

Z410 无人机使用教程目录

1 资料下载链接

- 1.1 missionplanner 下载链接
- 1.2 ardupilot 固件下载链接
- 1.3 操作端 ubuntu 镜像下载链接

2 Ubuntu 镜像安装视频教程

3 Z410 基础版教程

- 3.1 Z410 基础板介绍
- 3.2 Z410 基础板室外飞行基本操作

4 Z410-3B+进阶版教程

- 4.1 网络配置
- 4.2 使用 Dronekit 测试连接飞控并读取状态数据
- 4.3 启动 SITL 仿真环境并运行 python 示例
- 4.4 SITL 仿真环境环境参数设置
- 4.5 SITL 仿真结合 MP 地面站运行 python 示例
- 4.6 Python 脚本控制无人机飞行

5 Z410-3B+T265 视觉版教程

- 5.1 使用 pyrealsense2 启动 T265 及飞行测试
- 5.2 ROS 系统下 T265 启动及飞行测试

6 后续可开发更多功能

1. 资料下载链接

1.1 Missionplanner 地面站软件下载链接：

<https://pan.baidu.com/s/1h2o82bcXtuHvk-8DN4I7tg>

提取码：deq6

注意：电脑最好是 Win10 系统，飞控连接电脑后一般或自动识别安装驱动，分配端口。如果没有自动安装驱动，需要下载驱动精灵，从驱动精灵来安装飞控驱动。

1.2 ardupilot (3.6.11) 固件下载链接：

https://pan.baidu.com/s/1dhFrmA8FpGdzqO8_WAwpzQ

提取码：e1a0

无人机发货前已经组装调试好，不需要再下载固件，可留着备用。

2. Ubuntu 镜像安装教程

2.1 之前使用过 Ubuntu 系统的同学，可以不需要安装这个镜像。参考以下教程安装 Ardupilot 开发环境和 ROS 系统即可；

Ardupilot 开发环境搭建：

<https://mp.weixin.qq.com/s/bQKTX83QI3d-xDys4xPLOQ>

2.2 我们也制作好电脑端 Ubuntu 系统的操作镜像，此镜像已经安装好 Ardupilot 开发环境和 ROS 系统。大家可直接查看和学习飞控源码。

Ubuntu 镜像下载链接：

链接：<https://pan.baidu.com/s/1zSJz10zqVYUZRXxnDSQSNw>

提取码：zidd

请参考以下教程在电脑上安装镜像系统。

链接：<https://pan.baidu.com/s/1IjKYLR6pYuVm606e2LNNTg>

提取码：3yoe

3. Z410 机型介绍

Z410 机型是一款二次开发入门机型。基础版采用 pixhawk2.4.8 飞控和 M8N GPS 模块；进阶版在基础板机型上安装了树莓派 3b+ 机载电脑，可通过 dronekit-python 编程实现对无人机的外部控制。树莓派已经安装 ubuntu mate 系统，并且已经安装好 Dronekit, ROS, mavros, 思岚 A1 驱动, Google Cartographer 等。大家不用在这些驱动包上再花大量时间。可直接上手学习。

能学到什么？

1. 无人机的基本结构以及组装一台无人机的入门知识
2. 无人机的基本操作
3. 飞控与机载电脑的通讯
4. Ubuntu 系统的基本操作
5. ROS 机器人系统入门
6. 室内无 GPS 环境下的定位悬停(搭配 T265 的升级版)

以后扩展开发方向

1. 室内无 GPS 环境下的避障导航
2. 整套系统可移植到无人车或则无人船，加思岚 A1 激光雷达实现 slam 建图
3. 利用 dronekit-python 编程实现更复杂的控制

3.1 Z410 基础版室外飞行基本操作：

3.1.1、Z410 飞行前的准备：

http://v.youku.com/v_show/id_XNDY3MjgxNDExNg==.html?x&sharefrom=android&sharekey=137370a19bac8fe70206f0d0ea3f26087

3.1.2、遥控器介绍及自稳模式飞行：

http://v.youku.com/v_show/id_XNDY3MjgwMjk0NA==.html?x&sharefrom=android&sharekey=9598e833b88a29b0cdbe40474ce8035a4

3.1.3、定高，悬停，返航模式飞行：

http://v.youku.com/v_show/id_XNDY3Mjg1MDUwMA==.html?x&sharefrom=android&sharekey=ea17a33d73174f3fc17280e6d1e8d69e8

3.1.4、降落，绕圈，有头无头及结束飞行操作：

http://v.youku.com/v_show/id_XNDY3Mjg2NjIwNA==.html?x&sharefrom=android&sharekey=52f3a12a8376b91b9f0292fe93cb48b58

3.2 Pixhawk 调试教程

[图文并茂详细教程之——用 pixhawk 飞控组装一台 S500 四轴无人机（上）](#)

[图文并茂详细教程之——用 pixhawk 飞控组装一台 S500 四轴无人机（中）](#)

[图文并茂详细教程之——用 pixhawk 飞控组装一台 S500 四轴无人机（下）](#)

4 Z410-3B+进阶版教程

Z410 进阶版搭配了树莓派 3B+,可以在树莓派上通过 dronekit-python 编程实现对无人机的控制。我们需要先配置好网络,通过远程连接到树莓派。

4.1 网络配置

方法 1:

4.1.1 用网线将树莓派连接到与操作笔记本相同的路由器上,无人机接上电池(电池插上 BB 响报警器)。

4.1.2 打开浏览器,进入路由器管理界面,查看名为: abc 的设备 IP 地址,并记录下来;

4.1.3 笔记本打开一个终端,使用下面的命令修改笔记本上的 hosts 文件:

```
sudo vim /etc/hosts
```

将 abc 的 IP 地址加入到 hosts 文件中,如下图所示。并保存退出;

如果不会用 vim 编辑器,请查看以下教程的相关内容

https://mp.weixin.qq.com/s/GekdU6_kTDcfvTBq9jVJFQ



4.1.4 打开一个终端窗口,通过 ssh 连接树莓派, password:123456

```
ssh lj@abc
```

注意: ubuntu 系统下输入密码不会显示

4.1.5 添加 wifi 连接

```
cd /etc/wpa_supplicant
```

```
sudo vim wpa_supplicant.conf
```

将你的 WIFI 网络名称和密码加入,格式如下:

```

network={

    ssid=" huawei-cj"

    psk=" 123456"

    priority=1

}

network={

    ssid=" CMCC-yAFa"

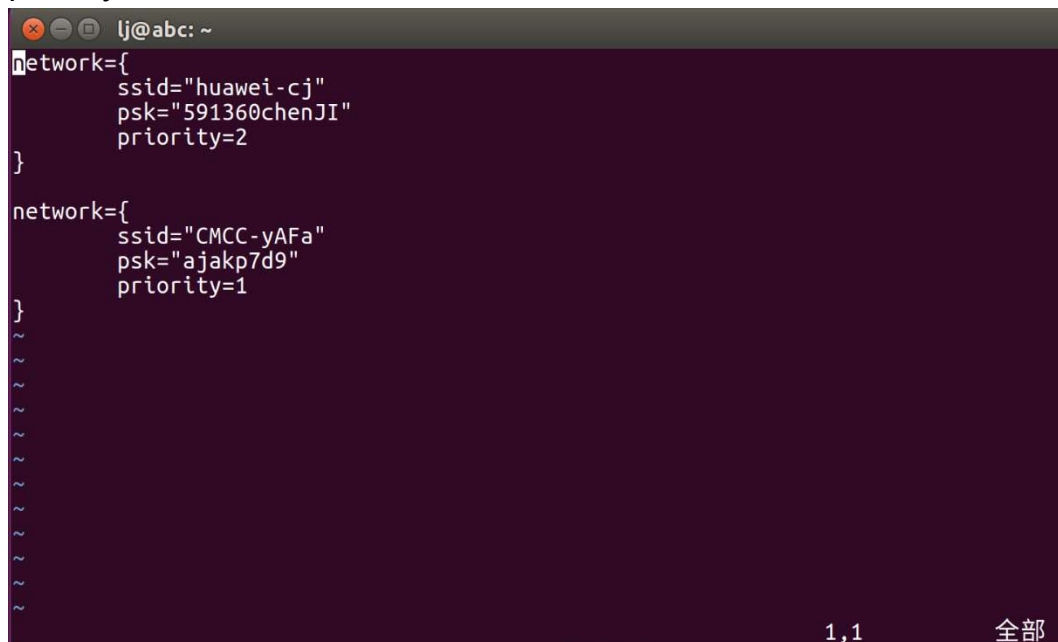
    psk=" 123456"

    priority=2

}

```

priority 数字越大表明优先级越高，开机后会自动选择优先级高的 WIFI 连接。



```

lj@abc: ~
network={
    ssid="huawei-cj"
    psk="591360chenJI"
    priority=2
}
network={
    ssid="CMCC-yAFa"
    psk="ajakp7d9"
    priority=1
}
~
~
~
~
~
~
~
~
~
~
1,1 全部

```

4.1.6 保存退出，断开网线，无人机重新上电；

4.1.7 再次进入路由器管理界面，查看使用 wifi 连接后的名为 abc 的设备的 IP 地址；

4.1.8 再次修改 hosts 文件，将 abc 以前的 IP 改为 WIFI 连接下的新的 IP 地址，保存退出；

4.1.9 这样就能使用 ssh 命令无线连接树莓派进行操作了。

方法 2:

4.2.1 将有 HDMI 接口的显示器，鼠标，键盘连接树莓派；

4.2.2 无人机接上电池，待树莓派启动后进入系统；

4.2.3 右键点击桌面，打开一个终端；

4.2.4 编辑 wpa_supplicant.conf 文件，添加 WIFI 连接，与方法 1 中的 4.1.5~4.1.9 步相同。

4.2 使用 Dronekit 测试连接飞控并读取状态数据

Dronekit 介绍:

Dronekit 也叫 DroneKit-Python，是一个用于控制无人机的 Python 库。有了它你就可以在机载电脑上通过 Python 编程实现对无人机的控制。它可以显著增强自动驾驶性能，为飞行器增加更多智能，以及执行计算密集或时间敏感的任务（例如，计算机视觉，路径规划或 3D 建模）。除了 DroneKit-Python 以外，还有 DroneKit-Android 以及 DroneKit-Cloud 的 API 供不同的开发者使用。API 通过 MAVLink 与飞控通信。它提供对连接飞控的遥测，状态和参数信息的编程访问，并实现任务管理和对飞行器运动和操作的直接控制。

相比之下，Dronekit 比搭建 ros 来控制无人机更容易上手一些。

对于 Dronekit，PX4(原生固件)被支持的较少，目前不可以进行模式切换。而对 Ardupilot(APM 固件)支持的比较多，可调用的函数也比较多。

进阶版机型我们已经在树莓派上安装好 Dronekit，并且在根目录下有一个 test 文件夹，存放了几个演示示例，大家可以根据下面的教程操作。

如果了解树莓派与 pixhawk 飞控具体如何连接，以及如何安装 dronekit 的过程，可参考以下链接有详细介绍：

第一节：<https://mp.weixin.qq.com/s/t0ur7p8Q-xN2qAHnjo2p4A>

第二节：https://mp.weixin.qq.com/s/GekdU6_kTDcfvTBq9jVJFQ

4.2.1 运行连接示例

使用 Dronekit 代码读取飞控当前状态，测试树莓派与飞控之间通讯是否正常：

4.2.1.1 无人机连接电池，笔记本开启一个终端窗口；

4.2.1.2 通过 ssh 命令，连接无人机上的树莓派；

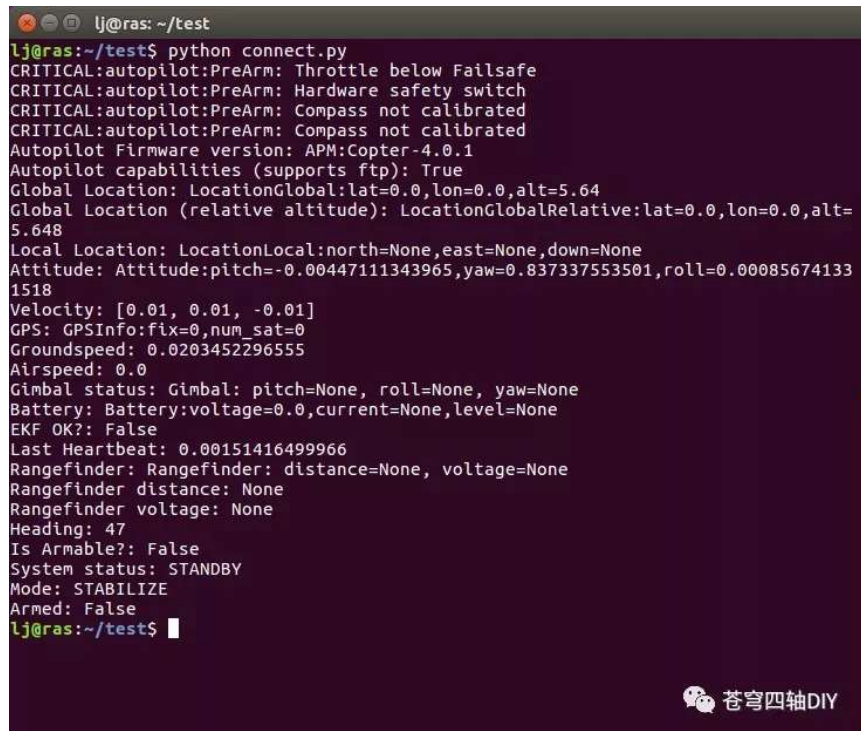
4.2.1.3 将路径切换到 test 文件夹：

```
cd test
```

4.2.1.4 运行 connect.py 脚本:

```
python connect.py
```

成功运行后,会显示如下信息:



```
lj@ras: ~/test
lj@ras:~/test$ python connect.py
CRITICAL:autopilot:PreArm: Throttle below Failsafe
CRITICAL:autopilot:PreArm: Hardware safety switch
CRITICAL:autopilot:PreArm: Compass not calibrated
CRITICAL:autopilot:PreArm: Compass not calibrated
Autopilot Firmware version: APM:Copter-4.0.1
Autopilot capabilities (supports ftp): True
Global Location: LocationGlobal:lat=0.0,lon=0.0,alt=5.64
Global Location (relative altitude): LocationGlobalRelative:lat=0.0,lon=0.0,alt=5.648
Local Location: LocationLocal:north=None,east=None,down=None
Attitude: Attitude:pitch=-0.00447111343965,yaw=0.837337553501,roll=0.000856741331518
Velocity: [0.01, 0.01, -0.01]
GPS: GPSInfo:fix=0,num_sat=0
Groundspeed: 0.0203452296555
Airspeed: 0.0
Gimbal status: Gimbal: pitch=None, roll=None, yaw=None
Battery: Battery:voltage=0.0,current=None,level=None
EKF OK?: False
Last Heartbeat: 0.00151416499966
Rangefinder: Rangefinder: distance=None, voltage=None
Rangefinder distance: None
Rangefinder voltage: None
Heading: 47
Is Armable?: False
System status: STANDBY
Mode: STABILIZE
Armed: False
lj@ras:~/test$
```

说明树莓派通过 Dronekit 读取到了目前飞控的数据: 系统警报、固件版本、姿态数据、电池电量、解锁状态、当前飞行模式等等。这样就成功运行了第一个 DroneKit-Python 脚本。

注意: 如若出现连接异常超时警报, 请检查物理连接(usb 转 TTL 模块)是否连接好; 或则给无人机重新启, 再执行以上程序。

4.2.1.5 如果想查看 connect.py 脚本, 可通过 vim 编辑器查看, 执行如下命令:

```
sudo vim connect.py
```

其中脚本注释如下:

```
# 飞控软件版本
```

```
print "Autopilot Firmware version: %s" % vehicle.version
```

```
# 全球定位信息 (经纬度, 高度相对于平均海平面)
```

```
print "Global Location: %s" % vehicle.location.global_frame
```


全球定位信息（经纬度，高度相对于 home 点）

```
print "Global Location (relative  
altitude): %s" % vehicle.location.global_relative_frame
```

相对 home 点的位置信息（向北、向东、向下）；解锁之前返回 None

```
print "Local Location: %s" % vehicle.location.local_frame
```

无人机朝向（欧拉角：roll, pitch, yaw, 单位为 rad, 范围 $-\pi$ 到 $+\pi$ ）

```
print "Attitude: %s" % vehicle.attitude
```

三维速度（m/s）

```
print "Velocity: %s" % vehicle.velocity
```

GPS 信息

```
print "GPS: %s" % vehicle.gps_0
```

地速（m/s）

```
print "Groundspeed: %s" % vehicle.groundspeed
```

空速（m/s）

```
print "Airspeed: %s" % vehicle.airspeed
```

云台信息（得到的为当前目标的 roll, pitch, yaw, 而非测量值。单位为度）

```
print "Gimbal status: %s" % vehicle.gimbal
```

电池信息

```
print "Battery: %s" % vehicle.battery
```

EKF（拓展卡曼滤波器）状态

```
print "EKF OK?: %s" % vehicle.ekf_ok
```

超声波或激光雷达传感器状态

```
print "Rangefinder: %s" % vehicle.rangefinder
```

无人机朝向（度）

```
print "Heading: %s" % vehicle.heading
```

是否可以解锁

```
print "Is Armable?: %s" % vehicle.is_armable
```

系统状态

```
print "System status: %s" % vehicle.system_status.state
```

当前飞行模式

```
print "Mode: %s" % vehicle.mode.name
```

解锁状态

```
print "Armed: %s" % vehicle.armed
```

4.3 启动 SITL 仿真环境并运行 python 示例

在 test 文件夹下我们几个演示脚本，如果单纯的运行脚本新手没有直观的感受。我们可以结合无人机仿真环境，来学习脚本具体的执行过程。

前面安装好 ubuntu 系统镜像，里面已经搭建好了 ardupilot 开发环境。SITL 仿真也就包含其中。

需要注意：DroneKit-SITL 目前仅为 Mac, Linux 和 Windows 提供 x86 二进制文件。不能在树莓派这样的 ARM 平台上运行它。

我们执行的 python 脚本是在树莓派系统上，而 SITL 仿真环境是搭建在笔记本系统上的。

4.3.1 树莓派端，进入 test 目录

```
cd test
```

4.3.2 编辑 example2.py

```
sudo vim example2.py
```

将其中 connection_string = '192.168.1.7:14550' IP 地址改为你的树莓派 IP

脚本里面都配有详细注解，大家可根据注解来理解 dronekit 的基本使用。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
© Copyright 2015-2016, 3D Robotics.
simple_goto.py: GUIDED mode "simple goto" example (Copter Only)
Demonstrates how to arm and takeoff in Copter and how to navigate to points using Vehicle.simple_goto.
"""

from __future__ import print_function
import time
from dronekit import connect, VehicleMode, LocationGlobalRelative

# 本示例在SITL模拟演示, 连接地址改为树莓派ip, 端口与SITL输出端口一致.
connection_string = '192.168.1.7:14550'
print('Connecting to vehicle on: %s' % connection_string)
# connect函数将会返回一个Vehicle类型的对象, 即此处的vehicle
# 即可认为是无人机的主体, 通过vehicle对象, 我们可以直接控制无人机
vehicle = connect(connection_string, wait_ready=True)

# 定义arm_and_takeoff函数, 使无人机解锁并起飞到目标高度
# 参数aTargetAltitude即为目标高度, 单位为米
def arm_and_takeoff(aTargetAltitude):
    # 进行起飞前检查
    print("Basic pre-arm checks")
    # vehicle.is_armable会检查飞控是否启动完成、有无GPS fix、卡曼滤波器
    # 是否初始化完毕。若以上检查通过, 则会返回True
    while not vehicle.is_armable:
        print("Waiting for vehicle to initialise...")
        time.sleep(1)

    # 解锁无人机 (电机将开始旋转)
    print("Arming motors")
    # 将无人机的飞行模式切换成"GUIDED" (一般建议在GUIDED模式下控制无人机)
    vehicle.mode = VehicleMode("GUIDED")
    print("example2.py" 100L, 4433C)
```

4.3.3 笔记本端, 首先要进入需要仿真的多旋翼无人机的目录下:

```
cd ardupilot/ArduCopter
```

第一次运行, 需要执行下面的命令对仿真环境进行初始化

```
sim_vehicle.py -w
```

或则是仿真参数被改的乱七八糟的时候, 也可用这个命令恢复初始参数。

启动完毕, 使用 Ctrl+C 终止正在运行的 sim_vehicle.py -w

4.3.4 接下来就可以启动模拟器了:

```
sim_vehicle.py --console --map
```

正常启动后, 就会看到三个窗口: Terminal, Console, Map, 这样最基本的软件在环仿真程序就运行起来了。

4.3.5 启动后通过 output 命令列出 MAVProxy 转发数据的接口, 如下所示:

```

arducopter"
Waf: Entering directory `/home/cj/ardupilot/build/sitl'
Embedding file sandbox.lua:libraries/AP_Scripting/scripts/sandbox.lua
Waf: Leaving directory `/home/cj/ardupilot/build/sitl'

BUILD SUMMARY
Build directory: /home/cj/ardupilot/build/sitl
Target      Text      Data      BSS      Total
-----
bin/arducopter 2349734 2952 73672 2426358

Build commands will be stored in build/sitl/compile_commands.json
'build' finished successfully (4.895s)
SIM_VEHICLE: Using defaults from (/home/cj/ardupilot/Tools/autotest/default_params/copter.parm)
SIM_VEHICLE: Run ArduCopter
SIM_VEHICLE: "/home/cj/ardupilot/Tools/autotest/run_in_terminal_window.sh" "ArduCopter" "/home/cj/ardupilot/build/sitl/bin/arducopter" "-S" "-I0" "--model" "+" "--speedup" "1" "--defaults" "/home/cj/ardupilot/Tools/autotest/default_params/copter.parm"
SIM_VEHICLE: Run MavProxy
SIM_VEHICLE: "mavproxy.py" "--master" "tcp:127.0.0.1:5760" "--sitl" "127.0.0.1:501" "--out" "127.0.0.1:14550" "--out" "127.0.0.1:14551" "--map" "--console"
RtW: Starting ArduCopter : /home/cj/ardupilot/build/sitl/bin/arducopter -S -I0 --model + --speedup 1 --defaults /home/cj/ardupilot/Tools/autotest/default_params/copter.parm
Connect tcp:127.0.0.1:5760 source_system=255
Loaded module console
Loaded module map
Log Directory:
Telemetry log: mav.tlog
MAV> Waiting for heartbeat from tcp:127.0.0.1:5760
STABILIZE> Received 1178 parameters
Saved 1178 parameters to mav.parm
output
STABILIZE> 2 outputs
0: 127.0.0.1:14550
1: 127.0.0.1:14551
STABILIZE>

```

4.3.6 使用 output add 命令增加树莓派 IP 接口：

output add 192.168.1.7:14550

其中地址为**你的树莓派的 IP**，与 example2.py 代码中地址一致。

```

Embedding file sandbox.lua:libraries/AP_Scripting/scripts/sandbox.lua
Waf: Leaving directory `/home/cj/ardupilot/build/sitl'

BUILD SUMMARY
Build directory: /home/cj/ardupilot/build/sitl
Target      Text      Data      BSS      Total
-----
bin/arducopter 2349734 2952 73672 2426358

Build commands will be stored in build/sitl/compile_commands.json
'build' finished successfully (4.895s)
SIM_VEHICLE: Using defaults from (/home/cj/ardupilot/Tools/autotest/default_params/copter.parm)
SIM_VEHICLE: Run ArduCopter
SIM_VEHICLE: "/home/cj/ardupilot/Tools/autotest/run_in_terminal_window.sh" "ArduCopter" "/home/cj/ardupilot/build/sitl/bin/arducopter" "-S" "-I0" "--model" "+" "--speedup" "1" "--defaults" "/home/cj/ardupilot/Tools/autotest/default_params/copter.parm"
SIM_VEHICLE: Run MavProxy
SIM_VEHICLE: "mavproxy.py" "--master" "tcp:127.0.0.1:5760" "--sitl" "127.0.0.1:501" "--out" "127.0.0.1:14550" "--out" "127.0.0.1:14551" "--map" "--console"
RtW: Starting ArduCopter : /home/cj/ardupilot/build/sitl/bin/arducopter -S -I0 --model + --speedup 1 --defaults /home/cj/ardupilot/Tools/autotest/default_params/copter.parm
Connect tcp:127.0.0.1:5760 source_system=255
Loaded module console
Loaded module map
Log Directory:
Telemetry log: mav.tlog
MAV> Waiting for heartbeat from tcp:127.0.0.1:5760
STABILIZE> Received 1178 parameters
Saved 1178 parameters to mav.parm
output
STABILIZE> 2 outputs
0: 127.0.0.1:14550
1: 127.0.0.1:14551
STABILIZE> output add 192.168.1.7:14550
STABILIZE> Adding output 192.168.1.7:14550

```

4.3.7 树莓派端执行：

运行 example2.py 代码

```
python example2.py
```

视频演示：https://v.youku.com/v_show/id_XNDY2ODYxMTE2OA==.html

4.3.8 我们还可以将航点文件导入 SITL 进行演示，具体参考如下视频

https://v.youku.com/v_show/id_XNDY2ODYxMjgzNg==.html

4.4 SITL 仿真环境环境参数设置

在前面教程的示例中，我们使用的是仿真环境的默认参数。如果我们在实际使用中加入了光流，或则禁用了 GPS，那么如何在仿真环境中做相应的设置呢？

4.4.1 在启动 SITL 仿真环境的终端里，使用命令行对参数进行修改，比如：

4.4.1.1 查看参数

查看参数的命令为

```
param show PARAM_NAME
```

使用任意参数名代替 PARAM_NAME 即可。

比如：显示全部参数

```
param show *
```

显示所有以“ EK2” 开头的参数

```
param show EK2*
```

4.4.1.2 修改参数

修改参数的命令为

```
param set PARAM_NAME VALUE
```

其中 PARAM_NAME 代表参数名，VALUE 代表数值

比如：使用以下命令禁用飞行前检查(Arming Check)

```
param set ARMING_CHECK 0
```

修改的参数会自动保存，下一次启动时不必再次修改。

比如：

禁用 GPS，设置虚拟光流和虚拟超声波

```
param set SIM_GPS_DISABLE 0
```

设置虚拟光流传感器


```
param set SIM_FLOW_ENABLE 1
```

启用光流传感器

```
param set FLOW_ENABLE 1
```

设置光流传感器参数

```
param set FLOW_FXSCALER 50
```

```
param set FLOW_FYSCALER 50
```

具体可参考：<https://ardupilot.org/dev/docs/using-sitl-for-ardupilot-testing.html>

4.4.2 如果 SITL 与 MP 地面站结合运行，那么我们可以直接在 MP 地面站的全部参数列表里，查找需要修改的参数进行修改。SITL 与 MP 地面站结合运行在下节有详细介绍。

4.5 SITL 仿真结合 MP 地面站运行 python 示例

首先，将树莓派无人机接上电池，确保笔记本，MP 地面站电脑和树莓派都在同一个局域网内，然后使用 ssh 命令连接树莓派。

笔记本端：

启动 SITL 仿真

```
cd ~/ardupilot/ArduCopter
```

```
sim_vehicle.py --console --map
```

使用 output add 命令，增加树莓派的 IP：

```
output add 192.168.1.7: 14550
```

再使用 output add 命令，增加 MP 地面站的 IP：

```
output add 192.168.1.5: 14551
```

树莓派端：

运行示例代码：

```
cd test
```

```
python example2.py
```

以下视频演示 SITL 仿真如何结合 MP 地面站运行

https://v.youku.com/v_show/id_XNDY2ODYxOTk2OA==.html

4.6 Python 脚本控制无人机飞行

根据 dronekit 的官方示例，我们编写了几个简单的程序控制无人机飞行。

编写完以后先在 SITL 仿真环境下运行看看是否和预想的一致。没有问题后，再连接无人机实地飞行。

example5.py 控制无人机起飞到 3 米高度，然后悬停 5 秒，再自动降落；

大家可根据我们提供的示例，结合自己的实际情况，编写控制程序。

Dronekit 室外控制无人机起飞降落演示：

https://v.youku.com/v_show/id_XNDY5NzA5OTQ2OA==.html

example4.py 在室外空旷处，此脚本控制无人机起飞到 3 米高度，然后飞出一个 2 米*2 米的正方形，再自动降落。

Dronekit 室外控制无人机飞出一个正方形演示：

https://v.youku.com/v_show/id_XNDY5NzEwMzExNg==.html

还有值得注意的是，python 脚本中设置属性：

少数的属性变量可以被设置，通过设置这些属性变量，可以控制无人机的运行状态。可设置的属性变量如下：

```
vehicle.home_location
```

```
vehicle.gimbal
```

```
vehicle.airspeed
```

```
vehicle.groundspeed
```

```
vehicle.mode
```

```
vehicle.armed
```

```
vehicle.disarmed
```

设置示例：

1. 锁定无人机：

```
vehicle.disarmed = False
```

2. 切换到 GUIDED 模式：

```
vehicle.mode = VehicleMode("GUIDED")
```

3. 设置航点模式下，无人机飞行的地速为 3.2m/s（注意：读、写 groundspeed 的意义不同）

```
print "Param: %s" % vehicle.parameters['THR_MIN']

vehicle.groundspeed = 3.2
```

读取参数

参数以字典（dictionary）的形式，存储在 vehicle.parameters 变量中。具体参数的名称即为相应的键值（key）。

例如，在屏幕上显示 THR_MIN 参数（THR_MIN 代表油门处于最低时的电机怠速，以 PWM 值表示）：

```
print "Param: %s" % vehicle.parameters['THR_MIN']
```

显示全部参数：

```
print "\nPrint all parameters (iterate `vehicle.parameters`):"

for key, value in vehicle.parameters.iteritems():

    print " Key:%s Value:%s" % (key,value)
```

设置参数

使用读取参数类似的方法，即可设置参数：

```
vehicle.parameters['THR_MIN']=100
```

以上只是截取了部分使用较多的知识点进行介绍，完整的教程可以到官网查看。

关于 dronekit 的一些参考资料：

有关项目文档，请访问 <https://readthedocs.org/projects/dronekit-python/>。这包括指南，示例和 API 参考资料。

示例源代码托管在 Github 上，作为 [/ dronekit-python / examples](#) 的子文件夹。

[DroneKit 论坛](#)是寻求有关如何使用该库的技术支持的最佳场所。也可以查看 [Gitter channel](#)。

说明文件： <https://dronekit-python.readthedocs.io/en/latest/about/index.html>

指南： <https://dronekit-python.readthedocs.io/en/latest/guide/index.html>

API 参考： <https://dronekit-python.readthedocs.io/en/latest/automodule.html>

例子： [/dronekit-python/examples](#)

论坛： <https://github.com/dronekit/dronekit-python/issues>

Gitter: <https://gitter.im/dronekit/dronekit-python>

5 Z410 视觉版教程

注意：默认情况下，无人机 pixhawk 飞控是使用 GPS 在室外定位。如果要使用 T265 双目相机进行室内定位，飞控参数需要进行如下设置：

把飞控连接地面站，点击 **配置调试---全部参数列表**：

AHRS_EKF_TYPE = 2 (默认使用 EKF2)

EK2_ENABLE = 1 (默认)

EK3_ENABLE = 0 (默认)

GPS_TYPE = 0 (不启用 GPS)

EK2_GPS_TYPE = 3(不启用 EKF 使用 GPS)

EK2_POSNE_M_NSE = 0.1

EK2_VELD_M_NSE = 0.1

EK2_VELNE_M_NSE = 0.1

COMPASS_USE = 0, **COMPASS_USE2** = 0, **COMPASS_USE3** = 0 (禁止 EKF

使用指南针，而是依靠外部导航数据的航向)

RTL_ALT=0 室内测试请将返航高度设置为0，让飞行器以当前高度返航。默认是1500mm。

(以下是飞控默认数据，要使用 GPS 定位改回来就可以了。)

```
AHRS_EKF_TYPE = 2 (the default)

EK2_ENABLE = 1 (the default)

EK3_ENABLE = 0 (the default)

GPS_TYPE = 1

EK2_GPS_TYPE = 0

EK2_POSNE_M_NSE = 1

EK2_VELD_M_NSE = 0.7

EK2_VELNE_M_NSE = 0.5

COMPASS_USE = 1, COMPASS_USE2 = 0, COMPASS_USE3 = 0
```

本机可搭载 Intel T265 双目相机，在室内没有 GPS 信号的情况下，代替 GPS 导航定位。双目定位后，如果室内空间足够，就可以进行悬停飞行，一键返航，失控返航，航点飞行，dronekit 编程控制飞行等以前只能在室外 GPS 定位后进行的操作。从而为以后室内测试更复杂的避障，导航规划等项目奠定基础。

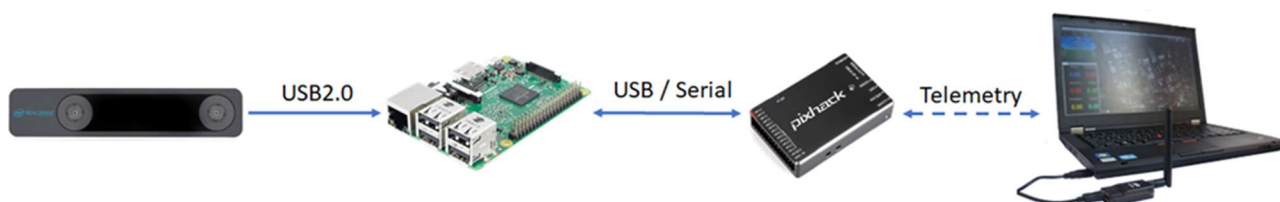
T265 在树莓派上有 2 种启动方式：

5.1 使用 pyrealsense2 启动双目相机

使用 pyrealsense2 定位原理：将从 Realsense T265 获得的 6 自由度姿态数据（位置和方向）和置信度数据交由 python 脚本处理(该脚本位于 vision_to_mavros/scripts/t265_to_mavlink.py)，处理后的结果通过 MAVLink 发送到 ArduPilot 无人机，从而实现定位。

该脚本将执行以下任务：

- 使用 pyrealsense2 的相关 API 获取 6 自由度姿势数据并跟踪置信度数据，pyrealsense2 是 librealsense 的 Python 包装器；
- 执行必要的矩阵变换，以匹配 Realsense T265 和 NED 的框架以及其他处理步骤；
- 将姿态数据打包到 MAVLink 消息 VISION_POSITION_ESTIMATE 中，并将置信度数据打包到虚拟消息中，然后将它们以预定频率发送到 ArduPilot，以免输入数据淹没飞控；
- 自动设置 EKF home 点，让设置和飞行都更为简单；



飞行器机载电脑已经安装好 librealsense 和 pyrealsense2，根据以下步骤即可启动 T265

5.1.1 无人机接上电池，打开 missionplanner 地面站，使用数传连接地面站和飞控；

5.1.2 使用 ssh 远程连接树莓派，操作和之前一样；

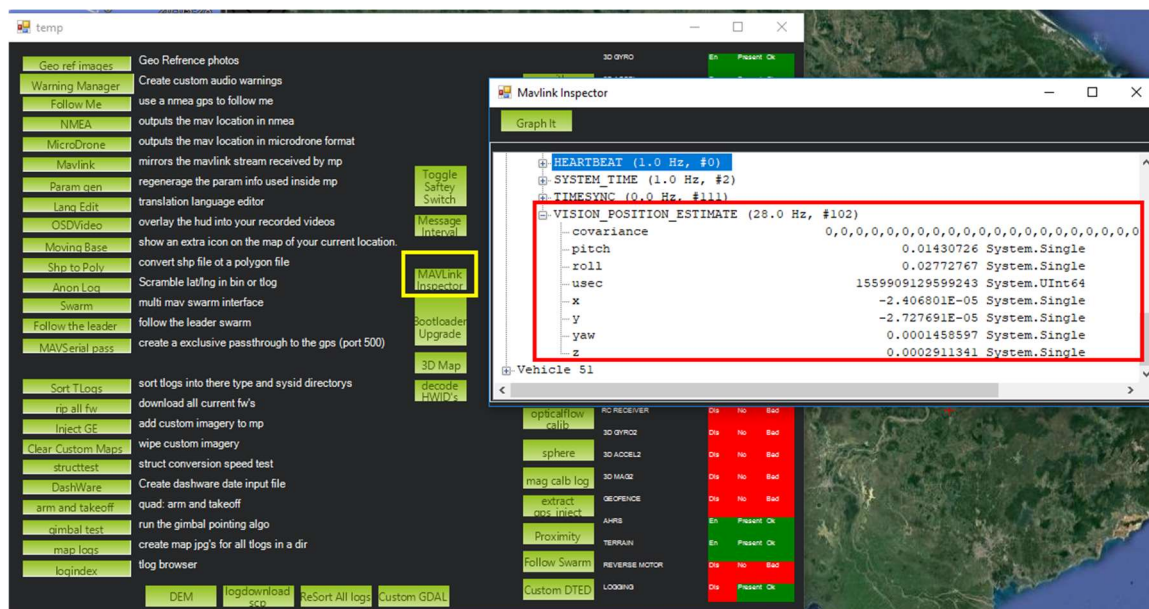
5.1.3 打开终端，切换路径到脚本所在目录：

```
cd ~/vision_to_mavros/catkin_ws/src/vision_to_mavros/scripts
```

5.1.4 执行下面脚本：

```
python3 t265_to_mavlink.py
```

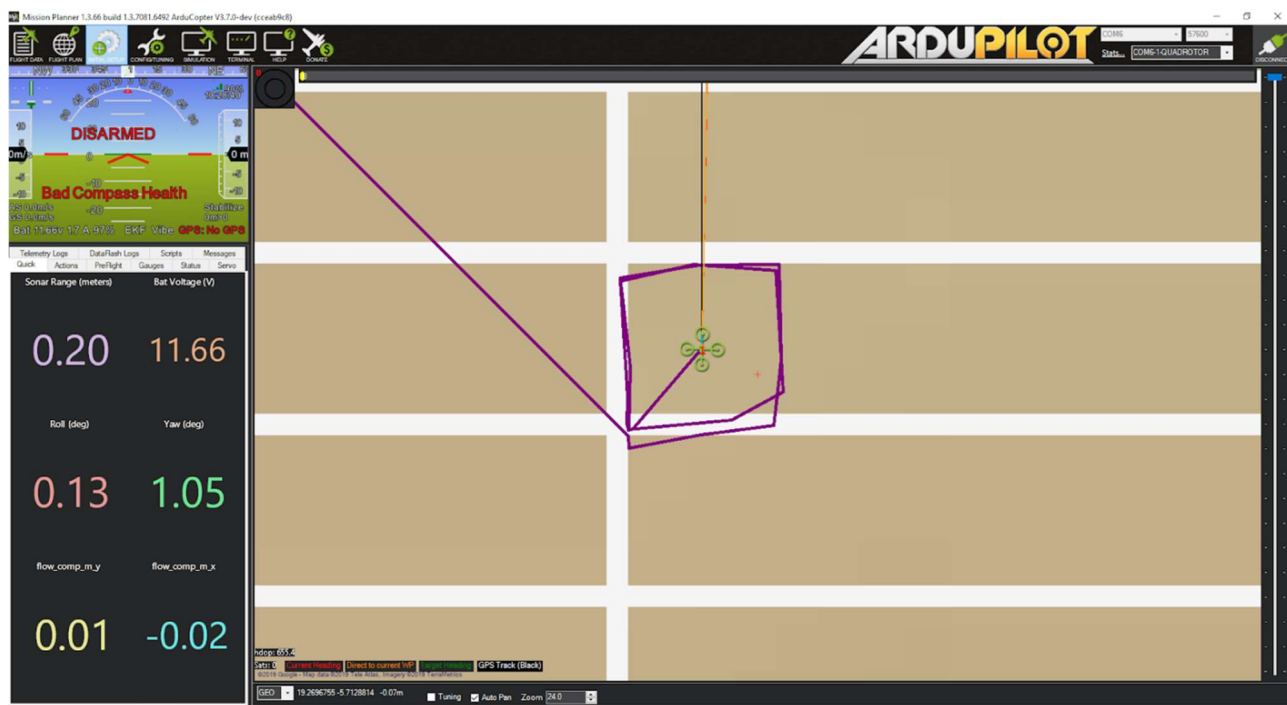
5.1.5 地面站软件，快捷键 *CTR+F*，点击 *Mavlink Inspector*，能看到数据已经上传到飞控了。



同时 usb 转 ttl 模块上的 rx 和 tx 指示灯会快速闪烁，表示有数据传输；

5.1.6 稍等片刻直到四轴飞行器图标出现在 Mission Planner 的地图上；

5.1.7 拿起无人机并四处走动，检查地图上是否显示了无人机的位置运动。地图上显示的无人机的轨迹应该反映真实的运动，而不会产生过多的失真或超调。以下是在 2m x 2m 的正方形中行走的示例。



5.1.8 如果外部定位导航数据由于任何原因丢失（跟踪丢失，脚本中断等），**重新运行脚本也不会成功定位**，需要重新启动无人机(重新上电)并执行之前的操作。

5.1.9 飞行测试：(请有无人机操作经验的人操作)

- 在自稳或则定高模式下解锁起飞，检查无人机飞行是否平稳；
- 使用遥控器操作无人机四处移动，并观察 Mission Planner 上的位置是否稳定；
- 以上没有问题，保持无人机在 0.8~1 米左右高度，油门 50% 左右，切换到 Loiter 模式，但是如果出现任何问题，请随时准备切换回 Stabilize (自稳)/ Alt-Hold(定高)；
- 正常情况下，无人机应该稳定悬停在空中；
- 遥控器操作无人机四处缓慢缓慢移动，查看无人机稳定状态。测试时注意始终准备切换回“稳定/保持”状态；
- 如果一切正常，下次就可以在 Loiter 模式下解锁和起飞。

注意：请确保双目相机水平向前安装，测试环境光照充足，双目前方特征明显的视觉范围测试。

5.2 数据查看：

5.2.1 视觉测距信息将出现在 VISO 数据闪存日志消息中。

5.2.2 EKF 的视觉测距信息将显示在 XKFD 消息中。

5.3 设置启动时自动运行

以上飞行操作熟悉后，可以进行以下设置，在无人机通电后机载电脑自动启动 T265，不再进行人为操作。

5.3.1 切换到脚本目录

```
cd ~/vision_to_mavros/catkin_ws/src/vision_to_mavros/scripts
```

5.3.2 按照如下修改 t265.sh 文件里 t265_to_mavlink.py 的路径

```
sudo vim t265.sh  
  
~/vision_to_mavros/catkin_ws/src/vision_to_mavros/scripts/  
t265_to_mavlink.py
```

5.3.3 获取权限

```
chmod +x /path/to/t265.sh
```

5.3.4 测试执行看看能否启动：

```
./t265.sh
```

5.3.5 新建一个 t265.service 文件

```
cd /etc/systemd/system  
  
sudo vim t265.service
```

复制如下内容：

[Unit]

Description=Realsense T265 Service

After==multi-user.target

StartLimitIntervalSec=0

Conflicts=

[Service]

User=lj

EnvironmentFile=

ExecStartPre=

ExecStart=

~/vision_to_mavros/catkin_ws/src/vision_to_mavros/scripts/t265.sh

Restart=on-failure

RestartSec=1

[Install]

WantedBy=multi-user.target

5.3.6 以上文件保存退出，执行下面命令：

```
systemctl start t265
```

```
systemctl enable t265
```

下次无人机启动后，T265 就会自动启动。室内定位运行成功，切换到 loiter 悬停模式，指示灯是蓝色，表示可以解锁；如果切换到 loiter 变成黄灯，表示室内定位运行失败，请重新检查再启动。

5.2 ROS 系统下启用 T265

ROS 系统下启用 T265 需要先配置主从机。

主从机设置

如果是通过 ROS 启动 T265 双目相机，就需要先进行主从机设置。

笔记本从机设置：

5.2.1 笔记本端修改 hosts 文件

```
sudo vim /etc/hosts
```

增加你的树莓派 IP，格式如下：

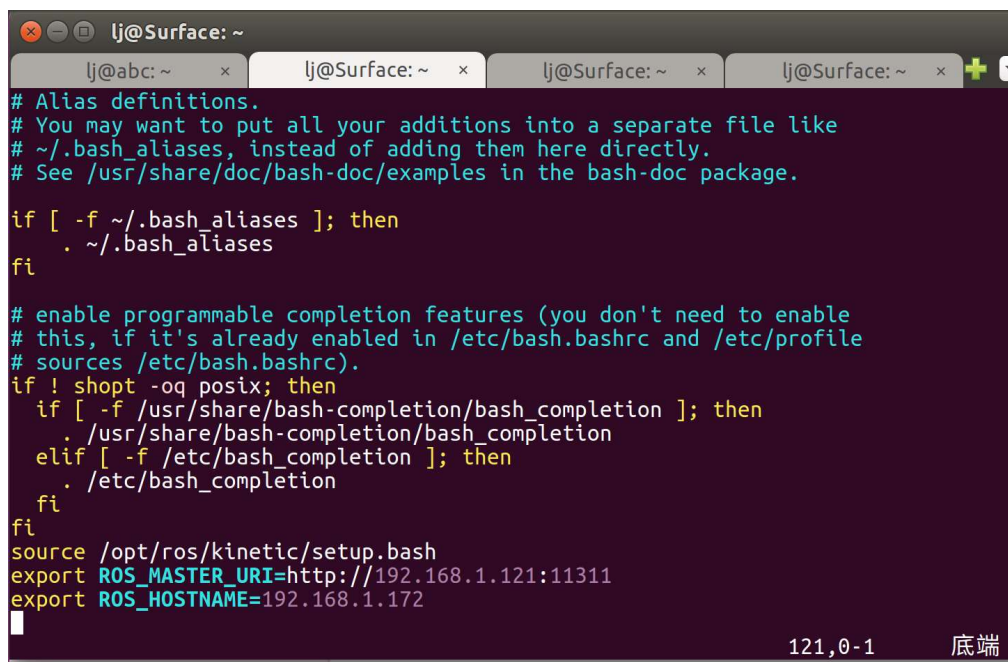
```
192.168.1.121    abc
```

5.2.2 笔记本端修改 bashrc 文件

```
sudo vim ~/.bashrc
```

在最后两行，将 ROS_MASTER_URI 修改为树莓派 IP

将 ROS_HOSTNAME 修改为笔记本 IP



```

lj@Surface: ~
lj@abc: ~ x  lj@Surface: ~ x  lj@Surface: ~ x  lj@Surface: ~ x +
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi
source /opt/ros/kinetic/setup.bash
export ROS_MASTER_URI=http://192.168.1.121:11311
export ROS_HOSTNAME=192.168.1.172
121,0-1 底端

```

5.2.3 保存退出。修改后记得 source ~/.bashrc

树莓派主机设置：

5.2.4 树莓派修改 hosts 文件

```
sudo vim /etc/hosts
```

增加你的笔记本的 IP 和笔记本名称，格式如下：

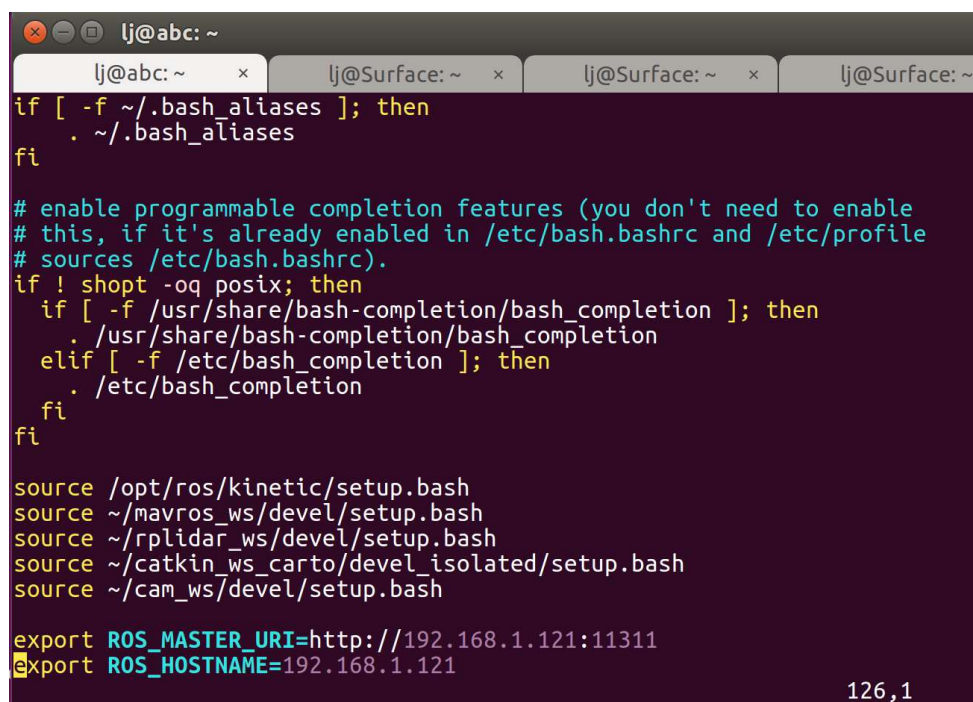
```
192.168.1.172      surface
```

5.2.5 树莓派端修改 hosts 文件

文末修改 IP 为你的树莓派的 IP：

```
export ROS_MASTER_URI=http://192.168.1.121:11311
```

```
export ROS_HOSTNAME=192.168.1.121
```



```

lj@abc: ~
lj@abc: ~ x  lj@Surface: ~ x  lj@Surface: ~ x  lj@Surface: ~
if [ -f ~/.bash_aliases ]; then
. ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
if [ -f /usr/share/bash-completion/bash_completion ]; then
. /usr/share/bash-completion/bash_completion
elif [ -f /etc/bash_completion ]; then
. /etc/bash_completion
fi
fi

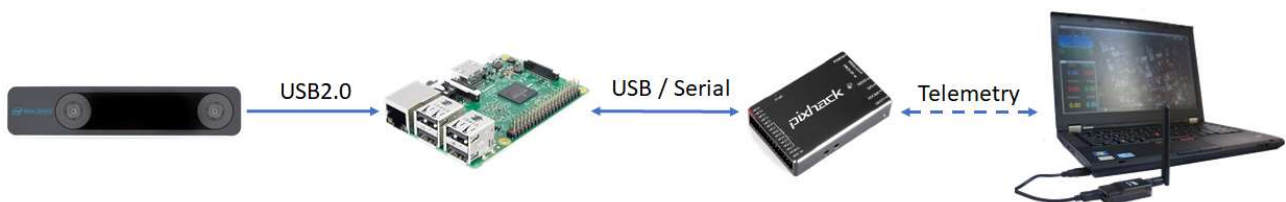
source /opt/ros/kinetic/setup.bash
source ~/mavros_ws/devel/setup.bash
source ~/rplidar_ws/devel/setup.bash
source ~/catkin_ws_carto/devel_isolated/setup.bash
source ~/cam_ws/devel/setup.bash

export ROS_MASTER_URI=http://192.168.1.121:11311
export ROS_HOSTNAME=192.168.1.121
126,1

```

5.2.6 保存退出，记得 source ~/.bashrc

ROS 下 双目相机 T265 的定位原理：从 realsense-ros 节点获得的位置数据由 vision_to_mavros 节点处理，并通过主题/ mavros / vision_pose / pose 发送到 mavros 节点。mavros 将负责 ENU-NED 帧转换，并通过 MAVLink 将其发送到 ArduPilot 无人机，从而实现室内定位。



本机已经安装好 librealsense、realsense-ros、mavros、vision_to_mavros 等必要的组件，根据以下教程即可启动 T265 定位。

1.1 主从机设置完成

1.2 无人机接上电池，打开 missionplanner 地面站，使用数传连接地面站和飞控；

1.3 使用 ssh 远程连接树莓派；

1.4 开启一个终端，运行 realsense-ros 节点

```
roslaunch realsense2_camera rs_t265.launch
```

/camera/odom/sample/ 和/tf 主题将以 200Hz 频率发布。

1.5 开启另一个终端，运行 mavros 节点

```
roslaunch mavros apm.launch
```

rostopic echo /mavros/vision_pose/pose 未发布。

1.6 开启第三个终端，运行 vision_to_mavros 节点

```
roslaunch vision_to_mavros t265_tf_to_mavros.launch
```

rostopic echo /mavros/vision_pose/pose 现在应该显示来自 T265 的姿态数据。

rostopic hz /mavros/vision_pose/pose 应该显示该主题以 30Hz 的频率发布。

以上 3 个节点运行成功，并且 FCU 开始接收 VISION_POSITION_ESTIMATE 消息，就看到地面站会消息框会显示 “ **GPS Glitch** ” 和 “ **GPS Glitch cleared** ”，确认系统已识别了外部本地化数据。

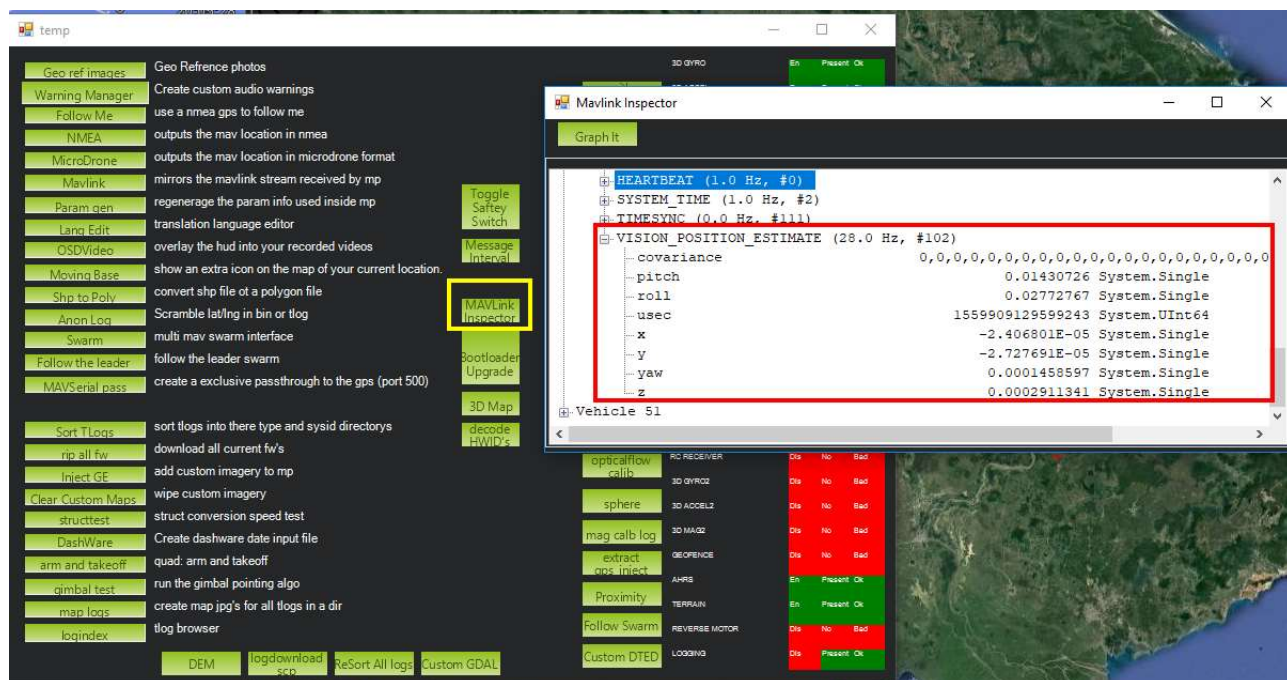
1.7 设置 EKF HOME 点

1.6.1 使用 Mission Planner: 右键单击地图上的任意点 > Set Home Here > Set EKF Origin Here.

1.6.2 使用脚本设置 EKF HOME 点，开启第四个终端，运行：

```
roslaunch vision_to_mavros set_origin.py
```

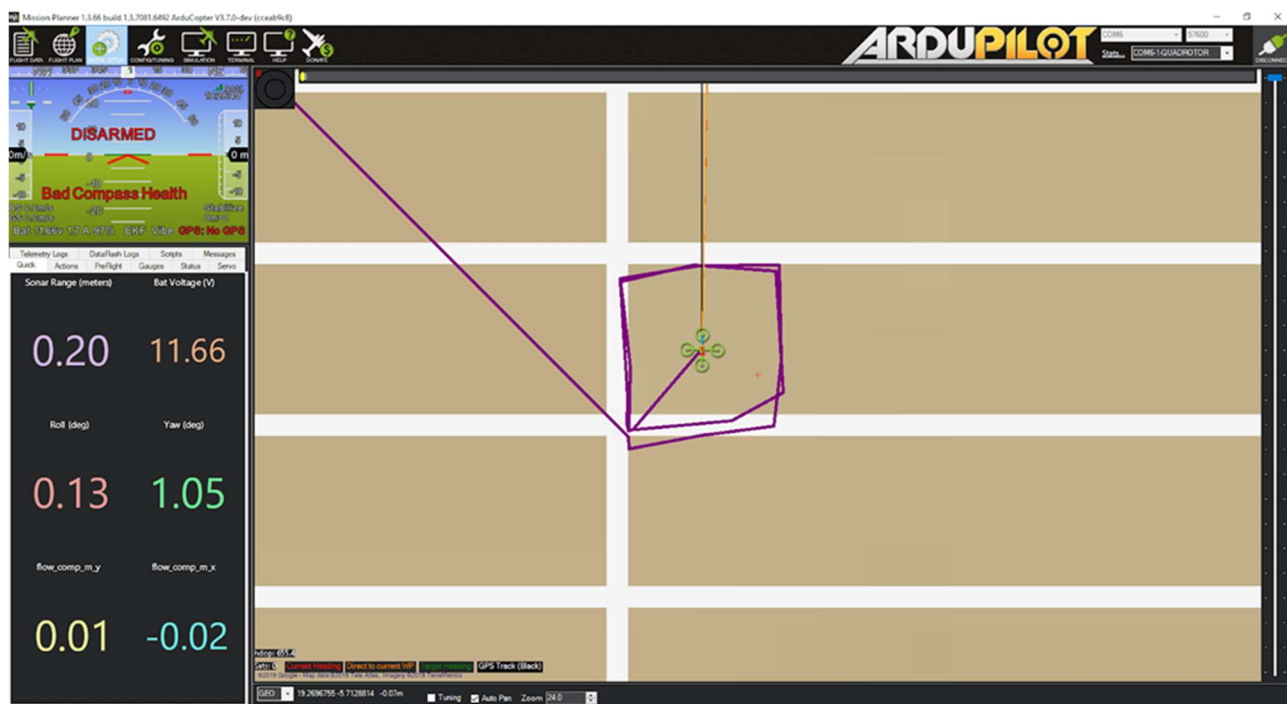
1.8 打开地面站软件，快捷键 CTR+F，点击 Mavlink Inspector，能看到数据已经上传到飞控了。室内定位运行成功，切换到 loiter 悬停模式，指示灯是蓝色，表示可以解锁；如果切换到 loiter 变成黄灯，表示室内定位运行失败，请重新检查再启动。



同时 usb 转 ttl 模块上的 rx 和 tx 指示灯会快速闪烁，表示有数据传输；

1.9 稍等片刻直到四轴飞行器图标出现在 Mission Planner 的地图上；

2.0 拿起无人机并四处走动，检查地图上是否显示了无人机的位置运动。地图上显示的无人机的轨迹应该反映真实的运动，而不会产生过多的失真或超调。以下是在 2m x 2m 的正方形中行走的示例



2.1 飞行测试: (请有无人机操作经验的人操作)

2.1.1 在自稳或则定高模式下解锁起飞, 检查无人机飞行是否平稳;

2.1.2 使用遥控器操作无人机四处移动, 并观察 Mission Planner 和 rviz 上的位置, 以查看跟踪是否稳定;

2.1.3 以上没有问题, 保持无人机在 0.8~1 米左右高度, 油门 50% 左右, 切换到 Loiter, 但是如果出现任何问题, 请随时准备切换回 Stabilize / Alt-Hold。

2.1.4 正常情况下, 无人机应稳定悬停并能够保持其位置。使用遥控器缓慢移动无人机 2-3 米, 验证比例 (便于在 rviz 上查看)

2.1.5 如果一切正常, 下次就可以在 Loiter 模式下解锁和起飞。

注意: 在切换到 Loiter 模式之前, 请始终确认位置反馈运行正常。注意环境中的工作边界, 即由于缺少功能, 请勿快速移动或旋转运动, 可能丢失跟踪定位。

2.1.6 如果外部定位导航数据由于任何原因丢失 (跟踪丢失, 脚本中断等), **重新运行脚本也不会成功定位**, 需要重新启动无人机(重新上电)并执行之前的操作。

2.2 数据查看:

视觉测距信息将出现在 VISO 数据闪存日志消息中。

演示视频:

https://v.youku.com/v_show/id_XNDY2ODM1ODYxNg==.html

https://v.youku.com/v_show/id_XNDY1NjMxOTQ4OA==.html

成功运行以上节点, 下次可以只需要开启一个终端, 使用下面这个命令一次启动所有节点:

```
roslaunch vision_to_mavros t265_all_nodes.launch
```

此命令一次执行 3 个 launch: rs_t265.launch,
apm.launch,t265_tf_to_mavros.launch

这样可避免开启多个终端, 简化操作, 方便飞行。

再使用脚本设置 EKF HOME 点:

```
roslaunch vision_to_mavros set_origin.py
```

6 后续可开发更多功能

以上教程是 Z410 的一些基本操作。大家可以利用 ubuntu 系统与 ROS 系统的开源特性, 在此基础上进行扩展, 比如避障, 跟踪, 导航等方面。大家也可以加入到 Z410 技术讨论微信群, 分享自己在开发方面的心得、方法、示例和遇到的问题进行讨论。