QAPASS 1602A LCD-Display mit dem Raspberry PI (Doku von Christoph)

**Pinout:**

(Pin 1-16, an Display angeschrieben)

| **LCD Pin** | **Bezeichnung** | **Raspberry Pi GPIO** |
| --- | --- | --- |
| 1 | VSS (GND) | GND (Pin 6) |
| 2 | VDD (+5V) | 5V (Pin 2) |
| 3 | VO (Kontrast) | Potentiometer-Mitte |
| 4 | RS (Register Select) | GPIO 26 (Pin 37) |
| 5 | RW (Read/Write) | GND (Schreibmodus) |
| 6 | E (Enable) | GPIO 19 (Pin 35) |
| 11 | D4 (Data 4) | GPIO 13 (Pin 33) |
| 12 | D5 (Data 5) | GPIO 6 (Pin 31) |
| 13 | D6 (Data 6) | GPIO 5 (Pin 29) |
| 14 | D7 (Data 7) | GPIO 11 (Pin 23) |
| 15 | A (Backlight +) | 5V (Pin 2) |
| 16 | K (Backlight -) | GND (Pin 6) |

**Libraries und Testprogramm:**

sudo apt-get update

sudo apt-get install python3-rpi.gpio

**Testprogramm:**

import RPi.GPIO as GPIO

import time

# LCD Pin-Konfiguration

RS = 26

E = 19

D4 = 13

D5 = 6

D6 = 5

D7 = 11

# GPIO-Setup

GPIO.setmode(GPIO.BCM)

GPIO.setup(RS, GPIO.OUT)

GPIO.setup(E, GPIO.OUT)

GPIO.setup(D4, GPIO.OUT)

GPIO.setup(D5, GPIO.OUT)

GPIO.setup(D6, GPIO.OUT)

GPIO.setup(D7, GPIO.OUT)

def lcd\_send\_byte(bits, mode):

# RS auf HIGH für Daten, LOW für Befehl

GPIO.output(RS, mode)

# High-Nibble

GPIO.output(D4, bits & 0x10 == 0x10)

GPIO.output(D5, bits & 0x20 == 0x20)

GPIO.output(D6, bits & 0x40 == 0x40)

GPIO.output(D7, bits & 0x80 == 0x80)

# Enable-Puls

lcd\_toggle\_enable()

# Low-Nibble

GPIO.output(D4, bits & 0x01 == 0x01)

GPIO.output(D5, bits & 0x02 == 0x02)

GPIO.output(D6, bits & 0x04 == 0x04)

GPIO.output(D7, bits & 0x08 == 0x08)

lcd\_toggle\_enable()

def lcd\_toggle\_enable():

time.sleep(0.0005)

GPIO.output(E, True)

time.sleep(0.0005)

GPIO.output(E, False)

time.sleep(0.0005)

def lcd\_init():

# Initialisierung im 4-Bit-Modus

lcd\_send\_byte(0x33, False) # 8-Bit-Modus → 4-Bit

lcd\_send\_byte(0x32, False) # 4-Bit-Modus

lcd\_send\_byte(0x28, False) # 2 Zeilen, 5x8 Pixel

lcd\_send\_byte(0x0C, False) # Display ein, Cursor aus

lcd\_send\_byte(0x01, False) # Display löschen

time.sleep(0.002) # Warten auf Löschvorgang

def lcd\_string(message, line):

# Nachricht an LCD senden (Zeile 1 oder 2)

lcd\_send\_byte(line, False)

for char in message:

lcd\_send\_byte(ord(char), True)

try:

lcd\_init()

lcd\_string("Hallo Raspberry!", 0x80) # Zeile 1

lcd\_string("LCD funktioniert!", 0xC0) # Zeile 2

time.sleep(5)

except KeyboardInterrupt:

pass

finally:

GPIO.cleanup()