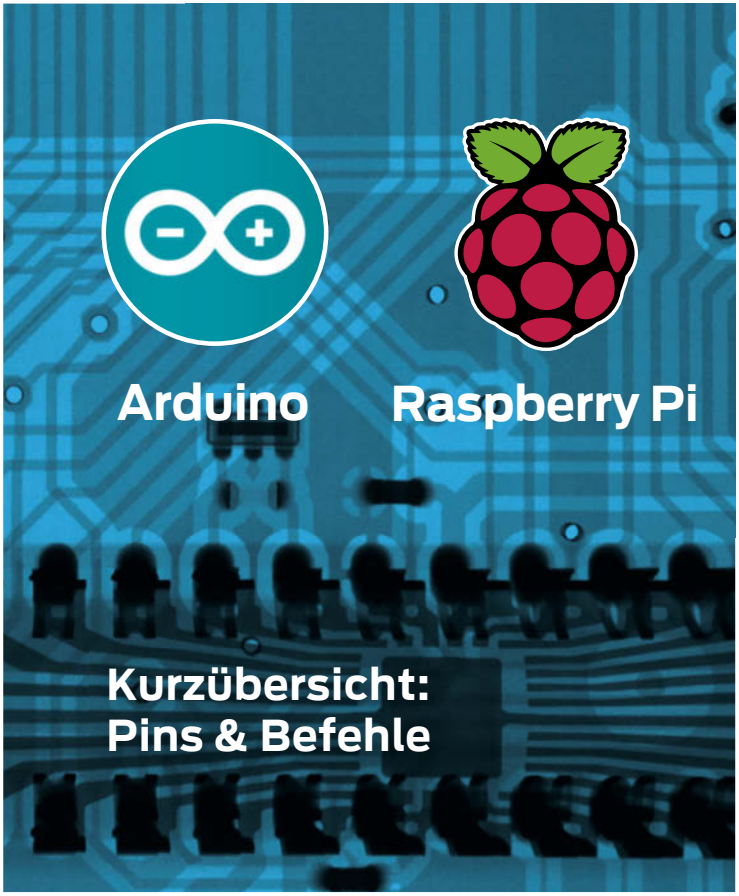
Make: kompakt



Arduino

Raspberry Pi

Kurzübersicht: Pins & Befehle

Impressum

Redaktion
Make: Magazin
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-300
Telefax: 05 11/53 52-417
Internet: www.make-magazin.de
Chefredakteur: Daniel Bachfeld (dab)
(verantwortlich für den Textteil)
Stellv. Chefredakteur: Peter König (pek)
Redaktion: Heinz Behling (hgb), Helga Hansen (hch), Carsten Meyer (cm), Florian Schäffer (fls), Elke Schick (esk)

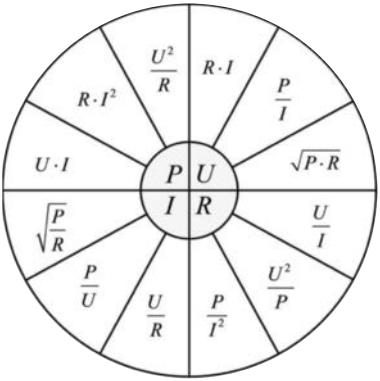
Verlag
Maker Media GmbH
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-0
Telefax: 05 11/53 52-129
Internet: www.make-magazin.de
Herausgeber: Christian Heise, Ansgar Heise
Geschäftsführer: Ansgar Heise, Dr. Alfons Schräder
Verlagsleiter: Dr. Alfons Schräder
Stellv. Verlagsleiter: Daniel Bachfeld
Anzeigenleitung: Michael Hanke (-167)
(verantwortlich für den Anzeigenteil),
www.heise.de/mediadaten/make
Leiter Vertrieb und Marketing: André Lux (-299)
Service Sonderdrucke: Julia Conrades (-156)

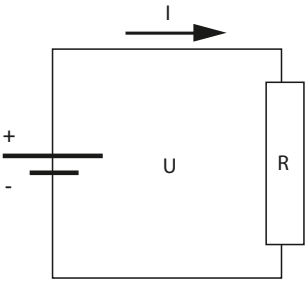
Druck: Goldschmidt GmbH,
Josefstraße 35, 49809 Lingen
Gestaltung & Layout: Sebastian Moock
Internet: www.sebastianmoock.com

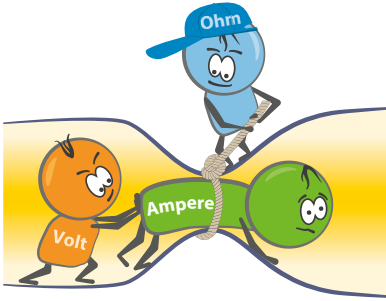
Das Ohmsche (Ω) Gesetz

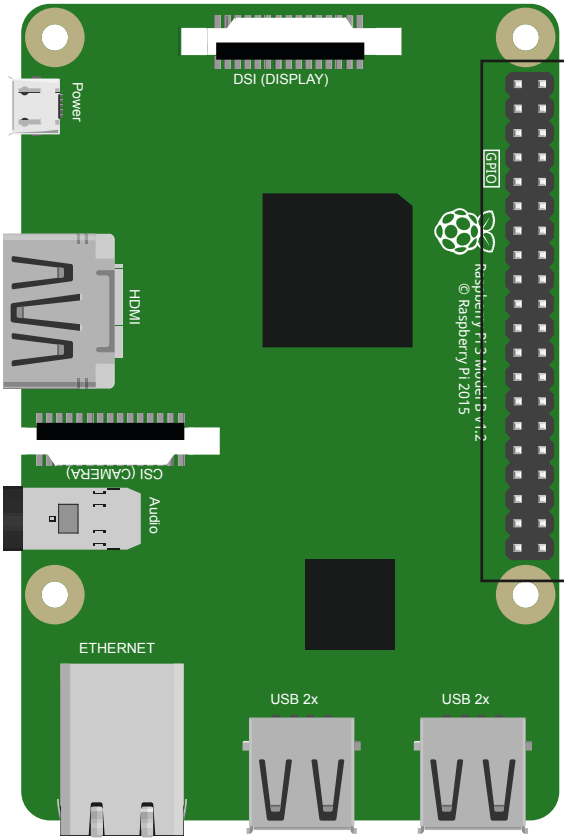
Das Ohmsche Gesetz beschreibt den Zusammenhang zwischen Spannung (U, Volt), Strom (I, Ampere) und Widerstand (R, Ohm) in einem Stromkreis. Der kleine Comic verdeutlicht nochmals die Wirkungsweise bildlich.

Aus der Ohmschen Scheibe kann man direkt die Formel zur Berechnung der gewünschten Größe herauslesen.



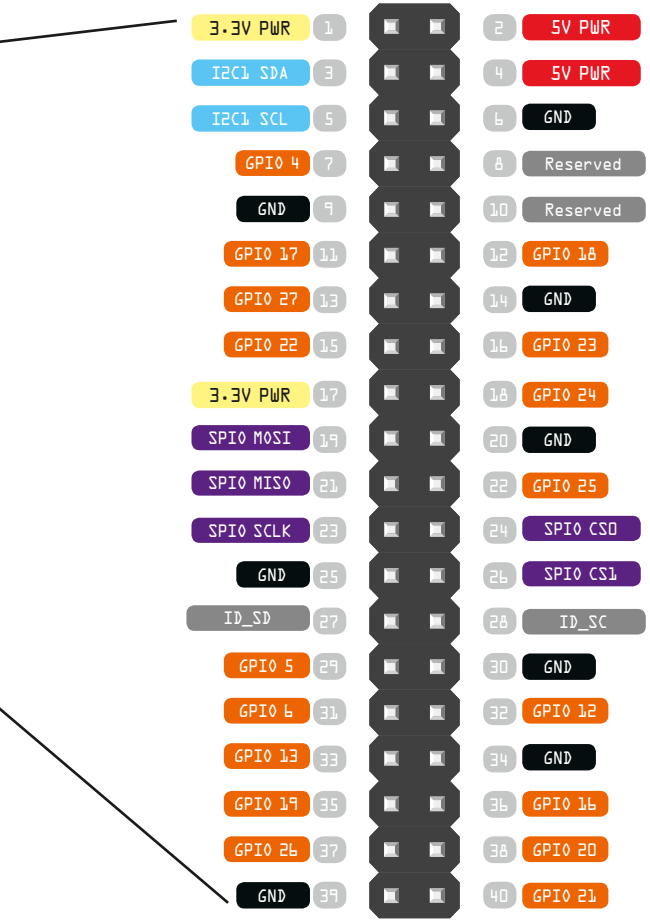






I2C1 GND GPIO SPI0

Reserved 3.3V 5V


3.3V PWR
1
2
5V PWR
I2C1 SDA
3
4
5V PWR
I2C1 SCL
5
6
GND
GPIO 4
7
8
Reserved
GND
9
10
Reserved
GPIO 17
11
12
GPIO 18
GPIO 27
13
14
GND
GPIO 22
15
16
GPIO 23
3.3V PWR
17
18
GPIO 24
SPI0 MOSI
19
20
GND
SPI0 MISO
21
22
GPIO 25
SPI0 SCLK
23
24
SPI0 CS0
GND
25
26
SPI0 CS1
ID_SD
27
28
ID_SC
GPIO 5
29
30
GND
GPIO 6
31
32
GPIO 12
GPIO 13
33
34
GND
GPIO 19
35
36
GPIO 16
GPIO 26
37
38
GPIO 20
GND
39
40
GPIO 21

Raspberry-GPIO: Ein-/Ausgabe in Python

```
 Umschaltung auf PIN-Nummerierung
GPIO.setmode(GPIO.BOARD)
 Umschaltung auf BCM-Nummerierung
GPIO.setmode(GPIO.BCM)
 Pin XX als Eingang setzen und Wert einlesen
import RPi.GPIO as GPIO
XX = 13
GPIO.setmode(GPIO.BOARD)
GPIO.setup(XX, GPIO.IN)
Eingang=GPIO.input(XX)
 Pin XX als Ausgang setzen und High bzw.Low ausgeben
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
XX = 13
GPIO.setup(XX, GPIO.OUT)
GPIO.output(XX.high)
GPIO.output(XX.low)
 PWM-Signal auf BCM 23 (Pin 16) ausgeben
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
XX = 23
GPIO.setup(XX, GPIO.OUT)
# Frequenz für PWM auf 50 Hz setzen
p = GPIO.PWM(XX,50)
```

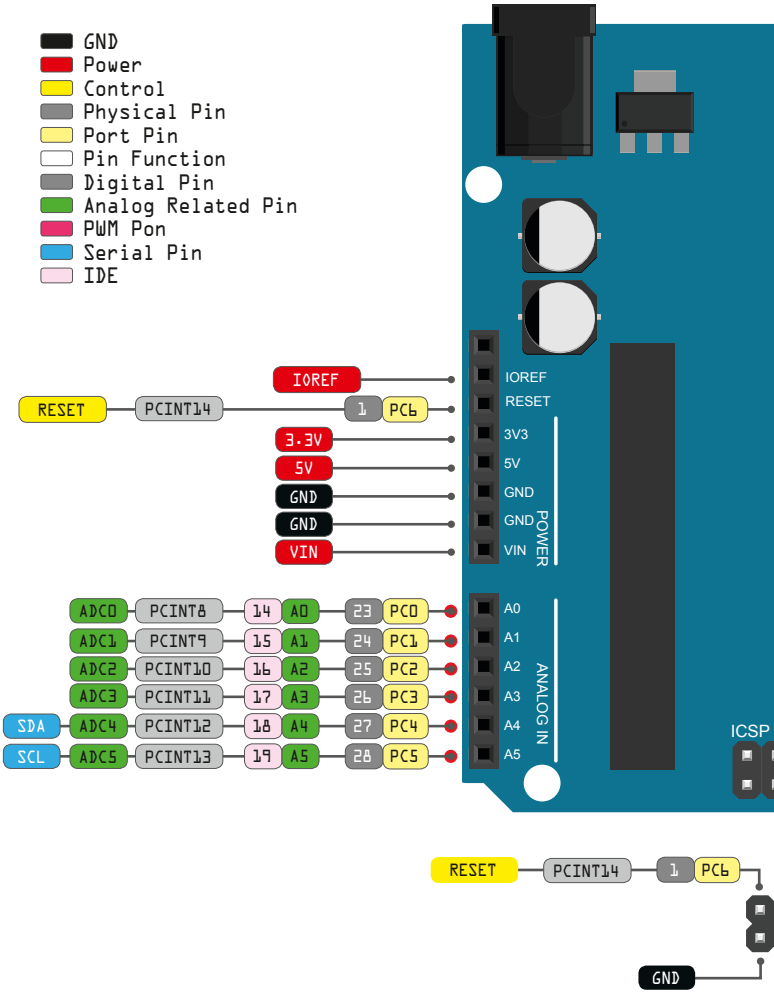
```
# PWM starten mit 50%-Signal
p.start(50)
# PWM-Signal auf 10% ändern
p.ChangeDutyCycle(10)
# PWM-Frequenz auf 100 Hz ändern
p.ChangeFrequency(100)
# PWM-Ausgabe beenden
p.Stop()
```

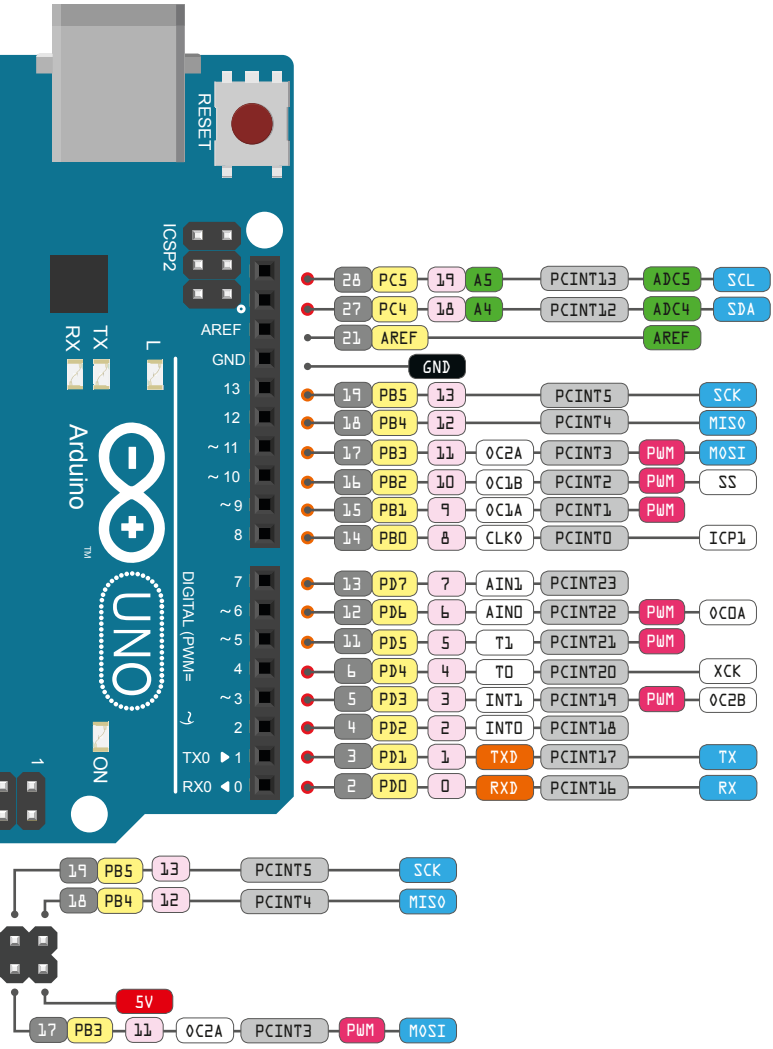
I2C-Geräteadressen scannen

```
sudo i2cdetect
```

Werte in I2C-Geräte-Register schreiben

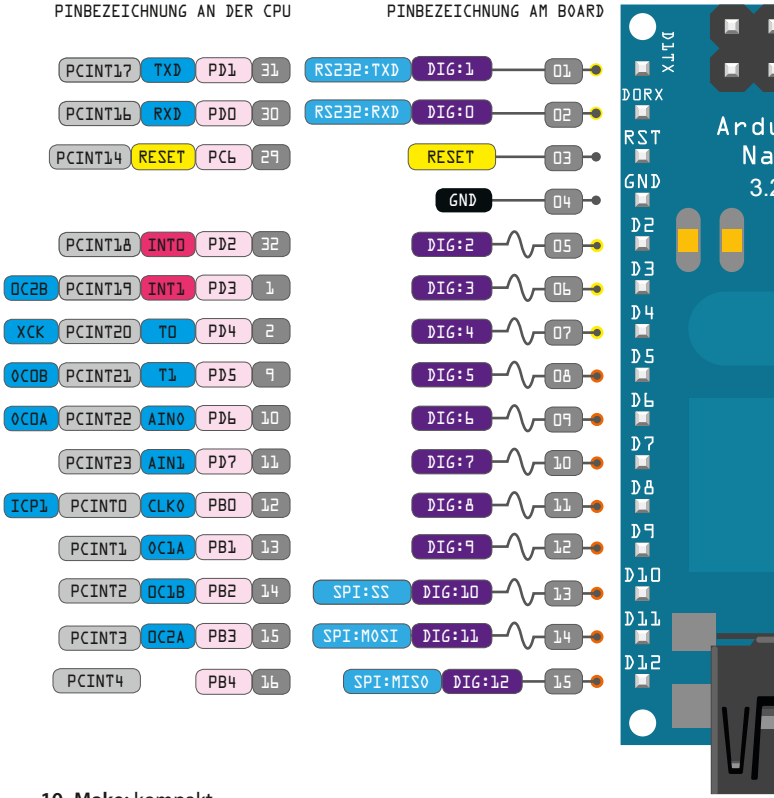
```
import smbus
bus = smbus.SMBus(1)
# Geräte-Adresse (z. B. 0x15)
DEVICE_ADDRESS = 0x15
DEVICE_REG_MODE1 = 0x00
DEVICE_REG_OUT0 = 0x1d
# in ein Register schreiben
bus.write_byte_data(DEVICE_ADDRESS,  DEVICE_REG_
MODE1, 0x80)
# in ein Register-Array schreiben
out_values = [0xff, 0xff, 0xff, 0xff, 0xff, 0xff]
bus.write_i2c_block_data(DEVICE_ADDRESS, DEVICE_
REG_LEDOUT0, out_values)
```

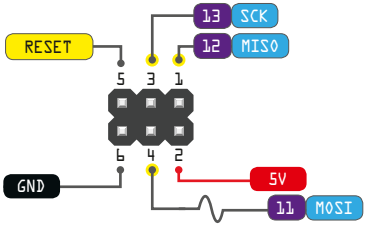

GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pon
Serial Pin
IDE
IOREF
RESET
PCINT14
1
PC6
3.3V
5V
GND
GND
VIN
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ADC0
PCINT8
14
A0
23
PC0
ADC1
PCINT9
15
A1
24
PC1
ADC2
PCINT10
16
A2
25
PC2
ADC3
PCINT11
17
A3
26
PC3
SDA
ADC4
PCINT12
18
A4
27
PC4
SCL
ADC5
PCINT13
19
A5
28
PC5
A0
A1
A2
A3
A4
A5
ANALOG IN
ICSP
RESET
PCINT14
1
PC6
GND


RESET
ICSP2
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX0 ▶ 1
RX0 ◀ 0
DIGITAL (PWM=~)
TX
RX
L
ON
Arduino
UNO
28 PC5 19 A5 PCINT13 ADC5 SCL
27 PC4 18 A4 PCINT12 ADC4 SDA
21 AREF AREF
GND
19 PB5 13 PCINT5 SCK
18 PB4 12 PCINT4 MISO
17 PB3 11 OC2A PCINT3 PWM MOSI
16 PB2 10 OC1B PCINT2 PWM SS
15 PB1 9 OC1A PCINT1 PWM
14 PB0 8 CLK0 PCINT0 ICP1
13 PD7 7 AIN1 PCINT23
12 PD6 6 AIN0 PCINT22 PWM OC0A
11 PD5 5 T1 PCINT21 PWM
6 PD4 4 T0 PCINT20 XCK
5 PD3 3 INT1 PCINT19 PWM OC2B
4 PD2 2 INT0 PCINT18
3 PD1 1 TXD PCINT17 TX
2 PD0 0 RXD PCINT16 RX
19 PB5 13 PCINT5 SCK
18 PB4 12 PCINT4 MISO
5V
17 PB3 11 OC2A PCINT3 PWM MOSI

PWM-Pin mit Pulsweiten-
modulation zur Ausgabe
digitaler Frequenzen

Der Strom für jede Gruppe
dieser Pins sollte 100mA
nicht überschreiten





USB-BUCHSE
MINI TYP B

PINBEZEICHNUNG AM BOARD

PINBEZEICHNUNG AN DER CPU

30	VIN	EINGANGS-SPANNUNG 7-12V =						
29	GND							
28	RESET		29	PC6	RESET	PCINT14		
27	5V							
26	A7	NUR ANALOGE EINGÄNGE	22		ADC7			
25	A6		19		ADC6			
24	DIG:19	A5 I2C:SCL	28	PC5	ADC5	PCINT13		
23	DIG:18	A4 I2C:SDA	27	PC4	ADC4	PCINT12		
22	DIG:17	A3	26	PC3	ADC3	PCINT11		
21	DIG:16	A2	25	PC2	ADC2	PCINT10		
20	DIG:15	A1	24	PC1	ADC1	PCINT9		
19	DIG:14	A0	23	PC0	ADC0	PCINT8		
18	AREF	ANALOGE REFERENZ-SPANNUNG	20		AREF			
17	3V3							
16	LED	DIG:13 SPI:SCK	17	PB5		PCINT5		

Arduino: Programmieren in C

Struktur und Kontrolle

```
void setup () {}
void loop () {}
if (a > b) {x = a;}
       else {x = b;}
for (i = 0; i <= x; i++) {}
while (a < b) {}
switch (x) {
       case 1 : ; break;
       [default: ;] }
```

Bedingungsoperator:

```
x = (a > b) ? a:b;
```

I/O

```
pinMode (pin, [INPUT|OUTPUT|INPUT_PULLUP]);
digitalWrite (pin, [HIGH|LOW]);
x = digitalRead (pin); // 0|1|LOW|HIGH
analogWrite (pin, [0..255]); // PWM
x = analogRead (pin); // 0..1023
```

Seriell

```
Serial.begin   ([300|600|1200|2400|4800|9600|14400|
19200|28800|38400|57600|115200]); // 8N1
x = Serial.available ();
Serial.write ((byte)x);
Serial.print|println („String“|‘C‘);
Serial.print|println ((int)x, HEX); // BIN|OCT|DEC
Serial.print|println ((float)x [, Dezimalstellen]);
x = Serial.read ();
```

Servo

```
#include <Servo.h>
Servo s;
s.attach (pin);
s.write ([0..180]); // Grad
s.writeMicroseconds (int); // üblich: 1000..2000
```

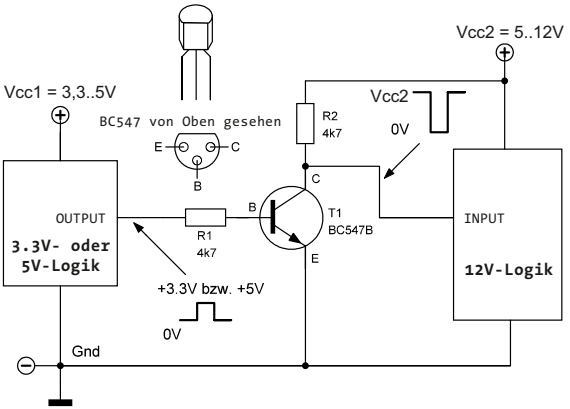
SPI (MISO, MOSI, CLK, CS)

```
#include <SPI.h>
pinMode (CS_Pin, OUTPUT);
SPI.beginTransaction (SPISettings ([MHz],
       [MSBFIRST|LSBFIRST], [SPI_MODE0|
       SPI_MODE1|SPI_MODE2|SPI_MODE3]));
SPI.begin ();
digitalWrite (CS_Pin, LOW);
SPI.transfer (address);
SPI.transfer (data);
digitalWrite (CS_Pin, HIGH);
```

I2C/TWI (SDA, SCL)

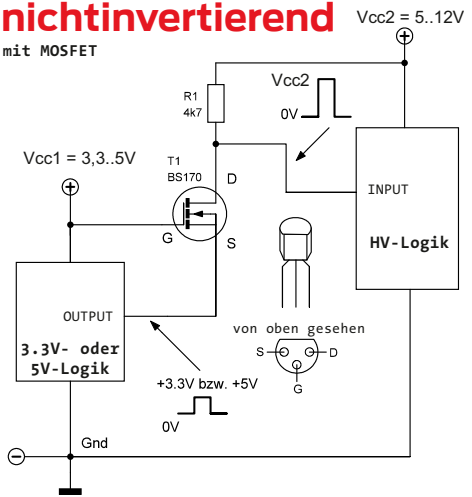
```
#include <Wire.h>
Wire.begin ();
Wire.beginTransmission (address);// 7 Bit Adresse
Wire.write (x);
Wire.requestFrom (address, len);
while (Wire.available ())
       x = Wire.read ();
Wire.endTransmission ();
```

Pegelwandler invertierend



nichtinvertierend

mit MOSFET

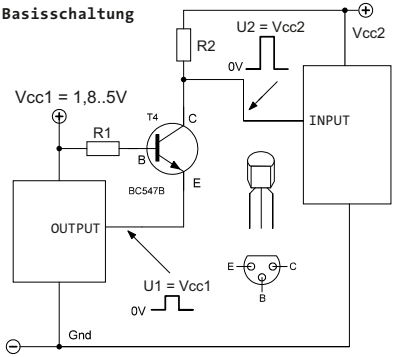


R1 ist an die Ausgangsspannung Vcc2 anzupassen!

Vcc2	R1
5V	2k2
9V	3k3
12V	4k7

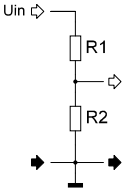
Pegelwandler nichtinvertierend

mit Bipolar-Transistor in Basisschaltung



Vcc1	R1
1,8V	1k
3,3V	2k2
5V	4k7

Vcc2	R2
3,3V	470R
5V	1k
9V	2k2
12V	4k7
24V	10k

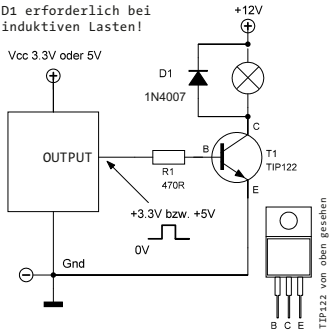


Uin	R1	R2
5V	470R	1k
5V	1k	2k2
5V	2k2	4k7
9V	4k7	2k2
9V	8k2	4k7
12V	3k3	1k
12V	6k8	2k2

Eingangs-Spannungsteiler für 3,3V-Logik (RasPi)

Gängige R-Werte, Kombinationen

Bipolare Schaltstufe 5A



MOSFET-Schaltstufe 10A