

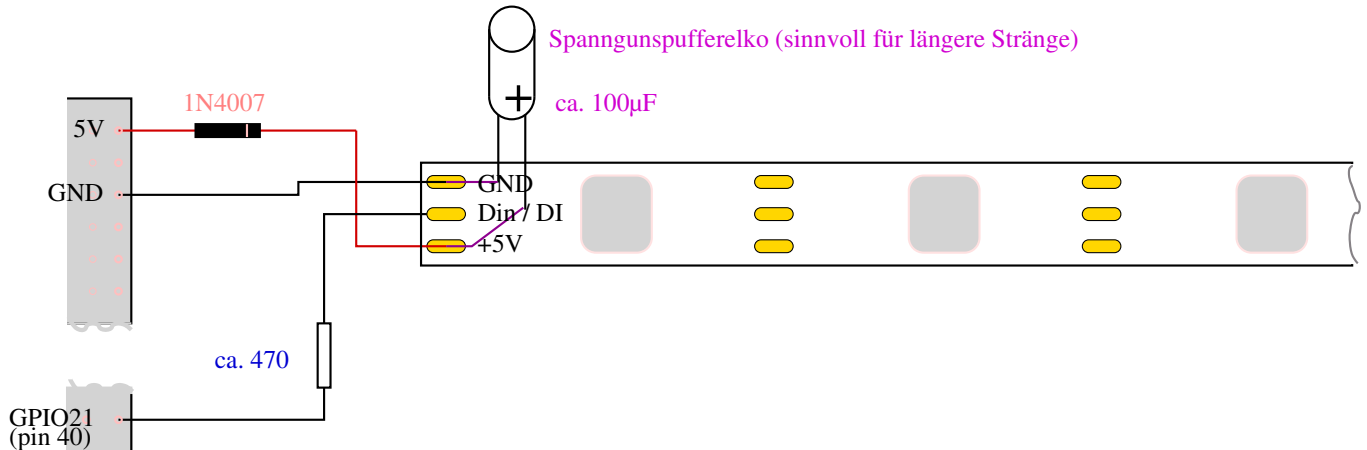
# RPi zum Ansteuern der WS2812B-Stränge

Mattscheibe, Januar 2021

Dies ist eine recht kostenintensive Variante. Ein Raspberry Pi Zero schlägt mit ca. 15 Euro zu Buche, ein mit Schere auftrennbarer Strang mit 240 LEDs kostet ca. 40 Euro. Ein typisches Kleingerät (Elektronischer Würfel, Tic Tac Toe o.Ä.) ist damit für nicht unter 18 Euro zu realisieren (inkl. Taster). Besser eignet sich der "große" Ansatz für Projekte, in denen wirklich die Rechenkraft und der Speicherplatz des Pi benötigt werden (z.B. für ein richtig schlaues spielendes "4 Gewinnt"-Spiel).

Ein am Pi entwickeltes Programm kann aber mit geringen Änderungen (hinsichtlich WS2812B nur beim ersten und beim letzten Befehl des Programms) auch auf einen AVR-Controller portiert werden. Somit bietet es sich evtl. an, den Pi als Entwicklungsplattform zu nutzen und für fertige Geräte dann durch z.B. einen Arduino Nano zu ersetzen (als Nachbau für ca. 3 Euro zu haben).

Es muss neben der Stromversorgung nur ein Pin angeschlossen werden. Hierzu muss zwingend der Pin GPIO18 am Pi gewählt werden (Lage siehe Abbildung). Ob eine Diode in der Stromversorgungsleitung nötig ist, hängt vom verwendeten Streifen ab. Für längere Streifen sollte ein Elko zur Spannungspufferung eingesetzt werden.



Zur Ansteuerung wird die Bibliothek [https://github.com/Mattscheibe/ws2812b\\_rpi](https://github.com/Mattscheibe/ws2812b_rpi) verwendet. Für sie dient die Bibliothek [https://github.com/jgarff/rpi\\_ws281x](https://github.com/jgarff/rpi_ws281x) als Grundlage. Verwendet werden deren Low-Level-Funktionen zur Übertragung von Farbdaten eines Arrays zum Strang. Bei `ws2812b_rpi` sind die Übertragungskommandos und die Datenstrukturen aber so angepasst, dass sie sich meiner Meinung nach besser für die Praxis eignen.

## Anwendung:

1. `sudo apt-get install cmake ; cd /tmp ; git clone https://github.com/jgarff/rpi_ws281x ; mkdir build ; cd build ; cmake .. ; sudo make install`
2. `cd /tmp ; git clone https://github.com/Mattscheibe/ws2812b_rpi ; cd ws2812b_rpi ; sudo make install`
3. `mkdir ~/Test ; cp /tmp/ws2812_rpi/Examples/Makefile ~/Test ; cd ~/Test`
4. Nach Aufruf von "make make" wird der Name für eine Hauptdatei vergeben, z.B. `test`. (Soll ein anderer Editor verwendet werden, wird dieser bitte zuvor im Makefile eingetragen!)
5. Es wird das Kommando "make edit" getippt und dann ein Programm eingegeben. Dabei gehört an den Anfang des Programms ein

```
#include <ws2812_rpi.h>
```

6. Die Datei wird gespeichert und durch Eingabe von "make run" gestartet.

*Im Editor `gedit` registriert man sich einmalig mit dem Menüpunkt "Externe Werkzeuge verwalten" ein eigenes Kommando mit der Befehlszeile "make run" und einer zugeordneten Tastenkombination (z.B. STRG-U) und gibt an, dass bei seiner Ausführung vorher gespeichert werden soll.*

Hier ein Beispielprogramm:

```
#include <ws2812b_rpi.h>
int main() {
    initLEDsPCM(1,1,1); // Mini-"Matrix": 1 Zeile, 1 Spalte, Typ 1 (also: 1 LED :-))
    setRGB(0,0,0x001100); // gruen bei 0,0 (einzige LED) wegen Farbschema RRGGBB
    showLEDs(); // zeigt nun alle (...) LEDs an
    sleep(4); // 4 sek. Pause
    endLEDs(); // das muss am Schluss jedes Programms stehen
    return 0;
}
```

Eine Übersicht aller verfügbaren Kommandos findet sich auf der Rückseite.

Für vollständige Programme mit *Ein- und Ausgabe*, *Tastenabfrage* etc. bietet sich als Zusatz die **wiringPi**-Bibliothek an, die geeignete Kommandos zur Verfügung stellt (siehe umseitig).

## Verfügbare Kommandos:

`initLEDsPCM(breite, hoehe, typ)` muss einmal zu Beginn des Programms stehen; die Verbindung zum Datenpin wird hergestellt, alle LEDs gehen aus; zur Typangabe siehe unten

`setR(zeile, spalte, Farbwert)` setzt das Pixel bei Zeile *zeile* und Spalte *spalte* auf einen Rotwert zwischen 0 (aus) und 255 (volle Helligkeit).

`setG(zeile, spalte, Farbwert)` dito für Grün

`setB(zeile, spalte, Farbwert)` dito für Blau

`setRGB(zeile, spalte, Farbwert)` setzt das Pixel bei Zeile *zeile* und Spalte *spalte* auf einen RGB-Farbwert zwischen 0x00000 und 0xFFFFFF; dabei gilt die Form "0xRRGGBB", also zuerst der Rotanteil **RR**, dann der Grünanteil **GG**, dann der Blauanteil **BB**. Der Vorsatz "0x" erlaubt hierbei die Eingabe hexadezimaler Farbwerte. Ohne ihn muss ein (umgerechneter) Dezimalwert angegeben werden. Es gehen auch Binär-Farbwerte, z.B. in der Form "0b 00000000 11111111 00000000".

`showLEDs()` zeigt den aktuellen Farbmatrixinhalt an den LEDs an — wird also am besten aufgerufen, nachdem man alle Farbwerte zuvor gesetzt resp. aktualisiert hat.

`clearLEDs()` macht alle LEDs dunkel und löscht zudem das Array mit den Farbwerten.

`getR(zeile, spalte)` liefert den Rotwert (zwischen 0 und 255) des Pixels in Zeile *zeile* und Spalte *spalte*.

`getG(zeile, spalte)` dito für Grün

`getB(zeile, spalte)` dito für Blau

`getRGB(zeile, spalte)` liefert als Dezimalwert den RGB-Farbwert des Pixels bei Zeile *zeile* und Spalte *spalte*.

`setMaxBrightness(maximalerFarbwert)` bestimmt, welcher Farbwert maximal ausgegeben werden darf — kann zum Dimmen der LEDs verwendet werden; mögliche Maximalwerte liegen zwischen 0 (nicht sinnvoll, alles bleibt dunkel) und 255 (keine Dimmung). Für Experimente mit schwacher Spannungsversorgung und vielen LEDs empfiehlt sich z.B., hier einen niedrigen Wert zu setzen, welcher später erhöht wird, wenn man eine bessere Spannungsversorgung zur Verfügung hat. Außerdem schont Dimmen die Netzhaut!!!

`endLEDs()` wird am Schluss einmal aufgerufen, um die Hardware des RPi wieder von ihrer Steuerungsfunktion für WS2812B-LEDs zu entbinden

Neben der Verwendung der obigen Befehle zum Setzen und Lesen der Pixel-Farbwerte kann auch direkt auf das Array `pixel` zugegriffen werden. Allerdings fehlt durch dessen lineare Struktur die einfache Zuordnung zu den Pixelkoordinaten, wie sie die Befehle je nach angegebenem Verdrahtungsschema auch bei Matrix-Anordnung der LEDs liefern.

---

Der Typ des Verdrahtungsschemas richtet sich nach der gewählten Anordnung der LEDs (Verdrahtungsschema):

Typ 0: °|\_|-↓ 1: °|/|/↓ 2: ↑\_|-|\_|° 3: ↓\\|\\°

Typ 4: °- 5: °- 6: °- 7: °-  
| / | \  
- - - -  
| / | \  
→ → ← ←

Typ 8: .|\_|-↑ 9: .|\\|\\↑ 10: ↑\_|-|\_|. 11: ↑/|/|.

Typ 12: → 13: → 14: ← 15: ←  
| \ | /  
- - - -  
| \ | /  
.- .- .- .-

° = Einspeisung , . = Einspeisung , | = Strang , | = Querverbindung

---

Mit der Bibliothek geschriebene Programme würden mit der *ws2812b*-Bibliothek für AVR in gleicher Form direkt auf einem AVR-Mikrocontroller oder auch auf einem Arduino-Board laufen (unterschiedlich wären hierbei nur das Include und die zugehörigen Makefile-Angaben zu Prozessor und Programmieradapter, der Befehl `initLEDs()` würde parameterlos aufgerufen und am Ende würde kein `endLEDs()` stehen).

Bindet man außerdem die *wiringPi*-Bibliothek ein (Befehl `#include <wiringPi.h>`), so stehen für vollständige Programme häufig benötigte Befehle für z.B. Tastenabfragen, allgemein: Ein- und Ausgabe von Signalen, zur Verfügung. Im Makefile ist dabei ein Kommentarzeichen zu entfernen. Die Bibliothek findet sich unter [wiringpi.com](http://wiringpi.com).