

RK1808 加速棒操作指南

发布版本:1.0.0

日期:2019.06

免责声明

本文档按“现状”提供，福州瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2019 福州瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

福州瑞芯微电子股份有限公司

Fuzhou Rockchip Electronics Co., Ltd.

地址：福建省福州市铜盘路软件园 A 区 18 号

网址：www.rock-chips.com

客户服务电话：+86-591-83991906

客户服务传真：+86-591-83951833

客户服务邮箱：fae@rock-chips.com

前言

概述

本文档是 RK1808 加速棒的操作指南。其中，RK1808 是从 DDR 启动，不同于常规的 eMMC 启动方式。

产品版本

芯片名称	内核版本
RK1808	4.4.1

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

日期	版本	作者	修改说明
2019-06-28	V1.0.0	LJH	

目录

1	RK1808 固件编译与烧写.....	4
1.1	SDK 获取	4
1.2	配置编译环境	4
1.3	编译 uboot	4
1.4	编译 kernel	4
1.5	编译 ramboot	4
1.6	从 RK3399 下载固件到 RK1808	5
1.7	从 X86 下载固件到 RK1808	7
2	RKNPUTool	10
2.1	下载	10
2.2	编译	10
3	Demo 使用说明	10
3.1	RK3399 使用 RK1808 加速棒	11
3.1.1	使用步骤	11
3.1.2	错误排查	12
3.2	X86 使用 RK1808 加速棒	13
3.2.1	使用步骤	13

1 RK1808 固件编译与烧写

1.1 SDK 获取

RK1808_LINUX_SDK 下载命令如下：

```
repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u  
ssh://git@www.rockchip.com.cn/linux/rk/platform/manifests -b linux -m  
rk1808_linux_release.xml
```

关于 SDK 的详细描述请参阅 RK1808_Linux_V1.0.0_20181227 发布说明。

1.2 配置编译环境

```
1808/sdk$ ./build.sh BoardConfig_rk1808_compute_stick.mk
```

注意：BoardConfig_rk1808_compute_stick.mk 文件中的 RK_KERNEL_DTS 要与实际使用的 RK1808 开发板对应，本文档以 RK1808 EVB 的板子做验证，dts 选用 rk1808-evb-v10。

```
--- a/rk1808/BoardConfig_rk1808_compute_stick.mk  
+++ b/rk1808/BoardConfig_rk1808_compute_stick.mk  
@@ -7,7 +7,7 @@ export RK_UBOOT_DEFCONFIG=rknpu-lion  
# Kernel defconfig  
export RK_KERNEL_DEFCONFIG=rk1808_linux_defconfig  
# Kernel dts  
-export RK_KERNEL_DTS=rk1808-compute-v10  
+export RK_KERNEL_DTS=rk1808-evb-v10  
# boot image type  
export RK_BOOT_IMG=boot.img  
# kernel image path
```

1.3 编译 uboot

```
1808/sdk$ ./build.sh uboot
```

1.4 编译 kernel

```
1808/sdk$ ./build.sh kernel
```

1.5 编译 ramboot

```
1808/sdk$ ./build.sh ramboot
```

1.6 从 RK3399 下载固件到 RK1808

- 1) 进入 `device/rockchip/rk1808/rk1808_compute_stick_tool/` 目录并执行 `./copy_rk1808_imgs.sh`，将上面步骤生成的固件拷贝到该目录下。
- 2) 将 `rk1808_compute_stick_tool` 目录拷贝到 RK3399 的 `tmp` 目录下并将 `npu_upgrade`、`aarch64_release/upgrade_tool`、`x64_release/upgrade_tool` 赋予可执行权限。
- 3) 将 RK3399 开发板与 RK1808 开发板通过 USB 相连。



4) 先按住 RK1808 开发板的 MASKROM 键后，再同时按下 RESET 键，经过 2S 后松开，使其进入 MASKROM 烧写模式（RK1808 从 DDR 启动，必须在 MASKROM 模式下烧写）。

5) 在 RK3399 的/tmp/rk1808_compute_stick_tool 目录下，执行命令下载固件到 RK1808。

```
sudo ./npu_upgrade MiniLoaderAll.bin uboot.img trust.img boot.img
```

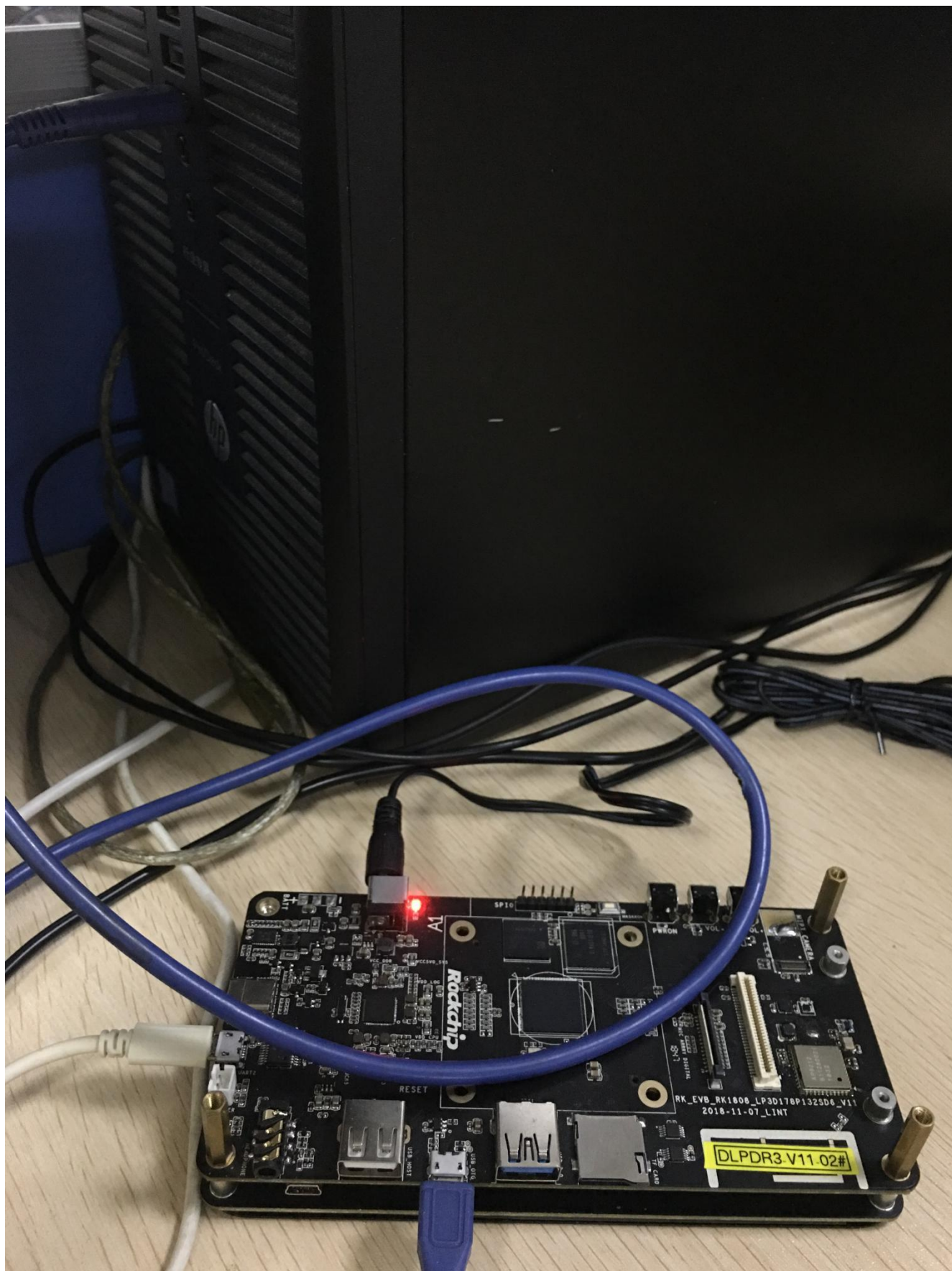
```
linaro@linaro-alip:/tmp/rk1808_compute_stick_tool$ sudo ./npu_upgrade MiniLoaderAll.bin uboot.img tr
ust.img boot.img
use arm version
start to wait device...
device is ready
start to download loader...
download loader ok
start to wait loader...
loader is ready
start to write uboot...
write uboot ok
start to write trust...
write trust ok
start to write boot...
write boot ok
start to run system...
run system ok
```

1.7 从 X86 下载固件到 RK1808

1) 进入 device/rockchip/rk1808/rk1808_compute_stick_tool/ 目录并执行 ./copy_rk1808_imgs.sh, 将上面步骤生成的固件拷贝到该目录下。

2) 将 rk1808_compute_stick_tool 目录拷贝到 X86 的 tmp 目录下并将 npu_upgrade、arch64_release/upgrade_tool、x64_release/upgrade_tool 赋予可执行权限。

3) 将 X86 与 RK1808 开发板通过 USB 相连。



4) 先按住 RK1808 开发板的 MASKROM 键后，再同时按下 RESET 键，经过 2S 后松开，使其进入 MASKROM 烧写模式（RK1808 从 DDR 启动，必须在 MASKROM 模式下烧写）。

5) 在 X86 的/tmp/rk1808_compute_stick_tool 目录下，执行命令下载固件到 RK1808。

```
sudo ./npu_upgrade MiniLoaderAll.bin uboot.img trust.img boot.img
```

```
linaro@linaro-alip:/tmp/rk1808_compute_stick_tool$ sudo ./npu_upgrade MiniLoaderAll.bin uboot.img tr
ust.img boot.img
use arm version
start to wait device...
device is ready
start to download loader...
download loader ok
start to wait loader...
loader is ready
start to write uboot...
write uboot ok
start to write trust...
write trust ok
start to write boot...
write boot ok
start to run system...
run system ok
```

2 RKNPUTool

2.1 下载

在 RK1808 SDK 根目录下，执行 `.repo/repo/repo sync` 同步最新代码，RKNPUTool 在 `sdk/external/` 目录下。

2.2 编译

进入 `device/rockchip/rk1808/rk1808_compute_stick_tool/rknputools` 目录，如果上位机是 ARM 执行：
`./make_linux_aarch64_tools.sh`，X86 执行：`./make_linux_x86_tools.sh`，执行成功会把可执行程序、资源与库文件拷贝到 `build` 目录下。

```
ljh@SYS3:~/1808/sdk/device/rockchip/rk1808/rk1808_compute_stick_tool/rknputools$ tree build/
build/
├── box_priors.txt
├── coco_labels_list.txt
├── dog.jpg
├── labels.txt
├── lib64
│   ├── libopencv_core.so
│   ├── libopencv_core.so.3.4
│   ├── libopencv_core.so.3.4.0
│   ├── libopencv_highgui.so
│   ├── libopencv_highgui.so.3.4
│   ├── libopencv_highgui.so.3.4.0
│   ├── libopencv_imgcodecs.so
│   ├── libopencv_imgcodecs.so.3.4
│   ├── libopencv_imgcodecs.so.3.4.0
│   ├── libopencv_imgproc.so
│   ├── libopencv_imgproc.so.3.4
│   ├── libopencv_imgproc.so.3.4.0
│   └── librknn_api.so
├── mobilenet_ssd.rknn
├── mobilenet_v1-tf.rknn
├── npu_transfer_proxy
├── rknn_mobilenet
├── rknn_ssd
└── road.bmp
```

3 Demo 使用说明

API SDK 的 Linux 目录下提供了两个使用 RKNN API 的 Demo，一个是基于 MobileNet 模型图像分类器 Demo，另一个是基于 MobileNet-SSD 模型的目标检测 Demo。目前该 Demo 只适用于 64 位的 RK3399 和 X86 Linux 系统，同时只提供 64 位的 `rknn_api` 库。

3.1 RK3399 使用 RK1808 加速棒

3.1.1 使用步骤

1) 将 device/rockchip/rk1808/rk1808_compute_stick_tool/rknpertools 目录, 拷贝到 RK3399 的/tmp/目录下, 然后执行 linux.sh。

2) 在 RK3399 上以 root 权限执行 npu_transfer_proxy。

```
root@linaro-alip:/tmp# ./npu_transfer_proxy
I NPUTransfer: Starting NPU Transfer Proxy, Transfer version 1.9.2 (126ddbdf@2019-05-20T09:15:08), devid = 0123456789ABCDEF, pid = 1043:652
```

3) 将 RK3399 与 RK1808 通过 USB 线相连。

4) 在 RK3399 上新开一个窗口运行 rknn_mobilenet, 成功会有如下打印:

```
linaro@linaro-alip:/tmp$ ./rknn_mobilenet
spec = local:transfer_proxy
D RKNNAPI:
D RKNNAPI: RKNN VERSION:
D RKNNAPI:   API: 0.9.5 (a949908 build: 2019-05-07 22:20:43)
D RKNNAPI:   DRV: 0.9.6 (c12de8a build: 2019-05-06 20:10:17)
D RKNNAPI:
0.890625: 156 Shih-Tzu
0.061615: 155 Pekinese
0.00874329: 205 Lhasa
0.00789642: 188 Yorkshire terrier
0.00579834: 263 Brabancon griffon
perf_run.run_duration = 13535 us
perf_run.perf_data =
Layer id:   Name:   Operation id:   Operator:   Target:   Time(us):
0   ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   523
1   ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   402
2   ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   373
3   ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   465
4   ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   305
5   ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   323
6   ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   328
7   ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   351
8   ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   274
9   ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   304
10  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   350
11  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   313
12  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   320
13  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   371
14  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   369
15  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   367
16  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   370
17  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   397
18  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   431
19  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   373
20  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   371
21  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   369
22  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   376
23  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   406
24  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   300
25  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   464
26  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   331
27  PoolingLayer2   0   POOLING   SH   335
28  ConvolutionReluPoolingLayer2   0   CONVOLUTION   NN   338
29  TensorTranspose   0   TENSOR_TRANS   TP   123
30  SoftMax2   0   SOFTMAX   SH   418
```

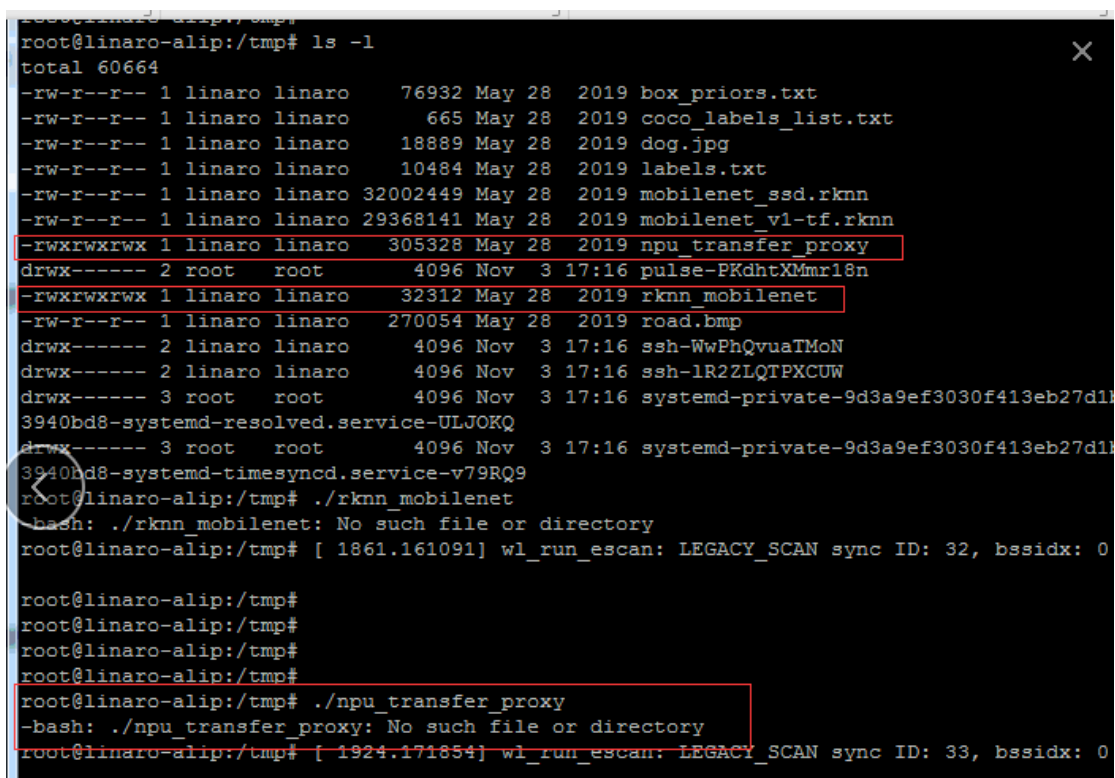
5) 执行 rknn_ssd 成功会有如下打印, 同时还会在 RK3399 的/tmp 目录下生成包含检测结果的图像 out.jpg, 可以导出 out.jpg 查看检测结果。

```
linaro@linaro-alip:/tmp$ ./rknn_ssd
D RKNNAPI: =====
D RKNNAPI: RKNN VERSION:
D RKNNAPI:   API: 0.9.7 (ad412e9 build: 2019-06-14 09:58:20)
D RKNNAPI:   DRV: 0.9.7 (d65a37f build: 2019-05-31 14:23:01)
D RKNNAPI: =====
validCount: 26
person @ (62, 503) (234, 843)
person @ (198, 535) (230, 634)
person @ (321, 531) (361, 626)
person @ (414, 468) (584, 764)
car @ (576, 519) (863, 689)
person @ (834, 468) (1037, 869)
bicycle @ (697, 637) (1110, 938)
write out.jpg succ!
linaro@linaro-alip:/tmp$
```

注意：本文档是基于 RK_EXCAVATOR_MAIN_V12 (RK3399) 开发板，运行 64 位 Debian 系统验证（固件下载地址：<https://eyun.baidu.com/s/3smEf8xn>）。

3.1.2 错误排查

1) 32 位 Debian 运行程序错误信息如下：



```
root@linaro-alip:/tmp# ls -l
total 60664
-rw-r--r-- 1 linaro linaro 76932 May 28 2019 box_priors.txt
-rw-r--r-- 1 linaro linaro 665 May 28 2019 coco_labels_list.txt
-rw-r--r-- 1 linaro linaro 18889 May 28 2019 dog.jpg
-rw-r--r-- 1 linaro linaro 10484 May 28 2019 labels.txt
-rw-r--r-- 1 linaro linaro 32002449 May 28 2019 mobilenet_ssd.rknn
-rw-r--r-- 1 linaro linaro 29368141 May 28 2019 mobilenet_v1-tf.rknn
-rwxrwxrwx 1 linaro linaro 305328 May 28 2019 npu_transfer_proxy
drwx----- 2 root root 4096 Nov 3 17:16 pulse-PKdhtXMmr18n
-rwxrwxrwx 1 linaro linaro 32312 May 28 2019 rknn_mobilenet
-rw-r--r-- 1 linaro linaro 270054 May 28 2019 road.bmp
drwx----- 2 linaro linaro 4096 Nov 3 17:16 ssh-WwPhQvuaTMoN
drwx----- 2 linaro linaro 4096 Nov 3 17:16 ssh-1R2ZLQTPXCW
drwx----- 3 root root 4096 Nov 3 17:16 systemd-private-9d3a9ef3030f413eb27d11
3940bd8-systemd-resolved.service-ULJOKQ
drwx----- 3 root root 4096 Nov 3 17:16 systemd-private-9d3a9ef3030f413eb27d11
3940bd8-systemd-timesyncd.service-v79RQ9
root@linaro-alip:/tmp# ./rknn_mobilenet
-bash: ./rknn_mobilenet: No such file or directory
root@linaro-alip:/tmp# [ 1861.161091] wl_run_escan: LEGACY_SCAN sync ID: 32, bssidx: 0
root@linaro-alip:/tmp#
root@linaro-alip:/tmp#
root@linaro-alip:/tmp#
root@linaro-alip:/tmp#
root@linaro-alip:/tmp# ./npu_transfer_proxy
-bash: ./npu_transfer_proxy: No such file or directory
root@linaro-alip:/tmp# [ 1924.171854] wl_run_escan: LEGACY_SCAN sync ID: 33, bssidx: 0
```

2) USB 通信异常 错误信息如下：


```

-rw-r--r-- 1 linaro linaro 32002449 May 28 18:43 mob
-rw-r--r-- 1 linaro linaro 29368141 May 28 18:43 mob
-rw-r--r-- 1 linaro linaro 305328 May 28 20:04 npu
-rw-r--r-- 1 linaro linaro 32312 May 28 19:23 rknn
-rw-r--r-- 1 linaro linaro 270054 May 28 18:43 rknn
drwxr-xr-x 3 linaro linaro 4096 May 29 2019 usr
linaro@linaro-alip:/media/linaro/NO NAME/tmp$ cp -dp
linaro@linaro-alip:/media/linaro/NO NAME/tmp$ cd /tmp
linaro@linaro-alip:/tmp$ ./rknn_mobilenet
spec = local:transfer proxy
E NPUTransfer: Transfer interface open failed!, ret
E RKNNAPI: rknn_init, driver open fail! ret = -1!
rknn_init fail! ret=-3
linaro@linaro-alip:/tmp$

```

先确认 USB 有没有连接好，在 RK3399 上用 `lsusb` 查看。再确认 RK1808 编译的 Buildroot 是否有打开 `BR2_PACKAGE_RKNPU NTB=y`

3.2 X86 使用 RK1808 加速棒

3.2.1 使用步骤

- 1) 将 `device/rockchip/rk1808/rk1808_compute_stick_tool/rknpertools` 目录，拷贝到 X86 的 `/tmp/` 目录下，然后执行 `linux.sh`。
- 2) 在 X86 上以 root 权限执行 `npu_transfer_proxy`。
- 3) 将 X86 与 RK1808 通过 USB 线相连。
- 4) 在 X86 上新开一个窗口先 `export LD_LIBRARY_PATH=/tmp`，然后执行 `rknn_mobilenet`，执行成功后会有执行时间和检测结果的打印。执行 `rknn_ssd`，执行成功后会有执行时间和检测结果的打印，同时还会在 `/tmp` 目录下生成包含检测结果的图像 `out.jpg` 可以打开 `out.jpg` 查看检测结果。

注：该 Demo 在 Ubuntu16.04 64 位系统上验证通过，运行 demo 成功的打印信息如 RK3399 平台。