Mira EVK

Quick Start Guide

Evaluation kit for Mira CMOS image sensor family

Published by ams-OSRAM AG
Tobelbader Strasse 30, 8141 Premstaetten, Austria
Phone +43 3136 500-0
ams-osram.com
© All rights reserved

Table of contents

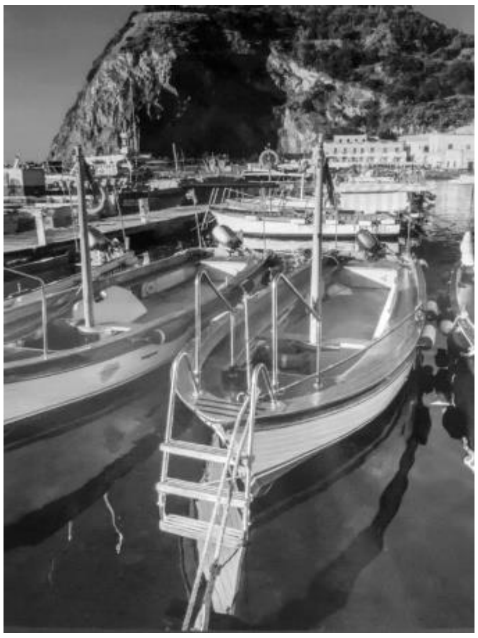
1 Introduction

Mira is a family of NIR-enhanced, global shutter CMOS image sensors with MIPI output interface.

The Mira Evaluation Kit is a platform for evaluating Mira products. This kit is based on Raspberry Pi® 4B. The Mira image sensor comes on a separate sensor board, included in the kit. The Raspberry Pi can operate standalone or connected to a laptop using a network or USB interface.

The purpose of this quick start guide is not to explain the sensor functionality, nor will it replace the Raspberry Pi manual. For that purpose, please refer to the appropriate datasheet/manual. The goal of this document is to get started quickly with this evaluation kit, to connect the camera board to Raspberry Pi™ board, how to configure the image sensor and how to capture images and videos.

Figure 1: Picture captured with Mira050 on the evaluation kit



Referring documents:

For further information on specific components of the Mira Evaluation Kit, please refer to the following documents:

- Mira220 Datasheet (DS000642)
- Mira050 Datasheet (up on request)
- Mira016 Datasheet (up on request)
- Libcamera framework (must-read)
- Raspberry Pi documentation
- The Picamera2 Library (raspberrypi.com)

1.1 Key features

- Mira image sensors support
- Fully configured ISP for best image quality
- Software based on libcamera, an open source camera stack and framework for Linux, Android, and ChromeOS
- RAW image capture
- Scripting possible in python using the picamera2 library
- REST API for remote control
- Web interface for remote control
- Standalone or USB/networked mode

2 Out of the box

Out of the box, there are multiple hardware components included, shown in Figure 2. They are listed below:

- Raspberry Pi 4B
- Mira130, Mira220 or Mira050 sensor board
- MIPI flat cable assembly
- M12 lens, lens holder and screws
- Quick start guide
- USB-C to USB-A cable (needed when connecting to a laptop)
- Hardware description

Not included but can be purchased separately for use in standalone mode (without laptop):

- Mini HDMI cable
  raspberrypi.com/products/micro-hdmi-to-standard-hdmi-a-cable/
- Raspberry Pi Power supply (multi-region)
  raspberrypi.com/products/type-c-power-supply/

Figure 2: Raspberry Pi and sensor board

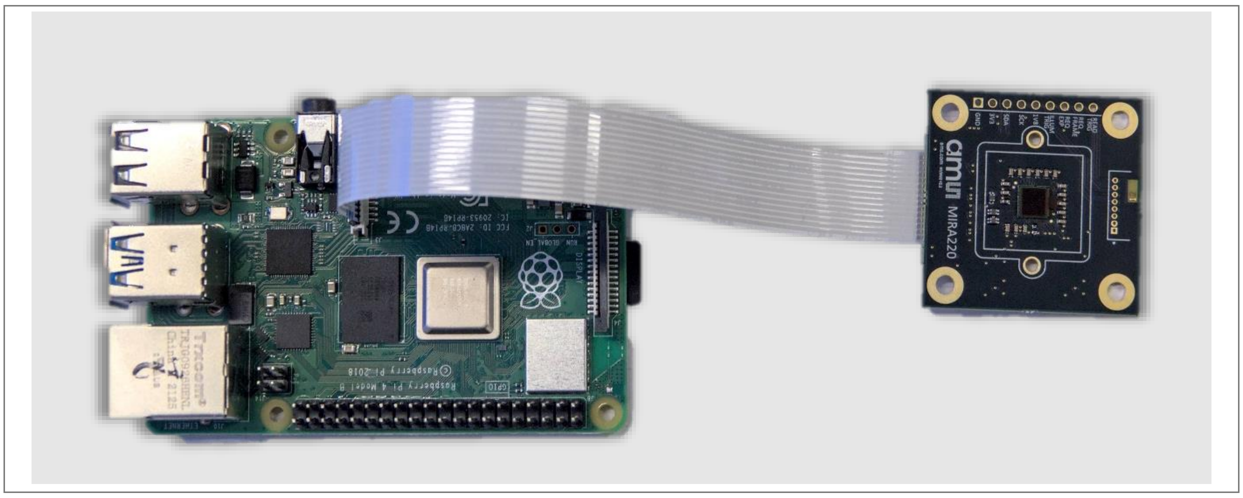
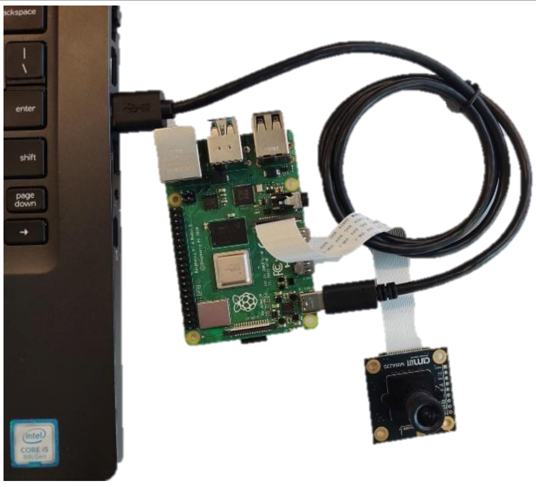


Figure 3: Connection of hardware components



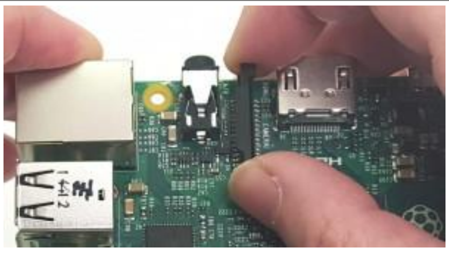
3 Getting started

3.1 Preparation of the hardware

› Open the camera port on the Raspberry Pi:

On the Raspberry Pi, the camera port is between the audio port and the HDMI port. To open the port, use two fingers and lift the ends up slightly. Please note that there is another port on the Pi board that looks just the same, but that other one is not meant for the camera.

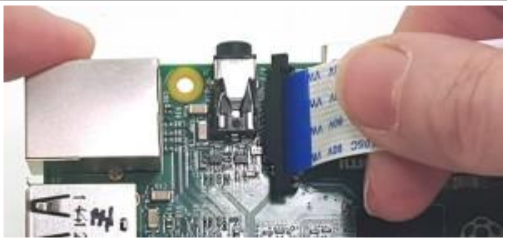
Figure 4: Camera port



› Insert the camera cable:

The cable has to be inserted with the right orientation: The blue has to face the Ethernet port, and the silver side is facing the HDMI port. Insert the cable so that almost no blue is showing. This photo shows the beginning of the insertion.

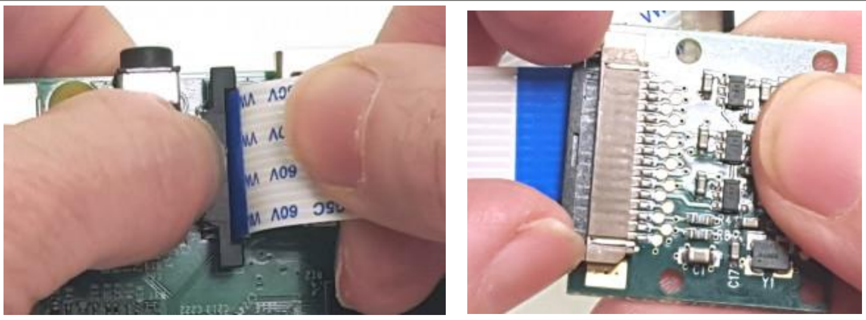
Figure 5: Insertion of camera cable



› Close the camera port:

To close the port and keep the cable snugly in place, push the top of the port while holding the cable with the other hand. Because you're pushing it down, more blue will be exposed. That's okay.

Figure 6: Closing the camera port



› Connecting and removing the cable from the camera board itself:

On the back of the camera, you will find a similar cable port. It opens and closes the same way as the one on the Raspberry Pi but it requires a little bit more pressure to get it opened than the one on the Pi. Mind the orientation of the blue connector.

In case of the USB peripheral mode (recommended):

› Plug in a USB-C cable into your laptop's USB 3 port. The USB port must be able to provide enough power. The smaller USB-C connector must be connected to the Pi. See Figure 9.

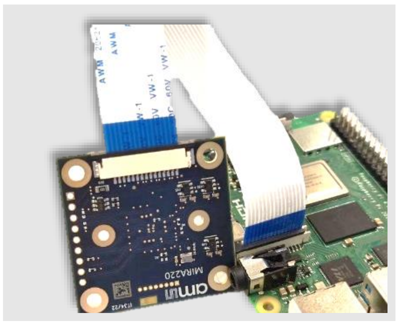
Information:

- The kit will boot automatically after power is connected. You should see some activity on the red and green LEDs.

Attention:

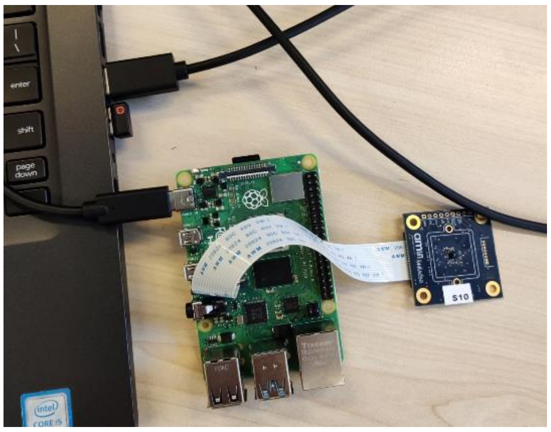
- Please pay attention to the metal contact side of the flat cables, which should face the correct direction, as shown in the example. The non-metal contact side is shielded by blue or black plastic.
  Also, see this video.

Figure 7: Cable connection for all Mira reference designs(1)



(1) Be very careful not to break the connector

Figure 8: Connecting the kit in USB peripheral mode



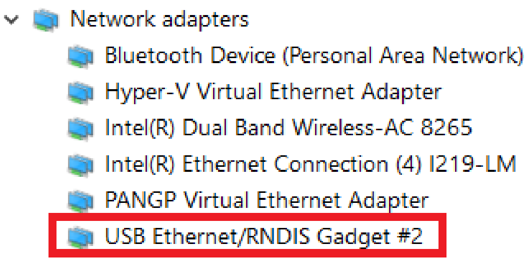
(1) Mind the orientation of the cable connection for all Mira220/Mira050 reference designs

3.2 Software

After connecting the Pi to your windows laptop, it will take about 2 minutes for the device to prepare itself for the first boot.

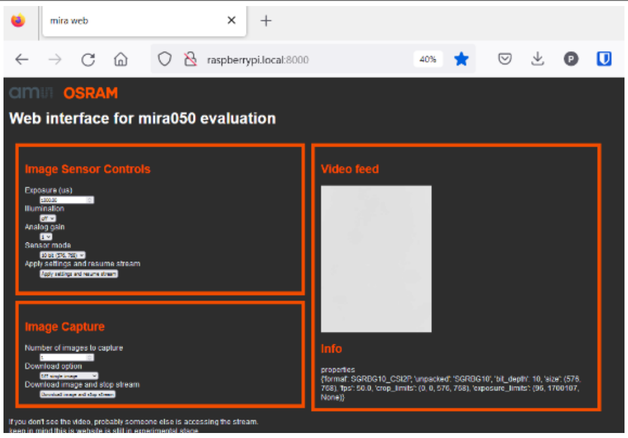
If you want to check if the eval kit is recognized by your laptop, visit the network adapters setting on your windows laptop. There should be a USB ETHERNET/RNDIS GADGET appearing.

Figure 9: Network adapters



Network adapters
- Bluetooth Device (Personal Area Network)
- Hyper-V Virtual Ethernet Adapter
- Intel(R) Dual Band Wireless-AC 8265
- Intel(R) Ethernet Connection (4) I219-LM
- PANGP Virtual Ethernet Adapter
- USB Ethernet/RNDIS Gadget #2

Figure 10: USB web viewer



Web interface for mira050 evaluation

properties
{'format': SGRBG10_CSI2P, 'unpacked': 'SGRBG10', 'bit_depth': 10, 'size': (576, 768), 'fps': 50.0, 'crop_limits': (0, 0, 576, 768), 'exposure_limits': (96, 1700107, None)}

If you don't see the video, probably someone else is accessing the stream.
Keep in mind this is website is still in experimental stage

3.2.1 Web viewer

There is a web server running on the evaluation kit. This can be accessed from the network or via USB mode. See Figure 10.

› Open a web browser and visit:
`raspberrypi.local:8000`

3.2.2 Jupyter lab for scripting

Along with the web interface, there is also a REST API to interface with the camera.

There is an example notebook at:

› `raspberrypi.local:8888`

The code in this notebook can be run in the web browser or can be run from a Windows machine.

Example notebooks are available to write registers, to capture a sequence of images.

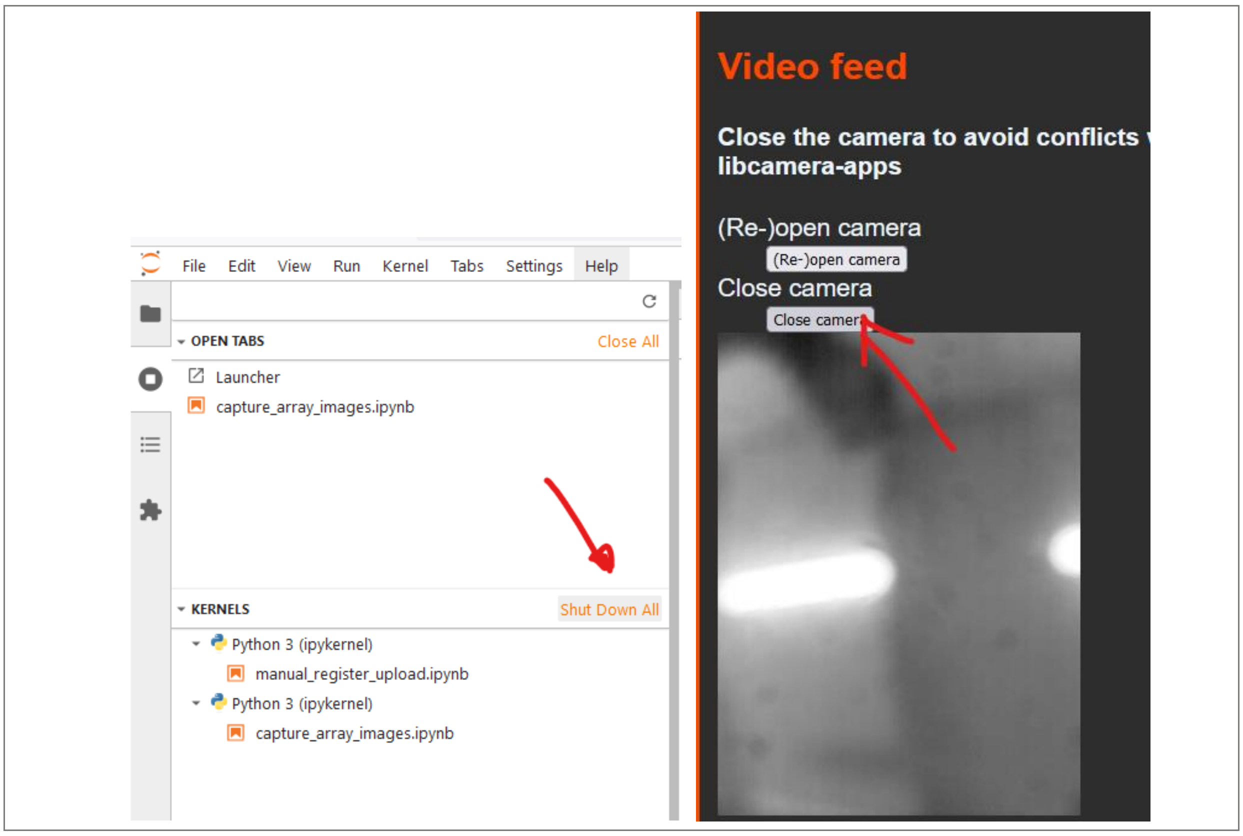
CAUTION:

When switching from the Web Viewer to the Jupyter Lab, make sure to 'close camera' first.

Information:

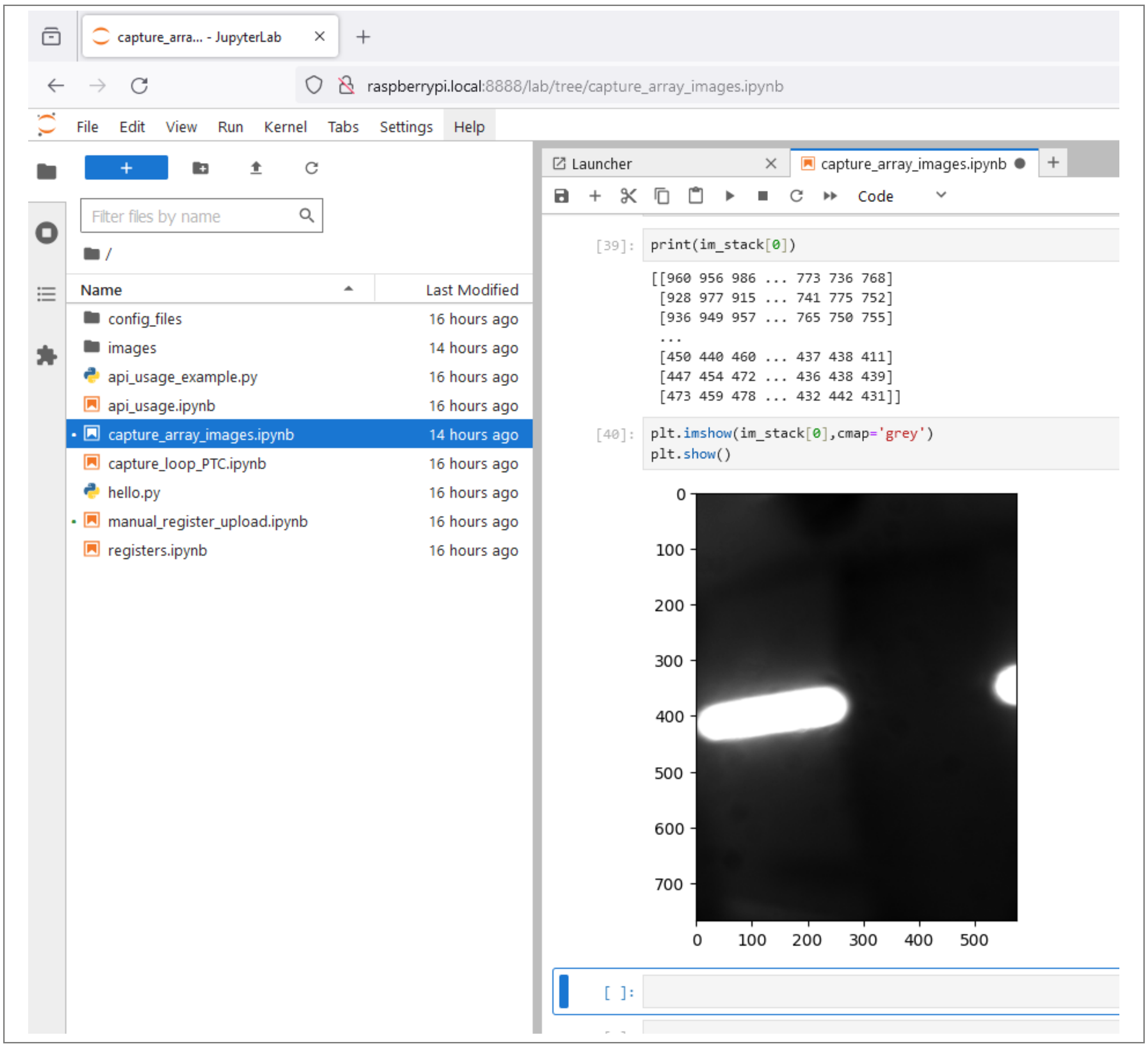
When switching from Jupyter notebook to web viewer, make sure to close the Jupyter kernels.

Figure 11: Switching from Jupyter notebook to web viewer



Additional notebooks/scripts can be offered on request.

Figure 12: Jupyter lab


capture_arra... - JupyterLab
raspberrypi.local:8888/lab/tree/capture_array_images.ipynb
File Edit View Run Kernel Tabs Settings Help
Filter files by name
Launcher
capture_array_images.ipynb
Code

Name	Last Modified
config_files	16 hours ago
images	14 hours ago
api_usage_example.py	16 hours ago
api_usage.ipynb	16 hours ago
capture_array_images.ipynb	14 hours ago
capture_loop_PTC.ipynb	16 hours ago
hello.py	16 hours ago
manual_register_upload.ipynb	16 hours ago
registers.ipynb	16 hours ago

```
[39]: print(im_stack[0])

[[960 956 986 ... 773 736 768]
 [928 977 915 ... 741 775 752]
 [936 949 957 ... 765 750 755]
 ...
 [450 440 460 ... 437 438 411]
 [447 454 472 ... 436 438 439]
 [473 459 478 ... 432 442 431]]

[40]: plt.imshow(im_stack[0],cmap='grey')
      plt.show()
```

4 In depth overview of the software stack

The Raspberry Pi is a single board computer, developed by the Raspberry Pi Foundation. Just like any computer, it needs to run an Operating System (think of Windows or Mac OS). This Raspberry Pi runs a Linux based operating system called Raspberry Pi OS.

The operating system can be split into two parts. The kernel space and the user space. The drivers, interacting with hardware, can be found in the kernel space. Most user applications will run in the user space. To interact with a piece of hardware, an appropriate driver needs to be present in the kernel.

The Raspberry Pi features a MIPI connector to which one could connect a camera board (the hardware). This camera board has an image sensor connected to the I²C bus. To properly initialize the camera, there needs to be an I²C camera driver in place, specific for this image sensor, which defines the registers on power-up, when changing exposure etc.

In the user space, there runs an application called Libcamera. This is the default camera framework for Raspberry Pi. This framework receives raw images from the camera and handles it to the V4L2 Broadcom ISP Driver, which does the debayering and other calculations. All control algorithms such as color matrix, sharpening, lens shading, white balancing, AEC/AGC… are handled by the Libcamera framework and translated into instructions for the Broadcom ISP hardware.

Next to the Libcamera framework, there are various applications called Libcamera-apps. The following apps are provided:

- libcamera-hello: A simple "hello world" application which starts a camera preview stream and displays it on the screen.
- libcamera-jpeg: A simple application to run a preview window and then capture high resolution still images.
- libcamera-still: A more complex still image capture application which emulates more of the features of raspistill.
- libcamera-vid: A video capture application.
- libcamera-raw: A basic application for capturing raw (unprocessed Bayer) frames directly from the sensor.
- libcamera-detect: This application is not built by default, but users can build it if they have TensorFlow Lite installed on their Raspberry Pi. It captures JPEG images when certain objects are detected.

There are also bindings for the Python programming language available. It is called picamera2.

Figure 13: Camera framework architecture

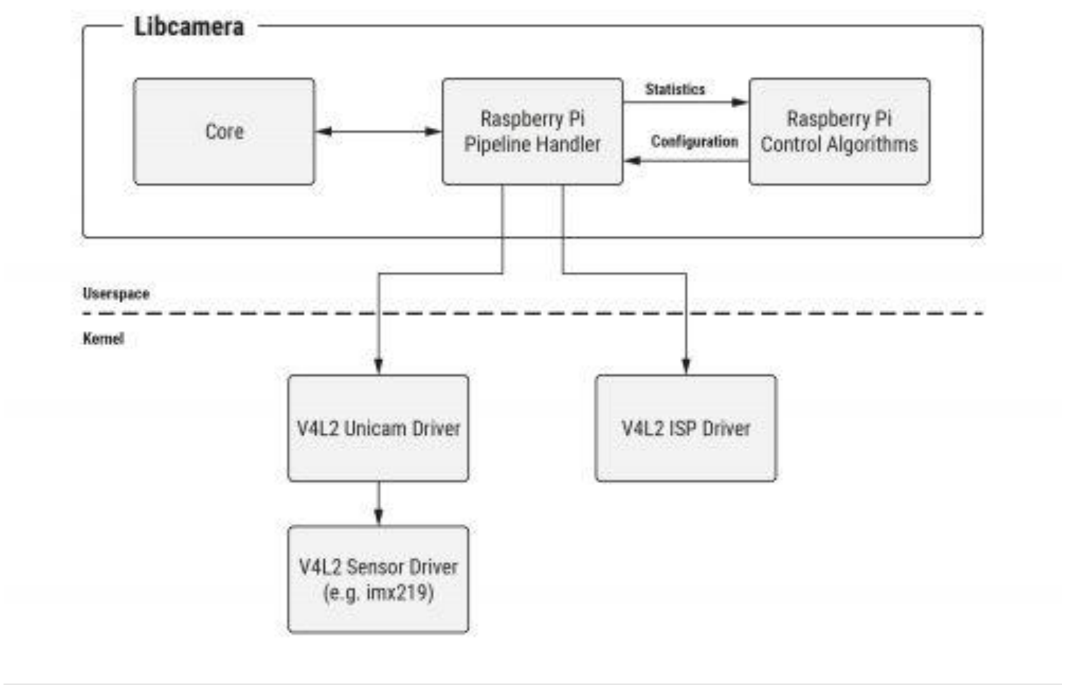
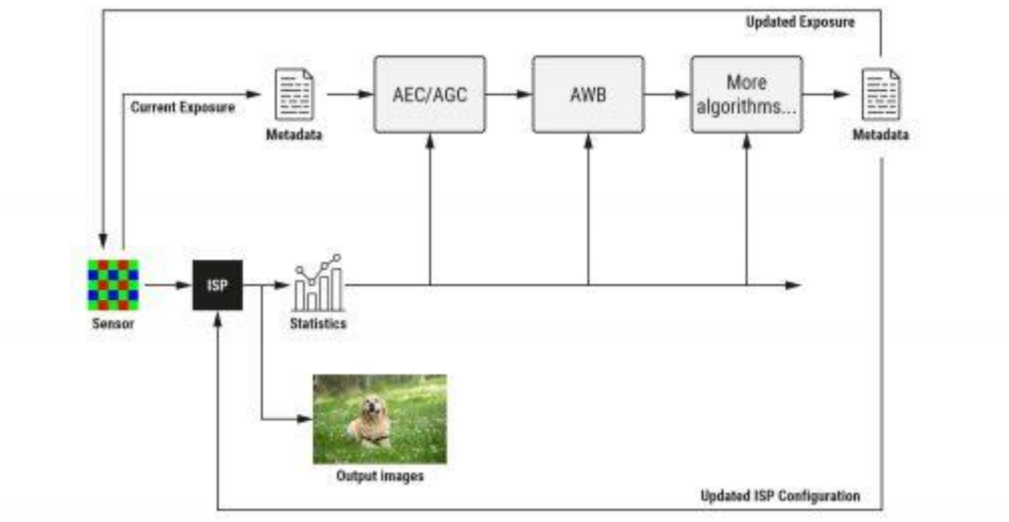

Libcamera
Core
Raspberry Pi Pipeline Handler
Statistics
Configuration
Raspberry Pi Control Algorithms
Userspace
Kernel
V4L2 Unicam Driver
V4L2 ISP Driver
V4L2 Sensor Driver (e.g. imx219)

Figure 14: Control algorithm



4.1 The camera driver

On the eval kit, there is a folder called ams_rpi_kernel.

In this folder, the drivers for all Mira cameras reside.

Using the script build_native.sh, one can modify and apply a customized driver.

4.2 The user space: Libcamera

On the eval kit, there is a folder called ams_rpi_software.

In this folder, the libcamera related items reside.

Also, the camera tuning files can be found here.

4.2.1 Capturing images with Libcamera

Information:

- Before running any commands, make sure the web service is disabled. See chapter 4.2.5.

For more details on the Libcamera-apps, we like to refer to the documentation.

Our image sensors and drivers are fully compatible with the libcamera framework, so please refer to their documentation.

Some examples will be provided below in a terminal window.

› Open the terminal by pressing Ctrl+Alt+T

View camera modes:

```
libcamera-hello --list-cameras
```

This command will list the available resolutions and bit modes.

To use the mode in one of the following commands, add the --mode argument, e.g. 1600:1400:10:P for a 10-bit image output.

Capture a jpg image:

The -o arguments is the output filename.

```
Libcamera-still -o test.jpg
```

Note that jpg images are compressed and not suited for sensor evaluation.

Capture raw image:

Add a -r argument to also save a raw image with *.dng extension.

```
libcamera-still -r -o test.jpg --qt-preview
```

Modify shutter time and gain. The -t alters the preview window time before capturing.

```
libcamera-still -r -o test.jpg -t 2000 --shutter 20000 --gain 1.5 --qt-
preview
```

Get help on the arguments:

```
libcamera-still --help
```

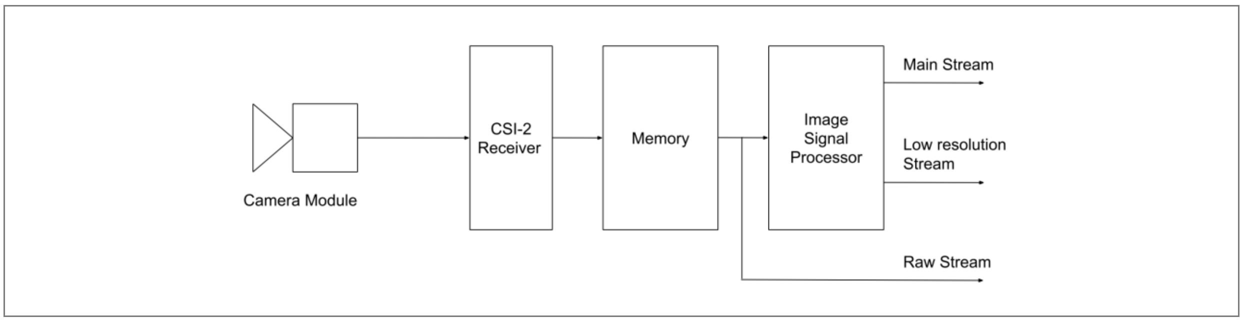
Capture video:

libcamera-vid is the video capture application. By default, it uses the Raspberry Pi's hardware H.264 encoder. It will display a preview window and write the encoded bitstream to the specified output. For example, to write a 10 second video to file use.

```
libcamera-vid -t 10000 -o test.h264 --qt-preview
```

```
libcamera-vid -t 10000 -o test.h264 --mode 1600:1400:10:P
```

Figure 15: Camera pipeline



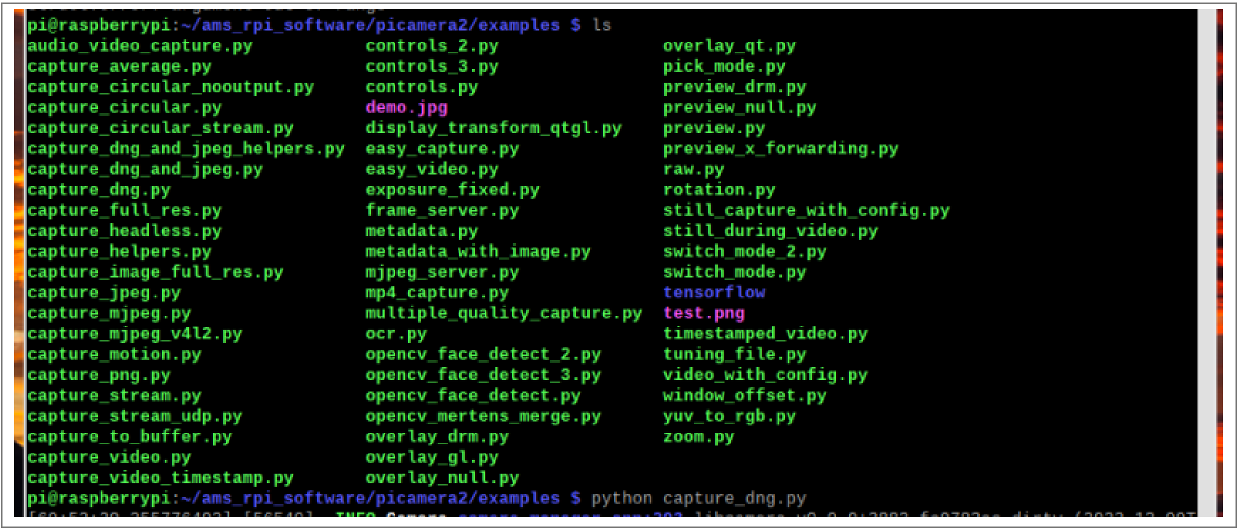
4.2.2 PiCamera2

There is a project called PiCamera2, which tries to create python bindings for the Libcamera framework.

This makes it very simple for users to create their own applications, either in a python script, or a graphical interface.

A few examples that come pre-installed in the `~/ams_rpi_software/picamera2/examples` folder are shown below.

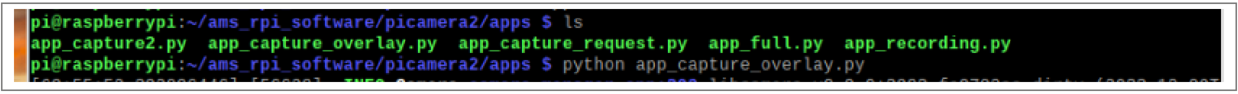
Figure 16: Examples of PiCamera2 preinstalled applications



```
pi@raspberrypi:~/ams_rpi_software/picamera2/examples $ ls
audio_video_capture.py            controls_2.py                  overlay_qt.py
capture_average.py                controls_3.py                  pick_mode.py
capture_circular_nooutput.py      controls.py                    preview_drm.py
capture_circular.py               demo.jpg                       preview_null.py
capture_circular_stream.py        display_transform_qtgl.py      preview.py
capture_dng_and_jpeg_helpers.py   easy_capture.py                preview_x_forwarding.py
capture_dng_and_jpeg.py           easy_video.py                  raw.py
capture_dng.py                    exposure_fixed.py              rotation.py
capture_full_res.py               frame_server.py                still_capture_with_config.py
capture_headless.py               metadata.py                    still_during_video.py
capture_helpers.py                metadata_with_image.py         switch_mode_2.py
capture_image_full_res.py         mjpeg_server.py                switch_mode.py
capture_jpeg.py                   mp4_capture.py                 tensorflow
capture_mjpeg.py                  multiple_quality_capture.py    test.png
capture_mjpeg_v4l2.py             ocr.py                         timestamped_video.py
capture_motion.py                 opencv_face_detect_2.py        tuning_file.py
capture_png.py                    opencv_face_detect_3.py        video_with_config.py
capture_stream.py                 opencv_face_detect.py          window_offset.py
capture_stream_udp.py             opencv_mertens_merge.py        yuv_to_rgb.py
capture_to_buffer.py              overlay_drm.py                 zoom.py
capture_video.py                  overlay_gl.py
capture_video_timestamp.py        overlay_null.py
pi@raspberrypi:~/ams_rpi_software/picamera2/examples $ python capture_dng.py
```

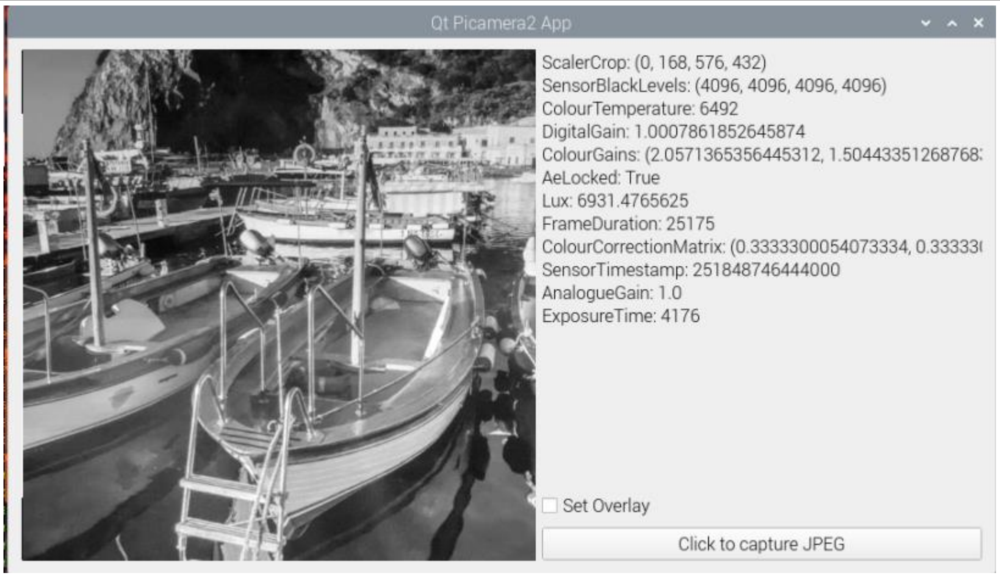
Also a few graphical examples come preinstalled such as:

Figure 17: Examples of PiCamera2 preinstalled applications



```
pi@raspberrypi:~/ams_rpi_software/picamera2/apps $ ls
app_capture2.py  app_capture_overlay.py  app_capture_request.py  app_full.py  app_recording.py
pi@raspberrypi:~/ams_rpi_software/picamera2/apps $ python app_capture_overlay.py
```

Figure 18: PiCamera2 app



```
ScalerCrop: (0, 168, 576, 432)
SensorBlackLevels: (4096, 4096, 4096, 4096)
ColourTemperature: 6492
DigitalGain: 1.0007861852645874
ColourGains: (2.0571365356445312, 1.50443351268768
AeLocked: True
Lux: 6931.4765625
FrameDuration: 25175
ColourCorrectionMatrix: (0.3333300054073334, 0.33333
SensorTimestamp: 251848746444000
AnalogueGain: 1.0
ExposureTime: 4176
```

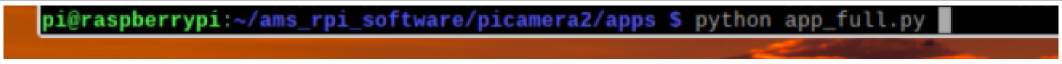
4.2.3 Full-featured PiCamera2 GUI

Another useful application is provided. There is a shortcut on the desktop to launch it. It allows you to capture images/videos and adjust gain/exposure and tune the ISP.

Information:

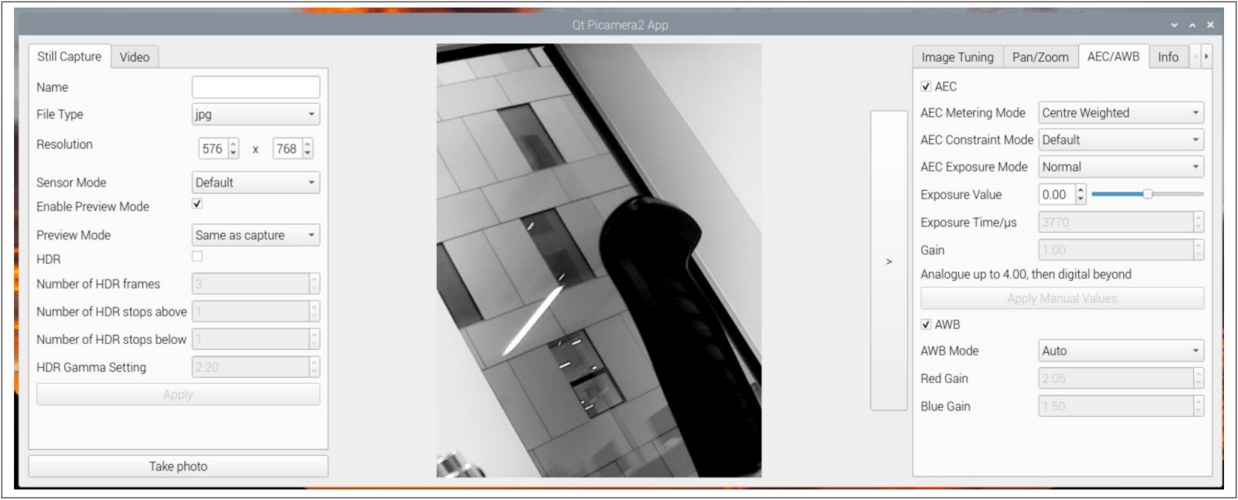
- Keep in mind, by default, the auto exposure and gain is enabled. Disable this and manually configure the exposure to have fine control.
- To capture RAW images, select either TIFF or DNG. Other formats such as jpg, png will be compressed images and are unsuitable for image characterization.

Figure 19: App_full.py



```
pi@raspberrypi:~/ams_rpi_software/picamera2/apps $ python app_full.py
```

Figure 20: PiCamera2 GUI



Name:
File Type: jpg
Resolution: 576 x 768
Sensor Mode: Default
Enable Preview Mode: ✓
Preview Mode: Same as capture
HDR: ☐
Number of HDR frames: 3
Number of HDR stops above: 1
Number of HDR stops below: 1
HDR Gamma Setting: 2.20

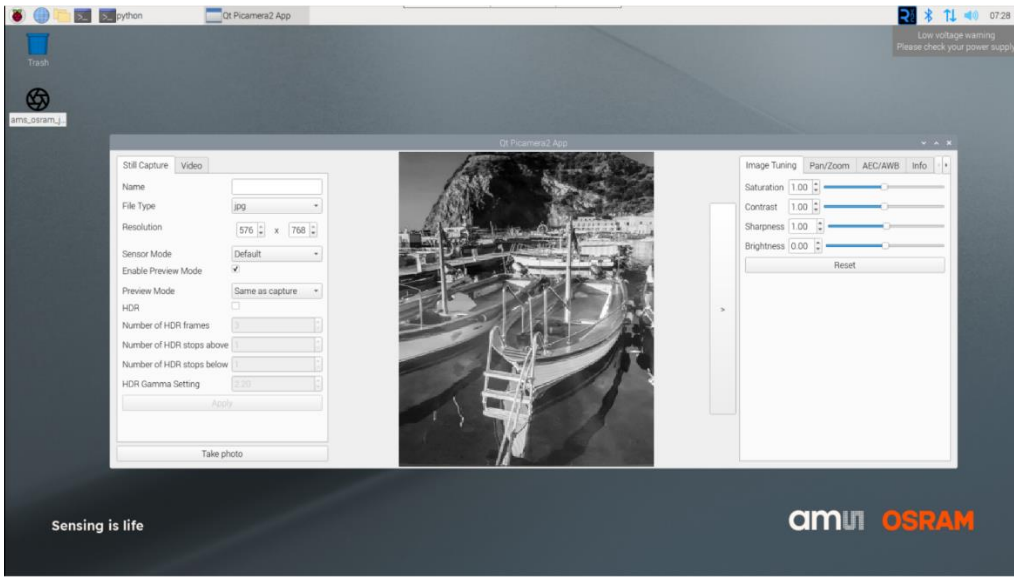
AEC: ✓
AEC Metering Mode: Centre Weighted
AEC Constraint Mode: Default
AEC Exposure Mode: Normal
Exposure Value: 0.00
Exposure Time/µs: 3770
Gain: 1.00
Analogue up to 4.00, then digital beyond
AWB: ✓
AWB Mode: Auto
Red Gain: 2.06
Blue Gain: 1.50

4.2.4 Launching the app

After boot, you should end up on the Raspberry Pi desktop.

1. To start capturing images, click the ams_osram icon on the desktop.
2. For more details on this app, go to 4.2.3 below.

Figure 21: The Raspberry Pi OS desktop with the Picamera2 application



Low voltage warning
Please check your power supply

Name:
File Type: jpg
Resolution: 576 x 768
Sensor Mode: Default
Enable Preview Mode: ✓
Preview Mode: Same as capture
HDR: ☐
Number of HDR frames: 3
Number of HDR stops above: 1
Number of HDR stops below: 1
HDR Gamma Setting: 2.20

Saturation: 1.00
Contrast: 1.00
Sharpness: 1.00
Brightness: 0.00

Information:

- If this step fails, verify the cable and go to chapter 4.2.6.

- Default username: pi
- Default password: pi

4.2.5 A web application – running in the background

There is a service, launched at boot, which runs a web server. This web server blocks the use of the camera, so if you want to use the video stream with other apps, you must disable this service.

The status of this web app can be checked using the following commands:

```
systemctl status picamera2-flask.service
```

To stop the service:

```
systemctl stop picamera2-flask.service
```

This command will be persistent over boot.

To start the service:

```
systemctl start picamera2-flask.service
```

For more details on the web server, see chapter 3.2.1.

4.2.6 Device tree overlay and drivers

Information:

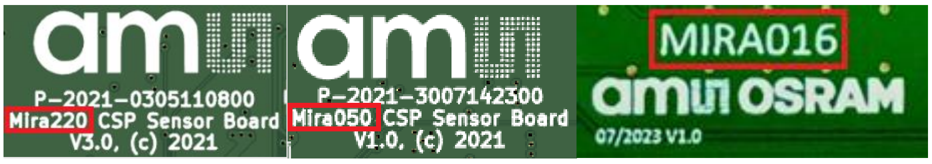
- If the camera works, you can skip this chapter.

If the camera fails, there are two common causes, which can be checked in the following steps:

- The MIPI cable connection is incorrect. Please double check if the metal contacts are facing the correct side and shown in Section 3.1.
- A mismatch between sensor hardware type and driver configuration. Details are provided next.

The Raspberry Pi platform currently supports various types of Mira sensors, such as Mira220, Mira050 and Mira016. Each sensor board has its type printed on the sensor PCB, as shown in Figure 22.

Figure 22: Mira sensor board types printed on the PCB



The driver needs to be configured to match the actual type of Mira sensor connected. To do so, open a command line window and type the following command.

```
› sudo nano /boot/config.txt
```

The above command may ask for a password. After typing the default password, it opens up an editor to edit the file `/boot/config.txt`. Scroll to the bottom of the file, where there is a line that specifies the device tree overlay "`dtoverlay`" for the Mira sensor.

If a Mira220 COLOR sensor is connected, the line should be the following (all letters are in lower case).

```
› dtoverlay=mira220color
```

The same idea applies to Mira050, Mira016 and others.

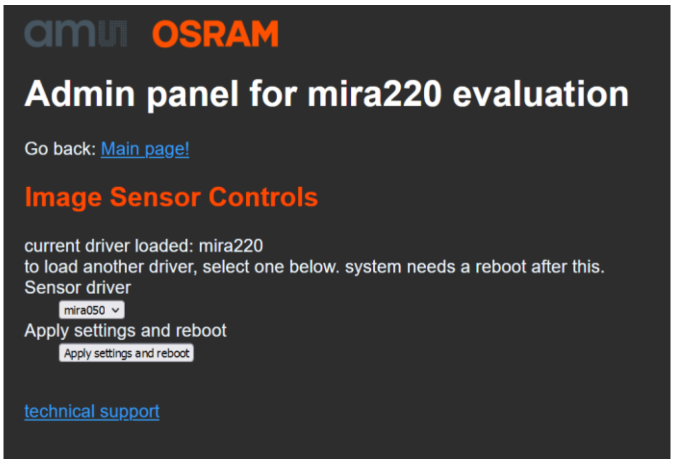
Please use the correct one (not both!) that matches the connected Mira sensor type. After editing the file, press “Ctrl+o Enter” to write out the changes to file, and then press “Ctrl+x” to exit the editor.

After exiting the editor, the changes require a reboot to take effect. Use either the desktop GUI or the command below to reboot the Raspberry Pi.

```
› sudo reboot
```

Visit `raspberrypi.local/admin` to reach this page. Select the correct sensor, then click apply. The kit will reboot.

Figure 23: Web interface admin panel



Admin panel for mira220 evaluation

Go back: Main page!

Image Sensor Controls

current driver loaded: mira220
to load another driver, select one below. system needs a reboot after this.
Sensor driver
mira050
Apply settings and reboot

5 Dual cameras on compute module 4

The regular Raspberry Pi only supports 1 camera. The Pi Compute Module 4 supports 2 cameras.

What you need:

- Pi compute module 4 or pi5
- 2 sensor boards
- Pi zero camera cable 15 to 22 pin FFC

Figure 24: Dual cameras on the Pi compute module

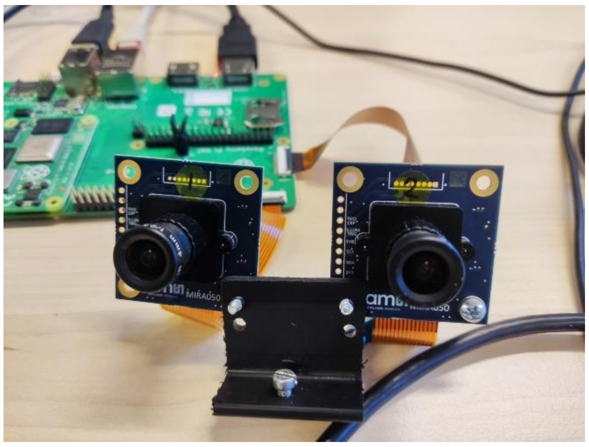
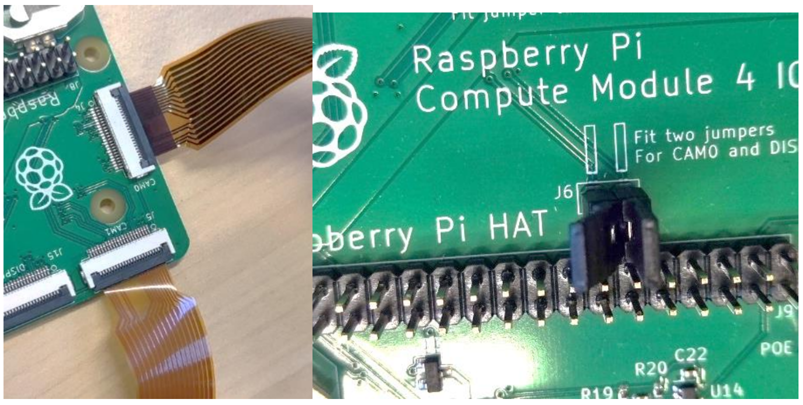


Figure 25: Dual cameras



(1) Connect the flat cable to cam0 and cam1.
(2) Connect the jumpers on J6 (only on pi compute module)

Instructions:

1. (compute module only) On the Compute Module, run `sudo raspi-config` and enable the camera.
2. (compute module only) Run

   ```
   sudo wget https://datasheets.raspberrypi.com/cmio/dt-blob-cam1.bin -O
   /boot/dt-blob.bin
   ```
   Also see here
3. Modify the device tree such as:
   `sudo nano /boot/config.txt`
   add/replace these lines at the bottom.
   ```
   dtoverlay=mira050,cam0
   dtoverlay=mira050
   ```
4. (compute module only) Power the Compute Module down.
5. (compute module only) Connect both camera cables to cam0 and cam1 on the csi port.
6. (compute module only) Connect jumpers on J6.
7. Power up.
   Using commands, you can test both cameras in parallel.

   ```
   rpicam-hello --camera 0 -t 0

   rpicam-hello --camera 1 -t 0
   ```

6 Repairing or upgrading the installation

Information:

- The demo kit has an OS image built in, which directly boots to a log-in screen. In case a new OS image is needed, follow the instructions below.

6.1 Flash OS image on the SD card

To perform this step, the following list of hardware is required:

- A host computer with micro-SD card reader where users have admin privilege. This document provides examples based on Windows OS. Linux OS and Mac OS are also supported.

CAUTION:

Do not power the Raspberry Pi while the SD card is inserted or removed.

Referring documents:

The following example is for a host computer with Windows OS. The same operation can be performed on a host computer with Linux OS or Mac OS, which are described in this document below:

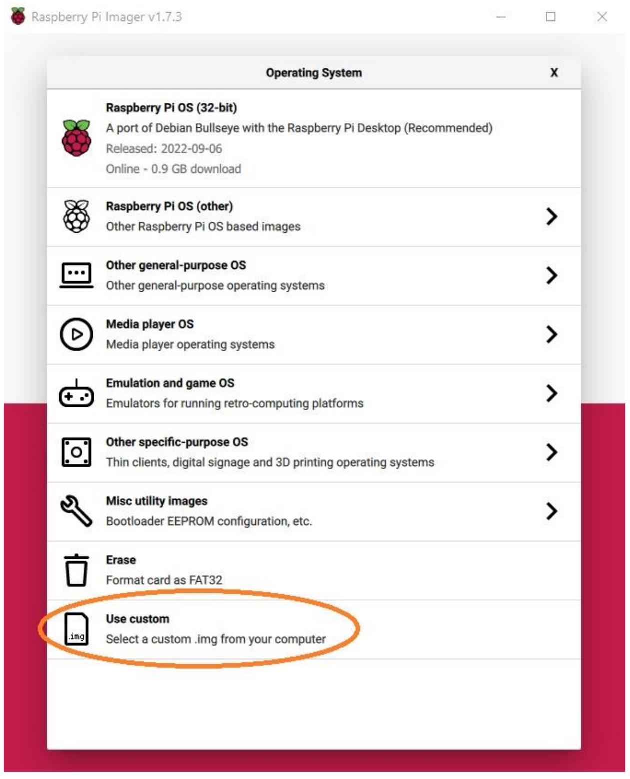
- Raspberry Pi documentation (raspberrypi.com/documentation)

› STEP 1: Obtain a customized Raspberry Pi OS image that has all the required drivers built in. Please contact your local ams OSRAM sales or FAE

Once the OS image file is downloaded, the next step is to write the OS image file onto the SD card.

- STEP 2: Download and install Raspberry Pi Imager (raspberrypi.com/software/) to write the OS image file to the Raspberry Pi. Other tools such as Balena etcher or win32diskimager also work.
- STEP 3: After launching the Raspberry Pi-imager, from the “CHOOSE OS” dropdown menu, select the “Use custom” option, as shown in Figure 26. Then select the OS image file that has been downloaded in the previous step. Afterwards, in the “CHOOSE STORAGE” menu, select the storage device that is the Raspberry Pi SD card. And then click the “WRITE” button and start the flashing.

Figure 26: Choose OS dropdown menu


Raspberry Pi Imager v1.7.3

Operating System

Raspberry Pi OS (32-bit)
A port of Debian Bullseye with the Raspberry Pi Desktop (Recommended)
Released: 2022-09-06
Online - 0.9 GB download

Raspberry Pi OS (other)
Other Raspberry Pi OS based images

Other general-purpose OS
Other general-purpose operating systems

Media player OS
Media player operating systems

Emulation and game OS
Emulators for running retro-computing platforms

Other specific-purpose OS
Thin clients, digital signage and 3D printing operating systems

Misc utility images
Bootloader EEPROM configuration, etc.

Erase
Format card as FAT32

Use custom
Select a custom .img from your computer

(1) In the “CHOOSE OS” dropdown menu, select “Use custom” option, and then select the OS image file that has been downloaded.

› STEP 4: After the Raspberry Pi Imager finishes writing the OS image, eject the storage device from the host computer and put it back in the Pi.

7 Revision information

Changes from previous released version to current revision v2-00	Page
Document contents transferred to latest ams OSRAM template	
Added Mira016 support information	22

- Page and figure numbers for the previous version may differ from page and figure numbers in the current revision.
- Correction of typographical errors is not explicitly mentioned.

8 Legal information

Headquarters	Please visit our website at ams-osram.com
ams-OSRAM AG	For information about our products go to Products
Tobelbader Strasse 30	For technical support use our Technical Support Form
8141 Premstaetten	For feedback about this document use Document Feedback
Austria, Europe	For sales offices and branches go to Sales Offices / Branches
Tel: +43 (0) 3136 500 0	For distributors and sales representatives go to Channel Partners