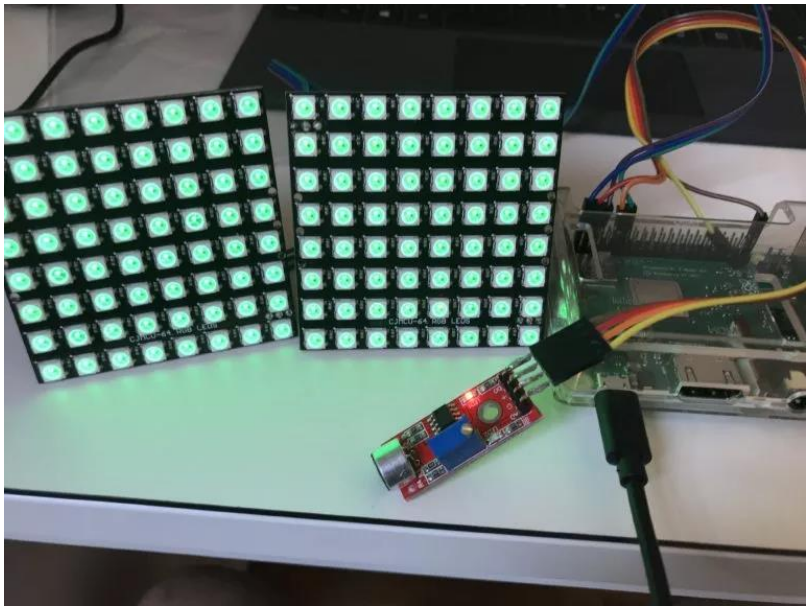
RASPBERRY PI LÄRMAMPEL



MATERIALLISTE

Beim LED-Panel müsst ihr darauf achten, dass diese WS2812 oder WS2812b sind. Diese können wir mit dem Pi direkt ansprechen.

- 1 x Raspberry Pi 3B
- Jumper Kabelbaum
- Netzteil 5V/3000mA
- MicroSD 16GB Karte
- 2 x WS2812 WS2812b LED 5050 RGB 8×8 64 LED Matrix
  oder WS2812B LED Strip Panel Kit Matrix 8×32 256 Pixel Digitales Flexible
- dB-Sensor
- ggf. Debugkabel
- Lötkolben und Zinn
- 10 x Klettkabel-binder (optional)
- ggf. ein kleiner Schrauberzieher oder eine Pinzette

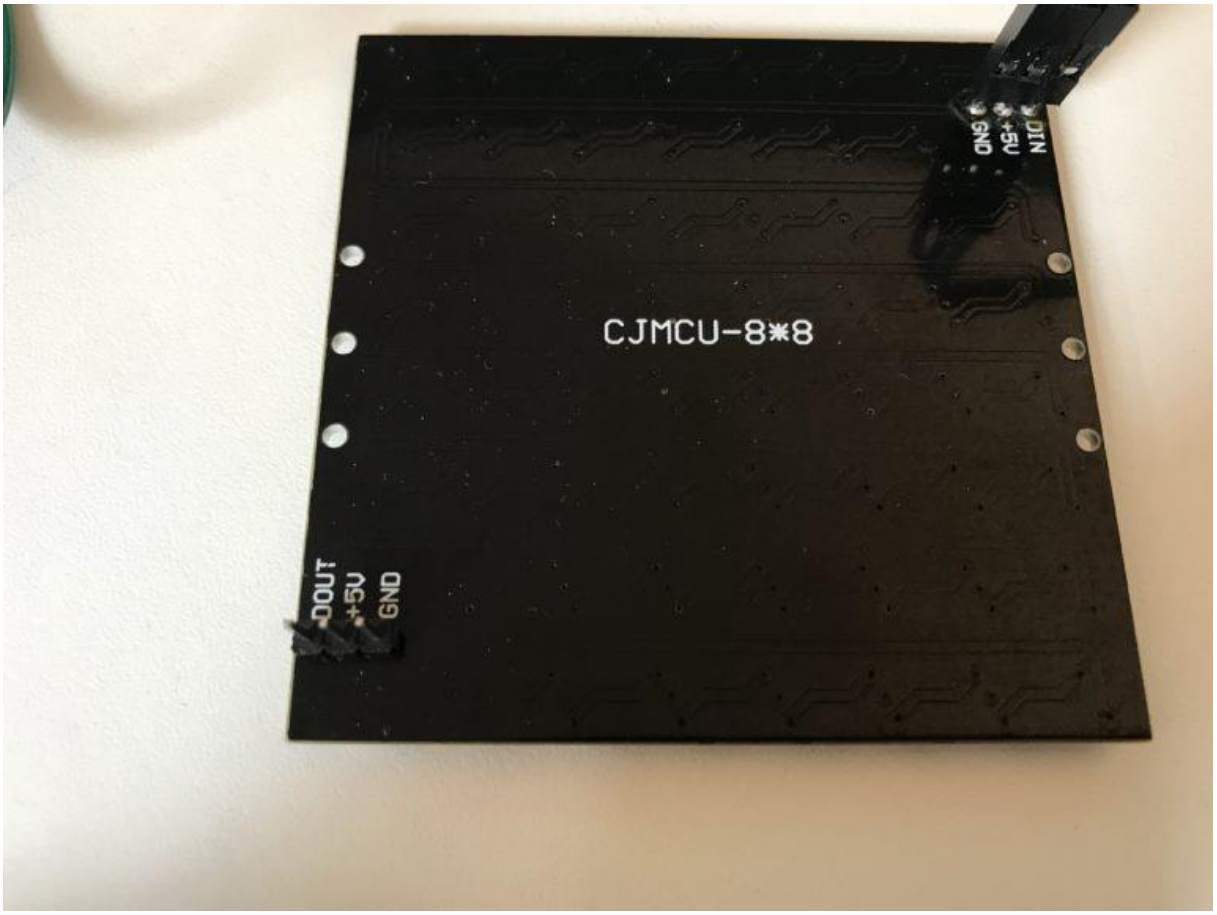
Bevor wir anfangen…

Bevor man die Platinen berührt, sollte man sich erden, um eventuelle Spannungen im Körper abzubauen. Das kannst du tun, indem du beispielsweise kurz den unlackierten Teil einer Heizung berührst. Auch beim Poti wirken Fremd- oder Körperspannungen als Störfaktor, die das einrichten, bzw. einstellen erschweren können. Daher sollte man hier ebenfalls geerdet sein (oder einfach Handschuhe tragen).
Der Poti könnte etwas eigenwillig sein und sich gerne mal von selbst nachjustieren. Das kann aber je nach Modell unterschiedlich sein.

ANLEITUNG

1. ELEMENTE VORBEREITEN

Als erstes bereiten wir die Panels vor. Dafür löten wir die Pins entsprechend an die beiden Panels. Im Lieferumfang sind jeweils 3 Pins vorhanden. Diese verlöten wir entsprechend. Da die Jumper-Kabel auf einer Seite Pins haben, können wir diese entsprechend mit verlöten.



Bei dem Sensor können wir die Pins ebenfalls verlöten. Danach können wir die Jumper-Kabel mit dem Raspberry Pi verbinden. Damit das ganze nachvollziehbar ist, sind alle Jumper-Kabel farblich unterschiedlich. So ist genau zu erkennen, wo das jeweilige Kabel angeschlossen werden muss um die initiale Funktionalität herzustellen.

Sobald alle Adern am Panel als auch am Sensor verlötet sind, schließen wir die Adern entsprechend am Raspberry Pi an.

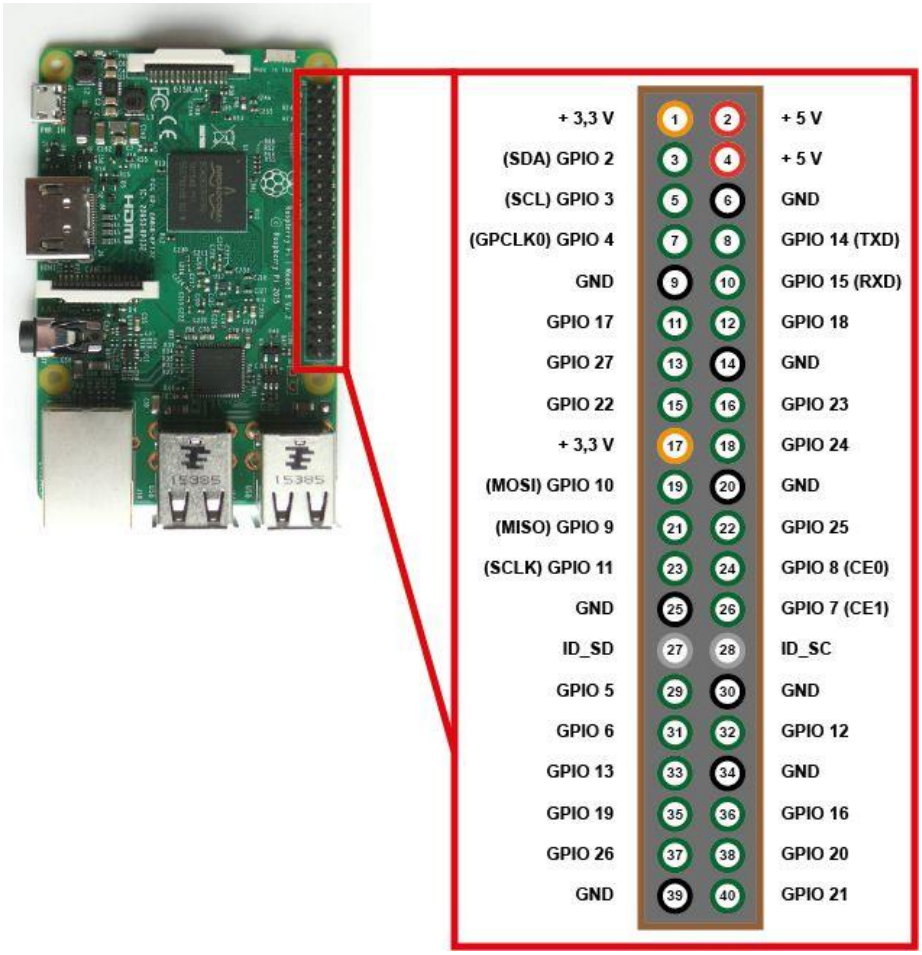
Folgende Pin Belegung habe ich an dieser Stelle gewählt:

Panel:

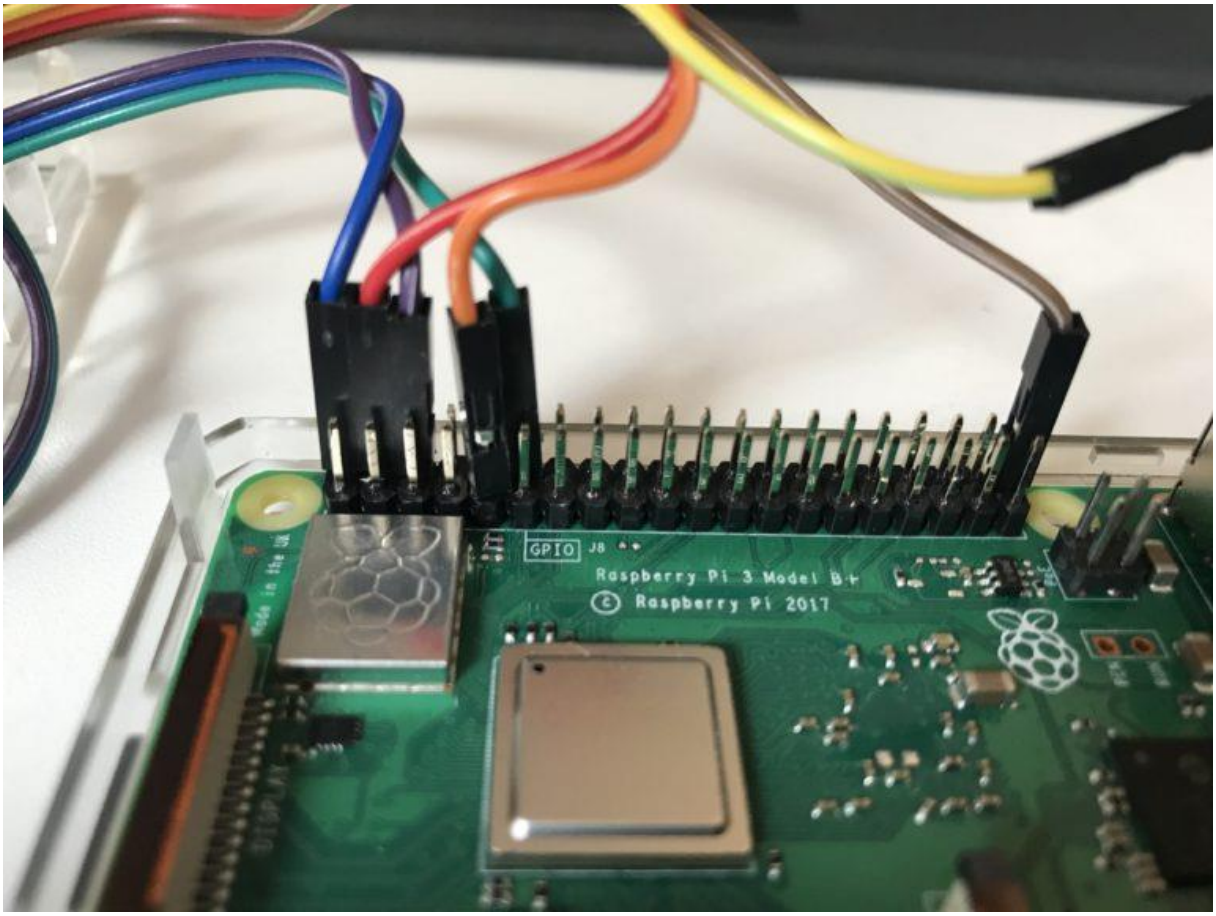
- 2 Pin: 5V Power(Blau) ⟶ + 5V LED Panel No.1 ⟶+ 5V LED Panel No.2
- 6 Pin: Ground (Lila) ⟶ GND LED Panel No.1 ⟶ GND LED Panel No.2
- 12 Pin: GIPO18(Grün) ⟶ DIN LED Panel No.1 ⟶ DOUT ⟶ DIN LED Panel No.2

Sensor:

- 40 Pin: SPI1 SCLK (Braun) ⟶ Do Sensor
- 4 Pin: 5V Power (Rot) ⟶ + Sensor
- 9 Pin: Ground (Orange) ⟶ G Sensor



Die gelbe Ader ist nicht angeschlossen, da dies der analoge Port des Sensors ist und der Raspberry Pi unterstützt keine analogen Signale.
Nachdem nun alles soweit angeschlossen ist, können wir uns dem Betriebssystem und der Software widmen.



2. GRUNDEINRICHTUNG

Als Betriebssystem wird Raspbian Lite (Raspbian Buster Lite) verwendet.
Dies kann man auf der Seite raspberrypi.org downloaden.
Damit wir das System auf die MicroSD Karte schreiben können, benötigen wir das Programm Win32DiskImager, als auch einen Kartenleser. Kopiert das entpackte Image mit dem Programm auf die SD Karte. Nachdem der Vorgang abgeschlossen ist, erscheint die SD Karte als „Boot" auf eurem Arbeitsplatz.
Hier könnt ihr nun die WLAN Daten hinterlegen und den SSH Port freigeben. Erstellt dafür eine leere „ssh" Textdatei. Diese Datei sorgt dafür, dass der Pi beim Booten über den Port 22 erreichbar ist, dieser ist sonst Standardmäßig geschlossen. Um die Daten zu hinterlegen, erstellt ihr eine Textdatei mit dem Namen „wpa_supplicant.conf" und füllt diese mit:

```
1. country=DE
2. ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
3. update_config=1
4. network={
      a. ssid="wlan-bezeichnung"
      b. psk="passwort"
      c. key_mgmt=WPA-PSK
5. }
```

Diese Datei wird dann entsprechend in die WLAN-Config des Pi verschoben und der Pi wählt sich ins WLAN ein.
Nun muss eine SSH-Verbindung zum Port 22 mit PUTTY des Pi`s hergestellt werden.
Den Pi könnt ihr im WLAN durch diverse Port-Scanner ausfindig machen oder euer Router hält die IP Adresse des Pi`s bereit.
Der Standard User ist „pi", das Standard Passwort ist „raspberry".
Damit könnt ihr euch mit dem Pi verbinden und erst einmal eine Grundeinrichtung durchführen.

3. UPDATES UND PAKETE INSTALLIEREN

Damit der Pi auch mit unserem Panel und Sensor interagieren kann, müssen wir erst mal einige Updates machen und nötige Pakete installieren.
Dazu meldet euch zuerst als Root an mit „sudo -i". Dadurch müsst ihr nicht vor jedem Befehl das „sudo" hängen.

```
apt-get update && apt-get upgrade -y && apt-get dist-upgrade -y
```

Nachdem die Updates durchgelaufen sind, können wir den raspbian konfigurieren. Dies machen wir mit dem Befehl „raspi-config"
Erweitert hier die SD Karte, damit ihr die volle Größe nutzen könnt, falls ihr dies später noch für etwas anderes nutzen möchtet.
Stellt auch das Datum und die Zeitzone ein, sowie, das Keyboard Layout.

Sollte der Pi danach ein Reboot benötigen, führt dies aus oder macht mit dem folgenden weiter:

```
apt-get install gcc make build-essential python-dev python-pip wget
git scons swig pigpio screen
```

Dieser Befehl lädt uns die benötigten Module wie z.b. „python-dev" oder „git" runter.
Danach müssen wir unterbinden, dass sein Modul geladen wird.
Dafür öffnen wir die Datei /etc/modprobe.d/snd-blacklist.conf mit dem Befehl

```
nano /etc/modprobe.d/snd-blacklist.conf
```

und fügen am Ende der File dies ein:

```
blacklist snd_bcm2835
```

Da wir an der Stelle PWM verwenden, müssen wir das Soundmodul der ARM CPU deaktivieren. Da es sonst zu Störungen kommen kann.
Zusätzlich müssen wir noch eine Konfig ändern mit:

```
nano /boot/config.txt
```

und fügen dort ein „#" vor die Zeile mit „dtparam=audio=on"

Wir schließen und speichern unsere Änderungen und führen einen „reboot" durch.
Nach dem Reboot können wir die nötigen Packages laden.

4. LED SOFTWARE PACKAGE LADEN

Mit

```
git clone https://github.com/jgarff/rpi_ws281x
```

holen wir uns nun ein Test-Suit. Mit diesem können wir das ganze Testen, als auch eine Installation machen, damit wir das ganze später in unserem Script nutzen können.

Danach wechseln wir in das neue Verzeichnis und installieren mit scons.

```
cd rpi_ws281x/ && sudo scons
```

Wenn dies durchgelaufen ist, kompilieren & installieren wir das Ganze:

```
cd python && sudo python setup.py build && sudo python setup.py
install
```

Damit sollten wir nun die Panel nutzen können. Testen können wir das Ganze mit:

```
sudo PYTHONPATH=".:build/lib.linux-armv7l-2.7" python
examples/strandtest.py
```

Wenn der Test erfolgreich war, können wir fortführen. Wenn nicht, sollte das Panel und die vorherigen Steps geprüft werden.

5. SIEHE → LÄRMAMPEL SCRIPT

6. AUTOSTART NACH REBOOT

Damit unser Ampelscript automatisch nach jedem Neustart, direkt mit dem System bootet, müssen wir dies entsprechend hinterlegen. Dazu öffnen wir mit nano die Datei /etc/rc.local

```
sudo nano /etc/rc.local
```

Und fügen dort, eine neue Zeile vor dem exit o ein:

```
screen -dmS lernampel sudo PYTHONPATH=".:build/lib.linux-armv7l-2.7"
python /home/pi/lernampel.py
```

7. FINETUNING

Wenn der Sensor ein Poti hat, muss der Ponti nun eingestellt werden. Das kann sich etwas hinziehen, da es keine fixen Einstellungspunkte gibt und auch keine dB Orientierungen. Hier müsst ihr dann nach eigenem Ermessen den Poti einstellen.