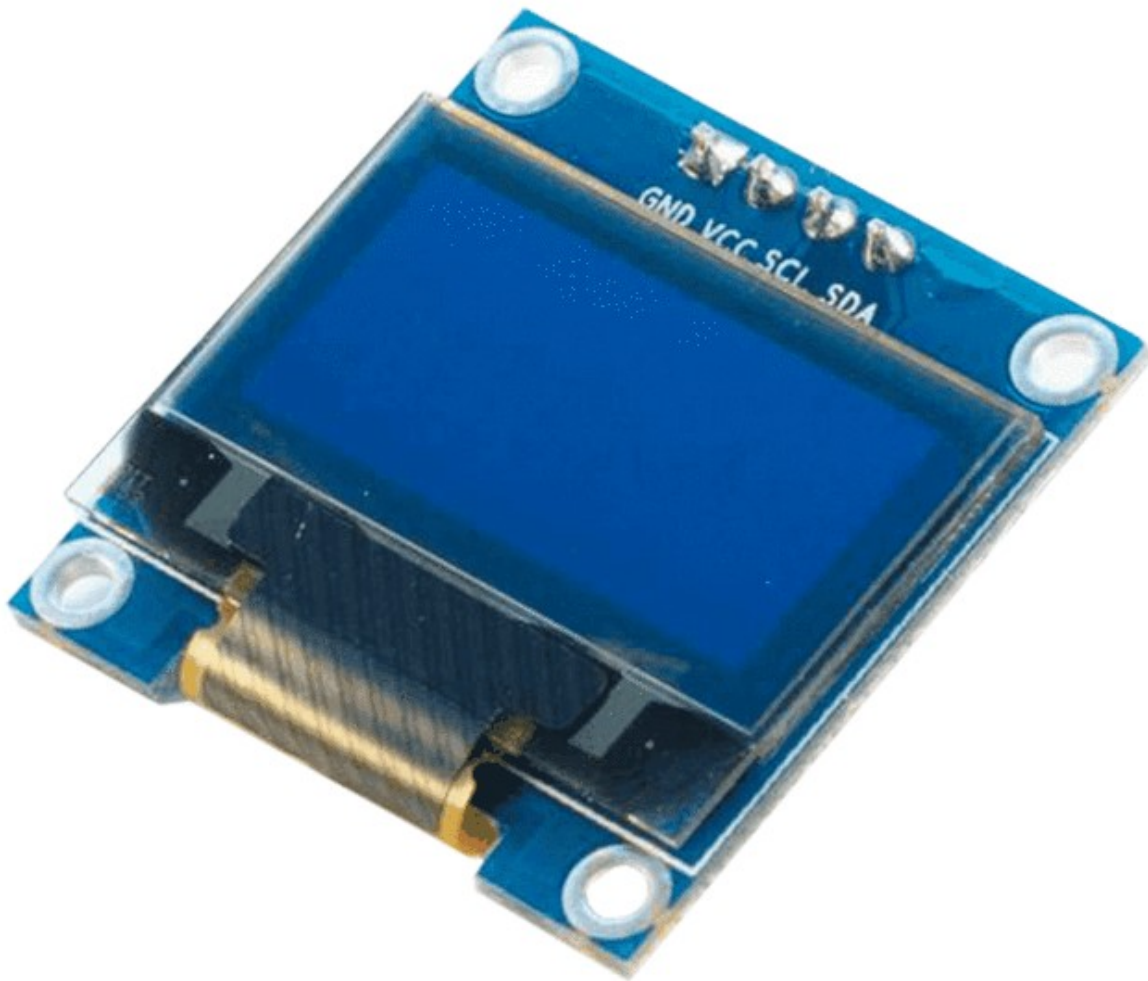


AZ-Delivery

Willkommen!

Vielen Dank, dass sie sich für unser AZ-Delivery 0.96" OLED Display entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

Viel Spaß!



Az-Delivery

OLED- und AMOLED-Displays ab den 2000er Jahren auf. OLEDs sind organische Leuchtdioden, und AMOLEDs sind organische Leuchtdioden mit aktiver Matrix, die auf OLEDs basieren.

Eine der Funktionen des Treiberchips SSD1306 besteht darin, jedes Pixel aus- und einzuschalten zu können. Die zum Betrieb dieses OLED-Displays und dem SSD1306-Treiberchips erforderliche Versorgungsspannung beträgt 3,3 V. Es gibt einen On-Board-Regler, durch den wir diese Displays mit einer 5V-Stromversorgung verwenden können. Es handelt sich um ein 128x64-Display (Höhe x Breite), was bei der Verwendung des Konstruktors am Anfang des Sketches zu beachten ist.

Das I2C-Kommunikationsprotokoll wird zur Steuerung der OLED-Anzeige verwendet. I2C, oder IIC, steht für "Inter Integrated Circuit". Es handelt sich um einen synchronen seriellen Computerbus. Synchron bedeutet eine ganze Umdrehung innerhalb eines bestimmten Taktzyklus, und seriell bedeutet, dass die Daten seriell, Bit für Bit, gesendet werden.

Folglich sind für diese Kommunikation nur 2 Drähte erforderlich:

- "SCL - Serielle Taktleitung

- "SDA - Serielle Datenleitung

Beide dieser Leitungen arbeiten im "Open-Drain-Modus", was bedeutet, dass das Gerät, welches das I2C-Protokoll verwendet, die SCL- oder SDA-Leitung low betreiben kann, während er nicht in der Lage ist, sie auf high zu betreiben.

AZ-Delivery

Die Leitungen sind high, wenn sie nicht heruntergezogen sind, was auf Pull-up-Widerstände zurückzuführen ist. Der I2C erfordert Pull-up-Widerstände auf beiden Seiten mit einem Widerstand zwischen 1kΩ und 10kΩ, wobei der am häufigsten verwendete Widerstand 4,7kΩ ist. Der Vorteil der Verwendung dieses OLEDs mit Microcontrollern besteht darin, dass Microcontroller über eingebaute Pull-up-Widerstände verfügt, wodurch Sie beim Einrichten Ihres Displays Zeit sparen.

Um das OLED-Display einzurichten, verbinden wir einfach den SDA-Pin (A4) des Microcontrollers mit dem SDA-Pin des Displays, sowie den SCL-Pin (A5) des Microcontrollers mit dem SCL-Pin des Displays. Natürlich ist die Programmierung des Microcontrollers erforderlich, um das Display zu steuern. Alle Geräte, die über I2C-Protokollleitungen verbunden sind, haben ihre eigenen Adressen, die für das Master-Gerät (den Mikrocontroller auf den Boards) erforderlich sind, um zu wissen, mit welchem Slave-Gerät (in diesem Fall das Display) kommuniziert werden soll.

Alle OLED-Displays, die AZ-Delivery anbietet, haben die gleichen Adressen, "0x3C". Das 0,96-Zoll-Display bietet Ihnen die Möglichkeit, einen Widerstand auf der Platine des Displays für eine andere Adresse, "0x3D", neu zu verlöten. Allerdings ist es nicht empfehlenswert, dies zu tun. Falls Sie mehrere Displays mit einem Microcontroller verwenden müssen, empfiehlt es sich, ein I2C-Multiplexer-Gerät oder einfach ein größeres Display zu verwenden.

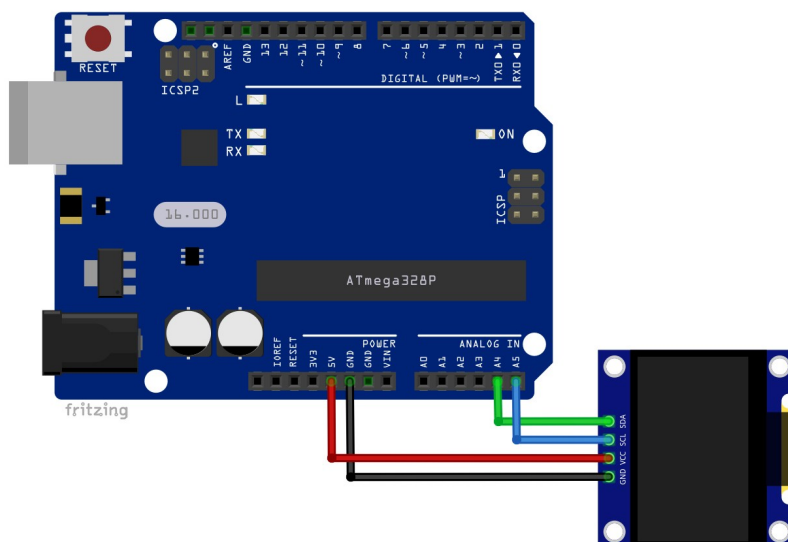
Az-Delivery

Verbindung des Displays mit dem Microcontroller

Verbinden Sie alles, wie unten abgebildet:



0-96 inch OLED I2C Display
Connection diagram with Uno



OLED Display pin	>	MC pin
VCC	>	5V
GND	>	GND
SCL	>	A5
SDA	>	A4

Roter Draht

Schwarzer Draht

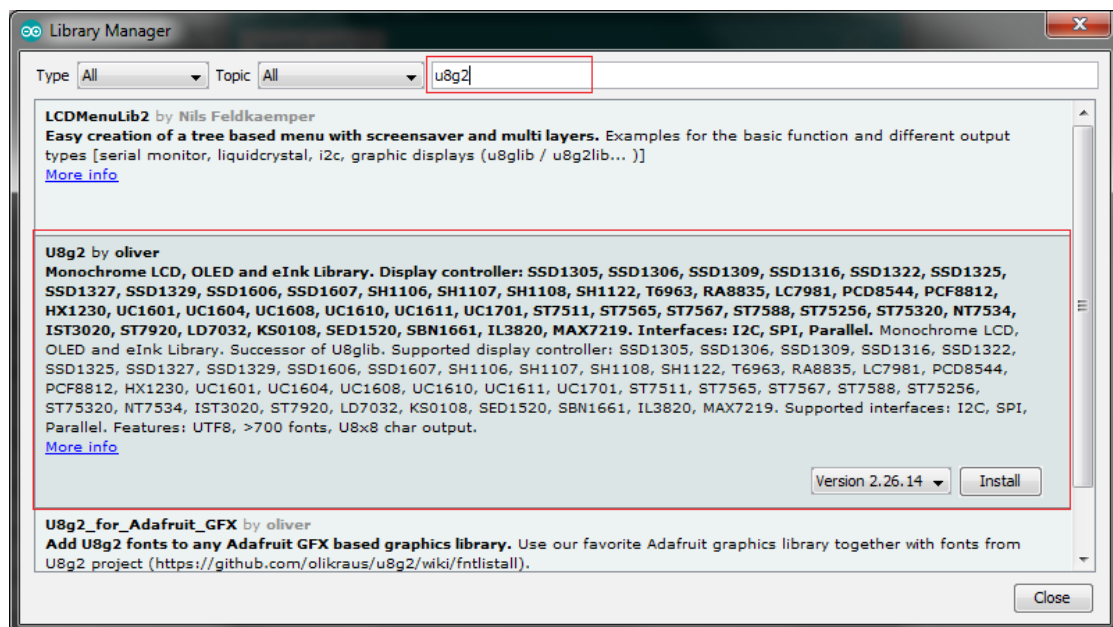
Grüner Draht

Blauer Draht

Az-Delivery

Library

Wir werden die “*u8g2*”-Library benutzen. Um diese Library runterzuladen, öffnen Sie ihr Arduino IDE. Gehen Sie auf *Tools > Manage Libraries*. In einem sich neu öffnenden Fenster geben Sie in dem Suchfeld “*u8g2*” ein und installieren Sie die “*U8g2*” library von “*oliver*”, wie unten abgebildet.



Die Installation ist sehr einfach. Klicken Sie dafür nur auf die Schaltfläche "Installieren", die erscheint, wenn Sie mit der Maus über das Suchergebnis fahren. Als nächstes gehen Sie zu:

File > Examples > U8g2 > full_buffer > Graphics Test

um das Gerät zu testen. Der im Beispiel verwendete Code ist jedoch ziemlich komplex, und deshalb werden wir eine einsteigerfreundlichere Version des Codes für die Verwendung von Library-funktionen erstellen.

Az-Delivery

Code:

```
#include <U8g2lib.h>
#include <Wire.h>
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE);
const char COPYRIGHT_SYMBOL[] = { 0xa9, '\\0' };
void u8g2_prepare() {
    u8g2.setFont(u8g2_font_6x10_tf);
    u8g2.setFontRefHeightExtendedText();
    u8g2.setDrawColor(1);
    u8g2.setFontPosTop();
    u8g2.setFontDirection(0);
}
void u8g2_box_frame() {
    u8g2.drawStr(0, 0, "drawBox");
    u8g2.drawBox(5, 10, 20, 10);
    u8g2.drawStr(60, 0, "drawFrame");
    u8g2.drawFrame(65, 10, 20, 10);
}
void u8g2_r_frame_box() {
    u8g2.drawStr(0, 0, "drawRFrame");
    u8g2.drawRFrame(5, 10, 40, 15, 3);
    u8g2.drawStr(70, 0, "drawRBox");
    u8g2.drawRBox(70, 10, 25, 15, 3);
}
void u8g2_disc_circle() {
    u8g2.drawStr(0, 0, "drawDisc");
    u8g2.drawDisc(10, 18, 9);
    u8g2.drawDisc(30, 16, 7);
    u8g2.drawStr(60, 0, "drawCircle");
    u8g2.drawCircle(70, 18, 9);
    u8g2.drawCircle(90, 16, 7);
}
```

Az-Delivery

```
void u8g2_string_orientation() {
    u8g2.setFontDirection(0);
    u8g2.drawStr(5, 15, "0");
    u8g2.setFontDirection(3);
    u8g2.drawStr(40, 25, "90");
    u8g2.setFontDirection(2);
    u8g2.drawStr(75, 15, "180");
    u8g2.setFontDirection(1);
    u8g2.drawStr(100, 10, "270");
}

void u8g2_line() {
    u8g2.drawStr( 0, 0, "drawLine");
    u8g2.drawLine(7, 10, 40, 32);
    u8g2.drawLine(14, 10, 60, 32);
    u8g2.drawLine(28, 10, 80, 32);
    u8g2.drawLine(35, 10, 100, 32);
}

void u8g2_triangle() {
    u8g2.drawStr( 0, 0, "drawTriangle");
    u8g2.drawTriangle(14, 7, 45, 30, 10, 32);
}

void u8g2_unicode() {
    u8g2.drawStr(0, 0, "Unicode");
    u8g2.setFont(u8g2_font_unifont_t_symbols);
    u8g2.setFontPosTop();
    u8g2.setFontDirection(0);
    u8g2.drawUTF8(10, 15, "☀");
    u8g2.drawUTF8(30, 15, "☁");
    u8g2.drawUTF8(50, 15, "☂");
    u8g2.drawUTF8(70, 15, "☃");
    u8g2.drawUTF8(95, 15, COPYRIGHT_SYMBOL); //COPYRIGHT SYMBOL
    u8g2.drawUTF8(115, 15, "\xb0"); // DEGREE SYMBOL
}
```

Az-Delivery

```
void setup(void) {
    u8g2.begin();
    u8g2.prepare();
}

void loop(void) {
    u8g2.clearBuffer();
    u8g2.prepare();
    u8g2_box_frame();
    u8g2.sendBuffer();
    delay(1500);
    u8g2.clearBuffer();
    u8g2_disc_circle();
    u8g2.sendBuffer();
    delay(1500);
    u8g2.clearBuffer();
    u8g2_r_frame_box();
    u8g2.sendBuffer();
    delay(1500);
    u8g2.clearBuffer();
    u8g2_prepare();
    u8g2_string_orientation();
    u8g2.sendBuffer();
    delay(1500);
    u8g2.clearBuffer();
    u8g2_line();
    u8g2.sendBuffer();
    delay(1500);
    u8g2.clearBuffer();
    u8g2_triangle();
    u8g2.sendBuffer();
    delay(1500);
    u8g2.clearBuffer();
    u8g2_prepare();
    u8g2_unicode();
    u8g2.sendBuffer();
    delay(1500);
}
```


Az-Delivery

Hier gehen wir die Funktionen durch und erklären was sie tun; Wir fangen mit "u8g2_prepare()" an, die einige "library"-Funktionen enthält:

» Die **setFont()** Funktion wird benutzt, um die Schriftart von Zeichen einzurichten. In dieser Funktion wird das Argument "*u8g2_font_5x10_tf*" benutzt, aber Sie können sich auch ein Anderes aussuchen, unter

<https://github.com/olikraus/u8g2/wiki/fntlist8x8>

» In der **setFontRefHeightExtendedText()** Funktion werden Zeichen ausgewählt. Eine detailliertere Erklärung finden sie unter

<https://github.com/olikraus/u8g2/wiki/u8g2reference#setFontrefheightextendedtext>

» Unter **setDrawColor()**, leuchtet nur Zeichen auf, wenn Sie das Argument "1" benutzen. Wenn Sie das Argument "0" benutzen, werden die Pixel für jeden Buchstaben invertiert, das heißt, dass der Hintergrund, aber nicht das Zeichen beleuchtet wird. Das Argument "2", liefert dasselbe Ergebnis wie Argument - 0.

» **setFontPosTop()** gibt es in drei Variationen: **setFontPosBaseline()**, **setFontPosCenter()** und **setFontPosBottom()**. Diese ändern die Position der Zeichen in der Zeile.

» Um Zeichenfolgen auf dem Display anzuzeigen, verwenden wir die **drawStr()** Funktion. Sie hat drei Argumente. Die X-Position und Y-Position der Zeichenfolge und die eigentliche Zeichenkette. Bevor wir diese Funktion nutzen, sollten wir "*u8g2_prepare()*" verwenden, um die Schriftart für den anzuzeigenden Text auszuwählen.

Az-Delivery

Um Formen anzuzeigen, müssen wir für jede spezifische Funktionen verwenden:

» ***drawBox()*** wird verwendet, um eine Box anzuzeigen. Sie verwendet vier Argumente, nämlich X- und Y-Positionen der linken oberen Ecke der Box, das dritte ist die Breite und das vierte die Höhe.

» ***drawFrame()***, wird verwendet, um einen Rahmen anzuzeigen. Es verwendet vier Argumente: Die X- und Y-Position der linken oberen Ecke des Rahmens, die Breite und die Höhe.

» ***DrawCircle()***, wird zum Zeichnen eines Kreises verwendet. Es verwendet drei Argumente: Die X- und Y-Positionen des Kreismittelpunktes und Kreisradius.

» ***drawDisc()***, zeichnet einen gefüllten Kreis mit Radius. Diese hat drei Argumente: Die X- und Y-Position eines Objekts und der Radius.

» ***drawLine()***, wird verwendet, um eine Linie anzuzeigen, es verwendet vier Argumente: Die X- und Y-Positionen des ersten Punktes der Linie und die X- und Y-Positionen des zweiten Punktes der Linie.

» ***drawTriangle()***, wird verwendet, um ein Dreieck anzuzeigen. Die Argumente, die es verwendet, sind so ähnlich wie bei der Linie, aber mit einem zusätzlichen Punkt also insgesamt sechs Argumente: X- und Y-Positionen des ersten Punktes, X- und Y-Positionen des zweiten Punktes und schließlich X- und Y-Positionen des dritten Punktes.

» Außerdem gibt es Funktionen zur Anzeige von Boxen und Rahmen mit einem festgelegten Eckenradius. ***DrawRFrame()*** und ***drawRBox()*** Funktionen sind so ähnlich wie ***drawFrame()*** und ***drawBox()***

Az-Delivery

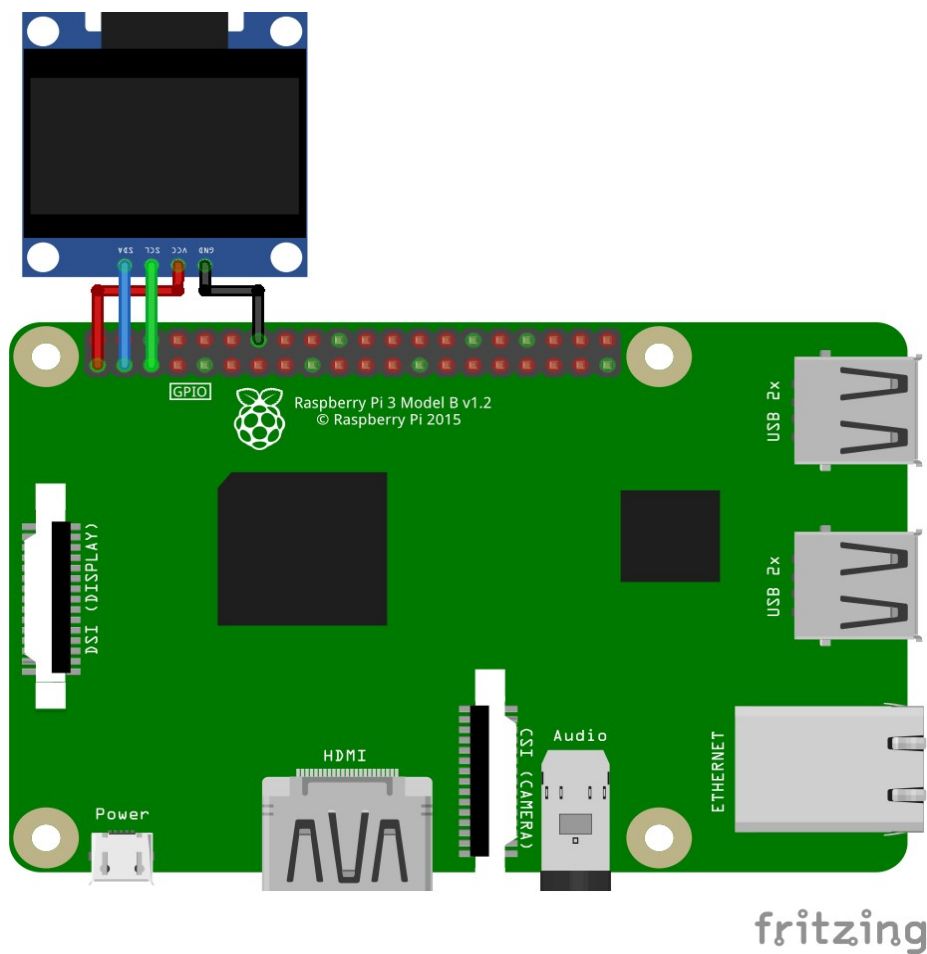
Funktionen. Allerdings akzeptieren ***drawRFrame()*** und ***drawRBox()*** Funktionen ein fünftes Argument als Radius-Argument.

» ***drawUTF8()*** wird zur Anzeige von "Unicode"-Zeichen verwendet. Seine Argumente sind die X- und Y-Positionen des Zeichens sowie das Zeichen-Argument für das Zeichen selbst. Um diese Zeichen anzuzeigen, können Sie entweder:

- » Kopieren Sie das Zeichen und fügen Sie es in den Sketch ein,
- » erstellen Sie eine Symbolvariable, ein Char-Array mit zwei Werten: eine hexadezimale "Unicode"-Nummer für ein Sonderzeichen und ein Escape-Zeichen (wie in unserem Beispiel mit der Variablen `COPYRIGHT_SYMBOL`),
- » eine hexadezimale Zahl in der Zeichenfolge verwenden, wie z.B. `"\xb0"`, das in unserem Beispiel das Gradsymbol ist.

Verbindung des Displays mit dem Raspberry Pi

Als Erstes verbinden Sie alles, wie unten abgebildet:

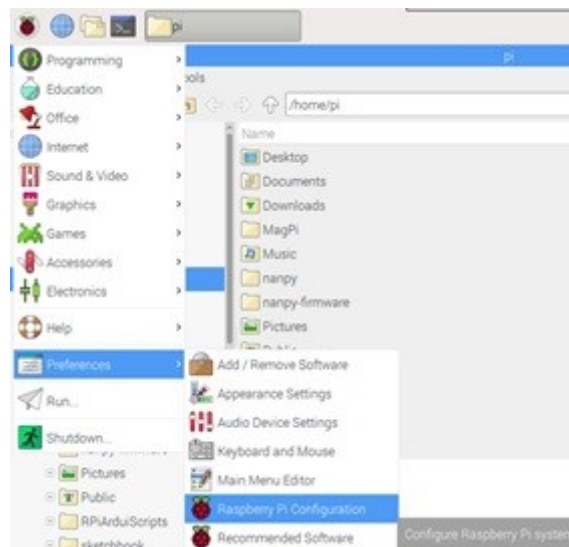


OLED Display pin	>	Raspberry pin	
VCC	>	3,3V [pin 1]	Roter Draht
GND	>	GND [pin 14]	Schwarzer Draht
SCL	>	GPIO3 [pin 5]	Grüner Draht
SDA	>	GPIO2 [pin 3]	Blauer Draht

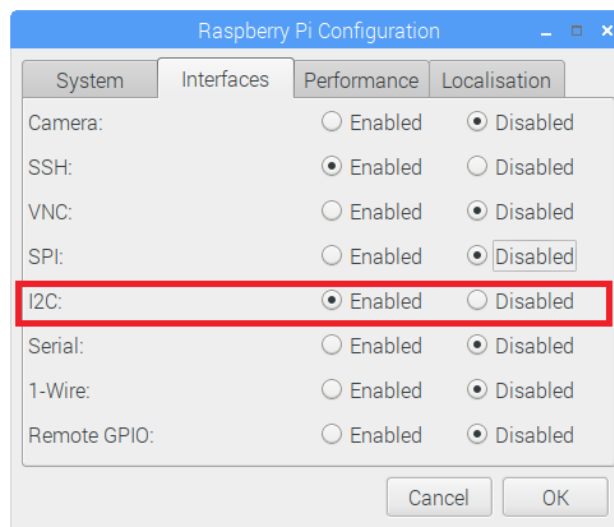
Az-Delivery

Aktivierung der I2C-Schnittstelle auf Raspbian

Dafür gehen Sie zu *Preferences > Raspberry Pi Configuration* wie unten abgebildet:

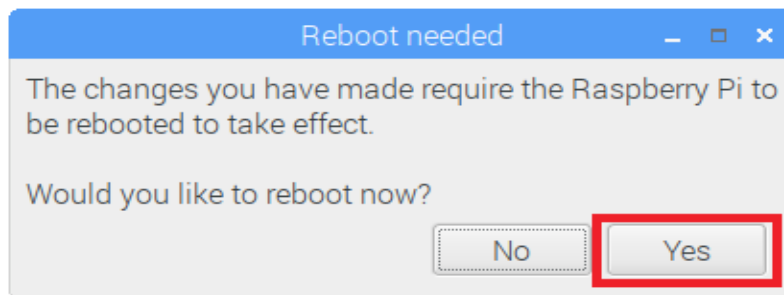


Als nächstes öffnen Sie den *“Interfaces”* Tab, finden Sie die *“I2C”* Taste and und schalten diese an, wie unten abgebildet:



Az-Delivery

Sie werden aufgefordert, das System neu zu starten. Wir empfehlen Ihnen, dies durch Klicken auf "Ja" zu tun, wie in der Abbildung unten dargestellt:



Az-Delivery

Finden der Adresse des OLED-Moduls

Wenn es aktiviert ist, werden wir den Befehl *i2detect* verwenden, um das Modul am I2C-Bus zu finden: **i2cdetect -y 1**

Das Ergebnis sollte so aussehen, wie unten abgebildet:

```
pi@rpi3py:~ $ i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Unser Gerät wurde mit der Adresse "0x3c" erkannt. Dies ist die Standardadresse für diese Art von Gerät.



Python-library

Für die Anzeige von Formen, Text und Bildern werden wir ein Python-library verwenden. Auf dem aktuellen Raspberry Pi OS ist Python3, pip3 und git bereits vorinstalliert, sollte das jedoch nicht der Fall sein können Sie das ganze mit folgenden Befehlen nachinstallieren:

```
sudo apt-get install python3-dev libffi-dev libssl-dev  
python3-pil libjpeg-dev zlib1g-dev libfreetype6-dev  
liblcms2-dev libopenjp2-7 libtiff5 -y
```

```
sudo apt-get install python3-rpi.gpio python3-pip -y
```

```
sudo apt-get install git -y
```

Als Bibliothek verwenden wir die "luma.oled" diese kann mit folgendem Befehl installiert werden:

```
sudo -H pip3 install luma.oled
```




Python-Skript

Im Ordner "pi" erstellen wir jetzt einen Ordner namens "oled" und begeben uns in diesen Ordner

```
sudo mkdir oled  
cd oled
```

luma.oled bietet viele Beispiele und diese können wir uns mit folgendem Befehl herunterladen:

```
sudo git clone  
https://github.com/rm-hull/luma.examples  
mit:
```

```
cd luma.examples/examples/
```

wechseln wir in den Ordner in dem die Beispiele liegen.

mit:

```
python3 demo.py
```

können wir eines der Beispiele starten. Sollte auf dem Display ein weißes rauschen zu sehen sein, muss der richtige controller übergeben werden. dies kann man mit:

```
python3 demo.py --device [controller]
```

luma.oled nimmt standartmäßig den SSD1306 her, hat man zu Beispiel SH1106 würde das starten des Skriptes wie folgt aussehen:

```
python3 demo.py --device ssh1106
```

Az-Delivery

In dem Ordner befinden sich weitere Beispiele.

Mit dem Befehl:

ls -a

kann man sie sehen.

Probieren Sie weitere Beispiele aus.

Mit:

sudo nano [Beispielscript]

lassen sich die Skripte editieren.



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

<https://az-delivery.de>

Viel Spaß!

Impressum

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