GPNTUG

GoPoint for i.MX Applications Processors User Guide

Rev. 6 — 15 December 2023 User guide

Document information

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

1 Introduction

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

The GoPoint for i.MX Applications Processors is for the ones who are interested in showcasing the various features and capabilities of the SoCs provided by NXP. 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 the GoPoint for i.MX Applications Processors. This document covers how to run the 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 the Included demos chapter.*

1.1 Installing the GoPoint for i.MX Applications Processors

The 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 the 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 must be used.

2 Demo launcher

This section describes the demo launcher.

2.1 Graphical user interface

On boards where the 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.

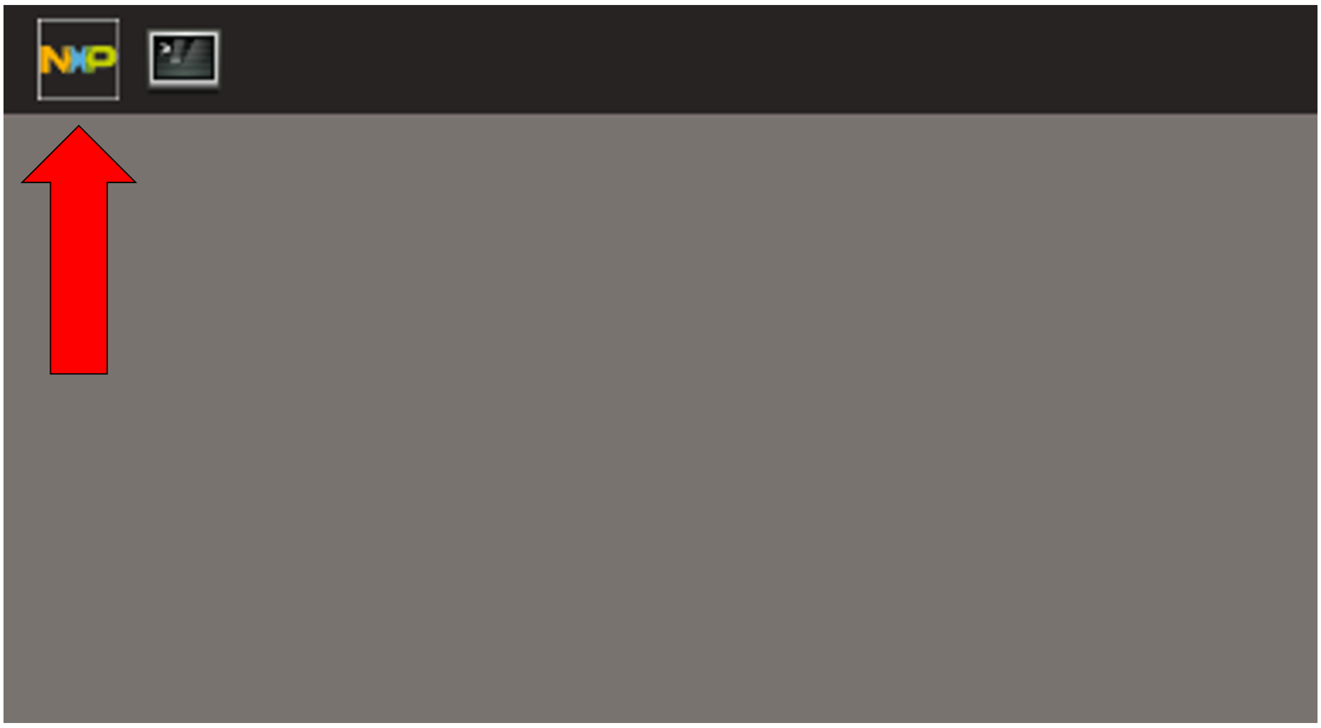
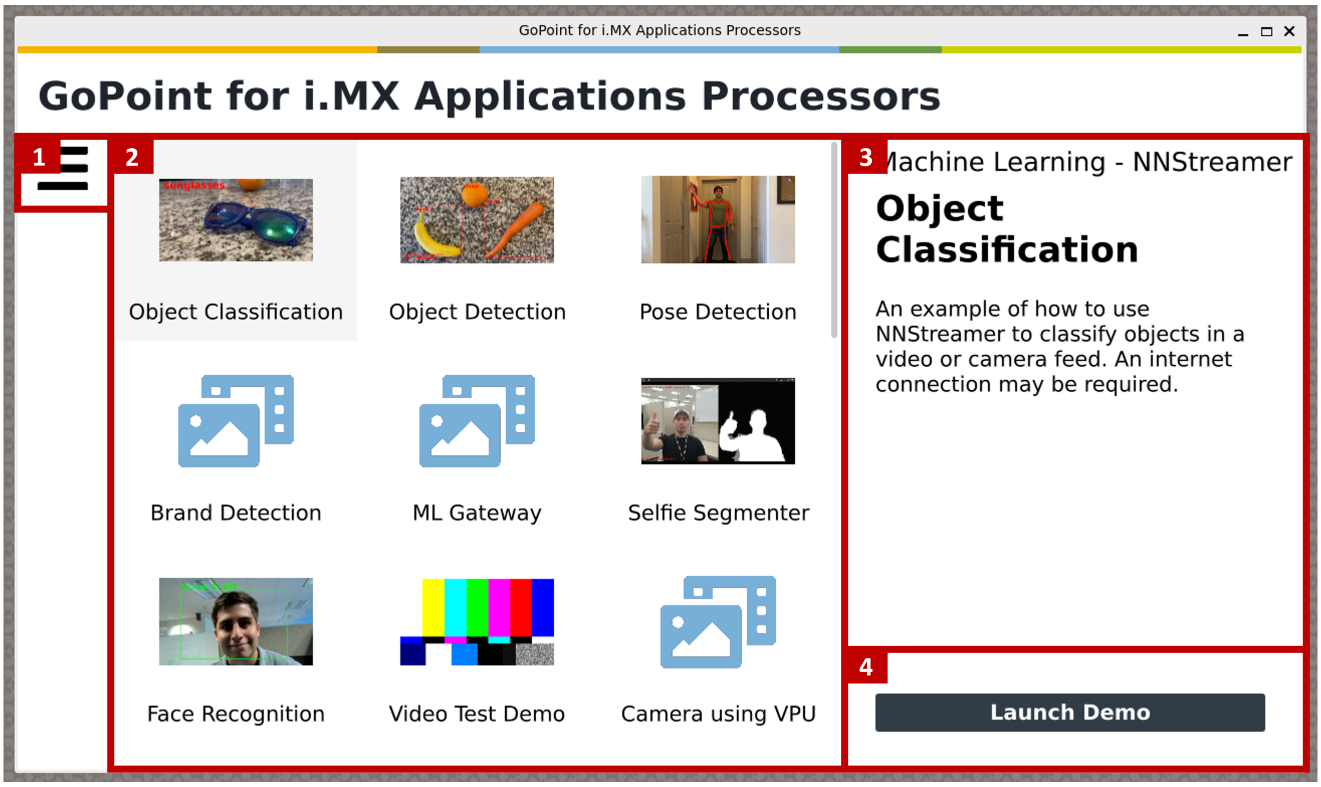


Figure 1. GoPoint for i.MX Applications Processors logo

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.*



An example of how to use NNStreamer to classify objects in a video or camera feed. An internet connection may be required.

Figure 2. GoPoint for i.MX Applications Processors

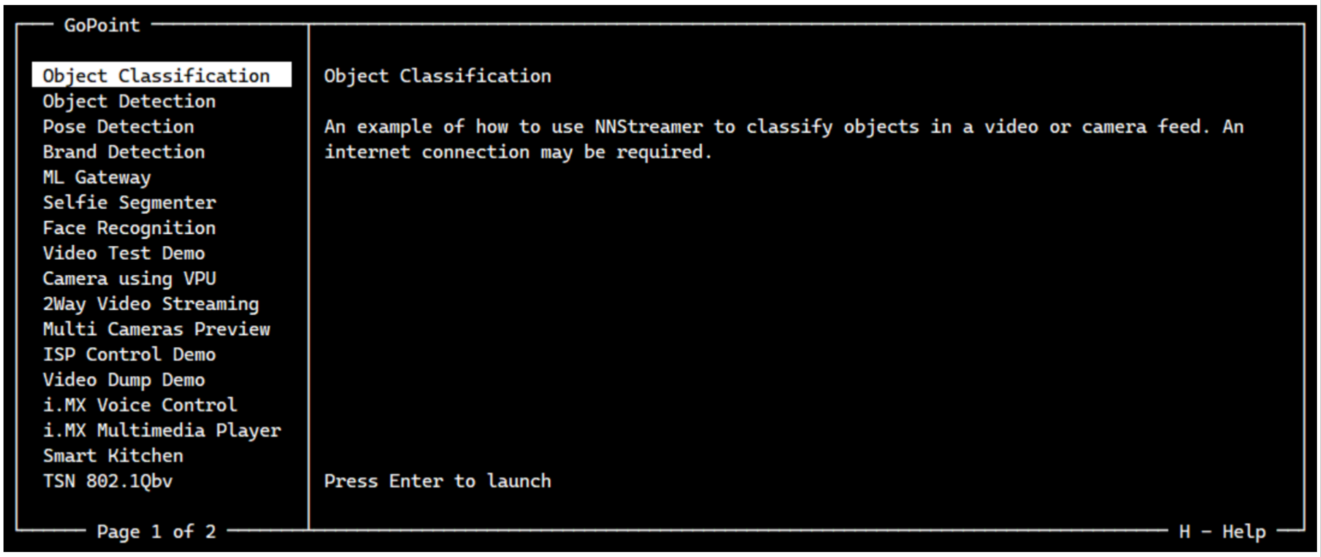
2.2 Text user interface

Demos can also be launched from the command line through log-in into the board remotely or using the onboard serial debug console. Keep in mind 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, type the following command into the command line:

```
# gopoint tui
```



```
GoPoint
Object Classification        Object Classification
Object Detection
Pose Detection               An example of how to use NNStreamer to classify objects in a video or camera feed. An
Brand Detection              internet connection may be required.
ML Gateway
Selfie Segmenter
Face Recognition
Video Test Demo
Camera using VPU
2Way Video Streaming
Multi Cameras Preview
ISP Control Demo
Video Dump Demo
i.MX Voice Control
i.MX Multimedia Player
Smart Kitchen
TSN 802.1Qbv                 Press Enter to launch

Page 1 of 2                                                                                        H - Help
```

Figure 3. Text user interface (TUI)

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 Included demos

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

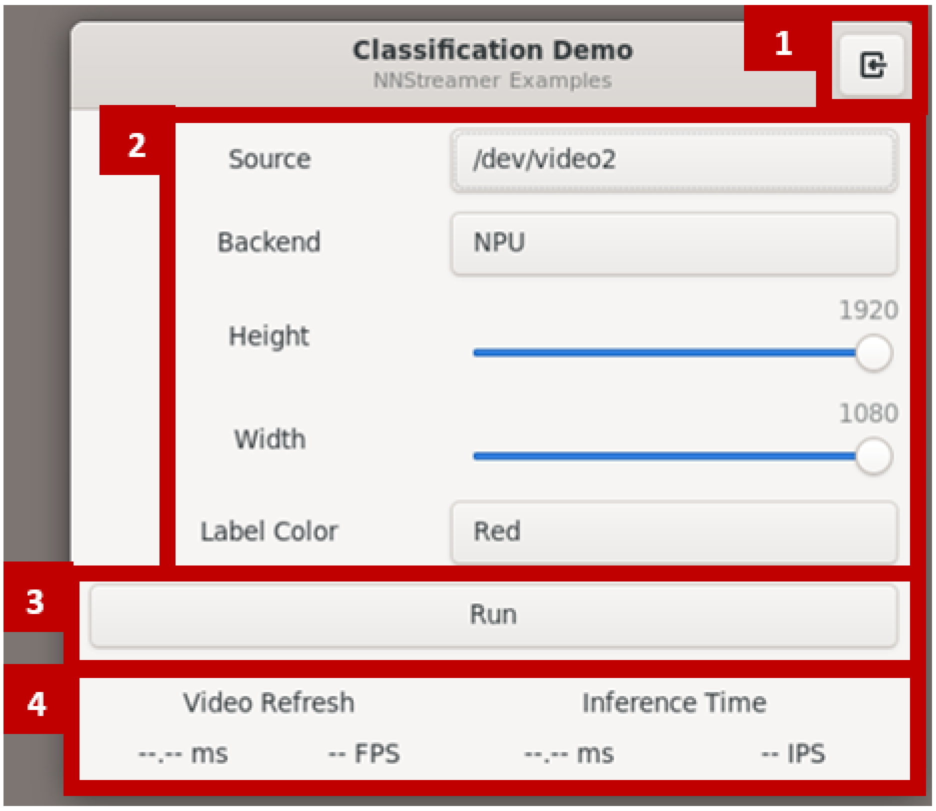
Note: *If the demo covers up the demo launcher window or another window that is required, drag the necessary windows, including video output windows, with the mouse.*

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 components that enable machine learning in video pipelines. The included demos use NNStreamer to create video outputs that overlay inference information onto the video.



Source: /dev/video2
Backend: NPU
Height: 1920
Width: 1080
Label Color: Red

Figure 4. NNStreamer launcher

All NNStreamer demos have identical 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 the camera to use or to use the example video provided (requires an Internet connection, not available in the "Brand Detection" 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.
   - **Label Color:** Select the color of the overlay labels.
3. Clicking the **Run** button locks the current settings and starts the camera feed and inferencing.
4. While the video plays, statistics about the current speed of the video feed and inferencing are displayed here. Video refresh represents the speed of the video inferencing pipeline. The inference time represents the time that it takes to complete one inference.

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

3.1.1.1 Object 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 and runs a classification inference on the video frames as they come in. This demo uses a pretrained quantized MobileNet

V1 TFLite model that 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.

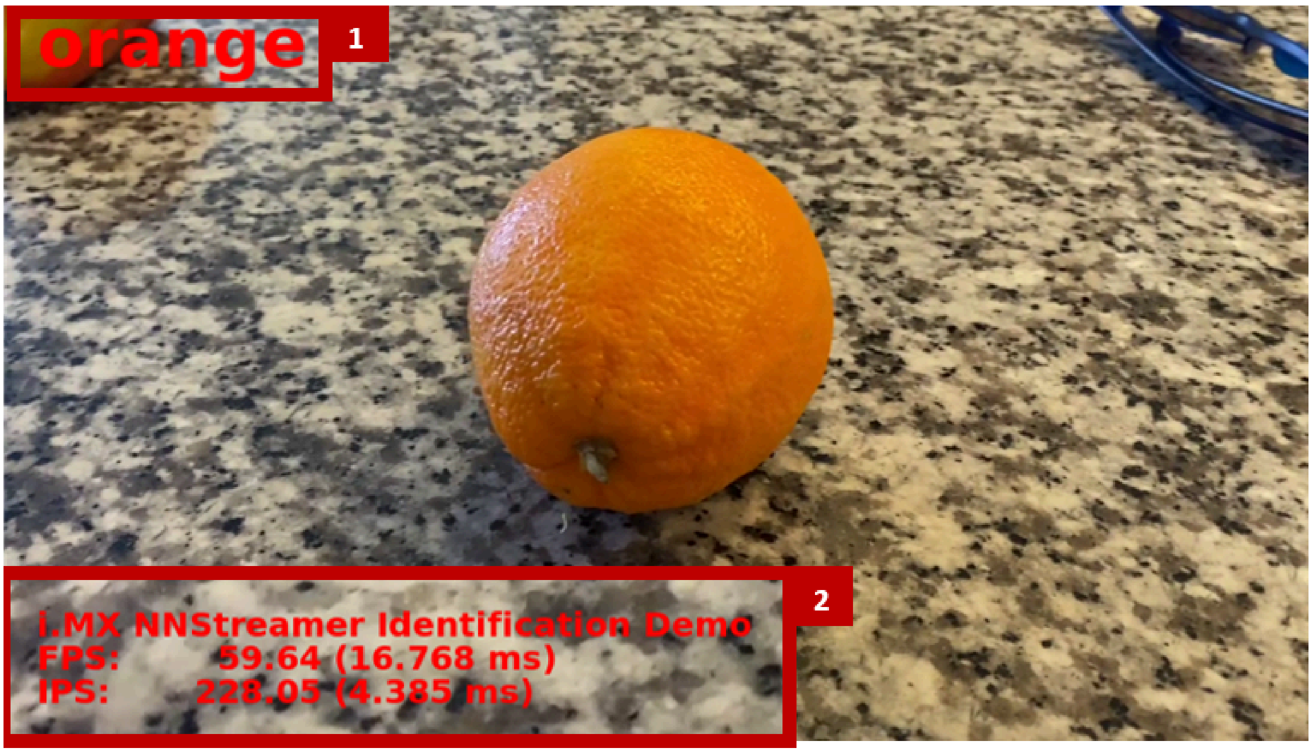


Figure 5. NNStreamer output classification example

When the demo starts, a video overlay of the following information is shown:

1. The label with the highest probability of being in the image.
2. Frames Per Second (FPS) of the video inferencing pipeline and Inferences Per Second (IPS) based on the time it takes to complete one inference.

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 and 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 that 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.

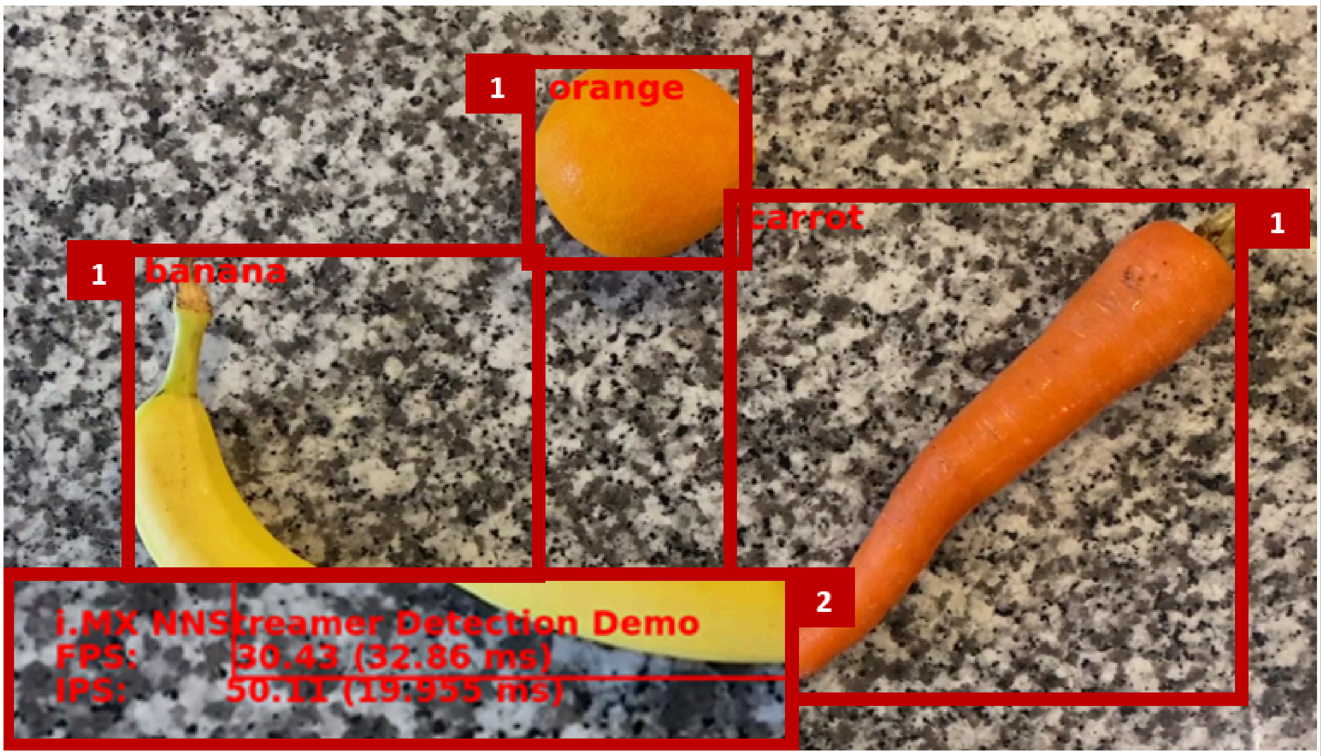


Figure 6. NNStreamer object detection example

When the demo starts, a video overlay of the following information is shown:

1. Detected objects have boxes drawn around them, and their labels are displayed near the top-left corner.
2. FPS of the video inferencing pipeline and the IPS based on the time it takes to complete one inference.

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 PoseNet ResNet-50 TFLite model that is trained on 17 body points. When the video starts, the detected pose is overlaid onto the incoming video feed.

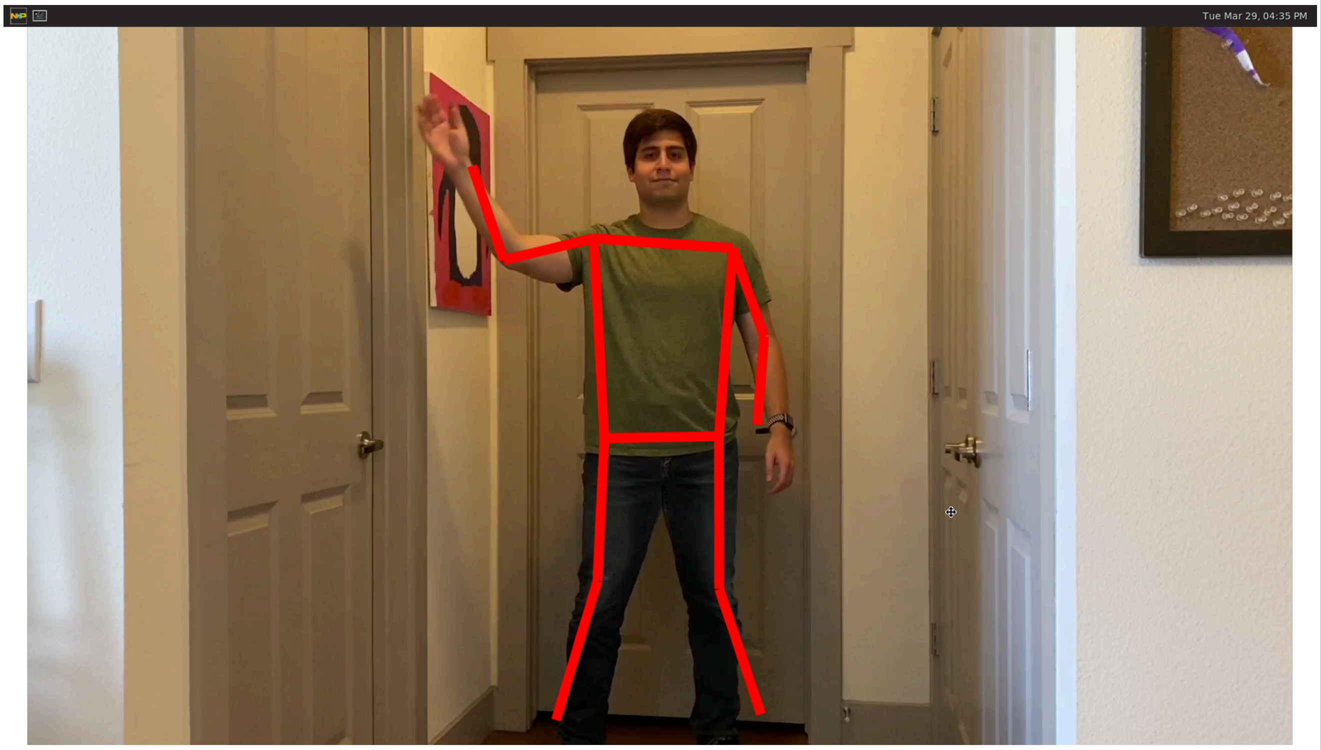


Figure 7. NNStreamer pose detection example

3.1.1.4 Brand 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 and runs a brand detection inference on the video frames as they come in. This demo uses a pretrained quantized Popular Products V1 TFLite model that is trained on 100,000 US products. The result of the inference is then displayed within the video frame.

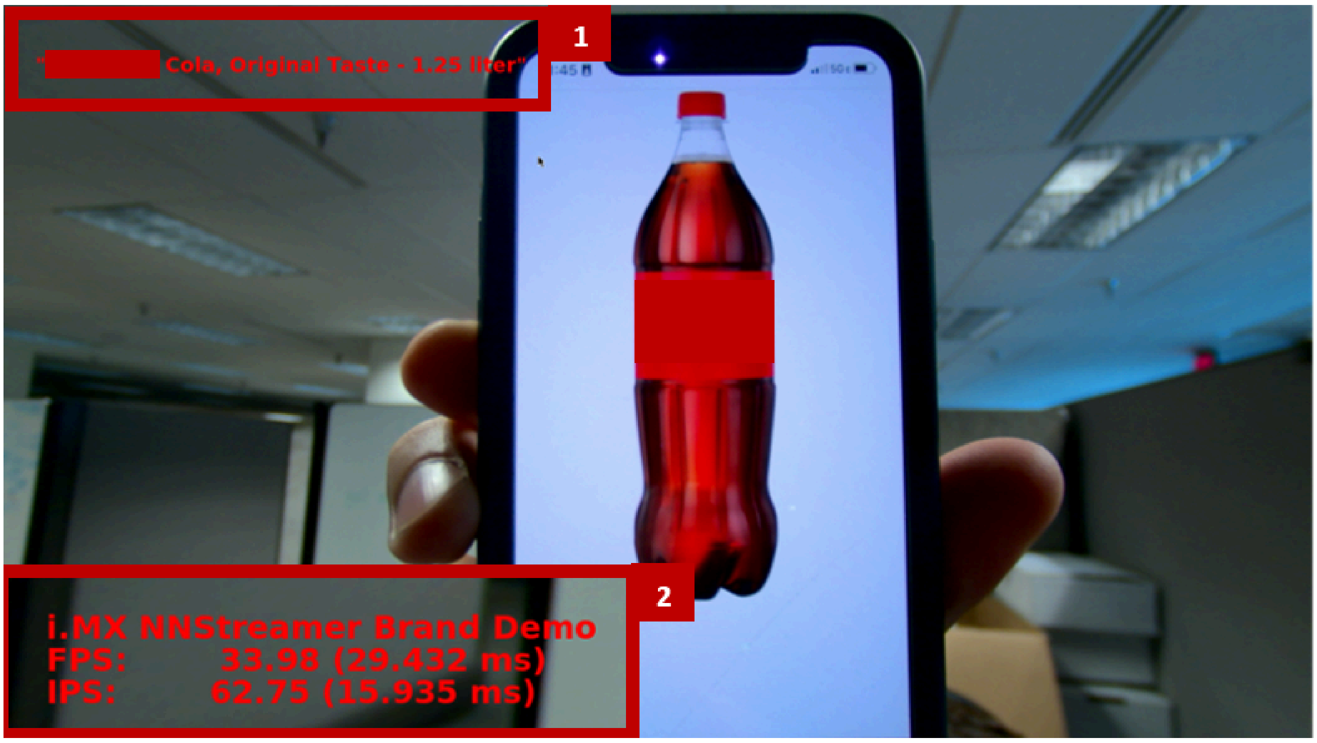

"[redacted] Cola, Original Taste - 1.25 liter"
1
i.MX NNStreamer Brand Demo
FPS: 33.98 (29.432 ms)
IPS: 62.75 (15.935 ms)
2

Figure 8. NNStreamer brand detection example

When the demo starts, a video overlay of the following information is shown:

1. The label with the highest probability of being in the image.
2. FPS of the video inferencing pipeline and IPS, based on the time it takes to complete one inference.

3.1.1.5 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 demo allows users to create an ML gateway on the i.MX 8M Plus EVK and have other devices connected to it, such as the i.MX 8M Mini EVK. Also, the server broadcasts its IP address so clients can connect to the gateway effortlessly. An NNStreamer pipeline is used to collect the data from the client, process it on the ML gateway (inference), and return the results back to the client. The client interprets the results from the ML gateway (decoding stage of bounding boxes for this demo) and shows them on the display. This demo requires two devices, both connected over a strong Internet connection.

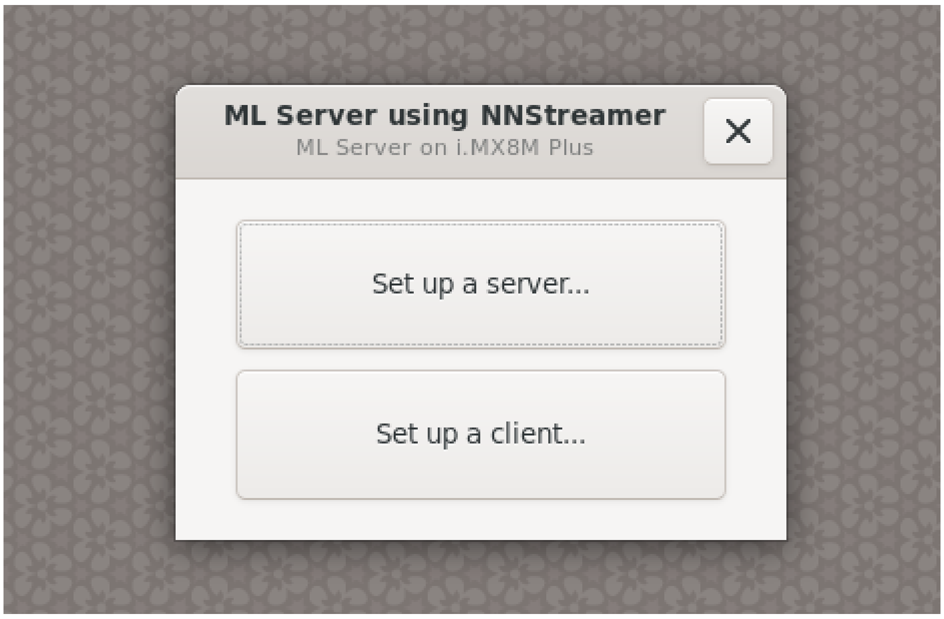


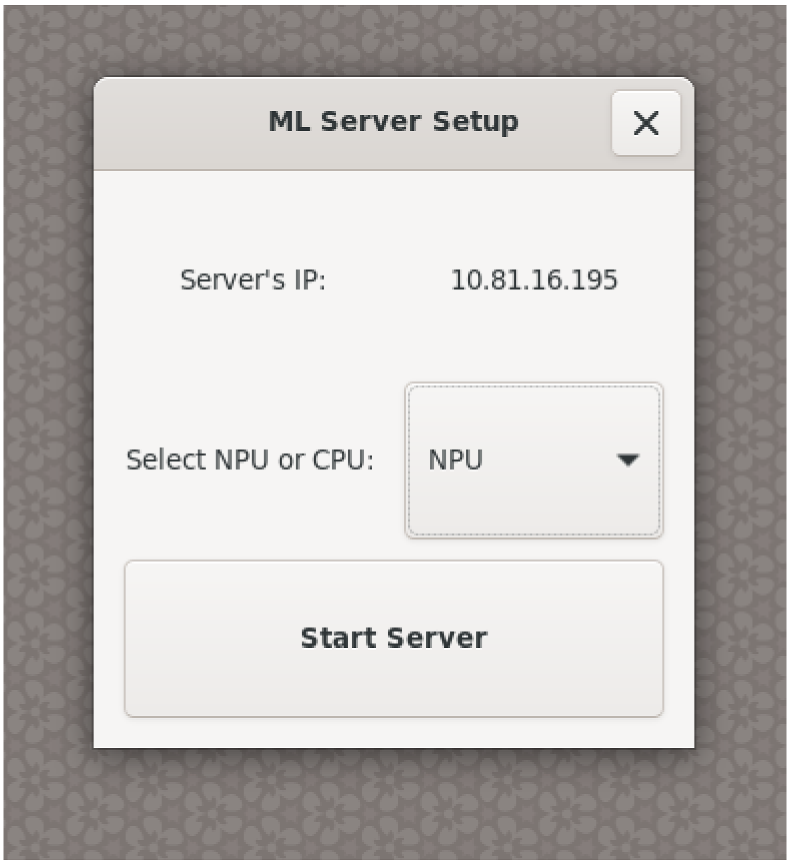
Figure 9. ML gateway device type select

When the demo starts, users must pick whether the device is a server or a client.

- If the device is acting as a server, select **Set up a server…**
- If the device is acting as a client, select **Set up a client…**

3.1.1.5.1 Setting up a server

Before the server starts up, it displays the current IP address next to the server IP. By selecting the appropriate option from the dropdown, the server enables users to choose whether to perform inferences on the CPU or the NPU (if available). When ready, click the **Start Server** button. If the server is set up successfully, the **Server is running** gets displayed.



Server's IP: 10.81.16.195

Select NPU or CPU: NPU

Figure 10. ML gateway server setup

3.1.1.5.2 Setting up a client

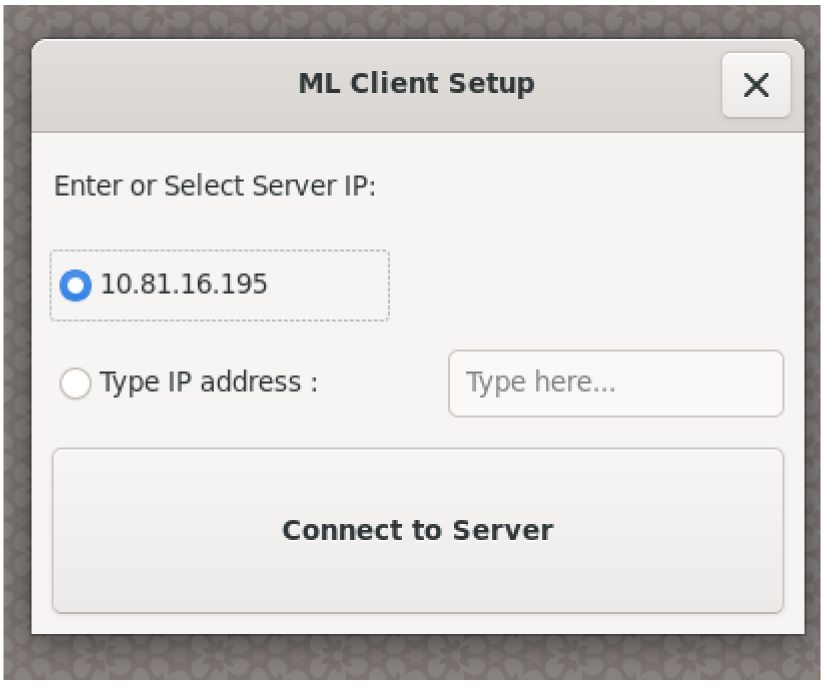


Figure 11. ML gateway client setup

When setting up a client, the device looks for a server IP and, when found, displays that as an option to pick in the next window. Users can also type in their own IP address. When ready, click the **Connect to Server** button. The device connects to the server and displays a video output with the detected objects marked.

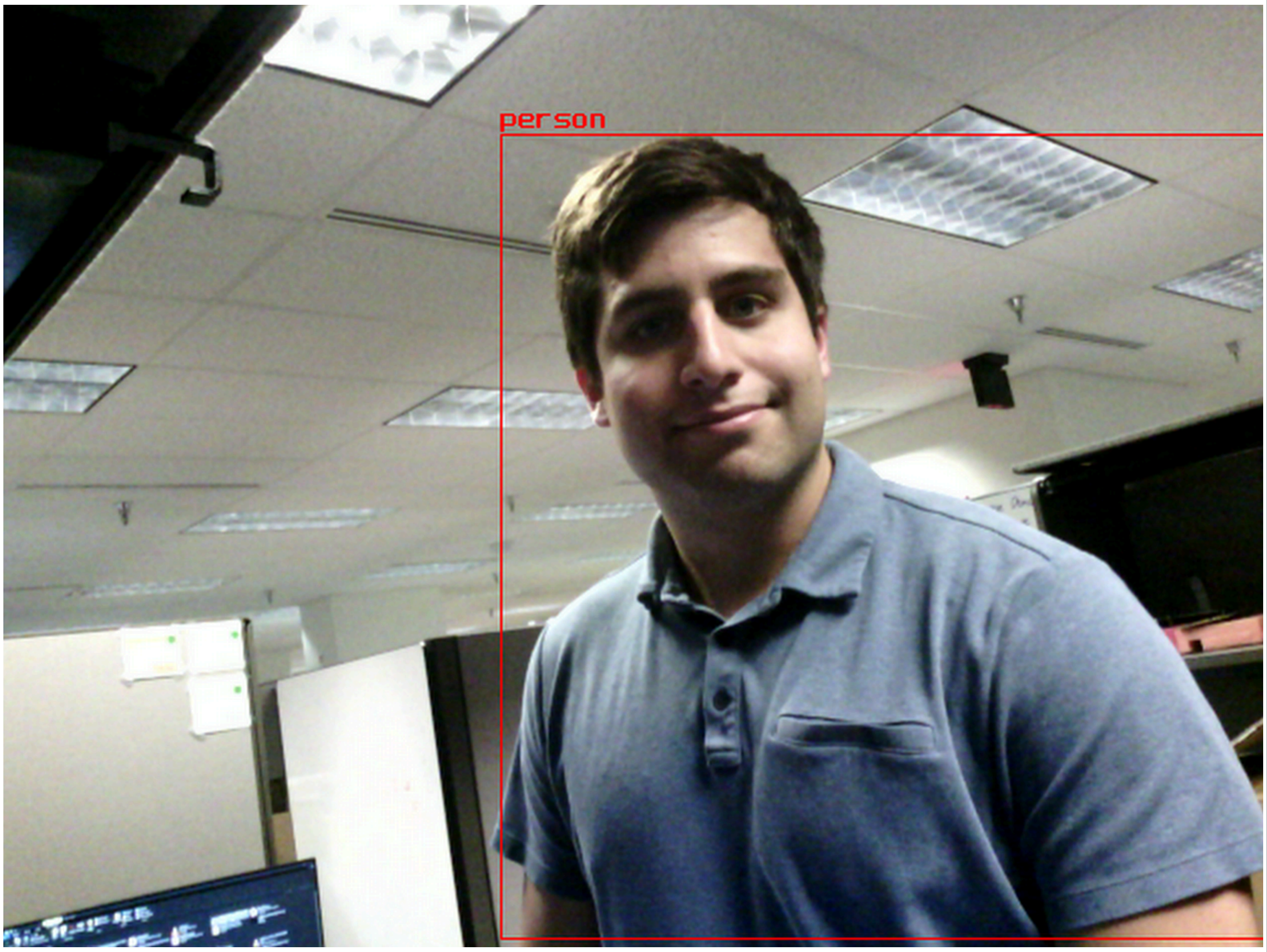


Figure 12. ML gateway client output

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 MobileNet V2 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 against faces saved previously.

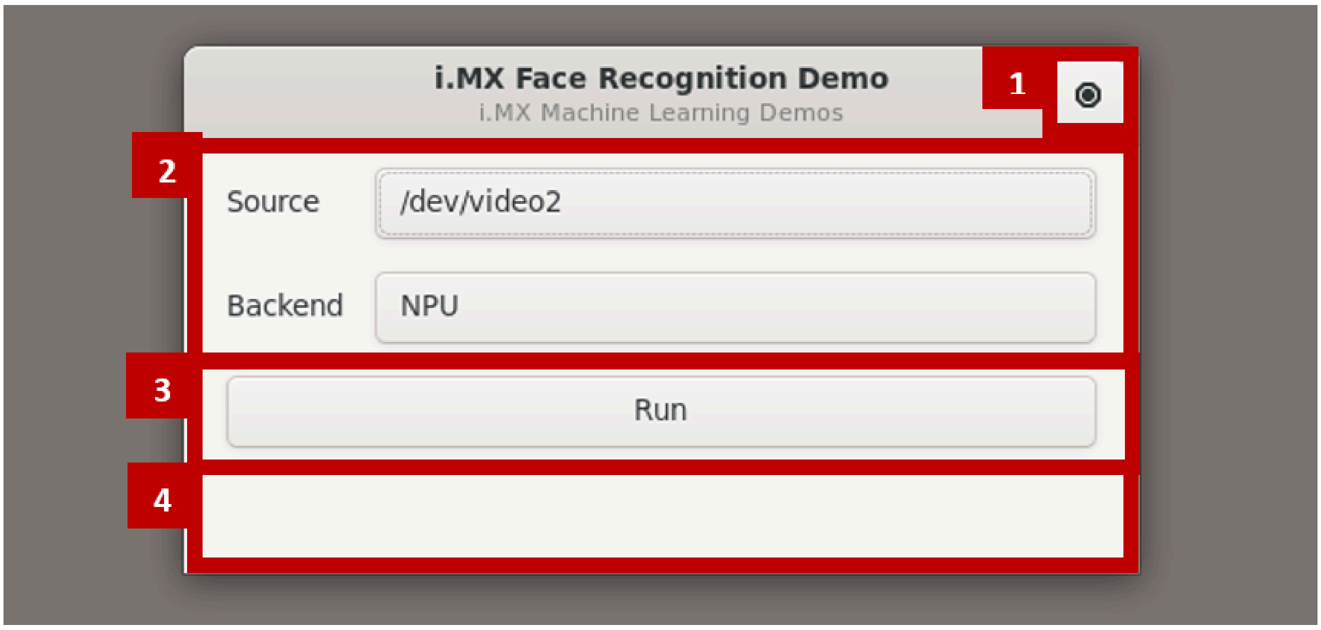
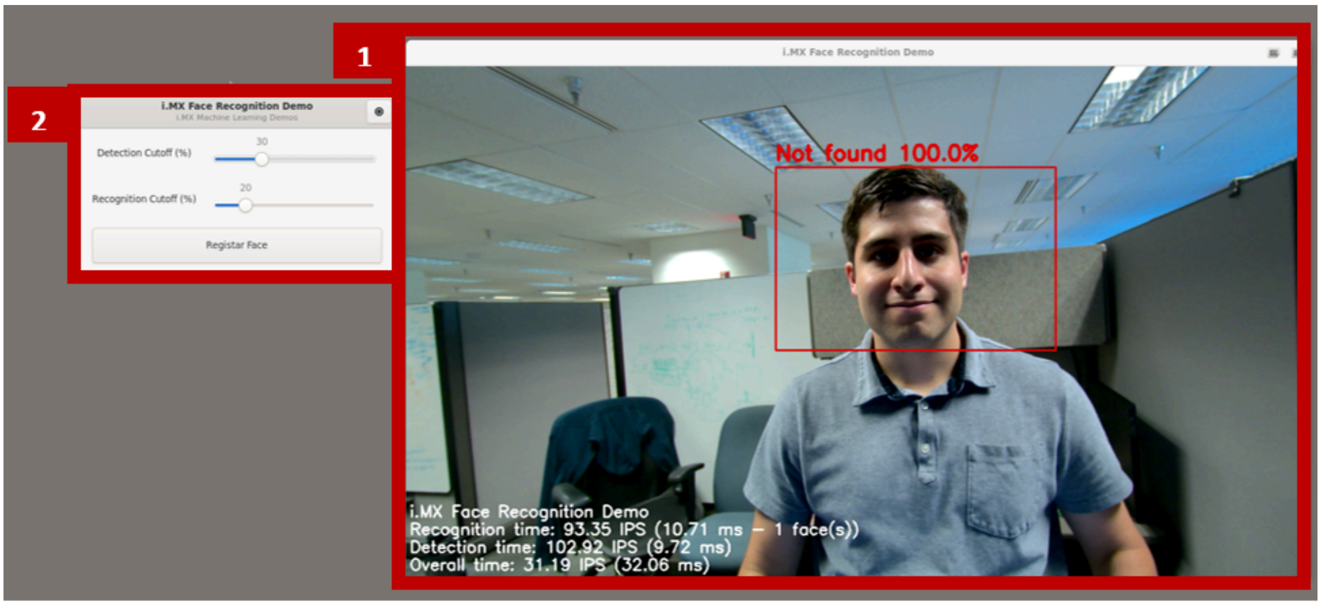


Figure 13. Face recognition options select

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 "warm-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.

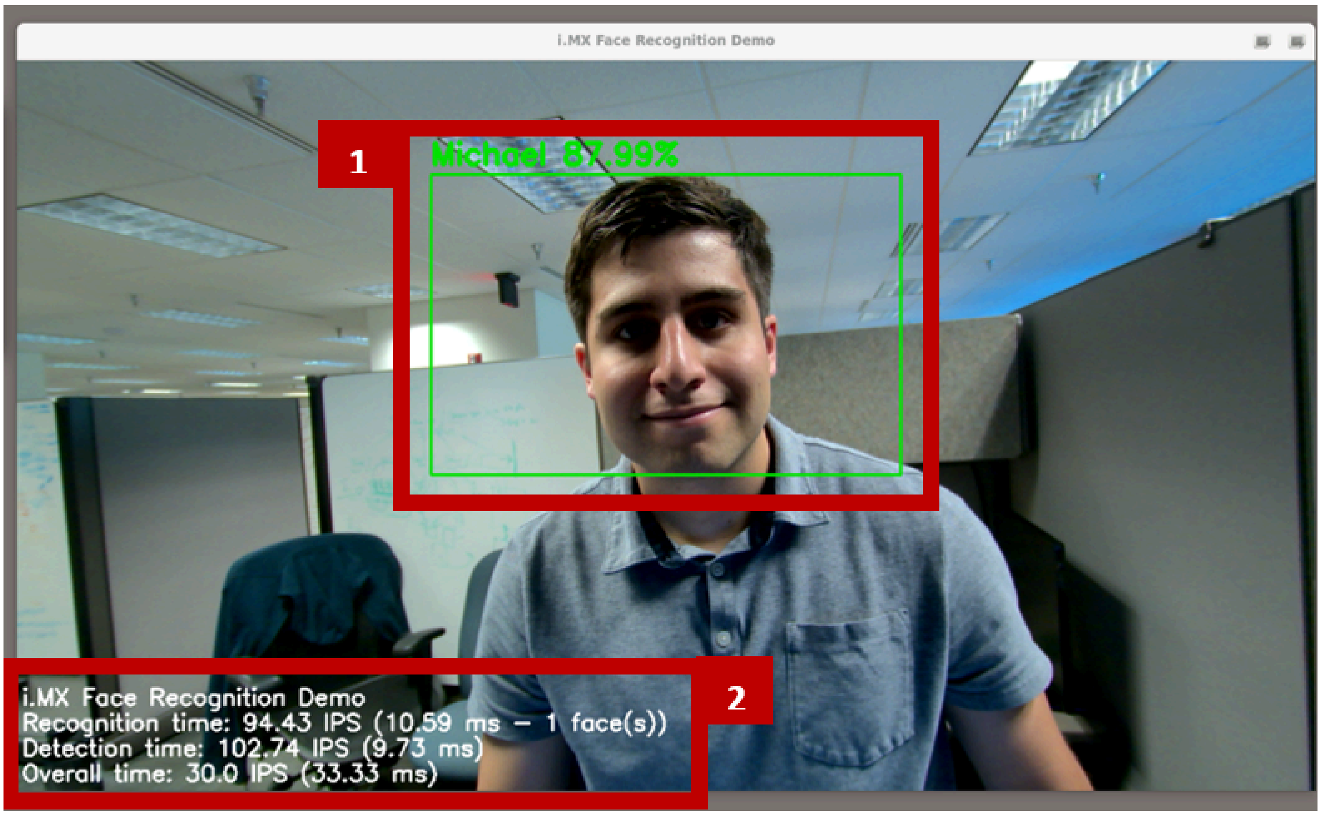

1
2
i.MX Face Recognition Demo
i.MX Machine Learning Demos
Detection Cutoff (%)
30
Recognition Cutoff (%)
20
Registar Face
i.MX Face Recognition Demo
Not found 100.0%

```
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 14. 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 15. Face recognition camera view

The camera view window has the following information:

- Any faces that are found 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 face detected is similar to any of the registered faces, the name of this registered face is displayed above the bounding box, together with the percentage similarity. Otherwise, the "Not found" label is displayed instead, implying that there is no match.
- The following statistics are shown:
  - **Recognition time:** The IPS and inference time in the milliseconds that it took to generate a face embedding for the faces in the picture. Since there can be multiple faces in the frame, the total time for all faces varies based on the number of faces in the frame.
  - **Detection time:** The IPS and inference time in milliseconds that it took to identify all the faces in the picture.
  - **Overall time:** The IPS and inference time in milliseconds that it took to prepare a video frame from when the application received it.

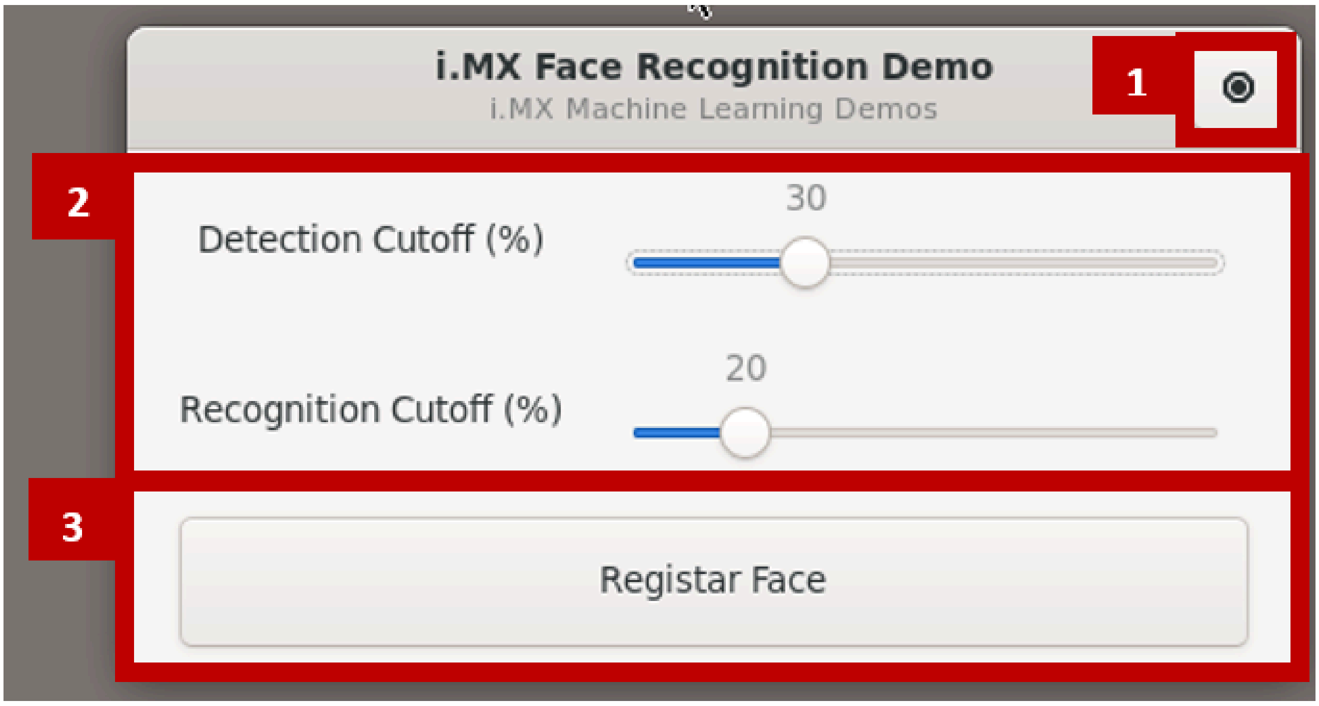


Figure 16. Face recognition control panel

The control panel has the following controls to change the behavior of the demo:

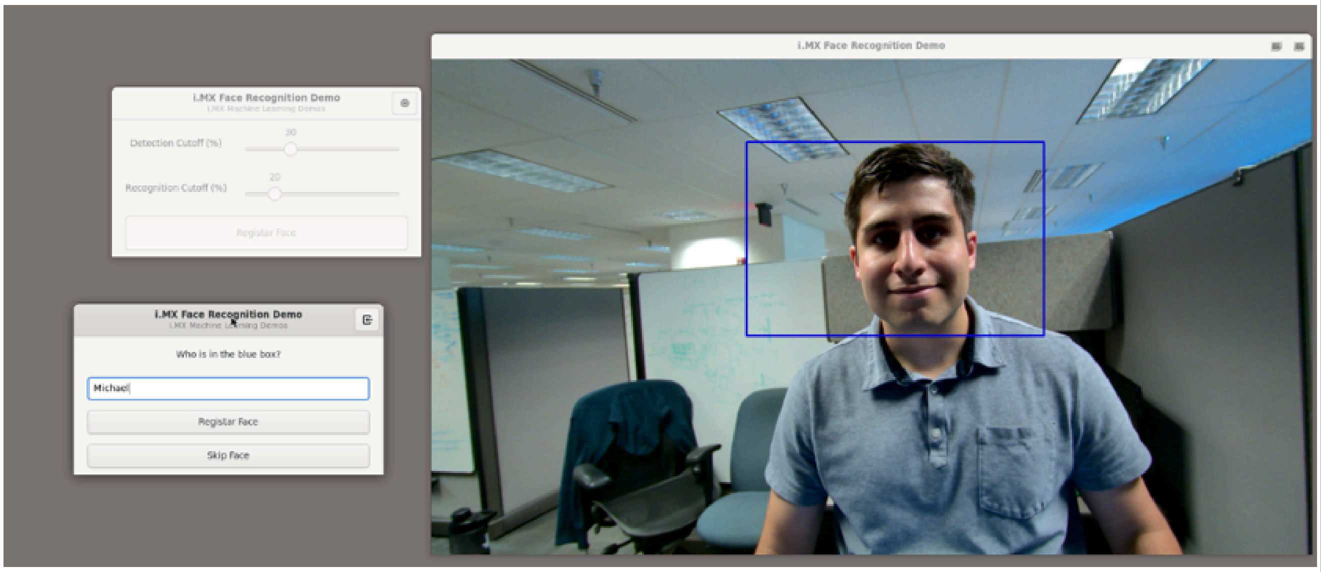
- **Detection Cutoff (%):** It sets the percentage confidence required to detect a face in a video frame. Slide this option up if other objects are being detected as faces or down if faces are not being detected.
- **Recognition Cutoff (%):** It sets the percentage similarity required to recognize a 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. This 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 17. 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: *Not always able to identify faces? Registering the same face multiple times helps increase the data pool the database has to pull from.*


i.MX Face Recognition Demo
i.MX Machine Learning Demos
Registar Face
i.MX Face Recognition Demo
i.MX Machine Learning Demos
Registar Face
Skip Face
i.MX Face Recognition Demo

Detection Cutoff (%) 30

Recognition Cutoff (%) 20

Who is in the blue box?

Michael

Figure 17. Face recognition for a registered face

3.1.3 TensorFlow Lite demos

TensorFlow Lite (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.

When running the ML benchmarking demo, two machine learning model files representing the same model are run on both the CPU and NPU. Before running, the demo attempts to download these models from the Internet. After a successful download, the models are made to run and the times are compared on the screen. Currently, the only model that can be run is a pretrained quantized MobileNet V1 TFLite model.

3.1.3.2 Driver monitoring system demo

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

Newer cars today have advanced navigation systems that can even drive the car for the user. While these systems are tested, no system is perfect and can require the driver to intervene. To ensure safe operation of these advanced systems, car manufacturers can implement smart safety systems to keep everyone on the road safe. One of these systems is a Driver Monitoring System (DMS). DMS watches the driver to ensure that they have their eyes on the road and are not displaying symptoms of fatigue. This demo shows an example implementation of a DMS demo. A camera and Internet connection are required for this demo.

When starting the demo, users can select a video device to use for the demo. Select the correct device from the dropdown menu and click "Start" to start the demo. The options window downloads any required files and starts the camera feed. On the left bar, the user's attention, doziness, and yawn status can be monitored. At the top, there is a large colorful bar that increases if the user does unsafe activities (leaves the frame, looks away, falls asleep) and decreases if the driver is safe. On the right, the driver that has been detected has a box around their face. The penalties for these unsafe actions can be adjusted by clicking the button on the top left of the window.

3.1.3.3 Selfie segmenter

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

This demo uses a machine-learning-based semantic segmentation model to segment the pixels from the input frame containing a person, which then can be used to replace or modify the background of the image. The architecture of the model is based on MobileNetV3 with added customized decoder blocks for segmentation.

There are two versions of this model:

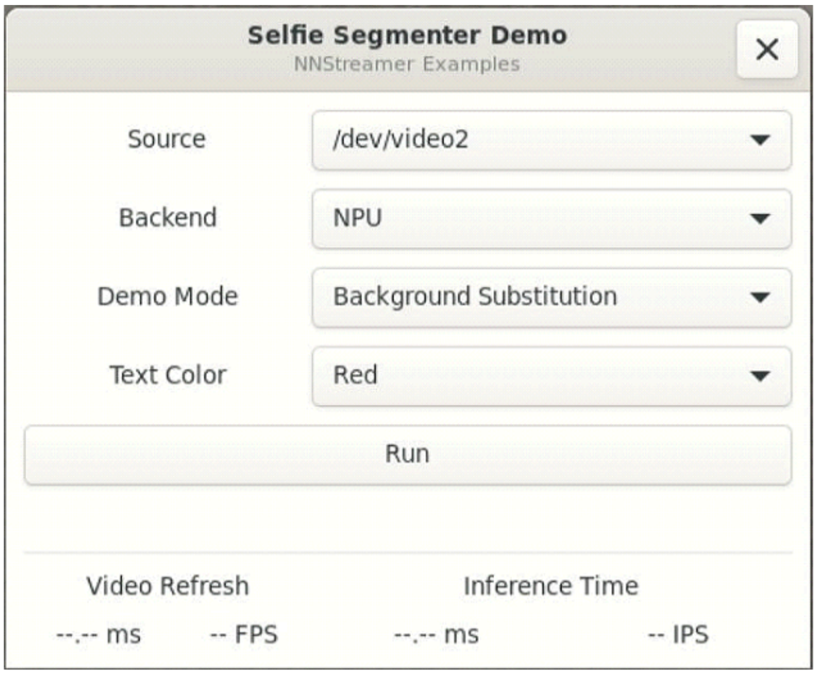
- General: The general model expects an input shape of [1, 256, 256, 3], containing an RGB image with the input value range of [0.0, 1.0]. Its output shape is a grayscale mask [1, 256, 256, 1] with the output value range of [0.0, 1.0], where 0.0 means that no person has been predicted in that pixel, and 1.0 contains a part of the person.
- Landscape: The landscape version has the same input and output ranges, but the tensor shapes are [1, 144, 256, 3] and [1, 144, 256, 1], respectively.

These models are part of MediaPipe solutions, which can be found at MediaPipe - Image Segmentation Guide. The general version of the model is used on the i.MX 8M Plus and the landscape version of the model is used on the i.MX 93.

Origin:

- Model card: Selfie Segmenter Model Card
- Solution: MediaPipe - Image Segmentation Guide
- Tingbo Hou, Siargey Pisarchyk, and Karthik Raveendran from Google have created this model.
- MediaPipe models are licensed under Apache-2.0 License.

Running the demo: This demo is intended to work with people as the foreground object. People too far away from the camera can result in bad segmentation. A camera and Internet connection are required for this demo.



Selfie Segmenter Demo
NNStreamer Examples

Source: /dev/video2
Backend: NPU
Demo Mode: Background Substitution
Text Color: Red

Figure 18. Selfie Segmenter options window

Select the source that must be used for the demo, and the back-end (CPU or NPU).

Two modes are available for this demo:

- Background substitution
- Segmentation mask

The first one segments the background from the person and replace it with a conference room image shown in Figure 19. The latter shows the input frame and the segmentation mask side-to-side shown in Figure 20.

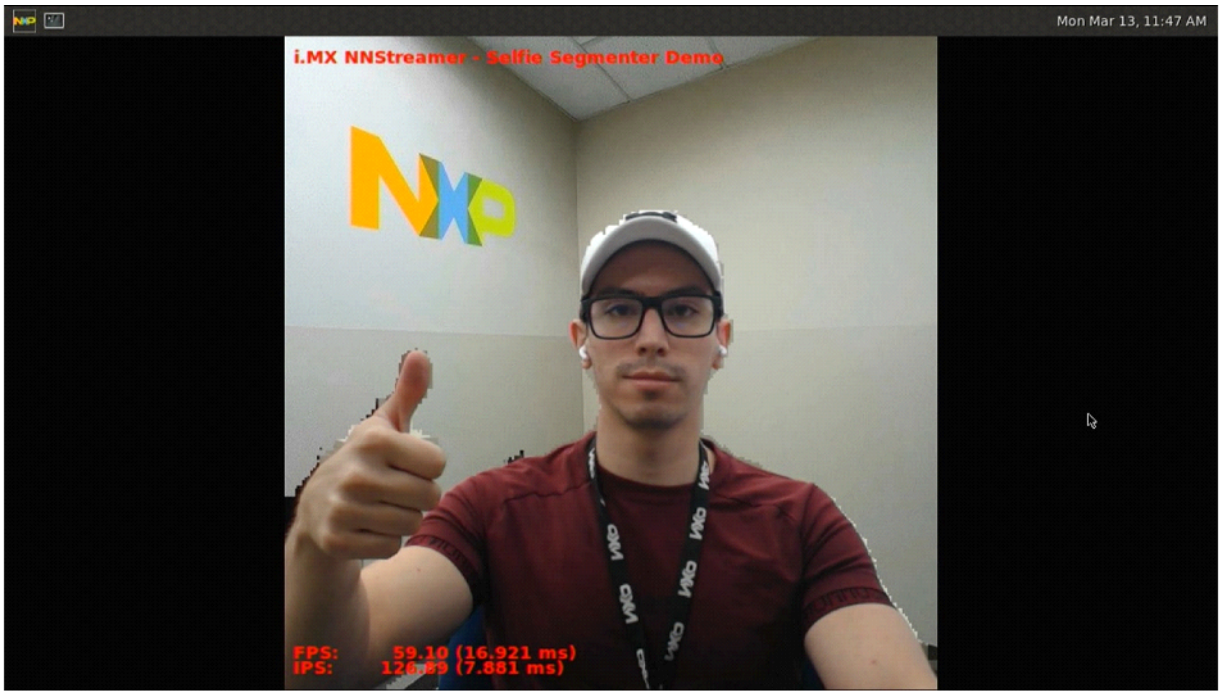


Figure 19. Selfie segmenter – Background substitution mode on the i.MX 8M Plus

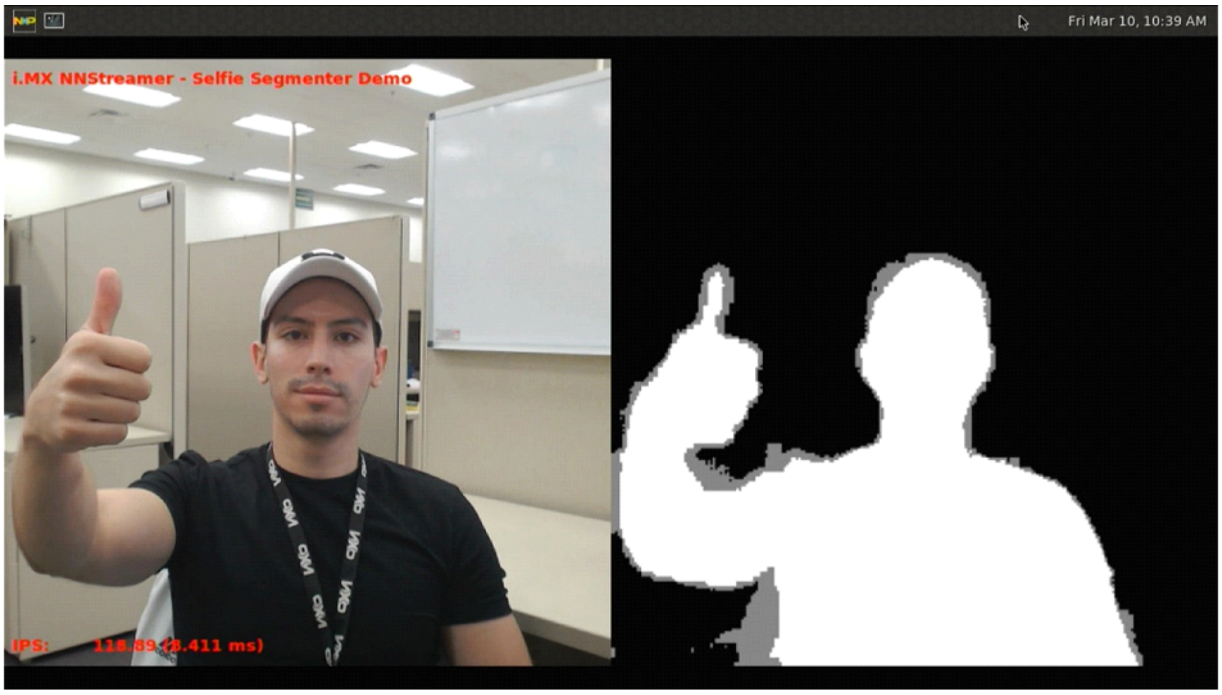


Figure 20. Selfie segmenter – Segmentation mask mode on the i.MX 8M Plus

When the demo starts, a video feed showing either the input frame with the replaced background or the segmentation mask side-to-side appears, depending on the selected mode. In the bottom-left corner of the screen, you can see the IPS and FPS.

Note: *If using a USB camera, make sure it can deliver input frames to the pipeline fast enough. Tests have shown that USB 2.0 cameras struggle to provide 30-FPS speed. A warning log can be displayed as follows:*

Warning: *[174.028300] xhci-hcd xhci-hcd.2.auto: WARN: HC couldn't access mem fast enough for slot 1 ep 2.*

This log does not appear when using a USB 3.0 type-C camera or the OV5640 camera.

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 a camera (optional)

This demo allows the user to test display outputs and cameras connected to the camera. When launched, users see the GUI shown in Figure 21, 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.

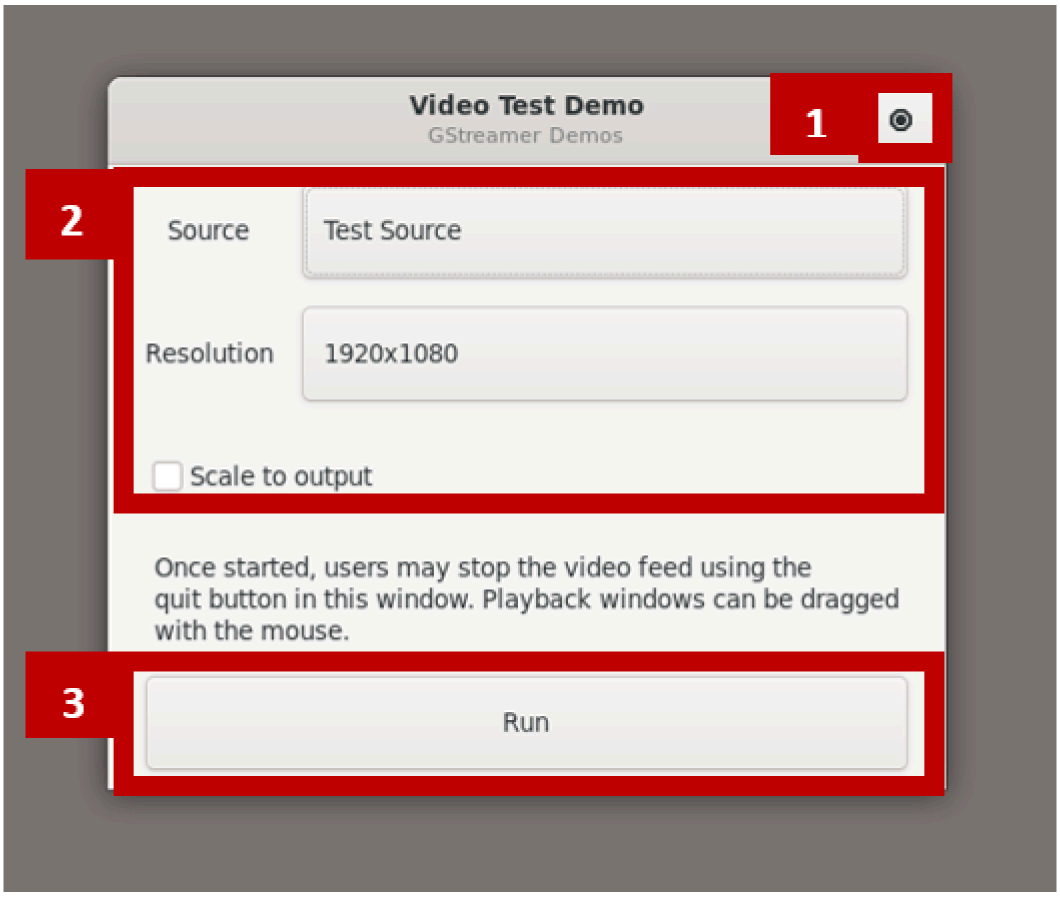

Video Test Demo
GStreamer Demos
1
2
Source
Test Source
Resolution
1920x1080
Scale to output
Once started, users may stop the video feed using the quit button in this window. Playback windows can be dragged with the mouse.
3
Run

Figure 21. Video test demo

The following options are available to users while running the demo, as shown in Figure 21:

1. This button quits the demo and closes all GStreamer windows that are running at the current time.
2. The following 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. For example, if 720x480 is selected as the resolution and the display is 1920x1080, first the video output is captured at or scaled down to keep the same format for consistency: "720x480" and then 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, make sure 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.

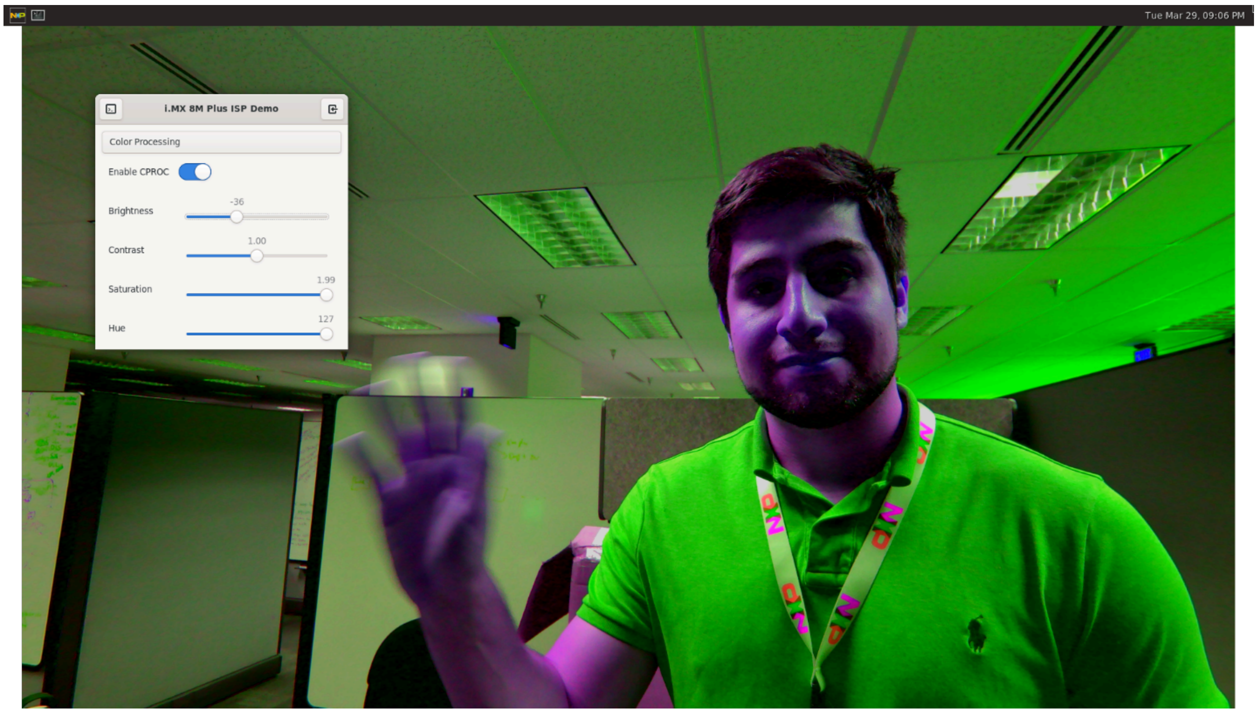
3.2.2.1 ISP control demo

Required materials: Mouse, display output, and a 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`)



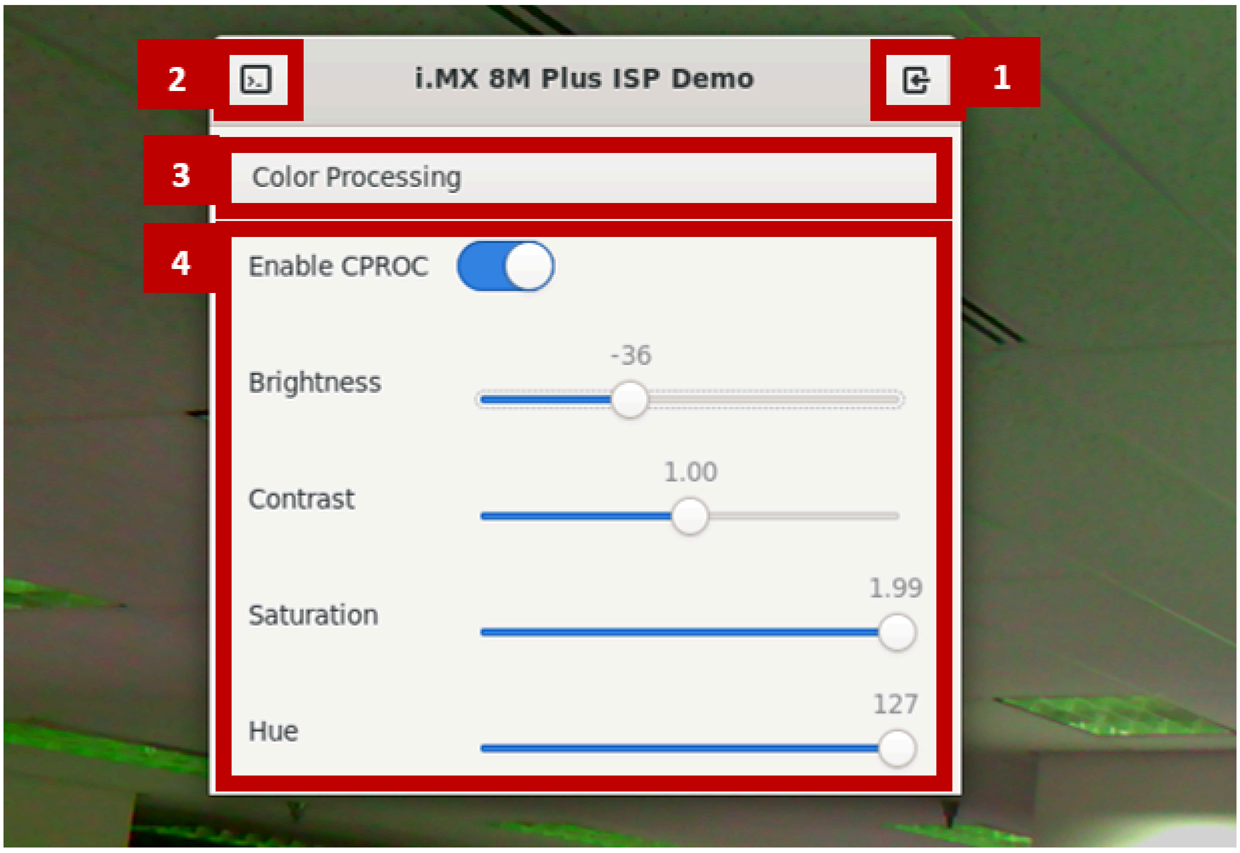
Enable CPROC: on
Brightness: -36
Contrast: 1.00
Saturation: 1.99
Hue: 127

Figure 22. 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)



Brightness -36
Contrast 1.00
Saturation 1.99
Hue 127

Figure 23. ISP control demo control panel

The ISP can be controlled using the control panel shown in Figure 23 that 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.

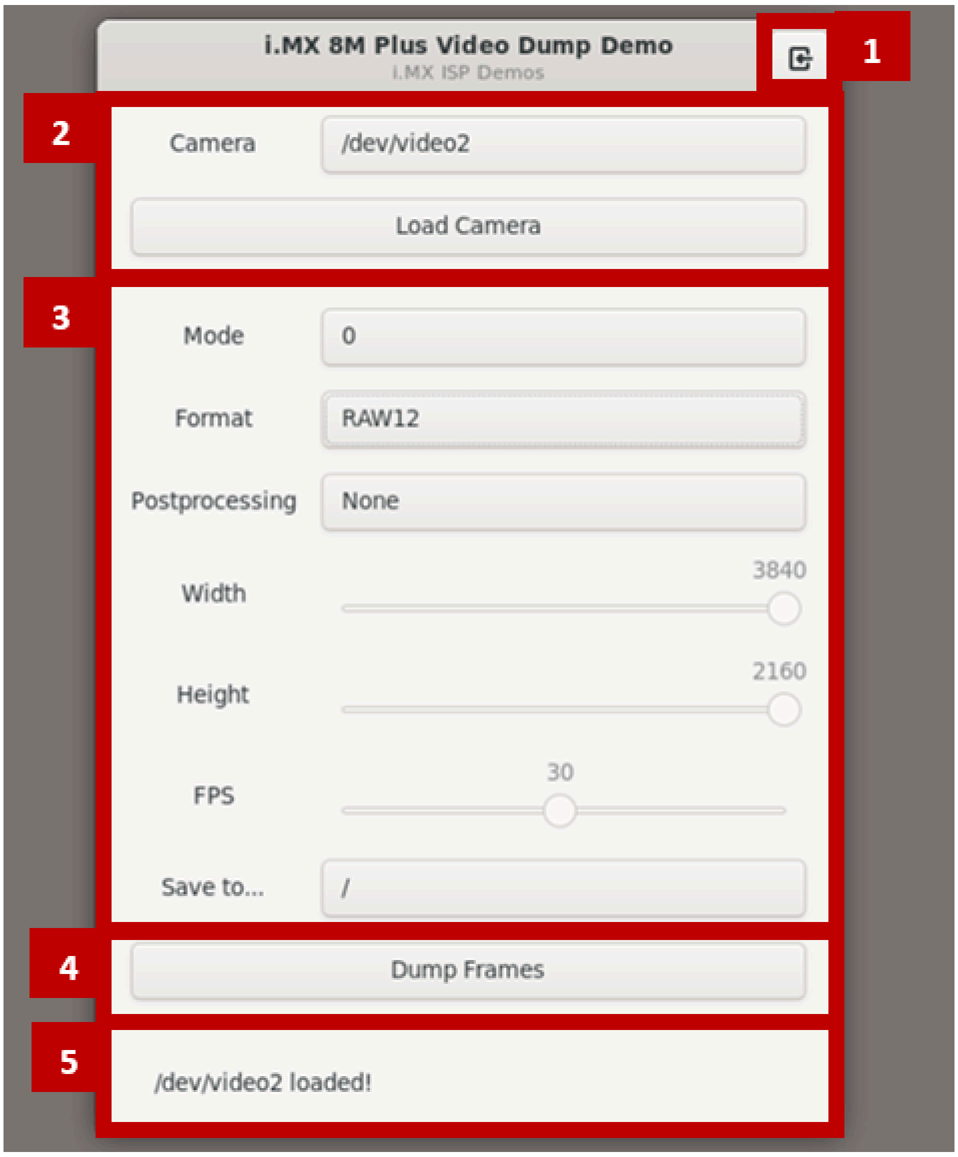
3. This dropdown allows the user to swap between different control panel sections.
4. This area includes the settings and options for the current section. Changes to these settings change the video feed playing in the background and then revert when the demo is exited.

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 24. Video dump example

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

1. This button can be used to quit the demo.

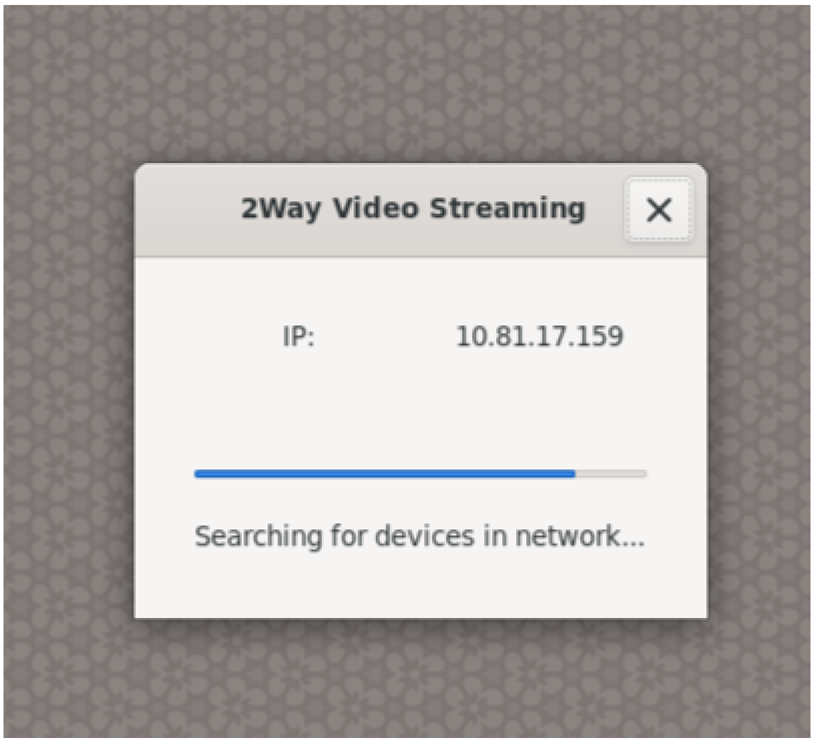
2. This area selects the camera to use and loads it into the application. To load the camera, select the camera from the **Camera** dropdown and click **Load Camera**. It gets the supported modes and tests whether the camera can be run with this demo. Once the camera is loaded in, the other options in 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.*
   - **Post-processing:** User sets modifications to the image if supported. "Crop" crops a frame while "scale" scales the frame to the selected size.
   - **Width:** Shows the width of the selected mode. If post-processing is enabled, it allows users to change the width of an image.
   - **Height:** Shows the height of the selected mode. If post-processing is enabled, it allows users to change the height of an image.
   - **FPS:** The frames per second of a selected mode. It is not 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.



Searching for devices in network...

Figure 25. 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 device the user is looking for has not been listed in the dropdown during the initial search, a **Search Again** button is available to find a targeted device in the network.

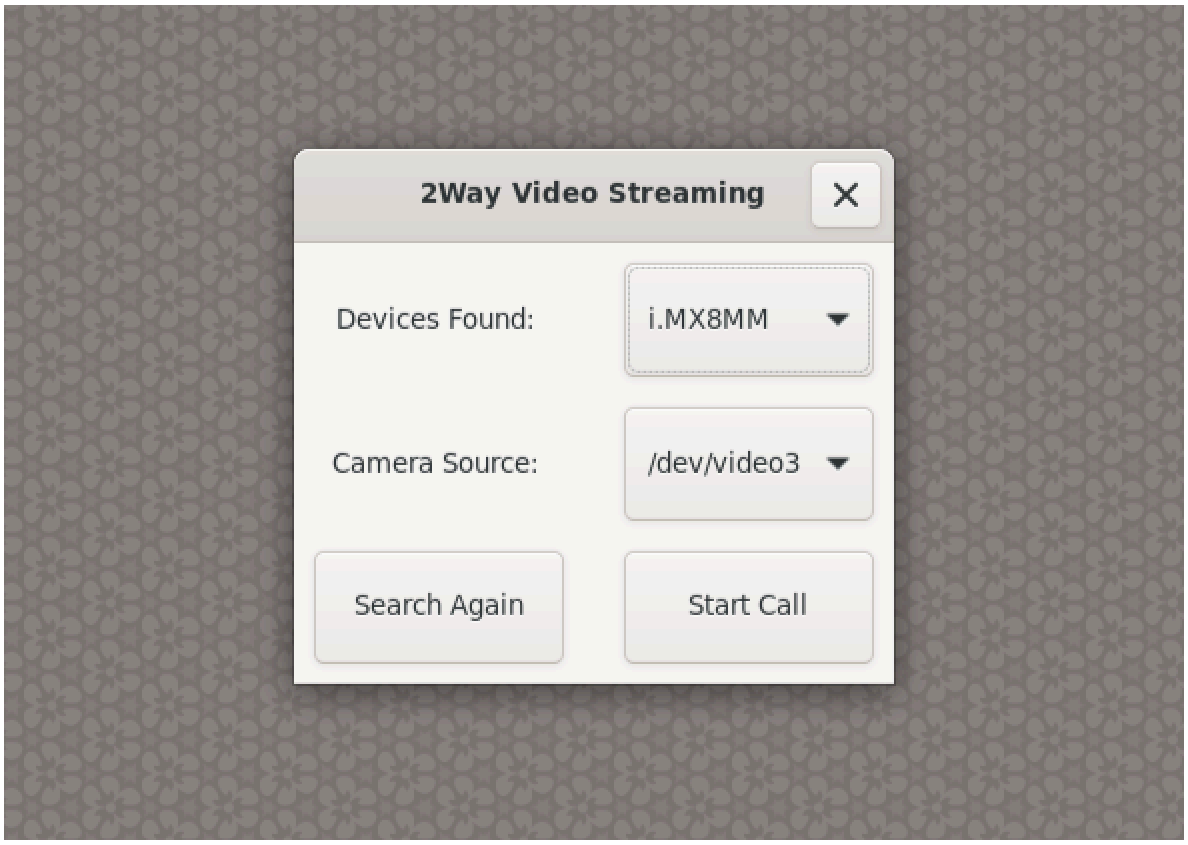


Figure 26. Menu for 2Way Video Streaming

When both the EVKS identifies the two targeted i.MX devices, start a video stream between them. Figure 27 shows an example for video stream between two i.MX devices in the local network.

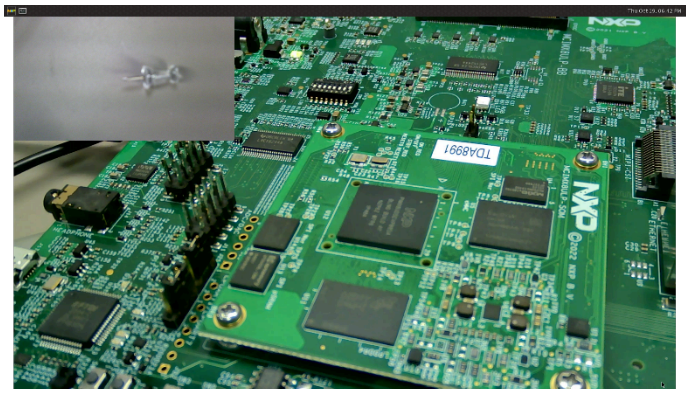


Figure 27. Video stream from 2Way Video Streaming

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

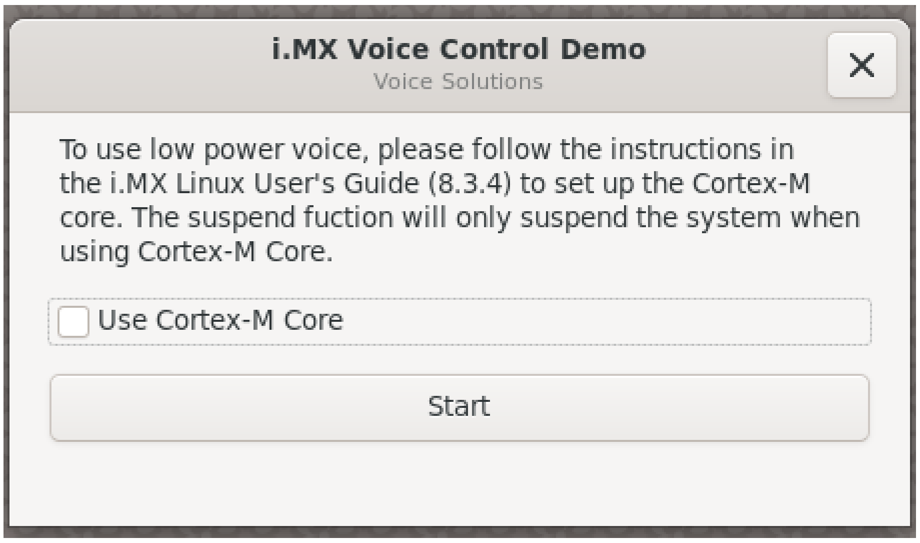
Similar to a mouse and keyboard, voice is a way for us to interact with edge devices. These demos show off NXP's voice technologies such as VoiceSeeker, VoiceSpot, and Voice Intelligent Technology (VIT).

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

You can control the GoPoint for i.MX Applications Processors with the power of your 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 the GoPoint for i.MX Applications Processors. The voice commands are all processed locally, meaning no cloud service is required.

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, you can retrieve the original file from `/etc/asound_org.conf`. This file contains the settings that were applied during the last successful run of the demo.*


i.MX Voice Control Demo

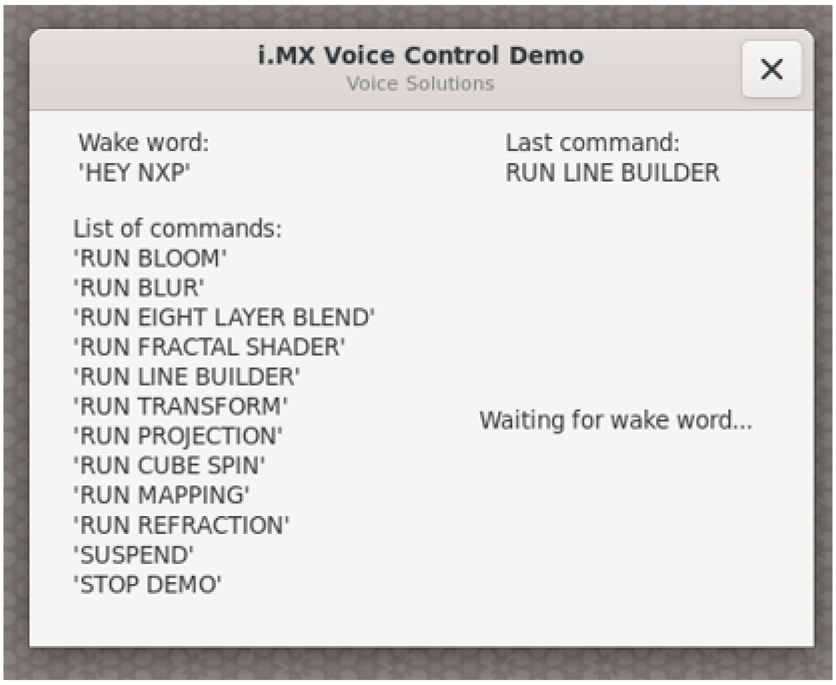
Voice Solutions

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 28. 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, follow the instructions in the "section 8.3.4" of 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 you to manually power cycle the EVK.*


i.MX Voice Control Demo
Voice Solutions

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 29. 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.

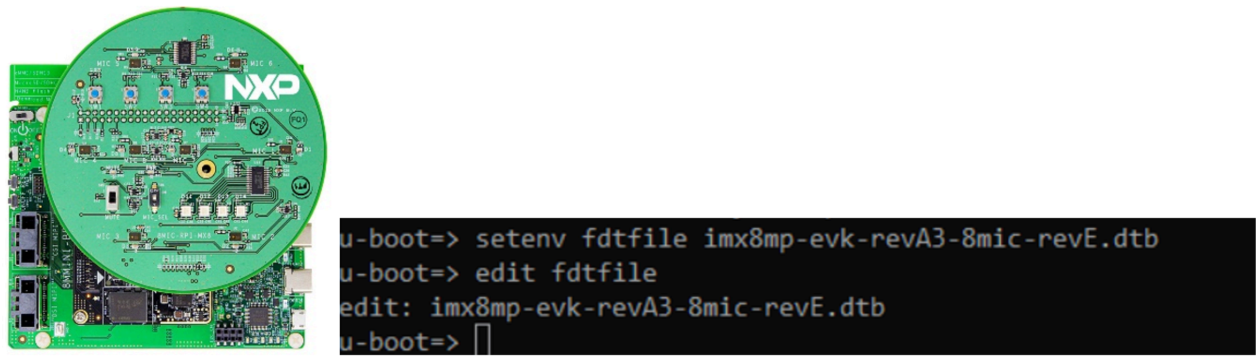
To wake up the board, say "Hey NXP!". When the status indicator switches to "Listening…", say the command that you wish to run. The demo is launched (can take a couple of seconds). To stop the demo, say "Hey NXP!" and say another command or "Stop Demo".

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

The 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.

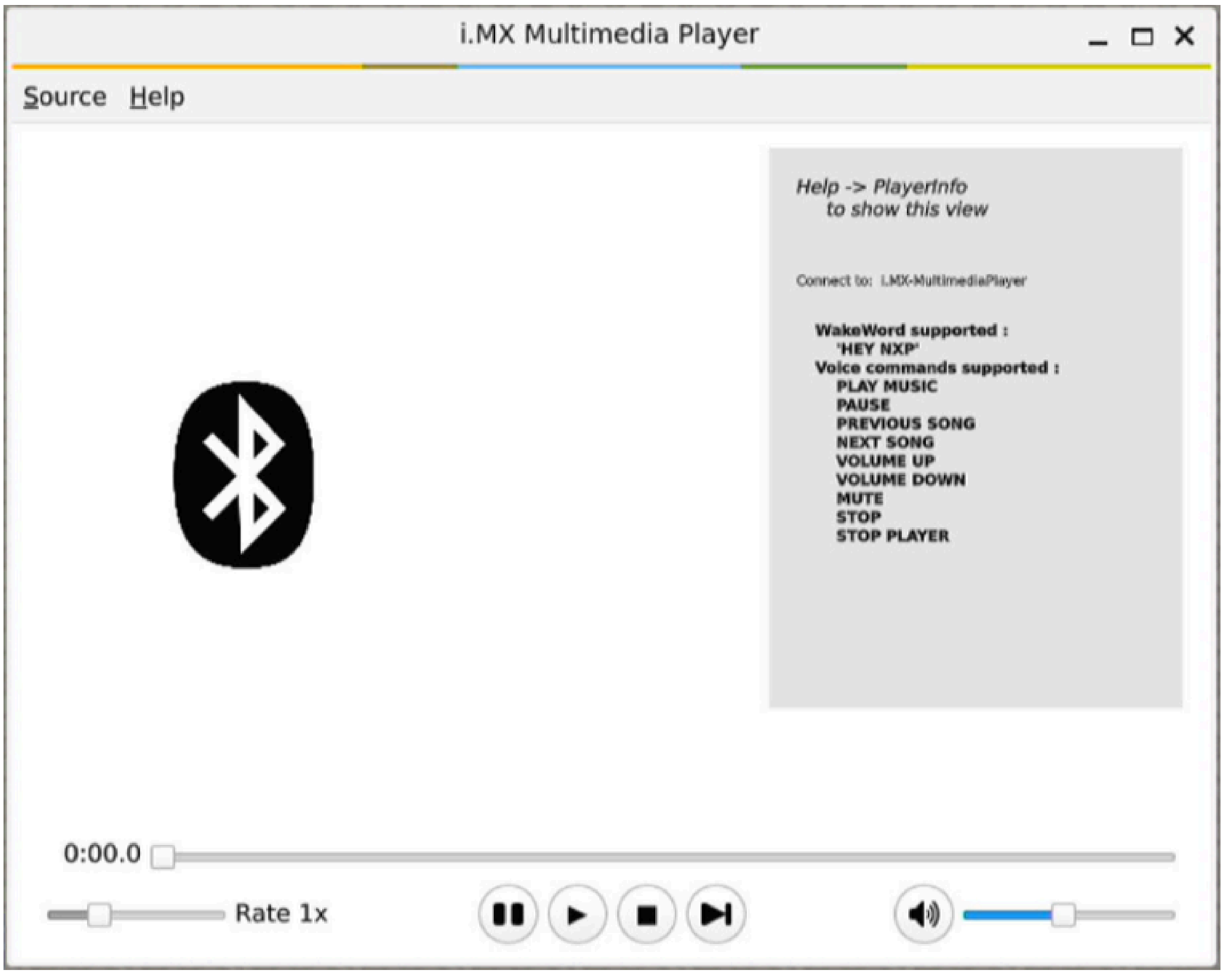
The multimedia player application uses Bluetooth to play back audio and controls Bluetooth commands using VIT voice commands. The application requires the 8MIC-RPI-MX8 installed on the i.MX hardware for voice enablement. The 8MIC-RPI-MX8 requires to set the FDTFILE to a proper 8-microphone board revision 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 30. 8MIC-RPI-MX8 module and required DTB

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

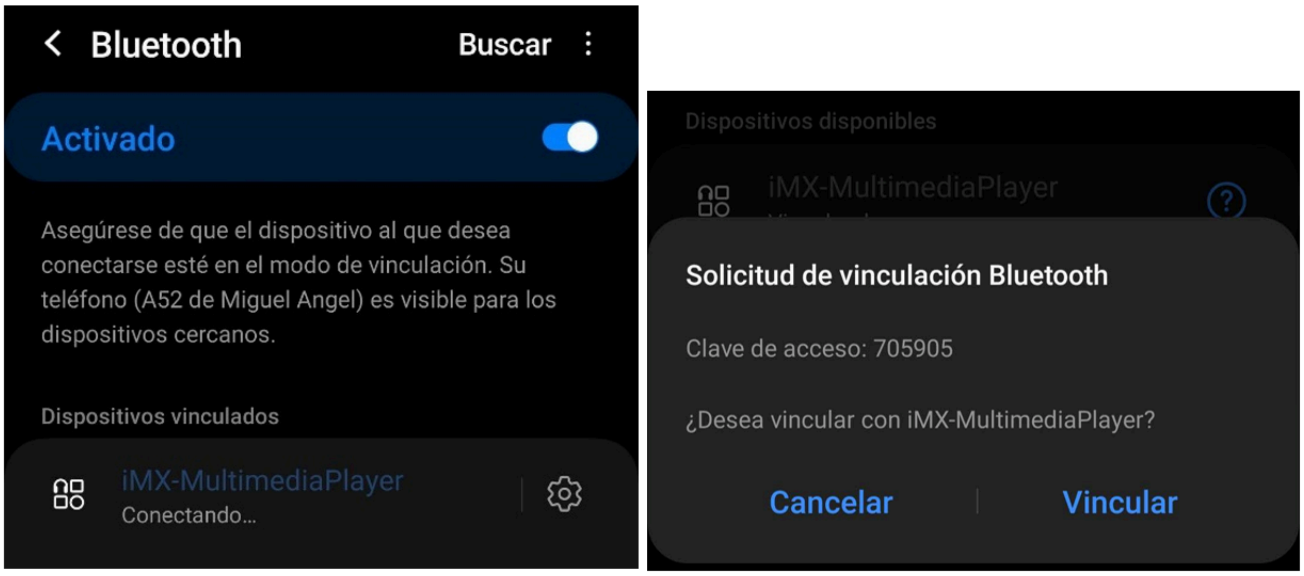


Help -> PlayerInfo
to show this view

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 31. 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.

Solicitud de vinculación Bluetooth

Clave de acceso: 705905

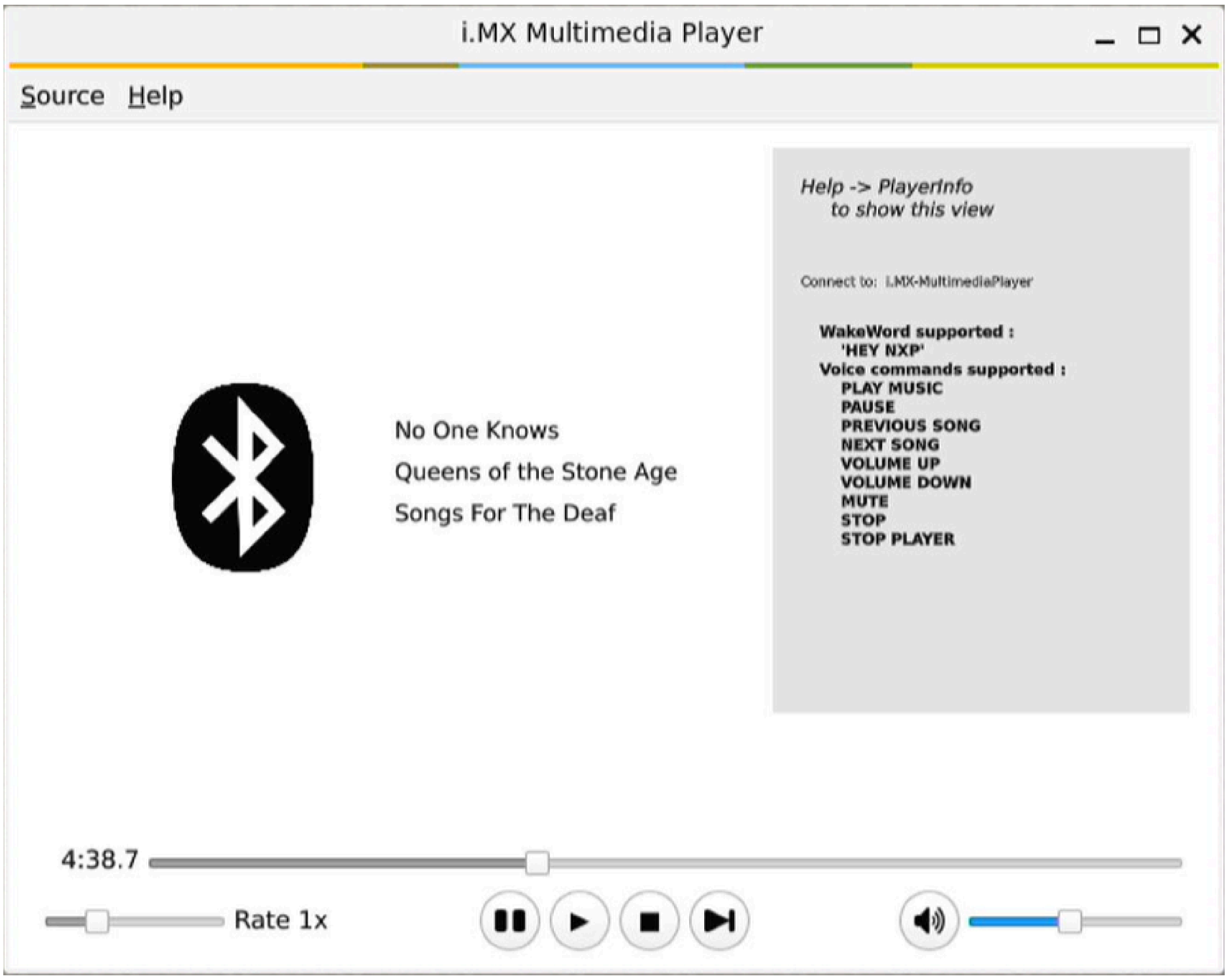
¿Desea vincular con iMX-MultimediaPlayer?

Figure 32. Bluetooth device

Once a Bluetooth device is connected, you can start controlling audio playback using voice commands. Use the "Hey NXP" wake word to let VIT wake up the device and then a voice command to control the audio playback. When a Bluetooth connection is established, you can see the audio metadata displayed in the demo GUI. You 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



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

Figure 33. 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 the NXP VIT supports the voice commands.

To configure a smart kitchen, connect the mouse and display to the board in the corresponding port, depending on the display, mouse, and board you are using. 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 no 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 the GoPoint for i.MX Applications Processors GUI. The smart kitchen GUI shows up on the screen, as shown in Figure 34.

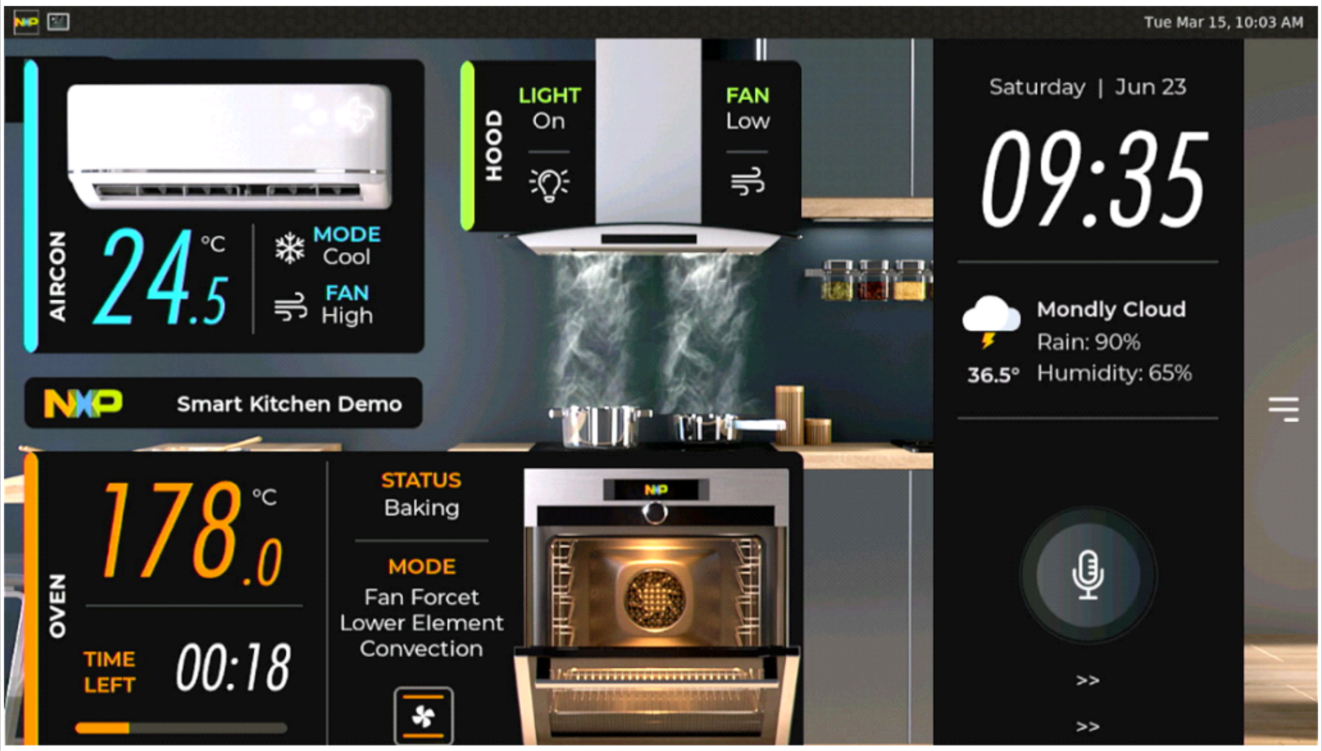


Figure 34. Smart Kitchen GUI

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 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, so it can be used to pause the demo for some time. The GUI can then 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.

3.3 TSN 802.1 Qbv demo

In this demo, we showcase the TSN 802.1Qbv Time Aware Scheduling (TAS) standard with two traffic classes, namely iperf to emulate best-effort traffic and camera streaming to emulate high priority time-sensitive traffic.

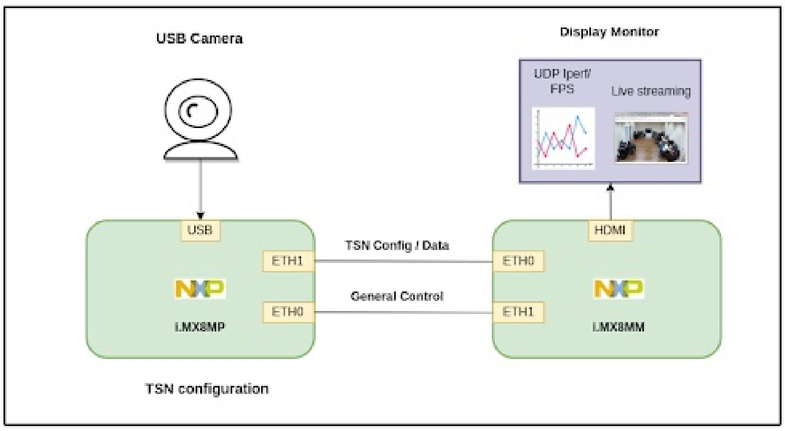


Figure 35. Setup diagram for TSN Qbv

Traffic is streamed between i.MX 8MP and 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 36

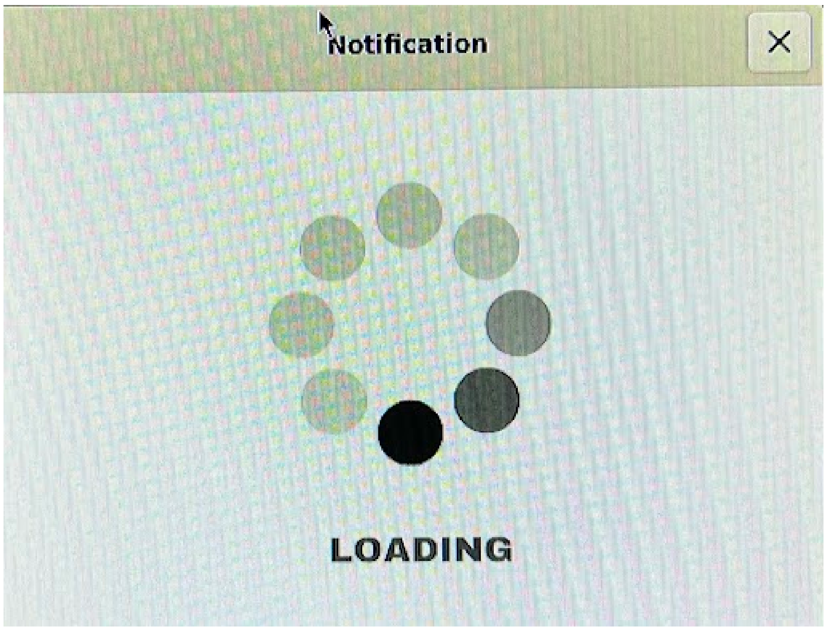


Figure 36. Loading gif

- If the connections are not successful, a pop-up window shows up, as shown in Figure 37. After confirming the connections and system setup, click the **OK** button and relaunch the demo when a camera is connected.

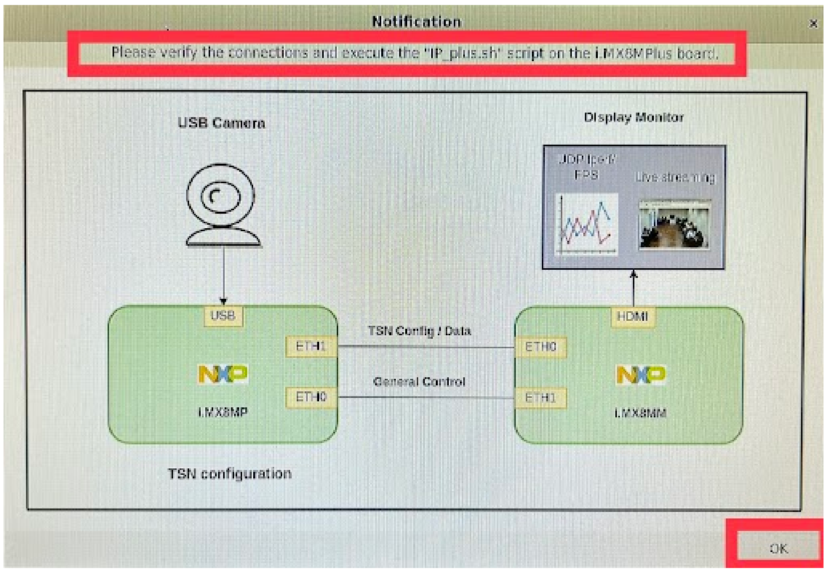
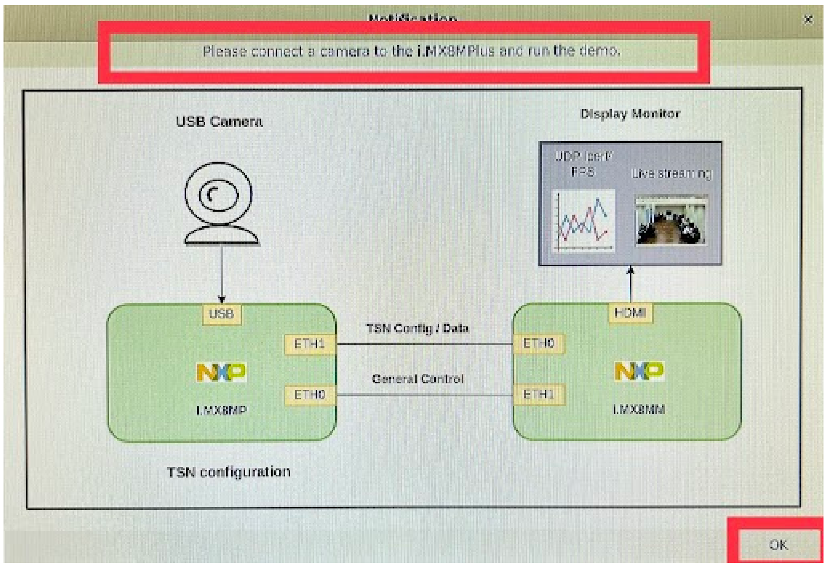

Please verify the connections and execute the "IP_plus.sh" script on the i.MX8MPlus board.

Figure 37. Pop-up window

- If Ethernet connections are successful and the camera is not connected, the window shown in the Figure 38 displays. To get back to the Launch demo window, click the **OK** button.

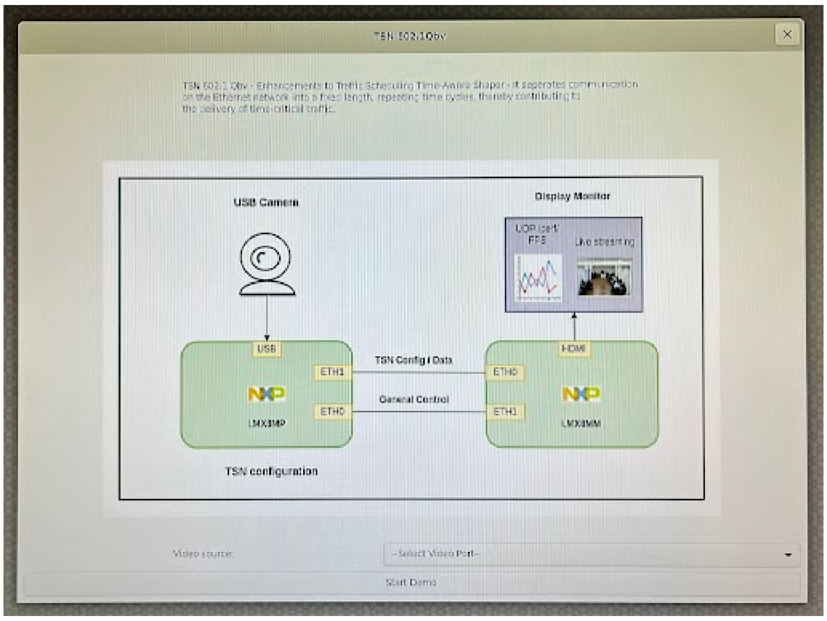


Notification

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

Figure 38. Click the OK button

- If the connections are successful, the TSN Qbv demo login window shows up, as shown in Figure 39.

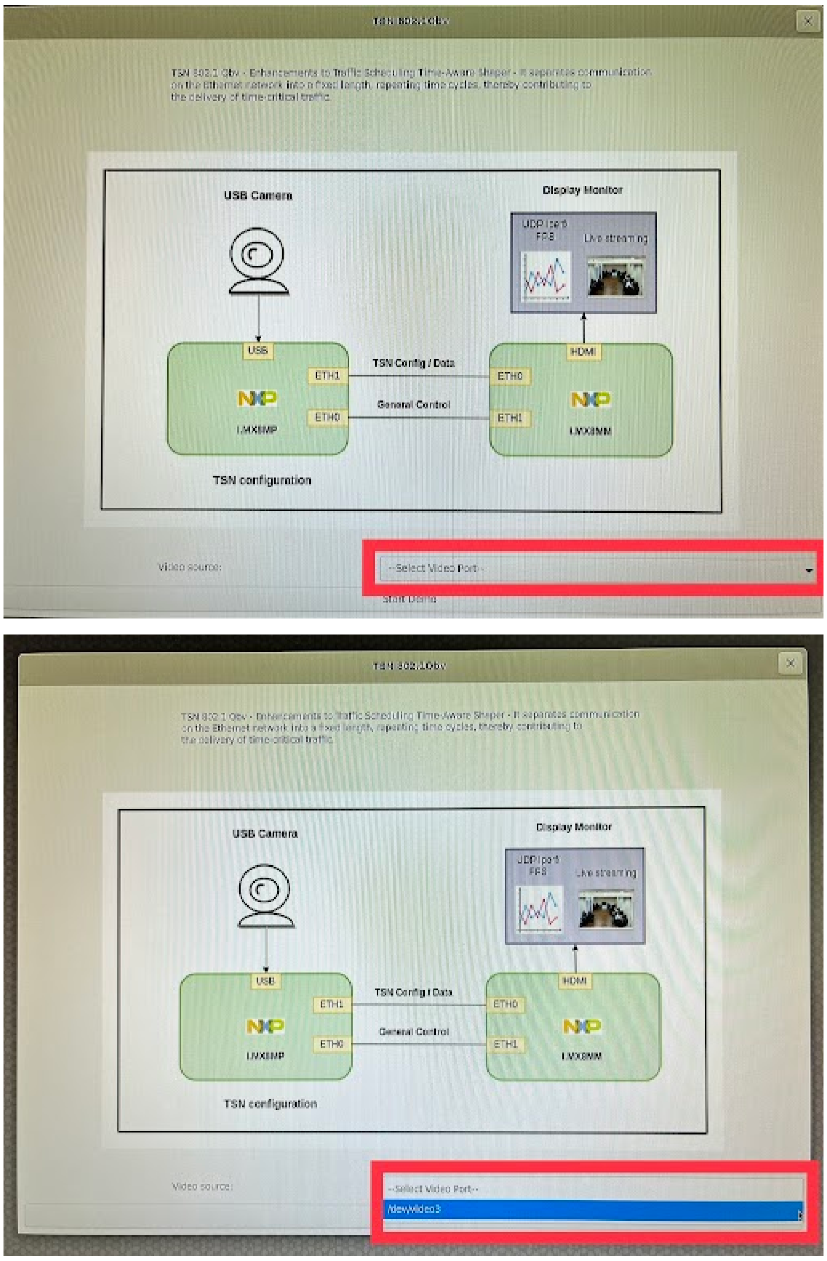


TSN 802.1Qbv

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 39. 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 40), the **Start Demo** button is enabled.



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 40. 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.

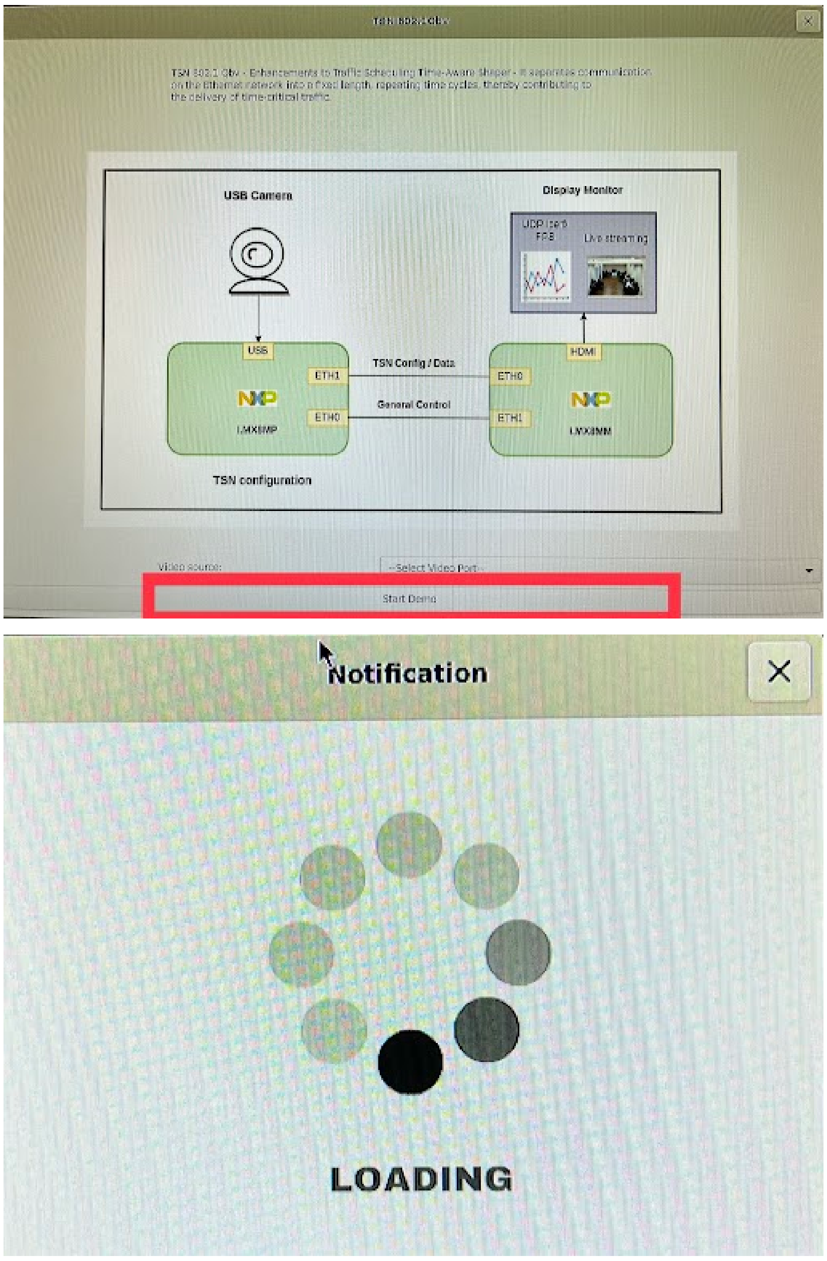

TSN 802.1 Qbv - Enhancements to Traffic Scheduling Time-Aware Shaper - It separates communication on the Ethernet network into a fixed length, repeating time cycles, thereby contributing to the delivery of time-critical traffic.

Figure 41. 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 42.

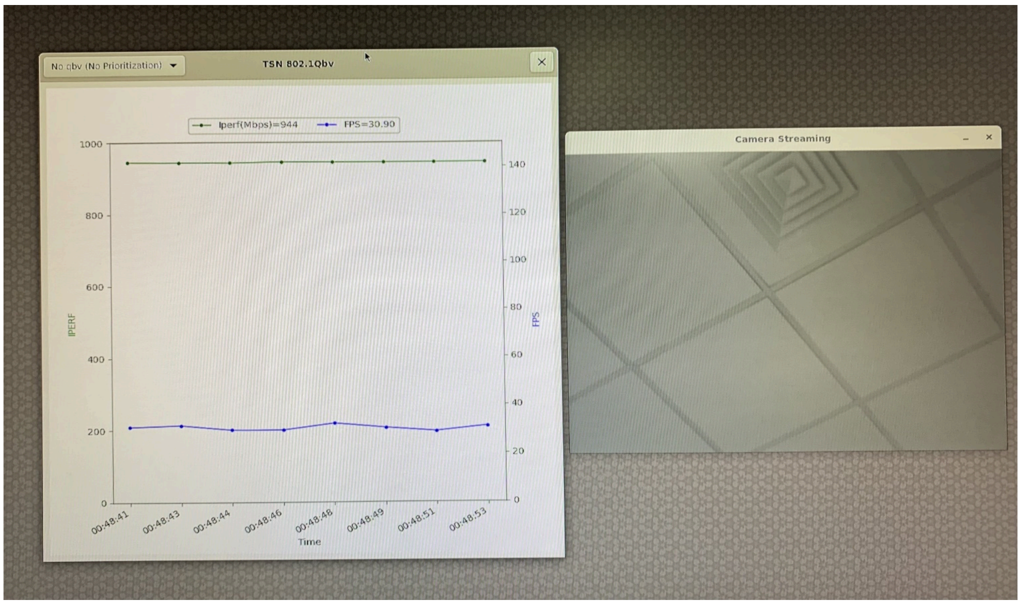


Figure 42. Default graph window and camera window

- 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 43).

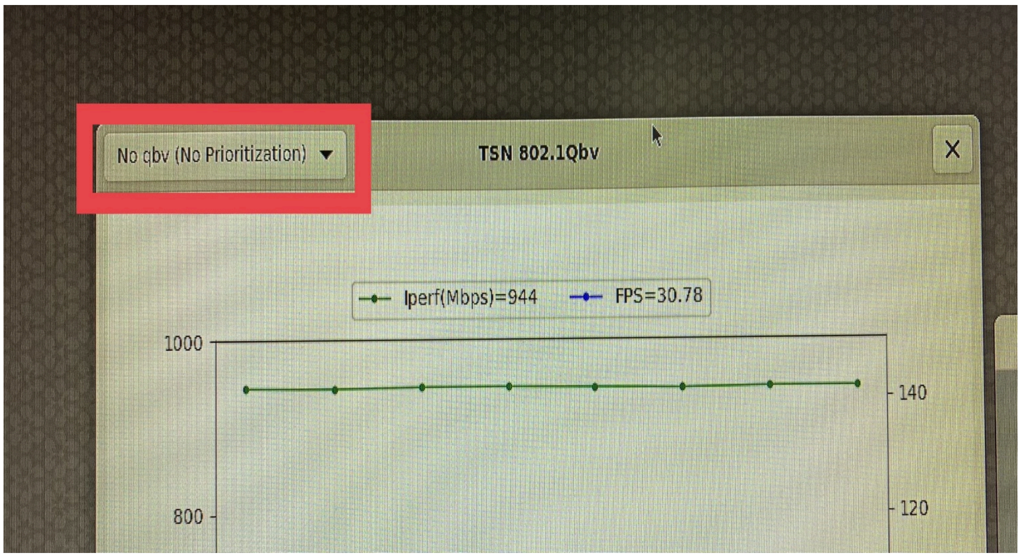


Figure 43. Configuration dropdown button

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

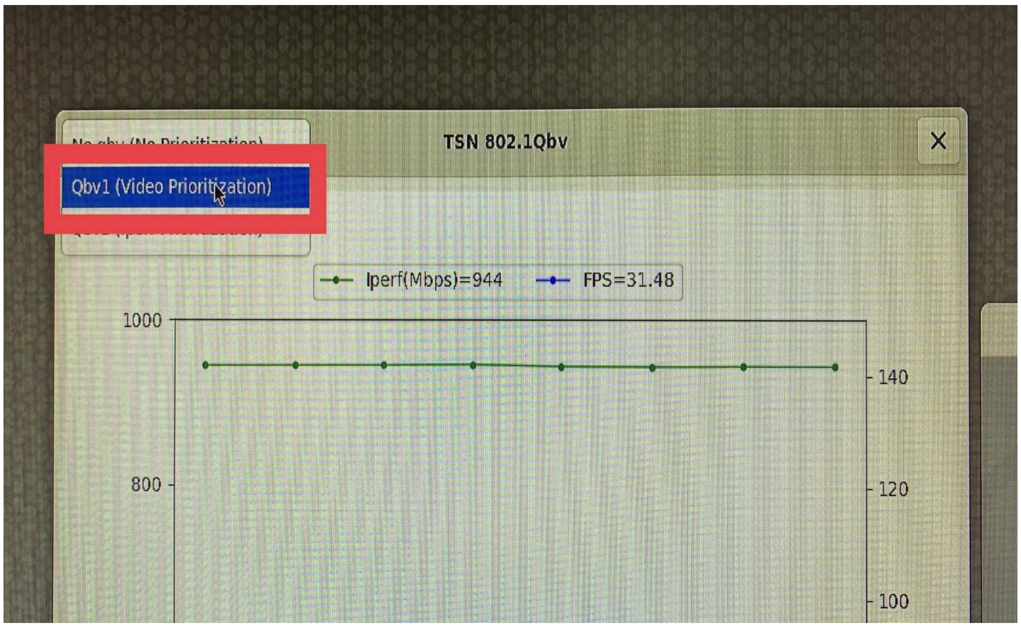
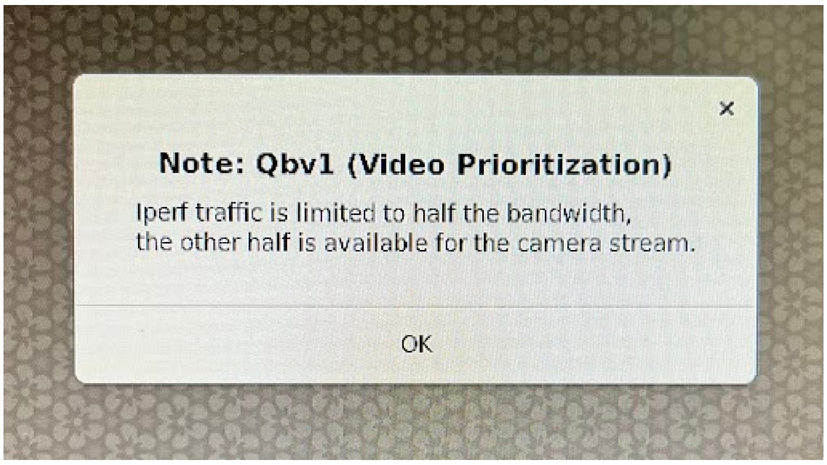


Figure 44. Qbv1 (Video Prioritization) configuration button

- After clicking the **Qbv1 (Video Prioritization)** button, as shown in Figure 44, the pop-up window shown in Figure 45 appears.



Note: Qbv1 (Video Prioritization)

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

Figure 45. 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 46.
- 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.

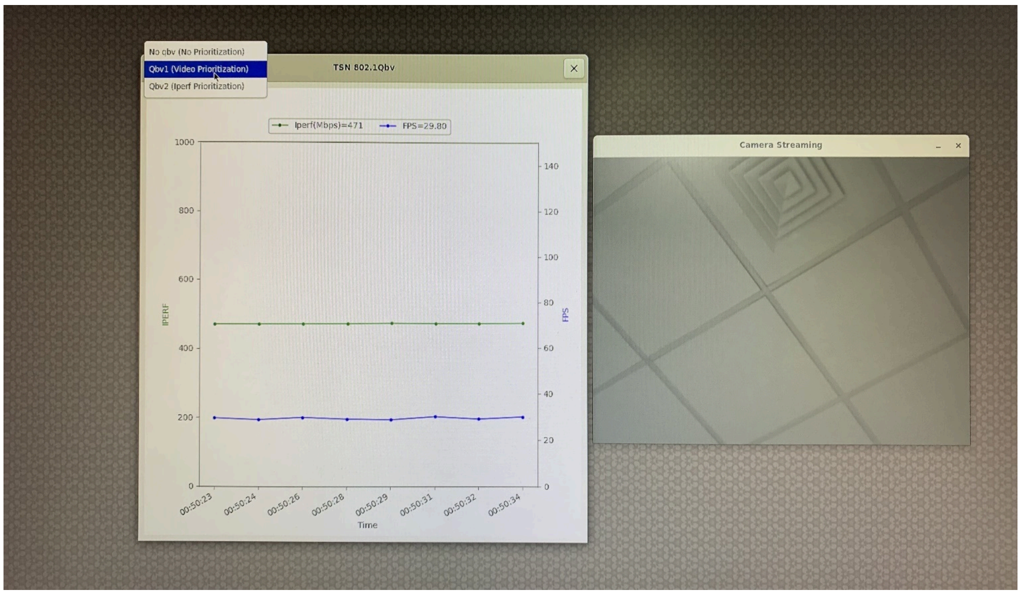

No qbv (No Prioritization)
Qbv1 (Video Prioritization)
Qbv2 (Iperf Prioritization)
TSN 802.1Qbv
Iperf(Mbps)=471
FPS=29.80
IPERF
FPS
Time
Camera Streaming

Figure 46. Qbv1 graph window and camera window

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

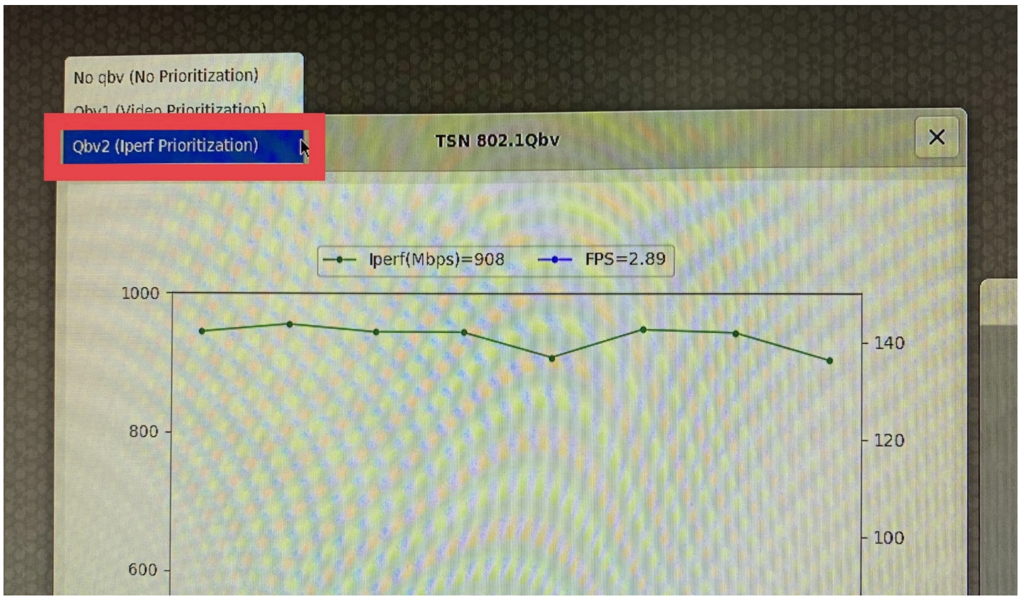
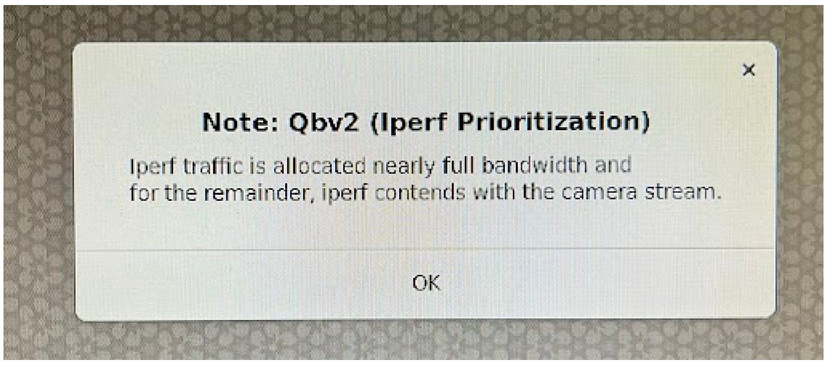

No qbv (No Prioritization)
Qbv1 (Video Prioritization)
Qbv2 (Iperf Prioritization)
TSN 802.1Qbv
Iperf(Mbps)=908
FPS=2.89

Figure 47. Qbv2 (Iperf Prioritization) configuration button

- After clicking the **Qbv2 (Iperf Prioritization)** button, a pop-up with a message appears, as shown in Figure 48.



Note: Qbv2 (Iperf Prioritization)

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

OK

Figure 48. 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.

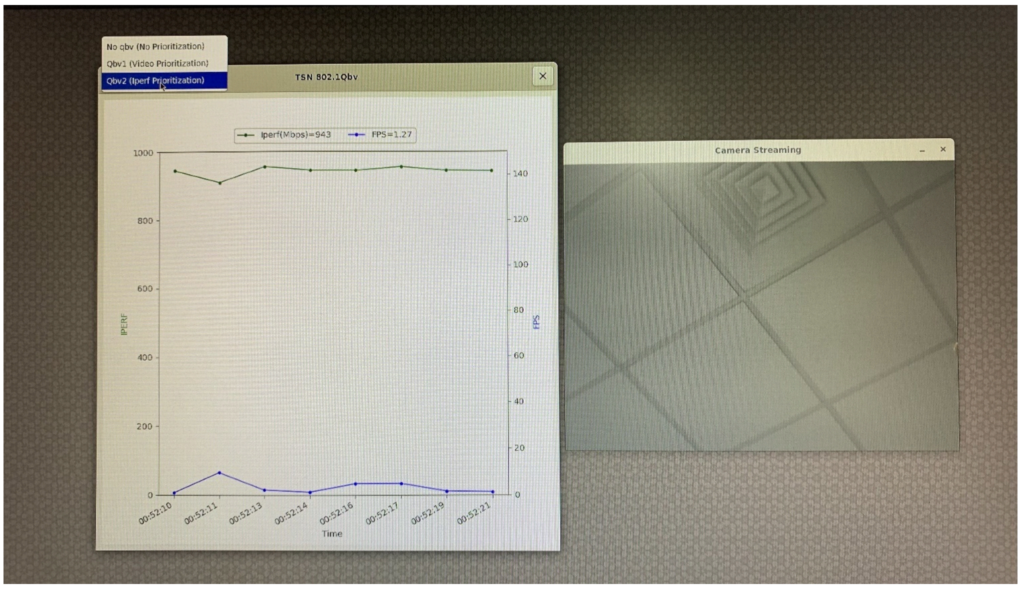

No qbv (No Prioritization)
Qbv1 (Video Prioritization)
Qbv2 (Iperf Prioritization)
TSN 802.1Qbv
Iperf(Mbps)=943
FPS=1.27
IPERF
FPS
Time
Camera Streaming

Figure 49. Qbv2 graph window and camera window

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

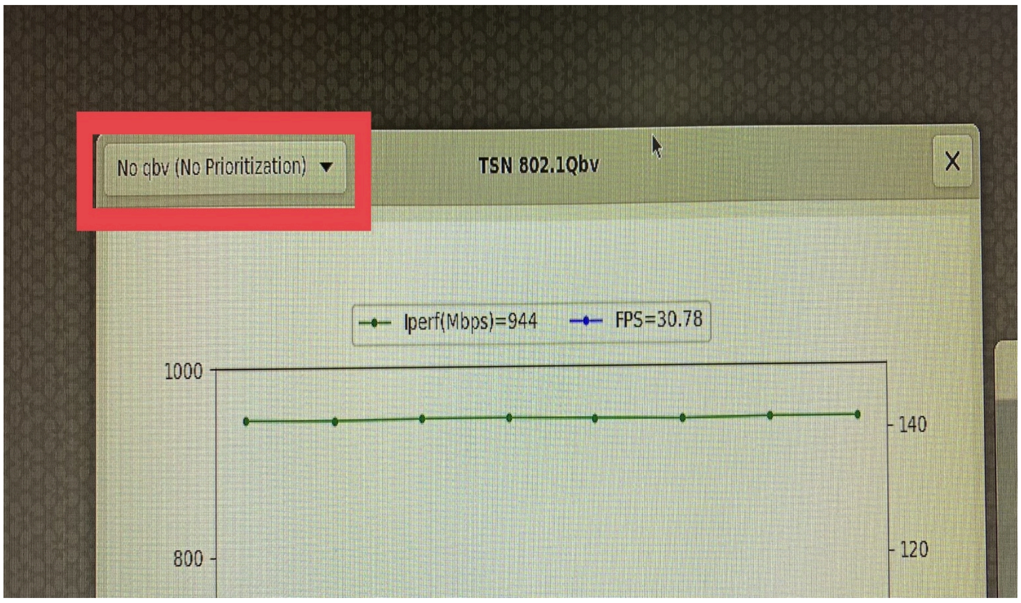
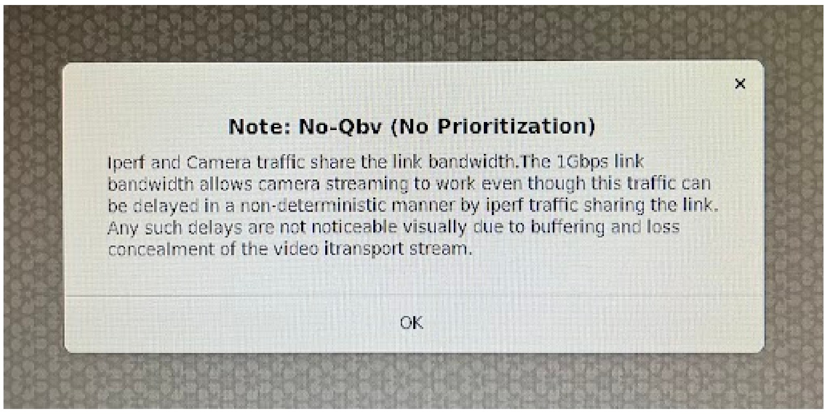


Figure 50. No qbv (No Prioritization) configuration button

- After clicking the **No qbv (No Prioritization)** button, as shown in Figure 50, a pop-up with a message appears, as shown in Figure 51.



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.

Figure 51. 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 and the camera must stream fine with FPS values of 25 FPS to 30 FPS.
- The iperf and the camera stream with full line rates.

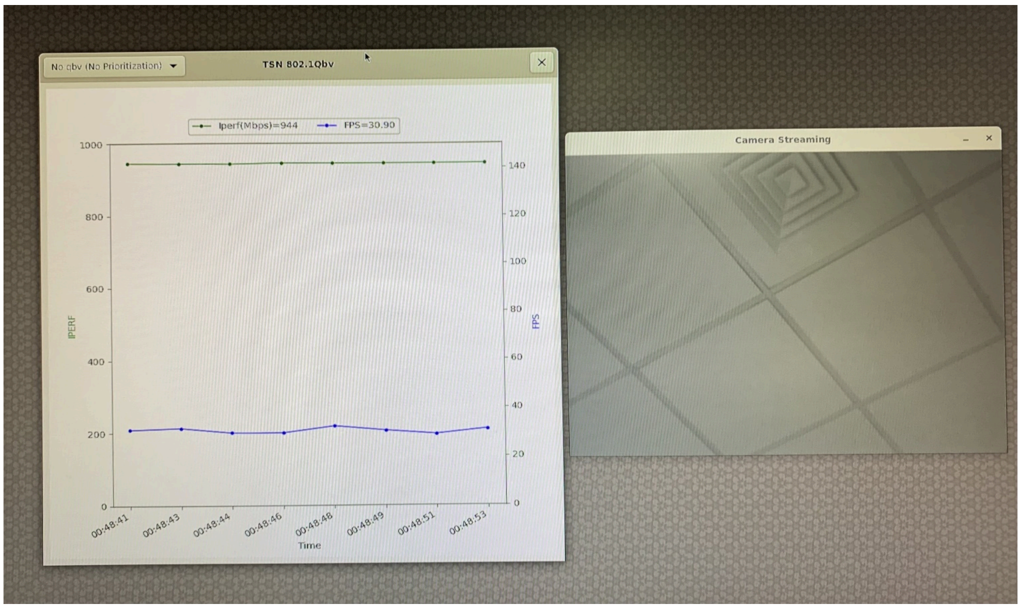


Figure 52. No-Qbv Graph window and Camera window

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

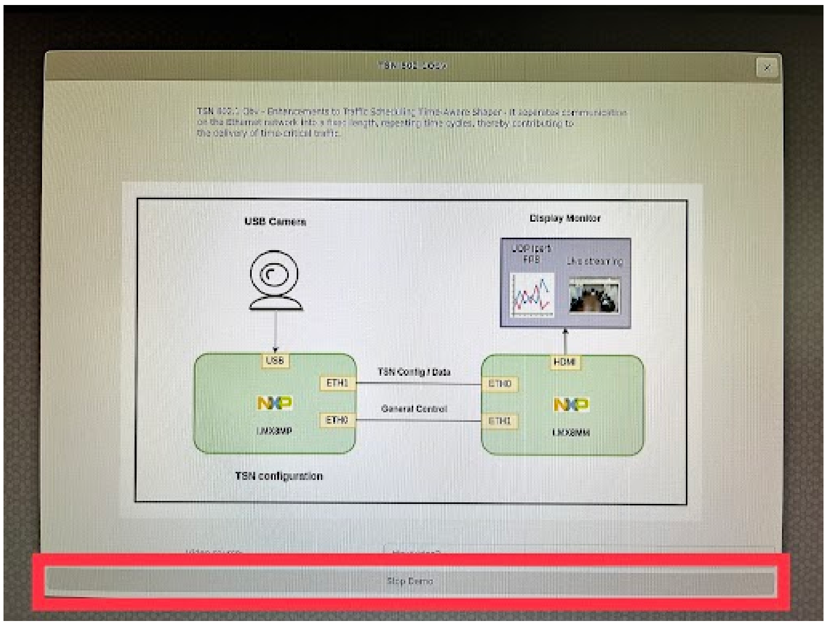

TSN 802.1Qbv - Enhancements to Traffic Scheduling: Time-Aware Shaper - It separates communication on the Ethernet network into a fixed length, repeating time cycles, thereby contributing to the delivery of time-critical traffic.

Figure 53. 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 GPU

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

3.4.1 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.

3.4.1.1 Tiger G2D

Required materials: Mouse and display output

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

3.4.2 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.

3.4.2.1 Vivante launcher

Required materials: Mouse and display output

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

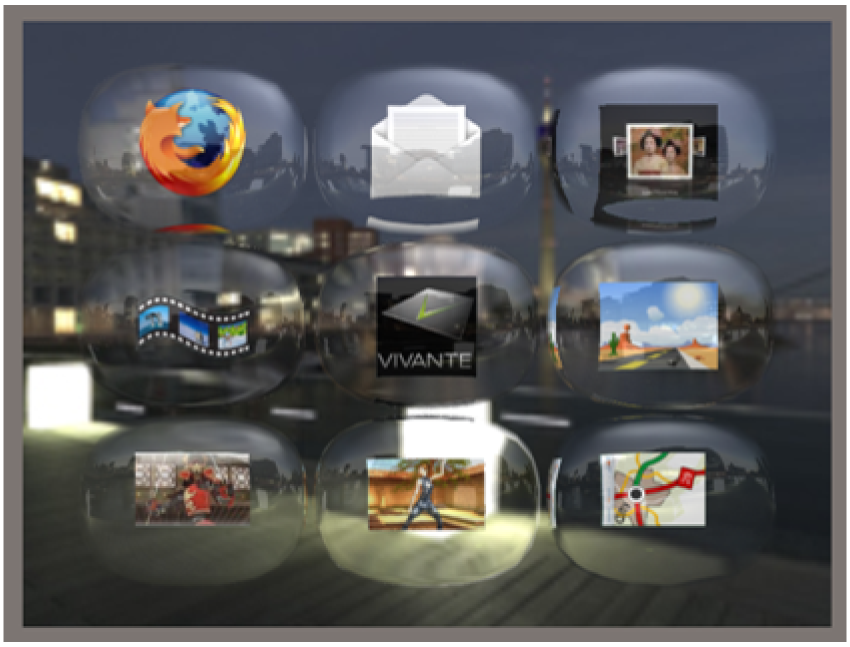


Figure 54. Example of a Vivante launcher

3.4.2.2 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.

3.4.2.3 Vivante tutorial

Required materials: Mouse and display output

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

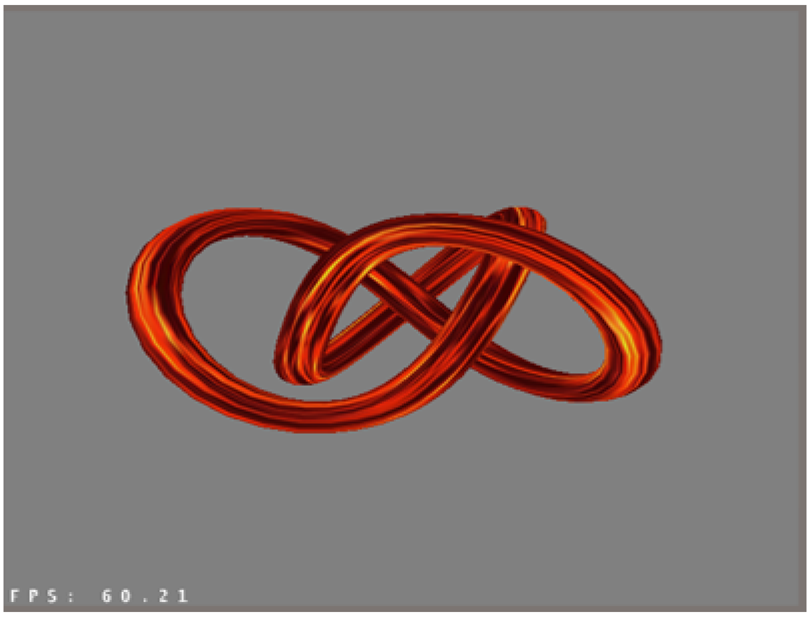


Figure 55. Vivante tutorial example

3.4.2.4 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.

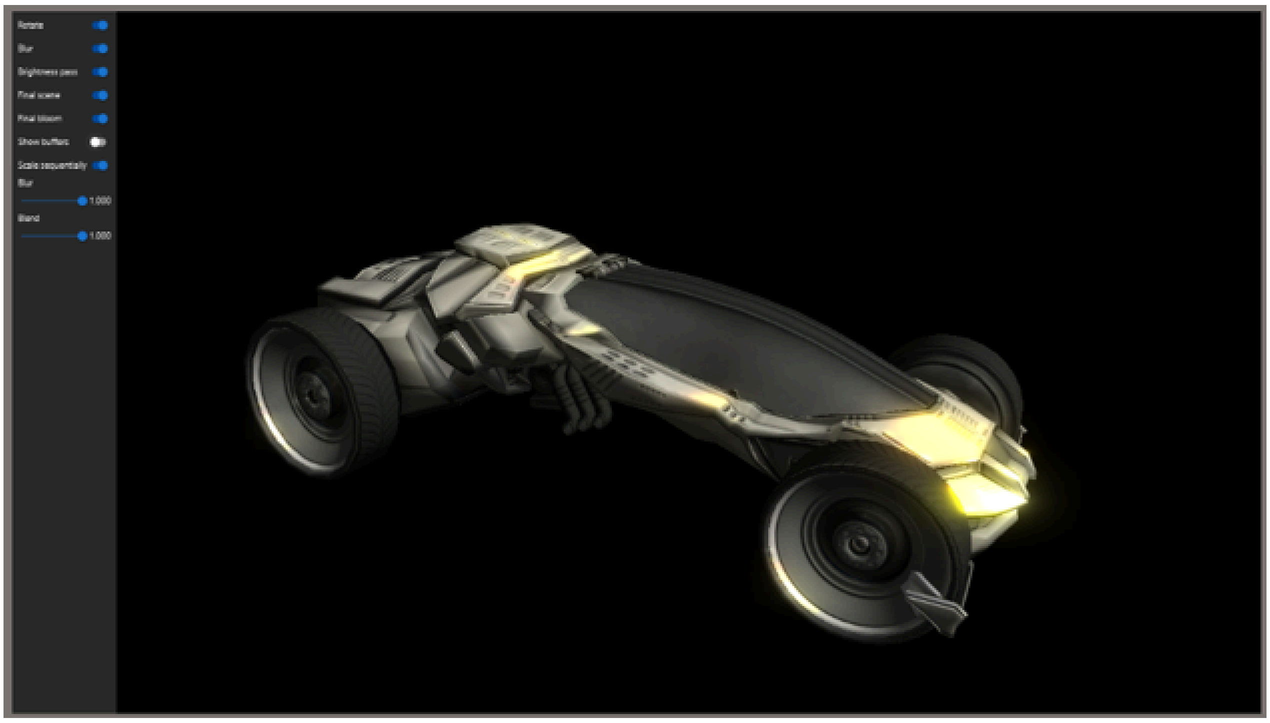


Figure 56. Bloom example

3.4.2.5 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's, but the downscaling produces visible artifacts with low sigma's.

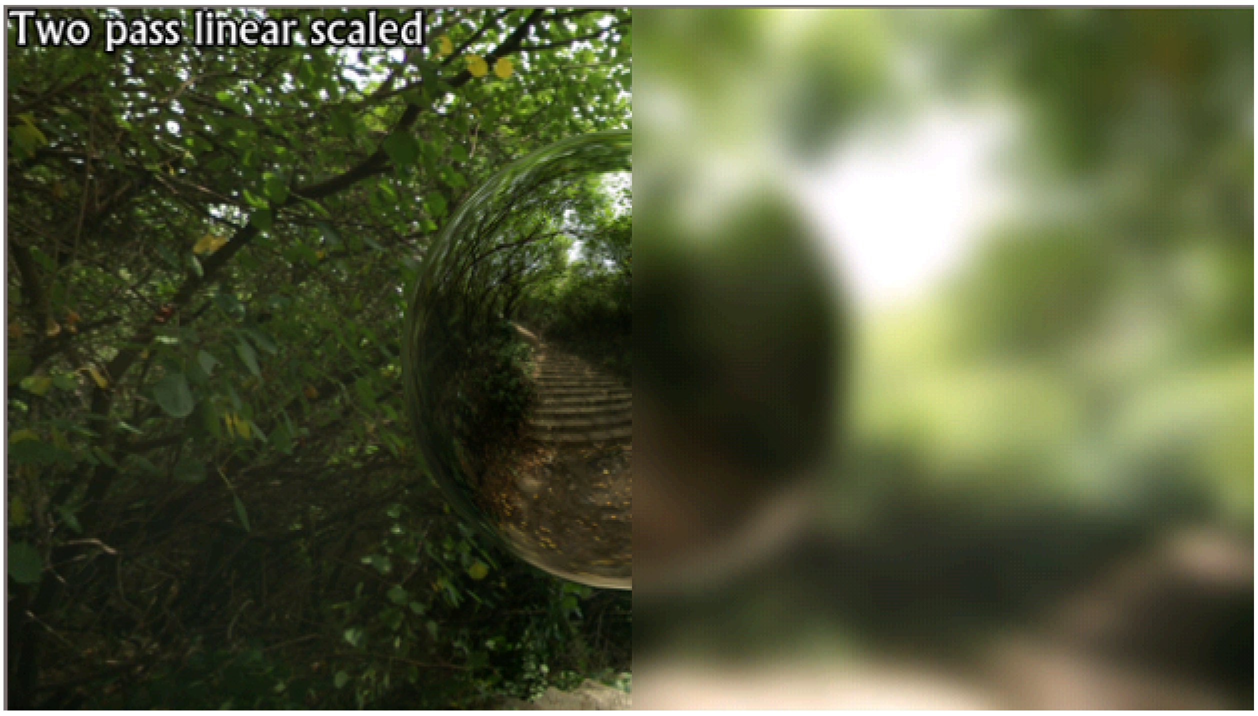


Figure 57. Blur example

3.4.2.6 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. This is not the most optimal way to do it as it uses eight passes. However, this 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.

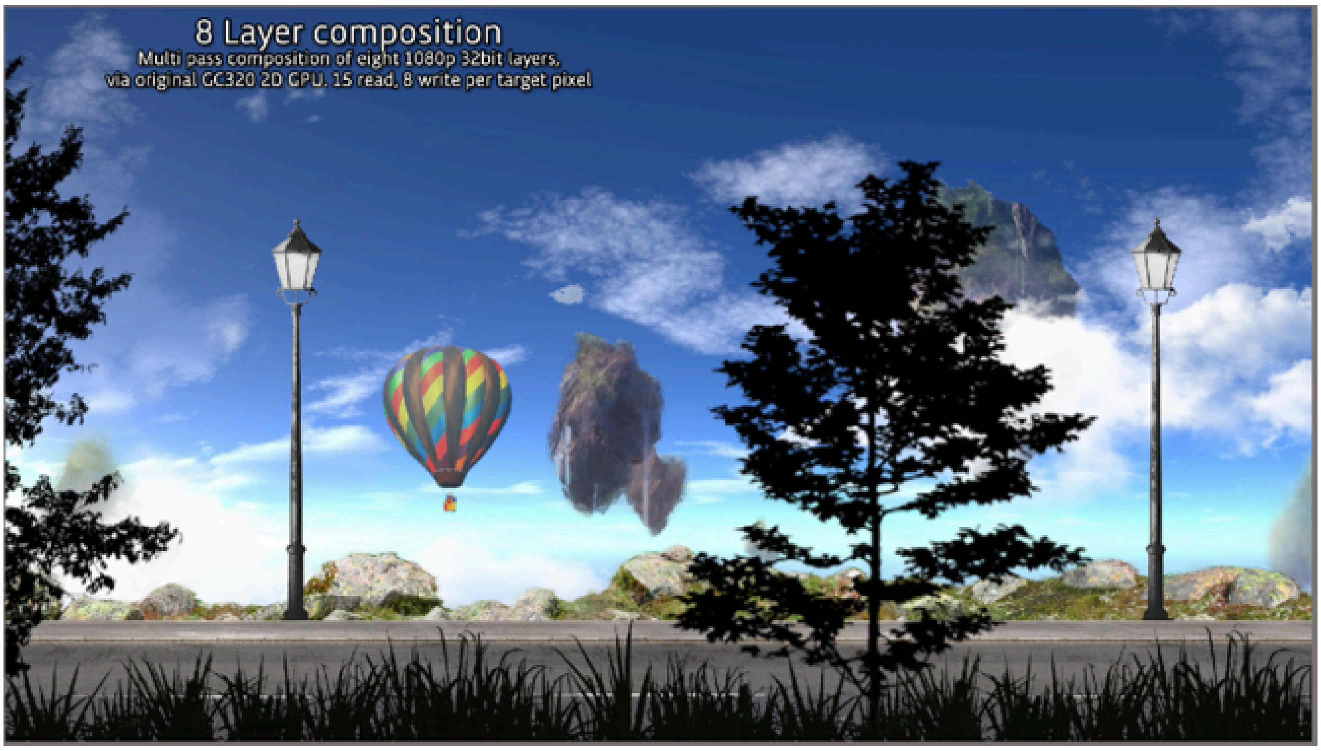
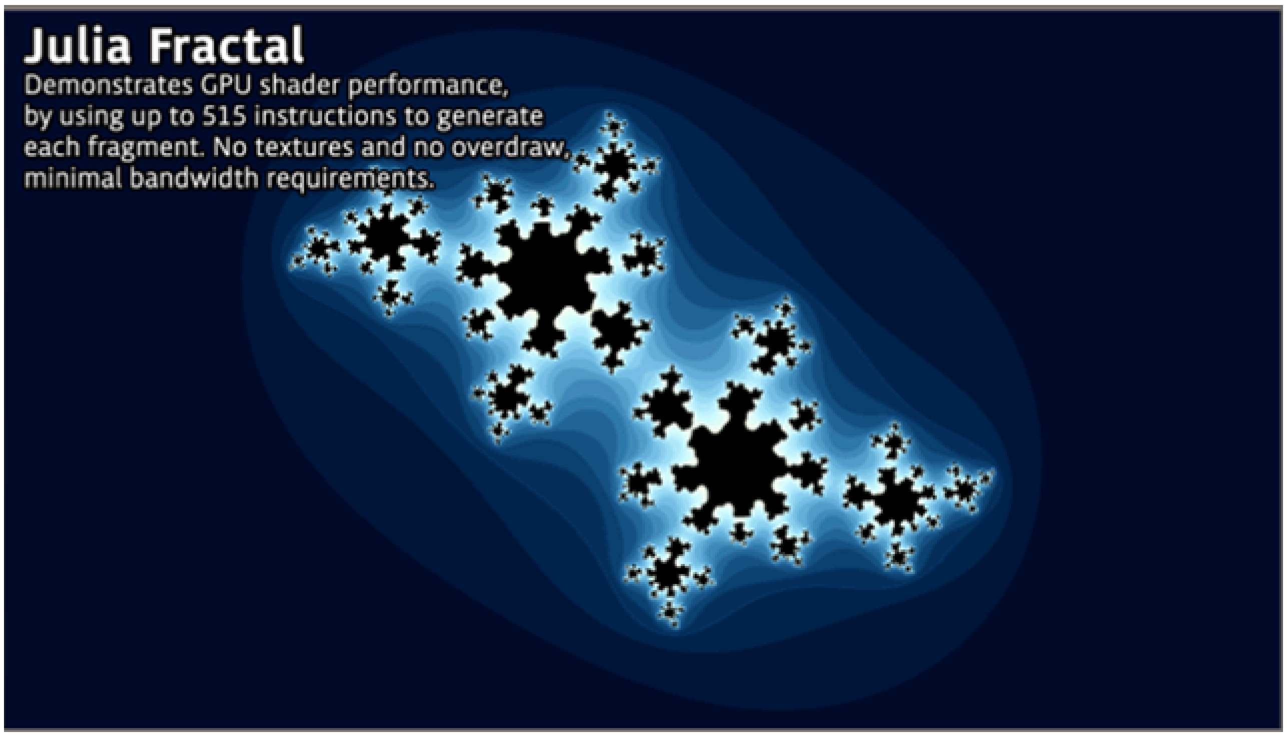

8 Layer composition
Multi pass composition of eight 1080p 32bit layers,
via original GC320 2D GPU. 15 read, 8 write per target pixel

Figure 58. Eight-layer blend example

3.4.2.7 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 59. Fractal shader example

3.4.2.8 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.

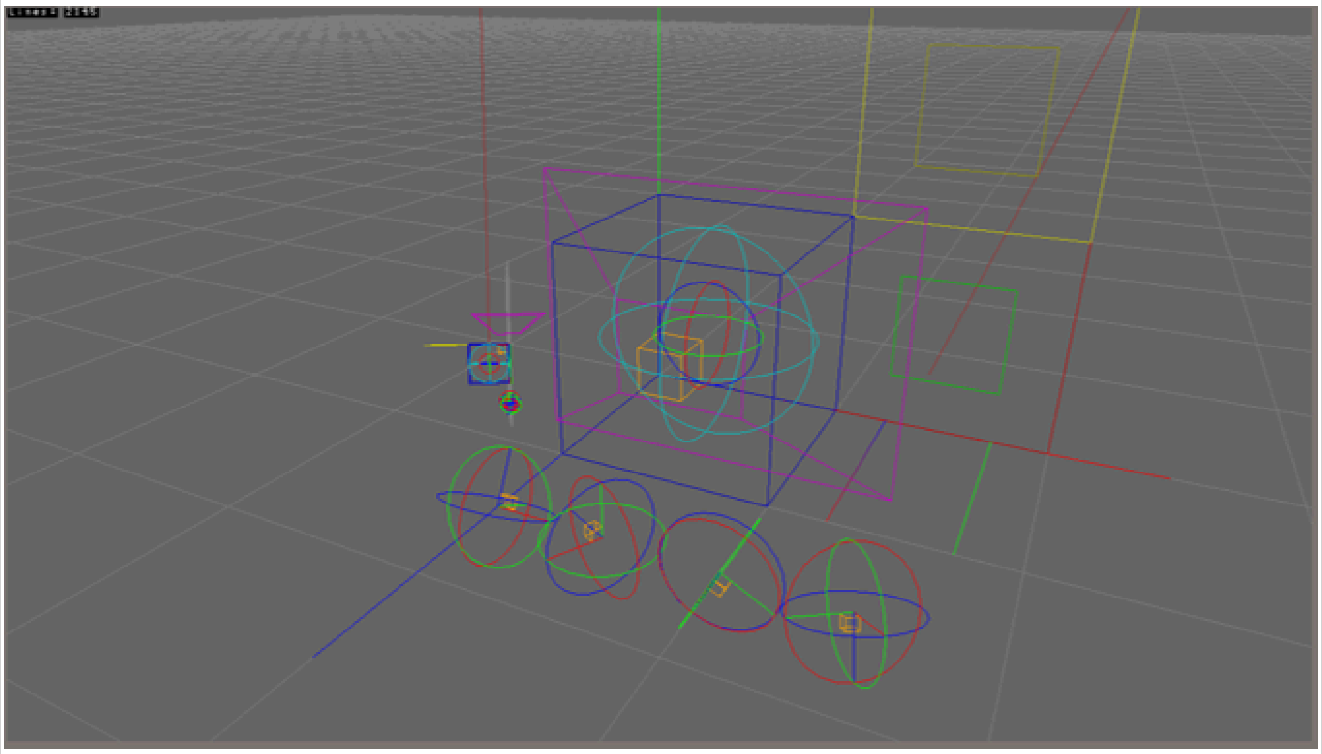


Figure 60. Line Builder 101 example

3.4.2.9 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.

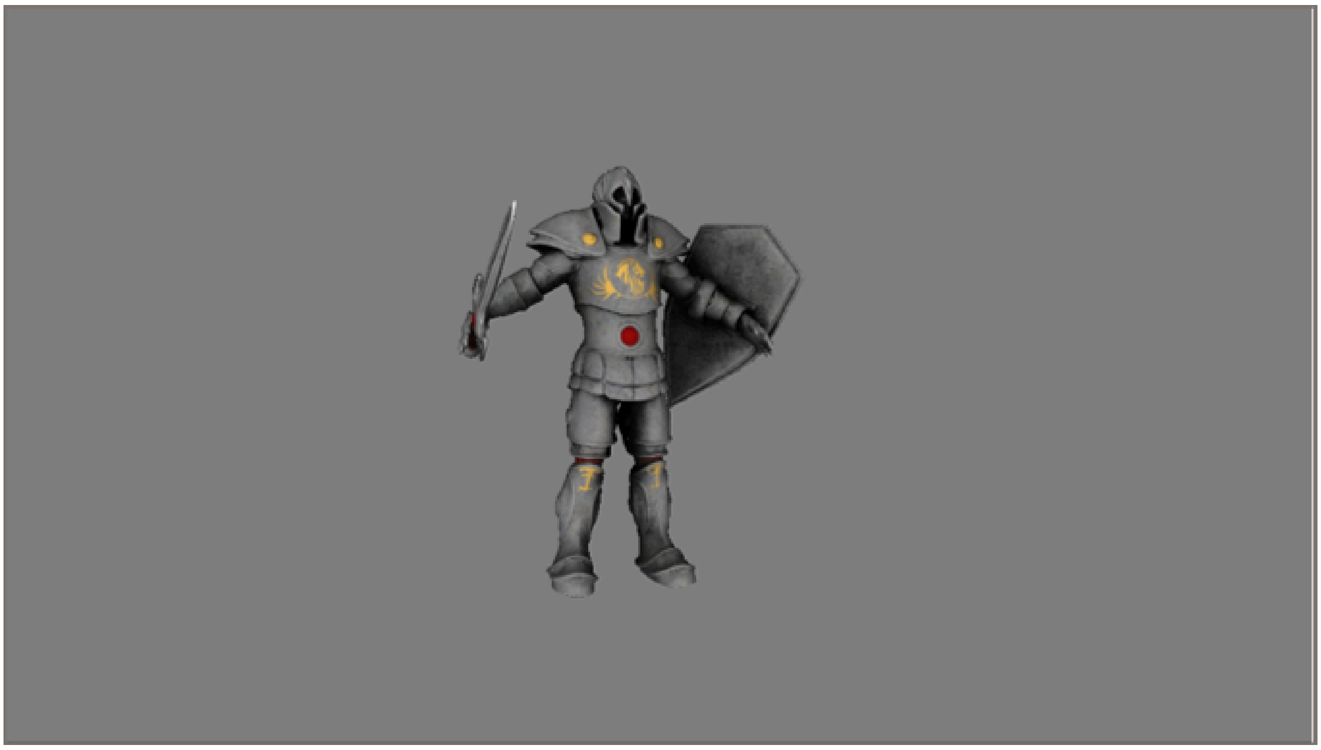


Figure 61. Model loader example

3.4.2.10 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 to use demoTime and deltaTime to do frame rate independent animation.

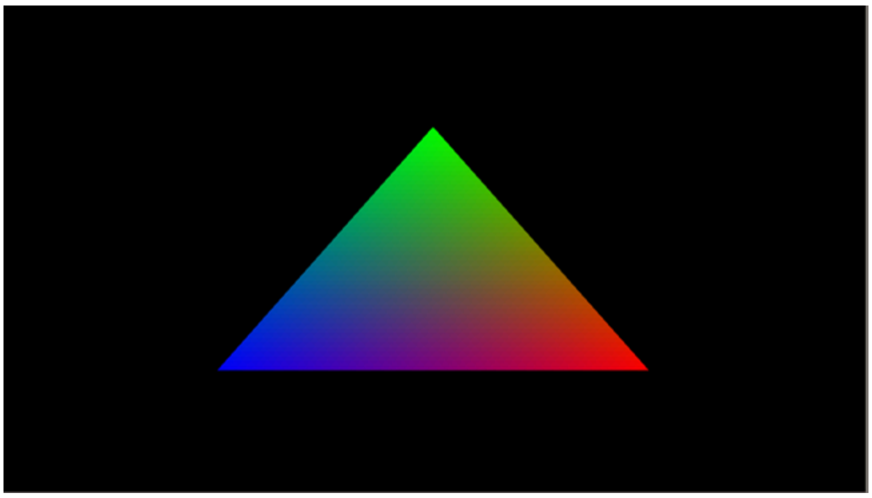


Figure 62. S03 transform example

3.4.2.11 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.

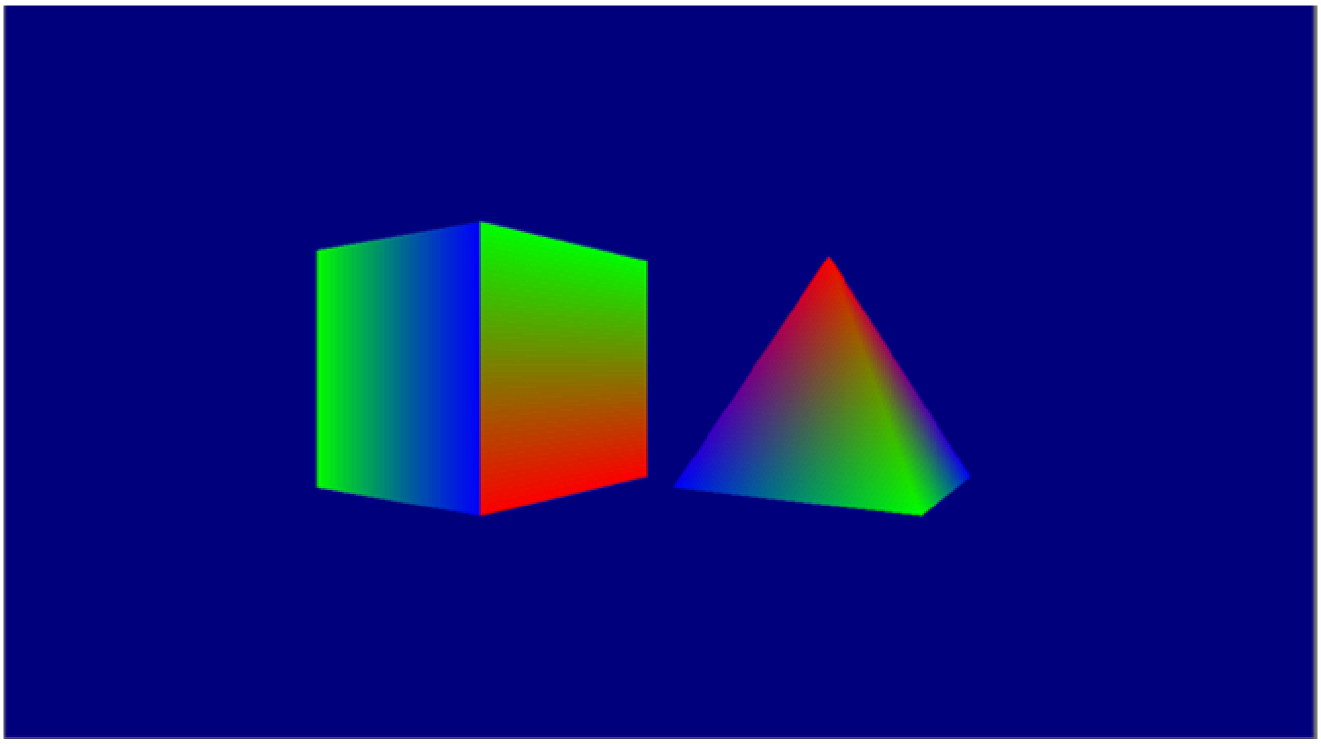


Figure 63. S04 projection example

3.4.2.12 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.

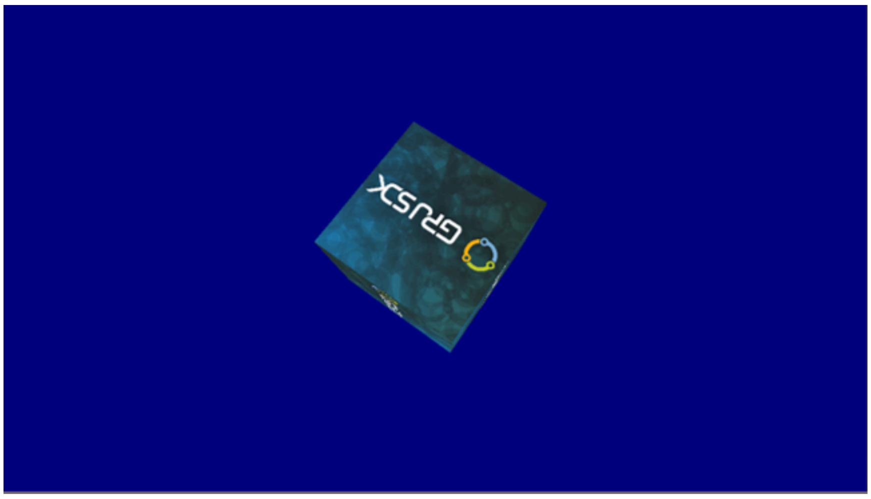


Figure 64. S06 texturing example

3.4.2.13 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 you 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.

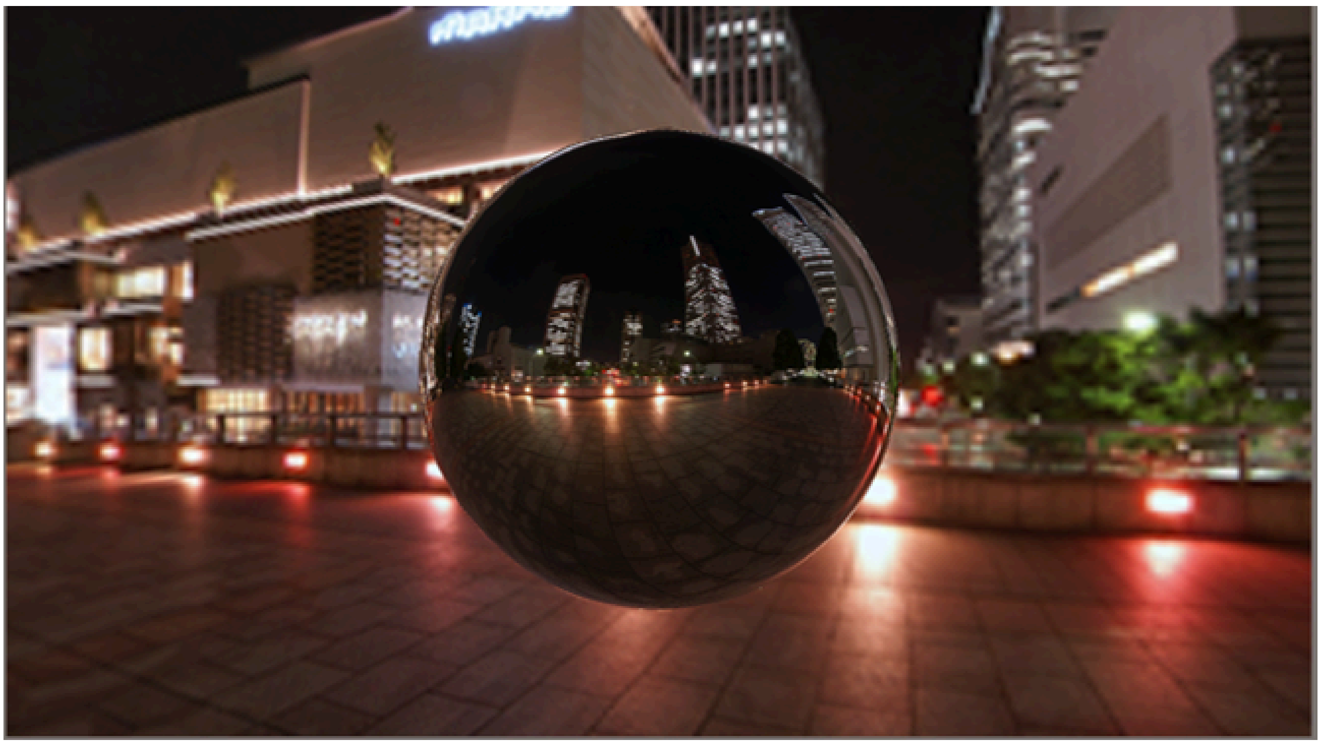


Figure 65. Mapping example

3.4.2.14 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.

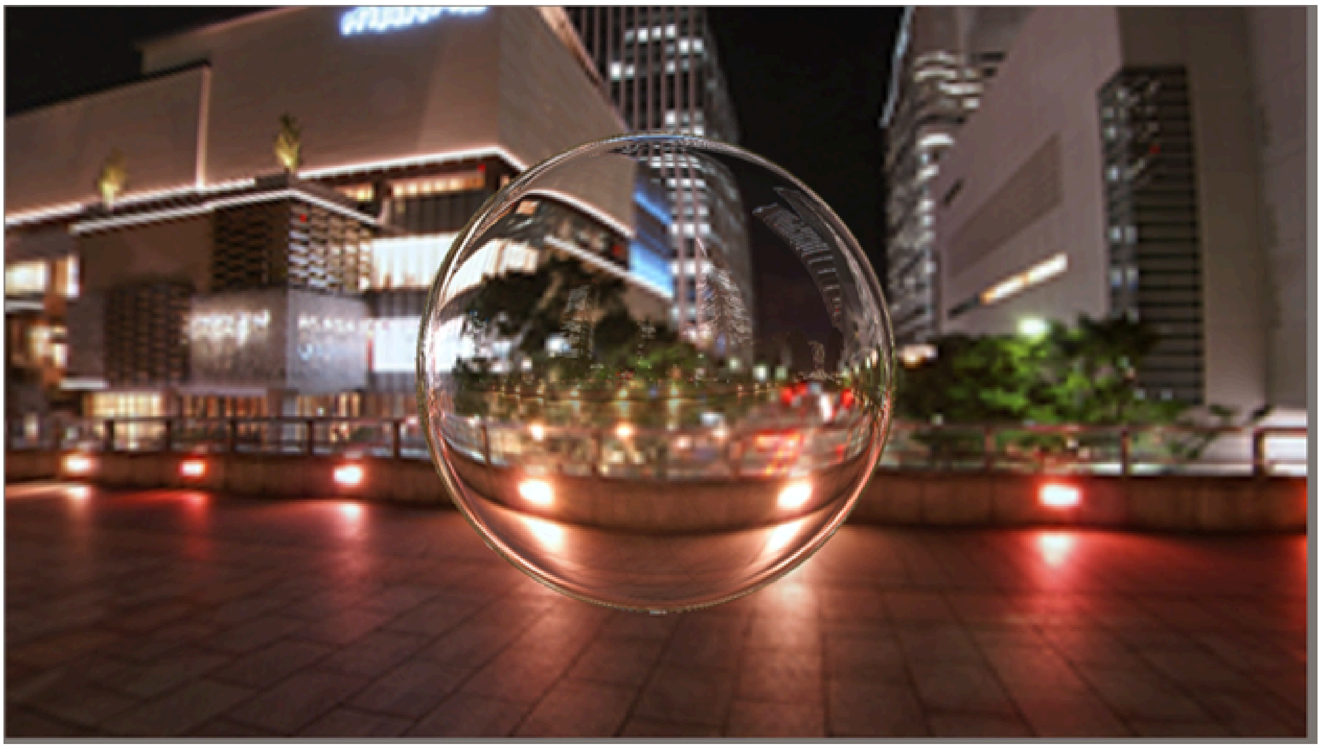


Figure 66. Mapping refraction example

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)

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 1 summarizes the revisions to this document.

Table 1. Revision history

Revision number	Release date	Description
6	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
5	30 October 2023	Updated for the 6.1.36_2.1.0 release
4	22 August 2023	Added i.MX multimedia player
3	28 June 2023	Added TSN 802.1 Qbv demo
2	07 December 2022	Updated for 5.15.71 release
1	16 September 2022	Updated for 5.15.52 release
0	24 June 2022	Initial release

Legal information

Contents

Date of release: 15 December 2023
Document identifier: GPNTUG