Product Document

Published by
ams OSRAM Group

Quick Start Guide

QG001035

Mira Evaluation Kit

Using Raspberry Pi

Evaluation Kit for Mira CMOS Image Sensor Family

v1-00 • 2023-Sep-18

Content Guide

1 Introduction

1.1 Description

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 the 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 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

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)
- Libcamera framework (must-read)
- Raspberry Pi Documentation
- The Picamera2 Library (raspberrypi.com)

1.2 Key Features

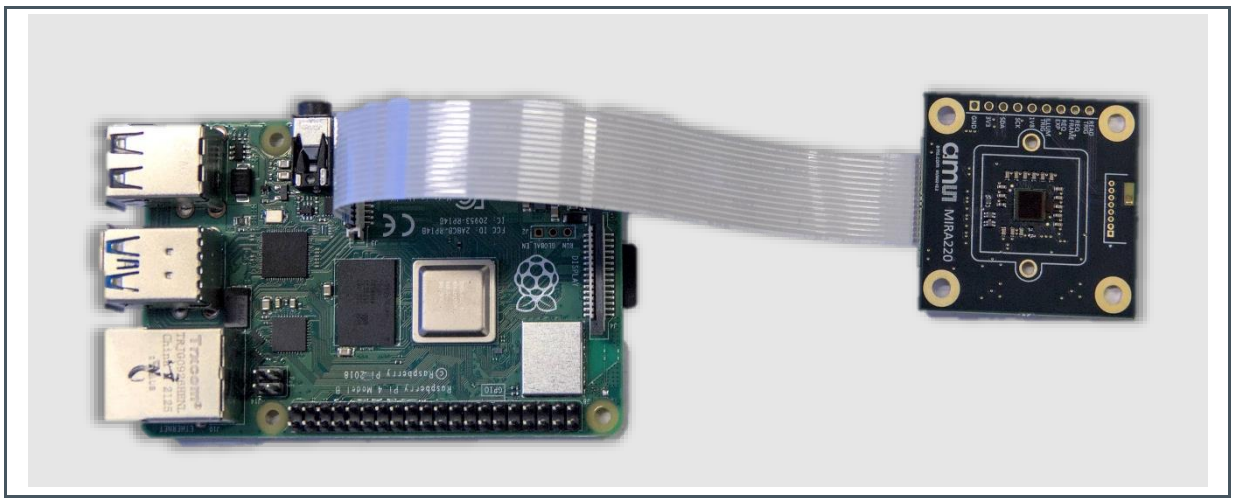
- Mira image sensors support
- Fully configured ISP for best image quality
- Software based on libcamera, an open source camera stack and framework for Linux, Android, and ChromeOS
- RAW image capture
- Scripting possible in python using the picamera2 library
- REST API for remote control
- Web interface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

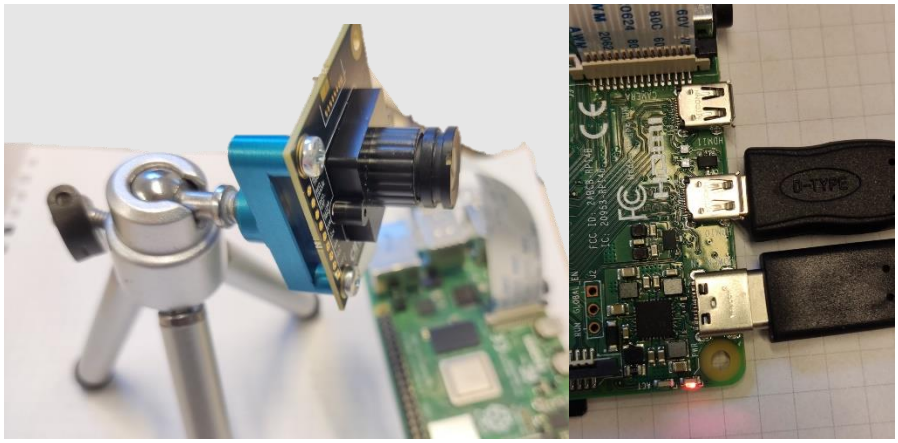
- Raspberry Pi 4B
- Raspberry Pi Power supply (multi-region)
- Mira130, Mira220 or Mira050 sensor board
- MIPI flat cable assembly
- Mini HDMI cable
- M12 lens and screws
- Quick start guide

Figure 2:
Hardware Components Out of the Box



The camera MIPI flat cable assembly is shown in Figure 4.

Figure 3:
Connection of Hardware Components



In addition, the demo requires additional hardware that are not included in the box, as listed below:

- HDMI-compatible monitor and an HDMI cable (connected to HDMI0 port of IO board)
- USB mouse and keyboard
- USB-C to USB-A cable (needed when connecting to a laptop)

3 Getting Started

There are various ways to operate the kit.

- Standalone mode: chapter 3.1
- USB Peripheral mode and Network mode, chapter 3.2

3.1 Standalone Mode

In this mode, the kit will operate standalone as a full-fledged desktop computer.

3.1.1 Preparation of the Hardware

The connection of hardware components is illustrated in Figure 3. There are three major steps.

- Connect MIPI cable with 15-pin 1 mm-pitch metal contacts to the Mira sensor board. The metal contacts of the MIPI cable should face the Mira sensor board PCB.
- Plug in the HDMI cable
- Plug in a USB Keyboard / Mouse
- Plug in the power supply

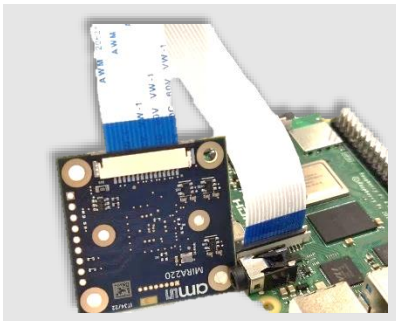
> The kit will boot automatically after power is connected.

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 4:
Camera MIPI Flat Cable Assembly(1)



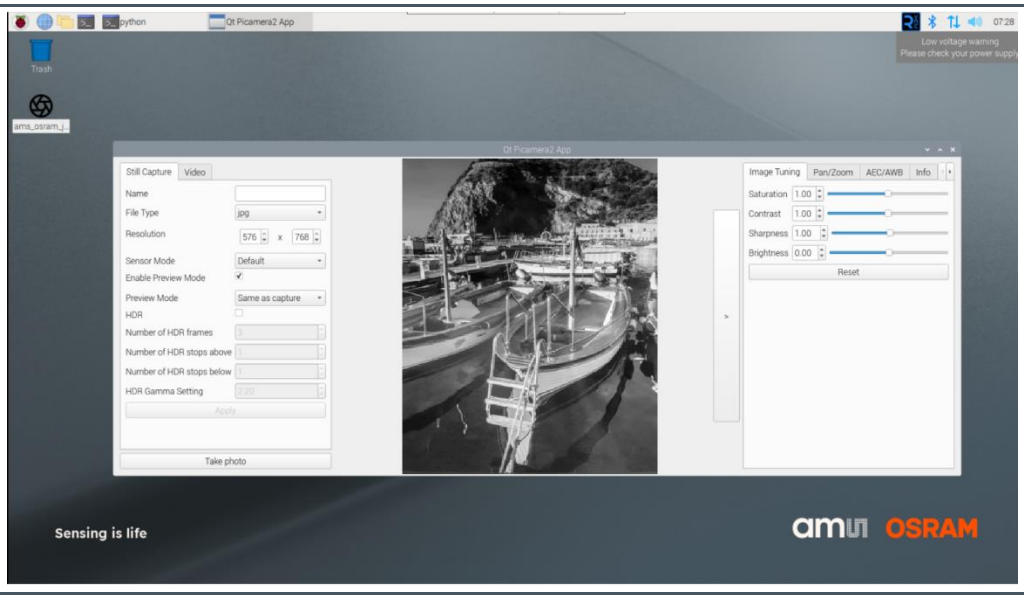
(1) Mind the Orientation of the cable connection for all Mira220/Mira050 reference designs.

3.1.2 Launching the App

After boot, you should end upon the Raspberry Pi desktop.

- To start capturing images, click the ams_osram icon on the desktop.
- For more details on this app, go to 4.1.3 below.

Figure 5:
The Raspberry Pi OS Desktop with the Picamera2 Application



Information

If this step fails, verify the cable and go to chapter 4.1.5.

- Default username: pi
- Default password: pi

3.2 Peripheral Mode and Networked Mode

Besides the standalone mode, there is another mode of operation.

- The Peripheral mode works by connecting the Pi with a USB cable to a Windows Laptop.
- The Networked mode works by connecting the Pi to a local network, and accessing it remotely.

Both modes are very similar in terms of functionality.

3.2.1 Preparation of the Hardware

The connection of hardware components is illustrated in Figure 3. There are three major steps.

- Connect MIPI cable with 15-pin 1 mm-pitch metal contacts to the Mira sensor board. The metal contacts of the MIPI cable should face the Mira sensor board PCB.

In case of the USB Peripheral mode:

- 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 6.

In case of the Networked mode:

- Connect power supply
- Connect Ethernet cable

Information

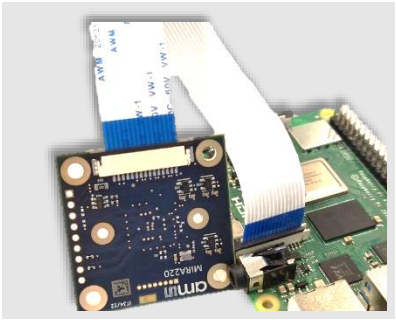
The kit will boot automatically after power is connected. You should see some activity on the red and green LEDs.

Figure 6:
Connecting the Kit in USB Peripheral Mode

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 Mira220/Mira050 Reference Designs



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

3.2.2 Installing the Required Drivers on a Windows Laptop

To make use of the peripheral mode, some drivers need to be in place.

Get the driver from ams webpage.

- Download the driver from the ams website:
  Mira-Eval-Kit

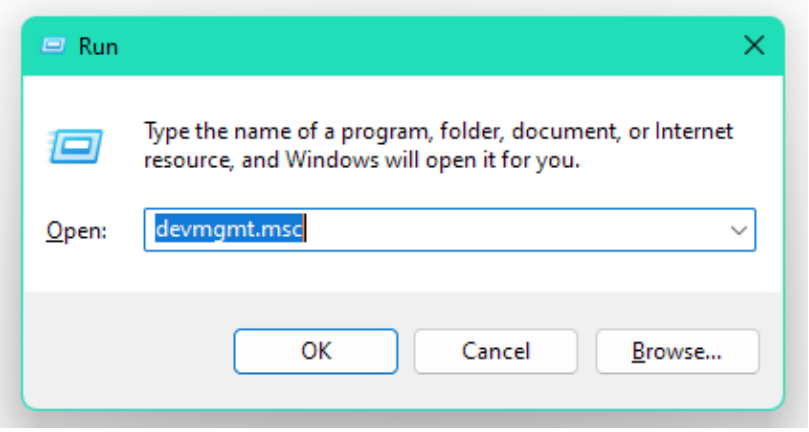
Figure 8:
Get the Driver from ams Webpage

Driver	Mira_Driver_RNDIS_v1-0-0.zip RNDIS driver for raspberry pi eval kit for mira	EN

Enable windows RNDIS driver.

- Press WIN+R and type devmgmt.msc

Figure 9:
Windows Run Command



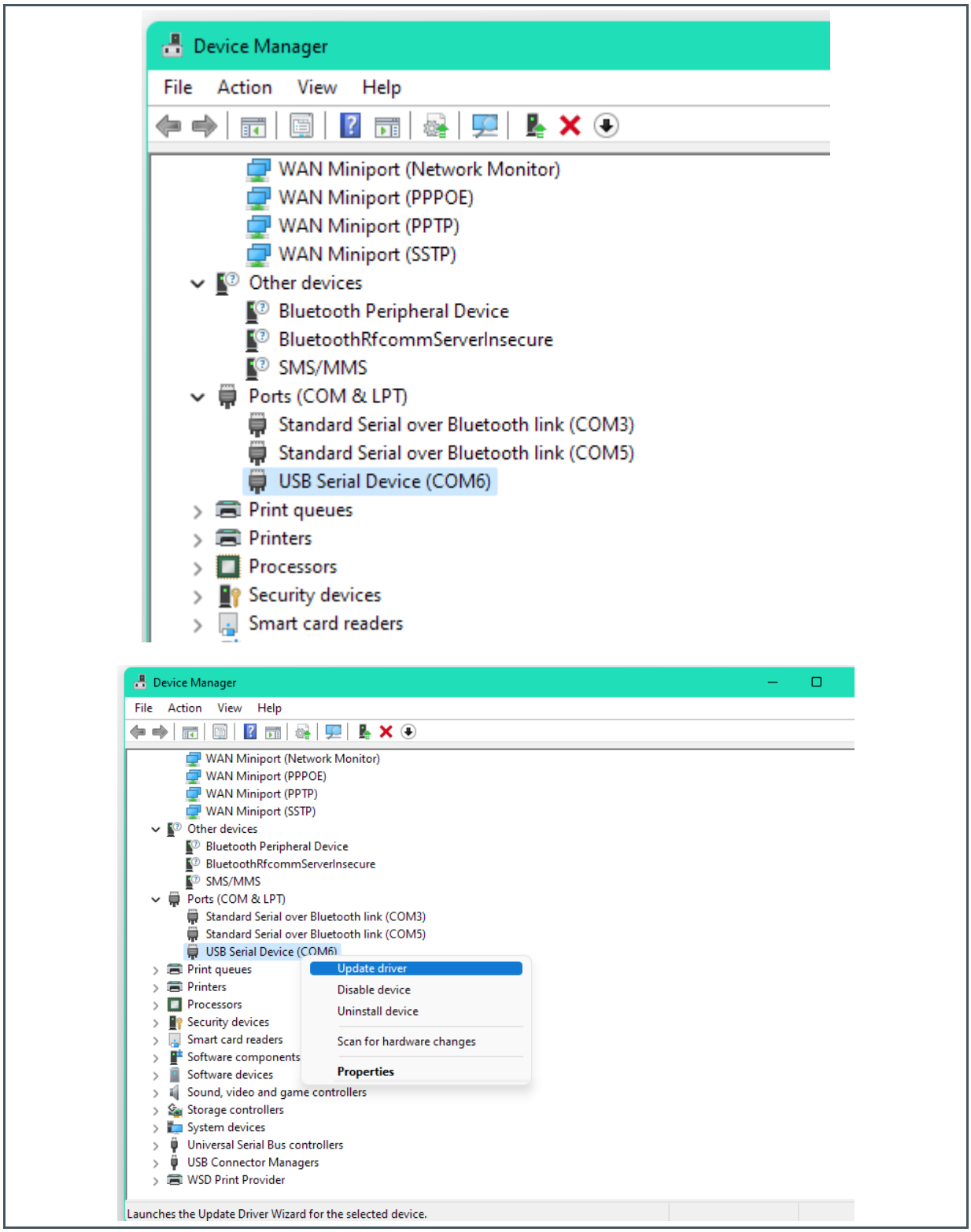
- Press CTRL + SHIFT + ENTER, then - enter your admin password.
- Find the new USB device under Ports.

Information

Note: If the Raspberry Pi does not show up under Ports (COM & LPT), check the USB-C cable if it can pass data. Some USB-C cables are power-only.

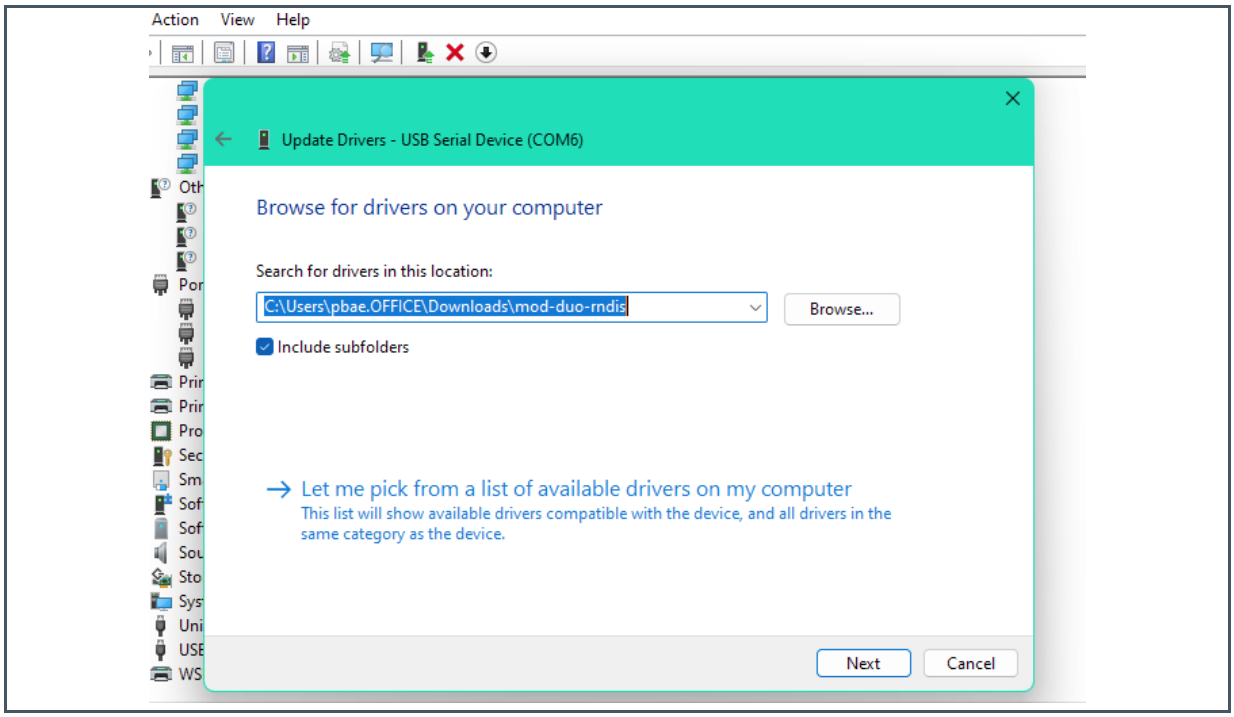
- Right click and update driver.

Figure 10:
Device Manager



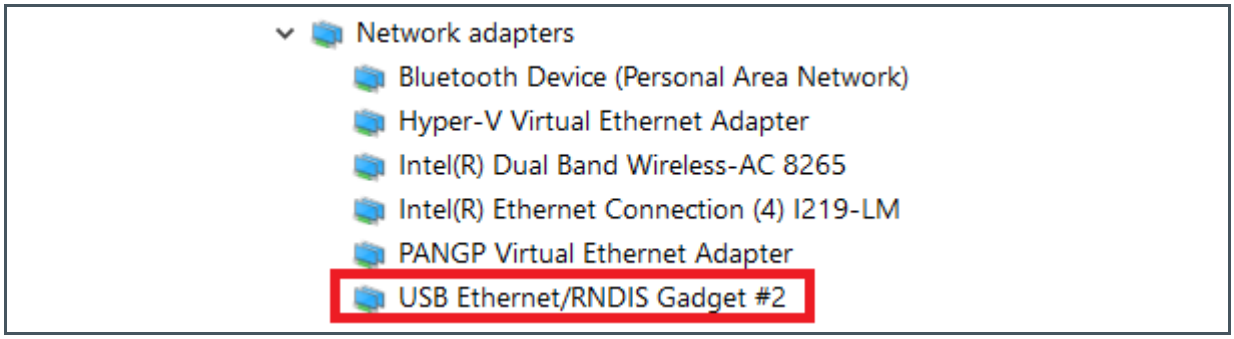
› Browse to the proper location where you downloaded the driver.

Figure 11:
Location of USB Drivers



After installing the driver, the Raspberry Pi will disappear from Ports, and will show up under Network adapters.

Figure 12:
Network Adapters



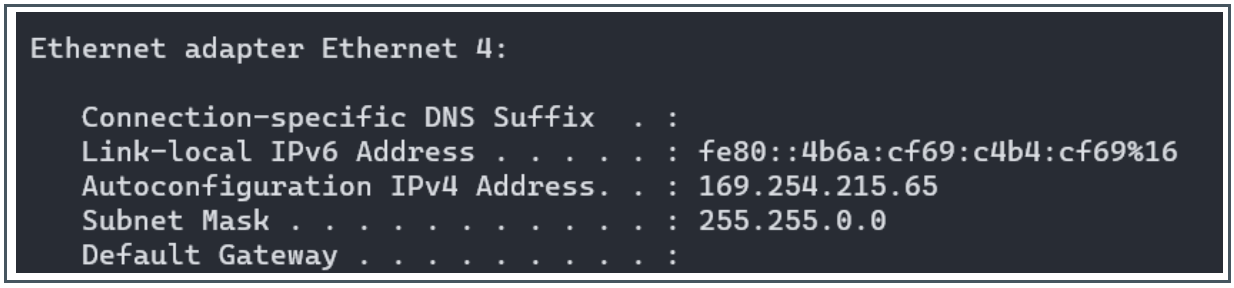
Information

Note: The driver needs to be installed for each USB port being used to connect to Raspberry Pi. Each addition of USB port with this driver installed has #2, #3 etc. appended to the name.

To verify that the driver works,

- Open a terminal and press ipconfig. **An adapter with address 169.254.xxx.xxx should be shown.**

Figure 13:
Terminal Window

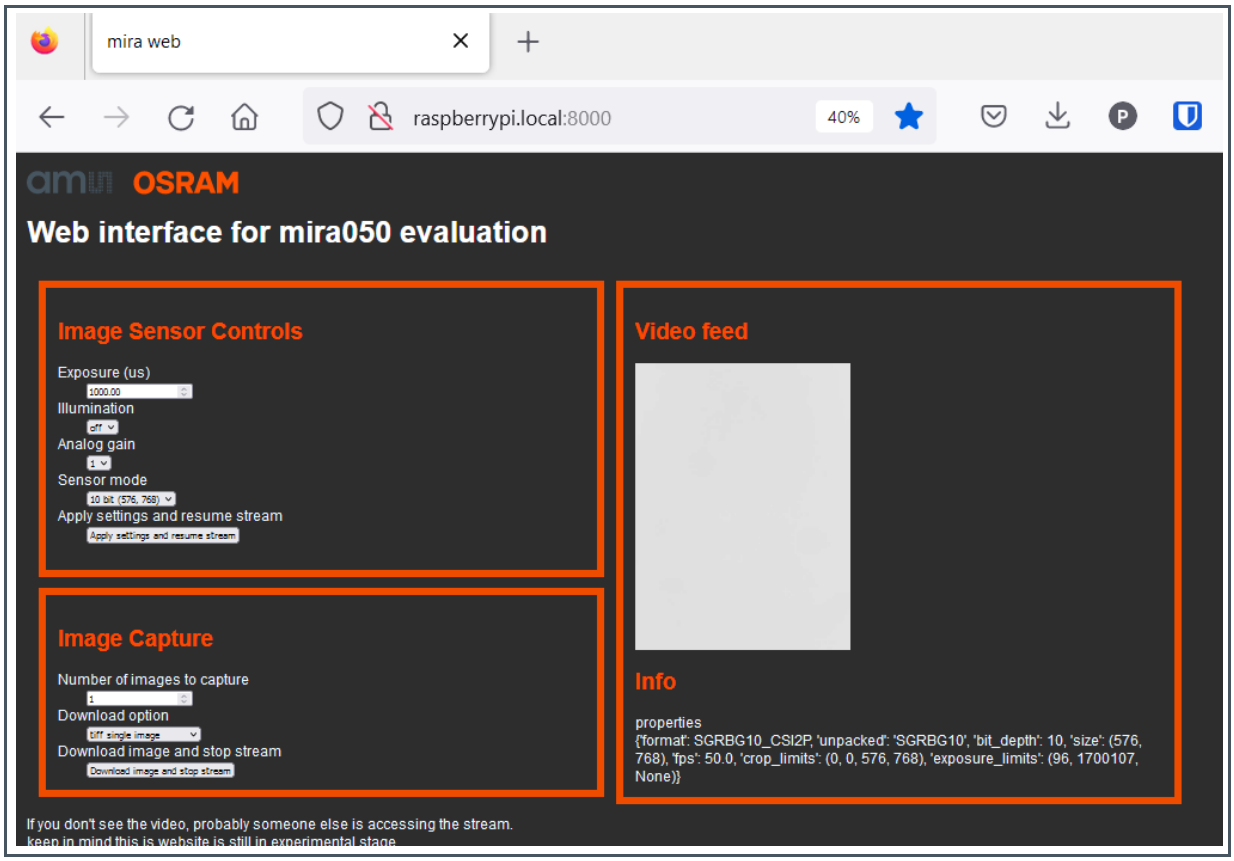


3.2.3 Web Viewer

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

- Open a web browser and visit:
- In USB mode: `raspberrypi.local:8000`
- In networked mode: *ipaddress_of_raspberrypi*:8000

Figure 14:
A Preview of the Web Interface



3.2.4 API for Remote Capturing and Scripting

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

By sending HTTP requests, we can control the camera and receive images.

For an example, see:

```
~/ams_rpi_software/common/picamera2-flask/windows/api_usage_example.py
```

4 A Brief 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 in two parts. The kernel space and the user space. The drivers, interacting with hardware, will 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 can 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 15:
Camera Framework Architecture

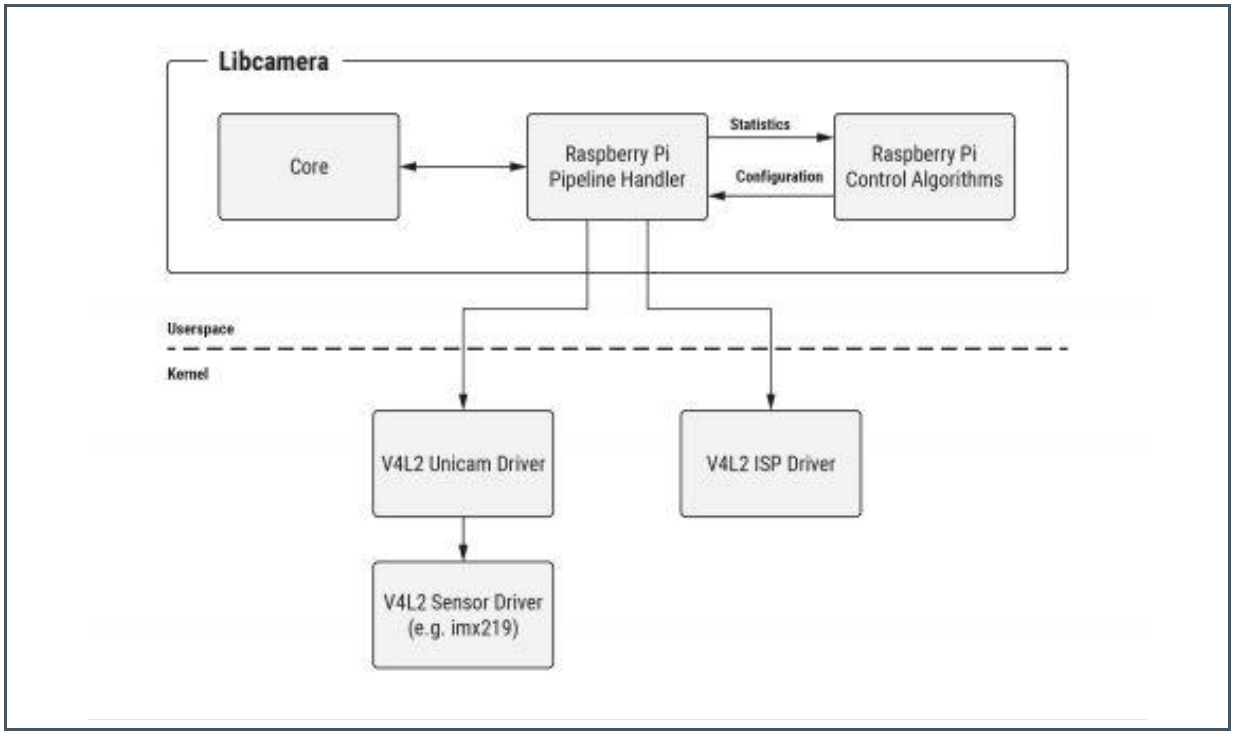
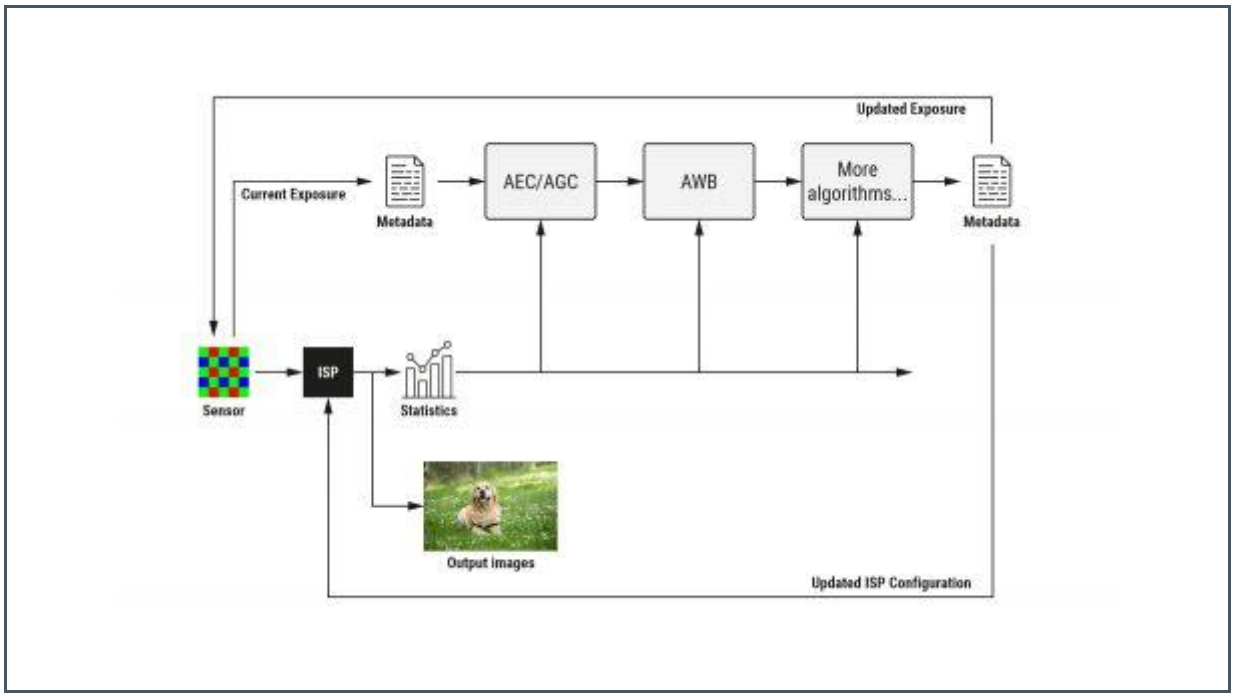


Figure 16:
Control Algorithm



4.1.1 Capturing Images with Libcamera

Information

Before running any commands, make sure the **web service** is disabled. See chapter 4.1.4.

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
```

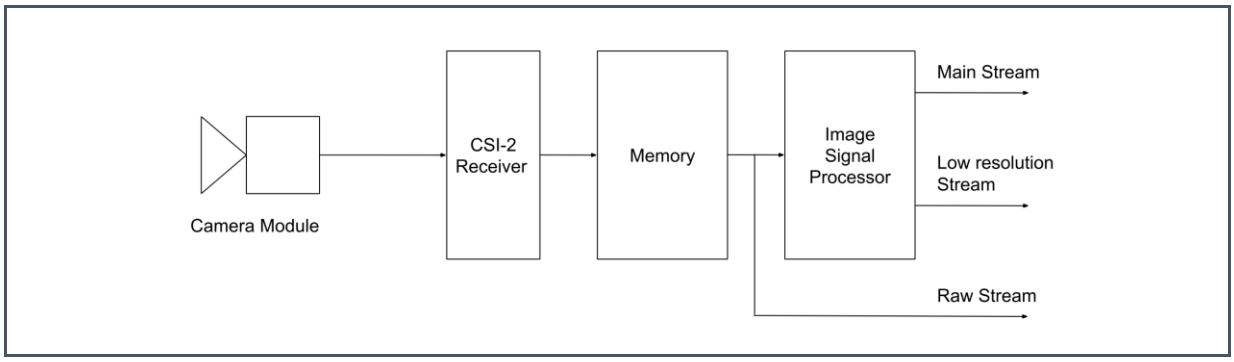
Capture video

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 17:
Camera Pipeline



4.1.2 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 18:
Examples of PiCamera2 Preinstalled Applications

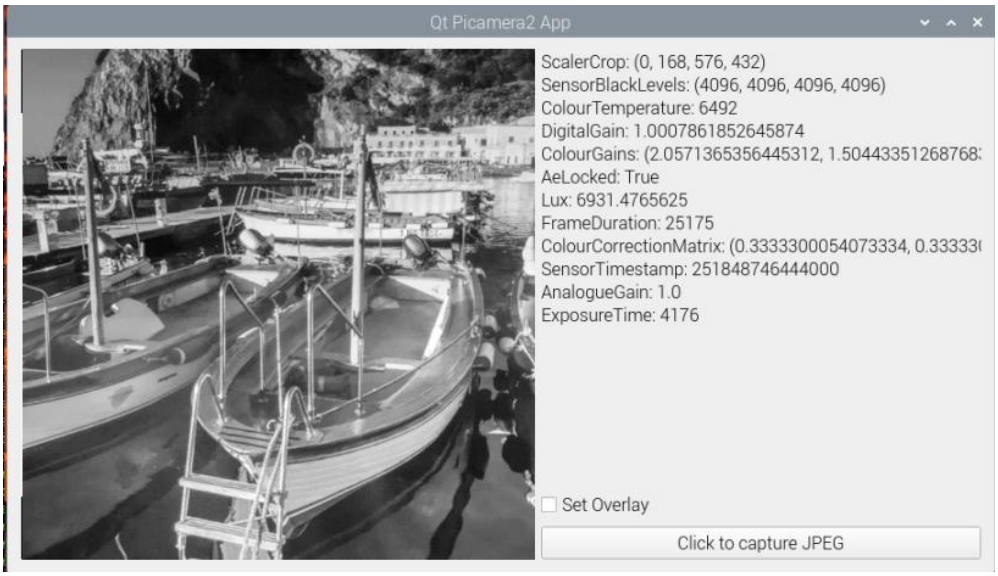
```
pi@raspberrypi:~/ams_rpi_software/picamera2/examples $ ls
audio_video_capture.py          controls_2.py                overlay_qt.py
capture_average.py              controls_3.py                pick_mode.py
capture_circular_nooutput.py    controls.py                  preview_drm.py
capture_circular.py             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 19:
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 20:
PiCamera2 App



4.1.3 Full-Featured PiCamera2 GUI

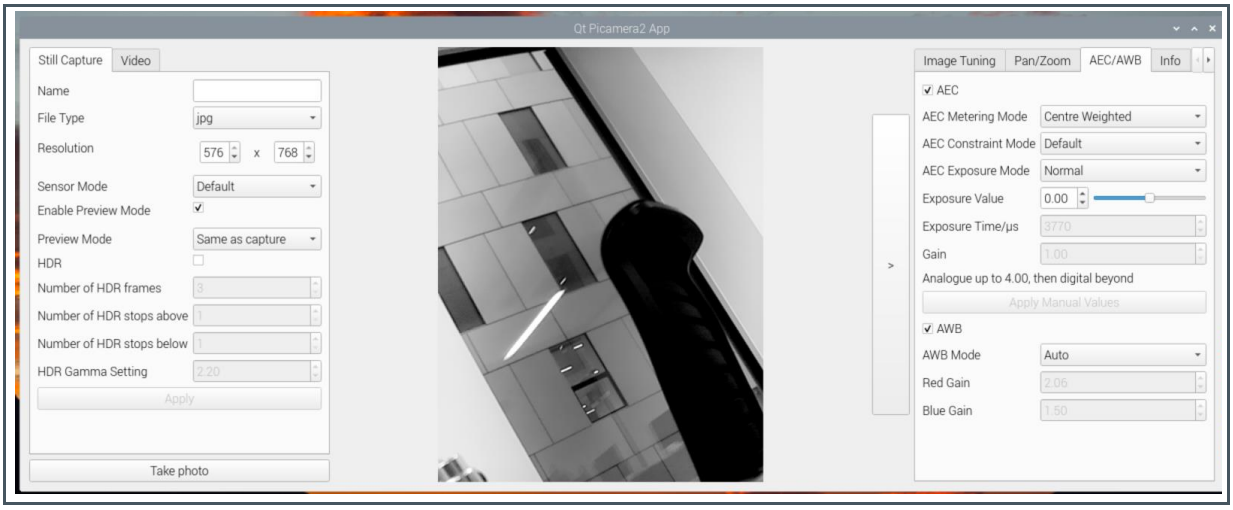
Another useful application is provided. There is a shortcut on the desktop to launch it. It allows you to capture images/videos and adjust gain/exposure and tune the ISP.

Information

1. Keep in mind, by default, the auto exposure and gain is enabled. Disable this and manually configure the exposure to have fine control.
2. 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 21:
App_full.py

Figure 22:
PiCamera2 GUI



4.1.4 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 3.2.3.

4.1.5 Device Tree Overlay and Drivers

Information

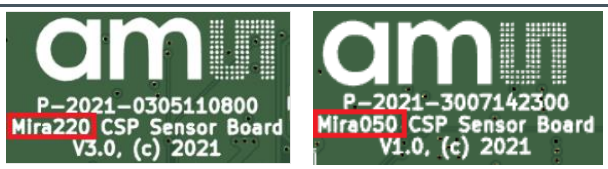
If the camera works, you can skip this chapter.

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

- MIPI cable connection is incorrect. Please double check the metal contacts are facing the correct side, and shown in Section 3.1.1.
- 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 and Mira050. Each sensor board has its type printed on the sensor PCB, as shown in Figure 23.

Figure 23:
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 sensor is connected, the line should be the following (all letters are in lower case).

```
dtoverlay=mira220
```

If a Mira050 sensor is connected, the line should be the following (all letters are in lower case).

```
dtoverlay=mira050
```

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 effects. Use either the desktop GUI or the command below to reboot the Raspberry Pi.

```
sudo reboot
```

5 Dual Cameras on Compute Module 4

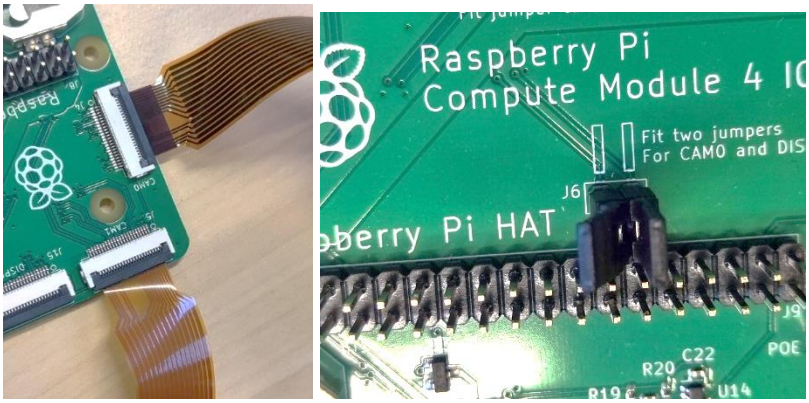
The regular Raspberry Pi only supports 1 camera. The Pi Compute Module 4 supports 2 cameras.

What you need:

- Pi compute module 4
- 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)(2)]



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

Instructions:

1. On the Compute Module, run `sudo raspi-config` and enable the camera.
2. 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. Power the Compute Module down.
5. Connect both camera cables to cam0 and cam1 on the csi port.
6. Connect jumpers on J6.
7. Power up.

Using commands, you can test both cameras in parallel.

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.

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 (https://www.raspberrypi.com/documentation/)

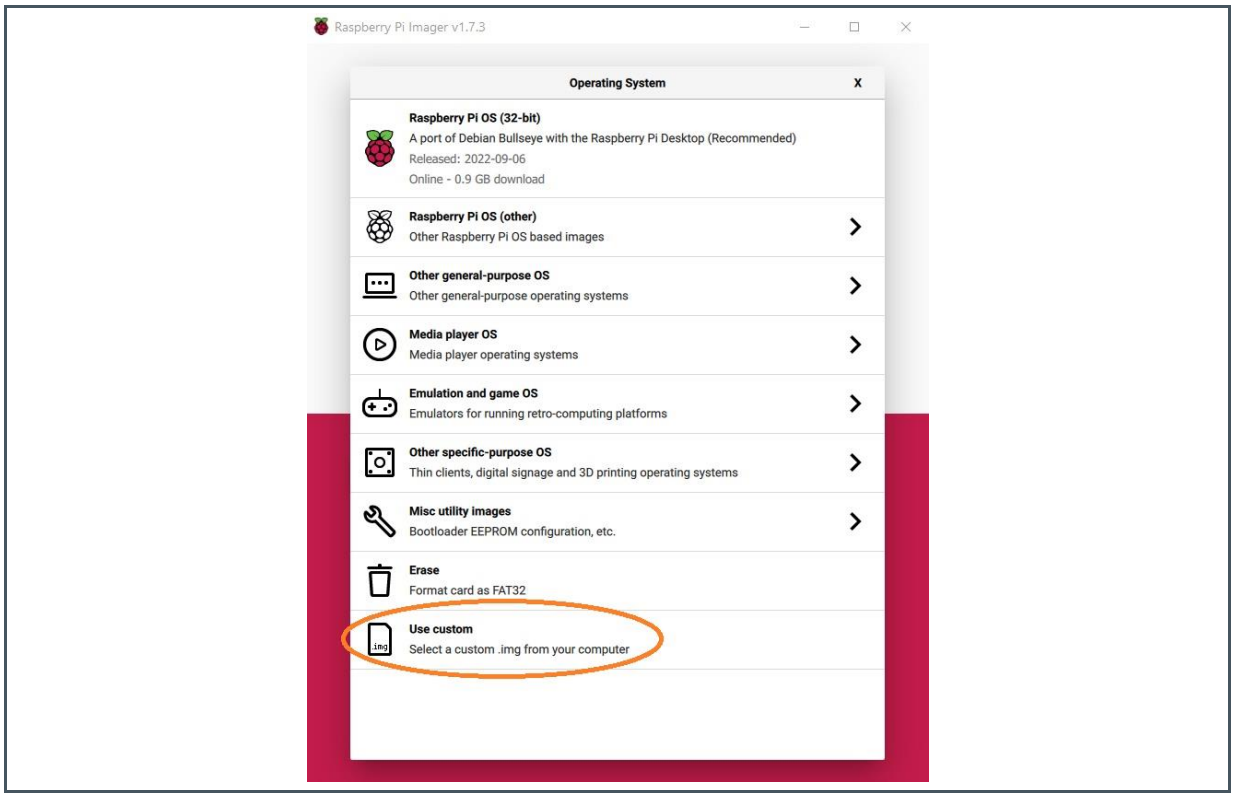
› Obtain a customized Raspberry Pi OS image that has all the required drivers built in. Please contact your local ams OSRAM sales or FAE.

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

› Download and install Raspberry Pi Imager (https://www.raspberrypi.com/software/) to write the OS image file to the Raspberry Pi. Other tools such as Balena etcher or win32diskimager also work.

The rpi-imager by default is installed to C:\Program Files (x86)\Raspberry Pi Imager\rpi-imager.exe. After launching the rpi-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.

Figure 26:
CHOOSE OS Dropdown Menu(1)



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

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.

› After the Raspberry Pi Imager finishes writing the OS image, eject the storage device from the host computer.

7 Revision Information

Definitions

Draft / Preliminary:
The draft / preliminary status of a document indicates that the content is still under internal review and subject to change without notice. ams-OSRAM AG does not give any warranties as to the accuracy or completeness of information included in a draft / preliminary version of a document and shall have no liability for the consequences of use of such information.

Changes from previous version to current revision v1-00	Page
Initial production version	

- 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
ams-OSRAM AG
Tobelbader Strasse 30
8141 Premstaetten
Austria, Europe
Tel: +43 (0) 3136 500 0

Please visit our website at ams-osram.com
For information about our products go to Products
For technical support use our Technical Support Form
For feedback about this document use Document Feedback
For sales offices and branches go to Sales Offices / Branches
For distributors and sales representatives go to Channel Partners