Autonomous Navigation of Unmanned Aerial Vehicles (October 2014)

O. Targhagh

Abstract — **Recently, there has been a lot of interest in the application of autonomous flight with small unmanned aerial vehicles (UAV's). Research was taken with the Robotics and Automation Labratory at the University of Western Australiain this area with a Hexacopter UAV to see if such a platform could be developed for various Search and Rescue applications, building upon work started last year.**

Using pre-existing components, a system was developed capable of autonomously mapping an outdoor area and returning information to the user about any interesting features. This thesis focused on the navigational capabilities of the Hexacopter system and how well it could be made to move between locations.

The performance of the platform was sufficiently upgraded, allowing it to be operated in a much more precise, controlled manner which would have been needed for our intended applications.

I. Introduction

A. *Background*

Aerial vehicles present an exciting and interesting area for research. There has recently been a lot of growth in this area, especially with regards to unmanned flight. The University of Western Australia Robotics and Automation Laboratory began a project last year investigating the capabilities of a small, autonomous platform that could be programmed to do a variety of tasks. The project was run by Professor Thomas Bräunl and Chris Croft along with the final year students Chris Venables and Rory O'Connor.

Together, they managed to develop a Hexacopter UAV (Unmanned Aerial Vehicle) platform capable of position tracking, along with on-board image processing [1][2]. The goal this year was to further develop that platform into a more robust system that could then be sent different objectives in flight, check for objects of interest and be controlled via a web interface.

My particular area of focus was on the Autonomous Navigation, concentrating on ways to optimise the motion of the Hexacopter. It wanted to be seen if it were possible to improve upon the navigation methods developed last year and have the Hexacopter perform some sort of ordered search, rather than simply fly to random locations.

This paper was submitted to the Mechanical Engineering Office and the University printing company UniPrint at the University of Western Australia on the 26th of October 2014 as my MCTX4422 Final Year Project Engineering Thesis, both times in an electronic format and typed according to the IEEE 2013 Standard.

This Thesis was supervised by Professor Dr. Thomas Bräunl and Chris Croft and done in a group with the students Michael Baxter, Alexander Mazur and Merrick Cloete.

B. *Hexacopters*

There are several different models of small, remote-controlled UAV's available commercially. The model being currently used is a Hexacopter, a small helicopter-like vehicle with six sets of rotor vertical blades, as shown below. A Hexacopter was determined to be the most suitable design last year as it was capable of vertically taking off, but also had redundant components in case of failure [1].

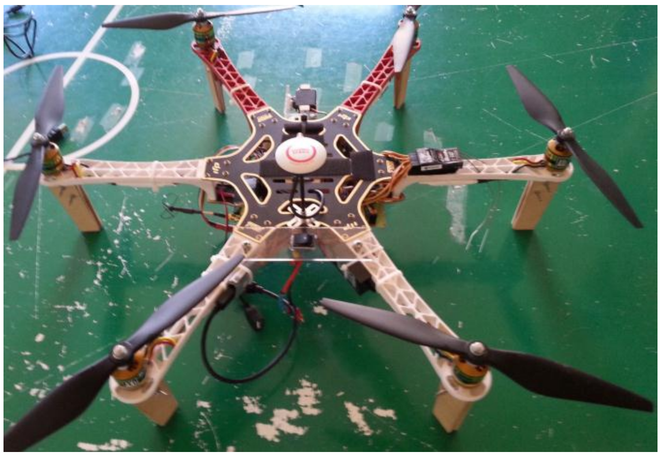


Fig. 1 - DJI F550 Hexacopter

Rather than create their own platform from scratch, the team last year selected a pre-existing Hexacopter platform to make their own changes to. The DJI Flamewheel F550 was chosen as the most capable design as it came with a pre-existing flight controller that offered better control than any corresponding open source models [1]. The F550 has its rotors arranged in a Hexa-V formation, as is shown below in **Error! Reference source not found.**, where the six blades are spaced equidistant around the outside of the Hexacopter, each spinning in the opposite direction to those adjacent to prevent the Hexacopter from rotating. Two of the arms form a 'V' at the front of the craft and are coloured red to help the pilot identify which way the craft is facing while it is in the air. Further information about the F550 can be found in Appendix A.

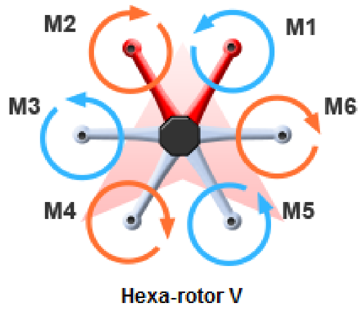


Fig. 2 - Layout of Hexacopter Blades

C. *Navigation*

In order for any robotic system to operate autonomously, it must have some sort of feedback about its position and orientation. Since a Hexacopter is capable of moving in three dimensions, this would indicate that a total of six different coordinates would be needed to record position and orientation, which can be seen below in Fig. 3. However our problem can be simplified by taking into account the mechanical restrictions the Hexacopter structure imposes on the system.

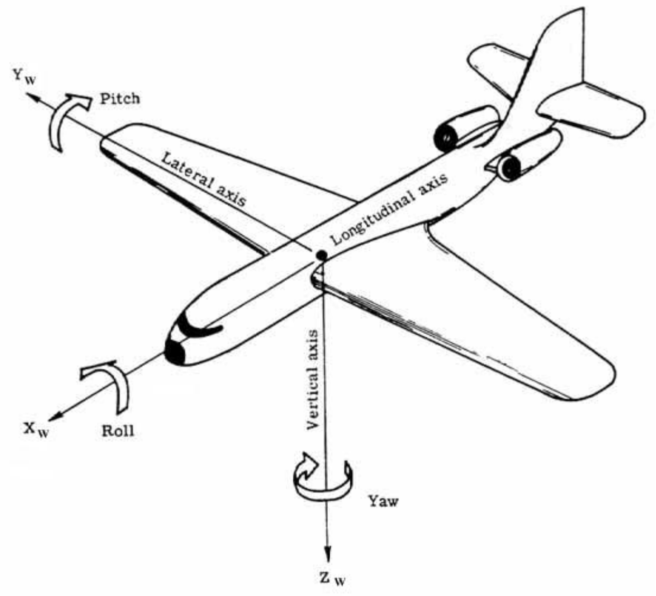


Fig. 3 - General Body Coordinates for an Aircraft [3]

Due to the lift generated by each of the blades, the Hexacopter will self-correct if it undergoes any small perturbations in its pitch and roll, provided it does not get perturbed so far that it flips over. Also, for reasons that I will explain later, the height of the Hexacopter was made to always be controlled manually. This reduces the three dimensional problem indicated earlier to a much simpler two dimensional one, requiring only three coordinates – two for position (x and y) and one for orientation (yaw) – that we needed to consider.

Although several methods for a UAV to determine its position with respect to some local coordinate system exist, these would require a pre-built environment for the UAV to operate in. As we wished develop a system capable of operating in many different outdoor environments, it was decided that we would need to use some form of universal method to determine the UAV's position. The most straightforward method to implement was with a GPS, or Global Positioning System, as these are frequently used a variety of outdoor setting, not just robotics.

D. *Search Patterns*

Once the UAV knows its position, it can then be given flight objectives in the form of position coordinates and then fly to them in turn. However having the user enter all these manually could be time consuming and required the user to be experienced enough to know what the best search pattern would be. In order to remove the emphasis on operator knowledge and to make it easier for unexperienced users to access the system via the web interface, it wanted to be seen if the Hexacopter could be made to follow a flight path that it generated itself, given very little user input, such as a start and finish location.

In Search and Rescue, one of the most common patterns used to scan over an area by air is the Creeping Line pattern [5]. As shown below, this pattern involves making alternating parallel sweeps of a target area, allowing the whole area to be covered in a relatively short time.

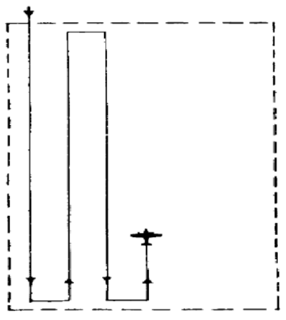


Fig. 4 – Creeping Line Search Pattern [5]

This pattern is especially useful when the objects that you are looking for can be anywhere within the area. For Search and Rescue operations this is ideal as we would have no pre-existing information about the location of any people or objects we would be looking for. Several other resources examined confirmed that this would be the best case for our purposes, as well as listing other patterns used for aerial searches. These additional patterns are mentioned in Appendix B.

E. *Literature Review*

Before we started our project, it was important for us to get an idea of the current state of the field. While the most helpful resources were the papers of the students who worked on robotics projects at this university last year, there were still a lot of other additional resources available in areas such as GPS tracking and aerial navigation. A full summary of these resources that I used in this project can be found in Appendix C.

II. Equipment

A. *Manual Flight*

Particular care was taken to learn how to control the Hexacopter manually so that all the relevant capabilities could then be replicated autonomously. Knowing how to pilot the craft was also important in case of an emergency, for example if an error occurred with our software we could confidently take over and bring the Hexacopter back under control.

Manually, the Hexacopter is controlled by the user sending it signals via a hand-held controller, which is connected to a receiver mounted on board the Hexacopter. The receiver then sends these signals to the Flightboard, a proprietary controller mounted in the centre of the Hexacopter that analyses the user commands and converts them into the necessary commands for each motor. This process can be seen in the figure below, where the four coordinate channels used to determine the pose of the Hexacopter are passed to the Flightboard by the Receiver and then converted to six motor commands.

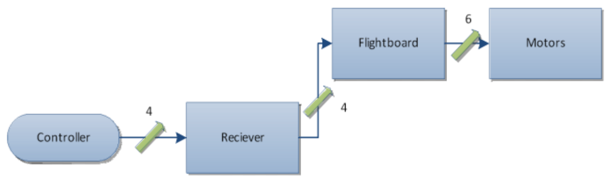


Fig. 5 - Manual Control Diagram

1) *Controller*

The controller used was the Futuba 14SG, which allowed us to use up to 8 different channels to control the Hexacopter [6]. The table below outlines the function of each of the 8 channels as used in our project. The first four are the coordinate channels, they allow us to change the position and yaw of the Hexacopter. The fifth and sixth switch the Hexacopter between a variety of control modes, the seventh was unused and the eighth controlled the camera pan, which was used for the Image Processing. More information about the controller can be found in Appendix D.

TABLE I
HEXACOPTER CHANNELS

Number	Name	Function
1	Aileron	Strafes Hexacopter Left/Right
2	Elevator	Strafes Hexacopter Forwards/Backwards
3	Throttle	Moves Hexacopter Vertically Up and Down
4	Rudder	Rotates Hexacopter on the Spot
5	Mode Control	Shifts the Hexacopter between its own Internal Modes
6	Command Control	Shifts Control of the Hexacopter between Manual and Automatic
7	Gimble Switch 2	Unused
8	Gimble Switch 1	Camera Pan

2) *DJI Flamewheel F550 Platform*

As mentioned earlier, the Hexacopter platform used in this project was a Flamewheel F550, developed by DJI industries [7]. This platform is has 6 motors, each mounted on an arm connected to two central plates. In between the plates, at the centre of the Hexacopter is mounted the Flightboard, the device responsible for converting the flight commands from the controller into motor commands.

This Flightboard provided several additional features which were quite useful for our project, such as inbuilt flight stabilisation, which corrected the position of the Hexacopter if it was hit by strong winds and adjusted the camera if necessary. The F550 also came with several safety features, such as an Automated Recovery System (ARS) that would return the Hexacopter to its starting location if it lost contact with the controller and a low-battery warning system that would land the Hexacopter immediately if it detected that it did not have sufficient power to keep flying.

3) *Batteries*

The Hexacopter platform was powered by a single Lithium Potassium, or LiPo, 11.1 V (3 Cell) rechargeable battery. These batteries were used as they have a reasonably high energy density and are used for many industrial multirotor applications [8]. However good care must be taken with these batteries in order to prolong their life and ensure that they can be used over and over again without breaking. Failure to do so can result in the internal resistance of the batteries substantially increasing, which is often visible as a bulge or swelling on the battery.

Generally speaking, LiPo batteries must not be allowed to drop below about 30% of their maximum charge and whenever they are being charged, the charging cycle must not be interrupted early. To make these guidelines easier to follow, we used a balance charger, the Imaxrc B6AC Pro, which gave us feedback on the charging progress of the batteries and also charged each cell equally.

When new, our batteries gave about 15 minutes of flying time and took about 2 hours to recharge. However as they were used more and more often resistance errors began to creep in. Eventually the batteries held so little charge that the low voltage thresholds on the Hexacopter would be triggered only 5 minutes after taking off, making them virtually useless.

At present the batteries have to be replaced and set on charge manually. While automatic charging stations for UAV's have been proposed [9], as we were only using one UAV the scale of our operation was not large enough to justify such a set-up.

B. *Autonomous Flight*

Of course, although the manual capabilities are important, the real purpose of the project was to explore the autonomous capabilities of such a platform. Autonomous control is important as in some cases an operator may not be able to move the Hexacopter precisely to where it is needed. Rather than design a whole new system from scratch, an autonomous system was used that replicated the signals sent by the controller, allowing the pilot to switch between manual and autonomous control in flight.

1) *Safety*

Even while operating a UAV manually, there is a clear and present risk that someone may be injured as a result, a fact that we were reminded of when someone was injured in Geralton after being hit by a Drone [10]. Of course, making the system autonomous only adds to the danger and we had to ensure that we were operating in clear and safe manner at all times. In Australia, the Civil Aviation Safety Authority (CASA) is the national agency responsible for all aviation safety, including regulations about autonomous flight. When operating the Hexacopter, we had to be sure to follow their requirements, especially making sure we were not operating the Hexacopter near crowds and that we could instantly take over manually when it was flying autonomously [11]. A summary of the relevant CASA regulations for this project can be found in Appendix E.

2) *Raspberry Pi*

In order to control the Hexacopter autonomously, a microcontroller was needed to both replicate the signals produced by the controller and to calculate when those signals had to be generated. Rather than use a separate device for each function, it was decided that it would be easier to simply use a microcontroller that could do both, which meant that we could mount fewer devices on the Hexacopter, improving battery life.

A Raspberry Pi was used as it not only fitted both our requirements, but also came with several standard software libraries that reduced the need for complex programming. Since we had to mount sensors that were USB compatible, the B+ model with 4 USB ports was the best one for our purposes. Raspberry Pi's also have the property that all their software is loaded onto an SD card which they then boot from, so that we could set up multiple cards each with slightly different software and test them all by switching between SD cards. Camera chips also specifically designed for the Pi exist, making it easier to support image processing.

A schematic of the B+ can be seen below, where we can see that a Raspberry Pi takes a 5V power supply, but the only power source in the Hexacopter was the 11.1V battery. Rather than use a separate power supply, a DC-DC converter that could output 5V was used to step down the supplied voltage to an appropriate level.

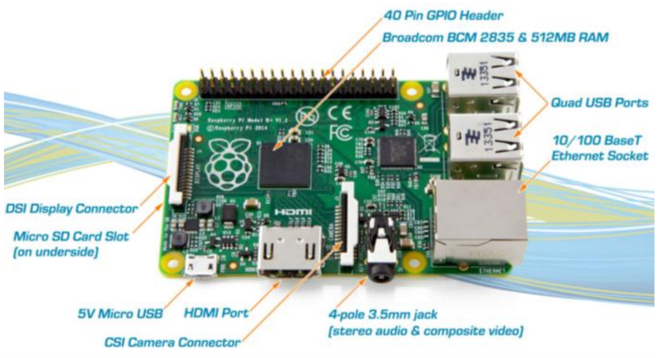

40 Pin GPIO Header
Broadcom BCM 2835 & 512MB RAM
Quad USB Ports
10/100 BaseT Ethernet Socket
DSI Display Connector
Micro SD Card Slot (on underside)
5V Micro USB
HDMI Port
4-pole 3.5mm jack (stereo audio & composite video)
CSI Camera Connector

Fig. 6 - Raspberry Pi Schematic, B+ Model [12]

3) *PWM Signals*

The signals generated by the Controller were Pulse Width Modulated, or PWM, signals. These were capable of being reproduced by the Pi, with help of a special software library known as wiringPi. The signals for each channel had a frequency of 66.67Hz, with a high pulse time that roughly ranged from 1200 to 1900 µs. Further details about the value of each particular channel used can be found in Appendix F.

4) *Switching Circuit*

In order to determine the flight mode, a switching circuit was used to change between the controller and the Raspberry Pi signals. The switching circuit used was developed by Jonathan Brant, a Senior Electronics Technician working at the UWA in the faculty of Electrical and Electronic Engineering and installed on the Hexacopter last year [1]. By reading a control signal, the switching circuit diverts one of two input channels to an output channel. Each channel is capable of supporting up to eight different signals, so it was more than capable of switching between the controller and the Raspberry Pi. A diagram showing how the switching circuit fitted into the circuit can be seen below, with a more detailed version in Appendix G.

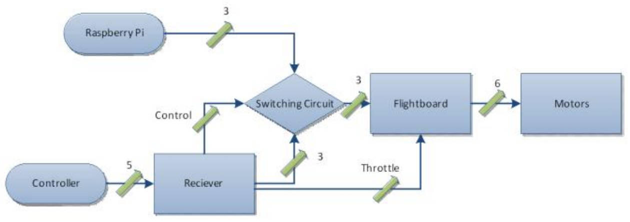


Fig. 7 – Autonomous Control with the Switching Circuit

It can be seen from the diagram that the Throttle, responsible for controlling the altitude of the Hexacopter, was actually hardwired directly into the Flightboard and bypassed the switching circuit entirely. Although the original intention had been to also control the altitude autonomously, this proved too difficult achieve safely as it was strongly dependent on the remaining power available in the battery. Unfortunately, as we were using LiPo batteries, this was not a simple linear relationship and to make matters worse, the exact Voltage-Current curve varied from battery to battery and slowly changed over time. This meant that trying to implement altitude control proved to be too risky to achieve safely and for this reason the control was simplified to a two dimensional problem as alluded to earlier.

5) *GPS Module*

As mentioned earlier, the most suitable way for the Hexacopter to determine its orientation was with a GPS system as this could be used in a variety of outdoor settings. While the F550 comes with its own GPS module, this was propriety hardware, meaning that it was locked and we could not access it for our purposes. This meant that we had to install a separate module for the Raspberry Pi to access so that it could reliably determine the position of the Hexacopter.

The module selected was the QSTARZ 818X-BT. The main reason that this model was selected was that it that had been successfully used in other Robotics Projects, including last year's Hexacopter Project [1][2]. However, it was not in any way inferior to our purposes as it had a standardised USB plug that the Pi could easily interface with and was capable of determining a position fix accurate to within 3m in less than 35 seconds. Research of other possible models indicated that this was one of, or close to, the best available in the field at the time.

Fig. 8 - QSTARZ 818-X GPS Module

III. SOFTWARE

A. *Standardised Libraries*

Rather than develop a whole software system for controlling the Hexacopter from scratch, it was decided that just as we had modified an existing platform for our uses, we would also take advantage of the many software libraries also developed for the Raspberry Pi. The Raspberry Pi is an open-source piece of hardware, and its creators have strongly encouraged the creation of software by its users, so a lot of people have developed software for others to use. In particular, a lot of the low-level functionality of the Pi such as reading and sending digital signals already existed.

1) *Raspian*

Raspian is a Linux based operating system (OS) provided by the Raspberry Pi Foundation that can be installed on a Raspberry Pi allowing it to boot up in a manner similar to that of a Desktop computer. Using Raspian meant that we had a pre-built environment to load and test our own programs without having to worry about the low-level microcontroller management of the Pi, which was the main benefit of actually using a Pi in the first place. The Raspian OS, along with instructions for its use, can be downloaded from the Raspberry Pi Foundation website [13].

2) *wiringPi*

In order to send and receive data with the Pi, the most common method is to use the General Purpose Input and Output (GPIO) pins. These allow for more generic interfacing compared to other data ports such as USB's and a lot more devices can be connected to them. The wiringPi libraries, allow these pins to be set up for any type of data transfer [14]. While wiringPi also has the ability to create PWM signals, we found that the signals it could generate were not suitable for our purposes, similar to what was found last year [1].

3) *servoBlaster*

To create PWM signals identical to those generated by the controller, the ServoBlaster library was used as this could generate PWM signals at the desired frequency in steps of 10μs, fine enough for our needs [15] . These signals were tested in the laboratory to ensure that they replicated the desired signals exactly. Appendix F contains further information about these signals.

B. *User Programs*

Of course, software did not exist that already covered all of our design requirements and so we wrote some of our own programs. Almost all of the programs generated were written in the C++ language as this was compatible with the low-level libraries used and could be written up and tested on other systems.

1) *Reading Sensors*

Although the GPS module easily interfaced with the Raspberry Pi through a USB connection, there was no software provided to process the data so we had to write our own libraries. Using wiringPi, we were able to read directly the raw data being emitted by the GPS, which was in National Maritime GPS Association (NMEA) form.

```
$GPGGA,050126.000,3158.7593,S,11548.9611,E,1,
5,5.84,18.6,M,-29.4,M,,*51
$GPGSA,M,3,30,26,15,07,28,,,,,,,,6.15,5.84,1.92*01
$GPGSV,3,1,11,28,69,137,33,26,56,212,36,05,40,30
7,,30,37,125,33*7C
$GPGSV,3,2,11,13,32,058,,17,27,038,,15,18,225,15,
10,14,359,*79
$GPGSV,3,3,11,07,10,105,17,09,01,048,,46,,,*7B
$GPRMC,050126.000,A,3158.7593,S,11548.9611,E,
0.58,346.65,261014,,,A*7D
$GPGGA,050126.200,3158.7592,S,11548.9611,E,1,
```

Fig. 9 - Sample NMEA GPS Data generated by our module

In the raw data, the position data is always expressed as a string beginning with the header '$GPGGA', which can be seen in the top and bottom sample lines. This string always contains the the latitude and longitude data, along with their hemisphere identifiers, separated by commas. This means that the position data can then be extracted by scanning the input from the GPS until a string with the appropriate header is found, then counting along the commas until the appropriate location.

Once the raw data had been identified, it had to be converted into an appropriate form. According to NMEA standards [16], the latitude (or longitude) is represented as a decimal number which is equal to one hundred times the degree value of the latitude (or longitude) plus the number of arc minutes, along with a character representing what hemisphere the GPS is in. This can be converted into an absolute value in of degrees by the formulas shown below.

$$lat_{actual} = \left(\left\lfloor \frac{lat_{raw}}{100} \right\rfloor + (lat_{raw} \% 100) \times 60 \right) \times lat_{identifier} \qquad (1)$$

$$long_{actual} = \left(\left\lfloor \frac{long_{raw}}{100} \right\rfloor + (long_{raw} \% 100) \times 60 \right) \times long_{identifier} \qquad (2)$$

Where:

- lat_{actual} is the latitude, in degrees
- $long_{actual}$ is the longitude, in degrees
- lat_{raw} is the raw latitude value outputted by the GPS, in NMEA format
- $long_{raw}$ is the raw longitude value outputted by the GPS, in NMEA format
- $lat_{identifier}$ is 1 if the latitude hemisphere character is 'N', -1 if it is 'S'
- $long_{identifier}$ is 1 if the longitude hemisphere character is 'E', -1 if it is 'W'
- $\lfloor x \rfloor$ is the floor, or whole number part, of x
- $\%$ is the modulo, or remainder operator

2) Waypoint Navigation

In order to travel between locations, the Hexacopter would measure its current position and compare that to its target. By determining the compass bearing between the two locations, this could then be converted into channel commands for the Hexacopter, based on its current orientation.

To determine its current orientation without a compass, the Hexacopter had to perform a bearing test so that it could calculate its bearing using only a GPS. It did this by measuring its current GPS location, then flying forwards for several seconds, then measuring its new location. By comparing the latitude and longitude of where it started and finished, its orientation could then be determined.

In a similar manner, the compass bearing the Hexacopter had to fly in could then be determined by comparing its current position to its target. Then, by comparing the bearing with the Hexacopter orientation, the relative direction the Hexacopter had to travel in could be determined. As we had no feedback about the orientation, the rudder command was set to zero so that the orientation would remain constant.

However while this gave a relative indication of the Aileron and Elevator commands, it gave no information about their actual values. This was calculated by determining the distance between the start and end positions, then using a proportional, or P, controller to determine their actual values. Since the start and end positions were points on a sphere, the Haversine formulas below in (3) and (4) was used to determine the distance, similar to what was used last year [1][2].

$$d = 2\,R\,sin^{-1}\left(\sqrt{sin^2\left(\frac{\emptyset_2-\emptyset_1}{2}\right)+cos(\emptyset_1)\,cos(\emptyset_2)\,sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right) \quad (3)$$

$$\theta = \tan^{-1}\left(\frac{\sin(\lambda_2-\lambda_1)\cos(\emptyset_2)}{\cos(\emptyset_1)\sin(\emptyset_2)-\sin(\emptyset_1)\cos(\emptyset_2)\cos(\lambda_2-\lambda_1)}\right) \quad (4)$$

Where:

- $\emptyset_1$ is the starting latitude
- λ_1 is the stating longitude
- $\emptyset_2$ is the ending latitude
- λ_2 is the ending longitude
- R is the radius of the Earth

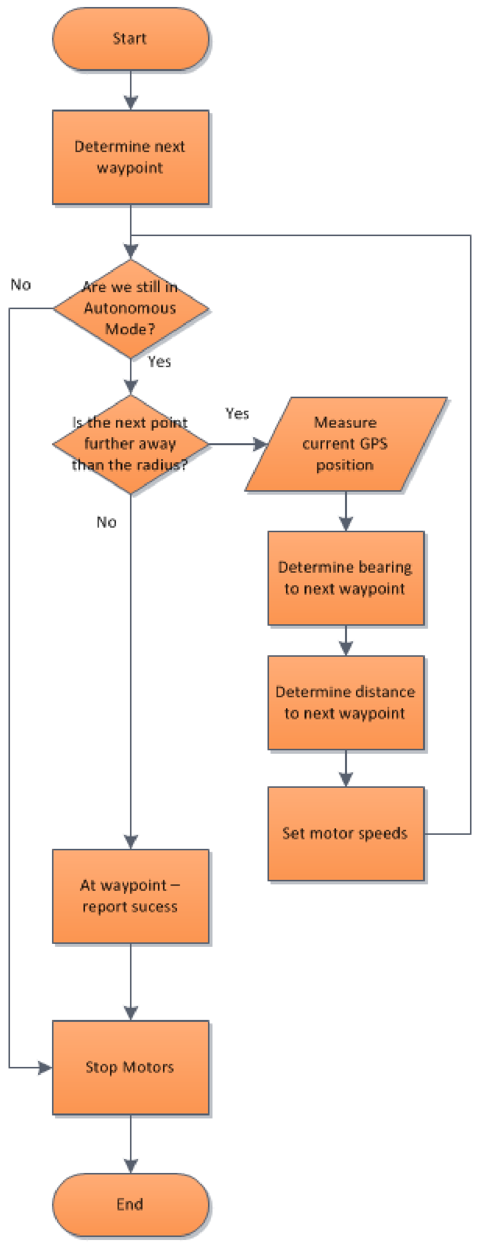

Start
Determine next waypoint
Are we still in Autonomous Mode?
No
Yes
Is the next point further away than the radius?
Yes
No
Measure current GPS position
Determine bearing to next waypoint
Determine distance to next waypoint
Set motor speeds
At waypoint – report sucess
Stop Motors
End

Fig. 10 - Process the Hexacopter would use to fly to a waypoint

3) Creeping Line Search

In order to search a target area in a Creeping Line pattern, the overall area was broken down into a series of target points so that it all the points ordered into a flight and then the method of flying to the waypoints could be called iteratively until all of the points had been reached. The area to search was assumed to be rectangular and aligned with lines of latitude and longitude as this allowed the Pi to calculate the internal points much quicker.

To calculate the points, the start and end points were fed into the program, which were then used to calculate all four corner points. From these, the ends of the sweeps were determined. Finally, intermediate points along the sweeps were also calculated in order to improve the accuracy of the flight, as can be seen in the figure below.

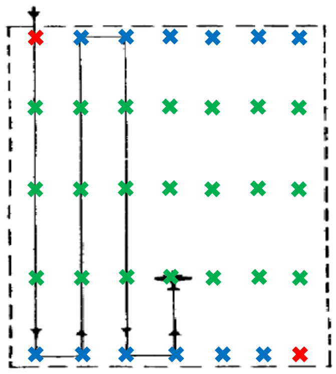


Fig. 11 – Generated points for the Creeping Line Pattern. The Red points are the starting locations, the Blue points are the ends of the sweeps and the Green points are the intermediate points. Background image courtesy of WDSOT, 1997 [5]

The flow chart following shows the overall flow of the Program used to fly to each point in the Creeping Line Pattern. Firstly, the Hexacopter would check that the GPS Module was attached and that sensible data could be read from it, in order to prevent the Hexacopter flying without a proper position fix. Then, the Hexacopter would generate all the points it had to fly to before waiting to be put into autonomous mode. Once it had permission to do so, it would then fly to each point in the list, checking the whole time that it was still allowed to be flying autonomously. If it was switched back into Manual mode, or it finished flying to all the points, it would then stop flying and exit the program. While the stop command would be ignored if it had been switched back into Manual mode, this was still important to include so that the Hexacopter would not continue to fly off in a random direction if it were inadvertently switched back.

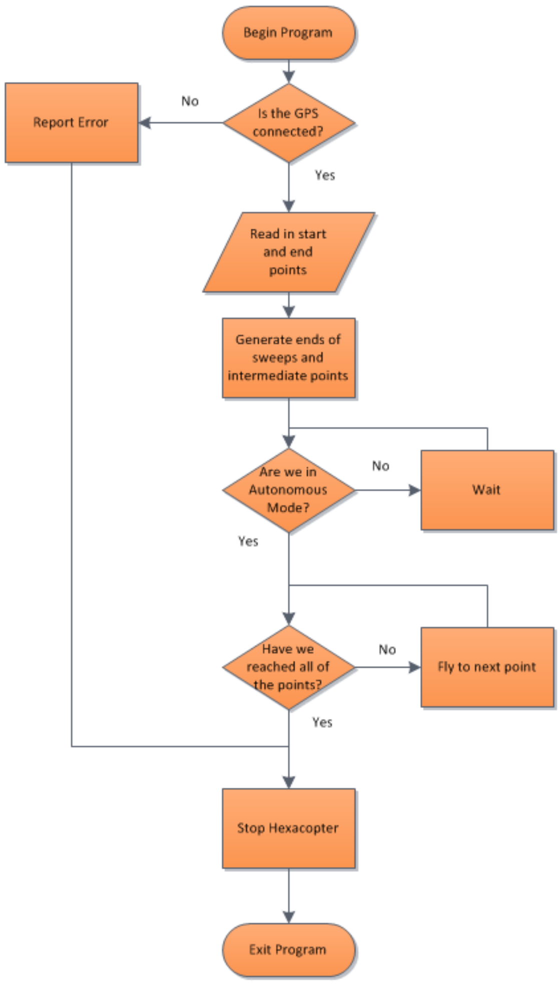

Begin Program
Is the GPS connected?
No
Report Error
Yes
Read in start and end points
Generate ends of sweeps and intermediate points
Are we in Autonomous Mode?
No
Wait
Yes
Have we reached all of the points?
No
Fly to next point
Yes
Stop Hexacopter
Exit Program

Fig. 12 - Generation of the Creeping Line Search Pattern

4) Image Processing

Of course, it was important that the Hexacopter was able to more than just fly around. Software was developed allowing it to process images in-flight using the Raspberry Pi Camera. To overcome the fact that we would be investigating objects some distance away, a Hue, Saturation and Value (HSV) scheme was used to identify colours as this was found to be superior to the Red Green Blue (RGB) system that is conventionally used in image processing.

In order to perform the Image Processing, the Raspberry Pi Camera module was used to capture images while in the air. While this module is normally capable of taking images with a resolution of 640×480 pixels, it was found that this was much too slow for our purposes. By reducing the resolution to a quarter of its standard value -320×240 pixels – we were able to get image processing speeds up to around 25 frames per second, almost fast enough so that the video feed would not seem jerky to an observer.

Most of the actual processing was done using the Open CV libraries which are used in a variety of Linux systems, not only Raspian. These libraries were used to perform a mean-shift search on the image so that the Hexacopter would be capable of tracking multiple objects at once.

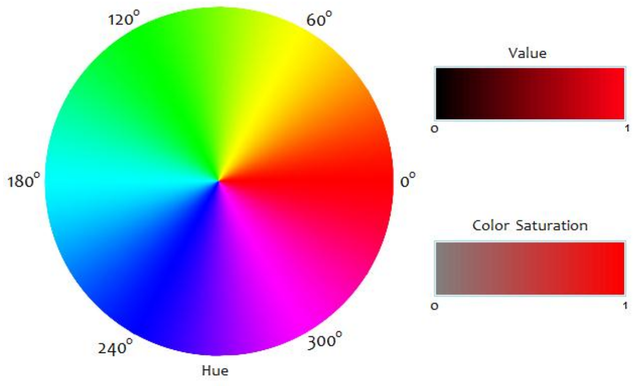


Fig. 13 - HSV Colour Wheel [17]

5) *Web Server*

All of our programs would be useless if they could not be accessed readily and easily. A web interface was developed so that users would be able to send commands to the Hexacopter in flight, from everyday devices such as a smartphone or a laptop.

The server was hosted on a network that the Pi generated itself on boot, meaning that a user could connect by simply being in Wi-Fi range, several hundred meters in our case, and then opening a website on a browser. For security reasons, the network was made password-protected so that we could restrict access and be sure no unauthorised flight commands were being sent to the Hexacopter.

The website was designed to be very user friendly, displaying the location of the Hexacopter superimposed over a satellite map of the local area, with a trace visually recording the flight path. Users could also see the live camera feed being streamed down from the Raspberry Pi Camera.

C. *Overall Layout*

The diagram below shows the overall structure of the software installed on the Pi for this project. The bottom or 'Base' level contains all of the basic libraries that were used throughout most of our programs. The middle, or 'Modules' level used elements from several of the Base libraries and combined into self-contained programs that fulfilled a specific purpose such as Navigating via GPS locations or analysing images. Most of the code that I wrote for this project was in this level. Finally, the top or 'Applications' level had programs that wrapped around the files from the Modules level and allowed them to be used in actual programs that could them be used on the PI, either via the Web Interface or smaller testing programs that only considered a particular aaspect.

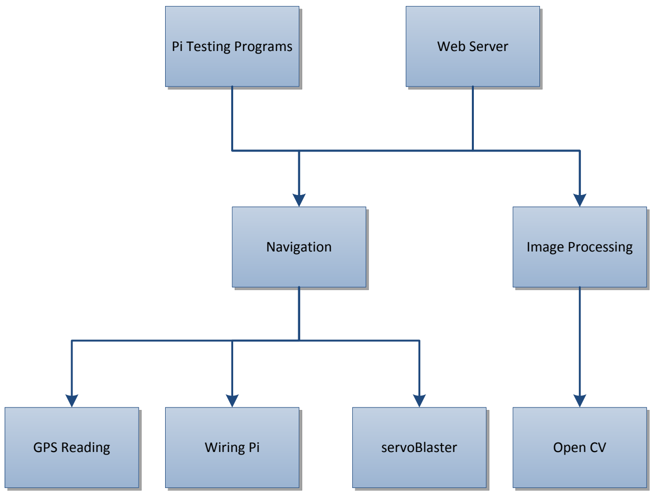


Fig. 14 - Software Structure, with the top 'Applications', middle 'Modules' and bottom 'Base' levels

D. *Additional Software*

In addition to the software running on the Pi, several other programs were used in the laboratory to assist with other functions such as writing code, connecting to the Pi and receiving feedback from the Hexacopter. As the Pi's processor, while still powerful, was not that fast compared to a desktop computer, it was often faster for us to write code on some other machine and then transfer it over to the Pi.

1) *PuTTY*

Using PuTTY, we were able to connect to the Raspberry Pi and run programs on it form our own computers using a terminal interface over a network connection. Similar to that found when you boot up a Pi, this meant that we could have multiple students using the same computer, each of us making their own changes to their part of the program. This was also handy to use in the field so that we did not have to use a screen or power in order to get our programs started.

2) *WinSCP*

While we could run programs on the Pi with PuTTY, we needed other programs to get files on there in the first place. WinSCP is a file transfer program for Windows that allows file to be transferred over two computers using the same network with a click and drag user interface. This was used in the lab to transfer files to and from the Hexacopter.

3) *NAZA-M Assistant*

DJI provided software for users to interact with the Hexacopter while it was landed. Using the NAZA-M assistant software, we were able to properly calibrate the Hexacopter before use and to set up the Flightboard. The interface also allowed us to get feedback about the PWM signals generated by the controller. Further information about this software can be found in Appendix A.

IV. FINDINGS

A. *GPS Drift*

Testing for the error associated with the position bearing of the QSTARZ GPS indicated that the nominal error quoted by the manufacturer did not tell the entire story. By leaving the GPS unit in one place and repeatedly taking position measurements, we were able to obtain a more accurate estimate of the error that would be associated with the position drift of the GPS. As shown in the figure below, this error was not constantly around 3m, as claimed by the manufacturer, or around 0 m as it was in reality, but instead slowly increased as we kept measuring for longer and longer periods.

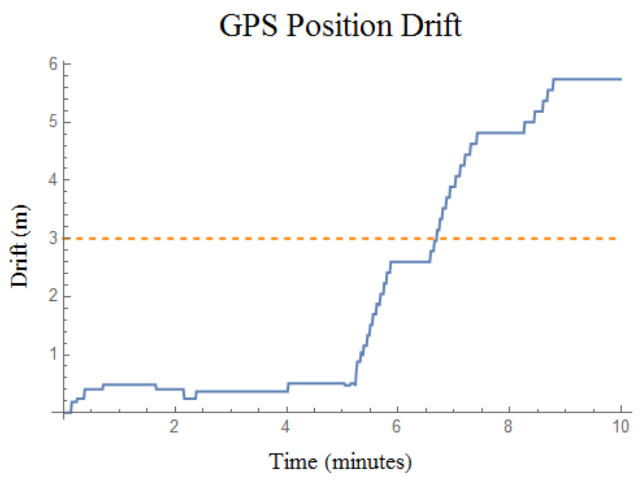


Fig. 15 - Error in position due to GPS Drift. The orange dashed line is the nominal error of 3m and the blue solid line is the measured drift.

Given that the error increased with time, the obvious solution was to make sure that not much time passed between the start and end of our flights. Even with new batteries, the maximum amount of flight time we could get was about fifteen minutes, but by flying in short bursts of no more than five minutes at a time, we could ensure that the maximum possible position error was kept to under a meter.

It was also noted that the accuracy of the GPS decreased substantially whenever we could not get a sufficiently strong connection to enough satellites. Measurements in the laboratory were noted to be substantially less accurate than those made outside, or on cloudy days. One of the advantages of using the safety features on the DJI F550 was that we could tell if it took a long time to start up because its own GPS/Compass unit would take a while to get a lock, then this would mean that the Qstarz module used by the Raspberry Pi would also struggle to obtain a strong signal.

B. *Waypoint Accuracy*

Once we knew the accuracy of our GPS, we knew what sort of error we would have to expect in our flight. As the error from the GPS could not be ignored, this meant that we could never be sure that we were exactly where the GPS claimed we were, so we could be trying to fly to a point but never exactly reach it, because the GPS would drift around too much.

To overcome this, we had to include an allowance for some positional error whenever we attempted to reach a waypoint. So rather than considering our waypoints as points, we had to consider them as circles and assume that the point had been 'reached' when the Hexacopter was inside that circle. By tightening the bounds more and more, managed to get more and more accurate runs and we were able to use targets as small as one meter.

However while reducing the size of our target circles improved accuracy, it also led to other problems. When using a proportional controller to determine speed, as we got closer and closer to the waypoint, the Hexacopter travelled slower and slower. This meant that our runs took longer to complete, increasing the risk of GPS drift as mentioned above. Also, the Flightboard interpreted any motor commands close to zero as zero, as a safety feature in case of improper calibration, so sometimes the Hexacopter did not move at all if it was given very small commands.

Clearly, something better was needed and so we switched from a simple P (proportional) controller to a PI (Proportional and Integral) one. By using a discrete PI controller, as seen in (5), we not only considered the present error in the position of the Hexacopter but also its past errors. This greatly improved performance and overcame the two problems mentioned above with the P model – Flight time and hitting the 'dead band' of the Flightboard.

$$s_i = K_p \times d_i + K_i \times \Delta t \times \sum_{j=i-n}^{i} d_j \qquad (5)$$

Where:

- K_p is the Proportional gain
- K_i is the Integral gain
- Δt is the time difference between measurements
- d_i is the ith distance between the Hexacopter's current position and its desired position
- s_i is the ith speed

Although adding in a Derivative Control element as well was considered, this was not included because the role of a Derivative Controller is to improve settling time of the system. However as we were considering waypoints to be circles, we did not have to worry about the system settling down to an exact value and so this element would have been redundant. Leaving out the Derivative Controller also meant that we did not have to filter out the high frequency noise that would have been amplified by a derivative term.

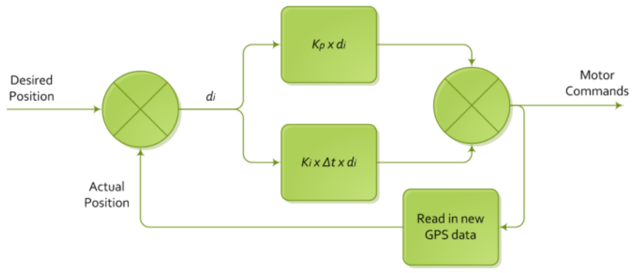


Fig. 16 - Control Loop used to determine the Hexacopter's speed

In order to properly tune the PI controller, the Zeiger-Nichols method was used to optimise the values of the parameters K_p and K_i. First, K_i was set to 0, and then K_p was slowly increased until the resulting loop oscillated about the mean position. Once this oscillation gain K_u was found, then K_p and K_i could be calculated according to the formulas below for a PI controller.

$$K_p = 0.45\, K_u \tag{6}$$
$$K_i = 1.2\, K_u \times T_u \tag{7}$$

Where:

– K_p is the tuned Proportional gain
– K_i is the tuned Integral gain
– K_u is the Proportional gain that just causes oscillation
– T_u is the period of oscillation

Using this method, I found the value of K_u that just caused oscillation to be 25 with a frequency of 1Hz, which led to K_p and K_i values of 12 and 2.4 respectively.

C. *Creeping Line Run*

Using these determined values for the Control Loop constants, I was able to get the Hexacopter to perform more reliable Creeping Line searches, an example of which can be seen below. In this example, the target waypoints generated by the Hexacopter show up as grey diamonds, with the dotted line between them being the desired path of travel and the actual path travelled being shown by the coloured lines (each different colour represents the Hexacopter flying to a different waypoint).

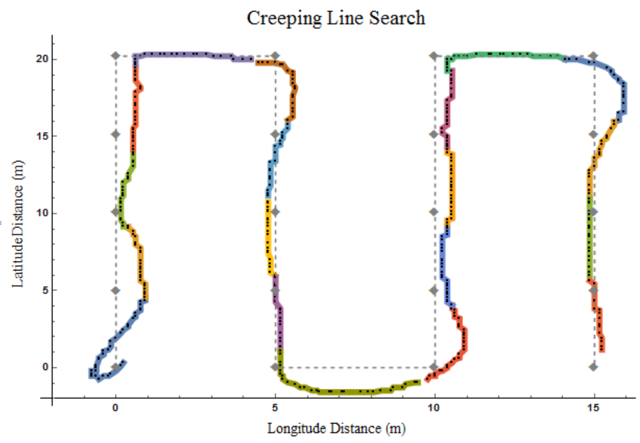


Fig. 17 – Creeping Line Run with Intermediate Points

It can be seen that the Hexacopter was able to follow the desired path very closely, with a few deviations of about a meter occurring at the ends of some of the sweeps when it changed direction. Since these only occur at some of the ends and are always in the same direction, they are almost certainly the result of wind affecting the Hexacopter and dragging it off course. This is further reflected by measurements we took regarding wind speed, which were in the same direction as the deviations to the plot.

It can also be seen that the deviations did not have the chance to affect the overall path the Hexacopter travelled, with the intermediate points, being added in at 5m intervals, causing it to quickly return to the desired path after it had left. Note also that with this Creeping Line search the Hexacopter finished on the same side that it started; this is because the overall target area was only 15m wide and the Hexacopter was told to make sweeps 5m apart, so the number of sweeps it needed to make to properly cover the area was even.

V. Conclusion

A. *Meeting Aims*

I am very happy with the progress made throughout the year and I believe I successfully managed to achieve the goals I had intended with regards to the Hexacopter navigation. I successfully managed to identify and implement a professional Search and Rescue method to scan an area by air with a UAV – as could be seen earlier, the Hexacopter used the Creeping Line Search pattern to navigate over an area with a reasonably high degree of accuracy. The Hexacopter was also able to navigate much closer to waypoints, around one meter rather than four, thanks to a tuned PI controller. The groundwork that I have put in this year will serve as an excellent base for any students who wish to continue working on this project in the future.

B. *Future Work*

Although a lot of work was done on the Hexacopter navigation this year, there certainly is room for improvement,. The following are a few suggested areas that students wishing to work on this project next year may want to look at to improve the reliability and the robustness of the existing system.

1) *Altitude Control*

While using the Throttle autonomously safely proved to be too difficult this year, turning the two dimensional motion of the Hexacopter into three dimensional motion should certainly be looked into in the future. Implementing altitude control will allow the user to be able to set waypoints in three dimensions, but a more accurate method of determining height may need to be considered as the only currently installed sensor, the Qstartz GPS is unable to determine height readings very accurately. Possible solutions may include a downward facing laser sensor that constantly measures off the ground or an accelerometer.

2) *Installing a Compass*

While using the GPS to determine the bearing was adequate for our purposes, it would probably be worthwhile to include a sensor that is capable of giving feedback about the orientation of the Hexacopter in in real time. Not being able to know the orientation meant that we had to assume that it was constant throughout the flight. While we were able to account for this in our code, it meant that the Hexacopter was unable to correct for external factors that spun it around mid-flight, causing its orientation to change.

3) *More Robust Flight Plan*

Adding intermediate points to the Creeping Line search pattern improved its accuracy, particularly in windy conditions, but also meant that the Hexacopter flew slower as it tried getting to more points. In order to improve performance while retaining speed, an idea that could potentially be implemented next year is to have the Hexacopter fly following a line, rather than to a point.

The Hexacopter would then fly to the ends of the sweeps using a PI controller as before, but also try and stay close to the line between the two using a ‘bang-bang’ or a P controller. As it would not be flying to close to its target point the PI controller would keep it moving quickly but the ‘bang-bang’ controller would keep it on the desired flight path.

4) *Simulator*

All of the data and testing of our flight code that we had to collect had to be done in the field, a process which was often rather time consuming. In order to save time, it may be worthwhile for any students wishing to continue this project next year to look into a Hexacopter simulator which can replicate the function of the F550. While a simulator would not replicate some of the outdoor features such as wind that you would get in the real world, it could still assist students by providing a platform where they could comfortably and quickly test their flight codes in the laboratory without any of the dangers of failures that you could get outdoors [18].

However, performing simulations of a Hexacopter system is inherently complicated as you have to simulate several complex factors, such as three dimensional motion. This was one of the reasons simulations were not used this year. A possible solution is to use one of the more advanced Robotic Control Systems available, such as ROS, Robotic Operating System, to handle the programming while using a separate simulator for the graphics [19] .

VI. Appendix

A. *DJI F550 Hexacopter*

Developed by DJI Industries [7], the F550 is an integrated Hexacopter kit that serves as a base for the user to develop their own UAV platform. The base kit consists of two integrated modules – the Flightboard and the GPS/Compass along with six sets of motors, rotors and ECG’s.

1) *Flightboard*

The Flightboard is an integrated module that is responsible for interpreting the commands coming in from the joystick and converting these to the desired motor speeds. The Flightboard must be placed as close to the centre of mass of the Hexacopter as possible in order to function properly. The one we were using could also be pre-programmed to perform several key tasks, such as an auto landing function if it ever lost connection with the controller, which we added in as an extra safety feature.

In order to monitor the status of the Flightboard while in the air, an external LED sensor relayed information back to the pilot by varying the colour and number of blinks. For example,

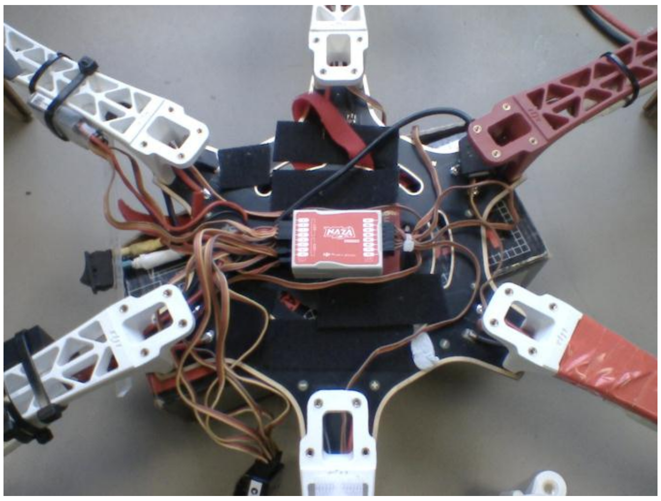


Fig. 18 - Basic DJI Kit, with the top plate removed showing the red Flightboard module mounted in the centre of the aircraft

2) *GPS/Compass*

Combined into one unit, which was mounted high above the main body of Hexacopter in order to reduce interference from the motor spin, the GPS/Compass was used by the Flightboard in order to determine the location and orientation of the Hexacopter. Although we were using our own version of these instruments on the Raspberry Pi, maintaining them as redundant components that interfaced directly with the Flightboard meant that we could be sure that the Hexacopter was capable of maintain its own stability and position control even if the Raspberry Pi or our own programs failed.

3) *Motors*

Each of the motors was mounted at the end of an arm, with an ECG mounted in the underside. Each ECG took in the DC

12V and Ground Voltages from the battery, as well as a control signal from the Flightboard, and converted these into the voltages sent to each motor.

4) NAZA-M Assistant Software

The Assistant Software provided with the Pi was very useful and allowed us to do a variety of tasks. With the Assistant Software, we could use our own computers to pre-set some of the DJI safety features such as the auto-landing function. Also, we had to use the software before we could take off to set the location and orientation of the GPS/Compass module relative to the Flightboard. This allowed the Hexacopter to use its own internal self-stabilisation routines, even when it was receiving signal from the Raspberry Pi.

However the most useful feature of the Assistant Software that meant we saved a lot of time during testing was that it outputted the relative strengths of the joystick commands, as received by the Flightboard. This was used when calibrating the PWM signals in order to determine what the rise time of the signals being sent by the Pi should have been. It was also used when testing our flight code, as it meant that we could see what signals the Pi was sending the Flightboard in the lab without having the Hexacopter take off.

The latest version of the Assistant Software that we were using at the time of writing this report was 2.20.

B. Alternative Search Patterns

Several other Search Patterns were considered, but ultimately found to be inferior for our purposes compared to the Creeping Line Search Pattern which we ultimately used to autonomously scan over an area.

1) Expanding Square Search Pattern

As can be seen below, this pattern involves spiralling outwards in an ever-expanding square. This pattern is useful if you do not wish to fly very far or if you know that the object that you are looking for is very close to your starting location, but it also requires you to have very good methods of maintaining your position in order to avoid gaps. As we were developing a system that could be used in outdoor settings where any external factors could cause us to deviate from our intended path and no information about our targets would be known we decided to not consider this pattern.

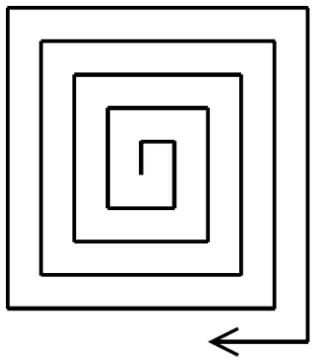


Fig. 19 - Expanding Square Search Pattern [5]

2) Sector Search

This pattern involves making repeated sweeps over a small target. This pattern is mainly used for obtaining more information about a particular target in a very small area. As we were only wished to identify specific objects over large areas this pattern was not particularly useful.

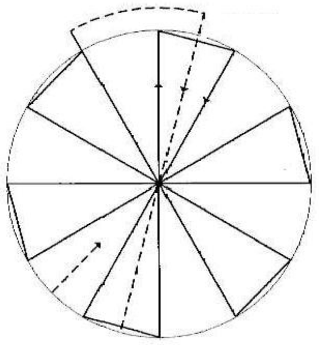


Fig. 20 - Sector Search Pattern [5]

3) Contour Search

With this pattern, the UAV tracked over an area, following contour lines so it would always be tracking a section of the ground at the same height above sea level. This pattern was deemed unsuitable as the Hexacopter was not able to control its height autonomously, so an operator would have had to assist with the search anyway

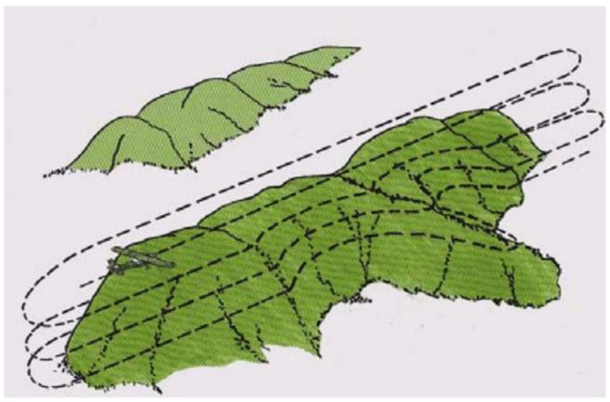


Fig. 21- Contour Search Pattern [5]

4) *Complex Scan*

This pattern is the same as the Creeping Line Pattern, but with a second pattern superimposed over the first one at right angles. For our purposes, as we intended to track objects that are moving much slower than the Hexacopter, any objects that we detect on the second sweep we will have already picked up on the first one, so there will be nothing gained.

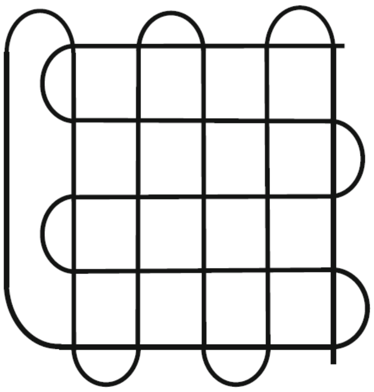


Fig. 22 - Complex Scan Search Pattern [20]

5) *Figure-of Eight*

One of the more simple patterns, this pattern involves making a continuous sweep in a figure-of-eight over an area. This pattern is only useful for obtaining repeated data about a particular target that you have already identified, not identifying objects of interest in the first place.

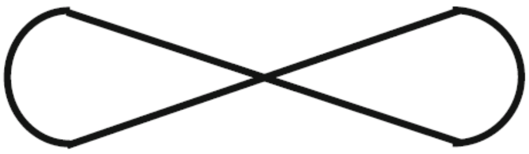


Fig. 23 - Figure-of-Eight Search Pattern [20]

6) *Particle Swarm Optimisation (PSO)*

This pattern involved using multiple UAV's in order to generate an evolutionary search pattern that rapidly converged on a moving target. However this was not useful for several reasons, namely that it required more than one UAV and prior knowledge of the target, neither of which we had.

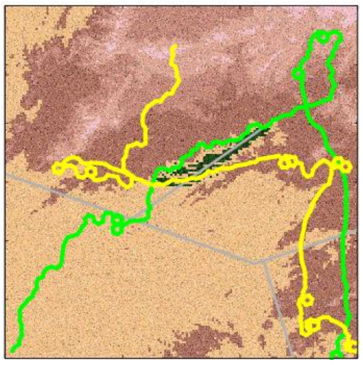


Fig. 24 - PSO Search Pattern. Each Coloured Line represents a different UAV [21]

7) *Independent Circular Track Pattern*

While this pattern was originally designed to be used to track objects moving in a straight line, it was considered to be suitable also to scan an area as it offered a high level of overlap in case objects were missed. However initial tests of the Image Processing suggested that the Hexacopter only needed to perform one sweep over an area to detect the desired object and so this pattern was dropped in favour of a simpler one.

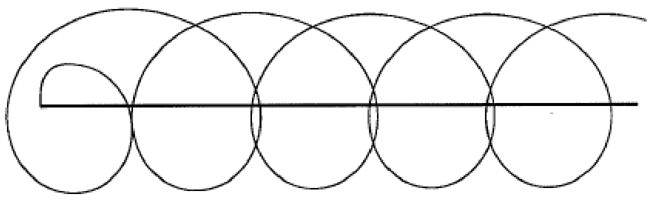


Fig. 25 - Independent Circular Track Pattern [22]

C. *Literature Review*

1) *Previous Students*

a) *Multirotor Umanned Aerial Vehicle Autonomous Operation in an Industrial Environment using On-board Image Processing, Venables C., 2013[1]*

This paper, written by Chris Venables in October 2013 for his Final Year Engineering Thesis on the Hexacopter, was the most widely-used resource for this project. It set the standard for our goals and aims and gave us a good background for what was achievable with the current system. The information on GPS navigation was extremely useful for me in particular, especially as many of the systems that I was working with were the same ones used last year. Unfortunately, the thesis was a little long to read (over 120 pages) and this made it a little hard to always find the relevant information.

b) *Developing a Multicopter UAV Platform to Carry Out Research into Autonomous Behaviours, using On-board Image Processing Techniques, O'Connor, R., 2013 [2]*

Rory O'Connor's paper still contained important information about the Hexacopter project, but did not go into as much detail, and focused more on the areas that I was not working on, such as Image Processing. However its brevity did make it somewhat easier to read and it was often handy for providing a quick summary about a particular area of the project.

c) Development of a Navigation Control System for an Autonomous Formula SAE-Electric Race Car, Drage, T.H., 2013[3]

While also dealing with autonomous navigation, this paper focused more on sensor fusion and tying together measurements from a whole range of devices such as laser scanners and compasses, not only GPS's. It still gave a good overview of the limitations of an autonomous system and also indicated the importance of always making sure to log data out in the field for later analysis in the laboratory, something which saved us a lot a of time and meant that we could perform tests more efficiently.

2) GPS Measurements

a) Validity and reliability of GPS for measuring distance travelled in field-based team sport,s Gray A.J. et al, 2010 [23]

This paper highlights a study done at the School of Human Movement Studies at the University of Queensland measuring the accuracy of GPS measurements of moving objects. The study found that moving in non-linear paths reduced the accuracy of GPS measurements and that this could also be overcome by increasing the GPS update rate (1 Hz in their case). This was a very useful finding, as it meant, as it meant that we could limit our Hexacopter motion to linear paths and still maintain a high degree of accuracy.

b) Advanced motion control and GPS guide car steering robot, Palmer, D., 2006 [24]

This journal article was rather short and brief, with not much information. About the only useful piece was the claim that the accuracy of the GPS's could get to as low as 10 cm – this was used as a goal that we should strive to.

c) Vehicle Dynamics Control Based on Low Cost GPS, Zhang, J., 2006 [25]

This article was quite in-depth and went into a lot of detail about using GPS measurements on vehicles. Most of the study involved the use of ground vehicles, but it was still applicable for our use. The study also went into detail about how GPS measurements can be used to supplement other navigation systems, but in our case the GPS was the only such system so this was not so useful.

3) Flight Planning

a) Chapter 11: Visual Search Patterns, WSDOT, 1997 [5]

While strictly speaking a government publication and not an academic publication, this reference was the most useful for this Thesis apart from the papers from last year. Its handy diagrams, some of which I have reproduced, were very good at readily conveying information about search atterns.

b) Incorporating Heuristically Generated Search Patterns in Search and Rescue, Woolan, H., 2004 [26]

This paper gave a good sense of what is used by professional Search and Rescue teams in order to find objects of interest. While some of the patterns studied referred to planes only and so could not be used, there was still enough general information within for this to be a useful resource.

c) Flight Plan Specification and Management for Unmanned Aircraft Systems, Santamaria E. et al, 2012 [20]

While this paper focused more on programming flight paths in planes, it still contained useful information such as alternative search patterns. It also confirmed our decision that the flight pattern we had selected was the correct one.

d) Mobile Ground Target Pursuit Algorithm, Xiaowei F., 2012[27]

The ideas contained within this paper related mainly to the tracking of moving objects autonomously by air. While the ideas were interesting, ultimately this line of research was not pursued. The paper still contained useful information about the manoeuvrability of Aerial Vehicles which was considered in my code.

e) A New Performance Metric for Search and Track Missions, Pitre et al, 2009 [21]

This paper covered an evolutionary search pattern known as Particle Swarm Optimisation, which could not be used for our purposes as it involved multiple UAV's. For a single UAV, the author recommended using a ladder search pattern, which functioned similar to the Creeping Line Pattern that we had already identified. This pattern was also noted as being superior for instances that you had no prior knowledge of the target, as was the case in our situation.

f) Path Generation Tactics for a UAV Following a Moving Target, Husby C.R., 2005 [22]

This paper focused more on tracking moving objects, but again their method for tracking a stationary object was similar to the Creeping Line Search Pattern. While the ideas for tracking a moving object could not be implemented this year, they may prove useful to future students working on this project.

4) Simulators

a) Real Time Multi-UAV Simulator, Göktoğan A.H. et al, 2003 [18]

This paper outlined the role UAV simulators could play in developing software, and the benefits that could be gained from such a system. While being rather brief, it still provided a reasonably detailed overview of using such a system.

b) Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo, Myer J. et al, 2012 [19]

This suggested a possible solution to the problem of having to simulate both an UAV system and the software to run it – using two different systems that were each responsible for half of the simulation. This solution was also good because both of the systems mentioned have been used by other students at this University on similar projects, so some work has already been done on how to integrate them into a system such as ours.

5) Batteries

a) Lithium batteries: Status, prospects and future, Scrosati B.et al, 2009 [8]

This article was a little old, and did not go into a lot of detail about LiPo batteries. However it still covered most of the basic information about them, including their construction, the internal chemical process that they use to generate energy and how to safely maintain and store them. This made it an excellent reference that we could use to refresh ourselves about LiPo batteries.

b) Automatic Battery Replacement System for UAVs: Analysis and Design, Suzuki K. et al, 2011 [9]

While the ideas mentioned in this paper about an automated charging system for a UAV ended up being too unfeasible to implement, it was still a useful resource, backing up some of the general statements made about LiPo batteries from our other sources and highlighting the benefits of using these in a UAV system.

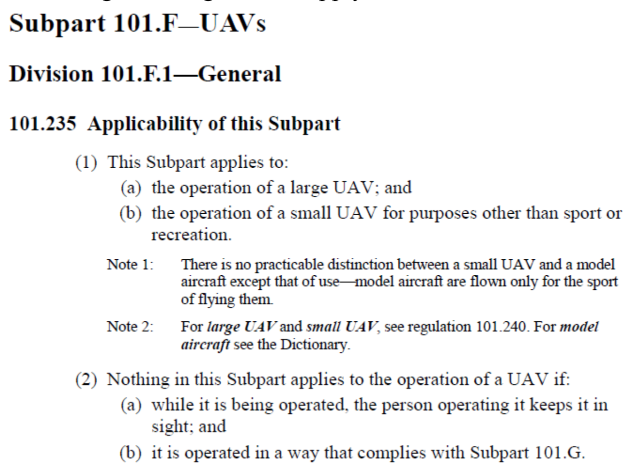
D. Controller

The controller used, the Futuba 14SG, allowed the user to potentially send 8 different control signals to the Hexacopter [6]. The controller transmitted over a 2.4GHz frequency, which meant that we were avoiding possible signal clashes with the Raspberry Pi because its internal processors only ran at 700 MHz. Tests found its range to be about 500 m – almost double the length of a standard oval – so we were confident that we could maintain a secure connection with it while performing all of our testing on James Oval.

The controller is very comfortable to use, which is not just an aesthetics bonus as this means that your hands do not become fatigued when you are using it and you are still able to react quickly to potential dangers. The graphics display, while basic, still relays important information such as the battery voltage and signal strength for each channel, meaning the pilot is able to obtain some feedback about the Hexacopter even when flying manually and the Raspberry Pi is not running.

E. CASA Regulations

CASA have outputted a number of regulations pertaining to the use of UAV's within Australia, both manned and unmanned, form as far back as 1998, with the latest revision to the rules being 2014. Part 101.F of their Safety Regulations [11] outlines the general rules surrounding UAV's, however as we operating a UAV for research purposes only, the majority of this section, particularly the part involving licencing, did not apply.



Subpart 101.F—UAVs

Division 101.F.1—General

101.235 Applicability of this Subpart

(1) This Subpart applies to:
(a) the operation of a large UAV; and
(b) the operation of a small UAV for purposes other than sport or recreation.

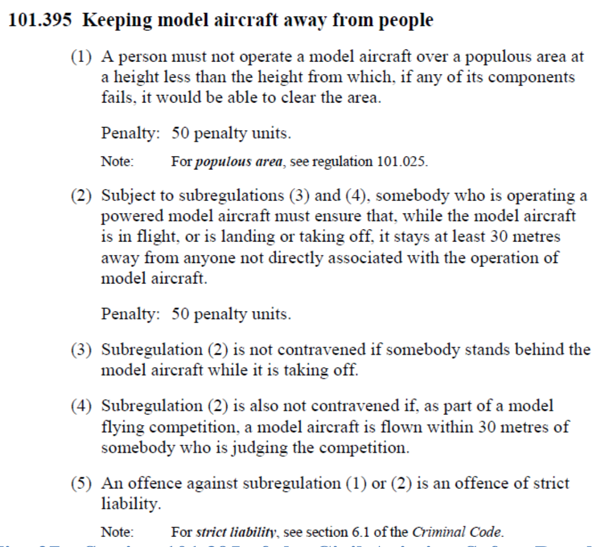
Note 1: There is no practicable distinction between a small UAV and a model aircraft except that of use—model aircraft are flown only for the sport of flying them.

Note 2: For ***large UAV*** and ***small UAV***, see regulation 101.240. For ***model aircraft*** see the Dictionary.

(2) Nothing in this Subpart applies to the operation of a UAV if:
(a) while it is being operated, the person operating it keeps it in sight; and
(b) it is operated in a way that complies with Subpart 101.G.

Fig. 26 - Section 101.235 of the Civil Aviation Safety Regulations (CASA 1998)

However, we still had to obey their general rules surrounding model aircraft as outlined in part 101.G, an excerpt from which is shown below, particularly the sections involving keeping it away from people. This often caused us problems as our flight area of choice, James Oval, was often very crowded between classes, forcing us to wait until the crowds dispersed. Also, members of the public were often very interested in our activities and would want to get close to the Hexacopter while it was flying.



101.395 Keeping model aircraft away from people

(1) A person must not operate a model aircraft over a populous area at a height less than the height from which, if any of its components fails, it would be able to clear the area.

Penalty: 50 penalty units.

Note: For ***populous area***, see regulation 101.025.

(2) Subject to subregulations (3) and (4), somebody who is operating a powered model aircraft must ensure that, while the model aircraft is in flight, or is landing or taking off, it stays at least 30 metres away from anyone not directly associated with the operation of model aircraft.

Penalty: 50 penalty units.

(3) Subregulation (2) is not contravened if somebody stands behind the model aircraft while it is taking off.

(4) Subregulation (2) is also not contravened if, as part of a model flying competition, a model aircraft is flown within 30 metres of somebody who is judging the competition.

(5) An offence against subregulation (1) or (2) is an offence of strict liability.

Note: For ***strict liability***, see section 6.1 of the *Criminal Code*.

Fig. 27 - Section 101.395 of the Civil Aviation Safety Regulation (CASA 1998)

For autonomous flight, CASA regulations state that the normal procedures apply, provided that a human operator is capable of instantaneously taking control of the aircraft. An advisory circular published in July 2002 [28] highlighted additional safety procedures that needed to be taken, including the need for an Automated Recovery System (ARS) that would land the Hexacopter automatically if the signal was lost, especially for operations in populated areas.

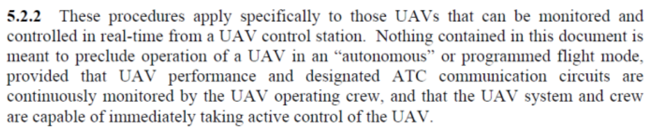

5.2.2 These procedures apply specifically to those UAVs that can be monitored and controlled in real-time from a UAV control station. Nothing contained in this document is meant to preclude operation of a UAV in an "autonomous" or programmed flight mode, provided that UAV performance and designated ATC communication circuits are continuously monitored by the UAV operating crew, and that the UAV system and crew are capable of immediately taking active control of the UAV.

Fig. 28 - Section 5.2.2 of the CASA Advisory Circular (CASA 2002)

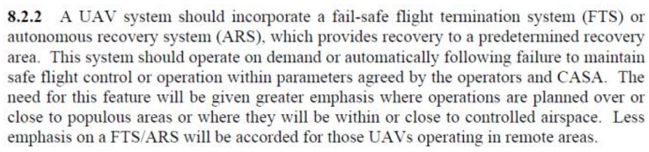

8.2.2 A UAV system should incorporate a fail-safe flight termination system (FTS) or autonomous recovery system (ARS), which provides recovery to a predetermined recovery area. This system should operate on demand or automatically following failure to maintain safe flight control or operation within parameters agreed by the operators and CASA. The need for this feature will be given greater emphasis where operations are planned over or close to populous areas or where they will be within or close to controlled airspace. Less emphasis on a FTS/ARS will be accorded for those UAVs operating in remote areas.

Fig. 29 - Section 8.2.2 of the CASA Advisory Circular (CASA 2002)

F. *Desired PWM Signals*

Every time changes were made to the Hexacopter wiring, the peak time of the PWM signals that had to be sent to the Flightboard to achieve a specific signal would change slightly. This meant that we then had to measure the PWM values produced by the transmitter again and determine what the new relationship would be between motor output and PWM peak time, then re-enter this data into our programs.

The measurements listed below are for the Aileron, Elevator, and Rudder channels, which were the three we were using to automate the movement. The peak times were measured with an oscilloscope while the motor outputs were measured with the NAZA-M assistant software as a percentage. The following data was recorded on 17 October 2014 and at the time this report was published, was the data stored on the Hexacopter for its internal calculations.

TABLE II
ELEVATOR PWM VALUES, 17 OCTOBER 2014

PWM Peak Time (µs)	Motor Output (percentage)
1200	79.8
1300	55.2
1400	31.0
1500	7.3
1600	-14.5
1700	-39.1
1800	-62.6
1900	-88.2

TABLE III
AILERON PWM VALUES, 17 OCTOBER 2014

PWM Peak Time (µs)	Motor Output (percentage)
1200	-78.0
1300	-54.3
1400	-29.5
1500	-5.5
1600	20.3
1700	43.6
1800	67.4
1900	90.8

TABLE IIV
RUDDER PWM VALUES, 17 OCTOBER 2014

PWM Peak Time (µs)	Motor Output (percentage)
1200	-76.7
1300	-51.1
1400	-30.3
1500	-5.1
1600	19.7
1700	42.6
1800	67.3
1900	91.3

G. *Wiring Schematic*

The diagram shown below is a more detailed version of Fig. 7, found in section II.B.4), which shows all of the connections between the various components, along with a photo displaying those connections in real life. The ports listed for the Raspberry Pi follow the standard naming configuration as listed by the manufacturer [13].

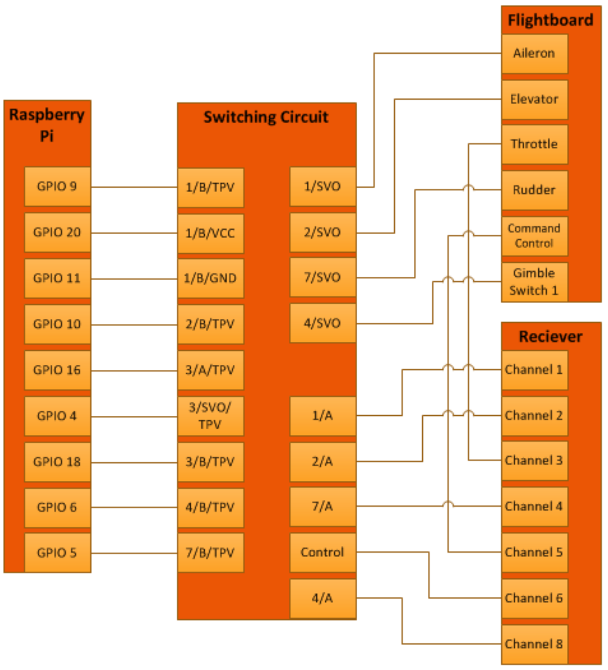


Fig. 30 - More detailed Schematic highlighting the various connections in the Switching Circuit

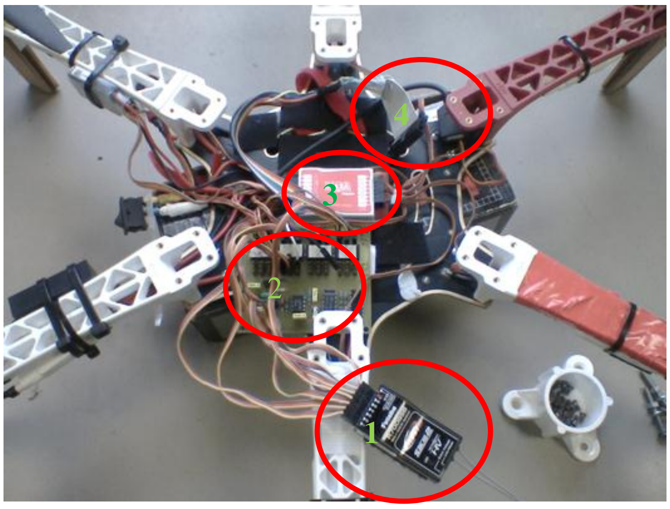


Fig. 31 - The actual wiring of the Switching Circuit inside the Hexacopter. Key: 1 – Reciever, 2 – Switching Circuit, 3 – Flightboard, 4 – Ribbon cable connecting to Raspberry Pi

ACKNOWLEDGEMENT

I would like to thank both of my supervisors, Professor Dr. Thomas Bräunl and Chris Croft for their support and guidance throughout this project. Also, I would like to acknowledge the students Michael Baxter, Alexander Mazur and Merrick Cloete that I worked with for their input and effort, especially Michael Baxter for his contributions in writing and adapting several of the Base libraries for the Hexacopter that my own programs had to call. Special mention must also go to my fellow year students Garrick Paskos and Ruvan Muthu-Krishna for their assistance throughout the year and Brian from the Perth RC Hobby Store, for lending us equipment for demonstrations at the UWA 2014 Open Day.

REFERENCES

[1] Venables, C., 2013. *Multirotor Umanned Aerial Vehicle Autonomous Operation in an Industrial Environment using On-board Image Processing*. Engineering Final Year Thesis. University of Western Australia.

[2] O'Connor, R., 2013. *Developing a Multicopter UAV Platform to Carry Out Research into Autonomous Behaviours, using On-board Image Processing Techniques*. Engineering Final Year Thesis. University of Western Australia.

[3] Drage T.H. 2013. *Development of a Navigation Control System for an Autonomous Formula SAE-Electric Race Car*. Engineering Final Year Thesis. University of Western Australia.

[4] Mahoney, M.J., *Aircraft_Attitude2* n.d. [image online] Available At: http://mtp.mjmahoney.net/www/notes/pointing/pointing.html [Accessed 6 October 2014]

[5] WDSOT (Washington State Department of Transportation), 1997, *Chapter 11: Visual Search Patterns*. [pdf] Washington: State Department of Transportation. Available at: http://www.wsdot.wa.gov/NR/rdonlyres/505EB17D-DE17-4FF0-A132-55282890DB84/0/WSDOTAircrewTrainingTextChpts1114.pdf [Accessed 12 May 2014]

[6] Futuba, 2014 *14SG: 14-Channel, 2.4GHz Computer Radio System*. [online] Available at: http://www.futaba-rc.com/systems/futk9410-14sg/ [Accessed 4 May 2014]

[7] DJI, 2014. *DJI: The Future of Possible*. [online] Available at: http://www.dji.com/ [Accessed on 15 August 2014]

[8] Scrosati B., Garche J., 2009, Lithium batteries: Status, prospects and future, *Journal of Power Sources*. 95(2010), pp 2419–2430

[9] Suzuki K.A.O., Filho P.K., Morrison J.R., 2012. Automatic Battery Replacement System for UAVs: Analysis and Design. *Journal of Intelligent & Robotic Systems*, [journal] 65(1), pp 563-586. Available through: University of Western Australia Library website <http://download.springer.com.ezproxy.library.uwa.edu.au/static/pdf/779/art%253A10.1007%252Fs10846-011-9616-y.pdf?auth66=1413863399_878278e0f3207f36d719fe5996bcd554&ext=.pdf> [Accessed 13 August 2014]

[10] ABC, 2014. *CASA Plans Legal Action over Drone Crash in Gerladton* [online] Available at: http://www.abc.net.au/news/2014-06-25/casa-plans-legal-action-over-drone-crash-in-geraldton/5550764 [Accessed 7 October 2014]

[11] Civil Aviation Safety Authority. 1998. *Civil Aviation Safety Regulations 1998, Statutory Rules No. 237, 1998 as amended, made under the Civil Aviation Act 1988. In Volume 3: rr. 99.005–137.300*, Office of Parliamentary Counsel, Canberra, 1998. Available From: http://www.comlaw.gov.au/Details/F2014C00612/Download, Current as of 1 May 2014.

[12] Element14, 2014. *Raspberry Pi Model B+*. [online] Available at: http://www.element14.com/community/community/raspberry-pi/raspberry-pi-bplus/blog [Accessed 6 October 2014]

[13] Raspberry Pi Foundation, 2014, *Downloads*.[online] Available at: http://www.raspberrypi.org/downloads/ [Accessed 13 March 2014]

[14] Drogon, 2014. *wiringPi: GPIO Interface for the Raspberry Pi*. [online] Available at: http://wiringpi.com/ [Accessed 23 May 2014]

[15] Hirst, R., 2013 *PiBits/servoBlaster* [online] Available at: https://github.com/richardghirst/PiBits/tree/master/ServoBlaster [Accessed 23 May 2014]

[16] DePriest, D. *NMEA Data* [online] Available at: http://www.gpsinformation.org/dale/nmea.htm [Accessed 4 May 2014]

[17] Had2Know, 2014, *HSV Colour*. [online] Available at: http://www.had2know.com/technology/hsv-rgb-conversion-formula-calculator.html [Accessed 9 October 2014]

[18] Göktoğan A.H., Netttleton E., Ridley M., Sukkarieh S., 2003, Real Time Multi-UAV Simulator, In: IEEE *Proceedings of the 2003 International Conference on Robotics and Automation,* Taipei, Taiwan, 14-19 September 2003, IEEE

[19] Myer J., Sendobry A., Kohlbrecher S., Kilngauf U., Stryk O. von, 2012, Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo, In: *Simulation, Modeling, and Programming for Autonomous Robots*, Tsukuba, Japan, 5-8 November 2012, SIMPAR

[20] Santamaria E., Pastor E., Barrado C., Prats X., Royo P., Perez M., 2012, Flight Plan Specification and Management for Unmanned Aircraft Systems. *Journal of Intelligent &Robotic Systems*, [Peer Reviewed Journal] 67 (2). Available through: University of Western Australia Library website http://download.springer.com/static/pdf/554/art%253A10.1007%252Fs10846-011-9648-3.pdf?auth66=1413611530_4ffeaa5a24f5c253bcff39f6ef7f782b&ext=.pdf [Accessed 15 May 2014]

[21] Pitre R.R. Li X.R., 2009, A New Performance Metric for Search and Track Missions, 2: Design and Application to UAV Search. *12th Inrenational Conference on Information Fusion*, Seatle, USA, 6-9 July 2009

[22] Husby, C.R., 2005. *Path Generation Tactics for a UAV Following a Moving Target*. Master of Science in Aeronautics and Astronautics Thesis. University of Washington.

[23] Gray A.J., Jenkins D., Andrews M.H., Taaffe D.R. & Glover M.J. 2010. *Validity and reliability of GPS for measuring distance travelled in field-based team sports*, Journal of Sports Sciences, 28:12, 1319-1325, DOI: 10.1080/02640414.2010.504783

[24] Palmer, D., 2006. Advanced motion control and GPS guide car steering robot, *Eureka* [e-journal] (26/2), Available through: University of Western Australia website http://search.proquest.com/docview/219365256 [Accessed 20 May 2014]

[25] Zhang, J., 2010, Vehicle Dynamics Control Based on Low Cost GPS, In: IEEE *International Conference on Information and Automation,* Harbin, China, 20-23 June 2010, IEEE

[26] Woolan, H., 2004, *Incorporating Heuristically Generated Search Patterns in Search and Rescue*. [online] Edinburgh, Scotland: University of Edinburgh. Available at: http://www.aiai.ed.ac.uk/project/ix/documents/2004/2004-msc-wollan-sar-patterns.pdf [Accessed 11 May 2014]

[27] Xiaowei F., Feng H., Xiaoguang G., 2012. UAV Mobile Ground Target Pursuit Algorithm, *Journal of Intelligent & Robotic Systems,* [journal] 68(3). Available through: University of Western Australia Library website http://download.springer.com/static/pdf/170/art%253A10.1007%252Fs10846-012-9690-9.pdf?auth66=1413618402_17b4b1486468b77649822578faf4cef1&ext=.pdf [Accessed 16 May 2014]

[28] Civil Aviation Safety Authority. 2002. *Advisory Circular AC 101-1(0) July 2002, Unmanned Aircraft and Rockets, Unmanned Aerial Vehicle (UAV) Operations, Design Specification, Maintenance and Training of Human Resources*. Available From: http://www.casa.gov.au/wcmswr/_assets/main/rules/1998casr/101/101c01.pdf