

Setting up the Board and Application Deployment

Introduction

This document shows how to set up the board and how to run Prophesee applications.

Warning

Prophesee Linux image does not start any windowing system. Hence, using the universal asynchronous receiver-transmitter (UART) interface is mandatory to launch the application.

Setting up the SD Card Image

Download [Prophesee Kria™ Starter Kit Linux image](#) and flash it into a microSD card (minimum 16 GB).

Note

If you don't have access to this Knowledge Center page, contact support@prophesee.ai asking for a KC account and a specific access to the Kria Starter Kit page.

Please follow AMD guide "[Setting up the SD Card Image](#)" to flash it with Prophesee's image. Once done, continue with this tutorial.

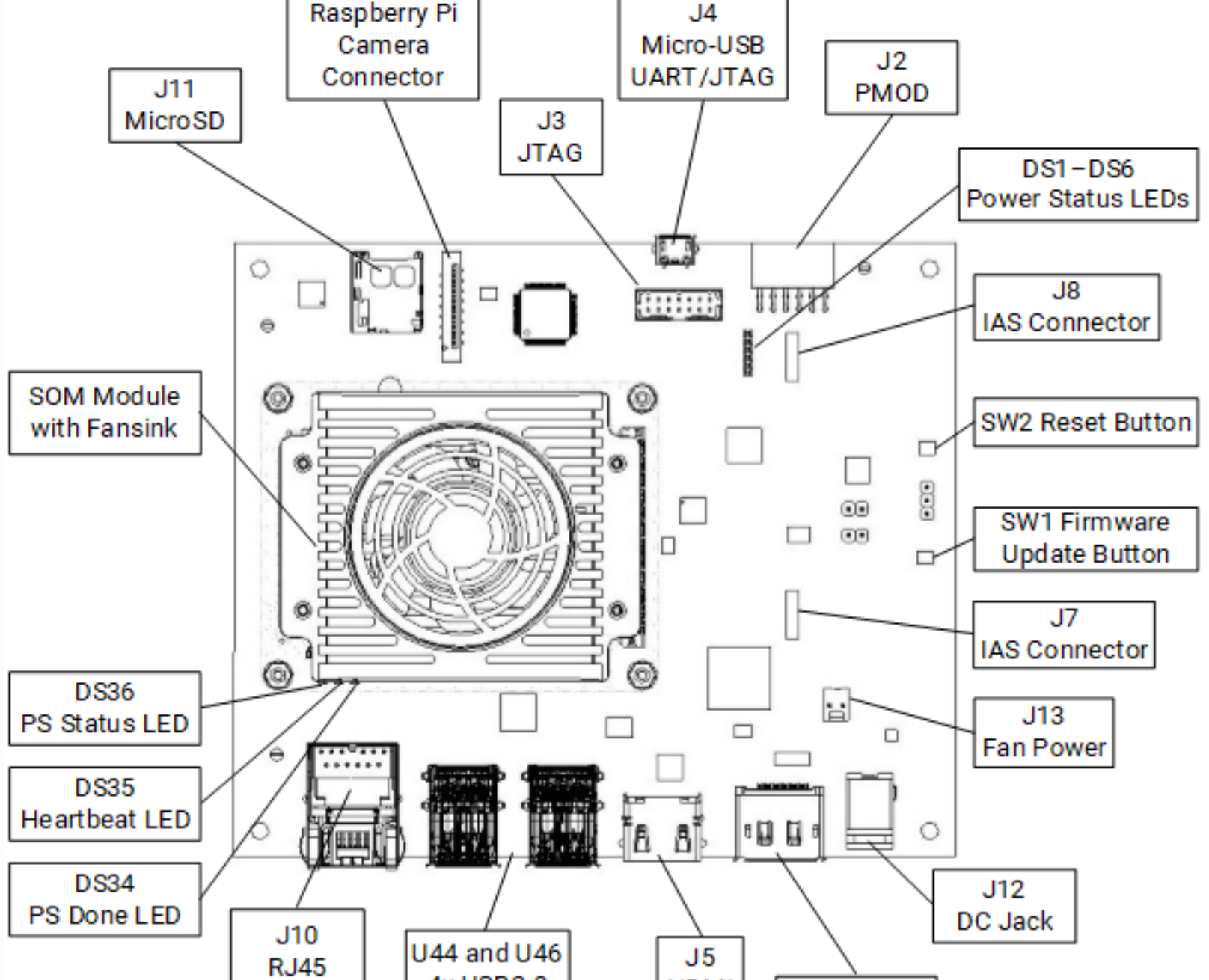
Hardware Setup

Let's see how to connect all the hardware interfaces required by Prophesee applications.

Tip

Before proceeding, we advise you to check [AMD Kria KV260 Getting Started](#). Pay special attention to those sections:

- [Connecting Everything](#)
- [Booting your Starter Kit](#)



Active Markers Application

Introduction

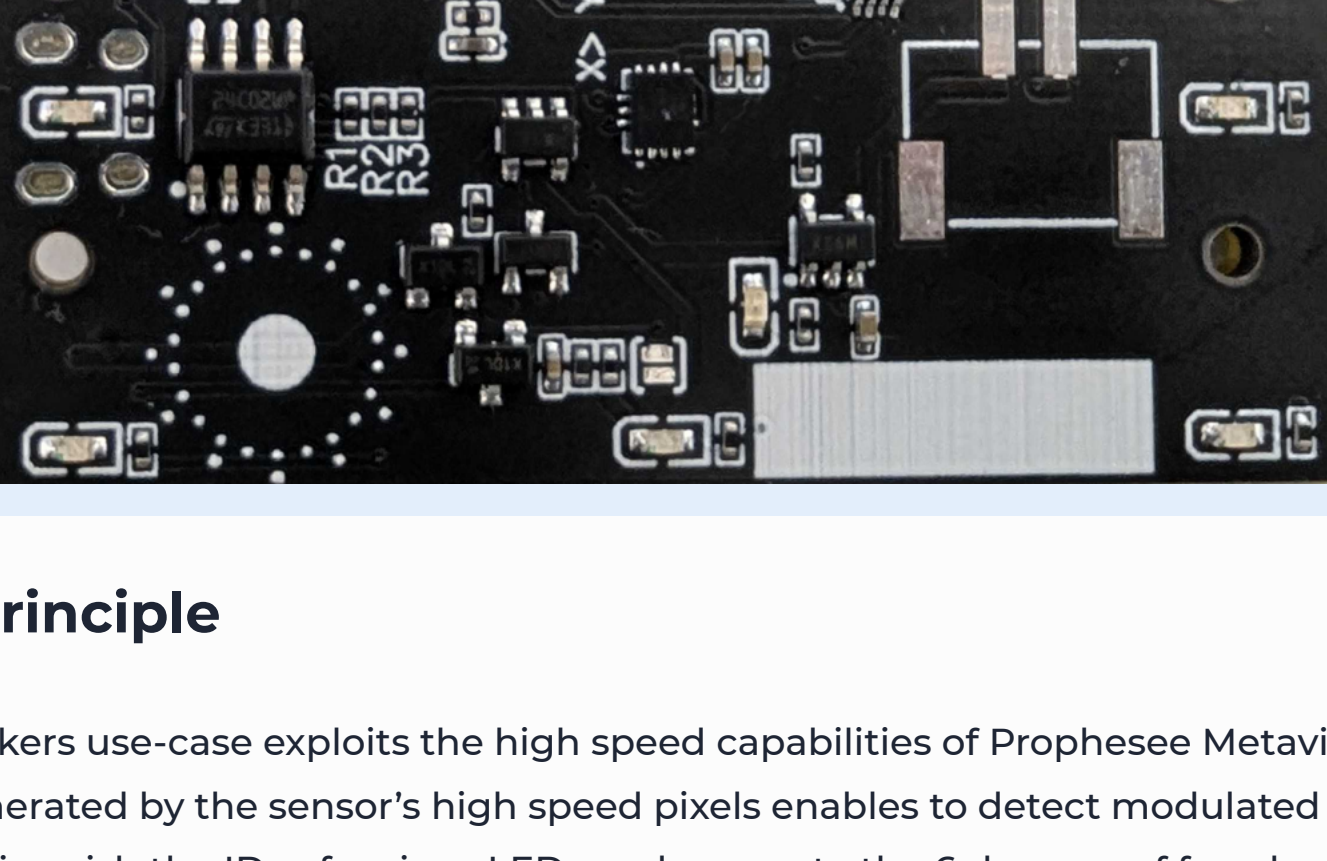
This document shows how to start the Active Markers demo provided with the starter kit. The content is an adaptation of the [existing documentation available](#)¹ in the Metavision SDK online documentation.

Required Material

The starter kit includes an Active Marker LED board, powered via USB and pre-programmed with the necessary firmware for this application.

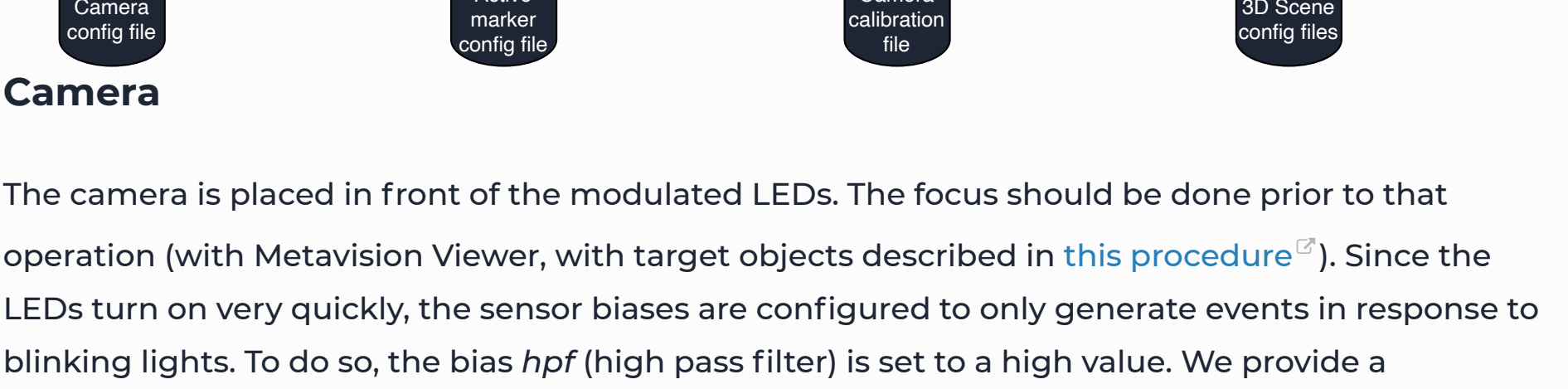
Note

If the LED board does not turn on when plugged into the USB port, try using one of the two black buttons on the backside to turn it on.



General Principle

The Active Markers use-case exploits the high speed capabilities of Prophesee Metavision sensors. The events generated by the sensor's high speed pixels enables to detect modulated light at high frequency, distinguish the IDs of various LEDs and compute the 6 degrees of freedom pose of the object to which the LEDs are attached.



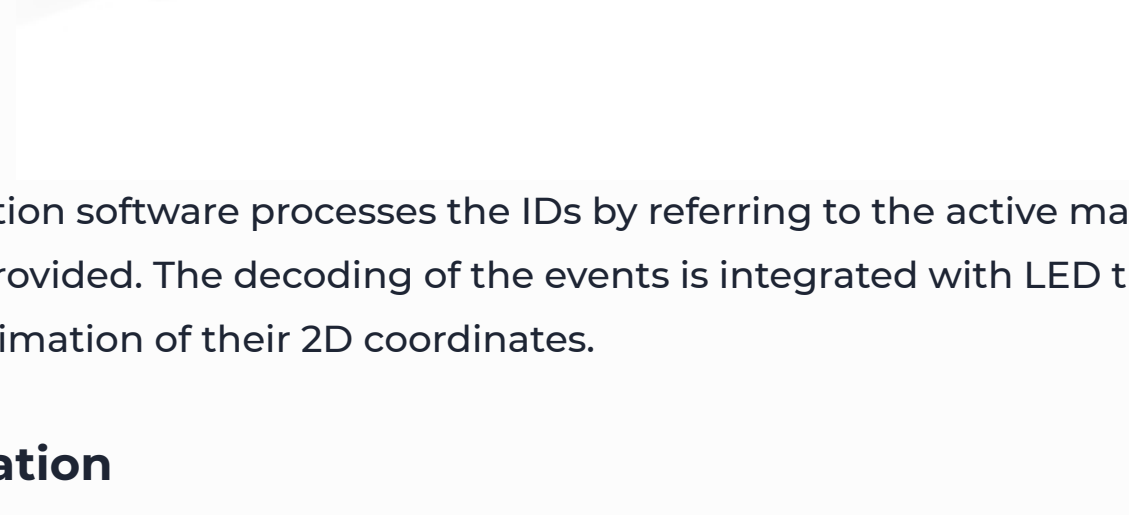
Camera

The camera is placed in front of the modulated LEDs. The focus should be done prior to that operation (with Metavision Viewer, with target objects described in [this procedure](#)²). Since the LEDs turn on very quickly, the sensor biases are configured to only generate events in response to blinking lights. To do so, the bias *hpf* (high pass filter) is set to a high value. We provide a configuration file that allows the camera to be set to the appropriate operating point.

Demodulation & Tracking

The LEDs on the marker board are programmed to emit unique time patterns, with each LED transmitting a time pattern that encodes its specific ID. More details on the modulation itself can be found in [this section of the SDK documentation](#)³.

By default, the LEDs on the supplied active marker board emit the IDs 20, 21, 22, 23, 24, 25, 26, 27 at the positions shown in the following figure:



The demodulation software processes the IDs by referring to the active marker configuration file, which is also provided. The decoding of the events is integrated with LED tracking, allowing for a continuous estimation of their 2D coordinates.

Pose Estimation

With the 2D position of the markers, the knowledge of their physical position on the board and the camera calibration, we compute the absolute pose of the marker with respect to the camera. This is done using the Embedded application only.

This 6 degrees of freedom pose is the solution to the [Perspective-n-Point problem](#)⁴ defined with the LEDs positions.

Warning

This stage requires a calibrated camera, with estimated intrinsics and distortion model. The calibration process is detailed in the [intrinsics calibration section of the SDK documentation](#)⁵.

This is currently an unsupported feature with the Kria Starter Kit and will only be made available in a future release. We provide a *canonical* calibration file that gives a rough calibration model estimate for the 5mm lens provided with the starter kit.

The purpose is to enable our customer to have an end-to-end experience all the way up to the visualization of 3D pose, but this does affect the precision of the pose estimation.

3D Rendering (Full Demo)

From the 6 DOF pose we render the marker in a virtual scene for illustration purposes. This last stage is meant to provide an example of a usage of the pose estimate.

In this instance, it's for rendering purposes, but the pose could also be used as an input device in a human-machine interface.

Note

The 3D rendering on the Kria board is a basic, non-real time implementation. It can be optimized to run directly on the Kria board, but for the time being the rendering is done on a laptop connected to it.

Running the Embedded Application Only

First, let's run the embedded-only application (`metavision_embedded_active_marker_3d_tracking`) that runs the Active Markers 2D tracking.

Before launching the application, the FPGA and camera must be initialized. Connect to the Kria board (via minicom or SSH) and launch the following commands:

```
cd /opt/metavision/embedded_active_marker_3d_tracking/embedded_active_marker
chmod +x init_camera_<sensor>.sh
./init_camera_<sensor>.sh
```

Now we recommend to do the camera focus by launching the following command:

- If you use UART connection via minicom:

```
killall -9 Xorg && sleep 1
Xorg &
DISPLAY=:0.0 VAL2_HEAP=reserved VAL2_SENSOR_PATH=/dev/v4l-subdev3 metavision_viewer
```
- If you use ethernet connection via SSH:

```
VAL2_HEAP=reserved VAL2_SENSOR_PATH=/dev/v4l-subdev3 metavision_viewer
```

This should show the Metavision Viewer window on the display connected to the board. Adjust the optic until the image is as sharp as possible.

Now, launch the `metavision_embedded_active_marker_3d_tracking` application using the `launch_<sensor>.sh` script:

```
cd /opt/metavision/embedded_active_marker_3d_tracking/embedded_active_marker
chmod +x launch_<sensor>.sh
./launch_<sensor>.sh
```

If you move the LED board in front of the camera, you should see that the LEDs are displayed and detected.

Running the Full Demo

The full demo allows to have the 3D rendering of the tracking on the host PC:



First, download the source code archive available at the [following page](#)⁶.

Note

If you don't have access to this Knowledge Center page, contact support@prophesee.ai⁷ asking for a KC account and a specific access to the Kria Starter Kit page.

The archive contains the following files:

```
├─ metavision_active_marker_3d_tracking-openeb_v5.0.0
│   ├── MakeLists.txt
│   ├── metavision_active_marker_3d_tracking
│   │   ├── ...
│   │   └─ metavision_embedded_active_marker_3d_tracking
│   │       ├── ...
│   │       ├── metavision_extract
│   │       │   ├── ...
│   │       │   ├── plugins.cfg.in
│   │       │   ├── README.md
│   │       │   └─ resources.cfg.in
│   │       └─ utils
│   │           ├── config_files
│   │           │   ├── active_marker_config.json
│   │           │   ├── camera_calibration_genx320.json
│   │           │   ├── camera_calibration_lm636.json
│   │           │   ├── camera_config_genx320.json
│   │           │   └─ camera_config_lm636.json
│   │           ├── scene
│   │           │   ├── active_marker_target.mesh
│   │           │   ├── decor.mesh
│   │           │   ├── industrial_scene.material
│   │           │   └─ industrial_scene.scene
│   │           ├── scripts
│   │           │   ├── host
│   │           │   └─ target
```

Some configurations files are delivered in the folder `utils` :

Camera config files: camera_config_<sensor>.json	Contains the description of the camera configuration. For this application a specific set of biases is used.
Active marker config file: active_marker_config.json	Contains the description of the marker used. The IDs of LEDs as well as their physical location on the board.
Camera calibration files: camera_calibration_<sensor>.json	Contains the projection model of the camera used. Comes from a calibration that estimates intrinsics matrix and distortion coefficients.
3D Scene files: "scene" folder	Contains the 3D content of an industrial scene in order to visualize the camera's poses in a virtual environment.

To run the full demo on Ubuntu 22, follow those steps:

- Install dependencies:

```
sudo apt update
sudo apt install -y software-properties-common gpg ca-certificates curl unzip cmake build-ess
sudo apt install -y libboost-program-options-dev libeigen3-dev libceres-dev
sudo apt install -y libgrog-1.12-dev libimgui-dev libfreetype-dev

# Install Sophus
curl -L "https://github.com/strasdat/Sophus/archive/1.22.10.zip" --output /tmp/Sophus.zip
unzip /tmp/Sophus.zip -d /tmp/
cmake -S /tmp/Sophus-1.22.10 -B /tmp/Sophus-1.22.10
sudo cmake --build /tmp/Sophus-1.22.10 --target install --parallel `nproc`
```
- Install OpenEB 5.0
Follow the [OpenEB install guide](#)⁸
- Extract the contents of the ZIP archive mentioned above and open a terminal in the unzipped folder.
- Compile the sample `metavision_active_marker_3d_tracking` as shown below:

```
cmake -S metavision_active_marker_3d_tracking/ -B /tmp/active_marker_build -DMAKE_BUILD_TYPE=Release
sudo cmake --build /tmp/active_marker_build --target install --parallel `nproc`
```

- Launch the script:

```
utils/scripts/host/launch_<sensor>.sh <kria.IP.address>
```

You should see the 3D rendering as mentioned at the beginning of the section

Note

This script connects to the Kria board via SSH using the root user. A root password must be set for this to work. If not, first connect to the Kria board and set a password using `sudo passwd root`.

Build the Embedded Application from Sources

As explained before, the embedded application is built and integrated into the image directly through Petalinux.

Let's see how to rebuild it from source in order to customize the `metavision_active_marker_3d_tracking` application.

- Install [Petalinux 2022.2](#)⁹ in <Petalinux_TOOLS>
- Clone the repository <https://github.com/prophesee-ai/petalinux-projects>¹⁰ in <Petalinux_PROJECT>:

```
git clone git@github.com:prophesee-ai/petalinux-projects.git -b kv260-2022.2
```

- Download the `metavision_active_marker_3d_tracking` source code at [this page](#)¹¹.

Note

If you don't have access to this Knowledge Center page, contact support@prophesee.ai¹² asking for a KC account and a specific access to the Kria Starter Kit page.

- If you want to customize the application, extract the archive and apply your changes in the source code. Then create a new archive and copy it in `petalinux-projects/project-spec/meta-user/recipes-vision/metavision-active-marker-3d-tracking/files/`

Note

It is important to keep the original file name `metavision_active_marker_3d_tracking-openeb_v5.0.0.tar.gz` as it is referenced in the recipe.

Otherwise, you should rename as well the reference of the file name <SRC_URI> inside the recipe file `<Petalinux_PROJECT>/project-spec/meta-user/recipes-vision/metavision-active-marker-3d-tracking/metavision_active_marker_3d_tracking-openeb_v5.0.0.bb`.

Similar step should be performed if you generated a new .xsa file to configure your FPGA, as explained in the page [Tutorials/Build the Vivado Design](#). In that case, the recipe file to check is `<Petalinux_PROJECT>/project-spec/meta-user/recipes-firmware/prophesee-kv260-<sensor>/prophesee-kv260-<sensor>.1.0.0.bb`. Modify the file to get the xsa from the "files" folder instead of the github artifacts.

Alternatively, you can also check the section [edit kria applications](#).

- Run Petalinux tools to build the system:

```
cd <Petalinux_PROJECT>
source <Petalinux_TOOLS>/settings.sh
petalinux-build
```

- Generate a microSD card image:

```
petalinux-package --wic --bootfiles "randisk.cpio.gz.u-boot,boot,scr,Image,system.dtb,system"
```

Then you can use your new generated image to [flash your SD card](#).

Custom Marker

Different active marker devices can be used as long as the modulation follows the same principle as explained [in the Metavision documentation page for active markers](#)¹³.

The IDs of the LEDs can be arbitrarily changed, as well as their position.

Two options are possible:

- Reprogram the Active Marker board: contact Prophesee customer support (support@prophesee.ai) to obtain a different firmware. Then, update the `active_marker_config.json` file.
- Design a custom Active Markers PCB: follow the suggested modulation to design your own active markers PCB. In this case, update both the IDs and the positions in the configuration to match the new board model.

Custom Scene

We offer an additional 3D scene configuration that allows users to run the app in a different virtual environment; e.g., a living room:



The scene files can be found in the [Active Marker source code archive](#)¹⁴, inside the folder `utils` as mentioned above.

Going Further

To go further with another application, you can also try the Event ML application developed by LogiTronix in coordination with Prophesee and AMD. The application demonstrates YoloV7 and YoloV4-tiny ML models for Vitis AI. The source code of the demo is available here:

<https://github.com/LogiTronixInc/Kria-Prophesee-Event-VitisAI>¹⁵