Tutorial: Raspberry Pi Projekt LED RGB Strip ansteuern und dimmen

Tutorial: RGB LED Strip, abhängig von der Helligkeit (Lichtsensor) und durch Bewegung (PIR Bewegungsmelder) ausgelöst, ansteuern und dimmen

Hier möchte ich eines meiner ersten kleinen Projekte vorstellen, die ich mit dem Raspberry Pi realisiert habe. Die Zielstellung war es, für die Nacht eine unaufdringliche indirekte Küchenbeleuchtung (Nachtlicht) zu schaffen, die automatisch, erst bei Dunkelheit und durch Bewegung aktiviert wird.

Als Leuchtmittel wählte ich einen RGB LED Strip (max. 5m). Zur Steuerung kommen hier ein passiver Infrarotbewegungsmelder (PIR) in Kombination mit einem digitalen Lichtsensor zum Einsatz. Mit dem Lichtsensor TSL2561 können Sie auf das Lux genau die Helligkeit festlegen, ab der die Beleuchtung eingeschaltet werden soll.

Da die LED über Pulsweitenmodulation (PWM) angesteuert werden, genügen einfache NPN-Siliziumtransistoren zur Ansteuerung. Zur Sicherheit nutze ich jedoch das 12V-RGB-MOSFET-Modul. Die MOSFETs können bei 12V je Kanal bis zu 25A schalten. Das sollte auch für längere Lichtschläuche ausreichend sein.

Da der LED-Strip mit 12V betrieben wird, nehmen wir die Versorgungsspannung des Lichtschlauches gleich her, um, über einen Spannungswandler auf 5V reduziert, auch den Raspberry Pi zu versorgen.

Grundsätzlich ist ein Pi natürlich mit dieser Funktion allein etwas unterbeschäftigt und sicherlich sind die Kosten für ein solches Projekt nicht ganz unerheblich. Sie bekommen jedoch, wenn wir mal die Funktion als Küchennachtbeleuchtung beiseite lassen, eine RGB-Lichtsteuerung, die problemlos über WLAN und Bluetooth angesprochen werden und damit hervorragend in jede Hausautomatisation integriert werden kann.

Und natürlich ist die Lichtsteuerung nicht die einzige Funktion, die der Raspi für Euch übernehmen kann. Aber sie dann eine von vielen sein.

Bei Bedarf lässt sich natürlich auch ein integriertes Webinterface für die Lichtschlauch-Ansteuerung erstellen.

Schritt 1 - Hilfsmittel und Vorbereitungen

Hilfsmittel

Als Hilfsmittel benötigen wir folgende Utensilien:

einen kleinen Kreuzschlitzschraubendreher mit einem Durchmesser von max. 2,5mm für das Gehäuse
12 Steckbrücken mit weiblichen Steckverbindern für die Verkabelung des Raspberry Pi.
1 USB-Netzteil mit Micro-USB-Stecker und mindestens 1A Ausgangsstrom
1 USB-Maus und Tastatur
1 Bildschirm mit HDMI-Schnittstelle
1 HDMI Anschlusskabel
1 Lötkolben und Lötzinn (Hohldraht mit Flussmittel)
1 Digital-Multimeter (nicht zwingend)
1 SD-Kartenleser für Micro-SD-Karten
1 Micro-SD-Karte (mindestens 8GB)

Betriebssystem einrichten

Zunächst erstellen wir eine Micro-SD-Karte mit einem aktuellen Raspbian-Image (Siehe Tutorial Grundeinrichtung).

Raspberry Pi starten
Da wir noch nicht über unsere 12V Stromversorgung verfügen, verwenden wir ein normales USB-Netzteil mit Micro-USB-Stecker für die ersten Schritte der Einrichtung unseres Raspberry Pi.

Wir stecken die erstellte Micro-SD-Karte in den Raspberry Pi, verbinden das USB-Netzteil mit dem Pi, schliessen Maus, Tastatur und Bildschirm an.

Kontrolliert noch einmal, ob alle Verbindungen korrekt sind.

Jetzt verbinden wir das USB-Netzteil mit dem Stromnetz und fahren den Raspberry Pi hoch. Anschliessend können wir mit der Grundeinrichtung fortfahren.

Netzwerkkarten des Pi einrichten
Natürlich wollen wir aus dem Netzwerk auf unseren Pi zugreifen. Dazu konfigurieren wir die Netzwerkkarten des Pi, wie in dem folgenden Tutorial angegeben: Grundeinrichtung der Nertzwerkverbindungen.

SSH und XRDP einrichten
Nachdem unser Pi jetzt im Netzwerk funktioniert, wollen wir ihn natürlich auch über das Netzwerk einrichten und ggf. auch fernsteuern können. Dabei hilft uns das Tutorial: SSH und XRDP einrichten.

Samba Netzwerkfreigabe einrichten
Um unser Steuerscript einfach aus dem Netzwerk konfigurieren und bearbeiten zu können, richten wir uns jetzt noch eine Samba Dateifreigabe für den Ordner /home/pi ein, in dem wir unsere Scriptdateien ablegen werden. Das Tutorial dafür findet Ihr hier: Samba (SMB) Freigabe einrichten.

Schritt 2 - Anschluss der Stromversorgung / Inbetriebnahme des Pi

Nachdem nun alle notwendigen und hilfreichen Grundeinstellungen abgeschlossen sind, können wir uns endlich der projektspezifischen Schritte annehmen.

Falls er noch in Betrieb ist, fahren wir jetzt unseren Raspberry Pi herunter und trennen ihn von dem 5V USB-Netzteil. Es wird für die folgenden Schritte nicht mehr benötigt.

Da der 5V-Anschluss für unseren Pi auf dem MOSFET-Modul angeordnet ist, nehmen wir zunächst dieses Modul und verbinden die Schraubklemmen 01 mit dem Masseanschluss und 02 mit dem 12V-Anschluss des 12V-Netzteiles für unseren LED-Stripe.

Wenn wir jetzt den Stecker des 12V-Netzteils mit einer Steckdose verbinden, sollten Sie mit dem Multimeter zwischen den Anschlussstiften 01 unds 02 der Stiftleiste eine Spannung von 5V messen können, wobei der Pluspol der Spannung auf Pin 01 und der Masseanschluss auf Pin 02 liegt.

Nachdem also unsere 12V-Stromversorgung erfolgreich die 5V für unseren Raspberry Pi zur Verfügung stellt, trennen wir das Netzteil noch einmal von der Steckdose. Jetzt verbinden wir Pin01 der Stiftleiste des MOSFET-Modules mit Pin2 des GPIO-Ports (+5V) am Raspberry Pi. Nun sind noch Pin02 der Stiftleiste und Pin6 des GPIO (Ground/GND/Masse/0V) zu verbinden. Dafür verwenden wir zwei von unseren Steckbrücken.

Der Pi ist jetzt über unser MOSFET-Modul mit der 5V-Stromversorgung verbunden. Wenn wir jetzt wieder das Netzteil an die Steckdose anschliessen, sollte unser Himbeerchen (Raspberry) hochfahren.

Damit ist dieser Schritt abgeschlossen und wir können im nächsten Teil dieses Tutorials mit der Verkabelung der Ausgänge unseres RGB-Treibermoduls fortfahren.

Schritt 3 - Anschluss des 12V 3-Kanal MOSFET Modules

Für diesen Schritt, wie für jede andere Arbeit, die wir mechanisch an unserem Projekt durchführen, fahren wir den Raspberry Pi wieder herunter und trennen ihn vom Stromnetz.

Zunächst Schliessen wir unseren LED-Streifen an die Schraubklemmen des MOSFET-Moduls an. Dafür verwenden wir die Schraubklemmen 09-12 wie folgt:

Klemme 09 -> Kanal Rot
Klemme 10 -> Kanal Grün
Klemme 11 -> Kanal Blau
Klemme 12 -> +12V

Die +5V und GND des Modules sind schon mit dem Pi verbunden. Jetzt nehmen wir drei weitere Steckbrücken und stecken sie auf die Stifte 08 (Kanal Rot), 09 (Kanal Grün) und 10 (Kanal Blau) am MOSFET-Modul.

Wenn wir jetzt das Netzteil wieder mit Strom versorgen, können wir die Schaltfunktion unseres Mosfet-Moduls testen. Die Ansteuerung des Modules durch den Raspberry Pi findet über ein 3,3V-Signal statt. Zum Testen können wir den Pin1 der GPIO, an dem 3,3V anliegen, verwenden.

Wir stecken dazu die Steckbrücken der einzelnen Kanäle (R, G und B) nacheinander auf den Pin1 der GPIO. Je nach Kanal müsste die entsprechende Farbe am Lichtstreifen aufleuchten. Nach dem Test trennen wir den Pi wieder von der Stromversorgung.

Die Enden unserer Steckbrücken, die wir soeben zum Testen verwendet haben, verbinden wir nun in der gleichen Folge mit den Pins 11 (GPIO 17 -> Rot), 13 (GPIO 27 -> Grün) und 15 (GPIO 12 -> Blau) der GPIO des Raspberry Pi.

Damit ist die Verkabelung des MOSFET-Modules abgeschlossen. Wir stecken jetzt das Netzteil wieder in die Steckdose, warten bis unser Pi gestartet ist und öffen ein Terminalfenster.

Zuerst definieren wir jetzt die GPIO-Anschlüsse 17, 22 und 27 als Ausgänge. Das geht so:

```
  $ gpio -g mode 17 out (enter)
$ gpio -g mode 22 out (enter)
$ gpio -g mode 27 out (enter)
```

Um jetzt einen Ausgang auf High (1) und damit auf einen Pegel von 3,3V zu setzen und einzuschalten, geben wir folgende Zeile ein:

```
  $ gpio -g write 17 1 (enter)
```

Für die anderen Kanäle tauschen Sie einfach die 17 in der Zeile gegen die 22 bzw. 27 aus.

Zum Resetten benutzen wir folgenden Code:

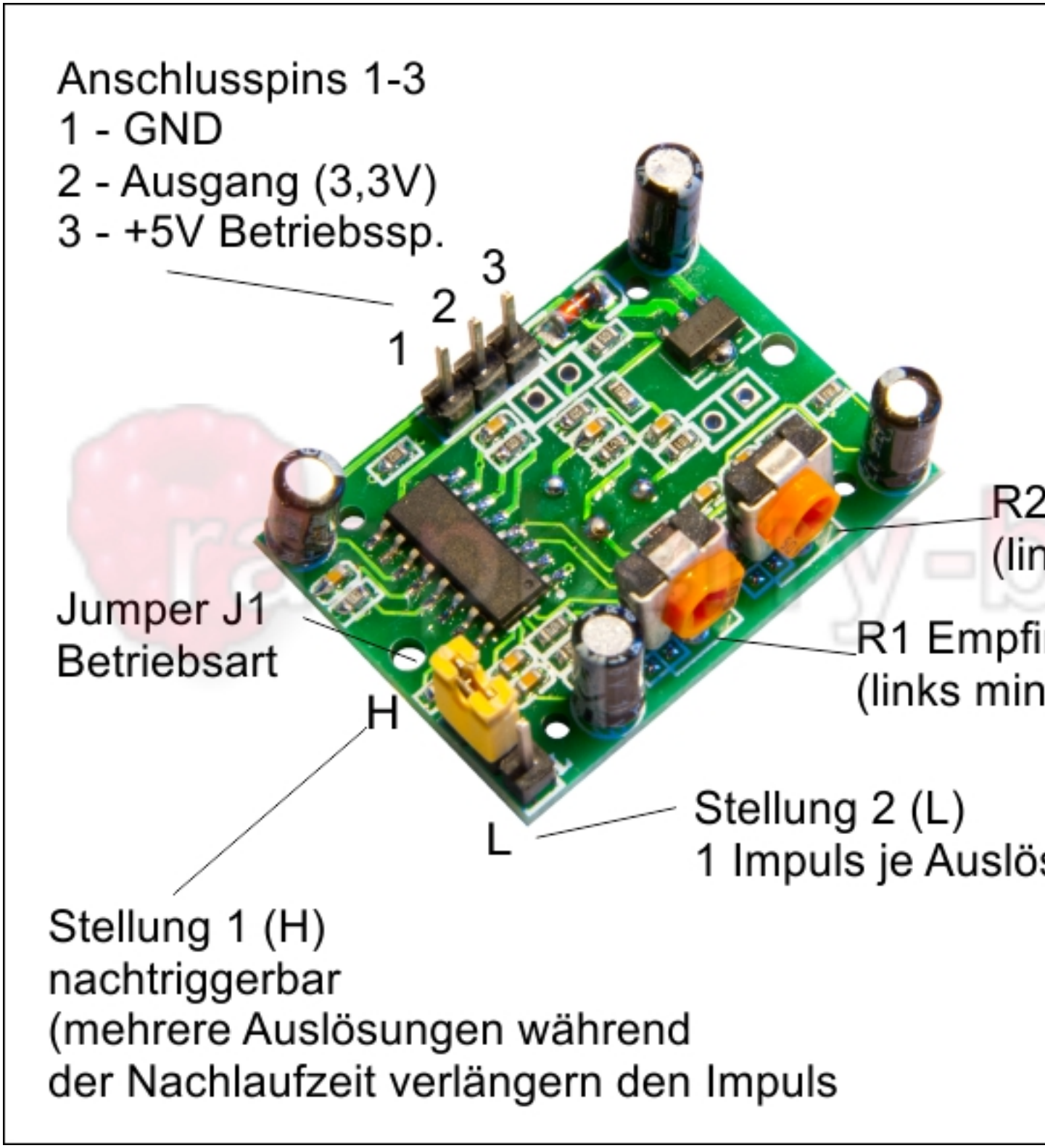
```
$ gpio -g write 17 0 (enter)
```

Auch hier tauschen wir für die anderen Kanäle die 17 wieder gegen die 22 bzw. 27 aus.
Damit können wir unsere Ausgänge schon einmal ein- bzw. ausschalten und dieser Schritt ist abgeschlossen.

Schritt 4 - PIR Bewegungsmelder am Raspberry Pi anschliessen und testen

Bevor wir die Sensoren mit dem Gehäuse verheiraten, prüfen wir zunächst einmal ihre Funktion. Dazu beginnen wir mit dem PIR-Bewegungsmelder.

Unser Bewegungsmelder verfügt über drei Anschlüsse (Siehe Abbildung)



Obwohl die Spannungsversorgung des Bewegungsmelders mit 5V stattfindet, bietet uns der Ausgang einen

H-Pegel von 3,3V, so dass wir diesen auch direkt mit dem GPIO-Eingang unseres Raspberry Pi verbinden können.

Bevor wir den Bewegungsmelder anschliessen, fahren wir unseren Pi wieder herunter und trennen das Netzteil von der Steckdose.

Wir nutzen für den Anschluss wieder drei unserer Steckbrücken. Die Anschlussbelegung gestalten wir wie folgt:

Pin1 (GND/Ground/Masse) -> GPIO Pin9
Pin3 (+5V) -> GPIO Pin4
Pin2 (Ausgang) -> GPIO.5 Pin29

Nachdem wir jetzt den PIR Bewegungsmelder verkabelt haben, nehmen wir noch ein paar Einstellungen vor. Die Reichweite und Haltezeit des Melders können wir, wie in der Abbildung gezeigt, mit den zwei Potenziometern R1 und R2 einstellen. Zum Testen bringen wir die beiden Regler in Mittelstellung.

Den Jumper J1 setzen wir auf Stellung 1 (H - nachtriggerbar). Nun können wir den Bewegungsmelder auch gleich testen. Dafür schliessen wir unser Netzteil wieder an und warten, bis der Pi hochgefahren ist.

Zunächst müssen wir den Pin29 (GPIO.5) des Raspberry Pi, an dem wir den Bewegungsmelder anschliessen, auch als Eingang definieren.

Dazu öffnen wir wieder ein Terminalfenster und tippen folgende Zeile ein:

$ gpio -g mode 5 in (enter)

Damit ist der Pin zum Eingang geworden. Jetzt können wir nachschauen, ob der Melder auch schaltet. Hierfür geben wir in das Terminalfenster folgenden Code ein

$ gpio readall (enter)

Jetzt listet uns der Pi im Terminalfenster die komplette Anschlussbelegung und die aktuellen Zustände an allen Pins an. Wir schauen uns hier speziell den GPIO.5 an und lesen den Status immer wieder aus. Wenn wir für Bewegung vor dem Bewegungsmelder sorgen, sollte sich der Status ab und zu ändern.

Unser Bewegungsmelder funktioniert!!!

Schritt 5 - Lichtsensor am Raspberry Pi anschliessen

Damit unser Nachtlicht nicht am Tag bzw. bei Tageslicht ständig angeht, richten wir uns jetzt noch einen digitalen Lichtsensor ein, mit dem wir auf das Lux genau die Helligkeit messen können.

An dieser Stelle wird sicherlich der eine oder andere Minimalist aufstehen und fragen, ob hier wirklich ein digitaler Lichtsensor notwendig ist. Normalerweise würde es sicherlich auch ein Fotowiderstand mit einer einfachen Schaltstufe machen aber hier sprechen gleich mehrere Gründe für den digitalen Lichtsensor.

Da wäre die kompakte Bauform. Der Lichtsensor ist quasi winzig klein, kleiner als ein Fotowiderstand mit selbst gebastelter Schaltstufe und Stellglied (Potenziometer) wäre.

Genauigkeit und Homogenität. Während ein Fotowiderstand eher ein Schätzinstrument ist, kann der digitale Lichtsensor auf das Lux genau messen und ist damit auch für andere Anwendungen ideal geeignet.

Der Preis ist nicht nur überschaubar, er ist mindestens so preiswert, wenn nicht sogar günstiger, als es eine Schaltstufe mit Fotowiderstand wäre.

Es ist keine Justierung notwendig. Wir lesen einfach die Helligkeitswerte aus und arbeiten mit absoluten numerischen Werten.

Damit liegen die Vorteile auf der Hand.

Bevor wir jedoch den Lichtsensor anschliessen und auslesen können, müssen wir zuerst die beiliegende Stiftleiste für den Anschluss mit der eigentlichen Platine des Lichtsensors verbinden. Dazu löten wir die Stiftleiste auf die Rückseite der Sensorplatine (das ist die Seite ohne Bauelemente) auf.

Jetzt benutzen wir die übrigen vier Steckbrücken und verbinden sie mit der Stiftleiste des Lichtsensors. Der Lichtsensor benötigt eine Betriebsspannung von 2,7V - 3,6V. Da uns der Raspberry Pi über die GPIO-Pins 1 und 17 eine Spannung von 3,5V zur Verfügung stellt, können wir diese dafür perfekt benutzen.

Die Programmierung und das Auslesen des Sensors erfolgen dann digital über die in den Sensor eingebaute I2C Schnittstelle, die wir dafür mit dem I2C-Bus des Raspberry Pi verbinden. Für die Kommunikation mit dem Sensor benutzen wir ein kleines Python-Programm. Doch dazu kommen wir später.

Zunächst fahren wir unseren Pi wieder herunter und trennen das Netzteil von der Steckdose.

Nun verbinden wir die Anschlüsse des Lichtsensors mit der GPIO-Leiste des Raspberry Pi wie folgt:

VCC -> GPIO Pin1
GND -> GPIO Pin9
SCL -> GPIO.3 Pin5
SDA -> GPIO.2 Pin3
INT -> bleibt frei

Um zu prüfen, ob unser TSL2561 korrekt mit dem Pi verbunden ist und am I2C-Bus auch erkannt wird, geben wir das folgende Kommando ein:

pi@rechnername ~ $ i2cdetect -y 1 (enter)

Jetzt listet uns unser Pi alle am I2C-Bus erkannten Geräteadressen auf. Die Adresse des TSL2561 ist standardmäßig die 39. Wenn die hier nicht angezeigt wird, müsst Ihr die Verkabelung des Lichtsensors noch einmal überprüfen.

Nachdem wir den Lichtsensor mit dem Raspberry Pi verbunden haben, sind jetzt alle notwendigen Komponenten zu einem Gerät vereint. Im nächsten Schritt beschäftigen wir uns mit dem Python-Script, mit dem wir Lichtsensor und Bewegungsmelder abfragen und den LED-Streifen ansteuern werden.

Schritt 6 - Script für die Lichtstreifensteuerung installieren und konfigurieren

Um Lichtsensor und Bewegungsmelder auszulesen und den LED-Streifen zu steuern benötigen wir noch ein

kleines Script. Dieses Script verwendet als Beispiel für die Steuerung der LED und um es möglichst einfach zu halten, jeweils nur eine der Grundfarben. Eine kleine Änderung ermöglicht jedoch natürlich auch das Mischen zu einer anderen möglichen RGB-Farbe.

Es fragt ständig den Bewegungsmelder und den Helligkeitssensor ab. Sobald der Bewegungsmelder auslöst und die gemessene Helligkeit einen gewissen Wert unterschreitet, blendet das Script die Helligkeit der gewählten Farbe bis zum gewählten Wert (1-255) auf, pausiert, solange der Melder ausgelöst ist plus eine gewählte Zeit in Sekunden und blendet danach wieder das Licht aus.

Der Quelltext unseres kleinen Demoscriptes lautet wie folgt:

```
 from os import system
system("sudo killall pigpiod")
system("sudo pigpiod")
import RPi.GPIO as GPIO
import time
import pigpio
import smbus

#I2C Bus initialsieren bus = smbus.SMBus(1)
#Lichtsensor (Adresse 0x39) initialisieren
#Power On Mode
bus.write_byte_data(0x39, 0x00 | 0x80, 0x03)
bus.write_byte_data(0x39, 0x01 | 0x80, 0x02)

 #kurze Pause
#time.sleep(0.5)

#Lichtsensor auslesen
data = bus.read_i2c_block_data(0x39, 0x0c | 0x80, 2)
data1 = bus.read_i2c_block_data(0x39, 0x0e | 0x80, 2)

#Lichtdaten aufbereiten
ch0 = data[1] * 256 + data[0] ch1 = data1[1] * 256 + data1[0]

#Lichtdaten ausgeben
print "Volles Spektrum (Infrarot+sichtbar) : %d lux" %ch0
print "Infrarot-Wert: %d lux" %ch1
print "sichtbares Licht-Wert : %d lux" %(ch0 - ch1)

#GPIO ports setzen und zuweisen
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
red = 17
green = 27
blue = 22
PIR = 5

GPIO.setup(red, GPIO.OUT)
GPIO.setup(green, GPIO.OUT)
GPIO.setup(blue, GPIO.OUT)
GPIO.setup(PIR, GPIO.IN)
pi = pigpio.pi()

#Variablen definieren
```

```
pi.set_PWM_dutycycle(red, 0)
pi.set_PWM_dutycycle(green, 0)
pi.set_PWM_dutycycle(blue, 0)
fade_in_delay = .1
fade_out_delay = .1
delay_time = 5
mycolor = blue
brightness = 200
min_light = 9
moment = 0

RUNNING = True

#Script starten
try:
    while RUNNING:
        data = bus.read_i2c_block_data(0x39, 0x0c | 0x80, 2)
        data1 = bus.read_i2c_block_data(0x39, 0x0e | 0x80, 2)
        ch0 = data[1] * 256 + data[0]
        ch1 = data1[1] * 256 + data1[0]
        light = ch0 - ch1
        print "sichtbares Licht-Wert : %d lux" %(light)
        if pi.read(PIR):
            print "PIR = ON"
        else:
            print "PIR = OFF"
        if moment &gt; 1:
            start = moment
        else:
            start = 0
        if (pi.read(PIR) &amp; (light &lt; min_light)):
            for x in range(start,brightness):
                pi.set_PWM_dutycycle(mycolor, x)
                time.sleep(fade_in_delay)
                moment = x
                while (pi.read(PIR) &amp; (light &lt; min_light)):
                    time.sleep(delay_time)
                if not (pi.read(PIR) &amp; (moment &gt; 0)):
                    for x in range(brightness, 0, -1):
                        pi.set_PWM_dutycycle(mycolor, x)
                        time.sleep(fade_out_delay)
                moment = 0
 except KeyboardInterrupt:
        RUNNING = False
```

Diesen Quelltext finden Sie komplett in der Datei mit dem Namen "PIR_LED_Stripe.py", die Sie über den folgenden Link herunterladen können - Link: PIR_LED_Stripe.py.

Diese Datei kopieren wir dann in das Verzeichnis /home/pi auf unserem Raspberry Pi. Dazu benutzen wir am besten unsere frisch eingerichtete SMB-Freigabe, indem wir in die Adressleiste des Windows-Explorers die folgende Zeile eingeben. Natürlich ersetzen Sie die xxx-Einträge durch die tatsächliche IP-Adresse Ihres Raspberry Pi: \\xxx.xxx.xxx.xxx .

Nachdem wir die Datei kopiert haben, öffnen wir auf unserem Raspberry-Pi Desktop das Himbeerchen (Startmenü), klicken dann auf Entwicklung und wählen dort den Punkt Python 2 (IDLE) aus.

In der IDLE-Oberfläche gehen wir dann im Menü auf > File > Open und wählen dann in dem Dialog das Verzeichnis /home/pi und dort die Datei: PIR_LED_Stripe.py aus. Anschliessend starten wir dann das Script durch Drücken von F5 oder > Run > Run Module F5.

Jetzt sollte Das Script laufen und uns die Helligkeit in Lux und den Zustanddes Bewegungsmelders anzeigen.

Schritt 7 - Python Script beim Booten automatisch starten

Damit das Script beim Booten des Raspberry Pi automatisch gestartet wird, führen wir es über einen cronjob aus. Um den croneditor zu starten, führen wir das folgende Kommano am Terminalfenster aus:

$ sudo crontab -e (enter)

Hier fügen wir dann folgende Zeile am Ende der Datei ein:

@reboot python /home/pi/PIR_LED_Stripe.py &

Jetzt noch speichern und schliessen. Anschliessend werden wir den Pi mit folgender Kommandozeile neu starten:

$ sudo reboot (enter)

Und fertig!!!

Schritt 8 - Watchdog aktivieren

Da es im Dauerbetrieb auch mal zum Aufhängen oder Absturz des Pi kommen kann, aktivieren wir zum Schluss noch den Hardware-Watchdog des Raspberry Pi.

Der überprüft in kurzen Intervallen, ob der Pi noch läuft und startet ihn bei Störungen automatisch neu. Die Anleitung zur Aktivierung des Watchdogs findet Ihr im folgenden Tutorial: Raspberry Pi - Hardware Watchdog installieren