|  |  |
| --- | --- |
| HTBLuVA Graz–Gösting  Ibererstraße 15-21  8051 Graz | bulme logo neu |

Abteilung für Elektronik und Technische Informatik

Ausbildungszweig Hardware-Software Co-Design und Biomedizintechnik

Smart-Display

Diplomarbeit in den Fächern  
HWE und FSST

Eingereicht von: Christian Lukas

Hannes Posch

im Schuljahr 2019/2020

der Klasse 5BHEL

bei Prof DI Rudolf Schamberger

am 03. März 2020

Inhaltsverzeichnis

[Abbildungsverzeichnis III](#_Toc36926855)

[Tabellenverzeichnis IV](#_Toc36926856)

[Abkürzungsverzeichnis V](#_Toc36926857)

[1 Danksagung 1](#_Toc36926858)

[2 Ziele und Inhalte 2](#_Toc36926859)

[2.1 Kurzbeschreibung 2](#_Toc36926860)

[2.2 Abstract 2](#_Toc36926861)

[3 Projektplanung 3](#_Toc36926862)

[3.1 Pflichtenheft 3](#_Toc36926863)

[3.2 Aufgabenteilung 3](#_Toc36926864)

[4 Kosten- und Zeitplan 4](#_Toc36926865)

[4.1 Kostenplan 4](#_Toc36926866)

[4.2 Zeitplan 5](#_Toc36926867)

[5 Projektrealisierung 7](#_Toc36926868)

[5.1 Mechanischer Aufbau 7](#_Toc36926869)

[5.1.1 Tischtennisballfläche 7](#_Toc36926870)

[5.1.2 Rahmen 9](#_Toc36926871)

[5.1.3 LED-Streifen 9](#_Toc36926872)

[5.1.4 Grundplatte 10](#_Toc36926873)

[5.1.5 Zusammenbau 11](#_Toc36926874)

[5.2 Hardware 12](#_Toc36926875)

[5.2.1 NodeMCU ESP8266 12](#_Toc36926876)

[5.2.2 WS2812B 13](#_Toc36926877)

[5.2.3 RTC DS3231 13](#_Toc36926878)

[5.2.4 DHT22 14](#_Toc36926879)

[5.2.5 LM1117 14](#_Toc36926880)

[5.2.6 Widerstand 18](#_Toc36926881)

[5.2.7 Kondensator 20](#_Toc36926882)

[5.2.8 Schaltplan und Layout 20](#_Toc36926883)

[5.2.9 Platinenfertigung 24](#_Toc36926884)

[5.2.10 Netzteil 24](#_Toc36926885)

[5.3 Software 24](#_Toc36926886)

[5.3.1 Testen des RGB-LED-Streifens 25](#_Toc36926887)

[5.3.2 Einstellen der RTC 27](#_Toc36926888)

[5.3.3 Testen des DHT22 29](#_Toc36926889)

[5.3.4 Erstellen einer geeigneten Matrix 31](#_Toc36926890)

[5.3.5 Testen der Matrix 32](#_Toc36926891)

[5.3.6 Font 34](#_Toc36926892)

[5.3.7 NTP-Server 35](#_Toc36926893)

[5.3.8 Programmiersprache C 37](#_Toc36926894)

[6 Diskussion 38](#_Toc36926895)

[6.1 Reflexion der Projektrealisierung 38](#_Toc36926896)

[6.1.1 Probleme bei der Platinenfertigung 38](#_Toc36926897)

[7 Literaturverzeichnis 39](#_Toc36926898)

[8 Quellen der Grafiken 40](#_Toc36926899)

[9 Eidesstattliche Erklärung 41](#_Toc36926900)

# Abbildungsverzeichnis

[Abbildung 5.1.1‑1: Zeitplan von Lukas Christian 5](#_Toc36926832)

[Abbildung 5.1.1‑2: Zeitplan von Posch Hannes 6](#_Toc36926833)

[Abbildung 5.1.1‑1: Zuschneiden eines Tischtennisballs 7](#_Toc36926834)

[Abbildung 5.1.1‑2: Konstruktion für das Kleben der Tischtennisbälle 8](#_Toc36926835)

[Abbildung 5.1.1‑3: Tischtennisballfläche 8](#_Toc36926836)

[Abbildung 5.1.2‑1: Rahmen 9](#_Toc36926837)

[Abbildung 5.1.3‑1: LED-Streifen 10](#_Toc36926838)

[Abbildung 5.1.4‑1: Grundplatte mit aufgeklebten LED-Streifen 10](#_Toc36926839)

[Abbildung 5.1.5‑1: Vorderseite des Smart-Displays 11](#_Toc36926840)

[Abbildung 5.1.5‑2: Rückseite des Smart-Displays 12](#_Toc36926841)

[Abbildung 5.1.5‑3: Befestigung der Grundplatte an den Rahmen 12](#_Toc36926842)

[Abbildung 5.2.5‑1: LM1117 15](#_Toc36926843)

[Abbildung 5.2.5‑2: Aufbau der Schaltung des LM1117 16](#_Toc36926844)

[Abbildung 5.2.5‑3: Einstellungen zur Simulation 17](#_Toc36926845)

[Abbildung 5.2.5‑4: Simulation der Spannung am ADJ-Pin 18](#_Toc36926846)

[Abbildung 5.2.6‑1: Schaltzeichen 18](#_Toc36926847)

[Abbildung 5.2.6‑2: Widerstand eines Leiters 19](#_Toc36926848)

[Abbildung 5.2.7‑1: Plattenkondensator 20](#_Toc36926849)

[Abbildung 5.2.8‑3: Schalplan Version 1.0 21](#_Toc36926850)

[Abbildung 5.2.8‑4: Layout Version 1.0 22](#_Toc36926851)

[Abbildung 5.2.8‑5: Aktuelle Version der Schaltung 23](#_Toc36926852)

[Abbildung 5.2.8‑6: Aktuelle Version des Layouts 24](#_Toc36926853)

[Abbildung 5.3.6‑1:Matrix und Font 34](#_Toc36926854)

# Tabellenverzeichnis

[Tabelle 4.1‑1: Kostenplan 4](#_Toc36926901)

[Tabelle 5.3‑1: Stellenwertsystem 35](#_Toc36926902)

# Abkürzungsverzeichnis

µC Mikrocontroller

LED Light-emitting diode

PSRR Power supply rejection ratio

RGB Rot, Grün und Blau

RTC Real-time clock

SoC System-on-a-Chip

# Danksagung

Wir bedanken uns herzlich bei Dipl.-Ing. Rudolf Schamberger für die Betreuung der Diplomarbeit. Außerdem möchten wir uns auch für seine antreibenden Worte bedanken.

Weiters möchten wir uns bei Ing. Günther Amtmann für die Unterstützung bei EAGLE und auch bei Ing. Jürgen Wieland für die Unterstützung bei der Fertigung der Platine bedanken.

Anschließend bedanken wir uns auch bei unserem Klassenvorstand Mag. Horst Kollingbaum, da wir immer mit seiner Unterstützung rechnen konnten.

Zu guter Letzt möchten wir uns bei Valentin Posch für die Unterstützung beim Aufbau und für die Zurverfügungstellung des Materials bedanken.

# Ziele und Inhalte

## Kurzbeschreibung

Das Diplomarbeitsprojekt ist eine mobile Tischtennisball-LED-Anzeige und als Ideengeber hat dazu das Projekt „Ping Pong Ball LED Clock“ gedient. Der Aufbau des Smart-Display verfügt über einen µC, dazu wird das NodeMCU ESP8266 verwendet, und hinter der Tischtennisballfläche befindet sich eine RGB-LED Matrix, bestehend aus WS2812B RGB-LED-Streifen. Die Daten für die Anzeige von Uhrzeit und Datum lesen wir von der RTC DS3231 aus. Der DHT22 Temperatur- und Luftfeuchtigkeitssensor dient für die Daten des gleichnamigen Sensors. Eine 7 x 5 Matrix im Code dient dazu Buchstaben, Zahlen und Sonderzeichen darzustellen.

## Abstract

The diploma thesis is a mobile table tennis ball LED display and the project "Ping Pong Ball LED Clock" served as the idea generator. The structure of the Smart-Display has a µC, for this purpose the NodeMCU ESP8266 is used, and behind the ping pong ball surface there is an RGB LED matrix consisting of WS2812B RGB LED strips. The data for the display of time and date is read from the RTC DS3231. The DHT22 temperature and humidity sensor is used for the data of the sensor with the same name. A 7 x 5 matrix in code is used to display letters, numbers and special characters.

# Projektplanung

## Pflichtenheft

Das Smart-Display umfasst einen mechanischen und elektronischen Aufbau, sowie einen Softwareteil.

Der mechanische Aufbau teilt sich in das Zuschneiden und Kleben der Tischtennisbälle, sowie in einen Holzaufbau, in dem die Tischtennisballfläche fixiert wird. Weiters wird hinter der Tischtennisballfläche eine RGB-LED Matrix auf einer Holzgrundplatte angebracht. Für die RGB-LED Matrix ist der WS2812b RGB-LED Streifen in mehrere kleine LED Streifen aufgeteilt worden und anschließend mit Litzenleitungen wieder verbunden worden. Zu guter Letzt ist die Grundplatte gebohrt worden, danach ist der RGB-LED Streifen hinaufgeklebt worden und hinter der Tischtennisballfläche mit Holzstücken fixiert worden.

Beim elektronischen Aufbau handelt es sich um die Erstellung einer Schaltung und eines Layouts, welches für die Fertigung der Platine verwendet wird. Nach der Bestückung der Platine kann auch diese hinter der Grundplatte befestigt werden.

Der Softwareteil teilt sich in das Auslesen der Uhrzeit und des Datums von der RTC DS3231, der Temperatur und Luftfeuchtigkeit des DHT22 und in das Beschreiben der RGB-LED Matrix mittels dem NodeMCU ESP8266. Für die Ausgabe von Buchstaben, Zahlen und Sonderzeichen ist im Code eine 7 x 5 Matrix erstellt worden.

## Aufgabenteilung

**Lukas Christian:**

Lukas Christian ist verantwortlich für die Software. Bei der Software kümmert er sich um die Anzeige von Uhrzeit, Datum, Temperatur und Luftfeuchtigkeit und um die Ausgabe des Lauftextes. Außerdem ist er für den Hardwareteil unterstützend da.

**Posch Hannes:**

Posch Hannes ist verantwortlich für die mechanischen und elektronischen Komponenten und deren Aufbau. Bei der Hardware ist er für den Schaltplan, das Layout und die Anfertigung und Bestückung der Platine zuständig.

# Kosten- und Zeitplan

## Kostenplan

|  |  |  |  |
| --- | --- | --- | --- |
| **Bezeichnung** | **Stückpreis** | **Anzahl** | **Preis** |
| Tischtennisbälle 6 Stk. 40 mm weiß | 0,79 € | 23 | 18,17 € |
| NEU026 Steckverbindungen, Stecker | 0,95 € | 1 | 0,95 € |
| LM 1117 MPX-ADJ | 1,16 € | 3 | 3,48 € |
| DEBO DHT 22 BRD | 5,69 € | 3 | 17,09 € |
| RTC DS3231 | 6,35 € | 1 | 6,35 € |
| Node MCU ESP8266 | 6,35 € | 1 | 6,35 € |
| WS2812b | 22,06 € | 1 | 22,06 € |
| 4 pcs 1m/3.28ft 3 Pin JST SM Männlich Weiblich Stecker | 9,99 € | 1 | 9,99 € |
| SMD WID. 470R 1206 | 0,05 € | 5 | 0,25 € |
| SMD-Widertand, 0603, 120 Ohm, 200 mW | 0,02 € | 6 | 0,10 € |
| SMD-Widertand, 0603, 200 Ohm, 200 mW | 0,02 € | 6 | 0,10 € |
| ELKO RAD. 100 μF 50V | 0,15 € | 5 | 0,75 € |
| Kondensator 0,1 μF KDPU 2,54 MM | 0,35 € | 5 | 1,75 € |
| MINI EL Kondensator 10 μF 35V | 0,25 € | 5 | 1,25 € |
| Dehner SYS 1449-1505-W2E Netzt Blister | 18,99 € | 1 | 18, 99 € |
| Platine | 1,92 € | 10 | 19,19 € |
| Holzrahmen, Grundplatte, Kleinmaterial, Heißklebepatronen |  |  | 45,00 € |
|  |  |  | 152,83 € |

Tabelle 4.1‑1: Kostenplan

## Zeitplan



Abbildung 5.1.1‑1: Zeitplan von Lukas Christian



Abbildung 5.1.1‑2: Zeitplan von Posch Hannes

# Projektrealisierung

## Mechanischer Aufbau

### Tischtennisballfläche

Als erstes wird eine Halterung mit drei Holzstücken für die Tischtennisbälle angefertigt, in einem Schraubstock eingespannt und mit einer Schraubzwinge befestigt. Anschließend wird ein Tischtennisball nach dem anderen eingespannt. Folgend wird mit der Taschenlampe geschaut, dass die Mittellinie nicht von der Vorderseite sichtbar ist und danach können die Tischtennisbälle durchgeschnitten werden. Zum Schluss können mittels eines Stanley-Messers und Schleifpapier überstehende Reste weggeschnitten und zurecht geschliffen werden.



Abbildung 5.1.1‑1: Zuschneiden eines Tischtennisballs

Anschließend werden die zugeschnittenen Tischtennisbälle zusammengeklebt. Als erstes nimmt man zwei Tischtennisbälle und spannt sie in eine Halterung von 4 Holzstücken ein und befestigt diese mit zwei Schraubzwingen. An der Seite werden noch zwei Holzstücke platziert, damit die Tischtennisbälle nicht verrutschen. Zum Schluss schaut man mit einer Holzleiste, dass sie waagrecht liegen. Als nächstes kann mit der Heißklebepistole an der Schnittstelle der zwei Tischtennisbälle mit einem Tropfen Heißkleber zusammenkleben. Sobald der Heißkleber getrocknet ist, kann man weitere Tischtennisbälle dazu kleben. Diesen Vorgang wiederholt man für jede Reihe. Nachdem alle Reihen geklebt wurden, beginnt man diese miteinander an jedem Berührungspunkt für eine gute Stabilität zusammenzukleben. Wenn zum Schluss von den Klebestellen noch Kleber weg steht, kann man mittels einer Pinzette, diesen entfernen.

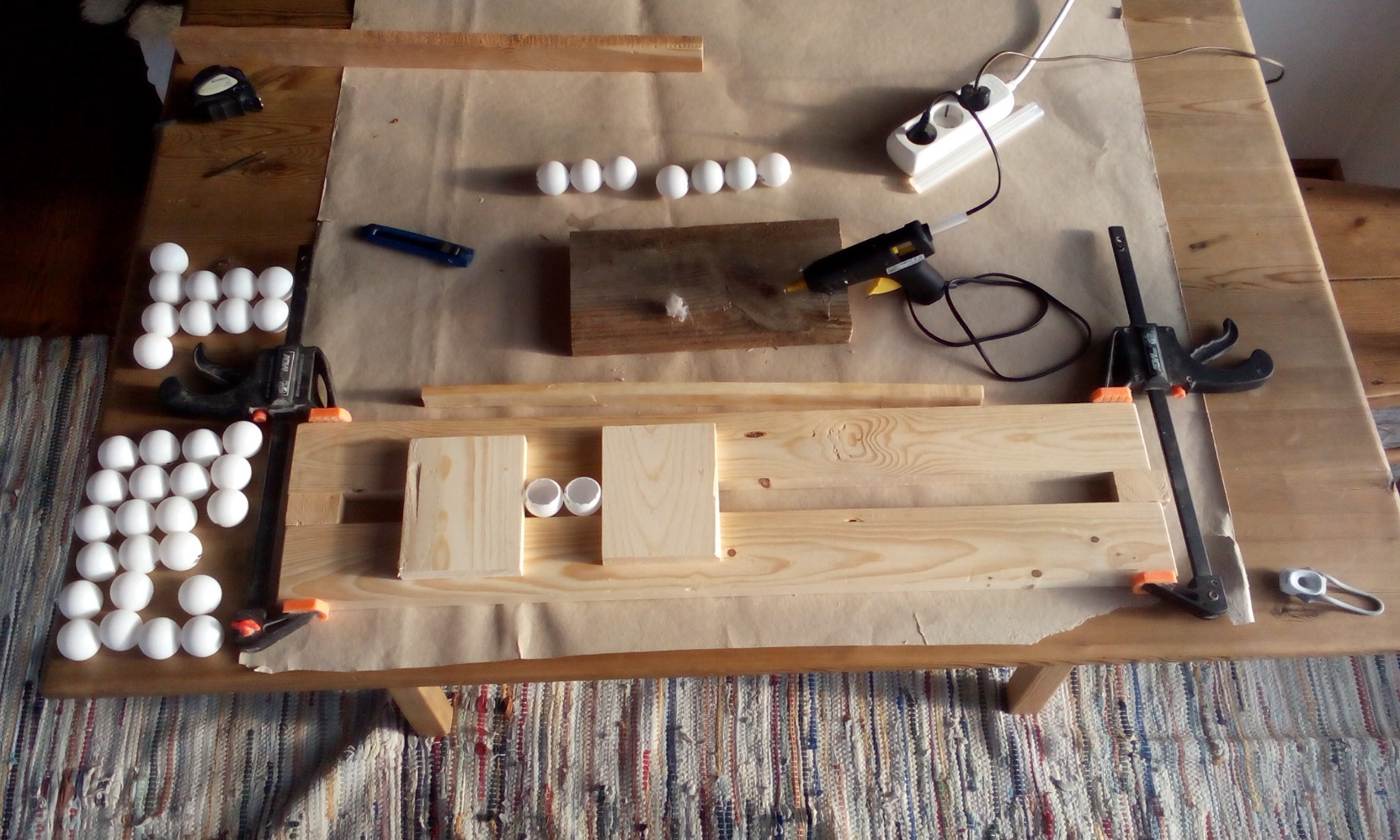


Abbildung 5.1.1‑2: Konstruktion für das Kleben der Tischtennisbälle



Abbildung 5.1.1‑3: Tischtennisballfläche

### Rahmen

Als nächstes werden mithilfe der Tischtennisballfläche und einem Lineal, die Außenmaße der Tischtennisballfläche (= Innenmaße des Rahmens) eingezeichnet. Anschließend werden die Maße auf das Holz eingezeichnet und ausgeschnitten. Danach wird mit der Nutfräse bei allen sechs Holzstücken beidseitig hineingefräst. Im Anschluss wird bei allen Berührungspunkten der einzelnen Holzstücke Holzleim aufgetragen und ein Flachdübel bei jeder Verbindung hineingepresst. Zum Schluss befestigt man den Rahmen mit Schraubzwingen. Wenn der Rahmen getrocknet ist, kann man ihn noch mit einem Lack bestreichen, damit er besser gegen die Witterung geschützt ist.



Abbildung 5.1.2‑1: Rahmen

### LED-Streifen

Im nächsten Schritt wird der LED-Streifen in einen LED-Streifen mit einer LED, in einen LED-Streifen mit drei LEDs, in einen LED-Streifen mit fünf LEDs und in 17 LED-Streifen mit sieben LEDs zugeschnitten. Von den Litzenleitungen werden pro Farbe 17 Stück zu je zehn cm und 2 Stück zu je 15 cm benötigt. Zum Schluss werden circa 0,5 cm von den Litzenleitungen mittels einer Abisolierzange abisoliert.

Anschließend werden an die abgeschnittenen LED-Streifen die Litzenleitungen angelötet. Nachdem alles fertig gelötet ist, kann die LED-Fläche für den Prototypenaufbau auf einen Karton mit Klebeband befestigt werden. Bei diesem Schritt kann auch geschaut werden, dass die LEDs passend hinter der Tischtennisballfläche positioniert werden.

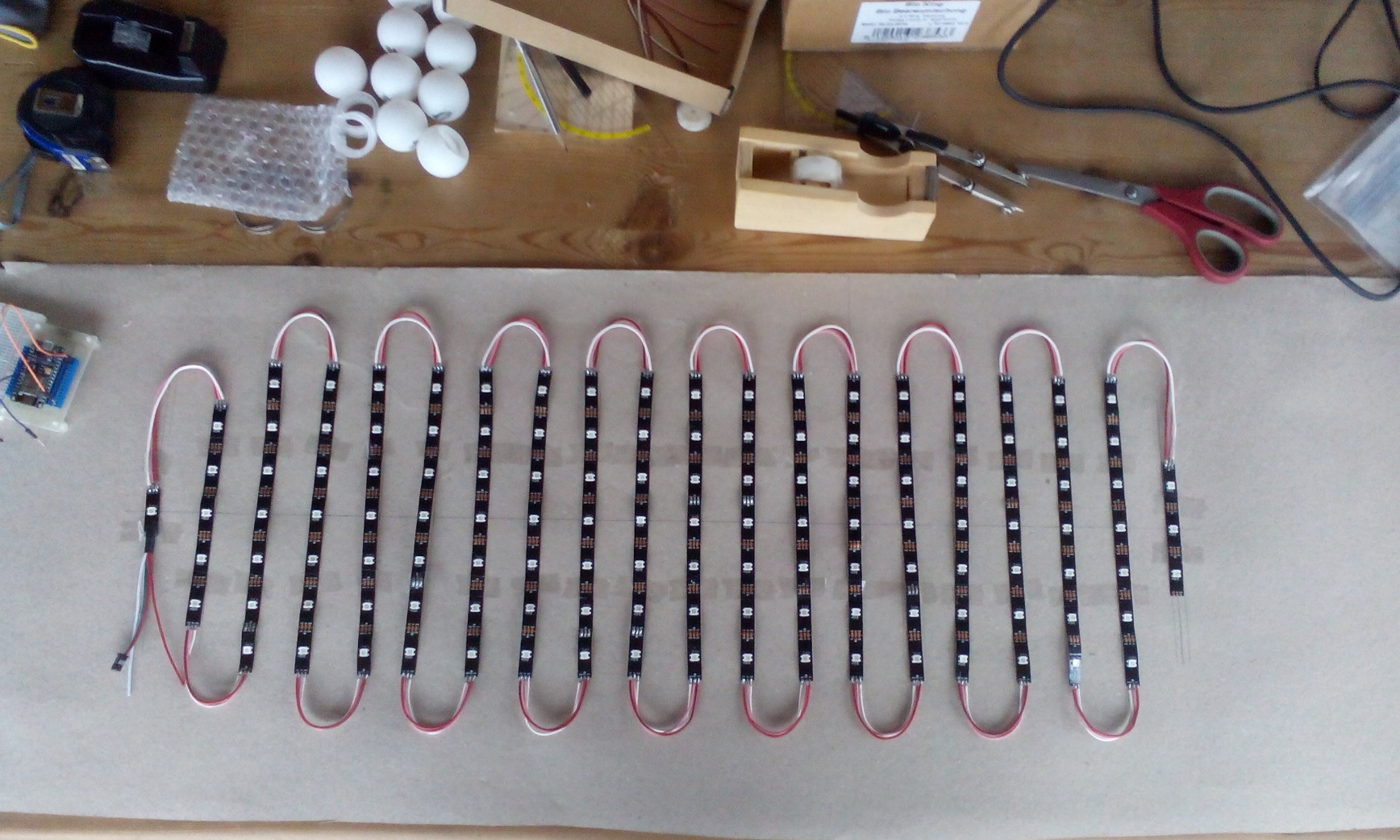


Abbildung 5.1.3‑1: LED-Streifen

### Grundplatte

Als erstes nimmt man die Außenmaße der Tischtennisballfläche (= Innenmaße des Rahmens) und überträgt diese auf eine Holzplatte. Als nächstes wird die Holzplatte zugeschnitten und die Positionen der LED-Streifen aufgezeichnet. Danach verwendet man einen 15 mm Bohrer und bohrt bei jedem eingezeichneten Punkt ein Loch. Anschließend schneidet man bei jeder Bohrung einen schmalen Pfad hinein, um im nächsten Schritt den LED-Streifen einfacher auf der Grundplatte zu befestigen können.

Folgend wird von den einzelnen LED-Streifen der Klebestreifen nacheinander abgezogen und aufgeklebt.

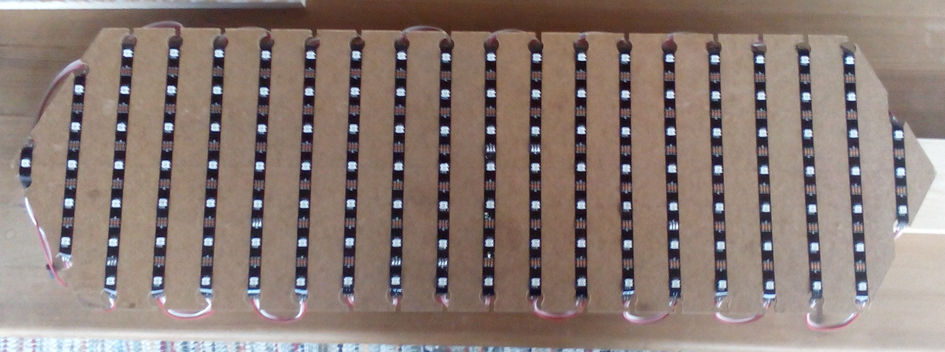


Abbildung 5.1.4‑1: Grundplatte mit aufgeklebten LED-Streifen

### Zusammenbau

Im nächsten Schritt nimmt man den Rahmen und verwendet die Grundplatte als Abstand, dass die Tischtennisballfläche nicht auf einer Linie mit dem Rahmen ist und legt die Tischtennisballfläche auf die Grundplatte in den Rahmen. Anschließend wird bei jedem Schnittpunkt zwischen dem Rahmen und der Tischtennisballfläche geklebt. Bei diesem Schritt sollte man aufpassen, dass der Heißkleber nicht in die Tischtennisbälle rinnt.



Abbildung 5.1.5‑1: Vorderseite des Smart-Displays

Der letzte Schritt beschäftigt sich mit der Fertigstellung des mechanischen Aufbaues. Als erstes werden acht Holzstücke auf eine beliebige Größe zugeschnitten und mit einem Bohrer durchgebohrt. Anschließend legt man die Grundplatte mit der aufgeklebten LED-Fläche in den Rahmen mit der aufgeklebten Tischtennisballfläche. Mit leichtem Druck schraubt man die Holzstücke mit einem beliebigen Abstand voneinander an den Rahmen an.

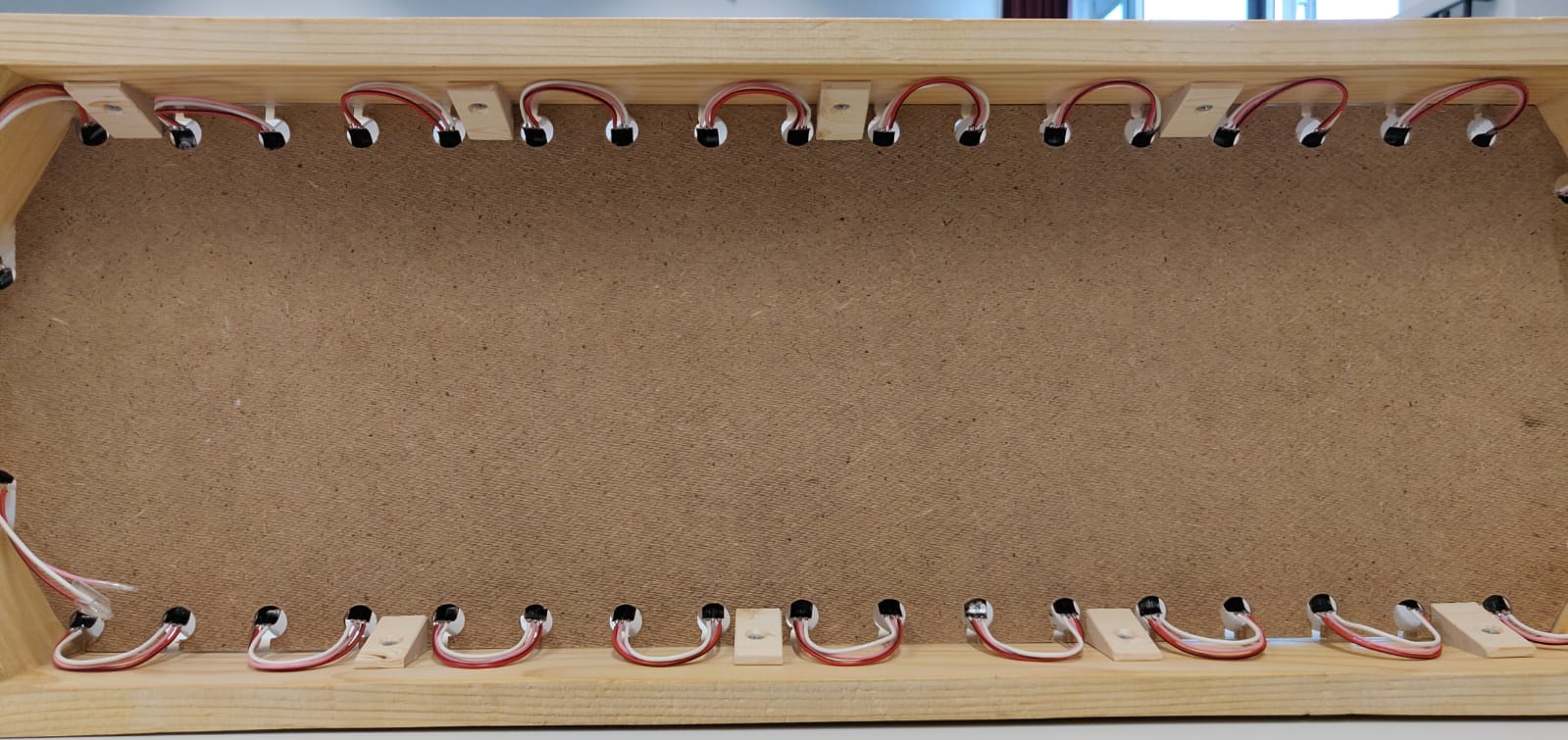


Abbildung 5.1.5‑2: Rückseite des Smart-Displays

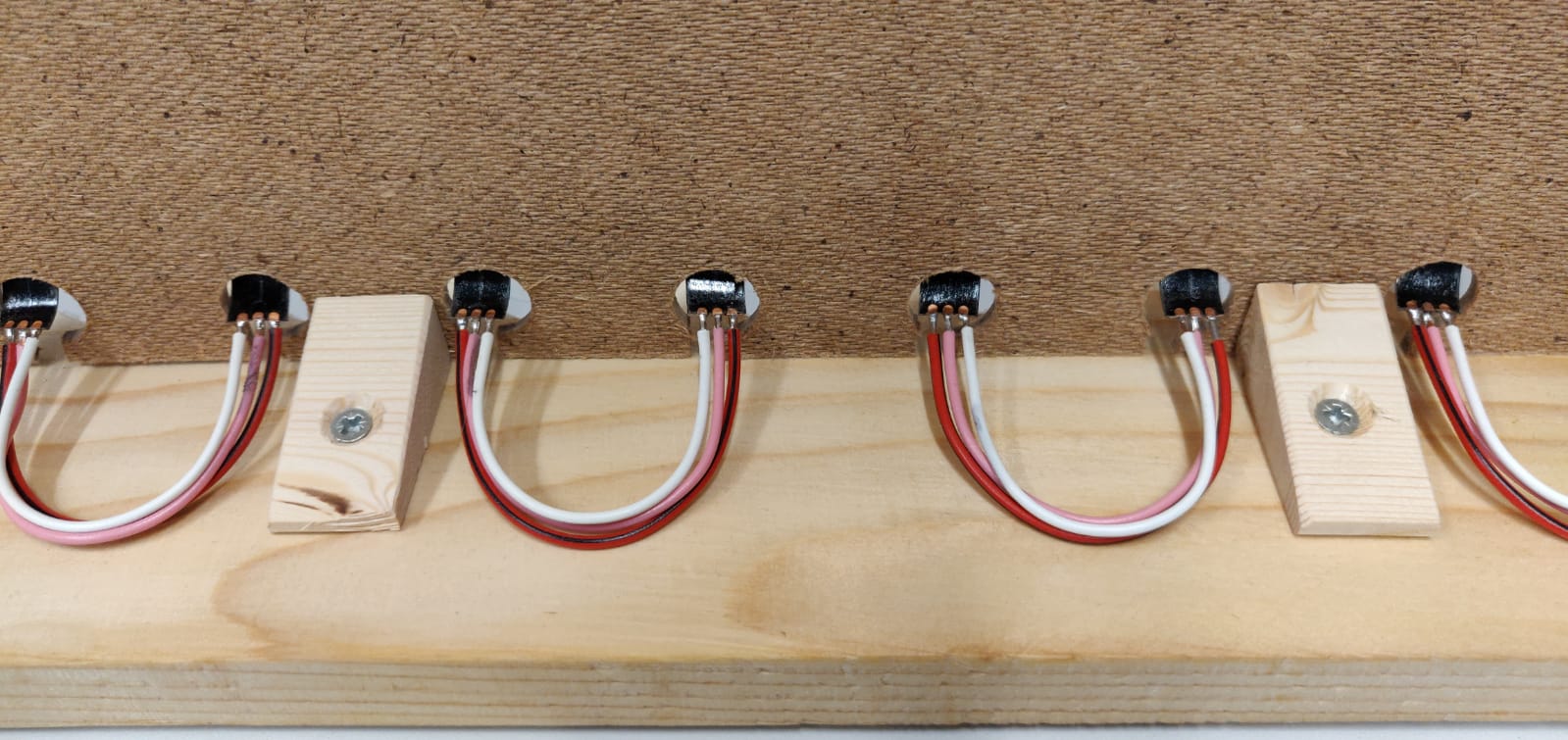


Abbildung 5.1.5‑3: Befestigung der Grundplatte an den Rahmen

## Hardware

### NodeMCU ESP8266

Als Mikrocontroller verwenden wir in unserer Diplomarbeit den NodeMCU ESP8266, dieser ist ein 32-Bit-Mikrocontroller. Wenn man sich entscheidet mit einem Arduino-Board zu arbeiten, bekommt auch gleich die Möglichkeit den Chip über die hauseigene Arduino IDE zu beschreiben. Das ESP8266-Modul von der chinesischen Firma „Espressif“ unterstützt verschiedene Netzwerkprotokolle. Der SoC selbst befindet sich nicht direkt auf dem NodeMCU, sondern ist auf dem „ESP8266mod“ Gehäuse verbaut. Die Betriebsspannung des NodeMCU ESP8266 ist 3,3 V. Weiters ist auf dem ESP8266 ein integrierter WLAN-Chip vorhanden. Da er auch sehr kostengünstig und einen geringen Leistungsbedarf hat, wird der ESP8266 für viele Anwendungen verwendet.[[1]](#footnote-1)[[2]](#footnote-2)

### WS2812B

Die WS2812b ist die bekannteste und meist verwendeter RGB-LED-Streifen. Die Besonderheit des WS2812b ist, dass sie im Gegensatz zu konventionellen LED-Streifen einen IC für jede einzelne RGB-LED bereitsteht. Damit können individuelle Signale empfangen werden und man kann sie damit auch direkt ansteuern. Der IC und die LED teilen sich dabei eine Stromquelle. Der Chip befindet sich auf einer 5 x 5 mm Komponente und ergibt somit ein vollständiges Pixel. Die integrierte Signalumformungsschaltung sorgt nach der Umformung auf den nächsten Treiber dafür, dass keine aufgestaute Wellenformverzerrung auftritt. Darüber hinaus verfügt es über einen elektrischen Reset-Schaltkreis. Jedes Pixel verfügt über 256 Helligkeitsstufen, damit können über 16 Millionen verschiedene Farben dar-gestellt werden und bei einer Scanfrequenz von 400 Hz/s. Die Besonderheit ist, dass es ein kaskadierendes Portübertragungssignal besitzt, welches über eine einzige Leitung funktioniert. Die Datenübertragung von einer LED zur nächsten funktioniert ohne Verstärkung bis zu einer Entfernung von 5 m. Bei einer Bildwiederholungsfrequenz von 30 Bildern pro Sekunde beträgt die Kaskadenzahl 1024 Punkte. Die Daten werden dabei mit 800 kbps übertragen. Die RGB-LEDs verfügen auch über einen intelligenten Verpolungsschutz, sodass der IC im Normalfall nicht beschädigt wird. Es gibt viele verschiedene Verwendungszwecke und dementsprechend haben sie auch „International Protection“ (IP) von IP20 bis IP65. Angesteuert werden können die RGB-LEDs über verschiedene Controller.[[3]](#footnote-3)

### RTC DS3231

RTCs werden dafür verwendet, dass die aktuelle Uhrzeit immer zur Verfügung steht, ohne dass die CPU etwas berechnen muss. Der DS3231 ist eine serielle RTC, die von einem temperaturkompensierten 32kHz Quarzoszillator angetrieben wird. Die RTC DS3231 wird über eine I2C Schnittstelle angesprochen und hat eine Betriebsspannung von 3,3 V. Außerdem verfügt die Uhr über einen Batterie-Eingang, welcher eine kontinuierliche Zeitmessung behält, falls die Hauptstromversorgung des Geräts unterbrochen wird. Die Echtzeituhr liefert Sekunden, Minuten, Stunden, Tag, Datum, Monat und Jahr. Das Datum wird am Monatsende für Monate mit weniger als 31 Tagen automatisch angepasst, einschließlich Korrekturen für das Schaltjahr. Weiters arbeitet sie entweder in einem 24-Stunden- oder in einem 12-Stunden-Format mit AM- / PM-Anzeige. Die Register für Zeit und Datum sind BCD-kodiert, dies erleichtert die Aufteilung in 10er- und 1er-Stellen für die Anzeige. Wird die Uhr gelesen oder beschrieben, ändern sich die Register nicht, somit können Fehllesungen oder -beschreibungen verhindert werden.[[4]](#footnote-4)

### DHT22

Der DHT22 ist ein Luftfeuchtigkeits- und Temperatursensor und kann von 3,3 bis 6 V betrieben werden, daher eignet sich der DHT22 zur Verwendung mit allen gängigen Boards. In unserer Schaltung verwenden wir das Breakoutboard des DHT22. Da nur ein Anschluss für die Daten am DHT22 vorhanden ist, erfolgt die Ausgabe als serielle Bitfolge. Der Temperaturmessbereich befindet sich zwischen -40 bis +80 °C und hat eine Genauigkeit von ± 0,5 °C. Der Luftfeuchtigkeitsbereich liegt zwischen 0 und 100 % und hat eine Messgenauigkeit von ± 2 % relative Luftfeuchte. Weiters eignet sich der Sensor für die Überwachung des Raumklimas.[[5]](#footnote-5)

### LM1117

Der LM1117 ist ein vielseitiger und leistungsstarker Linearregler mit einem breiten Temperaturbereich und einem engen Leitungs-/Last-Regelbetrieb. Ein Ausgangskondensator von mindestens 10 µF ist erforderlich, um das Einschwingverhalten und die Stabilität weiter zu verbessern. Bei der ADJ-Version des LM1117 kann der ADJ-Pin überbrückt werden, um sehr hohe Welligkeitsunterdrückungsraten zu erreichen. Der LM1117 ist vielseitig einsetzbar, unter anderem als Nachregler für DC/DC-Wandler, Batterieladegeräte und auch für die Stromversorgung von Mikroprozessoren.[[6]](#footnote-6)

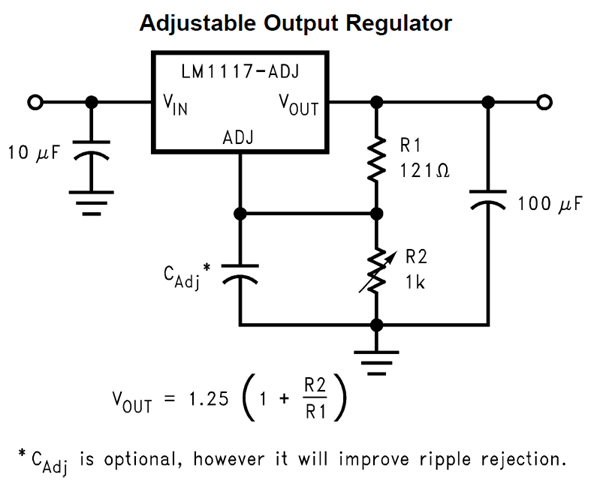


Abbildung 5.2.5‑1: LM1117

Die Anzahl der Komponenten für den LM1117 ist sehr gering. Die zwei Widerstände R1 und R2 werden in der Schaltung für einen Spannungsteiler und der Ausgangskondensator zur Lastregelung verwendet. Als Ausgangskondensator verwenden wir in unserer Schaltung, wie in Abbildung 5.2.5-1, einen 100 µF Kondensator. Ein 10 µF Kondensator am Eingang ist ein geeigneter Eingangskondensator für fast alle Anwendungen. Die Ausgangsspannung kann mit der Auswahl der beiden Widerstände R1 und R2 eingestellt werden. Ein optionaler Überbrückungskondensator CADJ über R2 kann ebenfalls zur Verbesserung des PSRR verwendet werden. Als Überbrückungskondensator verwenden wir in unserer Schaltung einen 100 nF Kondensator.[[7]](#footnote-7)

**Berechnung des Widerstandes für den LM1117:**

**Aufbau der Schaltung des LM1117 in Multisim:**

Nach der Berechnung des Widerstandes R2, ist es sicher auch gut zu wissen, wie viel Leistung die Widerstände benötigen. Als erstes werden die Bauteile eingefügt, positioniert und anschließend verbunden.

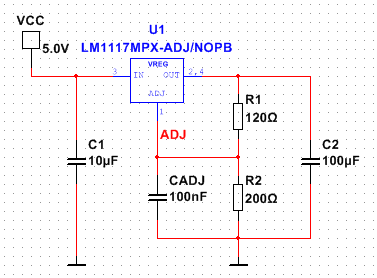


Abbildung 5.2.5‑2: Aufbau der Schaltung des LM1117

Als nächstes wird unter dem Punkt Simulieren auf Analysen und Simulation geklickt und beim Zeitverhalten die Spannung am ADJ-Pin hinzugefügt. Zum Schluss wird noch auf Start geklickt, um zu erfahren, wie viel Spannung am ADJ-Pin anliegt.

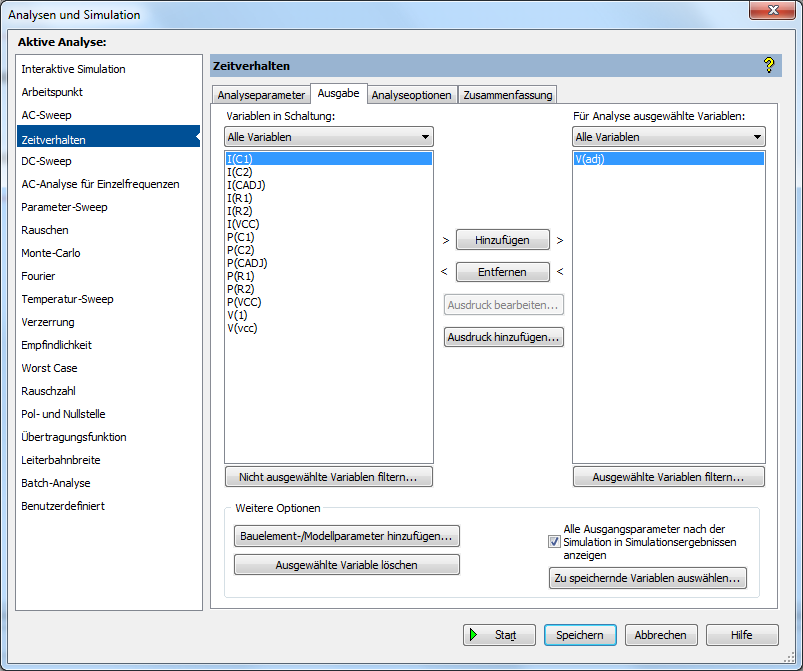


Abbildung 5.2.5‑3: Einstellungen zur Simulation

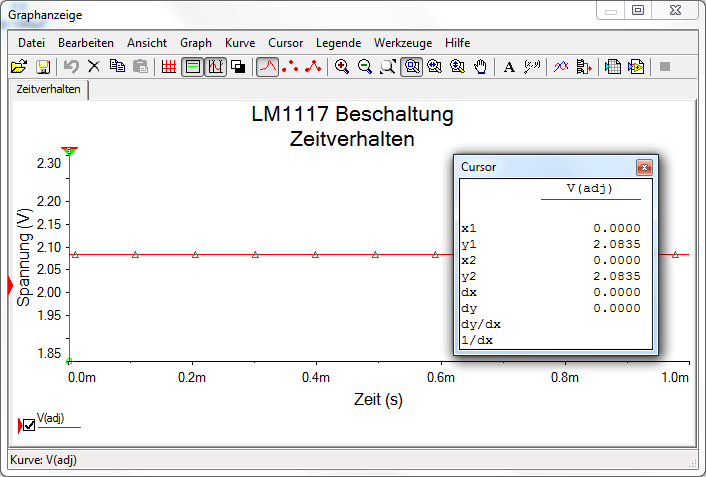


Abbildung 5.2.5‑4: Simulation der Spannung am ADJ-Pin

**Berechnung der Leistung an den Widerständen:**

### Widerstand

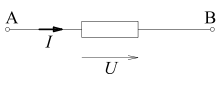


Abbildung 5.2.6‑1: Schaltzeichen

Das Formelzeichen des elektrischen Widerstandes ist R (engl. „resistor“, dt. „Widerstand“). Die Einheit des elektrischen Widerstandes ist Ohm mit dem Einheitenzeichen Ω (großes Omega). Der elektrische Widerstand zählt zu den abgeleiteten SI-Einheiten.

Zur Berechnung des elektrischen Widerstandes wird das ohmsche Gesetz verwendet: Das ohmsche Gesetz gibt den Zusammenhang zwischen Spannung und Strom an einem Widerstand an.

**Der elektrische Leitwert G:**

Der elektrische Leitwert zählt zu den abgeleiteten SI-Einheiten und hat die Einheit Siemens.

**Der elektrische Widerstand eines metallischen Leiters:**

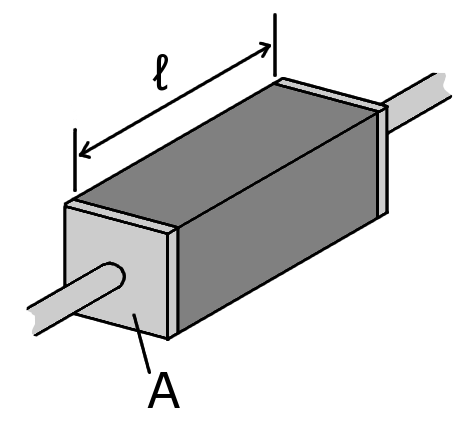


Abbildung 5.2.6‑2: Widerstand eines Leiters

Der elektrische Widerstand R eines Leiters ist abhängig von dessen Geometrie (A, l) und den Materialeigenschaften. Das Formelzeichen für das Material ist ρ (Rho).

l…Leiterlänge in m

A…Querschnitt in mm2

R…Widerstand des Leiters in Ω

ρ…spezifischer Widerstand in

Der spezifische Widerstand ρ ist der Widerstand eines Leiters von einem m Länge und einem mm2 Querschnitt. Die spezifische Leitfähigkeit γ ist der Kehrwert des spezifischen Widerstandes.

### Kondensator

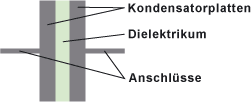


Abbildung 5.2.7‑1: Plattenkondensator

Ein Kondensator besteht aus zwei Metallelektroden (Platten), zwischen den Platten befindet sich ein Dielektrikum, welches als Isolator dient und somit keine elektrische Verbindung zwischen den Metallelektroden zulässt. Wenn eine Spannung an einem Kondensator angelegt wird, entsteht zwischen den beiden Metallplatten ein elektrisches Feld. Die Ladung auf den beiden Platten ist gleich groß, das heißt Q+ = Q-. Weiters gibt es gepolte Kondensatoren, auch Elektrolytkondensatoren genannt, bei diesen Kondensatoren ist zu achten, dass sie richtig angeschlossen werden und mit Gleichspannung betrieben werden.[[8]](#footnote-8)

Das Formelzeichen der elektrischen Kapazität ist C (engl. „capacity“, dt. „Kapazität“). Die Einheit der elektrischen Kapazität ist Farad mit dem Einheitenzeichen F. Die elektrische Kapazität zählt zu den abgeleiteten SI-Einheiten.

Für die Berechnung des Kondensators wird weiters die elektrische Ladung benötigt, welche als Formelzeichen Q mit der Einheit Coulomb mit dem Einheitenzeichen C besitzt und zählt, wie der elektrische Widerstand und die elektrische Kapazität, zu den abgeleiteten SI-Einheiten.

Zur Berechnung der elektrischen Kapazität wird diese Formel verwendet:

### Schaltplan und Layout

Der Schaltplan und das Layout wurden mit der Software Eagle 7.7.0 erstellt. Als erstes werden die benötigten Bauteile mit der Funktion „Add“ eingefügt und mit der Funktion „Net“ verbunden. Nachdem der Schaltplan fertig gestellt ist, wird mit der Funktion „Zum Board wechseln“ gleichnamiges erledigt. Im Board werden als erstes die Bauelemente auf eine geeignete Position gebracht, damit alles so einfach wie möglichst verbunden werden kann. Die Leiterbahnen sollen im 45° Modus verbunden werden, da der fließende Strom bei 90° Verbindungen, sonst mit voller Wucht gegen die Leitungen stoßen würde. Die Leiterbahnbreite wird an den Stromfluss angepasst.

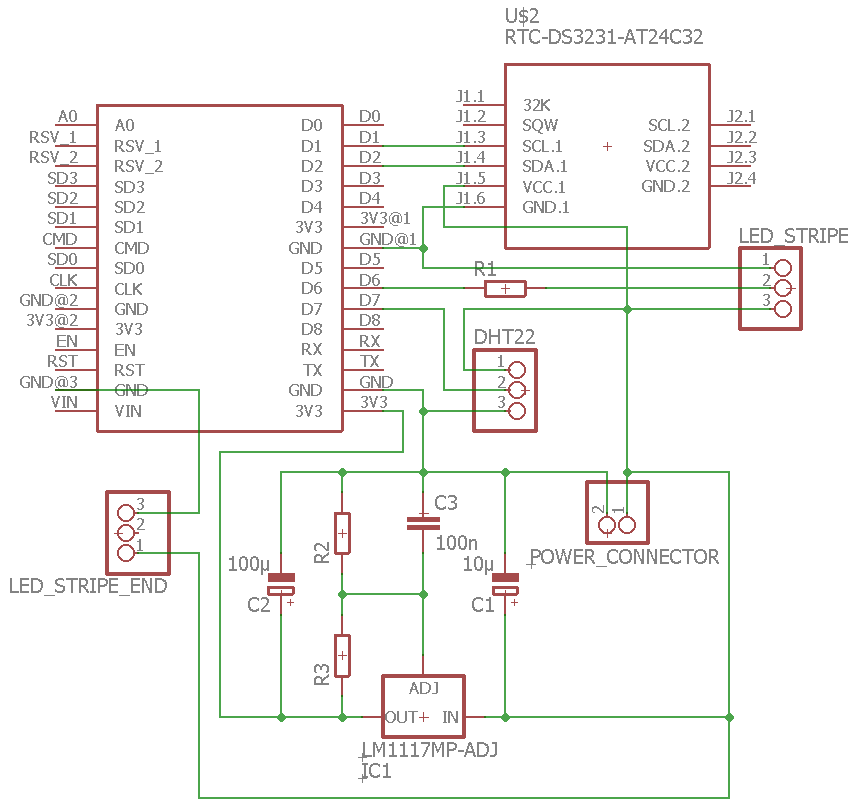


Abbildung 5.2.8‑3: Schalplan Version 1.0

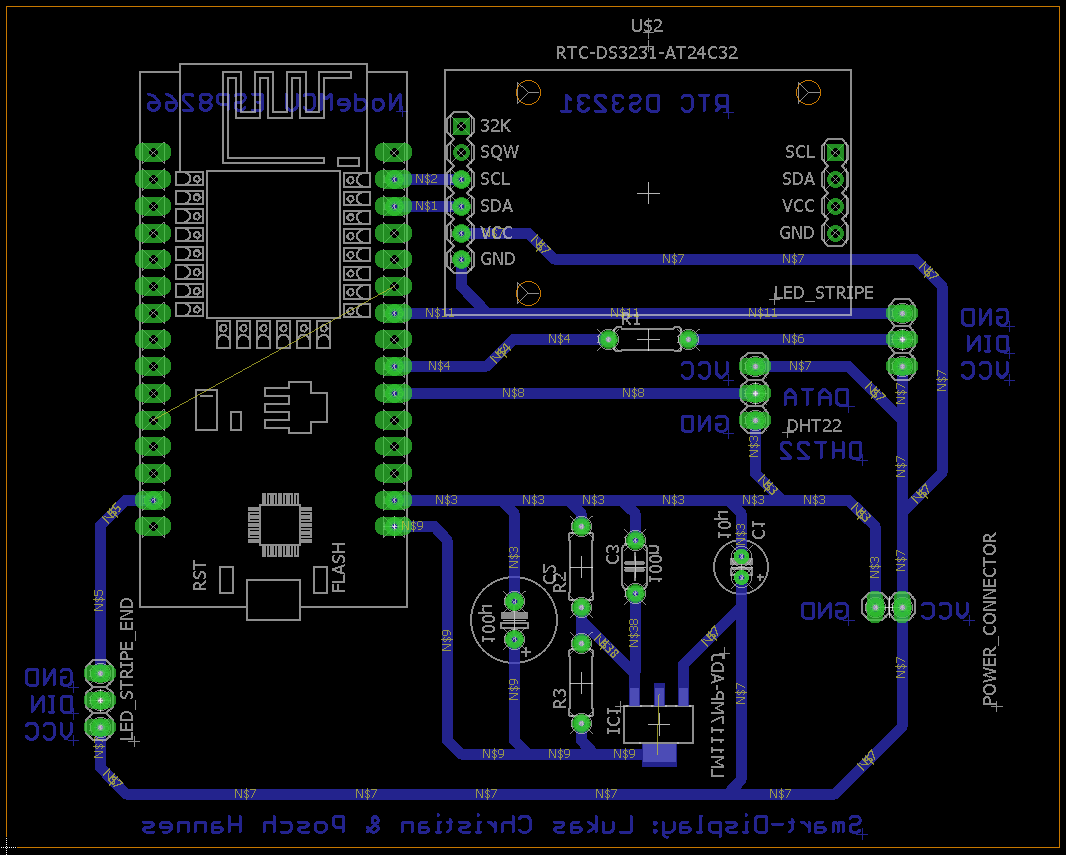


Abbildung 5.2.8‑4: Layout Version 1.0

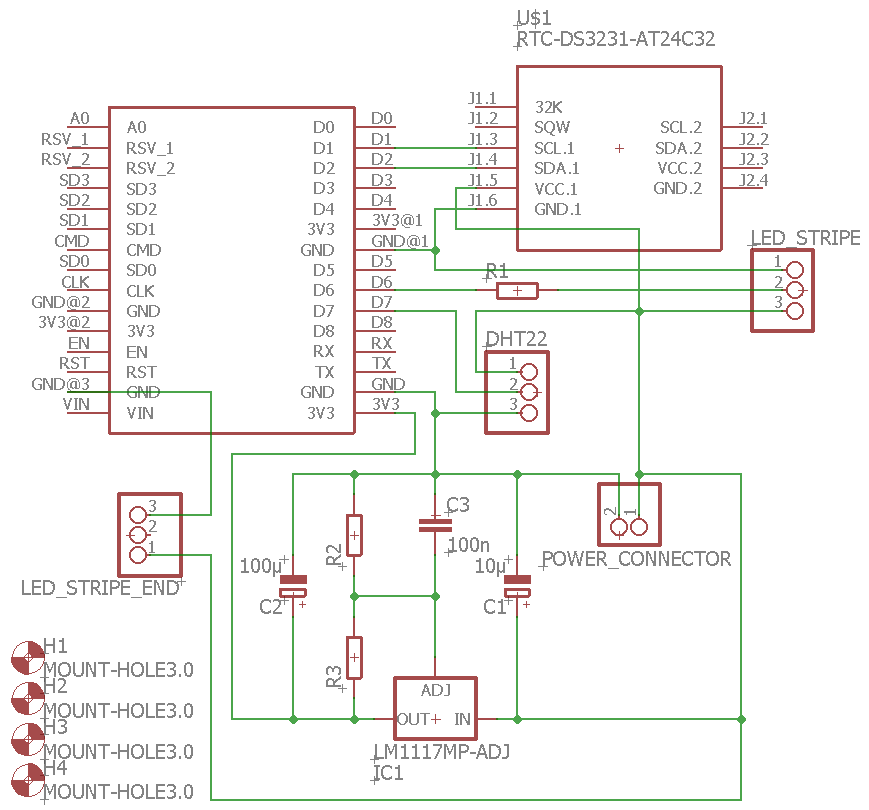


Abbildung 5.2.8‑5: Aktuelle Version der Schaltung

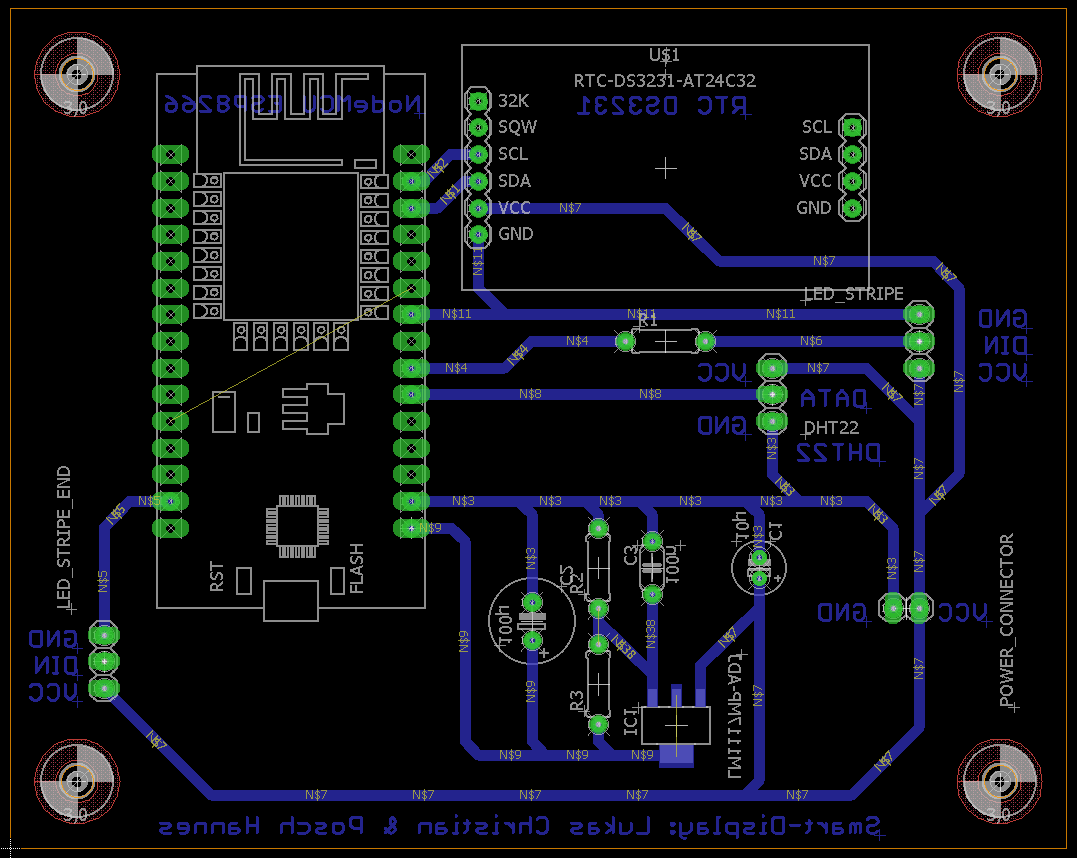


Abbildung 5.2.8‑6: Aktuelle Version des Layouts

### Platinenfertigung

Nachdem das Layout fertig ist, wird dieses mit den notwendigen Layern, dazu zählen Bottom, Pads, Vias und Dimension, ausgedruckt. Anschließend wird das Papier auf die Platine gelegt und mit UV-Strahlung für eine Minute belichtet. Nach dem Belichten wird es für sechs bis sieben Sekunden in die Natronlauge gehängt. Danach wird es für 15 bis 16 Minuten in das Ätzbad gehängt, um das Kupfer auf den Flächen herunterlösen zu können. Nachdem die Platine diesen Prozess durchgelaufen ist, kann sie gebohrt und anschließend bestückt werden.

Da es bei der Platinenfertigung einige Rückschläge gegeben hat, haben wir das Layout eingesendet und professionell fertigen lassen.

### Netzteil

## Software

Die Programmierung hat zuerst mit einem RGB-LED-Teststreifen begonnen, bei der wir nur ein paar RGB-LEDs zum Leuchten gebracht haben. Zu Beginn mussten wir entscheiden, welche Bibliothek wir verwenden wollen. Die Wahl lag bei der „FastLED“- und „Adafruit\_NeoPixel“-Bibliothek. Nachdem die „FastLED“-Bibliothek zu einigen Problemen geführt hat, die wir nicht sofort lösen konnten, haben wir uns dann für die „Adafruit\_NeoPixel“-Bibliothek entschieden. Die ersten Schritte sind dabei nicht die einfachsten gewesen. Nach vielen Tests und Fehlschlägen das Programm auf allen möglichen Mikrocontrollern hochzuladen, ist klar gewesen, dass etwas in der Software nicht funktioniert. Als nach einer Neuinstallation der Arduino IDE noch immer Fehler angezeigt worden sind, haben wir versucht andere bereits funktionierende Programme zu verwenden. Das Problem war, dass das NodeMCU einen weiteren Mikrocontroller installiert hat. Am NodeMCU befindet sich der ESP8266, welcher als Hauptprozessor angesprochen wurde. Da die Anschlüsse des NodeMCU und des ESP8266 ganz unterschiedliche Bezeichnungen haben, konnte der Mikrocontroller die Daten nicht korrekt einlesen und das Programm hat nicht funktioniert. Als das Problem gelöst worden ist, konnte weiter am Programm gearbeitet werden.

Der nächste große Schritt ist nur eine bestimmte Anzahl der LEDs in verschiedenen Farben leuchten zu lassen.

Danach ist es wichtig gewesen sich zu überlegen, wie man die lineare Anordnung der LEDs auf eine zweidimensionale Ebene projiziert. Da der LED-Streifen in kleinere Streifen geteilt worden ist, die untereinander nicht immer gleich groß sind, muss man sich eigene Funktionen ausdenken, die die einzelnen Streifen und dort die einzelnen LEDs einen Koordinaten X- und Y-Wert möglichst einfach und schnell zuordnet, damit die 128 LEDs nicht selbst per Hand eingeben müssen.

### Testen des RGB-LED-Streifens

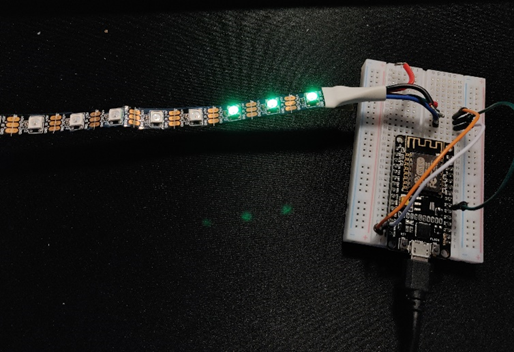
Zuerst ist getestet worden, ob alle RGB-LEDs funktionieren. Zur Ansteuerung des LED-Streifens verwenden wir die „Adafruit\_NeoPixel“-Bibliothek. Darin sind viele wichtige Funktionen enthalten, die uns die Programmierung erleichtern.

Als Erstes müssen die notwendigen Bibliotheken eingebunden werden. Verwendet werden können die Bibliotheken, wenn sie mit „#include <…>“ hinzugefügt werden. „#ifdef …“ ist eine Präprozessoranweisung. Falls ein Makro namens „\_\_AVR\_\_“ bekannt sein sollte, wird der Inhalt „<avr/power.h>“ hinzugefügt. Die „#ifdef“-Anweisung muss mit einem „#endif“ beendet werden. Mit dem „#define“ werden datentyplose Makros festgelegt, welche zur Unterscheidung nur mit Großbuchstaben geschrieben werden. Bei einem „#define“ wird zuerst immer der Name definiert und dann wird ihm erst in derselben Zeile ein Wert zugeschrieben. „pixels“ ist eine Instanz, oder auch ein Objekt von der Klasse „Adafruit\_NeoPixel“. Es beinhaltet Parameter, die in den geschwungenen Klammern angegeben sind.

* Parameter 1: Anzahl der RGB-LEDs
* Parameter 2: Datenpin des Mikrocontrollers
* Parameter 3: Art der Übertragung der RGB-Farben und Datenübertragung in Kilohertz

Der „DEALYVAL“ ist ein Verzögerungswert. In der „void setup()“-Funktion werden die Dinge festgelegt, die im Programmcode nur einmal definiert werden müssen.

Die „void loop“-Funktion ist der Teil des Codes, der seinen Inhalt in einer Endlosschleife wiederholt. In diesem Fall wird immer ein grüne LED mehr eingeschalten mit einer Helligkeit von 150 von 255. Das Ergebnis wird mit „pixels.show()“ und hat eine Pause „DEALYVAL“ von 500 Millisekunden.



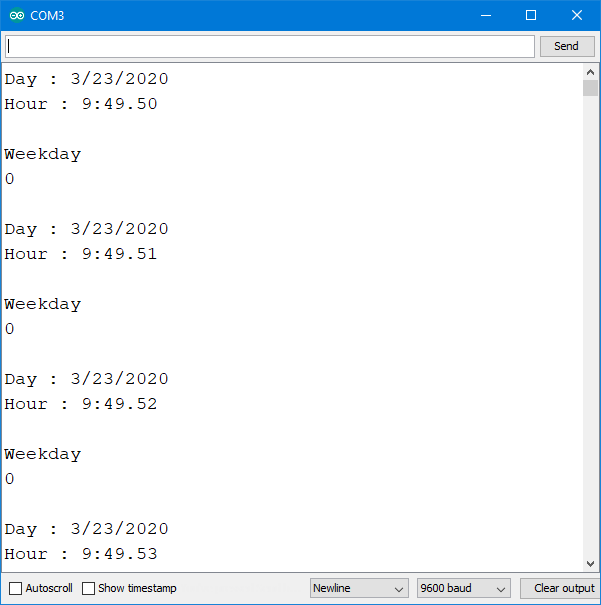
|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | // NeoPixel Ring simple sketch (c) 2013 Shae Erisson  // Released under the GPLv3 license to match the rest of the  // Adafruit NeoPixel library  #include <Adafruit\_NeoPixel.h>  #ifdef \_\_AVR\_\_  #include <avr/power.h> // Required for 16 MHz Adafruit Trinket  #endif  // Which pin on the Arduino is connected to the NeoPixels?  #define PIN D6 // On Trinket or Gemma, suggest changing this to 1  // How many NeoPixels are attached to the Arduino?  #define NUMPIXELS 128 // Popular NeoPixel ring size  // When setting up the NeoPixel library, we tell it how many pixels,  // and which pin to use to send signals. Note that for older NeoPixel  // strips you might need to change the third parameter -- see the  // strandtest example for more information on possible values.  Adafruit\_NeoPixel pixels(NUMPIXELS, PIN, NEO\_GRB + NEO\_KHZ800);  #define DELAYVAL 500 // Time (in milliseconds) to pause between pixels  void setup() {  // These lines are specifically to support the Adafruit Trinket 5V 16 MHz.  // Any other board, you can remove this part (but no harm leaving it):  #if defined(\_\_AVR\_ATtiny85\_\_) && (F\_CPU == 16000000)  clock\_prescale\_set(clock\_div\_1);  #endif  // END of Trinket-specific code.  pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)  }  void loop() {  pixels.clear(); // Set all pixel colors to 'off'  // The first NeoPixel in a strand is #0, second is 1, all the way up  // to the count of pixels minus one.  for(int i=0; i<NUMPIXELS; i++) { // For each pixel...  // pixels.Color() takes RGB values, from 0,0,0 up to 255,255,255  // Here we're using a moderately bright green color:  pixels.setPixelColor(i, pixels.Color(0, 150, 0));  pixels.setBrightness(150);  pixels.show(); // Send the updated pixel colors to the hardware.  delay(DELAYVAL); // Pause before next pass through loop  }  } |

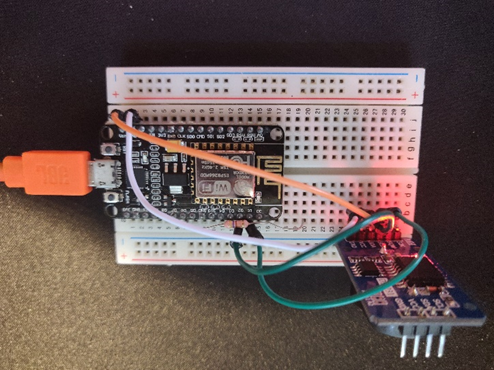
### Einstellen der RTC

Der nächste Schritt wird sein, dass die RTC richtig eingestellt wird. Damit wir die Real Time Clock richtig einstellen können, haben wir uns die „ds3231.h“- und „Wire.h“-Bibliothek zur Hilfe genommen. Die „Wire.h“-Bibliothek dient zur Übertragung der Daten über den „I2C-Bus“. Mit der „ds3231.h“ kann eingestellt werden:

* Uhrzeit (hh/min min/sec sec)
* Datum (dd/mm)
* Wochentag
* Jahrestag
* Monatstag
* Jahr

Zuerst wird ein Objekt „t“ eines „structs ts“ erstellt. Darin sind die ganzen Daten enthalten, um die einzelnen Zeitfunktionen aufzurufen. In der „void setup()“-Funktion wird zuerst die Übertragung über den „I2C-Bus“ mit dem Mikrocontroller der RTC gestartet. Daraufhin wird die richtige RTC eingestellt. Die Baudrate für eine serielle Datenübertragung wird auf 9600 Bd/s gesetzt. Danach werden den einzelnen Parametern des „structs ts“ die aktuellen Zeitwerte zugewiesen. Wenn alle Werte richtig gesetzt worden sind, können mithilfe des „DS3231\_set(t)“-Befehls alle Parameter überschrieben werden. In der „void loop()“-Funktion werden dann die Daten der RTC mit dem Befehl „DS3231\_get(&t)“ ausgelesen. Das Ergebnis kann daraufhin über den seriellen Monitor mit dem Befehl „Serial.print(“…“)“ in der Arduino IDE ausgegeben werden.

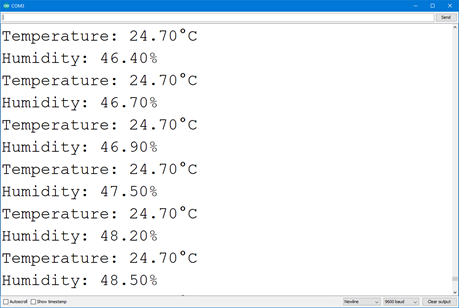


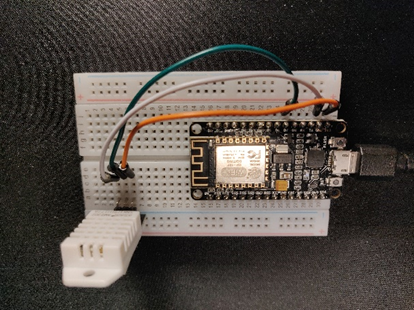


|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | #include <Wire.h>  #include <ds3231.h>  ts t; //ts is a struct findable in ds3231.h  void setup() {  Wire.begin(); //start i2c (required for connection)  DS3231\_init(DS3231\_INTCN); //register the ds3231 (DS3231\_INTCN is the default address of ds3231, this is set by macro for no performance loss)  Serial.begin(9600);  t.year = 2020;  t.mon = 2;  t.mday = 16;  t.hour = 19;  t.min = 11;  t.sec = 43;  t.wday = 7;  DS3231\_set(t);  }  void loop() {  DS3231\_get(&t); //get the value and pass to the function the pointer to t, that make an lower memory fingerprint and faster execution than use return  //DS3231\_get() will use the pointer of t to directly change t value (faster, lower memory used)  Serial.print("Day : ");  Serial.print(t.mon); //just print some data  Serial.print("/");  Serial.print(t.mday);  Serial.print("/");  Serial.println(t.year);  Serial.print("Hour : ");  Serial.print(t.hour);  Serial.print(":");  Serial.print(t.min);  Serial.print(".");  Serial.println(t.sec);  Serial.println("\nWeekday");  Serial.println(t.wday);  Serial.println();  delay(1000);  } |

### Testen des DHT22

Als Nächstes ist es wichtig zu testen, ob der Temperatur- und Luftfeuchtigkeitssensor richtig funktioniert. Zuerst werden die notwendigen Bibliotheken eingebunden, um eine Kommunikation mit dem Mikrocontroller aufzubauen. Mit dem „#define DHTPIN“ wird der zur Übertragung notwendige Pin am Mikrocontroller angegeben. Dadurch dass die Bibliothek verschiedene Versionen des DHT zur Verfügung stellt, muss mit „#define DHT-TYPE“ der richtige Sensor ausgewählt werden. Danach wird eine Instanz der Klasse „DHT\_Unified“ erstellt. In der „void setup()“-Funktion wird zuerst die die Baudrate der seriellen Kommunikation eingestellt. Danach wird der die Übertragung des DHT gestartet. Mit „dht.temperature().getSensor(&sensor)“ und „dht.humidity().getSensor(&sensor)“ werden die Daten aus dem Sensor geholt. Danach wird die Wartezeit für das Lesen des Sensors berechnet. Es wird eine Variable „event“ vom Typ „sensors\_event\_t“ erstellt. Es wird immer zuerst mit „isnan“ sichergestellt, dass man durch das „event“ einen Zahlenwert zurückbekommt. Wenn man keinen Zahlenwert zurückbekommt, wird ein Error ausgeben und im anderen Fall der Wert, den der Sensor liefert.





|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | #include <Adafruit\_Sensor.h>  #include <DHT.h>  #include <DHT\_U.h>  #define DHTPIN D7 // Digital pin connected to the DHT sensor  // Uncomment the type of sensor in use:  //#define DHTTYPE DHT11 // DHT 11  #define DHTTYPE DHT22 // DHT 22 (AM2302)  //#define DHTTYPE DHT21 // DHT 21 (AM2301)  // See guide for details on sensor wiring and usage:  // https://learn.adafruit.com/dht/overview  DHT\_Unified dht(DHTPIN, DHTTYPE);  uint32\_t delayMS;  void setup() {  Serial.begin(9600);  // Initialize device.  dht.begin();  // Print temperature sensor details.  sensor\_t sensor;  dht.temperature().getSensor(&sensor);  // Print humidity sensor details.  dht.humidity().getSensor(&sensor);  // Set delay between sensor readings based on sensor details.  delayMS = sensor.min\_delay / 1000;  }  void loop() {  // Delay between measurements.  delay(delayMS);  // Get temperature event and print its value.  sensors\_event\_t event;  dht.temperature().getEvent(&event);  if (isnan(event.temperature)) {  Serial.println(F("Error reading temperature!"));  }  else {  Serial.print(F("Temperature: "));  Serial.print(event.temperature);  Serial.println(F("°C"));  }  // Get humidity event and print its value.  dht.humidity().getEvent(&event);  if (isnan(event.relative\_humidity)) {  Serial.println(F("Error reading humidity!"));  }  else {  Serial.print(F("Humidity: "));  Serial.print(event.relative\_humidity);  Serial.println(F("%"));  }  } |

### Erstellen einer geeigneten Matrix

Der erste Schritt ist es eine Matrix zu erstellen, womit die RGB-Farben verändert werden können. Dazu haben wir einen „struct RGB“ erstellt. Ein „Struct“ vereint dabei mehrere Datentypen miteinander. Der nächste Schritt ist es eine geeignete Matrix für die Anordnung der RGB-LED-Streifen zu finden. Dazu wird zunächst den einzelnen RGB-LEDs eine x- und y-Koordinate zugewiesen. Dadurch, dass die LED-Streifen abwechselnd von oben und unten beginnen, muss jede zweite Reihe invertiert werden. Für alle LED-Streifen, die von oben nach unten gehen und 7 RGB-LEDs besitzen, wird die Funktion „void copyRow“ verwendet und für die, die von unten nach oben verlaufen wird die Funktion „void copyInvRow“ verwendet. Hinzu kommt, dass durch die spezielle Bauform die ersten zwei und die letzte Reihe unterschiedlich viele RGB-LEDs ausweisen. Für diese speziellen Reihen ist dann eine eigene Funktion erstellt worden.

* erste Reihe: „void rowZero()“
* zweite Reihe: „void rowOne()“
* letzte Reihe: „void lastRow()“

Um die gesamte Matrix darzustellen, werden die einzelnen Funktionen in der „void displ()“ zusammengefasst.

Damit die einzelnen Pixel wieder überschieben werden können, wird die Funktion „void overrideCoroPixels(bool copy\_black)“ verwendet. Wenn „copy\_black“ auf „true“ gesetzt wird, werden die Pixel überschrieben, ansonsten werden sie nur verändert, wenn sie gesetzt werden.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | struct RGB {  unsigned char r;  unsigned char g;  unsigned char b;  };  RGB matrix[MATRIX\_DIMENSION\_X][MATRIX\_DIMENSION\_Y]; ///< x is 0 to 19, y is 0 to 6  void overrideColorPixels(bool copy\_black, uint16\_t n, uint8\_t r, uint8\_t g, uint8\_t b) {  if (copy\_black) {  pixels.setPixelColor(n, r, g, b);  }  else {  if (r > 0 && b > 0 && g > 0) {  pixels.setPixelColor(n, r, g, b);  }  }  }  void copyRow(short row\_nmb, bool copy\_black) {  int startpix;  // 2 --> 6  // 4 --> 20  // 6 --> 34  startpix = (row\_nmb - 2) \* 7 + 6;  for (int i = 0; i <= 6; i++) {  overrideColorPixels(copy\_black, startpix + i, matrix[row\_nmb][i].r,  matrix[row\_nmb][i].g, matrix[row\_nmb][i].b);  }  }  void displ(bool copy\_black) {  for (int i = 0; i <= 19; i++) {  if (i % 2 == 0 && i <= 18 && i >= 2) {  copyRow(i, copy\_black);  }  else if (i % 2 != 0 && i <= 17 && i >= 3) {  copyInvRow(i, copy\_black);  }  else if (i == 0) {  rowZero(copy\_black);  }  else if (i == 1) {  rowOne(copy\_black);  }  else if (i == 19) {  lastRow(copy\_black);  }  }  } |

### Testen der Matrix

In diesem Schritt geht es darum, zu testen ob die Matrix die richtigen LEDs in der richtigen Farbe zum Leuchten bringt.

In der „void fillMatrix()“-Funktion wird mit Hilfe einer „for“-Schleife die gesamte Matrix auf die „r, g, b“-Farben gesetzt. Um zu wissen wo die entsprechenden LEDs wirklich liegen wird dann die Funktion „displ(true)“ aufgerufen. Damit am Ende die LEDs auch leuchten wird mit Hilfe der Funktion „pixels.show()“ die entsprechende Farbe an der entsprechenden LED angezeigt.

In der Funktion „void testMatrix()“ wird getestet, ob sich einzelne Segmente, die sich in der x- und y-Achse befinden, beliebig verändert werden können ohne dass Fehler auftreten. Dazu werden in den ersten zwei „for“-Schleifen die einzelnen Spalten eingefärbt und danach werden die Reihen in den nächsten zwei „for“-Schleifen umgefärbt.

Die Funktion „void writeRainbowToMatrix()“ soll einen Regenbogeneffekt auf die Matrix setzten. Dabei wird ein Laufparameter „iteration“ definiert. Mit „wheelpos“ wird die Farbe der RGB-LED berechnet. Dabei ist es immer so, dass eine Farbe ausgeschalten ist und die zwei anderen Farben gegengleich leuchten. Damit ist gemeint, dass die eine Farbe damit beginnt immer stärker zu leuchten und die andere von der maximalen Lichtstärke aus immer schwächer leuchtet. Dadurch kommt dann der typische Regenbogeneffekt zustande.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | void fillMatrix(unsigned char r, unsigned char g, unsigned char b) {  for (int x = 0; x < 20; x++) {  for (int y = 0; y < 7; y++) {  matrix[x][y] = { r, g, b };  }  }  displ(true); pixels.show();  }  void testMatrix() {  for (int x = 0; x < 20; x++) {  for (int y = 0; y < 7; y++) {  matrix[x][y] = { 0, 100, 100 };  }  displ(true); pixels.show(); delay(500);  }  delay(1000);  for (int y = 0; y < 7; y++) {  for (int x = 0; x < 20; x++) {  matrix[x][y] = { 100, 0, 0 };  }  displ(true);  pixels.show();  delay(500);  }  }  void writeRainbowToMatrix(unsigned char iteration) {  unsigned char wheelPos; ///< variable used to calculate the colors  unsigned char r; ///< red color value  unsigned char g; ///< green color value  unsigned char b; ///< blue color value  // iterate over all x and y values of the matrix  for (int x = 0; x < MATRIX\_DIMENSION\_X; x++) {  for (int y = 0; y < MATRIX\_DIMENSION\_Y; y++) {  unsigned char cycleCount = iteration;  wheelPos = (cycleCount + y + x \* MATRIX\_DIMENSION\_Y) & 255;  if (wheelPos < 85) {  r = wheelPos \* RAINBOW\_MULTIPLIER;  g = MAX\_RAINBOW\_COLOR - wheelPos \* RAINBOW\_MULTIPLIER;  b = 0;  }  else if (wheelPos < 170) {  wheelPos -= 85;  r = MAX\_RAINBOW\_COLOR - wheelPos \* RAINBOW\_MULTIPLIER;  g = 0;  b = wheelPos \* RAINBOW\_MULTIPLIER;  }  else {  wheelPos -= 170;  r = 0;  g = wheelPos \* RAINBOW\_MULTIPLIER;  b = MAX\_RAINBOW\_COLOR - wheelPos \* RAINBOW\_MULTIPLIER;  }  matrix[x][y] = { r, g, b };  }  }  } |

### Font

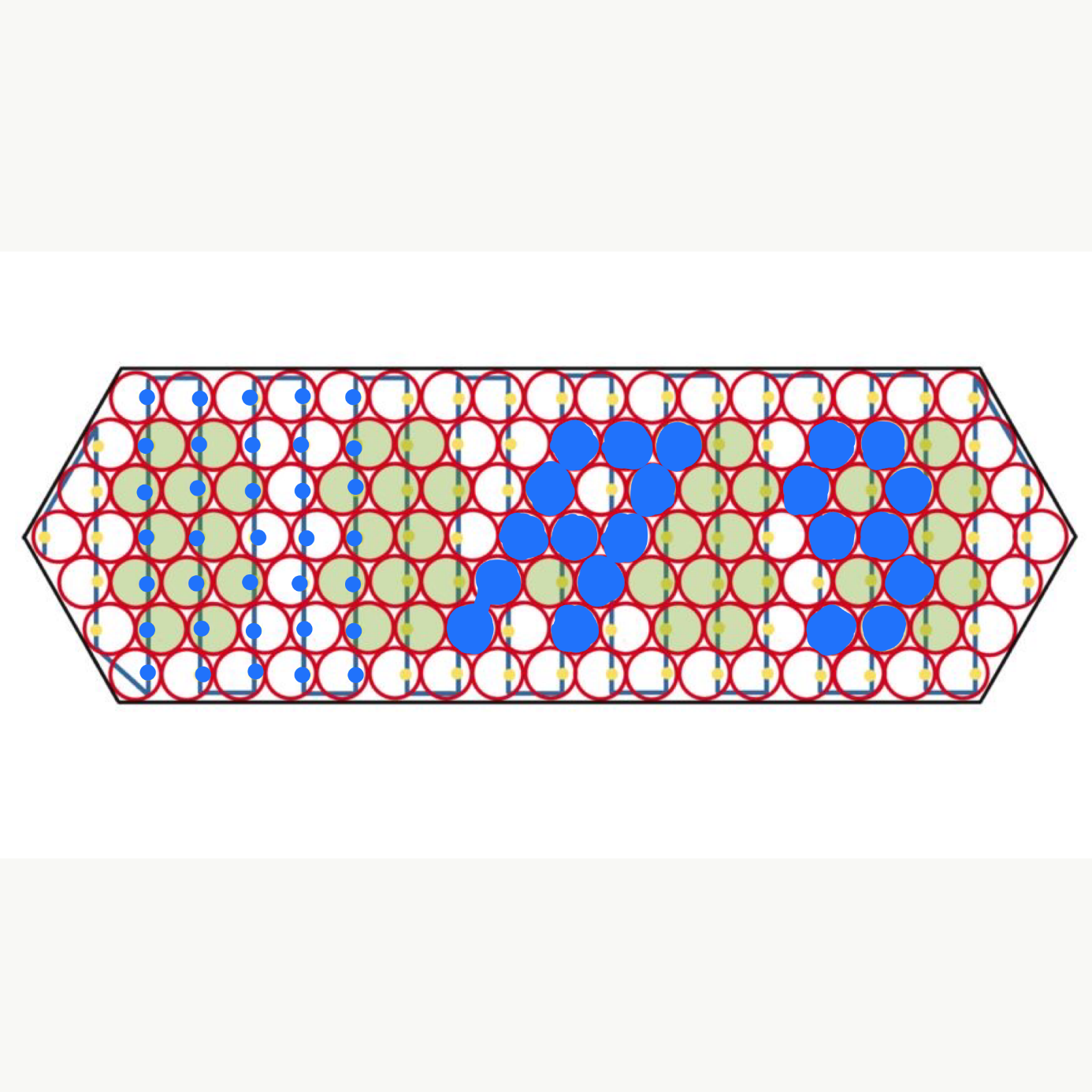


Abbildung 5.3.6‑1:Matrix und Font

Für die Darstellung der Buchstaben, Zahlen und Sonderzeichen ist eine 7x5 Matrix erstellt worden, wobei die erste und letzte Zeile nie gesetzt werden. Die Buchstaben sind meistens kursiv dargestellt, da dies auf unserem Smart-Display gut funktioniert. Die Zahlen sind hingegen zu den Buchstaben gerade dargestellt und die Sonderzeichen sind an beide Arten angepasst worden. Für die Darstellung der Zeichen im Code beginnt man links unten Spalte für Spalte hinauf zu zählen. Zuerst wird einzeln in den Spalten geschaut, welches Bit gesetzt werden muss und anschließend wird in der Tabelle des Dualsystems nachgeschaut, welche Hexadezimalzahl mit den gesetzten Bits herauskommt. Anschließend wird der Font im Programm in ein Zweidimensionales Array geschrieben.

|  |  |  |
| --- | --- | --- |
| **Dezimal** | **Dualsystem** | **Hexadezimal** |
| 00 | 0 0 0 0 | 0 |
| 01 | 0 0 0 1 | 1 |
| 02 | 0 0 1 0 | 2 |
| 03 | 0 0 1 1 | 3 |
| 04 | 0 1 0 0 | 4 |
| 05 | 0 1 0 1 | 5 |
| 06 | 0 1 1 0 | 6 |
| 07 | 0 1 1 1 | 7 |
| 08 | 1 0 0 0 | 8 |
| 09 | 1 0 0 1 | 9 |
| 10 | 1 0 1 0 | A |
| 11 | 1 0 1 1 | B |
| 12 | 1 1 0 0 | C |
| 13 | 1 1 0 1 | D |
| 14 | 1 1 1 0 | E |
| 15 | 1 1 1 1 | F |

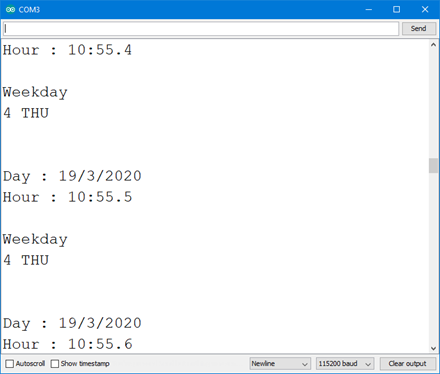
Tabelle 5.3‑1: Stellenwertsystem

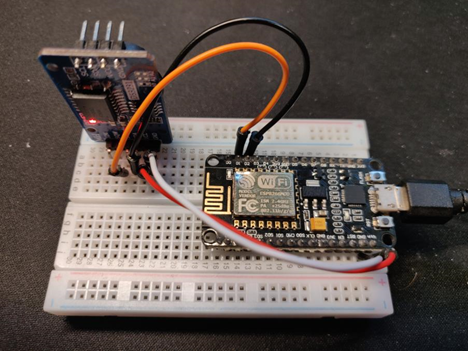
**Beispielfont für die Zeichen „A“ und „9“:**

{0x01,0x06,0x1D,0x16,0x18}, // A Stelle: 0  
{0x1D,0x15,0x0A,0x00,0x00}, // 9 Stelle: 35

### NTP-Server

Da es bei der RTC ab und zu vorkommt, dass sie sich wieder zurücksetzt, haben wir uns dazu entschieden, die Zeit der RTC mit einem NTP Server (Network Time Protocol) zu synchronisieren. In den ersten sechs Zeilen werden die notwendigen Bibliotheken eingebunden. Danach werden zwei Character Pointer mit den Namen „\*\_ssid“ und „\*\_password“ erstellt. Diese dienen dazu, um sich später mit dem entsprechende WLAN-Netzwerk verbinden zu können. Da es verschiedene Zeitzonen gibt muss es auch dementsprechend Zeitverschiebungen in anderen Gebieten geben. Bei uns entspricht diese Verschiebung zwei Stunden. In der elften Zeile wird ein zweidimensionales Character Array erstellt, um den entsprechenden Wochentag anzugeben. Danach wird mit der Klasse „WiFiUDP“ die Art der Verbindung als UDP-Verbindung festgelegt. Daraufhin erstellen wir uns eine Instanz der Klasse „NTPClient“, welche die Art der Übertragung, den Namen vom NTP-Server und die Zeitverschiebung in Sekunden als Parameter übergeben bekommt. Mit „WiFi.begin“ wird dann die Verbindung zum Netzwerk aufgebaut. Sobald das ESP mit dem WLAN-Netzwerk verbunden ist, wird der NTP-Client gestartet und aktualisiert. Die Daten vom „timeClient“ werden danach mit der RTC synchronisiert und danach in der „void loop()“-Schleife ausgegeben.





|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | #include <Wire.h>  #include <ds3231.h>  #include <NTPClient.h>  #include <ESP8266WiFi.h>  #include <WiFiUdp.h>  const char \*\_ssid = "\*\*\*\*\*\*\*\*";  const char \*\_password = "\*\*\*\*\*\*\*";  const long \_utcOffsetInSeconds = 3600 \* 2;  char \_wday[7][4] = {"SUN","MON","TUE", "WED", "THU", "FRI", "SAT" };  WiFiUDP \_ntpUDP;  NTPClient timeClient(\_ntpUDP, "0.at.pool.ntp.org",  \_utcOffsetInSeconds);  ts t;  void setup() {  Wire.begin(); //start i2c (required for connection)  DS3231\_init(DS3231\_INTCN); //register the ds3231 (DS3231\_INTCN  Serial.begin(115200);  WiFi.begin(\_ssid, \_password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  timeClient.begin();  timeClient.update();  t.hour = (uint8\_t)timeClient.getHours();  t.min = (uint8\_t)timeClient.getMinutes();  t.sec = (uint8\_t)timeClient.getSeconds();  t.mon = (uint8\_t)timeClient.getMonth();  t.mday = (uint8\_t)timeClient.getDate();  t.year = (uint8\_t)timeClient.getYear();  t.wday = (uint8\_t)timeClient.getDay();  DS3231\_set(t);  timeClient.end();  }  void loop() {  DS3231\_get(&t);  Serial.printf("Day : %d/%d/%d ",t.mday, t.mon, t.year);  Serial.printf("Hour : %d:%d.%d", t.hour, t.min, t.sec);  Serial.println("\nWeekday");  Serial.print(t.wday);  Serial.printf(" %s\n\n", \_wday[t.wday]);  Serial.println();  delay(1000);  } |

### Programmiersprache C

Zunächst ist es wichtig zu erläutern, was eine Programmiersprache ist. Eine Programmiersprache dient der Eingabe und Definierung von Abläufen, die abgearbeitet und ausgeführt werden sollen. Am Beginn der Entwicklung der Programmiersprachen haben sie sich wegen der Hardwarelimitierungen stark auf die Eigenschaften der verwendeten Komponenten bezogen.[[9]](#footnote-9)

Heute gibt es hunderte Programmiersprachen, davon sind aber nur eine Handvoll relevant und einigermaßen weit verbreitet. Als die Programmierung noch in den Kinderschuhen gesteckt hat, sind fast ausschließlich die Maschinen- oder Assemblersprache verwendet worden. Als die Betriebssysteme entwickelt und verbessert worden sind, haben sich Türen für eine neue Art des Programmierens gebildet. Durch sogenannte Compiler (Übersetzer) oder auch Interpreter ist es möglich gewesen fernab der Maschinen- und Assemblersprachen Programme zu schreiben. Man unterteilt Programmiersprachen heute in 3 verschiedenen Gruppen:

1. Maschinensprache
2. Assemblersprache
3. Problemorientierte höhere Programmiersprache[[10]](#footnote-10)

# Diskussion

## Reflexion der Projektrealisierung

### Probleme bei der Platinenfertigung

Das erste Problem, welches aufgetreten ist, ist das während des Anfertigens des Layouts der Anschluss der Rückführung des RGB-LED-Streifens den Weg des Micro-USB-Kabels zum NodeMCU ESP8266 versperrt hat, dieser Fehler wurde aber erst nach der Fertigung der Platine bemerkt. Weiters wurde der Datenanschluss mit dem Versorgungsanschluss des DHT22 vertauscht, genauso wie der erste Fehler wurde auch dieser erst nach der Fertigung der Platine bemerkt.

Nach der Korrektur dieser Fehler im Layout, ist die Platine das zweite Mal gefertigt worden, wobei dieses Mal die Platine zu lange im Ätzbad gewesen ist und der größte Teil des Kupfers sich gelöst hat, davon sind auch die Leitungen betroffen gewesen.

# Literaturverzeichnis

*DS3231 RTC Real Time Clock Datenblatt.* (28. März 2020). Von AZ-Delivery: https://cdn.shopify.com/s/files/1/1509/1638/files/DS3231\_RTC\_Real\_Time\_Clock\_Datenblatt.pdf?12002561494986697130 abgerufen

*ESP8266*. (30. März 2020). Von Wikipedia - Die freie Enzyklopädie: https://de.wikipedia.org/wiki/ESP8266 abgerufen

*Kondensatoren*. (1. April 2020). Von Elektronik Kompendium: https://www.elektronik-kompendium.de/sites/bau/0205141.htm abgerufen

*LM1117 800-mA Low-Dropout Linear Regulator.* (24. März 2020). Von Texas Instruments: http://www.ti.com/lit/ds/symlink/lm1117.pdf abgerufen

*NodeMCU*. (30. März 2020). Von Wikipedia - Die freie Enzyklopädie: https://de.wikipedia.org/wiki/NodeMCU abgerufen

*SEN-DHT22 - Temperatur- und Feuchtigkeitssensor.* (29. März 2020). Von Joy-IT: https://cdn-reichelt.de/documents/datenblatt/A300/DEBO\_SENS\_DHT22\_DB-DE.pdf abgerufen

# Quellen der Grafiken

# Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ort, am TT.MM.JJJJ Vor-/Zuname, Unterschrift

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Vor-/Zuname, Unterschrift

1. vgl. NodeMCU. <https://de.wikipedia.org/wiki/NodeMCU> [Zugriff: 30.03.2020] [↑](#footnote-ref-1)
2. vgl. ESP8266. <https://de.wikipedia.org/wiki/ESP8266> [Zugriff: 30.03.2020] [↑](#footnote-ref-2)
3. vgl. Wie funktioniert eine Ws2812b LED?. <https://www.zedfy.com/led-blog/wie-funktioniert-eine-ws2812b-led> [Zugriff: 02.04.2020] [↑](#footnote-ref-3)
4. vgl. DS3231 RTC Real Time Clock Datenblatt. <https://cdn.shopify.com/s/files/1/1509/1638/files/DS3231_RTC_Real_Time_Clock_Datenblatt.pdf?1200256149498669713> [Zugriff: 28.03.2020] [↑](#footnote-ref-4)
5. vgl. SEN-DHT22. Temperatur- und Feuchtigkeitssensor. <https://cdn-reichelt.de/documents/datenblatt/A300/DEBO_SENS_DHT22_DB-DE.pdf> [Zugriff: 29.03.2020] [↑](#footnote-ref-5)
6. vgl. LM1117 800-mA Low-Dropout Linear Regulator. <http://www.ti.com/lit/ds/symlink/lm1117.pdf> [Zugriff: 24.03.2020] [↑](#footnote-ref-6)
7. vgl. LM1117 800-mA Low-Dropout Linear Regulator. <http://www.ti.com/lit/ds/symlink/lm1117.pdf> [Zugriff: 24.03.2020] [↑](#footnote-ref-7)
8. vgl. Kondensatoren. <https://www.elektronik-kompendium.de/sites/bau/0205141.htm> [Zugriff: 01.04.2020] [↑](#footnote-ref-8)
9. vgl. Programmiersprache. <https://de.wikipedia.org/wiki/Programmiersprache> [Zugriff:] [↑](#footnote-ref-9)
10. vgl. Die Entwicklung der heutigen Programmiersprachen. <https://www.sachsen.schule/~ti/ti85/Seiten/pr_sprachen.html> [Zugriff:] [↑](#footnote-ref-10)