LED_Matrix mithilfe des Freenove FNK0054 "Projects Kit for Raspberry Pi"

1. Einführung

Quelle

Quelle

Projektziel

Das Ziel dieses Projekts ist es, eine LED-Matrix mithilfe eines Raspberry Pi und des Freenove FNK0054 "Projects Kit for Raspberry Pi" zu steuern. Dabei wird zwei 74HC595-Schieberegister verwendet, um die Anzahl der benötigten GPIO-Pins zu minimieren und verschiedene Muster, Zeichen und Symbole auf der LED-Matrix anzuzeigen.

Hintergrund

Das Freenove FNK0054 "Projects Kit for Raspberry Pi" ist ein umfangreiches Kit, das verschiedene Komponenten und Module enthält, darunter LEDs, Widerstände, Schieberegister und andere Bauelemente. Dieses Kit ermöglicht es, eine Vielzahl von Projekten durchzuführen, um die Fähigkeiten des Raspberry Pi in der Elektroniksteuerung zu demonstrieren. Eine LED-Matrix, gesteuert durch ein 74HC595-Schieberegister, zeigt, wie digitale Elektronik und Python-Programmierung kombiniert werden können, um visuelle Ausgaben zu erzeugen.

Relevanz

Das Projekt zeigt praxisnah, wie man eine LED-Matrix mit einem Raspberry Pi und Standardkomponenten aus einem Elektronik-Kit steuern kann. Es verdeutlicht, wie einfache Hardware-Schnittstellen programmiert werden können und ist ein exzellentes Beispiel für den Einsatz von Schieberegistern in der Elektronik.

2. Grundlagen und Theorie

LED-Matrix:

Eine LED-Matrix besteht aus einer Gruppe von LEDs, die in einem rechteckigen Gitter angeordnet sind. Jede LED kann individuell angesteuert werden, um komplexe Muster und Zeichen darzustellen.

74HC595 Schieberegister:

Der 74HC595 Schieberegister ist ein 8-Bit-Schieberegister mit seriellen Eingängen und parallelen Ausgängen. Es ermöglicht die Steuerung mehrerer LEDs mit einer geringen Anzahl von Steuerleitungen. Durch das Schieberegister wird die Anzahl der benötigten GPIO-Pins am Raspberry Pi reduziert, da die Daten seriell in das Register geschrieben werden und dann parallel an die LEDs ausgegeben werden.

- Schieberegister: Eine Übersicht und Definition

Freenove "Projects Kit for Raspberry Pi"

Das Freenove FNK0054 "Projects Kit for Raspberry Pi" ist ein umfassendes Elektronik-Kit, das speziell entwickelt wurde, um Anwendern zu helfen, Projekte mit einem Raspberry Pi durchzuführen. Es ist besonders gut geeignet für Einsteiger, die die Grundlagen der Elektronik und Programmierung erlernen möchten. Das Kit enthält eine Vielzahl von Komponenten und Bauteilen, mit denen eine breite Palette von Projekten realisiert werden kann.

- Offizielle Webseite
- PDF-Tutorial
- Video-Tutorial

3. Materialien und Werkzeuge

Software

- Raspbian OS
- Python 3
- Jupyter Notebook
- Python 3 Bibliotheken wie **gpiozero**

Hardware

- Raspberry Pi
- Freenove Projects Board für Raspberry Pi
- LED-Matrix
- GPIO Ribbon Kabel

- Freenove-Tutorial

Sensor/(Aktor)en, inkl. Datenblätter

der 74HC595 Schieberegister wird als Aktor benutzt und die Daten aus Datenblatt lautet:

Betriebsspannung (Vcc): 2V bis 6V, typischerweise 5V.

Maximaler Ausgangsstrom pro Pin: 35 mA.

Maximaler Gesamtstrom (alle Ausgänge kombiniert): 70 mA.

Berechnung des Vorwiderstands

der Vorwiderstand der LED kann wie folgt berechnet werden:

```
# Beispielcode zur Berechnung des Vorwiderstands:
V_supply = 5.0  # Versorgungsspannung in Volt
V_f = 2.0  # Vorwärtsspannung der LED in Volt
I_f = 0.02  # Vorwärtsstrom der LED in Ampere

R = (V_supply - V_f) / I_f
print("Erforderlicher Vorwiderstand: {} Ohm".format(R))
```

```
Erforderlicher Vorwiderstand: 150.0 Ohm
```

Pin Beschaltung

Die Pin-Beschaltung in Freenove Projects Kit für Raspberry Pi ist wie flogt:

74HC595 Schieberegister	GPIO Pin des Raspberry Pi
DS(PIN 14)	GPIO 22
SR_CP(PIN 12)	GPIO 27
SH_CP(PIN 11)	GPIO 17

Freenove-Tutorial

4. Schaltungsdesign

Schmatischer Entwurf

- Schaltung-Aufbau
- Freenove-Tutorial

Hardware-Verbindung

Freenove-Tutorial

5. Implementierung

Hardware-Aufbau

Der Hardware soll wie obbiges Bild aufgabeut werden.

- Raspberry einschalten und durch das Ribbon Kabel mit Freenove Projects Kit für Raspberry Pi verbunden.
- Power-Taste von Freenove Projects Kit für Raspberry Pi einschalten.

- Um LED-Matrix 8*8 nutzen zu können, sollen wir den Schalter 7 des Boards einschalten.

Software-Setup

1. Raspbian OS installieren.
2. Python und Jupyter Notebook installieren: `sh sudo apt-get update sudo apt-get install python3 jupyter`
3. Installation der benötigten Bibliotheken: ```sh sudo apt-get install python-gpiozero

```

## 6. Experimente und Ergebnisse

## Beispiel 1

### Installieren Sie gpiozero auf Ihrem Rechner

```
pip install RPi.GPIO
```

```
pip install gpiozero
```

```
pip install luma.led_matirx
```

```
pip install pillow
```

in diesem Beispiel wird Herz-Symbol gezeigt

```
# importieren von Bibliotheken
from gpiozero import OutputDevice

# importieren von time
import time

# LSB und MSB
LSBFIRST = 1
MSBFIRST = 2

# Pin-Definition
dataPin = OutputDevice(22)
latchPin = OutputDevice(27)
clockPin = OutputDevice(17)

# Testmuster für Herz
pic = [0x0C, 0x1E, 0x3F, 0x7F, 0x7F, 0x3E, 0x1C, 0x08]

# Funktion, um 8 Bits in der angegebenen Reihenfolge an das
Schieberegister zu senden
def shiftOut(order, val):
    for i in range(0, 8):
        clockPin.off()
```
```

```
        if order == LSBFIRST:
            dataPin.on() if (0x01 & (val >> i) == 0x01) else
dataPin.off()
        elif order == MSBFIRST:
            dataPin.on() if (0x80 & (val << i) == 0x80) else
dataPin.off()
        clockPin.on()

# Funktion, um das Herzmuster kontinuierlich auf der LED-Matrix
anzuzeigen
def loop():
    while True:
        for j in range(0, 500):  # Wiederholt das Muster zur
Stabilisierung der Anzeige
            x = 0x80
            for i in range(0, 8):  # Durchläuft jede Spalte des
Musters
                latchPin.off()
                shiftOut(MSBFIRST, pic[i])  # Überträgt die
Zeilendaten
                shiftOut(MSBFIRST, ~x)  # Überträgt die Spaltenauswahl
                latchPin.on()
                time.sleep(0.001)  # Verzögerung für eine flüssigere
Anzeige
                x >>= 1  # Verschiebt die Aktivierung zur nächsten
Spalte

# Funktion zum Löschen der LED-Matrix und Freigeben der Ressourcen
def destroy():
    latchPin.off()
    shiftOut(MSBFIRST, 0x00)
    shiftOut(MSBFIRST, 0x00)
    latchPin.on()
    dataPin.close()
    latchPin.close()
    clockPin.close()

# Hauptprogramm-Ausführung
if __name__ == '__main__':
    print('Programm startet...')
    try:
        loop()  # Startet die Anzeigeschleife
    except KeyboardInterrupt:
        print("Programm wird beendet")
    finally:
        destroy()  # Stellt sicher, dass die Ressourcen beim Beenden
freigegeben werden
```

Quelle: Freenove-Tutorial

Darstellenung in Matplotlib

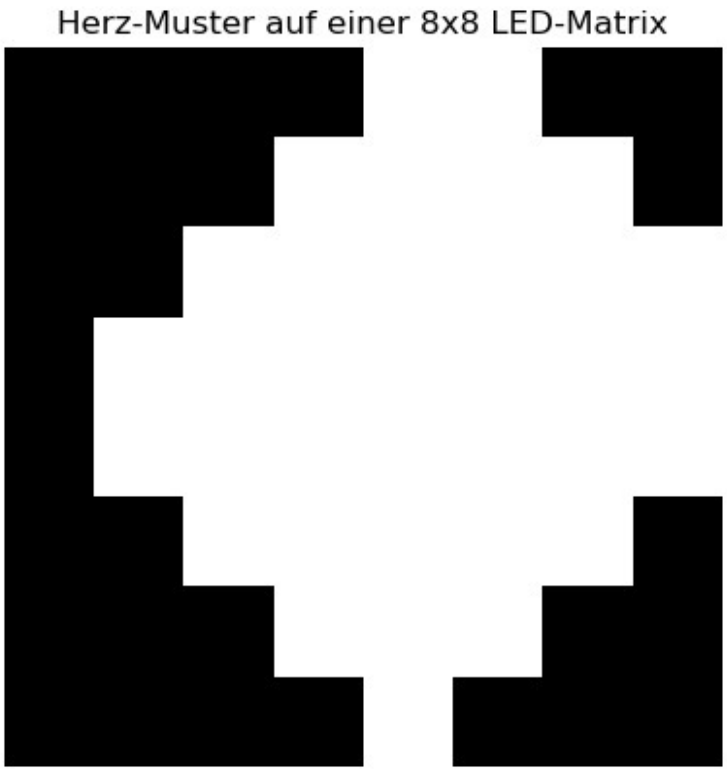
```
import matplotlib.pyplot as plt
import numpy as np

# Muster zur Anzeige eines Herzsymbols (wie in deinem ursprünglichen
Code)
pic = [0x0C, 0x1E, 0x3F, 0x7F, 0x7F, 0x3E, 0x1C, 0x08]

# Funktion, um das Herzmuster zu extrahieren und als 8x8-Array
darzustellen
def convert_to_matrix(pic):
    # Wir erzeugen ein 8x8 NumPy Array, um das Muster zu speichern
    matrix = np.zeros((8, 8), dtype=int)
    for i in range(8):
        # Wandelt das Hex-Wert-Muster in binäre Werte um und speichert
sie in der Matrix
        for j in range(8):
            if (pic[i] & (1 << (7 - j))) != 0:
                matrix[i, j] = 1
    return matrix

# Herzmuster in eine Matrix umwandeln
matrix = convert_to_matrix(pic)

# Visualisierung mit Matplotlib
plt.imshow(matrix, cmap='gray', interpolation='nearest')
plt.axis('off')  # Keine Achsen anzeigen
plt.title("Herz-Muster auf einer 8x8 LED-Matrix")
plt.show()
```



Was sind LSB und MSB ?

MSB (Most Significant Bit): das Bit mit Index Null hat den geringsten Wert.

LSB (Least Significant Bit): das Bit mit Index Null hat den höchsten Wert.

Aufgabe1:

Wie lautet das Schachbrettmuster ?

Aufgabe2

Versuchen Sie das obige Programm so anzupassen, dass Schachbrettmuster auf die LED-Matrix angezeigt wird.

Beispiel 2

Installieren Sie gpiozero auf Ihrem Rechner

```
sudo apt-get install python-gpiozero
```

in diesem Beispiel wird Hallo auf den LED-Matrix gezeigt

```
# importieren von Bibliotheken
from gpiozero import OutputDevice
```

```
# importieren von time
import time

# LSB und MSB
LSBFIRST = 1
MSBFIRST = 2

# Pin-Definition
dataPin = OutputDevice(22)
latchPin = OutputDevice(27)
clockPin = OutputDevice(17)

# Testmuster
H = [0x7F, 0x49, 0x49, 0x49, 0x49, 0x49, 0x49, 0x00]
A = [0x3E, 0x51, 0x49, 0x49, 0x49, 0x51, 0x3E, 0x00]
L = [0x7F, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00]
O = [0x3E, 0x41, 0x41, 0x41, 0x41, 0x41, 0x3E, 0x00]

data = H + [0x00] * 8 + A + [0x00] * 8 + L + [0x00] * 8 + L + [0x00] *
8 + O  # Kombiniert die Muster der Buchstaben mit Leerraum

# Funktion, um 8 Bits in der angegebenen Reihenfolge an das
Schieberegister zu senden
def shiftOut(order, val):
    for i in range(8):
        clockPin.off()

        if order == LSBFIRST:
            dataPin.on() if (0x01 & (val >> i)) else dataPin.off()
        elif order == MSBFIRST:
            dataPin.on() if (0x80 & (val << i)) else dataPin.off()
        clockPin.on()

# Funktion, um die Buchstaben von "HALLO" nacheinander auf der Matrix
anzuzeigen
def loop():
    while True:
        for k in range(0, len(data) - 8):
            for j in range(0, 20):
                x = 0x80
                for i in range(k, k + 8):
                    latchPin.off()
                    shiftOut(MSBFIRST, data[i])
                    shiftOut(MSBFIRST, ~x)
                    latchPin.on()
                    time.sleep(0.001)
                    x >>= 1

def destroy():
    latchPin.off()  # Setzt den Latch-Pin auf niedrig, um das
Schieberegister zu deaktivieren
```

```
    shiftOut(MSBFIRST, 0x00)  # Überträgt 0x00, um alle Zeilen zu
deaktivieren
    shiftOut(MSBFIRST, 0x00)  # Überträgt 0x00, um alle Spalten zu
deaktivieren
    latchPin.on()  # Setzt den Latch-Pin auf hoch, um die Ausgänge zu
aktualisieren
    dataPin.close()  # Schließt den Daten-Pin, um die Ressourcen
freizugeben
    latchPin.close()  # Schließt den Latch-Pin
    clockPin.close()  # Schließt den Takt-Pin

# Hauptprogramm-Ausführung
if __name__ == '__main__':
    print('Displaying "HALLO"...')
    try:
        loop()
    except KeyboardInterrupt:
        print("Ending program")
    finally:
        destroy()
```

```
Displaying "HALLO"...
Ending program
```

Quelle: Freenove-Tutorial

Darstallung in Matplotlib

Aufgabe4:

Zeigen Sie Wellcome auf den LED-Matrix.

Der Testmuster dafür lautet:

[

```
0x1E, 0x36, 0x36, 0x36, 0x36, 0x36, 0x00, 0x00,  # W

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # E

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # L

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # C

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # O

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # M

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # E
```

]

7. Diskussion und Fazit

Diskussion

Das Projekt demonstriert die Steuerung einer LED-Matrix mit einem Raspberry Pi und einem Schieberegister, unter Verwendung des Freenove FNK0054 Kits. Die erfolgreiche Implementierung zeigt die Vielseitigkeit des Kits und des Raspberry Pi und die Möglichkeiten der digitalen Elektronik.

Fazit

Das Projekt war erfolgreich und hat das Verständnis für die Steuerung von LED-Matrizen mit dem Raspberry Pi verbessert. Die Implementierung ist einfach, aber effektiv und kann als Grundlage für weiterführende Projekte dienen.