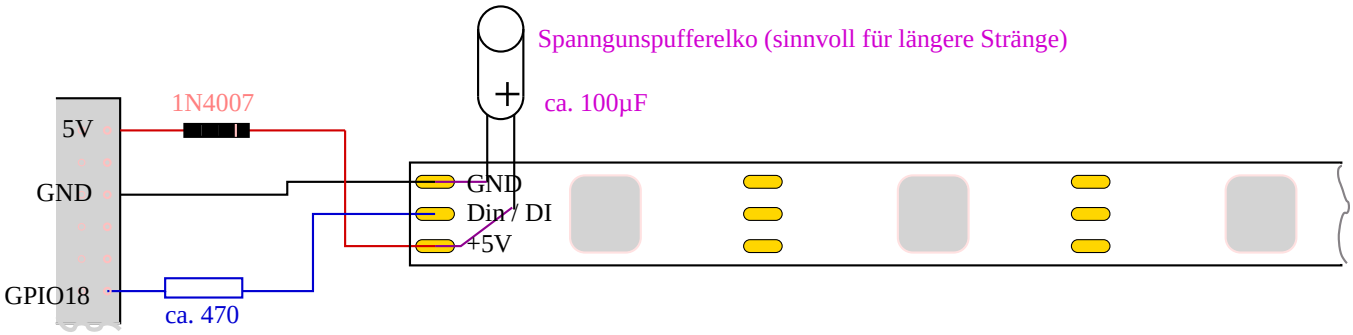
RPi zum Ansteuern der WS2812B-Stränge

Dies ist eine recht kostenintensive Variante. Ein Raspberry Pi Zero schlägt mit ca. 15 Euro zu Buch, ein mit Schere auftrennbarer Strang mit 240 LEDs kostet ca. 75 Euro. Ein typisches Kleingerät (Elektronischer Würfel, Tic Tac Toe o.Ä.) ist damit für nicht unter 20 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 schlau 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/ws2812_rpi verwendet. Für sie dient die Bibliothek https://github.com/626Pilot/RaspberryPi-NeoPixel-WS2812 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 den Bastelbetrieb eignen.

Anwendung:

1. ws2812b_rpi.h, ws2812b_rpi.c, ws2812-RPi.h, ws2812-RPi.c und ein Makefile werden ins Verzeichnis, in dem das eigene Programm entstehen soll, kopiert.
2. ws2812-RPi.h und ws2812-RPi.c werden editiert: es wird je einmal `#define LED_BUFFER_LENGTH 169` gesetzt (statt vorher 24).
3. Nach Aufruf von "`make make`" wird der Name für eine Hauptdatei vergeben, z.B. test. (Soll ein anderer Editor verwendet werden, wird dieser auch in diesem Schritt ausgewählt.)
4. Es wird das Kommando "`make edit`" getippt und dann ein Programm eingegeben. Dabei gehört an den Anfang des Programms ein

   ```
   #include "ws2812_rpi.h"
   ```

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

   Im Editor joe drückt man dazu erst ESC, dann C, bestätigt das Speichern und gibt als Kommando "`make run`" an.
   Im Editor gedit registriert man sich einmalig mit dem Menüpunkt "Externe Werkzeuge verwalten" ein eigenes Kommando mit der Befehlszeile "make prog" 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() {
  initLEDs(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
  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*, *Pausen* etc. bietet sich als Zusatz die **wiringPi**-Bibliothek an, die geeignete Kommandos zur Verfügung stellt (siehe umseitig).

Verfügbare Kommandos:

`initLEDs(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 "`0xRR`*`GG`*`BB`", 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 die aktuelle Belegung an — wird 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

`getG(`*`zeile`*`,`*`spalte`*`)` dito für Blau

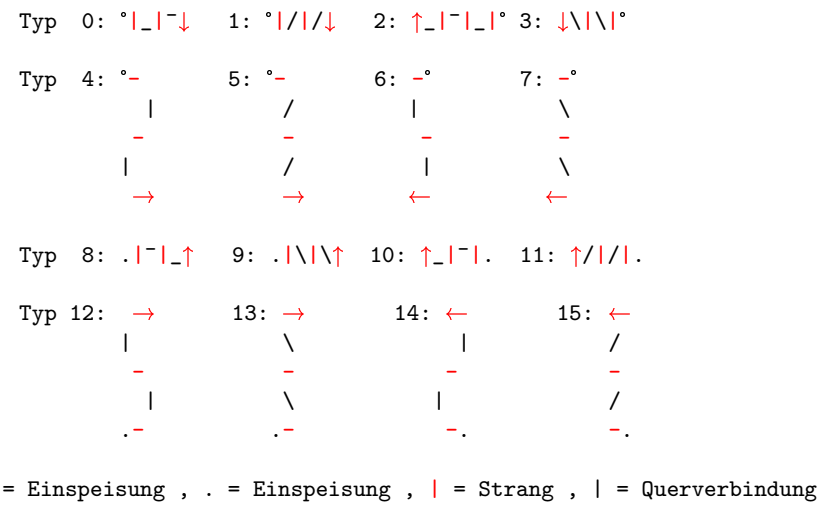
`getRGB(`*`zeile`*`,`*`spalte`*`)` liefert als Dezimalwert den RGB-Farbwert des Pixels bei Zeile *zeile* und Spalte *spalte*.

`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 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 (nicht vergessen!)

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):



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

---

Mit der Bibliothek geschriebene Programme würden mit der ws2812b-Bibliothek für AVRs 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>), wo stehen für vollständige Programme häufig benötigte Befehle für z.B. Pausen, Tastenabfragen, allgemein: Ein- und Ausgabe von Signalen, zur Verfügung. Im Makefile ist dabei ein Kommentarzeichen zu entfernen. Die Bibliothek findet sich unter wiringpi.com.