GPNTUG

GoPoint for i.MX Applications Processors User Guide

Rev. 10.0 — 30 September 2024 User guide

Document information

Information	Content
Keywords	GoPoint, Linux demo, i.MX demos, MPU, ML, machine learning, multimedia, ELE, GoPoint for i.MX Applications Processors, i.MX Applications Processors
Abstract	This document explains how to run GoPoint for i.MX Applications Processors, while also covering the included demos and how to operate them.

1 Introduction

GoPoint for i.MX Applications Processors is a user-friendly application that allows the user to launch preselected demonstrations included in the NXP provided Linux Board Support Package (BSP).

GoPoint for i.MX Applications Processors is for users who are interested in showcasing the various features and capabilities of NXP provided SoCs. The demos included in this application are meant to be easy to run for users of all skill levels, making complex use cases accessible to anyone. Users need some knowledge when setting up equipment on Evaluation Kits (EVKs), such as changing Device Tree Blob (DTB) files.

This user guide is intended for end users of GoPoint for i.MX Applications Processors. This document explains how to run GoPoint for i.MX Applications Processors while also covering the included demos and how to operate them.

To use this software, users need at a minimum:

- A supported NXP Evaluation Kit (EVK)
- A display output (MIPI DSI or HDMI)
- A connected mouse

Note: *Some demos require more than the minimum required equipment. To find the required materials for each demo, refer to Included demos.*

1.1 Installing GoPoint for i.MX Applications Processors

GoPoint for i.MX Applications Processors comes preinstalled on NXP-provided demo Linux images. These images are available at *Embedded Linux for i.MX Applications Processors* (document IMXLINUX). Alternatively, a user can build the demo images, which include GoPoint for i.MX Applications Processors, by following the *i.MX Yocto Project User Guide* (document IMXLXYOCTOUG). In both cases, the "imx-image-full" image and "fsl-imx-xwayland" distribution must be used.

2 Demo launcher

This section describes the demo launcher.

2.1 Graphical user interface

On boards where GoPoint for i.MX Applications Processors is available, an NXP logo is displayed on the top left-hand corner of the screen. Users can start the demo launcher by clicking this logo.

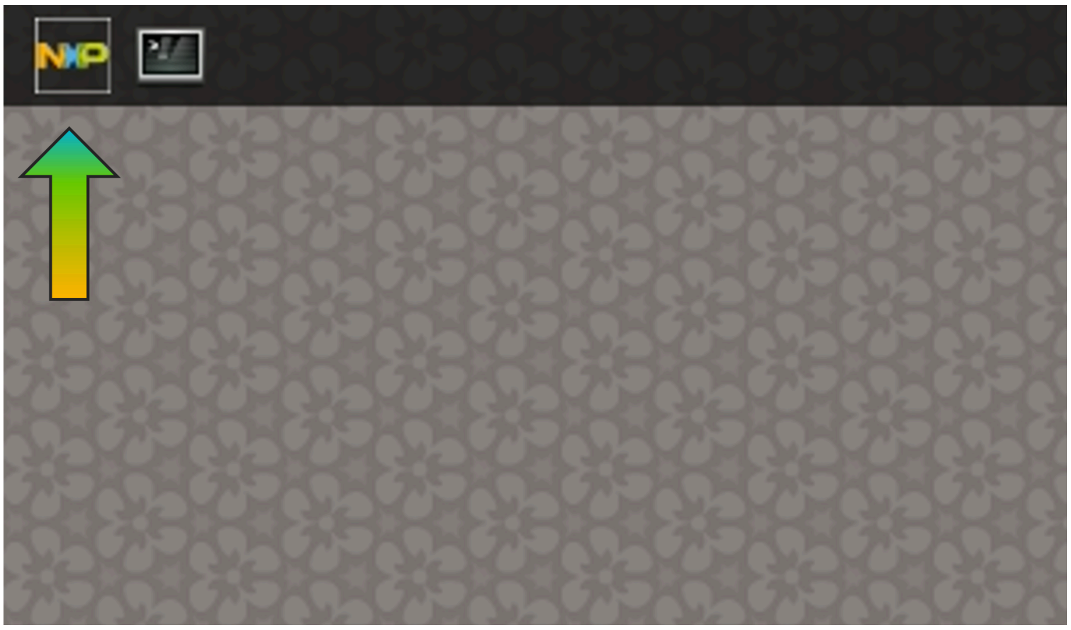


Figure 1. GoPoint for i.MX Applications Processors logo

After opening the program, users can launch demos using the following options shown in Figure 2:

1. To filter the list, select the icon on the left to expand the filter menu. From this menu, users can select a category or subcategory that filters the demos displayed in the launcher.
2. A scrollable list of all the demos supported on that EVK appears in this area with any filters applied. Clicking a demo in the launcher brings up information about the demo.
3. This area displays the names, categories, and description of the demos.
4. Clicking **Launch Demo** launches the currently selected demo. A demo can then be force-quit by clicking the **Stop current demo** button in the launcher (appears once a demo is started).
   Note: *Only one demo can be launched at a time.*

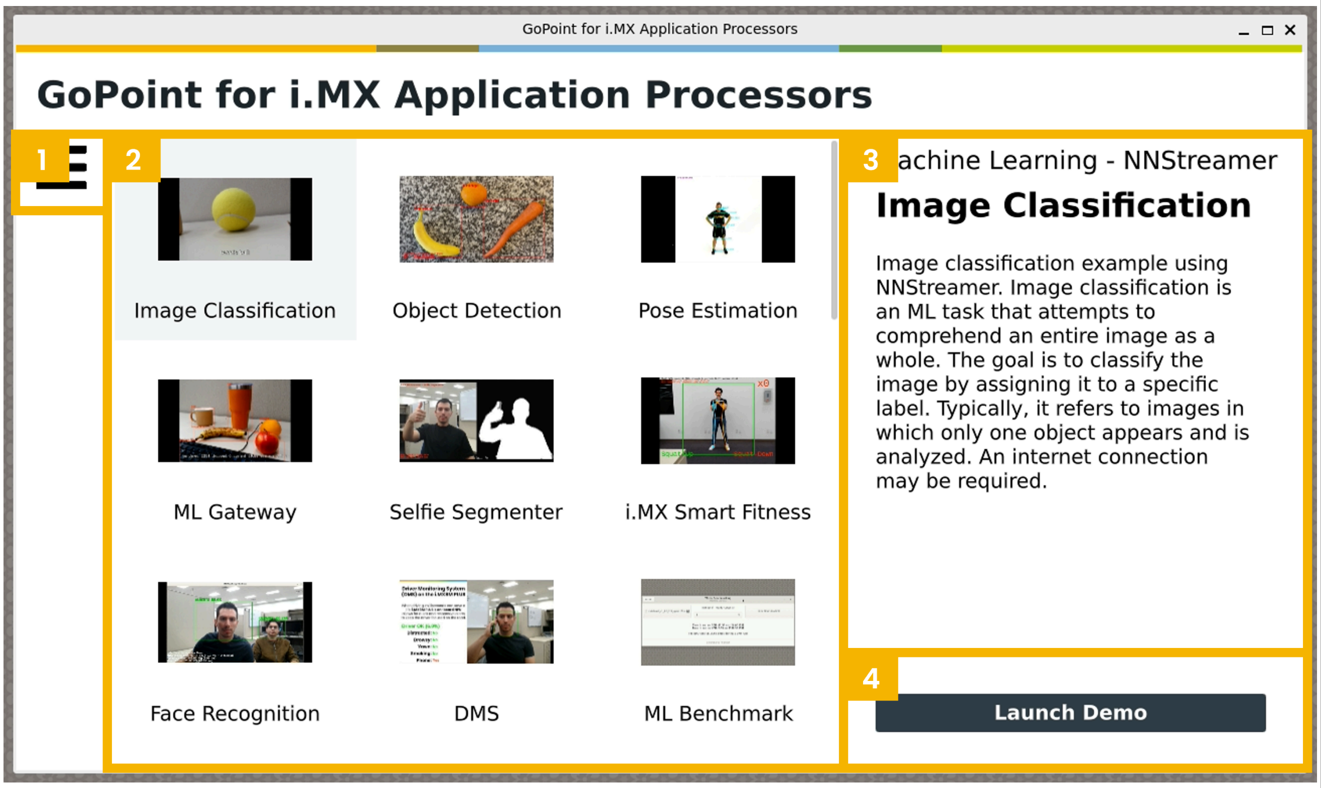


Image classification example using NNStreamer. Image classification is an ML task that attempts to comprehend an entire image as a whole. The goal is to classify the image by assigning it to a specific label. Typically, it refers to images in which only one object appears and is analyzed. An internet connection may be required.

Figure 2. GoPoint for i.MX Applications Processors

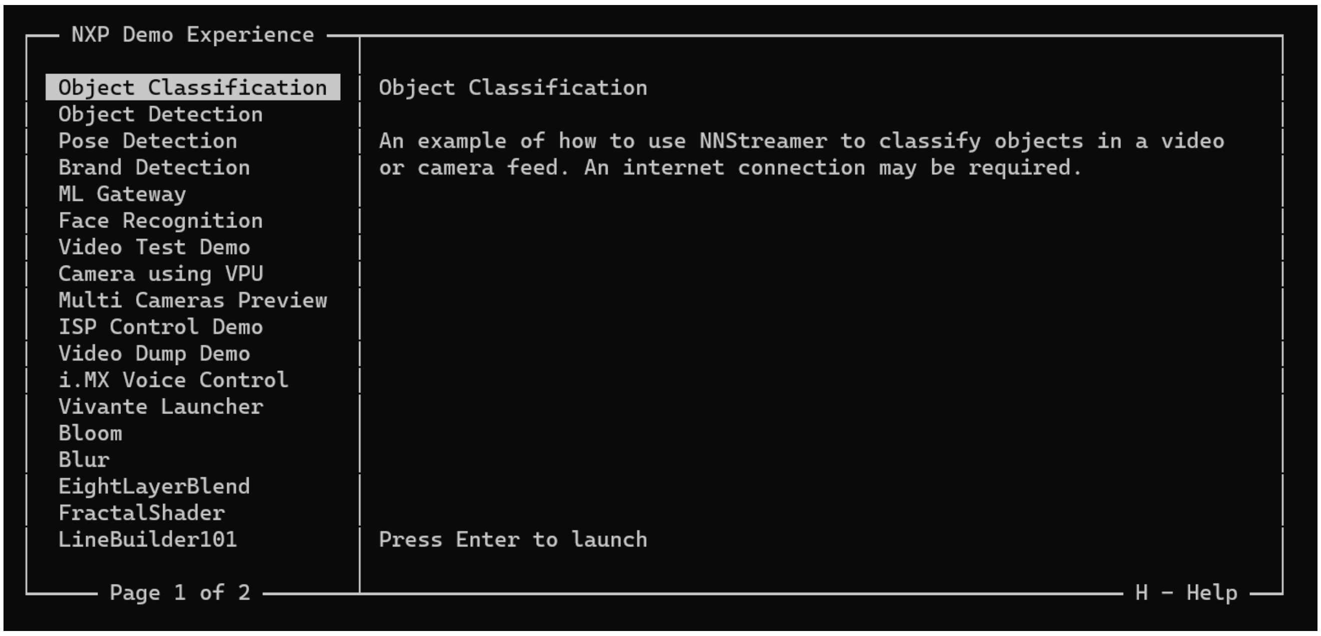
2.2 Text user interface

Demos can also be launched from the command line through log-in into the board remotely or using the onboard serial debug console. Remember that most demos still require a display to run successfully.

Note: *If prompted for a login, the default user name is "root" and no password is required.*

To start the text user interface (TUI), type the following command into the command line:

```
# gopoint tui
```



```
┌─ NXP Demo Experience ─┬──────────────────────────────────────────────────────────────────┐
│ Object Classification │ Object Classification                                            │
│ Object Detection      │                                                                  │
│ Pose Detection        │ An example of how to use NNStreamer to classify objects in a video │
│ Brand Detection       │ or camera feed. An internet connection may be required.          │
│ ML Gateway            │                                                                  │
│ Face Recognition      │                                                                  │
│ Video Test Demo       │                                                                  │
│ Camera using VPU      │                                                                  │
│ Multi Cameras Preview │                                                                  │
│ ISP Control Demo      │                                                                  │
│ Video Dump Demo       │                                                                  │
│ i.MX Voice Control    │                                                                  │
│ Vivante Launcher      │                                                                  │
│ Bloom                 │                                                                  │
│ Blur                  │                                                                  │
│ EightLayerBlend       │                                                                  │
│ FractalShader         │                                                                  │
│ LineBuilder101        │ Press Enter to launch                                            │
│                       │                                                                  │
└──── Page 1 of 2 ──────┴───────────────────────────────────────────────────── H - Help ───┘
```

Figure 3. Text user interface

The interface can be navigated using the following keyboard inputs:

- **Up and down arrow keys**: Select a demo from the list on the left
- **Enter key**: Runs the selected demo
- **Q key or Ctrl+C keys**: Quit the interface
- **H key**: Opens the help menu

Demos can be closed by closing the demo onscreen or pressing the "Ctrl" and "C" keys at the same time.

3 Demos included

This chapter describes the available demos that can be launched from GoPoint for i.MX Applications Processors's demo launcher. To see the available demos for a specific Linux version and board, refer to *GoPoint for i.MX Applications Processors* (document GPNTRN).

Note: *If the demo covers the demo launcher or any required window, drag the necessary windows, including video output windows, with the mouse.*

3.1 Machine learning demos

The following demos show machine learning use cases that are possible with the Neural Processing Unit (NPU) included on-chip.

3.1.1 NNStreamer demos

NNStreamer is a set of GStreamer plugins that enable machine learning in video pipelines. The included demos use NNStreamer to create video outputs that overlay inference information onto the video.

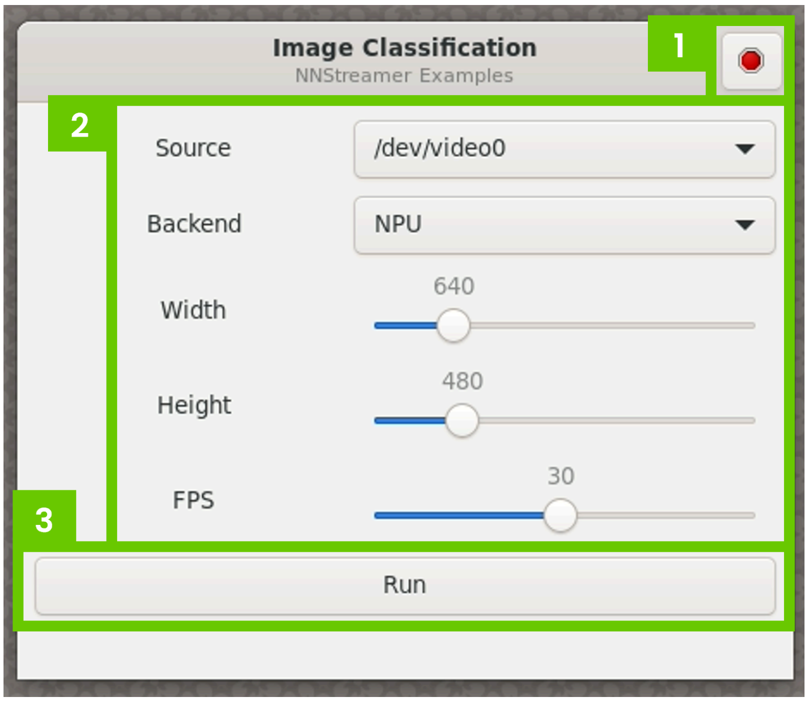


Figure 4. NNStreamer launcher

All NNStreamer demos have similar user interfaces. When these demos are launched, users are presented with a control panel, as shown in Figure 4.

1. This button can be used to quit the demo and stop video playback.
2. Various options can be set before a demo is run:
   - **Source**: Select a camera or use the example video provided (requires an Internet connection, only available in the "Pose Estimation" demo).
   - **Back-end**: Select whether to use the NPU (if available) or CPU for inference.
   - **Height**: Select the input height of the video if using a camera.
   - **Width**: Select the input width of the video if using a camera.
   - **FPS**: Select the input FPS of the video if using a camera.
3. Clicking the **Run** button locks the current settings and starts the camera feed and inferencing.

Note: *If the NPU is selected, the first time the user runs a demo, the NPU performs a process called warming up. Here, the NPU must convert the model file into a format that it can read. While this result is cached to speed up future runs, this process can take some time.*

3.1.1.1 Image classification

Required materials: Mouse, display, camera (if tested with camera), and Internet connection.

This demo launches a GStreamer pipeline that gets a video feed from a camera or video file. It runs a classification inference on the video frames as they come in. This demo uses a pretrained quantized MobileNetV1 TensorFlow Lite (TFLite) model. This model is trained on objects included in the ImageNet Large-Scale Visual Recognition Challenge 2012 (ILSVRC2012) object set. The result of the inference is then displayed within the video frame.

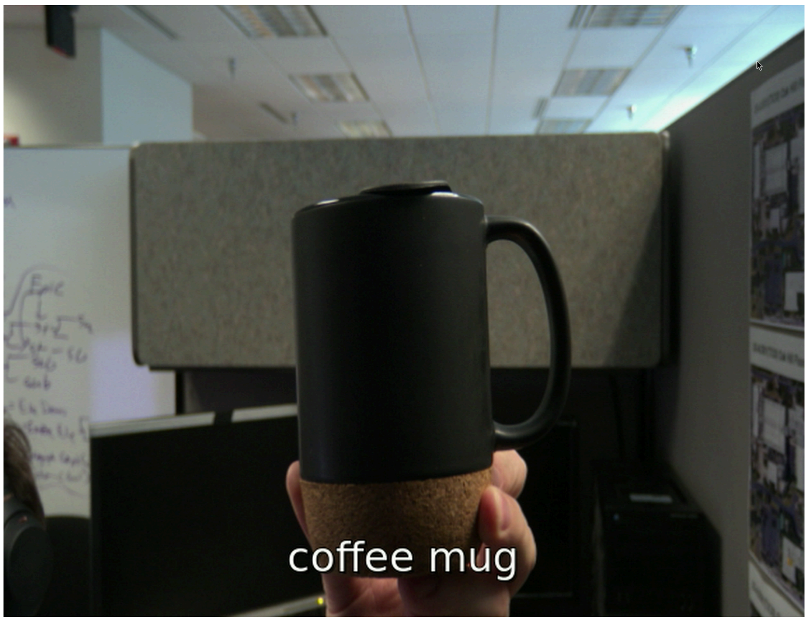


Figure 5. NNStreamer output classification example

When the demo starts, a video overlay with the label with the highest probability of being in the image is displayed.

3.1.1.2 Object detection

Required materials: Mouse, display, camera (if tested with camera), and Internet connection.

This demo launches a GStreamer pipeline that gets a video feed from a camera or video file. It runs a detection inference on the video frames as they come in. This demo uses a pretrained quantized MobileNet Single Shot Detection (SSD) V2 TFLite model. This model is trained on objects included in the Common Objects in Context (COCO) object dataset. The result of the inference is then displayed within the video frame.

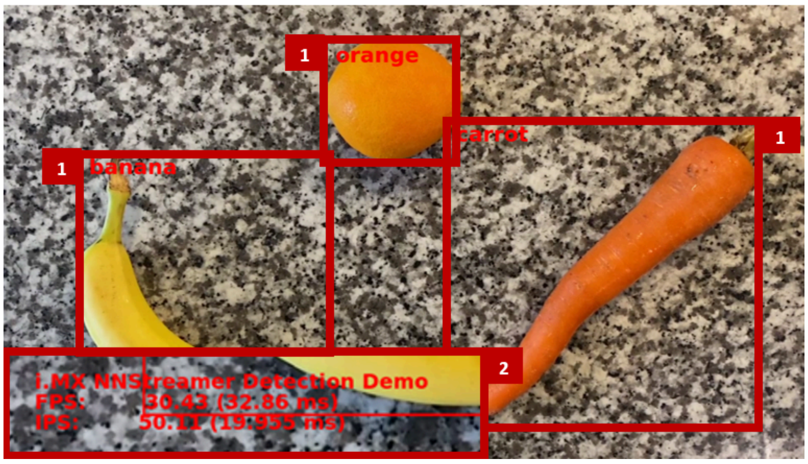


Figure 6. NNStreamer object detection example

When the demo starts, a video overlay of the detected objects has boxes drawn around them, and their labels are displayed near the top-left corner.

3.1.1.3 Pose detection

Required materials: Mouse, display, camera (if tested with camera), and Internet connection.

This demo launches a GStreamer pipeline that gets a video feed from a camera or video file and runs a pose detection inference on the video frames as they come in. This demo uses a pretrained quantized MoveNet Single Pose Lightning TFLite model that is trained on 17 body points. When the video starts, the detected pose is overlaid onto the incoming video feed.

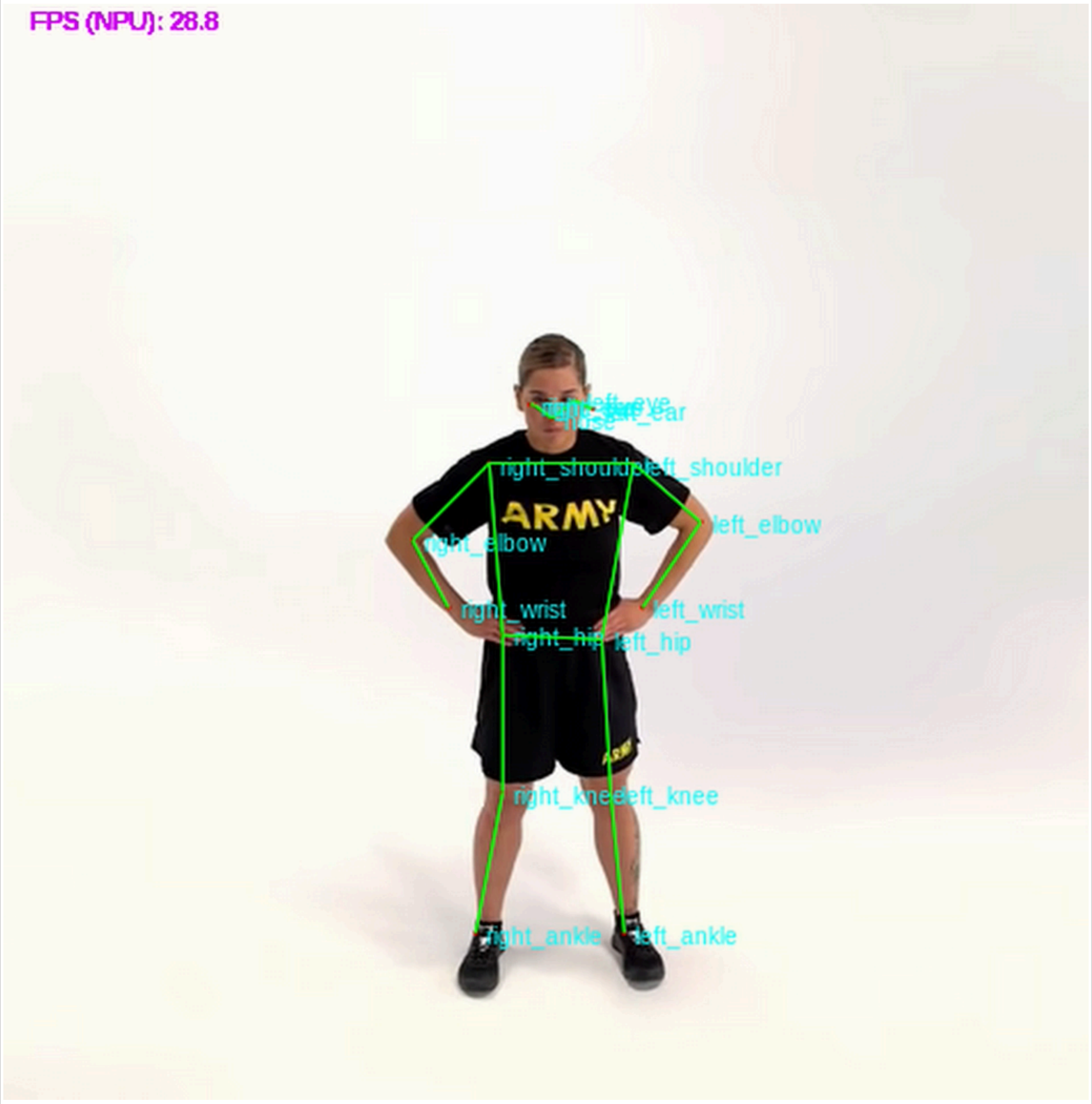


Figure 7. NNStreamer pose detection example

3.1.1.4 Machine learning gateway

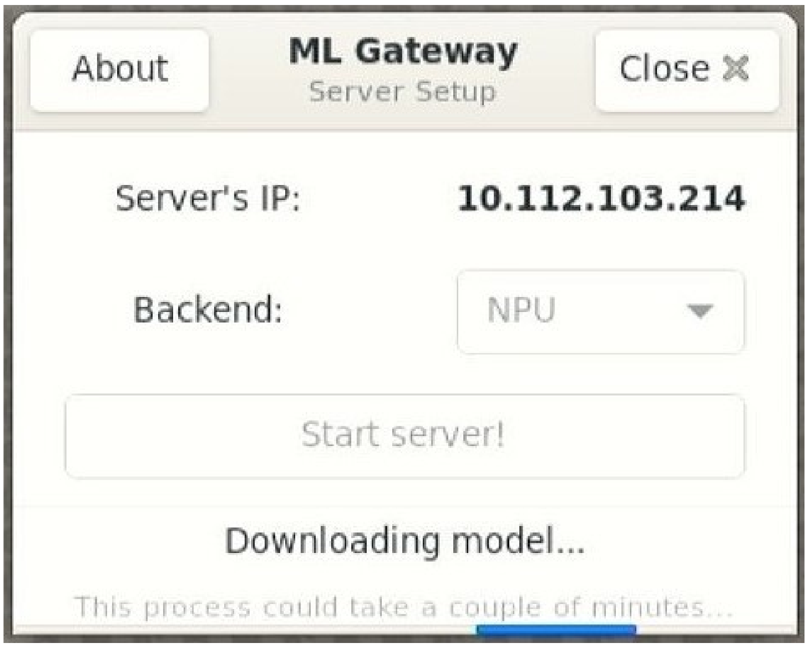
Required materials: Mouse, display, and camera for each device. Two devices that are connected to the same network are required.

Machine Learning (ML) Gateway allows devices with limited ML processing power to use the resources of another much more powerful device to accelerate ML inferences. This application configures the i.MX 8M Plus or i.MX 93 EVKs as servers. Therefore, it allows other devices such as the i.MX 8M Mini EVK to connect and use their NPU for ML acceleration. Also, the server broadcasts its IP address for easy client connection to the ML Gateway. This application has been developed based on the *i.MX 8M Plus Gateway for Machine Learning Inference Acceleration* (document AN13650).

3.1.1.4.1 Setup

To start the ML Gateway, perform the following steps:

1. Launch **GoPoint** on the board that is going to be used as a server, that is, i.MX 8M Plus or i.MX 93 EVK (both have NPU).
2. Click the **ML Gateway** application shown in the launcher menu.
3. To start ML Gateway, select the **Launch Demo** button, which automatically detects the board to be configured as a server. It starts downloading the model to be used for object detection tasks.
4. Wait until it displays "Model is ready for inference!". If the model fails to download, check the Internet connection. Also, if the package fails to download, ensure that the board has the current date and time when the demo is being tested.



Server's IP: 10.112.103.214

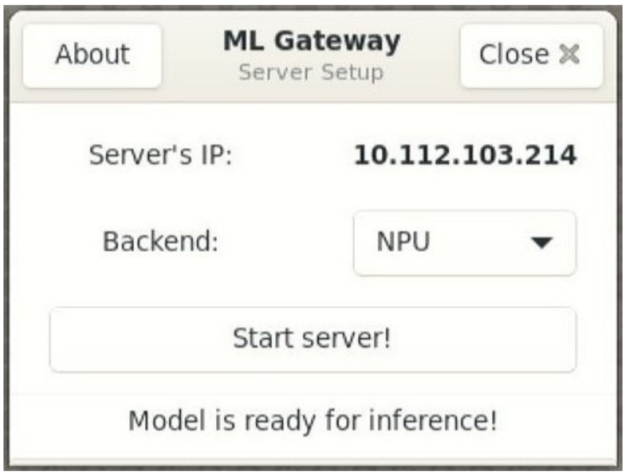
Downloading model...

This process could take a couple of minutes...

Figure 8. ML Gateway device type select

3.1.1.4.2 Start the server

The server board shows its current IP address in the GUI. By selecting the appropriate option from the dropdown, the server allows users to choose whether to perform inferences on the NPU or the CPU. It is expected to use the NPU. However, the user can test on the CPU to compare the inference performance when the NPU is not accelerating the ML model. When ready, click the **Start Server!**. If the server is set up successfully, the "Server is running" gets displayed.

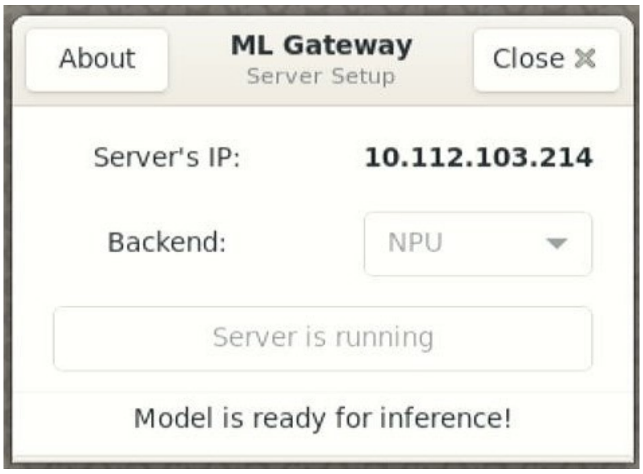


Server's IP: 10.112.103.214

Backend: NPU

Start server!

Model is ready for inference!



Server's IP: 10.112.103.214

Backend: NPU

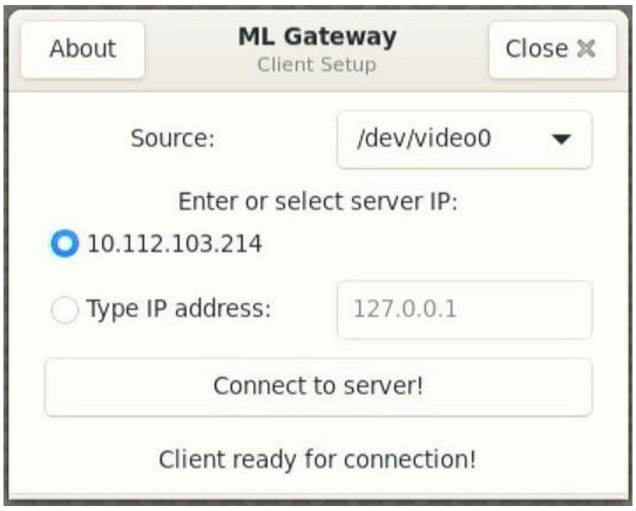
Server is running

Model is ready for inference!

Figure 9. ML gateway server setup

3.1.1.4.3 Connect client and start inference

When setting up a client, the device searches for a server IP and, if found, displays it as an option in the window. If the client is unable to detect the IP address, users can also type in their custom IP address. Ensure that the camera source is selected for the correct device. When ready, click the **Connect to Server!**. The device connects to the server and displays a video output with the detected objects marked.



Source: /dev/video0

Enter or select server IP:

● 10.112.103.214

○ Type IP address: 127.0.0.1

Connect to server!

Client ready for connection!

Figure 10. ML gateway client setup

Note: *Sometimes running the application for the first time on the client causes some latency seen in the bounding boxes being detected. It can happen only when the server is configured on the i.MX 8M Plus EVK, which needs a warming-up time to load the ML model. If this latency occurs, stop the client process and connect again to the server. It most probably fixes the latency issue and the bounding boxes must be shown in real time on the second run.*

3.1.1.4.4 Running the application

When ML Gateway starts running on a client, a video overlay is shown with the following information:

1. Class name of the detected objects and corresponding bounding boxes.
2. Total rendered frames, dropped frames, current frames per second (FPS) and average FPS.

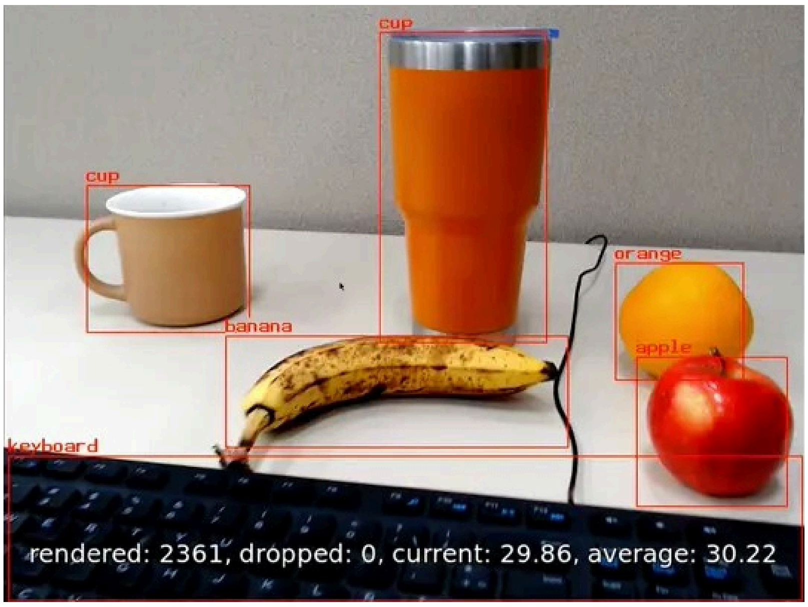


Figure 11. Object detection on client side (i.MX 8M Mini)

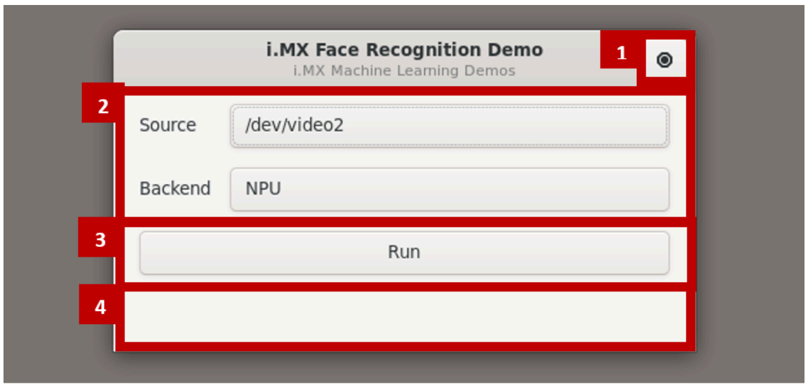
3.1.2 OpenCV demos

The following demo uses the OpenCV library to display video frames with inference data displayed on them.

3.1.2.1 Face recognition

Required materials: Mouse, display output, camera, and an Internet connection.

In this demo, users can register and recognize faces in a video feed. The demo first uses a quantized TFLite SSD MobileNetV2 model to detect the faces that are in the video frame. Then, it crops the faces and uses a quantized TFLite FaceNet model to create face embeddings. These face embeddings can be used to compare with previously saved faces.



Source: /dev/video2

Backend: NPU

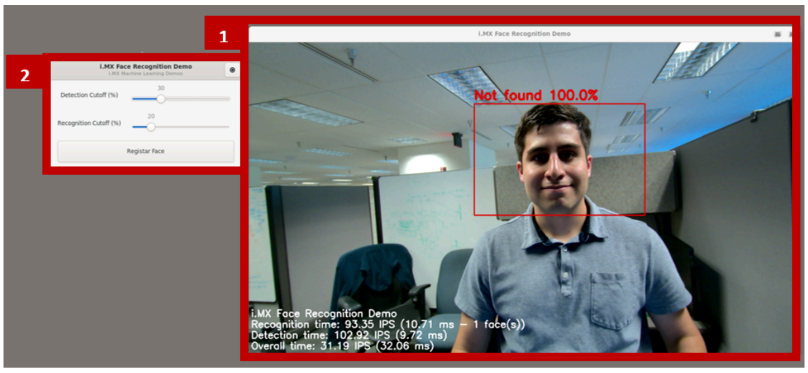
Figure 12. Face recognition options select

When the demo starts, a new window is displayed that allows the user to set face recognition options:

1. This button can be used to quit the demo.
2. These options allow certain aspects of demos to be changed:
   - **Source**: Select the camera to use.
   - **Back-end**: Select whether to use the NPU (if available) or CPU for inference.

3. This button confirms the settings and starts the demo.
4. This area displays the status updates while the demo is starting.

When the **Run** button is pressed, the demo first downloads the required model from the Internet and then warms up the NPU. The first time the demo is run, this warm-up time can take a couple of minutes. In future runs, the warm-up time is less.

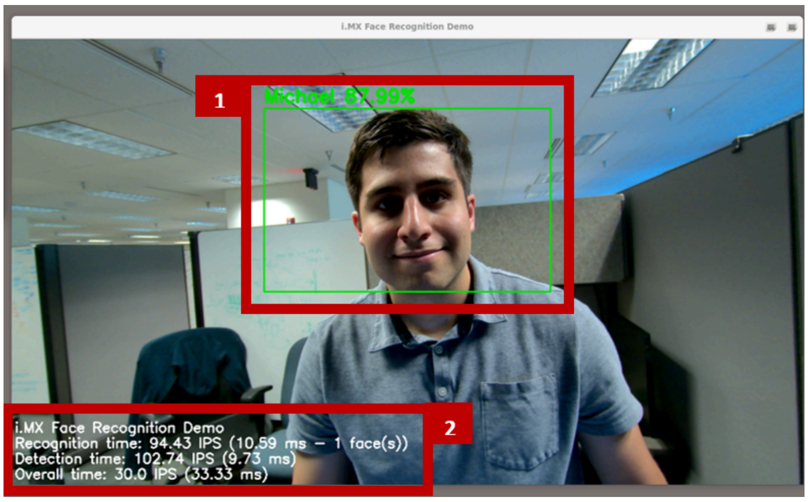


```
i.MX Face Recognition Demo
Recognition time: 93.35 IPS (10.71 ms – 1 face(s))
Detection time: 102.92 IPS (9.72 ms)
Overall time: 31.19 IPS (32.06 ms)
```

Figure 13. Face recognition demo

When the demo is ready to begin, the following window opens:

1. A camera view with inference information layered over the frame.
2. A control panel.



```
i.MX Face Recognition Demo
Recognition time: 94.43 IPS (10.59 ms – 1 face(s))
Detection time: 102.74 IPS (9.73 ms)
Overall time: 30.0 IPS (33.33 ms)
```

Figure 14. Face recognition camera view

The camera view window has the following information:

- Any faces that are detected are outlined with a box. A red box means that no face is saved with a similarity higher than the recognition cutoff (%) when compared to the face in the box. A green box means that a face is saved with a similarity higher than the recognition cutoff (%) when compared to the face in the box. If the detected face matches any of the registered faces, the name of this registered face is displayed above along with the percentage similarity. If there is no match, the label "Not found" is displayed instead.
- The following statistics are shown:
  - **Recognition time**: It is the IPS and inference time in milliseconds required to generate a face embedding for faces in the picture. If there are multiple faces in a frame, the total time for all faces varies based on the number of faces in the frame.

– **Detection time**: It is the IPS and inference time in milliseconds required to identify all the faces in the picture.
– **Overall time**: It is the IPS and inference time in milliseconds required to prepare a video frame from when the application receives it.

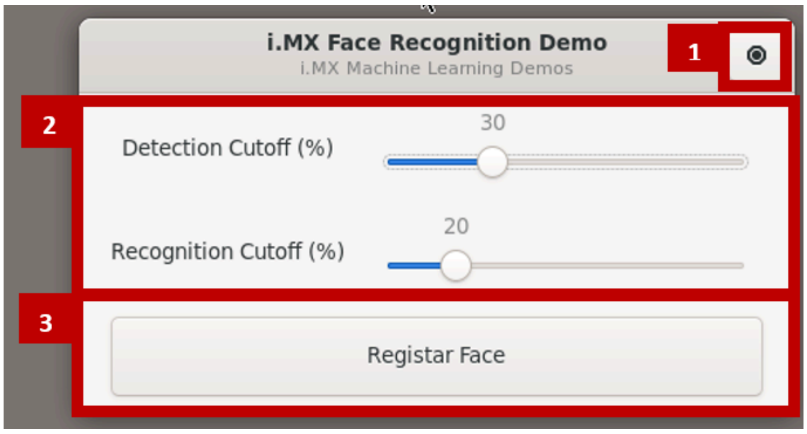


Figure 15. Face recognition control panel

To change the behavior of the demo, the control panel has the following controls:

- **Detection Cutoff (%)**: It sets the percentage confidence required to detect a face in a video frame. Slide this option up if other objects are being detected as faces or down if faces are not being detected.
- **Recognition Cutoff (%)**: It sets the percentage similarity required to recognize a face in a video frame. Slide this option up if the demo is falsely identifying faces or down if the demo does not identify the face.
- **Register Face**: It registers new faces to the database to be recognized. It is done locally and is removed when the application closes.

How to register a face:

1. Click the **Register Face** button on the control panel.
2. A countdown from 5 seconds starts. At the end of the countdown, the application saves the current frame and detects any faces in the frame.
3. For all faces in the frame, a window prompt appears, as shown in Figure 16. Type the name of the face that appears in the blue box in the camera view. When done, click the **Register Face** button. If a face must not be registered, click the **Skip Face** button.

Note: *Registering the same face multiple times can help increase the data pool that the database has to pull from, which improves facial recognition accuracy.*

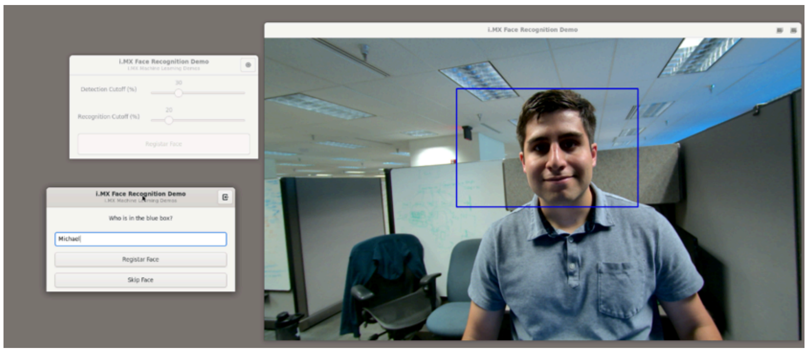

i.MX Face Recognition Demo
i.MX Machine Learning Demos
Detection Cutoff (%)
30
Recognition Cutoff (%)
20
Registar Face
i.MX Face Recognition Demo
i.MX Machine Learning Demos
Who is in the blue box?
Michael
Registar Face
Skip Face
i.MX Face Recognition Demo

Figure 16. Face recognition for a registered face

3.1.3 TensorFlow Lite demos

TFLite demos are basic demos that show the basic functionality of the TensorFlow library.

3.1.3.1 ML benchmark

Required materials: Mouse, display, and Internet connection.

This application allows the user to benchmark TFLite models on both the CPU and NPU to compare the performance easily. The tool is user-friendly, allowing users to select and load the desired model for benchmarking and also to select the number of threads used in the CPU. Ensure that the selected models are in TFLite format only. By default, a MobileNetV1 model is available for benchmarking.

3.1.3.2 Driver monitoring system demo

Required materials: Mouse, display, camera, and Internet connection.

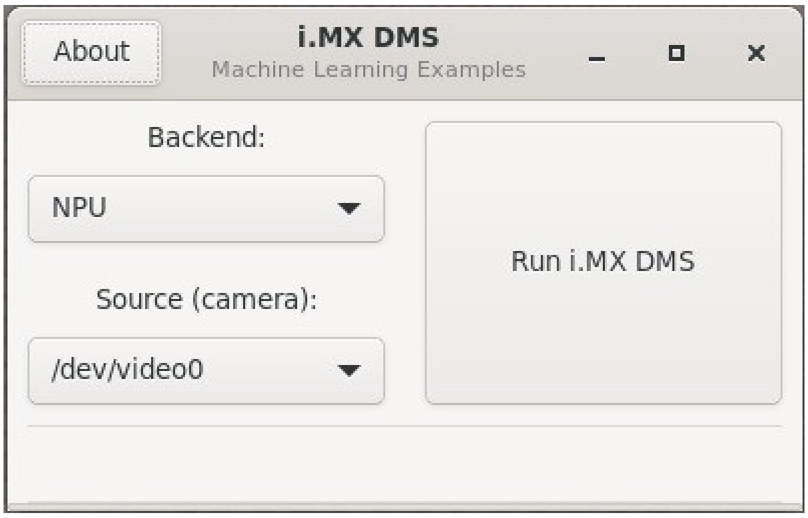
With rapid development in the automotive industry recently, the Driver Monitor System (DMS) has become a common requirement in modern vehicles. DMS is essentially a vehicle safety system that assesses the alertness of the driver and warn the driver if necessary. This system plays a crucial role in preventing road accidents and safeguarding the driver. With the help of ML and Neural Network (NN), DMS can now achieve high accuracy and low latency.

GoPoint application demonstrates the implementation of DMS on the i.MX 8M Plus and i.MX 93 platforms, and the NPU provided performance boost on these platforms.

3.1.3.2.1 Setup

To start the DMS, perform the following steps:

1. Launch **GoPoint** on the board and click the **DMS** application shown in the launcher menu.
2. To start DMS, select the **Launch Demo** button.
3. A window appears to let the user select the inference backend and camera source to be used. Ensure that a camera module, either MIPI-CSI or USB camera, is connected.
4. After selecting the inference backend and camera source in the drop-down menu, start the application by clicking **Run i.MX DMS**.



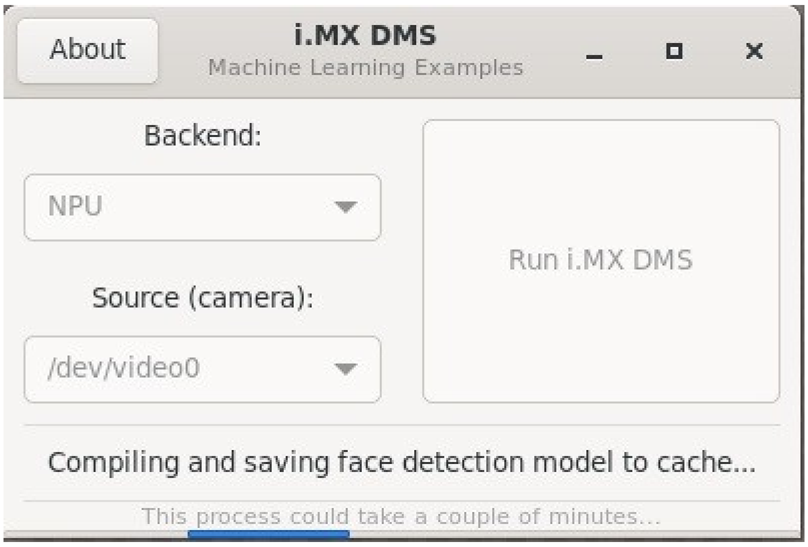
Backend: NPU

Source (camera): /dev/video0

Figure 17. DMS setup

When running the application on the i.MX 8M Plus, a warm-up time is needed for the models to be ready for acceleration on the NPU.

On the i.MX 93, the models are compiled using the Vela compiler for Ethos-U NPU acceleration. This process is automated, but can take up to a minute on each platform, during the initial run of the application. For future use the compiled models are stored as cache files and therefore, takes less time.

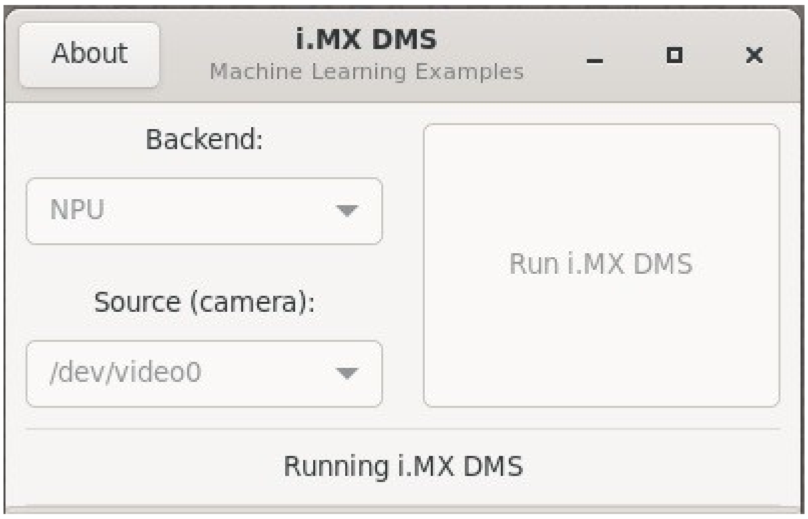


Compiling and saving face detection model to cache...

This process could take a couple of minutes...

Figure 18. DMS preprocessing

Once the process finishes and the models are ready, the application starts right away.



Running i.MX DMS

Figure 19. Running the DMS demo

3.1.3.2.2 Running the application

When the i.MX DMS starts running successfully, the following must be seen on the display:

1. Overall driver status and the five detailed statuses are displayed on the left side.
2. Camera preview and driver face bounding box are displayed on the right side.
3. When no human face is detected in the current camera frame, the overall driver status shows "Driver not found!" and five detailed statuses show "N/A". When multiple human faces are detected, the face that is closest to the camera center is chosen as the driver face.

4. The overall driver status is an estimation of the status of the driver based on all the detailed statuses. When a dangerous status/behavior is detected in the current frame, or the driver's face is not found, a penalty value is added to the overall driver status score. This value is shown on the right side as a percentage. The higher the score, the more dangerous the status of the driver is. When no dangerous status/behavior is detected in the current frame, the score subtracts a fixed number until it recovers to zero. The text and color of overall driver status changes according to the score.
5. Each of the five detailed statuses is an indicator of a dangerous driving status/behavior. The judgment of each status/behavior is based on the state given in Table 1.

Table 1. Detectable behavior

Status/behavior	Detected	Not detected
Distracted	Driver's face is facing left or right	Driver's face is facing front
Drowsy	Driver's eye is closed (blinking is excluded)	Driver's eye is open
Yawn	Driver's mouth is open	Driver's mouth is closed
Smoking	Cigarette is detected	Cigarette is not detected
Phone	Cell phone is detected	Cell phone is not detected

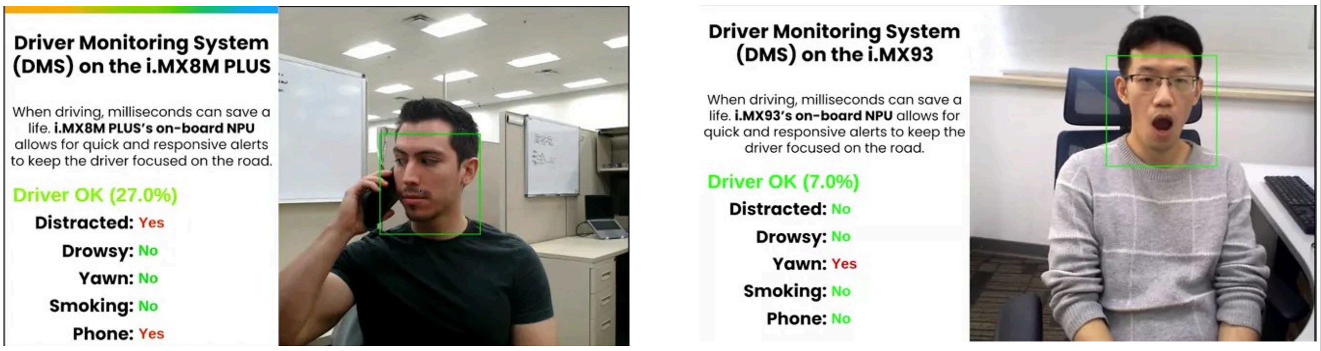

Driver Monitoring System (DMS) on the i.MX8M PLUS
When driving, milliseconds can save a life. i.MX8M PLUS's on-board NPU allows for quick and responsive alerts to keep the driver focused on the road.
Driver OK (27.0%)
Distracted: Yes
Drowsy: No
Yawn: No
Smoking: No
Phone: Yes
Driver Monitoring System (DMS) on the i.MX93
When driving, milliseconds can save a life. i.MX93's on-board NPU allows for quick and responsive alerts to keep the driver focused on the road.
Driver OK (7.0%)
Distracted: No
Drowsy: No
Yawn: Yes
Smoking: No
Phone: No

Figure 20. DMS demo output

3.1.3.3 Selfie segmenter

Required materials: Mouse, display, camera, and Internet connection.

Selfie Segmenter showcases the ML capabilities of i.MX SoCs by using an NPU to accelerate an instance segmentation model. This model lets the user segment the portrait of a person and can be used to replace or modify the background of an image. Its architecture is based on MobileNetV3 with added customized decoder blocks for segmentation.

The following are two versions of the same model:

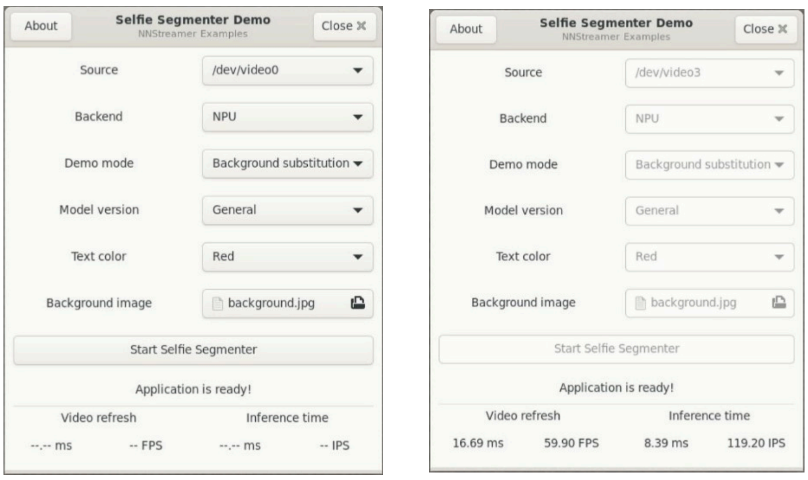
- *general* [256x256x3]: The general version of this model is more accurate than the landscape version due to its bigger input size. Therefore, more features can be extracted.
- *landscape* [144x256x3]: The landscape version runs faster and achieves real-time performance in both CPU (XNNPack delegate) or NPU (VX delegate and Ethos-U delegate).

This application is developed using GStreamer and NNStreamer. On the i.MX 93, PXP acceleration is used for the color space conversion and frame resizing during pre-processing and post-processing of data. On the i.MX 8M Plus, the 2D-GPU accelerator is used for the same purpose.

3.1.3.3.1 Setup

To start the application, perform the following steps:

1. Launch **GoPoint** on the board and click the **Selfie Segmenter** application shown in the launcher menu.
2. To start Selfie Segmenter, select the **Launch Demo** button.
3. A window appears to let the user select the camera source to be used. Ensure that a camera module, either MIPI-CSI or USB camera, is connected.
4. Choose the backend (NPU or CPU) for ML inference.
5. Two different application modes are available: **Background substitution** and **Segmentation mask**. The first one replaces the background with a selected image and the latter is the segmentation mask predicted by the model. Select the one you prefer to test.
6. Both the **General** and **Landscape** versions of this model can be tested. Choose the version that you want.
7. If desired, change the color of the text shown in the video output.
8. For the **Background substitution** mode, the background image can be changed to any `image.jpg` file.
9. Start the application by clicking the **Start Selfie Segmenter**.



Source: /dev/video0
Backend: NPU
Demo mode: Background substitution
Model version: General
Text color: Red
Background image: background.jpg
Application is ready!
Video refresh: --.-- ms -- FPS
Inference time: --.-- ms -- IPS

Source: /dev/video3
Backend: NPU
Demo mode: Background substitution
Model version: General
Text color: Red
Background image: background.jpg
Application is ready!
Video refresh: 16.69 ms 59.90 FPS
Inference time: 8.39 ms 119.20 IPS

Figure 21. Selfie Segmenter options window

When running the Selfie Segmenter application on the i.MX 8M Plus, a warm-up time is needed for models to be ready for acceleration on the NPU.

On the i.MX 93, the models are compiled using the Vela compiler for Ethos-U NPU acceleration. This process is automated, but can take up to a minute on each platform, during the initial run of the application. For future use the compiled models are stored as cache files and therefore, takes less time. Once the process finishes and the models are ready, the application starts right away. When the application is running, the video refresh and inference time are shown in the launcher.

3.1.3.3.2 Running the application

- **Selfie Segmenter - background substitution**
  When the Selfie Segmenter is chosen for background substitution, the following is seen on display:
  1. Video information displayed at the bottom-left corner showing average frames per second (FPS) and average inferences per second (IPS).
  2. The background is replaced with the default image. This image can be changed if desired.

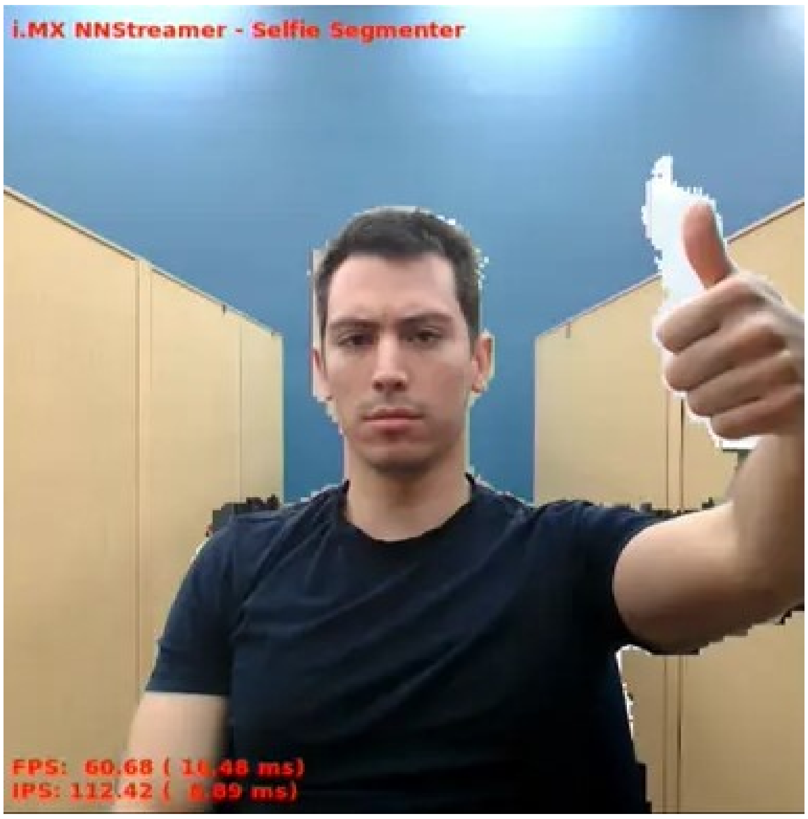


Figure 22. Selfie Segmenter – background substitution

- **Selfie Segmenter - segmentation mask**
  When the Selfie Segmenter is chosen for the segmentation mask, the following is seen on display:
  1. Video information displayed at the bottom-left corner showing average FPS and average IPS.
  2. Side to side comparison of input video and segmentation mask.

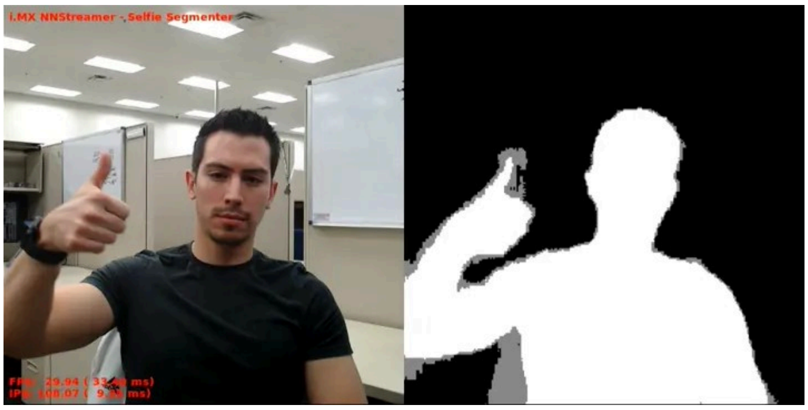


Figure 23. Selfie Segmenter – segmentation mask

3.1.3.4 i.MX smart fitness

i.MX Smart Fitness showcases the ML capabilities of i.MX SoCs by using an NPU to accelerate two Deep Learning vision-based models. Together, these models detect a person present in the scene and predict 33 3D-keypoints to generate a complete body landmark, known as *pose estimation*. From the pose estimation, a K-NN pose classifier classifies two-different body poses, which are as follows:

- Squat-Down
- Squat-Up

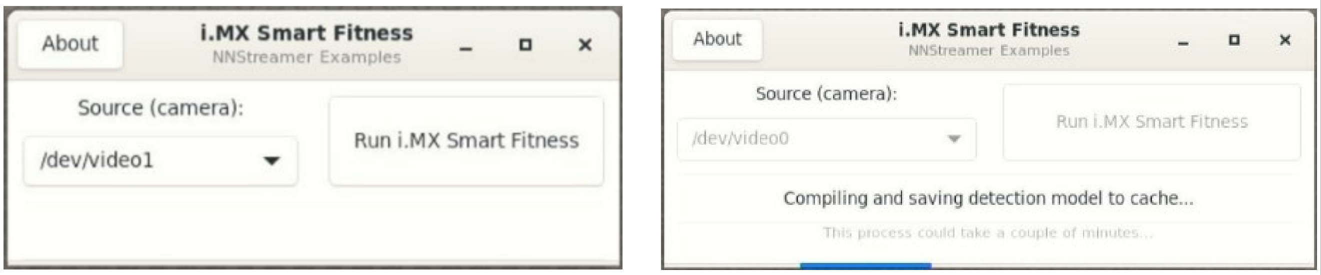
The application tracks the squat fitness exercise and the repetition counter is set to 12 repetitions in an infinite loop.

This application is developed using GStreamer and NNStreamer. On the i.MX 93, PXP acceleration is used for the color space conversion and frame resizing during pre-processing and post-processing of data. On the i.MX 8M Plus, the 2D-GPU accelerator is used for the same purpose.

3.1.3.4.1 Setup

To start the application, perform the following steps:

1. Launch **GoPoint** on the board and click the **i.MX Smart Fitness** application shown in the launcher menu.
2. To start i.MX Smart Fitness, select the **Launch Demo** button.
3. A window appears to let the user select the camera source to be used. Ensure that a camera module, either MIPI-CSI or USB camera, is connected.
4. Once detected and selected in the drop-down menu, start the application by clicking **Run i.MX Smart Fitness**.



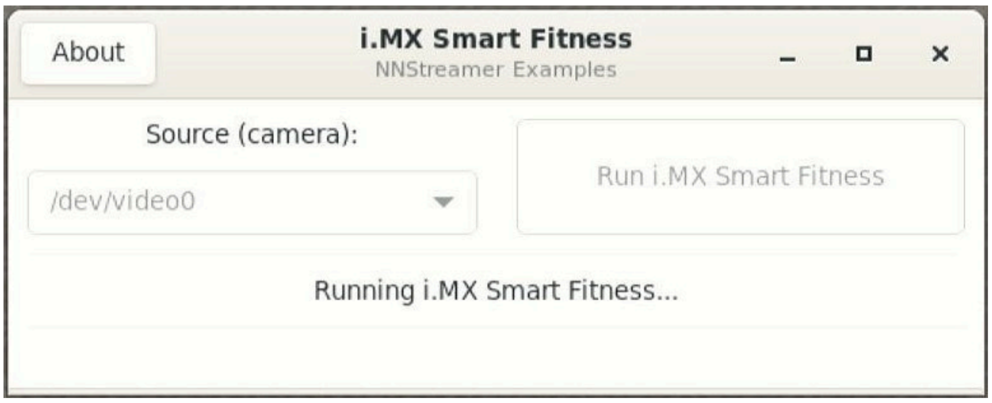
Compiling and saving detection model to cache...

This process could take a couple of minutes...

Figure 24. Smart Fitness setup window

When running the i.MX Smart Fitness application on the i.MX 8M Plus, a warm-up time is needed for models to be ready for acceleration on the NPU.

On the i.MX 93, the models are compiled using the Vela compiler for Ethos-U NPU acceleration. This process is automated, but can take up to a minute on each platform, during the initial run of the application. For future use the compiled models are stored as cache files and therefore, takes less time. Once the process finishes and the models are ready, the application starts right away.



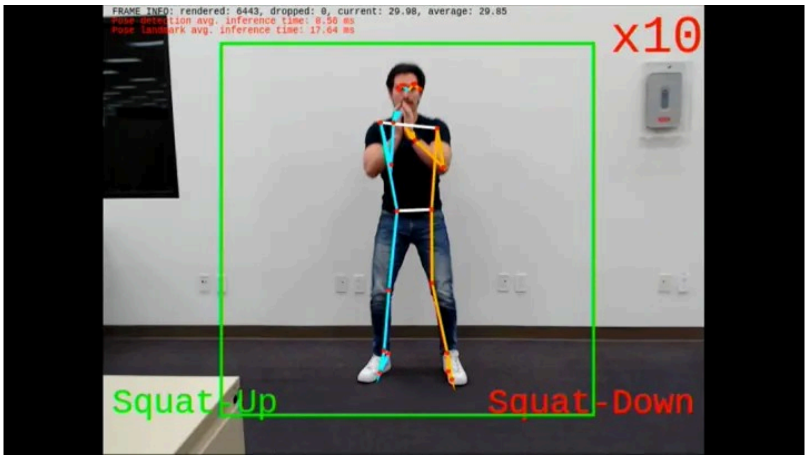
Running i.MX Smart Fitness...

Figure 25. Smart Fitness start up

3.1.3.4.2 Running the application

When i.MX Smart Fitness starts running, the following is seen on display:

1. Video information is displayed at the top-left corner showing total rendered frames, dropped frames, current FPS, and average FPS. Below is the average inference time of both ML models.
2. The counter of squats done by the person present in the scene is shown at the top-right corner. This counter starts at zero and goes up to twelve. Every twelve repetitions the counter resets to zero.
3. Classified poses are shown at the bottom corners, Squat-Up, and Squat-Down. These change color between green and red depending on which pose is being detected.
4. Bounding box is displayed for the detected human pose. Ensure that the person standing in front of the camera is visible in the scene otherwise the landmarks are not computed. If the camera is not able to see the person completely and the person stands in the center of the scene, the bounding box is red. It means that the person must adjust. If the pose is correctly detected, the bounding box is green. Therefore, allowing the computation of landmarks.
5. When the human pose is correctly detected and the bounding box is green, the 33 landmarks are shown on top of the human pose. These landmarks are used by the K-NN classifier to decide if the person is in the squat pose or not.



```
FRAME INFO: rendered: 6443, dropped: 0, current: 29.98, average: 29.85
Pose detection avg. inference time: 8.56 ms
Pose landmark avg. inference time: 17.64 ms
```

Figure 26. Running the Smart Fitness application

3.1.3.5 Low-power machine learning demo

The low-power machine learning application of the i.MX 93 showcases the machine learning capabilities of the i.MX 93 in low-power use cases. It uses the Cortex-M33 core to run ML model inference with the TFLite-micro framework. When running the application, the Cortex-A55 core running the Linux goes into Suspend mode to save power consumption. The NPU is not used in this application for the same reason.

The following two applications are implemented in the current release:

1. **Baby cry detection:** This application records one second of audio input from the MIC array on the i.MX 93 EVK board. It uses ML model inference to detect the sound of a crying baby. For this demo to function properly, the user must open a Cortex-M terminal and press "enter" to initiate recording. If a baby crying sound is detected, it wakes up the Cortex-A55 core and stops recording. If baby crying sound is not detected, it suspends the Cortex-M33 core for a configurable time interval and wakes up the Cortex-M33 core to record one second of audio again. This process runs in an infinite loop until a baby crying sound is detected.
2. **Keyword spot:** This application records one second of audio input from the MIC array on the i.MX 93 EVK board. It uses ML model inference to detect a keyword "UP" in the audio. If a keyword is detected, it wakes

up the Cortex-A55 core and stops recording. If no keyword is detected, it records one second of audio again. This process runs in an infinite loop until a keyword is detected.

3.1.3.5.1 ML models

The ML model used in baby cry detection is NXP trained and licensed under BSD-3-Clause license. The ML model used in the keyword spot application is originally from ARM-software under Apache-2.0 license.

3.1.3.5.2 Build steps and source code

The Cortex-M33 image of these two applications must be compiled with the i.MX 93 Cortex-M33 SDK. The source code is released in patch format in nxp-demo-experience-assets.

To build these two binaries, perform the following steps:

1. Generate and download the i.MX 93 Cortex-M33 SDK from MCUXpresso SDK Builder.
2. Select "i.MX 93 EVK" as board, "2.14.0" as SDK version, "Linux" as Host OS, and "ARM GCC" as Toolchain.
3. Extract the i.MX 93 Cortex-M33 SDK package on the Linux Host PC.
4. Set up toolchain according to the user guide in the SDK package.
5. To create a Git repository, use the `git init` command in the unzipped SDK folder for patch applying.
6. Download the patch "*0001-Add-low-power-baby-cry-detection-demo.patch*" and "*0001-Add-low-power-kws-detection-demo.patch*" from nxp-demo-experience-assets under patches folder.
7. Apply these two patches in the i.MX 93 Cortex-M33 SDK folder.
8. To build a Cortex-M33 image for a baby cry detection application, select the folder `M33_SDK/boards/mcimx93evk/demo_apps/lp_baby_detection/armgcc/` and run the `build_release.sh` script.
9. To build a Cortex-M33 image for a keyword spot application, select the folder `M33_SDK/boards/mcimx93evk/demo_apps/lp_kws_detection/armgcc/` and run the `build_release.sh` script.

3.2 Multimedia demos

This demo package has multimedia demos related to GStreamer, Image Signal Processing (ISP), and audio.

3.2.1 GStreamer demos

GStreamer is a flexible tool that allows users to build and deploy video pipelines quickly. The following examples show how to use GStreamer.

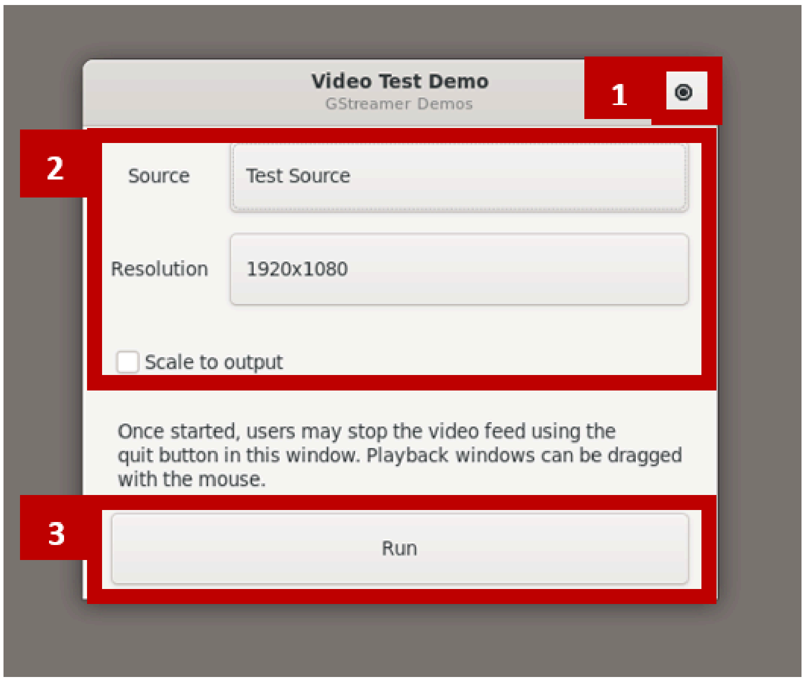
3.2.1.1 Video test demo

Required materials: Mouse, display output, and camera (optional)

This application allows the user to test cameras connected to the EVK. When launched, users see the GUI shown in Figure 27, where they can launch GStreamer pipelines.

Note: *The GStreamer windows do not have toolbars that allow users to close and move the window. To move windows, click anywhere in the video output area and drag the window to the correct area on the screen. The GStreamer windows are closed when the demo quits.*

The video test demo launches a GStreamer pipeline that plays a test source on the screen.



Once started, users may stop the video feed using the quit button in this window. Playback windows can be dragged with the mouse.

Figure 27. Video test demo

The following options are available to users while running the demo, as shown in Figure 27:

1. This button quits the demo and closes all GStreamer windows that are running at the current time.
2. These options allow users to control what the video output looks like:
   - **Source**: This option allows users to pick either a test source or a camera source to be shown in the output.
   - **Resolution**: This option allows users to change the video resolution of the output video.
   - **Scale to output**: This option allows users to scale the video up to fill the entire screen. Consider an example where 720x480 is selected as the resolution and the display is 1920x1080. In this example, first the video output is captured at or scaled down to keep the same format for consistency: "720x480". Then the output is scaled up to keep the same format for consistency: "1920x1080".
3. This button starts a video pipeline with the settings that the user has selected.

Note: *The current application only allows cameras to be used for a single video stream at a time. Also, ensure to select a supported resolution for the camera being used.*

3.2.1.2 Camera using the VPU

Required materials: Mouse, display output, and camera

This demo launches a GStreamer pipeline that encodes a video feed from a camera, then decodes, and displays the decoded video on the screen. The encoding and decoding are done using the on-chip Video Processing Unit (VPU).

3.2.1.3 Multi-camera preview

Required materials: Mouse, display output, OV5640 camera, and Basler camera

Supported cameras:

- Basler camera (`<EVKNAME>-evk-basler-ov5640.dtb`)

This demo launches a GStreamer pipeline that displays feed simultaneously from the two cameras on the screen.

3.2.2 Image signal processor demos

Some SoCs have built-in ISPs that allow on-chip image control similar to the one in a DSLR camera. The demos below show ways to use this powerful tool.

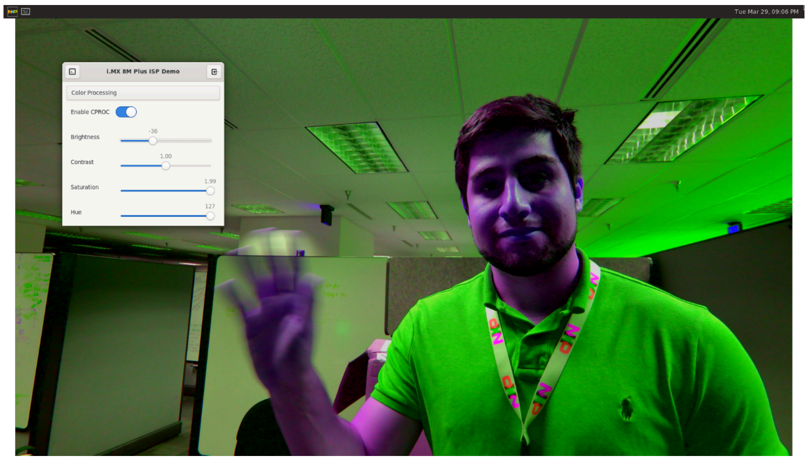
3.2.2.1 ISP control demo

Required materials: Mouse, display output, and supported camera

This demo launches a GStreamer pipeline that displays the current video output and a window that allows users to change and manipulate the video feed using API calls to the ISP.

Supported cameras:

- OS08A20 (`<EVKNAME>-evk-os08A20`)
- Basler camera (`<EVKNAME>-evk-basler.dtb`)



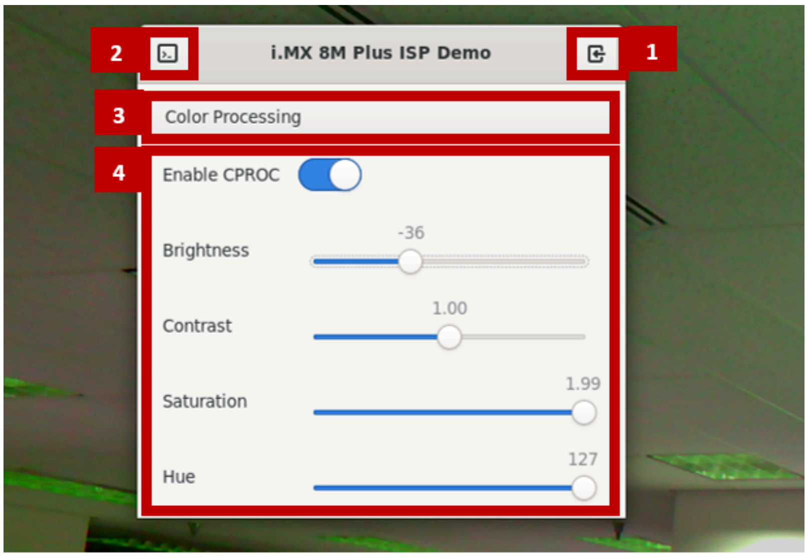
Enable CPROC: on
Brightness: -36
Contrast: 1.00
Saturation: 1.99
Hue: 127

Figure 28. ISP control demo example

The following options can currently be changed through the UI:

- Black level subtraction (red, green.r, green.b, blue)
- Dewarp
  - Dewarp on/off
  - Change dewarp mode
  - Vertical and horizontal flip
- FPS limiting
- White balance
  - Auto white balance on/off
  - White balance control (red, green.r, green.b, blue)
- Color processing
  - Color processing on/off
  - Color processing control (brightness, contrast, saturation, and hue)
- Demosaicing
  - Demosaicing on/off

  - Threshold control
- Gamma control
  - Gamma on/off
  - Gamma mode (logarithmic or equidistant)
- Filtering
  - Filter on/off
  - Filter control (denoise and sharpness)



Enable CPROC: on
Brightness: -36
Contrast: 1.00
Saturation: 1.99
Hue: 127

Figure 29. ISP control demo control panel

The ISP can be controlled using the control panel shown in Figure 29. This control panel appears after the video is started.

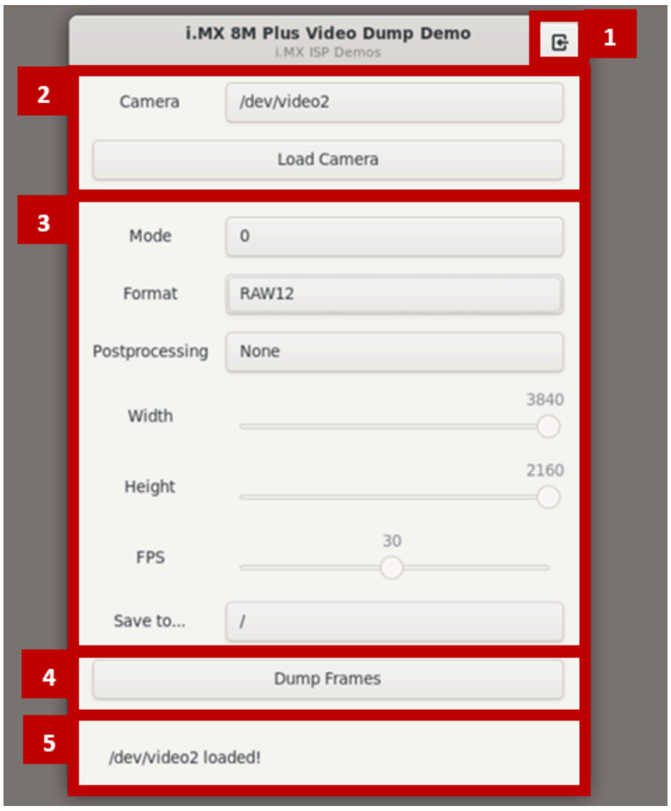
1. This button can be used to quit the demo and stop video playback.
2. This button displays an API call log for changes made in the control panel.
3. This dropdown allows the user to swap between different control panel sections.
4. This area includes the settings and options for the current section. Changes to these settings change the video feed playing in the background and then revert when the demo is exited.

3.2.2.2 Video dump demo

Required materials: Mouse, display output, external drive, and supported camera

Supported cameras:

- Basler camera (`<EVKNAME>-evk-basler.dtb`)



Camera: /dev/video2

Mode: 0

Format: RAW12

Postprocessing: None

Width: 3840

Height: 2160

FPS: 30

Save to...: /

/dev/video2 loaded!

Figure 30. Video dump example

The video dump demo allows users to dump raw camera frame data onto a connected drive. It also allows the user to set some settings before starting the frame dump process:

1. This button can be used to quit the demo.
2. This area selects the camera to use and loads it into the application. To load the camera, select the camera from the **Camera** dropdown and click **Load Camera**. It gets the supported modes and tests whether the camera can be run with this demo. Once the camera is loaded in, the other options in this window unlock. If the camera cannot be loaded, an error pop-up occurs.
3. Here, users can set the settings for the frames that get saved onto the drive:
   - **Mode**: Select the mode that the camera is in. It typically changes the height, width, FPS, or other camera features like HDR.
   - **Format**: Select the format that the raw frame data is saved in. The options are RAW12, RAW10, RAW8, NV16, NV12, and YUYV.
     Note: *All formats cannot be supported for all cameras.*
   - **Postprocessing**: If modifications are supported, the user sets modifications to the image. "Crop" crops a frame while "scale" scales the frame to the selected size.
   - **Width**: Shows the width of the selected mode. If postprocessing is enabled, it allows users to change the width of an image.
   - **Height**: Shows the height of the selected mode. If postprocessing is enabled, it allows users to change the height of an image.
   - **FPS**: Frames per second of a selected mode. It shows how quickly frames are dumped into storage.
   - **Save to... (it missed one)**: The user can pick a save location for the raw frame data.
4. This button starts the process of getting raw frame data and dumping it to the selected save location. Clicking this button again after the process is started causes the program to stop dumping frames and saves the data.

5. The status bar indicator shows updates on what is happening currently with the demo.
   Note: *Raw image data are large files that most image viewers cannot interpret. It is not a way to save processed images onto a drive.*
   Warning: *Do not disconnect the drive early! Disconnecting the drive early can result in the data saved onto the drive becoming corrupted.*

3.2.3 2Way video streaming

Required materials: Mouse, display, and camera for each device. Two devices that are connected to the same network are required.

The demo features a video stream between two i.MX devices such as i.MX 8M Plus and i.MX 8M Mini, connected to the same local network. Each device in the network is allowed to multicast its IP address, and search for other local devices at the same time. It is possible to gather more than one i.MX device and the user is given an option to pick one from the list. After a connection is established, the two devices are able to start a video stream with one video overlaying another.

Once the demo starts, the user gets an option to enter a name unique to their device.

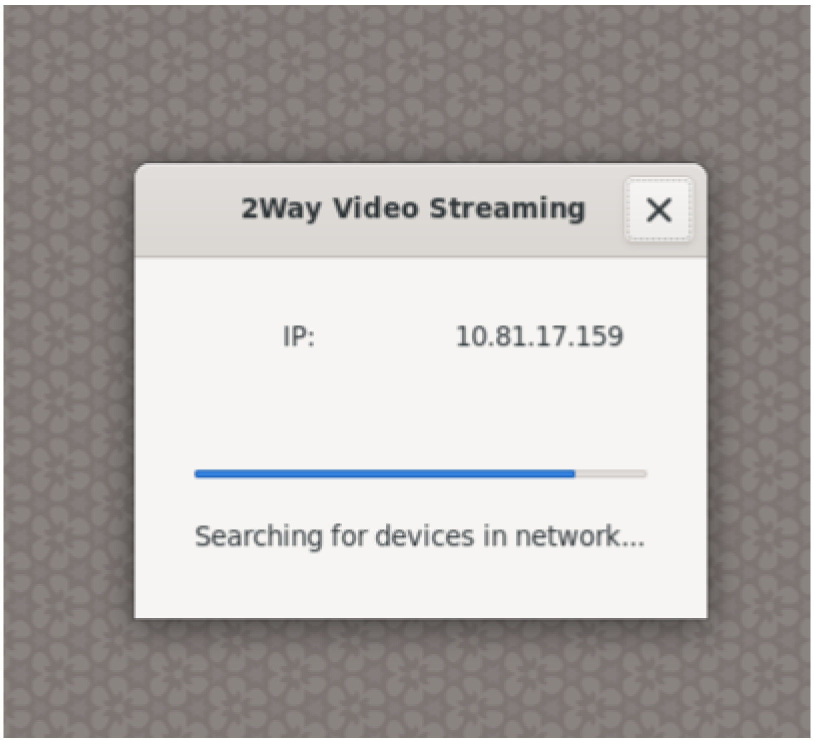


Figure 31. Status during device search in local network

On each device, a discovery and response module run in the background to identify and connect i.MX devices. When the search is complete, the devices are listed in the dropdown along with multiple cameras (if any). During the initial setup check, the demo also runs a camera device connect check to ensure that a valid camera device is connected. It then adds them to the dropdown list. If the desired device is not listed in the dropdown during the initial search, users can click the **Search Again** to search for a targeted device on the network.

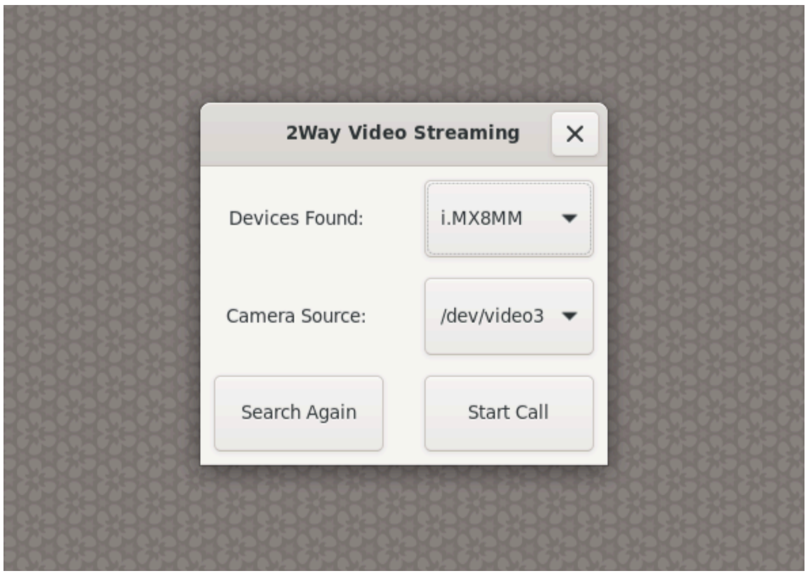


Figure 32. Menu for 2Way Video Streaming

When both the EVKs identify the two targeted i.MX devices, start a video stream between them. Figure 33 shows an example for video stream between two i.MX devices in the local network.

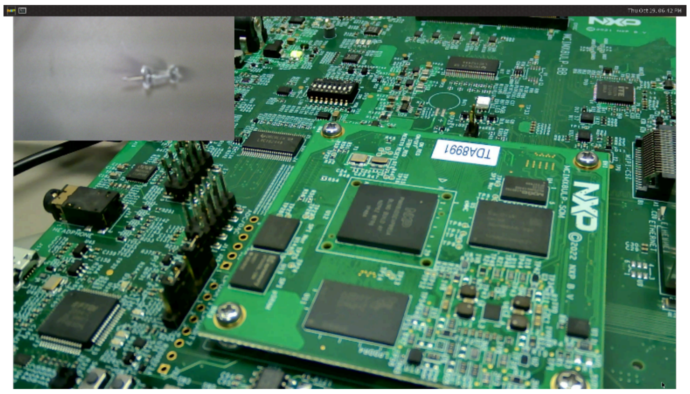


Figure 33. Video stream from 2Way Video Streaming

3.2.4 Audio

A few audio demos have also been included to test audio equipment. These tests have no graphical output.

3.2.4.1 Audio record

Required materials: Mouse, display, and microphone

This test records an audio file from the headphone input with a 10-second duration.

3.2.4.2 Audio play

Required materials: Mouse, display, and headphones

This test plays the audio file recorded on the audio record test.

Note: *This demo only works if the audio record is run first.*

3.2.5 Voice

Similar to a mouse and keyboard, voice is a way for us to interact with edge devices. These demos show off the voice technologies of NXP such as VoiceSeeker, VoiceSpot, and Voice Intelligent Technology (VIT).

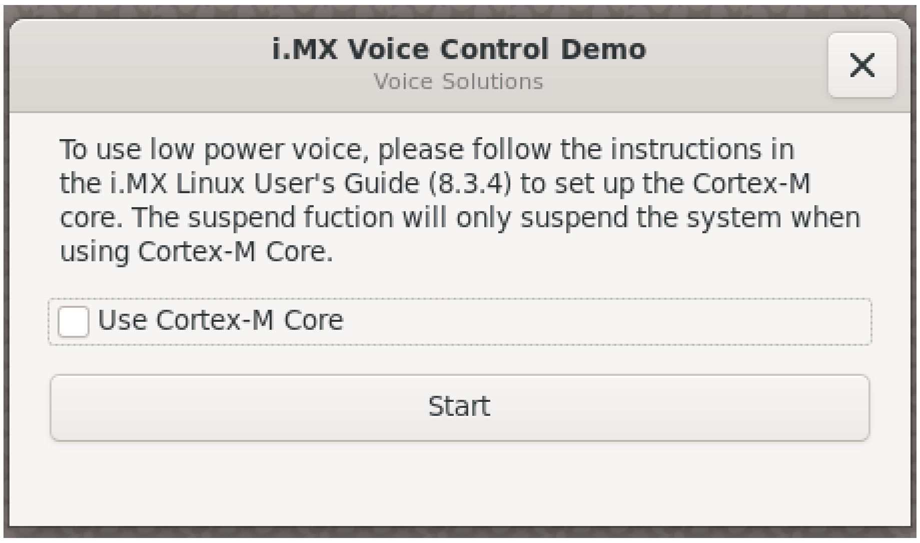
Note: *Attach a valid 8-microphone array board (8MIC-RPI-MX8) and set-up these demos for correct usage. To acquire one, see 8MIC-RPI-MX8.*

3.2.5.1 i.MX voice control

Required materials: Mouse, display, and 8MIC-RPI-MX8

The user can control GoPoint for i.MX Applications Processors with the power of their voice. This demo uses the trial version of VoiceSpot (locked to "Hey NXP!" as the wake word) and VoiceSeekerLite (no echo cancelation) to open and close some of the demos in GoPoint for i.MX Applications Processors. Voice commands are all processed locally, meaning no cloud service is required. For some boards, such as the i.MX 8M Mini and i.MX 8M Plus, a DTB file must be selected before the microphone can work.

Note: *This demo changes the file at `/etc/asound.conf`. If the demo closes normally, the demo attempts to undo the changes. If the demo crashes or terminates via another method rather than clicking the cross icon at the top-right corner of the window, the user can retrieve the original file from `/etc/asound_org.conf`. This file contains the settings that have been applied during the last successful run of the demo.*

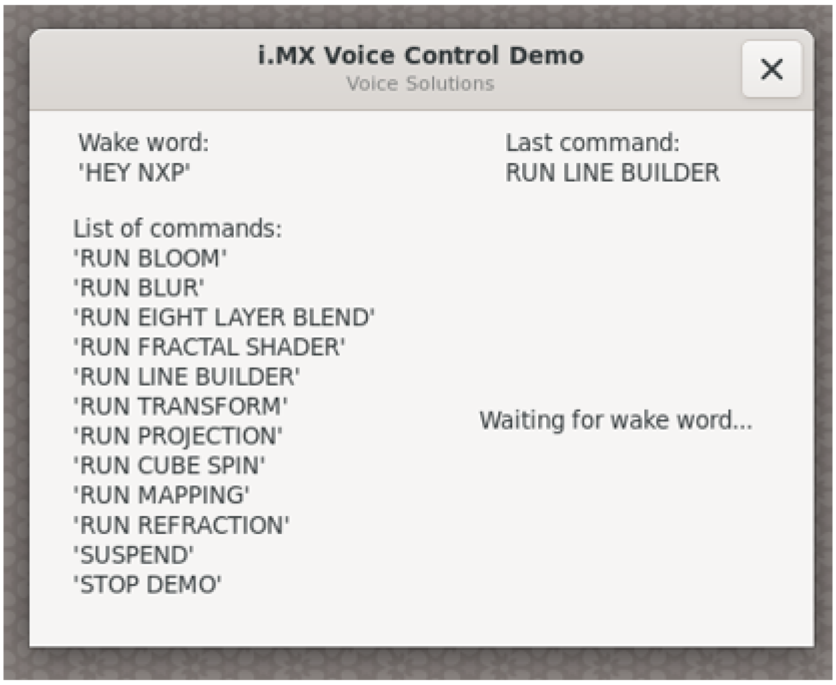


To use low power voice, please follow the instructions in the i.MX Linux User's Guide (8.3.4) to set up the Cortex-M core. The suspend fuction will only suspend the system when using Cortex-M Core.

Figure 34. i.MX Voice Control

The first window asks if the user likes to use the Cortex-M core for low-power voice operations. To set up this configuration, refer to the section "Cortex-M Image" in the *i.MX Linux User's Guide* (document IMXLUG) so that low-power voice is running on the Cortex-M core.

Note: *These settings are not enabled by default and the application cannot check whether the Cortex-M core is running correctly. Enabling this setting without the proper setup can require the user to manually power cycle the EVK.*



Wake word:
'HEY NXP'

Last command:
RUN LINE BUILDER

List of commands:
'RUN BLOOM'
'RUN BLUR'
'RUN EIGHT LAYER BLEND'
'RUN FRACTAL SHADER'
'RUN LINE BUILDER'
'RUN TRANSFORM'
'RUN PROJECTION'
'RUN CUBE SPIN'
'RUN MAPPING'
'RUN REFRACTION'
'SUSPEND'
'STOP DEMO'

Waiting for wake word...

Figure 35. i.MX Voice Control main window

The demo first attempts some setup steps before it starts listening for commands. If there are any errors, the setup stops, and the error displays. Once the demo is successfully set up, the wake word and the list of commands get listed on the left-hand side of the window.

To wake up the board, say "Hey NXP!". When the status indicator switches to "Listening…", say the command that you wish to run. The demo is launched (can take a couple of seconds). To stop the demo, say "Hey NXP!" and say another command or "Stop Demo".

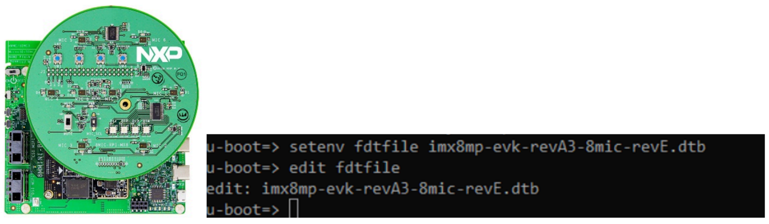
When the Cortex-M core setting is enabled, saying “Suspend” causes the EVK to put the Cortex-A core into Suspend mode. As a result, the screen turns black and the mouse and keyboards become unresponsive. To take the Cortex-A core out of Suspend mode, simply say “Hey NXP!” and the screen and controls must reappear.

3.2.5.2 i.MX Multimedia Player

Required materials: Mouse, display, and 8MIC-RPI-MX8

i.MX Multimedia Player is a demo application based on NXP VIT. VIT is a free library that provides voice recognition and integrates a complete audio front end, wake word engine, and voice-command solution to control IoT devices.

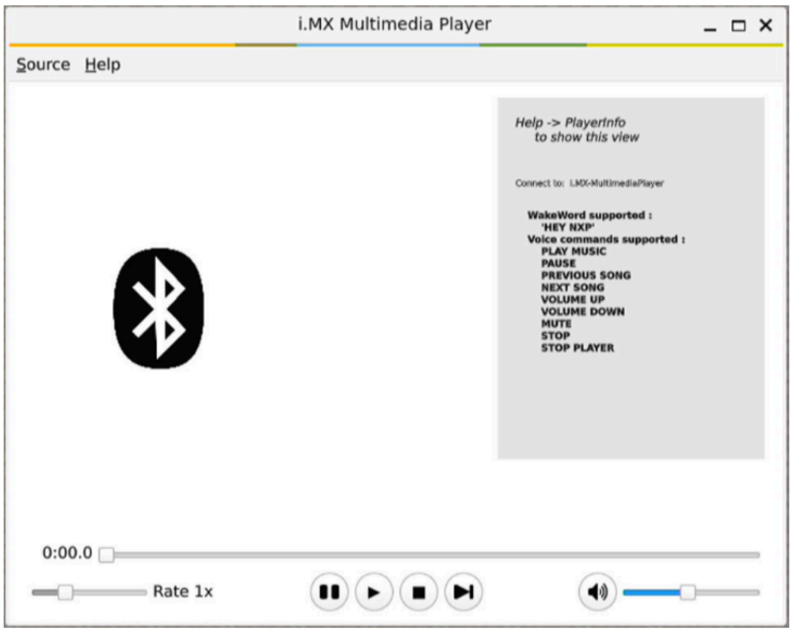
The multimedia player application uses Bluetooth to play back audio and controls Bluetooth commands using VIT voice commands. The application requires the 8MIC-RPI-MX8 installed on the i.MX hardware for voice enablement. The 8MIC-RPI-MX8 requires to set the FDTFILE to a proper 8-microphone board revision DTB in the U-Boot environment. For details, refer 8MIC-RPI-MX8.



```
u-boot=> setenv fdtfile imx8mp-evk-revA3-8mic-revE.dtb
u-boot=> edit fdtfile
edit: imx8mp-evk-revA3-8mic-revE.dtb
u-boot=> 
```

Figure 36. 8MIC-RPI-MX8 module and required DTB

To run the demo, launch i.MX Multimedia Player from GoPoint for i.MX Applications Processors GUI. It takes a few seconds to load the application and dependencies. Once i.MX Multimedia Player is launched, use a smartphone or a tablet to search for the "i.MX-MultimediaPlayer" Bluetooth device and pair it.

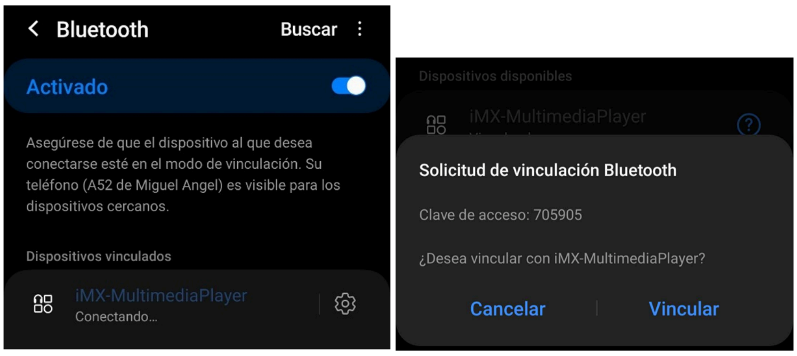


```
Help -> PlayerInfo
    to show this view

Connect to:  i.MX-MultimediaPlayer

  WakeWord supported :
     'HEY NXP'
  Voice commands supported :
     PLAY MUSIC
     PAUSE
     PREVIOUS SONG
     NEXT SONG
     VOLUME UP
     VOLUME DOWN
     MUTE
     STOP
     STOP PLAYER
```

Figure 37. i.MX Multimedia Player in GoPoint for i.MX Applications Processors GUI



Asegúrese de que el dispositivo al que desea conectarse esté en el modo de vinculación. Su teléfono (A52 de Miguel Angel) es visible para los dispositivos cercanos.

Dispositivos vinculados

iMX-MultimediaPlayer
Conectando...

Dispositivos disponibles

iMX-MultimediaPlayer

Solicitud de vinculación Bluetooth

Clave de acceso: 705905

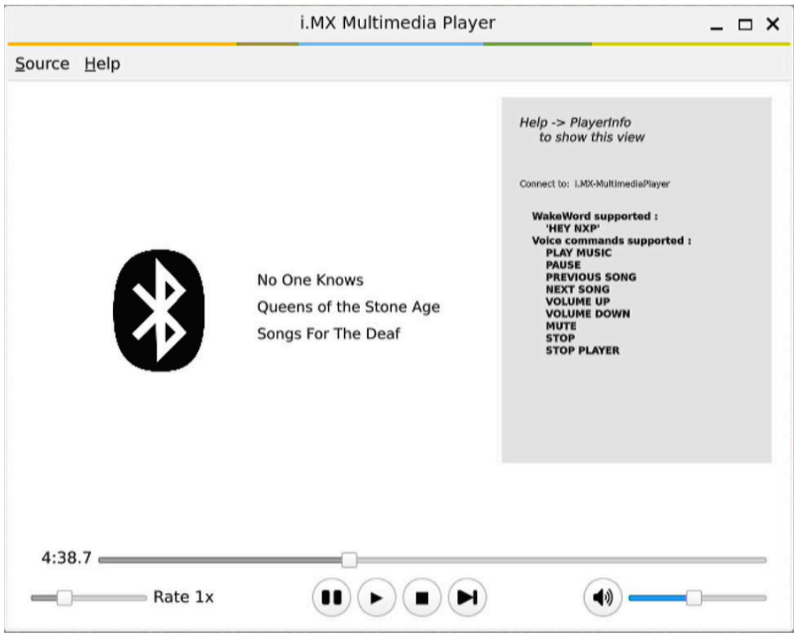
¿Desea vincular con iMX-MultimediaPlayer?

Figure 38. Bluetooth device

Once a Bluetooth device is connected, the user can start controlling audio playback using voice commands. Use the "Hey NXP!" wake word to let VIT wake up the device and then use a voice command to control the audio playback. When a Bluetooth connection is established, the user can see audio metadata displayed in the demo GUI. The user can control the audio using the "Pause, Next Song, Previous Song, Mute, Volume Up, Volume Down, and Stop" voice commands. To stop the demo, use the "Stop Player" command.

The following voice commands are supported:

- Play Music
- Pause
- Previous Song
- Next Song
- Volume Up
- Volume Down
- Mute
- Stop
- Stop Player



No One Knows
Queens of the Stone Age
Songs For The Deaf

Help -> PlayerInfo
to show this view

Connect to: i.MX-MultimediaPlayer

WakeWord supported :
'HEY NXP'
Voice commands supported :
PLAY MUSIC
PAUSE
PREVIOUS SONG
NEXT SONG
VOLUME UP
VOLUME DOWN
MUTE
STOP
STOP PLAYER

Figure 39. Voice commands

3.2.5.3 Smart Kitchen

Required hardware: Mouse, display, and microphone.

This application emulates an interactive kitchen through a GUI controlled by voice commands. The GUI is based on the Little Versatile Graphic Library (LVGL) and NXP VIT supports the voice commands.

To configure Smart Kitchen, connect the mouse and display to the board in the corresponding port, depending on the display, mouse, and board being used. USB-A to USB-C and MIPI-HDMI adapters are possibly needed.

The 8MIC-RPI-MX8 is needed for the i.MX 8MM and i.MX 8MP boards. For this case, connect the 8MIC-RPI-MX8 board to the EVK as described in the *i.MX 8MIC-RPI-MX8 Board Quick Start Guide* (document IMX-8MIC-QSG). Start the EVK and stop the boot process. To enable the microphone board, set the corresponding device tree in U-Boot:

For i.MX 8MM:

```
u-boot=> edit fdtfile
edit: imx8mm-evk-8mic-revE.dtb
u-boot=> saveenv
```

```
u-boot=> boot
```

For i.MX 8MP:

```
u-boot=> edit fdtfile
edit: imx8mp-evk-revA3-8mic-revE.dtb
u-boot=> saveenv
u-boot=> boot
```

The i.MX 93 EVK has its own microphones integrated in the board. Therefore, no microphone board and special-device tree are needed to enable them.

However, if an LVDS display panel is used, changing the device tree can be needed. The `fdtfile` variable must be edited as follows:

```
u-boot=> edit fdtfile
edit: imx93-11x11-evk-boe-wxga-lvds-panel.dtb
u-boot=> saveenv
u-boot=> boot
```

Note: *The device tree names can vary between EVK or 8MIC board versions. Verify that the right name is used before editing the* `fdtfile` *variable.*

To run a Smart Kitchen, follow the guide below.

The emulated kitchen has three interactive items that can perform some animations driven by voice commands as follows:

- A hood
- An oven
- An air conditioner

The wake word and a set of commands are specific for each kitchen item.

When the configuration is done, launch the Smart Kitchen application using GoPoint for i.MX Applications Processors GUI. The Smart Kitchen GUI appears on the screen, as shown in Figure 40.

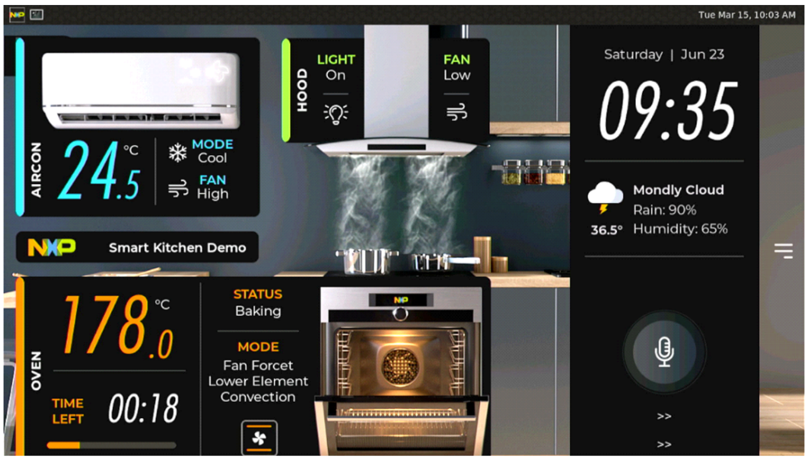


Figure 40. Smart Kitchen GUI

Now, the application is ready to receive voice commands.

Note: *Every voice command must be said after saying a wake word. If no wake word is said, the commands are not recognized. After a wake word is detected, only one command can be said.*

The wake words and commands supported are as follows:

- Wake words:
  - Hey hood
  - Hey oven
  - Hey aircon
- Common commands (work with any wake word):
  - ENTER
  - EXIT
  - RUN DEMO
  - STOP DEMO
- Hood commands:
  - FAN OFF
  - FAN ON
  - FAN LOW
  - FAN HIGH
  - LIGHT OFF
  - LIGHT ON
- Air conditioner commands:
  - DRY MODE
  - COOL MODE
  - FAN MODE
  - SWING OFF
  - SWING ON
  - FAN LOW
  - FAN HIGH
- Oven commands:
  - CLOSE DOOR
  - OPEN DOOR

The ENTER command zooms to the corresponding item to see the animations performed by other commands with more details. The EXIT command zooms out, returning to the initial view. Though the ENTER command emphasizes the animations of an item, some animations do not need them strictly.

The STOP DEMO command kills the GUI but keeps the VIT running in the background. Therefore, it can be used to pause the demo for some time. The GUI can be launched again using the RUN DEMO command.

The demo can be stopped by clicking the top-right corner of the window or using the `~/.nxp-demo-experience/scripts/multimedia/smart-kitchen/kill.sh` script.

3.2.5.4 i.MX E-Bike VIT

Required materials: Mouse, display, and 8MIC-RPI-MX8

i.MX 8MM and 8MP need extra microphones (i.MX 8MIC-RPI-MX8)

i.MX E-Bike VIT showcases the multimedia capabilities of i.MX Application Processors to emulate an interactive e-bike through a GUI controlled by voice commands. The GUI is based on the Little Versatile Graphic Library (LVGL) and NXP VIT supports voice commands.

3.2.5.4.1 Implementation using VIT and LVGL

Note: *Figure 41 is simplified and does not represent the complete e-bike VIT implementation. Some elements have been omitted and only the key elements are shown.*

The user gives voice commands to the EVK through wake words and pre-defined commands. NXP AFE module filters the voice sound and sends it to the Voice UI application module, which makes the system call the Python script. This Python application detects the wake word and commands provided. Then, it allocates them in the Message Queue attended by the e-bike VIT GUI application, finally showing the corresponding animation.

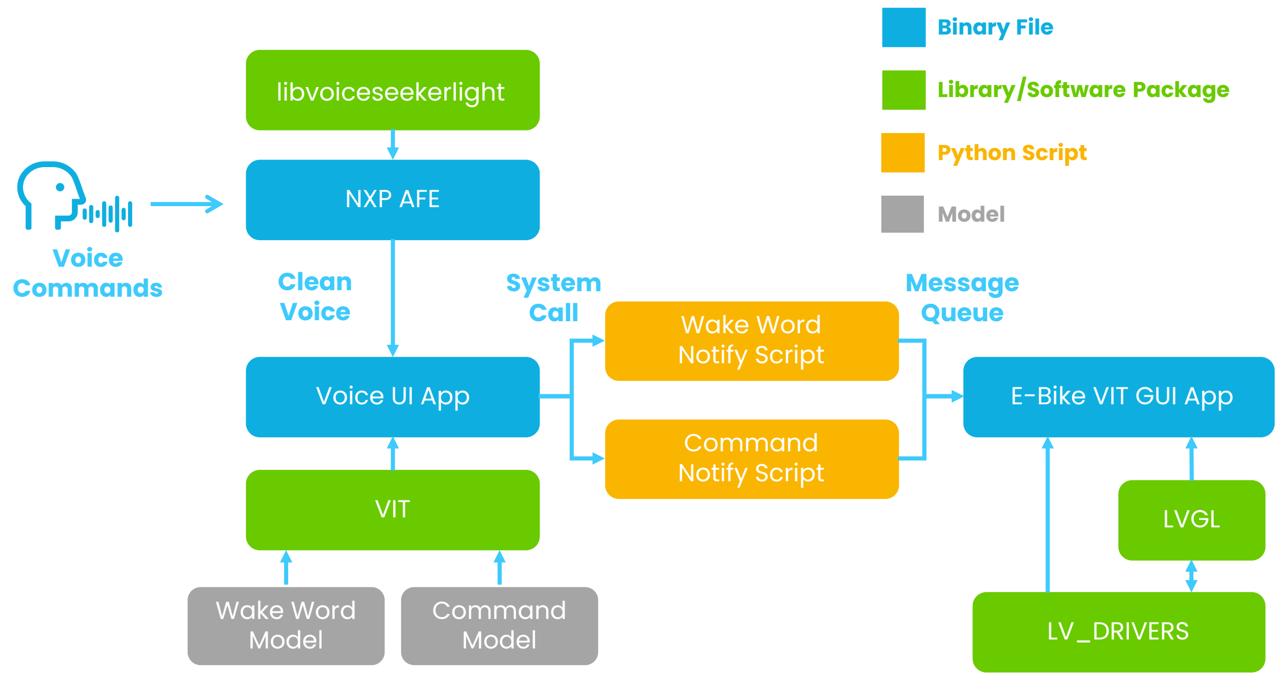


Figure 41. Voice pipeline for an example application

The components used in this application are as follows:

1. NXP AFE
   - NXP AFE is used as a feed for VIT voice recognition.
   - It helps to clean noise and echo by using the source and a reference of the speaker.
   - With NXP AFE, it is possible to have a clear single channel microphone audio that can be used for processing.
2. Voice UI app
   - This application uses the model to detect wake words and commands.
   - This binary file enables the connection of the wake word Model and Command Model created on VIT with the Python scripts `WakeWordNotify` or `WWCommandNotify` to detect and differentiate the system triggers.
3. E-Bike VIT GUI app
   - GUI allows the user to see a graphic representation of e-bike that gets the wake words and the commands.
   - This app shows the e-bike page switch corresponding to the wake words and commands given by the user.
   - To create a high-quality display, it uses the LVGL graphics library.

Once started, the UI must be displayed as shown in Figure 42, Figure 43, and Figure 44.

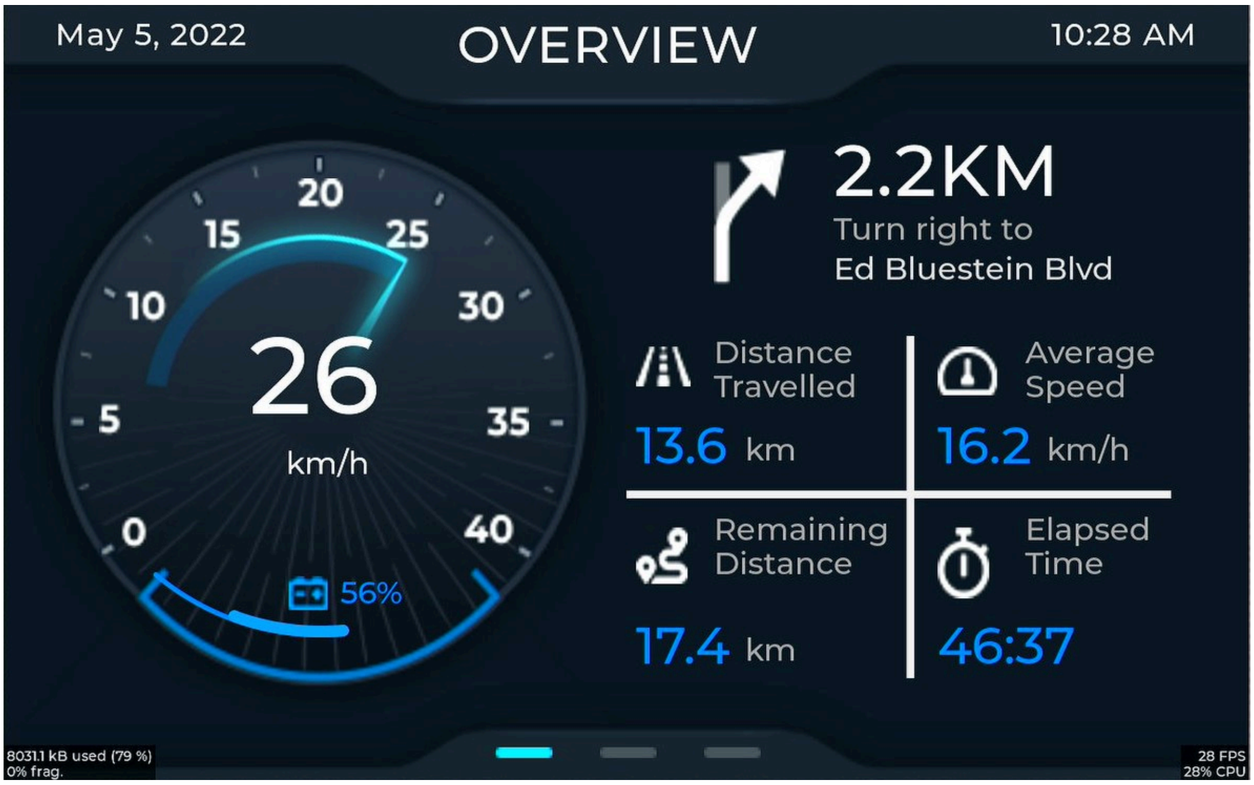


Figure 42. E-Bike screen 1

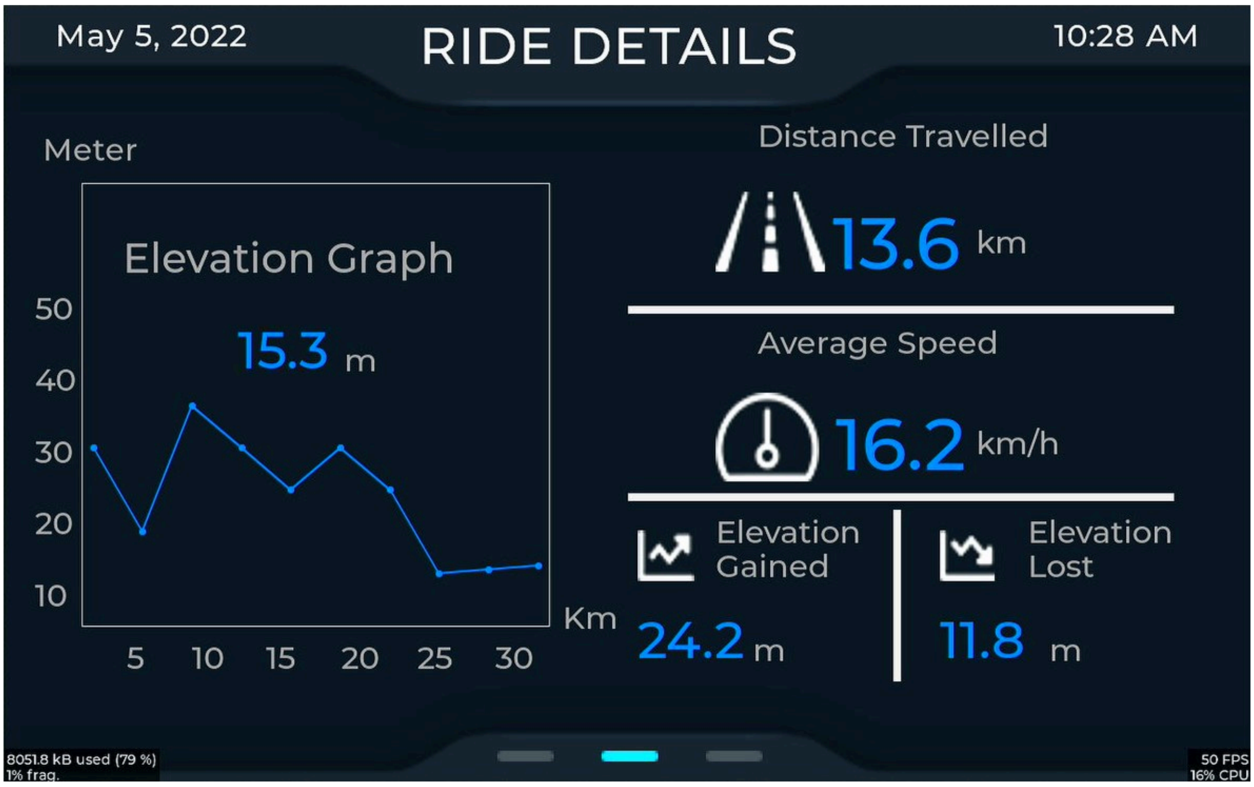


Figure 43. E-Bike screen 2

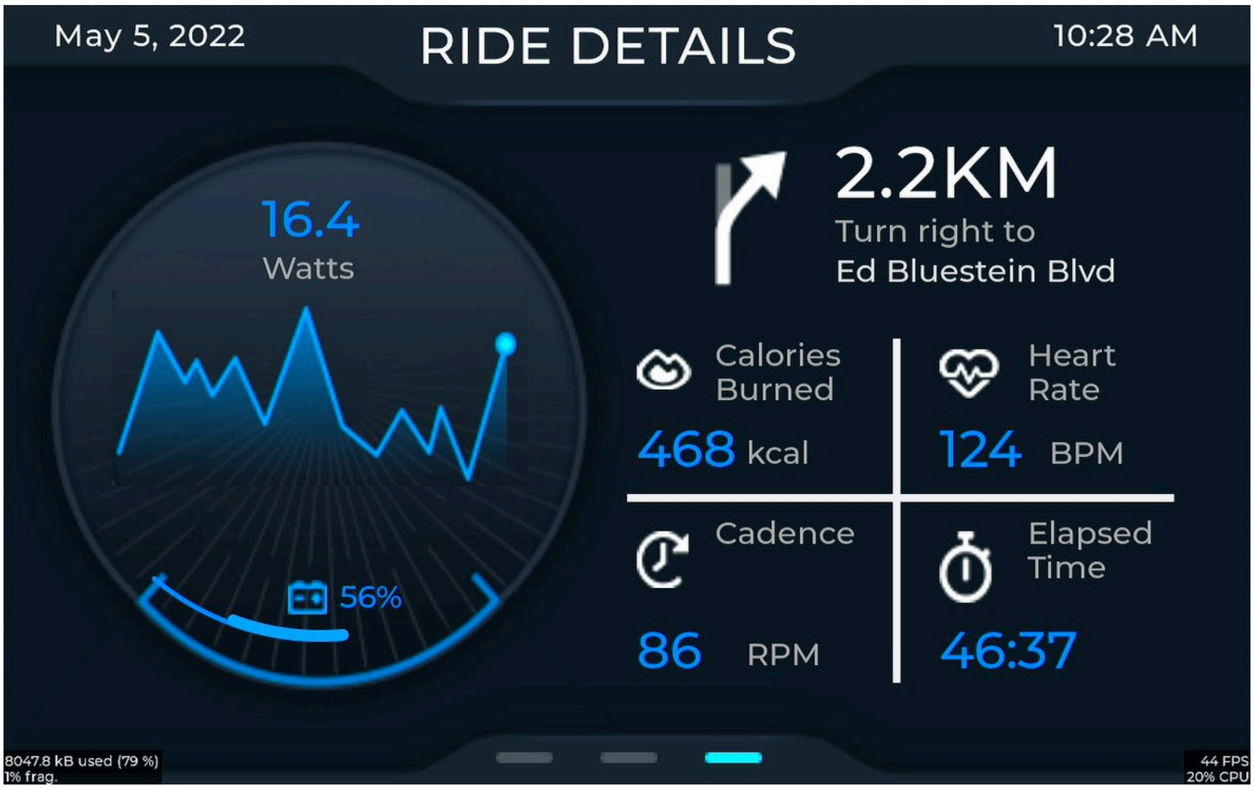


Figure 44. E-Bike screen 3

3.2.5.4.2 Usage

The VIT voice command interface works as follows:

1. Say a wake word first to wake up the e-bike control.
2. Then, say the command that you want to execute.

Table 2 lists the wake words and commands supported.

Table 2. Supported wake words and commands

Category	Supported
Wake words	• HEY NXP • HEY E Bike
Voice commands	*General commands (work with any wake word)* • NEXT PAGE • LAST PAGE • RUN DEMO • STOP DEMO

3.3 TSN 802.1 Qbv Demo

Required materials: Ethernet cables, USB camera, and display

This demo showcases the TSN 802.1Qbv Time Aware Scheduling (TAS) standard with two traffic classes as follows:

- *iperf* to emulate best-effort traffic

- *camera streaming* to emulate high priority time-sensitive traffic

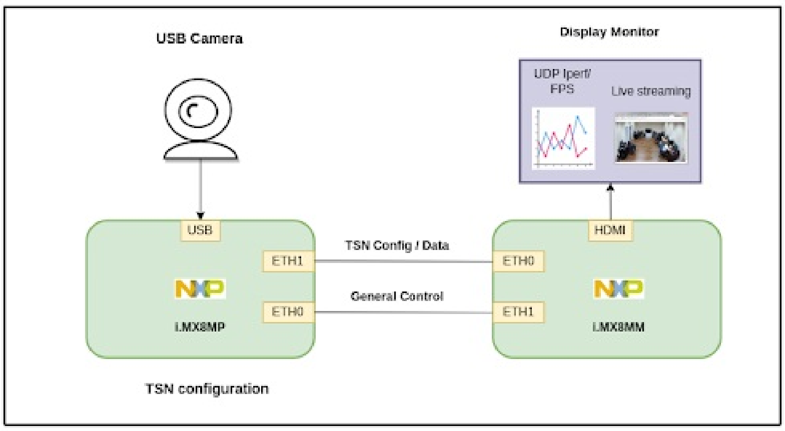


Figure 45. Setup diagram for TSN Qbv

Traffic is streamed between the i.MX 8MP and the i.MX 8MM boards to demonstrate configurable length and repeating Qbv gate schedules.

Test Steps:

- When the demo is launched, there is a short loading period as shown in Figure 46

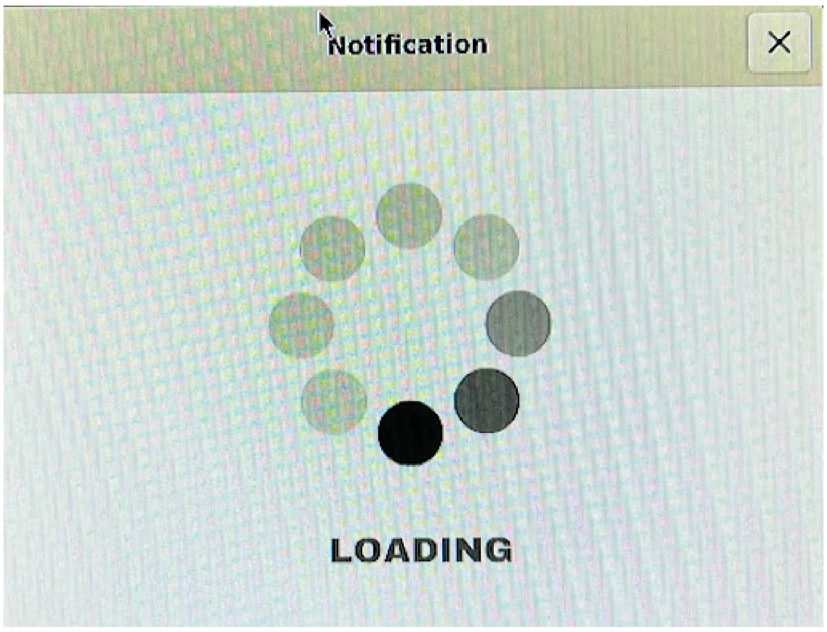
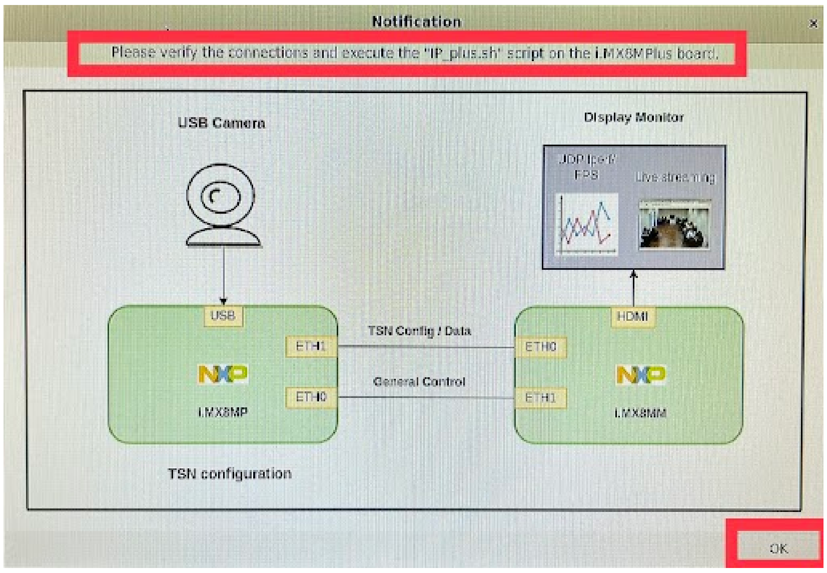


Figure 46. Loading gif

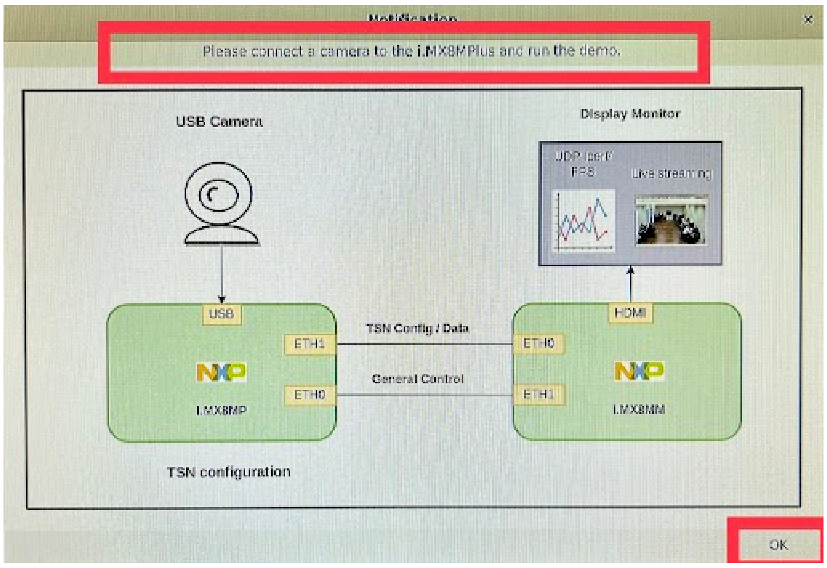
- If the connections are not successful, a pop-up window shows up, as shown in Figure 47. After confirming the connections and system setup, click the **OK** button and relaunch the demo when a camera is connected.



Please verify the connections and execute the "IP_plus.sh" script on the i.MX8MPlus board.

Figure 47. Pop-up window

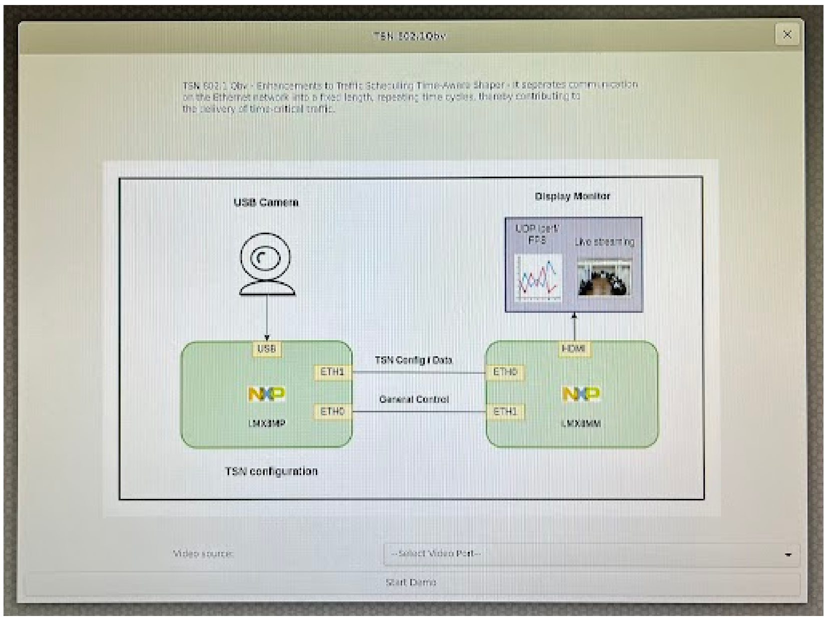
- If Ethernet connections are successful and the camera is not connected, the window shown in the Figure 48 displays. To get back to the Launch demo window, click the **OK** button.



Please connect a camera to the i.MX8MPlus and run the demo.

Figure 48. Click the OK button

- If the connections are successful, the TSN Qbv Demo login window shows up, as shown in Figure 49.

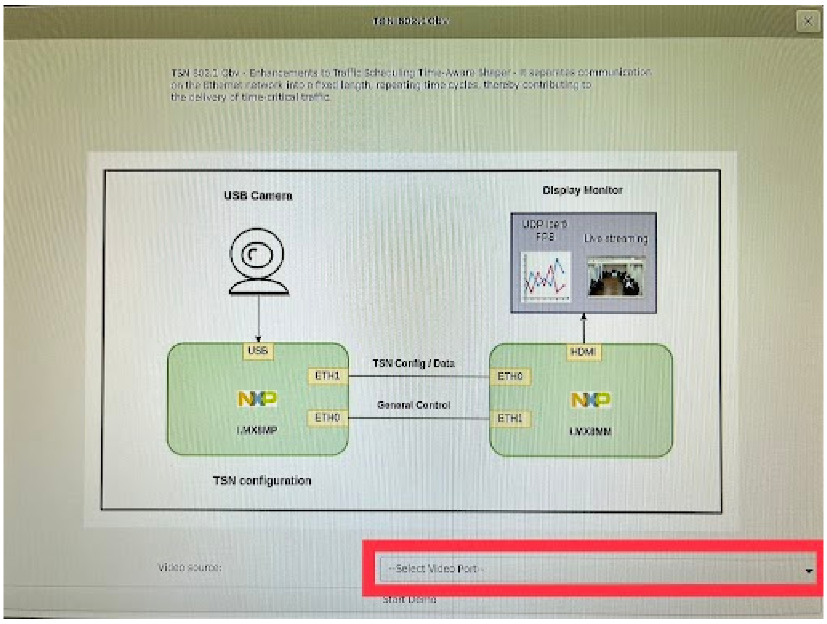

TSN 802.1Qbv

TSN 802.1 Qbv - Enhancements to Traffic Scheduling Time-Aware Shaper - It separates communication on the Ethernet network into a fixed length, repeating time cycles, thereby contributing to the delivery of time-critical traffic.

Video source: --Select Video Port--

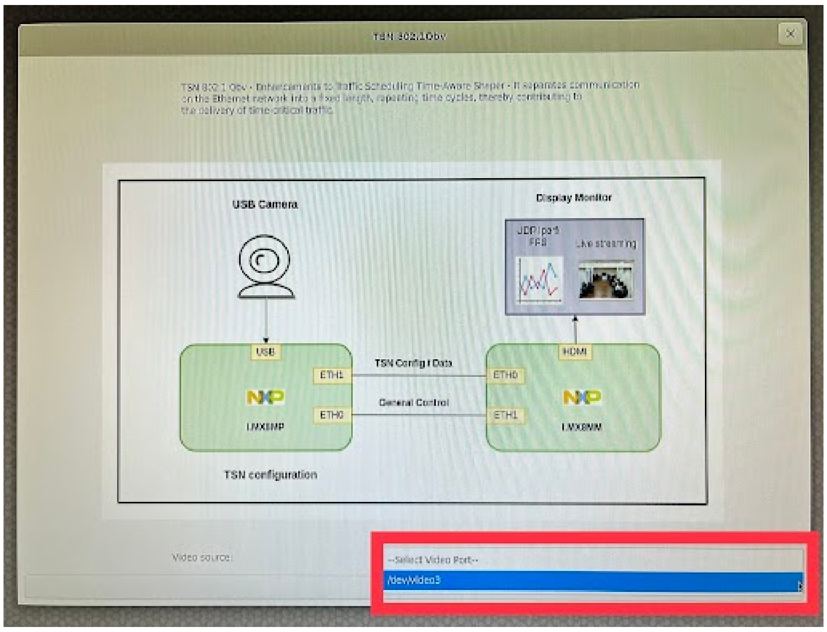
Figure 49. TSN Qbv Demo login window

- The **Video source** and **Start Demo** buttons are in the login window. After selecting the video source (as shown in Figure 50), the **Start Demo** button is enabled.


TSN 802.1Qbv

TSN 802.1 Qbv - Enhancements to Traffic Scheduling Time-Aware Shaper - It separates communication on the Ethernet network into a fixed length, repeating time cycles, thereby contributing to the delivery of time-critical traffic.

Video source: --Select Video Port--

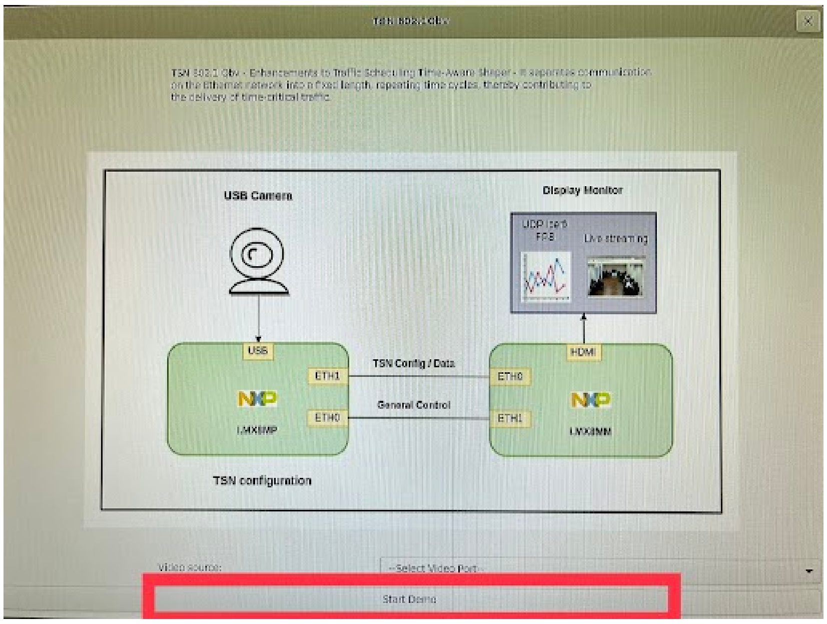


TSN 802.1Qbv

TSN 802.1 Qbv - Enhancements to Traffic Scheduling Time-Aware Shaper - It separates communication on the Ethernet network into a fixed length, repeating time cycles, thereby contributing to the delivery of time-critical traffic.

Figure 50. Video source button and dropdown menu

- If a camera is connected, the available video sources are displayed in a dropdown menu. To access the camera feed, select the desired video source.
- After selecting the video source, click the **Start Demo** button to start the demo. After clicking the **Start Demo** button, the "loading gif" window shows up.



TSN 802.1Qbv

TSN 802.1 Qbv - Enhancements to Traffic Scheduling Time-Aware Shaper - It separates communication on the Ethernet network into a fixed length, repeating time cycles, thereby contributing to the delivery of time-critical traffic.

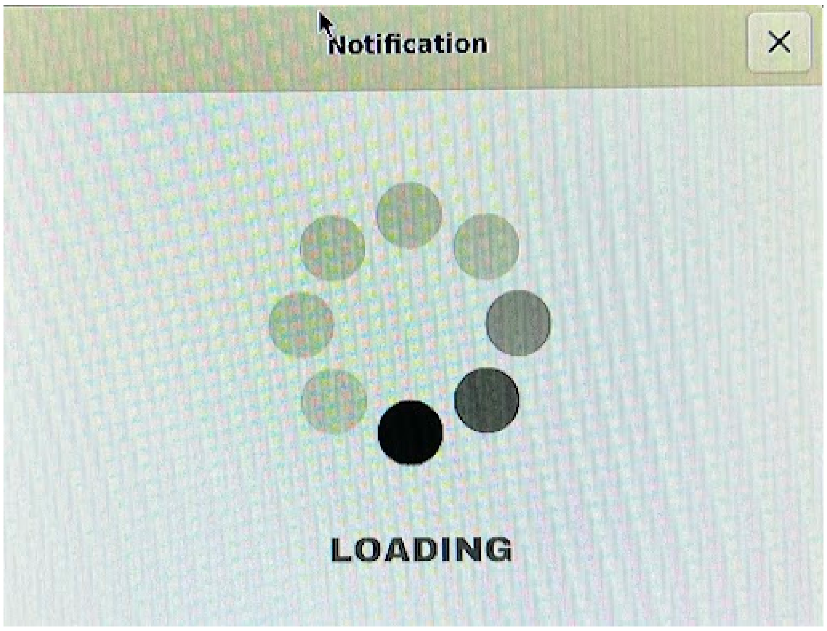


Figure 51. Start Demo button and window for loading gif

- The **Start Demo** button turns into **Stop Demo** and behavior must change accordingly.
- After a successful start of the demo, the default graph window and camera window show up, as shown in Figure 52.

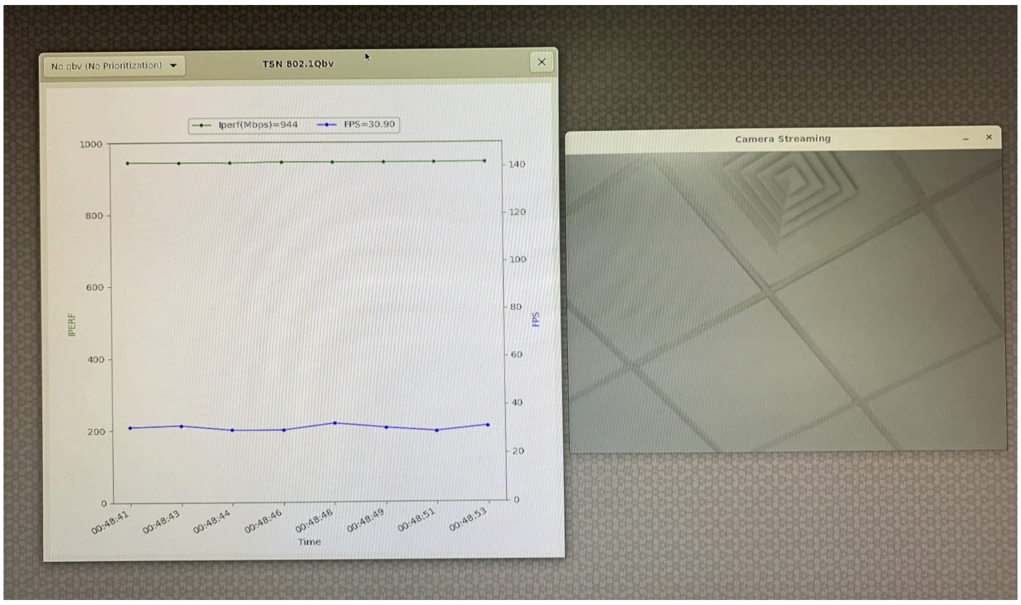


Figure 52. Default graph window and camera window

- By default, the demo starts with **No qbv** configuration. The iperf throughput ranges between 900 Mbit/s to 950 Mbit/s and the camera streams without glitches.
- To change the TSN Qbv configuration, select the dropdown by clicking the configuration dropdown button. After clicking the button, you can see the UI (Figure 53).

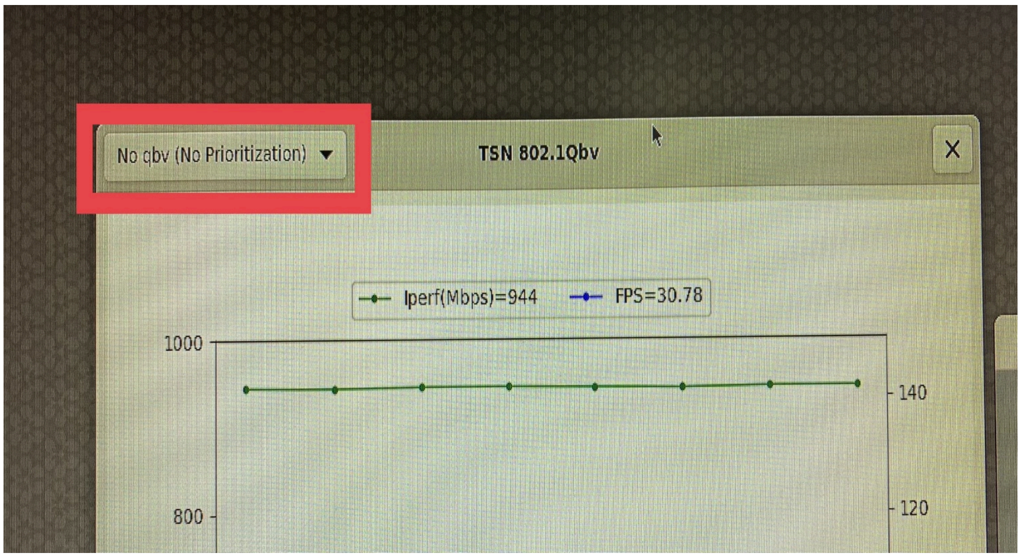


Figure 53. Configuration dropdown button

- Click the **Qbv1 (Video Prioritization)** configuration button.

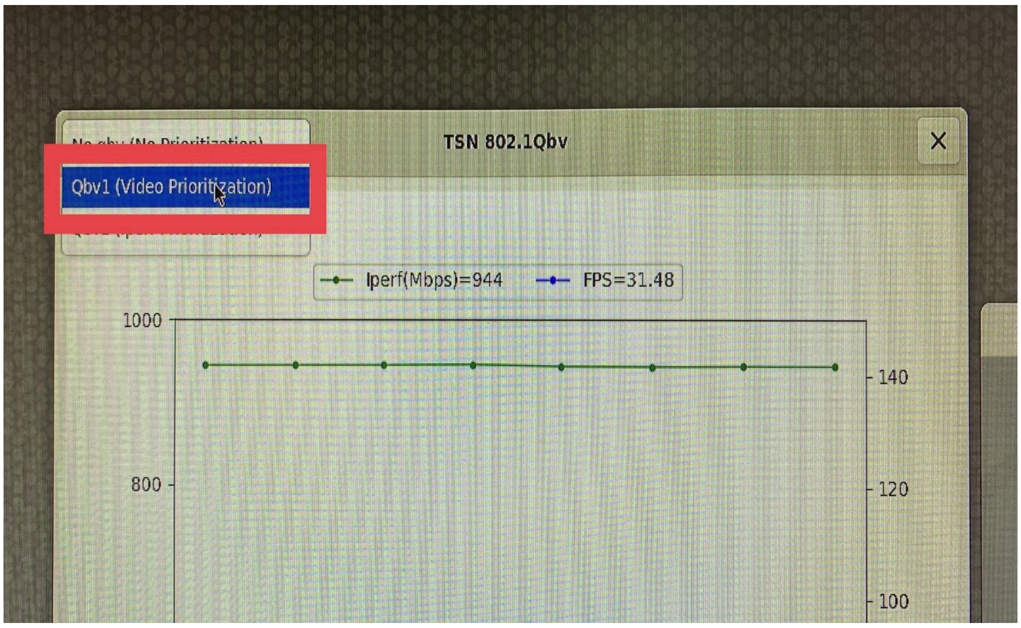
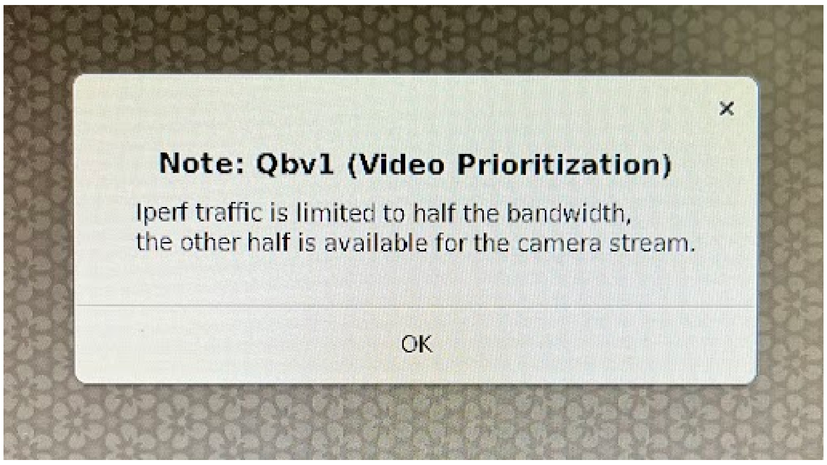


Figure 54. Qbv1 (Video Prioritization) configuration button

- After clicking the **Qbv1 (Video Prioritization)** button, as shown in Figure 54, the pop-up window shown in Figure 55 appears.



Note: Qbv1 (Video Prioritization)

Iperf traffic is limited to half the bandwidth,
the other half is available for the camera stream.

OK

Figure 55. About Qbv1 (Video Prioritization) pop-up window

- To continue the demo, click the **OK** button.

Expected results:

- The iperf limits with throughput of 450 Mbit/s to 500 Mbit/s and camera FPS values must be 25 FPS to 30 FPS, as shown in Figure 56.
- The iperf traffic is limited to a half of the line rate and the other half of the line rate is available for camera traffic. The camera is therefore able to stream at its maximum rate, which, in this example, is much less than 500 Mbit/s.

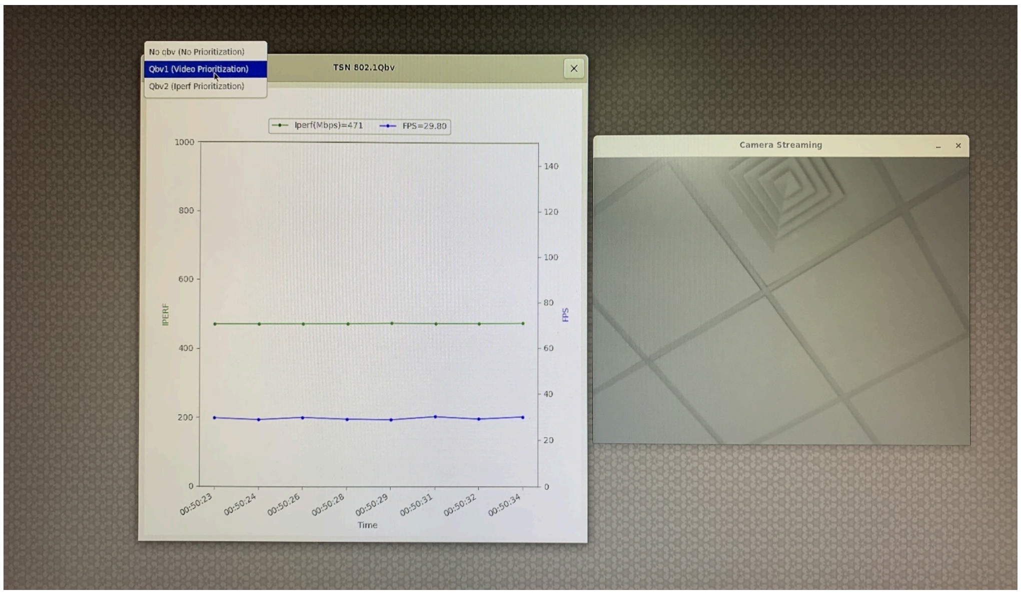

No qbv (No Prioritization)
Qbv1 (Video Prioritization)
Qbv2 (Iperf Prioritization)
TSN 802.1Qbv
Iperf(Mbps)=471
FPS=29.80
IPERF
FPS
Time
Camera Streaming

Figure 56. Qbv1 graph window and camera window

- To select the Qbv2 configuration, click the **Qbv2 (Iperf Prioritization)** configuration button.

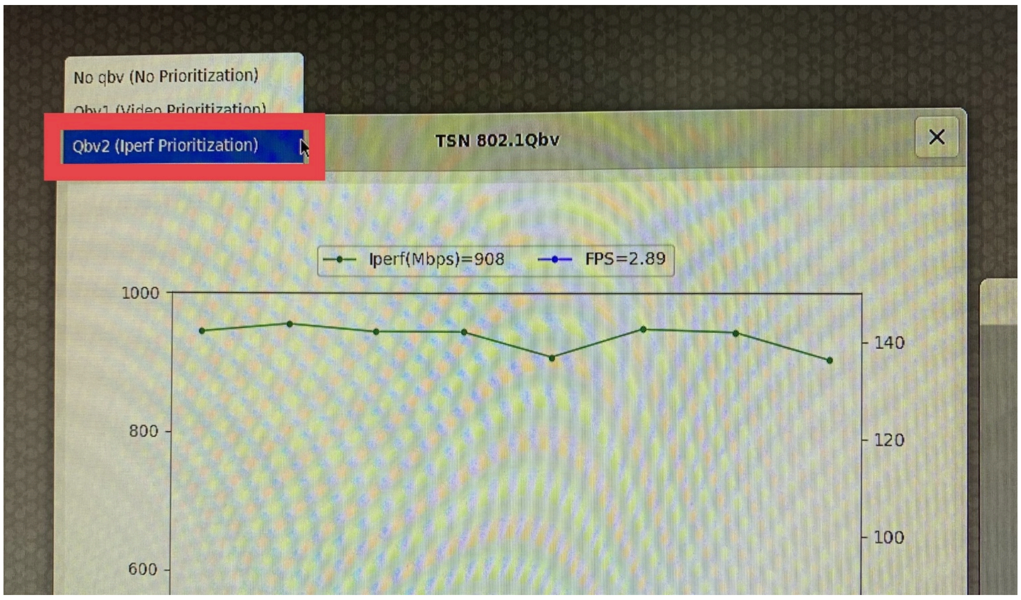
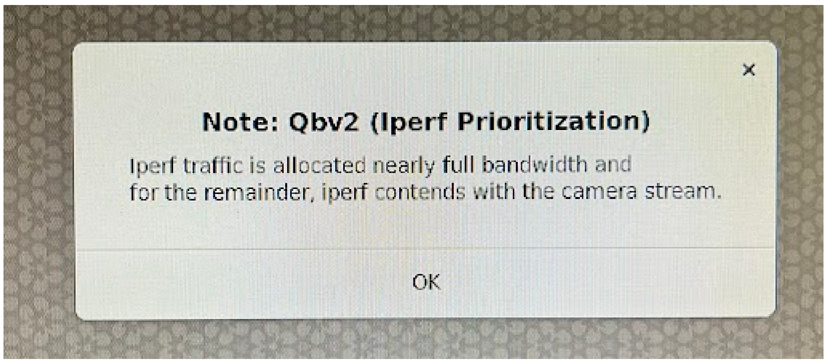

No qbv (No Prioritization)
Qbv1 (Video Prioritization)
Qbv2 (Iperf Prioritization)
TSN 802.1Qbv
Iperf(Mbps)=908
FPS=2.89
1000
800
600
140
120
100

Figure 57. Qbv2 (Iperf Prioritization) configuration button

- After clicking the **Qbv2 (Iperf Prioritization)** button, a pop-up with a message appears, as shown in Figure 58.



Note: Qbv2 (Iperf Prioritization)

Iperf traffic is allocated nearly full bandwidth and
for the remainder, iperf contends with the camera stream.

Figure 58. About Qbv2 (Iperf Prioritization) pop-up window

- To continue the demo, click the **OK** button.

Expected results:

- With the Qbv2 configuration, the camera streams with glitches. In graph iperf, the throughput ranges between 900 Mbit/s to 950 Mbit/s and the camera FPS range between 0 FPS to 10 FPS.
- The camera is limited to a half line rate (0 FPS to 15 FPS) and iperf with a full line rate (900 Mbit/s to 950 Mbit/s).
- The FPS values alter based on the lighting conditions. The values are limited from the current FPS when the configurations are applied.

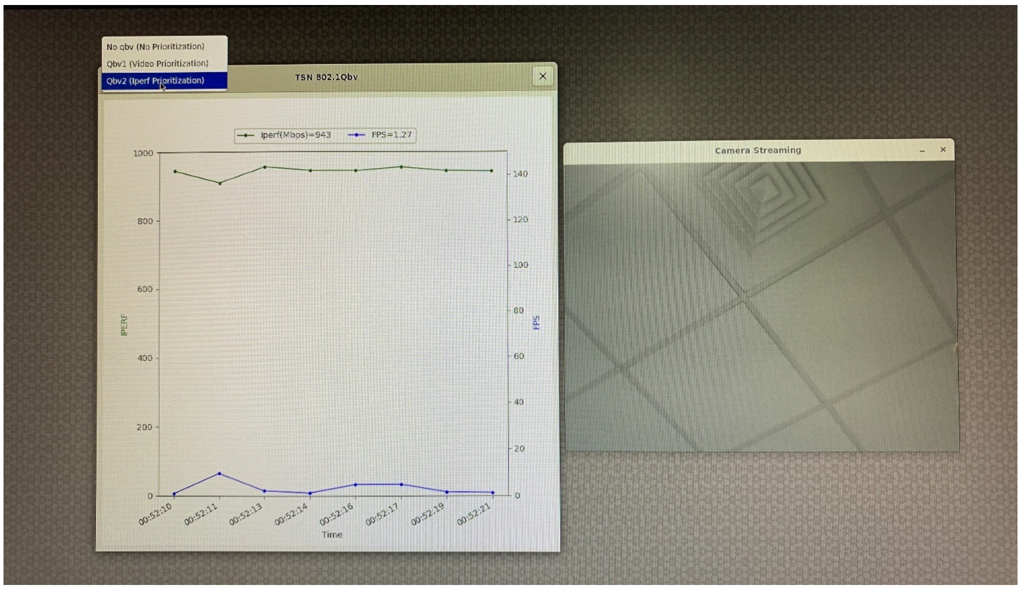

No qbv (No Prioritization)
Qbv1 (Video Prioritization)
Qbv2 (Iperf Prioritization)
TSN 802.1Qbv
Iperf(Mbps)=943
FPS=1.27
IPERF
FPS
Time
Camera Streaming

Figure 59. Qbv2 graph window and camera window

- You can select any of the TSN Qbv standards randomly.
- Click the **No qbv (No Prioritization)** configuration button.

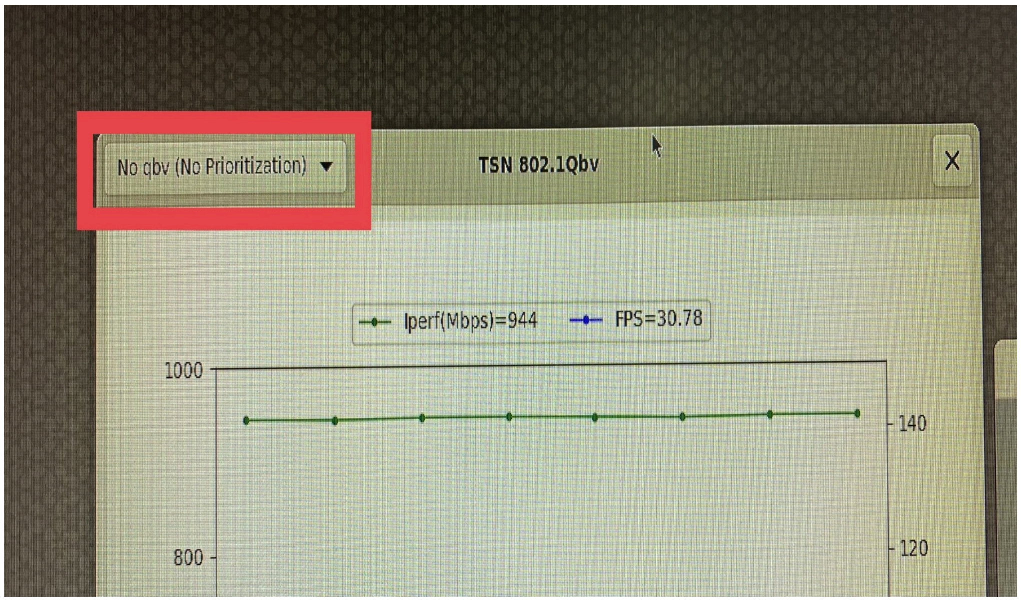
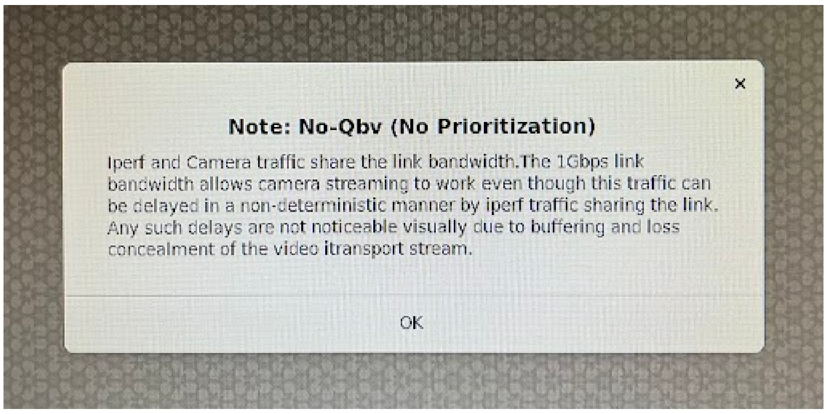


Figure 60. No qbv (No Prioritization) configuration button

- After clicking the **No qbv (No Prioritization)** button, as shown in Figure 60, a pop-up with a message appears, as shown in Figure 61.



Note: No-Qbv (No Prioritization)

Iperf and Camera traffic share the link bandwidth.The 1Gbps link bandwidth allows camera streaming to work even though this traffic can be delayed in a non-deterministic manner by iperf traffic sharing the link. Any such delays are not noticeable visually due to buffering and loss concealment of the video itransport stream.

OK

Figure 61. About No-Qbv (No Prioritization) pop-up

- To continue the demo, click the **OK** button.

Expected results:

- With No qbv (No Prioritization), the iperf throughput must result between 900 Mbit/s to 950 Mbit/s. The camera must stream fine with FPS values of 25 FPS to 30 FPS.
- The iperf and the camera stream with full line rates.

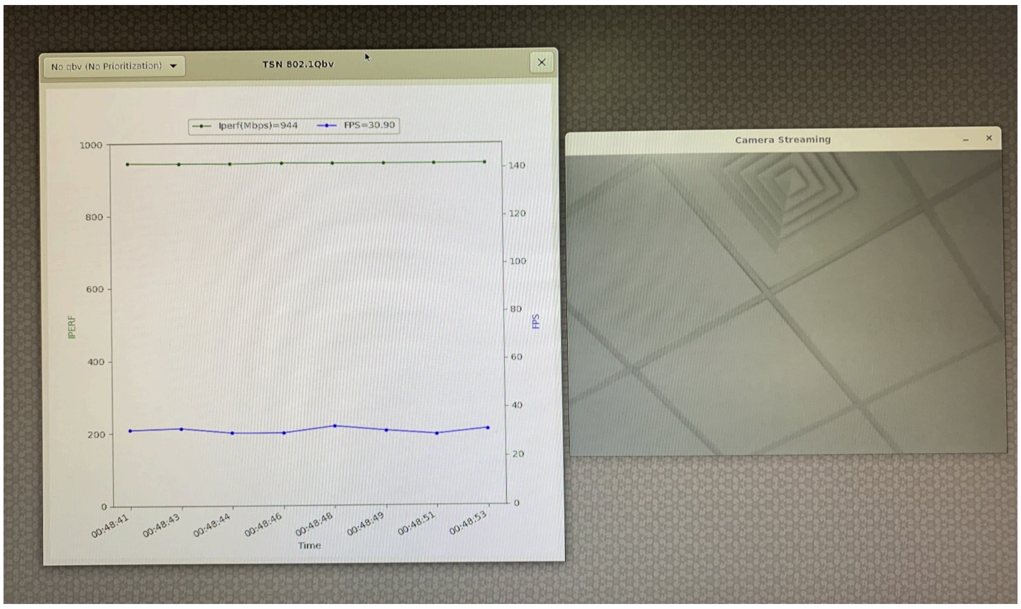


Figure 62. No-Qbv Graph window and Camera window

- To stop the demo, select the **Stop Demo** button in the TSN Qbv login window.

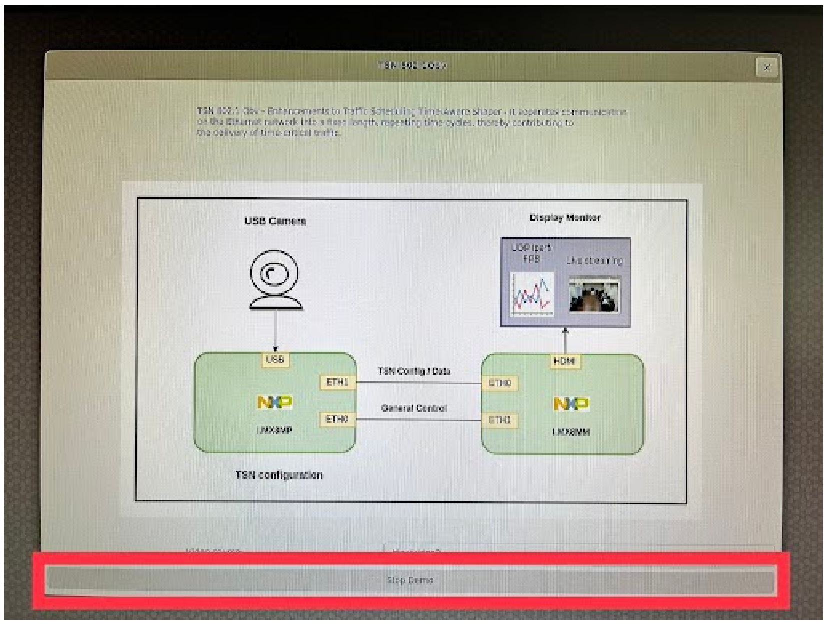


Figure 63. Stop demo button

- You can switch across any of these configurations to understand how the demo works.

3.3.1 Application note link

For the application note, refer *TSN 802.1Qbv Demonstration using i.MX 8M Plus* (document AN13995).

3.3.2 Limitations

Keep the following limitations in mind while working on the TSN Qbv Demo:

- During verification, the TSN Qbv Demo setup must not be disturbed.
- For a smooth demo experience, Ethernet cables and camera must not be disturbed.
- Mostly Logitech and Papalook cameras are used to verify the TSN Qbv Demo.
- Camera frame rate (number of frames captured per second, FPS) depends on the lighting conditions of the area where the camera is placed. For example, in a dark, low-light, or moderate-light area, the camera frame rate is from 10 FPS to 15 FPS.
- For the TSN Qbv Demo, the recommended frame rate is 25 FPS to 30 FPS. To achieve this frame rate, the camera must be placed in an area with bright light.
- The TSN Qbv Demo windows open on the monitor in a random order. The windows must be rearranged manually.
- During verification, the TSN Qbv Demo windows must not be closed.
- Initially, the iperf drops to 800 Mbit/s when selecting a Qbv2 configuration and streams with a full line rate. However, this behavior is achieved in rare cases.
- The TSN Qbv Demo must be stopped before switching to any other demo.

3.4 Security

The following demos show various security features on our products.

3.4.1 EdgeLock Secure Enclave example demo

Required materials: Display and mouse

The EdgeLock Enclave (ELE) is an independent security domain that provides security services. These services include key management, random number generation, data storage, execution of cryptographic services. This application is developed to make some of the ELE functionalities visible.

Currently, this demo has the following three functions:

- Random number generation (RNG)
- AES encryption and decryption
- Data storage

Figure 64 shows the ELE functions.

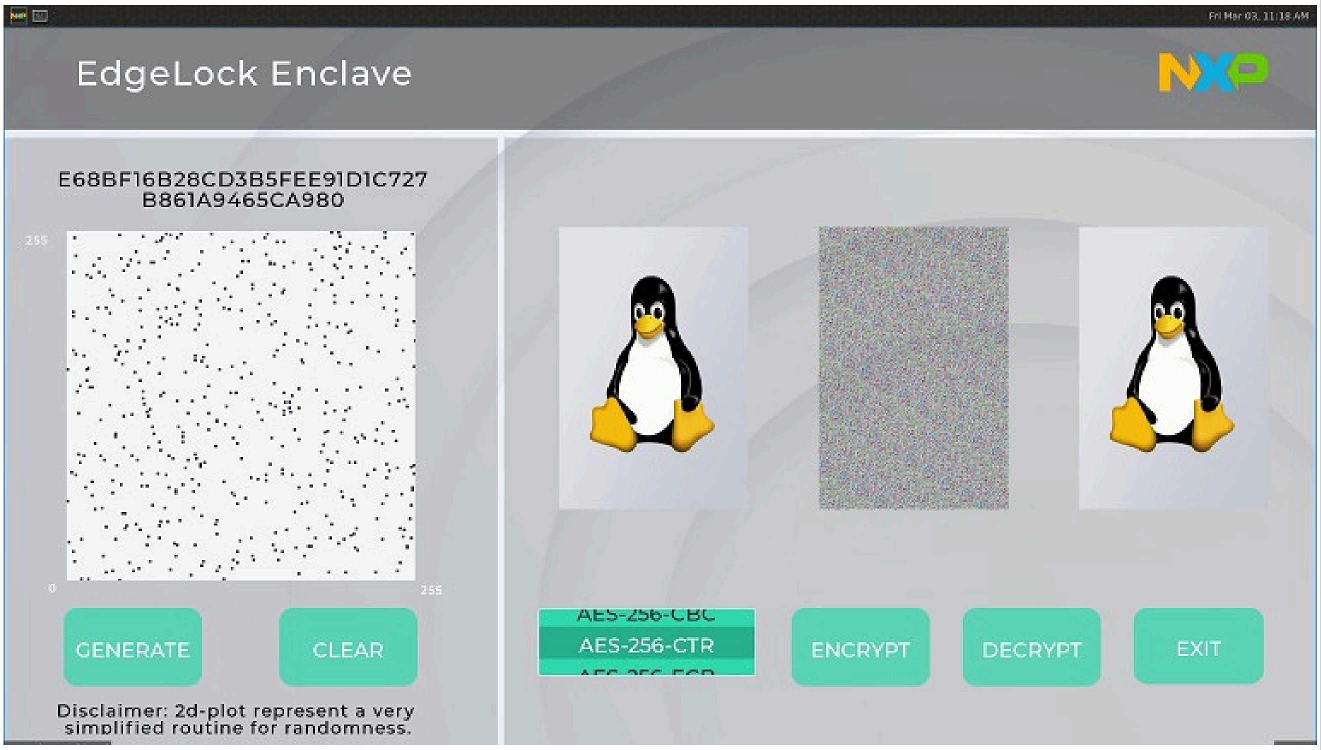


Figure 64. EdgeLock Secure Enclave example demo

3.4.1.1 Random number generation

For RNG, the UI screen consists of the following four elements from top to bottom:

- Text box: It shows 20 bytes of consecutive random numbers
- 2d-plot: The ELE RNG generates the position of the dot here
- Generate button: It generates 200 bytes of random numbers, add 100 pairs of dots into the 2d-plot, and update the text box with the first 20 bytes
- Clear button: It clears all the dots in the 2d-plot and text box

To generate 200 bytes of random numbers, press **GENERATE**. As a result, the 2d-plot and text box are updated.

To clear all the dots in the 2d-plot and the text box, press **CLEAR**.

3.4.1.2 AES crypto

The main idea of this function is to show that ELE has an AES crypto feature. This feature has the following four elements:

- 3 pictures: The first one is the original image, the second is the encrypted image, and the third is a decrypted image
- Mode select bar: Helps select one of the AES modes (AES-256-ECB/CBC/CTR)
- Encrypt button
- Decrypt button

Select one of the AES modes and press the **ENCRYPT** button. A window pops up to set your password. Input your password and click OK in the window. The second image updates when the encryption is finished.

To decrypt the encrypted image, press the **DECRYPT** button. A window pops up to confirm your password. If the password is verified, the third image gets updated. Otherwise, a warning is displayed indicating that the password is wrong.

3.4.1.3 Data storage

This demo uses the data storage of ELE to store the hash value of the password of the AES crypto. The ELE stores the SHA-256 hash value of the password when the image is encrypted. The hash value gets verified when the image is decrypted.

3.4.1.4 Troubleshooting

If the application fails to start, perform the following steps:

1. Ensure that the `nvm_daemon` and `rm /etc/ele/* -rf` are started.
2. Reboot the device by a power button switch rather than a Linux command `reboot`.

3.5 GPU

Several demos that show off the power of an on-chip GPU are included in this section.

3.5.1 OpenVG 2D

OpenVG 2D is a cross-platform API that provides low-level hardware acceleration for vector graphics. The following demo uses this library to perform their various actions.

3.5.1.1 Tiger G2D

Required materials: Mouse and display output

This demo shows a vector image being rotated and scaled using OpenVG.

3.5.2 GLES2

OpenGL Embedded Systems is a set of APIs for rendering 2D and 3D graphics using an onboard GPU. The following demos use this library to display various graphics.

3.5.2.1 Vivante launcher

Required materials: Mouse and display output

This demo shows an example of a launcher menu that is used in an infotainment system.

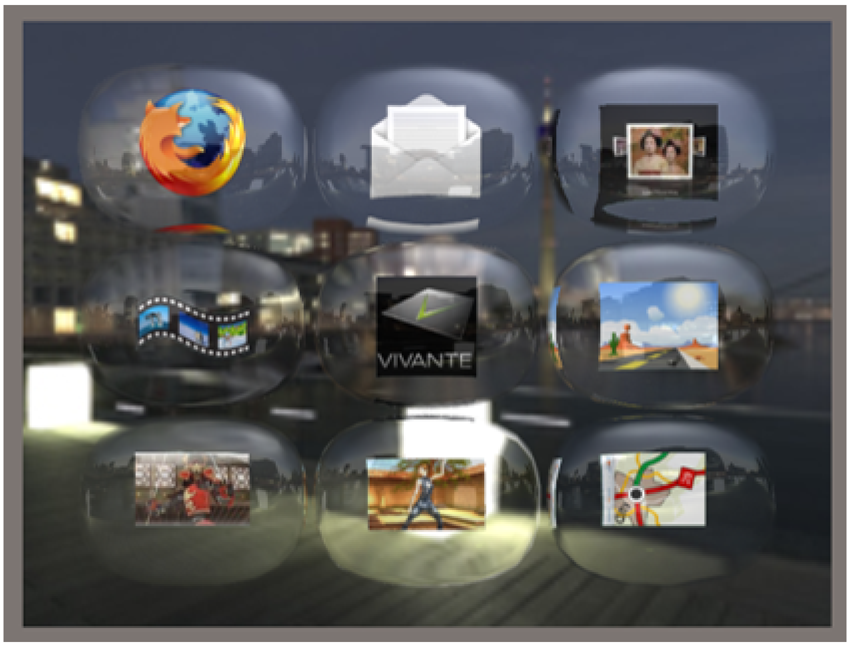


Figure 65. Example of a Vivante launcher

3.5.2.2 Cover flow

Required materials: Mouse and display output

This demo shows an example of a movie selection screen that is used in an infotainment system.

3.5.2.3 Vivante tutorial

Required materials: Mouse and display output

This demo shows an example of a complex shape generated using the GPU.

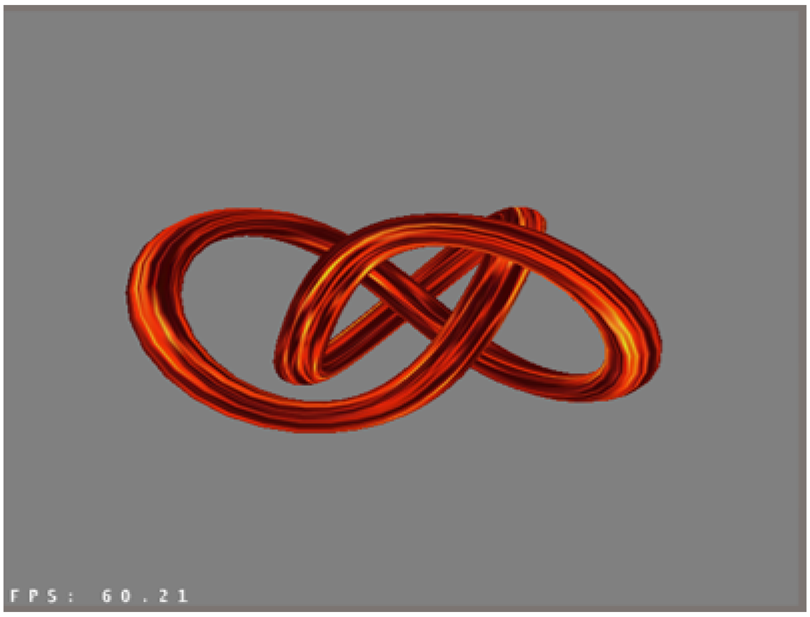


Figure 66. Vivante tutorial example

3.5.2.4 Bloom

Required materials: Mouse and display output

This demo shows an example of how to create a bloom effect. The idea is not to create the most accurate bloom, but something that is fairly fast to render. Instead of increasing the kernel size to get a good blur, do a fairly fast approximation by downscaling the original image to multiple smaller render targets. Then blur them using a relatively small kernel, and then finally rescale the result to the original size.

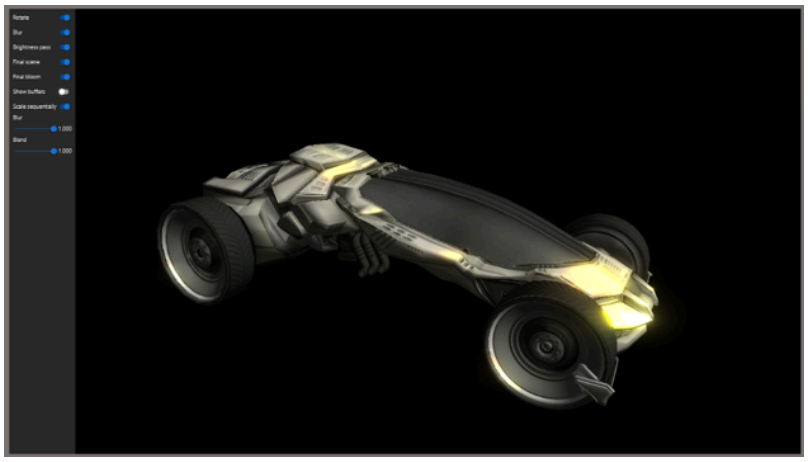


Figure 67. Bloom example

3.5.2.5 Blur

Required materials: Mouse and display output

This demo uses the two-pass linear technique. It further reduces the bandwidth requirement by downscaling the source image to 1/4 its size (1/2w x 1/2h) before applying the blur. It then upscales the blurred image to provide the final image. It works well for large kernel sizes and relatively high sigma, but the downscaling produces visible artifacts with low sigma.

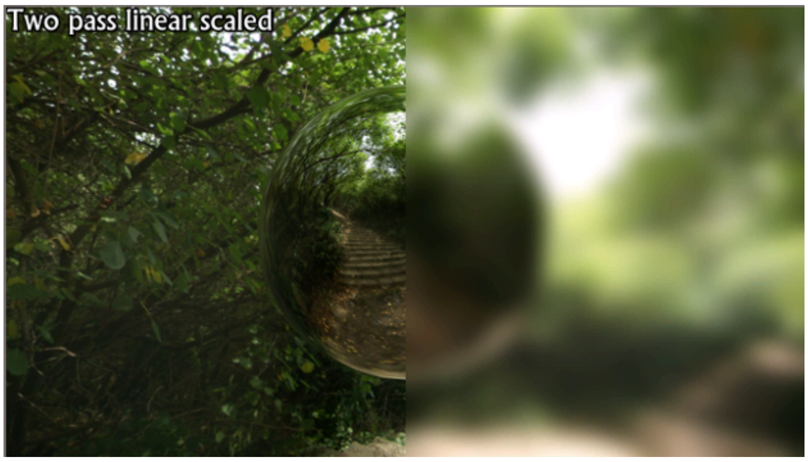
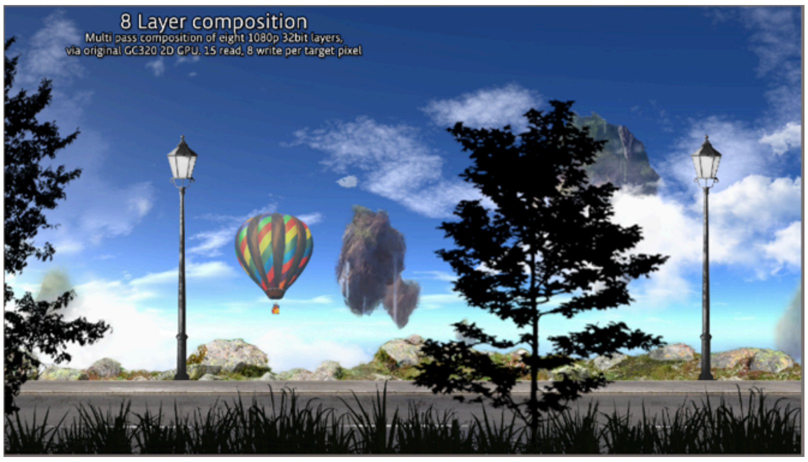


Figure 68. The blur example

3.5.2.6 Eight layer blend

Required materials: Mouse and display output

This demo creates a simple parallax scrolling effect by blending eight 32-bit per pixel 1080 p layers on top of each other. It is not the most optimal way as it uses eight passes. However, it is a good example of worst-case bandwidth usage in an operation. The demo is created to compare GLES to the G2D eight blend functionality.


8 Layer composition
Multi pass composition of eight 1080p 32bit layers,
via original GC320 2D GPU. 15 read, 8 write per target pixel
Figure 69. Eight-layer blend example

3.5.2.7 Fractal shader

Required materials: Mouse and display output

This demo can render both the Julia and Mandelbrot sets using a fragment shader. This demo is used to demonstrate GPU shader performance using roughly 515 instructions to render each fragment while generating the Julia set. It uses no textures, no overdraw, and has a minimal bandwidth requirement.

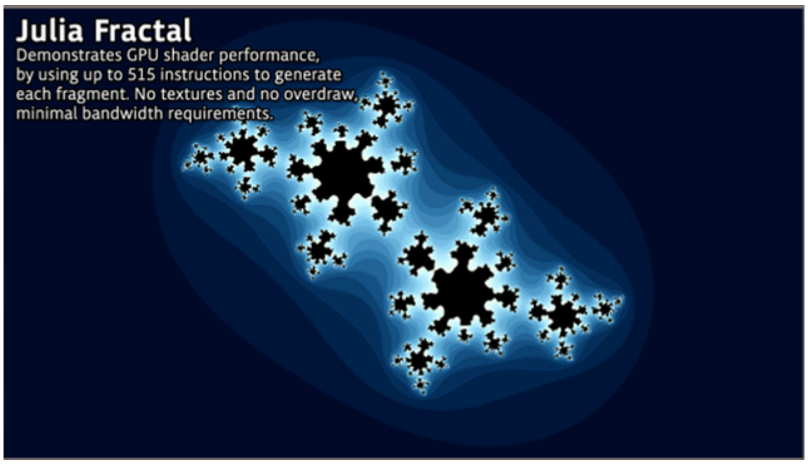

Julia Fractal
Demonstrates GPU shader performance, by using up to 515 instructions to generate each fragment. No textures and no overdraw, minimal bandwidth requirements.

Figure 70. Fractal shader example

3.5.2.8 Line Builder 101

Required materials: Mouse and display output

This demo shows a simple example of dynamic line rendering using the Line Builder 101 helper class. The Line Builder 101 has "Add" methods for most FslBase Math classes, such as BoundingBox, BoundingSphere, BoundingFrustrum, and Ray.

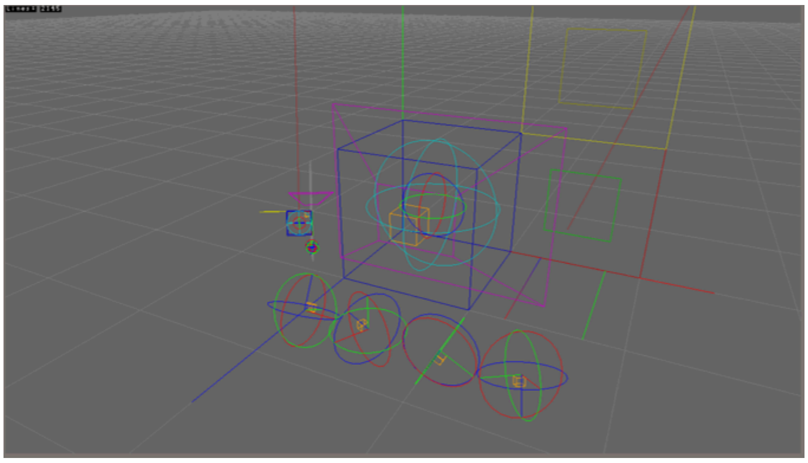


Figure 71. Line Builder 101 example

3.5.2.9 Model loader

Required materials: Mouse and display output

This demo demonstrates how to use the FslSceneImporter and Assimp to load a scene and render it using OpenGLES2. The model is rendered using a simple per-pixel directional light shader.

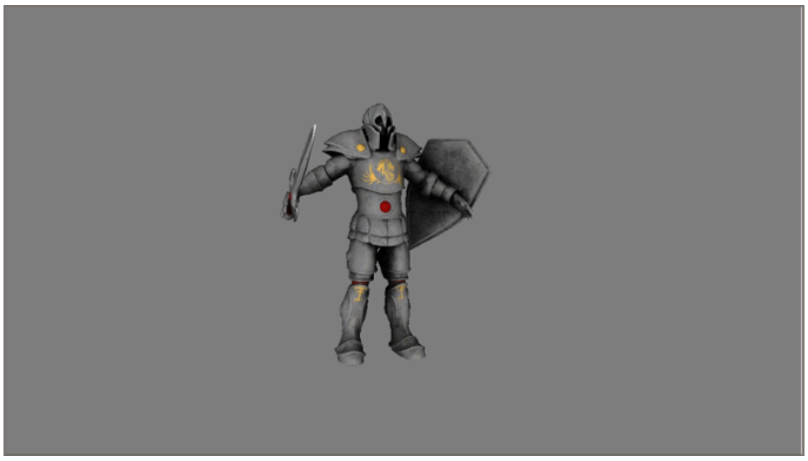


Figure 72. Model loader example

3.5.2.10 S03 transform

Required materials: Mouse and display output

This demo renders an animated vertex-colored triangle. It shows how to modify the model matrix to rotate a triangle and use demoTime and deltaTime to do frame rate-independent animation.

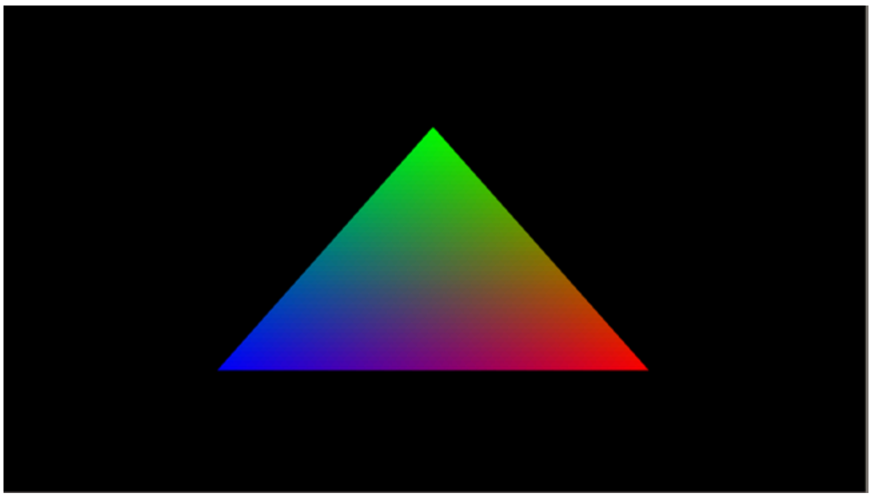


Figure 73. Example for S03 transform

3.5.2.11 S04 projection

Required materials: Mouse and display output

This demo shows how to build a perspective projection matrix and render two simple 3D models using frame rate-independent animation.

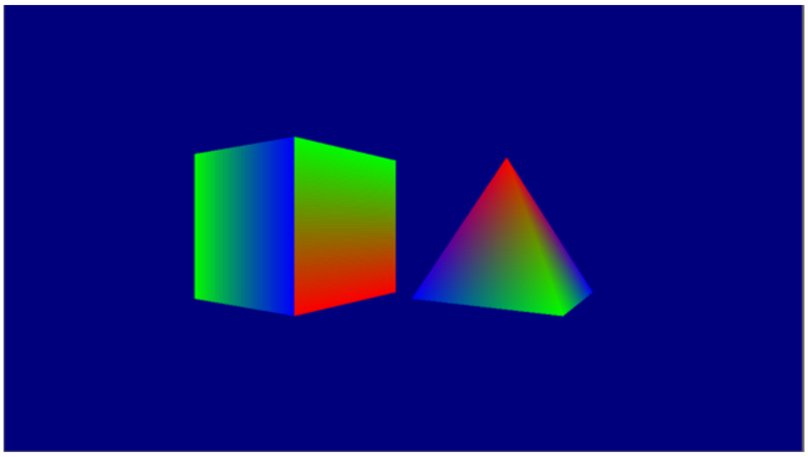


Figure 74. S04 projection example

3.5.2.12 S06 texturing

Required materials: Mouse and display output

This demo shows how to use the Texture class to use a texture in a cube. This demo also shows how to use the ContentManager service to load a `.png` file from the Content directory into a bitmap utility class, which is then used to create an OpenGL ES texture.

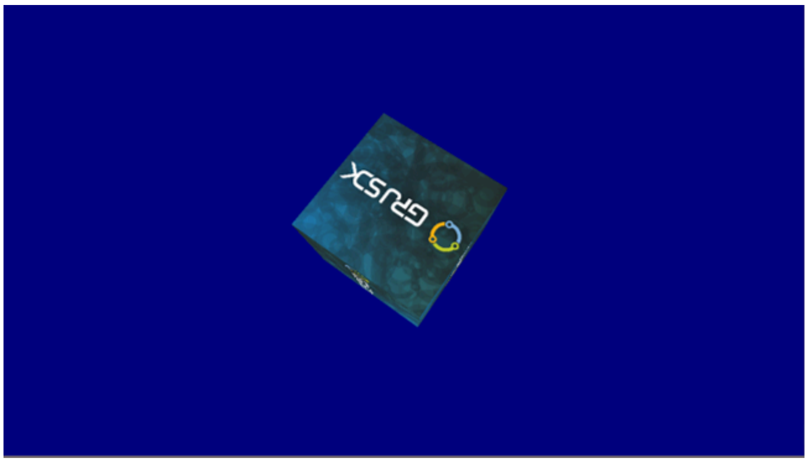


Figure 75. S06 texturing example

3.5.2.13 Mapping

Required materials: Mouse and display output

This demo shows how to use a cubemap texture to simulate a reflective material. This demo also shows how to use the ContentManager service to load a `.dds` file from the Content directory into a Texture utility class, which is then used to create an OpenGL ES cubemap texture.

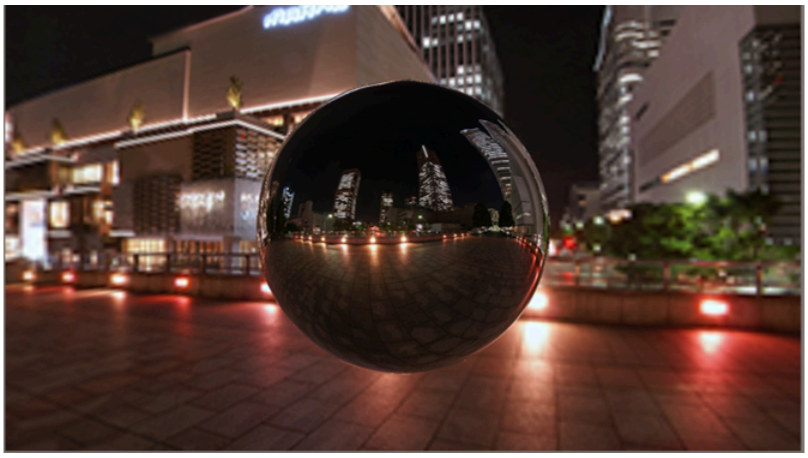


Figure 76. Mapping example

3.5.2.14 Mapping refraction

Required materials: Mouse and display output

This demo is a variation from "Mapping". In the previous example, a cubemap texture is used, but for this case, instead of simulating a reflective material, a refractive material is simulated. This demo also shows how to use the ContentManager service to load a `.dds` file from the Content directory into a Texture utility class, which is then used to create an OpenGL ES cubemap texture.

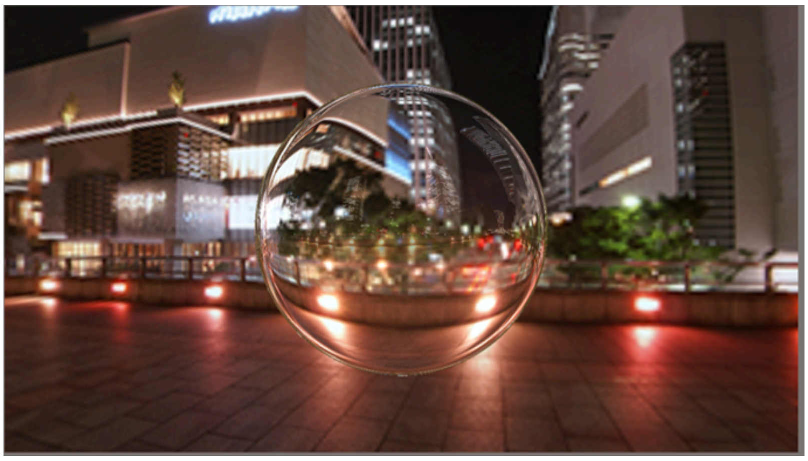


Figure 77. Mapping refraction example

4 References

The references used to supplement this document are as follows:

- *TSN 802.1Qbv Demonstration using i.MX 8M Plus* (document AN13995)
- *i.MX Yocto Project User Guide* (document IMXLXYOCTOUG)
- *Embedded Linux for i.MX Applications Processors* (document IMXLINUX)
- *GoPoint for i.MX Applications Processors* (document GPNTRN)
- 8-microphone array board: 8MIC-RPI-MX8
- *i.MX Linux User's Guide* (document IMXLUG)
- *i.MX 8MIC-RPI-MX8 Board Quick Start Guide* (document IMX-8MIC-QSG)

- *i.MX 8M Plus Gateway for Machine Learning Inference Acceleration* (document AN13650)

5 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

6 Revision history

Table 3 summarizes the revisions to this document.

Table 3. Revision history

Revision number	Release date	Description
GPNTUG v.10.0	30 September 2024	• Added i.MX E-Bike VIT • Updated References
GPNTUG v.9.0	8 July 2024	• Added Security
GPNTUG v.8.0	11 April 2024	• Updated NNStreamer demos • Updated Object classification • Updated Object detection • Removed section "Brand detection" • Updated Machine learning gateway • Updated Driver monitoring system demo • Updated Selfie segmenter • Added i.MX smart fitness • Added Low-power machine learning demo
GPNTUG v.7.0	15 December 2023	• Updated for the 6.1.55_2.2.0 release • Rename from NXP Demo Experience to GoPoint for i.MX Applications Processors • Added 2Way video streaming
GPNTUG v.6.0	30 October 2023	Updated for the 6.1.36_2.1.0 release
GPNTUG v.5.0	22 August 2023	Added i.MX multimedia player

Table 3. Revision history*...continued*

Revision number	Release date	Description
GPNTUG v.4.0	28 June 2023	Added TSN 802.1 Qbv demo
GPNTUG v.3.0	07 December 2022	Updated for 5.15.71 release
GPNTUG v.2.0	16 September 2022	Updated for 5.15.52 release
GPNTUG v.1.0	24 June 2022	Initial release

Legal information

Contents

Date of release: 30 September 2024
Document identifier: GPNTUG