

instructables

Let's Make ...



Featured

Write an Instructable

Login | Sign Up

Classes

Contests

Forums

Answers

Teachers

 **AUTODESK.** Make anything.

advertisement

Control a Pixhawk Drone Using ROS and Grasshopper by artiommaxim in robots



Download



9 Steps



+ Collection



I Made it!



Favorite



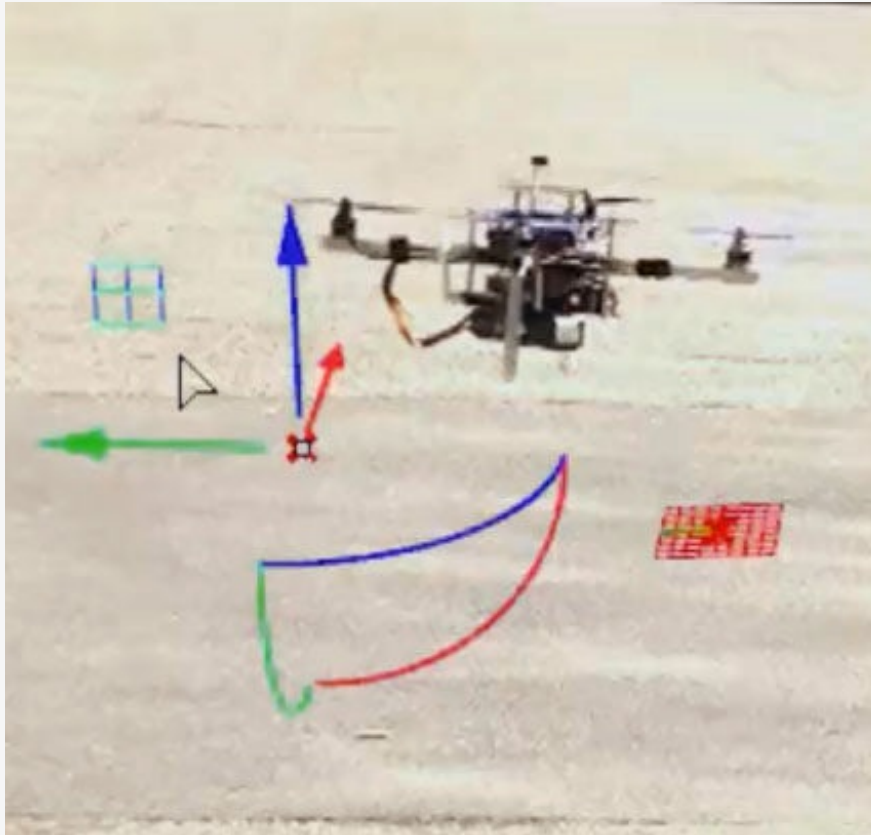
Share ▼



advertisement

About This Instructable





👁 7,349 views

💖 28 favorites

License:

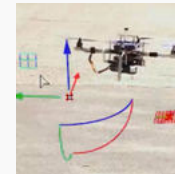


artyommaxim

Follow

9

More by artyommaxim:



advertisement



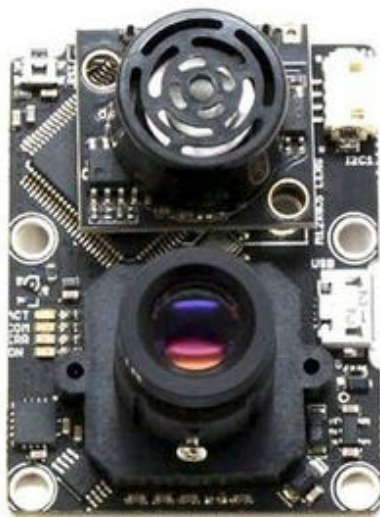
Modern flying robots, aka drones are amazing machines. Their application potential is huge and still growing. No wonder that numerous researchers, makers and entrepreneurs are turning their attention to this technology and coming up with new exciting usage scenarios. These challenging use cases push the development of the technology forward.

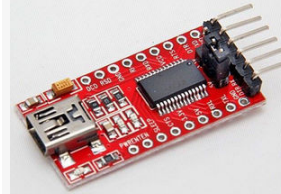
In this Instructable I'd like to share a workflow and basic concepts for setting up a computer controlled drone. The Robot Operating System (ROS) provides an endless source of possibilities for creating novel drone applications: aerial delivery,

smart surveying, automated area patrolling, ai driven behaviour and countless others.

I will show how to implement a simplest workflow for a ROS enabled Pixhawk drone and also create a fun user friendly control interface using Grasshopper.

Step 1: Item List





Show All 11 Items

Item list:

- Any 400-class (or larger) drone

Anything that is capable of carrying the weight of additional electronics. Pixhawk autopilot We're going to be using it with PX4 flight software. PX4 firmware is one of the so called "flight stacks" available for Pixhawk autopilot. Compared to APM firmware, PX4 is more suitable for customizability, modification and experimental setups.

- PX4Flow sensor or GPS module

Flow sensor provides a means of localizing vehicle in space, without the use of GPS. With this sensor, it is possible to fly the vehicle indoors or if gps is not available. It works by measuring the velocity of the vehicle by tracking the motion of features in its camera image. It also has an integrated sonar, which provides altitude measurements for the vehicle. The flow sensor is only necessary in this setup to provide a position estimation source. Alternatively, you can use GPS, the workflow would be exactly the same.

- Onboard computer with Linux Ubuntu and ROS (Odroid XU4)

http://www.hardkernel.com/main/products/prdt_info....

Having an onboard computer is necessary if you need to do some time critical computation onboard, such as computer vision and obstacle avoidance. Although we're not going to use CV in this setup, it can easily be extended to do so. In this setup, we're going to use onboard computer as a WiFi communication hub with autopilot. I've chosen to share the Odroid based setup, because it has a full support for Ubuntu, unlike Raspberry Pi. It's much much faster as well. And Ubuntu is a system that is fully supported by ROS. As my experience shows, trying to use RPi with ROS is a big pain in the a\$\$ really, to put it simple. Also, other Odroid models (C2) or other single board computers that support Ubuntu will work in this setup. Take a look at this page to know more about ROS: <http://www.ros.org/about-ros/>

There is an extensive user manual for XU4:

<http://magazine.odroid.com/wp-content/uploads/odroid-xu4-user-manual.pdf#page=1>

Periphery:

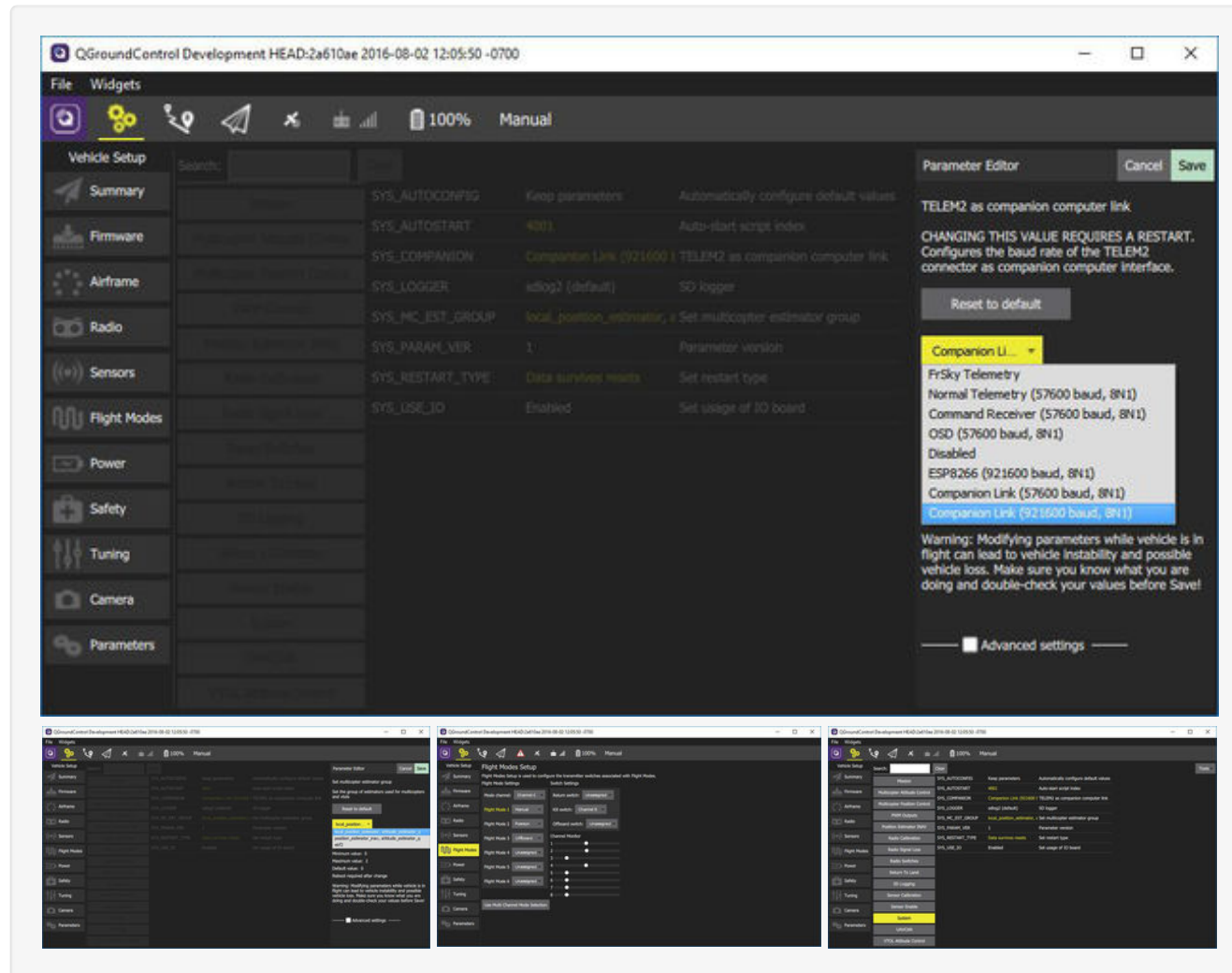
- **3.3v capable usb to serial adapter for connecting autopilot to onboard computer**
- **6-pin serial cable for Pixhawk**
- **5A 5V DC/DC converter for powering Odroid**
- **DC Plug Cable Assembly 5.5mm**

http://www.hardkernel.com/main/products/prdt_info...

- **Wifi adapter**

http://www.hardkernel.com/main/products/prdt_info... preferable to use an officially supported adapter, although any other will also probably work. If you're planning to have a long distance wifi connection (>10 meters), or if the environment is full with other wifi network signals, it's better to use an adapter with an external antenna.

Step 2: Setup PX4 Autopilot



1. Install QGC on your computer:

<https://donlakeflyer.gitbooks.io/qgroundcontrol-us...>

QGroundControl is a ground control station program for communicating with autopilot to configure, read data, send commands among other things in a user friendly fashion.

2. Using QGC, flash Pixhawk with PX4 firmware, configure airframe, calibrate sensors and remote as shown in the tutorial: <http://dev.px4.io/starting-initial-config.html>

3. Set parameters:

SYS_COMPANION to 921600 - for communication with companion computer.

SYS_MC_EST_GROUP to local_position_estimator - for using LPE estimator.

Estimator is one of the autopilot internal programs, dealing with estimating the position of the vehicle in space using available onboard sensors as well as external estimation sources, such as MOCAP. LPE is the most developed and tested of the estimators available, therefore we're going to be using it.

Configure a 3-position switch on your remote and set the corresponding flight modes to:

MANUAL -> POSITION CONTROL -> OFFBOARD

https://www.youtube.com/results?search_query=3+pos...

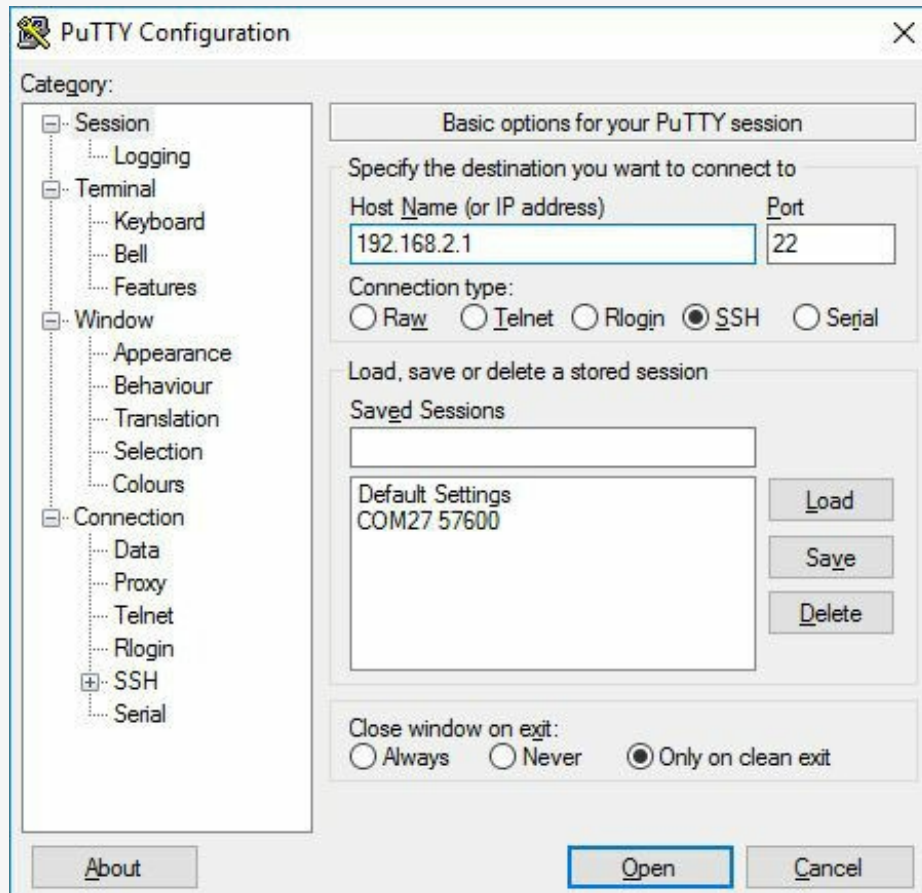
A good idea is to set a "Kill" switch which will allow to quickly stop the motors in case of emergency. And of course, a failsafe behaviour setup is always mandatory, which you can set in "Safety" tab.

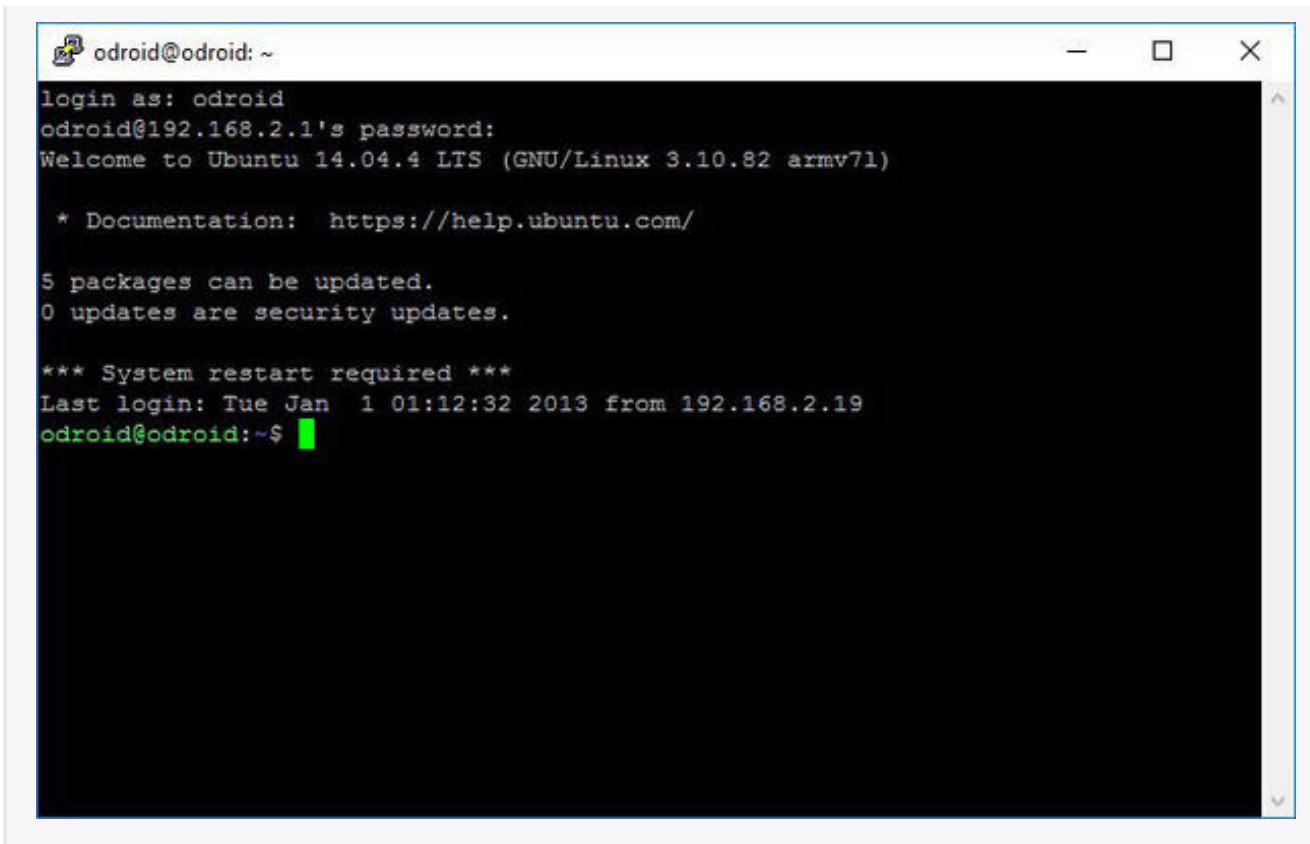
4. Configure and mount PX4Flow sensor using the guide:

http://dev.px4.io/flow_lidar_setup.html

5. Make sure you can get a stable position control, like shown in the video:

Step 3: Install Ubuntu



A terminal window titled 'odroid@odroid: ~' with standard window controls. The terminal text shows a login process for the 'odroid' user on an Ubuntu 14.04.4 LTS system (GNU/Linux 3.10.82 armv7l). It prompts for a password, displays a welcome message, provides documentation links, and reports that 5 packages can be updated, including 0 security updates. A message indicates a system restart is required. The last login was on Tue Jan 1 01:12:32 2013 from 192.168.2.19. The prompt 'odroid@odroid:~\$' is shown with a green cursor.

```
odroid@odroid: ~
login as: odroid
odroid@192.168.2.1's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.10.82 armv7l)

 * Documentation:  https://help.ubuntu.com/

5 packages can be updated.
0 updates are security updates.

*** System restart required ***
Last login: Tue Jan  1 01:12:32 2013 from 192.168.2.19
odroid@odroid:~$
```

1. On http://odroid.com/dokuwiki/doku.php?id=en:xu3_rele... download Ubuntu 14.04.01 (20150212) image.
2. Use this guide to flash the memory card: http://odroid.com/dokuwiki/doku.php?id=en:odroid_...
3. Connect a mouse, keyboard, screen and wifi adapter and power up XU4.
4. Setup a WIFI network connection: http://odroid.com/dokuwiki/doku.php?id=en:odroid_...
5. Resize root partition using Odroid Utility as shown here on 1:30 :
http://odroid.com/dokuwiki/doku.php?id=en:odroid_...
6. Install Putty http://odroid.com/dokuwiki/doku.php?id=en:odroid_... on your PC.

Putty is what will allow to connect to a terminal of Ubuntu system.

7. Install Fing Network scanner for your smartphone. This tool will allow to conveniently check the ip address of the XU4 on the network.

8. In Putty, use SSH to remotely login to Ubuntu. Open Putty and use SSH and the address you discovered in Fing to log in.

login: odroid

password: odroid

Note that while typing the password, the line will remain blank, that's ok.

9. Alternatively, you could setup a wifi hotspot on XU4 to avoid using a router. This comes handy if you want to test your setup outdoors for example.

http://odroid.com/dokuwiki/doku.php?id=en:odroid_...

Step 4: Install ROS

The ROS logo consists of a 3x3 grid of blue dots to the left of the letters "ROS" in a large, bold, blue sans-serif font.

Open Source Robotics Foundation

1. Now, we need to install ROS. XU4 is an ARM platform, therefore we'll be using the tutorial for installing ROS on ARM:

<http://wiki.ros.org/indigo/Installation/UbuntuARM>

2. Next, install Mavros plugin for ROS. <http://dev.px4.io/ros-mavros-installation.html>

Mavros is a communication plugin between autopilot and ROS. For communicating with external computers, autopilot uses Mavlink protocol. Mavlink is a number of common commands for aerial vehicle, that control many aspects of its operation. After we connect Pixhawk to the XU4 using serial interface, Mavros will be able to communicate with PX4 autopilot to send commands, read status of the vehicle and also forward the connection to other computers on the network.

3. Install rosbridge: <http://dev.px4.io/ros-mavros-installation.html>Mav... by running

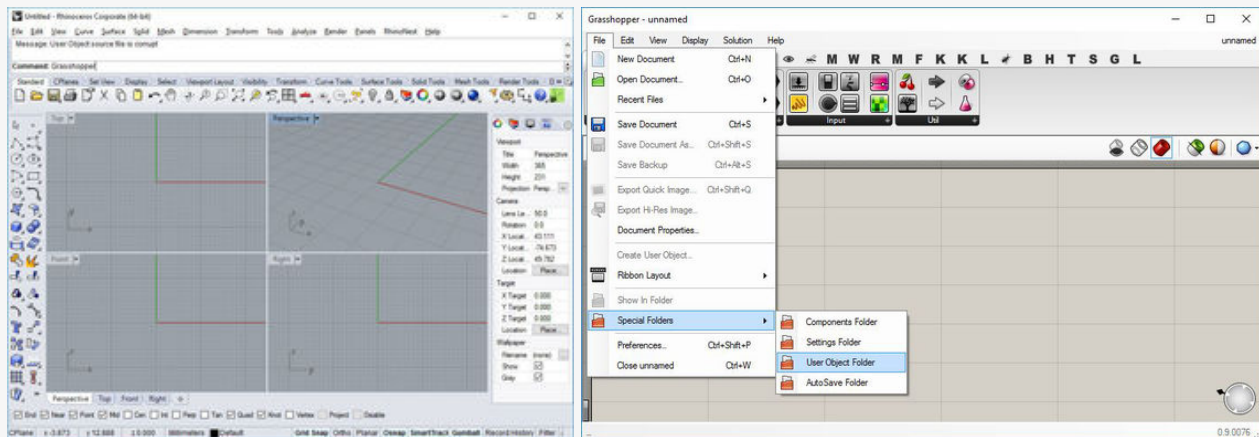
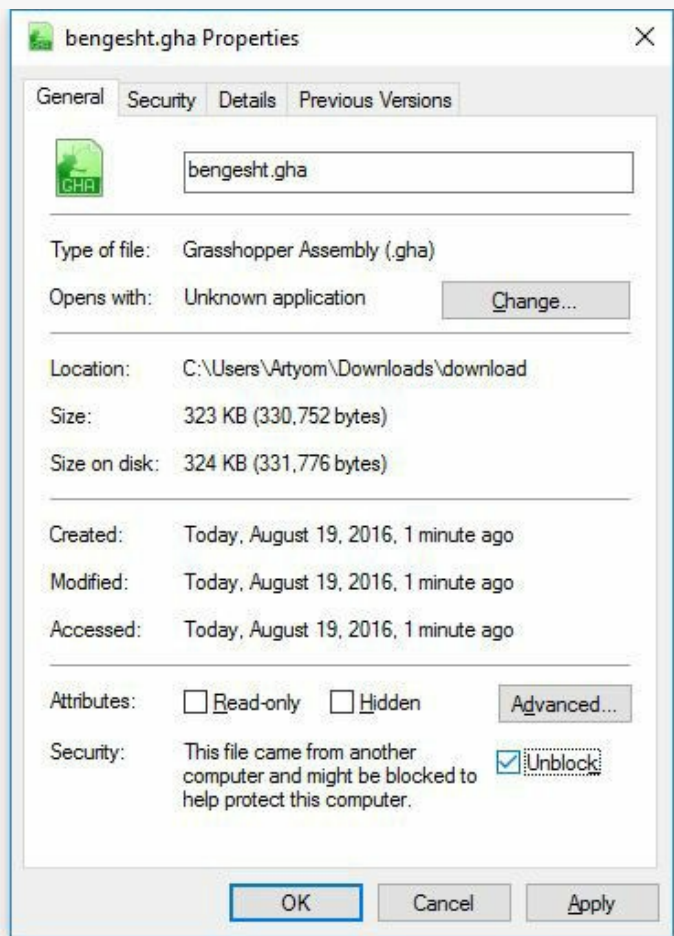
sudo apt-get install ros-indigo-rosbridge-server

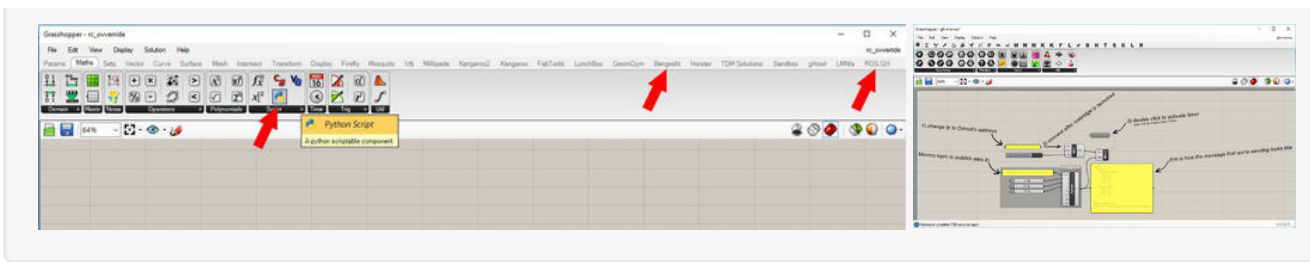
Rosbridge is a package that allows external programs to communicate with ROS. we will later use it to set up a connection to Grasshopper.

4. Install screen:

sudo apt-get install screen

Step 5: Ground Computer Software Setup





1. Download and install Rhino 5 <https://www.rhino3d.com/download/rhino/5/latest>

2. Install Grasshopper <http://www.grasshopper3d.com/page/download-1>

3. Download Bengesht and ROS.GH by [behrooz.tahanzadeh](https://github.com/behrooz-tahanzadeh) and GhPython

<http://www.food4rhino.com/project/bengesht?etx>

<https://github.com/behrooz-tahanzadeh/ROS.GH>

<http://www.food4rhino.com/project/ghpython?etx>

4. You need to “**unblock**” the **.gha files and the zip-archive** before you can use them, for that, right click the file -> properties -> unblock -> OK. Do that separately for both the files (bengesht.gha and ghpython.gha) and the archive. Unzip the archive.

5. Next, Open Rhino and in command line type “grasshopper” and press Enter. The grasshopper window will open.

6. Go to File -> Special Folders -> Components Folder and copy the unblocked .gha files into this folder.

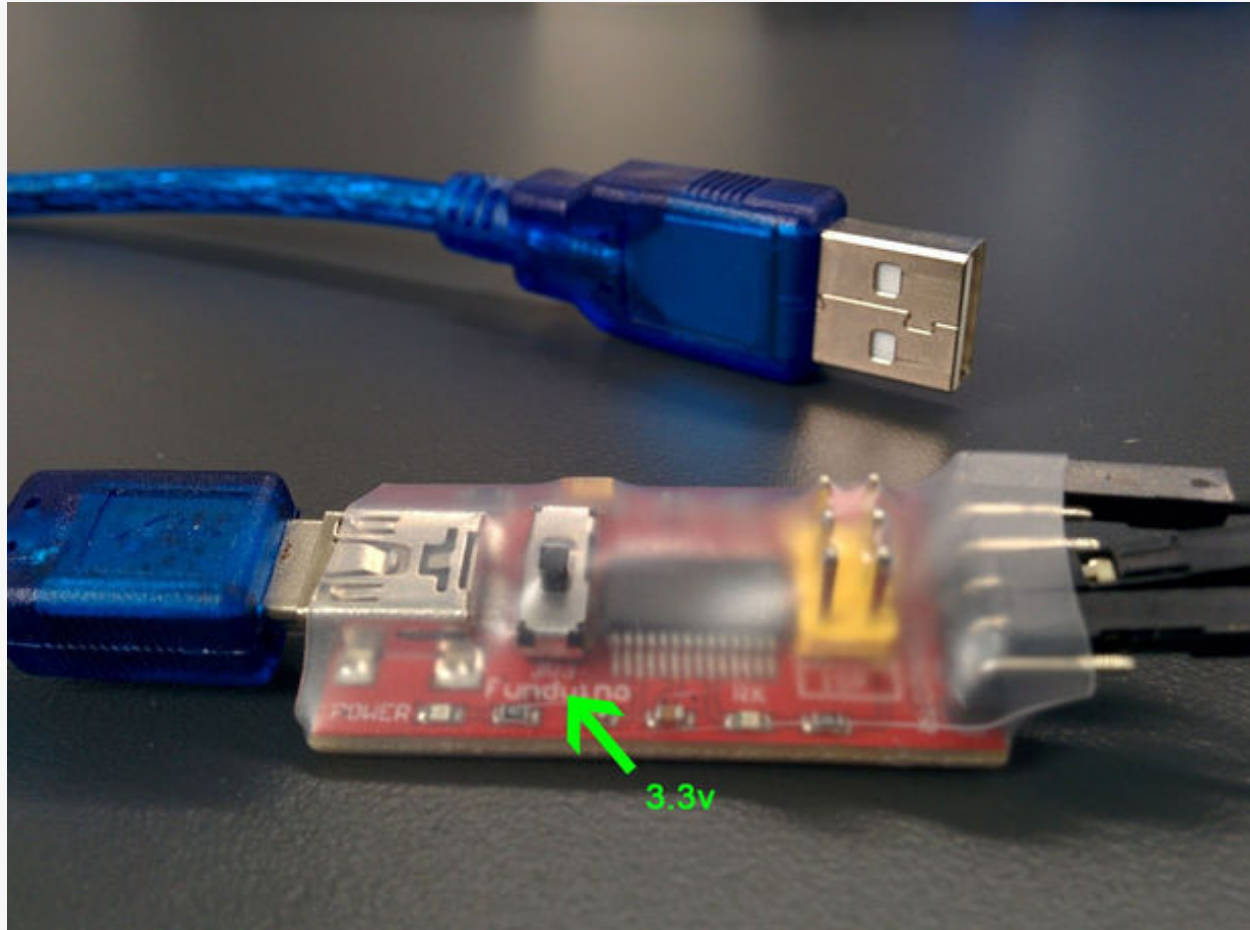
7. Back in Grasshopper again, go to File -> Special Folders -> User Object Folder and copy the unzipped ROS.GH-master folder here.

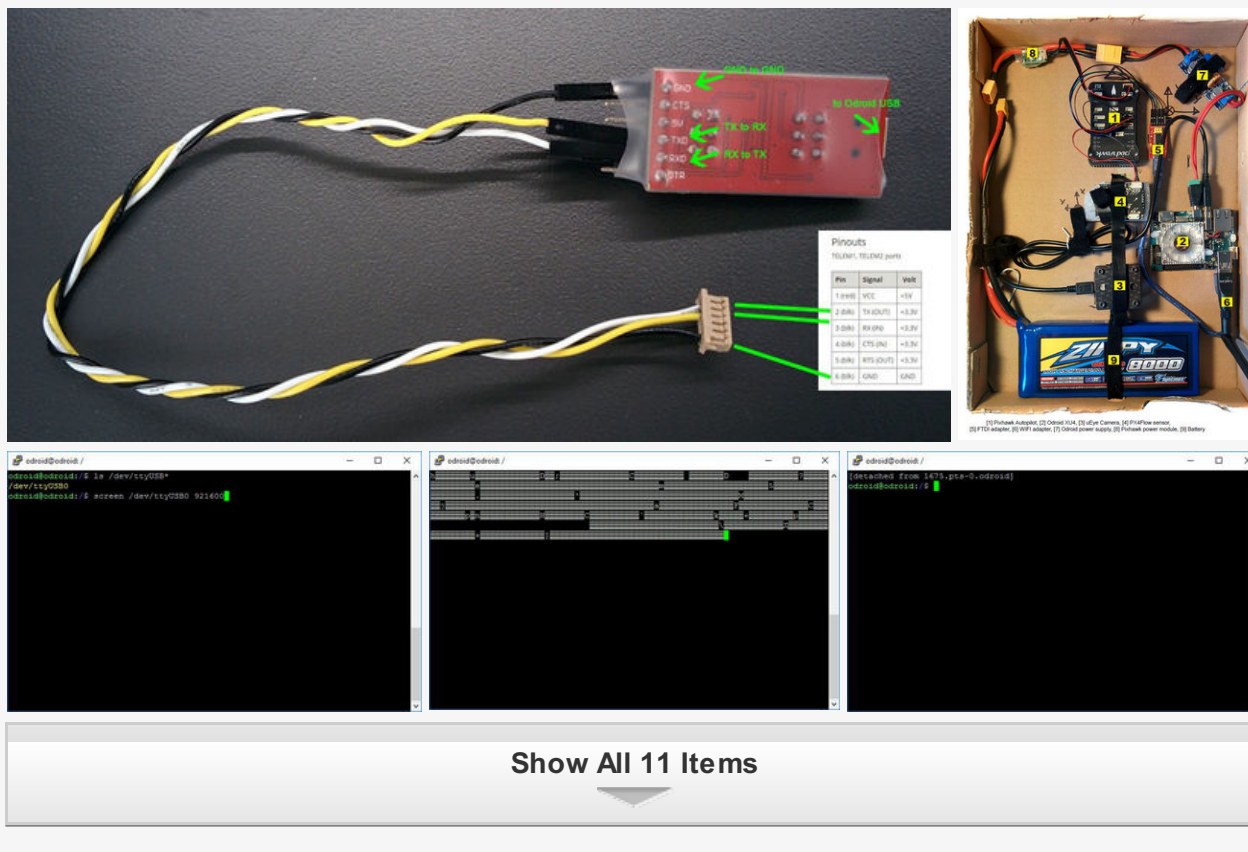
8. In rhino command line type “grasshopperunloadplugin”, that will unload grasshopper, then launch it again by using “grasshopper” command. You should now have all grasshopper plugins installed.

9. Open the gh-setpoint.gh definition:

<https://drive.google.com/file/d/0B2tng5FWxLIBWVFE...> Simply download the .gh file and drag it onto the Grasshopper canvas.

Step 6: Connect Autopilot





1. Use the USB to serial adapter to connect Pixhawk Telem2 port to XU4 USB.

Usefull info: <http://ardupilot.org/dev/docs/odroid-via-mavlink.h...>

It's important that we use 3.3v capable serial adapter, as Pixhawk serial ports work with 3.3v, as described here: <https://pixhawk.org/modules/pixhawk> In my case, the adapter has a switch, for 3.3v or 5v logic.

After connecting USB adapter, the serial port will be accessible in Ubuntu system at /dev/ttyUSB0, if you don't have other serial devices connected. Run

ls /dev/ttyUSB*

to list all of the usb serial ports in the system.

“ls” - is a command to list things in Linux. Also works alone, to browse the contents of the current folder for example.

“/” is the address of the main system folder, just like drive C:/ in windows.

“dev” The /dev directory contains the special device files for all the devices in the system.

“ttyUSB” is the port we’re looking for, and asterix,

“*” is the wildcard, which lists all available variants.

We can check if it is working by using the following command:

screen /dev/ttyUSB0 921600

If you see stuff printing in the terminal, that’s mavlink stream from the autopilot and means everything should work. You can exit from the “screen” program with ctrl-a-d, while holding ctrl, press a then d consecutively.

2. Launch Mavros. In the “Usage” section on mavros wiki page:

<http://wiki.ros.org/mavros> , we can see the launch command:

roslaunch mavros px4.launch

roslaunch - is what is used in ROS to launch the packages with launchfiles. In this case we’re launching mavros package using a launchfile named px4.launch. A launchfile is a simple text file with launch parameters. We can quickly see the contents of the px4.launch launchfile at mavros github repository in the launch folder: <https://github.com/mavlink/mavros/tree/master/mav...> Here we are interested in two important parameters:

The fcu_url is the address in the Ubuntu system, where mavros can access data stream from the autopilot, in our case it’s a serial port at the address /dev/ttyUSB0. So the default px4.launch file will not work for us, we need to override the fcu_url parameter when launching mavros:

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

The second parameter that we're interested in is `gcs_url`. This is the address of the ground control station that we can forward the mavlink data to. In this case, QGC is running on the other computer on the network, and we can access it over UDP connection. Again, to do that, we can override the parameter in the launchfile by specifying it in the launch command:

```
roslaunch mavros px4.launch fcu_url:=serial:///dev/ttyUSB0:921600  
gcs_url:=udp://@192.168.2.19:14550
```

Replace the address with ip of computer on the network running QGC. If we run this command, mavros should launch and connect to autopilot as well as to QGC:

3. **Using Screen.** Exit mavros for now by pressing ctrl-c. Besides mavros we'll need to launch rosbridge. To launch both programs without using two terminal windows, there is a handy tool called screen. Screen allows to create virtual terminal windows and switch between them easily. To create a screen session: `screen -S sessionname` "-S" is a parameter for creating a new session, and "sessionname" can be any name you like.

Now run mavros:

```
roslaunch mavros px4.launch fcu_url:=serial:///dev/ttyUSB0:921600  
gcs_url:=udp://@192.168.2.19:14550
```

To create a second virtual terminal window, press ctrl-a-c (just like in step 2, holding ctrl, press a, then c) Now in this new terminal, launch rosbridge:

```
roslaunch rosbridge_server rosbridge_websocket.launch
```

Great! To switch between terminals, use ctrl-a-p (for previous) and ctrl-a-n (for next) The nice thing about screen is that even if the ssh session exits for some reason, you can still access it by logging in with ssh again and running a reconnect command:

```
screen -r
```

More info on screen: <https://www.linode.com/docs/networking/ssh/using-...>

Step 7: Sending Data to Mavros

```
/opt/ros/indigo/share/rosbridge_server/launch/rosbridge_websocket.launch http://localhost:11311
odroid@odroid:/$ roslaunch rosbridge_server rosbridge_websocket.launch
... logging to /home/odroid/.ros/log/052735a0-53ad-11e2-867a-f4f26d0dd984/roslaunch-odroid-3722
.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://odroid:34306/

SUMMARY
=====

PARAMETERS
* /rosbridge_websocket/address:
* /rosbridge_websocket/authenticate: False
* /rosbridge_websocket/port: 9090
* /roscistro: indigo
* /rosversion: 1.11.16

NODES
/
  rosapi (rosapi/rosapi_node)
  rosbridge_websocket (rosbridge_server/rosbridge_websocket)

ROS_MASTER_URI=http://localhost:11311

core service [/rosout] found
process[rosbridge_websocket-1]: started with pid [3740]
process[rosapi-2]: started with pid [3741]
registered capabilities (classes):
- rosbridge_library.capabilities.call_service.CallService
- rosbridge_library.capabilities.advertise.Advertise
- rosbridge_library.capabilities.publish.Publish
- rosbridge_library.capabilities.subscribe.Subscribe
- <class 'rosbridge_library.capabilities.defragmentation.Defragment'>
- rosbridge_library.capabilities.advertise_service.AdvertiseService
- rosbridge_library.capabilities.service_response.ServiceResponse
- rosbridge_library.capabilities.unadvertise_service.UnadvertiseService
[INFO] [WallTime: 1357001647.633689] Rosbridge WebSocket server started on port 9090
[INFO] [WallTime: 1357004538.068747] Client connected. 1 clients total.
```


2. Make sure Mavros is getting data from GrasshopperRun rostopic list - to see the mavros topics available.

Rostopic is a command line tool to display information about ros topics, which are basically unidirectional streams of data. In this case the topics with names starting with /mavros are the streams containing various data from autopilot (published topics), as well as channels that we can write to thus sending commands to autopilot (subscribed topics). More on this: <http://wiki.ros.org/rostopic>
<http://wiki.ros.org/Topics>

Change the address in the yellow text node to the Odroid ip. Connect the address node to “WS” node and double click the timer. Create a new virtual terminal in screen and run:

rostopic echo /mavros/setpoint_position/local

Where “echo” is the command used to view the data in the topic “setpoint_position/local”, which we publish data to from Grasshopper. If you move the sliders in GH you should see the same values changing in the terminal.

3. With QGC connected, open the Analyze window. Now if you switch autopilot to offboard mode (with vehicle disarmed) you should see POSITION_TARGET_LOCAL_NED appear in the list of available autopilot topics with the values that we set in Grasshopper. You will also likely notice that the graph is unstable, there are sudden jumps in the data.

We can also see the same effect by running:

rostopic hz /mavros/setpoint_position/local

Where “hz” is the rostopic parameter to show the publication rate of a topic. You should see that topic update rate is rather slow, which is unacceptable for the autopilot. This is the effect of a wifi connection, which cannot provide a continuous and uninterrupted stream of data. The autopilot will put the vehicle in a failsafe mode every time there is a gap in the data stream. So in the next step, we’re going to fix

this. Instead of trying to feed the data continuously to Mavros directly from Grasshopper, we're going to feed it through a simple ROS plugin, that will ensure that the stream is continuous.

Step 8: Custom ROS Plugin

The image is a composite of three screenshots illustrating the development of a custom ROS plugin in Grasshopper.

Top Screenshot: A screenshot of the Grasshopper interface showing a complex network of components. Annotations with arrows point to specific parts of the network:

- 1) change ip to Odroid's address
- 2) connect after rosbridge is launched
- 3) create ROS topic
- 4) right click, "Set one Point"
- 5) subscribe to ROS topic
- message that we're sending to ROS
- message with drone position
- multiply coordinates by 1000 to convert to millimeters
- display point as a plane

Bottom Left Screenshot: A screenshot of the Grasshopper interface showing a 3D view of a point being set. Annotations with arrows point to specific parts of the interface:

- 1) create point
- 2) select point
- 3) reference to grasshopper
- 5) right click, "Set one Point"

Bottom Right Screenshot: A screenshot of a terminal window showing ROS logs. The logs indicate that the custom ROS plugin is running and receiving data from the drone.

```
roslaunch mavsros gh_mavros.py
[INFO] [WallTime: 1357004041.673321] Waiting for FCU connection...
[WARN] [WallTime: 1357004041.723821] FCU is disconnected...
[INFO] [WallTime: 1357004051.084795] INFO: Received new destination!
0.0, 0.0, 0.0
[INFO] [WallTime: 1357004052.735454] INFO: Received new destination!
0.058, 0.426, 0.0
[INFO] [WallTime: 1357004058.287711] INFO: Received new destination!
0.128, 0.426, 0.0
[INFO] [WallTime: 1357004059.977294] INFO: Received new destination!
0.277, 0.239, 0.0
[INFO] [WallTime: 1357004105.430845] INFO: Received new destination!
0.238, 0.104, 0.0
[INFO] [WallTime: 1357004101.430845] INFO: Received new destination!
0.091, 0.453, 0.0
```

So the ROS package that we're going to use will ensure that our stream of commands to autopilot is continuous.

The package is basically a simple Python script. I've included comments in the code, feel free to take a look to get an idea of what's going on.

<https://github.com/art-mx/gh-mavros/blob/master/s...> This is a standard way of installing ROS packages from source from GitHub:

1. Make sure Odroid has internet connection, then run the following commands:

cd ~ - This will bring you to the "home" folder of Ubuntu

mkdir new_ws - We are creating a folder for our new ROS workspace, named new_ws

cd new_ws - Navigating into that folder

mkdir src - This folder will contain the source files of the packages in the workspace

cd src

catkin_init_workspace

git clone <https://github.com/art-mx/gh-mavros.git> - makes a local copy of a github repository

cd ~/new_ws

catkin_make - build the packages in the workspace using catkin tool

echo "source ~/new_ws/devel/setup.bash" >> ~/.bashrc

source ~/.bashrc

That's it, our package should be installed and ready to be used.

More on workspaces and catkin: <http://wiki.ros.org/catkin/Tutorials>

2. Run the mavros and rosbridge, as we did in the step 4. Then in another terminal,

run:

roslaunch gh-mavros gh-commander.py

Here, we're using a "roslaunch" command, which allows you to run an executable, in our case - gh-commander.py "gh-mavros" is the name of our custom package "gh-commander.py" is the name of the executable python script in the package. If everything is ok, we should get a message that FCU (Flight Control Unit, or autopilot) is connected.

3. Now download the gh-targetpose.gh and open it.

<https://drive.google.com/file/d/0B2tng5FWxLlBek1K...>

In this definition we're doing a few more things:

- creating a new ROS topic and publishing data to - our custom plugin will receive updates from this topic and publish an uninterrupted stream to the Mavros topic (/mavros/setpoint_position/local) that we used to send data to directly in the previous step.
- replacing sliders with a point referenced from Rhino viewport - this will allow us to visualize and manipulate the target setpoint in the viewport
- subscribing to a Mavros topic containing information about the vehicle's position in space to visualize it in the viewport - will allow to visualize the motion of the vehicle in the Rhino viewport

4. Connect to rosbridge in the same way we did it before in step 6. Press the button to create a new ROS topic, and reference a new point from the Rhino viewport. Also press the button to subscribe to Mavros topic, we'll be getting data containing vehicle's position in space from it and visualizing it in the viewport.

5. If we now move the point in the viewport, in the terminal where we launched gh-commander.py, we should see a message saying that a new target has been received.

Step 9: Fly With Offboard Control

<http://dev.px4.io/offboard-control.html>


Before continuing: Warning, Offboard control is dangerous. Always be ready to switch back to manual control, or engage the kill switch, in case something goes wrong! Safety first!


When setting a reference point in grasshopper, be sure to put it close to the takeoff location of the drone. Take off manually, put the vehicle in the position control mode first, make sure it can hold position well and then switch to OFFBOARD. At this point you should be able to move the reference point in the Rhino viewport and observe the vehicle fly to the destination.

Grasshopper + ROS Autonomous Vehicle Control from Artyom Maxim on Vimeo.


advertisement


Comments





We have a be nice comment policy.
Please be positive and constructive.

 I Made it!

 Add Images

Post Comment

VinhK

2017-03-08

Reply

Because in the picture of your schematic, it did not look like from the
<https://pixhawk.org/modules/px4flow>

artyommaxim ▶ VinhK

2017-05-09

Reply

You probably mean the orientation of the sensor. Yes, my setup is slightly different from the one in the link. My sensor is rotated 90 degrees. There is an option for that in QGroundControl.

VinhK

2017-03-08

Reply

is your pixhawk connect to Px4Flow like so?<https://pixhawk.org/modules/px4flow>

VinhK

2017-03-08

Reply

If you could chat on gitter or something that would be tremendous. I would like to do a video documentation on this issue if I can for stable hovering. That would help PX4 communities.

VinhK

2017-03-08

Reply

could you give more instructions of how you were able to get stable hovering using PX4Flow? I have follow the tutorial on PX4 documentation and set the Local Position Estimator (LPE) but I could not have a stable hover.

fractalsuar

2016-10-21

Reply

What version of QGC did you use? I installed QGC v 3.0.1 on my Windows machine and I can not find the parameters you have listed in step 3. Not the SYS parameters or the flight modes. Any ideas as to what I should do? Thanks!

artyommaxim ▶ **fractalsuar**

2016-10-22

Reply

I'm using latest v3 as well. It has to be used with a corresponding latest PX4 firmware though. Have you flashed the latest stable firmware (currently 1.4.4 which works in my setup) from QGC as described in 2.2?
(<http://dev.px4.io/starting-initial-config.html>)

fractalsuar

2016-09-18

Reply

Is it possible to do the same thing with the new raspberry pi 3 running ubuntu mate?

artyommaxim ▶ **fractalsuar**

2016-09-19

Reply

If you manage to install ROS and Mavros - it's totally possible, which it is according to these threads:

<http://answers.ros.org/question/229860/ros-running...>

<http://answers.ros.org/question/230076/ros-on-rasp...>

It will likely be a challenge though.

shashankvkt8

2016-09-11

Reply

if u could also share this on diy drones it would be great! The drone community could recognize your work.

hope it helps. cheers

artyommaxim ▶ **shashankvkt8**

2016-09-19

Reply

Will do)

shashankvkt8

2016-09-11

Reply

this is really cool. I too have used odroid xu4 and pixhawk integrating it with ROS in my project if u could kindly check it out (<https://www.instructables.com/id/Vision-Based-Object-Tracking-and-Following-Using-3/>).

artyommaxim ▶ **shashankvkt8**

2016-09-19

Reply

Awesome project as well! Would be really interesting to know more details though. Anyway, this could totally be a base for a commercial product!

benguru

2016-09-07

Reply

Dude this is very cool !!

You could get a master thesis with what you explained here !

It is the most scientific article I have ever read on Instructables ! Well done !

artyommaxim ▶ **benguru**

2016-09-19

Reply

Hey, thanks man! It's actually coming from my master thesis)

seamster

2016-08-25

Reply

Very cool stuff. Drones are more than just toys for sure. Thanks for sharing! :)

artyommaxim ▶ **seamster**

2016-08-26

Reply

Thanks. Totally true. I'll be happy if this helps anyone)

↓ More Comments






Newsletter

Let your inbox help you discover our best projects, classes, and contests. Instructables will help you learn how to make anything!

About Us

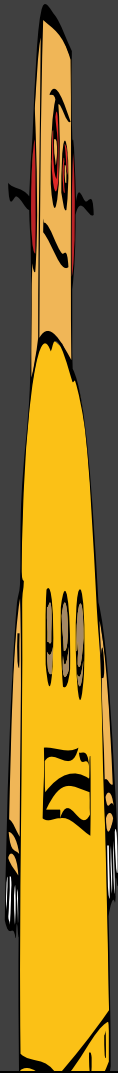
[Who We Are](#)
[Advertise](#)
[Contact](#)
[Jobs](#)
[Help](#)

Find Us

 [Facebook](#)
 [Youtube](#)
 [Twitter](#)
 [Pinterest](#)
 [Google+](#)

Resources

[For Teachers](#)
[Residency Program](#)
[Gift Premium Account](#)
[Forums](#)
[Answers](#)
[Sitemap](#)





© 2017 Autodesk, Inc.

[Terms of Service](#) | [Privacy Statement](#) | [Legal Notices & Trademarks](#) | [Mobile Site](#)

