图为 TW-T902 用户手册

TWOWIN TW-T902 USER MANUAL

让世界更智慧 让生活更美好

图为信息科技(深圳)有限公司

TWOWIN TECHNOLOGY Co. Ltd

图为信息科技（深圳）有限公司

目录

图为 TW-T902 用户手册 1
TWOWIN TW-T902 USER MANUAL 1
目录 1
文档修订目录 2
文档版本 2
Version1.0 2
安全警示及使用注意事项 4
简介 5
产品规格 6
处理器及核心模块 6
编解码 6
接口 7
供电 8
结构 8
环境 8
服务与支持 9
功能介绍 10
系统介绍 10
系统重刷 10
开关机 10
工作模式切换 11
接口说明 12
正面接口 12
背面接口 13
CAN 功能测试 15
WiFi 连接 17
4G、5G 使用 nmcli 工具拨号联网 20
采用 M.2 Key m SSD 为系统盘 26
Jtop 安装 37

文档修订目录

文档版本

Version1.0

文档版本号	修订日期	修订内容
V1.0	2022/11/25	初始发布

前言

在使用本手册之前，请您认真阅读以下使用许可协议，只有在同意以下使用许可协议的情况下方能使用本手册中介绍的产品。

版权声明

免责条款

本文档依据现有信息制作，其内容如有更改，恕不另行通知。图为信息科技(深圳)有限公司在编写该文档的时候已尽最大努力保证其内容准确可靠，但图为信息科技(深圳)有限公司不对本文档中的遗漏、不准确、或错误导致的损失和损害承担责任。

技术支持与信息反馈

如果您在使用我们的产品时遇到问题,或者您认为我们的产品有某些功能缺陷,请访问我们的官网 https://www.twowin.com 联系我们的客服,我们将为您解决问题和反馈;或者需要技术支持指导以及有任何宝贵意见，也请您通过官网或者电话联系我们：

联系人 ： 技术支持

手机：19925328992

电话：0755-82840481

网址：www.twowinit.com

地址：深圳市南山区科技南十二路长虹科技大厦 2512

安全警示及使用注意事项

- 安全说明

在使用本产品之前，必须先查阅本文档,对该产品有初步的认识与了解,且须遵守本产品使用手册中的安全说明以保证您的个人安全并避免损坏设备,若盲目操作造成损失或伤害,制造商对其错误操作造成的设备及个人生命财产安全的任何问题均不负责。

- 电源电压

TW-T902 边缘计算平台输入端电源稳定可靠，功率 25W-60W。

电源范围： 19- 24 V DC；电流：5A(MAX)

- 环境要求：

工作温度： -20℃ - +60℃

通风要求： 计算平台安装的周边必须有良好通风的条件。

- 接地要求

电源适配器的供电源必须有良好的接地，特殊情况需安装计算平台上接地螺丝接地。

- 静电防护

电子元件和电路对静电放电很敏感，虽然本公司在设计电路板卡产品时会对板卡上的主要接口做防静电保护设计，但很难对所有元件及电路做到防静电安全防护。因此在处理任何电路板组件时，建议遵守防静电安全保护措施。防静电安全保护措施包括，但不限于以下几点：

- 运输、存储过程中应将盒子放在防静电袋中，直至安装部署时再拿出板卡;
- 在身体接触盒子之前应将身体内寄存的静电释放掉：佩戴放电接地腕带;
- 仅在静电放点安全区域内操作盒子;
- 避免在铺有地毯的区域搬移盒子。

- 操作与维护

操作或维护人员需先经培训合格，方可参与操作或维护。

- 其他

注意:由于本机接口驱动是由我司研发,与 nvidia 开发板接口驱动不同,使用 upgrade 这个指令会升级内核,覆盖设备树,如果必须使用 upgrade 进行更新,在使用之前,请运行以下命令

```
sudo mv /etc/apt/sources.list.d/nvidia-l4t-apt-source.list /etc/apt/sources.list.d/nvidia-l4t-apt-source.list.bak
sudo apt-get update
sudo apt-get upgrade
```

默认用户名:nvidia

默认密码:nvidia

切换 root 用户:sudo -s 回车后输入 nvidia

简介

TW-T902 为一款基于 NVIDIA® Jetson AGX Orin Series 系列模块设计的计算平台，内置集成 Jetson AGX Orin 模块，预装 Ubuntu 20.04 操作系统，具备 200TOPS 的 AI 处理能力，支持电源适配器种电源输入方式，超强固轻型铝合金材料设计，有风扇结构传导主动散热，尺寸轻巧外观新颖，丰富 IO 接口类型，预留便于现场安装的底部支架，具备超长 MTBF 稳定运行能力，可应用于机器人、无人配送车、低空防御、智能巡检、智慧楼宇等自主化机器，是边缘端部署 AI 算力进行深度学习的理想载体。

TW-T902 边缘计算平台概述

- 内嵌 NVIDIA® Jetson AGX Orin
- M.2 KEY M (PCIe4.0 NVMe 2280)
- 支持 M.2 KEY E (PCIEX1 2230)
- 支持多种接口(如 CAN/USB/网口/串口/GPIO 等)
- 支持双频 WiFi/4G/5G 模组
- 支持 2x1000/10G BASE-T RJ45 端口
- 风扇主动散热设计
- 内置 Ubuntu 20.04 系统和 Jetpack SDKs
- 宽压 12-24V DC 电源输入.

产品规格

处理器及核心模块

Processor	**NVIDIA　AGX Orin**
CPU	8-core NVIDIA Arm® Cortex A78AE v8.2 64-bit CPU 2MB L2 + 4MB L3
CPU Max Frequency	2.2 GHz
GPU	1792-core NVIDIA Ampere GPU with 56 Tensor Cores
GPU Max Frequency	939 MHz
Memory	32 GB 256-bit LPDDR5
DI accelerator	2 x NVDLA x 2.0
DLA Max Frequency	1.4 GHz
Storage	64GB eMMC 5.1 1 x m.2 key m nvme 2280(选配安装)

编解码

Video Encode	1x 4K60 (H.265) 3x 4K30 (H.265) 6x 1080p60 (H.265) 12x 1080p30 (H.265)
Video Decode	1x 8K30 (H.265) 2x 4K60 (H.265) 4x 4K30 (H.265) 9x 1080p60 (H.265) 18x 1080p30 (H.265)

接口

	Interface	Quantity	Note
Network	Ethernet	2×RJ45 Gigabit Network port	1x GbE 1x 10GbE
	WIFI	1	2.4G/5.8G 300Mbps
	4G	1	有方 N720/移远 EC20CEFASG(选配安装)
	5G	1	移远 RM500Q-CN(选配安装)
Video output	HDMI	1x HDMI 2.0	不支持转接信号(例如 vga 转 hdmi 形式)
USB	USB	2×USB 3.2-2 TYPE A	USB 5V, 1A
I/O	USB-OTG	1xUSB2.0 TYPE-C	刷机接口
	GPIO	2xGPIO	独立 3.3V TTL 电平 GPIO
	CAN	2xCAN 2.0b	With CAN chip
	UART	1xRS232 1xRS485	/dev/ttyTHS0 /dev/ttyTHS4
	DEBUG	1	调试串口
	M.2	1×M.2 M Key	PCIE NVME 2280 SSD (选配安装)
Function Key	Reset KEY	1	Button
	Recovery KEY	1	Button
LED	Power	1	电源指示灯红色常亮
	Running	1	运行指示灯绿色常亮

供电

Power Supply	Spec
Input Type	1xDC
Input Voltage	Wide input 12-24V DC
Maximum power	60W

结构

Mechanical	Spec
Dimensions (W×D×H)	132mm×133mm ×51mm
Weight	980g

环境

Environmental	Spec
Operating Temperature	-20℃-60℃
Storage Humidity	5%-95% non-condensing

服务与支持

技术支持

如果您遇到问题，或者您认为您的产品有缺陷，请带着您的问题访问我们官网,浏览我们常见问题一栏以查找常见问题的解决方案,也可电话或微信联系我们，我们会及时根据您的需求做出相应的工作安排,为您排忧解难。

保修

保修期：图为设备保修期为自购买之日起一年。 保修条例：保修期内产品，若出现非人为损坏的故障图为将进行免费保修。请通过购买平台客服对话以及电话联系获取保修协助(详情请参考图为信息科技(深圳)有限公司保修条例)。

功能介绍.

系统介绍

M902 设备采用且预装了 Ubuntu20.04 系统，上电即可登录。

用户名：nvidia　密码：nvidia

默认未设置 root 用户名和密码,如需进入 root 用户,请执行如下命令进行操作:

sudo -s

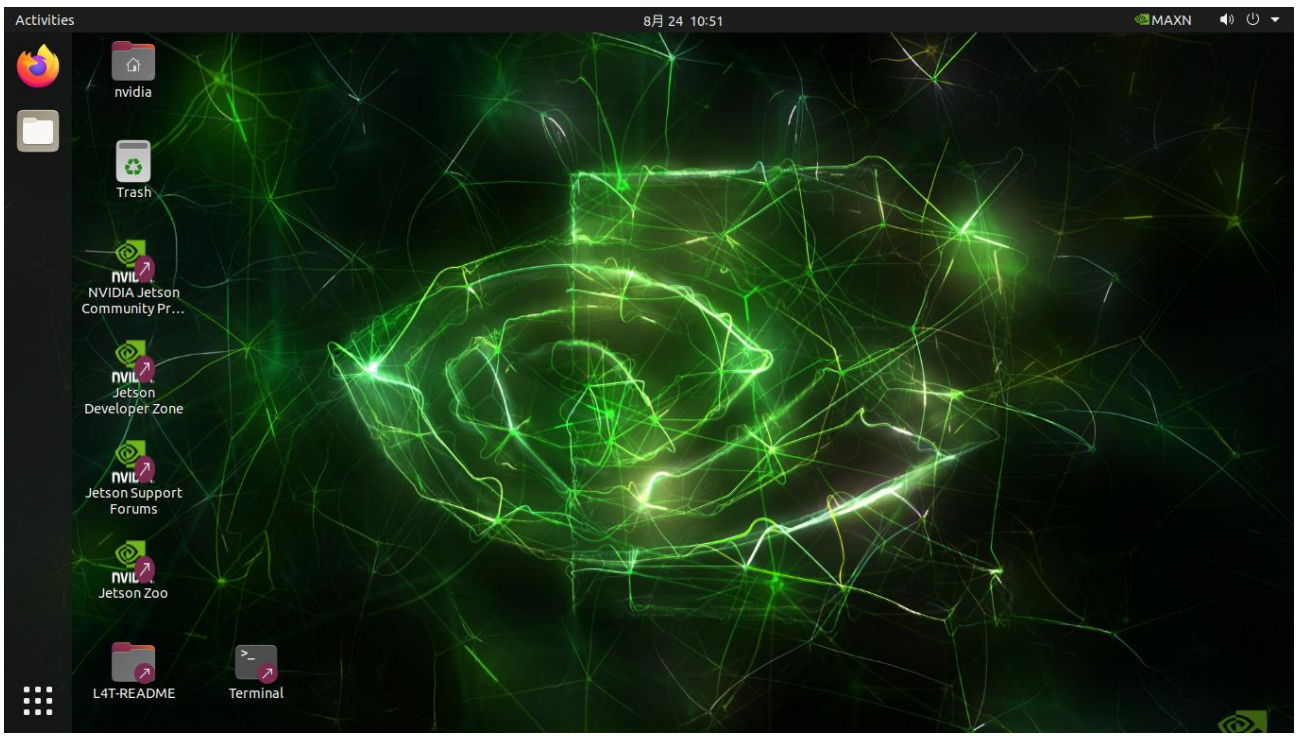
输入密码:nvidia

系统重刷

如遇特殊原因需要重刷系统，请联系技术支持人员协助。

开关机

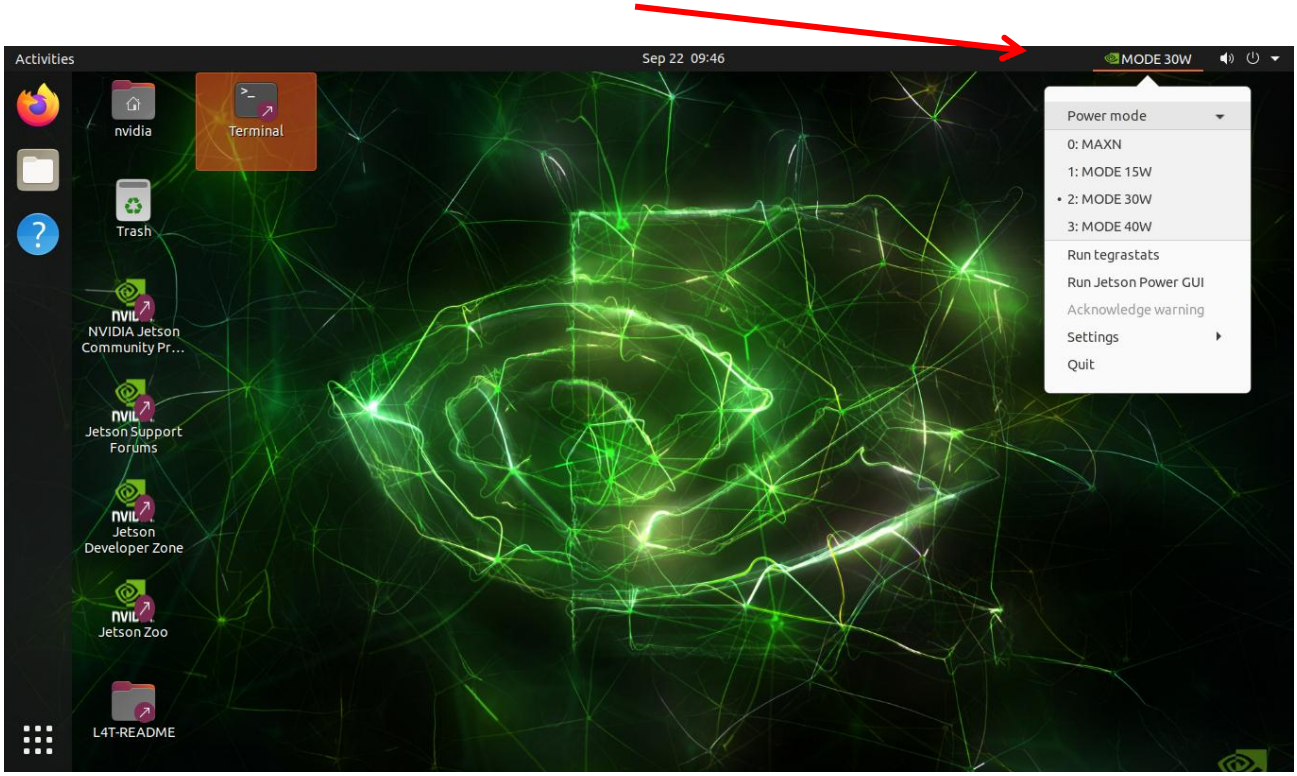
开机：TW-T902 设备默认开机模式为上电自启动。插入电源，并将显示器通过 HDMI 接口与设备相连，开机画面如图所示：



开机画面

工作模式切换

不同工作模式，使用 cpu 核心和功率不同，可按照自身需求进行选择，点击箭头所指选项，进行模式选择



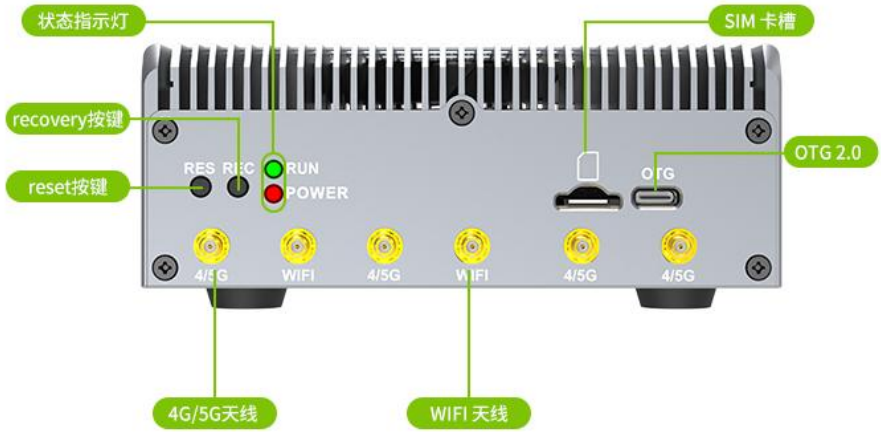
接口说明

正面接口

TW-T902 正面接口示意图

接口	接口说明
DC	12-24V 宽压输入
HDMI	HDMI 2.0
LAN1	Alternative RJ45 and waterproof port
LAN2	Alternative RJ45 and waterproof port
USB1/2	10G/5V-2A（<10W）
多功能接口	CAN.GPIO,RS232,RS485,DEBUG

背面接口



TW-T602 背面接口示意图

接口	接口说明
OTG	USB 2.0,可用于刷机以及连接 ssh
SIM	Sim 卡槽,支持手机卡与物联网卡,全网通
4/5G	外接 4G/5G 天线
WIFI	外接 WIFI 天线
RUNNING	运行指示灯
POWER	电源指示灯
RES	复位键
REC	单按无明显作用,配合复位键进行刷机

注:recover 模式:即刷机模式,主要用于重新安装系统以及使用 SDK manager 安装部分 sdk,本设备进入 recover 模式的方法为:先按住 REC 按键,不松手再按住 RES 按键,2 秒后松开 RES 按键,最后松开 REC 按键,主机终端输入 lsusb 查看是否有 nvidia corp,有则表示进入成功,无则检查 type-c 数据线是否连接好,以及主机端 usb 是否连接好,按键顺序及时间长短是否正确,主机端 usb 接口建议使用 usb3.0 接口.

多功能连接器 PIN 定 义			
PIN#	定义	PIN#	定义
1	CAN_H0	2	CAN_L0
3	CAN_H1	4	CAN_L1
5	RS485-A	6	RS485-B
7	GND	8	GND
9	RS232_RX	10	RS232_TX
11	IO(gpio326)	12	IO(gpio348)
13	DEBUG_RX	14	DEBUG_TX

CAN 功能测试

1、 can_test 目录

can_test 目录下包含如下文件：

canSendTest ：CAN 通信发送测试可执行文件

canRecvTest ： CAN 通信接收测试可执行文件

canUp.sh ： CAN 通信驱动加载脚本

ko 文件：SPI-CAN 驱动模块

2、 驱动加载，设置并设能 CAN 接口

在当前目录下运行命令

sudo ./canUp.sh

在运行结束后，打印“set can0 up success!!!” ，表示 can0 接口已设置并启用。

如下图，运行后，有 can0 、 can1 、can2 可以使用。

注意 bitrate 波特率。

```
nvidia@tegra-ubuntu:~/nfs$ sudo ./canUp.sh
[sudo] password for nvidia:
----------print arguments
bitrate = 500000
dbitrate = 2000000
spiCan = mcp251xfd
----------print end

----------set can down
set can0 down !!!
set can1 down !!!
set can2 down !!!

----------rmmoding CAN driver ...
rmmod: mcp251xfd
rmmod: mttcan
rmmod: can_raw
rmmod: can
----------rmmod CAN driver end

----------Loading CAN driver ...
modprobe: can
modprobe: can_raw
modprobe: mttcan
insmod: mcp251xfd
----------Loaded CAN driver end

----------set can up
set can0 up success!!!
set can1 up success!!!
set can2 up success!!!
bitrate=500000
dbitrate=2000000
spiCan=mcp251xfd
nvidia@tegra-ubuntu:~/nfs$
```

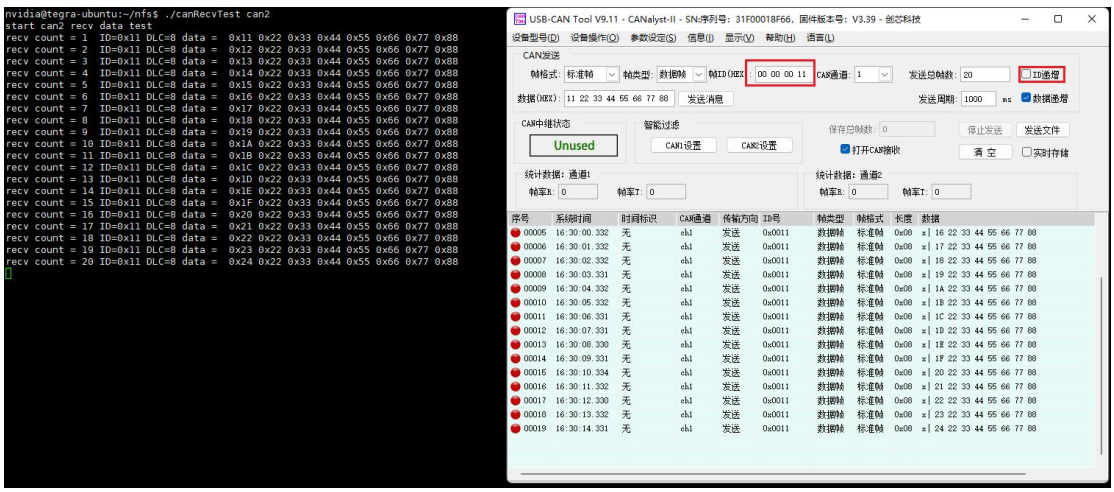
3、 can 接收测试（以 can2 测试为例）

运行命令 ./canRecvTest can2。此测试 demo 默认接收帧 ID 为 0x11。

运行命令后等待 can2 接收数据。

在 USB-CAN 工具设置帧 ID 为 “00 00 00 11“，取消勾选 ID 递增。设置波特率为上一步获取的 bitrate。点击“发送消息“ 按钮，发送数据到 CAN 总线。

在终端窗口即可接收到从 USB-CAN 发送的数据。



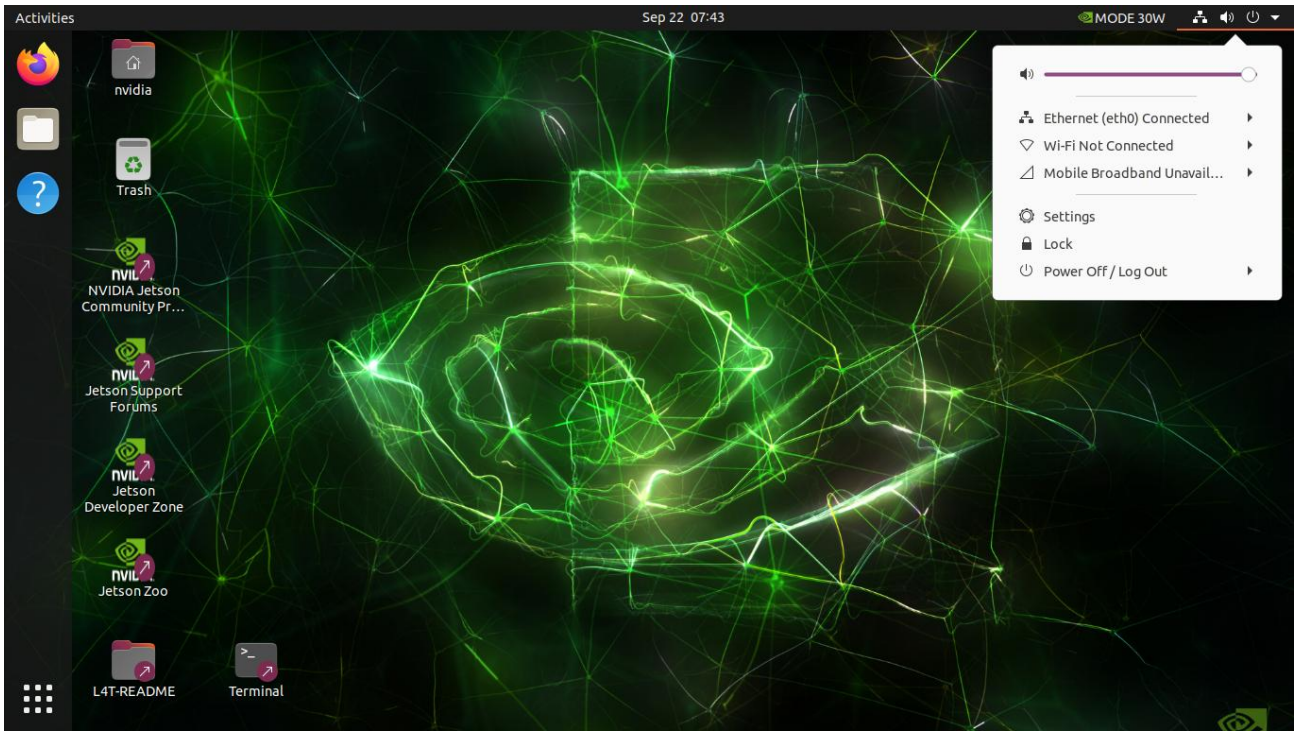
4、 can 发送测试（以 can2 测试为例）

运行命令 ./canSendTest can2。此测试 demo 默认交替发送帧 ID 围殴 0x11、0x22 的数据。

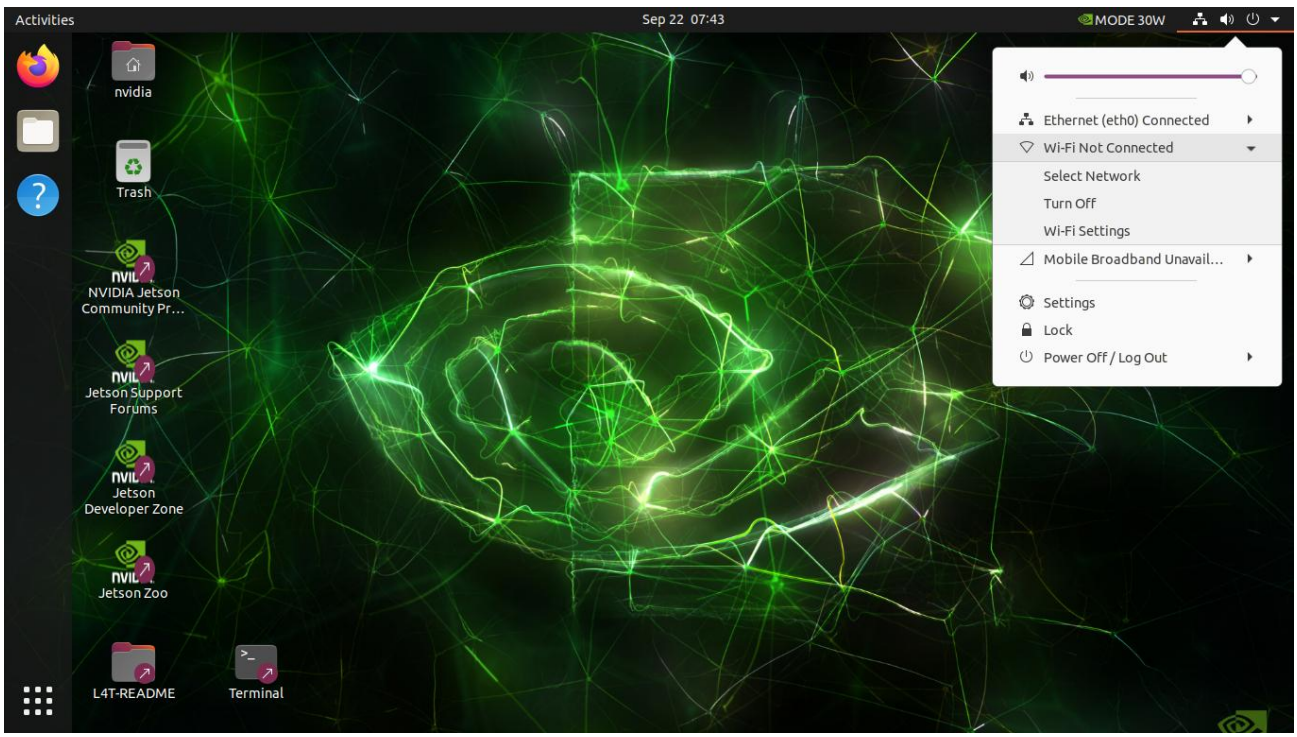
运行命令后 USB-CAN 接收到 can2 发送的数据。

WiFi 连接

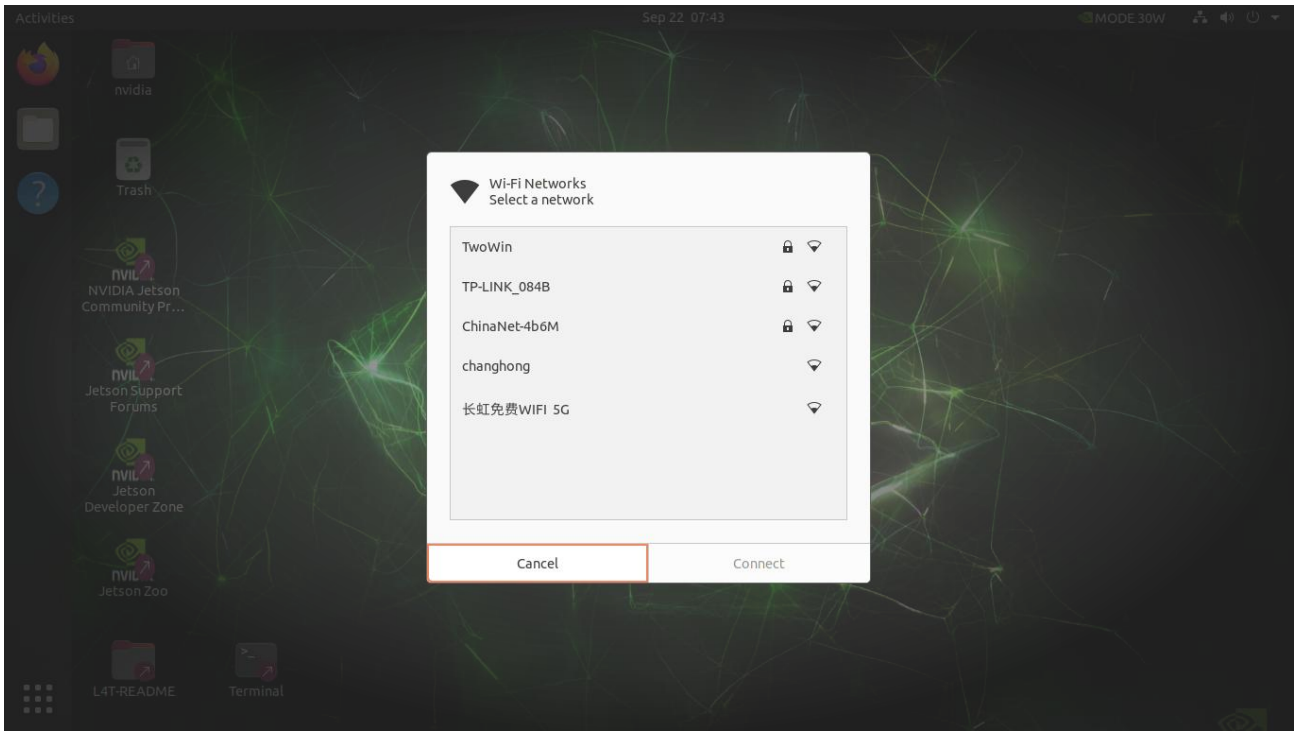
1. 点击箭头所示网络图标, 出现网络设置

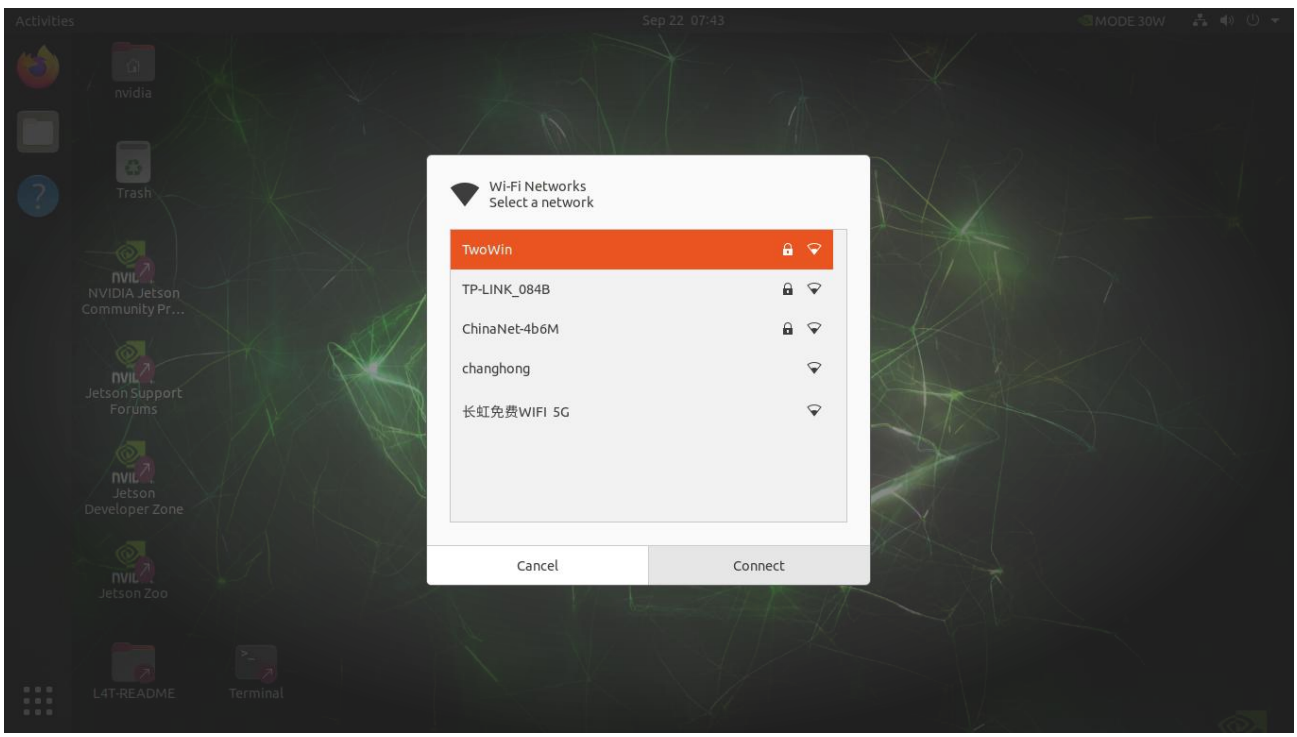


2. 选择第三项, 点击网络选择”Select Network”

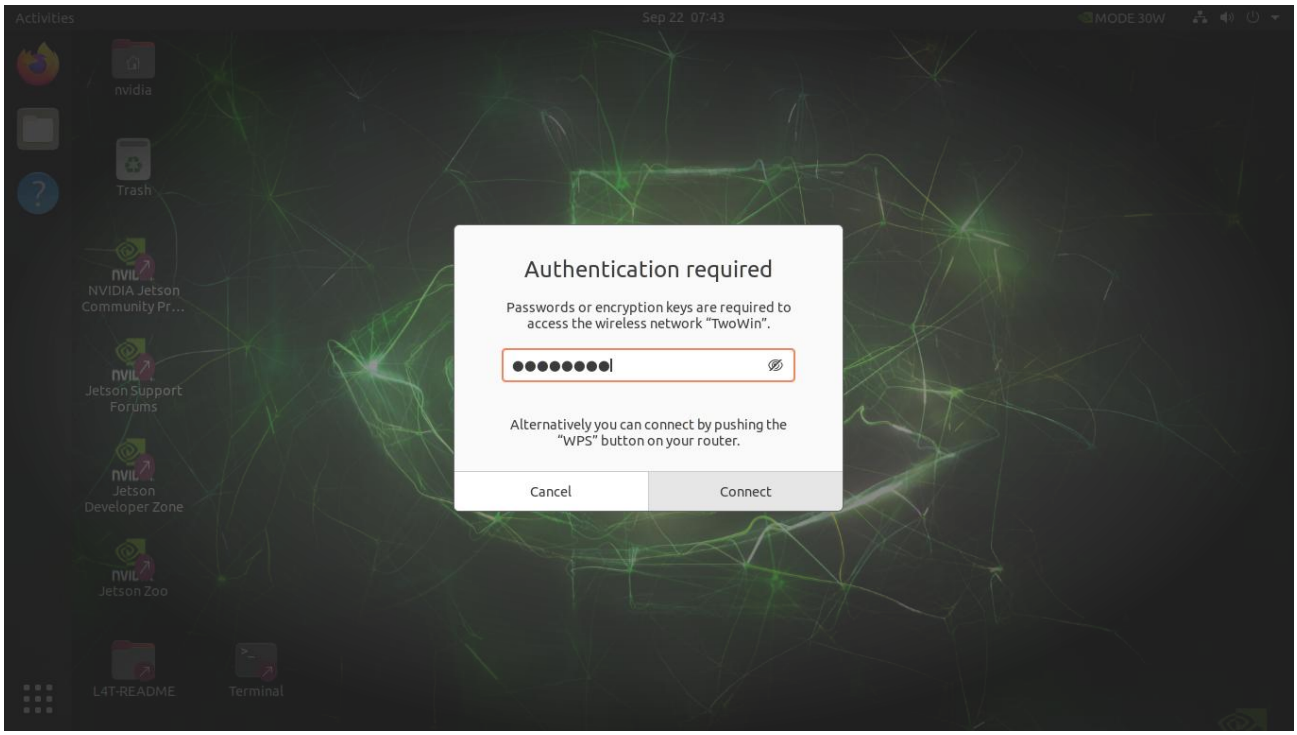


3.选择当前搜寻到的可用网络,找到适合的 wifi,选中后点击”Connect”





4. 输入 wifi 密码



5. 联网后打开网页确认是否网络正常

4G、5G 使用 nmcli 工具拨号联网

注意：

使用前请确认设备已选配安装 4G 或 5G 模块。

sim 卡不支持热插拔，请上电前插好。

1 检查 NetworkManager 服务是否正常运行

sudo systemctl status NetworkManager

```
nvidia@nvidia-desktop:~$ sudo systemctl status NetworkManager
● NetworkManager.service - Network Manager
     Loaded: loaded (/lib/systemd/system/NetworkManager.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2024-01-05 16:35:43 CST; 18min ago
       Docs: man:NetworkManager(8)
   Main PID: 640 (NetworkManager)
      Tasks: 3 (limit: 36322)
     Memory: 20.3M
     CGroup: /system.slice/NetworkManager.service
             └─640 /usr/sbin/NetworkManager --no-daemon

Jan 05 16:52:39 nvidia-desktop pppd[4125]: nm-ppp-plugin: status 5 / phase 'establish'
Jan 05 16:52:39 nvidia-desktop pppd[4125]: nm-ppp-plugin: status 11 / phase 'disconnect'
Jan 05 16:52:39 nvidia-desktop pppd[4125]: Connection terminated.
Jan 05 16:52:39 nvidia-desktop NetworkManager[4125]: Connection terminated.
Jan 05 16:52:40 nvidia-desktop pppd[4125]: nm-ppp-plugin: status 1 / phase 'dead'
Jan 05 16:52:40 nvidia-desktop pppd[4125]: nm-ppp-plugin: cleaning up
Jan 05 16:52:40 nvidia-desktop pppd[4125]: Exit.
Jan 05 16:52:40 nvidia-desktop NetworkManager[640]: <info>  [1704444760.4715] modem["ttyUSB0"]: modem state changed, '
Jan 05 16:52:46 nvidia-desktop NetworkManager[640]: <info>  [1704444766.5764] modem["ttyUSB0"]: modem state changed, '
Jan 05 16:52:46 nvidia-desktop NetworkManager[640]: <info>  [1704444766.5768] device (ttyUSB0): state change: deactivat
```

2 检查 ModemManager 服务是否正常运行

sudo systemctl status ModemManager

```
nvidia@nvidia-desktop:~$ sudo systemctl status ModemManager
● ModemManager.service - Modem Manager
     Loaded: loaded (/lib/systemd/system/ModemManager.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2024-01-05 16:48:06 CST; 5min ago
   Main PID: 3875 (ModemManager)
      Tasks: 3 (limit: 36322)
     Memory: 2.4M
     CGroup: /system.slice/ModemManager.service
             └─3875 /usr/sbin/ModemManager

Jan 05 16:49:43 nvidia-desktop ModemManager[3875]: <info>  [modem0] simple connect state (4/8): wait to get fully enabl
Jan 05 16:49:43 nvidia-desktop ModemManager[3875]: <info>  [modem0] simple connect state (5/8): register
Jan 05 16:49:43 nvidia-desktop ModemManager[3875]: <info>  [modem0] simple connect state (6/8): bearer
Jan 05 16:49:43 nvidia-desktop ModemManager[3875]: <info>  [modem0] simple connect state (7/8): connect
Jan 05 16:49:43 nvidia-desktop ModemManager[3875]: <info>  [modem0] state changed (registered → connecting)
Jan 05 16:49:43 nvidia-desktop ModemManager[3875]: <info>  [modem0] state changed (connecting → connected)
Jan 05 16:49:43 nvidia-desktop ModemManager[3875]: <info>  [modem0] simple connect state (8/8): all done
Jan 05 16:52:40 nvidia-desktop ModemManager[3875]: <info>  [modem0] state changed (connected → disconnecting)
Jan 05 16:52:46 nvidia-desktop ModemManager[3875]: <info>  [modem0] state changed (disconnecting → registered)
Jan 05 16:52:46 nvidia-desktop ModemManager[3875]: <info>  [modem0/bearer0] connection #2 finished: duration 183s, tx:
```

3 检查系统是否正常识别 4G 模块为 modem

mmcli -L

nmcli d

```
nvidia@nvidia-desktop:~$ mmcli -L
    /org/freedesktop/ModemManager1/Modem/0 [Quectel] EC20F
nvidia@nvidia-desktop:~$ nmcli d
DEVICE    TYPE      STATE         CONNECTION
l4tbr0    bridge    connected     l4tbr0
docker0   bridge    connected     docker0
ttyUSB2   gsm       disconnected  --
eth0      ethernet  unavailable   --
eth1      ethernet  unavailable   --
eth2      ethernet  unavailable   --
eth3      ethernet  unavailable   --
wlan0     wifi      unavailable   --
dummy0    dummy     unmanaged     --
rndis0    ethernet  unmanaged     --
usb0      ethernet  unmanaged     --
lo        loopback  unmanaged     --
nvidia@nvidia-desktop:~$
```

4 使用 nmcli 管理拨号

4.1 查看已有连接，将已有的 TYPE 类型为 gsm 的连接配置删除。如果不存在，跳过该操作。

nmcli connection

sudo nmcli connection delete uuid xxxxx（根据实际的 ID 号填写）

```
nvidia@nvidia-desktop:~$ nmcli connection
NAME                  UUID                                  TYPE      DEVICE
l4tbr0                6a8f933c-73bb-42fa-9794-f5d611110faf  bridge    l4tbr0
docker0               9d20eaac-a33c-420f-8ead-b66c47ed81d3  bridge    docker0
China Mobile Internet 60246731-703a-49ca-b80e-355ed602a237  gsm       --
Wired connection 1    d34ebb2e-8acb-3470-a3c1-cad4d96aa268  ethernet  --
Wired connection 2    b4cce976-8a88-3d15-aa55-b2076a71d671  ethernet  --
Wired connection 3    3e8d52cf-6e7d-3d27-a9ba-8ec330fe03d8  ethernet  --
netplan-eth0          626dd384-8b3d-3690-9511-192b2c79b3fd  ethernet  --
netplan-wlan0-图为2404 ffb0da53-9a6f-3fd3-8e7d-d153624e1f6d  wifi      --
图为2404              22db099b-8736-4527-ba75-c067465ce787  wifi      --
nvidia@nvidia-desktop:~$ sudo nmcli connection delete uuid 60246731-703a-49ca-b80e-355ed602a237
Connection 'China Mobile Internet' (60246731-703a-49ca-b80e-355ed602a237) successfully deleted.
nvidia@nvidia-desktop:~$ nmcli connection
NAME                  UUID                                  TYPE      DEVICE
l4tbr0                6a8f933c-73bb-42fa-9794-f5d611110faf  bridge    l4tbr0
docker0               9d20eaac-a33c-420f-8ead-b66c47ed81d3  bridge    docker0
Wired connection 1    d34ebb2e-8acb-3470-a3c1-cad4d96aa268  ethernet  --
Wired connection 2    b4cce976-8a88-3d15-aa55-b2076a71d671  ethernet  --
Wired connection 3    3e8d52cf-6e7d-3d27-a9ba-8ec330fe03d8  ethernet  --
netplan-eth0          626dd384-8b3d-3690-9511-192b2c79b3fd  ethernet  --
netplan-wlan0-图为2404 ffb0da53-9a6f-3fd3-8e7d-d153624e1f6d  wifi      --
图为2404              22db099b-8736-4527-ba75-c067465ce787  wifi      --
nvidia@nvidia-desktop:~$
```

4.2 查看并打开移动宽带开关。

nmcli radio

sudo nmcli radio wwan on

```
nvidia@nvidia-desktop:~$ nmcli radio
WIFI-HW  WIFI      WWAN-HW  WWAN
enabled  disabled  enabled  disabled
nvidia@nvidia-desktop:~$ sudo nmcli radio wwan on
nvidia@nvidia-desktop:~$ nmcli radio
WIFI-HW  WIFI      WWAN-HW  WWAN
enabled  disabled  enabled  enabled
nvidia@nvidia-desktop:~$
```

4.3 创建 4G 拨号连接

sudo nmcli connection add type gsm ifname '*' con-name twgsm

部分 sim 卡需要加 apn 参数方能创建成功，在最后加上参数 gsm.apn <apn_name>

其中 联通:3gnet 移动：cmnet 电信：ctnet

sudo nmcli connection add type gsm ifname '*' con-name twgsm gsm.apn 3gnet

```
nvidia@nvidia-desktop:~$ sudo nmcli connection add type gsm ifname '*' con-name twgsm
Connection 'twgsm' (ac738f0c-59c8-4c8d-8945-95173781a696) successfully added.
nvidia@nvidia-desktop:~$ nmcli connection
NAME                UUID                                  TYPE      DEVICE
twgsm               ac738f0c-59c8-4c8d-8945-95173781a696  gsm       ttyUSB2
l4tbr0              6a8f933c-73bb-42fa-9794-f5d611110faf  bridge    l4tbr0
docker0             9d20eaac-a33c-420f-8ead-b66c47ed81d3  bridge    docker0
Wired connection 1  d34ebb2e-8acb-3470-a3c1-cad4d96aa268  ethernet  --
Wired connection 2  b4cce976-8a88-3d15-aa55-b2076a71d671  ethernet  --
Wired connection 3  3e8d52cf-6e7d-3d27-a9ba-8ec330fe03d8  ethernet  --
netplan-eth0        626dd384-8b3d-3690-9511-192b2c79b3fd  ethernet  --
netplan-wlan0-图为2404 ffb0da53-9a6f-3fd3-8e7d-d153624e1f6d  wifi      --
图为2404            22db099b-8736-4527-ba75-c067465ce787  wifi      --
```

4.4 设置连接属性

```
nvidia@nvidia-desktop:~$ sudo nmcli connection modify twgsm connection.autoconnect yes
nvidia@nvidia-desktop:~$ sudo nmcli connection modify twgsm connection.autoconnect-retries  0
nvidia@nvidia-desktop:~$ sudo nmcli connection modify twgsm connection.autoconnect-priority 0
nvidia@nvidia-desktop:~$ sudo nmcli connection modify twgsm ipv6.method disable
nvidia@nvidia-desktop:~$ sudo nmcli connection modify twgsm connection.metered no
```

1. 开启自动连接

   sudo nmcli connection modify twgsm connection.autoconnect yes

2. 设置重试次数（0 为无限次）

   sudo nmcli connection modify twgsm connection.autoconnect-retries 0

3. 设置连接优先级

   sudo nmcli connection modify twgsm connection.autoconnect-priority 0

4. 关闭 IPv6 (使用 IPv6 就关闭 IPv4)

   sudo nmcli connection modify twgsm ipv6.method ignore

5. 关闭按流量计费

   sudo nmcli connection modify twgsm connection.metered no

4.5 查看拨号状态

```
nvidia@nvidia-desktop:~$ ifconfig ppp0
上一个 flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.19.48.238  netmask 255.255.255.255  destination 0.0.0.0
        ppp  txqueuelen 3  (Point-to-Point Protocol)
        RX packets 11  bytes 1031 (1.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 509 (509.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

nvidia@nvidia-desktop:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         0.0.0.0         0.0.0.0         U     700    0        0 ppp0
0.0.0.0         192.168.55.100  0.0.0.0         UG    32766  0        0 l4tbr0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 l4tbr0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
192.168.55.0    0.0.0.0         255.255.255.0   U     0      0        0 l4tbr0
nvidia@nvidia-desktop:~$ nmcli connection
NAME                UUID                                  TYPE      DEVICE
twgsm               7f060dc2-d405-47eb-bbe1-129d12fbf132  gsm       ttyUSB2
l4tbr0              5ce18233-2a87-4670-a59e-79d97ea1b2cb  bridge    l4tbr0
docker0             8286c4c1-7408-4660-838d-159414eaa235  bridge    docker0
Wired connection 1  d34ebb2e-8acb-3470-a3c1-cad4d96aa268  ethernet  --
Wired connection 2  b4cce976-8a88-3d15-aa55-b2076a71d671  ethernet  --
Wired connection 3  3e8d52cf-6e7d-3d27-a9ba-8ec330fe03d8  ethernet  --
netplan-eth0        626dd384-8b3d-3690-9511-192b2c79b3fd  ethernet  --
netplan-wlan0-图为2404 ffb0da53-9a6f-3fd3-8e7d-d153624e1f6d  wifi      --
图为2404            22db099b-8736-4527-ba75-c067465ce787  wifi      --
nvidia@nvidia-desktop:~$
```

其中 nmcli connection 上的信息颜色：

绿色：已连接

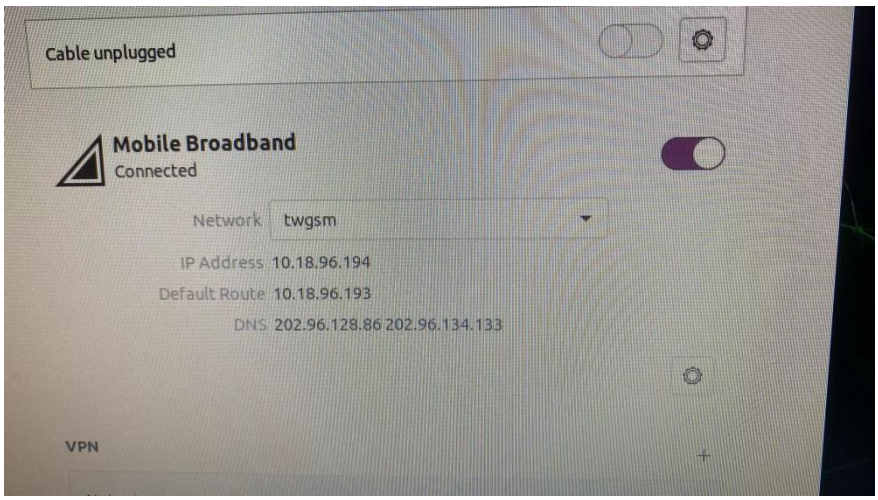
黄色：正在连接

白色：未连接

5 注意事项

5.1 部分 sim 卡（如物联网卡类）拨号连接失败，需向卡商确认卡的转态。（物联网卡中会有区域限制、锁卡限制、专项网络限制等等）

5.2 成功拨号后 Ubuntu20.04 的此 gui 界面会显示 IP 等，这里不要点关闭，否则需要重新创建添加 gsm 链接



5.3 如果 mmcli -L 找不到设备

创建文本：

sudo vim /lib/udev/rules.d/78-mm-allowlist-internal-modem.rules

写入以下信息：

ACTION!="add|change|move", GOTO="mm_allowlist_internal_modem_end"

ATTRS{idVendor}=="2c7c", ATTRS{idProduct}=="0125", ENV{ID_MM_DEVICE_PROCESS}="1"

ATTRS{idVendor}=="2949", ATTRS{idProduct}=="8247", ENV{ID_MM_DEVICE_PROCESS}="1"

ATTRS{idVendor}=="2c7c", ATTRS{idProduct}=="0800", ENV{ID_MM_DEVICE_PROCESS}="1"

LABEL="mm_allowlist_internal_modem_end"

sudo udevadm control --reload

sudo udevadm trigger

sudo systemctl restart NetworkManager

采用 M.2 Key m SSD 为系统盘

SSD 作用

NVMe SSD 硬盘仅作为系统盘（rootfs 和用户区），系统的启动引导依然是通过 SD 卡或 EMMC，比如升级设备树 dtb 还是在 SD 卡或 EMMC 中。

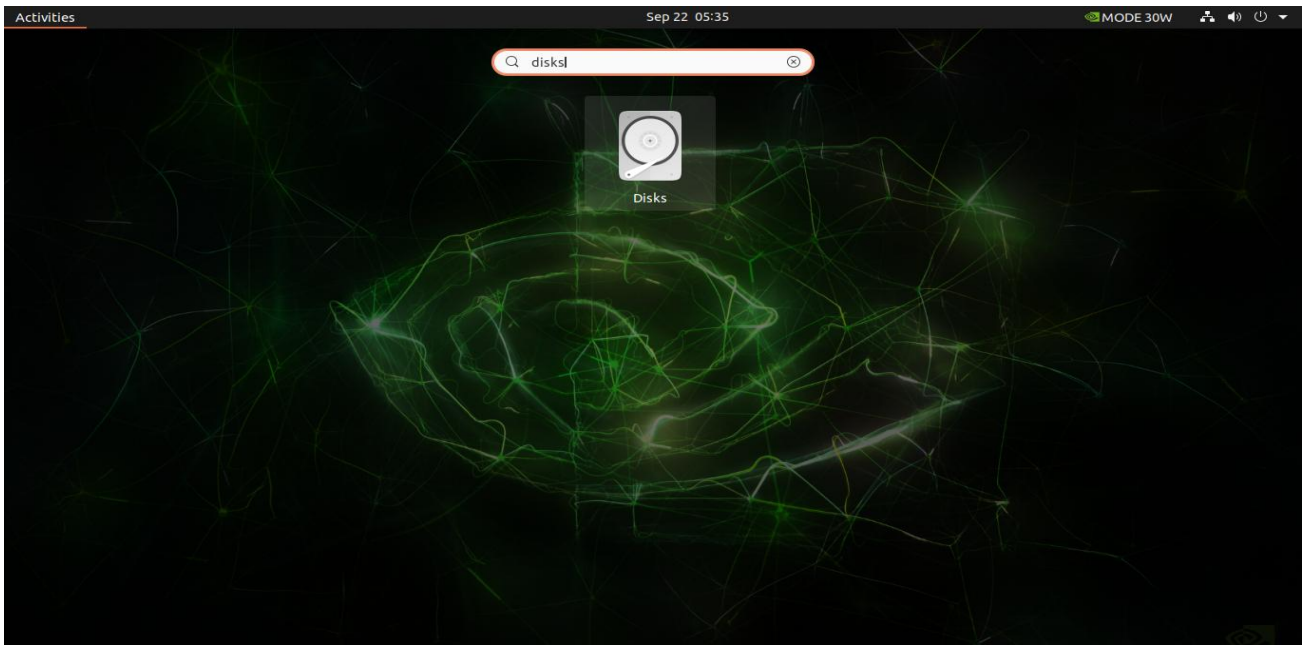
步骤一、准备 SSD 并格式化为 GPT

1.准备 M.2 Key M SSD

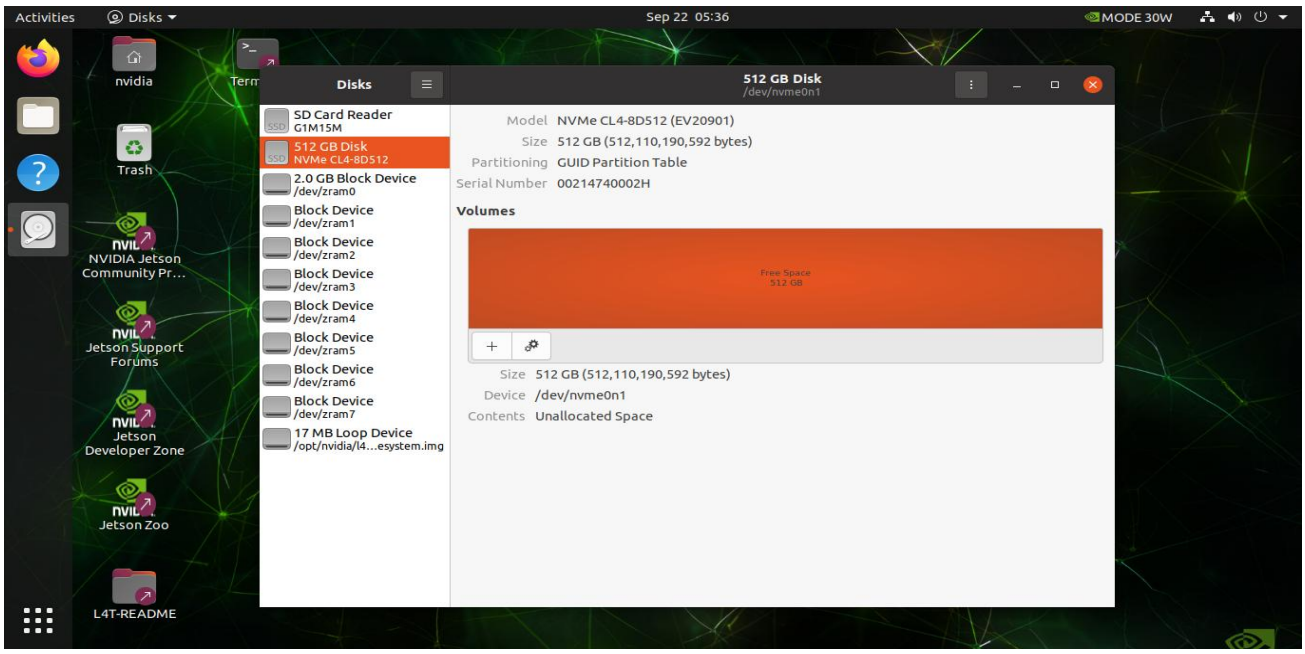
2.打开 Ubuntu20.04 自带 Disks 工具()，找到安装的 ssd 硬盘,首先按键”Ctrl+F”将其快速格式化为.

3.具体操作参考下图所示(需严格按照下面方式进行相关操作,避免造成失误导致设备无法进入桌面系统):

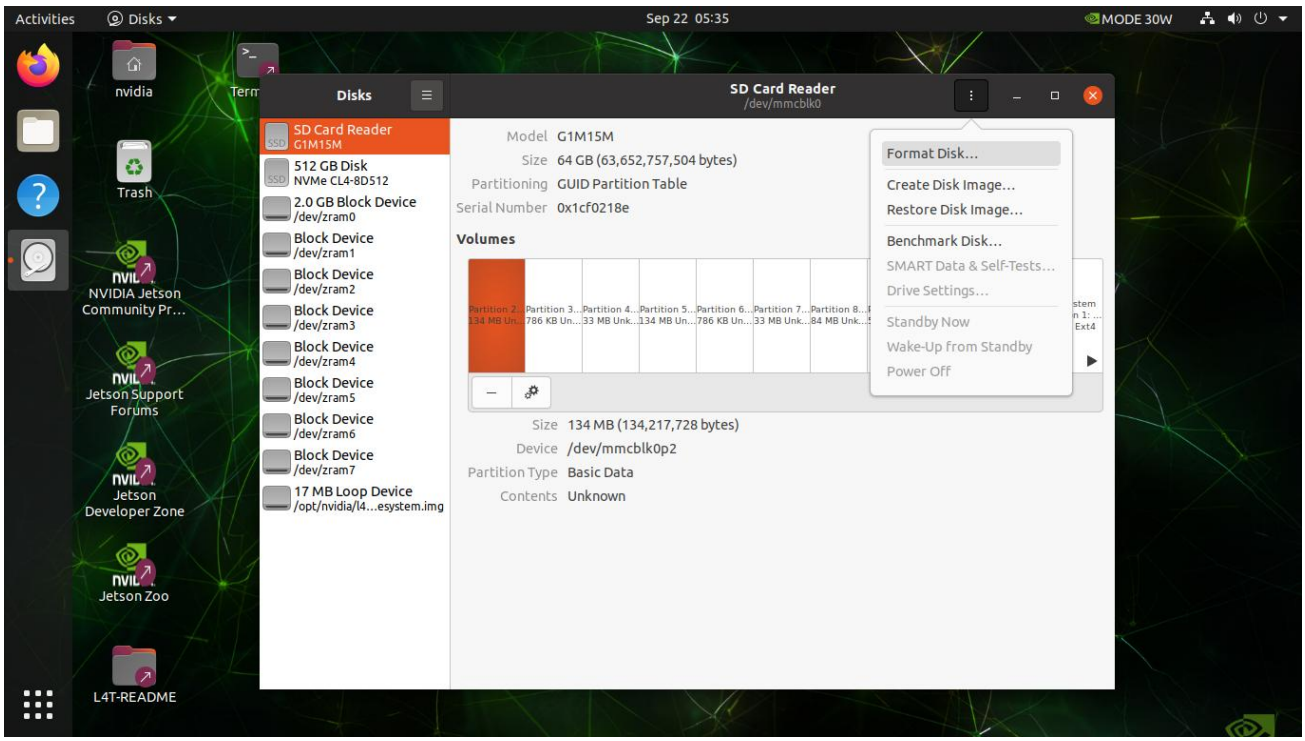
1)首先打开方式为按 win 键出现搜索框,搜索”disks”工具;

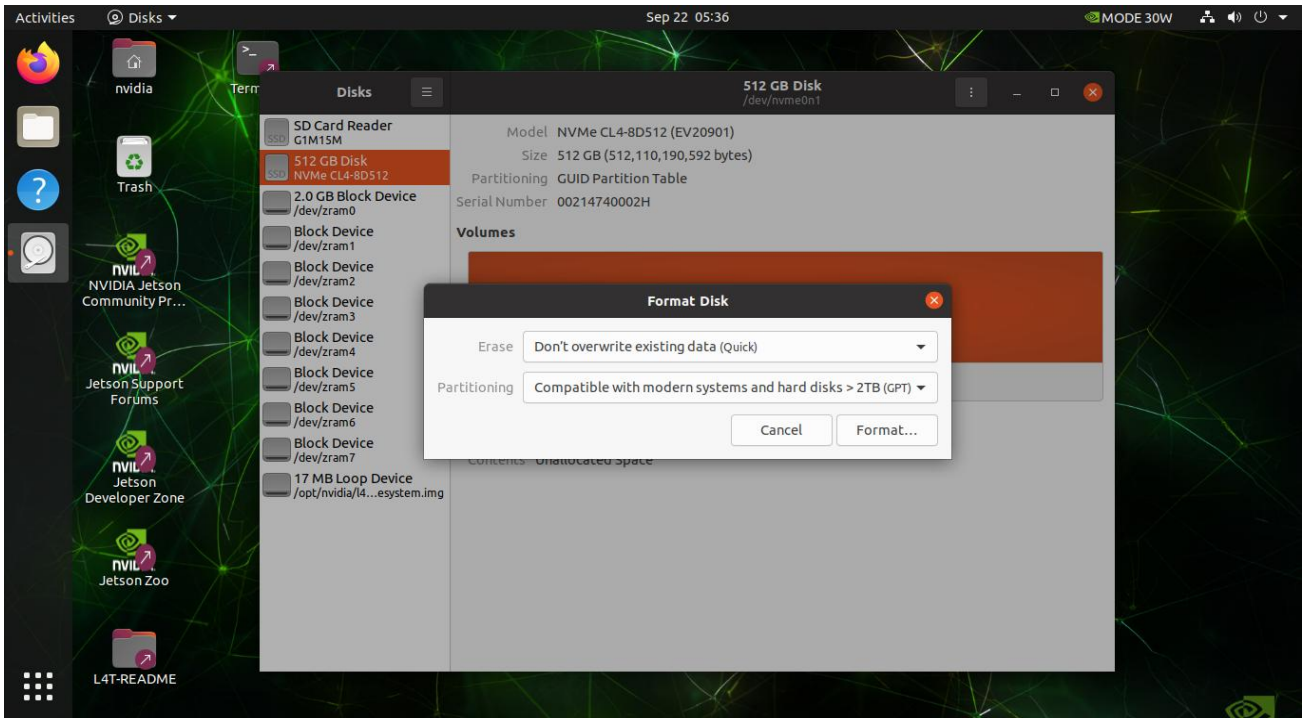


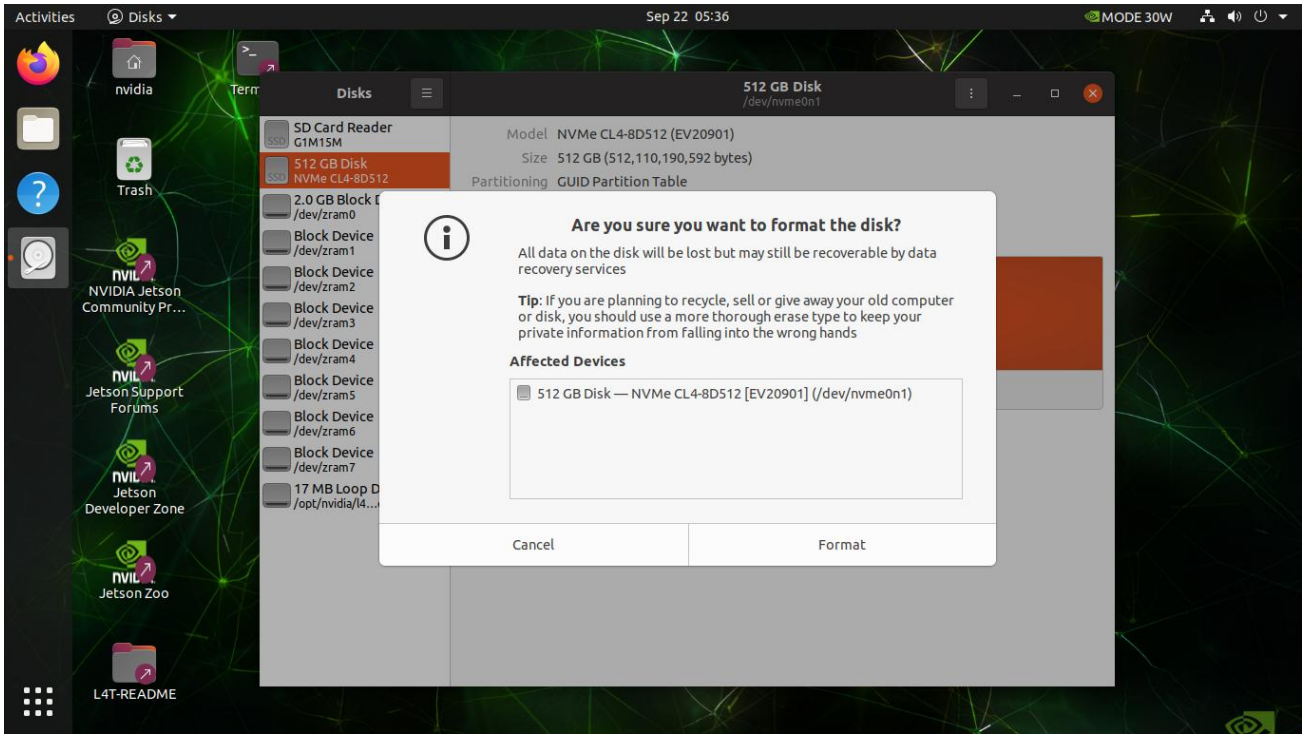
2) 进入 disks,找到安装的 ssd;



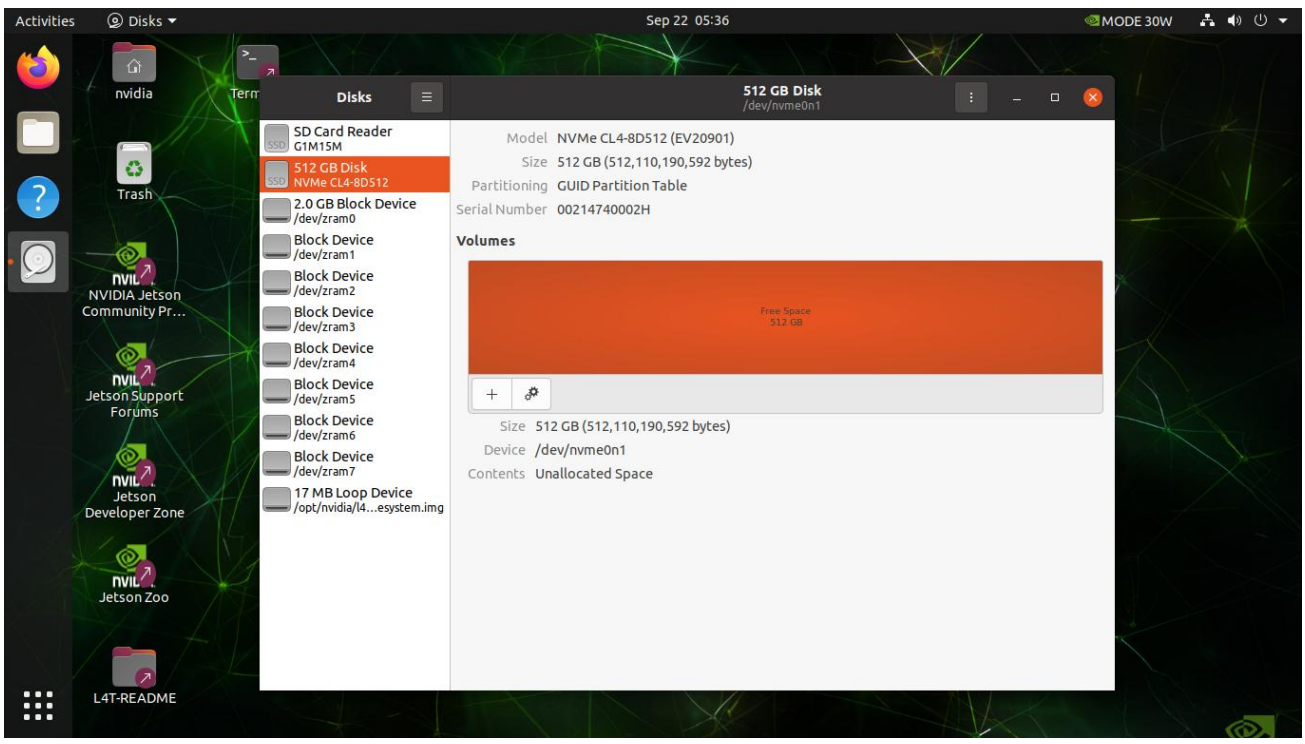
3)点击右上角”:”继续点击第一个选项”Format Disk”,然后出现弹窗点击”Format”



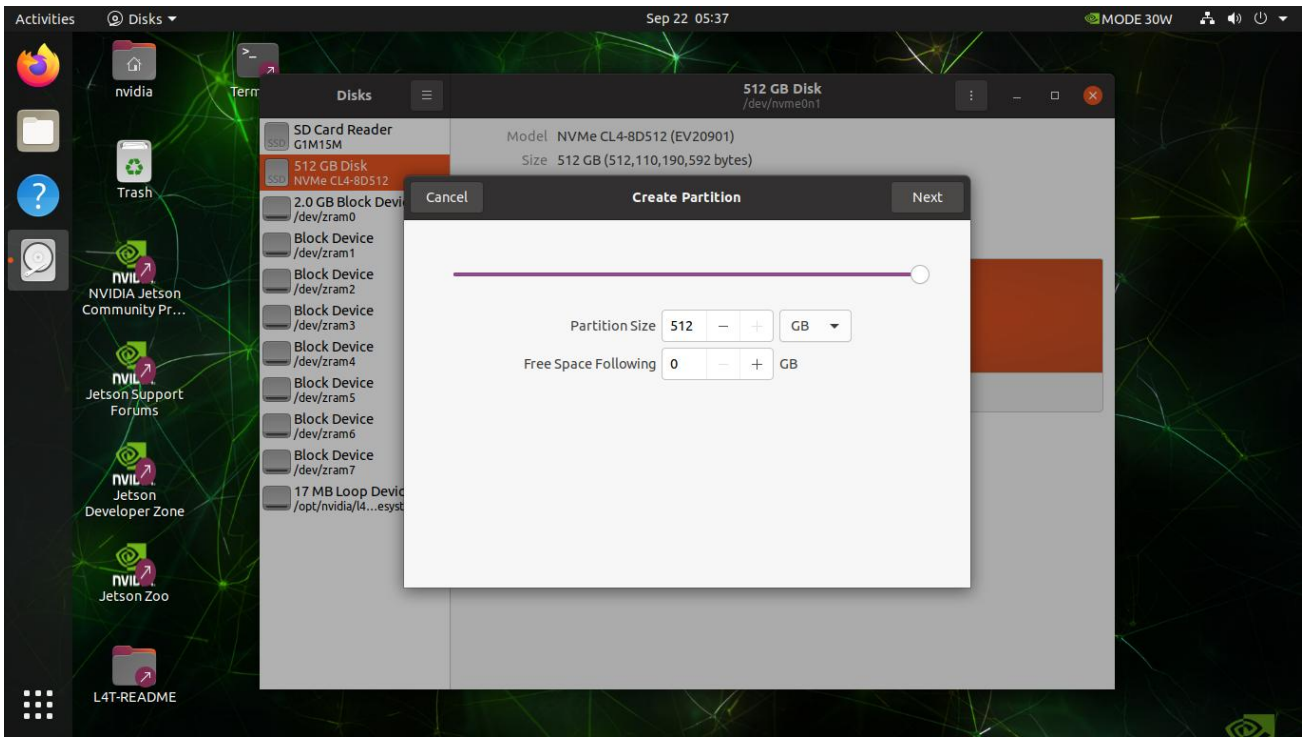




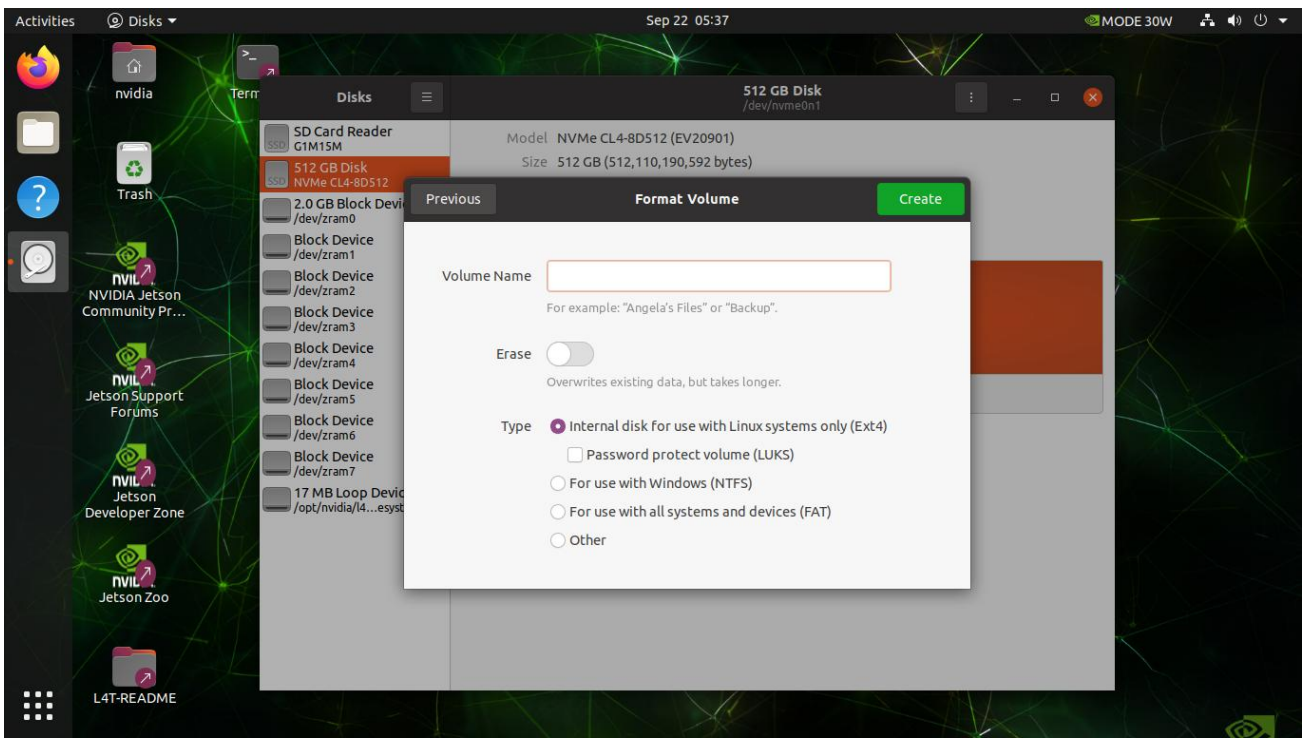
3） 格式化完毕后，点击”+”新建分区，



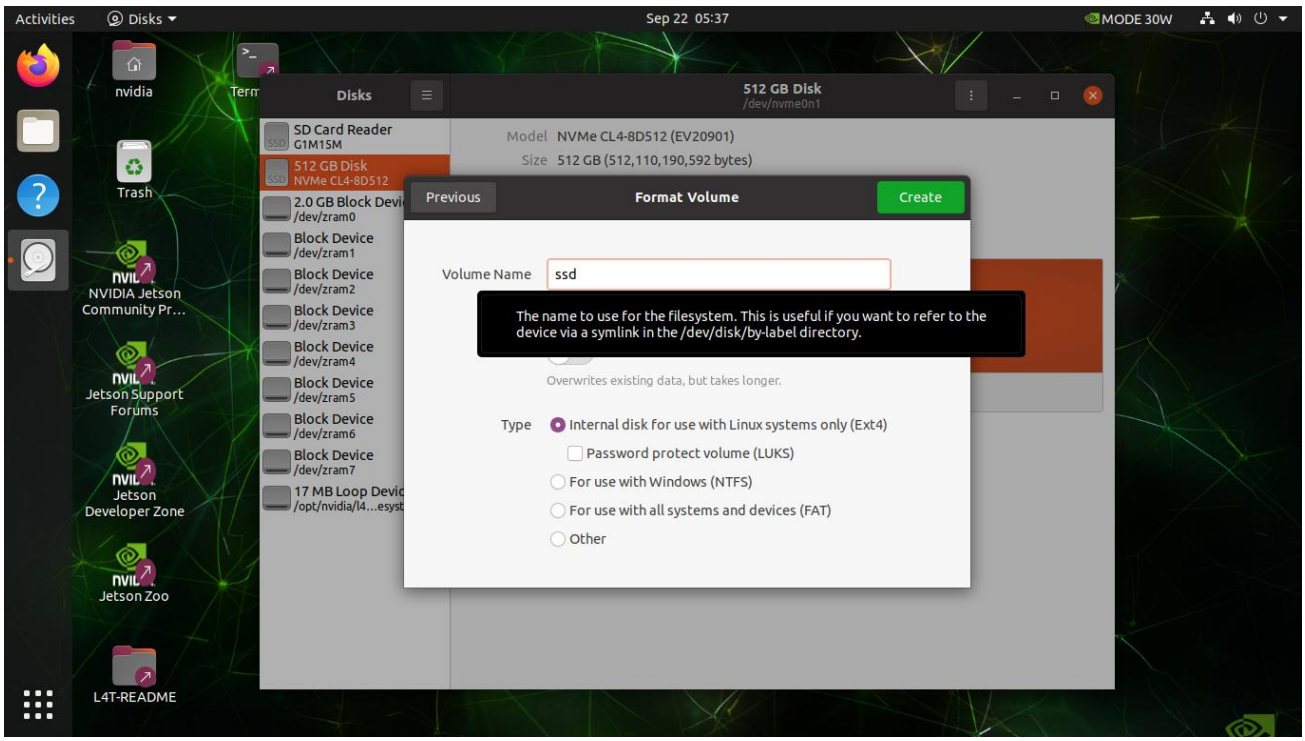
4)划分新建分区大小,默认全部分配,这个建议默认就好,如果需要更改,就根据自身需求进行合理分配,然后点击”Next”



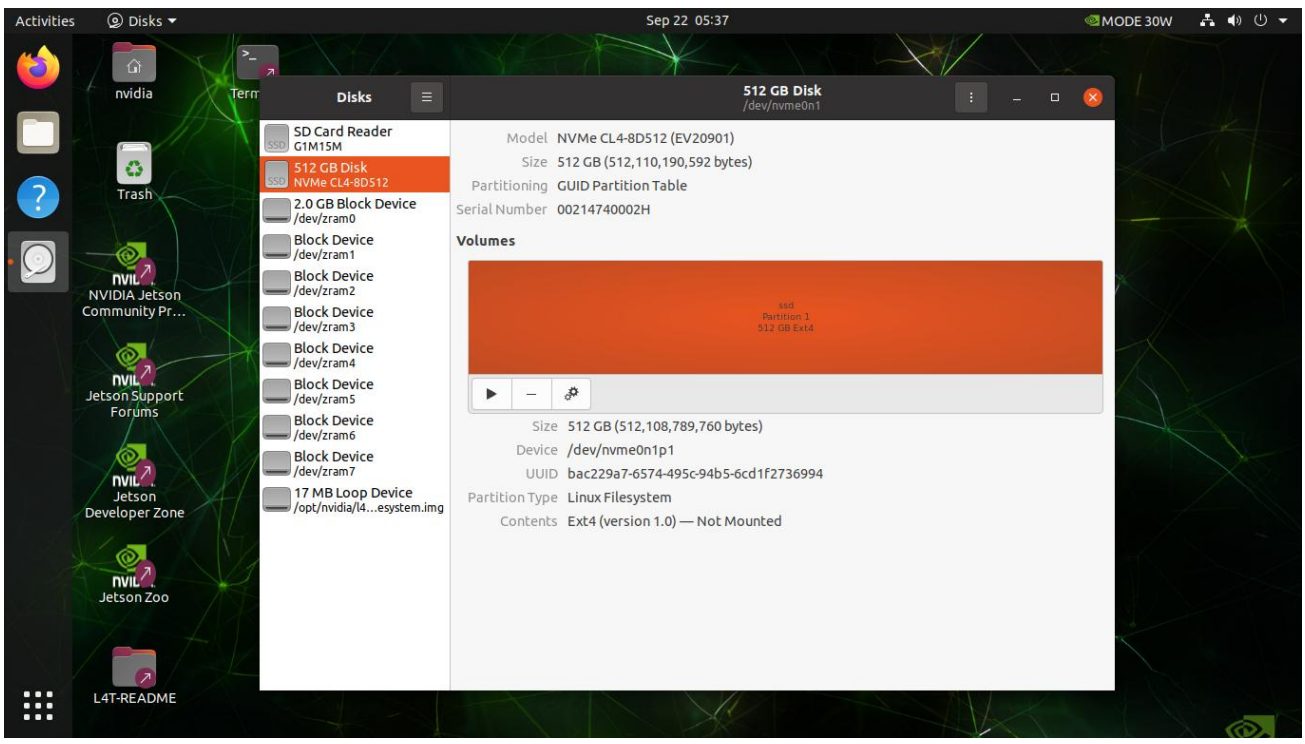
5)选择文件格式,默认为”ext4”,这个无需更改;



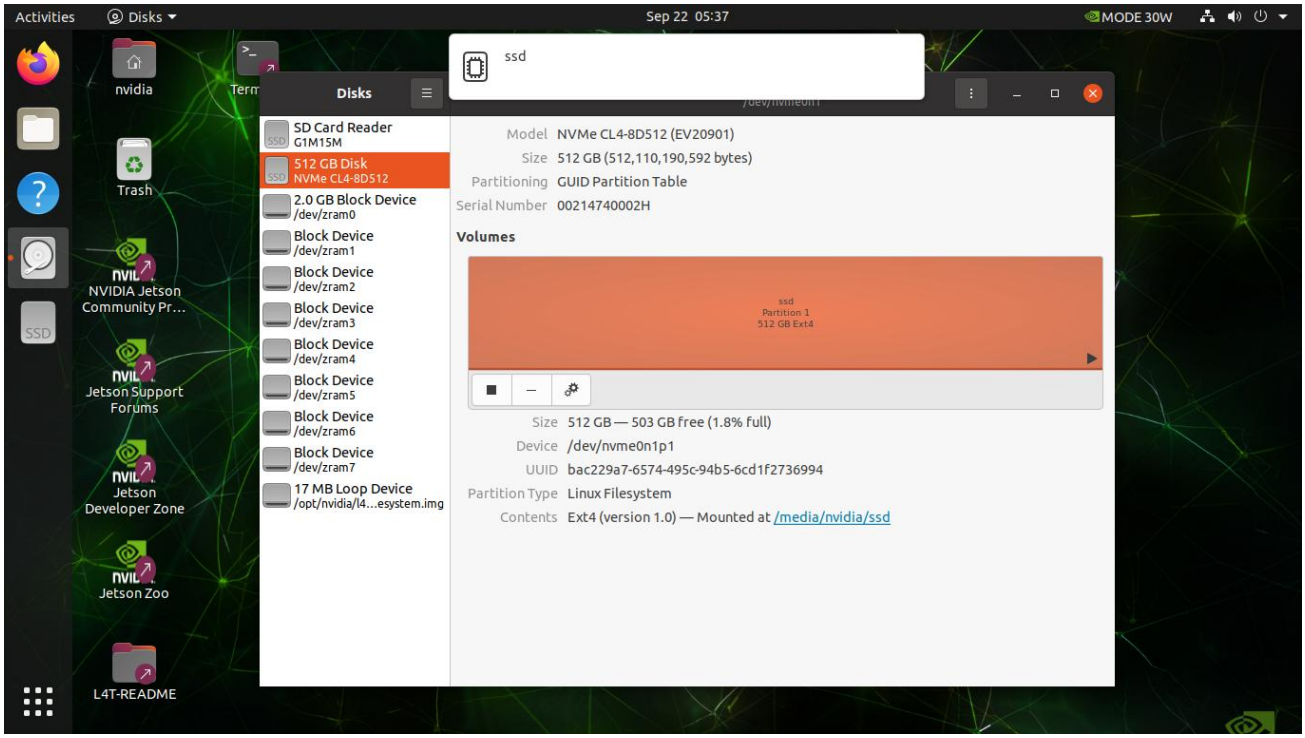
6)给新分区命名,这个按照自身意愿命名即可;



7)创建分区完毕后出现下图所示画面,点击三角符号挂载

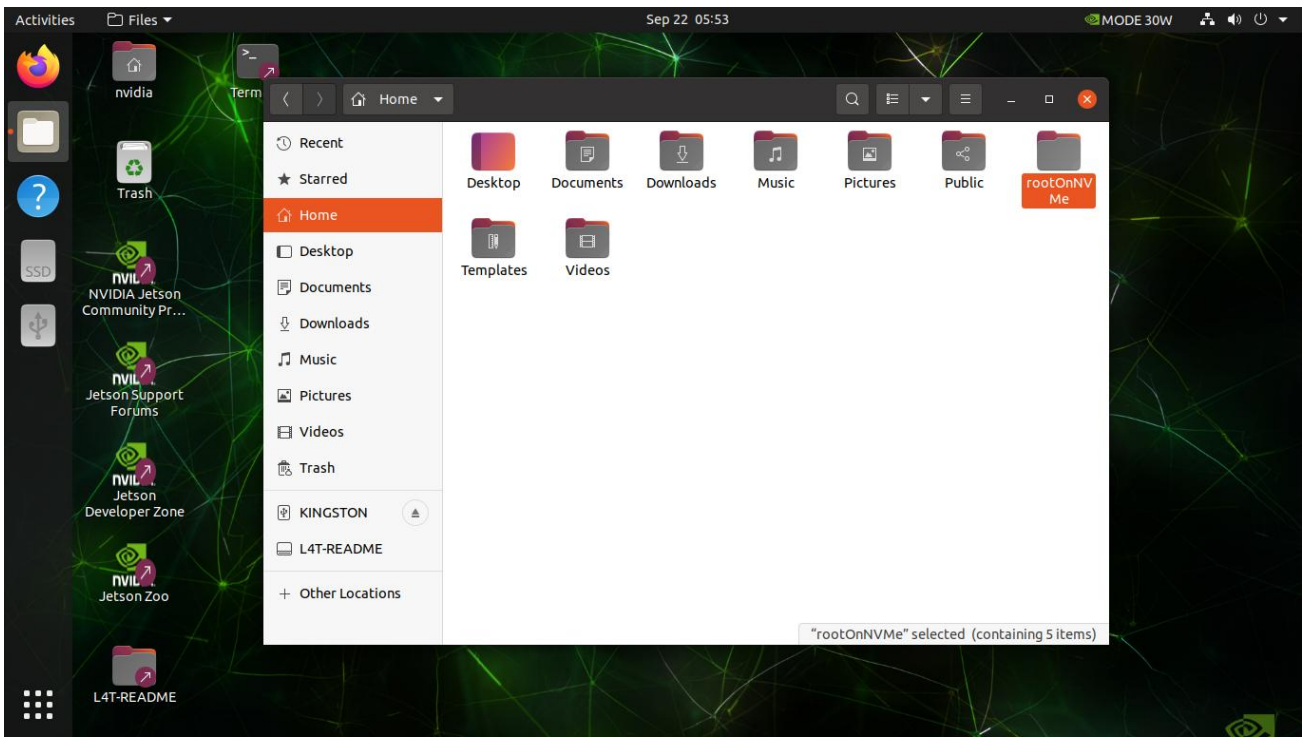


8)如下图显示则为挂载成功

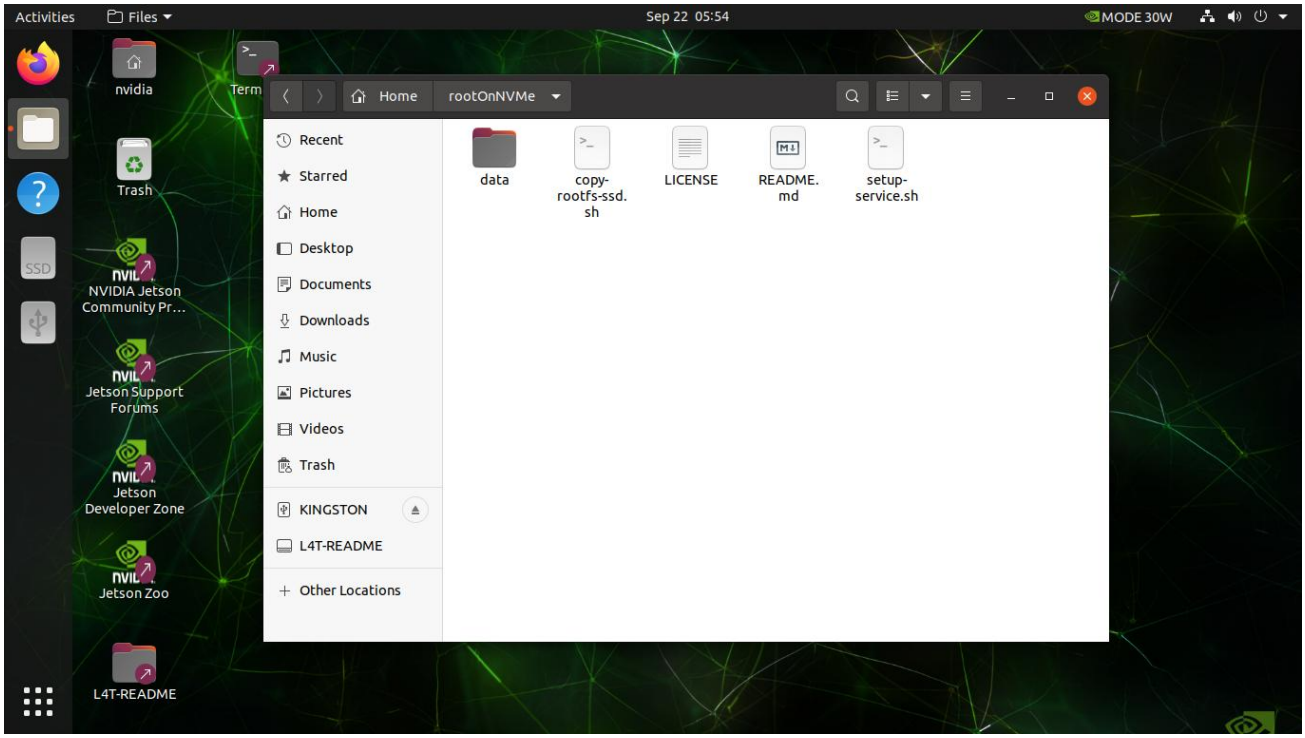


步骤二:(系统盘转换)

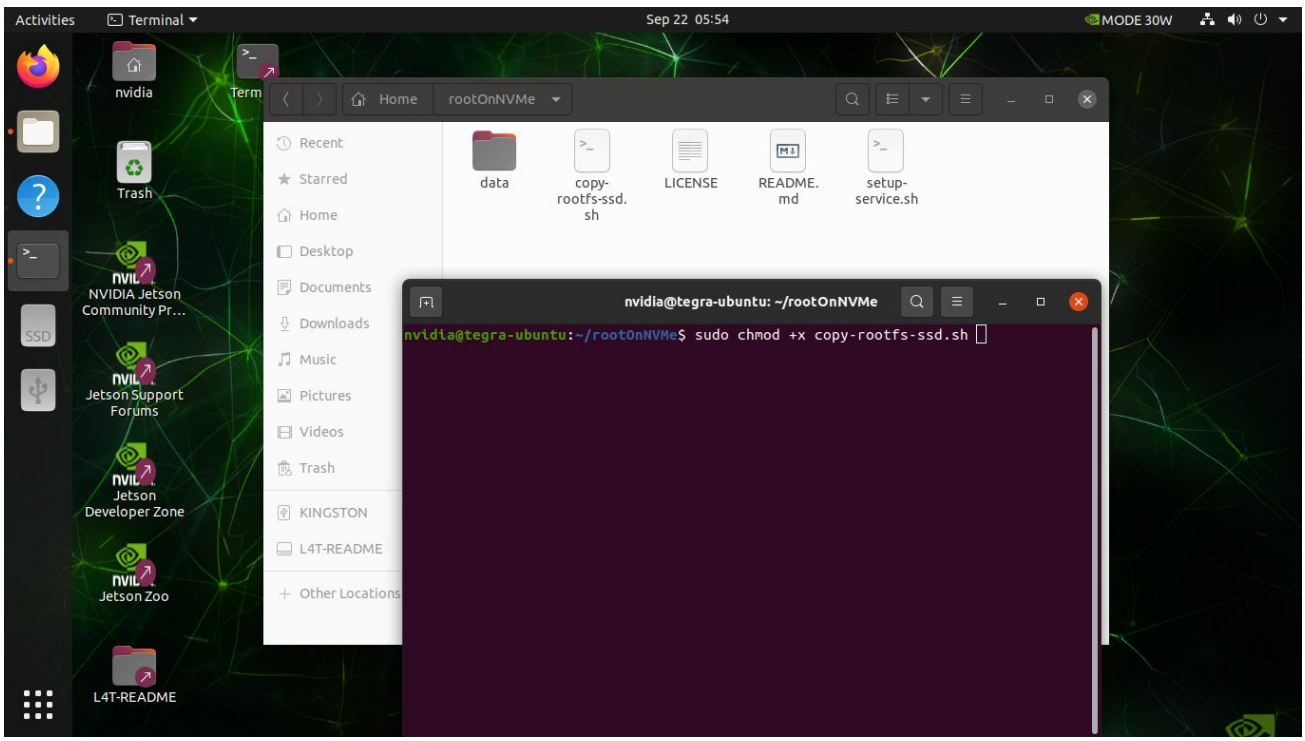
1)进入/home 目录,找到 rootOnNVMe 文件



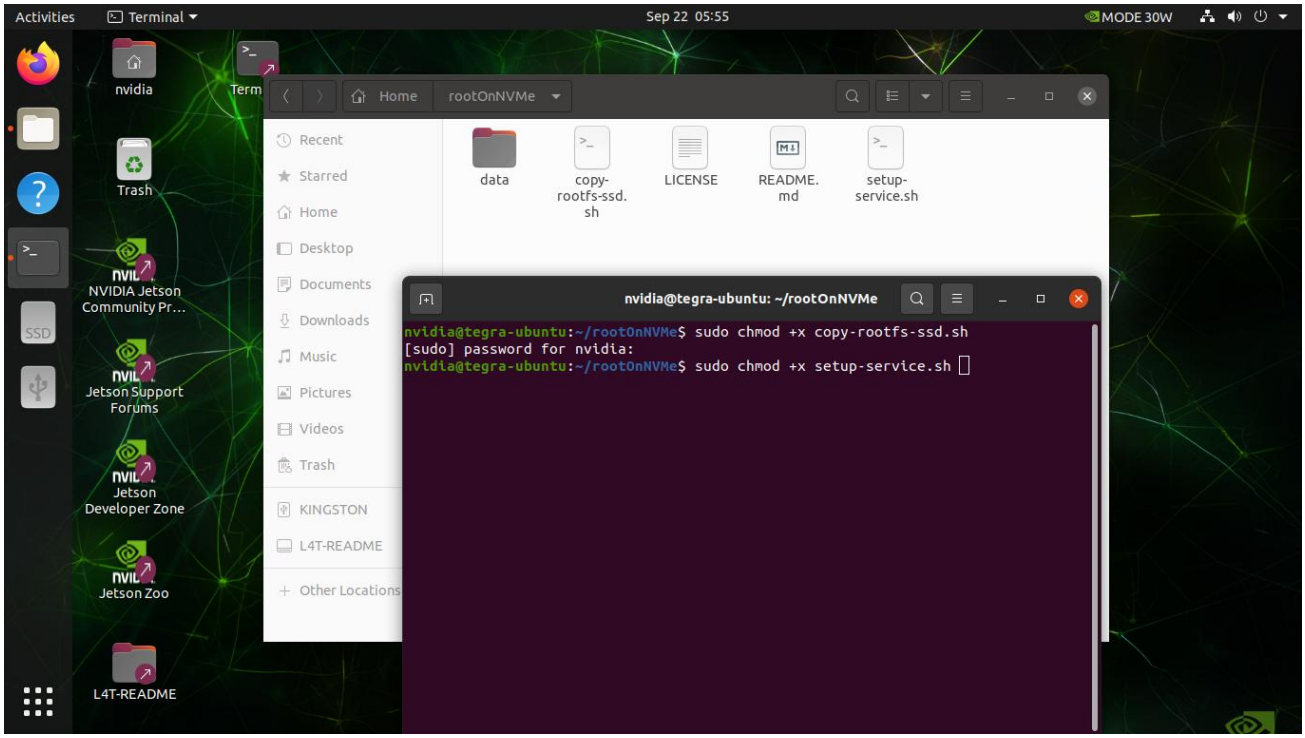
2)进入该文件夹;



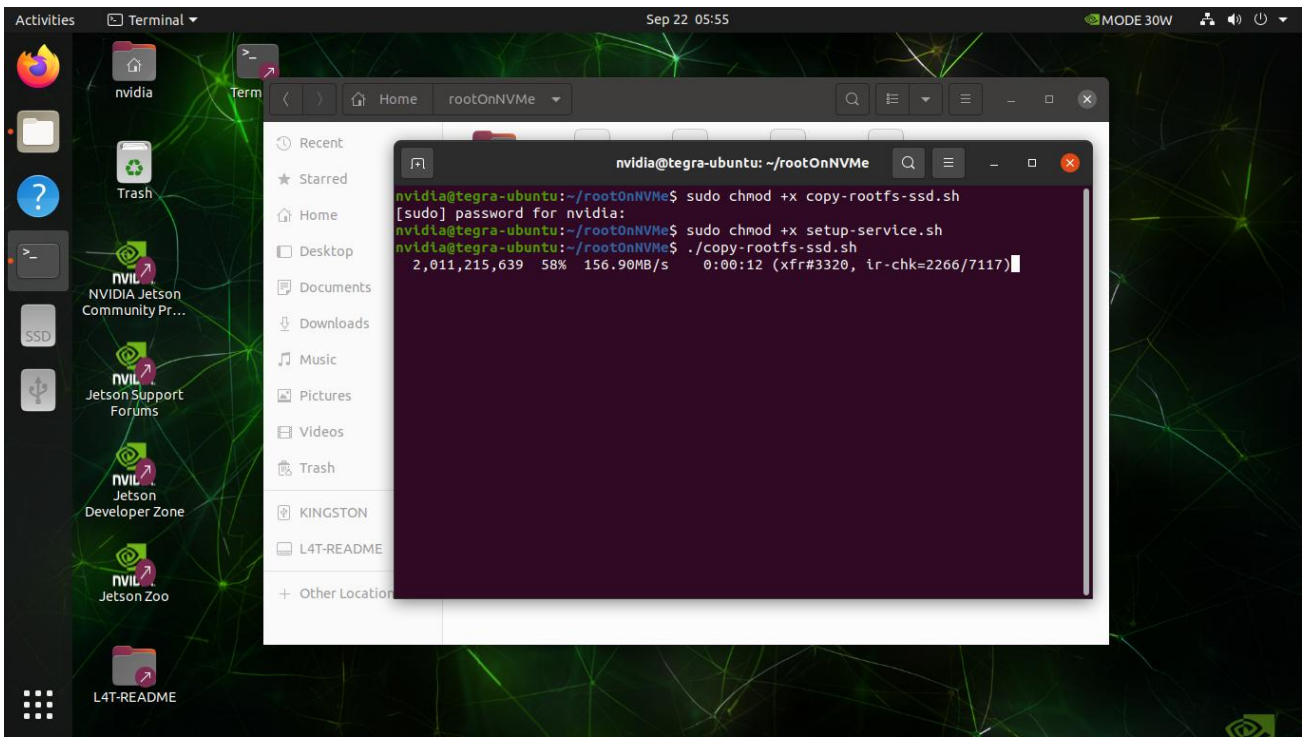
3)在文件夹空白处右键选择”Open in Terminal”,执行命令:”sudo chmod +x copy-rootfs-ssd.sh”

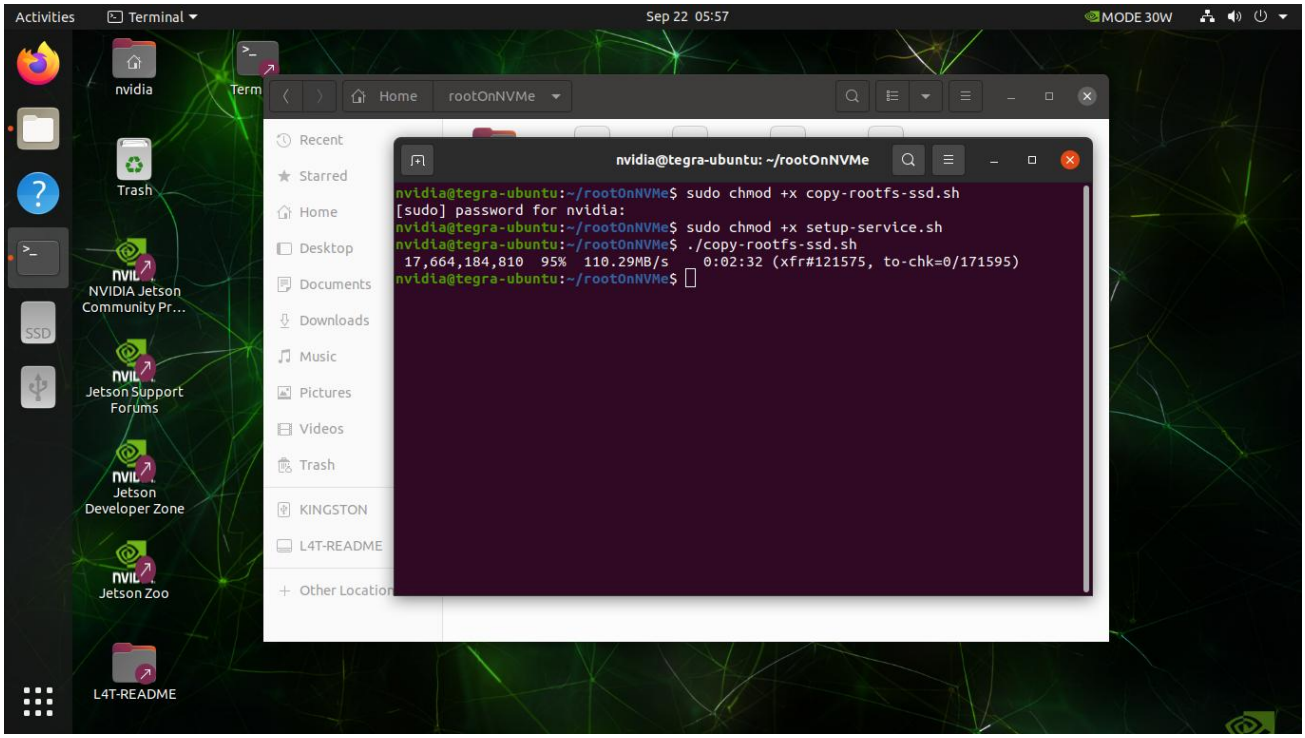


4)执行命令:”sudo chmod +x setup-service.sh”

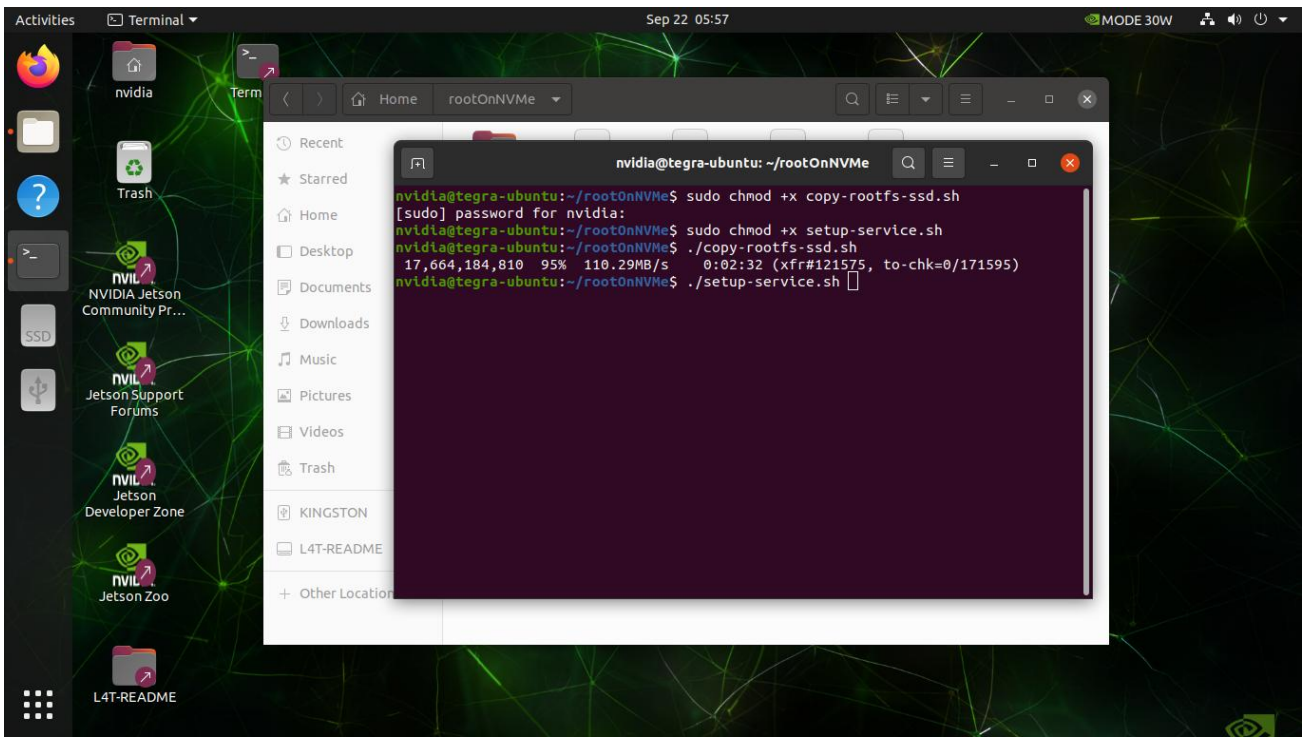


5)执行脚本:”copy-rootfs-ssd.sh”

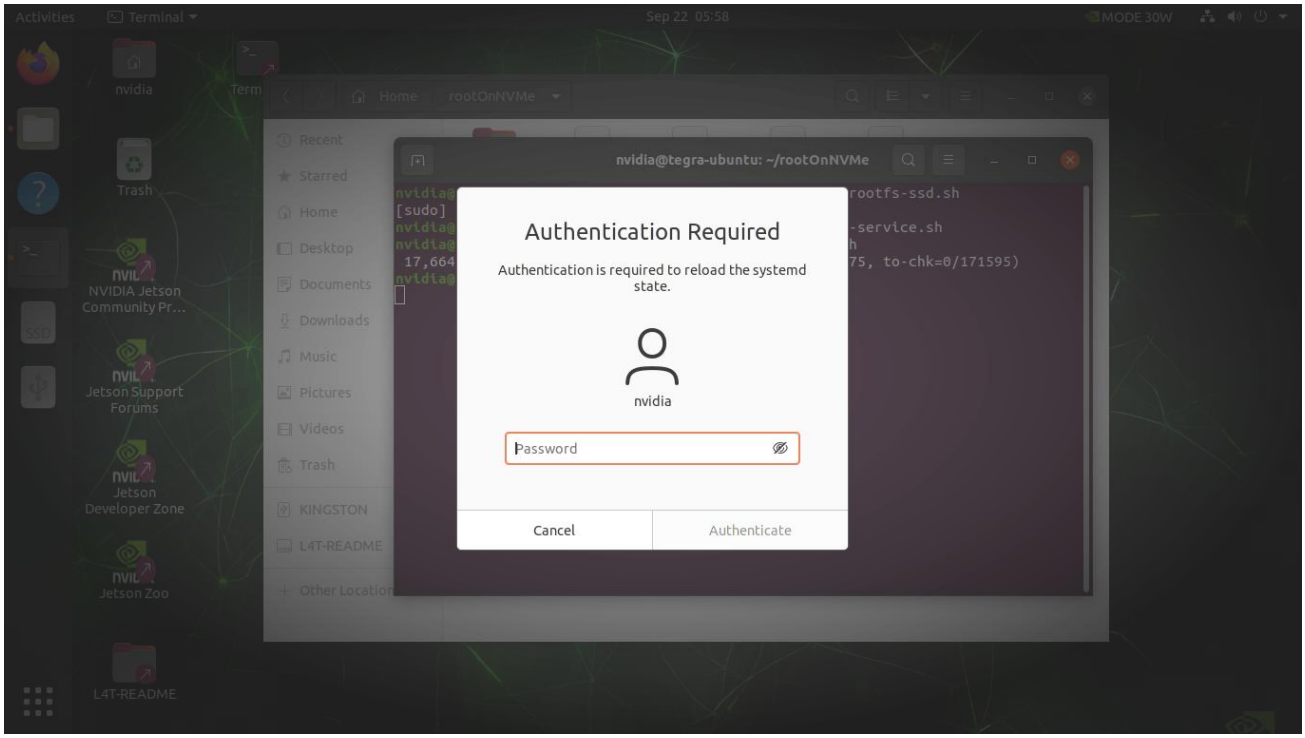


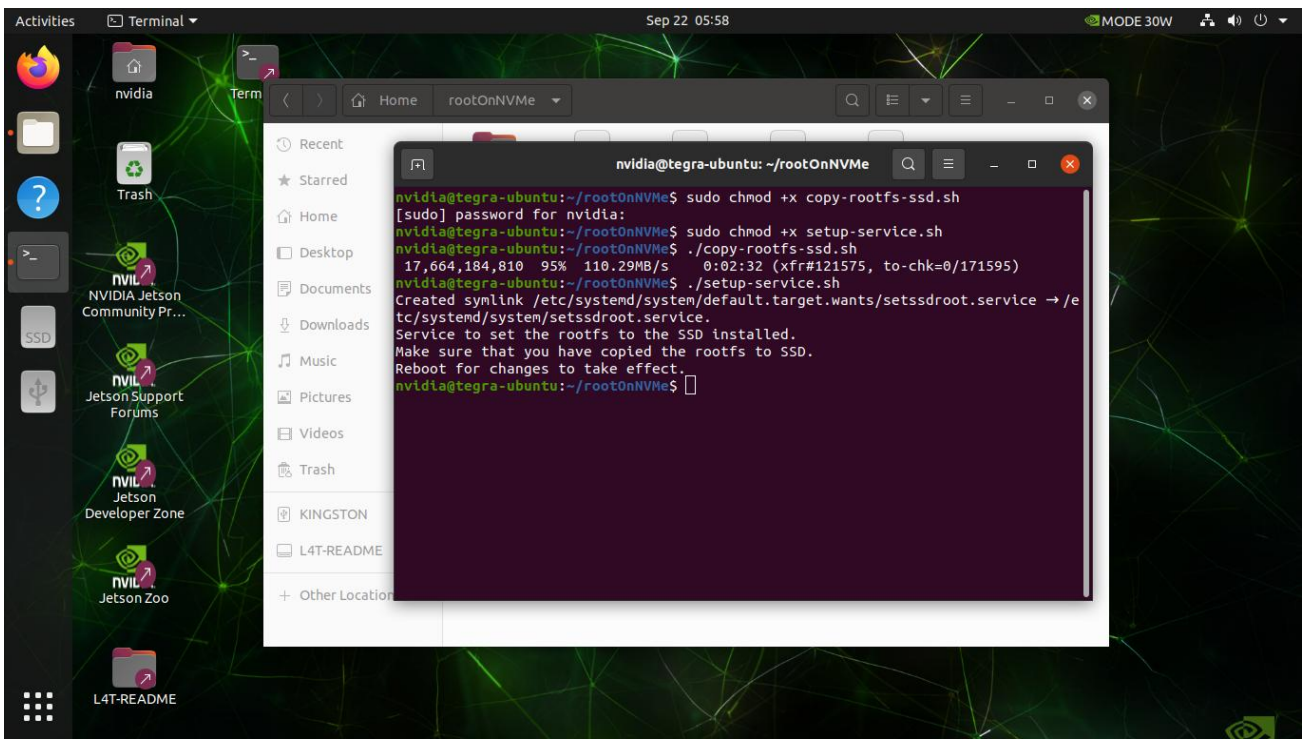


6)执行脚本:”setup-service.sh”

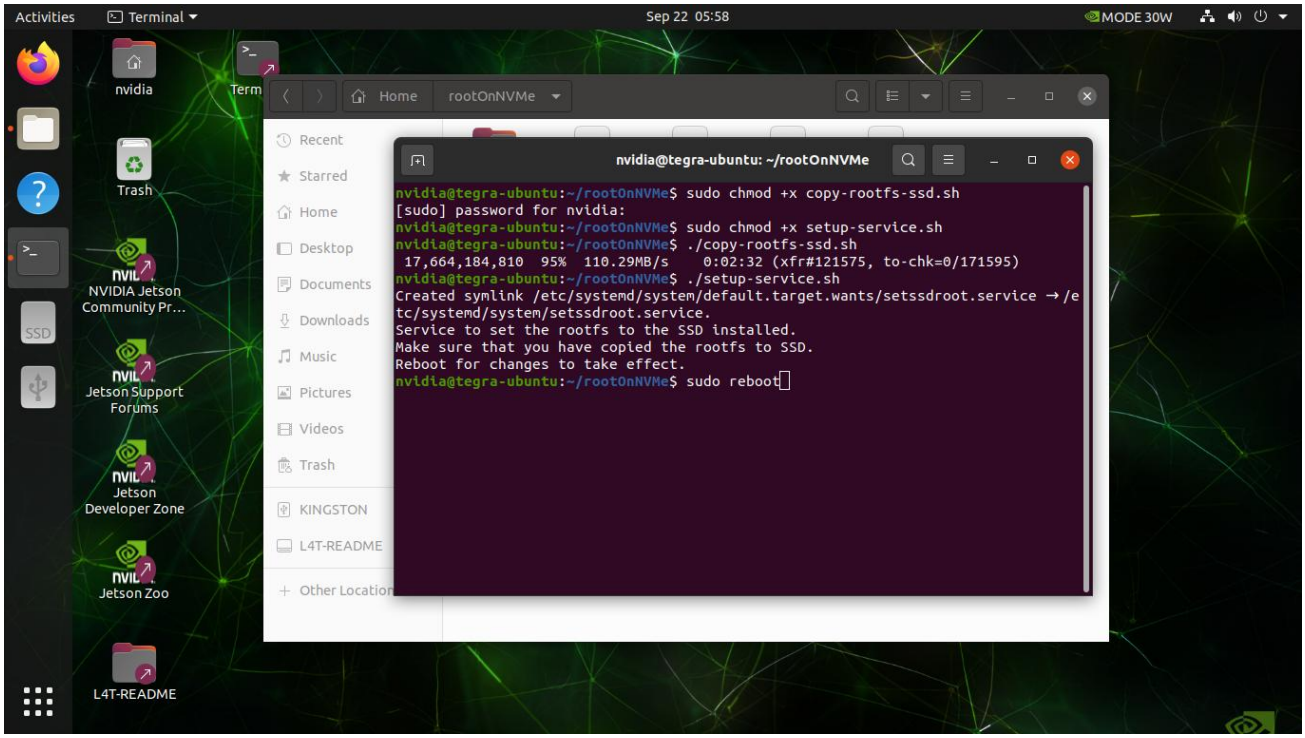


7)输入密码,默认为”nvidia”

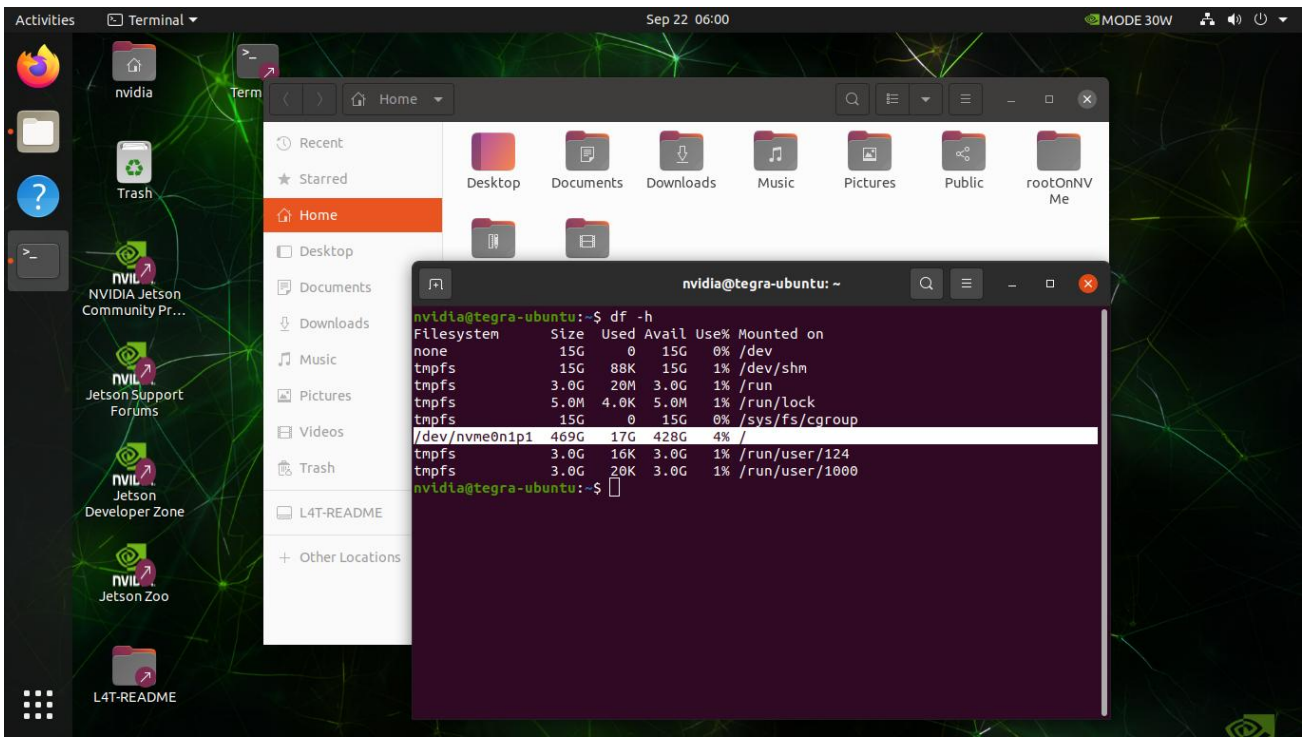




8)输入命令”sudo reboot”



9)等待重启完成后,打开终端输入命令”df -h”查看存储分布情况,如下图显示则为成功



Jtop 安装

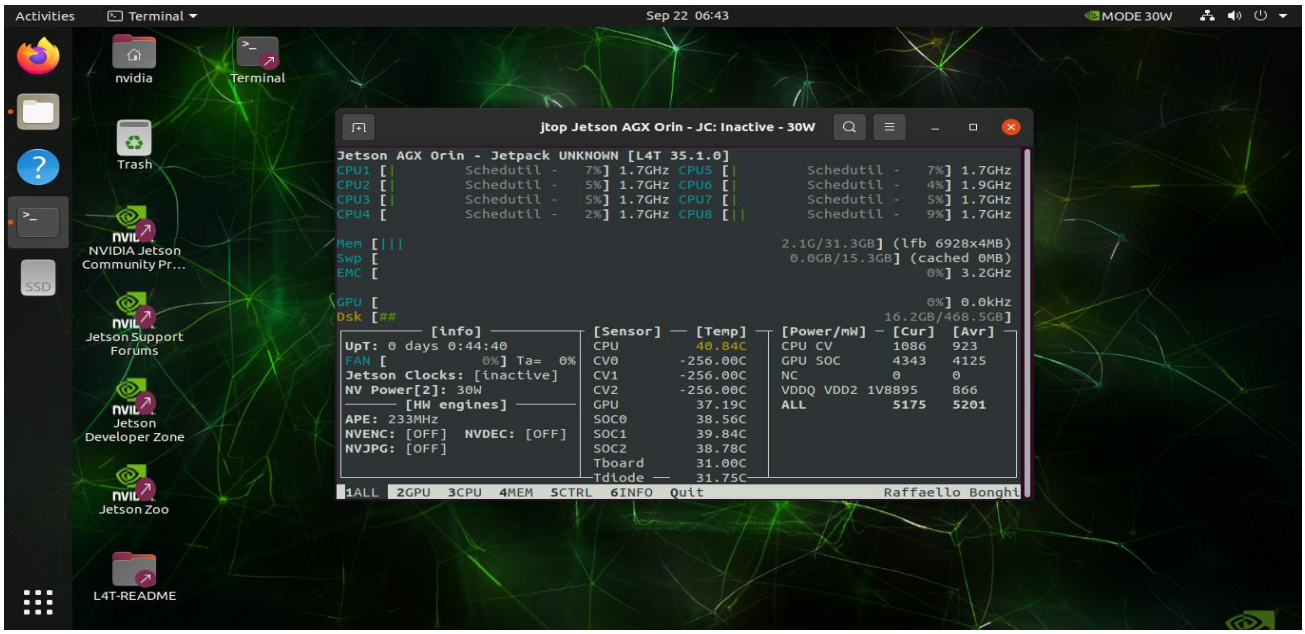
jtop（一个系统监视实用程序，可在终端上运行，并实时查看和控制 nvidia jetson 的状态 ），安装也非常方便，如果 jetson 产品上已安装了 jetpack sdk，可以按照如下步骤安装运行：

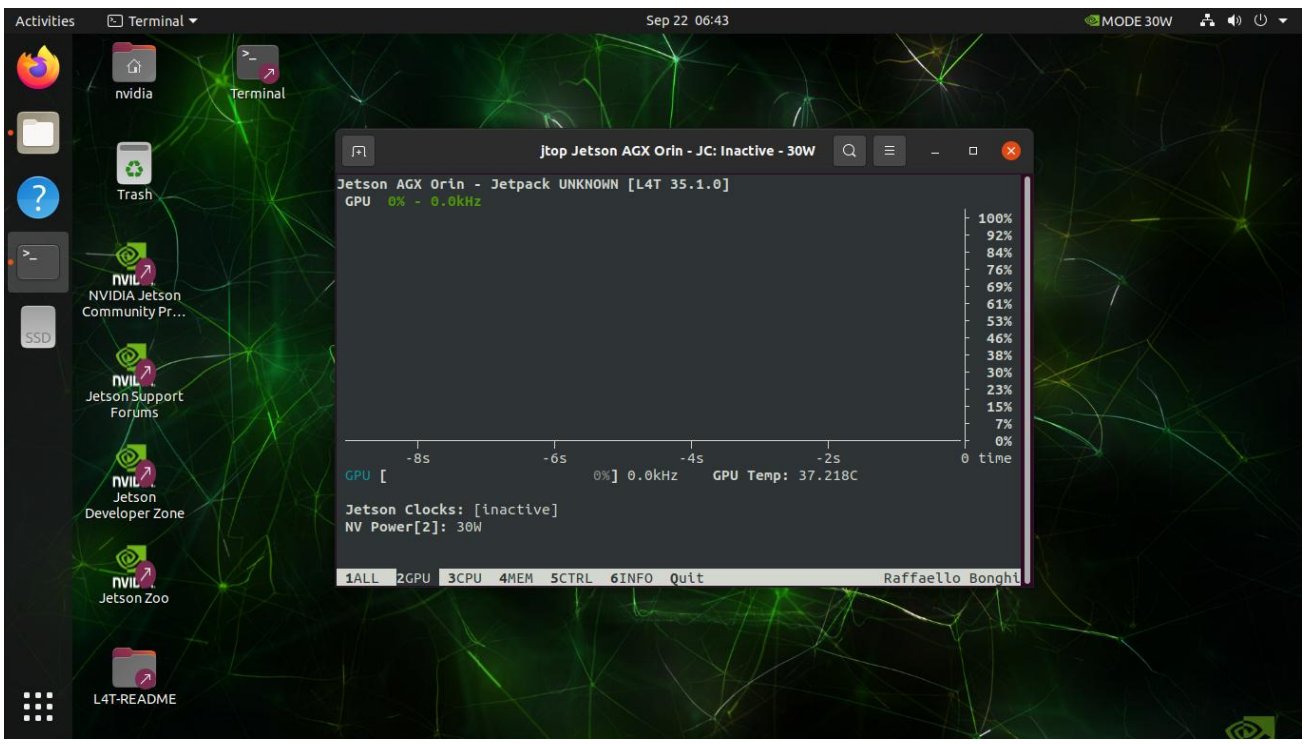
安装及运行

sudo apt install python3-pip

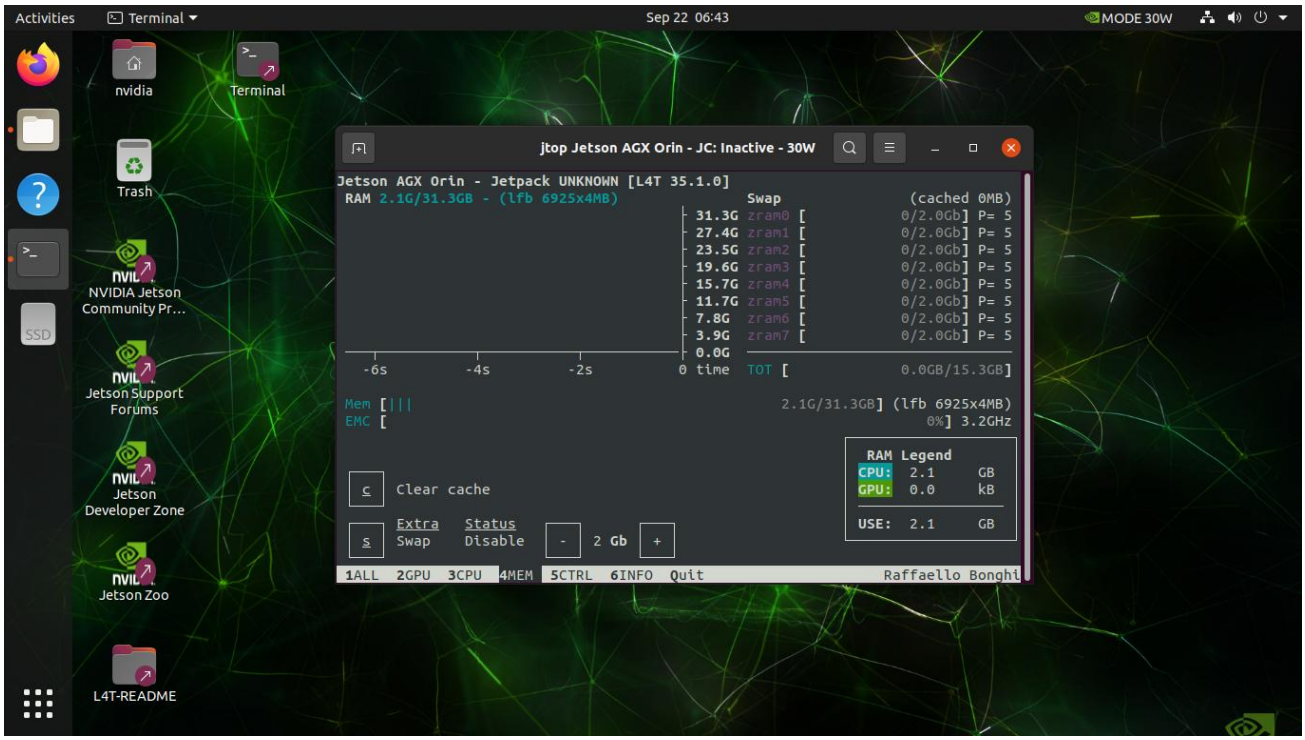
sudo pip3 install jetson-stats

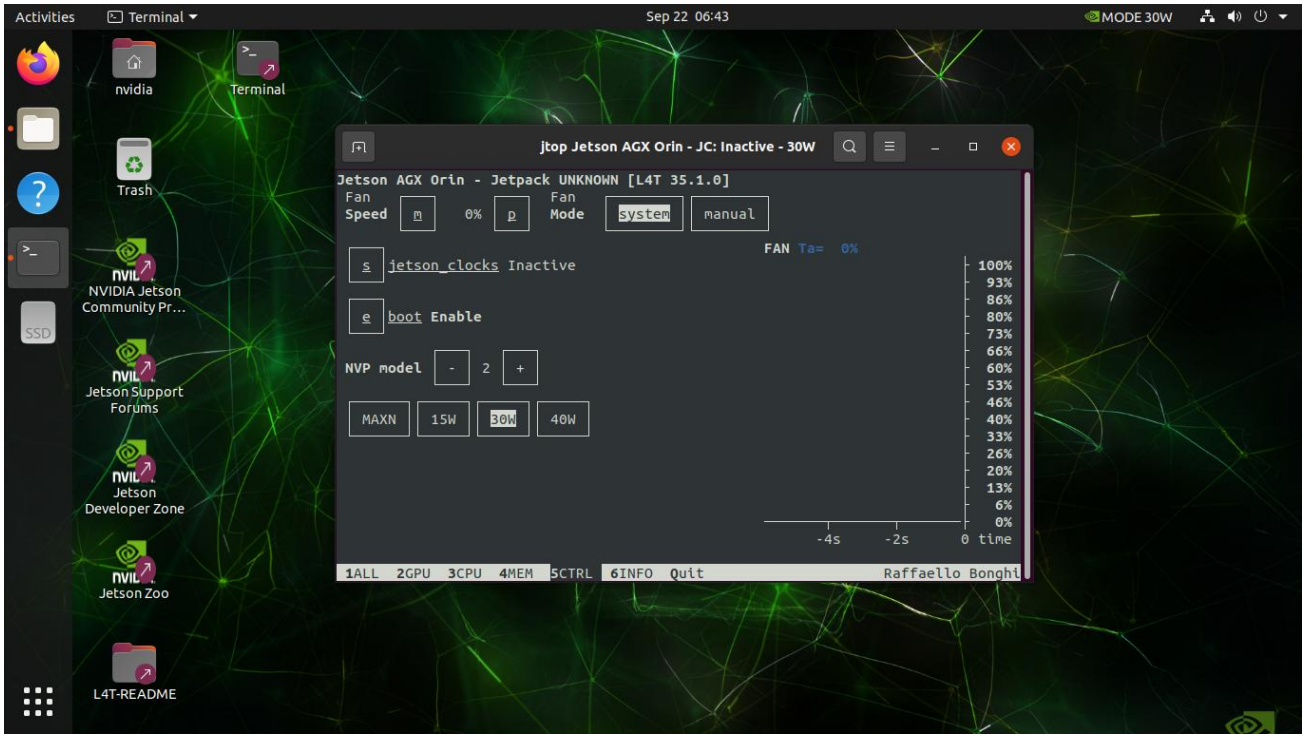
开启后的界面如下,分别按数字键 1 2 3 4 5 6 进行页面切换,按 Q 退出

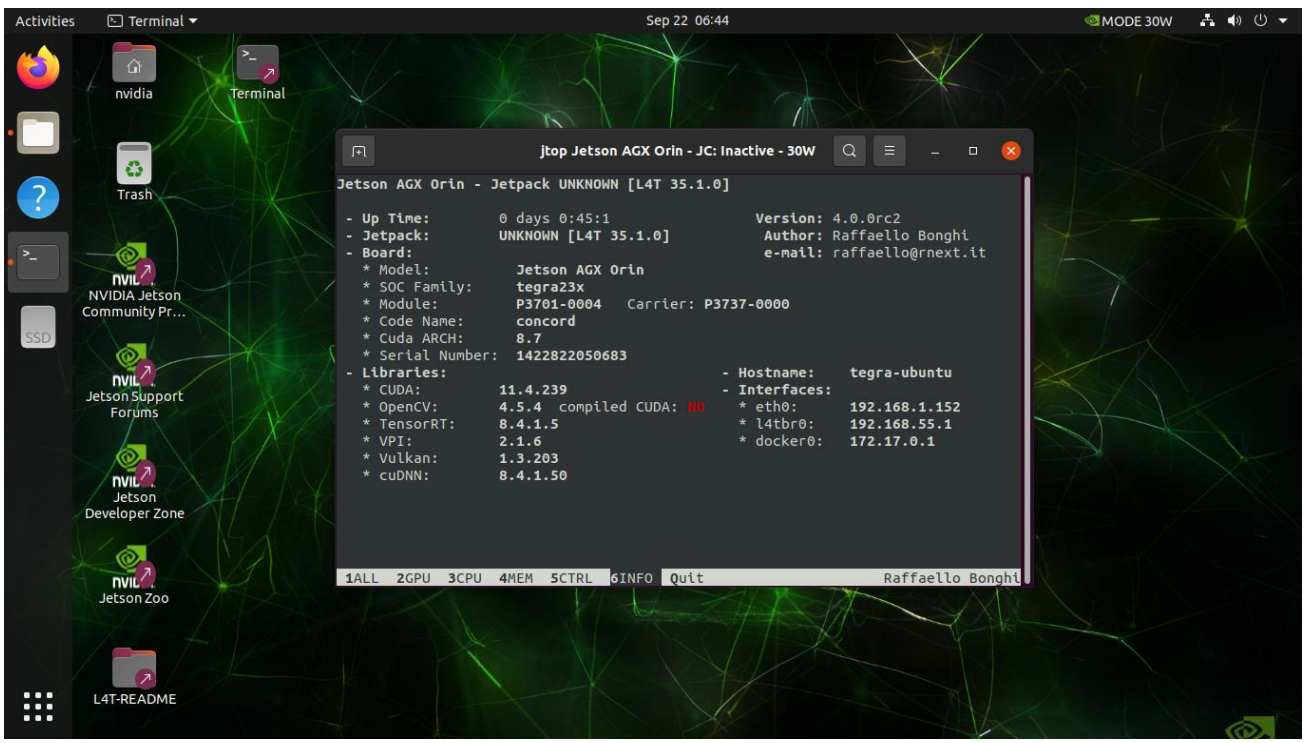





Activities
Terminal
Sep 22 06:43
MODE 30W
nvidia
Terminal
Trash
NVIDIA Jetson Community Pr...
Jetson Support Forums
Jetson Developer Zone
Jetson Zoo
L4T-README
jtop Jetson AGX Orin - JC: Inactive - 30W
Jetson AGX Orin - Jetpack UNKNOWN [L4T 35.1.0]
Platform
Arch: aarch64
Rel: 5.10.104-tegr
Cluster:
CPU1 ARMv8
Frq: 1.7GHz
CPU2 ARMv8
Frq: 1.7GHz
CPU3 ARMv8
Frq: 1.7GHz
CPU4 ARMv8
Frq: 1.7GHz
CPU5 ARMv8
Frq: 1.7GHz
CPU6 ARMv8
Frq: 1.7GHz
CPU7 ARMv8
Frq: 1.7GHz
CPU8 ARMv8
Frq: 1.7GHz
CPU1 3% S
CPU2 3% S
CPU3 9% S
CPU4 1% Schedut
CPU5 0% S
CPU6 9% S
CPU7 3% S
CPU8 1% Schedut
100%
87%
75%
62%
50%
37%
25%
12%
0%
-4s -2s 0 time
1ALL 2GPU 3CPU 4MEM 5CTRL 6INFO Quit
Raffaello Bonghi


Activities
Terminal
Sep 22 06:43
MODE 30W
nvidia
Terminal
Trash
NVIDIA Jetson Community Pr...
Jetson Support Forums
Jetson Developer Zone
Jetson Zoo
L4T-README
jtop Jetson AGX Orin - JC: Inactive - 30W
Jetson AGX Orin - Jetpack UNKNOWN [L4T 35.1.0]
RAM 2.1G/31.3GB - (lfb 6925x4MB)
Swap (cached 0MB)
31.3G zram0 [0/2.0Gb] P= 5
27.4G zram1 [0/2.0Gb] P= 5
23.5G zram2 [0/2.0Gb] P= 5
19.6G zram3 [0/2.0Gb] P= 5
15.7G zram4 [0/2.0Gb] P= 5
11.7G zram5 [0/2.0Gb] P= 5
7.8G zram6 [0/2.0Gb] P= 5
3.9G zram7 [0/2.0Gb] P= 5
0.0G
-6s -4s -2s 0 time
TOT [0.0GB/15.3GB]
Mem [|||
2.1G/31.3GB] (lfb 6925x4MB)
EMC [
0%] 3.2GHz
RAM Legend
CPU: 2.1 GB
GPU: 0.0 kB
USE: 2.1 GB
c Clear cache
s Extra Swap
Status Disable
- 2 Gb +
1ALL 2GPU 3CPU 4MEM 5CTRL 6INFO Quit
Raffaello Bonghi





非常方便地看到当前 jetson 机器上的各种完整信息，一般在首页就可以读取到很丰富的数据信息：

1)all

包含模块运行信息包括：cpu、内存、gpu、磁盘、风扇、jetson_clock 状态、nvpmodel 等等

2)gpu

实时 gpu 状态

3)cpu

实时 cpu 状态

4)mem

内存状态

5)ctrl

可以控制的状态,包括风扇转速以及功率模式的选择;

6)info

lib 库、cuda、serial number、interface 等信息,**自带 opencv 不支持 cuda,若需要支持 cuda,需要卸载自带 opencv 然后手动编译 opencv;**

可以使用以下键盘命令，控制 nvidia jetson 的相关配置：

在第 3 页 mem 中 ：

c ： 清除缓存

s ： 启用/禁用额外交换

+/-: 增加和减少交换大小

在第 4 页中 ctrl ：

启动 /停止 jetson_clocks 服务（注：仅 jetson_clocks 从时间 60 秒后开始）

e: 在启动时启用/禁用 jetson_clocks

+/-: 增加和减少 nvp 模型

f: 风扇的手动/ jetson_clocks 模式

p/m: 增加和降低风扇速度