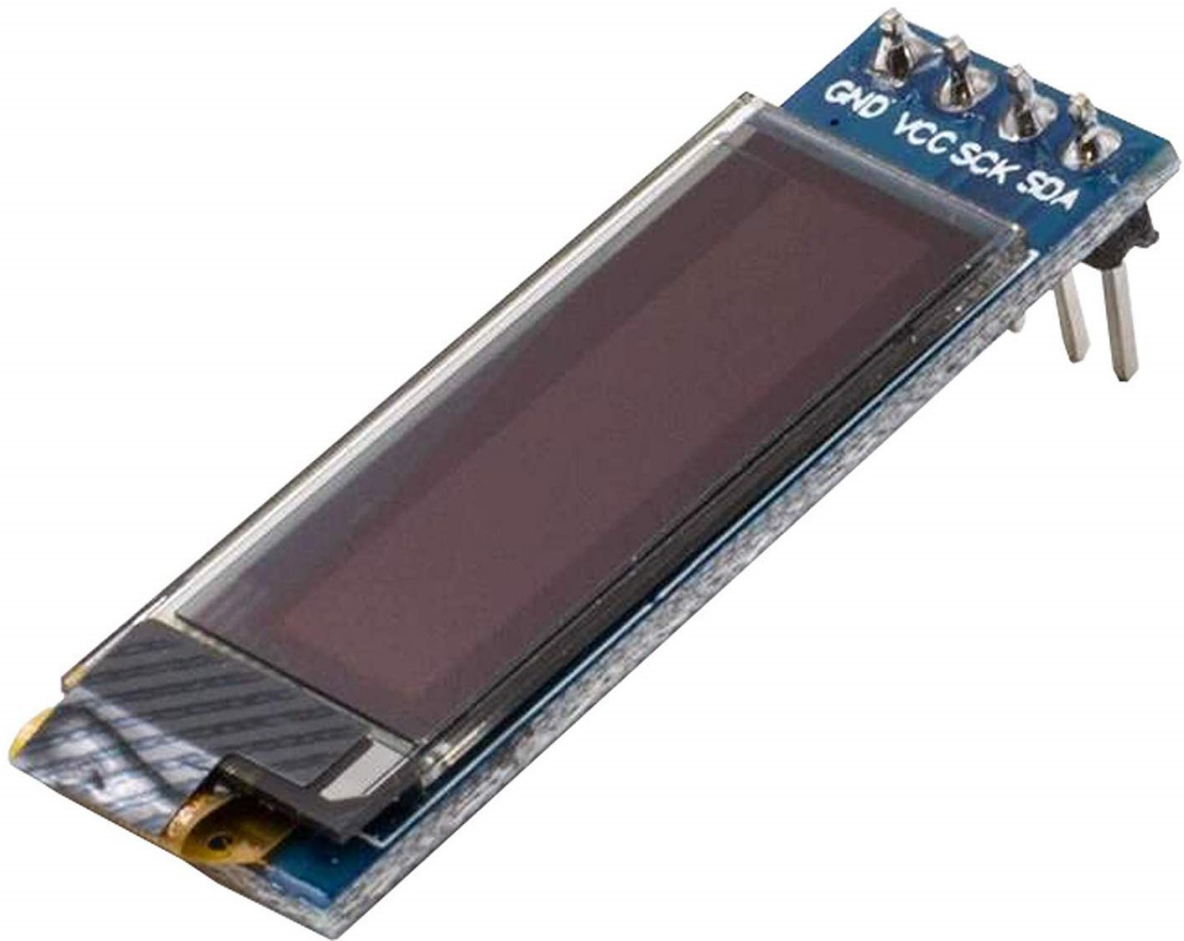


AZ-Delivery

Willkommen!

Vielen Dank, dass sie sich für unser AZ-Delivery 0.91" OLED I2C Display entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

Viel Spaß!



Az-Delivery

OLED – Eine organische Leuchtdiode ist eine Leuchtdiode (LED), bei der LED-Licht als Reaktion auf elektrischen Strom Licht ausstrahlt. Sie funktioniert ohne Hintergrundbeleuchtung, da sie nur sichtbares Licht ausstrahlt. Zum Beispiel auf PC-Monitoren, Fernsehbildschirmen, Mobiltelefonen usw. OLEDs sind in der Gemeinschaft weit verbreitet.

Technische Daten:

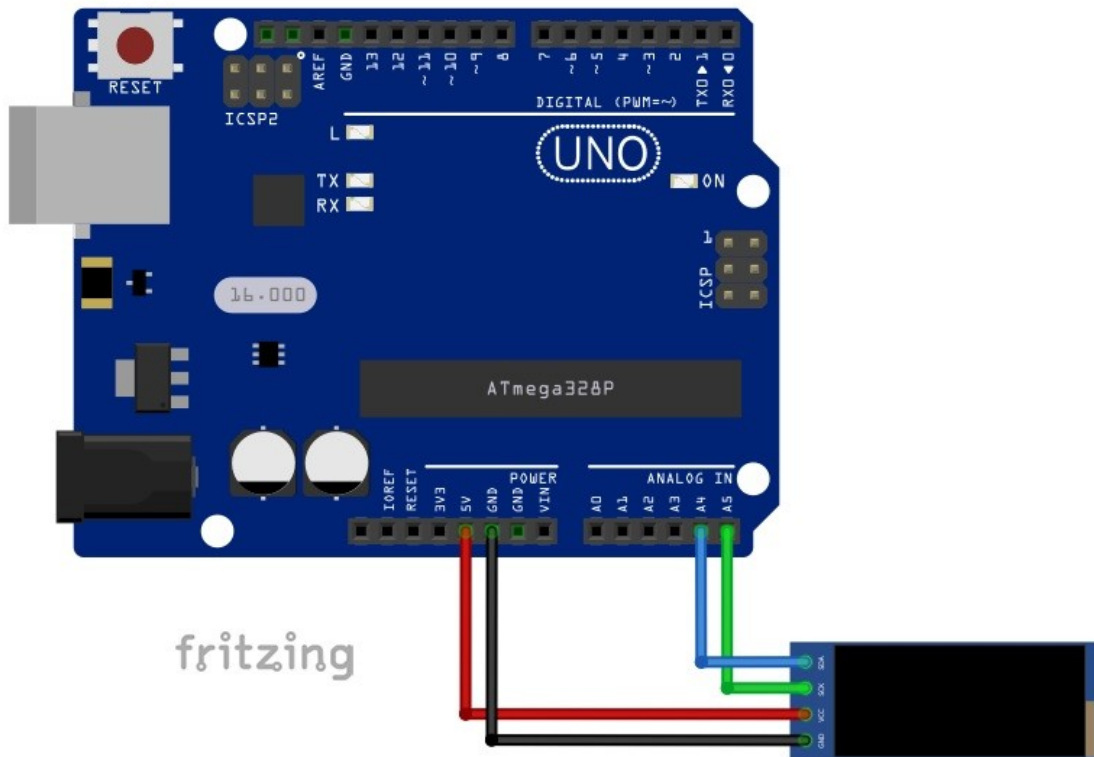
- » Versorgungsspannung: 3.3V - 5V
- » Kommunikationsschnittstelle: I2C
- » Pixelfarbe: White
- » Betriebstemperatur: -20°C ~ 70°C
- » Standart I2C-Adresse: 0x3C
- » Niedriger Stromverbrauch

I2C-Addressierung

Jedes Gerät, das wir an die I2C-Schnittstelle des Arduino anschließen, hat seine eigene Adresse. Dies ist wichtig, da der Arduino Uno als Master für die I2C-Schnittstelle fungiert und zur Kommunikation mit einem anderen Gerät die Adresse dieses Geräts erkennen muss. Wenn wir also mehrere OLED-Displays mit einem Arduino Uno verwenden müssen, empfiehlt es sich, ein I2C-Multiplexer-Gerät oder einfach ein größeres Display zu verwenden.

AZ-Delivery

Verbindung des Moduls mit dem Arduino Uno

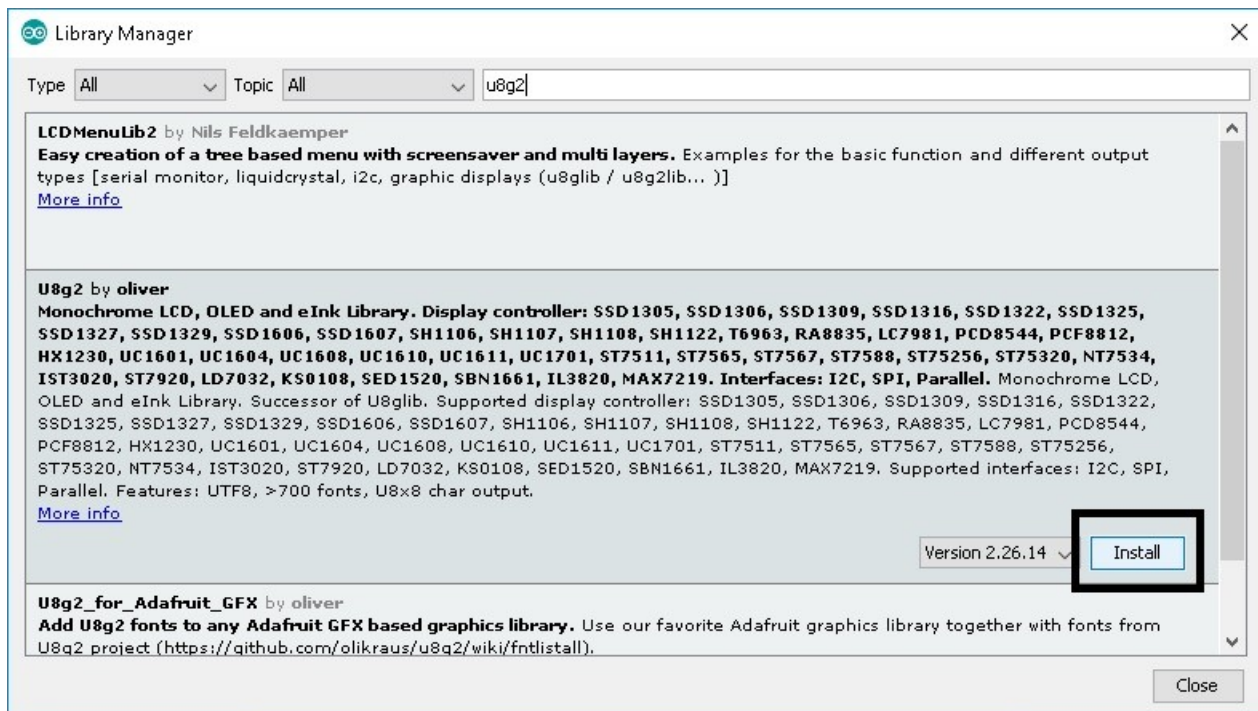


Module pin	>	Arduino pin	
SDA	>	A4	Blauer Draht
SCL	>	A5	Grüner Draht
VCC	>	5V	Roter Draht
GND	>	GND	Schwarzer Draht

Az-Delivery

Arduino IDE library

Wir werden die “*u8g2*”-Library benutzen. Um diese Library runterzuladen, öffnen Sie ihr Arduino IDE. Gehen Sie bis *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.



Diese Library enthält viele Sketch-Beispiele. Wir werden dieses Beispiel modifizieren: *File > Examples > U8g2 > full_buffer > GraphicsTest* und machen daraus einen weniger komplexen Sketch..

AZ-Delivery

Arduino code:

```
#include <Arduino.h>
#include <U8g2lib.h>
#include <Wire.h>
U8G2_SSD1306_128X32_UNIVISION_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 SIMBOL
    u8g2.drawUTF8(115, 15, "\xb0"); // DEGREE SYMBOL
}
```

AZ-Delivery

```
#define image_width 128
#define image_height 21
static const unsigned char image_bits[] U8X8_PROGMEM = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x06, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfc, 0x1f, 0x00, 0x00,
    0xfc, 0x1f, 0x00, 0x00, 0x06, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0xfe, 0x1f, 0x00, 0x00, 0xfc, 0x7f, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x07, 0x18, 0x00, 0x00, 0x0c, 0x60, 0x00, 0x00,
    0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x18, 0x00, 0x00,
    0x0c, 0xc0, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x03, 0x18, 0x00, 0x00, 0x0c, 0xc0, 0xf0, 0x1f, 0x06, 0x63, 0x80, 0xf1,
    0x1f, 0xfc, 0x33, 0xc0, 0x03, 0x18, 0x00, 0x00, 0x0c, 0xc0, 0xf8, 0x3f,
    0x06, 0x63, 0xc0, 0xf9, 0x3f, 0xfe, 0x33, 0xc0, 0x03, 0x18, 0x00, 0x00,
    0x0c, 0xc0, 0x18, 0x30, 0x06, 0x63, 0xc0, 0x18, 0x30, 0x06, 0x30, 0xc0,
    0xff, 0xff, 0xdf, 0xff, 0x0c, 0xc0, 0x18, 0x30, 0x06, 0x63, 0xe0, 0x18,
    0x30, 0x06, 0x30, 0xc0, 0xff, 0xff, 0xdf, 0xff, 0x0c, 0xc0, 0x98, 0x3f,
    0x06, 0x63, 0x60, 0x98, 0x3f, 0x06, 0x30, 0xc0, 0x03, 0x18, 0x0c, 0x00,
    0x0c, 0xc0, 0x98, 0x1f, 0x06, 0x63, 0x70, 0x98, 0x1f, 0x06, 0x30, 0xc0,
    0x03, 0x18, 0x06, 0x00, 0x0c, 0xc0, 0x18, 0x00, 0x06, 0x63, 0x38, 0x18,
    0x00, 0x06, 0x30, 0xc0, 0x03, 0x18, 0x03, 0x00, 0x0c, 0xe0, 0x18, 0x00,
    0x06, 0x63, 0x1c, 0x18, 0x00, 0x06, 0x30, 0xc0, 0x00, 0x80, 0x01, 0x00,
    0xfc, 0x7f, 0xf8, 0x07, 0x1e, 0xe3, 0x0f, 0xf8, 0x07, 0x06, 0xf0, 0xcf,
    0x00, 0xc0, 0x00, 0x00, 0xfc, 0x3f, 0xf0, 0x07, 0x1c, 0xe3, 0x07, 0xf0,
    0x07, 0x06, 0xe0, 0xcf, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x30, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0,
    0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0xe0, 0x00, 0xfc, 0x1f, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0x00, 0xfc, 0x1f, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f };
```

AZ-Delivery

```
void u8g2_bitmap() {
    u8g2.drawXBMP(0, 22, image_width, image_height, image_bits);
}
void setup(void) {
    u8g2.begin();
    u8g2_prepare();
}
float i = 0.0;
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);
}
```


AZ-Delivery

```
//one tab
    u8g2.clearBuffer();
    u8g2.prepare();
    u8g2_unicode();
    u8g2.sendBuffer();
    delay(1500);

    u8g2.clearBuffer();
    u8g2_bitmap();
    u8g2.sendBuffer();

    u8g2.clearBuffer();
    u8g2.setCursor(0, 0);
    u8g2.print(i);
    i = i + 1.5;

    u8g2.sendBuffer();
}
```

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 Funktion hat drei Argumente: Die X-Position und die 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()*** Funktionen. Allerdings akzeptieren ***drawRFrame()*** and ***drawRBox()*** Funktionen ein fünftes Argument als Radius-Argument.

Az-Delivery

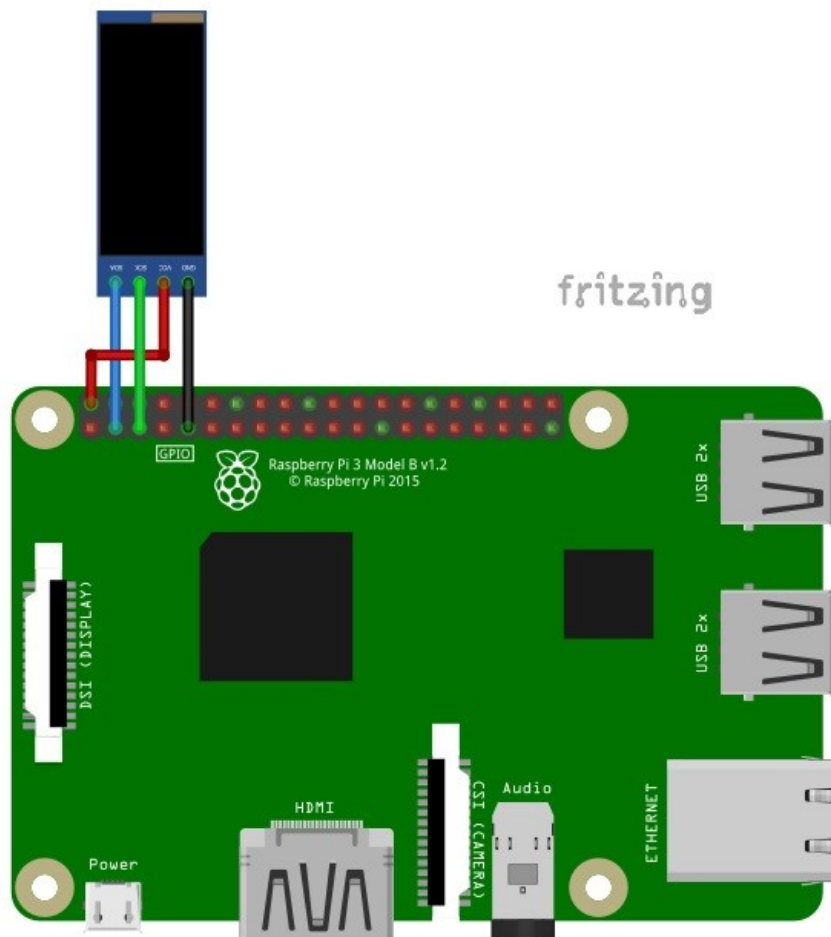
» **`drawUTF8()`** wird zur Anzeige von "Unicode"-Zeichen verwendet. Seine Argumente sind die X- und Y-Positionen des Zeichens sowie das Zeichen-Argument für das Zeichen selbst. Um diese Zeichen anzuzeigen, können Sie entweder:

- » 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.

Az-Delivery

Verbindung des Displays mit dem Raspberry Pi

In diesem Kapitel wird näher erklärt, wie der 0,91-Zoll-OLED-Bildschirm, über ein I2C-Interface mit Raspberry Pi verbunden, korrekt verwendet wird.



Display pin > Raspberry Pi pin

SDA > GPIO2 [pin 3]

SCL > GPIO3 [pin 5]

VCC > 3.3V [pin 1]

GND > GND [pin 9]

Blauer Draht

Grüner Draht

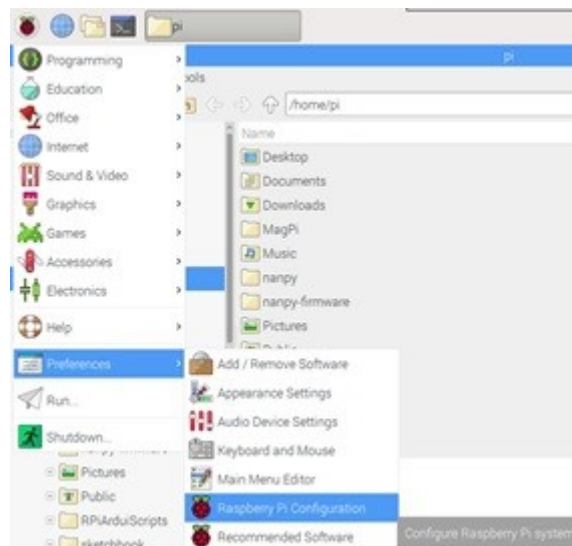
Roter Draht

Schwarzer Draht

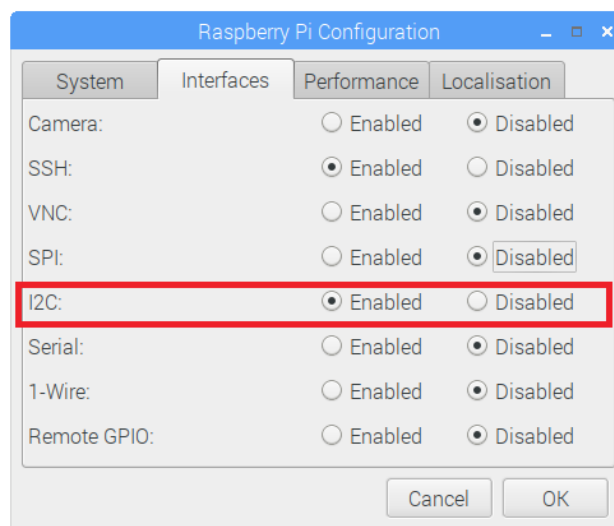
AZ-Delivery

Aktivierung der I2C-Schnittstelle auf Raspbian

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

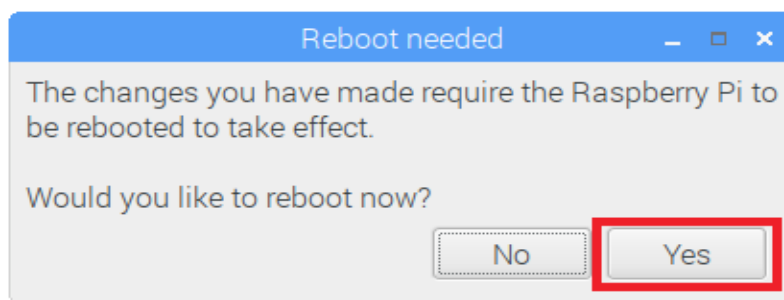


Als nächstes öffnen Sie den *“Interfaces”* Tab, finden Sie die *“I2C”* Taste and und schalten diese an, wie unten abgebildet:



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.

Az-Delivery

Python-library

Für die Anzeige von Formen, Text und Bildern werden wir die Adafruit-Python-library verwenden. Um sie zu installieren, müssen wir sicher sein, dass wir GIT-Tools auf unserem Rechner installiert haben. Wir tun dies, indem wir diesen Befehl im Terminal ausführen:

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

Um das "library repository" zu klonen:

```
git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git
```

Ändern Sie das Verzeichnis in das heruntergeladene Verzeichnis:

```
cd Adafruit_Python_SSD1306
```

Um die library zu installieren, führen Sie diesen Befehl aus:

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

Az-Delivery

Python-Skript

Jetzt können wir einige Beispiele testen. Gehen Sie zum Beispiel-Verzeichnis:

cd examples

und um Beispiele für Python-Skripte zu sehen, führen Sie diesen Befehl aus:

ls

Ihnen sollten diese Python-Skripten:

- » *animate.py*
- » *shapes.py*
- » *buttons.py*
- » *image.py*, etc.

Um eines davon zu verwenden, führen Sie diesen Befehl aus:

python3 shapes.py

AZ-Delivery

Jetzt gehen wir den Code durch:

```
import time
import Adafruit_SSD1306
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

disp = Adafruit_SSD1306.SSD1306_128_64()

# It might be worth mentioning that you can change the I2C
address # by passing an i2c_address parameter like:
# disp=Adafruit_SSD1306.SSD1306_128_64(rst=RST,i2c_address=0x3C)

disp.begin() # Initializes the library
disp.clear() # Clears the display
disp.display()

# Creates a blank image for drawing.
# Make sure to create image with mode '1' for 1-bit color.

width = disp.width
height = disp.height
image = Image.new('1', (width, height))
# Get drawing object to draw on image.

draw = ImageDraw.Draw(image)

# Draw a Schwarzer filled box to clear the image.
draw.rectangle((0,0,width,height), outline=0, fill=0)
```

Az-Delivery

```
# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 2
shape_width = 20
top = padding
bottom = height-padding

# Move left to right keeping track of
# the current x position for drawing shapes.
x = padding
# Draw a rectangle
draw.rectangle((x, top, x+shape_width, bottom), outline=1,
fill=0)
x += shape_width+padding
# Draw an ellipse
draw.ellipse((x, top, x+shape_width, bottom), outline=1, fill=0)
x += shape_width+padding
# Draw an X
draw.line((x, bottom, x+shape_width, top), fill=1)
draw.line((x, top, x+shape_width, bottom), fill=1)
x += shape_width+padding

# Draw a triangle
draw.polygon([(x, bottom), (x+shape_width/2, top),
(x+shape_width, bottom)], outline=1, fill=0)
x += shape_width+padding

# Display image
disp.image(image)
disp.display()
```

AZ-Delivery

```
# Load default font
font = ImageFont.load_default()
# Alternatively you could load a TTF font. But we won't be
# covering # that in this instructable.
# Writing two lines of text here
draw.text((x, top), 'AZ', font=font, fill=1)
draw.text((x, top+20), 'DLVRY', font=font, fill=1)
# Display image
disp.image(image)
disp.display()
```

Az-Delivery

Der Code selbst ist dem Code recht ähnlich, der bei der Arbeit mit dem Arduino Uno behandelt worden ist. Beginnen Sie mit dem Importieren der notwendigen libraries, damit der Code funktioniert. Wählen Sie als nächstes das Display, mit der wir arbeiten. Initialisieren Sie dann die Bibliothek und leeren Sie dann die Anzeige, so dass ein leeres Bild zum Zeichnen oder Schreiben zur Verfügung steht. Erläuterungen zu den verwendeten library-Funktionen sind:

- » Abgesehen von den Hauptargumenten akzeptieren Funktionen zwei zusätzliche Argumente, die `outline` und `fill` genannt werden. `outline`- und `fill`-Werte von Argumenten können 0 oder 1 sein. Das bedeutet, dass bei 0 – das Objekt transparent und bei 1 – das Objekt sichtbar sein wird. Das Argument `outline` bezieht sich auf die Objektkante und das Argument `fill` auf die Innenseiten des Objekts.
- » **`draw.rectangle()`** akzeptiert drei Argumente, das erste ist eine Sammlung mit vier Elementen (X- und Y-Position der linken oberen Ecke des Rechtecks, Breite und Höhe), das zweite ist `outline` und das dritte ist `fill`.
- » **`draw.ellipse()`** wird verwendet, um eine Ellipse anzuzeigen. Es hat drei Argumente: das erste ist eine Liste mit Elementen: X- und Y-Position des Mittelpunktes der Ellipse, Breite und Höhe der Ellipse; das zweite ist das `outline` und das dritte ist `fill`.
- » **`draw.line()`** wird verwendet, um eine Linie anzuzeigen. es hat zwei Argumente: das erste ist ein Set von zwei Listen: die erste Liste ist für die X- und Y-Position des ersten Punktes der Linie, und das zweite Argument ist `fill`. Die Linie hat keine Fläche, daher gibt es kein `outline`-Argument für diese Funktion.

Az-Delivery

» **`draw.polygon()`** wird verwendet, um ein Polygon anzuzeigen. Es akzeptiert drei Argumente: das erste ist ein Set von Listen: jede Liste hat zwei Werte, die X- und Y-Positionen des Punktes, der zur Anzeige des Polygons verwendet wird; das zweite Argument ist das `outline`-Argument und das dritte ist das `fill`-Argument. Die Punkte werden nach der Reihenfolge im Set verbunden, und der letzte Punkt ist mit dem ersten verbunden. In unserem Beispiel verwenden wir diese Funktion, um ein Dreieck anzuzeigen, also haben wir drei Listen, drei Punkte.

» **`ImageFont.load_default()`** wird zur Einstellung der Schriftart für die von uns verwendeten Zeichen verwendet. Alternativ könnte stattdessen auch eine andere TTF-Schriftart geladen werden, aber das werden wir in dieser Anleitung aufgrund von Lizenzproblemen, die an die Verwendung von benutzerdefinierten Schriftarten gebunden sind, nicht behandeln.

» **`draw.text()`**, wird verwendet, um einen Text anzuzeigen. Es verwendet vier Argumente: das erste ist eine Liste mit zwei Werten, die X- und Y-Position des Cursors, das zweite ist die Zeichenkette, die den Text enthält, den wir anzeigen wollen, das dritte ist die Schriftart und das vierte ist das `fill`-Argument.

Jetzt sollten Sie einen Unterordner namens "*examples*" in Ihrem `library`-Ordner haben. Er sollte einige Beispiele für den Code enthalten, die Sie zum weiteren Ausprobieren verwenden können.

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

AZ-Delivery

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 Arduino 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>