> 目标：在飞凌ls1046上安装ros，替代TX1控制器，进行激光slam的测试
>
> 当前TX1环境：
>
> Operating System: Ubuntu 18.04.5 LTS
> Kernel: Linux 4.9.201-tegra
> Architecture: arm64
>
> 当前AM3354环境：
>
> Operating System: Buildroot 2017.08
> Kernel: Linux 4.9.28-geed43d1050
> Architecture:arm

# 一.飞凌ls1046

环境

> 系统：
>
> Operating System: Ubuntu 18.04.1 LTS
> Kernel: Linux 4.14.47
> Architecture: arm64
>
> 网络：
>
> 开发板默认打开 fm1-mac3 网口(P13 上，即是偏右的最上边)，默认 IP 为 192.168.0.232
>
> runlevel：
> N 5
>
> 虽然运行级别为5，但是根据安装的环境及安装包判断是没有界面的

设置IP

原先系统自带的配置静态IP `192.168.0.232` 在 `/etc/systemd/network` 目录下的 `fm1-mac3.network`

# 0821

调研当前架构及系统能否安装ROS

安装ROS

测试功能包

目前的ubuntu系统比较完整，很多指令都有，如果ROS测试没有问题，证明该板卡的系统至少能运行基本的功能包程序与底层通信

目前遇到的问题，主要是网络，无线网卡模块没有选配、有线网卡没有路由器能够链接、5G模块需要的是中卡套、usbwifi模块识别不到（可能是驱动安装问题）

# 0822

目前已安装了ROS并进行测试，基本指令能够使用，但是emmc容量不足，当前安装base版本需要0.7G，安装全桌面版需要至少2.3G，以及当前飞凌的板卡出厂自带的驱动过少，无论是对wifi模块的支持还是硬盘格式的支持都很少，针对容量不足外接固态硬盘，但是在飞凌的板卡上无法是被exfat格式，又需要联网安装相关驱动，当前板卡的网络也成问题：四种方案

1.有线网卡：在公司没有能提供上网的路由

2.无线网卡：模块没有安装，而且当前购买的模块并不是飞凌支持的三个wifi模块的一种

3.5G模块：缺少卡套

4.usbwifi：lsusb识别到，但是缺少驱动，需要编译，但是飞凌板卡上的/lib的链接文件时制卡时的，在飞凌烧录的系统上是找不到该目录的

# 0823

关于第四点：

已经尝试了在虚拟机上编译，流程如下：

找到对应的源码编译

```
make
```

安装

```
make install
```

查看并加载依赖

```
lsmod | grep cfg80211
modprobe cfg80211
```

mark

参考手册，/home/forlinx/work/OK10xx-linux-fs/flexbuild/packages/linux位置是编译安装驱动的目录，目前有cryptodev-linux

测试重新编译生成image测试的时候就以有无新的usb无线wif模块驱动88x2bu为成功标志

```
export ARCH=arm64
export CROSS_COMPILE=aarch64-linux-gnu-
export PATH=$PATH:/home/forlinx/work/OK10xx-linux-
fs/flexbuild/build/rfs/OK10xx-linux-
ubuntu/rootfs_ubuntu_bionic_arm64/usr/bin
```

## 0824

目前完整的编译了一遍生成所有的镜像

```
flex-builder -a arm64 -m ls1046ardb -S 1133
```

而且找到了生成的模块驱动位置以及源码位置，目前在该路径下有同厂家的驱动源码

```
/home/forlinx/work/OK10xx-linux-fs/flexbuild/packages/linux/OK10xx-
linux-kernel/drivers/net/wireless/realtek/rtl88x2ce
```

在realtek目录下，添加rtl8822bu的源码，然后编译生成ko模块后，使用以下指令自动将驱动模块更新到文件系统

```
root@ubuntu:~/work/OK10xx-linux-fs/flexbuild# flex-builder -i merge-
component -a arm64 -m
ls1046ardb
```

最后编译成功的日志输出

```
 /home/forlinx/work/OK10xx-linux-fs/flexbuild/build/images/ubuntu.img
  [Done]
 arm64: Build ubuntu userland and apps components in
/home/forlinx/work/OK10xx-linux-fs/flexbuild/build/images!     [Done]

/home/forlinx/work/OK10xx-linux-fs/flexbuild/build/images/usb_update.itb
    [Done]
INSTRUCTION: genboot
MACHINE: ls1046ardb
gening /home/forlinx/work/OK10xx-linux-fs/flexbuild/build/images/boot,
waiting ...
 /home/forlinx/work/OK10xx-linux-fs/flexbuild/build/images/boot
[Done]
 Time of Build Done: Thu Aug 24 01:35:28 PDT 2023
 arm64 Autobuild Time: 42 Mins 18 Secs !
```

驱动模块链接报错

```
/home/forlinx/work/OK10xx-linux-fs/flexbuild/packages/linux/OK10xx-
linux-kernel/drivers/net/wireless/realtek/rtl8822bu/core/rtw_roch.c:224:
undefined reference to `cfg80211_remain_on_channel_expired'
```

mini车 4.9内核版本对比编译链接

结果；

同样代码编译成功，目前同样的代码已经能在gree、forlinx、wheeltec-mini-car三个系统上都已经编译链接成功了，这样看来应该是cfg80211的链接问题

目前在提供的linux源码中存在同款型号8822ce，尝试拷贝除了hal库的文件夹之外的所有文件替换编译链接

交叉编译时使用的目录

```
Using /home/forlinx/work/OK10xx-linux-
fs/flexbuild/packages/linux/OK10xx-linux-kernel as source for kernel
```

尝试解决方案

把`/home/forlinx/work/OK10xx-linux-fs/flexbuild/build/linux/linux/arm64/output/`

复制到板卡上的类似位姿`/usr/src/linux-headers-4.15.0-29`，把`/lib/modules/4.15.0-29-generic`目录下的build链接到`/usr/src/linux-headers-4.15.0-29`。然后在板卡上编译

## 0829-0830

5G模块使用成功，飞凌的手册写的太粗略了，昨天获取到的ip还是不能上网，是因为网关设置问题，如果开发板同时插上了网线，因为此系统默认会只让一个网卡连外网，设置网关为 5G 模块的路由规则。（来源于正点原子手册4G模块上网部分）

> 因为本人已经插上网线，上网会优先选择 eth0/eth1。如果用户没有插网线，就不需要添加路由表啦，因为你的网卡只有 4G 网卡上网，系统就会选择 4G 网卡上网。
>
> 我们需要先添加 4G 模块的网关地址，然后删除默认的 eth0 的网关地址，再使用 route 指令

拨号上网命令

```
/root/Net_Tools/quectel-CM >> /dev/null &
```

查看添加是否成功

```
route add default gw 10.0.47.1
route del default gw 192.168.0.1
route
```

通过`ping www.baidu.com`来测试是否能上网。-I 参数是指定 usb0(5G 网络)，按“Ctrl +c”结束 ping。

```
ping www.baidu.com -I usb0
```

目前的系统里只支持FAT32格式，即是32G以下的U盘，只有32G以下的才能被格式化为FAT32，超过32G格式为exfat，当前系统不支持，需要安装驱动

```
sudo apt install exfat-fuse exfat-utils
```

然后挂载

```
mount -t exfat /dev/sda1 /mnt
```

1. 设置**APT**源： 打开终端并输入以下命令，以添加ROS的APT源到你的系统中：

   ```
   sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
   ```

2. 设置密钥： 继续在终端中输入以下命令，以添加ROS的密钥到你的系统中：

   ```
   sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
   ```

3. 更新软件包列表：

今天的测试，ros-melodic-desktop-full版本已经完成安装，并且成功测试与别的ros系统通信，查看雷达及相机数据。

## 需求分析

在编译功能包时，要跑起来最小的激光slam系统，至少需要以下功能包

- turn_on_gree_robot（底层驱动）
- gmapping/cartographer（建图算法）
- navigation（导航stack）
  - map_server（建图保存）
  - amcl（导航定位）
  - costmap_2d（代价地图）
  - global_planner（全局路径规划）
  - move_base（控制核心）
  - dwa_local_planner/teb_local_planner（局部路径规划）
- lsn10（至少一款雷达驱动）
- robot_rc（键盘控制）
- robot_pose_ekf（扩展卡尔曼滤波）
- teb_local_planner（局部路径规划）

后续可选：

- costmap_prohibition_layer（电子围栏）
- rplidar_ros（思岚雷达）

至少需要的外设驱动如下：

- ch2102驱动（雷达串口线）
- ch343驱动（底层stm32通信串口线）
- usb无线网卡驱动（网络通信）
- 固态硬盘驱动（扩容）

## 问题汇总

根据以上分析，开始编译功能包及驱动，各自出现的问题如下

### turn_on_gree_robot

```
Could NOT find serial (missing: serial_DIR)
-- Could not find the required component 'serial'. The following CMake
error indicates that you either need to install the package with the
same name or change your environment so that it can be found.
CMake Error at /opt/ros/melodic/share/catkin/cmake/catkinConfig.cmake:83
(find_package):
  Could not find a package configuration file provided by "serial" with
any
  of the following names:

    serialConfig.cmake
    serial-config.cmake
```

```
sudo apt install ros-melodic-serial
```

安装serial依赖后编译成功

### gmapping/cartographer

原本应该是ros安装后有自带的功能包，但是ros-melodic-desktop-full却没有，需要查证一下；

```
sudo apt install ros-melodic-slam-gmapping
```

安装gmapping后，启动没有报错，由于雷达串口识别还没调通，暂时没有数据

### navigation

```
+++ processing catkin metapackage: 'navigation'
-- ==> add_subdirectory(navigation-melodic/navigation)
-- +++ processing catkin package: 'map_server'
-- ==> add_subdirectory(navigation-melodic/map_server)
-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy
CMake Error at /usr/share/cmake-
3.10/Modules/FindPackageHandleStandardArgs.cmake:137 (message):
  Could NOT find SDL (missing: SDL_LIBRARY SDL_INCLUDE_DIR)
Call Stack (most recent call first):
  /usr/share/cmake-3.10/Modules/FindPackageHandleStandardArgs.cmake:378
(_FPHSA_FAILURE_MESSAGE)
  /usr/share/cmake-3.10/Modules/FindSDL.cmake:190
(FIND_PACKAGE_HANDLE_STANDARD_ARGS)
  navigation-melodic/map_server/CMakeLists.txt:12 (find_package)
```

编译顺序为：

- voxel_grid
- map_server，需要安装以下依赖

```
sudo apt-get install libsdl-image1.2-dev
sudo apt-get install libsdl1.2-dev
```

解决参考

[ROS编译基本错误*could not find libg2o!*李德龙杰的博客-CSDN博客]

- fake_localization
- costmap_2d，需要安装以下依赖

```
sudo apt-get install ros-melodic-tf2-sensor-msgs
```

- nav_core
- navfn
- base_local_planner
- carrot_planner
- dwa_local_planner
- clear_costmap_recovery

- global_planner
- move_slow_and_clear
- rotate_recovery
- move_base，需要安装以下依赖

```
sudo apt-get install ros-melodic-move-base-msgs
```

- amcl

## lsn10

编译没有问题

## robot_rc

```
+++ processing catkin package: 'wheeltec_robot_rc'
-- ==> add_subdirectory(wheeltec_robot_rc)
-- Could NOT find joy (missing: joy_DIR)
-- Could not find the required component 'joy'. The following CMake
error indicates that you either need to install the package with the
same name or change your environment so that it can be found.
CMake Error at /opt/ros/melodic/share/catkin/cmake/catkinConfig.cmake:83
(find_package):
  Could not find a package configuration file provided by "joy" with any
of
  the following names:

    joyConfig.cmake
    joy-config.cmake
```

```
sudo apt-get install ros-melodic-joy
```

安装joy依赖后编译成功

## robot_pose_ekf

```
 +++ processing catkin package: 'robot_pose_ekf'
-- ==> add_subdirectory(robot_pose_ekf)
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Checking for module 'orocos-bfl'
--   No package 'orocos-bfl' found
CMake Error at /usr/share/cmake-3.10/Modules/FindPkgConfig.cmake:419
(message):
  A required package was not found
Call Stack (most recent call first):
  /usr/share/cmake-3.10/Modules/FindPkgConfig.cmake:597
(_pkg_check_modules_internal)
  robot_pose_ekf/CMakeLists.txt:6 (pkg_check_modules)
```

```
sudo apt-get install ros-melodic-bfl
```

安装bfl依赖后编译成功

### costmap_prohibition_layer

编译没有问题

### teb_local_planner

```
+++ processing catkin package: 'teb_local_planner'
-- ==> add_subdirectory(teb_local_planner-melodic-devel)
-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy
-- Could NOT find costmap_converter (missing: costmap_converter_DIR)
-- Could not find the required component 'costmap_converter'. The
following CMake error indicates that you either need to install the
package with the same name or change your environment so that it can be
found.
CMake Error at /opt/ros/melodic/share/catkin/cmake/catkinConfig.cmake:83
(find_package):
  Could not find a package configuration file provided by
"costmap_converter"
  with any of the following names:
```

```
sudo apt-get install ros-melodic-costmap-converter
sudo apt-get install ros-melodic-mbf-costmap-core
sudo apt-get install ros-melodic-mbf-msgs
sudo apt-get install libsuitesparse-dev
sudo apt-get install ros-melodic-libg2o
```

```
virtual memory exhausted: Cannot allocate memory
virtual memory exhausted: Cannot allocate memory
```

编译时可能会由于内存不足报错，多编译几次即可

### rplidar_ros

编译没有问题

## 0831

烧录文件系统，制作U盘启动，烧录之前先取消挂载，然后fdisk -l查看设备，注意 `/dev/sdb1` 是一个分区，而 `/dev/sdb` 是整个磁盘设备。需要将镜像文件 `ubuntu.img` 写入整个磁盘设备 `/dev/sdb`，而不是 `/dev/sdb1` 分区。这是因为操作系统的引导扇区和分区表位于整个磁盘设备上，而不仅仅是某个分区上。

所以，在执行 `dd` 命令时，正确的设备应该是 `/dev/sdb`，而不是 `/dev/sdb1`。

```
forlinx@ubuntu:~/work/OK10xx-linux-fs/flexbuild/build/images$ sudo dd
if=/home/forlinx/work/OK10xx-linux-fs/flexbuild/build/images/ubuntu.img
of=/dev/sdb bs=4M status=progress
2785017856 bytes (2.8 GB, 2.6 GiB) copied, 416 s, 6.7 MB/s
664+1 records in
664+1 records out
2785631224 bytes (2.8 GB, 2.6 GiB) copied, 611.438 s, 4.6 MB/s
664+1 records in
664+1 records out
2785631224 bytes (2.8 GB, 2.6 GiB) copied, 611.438 s, 4.6 MB/s
```

**Linux dd烧写系统 - kumata - 博客园 (cnblogs.com)**

这个错误消息表明 ext2/ext3/ext4 文件系统的超级块损坏或无法读取

```
ext2fs_open2: Bad magic number in super-block
fsck.ext2: Superblock invalid, trying backup blocks...
fsck.ext2: Bad magic number in super-block while trying to open
/dev/sdb1

The superblock could not be read or does not describe a valid
ext2/ext3/ext4
filesystem.  If the device is valid and it really contains an
ext2/ext3/ext4
filesystem (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate
superblock:
    e2fsck -b 8193 <device>
 or
    e2fsck -b 32768 <device>
```

# 0901

```
root@ubuntu:/home/forlinx/work/OK10xx-linux-fs/flexbuild/build/images#
file ubuntu.img
ubuntu.img: Android sparse image, version: 1.0, Total of 1828608 4096-
byte output blocks in 56573 input chunks.
root@ubuntu:/home/forlinx/work/OK10xx-linux-fs/flexbuild/build/images#
fdisk -l ubuntu.img
Disk ubuntu.img: 2.6 GiB, 2785630720 bytes, 5440685 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
ubuntu.img: Android sparse image, version: 1.0, Total of 1828608 4096-byte output blocks in 56573 input chunks
根据file命令的输出，ubuntu.img镜像文件被标识为Android稀疏映像，版本为1.0。它包含1828608个4096字节的输出块，由56573个输入块组成。

这意味着ubuntu.img镜像文件是一个经过稀疏处理的Android映像文件，而不是常规的分区表、引导记录和文件系统的镜像文件。稀疏映像是一种优化技术，用于在存储空间上节省映像文件的大小。

因此，对于这个特定的镜像文件，它不包含常规的分区表、引导记录和文件系统。如果需要查看或修改其中的内容，可能需要使用适用于Android稀疏映像的工具进行处理。
```

要查看Android稀疏映像文件（`ubuntu.img`）中的内容，可以使用`simg2img`工具将其转换为常规的镜像文件格式，然后使用适当的工具进行挂载或浏览。

以下是一般的步骤：

1. 首先，安装`simg2img`工具。在Ubuntu上，可以使用以下命令安装：

```
sudo apt-get install android-tools-fsutils
```

1. 使用`simg2img`工具将稀疏映像转换为常规镜像文件。运行以下命令：

```
simg2img ubuntu.img ubuntu.raw
```

这将生成一个名为`ubuntu.raw`的常规镜像文件。

1. 使用`file`命令检查转换后的镜像文件类型：

```
file ubuntu.raw
```

确保它被标识为正常的镜像文件类型，例如"Linux filesystem data"。

1. 使用`mount`命令将转换后的镜像文件挂载到某个目录：

```
sudo mount -o loop ubuntu.raw /mnt
```

这将将镜像文件的内容挂载到`/mnt`目录。

1. 现在，你可以浏览`/mnt`目录以查看镜像文件中的内容：

```
ls /mnt
```

你可以使用 `cd` 命令进入子目录，或者使用其他命令查看和处理文件。

完成后，使用 `umount` 命令卸载镜像文件：

```
sudo umount /mnt
```

已经使用ubuntu.raw烧录到/dev/sdb1中后，挂载能显示文件夹内容，但是

/dev/sdb1的文件系统大小为58.6G，但在挂载到/mnt/rootfs后，只显示为6.8G可用空间。这可能是由于文件系统被调整为较小的大小导致的。

可以尝试使用resize2fs命令来调整文件系统的大小，以便将剩余的容量分配给它。请按照以下步骤操作：

1. 确保已卸载/mnt/rootfs目录：

   ```
   sudo umount /mnt/rootfs
   ```

2. 使用resize2fs命令来调整文件系统的大小：

   ```
   e2fsck -f /dev/sdb1
   sudo resize2fs /dev/sdb1
   ```

   这将自动将文件系统调整为设备的最大可用空间。

3. 重新挂载设备到/mnt/rootfs：

   ```
   sudo mount /dev/sdb1 /mnt/rootfs
   ```

## ch2102驱动

目前连接上雷达lsn10，lsusb能够识别到usb设备，但是没有能生成/dev/tty*设备，缺少驱动，在当前板卡上编译驱动需要去找到/lib/modules/版本号/build，但是目前是交叉编译的系统，ln链接到是在编译主机上，在板卡上无法正常编译。

**NVIDIA Jetson TX2连接usb后无法找到设备*cp210x*安装后仍然无法识别看那片云的博客-CSDN博客**

**CP2102 USB to UART Bridge Controller 驱动安装（windows or Ubuntu）*cp2102usb to uart*合工大机器人实验室的博客-CSDN博客**

### ch343驱动

同上

### usb无线网卡驱动

同上，并且在主机编译的时候，编译没有报错，但是链接到cfg80211的时候失败，无法正常生成ko文件

### 固态硬盘驱动

TODO

## 0904

usb无线网卡驱动

```
make ARCH=arm64 CROSS_COMPILE= -C /lib/modules/4.15.0-20-generic/build
M=/home/forlinx/rtl882bu_arm modules -j
make ARCH=arm64 CROSS_COMPILE= -C /home/forlinx/work/OK10xx-linux-
fs/flexbuild/packages/linux/OK10xx-linux-kernel
M=/home/forlinx/RTL88x2BU_arm modules -j
```

```
make[2]: *** /home/forlinx/work/OK10xx-linux-
fs/flexbuild/packages/linux/OK10xx-linux-kernel: No such file or
directory.  Stop.
make[2]: Leaving directory '/usr/src/output'
Makefile:24: recipe for target '__sub-make' failed
make[1]: *** [__sub-make] Error 2
make[1]: Leaving directory '/usr/src/output'
Makefile:2420: recipe for target 'modules' failed
make: *** [modules] Error 2
```

【Linux】移植USB、CH340驱动到arm板，并作测试_菜老越的博客-CSDN博客

WG217 wifi模块RTL8811CU的移植（linux）_Wilburn0的博客-CSDN博客

## 0905

由于ssd固态硬盘也会识别为/dev/sda，启动烧录的时候可能会去ssd固态硬盘上去找文件，而烧录的镜像在u盘上，因此一直没有烧录成功，现象为红色指定灯一直不闪烁

> ls1046A板卡在烧录镜像的时候，红色led灯会快速闪烁，等待烧录完成后，红色led等依旧不断亮灭，只是没有闪烁那么快的频率

```
usbcore: registered new interface driver ch341
[  117.138471] usbserial: USB Serial support registered for ch341-uart
[  219.707013] usb 3-1: new full-speed USB device number 2 using xhci-
hcd
[  219.875528] usbcore: registered new interface driver cp210x
[  219.875572] usbserial: USB Serial support registered for cp210x
[  219.875654] cp210x 3-1:1.0: cp210x converter detected
[  219.877466] usb 3-1: cp210x converter now attached to ttyUSB5
```

以上为烧录了带有serial、ch341、cp210x等驱动后，dmesg显示的接入雷达串口后识别为 ttyUSB5

## 0906

编译内核的时候，读取使用的配置信息

```
make[2]: Entering directory '/home/forlinx/work/OK10xx-linux-
fs/flexbuild/packages/linux/OK10xx-linux-kernel'
make[3]: Entering directory '/home/forlinx/work/OK10xx-linux-
fs/flexbuild/build/linux/linux/arm64/output'
  GEN     ./Makefile
scripts/kconfig/conf  --silentoldconfig Kconfig
  CHK     include/config/kernel.release
  GEN     ./Makefile
  CHK     include/generated/uapi/linux/version.h
  Using /home/forlinx/work/OK10xx-linux-
fs/flexbuild/packages/linux/OK10xx-linux-kernel as source for kernel
  CHK     include/generated/utsrelease.h
  CHK     include/generated/timeconst.h
  CHK     include/generated/bounds.h
  CHK     include/generated/asm-offsets.h
  CALL    /home/forlinx/work/OK10xx-linux-
fs/flexbuild/packages/linux/OK10xx-linux-kernel/scripts/checksyscalls.sh
  CHK     scripts/mod/devicetable-offsets.h
```

```
  CHK     include/generated/compile.h
  GZIP    kernel/config_data.gz
  CHK     kernel/config_data.h
  UPD     kernel/config_data.h
  CC      kernel/configs.o
  AR      kernel/built-in.o
```

参考ch341所处的目录下的Makefile和Kconfig文件，添加ch343依赖，将ch343.c和ch343.h文件复制到该目录下，然后配置内核驱动ch343为M，重新编译内核，将生成的ch343.ko驱动安装到ls1046A的板卡上，insmod驱动，接入底盘串口线，识别成ttych343usb，之后就可以启动base_serial.launch

同样，配置rt2800usb驱动，将ko文件安装到ls1046A，该驱动是无线网卡usb模块驱动

## 0907

虚拟机上ralink2870无线usb模块使用，在正确的安装驱动之后

要配置 wpa_supplicant 连接 WiFi，需要编辑 wpa_supplicant 的配置文件（通常位于"/etc/wpa_supplicant/wpa_supplicant.conf"）并添加适当的网络配置，没有该文件直接创建即可。

以下是 wpa_supplicant 配置文件的基本结构：

```
ctrl_interface=/run/wpa_supplicant
update_config=1

network={
    ssid="yangfx"
    psk="201016YX86f"
}
```

在上面的示例中，需要替换 "ssid" 和 "psk" 分别为 WiFi 网络的名称和密码。

保存并关闭配置文件后，使用以下命令启动 wpa_supplicant 并连接到 WiFi 网络，注意-i后的wifi名称：

```
sudo wpa_supplicant -B -iwlan0 -c/etc/wpa_supplicant/wpa_supplicant.conf
```

其中，"-B" 选项表示在后台运行 wpa_supplicant，"-iwlan0" 指定要连接的无线网络接口，"-c" 选项后面是 wpa_supplicant 配置文件的路径。

wifi名被rename为wlx7cdd901b2360，暂时没有查出原因，使用如下命令替换

```
sudo wpa_supplicant -B -iwlx7cdd901b2360 -
c/etc/wpa_supplicant/wpa_supplicant.conf
```

如果连接成功，可以使用以下命令检查网络连接状态：

```
sudo wpa_cli status
```

然后动态获取ip

```
sudo udhcpc -i wlx7cdd901b2360
```

注意根据实际需求设置网关（可选），把不能上网的网卡网关删除，否则动态获取了ip也无法ping通

```
sudo route del default gw 192.168.0.0
```

## 0908

```
usb 3-1: new high-speed USB device number 2 using xhci-hcd
[   86.951012] usb 3-1: reset high-speed USB device number 2 using xhci-
hcd
[   87.109401] ieee80211 phy0: rt2x00_set_rt: Info - RT chipset 3070,
rev 0201 detected
[   87.118576] ieee80211 phy0: rt2x00_set_rf: Info - RF chipset 0005
detected
[   87.119763] ieee80211 phy0: Selected rate control algorithm
'minstrel_ht'
[   87.120609] usbcore: registered new interface driver rt2800usb
[   87.139118] rt2800usb 3-1:1.0 wlx7cdd901b2360: renamed from wlan0
[   87.206175] ieee80211 phy0: rt2x00lib_request_firmware: Info -
Loading firmware file 'rt2870.bin'
[   87.206591] rt2800usb 3-1:1.0: Direct firmware load for rt2870.bin
failed with error -2
[   87.206596] ieee80211 phy0: rt2x00lib_request_firmware: Error -
Failed to request Firmware
```

```
rt2800usb 3-1:1.0 wlx7cdd901b2360: renamed from wlan0
[   87.206175] ieee80211 phy0: rt2x00lib_request_firmware: Info -
Loading firmware file 'rt2870.bin'
[   87.206591] rt2800usb 3-1:1.0: Direct firmware load for rt2870.bin
failed with error -2
[   87.206596] ieee80211 phy0: rt2x00lib_request_firmware: Error -
Failed to request Firmware
[  175.175385] ieee80211 phy0: rt2x00lib_request_firmware: Info -
Loading firmware file 'rt2870.bin'
[  175.175401] rt2800usb 3-1:1.0: Direct firmware load for rt2870.bin
failed with error -2
[  175.175405] ieee80211 phy0: rt2x00lib_request_firmware: Error -
Failed to request Firmware
[  180.338718] ieee80211 phy0: rt2x00lib_request_firmware: Info -
Loading firmware file 'rt2870.bin'
[  180.338735] rt2800usb 3-1:1.0: Direct firmware load for rt2870.bin
failed with error -2
[  180.338739] ieee80211 phy0: rt2x00lib_request_firmware: Error -
Failed to request Firmware
```

缺少`rt2870.bin`，从主机上复制一份到ls1046A上，重新接入usb无线wifi，正确dmesg信息如下

```
ieee80211 phy1: rt2x00_set_rt: Info - RT chipset 3070, rev 0201 detected
[  790.898561] ieee80211 phy1: rt2x00_set_rf: Info - RF chipset 0005
detected
[  790.898942] ieee80211 phy1: Selected rate control algorithm
'minstrel_ht'
[  790.919168] rt2800usb 3-1:1.0 wlx7cdd901b2360: renamed from wlan0
[  790.971681] ieee80211 phy1: rt2x00lib_request_firmware: Info -
Loading firmware file 'rt2870.bin'
[  790.971760] ieee80211 phy1: rt2x00lib_request_firmware: Info -
Firmware detected - version: 0.36
[  791.155964] IPv6: ADDRCONF(NETDEV_UP): wlx7cdd901b2360: link is not
ready
```

```
sudo wpa_supplicant -B -iwlx7cdd901b2360 -c/etc/wpa_supplicant.conf
```

删除无法上网的fm1-mac3网关即可

```
sudo route del default gw _gateway dev fm1-mac3
```

设置主从机通信

从机设置：

```
export ROS_MASTER_URI=http://192.168.230.136:11311
export ROS_HOSTNAME=192.168.230.28
```

主机设置：

```
export ROS_MASTER_URI=http://192.168.230.136:11311
export ROS_HOSTNAME=192.168.230.136
```

自此，0901留下的ch2102驱动、ch343驱动、usb无线网卡驱动均已解决，并且已经解决了串口输出的问题，还有一个问题：将烧录在emmc更改为烧录到ssd，解决容量不足的问题。

内核启动中使用的参数：

```
 Kernel command line: console=ttyS0,115200
earlycon=uart8250,mmio,0x21c0500 root=PARTUUID=7da15185-03 rw rootwait
board_name=ls1046ardb serdes1=1133
```

无法使用串口打断uboot的可能原因，以下信息来源于系统启动输出

```
bootconsole [uart8250] disabled
```

一些有用的相关信息

```
fsl-quadspi 1550000.quadspi: w25q128 (16384 Kbytes)

 EXT4-fs (mmcblk0p3): 1 orphan inode deleted
[    3.685004] EXT4-fs (mmcblk0p3): recovery complete
[    3.691517] EXT4-fs (mmcblk0p3): mounted filesystem with ordered data
mode. Opts: (null)
[    3.699620] VFS: Mounted root (ext4 filesystem) on device 179:3.
[    3.705925] devtmpfs: mounted
[    3.709263] Freeing unused kernel memory: 1216K
```

存档：解析信息到

Welcome to Ubuntu 18.04.1 LTS!

# 0911

```
Starting Remount Root and Kernel File Systems...
Mounting Huge Pages File System...
```

在Linux系统启动过程中，初始根文件系统通常以只读（read-only）的方式挂载，这是为了确保文件系统的完整性和一致性。然而，在系统正常运行后，通常需要对根文件系统进行读写操作，例如写入日志文件、修改配置等。为了允许对根文件系统进行写操作，需要将其重新挂载为读写（read-write）模式。

因此，"Remount Root and Kernel File Systems" 的操作是在系统启动过程中将根文件系统从只读模式切换为读写模式的过程。这通常在启动过程的某个阶段执行，以确保系统在启动后可以正常进行写操作。

所有的系统启动信息以及解析完毕。

这将显示当前系统的内核命令行参数，其中包括引导加载程序的名称。

```
cat /proc/cmdline
```

拷贝制作根文件系统：以下为将当前的根文件系统复制到挂载的目录，该挂载的目录是来自于ssd固态硬盘，通过 `rsync` 命令能制作出一样的根文件系统到120G的ssd固态硬盘上，之后再调整uboot启动系统，挂载的uuid从emmc的改成ssd固态硬盘的即可

挂载之前，如果先/dev/sda没有分区，先建立/dev/sda1分区，然后使用 `mkfs.ext4` 命令来格式化磁盘或分区为 ext4 文件系统

```
sudo mkfs.ext4 /dev/sda1
```

挂载新存储设备，将当前的根文件系统内容复制到新的存储设备上

```
sudo mkdir /mnt/boot
sudo mount /dev/sda1 /mnt/boot
sudo rsync -avx / /mnt/boot
```

输出信息即是已经拷贝制作完成根文件系统到ssd固态硬盘上

```
sent 6,114,841,029 bytes  received 2,677,410 bytes  45,824,108.16
bytes/sec
total size is 6,104,303,376  speedup is 1.00
```

在终端中，运行以下命令来查看eMMC设备的UUID：

```
sudo blkid
```

7da15185是emmc，63d7c5d0是ssd

```
/dev/mmcblk0: PTUUID="7da15185" PTTYPE="dos"
/dev/mmcblk0p1: PARTUUID="7da15185-01"
/dev/mmcblk0p2: LABEL="boot" UUID="0000-0006" TYPE="vfat"
PARTUUID="7da15185-02"
/dev/mmcblk0p3: UUID="57f8f4bc-abf4-655f-bf67-946fc0f9f25b" TYPE="ext4"
PARTUUID="7da15185-03"
/dev/sda1: UUID="36051cfa-4c58-4f34-8e72-e6625f7f8865" TYPE="ext4"
PARTUUID="63d7c5d0-01"
```

这个命令将列出所有存储设备的UUID，包括eMMC、ssd设备。

使用ssd挂载根文件系统，在uboot上进行操作

```
setenv bootargs console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500
root=PARTUUID=63d7c5d0-01 rw rootwait board_name=ls1046ardb serdes1=1133
```

putty中打开输入字符回显的设置方式_wxchbhd的博客-CSDN博客

解决串口登录显示的问题

使用新的线，已经能通过DB9的串口进行登录，之前是因为线无法使用的问题。

## 0912

已经排查完毕，使用usb340模块时出现的能够正常输出log信息但是无法进行输入，是因为有一块芯片把输入强制上拉为高电平了，因此无法正常的输入信息。

长领哥把芯片去掉后，再重新使用usb340模块登录到uboot，能够通过按下 `enter` 键正常打断进入到uboot的命令行，而不是之前的情况：打断进入后，无法正常输入指令。

自此，DB9和usb340模块都能正常登陆到uboot。

当前启动uboot已经是识别到了ssd固态硬盘了，以下为uboot输出信息

```
AHCI 0001.0301 32 slots 1 ports 6 Gbps 0x1 impl SATA mode
flags: 64bit ncq pm clo only pmp fbss pio slum part ccc apst
Found 1 device(s).
```

1. **AHCI 0001.0301 32 slots 1 ports 6 Gbps 0x1 impl SATA mode:** 这是有关 SATA 控制器的信息，包括支持的规范版本、插槽数、端口数、速度和模式。
2. **Found 1 device(s):** 这表示已经发现了一个 SATA 设备。

更改根文件系统的挂载id为ssd固态硬盘的63d7c5d0-01 ，操作如下：

```
setenv bootargs console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500
root=PARTUUID=63d7c5d0-01 rw rootwait board_name=ls1046ardb serdes1=1133
```

保存新的引导配置以便下次引导时使用：

```
saveenv
```

查看是否更改正确

```
printenv
```

由下可知，已经更改成功，使用 `boot` 指令启动

```
bootargs=console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500
root=PARTUUID=63d7c5d0-01 rw rootwait board_name=ls1046ardb serdes1=1133
```

但是还是以之前的参数启动

```
root@localhost:~# cat /proc/cmdline
console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500
root=PARTUUID=7da15185-03 rw rootwait board_name=ls1046ardb serdes1=1133
```

有几种可能的原因：

1. 内核参数重写：在某些情况下，Linux 内核启动时会重写 `bootargs` 中的参数。这可能是因为启动过程中的引导程序或内核命令行参数覆盖了 U-Boot 中的设置。
2. 未正确保存 `bootargs`：确保在 U-Boot 中使用 `saveenv` 命令保存了 `bootargs` 的更改。只是在 U-Boot 命令行中修改 `bootargs` 不足以使更改永久生效。
3. 启动时加载了其他脚本或配置文件：有些 U-Boot 配置可能会在启动过程中加载其他脚本或配置文件，这些文件可能会覆盖 `bootargs` 的设置。需要检查 U-Boot 配置以查看是否存在这样的情况。

4. 硬件或平台相关的问题：某些硬件平台可能具有特定的配置要求，这可能会影响 `bootargs` 的设置。请确保硬件配置与 U-Boot 配置一致

正在修改ls1046ardb_boot.scr，修改之前做的记录 `root=PARTUUID=$partuuid3`

ls1046ardb_boot.scr脚本内容如下：

```
boot.scr[setenv secureboot_validate 'load $devtype $devnum:2
$kernelheader_addr_r /secboot_hdrs/ls1046ardb/hdr_linux.out; load
$devtype $devnum:2 $fdtheader_addr_r
/secboot_hdrs/ls1046ardb/hdr_dtb.out; esbc_validate
$kernelheader_addr_r; esbc_validate $fdtheader_addr_r'
part uuid $devtype $devnum:3 partuuid3; setenv bootargs
console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500
root=PARTUUID=$partuuid3 rw rootwait board_name=$board_name
serdes1=$serdes1 $othbootargs; if load $devtype $devnum:2 $load_addr
/boot/uEnv.txt; then echo Importing environment from uEnv.txt ...; env
import -t $load_addr $filesize; fi; load $devtype $devnum:2
$kernel_addr_r /boot/Image;load $devtype $devnum:2 $fdt_addr_r
/boot/fsl-ok1046a-$serdes1-$serdes2-$product.dtb; env exists secureboot
&& echo validating secureboot && run secureboot_validate;booti
$kernel_addr_r - $fdt_addr_r
xterm-256colorxterm-256color
```

1. 解码脚本：

```
mkimage -A arm64 -T script -C none -n "U-Boot script" -d
ls1046ardb_boot.scr ls1046ardb_boot.scr.new
```

这将解码原始的 `ls1046ardb_boot.scr` 文件并创建一个新的文件 `ls1046ardb_boot.scr.new`，你可以在其中进行编辑。

2. 编辑
3. 编辑完成后，将新的脚本重新编码为 U-Boot 脚本文件：

```
 mkimage -A arm64 -T script -C none -n "U-Boot script" -d
ls1046ardb_boot.scr.new ls1046ardb_boot.scr
```

---

在uboot中修改环境变量

```
setenv boot_scripts "ls1046ardb_update.scr ls1046ardb_boot.scr"
saveenv
```

分区的发现问题，emmc能够被查看，ssd即是（scsi）也能被查看

```
=> part list mmc 0

Partition Map for MMC device 0  --   Partition Type: DOS

Part    Start Sector    Num Sectors     UUID            Type
  1     2048            40960           7da15185-01     0c
  2     43008           204800          7da15185-02     0c
  3     247808          15022080        7da15185-03     83
=> part list scsi 0

Partition Map for SCSI device 0  --   Partition Type: DOS

Part    Start Sector    Num Sectors     UUID            Type
  1     2048            250067632       63d7c5d0-01     83
```

ssd查找也能找到，说明在uboot的时候，是能够知道ssd是63d7c5d0-01，但是为什么root启动参数设置为这个的时候就是失败呢？

```
=> scsi scan
scanning bus for devices...
  Device 0: (0:0) Vendor: ATA Prod.: KINGBANK KM100 Rev: SN12
            Type: Hard Disk
            Capacity: 122104.3 MB = 119.2 GB (250069680 x 512)
Found 1 device(s).
=> scsi part  0

Partition Map for SCSI device 0  --   Partition Type: DOS

Part    Start Sector    Num Sectors     UUID            Type
  1     2048            250067632       63d7c5d0-01     83
```

更改挂载前：

```
root@localhost:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.0G  6.3G  670M  91% /
devtmpfs        1.9G     0  1.9G   0% /dev
tmpfs           1.9G  4.0K  1.9G   1% /dev/shm
tmpfs           1.9G  956K  1.9G   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mmcblk0p2   99M   21M   78M  22% /run/media/mmcblk0p2
/dev/sda1       117G  6.2G  105G   6% /run/media/sda1
tmpfs           378M     0  378M   0% /run/user/0
```

## 0913

重新修改脚本

```
scsi scan;if scsi dev ${devnum}; then setenv devtype scsi;part uuid scsi
$devnum:1 partuuid1;
```

0912的解码后再重新编码为 U-Boot 脚本文件，存在问题，在该二进制文件中这个办法失效，即使是单纯的直接编码解码，不做任何修改，生成的ls1046ardb_boot.scr也是无法启动的

但是在提供的工具flex-builder中就是使用的该方法，flex-builder涉及的boot.scr如下：

```
mkimage -A arm64 -O linux -T script -C none -a 0 -e 0  -n "boot.scr" \
               -d $FBDIR/$bootscript_dec.tmp $FBDIR/$bootscript_dec
```

可能是-n "boot.scr"造成的影响，之前-n选项的参数不同，通过file命令查看，CRC校验也不同，

```
root@localhost:/run/media/mmcblk0p2/boot# file ls1046ardb_boot.scr_old
ls1046ardb_boot.scr_old: u-boot legacy uImage, boot.scr, Linux/ARM 64-
bit, Script File (Not compressed), 867 bytes, Tue Sep  5 02:36:15 2023,
Load Address: 0x00000000, Entry Point: 0x00000000, Header CRC:
0x6FBCC843, Data CRC: 0xD301222E
root@localhost:/run/media/mmcblk0p2/boot#
root@localhost:/run/media/mmcblk0p2/boot# file ls1046ardb_boot.scr
ls1046ardb_boot.scr: u-boot legacy uImage, U-Boot script, Linux/ARM 64-
bit, Script File (Not compressed), 1011 bytes, Sun Jan 28 15:59:25 2018,
Load Address: 0x00000000, Entry Point: 0x00000000, Header CRC:
0x77CCEA95, Data CRC: 0x59AE9DCE
root@localhost:/run/media/mmcblk0p2/boot# date
Mon Jan 29 00:00:19 CST 2018
root@localhost:/run/media/mmcblk0p2/boot# file ls1046ardb_boot.scr.new
ls1046ardb_boot.scr.new: u-boot legacy uImage, U-Boot script, Linux/ARM
64-bit, Script File (Not compressed), 939 bytes, Sun Jan 28 15:59:14
2018, Load Address: 0x00000000, Entry Point: 0x00000000, Header CRC:
0x6E9E59D0, Data CRC: 0xEE1A8114
```

进行修改

1. 解码脚本：

```
mkimage -A arm64 -T script -C none -n "boot.scr" -d ls1046ardb_boot.scr
ls1046ardb_boot.scr.new
```

这将解码原始的 `ls1046ardb_boot.scr` 文件并创建一个新的文件 `ls1046ardb_boot.scr.new`，你可以在其中进行编辑。

2. 编辑
3. 编辑完成后，将新的脚本重新编码为 U-Boot 脚本文件：

```
 mkimage -A arm64 -T script -C none -n "boot.scr" -d
ls1046ardb_boot.scr.new ls1046ardb_boot.scr
```

依旧是失败，根据飞凌的回答，目前先尝试直接重做 `ls1046ardb_boot.scr` 脚本

```
您好：
    直接修改脚本的话是行不通的，关于scr路径源码在OK10xx-linux-
fs/flexbuild/configs/board/ls1046ardb/manifest您可以自己尝试修改一下，关于
这方面的资料暂且还没有，我们这边也没做过此类型的操作测试，抱歉。
```

重做 `ls1046ardb_boot.scr` 脚本步骤：

1.找到 `/home/forlinx/work/OK10xx-linux-fs/flexbuild/configs/board/ls1046ardb` 目录的 `manifest` 文件，然后将 `root=PARTUUID=$partuuid3` 改为 `root=PARTUUID=63d7c5d0-01`

2.回到 `/home/forlinx/work/OK10xx-linux-fs/flexbuild/` 目录，编译如下

```
root@ubuntu:/home/forlinx/work/OK10xx-linux-fs/flexbuild# flex-builder -i mkdistroscr -a arm64 -m ls1046ardb -b qspi -S 1133
INSTRUCTION: mkdistroscr
DESTARCH: arm64
MACHINE: ls1046ardb
BOOTTYPE: qspi
SERDES1: 1133
Image Name:   boot.scr
Created:      Wed Sep 13 03:26:23 2023
Image Type:   AArch64 Linux Script (uncompressed)
Data Size:    868 Bytes = 0.85 kB = 0.00 MB
Load Address: 00000000
Entry Point:  00000000
Contents:
   Image 0: 860 Bytes = 0.84 kB = 0.00 MB
 build/firmware/u-boot/ls1046ardb/ls1046ardb_boot.scr     [Done]
Image Name:   boot.scr
Created:      Wed Sep 13 03:26:23 2023
Image Type:   AArch64 Linux Script (uncompressed)
Data Size:    301 Bytes = 0.29 kB = 0.00 MB
Load Address: 00000000
Entry Point:  00000000
Contents:
   Image 0: 293 Bytes = 0.29 kB = 0.00 MB
 build/firmware/u-boot/ls1046ardb/ls1046ardb_update.scr     [Done]
```

3.将新生成的ls1046ardb_boot.scr复制到板卡上的/run/media/mmcblk0p2/boot目录下替换原本的文件

4.重启

最终结果如下：已经挂载到63d7c5d0-01（120G的ssd）

```
root@localhost:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       117G  6.2G  105G   6% /
devtmpfs        1.9G     0  1.9G   0% /dev
tmpfs           1.9G  4.0K  1.9G   1% /dev/shm
tmpfs           1.9G  2.7M  1.9G   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mmcblk0p2   99M   21M   78M  22% /run/media/mmcblk0p2
/dev/mmcblk0p3  7.0G  6.4G  606M  92% /run/media/mmcblk0p3
tmpfs           378M     0  378M   0% /run/user/0
```

自此，ssd的挂载结束，可使用的容量已经增大。

## 0914-0915

增加realsense相机的使用

### 安装realsense的sdk

1.可能需要安装的一些功能包

```
sudo apt-get install git libssl-dev libusb-1.0-0-dev pkg-config libgtk-3-dev
sudo apt-get install libglfw3-dev libgl1-mesa-dev libglu1-mesa-dev
```

复制编译包【librealsense-master】到任意位置
2.给所有文件可执行权限

```
sudo chmod -R 777 librealsense-master
```

3.适配一些环境变量

```
cd librealsense-master
sudo ./scripts/setup_udev_rules.sh
echo 'hid_sensor_custom' | sudo tee -a /etc/modules
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
```

4.编译安装环境

```
cd librealsense-master
mkdir build
cd build
sudo cmake ..
sudo make
sudo make install
```

## 安装**realsense**的**ros**功能包

编译功能包

功能包【ddynamic_reconfigure-melodic-devel】
功能包【realsense-ros-development】

编译完后直接启动launch文件即可

# 二.自研**ls1046**

根据上面的结果进行分支

如果验证测试比较成功，说明只要做出的系统跟飞凌一致的即可

根据编译手册尝试进行移植，从uboot到整个ubuntu系统，接下来就是跟上面第一步一样，直接安装ROS进行测试