

SCAiLX AI platform



Videology Industrial-Grade Cameras

Excellence for More Than 25 Years

Founded in 1995, Videology is a global leader in the design, engineering and manufacturing of industrial-grade embedded video cameras, related systems, software, and solutions. For more than 25 years we have been providing performance excellence in a broad spectrum of applications including biomedical devices, life sciences, banking, aerospace, traffic management, photo ID, pipe inspection and more.

Over 1 Million Cameras Worldwide

At Videology, we specialize in meeting the customized specification requirements of OEMs, large-scale integrators and other partners, which have resulted in the delivery of over 1 million embedded cameras worldwide. We are an ISO 9001-certified company headquartered in Mansfield, Massachusetts (part of the greater Boston area), and our European operations are located in Eindhoven, the Netherlands. Read more about our new Eindhoven facility [here](#).

Our Brand Difference

Our deep commitment to the customer experience delivers performance excellence throughout the entire customer journey. This is Videology's brand difference and it's our company's most important priority in serving the needs of our customers across the globe.

Our Brand Promise

How do we support our brand difference? We do so with a sincere promise we make to every Videology customer as follows: We provide competence, attention to detail and personal care with a level of excellence that will delight every customer in every interaction. This is Videology's brand promise and it's been the key to our growth and success – from a small start-up more than 25 years ago to a global leader in today's imaging industry.

Prior to Using

Videology reserves the right to modify the information in this document as necessary and without notice. It is the user's responsibility to be certain they possess the most recent version of this document by going to www.videologyinc.com, searching for the model number, and comparing revision letters on the respective document, located in the document's footer.

License Agreement (Software)

This Agreement states the terms and conditions upon which Videology Industrial-Grade Cameras (hereafter referred to as "Videology") offer to license to you the software together with all related documentation and accompanying items including, but not limited to, the executable programs, drivers, libraries, and data files associated with such software.

The Software is licensed, not sold, to you for use only under the terms of this Agreement.

Videology grants to you, the purchaser, the right to use all or a portion of this Software provided that the Software is used only in conjunction with Videology's family of products.

In using the Software you agree not to:

- Decompile, disassemble, reverse engineer, or otherwise attempt to derive the source code for any Product (except to the extent applicable laws specifically prohibit such restriction);
- Remove or obscure any trademark or copyright notices.

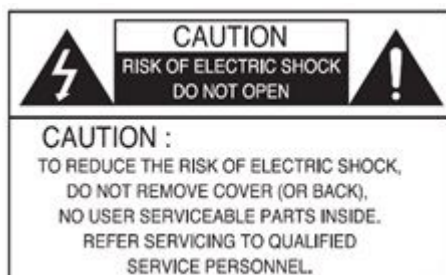
Limited Warranty (Hardware and Software)

ANY USE OF THE SOFTWARE OR HARDWARE IS AT YOUR OWN RISK. THE SOFTWARE IS PROVIDED FOR USE ONLY WITH VIDEOLOGY'S HARDWARE. THE SOFTWARE IS PROVIDED FOR USE "AS IS" WITHOUT WARRANTY OF ANY KIND, TO THE MAXIMUM EXTENT PERMITTED BY LAW, VIDEOLOGY DISCLAIMS ALL WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, QUALITY AND FITNESS FOR A PARTICULAR APPLICATION OR PURPOSE. VIDEOLOGY IS NOT OBLIGATED TO PROVIDE ANY UPDATES OR UPGRADES TO THE SOFTWARE OR ANY RELATED HARDWARE.

Limited Liability (Hardware and Software)

In no event shall Videology or its Licensor's be liable for any damages whatsoever (including, without limitation, incidental, direct, indirect, special or consequential damages, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use or inability to use this Software or related Hardware, including, but not limited to, any of Videology's family of products.

Warning and Safeguards



- **Read instructions before operating camera.**
- Please read/follow all instructions and read all warnings before operating the camera.
- Installation and servicing should only be done by Qualified Service and Installation Personnel.
- Installation shall be done in accordance with all local and national electrical and mechanical codes.
- Avoid mounting in direct sunlight.
- To reduce the risk of fire or electric shock, do not expose this appliance to rain, water or wet locations.
- If the camera is to be mounted outdoors a secondary waterproof enclosure should be used.

Precautions

- Do not put objects inside the unit. Make sure that no metal objects or flammable substances get inside the camera. It could cause fire, short-circuits or damage.
- Be careful when handling the unit.
- To prevent damage, do not drop the camera or subject it to strong shock or vibration.
- Install away from electric or magnetic fields.
- Protect from humidity and dust.
- Protect from high temperature.
- Be careful when installing the camera close to the ceiling, in a kitchen or boiler room, as the temperature may raise to high levels.
- Cleaning - Dirt can be removed from the cabinet only by wiping it with a soft cloth moistened with a soft detergent solution.
- Mounting Surface - The mounting surface material must be strong enough to secure the camera.
- Avoid viewing a very bright object (such as light fittings) during an extended period.

Care of the unit

- Remove dust or dirt on the surface of the lens with a blower (commercially available).
- Avoid the use of volatile solvents such as thinners, alcohol, benzene and insecticides. They may damage the surface finish and/or impair the operation of the camera.
- Be careful not to spill water or other liquids on the unit.

Operation and Storage

- Avoid extremely hot or cold places; operating temperature -40 °C - 60 °C (-40 °F - 140 °F)
- However, we recommend that the unit be used within a temperature range of 0 °C - 45 °C (32 °F - 113 °F)
- Avoid damp or dust places
- Avoid places exposed to rain
- Avoid places subject to strong vibration
- Avoid places close to generators of powerful electromagnetic radiation such as radio or TV transmitters.



- If the product is to be put out of operation definitively, take it to a local recycling plant for disposal which is not harmful to the environment.

Aegis Electronic Group
www.aegiselect.com

Document History

Revision	Issue Date	Reason	CN#
A	07-18-2024	First release	24-0031
B	08-05-2025	Release FW 0.11.5	25-0028

New under release 0.11.5

Hardware/software support for:

SCAILX-ACC26	SCAILX-PCIE + HAILO-8 M.2 2230 AI Accelerator
SCAILX-WIFI	SCAILX-PCIE + Intel AX210 Wifi/BT module

Table of Contents

1	Getting Started with SCAiLX	13
1.1	Find the device on the network	13
1.2	SSH connect to the SCAiLX device	14
1.3	SCAiLX SSH key	15
1.4	Video streaming (standard)	16
1.5	Web browser (GitHub source)	16
2	SCAiLX Hardware manual	19
2.1	SCAiLX Modules	19
2.1.1	SCAiLX-SOM-AI board	19
2.1.2	SCAiLX-LVDS-2-MIPI board	20
2.1.3	SCAiLX-ETH-POE board	20
2.1.4	SCAiLX-ETH-DC board	21
2.1.5	SCAiLX-2GS234-xY camera board	22
2.1.6	SCAiLX-ACC26 AI Accelerator	23
2.1.7	SCAiLX-WIFI	23
2.2	SCAiLX-SOM-AI	24
2.2.1	SCAiLX-SOM-AI	24
2.2.1.1	General description	24
2.2.1.2	SCAiLX-SOM-AI (rev. 4.0)	24
2.2.1.3	SCAiLX-SOM-AI	25
2.2.1.4	Features	25
2.2.1.5	Dimensions	26
2.2.2	SCAiLX-SOM board - Connector details	27
2.2.2.1	SCAiLX-SOM board - Connector details	27
2.2.3	SCAiLX-SOM board - Interfaces description	33
2.2.3.1	Board connectors	35
2.2.3.2	Datasheet	39
2.3	SCAiLX-LVDS-2-MIPI	39
2.3.1	SCAiLX-LVDS-2-MIPI (rev. 4.0)	39
2.3.1.1	Dimensions	40

2.3.1.2	Connection	40
2.3.1.3	KEL connector pinout	41
2.3.2	Programming and setup	42
2.3.3	Datasheet	42
2.3.4	Step 3D model	42
2.4	SCAILX-ETH-POE	42
2.4.1	SCAILX-ETH-POE board	42
2.4.1.1	Dimensions	43
2.4.1.2	Connection	43
2.4.1.3	Programming and setup	44
2.4.1.4	Datasheet	44
2.4.1.5	Step 3D model	44
2.5	SCAILX-ETH-DC	44
2.5.1	SCAILX-ETH-DC board	44
2.5.1.1	Dimensions	45
2.5.1.2	Connection	45
2.5.1.3	Programming and setup	46
2.5.1.4	Datasheet	46
2.5.1.5	Step 3D model	46
2.6	SCAILX-2GS234-xY camera	46
2.6.1	SCAILX-2GS234-xY 2 MP Global Shutter board level camera	46
2.6.1.1	Models	47
2.6.1.2	Blockdiagram	48
2.6.1.3	Specification	48
2.6.1.4	Dimensions	49
2.6.1.5	Connectors	49
2.6.1.6	Single and Dual camera configuration	50
2.6.1.7	ISP functionality	51
2.6.2	User Guide	52
2.6.3	Programming and setup	52
2.6.4	Software support	52
2.6.5	Datasheet	52
2.6.6	Step 3D model	52
2.7	SCAiLX - Peripheral board templates	53
2.7.1	Altium template	53

2.7.2	Kicad template	53
2.8	SCAILX-ACC26.....	54
2.8.1	SCAILX-ACC26 - SCAILX AI Accelerator board	54
2.8.2	Dimensions.....	55
2.8.3	SCAILX 12 Volt POWER IN	55
2.8.4	Board connectors.....	55
2.8.5	Pin description P1-P3	56
2.8.6	Assembly on SCAILX-SOM-AI	56
2.8.7	Alternative assembly	57
2.8.8	Datasheet	57
2.8.9	3D-Step Model.....	57
2.9	SCAILX-WIFI.....	57
2.9.1	SCAILX-WIFI - SCAILX WiFi PCIe interfacing.....	57
2.9.2	Dimensions.....	58
2.9.3	SCAILX 12 Volt POWER IN	58
2.9.4	Board connectors.....	59
2.9.5	Pin description P1 - P3	59
2.9.6	Assembly on SCAILX-SOM-AI	59
2.9.7	Alternative assembly	60
2.9.8	Specifications Intel AX210 PCIe module.....	60
2.9.8.1	Wi-Fi network setup	61
2.9.8.2	Bluetooth setup.....	61
2.9.9	Datasheet	62
2.9.10	3D-Step Model.....	62
3	SCAILX-ZB - Zoom block.....	63
3.1	Assembly	63
3.1.1	Datasheet	63
3.1.2	3D step model	63
4	SCAILX-CAM board level camera	64
4.1	Assembly	64
4.1.1	Assembled camera	64
4.1.2	SCAILX-DEV kit.....	65
4.1.3	Resources for the SCAILX-CAM.....	65
4.1.4	Datasheets and User Guide.....	65
5	SCAiLX Software manual.....	66

5.1	Software Update.....	66
5.1.1	SWUpdate web page.....	66
5.2	Playing a RTSP stream	68
5.2.1	Start a RTSP stream on device	68
5.2.2	gststreamer player on host	68
5.2.3	VLC media player on host	68
5.3	Web streaming - webRTC support	69
5.3.1	Stream to web	69
5.3.2	go2rtc application	70
5.3.2.1	Implementing a simple web server using go2rtc stream	71
5.4	Running AI models in python with Tensorflow-lite	71
5.4.1	SCAiLX AI application in Python3 – Object detection	71
5.4.2	Preparation and running AI application	71
5.4.3	Development of your own AI application	74
5.4.4	Useful tools	74
5.4.4.1	Source code	74
5.5	SCAILX-2GS234 - Global Shutter Camera - Software.....	74
5.5.1	SCAILX-2GS234 V4L commands.....	75
5.5.1.1	Description	75
5.5.1.2	TUI - Text User Interface.....	75
5.5.1.3	User Controls.....	76
5.5.1.4	Camera controls.....	79
5.5.1.5	Exposure Limits	83
5.5.1.6	Image Processing Controls.....	86
5.5.1.7	Detection Controls.....	86
5.5.2	Dual camera stream using the SCAILX-2GS234 Cameras.....	87
5.5.2.1	Starting the RTSP server	88
5.5.2.2	Play the streams on a PC	89
5.5.2.3	V4L camera controls	89
5.5.2.4	Synchronize both cameras using one camera as master.....	90
5.5.2.5	Synchronizing using a trigger GPIO from SCAILX	90
5.5.3	SCAILX-2GS234 python scripts.....	91
5.5.3.1	SCAILX-2GS234 Sync and Trigger Configuration	92
5.6	SCAILX-LVDS-2-MIPI zoom-block interface - Software	93
5.6.1	Python Helper functions (as of Scailx SW release 0.10.3).....	94

5.6.2	TTY interface	95
5.7	Developing Software on Scaillx	95
5.7.1	Visual Studio Code Remote.....	95
5.7.1.1	Opening /home/root in Visual Code Studio	95
5.7.2	Modifying device-trees	96
5.7.2.1	Devicetree overlays via Configfs.....	97
5.7.2.2	Devicetree overlays from u-boot.....	97
5.7.2.3	Check current devicetree.....	98
5.7.3	Writing Rust or Go applications	98
5.7.3.1	Rust.....	98
5.7.3.2	Go.....	99
5.7.4	Build loadable module on device	100
5.8	Yocto Embedded Linux Development.....	101
5.8.1	Using the Yocto SDK.....	101
5.8.2	Customising images or adding software	102
5.8.2.1	Modifying the template:.....	102
5.9	Hailo integration.....	103
5.9.1	Verify setup	103
5.9.2	Run examples.....	104
5.9.2.1	Detection Demo	105
5.9.2.2	LPR Demo.....	105
5.9.3	Known issues	106
5.10	Wi-Fi Connectivity	106
5.10.1	Verify wireless device is detected	107
5.10.2	Scan for access points using impala (a convenient Text User-Interface) on SCAILX.....	107
5.10.3	Using Impala:.....	108
5.10.4	Check Wi-Fi status	109
5.10.5	Network configuration.....	109
5.10.6	Further reading.....	109
5.11	Package management.....	110
5.11.1	OPKG usage	110
5.11.1.1	Get fresh package list.....	110
5.11.1.2	Get the list of available packages.....	110
5.11.1.3	Get the list of installed packages.....	111
5.11.1.4	Install package	111

5.11.1.5	Remove package.....	111
5.11.2	Other package Managers	111
6	Troubleshooting	112
6.1	Camera troubleshooting	112
6.1.1	Camera troubleshooting.....	112
6.1.2	Verify the camera is connected and driver loaded	112
6.1.2.1	Print the V4L2 Connection diagram.....	112
6.1.2.2	Check the /dev nodes for devices	113
6.1.3	Check dmesg for camera kernel modules errors	113
6.1.4	Scan I2C bus for connected cameras	113
6.2	SCAILX-2GS234 LED Error codes	114
6.3	SCAILX reset to defaults	114
6.3.1	Reset to factory defaults	114
6.3.2	Change password	115

Aegis Electronic Group
www.aegiselect.com



1 Getting Started with SCAiLX

SCAiLX-SOM-AI modules are pre-programmed processor boards that come from factory with the latest software release.

The SCAiLX-SOM-AI module will automatically boot after power applied to the board.

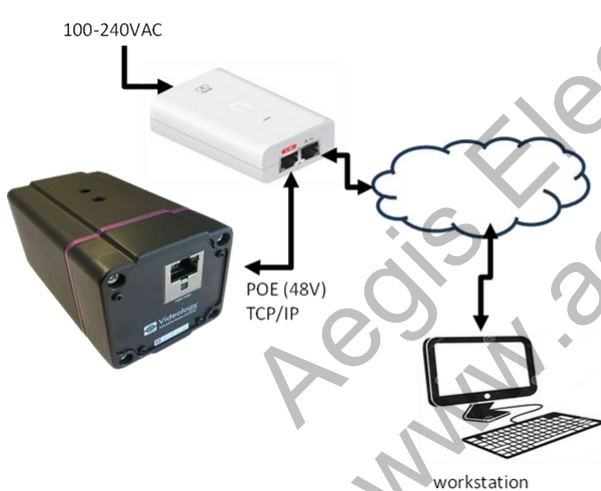
Power can be applied to the board:

1. via SCAiLX-ETH-POE board connected to board connector RGMII, or
2. via SCAiLX-ETH-DC board connected to board connector RGMII.

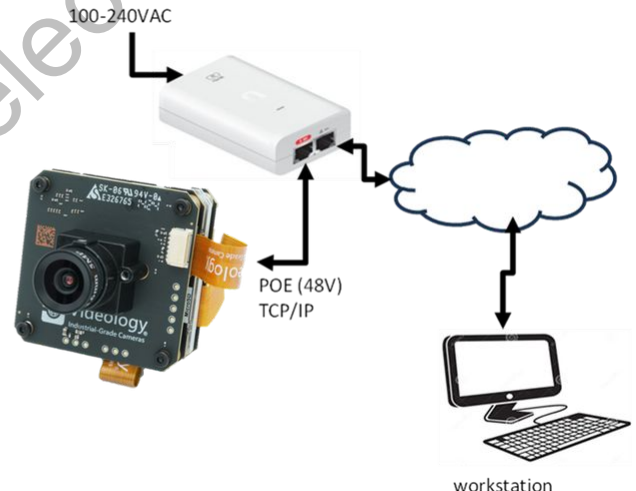
SCAiLX-SOM-AI board can be integrated in:

1. SCAiLX-ZB camera with aluminium housing and PoE interface, or
2. SCAiLX-CAM board stack with SCAiLX-ETH-POE or SCAiLX-ETH_DC power board.

Example of powering the SCAiLX-ZB camera via a PoE-injector connected to a local network with host PC.



Basic setup SCAiLX-ZB with PoE

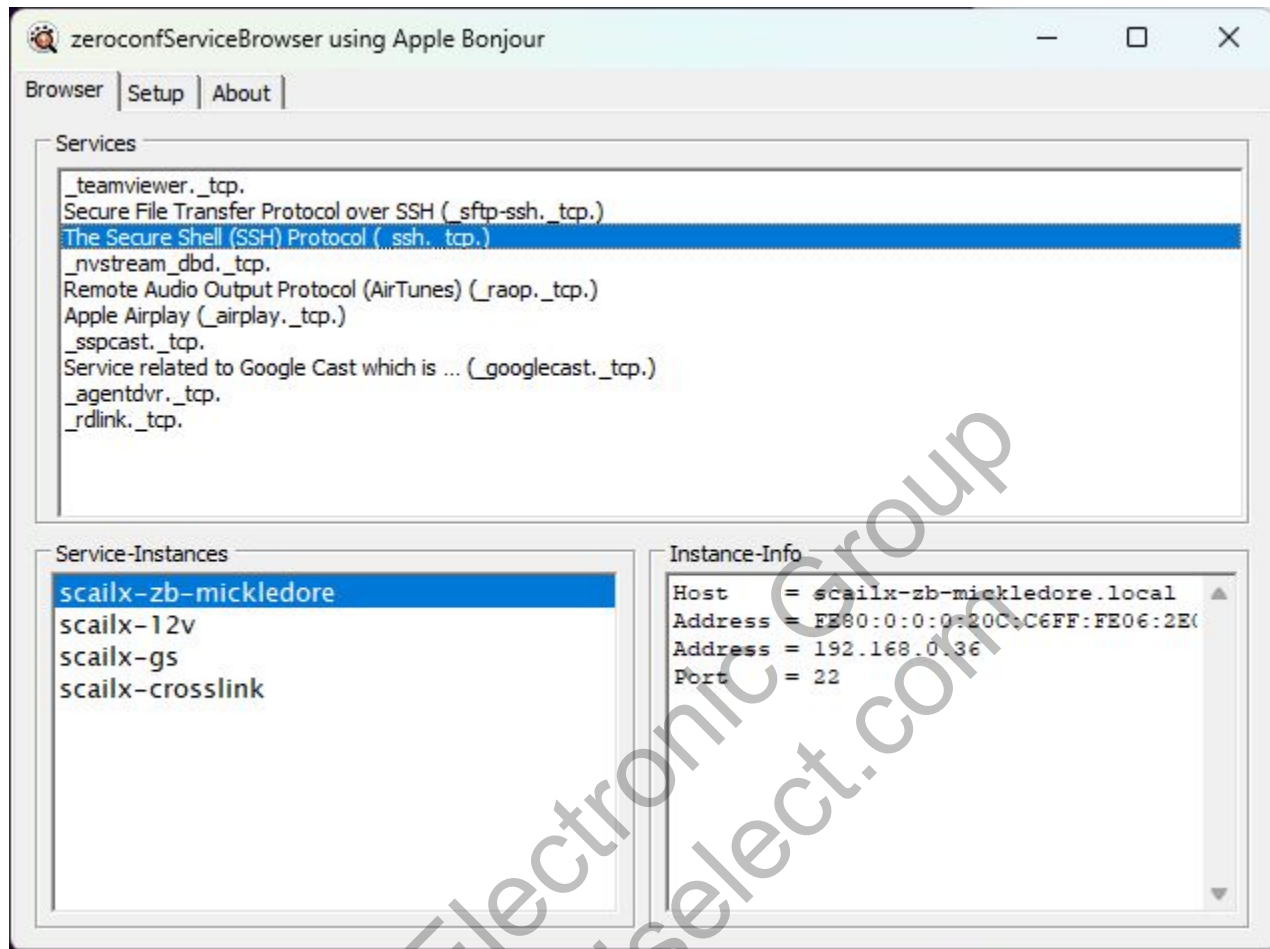


Basic setup SCAiLX-CAM with PoE

1.1 Find the device on the network

SCAiLX devices advertise themselves using Avahi (ZeroConf) as SSH/SFTP-capable on the network.

To scan the network for such devices, on Windows, a good free tool is ZeroconfServiceBrowser.



On Linux, avahi itself can be used:

```
avahi-browse -rt _ssh._tcp
```

The default host-name for scailx-devices is "scailx-ai".

1.2 SSH connect to the SCAiLX device

Once you know the device IP address or its hostname, you can connect to the device via SSH. With the default host-name of **scailx-ai**, the following should work.

```
ssh root@<hostname.local>
```

 The latest builds of Windows 10 and 11 have SSH support integrated. See [Here for more information](#). Note that you might need to enable this feature.

Without the SSH key, the scailx device should request users to create a password upon first-login.

 Some customers have experienced issues connecting as the SSH connection is requesting a password before one could be set. If this is the case, follow the SSH-KEY instructions below.

1.3 SCAiLX SSH key

The Following SCAiLX SSH-key can be used to access any SCAiLX device.

[scailx-dev](#)

[scailx-dev.pub](#)

Copy the 2 files into to C:\Users\<username>\.ssh folder or ~/.ssh in Linux.

 It may be necessary to adjust the permissions of the files in order for OpenSSH to use them:

```
chmod 0600 ~/.ssh/scailx*
```

Add the following to the **config** file in the .ssh folder (create the file if it doesn't exist):

```
Host scailx*
  User root
  IdentityFile ~/.ssh/scailx-dev
  StrictHostKeyChecking no
```

Following this, open a new terminal and try to connect via SSH again. It should not need to ask for a password.

 It may not be obvious which .ssh folder is being used in windows. Run

```
ssh -v root@<hostname.local>
```

to get info about where windows is searching for SSH keys.

 The hostname can be modified in the file `/etc/hostname`

1.4 Video streaming (standard)

SCAILX comes standard with video streaming of the connected cameras (board level or zoom block) via the webRTC implementation.

 `http://scaillx-ai.local:1984`

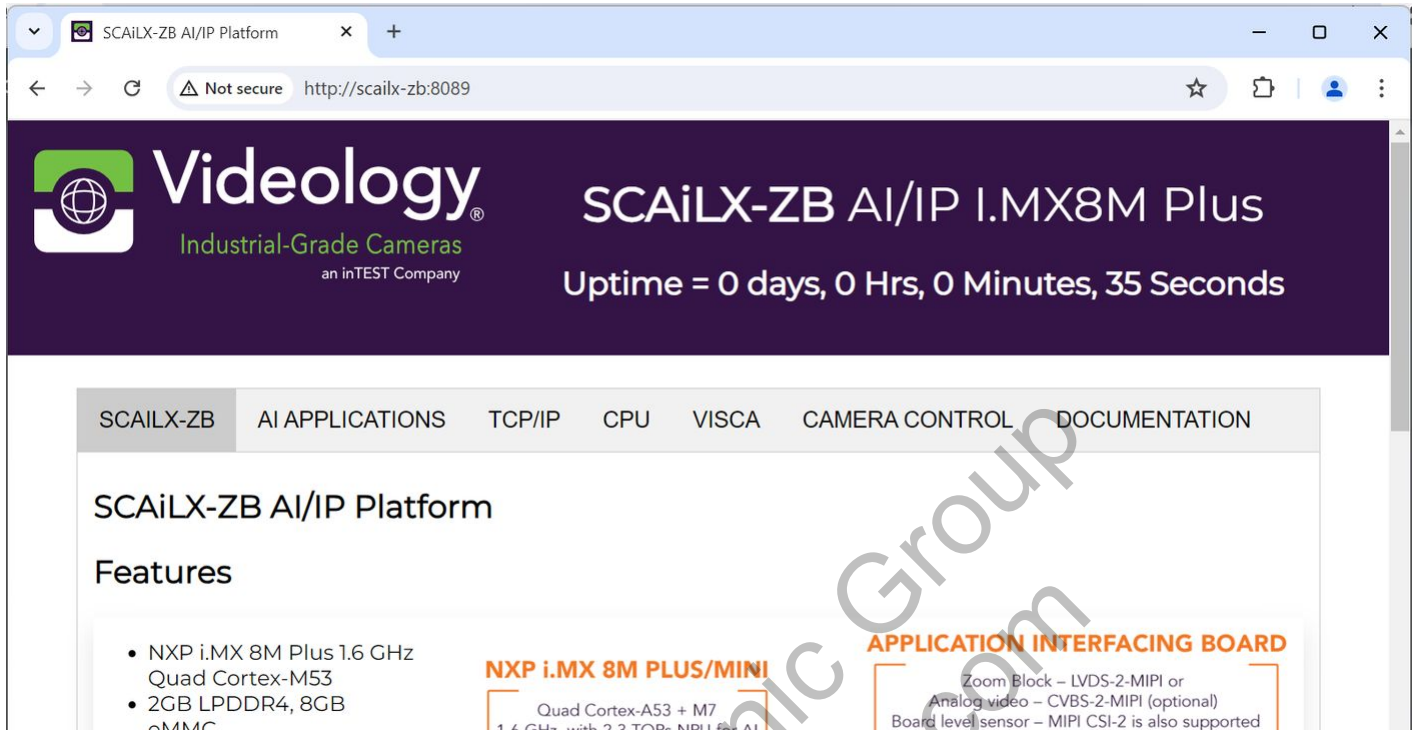
1.5 Web browser (GitHub source)

A web browser can be installed from GitHub page of SCAILX.

https://github.com/VideologyInc/scaillx_python_webgui.git

```
# ssh into the device
ssh root@scaillx-ai.local
git clone https://github.com/VideologyInc/scaillx_python_webgui.git
cd scaillx_python_webgui
python3 setup.py install
systemctl start webservice.service
```

The SCAILX web browser application can be access via port 8089: `http://<hostname>:8089`



The TABS [SCAiLX-ZB], [TCP/IP], [CPU] and [VISCA] are implemented for convenience.

[SCAiLX]	Home page
[TCP/IP]	Graphical information on network load
[CPU]	Graphical information on CPU loads, RAM utilization, Temperature graphs
[VISCA]	Basic VISCA access to zoom block camera.

Example of web browser with LIVE VIEWER tab for in browser streaming of video from video sources.



VISCA/CAM CONTROL tab are only active for zoom block camera using SCAiLX-LVDS-2-MIPI board.

LIVE VIEWER

Live viewer incorporated in web browser page of both camera inputs

In case SCAiLX-SOM-AI is either connected to an a SCAiLX-2GS234-xY camera board or integrated with zoom block camera, the SCAiLX-SOM-AI will automatically detect the video source and install the necessary drivers.

The video of the unit can after boot be accessed on host PC via:

- vlc video player, gstreamer, ...
- pre-installed application go2rtc with webRTC support.

The go2rtc video streaming can be accessed via a web portal page: 1984.

Open in internet browser: `http://<hostname>:1984`, where hostname by default is **scailx-ai**.



scan your network for detecting the unit on your network and obtain hostname and/or ip-address

2 SCAiLX Hardware manual

2.1 SCAiLX Modules

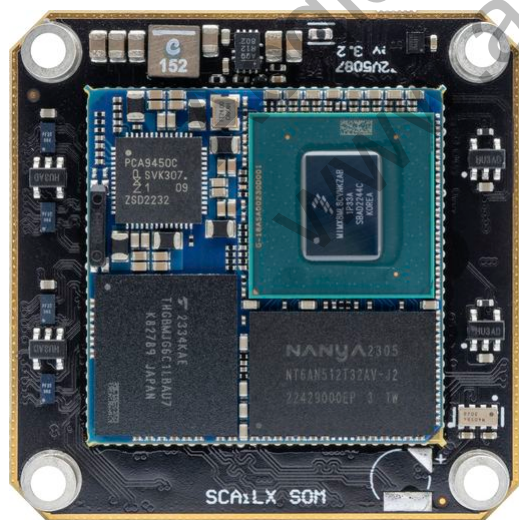
SCAiLX is a modular AI platform based on the NXP IMX8M Plus processor with a board size of 42x42 mm. The SCAiLX-SOM-AI main board can connect to various add-on boards for multiple types of connectivity or sensors.

2.1.1 SCAiLX-SOM-AI board

The SCAiLX-AI-SOM board is a 42x42 mm multilayer PCB board with a high performance IMX8M Plus SoM (System-on-Module) on the front side and 7 board-to-board/board-to-flex connectors on the back side.

Available connectors:

- **RGMII** (Reduced Gigabit Media Interface) for interfacing to 10/100/1000 Mbps interface board SCAiLX-ETH.
- **GPIO-A** (General Purpose Input and Output) multi pin connector with various interface signals for I²C, USB2, USB3, CAN bus, GPIO, reset pin.
- **GPIO-B** (General Purpose Input and Output) multi pin connector with various interface signals for I²C, USB2, USB3, CAN bus and GPIO, reset pin and Debug port (for console access).
- **DSI-OUT** (MIPI Display Serial Interface) for connecting a LCD type display.
- **PCIe** (Peripheral Component Interconnect Express) single lane PCIe bus for connecting different type of high speed interfaces, like AI Accelerator, WiFi and others.
- **CSI0-IN** (MIPI Camera Serial Interface) camera interface with 4 lanes up to maximum of 1.5Gbps, port 0.
- **CSI1-IN** (MIPI Camera Serial Interface) camera interface with 4 lanes up to maximum of 1.5Gbps, port 1.



front side



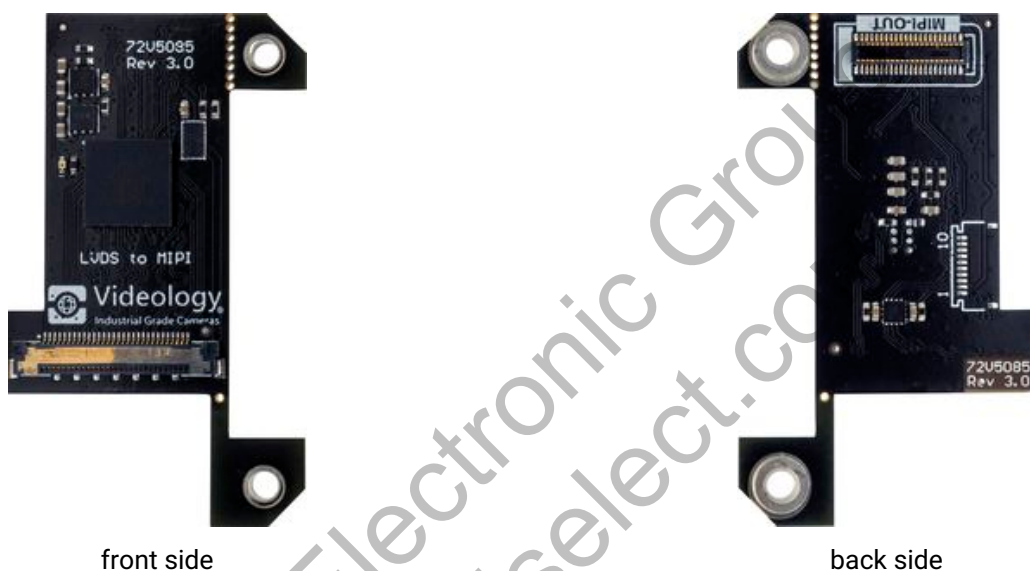
back side

2.1.2 SCAILX-LVDS-2-MIPI board

SCAiLX support LVDS based zoom block cameras via the integration through the SCAILX-LVDS-2-MIPI interface board.

The SCAILX-LVDS-2-MIPI board connects to the LVDS camera through a 30 core micro coaxial KEL cable for camera power (9-12 V DC), eight (8) LVDS signal pairs, an LVDS clock pair, a serial communication port Tx/RX and basic GPIO for camera reset (if applicable).

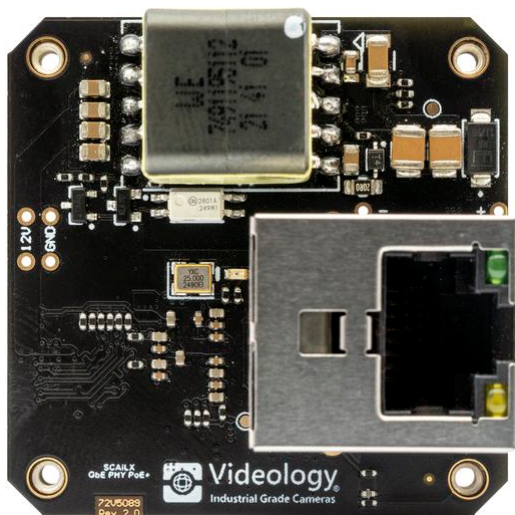
The SCAILX-LVDS-2-MIPI board is placed on the SCAILX-SOM-AI board connecting the MIPI-OUT connector on the CSI1-IN connector of the SOM board.



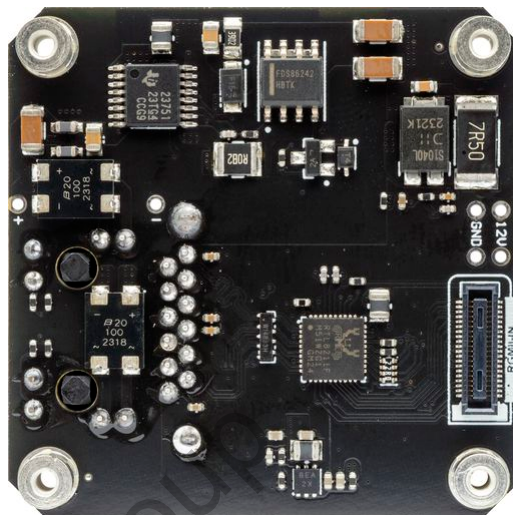
2.1.3 SCAILX-ETH-POE board

The SCAILX-ETH-POE is the SCAILX compatible RGMII interface board for 10/100/1000 Mbps full duplex network connectivity. The PoE (Power-over-Ethernet) is compatible with IEEE802.3af (21 W).

The SCAILX-ETH-POE board interface with a standard RJ45 cable to the local network.



front side



back side

The SCAILX-ETH-POE board interfaces to the SCAILX-SOM-AI board via the SCAILX-FLEX-ETH flexible cable or SCAILX-FLEX-60 for board-to-board connection with mating connectors for errorless connection.



SCAILX-FLEX-60*

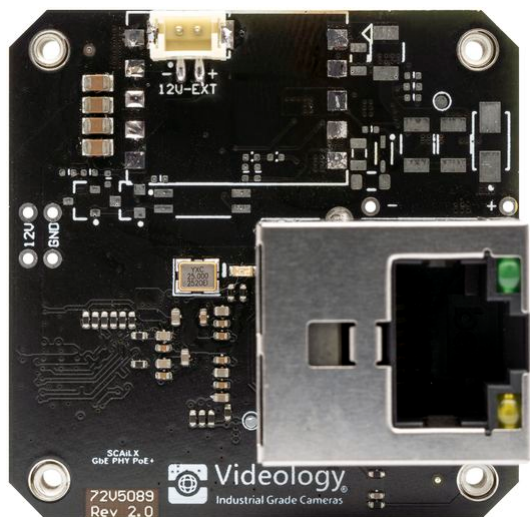


SCAILX-FLEX-ETH*

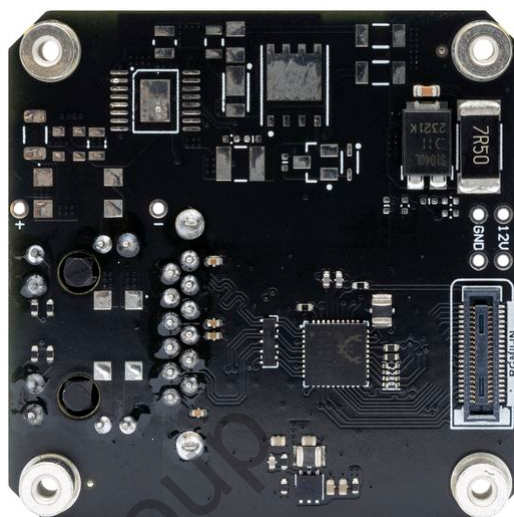
*: depending on assembly

2.1.4 SCAILX-ETH-DC board

The SCAILX-ETH-DC board interface with a standard RJ45 cable to the local network for Ethernet communication, but has a separate 2-pin connector for DC power supply. The SCAILX-ETH-DC board is a strip-down version of the SCAILX-ETH-POE board.



front side



back side

The SCAILX-ETH-DC board interfaces to the SCAILX-SOM-AI board via the SCAILX-FLEX-ETH flexible cable or SCAILX-FLEX-60 for board-to-board connection with mating connectors for errorless connection.



SCAILX-FLEX-60*



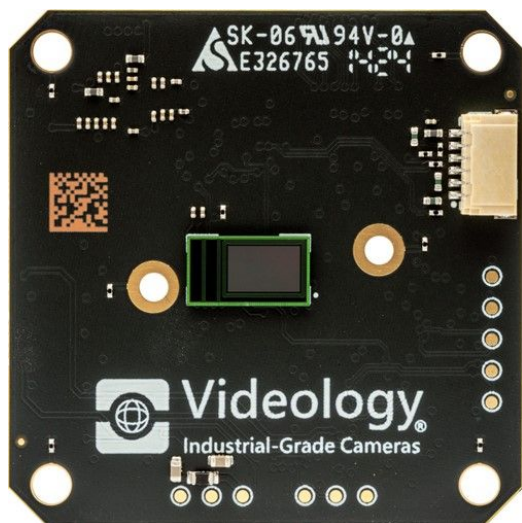
SCAILX-FLEX-ETH*

*: depending on assembly

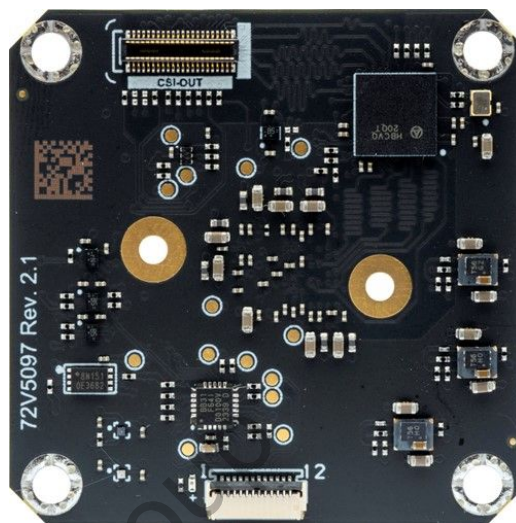
2.1.5 SCAILX-2GS234-xY camera board

The SCAILX-2GS234-xY is a board level camera board with MIPI output. The board has a MIPI-2-MIPI ISP (Image Signal Processor) with full video pipeline with RAW, YUV or compressed video output.

The camera supports multiple video formats, internal/external synchronization and triggering.



front side



back side

The SCAILX-2GS234 board interfaces to the SCAILX-SOM-AI board via the SCAILX-FLEX-xxx flexible cable. Available lengths are 60, 120, 240 or 480 mm.

2.1.6 SCAILX-ACC26 AI Accelerator

SCAILX-ACC26 board incorporates the **HAILO-8 M.2 AI Accelerator** module for up to 26 TOPS (Tera Operations Per Second AI inference operations).

The SCAILX-ACC26 is supported with Object Detection, Depth Estimation and License Plate Recognition.



SCAILX-PCIE + HAILO-8

2.1.7 SCAILX-WIFI

The SCAILX-WIFI board delivers WiFi and Bluetooth communication to SCAILX.

The SCAILX-WIFI board is delivered with the **Intel AX210 Wifi/BT** module.



SCAILX-PCIE + Intel AX210 Wi-Fi/Bluetooth

2.2 SCAILX-SOM-AI

2.2.1 SCAILX-SOM-AI

2.2.1.1 General description

The SCAILX-SOM-AI is based on a dedicated SoM (System-on-Module), which is the central processing unit for video and neural processing, the brains of the SCAILX platform. It has a high performance NXP IMX8M Plus processor, running a variant of NXP Yocto (<https://www.yoctoproject.org/>) Linux customized for SCAiLX product to include Videology designed add-on boards and include applications.

Customer can build on SCAiLX by writing applications on the device as described under [SCAiLX Software manual](#).

2.2.1.2 SCAILX-SOM-AI (rev. 4.0)

v4.0	New feature: INA231 current-shunt and power monitor chip in +12 V power line of SCAILX-SOM-AI module. - Voltage, Current and Power measurement via I²C - Hardware/Software compatible with 3.2	
v3.2	First release	

2.2.1.3 SCAILX-SOM-AI

2.2.1.4 Features



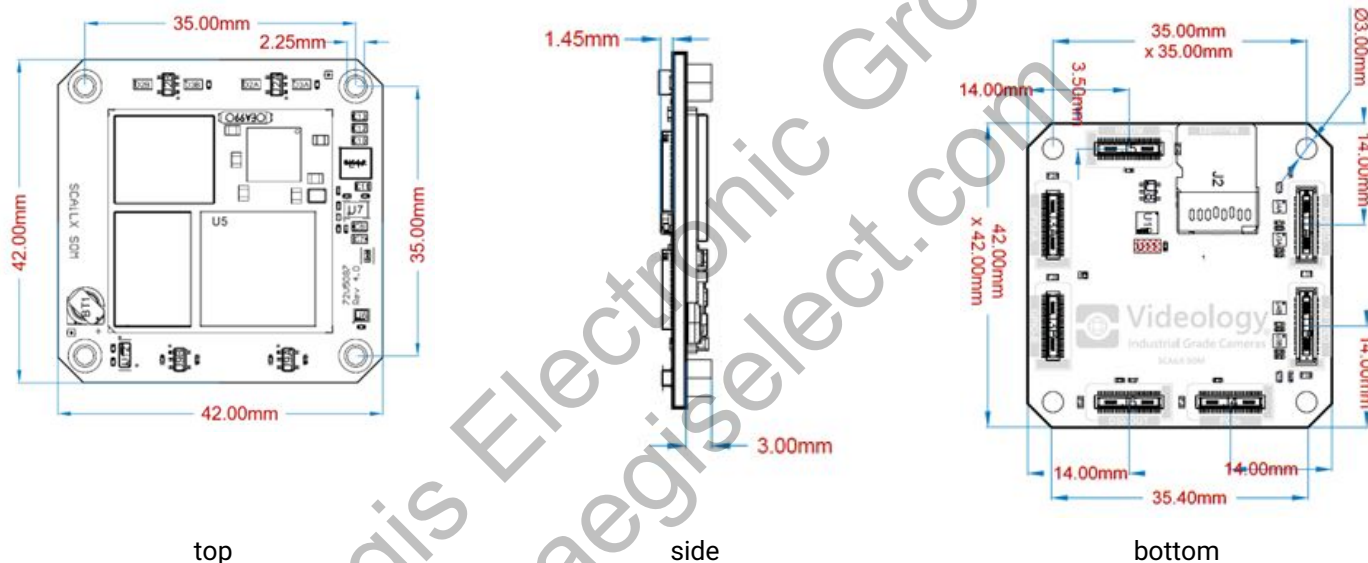
SCAILX-SOM-AI (rev. 3.2/4.0)

FEATURE	DETAILS
CPU	NXP IMX8M Plus 4x Arm® Cortex®-A53 @1.6 GHz, 1x Arm® Cortex®-M7
NPU	1 × NPU 2.3 TOPS
RAM	2 GB LPDDR4
OS Storage	8 GB eMMC
External Storage Support	MicroSD, NVMe
Networking	1x 1000baseT Ethernet
GPU	Vivante GC7000UL
3D GPU Support	OpenGL ES 3.1/3.0, Vulkan, Open CL 1.2 FP, OpenVG 1.1
Video Decoder	1080p@60 max.
Video Encoder	1080p@60 H.264, 1080p@60 H.265 max.
USB	1x USB3.0, 1x USB2.0
I/O Interfaces	2x UART, 2x I2C, 2x SPI, 1x PCIe(Gen 3.0), 1x SD/eMMC, 2x CAN FD
Display Interface	MIPI-DSI
Camera Interface	2x MIPI-CSI2 (4 Lane), 2x ISP

Supply Voltage	6-17 V
I/O Voltage	3.3 V
Power Consumption	Typical <5 W
Temperature range (SoM)	Industrial: -40 °C to 85 °C
Supported OS	Embedded Linux (Linux 6.6.1 or higher), YOCTO development environment
Form Factor	Module size 42 x 42 mm

The SCAiLX SoM is a 42x42 mm PCB with a microSD card holder and 7 board-to-wire 40 pin connectors.

2.2.1.5 Dimensions



Step file

[SCAiLX-SOM-STEP.zip](#)

[SCAiLX - Peripheral board templates](#)

Datasheet

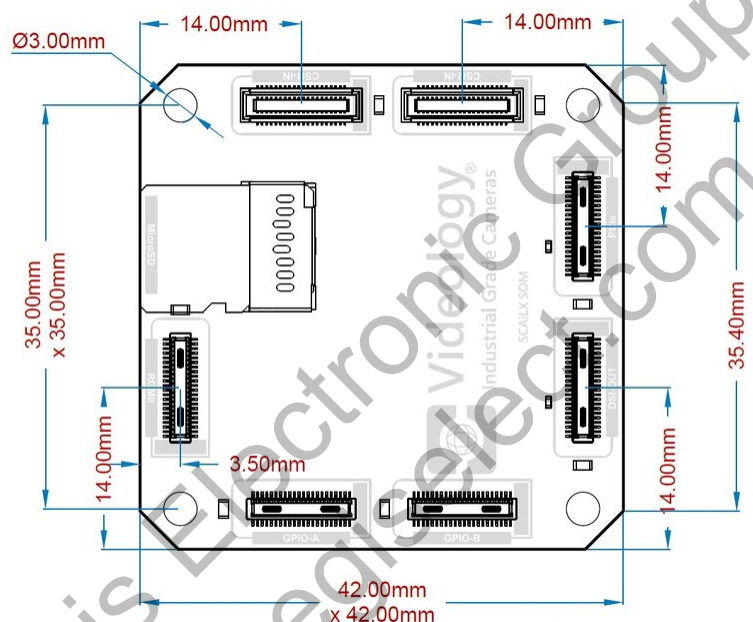
[PDS_SCAiLX-SOM-AI](#)

2.2.2 SCAiLX-SOM board - Connector details

2.2.2.1 SCAiLX-SOM board - Connector details

2.2.2.1.1 SCAiLX-SOM board dimensions and board connectors

The SCAiLX-SOM-AI boards features 7 board-to-wire or board-to-board connectors.



2.2.2.1.2 SCAiLX-SOM board connectors

The connectors available are:

Name	Description
CSI1-IN	4-lane MIPI port 0, I ² C port, camera control signals GPIO, 5 V power out, 12 V power out (fused)
CSI2-IN	4-lane MIPI port 1, I ² C port, camera control signals GPIO, 5 V power out, 12 V power out (fused)
GPIO-A	USB-2 signals, USB-3 signals, I ² C port, SPI port, UART port, 5V power out, 12 V power out (fused)
GPIO-B	USB-2 signals, USB-3 signals, I ² C port, SPI port, UART port, 5V power out, 12 V power out (fused)
PCI-e	1-lane PCIe with control, 12 V power out, 5V power out, I ² C port

Name	Description
DSI-OUT	4-lane MIPI DSI, 12 V power out fused, 5V power out, I ² C port, Display Enable, Display Back Light control
RGMII	RGMII TX and RX signals, TX and RX RGMII clocks, 12 V power IN fused, 5 V power out

2.2.2.1.3 SCAILX-SOM-AI board interface types

The 40 pin board-to-wire connectors have multiple ports. In the below table the different port types per connector are listed. The detailed description of each port and signal naming can be found in the preceding chapter.

Name	MIPI	I2C	UART	SPI	CAN	USB-2	USB-3	GPIO
CSI1-IN	IN	1	-	-	-	-	-	5
CSI2-IN	IN	1	-	-	-	-	-	5
GPIO-A	-	1	1	1	1	1	1	2
GPIO-B	-	1	1	1	1	1	1	2
PCIe	-	1	-	-	-	-	-	
DSI-OUT	OUT	1	-	-	-	-	-	2
RGMII	-	-	-	-	-	-	-	-

2.2.2.1.4 Pinout of board connectors

Pinout of board connectors

CSI1-IN(x=0) and CSI2-IN(x=1) connector pinout

Pin	Name	Type	Pin	Name	Type
1	MIPIx_CLK_N	I	2	GND	POWER
3	MIPIx_CLK_P	I	4	CAM FLASHx	O
5	GND	POWER	6	CAM TRIGx	I
7	MIPIx_D0_N	I	8	GND	POWER
9	MIPIx_D0_N	I	10	#CAM_NRESETx	O
11	GND	POWER	12	CAM ALERTx	I

Pin	Name	Type	Pin	Name	Type
13	MIPIx_D1_N	I	14	GND	POWER
15	MIPIx_D1_P	I	16	CAM PWR ENx	O
17	5V CAM	POWER OUT	18	5V CAM	POWER OUT
19	12V CAM	POWER OUT	20	12V CAM	POWER OUT
21	12V CAM	POWER OUT	22	12V CAM	POWER OUT
23	5V CAM	POWER OUT	24	5V CAM	POWER OUT
25	VDD_IO_CAMx	POWER OUT	26	CAMx_SCL	O
27	GND	POWER	28	CAMx_SDA	I/O
29	MIPIx_D2_N	I	30	GND	POWER
31	MIPIx_D2_P	I	32	RESERVED 1	
33	GND	POWER	34	RESERVED 2	
35	MIPIx_D3_N	I	36	GND	POWER
37	MIPIx_D3_P	I	38	RESERVED 3	
39	GND	POWER	40	RESERVED 4	

All I/O 3V3.

Connector: **DF40C(2.0)-40DS-0.4V(51)F40C**

GPIO-A and GPIO-B connector pinout

All signals on GPIO-A and GPIO-B are specific (except RESET and DBG EN which are shared on both connectors).
 DEBUG PORT (DBG TX/DBG RX) only on GPIO-B.

Pin	Name	Type	Pin	Name	Type
1	USB_D_P	I/O	2	GND	POWER
3	USB_D_N	I/O	4	I2C_SCL	O
5	GND	POWER	6	I2C_SDA	I/O
7	USB_RX_P	I/O	8	GND	POWER
9	USB_RX_N	I/O	10	UART_RXD	I
11	GND	POWER	12	UART_TXD	O
13	USB_TX_P	I/O	14	GND	POWER
15	USB_TX_N	I/O	16	DBG_EN	O

Pin	Name	Type	Pin	Name	Type
17	5V GPIO	POWER OUT	18	5V GPIO	POWER OUT
19	12V GPIO	POWER OUT	20	12V GPIO	POWER OUT
21	12V GPIO	POWER OUT	22	12V GPIO	POWER OUT
23	5V GPIO	POWER OUT	24	5V GPIO	POWER OUT
25	RESET	O	26	CAN_RX	O
27	GND	POWER	28	CAN_TX	I/O
29	SCLK	O	30	GND	POWER
31	MOSI	O	32	GPIO 1	I/O
33	GND	POWER	34	GPIO 2	I/O
35	MISO	I	36	GND	POWER
37	SSO	O	38	DBG RX*	I
39	GND	POWER	40	DBG TX*	O

All I/O 3V3.

Connector: **DF40C(2.0)-40DP-0.4V(51)F40C**

PCIe connector pinout

Pin	Name	Type	Pin	Name	Type
1	PCIE_REF_P		2	GND	POWER
3	PCI_REF_N		4	I2C SCL	O
5	GND	POWER	6	I2C SDA	I/O
7	PCIE_RXN_P		8	GND	POWER
9	PCIE_RXN_N		10		
11	GND	POWER	12		
13	PCIE_TXN_P		14	GND	POWER
15	PCIE_TXN_N		16		
17	5V PCIE	POWER OUT	18	5V PCIE	POWER OUT
19	12V PCIE	POWER OUT	20	12V PCIE	POWER OUT
21	12V PCIE	POWER OUT	22	12V PCIE	POWER OUT
23	5V PCIE	POWER OUT	24	5V PCIE	POWER OUT

Pin	Name	Type	Pin	Name	Type
25			26		
27	GND	POWER	28		
29	PERST		30	GND	POWER
31	WAKE		32		
33	GND	POWER	34		
35	W_DISABLE		36	GND	POWER
37	PCIE_CLK_REQ		38		
39	GND	POWER	40		

All I/O 3V3.

Connector: **DF40C(2.0)-40DP-0.4V(51)F40C**

DSI-OUT 4-lane MIPI DSI connector pinout

Pin	Name	Type	Pin	Name	Type
1	MIPI_DSI_D3_P	O	2	GND	POWER
3	MIPI_DSI_D3_N	O	4	I2C_SCL	O
5	GND	POWER	6	I2C_SDA	I/O
7	MIPI_DSI_D2_P	O	8	GND	POWER
9	MIPI_DSI_D2_N	O	10	DISP_EN	O
11	GND	POWER	12	DISP_BL	O
13	MIPI_DSI_D1_P	O	14	GND	POWER
15	MIPI_DSI_D1_N	O	16		
17	5V DSI	POWER OUT	18	5V DSI	POWER OUT
19	12V DSI	POWER OUT	20	12V DSI	POWER OUT
21	12V DSI	POWER OUT	22	12V DSI	POWER OUT
23	5V DSI	POWER OUT	24	5V DSI	POWER OUT
25			26		
27	GND	POWER	28		
29	MIPI_DSI_D0_P	O	30	GND	POWER
31	MIPI_DSI_D0_N	O	32		

Pin	Name	Type	Pin	Name	Type
33	GND	POWER	34		
35	MIPI_DSI_CLK_P	O	36	GND	POWER
37	MIPI_DSI_CLK_N	O	38		
39	GND	POWER	40		

All I/O 3V3.

Connector: **DF40C(2.0)-40DP-0.4V(51)F40C**

RGMII connector pinout

Pin	Name	Type	Pin	Name	Type
1	ENET_RX_CTL		2	GND	POWER
3	ENET_RXC		4	I2C SCL	O
5	GND	POWER	6	I2C SDA	I/O
7	ENET_RD0	I	8	GND	POWER
9	ENET_RD1	I	10	ENET_INT	
11	GND	POWER	12	ENET_CLK125	
13	ENET_RD2	I	14	GND	POWER
15	ENET_RD3	I	16	ENET_RST	
17	5V RGMII	POWER OUT	18	5V RGMII	POWER OUT
19	12V RGMII	POWER IN	20	12V RGMII	POWER IN
21	12V RGMII	POWER IN	22	12V RGMII	POWER IN
23	5V RGMII	POWER OUT	24	5V RGMII	POWER OUT
25			26	ENET_TX_CTL	
27	GND	POWER	28	ENET_TXC	O
29			30	GND	POWER
31			32	ENET_TD3	O
33	GND	POWER	34	ENET_TD2	O
35			36	GND	POWER
37			38	ENET_TD1	O
39	GND	POWER	40	ENET_TD0	O

All I/O 3V3.

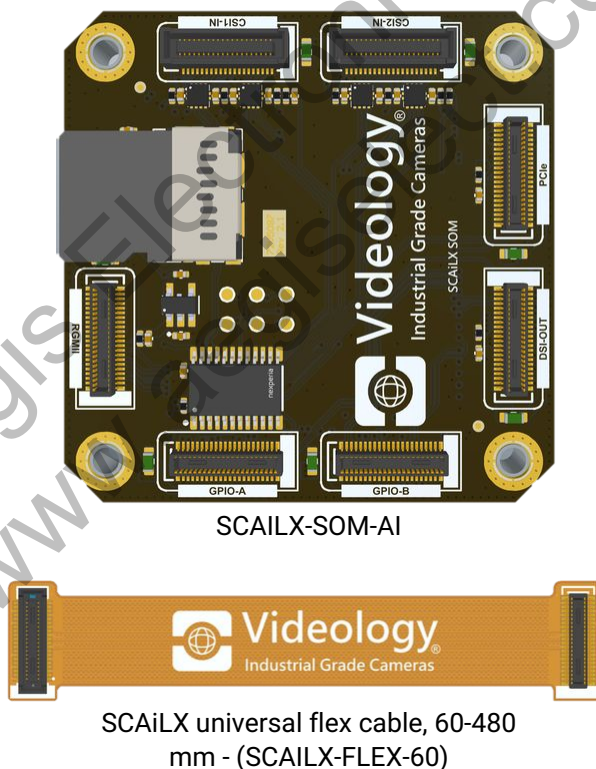
Connector: **DF40C(2.0)-40DP-0.4V(51)F40C**

Typical connection between SCAiLX boards:

From	Port	To	Port	Connection
SCAILX-SOM-AI	RGMII (DP)	SCAILX-ETH-POE	RGMII (DS)	SCAILX-FLEX-ETH
	CSI0-IN/CSI1-IN (DS)	SCAILX-2GS234	CSI-OUT (DP)	SCAILX-FLEX-xxx

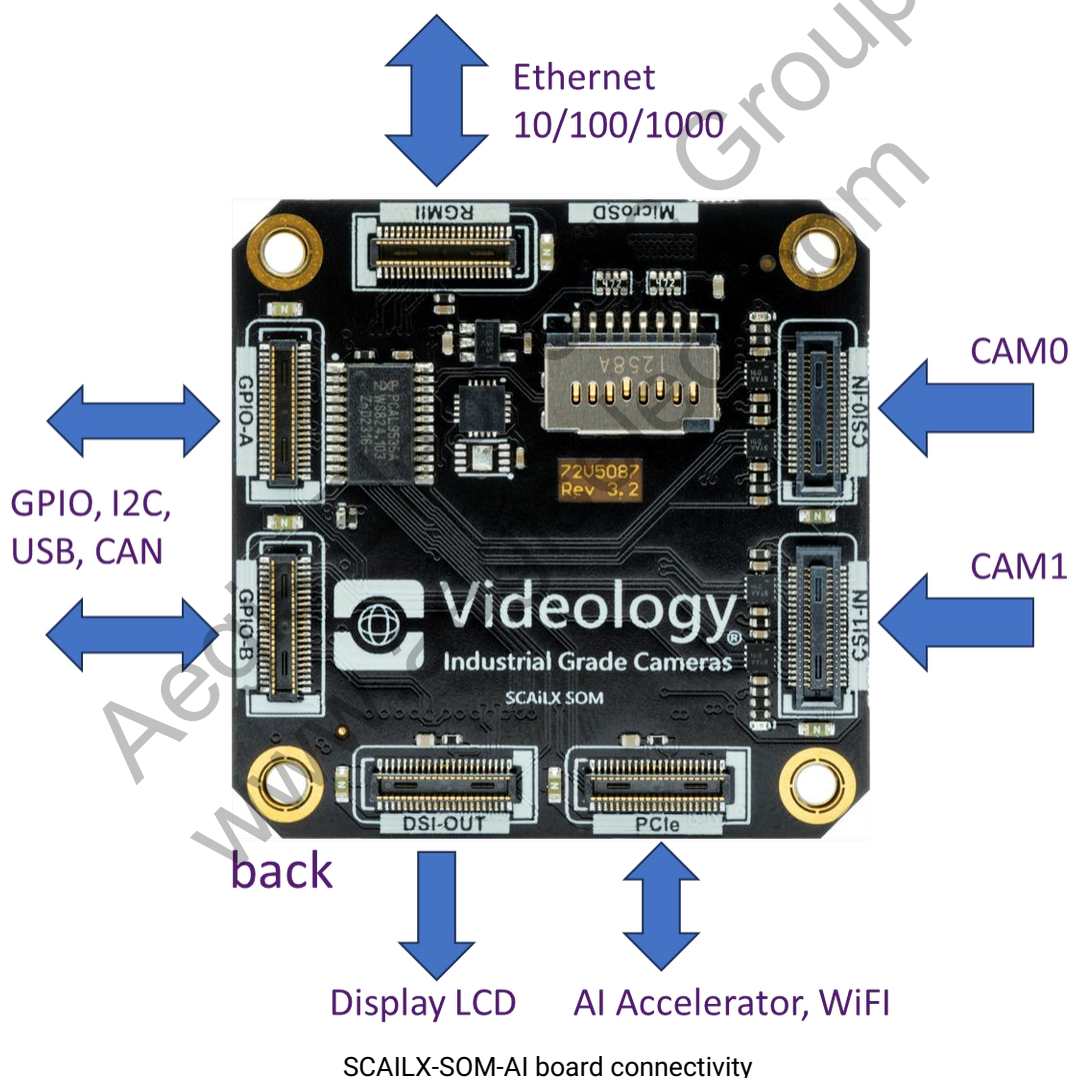
2.2.3 SCAiLX-SOM board - Interfaces description

The SCAiLX platform was designed with **modularity** in mind. This allows Videology or customers to make customized peripherals and camera modules to work with it. The processor board uses a standardized connector interface to connect to daughter-boards or cameras directly, or using flex cables.



⚠ Take note of the **orientation of the connectors**. The connector interface is rotation tolerant (plugging a cable in backwards shouldn't destroy anything) but interfaces obviously won't work. The flex cables have a **rounded side** and a **square side**, and that needs to line up with the white outline on the board. The **white text label** also indicates to side to which the flex cable will run.

⚠ B2B connectors are not rated for multiple plug-unplug cycles. Be careful when unplugging to to damage the connectors or the flex cables.



SCAiLX-SOM-AI board connectivity

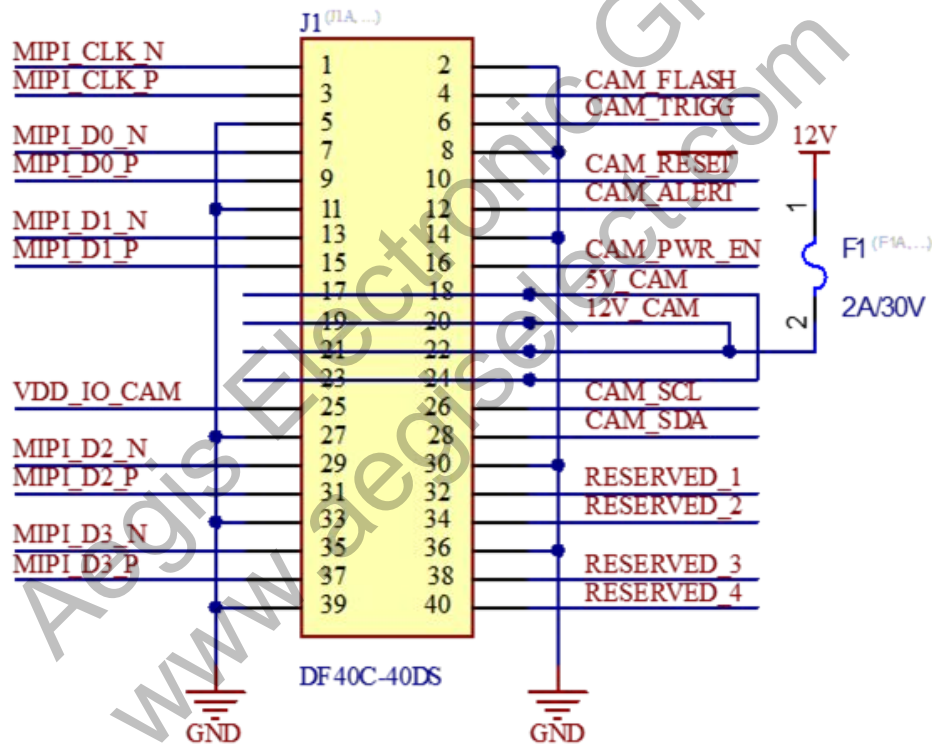
2.2.3.1 Board connectors

2.2.3.1.1 Camera MIPI CSI

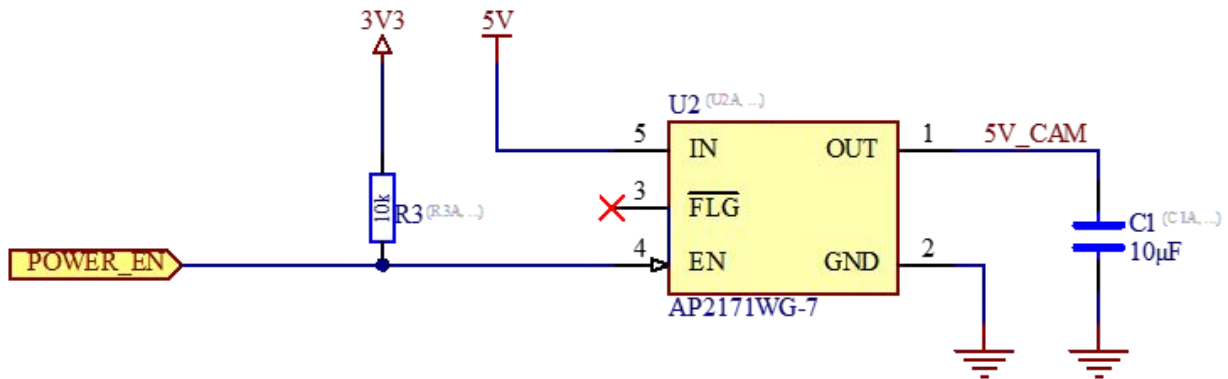
There are 2 camera interface connectors. Each camera interface consists of a 4-lane MIPI-CSI input, together with 5 V and 12 V for powering cameras, an I²C interface and GPIO's.

- MIPI-CSI2, 4 lanes + clock lane
- I²C bus
- Trigger input and strobe output.
- GPIO

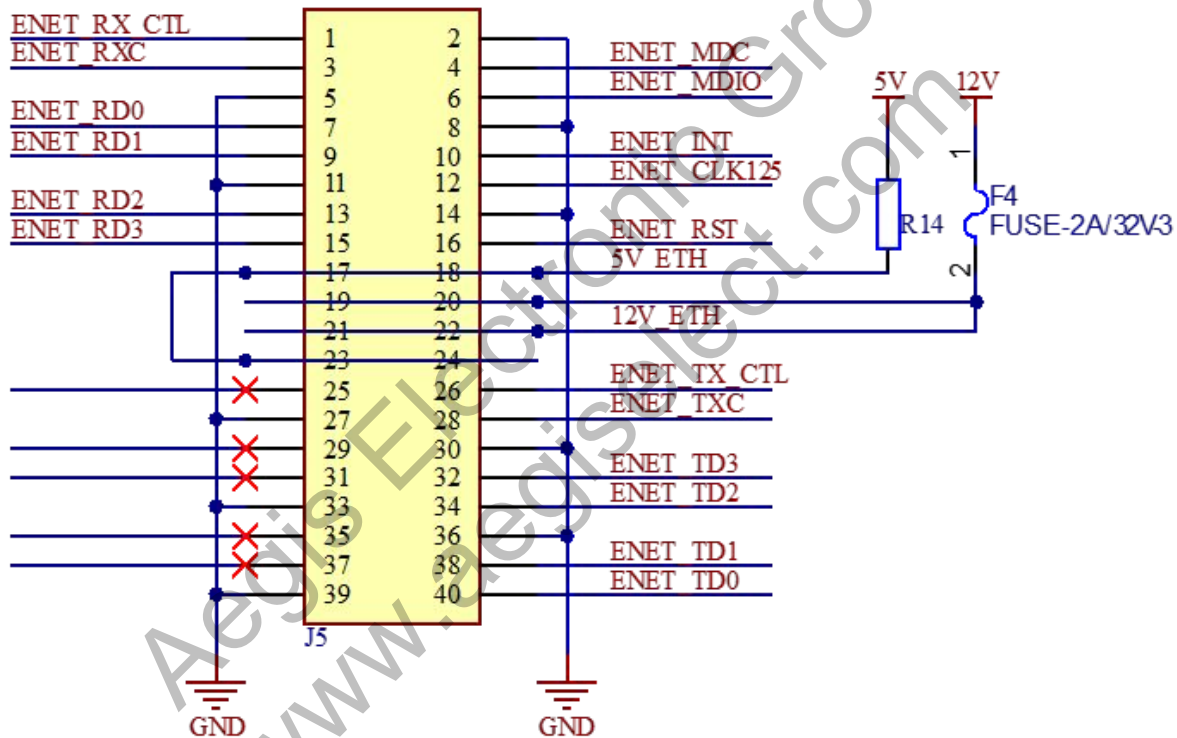
The processor board includes a **voltage-level-translator** for the GPIO's and I²C signals to support a range of camera IO voltages (1.65-5.5 V). The Camera in turn provides VDD-IO in pin 25.



The **5V** rail provided to the cameras is **switchable** and **current limited** to turn cameras off from the processor.



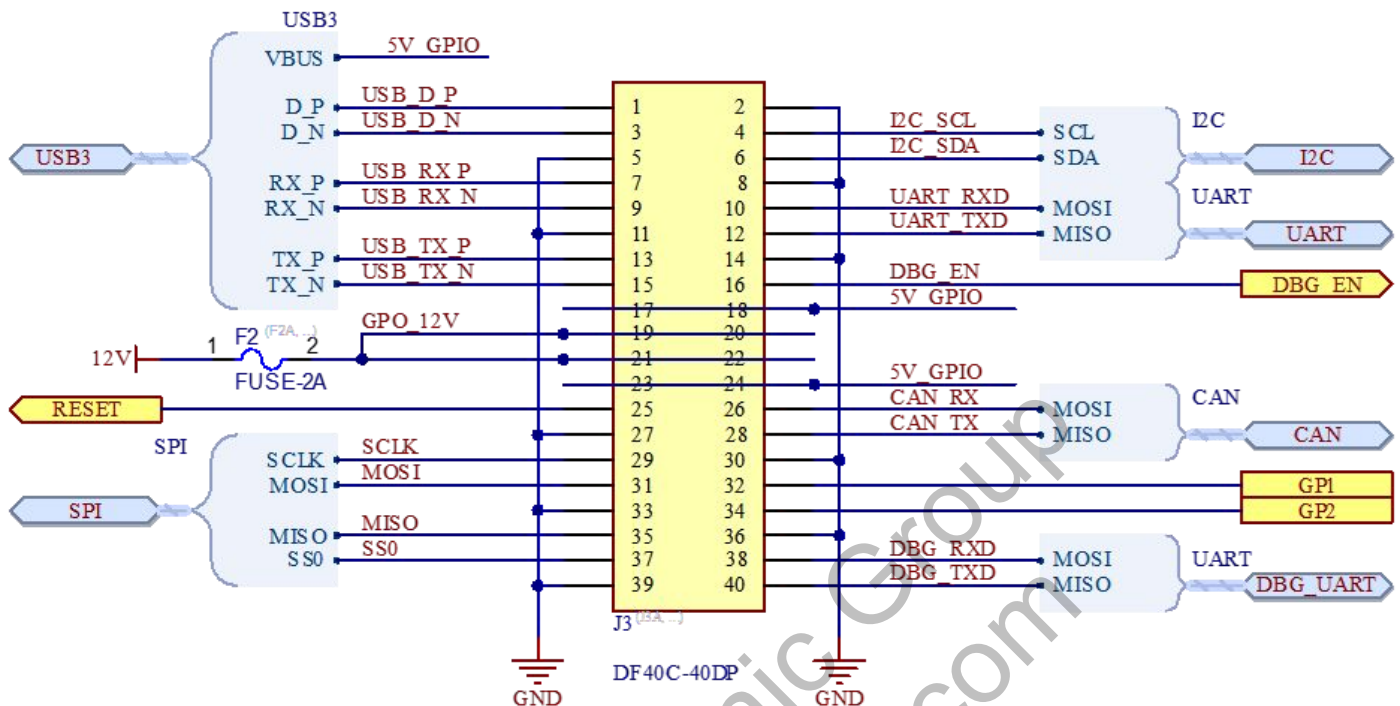
2.2.3.1.2 Ethernet RGMII



- RGMII interface with I²C and MDIO. Requires external ethernet PHY, but can allow other network interfaces such as 10BaseT1L or fiber.
- 12 V is used to power the processor board in the case of the POE Ethernet adapter.

2.2.3.1.3 GPIO - General Purpose IO

Two (2) GPIO connectors are provided for peripheral devices and debugging. The USB interface is also exposed on these connectors.



Note that the full USB 3.0 (SuperSpeed: 5 Gbps) interface is only available on the GPIO-A interface. The USB on GPIO-B is a USB 2.0(Hi-Speed: 480 Mbps) interface.

As with the cameras, the **5 V is a switched output**, while the 12 V is just fused.

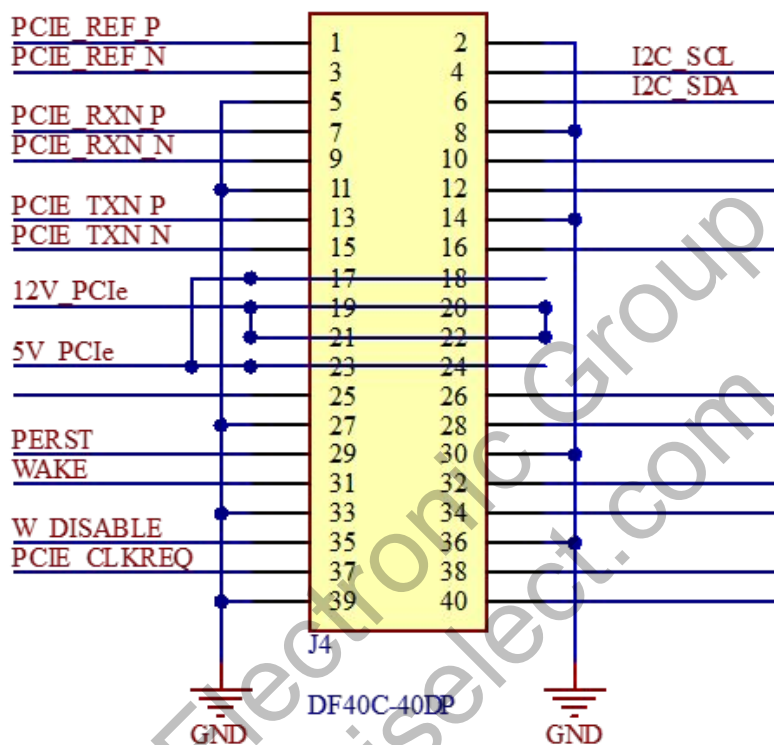
The GPIOs on these are **3.3 V level**.

- UART (DBG interface is a serial console for the processor, while the UART on pins 8/10 is for customer/ application use.
- SPI bus
- I²C bus
- CAN bus
- USB
- USB-C

The GPIO interface is also used for software loading at the factory. Pulling the **DBG_EN** line to 3.3 V and pulling the **RESET** line low briefly will allow the device to boot into DFU mode, for use with NXP's UUU firmware loading utility to load images via USB, but an **OTA software Update** mechanism is provided so customers do not need this. See software Architecture for details.

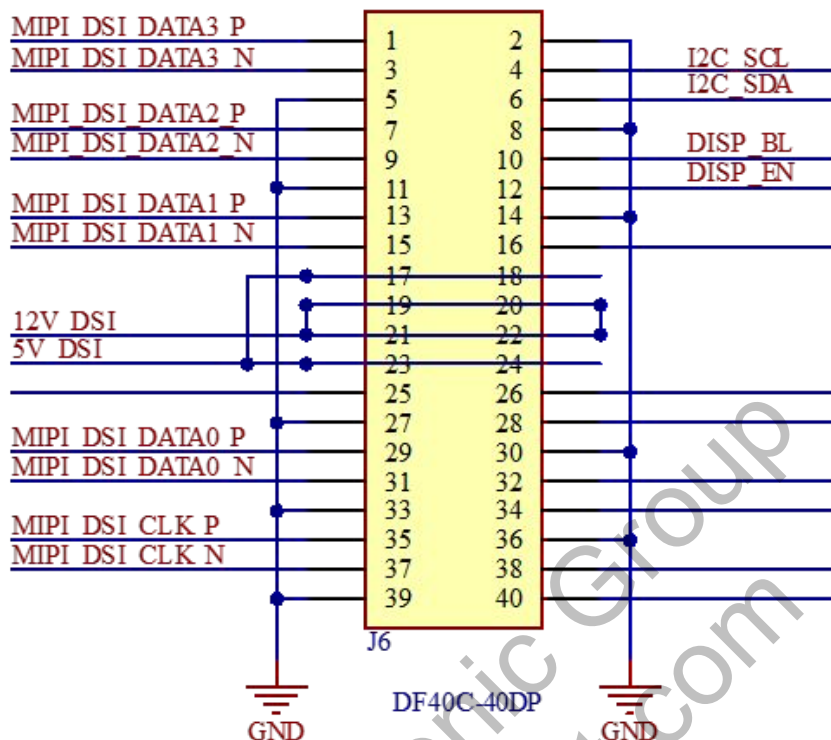
2.2.3.1.4 PCIe - PCI Express

A 1-lane Gen.3 PCIe interface is provided for peripherals such as Wi-Fi, 10G ethernet, NPU Accelerator cards, etc.



2.2.3.1.5 Display MIPI-DSI

A MIPI-DSI output is provided for use with an LCD panel or an external HDMI converter. The output is capable of 1920x1200 at 60 fps output.



2.2.3.2 Datasheet

[PDS_SCAiLX-SOM-AI](#)

2.3 SCAiLX-LVDS-2-MIPI

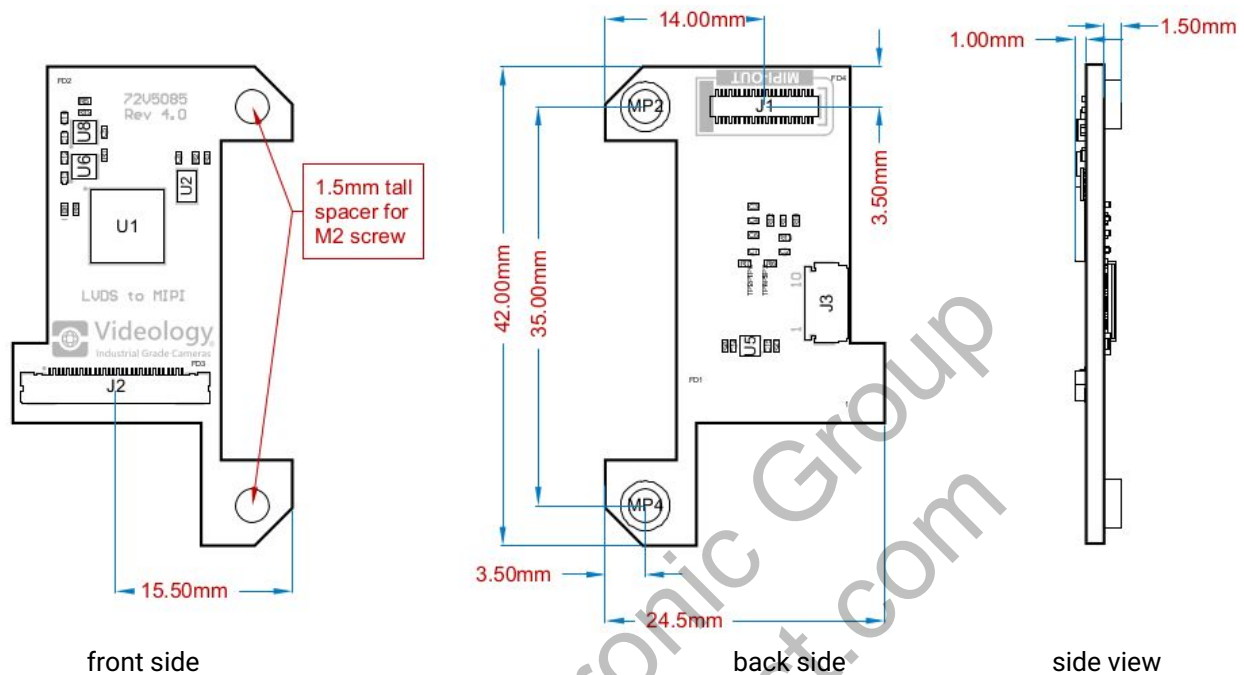
2.3.1 SCAiLX-LVDS-2-MIPI (rev. 4.0)

The LVDS-2-MIPI interface board is used to interface between the SCAiLX SOM and [Videology LVDS Zoom-block cameras](#), as well as 2 MP LVDS Zoom blocks from SONY and TAMRON. The video resolution of the MIPI ports on the SCAiLX-SOM-AI board is max. 1920x1080p at 60 fps.

The SCAiLX-LVDS-2-MIPI boards converts the single or double channel LVDS signals coming from the zoom block camera into a 4-lane MIPI signal. The MIPI signal is fed into the SCAiLX-SOM-AI module through the MIPI CSI port connector (J1), with additional power and I²C bus and camera control signals (GPIO).

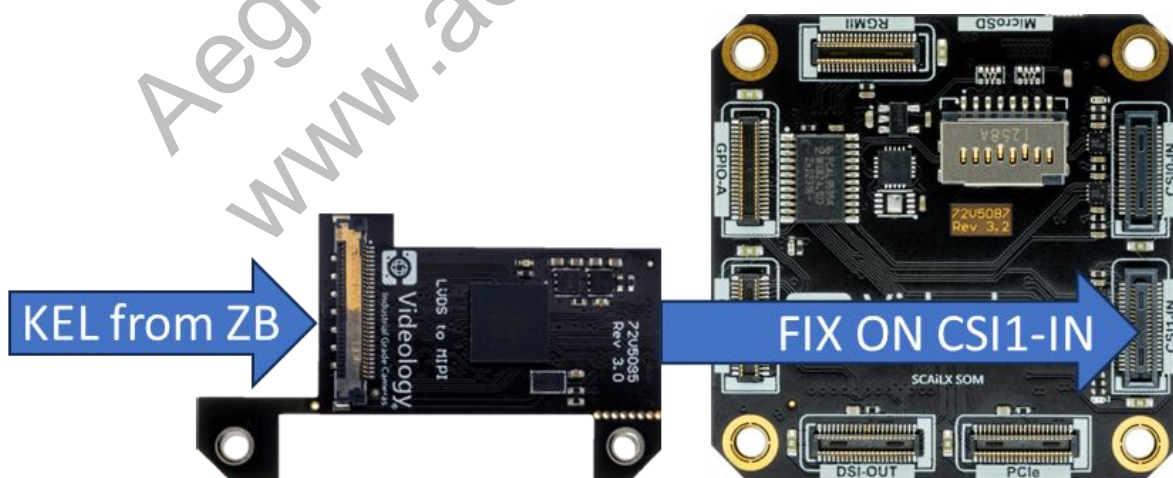
SCAiLX-LVDS-2-MIPI	Type	SCAiLX-SOM-AI board
J1	MIPI CSI port	CSI2 port
J2	KEL zoom block interface	-

2.3.1.1 Dimensions



2.3.1.2 Connection

The SCAiLX-LVDS-2-MIPI board is placed on the back of the SCAiLX-SOM-AI board, where the MIPI-OUT connector of the SCAiLX-LVDS-2-MIPI is connected to the CSI1-IN connector on the SCAiLX-SOM-AI board (as depicted below). The zoom block camera connects to the SCAiLX-LVDS-2-MIPI board via a KEL cable (pin 1 to pin 30 configuration). The LVDS zoom block camera is connected to connector J2 (KEL).



2.3.1.3 KEL connector pinout

Compatibility: LVDS camera with single and/or double LVDS based on THC63LVD827 or equivalent.

Kel Connector Pinout

Pin	KEL connector 30 pin
1	RX_LVDS_D4-
2	RX_LVDS_D4+
3	RX_LVDS_D5-
4	RX_LVDS_D5+
5	
6	
7	RX_LVDS_D6-
8	RX_LVDS_D6+
9	RX_LVDS_D7-
10	RX_LVDS_D7+
11	GND
12	GND
13	12V IN
14	12V IN
15	12V IN
16	12V IN
17	12V IN
18	RXD
19	TXD
20	GND
21	RX_LVDS_D0-
22	RX_LVDS_D0+
23	RX_LVDS_D1-
24	RX_LVDS_D1+
25	RX_LVDS_D2-

Pin	KEL connector 30 pin
26	RX_LVDS_D2+
27	RX_LVDS_CLK-
28	RX_LVDS_CLK+
29	RX_LVDS_D3-
30	RX_LVDS_D3+

2.3.2 Programming and setup

The FPGA configuration is automatically loaded to the SCAILX-LVDS-2-MIPI board at startup of the SCAILX-SOM-AI on detection of the presence of the SCAILX-LVDS-2-MIPI board. The MIPI video port is automatically initiated by an associated driver for the SCAILX-LVDS-2-MIPI board.

2.3.3 Datasheet

[PDS_SCAILX-LVDS-2-MIPI](#)

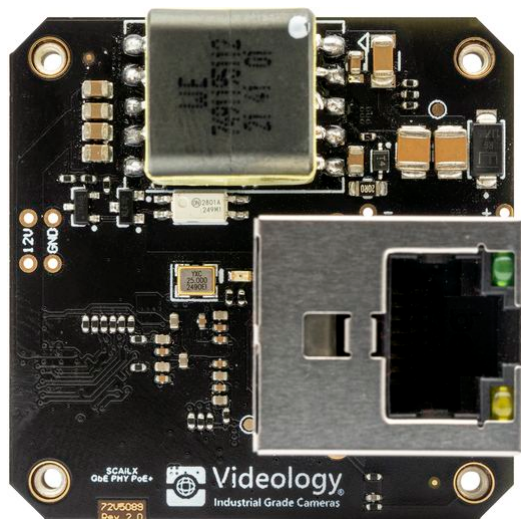
2.3.4 Step 3D model

[STEP_Crosslink_LVDS_to_MIPI.step](#)

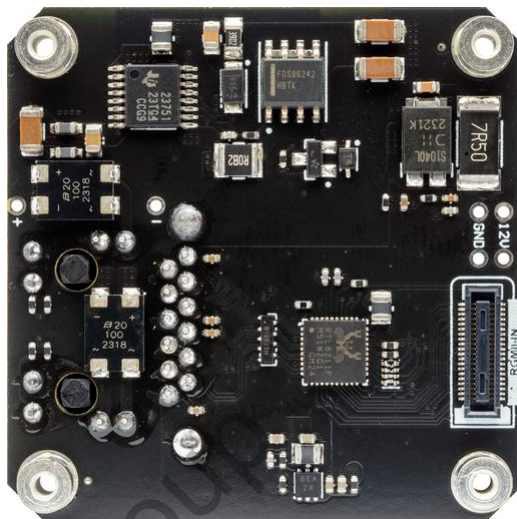
2.4 SCAILX-ETH-POE

2.4.1 SCAILX-ETH-POE board

The SCAILX-ETH-POE is the SCAILX compatible RGMII interface board for 10/100/1000 Mbps full duplex network connectivity. The PoE (Power-over-Ethernet) is compatible with IEEE802.3af (21 W).

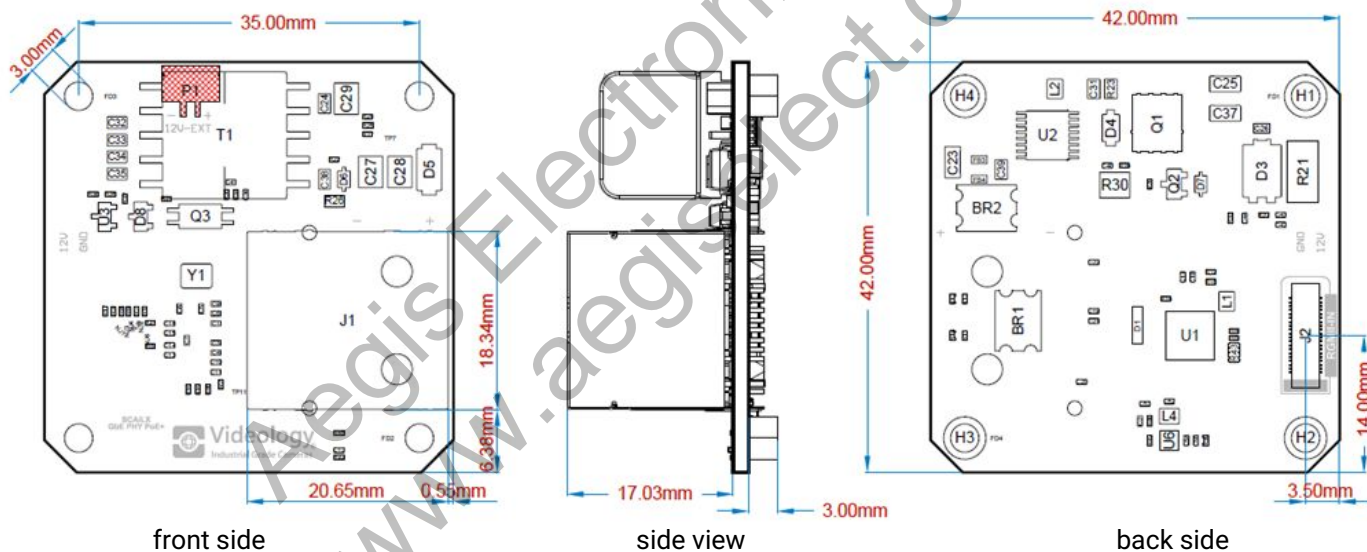


front side



back side

2.4.1.1 Dimensions



2.4.1.2 Connection

The SCAiLX-ETH-POE board is connected to the RGMII port on the SCAiLX-SOM board via a flex PCB connector with 40 pin board-2-board connector.



The SCAILX-FLEX-ETH is connected between RGMII port on SCAILX-ETH-POE and RGMII on SCAILX-SOM-AI.

SCAILX-ETH-POE	Type	SCAILX-SOM-AI board
J2	RGMII	RGMII port

2.4.1.3 Programming and setup

The SCAILX-ETH-POE board needs no configuration or setup.

2.4.1.4 Datasheet

[PDS_SCAILX-ETH-POE](#)

2.4.1.5 Step 3D model

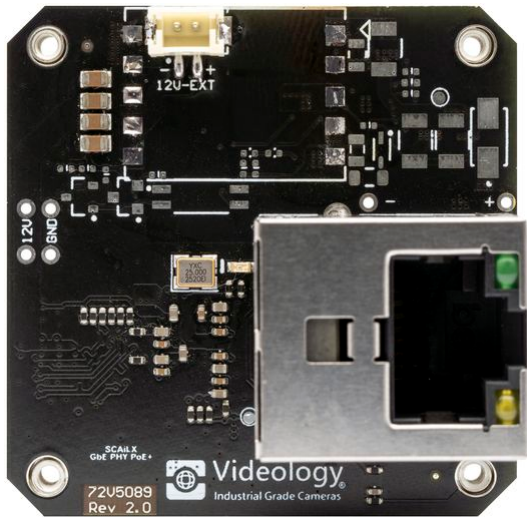
[POE_PHY_Board_r3.step](#)

2.5 SCAILX-ETH-DC

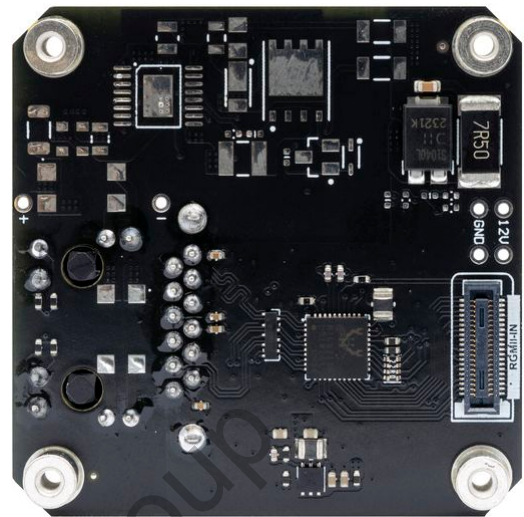
2.5.1 SCAILX-ETH-DC board

The SCAILX-ETH-DC is the SCAILX compatible RGMII interface board for 10/100/1000 Mbps full duplex network connectivity. The SCAILX-ETH-DC is based on the SCAILX-ETH-POE board without the PoE circuitry.

The 12 V DC power is connected to a 2-pin connector on the front side indicated by “12V-EXT”.

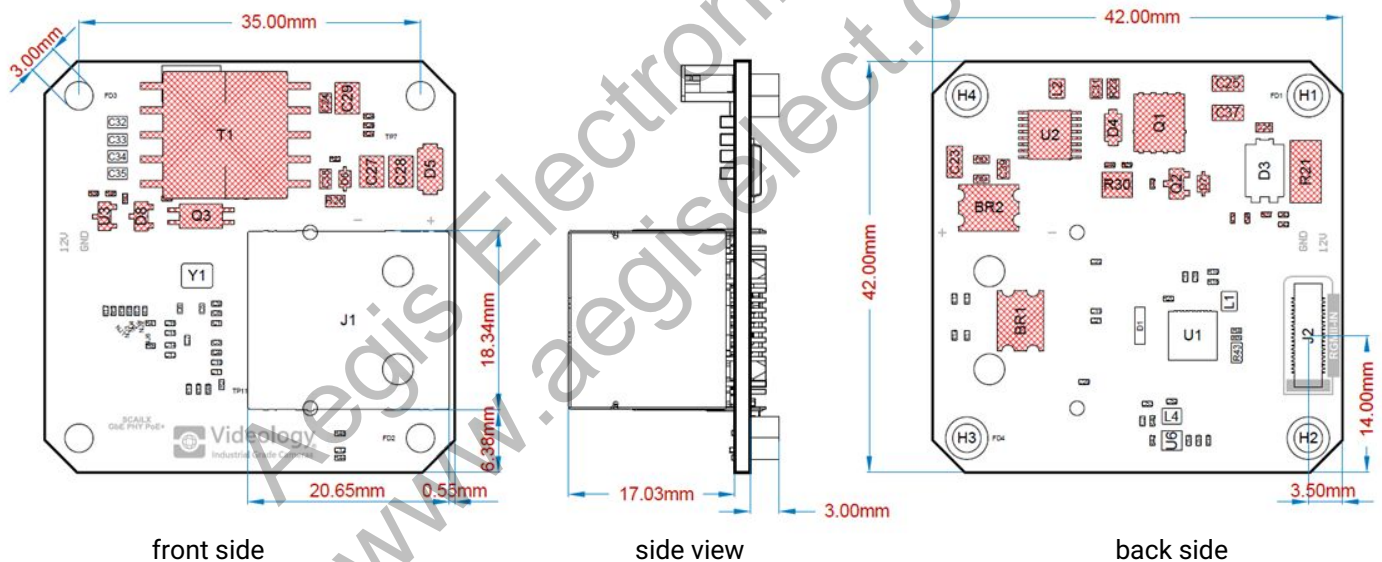


front side



back side

2.5.1.1 Dimensions



2.5.1.2 Connection

The SCAiLX-ETH-DC board is connected to the RGMII port on the SCAiLX-SOM board via a flex PCB connector with 40 pin board-2-board/wire connector.



The SCAILX-FLEX-ETH is connected between RGMII port on SCAILX-ETH-POE and RGMII on SCAILX-SOM-AI.

SCAILX-ETH-DC	Type	SCAILX-SOM-AI board
J2	RGMII	RGMII port

2.5.1.3 Programming and setup

The SCAILX-ETH-DC board needs no configuration or setup.

2.5.1.4 Datasheet

[PDS_SCAILX-ETH-DC](#)

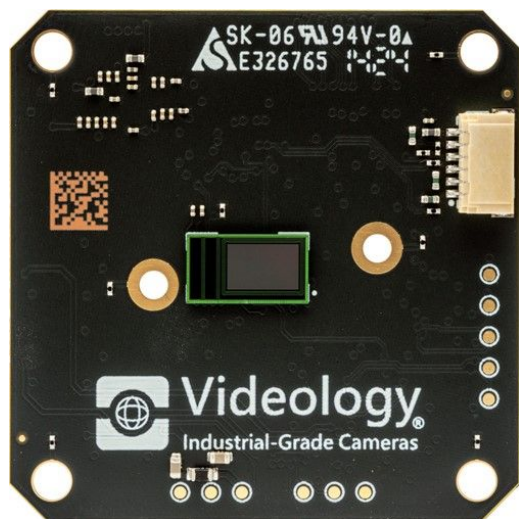
2.5.1.5 Step 3D model

[NON_POE_PHY_Board.step](#)

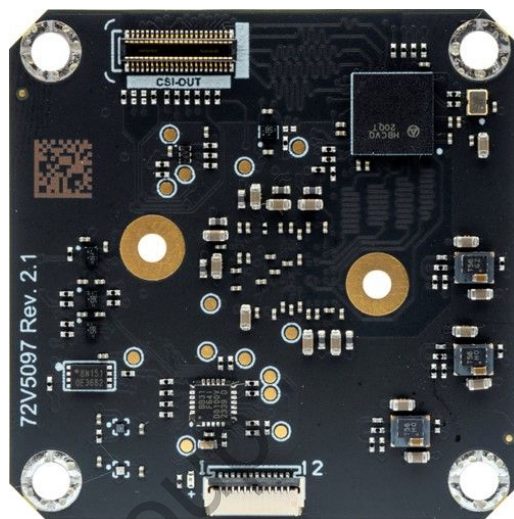
2.6 SCAILX-2GS234-xY camera

2.6.1 SCAILX-2GS234-xY 2 MP Global Shutter board level camera

The SCAILX-2GS234 is a board level camera with a 2 MP Global Shutter sensor, ISP processing chip and SCAiLX MIPI CSI compatible connector.



front side



back side

The SCAILX-2GS234 is based on the Onsemi AR0234 sensor and AP1302 ISP, and is available in Monochrome or Color versions.

2.6.1.1 Models

Partnumber	Type
SCAILX-2GS234-xM	Monochrome
SCAILX-2GS234-xC	Color
SCAILX-2GS234-xIR	Color, without IR filter

The x in the above model number indicates the selected lens mount option: 2=pinhole, 5=M12 mount, 7=no mount, 8=CS mount.

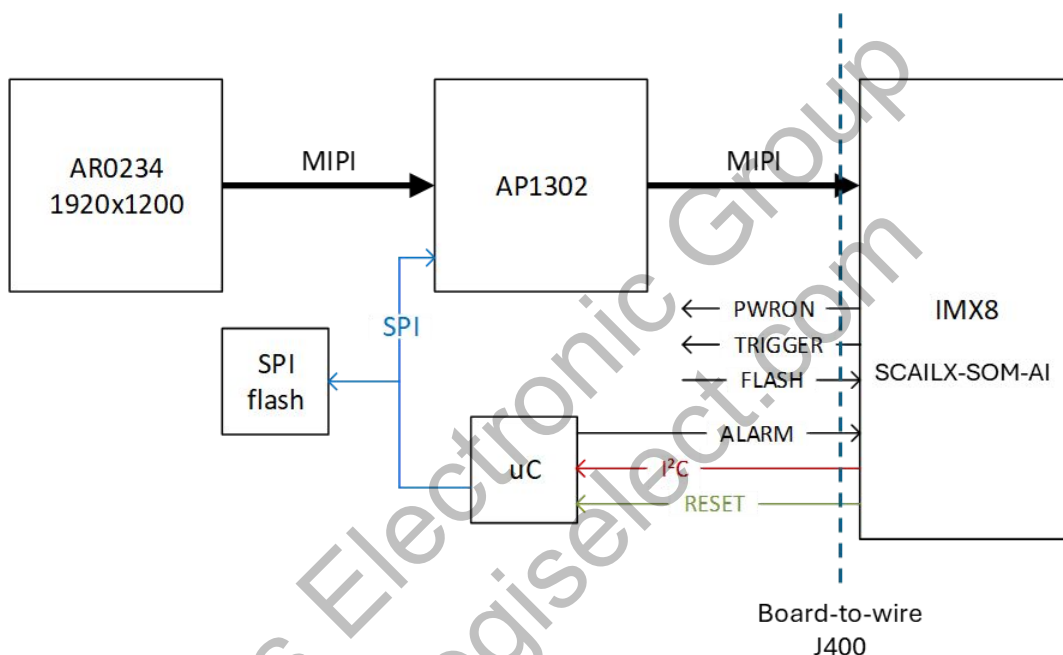
Example for the SCAILX-2GS234-xC (color) variants:

Part number	Description
SCAILX-2GS234-2C	Color sensor with pin hole lens
SCAILX-2GS234-5C	Color sensor with M12 lens mount
SCAILX-2GS234-7C	Color sensor with no mount
SCAILX-2GS234-8C	Color sensor with CS lens mount

2.6.1.2 Blockdiagram

The SCAILX-2GS234-xY is based on:

- AR0234 Onsemi Global shutter sensor with MIPI out
- AP1302 Onsemi ISP with MIPI-2-MIPI video processing
- On board microcontroller for I²C communication between SCAILX-SOM-AI and ISP
- Flash memory for microcontroller program and camera settings



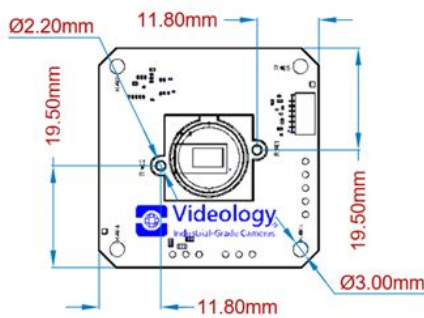
SCAILX-2GS234 block diagram

2.6.1.3 Specification

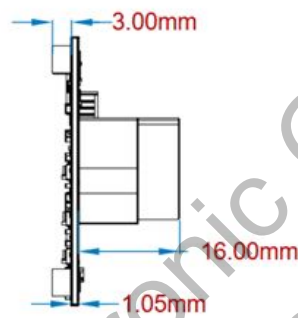
Item	Specification
Image sensor	OnSemi AR0234CS 2.3Mp Global Shutter
Recording Pixels	1920x1200
Pixel Size	3.0 μm
Image Size	5.76 x 3.6 mm
Optical size	1/2.6 inch (6.8 mm)
Shutter type	Global Shutter

Resolutions / Framerates	1920x1080 @ 30/60 fps 1280x720 @ 30/60 fps
Pixel clock rate	90 MHz
Sensitivity	Color: < 0.1 lux, Monochrome: < 0.05 lux
Signal to Noise ratio	37 dB
Dynamic range	70 dB

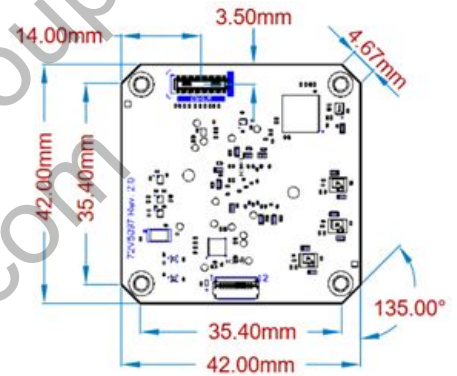
2.6.1.4 Dimensions



front side



with M12 lens mount

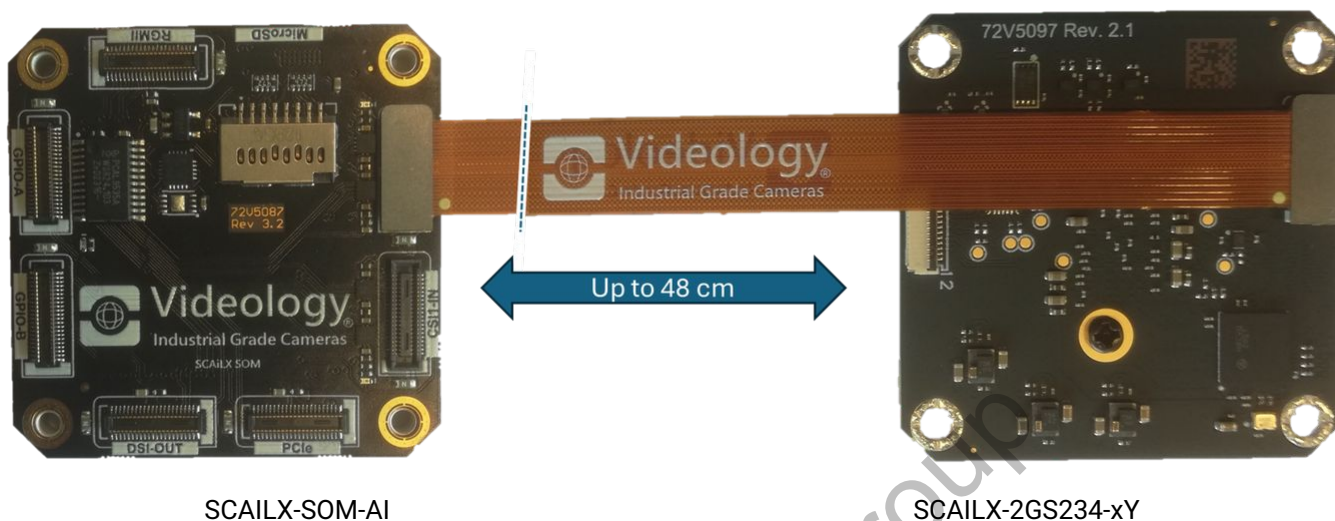


back side

2.6.1.5 Connectors

Connector	Connector type	Description
J400	DF40C-40DP-0.4V(51)	MIPI / Power connector, bioard-2-wire connector 40 pins on back side
P200	WR-WTB 1mm 6p	Trigger/ Sync & FLASH, connector on front side
J300	WR-FPC ZIF 0.5mm 12p	(Debug)

Connecting SCAiLX-FLEX between SCAiLX-2GS234-xY and SCAiLX-SOM-AI:



2.6.1.6 Single and Dual camera configuration

The SCAILX-2GS234 camera can be connected to the SCAILX-SOM board via J400 using a flex foil PCB, connecting to either CSI1 and/or CSI2 port.

SCAILX-SOM-AI SoM board	2x SCAILX-2GS234-xM/xC	2x 1080p60 max.
-------------------------	------------------------	-----------------

The SCAILX-SOM-AI board supports up to 2x SCAILX-2GS234-xM/xC in 1080p@60.

2.6.1.6.1 Single camera configuration

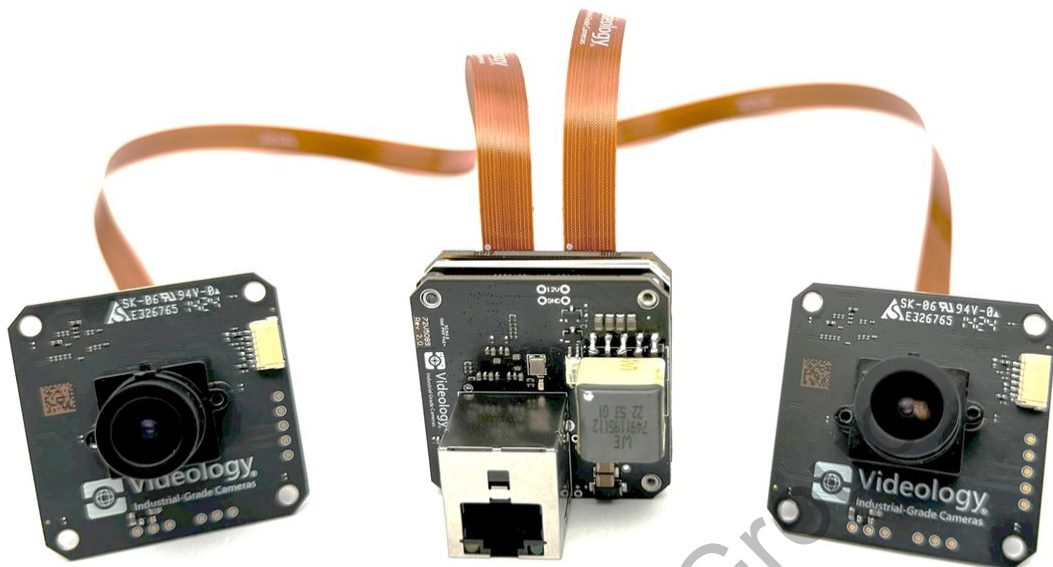
The SCAILX-SOM-AI board delivers two identical MIPI CSI-2 camera ports. The SCAILX-SOM-AI board will detect automatically which camera is connected and install the appropriate driver for it.

2.6.1.6.2 Dual camera configuration

The SCAILX-SOM-AI board delivers 2 identical MIPI CSI-2 camera ports. The SCAILX-SOM-AI board will support different MIPI camera boards.

Example of dual camera configuration using 2x SCAILX-2GS234-xY camera modules connected to the SCAILX-SOM-AI using 2x SCAILX-FLEX-480 (480 mm FlexPCB cables) for extended range between SCAILX-SOM and camera modules.

MIPI interfacing is restricted to 480 mm on each port.



SCAiLX-2GS234-5C/M

SCAiLX-SOM-AI
SCAiLX-ETH-POE/DC

SCAiLX-2GS234-5C/M

Dual camera stream using the SCAiLX-2GS234 Cameras

2.6.1.7 ISP functionality

Item	Specification
Automatic Exposure Control	Auto / Manual / Region Of Interest (ROI) / Face
Brightness	0 ~ 255
Contrast	0 ~ 255
Gamma	0.4 ~ 1.0
Saturation	0 ~ 255
Sharpness/Blur control	Yes
Noise reduction	Yes
Automatic Color Adjustment	Auto / Manual / Push2White / Set
Manual Color Adjustment	Yes
BLC	Yes
HDR	No
Mirror/Flip	Yes
Panning	Yes, (128 steps)

Digital zooming	Yes
Upgradable	Yes
Video format selection	Yes
Trigger	Yes
Flash	Yes
Sync	Yes
Lens shading correction	Yes
Face detection	Yes
Anti Flicker	Yes
JPEG quality control	Yes
Filter effects	Yes

The ISP functionality is supported via an extensive set of I²C commands through Videology's Python API.

2.6.2 User Guide

Refer to **UG_SCAILX-2GS234** for full description of camera functionality, I²C commands and more.

[ug-2.3mp-global-shutter.pdf](#)

2.6.3 Programming and setup

The SCAILX-2GS234 camera board requires no board level setting or configuration and will be detected and configured automatically at SCAILX-SOM startup.

2.6.4 Software support

[SCAILX-2GS234 V4L commands](#)

[SCAILX-2GS234 python scripts](#)

2.6.5 Datasheet

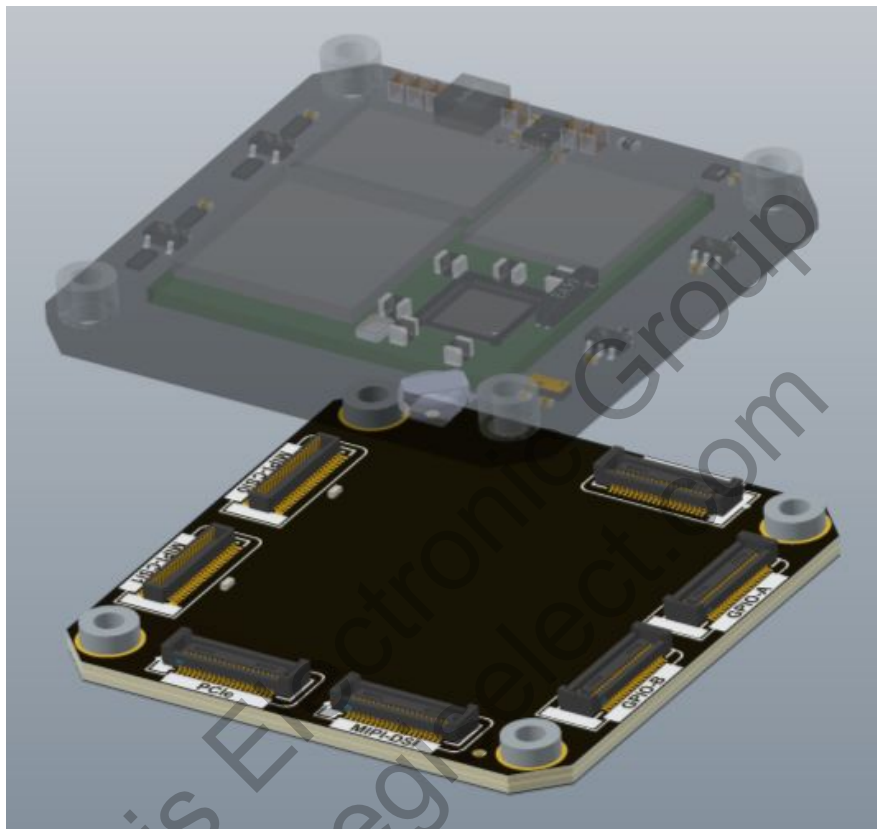
[PDS_SCAILX-2GS234-xY](#)

2.6.6 Step 3D model

[GS_2.3Mpix_camera_board.step](#)

2.7 SCAiLX - Peripheral board templates

Provided below is an Altium Designer and KiCad project template for an adapter board which can be used with the SCAiLX SOMs.



2.7.1 Altium template

[scailx-adapter-Altium.zip](#)

2.7.2 Kicad template

[Scailx-adapter-kicad.zip](#)



The template project includes all connectors to mate to a SCAiLX SOM, but it is ill-advised to connect to all SOM connectors from one board. Rather make single-purpose adapters which connect to one of the ports through the provided flex cables.

Usage

1. Open the project in Altium Designer, or KiCad.
2. Remove all connectors but the one('s) you're interested in.
3. Add necessary components circuitry as necessary
4. Route the board
5. Add the SOM board [SCAILX-SOM-STEP.zip](#) step 3D file to verify board placement in 3D.

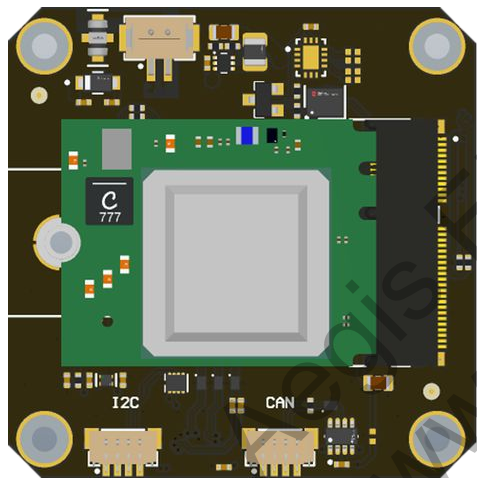
The standard-height DF40C-connectors create a 1.5 mm board-to-board spacing.

2.8 SCAILX-ACC26

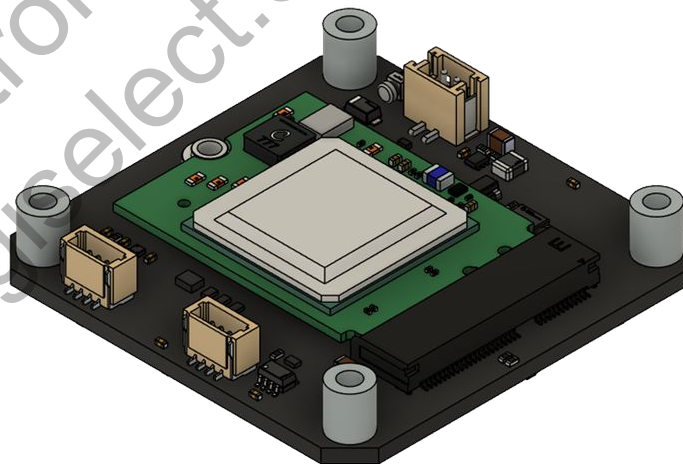
2.8.1 SCAILX-ACC26 - SCAILX AI Accelerator board

The SCAILX-ACC26 consists of the SCAILX-PCIE interface board with the HAILO-8 PCIe AI Accelerator board.

The HAILO-8 PCIe AI Accelerator can give up to 26 TOPS of AI inference performance boost to the SCAILX-AI-SOM module.



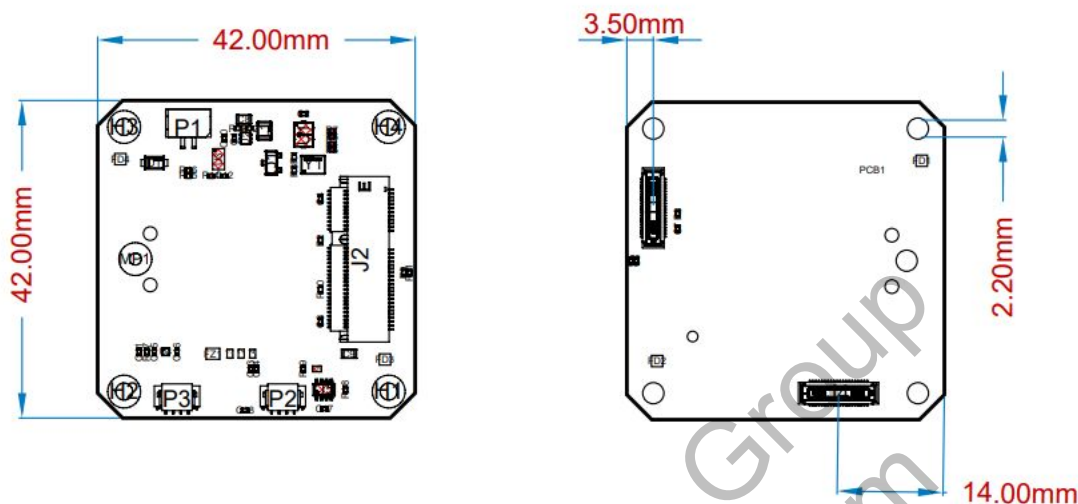
Top view



3D view

The HAILO-8 PCIe module interfaces to the SCAILX-SOM-AI module through a single lane PCIe bus for high speed data communication.

2.8.2 Dimensions



2.8.3 SCAiLX 12 Volt POWER IN

The SCAiLX-ACC26 can power a full SCAiLX board stack via connector P1. The power is supplied to the 2-pin Wurth WR WTB 1.5mm pitch connector.

The pin1 is marked with a dot on the PCB and is the ground connection (see table below).



In case SCAiLX is supplied through the 12V-IN on the SCAiLX-ACC26 board, then the SCAiLX-ETH-DC board should be used for Ethernet communication. It should be noted that the SCAiLX-ETH-DC board can also be used for supplying 9-12V DC to the SCAiLX device.

2.8.4 Board connectors

- PCIe E-key connector
 - o PCIe Gen.3 x1 channel
 - o I²C channel
- CAN bus connector (4 pin, 3V3@max. 700 mA, no fuse)
- I²C QWIIIC connector (4 pin, 3V3@max. 700 mA, no fuse)
- 12V power Input (ESD protection > 23kV, no fuse, polarity protected)
- Power out 3V3@max. 3A (PCIe, QWIIIC, CAN-bus connector) – no fuse

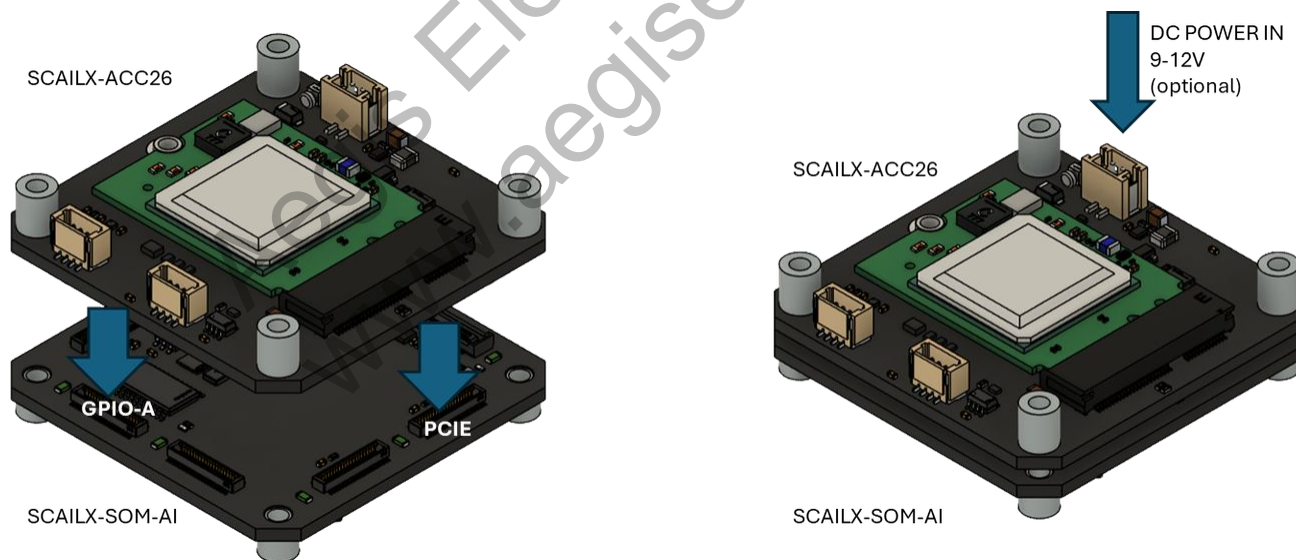
2.8.5 Pin description P1-P3

P1	Power IN 12V		P2	CAN bus		P3	I ² C bus
pin 1	GND		pin 1	GND		pin 1	GND
pin 2	VIN 9-12V		pin 2	3V3_GPIO		pin 2	3V3_GPIO
			pin 3	CAN_L		pin 3	SDA_QWIIC
			pin 4	CAN_H		pin 4	SCL_QWIIC

i **QWIIC** is a standardized, plug-and-play connectivity system developed by SparkFun Electronics. It uses a series of 4-pin JST SH connectors and cables to quickly and easily connect various sensors, displays, and other components using the I2C protocol without the need for soldering or wiring.

2.8.6 Assembly on SCAILX-SOM-AI

The SCAILX-ACC26 board can be directly fitted on the SCAILX-SOM-AI board matching the PCIE board-2-board connector and the GPIO-A board-2-board connector.



i The remaining board-2-board connectors RGMII, CSI0-IN, CSI1-IN, GPIO-B and DSI are not blocked by the SCAILX-WIFI board.

2.8.7 Alternative assembly

The SCAILX-ACC26 board can be connected to the SCAILX-SOM-AI board using the SCAILX-FLEX-60/120 connector cables. Connect the SCAILX-FLEX connectors observing the pin 1 indicator on the PCBs and the straight side of the SCAILX-FLEX connector.

2.8.8 Datasheet

[PDS_SCAILX-ACC26](#)

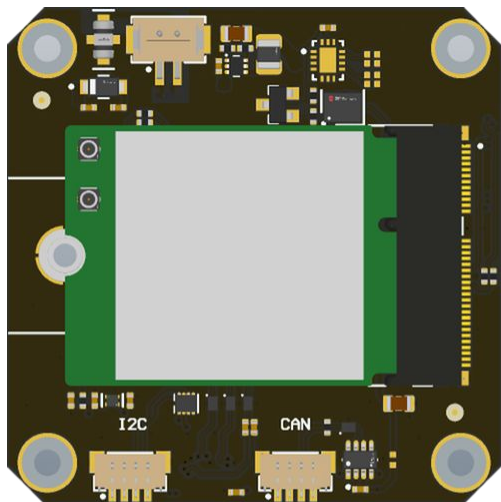
2.8.9 3D-Step Model

Refer to <http://www.videologyinc.com> SCAILX-WIFI product page.

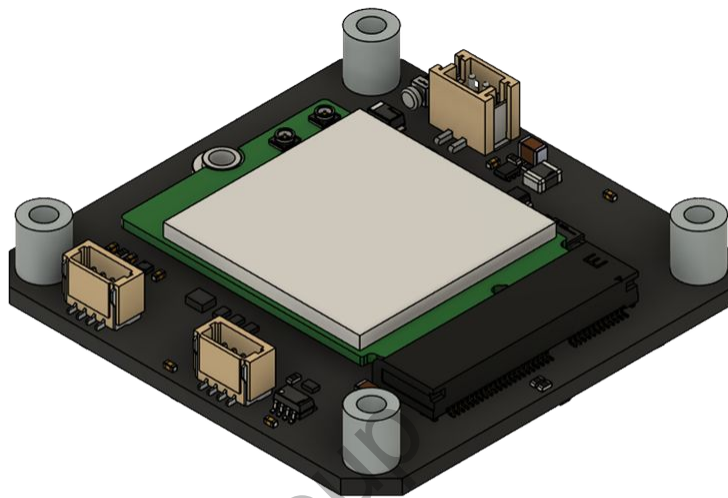
2.9 SCAILX-WIFI

2.9.1 SCAILX-WIFI - SCAILX WiFi PCIe interfacing

The SCAILX-WiFi board assembly consists of the SCAILX-PCIE board and the Intel AX201 Wi-Fi module.



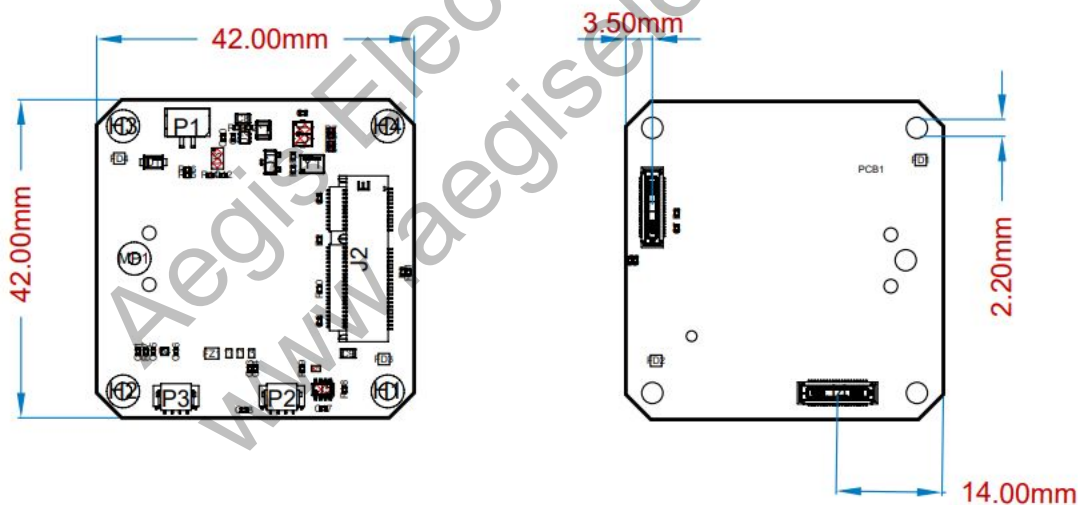
Top view



3D view

The Intel AX210 is an Intel® Wi-Fi 6 Series PCIe module with a M.2 2230 form factor (22mm x 30mm x 2.4mm).

2.9.2 Dimensions



2.9.3 SCAiLX 12 Volt POWER IN

The SCAiLX-WIFI can power a full SCAiLX board stack via connector P1. The power is supplied to the 2-pin Wurth WR WTB 1.5mm pitch connector.

The pin1 is marked with a dot on the PCB and is the ground connection (see table below).

2.9.4 Board connectors

- PCIe E-key connector
 - o PCIe Gen.3 x1 channel
 - o I²C channel
- CAN bus connector (4 pin, 3V3@max. 700 mA, no fuse)
- I²C QWIIC connector (4 pin, 3V3@max. 700 mA, no fuse)
- 12V power Input (ESD protection > 23kV, no fuse, polarity protected)
- Power out 3V3@max. 3A (PCIe, QWIIC, CAN-bus connector) – no fuse

2.9.5 Pin description P1 - P3

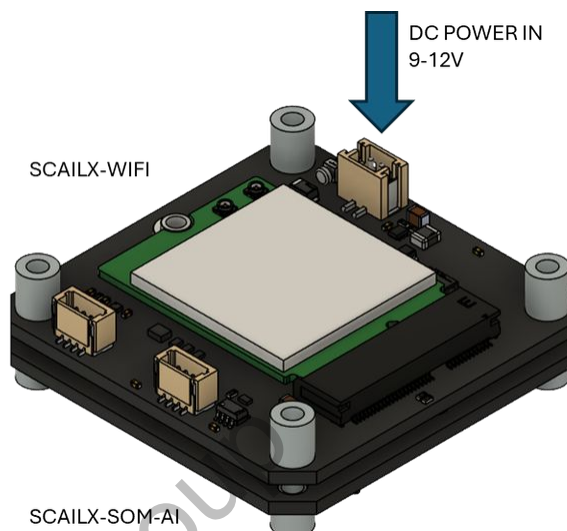
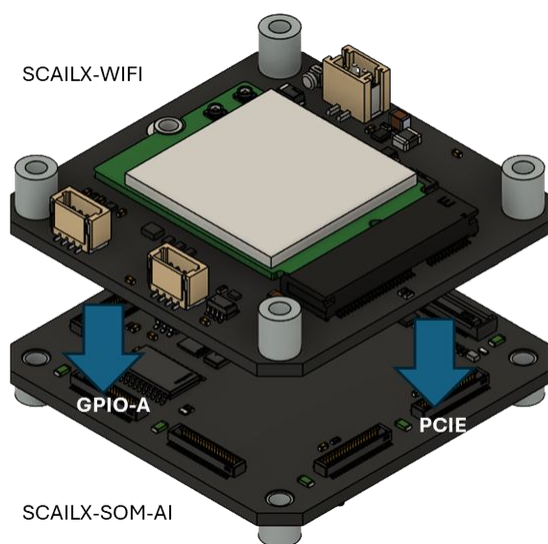
P1	Power IN 12V		P2	CAN		P3	I ² C bus
pin 1	GND		pin 1	GND		pin 1	GND
pin2	VIN 9-12 V		pin 2	3V3_GPIO		pin 2	3V3_GPIO
			pin 3	CAN_L		pin 3	SDA_QWIIC
			pin 4	CAN_H		pin 4	SCL_QWIIC



QWIIC is a standardized, plug-and-play connectivity system developed by SparkFun Electronics. It uses a series of 4-pin JST SH connectors and cables to quickly and easily connect various sensors, displays, and other components using the I2C protocol without the need for soldering or wiring.

2.9.6 Assembly on SCAiLX-SOM-AI

The SCAiLX-WIFI board can be directly fitted on the SCAiLX-SOM-AI board matching the PCIE board-2-board connector and the GPIO-A board-2-board connector.



- i** The remaining board-2-board connectors RGMII, CSI0-IN, CSI1-IN, GPIO-B and DSI are not blocked by the SCAILX-WIFI board.

2.9.7 Alternative assembly

The SCAILX-WIFI board can be connected to the SCAILX-SOM-AI board using the SCAILX-FLEX-60/120 connector cables. Connect the SCAILX-FLEX connectors observing the pin 1 indicator on the PCBs and the straight side of the SCAILX-FLEX connector.

2.9.8 Specifications Intel AX210 PCIe module

Specification:	
TX/RX Streams	2x2
Bands	2.4, 5 GHz (160MHz)
Max Speed	2.4 Gbps
Wi-Fi CERTIFIED*	Wi-Fi 6 (802.11ax)
Compliance	FIPS, FISMA
Bluetooth Version	5.2

2.9.8.1 Wi-Fi network setup

A Wi-Fi network can easily be set up using the 'impala' application, preinstalled on SCAILX.

Impala is a Linux based graphical tool for setting up SCAILX as an Wi-Fi Access Point (AP) or Station.

More information and usage can be found at: <https://github.com/pythops/impala>

Wi-Fi network select tool, module as STATION:

```
impala -m station
```

Device			
Name	Mode	Powered	Address
wlan0	station	On	e8:62:be:2b:e2:56

Station			
State	Scanning	Frequency	Security
connected	false	5.32 GHz	Open

Known Networks					
	Name	Security	Hidden	Auto Connect	Signal
♦	WLAN-PUB	open	false	true	84%

New Networks			
Name	Security	Signal	
Philips	8021x	84%	♦
WLAN-HTC	8021x	84%	♦
WLAN-HTC-IPSK	psk	84%	♦
T7c_z4Q9_2.4GHz	psk	72%	♦
mimotica_exp	psk	68%	♦
CT-Wm202007-NB0	psk	88%	♦
Counter_30104	psk	82%	♦

Practical Wi-Fi commands:

```
iw dev
#
iw phy#0 info
# gives all relevant information about supported frequency bands and more
```

2.9.8.2 Bluetooth setup

The Bluetooth device can be interrogated:

```
hciconfig
#
hci0:   Type: Primary   Bus: USB
        BD Address: 9C:B1:50:AF:3D:E9   ACL MTU: 1021:4   SCO MTU: 96:6
        DOWN
        RX bytes:20325 acl:0 sco:0 events:3297 errors:0
        TX bytes:814713 acl:0 sco:0 commands:3295 errors:0
hciconfig hci0 up
```

'bluetui' is a Linux based tool for setting up a Bluetooth device and discovering adjacent devices for pairing.

<https://github.com/pythops/bluetui>

Download the latest release to your host PC. Copy the executable to the SCAILX device and modify the file attribute to executable. Run the application.

```
# Download bluetui-aarch64-linux-gnu for github to Host PC
# copy application to SCAILX device
scp bluetui-aarch64-linux-gnu root@scaillx-ai:/root
# open ssh terminal for scaillx device
chmod a+x ./bluetui-aarch64-linux-gnu
./bluetui-aarch64-linux-gnu
```

See instructions on above GitHub of blueui to pair SCAILX Bluetooth device with adjacent device.

2.9.9 Datasheet

[PDS_SCAILX-WIFI](#)

2.9.10 3D-Step Model

Refer to www.videologyinc.com SCAILX-WIFI product page.

3 SCAILX-ZB - Zoom block

3.1 Assembly

The SCAILX-ZB assembly combines the SCAiLX processor board (**SCAILX-SOM-AI**) with a Ethernet PoE adapter board (**SCAILX-ETH-POE**), and the LVDS-MIPI adapter board (**SCAILX-LVDS-2-MIPI**) for zoom block video input.

The SCAILX-LVDS-2-MIPI supports zoom blocks from different manufactures based on LVDS output and resolutions up to 1920x1080p at 25/30/50/60 fps.

The SCAILX-ZB is base on the housing for the Videology zoom block camera: **24Z2.1W-10X-LVDS**



SCAILX-ZB 10x zoom



inside view LVDS + SOM stack



view of ETH board

The board stack is attached to a heat spreader (purple aluminium plate) together with the zoomblock is enclosed in an aluminium enclosure.

3.1.1 Datasheet

[SCAILX-ZB-v0.1](#)

[24Z2.1W-10X-LVDS](#)

3.1.2 3D step model

[SCAILX_ZB_MODEL.zip](#)

4 SCAILX-CAM board level camera

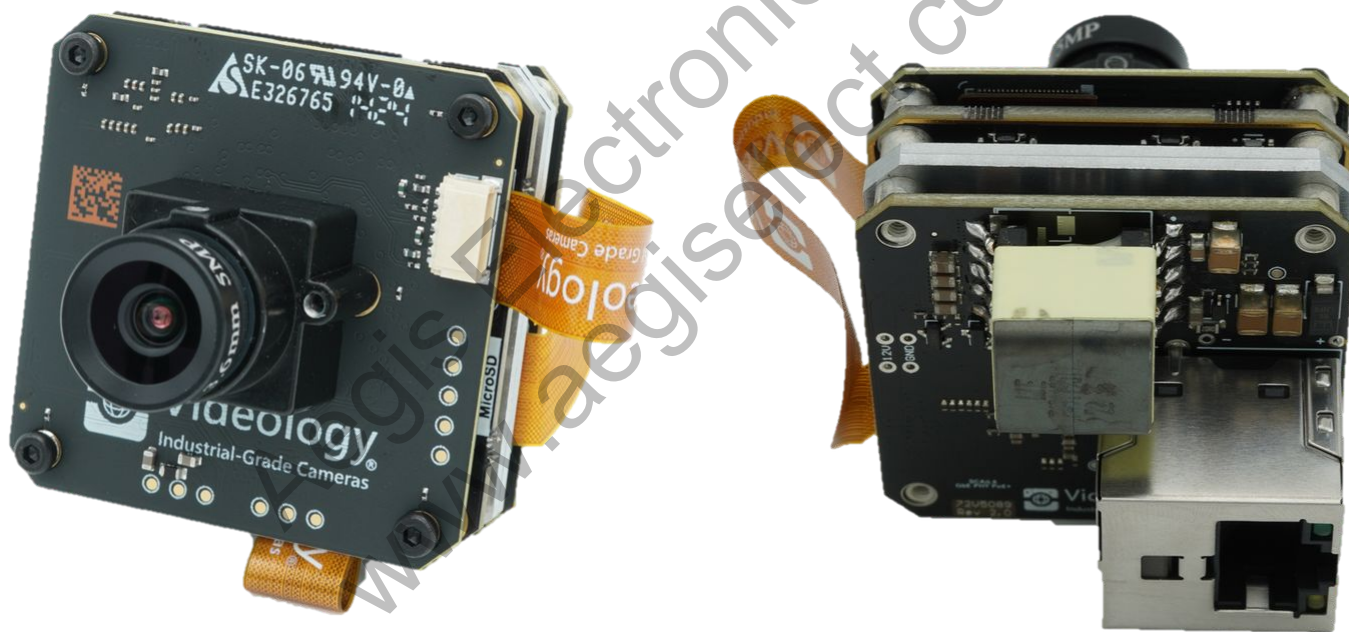
4.1 Assembly

SCAILX boards can be assembled into a compact camera solution without housing.

The board level camera assembly is initially intended for the concepting phase of applying the SCAILX modules in a engineering design.

The SCAILX camera is based on the 2 Mega Pixel Global shutter camera module SCAILX-2GS234 with Color or Monochrome sensor and M12 lens mount.

4.1.1 Assembled camera



The SCAILX-2GS234 camera has an onboard ISP (Image Signal Processor) with I²C support via the SCAILX-SOM-AI module and hardware functionality like Synchronization and Triggering.

Videology prepared a development kit based on the SCAiLX boards: SCAILX-DEV kit

4.1.2 SCAILX-DEV kit

Part*	Quantity
SCAILX-SOM-AI	1
SCAILX-ETH-POE (or SCAILX-ETH-DC)	1
SCAILX-2GS234-5C (or SCAILX-2GS234-5M)	1
SCAILX-FLEX-ETH	1
SCAILX-FLEX-60	1

*: Contact Videology for detailed information on SCAiLX camera development kit.

4.1.3 Resources for the SCAILX-CAM

[SCAILX-2GS234 Sync and Trigger Configuration](#)

[SCAILX-2GS234 V4L commands](#)

[SCAILX-2GS234 python scripts](#)

4.1.4 Datasheets and User Guide

- [PDS_SCAILX-2GS234-xY](#)
- [UG_SCAILX-2GS234-xY](#)

5 SCAiLX Software manual

SCAiLX uses an Embedded Yocto Linux based distribution, based on the NXP Linux platform.

The SCAiLX-SOM-AI can stream video from the two available video sources (MIPI CSI0 and MIPI CSI1). Video streaming applications available in basic configuration are:

- rtsp streaming using gstreamer and viewing with gstreamer or Videolan VLC [Playing a RTSP stream](#)
- web streaming using go2rtc, with an example of embedding a viewer in html [Web streaming - webRTC support](#)
- Running AI model with Python application [Running AI models in python with Tensorflow-lite](#)

For developing user application or Kernel drivers for advanced developers, SCAiLX offers:

- Visual Code Studio support [Developing software on SCAiLX - Visual Code Studio](#)
- YOCTO development environment [Yocto Embedded Linux Development](#)

The SCAiLX-2GS234 camera has extensive software support available here [SCAiLX-2GS234 - Global Shutter Camera - Software](#)

5.1 Software Update

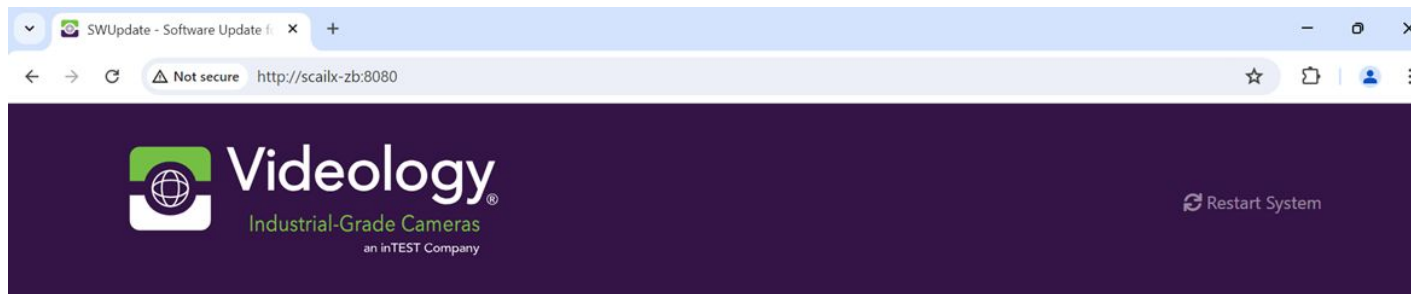
SCAiLX uses [SWUpdate](#) for software updates. This provides:

- Convenient web interface at <device ip>:8080 to drag-n-drop update images.
- Update from USB or SD-Card.
- Fault-tolerant updates by utilising redundant OS images.

5.1.1 SWUpdate web page

Scailx hosts the web ui for swupdate on port 8080.

1. Get the latest firmware file from the [Scailx Latest Release](#) page
2. Open the SWUpdate page on the device as: <http://scailx-ai.local:8080>
3. Drap-and-drop the .swu file onto the “**click Here...**” block on the page.
4. Update will auto install the firmware and reboot the device.



Software updater

Upload a .swu software image below.

Warning: Modifications to `/etc/` files will be re-applied after update. This preserves hostname changes, SSH auth, and user started services.

If this is not wanted, either run `touch /storage/clear_overlay` to factory-reset all filesystem changes on the next boot, or manually delete unwanted files in `/storage/overlay/upper`.

Any Other filesystem changes or user files will be saved in a backup folder.

For example, if you created `/home/root/myfile.txt` it will be saved as `/storage/overlay/backup/home/root/myfile.txt`, and `/home/root/` will be clear after update.

You can restore files from the backup when logging in, or copy them manually.

 **Software Update**

Click here, or drag and drop a software update image file to this area.

 Update not started.

5.2 Playing a RTSP stream

5.2.1 Start a RTSP stream on device

```
gst-variable-rtsp-server -p 9001 -u "v4l2src device=/dev/video0 ! video/x-raw,width=1280,height=720 ! queue ! vpuenc_h264 ! rtph264pay name=pay0 pt=96"
```

The `gst-variable-rtsp-server` is a **GStreamer plugin** that facilitates **RTSP (Real-Time Streaming Protocol)** server functionality. It allows you to stream multimedia content over a network connection.

It starts a gstreamer-pipeline (from the `-u` parameter) and hosts it on the specified port (`-p`).

In this case, the video is first encoded using the imx8's built-in h.264 encoder (`vpuenc_h264`) and packaged in h.264 rtp payloads (`rtph264pay`).

5.2.2 gstreamer player on host

On a computer with [Gstreamer installed](#), run:

```
gst-launch-1.0 rtspsrc location=rtsp://scailx-zb.local:9001/stream latency=0 buffer-mode=auto ! decodebin ! autovideosink
```

This starts a gstreamer pipeline which

1. Connects to an **RTSP** stream at `rtsp://scailx-zb.local:9001/stream`
2. Determines the incoming data format (h.264) and decodes it (`decodebin`)
3. Displays the video to the default video display device (computer monitor)

5.2.3 VLC media player on host

On the computer with vlc player installed:

1. Start vlc media player
2. Open Media → Open Network Stream ...
3. Enter rtsp URL in Network URL box: `rtsp://scailx-zb.local:9001/stream`
4. Check 'Show more options' and reduce Caching to 100ms. Start streaming by pressing Play.

5.3 Web streaming - webRTC support

SCAILX comes with a convenient webrtc-sink gstreamer element for streaming to modern browsers. The [go2rtc](#) application is also pre-installed, which is a multi-protocol streaming application that supports multiple input formats and multiple output formats.

5.3.1 Stream to web

scailx's "websink" gstreamer plugin streams a H264 to your browser using WebRTC. There is also a convenience wrapper around websink which is set up as the default videosink element. So streaming to the browser is as simple as:

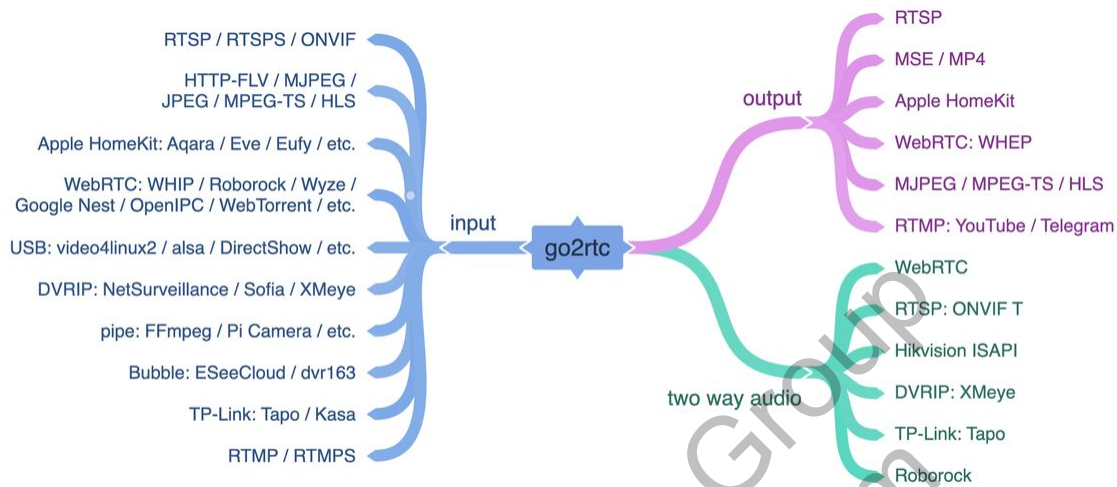
```
gst-launch-1.0 v4l2src device=/dev/video-isi-csi0 ! video/x-raw,width=1920,height=1080,framerate=60/1 ! autovideosink
```

```
root@scailx-reference:~#  
root@scailx-reference:~#  
root@scailx-reference:~#  
root@scailx-reference:~# gst-launch-1.0 v4l2src device=/dev/video-isi-csi1 ! video/x-raw,width=1920,height=1080,framerate=60/1 ! autovideosink  
Setting pipeline to PAUSED ...  
HTTP server started at http://scailx-reference.local:8091 and http://192.168.0.126:8091  
HTTP server started at http://scailx-reference.local:8091 and http://192.168.0.126:8091  
Pipeline is live and does not need PREROLL ...  
Pipeline is PREROLLED ...  
Setting pipeline to PLAYING ...  
New clock: GstSystemClock  
Redistribute latency...  
0:00:00.2 / 99:99:99.
```

1 example of using "autovideosink" to stream to browser.

Open the link printed in the terminal to view the stream in your browser.

5.3.2 go2rtc application



Windows Linux macOS zero-dependency zero-delay zero-cost camera streaming app

go2rtc hosts a web front-end at port 1984 of the device's address. Opening for example scailx-ai.local:1984 shows the following:

Version: 1.9.4, Config: /etc/default/go2rtc.yaml

☒ webrtc ☒ mse ☒ hls ☒ mjpeg

☐ Name	Online	Commands
☐ cam0-gs-AR0234_1080p	0 / info / probe / net	stream links delete
☐ cam0-gs-AR0234_720p	0 / info / probe / net	stream links delete

2 go2rtc web interface hosted at port 1984

You can start a stream from the list of cameras shown by clicking on the "stream" link.

You can also navigate to "links" for any of the cameras to get a corresponding RTSP, MSE or HLS stream link.

Its also possible to use go2rtc to stream to RTMP for streaming to a broadcast service like Youtube or Twitch.

The status of the application go2rtc can be checked on the device through ssh access:

```
# open ssh connection to the SCAILX device
ssh root@<ip-address>
```

```
# check if go2rtc is up and running
systemctl status go2rtc.service
```

5.3.2.1 Implementing a simple web server using go2rtc stream

its possible to create a simple web application which displays the streams from the go2rtc application.

Refer to the [go2rtc](#) documentation or contact Videology support for examples.

5.4 Running AI models in python with Tensorflow-lite

5.4.1 SCAiLX AI application in Python3 – Object detection

The AI application is a good example of capturing live images from the camera, resize the image for the AI model, invoke the AI model, receive data of the objects detected, draw bounding boxes and stream the resulting video as an rtsp stream.

The application is written in **python3**, uses **gstreamer** module for capture the live images and streaming the video and TensorFlow-lite module for doing the inference.



5.4.2 Preparation and running AI application

The Object Detection demo is not pre-installed on SCAiLX-SOM-AI or included in the Firmware. The source code can be downloaded or cloned from:

<https://github.com/VideologyInc/python-tflite-mobilenet-demo>

Power on the device by connecting the SCAiLX unit to a PoE Ethernet switch or a PoE injector.

Connect the Ethernet switch or PoE connector to a DHCP enabled network.

The unit will come up as 'scaix-ai.local' or equivalent. The '.local' can be different depending on your network.

The AI application can be using /dev/video0 or /dev/video1 port on the SCAiLX-SOM-AI and will capture the images and perform the AI Inference. The resulting framerate of the application depends on the different steps in the pipeline.

The On Screen Display text will give real time information of all steps in the python pipeline.

The application can be run via entering:

```
cd /root/python-tflite-mobilenet-demo
```

```
python3 object-detection.py -d /dev/video0
```

use --help for more information of the command line options.

Run the video stream on the HOST PC as:

```
rtsp://scailx-ai.local:8554/stream # in VLC or other IP stream capable viewer
```

or using gstreamer:

```
gst-launch-1.0 rtspsrc location=rtsp://scailx-ai.local:8554/stream latency=0 connection-speed=3000 ! queue !  
decodebin ! queue ! videoconvert ! autovideosink sync=false
```

The screen would look like this:



In the top left corner the application generated OSD will give timing information of the AI pipeline from image capture up to sending the image out for streaming.

The average elapsed time for AI Inference (invoke) for the object detection by the NPU is 9~10 ms. The overall elapsed time for capturing and sending out for streaming of each image is ~30 ms resulting in Full HD 1920x1080@25 fps or HD 1280x720@30fps.

On Windows and Linux host computers with gstreamer installed, video stream can be run via:

Windows/Linux :

```
gst-launch-1.0 rtspsrc location=rtsp://scailx-ai.local:8554/stream latency=0 connection-speed=3000 ! queue !
decodebin ! videoconvert ! autovideosink sync=false
```

Windows :

```
gst-launch-1.0 rtspsrc location=rtsp://scailx-ai.local:8554/stream latency=0 connection-speed=30000 ! queue !
decodebin ! videoconvert ! fpsdisplaysink sync=false
```

The total time of capturing the image, performing the AI Inference and outputting the image buffer to the rtsp server takes roughly 20-40 ms.

Other command line options of object-detection.py:

```
python3 object-detection.py --help
usage: object-detection.py [-h] [--model_path MODEL_PATH] [--model MODEL] [--label LABEL] [--device DEVICE]
[--resolution RESOLUTION] [--framerate FRAMERATE] [--accl ACCL]

Object detection - TensorFlow Lite

options:
  -h, --help            show this help message and exit
  --model_path MODEL_PATH, -p MODEL_PATH
                        Path to model and label files
  --model MODEL, -m MODEL
                        Name of model file
  --label LABEL, -l LABEL
                        Name of label file
  --device DEVICE, -d DEVICE
                        Video device [/dev/video1]
  --resolution RESOLUTION, -r RESOLUTION
                        [1080p] or 720p
  --framerate FRAMERATE, -f FRAMERATE
                        Capture framerate [60] or 30
  --accl ACCL, -a ACCL  Use accelerator [1] or 0
```

[..] default setting, if no parameter supplied.

5.4.3 Development of your own AI application

Starting with the source code of object-detection.py, you can gain an understanding of capturing images from the camera, preparing for inference, invoking the inference, modifying the image AI information and outputting the resulting frame for streaming in compressed format over the network.

5.4.4 Useful tools

Network IP scanner software

Advanced IP scanner (<http://www.advanced-ip-scanner.com>)

Viewer software

VLC : <http://www.videolan.org>

gstreamer : <https://gstreamer.freedesktop.org/download/>

Model viewer

NETRON: (<https://github.com/lutzroeder/netron>)

5.4.4.1 Source code

The source code for the object detection can be found here:

<https://github.com/VideologyInc/python-tflite-mobilenet-demo>

5.5 SCAILX-2GS234 - Global Shutter Camera - Software

The SCAILX-2GS234 Global Shutter camera integrates to the SCAILX platform via a Embedded Linux Kernel driver that takes care of the communication of the processor with the onboard microprocessor. The ISP (Image Signal Processor) and images sensor are controlled by the microprocessor via a dedicated I²C bus.

The I²C commands are in full detailed described in the User Guide **UG_2.3MP-Global-Shutter**, available for download on the Videology website.

Available information:

- The SCAILX-2GS234 camera supports synchronization and trigger as described in detail under [SCAILX-2GS234 Sync and Trigger Configuration](#) .
- The integration with the Embedded Linux on the SCAILX-SOM-AI is via the support of Video-4-Linux driver, as described in [SCAILX-2GS234 V4L commands](#) .
- Utilities in python can be found here: [SCAILX-2GS234 python scripts](#)

5.5.1 SCAILX-2GS234 V4L commands

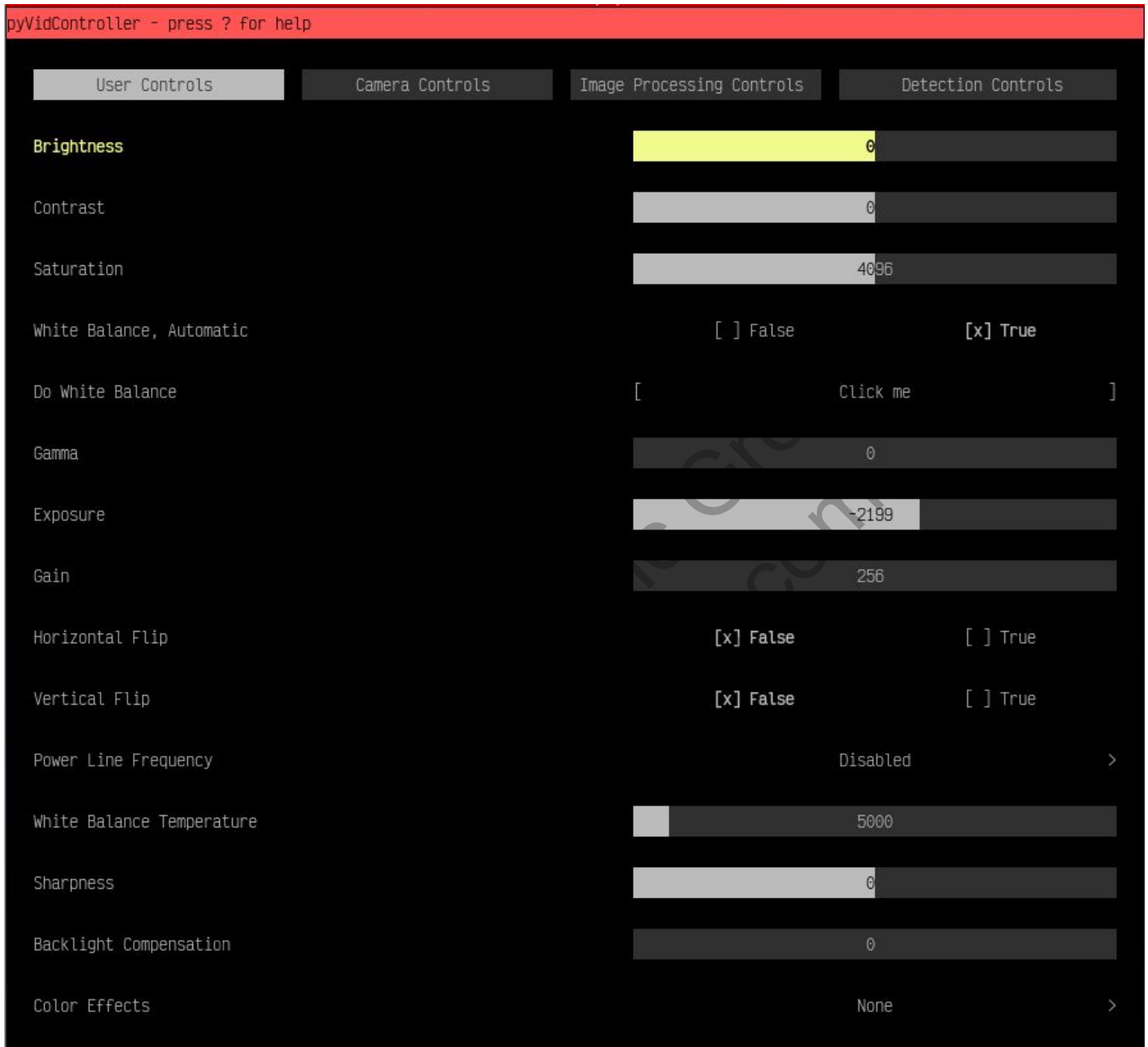
5.5.1.1 Description

This document describes the V4L2 interface commands and the usage guide for the Global shutter MIPI camera for the SCAILX platform.

5.5.1.2 TUI - Text User Interface

Scailx includes [pyvidctrl](#), a terminal GUI to conveniently adjust V4L2 control values. To use it, open an SSH terminal and execute:

```
pyvidctrl -d /dev/links/gs_AR0234_mipi_0
```



5.5.1.3 User Controls

Command	Code	Type	Controls
brightness	0x00980900	int	min=-4096 max=4096 step=32 default=0 value=0 flags=slider
contrast	0x00980901	int	min=-32768 max=32767 step=256 default=0 value=0 flags=slider

Command	Code	Type	Controls
saturation	0x00980902	int	min=0 max=8192 step=32 default=4096 value=4096 flags=slider
white_balance_automatic	0x0098090c	bool	default=1 value=1
do_white_balance	0x0098090d	button	value=0 flags=write-only, execute-on-write
gamma	0x00980910	int	min=0 max=32767 step=128 default=0 value=0 flags=slider
exposure	0x00980911	int	min=-8192 max=2048 step=40 default=-2192 value=-2199
gain	0x00980913	int	min=256 max=32767 step=1 default=256 value=256
horizontal_flip	0x00980914	bool	default=0 value=0
vertical_flip	0x00980915	bool	default=0 value=0
power_line_frequency	0x00980918	menu	min=0 max=3 default=3 value=0 (Disabled)
white_balance_temperature	0x0098091a	int	min=0 max=65535 step=1 default=6500 value=5000
sharpness	0x0098091b	int	min=-32768 max=32767 step=256 default=0 value=0 flags=slider
backlight_compensation	0x0098091c	int	min=0 max=128 step=1 default=0 value=0
color_effects	0x0098091f	menu	min=0 max=15 default=0 value=0 (None)
noise_reduction	0x009810d0	int	min=-32768 max=32767 step=256 default=0 value=0 flags=slider

Table 1 V4L2 user controls

5.5.1.3.1 Brightness

The brightness value ranges from -4096 to 4096, 32.
 The format is a two's complement value.

5.5.1.3.2 Contrast

The contrast value ranges from -32768 to 32767, step 256.
 The format is a two's complement value.

5.5.1.3.3 Saturation

Saturation ranges from 0x0000 to 0x2000, step 32.
e.g. the default 1.0 is 0x1000.

5.5.1.3.4 White balance

White balance value 0 for manual or 1 for automatic.

5.5.1.3.5 Gamma

The format is s3.12. Gamma is controllable from 0x600 to 0x8000, step 128.
Gamma values lower than 0x600 result in the sRGB gamma curve.

e.g:

Gamma of 1.0 = 0x1000

Gamma of 2.5 = 0x2800

5.5.1.3.6 Exposure

Exposure sets the target level from -8192 to 2024 in steps of 40.

5.5.1.3.7 Horizontal Flip

Horizontal flip value 0 to disable or 1 to enable.

5.5.1.3.8 Vertical Flip

Vertical flip value 0 to disable or 1 to enable.

5.5.1.3.9 Anti Flicker (power line frequency mode)

- 0 Disable.
- 1 Force correction to 50 Hz
- 2 Force correction to 60 Hz
- 3 Automatic detection

5.5.1.3.10 White balance temperature

Set the white balance temperature in Kelvin when the White balance mode is set to manual.

5.5.1.3.11 Sharpness

The sharpness value ranges from -32768 to 32767, step 256.
 The format is a two's complement value.

5.5.1.3.12 Backlight compensation

The backlight compensation level can be set from 0 to 128.

5.5.1.3.13 Color Effects

Set color effects:

- 0 Normal
- 1 Black & White
- 2 Sepia
- 3 Negative
- 4 Emboss
- 5 Sketch
- 6 Sky Blue
- 7 Green Grass
- 8 Skin Whiten (defaults to Normal)
- 9 Vivid (defaults to Normal)
- 10 Aqua (defaults to Normal)
- 11 Art Freeze
- 12 Silhouette
- 13 Solarization
- 14 Antique
- 15 Set Cb/Cr (defaults to Normal)

5.5.1.3.14 Noise Reduction

The noise reduction value ranges from -32768 to 32767, step 256.
 The format is a two's complement value.

5.5.1.4 Camera controls

Command	Code	Type	Controls
auto_exposure	0x009a0901	menu	min=0 max=3 default=0 value=0 (Auto Mode)
exposure_time_absolute	0x009a0902	int	min=0 max=65535 step=1 default=333 value=333
pan_absolute	0x009a0908	int	min=0 max=128 step=1 default=64 value=64
tilt_absolute	0x009a0909	int	min=0 max=128 step=1 default=64 value=64

Command	Code	Type	Controls
zoom_absolute	0x009a090d	int	min=0 max=10240 step=1 default=256 value=256
white_balance_auto_preset	0x009a0914	menu	min=0 max=9 default=6 value=6 (Daylight)
exposure_metering_mode	0x009a0919	menu	min=0 max=3 default=1 value=1 (Center Weighted)
zoom_speed	0x009a0932	int	min=-128 max=127 step=1 default=-128 value=-128 flags=slider
region_of_interest_bound	0x009a0933	bool	default=0 value=0
region_of_interest_lock	0x009a0934	bool	default=0 value=0
region_of_interest_face	0x009a0935	bool	default=1 value=1
exposure_upper_x_100us	0x009a0936	int	min=0 max=65535 step=1 default=333 value=333 flags=slider
exposure_max_x_100us	0x009a0937	int	min=0 max=65535 step=1 default=333 value=333 flags=slider
gain_upper_value_u8_8	0x009a0938	int	min=0 max=65535 step=1 default=2048 value=2048 flags=slider
gain_max_max_2_value_s7_8	0x009a0939	int	min=0 max=65535 step=1 default=1408 value=1408 flags=slider
blc_window_x0_0_128	0x009a093a	int	min=0 max=128 step=1 default=0 value=0 flags=slider
blc_window_y0_0_128	0x009a093b	int	min=0 max=128 step=1 default=0 value=0 flags=slider
blc_window_x1_0_128	0x009a093c	int	min=0 max=128 step=1 default=128 value=128 flags=slider
blc_window_y1_0_128	0x009a093d	int	min=0 max=128 step=1 default=128 value=128 flags=slider
blc_ratio_0_128	0x009a093e	int	min=0 max=128 step=1 default=128 value=128
blc_face_level_0_128	0x009a093f	int	min=0 max=128 step=1 default=128 value=128 flags=slider
blc_face_weight_0_128	0x009a0940	int	min=0 max=128 step=1 default=128 value=128 flags=slider
blc_roi_level_0_128	0x009a0941	int	min=0 max=128 step=1 default=0 value=0 flags=slider
awb_manual_x	0x009a0942	int	min=-32768 max=32767 step=1 default=0 value=0 flags=slider
awb_manual_y	0x009a0943	int	min=-32768 max=32767 step=1 default=0 value=0 flags=slider

Command	Code	Type	Controls
store_registers_to_nvm	0x009a0946	button	value=0 flags=write-only, execute-on-write
restore_registers_from_nvm	0x009a0947	button	value=0 flags=write-only, execute-on-write
restore_to_factory_settings	0x009a0948	button	value=0 flags=write-only, execute-on-write
reboot_camera	0x009a0949	button	value=0 flags=write-only, execute-on-write

Table 2 V4L2 camera controls

5.5.1.4.1 Auto Exposure

Sets the Exposure modes:

- 0 Automatic mode
- 1 Manual mode
- 2 Shutter Priority mode
- 3 Aperture Priority mode (unused, defaults to Automatic mode)

5.5.1.4.2 Exposure Time, Absolute

In manual mode the exposure time can be set in steps of 100 us.

e.g.: 333 = 33.3 ms

5.5.1.4.3 Pan, Absolute

Allows left/right panning when the zoom factor is greater than 1x.

- 0 Full left pan
- 64 Center
- 128 Full right pan

5.5.1.4.4 Tilt, Absolute

Tilt up/down

- 0 Full up tilt
- 64 Center
- 128 Full right tilt

5.5.1.4.5 Zoom, Absolute

Zoom factor. Format s7.8., max 40x = 0x2800.

- 1x 0x0100
- 1.5x 0x0180
- 32x 0x2000

5.5.1.4.6 White Balance, Auto & Preset

When white balance is in manual mode a preset can be selected

- 0 Manual
- 1 Auto (reserved) Do not use!
- 2 Incandescent
- 3 Fluorescent
- 4 Horizon
- 5 Daylight
- 6 Flash
- 7 Cloudy
- 8 Shade

5.5.1.4.7 Exposure, Metering Mode

The metering mode determines how the exposure algorithm uses the picture.

- 0 Average
- 1 Center Weighted
- 2 Spot
- 3 Matrix

5.5.1.4.8 Zoom speed

The zoom speed determines how quickly the camera moves to the set zoom factor.

- 128 Immediate
- 0 Stop (when moving toward the set zoom factor the zoom movement can be stopped)
- <128 Linear (speed is constant)
- >128 Fractional (speed slows down as zoom factor is reached)

5.5.1.4.9 Region of Interest (Bound)

The Automatic exposure algorithm will use the ROI. This function can be enabled or disabled.

5.5.1.4.10 Region of Interest (Lock)

When using the ROI, once settled the ROI can be locked. This function can be enabled or disabled.

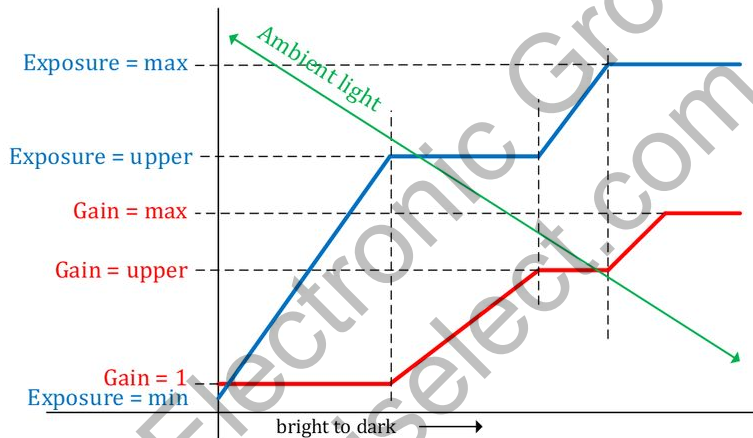
5.5.1.4.11 Region of Interest (Face)

The automatic exposure algorithm uses face-based exposure. This function can be enabled or disabled.

5.5.1.5 Exposure Limits

The exposure limits: "exposure upper" and "exposure max" are used by the AEX algorithm to limit the exposure time. Setting these limits beyond the framerate will lower the framerate to maintain the wanted brightness level.

1. Minimum gain is used from start and AE is increasing exposure time up to "upper exposure".
2. Once "upper exposure" is exhausted AE starts increasing gain up to "upper gain". Best performing gain is always used first, so analog gain will typically be used up before digital gain.
3. Once "upper gain" is exhausted AE again starts increasing exposure time up to "max exposure".
4. After that gain is increased up to max gain.
(this value is logarithmic, e.g., gain max = 3.0 means 8x gain).



3 Figure 1 Exposure and Gain

5.5.1.5.1 Exposure Upper (x 100 us)

The upper exposure limit can be set in steps of 100us.

5.5.1.5.2 Exposure Max (x 100 us)

The maximum exposure limit can be set in steps of 100us.

5.5.1.5.3 Gain Upper (value = u8.8)

The upper gain limit can be set. Format u8.8.

e.g:

0x10 Gain = 1.0x

0x18 Gain = 1.5x

5.5.1.5.4 Gain Max (max = 2^{value} , s7.8)

The upper gain limit can be set. Format s7.8.

This value is logarithmic.

e.g.: gain max = 3.0 means max 8x gain.

5.5.1.5.5 BLC Window

Command	Code
BLC window X0 (0~128)	BLC (or ROI) x-start position
BLC window Y0 (0~128)	BLC (or ROI) y-start position
BLC window X1 (0~128)	BLC (or ROI) x-end position
BLC window Y1 (0~128)	BLC (or ROI) y-end position
BLC ratio (0~128)	Sets the ratio of the exposure algorithm uses between the ROI and the main picture.
BLC face Level (0~128)	Sets the amount of BLC level used for faces.
BLC face weight (0~128)	Sets the weight (or ratio) value between faces and ROI.
BLC ROI level (0~128)	Sets the amount of BLC level used for the ROI.

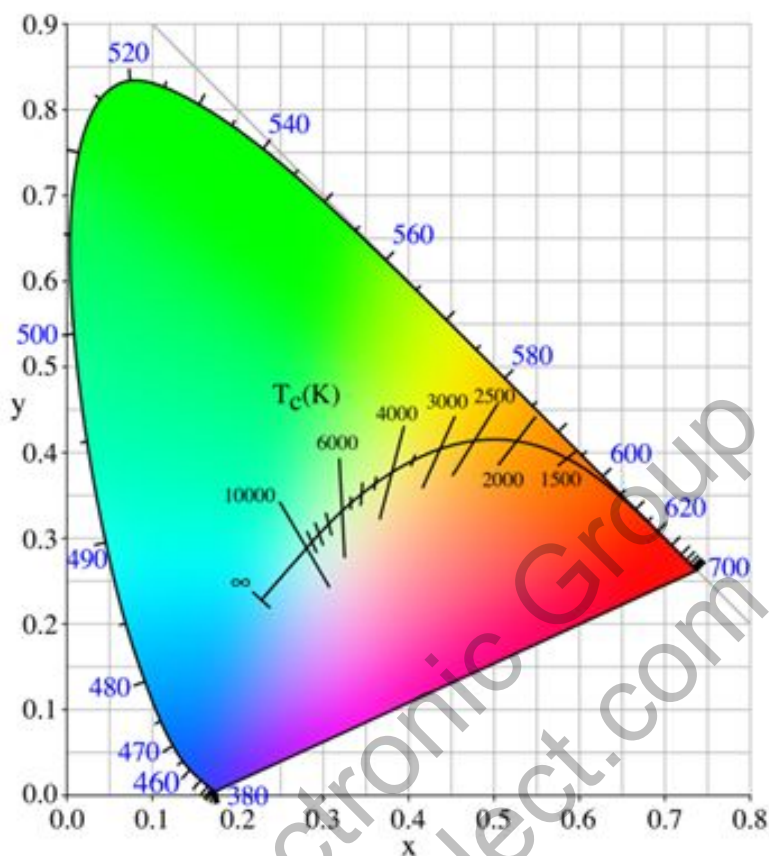
5.5.1.5.6 AWB manual white point (X, Y)

White balance must be set to manual!

White Balance, Auto & Preset must be set to manual!

AWB Manual X: Sets the X coordinate for the white point. From Blue to RED

AWB manual Y: Sets the Y coordinate for the white point. From GREEN to VIOLET



4 Figure 2 White point x,y graph

5.5.1.5.7 Store Registers (User Settings) to NVM

Set to store registers to non-volatile memory. After reboot or re-power, the stored register values will be used.

5.5.1.5.8 Restore Registers (User settings) from NVM

Set to restore the register values when register values have been changed by the user. Instead of rebooting or re-powering.

5.5.1.5.9 Restore to Factory Settings

Set to restore factory settings.

5.5.1.5.10 Reboot Camera

Set to reboot the camera.

i After restoring registers (user settings) from the NVM, the stream should be re-started to make sure the stream uses the restored settings. In case the **pyvidctrl** python tool was used also restart this tool. After a camera reboot, the stream should be re-started

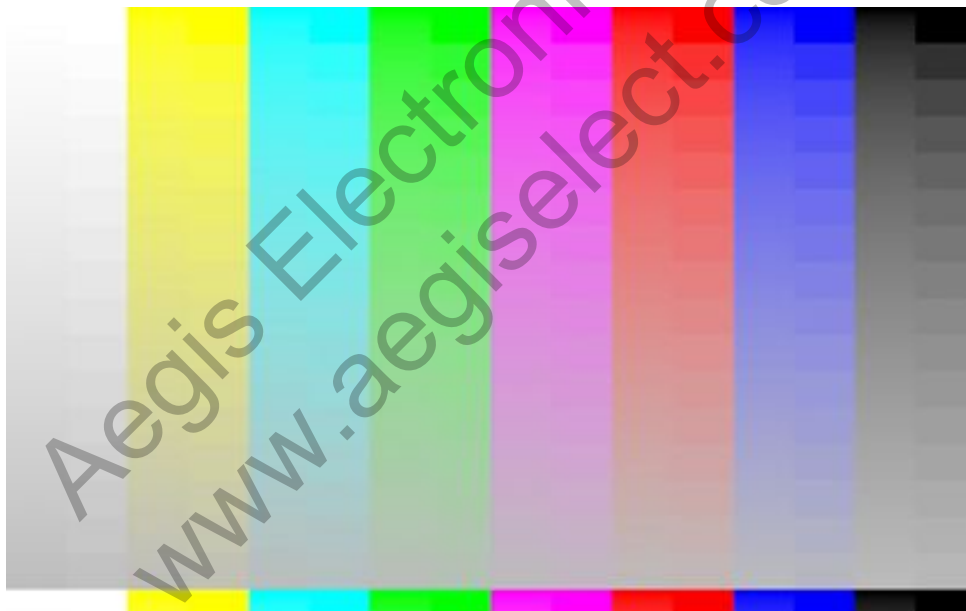
5.5.1.6 Image Processing Controls

Command	Code	Type	Controls
test_pattern	0x009f0903	menu	min=0 max=1 default=0 value=0 (Disabled)

Table 3 V4L2 image processing controls

5.5.1.6.1 Test pattern

Set value 0 to disable or 1 to enable.



5 Figure 3 Test Pattern

5.5.1.7 Detection Controls

Command	Code	Type	Controls
face_detection_enable	0x00a30932	bool	default=0 value=0

Command	Code	Type	Controls
face_detection_rectangles	0x00a30933	bool	default=0 value=0
face_detection_saturated	0x00a30934	bool	default=0 value=0
face_detection_speed_x_0_1s	0x00a30935	int	min=0 max=255 step=1 default=0 value=0 flags=slider
face_detection_theshold	0x00a30936	int	min=0 max=255 step=1 default=128 value=128 flags=slider
face_chroma_theshold	0x00a30937	int	min=0 max=31 step=1 default=12 value=12 flags=slider
face_minimum_size	0x00a30938	int	min=0 max=16384 step=1 default=1024 value=1024 flags=slider
face_maximum_size	0x00a30939	int	min=0 max=16384 step=1 default=16384 value=16384 flags=slider

Table 4 V4L2 detection controls

5.5.1.7.1 Detection controls description

Command	Description
Face detection (enable)	Set value 0 to disable or 1 to enable.
Face detection (rectangles)	Show rectangle around faces. Set value 0 to disable or 1 to enable.
Face detection (saturated)	Show saturated regions. Set value 0 to disable or 1 to enable.
Face detection speed (x 0.1s)	Sets the face detection speed in steps of 0.1 seconds.
Face detection threshold	Sets the face detection threshold.
Face chroma threshold	Sets the face detection threshold.
Face minimum size	Sets the minimum face size detection threshold.
Face maximum size	Sets the maximum face size detection threshold.

5.5.2 Dual camera stream using the SCAiLX-2GS234 Cameras



Please note that dual camera streaming is only supported from [Scailx release 0.6](#) onwards.

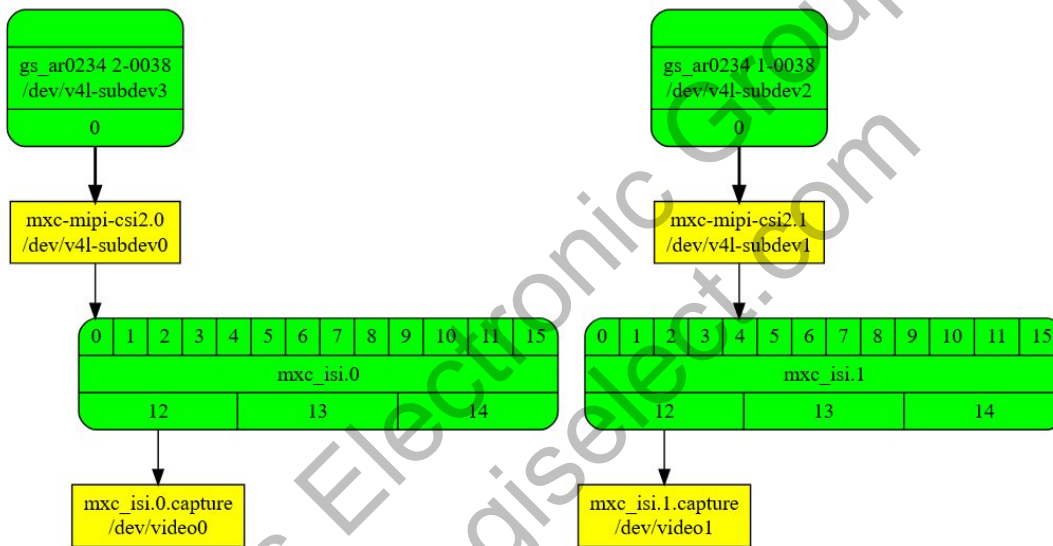
run media-ctl to verify that you have 2 cameras connected correctly:

```
media-ctl --print-dot
# verify the output by pasting it to https://dreampuf.github.io/GraphvizOnline
```

The result will be a picture as show below.

When two cameras are connected (in this case the SCAiLX-2GS234 cameras) the will be two capture device: One on `/dev/video0` and the other on `/dev/video1`. These will be used to start the rtsp server instances.

The `/dev/v4l-subdev3` and `/dev/v4l-subdev2` will be used to control the cameras using the v4l module for the GS234 camera.



5.5.2.1 Starting the RTSP server

Setting up 2 streams at 25fps.

Open a terminal and SSH into the SCAiLX camera, then type the following:

```
gst-variable-rtsp-server -p 9001 -u "v4l2src device=/dev/video0 ! \
video/x-raw,width=1920,height=1080,framerate=25/1 ! vpuenc_h264 bitrate=16000 ! \
rtph264pay name=pay0 pt=96"
```

Open a second terminal and SSH into the SCAiLX camera, then type the following:

```
gst-variable-rtsp-server -p 9002 -u "v4l2src device=/dev/video1 ! \
video/x-raw,width=1920,height=1080,framerate=25/1 ! vpuenc_h264 bitrate=16000 ! \
rtph264pay name=pay0 pt=96"
```

Note:

The framerate can be set upto 60fps: `framerate=60/1`

The bitrate can be set: `bitrate=64000` (for 64Mbit/s)

To use the HEVC encoder use: `vpucnc_hevc` and `rtph265pay`

5.5.2.2 Play the streams on a PC

Either use gstreamer for PC or VLC to play the stream. In this case we use gstreamer:

Open a PC console window then type the following to play the video0 stream using gstreamer :

```
gst-launch-1.0 rtspsrc location=rtsp://scailx-romke.local:9001/stream latency=0 \  
buffer-mode=auto ! decodebin ! fpsdisplaysink sync=false
```

Open a second PC console window then type the following to play the video1 stream using gstreamer :

```
gst-launch-1.0 rtspsrc location=rtsp://scailx-romke.local:9002/stream latency=0 \  
buffer-mode=auto ! decodebin ! fpsdisplaysink sync=false
```

Note:

For dual stream, each stream needs it's own port number. in this case: 9001 and 9002

5.5.2.3 V4L camera controls

Open a third terminal and SSH into the SCAILX camera, then type the following:

Only required when pyvidctrl is not already installed (on new firmware it is installed):

pip3 install git+https://github.com/antmicro/pyvidctrl

This will install a python-based GUI for the camera's control.

Type the start the GUI using v4l-subdev3 on the first camera :

```
pyvidctrl -d /dev/v4l-subdev3
```

Open a fourth terminal and SSH into the SCAILX camera, then type the following to control the second camera:

```
pyvidctrl -d /dev/v4l-subdev2
```

Note: the v4l-subdev numbers may not always be the same, always check using the `media-ctl` command.

5.5.2.4 Synchronize both cameras using one camera as master

To synchronize the second camera with the first camera the `sync_trigger` python script is used to setup both the cameras. The camera in CSI-0 will be master and generate the sync signal. Both cameras will synchronize to this sync signal.

To get the options type:

```
vdlg_ap1302 sync_trigger -h
```

To configure the camera's to be synchronized to a frame0rate of 25 fps with the camera to the CSI-0 mipi port as master of the trigger signal, type the following:

```
vdlg_ap1302 sync_trigger -c -f 25
```

To change the frame-rate to e.g. 31.6 fps type the following:

```
vdlg_ap1302 sync_trigger -f 31.6
```

Note: Either camera can provide the sync signal, however for now the `sync_trigger` python script only makes the camera connected to CSI-0 the sync master.

5.5.2.5 Synchronizing using a trigger GPIO from SCAILX

Both SCAILX-2GS234 cameras need to be configured to receive a trigger signal from SCAILX.

1. Edit the `/etc/modprobe.d/vid_isp_ar0234.conf` to use version 0.2 of the NVM (v0.1 uses camera's trigger signal on the connector, v0.2 uses SCAILX's internal trigger signal).
 - a. `/etc/modprobe.d/vid_isp_ar0234.conf`

b.

```
#
# This file contains the update version numbers and files for the vid_isp_ar0234
# camera
# NVM (Major.Minor) version numbers examples: 0x0001 = v0.1, 0x0203 = v2.3
# Make sure that version number and file name match!
# Version numbers of zero prevents updating
# Currently a maximum of 10 cameras are supported, but SCAILX IMX8 only 2 camera
# can be connected.
#
```

```
options vid_isp_ar0234 nvm_firmware_versions=0x0002,0x0002
nvm_firmware_names=SFT-23363_nvm_0.2.img,SFT-23363_nvm_0.2.img
```

2. Reboot or run `rmmod -f vid_isp_ar0234 && modprobe vid_isp_ar0234`
 - a. This will force an update of the NVM of the cameras to version 0.2 and make the both cameras listen to the internal trigger signal from SCAILX, when synchronizing to trigger is enabled as defined in
3. Start the streams on both cameras first.

```
gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw,width=1280,height=720,framerate=25/1 ! fpsdisplaysink
gst-launch-1.0 v4l2src device=/dev/video1 ! video/x-raw,width=1280,height=720,framerate=25/1 ! fpsdisplaysink
```

- a. The streams need to be started first, because starting the stream will initialize the cameras to their default or stored setting. Alternatively, sink the streams to `! perf ! fakesink` to get a printout of the FPS without seeing any video.
4. Configure to camera to use an external trigger signal. The `sync_trigger` python script includes a function to output a fixed frequency to the CSI trigger GPIOs:
 - a. Run:
 - b. `vdlg_ap1302 sync_trigger -c -p 36`
 - c. The `-c` parameter will freeze the sensor until a trigger signal is present on the sync inoput of the sensor. The `-p` parameter starts a loop which toggles the scailx trigger gpio using a timer.
5. Note the change in FPS.



Note:

- Steps (1) and (2) only need to be done once.
- Steps (3) and (4) need to be run any time a stream with GPIO trigger sync is needed.

SCAILX-2GS234 Sync and Trigger Configuration

5.5.3 SCAILX-2GS234 python scripts

The command `vdlg_ap1302` call pre programmed python scripts

```
vdlg_ap1302
Usage: ap1302py.main command [args] [--help]
Commands:
  readreg
  reboot
```

```
flashisp  
flashapp  
flashnvm  
status  
sync_trigger  
writereg
```

To get status from the camera:

```
vdlg_ap1302 status -i /dev/links/csi1_i2c  
MCU version = 0.29  
NVM version = 0.1  
ISP version = 443  
Camera Type = Color  
CAMERA_STATE = 4  
CAMERA_PASSWORD = 0  
MCU_TEMP = 0  
GET_SENSOR_TEMP = 37
```

5.5.3.1 SCAILX-2GS234 Sync and Trigger Configuration

- python function `vdlg_ap1302 sync_trigger` can be used to configure most sync/trigger configuration options. This is supplemental information for raw I2C commands which can accomplish the same result as documented in the 2GS234 data-sheet.

5.5.3.1.1 Setting up the sync (PWM) output signal on connector P200

- Set the duty cycle to 0.1 to 0.5
Write 0x3FFF to register 0X60 for a duty cycle of 0.25
- Set the Sync frequency.
- Register 0x64 value is $(1000_000/\text{fps}) * 256$
25fps = 0x9C4000
30fps = 0x823500
50fps = 0x4E2000
60fps = 0x411AAB

4. Enable the Sync signal
Write 0x0F to register 0x68

Note: Be aware that the maximum exposure time may need to be adjusted. If the exposure time exceeds the set sync frequency, then the exposure time will overrule the framerate set by the sync signal.

Python example

```
gsi2c.i2c = I2C("/dev/links/csi0_i2c") # set csi0_i2c
gsi2c.write32(0x64, round(256 * 1000_000 / 60)) # set fps to 60Hz
gsi2c.write16(0x60, 0x3FFF) # set duty cycle to 0.25
gsi2c.write8(0x68, 0x07) # enable sync (pwm)
```

5.5.3.1.2 Configuring the Trigger input signal.

```
# Write 0x302 to register 0x58
```

This will:

- Output a frame when sync signal is detected.
- Sync at the start of a frame.
- Reset the uptime every frame.
- Reset the frame counter at every frame

5.5.3.1.2.1 Python example

```
gsi2c.i2c = I2C("/dev/links/csi0_i2c") # set csi0_i2c
gsi2c.write16(0x58, 0x302) # wait for trigger on csi0

gsi2c.i2c = I2C("/dev/links/csi1_i2c") # set csi1_i2c
gsi2c.write16(0x58, 0x302) # wait for trigger on csi1
```

5.6 SCAILX-LVDS-2-MIPI zoom-block interface - Software

The SCAILX-LVDS-2-MIPI zoom-block interface has a videology provided kernel driver, which is autoloading upon detection.

The Driver supports no V4L2 settings, as the camera controls for the zoomblocks are configured via VISCA interface.

The SCAILX-LVDS-2-MIPI includes a I2C to serial converter to facilitate VISCA communication. From a V4L2 perspective, the driver reports only support for the currently detected resolution and framerate coming from the zoomblock.

5.6.1 Python Helper functions (as of Scaix SW release 0.10.3)

The driver includes the following python applications:

```
root@scaix-ai:~# vdlg-lvds-  
vdlg-lvds-getres vdlg-lvds-setres vdlg-lvds-visca
```

vdlg-lvds-getres:

print the resolution and frame rate zoom-block is outputting.

```
root@scaix-ai:~# vdlg-lvds-getres -d /dev/links/lvds2mipi_1  
1920 x 1080 @ 30.1
```

vdlg-lvds-setres:

send visca commands to set resolution.

Note that only Videology and select SONY zoomblocks are supported. Customers will have to write their own VISCA commands for other devices.

```
root@scaix-ai:~# vdlg-lvds-getres -d /dev/links/lvds2mipi_1  
1920 x 1080 @ 60.2  
  
root@scaix-ai:~# vdlg-lvds-setres -d /dev/links/lvds2mipi_1 720p30  
Camera ID response: 905000200466010003ff.  
Detected VIDEOLOGY zoomblock  
cmd: 8101042472000EFF, res: 9041ff90ff  
cmd: 81010424740000FF, res: 9041ff9051ff  
  
root@scaix-ai:~# vdlg-lvds-getres -d /dev/links/lvds2mipi_1  
1280 x 720 @ 30.1
```

Only supports **720p25 720p30 720p50 720p60 1080p25 1080p30 1080p50 1080p60**

vdlg-lvds-visca:

Send a visca packet over the i2c to serial interface, and wait for a response:

```
vdlg-lvds-visca -d /dev/links/lvds2mipi_1 81090002FF
```

5.6.2 TTY interface

The driver exposes a TTY (serial) interface for use with regular serial port applications, such as pyserial.



Its not advisable to use this interface. Its a polled driver, so any time the tty is in use the I2C bus will be flooded with messages. It also does not support standard tty capabilities like setting custom line-break characters, etc.

```
root@scaillx-ai:~# serial-xfer 9600 /dev/ttyVISCA1 81090002FF  
905000200466010003FF
```

5.7 Developing Software on Scaillx

5.7.1 Visual Studio Code Remote

SCAiLX devices come with SSH and is capable of running the vscode remote server.

Please read the following link for setting up Visual Code Studio and install the relevant extension:

<https://code.visualstudio.com/docs/remote/ssh>

- Open Visual Studio Code
- Press (F1 , Ctrl+Shift+P)
- Type: root@<hostname> SCAILX>
- Open directory /home/root of the device



5.7.1.1 Opening /home/root in Visual Code Studio

Example of accessing SCAILX device and working in the /home/root folder:



Note that vscode can take a lot of memory, especially when **enabling too many extensions**. Scailx implements RAM/CPU ulimits per session to ensure the system is not compromised by any one application taking too much resources. As a consequence, vscode may hang or terminate if its using too much ram. Its advisable to keep extensions to a minimum.
A less resource intensive coding UI option is [Zed](#), but it does not have all the features vscode has.

5.7.2 Modifying device-trees

From 0.10.3, Scailx includes the devicetree sources so customers can easily modify and rebuild the devicetrees to suite their hardware requirements without customising the Yocto linux build.

- In a device SSH terminal, go to `/usr/src/scailx-devicetrees`
- modify the base devicetree file, or create a new one which includes it.

```
nano /usr/src/scailx-devicetrees/src/arm64/scailx/scailx_karo_cameras.dts
```

- modify the file according, and compile the devicetrees by running `make all`

```
make all
```

- The makefile includes a convenience script to install the built devicetrees to `/boot/bspX/devicetrees`:

- `make scailx_install`



modifying devicetrees can cause a device to fail to boot. Do not modify the **ethernet** or **uart** sections without a debug-adaptor in case you need to recover using [uuu](#).
A safer option is to work with devicetree-overlays as described below.

5.7.2.1 Devicetree overlays via Configfs

Scailx uses [userspace devicetree overlay via configfs](#) to load different camera or hardware configurations on-the-fly. This is a convenient method of supporting differing hardware with a single image.

Loading devicetree overlays also offers a fault tolerant method for testing devicetree changes.



Note however, that this is **not a standard Linux kernel feature**. The kernel is not designed to read devicetree settings multiple times, and as such only **modules loaded after a devicetree overlay is applied** will be affected by its changes. Therefore built-in modules like those for GPIO, PGU, etc will not reflect the changes. In order to use such devicetree overlays, they would need to be applied from uboot as mentioned below.

to apply a dtbo file via configfs:

```
# make a folder for the overlay. name doesnt matter:
mkdir /sys/kernel/config/device-tree/overlays/my-overlay

# copy the overlay into the dtbo section:
cat my-overlay.dtbo > /sys/kernel/config/device-tree/overlays/my-overlay/dtbo
```

To always load the overlay, either create a systemd service for it, or add it to the `/storage/config/overlays` file.

5.7.2.2 Devicetree overlays from u-boot

to apply a devicetree from uboot before the kernel is loaded, change the uboot variable "overlays" to include the name of the overlay, and then place the overlay in the `/boot/bspX/devicetrees` folder.

```
# move overlay to bsp folder:
cp my-overlay.dtbo /boot/bsp0/devicetree/ && cp my-overlay.dtbo /boot/bsp1/devicetree/
```

```
# check the current value:
fw_printenv overlays

# set the variable to include our overlay:
fw_setenv overlays "my-overlay"
```

5.7.2.3 Check current devicetree

Which ever method was used, if the devicetree was applied, it will reflect in proc-fs.

```
# if the overlay includes a "chosen" field, it will show up:
ls /proc/device-tree/chosen/overlays/

# otherwise print the resultant devicetree and check if the fetures added are present:
dtc -I fs -O dts /proc/device-tree

# use grep to search it:
dtc -I fs -O dts /proc/device-tree | grep "my-feature"
```

5.7.3 Writing Rust or Go applications

5.7.3.1 Rust

 Rust support via packages as of Scailx 0.11.0

Installing rust via rustup doesn't work as Scailx devices have insufficient space.

Rather, install it from the Scailx repository.

```
opkg update
opkg install rust cargo
```

5.7.3.1.1 Hello world

This is the source code of the traditional Hello World program.

Create file **hello.rs** with editor (like nano): **nano hello.rs**

```
// Hello world application
```

```
// File: hello.rs

// This is the main function.
fn main() {
    // Statements here are executed when the compiled binary is called.

    // Print text to the console.
    println!("Hello World!");
}
```

Compile:

```
rustc hello.rs
```

Run:

```
./hello
Hello World!
```

5.7.3.2 Go

 Go support as of Scailx 0.11.0

The easiest way to get Golang to develop applications is to [follow the online instructions](#), but don't install to /usr/local as this folder is empty.

Installation of Go:

```
wget https://go.dev/dl/go1.23.5.linux-arm64.tar.gz
rm -rf /usr/go; tar -C /usr -xzf go1.23.5.linux-arm64.tar.gz
echo "export PATH=$PATH:/usr/go/bin:~/go/bin" > /etc/profile.d/go.sh
source /etc/profile.d/go.sh

go version
```

Test by installing a small application:

```
go install github.com/m1/go-generate-password/cmd/go-generate-password@latest
```

Run application:

```
go-generate-password  
^-Km@KEy5Hp?KHmHWgnV*T2+
```

5.7.3.2.1 Hello World

Create file **hello-world.go** with editor (like nano): **nano hello-world.go**

```
package main  
import "fmt"  
func main() {  
    fmt.Println("Hello world!")  
}
```

Run the application

```
go run hello-world.go  
Hello World!
```

Build the application:

```
go build hello-world.go
```

Run the executable:

```
./hello-world  
Hello World!
```

5.7.4 Build loadable module on device

It's possible to compile kernel-modules on the Scaix device, just as with any C++ application. Doing so however, requires some preparation:

- On the device:
 - Prepare for building kernel modules by running make-prepare

- ```
cd /usr/src/kernel
gunzip < /proc/config.gz > .config
make oldconfig
make prepare
make scripts
```

- Make a folder with your module source code, and add simple Makefile:

- ```
obj-m += my_module_file.o

EXTRA_CFLAGS += -DDEBUG
KERNEL_SRC ?= /usr/src/kernel

default:
    make -C ${KERNEL_SRC} M=$(CURDIR) modules
modules_install:
    make -C ${KERNEL_SRC} M=$(CURDIR) modules_install
clean:
    make -C ${KERNEL_SRC} M=$(CURDIR) clean
```

- from the that folder, run `make ; make modules_install`



For external drivers, note the kernel version for which they are valid. You can check the scailx kernel version by running `uname -a`.

Make sure your driver Makefile variables are in order, especially `KERNEL_SRC`

5.8 Yocto Embedded Linux Development

The Scailx embedded Linux is built using [Yocto](#) which builds the standard NXP Yocto Linux BSP with some Scailx hardware, software and kernel modifications.

The Scailx Yocto source is open-source, and hosted on [Github here](#). Customers are encouraged to customise the Scailx image to suite their needs or install their own custom software.

5.8.1 Using the Yocto SDK

Releases of Scailx are published together with their corresponding **Yocto SDK**. The Yocto SDK provides a cross-development tool-chain and libraries tailored to the contents of a specific Yocto image. This can be used to cross-compile software intended for the Scailx platform with minimal configuration. read more about how to use the Yocto SDK on the [Yocto documentation page](#).

note that it is also possible to compile simpler applications on device [Developing Software on Scailx](#)

5.8.2 Customising images or adding software

If multiple changes are required to suite a customer use-case, its advised to create a new Yocto layer to house the changes.

A convenient template project was created to facilitate customers making their own scailx modifications:

<https://github.com/VideologyInc/scailx-customer-bsp>

The template include 2 different configs:

- a basic config which is just a [kas](#) config file. Using this you can add a few common packages to the image or add a yocto define to the local.conf file.
- A customer layer config. This uses the folder "my-layer" as an additional layer over scailx and its layers, and allows customers to add their own recipes in this layer directory.

5.8.2.1 Modifying the template:

5.8.2.1.1 Simple config:

- remove the my-layer directory and my-layer.yml
- add packages to install in the image in the **simple.yml** file under **CORE_IMAGE_EXTRA_INSTALL**
- in a terminal, run
 - `kas shell`
 - Its advised to run the fetch-all first as some of the fetch tasks may fail due to github rate limiting.
 - `bitbake scailx-ml --runall fetch`
 - `bitbake scailx-ml`

5.8.2.1.2 Custom layer config:

- rename **my-layer** folder and **my-layer.yml** to your desired layer-name.
- (optionally) delete simple.yml and redirect the symbolic-link of .config.yml to your renamed file:
 - `ln -sf -T some-other-filename.yml .config.yml`
- find all references to "my-layer" in files (currently on in the layer.conf file and the yml file itself) and rename them to your desired layer-name.
- in a terminal, build the scailx-ml image (or your own image):
 - `kas shell`

- Its advised to run the fetch-all first as some of the fetch tasks may fail due to github rate limiting.
 - `bitbake scailx-ml --runall fetch`
- `bitbake scailx-ml` or `bitbake my-image`

Refer back the to [the Yocto documentation](#) for how to add recipes or modify existing ones.

5.9 Hailo integration

🔧 Support as of v0.11.0

SCAiLX supports accelerated workloads using Hailo8 PCIe modules. This allows for up to 20 TOPs of AI inference.

5.9.1 Verify setup

Check that the Hailo device is recognized:

Check kernel driver:

```
root@scailx-ai:~# ls /dev/hailo* && lsmod | grep hailo && lspci
/dev/hailo0
hailo_pci          90112  0
00:00.0 PCI bridge: Synopsys, Inc. DWC_usb3 / PCIe bridge (rev 01)
01:00.0 Co-processor: Hailo Technologies Ltd. Hailo-8 AI Processor (rev 01)
```

Check hailortCLI:

```
root@scailx-ai:~# hailortcli fw-control identify
Executing on device: 0000:01:00.0
Identifying board
Control Protocol Version: 2
Firmware Version: 4.19.0 (release,app,extended context switch buffer)
Logger Version: 0
Board Name: Hailo-8
Device Architecture: HAIL08
Serial Number: HLLWME0214600081
Part Number: HM218B1C2KA
Product Name: HAILO-8 AI ACCELERATOR M.2 A+E KEY MODULE
```

Check gstreamer plugins:

```
root@scaix-ai:~# gst-inspect-1.0 | grep hailo
hailo: hailodevicestats: hailodevicestats element
hailo: hailonet: hailonet element
hailo: synchailonet: sync hailonet element
hailotools: hailoaggregator: hailoaggregator - Cascading
hailotools: hailocounter: hailocounter - postprocessing element
hailotools: hailocropper: hailocropper
hailotools: hailoexportfile: hailoexportfile - export element
hailotools: hailoexportzmq: hailoexportzmq - export element
hailotools: hailofilter: hailofilter - postprocessing element
hailotools: hailogallery: Hailo gallery element
hailotools: hailograytonv12: hailograytonv12 - postprocessing element
hailotools: hailoimportzmq: hailoimportzmq - import element
hailotools: hailomuxer: Muxer pipeline merging
hailotools: hailonv12togray: hailonv12togray - postprocessing element
hailotools: hailonvalve: HailoNValve element
hailotools: hailooverlay: hailooverlay - overlay element
hailotools: hailoroundrobin: Input Round Robin element
hailotools: hailostreamrouter: Hailo Stream Router
hailotools: hailotileaggregator: hailotileaggregator
hailotools: hailotilecropper: hailotilecropper - Tiling
hailotools: hailotracker: Hailo object tracking element
hailotracers: bitrate (GstTracerFactory)
hailotracers: buffer (GstTracerFactory)
hailotracers: bufferdrop (GstTracerFactory)
hailotracers: cpuusage (GstTracerFactory)
hailotracers: detections (GstTracerFactory)
hailotracers: framerate (GstTracerFactory)
hailotracers: graphic (GstTracerFactory)
hailotracers: interlatency (GstTracerFactory)
hailotracers: numerator (GstTracerFactory)
hailotracers: proctime (GstTracerFactory)
hailotracers: queuelevel (GstTracerFactory)
hailotracers: scheduletime (GstTracerFactory)
hailotracers: threadmonitor (GstTracerFactory)
```

5.9.2 Run examples

The example scripts are provided in the **/opt/hailo/apps** directory.



Note that the demo scripts download Hailo models and resources from the Hailo model CDN, as these models are too large to store in the main OS update image. Thus these demos need an **active internet connection** the first time they are run.

5.9.2.1 Detection Demo

- go to detection demo folder:

```
cd /opt/hailo/apps/detection
```

- Run the demo script

```
./imx8_detection.sh  
# or to specify a sensor device:  
./imx8_detection.sh -i /dev/video-isi-csi0
```

- Open the Browser link printed from the script: <http://scailx-ai.local:8091> (default)

5.9.2.2 LPR Demo

- go to LPR demo folder:

```
cd /opt/hailo/apps/license_plate_recognition
```

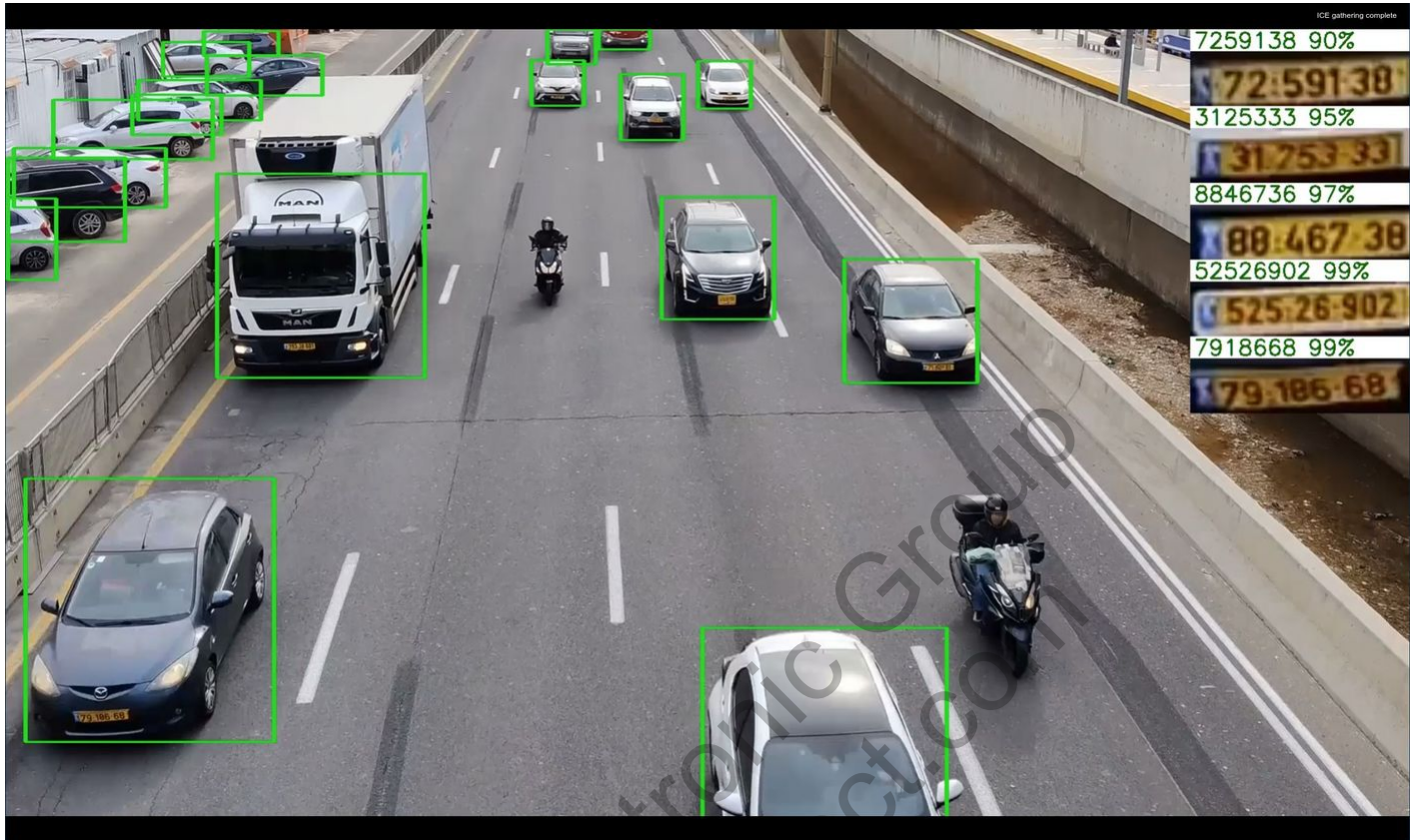
- The LPR demo can be run with a camera input (sensor) or a video file (.mp4) and is available as a python or bash script:

```
# Python script with sensor or video:  
./scailx_hailo_lpr.py /dev/video-isi-csi0  
./scailx_hailo_lpr.py ./resources/lpr.mp4  
  
# bash demo with sensor or video:  
./imx8_license_plate_recognition.sh --use-cam  
./imx8_license_plate_recognition.sh
```

- Open the browser link printed from the script: <http://scailx-ai.local:8091> (default)

```
root@scailx-ai:~# cd /opt/hailo/apps/license_plate_recognition/  
root@scailx-ai:/opt/hailo/apps/license_plate_recognition# ./scailx_hailo_lpr.py ./resources/lpr.mp4  
Running License Plate Recognition  
No g2d kernel cache file /tmp/g2d_openccl_kernel.bin  
Building g2d kernel ... buildBinarySize = 29831  
HTTP server started at http://scailx-ai.local:8091 and http://192.168.0.113:8091  
HTTP server started at http://scailx-ai.local:8091 and http://192.168.0.113:8091
```

Click on the link: <http://scailx-ai.local:8091>



the LPR demo is trained only for number plates with yellow background and only numbers 0-9.

5.9.3 Known issues

- Many Hailo online demos use OpenGL elements to scale or do colour conversion. These fail on Scaix without a display connected as the GPU is non-functional. The imx8 supports imxvideoconvert_g2d for most colour conversions and resizing.
- gststreamer element "hailonet" not working with direct memory. Use "synchailonet":
<https://community.hailo.ai/t/unexpected-hailo-driver-fail-36-errors-occurring-with-hailort-4-19/6495>

5.10 Wi-Fi Connectivity



SCAiLX supports Wi-Fi connectivity as of software release **0.11.0**.

SCAiLX supports the following Wi-Fi device without modifications:

- **High performance using SCAILX-WIFI board:**
 - Intel AX210
 - > 2Gbps in 802.11ax mode.



5.10.1 Verify wireless device is detected

Using `ifconfig -a` or `ip link`:

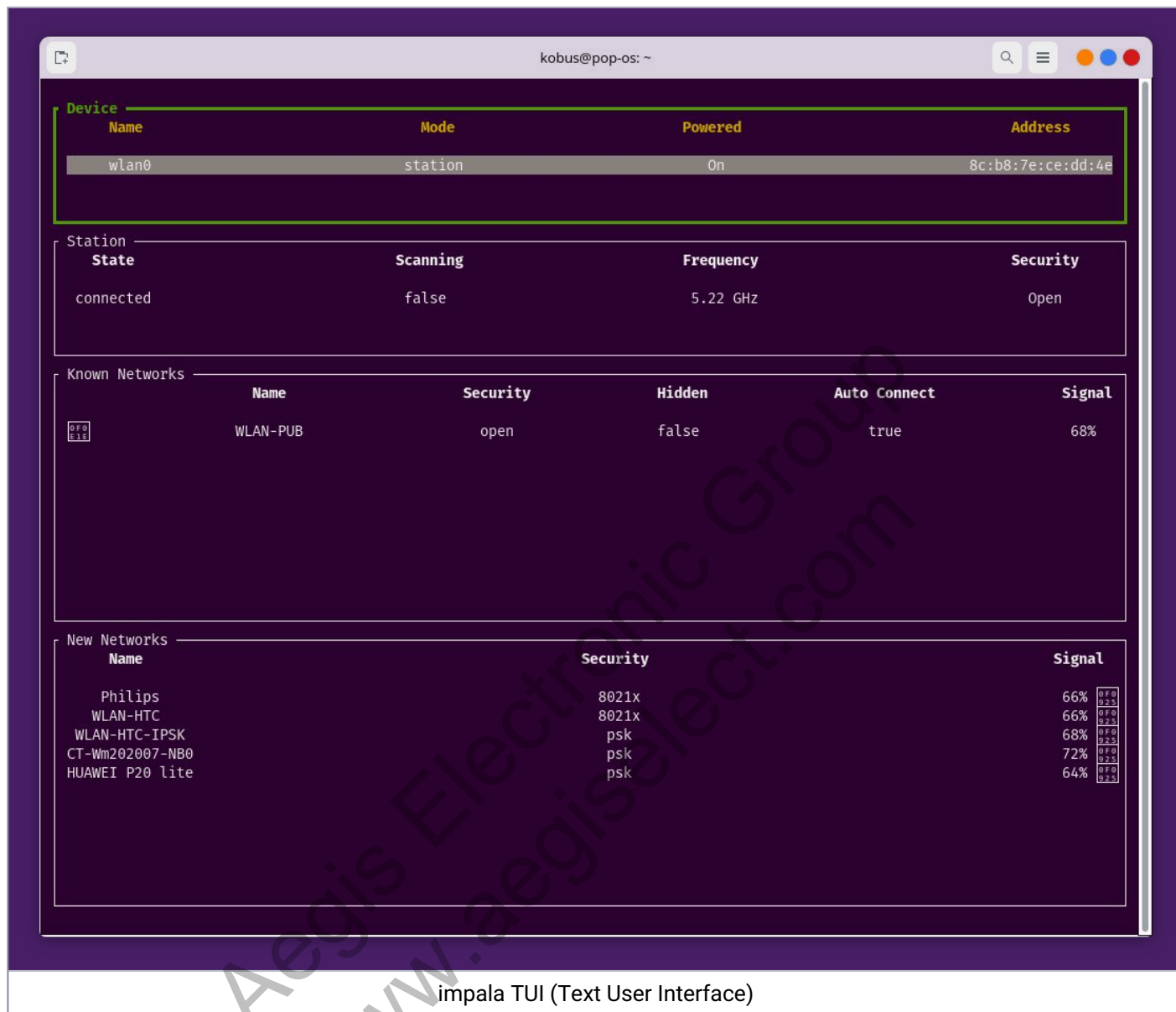
```
root@scaillx-ai:~# ip link
...
6: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DORMANT
group default qlen 1000
    link/ether 8c:b8:7e:ce:dd:4e brd ff:ff:ff:ff:ff:ff
```

5.10.2 Scan for access points using impala (a convenient Text User-Interface) on SCAILX

- i** Its advised to use your phone's **Wi-Fi hotspot** for initial testing. Corporate and home Wi-Fi access-points can have special authentication methods, proxies, etc.
First set your hotspot to "Open" to connect without a password just to verify that Wi-Fi is working.

Connect to to Access point:

```
impala
```



5.10.3 Using Impala:

- use tab to move down to the **new networks** block.
- use the up/down arrows to select an access point
- press **spacebar** to connect a network.

The system wireless network manager **iw** will remember this access-point credentials. (iwid stores config files under `/var/lib/iwd/`)

5.10.4 Check Wi-Fi status

```
root@scaix-ai:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.57.193.61 netmask 255.255.248.0 broadcast 10.57.199.255
    inet6 fe80::2e0:4cff:fec6:1bb9 prefixlen 64 scopeid 0x20<link>
    ether 00:e0:4c:c6:1b:b9 txqueuelen 1000 (Ethernet)
    RX packets 23547 bytes 4016803 (3.8 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 35 bytes 5650 (5.5 KiB)
    TX errors 0 dropped 41 overruns 0 carrier 0 collisions 0

root@scaix-ai:~# ping -I wlan0 google.com
PING google.com (172.217.168.238) from 10.57.193.61 wlan0: 56(84) bytes of data.
64 bytes from ams15s40-in-f14.1e100.net (172.217.168.238): icmp_seq=1 ttl=117 time=9.53 ms
64 bytes from ams15s40-in-f14.1e100.net (172.217.168.238): icmp_seq=2 ttl=117 time=6.14 ms
...
root@scaix-ai:~#
```

5.10.5 Network configuration

Once a wireless access-point has been connected to, it will **automatically set up in DHCP mode** to get an IP address from it. This is because scaix includes a systemd-networkd configuration file `/usr/lib/systemd/network/55-scaix-wifi.network` which configures any Wi-Fi networks for DHCP-client.

Customers can create their own systemd-network configurations to override this in either `/etc/systemd/network/` (persistent) or `/usr/lib/systemd/network/` (temporary between updates).

5.10.6 Further reading

It's possible to do much more with Wi-Fi, which is beyond the scope of this document. For example:

- connect to WPA-Enterprise access points
- setup ad-hoc and mesh Wi-Fi networks
- set static IP's
- Set router priority to prefer Wi-Fi over Ethernet
- Bridge Wi-Fi and Ethernet networks
- Configure vlans
- Create a Wi-Fi router/ access point with NAT and DHCP server

Refer to the official documentation for iw, iwd, and systemd-networkd to learn how to accomplish this. A good source of information and guides on this is [ArchWiki](#).

5.11 Package management

 Scailx supports Packagemanagement as of software release **0.11.0**.

The Scailx software Yocto Linux build supports package management using OPKG. A number of packages built by Yocto are available to install on Scailx devices which may not be part of the distributed systems.

installing packages requires an internet connection.

 As these packages are built by Yocto, not every Linux application will be supported, so the scope of available packages will be much less than one could expect from a traditional Linux distribution like apt for Debian or rpm for Fedora.

The packages include development libraries and convenience applications such as:

- htop for process monitoring
- Meson and ninja for developing gstreamer applications
- -dev packages for all installed libraries. This can help when compiling software on a Scailx device
- llvm and clang (includes clangd) for C++ LSP.
- Rust and Go for application development.

5.11.1 OPKG usage

5.11.1.1 Get fresh package list

```
opkg update
```

fetches the list of available packages from online.

5.11.1.2 Get the list of available packages

```
opkg list  
# search for package using grep. 'libglib' in this case  
opkg list | grep "libglib"
```

5.11.1.3 Get the list of installed packages

```
opkg list-installed
```

5.11.1.4 Install package

```
opkg install <package>
```

5.11.1.5 Remove package

```
opkg remove <package>
```



While package management is useful, it's not advisable as a long term solution as the packages take up a lot of space and will be cleared from the overlay when updating scailx. Also note that while it's possible to uninstall apps distributed in the official Scailx release, it will only be blanking the file-system overlay. it will not free up any space.

If You need to clear up space, use you can factor-reset the device.

5.11.2 Other package Managers

Apart from the official Scailx packages, it's possible to use **go**, **cargo** and **pip** to install go, rust or python applications.



If there are any Yocto provided applications you wish us to add to the **scailx** repository, please contact us.

6 Troubleshooting

6.1 Camera troubleshooting

6.1.1 Camera troubleshooting

If your camera **gstreamer** pipeline does not seem to work, a camera device may not be correctly configured, may not have been detected by the kernel, or its module may not have loaded correctly.

If for any reason a camera device is not working, its recommended to update to the latest SCAiLX release, and remove any user-modifications.

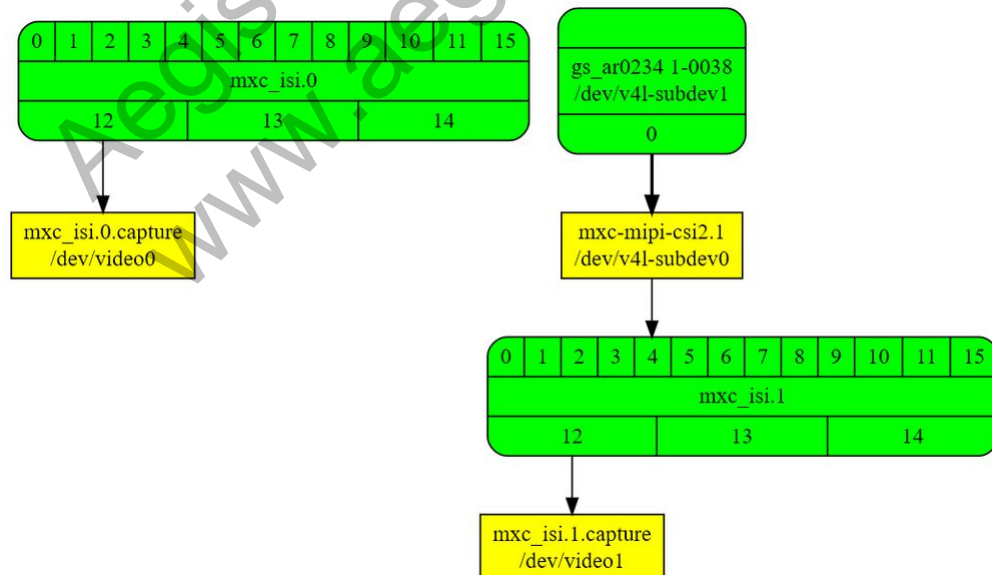
The following are a few trouble-shooting tips to determine the cause of the issue:

6.1.2 Verify the camera is connected and driver loaded

6.1.2.1 Print the V4L2 Connection diagram

```
media-ctl --print-dot
```

Paste the output into an online GraphViz viewer, like <https://dreampuf.github.io/GraphvizOnline/>.



6 media-ctl V4L2 connection diagram showing Videology GS-AR0234 Camera connected to /dev/video1

Rationale

If the camera is up and running, as a final step, the V4l2 device would be connected like above from an input device (the **sensor driver**, ar0234 in this case) through a pipeline device (Like **ISI** or **ISP**) to an output device like /dev/video*. If all this is working, the camera likely has no issues.

6.1.2.2 Check the /dev nodes for devices

```
ls -l /dev/v4* && ls -l /dev/video*
```

Rationale

This is another check to see if the kernel modules loaded the camera devices correctly. You should see the /dev/video0 or video1 device to read from, and a few. If the camera is up and running, as a final step, the V4l2 device would be connected like above from an

6.1.3 Check dmesg for camera kernel modules errors

```
dmesg -k
```

Rationale

It's possible that camera modules may have errors. Any kernel errors will show up red in the `dmesg -k` print (ignore the overlay warnings. Expected behavior).

6.1.4 Scan I2C bus for connected cameras

If everything else has failed, and you see no signs of a connected camera, you can check to see if there is indeed anything connected through the I²C bus.

```
# Scailx provides i2c-bus names for convenience
i2cdetect -y `cat /dev/i2c_names/csi0_i2c`
i2cdetect -y `cat /dev/i2c_names/csi1_i2c`
```

Rationale

Each camera is connected to an I²C bus. They should show "UU" in the I²C address graph to indicate that they have been pick-up by the corresponding kernel module. Otherwise the device I²C address is displayed. If no address is displayed but a camera is connected, it may be powered off. Its possible to power the cameras off and back on using the camera power control GPIO as shown in the terminal trace shown above.

Known I2C addresses for scailx cameras:

- 0x40, 0x43, 0x1c Videology LVDS to MIPI for use with Videology zoom-block cameras (released)
- 0x3c Low cost Omnivision ov5640 5MP sensor with built-in ISP (planned)
- 0x38 Videology AR0234 2MP global shutter with advanced ISP camera module (released)
- 0x36 Omnivision OS08A20 sensor with support of NXP IMX8M Plus VVCAM ISP (planned)

6.2 SCAILX-2GS234 LED Error codes

Status / Error	LED Code	Description
Startup	OFF	Camera is starting
Init	ON	Camera is initializing
Format Change	ON	Host set camera to formats change mode, MIPI lanes are disabled.
Format Change Done	OFF	Host sets camera to format change Done, MIPI is enabled.
Normal operating	ON (250ms) / OFF (1s)	Normal operation mode
Upgrade	ON	Camera is in upgrade mode
Power Down / Standby	ON (100ms) / OFF (3s)	Camera is in power down mode
Error	ON (250ms) / OFF (250ms)	Camera is in Error state.
Error / Debug	ON (1s) / OFF (250ms)	Camera is in Error or Debug state.

6.3 SCAILX reset to defaults

The main authentication of the SCAILX device is done via the SSK Key, as described under [Getting Started with SCAiLX](#).

6.3.1 Reset to factory defaults

To reset a device to factory defaults, just create a file in the storage driver called `clear_overlay` and reboot.

```
touch /storage/clear_overlay
```

This will clear the overlay of any applied changes.

Alternatively, is possible to create this file, and then apply an update from swupdate [Software Update](#) to install a fresh image without changes.

6.3.2 Change password

In order to reset the root user password, it is necessary to log in via SSH first.

-  if the customer forgot his/her password of the SCAiLX device, access can only be established with the device use of the SSH Key as described here:
[Getting Started with SCAiLX](#)

Create a new password via the `passwd` command. This will prompt you for entering a new password, it will not ask for the current password.

Aegis Electronic Group
www.aegiselect.com



More Than One Million Embedded Cameras

Designed and Delivered Since 1995.

Our Brand Difference

Our deep commitment to the customer experience delivers performance excellence throughout the entire customer journey. This is Videology's brand difference and it's our company's most important priority in serving the needs of our customers across the globe.

Our Brand Promise

How do we support our brand difference? We do so with a sincere promise we make to every Videology customer as follows: We provide competence, attention to detail and personal care with a level of excellence that will delight every customer in every interaction. This is Videology's brand promise and it's been the key to our growth and success – from a small start-up more than 25 years ago to a global leader in today's imaging industry.



Videology®

Industrial-Grade Cameras

an inTEST Company

HEADQUARTERS LOCATION

Videology Industrial-Grade Cameras
35 Hampden Road
Mansfield, MA 02048 United States
Tel: +1 401 949 5332 | Fax: +1 401 949 5276
sales@videologyinc.com

EUROPE LOCATION

Videology Industrial-Grade Cameras
High Tech Campus 5
5656 AE Eindhoven, The Netherlands
Tel: +31 40 7200159
sales-eu@videologyinc.com



www.videologyinc.com

Excellence.
Every day. Every time.