

Tutorial: Raspberry Pi - Passiv Infrarot Bewegungsmelder PIR

Tutorial: Raspberry Pi - Passiv Infrarot Bewegungsmelder (PIR)

PIR Bewegungsmelder am Raspberry Pi, Verschaltung, Einstellungen und Abfrage mit Python.

In vielen Situationen, in denen gesteuert oder geregelt werden soll, ist es notwendig, Bewegungen zu detektieren, um sie als Auslöser (Trigger) für bestimmte Aktionen einer Steuerung zu verwenden.

Zu diesem Zweck gibt es auf dem Markt viele verschiedene Typen an sogenannten Bewegungsmeldern. Eine Variante ist die Erkennung der Bewegung von "warmen" Objekten vor einem "kalten" Hintergrund. Dafür verwendet man Passiv Infrarot Bewegungsmelder (PIR).

In diesem Tutorial möchte ich Euch zeigen, wie man einen solchen PIR-Melder richtig einstellt und mit dem Raspberry Pi korrekt verbindet. Anschliessend schauen wir uns dann an, wie wir die Aktionen des Melders mit dem Pi auswerten können.

Vorbereitungen, Hilfsmittel

Werkzeuge benötigen wir für dieses Tutorial nicht. Als Hilfsmittel dienen uns neben dem PIR Bewegungsmelder und dem fertig grundeingerichtetem (Siehe Tutorials Grundlagen -&gt; Betriebssystem einrichten und Fernsteuerung mit SSH und XRDP) Raspberry Pi nur noch ein USB-Netzteil und drei Steckbrücken, um den Bewegungsmelder und den Pi miteinander zu verbinden.

Weiterhin ist mal wieder eine funktionierende Internetverbindung (WLAN/LAN) für den Raspberry Pi notwendig.

Damit wir, nachdem Pi und Bewegungsmelder verbunden sind, den Status des Bewegungsmelders auch auslesen können, benötigen wir die Scriptsprache Python. Python ist mit der Grundinstallation auch schon auf unserem Pi eingerichtet. Um mit Python die Ein- und Ausgänge des Raspberry Pi einfach benutzen zu können, installieren wir aber noch eine Erweiterung - das RPi.GPIO Modul.

Um dieses Modul zu installieren, verbinden wir uns per Maus/Tastatur/Bildschirm direkt oder per SSH oder XRDP über unseren PC mit unserem Raspberry Pi und starten ein Terminal.

Zuerst aktualisieren wir mal unseren Paketkatalog:

```
pi@rechnername: ~ $ sudo apt-get update (enter)
```

Dann installieren wir unsere GPIO-Erweiterung:

```
pi@rechnername: ~ $ sudo apt-get install python-rpi.gpio python3-rpi.gpio (enter)
```

Fragen, ob wir uns sicher sind, dieses Paket installieren zu wollen, beantworten wir mal wieder mit J wie ja.

Damit sind dann auch die Vorbereitungen schon abgeschlossen.

#### PIR der Passive Infrarot Bewegungsmelder PIR

Um mit dem Bewegungsmelder vernünftig zu arbeiten, müssen wir zuerst einmal seine Arbeitsweise richtig verstehen. Ein PIR-Bewegungsmelder stellt, einfach gesagt, die Bewegung von Objekten fest, deren Temperatur sich signifikant von der des Hintergrundes unterscheidet. Das bezieht sich hauptsächlich auf vertikale und horizontale Bewegungen im Verhältnis zum Bewegungsmelder.

Bewegungen hin zum oder weg vom Bewegungsmelder werden hingegen schlecht oder gar nicht registriert. Die Reichweite des Melders in der Tiefe beträgt etwa sieben Meter. Der Öffnungswinkel liegt bei etwa 120°.

Je länger der Melder im Ruhezustand (ohne Bewegungsmeldung) verbleibt, desto empfindlicher wird er bis hin zur maximalen Empfindlichkeit. Wird ein Melder ständig in kurzen Abständen immer wieder ausgelöst, so verhält er sich wie ein geblendetes Auge und nimmt Bewegungen schlechter wahr, löst also schlechter aus.

Der Melder, den wir verwenden, verfügt über drei Anschlusspins (Pin1-3), einen Jumper (Steckbrücke - J1) und zwei Potenziometer (R1 und R2) zur Einstellung von Betriebsart, Empfindlichkeit und Auslösedauer des Melders.

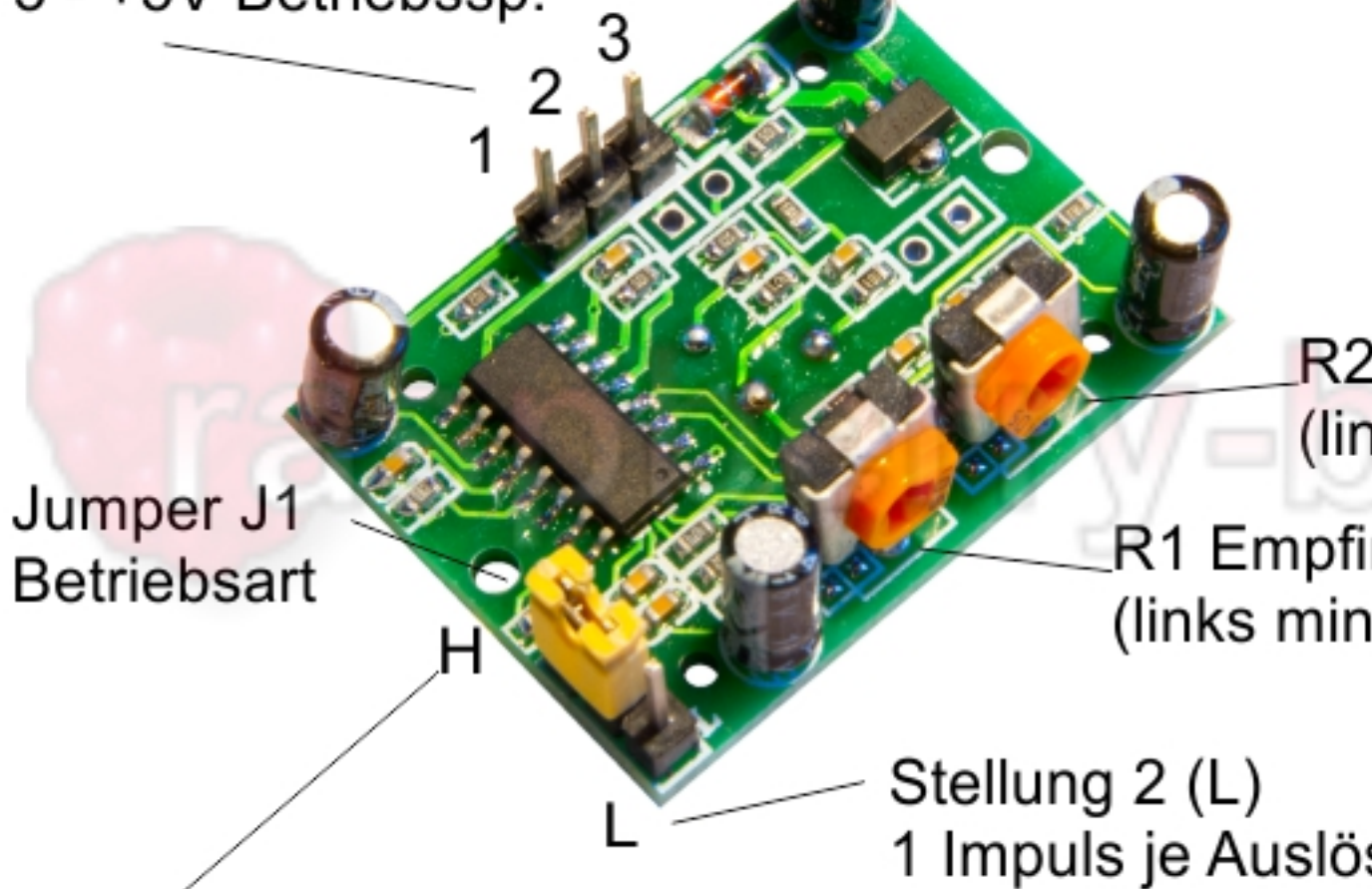
#### Bewegungsmelder Anschlüsse und Bedienelemente

## Anschlusspins 1-3

1 - GND

2 - Ausgang (3,3V)

3 - +5V Betriebssp.



Jumper J1  
Betriebsart

R2  
(lin

R1 Empf  
(links min

Stellung 2 (L)  
1 Impuls je Auslös

Stellung 1 (H)  
nachtriggerbar  
(mehrere Auslösungen während  
der Nachlaufzeit verlängern den Impuls

#### Anschlusspins des Bewegungsmelders Pin 1-3

Beginnen wir mit den drei Anschlusspins. Für die 5V Stromversorgung unseres Melders nutzen wir Pin 1 (Ground oder Masse) und Pin 3 (+5V). Pin 2 ist der Ausgang unseres Bewegungsmelders. Im Ruhezustand (keine Bewegung) liegt der Ausgang auf 0V. Trotz der 5V Betriebsspannung unseres PIR-Melders liefert der Ausgang bei Bewegungsmeldung einen H-Pegel von 3,3V. Das kommt uns bei der Verwendung mit dem Raspberry Pi sehr entgegen, da unsere GPIO auch mit 3,3V Pegeln arbeitet.

Wir können den Pin 2 des Melders also direkt an einen als Eingang programmierten Pin unseres Pi anschließen. Auch die 5V Betriebsspannung und den Masseanschluss (Ground) können wir uns direkt per Steckbrücke vom Pi holen. Der PIR-Bewegungsmelder hat eine Stromaufnahme im  $\mu\text{A}$  (Mikroampere)-Bereich. Deshalb können wir den Stromverbrauch des Melders gern vernachlässigen.

#### Die Betriebsarten des Melders (J1)

Mit der Steckbrücke (Jumper J1) lässt sich der Bewegungsmelder zwischen zwei Betriebsarten umschalten. In Stellung 1 (H) bleibt der Ausgang des Melders bei Feststellung einer Bewegung so lange auf H (3,3V), bis die Bewegungsmeldung beendet und die Nachlaufzeit vergangen ist. Wird innerhalb der Nachlaufzeit eine weitere Bewegung festgestellt, verlängert sich die Impulszeit wieder bis Ende der Bewegungsmeldung plus Nachlaufzeit. Erst nach Ablauf der Nachlaufzeit geht der Ausgang wieder auf 0V zurück.

Steht der Jumper jedoch auf Stellung 2 (L), so wird der Ausgang des Melders bei Bewegungsdetektion für die Dauer der eingestellten Nachlaufzeit auf H (3,3V) gesetzt. Danach geht der Ausgang in jedem Fall erst wieder in den L(0V)-Zustand über. Erst nach einer festen Sperrzeit von zwei Sekunden kann der Ausgang dann wieder durch eine Bewegungsmeldung auf H gesetzt werden.

#### Regler für Empfindlichkeit (R1)

Mit dem Regler R1 regelt man die Empfindlichkeit des Bewegungsmelders. Damit kann man sozusagen einstellen, wie stark der Temperaturunterschied zwischen Hintergrund und zu detektierendem Objekt sein muss. Hier müsst Ihr einfach ausprobieren, welche Einstellung für Eure spezielle Anwendung die beste ist. Linksanschlag des Reglers ist die geringste und dementsprechend Rechtsanschlag die höchste Empfindlichkeit des Melders.

#### Regler für die Nachlaufzeit/Haltezeit des Melders (R2)

Der Regler R2 ist zuständig für die Haltezeit bzw. Nachlaufzeit des Ausgangs des PIR-Bewegungsmelders nach erster bzw. letzter Auslösung (je nach Stellung des Jumpers J1). Am Linksanschlag des Reglers erzielen wir die kürzeste Nachlaufzeit (einige Sekunden). Am rechten Anschlag werden das dann schon einige Minuten. Auch hier müsst ihr wieder probieren, welche Einstellung für Euch die richtige ist.

### Raspberry Pi und Bewegungsmelder (PIR) verbinden

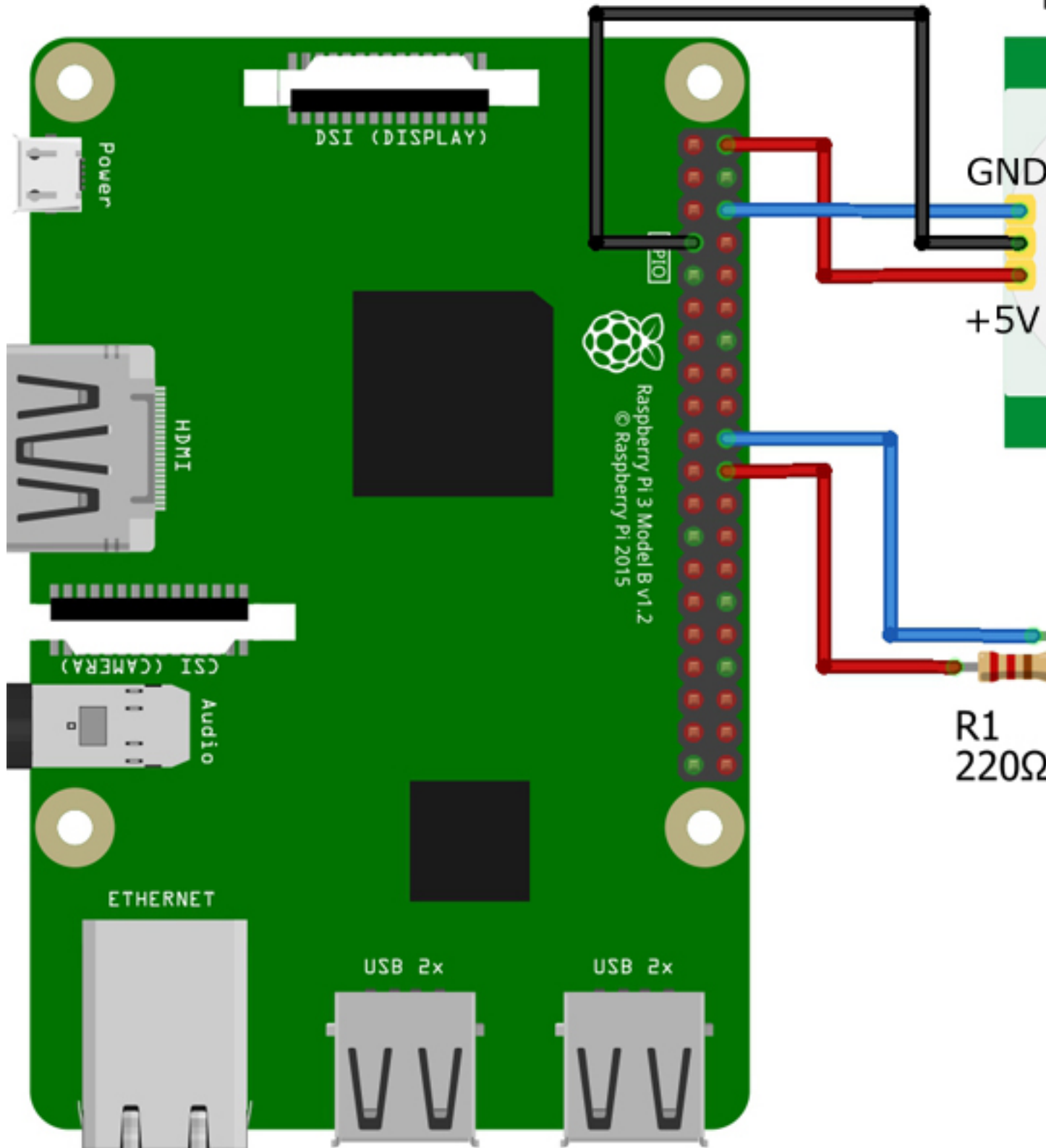
Nachdem wir verstanden haben, wie der PIR Bewegungsmelder funktioniert und auch wissen, wie wir ihn korrekt einstellen können, befassen wir uns in diesem Schritt damit, den Bewegungsmelder mit unserem Raspberry Pi zu verbinden.

Dazu nutzen wir die Anschlüsse der GPIO am Pi, die über eine 40polige Stiftleiste nach aussen geführt werden. Die GPIO-Anschlüsse stellen neben den Betriebsspannungen (3,3V und 5V) und einigen Schnittstellen bis zu 26, teilweise vorbelegte und teilweise frei belegbare, Ein- und Ausgänge zur Verfügung, von denen wir einen für die Auswertung unseres PIR Bewegungsmelders verwenden werden.

Die konkrete Belegung der GPIO finden Sie bei den Grundlagen in einem detaillierten Tutorial zur GPIO des Raspberry Pi.

In unserem Fall nutzen wir den GPIO4 (Pin 07), um den Ausgang des Bewegungsmelders auszulesen. Zur Kontrolle und Anzeige des Zustandes unseres PIR verwenden wir eine LED, die wir über einen 220 Ohm Vorwiderstand an den GPIO25 (Pin 22) anschliessen. Wir verbinden den Pi mit dem Bewegungsmelder, dem Widerstand und der LED, wie in der folgenden Abbildung dargestellt:

## Raspberry Pi RPI-3-V1.2



## Bewegungsmelder mit Python abfragen und auswerten

Das folgende Python 2 - Script, das dem Inhalt der Datei PIR\_in\_LED\_out.py entspricht, die Ihr mit einem Rechtsklick auf den Dateinamen herunterladen könnt, fragt in einer Endlosschleife den GPIO-Port des Bewegungsmelders ab und schaltet die LED bei Auslösung des Melders für eine definierte Zeit an.

```
#!/usr/bin/env python
#coding: utf8

import time
from os import system
system("sudo killall pigpiod")
time.sleep(5)
system("sudo pigpiod")
import RPi.GPIO as GPIO
import pigpio

# Zählweise der Pins festlegen
GPIO.setmode(GPIO.BCM)
#Warnmeldungen ausschalten
GPIO.setwarnings(False)

#Pins Funktionen zuweisen
LED_OUT = 25
PIR = 4

#Zeiten festlegen
ON_DELAY = 5
OFF_DELAY = 5

pi = pigpio.pi()

# Pin 7 (GPIO 4) als Eingang festlegen
GPIO.setup(PIR, GPIO.IN)
# Pin 22 (GPIO 25) als Ausgang festlegen
GPIO.setup(LED_OUT, GPIO.OUT)

#Ausgang zunächst einmal auf LOW setzen
GPIO.output(LED_OUT, GPIO.LOW)

RUNNING = True

try:
    while RUNNING:
        print "Script: PIR_in_LED_out.py gestartet"
        if (pi.read(PIR)):
            print "LED ON"
            GPIO.output(LED_OUT, GPIO.HIGH)
        while (pi.read(PIR)):
```



```
        time.sleep(ON_DELAY)
        GPIO.output(LED_OUT, GPIO.LOW)
        print "LED OFF"
        time.sleep(OFF_DELAY)
except KeyboardInterrupt:
    RUNNING = False

# Ausgänge wieder freigeben
GPIO.cleanup()
```

Um dieses Script auszuführen, kopiert es auf Euren Raspberry Pi in das Verzeichnis /home/pi, loggt Euch in den Desktop Eures Raspberry Pi ein und startet die Python Oberfläche IDLE2.

Jetzt wählt Ihr im Menü 'File' den Punkt 'Open...' aus, geht in dem sich öffnenden Dialogfeld auf den Ordner /home/pi, wählt die Datei 'PIR\_in\_LED\_out.py' aus und klickt auf die Schaltfläche 'öffnen'.

Dann sollte das Script in Eurem Python-Editor (IDLE 2) angezeigt werden. Um das Script jetzt auch noch zu starten, geht Ihr im Menü 'Run' auf den Punkt 'Run Module' und klickt ihn an. Schon geht es los.