(advanced) Projects

home

Installation for ARM (Raspberry Pi)

How to install or upgrade UV4L on Raspbian Wheezy & Raspbian Jessie for Raspberry Pi

The following instructions explain how to install UV4L on the official *Raspbian Linux* distributions available for any model of the Raspberry Pi boards: *Zero*, *Zero W (Wireless)*, *1*, *2*, *3*, *Compute Module 1, Compute Module 3.*

Other distributions than Raspbian and other ARM-based boards are known to work, but they are not officially supported.

As these instructions are updated and improved very frequently without notice, it is suggested to read them from scratch in case of problems and especially whenever a new UV4L module is announced. Important notes about specific drivers, modules, configurations, etc.. can be found at the bottom of this page.

All the software is provided "as is" and with absolutely no warranty. Exact license terms are included in each package.

The core of the UV4L software falls in the *middleware* category. It consists of a series of highly configurable *drivers*, an optional *Streaming Server* component providing a *RESTful API* for custom development and various *extensions* for the server that cooperate together. The Streaming Server also provides the necessary *web UI* for the end-users to try or use all the key functionalities directly. For maximum efficiency, each instance of UV4L runs as a single system process which exploits the underlying hardware natively whenever possible. Here is a more detailed list of features.

Below we will see how to install all the components to get the best from UV4L, with particular focus on the driver for the Raspberry Pi camera boards, although all other drivers can be installed similarly.

This same driver has been extended to support the **TC358743 HDMI to MIPI chipset converter** on all Raspberry Pi boards (this chipset is present on the B101 HDMI to CSI-2 Bridge, for example). However, instructions on how to enable the TC358743 on boards different from *Raspberry Pi 3* (e.g. *Zero*, *ZeroW*, *CM3*, etc…) will be provided upon request only.

To install UV4L open a terminal and type the following commands:

```
$ curl http://www.linux-projects.org/listing/uv4l_repo/lrkey.asc | sudo apt-
key add -
```

On *Raspbian Wheezy* add the following line to the file */etc/apt/sources.list*:

```
deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/ wheezy main
```

while on *Raspbian Jessie* add this line:

```
deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/ jessie main
```

```
$ sudo apt-get update
$ sudo apt-get install uv4l uv4l-raspicam
```

The above two commands will upgrade UV4L to the most recent version, if it's already installed.

If you want the driver to be loaded at boot, also install this optional package:

```
$ sudo apt-get install uv4l-raspicam-extras
```

As a convenience, the above package will install a service script for starting, stopping or restarting the driver at any time, for example:

```
$ sudo service uv4l_raspicam restart
```

When (re)starting the service, *uv4l* will be instructed to parse the configuration file */etc/uv4l/uv4l-raspicam.conf* to get the default values for the driver and the server options. You can edit that file to add, remove or change the default values. This same service is started at boot.

If you are using the TC358743, the *uv4l-tc358743-extras* package has to be installed for it to work:

```
$ sudo apt-get install uv4l-tc358743-extras
```

The above package will automatically install the *uv4l-raspicam-extras* package and some other helper programs. Before using the TC358743 make sure that both the *Camera* interface and the *I2C* bus are enabled in the *raspi-config* system tool and that the line *tc358743=yes* is present or uncommented in the configuration file */etc/uv4l/uv4l-raspicam.conf*.

Now the UV4L core component and the Video4Linux2 driver for the CSI Camera Board are installed. If you occasionally get unexpected errors from the driver, make sure the camera is enabled and enough memory is reserved for the *GPU* (not less than 128MB is suggested) from this menu:

```
$ sudo raspi-config
```

Also consider updating the firmware with the following command:

```
$ sudo rpi-update
```

For detailed informations, options, etc… about the modules installed type accordingly:

```
$ man uv4l
$ man uv4l-raspicam
```

To get the list of all available options:

```
$ uv4l --help --driver raspicam --driver-help
```

If you have not installed the optional *uv4l-raspicam-extras* package (which provides a convenient script for starting uv4l with the settings taken from a configuration file) and want to quickly test uv4l, load it manually:

```
$ uv4l --driver raspicam --auto-video_nr --width 640 --height 480 --encoding
jpeg
```

and take a JPEG snapshot from the camera:

```
$ dd if=/dev/video0 of=snapshot.jpeg bs=11M count=1
```

For a list of other use cases click here.

To manually terminate a running driver, close all the applications accessing the device and kill the corresponding *uv4l* process:

```
$ pkill uv4l
```

Apart from the driver for the Raspberry Pi Camera Board, the following Streaming Server front-end and drivers can be optionally installed:

```
$ sudo apt-get install uv4l-server uv4l-uvc uv4l-xscreen uv4l-mjpegstream
uv4l-dummy uv4l-raspidisp
```

for which the manual pages are available:

```
$ man uv4l-server
$ man uv4l-uvc
$ man uv4l-xscreen
$ man uv4l-mjpegstream
$ man uv4l-dummy
$ man uv4l-raspidisp
```

The *WebRTC* extension for the Streaming Server is also available with two alternative packages depending on the Raspberry Pi model in use. If you have a *Raspberry Pi 1*, *Compute Module 1*, *Zero* or *Zero W* (Wireless), type:

```
$ sudo apt-get install uv4l-webrtc-armv6
```

Otherwise, if you have any other model (e.g. Raspberry Pi 2 or 3), type:

```
$ sudo apt-get install uv4l-webrtc
```

As the Streaming Server is able to serve and run any custom web applications, an optional package containing the source code of some of the demos mentioned in the examples (e.g. this one) is available. The files will be installed in */usr/share/uv4l/demos/*:

```
$ sudo apt-get install uv4l-demos
```

Note that some browsers may no longer exploit many of the WebRTC functionalities over HTTP for security reasons. You will need to configure secure *HTTPS* in the Streaming Server instead. To do this, you must provide a **password-less private key** and a valid **certificate** via the *–ssl-private-key-file* and the *–ssl-certificate-file* server options. A private key and a self-signed certificate can be generated as follows:

```
$ openssl genrsa -out selfsign.key 2048 && openssl req -new -x509 -key
selfsign.key -out selfsign.crt -sha256
```

Once you have installed and eventually configured the HTTP(S) Streaming Server module as shown above, make sure to reload *uv4l* for it to notice and start the server. Afterwards you can access the server with the browser at the default address and port *https://raspberry:8080/* (where *raspberry* has to be replaced with the actual hostname or IP address of your RaspberryPi and the protocol can be either *http* or *https*).

Many of the Streaming Server and WebRTC settings can be changed *on-the-fly* thanks to the RESTful API without restarting UV4L.

Thanks to the WebRTC extension mentioned above, it's also possible to broadcast both live audio and video contents from the Raspberry Pi 2 to all the participants or viewers joining a room of a *Jitsi Meet* conference on the Web. Furthermore, no browser and no GUI will have to be used on the Raspberry Pi. For this to be possible, it's necessary to install the additional *xmpp-bridge* service, which will be automatically started once the installation has finished or when the system boots:

```
$ sudo apt-get install uv4l-xmpp-bridge
```

TC358743

As mentioned before, the *TC358743* requires the *uv4l-tc358743-extras* package to be installed. Make sure you have both the *Camera* and *I2C* enabled via *raspi-config*, which essentially comments out the option *dtparam=i2c_arm=on* in */boot/config.txt* and adds the *i2c-bcm2708* and *i2c-dev* lines in */etc/modules*.

As the support for this chip is essentially an extension of the original *uv4l-raspicam* driver which has been written for image sensors primarily, the *TC358743* has to be explicitly enabled with the *–tc358743* driver option specified (via command line or in the configuration file) before it can be used. Some of the options listed in the *uv4l-raspicam* manual page affect the TC358743 as well; you are encouraged to read the driver manual for more informations.

A quick test to see if capturing works if you have no device with HDMI-out ready can be made by simply connecting the *HDMI-out* port of the Raspberry Pi to the

HDMI-in port of the B101 board on the same Raspberry Pi. Make sure *hdmi_force_hotplug=1* is set in */boot/config.txt* for the HDMI signal to be detected.

Audio configuration

By default WebRTC gets its input audio from the default recording device, which usually is the first entry that appears in the ordered list given by the *arecord –list-pcms* command. If you want to make use of another device, it's enough to set the –*webrtc-recdevice-index* configuration option to the corresponding position in the list (starting from 0, top-down).

Alternatively, if you want, for example, to use an USB sound card as your default recording device, then do the following: in *Raspbian Jessie* edit or create */etc/asound.conf* and insert:

```
pcm.!default {
   type asym
   playback.pcm "plug:hw:0"
   capture.pcm "plug:dsnoop:1"
}
```

where "0" is usually the index of the default sound card embedded in the Raspberry Pi that will be used for playback, while "1" usually corresponds to the index of the USB sound card that will be used for capture. After you have rebooted, make sure capture volumes are unmuted and increased to the desired level. *alsamixer* is an handful tool for this purpose. If the quality or the volume of the captured audio is not as expected for some reasons, try with a sample rate of 44100 Hz. If the hardware does not support this sample rate natively, try with the ALSA sample rate converter plug-in.

in *Raspbian Wheezy*, edit */etc/modprobe.d/alsa-base.conf* and modify or add the following lines and reboot the system:

```
options snd-usb-audio index=0
options snd_bcm2835 index=1
```

Note for the UVC driver users

Since recent versions of UV4L, there should be no longer need to manually load the driver to use your *webcam*. After you have rebooted for the first time, it's enough to plug in (or unplug) the webcam to have it recognized by the system service and have *uv4l* loaded (or unloaded) automatically. If you have installed the UV4L Streaming Server, each time you attach or detach the webcam, the corresponding server instance is also automatically created or destroyed respectively. The port the server will be listening to is specified in the configuration file (i.e. */etc/uv4l/uv4l-uvc.conf*) and should be *8090* by default. You do not typically need to modify the configuration file.
Although nothing forbids to have multiple instances of *uv4l* running at the same time, the above automatism will only work for one UVC-based device plugged in, at most, at the same time – which is the common case. Unfortunately, from the second webcam onwards the system service is not so smart: as it does not internally make any distinction between uvc-based webcams and reads the same configuration file each time, the new Streaming Server instances will try to use the same port again, which might be in conflict with the first running instance. In this case, be prepared to tweak the service script (namely */etc/init.d/uv4l_uvc*) and eventually use multiple configuration files.

Raspidisp driver

The *raspidisp* driver turns *HDMI-out* into a virtual *Video4Linux-compliant input device* (like a camera). You do not necessarily need a display connected to the Raspberry Pi HDMI-out port. In other words, this driver can capture whatever you see or would see on the screen.

This is useful if you have an headless Raspberry Pi. With the help of the *UV4L Streaming Server* you can have full control of the Raspberry Pi from within any browser (running on a PC in the same network, for example). In facts, it's possible to stream the frame buffer of Raspberry Pi to a standard web page in the browser (plugin-free) with very low latency and, at the same time, forward keyboard or mouse input events from the PC to the Raspberry Pi. If you want to try this, apart from *uv4l-raspidisp*, the *uv4l-raspidisp-extras* package has to be installed:

```
$ sudo apt-get install uv4l-raspidisp-extras
```

The above package includes a system service that automatically starts an *uv4l* instance at boot and, if installed, also an instance of the *Streaming Server* listening to port *9080* by default. The service instructs *uv4l* to parse the configuration file */etc/uv4l/uv4l-raspidisp.conf* for the initial setting values.