

## Raspberry Pi boot self-starting setting

There are many ways to start the Raspberry Pi boot. Choose one of them (new .desktop file) as a demonstration.

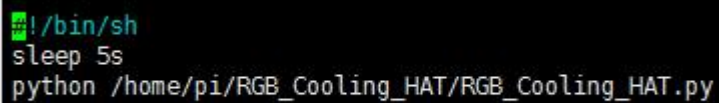
### 1. Add boot self-starting

#### 1.1 New startup script start.sh

```
nano /home/pi/RGB_Cooling_HAT/start.sh
```

Input following content:

```
#!/bin/sh  
sleep 5s  
python /home/pi/RGB_Cooling_HAT/RGB_Cooling_HAT.py
```



```
#!/bin/sh  
sleep 5s  
python /home/pi/RGB_Cooling_HAT/RGB_Cooling_HAT.py
```

Press **ctrl+X**, press **Y** to save, press **Enter**.

### 2. New create boot startup program

#### 2.1 Input following command to open .config folder

```
cd /home/pi/.config
```

#### 2.2 Input following command to new create autostart folder

```
mkdir autostart
```

#### 2.3 Input following command to enter autostart folder

```
cd autostart
```

#### 2.4 Input following command to new create shortcut for self-starting

```
nano start.desktop
```

Input following content:

```
[Desktop Entry]  
Type=Application  
Exec=sh /home/pi/RGB_Cooling_HAT/start.sh
```

Press **ctrl+X**, press **Y** to save, press **Enter**.

Exec=Start command.

!!!Note:

Since this self-starting method needs to be started after the desktop is started, the start up will be slower. If it is found that it cannot be self-started after adding, please check if there is a **##** in front

of `hdmi_force_hotplug=1` in the `/boot/config.txt` file. If there is a # number, please delete the # number. The picture will prevail.

```
# uncomment to force a console size. By default it will be display's size minus
# overscan.
#framebuffer_width=1280
#framebuffer_height=720

# uncomment if hdmi display is not detected and composite is being output
hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (this will force VGA)
#hdmi_group=1
#hdmi_mode=1
```

### 3. Restart Raspberry Pi

After restarting, the `temp_control` program will start automatically, and the fan, RGB light and oled screen will have corresponding responses.

Input following command to restart Raspberry Pi:

`sudo reboot`

### 4. Exit the program

Since the self-starting program is running in the background, we cannot directly exit the program in the open terminal. If you need to modify the program, but the background process interferes with our debugging results, so you need to see the process number (PID) of the daemon, and then end this process.

4.1 Input `top` in the terminal to open the process list.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1018	pi	20	0	22132	13304	6024	S	2.0	0.2	0:03.92	python
3406	pi	20	0	9280	3144	2656	R	1.0	0.0	0:00.08	top
31	root	20	0	0	0	0	I	0.3	0.0	0:00.12	kworker/0:1-events_freezable
136	root	20	0	0	0	0	I	0.3	0.0	0:00.01	kworker/u8:2-flush-179:0
1	root	20	0	165172	9360	7024	S	0.0	0.1	0:03.07	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0-pm
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-mmc_complete
7	root	20	0	0	0	0	I	0.0	0.0	0:00.20	kworker/u8:0-events_unbound
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	S	0.0	0.0	0:00.06	ksoftirqd/0
10	root	20	0	0	0	0	I	0.0	0.0	0:00.21	rcu_preempt
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
14	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/1
15	root	20	0	0	0	0	S	0.0	0.0	0:00.03	ksoftirqd/1
16	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0-events
17	root	0	-20	0	0	0	I	0.0	0.0	0:00.04	kworker/1:0H-kblockd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2
19	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/2
20	root	20	0	0	0	0	S	0.0	0.0	0:00.05	ksoftirqd/2
21	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/2:0-rcu_gp
22	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/2:0H
23	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/3
24	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/3
25	root	20	0	0	0	0	S	0.0	0.0	0:00.02	ksoftirqd/3

Sometimes it may take a while for the system to queue up processes that take up a lot of CPU resources. From the picture we can see the boot **temp\_control** program, its PID is 718, so we kill this process number, temp\_control program will not run in the background.

Press **Ctrl+C** to exit top.

#### 4.2 Input following command to end the process

**sudo kill -9 PID**

For example: In the above case, we can run the **sudo kill -9 718** command to end the temp\_control process running in the background. If you run it again, it will be prompted that the process does not exist.

```
pi@raspberrypi:~ $ sudo kill -9 718
pi@raspberrypi:~ $ sudo kill -9 718
kill: (718): No such process
pi@raspberrypi:~ $
```

#### 4.3 Restart the background operation

If we have finished the process running in the background, but we want to restart the background process.

First method: restart the Raspberry Pi,

Second method: Add an "&" to the running program.

For example, we still run the **temp\_control** program in the background:

We need to input following command to enter the target folder (based on the location of the file saved by the individual)

**cd ~/temp\_control**

**./temp\_control &**

```
pi@raspberrypi:~/RGB_Cooling_HAT $ python RGB_Cooling_HAT.py &
[1] 31904
pi@raspberrypi:~/RGB_Cooling_HAT $ idle:401 total:402
usageRate:0
idle:400 total:400
usageRate:0
idle:401 total:403
usageRate:0
idle:401 total:403
usageRate:0
idle:401 total:403
usageRate:0
idle:401 total:403
usageRate:0
idle:401 total:403
usageRate:0
```

At this point, the system will prompt the PID of this process (1015).

Press Ctrl+C again. You can see that the terminal can enter other commands, and the program is running in the background.