendung der PWM-Werte auf die entsprechenden Pins.
79
80    analogWrite(RED_PIN, red);
81    analogWrite(GREEN_PIN, green);
82    analogWrite(BLUE_PIN, blue);
83  }
84
85  void handle_button_press() {
86    // Diese Funktion wird durch den Button-Interrupt ausgelöst und sorgt dafür,
87    // dass die Statusänderung des Buttons (gedrückt/nicht gedrückt) erkannt wird.
88
89    static uint32_t last_press = 0;
90    if (millis() - last_press > 200) {
91      button_pressed = true;
92      last_press = millis();
93    }
94  }
95
96  void waitForCommissioning() {
97    // Diese Funktion sorgt dafür, dass das Gerät wartet, bis es erfolgreich
98    // mit dem Matter-Netzwerk gekoppelt und verbunden wurde.
99
100   while (!Matter.isDeviceCommissioned()) {
101     Serial.println("Matter-Gerät ist nicht eingebunden");
102     delay(200);
103   }
104
105   // Wartet, bis das Gerät mit einem Thread-Netzwerk verbunden ist
106   Serial.println("Warte auf Thread-Netzwerk...");
107   while (!Matter.isDeviceThreadConnected()) delay(200);
108   Serial.println("Mit Thread-Netzwerk verbunden");
109 }
110
```

Diese Hauptschleife sorgt dafür, dass das Licht entweder über den Button oder über den Matter-Hub gesteuert werden kann.

Funktion zur Farbsteuerung

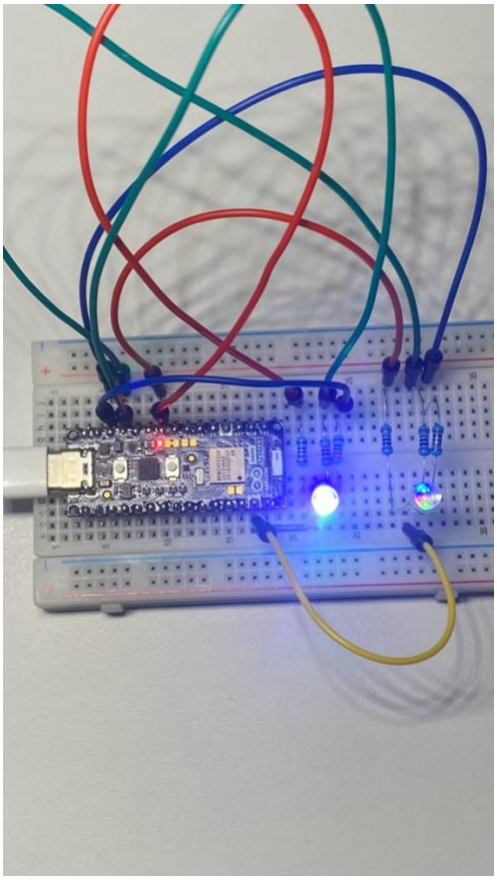
Die Funktion setRGBColorFromMatter() liest die aktuellen RGB-Werte von der Matter-Glühbirne aus und setzt diese Werte auf die RGB-LED um. Dabei werden die Farbwerte skaliert, um sicherzustellen, dass auch bei niedrigen Werten eine ausreichende Helligkeit erreicht wird. Diese Funktion ermöglicht die dynamische Anpassung der LED-Farbe, basierend auf den Einstellungen im Matter-Netzwerk.

Interrupt für den Button
Die Funktion handle_button_press() wird durch einen Interrupt aufgerufen, wenn der Button gedrückt wird. Diese Funktion sorgt dafür, dass der Button-Status erfasst wird, um das Licht an- oder auszuschalten.

Warten auf Kommissionierung und Netzwerkverbindung
Die Funktion waitForCommissioning() wartet, bis das Gerät erfolgreich mit einem Matter-Netzwerk verbunden wurde. Diese Warteschleifen sorgen dafür, dass der Arduino nicht weiterarbeitet, bis die Verbindung zum Netzwerk und dem Matter-Hub hergestellt ist. Das gewährleistet, dass das Gerät erst dann als steuerbare Einheit agiert, wenn es vollständig integriert ist.

Der gesamte Code kann unter diesem GitHub-Link abgerufen werden.

Um eine genauere Analyse zu ermöglichen, habe ich zusätzlich zur ersten RGB-LED eine zweite RGB-LED angeschlossen, wobei ich die gleichen Pins wie bei der ersten verwendet habe. Allerdings habe ich diesmal einen 10K-Ohm-Widerstand eingesetzt, um Unterschiede in der Helligkeit zu beobachten.



Unter diesem Link kann man auch ein Video meines gesamten Projekts sehen, in dem ich die RGB-LEDs erfolgreich über Alexa gesteuert habe.

6. Fazit und Ausblick

Rückblickend war der Weg, dieses Projekt umzusetzen, alles andere als einfach. Aufgrund familiärer Verpflichtungen reiste ich nach Spanien, was meine ursprüngliche Projektplanung unterbrach. Vor Ort entschied ich mich, das Projekt auf ein Smart-Licht für meinen Großvater umzulenken, da er aufgrund seiner Sehschwäche Schwierigkeiten hatte, Lichtschalter zu finden. Während ich an dieser Idee arbeitete, gab mein **MacBook** den Geist auf, was mich zwang, nach Deutschland zurückzureisen, um ein neues **M3 MacBook** von meinem Arbeitgeber zu holen. Dieser unerwartete Rückschlag kostete mich Zeit und bremste meinen Fortschritt.
Trotz dieser Hürden war ich entschlossen, weiterzumachen. Ich machte erste Fortschritte mit **Bluetooth**, indem ich den eingebauten LED-Schalter des Arduino über mein iPhone steuern konnte. Das funktionierte gut, doch ich war noch nicht zufrieden und wollte mehr erreichen. Als ich dann versuchte, einen **WS2812B LED-Strip** in das Projekt zu integrieren, stellte sich heraus, dass mir wichtige Bauteile wie ein **Level Shifter** fehlten, was erneut eine Verzögerung bedeutete. Gleichzeitig stellte ich fest, dass meine vorhandenen Smart-Geräte keine **Matter**-Unterstützung hatten, was mich dazu brachte, meine Strategie anzupassen.
Die Einrichtung von **Home Assistant**, **OTBR** und **Matter** auf meinem neuen Notebook verlief ebenfalls nicht reibungslos, was zusätzliche Zeit erforderte. Schließlich entschied ich mich, einen **Amazon Echo 4** zu kaufen, der als Thread Border Router fungiert und Matter unterstützt. Mit dieser Lösung konnte ich schließlich die **RGB-Lichter** in mein Smart-Home-System integrieren und über Alexa steuern.

Obwohl es auf diesem Weg viele Hindernisse gab und ich oft frustriert war, weil ich mehr erreichen wollte, habe ich unglaublich viel gelernt. Angefangen bei den Technologien **Matter** und **Thread**, über den Umgang mit **Mikrocontrollern**, bis hin zur Programmierung in der **Arduino IDE**. Auch die Arbeit mit **Home Assistant**, **OTBR**, **Docker** und der Kommandozeile hat meinen technischen Horizont erweitert.
Am Ende bin ich stolz darauf, was ich erreicht habe, und es hat mir viel Freude bereitet, das Projekt umzusetzen. Es hat mich dazu motiviert, weiter an meinen Ideen zu arbeiten, und ich freue mich auf die zukünftigen Möglichkeiten, das Projekt weiterzuentwickeln.

Literaturverzeichnis

https://docs.arduino.cc/tutorials/nano-matter/user-manual/#matter
https://github.com/SiliconLabs/arduino?tab=readme-ov-file
https://store.arduino.cc/en-de/pages/nano-matter
https://www.google.de/url?sa=i&url=https%3A%2F%2Fraydiy.de%2Fneopixel-mit-arduino-und-esp32-der-led-streifen-ultra-guide%2F&psig=AOvVaw0iNph_WNZ12dwqYVERomOl&ust=1727452268205000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCLD4vrL74IgDFQAAAAAdAAAAABAE
https://img.goodfon.com/wallpaper/big/0/3c/minions-bob-look-happy.webp