



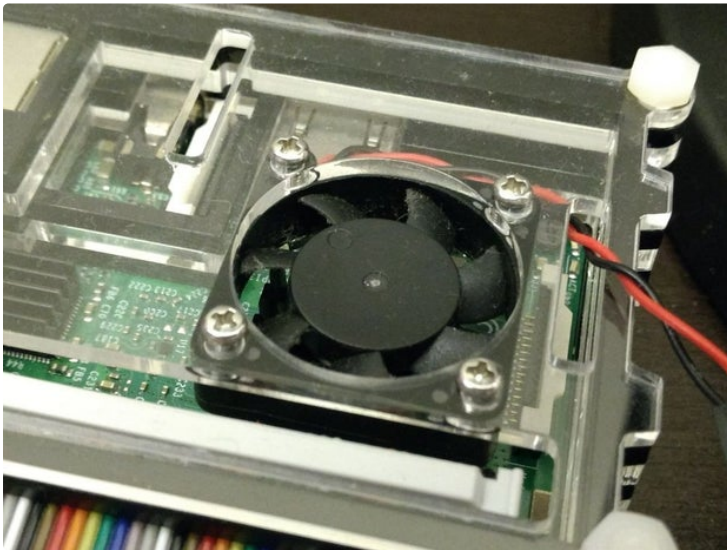
PWM Regulated Fan Based on CPU Temperature for Raspberry Pi



by Aerandir14

Many cases for Raspberry Pi come with a little 5V fan in order to help cooling the CPU. However, these fans are usually pretty noisy and many people plug it on the 3V3 pin to reduce the noise. These fans are usually rated for 200mA which is pretty high for the 3V3 regulator on the RPi. This project will teach you

how to regulate the fan speed based on CPU temperature. Unlike most of tutorials covering this subject, we won't only turn on or off the fan, but will control its speed like it's done on mainstream PC, using Python.



Step 1: Parts Needed

For this project, we will use only a few components that are usually included in electronics kits for hobbyist that you can find on Amazon, [like this one.](#)

- Raspberry Pi running Raspbian (but should work with other distribs).
- 5V Fan (but a 12V fan could be used with an adapted transistor and a 12V power supply).
- NPN transistor that supports at least 300mA, like a 2N2222A.
- 1K resistor.
- 1 diode.

Optional, to put the components inside the case (but not done yet):

- A little piece of protoboard, to solder the components.
 - Large heat shrink, to protect the board.
-

Step 2: Electrical Connections

Resistor can be plug in either way, but be careful about transistor's and diode's direction. Diode's cathode must be connected to the +5V (red) wire, and anode must be connected to the GND (black) wire. Check your transistor doc for Emitter, Base and Collector pins. Fan's ground must be connected to the Collector, and Rpi's ground must be connected to Emitter.

In order to control the fan, we need to use a transistor that will be used in **open collector configuration**. By doing this, we have a switch that will connect or disconnect the ground wire from the fan to the ground of the raspberry pi.

A **NPN BJT** transistor conducts depending on the current that flows in its gate. The current that will be allowed to flow from the collector (C) to the emitter (E) is:

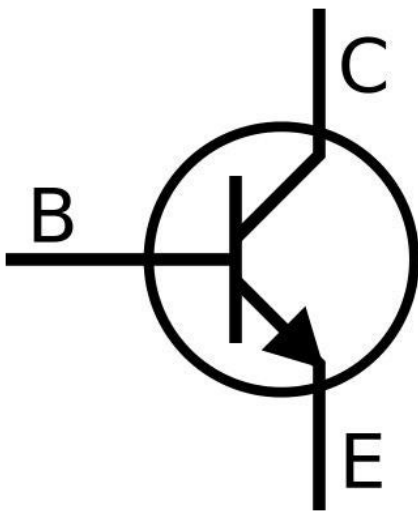
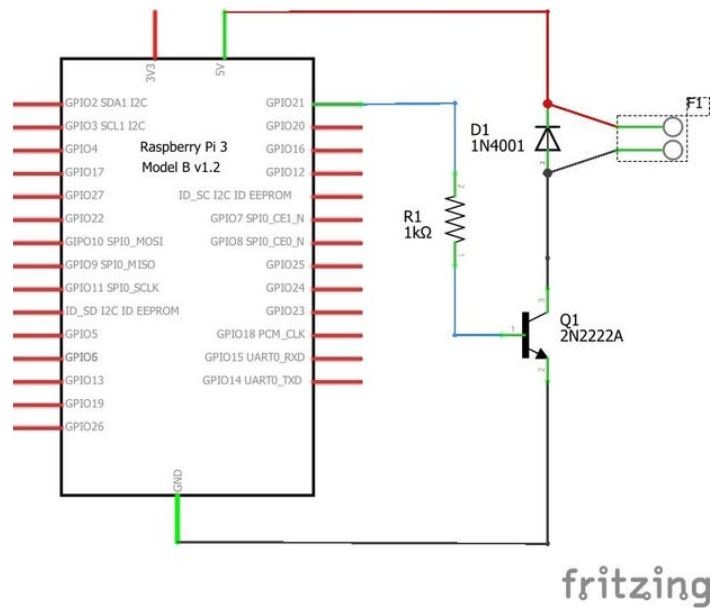
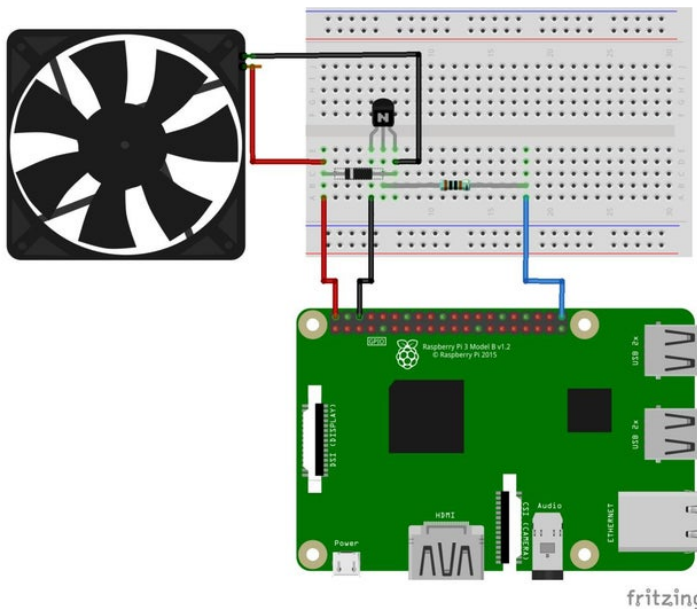
$$I_c = B * I_b$$

I_c is the current that flows through the collector the emitter, I_b is the current that flows through the base to the emitter, and B (beta) is a value depending on

each transistor. We approximate **B = 100**.

As our fan is rated as **200mA**, we need at least 2mA through the base of the transistor. The tension between the base and the emitter (V_{be}) is considered constant and **$V_{be} = 0,7V$** . This means that when the GPIO is on, we have **$3.3 - 0.7 = 2.6V$ at the resistor**. To have 2mA through that resistor, we need a resistor of, maximum, **$2.6 / 0.002 = 1300 \text{ ohm}$** . We use a resistor of **1000 ohm** to simplify and keep a margin of error. We will have 2.6mA through the GPIO pin which is totally safe.

As a fan is basically an electrical motor, it is an **inductive charge**. This means when the transistor stops conducting, the current in the fan will continue flowing as an inductive charge tries to keep the current constant. This would results in a high voltage on the ground pin of the fan and **could damage the transistor**. That's why we need a diode in parallel with the fan which will make the current flow constantly through the motor. This type of diode setup is called a **Flywheel diode**



Step 3: Program to Control the Fan Speed

To control fan speed, we use a software PWM signal from the RPi.GPIO library. A PWM Signal is well adapted to drive electric motors, as their reaction time is very high compared to the PWM frequency.


Use the `calib_fan.py` program to find the `FAN_MIN` value by running in the terminal:


```
python calib_fan.py
```

Check several values between 0 and 100% (should

be around 20%) and see what is the **minimum value for your fan to turn on**.

You can change the correspondence between temperature and fan speed at the beginning of the code. There must be as many `tempSteps` as `speedSteps` values. This is the method that is generally used in PC motherboards, moving points on a Temp / Speed 2-axis graph.

 <https://www.instructabl...> Download

 <https://www.instructabl...> Download

Step 4: Run the Program at Startup

To run the program automatically at startup, I made a bash script where I put all the programs I want to launch, and then I launch this bash script at startup with rc.locale

1. Create a directory `/home/pi/Scripts/` and place the `fan_ctrl.py` file inside that directory.
2. In the same directory, create a file named `launcher.sh` and copy the script bellow.
3. Edit the `/etc/rc.locale` file and add a new line before the "exit 0": `sudo sh`
`'/home/pi/Scripts/launcher.sh'`


launcher.sh script:

```
#!/bin/sh
#launcher.sh # navigate to home directory, then to this directory, then execute python script, then back home
locale
cd /
cd /home/pi/Scripts/
sudo python3 ./fan_ctrl.py &
cd /
```

If you want to use it with OSMC for example, you need to start it as a service with systemd.

1. Download the `fanctrl.service` file.
2. Check the path to your python file.
3. Place `fanctrl.service` in `/lib/systemd/system`.
4. Finally, enable the service with `sudo systemctl enable fanctrl.service`.

This method is safer, as the program will be automatically restarted if killed by the user or the system.

 <https://www.instructabl...> Download



Thanks for nice tutorial, can You please describe in more details how to modify components and schema to use 12V ? I have hifiBerry amp with 12V output and 12V 3 wires fan so I'd like to leverage them.



Hi,
A 2N2222A transistor should easily support a 12V. As long as the current specs of your fan are good, you should not have to modify anything (apart from taking power from the 12V output and using a 12V fan). As explained in another comment about 3 wires fan, you can use only 2 of the 3 as the 3rd wire is for speed reading.



Hi,
Thank you for your wonderful tutorial.

I found a little mistake inside the fan_ctrl.py script at line 46 and this mistake make the fan can't slow down.

So I changed it from:

- if abs(cpuTemp - cpuTempOld > hyst):

to:

+ if abs(cpuTemp - cpuTempOld) > hyst:

and everything work fine again!



Hi,

Sorry about this error and thanks for your help!

It should now be fixed :)



Thanks ReinforceZwei, I couldn't get my fan to spin down until I made the change you suggested.

Also, thanks Aerandir14 for the great tutorial.



Thanks!



Hi you mention we need at least 2mA through the base of the transistor. Why 2mA? Please and thanks for your help. Just trying to figure out the math



Hi,

The fan I used it rated for 200mA max and the transistor has a B value of 100.

What it means is in order to get 200mA through the transistor, it needs $200/100 = 2\text{mA}$ through the gate.

As written in the tutorial:

A NPN BJT

transistor conducts depending on the current that flows in its gate.

The current that will be allowed to flow from the collector (C) to the emitter (E) is:

$$I_c = B * I_b$$



I made this! It worked fine for about a year, but now it fails to start fanctrl.service on a Raspberry Pi

3. The log shows a segmentation fault:

Main process exited, code=killed, status=11/SEGV

Running the command-line:

~ \$ sudo python /home/pi/Scripts/fan_ctrl.py

Segmentation fault

I've tried getting latest updates.

I'm at a loss as to how to proceed.



Hi, thanks for your comment.

That's strange. I fixed some bugs right now, could you download the files and see if you still get the same issue?



Hi, thank you for the nice tutorial.

I have a fan which is rated at 0.07A, 0,35W. I am not able to control the fan with your setup. Do I need to change the resistor? what am I doing wrong?

Thanks and best regards

Chris



Hi,

There were a bug with the GPIO pin as reported by several people. It should be working now :)



the name of the program fan_ctrl.py not ctrl_fan.py, right?



Thanks, correction done :)



Very nice! I built it a works like a charm.

I found one small bug (other than the wrong pin): You forgot to put
CpuTempOld = CpuTemp
at the end of the loop.



Thanks you! It should be fixed now :)



For those running this setup on Raspberry Pi 4, please note that the GPIO PIN specified in the code should be 21 (not the default 24 came with the provided source code) - if you connect transistor's base to the very end pin near Ethernet adapter on the outside row like the picture shown in this post. See <https://www.raspberrypi.org/documentation/usage/gpio/> for the PIN number allocations for RPI 4.



Thanks! I fixed the code :)



I have a noctua fan (NF-A4x10 5V) with 3 wires. Can anyone tell me how to go about with the PWM wire? Do I need transistor and resistor? I could not find any infos how to wire it - well, maybe I have but as I am a total dimwit with electronics I just did not understand.....

Speaking of which: Using an RPi 3B, what GPIOs can or cannot be used for (soft?) PWM? For the sake of a small JST connector, I would like to use either GPIO02 (pin 03) or GPIO03 (pin 05)...hmmmmm



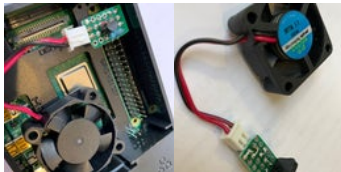
Hi,

3 wires computer fan don't have a PWM wire. The 3rd pin is for fan speed sensing. You'd need the 5V PWM version (4 pins) to be able to use this tutorial without the transistor. Even with a 4 pin fan, You would still need the resistor though. Noctua specifies a maximum current of 5mA in case of internal failure of the fan (https://noctua.at/media/wysiwyg/Noctua_PWM_specifications_white_paper.pdf). I'd try with a 1K resistor on the PWM wire and see how it goes.

Anyway, with a 3 pin wires, you have no choice but using an external resistor and transistor as explained in this tutorial.



My version works like a phat of sorts.... it only needs three pins after all, but I used a 4 pin socket so I know it will always be mounted at the very top left of the GPIO rack



Building my brand new OMV5 NAS with R pi 4



For the raspberry pi 3b+ you must modify the fan pin. it is 21 NOT 24 ;)



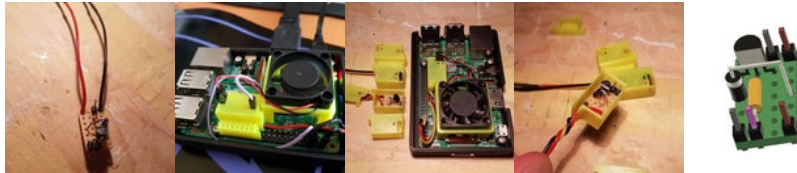


Built 4x of these for my various pies.

And also designed and 3d printed a few different small case designs for a little PCB case to protect it and ofc also the Pi it-self from any shorts.

Based my PCB layout on "dastasha"s variant, but had to swap the location of the Diode since my transistors were different.

Modified the python code to use GPIO pin 4 instead, to have all the cables close to each other.



I just wanted to tell you that this worked exactly as intended on my Raspberry Pi 4B & a 5V 200mA fan. Thanks!



Modified script and values. Temperature is stable and fan is quiet. Thank you!



I have modified your original script to better suite my intentions. I am sharing it, but if you don't like something I have done or preferred me not to share it feel free to delete and/or contact me. Great instructable! This prints out current speed(%) and cpu temp.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO
import time
import sys
import os
import signal
# Configuration
FAN_PIN = 14 # BCM pin used to drive transistor's base
WAIT_TIME = 5 # [s] Time to wait between each refresh
FAN_LSPD = 20 # [%] Fan minimum speed.
PWM_FREQ = 25 # [Hz] Change this value if fan has strange behavior
# Setup GPIO pin
GPIO.setmode(GPIO.BCM)
GPIO.setup(FAN_PIN, GPIO.OUT, initial=GPIO.LOW)
fan=GPIO.PWM(FAN_PIN,PWM_FREQ)
fan.start(0);
def getCPUtemperature():
    res = os.popen('/opt/vc/bin/vcgencmd measure_temp').readline()
    temp =(res.replace("temp=", "").replace("\n", ""))
    print("Temp is {0}°C".format(temp)) #Uncomment here for testing
    return temp
def getTEMP():
    CPU_temp = float(getCPUtemperature())
    return()
i = 0
hyst = 1 # Fan speed will change only if the difference of temperature is higher than hysteresis
tempSteps = [30, 35, 40, 45, 50, 60, 65, 70] # [°C] # Configurable temperature and fan speed steps
```



```

speedSteps = [0, 20, 40, 60, 80, 100, 100, 100] # [%]
cpuTempOld=0
fanSpeedOld=0
# We must set a speed value for each temperature step
if(len(speedSteps) != len(tempSteps)):
print("Numbers of temp steps and speed steps are different")
exit(0)
try:
while 1:
getTEMP()
time.sleep(5) # Read the temperature every 5 secs
cpuTempFile=open("/sys/class/thermal/thermal_zone0/temp","r")
cpuTemp=float(cpuTempFile.read())/1000
cpuTempFile.close() # Read CPU temperature
if(abs(cpuTemp-cpuTempOld > hyst)): # Calculate desired fan speed
if(cpuTemp < tempSteps[0]): # Below first value, fan will run at min speed.
fanSpeed = speedSteps[0]
elif(cpuTemp >= tempSteps[len(tempSteps)-1]): # Above last value, fan will run at max speed
fanSpeed = speedSteps[len(tempSteps)-1]
else: # If temperature is between 2 steps, fan speed is calculated by linear interpolation
for i in range(0,len(tempSteps)-1):
if((cpuTemp >= tempSteps[i]) and (cpuTemp < tempSteps[i+1])):
fanSpeed = round((speedSteps[i+1]-speedSteps[i])\
/(tempSteps[i+1]-tempSteps[i])\
*(cpuTemp-tempSteps[i])\
+speedSteps[i],1)
if((fanSpeed != fanSpeedOld) ):
if((fanSpeed != fanSpeedOld)\
and ((fanSpeed >= FAN_LSPD) or (fanSpeed == 0))):
fan.ChangeDutyCycle(fanSpeed)
fanSpeedOld = fanSpeed
print("Speed is {0}%".format(fanSpeed))
time.sleep(WAIT_TIME) # Wait until next refresh
except(KeyboardInterrupt): # If a keyboard interrupt occurs (ctrl + c), the GPIO is set to 0 and the
program exits.
print("Cancelled... Fan OFF")
GPIO.cleanup()
sys.exit()

```



I think we can use the function `numpy.interp(CPU_temp, temp_range, fan_power_range)` the code shall be short and more clear.

<https://docs.scipy.org/doc/numpy/reference/generated/numpy.interp.html>



Useful instruction and it works well?

(Code given does not correspond to scheme - GPIO24 in code, GPIO 21 on graphics).



Thx! Finally a quiet Nespi Case + !



My first solder job too quiet down a noisy fan ... great instructions, thx a lot!



Thanks for the article, I used your design with a cheap metal rPi case with a built-in fan, and it worked like a charm. I used a 470k resistor instead of 1k, but otherwise everything is the same. However, my fan makes a clicking noise on any PWM duty cycle setting lower than ~85%, so in my case I had to configure it to run 0-or-100 instead of variable speed. From what I could find online it's down to some fan motors just not playing well with PWM control.



Hi, looks great! Question; did you shrink the components and wires too? Of course in a way they aren't touching.. I was thinking of doing this also but wasn't sure if shrink over components is ok?



I did, it shouldn't get too hot - and it's right in the path of the airflow from the fan anyway



Hi! Thanks for your comment.

With a 470k resistor, you probably don't get enough current on the transistor's base to drive the fan current correctly.

About your clicking noise, try to adjust the PWM frequency. Try to lower it and see if it gets better :)



Mine does that too



Hi,

It should work well if the transistor allow it. What transistor will you be using ?

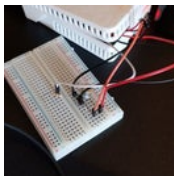
You should also lower a little bit the resistor value in order to get 300 mA through the transistor, but it also depends on the B factor of your transistor.



Hi I was planning to use the 2N2222A (800MA 40V) , that work? As far as the resistor I will go buy whatever I need, what do you suggest? Lastly the diode I have a 1N4001



Sweet! First draft



Thanks for the instructions

Had to change the code because the pin on my raspberry pi that goes on the (BCM) pin shown in your pictures matches pin 21 and not 24 as in the code. I read on the reference that you can map pins using BOARD mapping, would that be an improvement to avoid people getting wrong pins like I did?

Also, made a systemd service for the whole think, something like this:

[Unit]

Description=Fan Speed Controller

After=multi-user.target

```
[Service]
Restart=always
RestartSec=15
Type=simple
ExecStart=/usr/local/sbin/fan_ctrl.py
User=pi
```

```
[Install]
WantedBy=multi-user.target
```



My version was pretty quick and dirty with everything soldered to the transistor and lengths of Dupont wire. It's not particularly neat, but it lets me leave the JST connector attached to the fan leads, meaning I can take this off easily if I need to, and it leaves lots of room for air to circulate inside the case.



Hello. I am curious as to what the diode is used for? I've seen several sets of instructions that don't use them.



Hi,

The diode is used to protect the transistor. When the PWM signal is low, the fan works as a generator due to its inertia. This raises the voltage at the transistor's collector and can damage it. The diode allows the current to pass through it and go back to the positive terminal of the fan and pass through it again.



Thanks for the explanation.



Your code is nice, it helped me achieve what I wanted to do. From what I remember from high school computer class is most fans have a diode built in, so I'm gonna bank on that fact and not add a 30cent diode. The N2222, should on paper act as a resistor (I MIGHT BE WRONG), either way, the fans only gonna draw the amperage it needs. And that's lower than the GPIO capability. Also, I removed the use for launcher.sh, can you tell me why you used it when
`sudo python /home/pi/Scripts/fan_ctrl.py &`
would probably do the same thing.

I would really be interested in some criticism on my setup.

Thank you and ultimately, good job.



It's very easy to check if your fan does have a diode built-in or not, with a simple multimeter. I doubt it does.



Hey there, I've been looking for just this. Your code was perfect for me to learn, I changed some things, like GPIO17 and I removed the launcher script all together and edited the already existing "rc.local" not rc.locale. (because of my nespri+ case safe shutdown)

Alas it worked.

My question to you sir, is what are you trying to do with the launcher when, "`sudo python /home/pi/Scripts/fan_ctrl.py &`" can be placed at the top of the rc.local file.



Nice one! Thanks you sir.

Here I changed the transistor: NPN BC337-25 (had no other to use :P)

The fan start working at 50%.

I don't know if it will be enough for cooling the raspi if I run something heavy.

On the bright side, I have almost 0 noises :D



Hi! in step 4 is order 2 and 3 correct?



That's a neat simple setup :)