the daily recoding

fengxuewei

August 7, 2020

Contents

第一章　七月份

1.1 第一周

1.1.1 第一天(20200731)

继续调节debug(vf一直计算的姿态期望值是固定的 $roll|yaw = -1.56339$)

- 拿到输入，相同的输入, 就需要去比对当前的值和msgsSubAndPub的值能否是实时的?
- 如果我的输出和昌伟的输出结果都是一样的，那么就会是输入值的问题 ==> 输出是相同的
- 相同的输出: 给到px4内部现象是什么样子的呢? ==> 相反的情况

感觉情况像是发送进来的rc指令为空. 测试:

- 只是发送一个空的值, 使能ALTCTL模式 ==> 单独使用节点进入定高模式 ==> 沿着切线方向去飞
- 拿标准版的飞控进行接收数据.

第二章　八月份

2.1 第一周

2.1.1 20200801

再次查找执行情况不对的原因, 在软件在环下面: 执行下面的情况

- $yaw == 0|1.56339$, 正西
- $roll == 0$: 斜上方

重新刷了一次固件(硬件用的是第一版), 现象可以了, 但是没有办法通过树莓派进入定高模式, 需要通过手动来切换(这个是偶然现象)

为了排查根本原因, 进行了下面的排查:

1). SD卡换到另外一个飞控, 现象不对

2). SD卡再次插回到最初的飞控, 现象ok

最后可以得出应该是px4的问题. 得出了这个结论, 又把这个ok的px4的SD卡放到另外一个px4(这个px4的执行情况现象不对). 发现是不能正确执行, 不确定的是当px4刷入固件的时候, 代码是放到硬件内部了, 还是存放在sd卡上了; 若是存放在sd卡, 那么就是硬件的问题; 反之有可能是px4硬件的问题, 也有可能是刷写时候代码没有正确烧录的问题.

后来把将树莓派中的参数修改了之后, 接入遥控器, 现象不对.

2.1.2 20200802

1. 理清楚px4 upload的时候代码是存放在哪里? px4硬件内部, 还是sd卡
2. 重新刷几个固件看看是不是现象不对
3. 更换 USB 接口 ==> 进入定高模式, 计算出来的值是1.5895, 朝着右下方去飞了
4. 换了个固件, 刷了同样的 Firmware, 现象是一样的
5. 换了一个线, 现象也是一样的

PX4 的数据可以被正确执行, PX4 的硬件在环的数据流是什么样子的? PX4 连接 QGC 连接 XPlane10.

XPlane10 为仿真提供位姿数据, 在 QGC 中进行显示. XPlane10 和 QGC 内部连接, 所以不能得出什么结论.

run changwei's px4 code

在运行昌伟代码的时候, 使用今天的新固件不能切换模式(手动和树莓派切换都是不ok的); 但是使用昨天的固件, 刷入程序, 可以手动进入mission模式并且返航.

不接遥控器的时候, 现象不稳定.

现在昌伟这个进不去 mission 模式, 手动切入(有的时候好, 有的时候不行, 需要多插拔几次)

明天: attitude controller −− > PX4降到20

2. 1. 3 20200803

enableATLTCL 一直发送同一个RC指令; 同时 Attitude controller 发送计算之后的RC指令, 现象如何 ===> 可以按照 Attitude controller 计算的指令进行运行.

先这样看看 protect 节点逻辑如何, 能不能事先退出和进入 ==¿ OK 的

protect 发送指令需要通过processor间接控制, 还是直接控制?

因为在实际飞行过程中, 需要遥控器或者地面站来让它进入定高模式, 所以才有节点 enableALTCTL

遥控器发送切换模式指令, attitude controller 发送RC指令首先飞控接入遥控器, 遥控器会给飞控发送RC指令, 这个时候, 启动树莓派内部的算法逻辑, 进入定高模式, 执行算法. 当通过地面站切回到return模式的时候 ==¿ 需要退出定高模式, 只能通过控制 publish 输入. 而退出来了, 就没有办法再进去了

进入定高模式需要两个条件, 一个是使能 ALTCTL; 另外一个是 publish RC 指令进入定高模式之后, 再退出(禁止publish), 想要进入, 第一个条件满足, 第二个条件就没有办法; 我们可以通过其他方式第二次进入

那么这个第二次进入就同时需要 mode == "ALTCTL" And publish();

1. 在代码中加入保护机制, 可以退出也可以进入
2. enableATLTCL 一直发送同一个RC指令; 同时 Attitude controller 发送计算之后的RC指令, 现象如何 ===> 可以按照 Attitude controller 计算的指令进行运行.
3. 需要解决树莓派上电自启动的事情
4. protect 发送指令需要通过processor间接控制, 还是直接控制? ===> 间接控制
5. 明天: attitude controller −− > PX4降到20

目的: 找到一个方法去让节点开机自启动; 需要找到一个方法可以验证一个进程开机启动了; 拿一个 UDP send 和 UDP Receive 来进行测试; UDP send, UDP Receive的本地源代码.

2. 1. 4 20200804

现在有了可以后台启动的方法, 如下所示.

- 1. 按照脚本的方法进行启动
- 2. 按照一条指令一条指令的执行(在rc.local文件中执行)

若要按照脚本的方法进行启动, 那么需要验证:

- 当launch文件没有启动完全的时候, 后面的节点会等待(verified)
- 或者是设置一个小小的时延

脚本内部的执行逻辑

```
#!/bin/bash
a=0
b=10
echo "Will it execute the source command!"
source ~/fixedWing_ws/devel/setup.bash
echo "Will it execute the UDP_Send command!"
rosrun communication UDP_Send 0 -mt UH -p 8002&
echo "Will it execute the while()!"
while [ "$a" != "$b" ];
do
#       /home/pi/UDP/UDP_Send &
        sleep 1s
        a=$(($a+1))
        echo "waiting 1s... $a"
done

echo "Will it open the px4.launch!"
roslaunch mavros px4.launch fcu_url:=/dev/serial0:921600 &
```

在rc.local中执行脚本失败了, 卧槽! 现象是*UDP_Send*是被执行了, 但是后面打开px4.launch文件失败了(会是while导致的吗?)

执行输出打印在终端, 这个有点不可行 中间的时延是必需的. 避免程序的并发性.

- 目的: 找到一个方法去让节点开机自启动;
- 需要找到一个方法可以验证一个进程开机启动了;
- 拿一个 UDP send 和 UDP Receive 来进行测试;
- UDP send, UDP Receive 的本地源代码.
- 验证方法, 修改rc.local文件是可以得到目的的
- 通过修改rc.local文件可以达到开机自启动一个脚本
- 有了开机自启动的方法之后, 启动一个脚本, 在脚本中添加执行逻辑
- 脚本内部的执行逻辑

在rc.local中执行单个指令是ok的, 但是不能执行脚本.
那如果执行两个指令呢? 是ok的. 执行ros相关指令没有通过

解决rc.local不能执行脚本文件

2.1.5 20200805

解决rc.local不能执行脚本文件的问题 == 运行通过sh指令来执行
解决rc.local不能执行脚本文件

rc.local 文件中的执行指令为 sudo sh /home/pi/UDP/test.sh 可以被正确执行;

- 在执行命令的时候, 我们需要加上 sudo 权限.

- rc.local 文件中的执行指令为 sudo sh /home/pi/UDP/test.sh 可以被正确执行;
- rc.local 文件中的执行指令为 sh /home/pi/UDP/test.sh? 可以被正确执行;
- rc.local 文件中的执行指令为 source /home/pi/UDP/test.sh? 不可以;

测试:通过脚本的方法来执行两条指令

单独执行脚本是可以执行的. 但是将执行脚本的启动指令放在rc.local中却不能执行. 我可以验证: 通过在前面加上一个创建文件或者是往一个文件中写入一条输出的方式进行.

```
#!/bin/bash

touch /home/pi/UDP/startTest_1.txt
source ~/fixedWing_ws/devel/setup.bash
touch /home/pi/UDP/startTest_2.txt
roslaunch mavros px4.launch fcu_url:=/dev/serial0:921600 &
sleep 1s
touch /home/pi/UDP/startTest_3.txt
rosrun communication UDP_Send 0 -mt UH -p 8002 &
```

执行结果是*startTest_1.txt*和*startTest_2.txt*和*startTest_3.txt*都能被正确创建. 那结果就是source指令, roslaunch指令以及rosrun指令都没有被执行.

那就先测试单个语句能否被成功执行.

```
    #!/bin/bash

    touch /home/pi/UDP/startTest_1.txt
    source ~/fixedWing_ws/devel/setup.bash
    touch /home/pi/UDP/startTest_2.txt
    roslaunch mavros px4.launch fcu_url:=/dev/serial0:921600 &
    # sleep 1s
    # touch /home/pi/UDP/startTest_3.txt
    # rosrun communication UDP_Send 0 -mt UH -p 8002 &
```

执行结果是: roslaunch指令没有被执行. == 如果加上了sudo呢?

- 1. 在test.sh脚本中加(现象不对)
- 2. 在rc.local中加(现象不对)
- 判断是否 roslaunch 文件是否被执行, 我们可以将输出重定向到指定的文件中. 比如下面的代码段.

```
    roslaunch myrobot_bringup minimal.launch >> /home/username/
        catkin_ws/src/myrobot_bringup/start.log 2>&1
```

文件不存在会自动创建

命令 >> 文件: 将命令执行的标准输出结果重定向输出到指定的文件中，如果该文件已包含数据，新数据将写入到原有内容的后面。

命令 >> 文件 2 > &1 或者命令 & >> 文件: 将标准输出或者错误输出写入到指定文件，如果该文件中已包含数据，新数据将写入到原有内容的后面。注意，第一种格式中，最后的 2 > &1 是一体的，可以

认为是固定写法.
command > out.file 2> &1 &: 2 > &1 是将标准出错重定向到标准输出，这里的标准输出已经重定向到了out.file文件，即将标准出错也输出到out.file文件中。

- 覆盖式输出重定向：>
- 追加式输出重定向：>>
- 错误输出重定向：2 > &1

```
#!/bin/bash



touch /home/pi/UDP/startTest_0.txt
echo "1>> "  >> /home/pi/log/start.log 2>&1
source /opt/ros/kinetic/setup.bash  >> /home/pi/log/start.log
    2>&1
# source /opt/ros/kinetic/setup.sh
touch /home/pi/UDP/startTest_1.txt
echo "2>> "  >> /home/pi/log/start.log 2>&1
source /home/pi/fixedWing_ws/devel/setup.bash >> /home/pi/log/
    start.log 2>&1
touch /home/pi/UDP/startTest_2.txt
echo "3>> "  >> /home/pi/log/start.log 2>&1
sudo roslaunch mavros px4.launch fcu_url:=/dev/serial0:921600 >>
    /home/pi/log/start.log 2>&1 &

// output in the start.log
pi@raspberrypi:~ $ more log/start.log
1>>
/home/pi/UDP/test.sh: 6: /home/pi/UDP/test.sh: source: not found
2>>
/home/pi/UDP/test.sh: 10: /home/pi/UDP/test.sh: source: not found
3>>
sudo: roslaunch: command not found
```

1. source: not found

- 因为 bash 文件的标头为: #!/*bin*/*bash* 所以才会出现这个问题 ==> no
- 将source 改为 sh ==> no
- 将source 改为 bash ==> yes

bash 和 source 有什么不一样的地方吗?
2. roslaunch: command not found

- 是因为没有 source 启动脚本, 那么这样的话, 上面的bash启动脚本是不起作用的.

会不会是在执行该程序的时候, 还有一些其他的程序没有被执行. 导致咱们的程序没有办法执行. 执行roslaunch的时候, 需要什么服务?

2.1.6 20200806

如果仍然报 roslaunch 不存在, 那就是说明是 source /opt/ros/kinetic/setup.bash 失败! 猜想: 系统启动的时候, 需要启动一些关于 source 的相关内容 ==> linux 启动的顺序

- 目的: 解决 roslaunch: command not found 错误
- 1. 之所以出现这个问题是因为没有 source /opt/ros/kinetic/setup.bash, 但是之前的已经source过了, 但是没有使用的source, 使用的是bash.
- 2. 尝试使用 source, 但是会出现 source: not found 错误, 所以应该是 setup.bash 没有执行成功.

备选: 链接地址
1. 修改ExecStart后的命令即可执行程序或者shell命令。使用服务开启

2.1.7 20200807

- 目的: 解决 roslaunch: command not found 错误
- 1. 之所以出现这个问题是因为没有 source /opt/ros/kinetic/setup.bash, 但是之前的已经source过了, 但是没有使用的source, 使用的是bash.
- 2. 尝试使用 source, 但是会出现 source: not found 错误, 所以应该是 setup.bash 没有执行成功.
- 那么.bashrc是在什么时候启动的呢?
- 那么如果在.bashrc里面添加了脚本是不是可以执行呢? ==> 必须打开终端才会执行
- /rc.local和/etc/init.d/
- 在 /etc/init.d/ 目录下创建了一个自己的服务, 并且添加到自启动的时候, 可以参数作用, 但是将其添加到通过指令来进行自启动却没有产生作用, 没有搞明白这个目录下的文件如何产生作用.需要网络服务
- 可以在 rc.local 中启动一些服务.
- 将执行脚本放在 /etc/profile 配置文件中, 看看是否可以执行 ==¿ 需要连接上 ssh 才可以执行
- 将执行脚本放在 /etc/profile.d/ 路径下, 看看是否可以执行 ==¿ 不可以

  因为 /etc/profile.d/ 路径下的脚本文件都是在 /etc/profile 脚本启动的时候进行遍历执行的.

/etc/rc0.d /etc/rc6.d 分别是什么意思? 启动的时候需要执行哪一个?

查看当前的运行级别

```
pi@raspberrypi:/etc $ who -r
    run-level 5  2020-08-07 11:28

pi@raspberrypi:/etc/rc5.d $ ls -la
```

```
total 8
drwxr-xr-x  2 root root 4096 Aug  7 09:46 .
drwxr-xr-x 99 root root 4096 Aug  7 11:38 ..
lrwxrwxrwx  1 root root   26 Apr  8  2019 S01console-setup.sh -> ../init.d/console-setup.sh
lrwxrwxrwx  1 root root   23 Aug  7 09:46 S02Auto_udp_send -> ../init.d/Auto_udp_send
lrwxrwxrwx  1 root root   24 Aug  4 09:45 S02binfmt-support -> ../init.d/binfmt-support
lrwxrwxrwx  1 root root   16 Aug  4 09:45 S02dhcpcd -> ../init.d/dhcpcd
lrwxrwxrwx  1 root root   17 Aug  4 09:45 S02rsyslog -> ../init.d/rsyslog
lrwxrwxrwx  1 root root   22 Aug  4 09:45 S02triggerhappy -> ../init.d/triggerhappy
lrwxrwxrwx  1 root root   14 Aug  4 09:45 S03cron -> ../init.d/cron
lrwxrwxrwx  1 root root   14 Aug  4 09:45 S03dbus -> ../init.d/dbus
lrwxrwxrwx  1 root root   24 Aug  4 09:45 S03dphys-swapfile -> ../init.d/dphys-swapfile
lrwxrwxrwx  1 root root   17 Aug  4 09:45 S03paxctld -> ../init.d/paxctld
lrwxrwxrwx  1 root root   22 Aug  4 09:45 S03raspi-config -> ../init.d/raspi-config
lrwxrwxrwx  1 root root   19 Aug  4 09:45 S03rng-tools -> ../init.d/rng-tools
lrwxrwxrwx  1 root root   15 Aug  4 09:45 S03rsync -> ../init.d/rsync
lrwxrwxrwx  1 root root   13 Aug  4 09:45 S03ssh -> ../init.d/ssh
lrwxrwxrwx  1 root root   22 Aug  4 09:45 S04avahi-daemon -> ../init.d/avahi-daemon
lrwxrwxrwx  1 root root   19 Aug  4 09:45 S04bluetooth -> ../init.d/bluetooth
lrwxrwxrwx  1 root root   18 Aug  4 09:45 S05plymouth -> ../init.d/plymouth
```

由此可见, 在当前运行级别下, 没有networking的启动服务. 我们可以将networking的服务项加到自动启动项中 ==> 添加失败;

或许我们可以将代码添加到 networking 的启动服务内部.

尝试在 networking 中添加指令.

近几日总结 尝试了挺多种方法, 总结如下:

- /etc/profile: 只有 Login shell 启动时才会运行 /etc/profile 这个脚本，而Non-login shell 不会调用这个脚本。
- /etc/profile.d/: 在执行/etc/profile 文件的时候, 会遍历该路径下的所有脚本, 依次执行.

- /.bashrc: 打开一个shell端口的时候, 会被执行
- /etc/init.d/: 内部存放一些服务文件, 通过自己添加自定义的文件, 手动打开服务可以, 重启却没有反应
- /etc/rc.local: 文件中可以添加一些上电自启动的启动指令, 但是在执行ros相关指令的时候, 会出现报错, 目前使用的是这种方式进行尝试.
- 目前采用的是, 在 /etc/init.d/ 下面添加一个启动服务文件, 之后在 rc.local 文件中进行 start. 开机之后通过指令查看该服务的运行情况, 发现其先与网络服务打开, 导致后面UDP_Send没有办法执行(这个应该是和内部的启动优先顺序有关系).

思路: 现在我们可以先不用udp, 因为是单机, 所以, 我们只需要测试, 当树莓派启动的时候, 登录一下, 执行后台进程, 当把shell关闭了之后, 该进程还会不会存在? ==¿ 不会, 相关进程也会关闭, 因为进程号

使用 *nohup* 来使用.

候选方法: (使用的是/etc/profile脚本)

1. 在使用之前需要先进入 shell 终端一下(为了开启一下脚本) , 好像不行, 还是需要ssh.
2.