

Prepared by: Saeid

August 2021

Drone's OFFBOARD with Position Control

Topic: *'mavros/setpoint_position/local'*

Test Report:

VMware : Ubuntu 18.04.5

ROS Version: melodic

Protocol: Mavros

Flight Controller: Pixhawk4

Flight Firmware: PX4 v1.12.0

Baudrate: 921600

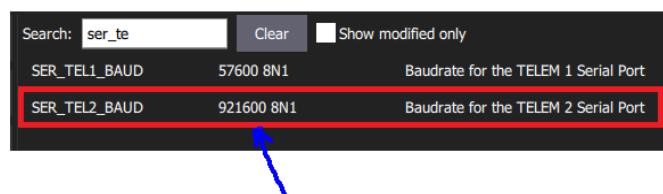
Note: Laptop is connected to Pixhawk' TELEM2 via USB-to-Serial

Note: There is no RC in my setup so that RC receiver is not connected to the Pixhawk.

Process:

Step#1: QGround control setting:

- Baudrate setting and enable TELEM2



Step#2: Connection:

```
roslaunch mavros px4.launch fcu_url:=/dev/ttyUSB0:921600
```

The Mavlink connection is established:

Step#3: Send the set points:

Send the setpoint in separate thread

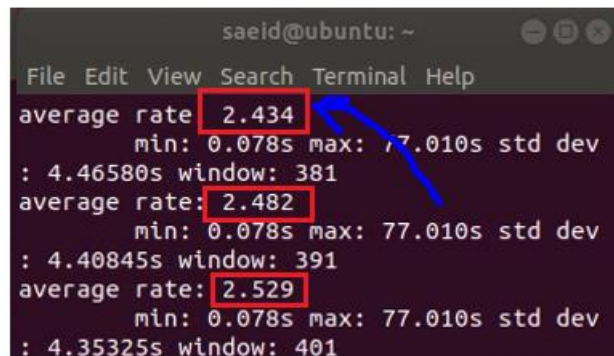
```
def send_pos(self):  
  
    rate = rospy.Rate(10) # 10Hz  
    self.pos.header = Header()  
    self.pos.header.frame_id = "1"
```

Prepared by: Saeid

August 2021

```
while not rospy.is_shutdown():
    self.pos.header.stamp = rospy.Time.now()
    self.pos_setpoint_pub.publish(self.pos)

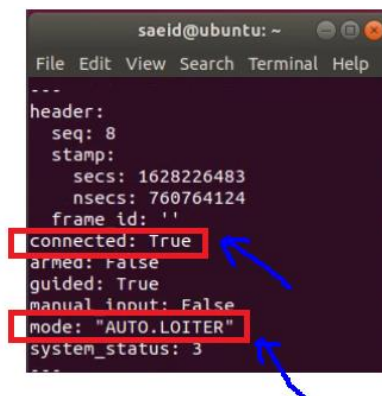
    try:
        rate.sleep()
    except rospy.ROSInterruptException:
        pass
```



A terminal window titled 'saeid@ubuntu: ~' showing the output of a script. The output consists of three lines of statistics, each with 'average rate' and 'window' values highlighted in red boxes. A blue arrow points from the first 'average rate' box to the second one.

```
saeid@ubuntu: ~
File Edit View Search Terminal Help
average rate: 2.434
min: 0.078s max: 77.010s std dev
: 4.46580s window: 381
average rate: 2.482
min: 0.078s max: 77.010s std dev
: 4.40845s window: 391
average rate: 2.529
min: 0.078s max: 77.010s std dev
: 4.35325s window: 401
```

Note: Initial drone mode is “AUTO.LOITER”



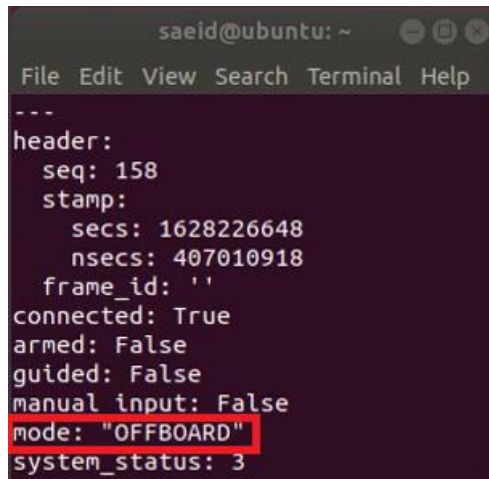
A terminal window titled 'saeid@ubuntu: ~' showing the output of a script. The output is a dictionary-like structure with various status fields. Two fields, 'connected: True' and 'mode: "AUTO.LOITER"', are highlighted in red boxes. A blue arrow points from the first box to the second one.

```
saeid@ubuntu: ~
File Edit View Search Terminal Help
---
header:
  seq: 8
  stamp:
    secs: 1628226483
    nsecs: 760764124
  frame id: ''
connected: True
armed: False
guided: True
manual input: False
mode: "AUTO.LOITER"
system_status: 3
---
```

Step#4: Change the drone mode to OFFBOARD mode:

```
def setOffboardMode(self):
    rospy.wait_for_service('mavros/set_mode')
    try:
```

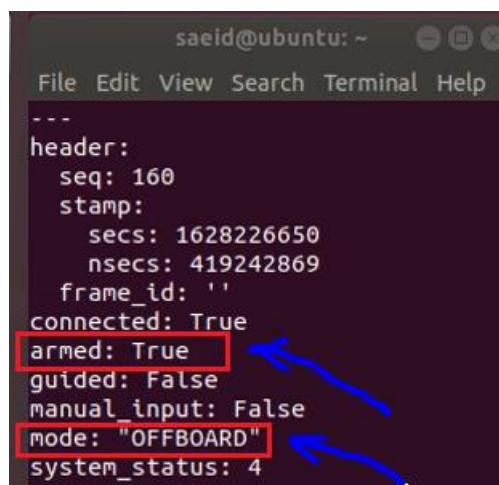
```
flightModeService = rospy.ServiceProxy('mavros/set_mode',
mavros_msgs.srv.SetMode)
flightModeService(custom_mode='OFFBOARD')
except rospy.ServiceException, e:
print "service set_mode call failed: %s. Offboard Mode could
not be set."%e
```



```
saeid@ubuntu: ~
File Edit View Search Terminal Help
---
header:
  seq: 158
  stamp:
    secs: 1628226648
    nsecs: 407010918
  frame_id: ''
connected: True
armed: False
guided: False
manual input: False
mode: "OFFBOARD"
system_status: 3
```

Step#4: Arm the drone :

```
def setArm(self):
    rospy.wait_for_service('mavros/cmd/arming')
    try:
        armService = rospy.ServiceProxy('mavros/cmd/arming',
mavros_msgs.srv.CommandBool)
        armService(True)
    except rospy.ServiceException as e:
        print ("Service arming call failed: %s"%e)
```



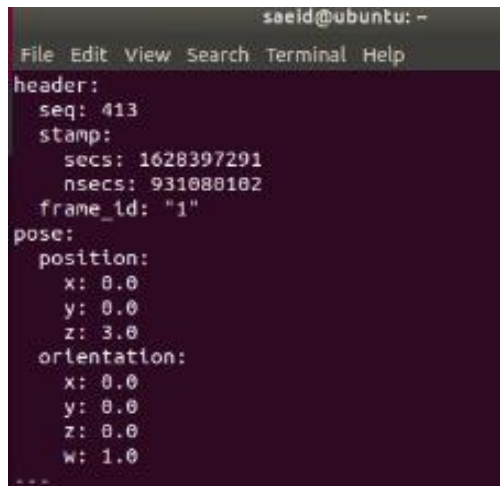
```
saeid@ubuntu: ~
File Edit View Search Terminal Help
---
header:
  seq: 160
  stamp:
    secs: 1628226650
    nsecs: 419242869
  frame_id: ''
connected: True
armed: True
guided: False
manual input: False
mode: "OFFBOARD"
system_status: 4
```

Prepared by: Saeid

August 2021

Step#4: update and Send a new setpoints :

```
self.pos.pose.position.x= 0  
self.pos.pose.position.y= 1  
self.pos.pose.position.z= 3
```



```
saeid@ubuntu: ~  
File Edit View Search Terminal Help  
header:  
  seq: 413  
  stamp:  
    secs: 1628397291  
    nsecs: 931088102  
  frame_id: "1"  
pose:  
  position:  
    x: 0.0  
    y: 0.0  
    z: 3.0  
  orientation:  
    x: 0.0  
    y: 0.0  
    z: 0.0  
    w: 1.0
```

Information: Log Analysis using Flight Review:

https://logs.px4.io/plot_app?log=a29f7df0-adc9-4f7f-8396-01a444951d24

Logged Messages

#	Time	Level	Message
0	0:01:10	INFO	[commander] Armed by external command
1	0:01:10	INFO	[logger] Start file log (type: full)
2	0:01:10	INFO	[logger] [logger] /fs/microsd/log/2021-08-06/05_17_18.ulg
3	0:01:10	INFO	[logger] Opened full log file: /fs/microsd/log/2021-08-06/05_17_18.ulg
4	0:01:12	INFO	[commander] Takeoff detected
5	0:01:13	WARNING	[ekt2] primary EKF changed 3 (timeout) -> 0
6	0:01:58	WARNING	[commander] Failsafe enabled: no RC and no offboard
7	0:01:58	INFO	[commander] Failsafe mode activated

Expectation:

- Drone has to take-off and mover to 2 meter above of the ground.

Problem:

- Drone doesn't take-off

Code:

```
#!/usr/bin/env python

# _____ include Libraries _____

import rospy
from geometry_msgs.msg import PoseStamped, Quaternion
from mavros_msgs.srv import CommandBool, CommandTOL, SetMode
from mavros_msgs.msg import *
from std_msgs.msg import Header
from threading import Thread
from tf.transformations import quaternion_from_euler
import math
import time

# _____

class MavrosOffboardPosctlTest():

    def __init__(self):

        print("class")
        self.setUp()

    def setUp(self):
        self.pos = PoseStamped()
        self.pos_setpoint_pub =
rospy.Publisher('mavros/setpoint_position/local', PoseStamped,
queue_size=1)

        #send setpoint in seperate thread to better prevent failsafe
        self.pos_thread = Thread(target=self.send_pos, args=())
        self.pos_thread.daemon = True
        self.pos_thread.start()

    def send_pos(self):
        rate = rospy.Rate(10) # 10Hz
        self.pos.header = Header()
        self.pos.header.frame_id = "1"

        while not rospy.is_shutdown():
            self.pos.header.stamp = rospy.Time.now()
```

```
        self.pos_setpoint_pub.publish(self.pos)
        #print('Topic: mavros/setpoint_position/local')

        try:
            rate.sleep()
        except rospy.ROSInterruptException:
            pass

    def setTakeoff(self):
        rospy.wait_for_service('mavros/cmd/takeoff')
        try:
            takeoffService = rospy.ServiceProxy('mavros/cmd/takeoff',
mavros_msgs.srv.CommandTOL)
            takeoffService(altitude = 3)
        except rospy.ServiceException as e:
            print ("Service takeoff call failed: %s"%e)

    def setland(self):
        rospy.wait_for_service('mavros/cmd/land')
        try:
            takeoffService = rospy.ServiceProxy('mavros/cmd/land',
mavros_msgs.srv.CommandTOL)
            takeoffService(altitude = 3)
        except rospy.ServiceException as e:
            print ("Service takeoff call failed: %s"%e)

    def setArm(self):
        rospy.wait_for_service('mavros/cmd/arming')
        try:
            armService = rospy.ServiceProxy('mavros/cmd/arming',
mavros_msgs.srv.CommandBool)
            armService(True)
        except rospy.ServiceException as e:
            print ("Service arming call failed: %s"%e)

    def setDisarm(self):
        rospy.wait_for_service('mavros/cmd/arming')
        try:
            armService = rospy.ServiceProxy('mavros/cmd/arming',
mavros_msgs.srv.CommandBool)
            armService(False)
        except rospy.ServiceException as e:
            print ("Service disarming call failed: %s"%e)

    def setOffboardMode(self):
        rospy.wait_for_service('mavros/set_mode')
        try:
            flightModeService = rospy.ServiceProxy('mavros/set_mode',
mavros_msgs.srv.SetMode)
            flightModeService(custom_mode='OFFBOARD')
        except rospy.ServiceException, e:
            print "service set_mode call failed: %s. Offboard Mode could
not be set."%e

    def testPosctl(self,x,y,z,yaw):
```

```
self.pos.pose.position.x= x
self.pos.pose.position.y= y
self.pos.pose.position.z= z
yaw_degree = yaw #----- North
yaw_radians = math.radians(yaw_degree)
print(yaw)
quaternion = quaternion_from_euler(0,0,yaw_radians)
self.pos.pose.orientation = Quaternion(*quaternion)

def main():

    print("\r\n ----- Start the Program -----\r\n")
    rospy.init_node('Positionctl_node', anonymous=True)
    node = MavrosOffboardPosctlTest()

    time.sleep(2)
    node.setOffboardMode() # activate OFFBOARD mode
    rospy.loginfo("Activated OFFBOARD MODE is Done..")

    time.sleep(2)
    node.setArm()
    rospy.loginfo("ARM Drone is Done..")

    time.sleep(2)
    node.testPosctl(0,0,3,0)
    rospy.loginfo("x=0 , y=0 , z=1 , yaw=0")

    time.sleep(10)
    rospy.loginfo("LANDING ... ")

    node.setland()
    time.sleep(2)

    rospy.loginfo("DISARM ... ")

    node.setDisarm()

    while True:
        time.sleep(1)
        # rospy.spin()

if __name__ == '__main__':
    main()
```