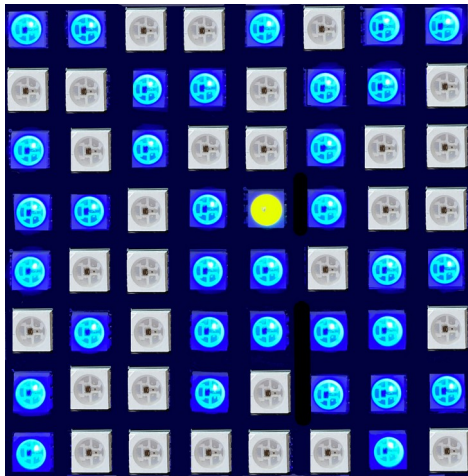


Inhaltsverzeichnis

Ziel des Projekts.....	2
Die Hardware.....	2
Benötigte Bauteile.....	2
Der Schaltplan.....	3
Benötigte Bibliotheken.....	3
Beispiele.....	6
Lauflicht.....	6
Grafikfunktionen.....	7
Spiel.....	10



Ziel des Projekts

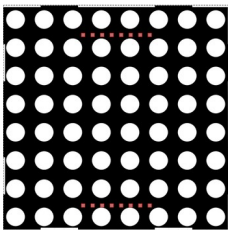


Auf einer RGB-Matrix leuchten zufällig eine Reihe von LEDs. In der Mitte leuchtet beim Start eine gelbe LED. Das Tastenpad „bewegt“ diese LED über die RGB-Matrix und „löscht“ dabei die LED auf der aktuellen Position.

Wenn die äußere rechte Taste des Tastenpads gedrückt wird, zeigt der Serielle Monitor die benötigte Zeit und die Anzahl der Klicks an.



Die Hardware



Die RGB-Matrix besteht aus miteinander verbundenen RGB-LEDs. Jede besitzt einen eigenen Controller und kann einzeln angesteuert werden. Die RGB-Matrix benötigt nur einen digitalen Eingang.

RGB ist eine Mischung der Farben Rot, Grün und Blau. Jede Farbe kann von 0 bis 255 gesetzt werden, die Werte werden durch Kommata getrennt.

Beispiele:

163, 0, 93

226, 176, 50

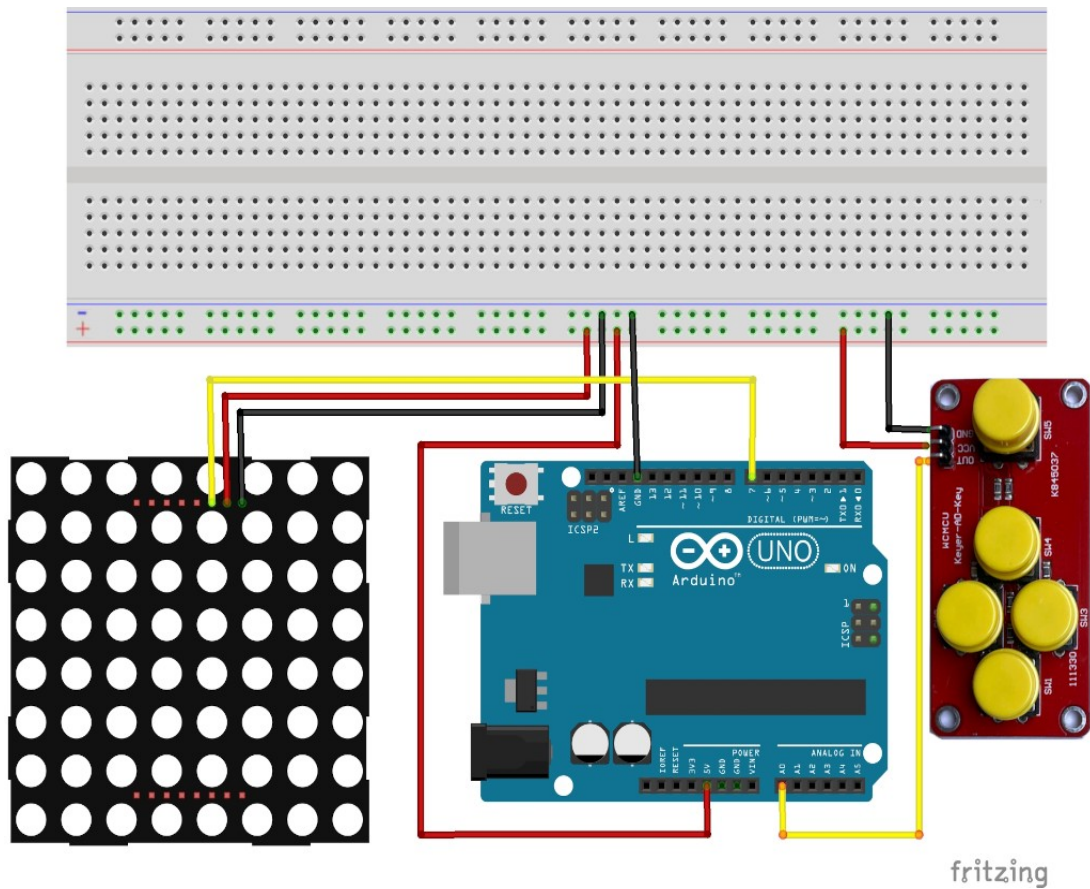
255, 255, 0

255, 228, 225

Benötigte Bauteile

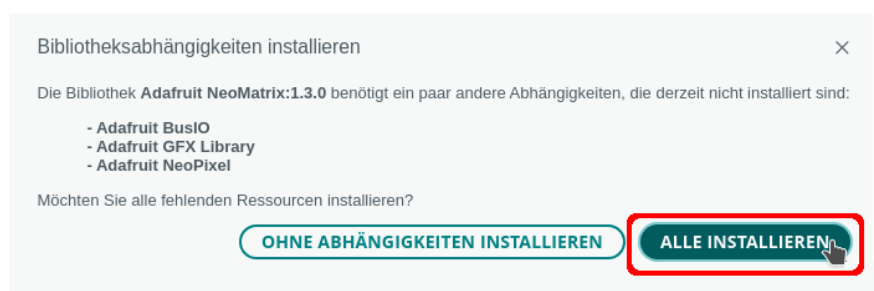
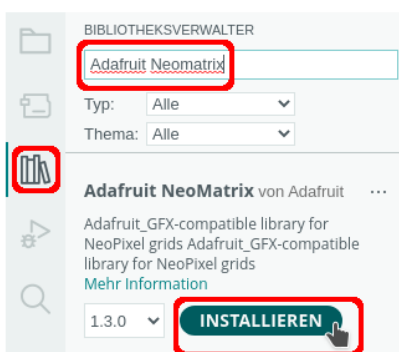
- ➔ RGB-Matrix 8x8
- ➔ Tastenpad
- ➔ Leitungsdrähte

Der Schaltplan



Benötigte Bibliotheken

Sketch → Bibliothek einbinden → Bibliotheken verwalten





Ob die Ausrichtung der RGB-Matrix korrekt ist, kannst du mit diesem Programm feststellen:

```
# include <Adafruit_NeoMatrix.h>

# define RGBMatrixPin 13

// 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);

void setup()
{
  RGBMatrix.setBrightness(10);

  // RGBMatrix starten
  RGBMatrix.begin();
}

void loop()
{
  RGBMatrix.clear();

  // horizontale Linie
  RGBMatrix.drawFastHLine(0, 2, 8, RGBMatrix.Color(0, 0, 255));

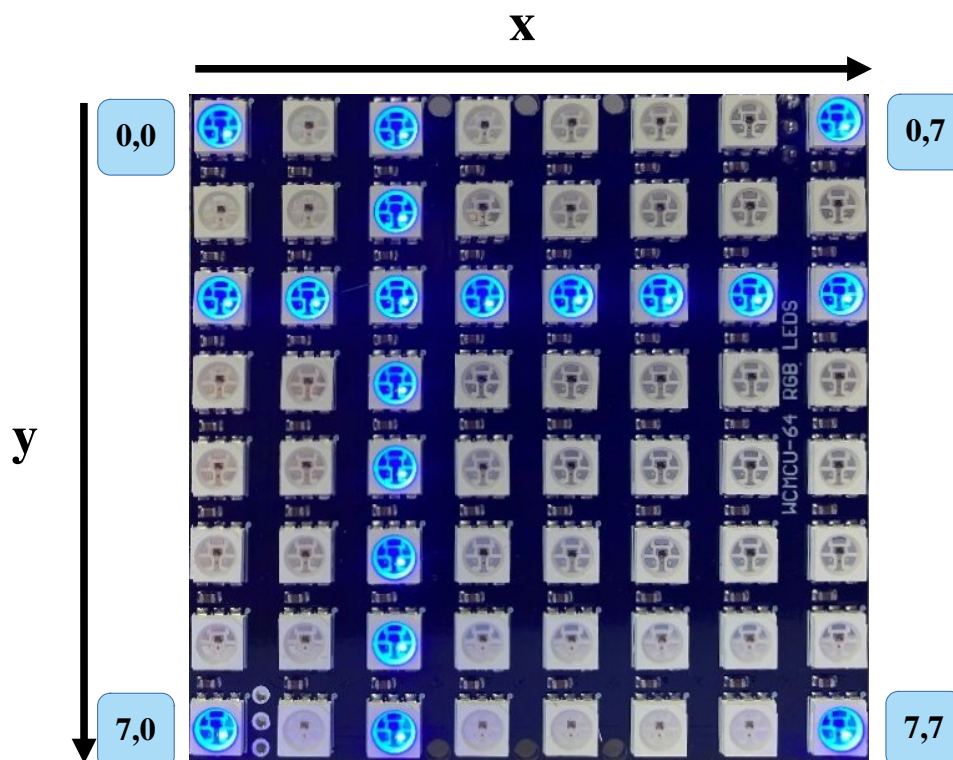
  // vertikale Linie
  RGBMatrix.drawFastVLine(2, 0, 8, RGBMatrix.Color(0, 0, 255));

  /*
    leuchtende LEDs in den Ecken
    oben links
    oben rechts
    unten links
    unten rechts
  */
  RGBMatrix.drawPixel(0, 0, RGBMatrix.Color(0, 0, 255));
  RGBMatrix.drawPixel(7, 0, RGBMatrix.Color(0, 0, 255));
  RGBMatrix.drawPixel(0, 7, RGBMatrix.Color(0, 0, 255));
  RGBMatrix.drawPixel(7, 7, RGBMatrix.Color(0, 0, 255));

  RGBMatrix.show();
}
```



So muss es aussehen:



Übersicht über die Funktionen der Bibliothek Adafruit_NeoMatrix (Auswahl)

Schlüsselwort	Parameter	Aktion
begin();		RGB-Matrix starten
setRotation(Richtung);	Richtung = 0 → nicht drehen Richtung = 1 → 90° drehen Richtung = 2 → 180° drehen	Bildschirm ausrichten
setBrightness(Parameter)	0 = aus, 255 = größte Helligkeit	Bildschirmhelligkeit setzen
fillScreen(Farbe)	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)	Bildschirmhintergrund
drawLine(StartX, StartY, EndeX, EndeY, Farbe);		Linie zeichnen
drawFastHLine(StartX, StartY, Länge, Farbe);		Horizontale Linie zeichnen
drawFastVLine(StartX, StartY, Länge, Farbe);		Vertikale Linie zeichnen
drawRoundRect(StartX, StartY, Breite, Höhe, Eckenradius, Farbe);		
fill.Rect(StartX, StartY, Breite, Höhe, Farbe);		



Schlüsselwort	Parameter	Aktion
drawCircle(MittelpunktX, MittelpunktY, Radius, Farbe);		Kreis zeichnen
fillCircle(MittelpunktX, MittelpunktY, Radius, Füllfarbe);		Ausgefüllten Kreis zeichnen
drawChar(0, 1, Zeichen, Rot, Hintergrund, Textgröße);	drawChar(0, 1, 'Z', Rot, Weiss, 1);	Einzelnes Zeichen schreiben
setCursor(x, y);		Cursor setzen
setTextSize(Textgröße);	Textgröße: 1 – 4 bei einer Matrix nur 1 möglich	Textgröße setzen
setTextColor(Farbe);		Textfarbe setzen
print("Text"); println("Text");		Text schreiben

Beispiele

Lauflicht

```
#include "Adafruit_NeoMatrix.h"

// Startposition links oben
int Spalte = 0;

#define RGBMatrixPin 7

// 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);

void setup()
{
  // Helligkeit setzen
  RGBMatrix.setBrightness(10);

  // RGBMatrix starten
  RGBMatrix.begin();
}
```



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

  // von oben nach unten
  for (int i = 0; i <= 7; i++)
  {
    RGBMatrix.drawPixel(Spalte, i, RGBMatrix.Color(0, 0, 255));
    delay(100);
    RGBMatrix.show();
  }

  RGBMatrix.clear();

  // eine Spalte nach rechts
  Spalte++;

  // von unten nach oben
  for (int i = 7; i >= 0; i--)
  {
    RGBMatrix.drawPixel(Spalte, i, RGBMatrix.Color(0, 0, 255));
    delay(100);
    RGBMatrix.show();
  }

  // solange das Ende (Spalte = 7) nicht erreicht ist
  // -> eine Spalte hinzufügen
  if (Spalte < 6) Spalte++;

  // Ende erreicht, Spalte wieder auf 0 setzen
  else Spalte = 0;
}
```

Grafikfunktionen

```
#include "Adafruit_NeoMatrix.h"

#define RGBMatrixPin 7

// 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)

void setup()
{
    // Helligkeit setzen
    RGBMatrix.setBrightness(10);

    // RGBMatrix starten
    RGBMatrix.begin();
}

void loop()
{
    RGBMatrix.clear();
    char Text[7] = {'A', 'r', 'd', 'u', 'i', 'n', 'o'};

    for (int i = 0; i < sizeof(Text); i++)
    {
        RGBMatrix.drawChar(0, 1, Text[i], Rot, 1, 1);
        RGBMatrix.show();
        delay(500);
        RGBMatrix.clear();
    }
    delay(1000);
    RGBMatrix.clear();

    // Bildschirm mit Farbe füllen
    RGBMatrix.fillScreen(Blau);
    RGBMatrix.show();
    delay(500);
    RGBMatrix.clear();

    RGBMatrix.fillScreen(Gelb);
    RGBMatrix.show();
    delay(500);
    RGBMatrix.clear();

    RGBMatrix.fillScreen(Rot);
    RGBMatrix.show();
    delay(500);

    RGBMatrix.clear();
    RGBMatrix.fillScreen(Gruen);
```




```
RGBMatrix.show();
delay(500);
RGBMatrix.clear();

RGBMatrix.fillScreen(Magenta);
RGBMatrix.show();
delay(500);
RGBMatrix.clear();

// einzelnes Zeichen schreiben
RGBMatrix.setCursor(0, 1);
RGBMatrix.setTextColor(Pink);
RGBMatrix.setTextSize(1);
RGBMatrix.print('Z');
RGBMatrix.show();
delay(500);
RGBMatrix.clear();

// Linie zeichnen
for (int i = 0; i <= 7; i++)
{
    RGBMatrix.drawLine(i, 0, i, 8, Gelb);
    RGBMatrix.show();
    delay(200);
}
delay(500);
RGBMatrix.clear();

// horizontale Linie zeichnen
for (int i = 1; i < 9; i++)
{
    RGBMatrix.drawFastHLine(0, i, i, Gruen);
    RGBMatrix.show();
    delay(200);
}
delay(500);
RGBMatrix.clear();

// vertikale Linie zeichnen
for (int i = 1; i < 9; i++)
{
    RGBMatrix.drawFastVLine(i, 0, i, Magenta);
    RGBMatrix.show();
    delay(100);
}
delay(500);
RGBMatrix.clear();

// Kreis zeichnen
RGBMatrix.drawCircle(4, 4, 3, Weiss);
RGBMatrix.show();
delay(500);
RGBMatrix.clear();

// ausgefüllten Kreis zeichnen
RGBMatrix.fillCircle(4, 4, 3, Blau);
```



```
RGBMatrix.show();
delay(500);

// Rechtecke zeichnen
RGBMatrix.clear();
for (int i = 1; i < 9; i++)
{
    RGBMatrix.drawRect(0, 0, i, i, RGBMatrix.Color(0, 255, 255));
    RGBMatrix.show();
    delay(200);
}
delay(500);
RGBMatrix.clear();
}
```

Spiel

Binde die benötigten Bibliotheken ein und definiere die Variablen. Beachte die Kommentare.

```
#include "Adafruit_NeoMatrix.h"

#define RGBMatrixPin 7

// Anzahl der LEDs für das Pixelmuster → kann angepasst werden
#define AnzahlLED 40

// RGBMatrix -> Name der Matrix
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 Schwarz    RGBMatrix.Color(0,0,0)
#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 MarineBlau RGBMatrix.Color(0,0,128)
#define DunkelGruen RGBMatrix.Color(0,100,0)
#define Golden     RGBMatrix.Color(205,149,12)
#define ZufallsFarbe RGBMatrix.Color(random(1, 255), random(1, 255), random(1, 255))

// PixelFarbe -> Farbe der zufällig leuchtenden Pixel
// PixelCursor -> Farbe des sich bewegenden Pixels
#define PixelFarbe Blau
#define PixelCursor Gelb

// Variablen der Zeit
float StartZeit;
float VerstricheneZeit;
```



```
float Sekunden;  
  
// entscheidet über den Neustart  
bool Start = true;  
  
int AnzahlKlicks = 1;  
  
// Startposition der LED  
int Zeile = 4;  
int Spalte = 4;  
int Taster;  
int Analogwert;
```

Der setup-Teil:

```
void setup()  
{  
    // setBrightness(0..255)  
    RGBMatrix.setBrightness(10);  
  
    // NeoPixel Bibliothek initialisieren  
    RGBMatrix.begin();  
  
    // Zufallsgenerator starten  
    randomSeed(analogRead(0));  
    Serial.begin(9600);  
}
```

Die Tasten des Tastenpads werden mit der Funktion Tasterabfrage gelesen.

```
int Tasterabfrage()  
{  
    Analogwert = analogRead(A0);  
  
    // kurzes delay() -> doppelten Tastendruck so weit wie möglich verhindern  
    delay(200);  
    /* für die Abfrage von Wertebereichen muss  
       # include <stdio.h> eingebunden werden  
       A0 gibt je nach Taster einen Wert aus  
       über den Seriellen Monitor kann dieser Wert angezeigt  
       und kann dann eventuell angepasst werden  
    */  
    // Serial.println(Analogwert);  
  
    switch (Analogwert)  
    {  
        case 0 ... 20:  
            Taster = 1;  
            break;
```



```
case 30 ... 60:
    Taster = 2;
    break;
case 70 ... 120:
    Taster = 3;
    break;
case 150 ... 200:
    Taster = 4;
    break;
case 300 ... 400:
    Taster = 5;
    break;
default:
    return 0;
}

// gedrückten Taster zurückgeben
return Taster;
}
```

Der „Parcour“ wird mit der Funktion ParcourBauen() erstellt.

```
void ParcourBauen()
{
    int Minimum = 0;
    int Maximum = 8;
    RGBMatrix.clear();

    for (int i = 0; i < AnzahlLED; i++)
    {
        // Zufallsposition der Pixel
        Spalte = random(Minimum, Maximum);
        Zeile = random(Minimum, Maximum);

        RGBMatrix.drawPixel(Zeile, Spalte, PixelFarbe);
    }
    Zeile = 4;
    Spalte = 4;
    RGBMatrix.drawPixel(Zeile, Spalte, PixelCursor);
    RGBMatrix.show();
}
```



Der loop-Teil:

```
void loop()
{
  {
    // wenn Start = true -> Parcours erstellen
    if (Start)
    {
      RGBMatrix.clear();
      ParcoursBauen();
      Start = false;
      StartZeit = millis();
      int Zeile = 4;
      int Spalte = 4;
    }

    Taster = Tasterabfrage();

    // Ergebnisse anzeigen und Neustart
    if (Taster == 5)
    {
      Start = true;

      // Zeit berechnen
      float Sekunden;
      VerstricheneZeit = millis() - StartZeit;
      Sekunden = VerstricheneZeit / 1000;
      String GesamtSekunden = String(Sekunden);

      // . durch , ersetzen
      GesamtSekunden.replace(".", ",");

      // Ausgabe im Seriellen Monitor
      Serial.println("Sekunden insgesamt: " + GesamtSekunden + " Sekunden");

      // Minuten berechnen
      int Minute = int(Sekunden) / 60;
```



```
// nur Ausgabe der Minuten wenn Minute > 0
if (Minute > 0)
{
    // Ausgabe verschönern, wenn Minute > 1 -> Ausgabe "Minuten"
    // "Minute"
    if (Minute > 1)
    {
        Serial.print(String(Minute) + " Minuten ");
    }
    else
    {
        Serial.print(String(Minute) + " Minute ");
    }
}

// von Sekunden Anzahl der Minuten abziehen
Sekunden = Sekunden - Minute * 60;

// Sekunden in String umwandeln
// damit . durch , ersetzt werden kann
String AnzahlSekunden = String(Sekunden);

// . durch , ersetzen
AnzahlSekunden.replace(".", ",");
Serial.println(AnzahlSekunden + " Sekunden");
Serial.println(String(AnzahlKlicks) + " Klicks!");
AnzahlKlicks = 1;
}

/*
    bei der Matrix gibt es kein oben oder unten
    links oder rechts
    die Funktion der Tasten muss durch Drehen der Matrix
    entsprechend angepasst werden
*/
// links
if (Taster == 1)
{
    RGBMatrix.drawPixel(Zeile, Spalte, Schwarz);
    if (Zeile < 7) Zeile++;
    RGBMatrix.drawPixel(Zeile, Spalte, PixelCursor);
    RGBMatrix.show();
    AnzahlKlicks ++;
}
```



```
// oben
if (Taster == 2)
{
    RGBMatrix.drawPixel(Zeile, Spalte, Schwarz);
    if (Spalte < 7) Spalte++;
    RGBMatrix.drawPixel(Zeile, Spalte, PixelCursor);
    RGBMatrix.show();
    AnzahlKlicks ++;
}

// unten
if (Taster == 3)
{
    RGBMatrix.drawPixel(Zeile, Spalte, Schwarz);
    if (Spalte > 0) Spalte--;
    RGBMatrix.drawPixel(Zeile, Spalte, PixelCursor);
    RGBMatrix.show();
    AnzahlKlicks ++;
}

// rechts
if (Taster == 4)
{
    RGBMatrix.drawPixel(Zeile, Spalte, Schwarz);
    if (Zeile > 0) Zeile--;
    RGBMatrix.drawPixel(Zeile, Spalte, PixelCursor);
    RGBMatrix.show();
    AnzahlKlicks ++;
}
}
}
```