How to setup a VC MIPI OV9281 on a Raspberry PI Model 3B+

Hardware and Software Setup

Revision:	0.2
Date:	2019-04-05
Copyright:	2019 Vision Components GmbH Ettlingen, Germany
Author:	VC Support

Foreword and Disclaimer

This documentation has been prepared with most possible care. However Vision Components GmbH does not take any liability for possible errors. In the interest of progress, Vision Components GmbH reserves the right to perform technical changes without further notice.

Please notify support@vision-components.com if you become aware of any errors in this manual or if a certain topic requires more detailed documentation.

This manual is intended for information of Vision Component's customers only. Any publication of this document or parts thereof requires written permission by Vision Components GmbH.

Image symbols used in this document

Symbol	Meaning
	The Light bulb highlights hints and ideas that may be helpful for a development.
	This warning sign alerts of possible pitfalls to avoid. Please pay careful attention to sections marked with this sign.
	This is a sign for an example.

Trademarks

Linux, Debian, the Tux logo, Vivado, Xilinx and Zynq, ARM, Cortex, Windows XP, Total Commander, Tera Term, Motorola, HALCON, Vision Components are registered Trademarks. All trademarks are the property of their respective owners.

Raspberry Pi and Raspbian are also registered Trademarks.

ESD sensitivity

Warning

The components are very sensitive to electrostatic discharge (ESD)! Please take all the precautions necessary to avoid ESD!

ESD

The electronic components and circuits are sensitive to ElectroStatic Discharge (ESD). When handling any circuit board assemblies, it is necessary that ESD safety precautions be observed.

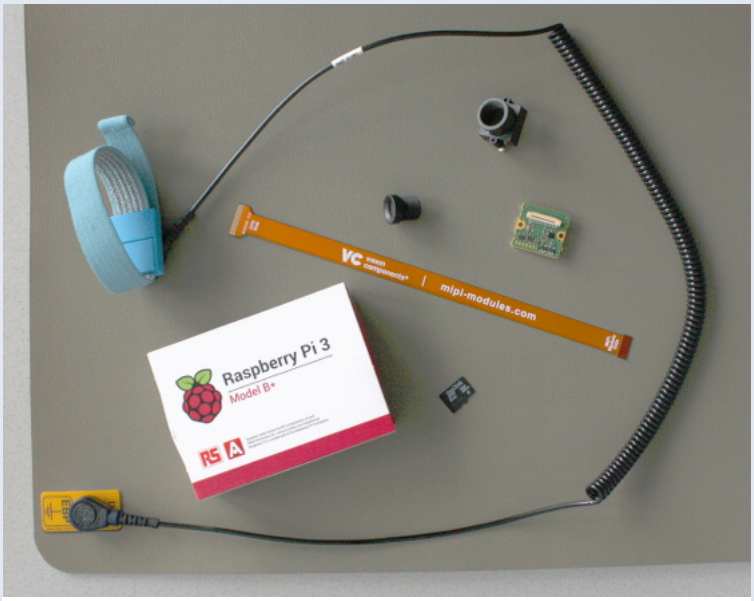
ESD safe best practices include, but are not limited to:

- Leaving circuit boards in their antistatic packaging until they are ready to be installed.
- Using a grounded wrist strap when handling circuit boards.
- Working on a grounded ESD table mat.
- Only handling circuit boards in ESD safe areas, which may include ESD floor and table mats, wrist strap stations and ESD safe lab coats.
- Avoiding handling circuit boards in carpeted areas.
- Try to handle the board by the edges, avoiding contact with components.

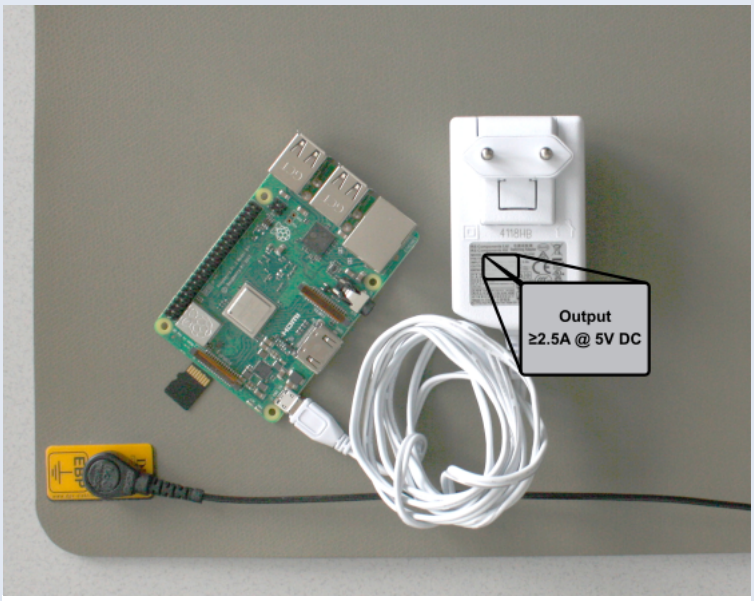
This note is not an exhaustive information about the protection against electrostatic discharge (ESD).

Table of Contents

1 Overview



Overview of relevant components excluding power supply, monitor, USB keyboard, cables



The power supply must at least provide 2.5A at 5V.

2 Hardware Setup

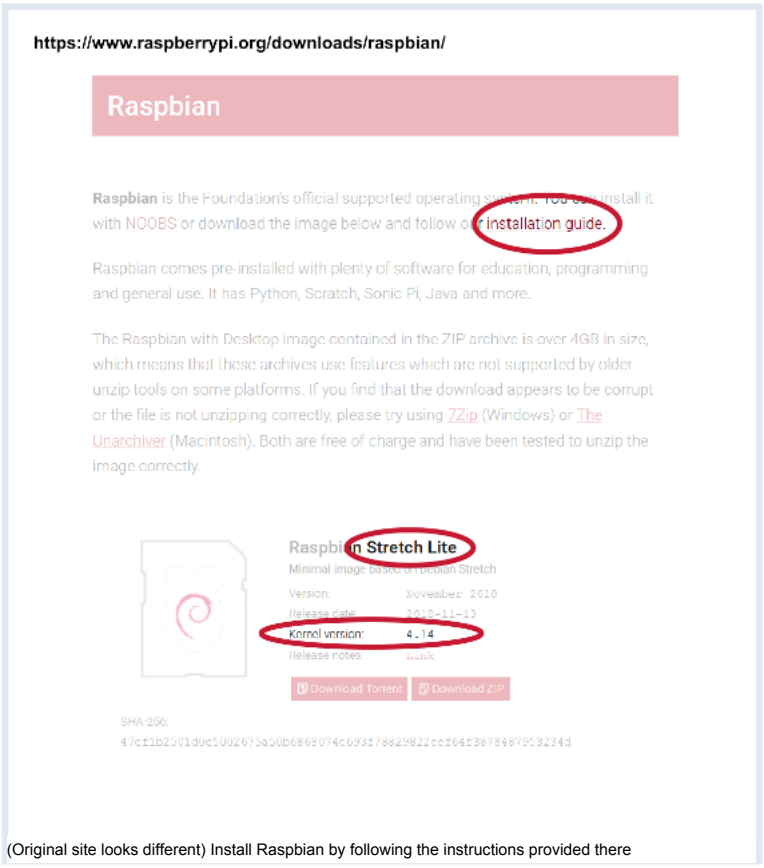
2.1 Hardware Pre-Check: Install Raspbian

First step is to install Raspbian from

https://www.raspberrypi.org/downloads/raspbian/

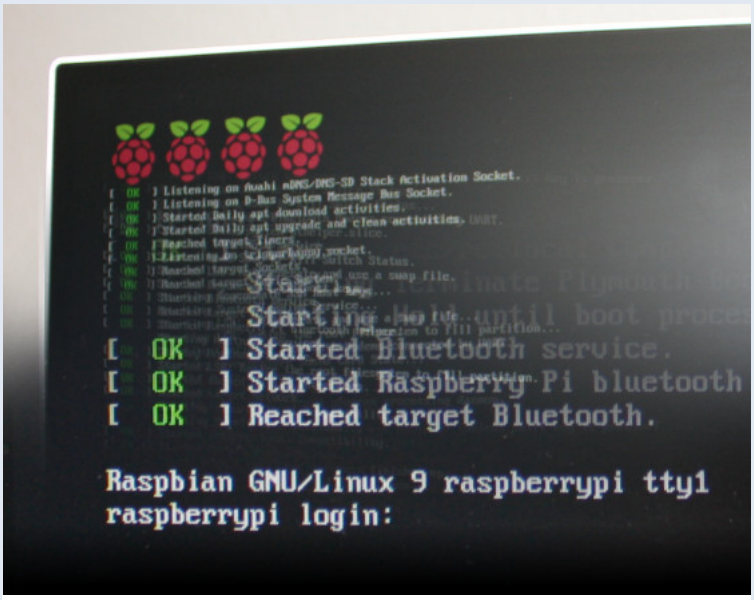
The driver is known to work with the *kernel version 4.14*, so download the appropriate Raspbian version. *Raspbian Stretch Lite* is sufficient, and this guide expects this version to be installed not only for the framebuffer output handling.

For more installation instructions see the Raspbian Installation Manual; the procedure depends on the platform type where the OS is going to be installed.



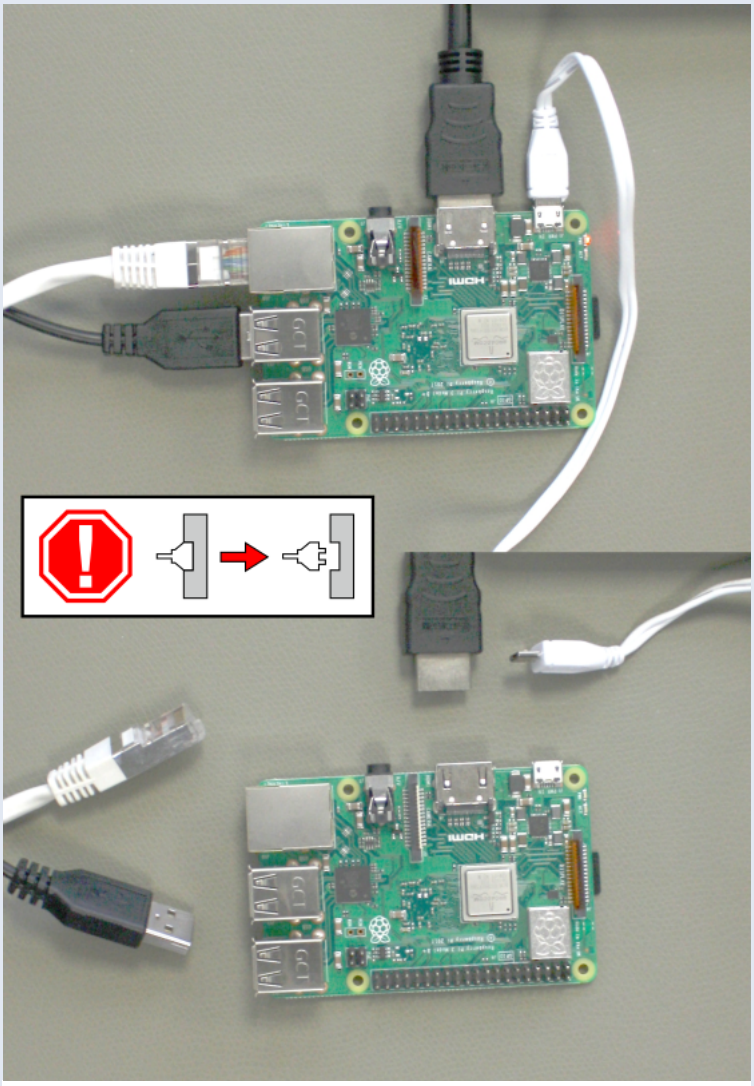
(Original site looks different) Install Raspbian by following the instructions provided there

The display shows a login prompt after successful installation. If this is not the case, you have to check your Raspbian installation. The most relevant information to succeed can be found at the Raspbian website or at the web.



Raspbian showing login screen (user is usually *pi* with password *raspberry*)

2.2 Connect the MIPI module

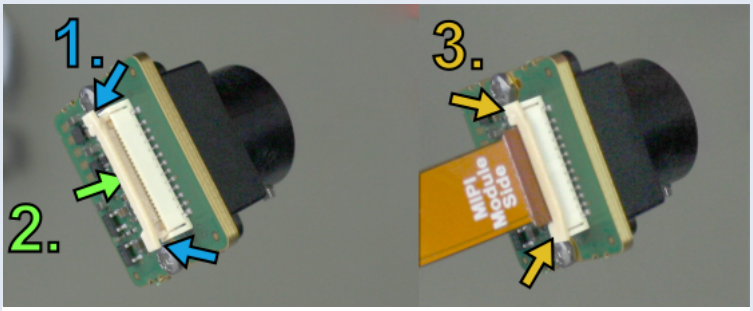


Always disconnect other cables before connecting or disconnecting the MIPI module!

Warning

Always disconnect all cables before connecting or disconnecting the MIPI module!

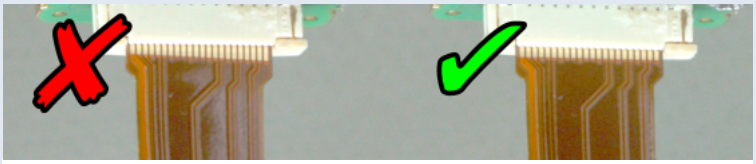
The ends of the MIPI module connector cable is marked with the hardware to connect to. Open the socket connectors first by raising their lid, insert the cable and press their lid back when mounted correctly. You should then not be able to pull the cable out.



Open the MIPI module socket, put in the cable, close the MIPI module socket

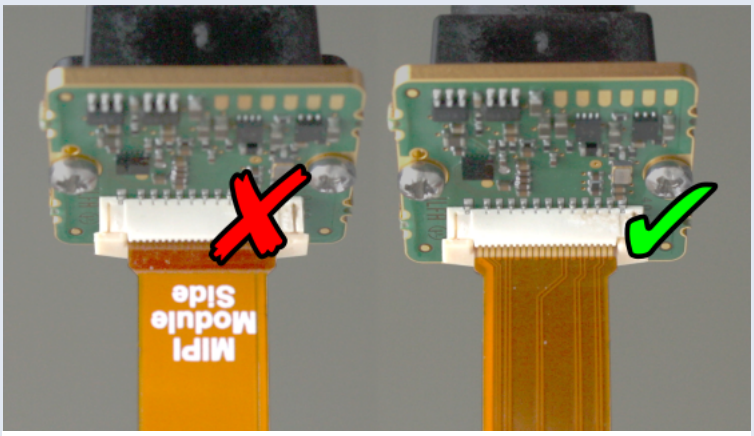
Warning

The connection at this type of socket is not protected against bad alignment, so always check the orthogonality, and if it is bent, correct it! Also watch out for the right orientation of the cable! The MIPI module or the Raspberry Pi can be irrevocably damaged if the cable is not inserted the right way, and warranty is lost!



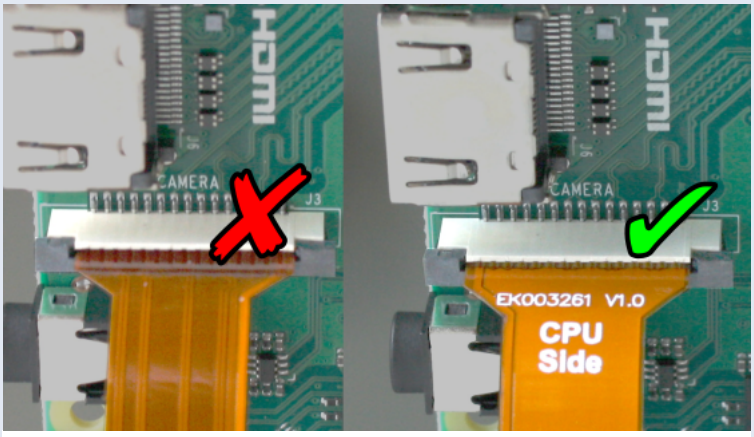
Harmfully angled (left) and good (right) cable fixation

The socket type is also not protected against wrong orientation, so compare your setup to the figures below before switching the power on.



Watch the orientation of the cable (left: bad, right: good)

There may be a dust prevention sticker at the socket named *CAMERA* at the raspberryPi, remove it first. Like at the sensor module, open the lid first, insert the cable to be orthogonally fixed after shutting the lid. Also check the orthogonality here and correct it if the cable is angled!



Connect the cable to the CAMERA socket at the raspberry Pi equally (left: bad, right: good)

Warning

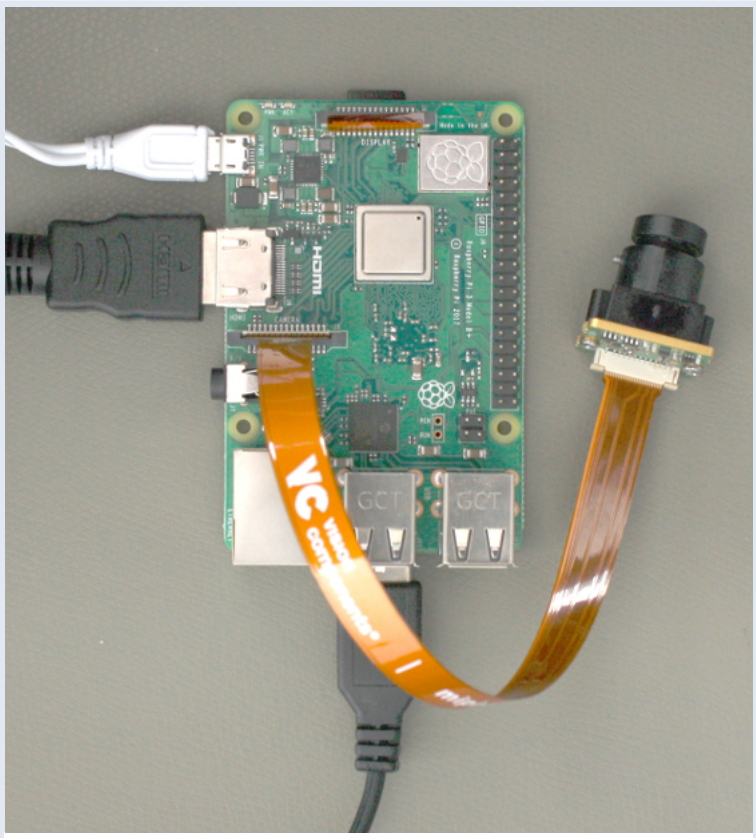
Do not connect other devices to the I²C bus named *VC*, since it can affect the communication between the camera sensor and the driver!

For example, running the touch screen of the Raspberry PI 7 inch display will lead to communication problems between driver and camera sensor. The display may work with the following line appended to the */boot/config.txt*, but test first without connecting it to the Raspberry PI to be sure everything works so far:

```
disable_touchscreen=1
```

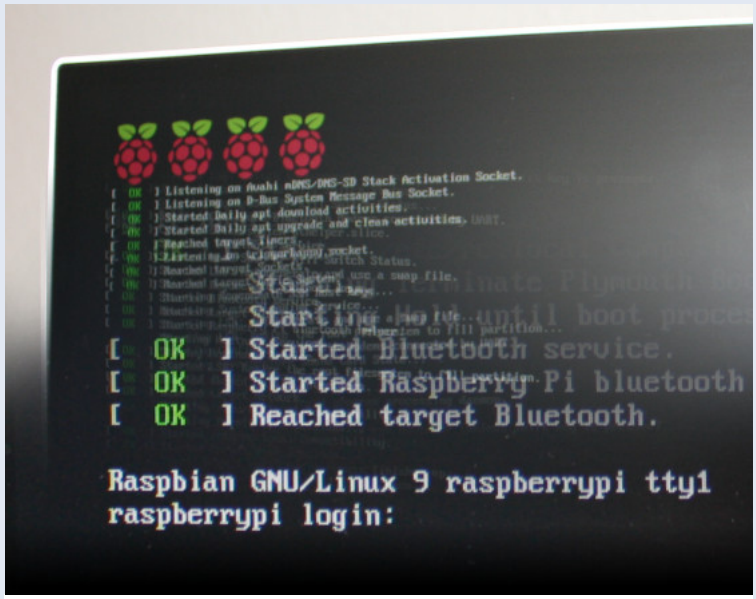
Don't connect the *SDA*/*SCK* of the 7 inch display since this would connect the I²C bus *VC* from the socket named *DISPLAY* with the I²C bus *ARM* at the pinout!

Reconnect the other peripherials to the Raspberry Pi.



Connection setup for the first image acquisition test.

You should have the login prompt back after switching the system on.



Raspbian showing login screen (user is usually *pi* with password *raspberry*)

3 Software Setup

3.1 Install necessary Raspbian packages

Before beginning with the installation, do the following steps first. This requires your Raspberry PI to already have an internet connection; otherwise you have to install the packages mentioned manually, search the web for the procedure needed.

1. Update the *raspberrypi-kernel* package and your system by calling:

```
sudo apt-get update && sudo apt-get upgrade
```

2. **Reboot**.
3. Install the *raspberrypi-kernel-headers* and *device-tree-compiler* package by using the following command:

```
sudo  apt-get install  raspberrypi-kernel-headers  device-tree-compiler
```

4. Test if the version of the running kernel matches the version of the kernel headers, the following command should show the directory for compiling the sensor module kernel module driver:

```
ls "/usr/src/linux-headers-$(uname -r)"
```

5. If the test was successful you may succeed.

3.2 Get the driver and demo code

You can download the driver and demo code with sources from the website http://www.vcmipi-modules.com
The archive file to get is named as follows:

vc_mipi_package_raspberryPi3_ov9281.tar.gz

Transfer the archive file to your Raspberry PI, you can do this for example by copying it to an USB stick and mount it at the Raspbian system.
Extract the archive at your home directory, there should then be a subdirectory named

vc_mipi_package_raspberryPi3_ov9281/

In this section, it is assumed, that you are in that subdirectory.

3.3 Driver Installation

1. Change to the subdirectory of the extracted archive named

   vc_mipi_driver_raspberryPi3_ov9281/

2. The directory with the sources contains a Makefile to compile the driver. Do so by calling:

```
make clean all
```

3. Calling make the following way will install the kernel module in your current kernel module path:

```
make install
```

4. Follow the instructions displayed by the make command, this *may* be similar to the following lines:

```
ADD 'dtparam=i2c_vc=on' and 'dtoverlay=vc_mipi_ov9281' to your /boot/config.txt
ADD 'disable_touchscreen=1' to your /boot/config.txt if a touchscreen is attached
ADD 'cma=128M' to your /boot/cmdline.txt
```

5. **Reboot**.
6. After reboot check the installation success by reentering the subdirectory named 'vc_mipi_driver_raspberryPi3_ov9281/' and executing the following commands:

```
chmod +x ./check_installation.sh
./check_installation.sh
```

   If one of the outputs show a failure, check first if you followed all instructions correctly. Should the failure not vanish refer to the Q/A section and/or read the Driver knowledge section for more information to be able to identify the source of your problem.

3.4 First Image Acquisition Test

A sensor device should be listed as *Video input* at the following command output (from the Video4Linux-Control):

```
v4l2-ctl --all
```

The following command dumps sensor data:

```
v4l2-ctl --stream-mmap --stream-count=-1 -d /dev/video0 --stream-to=/dev/null
```

It will output subsequent lines ending with a frames-per-second information (in the example named *[number]*) until pressing CTRL-C:

```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< [number] fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< [number] fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< [number] fps
<<<<<<<<<<<<<<<<<<<<<<<<^C
```

3.5 Running the Demo

The demo itself is a program named *vcmipidemo* and its source code is mainly at the file *vcmipidemo.c*. However more programs are provided, namely the *vcimgnetsrv*, a network image server, and its counterpart *vcimgnetclient.py*. The *vcimgnetsrv* is started as background service, and the *vcmipidemo* connects to it. Then you can use the *vcimgnetclient.py* on your PC to view live captured images.

But for the first run it is better to just run the *vcmipidemo* and check if it shows the ascii representation. This works without any network cable attached. You can then output the captured image to the framebuffer of the display by using the *-f* command line switch.

3.5.1 Compile the programs

1. Change to the subdirectory named *vc_mipi_demo/src*.
2. The source directory contains a Makefile to compile the driver. Do so by calling:

```
make clean all
```

3.5.2 Execute the demo

Just run the demo itself:

```
./vcmipidemo
```

or with framebuffer output:

```
./vcmipidemo -f
```

or with live view over ethernet:

```
./vcimgnetsrv &
./vcmipidemo
```

For live view over ethernet, execute the *vcimgnetclient.py* at the client (needs Python 2 and PyGTK).

3.6 Switching Sensor Configuration

The sensor driver provides different modes which support several features. They can be switched by changing values of sensor driver parameters.

To list available parameters of the sensor driver kernel module, use the following command:

```
modinfo vc_mipi_ov9281
```

The current parameter values of the loaded sensor driver module can be readout as content of a file named after the parameter name at the following directory:

```
/sys/module/vc_mipi_ov9281/parameters/
```

To change the sensor behaviour on the fly the drivers needs to be reloaded. First unload the platform driver (bcm2835-unicam) then the sensor driver (vc_mipi_ov9281), followed by loading the platform driver (bcm2835-unicam) with possible other parameters, and finally loading the sensor driver (vc_mipi_ov9281) with its new parameter set.

Parameters can also be set for bootup time. To do so, add the desired value to the parameter at */boot/cmdline.txt*; the syntax is:

```
moduleName.parameterName=parameterValue
```

Example

?! Parameters are shown after executing the following command:

```
modinfo vc_mipi_ov9281
```

Here is a part of a sample output **which will be different at your setup**:

```
⋮
parm:           sensor_mode:VC Sensor Mode: 0=10bit_stream 1=8bit_stream 2=10bit_ext_trig 3=8bit_ext_trig (int)
⋮
```

Providing that the driver is loaded, the current value of the parameter *sensor_mode* can be displayed by executing:

```
cat /sys/module/vc_mipi_ov9281/parameters/sensor_mode
```

To change the *sensor_mode* to 0, execute the following commands:

```
sudo /sbin/modprobe -r bcm2835-unicam
sudo /sbin/modprobe -r vc_mipi_ov9281
sudo /sbin/modprobe bcm2835-unicam debug=3
sudo /sbin/modprobe vc_mipi_ov9281 sensor_mode=0
```

If you are happy with it, and you want to have this setting available directly after booting, add the following entry to the linux kernel commandline */boot/cmdline.txt*:

```
vc_mipi_ov9281.sensor_mode=0
```

4 Troubleshooting and Background Information

4.1 Q/A

Problem:

Running *make* fails with an error:

```
⋮
make[1]: *** /lib/modules/4.14.79-v7+/build: No such file or directory. Stop.
⋮
```

Solution:

The system needs the build tools of the kernel to build the sensor driver (which itself is a kernel module). They can be obtained by installing the RaspberryPi Kernel Headers package named *raspberrypi-kernel-headers*, see https://www.raspberrypi.org/documentation/linux/kernel/headers.md

---

Problem:

The sensor module driver cannot be started, it shows an error:

```
⋮
[ 4.773298] ov9281 0-0060: Error -5 setting default controls
[ 4.773346] ov9281: probe of 0-0060 failed with error -5
⋮
```

Solution:

Be sure no other device is connected to the I²C bus 0! For example, the touch screen controller of the Raspberry PI display may not be connected.
Check the orientation of the cable at the sensor side as well as at the cpu side. Also check if the cable and the sockets are orthogonal.

4.2 Driver Knowledge

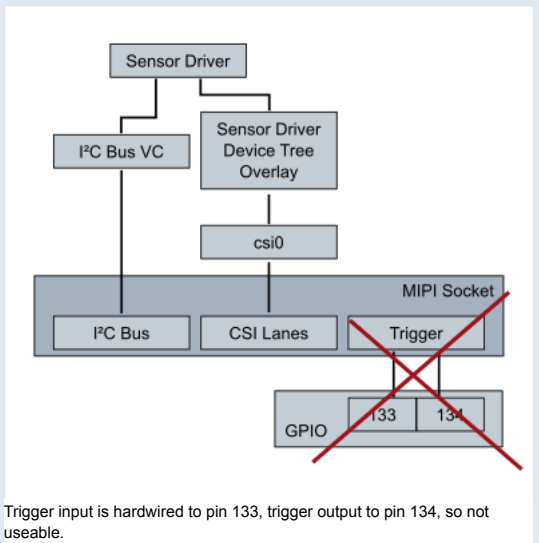
The following tasks have to be done to do an image acquisition with the camera sensor:

- Information about the new sensor hardware and its connector must be provided to the kernel by adding it to the so-called kernel device tree as overlay.
- This device tree overlay must be applied to the kernel device tree.
- For the driver to communicate with the sensor the I²C bus must be set up to connect between the CPU and the MIPI socket.
- The driver itself must be installed as kernel modules.
- Contiguous memory must be reserved for the captured image.

Note

On this Raspberry PI model the pins used for trigger input and output are hard-wired to some GPIO, so external triggering is not possible.

4.2.1 Providing the device tree overlay



Trigger input is hardwired to pin 133, trigger output to pin 134, so not useable.

The so-called kernel device tree overlay contains information about the socket and periphery where the mipi module is connected to. For the RaspberryPi 3B+ there is only one socket available, so there is no need to change the CSI port information at this device tree overlay.

Here are the steps to compile the device tree overlay by yourself:

At the driver source directory the device tree overlay source file can be found. It is named

vc_mipi_ov9281-overlay.dts

1. Install the *device-tree-compiler* package via:

```
sudo  apt-get install  device-tree-compiler
```

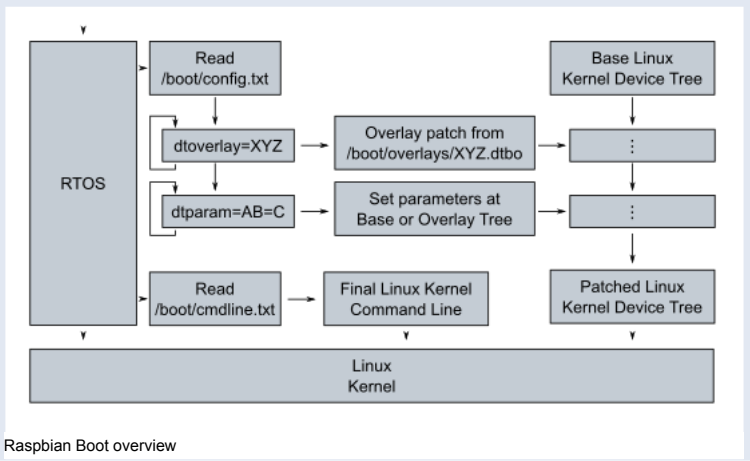
2. Compile the dtbo kernel device tree overlay binary representation by using the following command:

```
dtc -@ -I dts -O dtb -o vc_mipi_ov9281.dtbo vc_mipi_ov9281-overlay.dts
```

3. Copy the binary to

```
/boot/overlays/vc_mipi_ov9281.dtbo
```

4.2.1.1 Telling the RTOS to use the device tree overlay



Raspbian Boot overview

Before starting the linux kernel, the Raspberry Pi first boots a real-time operating system (RTOS) on the GPU. This RTOS looks into the file */boot/config.txt*. It loads a default Kernel device tree and patches it by overlaying the device tree parts listed by the *dtoverlay* entries at the file */boot/config.txt*. To add information about the new sensor this config-file needs the following entry:

```
dtoverlay=vc_mipi_ov9281
```

The device tree will then be modified by the overlay at

```
/boot/overlays/vc_mipi_ov9281.dtbo
```

before the linux kernel is run. With this customized kernel device tree the linux kernel will be started and - as a result - provides the node */dev/video0* using the capture driver module at

```
/lib/modules/$(uname -r)/kernel/drivers/media/i2c/vc_mipi_ov9281.ko
```

To check the behaviour of the RTOS one can look at the output by the following command:

```
sudo vcdbg log msg
```

Example

?! After reboot the applied overlays can be shown by executing the following command:

```
sudo vcdbg log msg  2>&1 | grep '^[0-9\.]\+: Loaded overlay'
```

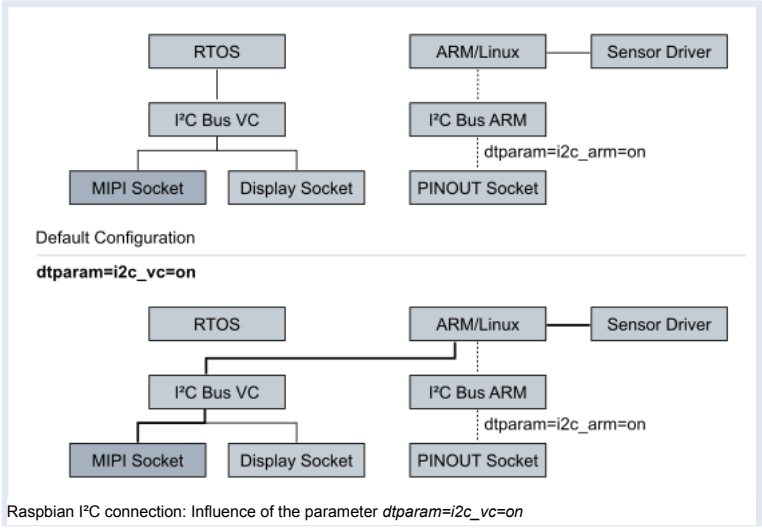
Here is a sample output:

```
002143.555: Loaded overlay 'ov9281'
```

A deeper insight into the device tree overlays and parameters can be found at https://www.raspberrypi.org/documentation/configuration/device-tree.md

4.2.2 Set up the I²C bus for driver-sensor communication

The sensor driver needs to communicate via the I²C Bus named *VC*. To be able to access it, assigning it to the CPU is mandatory.



Raspbian I²C connection: Influence of the parameter *dtparam=i2c_vc=on*

The I²C bus is assigned by the RTOS. So the file '/boot/config.txt' needs an additional entry:

```
dtparam=i2c_vc=on
```

It changes the physical I²C bus *VC* accessor from the default, the GPU, to the CPU.

Note

Some hardware like the touch display demands exclusiveness over the I²C Bus *VC* or their drivers assume the I²C Bus *VC* is connected to the RTOS. Since the sensor driver must communicate with the sensor module connected to the MIPI socket, neither the exclusiveness nor the RTOS connectedness is given. So the I²C bus *VC* cannot be used for other purposes when the sensor is attached.

Example

After reboot the *dtparam* line can be shown by executing the following command:

```
sudo vcdbg log msg  2>&1 | grep '^[0-9\.]\+: dtparam:'
```

Here is a sample output:

```
002077.358: dtparam: audio=on
002096.467: dtparam: i2c_vc=on
```

4.2.3 Providing the sensor driver as kernel module

Here are the steps to compile the kernel modules by yourself:

1. Change to the subdirectory named

   vc_mipi_driver_raspberryPi3_ov9281/

2. The source directory contains a Makefile to compile the driver. Do so by calling:

   ```
   make clean all
   ```

The directory then contains the driver as two modules. They must be copied to their places

- vc_mipi_ov9281.ko to */lib/modules/$(uname -r)/kernel/drivers/media/i2c/*
- bcm2835-unicam.ko to */lib/modules/$(uname -r)/kernel/drivers/media/platform/*

Afterwards the new modules must be registered by calling *depmod -a*.

The module drivers will then be loaded by calling the following commands in that order:

```
modprobe vc_mipi_ov9281
modprobe bcm2835-unicam
```

Example

After reboot you can display the output of the vc_mipi_ov9281 kernel module by executing the following command:

```
dmesg | grep '^[^]]*\] vc_mipi_ov9281:'
```

Here is a sample output **which will be different at your setup**:

```
[ 4.107734] vc_mipi_ov9281: loading out-of-tree module taints kernel.
[ 4.231604] vc_mipi_ov9281 0-0060: VC Sensor MODE=0 PowerOn STATUS=0x80 i=1
[ 4.231719] vc_mipi_ov9281 0-0060: GAIN = 12
[ 4.239752] vc_mipi_ov9281 0-0060: EXPOSURE = 12345
[ 4.252885] vc_mipi_ov9281 0-0060: Model ID 0x1234, Lot ID 0x123456, Chip ID 0x1234
```

4.2.4 Reserving Contiguous Memory for the Image Captures

In contrast to the normally used non-contiguous memory the capture hardware needs a contiguous memory region to transfer pixel data to by using direct memory access (DMA).

To reserve 128MByte of memory for capturing images, edit the file named

/boot/cmdline.txt

and append the following key-value-pair to the current line:

cma=128M

After reboot the kernel message line beginning with *Memory:* will show an additional entry:

131072K cma-reserved

Example

?! After reboot you can show the line by executing the following command:

```
dmesg | grep '^[^]]*\] Memory:'
```

Here is a sample output which will look slightly different at your setup:

```
[    0.000000] Memory: 817108K/970752K available (7168K kernel code, 576K rwdata, 2076K rodata, 1024K init, 698K bss, 22572K reserv
```

Vision Components GmbH
Ottostr. 2
76275 Ettlingen
Germany

Phone: +49 (0) 7243 2167-0
www.vc-linux.com
www.vision-components.com