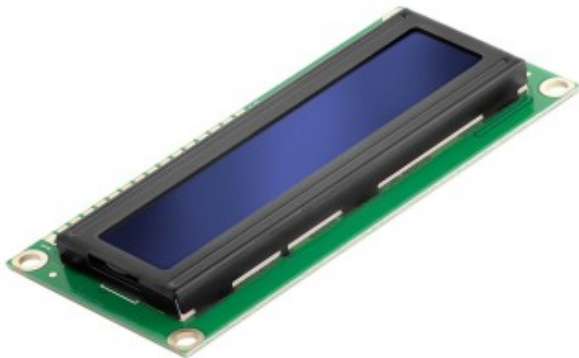


# AZ-Delivery

## Willkommen!

Vielen Dank, dass sie sich für unseren *LCD-Bildschirm von AZ-Delivery* entschieden haben. Diese gibt es einzeln oder mit I2C Converter im Bundle in grün oder blau, sowie in zwei größen: 16x02 und 20x04. In den folgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

**Viel Spaß!**



## Inhaltsverzeichnis

Einführung.....	3
Technische Daten.....	6
Pinbelegung.....	8
Wie man die Arduino IDE einrichtet.....	10
Wie man den Raspberry Pi und Python einrichtet.....	15
Verbindung des Moduls mit dem Microcontroller.....	16
Sketch-Beispiel.....	18
Verbindung des Moduls mit dem Microcontroller über I2C-Adapter.....	19
Library für Arduino IDE.....	20
Sketch-Beispiel.....	21
Verbindung des Bildschirms mit dem Raspberry Pi.....	24
Python-Skript.....	26
Verbindung des Bildschirms mit dem Raspberry Pi über I2C-Adapter.....	34
Libraries und Tools für Python.....	36
Python-Skript.....	38

# Az-Delivery

## Einführung

Eine Flüssigkristallanzeige, auch LCD genannt, ist ein Gerät, das Flüssigkristalle verwendet, um visuelle Zeichen anzuzeigen.

Flüssigkristalle emittieren selber kein Licht. Die LCD-Bildschirme verwenden eine Hintergrundbeleuchtung und Flüssigkristalle, um Licht zu blockieren. Wenn kein Strom durch die Flüssigkristalle fließt, befinden sie sich in einem chaotischen Zustand. Licht von der Hintergrundbeleuchtung geht mühelos durch Flüssigkristalle hindurch. Wenn Strom durch Flüssigkristalle fließt, ordnen sie sich in einem uniformen Zustand an. Dadurch entsteht ein Schatten auf dem Bildschirm, der Objekte erzeugt.

Einfach ausgedrückt, LCD verwenden Flüssigkristalle, um das von der Hintergrundbeleuchtung kommende Licht durchzulassen oder zu blockieren. Auf diese Weise werden Pixel erzeugt. Sie werden kombiniert, um jegliche sichtbaren 2D-Objekte auf dem Bildschirm anzuzeigen. Jede Pixelfläche kann EIN- und AUSgeschaltet werden, indem Elektrizität über einen On-Board-Controller-Chip zugeführt wird.

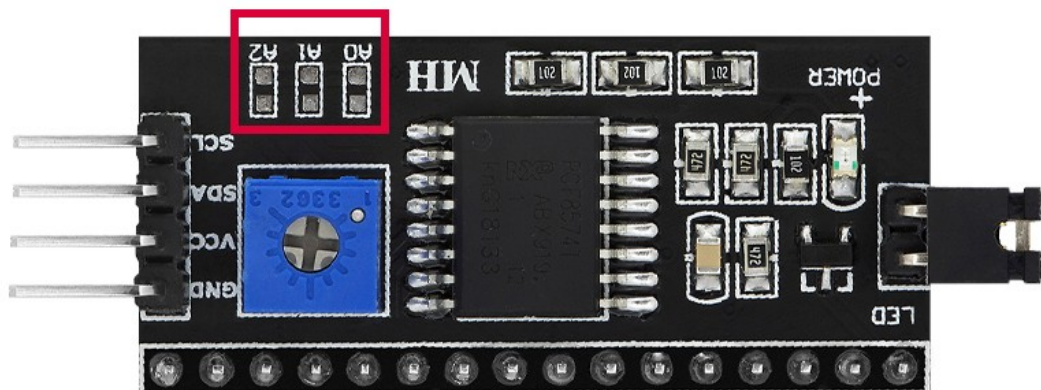
Jedes Segment des Displays mit Flüssigkristallen stellt ein Pixel dar, und muss separat gesteuert werden. Aus diesem Grund wird eine spezielle integrierte Schaltung, ein so genannter Treiberchip, benötigt. Das 16x02(20x04) LCD hat einen Treiberchip namens "HD44780". Um den Bildschirm zu steuern, muss der Mikrocontroller mit dem Treiberchip kommunizieren. Der Treiberchip verwendet eine Art SPI-Schnittstelle, um mit einem Mikrocontroller zu kommunizieren.

## Der I2C-Adapter

Der I2C-Adapter ist ein Gerät, das die Verbindung zwischen LCD-Bildschirmen und Mikrocontrollern vereinfacht. Er verwendet die I2C-Schnittstelle zur Kommunikation mit dem Mikrocontroller. Mehrere Adapter können an dieselbe I2C-Schnittstelle angeschlossen werden. Der Adapter ist mit LCD-Bildschirmen, mit integrierten HD44780-Chips, kompatibel. Der I2C-Adapter ist sowohl mit 16x02 LCD-Bildschirmen als auch mit 20x04 LCD-Bildschirmen kompatibel, die AZ-Delivery anbietet.

Der I2C-Adapter wird mit einer vordefinierten I2C-Adresse geliefert, welche 0x27 ist. Sie kann aber durch das Anlöten der mit A0, A1 und A2 beschrifteten Pads auf dem Adapter geändert werden.

### SOLDERING PADS (A0-A1)



# Az-Delivery

Wie eine bestimmte I2C-Adresse des Adapters eingestellt wird, wird in der folgenden Tabelle dargestellt:

PADS			I2C ADDRESS
A2	A1	A0	
C	C	C	0x20
C	C	O	0x21
O	C	O	0x22
C	O	O	0x23
O	C	C	0x24
O	C	O	0x25
O	O	C	0x26
O	O	O	0x27
<b>O - Open</b>			<b>C - Closed</b>

## Technische Daten

» Betriebsspannungsbereich:	3.3V bis 5V
» Anzeigebereich:	12 x 56mm
» LCD-Typ:	STN, positiv, transflektiv, grün/blau
» Hintergrundbeleuchtung:	ED, weiß
» Blickwinkel:	180°
» Modi:	parallel (8-bit und 4-bit)
» Betriebstemperatur:	-10 °C bis 60 °C
» Dimensionen:	36 x 80 x 12.5mm [1.4 x 3.1 x 0.5in]
» Interface:	I2C/parallel
» I2C-Adresse:	0x20 – 0x27
» Einstellen des Kontrasts :	Potentiometer
» Einstellen der Hintergrundbel.:	Jumper

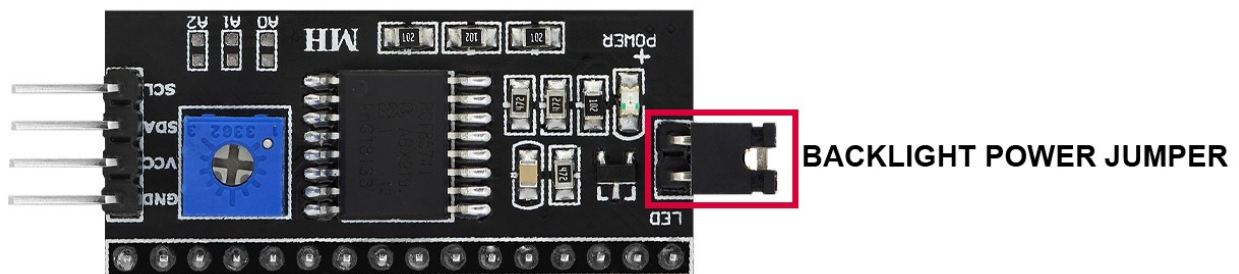
Die Spannung für die Hintergrundbeleuchtung beträgt 5V DC. Für Logikfunktionen beträgt der Stromverbrauch im Betrieb 1,5mA und für die Hintergrundbeleuchtung 30mA.

Die Auflösung des Bildschirms beträgt 16x02, was bedeutet, dass Zeichen in 2 Zeilen mit 16 Zeichen pro Zeile dargestellt werden können. Jedes Zeichen besteht aus 5x7 Pixeln.

Zur Einstellung des Kontrasts des Displays besitzt der I2C-Adapter einen eingebauten Potentiometer. Zur Einstellung wird daher ein kleiner Schraubendreher benötigt.

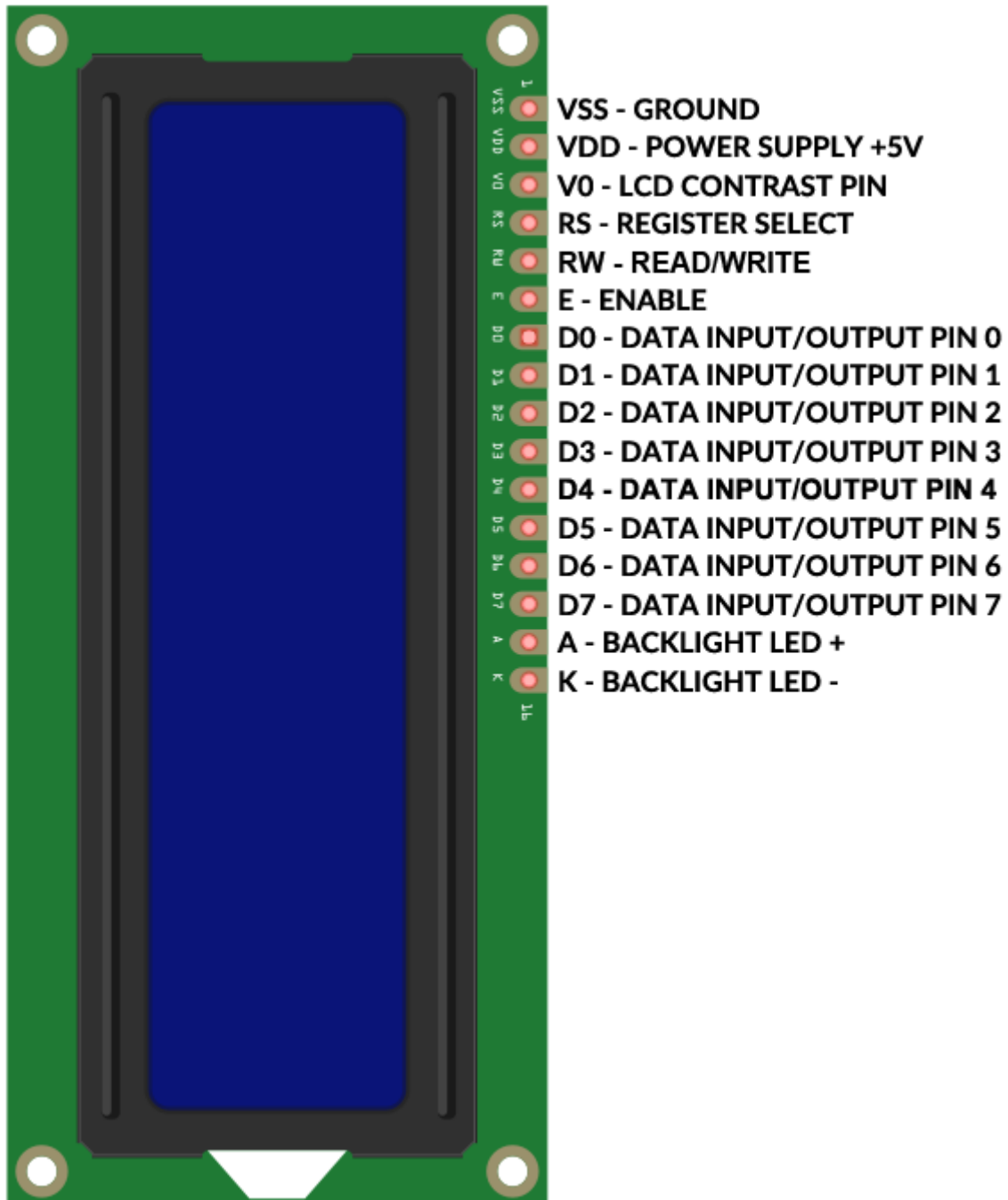
# Az-Delivery

Zur Steuerung der Hintergrundbeleuchtung des LCD-Bildschirms verfügt der I2C-Adapter über eine Leistungsbrücke für die Hintergrundbeleuchtung. Der Jumper wird zum Ein-/Ausschalten der Hintergrundbeleuchtung verwendet. Wenn der Jumper angeschlossen wird, wird damit die Stromversorgung für die Hintergrundbeleuchtung angeschlossen. Wenn der Jumper abgesteckt wird, wird die Stromversorgung der Hintergrundbeleuchtung unterbrochen.



## Pinbelegung ohne I2C-Adapter

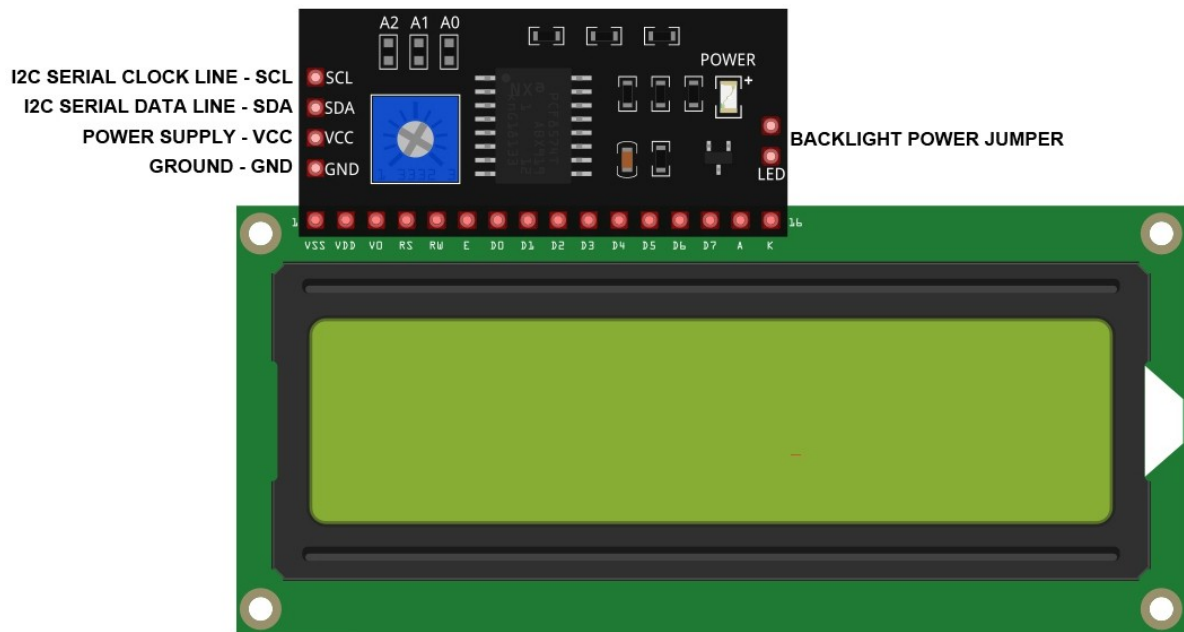
Der 16x02 LCD hat 16 Pins. Die Pinbelegung ist wie folgt (gilt auch für 20x04):





## Pinbelegung mit I2C-Adapter

Der 16x02 LCD hat 16 Pins und der I2C-Adapter 20. Der Adapter wird wie folgt an den LCD-Bildschirm angeschlossen(dies gilt auch für den 20x04):



**Hinweis:** Es ist notwendig, den I2C-Adapter und die LCD-Anzeige genau, wie oben abgebildet, anzuschließen. Wenn anders angeschlossen wird, können die Geräte beschädigt werden.

# AZ-Delivery

**Hinweis für den Raspberry Pi:** Die Spannung des TTL-Logikpegels der E/A-Pins beträgt 5V. Um den LCD-Bildschirm und den I2C-Adapter mit dem Raspberry Pi zu verwenden, muss ein Logic Level Converter verwendet werden. Andernfalls kann das Einführen des Signals über die E/A-Pins des Moduls zu den GPIO-Pins des Raspberry Pi zu Schäden führen. Verwenden Sie daher den von AZ-Delivery angebotenen [TXS0108E 8ch Logic Level Converter](#).

# Az-Delivery

## Wie man die Arduino-IDE einrichtet

Falls die Arduino-IDE nicht installiert ist, folgen Sie dem [link](#) und laden Sie die Installationsdatei für das Betriebssystem Ihrer Wahl herunter.

Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left, there is a teal circular logo with a white infinity symbol containing a minus sign on the left and a plus sign on the right. To the right of the logo, the text reads: **ARDUINO 1.8.9**. Below this, it says: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions." On the right side of the page, there are several download options: "Windows Installer, for Windows XP and up" and "Windows ZIP file for non admin install"; "Windows app Requires Win 8.1 or 10" with a "Get" button; "Mac OS X 10.8 Mountain Lion or newer"; "Linux 32 bits"; "Linux 64 bits"; "Linux ARM 32 bits"; and "Linux ARM 64 bits". At the bottom right, there are links for "Release Notes", "Source Code", and "Checksums (sha512)".

Für Windows Benutzer: Doppelklicken Sie auf die heruntergeladene `.exe`-Datei und folgen Sie den Anweisungen im Installationsfenster.

# Az-Delivery

Für *Linux* Benutzer, laden Sie eine Datei mit der Erweiterung *.tar.xz* herunter, die extrahiert werden muss. Wenn sie extrahiert ist, gehen Sie in das extrahierte Verzeichnis und öffnen Sie das Terminal in diesem Verzeichnis. Zwei *.sh* Skripte müssen ausgeführt werden, das erste namens *arduino-linux-setup.sh* und das zweite heißt *install.sh*.

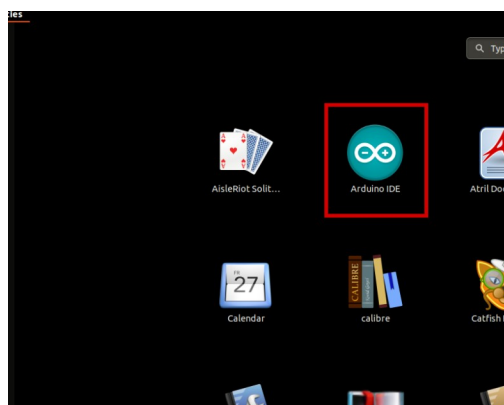
Um das erste Skript im Terminal auszuführen, öffnen Sie das Terminal im extrahierten Ordner und führen Sie den folgenden Befehl aus:

```
sh arduino-linux-setup.sh user_name
```

**user\_name** - ist der Name eines Superusers im Linux-Betriebssystem. Ein Passwort für den Superuser muss beim Start des Befehls eingegeben werden. Warten Sie einige Minuten, bis das Skript vollständig abgeschlossen ist.

Das zweite Skript mit der Bezeichnung *install.sh*-Skript muss nach der Installation des ersten Skripts verwendet werden. Führen Sie den folgenden Befehl im Terminal (extrahiertes Verzeichnis) aus: **sh install.sh**

Nach der Installation dieser Skripte gehen Sie zu *All Apps*, wo die *Arduino-IDE* installiert ist.



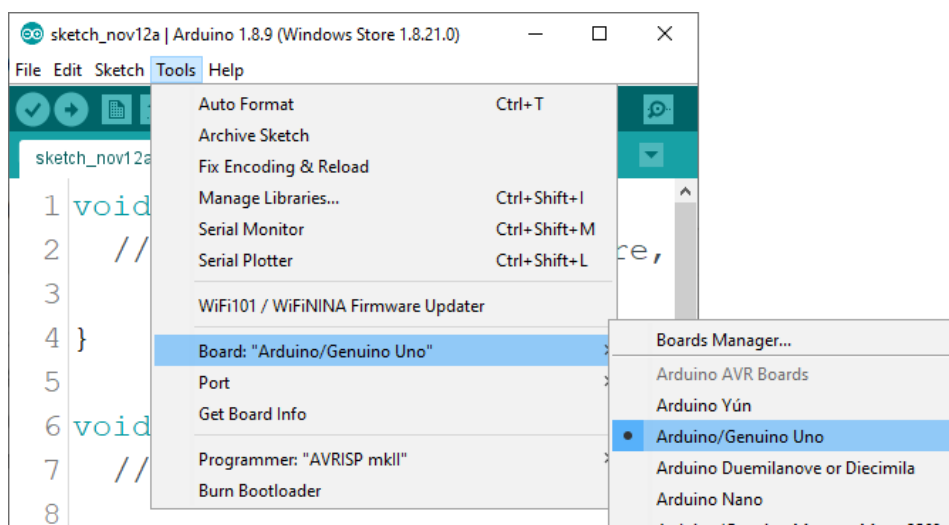
# Az-Delivery

Fast alle Betriebssysteme werden mit einem vorinstallierten Texteditor ausgeliefert (z.B. *Windows* mit *Notepad*, *Linux Ubuntu* mit *Gedit*, *Linux Raspbian* mit *Leafpad* usw.). Alle diese Texteditoren sind für den Zweck des eBooks vollkommen in Ordnung.

Zunächst ist zu prüfen, ob Ihr PC ein Arduino-Board erkennen kann. Öffnen Sie die frisch installierte Arduino-IDE, und gehen Sie zu:

*Tools > Board > {your board name here}*

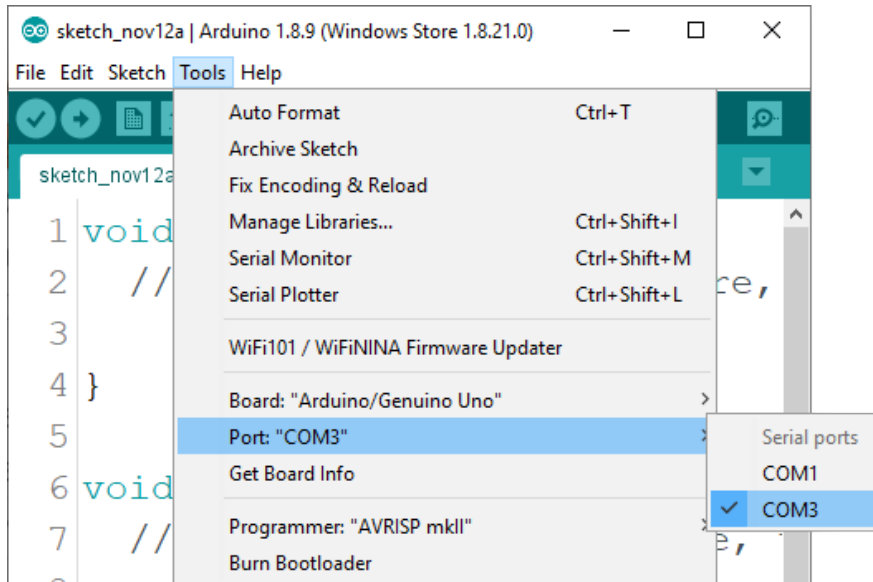
*{your board name here}* sollte der *Arduino/Genuino Uno* sein, wie es auf dem folgenden Bild zu sehen kann:



Der Port, an dem das Microcontroller-Board angeschlossen ist, muss ausgewählt werden. Gehe zu: *Tools > Port > {port name goes here}* und wenn das Microcontroller-Board an den USB-Port angeschlossen ist, ist der Portname im Drop-down Menü auf dem vorherigen Bild zu sehen.

# Az-Delivery

Wenn die Arduino-IDE unter Windows verwendet wird, lauten die Portnamen wie folgt:



Für *Linux* Benutzer, ist zum Beispiel der Portname `/dev/ttyUSBx`, wobei `x` für eine ganze Zahl zwischen `0` und `9` steht.

## Wie man den Raspberry Pi und Python einrichtet

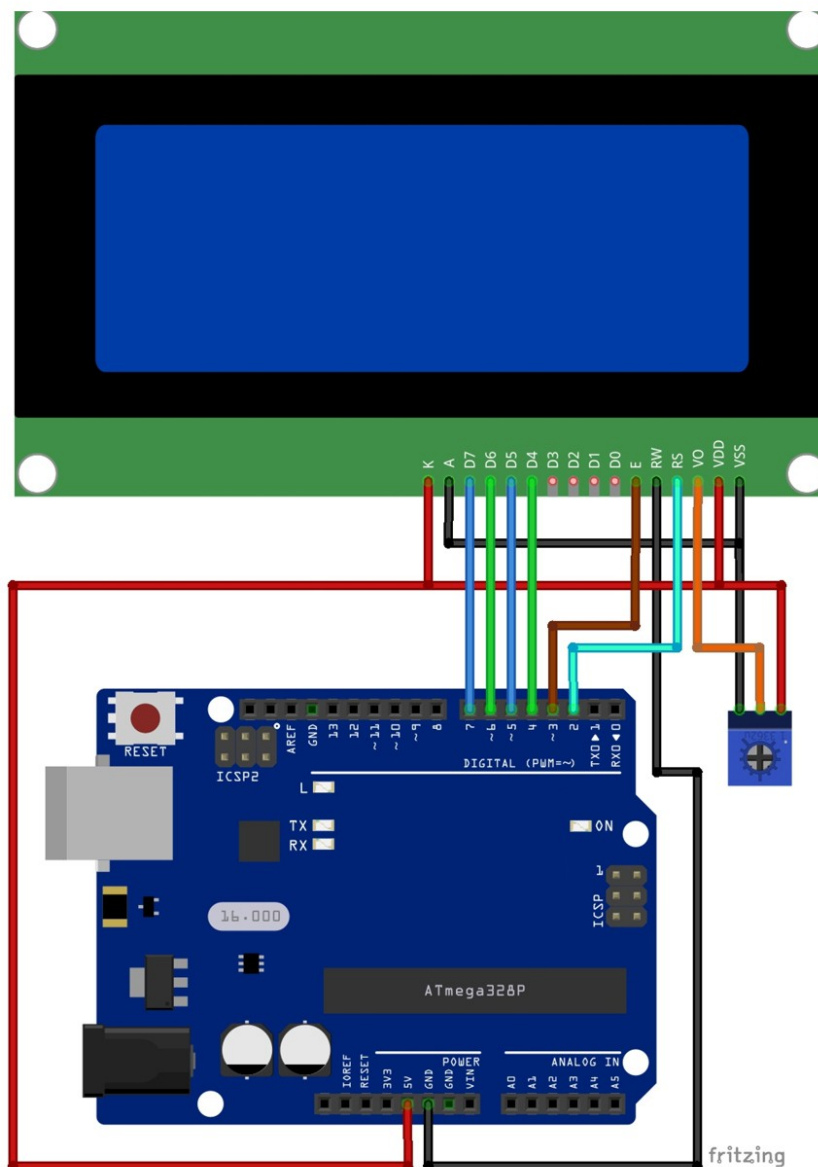
Für den Raspberry Pi muss zuerst das Betriebssystem installiert werden, dann muss alles so eingerichtet werden, dass es im Headless-Modus verwendet werden kann. Der Headless-Modus ermöglicht eine Fernverbindung zum Raspberry Pi, ohne dass ein PC-Bildschirm, eine Maus oder eine Tastatur erforderlich ist. Die einzigen Dinge, die in diesem Modus verwendet werden, sind der Raspberry Pi selbst, die Stromversorgung und die Internetverbindung. Das alles wird in dem kostenlosen eBook ausführlich erklärt:

[Raspberry Pi Quick Startup Guide](#)

**Das Betriebssystem Raspbian wird mit vorinstalliertem *Python* ausgeliefert.**

## Verbindung des Moduls mit dem Microcontroller

Verbinden Sie den Bildschirm mit dem Microcontroller wie unten abgebildet(gilt genauso für 16x02):





# Az-Delivery

LCD Pin	MC Pin	Draht Farbe
VSS	GND	Schwarzer Draht
VDD	5V	Roter Draht
V0	Poti	Oranger Draht
RS	D2	Ochre Draht
RW	GND	Schwarzer Draht
E	D3	Ochre Draht
D4	D4	Grüner Draht
D5	D5	Blauer Draht
D6	D6	Grüner Draht
D7	D7	Blauer Draht
K	5V	Roter Draht
A	GND	Schwarzer Draht

# AZ-Delivery

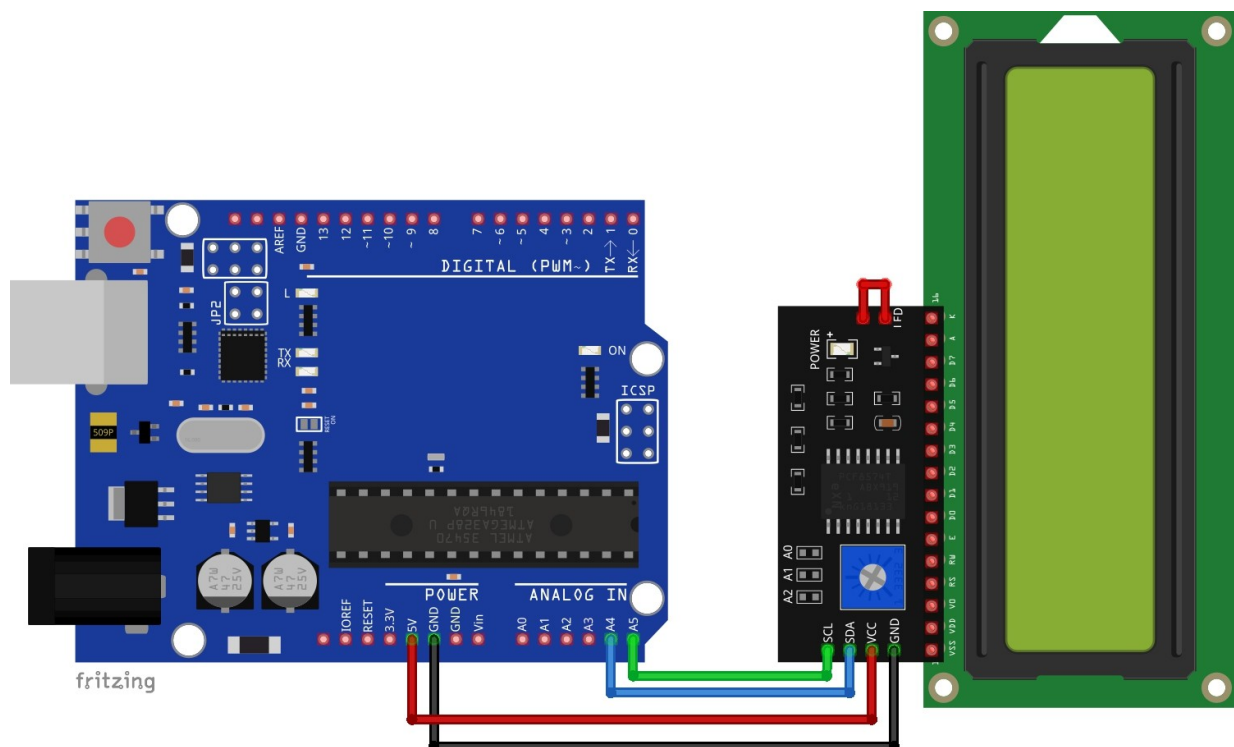
## Sketch-Beispiel

Das folgende Sketch-Beispiel ist ein modifizierter Sketch von der Arduino IDE: *File > Examples > LiquidCrystal > HelloWorld*

```
#include <LiquidCrystal.h>
const uint8_t rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
  lcd.begin(16, 2); // in "20, 4" ändern wenn 20x04 benutzt wird
  lcd.clear();
}
void loop() {
  lcd.setCursor(0, 0);
  lcd.print("AZ-Delivery");
  lcd.setCursor(0, 1);
  lcd.print(millis() / 1000);
}
```

## Verbindung des Moduls mit dem Microcontroller mit I2C-Adapter

Verbinden Sie den 16x02 Bildschirm und den I2C-Adapter mit dem Microcontroller wie unten abgebildet(gilt genauso für 20x04):



I2C-Adapter Pin	MC Pin	Drahtfarbe
SCL	A5	Grüner Draht
SDA	A4	Blauer Draht
VCC	5V	Roter Draht
GND	GND	Schwarzer Draht

# Az-Delivery

## Library für die Arduino IDE

Um das Modul mit der Arduino IDE zu verwenden, wird empfohlen, eine externe Library dafür herunterzuladen. Die in diesem eBook verwendete Library heißt *LiquidCrystal\_I2C*. Um Sie runterzuladen, klicken Sie auf diesen [link](#) und laden Sie die .zip-Datei herunter.

Um die Library einzubinden, gehen Sie in die Arduino IDE und gehen zu:

*Sketch > Include Library > Add .ZIP Library*

und wenn sich ein neues Fenster öffnet, suchen und wählen Sie die heruntergeladene .zip-Datei aus.

# AZ-Delivery

## Sketch-Beispiel

Das folgende Sketch-Beispiel ist ein modifizierter Sketch von der Arduino IDE: *File > Examples > LiquidCrystal > HelloWorld*

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 16, 2);
//in "20, 4" ändern wenn 20x04 benutzt wird

void setup() {
  lcd.init();
  lcd.backlight();
  delay(250);
  lcd.noBacklight();
  delay(1000);
  lcd.backlight();
  delay(1000);
}

void loop() {
  lcd.setCursor(0, 0);
  lcd.print("AZ-Delivery");
  lcd.setCursor(0, 1);
  lcd.print(millis() / 1000);
  delay(100);
}
```

# Az-Delivery

Der Sketch beginnt mit der Einbeziehung von Libraries mit den Namen *Wire* und *LiquidCrystal\_I2C*.

Dann wird ein Objekt mit dem Namen *lcd* erzeugt. Das Objekt repräsentiert die Anzeige selbst, und um dieses Objekt zu erzeugen, wird die folgende Codezeile verwendet:

```
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 16, 2);
```

Wobei *0x27* die I2C-Adresse des I2C-Adapters ist. *16* ist die Anzahl der Zeichen pro Zeile und *2* die Anzahl der Zeilen.

In der *setup()* Funktion wird das *lcd* -Objekt mit der folgenden Codezeile initialisiert: `lcd.init();`

Am Ende der *setup()* Funktion, wird die Hintergrundbeleuchtung getestet, indem sie mit einer Verzögerung von *1000ms* (*1s*) AUS- und EINGeschaltet wird.

In der *loop()* Funktion werden zwei vordefinierte Funktionen von der *LiquidCrystal\_I2C* Library verwendet.

# Az-Delivery

Die erste Funktion heißt *setCursor()*. Die Funktion hat zwei Argumente und gibt keinen Wert zurück. Die Werte der Argumente sind ganze Zahlen. Die erste Zahl stellt die Y-Position des Cursors dar, mit Werten im Bereich von 0 bis 1, wobei 0 die erste Zeile und 1 die zweite Zeile des Bildschirms darstellt. Das zweite Argument stellt die X-Position des Cursors dar, mit Werten im Bereich von 0 bis 15, wobei 0 die erste Spalte und 15 die letzte Spalte des Bildschirms widerspiegelt.

Die Funktion muss vor der *print()*-Funktion verwendet werden. Um der *print()*-Funktion zu zeigen, wo der Text angezeigt werden soll. Wenn Sie die Funktion *setCursor()* nicht verwenden, zeigt die Funktion *print()* den Text an der Position (0, 0) an.

Die *print()*-Funktion hat ein Argument und gibt keinen Wert zurück. Das Argument stellt den Text, einen Zeichenkettenwert, dar, der auf dem Bildschirm angezeigt wird.

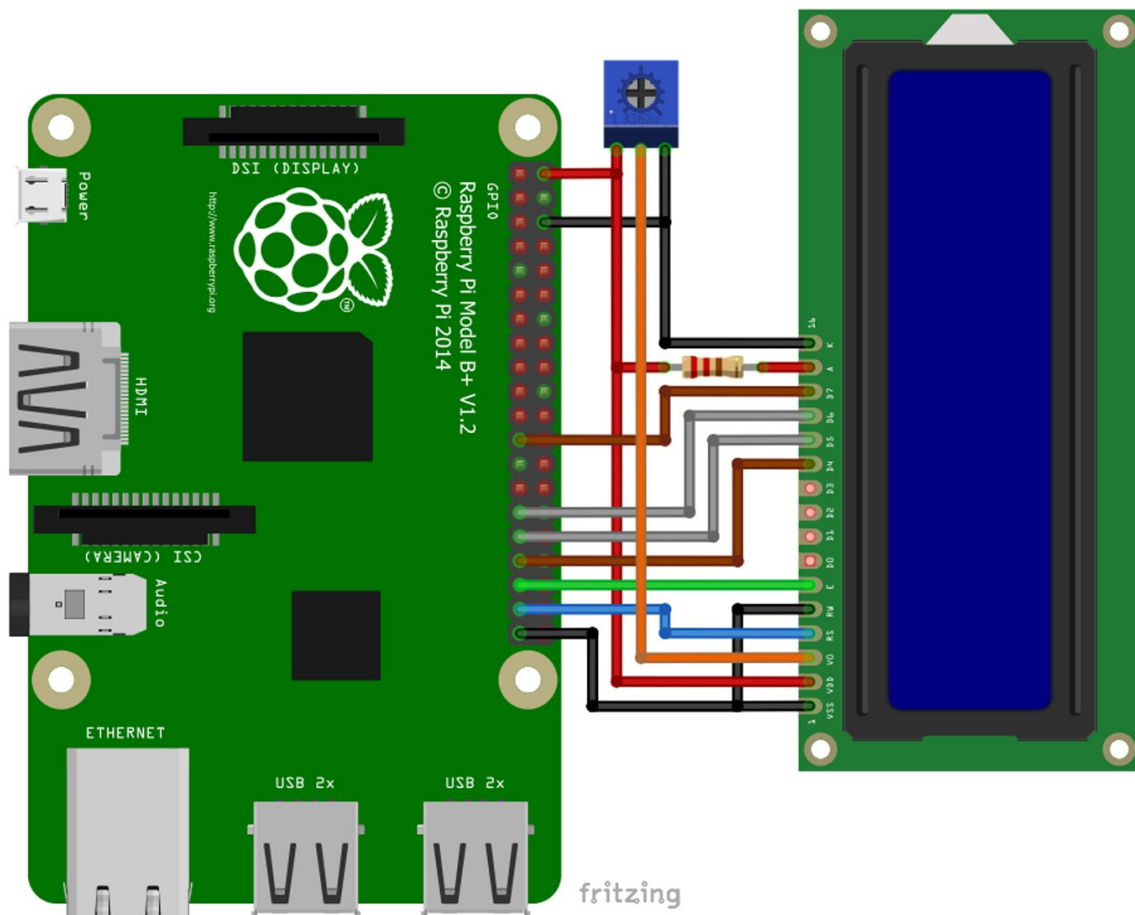
In der *loop()*-Funktion wird zuerst der Cursor auf die erste Zeile gesetzt und dann zeigt die *print()*-Funktion die Meldung *AZ-Delivery* an. Dann wird der Cursor auf die zweite Zeile gesetzt, und die Anzahl der Sekunden seit dem letzten Einschalten oder dem letzten Zurücksetzen des Microcontrollers wird angezeigt.

**HINWEIS: Sollte Ihr Display nichts anzeigen. müssen Sie an dem Poti auf dem I2C Adapter drehen.**

## Verbindung des Bildschirms mit dem Raspberry Pi

Verbinden Sie das Modul mit dem Raspberry Pi wie unten abgebildet (gilt genauso für 20x04):

16X02 Blue LCD Screen  
Connection diagram with Raspberry Pi





# Az-Delivery

Screen Pin	Raspberry Pi Pin	Physical Pin	Draht color
VSS	GND	6	Schwarzer Draht
VDD	5V	2	Roter Draht
RS	GPIO26	37	Blauer Draht
RW	GND	39	Schwarzer Draht
E	GPIO19	35	Grüner Draht
D4	GPIO13	33	Brown Draht
D5	GPIO6	31	Grauer Draht
D6	GPIO5	29	Grauer Draht
D7	GPIO11	23	Brown Draht
K	GND	6	Roter Draht
A	5V, via 220Ω resistor	2	Oranger Draht
V0	potentiometer center pin		

Potentiometer			
GND	Rechter Pin	6	Schwarzer Draht
5V	Linker Pin	2	Roter Draht

# Az-Delivery

## Python-Skript

Zwei Skripte werden erstellt, eins für alle Funktionen und das andere um diese Funktionen zu verwenden. Nachfolgend finden Sie den code für das erste Skript:

```
import RPi.GPIO as GPIO
import time

LCD_RS = 26
LCD_E  = 19
LCD_D4 = 13
LCD_D5 = 6
LCD_D6 = 5
LCD_D7 = 11

# Define some device constants
LCD_WIDTH = 16      # Maximum characters per line in "20" ändern
                    # bei 20x04
LCD_CHR = True
LCD_CMD = False

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line

# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005
```

# Az-Delivery

```
# one tab
def lcd_init(RS, E, D4, D5, D6, D7):
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)
    global LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7
    LCD_RS = RS
    LCD_E = E
    LCD_D4 = D4
    LCD_D5 = D5
    LCD_D6 = D6
    LCD_D7 = D7
    GPIO.setup(LCD_E, GPIO.OUT) # E
    GPIO.setup(LCD_RS, GPIO.OUT) # RS
    GPIO.setup(LCD_D4, GPIO.OUT) # DB4
    GPIO.setup(LCD_D5, GPIO.OUT) # DB5
    GPIO.setup(LCD_D6, GPIO.OUT) # DB6
    GPIO.setup(LCD_D7, GPIO.OUT) # DB7
    # Initialise display
    lcd_byte(0x33, LCD_CMD) # 110011 Initialise
    # 110010 Initialise
    lcd_byte(0x32, LCD_CMD)
    # 000110 Cursor move direction
    lcd_byte(0x06, LCD_CMD)
    # 001100 Display On,Cursor Off, Blink Off
    lcd_byte(0x0C, LCD_CMD)
    # 101000 Data length, number of lines, font size
    lcd_byte(0x28, LCD_CMD)
    # 000001 Clear display
    lcd_byte(0x01, LCD_CMD)
    time.sleep(E_DELAY)
```

# Az-Delivery

```
# one tab
def lcd_byte(bits, mode):
    GPIO.output(LCD_RS, mode) # RS
    GPIO.output(LCD_D4, False)
    GPIO.output(LCD_D5, False)
    GPIO.output(LCD_D6, False)
    GPIO.output(LCD_D7, False)
    if bits & 0x10 == 0x10:
        GPIO.output(LCD_D4, True)
    if bits & 0x20 == 0x20:
        GPIO.output(LCD_D5, True)
    if bits & 0x40 == 0x40:
        GPIO.output(LCD_D6, True)
    if bits & 0x80 == 0x80:
        GPIO.output(LCD_D7, True)
    lcd_toggle_enable() # Toggle 'Enable' pin
    GPIO.output(LCD_D4, False)
    GPIO.output(LCD_D5, False)
    GPIO.output(LCD_D6, False)
    GPIO.output(LCD_D7, False)
    if bits & 0x01 == 0x01:
        GPIO.output(LCD_D4, True)
    if bits & 0x02 == 0x02:
        GPIO.output(LCD_D5, True)
    if bits & 0x04 == 0x04:
        GPIO.output(LCD_D6, True)
    if bits & 0x08 == 0x08:
        GPIO.output(LCD_D7, True)
    # Toggle 'Enable' pin
    lcd_toggle_enable()
```

# Az-Delivery

```
# one tab
def lcd_toggle_enable():
    # Toggle enable
    time.sleep(E_DELAY)
    GPIO.output(LCD_E, True)
    time.sleep(E_PULSE)
    GPIO.output(LCD_E, False)
    time.sleep(E_DELAY)

def lcd_string(message, line):
    # Send string to display
    LCD_LINE_1 = 0x80
    LCD_LINE_2 = 0xC0
    message = message.ljust(LCD_WIDTH, " ")
    if line == 0:
        lcd_byte(LCD_LINE_1, LCD_CMD)
    elif line == 1:
        lcd_byte(LCD_LINE_2, LCD_CMD)
    else:
        print('This lcd has two lines, line 0 and line 1!')
    for i in range(LCD_WIDTH):
        lcd_byte(ord(message[i]), LCD_CHR)

def lcd_clear():
    lcd_byte(0x01, LCD_CMD)
```

Speichern Sie das Skript unter dem Namen `lcd16x02.py`.

Der Code im Skript ist ein modifizierter Code aus dem Skript unter [link](#):

# AZ-Delivery

Nachfolgend finden Sie den Code für das Haupt-Skript:

```
import lcd16x02
from time import sleep
LCD_RS = 26
LCD_E  = 19
LCD_D4 = 13
LCD_D5 = 6
LCD_D6 = 5
LCD_D7 = 11
# Initialise display
lcd16x02.lcd_init(LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7)
i = 0
print('[Press CTRL + C to end the script!]' )
try:
    lcd16x02.lcd_string('AZ-Delivery', 0)
    print('AZ-Delivery')
    print('Printing variable on the LCD...')
    while True:
        lcd16x02.lcd_string('{}'.format(i), 1)
        i+=1
        sleep(0.001) # 1 millisecond delay

except KeyboardInterrupt:
    print('Script end!')

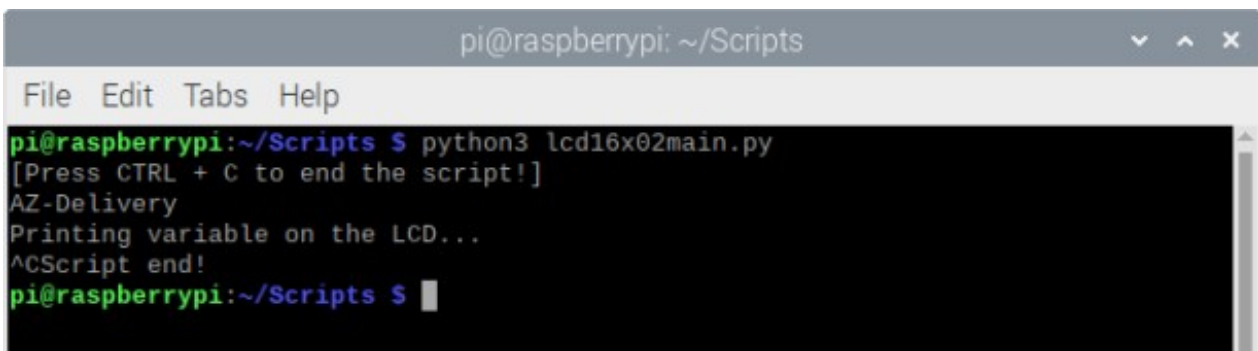
finally:
    lcd16x02.lcd_clear()
```

# AZ-Delivery

Speichern Sie das Skript unter dem Namen `lcd16x02main.py` im gleichen Verzeichnis, wie das vorherige Skript. Um das Skript auszuführen, öffnen Sie das Terminal in dem Verzeichnis, in dem das Skript gespeichert ist, und führen Sie den folgenden Befehl aus:

```
python3 lcd16x02main.py
```

Die Ausgabe sollte wie folgt aussehen:



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ python3 lcd16x02main.py
[Press CTRL + C to end the script!]
AZ-Delivery
Printing variable on the LCD...
^CScript end!
pi@raspberrypi:~/Scripts $
```

Um das Skript zu stoppen, drücken Sie 'Strg + C' auf der Tastatur.

# AZ-Delivery

Das erste Skript wird verwendet, um alle Funktionen zur Steuerung des LCD-Bildschirms zu erstellen, was in diesem eBook nicht behandelt wird. Es wird nur die Hauptfunktion des Skripts erklärt.

Das Hauptskript beginnt mit dem Importieren des ersten Skripts und mit dem Importieren der *Sleep*-Funktion aus der *time* Library.

Dann werden sechs Variablen definiert, die Pins des Bildschirms darstellen, welche mit den Pins des Raspberry Pi verbunden sind.

Als nächstes wird der Bildschirm mit der folgenden Codezeile initialisiert:

```
lcd16x02.lcd_init(LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7)
```

Dann wird die Variable "*i*" erstellt und mit dem Wert Null initialisiert. Diese wird verwendet, um sich verändernde Daten auf dem Display anzuzeigen.

Dann wird ein *try-except-finally* Codeblock erstellt. Im *try* block wird zuerst die Meldung *AZ-Delivery* in der ersten Zeile des Bildschirms angezeigt, und dann wird ein indefinite loop block (*while True:*) erstellt. Darin wird die Variable *i* auf der zweiten Zeile des Displays angezeigt und der Wert der Variablen *i* um 1 erhöht. Zwischen jeder Wiederholung des indefinite loop Blocksgibt es eine Pause von einer Millisekunde (*sleep(0.001)*).



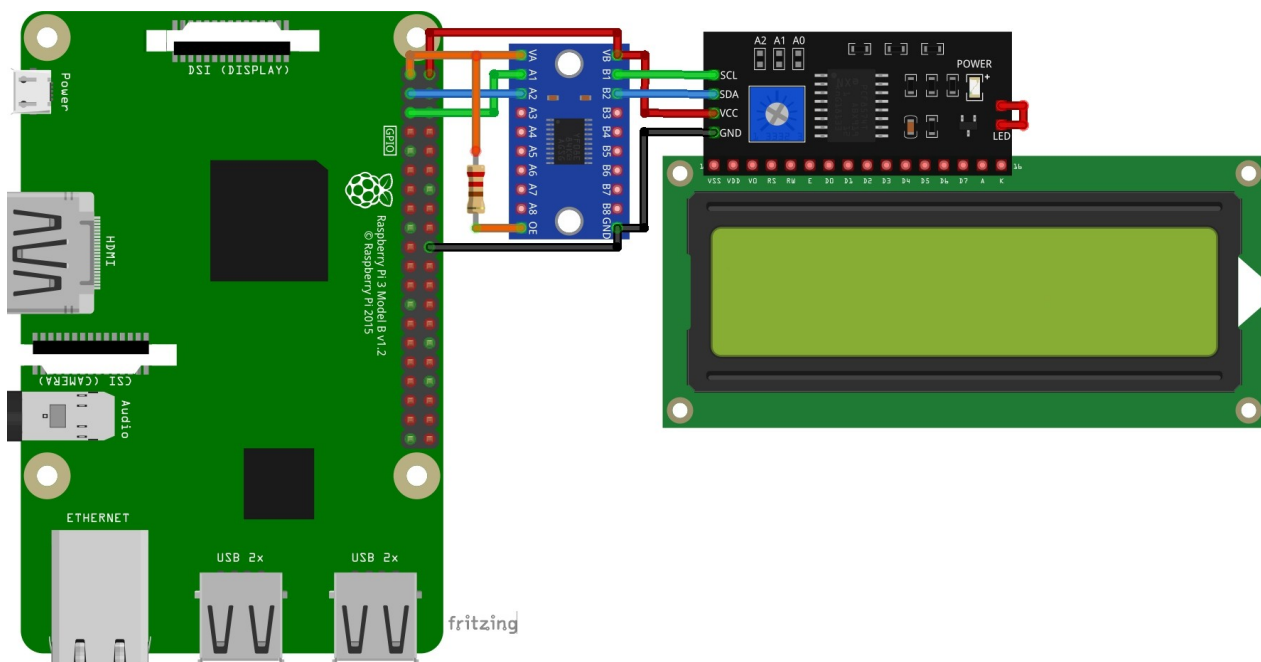
# Az-Delivery

Der *except* Codeblock wird mit *Strg+C* ausgeführt. Das wird *KeyboardInterrupt* genannt. Wenn dieser Codeblock ausgeführt wurde, wird im Terminal die Nachricht *Script end!* angezeigt.

Der *finally* Codeblock wird nach dem Skript ausgeführt. Wenn der *finally* Codeblock ausgeführt wurde, wird die *lcd\_clear()* Funktion aufgerufen, welche den Datenpuffer des Bildschirms löscht.

## Verbindung des Bildschirms mit dem Raspberry Pi mit I2C-Adapter

Verbinden Sie den 16x02 Bildschirm und den I2C-Adapter mit dem Raspberry Pi, wie unten abgebildet (gilt genauso für 20x04):



Hier muss der Logikpegelwandler verwendet werden, da der I2C-Adapter nur im 5V-Bereich arbeitet. Der in diesem eBook verwendete Logikpegelwandler heißt [\*TXS0108E Logic Level Converter\*](#).

Verbinden Sie den I2C-Adapter mit dem LCD-Bildschirm wie im Anschlussdiagramm dargestellt. Stellen Sie sicher, dass der integrierte Jumper für die Hintergrundbeleuchtung angeschlossen ist (**Roter Draht**, die rechte Seite des I2C-Adapters auf dem Anschlussschema).

# Az-Delivery

I2C-Adapter Pin	LLC Pin	Drahtfarbe
SCL	B1	Grüner Draht
SDA	B2	Blauer Draht
VCC	VB	Roter Draht
GND	GND	Schwarzer Draht

LLC Pin	Raspberry Pi Pin	Physischer Pin	Drahtfarbe
VA	3.3V	1	Orange Draht
A1	GPIO3	5	Grüner Draht
A2	GPIO2	3	Blauer Draht
OE	3.3V (via resistor)	1	Orange Draht
GND	GND	20	Schwarzer Draht
VB	5V	2	Roter Draht

## Libraries und Tools für Python

Um den Bildschirm mit der Raspberry Pi zu benutzen, wird empfohlen, eine externe Library herunterzuladen. Um sie herunterzuladen, gehen Sie auf den folgenden [link](#) und laden Sie das Skript `lcd_class_i2c.py` herunter. Speichern Sie das Skript in demselben Verzeichnis, in dem auch das Skript des nächsten Kapitels gespeichert ist.

Das Skript `lcd_class_i2c.py` verwendet die `smbus` Library für Python. Falls diese noch nicht installiert ist, öffnen Sie das Terminal und führen Sie nacheinander die folgenden Befehle aus:

```
sudo apt-get update
```

```
sudo apt-get install python3-smbus python3-dev
```

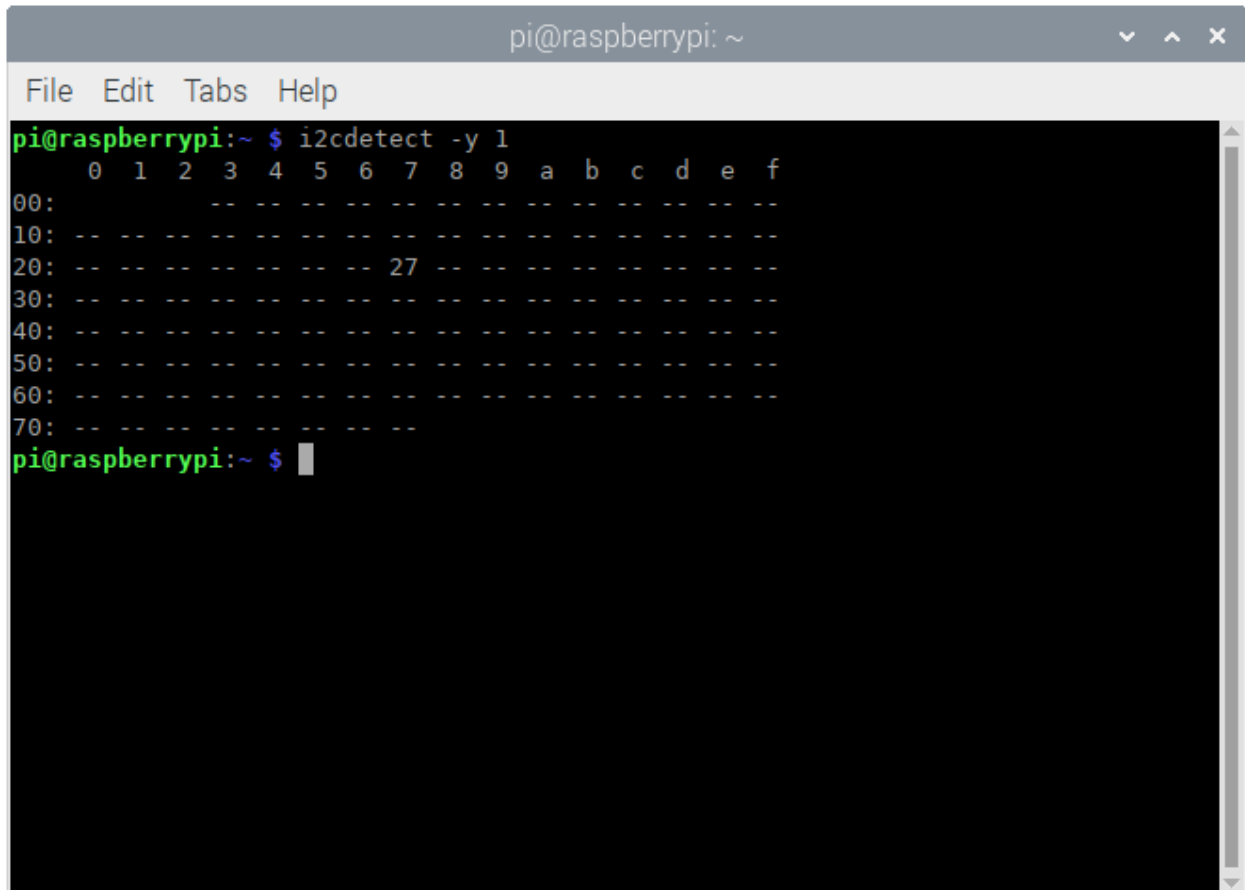
Um die I2C-Adresse des I2C-Adapters zu ermitteln, muss das `i2c-tools` installiert sein. Falls es nicht bereits installiert ist, führen Sie den folgenden Befehl im Terminal aus:

```
sudo apt-get install i2c-tools
```

# Az-Delivery

Um die I2C-Adresse des I2C-Adapters zu ermitteln, führen Sie den folgenden Befehl aus:

```
i2cdetect -y 1
```



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ i2cdetect -y 1  
 0 1 2 3 4 5 6 7 8 9 a b c d e f  
00: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
10: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
20: -- -- -- -- -- -- -- 27 -- -- -- -- -- --  
30: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
40: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
50: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
60: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
70: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
pi@raspberrypi:~ $
```

wobei 0x27 die I2C-Adresse des I2C-Adapters darstellt.

# AZ-Delivery

## Python-Skript

Nachfolgend finden Sie den Code für das Hauptskript:

```
import lcd_class_i2c as LCD
import time

I2C_ADDR = 0x27
LINE_WIDTH = 16

screen = LCD.lcd(line_width=LINE_WIDTH, i2c_address=I2C_ADDR)

print('[Press CTRL + C to end the script!]\n')
try:
    while True:
        print('Printing messages on the screen')
        screen.lcd_print('AZ-DELIVERY', 'LINE_1', 'CENTER')
        time.sleep(1) # 3 second delay

        print('Printing variable on the screen')
        for i in range(100):
            if i < 10:
                screen.lcd_print('0{}'.format(i), 'LINE_2', 'CENTER')
            else:
                screen.lcd_print('{}'.format(i), 'LINE_2', 'CENTER')
            time.sleep(0.00001)
```

# Az-Delivery

```
# one tab
    print('Testing backlight')
    time.sleep(1)
    screen.backlight('OFF')
    time.sleep(1)
    screen.backlight('ON')
    time.sleep(1)
    print('Clearing the screen\n')
    # Blank display
    screen.clear_screen()
    time.sleep(1)

except KeyboardInterrupt:
    print('\nScript end!')

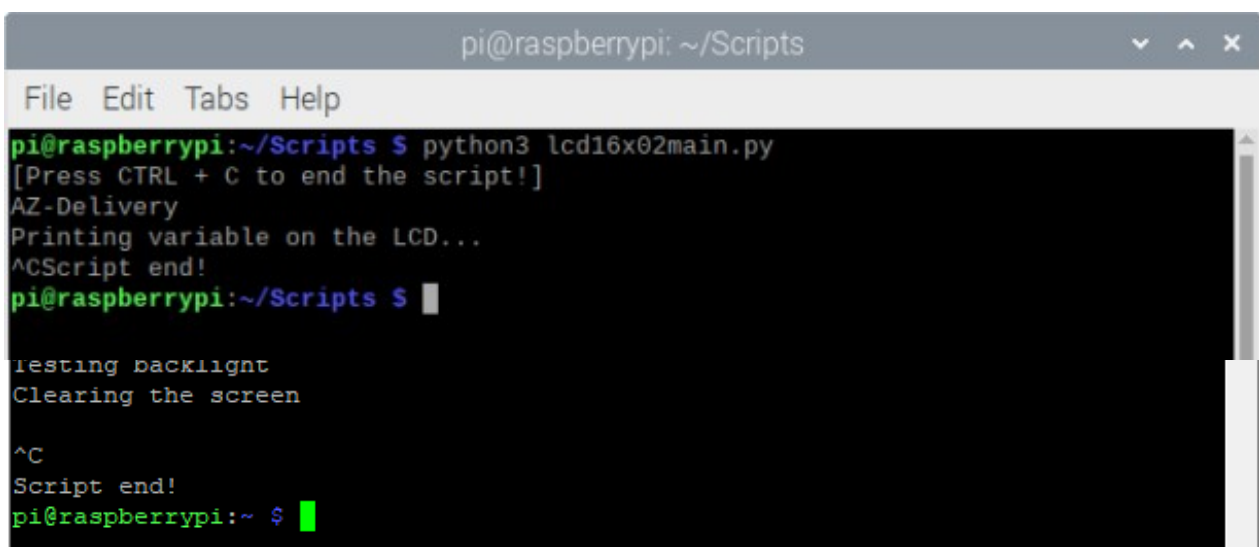
finally:
    screen.clear_screen()
```

# AZ-Delivery

Speichern Sie das Skript unter dem Namen `lcd16x02i2c.py` im gleichen Verzeichnis, wie das Skript `lcd_class_i2c.py`. Um das Skript auszuführen, öffnen Sie das Terminal in dem Verzeichnis, in dem die Skripten gespeichert sind, und führen Sie den folgenden Befehl aus:

```
python3 lcd16x02main.py
```

Die Ausgabe sollte wie folgt aussehen:



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ python3 lcd16x02main.py
[Press CTRL + C to end the script!]
AZ-Delivery
Printing variable on the LCD...
^CScript end!
pi@raspberrypi:~/Scripts $ █

Testing backlight
Clearing the screen

^C
Script end!
pi@raspberrypi:~ $ █
```

Um das Skript zu stoppen, drücken Sie 'Strg + C' auf der Tastatur.



# Az-Delivery

Das erste Skript wird verwendet, um alle Funktionen zur Steuerung des LCD-Bildschirms zu erstellen, was in diesem eBook nicht behandelt wird.

Das Hauptskript beginnt mit dem Importieren des ersten Skripts `lcd_class_i2c.py` und der `time` Library.

Als nächstes wird das Objekt `screen` erstellt. Damit wird der Bildschirm gesteuert. Wir erstellen das Objekt mit der folgenden Codezeile:

```
screen=LCD.lcd(line_width=LINE_WIDTH, i2c_address=I2C_ADDR)
```

Wobei der `lcd()` Konstruktor zwei Argumente hat. Das erste Argument heißt `line_width` und stellt die Anzahl der Zeichen pro Zeile des Bildschirms dar. Das `lcd_class_i2c.py` Skript kann für 16x02 LCD-Bildschirme, als auch für 20x04 Bildschirme verwendet werden. Also sind die Werte, die dem `line_width` Argument übermittelt werden, 16 oder 20. Jeder andere Wert führt zu einem Fehler, und das Skript setzt den Wert dieses Arguments auf 16. Das zweite Argument, `i2c_address` genannt stellt die I2C-Adresse des I2C-Adapters dar, was in diesem Fall 0x27 beträgt.

# AZ-Delivery

Dann wird ein *try-except-finally* Codeblock erstellt. Im *try* block wird ein Indefinite loop Block (*while True:*) erstellt. Im Indefinite loop Block, wird zuerst die Meldung *AZ-Delivery* in der ersten Zeile des Bildschirms angezeigt. Die Nachricht ist in der Mitte der Zeile positioniert. Das wird mit der folgenden Codezeile gemacht:

```
screen.lcd_print('AZ-DELIVERY', 'LINE_1', 'CENTER')
```

Wobei die *lcd\_print()* Funktion verwendet wird. Diese Funktion zeigt eine Nachricht auf dem Bildschirm an. Sie hat drei Argumente und gibt keinen Wert zurück. Das zweite und dritte Argument sind optional. Das erste Argument ist eine Zeichenfolge, welche die auf dem Bildschirm angezeigte Nachricht darstellt. Das zweite Argument, ebenfalls eine Zeichenkette, stellt die Zeile dar, auf der die Nachricht angezeigt wird. Die Werte für dieses Argument sind: *LINE\_1* or *LINE\_2*. Der Standardwert ist *LINE\_1*, welcher ausgewählt wird, wenn das Argument nicht verwendet wird. Jeder andere Wert führt zu einem Fehler, und der Wert wird auf den Standardwert *LINE\_1* gesetzt. Das dritte Argument, ebenfalls eine Zeichenfolge, stellt die Ausrichtung des Textes in der Zeile dar. Die Werte dieses Arguments sind: *LEFT*, *CENTER* or *RIGHT*. Der Standardwert ist *LEFT*, welcher ausgewählt wird, wenn das Argument nicht verwendet wird. Jeder andere Wert führt zu einem Fehler, und der Wert wird ebenfalls auf den Standardwert *LEFT* gesetzt.

Um eine Variable auf dem Bildschirm anzuzeigen, verwenden Sie die folgenden Codezeilen:

```
my_var = 10
```

# Az-Delivery

```
screen.lcd_print('{}'.format(my_var))
```

# Az-Delivery

Um die Hintergrundbeleuchtung des Bildschirms zu steuern, wird die Funktion `backlight()` verwendet. Diese Funktion hat ein Argument und gibt keinen Wert zurück. Der Wert des Arguments ist eine Zeichenfolge, die nur zwei Werte haben kann: EIN oder AUS. Um den Bildschirm EINzuschalten, wird die folgende Codezeile verwendet:

```
screen.backlight('ON')
```

Um sie auszuschalten, verwenden Sie Folgendes:

```
screen.backlight('OFF')
```

Um den Bildschirm zu leeren (Datenpuffer des Screens), verwenden Sie die `clear_screen()` Funktion. Diese Funktion hat keine Argumente und gibt keinen Wert zurück.

Der *except* Codeblock wird ausgeführt, wenn `STRG + C` auf der Tastatur gedrückt wird. Das wird *KeyboardInterrupt* genannt. Wenn dieser Block ausgeführt wurde, wird die Nachricht *Script end!* im Terminal angezeigt.

Der *finally* Codeblock wird nach dem Skript ausgeführt. Wenn der *finally* Codeblock ausgeführt wurde, wird die `clear_screen()` Funktion ausgeführt. Diese Funktion leert den Datenpuffer des Bildschirms.

**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 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>