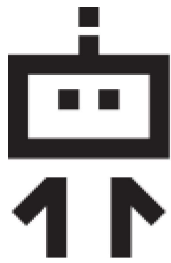



Sign In (/my-account?redirect_to=https%3A%2F%2Fwww.dexterindustries.com%2Fhowto%2Frun-a-program-on-your-raspberry-pi-at-startup%2F)



DEXTER
INDUSTRIES



(<https://www.dexterindustries.com/my-account/cart/>) 

[Home \(https://www.dexterindustries.com/\)](https://www.dexterindustries.com/)

[Robots](#)

\$0.00 (0)



[Projects \(https://studio.dexterindustries.com/\)](https://studio.dexterindustries.com/)

[Shop \(https://www.dexterindustries.com/shop/\)](https://www.dexterindustries.com/shop/)

[Education \(https://www.dexterindustries.com/dextered/\)](https://www.dexterindustries.com/dextered/)

[Blog \(/blog\)](/blog/)

[Forum \(http://forum.dexterindustries.com/categories\)](http://forum.dexterindustries.com/categories)

Five Ways To Run a Program On Your Raspberry Pi At Star

FIVE WAYS TO RUN A PROGRAM ON YOUR RASPBERRY PI AT STARTUP

Run a Program On Your Raspberry Pi At Startup

In this tutorial we show you five ways you can run a program on your Raspberry Pi at startup. The five

methods that are available to run a program at boot are:

- rc.local
- .bashrc
- init.d tab
- systemd
- crontab

Sample Program

You can use any program that you want to run at boot; for this tutorial we are using a sample python program which will speak at the startup of Raspberry Pi. This sample program will use the **Espeak** package to make the Raspberry pi speak **“Welcome to the world of Robots”**.



If you do not have the **Espeak** package installed , run the following in terminal to install:

```
sudo apt-get install espeak
```

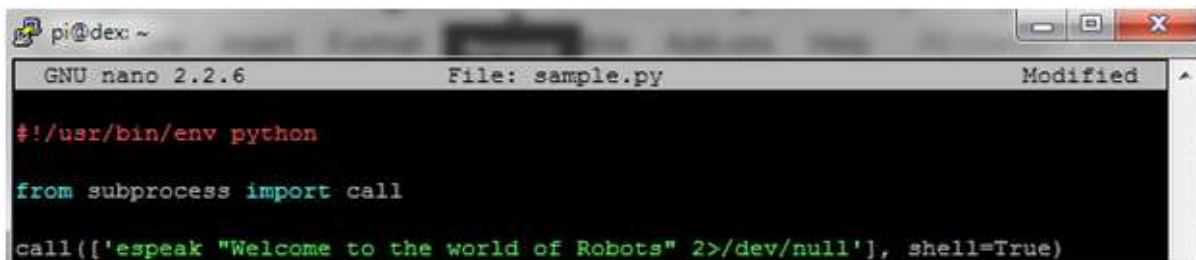
To learn more about how to get the Raspberry Pi speak, we have a tutorial (/howto/make-your-raspberry-pi-speak/)here.

In the /home/pi directory, open a file for editing:

```
sudo nano sample.py
```

And enter the following code and save it (press CTRL+X and enter Y).

```
#!/usr/bin/env python
from subprocess import call
call(['espeak "Welcome to the world of Robots" 2>/dev/null'], shell=True)
```



(<https://32414320wji53mwwch1u68ce-wpengine.netdna-ssl.com/wp-content/uploads/2014/11/test.png>)

Method 1: rc.local

The first method to run a program on your Raspberry Pi (/shop/raspberry-pi-3/) at startup is to use the file **rc.local**. In order to have a command or program run when the Pi boots, you can add commands to the **rc.local** file. This is especially useful if you want to power up your Pi in headless mode (that is without a connected monitor), and have it run a program without configuration or a manual start.

Editing rc.local

On your Pi, edit the file /etc/rc.local using the editor of your choice. You must edit it with root permissions:

```
sudo nano /etc/rc.local
```

Add commands to execute the python program, preferably using absolute referencing of the file location (complete file path are preferred). Be sure to leave the line **exit 0** at the end, then save the file and exit. In nano, to exit, type Ctrl-x, and then Y.

```
# First get update the hostname.  
sudo bash /home/pi/di update/Raspbian_For_Robots/upd_script/rc.sh  
sudo python /home/pi/sample.py &  
exit 0
```

(<https://32414320wji53mwwch1u68ce-wpengine.netdna-ssl.com/wp-content/uploads/2014/11/test1.png>)

If your program runs continuously (runs an infinite loop) or is likely not to exit, you must be sure to fork the process by adding an ampersand ("&") to the end of the command, like:

```
sudo python /home/pi/sample.py &
```

The Pi will run this program at bootup, and before other services are started. If you don't include the ampersand and if your program runs continuously, the Pi will not complete its boot process. The ampersand allows the command to run in a separate process and continue booting with the main process running.

Now reboot the Pi to test it:

```
sudo reboot
```

Hints

Also, be sure to reference absolute file names rather than relative to your home folder. For example use ``/home/pi/myscript.py`` instead of ``myscript.py``.

If you add a script into `/etc/rc.local`, it is added to the boot sequence. If your code gets stuck then the boot sequence cannot proceed. So be careful as to which code you are trying to run at boot and test the code a couple of times. You can also get the script's output and error written to a text file (say `log.txt`) and use it to debug.

```
sudo python /home/pi/sample.py & > /home/pi/Desktop/log.txt 2>&1
```

Method 2: `.bashrc`

The second method to run a program on your Raspberry Pi (<http://www.dexterindustries.com/shop/raspberry-pi-3/>) at startup is to modify the `.bashrc` file. With the `.bashrc` method, your python program will run on boot and also every time when a new terminal is opened, or when a new SSH connection is made. Put your command at the bottom of `/home/pi/.bashrc`. The program can be aborted with 'ctrl-c' while it is running!

```
sudo nano /home/pi/.bashrc
```

Go to the last line of the script and add:

```
echo Running at boot
sudo python /home/pi/sample.py
```

The echo statement above is used to show that the commands in `.bashrc` file are executed on bootup as well as connecting to bash console.

```

alias la='ls -A'
alias l='ls -CF'

# Alias Definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

xhost +
echo Running at boot
sudo python /home/pi/sample.py

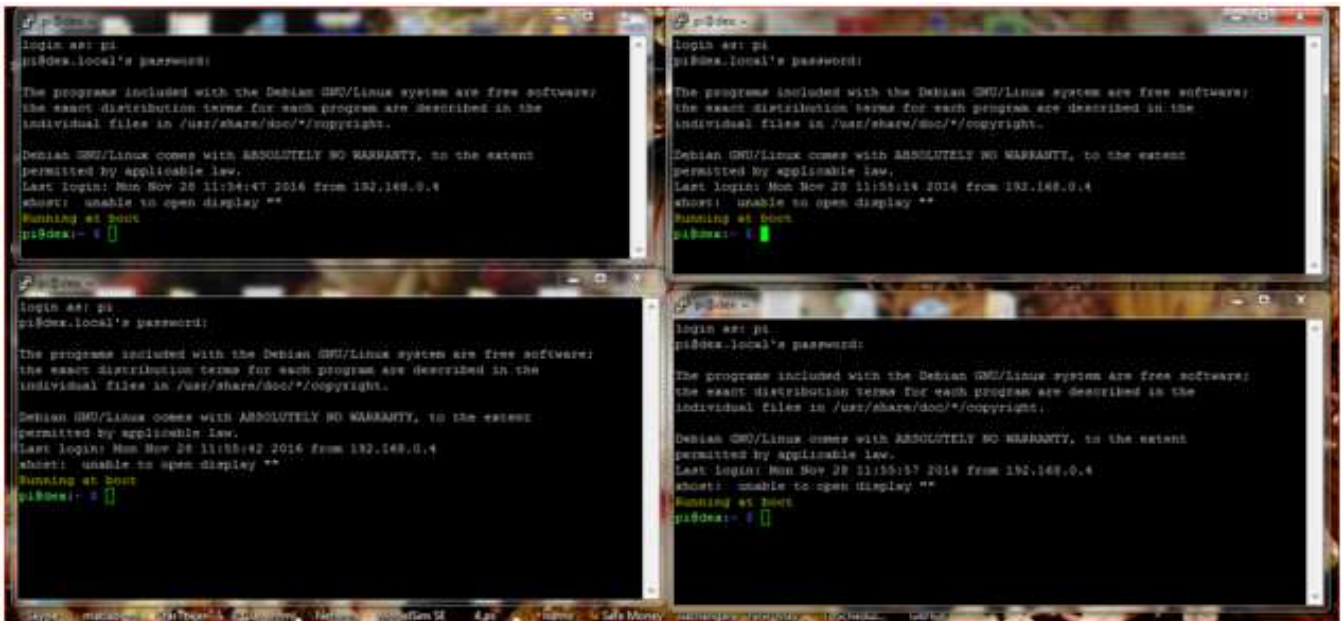
```

(<https://32414320wji53mwwch1u68ce-wpengine.netdna-ssl.com/wp-content/uploads/2014/11/test3.png>)

Now reboot the Pi to hear the Pi speak at startup.

```
sudo reboot
```

The below image shows that the commands added to .bashrc file get executed even while opening a new terminal.



(<https://32414320wji53mwwch1u68ce-wpengine.netdna-ssl.com/wp-content/uploads/2014/11/test4.png>)

Method 3: init.d directory

The third method to run a program on your Raspberry Pi

(<http://www.dexterindustries.com/shop/raspberry-pi-3/>) at startup is to add the program (to be run on boot) to the **/etc/init.d** directory. This directory contains the scripts which are started during the boot process (in addition, all programs here are executed when you shutdown or reboot the system).

Add the program to be run at startup to the init.d directory using the following lines:

```
sudo cp /home/pi/sample.py /etc/init.d/
```

Move to the init directory and open the sample script

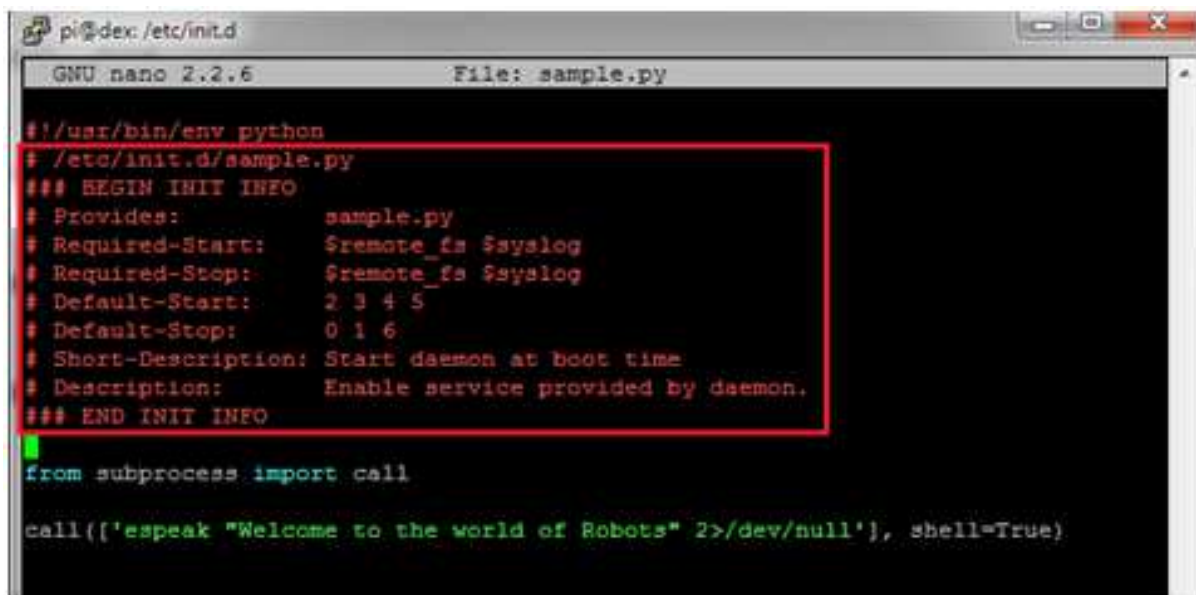
```
cd /etc/init.d  
sudo nano sample.py
```

Add the following lines to the sample script to make it a Linux Standard Base (LSB) (A standard for software system structure, including the filesystem hierarchy used in the **Linux** operating system) init script.

```
# /etc/init.d/sample.py
### BEGIN INIT INFO
# Provides:          sample.py
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start daemon at boot time
# Description:       Enable service provided by daemon.
### END INIT INFO
```

init.d scripts require the above runtime dependencies to be documented so that it is possible to verify the current boot order, the order the boot using these dependencies, and run boot scripts in parallel to speed up the boot process.

You can learn to write init.d scripts following the guide here (<https://wiki.debian.org/LSBInitScripts>).



```
pi@dex: /etc/init.d
GNU nano 2.2.6      File: sample.py

#!/usr/bin/env python
# /etc/init.d/sample.py
### BEGIN INIT INFO
# Provides:          sample.py
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start daemon at boot time
# Description:       Enable service provided by daemon.
### END INIT INFO

from subprocess import call

call(['espeak "Welcome to the world of Robots" 2>/dev/null'], shell=True)
```

(<https://32414320wji53mwwch1u68ce-wpengine.netdna-ssl.com/wp-content/uploads/2014/11/test5.png>)

Make the sample script in the init directory executable by changing its permission.

```
sudo chmod +x sample.py
```

Run this command:

```
sudo update-rc.d sample.py defaults
```

Now reboot to hear the Pi speak at startup.

```
sudo reboot
```

Method 4: SYSTEMD

The fourth method to run a program on your Raspberry Pi at startup is to use the **systemd** files. **systemd** provides a standard process for controlling what programs run when a Linux system boots up. Note that **systemd** is available only from the Jessie versions of Raspbian OS.

Step 1– Create A Unit File

Open a sample unit file using the command as shown below:

```
sudo nano /lib/systemd/system/sample.service
```

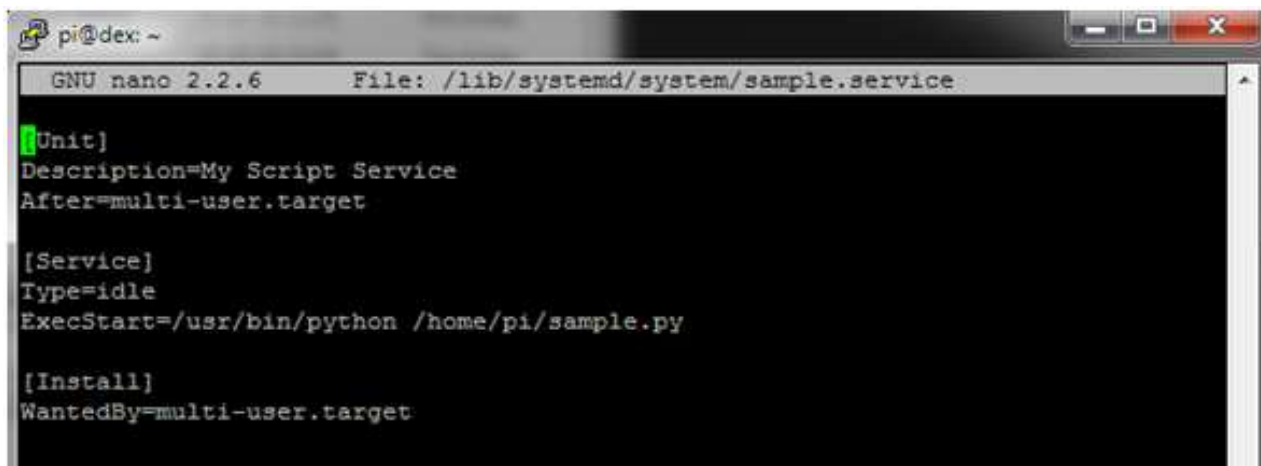
Add in the following text :

```
[Unit]
Description=My Sample Service
After=multi-user.target

[Service]
Type=idle
ExecStart=/usr/bin/python /home/pi/sample.py

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

You should save and exit the nano editor.

A screenshot of a terminal window on a Raspberry Pi. The window title is 'pi@dex: ~'. The terminal shows the GNU nano 2.2.6 text editor editing the file '/lib/systemd/system/sample.service'. The content of the file is as follows:

```
[Unit]
Description=My Script Service
After=multi-user.target

[Service]
Type=idle
ExecStart=/usr/bin/python /home/pi/sample.py

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

(<https://32414320wji53mwwch1u68ce-wpengine.netdna-ssl.com/wp-content/uploads/2014/11/test6.png>)

This defines a new service called “Sample Service” and we are requesting that it is launched once the multi-user environment is available. The “ExecStart” parameter is used to specify the command we want to run. The “Type” is set to “idle” to ensure that the ExecStart command is run only when everything else has loaded. Note that the paths are absolute and define the complete location of Python as well as the location of our Python script.

In order to store the script’s text output in a log file you can change the ExecStart line to:

```
ExecStart=/usr/bin/python /home/pi/sample.py > /home/pi/sample.log 2>&1
```

The permission on the unit file needs to be set to 644 :

```
sudo chmod 644 /lib/systemd/system/sample.service
```

Step 2 – Configure systemd

Now the unit file has been defined we can tell systemd to start it during the boot sequence :

```
sudo systemctl daemon-reload
sudo systemctl enable sample.service
```

Reboot the Pi and your custom service should run:

```
sudo reboot
```

```
pi@dex:~$ sudo chmod 644 /lib/systemd/system/sample.service
pi@dex:~$ sudo systemctl daemon-reload
pi@dex:~$ sudo systemctl enable sample.service
Created symlink from /etc/systemd/system/multi-user.target.wants/sample.service to /lib/systemd/system/sample.service.
pi@dex:~$ sudo reboot
```

(<https://32414320wji53mwwch1u68ce-wpengine.netdna-ssl.com/wp-content/uploads/2014/11/test7.png>)

Method 5: crontab

A detailed tutorial on using **crontab** to get a program running on boot can found here

(<https://www.dexterindustries.com/howto/auto-run-python-programs-on-the-raspberry-pi/>).

You can use any of these methods to run your program on boot as long as the point at which your Python script is run in the startup sequence is not vital. If your script relies on any system features being available at that point in time such as the network being connected and available, the /home/pi directory is mounted and ready for use or the System time has been updated by NTP then it would be ideal to use either systemd or init.d methods. These methods control the point in bootup at which your script is executed while booting.