Willkommen!

Vielen Dank, dass sie sich für unser Echtzeituhrmodul mit dem "DS1302" von AZ-Delivery entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

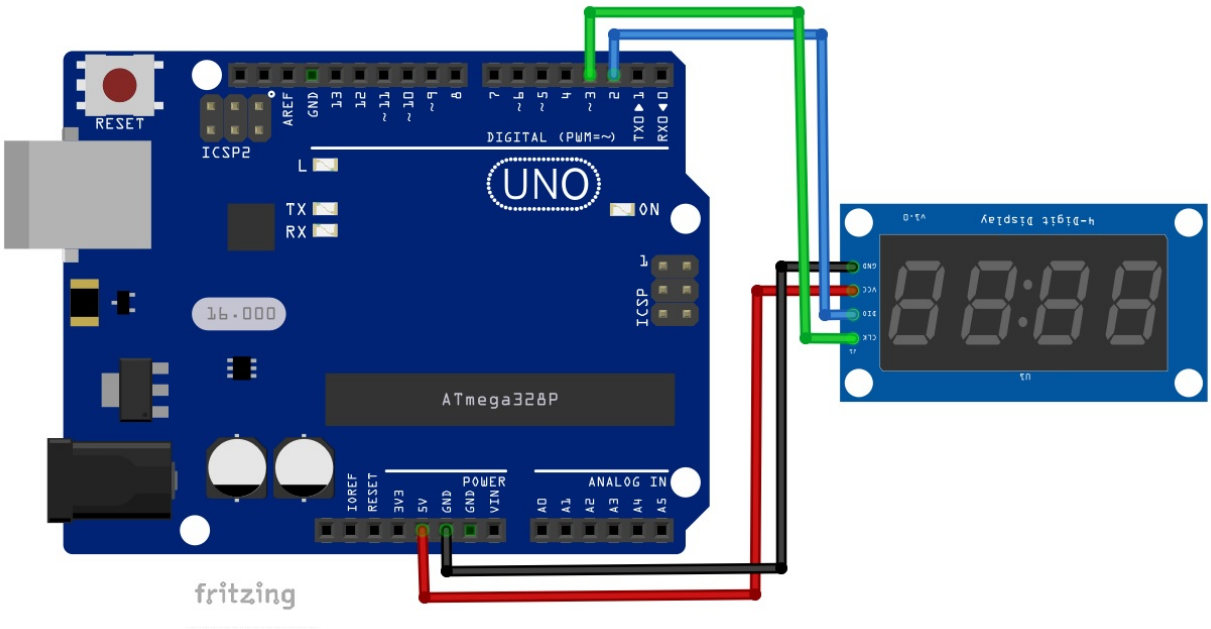
Viel Spaß!

Das 7-Segment-Display enthält sieben einzeln ansteuerbare LEDs in einem Gehäuse, welche von A bis G beschriftet sind. In dem 7-Segment-Display mit Platz für 4 Zahlen, gibt es vier Displays, so dass wir mit diesem Modul 4-stellig darstellen können. Diese Art von Modulen benötigt normalerweise 12 digitale Pins, um sie zu steuern, aber das Modul, über das wir in diesem eBook schreiben, hat nur vier. Der Grund dafür ist der Treiberchip "TM1637". Der Treiberchip ist eine spezielle Schaltung zum Ansteuern von LED-Anzeigen. Um mit dem Treiberchip zu kommunizieren, verwenden wir eine Form der Zwei-Draht-Schnittstelle. Es ist keine I2C-Schnittstelle, da der Treiberchip nicht wie die I2C-Schnittstelle kommuniziert. Wir verwenden 4 Drähte, um das Modul mit einem Mikrocontroller zu verbinden: zwei beliebige digitale Pins für Kommunikations-, Strom- und Massekabel.

Technische Daten:

» Spannungsversorgungs- und Logikbereich:	von 3,3V bis 5V DC
» Betriebsgleichstrom:	30mA - 80mA
» LED-Farbe:	ROT
» Betriebstemperatur:	von -10°C bis 80°C
» Veränderbare Helligkeit:	in der Software

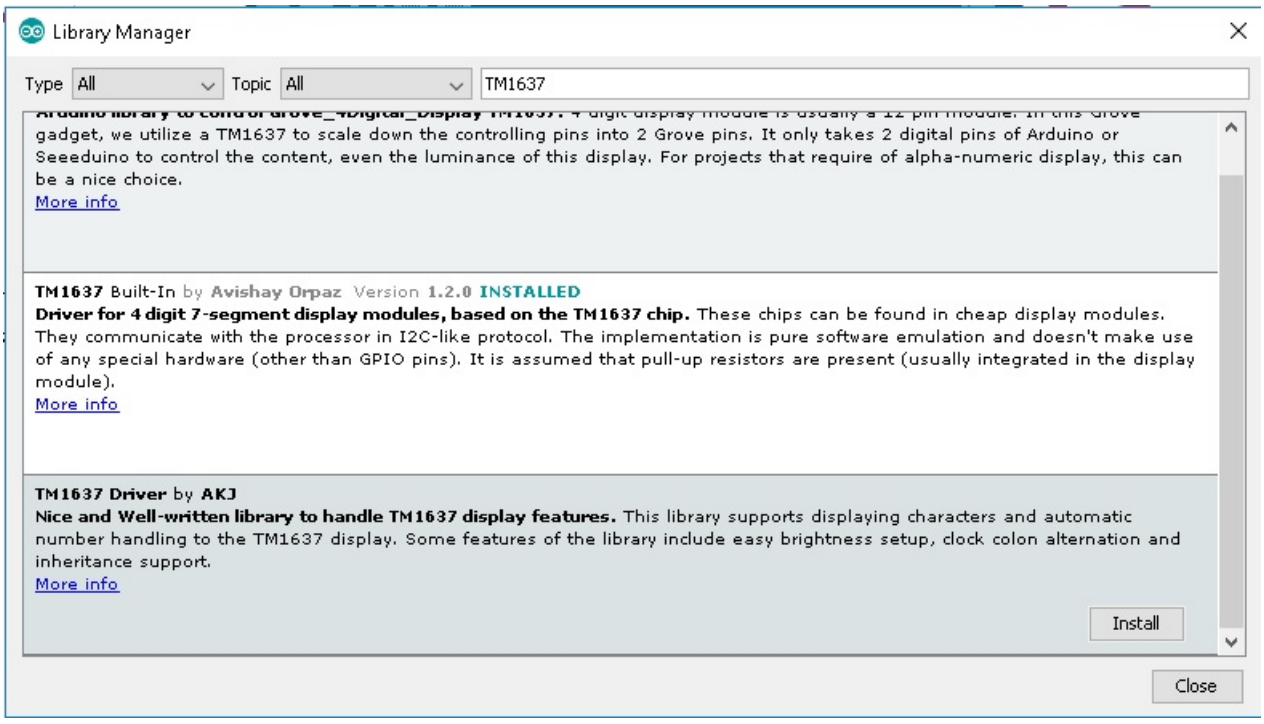
Verbindung des Moduls mit dem Uno



Module Pin	>	Uno Pin	
GND	>	GND	**Schwarzer Draht**
VCC	>	5V	**Roter Draht**
DIO	>	D3	**Grüner Draht**
CLK	>	D2	**Blauer Draht**

Arduino IDE library

Um das Modul mit dem Uno zu verwenden, ist es ratsam, eine library dafür herunterladen. Öffnen Sie die Arduino-IDE und gehen Sie zu *Tools* > *Manage Libraries*, und ein neues Fenster wird geöffnet. Geben Sie "TM1637" in das Suchfeld ein und installieren Sie die library "TM1637" von "Avishay Orpaz", wie unten abgebildet:



Um ein Sketch-Beispiel auszuführen, öffnen Sie folgenden Sketch

File > Examples > TM1637 > TM1637Test

Sketch-Code:

```
#include <Arduino.h>
#include <TM1637Display.h>
#define CLK   2
#define DIO   3
#define TEST_DELAY   2000 //  delay in miliseconds
const uint8_t SEG_DONE[] = {
  SEG_B | SEG_C | SEG_D | SEG_E | SEG_G,           // d
  SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F,   // O
  SEG_C | SEG_E | SEG_G,                           // n
  SEG_A | SEG_D | SEG_E | SEG_F | SEG_G };         // E
TM1637Display display(CLK, DIO);
uint8_t data[] = {0xff, 0xff, 0xff, 0xff};
uint8_t blank[] = {0x00, 0x00, 0x00, 0x00};
void setup() {
  display.setBrightness(0x0f);
}
void loop() {
  turnON_allSegments();
  turnON_segment();
  display_numbers();
  brightness_test();
  ON_OFF_test();
  display_DONE();
  delay(5000);
}
void turnON_allSegments() {
  display.setSegments(data);
```

```
  delay(TEST_DELAY);
}

void turnON_segment() {
  // Selectively set different digits
  data[0] = display.encodeDigit(0);
  data[1] = display.encodeDigit(1);
  data[2] = display.encodeDigit(2);
  data[3] = display.encodeDigit(3);
  display.setSegments(data);
  delay(TEST_DELAY);
  display.clear();
  display.setSegments(data+2, 2, 2);
  delay(TEST_DELAY);
  display.clear();
  display.setSegments(data+2, 2, 1);
  delay(TEST_DELAY);
  display.clear();
  display.setSegments(data+1, 3, 1);
  delay(TEST_DELAY);
}
void display_numbers() {
  // How to show decimal numbers in many ways
  display.showNumberDec(0, false); // Expect: ___0
  delay(TEST_DELAY);
  display.showNumberDec(0, true);  // Expect: 0000
  delay(TEST_DELAY);
  display.showNumberDec(1, false); // Expect: ___1
  delay(TEST_DELAY);
  display.showNumberDec(1, true);  // Expect: 0001
  delay(TEST_DELAY);
  display.showNumberDec(301, false); // Expect: _301
```

```
  delay(TEST_DELAY);
  display.showNumberDec(301, true); // Expect: 0301

// one tab
  delay(TEST_DELAY);
  display.clear();
  display.showNumberDec(14, false, 2, 1); // Expect: _14_
  delay(TEST_DELAY);
  display.clear();
  display.showNumberDec(4, true, 2, 2);  // Expect: __04
  delay(TEST_DELAY);
  display.showNumberDec(-1, false);  // Expect: __-1
  delay(TEST_DELAY);
  display.showNumberDec(-12);        // Expect: _-12
  delay(TEST_DELAY);
  display.showNumberDec(-999);       // Expect: -999
  delay(TEST_DELAY);
  display.clear();
  display.showNumberDec(-5, false, 3, 0); // Expect: _-5_
  delay(TEST_DELAY);
  display.showNumberHexEx(0xf1af);        // Expect: f1Af
  delay(TEST_DELAY);
  display.showNumberHexEx(0x2c);          // Expect: __2C
  delay(TEST_DELAY);
  display.showNumberHexEx(0xd1, 0, true); // Expect: 00d1
  delay(TEST_DELAY);
  display.clear();
  display.showNumberHexEx(0xd1, 0, true, 2); // Expect: d1__
delay(TEST_DELAY);
}
```

```
void turn_dots() {
  for(int k = 0; k <= 4; k++) {
    display.showNumberDecEx(0, (0x80 >> k), true);
    delay(TEST_DELAY);
  }
}
void brightness_test() {
  for(int k = 0; k < 4; k++) {
    data[k] = 0xff;
  }
  for(int k = 0; k < 7; k++) {
    display.setBrightness(k);
    display.setSegments(data);
    delay(TEST_DELAY);
  }
  display.setBrightness(4);
}
void ON_OFF_test() {
  for(int k = 0; k < 4; k++) {
    display.setBrightness(7, false);  // Turn off
    display.setSegments(data);
    delay(TEST_DELAY);
    display.setBrightness(7, true); // Turn on
    display.setSegments(data);
    delay(TEST_DELAY);
  }
}
void display_DONE() {
  display.setSegments(SEG_DONE);
}
```

Am Anfang der Skizze werden zwei **Makros** erstellt, die CLK und DIO heißen. Diese Makros stellen digitale Pinnamen des Uno dar, die verwendet werden, um das Modul mit dem Uno zu verbinden.

Danach wird ein weiteres Makro, mit Namen *TEST_DELAY,* erstellt; es ist ein Zeitintervall zwischen zwei Anzeigefunktionen und wird auf 2000 Millisekunden oder 2 Sekunden initialisiert.

Nach den Makros erstellen wir ein Constant-Array und zwei variable Arrays. Das Constant-Array heißt SEG_DONE und wird verwendet, um vier Elemente zu speichern. Jedes Element enthält Makros für das Segment, das, auf der siebenstelligen Segmentziffer des Moduls, eingeschaltet ist. Um den Buchstaben "d" anzuzeigen, müssen wir daher diese Segmente einschalten (oder diese Makros in einem Element des Arrays speichern): SEG_B, SEG_C, SEG_D, SEG_E und SEG_G.
Wir verwenden dieses Constant-Array, um das Wort "dOnE" anzuzeigen.
Die anderen Arrays heißen *`data[]`* und *`blank[]`*. Beide Arrays enthalten vier Elemente. Ein Element für jede Siebensegmentanzeige auf dem Modul. Mit dem Leerzeichen[ ] werden alle Segmente des Moduls **ausgeschaltet**.
`Data[]` wird verwendet, um bestimmte Zeichen auf dem Modul anzuzeigen. Später im Text werden wir Beispiele für die Verwendung dieses Arrays näher besprechen.

In der `setup`() Funktion verwenden wir die `setBrightness`() Funktion, um die Helligkeit des Displays einzustellen. Diese Funktion akzeptiert ein Argument und ist eine hexadezimale Zahl; der maximale Helligkeitswert ist *0x0f*.

In der `loop()` Funktion rufen wir alle anderen Funktionen auf, die wir in der Skizze erstellt haben, und warten 5 Sekunden zwischen jeder Wiederholung der `loop()` Funktion.

Die erste Funktion, die in der Funktion `loop()` verwendet wird, ist `turnON_allSegments()`, und sie wird verwendet, um alle Segmente des Moduls einzuschalten. Warten Sie dann 2 Sekunden lang (`TEST_DELAY` Makrowert). Dies geschieht durch die Verwendung der folgenden Zeile des Codes: `display.setSegments(data)`
`setSegments()` ist eine Funktion, die ein Argument akzeptiert, und es ist `data[]` array, das wir zuvor erstellt haben.

Die nächste Funktion ist `turnON_segment()`. Sie wird verwendet, um zu zeigen, wie vielfältig man die Funktion `setSegments()` verwenden kann. Zuerst kodieren wir die Ziffern 0, 1, 2 und 3 und speichern sie im `data[]` Array. Dann geben wir das mit der folgenden Zeile des Codes aus: `display.setSegments(data)`.
Danach werden drei Beispiele angeführt, wie man das `Data[]` Array auf den Segmenten des Moduls verschiebt. Um die Anzeige zu löschen oder die Module **auszuschalten**, können Sie eine andere Methode verwenden, indem Sie die folgende Funktion aufrufen:
`display.clear()`

Mit der Funktion `display_numbers()` wird auf die verschiedenen Arten, wie eine bestimmte Zahl angezeigt werden kann, hingewiesen. Jede Zeile dieser Funktion wird kommentiert und näher erläutert.

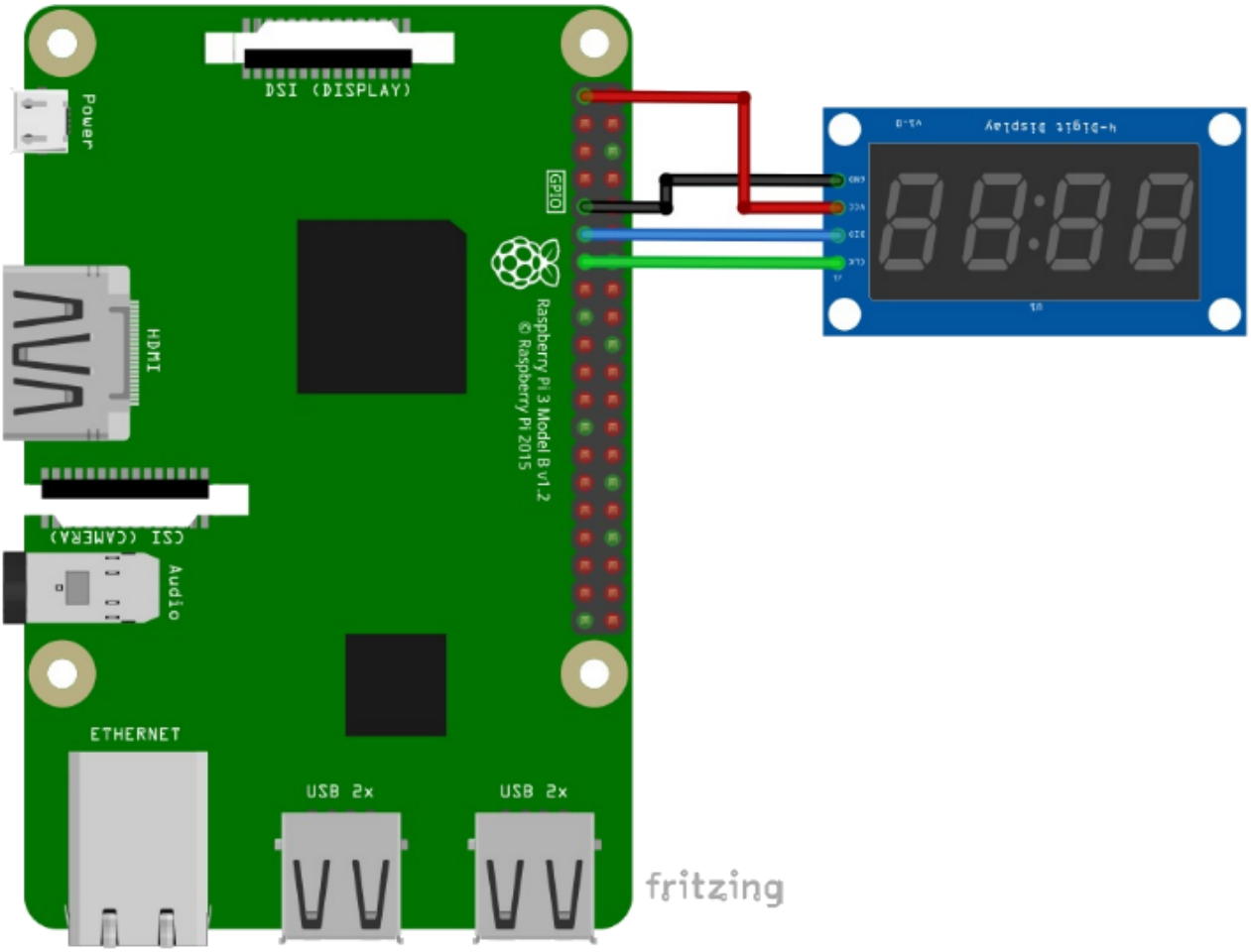
Die Funktion `turn_dots()` wird verwendet, um zwei LEDs zwischen dem zweiten und dritten Segment des Siebensegments des Moduls zu wechseln.

Mit der Funktion `brightness_test()` werden alle Helligkeitsstufen angezeigt, die für dieses Modul verwendet werden können.

Die Funktion `ON_OFF_test()` zeigt Ihnen eine weitere Möglichkeit, wie Sie alle Segmente des Moduls *ein- oder ausschalten* können.

Und mit der Funktion `display_DONE()` wird das Wort "dOnE" aus dem *SEG_DONE*-Array angezeigt.

Verbindung des Moduls mit dem Raspberry Pi



Modul Pin	>	Raspberry Pi Pin		
VCC	>	3V3	[Pin 1]	**Roter Draht**
GND	>	GND	[Pin 9]	**Schwarzer Draht**
DIO	>	GPIO17	[Pin 11]	**Blauer Draht**
CLK	>	GPIO27	[Pin 13]	**Grüner Draht**

Python-library

Um das Modul mit dem Raspberry Pi zu verwenden, wird empfohlen, eine library herunterzuladen. Wir werden die library "*raspberrypi-python-tm1637*" verwenden. Um diese library herunterladen und installieren zu können, benötigen wir die wiringPi- und GIT-App. Diese Tools sind auf Raspbian vorinstalliert. Sollte diese nicht installiert sein, erfahren Sie hier, wie Sie weiter vorgehen. Öffnen Sie das Terminal und führen Sie die folgenden Befehle aus:

`sudo apt install git -y` - für *git*

und

`sudo pip3 install wiringpi` - für *wiringPi*

Um die library herunterzuladen, führen Sie folgenden Befehl aus:

git clone https://github.com/depklyon/raspberrypi-python-tm1637.git

Ändern Sie danach das Verzeichnis, um den Ordner mit:

`cd raspberry-python-tm1637`

herunterzuladen

Um diese zu installieren, führen Sie folgenden Befehl aus:

`sudo python3 setup.py install`

Python-Skript:

```
import tm1637
from time import sleep
tm = tm1637.TM1637(clk=27, dio=17)
def show():
    tm.write([127, 255, 127, 127]) # all LEDS on "88:88"
    sleep(1)
    tm.write([0, 0, 0, 0]) # all LEDS off
    sleep(1)
    tm.write([63, 6, 91, 79]) # show "0123"
    sleep(1)
    tm.write([0b00111001, 0b00111111, 0b00111111, 0b00111000]) # "COOL"
    sleep(1)
    tm.show('help') # show "HELP"
    sleep(1)
    tm.hex(0xdead) # display "dEAd"
    sleep(1)
    tm.hex(0xbeef) # display "bEEF"
    sleep(1)
    tm.numbers(11, 55) # show "11:55"
    sleep(1)
    tm.number(-955) # show "-955"
    sleep(1)
    tm.temperature(22) # show temperature '22*C'
    sleep(1)
print('[press ctrl + c to stop the script]')
try:
    while True:
        show()
        sleep(1)
except KeyboardInterrup:
    print('Script end!')
finally:
    tm.write([0, 0, 0, 0])
```

Speichern Sie diesen Code in einem Skript namens "*sevenSegment.py*" und stellen Sie sicher, dass das Skript im Verzeichnis "*raspberrypi-python-tm1637*" gespeichert ist. Um das Skript auszuführen, öffnen Sie das Terminal in diesem Verzeichnis und führen Sie den folgenden Befehl aus:

```
python3 sevenSegment.py
```

Hinweis: Skript-Code ist selbsterklärend

Sie haben es geschafft. Sie können jetzt unser Modul nun 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 noch 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

Have Fun!

Impressum

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