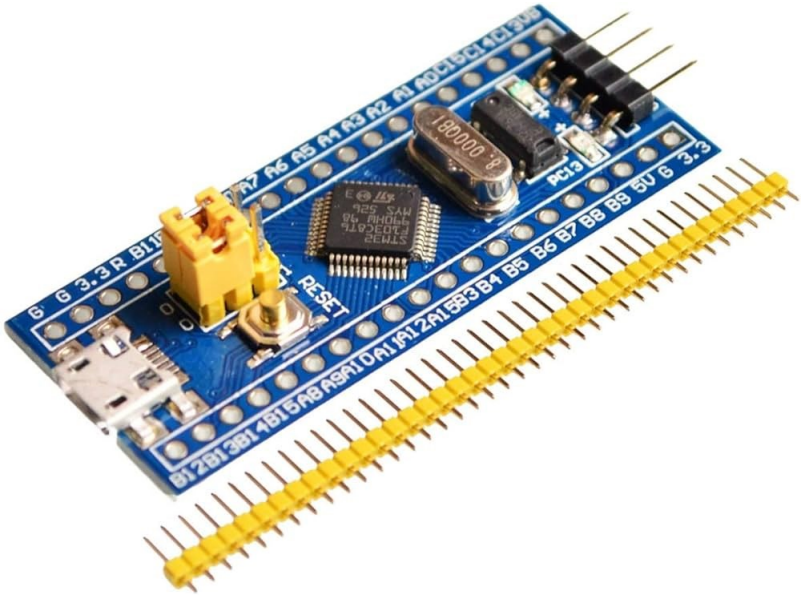
实验设备搭建

- 1、设备
  - 1、计算平台：Allspark2
    - 1、产品使用手册

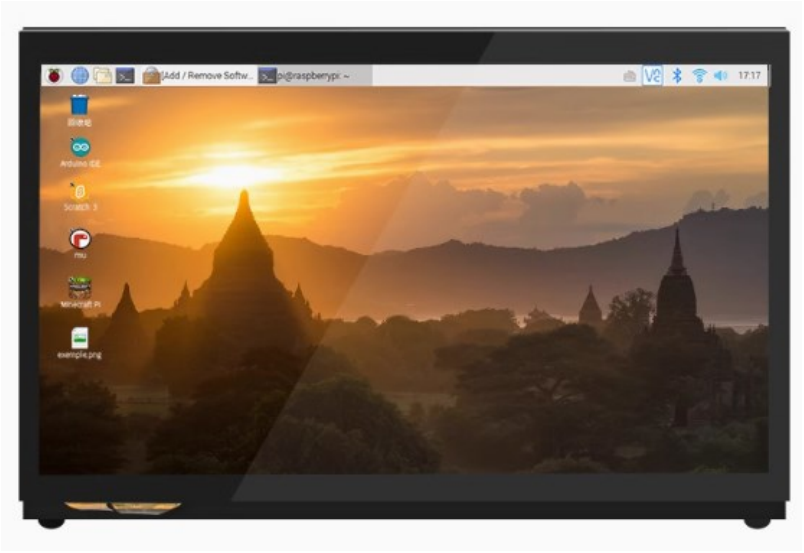
  - 2、无人机平台：M300rtk+禅思 L1（无法二次开发，后续可作为真值使用）
    - 1、无人机

    - 2、禅思 L1

  - 3、外设
    - 1、STM32F103C8T6 已购买



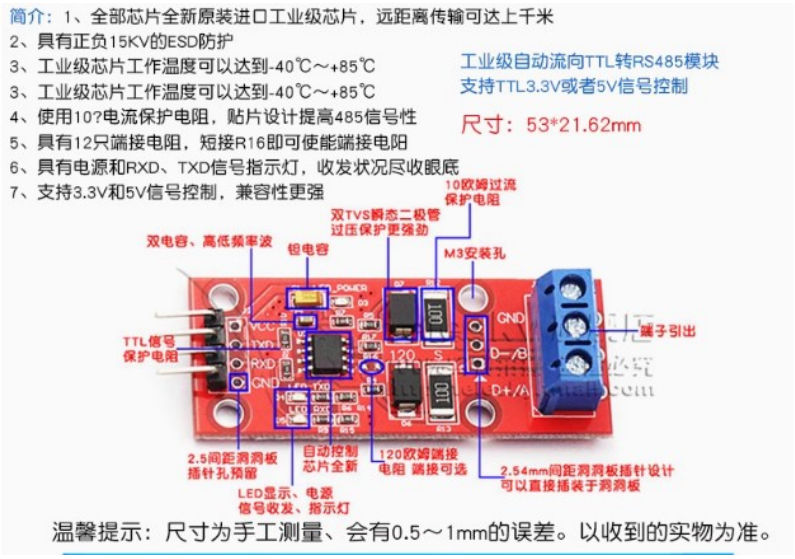
- 2、IPS Screen 10"



- 3、4800mah

- 4、TTL to USB

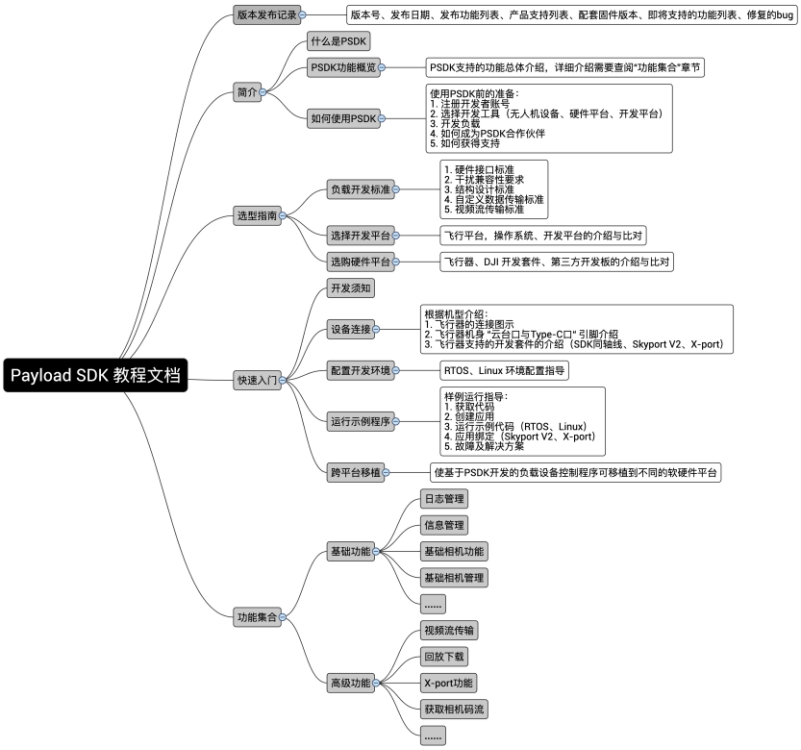
- 5、TTL to 485



- 4、感知传感器
  - 1、相机
    - **HIKROBOT MV-CS050-10GC**
  - 2、激光雷达
    - Livox AVIA
  - 3、激光雷达内置惯导
    - BIM088
- 5、手持平台
  - Fast-livo2 手持平台（相机板和 PC 板需重新设计）
  - 网口需要买新的转接口。
- 6、相关知识
  - 关于时间硬同步：https://zhuanlan.zhihu.com/p/646137909
  - 变种方案：https://www.jianshu.com/p/d4b5cf3c1475

- STM32 改造：https://bbs.huaweicloud.com/blogs/367173

- 2、数据流打通
  - 1、-PSDK----RTK 输出



不支持 L1 放弃 支持相机及 RTK 数据读取

- 1、PSDK　仅支持 L2

  开发可挂载在 DJI 无人机上的负载设备，即通过 PSDK 可以将负载的数据流通过 Dji 接口传输至 PC 端。

- 2、OSDK--OSDK 4.x 所有功能已迁移至 PSDK V3，建议切换 PSDK V3 版本使用。

  OSDK 是一个用于开发无人机应用程序的开发工具包，基于 OSDK 开发的应用程序能够运行在机载计算机上（如 Manifold 2），开发者通过调用 OSDK 中指定的接口能够获取无人机上的各类数据，

- 3、RTK 开发

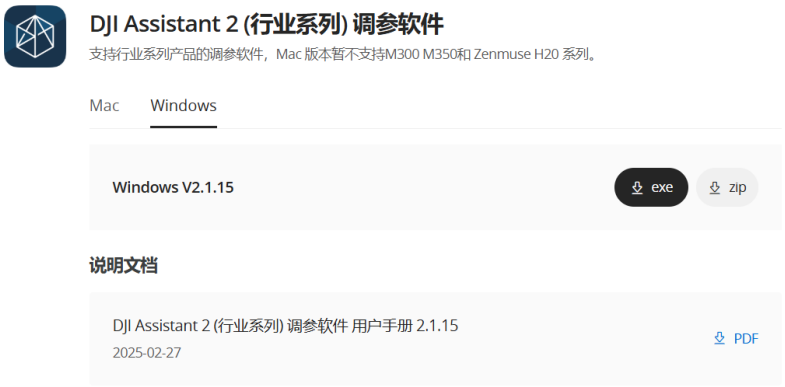
  消息订阅：负载设备能够订阅无人机上各个部件实时产生的传感器数据以及无人机系统状态信息，如姿态四元数、融合海拔高度及 RTK 位置等。

  - 1、基于 PSDK3.X（定位加航向）

    PSDK 中如何订阅数据？ 已有的 git 项目开发 psdk_ros 版：
    https://github.com/Gaiwqqq/DjiPayloadSDKInterfaceForRos1.git

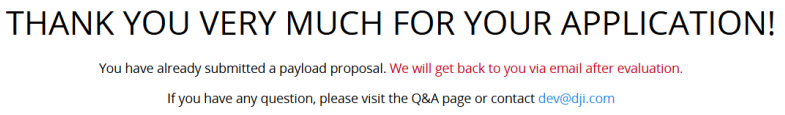
    - 1、前期准备

- 1、固件升级 https://www.dji.com/cn/downloads/products/matrice-300#tuning_params



使用 **DJI Assistant2** 升级无人机和硬件平台的固件。 但 M300 无人机 OSDK 依旧支持广播，广播和订阅均通过串口通道传输。为避免广播包占用带宽，建议在使用 PSDK 3.0 开发时，在 DJI Assistant2 上 OSDK 界面将各数据均设置为不发送。

- 2、注册 DJI PSDK 企业账号&应用注册 https://developer.dji.com/payload-sdk/apply/ 获得应用信息 否则无法创建 PSDK 应用

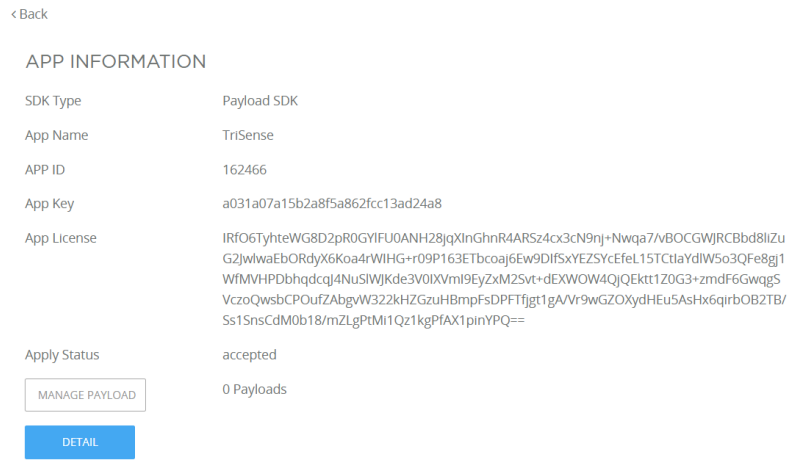


https://developer.dji.com/user/apps/#all fengpan97618@163.com 4010123456A

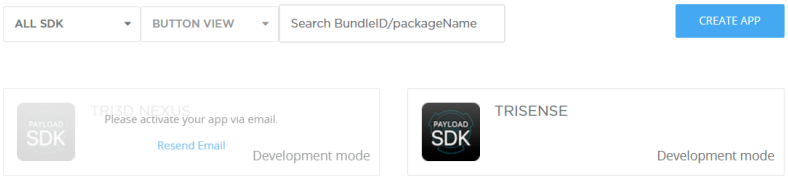
- 1、之后在用户中心处出现如下图标

- 2、点击重发邮件会获得一个链接

- 3、点击链接进入页面 出现对应的信息



- 4、出来之后图标突出化



- 3、无人机

  经纬 matrice M300 RTK

- 4、硬件平台---选用 E-PORT

选用 E-PORT 开发板 适配机型： M350 RTK Mavic 3E/3T Matrice 30/30T **M300 RTK**

- 5、开发平台 本案例为 allspark2（jetson orin NX）Linux

PSDK 支持使用如下工具编译基于PSDK 开发的负载设备，请根据选用的**开发平台**正确地选择工具链。

说明：常用工具链静态库可参考共享链接，有关跨平台移植的详细说明请参见跨平台移植。

工具链名称	目标平台	典型芯片型号	推荐开发平台
aarch64-linux-gnu-gcc	aarch64-linux-gnu	NVIDIA Jetson TX2、Rockchip RK3399 pro	Manifold2-G、瑞芯微Toybrick开发板
x86_64-linux-gnu-gcc	x86_64-linux-gnu	64位intel处理器，如 Intel Core i7-8550U	Manifold2-C
arm-linux-gnueabi-gcc	arm-linux-gnueabi	ZYNQ、I.MX6Q	-
arm-linux-gnueabihf-gcc	arm-linux-gnueabihf	支持硬件浮点运算的处理器，如OK5718-C等	-
armcc-cortex-m4	Cortex M4/M4F系列MCU	STM32F407IGT6、STM32F405RGT6	STM32F407-Eval、STM32F407探索者开发板等

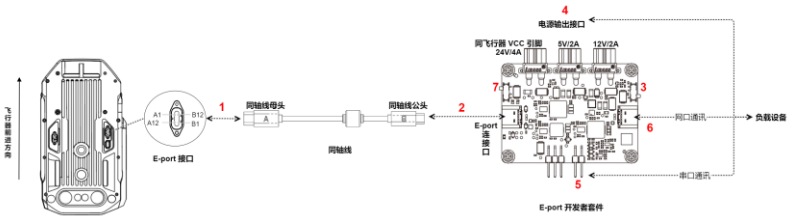
- 2、硬件组装-连接

1. 准备硬件设备

硬件	个数	用途
英伟达 Jetson Nano 开发套件	1	PSDK 的运行平台
高清显示器（HDMI 或 DP 接口）+ HDMI 或 DP 连接线	1	用于开发板 Ubuntu 系统的界面显示
键盘 + 鼠标	1	用于开发板 Ubuntu 系统的人机交互
无线网卡/网线	1	用于开发板访问网络资源
E-Port 开发者套件	1	用于连接开发板和飞行器
M350 RTK 飞行器 + 配套产品（遥控器+电池+负载+充电箱）	1	应用开发硬件平台
笔记本电脑/台式 PC	1	用于连接飞行器/负载进行升级以及模拟器
USB 转 TTL 串口模块	1	用于连接开发板和 E-Port 开发者套件
USB-C USB 连接线	1	用于连接飞行器的调参口，用于对飞行器/负载进行升级以及在 PC 上使用模拟器
Micro USB 连接线	1	用于连接开发板和 E-Port 开发者套件
USB-C OTG 转接线	1	用于连接开发板和 E-Port 开发者套件
2.54mm 杜邦线	若干	用于连接开发板和 E-Port 开发者套件

https://developer.dji.com/doc/payload-sdk-tutorial/cn/quick-start/quick-guide/jetson-nano.html

- 1、无人机与 E-port 连接



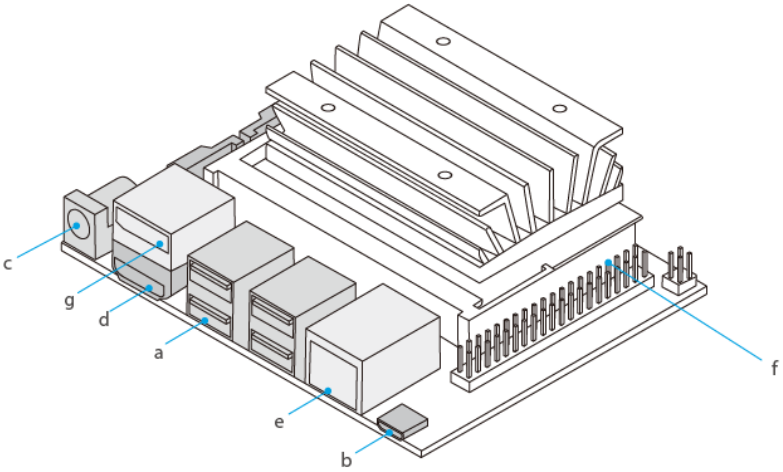
准备一台高清显示器、一套键盘鼠标和宽敞的桌面环境。 准备一套 M350 RTK 飞行器环境，安装电池后将飞行器摆放到桌面上合适的位置。 如果需要执行飞行控制，使用笔记本或者台式 PC 通过 USB-C 连接线连接飞行器的调参口（调参口为下图中 E-Port 接口旁的 USB-C 接口）。 如图. 飞行器设备连接所示，准备 E-Port 开发者套件并连接到飞行器。 检查并确认 E-Port 开发者套件的电源拨码开关（标识 7）处于 OFF 状态。 将 E-Port 开发者套件中的 USB 口（标识 2）通过同轴线连接到飞行器的 E-Port 接口（需要注意 A/B 面）。

- 2、计算平台与 E-port 连接 TTL 转 USB 使用 FT232 芯片的。

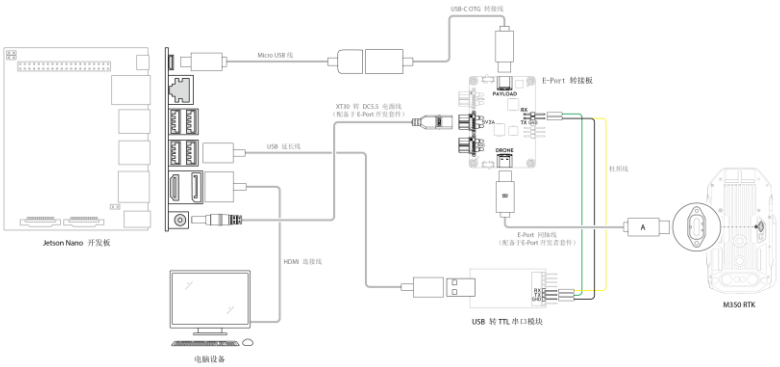
1、通过杜邦线连接 E-Port 开发者套件的 UART（标识 5）引脚连接 USB 转 TTL 串口模块（开发者套件 TX、RX、GND 分别连接串口模块的 RX、TX、GND）。 2、将 USB 转 TTL 串口模块连接到 Jetson Nano 开发板的 USB 口（标识 a）。 3、将 E-Port 开发者套件的 USB 主从切换拨码开关（标识 3）拨到 Host 状态。 4、将 E-Port 开发者套件的 USB （标识 6）通过 USB-C OTG 转接线和 Micro USB 线连接到开发板的 Micro USB OTG 口（标识 b）。 *5、连接 E-Port 开发者套件的 PPS 引脚（标识 5）到开发板 40pins GPIO 的引脚 7（标识 f），可以查阅 open in new window。本连接步骤仅用于时间同步，不需要时间同步则本步骤可以忽略。 6、通过 XT30 转 DC 供电线，连接 E-Port 开发者套件的电源输出接口（标识 4，可选 12V/2A、5V/2A) 到开发板的 DC 电源接口（标识 c，按需选择是否使用外部电源供电）。 7、将开发板的 HDMI 显示输出（标识 d）接入到高清显示器，并且接入键盘鼠标到开发板的 USB 口（标识 a），接入无线网卡或者具备网络能力的有线网口到开发板网口（标识 e）。 8、检查所有连接是否稳定和正常。 9、进行飞行器整机启动，将 E-Port 开发者套件的电源拨码开关打开，检查绿色指示

灯是否亮起，并且检查开发板是否正常供电并启动。

- 计算平台



- 连接图



- 3、环境部署
  - 1、依赖

    FFmpeg ffplay OpenCV libaio CMake libusb /* 更新软件仓库 */ sudo apt-get update sudo apt-get upgrade sudo apt-get install automake sudo apt-get install libaio-dev /* 从 https://opus-codec.org/ 上下载 opus-1.3.1 源码，并进行安装 */ tar -xzvf opus-1.3.1.tar.gz cd opus-1.3.1/ autoreconf -f -i ./configure make -j16 sudo make install /* 从 GitHub 下载并安装 FFmpeg 4.3 源码 */ tar -zxvf ffmpeg-4.3.2.tar.gz ./configure --enable-sharedm ake && make install /* 检查 OpenCV 是否安装 */ opencv_version /* 安装 libusb */ sudo apt-get install libusb-1.0-0-dev

  - *2、开启 Jetson Nano（ORIN NX） 的 USB bulk 功能// **如果是 M300 机型，不需要配置 BULK 环境，连接后检查端口生成没问题可以直接使用。**https://github.com/dji-sdk/Payload-SDK/issues/228#issuecomment-2516833392

```
/* Back up the bulk configuration file of jetson nano */
cp /opt/nvidia/l4t-usb-device-mode-start.sh ~/Desktop/nano-usb-config/l4t-usb-device-mode-start-bk.sh
/* Using PSDK's bulk configuration file */
sudo cp startup_bulk/psdk-usb-configure.sh  /opt/nvidia/l4t-usb-device-mode-start.sh
/* Set the startup_bulk file path in the script */
/home/dji/Desktop/startup_bulk/startup_bulk /dev/usb-ffs/bulk1 &
/home/dji/Desktop/startup_bulk/startup_bulk /dev/usb-ffs/bulk2 &
/* After restarting the system, make sure that startup_bulk is running normally */
ps -aux | grep startup_bulk
```

- 3、下载 PSDK 软件包

  https://github.com/dji-sdk/Payload-SDK

- 4、修改配置
  - 1、将生成的 PSDK 应用信息，填入前一步获取的 PSDK 软件开发包的指定文件中，波特率在 DJI Assistant2 的 Payload SDK 部分。
    - samples/sample_c/platform/linux/nvidia_jetson/application/dji_sdk_app_info.h
    - samples/sample_c++/platform/linux/nvidia_jetson/application/dji_sdk_app_info.h

```
/* Exported constants --------------------------------------------------------
// ATTENTION: User must goto https://developer.dji.com/user/apps/#all to
// information then fill in the application information here.
#define USER_APP_NAME               "your_app_name"
#define USER_APP_ID                 "your_app_id"
#define USER_APP_KEY                "your_app_key"
#define USER_APP_LICENSE            "your_app_license"
#define USER_DEVELOPER_ACCOUNT      "your_developer_account"
#define USER_BAUD_RATE              "460800"
```

  - 2、对 PSDK 的硬件连接配置文件进行对应的修改。 修改为 DJI_USE_ONLY_UART（官方手册是出现端口号后在头文件中设置为 USB&bulk 即可）
    - samples/sample_c/platform/linux/nvidia_jetson/application/dji_sdk_config.h
    - samples/sample_c++/platform/linux/nvidia_jetson/application/dji_sdk_config.h

```
#define DJI_USE_ONLY_UART                (0)
#define DJI_USE_UART_AND_USB_BULK_DEVICE   (1)
#define DJI_USE_UART_AND_NETWORK_DEVICE    (2)

/*!< Attention: Select your hardware connection mode here.
* */
#define CONFIG_HARDWARE_CONNECTION         DJI_USE_UART_AND_NETWORK_DEVICE
```

    如果使用的连接方式是 DJI_USE_UART_AND_NETWORK_DEVICE，需要在以下配置文件中对当前使用的网卡设备名称和 VID、PID 进行配置。 连接方式是 DJI_USE_UART_AND_USB_BULK_DEVICE，需要在以下配置文件中对当前使用的 USB Bulk 端点进行配置。

- 5、编译及运行

  cd Payload-SDK/ mkdir cmake-build-debug && cd cmake-build-debug cmake ../ && make -j8

- 4、ROS 发布 RTK 数据

  https://blog.csdn.net/Donald_Shallwing/article/details/144652456?fromshare=blogdetail&sharetype=blogdetail&sharerId=144652456&sharerefer=PC&sharesource=CCChester&sharefrom=from_link

- **1、了解 PSDK 架构----samples 是大疆官方给出的代码示例，有 C 和 C++两种代码，包含了 GPS 信息获取和图像传输等功能，后续的 ROS 代码也是在 samples 的基础上进行修改的**

  CMakeLists.txt doc EULA.txt psdk_lib samples tools psdk_lib 中包括了 SDK 中所有库的头文件，以及不同平台下的编译器；samples 是大疆官方给出的代码示例，有 C 和 C++两种代码，包含了 GPS 信息获取和图像传输等功能，后续的 ROS 代码也是在 samples 的基础上进行修改的；CMakeLists.txt 则是使用 CMake 编译源码时的编译配置文件。

- 2、编译过程中遇到的问题

  opencv 的版本管理

- 3、编译测试 可执行文件 dji_sdk_demo_linux

  深入了解 demo 源码的含义

  - 1、module_sample

    机载传感器模块代码。 传感器模块代码实现了每个传感器数据接收的功能，这包括机载相机管理、飞行控制、**云台控制（gimbal）**，**健康管理系统（HMS）**、**实时视频流（liveview）**、**视觉图像（perception）**等

  - 2、platform

    主函数实现的代码. 程序需要创建一个 Application，这个 Application 是一个类，当 Application 类创建时会调用相应的构造函数来创建一个 Dji 应用程序核心，这个核心与 ROS 核心类似。可以推测出，大疆对机载传感器的管理借鉴了 ROS 中传感器节点的管理方法，即将每一个传感器节点看作一个应用线程。程序开始定义各种后续需要使用的变量，其中 T_DjiOsalHandler 是一个**操作系统抽象层(Opertating System Level)**类，这个类用于实现与不同操作系统之间的兼容。 **T_DjiDataTimestamp, T_DjiFcSubscriptionRtkPosition, T_DjiFcSubscriptionGpsPosition** 分别记录了时间戳、RTK 和 GPS 信息，而此处的时间戳是指的程序启动到当前的运行时间间隔，并非是 UNIX 时间戳。

- 4、

- 2、基于 OSDK---不推荐 参考资料没有任何更新，较久远

  - OSDK 4.x 所有功能已迁移至 PSDK V3，建议切换PSDK V3版本使用。
  - OSDK 4.0 支持M210 V2、M210 RTK V2、M300 RTK机型，该版本会继续保持基础维护。
  - A3、N3、M100、M210 V1、M600和M600 Pro机型仅支持使用OSDK 3.9，该版本会继续保持基础维护。

  https://developer.dji.com/cn/document/433991be-9586-43cb-b171-ffafe8468684 在 M300 RTK 机型上，PSDK 3.X 既可以用来连接至 OSDK (Onboard SDK) 端口开发 OSDK 功能，也可以连接至 PSDK 云台端口集成第三方负载设备，分别**继承并拓展**了原本 OSDK 4.X 和 PSDK 2.X 上的功能。 版本迁移主要涉及 M300 RTK 机型，因其 OSDK 端口和 PSDK 端口独立，且旧版本是独立的开发包。已经在 M300 RTK 上完成 OSDK 和 PSDK 功能或设

备的开发者，需要从旧版本 OSDK 4.X 和 PSDK 2.X 迁移到官方维护的 PSDK 3.X 版本，可以与 DJI SDK 版本同步，方便适配 SDK 新增的功能以及后续的版本维护。

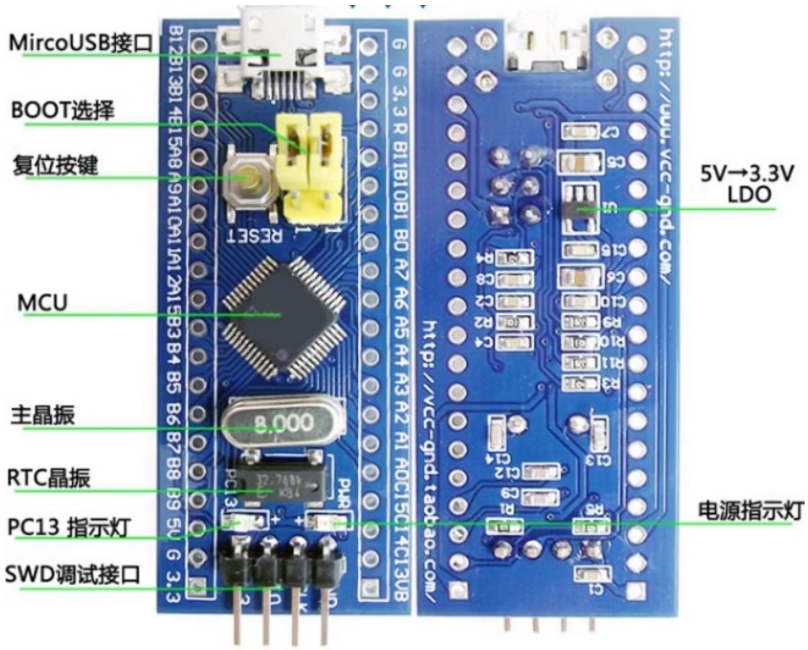
- 1、整体流程

- 2、M300RTK 推荐使用 E-Port

M300 RTK	OSDK 接口	E-Port 开发者套件 SDK 同轴线套装 OSDK 拓展组件（不推荐）
	负载接口（云台口）	SkyPort V2 开发套件 SkyPort V2 量产套装 X-Port 标准负载云台 SkyPort V1 开发套件 SkyPort V1 量产套装

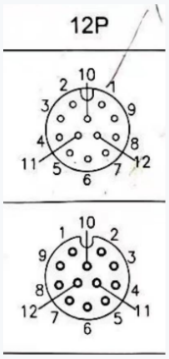
- 3、外部 RTK（仅定位）

- 2、手持设备数据流打通
  - 1、学习实操视频/文档
    https://gitee.com/gwmunan/ros2/wikis/pages?sort_id=10502383&doc_id=4855084
    FAST-LIVO Reproduction (Part 1) FAST-LIVO Reproduction (Part 2) FAST-LIVO Reproduction (Part 3) FAST-LIVO Reproduction (Part 4) FAST-LIVO Reproduction (Part 5)
  - 2、设备连接
    - 1、STM32 同步
      - 1、STM32 引脚设置 关注引脚：PA1 PB5 PA9 VCC GND



- pB5 1hz pwm **from TTL to RS-485 连 TTL2485 RXD // 产生 RS485_Output A+ and RS485_Output B-// RS485_Output A+ 接到 avia 航空插头（那根线）的 PIN12 (Sync+) RS485_Output B- 接到 avia 的 PIN11 (Sync-)**
- pA1 10HZ PWM 接 MVS camera PIN2 (OPTO_IN)
- pA9 GPRMC 1HZ TTL to USB (RXD) 连到电脑上 PA10 也要连接对应到 TXD
- VCC （3.3v） 连 TTL to USB(VCC)
- GND 连 TTL to USB(GND)

- 2、激光雷达 livox avia

Livox AVIA LiDAR M12	Peripheral Function	Diagram
PIN 1	Power: +	
PIN 2	Ground	
PIN 7	Ethernet: RX-	
PIN 6	Ethernet: RX+	
PIN 5	Ethernet: TX-	
PIN 4	Ethernet: TX+	
PIN 11 (Sync-)	RS485_Output B-	
PIN 12 (Sync+)	RS485_Output A+	

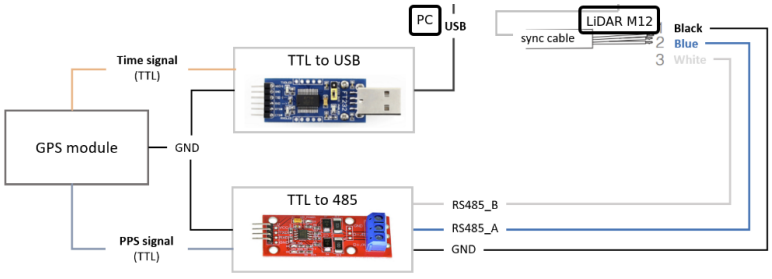
AVIA 12-Pin Interface, top: female bottom: male

- 1、4.5.6.7 接 RJ45 网口的 TX+，TX-，RX+，RX-，传输点云数据
- 2、1.2 接电源正（蓝色/白色）、负极（银色裸线）（电压范围：10~15V），供电
- 3、pin11 接 RS485_Output B- pin12 接 RS485_Output A+ // 如果使用 Livox 转换器 **直接 连 STM32 PB5 (PPS signal) 到转换器 Sync Port 无需 转换 TTL to 485 level.**

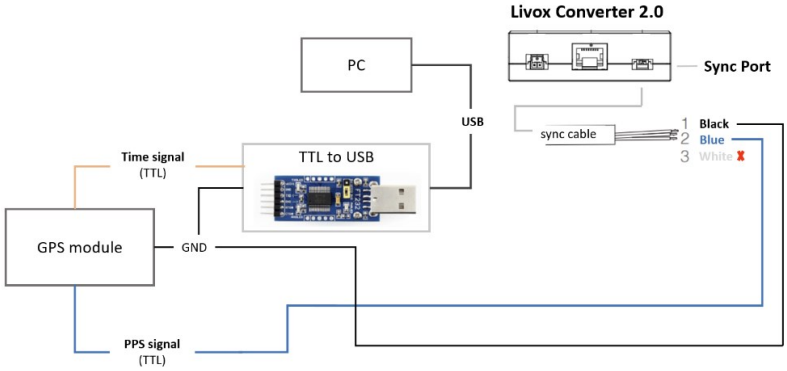
表 2.2.2 同步信号线线序

针	信号	属性	说明	颜色
1	Ground	Power	Ground	黑
2	Sync+	Input	3.3V LVTTL 电平，秒脉冲	蓝
3	Reserved	Reserved	未定义信号	白

- 1、直连



- 2、转换器



- 4、pin3 pin8 pin10 地线 银色
- 5、pin9 供电 蓝色
- 6、时间同步方式

- 3、相机 HIKROBOT MV-CS050-10GC

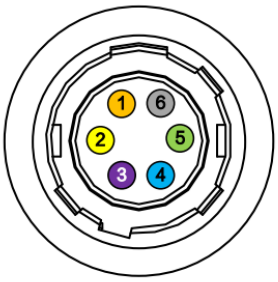


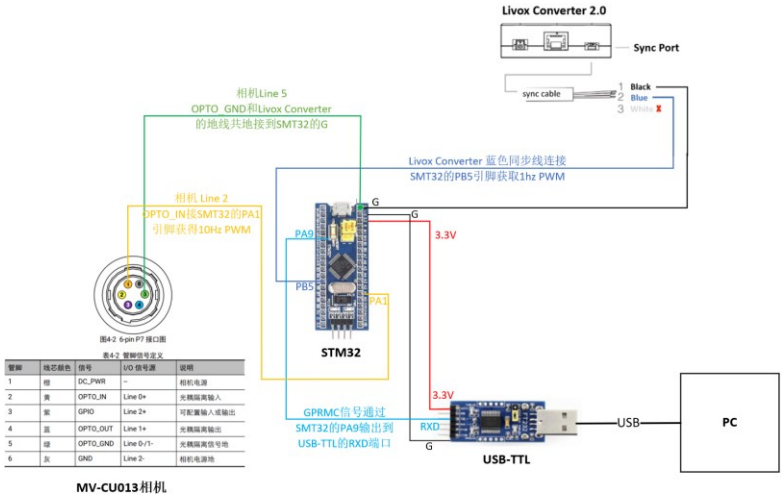
图4-8 6-pin P7 接口

表4-2 6-pin P7 管脚定义

管脚	线芯颜色	信号	I/O 信号源	说明
1	橙	DC_PWR	--	相机电源
2	黄	OPTO_IN	Line 0+	光耦隔离输入
3	紫	GPIO	Line 2+	可配置输入或输出
4	蓝	OPTO_OUT	Line 1+	光耦隔离输出
5	绿	OPTO_GND	Line 0-/1-	光耦隔离信号地
6	灰	GND	Line 2-	相机电源地

1、红，2，绿，3、蓝，4，黄，5，棕，6，黑

- 1、pin2 接 STM32 的 PA1 引脚，获取外部 10HZ 的 PWM 信号
- 2、其余引脚不动
-



- 2、GPS 信号同步（精度更高）

- 3、驱动烧录
  - 1、激光雷达驱动

    git 下载 livox 官网链接

  - 2、相机驱动

    修改双持平台驱动 / HIKROBOT-MVS-CAMERA-ROS 海康 Camera MVS Linux SDK 二次开发封装 ROS packge 过程记录

    - 1、相机驱动无法打开设备-换一个驱动就好使了

      1、经过参数对照和报错码查询，发现是因为设备无访问权限。但是 MVS 可以打开，经过上午对硬件同步线的拔插之后，MVS 也无法打开，无访问权限。 2、之后使用 wireshark 后可以看到相机对应的网口一直被占用，关闭以太网连接依然被占用，拔了硬件同步 USB 后占用消失，MVS 可以访问。后面又一次遇到该问题，即使拔了 USB 依然无访问权限，断电之后可以访问了。 3、通过观察第一次报错，发现设备能够被打开，但是在 StartGrabing 函数那里会直接 die 掉，也没有错误码。改成无触发依然报错。 **4、明确了一件事，设备无法打开的原因是 grabing 崩溃之后，进/线程没有被正确关闭，导致相机流一直被占用！并非是 USB 导致的！** （换成网口，无触发仍然报错。如果打开设备没有问题，但是抓取数据时崩溃，是否是参数设置导致有问题导致报错？） 5、客户端的格式为 RGB8,驱动也为 RGB8,但像素类型头文件中没有这个，只有 RGB8_Packed。换成 BayerRG8 试试？打开心跳模型后，数据占用问题消失！

      - 1、opencv 版本冲突

        同样的代码在 orin 32G 中没有问题，警告一样，很可能不是该原因

      - 2、参数设置不匹配

        对参数多次对比，保证一致仍然失败

      - 3、设备访问权限未打开

        根本原因是相机被野线程占用了

- 4、TTL 转 USB 对驱动影响

  拔了之后，仍然会崩溃，说明是驱动代码原因。

- 5、驱动中抓取函数崩溃导致野线程--因为心跳被关闭了。
  - 1、对比启动前后资源消耗情况，如果有野线程，资源消耗一定会比初始态高。
  - 2、即使断了网线有重新连接，依然有数据不停地被接受。
  - 3、客户端与图像格式编码是否正确？
- 6、在开启线程那里报错！

```
// if (image_scale > 0.0) {
//   cv::resize(srcImage, srcImage, cv::Size(srcImage.cols * image_scale, srcImage.rows * image_scale), cv::INT
// } else {
//   printf("Invalid image_scale: %f. Skipping resize.\n", image_scale);
// }
```

经过一步步打印排查，最后原因竟然是这个图像尺寸放缩！？

- 2、新驱动能否使用 RGB8 格式？可以使用，注意驱动将拿到图的时间设为当前 ros 时间戳了，需要跟雷达的时间戳统一。
- 3、原驱动为什么不行？区别。有效驱动多了以下内容
  - 1、在枚举设备列表时，可以看到有效的驱动对 USB 设备和 GiGE 设备进行了判断。即判断配置文件中的设备序列号与相机获取的序列号是否一致。

```
if (stDeviceList.nDeviceNum > 0)
{
    for (int i = 0; i < stDeviceList.nDeviceNum; i++)
    {
        ROS_INFO("[device %d]:", i);
        MV_CC_DEVICE_INFO* pDeviceInfo = stDeviceList.pDeviceInfo[i];
        if (NULL == pDeviceInfo){
            break;
        }
        if (stDeviceList.pDeviceInfo[i]->nTLayerType == MV_GIGE_DEVICE)
        {
            if(strcmp(cam_info->DeviceSerialNumber.c_str(), (char *)pDeviceInfo->SpecialInfo.stGigEInfo.chSerialNumber
            {
                nIndex = i;
            }
        }
        else if (stDeviceList.pDeviceInfo[i]->nTLayerType == MV_USB_DEVICE)
        {
            if (strcmp(cam_info->DeviceSerialNumber.c_str(),(const char*)pDeviceInfo->SpecialInfo.stUsb3VInfo.chSerialN
            {
                nIndex = i;

            }
        }
        PrintDeviceInfo(pDeviceInfo);
    }
```

  - 2、对 GIGE 包的大小进行判断和设置

```
// ch:探测网络最佳包大小(只对GigE相机有效) | en:Detection network optimal package
if (stDeviceList.pDeviceInfo[nIndex]->nTLayerType == MV_GIGE_DEVICE)
{
    int nPacketSize = MV_CC_GetOptimalPacketSize(handle_);
    if (nPacketSize > 0)
    {
        nRet = MV_CC_SetIntValue(handle_,"GevSCPSPacketSize",nPacketSize);
        if(nRet != MV_OK)
        {
            ROS_WARN("Warning: Set Packet Size fail nRet [0x%x]!", nRet);
        }
    } else
    {
        ROS_WARN("Warning: Get Packet Size fail nRet [0x%x]!", nPacketSize);
    }
}
// 设置触发模式为off
```

  - 3、对图像的长宽进行设置

```
//设置图像大小
nRet = MV_CC_SetIntValueEx(handle_, "Width", cam_info->Width);
if (MV_OK != nRet)
{
    ROS_ERROR("MV_CC_Set Width fail! nRet [%x]", nRet);
    return false;
}
nRet = MV_CC_SetIntValueEx(handle_, "Height", cam_info->Height);
if (MV_OK != nRet)
{
    ROS_ERROR("MV_CC_Set Height fail! nRet [%x]", nRet);
    return false;
}
```

- 4、图像实际是 BGR8，改为 RGB8 后依然有效。
- 5、原驱动使用共享内存时间戳，这是什么意思，

  反正需要先使用 LIVOX 的驱动生产一个 timeshare 文件，然后相机驱动读取这个文件中的数据实现数据的纳秒级对齐。

- 3、STM32 驱动

  使用 Keil 打开和烧录

- 4、SLAM 算法部署
  - 1、编译过程中找不到 CustomMsg.h，奇怪，多编译几次就没报错了，操！
  - 2、fout_imu 报错，，在/your_ws/FAST-LIVO2/include/IMU_Processing.h 添加 #include <fstream>
  - 3、/usr/bin/ld: /usr/lib/gcc/aarch64-linux-gnu/9/../../../aarch64-linux-gnu/libpcl_io.so: undefined reference to `libusb_set_option'
  - 4、初始化时出现 [laserMapping-2] process has died

    需要注释掉 cv::initUndistortRectifyMap 奇怪

  - 5、libvikit_common.so 未定义得 Sophus::SE3::operator*

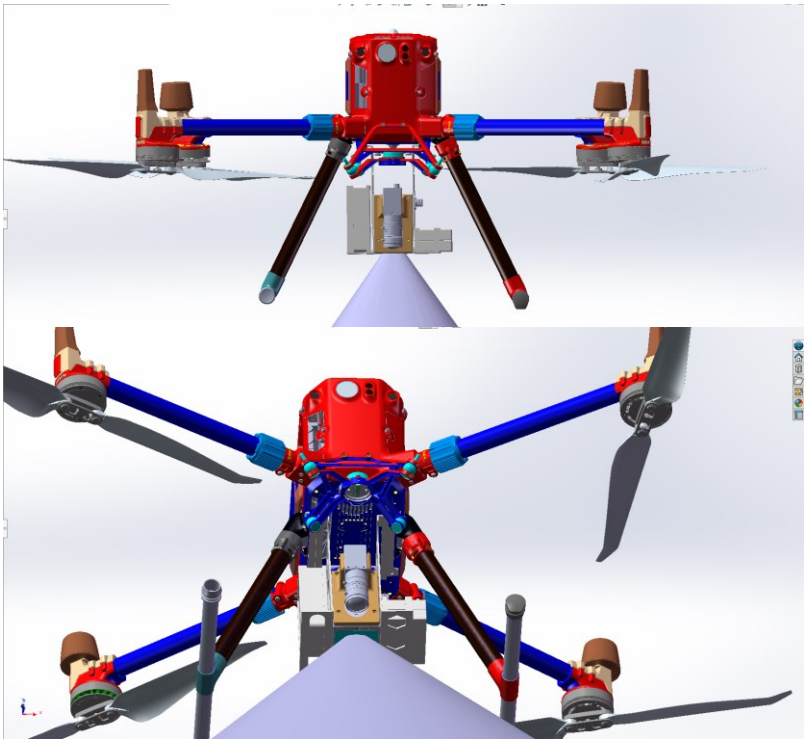
    solve it by adding set(Sophus_LIBRARIES libSophus.so) in CMakeLists.txt file in vikit_commo

- 4、参数标定
  - 1、相机内参标定（Camera Calibration）
    - 1、相机帧率设为 4hz
    - 2、采集时间 20s 以内
    - 3、内参及畸变系数

      1707.22517434064 0 1031.37726149095 0 1708.98591016819 820.165428024337 0 0 1 畸变：-0.072189 0.033258 -0.005458 0.000739 0.000000

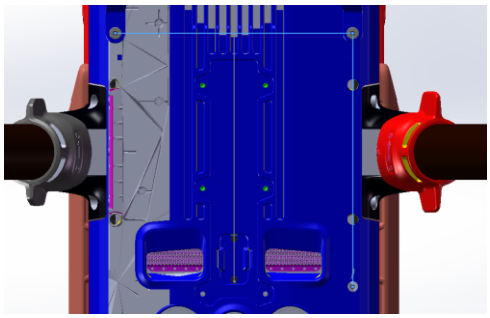
  - 2、激光相机标定

        - 1、捕获一帧相机图像
        - 2、录制 5 秒的激光雷达数据
    - 3、*GPS 添加
- 3、无人机搭载部件设计

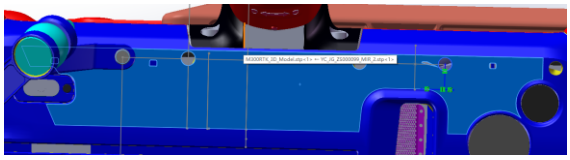


可参考的工作 mars lvig 正面朝下采集

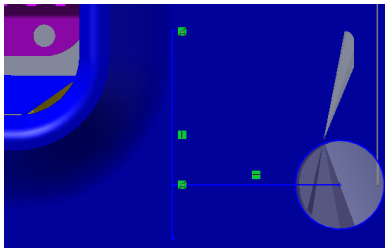
- 1、准备工作
    - 1、无人机 3D 模型
        - 1、安装孔位 左右间距 106mm，前后间距 113mm。孔位距离中间顶点有≤5mm 的落差。



        - 2、孔位安装面并非水平有一定角度，经测量约有 倾角 5.51°

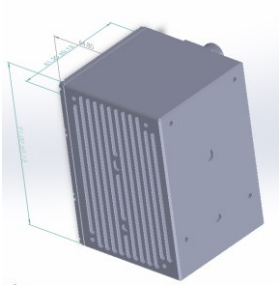


        - 3、孔位中心距凸起边缘距离 8.9mm

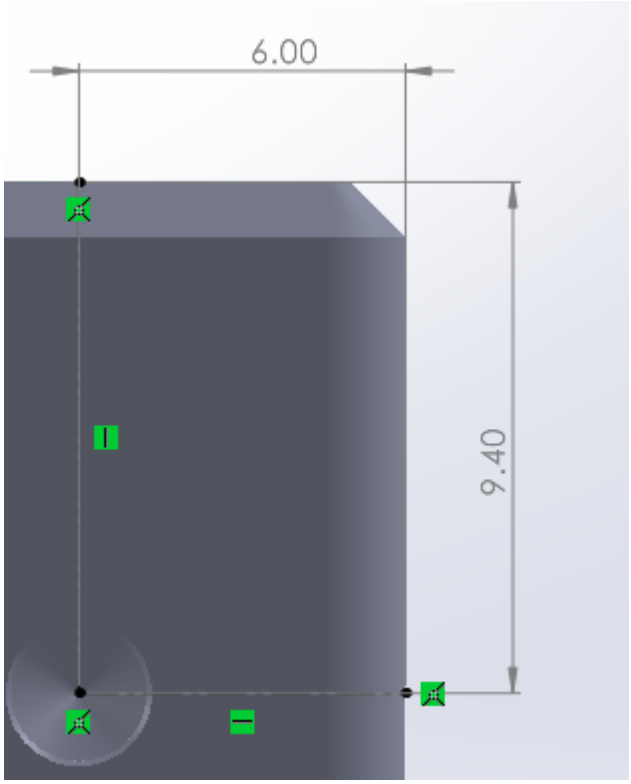


- 要考虑脚架避让问题　前孔距脚架中心 53mm，后孔距脚架中心 60mm。

- 2、传感器尺寸



- 1、孔位



- 3、计算机尺寸

  102.5mm *62.5mm* 31mm

- 4、电池尺寸*两个

  87*57*17

- 2、部件

- 4、无线通信模块

  接收消息并以话题的形式播送。

- * 问题
  - 1、PC 只有两个百兆网口，而相机是千兆，相机帧率无法满足。

    1、测试相机网口实际采集帧率。
  - 2、系统存在两种 opencv 库，本来编译时存在冲突，但是使用 sudo apt-get install libopencv-dev=4.2.0+dfsg-5 后警告就消失了？

    3. 确保安装 OpenCV 4.2 完整套件

    bash 复制

    ```
    sudo apt-get install libopencv-dev=4.2.0+dfsg-5
    ```

  - 3、rosbag 录制数据集时相机与激光雷达差 100ms，直接使用算法似乎没有这个问题.。没有录制 IMU，实际已经完全对齐了
  - 4、如何可视化边缘电脑桌面------虚拟显示器。
  - 5、将单片机所有引线完全替换为锡焊

    在外场实测时，单片机供电不稳定。
  - 6、打通 RTK 从无人机到边缘平台的传输。