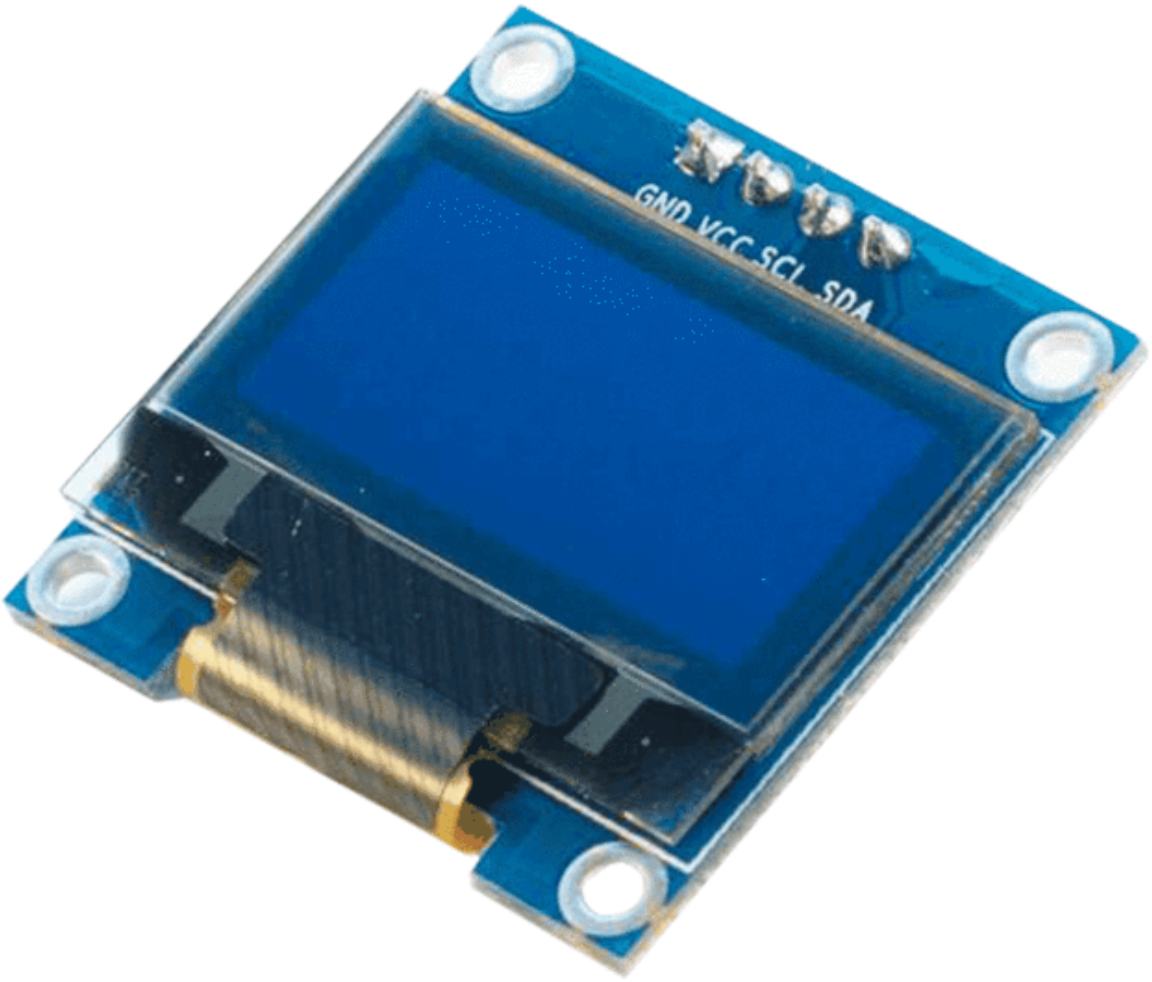
Willkommen!

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!



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 Versorgungs-spannung 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.

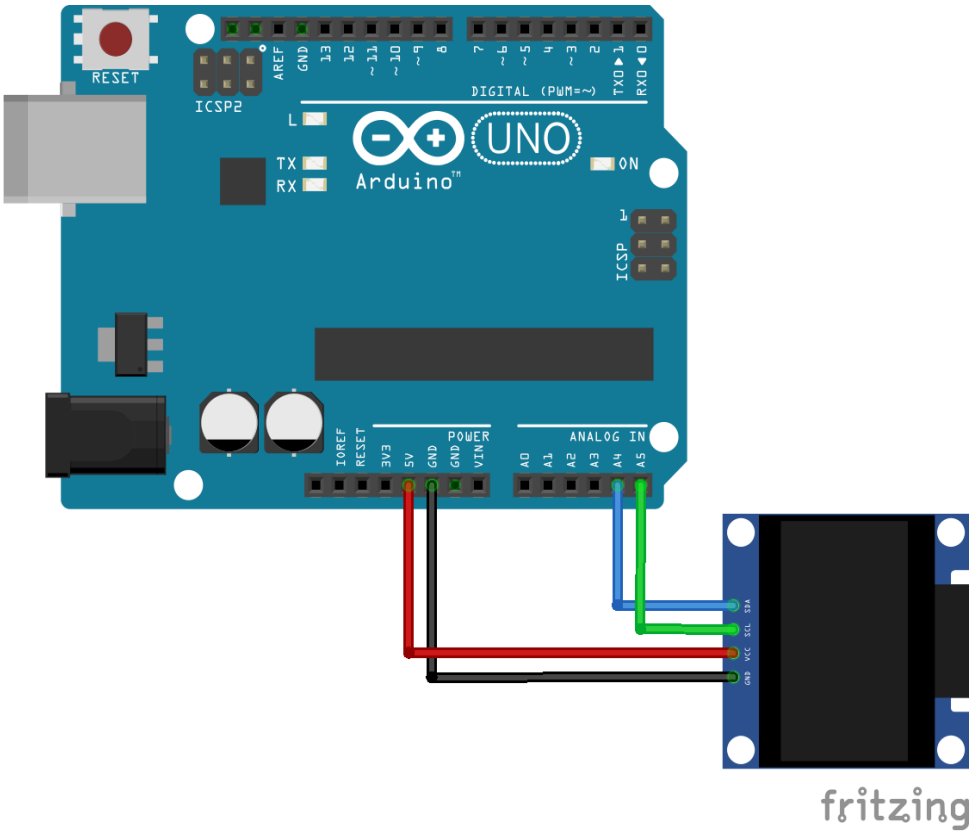
Die Leitungen sind high, wenn sie nicht heruntergezogen sind, was auf Pull-up-Widerstände zurückzuführen ist. Der I2C erfordert Pull-up-Widerstände auf beiden Seiten mit einem Widerstand zwischen 1kΩ und 10kΩ, wobei der am häufigsten verwendete Widerstand 4,7kΩ ist. Der Vorteil der Verwendung dieses OLEDs mit Arduino besteht darin, dass Arduino 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 Arduino mit dem SDA-Pin des Displays, sowie den SCL-Pin (A5) des Arduino mit dem SCL-Pin des Displays. Natürlich ist die Programmierung des Arduino 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 Arduino-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 Arduino Uno verwenden müssen, empfiehlt es sich, ein I2C-Multiplexer-Gerät oder einfach ein größeres Display zu verwenden.

Verbindung des Displays mit dem Arduino Uno

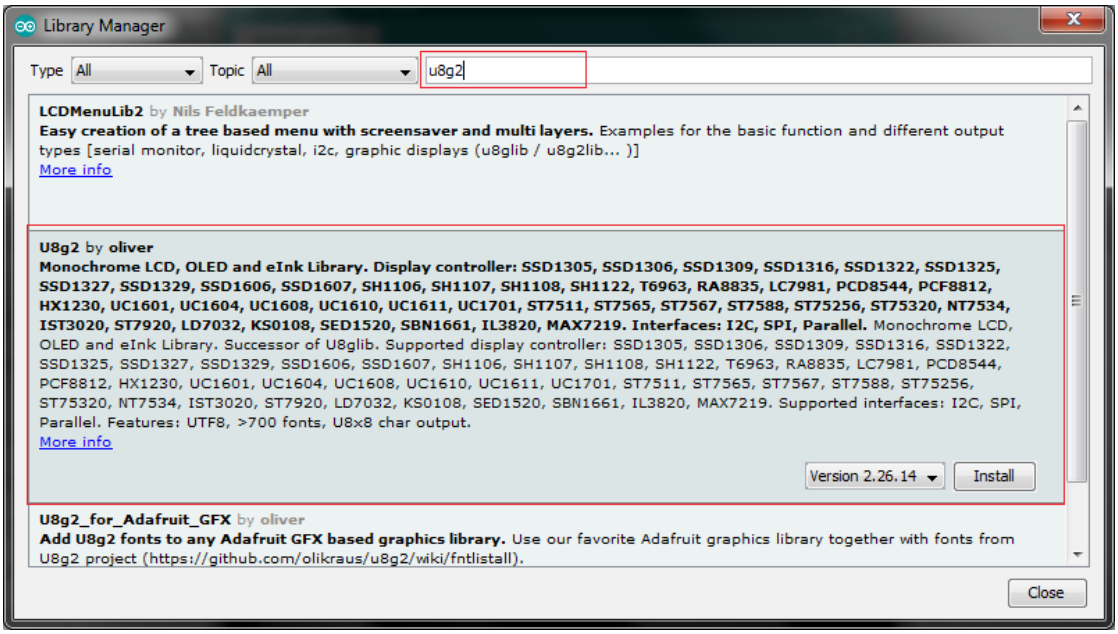
Verbinden Sie alles, wie unten abgebildet:



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

Arduino 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.



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.

Arduino-Code:

```
#include <U8g2lib.h>
#include <Draht.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);
}
```

```
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
}



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);
}
```

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 detailiertere 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.

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- undY -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.

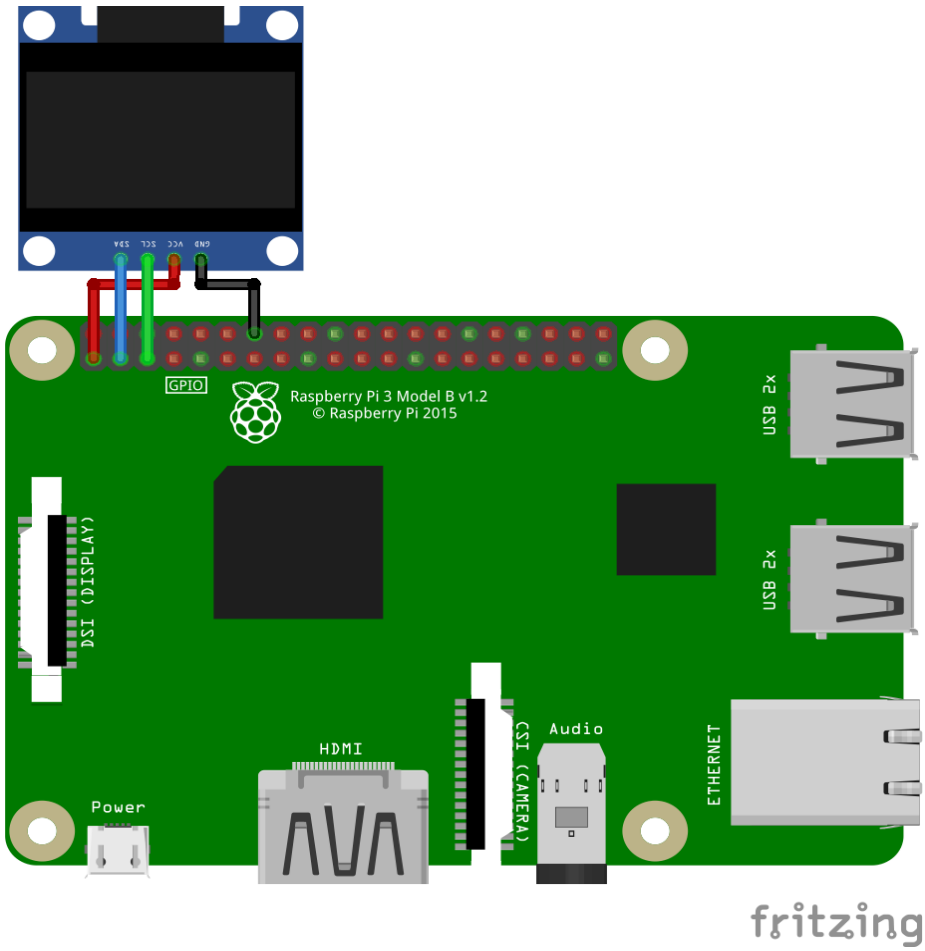
» Außerdem gibt es Funktionen zur Anzeige von Boxen und Rahmen mit einem festgelegten Eckenradius. ***DrawRFrame()*** und ***drawRBox()*** Funktionen sind so ähnlich wie ***drawFrame()*** und ***drawBox()*** Funktionen. Allerdings akzeptieren ***drawRFrame()*** and ***drawRBox()*** Funktionen ein fünftes Argument als Radius-Argument.

» ***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

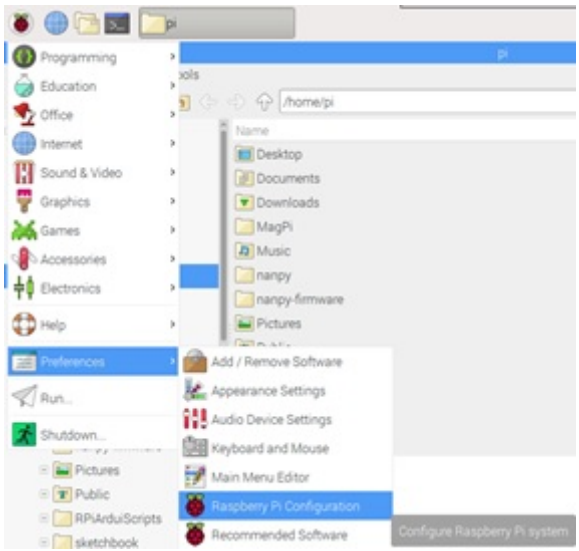
Als Erstes verbinden Sie alles, wie unten abgebildet:



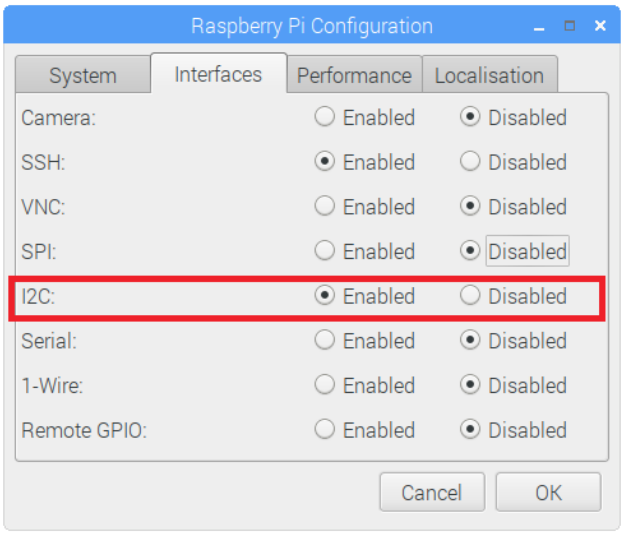
OLED Display pin	**>**	**Raspberry pin**		
VCC	>	3,3V	[pin 1]	**Roter Draht**
GND	>	GND	[pin 14]	**Schwarzer Draht**
SCL	>	GPIO3	[pin 5]	**Grüner Draht**
SDA	>	GPIO2	[pin 3]	**Blauer Draht**

Aktivierung der I2C-Schnittstelle auf Raspbian

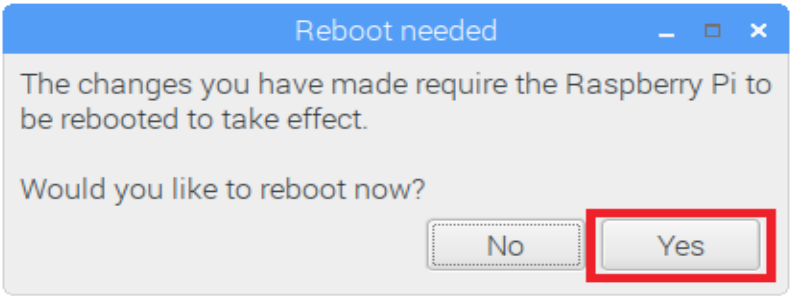
Dafür gehen Sie zu *Preferences* > *Raspberry Pi Configuration* wie unten abgebildet:



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



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



Libraries und Tools für Python

Sie werden *“python-smbus”* und *“i2c-tools”* installieren müssen, so dass das Interface innerhalb der Programmiersprache “Python” verwendet werden kann.

Dafür öffnen Sie die Terminal App und geben Folgendes ein:

```
sudo apt-get update
sudo apt-get install -y python-smbus i2c-tools
```

Danach schalten Sie das Gerät aus, indem Sie Folgendes eingeben:

```
sudo halt
```

Nach einem kurzen Augenblick, können Sie Ihren Pi von der Stromversorgung trennen und ihre I2C-Hardware anschließen.

Sobald Ihre Hardware angeschlossen ist, empfehlen wir Ihnen, Ihre Pins noch einmal zu überprüfen. Anschließend können Sie Ihren Pi einschalten und warten, bis er hochgefahren ist.

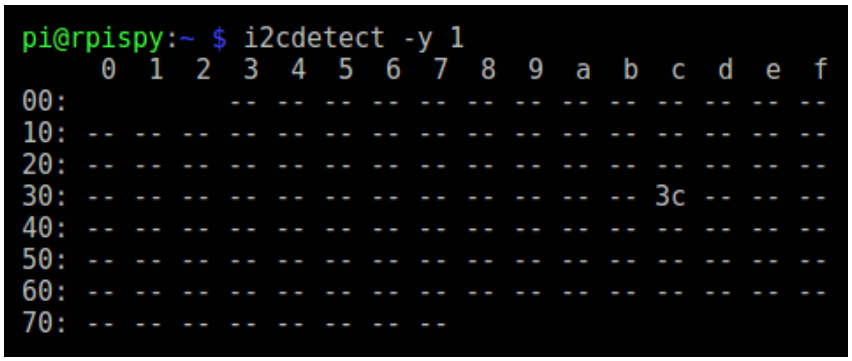
Als nächstes werden Sie einige I2C-Libraries installieren, und obwohl Sie vielleicht schon einige davon installiert haben, empfehlen wir Ihnen, diese Befehle sicherheitshalber trotzdem auszuführen:

```
sudo apt install -y python3-dev
sudo apt install -y python-imaging python-smbus i2c-tools
sudo apt install -y python3-pil
sudo apt install -y python3-pip
sudo apt install -y python3-setuptools
sudo apt install -y python3-rpi.gpio
```

Um die Adresse Ihres OLED-Displays auf dem I2C-Bus zu finden, können Sie diesen Befehl eingeben:

```
i2cdetect -y 1
```

Die Resultate sollten wie folgt aussehen:



Was wir auf dem Bild sehen, ist die Adresse eines Geräts, welches wir mit dem Befehl gefunden haben; in diesem Fall lautet die Adresse "3c".

Als nächstes können wir mit der Installation der OLED-Python-Library fortfahren, wir werden die *“ Adafruit_Python_SSD1306”* verwenden. Sie erlaubt es uns, Formen, Bilder und Text anzuzeigen.

Wir fangen mit Folgendem an:

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

Um sicherzugehen, dass *“git”* installiert ist.

Als Nächstes klonen wir das “depository”:

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

Danach gehen Sie zum Library-Verzeichnis:

```
cd Adafruit_Python_SSD1306
```

Danach geben Sie Folgendes ein:

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

Auf diese Weise können Sie die Library in Ihre Python-Skripte einbinden.

Nun können Sie testweise einige Beispiel-Skripte ausführen.

Gehen Sie zum Verzeichnis "examples":

```
cd examples
```

Innerhalb des “despository”, finden sich viele Skripts:

» shapes.py

» image.py

» stats.py

Und viele mehr, aber wir werden auf Formen beschränken.

Um das Beispiel auszuführen, geben Sie Folgendes ein:

```
python3 shapes.py
```

Als Nächstes 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)
# 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()
# 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()
```

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.

» ***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 bei benutzerdefinierten Schriftarten 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.

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