

**NVIDIA Jetson Camera Software Development Guide**

**1. Camera Architecture Overview**

NVIDIA Jetson's camera software stack provides a flexible framework for developing camera solutions, supporting both NVIDIA ISP-enabled sensors and third-party cameras. Key components include:

**Core Components**

* **libargus**: Low-level API for direct camera control (exposure, focus, ISP tuning).
* **nvarguscamerasrc**: GStreamer plugin leveraging ARGUS API for ISP processing (e.g., Bayer → YUV conversion).
* **V4L2 Framework**: Standard Linux interface for non-ISP cameras (USB/YUV sensors).

**Supported Camera Types**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Interface | ISP Usage | Example Sensors |
| Bayer | CSI | Yes (via Jetson ISP) | OV5693 (NVIDIA reference) |
| YUV | CSI | No | IMX185 (V4L2 driver) |
| USB | USB | No | UVC-compliant cameras |

**2. Sensor Driver Development**

**V4L2 Driver Framework**

Two versions available for driver development:

**Version 1.0 (Legacy)**

* Direct IOCTL calls to V4L2 kernel subsystem.
* Example: imx185\_v1.c driver.
* **Use Case**: Simple sensors without advanced features.

**Version 2.0 (Recommended)**

* Modular design using Jetson V4L2 Camera Framework.
* Encapsulates common code (I2C/register access).
* Example: imx185.c driver.
* **Advantages**:
  + Easier maintenance
  + Better integration with NVIDIA ISP
  + Supports complex sensors

**3. Device Tree Configuration**

**Camera Module Definition**

Add entries to tegra-camera-platform node in device tree:

tegra-camera-platform {  
 compatible = "nvidia,tegra-camera-platform";  
 modules {  
 module0 {  
 badge = "imx185\_bottom";  
 position = "bottom";  
 orientation = "0";  
 drivernode0 {  
 pcl\_id = "v4l2\_sensor";  
 devname = "imx185 30-001a";  
 proc-device-tree = "/proc/device-tree/i2c@3180000/tca9546@70/i2c@0/imx185\_a@1a";  
 };  
 };  
 };  
};

**Key Properties**

* badge: Unique module identifier (format: <sensor>\_<position>)
* position: Physical placement ("bottom", "front", etc.)
* orientation: Sensor rotation (0°, 90°, etc.)

**4. Kernel Configuration**

**Essential Steps**

1. **Enable Media Controller**:

CONFIG\_MEDIA\_CONTROLLER=y  
CONFIG\_VIDEO\_DEV=y

1. **Add Sensor Support**:

CONFIG\_VIDEO\_IMX185=y

1. **Build & Install**:

make -j$(nproc) modules  
sudo make modules\_install

**5. Driver Implementation**

**V4L2 2.0 Driver Structure**

struct imx185 {  
 struct v4l2\_subdev subdev;  
 struct media\_pad pad;  
 struct v4l2\_ctrl\_handler ctrl\_handler;  
 struct regmap \*regmap;  
 // Sensor-specific data  
};

**Key Functions**

* imx185\_probe(): Initializes I2C communication
* imx185\_s\_stream(): Starts/stops video streaming
* imx185\_set\_format(): Configures resolution/format

**6. Verification & Testing**

**Basic Validation**

1. Check device detection:

dmesg | grep -i imx185

1. Verify media controller links:

media-ctl -p -d /dev/media0

**Capture Test**

# Capture RAW Bayer data  
v4l2-ctl --set-fmt-video=width=1920,height=1080,pixelformat=RG10 \  
 --stream-mmap --stream-count=1 -d /dev/video0 \  
 --stream-to=capture.raw

**7. GStreamer Integration**

**ISP-Enabled Pipeline (Bayer Sensor)**

gst-launch-1.0 nvarguscamerasrc ! \  
 'video/x-raw(memory:NVMM), width=1920, height=1080, format=NV12' ! \  
 nvoverlaysink

**V4L2 Pipeline (YUV Sensor)**

gst-launch-1.0 v4l2src device=/dev/video0 ! \  
 'video/x-raw, format=UYVY, width=640, height=480' ! \  
 nvvidconv ! 'video/x-raw(memory:NVMM)' ! nvoverlaysink

**8. Advanced Features**

**Infinite Timeout Support**

For triggered cameras:

sudo enableCamInfiniteTimeout=1 nvargus-daemon

**ISP Configuration**

* Tuning files located in /var/nvidia/argus/
* Modify OV5693.xml for reference sensor tuning

**9. Partner Ecosystem**

NVIDIA collaborates with certified partners for:

* Custom sensor characterization
* ISP tuning services
* Optical calibration
* **Preferred Partners**: See [Jetson Partner Supported Cameras](https://developer.nvidia.com/embedded/jetson-partner-supported-cameras)

**Key Takeaways**:

1. Use V4L2 2.0 framework for new sensor development
2. Leverage ARGUS API for ISP-enabled cameras
3. Validate through media controller and v4l2-ctl
4. Partner with NVIDIA-certified vendors for complex integrations

*For full API details, refer to the* ***Jetson Linux API Reference in the official documentation.***

⁂