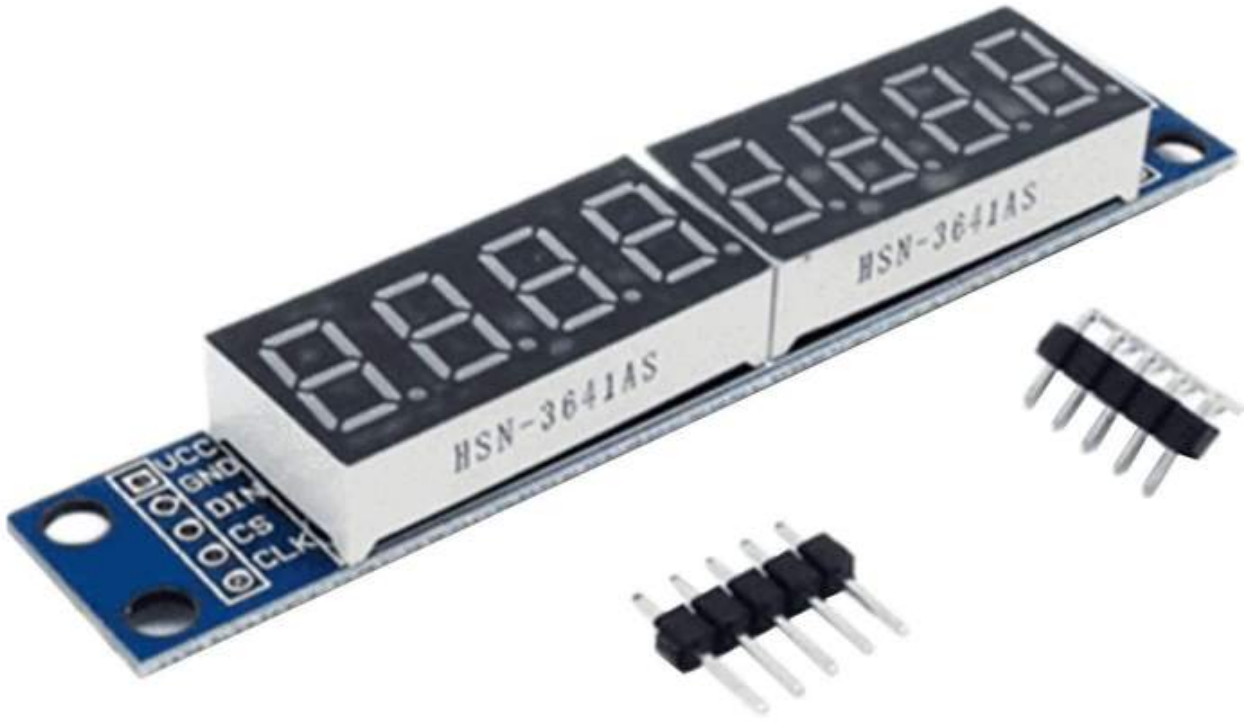
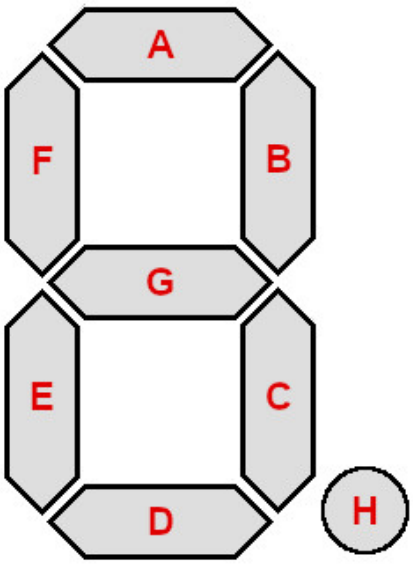
Willkommen!

Vielen Dank, dass sie sich für unser *8-Bit-7-Segment-Display-Modul* von AZ-Delivery entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

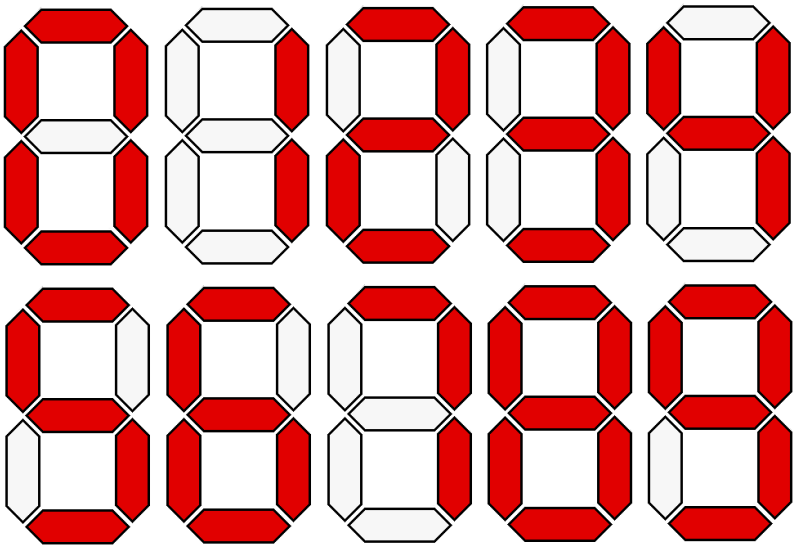
Viel Spaß!



Ein Sieben-Segment-Display ist ein Bildschirm mit sieben Segmenten, die der Zahl 8 nach, angeordnet sind. Genauer gesagt haben diese Bildschirme 8 Segmente, wobei das achte Segment ein Punkt in der Nähe der digitalen Zahl 8 ist. Jedes Segment hat seinen eigenen Namen, jeweils beschriftet mit den Buchstaben A bis G (bis H):



Durch das *Einschalten* bestimmter Segmente (LEDs) werden digitale Zahlen oder Zeichen erzeugt:



Zur Steuerung eines Segments oder LED sind zwei Pins erforderlich. Ein Pin für die Stromversorgung und der andere für die Masse. Alle Masse-Pins sind miteinander verbunden. Andere Pins (oder LED-Steuerpins) sind getrennt. Das bedeutet, dass sieben Segmentdisplays *8* Steuerpins und einen Massepin haben (insgesamt *9* Pins). Zur Steuerung dieser Displays werden *8* digitale Pins des Mikrocontrollers benötigt, was unpraktisch sein kann. Wenn Sie mehr als einen der sieben Displays steuern möchten, würden Sie zu viele Pins benötigen, über die die meisten Mikrocontroller nicht verfügen. Aus diesem Grund verwenden wir Treiberchips, eine integrierte Schaltung, die für die Ansteuerung dieser Art von Bildschirmen bestimmt ist. In unserem Fall verwenden wir einen Treiberchip mit der Bezeichnung "*MAX7219*". Der Treiberchip ist für die Ansteuerung von Bildschirmen mit vielen LEDs vorgesehen. Der Treiberchip ist in der Lage, einen oder bis zu 8 Displays anzusteuern. Ein Vorteil des Chips besteht darin, dass nur 4 digitale Pins für die Kommunikation mit ihm nötig sind und es kann 8 der Sieben-Segmentbildschirme ansteuern. Ein weiterer Vorteil besteht darin, dass Sie mehrere Treiberchips seriell anschließen und dieselben 4 Pins verwenden können, um mehr als 8 Displays anzusteuern.

Die SPI-Schnittstelle ermöglicht die Kommunikation zwischen Mikrocontroller und dem MAX7219-Treiberchip. SPI steht für "Serial Peripheral Interface" und wird zum Senden und Empfangen von Daten zwischen einem Mikrocontroller und anderen Peripheriegeräten (Sensoren, Speicher usw.) verwendet. In einigen Kreisen wird sie als 4-Draht-Protokoll bezeichnet, da nur 4 Drähte benötigt werden, um Hardware-Verbindungen herzustellen (plus Massedraht). Gewöhnlich wird die Kommunikation zwischen einem Master und einem oder mehreren Slave-Geräten hergestellt.

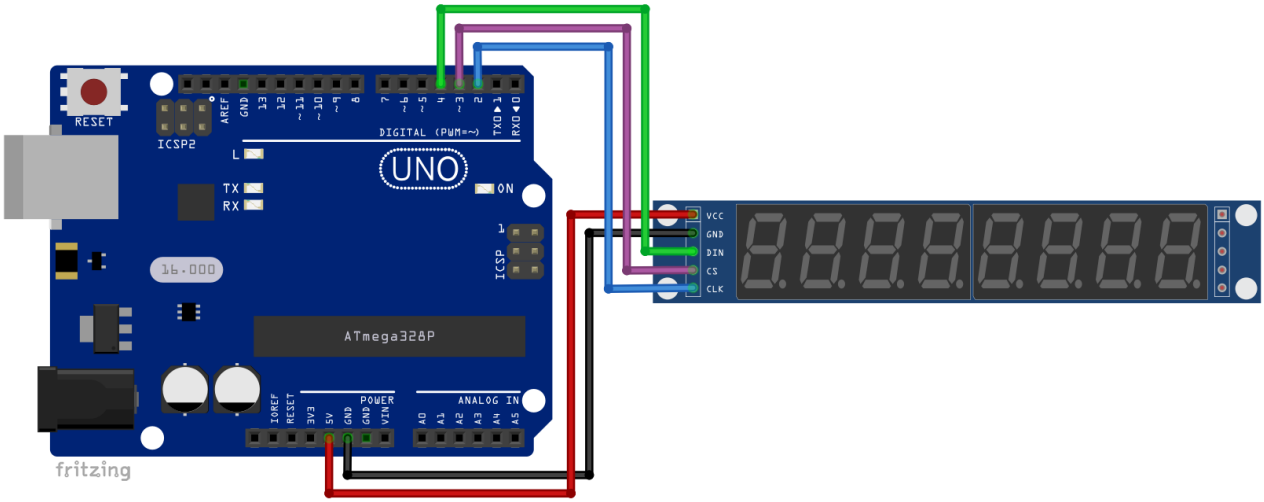
Technische Daten:

» Stromvers.- und Logikspannungsbereich:	3.3V bis 5V DC
» Treiberchip:	MAX7219
» Interface-Typ:	SPI-Schnittstelle
» Maximale Taktfrequenz:	10MHz
» Anzahl der Segmente:	8
» LED-Farbe der Ziffer:	Rot
» Helligkeitsregelung:	16 Stufen
» Kaskadenfähigkeit:	Ja, Daisy chain
» Dimensionen:	84 x 16mm [3.3 x 0.6in]

Es ist möglich, dieselben 4 Drähte zur Steuerung mehrerer 8-Bit-Siebensegmentmodule zu verwenden. Dies wird als "Daisy-Chaining" der Module über dieselbe SPI-Schnittstelle bezeichnet. Alles, was wir tun müssen, ist, das nächste Modul an die Ausgangspins des ersten Moduls anzuschließen. Aber Vorsicht, mehr Module benötigen mehr Strom. Verwenden Sie daher ein externes Netzteil und keinen Arduino-Board-Spannungsregler.

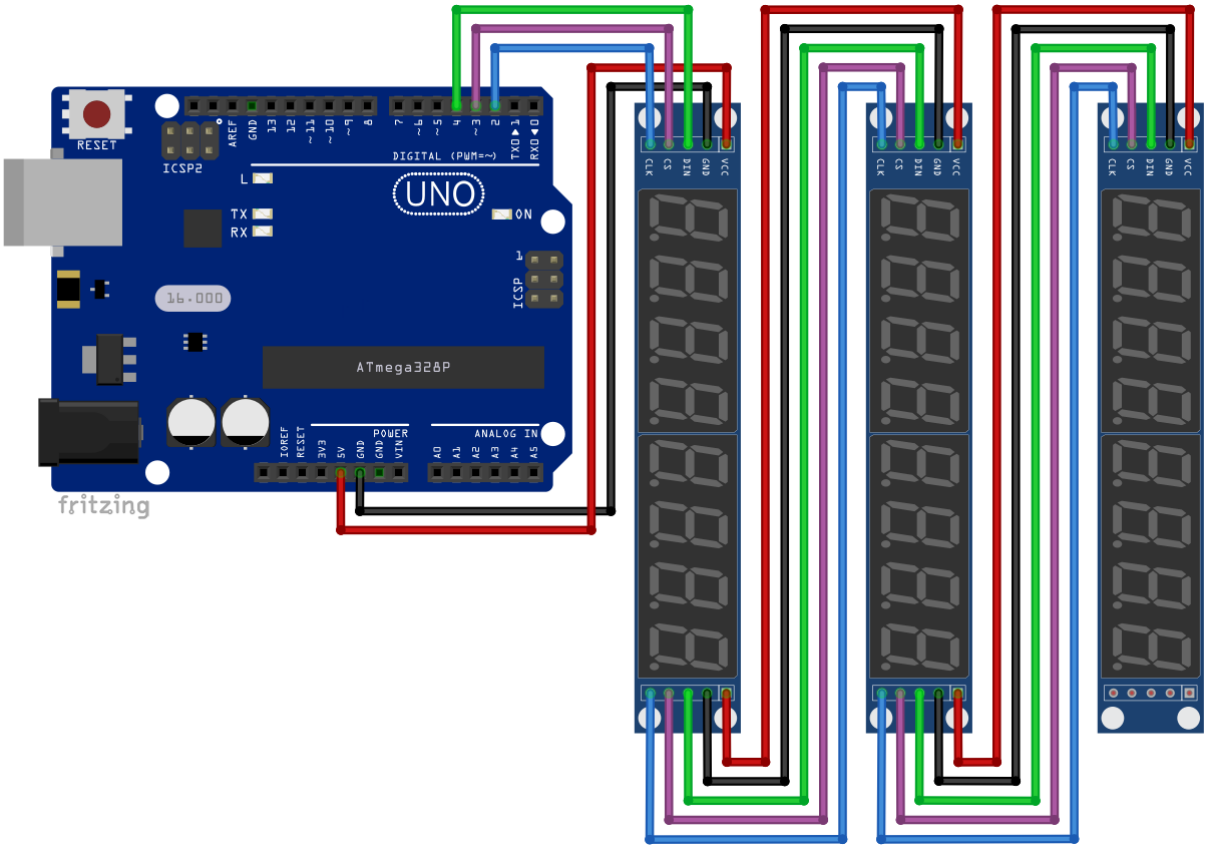
Verbindung des Moduls mit dem Uno

Verbinden Sie das Modul mit dem Uno wie unten abgebildet:



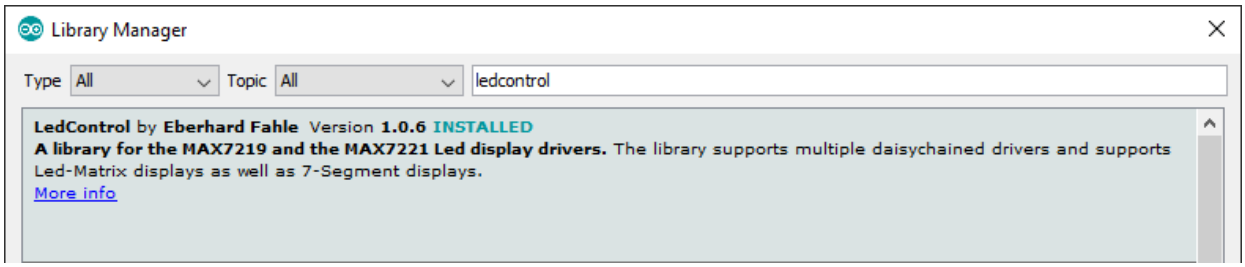
Module Pin	>	Uno Pin	
VCC	>	5V	**Roter Draht**
GND	>	GND	**Schwarzer Draht**
DIN	>	D4	**Grüner Draht**
CS	>	D3	**Violetter Draht**
CLK	>	D2	**Blauer Draht**

Ein Beispiel für ein „Daisy Chaining“-Verbindungsdiagramm:



Arduino IDE library

Um dieses Modul mit dem Arduino zu verwenden, müssen wir als Erstes die benötigte Library herunterladen. Gehen Sie zu *Tools* > *Manage Libraries*. In einem sich neu öffnenden Fenster geben Sie in dem Suchfeld "*ledcontrol*" ein und installieren Sie die "*LedControl*" Library von "*Eberhard Fahle*", wie unten abgebildet:



In dieser Library sind drei Sketch-Beispiele enthalten. Wir verwenden und modifizieren den Sketch "*LCDemo7Segmet*":
File > *Examples* > *LedControl* > *LCDemo7Segmet*.

Sketch-Code:

```
#include "LedControl.h"
// DIN = 4, CS = 2, CLK = 3, Number of MAX7219 chips = 1
LedControl lc = LedControl(4, 2, 3, 1);

void setup() {
  // The MAX7219 chip is in power-saving mode on startup
  lc.shutdown(0, false);  // wakeup call
  lc.setIntensity(0, 8);  // Set brightness level (0 is min, 15 is max)
  lc.clearDisplay(0);     // Clear display register
}
void loop() {
  for(int i = 0; i < 21; i++) {
    lc.setChar(0, i, 'a', false);       //  a
    lc.setDigit(0, i - 1, 2, false);    //  z
    lc.setRow(0, i - 2, 0x01);          //  -
    lc.setChar(0, i - 3, 'd', false);   //  d
    lc.setChar(0, i - 4, 'e', false);   //  e
    lc.setRow(0, i - 5, B00001110);     //  l
    lc.setDigit(0, i - 6, 1, false);    //  i
    lc.setRow(0, i - 7, B00111110);     //  v
    lc.setChar(0, i - 8, 'e', false);   //  e
    lc.setRow(0, i - 9, 0x05);          //  r
    lc.setRow(0, i - 10, B00111011);    //  y
    lc.setChar(0, i - 11, ' ', false); //  space (empty)
    delay(150);
  }
}
```

Zu Beginn des Sketches binden wir die *LedControl* Library ein und erstellen , sowie initialisieren das Bildschirmobjekt "*lc*". Das Bildschirm-objekt wird in der folgenden Zeile des Codes initialisiert:

```
LedControl lc = LedControl(4, 2, 3, 1)
```

Wir nennen den Konstruktor `LedControl()`, der vier Argumente akzeptiert. Die ersten drei Argumente stellen Verbindungen der Modul-Pins mit Uno-Pins dar:

Modul Pins	>	Uno Pins
DIN (MOSI)	>	D4
CS (SS)	>	D3
CLK	>	D2

Das vierte Argument ist die Anzahl der verwendeten MAX7219-Treiberchips. Wir haben einen in unserem Sketch verwendet. Wenn Sie jedoch mehrere 8-Bit-Siebensegment-Bildschirmmodule miteinander verbinden (Daisy-Chaining), müssen Sie die Anzahl der Module in diesem vierten Argument angeben.

In der *`setup()`*-Funktion müssen wir zuerst das Modul EINSCHALTEN. Der MAX7219-Treiberchip befindet sich im Stromsparmodus, wenn er nicht benutzt wird. Um ihn EINZUSCHALTEN, müssen wir einen "*wakeup call*" machen. Wir tun dies mit dieser Zeile Code: *`lc. shutdown(0, false)`*

Der erste Argumentwert "*0*" ist die Nummer des MAX7219-Chips. Wenn Sie mehrere Module in einer Daisy-Chain verbunden haben, müssen Sie alle MAX7219-Chips wecken. Der Wert des ersten Arguments ist also die Anzahl der MAX7219 in einer Reihe von Modulen, beginnend bei *0*. Das zweite Argument "*`false`*" wird verwendet, um das Modul aufzuwecken. Wenn Sie es auf "*`true`*" ändern, wird das Modul in den Stromsparmodus versetzt.

Das zweite, was wir in der *setup()*-Funktion tun müssen, ist die Helligkeitsstufe der sieben Segmentbildschirme einzustellen. Wir tun dies mit der folgenden Zeile des Codes: `ls.setIntensity(0, 8)`
Das erste Argument "*0*" ist die Nummer des Moduls in der Daisy Chain, und das zweite Argument "8" ist die Helligkeitsstufe. Die unterste Helligkeitsstufe ist 0 - ausgeschaltet, und die maximale Stufe ist 15.

Schließlich leeren wir den Bildschirm mit der folgenden Zeile des Codes:
`lc.clearDisplay(0)`
Das eine Argument "*0*" ist die Nummer des Moduls in der Daisy Chain.

In der *loop()*-Funktion scrollen wir die Nachricht "AZ-Delivery" auf dem Modul. Dazu verwenden wir mehrere Funktionen, die in der *LedControl* Library definiert sind. Um eine bestimmte Nummer oder ein bestimmtes Zeichen auf dem Modul auszugeben, verwenden wir die folgende Funktion:
`lc.setChar(chip, position, character, point)`
Das erste Argument ist die Nummer des Moduls in der Daisy Chain. Das zweite Argument ist die Zeichenposition auf dem Modul. Das Modul hat *8* Sieben-Segmentbildschirme, so dass die Positionswerte bei *0* bis *7* liegen.

Das dritte Argument ist eine Konstante vom Typ *char*, ein Zeichen oder eine Zahl, und seine Werte sind folgende Zeichen:

- Einstellige Zahlen: 0, 1, 2, 3, 4, 5, 6, 7, 8 und 9
- Buchstaben:
  - “A” oder “a” - zeigt Großbuchstaben
  - “B” oder “b” - zeigt Kleinbuchstaben
  - “C” oder “c” - zeigt Kleinbuchstaben

- » “D” oder “d” - zeigt Kleinbuchstaben
- » “E” oder “e” - zeigt Großbuchstaben
- » “F” oder “f” - zeigt Großbuchstaben
- » “H” oder “h” - zeigt Großbuchstaben
- » “L” oder “l” - zeigt Großbuchstaben
- » “P” oder “p” - zeigt Großbuchstaben
- » “-” - zeigt das Strichzeichen (Minuszeichen)
- » “.” - Dezimalpunkt einschalten, H-Segment
- » “_” - zeigt den Unterstrich an
- » <SPACE> - zeigt Leerzeichen oder Leerstellen

Das vierte Argument ist ein Punkt, oder das H-Segment des Siebensegmentbildschirms. Sie können es mit *point* = *false* AUSSCHALTEN oder mit *point* = *true* EINSCHALTEN.
Zum Beispiel: `lc.setChar(0, 0, 'a', false)`
das gibt den Buchstaben '*a*' auf dem ersten Siebensegmentbildschirm des Moduls aus, wobei das *H*-Segment AUSGESCHALTET ist.

Ähnlich ist: `lc.setDigit(chip, position, number, point)`
Alle Argumente sind die gleichen wie in der *`setChar()`*-Funktion, mit Ausnahme des dritten Arguments. Das dritte Argument ist eine konstante Ganzzahl, deren Werte im Bereich von 0 bis 9 liegen. Zum Beispiel:
`lc.setDigit(0, 0, 2, false)`
was die Nummer *2* auf dem zweiten Siebensegmentbildschirm des Moduls ausgibt, wobei das *H*-Segment AUSGESCHALTET ist.

Und die letzte in der Sketch verwendete Funktion ist *setRow()*. Diese Funktion akzeptiert drei Argumente. Das erste ist die Nummer des Moduls in der Daisy Chain. Das zweite Argument ist die Zeichenposition auf dem Modul, wie in den Funktionen *setChar()* oder *setDigit()*. Und das dritte Argument ist eine Hexadezimal- oder Binärzahl, deren Wert das Zeichen darstellt, das auf dem Modul angezeigt wird. Der binäre Wert wird wie folgt dargestellt: *BHABCDEFG*

Wobei das erste "B" eine Binärzahl darstellt und die Zeichen *"HABCDEFG"* die Segmentbuchstaben des spezifischen Siebensegmentbildschirms sind. Um ein bestimmtes Segment, z.B. ein B-Segment, EINZUSCHALTEN, ist der Wert dieser Binärzahl *B00100000*, oder wenn wir Nummer 6 ausgeben wollen, ist der Wert *B01011111*.

Wir können diesen binären Wert in hexadezimale Zahlen (hier: 6) umwandeln: `B 0101 1111`

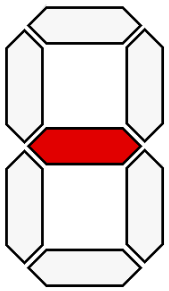
8-Bit-Binärzahl in zwei 4-Bit-Zahlen aufteilen und in einen hexadezimalen Wert umwandeln, für einen zweistelligen hexadezimalen Wert:

0x5f

Im Sketch können wir mit dieser Funktion ein Strichzeichen "-" anzeigen:

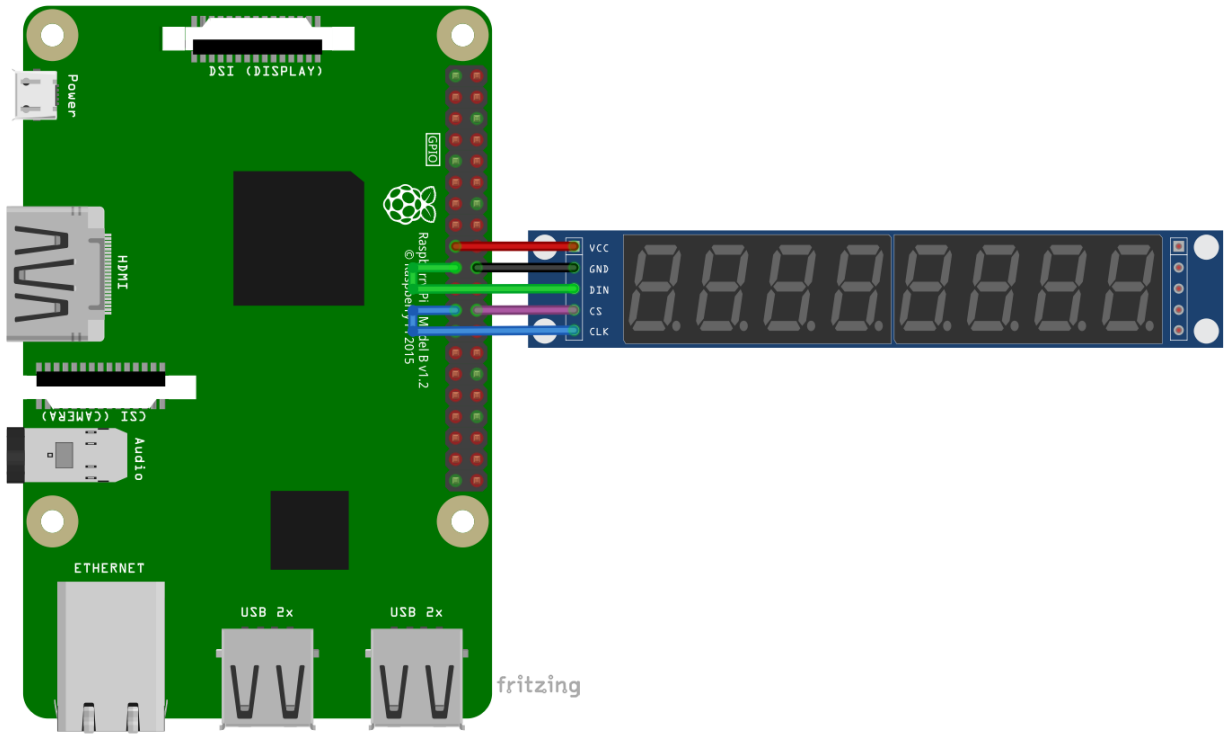
```
lc.setRow(0, number, 0x01)
```

Wo der hexadezimale Wert *0x01* gleich *B00000001* ist oder das G-Segment des Siebensegmentbildschirms auf EIN gestellt wird.



Verbindung des Moduls mit dem Raspberry Pi

Verbinden Sie das Modul mit dem Raspberry Pi wie folgt:



Module pin	>	Raspberry Pi pin		
VCC	>	3V3	[pin 17]	**Roter Draht**
DIN	>	GPIO10	[pin 19]	**Grüner Draht**
GND	>	GND	[pin 20]	**Schwarzer Draht**
CLK	>	GPIO11	[pin 23]	**Blauer Draht**
CS	>	GPIO08	[pin 24]	**Violetter Draht**

Python Library

Um das Modul mit dem Raspberry Pi zu verwenden, empfehlen wir ihnen, eine Library herunterzuladen. Wir werden die Library namens "*luma.led_matrix*" verwenden. Es müssen mehrere Schritte befolgt werden, um diese Library herunterzuladen und zu installieren. Der erste Schritt besteht darin, Raspbian zu aktualisieren und dann die Library herunterzuladen und zu installieren. Öffnen Sie das Terminal und führen Sie die folgenden Befehle aus, um Raspbian zu aktualisieren:

```
sudo apt-get update
sudo apt-get upgrade -y
```

Um die Library runterzuladen führen Sie folgenden Befehl aus:

```
git clone https://github.com/rm-hull/luma.led_matrix.git
```

Wechseln Sie danach das Verzeichnis in den heruntergeladenen Ordner:

```
cd luma.led_matrix
```

Um die Library zu installieren führen Sie folgenden Befehl aus:

```
sudo python3 setup.py install
```

Python-Skript:

```
import time
from datetime import datetime
from luma.led_matrix.device import max7219
from luma.core.interface.serial import spi, noop
from luma.core.virtual import sevensegment

def date(seg):
    # Display current date on device
    now = datetime.now()
    seg.text = now.strftime('%d.%m.20%y')

def clock(seg, seconds): # Display current time on device
    for i in range(int(seconds / 0.5)):
        now = datetime.now()
        seg.text = now.strftime('%H-%M-%S')
        # calculate blinking dot
        if i % 2 == 0:
            seg.text = now.strftime('%H-%M-%S')
        else:
            seg.text = now.strftime('%H %M %S')

        time.sleep(0.5)

def show_message(seg, msg, delay=0.1): # Scrolls the message
    width = seg.device.width
    padding = ' ' * width
    msg = padding + msg + padding
    for i in range(len(msg)):
        seg.text = msg[i:i + width] # string slicing, msg[index:index]
        time.sleep(delay)
```

```
def main():
    serial = spi(port=0, device=0, gpio=noop())
    device = max7219(serial, cascaded=1)
    seg = sevensegment(device)

    print('Simple text...')
    seg.text = 'HELLO'
    time.sleep(2)
    seg.text = ' GOODBYE'
    time.sleep(2)

    print('Scrolling alphabet text...')
    show_message(seg, 'AZ-Delivery')
    show_message(seg, 'PI number 3.14159')
    s1 = '0123456789 '
    s2 = 'abcdefghijklmnopqrstuvwxyz '
    s3 = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    show_message(seg, '{}{}{}'.format(s1, s2, s3))

    date(seg)
    time.sleep(5)
    clock(seg, seconds=10)

    print('Brightness...')
    for x in range(5):
        for intensity in range(16):
            seg.device.contrast(intensity * 16)
            time.sleep(0.1)
    device.contrast(0x7F)

try:
    while True:
        main()
        time.sleep(1)

except KeyboardInterrupt:
    print('Script end!')
```

Speichern Sie dieses Skript unter dem Namen "*sevenSegments.py*". Um das Skript auszuführen, öffnen Sie das Terminal in dem Verzeichnis, in dem Sie das Skript gespeichert haben, und führen Sie folgenden Befehl aus: **`python3 Shock2.py`**

Um das Skript zu stoppen, drücken Sie STRG + C.

Das Skript beginnt mit dem Import von Libraries: Uhrzeit, Datum/Uhrzeit und modulbezogene Libraries.

Dann erstellen wir vier Funktionen.

Die erste Funktion heißt *`date()`*, sie wird zur Anzeige des aktuellen Datums verwendet. Sie akzeptiert ein Argument und gibt keinen Wert zurück. Das Argument heißt "*seg*" und stellt ein Modulobjekt dar (später im Text werden wir dieses Objekt ausführlich erklären). Um das aktuelle Datum zu erhalten, verwenden Sie folgende Zeile Code:

```
now = datetime.now()
```

Die Funktion gibt unix timestamp zurück, das ist die Anzahl der Sekunden vom *1. Januar 1970* bis zur aktuellen Zeit des Raspbian-Betriebssystems.

Um diesen Zeitstempel in eine lesbare Zeichenfolge umzuwandeln, verwenden Sie:

```
now.strftime('%d.%m.20%y')
```

Wobei *%d* den Tag, *%m* den Monat und *%y* das Jahr darstellt. Wir verwenden '*%d.%m.20%y*' was zum Beispiel '*29.11.2019*' ausgibt. Zur Ausgabe dieses Datumszeichenfolgenwerts verwenden wir die *seg.text*-Eigenschaft des "*seg*"-Objekts.

Die zweite Funktion heißt *clock()*, die zur Anzeige der Uhrzeit verwendet wird. Sie akzeptiert zwei Argumente und gibt keinen Wert zurück. Das erste Argument stellt das Modulobjekt dar, und das zweite Argument ist die Anzahl der Sekunden, ein Zeitintervall, nach dessen Ablauf die Uhrzeit auf dem Modul angezeigt wird. Innerhalb dieser Funktion gibt es eine for-loop-Funktion, die die Anzahl der Sekunden im *seconds*-Argument durchläuft.

In jedem Loop wird die aktuelle Zeit im Format '*HH-MM-SS*' für eine halbe Sekunde angezeigt, und die andere halbe Sekunde wird die gleiche Zeit angezeigt, ABER im Format '*HH MM SS*'. Dies erzielt den Effekt von blinkenden Strichen zwischen den Zeitangaben.

Die dritte Funktion heißt *show_message()*, die zur Anzeige von Lauftext auf dem Modul verwendet wird. Sie akzeptiert drei Argumente, wobei das letzte Argument optional ist. Das erste Argument stellt ein Modulobjekt dar, das zweite Argument ist eine Zeichenfolgennachricht, die gescrollt wird. Das letzte Argument, ein optionales Argument, ist eine Verschiebung des Bildlaufintervalls, oder man könnte sagen, eine Bildlaufgeschwindigkeit. Der Standardwert dieses Arguments ist 0,1 Sekunden, und Sie können ihn ändern, wenn Sie die Funktion auf die folgende Weise aufrufen:

```
show_message(seg, “message”, delay=1.5)
```

Innerhalb der Funktion *show_message()* durchlaufen wir die Zeichenkettennachricht und zerlegen sie in kleinere Zeichenketten mit der Länge der Modulbreite (die 8 Zeichen beträgt, aber wenn Sie mehrere Module innerhalb der Daisy Chain verwenden, ist dies ein höherer Wert). Nur geschnittene Zeichenketten werden auf dem Modul angezeigt, was den Effekt eines scrollenden Textes ergibt.

Die letzte Funktion heißt *main()*, mit der alle notwendigen Objekte erstellt und alle anderen Funktionen ausgeführt werden können. Sie akzeptiert keine Argumente und gibt keine Werte zurück. Zu Beginn dieser Funktion erstellen wir drei Objekte. Das erste Objekt ist für die SPI-Schnittstelle, wobei wir den SPI-Hardware-Port *0*, die Gerätenummer in einer Daisy Chain (in unserem Fall ist dies *0*, da wir nur ein Modul haben) und GPIO-Pin-Namen definieren. Das zweite Objekt wird verwendet, um alle Register innerhalb des Treiberchips MAX7219 einzurichten. Wir verwenden das SPI-Schnittstellenobjekt, um das Treiberchip-Objekt zu initialisieren.

Wenn wir das Treiberchip-Objekt initialisieren, können wir auch angeben, wie viele Module in der Daisy Chain kaskadiert sind (in unserem Fall ist dieser Wert 1). Und das letzte Objekt ist das Modul-Objekt, genannt "*seg*", und um es zu initialisieren, verwenden wir das Treiberchip-Objekt. Wir müssen dieses Objekt erstellen, da der Treiberchip MAX7219 auch zur Steuerung anderer Bildschirme verwendet wird. Mit dem "*seg*"-Objekt geben wir also an, dass wir sieben Segmente mit dem MAX7219-Treiberchip ansteuern.

Danach zeigen wir Text an und rufen andere Funktionen auf. Zuerst zeigen wir '*HELLO*' und '*GOODBYE*' an, dann scrollen wir durch drei Nachrichten. Die erste Nachricht ist '*AZ-Delivery*', die zweite ist '*PI-number 3.14159*' und die letzte enthält alle Zahlen und alle Klein- und Großbuchstaben. Die letzte Nachricht zur Verständlichkeit in drei Strings aufgeteilt.

Danach rufen wir die Funktionen *date()* und *clock()* auf. Wir setzen die Sekunden-Argumente der *clock()*-Funktion auf 10 Sekunden, so dass diese Funktion 10 Sekunden der aktuellen Zeit ausgibt.

Schließlich ändern wir am Ende der main()-Funktion die Helligkeit des Moduls. Wir tun dies mit der folgenden Zeile des Codes:

```
seg.device.contrast(number)
```

wobei die Zahl die Helligkeitsstufe ist. Es gibt 16 verschiedene Werte für die Helligkeitsstufe, aber dieser Wert muss eine Zahl sein, die, wenn sie durch 16 geteilt wird, eine ganze Zahl ergibt, da aufgrund des MAX7219-Treiberchips die internen Register auf diese Weise eingestellt werden.

Am Ende des Skripts erzeugen wir einen *try-except*-Block mit einem infinite-loop-Block (*while True:*) im Inneren. Im infinite-loop-Block rufen wir die *main()*-Funktion auf.

Um das Skript zu stoppen, drücken Sie STRG + C. Dies wird als Tastaturunterbrechung bezeichnet, und wir warten im Except-Block darauf:

```
except KeyboardInterrupt:
   print('Script end!')
```

Sie haben es geschafft. Sie können jetzt unser Modul für Ihre Projekte nutzen.

Jetzt sind Sie dran! Entwickeln Sie Ihre eigenen Projekte und Smart-Home Installationen. Wie Sie das bewerkstelligen können, zeigen wir Ihnen unkompliziert und verständlich auf unserem Blog. Dort bieten wir Ihnen Beispielskripte und Tutorials mit interessanten kleinen Projekten an, um schnell in die Welt der Mikroelektronik einzusteigen. Zusätzlich bietet Ihnen auch das Internet unzählige Möglichkeiten, um sich in Sachen Mikroelektronik weiterzubilden.

Falls Sie nach weiteren hochwertigen Produkten für Arduino und Raspberry Pi suchen, sind Sie bei AZ-Delivery Vertriebs GmbH goldrichtig. Wir bieten Ihnen zahlreiche Anwendungsbeispiele, ausführliche Installationsanleitungen, E-Books, Bibliotheken und natürlich die Unterstützung unserer technischen Experten.

https://az-delivery.de

Viel Spaß!

Impressum

https://az-delivery.de/pages/about-us