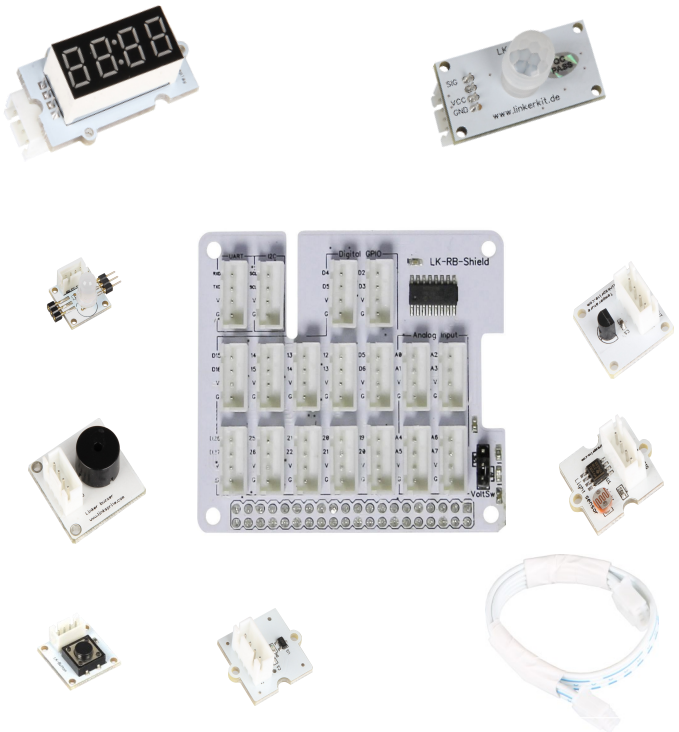
LINKERKIT BASE SET

für Raspberry Pi



1. ALLGEMEINE INFORMATIONEN

Sehr geehrter Kunde,
vielen Dank, dass Sie sich für unser Produkt entschieden haben. Im Folgenden zeigen wir Ihnen, was bei der Inbetriebnahme und der Verwendung zu beachten ist.

Sollten Sie während der Verwendung unerwartet auf Probleme stoßen, so können Sie uns selbstverständlich gerne kontaktieren.

Das Linker Kit Base Set für den Raspberry Pi macht die Verwendung verschiedener Module und Sensoren sehr einfach. Durch das Linker Kit Stecksystem ist eine Verpolung der einzelnen Module ausgeschlossen.

In dieser Anleitung finden Sie detaillierte Erklärungen und Beispiele zu allen in diesem Set enthaltenden Module.

Probieren sie es aus und erweitern sie die Beispiele nach Ihren eigenen Vorstellungen.

Im Lieferumfang enthalten sind:

Baseboard für Raspberry Pi, Buttonmodul, Buzzermodul,

5x Kabel, Digitalanzeigemodul, Hallsensor, RGB LED Modul,

Lichtsensor, Bewegungssensor, Temperatursensor

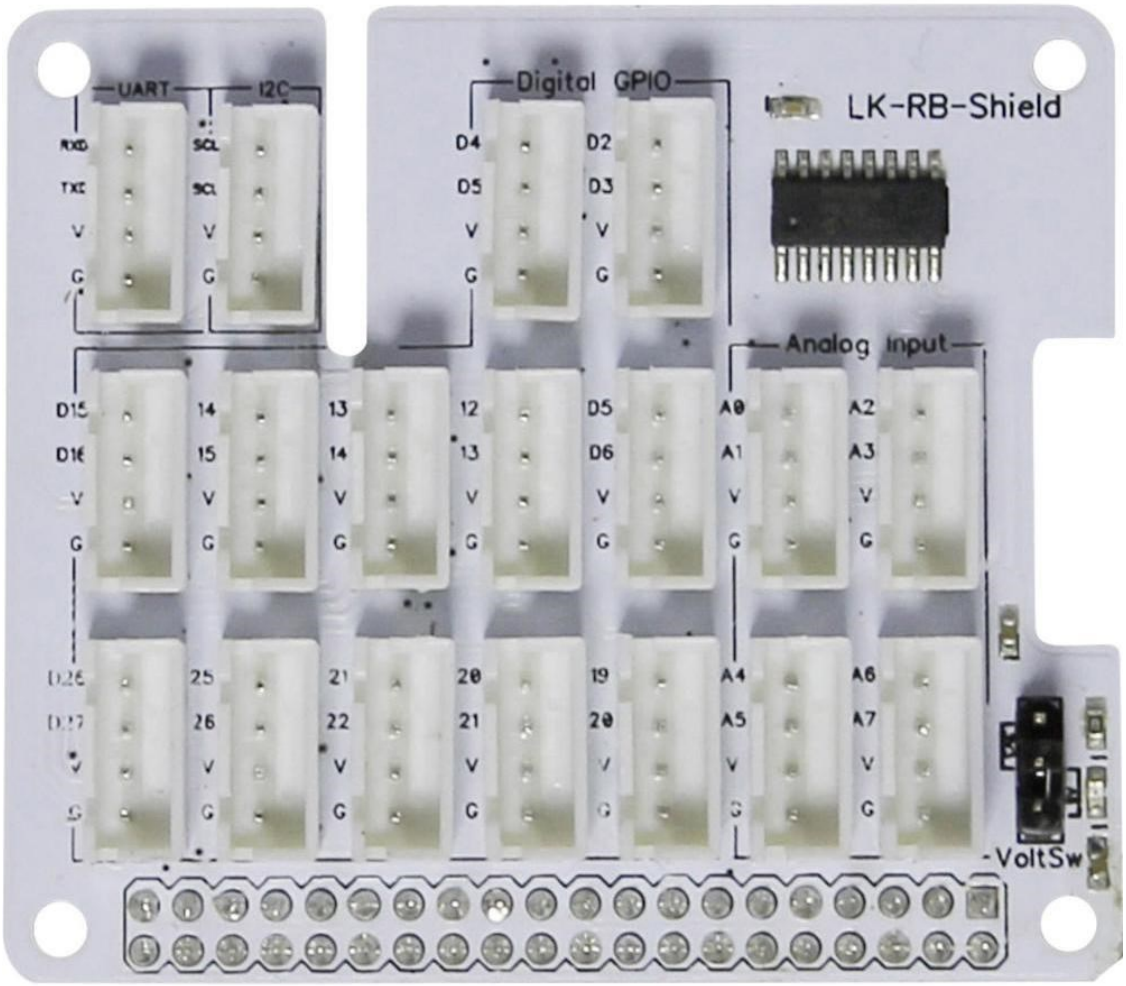
2. DAS BASEBOARD

Das Baseboard ist das Herzstück dieses Sets, es wird einfach auf Ihren Raspberry Pi gesteckt und ist sofort einsatzbereit.

Es können insgesamt 18 Module an das Board angeschlossen werden.

Neben den zwei bekannten UART und I2C können noch 14 weitere digitale und 4 Analoge Module
angeschlossen werden.

Die Betriebsspannung kann auf entweder 3.3V oder auf 5V eingestellt werden.



2. DAS BASEBOARD

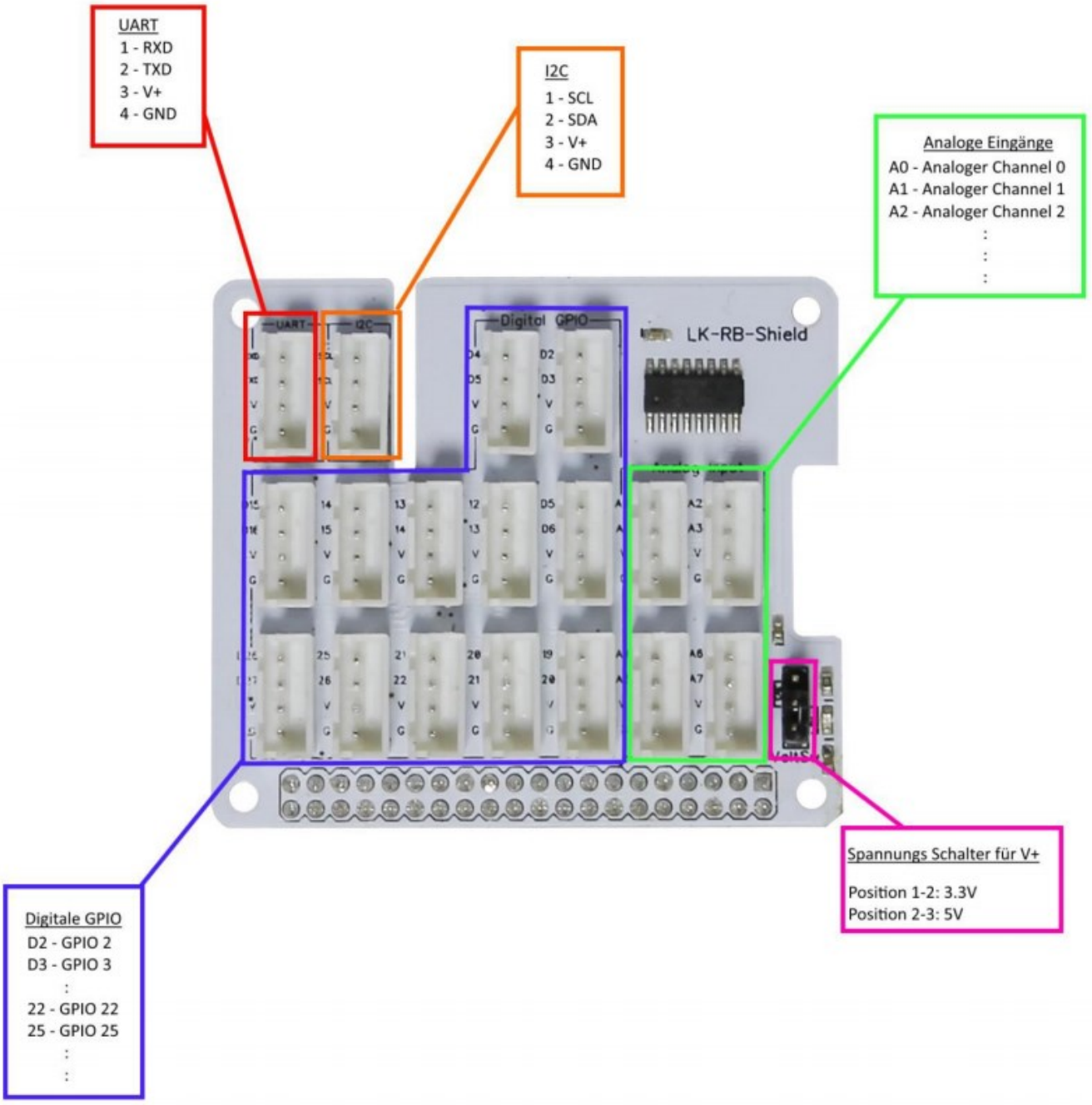
Das Baseboard ist das Herzstück dieses Sets, es wird einfach auf Ihren Raspberry Pi gesteckt und ist sofort einsatzbereit.

Es können insgesamt 18 Module an das Board angeschlossen werden.

Neben den zwei bekannten UART und I2C können noch 14 weitere digitale und 4 Analoge Module
angeschlossen werden.

Die Betriebsspannung kann auf entweder 3.3V oder auf 5V eingestellt werden.

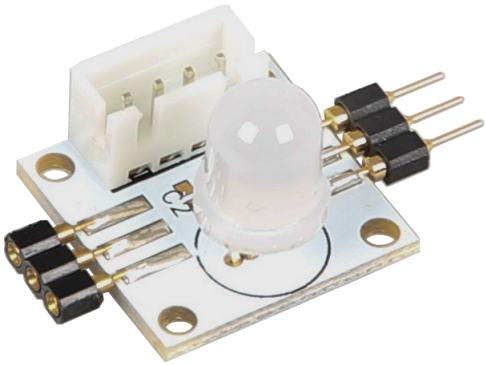
Dem nachfolgendem Bild können Sie einen detaillierten Anschlussplan für das Baseboard entnehmen

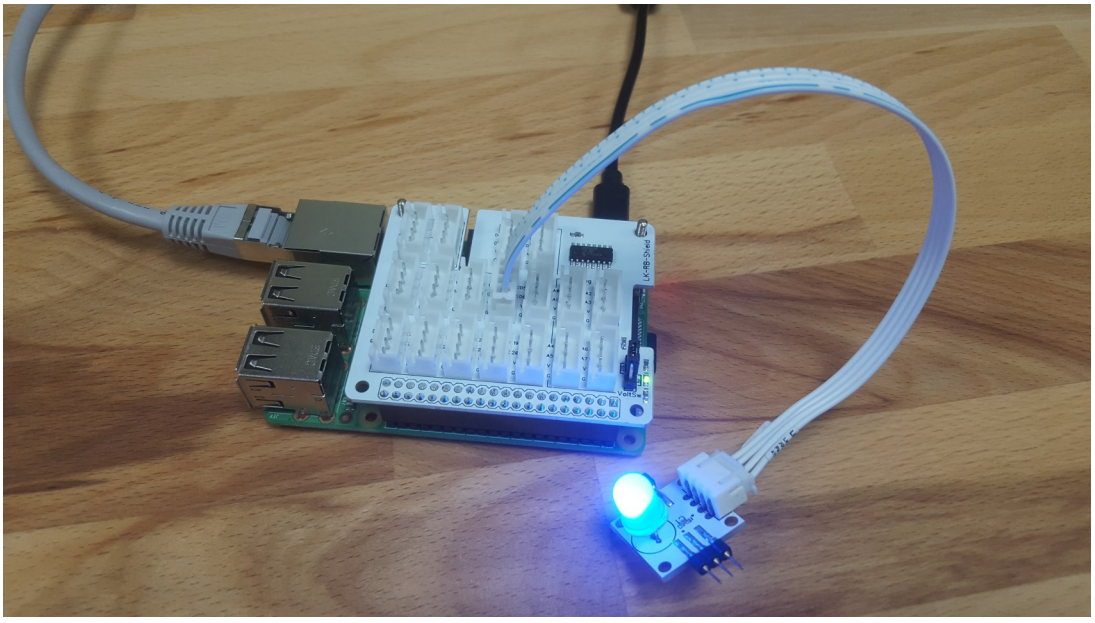


3. DIE RGB-LED

Egal ob zur Beleuchtung oder zur Statusanzeige die LED gehört in jedes Set.

Mehrere LEDs lassen sich auch zusammen stecken und gemeinsam ansteuern.





Die LED kann in vielen verschiedenen Farben leuchten.

In diesem Beispiel zeigen wir wie Sie die LED leuchten lassen und Ihre Farbe automatisch ändern können. Dies geht mit der Neopixel Bibliothek ganz einfach.

Diese müssen wir jedoch zunächst herunterladen, zusätzlich laden wir auch direkt die Beispieldatei herunter.

Geben sie folgende Befehle ins Terminal ein:

```
sudo apt-get update
sudo apt-get install build-essential python-dev git scons swig
git clone https://github.com/jgarff/rpi_ws281x
cd rpi_ws281x
sudo apt-get install scons
sudo scons
cd python
sudo python ez_setup.py
sudo python setup.py install
cd examples/
```

Nun müssen Sie die Beispieldatei mit folgendem Befehl bearbeiten:

```
sudo nano strandtest.py
```

Bearbeiten Sie folgende zwei Zeilen:

Ändern Sie sie folgendermaßen

```
LED_COUNT = 1
LED_PIN = 12
```

Jetzt können sie die Datei mit dem Befehl

```
sudo python strandtest.py
```

ausführen.

Schließen Sie dafür die LED an Digital-Pin 12 an.

Auf den folgenden 3 Seiten ist der Beispielcode noch einmal in Textform:

```
#!/usr/bin/env python3
# NeoPixel library strandtest example
# Author: Tony DiCola (tony@tonydicola.com)
#
# Direct port of the Arduino NeoPixel library strandtest example.  Showcases
# various animations on a strip of NeoPixels.

import time
from neopixel import *
import argparse

# LED strip configuration:
LED_COUNT      = 16      # Number of LED pixels.
LED_PIN        = 18      # GPIO pin connected to the pixels (18 uses PWM!).
#LED_PIN        = 10      # GPIO pin connected to the pixels (10 uses SPI /dev/
spidev0.0).
LED_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
LED_DMA        = 10      # DMA channel to use for generating signal (try 10)
LED_BRIGHTNESS = 255     # Set to 0 for darkest and 255 for brightest
LED_INVERT     = False   # True to invert the signal (when using NPN transistor level
shift)
LED_CHANNEL    = 0       # set to '1' for GPIOs 13, 19, 41, 45 or 53



# Define functions which animate LEDs in various ways.
def colorWipe(strip, color, wait_ms=50):
    """Wipe color across display a pixel at a time."""
    for i in range(strip.numPixels()):
        strip.setPixelColor(i, color)
        strip.show()
        time.sleep(wait_ms/1000.0)

def theaterChase(strip, color, wait_ms=50, iterations=10):
    """Movie theater light style chaser animation."""
    for j in range(iterations):
        for q in range(3):
            for i in range(0, strip.numPixels(), 3):
                strip.setPixelColor(i+q, color)
            strip.show()
            time.sleep(wait_ms/1000.0)
            for i in range(0, strip.numPixels(), 3):
                strip.setPixelColor(i+q, 0)


def wheel(pos):
  """Generate rainbow colors across 0-255 positions."""
  if pos < 85:
    return Color(pos * 3, 255 - pos * 3, 0)
  elif pos < 170:
    pos -= 85
    return Color(255 - pos * 3, 0, pos * 3)
  else:
    pos -= 170
    return Color(0, pos * 3, 255 - pos * 3)
```

Beispielcode fortgeführt:

```
def rainbow(strip, wait_ms=20, iterations=1):
    """Draw rainbow that fades across all pixels at once."""
    for j in range(256*iterations):
        for i in range(strip.numPixels()):
            strip.setPixelColor(i, wheel((i+j) & 255))
        strip.show()
        time.sleep(wait_ms/1000.0)

def rainbowCycle(strip, wait_ms=20, iterations=5):
    """Draw rainbow that uniformly distributes itself across all pixels."""
    for j in range(256*iterations):
        for i in range(strip.numPixels()):
            strip.setPixelColor(i, wheel((int(i * 256 / strip.numPixels()) + j) & 255))
        strip.show()
        time.sleep(wait_ms/1000.0)

def theaterChaseRainbow(strip, wait_ms=50):
    """Rainbow movie theater light style chaser animation."""
    for j in range(256):
        for q in range(3):
            for i in range(0, strip.numPixels(), 3):
                strip.setPixelColor(i+q, wheel((i+j) % 255))
            strip.show()
            time.sleep(wait_ms/1000.0)
            for i in range(0, strip.numPixels(), 3):
                strip.setPixelColor(i+q, 0)

# Main program logic follows:
if __name__ == '__main__':
    # Process arguments
    parser = argparse.ArgumentParser()
    parser.add_argument('-c', '--clear', action='store_true', help='clear display on exit')
    args = parser.parse_args()

    # Create NeoPixel object with appropriate configuration.
    strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT,
LED_BRIGHTNESS, LED_CHANNEL)
    # Intialize the library (must be called once before other functions).
    strip.begin()

    print ('Press Ctrl-C to quit.')
    if not args.clear:
        print('Use "-c" argument to clear LEDs on exit')
```

Beispielcode fortgeführt:

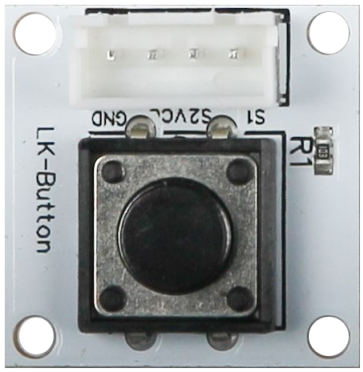
```
        try:

            while True:
                print ('Color wipe animations.')
                colorWipe(strip, Color(255, 0, 0))  # Red wipe
                colorWipe(strip, Color(0, 255, 0))  # Blue wipe
                colorWipe(strip, Color(0, 0, 255))  # Green wipe
                print ('Theater chase animations.')
                theaterChase(strip, Color(127, 127, 127))  # White theater chase
                theaterChase(strip, Color(127,   0,   0))  # Red theater chase
                theaterChase(strip, Color(  0,   0, 127))  # Blue theater chase
                print ('Rainbow animations.')
                rainbow(strip)
                rainbowCycle(strip)
                theaterChaseRainbow(strip)

        except KeyboardInterrupt:
            if args.clear:
                colorWipe(strip, Color(0,0,0), 10)
```

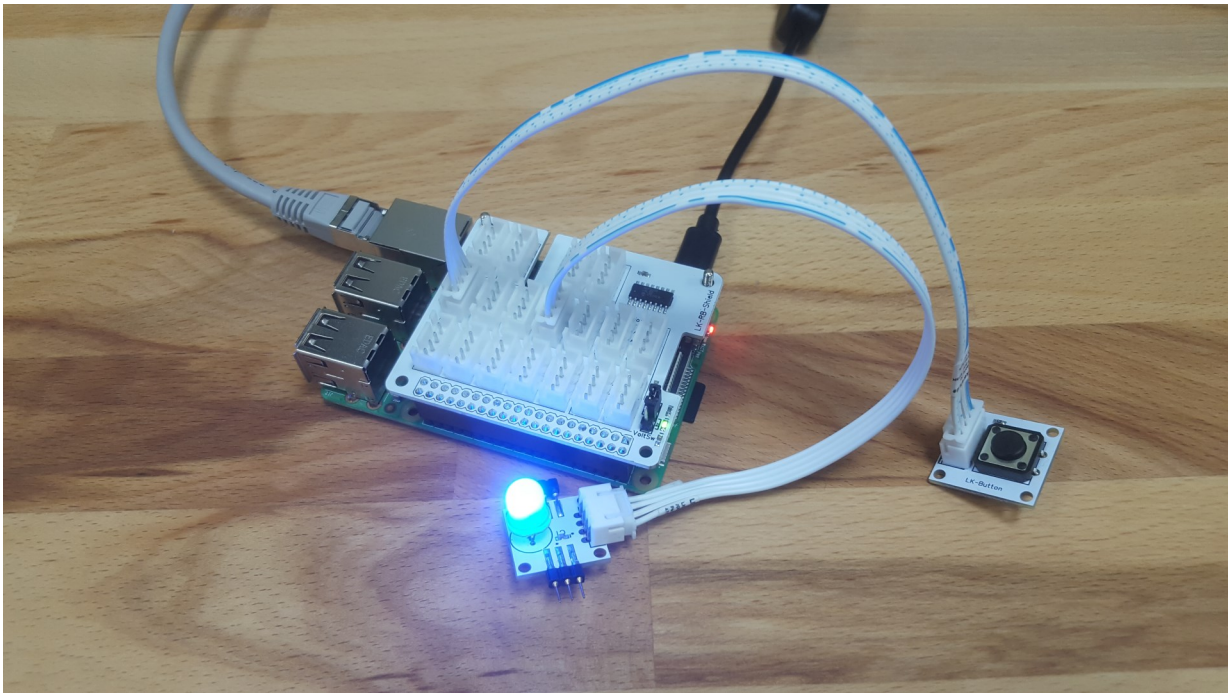
Führen Sie den Beispielcode mit folgenden Befehlen im Terminal aus:

```
cd /rpi_ws281x/python/examples
sudo python strandtest.py
```

4. DAS BUTTONMODUL

Das Buttonmodul eignet sich hervorragend zum steuern anderer Module und kann für viele verschiedene Aufgaben verwendet werden.





In diesem Beispiel kombinieren wir die bereits kennengelernte LED mit dem Buttonmodul.

Wir werden diesmal die LED mittels Knopfdruck zum Leuchten bringen.

Schließen Sie die LED an Digital-Pin 12 und den Button an Digital-Pin 15 an.

Bitte kopieren sie den folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.

Bitte beachten Sie, dass Sie das Skript im selben Verzeichnis speichern müssen indem sich die anderen Beispieldateien der LED befindet:

home/pi/rpi_ws281x/python/examples

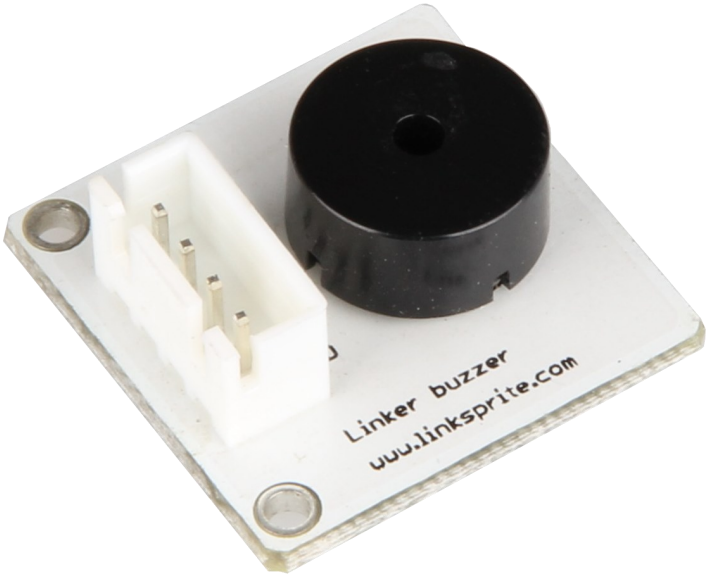
```
from neopixel import *
import RPi.GPIO as GPIO
import time
import argparse
#LED Strip configuration
LED_COUNT = 1 #Number of LED pixels
LED_PIN = 12 #GPIO Pin connected to LED
LED_FREQ_HZ = 800000
LED_DMA = 10
LED_BRIGHTNESS = 255
LED_INVERT = False
LED_CHANNEL = 0
BUTTON = 15

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup (LED_PIN, GPIO.OUT)

def colorWipe(strip, color, wait_ms=50):
     for i in range (strip.numPixels()):
          strip.setPixelColor(i, color)
          strip.show()
          time.sleep(wait_ms/1000.0)

if __name__ == '__main__':
     parser = argparse.ArgumentParser()
     parser.add_argument('-c', '--clear', action='store_true', help='clear the dis-
play on exit')
     args = parser.parse_args()

strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT,
LED_BRIGHTNESS, LED_CHANNEL)
strip.begin()

while True:
     if GPIO.input(BUTTON) == GPIO.HIGH:
          colorWipe(strip, Color(255, 0, 0), 50)
     else:
          colorWipe(strip, Color(0, 0, 0), 50)
```

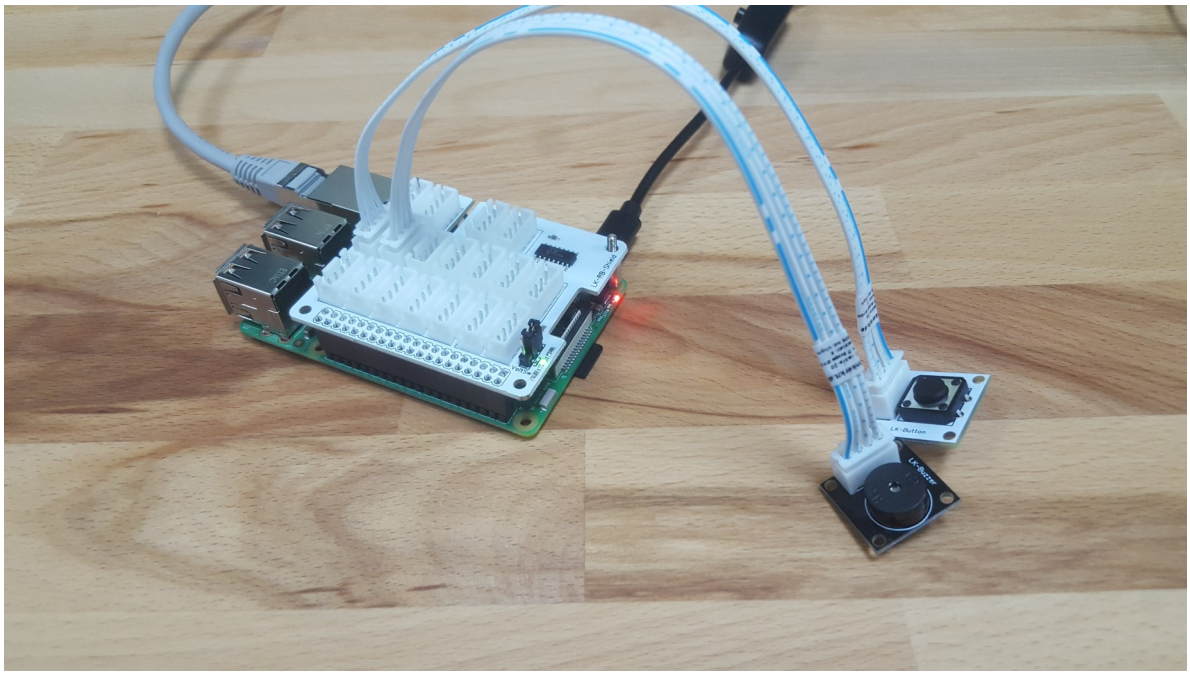
Führen Sie den Beispielcode mit folgenden Befehlen im Terminal aus:

```
cd /rpi_ws281x/python/examples
sudo python Button.py
```

5. DAS BUZZWEMODUL

Dieser Buzzer kann an Digitale Ausgänge angeschlossen werden, um einen Ton zu erzeugen. Alternativ kann er an einem Analogen Impulsbreitenmodulationsausgang angeschlossen werden um unterschiedliche Töne und Effekte zu erzeugen





In diesem Beispiel nehmen wir das das Buttonmodul und kombinieren es mit dem Buzzermodul, sodass der Buzzer, bei jedem drücken des Knopfes ertönt.

Der Buzzer wird in diesem Beispiel an dem Steckplatz mit dem digitalen Port 14 und der Button an den mit dem digitalen Port 15 angeschlossen.

Bitte kopieren sie den folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.

Speichern Sie die Datei am Besten in Ihrem Dokumente Ordner unter dem Namen Buzzer.py

```
import RPi.GPIO as GPIO
from time import sleep

buzzer = 14
button = 15

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(buzzer, GPIO.OUT)

while True:
   if GPIO.input(button) == GPIO.HIGH:
        GPIO.output(buzzer, True)
        sleep(0.0005)
        GPIO.output(buzzer, False)
                       sleep(0.0005)
```

Führen Sie den Beispielcode mit folgenden Befehlen im Terminal aus:

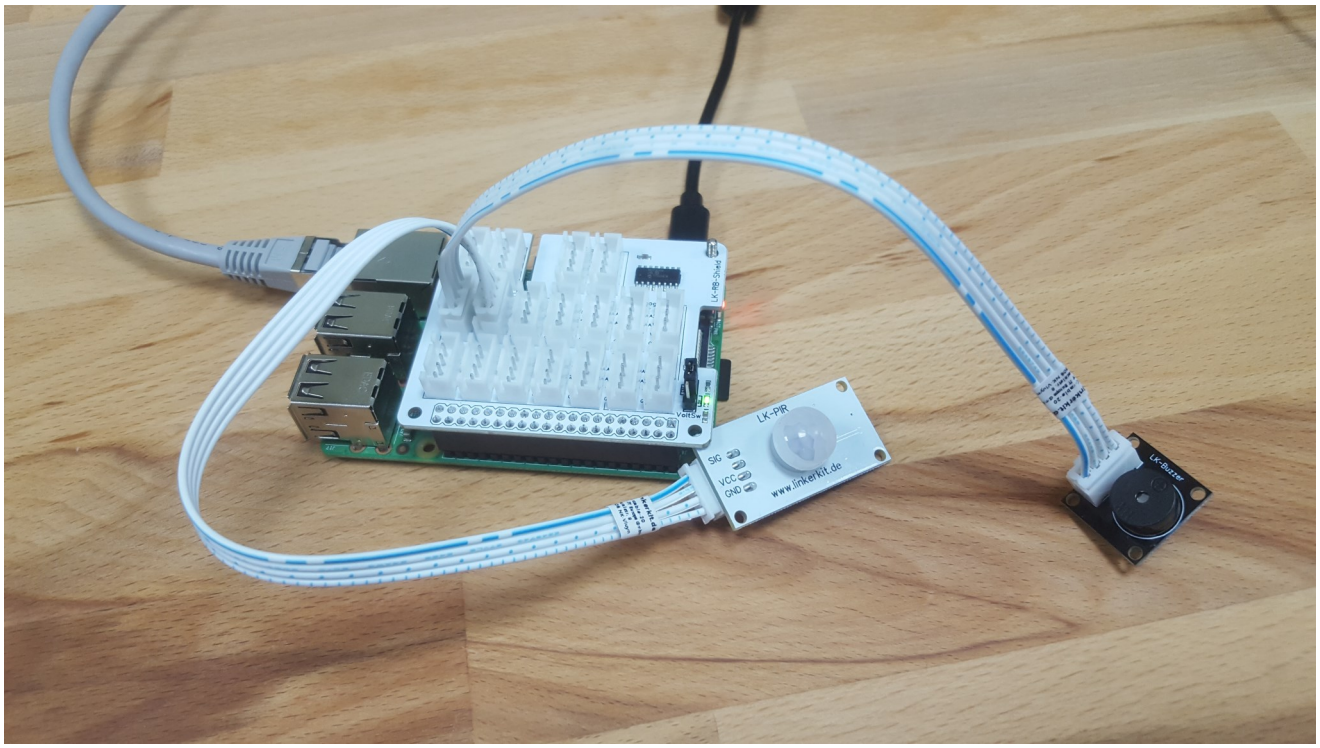
```
cd /Documents
sudo python Buzzer.py
```

6. DER BEWEGUNGSSSENSOR

Dieser Bewegungssensor hat einen Erfassungswinkel von 120° und eine Reichweite von bis zu 6 Metern.

Mit diesem Sensor kann man zum Beispiel Lampen bauen, die sich automatisch einschalten wenn man den Raum betritt.

Bewegungssensoren werden auch oft in Alarmanlagen eingesetzt.



In diesem Beispiel Kombinieren wir den Bewegungssensor mit dem Buzzer, sodass sobald das Modul eine Bewegung erkennt ein Alarmton ausgegeben wird.

Verbinden Sie den Bewegungssensor mit Digital-Pin 14 und den Buzzer mit Digital Pin 15.

Bitte kopieren sie den folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.

Speichern Sie die Datei am Besten in Ihrem Dokumente Ordner unter dem Namen Pir.py

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

#Initialisierung des PIR-Sensors auf PIN 12 und der LED auf PIN 15
PIR = 14
buz = 15

GPIO.setup(PIR, GPIO.IN)
GPIO.setup(buz, GPIO.OUT)

print ("PIR-Sensor aktiv!")

def ausgabeFunktion(null):
  i = 1
    print("Bewegung erkannt")
    while i <= 10:
        GPIO.output(buz,True)
        time.sleep(0.0005)
        GPIO.output(buz,False)
        time.sleep(0.0005)
        i += 1
    while (GPIO.input(PIR)):
        time.sleep(1)


#Ausfuehren der Ausgabefunktion beim Signalanstieg durch PIR-Sensor
GPIO.add_event_detect(PIR, GPIO.RISING, callback=ausgabeFunktion,
bouncetime=100)

while True:
    time.sleep(1)
```

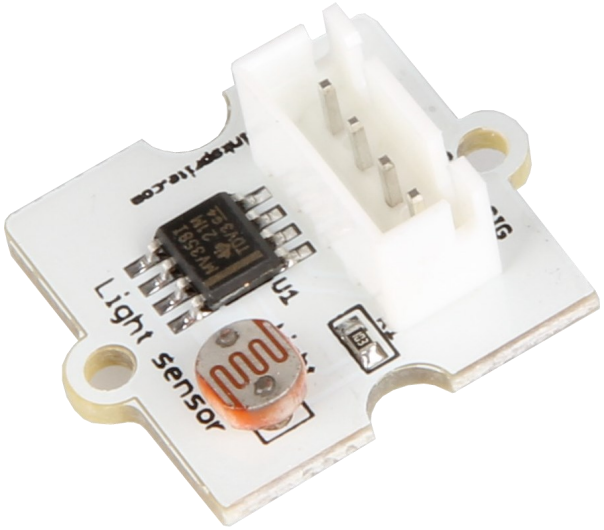
Führen Sie den Beispielcode mit folgenden Befehlen im Terminal aus:

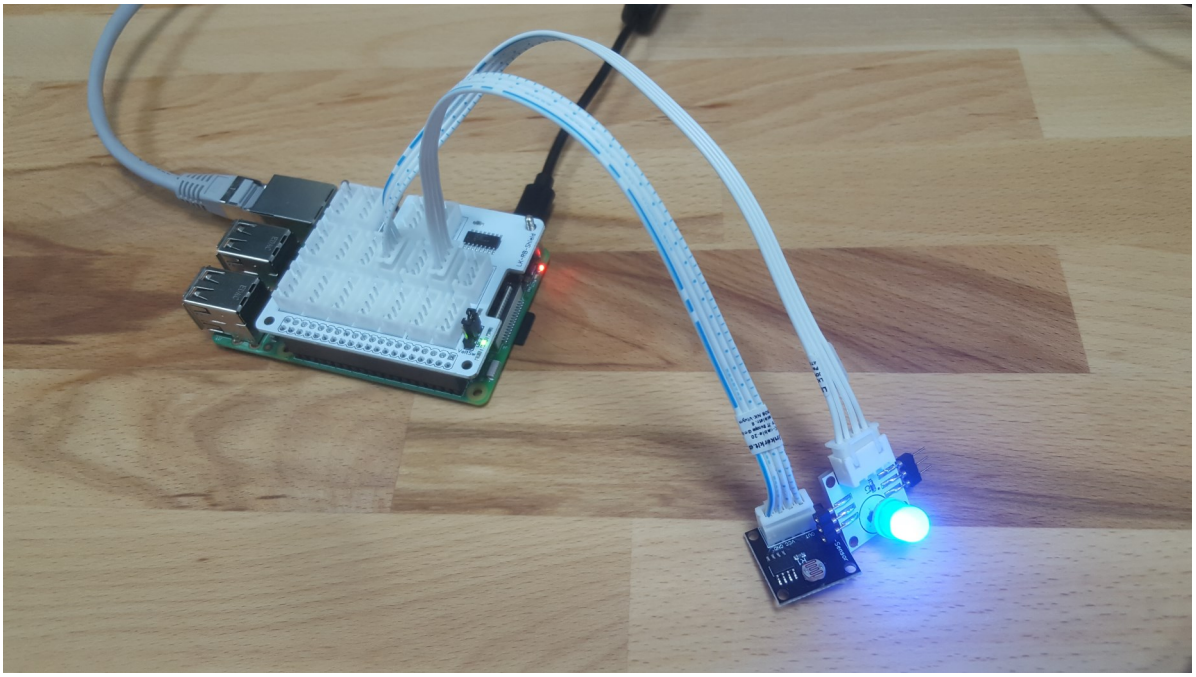
```
cd /Documents
sudo python Pir.py
```

7. DER LICHTSENSOR

Dieser Lichtsensor ist ein vom Licht abhängiger Widerstand (LDR). Der Widerstand des Sensors verringert sich, sobald die Helligkeit in der Umgebung zu nimmt.

Mit Hilfe dieses Sensors kann man eine Lampe bauen, die automatisch an geht sobald es dunkel wird.





In diesem Beispiel kombinieren wir den Lichtsensor mit der LED, um die LED einzuschalten sobald es dunkel wird. Um Dunkelheit zu simulieren können sie den Lichtsensor einfach mit der Hand abdecken.

Schließen Sie die LED an Digital-Pin 12 und den Lichtsensor an Analog-Pin 0 an.

Bitte kopieren sie den auf den nächsten zwei Seiten folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.

Speichern Sie das Skript bitte im selben Verzeichnis in dem sich die Beispieldateien für die LED befinden unter dem Namen Lichtsensor.py

/rpi_ws281x/python/examples

```
from neopixel import *
import RPi.GPIO as GPIO
import time
import argparse
import spidev
#LED Strip configuration
LED_COUNT = 1 #Number of LED pixels
LED_PIN = 12 #GPIO Pin connected to LED
LED_FREQ_HZ = 800000
LED_DMA = 10
LED_BRIGHTNESS = 255
LED_INVERT = False
LED_CHANNEL = 0
SENSOR = 0

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
#GPIO.setup(BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup (LED_PIN, GPIO.OUT)

spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz = 1350000

def readadc(adcnum):
# SPI-Daten auslesen
    r = spi.xfer2([1,8+adcnum <<4,0])
    adcout = ((r[1] &3) <<8)+r[2]
    return adcout


def colorWipe(strip, color, wait_ms=50):
  for i in range (strip.numPixels()):
       strip.setPixelColor(i, color)
       strip.show()
       time.sleep(wait_ms/1000.0)
```

Beispielcode fortgeführt.

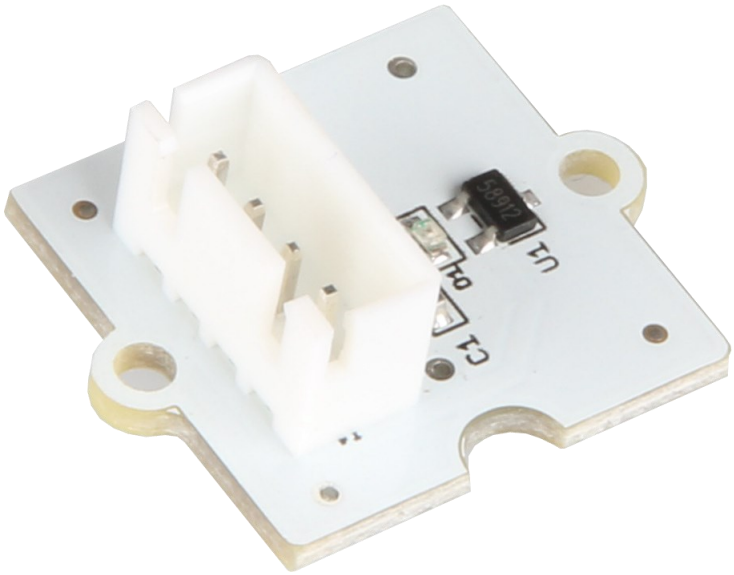
```
if __name__ == '__main__':
  parser = argparse.ArgumentParser()
  parser.add_argument('-c', '--clear', action='store_true', help='clear the dis-
play on exit')
  args = parser.parse_args()

strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT,
LED_BRIGHTNESS, LED_CHANNEL)
strip.begin()

while True:
        value = readadc(SENSOR)
        print("Value: " + str(value))
        time.sleep(0.5)
        if int(value) < int(500):
            print ("Licht an")
            colorWipe(strip, Color(255, 0, 0), 50)
        else:
            print ("Licht aus")
                        colorWipe(strip, Color(0, 0, 0), 50)
```

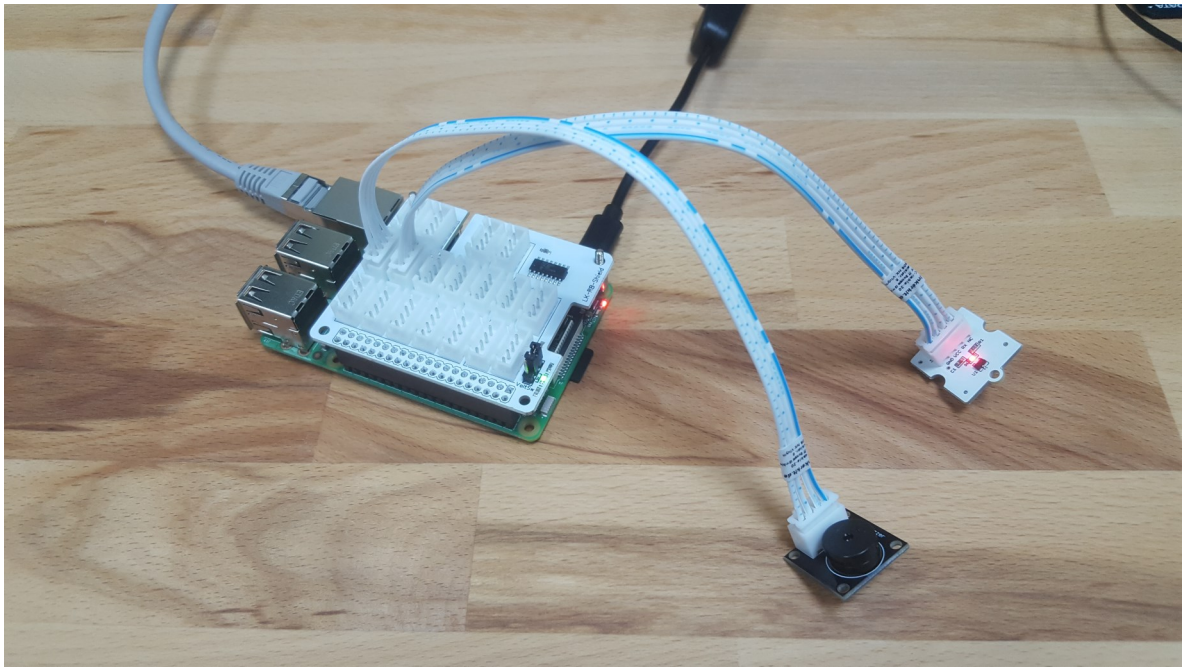
Führen Sie den Beispielcode mit folgenden Befehlen im Terminal aus:

```
cd /rpi_ws281x/python/examples
sudo python Lichtsensor.py
```

8. DER HALLSENSOR

Mit diesem Hallsensor können Sie Magnetfelder in der Umgebung, mit Hilfe des Hall-Effekts, erfassen.





In diesem Beispiel kombinieren wir den Hallsensor und den Buzzer. Wenn der Sensor ein Magnetfeld erkennt wird der Buzzer ein Signal von sich geben.

Verbinden Sie den Hallsensor mit Digital-Pin 14 und den Buzzer mit Digital-Pin 15

Bitte kopieren sie den folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.

Speichern Sie die Datei am Besten in Ihrem Dokumente Ordner unter dem Namen Hall.py

```
import RPi.GPIO as GPIO
from time import sleep

#Initialisiere LED auf PIN15 und Sensor auf PIN14
buz = 15
sensor = 14


GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(buz, GPIO.OUT)
GPIO.setup(sensor, GPIO.IN)

i = 1

while True:
   if (GPIO.input(sensor) == GPIO.HIGH):
      while i <= 10:
            GPIO.output(buz,True)
            time.sleep(0.0005)
            GPIO.output(buz,False)
            time.sleep(0.0005)
            i += 1
      print ("Magnetfeld erkannt")
      i = 1
   else:

      print ("Magnetfeld nicht erkannt")
```

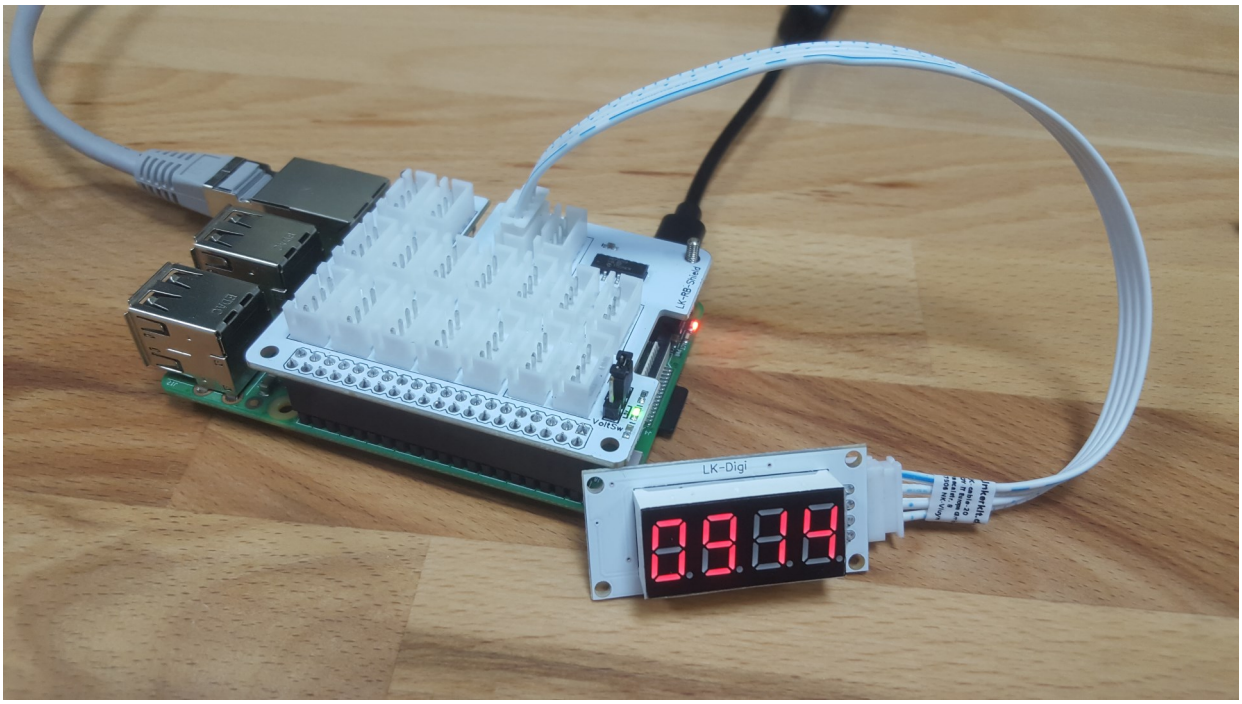
Führen Sie den Beispielcode mit folgenden Befehlen im Terminal aus:

```
cd /Documents
sudo python Hall.py
```

9. DIE DIGITALANZEIGE

Mit Hilfe der Digitalanzeige können wir uns die Zeit oder andere Daten anzeigen lassen.

Sie wird in vielen industriellen Lösungen wie Aufzügen verwendet! Und wird für Sie in der Zukunft mit Sicherheit von Nutzen sein.



In diesem Beispiel werden wir die aktuelle Uhrzeit auf der Digitalanzeige anzeigen lassen. Dafür müssen wir zunächst eine zusätzliche Konfigurations-Datei mit dem Namen „tm1637.py“ anlegen. In diese Datei kopieren wir folgenden Code. Der Code ist ziemlich umfangreich, deshalb haben wir ihn auf vier Seiten aufgeteilt.

Speichern Sie die Datei am Besten in Ihrem Dokumente Ordner unter dem Namen tm1637.py

```
import sys
import os
import time
import RPi.GPIO as IO

IO.setwarnings(False)
IO.setmode(IO.BCM)

HexDigits =
[0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39
,0x5e,0x79,0x71]

ADDR_AUTO = 0x40
ADDR_FIXED = 0x44
STARTADDR = 0xC0
BRIGHT_DARKEST = 0
BRIGHT_TYPICAL = 2
BRIGHT_HIGHEST = 7
OUTPUT = IO.OUT
INPUT = IO.IN
LOW = IO.LOW
HIGH = IO.HIGH

class TM1637:
    __doublePoint = False
    __Clkpin = 0
    __Datapin = 0
    __brightnes = BRIGHT_TYPICAL;
    __currentData = [0,0,0,0];

    def __init__( self, pinClock, pinData, brightnes ):
        self.__Clkpin = pinClock
        self.__Datapin = pinData
        self.__brightnes = brightnes;
        IO.setup(self.__Clkpin,OUTPUT)
        IO.setup(self.__Datapin,OUTPUT)
    # end  __init__
```

Code fortgeführt:

```
    def Clear(self):
        b = self.__brightnes;
        point = self.__doublePoint;
        self.__brightnes = 0;
        self.__doublePoint = False;
        data = [0x7F,0x7F,0x7F,0x7F];
        self.Show(data);
        self.__brightnes = b;               # restore saved
brightnes
        self.__doublePoint = point;
    # end  Clear

    def ShowInt(self, i):
        s = str(i)
        self.Clear()
        for i in range(0,len(s)):
            self.Show1(i, int(s[i]))

    def Show( self, data ):
        for i in range(0,4):
            self.__currentData[i] = data[i];

        self.start();
        self.writeByte(ADDR_AUTO);
        self.stop();
        self.start();
        self.writeByte(STARTADDR);
        for i in range(0,4):
            self.writeByte(self.coding(data[i]));
        self.stop();
        self.start();
        self.writeByte(0x88 + self.__brightnes);
        self.stop();
    # end  Show

    def SetBrightnes(self, brightnes):      # brightnes 0...7
        if( brightnes > 7 ):
            brightnes = 7;
        elif( brightnes < 0 ):
            brightnes = 0;
```

Code fortgeführt:

```
            if( self.__brightnes != brightnes):
                self.__brightnes = brightnes;
                self.Show(self.__currentData);
            # end if
        # end  SetBrightnes

        def ShowDoublepoint(self, on):          # shows or hides the
    doublepoint
            if( self.__doublePoint != on):
                self.__doublePoint = on;
                self.Show(self.__currentData);
            # end if
        # end  ShowDoublepoint

        def writeByte( self, data ):
            for i in range(0,8):
                IO.output( self.__Clkpin, LOW)
                if(data & 0x01):
                    IO.output( self.__Datapin, HIGH)
                else:
                    IO.output( self.__Datapin, LOW)
                data = data >> 1
                IO.output( self.__Clkpin, HIGH)
            #endfor

            # wait for ACK
            IO.output( self.__Clkpin, LOW)
            IO.output( self.__Datapin, HIGH)
            IO.output( self.__Clkpin, HIGH)
            IO.setup(self.__Datapin, INPUT)

            while(IO.input(self.__Datapin)):
                time.sleep(0.001)
                if( IO.input(self.__Datapin)):
                    IO.setup(self.__Datapin, OUTPUT)
                    IO.output( self.__Datapin, LOW)
                    IO.setup(self.__Datapin, INPUT)
                #endif
            # endwhile
            IO.setup(self.__Datapin, OUTPUT)
        # end writeByte
```

Code fortgeführt:

```
    def start(self):
        IO.output( self.__Clkpin, HIGH) # send start signal to
TM1637
        IO.output( self.__Datapin, HIGH)
        IO.output( self.__Datapin, LOW)
        IO.output( self.__Clkpin, LOW)
    # end start

    def stop(self):
        IO.output( self.__Clkpin, LOW)
        IO.output( self.__Datapin, LOW)
        IO.output( self.__Clkpin, HIGH)
        IO.output( self.__Datapin, HIGH)
    # end stop

    def coding(self, data):
        if( self.__doublePoint ):
            pointData = 0x80
        else:
            pointData = 0;

        if(data == 0x7F):
            data = 0
        else:
            data = HexDigits[data] + pointData;
        return data
    # end coding

# end class TM1637
```

Nun folgt der eigentliche Code um die Uhrzeit auf dem Display anzeigen zulassen.

Die Datei muss im selben Verzeichnis wie die Konfigurations-Datei gespeichert werden.

Der Display muss an Digital-Pin 4 angeschlossen werden.

Bitte kopieren sie den folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.

Speichern Sie die Datei am Besten in Ihrem Dokumente Ordner unter dem Namen Time.py

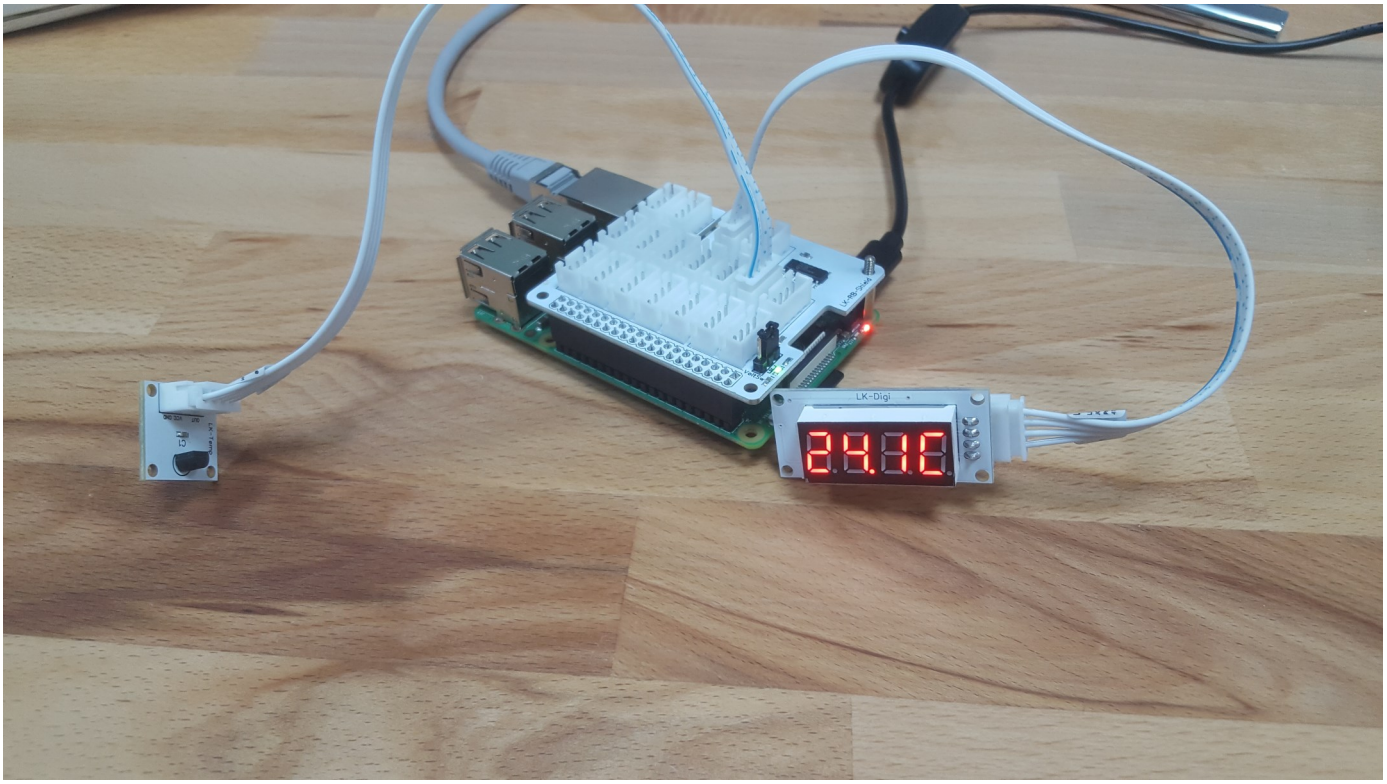
```
import sys
import time
import datetime
import RPi.GPIO as GPIO
import tm1637

Display = tm1637.TM1637(4,5,tm1637.BRIGHT_TYPICAL)

Display.Clear()
Display.SetBrightnes(1)

while(True):
    now = datetime.datetime.now()
    hour = now.hour
    minute = now.minute
    second = now.second
    currenttime = [ int(hour / 10), hour % 10, int(minute / 10), minute % 10 ]

    Display.Show(currenttime)
    Display.ShowDoublepoint(second % 2)

    time.sleep(1)
```

Führen Sie den Beispielcode mit folgenden Befehlen im Terminal aus:

```
cd /Documents
sudo python Time.py
```

10. DER TEMPERATURSENSOR

Dieser Temperatursensor verwendet einen Thermistor, um die Umgebungstemperatur zu erfassen. Der Widerstand des Thermistors erhöht sich, wenn die Umgebungstemperatur abnimmt. Durch diese Charakteristik wird die Umgebungstemperatur ausgemessen.



In diesem Beispiel kombinieren wir den Temperatursensor mit der Digitalanzeige um uns die aktuelle Raumtemperatur anzeigen zulassen. Auch diese Datei muss im selben Verzeichnis wie die Konfigurations-datei der Digitalanzeige gespeichert werden. Die Konfigurationsdatei muss jedoch auch angepasst werden, die angepasste Version finden Sie ab der nächsten Seite.

Der Temperatursensor wird an Analog-Pin 0 und Der Display an Digital-Pin 4 angeschlossen

Bitte kopieren sie den folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.

Speichern Sie die Datei zusammen mit der noch folgenden angepassten Konfigurationsdatei in einem
extra Ordner z.B. unter /Documents/Temperature unter dem Namen Temp.py

```
import time
import RPi.GPIO as GPIO
import tm1637
import spidev

temp = 0

spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz = 1350000

def readadc(adcnum):
# SPI-Daten auslesen
    r = spi.xfer2([1,8+adcnum <<4,0])
    adcout = ((r[1] &3) <<8)+r[2]
    return adcout

while True:
        value = readadc(temp)
        volts = (value * 3.3) / 1024
        temperature_C = (volts - 0.5) * 100 #Temperatur berechnen
        temperature_C = '%.1f' % temperature_C # Temperatur runden eine Stelle nach dem Komma
        print("Temperatur: " + temperature_C +  " C") # Temperatur ausgeben

        temp1 = temperature_C[-1:]  # letzte Zahl
        temp2 = temperature_C[-3:-2] # vorletzte Zahl
        temp3 = temperature_C[-4:-3] # erste Zahl

        Display = tm1637.TM1637(4,5,tm1637.BRIGHT_TYPICAL)  #Display vorbereiten
        Display.Clear()
        Display.SetBrightnes(7)

        temp2 = Display.coding(int(temp2)) + 128  # füge das Komma hinzu

        if temperature_C < str(0):
            Display.Show([0x40, Display.coding(int(temp3)), temp2, Display.coding(int
(temp1))])      #Minusgrade anzeigen
        else:
            Display.Show([Display.coding(int(temp3)), temp2, Display.coding(int(temp1)),
0x39])    #Temperatur anzeigen



        time.sleep(3)
```

Konfigurationsdatei tm1637.py fortgeführt.

```
    def ShowInt(self, i):
        s = str(i)
        self.Clear()
        for i in range(0,len(s)):
            self.Show1(i, int(s[i]))

    def Show( self, data ):
        for i in range(0,4):
            self.__currentData[i] = data[i];

        self.start();
        self.writeByte(ADDR_AUTO);
        self.stop();
        self.start();
        self.writeByte(STARTADDR);
        for i in range(0,4):
            self.writeByte(data[i]);
        self.stop();
        self.start();
        self.writeByte(0x88 + self.__brightnes);
        self.stop();
    # end  Show

    def SetBrightnes(self, brightnes):      # brightnes 0...7
        if( brightnes > 7 ):
            brightnes = 7;
        elif( brightnes < 0 ):
            brightnes = 0;

        if( self.__brightnes != brightnes):
            self.__brightnes = brightnes;
            self.Show(self.__currentData);
        # end if
    # end  SetBrightnes

    def ShowDoublepoint(self, on):          # shows or hides the doublepoint
        if( self.__doublePoint != on):
            self.__doublePoint = on;
            self.Show(self.__currentData);
        # end if
    # end  ShowDoublepoint

    def writeByte( self, data ):
        for i in range(0,8):
            IO.output( self.__Clkpin, LOW)
            if(data & 0x01):
                IO.output( self.__Datapin, HIGH)
            else:
                IO.output( self.__Datapin, LOW)
            data = data >> 1
            IO.output( self.__Clkpin, HIGH)
        #endfor
```

Konfigurationsdatei tm1637.py fortgeführt.

```
            # wait for ACK
            IO.output( self.__Clkpin, LOW)
            IO.output( self.__Datapin, HIGH)
            IO.output( self.__Clkpin, HIGH)
            IO.setup(self.__Datapin, INPUT)

            while(IO.input(self.__Datapin)):
                time.sleep(0.001)
                if( IO.input(self.__Datapin)):
                    IO.setup(self.__Datapin, OUTPUT)
                    IO.output( self.__Datapin, LOW)
                    IO.setup(self.__Datapin, INPUT)
                #endif
            # endwhile
            IO.setup(self.__Datapin, OUTPUT)
        # end writeByte

        def start(self):
            IO.output( self.__Clkpin, HIGH) # send start signal to TM1637
            IO.output( self.__Datapin, HIGH)
            IO.output( self.__Datapin, LOW)
            IO.output( self.__Clkpin, LOW)
        # end start

        def stop(self):
            IO.output( self.__Clkpin, LOW)
            IO.output( self.__Datapin, LOW)
            IO.output( self.__Clkpin, HIGH)
            IO.output( self.__Datapin, HIGH)
        # end stop

        def coding(self, data):
            if( self.__doublePoint ):
                pointData = 0x80
            else:
                pointData = 0;

            if(data == 0x7F):
                data = 0
            else:
                data = HexDigits[data] + pointData;
            return data
        # end coding

    # end class TM1637
```

Führen Sie den Beispielcode mit folgenden Befehlen im Terminal aus:

```
cd /Documents/Temperature
sudo python Temp.py
```

11. SONSTIGE INFORMATIONEN

Unsere Informations- und Rücknahmepflichten nach dem Elektrogesetz (ElektroG)

Symbol auf Elektro- und Elektronikgeräten:

Diese durchgestrichene Mülltonne bedeutet, dass Elektro- und Elektronikgeräte **nicht** in den Hausmüll gehören. Sie müssen die Altgeräte an einer Erfassungsstelle abgeben. Vor der Abgabe haben Sie Altbatterien und Altakkumulatoren, die nicht vom Altgerät umschlossen sind, von diesem zu trennen.

Rückgabemöglichkeiten:
Als Endnutzer können Sie beim Kauf eines neuen Gerätes, Ihr Altgerät (das im Wesentlichen die gleiche Funktion wie das bei uns erworbene neue erfüllt) kostenlos zur Entsorgung abgeben. Kleingeräte bei denen keine äußere Abmessungen größer als 25 cm sind können unabhängig vom Kauf eines Neugerätes in haushaltsüblichen Mengen abgeben werden.

Möglichkeit Rückgabe an unserem Firmenstandort während der Öffnungszeiten:
SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

Möglichkeit Rückgabe in Ihrer Nähe:
Wir senden Ihnen eine Paketmarke zu mit der Sie das Gerät kostenlos an uns zurücksenden können. Hierzu wenden Sie sich bitte per E-Mail an Service@joy-it.net oder per Telefon an uns.

Informationen zur Verpackung:
Verpacken Sie Ihr Altgerät bitte transportsicher, sollten Sie kein geeignetes Verpackungsmaterial haben oder kein eigenes nutzen möchten kontaktieren Sie uns, wir lassen Ihnen dann eine geeignete Verpackung zukommen.

12. SUPPORT

Wir sind auch nach dem Kauf für Sie da. Sollten noch Fragen offen bleiben oder Probleme auftauchen stehen wir Ihnen auch per E-Mail, Telefon und Ticket-Supportsystem zur Seite.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Telefon: +49 (0)2845 98469 – 66 (10 - 17 Uhr)

Für weitere Informationen besuchen Sie unsere Website:
www.joy-it.net

Veröffentlicht: