JETSON ORIN NANO ????

本章主要　　Jetson特色的　　AI功能使用

???

??jtop

jtop 是英伟达专为 **Jetson** 系列边缘计算设备 开发的 交互式系统监控工具。

1 ??jtop

- 安装 jtop需要的依赖库

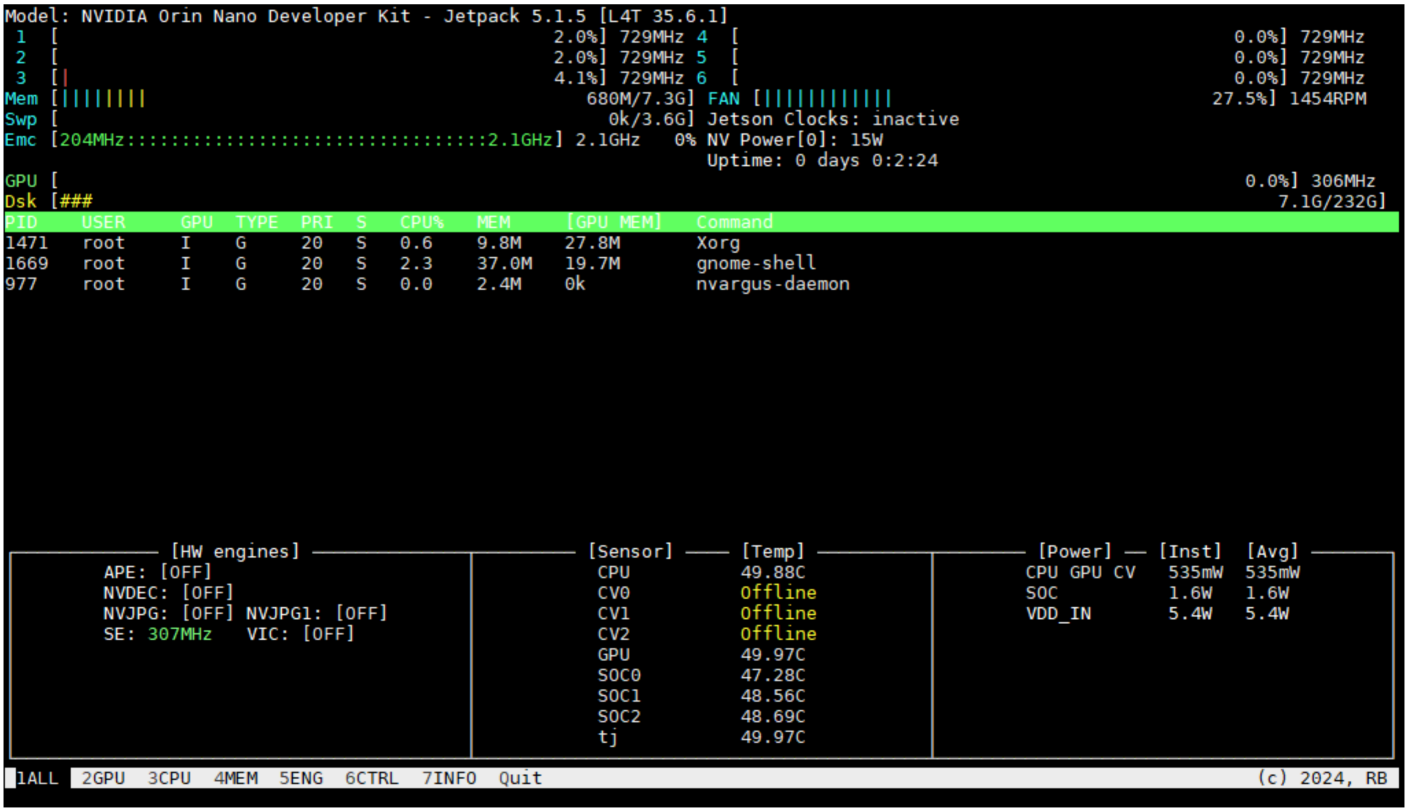
```
sudo apt update
sudo apt install python3
sudo apt install python3-pip
```

- 安装 jtop

```
sudo pip3 install -U pip -i https://pypi.tuna.tsinghua.edu.cn/simple
sudo pip3 install jetson-stats -i https://pypi.tuna.tsinghua.edu.cn/simple
sudo systemctl restart jtop.service
```

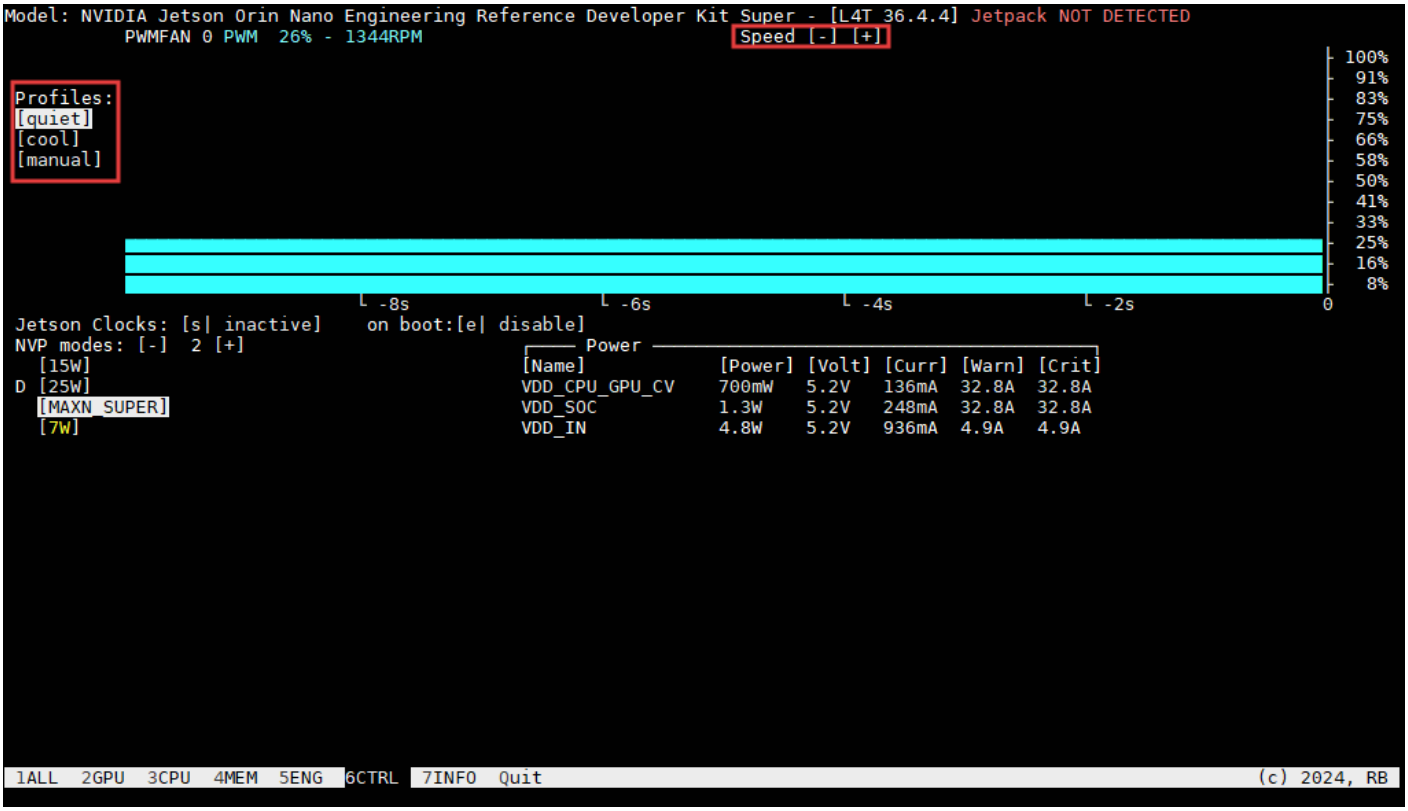
2 jtop???

- 安装完成后在控制台中输入 `jtop` 打开主界面

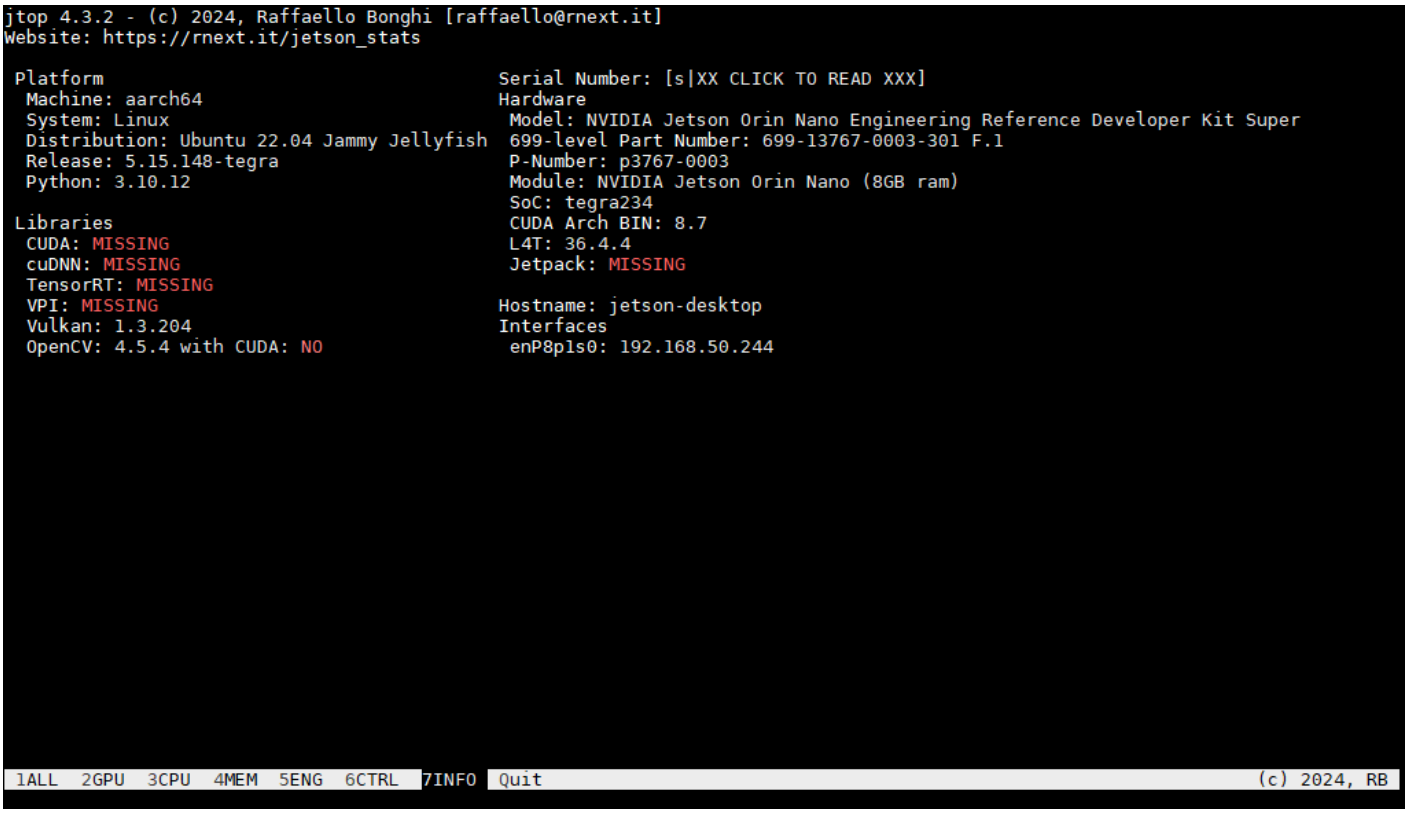


2.1 监视各个模块的工作信息

2.2 □□□□


Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - [L4T 36.4.4] Jetpack NOT DETECTED
PWMFAN 0 PWM 26% - 1344RPM
Speed [-] [+]
Profiles:
[quiet]
[cool]
[manual]
100%
91%
83%
75%
66%
58%
50%
41%
33%
25%
16%
8%
-8s
-6s
-4s
-2s
0
Jetson Clocks: [s| inactive]
on boot:[e| disable]
NVP modes: [-] 2 [+]
[15W]
D [25W]
[MAXN_SUPER]
[7W]
Power
[Name] [Power] [Volt] [Curr] [Warn] [Crit]
VDD_CPU_GPU_CV 700mW 5.2V 136mA 32.8A 32.8A
VDD_SOC 1.3W 5.2V 248mA 32.8A 32.8A
VDD_IN 4.8W 5.2V 936mA 4.9A 4.9A
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit
(c) 2024, RB

2.3 □□□□□□□□

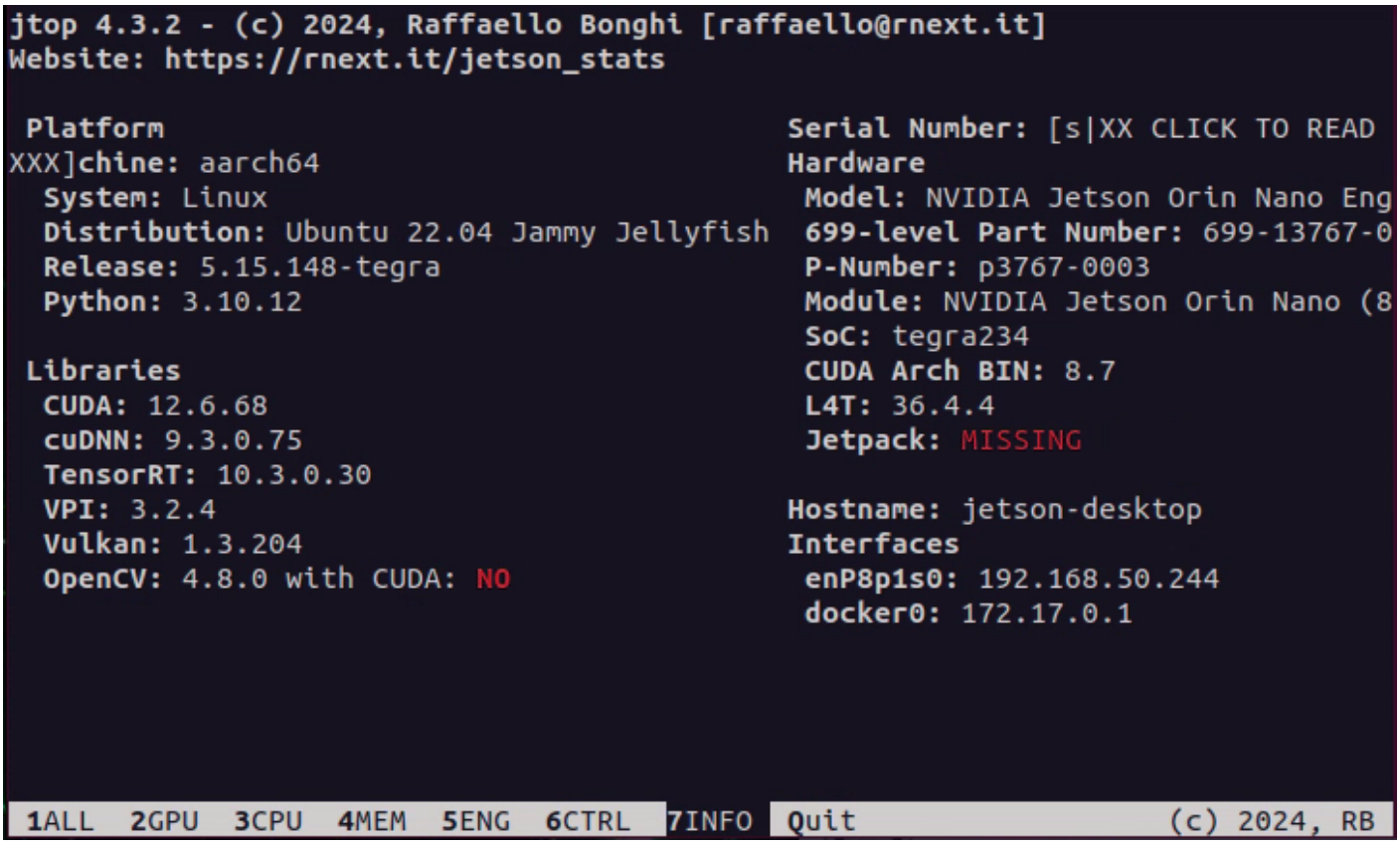

jtop 4.3.2 - (c) 2024, Raffaello Bonghi [raffaello@rnext.it]
Website: https://rnext.it/jetson_stats
Platform
Machine: aarch64
System: Linux
Distribution: Ubuntu 22.04 Jammy Jellyfish
Release: 5.15.148-tegra
Python: 3.10.12
Libraries
CUDA: MISSING
cuDNN: MISSING
TensorRT: MISSING
VPI: MISSING
Vulkan: 1.3.204
OpenCV: 4.5.4 with CUDA: NO
Serial Number: [s|XX CLICK TO READ XXX]
Hardware
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super
699-level Part Number: 699-13767-0003-301 F.1
P-Number: p3767-0003
Module: NVIDIA Jetson Orin Nano (8GB ram)
SoC: tegra234
CUDA Arch BIN: 8.7
L4T: 36.4.4
Jetpack: MISSING
Hostname: jetson-desktop
Interfaces
enP8p1s0: 192.168.50.244
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit
(c) 2024, RB

??CUDA

Jetson CUDA NVIDIA AI ARM Jetson **GPU**

??CUDA JETSON SDK

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



??????????

> **JetPack 6.1 (rev1) JetPack 5.1.5**

```
sudo nvpmodel -m 2 #nano 模式2为super
sudo jetson_clocks --fan
```

cpu压力测试:

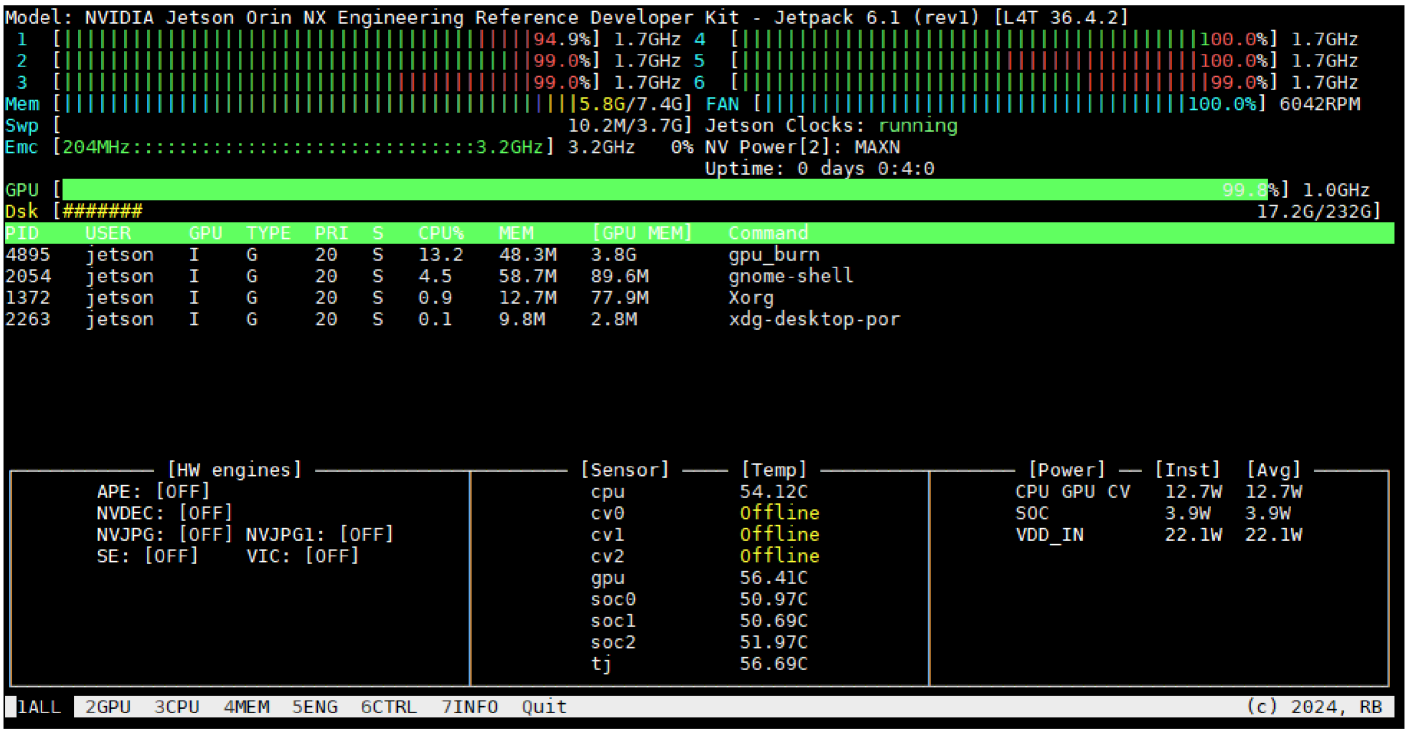
```
sudo apt install stress
stress --cpu 8 --io 4 --vm 2 --vm-bytes 128M --hdd 1 --hdd-bytes 1024M
```

gpu压力测试:

```
git clone https://github.com/anseeto/jetson-gpu-burn/
cd jetson-gpu-burn
make
./gpu_burn 100000
```

状态查询:

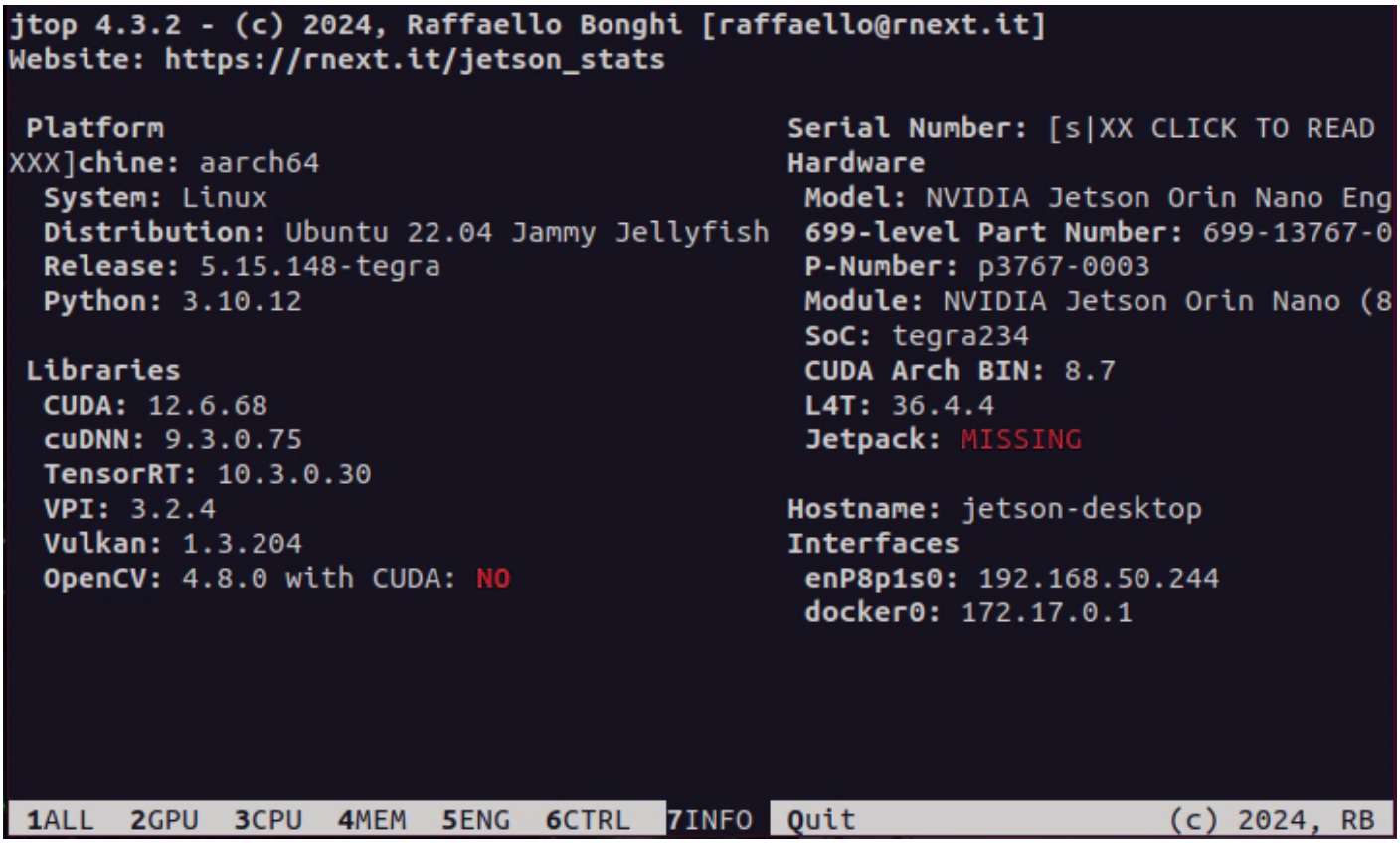
```
sudo jtop
```



> Orin Nano 整机最大功率为 **30W.**
>
> Orin NX 整机最大功率为 **40W.**

????opencv with cuda

> JetPack□□□ opencv□□□□ cuda□□□□□□□□□□



1. ??????

```
#!/bin/bash
# opencv_install.sh
# Modified from
https://github.com/AastaNV/JEP/blob/master/script/install_opencv4.10.0_Jetpack6.1.sh

version="4.10.0"
folder="workspace"
remove_old=""

set -e

# Parse command-line arguments
for arg in "$@"; do
```

```
    case $arg in
        --version=*)
            version="${arg#*=}"
            ;;
        --folder=*)
            folder="${arg#*=}"
            ;;
        --remove-old=*)
            remove_old="${arg#*=}"
            ;;
        --help|-h)
            echo "Usage: $0 [--version=4.x.x] [--folder=dir] [--remove-old=yes/no]"
            exit 0
            ;;
        *)
            echo "Unknown parameter: $arg"
            echo "Usage: $0 [--version=4.x.x] [--folder=dir] [--remove-old=yes/no]"
            exit 1
            ;;
    esac
done

# Create installation directory if it doesn't exist
if [ ! -d "$folder" ]; then
    echo "Creating directory: $folder"
    mkdir -p "$folder"
fi
cd "$folder" || exit

# Old OpenCV removal logic
if [ -z "$remove_old" ]; then
    read -rp "Do you want to remove system-installed OpenCV? (yes/no): " remove_old
fi

case "$remove_old" in
    [yY] | [yY][eE][sS])
        echo "** Removing system OpenCV packages"
        sudo apt -y purge *libopencv*
        sudo apt -y autoremove
        ;;
```

```
    *)
        echo "** Skipping system OpenCV removal"
        ;;
esac

echo "-----------------------------------"
echo "** Installing dependencies (1/4)"
echo "-----------------------------------"
sudo apt-get update
sudo apt-get install -y build-essential cmake git libgtk2.0-dev pkg-config \
    libavcodec-dev libavformat-dev libswscale-dev libgstreamer1.0-dev \
    libgstreamer-plugins-base1.0-dev python3-dev python3-numpy libtbb2 \
    libtbb-dev libjpeg-dev libpng-dev libtiff-dev libv4l-dev v4l-utils qv4l2 curl

# Verify essential dependencies installed
for dep in g++ cmake git pkg-config; do
    if ! command -v "$dep" > /dev/null; then
        echo "Error: $dep installation failed"
        exit 1
    fi
done

echo "-----------------------------------"
echo "** Downloading OpenCV ${version} (2/4)"
echo "-----------------------------------"

# Check if source files already exist
download_opencv=false
download_contrib=false

if [ ! -f "opencv-${version}.zip" ]; then
    echo "Downloading opencv-${version}.zip"
    wget -O opencv-${version}.zip https://github.com/opencv/opencv/archive/${version}.zip || {
        echo "Download failed! Check your internet connection or verify the version exists"
        exit 1
    }
    download_opencv=true
else
    echo "opencv-${version}.zip exists, skipping download"
fi
```

```
if [ ! -f "opencv_contrib-${version}.zip" ]; then
    echo "Downloading opencv_contrib-${version}.zip"
    wget -O opencv_contrib-${version}.zip
https://github.com/opencv/opencv_contrib/archive/${version}.zip || {
        echo "Download failed! Check your internet connection or verify the version exists"
        exit 1
    }
    download_contrib=true
else
    echo "opencv_contrib-${version}.zip exists, skipping download"
fi

# Unpack source files
if [ ! -d "opencv-${version}" ] || $download_opencv; then
    if [ -d "opencv-${version}" ]; then
        echo "Removing existing opencv-${version} directory"
        rm -rf "opencv-${version}"
    fi
    echo "Unpacking opencv-${version}.zip"
    unzip -q opencv-${version}.zip || {
        echo "Extraction failed! File may be corrupt"
        exit 1
    }
fi

if [ ! -d "opencv_contrib-${version}" ] || $download_contrib; then
    if [ -d "opencv_contrib-${version}" ]; then
        echo "Removing existing opencv_contrib-${version} directory"
        rm -rf "opencv_contrib-${version}"
    fi
    echo "Unpacking opencv_contrib-${version}.zip"
    unzip -q opencv_contrib-${version}.zip || {
        echo "Extraction failed! File may be corrupt"
        exit 1
    }
fi

# Clean up zip files after successful extraction
if [ $? -eq 0 ]; then
```

```
    rm -f opencv-${version}.zip opencv_contrib-${version}.zip
fi

cd opencv-${version} || exit

echo "-----------------------------------"
echo "** Building OpenCV ${version} (3/4)"
echo "-----------------------------------"
mkdir -p release
cd release

# Auto-detect CUDA architecture
cuda_arch=""
if command -v nvidia-smi &> /dev/null; then
    gpu_name=$(nvidia-smi --query-gpu=name --format=csv,noheader | head -n1)
    if [[ $gpu_name == *"Orin"* ]] || [[ $gpu_name == *"Jetson"* ]]; then
        cuda_arch="8.7"
    elif [[ $gpu_name == *"A100"* ]]; then
        cuda_arch="8.0"
    fi
fi

cmake_cmd="cmake -D WITH_CUDA=ON -D WITH_CUDNN=ON -D OPENCV_GENERATE_PKGCONFIG=ON "
cmake_cmd+="-D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-${version}/modules "
cmake_cmd+="-D WITH_GSTREAMER=ON -D WITH_LIBV4L=ON -D BUILD_opencv_python3=ON "
cmake_cmd+="-D BUILD_TESTS=OFF -D BUILD_PERF_TESTS=OFF -D BUILD_EXAMPLES=OFF "
cmake_cmd+="-D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local "

# Add CUDA architecture if detected
if [ -n "$cuda_arch" ]; then
    echo "Detected NVIDIA GPU: ${gpu_name}, using CUDA_ARCH_BIN=${cuda_arch}"
    cmake_cmd+="-D CUDA_ARCH_BIN=${cuda_arch} -D CUDA_ARCH_PTX=\"\" "
else
    echo "No supported GPU detected, skipping CUDA architecture flags"
fi

# Execute CMake configuration
echo "CMake command: $cmake_cmd .."
$cmake_cmd .. || {
    echo "CMake configuration failed"
```

```
    exit 1
}

# Parallel build (leave one core for system stability)
cpu_cores=$(($(nproc) - 1))
[ $cpu_cores -lt 1 ] && cpu_cores=1
echo "Building with ${cpu_cores} CPU cores"
make -j${cpu_cores} || {
    echo "Compilation failed"
    exit 1
}

echo "-----------------------------------"
echo "** Installing OpenCV ${version} (4/4)"
echo "-----------------------------------"
sudo make install || {
    echo "Installation failed"
    exit 1
}

# Add environment variables to .bashrc (only if not already present)
bashrc=~/.bashrc
env_lines=(
    "export LD_LIBRARY_PATH=/usr/local/lib:\$LD_LIBRARY_PATH"
    "export PYTHONPATH=/usr/local/lib/python3.10/site-packages/:\$PYTHONPATH"
)

for line in "${env_lines[@]}"; do
    if ! grep -Fxq "$line" "$bashrc"; then
        echo "Adding to .bashrc: $line"
        echo "$line" >> "$bashrc"
    else
        echo "Environment variable already exists: $line"
    fi
done

source ~/.bashrc

echo "** OpenCV ${version} installation completed"
echo "Verification commands:"
```

```
echo "  pkg-config --modversion opencv4"
echo "  python3 -c 'import cv2; print(cv2.__version__)'"

echo "** Installation successful!"
```

2. ????

2.1 卸载自带 opencv

```
sudo apt-get purge libopencv*
sudo apt autoremove
sudo apt-get update
```

2.2 安装前置软件包

```
sudo apt-get update
sudo apt-get install -y build-essential cmake git libgtk2.0-dev pkg-config libavcodec-dev
libavformat-dev libswscale-dev
sudo apt-get install -y libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev python3.10-dev
python3-numpy
sudo apt-get install -y libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libv4l-dev v4l-
utils qv4l2
sudo apt-get install -y curl
```

2.3 获取 opencv源码 （以 4.10.0版本为例 ）

```
version=4.10.0
wget -O "opencv-${version}.zip" "https://github.com/opencv/opencv/archive/${version}.zip"
wget -O "opencv_contrib-${version}.zip"
"https://github.com/opencv/opencv_contrib/archive/${version}.zip"
unzip "opencv-${version}.zip"
unzip "opencv_contrib-${version}.zip"
rm "opencv-${version}.zip" "opencv_contrib-${version}.zip"
cd "opencv-${version}/"
```

2.4 编译源码

> 此步骤至少需要半小时以上。

>

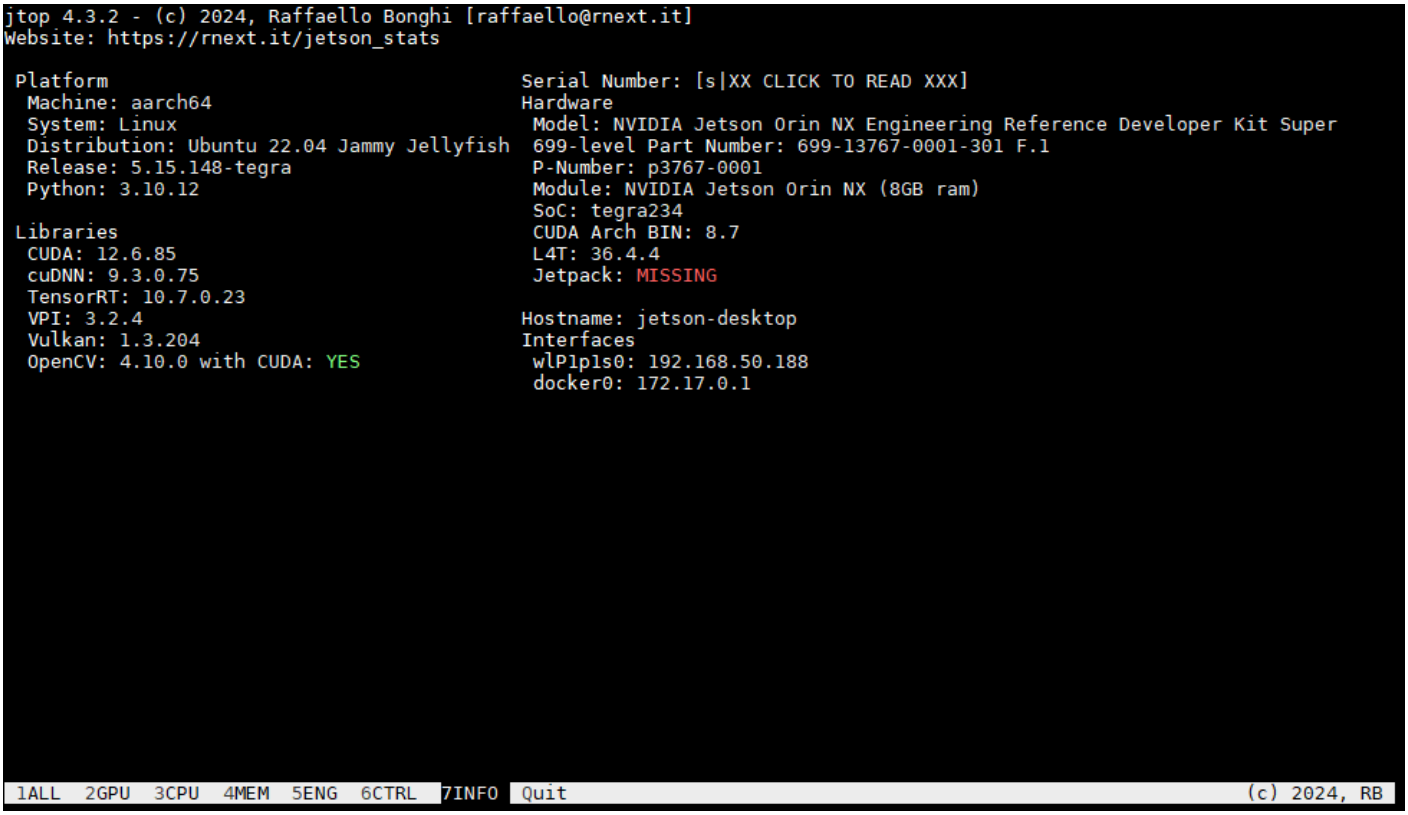
□□□□□□□□□□□□□□□□□□□□□□□□□□

```
mkdir build
cd build/
cmake -D WITH_CUDA=ON -D WITH_CUDNN=ON -D CUDA_ARCH_BIN="8.7" -D CUDA_ARCH_PTX="" -D
OPENCV_GENERATE_PKGCONFIG=ON -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-
${version}/modules -D WITH_GSTREAMER=ON -D WITH_LIBV4L=ON -D BUILD_opencv_python3=ON -D
BUILD_TESTS=OFF -D BUILD_PERF_TESTS=OFF -D BUILD_EXAMPLES=OFF -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local ..
make -j$(nproc)
```

2.5 □□

```
sudo make install
echo 'export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH' >> ~/.bashrc
echo 'export PYTHONPATH=/usr/local/lib/python3.10/site-packages/:$PYTHONPATH' >> ~/.bashrc
source ~/.bashrc
```

??????



```
#--test_cuda.cpp

#include <opencv2/opencv.hpp>
#include <opencv2/core/cuda.hpp>
#include <opencv2/cudaarithm.hpp>
#include <iostream>
#include <chrono>

// CPU □□□□
void cpu_matrix_mult(cv::Mat& a, cv::Mat& b, cv::Mat& result) {
    for (int i = 0; i < 50; i++) {
        result = a * b;
    }
}

// GPU □□□□
void gpu_matrix_mult(cv::cuda::GpuMat& d_a, cv::cuda::GpuMat& d_b, cv::cuda::GpuMat& d_result)
{
    cv::cuda::Stream stream;

    for (int i = 0; i < 50; i++) {
        cv::cuda::gemm(d_a, d_b, 1.0, cv::cuda::GpuMat(), 0, d_result, 0, stream);
        stream.waitForCompletion();
    }
}

int main() {
    try {
        std::cout << "--- OpenCV CUDA Matrix Multiplication Test ---\n";

        // □□□□ 1000x1000 □□□□□
        cv::Mat mat_a(1000, 1000, CV_32FC1);
        cv::Mat mat_b(1000, 1000, CV_32FC1);
        cv::randu(mat_a, 0.0f, 1.0f);
        cv::randu(mat_b, 0.0f, 1.0f);

        cv::Mat cpu_result;

        // CPU □□
        auto start_cpu = std::chrono::high_resolution_clock::now();
```

```
        cpu_matrix_mult(mat_a, mat_b, cpu_result);
        auto end_cpu = std::chrono::high_resolution_clock::now();
        double cpu_time = std::chrono::duration_cast<std::chrono::milliseconds>(end_cpu -
start_cpu).count();

        // GPU □□
        cv::cuda::GpuMat d_mat_a, d_mat_b, d_result;
        d_mat_a.upload(mat_a);
        d_mat_b.upload(mat_b);

        auto start_gpu = std::chrono::high_resolution_clock::now();
        gpu_matrix_mult(d_mat_a, d_mat_b, d_result);
        auto end_gpu = std::chrono::high_resolution_clock::now();
        double gpu_time = std::chrono::duration_cast<std::chrono::milliseconds>(end_gpu -
start_gpu).count();

        // □□□□□□□□□
        cv::Mat gpu_result;
        d_result.download(gpu_result);

        // □□□□(□□□□)
        double diff = cv::norm(cpu_result, gpu_result, cv::NORM_L2);
        std::cout << "Result difference: " << diff << "\n";

        std::cout << "Performance Results:\n"
                  << " - CPU time: " << cpu_time << " ms\n"
                  << " - GPU time: " << gpu_time << " ms\n"
                  << " - Speedup: " << cpu_time / gpu_time << "x\n";

        std::cout << "\n CUDA matrix multiplication test completed\n";
        return 0;

    } catch (const cv::Exception& e) {
        std::cerr << "OpenCV Error (" << e.err << "): " << e.what() << "\n";
        return -1;
    } catch (const std::exception& e) {
        std::cerr << "Standard Error: " << e.what() << "\n";
        return -2;
    }
}
```

□□□□

```
jetson@jetson-desktop:~/work$ g++ test_cuda.cpp -o test_cuda `pkg-config --cflags --libs opencv4`
jetson@jetson-desktop:~/work$ ./test_cuda
--- OpenCV CUDA Performance Test ---
Performance Results:
 - CPU time: 2451 ms
 - GPU time: 918 ms
 - Speedup: 2.66993x

□ CUDA performance test completed
```

????pytorch?torchvision

PyTorch 是 Python 中最流行、最易用的深度学习框架之一。它让开发者能够像操作普通 Python 代码一样，直观、灵活地设计和训练复杂的神经网络模型。其简洁的 API 设计和强大的 GPU 加速支持，使得从研究想法到实际部署的开发过程都极其高效便捷，广受开发者青睐。

NVIDIA 为 Jetson 系列设备专门适配了对应的软件包，其版本依赖关系如下：

PyTorch Version	NVIDIA Framework Container	NVIDIA Framework Wheel	JetPack Version
2.8.0a0+5228986c39	25.06	-	6.2
2.8.0a0+5228986c39	25.05	-	6.2
2.7.0a0+79aa17489c	25.04	-	6.2
2.7.0a0+7c8ec84dab	25.03	-	6.2
2.7.0a0+6c54963f75	25.02	-	6.2
2.6.0a0+ecf3bae40a	25.01	-	6.1
2.6.0a0+df5bbc09d1	24.12	-	6.1
2.6.0a0+df5bbc0	24.11	-	6.1
2.5.0a0+e000cf0ad9	24.10	-	6.1
2.5.0a0+b465a5843b	24.09	24.09	6.1
2.5.0a0+872d972e41	24.08	-	6.0
2.4.0a0+3bcc3cddb5	24.07	24.07	6.0
2.4.0a0+f70bd71a48	24.06	24.06	6.0
2.4.0a0+07cecf4168	24.05	24.05	6.0
2.3.0a0+6ddf5cf85e	24.04	24.04	6.0 Developer Preview
2.3.0a0+40ec155e58	24.03	24.03	
2.3.0a0+ebedce2	24.02	24.02	
2.2.0a0+81ea7a4	23.12, 24.01	23.12, 24.01	
2.2.0a0+6a974bec	23.11	23.11	

PyTorch Version	NVIDIA Framework Container	NVIDIA Framework Wheel	JetPack Version
2.1.0a		23.06	5.1.x
2.0.0		23.05	
2.0.0a0+fe05266f		23.04	
2.0.0a0+8aa34602		23.03	
1.14.0a0+44dac51c		23.02, 23.01	
1.13.0a0+936e930		22.11	5.0.2
1.13.0a0+d0d6b1f		22.09, 22,10	
1.13.0a0+08820cb	22.07	22.07	
1.13.0a0+340c412	22.06	22.06	5.0.1
1.12.0a0+8a1a93a9	22.05	22.05	5.0
1.12.0a0+bd13bc66		22.04	
1.12.0a0+2c916ef		22.03	
1.11.0a0+bfe5ad28		22.01	4.6.1

□□□□ **JetPack6.2.1 cuda12.6** □□□

1.??torch???

1.1?????torch , torchvison

```
wget https://pypi.jetson-ai-lab.io/jp6/cu126/+f/62a/1beee9f2f1470/torch-2.8.0-cp310-cp310-linux_aarch64.whl
wget https://pypi.jetson-ai-lab.io/jp6/cu126/+f/907/c4c1933789645/torchvision-0.23.0-cp310-cp310-linux_aarch64.whl
pip install torch-2.8.0-cp310-cp310-linux_aarch64.whl torchvision-0.23.0-cp310-cp310-linux_aarch64.whl -i https://pypi.tuna.tsinghua.edu.cn/simple
```

1.2 ????????

□□ python□□□□□□□

```
jetson@jetson-desktop:~$ python
Python 3.10.16 (main, Dec 11 2024, 16:18:56) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> import torch
>>> print(torch.__version__)
2.8.0
>>> print(torch.cuda.is_available())
True
```

2. ??YOLO11

YOLO
[illegible]
[illegible] YOLO
[illegible]

2.1 ??miniconda

```
curl -L https://repo.anaconda.com/miniconda/Miniconda3-py310_25.3.1-1-Linux-aarch64.sh | bash
source ~/.bashrc
conda --version
```

2.2 conda??

```
conda config --add channels https://mirrors.ustc.edu.cn/anaconda/pkgs/main/
conda config --add channels https://mirrors.ustc.edu.cn/anaconda/pkgs/free/
conda config --add channels https://mirrors.ustc.edu.cn/anaconda/cloud/conda-forge/
conda config --add channels https://mirrors.ustc.edu.cn/anaconda/cloud/msys2/
conda config --set show_channel_urls yes
```

2.3??conda??

```
conda create -n jetson-ai python=3.10
```

2.4 ??conda??

```
conda activate jetson-ai
```

2.5 ??torch?torchvison

```
wget https://pypi.jetson-ai-lab.io/jp6/cu126/+f/62a/1beee9f2f1470/torch-2.8.0-cp310-cp310-linux_aarch64.whl
wget https://pypi.jetson-ai-lab.io/jp6/cu126/+f/907/c4c1933789645/torchvision-0.23.0-cp310-cp310-linux_aarch64.whl
pip install torch-2.8.0-cp310-cp310-linux_aarch64.whl torchvision-0.23.0-cp310-cp310-linux_aarch64.whl -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2.6 ??ultralytics

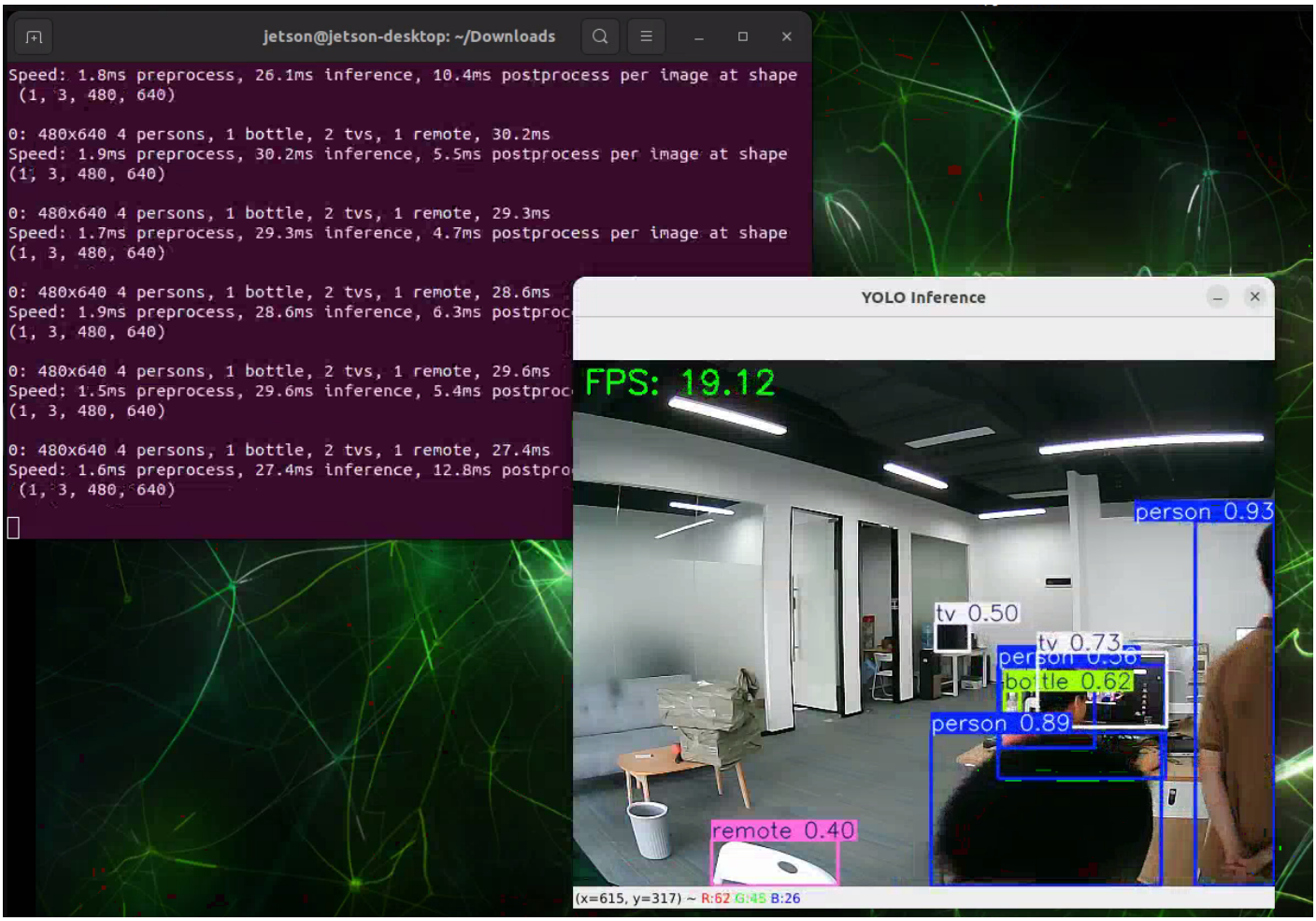
```
pip install ultralytics -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2.7 ???????????

接入摄像头，并在上面创建的环境中运行如下程序。

```
import cv2
import time
from ultralytics import YOLO
from ultralytics import YOLOWorld

# Load the YOLO model
model = YOLO("yolo11s.pt")

# Open the video file
video_path = 0
cap = cv2.VideoCapture(video_path)

# Loop through the video frames
while cap.isOpened():

    # Read a frame from the video
    success, frame = cap.read()
    start = time.time()
    if success:
        # Run YOLO inference on the frame
        results = model(frame)
        inf_time = time.time() - start
        # Visualize the results on the frame
        annotated_frame = results[0].plot()
        fps = 1.0 / inf_time if inf_time > 0 else 0
        # show FPS
        cv2.putText(annotated_frame, f"FPS: {fps:.2f}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,255,0), 2)
        cv2.imshow("YOLO Inference", annotated_frame)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
```

```
    else:
        # Break the loop if the end of the video is reached
        break

# Release the video capture object and close the display window
cap.release()
cv2.destroyAllWindows()
```



□□□□□□ [Ultralytics YOLO11 -Ultralytics YOLO □□]

????ollama

1. ????????ollama

```
curl -fsSL https://ollama.com/install.sh | sh
```

2. ????????

```
jetson@jetson-desktop:~$ ollama -v
ollama version is 0.9.6
```

3. ?????deepseek-r1 1.5b??

```
ollama run deepseek-r1:1.5b
```

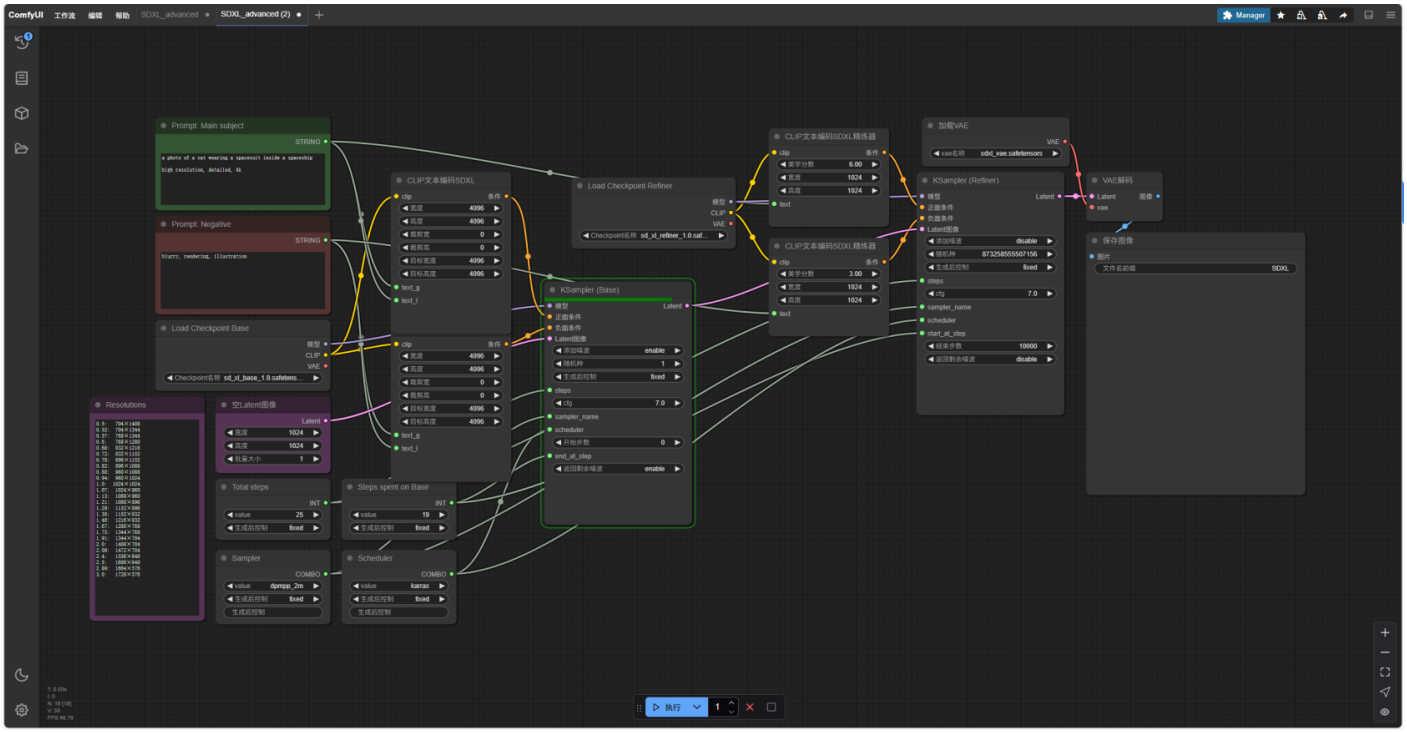
????Jetson-container

> [illegible] docker[illegible]

Jetson-container[NVIDIA[Jetson[illegible] Docker[illegible] CUDA[cuDNN[TensorRT [illegible] AI[illegible]

[illegible] **Jetson Orin NX 16GB[JetPack6.2.1 [illegible] Comfy-UI [illegible]** [illegible]

Comfy-UI [illegible] Stable Diffusion[illegible] LoRA[ControlNet[illegible]



1. [illegible] [jetson-container[illegible]

```
git clone https://github.com/dusty-nv/jetson-containers.git
cd jetson-containersbash
jetson-containers/install.sh
```

2. [illegible] docker[illegible]

```
jetson-containers run dustynv/comfyui:r36.4.3
```

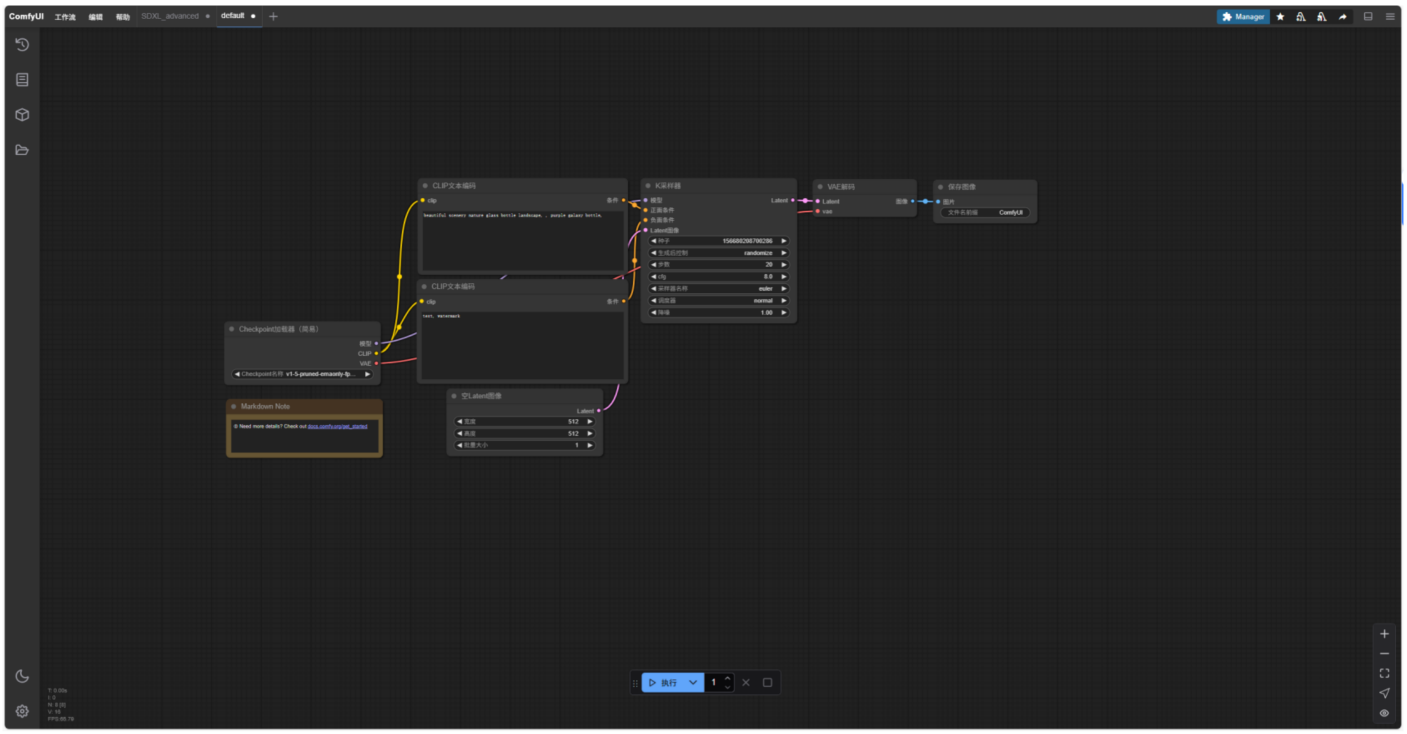
3. [illegible]

```
  File "<frozen importlib._bootstrap>", line 241, in _call_with_frames_removed
  File "/opt/ComfyUI/custom_nodes/ComfyUI-Crystools/__init__.py", line 18, in <module>
    from .nodes.image import CImagePreviewFromImage, CImageLoadWithMetadata, CImageGetResolution, CImagePrev
iewFromMetadata, \
  File "/opt/ComfyUI/custom_nodes/ComfyUI-Crystools/nodes/image.py", line 5, in <module>
    import piexif
ModuleNotFoundError: No module named 'piexif'

Cannot import /opt/ComfyUI/custom_nodes/ComfyUI-Crystools module for custom nodes: No module named 'piexif'
### Loading: ComfyUI-Manager (V3.30.4)
[ComfyUI-Manager] network_mode: public
### ComfyUI Version: v0.3.26-15-gbc219a64 | Released on '2025-03-11'
ComfyUI found: /opt/ComfyUI
'/opt/ComfyUI' added to sys.path

Import times for custom nodes:
   0.0 seconds: /opt/ComfyUI/custom_nodes/websocket_image_save.py
   0.0 seconds: /opt/ComfyUI/custom_nodes/comfyui-flux-accelerator
   0.1 seconds: /opt/ComfyUI/custom_nodes/ComfyUI-Manager
   0.1 seconds: /opt/ComfyUI/custom_nodes/ComfyUI-to-Python-Extension
   0.7 seconds (IMPORT FAILED): /opt/ComfyUI/custom_nodes/ComfyUI-Crystools

Starting server

To see the GUI go to: http://0.0.0.0:8188
[ComfyUI-Manager] default cache updated: https://raw.githubusercontent.com/ltdrdata/ComfyUI-Manager/main/alt
er-list.json
[ComfyUI-Manager] default cache updated: https://raw.githubusercontent.com/ltdrdata/ComfyUI-Manager/main/git
hub-stats.json
[ComfyUI-Manager] default cache updated: https://raw.githubusercontent.com/ltdrdata/ComfyUI-Manager/main/cus
tom-node-list.json
[ComfyUI-Manager] default cache updated: https://raw.githubusercontent.com/ltdrdata/ComfyUI-Manager/main/mod
el-list.json
[ComfyUI-Manager] default cache updated: https://raw.githubusercontent.com/ltdrdata/ComfyUI-Manager/main/ext
ension-node-map.json
FETCH ComfyRegistry Data: 5/94
FETCH ComfyRegistry Data: 10/94
FETCH ComfyRegistry Data: 15/94
FETCH ComfyRegistry Data: 20/94
FETCH ComfyRegistry Data: 25/94
FETCH ComfyRegistry Data: 30/94
```

4. □□□□□□□ GUI□□

□□ Jetson□□□□□□□□ http://0.0.0.0:8188

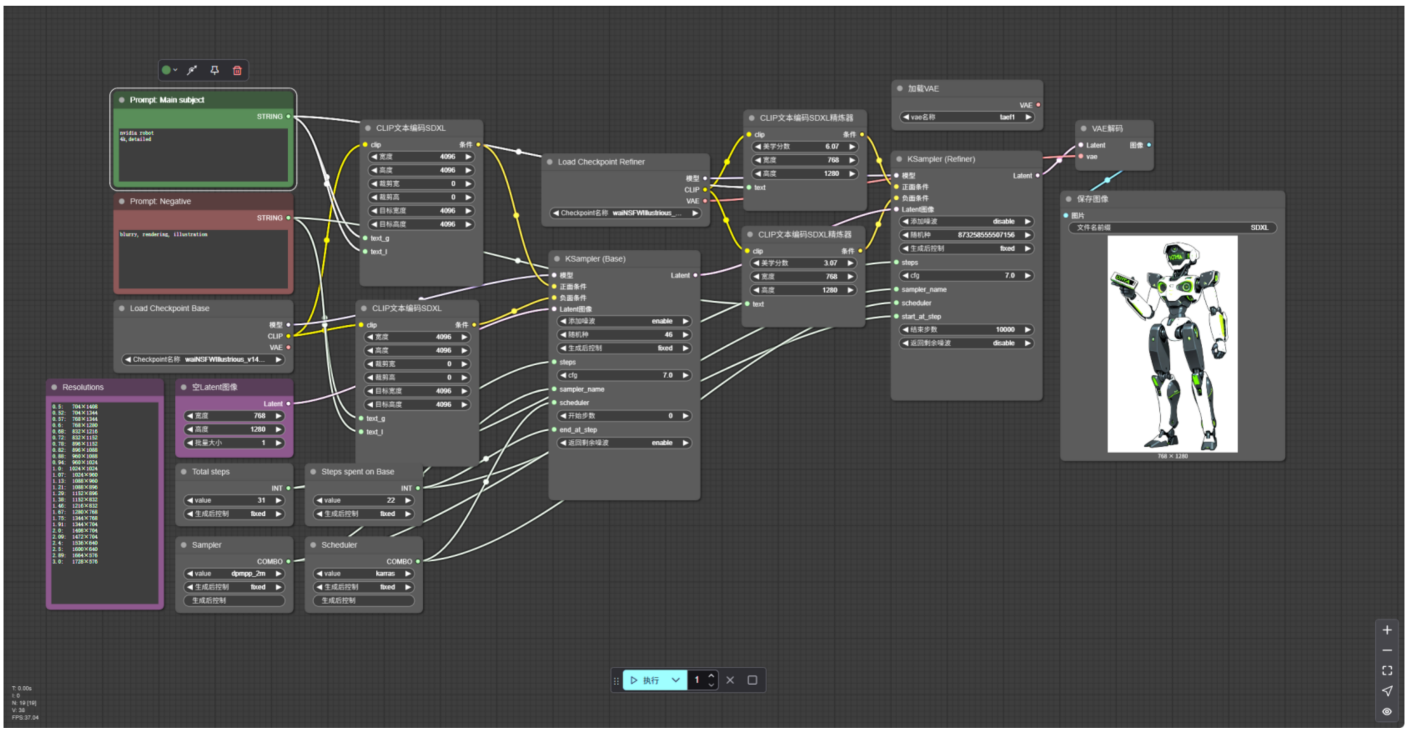
□□□□□□□□□□□□□□□ http://<jetson□□ IP>:8188



5. 设定映射路径，将开发板里下载好的模型映射到容器内部目录

```
jetson-containers run ~/models/:/opt/ComfyUI/models/checkpoints dustynv/comfyui:r36.4.3
```

6. 按照需求搭建工作流并调整参数生成图像



关于 ComfyUI的使用教程请参考其官网 ComfyUI | 用 AI生成视频、图像、音频

??TensorRT??

c++

python□□

????

4G??????

移远通信 LTE Standard EM05系列是一款专为 IoT/M2M应用而设计的 LTE Cat 4模块。采用节省空间的 M.2（NGFF）封装类型，超薄、紧凑的设计使其更易于嵌入到小尺寸产品中。

EM05系列支持最大下行速率 150 Mbps，最大上行速率 50 Mbps,包含三个型号: EM05-CN、EM05-E和 EM05-G。支持 LTE-FDD、LTE-TDD、DC-HSDPA、HSPA+、HSDPA、HSUPA、WCDMA和 CDMA 等多种网络制式。

要在 Jetson系列上使用 EM-05模块需要重新编译系统内核并刷入新编译的系统。

?????????

1.1 ????????

1.2 ?????????

1.3 ????????????

1.4 ?????????

????????????

2.1 ????????

- 安装编译依赖工具

```
sudo apt install build-essential bc
sudo apt-get install git bison flex libssl-dev zip libncurses-dev make git
sudo apt-get install build-essential bc
```

- 创建工具链放置目录

```
mkdir $HOME/l4t-gcc-toolchain
cd $HOME/l4t-gcc-toolchain
```

- 下载并解压交叉编译工具链

```
wget https://developer.nvidia.com/downloads/embedded/l4t/r36_release_v3.0/toolchain/aarch64--glibc--stable-2022.08-1.tar.bz2
tar xf aarch64--glibc--stable-2022.08-1.tar.bz2
```

- 进入到缓存好的刷机固件目录

```
cd
$HOME/nvidia/nvidia_sdk/JetPack_6.2.1_Linux_JETSON_ORIN_NANO_TARGETS/Linux_for_Tegra/source  #
根据实际目录进行修改
```

- 根据 release tag 拉取源码。 Jetson Linux Archive | NVIDIA Developer

```
 ./source_sync.sh -k -t <release-tag> #<release-tag>替换为你自己的发行版本如 jetson_36.4.3
```

2.2 ??????

> 本部分内容可参考 移远__Linux&Andriod驱动技术手册_____

Jetson Orin Nano配置文件路径: arch/arm64/configs/defconfig

- 配置 USB转串口驱动添加如下内容

```
CONFIG_USB_SERIAL=y
CONFIG_USB_SERIAL_WWAN=y
CONFIG_USB_SERIAL_OPTION=y
CONFIG_USB_NET_DRIVERS=y
CONFIG_USB_USBNET=y
CONFIG_USB_NET_QMI_WWAN=y
CONFIG_USB_WDM=y
```

- 修改 drivers/usb/serial/option.c, 向 USB转串口驱动添加 VID和 PID

```
static const struct usb_device_id option_ids[] = {
#if 1 // 2025-04-24 Added by Quectel
    { USB_DEVICE(0x2C7C, 0x0125) },
#endif
    ... ...
}
```

- 使用 USBNet驱动, 文件路径: drivers/usb/serial/option.c

```
static int option_probe(struct usb_serial *serial,
            const struct usb_device_id *id)
{
    ... ...
```

```
#if 1  // 2025-04-24 Added by Quectel
if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
        __u16 idProduct = le16_to_cpu(serial->dev->descriptor.idProduct);
        struct usb_interface_descriptor *intf = &serial->interface->cur_altsetting->desc;

        if (intf->bInterfaceClass != 0xFF || intf->bInterfaceSubClass == 0x42) {
                //ECM, RNDIS, NCM, MBIM, ACM, UAC, ADB
                return -ENODEV;
        }

        if ((idProduct&0xF000) == 0x0000) {
                //MDM interface 4 is QMI
                if (intf->bInterfaceNumber == 4 && intf->bNumEndpoints == 3
                        && intf->bInterfaceSubClass == 0xFF && intf->bInterfaceProtocol ==
0xFF)
                        return -ENODEV;
        }
}
#endif

    /* Store the device flags so we can use them during attach. */
    usb_set_serial_data(serial, (void *)device_flags);

    return 0;
}
```

- □□□□□□□□□□□□ drivers/usb/serial/usb_wwan.c

```
static struct urb *usb_wwan_setup_urb(struct usb_serial_port *port,
                      int endpoint,
                      int dir, void *ctx, char *buf, int len,
                      void (*callback) (struct urb *))
{
 ... ...
    usb_fill_bulk_urb(urb, serial->dev,
              usb_sndbulkpipe(serial->dev, endpoint) | dir,
              buf, len, callback, ctx);

#if 1   //2025-04-24 Added by Quectel for zero packet
```

```
    if (dir == USB_DIR_OUT) {
        struct usb_device_descriptor *desc = &serial->dev->descriptor;

        if (desc->idVendor == cpu_to_le16(0x2C7C))
        urb->transfer_flags |= URB_ZERO_PACKET;
    }
#endif
    return urb;
}
```

- 添加 Reset-resume机制（休眠唤醒）文件路径: drivers/usb/serial/option.c

```
static struct usb_serial_driver option_1port_device = {
    ... ...
#ifdef CONFIG_PM
    .suspend           = usb_wwan_suspend,
    .resume            = usb_wwan_resume,
#if 1  //2025-04-24 Added by Quectel
    .reset_resume   = usb_wwan_resume,
#endif
#endif
};
```

- 添加 QMI_WWAN驱动
  移远提供了 QMI_WWAN驱动源文件 qmi_wwan_q.c，将其复制到 drivers/net/usb/ 目录下。同时修改 drivers/net/usb/Makefile，使其能编译 qmi_wwan_q.c

```
#Makefile
# must insert qmi_wwan_q.o before qmi_wwan.o
obj-${CONFIG_USB_NET_QMI_WWAN} += qmi_wwan_q.o
obj-${CONFIG_USB_NET_QMI_WWAN} += qmi_wwan.o
```

2.3 ????

> 此步骤可能需要进行半个小时以上

- 创建输出目录

```
cd ../../
mkdir kernel_out
```

- 编译安装内核

```
export CROSS_COMPILE=$HOME/l4t-gcc-toolchain/aarch64--glibc--stable-2022.08-1/bin/aarch64-buildroot-linux-gnu-         #设置交叉编译工具
make -C kernel                          #构建 Jetson Linux 内核镜像
sudo -E make install -C kernel      #安装内核模块和树内模块
```

- 将内核镜像复制到刷机目录下

```
cp kernel/kernel-jammy-src/arch/arm64/boot/Image ../Linux_for_Tegra/kernel/Image
```

- 构建 NVIDIA树外模块（驱动程序）

```
export CROSS_COMPILE=$HOME/l4t-gcc-toolchain/aarch64--glibc--stable-2022.08-1/bin/aarch64-buildroot-linux-gnu-
export KERNEL_HEADERS=$PWD/kernel/kernel-jammy-src
make modules
```

- 安装到刷机目录下

```
export INSTALL_MOD_PATH=$HOME/nvidia/nvidia_sdk/JetPack_6.2.1_Linux_JETSON_ORIN_NANO_TARGETS/Linux_for_Tegra/rootfs/ #根据实际目录进行修改
sudo -E make modules_install
```

- 开始构建 DTB

```
export CROSS_COMPILE=$HOME/l4t-gcc-toolchain/aarch64--glibc--stable-2022.08-1/bin/aarch64-buildroot-linux-gnu-
export KERNEL_HEADERS=$PWD/kernel/kernel-jammy-src
make dtbs
```

- 将生成的 dtb文件拷贝到刷机目录

```
cp kernel-devicetree/generic-dts/dtbs/* /Linux_for_Tegra/kernel/dtb/
```

2.4 ????

- 进入刷机固件缓存目录

```
cd $HOME/nvidia/nvidia_sdk/JetPack_6.2.1_Linux_JETSON_ORIN_NANO_TARGETS/Linux_for_Tegra/source  #
```

根据实际目录进行修改

使用命令行刷机:

- super模式

```
sudo ./tools/kernel_flash/l4t_initrd_flash.sh --external-device nvme0n1p1 -c
tools/kernel_flash/flash_l4t_t234_nvme.xml -p "-c bootloader/generic/cfg/flash_t234_qspi.xml"
--showlogs --network usb0 jetson-orin-nano-devkit-super internal
```

- 普通模式

```
sudo ./flash.sh jetson-orin-nano-devkit-nvme internal
```

2.5 ????

- 将 Quectel_Qconnectmanager上传到开发板
- 解压并编译

```
unizp Quectel_QConnectManager_Linux_V1.6.5.zip
cd Quectel_QConnectManager_Linux_V1.6.5
make
```

成功运行便可　　4G拨号上网

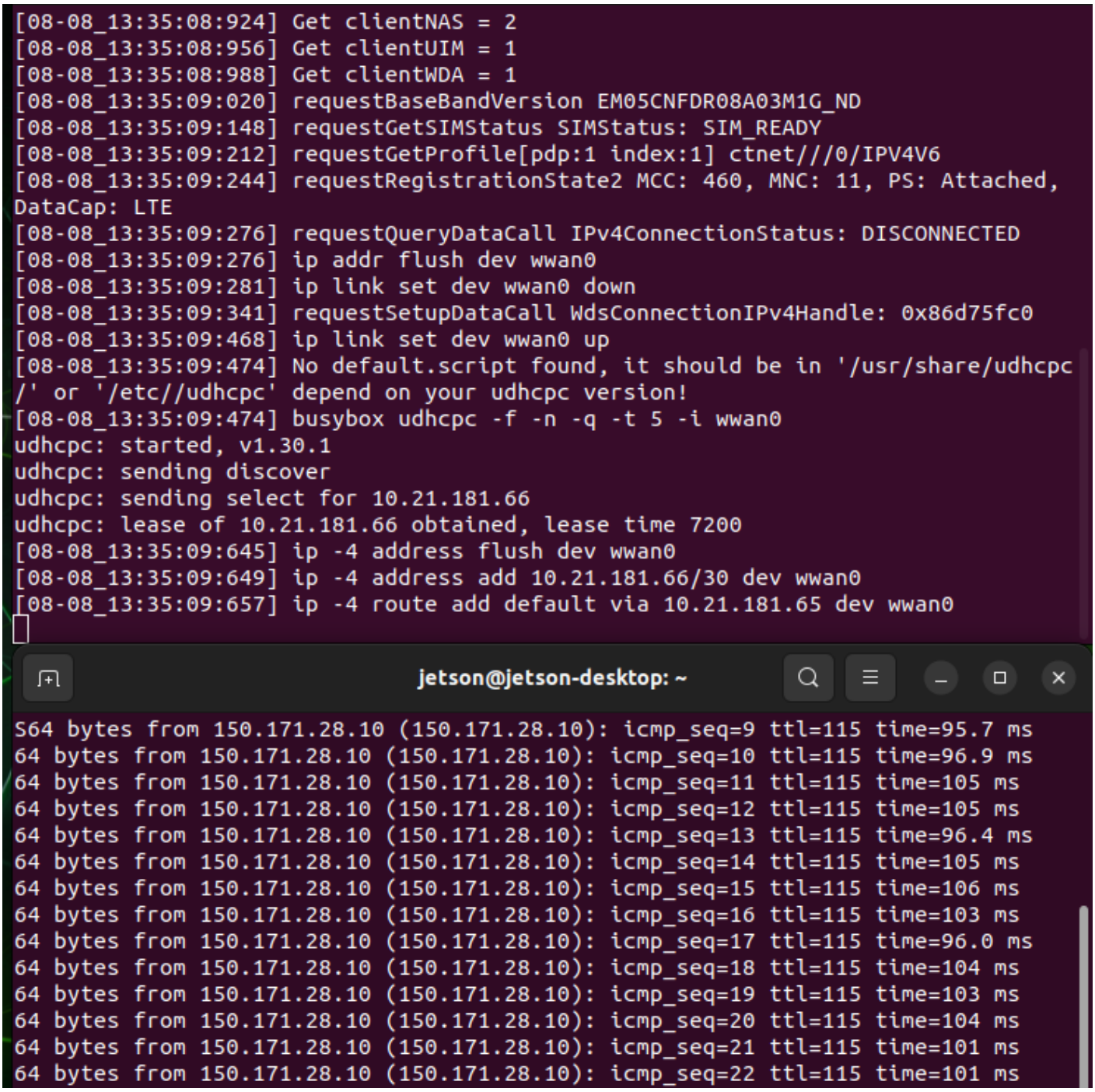
```
jetson@jetson-desktop:~/Downloads/Quectel_QConnectManager_Linux_V1.6.5$ sudo ./quectel-CM
[08-08_11:39:04:415] QConnectManager_Linux_V1.6.5
[08-08_11:39:04:416] Find /sys/bus/usb/devices/1-2.2 idVendor=0x2c7c idProduct=0x125,
bus=0x001, dev=0x007
[08-08_11:39:04:416] Auto find qmichannel = /dev/cdc-wdm0
[08-08_11:39:04:417] Auto find usbnet_adapter = wwan0
[08-08_11:39:04:417] netcard driver = qmi_wwan_q, driver version = V1.2.6
[08-08_11:39:04:417] Modem works in QMI mode
[08-08_11:39:04:447] cdc_wdm_fd = 7
[08-08_11:39:04:523] Get clientWDS = 5
[08-08_11:39:04:557] Get clientDMS = 1
[08-08_11:39:04:589] Get clientNAS = 2
[08-08_11:39:04:620] Get clientUIM = 1
[08-08_11:39:04:653] Get clientWDA = 1
[08-08_11:39:04:684] requestBaseBandVersion EM05CNFDR08A03M1G_ND
[08-08_11:39:04:812] requestGetSIMStatus SIMStatus: SIM_READY
[08-08_11:39:04:876] requestGetProfile[pdp:1 index:1] ctnet///0/IPV4V6
```

```
[08-08_11:39:04:908] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached, DataCap: LTE
[08-08_11:39:04:940] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[08-08_11:39:04:941] ip addr flush dev wwan0
[08-08_11:39:04:947] ip link set dev wwan0 down
[08-08_11:39:05:003] requestSetupDataCall WdsConnectionIPv4Handle: 0x8723e530
[08-08_11:39:05:132] ip link set dev wwan0 up
[08-08_11:39:05:141] No default.script found, it should be in '/usr/share/udhcpc/' or
'/etc//udhcpc' depend on your udhcpc version!
[08-08_11:39:05:142] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.30.1
udhcpc: sending discover
udhcpc: sending select for 10.21.181.66
udhcpc: lease of 10.21.181.66 obtained, lease time 7200
[08-08_11:39:05:282] ip -4 address flush dev wwan0
[08-08_11:39:05:286] ip -4 address add 10.21.181.66/30 dev wwan0
[08-08_11:39:05:292] ip -4 route add default via 10.21.181.65 dev wwan0
```

□□□□□

```
jetson@jetson-desktop:~$  ifconfig wwan0
wwan0: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet 10.21.181.66  netmask 255.255.255.252
        inet6 fe80::5804:41ff:feda:ce83  prefixlen 64  scopeid 0x20<link>
        ether 5a:04:41:da:ce:83  txqueuelen 1000  (Ethernet)
        RX packets 9  bytes 2304 (2.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 22  bytes 1854 (1.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

□□□□


[08-08_13:35:08:924] Get clientNAS = 2
[08-08_13:35:08:956] Get clientUIM = 1
[08-08_13:35:08:988] Get clientWDA = 1
[08-08_13:35:09:020] requestBaseBandVersion EM05CNFDR08A03M1G_ND
[08-08_13:35:09:148] requestGetSIMStatus SIMStatus: SIM_READY
[08-08_13:35:09:212] requestGetProfile[pdp:1 index:1] ctnet///0/IPV4V6
[08-08_13:35:09:244] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached,
DataCap: LTE
[08-08_13:35:09:276] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[08-08_13:35:09:276] ip addr flush dev wwan0
[08-08_13:35:09:281] ip link set dev wwan0 down
[08-08_13:35:09:341] requestSetupDataCall WdsConnectionIPv4Handle: 0x86d75fc0
[08-08_13:35:09:468] ip link set dev wwan0 up
[08-08_13:35:09:474] No default.script found, it should be in '/usr/share/udhcpc
/' or '/etc//udhcpc' depend on your udhcpc version!
[08-08_13:35:09:474] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.30.1
udhcpc: sending discover
udhcpc: sending select for 10.21.181.66
udhcpc: lease of 10.21.181.66 obtained, lease time 7200
[08-08_13:35:09:645] ip -4 address flush dev wwan0
[08-08_13:35:09:649] ip -4 address add 10.21.181.66/30 dev wwan0
[08-08_13:35:09:657] ip -4 route add default via 10.21.181.65 dev wwan0
jetson@jetson-desktop: ~
S64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=9 ttl=115 time=95.7 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=10 ttl=115 time=96.9 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=11 ttl=115 time=105 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=12 ttl=115 time=105 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=13 ttl=115 time=96.4 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=14 ttl=115 time=105 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=15 ttl=115 time=106 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=16 ttl=115 time=103 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=17 ttl=115 time=96.0 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=18 ttl=115 time=104 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=19 ttl=115 time=103 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=20 ttl=115 time=104 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=21 ttl=115 time=101 ms
64 bytes from 150.171.28.10 (150.171.28.10): icmp_seq=22 ttl=115 time=101 ms