

# 8.Set boot self-start

---

## 1.Add auto-start at boot

### 1.1 New create start.sh

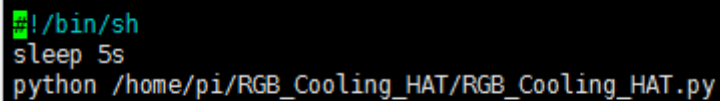
```
nano /home/pi/RGB_Cooling_HAT/start.sh
```

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, and then press "Enter key".

### 1.2 Create a new startup program

Enter the following command to open the .config folder

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

Create a new autostart folder. If it already exists, please ignore this step.

```
mkdir autostart
```

Enter the autostart folder

```
cd autostart
```

Create a new self-starting shortcut

```
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, and then press "Enter key".

Where Exec=start command.

Since this self-starting method requires the desktop to be started before it can be started, the startup will be slow. If you find that it cannot start automatically after adding it, please check whether there is a # in front of `hdmi_force_hotplug=1` in the `/boot/config.txt` file. If there is a # sign, please delete the # sign. Subject to the picture.

```
# 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
```

## 2.Reboot Raspberry Pi

Enter the following command in the terminal to restart the Raspberry Pi. After the restart, the `RGB_Cooling_HAT.py` program will automatically start, and the fan, RGB lights and OLED screen will respond accordingly.

```
sudo reboot
```

## 3.Exit code

Since the self-starting program runs in the background, we cannot exit the program directly from the open terminal. What should we do if we need to modify the program, but the background process interferes with our debugging results?

Of course, you need to check the process ID (PID) of the background program and then end the process.

3.1 Enter 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 you may need to wait for a while, and the system will queue processes that occupy higher CPU resources to the top. From the picture, we can see that the python program that starts automatically after booting has a PID of 1018, so we kill this process number and the RGB\_Cooling\_HAT.py program will no longer run in the background.

Press Ctrl+C to exit top.

### 3.2 End the process

```
sudo kill -9 PID
```

For example: In the above situation, we run the `sudo kill -9 1018` command to end the RGB\_Cooling\_HAT.py process running in the background. If you run it again, you 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:~ $
```

### 3.3 Restart background operation

If we have ended the process running in the background, but do not want to restart the background program, the first method is to restart the Raspberry Pi, but this seems to be a waste of time, and every inch of time is worth every inch of money. So we use the second method: add an `&` symbol after the running program.

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

First enter the target folder (depending on the location of the personal saved files)

```
cd ~/RGB_Cooling_HAT
```

Add an ampersand at the end of the command to indicate running in the background

```
python RGB_Cooling_HAT.py &
```

```
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 time, the system will prompt the PID of this process (1015).

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

