Post 21 Raspberry Pi Foyer Slideshow

Contents

OS Maintenance

To update your system, including the bootloader:

```
sudo apt update
sudo apt full-upgrade
sudo apt autoremove
sudo reboot
```

Notes

Editing System Files: Sudo -e NAME

Software and Scripts

File Structure

Slides: home/pi/Pictures/SPSlides

Special Slides: /home/pi/Pictures/Add in February (Black History Month)

Old Slide Shows: /home/pi/Pictures/Post Spinners (as storage permits)

Software that needs to be installed

FEH

Desktop Configuration File

Slideshow - Used to manually start the slideshow from the desktop

```
[Desktop Entry]
Name=Slideshow
Comment=THis is to run home/pi/runslideshow.sh
Icon=/usr/share/pixmaps/openbox.xpm
Exec=/home/pi/runslideshow.sh
Type=Application
Encoding=UTF-8
Terminal=false
Categories=None;
```

Slideshow Shell Script (.sh)

/home/pi/runslideshow.sh

```
 #!/bin/sh
cd /home/pi/Pictures/SPSlides
feh -Z -F -D 20 --hide-pointer --auto-rotate
# -Z --auto-zoom Zoom pictures to screen size in fullscreen / fixed geometry mode.
# -F --fullscreen
# -D --slideshow-delay float seconds
```

Autostart Mechanism

/home/pi/.config/lxsession/LXDE-pi/autostart   (plain text document)

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
#@xscreensaver -no-splashpoint-rpi
@xset s noblank
@xset s off
@xset -dpms
@/bin/sh /home/pi/runslideshow.sh
```

Disable Screen Blanking

There is no powersave or sleep mode. Screen blanking turns the screen black after 10 min… nothing else.

To disable, add the following 3 lines to autostart:

```
nano /home/pi/.config/lxsession/LXDE-pi/autostart
        @xset s noblank
        @xset s off
        @xset -dpms
```

-OR-

Install xscreensaver and set mode to [no blanking]

Parameters and Pins

Raspberry pi Configuration
- Hostname raspberrypi
- User pi
- password: Mighty21!

Preferences / Raspberry Pi Configuration
- Display/Disable Screen Blanking
- Interfaces Enable SSH and VNC
- Set Timezone, etc.

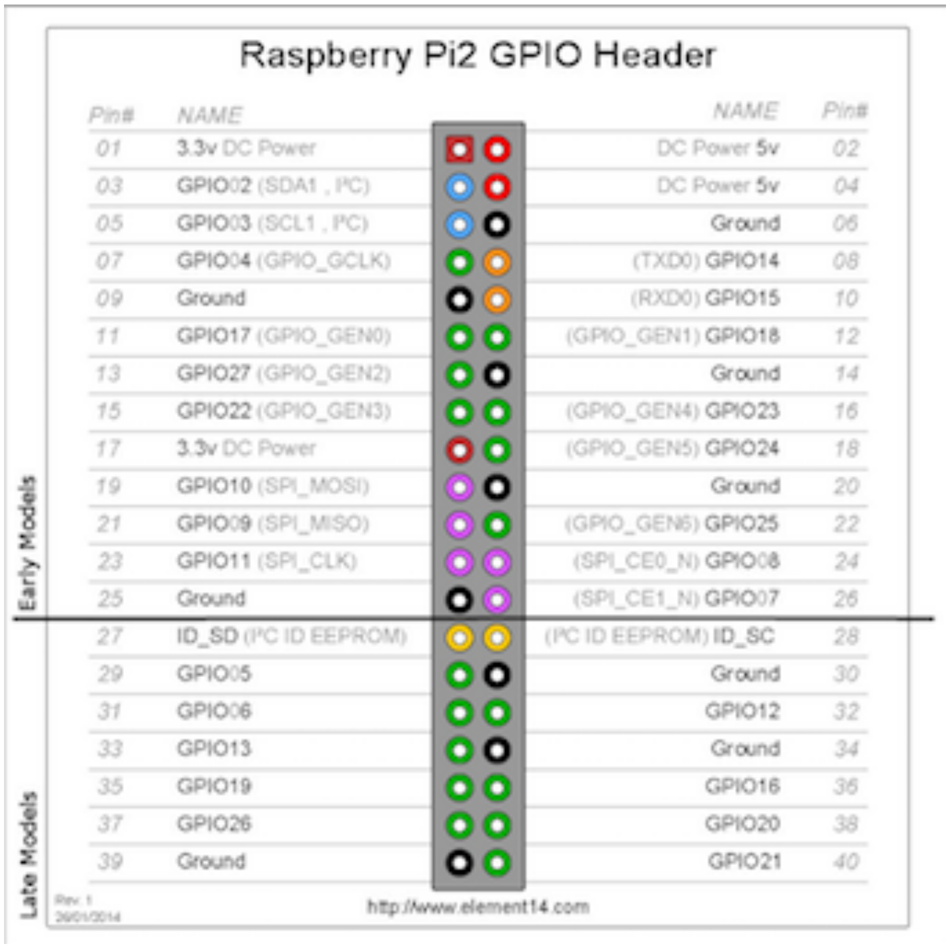
Appearance Settings
- Uncheck Desktop Wastebasket and Mounted Disks
- Install Legion Desktop Picture

Pins

Fan
- Red Pin 2 / Black Pin 9 / Yellow Pin 11

Power Up/Down Switch:
- Pins 5 and 6 (Polarity does not matter)

Power LED
- Pin 8 Red (TXD)
- Pin 14 Black (GND)



Raspberry Pi2 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Early Models
Late Models

Rev. 1
30/01/2014

http://www.element14.com

Power Button

https://howchoo.com/g/mwnlytk3zmm/how-to-add-a-power-button-to-your-raspberry-pi

Create the script: sudo nano listen-for-shutdown.py

Paste the following code into that file

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#!/usr/bin/env python

import RPi.GPIO as GPIO
import subprocess


GPIO.setmode(GPIO.BCM)
GPIO.setup(3, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.wait_for_edge(3, GPIO.FALLING)

subprocess.call(['shutdown', '-h', 'now'], shell=False)
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Place the script in /usr/local/bin and make it executable:

```
sudo mv listen-for-shutdown.py /usr/local/bin/
sudo chmod +x /usr/local/bin/listen-for-shutdown.py
```

A script called listen-for-shutdown.sh that will start/stop our service.

```
sudo nano listen-for-shutdown.sh
```

Enter the following code in that file and save it:

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#! /bin/sh

### BEGIN INIT INFO
# Provides:          listen-for-shutdown.py
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5

# Default-Stop:      0 1 6
### END INIT INFO

# If you want a command to always run, put it here

# Carry out specific functions when asked to by the system
case "$1" in
  start)
    echo "Starting listen-for-shutdown.py"
    /usr/local/bin/listen-for-shutdown.py &
    ;;
  stop)
    echo "Stopping listen-for-shutdown.py"
    pkill -f /usr/local/bin/listen-for-shutdown.py
    ;;
  *)
    echo "Usage: /etc/init.d/listen-for-shutdown.sh {start|stop}"
    exit 1
    ;;
esac

exit 0
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Place this file in /etc/init.d and make it executable.

```
sudo mv listen-for-shutdown.sh /etc/init.d/
sudo chmod +x /etc/init.d/listen-for-shutdown.sh
```

Register the script to run on boot.

```
sudo update-rc.d listen-for-shutdown.sh defaults
```

Start it with:

```
sudo /etc/init.d/listen-for-shutdown.sh start
```

LED Power Light

adding Pi LED status indicators

https://howchoo.com/g/ytzjyzy4m2e/build-a-simple-raspberry-pi-led-power-status-indicator

The LED Is connected to your Pi's TxD pin, which monitors the serial console. The LED will flicker a tad while booting, stay solid while your Pi is running, and turn off when it's safe to remove power.
Pros: Simplicity. No code is needed and it just sort of works. Also, this is a great foray into the hardware portion of your Pi.
Cons: Limited to providing information about when the Pi is on or off—a very binary solution.

Enable the GPIO serial port

Newer versions of Raspbian (May 2016 and later) have the GPIO serial port disabled by default; the end result is your LED will not light up! Luckily, enabling it is super easy.

Edit your /boot/config.txt file and add the following line:

```
enable_uart=1
```

You can edit this file by connecting to your Pi via SSH or by putting the SD card into your computer and editing the file directly. This file is accessible from the SD card.

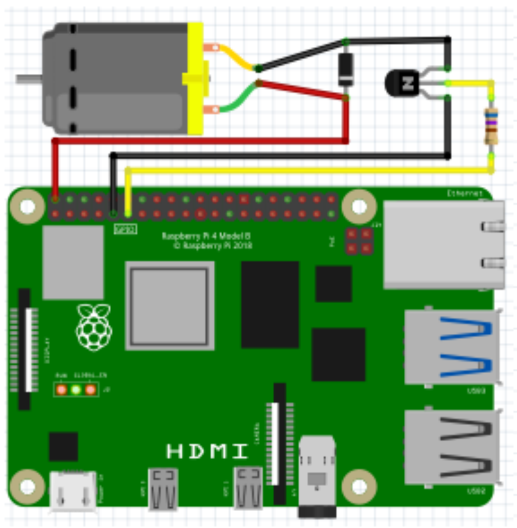
Post 21 Raspberry Pi 4B Fan Control

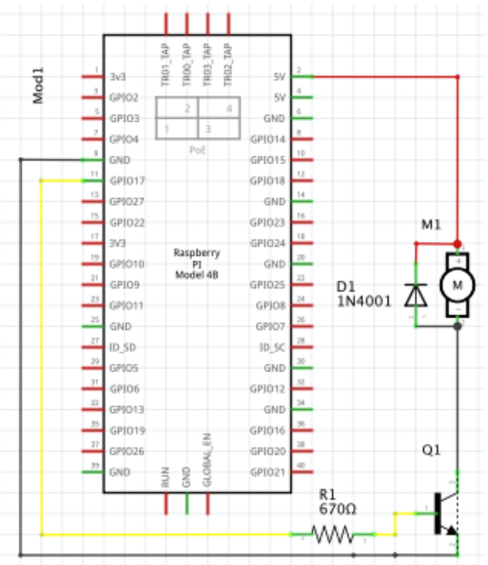
https://howchoo.com/g/ote2mjkzzta/control-raspberry-pi-fan-temperature-python

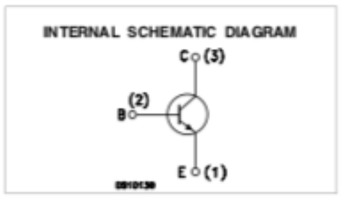
Intermittent Fan (based on temperature)

To make the fan function run intermittently as well, I used the bash script shown above, set up to run via cron once every minute. When triggered, it uses **vcgencmd measure_temp** to get the temperature of the Raspberry Pi's processor. It then compares this temperature using an if/then/else statement to either turn the fan on with the line **gpio -g write 3 1** or off with **gpio -g write 3 0**. It's not the most responsive solution, able to turn on or off only once per minute, but it's much simpler than anything else I've seen so far program-wise.
To directly power the fan itself I used a 2N2222 NPN transistor through one of the Pi's 5v pins. Here GPIO pin 3 is fed to the transistor's base, allowing current to flow through the fan, then through the transistor's collector and emitter, and finally to ground. A resistor is used between the GPIO and base to limit the current output [to the transistor]. I also added a flyback diode to the design to account for voltage spikes when the fan is switched off, though it's optional, and isn't actually used in my current setup.

A Simple PCB







(Transistor numbering (1,2,3) in the above sketch is unusual, a limitation of the RPi fzbz I used)
From top to bottom in the picture and diagram it should be:

Q1-1 Collector
Q1-2 Base
Q1-3 Emitter

D1 1N4001 Diode - Flyback Diode - Protection from BEMF when fan is turned off
R1 670Ω Resistor - Protect Transistor
Q1 2N3904 Transistor - Switch

1N4001 Diode Features

- Average forward current is 1A
- Non-repetitive Peak current is 30A
- Reverse current is 5uA.
- RMS reverse voltage is 35V
- Peak repetitive Reverse voltage is 50V

2N3904 SMALL SIGNAL NPN TRANSISTOR

SILICON EPITAXIAL PLANAR NPN TRANSISTOR
TO-92 PACKAGE SUITABLE FOR THROUGH-HOLE PCB ASSEMBLY
THE PNP COMPLEMENTARY TYPE IS 2N3906

APPLICATIONS
WELL SUITABLE FOR TV AND HOME APPLIANCE EQUIPMENT
SMALL LOAD SWITCH TRANSISTOR WITH HIGH GAIN AND LOW SATURATION VOLTAGE

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
VCBO	Collector-Base Voltage (IE = 0)	60	V
VCEO	Collector-Emitter Voltage (IB = 0)	40	V
VEBO	Emitter-Base Voltage (IC = 0)	6	V
IC	Collector Current	200	mA
Ptot	o Total Dissipation at TC = 25 C	625	mW
Tstg	Storage Temperature	-65 to 150	oC
Tj	Max. Operating Junction Temp	150	oC

https://howchoo.com/g/ote2mjkzzta/control-raspberry-pi-fan-temperature-python

Write the fan controller code

Continuously monitors the core temperature and turns on the fan when the temperature reaches a certain threshold.

To create this file, run:

```
nano fancontrol.py
```

Add the following to the file, save, and exit:

++

```
#!/usr/bin/env python3

import subprocess
import time

from gpiozero import OutputDevice


ON_THRESHOLD = 65  # (degrees Celsius) Fan kicks on at this temperature.
OFF_THRESHOLD = 55  # (degress Celsius) Fan shuts off at this temperature.
SLEEP_INTERVAL = 5  # (seconds) How often we check the core temperature.
GPIO_PIN = 17  # Which GPIO pin you're using to control the fan.


def get_temp():
    """Get the core temperature.
    Run a shell script to get the core temp and parse the output.
    Raises:
        RuntimeError: if response cannot be parsed.
    Returns:
        float: The core temperature in degrees Celsius.
    """
    output = subprocess.run(['vcgencmd', 'measure_temp'], capture_output=True)
    temp_str = output.stdout.decode()
    try:
        return float(temp_str.split('=')[1].split('\'')[0])
    except (IndexError, ValueError):
        raise RuntimeError('Could not parse temperature output.')


if __name__ == '__main__':
    # Validate the on and off thresholds
    if OFF_THRESHOLD >= ON_THRESHOLD:
        raise RuntimeError('OFF_THRESHOLD must be less than ON_THRESHOLD')

    fan = OutputDevice(GPIO_PIN)

    while True:
        temp = get_temp()

        # Start the fan if the temperature has reached the limit and the fan
        # isn't already running.
        # NOTE: `fan.value` returns 1 for "on" and 0 for "off"
        if temp > ON_THRESHOLD and not fan.value:
            fan.on()

        # Stop the fan if the fan is running and the temperature has dropped
        # to 10 degrees below the limit.
        elif fan.value and temp < OFF_THRESHOLD:
            fan.off()

        time.sleep(SLEEP_INTERVAL)
```

++

Move the script to /usr/local/bin, and make it executable.

```
sudo mv fancontrol.py /usr/local/bin/
sudo chmod +x /usr/local/bin/fancontrol.py
```

Execute the fan controller code on boot

Create a shell script that will execute on boot and launch our script.

Create a file called fancontrol.sh and add the following:

```
nano fancontrol.sh
```

++

```
#! /bin/sh

### BEGIN INIT INFO
# Provides:          fancontrol.py
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
### END INIT INFO

# Carry out specific functions when asked to by the system
case "$1" in
  start)
    echo "Starting fancontrol.py"
    /usr/local/bin/fancontrol.py &
    ;;
  stop)
    echo "Stopping fancontrol.py"
    pkill -f /usr/local/bin/fancontrol.py
    ;;
  *)
    echo "Usage: /etc/init.d/fancontrol.sh {start|stop}"
    exit 1
    ;;
esac

exit 0
```

++

Move this file to /etc/init.d, and make it executable:

```
sudo mv fancontrol.sh /etc/init.d/
sudo chmod +x /etc/init.d/fancontrol.sh
```

Register the script to run on boot:

```
sudo update-rc.d fancontrol.sh defaults
```

Either restart your machine, or kick this off manually since it won't already be running:

sudo reboot **or** sudo /etc/init.d/fancontrol.sh start

===

Further Explanations:

https://www.embedded-computing.com/guest-blogs/raspberry-pi-cooling-fan-control-with-bash-scripting

===

Summary of Commands: Copied from Terminal screen

```
pi@Post21RPi:~ $ sudo nano listen-for-shutdown.py
pi@Post21RPi:~ $ sudo mv listen-for-shutdown.py /usr/local/bin/
pi@Post21RPi:~ $ sudo chmod +x /usr/local/bin/listen-for-shutdown.py
pi@Post21RPi:~ $ sudo nano listen-for-shutdown.sh
pi@Post21RPi:~ $ sudo mv listen-for-shutdown.sh /etc/init.d/
pi@Post21RPi:~ $ sudo chmod +x /etc/init.d/listen-for-shutdown.sh
pi@Post21RPi:~ $ sudo update-rc.d listen-for-shutdown.sh defaults
pi@Post21RPi:~ $ nano fancontrol.py
pi@Post21RPi:~ $ sudo mv fancontrol.py /usr/local/bin/
pi@Post21RPi:~ $ sudo chmod +x /usr/local/bin/fancontrol.py
pi@Post21RPi:~ $ nano fancontrol.sh
pi@Post21RPi:~ $ sudo mv fancontrol.sh /etc/init.d/
pi@Post21RPi:~ $ sudo chmod +x /etc/init.d/fancontrol.sh
pi@Post21RPi:~ $ sudo update-rc.d fancontrol.sh defaults
pi@Post21RPi:~ $ sudo reboot
```