Tutorial: Aktiver temperaturgesteuerter Lüfter am Raspberry Pi

Obwohl der Raspberry Pi in der Regel ja durch seine "Sparsamkeit" in Sachen Leistungsaufnahme und Stromverbrauch brilliert, kann es fest in ein Gehäuse eingebaut und/oder in einer recht warmen Umgebung notwendig werden, einen Lüfter zur Kühlung des Pi einzusetzen.

Besonders der Pi4 neigt zur starken Wärmeentwicklung, so dass derzeit auch verschiedene Hersteller Gehäuse mit eingebautem Lüfter auf dem Markt anbieten. Leider sind diese Lüfter, auch Aufgrund des scharfen Preiskampfes auf diesem Sektor, oft nicht von hoher Qualität und auch nicht mit Kugellagern ausgestattet. Das sorgt bei Dauerbetrieb auf Volllast schon nach wenigen Wochen zu Lüftergeräuschen und Ausfall, wass sich dann wiederum negativ auf den Raspberry Pi auswirkt.

Da sich hier die Möglichkeiten, die unser kleiner Computer hier zur Verfügung stellt, geradezu aufdrängen, mit geringstem Aufwand hier zu einer temperaturgesteuerten Lüfterregelung zu kommen, habe ich mir in diesem Tutorial vorgenommen, Euch eine Anleitung zu schreiben, wie Ihr schnell zu einer sehr brauchbaren Lösung kommen könnt. Ob Ihr dafür einen Lüfter mit Gleit- oder Kugellagern verwendet, liegt natürlich ganz bei Euch oder Eurem Budget.

Für die Temperaturmessung verwende ich den chipinternen Temperatursensort der CPU. Der ist am nächsten dran und liefert deshalb Ergebnisse aus erster Hand.

Zur Ansteuerung des 5V-Lüfters werden als zusätzliche "Hardware" nur ein 10kOhm-Widerstand und ein kleiner 0,5W NPN-Transistor benötigt.

Falls Ihr ein Steckbrett und Steckbrücken zur Verfügung habt, benötigt Ihr kein Werkzeug. Ansonsten werden Lötkolben, Lötzinn, Litze und idealerweise ein 40-poliger Pfostenverbinder für den Aufbau benötigt. Der Rest ist Software. Es wäre also auch wichtig, dass Euer Pi grundsätzlich eingerichtet ist, idealerweise über ssh und/oder xrdp fernbedienbar ist und über eine funktionierende Internetverbindung verfügt. Anleitungen für diese Vorbereitungen findet Ihr in den folgenden Tutorials:

- Raspberry Pi - Grundeinrichtung
- Raspberry Pi - Lan und WLAN Konfiguration
- Raspberry Pi - fernsteuern - ssh und Remote-Desktop

Um sicher zu gehen, dass auch Python installiert ist und Ihr wisst, wie die GPIO funktioniert, schaut Euch am besten auch noch dieses Tutorial an:

- Raspberry Pi - GPIO Pinbelegung und WiringPi

Mit diesem Rüstzeug solltet Ihr perfekt für diese Anleitung vorbereitet sein.

Passende Komponenten aus dem Shop zur Realisierung dieses Projektes findet Ihr am Ende dieses Tutorials aufgelistet.

Ich wünsche Euch viel Spass und Erfolg!

Ansteuerung des Lüfters über die GPIO des Raspberry Pi

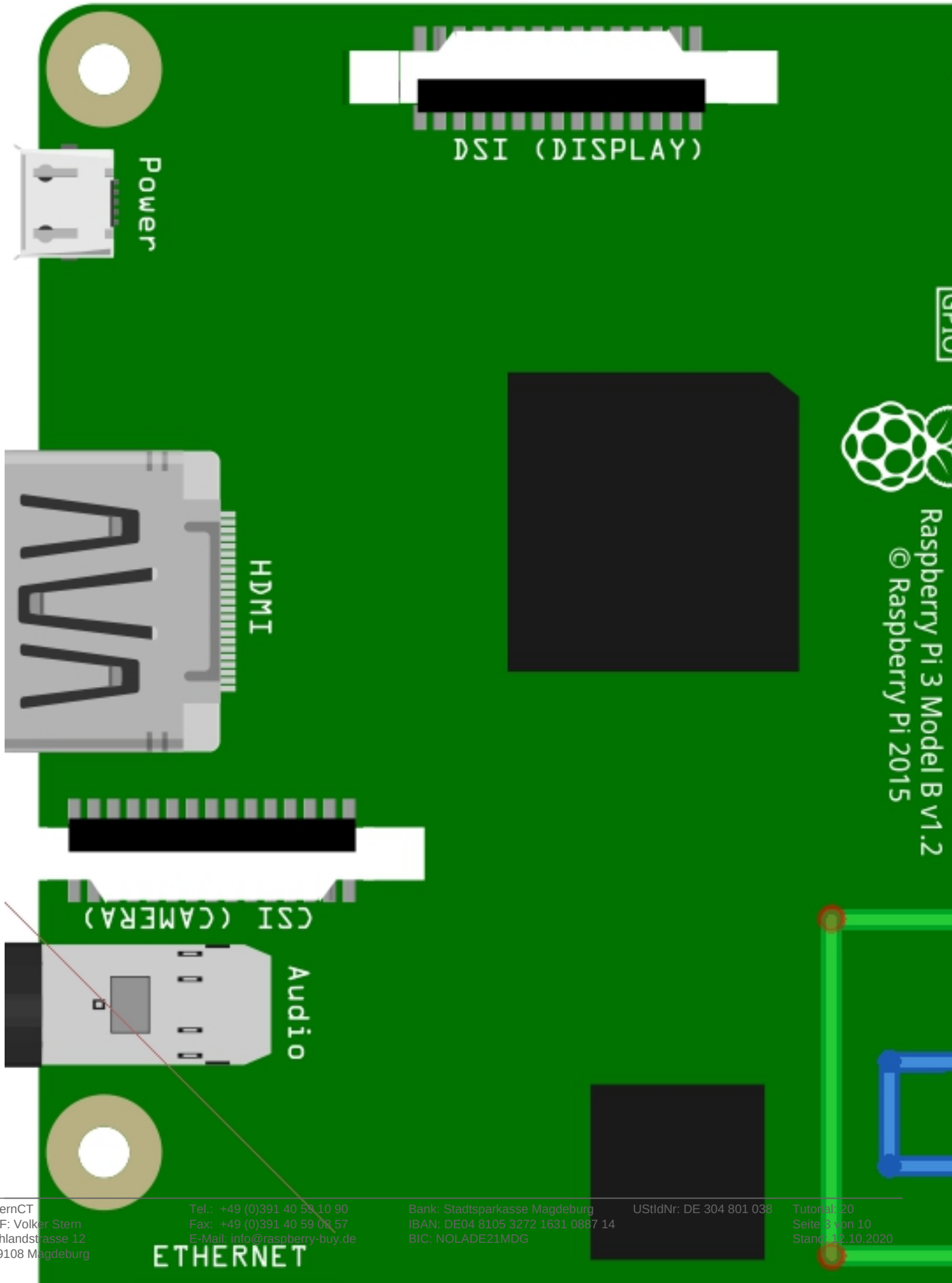
Bevor wir beginnen, den Lüfter und seine Geschwindigkeit temperaturabhängig zu regeln, beschäftigen wir uns zunächst einmal mit den elektrischen Eigenschaften der GPIO des Raspberry Pi und stellen den korrekten Anschluss des Lüfters sicher.

Unser Lüfter funktioniert mit einer Betriebsspannung von 5V und benötigt für seinen Betrieb einen Strom von ca. 0,2A. Damit hat er eine Leistungsaufnahme von (5V*0,2A) maximal 1W. Falls Ihr also Euren Pi mit zusätzlichen Stromverbrauchern (USB- oder andere Geräte, die auch Strom benötigen) betreiben möchtet, solltet Ihr sicherstellen, dass das Netzteil, mit dem Ihr Euren Pi versorgt, diesen Strom auch leisten kann.

Die GPIO-Pins des Raspberry Pi liefern im Zustand High einen Pegel von 3,3V am Ausgang und können mit maximal 0,016A (16mA) belastet werden. Da weder Spannung noch Strom am Ausgang des Pi für unseren Lüfter ausreichen, benötigen wir also etwas Hilfe, um den Lüfter ansteuern zu können. Diese Hilfe erhalten wir durch einen kleinen NPN-Transistor.

Nun könnte man meinen, dass der Lüfter auch mit 3,3V laufen wird und dann auch weniger Strom verbraucht wird und sogar noch weniger, wenn wir den Ausgang per PCM (PWM) ansteuern. Das mag ja sogar sein und kann vielleicht auch eine Zeit lang gut gehen. Zum Einen riskieren wir jedoch die Beschädigung oder sogar die Zerstörung des Raspberry Pi, zum Anderen können wir bei Bedarf gar nicht die Volle Kühlleistung unseres Lüfters abfordern. Wir steuern unseren Lüfter also auf dem sicheren Weg über einen Transistor an.

Das Schema für die Schaltung, die wir dafür verwenden, könnt Ihr dem folgenden Bild entnehmen:


DSI (DISPLAY)
Power
HDMI
CSI (CAMERA)
Audio
ETHERNET
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015

Bevor Ihr den grünen Draht mit der GPIO verbindet, könnt Ihr zum Testen der grundsätzlichen Funktion, bei eingeschalteter Stromversorgung, den Anschluss einmal mit der 3,3V Spannungsversorgung des Raspberry Pi (GPIO-Pin 1 oder 17) verbinden. Wenn alles richtig beschaltet ist, sollte in diesem Moment der Lüfter mit voller Leistung starten. Wenn das nicht funktioniert, überprüft noch einmal Eure Verdrahtung.

Wenn der Lüfter jedoch funktionierte, dann könnt Ihr den Pi vom Strom trennen und dann den grünen Draht an dem entsprechenden GPIO-Pin 33 (GPIO13 => PWM1) verbinden.

Im nächsten Schritt schalten wir den Lüfter über die GPIO ein und aus.

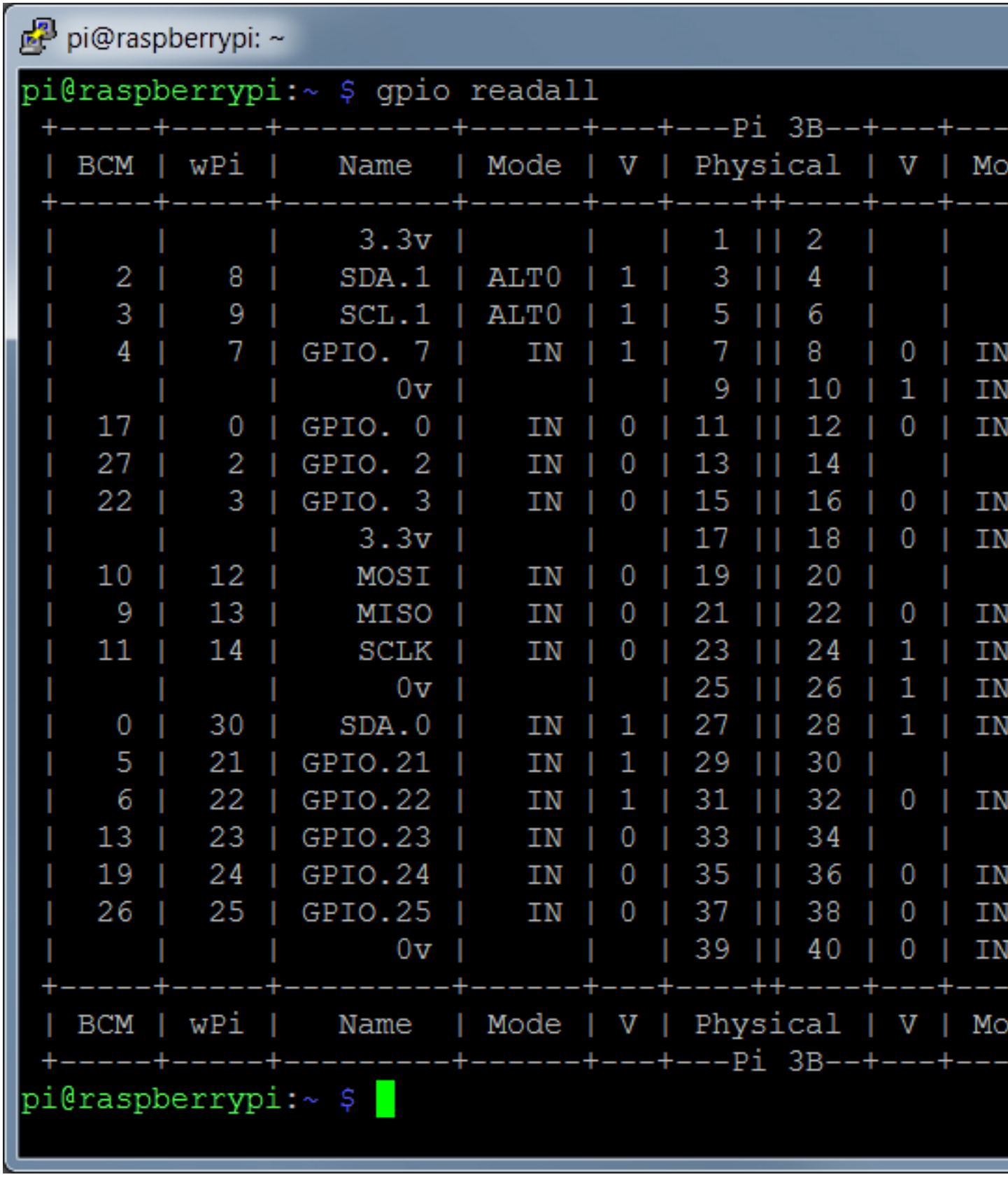
Schalten des Lüfters über den GPIO-Ausgang des Raspberry Pi

Nachdem wir sichergestellt haben, dass ein H-Pegel an unserem grünen Draht den Lüfter einschaltet, wollen wir natürlich auch sehen, dass unser Raspberry Pi diese aufgabe übernimmt. Dazu verbinden wir diesen Draht dann auch, wie auf unserem Schaltschema zu erkennen ist, mit dem GPIO Pin 33 (GPIO.13).

Anschliessend öffnen wir, entweder über SSH, Remote-Desktop oder direkt am Desktop des Pi, ein Terminal und schauen uns zunächst mal den aktuellen Zustand unserer GPIO an. Das geht mit dem folgenden Befehl:

pi@rechnername ~ : $ gpio readall (enter)

Das Ergebnis dieser Abfrage ist dann die komplette aktuelle Belegung der Raspberry Pi GPIO mit allen aktuellen Zuständen wie folgt:


pi@raspberrypi: ~
pi@raspberrypi:~ $ gpio readall
+-----+-----+---------+------+---+---Pi 3B--+---+---
| BCM | wPi | Name | Mode | V | Physical | V | Mo
+-----+-----+---------+------+---+----++----+---+---
		3.3v			1		2	
2	8	SDA.1	ALT0	1	3		4	
3	9	SCL.1	ALT0	1	5		6	
4	7	GPIO. 7	IN	1	7 || 8	0	IN
		0v			9 || 10	1	IN
17	0	GPIO. 0	IN	0	11 || 12	0	IN
27	2	GPIO. 2	IN	0	13		14	
22	3	GPIO. 3	IN	0	15 || 16	0	IN
		3.3v			17 || 18	0	IN
10	12	MOSI	IN	0	19		20	
9	13	MISO	IN	0	21 || 22	0	IN
11	14	SCLK	IN	0	23 || 24	1	IN
		0v			25 || 26	1	IN
0	30	SDA.0	IN	1	27 || 28	1	IN
5	21	GPIO.21	IN	1	29		30	
6	22	GPIO.22	IN	1	31 || 32	0	IN
13	23	GPIO.23	IN	0	33		34	
19	24	GPIO.24	IN	0	35 || 36	0	IN
26	25	GPIO.25	IN	0	37 || 38	0	IN
		0v			39 || 40	0	IN
+-----+-----+---------+------+---+----++----+---+---
BCM	wPi	Name	Mode	V	Physical	V	Mo
+-----+-----+---------+------+---+---Pi 3B--+---+---
pi@raspberrypi:~ $

Der Pin 33 (GPIO.23) ist derzeit noch als Eingang (Mode = IN) eingerichtet. Um useren Lüfter damit schalten zu können, müssen wir also aus diesem Pin zunächst einmal einen Ausgang machen. Das geht dann so:

pi@rechnername ~ : $ gpio export 13 out (enter)

Jetzt wird uns beim erneuten readall der Pin auch tatsächlich als Ausgang (Mode = OUT) angezeigt. Um nun unseren Lüfter zum Drehen zu bewegen, müssen wir einfach den Pegel unseres neuen Ausganges auf HIGH setzen. Dazu tippen wir folgenden Befehl ein:

pi@rechnername ~ : $ gpio -g write 13 1 (enter)

Und schon dreht sich unser Lüfterrad. Mit:

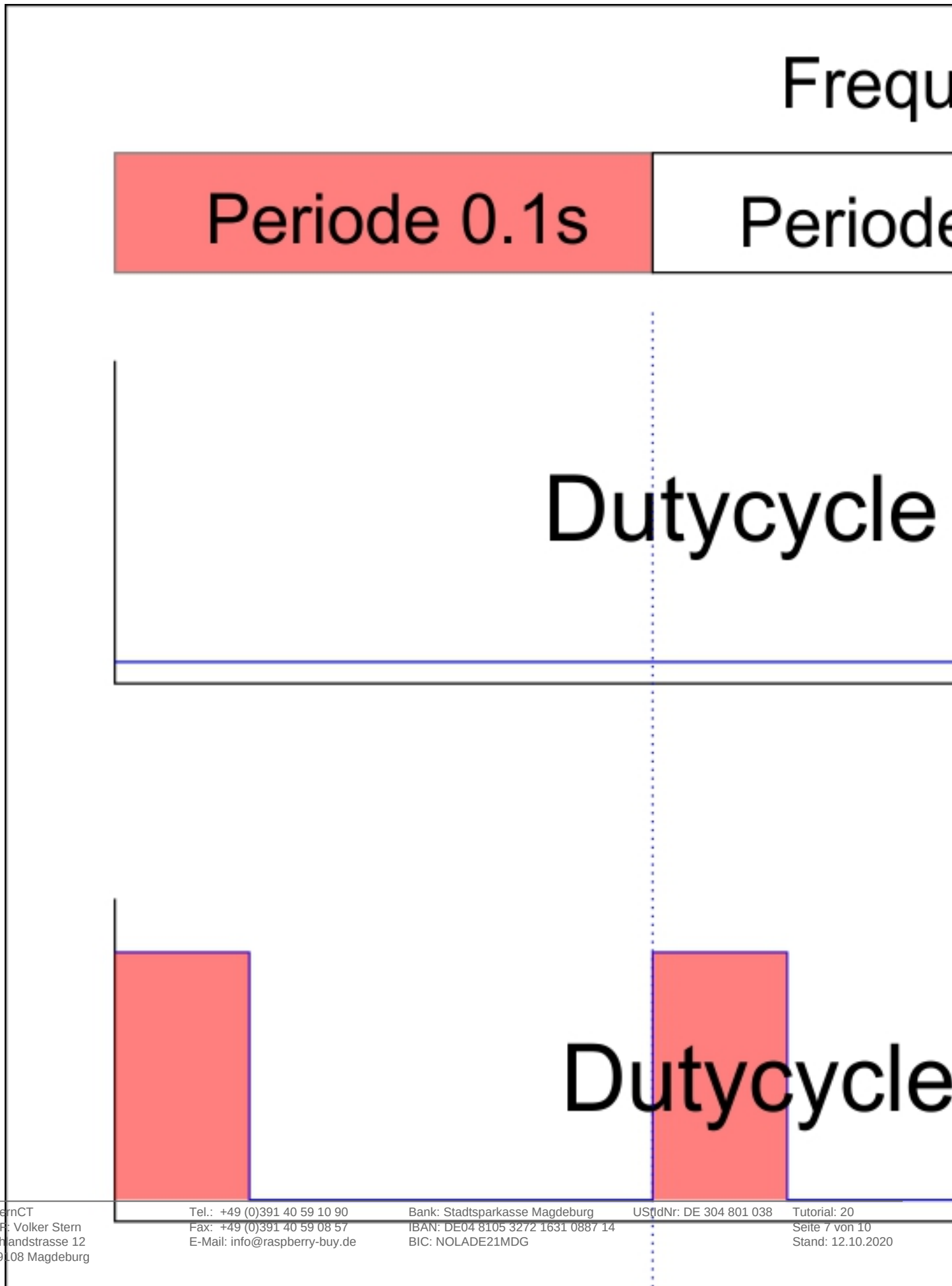
pi@rechnername ~ : $ gpio -g write 13 0 (enter)

schalten wir den Lüfter wieder aus. Nachdem also unsere Schaltung und die Ansteuerung über die GPIO funktionieren, können wir uns daran machen, den Ausgang per Python-Programm mit Pulsweitenmodulation (PWM) zu steuern und mittels der Abfrage der CPU-Temperatur unseres Pi auch zu regeln.

So funktioniert Pulsweitenmodulation (PWM)

Im letzten Schritt haben wir ausprobiert, wie wir unseren Lüfter per Terminal ein- und wieder ausschalten können. Ziel unseres kleinen Tutorials ist es jedoch, den Pi die Lüfterdrehzahl temperaturabhängig regeln zu lassen. Für die Anpassung der Drehzahl unseres kleinen Lüfters bedienen wir uns der PWM-Funktion, die uns der Raspberry Pi zur Verfügung stellt.

Während oftmals für die Verringerung von Helligkeit, Geschwindigkeit etc. einfach die Spannung reduziert wird, bleibt bei der PWM der Spannungspegel konstant. Hier findet die Regelung über ein periodisches An- und Abschalten der Stromversorgung des Verbrauchers (Lüfters) statt. Je länger der Einschaltanteil in der Periode (Dutycycle) ist, desto schneller dreht der Lüfter. Das Prinzip erklärt Euch die folgende Grafik:


Frequ
Periode 0.1s
Periode
Dutycycle
Dutycycle

Damit wir nicht endlos mit Zeitschleifen arbeiten müssen, stellt uns Python für den Raspberry Pi schon PWM-Funktionen zur Verfügung.

Lüfterdrehzahl mit Python und PWM temperaturabhängig regeln

Nun wissen wir also, was PWM ist und wie es funktioniert. Damit kommen wir zu unserem kleinen Python-Programm. Der meiner Meinung nach ausreichend kommentierte Quellcode folgt jetzt.

```
#! /usr/bin/python3
# -*- coding: utf-8 -*-

import RPi.GPIO as GPIO
import time
import os

# Zählweise der Pins auf GPIO-Nummern festlegen
GPIO.setmode(GPIO.BCM);

# Warnmeldungen ausschalten
GPIO.setwarnings(False);

# Konstanten definieren
FAN_OUT_PIN = 13                    # Lüfter-Ausgang (FAN_OUT_PIN) ist GPIO33 (PIN33)

# GPIO Eingänge definieren
GPIO.setup(FAN_OUT_PIN, GPIO.OUT)       # definiert unseren Lüfter-Pin als Ausgang
GPIO.output(FAN_OUT_PIN, GPIO.LOW)      # setzt unseren Ausgang auf Low (0 =&gt; 0V)

# PWM-Channel initialisieren
pwm1 = GPIO.PWM(FAN_OUT_PIN, 100)       # Frequenz ab 10 Hz funktioniert ganz gut
pwm1.start(100.0)

# Funktionen definieren
def getCpuTemp(): # ermittelt die CPU-Temperatur und gibt sie als String zurück
    raw = os.popen("vcgencmd measure_temp").readline()
    cpuTemp = (raw.replace("temp=","").replace("'C\n",""))
    return(cpuTemp)

def setFanDuty(sfdc = 0.0): # Setzt den Dutycycle (in % von 0.0-100.0) für den Lüfter
    pwm1.ChangeDutyCycle(sfdc)
    return

def startFan(): # Setzt den Lüfter für 1s Anlaufzeit auf 100%
    fanOutput = 100.0
    setFanDuty(fanOutput)
    print("Anlaufzeit: 1s")
    time.sleep(1)
    return

# Hauptprogramm starten
cpuTemp = getCpuTemp()
print("CPU-Temperatur: " + str(cpuTemp) + " &deg;C")
```

```
fanOutput = 0.0

while True:
    cpuTemp = float(getCpuTemp())
    if cpuTemp &lt; 40.0:
        fanOutput = 0.0
    elif cpuTemp &gt; 59.9:
        fanOutput = 100.0
    elif cpuTemp &gt; 49.9:
        if fanOutput &lt; 80.0:
            startFan()
        fanOutput = 80.0
    elif cpuTemp &gt; 39.9:
        if fanOutput &lt; 66.6:
            startFan()
        fanOutput = 66.6
################################################## Die ON/OFF-Variante
#    if cpuTemp &gt; 45.0:
#        fanOutput = 100.0
#    else:
#        fanOutput = 0.0
##################################################
    setFanDuty(fanOutput)
    print("CPU-Temperatur: " + str(cpuTemp) + " &deg;C / Lüfterausgang: " + str(fanOutput) + "%")
    time.sleep(5) # 5Sekunden Pause vor dem nächsten Durchlauf

pwm1.stop()
```

Die komplette Datei könnt Ihr Euch (als ZIP-Archiv) unter dem folgenden Link herunterladen: fan_controller.py

Der Code lässt sicherlich Spielraum für Verbesserungen, sollte jedoch zunächst einmal funktionieren. Falls Ihr Fehler findet oder weltbewegende Änderungen vorschlagen möchtet, schickt mir einfach eine Mail.

Der kurze auskommentierte Bereich enthält noch eine ON/OFF-Variante, die ab 45 °C CPU-Temperatur den Lüfter voll zuschaltet. Das ginge natürlich auch mit Ein- und Ausschalten des GPIO-Ausganges, lässt so jedoch Möglichkeiten für geringere Lüfterdrehzahlen und damit höhere Haltbarkeit des Lüfters offen.

Noch ein kurzter Hinweis zum Thema Brushless-Lüfter und PWM. Moderne bürstenlose Lüfter (mit zwei Anschlüssen) sind eigentlich nicht dafür gemacht, über PWM angesteuert zu werden. Sie verfügen über einen eigenen kleinen Microcontroller, der das Drehfeld für den Antrieb erzeugt und so die Rotorwelle zum Drehen bringt. Trotzdem funktioniert die PWM-Steuerung ganz gut. Ihr dürft nur nicht erwarten, dass Ihr den Motor stufenlos bis auf einstellige Drehzahlbereiche herunterregeln könnt.

Meiner Erfahrung nach, läuft der Lüfter ab ca. 50% Dutycycle. Das ist sicherlich von Modell zu Modell unterschiedlich. Da müsst Ihr ein bisschen experimentieren. Da sich bei meinem Versuchsaufbau der Lüfter ab und zu weigerte anzuspringen, habe ich einfach die Anlauffunktion (startFan) mit eingebaut. Die lässt den Lüftermotor für eine Sekunde mit 100% laufen und regelt dann auf den gewünschten Wert herunter.

Ein weiteres Thema, dass ich ansprechen möchte, ist die Geräuschentwicklung, wenn der Lüfter mit weniger als 100% angesteuert wird. PWM erzeugt ja eine Quasi-Wechselspannung mit der Frequenz, die wir dem PWM-Controller zuweisen. In Verbindung mit den Spulen des Motors kommt es dabei zu dem Effekt, dass wir sozusagen die PWM-Frequenz, mit der der Motor angesteuert wird hören können. Dem kann man, wenn es denn stört, mit entsprechenden capazitiven und/oder induktiven Elementen entgegenwirken. Das Internet gibt Rat.

Falls Ihr einen Lüfter mittels externer Temperaturmessung regeln möchtet, findet Ihr hier noch ein kleines Tutorial

zu Abfrage eines LM75 Temperatursensors per I2C-Bus.

Ich hoffe, dass ich Euch ein bisschen inspirieren konnte und erkläre damit dieses Tutorial für beendet.