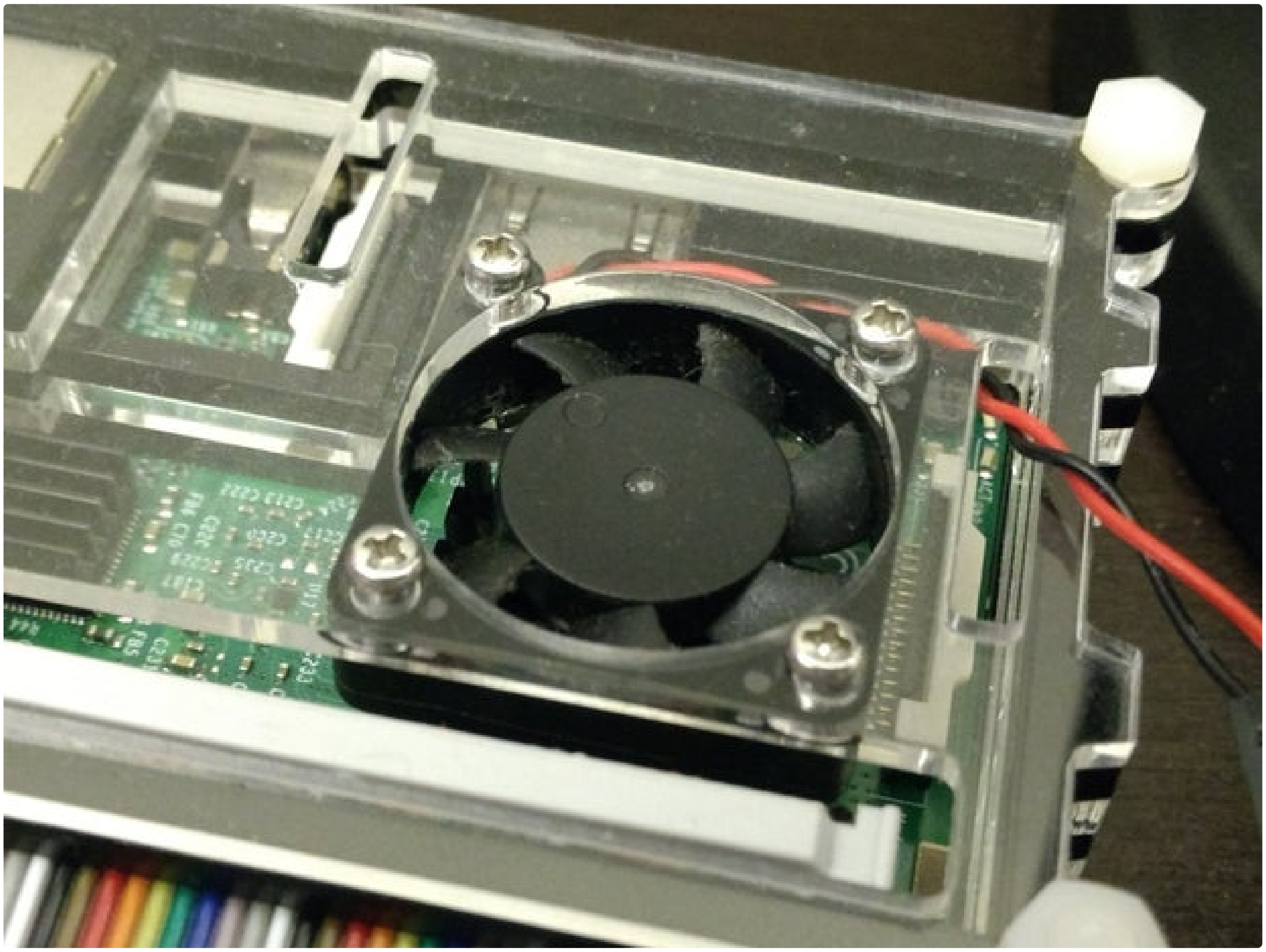
instructables

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.

For those who might not want to build there own circuit board, you can buy a ready-to-use board made by Jeremy Cook on tindie.com: https://www.tindie.com/products/jeremycook/ez-fan2-tiny-raspberry-pi-fan-controller/

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.

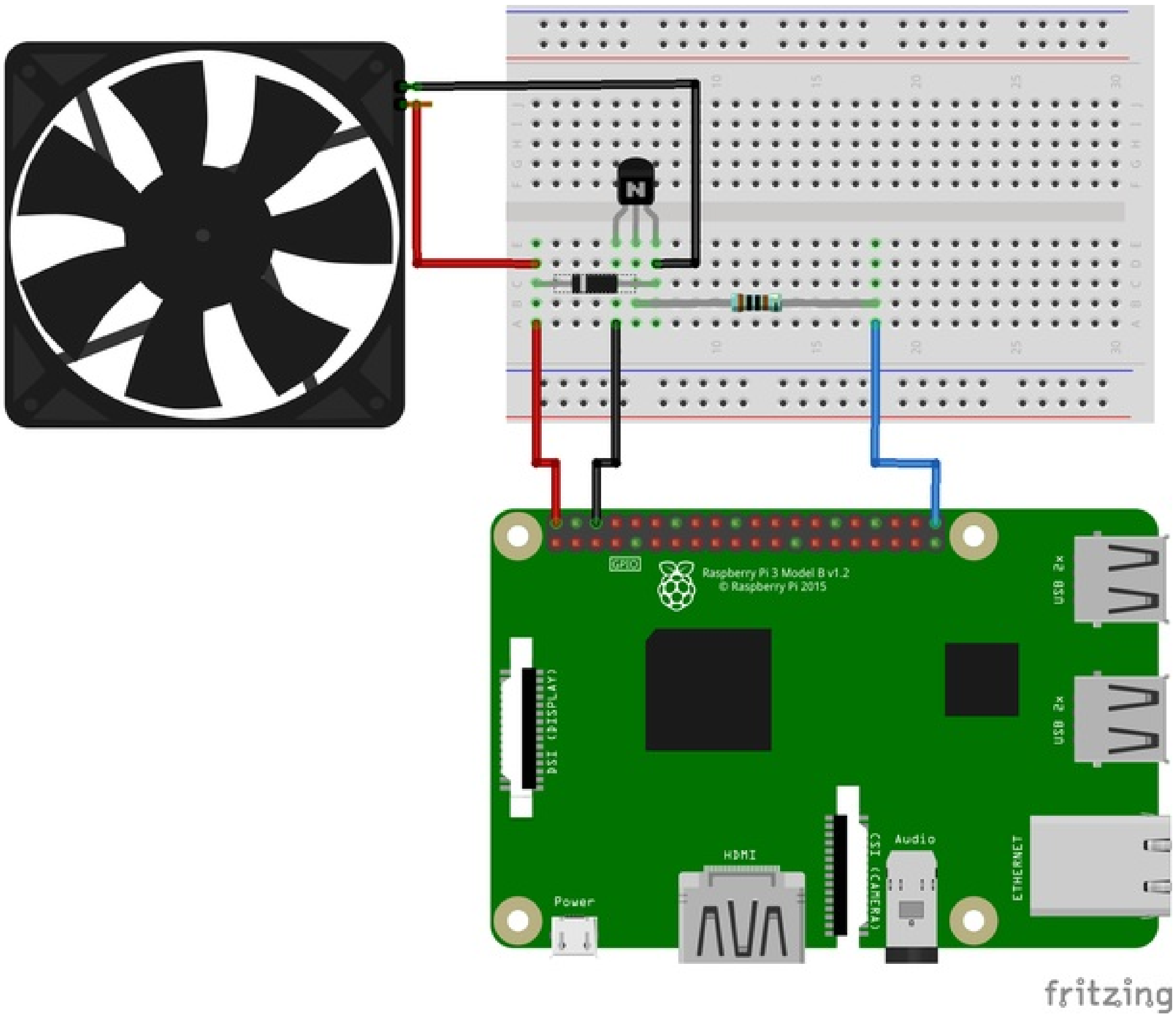
A **NPN BJT** transistor conducts depending on the current that flows in its gate. The current that will be allowed to flow from the collector (C) to the emitter (E) is:

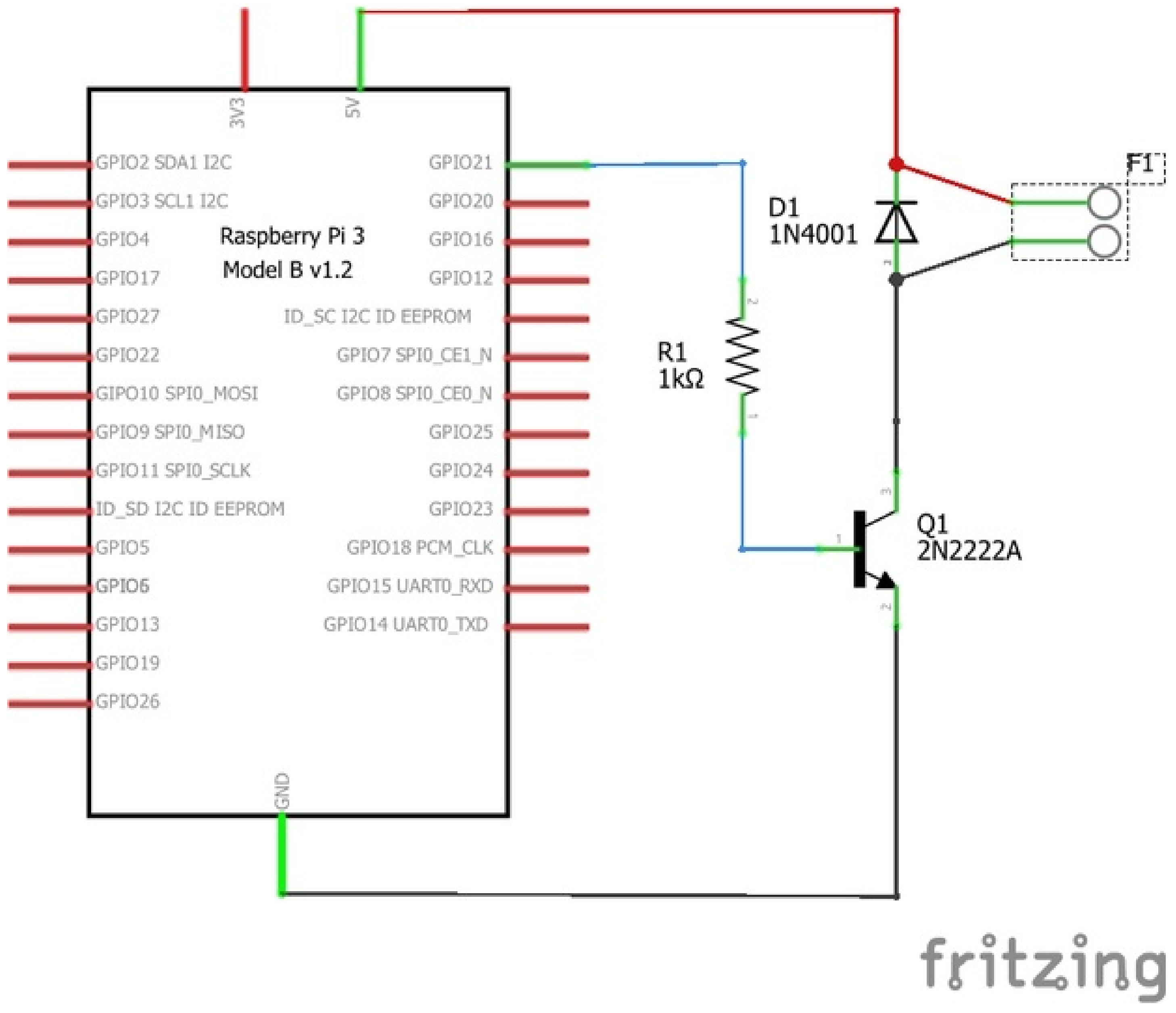
Ic = B * Ib

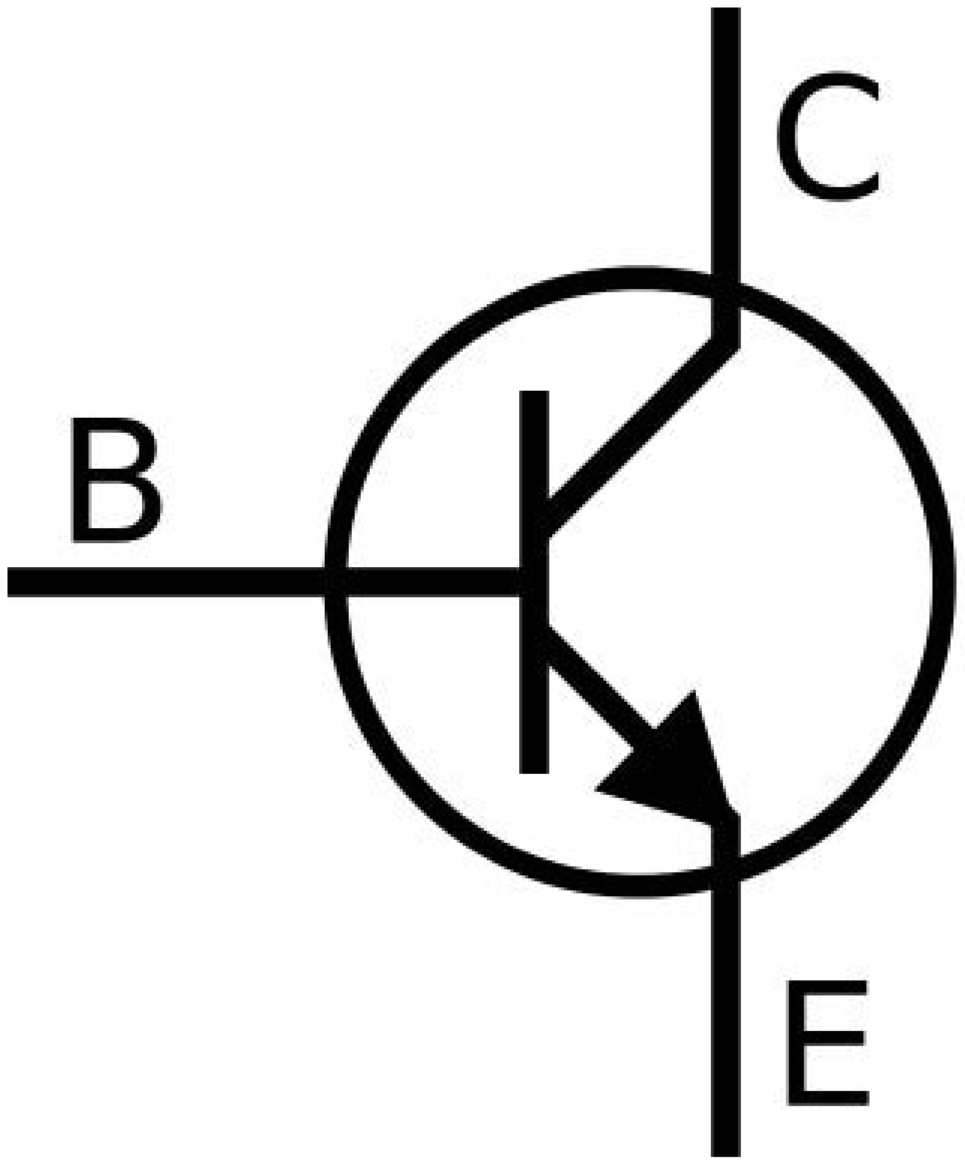
Ic is the current that flows through the collector the emitter, Ib is the current that flows through the base to the emitter, and B (beta) is a value depending on each transistor. We approximate **B = 100.**

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 (Vbe) is considered constant and **Vbe = 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 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**


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


3V3
5V
GPIO2 SDA1 I2C
GPIO3 SCL1 I2C
GPIO4
GPIO17
GPIO27
GPIO22
GIPO10 SPI0_MOSI
GPIO9 SPI0_MISO
GPIO11 SPI0_SCLK
ID_SD I2C ID EEPROM
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
Raspberry Pi 3
Model B v1.2
GPIO21
GPIO20
GPIO16
GPIO12
ID_SC I2C ID EEPROM
GPIO7 SPI0_CE1_N
GPIO8 SPI0_CE0_N
GPIO25
GPIO24
GPIO23
GPIO18 PCM_CLK
GPIO15 UART0_RXD
GPIO14 UART0_TXD
GND
D1
1N4001
R1
1kΩ
Q1
2N2222A
F1
fritzing



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.

Download
https://www.instructables.com/ORIG/FD9/ORSV/K437E94B/FD9ORSVK437E94B.py

Download
https://www.instructables.com/ORIG/FUH/U2I7/K5ADR365/FUHU2I7K5ADR365.py

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 /
```

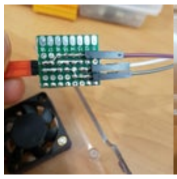
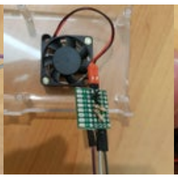
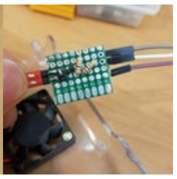
If you want to use it with OSMC for example, you need to start it as a service with systemd.

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.

Download
https://www.instructables.com/ORIG/FRE/066V/JCKUMX7E/FRE066VJCKUMX7E.service

Made it to cool down my home assistant server on a raspberry pi 3b+.

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 im 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 thats lower then 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.

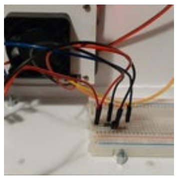
thank you and ultimatly, good job.

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.

Hi, thanks for your instructions. I'm beginner on raspberry and I learnt a lot with this. I have same question with Matt. Why do you need to create Launcher.sh? I also think that "sudo python /home/pi/Scripts/fan_ctrl.py &" can do same thing, isn't it true?

Can you please let me know the python code to interface dht11 sensor and dc motor to raspberry pi and write a python program to perform following using PWM technique
1.Rotate the motor with 20% speed if temperature is between 20 to 30 degree Celsius
2.1.Rotate the motor with 60% speed if temperature is between 31 to 40 degree Celsius
3.1.Rotate the motor with 90% speed if temperature is between 41 to 50 degree Celsius

Need to make a small PCB with connectors now.



Hello. The first value of the array speedSteps should be FAN_MIN, to avoid undervoltage to the fan. I corrected and it works ok.
Thanks for your work!

Hello
im really begginer can someone help me with code?

i want the fan works like this

temp 45 degree 40% speed

temp 55 degree 60% speed

temp 65 degree 100% speed

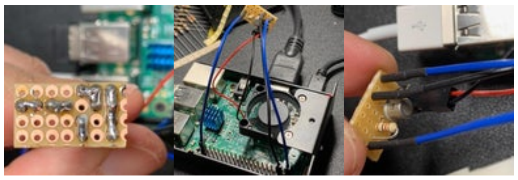
i use the pin 36
thanks

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO
import time
import sys
GPIO.setwarnings(False)
# Configuration
FAN_PIN = 36 # Physical pin used to drive transistor's base
WAIT_TIME = 1 # [s] Time to wait between each refresh
FAN_MIN = 40 # [%] Fan minimum speed.
PWM_FREQ = 25 # [Hz] Change this value if fan has strange behavior
# Configurable temperature and fan speed steps
tempSteps = [45, 55, 65] # [°C]
```

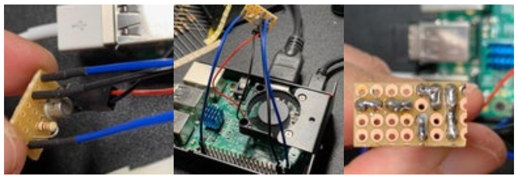
```
speedSteps = [40, 60, 100] # [%]
# Fan speed will change only of the difference of temperature is higher than hysteresis
hyst = 1
# Setup GPIO pin
GPIO.setmode(GPIO.BOARD)
GPIO.setup(FAN_PIN, GPIO.OUT, initial=GPIO.LOW)
fan = GPIO.PWM(FAN_PIN, PWM_FREQ)
fan.start(0)
i = 0
cpuTemp = 0
fanSpeed = 0
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:
# Read CPU temperature
cpuTempFile = open("/sys/class/thermal/thermal_zone0/temp", "r")
cpuTemp = float(cpuTempFile.read()) / 1000
cpuTempFile.close()
# Calculate desired fan speed
if abs(cpuTemp - cpuTempOld) > hyst:
# Below first value, fan will run at min speed.
if cpuTemp < tempSteps[0]:
fanSpeed = speedSteps[0]
# Above last value, fan will run at max speed
elif cpuTemp >= tempSteps[len(tempSteps) - 1]:
fanSpeed = speedSteps[len(tempSteps) - 1]
# If temperature is between 2 steps, fan speed is calculated by linear interpolation
else:
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_MIN or fanSpeed == 0)):
fan.ChangeDutyCycle(fanSpeed)
fanSpeedOld = fanSpeed
cpuTempOld = cpuTemp
# Wait until next refresh
time.sleep(WAIT_TIME)
# If a keyboard interrupt occurs (ctrl + c), the GPIO is set to 0 and the program exits.
except KeyboardInterrupt:
print("Fan ctrl interrupted by keyboard")
GPIO.cleanup()
sys.exit()
```

thanks for replay
i have twister os can you help me how to run the fan on it?
rubashkaic5@gmail.com

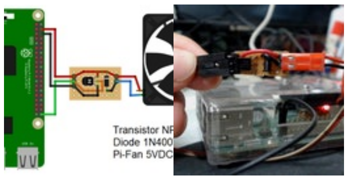
Thanks a lot! This was fun to make and easy, thanks to your good description. However, I changed the software to only switch on/off at certain levels, because by fan does not operate well with PWM.



Thank you very much! This was a nice little project and a lot of fun. Although I modified the script to only switch on and off because my fan acted a little weird when using PWM.



Thanks a lot. Following your tutorial, I would like to share my project on github
https://github.com/TaLucGiaHoang/raspberrypi-cpu-c...



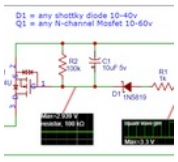
Just for anyone checking this out, it is not wired correctly and will not work. The collector pin on the transistor (the top of the three in the diagram) should not be connected to the (+) red wire. Instead it should connect to the ground wire coming out of the diode connected to the fan.

There also has to be a current limiting resistor where the orange wire is now so you don't blow the resistor by accident.

With the circuit in the diagram the fan will never turn on as the ground of the fan is not electrically connected to anything.

Great guide, Thanks :D
I used a different transistor, a 9013 (because it's all I had available) but I adjusted the calculations according to it's datasheet and used your formula as a guide.
Really happy with it.

Nice and neat DIY circuit, thank you for sharing!
...a similar to this design using the Pi-OS standard 2wire Fan control via GPIO #14 is this (no script needed, >Raspberry Pi configuration/ Performance/Fan [Enable]):


D1 = any shottky diode 10-40v
Q1 = any N-channel Mosfet 10-60v
R2
100k
C1
10uF 5v
R1
1k
D1
1N5819
Max=3.3 V

Could I use a 10uF 50v electrolitic capacitor and a 2N2222A transistor for this? Also, what is the purpose of Q1? Is it two diodes? What happened to the protective diode in the original project? Could I smooth out the PWN signal using the original circuit from the post and just adding R2, C1 and D1 from yours?

Sorry if these are stupid questions, I bought my first breadboard just for this project and I'm still figuring everything out.

Hi, there are no stupid questions.
"Could I use a 10uF 5v electrolytic capacitor and a 2N2222A transistor for this?" Yes you can.
Q1 is N-Ch mosfet with very low internal resistor, same results as 2N2222
No two diodes, D1 is the only diode; the PWM is just an input sign.
You don't need protection diode for DC brushless motors, they have internal circuit with protection.
"Could I smooth out the PWN signal using the original circuit from the post and just adding R2, C1 and D1 from yours?" of course you can (you don't need the R2, this resistor needed only with FET transistor).
R1 is optional/nice to have (current limiter/protection for the GPIO)
as per my comment this design is just an alternative, mosfet based, to the working design already posted.

Thank you!

Hi there , thanks this is super simple and does exactly what i want

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.

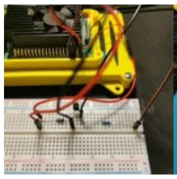
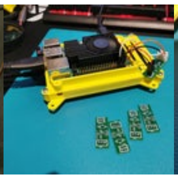
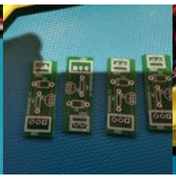
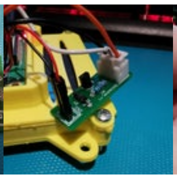
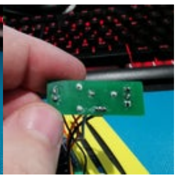
https://www.delta-fan.com/Download/Spec/ASB0305HP-00CP4.pdf In this specification, the resistance value between v+ and pwm is about 200KOhm.

I don't think it is necessary to add an 1K resistor if you are using a PWM fan, the resistance value between PWM wire and GND should be well above 1MOhm.

Hi,
I already have this noctua fan (the three pin). Once built the circuit it seems to work ...but in my case it only starts spinning at 100%. I am using the 1k resistor
¿Could It be something related to the circuit or should I use another resistor?
Thanks for sharing the project, quite nice!

Somebody else reported being able to spin that fan at 70% min, I think it doesn't handle low speeds

This is cool :)

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("'C\n",""))
print("Temp is {0}Â°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 of the difference of temperature is higher than hysteresis
tempSteps = [30, 35, 40, 45, 50, 60, 65, 70] # [Ã‚Â°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()
```

hi
code not working
please help me

caliskanmuhittin@gmail.com

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

I did this with a 2N2222A (metal version) 5v 3d printer fan, 240ohm resistor (generic diode). Work from 20 to 100% Thanks.

I built this and it runs - although I like the look of the protoboard version posted in the images.

To get the systemd.service running on Raspberry Pi OS (64-bit in my case, but shouldn't matter), the fanctrl.service (in /lib/systemd/system/fanctrl.service) needs to be modified as follows:

```
[Unit]
Description=PWM Fan Control for Raspberry Pi
After=ssh.service
[Service]
Type=simple
User=pi
ExecStart=/usr/bin/python3 /home/pi/Scripts/fan_ctrl.py
Restart=always
[Install]
WantedBy=default.target
```

Start with:

```
sudo systemctl start fanctrl.service
```

Check it is running (you should see it listed) with:

```
systemctl | grep fan
```

```
fanctrl.service loaded active running PWM Fan Control for Raspberry Pi
```

Permanently enable with:

```
sudo systemctl enable fanctrl.service
```

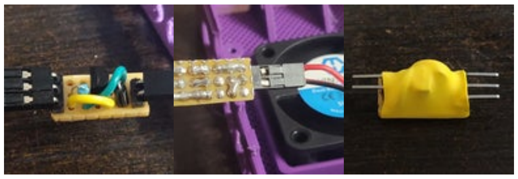
Test with a heavy load - I used Java Minecraft and it turned on during map generation and turned off after quitting, but it never became loud (although I have not overclocked this Pi).

Is it possible to use a PNP transistor instead of NPN?
Naturally changing the electric scheme.
I have in BC 139, I think it has a similar performances.

My first use of soldering since 20 years :) works like charm, from 30%

I also found a link to this fella who has a YouTube video (I'm not sure he's aware of these scripts) and he has created gerber files to have PCBs printed and populated with these components:
https://github.com/JeremySCook/RaspberryPi-Fan-Control

I had fun fitting these components and making the traces with the prototyping board. Thanks for posting these wonderful scripts!



Changed the transistor for a hefty BD139 and lowered the resistor a bit based on measeurements taken from the prototype. Integrating the final build inside the case turned out to be the trickiest part, but I'm pleased with the result.

I know this thread is a bit old, but I am hoping someone can answer my question. Is it possible to set this up to control 2 fans, not just one? Can I just use a second GPIO pin? I have built an NAS server with 2 RAID hard drives and all enclosed inside a vintage NES console. I have one fan on the Raspberry Pi itself to cool it and a second fan attached to the NES to help circulate air and keep the hard drives cool. I would really like to have them both controlled by the CPU temp because they are very loud. I currently have the case fan on the 3.3v pin to help with the noise. Is it possible to setup both of these fans to be controlled by the PWM software described in the guide?

Sure, but it would be probably better to use the same fans.
This should do the trick :

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

import RPi.GPIO as GPIO
import time
import sys

# Configuration
FAN_PIN_1 = 21 # BCM pin used to drive 1st fan transistor's base
FAN_PIN_2 = 22 # BCM pin used to drive 2nd fan transistor's base
WAIT_TIME = 1 # [s] Time to wait between each refresh
FAN_MIN = 20 # [%] Fan minimum speed.
PWM_FREQ = 25 # [Hz] Change this value if fan has strange behavior

# Configurable temperature and fan speed steps
tempSteps = [50, 70] # [°C]
speedSteps = [0, 100] # [%]

# Fan speed will change only of the difference of temperature is higher than hysteresis
hyst = 1

# Setup GPIO pin
GPIO.setmode(GPIO.BCM)
GPIO.setup(FAN_PIN_1, GPIO.OUT, initial=GPIO.LOW)
fan1 = GPIO.PWM(FAN_PIN_1, PWM_FREQ)
GPIO.setup(FAN_PIN_2, GPIO.OUT, initial=GPIO.LOW)
fan2 = GPIO.PWM(FAN_PIN_2, PWM_FREQ)
fan1.start(0)
fan2.start(0)


i = 0
cpuTemp = 0
fanSpeed = 0
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:
# Read CPU temperature
cpuTempFile = open("/sys/class/thermal/thermal_zone0/temp", "r")
cpuTemp = float(cpuTempFile.read()) / 1000
cpuTempFile.close()

# Calculate desired fan speed
if abs(cpuTemp - cpuTempOld) > hyst:
```

```
# Below first value, fan will run at min speed.
if cpuTemp < tempSteps[0]:
fanSpeed = speedSteps[0]
# Above last value, fan will run at max speed
elif cpuTemp >= tempSteps[len(tempSteps) - 1]:
fanSpeed = speedSteps[len(tempSteps) - 1]
# If temperature is between 2 steps, fan speed is calculated by linear interpolation
else:
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_MIN or fanSpeed == 0)):
fan1.ChangeDutyCycle(fanSpeed)
fan2.ChangeDutyCycle(fanSpeed)
fanSpeedOld = fanSpeed
cpuTempOld = cpuTemp

# Wait until next refresh
time.sleep(WAIT_TIME)


# If a keyboard interrupt occurs (ctrl + c), the GPIO is set to 0 and the program exits.
except KeyboardInterrupt:
print("Fan ctrl interrupted by keyboard")
GPIO.cleanup()
sys.exit()
```

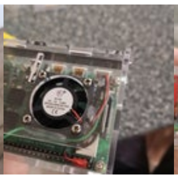
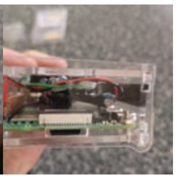
I have a laptop fan that not starting on low pmw, speed, i have to help him with my finger to get it running. On boot, its start on full speed.
My question, how can i give him a bigger speed to start before the loop begin?
I changed the value of fa.start to 100 but nothing, added before while fan.ChangeDutyCycle(100) but again, nothing. It needs a little push to spin up on lowest values that are set in the script. then, its works flawless.

Hi,
Try that just before the while loop starts:

```
...
try:
fan.ChangeDutyCycle(100)
time.sleep(2)
while 1:
...
```

Thanks for sharing.

Great circuit

Hello, i should have every materials needed except for the fan which is 5V 130mA. Should everything be okay or should i increase resistor's resistance? Based on the math, i should have maximum 2K Ohms right? 2.6V÷0.13A÷100. Also, when you said you're getting a 1K resistor to anticipate room for error, should i lower 2K resistor to get the same effect? -im will be using IN4007 diode and 2N2222 (without A) transistor by the way.

Hi,
You did the math well, a 2kohm resistor should be good for you. The margin of error is to use a lower resistor so you are sure the full current can pass through the transistor. If you take a higher resistor, the current passing through the transistor from collector to emitter would be lower (as the current through base would be lower) and your fan could not spin at max speed even with 100% pwm signal.
The downside of a lower resistor is that it drains more amps from the Raspberry Pi GPIO. As long as it stays bellow the maximum rated current for GPIOs, it's fine. But you use less power with fewer amps from the GPIO.

The components your have should do the job :)

Thank sir! This tutorial still the best one as you include explained what materials are needed and why are they needed. Thanks!

Hello, what is the program you are using on the screenshots above? It looks damn interesting!

Look for EveryCircuit in Play Store, not sure if it's available in ios.

Hello,
I'm currently planning to use a fan that is rated as 5V / 0.007A.
Can I still use your instructable to make the setup work?

Edit: After some math I think I might need a 37k resistor. Is this correct?

Made an awesome addition to my Octoprint Pi :D

My first small electronics project. Everything worked well. Just adjusted a few parameters for it and let it run. Definitely quieter now. Thank!