Quick Start Guide

Document Number

Mira Evaluation Kit

For Mira image sensors

Mipi image sensors on raspberry pi

v0-02 • 2023-Jun-26

Content Guide

1 Introduction

1.1 Description

Mira Evaluation Kit is a platform for evaluating the Mira product family of image sensor. This platform is based on Raspberry Pi 4B, although other variants are also compatible.. 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 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)

Formatted: Not Highlight

1.2 Key Features

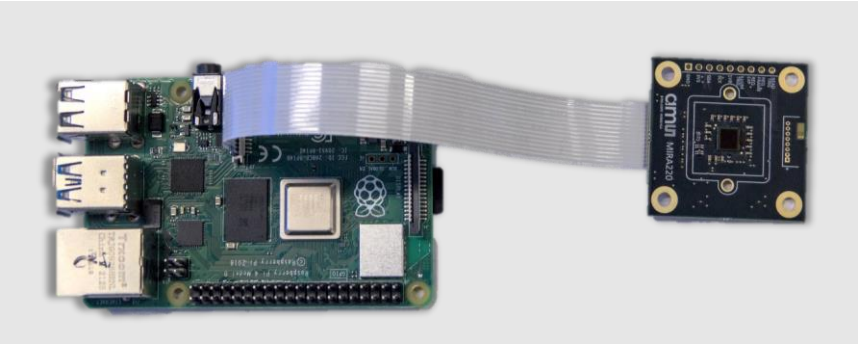
- Mira220 and Mira050 image sensors support
- Fully configured ISP for best image quality
- Software based on libcamera, an open source camera stack and framework for Linux, Android, and ChromeOS
- RAW image capture
- Scripting possible in python using the picamera2 library.

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.
- Mira220 or Mira050 sensor board.
- MIPI flat cable assembly.
- Mini HDMI cable
- Tripod, tripod holder, screws
- M12 lens and screws

Figure 2: Hardware components out of the box.



The camera MIPI flat cable assembly is shown in Figure 3.

3 Connect Hardware

The connection of hardware components is illustrated in Figure 4. There are three major steps.

- Connect MIPI cable with 15-pin 1mm-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

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 3: Orientation of the cable connection for all Mira220/Mira050 ref designs

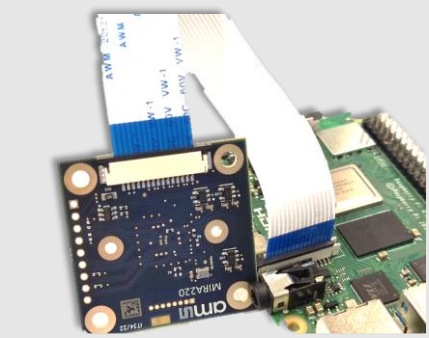
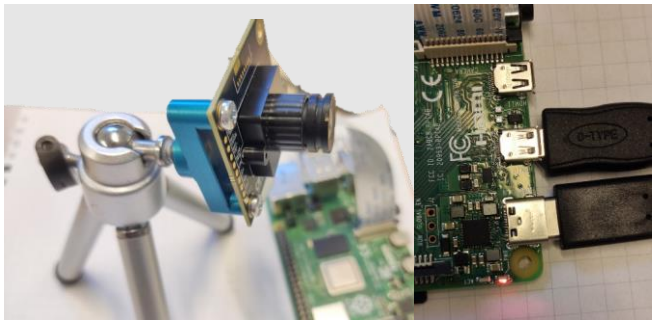


Figure 4: Connection of hardware components



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

- An HDMI-compatible monitor and an HDMI cable (connected to HDMI0 port of IO board)
- A USB mouse and keyboard

4 Software

> If you want to quickly capture an image, skip to chapter 4.5.1.

4.1 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 a 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 5:
Camera framework architecture

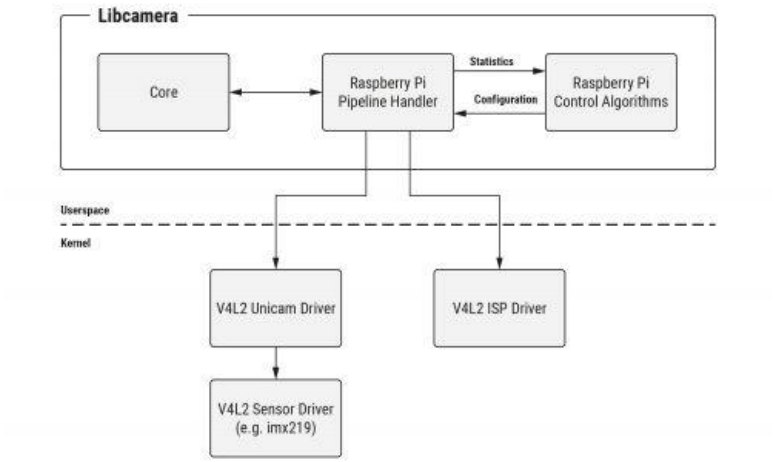
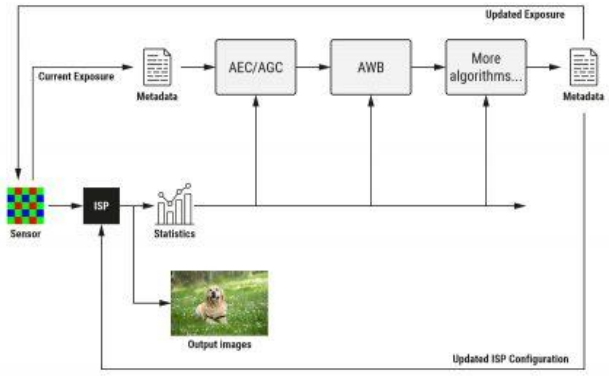


Figure 6:
Control Algorithm



4.2 Log in for the first time

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

- An HDMI-compatible monitor and an HDMI cable (connected to HDMI1 port of IO board).

A USB mouse and keyboard.

- Connect an HDMI cable to the HDMI port of the IO board; connect a USB mouse and keyboard to the USB ports of the IO board. Reboot the IO board by unplugging its power supply for a few seconds.

After rebooting, the monitor shows up a “log in” screen. Use the following credential to log in.

- Default username: pi
- Default password: pi

After log in, users can decide to proceed with subsequent steps in two different modes.

- **Standalone Mode**: In this mode, the Raspberry Pi is not connected to a network. Users need to rely on a monitor, mouse, and keyboard to perform subsequent steps.
- **Networked Mode**: In this mode, the Raspberry Pi is connected to a network. User can set up a remote desktop client, such as VNC Viewer, to access the Raspberry Pi from a host computer. In this mode, monitor, mouse, or keyboard are not required.

Information

The Network Mode utilizes the built-in VNC server in the Raspberry Pi OS. Users need to install a VNC client on a host computer. For host computers with Windows OS, one popular VNC client is the VNC Viewer available at https://www.realvnc.com/en/connect/download/viewer/.

4.3 Selecting the Sensor

Before starting the demo, it is recommended to first check whether the camera is working as expected. To do so, open a command line window and issue the command below to start a GUI for basic video streaming. The GUI window automatically closes after 5 seconds, or can be closed earlier by pressing “Ctrl+c” in the command line window.

```
libcamera-hello
```

If the above step fails, there are two common causes, which can be checked in the following steps.

- The camera
  MIPI cable connection is incorrect. Please double check the metal contacts are facing the correct side, and shown in Section 3.
- A mismatch between sensor hardware type and driver configuration. Details are provided next.

The Raspberry Pi platform currently supports two types of Mira sensors, mira220 and mira050. Each sensor board has its type printed on the PCB, as shown in Figure 7.

Formatted: amsBodyList1

Deleted: ¶
¶

Figure 7:
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” 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
```

4.4 Capturing images with libcamera

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 provide 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
```

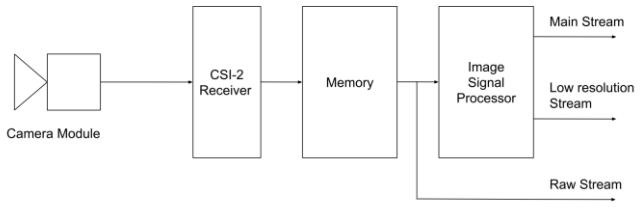
Capture video

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 8:
Camera pipeline

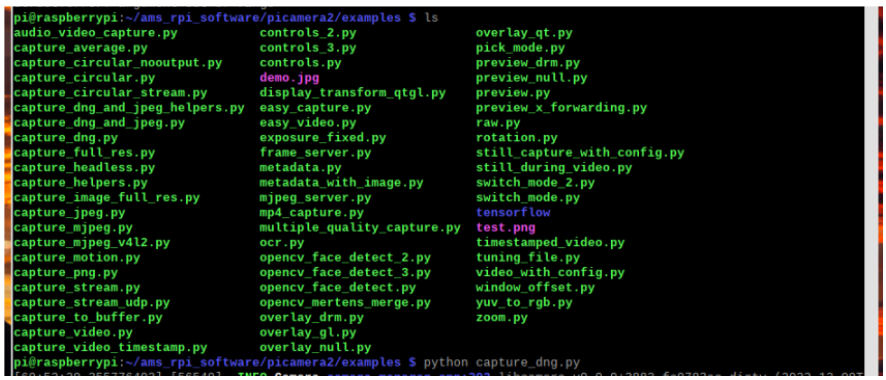


4.5 PiCamera2

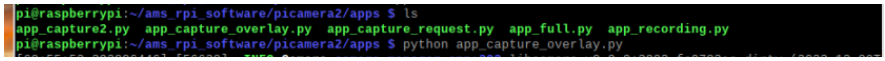
There is a project called PiCamera2, which tries to create python bindings for the Libcamera framework.

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.



Also a few graphical examples come preinstalled such as:

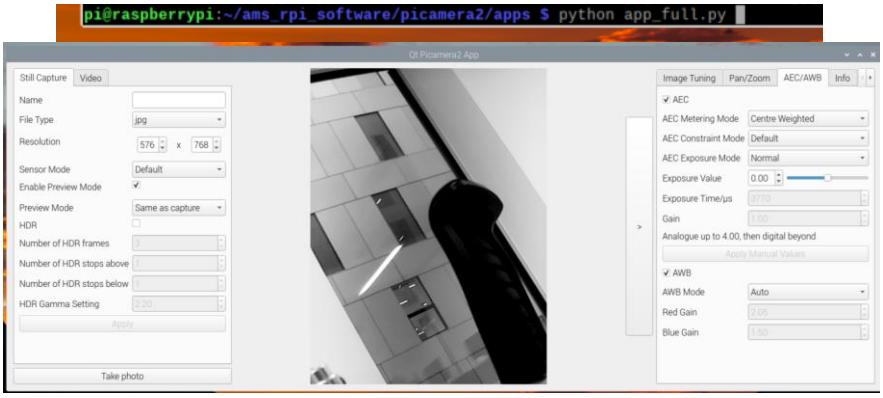


4.5.1 Full-featured picamera2 GUI.

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.

Keep in mind, by default, the auto exposure and gain is enabled. Disable this and manually configure the exposure to have fine control.

Figure 9:
App_full.py



5 Dual Cameras on Compute Module 4

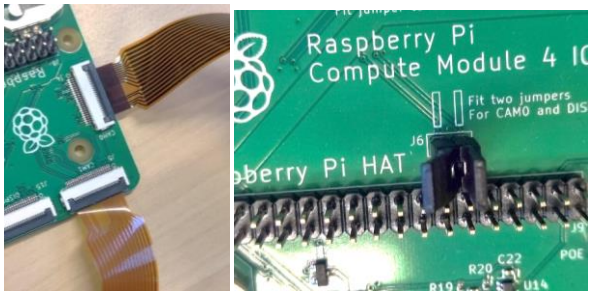
The regular raspberry pi only supports 1 camera. The Pi Compute Module 4 supports 2 cameras.

What you need:

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

Figure 10:
Dual cameras on the pi compute module

Figure 11:
Dual cameras: connect the flat cable to Cam0 and Cam1. 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.

Deleted: s

6.1 Flash OS image on the SD card

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

- A host computer with microSD 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:

- 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.

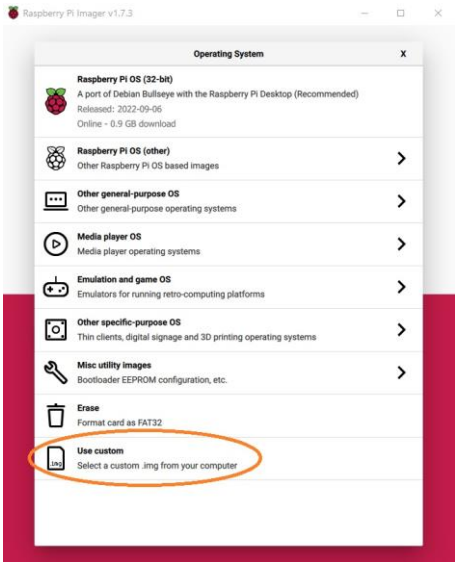
Once the OS image file is downloaded, the next step is to write the OS image file onto the sd card.

› 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 12: In the “CHOOSE OS” dropdown menu, select “Use custom” option, and

then select the OS image file that has been downloaded. Then select the OS image file that has been downloaded in the previous step.

Figure 12: 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.