Raspberry Pi Official Original Camera use tutorial

1.You need to connect camera to Raspberry Pi board.

2.You must to install a new operating system before use camera, which can can identify if the camera module is connected. The simplest method is download Raspbain system from Raspberry Pi website, and install it to new SD card.

Regardless of the version of the Raspbian system you are using, I recommend that you update the system with the following command:

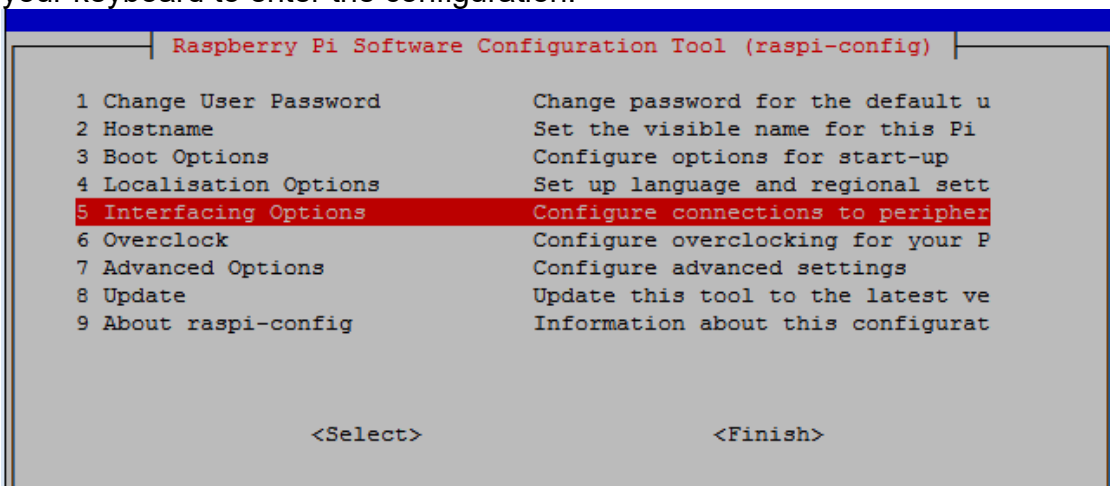
sudo apt-get update

sudo apt-get upgrade

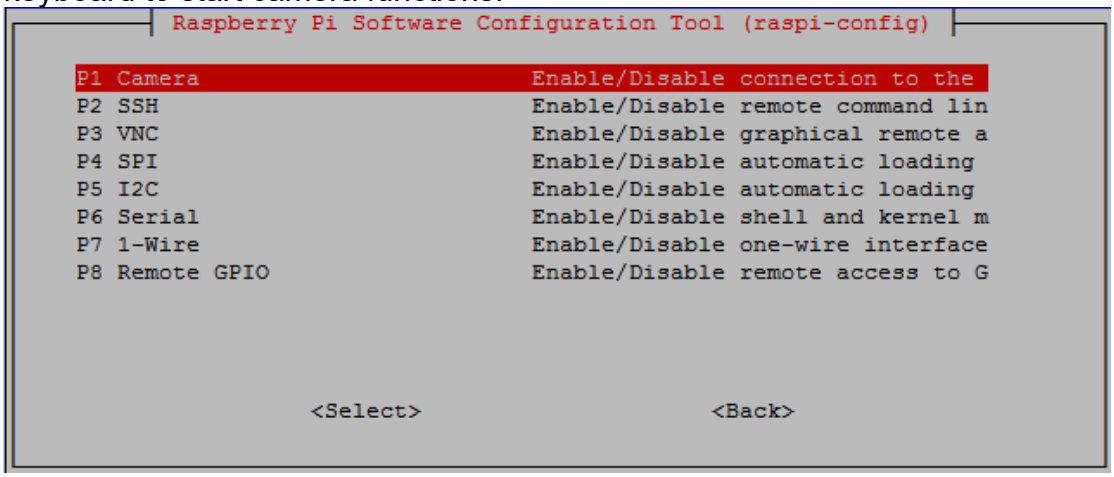
3.Enable camera options

3.1)You need to input: **sudo raspi-config** to enter the following interface:

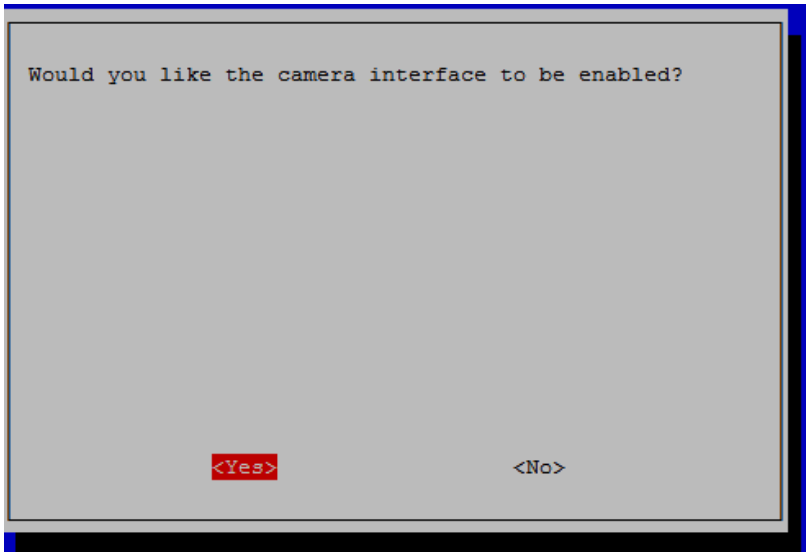
3.2)You need to choose “Interfacing Options”, press the “Enter” key of your keyboard to enter the configuration.



3.3)You need to choose “P1 Camera”, press the “Enter” key of your keyboard to start camera functions.



3.4) You need to choose “Yes”.



After the above steps, the camera function of Raspberry Pi has been turned on.

4. Use camera

Method 1:You can use Raspberry Pi official original camera by system command

raspistill -------This command is to get the static image

The picture below shows the parameter information about the command displayed directly by the command raspistill.

```
root@raspberrypi:/home/pi# raspistill

raspistill Camera App v1.3.11

Runs camera for specific time, and take JPG capture at end if requested

usage: raspistill [options]

Image parameter commands

-?, --help      : This help information
-w, --width     : Set image width <size>
-h, --height    : Set image height <size>
-q, --quality   : Set jpeg quality <0 to 100>
-r, --raw       : Add raw bayer data to jpeg metadata
-o, --output    : Output filename <filename> (to write to stdout, use '-o -'). If not specified, no f
ile is saved
-l, --latest    : Link latest complete image to filename <filename>
-v, --verbose   : Output verbose information during run
-t, --timeout   : Time (in ms) before takes picture and shuts down (if not specified, set to 5s)
-th, --thumb    : Set thumbnail parameters (x:y:quality) or none
-d, --demo      : Run a demo mode (cycle through range of camera options, no capture)
-e, --encoding  : Encoding to use for output file (jpg, bmp, gif, png)
-x, --exif      : EXIF tag to apply to captures (format as 'key=value') or none
-tl, --timelapse        : Timelapse mode. Takes a picture every <t>ms. %d == frame number (Try: -o im
g_%04d.jpg)
-fp, --fullpreview      : Run the preview using the still capture resolution (may reduce preview fps)
-k, --keypress  : Wait between captures for a ENTER, X then ENTER to exit
-s, --signal    : Wait between captures for a SIGUSR1 or SIGUSR2 from another process
-g, --gl        : Draw preview to texture instead of using video render component
-gc, --glcapture        : Capture the GL frame-buffer instead of the camera image
-set, --settings        : Retrieve camera settings and write to stdout
-cs, --camselect        : Select camera <number>. Default 0
-bm, --burst    : Enable 'burst capture mode'
-md, --mode     : Force sensor mode. 0=auto. See docs for other modes available
-dt, --datetime : Replace output pattern (%d) with DateTime (MonthDayHourMinSec)
-ts, --timestamp        : Replace output pattern (%d) with unix timestamp (seconds since 1970)
-fs, --framestart       : Starting frame number in output pattern(%d)
-rs, --restart  : JPEG Restart interval (default of 0 for none)

Preview parameter commands

-p, --preview   : Preview window settings <'x,y,w,h'>
-f, --fullscreen        : Fullscreen preview mode
-op, --opacity  : Preview window opacity (0-255)
-n, --nopreview : Do not display a preview window
```

raspivid -------This command is to get the video information

```
root@raspberrypi:/home/pi# raspivid

raspivid Camera App v1.3.12

Display camera output to display, and optionally saves an H264 capture at requested bitrate


usage: raspivid [options]

Image parameter commands

-?, --help      : This help information
-w, --width     : Set image width <size>. Default 1920
-h, --height    : Set image height <size>. Default 1080
-b, --bitrate   : Set bitrate. Use bits per second (e.g. 10MBits/s would be -b 10000000)
-o, --output    : Output filename <filename> (to write to stdout, use '-o -').
                  Connect to a remote IPv4 host (e.g. tcp://192.168.1.2:1234, udp://192.168.1.2:1234)
                  To listen on a TCP port (IPv4) and wait for an incoming connection use -l
                  (e.g. raspvid -l -o tcp://0.0.0.0:3333 -> bind to all network interfaces, raspvid -
l -o tcp://192.168.1.1:3333 -> bind to a certain local IPv4)
-v, --verbose   : Output verbose information during run
-t, --timeout   : Time (in ms) to capture for. If not specified, set to 5s. Zero to disable
-d, --demo      : Run a demo mode (cycle through range of camera options, no capture)
-fps, --framerate       : Specify the frames per second to record
-e, --penc      : Display preview image *after* encoding (shows compression artifacts)
-g, --intra     : Specify the intra refresh period (key frame rate/GoP size). Zero to produce an init
ial I-frame and then just P-frames.
-pf, --profile  : Specify H264 profile to use for encoding
-td, --timed    : Cycle between capture and pause. -cycle on,off where on is record time and off is p
ause time in ms
-s, --signal    : Cycle between capture and pause on Signal
-k, --keypress  : Cycle between capture and pause on ENTER
-i, --initial   : Initial state. Use 'record' or 'pause'. Default 'record'
-qp, --qp       : Quantisation parameter. Use approximately 10-40. Default 0 (off)
-ih, --inline   : Insert inline headers (SPS, PPS) to stream
-sg, --segment  : Segment output file in to multiple files at specified interval <ms>
-wr, --wrap     : In segment mode, wrap any numbered filename back to 1 when reach number
-sn, --start    : In segment mode, start with specified segment number
-sp, --split    : In wait mode, create new output file for each start event
-c, --circular  : Run encoded data through circular buffer until triggered then save
-x, --vectors   : Output filename <filename> for inline motion vectors
-cs, --camselect        : Select camera <number>. Default 0
```

Raspistill related commands:

1.After a delay of 1.1 seconds (in milliseconds), take a photo and name it image.jpg

raspistill -t 1000 -o image.jpg

2. Take a photo of custom size and frame rate

raspistill -t 1000 -o image.jpg -w 640 -h 480 -q 5

3. Set embossed style image effects

raspistill -t 1000 -o image.jpg -ifx emboss

4. Get a photo and send it to a standard output device (such as a monitor)

raspistill -t 1000 -o

Raspivid related commands:

1. Shoot a video: The default is video length is 5s, resolution is 1920*1080, frame rate: 17

raspivid -o myvideo.h264

2. Take a video: resolution is 640*480, time is 10s

raspivid -o myvideo.h264 -t 10000 -w 640 -h 480

Note: raspivid outputs an uncompressed H.264 video stream. In order to allow our normal video player to play, you need to install the gpac package.

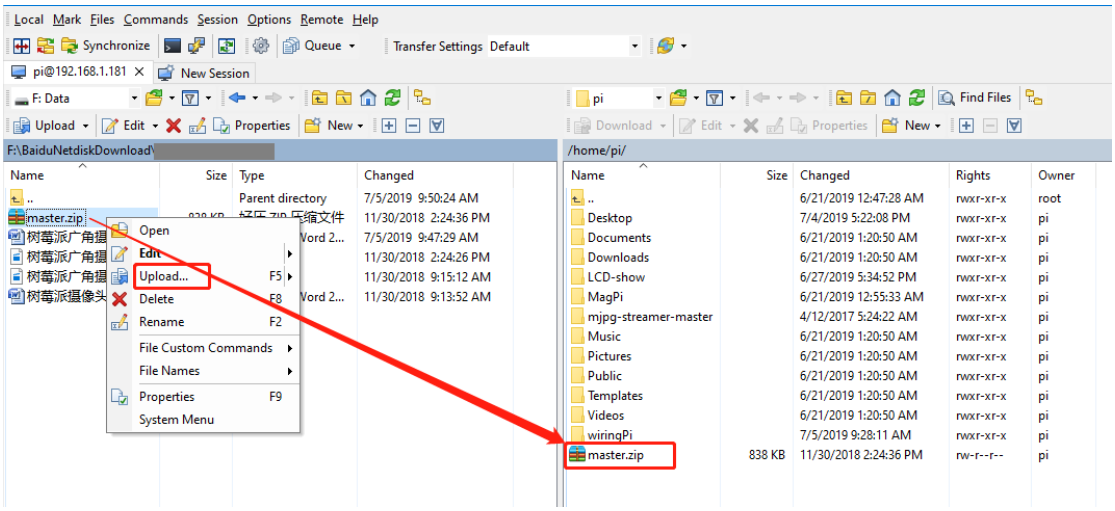
apt-get install -y gpac

Then use the MP4Box application in the gpac package to convert the H.264 format video stream to 10 frames per second MP4 format video.

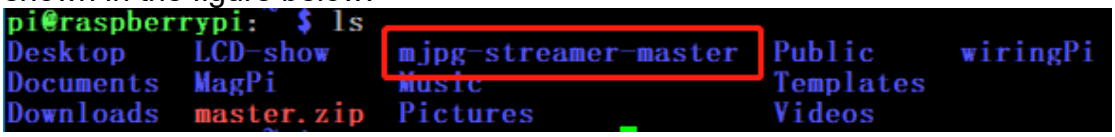
MP4Box -fps 10 -add myvideo.h264 myvideo.mp4

Method 2: You can transfer the video which captured by the Raspberry Pi to the web page.

2.1 You need to log in to the WinSCP software and transfer the master.zip file to the pi directory of the Raspberry Pi.



2.2 You should input: #unzip master.zip
(This command to unzip mater.zip)
After complete unzip, input **ls** you can see **migp-streamer-master** folder. As shown in the figure below.



2.3 Compiling this project requires cmake: perform the following command to install download.
(apt-get is a networked installation, so you need make Raspberry Pi successfully connect to the network.)
You should input:
#sudo apt-get install cmake
A prompt will appear when installing, you need to enter Y.
After completion, As shown in the figure below.

```
Suggested packages:
  codeblocks eclipse ninja-build
The following NEW packages will be installed:
  cmake cmake-data
0 upgraded, 2 newly installed, 0 to remove and 6 not upgraded.
Need to get 2,845 kB of archives.
After this operation, 14.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://mirrordirector.raspbian.org/raspbian/ jessie/main cmake-data all 3.
0.2-1+deb8u1 [929 kB]
Get:2 http://mirrordirector.raspbian.org/raspbian/ jessie/main cmake armhf 3.0.2
-1+deb8u1 [1,915 kB]
Fetched 2,845 kB in 3s (861 kB/s)
Selecting previously unselected package cmake-data.
(Reading database ... 124054 files and directories currently installed.)
Preparing to unpack .../cmake-data_3.0.2-1+deb8u1_all.deb ...
Unpacking cmake-data (3.0.2-1+deb8u1) ...
Selecting previously unselected package cmake.
Preparing to unpack .../cmake_3.0.2-1+deb8u1_armhf.deb ...
Unpacking cmake (3.0.2-1+deb8u1) ...
Processing triggers for man-db (2.7.0.2-5) ...
Setting up cmake-data (3.0.2-1+deb8u1) ...
Setting up cmake (3.0.2-1+deb8u1) ...
root@raspberrypi:/home#
```

2.4 You need to install library

You should input:

#sudo apt-get install libjpeg8-dev

After completion, As shown in the figure below.

```
root@raspberrypi:/home# sudo apt-get install libjpeg8-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  libjpeg8-dev
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 205 kB of archives.
After this operation, 436 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ jessie/main libjpeg8-dev a
f 8d1-2 [205 kB]
Fetched 205 kB in 1s (150 kB/s)
Selecting previously unselected package libjpeg8-dev:armhf.
(Reading database ... 125733 files and directories currently installed.)
Preparing to unpack .../libjpeg8-dev_8d1-2_armhf.deb ...
Unpacking libjpeg8-dev:armhf (8d1-2) ...
Setting up libjpeg8-dev:armhf (8d1-2) ...
root@raspberrypi:/home#
```

2.5 Compiling

You need to enter: /home/pi/mjpg-streamer-master/mjpg-streamer-experimental/

cd /home/pi/mjpg-streamer-master/mjpg-streamer-experimental/

You need to enter:

ls

```
pi@raspberrypi:~ $ cd /home/pi/mjpg-streamer-master/mjpg-streamer-experimental/
pi@raspberrypi:~/mjpg-streamer-master/mjpg-streamer-experimental $ ls
cmake           docker-start.sh  mjpg_streamer.c  README.md  TODO     www
CMakeLists.txt  LICENSE          mjpg_streamer.h  scripts    utils.c
Dockerfile      Makefile         plugins          start.sh   utils.h
pi@raspberrypi:~/mjpg-streamer-master/mjpg-streamer-experimental $
```

And you need to run the following command directly

#make clean all

You can wait for the compilation to complete, you can see the interface shown below.

```
Linking C shared library output_http.so
make[3]: Leaving directory '/home/mjpg-streamer-master/mjpg-streamer-experime
ntal/_build'
[ 87%] Built target output_http
make[3]: Entering directory '/home/mjpg-streamer-master/mjpg-streamer-experim
ental/_build'
Scanning dependencies of target output_rtsp
make[3]: Leaving directory '/home/mjpg-streamer-master/mjpg-streamer-experime
ntal/_build'
make[3]: Entering directory '/home/mjpg-streamer-master/mjpg-streamer-experim
ental/_build'
[ 93%] Building C object plugins/output_rtsp/CMakeFiles/output_rtsp.dir/outpu
t_rtsp.c.o
Linking C shared library output_rtsp.so
make[3]: Leaving directory '/home/mjpg-streamer-master/mjpg-streamer-experime
ntal/_build'
[ 93%] Built target output_rtsp
make[3]: Entering directory '/home/mjpg-streamer-master/mjpg-streamer-experim
ental/_build'
Scanning dependencies of target output_udp
make[3]: Leaving directory '/home/mjpg-streamer-master/mjpg-streamer-experime
ntal/_build'
make[3]: Entering directory '/home/mjpg-streamer-master/mjpg-streamer-experim
ental/_build'
[100%] Building C object plugins/output_udp/CMakeFiles/output_udp.dir/output_
udp.c.o
Linking C shared library output_udp.so
make[3]: Leaving directory '/home/mjpg-streamer-master/mjpg-streamer-experime
ntal/_build'
[100%] Built target output_udp
make[2]: Leaving directory '/home/mjpg-streamer-master/mjpg-streamer-experime
ntal/_build'
make[1]: Leaving directory '/home/mjpg-streamer-master/mjpg-streamer-experime
ntal/_build'
root@raspberrypi:/home/mjpg-streamer-master/mjpg-streamer-experimental#
```

7. Restart system

You need to input: **sudo reboot** to reboot system

Plug in the camera and restart the system.

8. After rebooting, enter the system.

You need to enter the **mjpg-streamer-experimental** directory by command:

cd /home/pi/mjpg-streamer-master/mjpg-streamer-experimental/

And use the following command to start the Raspberry Pi Official Original Camera:

#./mjpg_streamer -i "./input_raspicam.so" -o "./output_http.so -w ./www"

(Note:Raspberry Pi special camera includes: Raspberry Pi night vision camera, Raspberry Pi official original camera, Raspberry pie wide-angle camera)

If you want to modify the resolution of the captured video,you need to input the following command:

#./mjpg_streamer -i "./input_raspicam.so -w 320 -h 240 -fps 10" -o "./output_http.so -w ./www"

Some cameras will report an error when executing this command. If they do not return to the command prompt and display "Starting Camera", it means success.
As shown in the figure below, the camera is successfully turned on:

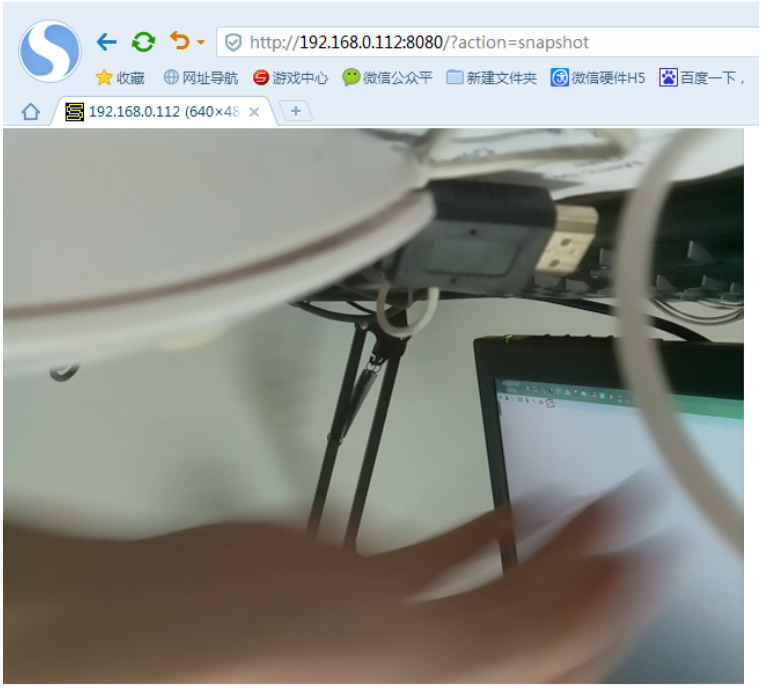
```
pi@raspberrypi:~/mjpg-streamer-master/mjpg-streamer-experimental $ ./mjpg_stream
er -i "./input_raspicam.so" -o "./output_http.so -w ./www"
MJPG Streamer Version.: 2.0
 i: fps.............: 5
 i: resolution........: 640 x 480
 i: camera parameters..............:

Sharpness 0, Contrast 0, Brightness 50
Saturation 0, ISO 0, Video Stabilisation No, Exposure compensation 0
Exposure Mode 'auto', AWB Mode 'auto', Image Effect 'none'
Metering Mode 'average', Colour Effect Enabled No with U = 128, V = 128
Rotation 0, hflip No, vflip No
ROI x 0.000000, y 0.000000, w 1.000000 h 1.000000
 o: www-folder-path......: ./www/
 o: HTTP TCP port........: 8080
 o: HTTP Listen Address..: (null)
 o: username:password....: disabled
 o: commands.............: enabled
 i: Starting Camera
Encoder Buffer Size 81920
```

9. Test results
View the image, open the browser on the PC side, you need to enter the following URL to see the static screenshot:

http://<RaspberryPi IP>:8080/?action=snapshot

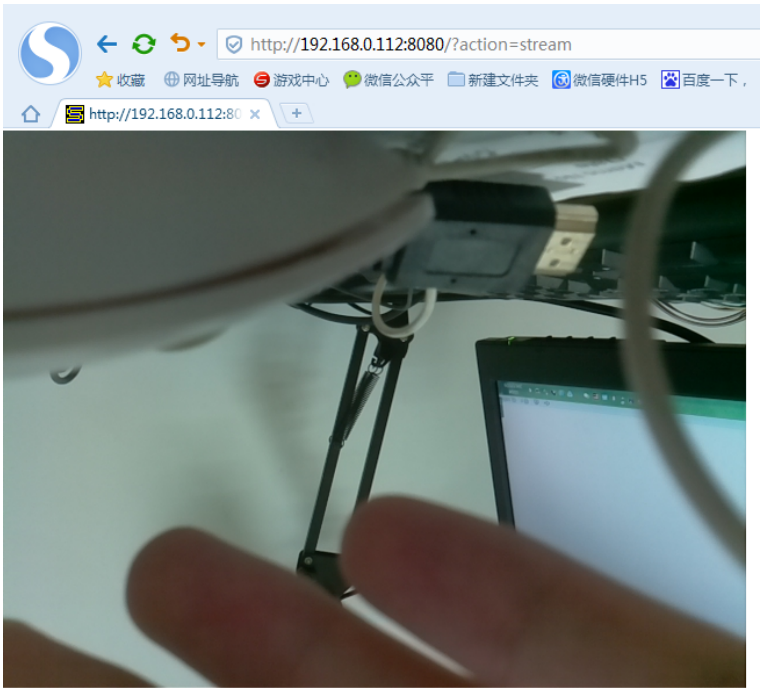
My URL is：http://192.168.0.112:8080/?action=snapshot



You should input the following URL to see the dynamic image:

http://<RaspberryPi IP>:8080/?action=stream

My URL is：http://192.168.0.112:8080/?action=stream



You can also use the following URL:

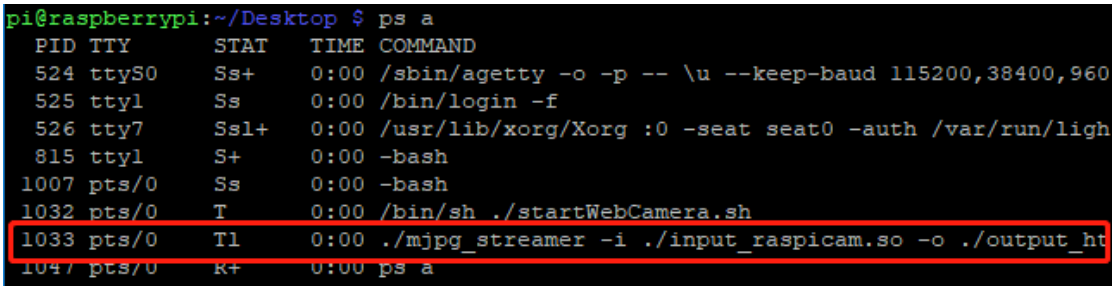
http://<RaspberryPi IP>:8080/javascript_simple.html

PS: When using the RaspiCamera dedicated camera, the dynamic image screen will have a transmission delay, about 1.4 frames / sec. It is recommended to use static capture.

Note: After running the camera web service, it will occupy the camera, causing other camera commands to fail. Please kill the process before running other camera commands.

Enter command to view camera progress number:

ps a



Kill the program's PID process number:

sudo kill -9 1033