# ROScube X BSP Quick Start Guide

Document Generated Date : 2021-04-01
Document Released for: copyright

*More information are available at* ADLINK Technology Inc. Advanced Robotic Platform Group

# Preface

## Copyright

Copyright © 2021ADLINK Technology, Inc. This document contains proprietary information protected by copyright. All rights are reserved. No part of this manual may be reproduced by any mechanical, electronic, or other means in any form without prior written permission of the manufacturer.

## Disclaimer

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer. In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

## Environmental Responsibility

ADLINK is committed to fulfill its social responsibility to global environmental preservation through compliance with the European Union's Restriction of Hazardous Substances (RoHS) directive and Waste Electrical and Electronic Equipment (WEEE) directive. Environmental protection is a top priority for ADLINK. We have enforced measures to ensure that our products, manufacturing processes, components, and raw materials have as little impact on the environment as possible. When products are at their end of life, our customers are encouraged to dispose of them in accordance with the product disposal and/or recovery programs prescribed by their nation or company.

## Trademarks

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

# TABLE OF CONTENTS

CHAPTER

# ONE

# REVISIONS

Document version: 1.0.7

Compatible with ROScube-X BSP version:

- L4T-32.4.3-Kernel-1.0.6
- L4T-32.4.4-Kernel-1.0.7
- L4T-32.5.0-Kernel-1.0.8
- L4T-32.5.1-Kernel-1.0.8

| Author | Revision |
|---|---|
| Chester Tseng | L4T-32.4.3-Kernel-1.0.6 |
| Chester Tseng | L4T-32.4.4-Kernel-1.0.7 |
| Chester Tseng | L4T-32.5.0-Kernel-1.0.8 |
| Chester Tseng | L4T-32.5.1-Kernel-1.0.8 |

## 1.1 BSP MFI Image Naming Rule

ROScube-X BSP MFI file follows the naming rule.

```
mfi_<model name>_<filesystem and distro type>_<L4T version>-Kernel-<kernel version>.
tbz2
```

For example: `mfi_rqx580_nvidia-ubuntu-rootfs-bionic_L4T-32-4-4-Kernel-1-0-7.tbz2`.

### 1.1.1 Model Name

- `rqx580` - ROScube-X without GMSL feature.
- `rqx58g` - ROScube-X with GMSL feature.

### 1.1.2 FileSystem Type

- `nvidia-sample-rootfs` - Nvidia provided sample filesystem. It is based on Ubuntu 18.04
- `custom-slim-rootfs-bionic` - Custom rootfs build from Ubuntu base image 18.04.5. This file type is much smaller.
- `custom-slim-rootfs-focal` - Custom rootfs build from Ubuntu base image 20.04.1. This file type is much smaller.

### 1.1.3 L4T Version

The L4T version, for example 32.4.3, 32.3.4, 32.5.0 or 32.5.1

### 1.1.4 Kernel Version

ROScube-X kernel and device tree configuratiion version, which is affected by kernel and device tree from ADLINK's internal engineering development.

## 1.2 Release Note

### 1.2.1 L4T-32.4.3-Kernel-v1.0.6

- First release RQX-580, RQX-58G BSP image.

### 1.2.2 L4T-32.4.4-Kernel-v1.0.7

- Upgrade to L4T 32.4.4

### 1.2.3 L4T-32.5.0-Kernel-v1.0.8

- Upgrade to L4T 32.5.0
- Pre-install roscube-init package.

### 1.2.4 L4T-32.5.1-Kernel-v1.0.8

- Upgrade to L4T 32.5.1

CHAPTER

TWO

# IMAGE RECOVERY / FLASHING

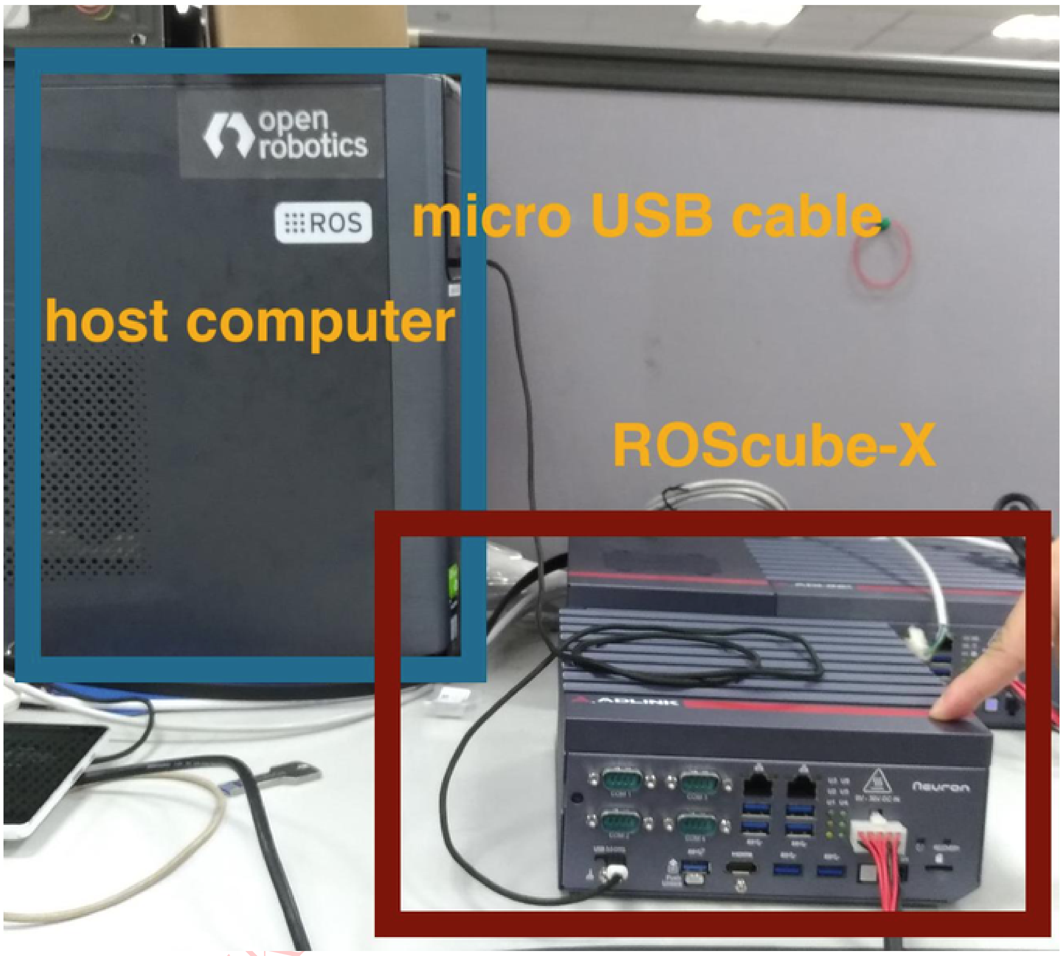
## 2.1 Prerequisite

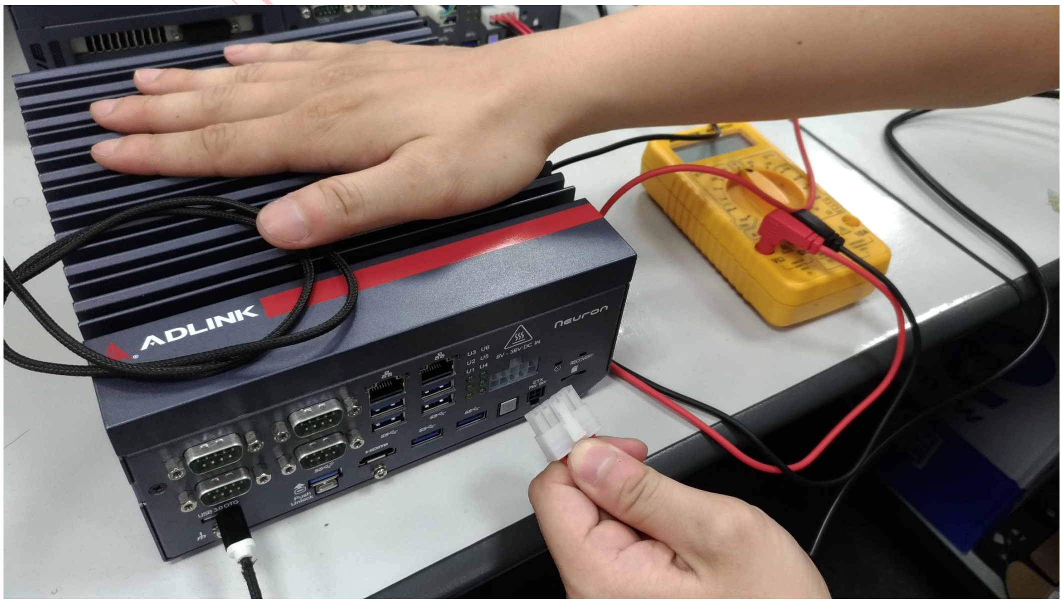
Before flashing image to ROScube, you should prepare the following items:

- **Host PC** with **Ubuntu 18.04 or 20.04** operating system.
- A good quality micro-usb for connecting to ROScube.

## 2.2 Procedures

### 2.2.1 1. Connect power cable to ROScube and power on it.

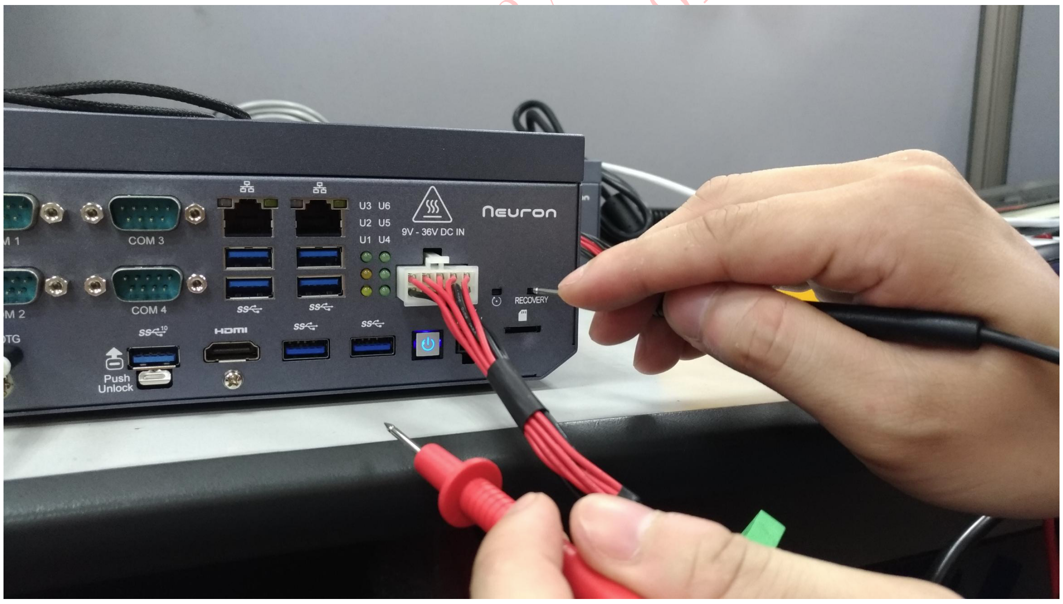




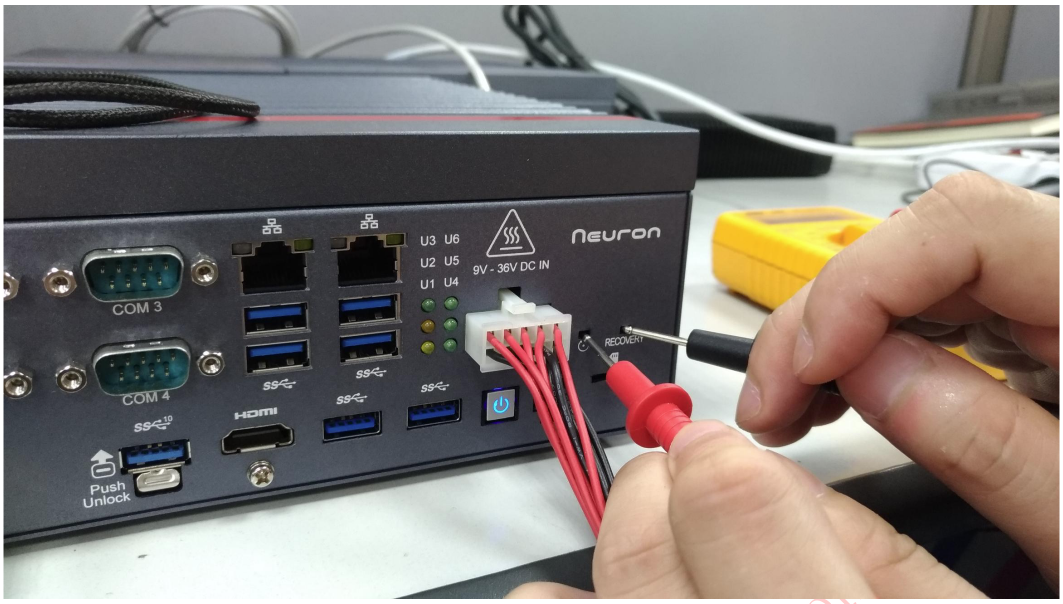
### 2.2.2 2. Make sure Micro USB cable is in good quality.

### 2.2.3 3. Reset to recovery mode.

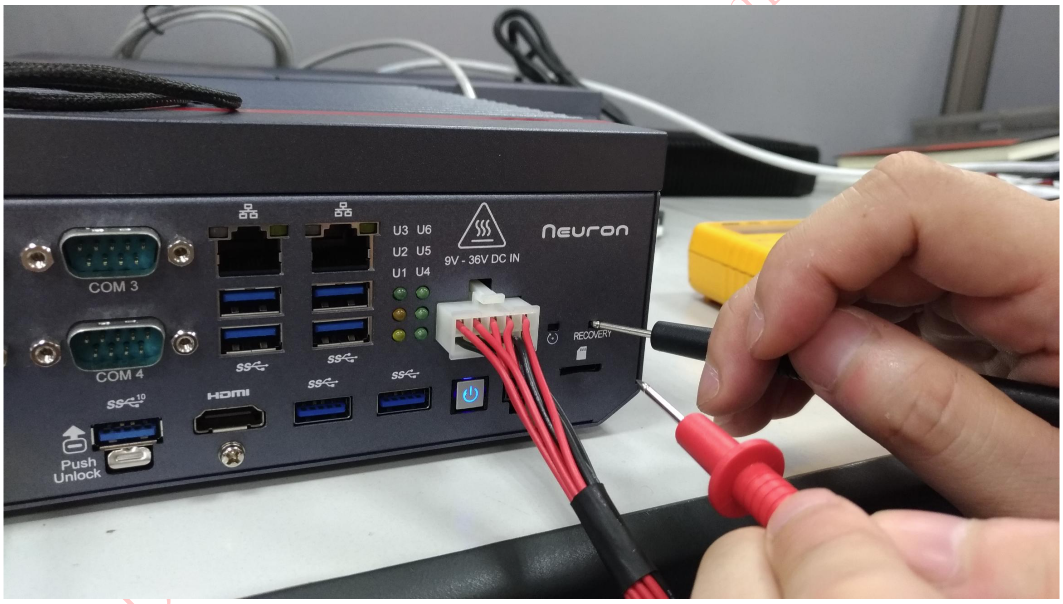
i) Steady pressed RECOVERY button.



ii) Hold RECOVERY button pressed and then press RESET button. (The button next to RECOVERY button)



iii) Release RESET button dirst and then release RECOVERY button.



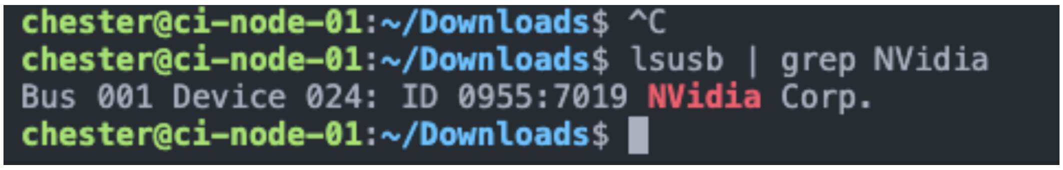
### 2.2.4 4. Now ROScube is in recovery mode now.

### 2.2.5 5. Parepare mfi image on your host computer.

i) Download mfi image.
ii) Install required tools.

```
$> sudo apt install python qemu-user-static
```

iii) Check ROScube is in recovery mode now.



```
chester@ci-node-01:~/Downloads$ ^C
chester@ci-node-01:~/Downloads$ lsusb | grep NVidia
Bus 001 Device 024: ID 0955:7019 NVidia Corp.
chester@ci-node-01:~/Downloads$ 
```

### 2.2.6 6. Flash image to ROScube

Assuming image's file name is `mfi_rqx580_nvidia-ubuntu-rootfs-bionic_L4T-32-4-3-Kernel-1-0-7.tbz2`. Un-archive this file first.

```
$> tar xvf mfi_rqx580_nvidia-ubuntu-rootfs-bionic_L4T-32-4-3-Kernel-1-0-7.tbz2
```

Then, use internal tool, `nvmflash.sh` to run the flashing procedure. **Please make sure your Host PC has attached to ROScube-X**. Besure run ./nvmflash.sh as root permission.

```
$> cd mfi_rqx_580
$> sudo ./nvmflash.sh
```

**Note:** You may need to input your **host PC**'s root password when flashing the image.

**Note:** The flashing procedure might take 8 ~ 10 minutes.

## 2.3 Massive Flashing

`nvmflash.sh` supports massive flashing, which means you can attach multiple ROScube and run nvmflash.sh to flash the image to multiple boards at the same time.

**Warning:** Please do not attach different models (e.g. RQX580 and RQX58G) to your host computer when using nvmflash.sh.

CHAPTER

THREE

# ROSCUBE-X BSP QUICK START

## 3.1 Default User and Password

ROScube-X BSP can login as root user within initial stage.

- username: **root**
- password: **adlinkros**

## 3.2 Check BSP Version

You can use the following command to check BSP and L4T version.

To check L4T version, please use the following command.

```
$> cat /etc/nv_tegra_release
```

To check kernel version, please use the following command.

```
$> uname -a
```

## 3.3 Hardware Monitor

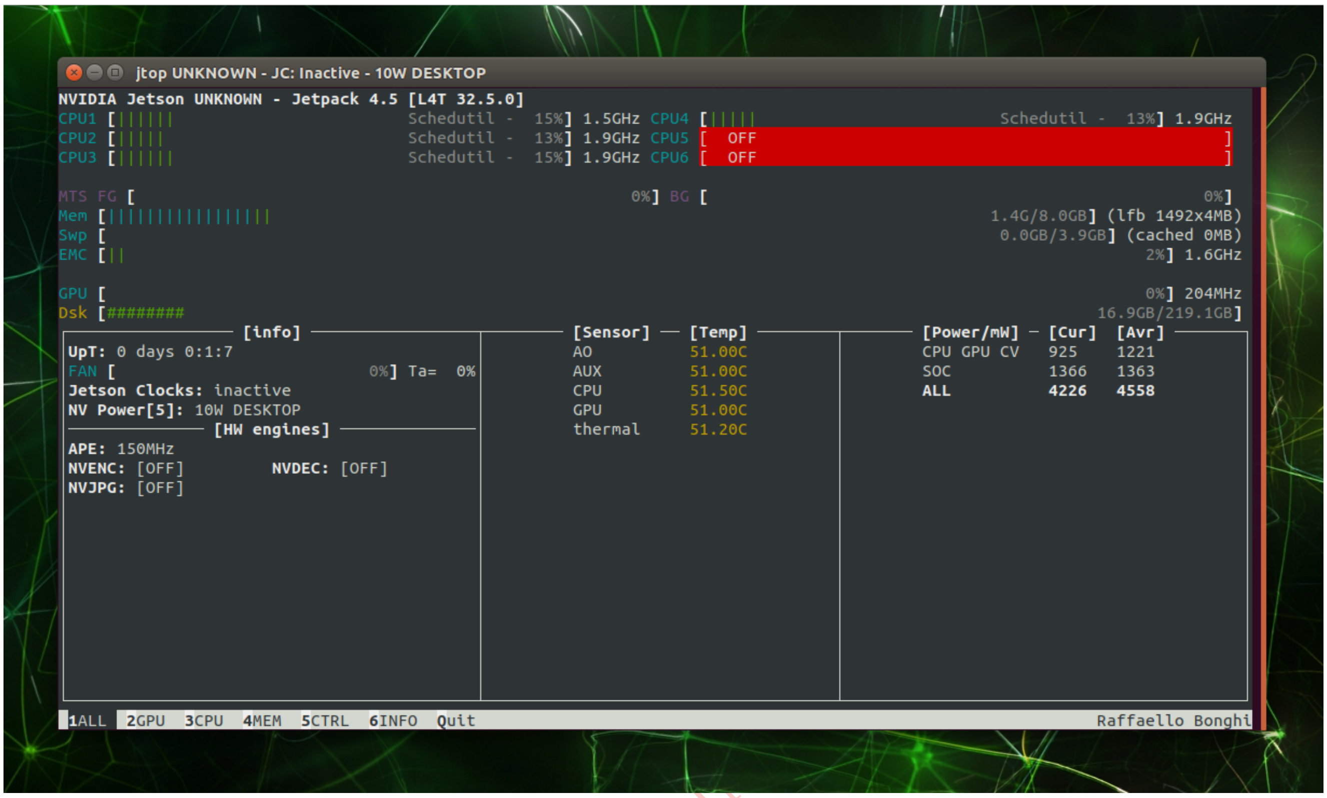
Since generic x86 computer with Nvidia GPU PCIe card under linux has a tool `nvidia-smi` to check GPU status, but ROscube-X and ROScube-Pico are ARM based CPU, there's no `nvidia-smi` tool available. You could use a third party tool: `jetson-stats` to monitor Jetson's CPU, GPU and memory status.

```
$> sudo apt install python-pip python-dev build-essential
$> sudo pip install --upgrade pip
$> sudo pip install jetson-stats

# Then reboot system to apply jetson-stats systemcv service.

$> sudo jtop
```

- Overall Hardware Monitor Screenshoot



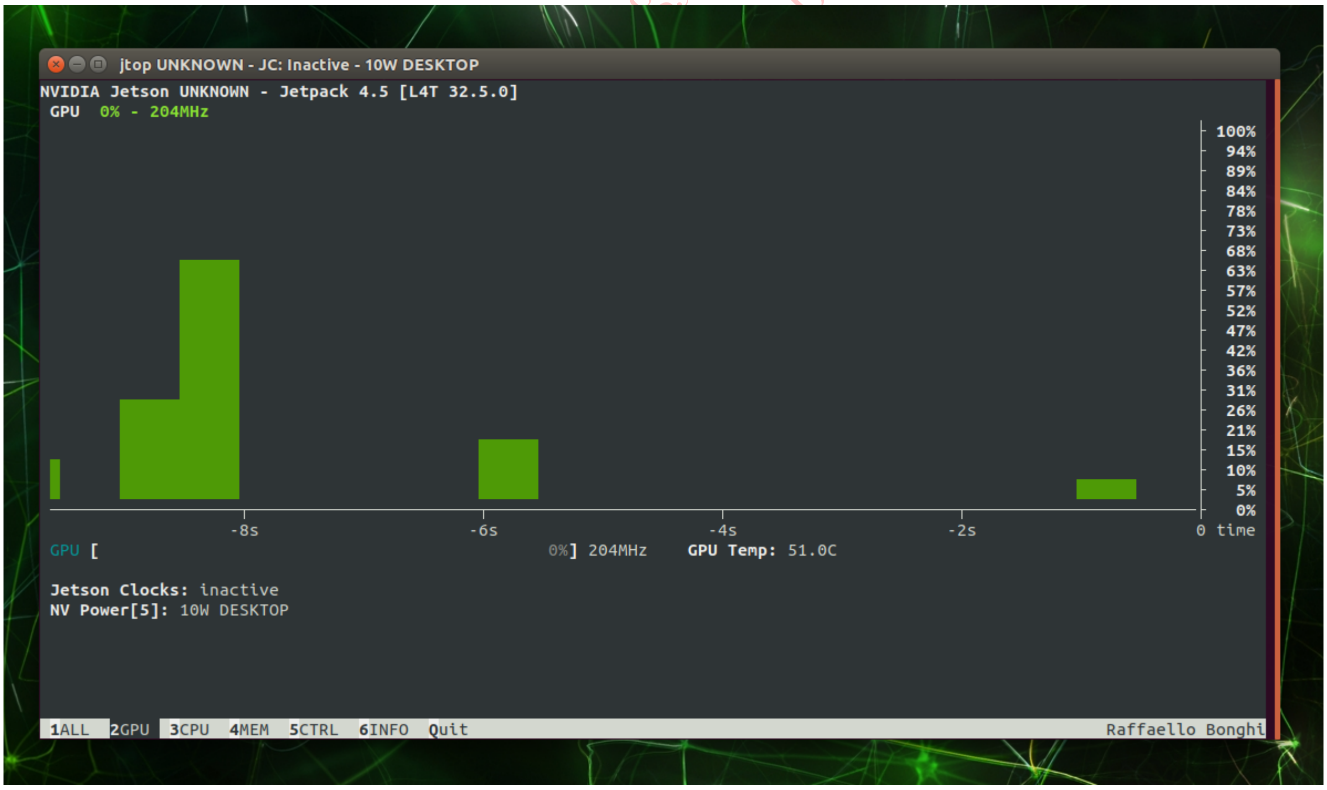
```
jtop UNKNOWN - JC: Inactive - 10W DESKTOP
NVIDIA Jetson UNKNOWN - Jetpack 4.5 [L4T 32.5.0]
CPU1 [||||||                        Schedutil -  15%] 1.5GHz CPU4 [|||||                        Schedutil -  13%] 1.9GHz
CPU2 [|||||                         Schedutil -  13%] 1.9GHz CPU5 [  OFF                                            ]
CPU3 [||||||                        Schedutil -  15%] 1.9GHz CPU6 [  OFF                                            ]

MTS FG [                                        0%] BG [                                                     0%]
Mem [||||||||||||||||                                                         1.4G/8.0GB] (lfb 1492x4MB)
Swp [                                                                          0.0GB/3.9GB] (cached 0MB)
EMC [||                                                                                2%] 1.6GHz

GPU [                                                                                  0%] 204MHz
Dsk [########                                                                    16.9GB/219.1GB]
             [info]                             [Sensor] ─ [Temp]                [Power/mW] ─ [Cur]  [Avr]
UpT: 0 days 0:1:7                            AO          51.00C                 CPU GPU CV    925    1221
FAN [                  0%] Ta=  0%           AUX         51.00C                 SOC           1366   1363
Jetson Clocks: inactive                      CPU         51.50C                 ALL           4226   4558
NV Power[5]: 10W DESKTOP                     GPU         51.00C
           [HW engines]                      thermal     51.20C
APE: 150MHz
NVENC: [OFF]          NVDEC: [OFF]
NVJPG: [OFF]

1ALL  2GPU  3CPU  4MEM  5CTRL  6INFO  Quit                                                      Raffaello Bonghi
```

- GPU RealTime Monitor Screenshoot



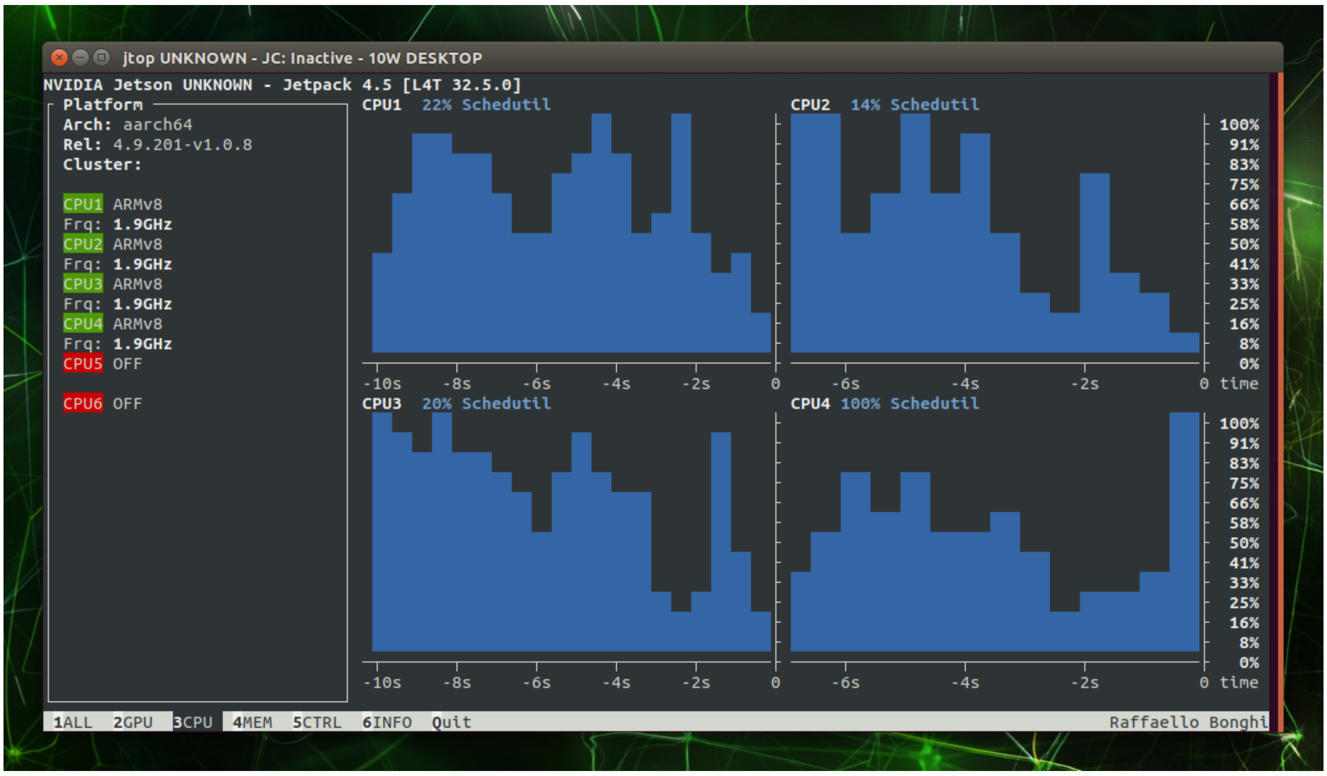
```
jtop UNKNOWN - JC: Inactive - 10W DESKTOP
NVIDIA Jetson UNKNOWN - Jetpack 4.5 [L4T 32.5.0]
 GPU  0% - 204MHz
```

```
GPU [                                  0%] 204MHz    GPU Temp: 51.0C

Jetson Clocks: inactive
NV Power[5]: 10W DESKTOP

1ALL  2GPU  3CPU  4MEM  5CTRL  6INFO  Quit                                                      Raffaello Bonghi
```

- CPU RealTime Monitor Screenshoot



```
jtop UNKNOWN - JC: Inactive - 10W DESKTOP
NVIDIA Jetson UNKNOWN - Jetpack 4.5 [L4T 32.5.0]
 Platform             CPU1  22% Schedutil                          CPU2  14% Schedutil
 Arch: aarch64                                                                          100%
 Rel: 4.9.201-v1.0.8                                                                     91%
 Cluster:                                                                                83%
                                                                                         75%
 CPU1 ARMv8                                                                              66%
 Frq: 1.9GHz                                                                             58%
 CPU2 ARMv8                                                                              50%
 Frq: 1.9GHz                                                                             41%
 CPU3 ARMv8                                                                              33%
 Frq: 1.9GHz                                                                             25%
 CPU4 ARMv8                                                                              16%
 Frq: 1.9GHz                                                                              8%
 CPU5 OFF                                                                                 0%
                      -10s   -8s    -6s    -4s    -2s    0    -6s       -4s       -2s     0 time
 CPU6 OFF             CPU3  20% Schedutil                          CPU4 100% Schedutil
                                                                                        100%
                                                                                         91%
                                                                                         83%
                                                                                         75%
                                                                                         66%
                                                                                         58%
                                                                                         50%
                                                                                         41%
                                                                                         33%
                                                                                         25%
                                                                                         16%
                                                                                          8%
                                                                                          0%
                      -10s   -8s    -6s    -4s    -2s    0    -6s       -4s       -2s     0 time
 1ALL  2GPU  3CPU  4MEM  5CTRL  6INFO  Quit                                    Raffaello Bonghi
```

**Note:** Before using jetson-stats, you should install nvidia-jetpack first.

CHAPTER

FOUR

# DRIVER PACKAGE USAGE

Driver Package is a package contains several assets that runs on host computer, which allows you to do:

1. Customize system file system.
2. Flash system image to target machine.
3. System image back up.
4. Create a custom mfi image.

User can modify their rootfs on their host computer before flashing the image to target machine.

You could use tar tool to unarchive driver_package.

```
$> tar xvf RQX-580-Driver-Package-L4T-32-4-4-Kernel-1-0-8.tar.gz
```

## 4.1 Customize System FileSystem

### 4.1.1 Prepare Filesystem

Driver package allows user to customize their filesystem. Driver package provides a script (`create_image.sh`) to create a board specific image base on several rootfs selection, here's what we support.

1. nvidia_sample_rootfs - Nvidia provided sample file system.

Please download the correct L4T version from https://developer.nvidia.com/embedded/linux-tegra-archive. For example, if you are using ADLINK provided driver package 32.4.4, you should download https://developer.nvidia.com/embedded/L4T/r32_Release_v4.4/r32_Release_v4.4-GMC3/T186/Tegra_Linux_Sample-Root-Filesystem_R32.4.4_aarch64.tbz2

2. custom_slim_rootfs_bionic - Ubuntu provided arm64 18.04.05 minimun image.

Please download ubuntu 18.04.5 base image from Ubuntu's website. https://cdimage.ubuntu.com/ubuntu-base/releases/18.04.5/release/. The binary link is at: https://cdimage.ubuntu.com/ubuntu-base/releases/18.04.5/release/ubuntu-base-18.04.5-base-arm64.tar.gz

3. custom_slim_rootfs_focal - Ubuntu provided arm64 20.04.01 minimun image. (Experimental, not fully tested yet)

Please download ubuntu 20.04.1 base image from Ubuntu's website. https://cdimage.ubuntu.com/ubuntu-base/releases/20.04.1/release/. The binary link is at: https://cdimage.ubuntu.com/ubuntu-base/releases/20.04.1/release/ubuntu-base-20.04.1-base-arm64.tar.gz

**Note:** Before customizing the rootfs, all these rootfs binaries should be downloaded from their official website. ADLINK doesn't provide them.

After you download rootfs binary successfully, put the binary under the driver package folder. Then, run `./create_image.sh`.

```
$> cd <driver_package_path>
$> sudo ./create_image.sh nvidia_sample_rootfs
```

### 4.1.2 Modify Filesystem under Chroot

Users can chroot into filesystem, and install packages or put files to filesystem. That allows user to put their software and configurations into rootfs.

Mount host environment to filesystem.

```
$> sudo apt update qemu-user-static
$> cd <driver package>
$> cd rootfs
$> sudo cp /usr/bin/qemu-aarch64-static usr/bin/
$> sudo mount --bind /dev/ dev/
$> sudo mount --bind /sys/ sys/
$> sudo mount --bind /proc/ proc/

# Copy host environment dns settings.
$> sudo cp /etc/resolv.conf etc/resolv.conf.host
$> sudo mv etc/resolv.conf etc/resolv.conf.saved
$> sudo mv etc/resolv.conf.host etc/resolv.conf
```

Now you can chroot into filesystem and modify filesystem.

```
$> sudo LC_ALL=C LANG=C.UTF-8 chroot . /bin/bash
$(chroot)> apt update
$(chroot)> apt install wget vim

# or you can install jetpack sdk
$(chroot)> apt install nvidia-l4t-jetson-multimedia-api=32.4.4-20201016123640
$(chroot)> apt install nvidia-jetpack
```

Once you finish modification, you can exit chroot environment by pressing ctrl+D. Then, remove cache files, e.g. apt cache to save storage space.

```
$(chrooot)> exit
```

(continues on next page)

(continued from previous page)

```
$> sudo umount ./proc
$> sudo umount ./sys
$> sudo umount ./dev
$> sudo rm usr/bin/qemu-aarch64-static

# Restore dns setting.
$> sudo rm etc/resolv.conf
$> sudo mv etc/resolv.conf.saved etc/resolv.conf

# Remove caches and logging files.
$> sudo rm -rf var/lib/apt/lists/*
$> sudo rm -rf dev/*
$> sudo rm -rf var/log/*
$> sudo rm -rf var/tmp/*
$> sudo rm -rf var/cache/apt/archives/*.deb
$> sudo rm -rf tmp/*
```

## 4.2 Flash System Image to Target Machine

Once you prepared rootfs in step: *customize rootfs*. You can use *flash.sh* tool to flash image to target machine. **Remember to put machine into recovery mode before flashing.**

```
# Put machine into recovery mode first.
$> sudo ./flash.sh <target name> mmcblk0p1
```

| Model | Target Name |
|---|---|
| RQX-580 | rqx_580 |
| RQX-58G | rqx_58g |
| NPN-1 (Pico Nano) | rqp_nano |
| NPN-2 (Pico AGX NX) | rqp_nx |

CHAPTER

FIVE

# BOOT ON EXTERNAL STORAGE

ROScube can boot (strictly mount) filesystem from external storage. Available types are:

1. SD card
2. NVMe M.2 SSD

ADLINK provides a tool to boot on external storage.

https://github.com/Adlink-ROS/rootOnStorage

## 5.1 Mounting FileSystem From SD Card

When inserting a sd card to RQX-580 or RQX-58G, the storage will be appeared as a `/dev/mmcblk1` storage.

**Note:** ROScube's internal storage (eMMC) is appeared as /dev/mmcblk0.

The first step is: `format SD card to EXT4 filesystem`.

```
$> cd rootOnStorage/sdmmc
$> sudo ./copy-rootfs-sdmmc.sh
$> sudo ./setup-service.sh
# Then, reboot system.
```

## 5.2 Mounting FileSystem From NVMe M.2 SSD

When mounting a NVMe M.2 SSD to RQX-580 or RQX-58G, the storage will be appeared as a `/dev/nvme0n1` storage.

The first step is: `format NVMe M.2 SSD to EXT4 filesystem`.

```
$> cd rootOnStorage/nvme
$> sudo ./copy-rootfs-nvme.sh
```

(continues on next page)

(continued from previous page)

```
$> sudo ./setup-service.sh
# Then, reboot system.
```

CHAPTER

SIX

# NVIDIA JETSON SOFTWARE STACK

## 6.1 Version Table

| L4T | Linux Kernel | Jetpack SDK | CUDA | TensorRT | CUDNN | VisionWork | OpenCV | VPI | DeepStream SDK |
|---|---|---|---|---|---|---|---|---|---|
| 32.4.3 | 4.9.190 | 4.4.0 | 10.2 | 7.1.3 | 8.0 | 1.6 | 4.1.1 | 1.0.3 | 5.0 |
| 32.4.4 | 4.9.190 | 4.4.1 | 10.2 | 7.1.3 | 8.0 | 1.6 | 4.1.1 | 1.0.3 | 5.0 |
| 32.5.0 | 4.9.210 | 4.5.0 | 10.2 | 7.1.3 | 8.0 | 1.6 | 4.1.1 | 1.1.0 | 5.0 |
| 32.5.1 | 4.9.210 | 4.5.1 | 10.2 | 7.1.3 | 8.0 | 1.6 | 4.1.1 | 1.0.3 | 5.1 |

## 6.2 JetPack SDK Installation

Since Nvidia L4T 32.4.x support L4T OTA, that users can download Nvidia JetPack from Nvidia's official APT repository. For ROScube-X users, you can use `apt` to install Nvidia JetPack components, e.g. tensorrt, cuda.

**Before installing JetPack SDK, you should install `nvidia-l4t-jetson-multimedia-api` in a proper version coresponding to L4T version (e.g. 32.4.3 or 32.4.4).**

```
$> sudo apt update
$> sudo apt install nvidia-l4t-jetson-multimedia-api=32.4.4-20201016123640
```

**To install the whole JetPack SDK**

```
$> sudo apt update
$> sudo apt install nvidia-jetpack
```

**To install particular JetPack SDK compoment**, e.g. nvidia-tensorrt.

```
$> sudo apt update
$> sudo apt install nvidia-tensorrt
```

> **Warning:** It's not suggested to install `nvidia-jetpack` directly, because nvidia-jetpack is a meta package which will install `ALL` JetPack SDK compoments. All JetPack SDK compomenets occupy `8GB`.

## 6.3 JetPack SDK Example Usage

### 6.3.1 YOLO Object Detection

*1. Set up your environment*

```
$> export CUDA_HOME=/usr/local/cuda
$> export PATH=$CUDA_HOME/bin:$PATH
$> export LD_LIBRARY_PATH=$CUDA_HOME/lib64:$LD_LIBRARY_PATH
```

*2. Check* `nvcc` *is workable*

```
$> nvcc --version
...
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
...
```

*3. Retrive darknet repository from github*

```
$> sudo apt update
$> sudo apt install git
$> git clone https://github.com/AlexeyAB/darknet.git
```

*4. Build darknet with CUDNN and OpenCV support*

Modify darknet's Makefile with the following..

- GPU=1
- CUDNN=1
- OPENCV=1

```
$> sudo apt update
$> sudo apt install make build-essential
$> cd darknet
$> # Edit Makefile GPU=1, CUDNN=1, OPENCV=1
$> make
```

*5. Download the pre-trained weights*

```
$> wget https://pjreddie.com/media/files/yolov3-tiny.weights
```

**Warning:** ADLINK doesn't own the pre-train data. Thsi pre-trained data is a contribution from original author in community.

*6. Run object detector from example image.*

```
$> ./darknet detect cfg/yolov3-tiny.cfg yolov3-tiny.weights data/dog.jpg
```

## 6.4 DeepStream SDK Installation

Download the DeepStream 5.0 for Jetson

https://developer.nvidia.com/deepstream-getting-started

```
$> sudo apt install ./deepstream_sdk_5.0_arm64.deb
```

**Note:** Please refer *jetson software version table* to download proper DeepStream SDK version.

## 6.5 DeepStream SDK Example Usage

After finished DeepStream SDK installation, there will be some DeepStream examples and tools available under `/opt/nvidia/deepstream/deepsteeam-<version>`

### 6.5.1 DeepStream Sample App

```
$> deepstream-app -c /opt/nvidia/deepstream/deepstream-5.0/samples/configs/deepstream-app/
↪<config file>
```

Where <config file> could be replaced with the following table:

| Model | Config File |
|---|---|
| RQX-580 | source12_1080p_dec_infer-resnet_tracker_tiled_display_fp16_tx2.txt |
| RQX-58G | source12_1080p_dec_infer-resnet_tracker_tiled_display_fp16_tx2.txt |
| NPN-1 (Pico Nano) | source8_1080p_dec_infer-resnet_tracker_tiled_display_fp16_nano.txt |
| NPN-2 (Pico AGX NX) | source12_1080p_dec_infer-resnet_tracker_tiled_display_fp16_tx2.txt |