Tuesday, January 28, 2014

Python Scripting for ArduCopter

Summary of Steps

This can be kind of tricky. I had to modify some file path stuff to get this to go, and it's sort of annoying if you screw up so make sure you do these steps in this order.

I'm assuming you have Python 2.x on your computer. Not Python 3. If neither, Internet your way to a copy of Python 2.x.

1. **DOWNLOAD**: Download a copy of the pymavlink library for Python. It's available at https://github.com/mavlink/mavlink (click the download zip button) or you can use git to pull the folder down using this terminal command:
   git clone git://github.com/mavlink/mavlink.git

   which will pull down the source to the folder that you issued the command from.

2. **FILE STRUCTURE MODIFICATION:** Once you're in the /mavlink folder, you should navigate to /pymavlink/generator. Open up the file called "mavgen_python.py" and go to line 26. Change that line to read:
   from pymavlink.generator.mavcrc import x25crc

   This is necessary because for some reason Python will not recognize relative file paths, at least on my machine. More in a little bit.

3. **GENERATE PYTHON MAVLINK COMMANDS**: You should still be in the generator folder. Issue the following command to generate python-readable versions of the common Mavlink commands:

   python mavgen.py -o mavlinkv10.py ../../message_definitions/v1.0/common.xml

   When we do custom xml for new Mavlink commands, we change the "../../message_definitions/v1.0/common.xml" file to our xml file that we want to generate, then run this command, and do step 4 again.

   When this step is complete, a new file will exist (or the old one will be overwritten) called mavlinkv10.py. We need this as our library of Mavlink commands that other scripts will be referencing.

4. **INSTALL THE PACKAGE**: Change directories up to /mavlink/pymavlink. (*cd ..*). You should now see a script called *setup.py*. We want to run this script. It will build and install the python library "pymavlink" to your machine so that you can use "import pymavlink" in a script. The terminal command is as follows:

   sudo python setup.py install

   Type your password to allow changes and watch it install the package. For windows, you will not need the "sudo" section of the command.

5. **RUN A TEST SCRIPT**: Navigate your way to the /mavlink/pymavlink/examples folder, and open up the script called "mavtest.py." Change line 5 to read:

   from pymavlink.generator import mavlinkv10 as mavlink

   This is, you guessed it, to solve our file path problems again. The script itself will make a virtual instance of our Mavlink communication setup, i.e. you don't need a ArduPilot to use it. Once you've changed line 5, see if you can run it in python:

   python mavtest.py

   If it prints something about message received had data (stuff), then you're good.

6. **INSTALL PYSERIAL (IF NECESSARY)**: We need to hook this thing up to the APM 2.x. If you don't have it already, you need to install the PySerial library. A good way to test if you have it is type "python" to open the interactive interpreter, then:

   import serial

   If you see something about import failed, you don't have it. To install it, head to http://sourceforge.net/projects/pyserial/ and then click the download button. I'm not sure how to unzip a tar ball on a Windows machine, perhaps a little google searching will work. Once you save that directory off and get it expanded, then open the directory in a terminal window and type

   python setup.py install

   to install the package. A quick repeat of the import serial command shown above should show no feedback. This means you have the package now.

7. Go back to the /mavlink/pymavlink/examples folder and locate the apmsetrate.py script. This is how we will view some debug information from the ArduPilot. Be sure that the ArduPilot is connected via USB or telemetry, and then issue the following:

   python apmsetrate.py --device /dev/cu.usbmodem621 --baudrate 115200 --showmessages

   That's what the command looked like on my machine. I use a Mac. If you don't know what to put after the —device tag, then Python and PySerial can help you out. Enter the interpreter by typing "python", then enter the following:

   >>> import os
   >>> import serial
   >>> from serial.tools import list_ports
   >>> for port in list_ports.comports():
   … print port
   … <press Enter>

   This should yield something like the following:

```
['/dev/cu.Bluetooth-Incoming-Port', 'n/a', 'n/a']
['/dev/cu.Bluetooth-Modem', 'n/a', 'n/a']
['/dev/cu.HughsiPhone-WirelessiAP', 'n/a', 'n/a']
['/dev/cu.usbmodem621', 'Arduino Mega 2560', 'USB VID:PID=2341:10
SNR=64033353330351702132']
```

So all I did was scrape 'dev/cu.usbmodem621' from this output and put it after the device classifier. The script should run after that.