Low Latency Video Streaming with the Smart Radio and Nvidia Jetson Nano

Introduction

This document is a basic tutorial for how to get started with the Doodle Labs Smart Radio and the Nvidia Jetson Nano in a video streaming application. The Nvidia Jetson Nano is a popular System-on-Module (SOM) used for emerging IoT applications such as drones, robots, and generally devices which can make use of its powerful AI capabilities. It includes a GPU which can perform fast H.264/H.265 video encoding and decoding making it ideal for low latency video streaming. This tutorial is divided into the following sections:

1. Video System Block Diagram
2. Hardware Setup
3. Getting the firmware and first time configuration
4. Smart Radio Configuration
5. Preparing GStreamer
6. Video streaming

This tutorial makes use of the gstreamer command-line tools, gst-launch-1.0 and gst-inspect-1.0. Building a C-application is beyond the scope of the tutorial.

Video System Block Diagram

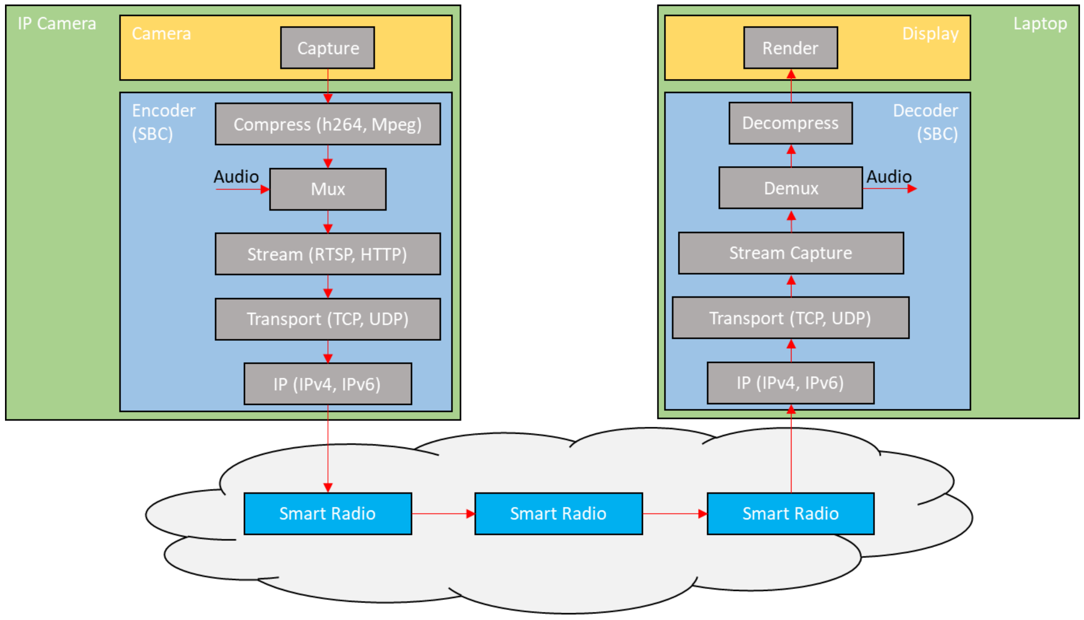
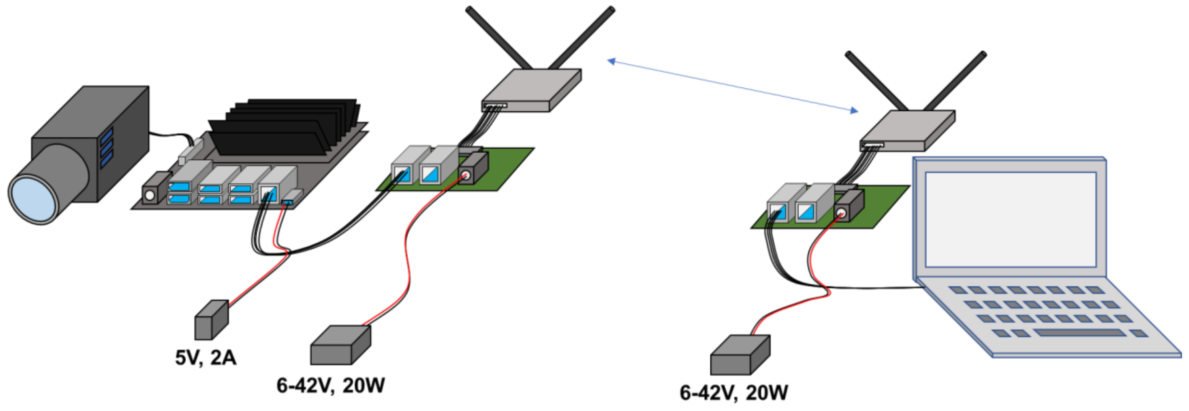


Figure 1 – Video Data Flow

The block diagram above illustrates the data flow path of a simple streaming application. The cloud of Smart Radios represents a mesh situation where the video feed may hop across a node before arriving at the destination Smart Radio. For a discussion of the different blocks , please see the document, “Smart Radio Video Streaming Tutorial”.

Hardware Setup

This tutorial makes use of the Nvidia Jetson Nano Developer Kit which has standard interfaces like USB, HDMI etc. Basic hardware setup for streaming is shown below.



Power

The Jetson Nano can be powered over the standard USB Micro device port (J28) or the barrel connector (J25). Use a 5-V supply with at least a 2-A current rating. The camera is powered over the MIPI CSI2 interface. The Smart Radio requires 20W supplies. The voltage range of the Smart Radio power supply is 6-42 V.

Camera Setup

We recommend using the MIPI CSI2 interface for the camera, however if desired a USB camera could also be used. USB 2.0 has a maximum link speed of 480 Mbps which is unsuitable for raw 1080p30 video, which means that you will need at least a USB 3.0 interface. Note that the HDMI port on Nano is an output port, so you cannot connect an HDMI camera to it. For more discussion on video codec and data rate, please refer to our companion application note – Video Streaming Tutorial

First Time Jetson Nano Setup

The first time you use the Jetson Nano, you may find it easier to get it setup using a monitor and keyboard. Hook up your monitor to the HDMI port, and keyboard/mouse to the USB ports.

Getting the Firmware

There are extensive guides available on the Nvidia website which detail how to prepare your Jetson Nano. The basic steps are:

1. Download and extract the latest firmware image. In this tutorial, we used JP 4.4.

2. Format your SD card and burn the firmware image to the SD card using a program such as Balena Etcher.
3. Inset the SD card into the Jetson Nano, and power up.

The first time you boot up the Jetson Nano, you will be asked to setup a username and password, and these will be required for SSH or Serial access later.

By default, the Jetson Nano is setup as a DHCP client. Therefore, you can connect it to your office router so that it can get an IP address and access the internet. Once you have connected to the internet, run

```
$ sudo apt update
$ sudo apt upgrade
```

This will make sure that your package lists and packages are up to date. After that make sure that your Jetson clock is synchronized. This is necessary to access secure websites.

```
$ timedatectl
                      Local time: Tue 2020-05-05 16:23:05 +08
                  Universal time: Tue 2020-05-05 08:23:05 UTC
                        RTC time: Tue 2020-05-05 08:23:06
                       Time zone: Asia/Singapore (+08, +0800)
       System clock synchronized: yes
systemd-timesyncd.service active: yes
                 RTC in local TZ: no
```

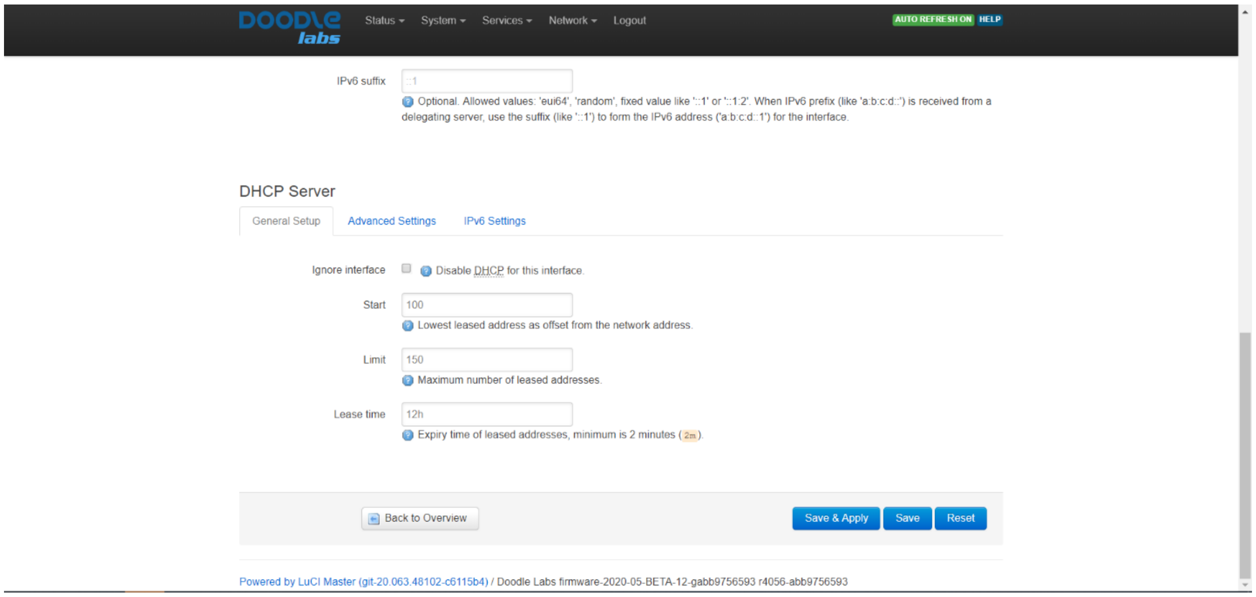
After the initial setup is complete, connect the Jetson Nano to the Smart Radio as shown in the hardware diagram.

Smart Radio Configuration

In this section, we describe simple steps to setup the Smart Radio for video streaming. If you are not sure how to make any of these configuration settings, then please consult the Smart Radio Configuration Guide.

IP Configuration

The Jetson Nano is pre-configured as a DHCP client. However, by default, the Smart Radio does not have the DHCP server enabled. So, we will simply enable the DHCP server on one of the Smart Radios. The DHCP server should be enabled on the WAN2 interface. In the GUI, navigate to `Network → Interfaces` and `EDIT` the `WAN2` interface. Wait for the configuration page to load, then scroll to the bottom and click `Setup DHCP Server`. The default settings are ok, so just click `Save & Apply`. At this point you should make sure all nodes on the network are setup as DHCP clients. This is true by default for the Smart Radio which are DHCP clients, but also have a fixed IP address.



IPv6 suffix ::1
Optional. Allowed values: 'eui64', 'random', fixed value like '::1' or '::1:2'. When IPv6 prefix (like 'a:b:c:d::') is received from a delegating server, use the suffix (like '::1') to form the IPv6 address ('a:b:c:d::1') for the interface.

DHCP Server

General Setup | Advanced Settings | IPv6 Settings

Ignore interface ☐ Disable DHCP for this interface.

Start 100
Lowest leased address as offset from the network address.

Limit 150
Maximum number of leased addresses.

Lease time 12h
Expiry time of leased addresses, minimum is 2 minutes (2m).

Back to Overview | Save & Apply | Save | Reset

Powered by LuCI Master (git-20.063.48102-c6115b4) / Doodle Labs firmware-2020-05-BETA-12-gabb9756593 r4056-abb9756593

You can find the IP addresses of all nodes on the network by logging into the Smart Radio over SSH and using the address resolution protocol. You can identify connected devices by their hardware address.

```
root@smartradio-301a4e2006:~# arp
IP address       HW type     Flags       HW address            Mask     Device
192.168.1.10     0x1         0x0         b0:25:aa:2d:d3:8e     *        br-lan
10.223.0.179     0x1         0x2         00:04:4b:e6:b4:76     *        br-wan
10.223.0.159     0x1         0x2         b0:25:aa:2d:d3:8e     *        br-wan
10.223.0.182     0x1         0x2         00:30:1a:4e:20:0a     *        br-wan
10.223.0.40      0x1         0x0         b0:25:aa:2d:d3:8e     *        br-wan
```

Note that the above steps apply for both WDS AP/Client and Mesh modes.

Setup Steps

1. Switch to WDS AP/Client Mode. This is the recommended mode for point-to-point or star networks. The AP should be at the center of the star network, but in a point-to-point network, you may find it more convenient having the AP as the remote device. Note that all traffic must pass through the AP. The easiest way to switch to WDS AP/Client mode is using the simpleconfig menu in the web GUI.
2. In the AP, turn on the DHCP server on WAN2.
3. Non WiFi-Compliant Mode. This step only applies to RM-2450 model to avoid the interference from other WiFi nodes in the area. You can switch to any channel bandwidth other than 20/40 MHz to avoid WiFi compliance. This mode tends to lead to a more stable latency.
4. In the Xtreme models, reduce the distance setting to the maximum required in your application. This setting is found in the advanced tab of the wireless configuration.
5. Keep the TX Power in Auto mode. This provides the best range.
6. Make sure that the Smart Radio link is able to support the required throughput for the desired video resolution. Iperf can be used to measure throughput. We suggest to keep an extra 50% headroom for the interference margin. Refer to Table 1 for throughput requirements.
7. In the differentiated services UI page, enable video optimization, and add a new rule to send packets originating from the IP address of the IP camera to the video queue (screenshot below). In our

example, we are simply putting all traffic coming from the IP camera in the video queue. It is possible to filter traffic by the IP addresses, protocol, or port number.

8. Note that Doodle Labs Mesh Rider uses special radio parameters for this queue to optimize the video transmission over wireless medium in high interference areas.

Preparing GStreamer

GStreamer is a framework for creating multimedia streaming applications available in multiple platforms including Windows, iOS, Android, and Linux. Extensive documentation is available online. GStreamer is installed in the Jetson Nano by default, but in order to stream using rtsp, you either need to write your own application, or use gst-rtsp-server. gst-rtsp-server requires the gtk-doc-tools package to be installed.

```
$ sudo apt install gtk-doc-tools
```

In order to use gst-rtsp-server, you need to clone the repository, checkout the version of gst-rtsp-server suitable for your GStreamer version, and the build the application. Start by creating a working directory.

```
$ mkdir workingDir
$ cd workingDir
$ git clone https://github.com/GStreamer/gst-rtsp-server.git
$ cd gst-rtsp-server
$ gst-launch-1.0 -version
GStreamer 1.14.5
https://launchpad.net/distros/ubuntu/+source/gstreamer1.0
```

The version of GStreamer we have is 1.14.5, so checkout the corresponding git branch.

```
$ git checkout 1.14.5
$ ./autogen.sh
$ ./configure.sh
$ make
$ sudo make install
```

gst-rtsp-server is now ready to be used.

Video Streaming

We will demonstrate video streaming for a few different setups. Before starting a video stream, first get the information on your video camera's capabilities. You can list the cameras attached to the Jetson Nano and check their capabilities using `v4l2-ctl`.

```
$ v4l2-ctl --list-devices
vi-output, imx219 6-0010 (platform:54080000.vi:0):
        /dev/video0
$ v4l2-ctl -d /dev/video0 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'RG10'
        Name        : 10-bit Bayer RGRG/GBGB
                Size: Discrete 3264x2464
                        Interval: Discrete 0.048s (21.000 fps)
                Size: Discrete 3264x1848
                        Interval: Discrete 0.036s (28.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.017s (60.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.017s (60.000 fps)
```

In our case, we have one camera which is attached and it is exposed to the user as /dev/video0. Our tests will be conducted using 1920x1080 at 30 fps. Fast encoding at H.264 can be accomplished using the omxh264enc plugin. You can see details and options of the omxh264enc plugin using

```
$ gst-inspect-1.0 omxh264enc
```

The output is very long and is not shown. The equivalent H.265 encoder plugin is omxh265enc. Nvidia has an Accelerated GStreamer User Guide available online which details some of the capabilities of the Jetson Nano when used with GStreamer.

RTSP Streaming

RTSP streaming can be started using

```
$ ./gst-rtsp-server/examples/test-launch "nvarguscamerasrc ! video/x-
raw(memory:NVMM) width=1920 height=1080 framerate=30/1 format=NV12 ! omxh264enc
iframeinterval=15! h264parse ! rtph264pay name=pay0 pt=96"
```

Note that the command points to the gst-rtsp-server directory which was cloned earlier. To encode with H.265, change omxh264enc to emxh265enc, h264parse to h265parse, and rtph264pay to rtph265pay.

The stream can be picked up on the receiving PC using

```
$ gst-launch-1.0 -v rtspsrc location=rtspt://<IP Address>:8554/test !
application/x-rtp, payload=96 ! rtph264depay ! avdec_h264 ! videoconvert !
autovideosink sync=false
```

where <IP Address> is the IP address of the Jetson Nano. Again, to decode using H.265, change rtph264depay to rtph265depay and avdec_h264 to avdec_h265. In the command above we use TCP as the transport protocol. To use UDP, the “location=rtspt://” part should be changed to “location=rtsp://”.

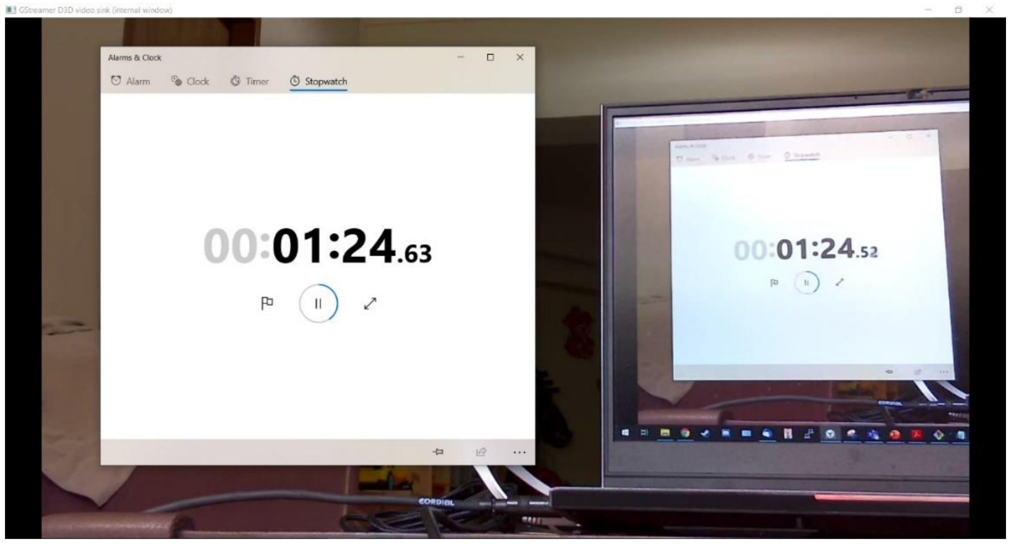
Results

With the above settings, the glass-to-glass latency was typically 110ms with only about 10ms for the transport through the Smart Radios.

The difference between TCP and UDP was around 3 ms.

Note that Doodle Labs Mesh Rider uses special radio and parameters to optimize the video transmission over wireless medium in high interference areas. For video transmission within Smart Radio private network, we recommend use of TCP and this video queue.

A screen shot of a typical output is shown below.


GStreamer D3D video sink (internal window)
Alarms & Clock
Alarm
Clock
Timer
Stopwatch
00:01:24.63
00:01:24.52