

Linker-Kit RGB LED – Anleitung für den Arduino und Raspberry Pi

Artikel-NR: LK-LED-RGB

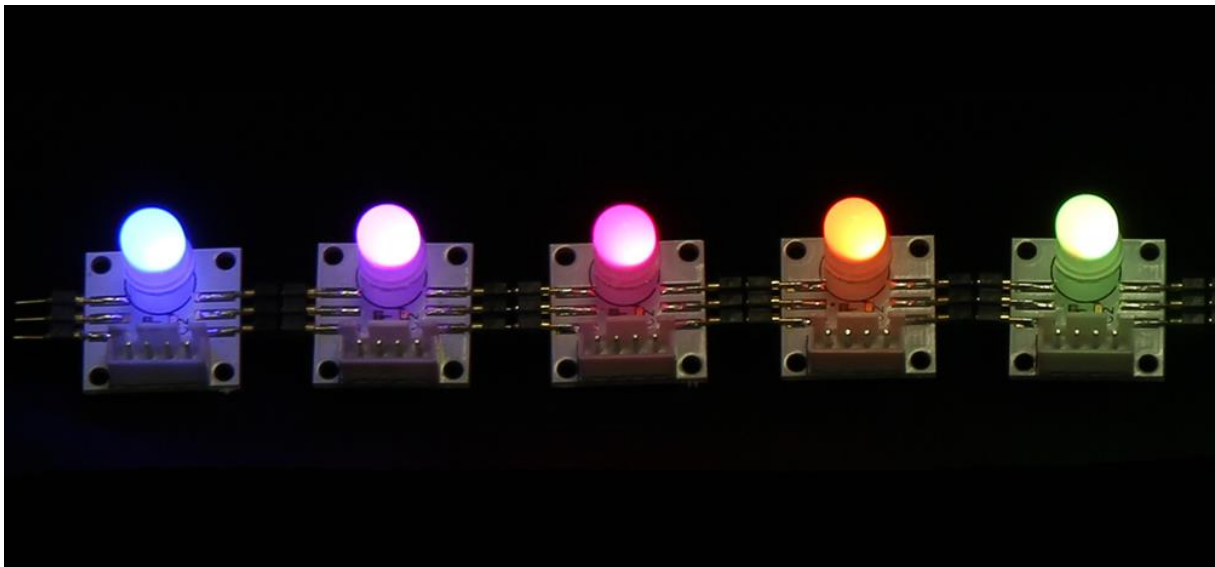
Verwendeter LED-Controller: WS2812

Zulässiger Spannungsbereich: 3-5VDC

Bestimmungsgemäße Verwendung:

Einsatz in Privathaushalten für Hobby Zwecke / Versuchsaufbauten, als mehrfarbiges Leuchtmittel.
Nicht für den Außeneinsatz geeignet; Nutzung nur in trockenen Innenräumen.

Achtung dieses Produkt ist kein Spielzeug! Setzen Sie es nicht für kritische Anwendungen ein.



Sehr geehrter Kunde,

vielen Dank, dass Sie sich für unser Produkt entschieden haben.

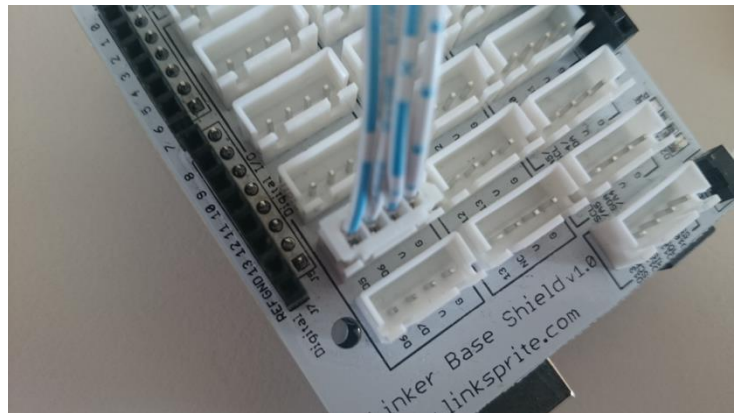
Im Folgenden haben wir aufgelistet, was bei der Inbetriebnahme zu beachten ist:

Anleitung für den Arduino

Schritt 1 - Anschließen der RGB-LED

Nutzen Sie das LinkerKit-System, so können Sie die RGB-LED einfach mit einem entsprechenden LinkerKit Kabel direkt an einen freien **digitalen Ausgang** verbinden.

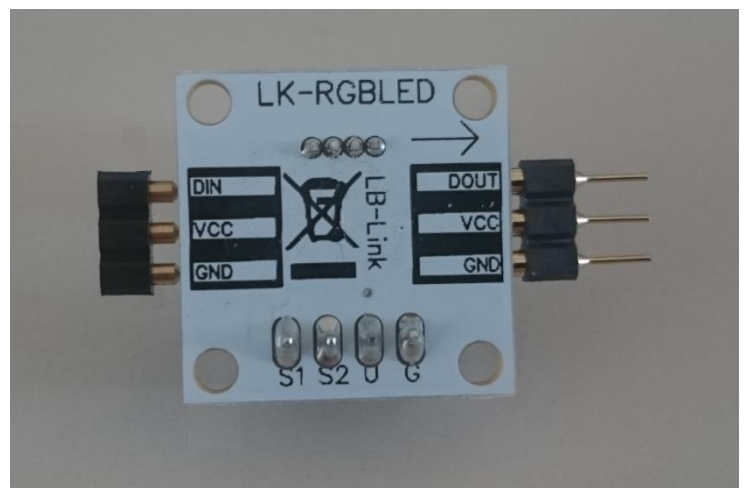
Hier in unserem Beispiel verwenden wir den digitalen Ausgang „D5“ des LinkerKit Base Shields.



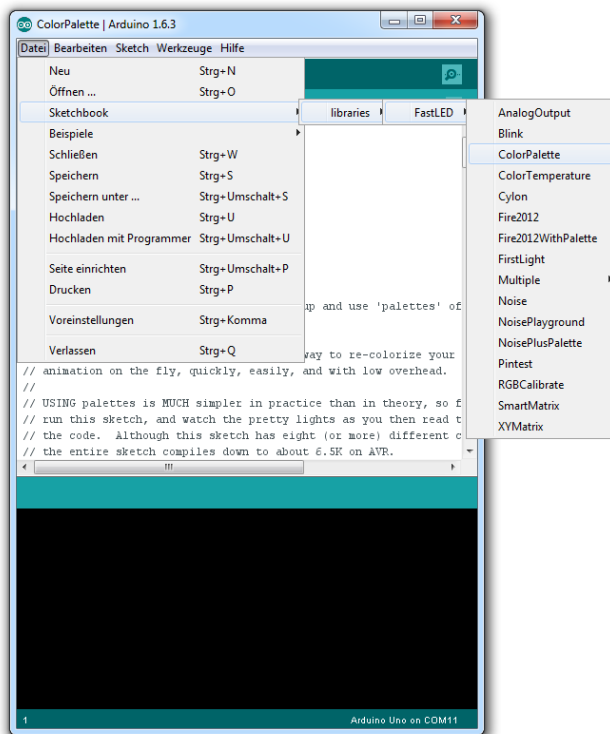
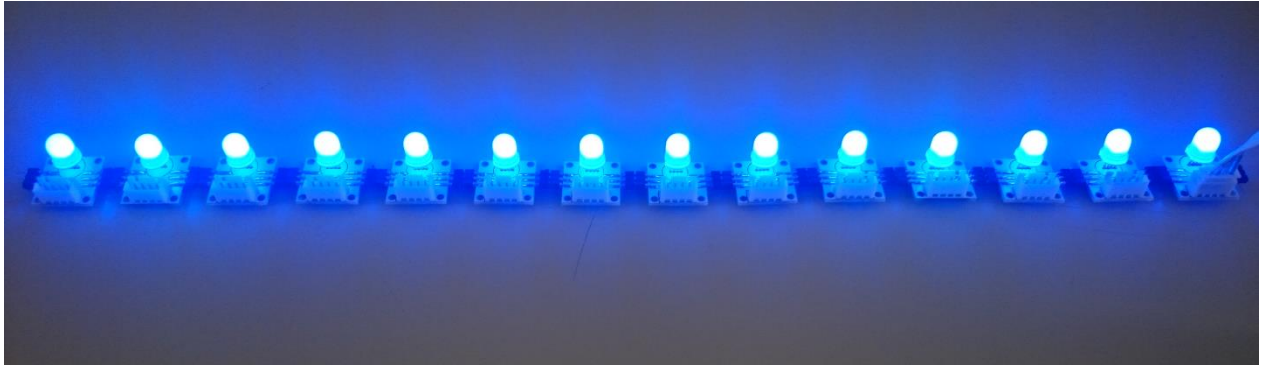
Möchten Sie die LED direkt ohne LinkerKit anschließen, so achten Sie auf die Pinbezeichnung auf der Rückseite der LED-Platine (S1: digitales Signal, S2: nicht verbunden, V: 5V, G: Masse). Dieses Vorgehen ist sinnvoll, wenn Sie z.B. eine externe Spannungsversorgung für eine größere Anzahl an LED's (> 15) anschließen möchten.



Sinngemäß können Sie mehrere RGB-LED's miteinander zur einer Kette verbinden. Hierbei beachten Sie den Richtungspfeil, der auf der Rückseite angegeben ist – die erste LED ist immer die, die am LinkerKit Kabel verbunden ist.



Haben Sie die LED's angeschlossen, so leuchten diese standardmäßig vorab blau.



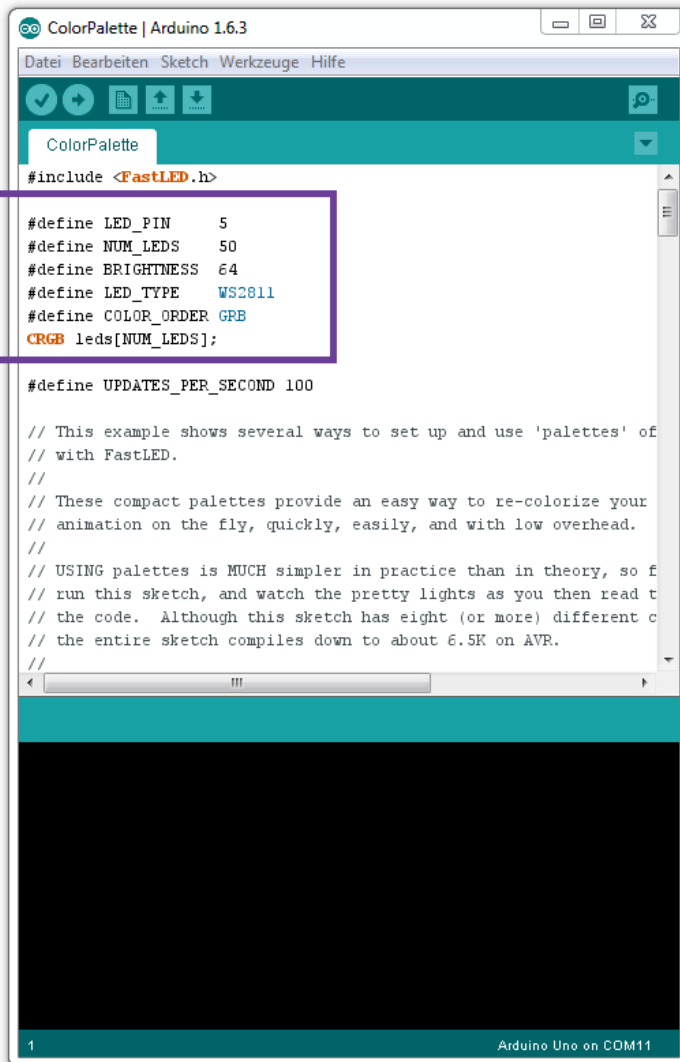
Schritt 2- Installation der Beispiel-Software

Laden Sie [hier \(FastLED.io github\)](https://github.com/FastLED/FastLED) die aktuelle FastLED library als zip herunter und entpacken Sie den enthaltenen Ordner unter C:\Benutzer\[IhrBenutzername]\Dokumente\Arduino\libraries\

Starten Sie hier nach Ihre Arduino-Software. Nun sollte unter dem Menü-Punkt „Datei-> Sketchbook-> libraries->“ der neue Punkt FastLED verfügbar sein.

Unter diesem befinden sich mehrere Beispiele, anhand dieser man eine eigene Programmierung der RGB-Led entwickeln kann.

Schritt 3 – Konfiguration der Code-Beispiele



```
#include <FastLED.h>

#define LED_PIN    5
#define NUM_LEDS  50
#define BRIGHTNESS 64
#define LED_TYPE   WS2811
#define COLOR_ORDER GRB
CRGB leds[NUM_LEDS];

#define UPDATES_PER_SECOND 100

// This example shows several ways to set up and use 'palettes' of
// with FastLED.
//
// These compact palettes provide an easy way to re-colorize your
// animation on the fly, quickly, easily, and with low overhead.
//
// USING palettes is MUCH simpler in practice than in theory, so if
// you run this sketch, and watch the pretty lights as you then read
// the code.  Although this sketch has eight (or more) different
// the entire sketch compiles down to about 6.5K on AVR.
```

Die meisten Code-Beispiele der FastLED library beginnen mit einem Konfiguration-Abschnitt im Header.

Dieser ist wie folgt erklärt:

LED_PIN: Auswahl des digitalen Ausganges, an dem die Datenleitung (S1) der LED angeschlossen ist. In unserem o.g. Beispiel ist das Pin 5

NUM_LEDS: Das ist die Anzahl der angeschlossenen LED's. Im oben gezeigten Bild sind z.B. 14 Stück in einer Reihe angeschlossen.

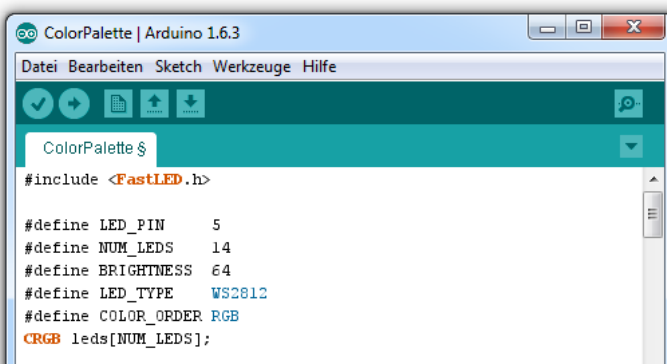
BRIGHTNESS: Beschreibt die eingestellte Helligkeit der LED's

LED_TYPE: Hier muss der entsprechende Chipsatz angegeben werden. Für die LinkerKit RGB-LED ist das der **WS2812**.

Bei manchen FastLED Beispielen muss anstatt dieser Angabe, auch manchmal eine kommentierte Zeile umgeschrieben werden.

COLOR_ORDER: Beschreibt die Reihenfolge der Farben, wie diese im Datenstrom angesprochen werden. Für die LinkerKit RGB-LED ist dies [RGB]

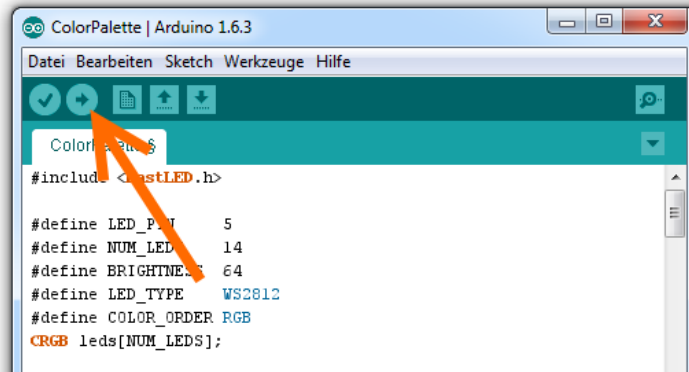
Für unser Beispiel, sehen Sie links im Bild die benötigten Einstellungen.



```
#include <FastLED.h>

#define LED_PIN    5
#define NUM_LEDS  14
#define BRIGHTNESS 64
#define LED_TYPE   WS2812
#define COLOR_ORDER RGB
CRGB leds[NUM_LEDS];
```

Schritt 4 – Programmieren des Arduinos



```
ColorPalette | Arduino 1.6.3
Datei Bearbeiten Sketch Werkzeuge Hilfe
ColorPalette.ino
#include <FastLED.h>

#define LED_PIN 5
#define NUM_LEDS 14
#define BRIGHTNESS 64
#define LED_TYPE WS2812
#define COLOR_ORDER RGB
CRGB leds[NUM_LEDS];
```

Mit dem angepassten Code, kann nun der Arduino programmiert werden.

Bitte beachten Sie die Auswahl der richtigen Programmierschnittstelle

(Werkzeuge->Port)

Nach dem Hochladen leuchten die RGB-LED's in verschiedenen Farben anhand des verwendeten Beispiels.



Anleitung für den Raspberry Pi

Schritt 1 - Anschließen der RGB-LED

Nutzen Sie das LinkerKit-System, so können Sie die RGB-LED einfach mit einem entsprechenden LinkerKit Kabel direkt an das LinkerKit BaseBoard anschließen.

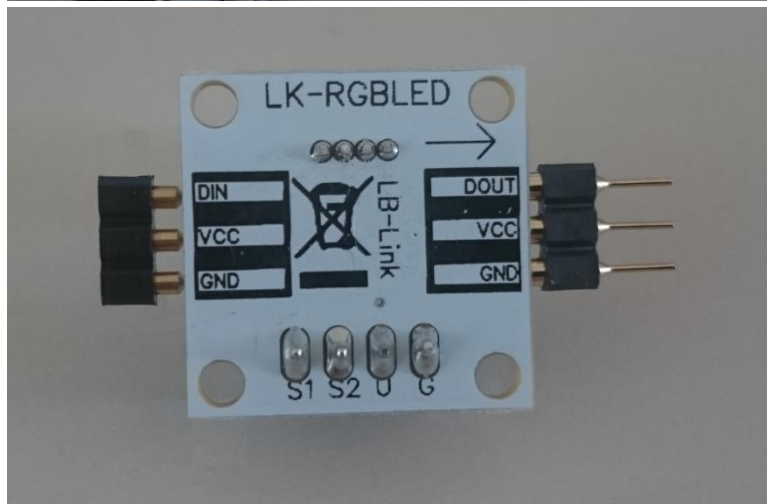
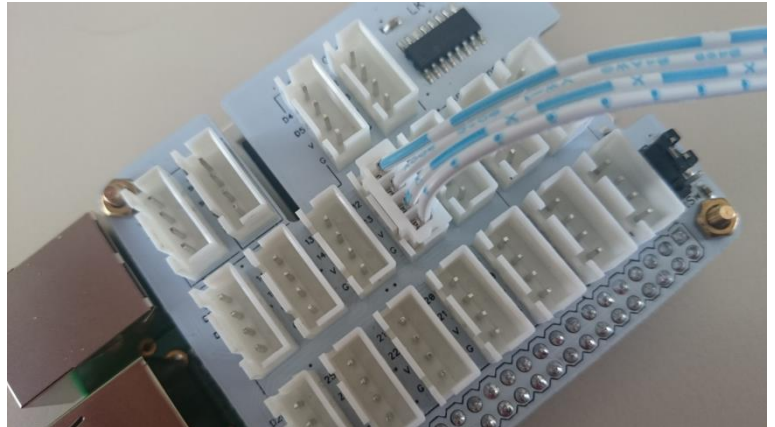
Beachten Sie hierbei, dass Sie einen Ausgang verwenden der das Hardware- PWM des Raspberry Pi nutzen kann [GPIO12/Channel0; GPIO18/Channel0; GPIO13/Channel1; GPIO19/Channel1]

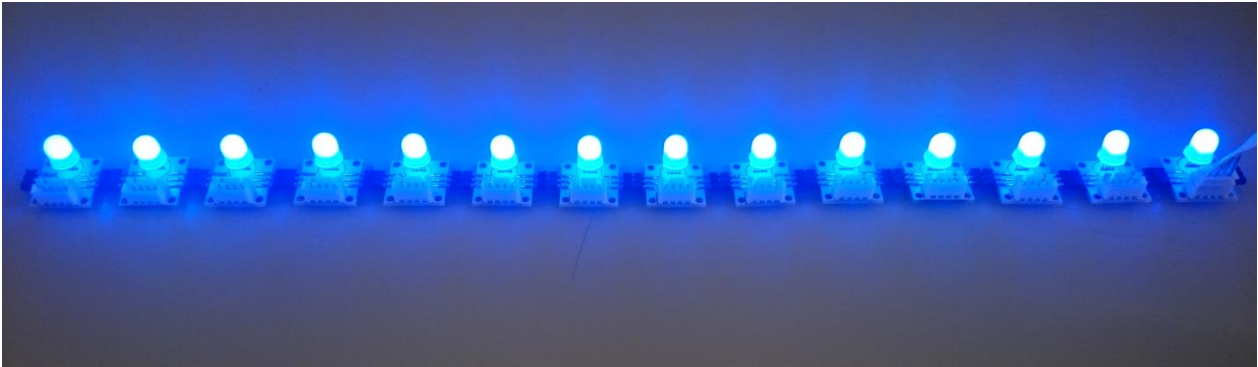
Hier in unserem Beispiel verwenden wir den GPIO 12 mit dem PWM-Channel 0 des Linker Base Shields.

Bitte beachten Sie, dass für einen der RGB-LED mit den Einsatz mit dem Raspberry Pi, die Versorgungsspannung 3.3V betragen sollte. Nutzen Sie hier das Linkerkit Base Shield, so können Sie die Spannung mit dem rechts gezeigten Jumper auf der rechten Position die Spannung auf 3.3V wechseln.

Möchten Sie die LED direkt anschließen, so achten Sie auf die Pinbezeichnung auf der Rückseite der LED-Platine (S1: digitales Signal, S2: nicht verbunden, V: 3.3V, G: Masse). Dieses Vorgehen ist sinnvoll, wenn Sie z.B. eine externe Spannungsversorgung für eine größere Anzahl an LED's (> 15) anschließen möchten.

Sinngemäß können Sie mehrere RGB-LED's miteinander zur einer Kette verbinden. Hierbei beachten Sie den Richtungspfeil, der auf der Rückseite angegeben ist – die erste LED ist immer die, die am LinkerKit Kabel verbunden ist.





Haben Sie die LED's angeschlossen, leuchten diese standardmäßig vorab blau.

Schritt 2- Installation der Beispiel-Software

Mittels der folgenden Befehle, laden Sie die rpi_w281x Library herunter und installieren diese. Jeder Befehl muss in die Kommandozeile eingegeben und mit [Enter] bestätigt werden. In diesem Beispiel verwenden wir das Betriebssystem Raspbian (Debian Wheezy).

```
sudo apt-get update  
sudo apt-get install build-essential python-dev git scons swig  
git clone https://github.com/richardghirst/rpi\_ws281x.git  
cd rpi_ws281x  
sudo apt-get install scons  
sudo scons  
cd python  
sudo python ez_setup.py  
sudo python setup.py install  
cd examples/
```

Schritt 3 – Konfiguration der Code-Beispiele

In dem Ordner „examples“ befinden sich zwei Beispiele, die aufzeigen, wie man mit dem Raspberry Pi die Linker Kit RGB-LED ansteuern kann. Dies sind die Dateien „strandtest.py“ und „lowpower.py“

Mit dem folgenden Befehl öffnen wir nun eine der beiden Datei im Editor, um sie zu konfigurieren:

```
nano strandtest.py
```

```
GNU nano 2.2.6 FI
# NeoPixel library strandtest example
# Author: Tony DiCola (tony@tonydicola.com)
#
# Direct port of the Arduino NeoPixel library strandtest example. Showcases
# various animations on a strip of NeoPixels.
import time

from neopixel import *

# LED strip configuration:
LED_COUNT      = 16      # Number of LED pixels.
LED_PIN        = 18      # GPIO pin connected to the pixels (must support PWM!).
LED_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
LED_DMA        = 5       # DMA channel to use for generating signal (try 5)
LED_BRIGHTNESS = 255     # Set to 0 for darkest and 255 for brightest
LED_INVERT     = False   # True to invert the signal (when using NPN transistor level shift)

# Define functions which animate LEDs in various ways.
def colorWipe(strip, color, wait_ms=50):
    """Wipe color across display a pixel at a time."""
    for i in range(strip.numPixels()):
        strip.setPixelColor(i, color)
        strip.show()
        time.sleep(wait_ms/1000.0)

def theaterChase(strip, color, wait_ms=50, iterations=10):
    """Movie theater light style chaser animation."""
    for j in range(iterations):
        for q in range(3):
            for i in range(0, strip.numPixels(), 3):
                strip.setPixelColor(i+q, color)
            strip.show()
            time.sleep(wait_ms/1000.0)
            for i in range(0, strip.numPixels(), 3):
                strip.setPixelColor(i+q, 0)

def wheel(pos):
    """Generate rainbow colors across 0-255 positions."""
```

Die Dateien besitzen einen Header, in dem verschiedene Konfigurationsparameter eingestellt werden können. Diese sind wie folgt erklärt:

LED_COUNT: Das ist die Anzahl der angeschlossenen LED's. Im oben gezeigten Bild sind z.B. 14 Stück in einer Reihe angeschlossen.

LED_PIN: Auswahl des GPIO's, an dem die Datenleitung (S1) der LED angeschlossen ist. Diese muss zwingend an einem GPIO angeschlossen sein, die das Hardware-PWM vom Raspberry Pi unterstützt. [GPIO12/Channel0; GPIO18/Channel0; GPIO13/Channel1; GPIO19/Channel1]

In unserem o.g. Beispiel ist das LED Pin 12

LED_FREQ_HZ: Bezeichnet die Signalfrequenz, mit der die LED's angesteuert werden [Standard 800000 (800kHz)]

LED_DMA: DMA-Channel zur Generierung des benötigten Signals [Standard Channel:5]
Nur bei Bedarf/Konflikten ändern.

LED_BRIGHTNESS: Beschreibt die eingestellte Helligkeit der LED's [0 dunkel – 255 hell]

LED_INVERT: Invertiert das Datensignal für die RGB-LED's. Dies wird benötigt, wenn das Signal vorab durch Level-Shifting mit NPN-Transistoren invertiert wird.

[Nicht benötigt beim LinkerKit BaseShield und beim Betrieb bei 3.3V]

Für unser Beispiel, sehen Sie unten im Bild die benötigten Einstellungen.

```
GNU nano 2.2.6
# NeoPixel library strandtest example
# Author: Tony DiCola (tony@tonydicola.com)
#
# Direct port of the Arduino NeoPixel library strandtest example. Showcases
# various animations on a strip of NeoPixels.
import time

from neopixel import *

# LED strip configuration:
LED_COUNT      = 14      # Number of LED pixels.
LED_PIN        = 12      # GPIO pin connected to the pixels (must support PWM!).
LED_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
LED_DMA        = 5       # DMA channel to use for generating signal (try 5)
LED_BRIGHTNESS = 255     # Set to 0 for darkest and 255 for brightest
LED_INVERT     = False   # True to invert the signal (when using NPN transistor level shift)

# Define functions which animate LEDs in various ways.
def colorWipe(strip, color, wait_ms=50):
    """Wipe color across display a pixel at a time."""
    for i in range(strip.numPixels()):
        strip.setPixelColor(i, color)
        strip.show()
        time.sleep(wait_ms/1000.0)
```

Mit der Tastenkombination [Strg + X] und einem darauffolgenden [Y] und [Enter] zur Bestätigung, werden die Änderungen gespeichert.

Mit dem folgenden Befehl

```
sudo python strandtest.py
```

Wird das Skript gestartet und die LED's fangen an in verschiedenen Farben zu leuchten.



[Profi-Tipps]

- Die LED's müssen nicht miteinander gesteckt werden, sondern können auch untereinander mittels Kabel verbunden werden; achten Sie hierbei auf die Pin-Belegung, die auf der Rückseite angegeben ist. Auf diesem Wege, können die LED's über einen größeren Raum verteilt werden – Eine zu hohe Kabellänge sollte jedoch vermieden werden.
- Besitzen Sie z.B. einen LED-Streifen mit mehreren LED's, so kann dieser auch an die LinkerKit RGB angeschlossen werden, solange auch diese mit einem WS2812 kompatiblen Chipsatz ausgestattet sind.