nccgroup
freedom from doubt

An NCC Group Publication

Creation of WiMap, the Wi-Fi Mapping Drone

Prepared by:

Michael Johnson

Contents

1 Introduction

The objective of this project is to detail the methods used to create, from parts, a hexacopter capable of being controlled over 3/4G and equipped to perform wireless and infrastructure assessments. While the hardware used can be acquired from regular retail sources, much of the software is made up of scripts written as part of the project. These are provided as part of this tutorial, but details of the scripts are not provided, other than giving examples of them in use. The flight-control software is off the shelf, but has been integrated into the project using a number of boot-up scripts on the drone-carried Raspberry Pi.

The first of these scripts, WiMap, starts three screen sessions; the first screen session attempts to connect to the VPS Bridge using MavLink, this allows us to control the hexacopter over the 3/4G network and to receive telemetry.

The second session sets the wireless interface into monitor mode, and the script then spawns a new process so the network interface can channel hop through 1-14.

The core logic of WiMap uses Python's Scapy library to read packets from the network interface, specifically looking for available access points in range. If the main script detects that there are not enough satellites in view when it sees a new access point it will not plot them. However if it detects that there is a 3D GPS fix, it will then write the network information, including the BSSID and GPS locations at which the Wi-Fi access point was seen, to a temporary cache file for later processing.

The third screen session starts another Python script named SendData, which then attempts to read the temporary cache directory every five seconds, to detect if the first script has written any networks. If the first script has written networks, the second script will connect to the remote virtual server's MySQL database and replace the data in the database. The reason replacement is being used is that networks may get changed, so if the mapping tool goes past the network multiple times it can update the information in the database.

The final script that is run on the device is a Cron job that will check for an SSH process. If an SSH process is not found, the script will create a reverse SSH connection to the remote VPS Bridge mapping server for remote administration (don't want our drone payload being hacked).

The remote server's web interface currently includes:

- Google map with colour corresponding to authentication types: GREEN/WPA YELLOW/WEP OPN/RED.
- Options to update, add, and remove network information.
- A pie chart of the gathered networks, with statistics.
- The ability to export networks in CSV format.

It should be possible for an interested reader to follow the steps and build a fully functional Pentestacopter. Learning to fly it is another matter.

2 Requirements

2.1 Small Unmanned Aircraft Laws and Regulations

The use of various types of unmanned aircraft, popularly known as drones, has increased rapidly in recent years - both for private leisure use, and for commercial aerial work. Unmanned aircraft are generally fitted with cameras, unlike traditional remote-controlled model aircraft which have been used by enthusiasts for many years. As such, drones are likely to be operated in a way that may pose a greater risk to the general public and other aircraft. Unlike manned or model aircraft, there are no established operating guidelines, so operators may not be aware of the potential dangers or indeed the responsibility they have towards avoiding collisions. Anyone flying a drone either recreationally or commercially has to take responsibility for doing so safely.

Aircraft of 20 kg or less are referred to as “small unmanned aircraft”. The requirements for these are a little less stringent and are covered under Articles 166 and 167 of the Air Navigation Order 2009 (reproduced below).

2.1.1 Article 166

1. A person shall not cause or permit any article or animal (whether or not attached to a parachute) to be dropped from a small aircraft so as to endanger persons or property.
2. The person in charge of a small unmanned aircraft may only fly the aircraft if reasonably satisfied that the flight can safely be made.
3. The person in charge of a small unmanned aircraft must maintain direct, unaided visual contact with the aircraft sufficient to monitor its flight path in relation to other aircraft, persons, vehicles, vessels and structures for the purpose of avoiding collisions.
4. The person in charge of a small unmanned aircraft which has a mass of more than 7 kg excluding its fuel but including any articles installed in or attached to the aircraft at the commencement of its flight, must not fly such an aircraft:
   A. In Class A, C, D or E airspace unless the permission of the appropriate air traffic control unit has been obtained.
   B. within an aerodrome traffic zone during the notified hours of watch of the air traffic unit (if any) at that aerodrome unless the permission of any such air traffic control unit has been obtained.
   C. At a height of more than 400 feet above the surface unless it is flying in airspace described in sub-paragraph (a) or (b) above and in accordance with the requirements for that airspace.
5. The person in charge of a small unmanned aircraft must not fly such an aircraft for the purposes of aerial work except in accordance with a permission granted by the CAA.

2.1.2 Article 167

1. The person in charge of a small unmanned surveillance aircraft must not fly the aircraft in any of the circumstances described in paragraph (2) except in accordance with a permission issued by the CAA.
2. The circumstances referred to in paragraph (1) are:
   A. over or within 150 metres of any congested area;
   B. over or within 150 metres of an organised open-air assembly of more than 1,000 persons;
   C. Within 50 metres of any vessel, vehicle or structure which is not under the control of the person in charge of the aircraft.
   D. Subject to paragraphs (3) and (4), within 50 metres of any person.
3. Subject to paragraph (4), during take-off or landing, a small unmanned surveillance aircraft must not be flown within 30 metres of any person.
4. Paragraphs (2) (d) and (3) do not apply to the person in charge of the small unmanned surveillance aircraft or a person under the control of the person in charge of the aircraft.
5. In this article ‘a small unmanned surveillance aircraft’ means a small unmanned aircraft which is equipped to undertake any form of surveillance or data acquisition.

2.2 Required Hardware and Tools

During the build process of the hexacopter, a number of pieces of hardware and tools will be required to ensure everything is working as intended.

The list below covers the equipment used at the time of writing this paper.

- Laptop or desktop computer
- 1 Meter USB extension cable
- 3/4G USB network interface.
- Raspberry Pi 2
- Micro SD card
- Wireless USB network interface
- APM 2.6 with GPS, compass and power module
- DJI F550 frame kit
- 4x Turnigy 5000mAh 4S 30C Lipo batteries.
- Walkera DEVO 7 7-channel 2.4Ghz transmitter
- Walkera Devo RX701 2.4Ghz 7ch receiver
- Smartstick USB portable battery.
- 2.0mm hex wrench - for frame and motor installation.
- Lock tight screw glue - for fastening screws.
- Nylon cable ties
- Superglue
- Diagonal cutting pliers
- Insulating tape – for insulating self-made connectors
- Foam double-sided adhesive tape - for fixing receiver, controller, and other modules
- Soldering iron and wires - for connecting ESCs' power cables to bottom plates

3 Build and Configuration

3.1 Wiring ESCs to the Baseplate

To start the build process the ESCs (electronic speed controller) first need soldering to the power pads located on the bottom plate, as shown in figure 1. It should be noted that the rotation of each motor should be set figure 2 shows; if at any point during the build process the motors are not turning in the correct direction, the wires will need to be unplugged and switched over.

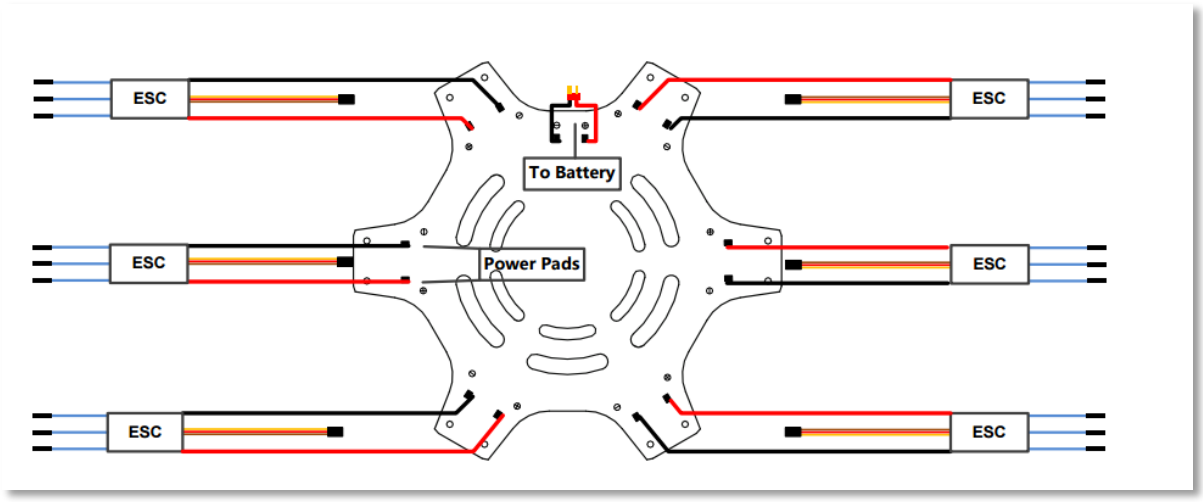

Figure 1: ESC Configuration

3.2 Assembly of the F550 Frame

The next step is to assemble the hexacopter's frame. It should be noted that the arms with labels ① ② are the front of craft, (the two front arms are normally a different colour from the others supplied in the same packaging) and the arms labelled ④⑤ are at the rear of the craft. The motors on the arms ①③⑤ rotate counter-clockwise, while the motors on arms ②④⑥ rotate clockwise. For best performance use carbon fibre upgrades of the 1038 propellers

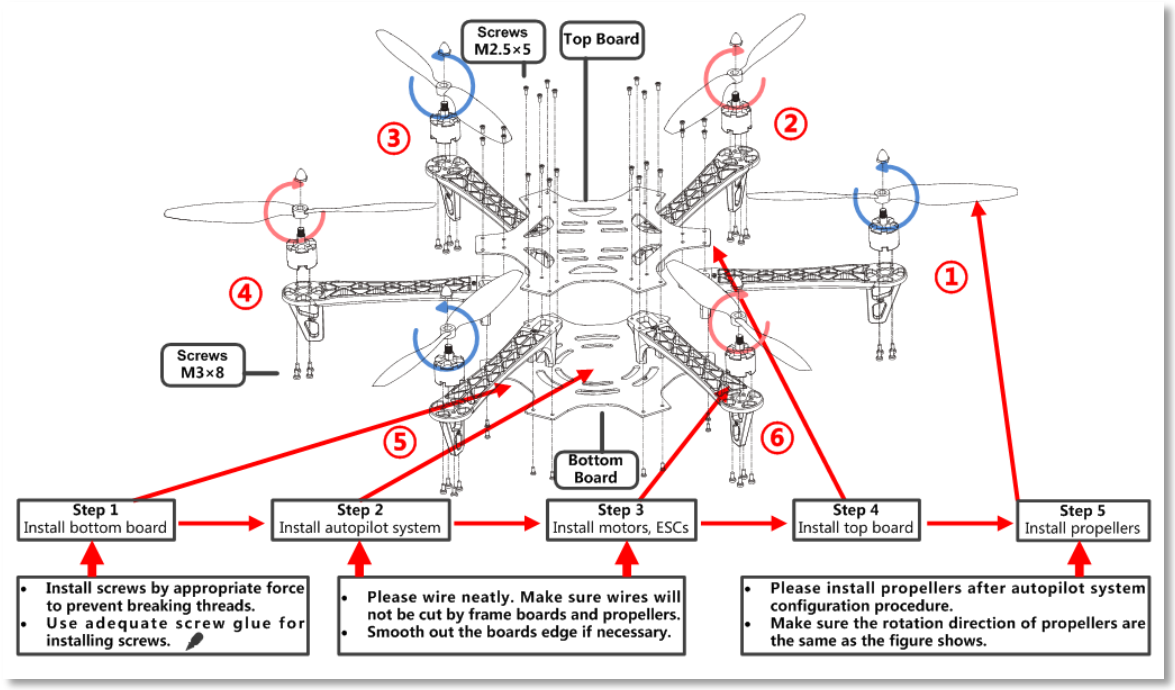

Figure 2: F550 Frame Assembly

3.3 Connecting to the APM from the Raspberry Pi

After the frame has been successfully assembled, a connection to the APM from the Raspberry PI needs to be established. This can be achieved by creating a simple cable with the parts from the MinimOSD (Minim on-screen display) and attaching to the Raspberry Pi as shown in figure 3. Ensure the positive wire is not connected to the APM from the Raspberry Pi, as this powers the Raspberry Pi but does not supply enough power for the 3G USB network interface. It can also reboot the APM if the Raspberry Pi has a kernel panic or some other issue, and as it would be using the MinumOSD port up it would no longer be possible to get a video feed between the ground station and the hexacopter. By using the Raspberry Pi it is possible to attach a camera and stream the data over a 3G network connection.

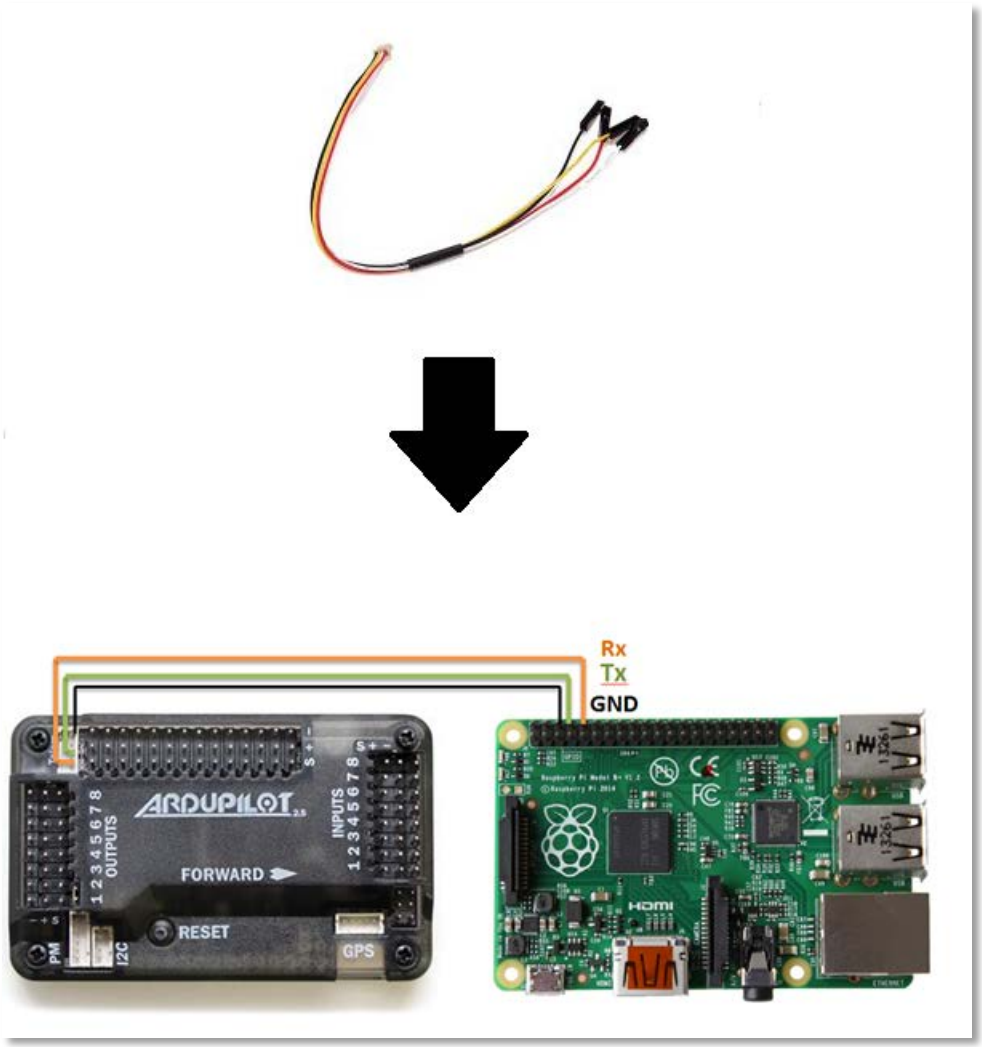

Figure 3: APM to Raspberry Pi Connection

3.4 Binding the Receiver and Setting Fixed ID on the transmitter

1. When binding the transmitter and receiver, ensure nothing is plugged in to the BATT port on the RX701 receiver. Then proceed to turn on the transmitter and go to MODEL > FIXED ID and then press ENT. If it is showing the code, then press ENT again, press one more time and it should now be showing RUN. Press "R" to choose NO and then press ENT again. It now should be showing FIXID; press "R" again to choose OFF, then press EXT, and turn off the transmitter.

2. Plug the bind plug into the BATT port on the RX701 receiver, with the plug still in the BATT port and the hexacopter being powered up. The receiver will start flashing red slowly; this means old code has been erased. The hexacopter can now be unplugged, and the bind plug needs to be removed.

3. The next step is to activate the fixed ID function within the DEVO 7 transmitter and turn on the transmitter. Once you have done this, ensure the throttle stick is all the way down, all trimming is neutral, both the corner FMOD switch and throttle switch are off (pointing backward) and all switches on the transmitter are pointing upwards, and then turn off the transmitter.

4. Connect the battery to the power module so the receiver will start flashing, and then place the drone on a flat surface. Turn on the transmitter, and you should see black box running on the transmitter screen. Do no touch anything until the transmitter stops flashing (around seven to ten seconds) or you will break the searching mode. Once the RX701 receiver has solid light, binding has completed.

5. Turn on the transmitter again, and go to MODEL > FIXED ID. Turn fixed ID ON, and then press the DN button to confirm the code. Press ENT, and press ENT again to confirm. It will ask you to RUN; choose YES and press ENT. From now on your receiver is bound to this memory on your DEVO 7.

DEVO 7 Configuration Parameters

Use the following parameters on the DEVO 7 transmitter for a default configuration; these may need tweaking later;

```
[MODEL] > [TYPE] > AERO
[MODEL] > [INPUT] > FM SW = FMD
[MODEL] > [INPUT] > FMTRM = COMM
[MODEL] > [INPUT] > HLDSW = HOLD
[MODEL] > [OUTPUT] > GEAR = FMD and ACT
[MODEL] > [OUTPUT] > FLAP = MIX and ACT
[MODEL] > [OUTPUT] > AUX2 = AUX2 and ACT
[MODEL] > [AMPLI] > +20
[FUNCTION] > [REVSW] > GEAR = REV
[FUNCTION] > [REVSW] > FLAP = REV
[FUNCTION] > [REVSW] > AUX2 = REV
[FUNCTION] > [TRVAD] > GEAR = +0.0% / -83.5%
[FUNCTION] > [TRVAD] > FLAP = U88.0% / D88.0%
[FUNCTION] > [SUBTR] > GEAR = -5.5%
[FUNCTION] > [SUBTR] > FLAP = D5.0%
[FUNCTION] > [SAFE] > GEAR = SAFE / +40%
```

3.5 Attaching the Power Module

The APM 2.6 has a dedicated connector for attaching the 3DR power module.

This is useful because it provides a stable 5.37 volt and 2.25 amp power supply, which reduces the chances of a brown-out. This makes it possible to monitor the battery's voltage and current, and to trigger a return-to-launch when the voltage becomes low or the total power consumed during the flight approaches the battery's capacity, and allows the autopilot firmware to more accurately compensate for the interference on the compass from other components

The power module can be attached by plugging the small plug from the 3DR power module into the PM socket on the APM.

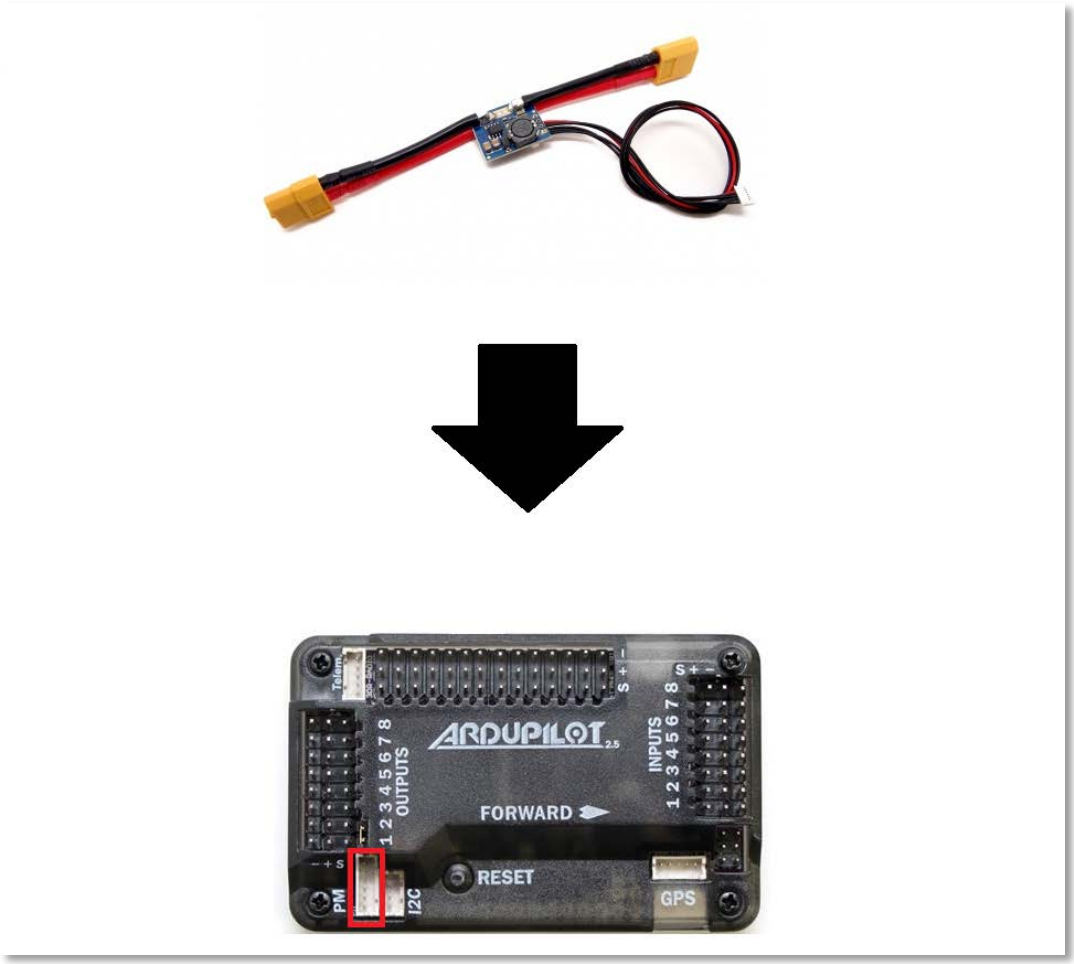

Figure 4: Connection of Power Module

3.6 Attaching the GPS and Compass Module

The GPS and compass are attached to the hexacopter towards the end of the hardware build, as the wires are short so there is little room for movement. It is also best to stick down the compass while configuring the hexacopter through the Mission Planner interfaces, as you will be able to see if the compass is pointing north correctly.

Please note that the compass supplied with the APM kit may have the compass installed upside down. This can be taken into account by using the Mission Planner configuration utilities and reversing the directions of the GPS.

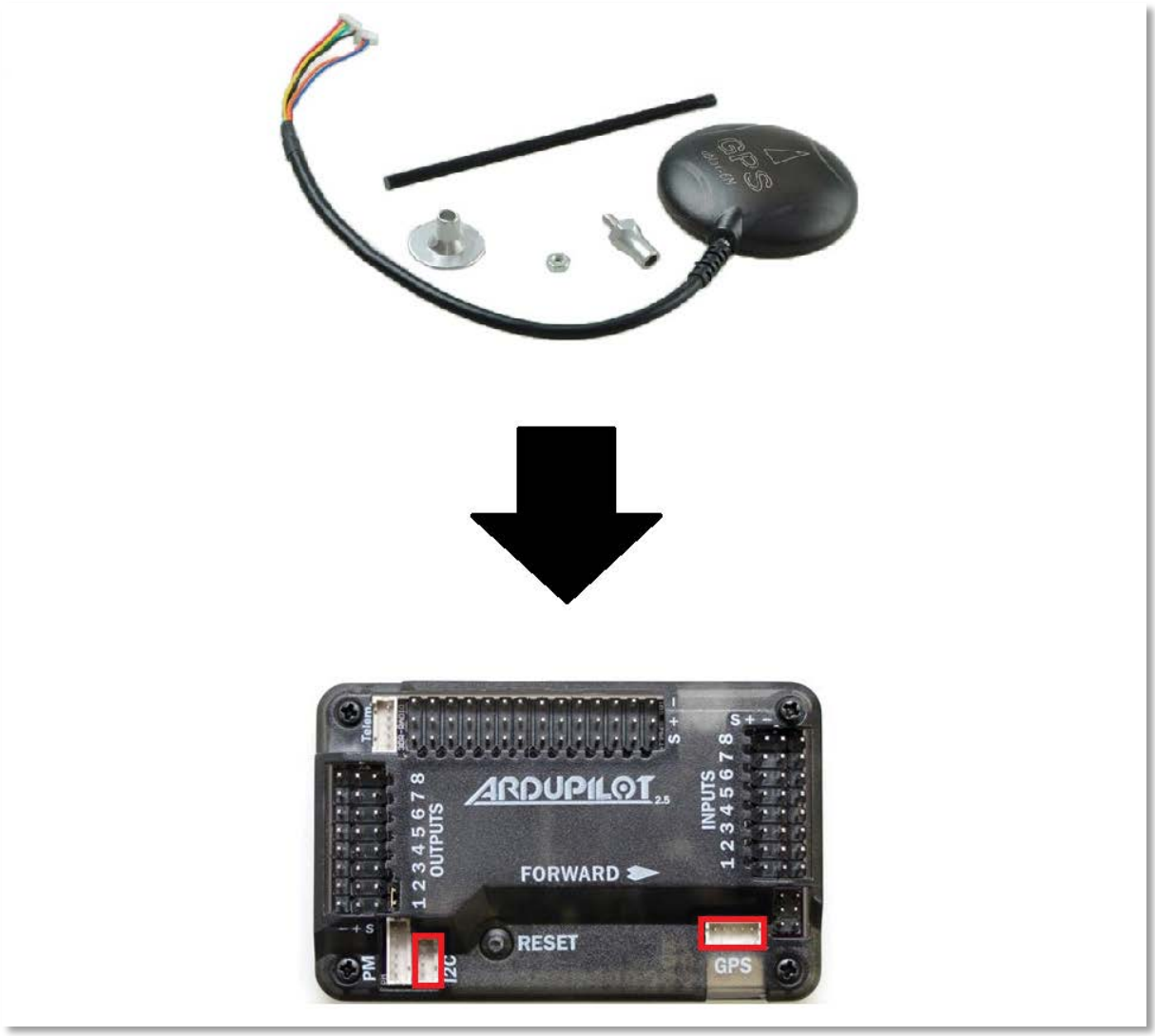

Figure 5: Connection of GPs Module

When all the steps above have been completed, the APM, the Raspberry Pi, and the power supply for the Raspberry Pi can be zip-tied to the F550 top plate; this will allow space for an optional gimbal, or for carrying multiple batteries on the bottom of the hexacopter.

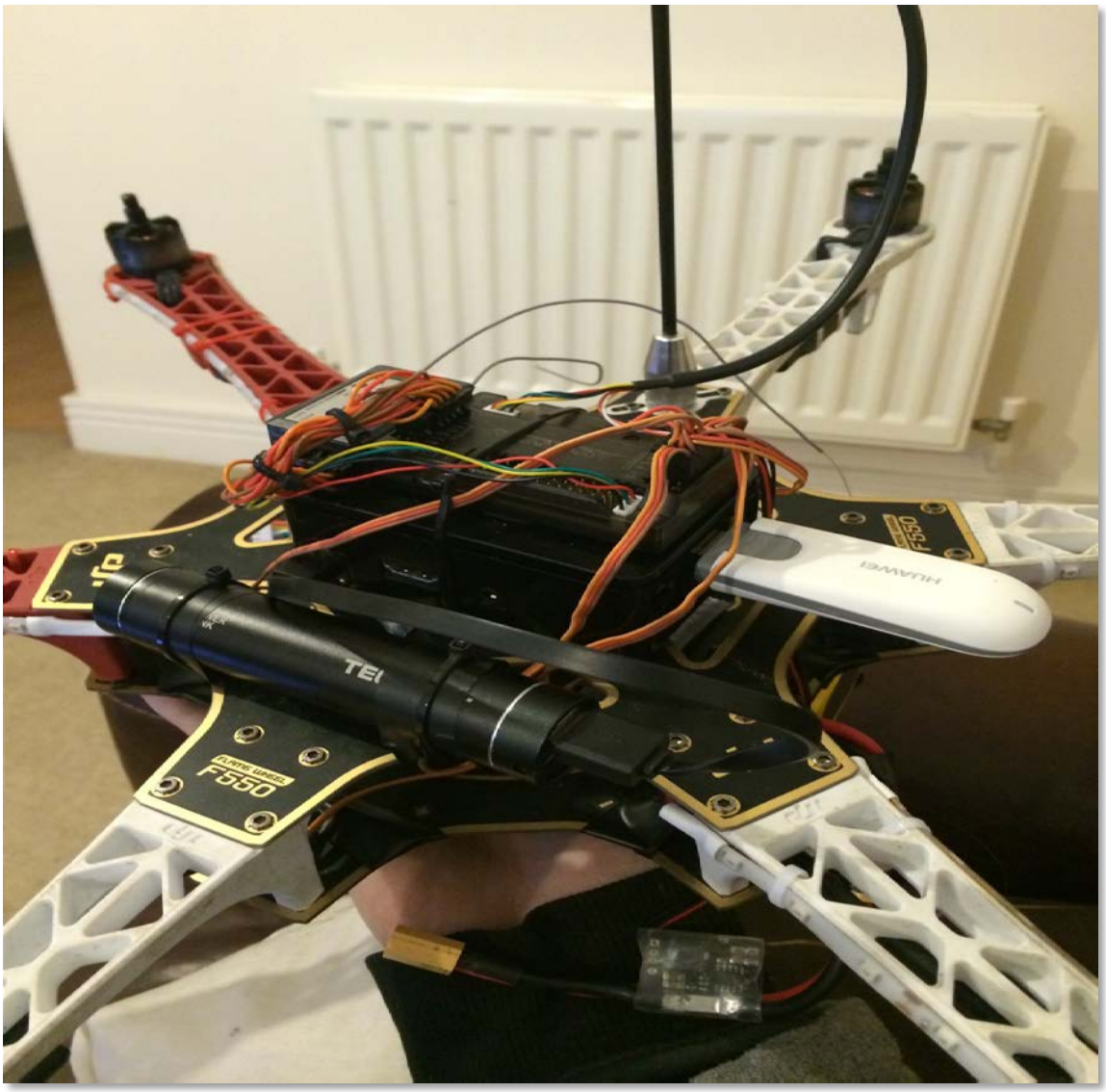

Figure 6: Completion of Hardware Build

3.7 Flashing Firmware on the APM

Flashing the firmware onto the APM has to be completed over USB for safety reasons, as if a drop in network connectivity occurs it could cause the firmware to fail and the APM to malfunction. Flashing the firmware onto the APM tells it what frame type that is in use.

To flash the firmware on to the device, simply connect the hexacopter to the laptop via USB and click the connect button on the top right hand corner of mission planner, then navigate to “Initial setup” and select “Install firmware” from the left-hand menu. This will list a selection of frame types; select APMCopter V3.3.2 Hexa. The firmware will be downloaded and flashed on to the APM

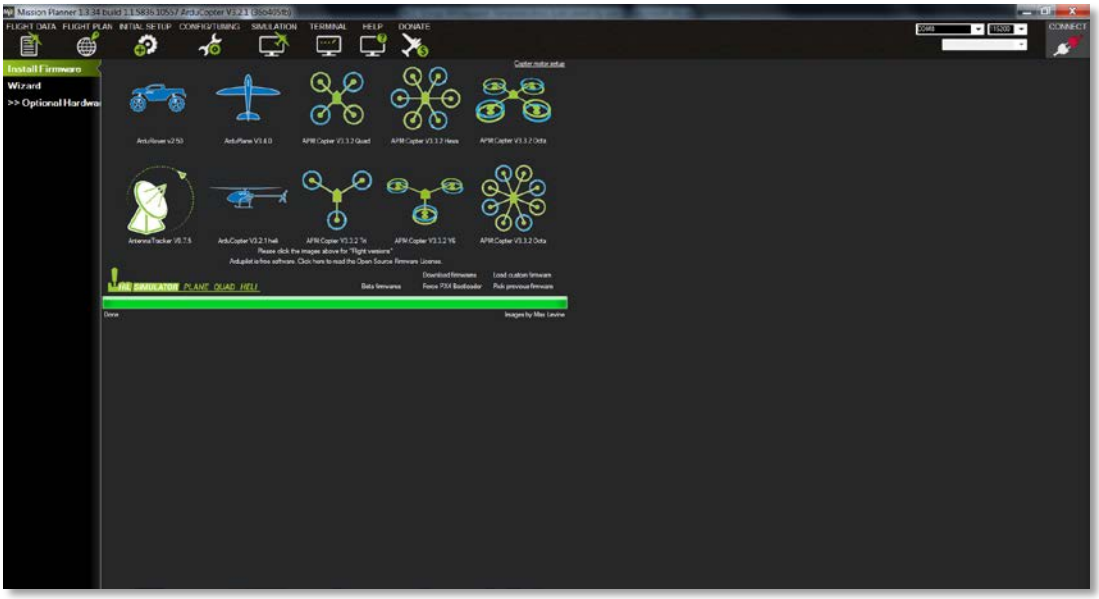

Figure 7: Flashing Firmware

3.8 Configuring APM with Mission Planner

The APM can be configured to have the best parameter setting for the build, by using the on-screen wizard within the Mission Planner and then tweaking parameters later on.
This will allow configuration of:

- Frame type
- Accelerometer
- Compass
- Radio transmitter
- ESC calibration
- Flight modes
- Failsafe

Figure 8: Configure APM with Mission Planner

3.9 Install Kali Linux on the Raspberry Pi

After completion of the assembly and configuration of the hardware, the Raspberry Pi requires an operating system. The system we will discuss in this paper is Kali Linux, as it has all the required tools for infrastructure and wireless assessments. The current version of Kali Linux 2.1 for the Raspberry Pi 2 was found to suffer kernel panics, so the Raspberry Pi 1 image was used.

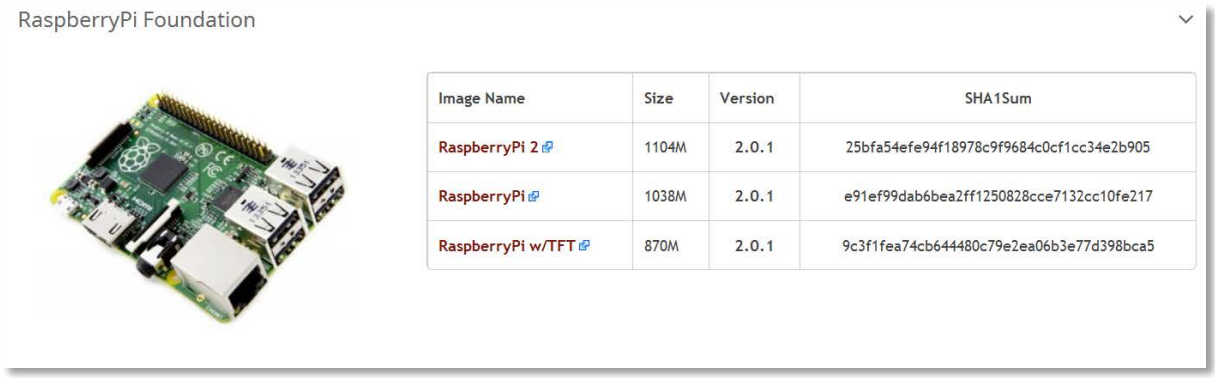

Figure 9: Kali Linux for Raspberry Pi

Linux has a built-in utility to download files from the web, "wget". This can be seen in use, downloading the Kali image, below:

```
root@kali-Laptop:~# wget http://images.kali.org/kali-2.0.1-rpi.img.xz
--2015-11-20 19:22:02--  http://images.kali.org/kali-2.0.1-rpi.img.xz
Resolving images.kali.org (images.kali.org)... 199.189.86.7, 188.138.17.16
Connecting to images.kali.org (images.kali.org)|199.189.86.7|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1064607380 (1015M) [application/octet-stream]
Saving to: 'kali-2.0.1-rpi.img.xz'

kali-2.0.1-rpi.img.xz
100%[=============================================>]   1015M   942KB/s   in
23m 10ss

2015-11-20 19:45:12 (748 KB/s) - 'kali-2.0.1-rpi.img.xz' saved
[1064607380/1064607380]

root@kali:~#
```

The Kali image comes as a compressed package when downloaded from the website, so it is necessary to unzip it:

```
root@kali-Laptop:~# unxz kali-2.0.1-rpi.img.xz
root@kali-Laptop:~#
```

Once the image has been extracted, locate the partition on the SD card so the image can be cloned onto it.

```
root@kali-Laptop:~# fdisk  -l

Disk /dev/sda: 30 GiB, 32212254720 bytes, 62914560 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xb2d1b90f

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *        2048 60262399 60260352 28.8G 83 Linux
/dev/sda2       60264446 62912511  2648066  1.3G  5 Extended
/dev/sda5       60264448 62912511  2648064  1.3G 82 Linux swap / Solaris

Disk /dev/sdj: 7.4 GiB, 7948206080 bytes, 15523840 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000ddd94

Device     Boot  Start      End  Sectors  Size Id Type
/dev/sdj1            1   125000   125000   61M  c W95 FAT32 (LBA)
/dev/sdj2       125001 14335999 14210999  6.8G 83 Linux
```

To clone the image on to the SD card, use the built-in Linux utility DD, as shown below:

```
root@kali-Laptop:~# dd bs=4M if=kali-2.0.1-rpi.img of=/dev/sdj
1750+0 records in
1750+0 records out
7340032000 bytes (7.3 GB) copied, 933.588 s, 7.9 MB/s
root@kali-Laptop:~#
```

3.10 Installing MavLink on the Kali Linux

Once Kali has successfully booted and SSH access has been established, MAVLink must be installed, for controlling the APM and allowing the use of custom script integration:

```
root@kali-Pi:~# apt-get –y update
root@kali-Pi:~# apt-get –y install screen python-matplotlib python-opencv python-
pip python2.7-dev python-numpy
root@kali-Pi:~# pip install pymavlink
root@kali-Pi:~# pip install mavproxy
```

To test the connection between the Raspberry Pi and APM, ensure both are powered on, and then enter the following commands into the Raspberry Pi's terminal:

```
root@kali-Pi:~# mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600
Connect /dev/ttyAMA0 source_system=255
Log Directory:
Telemetry log: mav.tlog
MAV> Waiting for heartbeat from /dev/ttyAMA0
fence breach
online system 1
GUIDED> Mode GUIDED
APM: ArduCopter V3.2.1 (36b405fb)
APM: Frame: HEXA
Flight battery 100 percent
Received 306 parameters
Saved 306 parameters to mav.parm

GUIDED>
```

Once MAVProxy has started it should be possible to type in the following command to display the ARMING_CHECK parameters value.

Ensure the transmitter has been bound and powered on, a LIPO battery is connected to the power module, propellers are not attached, and any wires are away from the moving parts, before attempting the following commands:

```
STABILIZE> param show ARMING_CHECK
STABILIZE> ARMING_CHECK     1.000000
STABILIZE> param set ARMING_CHECK 0
STABILIZE> arm throttle
STABILIZE> APM: Calibrating barometer
APM: barometer calibration complete
APM: Initialising APM...
Got MAVLink msg: COMMAND_ACK {command : 400, result : 0}
ARMED

STABILIZE> disarm
STABILIZE> Got MAVLink msg: COMMAND_ACK {command : 400, result : 0}
DISARMED

STABILIZE>
```

3.11 Installing WiMap on the Kali Linux

When the MavLink connection has been successfully established, a few Python libraries are required. The Python Scapy library is necessary for WiMap to read 802.11 packets such as management frames, which is how WiMap identifies wireless access points:

```
root@kali-Pi:~# apt-get install python-scapy python-mysqldb
```

The Python Dronekit library is necessary for WiMap to obtain the current location of the hexacopter. Using this library makes it possible to use the APM GPS module, thus saving weight, as a secondary GPS module would normally be required for the Raspberry Pi:

```
root@kali-Pi:~# git clone https://github.com/dronekit/dronekit-python
Cloning into 'dronekit-python'...
remote: Counting objects: 4465, done.
remote: Compressing objects: 100% (66/66), done.
remote: Total 4465 (delta 27), reused 0 (delta 0), pack-reused 4395
Receiving objects: 100% (4465/4465), 2.94 MiB | 521.00 KiB/s, done.
Resolving deltas: 100% (2632/2632), done.
Checking connectivity... done.
root@kali-Pi:~#
root@kali-Pi:~# cd dronekit-python/
root@kali-Pi:~/dronekit-python# python setup.py build
root@kali-Pi:~/dronekit-python# python setup.py install
```

To ensure all the Python libraries are installed and working correctly, start a screen session that will then start MavProxy to allow connections on port 57600:

```
root@kali-Pi:~# screen –dm  -S MavLink -s /bin/bash mavproxy.py --
out=127.0.0.1:57600 --master=/dev/ttyAMA0 --baudrate=57600
```

This Python script is for testing a connection to the APM and retrieving GPS locational data, to ensure WiMap will be able to find its current location and plot the networks in range.

```
root@kali-Pi:~# cat  test-connection.py
from dronekit import connect

print 'Connecting to vehicle on: 127.0.0.1:57600'
vehicle = connect('127.0.0.1:57600', wait_ready=True, heartbeat_timeout=30,
source_system=255)

if vehicle.location.global_frame.lat or vehicle.location.global_frame.alt is not
None:
      Lat = vehicle.location.global_frame.lat
      Lon = vehicle.location.global_frame.lon
      Alt = vehicle.location.global_frame.alt
      Fix = vehicle.gps_0.fix_type
      NumSat  = vehicle.gps_0.satellites_visible
      print Lat,Lon,Alt,Fix,NumSat
root@kali-Pi:~#
```

If run successfully, the connection script should output the current attitude, longitude, altitude, GPS fix type, and satellites in view. If the script returns back 0.0 0.0 or something similar to this, it is still successful but the APM currently has no GPS fix:

```
root@kali-Pi:~# python test-connection.py
Connecting to vehicle on: 127.0.0.1:57600
>>> Frame: HEXA
5X.5X4X1X8 -0.X6X1X13 64.4 3 4
root@kali-Pi:~#
```

Copy the zip file from the Kali Linux desktop to the Raspberry Pi using SCP:

```
root@kali-Lappy:~/Desktop# scp Raspberry\ Pi\ Files.zip root@192.168.1.1:/root/
root@192.168.1.1's password:
Raspberry Pi Files.zip                          100% 7399     7.2KB/s   00:00
root@kali-Lappy:~/Desktop# ssh root@192.168.1.1
```

Install Zip, as Kali Linux does not have Zip installed by default, and then unzip the WiMap files. All the dependencies should now be installed, so you can run WiMap, as shown in figure 10:

```
root@kali-Pi:~# apt-get install unzip
root@kali-Pi:~# unzip Raspberry Pi Files.zip
root@kali-Pi:~# cd Raspberry Pi Files
root@kali-Pi:~/Raspberry Pi Files# python WiMap.py wlan0
```

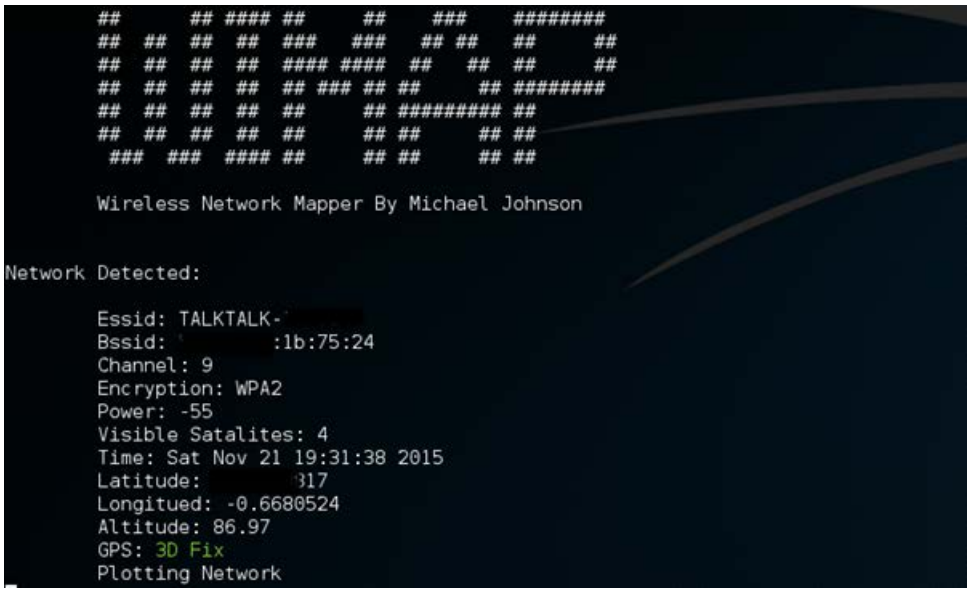

Figure 10: WiMap Running

The final Python script, SendData.py, needs to be edited with the valid MySQL user credentials. Open a cache file written from WiMap and then go on to read the identified networks; the script will then connect to the bridge's MySQL database and insert the networks for later viewing via the web interface.

```
root@kali-Pi:~/Raspberry Pi Files# cat SendData.py
import MySQLdb as mysql
import time,os

IP="BrdigeIPHere"
Username="wimapclient"
Password="passwordhere"
Database="WiMap"
[SNIP]
```

3.12 Auto Connecting Raspberry Pi to VPS Bridge

Raspberry Pi

Enable starting up on boot for the 3G dongle

```
root@kali:~# cat /etc/network/interfaces
auto lo
iface lo inet loopback
allow-hotplug eth1
iface eth1 inet dhcp
root@kali:~#
```

The following rules will stop the 3G USB dongle from mounting as a mass storage device instead of a USB modem:

```
root@kali:~# wget https://raw.githubusercontent.com/digidietze/usb-modeswitch-
data/master/40-usb_modeswitch.rules
--2015-11-22 02:27:51--  https://raw.githubusercontent.com/digidietze/usb-
modeswitch-data/master/40-usb_modeswitch.rules
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.31.18.133,
64:ff9b::b91f:1285
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.31.18.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26132 (26K) [text/plain]
Saving to: ‘40-usb_modeswitch.rules’

40-usb_modeswitch.rules
100%[======================================>]  25.52K  --.-KB/s   in 0.04s

2015-11-22 02:27:52 (571 KB/s) - ‘40-usb_modeswitch.rules’ saved [26132/26132]

root@kali:~# mv 40-usb_modeswitch.rules /lib/udev/rules.d/
root@kali:~# reboot
```

The start-up script, WiMap, needs to be edited with the IP address of the VPS bridge; this allows Mav Proxy to connect to the VPS bridge and relay commands sent from Mission Planner and other applications:

```
root@kali:~/WiMap# cat WiMap
echo "Starting MavLink Proxy"
(
date
echo $PATH
PATH=$PATH:/bin:/sbin:/usr/bin:/usr/local/bin
export PATH
cd /root/
screen -d -m -S MavLink -s /bin/bash  mavproxy.py --master=/dev/ttyAMA0 --
baudrate 57600 --out=<vpsbridgeip>:14550 --out=127.0.0.1:57600
) > /tmp/rc.log 2>&1

root@kali:~/WiMap#
```

A successful run of the script will start three simultaneous screen sessions that have been detached; these can be reattached at any time for debugging purposes:

```
root@kali:~/WiMap# ./WiMap
Starting MavLink Proxy
Starting WiMap
Starting WiMap Data Service
root@kali:~/WiMap# screen -dr
There are several suitable screens on:
       24830.SendData      (11/21/2015 09:11:24 PM)  (Detached)
       24827.WiMap  (11/21/2015 09:11:24 PM)  (Detached)
       24824.MavLink(11/21/2015 09:11:24 PM)  (Detached)
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
root@kali:~/WiMap#
```

VPS Bridge

```
root@WiMap-Server:~# sudo apt-get update
root@WiMap-Server:~# sudo apt-get install screen python-matplotlib python-opencv
python-pip python-numpy python-dev
root@WiMap-Server:~# sudo pip install pymavlink
root@WiMap-Server:~# sudo pip install mavproxy
```

MavProxy needs to listen on the bridge, as the clients may connect via NAT and the hexacopter will not be able to communicate with the remote base station:

```
root@WiMap-Server:~# screen -dms /bin/bash mavproxy.py –out=udpin:0.0.0.0:57600 -
-aircraft Drone
```

Whenever the Raspberry Pi connects to the bridge, three files will be created.

After Mav Proxy has started on the bridge, ensure the APM and Raspberry Pi are powered, with Mav Proxy running, and then establish the connection between the Raspberry Pi and the bridge. It is then possible to connect APM Planner to the Mav Proxy.

Figure 11: Mission Planner Interface

WiMap Web Interface

On the VPS bridge, an Apache webserver with PHP and MySQL is required in order to host the web interface for WiMap. These can be installed using the APT package manager in Ubuntu:

```
root@WiMap-Server:~# apt-get –y install apache php mysql
```

After the dependencies for the web interface have been installed, ensure the services have been started successfully. Copy the files from the html directory into the webroot of the Ubuntu server. After all the files have been copied across, edit the DB-Install file with passwords and usernames of your own choice, and then install the database with the MySQL command line interface:

```
root@WiMap-Server:~ /WiMap/VPS Bridge Web Interface/html/DB-Install# mysql -uroot
< Database-Install
```

After the database has been installed, ensure that the accounts have access to connect from a remote IP address, as the SendData.py script will connect to the database and insert networks that it has identified.

Once all checks have been completed, edit the “config.php” file with the username and password that was previously inserted into the database, then browse to the IP address of the bridge to be presented with a login form:

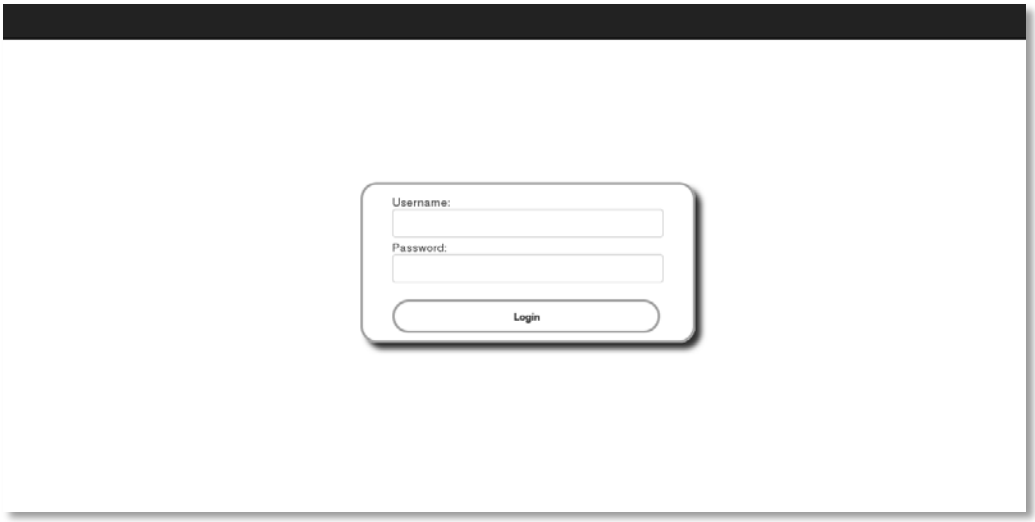

Figure 12: WiMap Web Interface

Note that the map and graph will populate once the drone has a 3D GPS fix and there are networks within range. As there is currently no data within the database, it is possible to login to the WiMap interface and manually add networks.

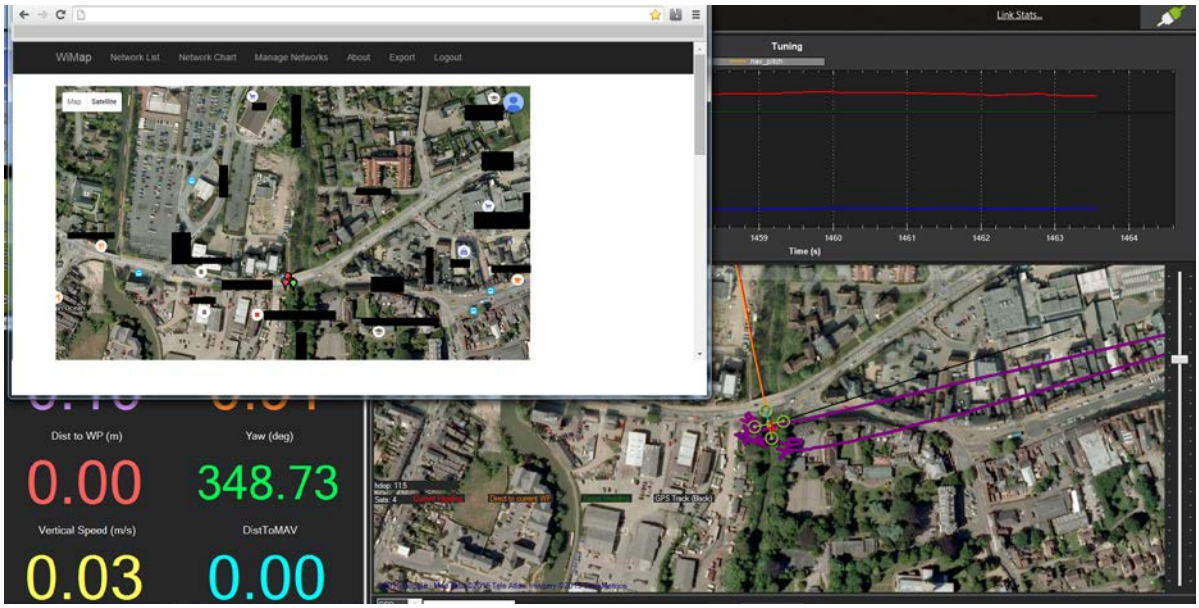

Figure 13: Example of WiMap Plotting Wireless Access Points

3.13 Remote Administration of the Raspberry Pi

For remote administration to work, some SSH keys will need to be generated. As the 3/4G network is possibly behind a firewall, the network will require a reverse connection, thus bypassing the firewall:

```
root@kali-Pi:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
2f:f0:50:6f:6b:a0:af:08:1a:af:31:d9:34:2c:32:c8 root@kali-Pi
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|                 |
|        .        |
|o.     . .       |
|=E+   o S o      |
|.* .   = + .     |
|= o   . o +      |
| * . . . o       |
|o.. . ...        |
+-----------------+
```

Use SSH-copy to copy the SSH keys to the VPS bridge and allow the use of auto login:

```
root@kali-Pi:~# ssh-copy-id -i /root/.ssh/id_rsa.pub droneclient@<vpsbriddgeip>
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
droneclient@<vpsbriddgeip>'s password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'droneclient@<vpsbriddgeip>'"
and check to make sure that only the key(s) you wanted were added.
```

Start a Cron job to execute the AutoSSH script every minute:

```
root@kali-Pi:~# crontab –e
*/1 * * * * /root/WiMap/AutoSSH > /root/WiMap/ssh.log 2>&1
root@kali-Pi:~# chmod +x /root/WiMap/AutoSSH
```

Ensure the Raspberry Pi automatically connects to the VPS bridge:

```
root@WiMap-Server:~# netstat –antop |grep 8080
tcp  0 0 127.0.0.1:8080  0.0.0.0:*  LISTEN      30770/sshd: zy0d0x off (0.00/0/0)
tcp  0 0 127.0.0.1:54589 127.0.0.1:8080  TIME_WAIT      timewait (47.66/0/0)
root@WiMap-Server:~#
```

Connect to the Raspberry Pi to ensure everything is working:

```
root@WiMap-Server:~# ssh root@127.0.0.1 -p 8080
root@127.0.0.1's password:

The programs included with the Kali GNU/Linux system are free software;
```

```
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Nov 22 02:17:23 2015 from localhost
root@kali-Pi:~# uname -a
Linux kali 4.0.9 #1 PREEMPT Thu Aug 13 12:26:22 CDT 2015 armv6l GNU/Linux
root@kali-Pi:~#
```

4 Conclusion

The hexacopter is now constructed, with a working 3/4G connection to the ground station which has the ability to connect back to the Raspberry Pi via SSH. The Raspberry Pi has Kali Linux installed, along with the custom scripts that allow it to record and transmit access point and network information back to the ground station, which will load them into the database and plot them on a map using a web server.

Additional items to be added in the future:

- VPN connection between hexacopter, VPS bridge and ground station.
- FPV (First Person View) camera system running at 720 pixels at 15-30 frames a second.
- Full-disk encryption with nuke keys, so client data is never lost in the event of an emergency landing.
- Remove the Smartstick USB portable battery and replace with a logic level converter.

5 References and Further Reading

APM or Mission Planner Wiki

http://copter.ardupilot.com/

DJI Flame Wheel ARF Kit F550 User Manual

http://dl.djicdn.com/downloads/flamewheel/en/F550_User_Manual_v2.0_en.pdf

Laws & Regulations Unmanned Aircraft | UK Civil Aviation Authority

https://www.caa.co.uk/Commercial-Industry/Aircraft/Unmanned-aircraft/Unmanned-Aircraft/

Binding the Receiver and Setting Fixed ID on the transmitter

http://image.helipal.com/helipal-f450-rtf-naza-v2-manual-01.pdf

Kali Linux Raspberry Pi Install Guide

http://docs.kali.org/kali-on-arm/install-kali-linux-arm-raspberry-pi

