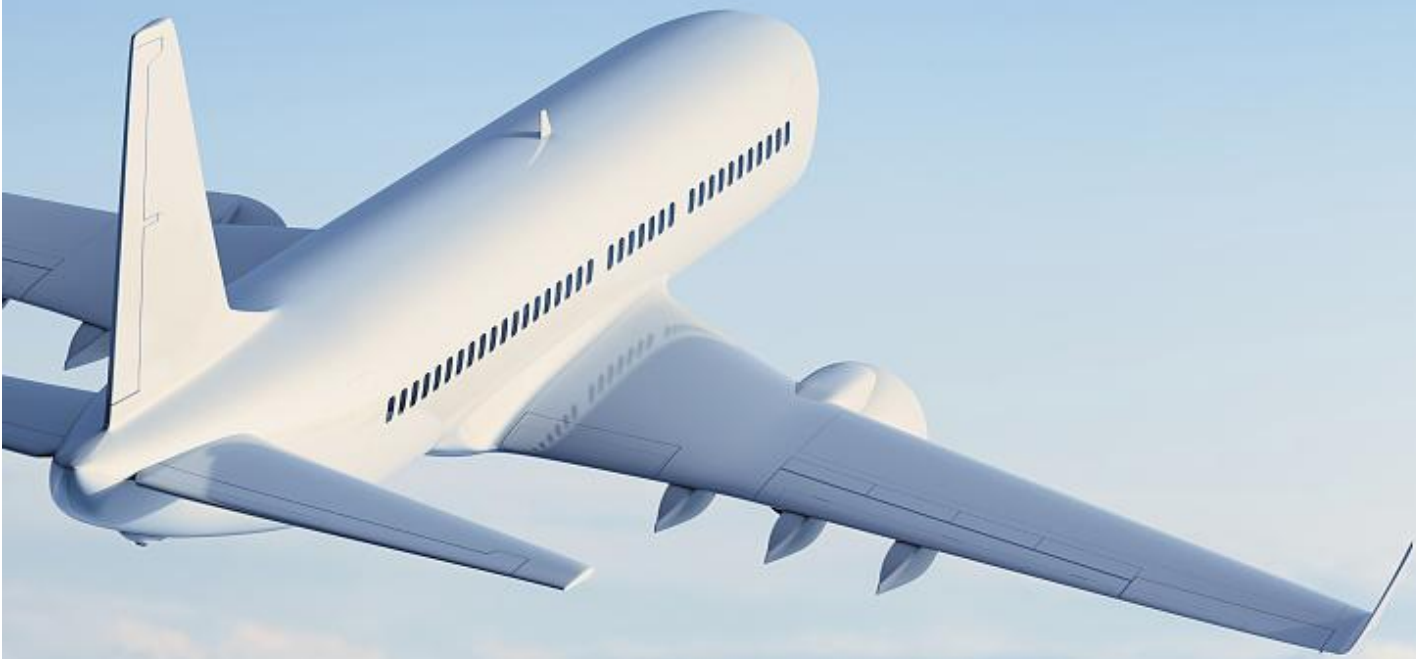
图为 TW-T50101S 手册

TWOWIN TW-T50101S MANUAL



引领边缘计算，成为人工智能领域解决方案的领军企业

图为信息科技(深圳)有限公司
TWOWIN TECHNOLOGY Co. Ltd

目录

1. 简介

1.1 Jetson Nano 模组介绍

Jetson Nano 是一款小型、功能强大的计算机，适用于嵌入式 AI 系统和物联网，可在低功耗平台中提供现代 AI 的强大功能。借助 NVIDIA Jetpack SDK 和完整的桌面 Linux 环境快速入门，并开始探索嵌入式产品的新世界。

下面详细列举一些 Jetson Nano 的优势：

(1) 体型小巧，性能强大，价格实惠，整体采用类似树莓派的硬件设计，支持一系列流行的 AI 框架，并且英伟达投入了大量的研发精力为其打造了与之配套的 Jetpack SDK 开发包，通过该开发包可以使学习和开发 AI 产品变得更加简单和便捷。

(2) 专为 AI 而设计，性能相比树莓派更强大，搭载 Quad-core ARM Cortex-A57 MPCore processor 处理器，NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores GPU 及 4GB LPDDR4 内存。

(3) 支持英伟达的 NVIDIA JetPack 组件包，其中包括用于深度学习、计算机视觉、GPU 计算、多媒体处理等的板级支持包，CUDA，cuDNN 和 TensorRT 软件库。

(4) 支持一系列流行的 AI 框架和算法，比如 TensorFlow，PyTorch，Caffe / Caffe2，Keras，MXNet 等，使得开发人员能够简单快速的将 AI 模型和框架集成到产品中，轻松实现图像识别，目标检测，姿势估计，语义分割，视频增强和智能分析等强大功能。

目前人工智能风口开始逐步进入落地应用阶段，更多产品希望能够将人工智能算力运用于实际终端，即实现所谓的边缘计算需求。从根本上来说近几年推动人工智能的核心在于深度学习算法，但是深度学习的推理加速离不开高速 GPU 的支持，而一般桌面 PC 或服务器级别的显卡（如英伟达 1080Ti 等）价格非常昂贵，不适合边缘计算需求，而且体积于庞大。因此，英伟达推出的这款 Jetson Nano 非常契合当前行业需求.

以下为 Jetson Nano 模组具体参数列表：

	Jetson Nano
AI Performance	472 GFLOPS
GPU	128-core NVIDIA Maxwell™ architecture GPU
GPU Max Frequency	921MHz
CPU	Quad-core ARM Cortex-A57 MPCore processor
CPU Max Frequency	1.43GHz
运行内存	4GB 64-bit LPDDR4 25.6GB/s
存储	16GB eMMC5.1
视频编码	1x 4K30 (H.265) 2x 1080p60 (H.265)
视频解码	1x 4K60 (H.265) 4x 1080p60 (H.265)
摄像头	Up to 4 cameras 12 lanes MIPI CSI-2D-PHY 1.1 (up to 18 Gbps)
网络连接	Wi-Fi
	10/100/1000 BASE-T 以太网
显示	2 multi-mode DP 1.2/eDP 1.4/HDMI 2.01 x2 DSI (1.5Gbps/lane)
USB*	1x USB 3.0 (5 Gbps) 3x USB 2.0

PCIE*	1 x4(PCIe Gen2)
I/O	3x UART, 2x SPI, 2x I2S, 4x I2C, GPIOs
Power	5W-10W
大小	69.6 mm x 45 mm
规格尺寸	260-pin SO-DIMM connector

1.2 TW-T50101S 介绍

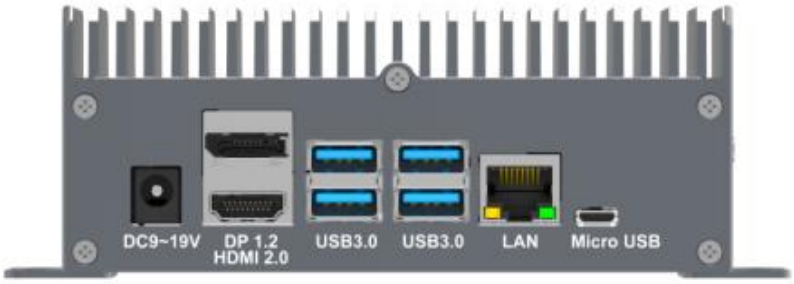
TW-T50101S 为一款基于 NVIDIA® JETSON Nano™系列模块面向无人驾驶车载系统的计算平台，内置集成 Nano 模块，可预装 Ubuntu 18.04 操作系统，具备 472 GFLOPS 的 AI 处理能力，采用超强固轻型铝合金材料设计，风扇主动+外壳被动散热，具备优秀的散热能力，外观新颖，支持 USB、UART、GPIO 等丰富 IO 接口类型，可内置 WIFI 模块支持无线网络连接。预留便于坚固安装支架，具有计算能力强、可靠性高、集成度高、功耗低特点，可应用于无人清洁车、无人配送车、智能巡检、AGV 等无人驾驶领域。

TW-T50101S 边缘计算平台亮点概述

- 内嵌 NVIDIA® JETSON Nano™
- 支持 M.2 KEY M (PCIEX4 NVME 2280)
- 支持多种接口(如 USB/以太网口/串口/GPIO 等)
- 支持双频 WIFI 2.4G/5G
- 外壳被动散热设计
- 内置 ubuntu 18.04 系统和 JetPack SDKS

整机展示图

接口介绍

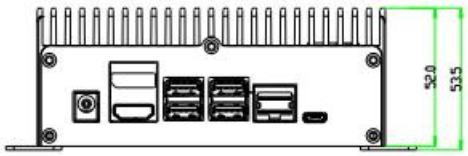


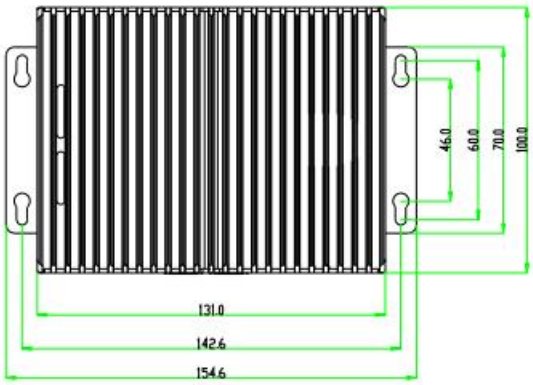
正面接口示意图

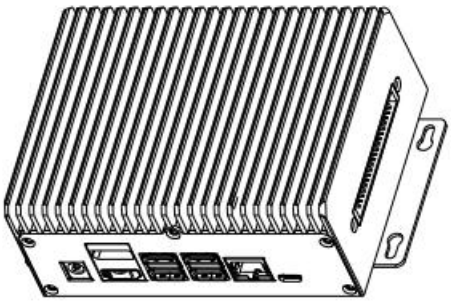
接口名称	数量	描述
DC	1	DC（直流）供电口 支持宽压 9-19V 输入
DP	1	DP 1.2 视频输出口（不可作为输入）
HDMI	1	HDMI 2.0 视频输出口（不可作为输入）
USB	4	USB 3.0(由内部核心接到 1 个 HUB 芯片引出 4 个 USB3.0,共享 5G 带宽 ）5V-1A
LAN	1	RJ45 千兆网口（正常状态为：橙色常亮，绿色闪烁）
Micro-USB	1	USB2.0 接口 具备对外供电以及数据传输功能，最重要功能为 USB 刷机口

WiFi 天线连接示意图

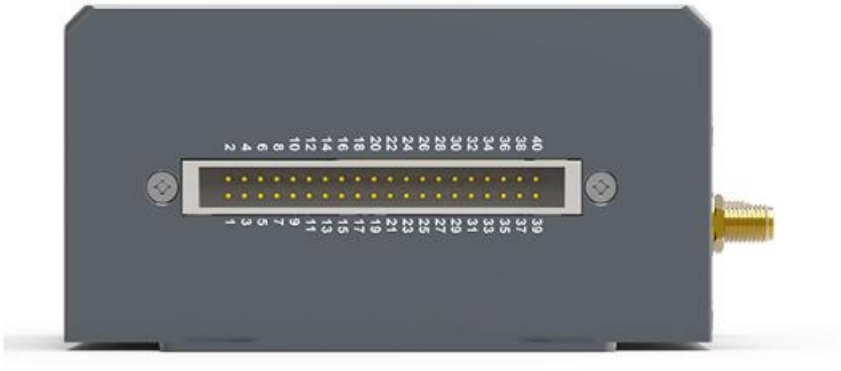
整机尺寸示意图







40 PIN 展示图



PIN/BOARD NUM	BCM NUM	GPIO NUM	PIN/BOARD NUM	BCM NUM	GPIO NUM
7	4	216	**11**	17	50
12	18	79	**13**	27	14
15	22	194	**16**	23	232
18	24	15	**19**	10	16
21	9	17	**22**	25	13
23	11	18	**24**	8	19
26	7	20	**29**	5	149
31	6	200	**32**	12	168
33	13	38	**35**	19	76
36	16	51	**37**	26	12
38	20	77	**40**	21	78

注意：

❶ 8 &10 脚为本设备**唯一**可以用来**数据传输**的 UART(TTL 信号）串口，串口设备号为“/dev/ttyTHS1”；

❷ 40 PIN 中的多数引脚具有复用功能，默认状态为 GPIO，若需要修改其默认状态，则需参考以下链接介绍：

https://docs.nvidia.com/jetson/archives/r36.3/DeveloperGuide/HR/ConfiguringTheJetsonExpansionHeaders.html

❸ GPIO 使用终端命令方法简述：(以下（）内容为备注说明）

# 切入 root 账号：
sudo -s（按回车并输入密码“nvidia”）
# 将需要使用的 GPIO 号输入/sys/class/gpio/export 文件 （以下以 7 号脚 216 为例）
echo 216（216 为 GPIO NUM） > /sys/class/gpio/export
进入对应的 GPIO 文件
cd /sys/class/gpio/gpio216
赋予执行权限
chmod 777 value
赋予执行权限
chmod 777 direction
设置为输入状态
echo in > direction
设置为输出状态
echo out > direction
设置为输出高电平
echo 1 > value
设置为输出低电平
echo 0 > value

❹ 应用示例

使用 Python 调用 GPIO 示例：
/usr/share/doc/jetson-gpio-commmon/samples(本机自带路径)
使用 C++调用 GPIO 示例：
https://github.com/pjueon/JetsonGPIO/tree/master(github 示例需自行研究）

2. 环境配置

2.1 烧录系统(我司出厂产品均已安装好系统，本节内容给需要刷机的客户参考用)

英伟达官方为 Jetson Nano 提供的刷机方式只能只用线刷的方式,通过 nvidia 官网提供的 SDK Manager 工具或者我司提供的系统镜像两种方式进行刷机,以下进行分别介绍,两种方式任选一种即可.

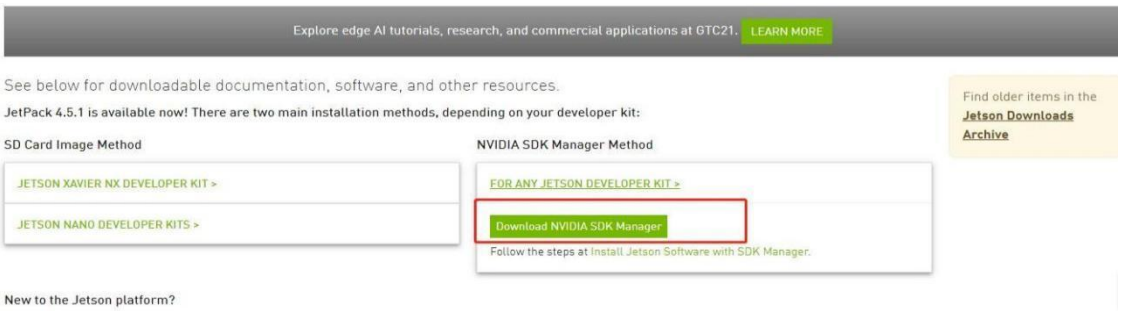
2.1.1 SDK manager 介绍

SDK Manager 是 NVIDIA 官方提供的一个 Jetson 系统管理工具,用来进行 Jetson 系列产品的系统及软件安装,是一个优缺点并存的工具.优点是支持目前 nvidia jetson 系列所有产品的系统及软件安装,步骤简单,操作方便,整合了多个 Jetpack 版本以及对应的 SDK,无需再去官网下载 L4T 文件,方便用户能更好安装例如 cuda,cudnn,opencv 以及 deepstream 这类 sdk.

2.1.2 安装与准备

首先说明 SDK manager 是需要安装在一台 ubuntu18.04 或者 20.04 的电脑上,而不是安装在任何一款 jetson 系列产品上.因为 jetson 系列产品为 arm64 位架构,SDK manager 并不支持 arm 架构.

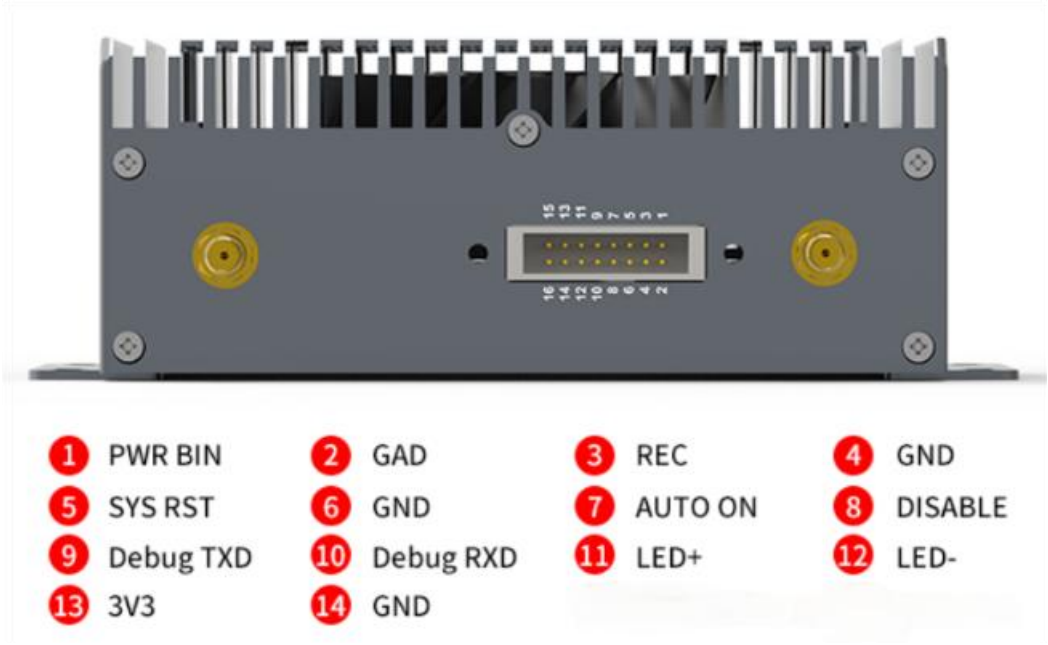
SDK manager 下载链接：
https://developer.nvidia.com/embedded/downloads



百度云盘下载链接:
链接：https://pan.baidu.com/s/1HOLORIpblrhpmJvzweYfOg 提取码：tw12

2.1.3 系统及软件安装

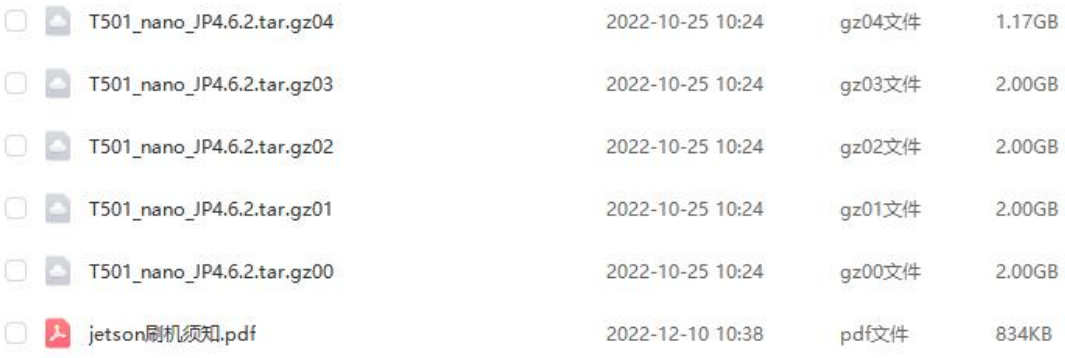
首先将 TW-T50101S 与 ubuntu 电脑通过 Micro-USB 数据线相连,使用跳线帽短接 3 号脚 REC 和 4 号 GND 针脚,针脚定义如下图所示,连接完毕后，需要给 jetson 设备上电,不需要另外接显示器等其他外设.



2.1.4 使用镜像包刷机

镜像包下载地址

链接：https://pan.baidu.com/s/1V5TDZhqw17X950aNfYDJ5g?pwd=qk8t



具体使用方法请查看镜像包里的“jetson 刷机须知.txt”文件.

2.2 开发环境配置(以下内容非必须进行的操作,请按照个人需求进行相应操作)

2.2.1 更新源和软件

安装完系统后首先应该更新源，否则后续更新和升级会非常慢。但是由于 Jetson Nano 采用的是 aarch64 架构的 Ubuntu 18.04 LTS 系统，与 AMD 架构的 Ubuntu 系统不同，因此需要替换成 aarch64 的源，这里一定要注意，不要替换成 x86-64 的源了。

我们这里选择清华的源进行更新。首先备份原本的源，更改 source.list 文件的名字，以备不时之需：

sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak

sudo vi /etc/apt/sources.list

然后删除所有内容，复制以下内容并进行保存：

deb http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-updates main restricted universe multiverse

deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-updates main restricted universe multiverse

deb http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-security main restricted universe multiverse

deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-security main restricted universe multiverse

deb http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-backports main restricted universe multiverse

deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-backports main restricted universe multiverse

deb http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic main universe restricted

deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic main universe restricted

打开终端，输入下述命令进行更新：

sudo apt-get update

上述更新时间较长，中间可能由于网速的关系会更新失败，此时不要关机重新执行命令即可，会自动断点续传的。

2.2.2 用命令设置固定 IP

(1)在/etc/network/interfaces.d 目录下创建 eth0 文件：

sudo touch eth0

sudo vim eth0

在文件中加入如下内容：

auto eth0

iface eth0 inet static

address 172.1.6.88(此处按照自身需求进行静态 ip 的设置)

netmask 255.255.255.0

gateway 172.1.1.254

(2)修改/etc/network 目录下的 interfaces 文件：

sudo vim interfaces

增加一行：

source interfaces.d/eth0

(3)重启设备使设置生效

sudo reboot

2.2.3 安装 Qt5

在实际的产品部署阶段，考虑到终端设备速度、稳定性、内存占用等因素，一般会采用 C++来开发最终的成品，而只有在产品模型设计阶段才会使用 python 进行算法开发。因此，需要一款能够在 Jetson NX 中开发 C++的编译器方便我们开发落地产品。VS Code 本身可以开发 C++应用，但是 Code-OSS 对于 C++的支持并不好，因此，需要另外安装一个优秀的 C++编译器来完成 C++开发任务。本文推荐使用 Qt。

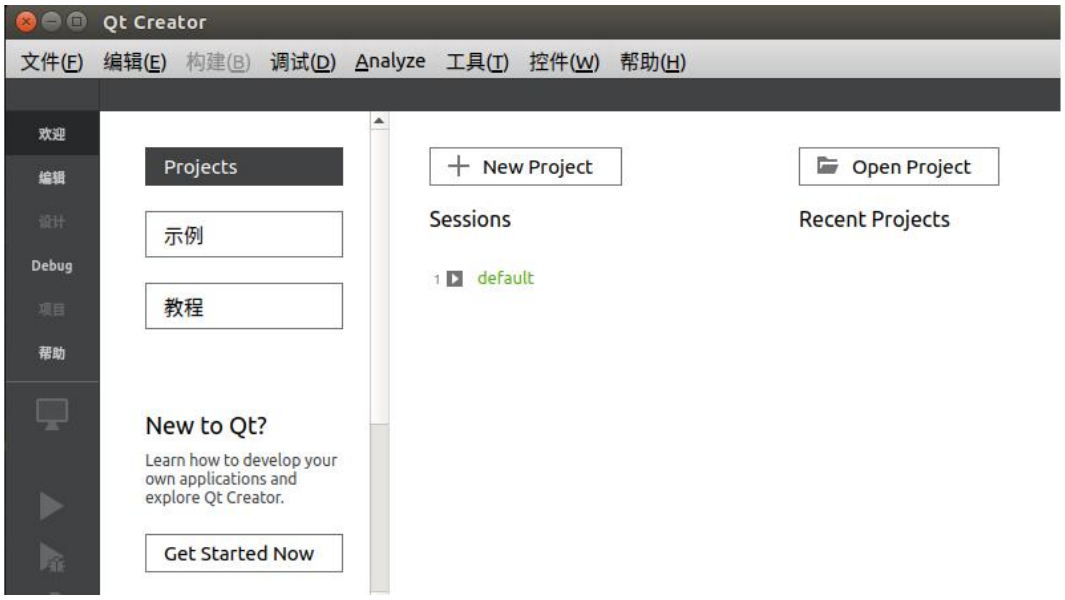
Qt 是一个跨平台的 C++ 开发库，主要用来开发图形用户界面（Graphical User Interface，GUI）程序，当然也可以开发不带界面的命令行（Command User Interface，CUI）程序。Qt 是纯 C++ 开发的，所以用它来开发 C++应用具有天然的优势。Qt 支持的操作系统有很多，例如通用操作系统 Windows、Linux、Unix，智能手机系统 Android、iOS、WinPhone 以及嵌入式系统 QNX、VxWorks 等等。当然，QT 也完全支持 Jetson NX 的 Ubuntu 环境。

Jetson NX 下安装 QT 比较简单，只需要输入命令：

```
sudo apt-get install qt5-default qtcreator -y
```

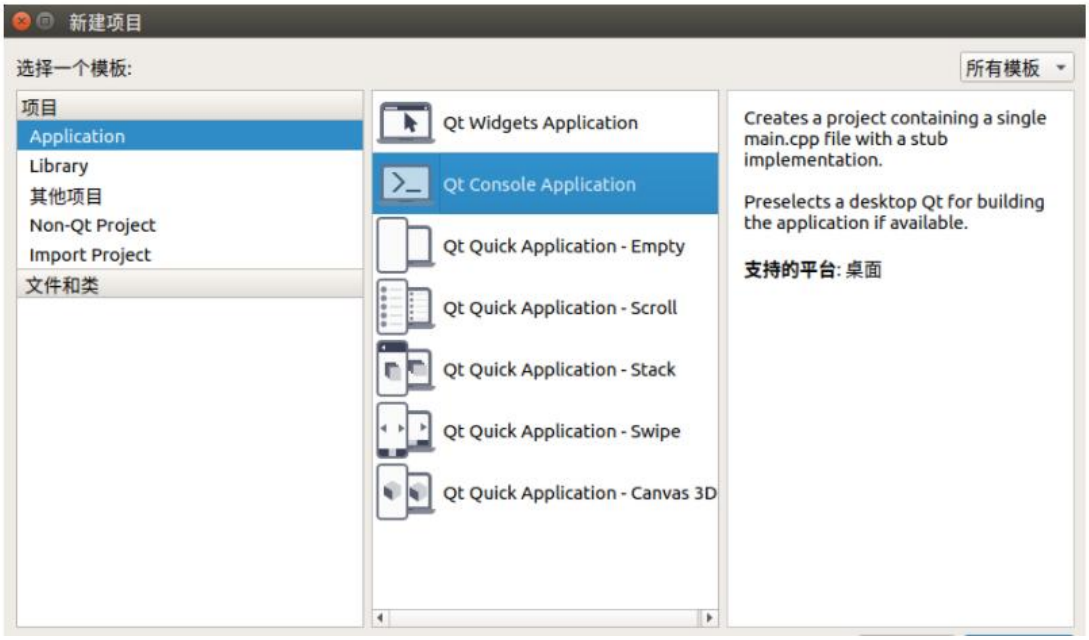
安装完成后，同样在搜索菜单中搜索 Qt，然后会出现 Qt Creator，这个即为 Qt 的 IDE，打开它。接下来简单演示如何创建一个简单的 C++控制台程序。

打开 Qt Creator，如下图所示：

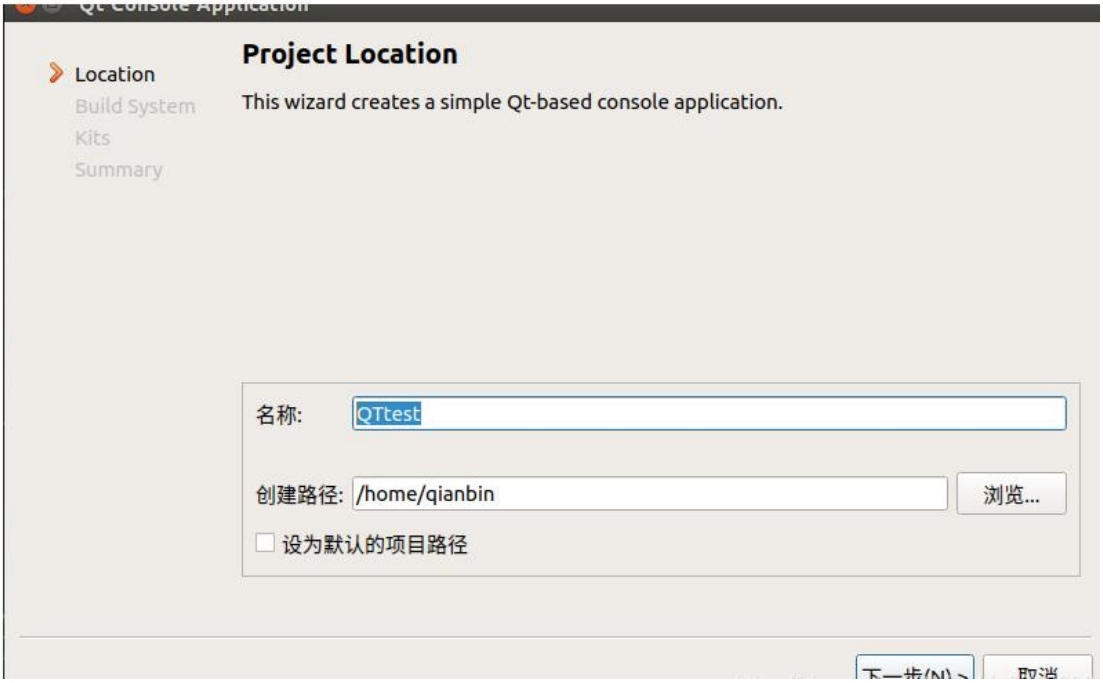


单击 New Project 创建一个新项目，这里选择 Application 下的 Qt COnsole Appliation 应用，即创建一个 Qt

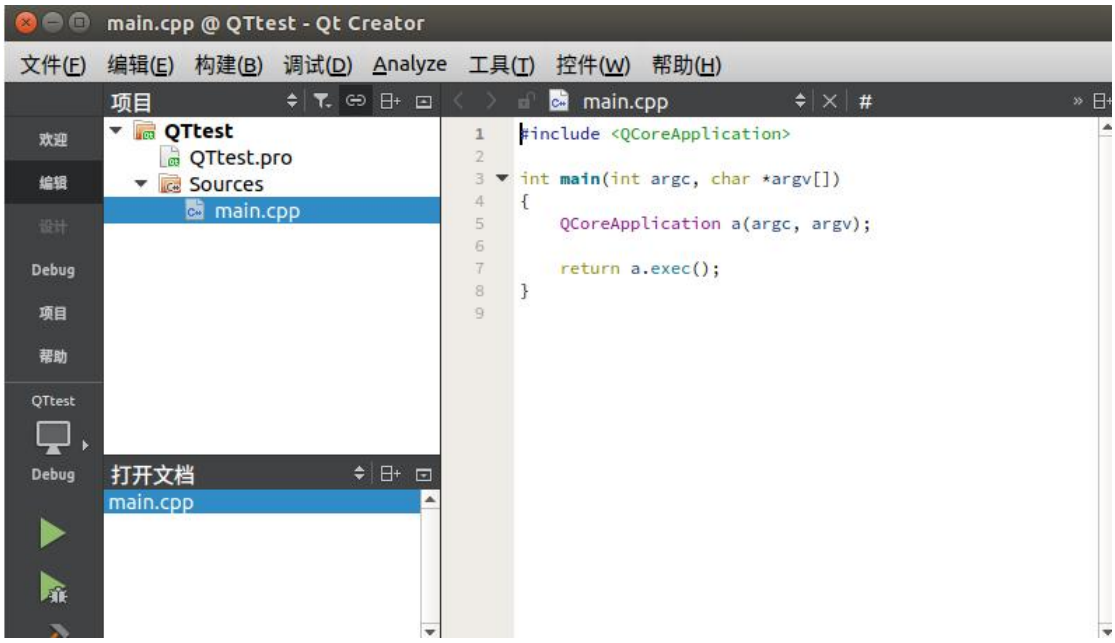
版的 C++控制台程序：



然后工程命名为 QTtest：



然后一直默认单击下一步即可完成项目的创建。可以看到，Qt 已经为我们创建了一个 C++文件 main.cpp 用于编写 C++代码，并且还有一个 QTtest.pro 配置文件用于为整个项目进行配置，效果如下图所示：



此时可以直接按 ctrl+r 键运行项目，但是由于我们并没有任何输出代码，所以弹出的终端也没有输出任何值。我们修改一下 main.cpp 的代码，同样来执行两个整数的相加并输出其结果，完成代码如下：

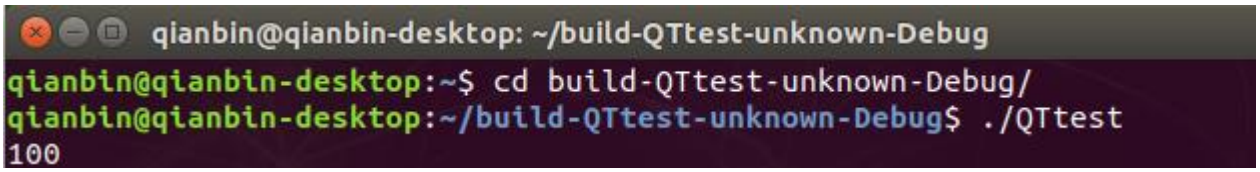
```
#include <QCoreApplication>
#include <QtDebug>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    int c=64;
    int d=36;
    qDebug() << (c+d);
    return a.exec();
}
```

此时，再重新按 ctrl+r 键运行项目输出下图所示结果：

最后我们再看一下在主目录下生成了一个与 QTtest 对照的 debug 可执行项目 build-QTtest-unknown-Debug，在这个文件夹中生成来 debug 版本的 QTtest 可执行程序。通过终端 cd 命令进入到该文件夹，然后输入

./QTtest

会直接执行程序，如下图所示：



也就是说本质上我们已经成功的部署开发了一个应用，该应用功能很简单，仅仅实现了两个固定整数的相加。尽管简单，但是却梳理了我们正常开发人工智能产品的一种比较常见的形式，即先在 VS Code 中用 python 脚本进行算法验证，最后再用 QT 编写对应的 C++应用，最后生成二进制可执行程序，这个最终生成的二进制可执行程序就是我们的“产品”，这个可执行程序代码是封装起来的、不可见的、可以直接运行的。

至此，我们已经完成了 Jetson NX 的常规开发配置，接下来会进行几个小项目的演示，使读者可以更深入的学习 Jetson NX 的开发方法。

3. Linux 操作基础

3.1 常见指令介绍

sudo

- sudo 命令以系统管理者的身份执行指令。
- 要想使用 root 用户，可使用 waveshare 用户登录，执行下面命令

```
sudo  su     #切换为超级用户
su  waveshare  #切换普通用户
```

ls

- ls 命令用于显示指定工作目录下之内容（列出目前工作目录所含之文件及子目录)。
- 常用的指令：

```
ls
ls  -a     #显示所有文件及目录  (. 开头的隐藏文件也会列出)
ls  -l     #除文件名称外，亦将文件型态、权限、拥有者、文件大小等资讯详细列出
ls  -lh    #文件大小以容易理解的格式列出，例如 4K
```

- 想要学习了解指令更多参数，我们可以使用 help 指令来查看：

```
ls  --help
```

touch

- touch 命令用于修改文件或者目录的时间属性，包括存取时间和更改时间。若文件不存在，系统会建立一个新的文件。
- 例如，在当前目录下，使用该指令创建一个空白文件"file.txt"，输入如下命令：

```
touch  file.txt
```

mkdir

- mkdir 命令用于创建目录。
- 在工作目录下，建立一个名为 waveshare 的子目录：

```
sudo mkdir waveshare
```

-
- 在工作目录下建立一个名为 waveshare/test 的目录。

```
sudo mkdir -p waveshare/test
```

-
- 若 waveshare 目录原本不存在，则建立一个。（注：本例若不加 -p 参数，且原本 waveshare 目录不存在，则产生错误。）

cd

- 切换当前工作目录。

```
cd ..                #返回上一层目录
cd /home/waveshare #进入/home/waveshare 目录
cd                   #返回用户目录
```

cp

- cp 命令主要用于复制文件或目录。
- 参数:
  - -a：此选项通常在复制目录时使用，它保留链接、文件属性，并复制目录下的所有内容。其作用等于 dpR 参数组合。
  - -d：复制时保留链接。这里所说的链接相当于 Windows 系统中的快捷方式。
  - -f：覆盖已经存在的目标文件而不给出提示。
  - -i：与 -f 选项相反，在覆盖目标文件之前给出提示，要求用户确认是否覆盖，回答 y 时目标文件将被覆盖。
  - -p：除复制文件的内容外，还把修改时间和访问权限也复制到新文件中。
  - -r：若给出的源文件是一个目录文件，此时将复制该目录下所有的子目录和文件。
  - -l：不复制文件，只是生成链接文件。

- 使用指令 cp 将当前目录 test/下的所有文件复制到新目录 newtest 下，输入如下命令：

```
sudo cp –r test/ newtest
```

mv

- mv 命令用来为文件或目录改名、或将文件或目录移入其它位置。
- 参数:
  - -b: 当目标文件或目录存在时，在执行覆盖前，会为其创建一个备份。
  - -i: 如果指定移动的源目录或文件与目标的目录或文件同名，则会先询问是否覆盖旧文件，输入 y 表示直接覆盖，输入 n 表示取消该操作。
  - -f: 如果指定移动的源目录或文件与目标的目录或文件同名，不会询问，直接覆盖旧文件。
  - -n: 不要覆盖任何已存在的文件或目录。
  - -u：当源文件比目标文件新或者目标文件不存在时，才执行移动操作。
- 使用指令 mv 将当前目录 test/下的 file1 文件复制到新目录/home/waveshare 下，输入如下命令：

```
sudo mv file1 /home/waveshare
```

rm

- rm 命令用于删除一个文件或者目录。
- 参数:
  - -i 删除前逐一询问确认。
  - -f 即使原档案属性设为唯读，亦直接删除，无需逐一确认。
  - -r 将目录及以下之档案亦逐一删除。
-
  - 删除文件可以直接使用 rm 命令，若删除目录则必须配合选项"-r"，例如：

```
sudo rm test.txt
```

  - rm：是否删除 一般文件 "test.txt"? y

```
sudo rm homework
```

  - rm: 无法删除目录"homework": 是一个目录

```
sudo rm -r homework
```

- rm：是否删除 目录 "homework"? y

reboot

- reboot 命令用于用来重新启动计算机,更改 Tinker Board 2 的配置经常需要重启。
- 参数:
  - -n：在重开机前不做将记忆体资料写回硬盘的动作
  - -w：并不会真的重开机，只是把记录写到 /var/log/wtmp 档案里
  - -d：不把记录写到 /var/log/wtmp 档案里（-n 这个参数包含了 -d）
  - -f：强迫重开机，不呼叫 shutdown 这个指令
  - -i：在重开机之前先把所有网络相关的装置先停止
- 重新启动

```
sudo reboot
```

shutdown

- Jetson Nano 的关机是不能直接拔掉电源线的,因为 Tinker Board 2 会将内存作为暂存区,如果直接拔掉电源线会使一些在内存中的数据没有来得及写入 SD 卡中，从而造成数据的丢失或是损坏 SD 卡上的数据,造成系统无法启动。
- 参数
  - -t seconds：设定在几秒钟之后进行关机程序。
  - -k：并不会真的关机，只是将警告讯息传送给所有使用者。
  - -r：关机后重新开机。
  - -h：关机后停机。
  - -n：不采用正常程序来关机，用强迫的方式杀掉所有执行中的程序后自行关机。
  - -c：取消目前已经进行中的关机动作。
  - -f：关机时，不做 fsck 动作(检查 Linux 档系统)。
  - -F：关机时，强迫进行 fsck 动作。
  - time：设定关机的时间。
  - message：传送给所有使用者的警告讯息。
- 实例
  - 立即关机

```
sudo shutdown -h now
```

- 指定 10 分钟后关机

```
sudo shutdown -h 10
```

- 重新启动计算机

```
sudo shutdown -r now
```

- 无论使用哪一个命令来关闭系统都需要 root 用户权限，如果用户使用 linaro 这样的普通用户，可以使用 sudo 命令暂时获得 root 权限。

pwd

- 该 pwd 命令显示当前工作目录的名称：在 Jetson NX 上，输入 pwd 将输出类似/home/waveshare。

head

- 该 head 命令显示文件的开头。可用于-n 指定要显示的行数（默认为 10 行），或与-c 指定字节数。

```
head test.py -n 5
```

tail

- 该 tail 显示文件的结尾。-c 字节或-n 行数指定文件中的起始点

df

- 用于 df 显示已安装文件系统上可用和使用的磁盘空间。用于 df -h 以可读的格式查看输出，使用 M 表示 MB，而不是显示字节数。

```
df -h
```

tar

- tar 命令是用来建立，还原备份文件的工具程序，它可以加入，解开备份文件内的文件。
- 压缩文件：

```
tar -cvzf waveshare.tar.gz *
```

- 解压文件：

```
tar -xvzf waveshare.tar.gz
```

apt

- apt（Advanced Packaging Tool）是一个在 Debian 和 Ubuntu 中的 Shell 前端软件包管理器。
- apt 命令提供了查找、安装、升级、删除某一个、一组甚至全部软件包的命令，而且命令简洁而又好记。
- apt 命令执行需要超级管理员权限(root)。
- apt 常用命令
  - 列出所有可更新的软件清单命令：sudo apt update
  - 升级软件包：sudo apt upgrade
  - 列出可更新的软件包及版本信息：apt list --upgradeable
  - 升级软件包，升级前先删除需要更新软件包：sudo apt full-upgrade
  - 安装指定的软件命令：sudo apt install <package_name>
  - 安装多个软件包：sudo apt install <package_1> <package_2> <package_3>
  - 更新指定的软件命令：sudo apt update <package_name>
  - 显示软件包具体信息,例如：版本号，安装大小，依赖关系等等：sudo apt show <package_name>
  - 删除软件包命令：sudo apt remove <package_name>
  - 清理不再使用的依赖和库文件: sudo apt autoremove
  - 移除软件包及配置文件: sudo apt purge <package_name>
  - 查找软件包命令： sudo apt search <keyword>
  - 列出所有已安装的包：apt list --installed
  - 列出所有已安装的包的版本信息：apt list --all-versions
- 例如我们安装 NX 编辑器

```
sudo apt install NX
```

ifconfig

- 用于在不带任何参数（即）ifconfig 运行时显示当前系统上接口的网络配置详细信息。
- 用 SSH 连接时可以通过 ifconfig 查找 IP 地址，终端输入

```
ifconfig
```

- 查看有线网络 IP 地址，终端输入

```
ifconfig  eth0
```

- 查看无线网络 IP 地址，终端输入

```
ifconfig  wlan0
```

3.2 vim 编辑器的使用

Vim 编辑器是所有 Unix 及 Linux 系统下标准的编辑器，它的强大不逊色于任何最新的文本编辑器，这里只是简单地介绍一下它的用法和常用命令。

基本上 vim 共分为三种模式，分别是命令模式（Command mode），输入模式（Insert mode）和底线命令模式（Last line mode）。

- 命令模式：控制屏幕光标的移动，字符、字或行的删除，移动复制某区段;
- 输入模式：在此模式下输入字符，编辑文件;
- 底线模式：将文件保存或退出 vim，也可以设置编辑环境，如寻找字符串、列出行号等;
- 我们可以将这三个模式想成底下的图标来表示;

1.首先删除默认 Vi 编辑器

```
sudo apt-get remove vim-common
```

2.然后重装 Vim

```
sudo apt-get install vim
```

3.为方便使用还得在/etc/vim/vimrc 文件后面添加下面三句

```
set nu   #显示行号
syntax on   #语法高亮
set tabstop=4   #tab 退四格
```

4.常用命令

打开文件、保存、关闭文件(vi 命令模式下使用)

```
vim filename       //打开 filename 文件
    :w                    //保存文件
    :q                    //退出编辑器，如果文件已修改请使用下面的命令
    :q!                   //退出编辑器，且不保存
    :wq                   //退出编辑器，且保存文件
    :wq!                  //强制退出编辑器，且保存文件
```

```
ZZ                    //退出编辑器，且保存文件
ZQ                    //退出编辑器，且不保存
```

插入文本或行(vi 命令模式下使用，执行下面命令后将进入插入模式，按 ESC 键可退出插入模式)

```
a          //在当前光标位置的右边添加文本
i          //在当前光标位置的左边添加文本
A          //在当前行的末尾位置添加文本
I          //在当前行的开始处添加文本(非空字符的行首)
O          //在当前行的上面新建一行
o          //在当前行的下面新建一行
R          //替换(覆盖)当前光标位置及后面的若干文本
J          //合并光标所在行及下一行为一行(依然在命令模式)
```

删除、恢复字符或行(vi 命令模式下使用)

```
x          //删除当前字符
nx         //删除从光标开始的 n 个字符
dd         //删除当前行
ndd        //向下删除当前行在内的 n 行
u          //撤销上一步操作
U          //撤销对当前行的所有操作
```

复制、粘贴(vi 命令模式下使用)

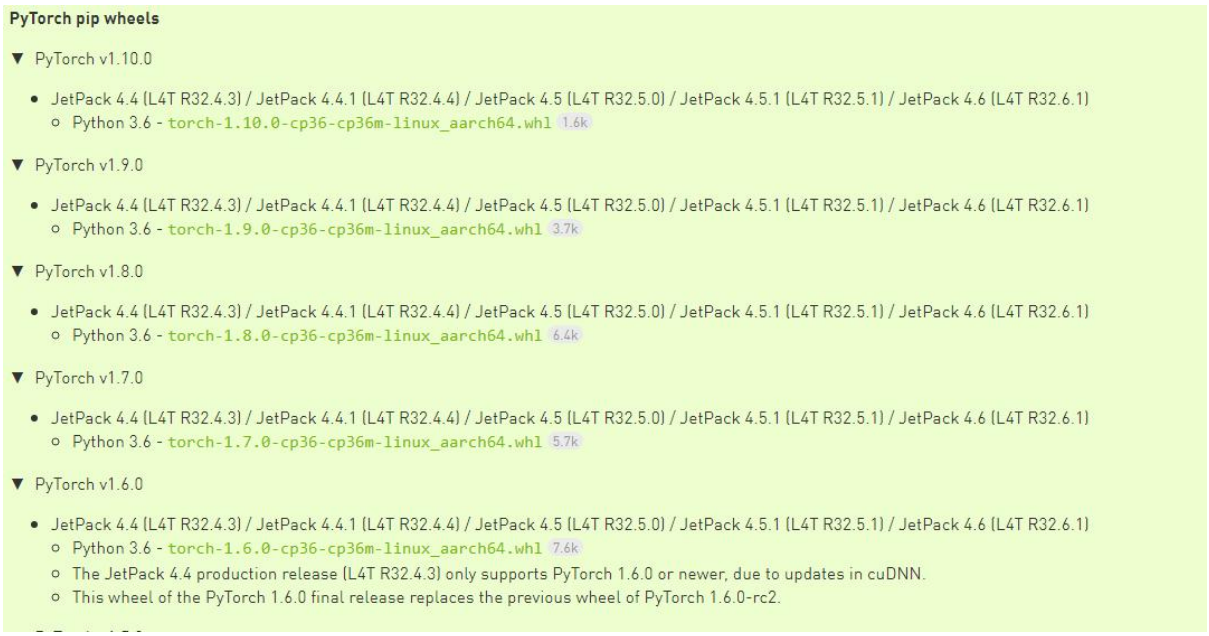
```
yy         //将当前行复制到缓存区
nyy        //将当前行向下 n 行复制到缓冲区
yw         //复制从光标开始到词尾的字符
nyw        //复制从光标开始的 n 个单词
y^         //复制从光标到行首的内容
y$         //复制从光标到行尾的内容
```

```
p       //粘贴剪切板里的内容在光标后

P       //粘贴剪切板里的内容在光标前
```

4. 基础算法框架安装

4.1 Pytorch 安装

Pytorch 安装跟系统版本紧密联系,安装 pythoch 不同版本之前,先确定设备的 jetpack 版本号,根据下面图片来确定需要安装什么版本的 pythrch 才能正常使用.



基于 python3.8 安装方式(以下方式为 1.8.0 为例):

```
wget https://nvidia.box.com/shared/static/p57jwntv436lfrd78inwl7iml6p13fzh.whl -O
torch-1.8.0-cp36-cp36m-linux_aarch64.whl
sudo apt-get install python3-pip libopenblas-base libopenmpi-dev
pip3 install Cython
pip3 install numpy torch-1.8.0-cp36-cp36m-linux_aarch64.whl
```

4.2 Torchvision 安装

Torchvision 安装跟 pytorch 版本紧密联系,安装 torchvision 之前需要核对 pytorch 版本,根据下面图片来确定需要安装什么版本的 torchvision 才能正常使用.

```
sudo apt-get install libjpeg-dev zlib1g-dev libpython3-dev libavcodec-dev libavformat-dev
libswscale-dev

git clone --branch v0.x.0 https://github.com/pytorch/vision torchvision

cd torchvision

export BUILD_VERSION=0.x.0 # (填写 torchvision 安装的版本号)

python3 setup.py install
```

验证:

```
python3
>>> import torch
>>> print(torch.__version__)
>>> print('CUDA available: ' + str(torch.cuda.is_available()))
>>> print('cuDNN version: ' + str(torch.backends.cudnn.version()))
>>> a = torch.cuda.FloatTensor(2).zero_()
>>> print('Tensor a = ' + str(a))
>>> b = torch.randn(2).cuda()
>>> print('Tensor b = ' + str(b))
>>> c = a + b>>> print('Tensor c = ' + str(c))
>>> import torchvision
>>> print(torchvision.__version__)
```

Pytorch whl 包下载地址:
链接：https://pan.baidu.com/s/1J2DeI9HMt0MkH-ZgGfsFnA
提取码：wlf1

4.3 Yolo v4 环境搭建

1.首先在 github 上下载 darknet

```
git clone https://github.com/AlexeyAB/darknet.git
```

2.下载完成后，需要修改下 Makefile 文件

```
cd darknet

sudo vim Makefile
```

3.将前四行 0 改成 1

```
GPU=1

CUDNN=1

CUDNN_HALF=1

OPENCV=1
```

4.cuda 版本和路径也要改成我们的实际版本和路径，否则会编译失败

```
将 NVCC=nvcc 修改为

NVCC=/usr/local/cuda-10.2/bin/nvcc
```

5.修改完成后进行编译，在终端输入

```
sudo make
```

6.使用 YOLOv4 进行推理

(1)基本的推理方法有三种：图片、视频、摄影头（ 实时影像 ）

(2)选择 Yolo v4 和 Yolo v4-tiny（更轻量化的模型，适合在 Jetson NX 上运行）进行测试。首先需要下载已训练好的模型权重文件

图片测试

```
./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/dog.jpg
```

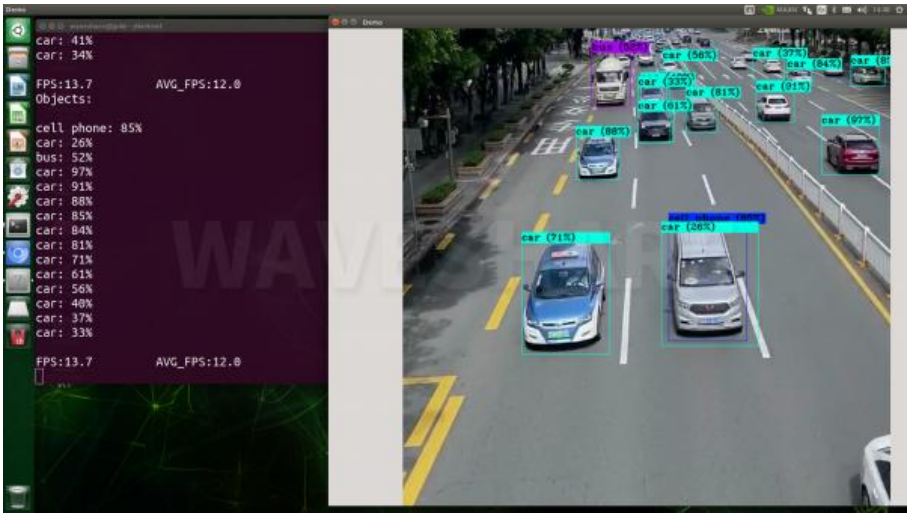
如果要开启图片的话需使用 test 模式，他会在执行之后要你输入图片的位置

```
./darknet detector test ./cfg/coco.data ./cfg/yolov4.cfg ./yolov4.weights
```

视频测试

Yolov4-tiny 视频的检测(github 下来的 data 里面并没有该视频文件，需要用户自行上传要检测的视频文件到 data 文件夹下)

```
./darknet detector demo cfg/coco.data cfg/yolov4-tiny.cfg yolov4-tiny.weights data/xxx.mp4
```



开发资料下载链接：

链接：https://pan.baidu.com/s/1iMEFFJu9sjLpXn6Xq1ozzw?pwd=mmkc