

Communicating with Raspberry Pi via MAVLink

Saif Aldeen Saad Obayes, IEEE Member, Saifaldeen.Saad@ieee.org

Abstract

This paper explains how to connect and configure a Raspberry Pi so that it is able to communicate with a Pixhawk flight controller using the MAVLink protocol over a serial connection. This can be used to perform additional tasks such as image recognition which simply cannot be done by the Pixhawk due to the memory requirements for storing images.

Key Words: Raspberry Pi, flight controller, MAVLink, Pixhawk.

1. Connecting the Pixhawk and Raspberry Pi

Connect the Pixhawk's TELEM2 port to the Raspberry Pi Ground, TX and RX pins as shown in the image above. More details on the individual Raspberry Pi pin functions can be found [here](#). The Raspberry Pi can be powered by connecting the red V+ cable to the +5V pin (as shown above) or from USB in (for example, using a separate 5V BEC hooked up to the USB power). Powering via USB is recommended as it is typically safer - because the input is regulated. If powering via USB, do not also connect the +5V pin as shown (still connect common ground).

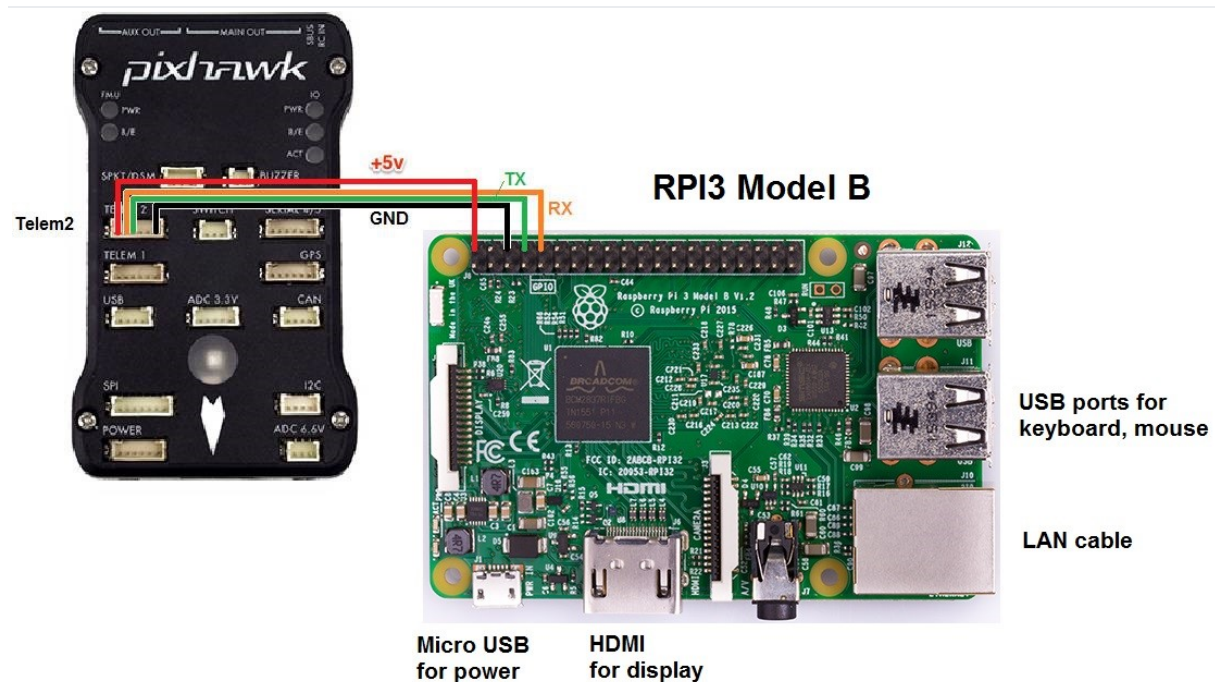


Figure 1 Connecting Pixhawk with Raspberry Pi

1.1 Setup the Raspberry Pi

The easiest way to setup the Raspberry Pi is to flash one of the existing `:ref:`APSync <apsync-intro>`` images:

1. purchase a formatted 8GB or 16GB SD card (16GB is better because some 8GB cards will not be quite large enough to fit the image) and insert into your laptop/desktop computer's SD card slot
2. download the latest [image from firmware.ardupilot.org](http://image.from.firmware.ardupilot.org). Look for the file starting with "apsync-Raspberry Pi".
3. extract the image. On Windows you may use [7-zip](#).
4. For Windows download and install Win32DiskImager and follow the [instructions here](#).
5. For [Linux follow these instructions](#).
6. For [Mac follow these instructions](#).

When extracting the contents of the compressed file on Mac you may get into an infinite loop of extraction (.xz to .cpgz and vice versa) using the default Archiver. In order to correctly extract the .img file you will need to use the Unarchive (<http://unarchiver.c3.cx/unarchiver>).

1.2 Setting up the Pixhawk

Connect to the Pixhawk with a ground station (i.e. Mission Planner) and set the following parameters:

1. `:ref:`SERIAL2_PROTOCOL <copter:SERIAL2_PROTOCOL>`` = 1 (the default) to enable MAVLink on the serial port.
2. `:ref:`SERIAL2_BAUD <copter:SERIAL2_BAUD>`` = 921 so the Pixhawk can communicate with the Raspberry Pi at 921600 baud.
3. `:ref:`LOG_BACKEND_TYPE <copter:LOG_BACKEND_TYPE>`` = 3 if you are using APSync to stream the dataflash log files to the Raspberry Pi.

1.3 Connecting to Raspberry Pi with an SSH/Telnet client

These steps assume that you have [set-up your Raspberry Pi](#) so that it is running Raspbian. These instructions are not required if you are using APSync as described above. To avoid the requirement to plug a keyboard, mouse and HDMI screen into your Raspberry Pi it is convenient to be able to connect from your Desktop/Laptop computer to your Raspberry Pi using an SSH/Telnet client such as [PuTTY](#).

1. Connect the Raspberry Pi to your local network by one of the following methods:
 - Connecting an Ethernet LAN cable from the Raspberry Pi board to your Ethernet router, or

- [Use a USB dongle to connect your Raspberry Pi to the local wireless network,](#)
or
- Connect the Ethernet LAN cable between the Raspberry Pi and your desktop/laptop computer. Open the control panel's Network and Sharing Center, click on the network connection through which your desktop/laptop is connected to the internet, select properties and then in the sharing tab, select "Allow other networks to connect through this computer's Internet connection"

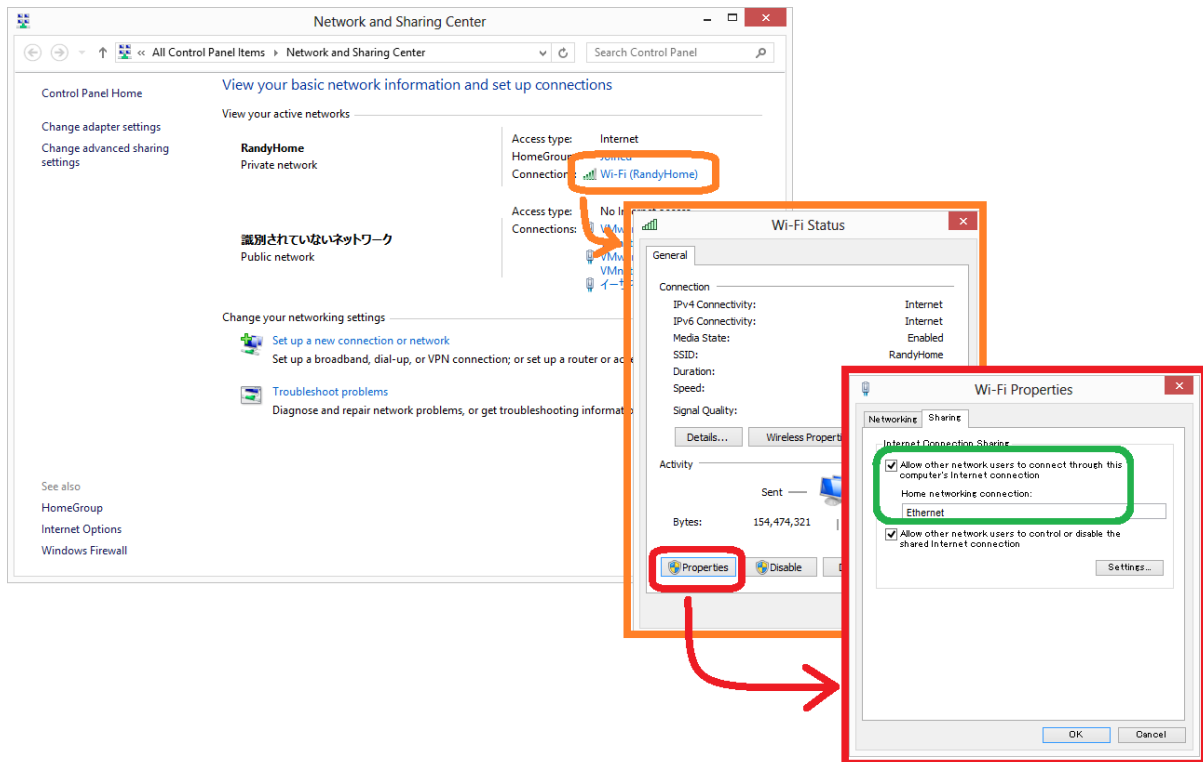


Figure 2 Connect the Ethernet LAN cable between the Raspberry Pi and desktop or laptop computer

2. Determine the Raspberry Pi IP address:

- If you have access to the Raspberry Pi terminal screen (i.e. you have a screen, keyboard, mouse connected) you can use the `/sbin/ifconfig` command.
- If your Ethernet router has a web interface it may show you the IP address of all connected devices.



Figure 3 Determine the Raspberry Pi IP address

3. Connect with [Putty](#):

If all goes well you should be presented with the regular login prompt to which you can enter the username/password which defaults to pi/raspberry

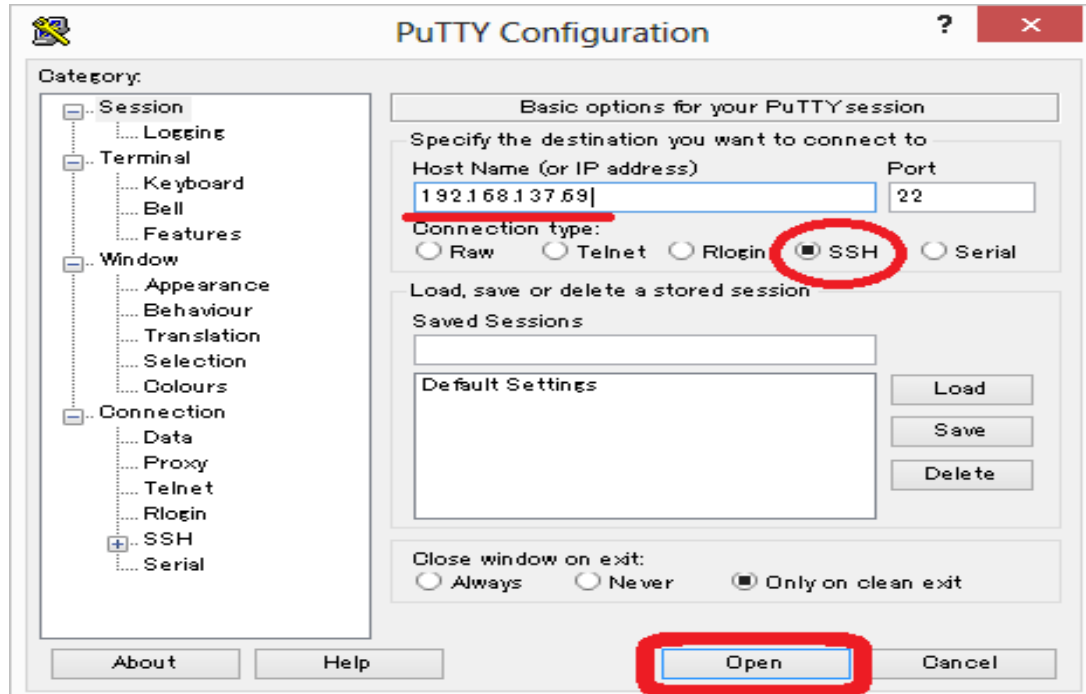


Figure 4 Putty software configurations.

1.4 Install the required packages on the Raspberry Pi

Log into the Raspberry Pi board (default username password is pi/raspberry) and check that its connection to the internet is working:

```
ping google.com
```

OR

```
ping 173.194.126.196
```

If the first fails but the second succeeds, then there is a problem with the DNS server that your Raspberry Pi is attempting to use. Please edit the `/etc/resolv.conf` file and add the IP address of a nearby DNS server. During the creation of this wiki, the first two parts of the desktop machine's IP address plus ".1.1" worked. To stop other processes from later updating this file you may wish to run the `chattr +i /etc/resolv.conf` command (this can be undone later with `chattr -i /etc/resolv.conf`). That sets the "immutable" bit on `resolv.conf` to prevent other software from updating it.

After the internet connection is confirmed to be working install these packages:

```
sudo apt-get update #Update the list of packages in the software center
```

```
sudo apt-get install screen python-wxgtk2.8 python-matplotlib python-opencv python-pip python-numpy  
python-dev libxml2-dev libxslt-dev
```

```
sudo pip install future
```

```
sudo pip install pymavlink
```

```
sudo pip install mavproxy
```

The packages are [:ref: mostly the same as when setting up SITL <setting-up-sitl-on-windows>](#). Reply 'y' when prompted re additional disk space.

1.5 Disable the OS control of the serial port

Use the Raspberry Pi configuration utility for this.

Type:

```
sudo raspi-config
```

And in the utility, select "Advanced Options":

RasPiConfiguration Utility: Serial Settings: Advanced Options

And then "Serial" to disable OS use of the serial connection:

Reboot the Raspberry Pi when you are done.

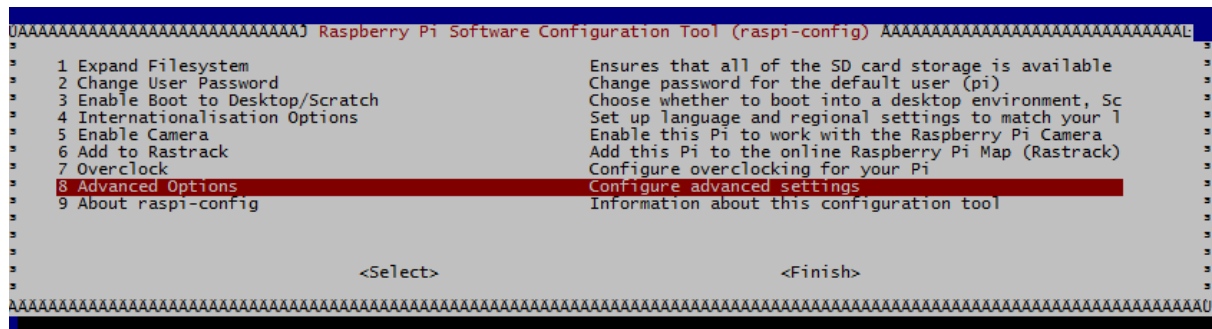


Figure 5 Raspberry Pi configuration utility

2. Testing the connection

To test the Raspberry Pi and Pixhawk are able to communicate with each other first ensure the Raspberry Pi and Pixhawk are powered, then in a console on the Raspberry Pi type:

```
sudo -s
mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --aircraft MyCopter
```

On newer versions of Raspberry Pi 3 the uart serial connection may be disabled by default. In order to enable serial connection on the Raspberry Pi edit **/boot/config.txt** and set `enable_uart=1`. the build-in serial port is `/dev/ttyS0`.

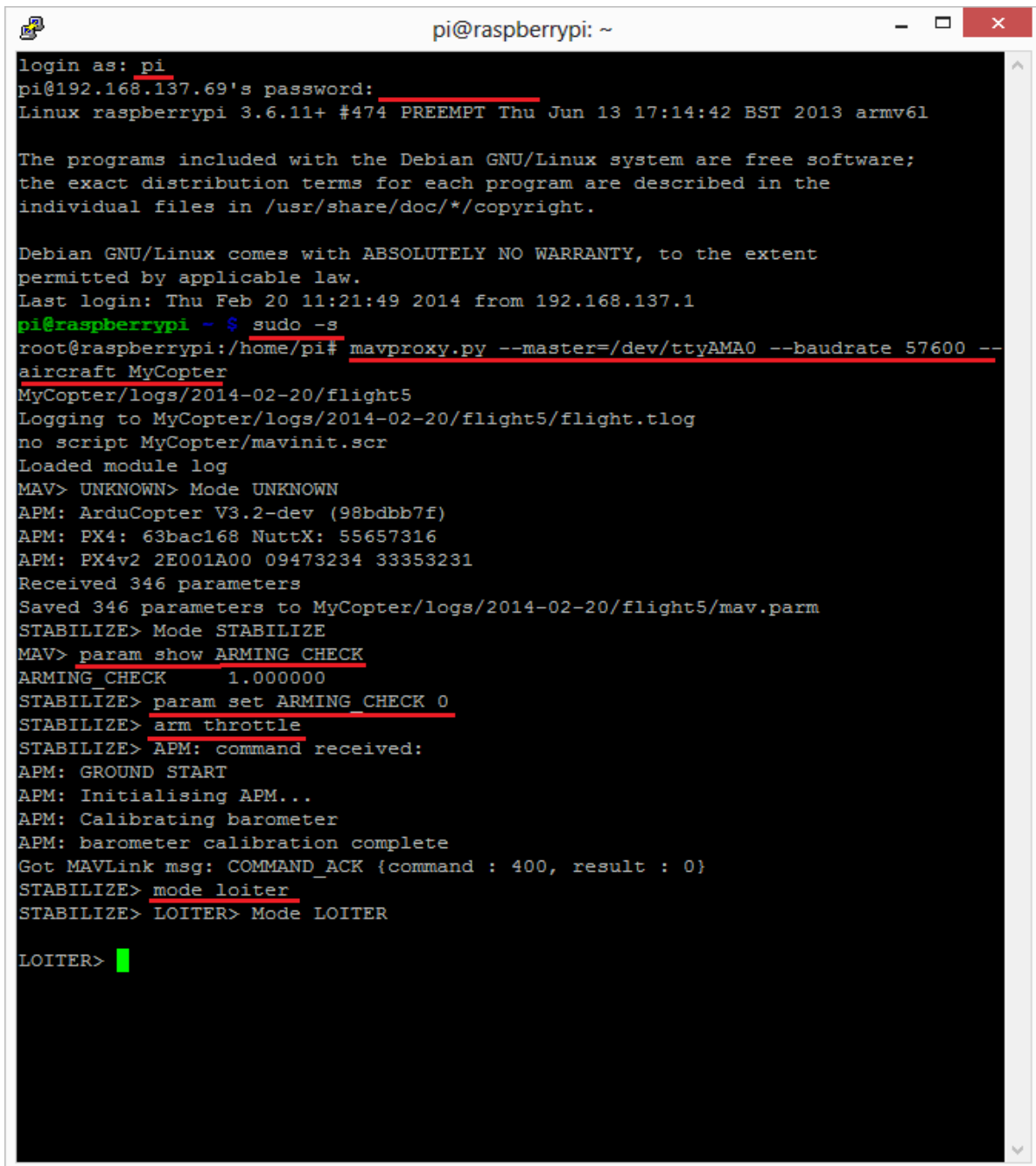
Once MAVProxy has started you should be able to type in the following command to display the `ARMING_CHECK` parameters value

```
param show ARMING_CHECK
param set ARMING_CHECK 0
arm throttle
```

If you get an error about not being able to find log files or if this example otherwise doesn't run properly, make sure that you haven't accidentally assigned these files to another username, such as Root.

Entering the following at the Linux command line will ensure that all files belong to the standard Pi login account:

```
sudo chown -R pi /home/pi
```

A terminal window titled 'pi@raspberrypi: ~' showing the process of connecting to a MAVLink-enabled aircraft. The user logs in as 'pi' and runs 'sudo -s' to become root. Then, they run 'mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --aircraft MyCopter'. The terminal output shows the MAVProxy software initializing, receiving parameters from the aircraft, and switching to 'LOITER' mode. A green cursor is visible on the 'LOITER>' line.

```
login as: pi
pi@192.168.137.69's password:
Linux raspberrypi 3.6.11+ #474 PREEMPT Thu Jun 13 17:14:42 BST 2013 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Feb 20 11:21:49 2014 from 192.168.137.1
pi@raspberrypi ~ $ sudo -s
root@raspberrypi:/home/pi# mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --
aircraft MyCopter
MyCopter/logs/2014-02-20/flight5
Logging to MyCopter/logs/2014-02-20/flight5/flight.tlog
no script MyCopter/mavinit.scr
Loaded module log
MAV> UNKNOWN> Mode UNKNOWN
APM: ArduCopter V3.2-dev (98bdbb7f)
APM: PX4: 63bac168 NuttX: 55657316
APM: PX4v2 2E001A00 09473234 33353231
Received 346 parameters
Saved 346 parameters to MyCopter/logs/2014-02-20/flight5/mav.parm
STABILIZE> Mode STABILIZE
MAV> param show ARMING_CHECK
ARMING_CHECK 1.000000
STABILIZE> param set ARMING_CHECK 0
STABILIZE> arm throttle
STABILIZE> APM: command received:
APM: GROUND START
APM: Initialising APM...
APM: Calibrating barometer
APM: barometer calibration complete
Got MAVLink msg: COMMAND_ACK {command : 400, result : 0}
STABILIZE> mode loiter
STABILIZE> LOITER> Mode LOITER

LOITER> █
```

Figure 6 Testing the connection

3. Configure MAVProxy to always run

To setup MAVProxy to start whenever the Raspberry Pi is restarted open a terminal window and edit the `/etc/rc.local` file, adding the following lines just before the final "exit 0" line:

```
(
date
echo $PATH
PATH=$PATH:/bin:/sbin:/usr/bin:/usr/local/bin
```

```
export PATH
cd /home/pi
screen -d -m -s /bin/bash mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --aircraft
MyCopter
) > /tmp/rc.log 2>&1
exit 0
```

Whenever the Raspberry Pi connects to the Pixhawk, three files will be created in the /home/pi/MyCopter/logs/YYYY-MM-DD directory:

- **mav.parm** : a text file with all the parameter values from the Pixhawk
- **flight.tlog** : a telemetry log including the vehicles altitude, attitude, etc which can be opened using the mission planner (and a number of other tools)
- **flight.tlog.raw** : all the data in the .tlog mentioned above plus any other serial data received from the Pixhawk which might include non-MAVLink formatted messages like startup strings or debug output

If you wish to connect to the MAVProxy application that has been automatically started you can log into the Raspberry Pi and type:

```
sudo screen -x
```

To learn more about using MAVProxy please read the [MAVProxy documentation](#).

It is also worth noting that MAVProxy can do a lot more than just provide access to your Pixhawk. By writing python extension modules for MAVProxy you can add sophisticated autonomous behaviour to your vehicle. A MAVProxy module has access to all of the sensor information that your Pixhawk has, and can control all aspects of the flight. To get started with MAVProxy modules please have a look at the [existing modules](#) in the MAVProxy source code.

4. Installing DroneKit on Raspberry Pi

The most up-to-date instructions for [Installing DroneKit](#) on Linux are in the DroneKit-Python documentation. This information is a summary, and might go out of date.

To install DroneKit-Python dependencies (most of which will already be present from when you installed MAVProxy) and set DroneKit to load when MAVProxy starts:

```
sudo apt-get install python-pip python-dev python-numpy python-opencv python-serial
python-pyparsing python-wxgtk2.8 libxml2-dev libxslt-dev
sudo pip install droneapi
echo "module load droneapi.module.api" >> ~/.mavinit.scr
```

Then open the MAVProxy terminal in the location where your DroneKit script is located and start an example:

```
MANUAL> api start vehicle_state.py
```

If you get a warning that droneapi module has not loaded, you can do so manually in MAVProxy:

```
MANUAL> module load droneapi.module.api
```

5. Connecting with the Mission Planner

The Pixhawk will respond to MAVLink commands received through Telemetry 1 and Telemetry 2 ports (see image at top of this page) meaning that both the Raspberry Pi and the regular ground station (i.e. Mission planner, etc) can be connected. In addition it is possible to connect the Mission Planner to the MAVProxy application running on the Raspberry Pi [:ref: similar to how it is done for SITL <setting-up-sitl-on-windows connecting with the mission planner>`](#).

Primarily this means adding an `--out <ipaddress>:14550` to the MAVProxy startup command with the being the address of the PC running the mission planner. On windows the ipconfig can be used to determine that IP address. On the computer used to write this wiki page the MAVProxy command became:

```
mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --out 192.168.137.1:14550 --aircraft MyCopter
```

Connecting with the mission planner is shown below:



Figure 7 the mission planner dashboard

References:

1. Saif Aldeen Saad Obayes Al-Kadhim, and et al., Prototype Wireless Controller System based on Raspberry Pi and Arduino for Engraving Machine, UKSim-AMSS 19th International Conference on Modelling & Simulation, DOI: 10.1109/UKSim.2017.20.
2. Saif Aldeen Saad Obayes Al-Kadhim, Industrial Workshop Based on Internet of Things: Automated Manufacturing Systems Technology, Noor Publishing (August 1, 2017), ISBN-10: 3330965754, ISBN-13: 978-3330965751.
3. Saif Aldeen Saad Obayes Al-Kadhim, and et al., CNC Machine Based on Embedded Wireless and Internet of Things for Workshop Development **Article in** IJCDS Journal Volume 6(Issue 4):205 · July 2017, DOI: 10.12785/ijcds/060406.