AN15042

Smart Doorbell Reference Design using i.MX RT700 Crossover MCU

Rev. 1.0 — 24 September 2026 Application note

Document information

Information	Content
Keywords	AN15042, i.MX RT700 crossover MCU, AI-powered smart doorbell with face recognition, UWB presence detection, and real-time audio video
Abstract	This application note presents the Smart Doorbell Reference Design using the NXP i.MX RT700 (MIMXRT798S) dual-core Cortex-M33 crossover MCU.

1 Introduction

Residential and commercial access-control products increasingly require on-device AI, real-time multimedia communication, and energy-efficient presence detection. All these features are required within the cost and power envelope of a single embedded MCU. This application note presents the Smart Doorbell Reference Design, which addresses these requirements using the NXP i.MX RT700 (MIMXRT798S) dual-core Cortex-M33 crossover MCU. The design achieves face recognition, two-way audio/video communication, and UWB/radar-based presence detection running concurrently, without an external application processor or discrete DSP.

This document guides developers through the complete workflow for the design. These steps include:

- hardware assembly
- SDK installation (MCUXpresso SDK v26.06.00).
- Importing the project,
- Building it using MCUXpresso for VS Code,
- Runtime validation, and
- Deployment on the MIMXRT700-EVK Rev B

It also includes the architecture, best practices, troubleshooting, and measured results for each major subsystem. This document also enables readers to build and run a fully functional smart doorbell demonstrator. Designers can use this document as a starting point for their own access-control product development.

2 Smart Doorbell Reference Design overview

Smart doorbells and video intercoms are rapidly evolving from simple camera-and-button devices into intelligent edge nodes capable of visitor identification, proactive presence sensing, and real-time voice/video interaction. Achieving this level of intelligence at the door requires combining AI inference, multimedia codecs, wireless connectivity, and low-power management. Traditionally, large, expensive application processors handle these features, which the cost-effective, production-grade i.MX RT700 MCU platform provides.

The Smart Doorbell Reference Design demonstrates how the NXP i.MX RT700 crossover MCU meets this challenge. It uses the dual Cortex-M33 cores, integrated eIQ AI/ML accelerator, USB host interface for UVC camera input, MIPI-DSI display output, and a flexible peripheral set of the MCU. These features make it possible to run face recognition, two-way audio-video streaming, and UWB presence detection on a single chip. This document helps developers gain hands-on experience with the NXP eIQ inference engine, MCUXpresso SDK middleware, LVGL/VGLite display stack, and the complete CMake + sysbuild dual-core build flow.

3 Solution overview

This section summarizes the high-level system architecture, key capabilities, and differentiating features of the Smart Doorbell Reference Design.

3.1 System architecture

This section describes the hardware and software architecture of the proposed solution.

3.1.1 Hardware architecture

The smart doorbell reference design hardware platform includes the following components:

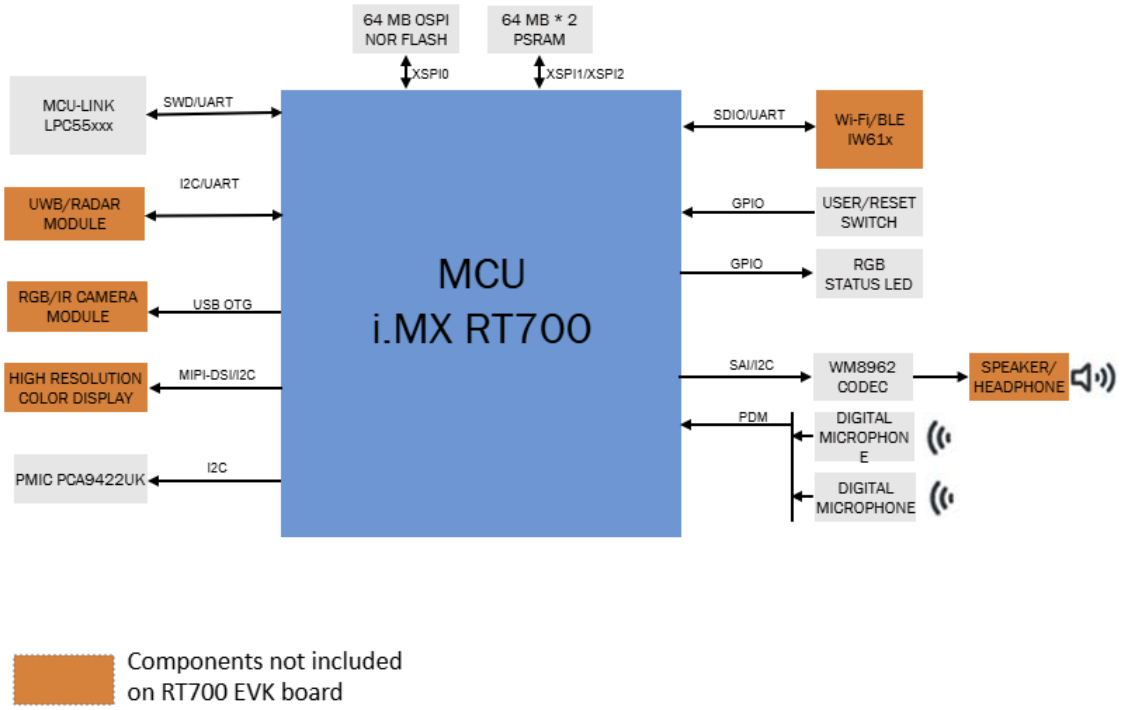

Figure 1. Smart doorbell hardware block diagram

- **MIMXRT700-EVK board (Rev B)**: It is the main processing platform that uses the i.MX RT798S dual Cortex-M33 SoC.
- **RGB + IR USB Camera module (J40)**: UVC-class MJPEG camera with separate RGB and IR sensors. Used for face detection, antispoofing, and recognition.
- **5.5-inch MIPI-DSI LCD (J52)**: Primary user display used for GUI rendering and interface (Part number NXP RK055HDMIPI4MA0).
- **IW61x Wi-Fi module (J44)**: Dual-band 802.11 a/b/g/n/ac Wi-Fi module via SDIO that provides wireless connectivity.
- **ETNA TS250 DEVKIT** (hereafter referred to as the **UWB/Radar module**): The module is responsible for presence detection and low-power wake-on-approach and is mounted on Arduino headers J5, J6, J23, and J24.
- **Digital microphones (DMIC)**: Onboard PDM digital microphones for audio capture at U390/U391 and U66/U67. Only 2 units are used in the current firmware.
- **WM8962 audio codec (U29)**: The onboard audio codec for audio output via SAI0.

- **Speaker (optional, J29 3.5 mm audio jack)**: Passive 8 Ω /1 W speaker for received audio stream output.
- **Debug micro-USB interface (J54)**: Programs, debugs, and provides serial console access. Baud rate: 115200 8N1.
- **5 V DC power supply (J45 barrel connector)**: Primary board power supply. Required: 5 V DC, center positive. To enable the power supply, short J2 pins 1 and 2.

3.1.2 Software architecture

The platform is built on the i.MX RT798S dual-core Cortex-M33 processor (up to 325 MHz). The workload is partitioned across two cores. Figure 2 shows the software architecture overview.

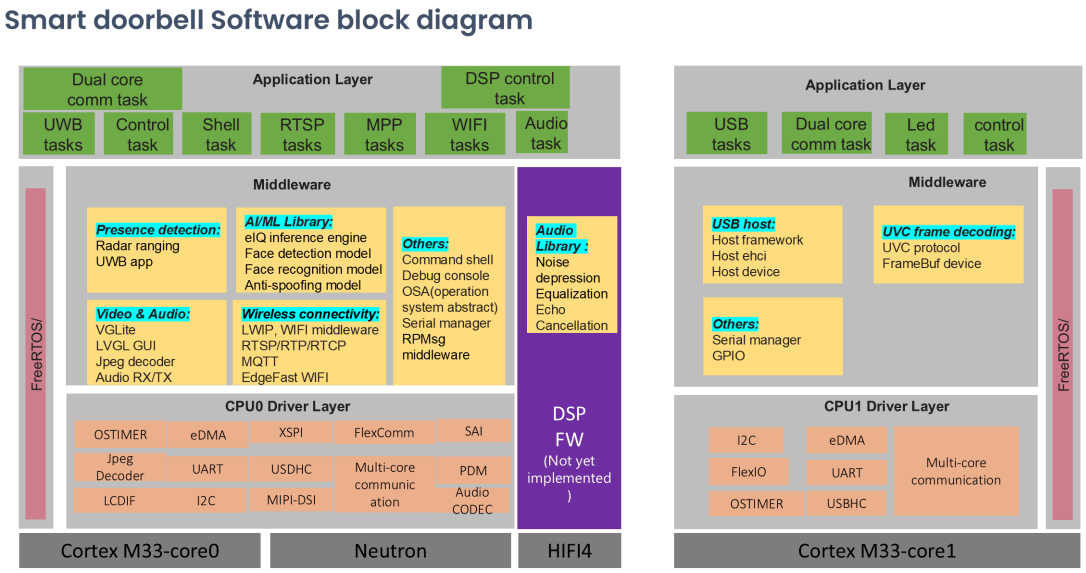

Figure 2. Software architecture overview

- **Core 0 (CM33 Core 0):** Runs the main application firmware under Free RTOS. Hosts the MPP (Media Processing Pipeline) framework that orchestrates the camera-to-inference pipeline. The responsibilities include the following:
  - Presence detection control tasks (via UWB/Radar module).
  - Audio/video capture and display tasks.
  - TCP/IP networking stack.
  - RTSP/RTP media streaming protocols.
  - MQTT client for cloud/broker connectivity.
  - LCD rendering via LVGL and VGLite.
  - The top-level main control task.
- **Core 1 (CM33 Core 1):** Dedicated to USB host operations. Runs the USB driver stack and handles UVC (USB Video Class) decoding for the RGB + IR USB Camera module (J40) connected via the USB host interface. Decoded frame buffers are transferred to Core 0 via RPMsg-Lite (remote processor messaging lite middleware) over shared memory.
- **NPU (Neural Processing Unit):** Executes all AI inference workloads offloaded from Core 0: face detection, face recognition, and antispoofing classification. Models are quantized and compiled for the i.MX RT798S NPU using the NXP eIQ toolkit.

- **DSP (HiFi4 DSP):** Reserved for advanced audio signal processing algorithms (noise suppression, echo cancellation, and beamforming). In the current firmware version, these algorithms are not yet implemented; audio is passed through directly via the WM8962 codec SAI0 interface.

3.2 Key features

The key features of the smart doorbell reference design are listed below:

1. **Face detection/recognition/registration**
   The smart doorbell integrates a dual-camera (RGB + IR) vision system combined with NXP eIQ AI/ML inference engine to enable reliable face detection, antispoofing, and recognition.
   - The RGB and IR cameras capture parallel frames, which are processed through JPEG decode → YUV conversion → RGB formatting before entering the AI pipeline.
   - The system performs face detection, then evaluates anti-spoofing using IR image cues, and finally extracts features for face recognition against an enrolled face database.
   - After recognition, results (identity, confidence scores, landmarks) are displayed on LCD via VGLite/LVGL rendering.
   - The user can also add new faces during registration sequences, enabling a fully personalized access-control setup.

   This feature enables the doorbell to identify visitors intelligently, differentiate between known and unknown individuals, and support cloud-assisted or offline recognition modes.
2. **Audio/video communication**
   The doorbell provides a real-time, Wi-Fi-based audio/video communication channel between the outdoor unit (i.MX RT700 MCU) and the indoor controller.
   - Video is streamed using an **RTSP server on the i.MX RT700**, with the controller acting as the RTSP client. Multiple sockets carry control, audio, and video packets simultaneously.
   - Audio supports **full duplex communication**, using RTP streams for both uplink and downlink. Raw PCM formats (16/32-bit, 8–16 kHz) are supported as specified in the source code.
   - The audio pipeline includes **noise suppression, echo cancellation, and equalization**, assisted by the HIFI4 DSP. (not implemented yet)
   - Session control is triggered through **MQTT commands** such as *Ring event*, *AV start*, *AV stop*, and GUI status indicators.
   - During an AV session, the LCD updates to reflect call state, streaming activity, and user options.

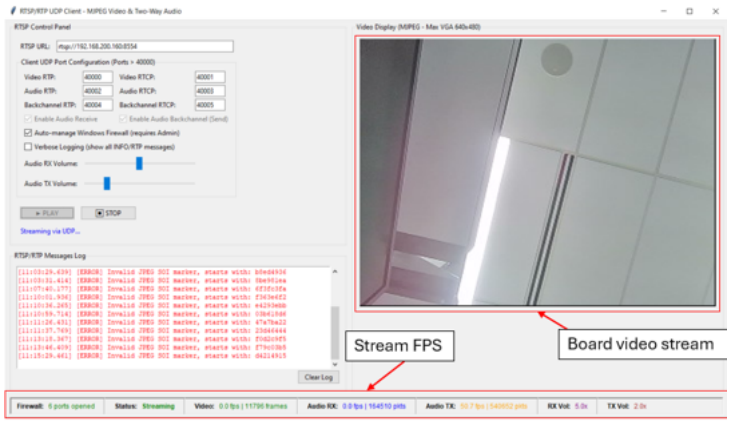

Figure 3. Controller GUI showing the incoming doorbell video/audio stream

As shown in Figure 3, the frame rate (FPS) counters are displayed at the bottom of the video panel.

This feature enables real-time interaction, allowing the homeowner to see and speak with visitors directly through a paired smart controller.

3. **Presence detection (low-power wake feature)**
   The system incorporates a **UWB/Radar module** to reduce power consumption while maintaining responsiveness.
   - When no motion or presence is detected, the system enters a **low-power sleep state** where the display and AI processing are disabled.
   - If a person approaches within approximately 1 meter (configurable via a parameter), the UWB/radar module triggers a wake-up event. This step activates the face recognition pipeline and turns on the LCD display.
   - State transitions (**Sleep** → **Face Recognition** → **Standby**) are controlled through the core state machine, which handles UWB events, timeouts, and recognition results.

   This feature makes the doorbell responsive without requiring constant high-power operation, enabling a more battery-friendly or thermally conservative design.

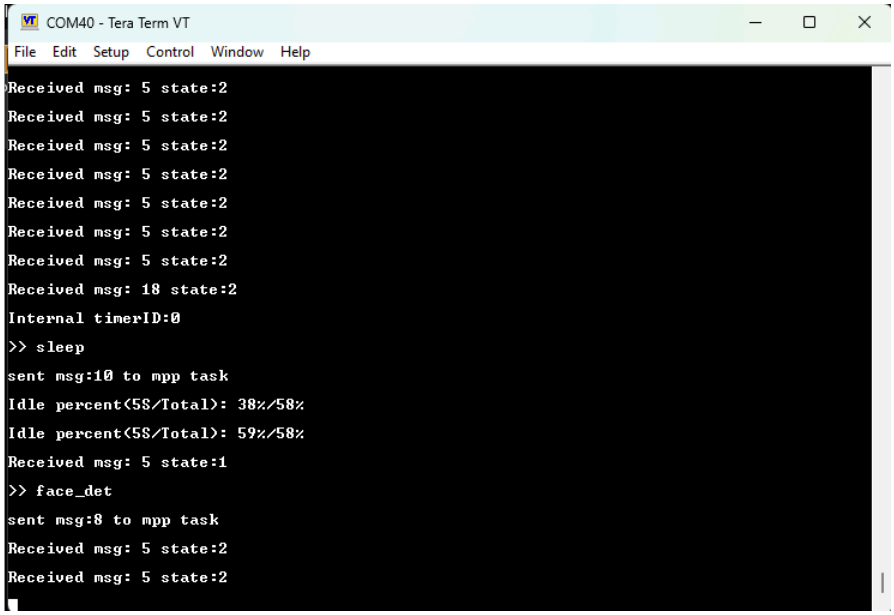

Figure 4. Core 0 console output showing UWB/radar module status

The module enters the low-power Sleep mode when no presence is detected. It wakes automatically when a person approaches within approximately 1 meter.

3.3 Interaction diagram

Figure 5 illustrates the complete interaction sequence between the smart doorbell unit and the indoor controller. The flow covers four main phases:

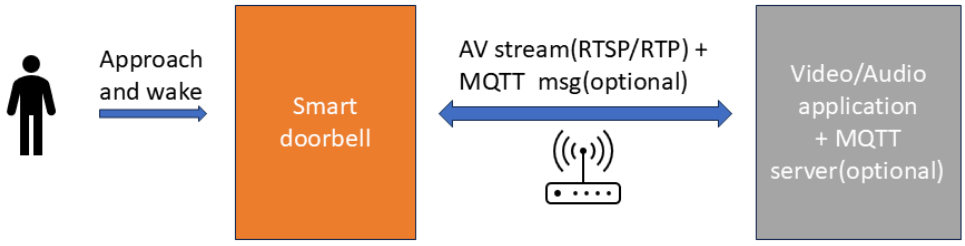

Figure 5. Interaction diagram

- **Presence Detection:** The UWB/Radar module on the doorbell continuously monitors the approach zone. When a person enters the detection range, the doorbell activates the camera pipeline and begins streaming video to the indoor controller via RTSP over the Wi-Fi network.
- **Face Recognition:** The doorbell's Core 0 runs the NXP eIQ MPP pipeline to capture frames from the RGB + IR USB camera module (J40). The NPU executes face detection, anti-spoofing, and face recognition inference. The result (known/unknown visitor) is published to the MQTT broker and displayed on the 5.5-inch MIPI-DSI LCD.
- **Audio/Video communication:** The indoor controller always receives the one-way RTSP video stream from the doorbell. When the user on the indoor controller initiates an audio/video session, two-way audio is established using RTP over UDP. Audio capture uses the onboard PDM DMICs and playback uses the WM8962 codec via SAI0.
- **Control messaging:** The indoor controller communicates control events (door unlock, session initiation, alerts) to the doorbell via MQTT. The doorbell subscribes to the relevant MQTT topics and reacts accordingly through the main control task running on Core 0.

4 Design and implementation details

This section describes the hardware connections required for the reference design. It includes the applicable hardware rework or verification steps, the system operation workflow, and UWB/Radar module firmware architecture.

4.1 Hardware connections

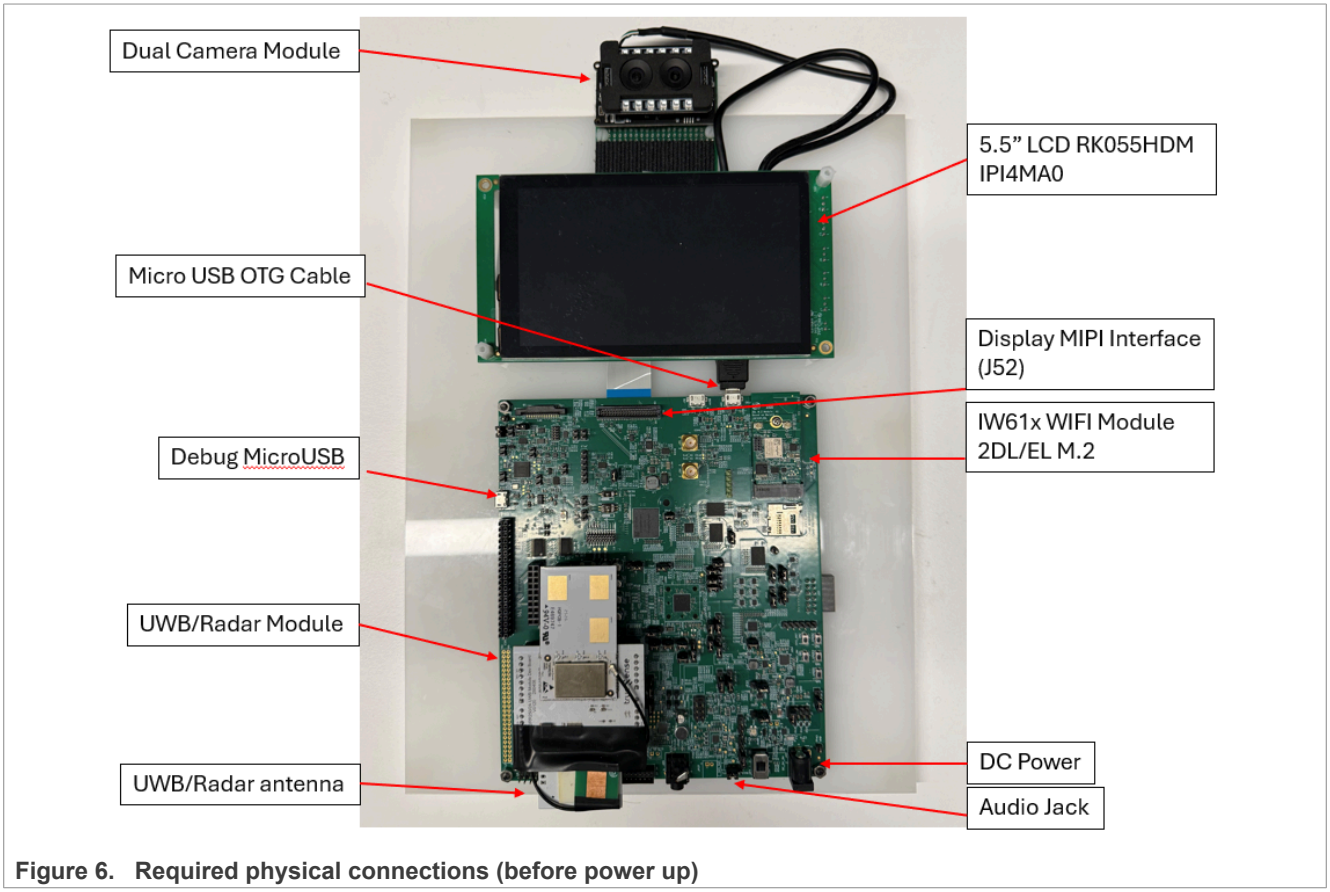

Figure 6. Required physical connections (before power up)

Table 1 lists all required hardware connections with their connector locations, descriptions, and sourcing information. Connect all peripherals before applying power.

Table 1. Required physical connections (before power up)

Component	Connector/Location	Description	Source/Link
MIMXRT700 EVK board (Rev B)	Base board	Main processing platform. i.MX RT798S dual Cortex-M33 @ up to 325 MHz Required: Rev B or later.	MIMXRT700 EVK
RGB + IR USB Camera module	J40	UVC-class MJPEG camera with separate RGB and IR sensors. HM2131 sensor recommended. VGA (640x480) resolution minimum.	Standard UVC dual-sensor module (RGB + IR) , USB OTG is used.
5.5-inch MIPI-DSI LCD	J52	Primary user display used for GUI rendering and information presentation.	NXP RK055HDMIPI4MA0

Table 1. Required physical connections (before power up)*...continued*

Component	Connector/Location	Description	Source/Link
IW61x Wi-Fi module	J44	Dual-band 802.11 a/b/g/n/ac Wi-Fi module via SDIO/M.2 interface.	EAR00422
UWB/Radar module	Arduino headers J5, J6, J23, J24	Presence detection and low-power wake-on-approach.	ETNA TS250 DEVKIT
Digital microphones (DMIC)	U390/U391 U66/U67	Onboard PDM digital microphones for audio capture.	Only 2 units are used.
WM8962 Audio codec	U29	Audio codec for audio output via SAI0. Supports 8 kHz / 16 kHz PCM, 16-bit. Controlled via I2C from Core 0.	Built into the MIMXRT700-EVK
Speaker (Optional)	J29 (3.5 mm Audio jack)	Passive speaker for received audio stream output.	Any standard 8-Ω/1 W speaker (optional accessory)
Debug micro-USB interface	J54	Programs, debugs, and provides serial console access. Two virtual COM ports are exposed (one for CM33 Core 0 and the other for Core 1).	Standard micro-USB interface. Baud rate: 115200 8N1.
5 V DC power supply	J45 (barrel connector)	Primary board power supply. Required: 5 V DC, center positive, minimum 1 Ampere.	Short J2 pins 1 and 2 to enable the DC power supply.

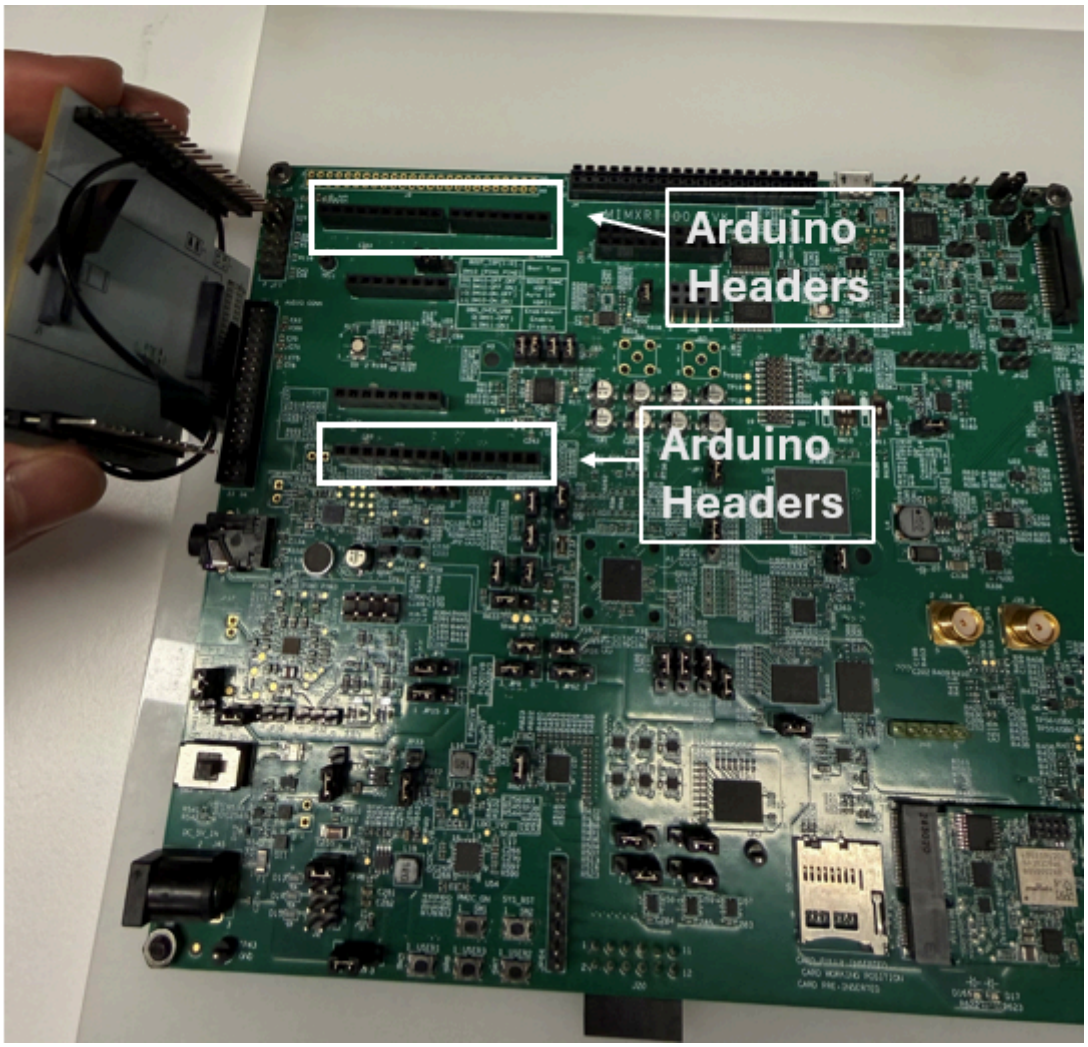

Figure 7. Arduino headers

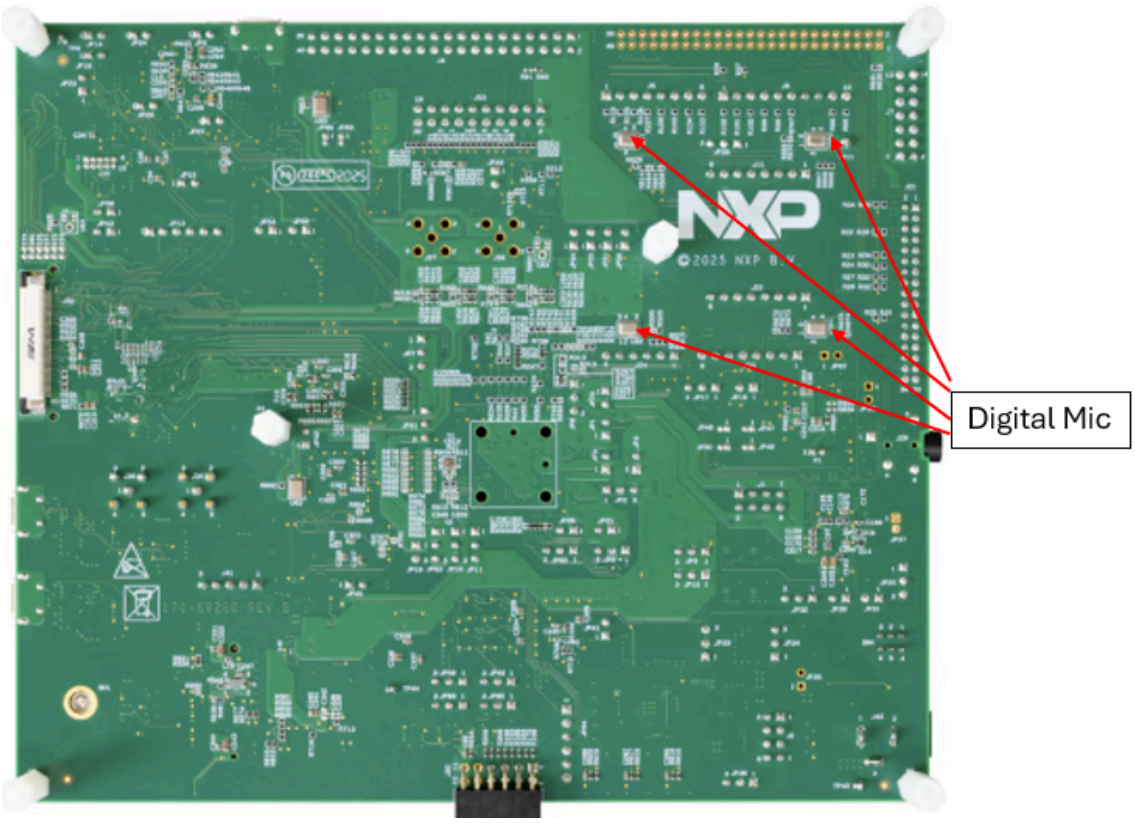

Figure 8. DMIC on the back side of the MIMXRT700-EVK board

4.2 Hardware rework notes

The following hardware modifications are required:

- UWB/Radar module power: Solder a wire connection between J23 pin 5 and JP17 pin 1.
- UWB antenna orientation: The antenna patch on the TS250 DEVKIT must face the same direction as the camera and LCD (front-facing). Reversed orientation causes incorrect ranging and missed presence events.
- Camera USB enumeration: If the RGB + IR USB camera module is not detected, check that J40 is configured as the USB host. (Use a proper OTG cable with the ID pin pulled low.)

Note: *All the items above apply to MIMXRT700-EVK Rev B.*

4.3 System operation workflow

1. **Event triggered**: The process begins when the doorbell button is pressed, or motion is detected at the door.
2. **Face recognition initiated**: the smart doorbell captures video of the visitor and runs face recognition to determine whether the person is known or unknown.
3. **Recognized visitor (Success path)**
4. **Unrecognized visitor (Failure path)**:
   - If the face is not recognized, a different notification is sent to the home controller indicating an unrecognized visitor.
   - A video or audio chat request is initiated to allow the home owner to interact with the visitor.
   - To register a face onto the board, press the button **SW5** or **User1**.
   - Position the face in front of the camera and press **SW5**.

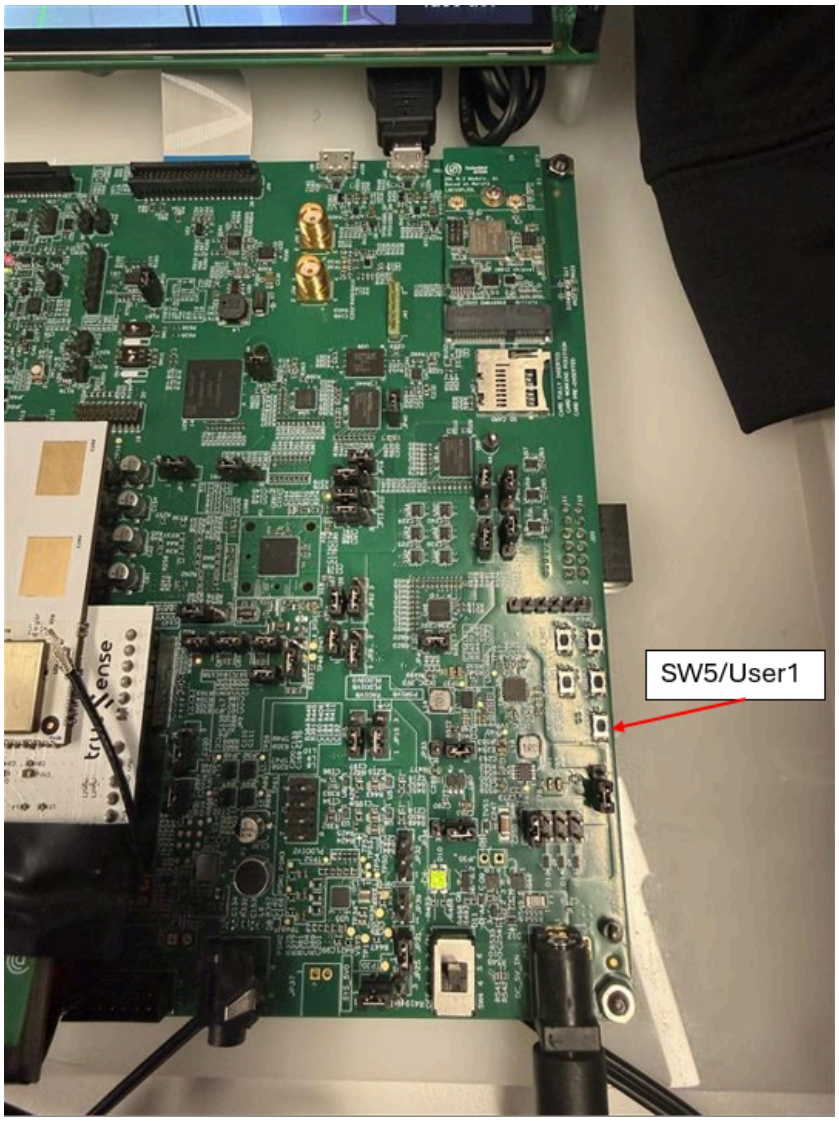

Figure 9. Press SW5 button

5. **Two-way video/audio communication**: The home owner and the visitor communicate in real time via a secure video and audio channel that the smart doorbell system provides.
6. **User decision and control**: During the conversation, the home owner decides whether to grant access or end the interaction based on the visitor and context.

4.4 UWB/Radar module architecture and firmware

The UWB/Radar module requires a specific firmware image to operate in the radar (presence-detection) mode. The firmware binary is compiled into the i.MX RT700 application as a C header array. The binary is automatically downloaded to the module over SPI on every boot. Therefore, a separate standalone firmware-update step is normally not required.

To update the firmware to a newer version, refer Section 5.6.

4.4.1 Firmware image location

The bundled firmware image is available at:

```
core0/source/uwb/uwb-sr250/firmware_images/SR2XX/Mainline_Firmware.h
```

This header is compiled into the i.MX RT700 binary. The UWB IoT SDK (`UWBDemo_Init()` in `uwb_app.c`) streams it to the module via LPSPI16 during system initialization.

5 Step-by-step procedure

This section provides the complete, step-by-step procedure for setting up the development environment, importing, and building the project. It also includes the steps for Wi-Fi configuration, AV communication, face registration, and updating the UWB/Radar module firmware.

5.1 Software environment setup

Install the following tools on a Windows 10/11 x64 development PC before cloning the SDK or the doorbell project.

5.1.1 Required tools

1. Install the latest version of Visual Studio Code from https://code.visualstudio.com/.
2. In VS Code extensions, search "**MCUXpresso for VS Code**" and install it. Use MCUXpresso for VS Code to complete the installation of required tools listed in Table 2. The below link provides detailed instructions: https://mcuxpresso.nxp.com/mcux-vscode/latest/MCUXpresso-Installer.html#installer-overview

Table 2. Tools required

Tool	Version/Notes	How to obtain
Arm GNU Toolchain	14.2.Rel1 arm-none-eabi or later	Installed automatically by MCUXpresso extension to `~/.mcuxpressotools/`
CMake	3.22 or later	Bundled with MCUXpresso extension, or https://cmake.org/download/
Ninja build	1.11 or later	Bundled with MCUXpresso extension, or https://github.com/ninja-build/ninja/releases
Python	3.10 or later	https://www.python.org/downloads/
west (meta-tool)	1.2 or later	`pip install west`
Git	2.35 or later	https://git-scm.com/

5.1.2 Install MCUXpresso extension, import SDK repo

Import a MCUXpresso SDK version that supports the i.MX RT700 MCU.

Important: *This design uses SDK version 26.06.00, which is recommended for use. Newer SDK versions may also be compatible. However, they have not been verified.*

The detailed process is listed below:

https://mcuxpresso.nxp.com/mcux-vscode/latest/Import-Repository.html#import-repository

5.2 Import Smart Doorbell Project

This section describes the steps to import the Smart Doorbell Project into VS Code and associate the SDK.

5.2.1 Import the project into VS Code

1. In the MCUXpresso for VS Code Quickstart Panel, select **Application Code Hub**.
2. In the Search field, enter **smart doorbell**. The Smart Doorbell application appears in the search results.
3. Specify a unique name and location for the new Project.
4. Click **Import Project**. The MCUXpresso for VS Code extension clones the repository and imports the project into the active workspace.
5. Detailed instructions to use *Application Code Hub* (ACH) are available at the URL: https://mcuxpresso.nxp.com/mcux-vscode/latest/Application-Code-Hub.html#application-code-hub.
6. Alternatively, use the following GitHub link to access the project source code: https://www.github.com/nxp-appcodehub/an-smart-doorbell-on-rt700.
   Clone the repository to a local folder, and then import the project by following the instructions:Project Management — MCUXpresso for VS Code documentation.

5.2.2 Project directory structure

The project directory structure is shown below:

```
smart_doorbell/
├── core0/          CM33 Core 0 application:
│                   • eIQ
│                   • MPP
│                   • Audio processing
│                   • Video decoding and display
│                   • Wi-Fi
│                   • UWB
│                   • RTSP
│                   • MQTT
│
├── core1/          CM33 Core 1 application:
│                   • UVC protocol handling
│
├── multicore/      Shared RPMsg-Lite definitions and multicore communication code
│
├── .vscode/        Visual Studio Code and MCUXpresso configuration files
│
├── license/        License files
│
└── flash_debug/    CMake-generated build output directory
```

5.2.3 Associate the SDK

1. Right-click the project in the **MCUXpresso panel -> Config-> Associate SDK**.
2. Select the MCUXpresso SDK imported.
3. The extension writes the correct `SdkRootDirPath` into the `core0/mcux_include.json` file.

5.2.4 Associate the Toolchain

1. Right-click the project in the **MCUXpresso panel** -> **Config-> Associate Toolchain**.
2. Select **ARM GNU Toolchain 14.2.Rel1** (`arm-none-eabi`).

3. The toolchain path is saved in `core0/mcux_include.json` under `ARMGCC_DIR`.

5.3 Build/program instructions

The project uses CMake and Ninja with the MCUXpresso SDK sysbuild extension. Sysbuild builds both Core 0 and Core 1 in a single configure-and-build step. The build process uses the SDK sysbuild *CMakeLists.txt* file as the entry point, while *core0/CMakePresets.json* provides the preset build configuration.

5.3.1 Build/program via VS Code (recommended)

Follow the steps below to build or program via VS Code, which is the recommended method.

- Right-click the project in the **MCUXpresso panel -> Config -> Set Sysbuild**.
- Confirm that `sysbuild` is enabled.
- Right-click the project in the **MCUXpresso panel -> Pristine Build/Rebuild Project**.
  A successful build produces the output below:

```
flash_debug/core0/mimxrt700evk_doorbell.elf
flash_debug/core1/mimxrt700evk_doorbell_core1.elf
flash_debug/core1_image.bin   // embedded in Core 0 binary
```

- Connect the board and developing PC by a USB cable through J54 (micro-USB). Then, ensure that the board is powered (DC power connector J45 is connected and SW4 is “ON”).
- To program the firmware into flash, right-click the project in the **MCUXpresso panel -> Flash the selected target**.
- Select `core 0` and `doorbell_cm33_core0.elf`.

5.3.2 Build via command line

To make a clean build, delete the `flash_debug` folder.

Step 1 - Configure (from smart_doorbell directory):

```
cmake -S <your_SDK_root>\mcuxsdk\cmake\extension\sysbuild --preset flash_debug -
Dboard=mimxrt700evk -Dcore_id=cm33_core0
```

Step 2 - Build:

```
cmake --build flash_debug
```

Note: ***Do NOT*** *run cmake --preset from inside core0 or the workspace root. That command builds only Core 0 and fails with a missing* `core1_image.bin` *link error. Always use -S to point to the sysbuild directory as shown in the commands above.*

Step 3 - Program:

```
LinkServer flash MIMXRT798S:MIMXRT700-EVK load <your_prj_root>\vss_prj
\flash_debug\core0\doorbell_cm33_core0.elf
```

5.4 Wi-Fi configuration and AV communication

This section describes the steps for Wi-Fi configuration and AV communication.

5.4.1 Wi-Fi configuration

There is no Wi-Fi configuration information by default. Users must configure the Wi-Fi by using the console command line before communication with the AV application.

1. Run a serial connection tool like PuTTY. Setting: 115200, 8N1
2. There are 2 debug UARTs. Connect the one that is meant for Core0 communication.
3. Enter at the shell command line:

```
SHELL>>
```

4. Input the command "`help`", it shows you the commands list.
5. Input the below command to set the SSID and PWD of the Wi-Fi AP.

```
wlan_connect_with_password <ssid> <password>
```

6. Once the network connection is done, the board IP address is displayed after logging in, as shown below:

```
Network joined
IP address: 192.168.200.246
```

5.4.2 AV communication

The indoor controller simulator is a Python-based software tool that communicates with the doorbell over Wi-Fi. Use the following steps to set it up:

1. **Open a Command prompt with administrator privileges and follow the steps below.**
   - Click the **Start** menu.
   - Search for the **Command Prompt**.
   - Right-click it and select **Run as administrator**.

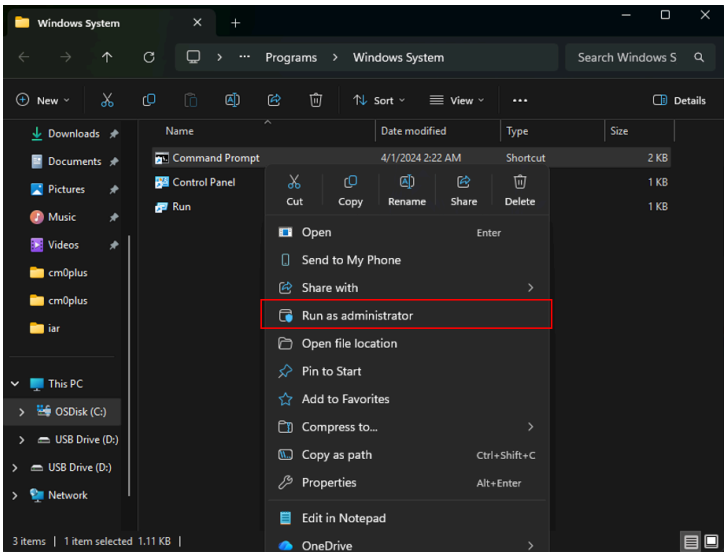

Figure 10. Run as administrator

2. **Install OpenCV and Pillow dependencies:**
   - In the Command prompt, type the following command and press **Enter**:

```
pip install opencv-python
```

   - The preceding command installs OpenCV and includes cv2, so no separate installation for cv2 is required.
   - In the same Command prompt window, type the following command and press **Enter**:

```
pip install pillow
```

   - Wait for the installation process to be completed before continuing.
3. **Verify the Wi-Fi connection:**
   - Ensure that your computer is connected to a Wi-Fi Access point.
   - Remember the IP address of your PC.
4. **Confirm administrative access**
   - Ensure that all commands are run in a Command Prompt window with **administrative rights** to avoid permission issues.
   - Open a Command Prompt with administrator privileges and run the application.

```
SmartDoorbellPrjDir python rtsp_client_V2.4.py
```

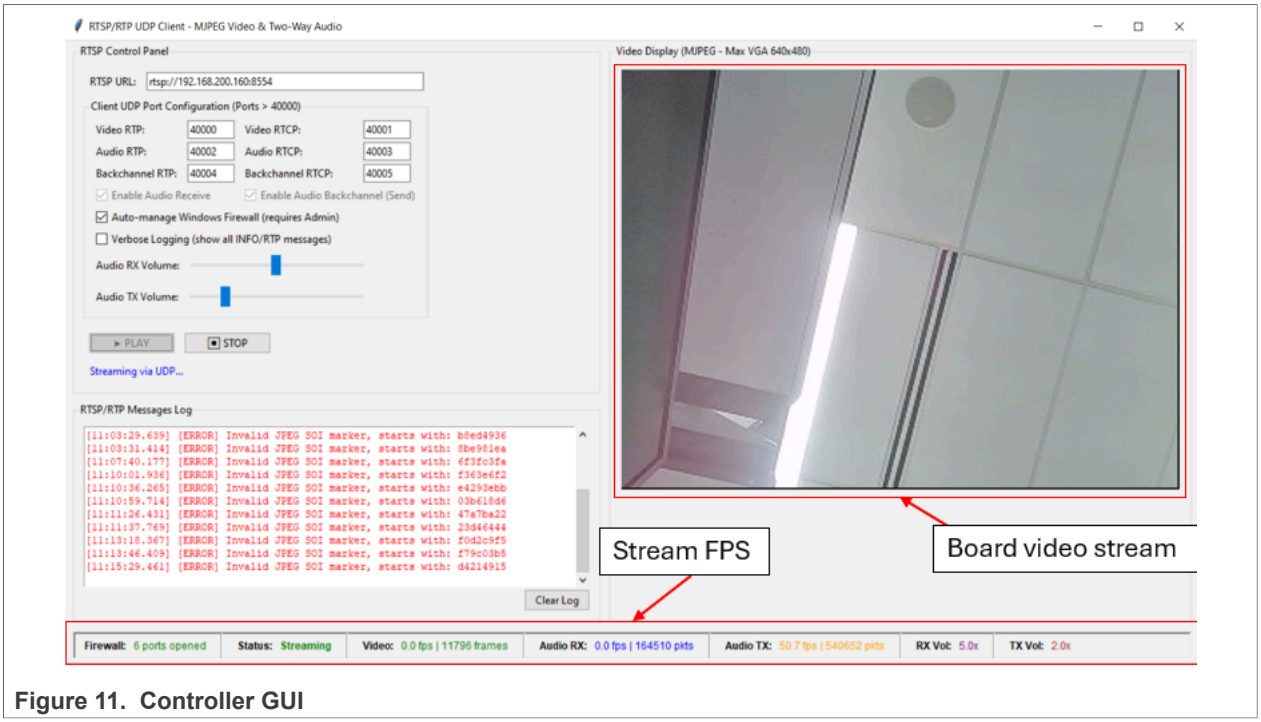

Figure 11. Controller GUI

- The controller GUI opens. In the RTSP URL field, enter:

```
rtsp://<board-IP>:8554
```

  You can find `<board-IP>` on the Core 0 serial console during boot.
- The session uses three UDP streams: video (RTP/RTCP), audio from the doorbell to the controller (RTP), and a back-channel audio stream from the controller to the doorbell. The user can enable/disable the audio channel/back-channel as required.
- Port numbers are configurable in the Client UDP port configuration panel.
- Check the **Auto-manage Windows Firewall** checkbox. If this option is not selected, the firewall can block some ports.
- **The RX Volume** option enables controlling the playback level of audio received from the doorbell. The **TX Volume** option controls the level of audio sent from the controller to the doorbell. Both are adjustable in the RTSP/RTP client interface.
- Plug a headphone into the audio jack J29. To start audio or video streaming, click the **Play** button.

5.5 Face registration and recognition

This section explains how to enroll one or more faces into the doorbell and then experience the face recognition feature end to end. Face embeddings are stored in RAM and are lost on every power cycle; re-enrollment is required after each reboot.

5.5.1 Prerequisites

Before starting, ensure that all the following requirements are met:

- The firmware is flashed and the board has booted successfully (Core 0 console active).
- The RGB + IR USB Camera module (J40) is connected and the LCD is displayed in the camera view.

- The system has woken up from the Sleep state. The LCD backlight must be ON (face detection, face recognition, standby, or AV streaming state). SW5 is silently ignored in Sleep state, INIT stage, and while a registration is already in progress.

5.5.2 Step 1: register a face (SW5 button method)

Face registration stores a 128-dimensional face embedding in the RAM. Up to 100 faces can be enrolled per session. Each face receives an automatically generated name based on an uptime timestamp (for example, "user1234567").

1. Wake the system from Sleep mode as described in the Prerequisites above.

2. Stand in front of the camera at 0.5 meters to 1.5 meters. Ensure that your face is well-lit and fully visible to both the RGB and IR sensors.

3. Press **SW5** (USER1 button) on the MIMXRT700 EVK board. The Core 0 console immediately prints two lines:

```
Reg request for user&lt;timestamp&gt;
 >> face_reg
```

4. Hold still for up to 1 second (the registration timeout). The system runs the face detection and feature extraction pipeline. On success, the console prints:

```
 Registration completed successfully in <N> ms
   >> standby
```

On timeout (no face detected within the configured timeout), the console prints:

```
 Registration timeout - forcing cancellation after <N> ms
 >> standby
```

5. The system automatically transitions to Standby state in both cases (success or timeout).

Note: ***SW5*** *is ignored if the system is in INIT, Sleep, or already in FACE_REGISTRATION state. If nothing happens after pressing* ***SW5****, verify that the LCD backlight is on and repeat from step 1. Pressing* ***SW5*** *a second time while registration is active displays "*`Already registered!`*" and is otherwise ignored.*

5.5.3 Step 1 (alternative): register via Shell command

To register a face with a custom name and timeout, use the shell command:

`simu_face_reg <name> <timeout_seconds>`

Example: `simu_face_reg JohnDoe 10`

This command sends `CTRL_TASK_FACE_REG_REQ` directly to the control task regardless of the button, but the system must not be in INIT or SLEEP state.

5.5.4 Step 2: verify face recognition

After enrollment, the system returns to Standby state. To verify recognition, perform the steps below:

1. Move away from the camera. After the standby timer expires (5 seconds by default), the system enters Sleep state and the LCD backlight turns off.
2. Approach the camera again within approximately 1 meter. The UWB/radar module triggers a wake event. The LCD turns on and face detection begins.

3. The system transitions to Face Recognition state. The NPU runs the recognition model against all enrolled embeddings.
4. If a match is found above the confidence threshold (default 0.96), the LCD displays "`Known`" and the MQTT message is as below: (if the MQTT is configured with a valid broker IP via `set_broker`).

```
Topic: doorbell/ring
Payload: Known
```

5. If no match is found, the visitor is treated as a stranger. "Unknown" is published to doorbell/ring and an A/V streaming session is initiated toward the indoor controller (if MQTT is configured with a valid broker IP via `set_broker`).

5.5.5 Step 3: simulate events via Shell (optional)

The following shell commands drive the control state machine without requiring a physical face or physical button press. They are useful for integration testing.

- Simulate presence detection (wake-up from Sleep): `simu_pres_det`
- Simulate a known face recognized event: `simu_face_rec_ind <name>`
- Example: `simu_face_rec_ind JohnDoe`
- Simulate an unknown face (stranger) event: `simu_face_rec_ind 0`

5.5.6 Notes and limitations

Face embeddings are stored in the RAM only. They are erased on every power cycle or board reset. For production deployments, implement persistent storage of the face database in external flash or EEPROM (see Best Practices and Recommendations section).

Recognition accuracy is highest under diffuse and uniform indoor lighting at a distance of 0.5 to 1.5 meters. Avoid strong backlighting or direct sunlight on the camera lens.

The default recognition confidence threshold is 0.96. A lower value increases recall but may produce false positives. The threshold is configured at runtime in the `mpp_app_start()` function call via the `mpp_app_params_t.recognition_threshold` field in (`core0/source/main.c`).

5.6 UWB/Radar module firmware update

This section describes the UWB/Radar module firmware update and verification procedure.

5.6.1 Update procedure

1. Obtain the `.bin` file for the new firmware from the latest NXP UWB IoT SDK package.
2. Use the bin-to-header conversion utility (typically `uwb-sr250/tools/bin2h.py`) to generate a new C header array from the `.bin` file.
3. Replace `core0/source/uwb/uwb-sr250/firmware_images/SR2XX/Mainline_Firmware.h` with the newly generated file.
4. Rebuild the doorbell project (see Section 5.3). The new firmware is embedded in the application binary and downloaded to the module on the next boot.

5.6.2 Verification

After flashing, verify on the Core 0 serial console:

```
>>> UWB init done
```

Move a hand or person within approximately 1 meter of the UWB antenna. At this step, presence detection events appear in the console, and the LCD wakes up from sleep.

6 Results and validation

This section summarizes the measured and observed results for each major subsystem of the Smart Doorbell Reference Design on the MIMXRT700 EVK Rev B.

6.1 Boot and connectivity

The system boots to face-recognition-ready state within approximately 5 seconds of being powered on. Both Core 0 and Core 1 serial consoles (115200 8N1) confirm successful initialization. The Wi-Fi association and IP address assignment steps complete within 10 seconds under typical network conditions. (The CLI triggers these steps). The assigned IP is displayed at the end of the Core 0 boot log.

6.2 Face recognition

Face detection, antispoofing, and recognition are validated using the dual RGB + IR camera with the eIQ inference engine running on Core 0. Detection operates at the camera frame rate (VGA MJPEG). The system correctly distinguishes live faces from photographs using IR channel cues. Recognition results (identity, confidence score, landmarks) are displayed on the 5.5-inch LCD. Up to 100 face embeddings can be enrolled per session.

6.3 Audio/video communication

The RTSP/RTP AV session is validated between the RT700 doorbell and the controller PC running the Python RTSP client. Video streams at the configured frame rate with latency suitable for real-time interaction. Full-duplex audio operates at 8 kHz or 16 kHz, 16-bit PCM. The back-channel stream reaches the doorbell speaker with latency acceptable for voice interaction. The controller GUI displays the live video feed with an FPS counter.

6.4 Presence detection

The UWB/Radar module reliably wakes the system when a person enters within approximately 1 meter. The Core 0 console message `>>> UWB init done` confirms successful firmware load and initialization over LPSPI16. State transitions (**Sleep -> Face Recognition -> Standby**) complete within the expected timeout periods defined in the core state machine.

7 Best practices and recommendations

The following recommendations are based on development and validation experience with the Smart Doorbell Reference Design.

7.1 Hardware

- Use MIMXRT700-EVK Rev B or later. Earlier revisions may have GPIO or connector differences that affect LCD backlight, camera, or power delivery.
- The UWB/Radar module antenna must face the same direction as the camera and LCD. Reversed orientation significantly degrades presence detection range and accuracy.
- Apply power (J45, 5 V DC, center positive, >= 1 A) in the last, after ensuring that all other connectors are seated.
  Important: *Hot-plugging the LCD FPC (J52) while the board is powered on can damage the MIPI-DSI interface.*
- Use a proper USB OTG cable for the camera (J40). Cables without the ID pin correctly wired prevent USB host enumeration of the dual-sensor camera.

7.2 Building and installing the SDK

- Always build via sysbuild from the workspace root. Building core0 standalone fails with a missing `core1_image.bin` linker error.
- After changing SDK association or toolchain path in `mcux_include.json`, delete the `flash_debug/` directory and perform a clean configure + build.
- Keep the west SDK workspace and the doorbell project on the same drive letter to avoid Windows path-length issues during CMake configuration.

7.3 Wi-Fi and networking

- The doorbell board and controller PC must be on the same Wi-Fi subnet. NAT or subnet separation prevent RTSP and MQTT connections.
- Assign a static IP or configure mDNS/DNS-SD to avoid stale IP addresses after board reboots. The RTSP URL must match the current board IP.
- Ensure that no local firewall rules block the RTSP port (8554) or the RTP/RTCP UDP ports configured in the RTSP client.

7.4 Face enrollment and production

- Face embeddings are stored in RAM and are lost on power cycle. For production deployments, implement persistent storage of face embeddings in external flash or EEPROM.
- Register faces under representative indoor lighting conditions and at a distance of 0.5 to 1.5 meters to maximize recognition accuracy.
- The system supports up to 100 enrolled faces per session. Implement enrollment management logic to prevent silent overwrites in production.

8 Troubleshooting

The following table describes common issues encountered during build, flash, and runtime, along with their likely causes and resolutions.

8.1 Build and configuration issues

This section describes the common issues encountered during build, flash, and runtime, along with their likely causes and resolutions.

Table 3. Troubleshooting

Symptom	Likely cause	Fix
`ARMGCC_DIR` not set	The toolchain environment variable is missing	Update ARMGCC_DIR in `core0/mcux_include.json`.
`SdkRootDirPath` not found	The wrong SDK path was used	Verify that the `SdkRootDirPath` points to `c:/workspace/mcuxsdk_github/mcuxpresso-sdk`.
Missing `core1_image.bin`	Sysbuild not used	Run `cmake` with -S pointing to sysbuild (see the Section above).
ninja: stale build errors	Incremental build broken	Delete `flash_debug/` and reconfigure.

Flash the firmware using the MCUXpresso VS Code extension (press **F5** or use the **Debug** toolbar). Two virtual COM ports appear on the host PC: one for Core 0 and another for Core 1. To monitor boot and runtime output, open both ports at 115200 8N1.

Figure 12 shows the Core 0 serial console output after successful Wi-Fi association. The assigned IP address is printed at the end of the boot log.

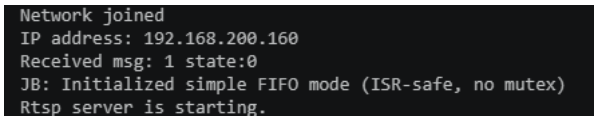

Figure 12. Core 0 serial console output after successful Wi-Fi association

Figure 13 shows the hardware connection between the serial console, the controller PC, and board.

Figure 13. Connection settings association

On the first boot, the LCD displays no username because no faces have been enrolled. Register at least one face before testing face recognition.

Figure 14 shows an unrecognized face.

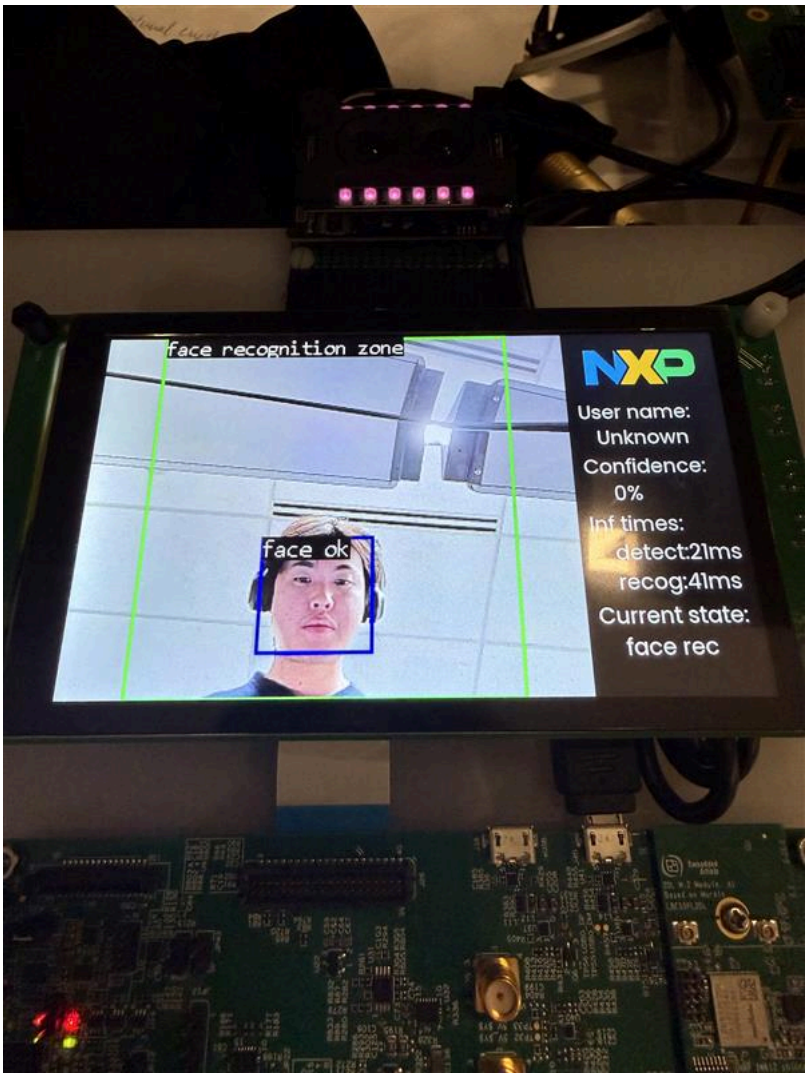

Figure 14. Face unrecognized

To enroll a face: stand in front of the camera, then press **SW5** (User 1 button). The system captures and stores the face embedding in RAM (up to 100 faces). The LCD displays `Known User` on subsequent successful recognitions.

Note: *Face embeddings are stored in RAM and are lost on the power cycle. Reenrollment is required after each reboot.*

Figure 15 shows the console message when a face is recognized.

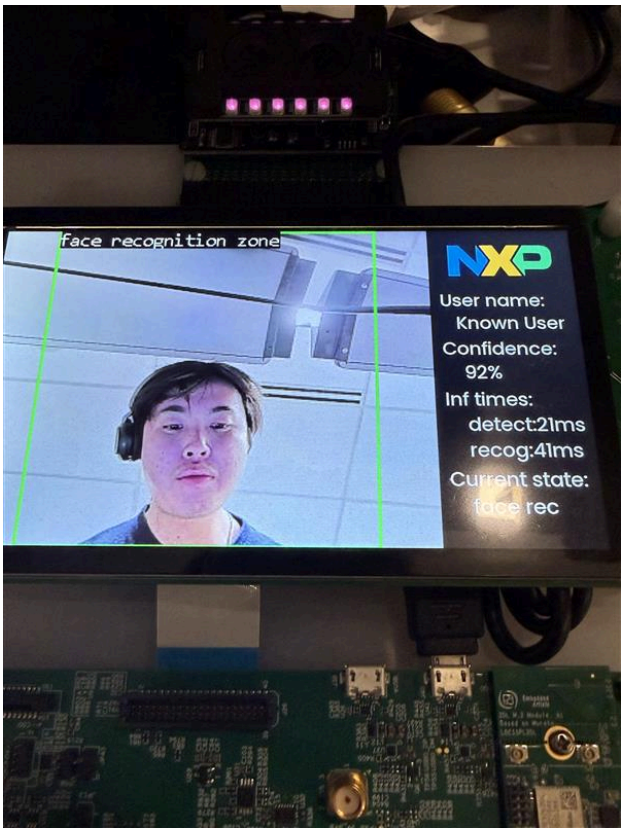

Figure 15. Face recognized

9 Conclusion

The Smart Doorbell Reference Design demonstrates the capability of the NXP i.MX RT700 MCU to serve as a high-performance, AI-enabled edge platform for intelligent access-control applications. The MCU integrates on-device face recognition, two-way audio/video communication, UWB/radar presence detection, and Wi-Fi connectivity on a single dual-core MCU. The design delivers a complete, production-relevant starting point for residential and commercial doorbell products.

The recommended next steps for developers extending this reference design are:

- Add persistent face enrollment storage in external flash for production-grade identity management.
- Integrate with commercial home-automation ecosystems via MQTT or Matter for broader ecosystem compatibility.
- Upgrade to a higher-resolution dual-sensor camera module for improved face recognition accuracy at longer distances.
- Explore other NXP eIQ AI models for gesture recognition, package detection, or multiperson tracking.

NXP software (MCUXpresso SDK, eIQ middleware, and VS Code extension) provides the complete toolchain needed to accelerate time to market for AI-powered embedded applications on the i.MX RT700 MCU.

10 Acronyms

Table 4 lists the acronyms used in this document.

Table 4. Acronyms

Acronym	Description
AI	Artificial intelligence
AV	Audio/video
CLI	Command-line interface
CMake	Cross-platform make build system
COM	Communication port
CPU	Central processing unit
DC	Direct current
DMIC	Digital microphone interface/microphone
DNS-SD	DNS service discovery
DSP	Digital signal processor
EEPROM	Electrically erasable programmable read-only memory
EVK	Evaluation kit
FPC	Flexible printed circuit
FPS	Frames per second
GPIO	General-purpose input/output
GUI	Graphical user interface
I2C	Inter-integrated circuit
IoT	Internet of things
IP	Internet protocol
IR	Infrared
JPEG	Joint photographic experts group
LCD	Liquid crystal display
LVGL	Light and versatile graphics library
MCU	Microcontroller unit
mDNS	Multicast DNS
MIPI-DSI	Mobile industry processor interface display serial interface
MJPEG	Motion JPEG
MQTT	Message queuing telemetry transport
NAT	Network address translation
NPU	Neural processing unit
OTG	On-the-go
PCM	Pulse code modulation
PDM	Pulse density modulation

Table 4. Acronyms...*continued*

Acronym	Description
PSIRT	Product security incident response team
RAM	Random access memory
RGB	Red green blue
RTCP	Real-time transport control protocol
RTOS	Real-time operating system
RTP	Real-time transport protocol
RTSP	Real time streaming protocol
SDIO	Secure digital input output
SDK	Software development kit
SoC	System on chip
SPI	Serial peripheral interface
TCP/IP	Transmission control protocol/Internet protocol
UART	Universal asynchronous receiver-transmitter
UDP	User datagram protocol
USB	Universal serial bus
UVC	USB video class
UWB	Ultra-wideband
VGA	Video graphics array
VS	Visual studio
Wi-Fi	Wireless fidelity
YUV	Luma-chroma video color encoding

11 References

1. NXP i.MX RT700 Crossover MCU — MIMXRT700-EVK Product Page: https://www.nxp.com/MIMXRT700-EVK
2. NXP MCUXpresso SDK v26.6.0 — GitHub Manifest Repository: https://github.com/nxp-mcuxpresso/mcuxsdk-manifests
   Branch: release/26.06.00-lts
3. MCUXpresso for VS Code Extension — VS Code Marketplace: https://marketplace.visualstudio.com/items?itemName=NXPSemiconductors.mcuxpresso
4. NXP eIQ Machine Learning Software for Embedded Devices: https://www.nxp.com/eiq
5. ETNA TS250 DEVKIT: Contact NXP for availability
6. NXP RK055HDMIPI4MA0 5.5-inch MIPI-DSI LCD panel: https://www.nxp.com/part/RK055HDMIPI4MA0
7. Embedded Artists IW612 Wi-Fi module (EAR00422): https://www.digikey.com
8. ARM GNU Toolchain 14.2.Rel1: https://developer.arm.com/downloads/-/arm-gnu-toolchain-downloads
9. LVGL Embedded Graphics Library: https://lvgl.io
10. west Build Meta-Tool (Zephyr Project): https://docs.zephyrproject.org/latest/develop/west/index.html

12 Revision history

Table 5 summarizes revisions to this document.

Table 5. Revision history

Document ID	Release date	Description
AN15042 v.1.0	24 September 2026	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents