Digitale LED-Streifen

- Digitale LED-Streifen
- Motivation
- Projekt Lichtspiele
- LED-Streifen kaufen
- Teile
- Digitaler LED-Streifen
- Anordnung der LED Streifen
- Anordnung Wortuhr
- Arduino Nano
- Schaltplan UNO I
- Schaltplan UNO II
- Stromverbrauch
- Eingangsspannung UNO
- Stromversorgung UNO
- Stromstärken UNO
- Mikrokontroller, IDE, Library
- Libraries
- Farbmischung
- RGB Farben
- RGB Farben HEX
- RGB Farben HTML
- HSL Code
- FastLED HSV
- FastLED HSV
- Anordnung Lauflicht
- LED-Adressen
- Figuren
- Erster Sketch I
- Erster Sketch I
- Links

H39@email.de

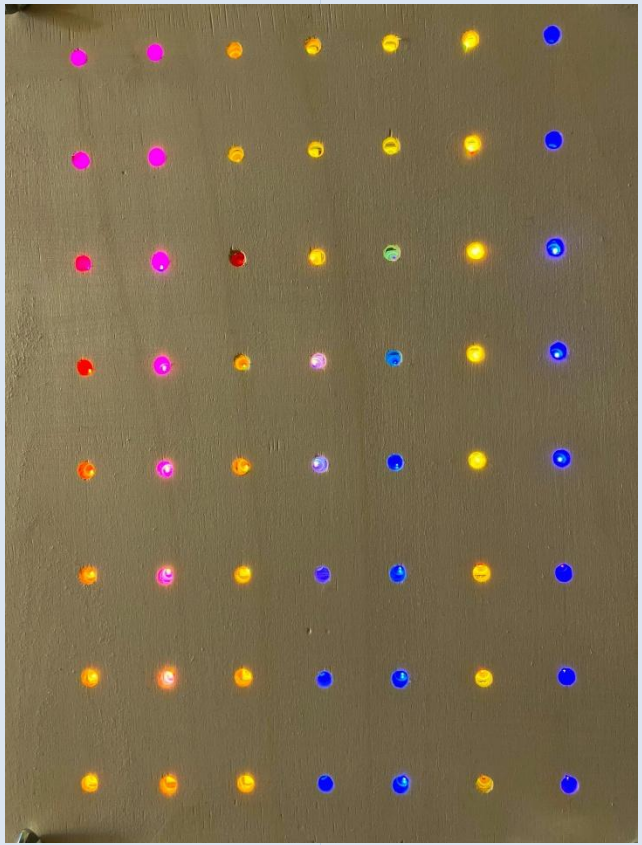
Motivation

Projekt Wortuhr

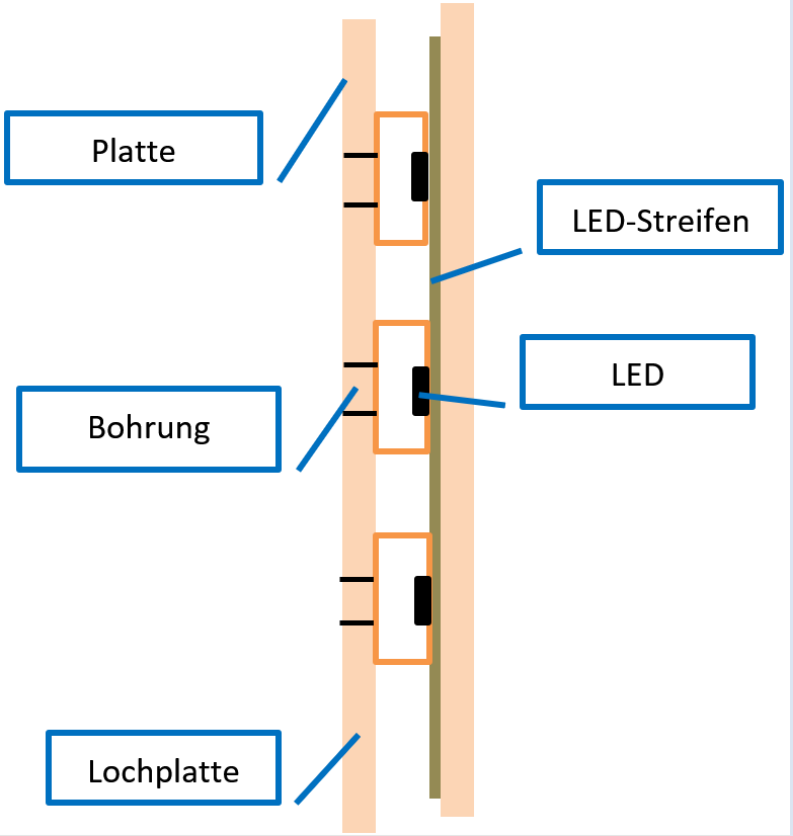
- Mit Zeitangaben in englischer Sprache.
- Wechsel der Anzeige alle 5 Minuten.
- Leds in der unteren LED-Leiste zeigen Minuten an.

Hier:
„IT‘s FIVE PAST TWELVE“
und eine LED ergeben 7:06.

Projekt Lichtspiele



Blickrichtung

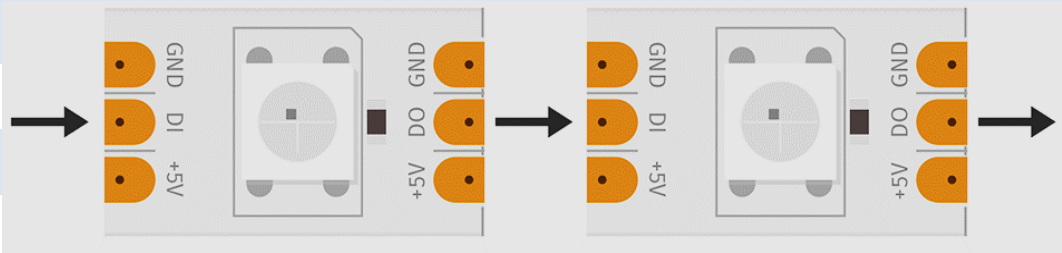


LED-Streifen kaufen

Anzahl der LEDs pro Meter	LED-Streifen unterscheiden sich in der Anzahl der LEDs pro Meter Hier eine Liste mit üblichen Werten:
	• 144 LEDs/m • 100 LEDs/m • 90 LEDs/m • 60 LEDs/m • 30 LEDs/m
LED-Typ	WS2812B oder Neopixel (Adafruit)
IP-Schutzklassen	IP 30: kein Schutz vor Wasser IP 65: beschichtet mit Silikon, daher etwas schwieriger zu schneiden und zu verlöten. Durch das Schneiden wird die Schutzklasse höchstwahrscheinlich zerstört und steigt auf IP 30 herab. IP 67: ummantelt von einem festen Silikon-Schlauch.
Quelle	Neopixel mit Arduino und ESP32 – der LED-Streifen Ultra Guide

Teile	
Anzeige	Sperrholzplatte DIN A4 als Montageplatte für LED-Streifen. Aufgeklebte LED-Streifen. Sperrholzplatte DIN A4 mit Löchern.
LED-Streifen	Digitaler LED-Streifen (Neopixel kompatibel), 56 Leds
Beispiel	5V 12V WS2812B RGB LED Streifen Strip 1-5m Lichtleiste Individuell Adressierbar | eBay
	(obiger LED-Streifen ist wasserdicht, nicht zu empfehlen) besser nur IP30.
LED-Typ	WS2812B
Mikrokontroller	Arduino UNO
	Arduino Nano
	ESP32-Wroom-32

Digitaler LED-Streifen

Aufbau	Als Erweiterung zu – analogen – LED Streifen hat jede LED einen Chip.
	Jede LED ist adressierbar und kann einzeln aus- und eingeschaltet werden. Auch können Farbe und weitere Effekte eingestellt werden.
	
	Hier hat der Streifen 3 Kontakte: Außen GND und +5V Innen DI, Datenleitung Eingang Innen DO, Datenleitung Ausgang
Datenleitung	Die Datenleitung hat eine Richtung. Zwei Streifen verbinden von DO nach DI.
Stromversorgung	Grundsätzlich sollte der LED Streifen eine eigene Stromversorgung haben.
Quelle	https://starthardware.org/viele-leds-mit-arduino-steuern-ws2812/

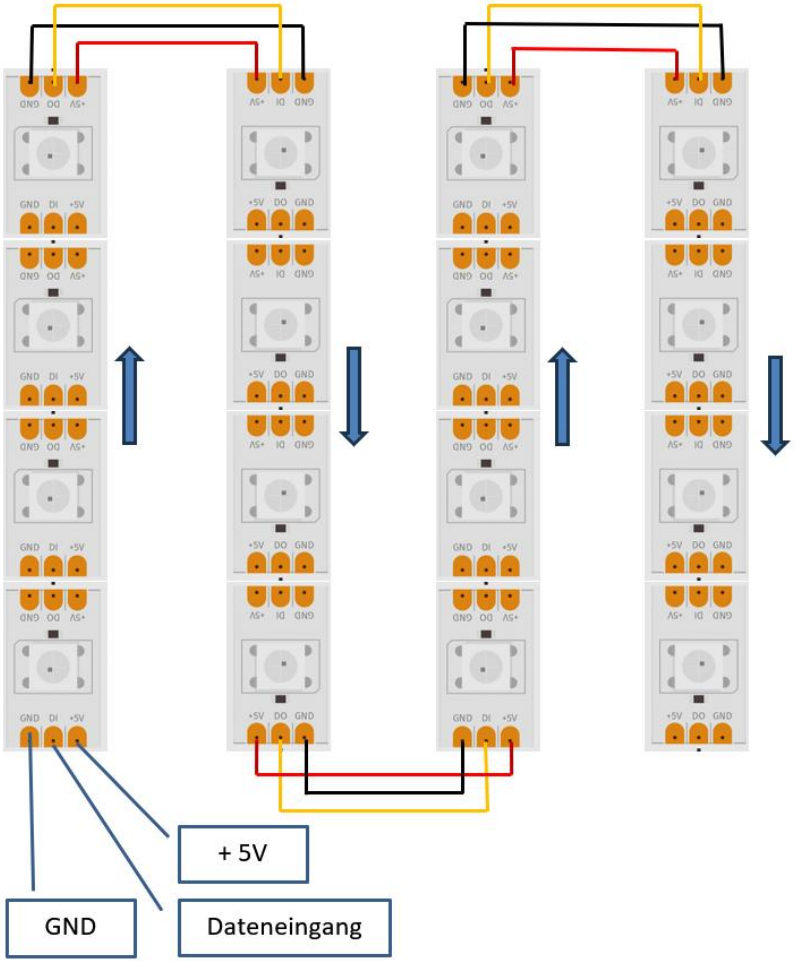
Anordnung der LED Streifen

Der LED-Streifen kann an den Kontakten aufgetrennt werden.

Da die LED Streifen eine Richtung haben, muss die Datenleitung immer in eine Richtung verlaufen.
Die Kontakte GND, +5V und Dateneingang „DI“ werden über Brücken verbunden.

Der Dateneingang „DI“ wird mit einem digitalem Arduino-Pin verbunden.

In dieser Skizze sind 12 Leds vorhanden. Diese haben die Adressen von 0 bis 11.

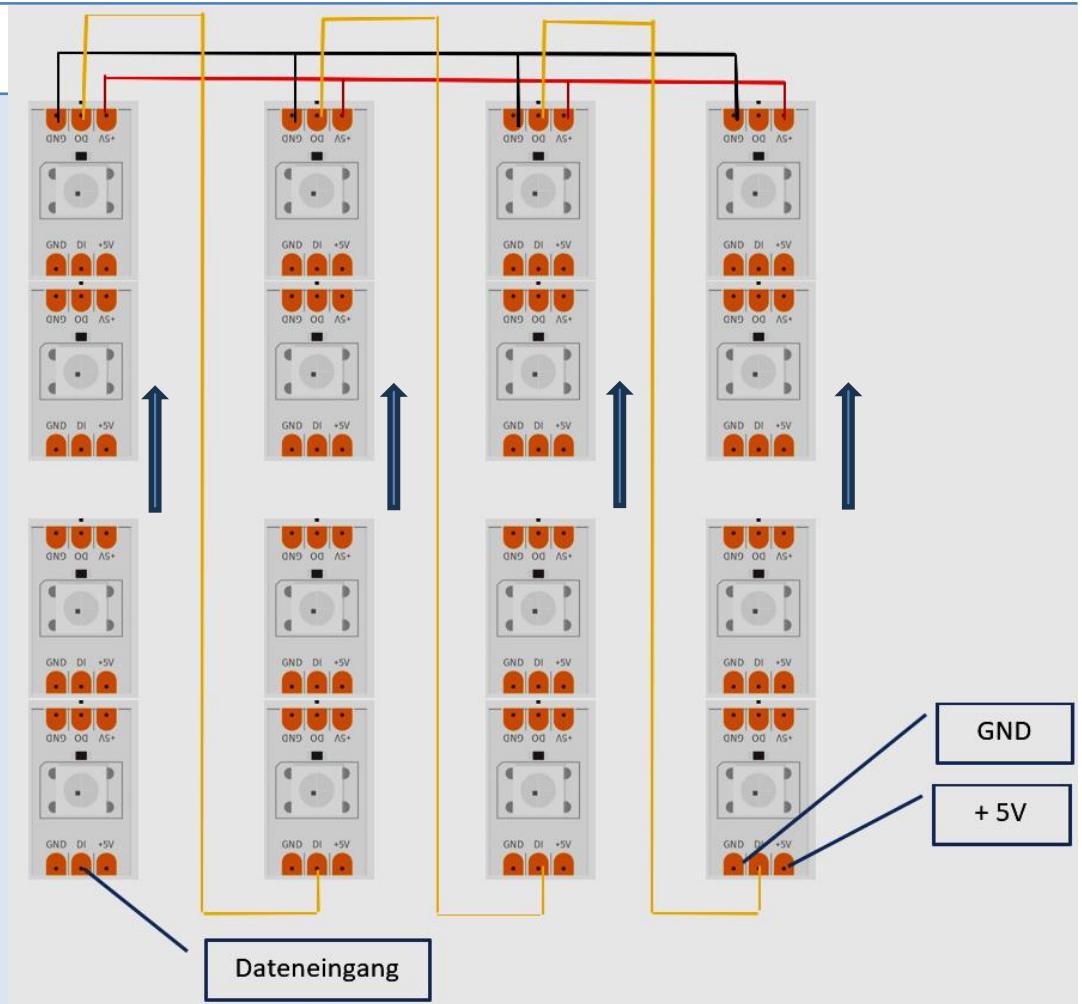


Anordnung Wortuhr

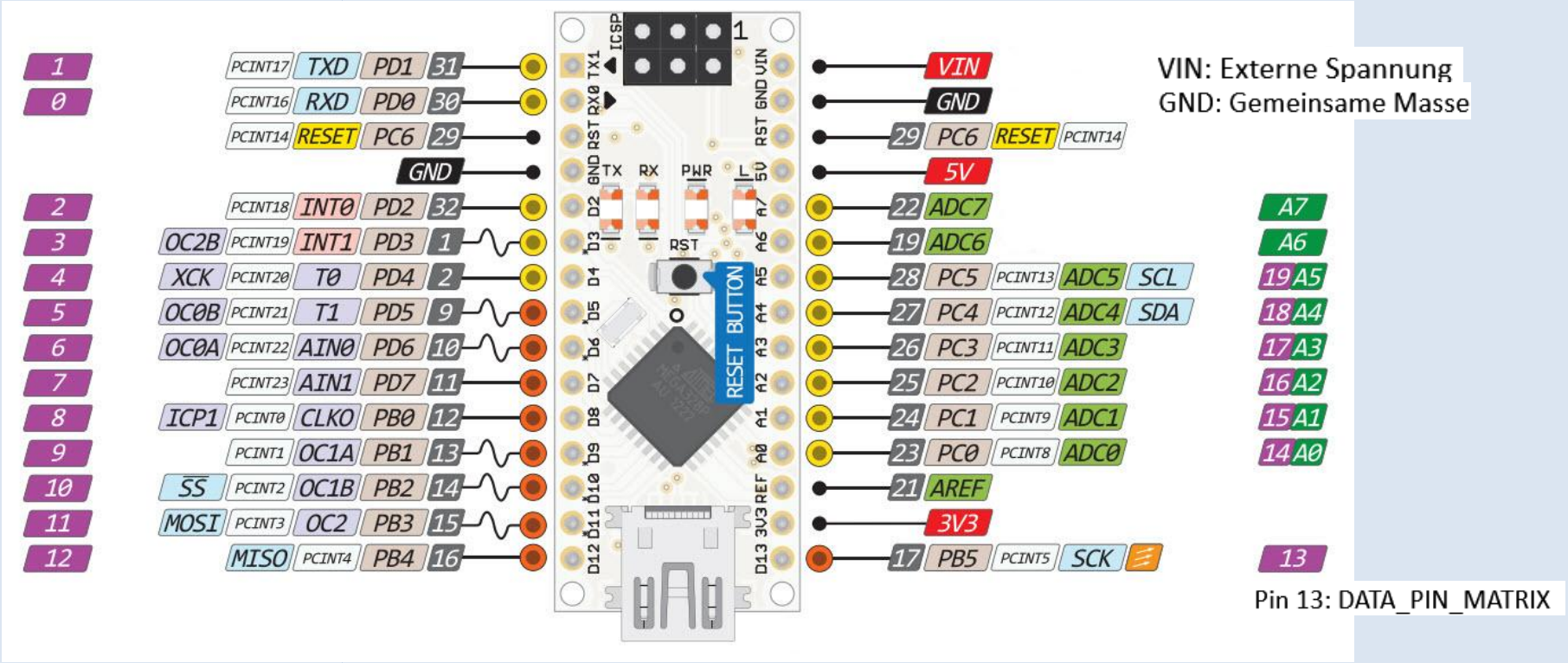
Die Kontakte GND und +5V werden parallel verbunden.
Ein Dateneingang „DI“ wird mit einem digitalem Arduino-Pin verbunden.

Die Datenleitungen müssen seriell verdrahtet werden.
Daher werden die Dateneingänge „DI“ unter Beachtung der Pfeilrichtung verbunden.

In dieser Skizze sind 12 Leds vorhanden.
Diese haben die Adressen von 0 bis 11.

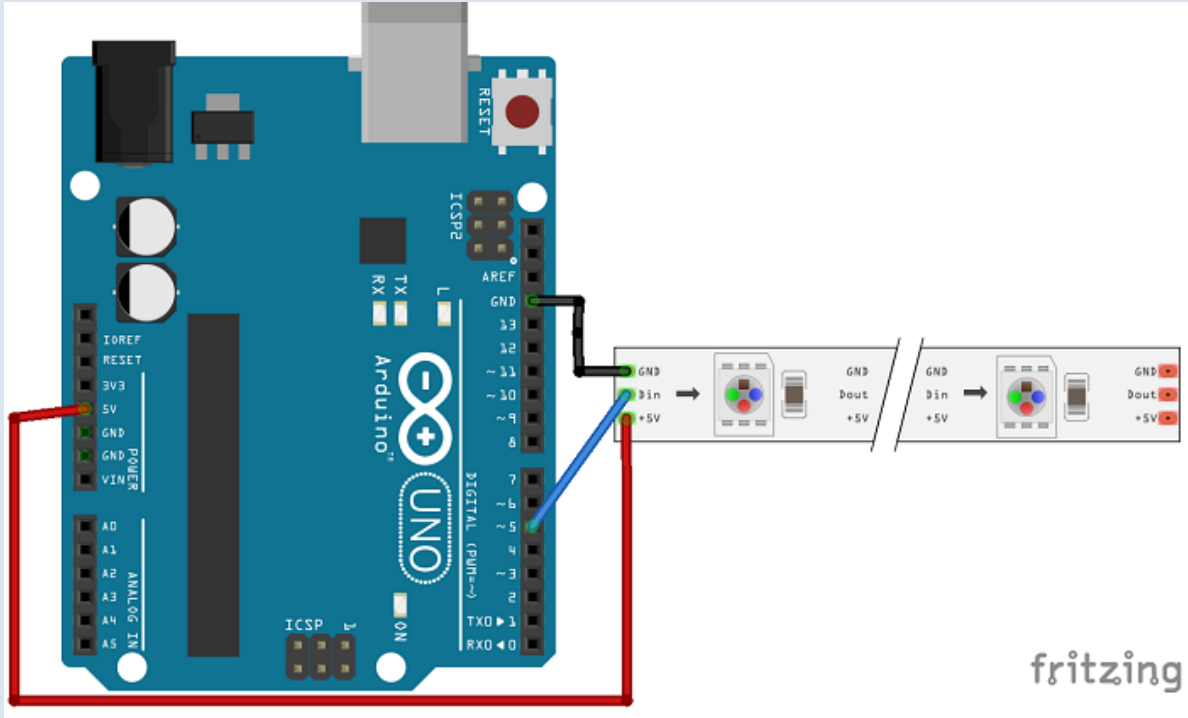


Arduino Nano



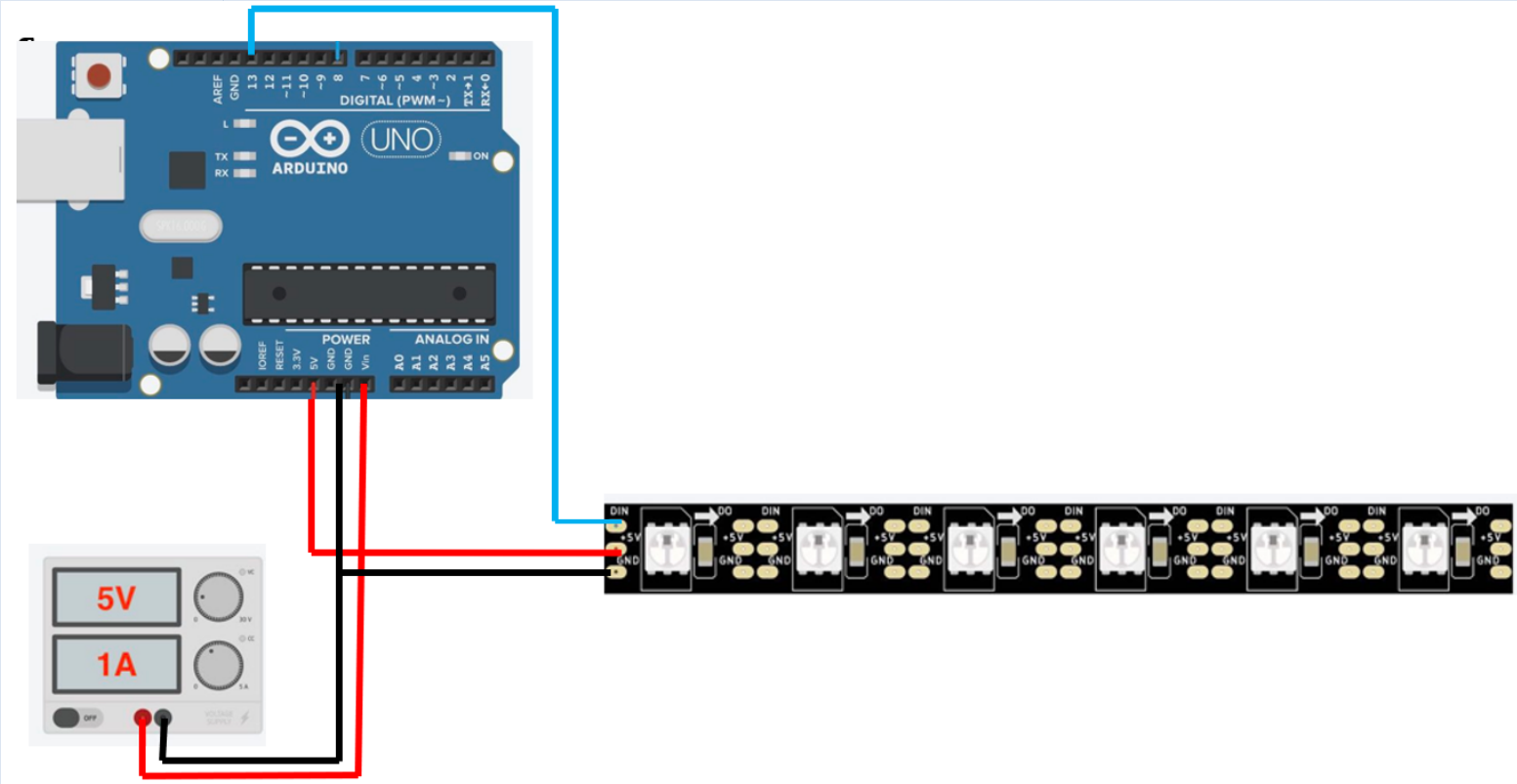
Schaltplan UNO I

Maximal 8 LEDs.
Nicht empfohlen.



Quelle [Guide for WS2812B Addressable RGB LED Strip with Arduino | Random Nerd Tutorials]

Schaltplan UNO II



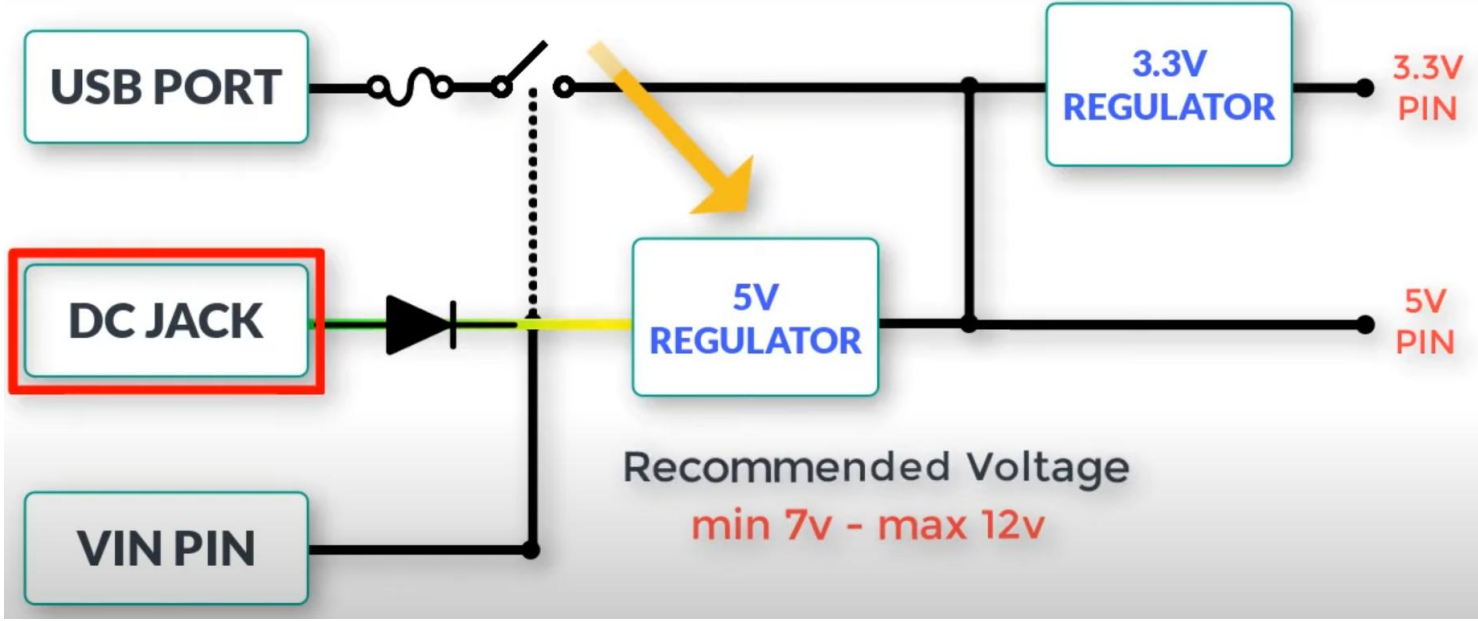
Quelle [Neopixel mit Arduino und ESP32 – der LED-Streifen Ultra Guide]

Stromverbrauch

Eine LED	Als Richtwert rechnet man für eine LED bei voller Intensität mit 20 mA Strom.	
RGB-LED weiß	RGB LEDs haben für jede Farbe eine eigene LED. D.h. wenn alle Farben leuchten, kann die RGB-LED 60 mA verbrauchen.	
Rechenbeispiele	30 RGB LEDs: 60 RGB LEDs: 144 RGB LEDs:	30 * 60 mA = 1800 mA. 60 * 60 mA = 3600 mA 144 * 60 mA = 8640 mA
	Übrigens: wenn der Arduino Uno mit USB gespeist wird, kann der 5V Output noch um die 500 mA liefern.	
Empfehlung	Am besten arbeitet Ihr gleich mit einem externen Netzteil.	
Quelle	https://raydiy.de/neopixel-mit-arduino-und-esp32-der-led-streifen-ultra-guide/	

Eingangsspannung UNO

Welchen Anschluss?	USB Port / DC Jack / VIN



Link	https://www.youtube.com/watch?v=3rbn0pNoGa8 https://www.programmingelectronics.com/power-arduino/

Stromversorgung UNO

Arduino Stromversorgung	
DC-Jack (DC-Buchse)	**7 V bis 12 V (mit Verpolungsschutz) empfohlen.**
USB	5 V (meist kleiner).
VIN	7 V bis 12 V (ohne Verpolungsschutz).
5 V Pin	Als externe Stromversorgung nicht geeignet (> 5 V führt zur Zerstörung).

Steckbrett Stromversorgung	
5 V Pin	5 V (reguliert über den Spannungsregler)
VIN	7 V bis 12 V (in Abhängigkeit von der DC-Buchse). Nicht empfohlen, da Sensoren und Aktoren meist 5 V benötigen.
Link	https://docs.arduino.cc/hardware/uno-rev3
	How do I power my Arduino? | The Pi Hut

Stromstärken UNO

Arduino Stromversorgung	
USB	500 mA (resettable polyfuse)
DC-Jack (DC-Buchse)	500 mA bis 1 A
VIN	500 mA bis 1 A
5V Pin	Als externe Stromversorgung nicht geeignet.

Pins	
pro Pin	20 mA (absolute maximale Stromstärke 40 mA)
Σ digitale & analoge Pins	200 mA (d. h. maximal 10 LEDs mit 20 mA)

Verwendung als Spannungsquelle für das Steckbrett	
5V Pin	500 mA (800 mA, abhängig vom Spannungsregler)
VIN	1 A in Abhängigkeit von der DC-Buchse. Nicht empfohlen!

Mikrokontroller, IDE, Library

Mikrokontroller	Board:	Arduino UNO oder Arduino Nano
	Prozessor:	Atmega328P (5V, 16 MHz)
IDE	Legacy IDE 1.8.19	https://downloads.arduino.cc/arduino-1.8.19-windows.zip
Trick	Ordner „portable“ innerhalb der IDE anlegen. Alle Erweiterungen landen dann in „portable“. Damit wird die Installation portierbar.	
FastLED-Libray	Empfehlung: Am besten über den IDE-Bibliotheksverwalter FastLED einbinden.	
Manuell	Library manuell installieren: https://github.com/fastled/fastled	

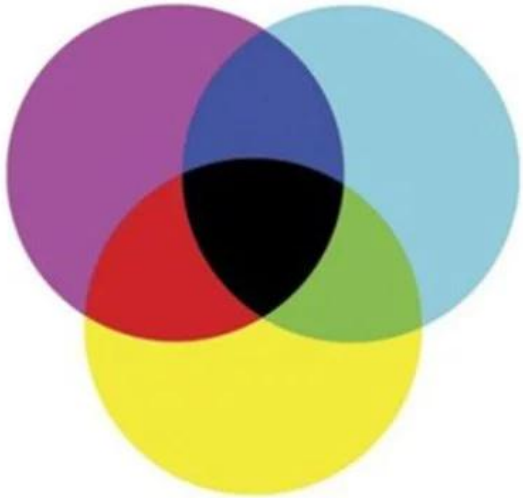
Libraries

Alternative 1	FastLed Library https://github.com/FastLED/FastLED (am besten über den Bibliotheksverwalter „FastLED“ einbinden)
Alternative 2	Adafruits Neopixel Library https://github.com/adafruit/Adafruit_NeoPixel (am besten über den Bibliotheksverwalter „neoPixel strip“ einbinden)
Alternative 3	NeoPixelBus Library https://github.com/Makuna/NeoPixelBus (am besten über den Bibliotheksverwalter NeoPixelBus einbinden)
	(und weitere)
Meine Wahl	FastLed Library.
Quelle	Neopixel mit Arduino und ESP32 – der LED-Streifen Ultra Guide

Farbmischung

Wie funktioniert Additive und Subtraktive Farbmischung?

Additive Farbmischung



Subtraktive Farbmischung

	Additive Farben	Subtraktive Farben
	Rot, grün, blau	Magenta, Cyan, Gelb
	Licht abgebende Quellen (TFT).	Licht absorbierende Flächen (Drucker).

RGB Farben

Farbwert allgemein	rgb(ROT , GRÜN , BLAU)	
	Jeder Parameter definiert die Intensität der Farbe als Ganzzahl zwischen 0 und 255.	
FastLED-Befehl C++	Größte Intensität von ROT:	CRGB(255, 0, 0)
Farbe	CRGB	Name
	CRGB(255, 0, 0)	Red
	CRGB(0, 255, 0)	Green
	CRGB(0, 0, 255)	Blue
	Grautöne werden durch gleiche Werte für alle drei Lichtquellen definiert:	
	CRGB(0, 0, 0)	Black
	CRGB(128, 128, 128)	Gray
	CRGB(255, 255, 255)	White

RGB Farben HEX

Farbwert allgemein	rgb(ROT , GRÜN , BLAU)	
	Jeder Parameter definiert die Intensität der Farbe als HEX-Wert zwischen FF und 0.	
FastLED-Befehl C++	Größte Intensität von ROT:	0xFF0000
Farbe	Parameter	Farbname
	0xFF0000	Red
	0x00FF00	Green
	0x0000FF	Blue
	Grautöne werden durch gleiche Werte für alle drei Lichtquellen definiert:	
	0x000000	Black
	0x808080	Gray
	0xFFFFFF	White

RGB Farben HTML

Farbwert allgemein	rgb(ROT , GRÜN , BLAU)	
	HSL steht für Farbton, Sättigung und Helligkeit.	
FastLED-Befehl C++	Größte Intensität von ROT:	CRGB::Red
Farbe	Parameter	Farbname
	CRGB::Red	Red
	CRGB::Green	Green
	CRGB::Blue	Blue
	Grautöne werden durch gleiche Werte für alle drei Lichtquellen definiert:	
	CRGB::Black	Black
	CRGB::Gray	Gray
	CRGB::White	White

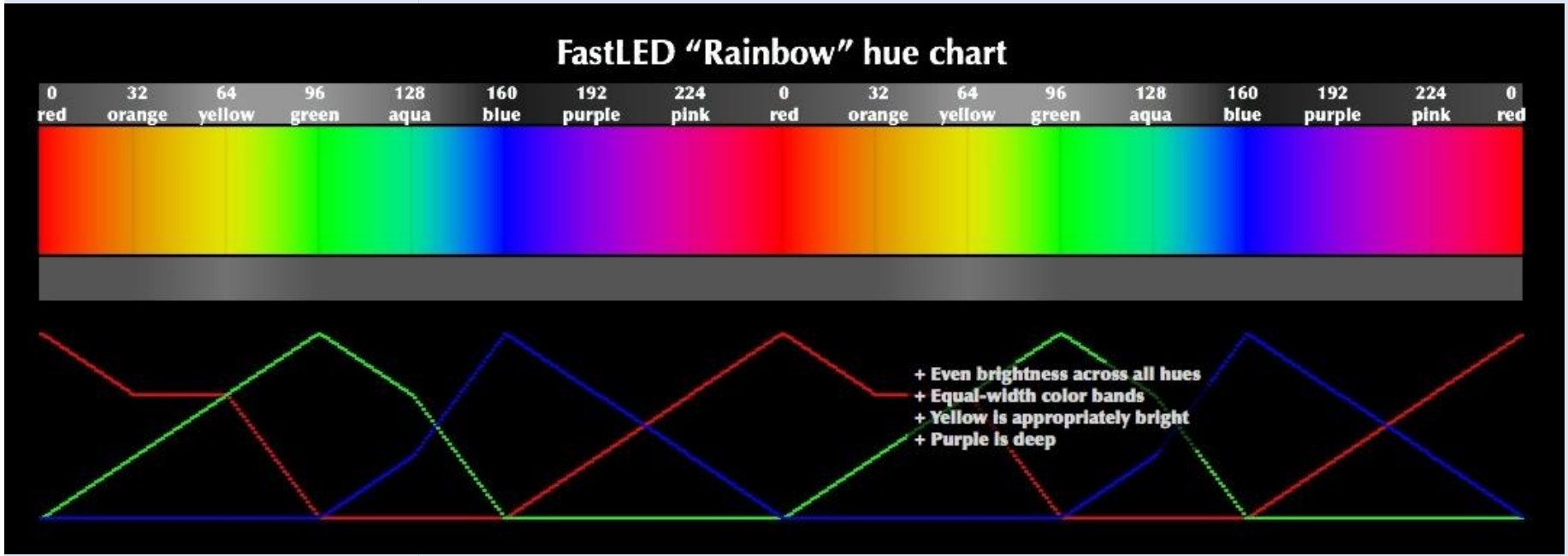
HSL Code

Farbwert allgemein	hsl (Farbton, Sättigung , Helligkeit)
englisch	HSL steht für hue, saturation and lightness.
deutsch	HSL steht für Farbton, Sättigung und Helligkeit.
Farbton hue	Der Farbton ist ein Grad auf dem Farbkreis von 0 bis 360. 0° (oder 360°) ist Rot, 120° ist Grün, 240° ist Blau
Sättigung saturation	Die Sättigung kann als Intensität einer Farbe beschrieben werden. Es handelt sich um einen Prozentwert von 0 % bis 100 %.
	100 % ist Vollfarbe, keine Grautöne. 50 % sind 50 % Grau, aber Sie können die Farbe immer noch sehen. 0 % ist komplett grau, die Farbe ist nicht mehr zu sehen.
Helligkeit value/lightness	Wie viel Licht möchte man der Farbe geben, wobei 0 % kein Licht (dunkel) bedeutet, 50 % 50 % Licht (weder dunkel noch hell) und 100 % volles Licht bedeutet.

FastLED HSV

HSV Definition	hsl(hue, saturation, lightness)	
HSV-Farbmodell	• hue von 0 bis 360 Grad. • saturation und lightness als Prozentsatz von 0 bis 100 %.	
FastLED	**Verwemdet nicht die traditionellen HSV-Werte!**	
FastLED-Befehl C++	CHSV (hue, sat, value);	// mit umgerechneten Werten
Umrechnung	hsl traditionell	CHSV
	hue 0 bis 360°	hue 0 bis 255
	sat 0 bis 100 %	sat 0 bis 255
	value 0 bis 100 %	value 0 bis 255
	Weiterhin verwendet FastLED standardmäßig die „Regenbogen“-Farbkarte statt der herkömmlichen „Spektrum“-Farbkarte.	
Abhilfe	Umrechnung der Werte mit C++ Funktionen.	
Quelle	FastLED HSV-Farben · FastLED/FastLED Wiki · GitHub	

FastLED HSV

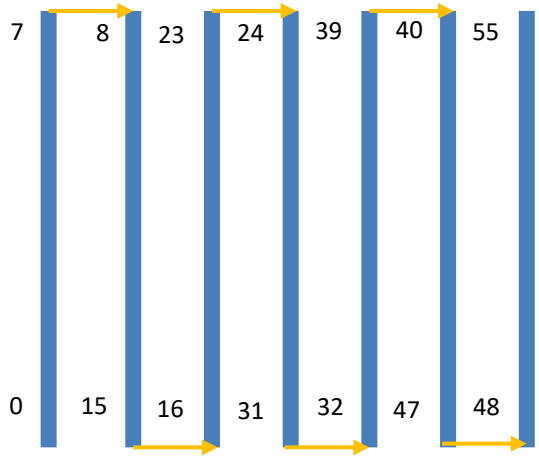


Quelle [FastLED HSV-Farben · FastLED/FastLED Wiki · GitHub]

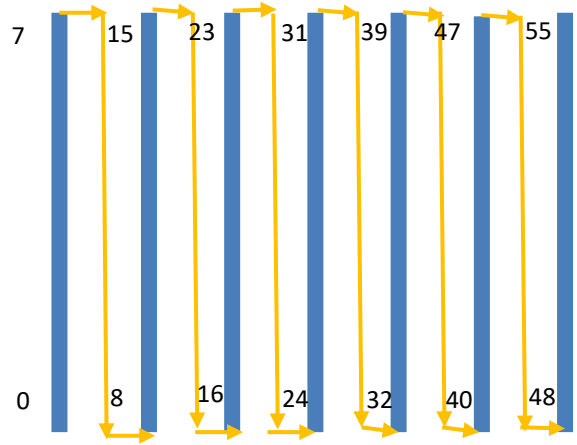
Anordnung Lauflicht

Anordnung der LED-Streifen

Fortlaufend



Matrix



05_Moving_Led.ino

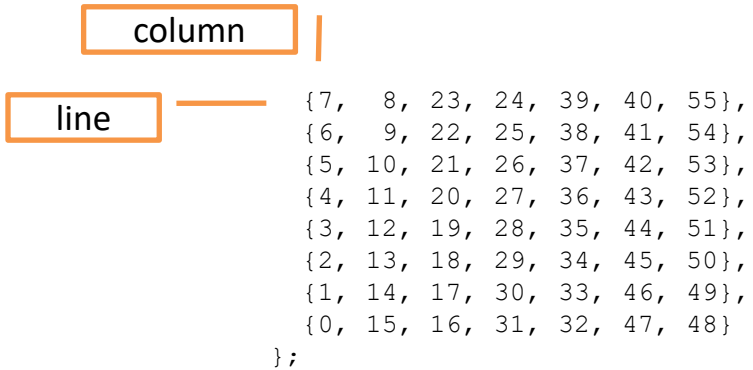
```
for(int myLed = 0; myLed < 56; myLed++) {
    // Turn our current led on to Blue, then show the leds
    leds[myLed] = CRGB::Blue;

    FastLED.show();       // show the leds
    delay(200);           // wait a little bit

    // Turn our current led back to black for the next loop around
    leds[myLed] = CRGB::Black;
  }
```

LED-Adressen

	LED-Adressen

C++ Array	`// pattern of adresses`
	`byte address[8][7]={`
	 
	
	`{7, 8, 23, 24, 39, 40, 55},`
	`{6, 9, 22, 25, 38, 41, 54},`
	`{5, 10, 21, 26, 37, 42, 53},`
	`{4, 11, 20, 27, 36, 43, 52},`
	`{3, 12, 19, 28, 35, 44, 51},`
	`{2, 13, 18, 29, 34, 45, 50},`
	`{1, 14, 17, 30, 33, 46, 49},`
	`{0, 15, 16, 31, 32, 47, 48}`
	`};`

	Das Array „address[8][7]“ ist zweidimensional. Die erste Dimension steht für eine waagerechte Reihe. Die zweite Dimension steht für die Spalten. Darin die Werte der Adressen.
Variablen in Algorithmus	`address[line][col]`

Figuren

	LED-Adressen	Baummuster (pattern)
C++ Array	`// pattern of adresses` `byte address[8][7]={`	

column 0 / column 6 / line 0 / line 7	`{7, 8, 23, 24, 39, 40, 55},`
	`{6, 9, 22, 25, 38, 41, 54},`
	`{5, 10, 21, 26, 37, 42, 53},`
	`{4, 11, 20, 27, 36, 43, 52},`
	`{3, 12, 19, 28, 35, 44, 51},`
	`{2, 13, 18, 29, 34, 45, 50},`
	`{1, 14, 17, 30, 33, 46, 49},`
	`{0, 15, 16, 31, 32, 47, 48} };`
		`// pattern of a tree`
		`byte tree[8][7]={`
		`{0, 0, 0, 1, 0, 0, 0},`
		`{0, 0, 1, 0, 1, 0, 0},`
		`{0, 1, 0, 0, 0, 1, 0},`
		`{1, 0, 0, 0, 0, 0, 1},`
		`{1, 1, 1, 1, 1, 1, 1},`
		`{0, 0, 0, 1, 0, 0, 0},`
		`{0, 0, 0, 1, 0, 0, 0},`
		`{0, 0, 1, 1, 1, 0, 0} };`

	address[8][7] ist zweidimensional	tree[8][7] ist ebenfalls zweidimensional

Algorithmus
06_Letters.ino

```
// start with line one
for (int line = 0; line < 8; line++){
  // for line one step through the columns of the line
  for (int col = 0; col < 7; col++){
    int led_address;
    if ( tree[line][col] == 1 ) {
      led_address = address[line][col];
      leds[led_address] = CRGB::Green;
    }
  }
```

Erster Sketch I

00_First_Trial.ino

```
// EBW: 00_First_Trial.ino

// including the previous installed library
#include <FastLED.h>

#define DATA_PIN    (13)               // replace number with your Arduino-Pin
#define NUM_LEDS    (4)                // replace number with your number of leds
#define LED_TYPE    WS2812B            // replace Type with your present type
#define COLOR_ORDER GRB

// ----------------------------------nothing to change-------------------------------
// set up a block of memory as an array
// this means e.g. leds[12] is the address where led-data are stored
CRGB leds[NUM_LEDS];                   // define an array of leds

void setup() {
  // sanity check delay - allows reprogramming if accidently blowing power w/leds
  delay(2000);
  Serial.begin(9600);                  // setup serial monitor for messages
  Serial.println("EBW Ready to go!");

  // initialize FastLED-library
  // possible types in my case: WS2812B, NEOPIXEL
  FastLED.addLeds<LED_TYPE, DATA_PIN, COLOR_ORDER>(leds, NUM_LEDS);

  FastLED.setBrightness(50);           // overall setting of brightness
}
// ----------------------------------end of nothing to change-----------------------
```

Erster Sketch I

00_First_Trial.ino

```
void loop() {
  FastLED.clear();                    // clear the data-package
  delay(500);

  // https://www.w3schools.com/colors/colors_names.asp
  // set color via any named HTML web color
  leds[0] = CRGB::Magenta;    delay(200); FastLED.show();
  leds[1] = CRGB::RoyalBlue;  delay(200); FastLED.show();
  leds[2] = CRGB::Green ;     delay(200); FastLED.show();
  leds[3] = CRGB::Yellow;     delay(200); FastLED.show();
  leds[3] = CRGB::Black;      delay(200); FastLED.show();
  leds[2] = CRGB::Black;      delay(200); FastLED.show();
  leds[1] = CRGB::Black ;     delay(200); FastLED.show();
  leds[0] = CRGB::Black;      delay(200); FastLED.show();
}
```

Links
https://raydiy.de/neopixel-mit-arduino-und-esp32-der-led-streifen-ultra-guide/
Arduino und WS2812 – Viele LEDs einzeln mit Arduino steuern
Basic usage · FastLED/FastLED Wiki · GitHub
The Magic of NeoPixels | Adafruit NeoPixel Überguide | Adafruit Learning System
FastLED Basics Episode 1 - Getting started
https://github.com/FastLED/FastLED
Colors Tutorial
HTML Color Picker
Guide for WS2812B Addressable RGB LED Strip with Arduino | Random Nerd Tutorials

E-Mail

Sketch auf Anfrage von: H39@email.de