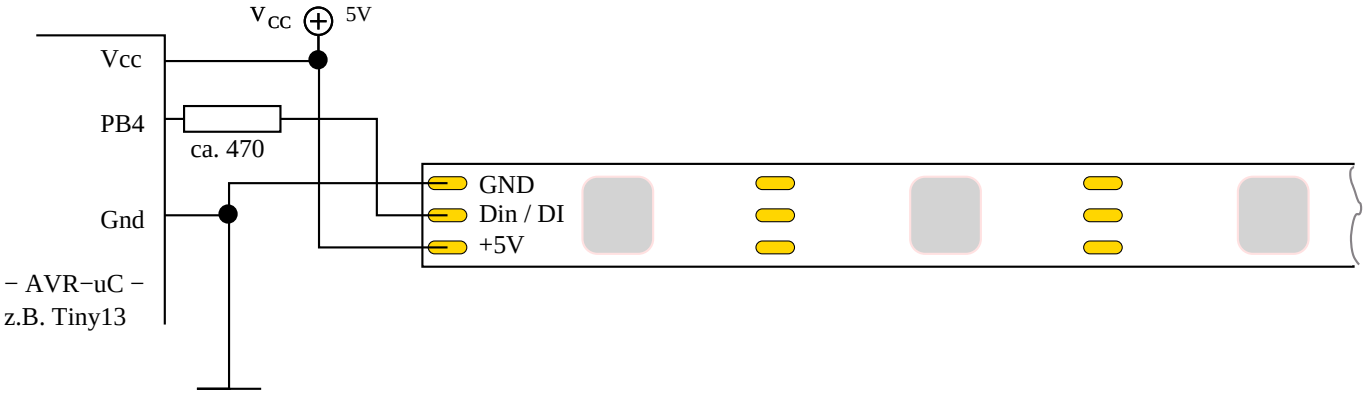
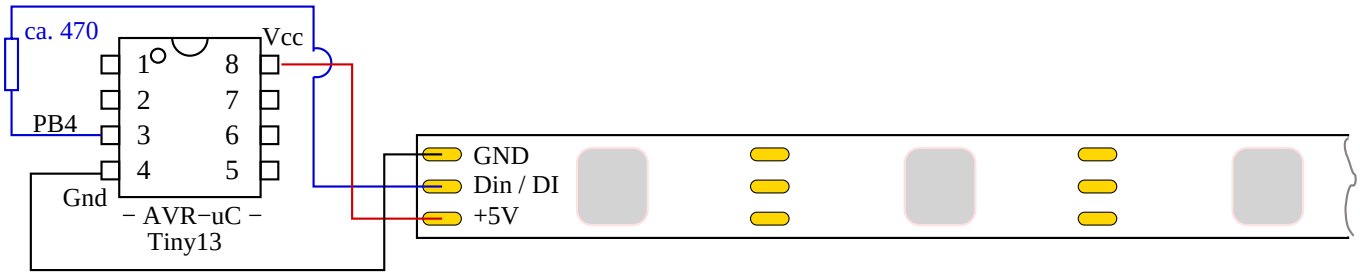
AVR-Mikrocontroller zum Ansteuern der WS2812B-Stränge

Dies ist die kostengünstigste Variante. Ein passender Controller kostet ca. 1,80 Euro, ein mit Schere auftrennbarer Strang mit 240 LEDs ca. 100 Euro. Ein typisches Kleingerät (Elektronischer Würfel, Tic Tac Toe o.Ä.) ist damit für unter 8 Euro zu realisieren (inkl. Taster, Batterie etc.).

Es muss neben der Stromversorgung nur ein Pin angeschlossen werden. Ein beliebiger Ausgangspin des AVR kann gewählt werden. Hier ein Beispiel mit einem AVR Tiny13 Mikrocontroller, dessen Pin PB4 als Anschlusspin verwendet wird:



oder anders dargestellt:



Zur Ansteuerung wird die heute bereitgestellte Bibliothek ws2812b verwendet. Für sie dient die Bibliothek light-ws2812 von https://github.com/cpldcpu/light_ws2812 als Grundlage. Verwendet wird deren Assembler-Inline-Routine, die die Farbdaten eines Arrays zum Strang überträgt. Bei ws2812b sind die Übertragungskommandos und die Datenstrukturen aber so angepasst, dass sie sich meiner Meinung nach besser für den Bastelbetrieb eignen.

Anwendung:

1. ws2812b.h und ws2812b.c und ein Makefile werden ins Verzeichnis, in dem das eigene Programm entstehen soll, kopiert.
2. ws2812b.h wird editiert: Anzahl und Anordnung der LEDs sowie der bei der Verdrahtung verwendete Datenpin werden eingegeben.
3. Das Makefile wird editiert: Prozessortyp und Taktfrequenz werden angepasst, ebenso der zur Übertragung zum Mikrocontroller verwendete Programmieradapter (z.B. “usbtiny”).
4. Eine Hauptdatei, z.B. test.c, wird erstellt (dann müsste der Name “test” – ohne `.c` – im Makefile eingetragen sein). Das lässt sich auch mit dem Kommando “`make edit`” einleiten.
5. Die Datei wird gespeichert, übersetzt und zum Mikrocontroller übertragen.

   Im Editor joe drückt man dazu erst ESC, dann C, bestätigt das Speichern und gibt als Kommando “`make prog`” 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.h"
void main() {
  initLEDs(); // immer einmal beim Start
  setRGB(0,0,0x00FF00); // gruen bei 0,0
  showLEDs(); // zeigt nun alle (...) an
}
```

Eine Übersicht aller verfügbaren Kommandos und ein weiteres Beispielprogramm finden sich auf der Rückseite.

Verfügbare Kommandos:

`initLEDs()` muss einmal zu Beginn des Programms stehen: die Verbindung zum Datenpin wird hergestellt, alle LEDs gehen aus

`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

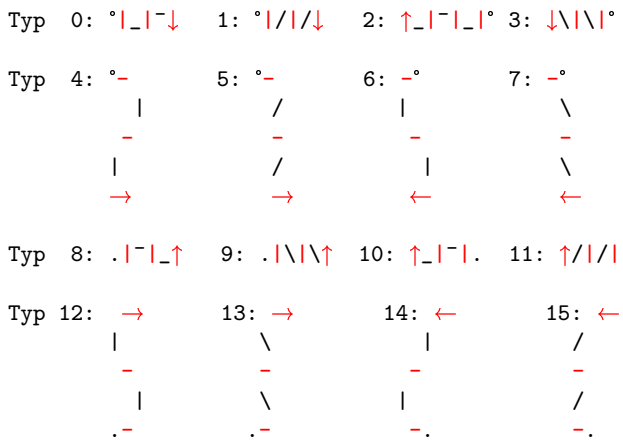
`getRGB(`*`zeile`*`,`*`spalte`*`)` liefert als Dezimalwert den RGB-Farbwert des Pixels bei Zeile *zeile* und Spalte *spalte*.

`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.

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.

Das Programm würde in gleicher Form direkt auf einem AVR-Mikrocontroller oder auch auf einem Arduino-Board laufen (unterschiedlich wären hierbei nur die zugehörigen Makefile-Angaben zu Prozessor und Programmieradapter).

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



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

---

Hier noch ein (platzsparend formatiertes ;-) Blinklicht für eine LED (Anpassungen nach "make config" in ws2812b_config.h nicht vergessen!):

```
#include "ws2812b.h" // definiert auch Funktion delay(ms)
void main() {
  initLEDs(1,1,1); // Matrix 1x1 Typ 1 = 1 LED
  while(1) { // loop
    setRGB(0,0,0x030201);showLEDs();delay(100);clearLEDs();delay(500);
  }
}
```