The use of Hexacopter DJI S900 for [redacted]

Sven Eriksson

September 13, 2016

1 Purpose

To purchase a multi rotor drone for [redacted].
Shall also be able to carry ≈ 1kg of equipment for [redacted].

At least 10 min of flight time, but more is better.

It has been done:
`https://www.youtube.com/watch?v=i5q1iM6_37U`

2 Limitations

Flight regulations managed by Transportstyrelsen. Licence type 1B seems like the best fit for our purposes. Licence type 1B limits the weight of the drone to a maximum of 7kg and the maximum kinetic energy to 1000J.

Some vaguely defined budget constraint. $3000 seems acceptable.

Ground station shall not depend on access to a power socket.

Data link should preferably use 2.4GHz, 433MHz is preferred over 5GHz.

The drone may not be further away than 600m from the pilot.

3 List of parts

3.1 Drone

DJI S900, good flight time with a weight of 6.9kg. Estimated flight time with two 6S 13Ah batteries is 30-35 min.

3.1.1 Reseller

Not in stock: `https://mikrokoptersweden.nordicshops.com/`
In stock: DJI

3.1.2 Alternatives

Drones from Yuncee and Vulcan UAV. Both seem more expensive and are often designed with a camera. Vulcan UAV have a few drones that are for lifting equipment but those are heavier. They are designing a model that would fit our requirements but that hasn't reached the market yet.

DYS D800 V-6 Seems like another good option. Cheaper but difficult to find.

Design our own drone by choosing a frame, ESC, motors and other parts. People have been lifting up to a kg with the frame F550 from DJI. This is a cheaper option should it work but is also more risky. An option is to buy this for tests with lighter loads.

3.2 Autopilot

Pixhawk, large community and we already have some experiance working with it.

Also power unit:
`https://www.sparkfun.com/products/10644`
`http://www.uav-store.de/sensoren/attopilot/#cc-m-product-9803791793`

GPS / Compass if not included:
`http://www.uav-store.de/gps/ublox/#cc-m-product-8282839193`

Dampening system:
`http://www.hobbyking.com/hobbyking/store/__76727__APM_Flight_Controller_Damping_Platform.html`

External LED and usb:

Everything we need, company based in china:
`https://www.foxtechfpv.com/pixhawk-433mhz-combo-p-1896.html`

3.2.1 Reseller

In stock:
3d Robotics

Maybe:
http://www.uav-store.de/autopilot/pixhawk/

Not in stock:
http://www.rctech.se/
http://www.autopartner.se/ http://www.robotshop.com/eu/en/3dr-px4-pixhawk-advance
html

3.2.2 Alternatives

All autopilots from DJI. The drone is designed together with these autopilots but it does not provide the interface that we are used to. Probably harder to write custom code for.

3.3 Battery

To be decided. 6S lipo batteries with good capacity to weight ratio as well as enough capacity and C rating. At least 10Ah per battery.

Alternatives from Swedish resellers 16Ah (13Ah can also be used):
http://www.pitchup.se/uav-power-cell-6s-16000mah-lipo-batteri.

Lipo tester / voltage checker.
http://www.autopartner.se/elektronik-1/batterier/tillbehor/gt-power-lipotestare-

3.4 Battery Eliminator Circuit

To power the equipment running on 5V from the 6S battery. Get 2-3, use in parallel for redundancy.

Castle Creation CC BEC 10A-PEAK 5-25V 2-6S LiPo
5.1V at delivery. Programmable so check voltage, buy device required to reprogram? Castle Creation CASTLE LINK USB Programming Kit

3.4.1 Reseller

In stock: http://www.autopartner.se/elektronik-1

3.5 Cables and connectors

Extension Servo cables. For ESC and servos.

DF13 GPS extension cable. Pixhawk GPS and reach placement.

XT-90 power connectors.

Power cables.

3.5.1 Reseller

In stock: http://www.autopartner.se/elektronik-1
HobbyKing

3.6 Receiver - Transmitter

FrSky X8R Receiver together with existing transmitter.

Telemetry cable from:
http://www.craftandtheoryllc.com/

See http://www.dronetrest.com/t/pixhawk-telemetry-via-frsky-x8r-receiver/854/2 For telemetry cable in use.

Range shouldn't be a problem. Just buy transmitter and receiver from the same manufacturer. Look at telemetry data link from drone to receiver.

Just make sure that all 2.4GHz equipment uses frequency hopping technology.

3.6.1 Reseller

In stock:
http://www.rctech.se/index.php?route=product/product&path=85_101&product_id=226

3.6.2 Alternatives

FrSky X8R Receiver, more channels. But 6 should be enough. And due to the fact that SBUS will be used in communication with pixhawk. X6R might already provide 16 channels.

3.7 Data link

For all of these product the antenna can be changed in order to provide more range in certain directions.

For 433MHz:
http://rctimer.com/product-834.html

Ubiquity data link used in competition: http://www.auvsi-suas.org/static/competitions/2015/journals/auvsi_suas-2015-journals-universite_de_sherbrooke_vamudes.pdf

Picostation (100g, short range), Bullet M2 for ground. (Picostation can be replaced with Bullet, but it is heavier $\approx$ 180g.) Knowledge regarding antennas would be useful:

https://www.senetic.se/ubiquiti/airmax/bullet_m/

https://www.dustinhome.se/product/5010471164/picostationm2hp

Just make sure that all 2.4GHz equipment uses frequency hopping technology.

Both can be used in parallel. Only adds weight. I guess it can be considered extra safety. In this case two laptops should be used.

3.7.1 Alternatives

433MHz Serial radio with UART interface.

2.4GHz or 5GHz ubiquiti bullets, ethernet cables connectors. `https://dl.ubnt.com/datasheets/bulletm/bm_ds_web.pdf`

Things to consider:
Frequency
Serial vs. TCP / IP.
Range
Weight
Power requirements

The option that is chosen need to be tested for throughput as well as range.

4 Weight estimation

(?) means that it is an estimate made by Sven due to lack of information.

Have not accounted for cables. Some are additional cables are needed but those should be light.

Type	Item	Number	Individual Weight	Weight
Drone				
	DJI S900	1	3.3kg	3.3kg
Autopilot				163g
	Pixhawk (without GPS)	1	38g	38g
	Pixhawk GPS & Compass	1	17g	17g
	Pixhawk Dampening	1	16g	16g
	Raspberry Pi	1	45g	45g
	X6R Receiver	1	13g	13g
	X6R Telemetry cable	1		(?)
	RTK (reach without antenna)	1	14g	14g
	RTK (reach antenna)	1	20g (?)	20g (?)
Power (alt 1)				2.103kg
	Castle Creation CC BEC	3	11g	33g
	Power Unit	1	< 20g (?)	< 20g (?)
	16Ah Battery	1	2.05kg	2.05kg
Power (alt 2)				1.593kg
	Castle Creation CC BEC	3	11g	33g
	Power Unit	1	< 20g (?)	< 20g (?)
	13Ah Battery	1	1.54kg	1.54kg
Data Link				<0.4kg (?)
Payload				1 kg
Total (alt 1)				6.866kg
Total (alt 2)				6.356kg

5 Cost estimation

(?) means that it is an estimate made by Sven due to lack of information. Have not accounted for transport. $1 = 8.21 SEK, 1 euro = 9.27 SEK.

Type	Item	Cost	Cost SEK
Drone			
	DJI S900	$1,400	11,494 SEK
Autopilot			
	Pixhawk (without GPS)	215 euro	1,993 SEK
	Pixhawk GPS & Compass	85 euro	788 SEK
	Pixhawk Dampening	3 euro	28 SEK
	Raspberry Pi	449 SEK	449 SEK
	X6R Receiver	411 SEK	411 SEK
	X6R Telemetry cable	$ 29	238 SEK
	RTK (reach without antenna)	$235	1,929 SEK
	RTK (reach antenna)	$60	493 SEK
Power			
	Castle Creation CC BEC	257 SEK	771 SEK
	Castle Creation Link	258 SEK	258 SEK
	Lipo Tester	89SEK	89 SEK
	Chargers	?	?
	Attopilot Power Unit	30 euro	280 SEK
	16Ah Battery	2,990 SEK	
	13Ah Battery	2,799 SEK	
Cable and Connectors			(?)
Data Link			
	433MHz	$25	206 SEK
	Picostation	1,045 SEK	1,045 SEK
	Bullet M2	1,031 SEK	1,031 SEK
Payload			
Total	3 * 16Ah Batteries		≈ 30,500 SEK

Possible to buy at Dustin

Picostation:
https://www.dustinhome.se/product/5010471164/picostationm2hp

Bullet M2:
https://www.dustinhome.se/product/5010654476/airmax-bullet-m2-titanium

Raspberry Pi 3:
https://www.dustinhome.se/product/5010909893/3-model-b-12ghz-64-bit-arm-1gb-ram-

Micro SD card:
https://www.dustinhome.se/product/5010810591/ultra

Additional things we bought

Extension cables for servo cables.

Extension cable for balancing connector on battery (for charging).

Anti-spark connector XT90-S. Also normal XT90 instead of AS150. Based on recommendation. The XT90 are more commonly used and even though we might be slightly above rated current there doesn't seem to be a heat issue.

Ammunition box for battery storage.

Battery charger from pitchup. $\approx$ 5000SEK.

Additional cable (12AWG), tape and adhesive pads.

6 Rules and regulations

Nice pdf / presentation for the different rules:
`http://skt.se/wp-content/uploads/2015/11/Dr%C3%B6nare-och-lagstiftningen.pdf`

6.1 Fligth restrictions

Do only fly within sight.

No more than 500m horizontally away from pilot and no higher than 120m.

Do not fly in restricted airspace. Map:
`http://daim.lfv.se/echarts/dronechart/`
[redacted] lie within restricted airspace.

Do not fly in such a way so that it poses a danger to other people or others property.

6.2 Licence and insurance

In order to get a licence from Transportstyrelsen one needs an insurance that specifies that it covers accidents with drones up to ≈9,000,000 SEK. Inter Hannover seems to be a popular choice according to certain forums.

6.2.1 Text for application

Following the numbering in this example of application:
`http://www.transportstyrelsen.se/globalassets/global/blanketter/luftfart/flygtrafik/anvisningar-for-att-fylla-i-ansokan---kat-1b.pdf`

1. [redacted]
2. [redacted]
3. [redacted]

4. [redacted]
5. Hexacopter, multirotor
6. DJI Spreading Wings S900 med Pixhawk som styrkort
7. El, 6 motorer
8. Pixhawk med funktionalitet för automatisk flygning. Samma utrustning för manuell flygning.
9. 0,38 m rotordiameter.
10. 0,9m i längd och bredd (nästan cirkulär). 0,5m hög med nedfällda landningsställ.
11. 6,9kg
12. 16m/s
13. 883,2J
14. Inbyggt i styrkortet Pixhawk som kör mjukvaran APM. Vid bortfall av kommunikationslänk sätts drönaren automatiskt i ett läge som automatiskt flyger tillbaka till platsen för uppstart och landar. Samma funktionalitet kan aktiveras av piloten via en knapp.
15. Överstiger ej 1000J
16. Bilder, se manual sida 1-2? (`http://dl.djicdn.com/downloads/s900/en/S900_User_Manual_en_v1.4.pdf`)
17. Behöver tillståndsnummer
18. Förstått gällande avgifter.

7 Implementation and documentation

7.1 Manual receiver and transmitter

Receiver X8R. Manual is downloaded, can otherwise be found at `http://frsky-rc.com/download/view.php?sort=Manual&down=102&file=Manual-X8R`.

Transmitter FS-TH9X with 3DR software containing a DJT transmitter module from FrSKY.

Bind X8R according to manual to mode 1. Bind plug between channel 7 and 8. Video:
`https://www.youtube.com/watch?v=BjFTEFzLtz0`

Failsafe set to option 2, no-pulse fail-safe. Press F/S on the receiver briefly when the receiver is on but the transmitter is off. Check manual. Verify fail-safe and transmitter functionality through Mission Planner.

8 channels can be used in total:

Channel #	Pixhawk Use	Transmitter switch
1	Roll	Right stick horizontal
2	Pitch	Right stick vertical
3	Thrust	Left stick vertical
4	Yaw	Left stick horizontal
5	Flight Mode	RTL and Mode Switch
6	–	Gear (on top right)
7	Landing gear	CH7
8	–	RUD. D/R (on top left)

Which button / switch that corresponds to each channel is set in the transmitter. The correct menu is called mixer under the HEX profile.

What each channel corresponds to (not 1-4, they are standard) is configured in the pixhawk using Mission Planner.

Channel 7 should be set to on at start (landing gears are down / activated).

To see telemetry data. Go to the normal view to control the model and press / hold the button down. You will now see the telemtry data. Interesting information in this menu is battery voltage and mode.

7.2 Power Schematic

The BEC from Castle Creation can be reprogrammed to provide any voltage between 5-9 (also some value outside of this range). This could be useful for the external sensors used in future courses.

The power that goes through the pixhawk mostly goes through its power

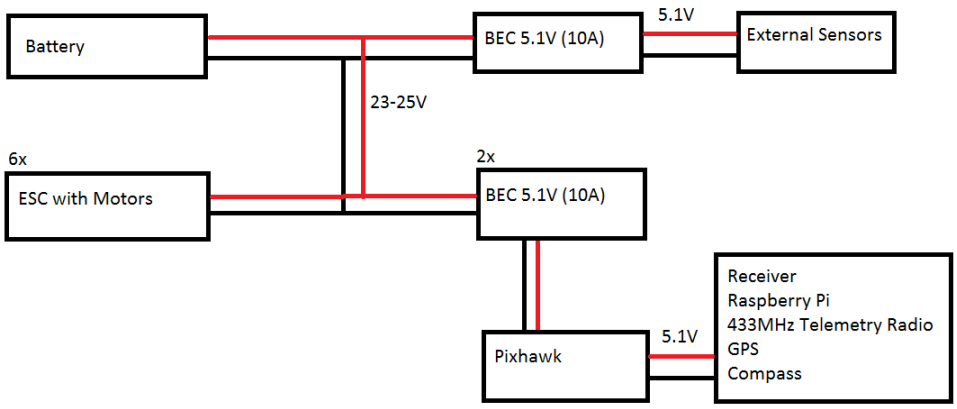

Figure 1: A simple picture of the power system, some components are not here.

rail. There are 2 BECs powering the flight electronics so that a BEC failure doesn't result in a crash. Do occasionally test the BECs using a multimeter and the 4S battery from the IRIS+.

Batteries was bought from Pixup, 6S battery with 16000mAh. Cable size is 10 AWG (so that should be enough for all uses). XT90 and XT90-S is used to connect the battery. XT60 is used to connect the BECs the power distribution.

7.3 Hardware

7.3.1 DJI 900

Follow the manual for S900. Note that there is a different numbering of motors between the hexacopter and what the arducopter (pixhawk) expects. Incorrectly wired motors will result in problems, possible crash.

7.3.2 BEC from Castle Creation

Programmable through the Castle Creation Programming Cable. USB cable to 3 pins through a rather small chip. All three are currently set to output 5.1V.

7.4 Pixhawk configuration

7.4.1 Telemtry to transmitter

`http://ardupilot.org/copter/docs/common-frsky-telemetry.html`

Connect the "Pixhawk to FrSky Telemetry Cable (X6R, X8R, FLVSS)" cable from Craft and Theory to telemetry port 2 on the pixhawk. Connect the other end to the smart port on the receiver (port next to the external antennas).

Change the parameter SERIAL2_PROTOCOL to 4.

7.4.2 Landing gear

`http://ardupilot.org/copter/docs/landing-gear.html`

When using AUX1 as output for the landing gear. Set parameter RC9_function to 29 (landing gear).

Under extended tuning, set channel 7 to landing gear.

Channel 7 should be set to on at start (landing gears are down / activated).

7.4.3 Failsafe

Currently only RTL on loss of manual radio. Failsafe for batteries remains to be configured.

7.4.4 Voltage meter

In order to measure voltage we use a "AttoPilot Voltage and Current Sense Breakout - 180A"
https://www.sparkfun.com/products/10644
http://cdn.sparkfun.com/datasheets/Sensors/Current/DC%20Voltage%20and%20Current%20Sense%20PCB%20with%20Analog%20Output.pdf

We have verified that it outputs approximately the specified value of 63.69mV / Volt.

It is connected according to the ardupilot's manual:
http://ardupilot.org/copter/docs/common-using-a-current-sensor.html

In our case we only use a the voltage meter capability. The blue wire goes from the hole marked V on Attopilot. Black from ground.

The software is also calibrated to this unit so that it shows the correct value. Follow instructions at:
http://ardupilot.org/copter/docs/common-power-module-configuration-in-mission-planner.html#common-power-module-configuration-in-mission-planner

As we only use the sensor to measure voltage set "Monitor" in the guides above to 3. "Sensor" to other and then "APM ver" to Pixhawk.

7.5 Raspberry Pi configuration

The Raspberry Pi is running the standard Raspian operating system. It is based on debian and adapted to work well with the Raspberry Pi.

7.5.1 Mavlink

The pixhawk communicates with the ground station through the use of mavlink messages. These are sent through the 433MHz telemetry radio and through its usb port to the raspberry Pi where they are forwareded to the ground station.

A list of common messages. Not all messages are used by the pixhawk / APM copter.

`https://pixhawk.ethz.ch/mavlink/`

The messages the are sent from the autopilot (software: arducopter) when it is unarmed are:

- SYSTEM_TIME
- TERRAIN_REPORT
- EKF_STATUS_REPORT
- VIBRATION
- ATTITUDE
- AHRS2
- AHRS3
- VFR_HUD
- SYSTEM_STATUS
- POWER_STATUS
- MEMINFO
- MISSION_CURRENT
- GPS_RAW_INT
- NAV_CONTROLLER_OUTPUT
- RAW_IMU
- SCALED_IMU2
- SCAÖED_PRESSURE
- GLOBAL_POSITION_INT
- LOCAL_POSITION_NED
- SERVO_OUTPUT_RAW
- RC_CHANNELS_RAW

- AHRS
- HWSTATUS
- RANGEFINDER

7.5.2 MAVProxy

The messages sent through the usb port to the raspberry pi are handled by a software called MAVProxy. MAVProxy is in itself a ground station software but is also capable of forwarding messages between the pixhawk and the laptop that is running mission planner.

A link to mavproxy documentation:
`http://dronecode.github.io/MAVProxy/html/index.html`

In addition to using it to forward messages we are also using it to expose some pixhawk parameters to other software running on the raspberry pi. This is done by writing a MAVProxy module that runs a TCP server that other programs may connect to.

I do usually use this command to start MAVProxy

```
mavproxy.py –aircraft test –out [IP for laptop with mission planner]
–load-modules time_server, [other modules you want to run]
```

As the SSH connection that started mavproxy might be lost it is useful to start in a screen session.

```
screen -d -m mavproxy.py –aircraft test –out [IP for laptop with
mission planner] –load-modules time_server, [other modules you want
to run]
```

If you did not load all modules you wanted at start-up you can use this command to load them while MAVProxy is running:

```
module load [name of module]
```

For more information on how to use MAVProxy, see the MAVProxy documentation.

7.5.3 Custom MAVProxy modules

In order to make custom modules follow these instructions:
`http://dronecode.github.io/MAVProxy/html/development/index.html`

I do often install mavproxy for all users rather than the user pi. Use this command to install for all users.

```
cd ~/MAVProxy
sudo python setup.py build install
```

There are currently a few custom modules. If you want to develop others with similar functionality feel free to use any of them as a reference. The modules class name needs to be changed in a few places in order for the new module to work.

message_list Prints out the name of the message as it arrives. Useful for checking that messages are being sent by the autopilot.

message_server Exposes all the contents of all mavlink messages being sent from the autopilot to the raspberry pi. One needs to specify what messages one is interested in.

time_server Exposes the internal pixhawk clock to other software. The code for this module can be found in A.2.1. This module waits for the mavlink message SYSTEM_TIME and reads one of its values.

orientation_server Gives values for the orientation of the drone.

position_server Gives values for the position of the drone.

The source code for these modules (as well as other custom modules that you write) are located in:

```
~/MAVProxy/MAVProxy/modules
```

7.5.4 TCP clients for parameters, General

Each mavlink message has a function to_json() that is used to make one server instead of several smaller ones. One asks for the mavlink messages one wants

and get those as well as the time since they where received. Something like JSON is a good format for this.

The message sent to the server is then also important. As it specifies the list of mavlink messages that are of interest.

"Question" (example):

```
SYSTEM_TIME,POWER_STATUS,DUMMY
```

Incorrect questions will be answered, see the answer for DUMMY. No spaces between commas and mavlink message names.

"Answer" (example):


```
{
'POWER_STATUS':
{'content': '{"mavpackettype": "POWER_STATUS", "Vcc": 4529, "flags": 4,
"Vservo": 0}', 'time_diff': 178},
'SYSTEM_TIME':
{'content': '{"mavpackettype": "SYSTEM_TIME", "time_boot_ms": 8927223,
"time_unix_usec": 1468581578083000}', 'time_diff': 155}
'DUMMY': {}
}
```


Notice the additional ' around the data for content.

This is currently configured to listen to port 10000. It should work for all previously listed mavlink messages. For the specific meaning of each value, see the mavlink message documentation.

7.6 Network

Ip-address of the raspberry pi: x.x.x.x

Ip-address of the laptop: y.y.y.y

The Raspberry pi needs a fixed address so that it can be accessed through SSH. It is possible to automate a bit more of the scripts on the raspberry pi

if the ip-address of the laptop running missionplanner is known. Even easier if it is fixed.

7.6.1 Ubiquiti configuration

Followed this guide to get the two units in bridge mode:
`https://help.ubnt.com/hc/en-us/articles/205142890-airMAX-Configure-a-Point-to-Po`

They are currently configured to 192.168.1.2 and 192.168.1.3, this means that the ip-range for the laptop and other equipment is 192.168.1.z

Picostation: Bullet M2:

8 To do list

8.1 Data link

Design and make attachment for Picostation.

Power for picostation and bullet. Take two ethernet cables. One is done, test that one first? Same cable can be used to test both units. Test 1: Connected the battery to the bullet through the cable. Spark sound and destroyed bullet. Test 2: Connection through the anti-spark adapter, smoking cable. Working picostation. Potential solution, anti-spark adapter and new bullet ($\approx$ 1,000 SEK).

Working alternative: Use same adapter as in thesis work to host a wifi network. Short range but enough to start up scripts on the raspberry pi (sensor logging etc.) (Reach RTK will not work in real time if that is planned).

8.2 Network Configuration

Network configurations. Set fixed ip for raspberry pi. Either get external network card for the laptop or reconfigure it each time. Easy to reconfigure.

8.3 Mavproxy Script

Script to start mavproxy for the raspberry pi. (Done, except for fixed ip) Is it screen?

8.4 Attach Raspberry Pi

Get a box for it and stick it to the hexacopter. Use usb cable to connect and the GPIO pins to power the pi.

8.5 Test Flights

8.5.1 Already done

Initial test flight: 2016-07-15
It seemed to handle well. Mostly hovering in the same area. Flew around 60 seconds and had to charge the battery for 1000mAh. Note that this battery had been used inside to power the electronics during some preparations.

Second test flight: 2016-07-20
It seemed to handle well. Mostly hovering in the same area. Flew $\geq$4.5 minutes and had to charge the battery for 2566mAh.

First test flight after reinstallation of arducopter on pixhawk: 2016-08-15
Flew $\approx$ 10 minutes. Pixhawk reported a voltage of 22.5V under load at the end. The charger claimed that the battery still had 54% of total capacity left. Charged it with 6284 mAh (39 % of total capacity used, so almost half to the 80% recommendation). Tested take-off and landing using altitude hold, worked well. A little bit scary during take-off as it won't throttle up until you reach above 50% throttle (less tells the UAS to descend or hold altitude). While testing loiter the UAS remained in approximately the same position.

First test flight after reinstallation of arducopter on pixhawk: 2016-08-15
Mostly hovered $\approx$ 10 minutes. Pixhawk reported a voltage of 22.5V under load at the end. The charger claimed that the battery still had 54% of total capacity left. Charged it with 6284 mAh (39 % of total capacity used, so

almost half to the 80% recommendation). Tested take-off and landing using altitude hold, worked well. A little bit scary during take-off as it won't throttle up until you reach above 50% throttle (less tells the UAS to descend or hold altitude). While testing loiter the UAS remained in approximately the same position.

Test flight of an automated mission : 2016-08-15
Flew in manual and automated mores for $\approx$ 13 minutes. Pixhawk reported a voltage of 22.1V under load at the end. The charger claimed that the battery still had 37% of total capacity left. Charged it with 8755 mAh (55 % of total capacity used. Almost 4000mAh left until the 80% recommendation). Tested automated flight following a set mission. Repeated the mission 3 times as is took less than 3 minutes to do.

Test flight of an automated mission: 2016-08-15
Flew in automated mode in a big circle. Flew in manual and automated mores for 16 minutes and 45 seconds. Pixhawk reported a voltage of 22.0 V under load at the end. Charged it with 9512 mAh (59 % of total capacity used.

8.5.2 Skipped

Run short mission with automated landing.

Test failsafe for loss of manual transmitter (Should result in auto-land that isn't tested).

9 Reasoning why some things work

9.1 GPS Time

Purpose: Provide the current GPS time to a program running on the raspberry Pi.

Sources: Pixhawk / Reach / Raspberry Pi GPS hat?

Raspberry Pi hat:
`http://www.modmypi.com/blog/raspberry-pi-gps-hat-and-python`

This solution does provide GPS timestamps. Unsure if there is any latency or off-set.

Pixhawk option:
SYSTEM_TIME is one of the many messages being sent over the MAVLINK protocol. I checked by using MAVProxy to log all incoming messages. `https://pixhawk.ethz.ch/mavlink/#SYSTEM_TIME`.

It contains a value called time_unix_usec that is described as "Timestamp of the master clock in microseconds since UNIX epoch.".

According to a answer on StackExchange, `http://stackoverflow.com/questions/34313892/how-can-i-get-the-gps-time-from-the-pixhawk-on-a-companion-computer`, this internal clock is synced with gps time. This answer is supported by a discussion on github regarding a new feature `https://github.com/ArduPilot/ardupilot/issues/2289#issuecomment-101644815`.

The message SYSTEM_TIME is being sent around 4 times a second. How do one verify that this indeed is synced with the GPS time?

There is now a working TCP server that provides the time_unix_usec from the last message as well as the number of milliseconds since the last message was received by mavproxy. I do still need to figure out a way to use this when (/if) the ip address of the pi changes as it is currently part of the code.

Mavproxy module: mavproxy_time_server.py

It is currently started manually and contains a lot of debug printouts. Should be included in a start script for mavproxy and debug printouts should be removed.

9.2 Ubiquiti power

Ethernet cable pin assignment:
`https://www.trangosys.com/cat-5-ethernet-cable-standards-pin-out-assignments/`

Picostation datasheet:
`https://dl.ubnt.com/datasheets/picostationm/picom2hp_DS.pdf`
Power through pair 4,5 and ground through pair 7,8. Test with voltmeter?

Bullet M2 datasheet:
Same pins for power.

A Code

A.1 TCP client to get parameter values

Python 2 example code:
https://docs.python.org/2/library/socketserver.html#socketserver-tcpserver-examp

```
import socket
import sys

import time

#IP address for the server, localhost if running on the same computer
#The port for the parameter you want to access. Check documentation.
HOST, PORT = "129.16.36.224", 10000


# Create a socket (SOCK_STREAM means a TCP socket)
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    sock.connect((HOST, PORT))

    # Connect to server and send data
    sock.sendall("SYSTEM_TIME,POWER_STATUS,DUMMY")

    # Receive data from the server and shut down
    received = sock.recv(16384)
finally:
    sock.close()

print received
```

A.2 Mavproxy modules

A.2.1 message_server

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

import time, math
import datetime
import SocketServer
import threading


from pymavlink import mavutil
from MAVProxy.modules.lib import mp_module
from MAVProxy.modules.lib.mp_settings import MPSetting

class TimeModule(mp_module.MPModule):
    def __init__(self, mpstate, server):
        super(TimeModule, self).__init__(mpstate, "timeServer", "timeServer")
self.server = server

    def mavlink_packet(self, m):
        '''handle an incoming mavlink packet'''
        self.server.newMessage(m)

server = ""

class MyTCPServer():
    def __init__(self):
        self.allMessages = {}
        HOST, PORT = "0.0.0.0", 10000
        s = ThreadedTCPServer((HOST, PORT), ThreadedTCPHandler)
        s_thread = threading.Thread(target=s.serve_forever)
        s_thread.daemon = True
        s_thread.start()

    def newMessage(self, message):
        key = str(message.get_type())
        self.allMessages[key] = {}
        self.allMessages[key]["content"] = message.to_json()
```

```
        self.allMessages[key]["time"] = datetime.datetime.now()


class ThreadedTCPHandler(SocketServer.BaseRequestHandler):
    def handle(self):
        currentTime = datetime.datetime.now()
        receivedData = str(self.request.recv(1024))
        if receivedData == "":
            receivedData = " "
        keys = []

        while 1==1:
            pos = receivedData.find(",")
            if pos != -1:
                keys.append(receivedData[:pos])
                receivedData = receivedData[pos+1:]
            else:
                keys.append(receivedData)
                break
        response = {}

        for key in keys:
            response[key] = {}

            try:
                timeDiff = datetime.datetime.now() - server.allMessages[key]["ti
                timeDiffMiliseconds = timeDiff.seconds * 1000 + timeDiff.microse

                response[key]["time_diff"] = timeDiffMiliseconds
                response[key]["content"] = server.allMessages[key]["content"]
            except:
                response[key] = {}

        self.request.sendall(str(response))

class ThreadedTCPServer(SocketServer.ThreadingMixIn, SocketServer.TCPServer):
        pass

def init(mpstate):
    global server
    server = MyTCPServer()
```

```
    return TimeModule(mpstate, server)
```