IRONKERNEL EMBEDDED

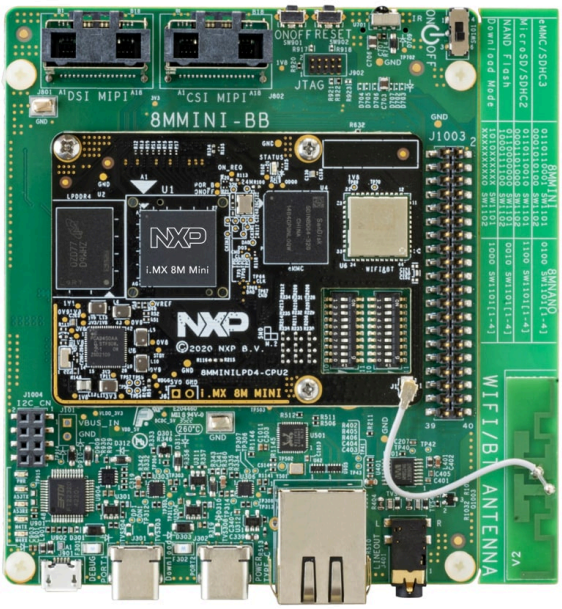
IronForge on the i.MX 8M Mini EVK

Embedded Linux with USB network boot and a microSD boot image

Connectivity
M.2 PCI Express, Wi-Fi and Gigabit Ethernet

Storage
eMMC flash, microSD access and QSPI NOR

Audio and display
Stereo audio and HDMI via a MIPI DSI adapter



i.MX 8M Mini LPDDR4 EVK family. Product photograph courtesy of NXP Semiconductors.

IronForge builds the bootloader, Linux kernel and a BusyBox-based userspace on a native Windows workstation. Developers can bring up the EVK from USB, inspect it over serial and iterate on the kernel, device tree and userspace over Ethernet. The network boot path leaves the installed startup image available for standalone boot.

Platform	Configuration
Processor	NXP i.MX 8M Mini Quad, four Arm Cortex-A53 cores and a Cortex-M4
Memory and storage	2 GiB LPDDR4, 16 GB eMMC 5.1, 32 MiB QSPI NOR and removable microSD
Connectivity	Gigabit Ethernet, USB 2.0 Type-C and USB UART; SDIO Wi-Fi depends on module revision
Audio and display	WM8524 stereo line output; MIPI DSI display connector with a separate HDMI adapter
Boot and userspace	U-Boot 2026.10, Trusted Firmware-A 2.12.0, Linux 7.3.4 and BusyBox 1.38.0

The LPDDR4 kit uses managed NAND inside its eMMC package and a separate QSPI NOR chip. NXP assigns raw NAND boot to the DDR4 EVK. NXP LPDDR4 EVK hardware guide

From USB download to Linux

In serial download mode the on-chip ROM accepts the SPL from NXP's Universal Update Utility. SPL trains LPDDR4 and receives the rest of the boot image over USB. Arm Trusted Firmware hands off to U-Boot, whose prompt is available on the A53 serial port at 115200 baud.

UART CAPTURE / USB RAM BOOT

```
U-Boot SPL 2026.10 (Jan 01 2025 - 00:00:00 +0000)
Trying to boot from USB SDP
Downloading file of size 1182616 to 0x40400000... done
NOTICE:  BL31: v2.12.0(release):development build
U-Boot 2026.10 (Jan 01 2025 - 00:00:00 +0000)
CPU:   NXP i.MX8MMQ Rev1.0 A53 at 1200 MHz
Model: FSL i.MX8MM EVK board
DRAM:  2 GiB
MMC:   FSL_SDHC: 1, FSL_SDHC: 2
Net:   eth0: ethernet@30be0000
u-boot=>
```

Selected lines from the EVK's UART capture during a USB RAM boot. This capture shows the ROM download handoff through the U-Boot prompt.

Step	Development workflow
Load over USB	The workstation sends the IronForge boot image using UUU's `spl` mode.
Fetch Linux	U-Boot uses DHCP and TFTP over wired Ethernet to load `Image` and the board device tree from the workstation.
Mount the root filesystem	Linux uses NFSv3 over TCP to mount the shared IronForge root filesystem. BusyBox starts the serial shell.
Inspect the target	Serial remains available during network changes. The demo adds key-based SSH and a board-hosted HTTP dashboard for remote work.

The image build also produces a microSD layout with an ext4 root filesystem and a RAM rescue entry. The USB/NFS route is suited to frequent edits during bring-up, while the microSD layout supports standalone deployment.

On the test board, U-Boot obtained an address over wired DHCP and reached the workstation. The soldered eMMC reported a 14.7 GiB user area in HS400ES mode. The inserted microSD card reported 29.7 GiB and negotiated SDR104. U-Boot probed the 32 MiB QSPI NOR with 256-byte pages and 4 KiB erase sectors and read the first 256 bytes successfully.

Ethernet handoff to Linux

In the serial capture below, U-Boot negotiates the Ethernet link, obtains a DHCP lease, and transfers the 13,971,464-byte kernel image and 56,188-byte device tree. U-Boot reported **5.9 MiB/s** for the kernel transfer. It passes the device tree to Linux, along with the address of the NFSv3 export.

```
EVK SERIAL CAPTURE  /  NETWORK BOOT

ethernet@30be0000 Waiting for PHY auto negotiation to complete..... done
BOOTP broadcast 1
DHCP client bound to address [LAN address] (18 ms)
Using ethernet@30be0000 device
TFTP from server [LAN address]; our IP address is [LAN address]
Filename 'Image'.
Load address: 0x40480000
Loading: *#################################################################
         5.9 MiB/s
done
Bytes transferred = 13971464 (d53008 hex)
Using ethernet@30be0000 device
TFTP from server [LAN address]; our IP address is [LAN address]
Filename 'system.dtb'.
Load address: 0x43000000
Loading: *#####
         4.5 MiB/s
done
Bytes transferred = 56188 (db7c hex)
Moving Image from 0x40480000 to 0x40600000, end=0x413b0000
## Flattened Device Tree blob at 43000000
   Booting using the fdt blob at 0x43000000
Working FDT set to 43000000
   Loading Device Tree to 000000004ffef000, end 000000004ffffb7b ... OK
Working FDT set to 4ffef000
Starting kernel ...
```

UART transcript rendered from the physical board. Network addresses are masked in the figure.

Linux and the NFS root

Linux brings all four Cortex-A53 cores online, opens the wired link at 1 Gb/s full duplex and mounts the IronForge NFS export. BusyBox starts the serial shell, the board HTTP server and the application startup hooks. The root uses NFSv3 over TCP, while the temporary workspace is a RAM filesystem.

```
EVK SERIAL CAPTURE  /  LINUX SHELL

IP-Config: Complete:
     device=eth0, hwaddr=[MAC], ipaddr=[LAN address], mask=[LAN address], gw=[LAN address]
     host=[LAN address], domain=, nis-domain=(none)
     bootserver=[LAN address], rootserver=[LAN address], rootpath=
     nameserver0=[LAN address], nameserver1=[LAN address]
     ntpserver0=[LAN address]
cfg80211: Loading compiled-in X.509 certificates for regulatory database
Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
Loaded X.509 cert 'wens: 61c038651aabdcf94bd0ac7ff06c7248db18c600'
clk: Disabling unused clocks
faux_driver regulatory: Direct firmware load for regulatory.db failed with error -2
cfg80211: failed to load regulatory.db
PM: genpd: Disabling unused power domains
ALSA device list:
  #0: wm8524-audio
VFS: Mounted root (nfs filesystem) on device 0:17.
devtmpfs: mounted
VFS: Pivoted into new rootfs
Freeing unused kernel memory: 1664K
Run /init as init process
mount: mounting devtmpfs on /dev failed: Resource busy
IronKernel.com | IronForge ARM bring-up
Hardware acceptance required. Dashboard: http://<DHCP address>/
init started: BusyBox v1.38.0 (2025-01-01 00:00:00 UTC)
starting pid 143, tty '': '/bin/true'
starting pid 144, tty '': '/bin/cttyhack /bin/sh'
BusyBox v1.38.0 (2025-01-01 00:00:00 UTC) built-in shell (ash)
# random: crng init done
```

The Linux console confirms the root mount and shell. Network addresses are masked.

Network and application demos

The workbench runs entirely in a browser and on the A53 serial console. The board serves its own HTTP status page with Linux CPU and memory readings, wired and wireless interfaces, the root mount, storage devices, the audio card and display connector state. A SQLite sampler takes a reading about every five seconds and keeps the latest 120 samples in `/tmp`. The page uses those samples to calculate recent receive and transmit rates for each interface.

Application	What to demonstrate
BusyBox HTTP server	Open the board's IP address and follow live system and network activity while a workload runs.
SQLite	Query the local sample database from the serial shell or SSH. Compare CPU load and Ethernet traffic before and during a benchmark.
CoreMark	Compare a single pinned worker with four concurrent workers, one per A53 core. Retain each worker's validation result and score.
iperf3	Measure TCP throughput in both directions against a host. Bind the client to the WLAN address to measure the wireless link separately.
wpa_supplicant	Scan for nearby networks, then join a chosen network using a private configuration. The wireless routing table keeps the wired NFS path available.
Dropbear	Install a workstation's Ed25519 public key over serial. Verify the printed host fingerprint before connecting. The demo keeps keys in RAM for the current boot.
Mosquitto	Run a broker and exchange a local publish and subscribe message with the bundled MQTT clients.

Application checks on the EVK

Four concurrent CoreMark workers pinned to A53 cores 0–3 validated their results and measured **3,090**, **3,911**, **3,757** and **3,889 iterations/s**. The local SQLite database accepted samples and Linux identified both MMC devices and the 32 MiB NOR. Dropbear accepted an Ed25519 key provisioned over serial, and Mosquitto delivered a message from the publisher to the subscriber over loopback.

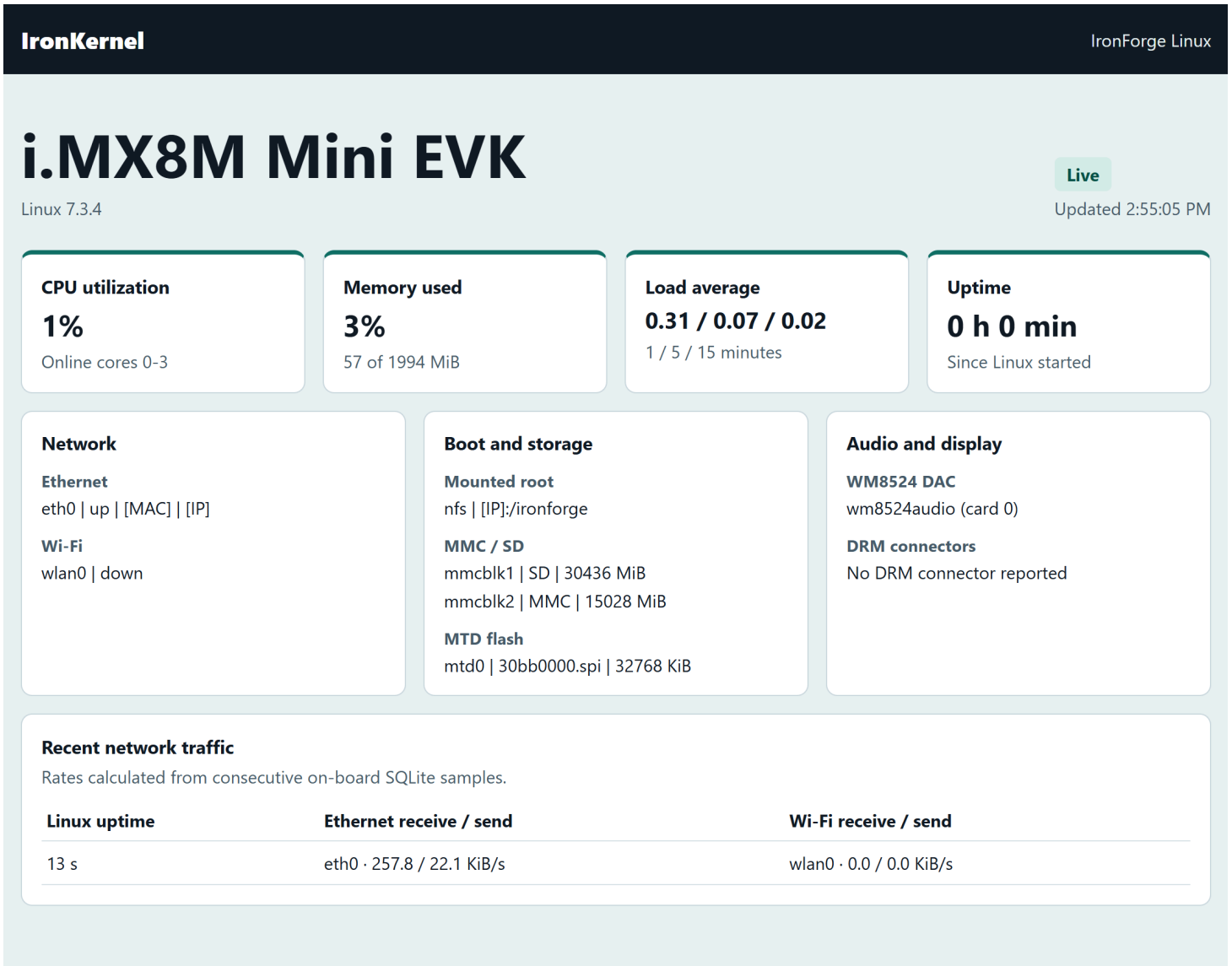
```
EVK SERIAL CAPTURE  /  APPLICATIONS

IRONFORGE APPLICATION CHECKS
Linux 7.3.4
Online cores 0-3
mtd0: 02000000 00001000 "30bb0000.spi"
/dev/mmcblk2  /dev/mmcblk1
iperf 3.18 (cJSON 1.7.15)
card 0: wm8524audio [wm8524-audio], device 0: sai-tx-rx-wm8524-hifi wm8524-hifi-0 [sai-tx-rx-w
m8524-hifi wm8524-hifi-0]
# ironforge-wifi scan wlan0; sleep 5; ironforge-wifi results wlan0 | wc -l
OK
Snapshot /tmp/cube-shots/cube-a.ppm
Snapshot /tmp/cube-shots/cube-b.ppm
Snapshot /tmp/cube-shots/cube-c.ppm
Rendered three 960x540 cube frames in memory
IRONFORGE APPLICATION END
```

Selected lines from the live application run on the EVK.

Live board dashboard

The screenshot shows the EVK serving system status over Ethernet. The application reads counters directly from Linux and calculates rates from on-board SQLite samples. The QCA9377 radio initialized, and a Wi-Fi scan returned 24 entries. Joining a wireless network requires network credentials.



Browser capture from the running EVK. Network addresses are masked.

Wired throughput and Wi-Fi scan

The EVK ran iperf3 as the server while the Windows workstation used the wired gigabit link to test both directions. Reported throughput is the receiver's application payload rate. UDP tests include the requested send rate, datagram size and receiver loss.

Protocol and offered rate	Direction	Receiver rate	Packet loss
TCP	Workstation to EVK	943 Mb/s	—
TCP	EVK to workstation	947 Mb/s	—
UDP at 100 Mb/s	Workstation to EVK	76.4 Mb/s	23%
UDP at 100 Mb/s	EVK to workstation	100 Mb/s	0%
UDP at 500 Mb/s	Workstation to EVK	372 Mb/s	25%
UDP at 500 Mb/s	EVK to workstation	500 Mb/s	0%

Each test used a single stream for five seconds. UDP used 1,200-byte datagrams. The receive-side UDP loss is specific to this test setup and shows why offered rate and received rate are reported separately.

```
iperf3 -c <evk-ip> -t 5
iperf3 -c <evk-ip> -t 5 -u -b 500M -l 1200
```

Add `-R` to either workstation command to measure traffic from the EVK.

The EVK also scanned with the onboard QCA9377 while the root filesystem remained mounted over Ethernet. Two scans returned **24 and 26 BSS entries**. A single network can appear more than once when multiple access points advertise the same name.

```
ip link show wlan0
ironforge-wifi scan wlan0
ironforge-wifi results wlan0
```

After joining a test network and obtaining an address, bind the iperf3 client to that WLAN address to measure the wireless path separately from the Ethernet session.

Live Wi-Fi scan and access points

The EVK's QCA9377 accepted a scan on `wlan0` and returned **22 BSS entries** in this capture. The results show each advertised network name (SSID), frequency, received signal and security flags. Yellow bars cover the access point MAC addresses (BSSIDs). Entries span 2.4 GHz and 5 GHz. Access points can share the same SSID.

```
EVK SERIAL CAPTURE  /  WIFI SCAN

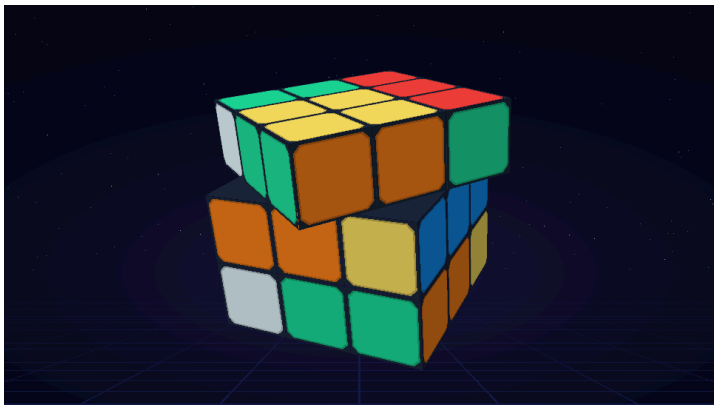
ironforge-wifi scan wlan0
OK
```

```
EVK SERIAL CAPTURE  /  WIFI RESULTS

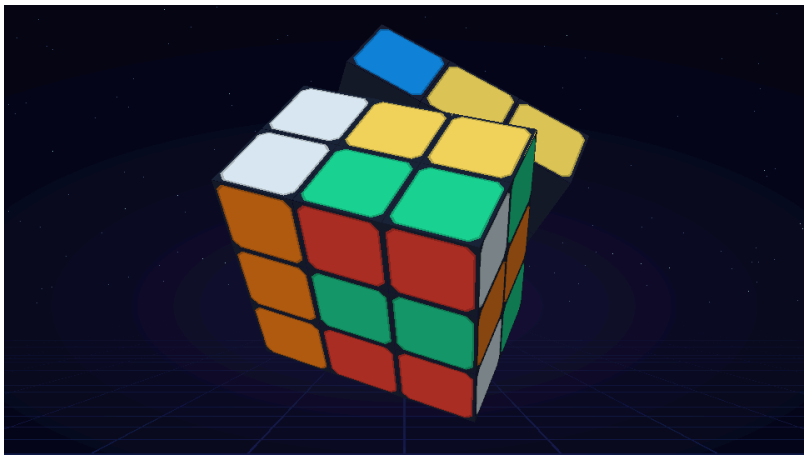
# ironforge-wifi results wlan0
BSSID              MHz  dBm  SSID                Flags
[redacted]         5220  -35                      [WPA2-PSK-CCMP][ESS]
[redacted]         5220  -36  SpecRacer009         [WPA2-PSK-CCMP][ESS]
[redacted]         5220  -54  SpecRacer009         [WPA2-PSK-CCMP][ESS]
[redacted]         5220  -54                      [WPA2-PSK-CCMP][ESS]
[redacted]         5220  -66                      [WPA2-PSK-CCMP][ESS]
[redacted]         5220  -67  SpecRacer009         [WPA2-PSK-CCMP][ESS]
[redacted]         5300  -68  ATTPhVx4RP           [WPA2-PSK-CCMP][ESS]
[redacted]         5220  -69  SpecRacer009         [WPA2-PSK-CCMP][ESS]
[redacted]         5220  -70                      [WPA2-PSK-CCMP][ESS]
[redacted]         2437  -31  SpecRacer009         [WPA2-PSK-CCMP][ESS]
[redacted]         2437  -51                      [WPA2-PSK-CCMP][ESS]
[redacted]         2437  -52  SpecRacer009         [WPA2-PSK-CCMP][ESS]
[redacted]         2437  -64                      [WPA2-PSK-CCMP][ESS]
[redacted]         2462  -64  ATTPhVx4RP           [WPA2-PSK-CCMP][ESS]
[redacted]         2457  -66  SpecRacer009         [WPA2-PSK-CCMP][ESS]
[redacted]         2437  -67  SpecRacer009         [WPA2-PSK-CCMP][ESS]
[redacted]         5745  -76                      [WPA2-PSK-CCMP][ESS]
[redacted]         5745  -76  NETGEAR99            [WPA2-PSK-CCMP][SAE-H2E][ESS]
[redacted]         2417  -73  NETGEAR99            [WPA2-PSK-CCMP][SAE-H2E][ESS]
[redacted]         2437  -77  ATT7yh16dp           [WPA2-PSK-CCMP][ESS]
[redacted]         2437  -87  SpectrumSetup-28     [WPA2-PSK-CCMP][ESS]
[redacted]         5745  -76  Kitchen speaker.d    [ESS]
```

IronKernel cube demo

The A53 rendered these three frames directly on the EVK with the same software rasterizer used for the Zybo demo. The headless capture mode writes 960 × 540 frames to RAM. Each image records a different point in the cube animation. This mode also works with the display connector empty.







```
mkdir -p /tmp/cube-shots
ironforge-cube --headless --snapshots /tmp/cube-shots
```

eMMC performance

The soldered eMMC negotiated HS400 enhanced strobe mode. IronForge measured sequential file I/O on its ext4 partition with the system running from wired NFS. Each pass used a 512 MiB temporary file, the Linux `O_DIRECT` flag and one synchronous I/O at a time. Timed writes include `fdatasync`.

Block size	Sequential read	Sequential write
4 KiB	69.12 MiB/s	30.80 MiB/s
128 KiB	275.08 MiB/s	79.41 MiB/s
1 MiB	308.61 MiB/s	79.18 MiB/s

These figures describe this EVK's eMMC and filesystem. Larger operations raised read throughput substantially. Write throughput leveled off at about 79 MiB/s at the tested larger block sizes. The temporary file was unlinked during the test and the partition was unmounted afterward.

Storage, audio and display

The EVK can be evaluated from the workstation browser and serial console. Local display evaluation uses the EVK's MIPI-DSI-to-HDMI accessory. The board's analog jack is a stereo line output for an amplified speaker or line input. Analog capture requires an external ADC or USB audio interface.

Evaluation path	IronForge tooling
eMMC and microSD	Identify each device and its capacity before reading. The removable card can also carry the generated standalone boot image.
QSPI NOR	Identify the geometry with `sf probe`, then read a small region into RAM. NOR boot is a separate validation step.
M.2 PCIe	Linux enumerated the EVK root port at `00:00.0` with the slot empty. The image includes NVMe support. Use `lspci` after cold boot to identify a fitted endpoint.
DAC playback	List ALSA devices with `aplay -l`. Play the included two-second PCM tone through the WM8524 and connect amplified speakers to assess line-level output.
Display	Inspect the Linux DRM connector after attaching the MIPI-to-HDMI adapter. A connected sink is required to verify visible output.
ADC and audio capture	Add an external converter or USB audio input and validate recordings through the corresponding Linux driver and tools.

Contact IronKernel

Contact IronKernel to discuss integrating IronForge on an i.MX target, adapting a reference design or developing a custom U-Boot, Linux device tree or peripheral driver. Send your target board or module and your project goal.

https://ironkernel.com/ · support@ironkernel.com