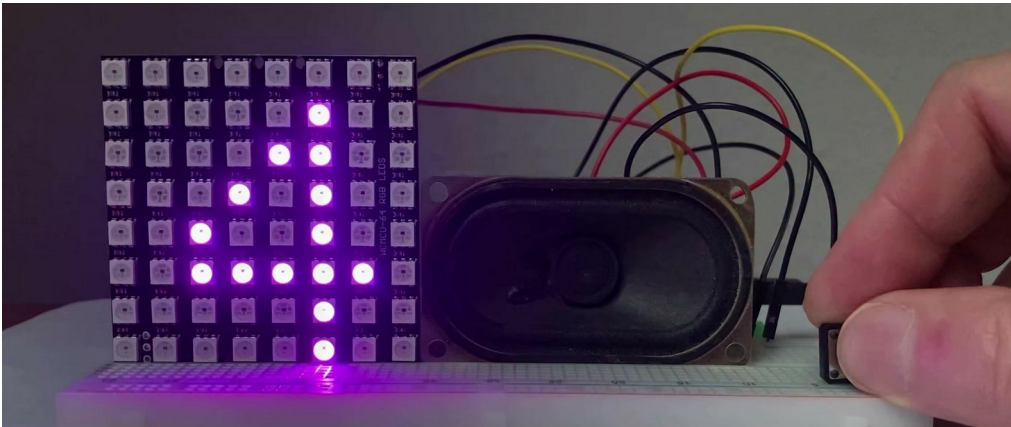


## Inhaltsverzeichnis

|                           |   |
|---------------------------|---|
| Ziel des Programms.....   | 2 |
| Benötigte Bauteile.....   | 2 |
| Schaltplan.....           | 2 |
| Das Programm.....         | 3 |
| Benötigte Bibliothek..... | 3 |
| Der setup-Teil.....       | 4 |
| Der loop-Teil.....        | 5 |

## Ziel des Programms

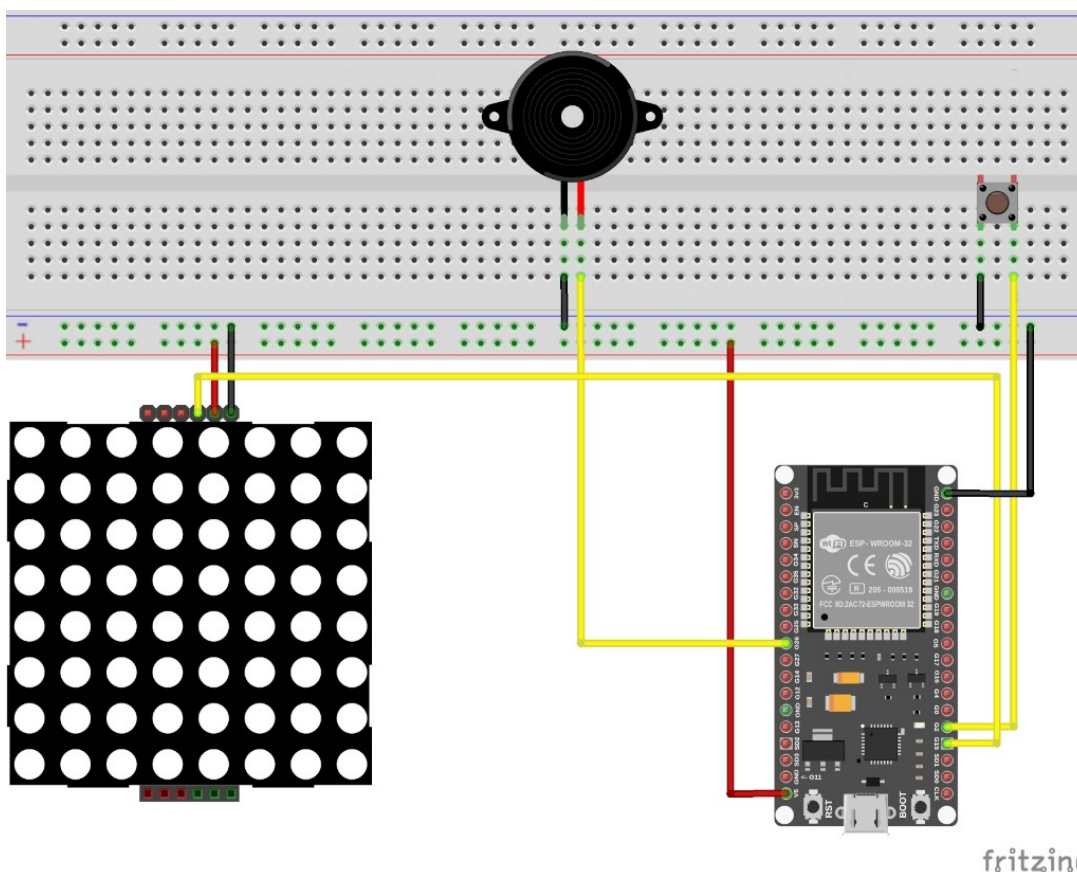


Wird der Taster gedrückt zeigt die RGB-Matrix in zufälliger Reihenfolge und in verschiedenen Farben Zahlen zwischen 1 und 9. Zusätzlich wird der Wechsel der Zahlen mit einem kurzen Ton ergänzt. Wenn der Taster losgelassen wird, vergleicht das Programm die letzten beiden Zahlen und zeigt die Zahlen an.

## Benötigte Bauteile

- ➔ RGB-Matrix 8x8
- ➔ Taster
- ➔ Lautsprecher
- ➔ Leitungsdrähte

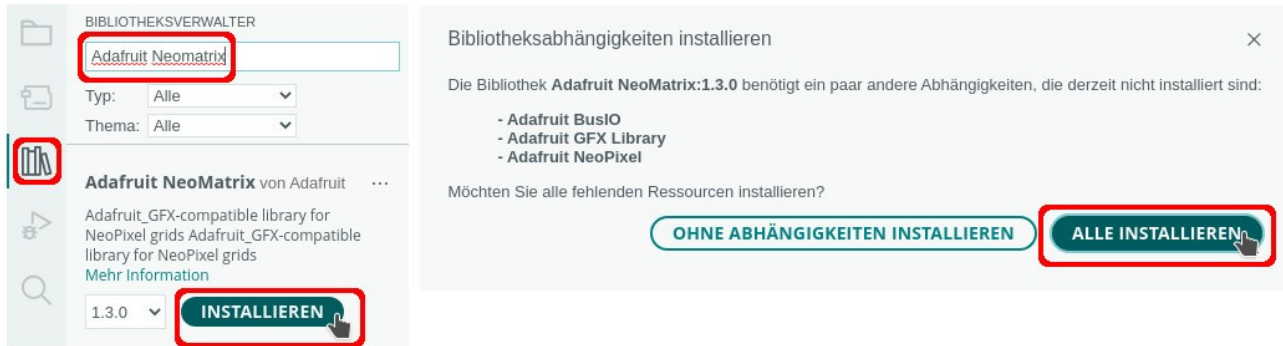
## Schaltplan



Das Programm wird mit einem ESP32-Wroom realisiert, es lässt sich aber leicht auf andere Mikrocontroller übertragen.

## Das Programm

### Benötigte Bibliothek



Benötigte Bibliotheken einbinden und Variablen definieren

```
#include "Adafruit_NeoMatrix.h"

// Pins ESP32-Wroom, für UNO anpassen
#define RGBMatrixPin 15
#define TASTER 2
#define LAUTSPRECHER 26

// RGBMatrix -> Name der RGB-Matrix
/*
  die wichtigsten Parameter:
  Parameter 1 = Breite der Matrix (8)
  Parameter 2 = Höhe der Matrix (8)
  Parameter 3 = Name des Daten-Pins (RGBMatrixPin)
*/
Adafruit_NeoMatrix RGBMatrix = Adafruit_NeoMatrix
(
  8, 8, RGBMatrixPin,
  NEO_MATRIX_TOP + NEO_MATRIX_RIGHT + NEO_MATRIX_COLUMNS +
  NEO_MATRIX_PROGRESSIVE,
  NEO_GRB + NEO_KHZ800
);

// Farben definieren
#define Rot RGBMatrix.Color(255, 0, 0)
#define Gruen RGBMatrix.Color(0, 255, 0)
#define Blau RGBMatrix.Color(0, 0, 255)
#define Magenta RGBMatrix.Color(139, 0, 139)
#define Pink RGBMatrix.Color(255, 20, 147)
#define Weiss RGBMatrix.Color(255, 255, 255)
#define Gelb RGBMatrix.Color(255, 255, 0)
```

```
#define Schwarz RGBMatrix.Color(0, 0, 0)
#define Rosa RGBMatrix.Color(236, 170, 170)
#define Orange RGBMatrix.Color(255, 165, 0)

// Variablen: Farbe -> Farbe der Zahl, Zahl -> zufällig ermittelte Zahl
int Farbe;
int Zahl;

// Array für die Zufallszahlen
int ZahlWert[2];

// Startwert für die Anzahl der Durchläufe
int Durchlauf = 0;

// Variable für den Zustand des Tasters
bool TasterGedrueckt = true;
```

## Der setup-Teil

```
void setup()
{
    pinMode(TASTER, INPUT_PULLUP);

    // Helligkeit der RGBMatrix
    RGBMatrix.setBrightness(30);

    // RGBMatrix starten
    RGBMatrix.begin();

    // Zufallsgenerator starten
    randomSeed(analogRead(0));

    // Pfeil anzeigen
    RGBMatrix.drawFastHLine(0, 3, 8, Gelb);
    RGBMatrix.drawFastHLine(0, 4, 8, Gelb);
    RGBMatrix.drawPixel(6, 2, Gelb);
    RGBMatrix.drawPixel(6, 5, Gelb);
    RGBMatrix.show();
}
```

## Der loop-Teil

```
void loop()
{
    // solange der Taster gedrückt wird laufen zufällige Zahlen
    if (!digitalRead(TASTER))
    {
        TasterGedrueckt = false;
        RGBMatrix.clear();

        // ASCII-Werte der Zahlen: 49 = 1 ... 57 = 9
        Zahl = random(49, 57);

        // Farben der Zahlen definieren
        switch (Zahl)
        {
            case 49:
                Farbe = Rot;
                break;

            case 50:
                Farbe = Gruen;
                break;

            case 51:
                Farbe = Blau;
                break;

            case 52:
                Farbe = Magenta;
                break;

            case 53:
                Farbe = Pink;
                break;

            case 54:
                Farbe = Weiss;
                break;

            case 55:
                Farbe = Gelb;
                break;

            case 56:
                Farbe = Rosa;
                break;
```

```

        case 57:
            Farbe = Orange;
            break;
    }

    // Zahl anzeigen
    RGBMatrix.drawChar(2, 1, char(Zahl), Farbe, Schwarz, 1);
    RGBMatrix.show();
    tone(LAUTSPRECHER, 1000, 30);
    delay(100);
}

// wenn der Taster nicht gedrückt wurde
// und (&&) TasterGedruickt false ist
if (digitalRead(TASTER) && !TasterGedruickt)
{
    TasterGedruickt = true;

    // aktuelle Zahl im Array speichern
    ZahlWert[Durchlauf] = Zahl;

    // wenn zwei Zahlen im Array gleich sind
    if (ZahlWert[0] == ZahlWert[1])
    {
        RGBMatrix.clear();

        // Pfeil anzeigen
        RGBMatrix.drawFastHLine(0, 3, 8, Gelb);
        RGBMatrix.drawFastHLine(0, 4, 8, Gelb);
        RGBMatrix.drawPixel(6, 2, Gelb);
        RGBMatrix.drawPixel(6, 5, Gelb);
        RGBMatrix.show();

        delay(1000);
        RGBMatrix.clear();

        // identische Zahlen anzeigen
        // erste Zahl anzeigen
        RGBMatrix.drawChar(2, 1, char(ZahlWert[0]), Farbe, Schwarz, 1);
        RGBMatrix.show();
        delay(2000);
        RGBMatrix.clear();

        // Pfeil anzeigen
        RGBMatrix.drawFastHLine(0, 3, 8, Gelb);
        RGBMatrix.drawFastHLine(0, 4, 8, Gelb);
        RGBMatrix.drawPixel(6, 2, Gelb);
        RGBMatrix.drawPixel(6, 5, Gelb);
        RGBMatrix.show();
        delay(2000);
        RGBMatrix.clear();
    }
}

```

```
// zweite Zahl anzeigen
RGBMatrix.drawChar(2, 1, char(ZahlWert[1]), Farbe, Schwarz, 1);
RGBMatrix.show();
delay(2000);

// Smiley
RGBMatrix.clear();
RGBMatrix.drawCircle(4, 4, 3, Gelb);
RGBMatrix.drawPixel(3, 3, Gelb);
RGBMatrix.drawPixel(5, 3, Gelb);
RGBMatrix.show();

// Melodie abspielen
tone(LAUTSPRECHER, 100, 100);
tone(LAUTSPRECHER, 200, 100);
tone(LAUTSPRECHER, 300, 100);
tone(LAUTSPRECHER, 400, 100);
tone(LAUTSPRECHER, 300, 100);
tone(LAUTSPRECHER, 200, 100);
tone(LAUTSPRECHER, 100, 100);
}

// Durchlauf erhöhen
Durchlauf ++;

// wenn Durchlauf > 1 -> Werte zurücksetzen
if (Durchlauf > 1)
{
    Durchlauf = 0;
    ZahlWert[0] = NULL;
    ZahlWert[1] = NULL;
}
}
```