16 x 16 RGB-LED-Matrix

Wir programmieren eine 16 x 16 RGB-Matrix, mischen eigene Farben, arbeiten mit Koordinaten und bringen eigene PixelArt zum Leuchten. Extra: Animationen.

Dein Weg zur Lichtdesignerin

Schritt	Das machen wir
1. Einrichten	Arduino, Stromversorgung und Matrix verbinden
2. Erstes Programm	LED Nummer 0 leuchtet weiß
3. Farben	Eigene RGB-Farben ausprobieren
4. Koordinaten	Aus x und y die richtige LED berechnen
5. PixelArt laden	Ein RGB-Array aus dem Browser auf die Matrix übertragen
6. Challenge	Eigene PixelArt entwerfen, testen und verbessern
Extra	Wenn Zeit bleibt: Animationen programmieren

SCHRITT 1

Einrichten: Arduino + Matrix

Material

Arduino Uno

USB-Kabel

Netzteil

USB-C Pin-Adapter

16 x 16 Matrix

Matrix-Stecker

Jumper-Kabel

Klemme

Aufbau Schritt für Schritt

Der **USB-C Pin-Adapter** hat die Anschlüsse `V`, `D−`, `D+` und `G`. Für die Stromversorgung benutzen wir `V` als Plus / `5V` und `G` als Minus / `GND`. `D−` und `D+` bleiben bei unserem Aufbau frei. **Jumper-Kabel** sind die kleinen Verbindungskabel zwischen Arduino, Adapter, Matrix-Stecker und Klemme.

1. Verbinde den **Arduino Uno** mit dem **USB-Kabel** mit dem Computer.
2. Schließe das **Netzteil** an den **USB-C Pin-Adapter** an.
3. Verbinde die **16 x 16 Matrix** mit dem **Matrix-Stecker**.
4. Lege die **Klemme** bereit. Sie sammelt alle `GND`-Verbindungen.
5. Verbinde mit einem **Jumper-Kabel** `GND` vom Arduino mit der Klemme.
6. Verbinde mit einem **Jumper-Kabel** `GND` / `−` der Matrix mit derselben Klemme.
7. Verbinde mit einem **Jumper-Kabel** `G` vom USB-C Pin-Adapter mit derselben Klemme.
8. Verbinde mit einem **Jumper-Kabel** `V` vom USB-C Pin-Adapter mit `+` / `5V` der Matrix.
9. Verbinde mit einem **Jumper-Kabel Arduino Pin** `8` mit dem **Datenpin** `DIN` der Matrix.

SCHRITT 2

Erstes Programm: Eine LED leuchtet

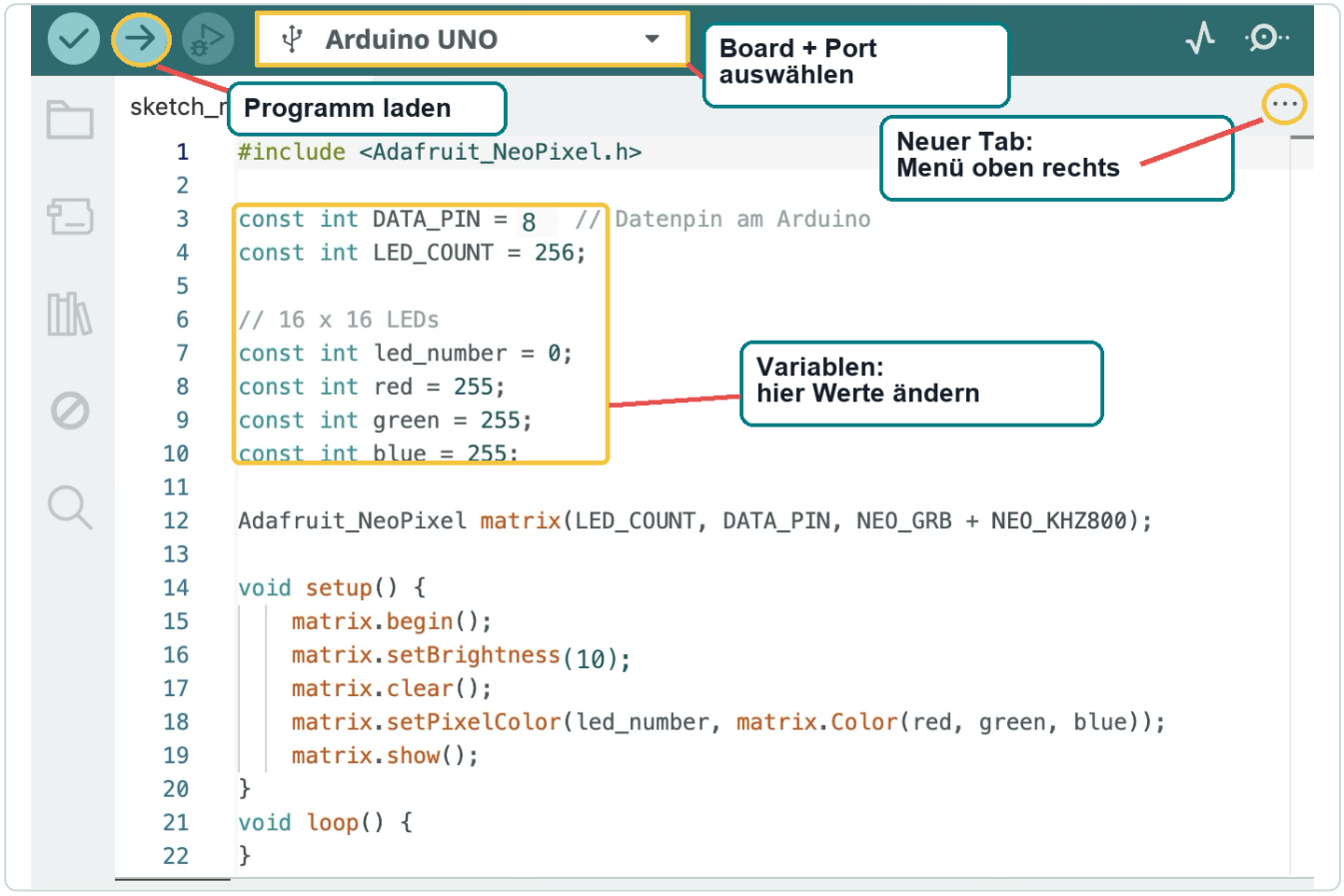
Ziel: Bringe LED Nummer `0` zum weißen Leuchten.

Ein Programm läuft auf dem Arduino immer wieder in zwei Teilen:

- `setup()` wird einmal beim Start ausgeführt.
- `loop()` wird anschließend immer wieder wiederholt.

Arduino IDE starten und Programm hochladen

1. Drücke die **Windows-Taste**.
2. Tippe `arduino` ein und öffne die **Arduino IDE**.
3. Wähle oben **Arduino UNO** und den passenden **Port** aus.
4. Über das Menü oben rechts kannst du **neue Tabs** anlegen.
5. Klicke auf **Upload**, um das Programm auf den Arduino zu laden.



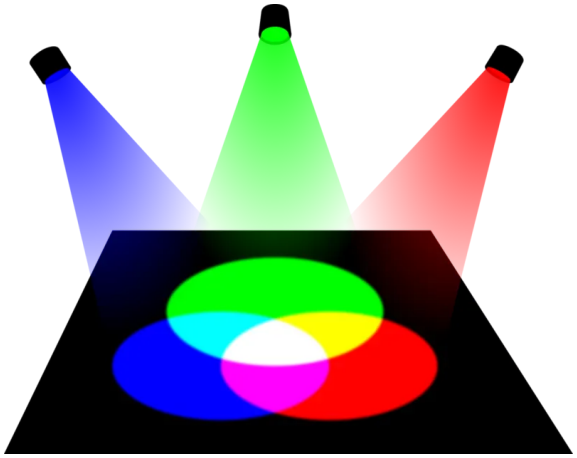
Hurra, die erste LED leuchtet!

SCHRITT 3

Farbdarstellung: RGB mischen

Jede LED enthält drei kleine Lichtquellen: **R**ot, **G**rün und **B**lau. Der Computer steuert jede dieser Lichtquellen mit einer Zahl. Computer arbeiten intern nur mit `0` und `1`. Eine einzelne solche Stelle heißt **Bit**. Acht Bits zusammen heißen **Byte**. Ein Byte verwendet der Computer als kleinste Speichereinheit für solche Zahlen. Mit 8 Bits gibt es `2^8 = 256` verschiedene Möglichkeiten. Deshalb passt in ein Byte eine Zahl von `0` bis `255`.

Für eine RGB-Farbe benutzen wir drei Werte: ein Byte für Rot, ein Byte für Grün und ein Byte für Blau. Zusammen sind das **3 Byte**. Damit entstehen `256 * 256 * 256 = 16.777.216` mögliche Farbkombinationen.



Rot, Grün und Blau mischen sich zu weiteren Farben.

Auch ein Bildschirm besteht aus winzigen RGB-Lichtpunkten.

In Schritt 2 änderst du für neue Farben nur diese drei Variablen:

```
const int red = 255;
const int green = 255;
const int blue = 255;
```

Eigene Farben testen

Farbname	red	green	blue	Notizen / Ergebnis
Sonnenorange	255	100	0	Startbeispiel.

SCHRITT 4

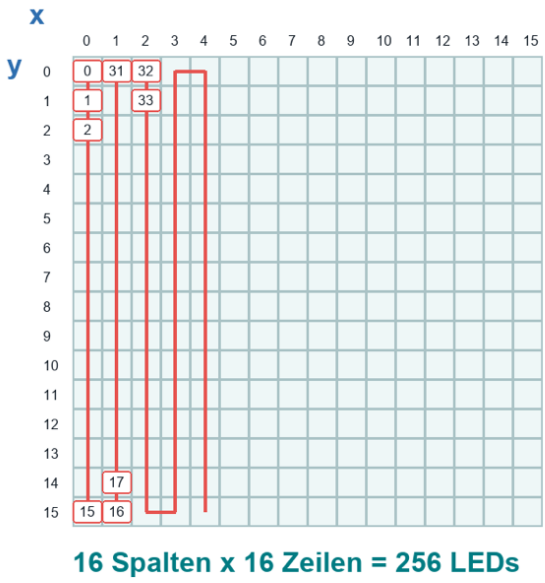
Von LED-Nummern zu x und y

Bei dieser Matrix läuft die Verdrahtung spaltenweise: zuerst von oben nach unten, in der nächsten Spalte von unten nach oben. Dadurch entsteht eine Schlangenlinie. Für PixelArt ist diese Reihenfolge unpraktisch, weil wir normalerweise mit Koordinaten arbeiten.

`x` beschreibt die Spalte von links nach rechts.

`y` beschreibt die Zeile von oben nach unten.

Die Funktion `XY()` berechnet daraus die passende LED-Nummer.



16 Spalten x 16 Zeilen = 256 LEDs

XY-Funktion einfügen

Füge diese Funktion **zwischen den Variablen und 'setup()'** ein.

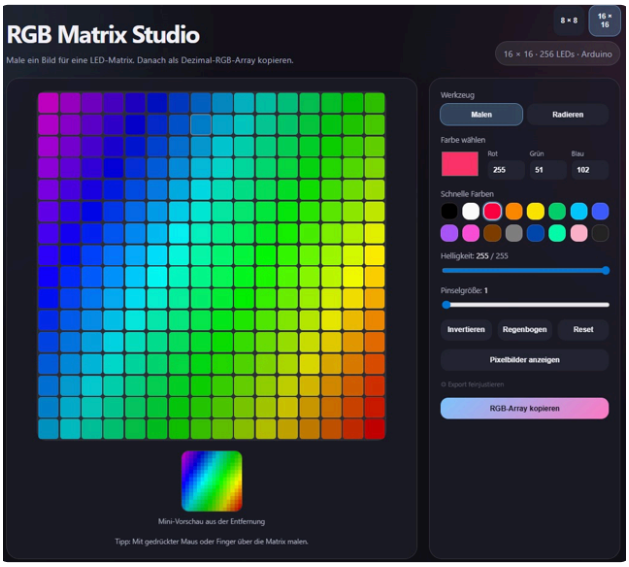
```
int XY(int column, int row) {
  if (column % 2 == 0) {
    return column * 16 + row;
  }

  return column * 16 + (15 - row);
}
```

Jetzt nutzen wir zwei Schleifen: Die äußere Schleife geht durch die Zeilen, die innere durch die Spalten.

```
void loop() {
  for (int y = 0; y < 16; y++) {
    for (int x = 0; x < 16; x++) {
      matrix.clear();
      matrix.setPixelColor(XY(x, y), matrix.Color(red, green, blue));
      matrix.show();
      delay(40);
    }
  }
}
```

SCHRITT 5

Browser: PixelArt laden



Bilddaten kopieren

1. Öffne unter **Downloads** die Datei **Matrix_Studio**. Sie öffnet sich im Browser. Weitere Motive findest du über **PixelArt anzeigen**.
2. Klicke unten rechts auf **RGB-Array kopieren**.

Arduino vorbereiten

Lege in der Arduino IDE einen neuen Tab an und nenne ihn **bild.h**. Füge dort mit **Strg + V** den kopierten Array-Code ein. Ganz oben im normalen Arduino-Tab kommt diese Zeile hin:

```
#include "bild.h"
```

Bild anzeigen

Das Array enthält für jedes Pixel drei Werte: `bild[y][x][0]` für Rot, `bild[y][x][1]` für Grün und `bild[y][x][2]` für Blau. `matrix.show()` steht erst nach den Schleifen, damit das komplette Bild auf einmal erscheint.

```
void loop() {
  matrix.clear();

  for (int y = 0; y < 16; y++) {
    for (int x = 0; x < 16; x++) {
      matrix.setPixelColor(XY(x, y), matrix.Color(
        bild[y][x][0], bild[y][x][1], bild[y][x][2]
      ));
    }
  }

  matrix.show();
}
```

SCHRITT 6

Challenge: Deine eigene PixelArt

Gestalte ein eigenes Bild oder Emoji. Auf 16 x 16 LEDs funktionieren klare Formen, starke Kontraste und wenige Farben besonders gut.

Entwurf

Name meiner PixelArt

Meine Hauptfarben

Woran erkennt man das Motiv?

Was soll auf der Matrix besonders gut wirken?

Testen und verbessern

Das ändere ich

Warum?