Eieruhr - Montage Anleitung

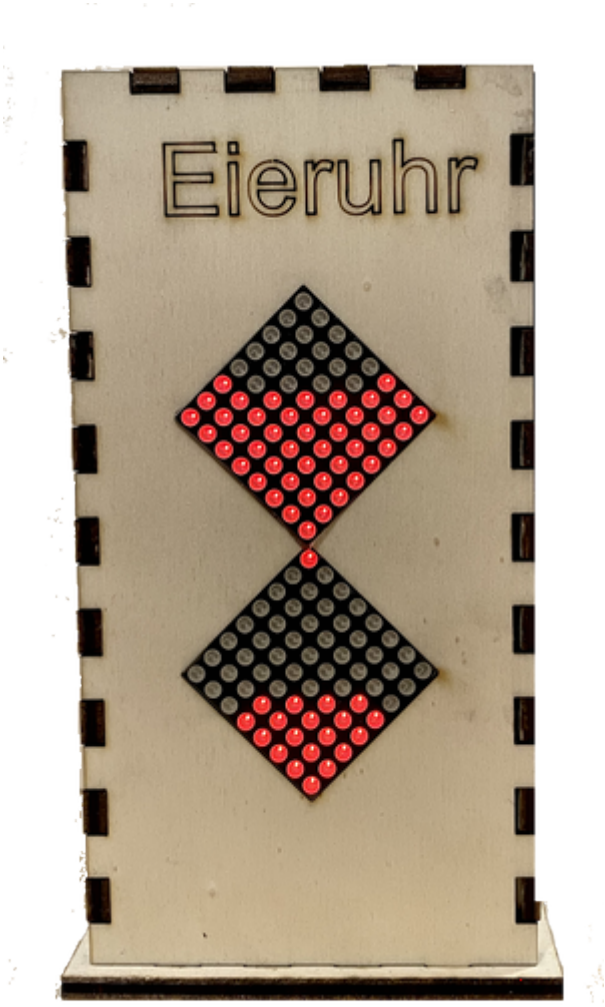
Mit diesem Projekt wird eine Eieruhr mit zwei 64LED-Matrix-Modulen realisiert. Schalter 1 ist der Einschalter und mit Schalter 2 kann zwischen 3 Zeiten gewählt werden, 5 Minuten, 5 Minuten 30 Sekunden und 6 Minuten. Beim Einschalten ertönen, je nach gewählter Zeit, unterschiedliche Signale. Zusätzlich wird bei gewählten 5,5 Minuten zusätzlich ein Signal bei abgelaufenen 5 Minuten, bei gewählten 6 Minuten ein Signal bei 5 und 5,5 Minuten ausgegeben. Diese Zeiten können softwareseitig in nachfolgender Zeile verändert werden:

int pause_5 = 136; int pause_5_5 = 149; int pause_6 = 162;

Die Zeiten für die Zwischensignale in der Zeile:

byte beep_1 = 50; byte beep_2 = 44; byte beep_3 = 50;

Beide Zeilen befinden sich im Deklarationsteil des Sketches.



Die Eieruhr

Hardware

Die Stückliste für die Eieruhr

- 1 x NANO
- 1 x NANO-Shield
- 2 x Buchsenleisten
- 2 x Stiftleisten
- 1 x LM2536
- 2 x 64 LED Matrix Module
- 1 x Buzzer
- 1 x Wechselschalter

- 1 x Schalter 1pol.
- 1 x 9V Batterieclip
- 1 x 9V Batterie
- 1 x Gelasertes Gehäuse (4mm Sperrholz 600x300)

Aufbau und Montage

Der Aufbau und die Montage der Eieruhr ist in mehrere Schritte unterteilt.

Schritt 1: Das Gehäuse

Erstelle auf Boxes.PY eine "BasedBox" mit den Abmessungen x=88, y=88 und h=160.

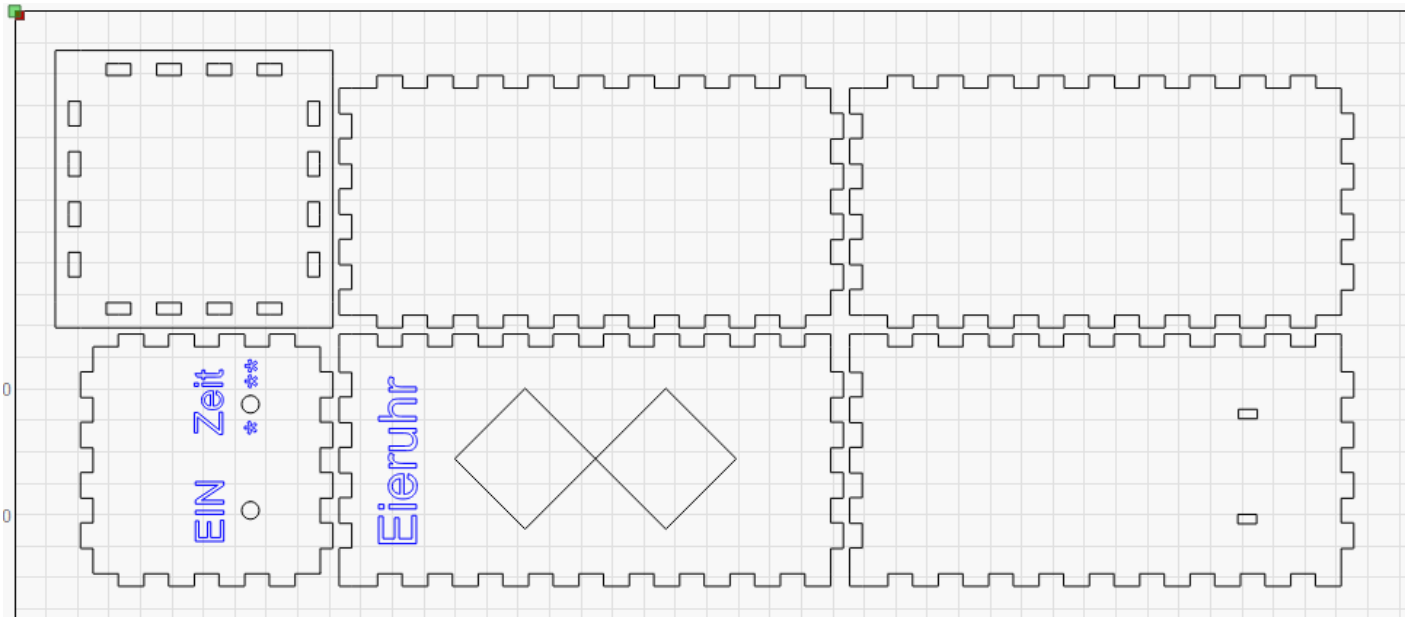
Füge in die erstellte SVG-Grafik die Ausschnitte für die LED Matrix und die Schalter ein

Füge in die erstellte SVG-Grafik den Text "Eieruhr, Ein und Zeit" ein

Schneide das Gehäuse mit dem Lasercutter aus

Montiere die Bauteile an das Gehäuse

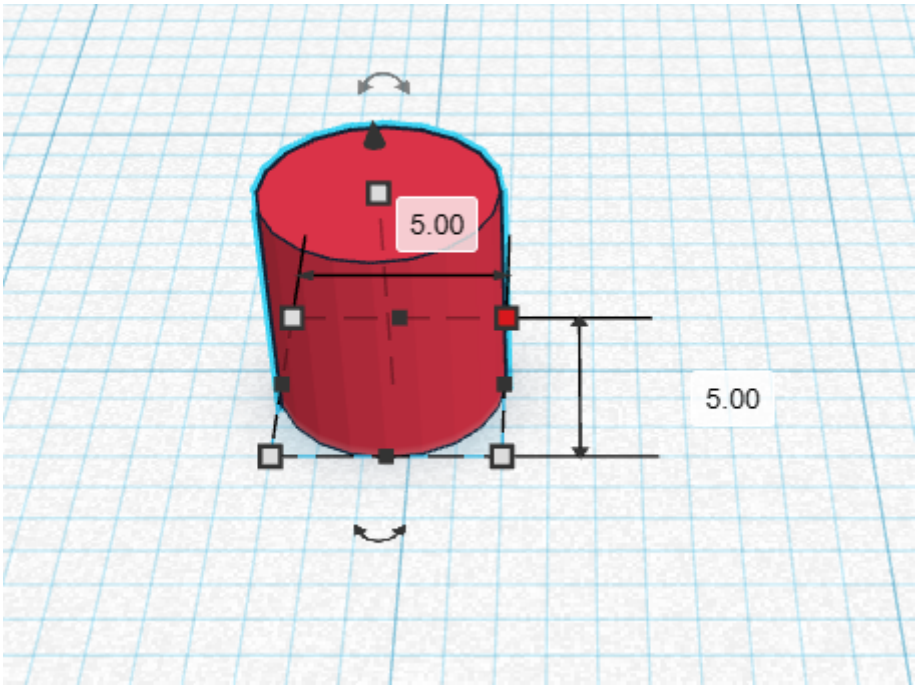
Verklebe die Gehäuseteile. Die untere Rückwand wird nicht verklebt



Schritt 2: 3D-Druck

Zeichne und drucke 8 Platinen Füße für den NANO und das Step-Down-Modul, D=5mm x 5mm

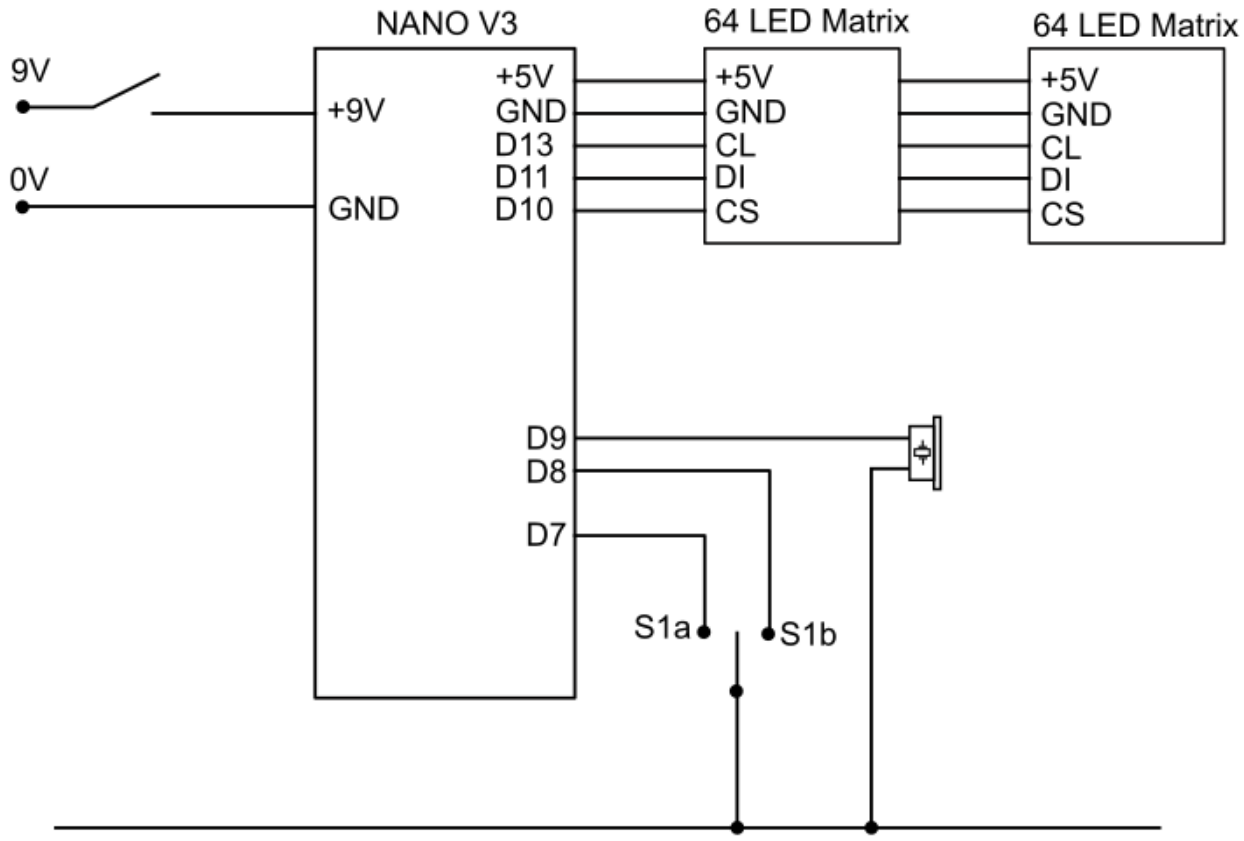
Klebe die Füße auf die Unterseite des NANO-Shields und des Step-Down-Moduls

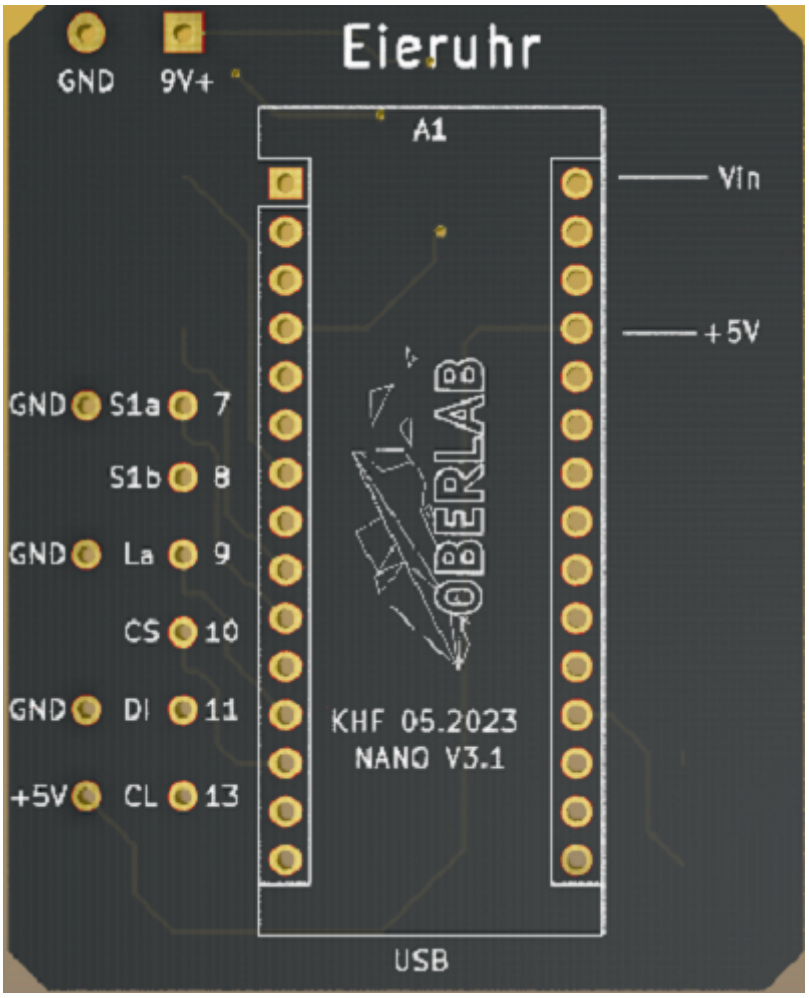


Schritt 3: Verdrahtung und Lötarbeiten

Bestücke das NANO-Shield mit den Buchsenleisten

Verdrahte das Shield, die LED Matrix, Schalter und den Buzzer nach den Verdrahtungsplan


NANO V3
64 LED Matrix
64 LED Matrix
9V
0V
+9V
GND
+5V
GND
D13
D11
D10
+5V
GND
CL
DI
CS
+5V
GND
CL
DI
CS
D9
D8
D7
S1a
S1b



Schritt 4: Inbetriebnahme

- Stelle das Step-Down-Modul auf 5V Ausgangsspannung ein
- Kontrolliere alle Verbindungen

  Lade das LED-Matrix-Test Programm in den NANO

  Versorge die Eieriuhr mit Spannung

  Teste die Funktion

  Wenn die Funktion der LED Matrix ok ist, lade das Haupt-Programm in den NANO

  Das Programm befindet sich auf deinen Laptop unter Nextcloud/Eieruhr

  Teste die Funktion

LED Matrix - Text Programm

```
/*****************************************************************************************
****
                                        PROGRAMMINFO
*****************************************************************************************
****
Funktion: 64-LED-Matrix-Screen-Modul - Testprogramm

*****************************************************************************************
****
Version: 23.02.2022
*****************************************************************************************
****
Board: NANO V3
*****************************************************************************************
****
C++ Arduino IDE V1.8.13
*****************************************************************************************
****
Einstellungen:
https://dl.espressif.com/dl/package_esp32_index.json
http://dan.drown.org/stm32duino/package_STM32duino_index.json
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-
pages/package_esp32_dev_index.json
*****************************************************************************************
****
Librarys
- WiFi.h V0.16.1
*****************************************************************************************
****

 *****************************************************************************************
*****/



#include <MD_MAX72xx.h >
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 1
#define CLK_PIN 13 // or SCK
#define DATA_PIN 11 // or MOSI
#define CS_PIN 10 // or SS
```

```
#define DELAYTIME 100 // in milliseconds
MD_MAX72XX mx = MD_MAX72XX(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);
void scrollText(char *p) {
uint8_t charWidth;
uint8_t cBuf[8];
mx.clear();
while (*p != '\0') {
charWidth = mx.getChar(*p++, sizeof(cBuf) / sizeof(cBuf[0]), cBuf);
for (uint8_t i=0; i<=charWidth; i++) {
mx.transform(MD_MAX72XX::TSL);
if (i < charWidth) {
mx.setColumn(0, cBuf[i]);
}
delay(DELAYTIME);
}
}
}
void rows() {
mx.clear();
for (uint8_t row=0; row<ROW_SIZE; row++) {
mx.setRow(row, 0xff);
delay(2*DELAYTIME);
mx.setRow(row, 0x00);
}
}

void columns() {
mx.clear();
for (uint8_t col=0; col<mx.getColumnCount(); col++) {
mx.setColumn(col, 0xff);
delay(DELAYTIME/MAX_DEVICES);
mx.setColumn(col, 0x00);
}
}
void intensity() {
uint8_t row;
mx.clear();
for (int8_t i=0; i<=MAX_INTENSITY; i++) {
mx.control(MD_MAX72XX::INTENSITY, i);
mx.setRow(0, 0xff);
```

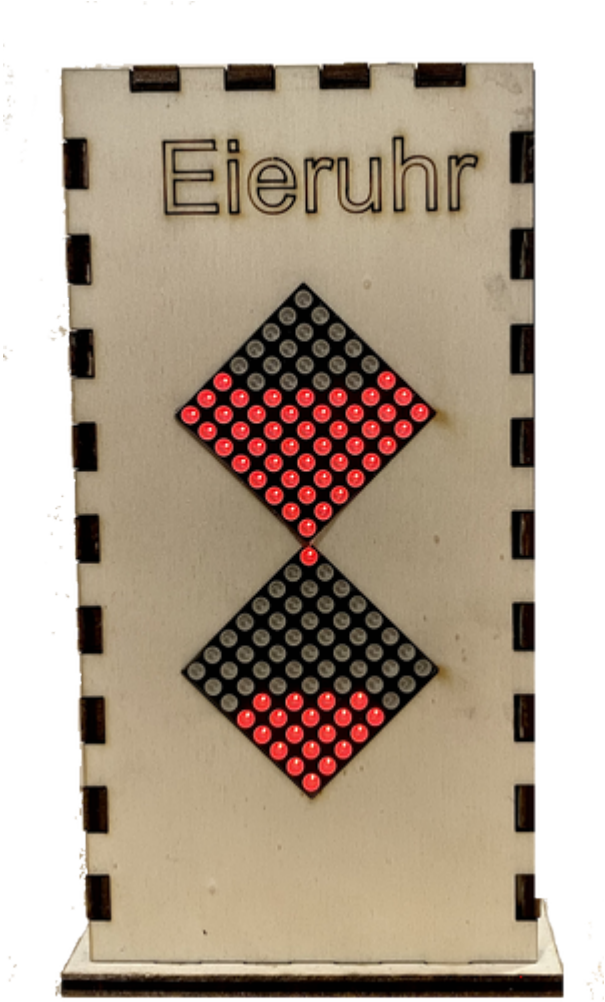
```
delay(DELAYTIME*10);
}
mx.control(MD_MAX72XX::INTENSITY, 8);
}

void transformation() {
uint8_t arrow[COL_SIZE] = {
0b00001000,
0b00011100,
0b00111110,
0b01111111,
0b00011100,
0b00011100,
0b00111110,
0b00000000 };
MD_MAX72XX::transformType_t
t[] = {
MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL,
MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL,
MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL,
MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL,
MD_MAX72XX::TFLR,
MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR,
MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR,
MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR,
MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR,
MD_MAX72XX::TRC,
MD_MAX72XX::TSD, MD_MAX72XX::TSD, MD_MAX72XX::TSD, MD_MAX72XX::TSD,
MD_MAX72XX::TSD, MD_MAX72XX::TSD, MD_MAX72XX::TSD, MD_MAX72XX::TSD,
MD_MAX72XX::TFUD,
MD_MAX72XX::TSU, MD_MAX72XX::TSU, MD_MAX72XX::TSU, MD_MAX72XX::TSU,
MD_MAX72XX::TSU, MD_MAX72XX::TSU, MD_MAX72XX::TSU, MD_MAX72XX::TSU,
MD_MAX72XX::TINV,
MD_MAX72XX::TRC, MD_MAX72XX::TRC, MD_MAX72XX::TRC, MD_MAX72XX::TRC,
MD_MAX72XX::TINV };
mx.clear();
mx.control(MD_MAX72XX::UPDATE, MD_MAX72XX::OFF);
for (uint8_t j=0; j<mx.getDeviceCount(); j++) {
mx.setBuffer(((j+1)*COL_SIZE)-1, COL_SIZE, arrow);
}
```

```
// one tab
mx.control(MD_MAX72XX::UPDATE, MD_MAX72XX::ON);
delay(DELAYTIME);
mx.control(MD_MAX72XX::WRAPAROUND, MD_MAX72XX::ON);
for (uint8_t i=0; i<(sizeof(t)/sizeof(t[0])); i++) {
mx.transform(t[i]);
delay(DELAYTIME*4);
}
mx.control(MD_MAX72XX::WRAPAROUND, MD_MAX72XX::OFF);
}
void showCharset() {
mx.clear();
mx.update(MD_MAX72XX::OFF);
for (uint16_t i=0; i<256; i++) {
mx.clear(0);
mx.setChar(COL_SIZE-1, i);
if (MAX_DEVICES >= 3) {
char hex[3];
sprintf(hex, "%02X", i);
mx.clear(1);
mx.setChar((2*COL_SIZE)-1, hex[1]);
mx.clear(2);
mx.setChar((3*COL_SIZE)-1, hex[0]);
}
mx.update();
delay(DELAYTIME*2);
}
mx.update(MD_MAX72XX::ON);
}

void setup() {
mx.begin();
}
void loop() {
scrollText("Graphics");
rows();
columns();
intensity();
transformation();
```

```
showCharset();
delay(3000);
}
```



Fertig!

Version #1
Erstellt: 2025-03-31 22:19:49 UTC von Joel Hatsch
Zuletzt aktualisiert: 2025-03-31 22:25:53 UTC von Joel Hatsch