NIWA High Altitude Payload Recovery: Final Report

ENEL400

Contributors:

Thomas Shortt, 62947134

Brodie Greenfield, 13814427

Callum White, 96979533

Sam Stephenson, 75817294

Supervisor: **Maan Alkaisi**

NIWA: **Richard Querel**

With special thanks to:

Kelvin Barnsdale and Adriel Kind

University of Canterbury

18 October 2015

Contents

Executive Summary

On a weekly basis the National Institute of Water and Atmospheric Research, NIWA, launches weather balloons for the purpose of atmospheric research. Currently the on-board payload electronics, used to collect data, are left to fall to the ground when the balloon bursts at approximately thirty kilometres in altitude. This results in the payload being damaged or lost. Ultimately, NIWA wants to avoid the loss of the on-board payloads, which cost approximately NZ$800 per launch.

NIWA approached the University of Canterbury in order to have a solution developed. It was specified that a self-recovering payload solution be developed which could operate at altitudes up to thirty kilometres. NIWA also required the payloads to be returned in a reusable state. The team involved also decided that the solution must be available using off the shelf components.

The solution designed by the team is a fixed wing retrieval aircraft. In order to come to the final solution, the team performed feasibility studies and statistical modelling; decision making for both hardware and software; hardware and software calibrations; controller tuning; and algorithm design for return to launch operations. The team also investigated legal issues surrounding high altitude unmanned aerial vehicle flight.

Initial research into the problem determined that high wind speeds when dropping from thirty kilometres would be the main challenge to overcome. The airframe selected to perform recovery is known as the Mini Talon. It was selected due to its reduced drag from having a v-tail, and also the increase stability over flying wing designs, which comes from having a tail pane. The system is controlled via a Pixhawk flight controller running ArduPlane software. The pitch, roll and yaw controllers were tuned manually, following a Ziegler-Nichols tuning approach. When testing the final solution at low altitudes, within legal line of sight limits, it was found that the control system was able to successfully correct unstable modes of flight such as tumbling and stalling.

The final system was not tested at altitudes outside of line of sight due to legal restrictions. However, from modelling and testing, it is believed that the Mini Talon recovery solution will be able to recover a worst case estimate of 28% of NIWA's launched electronics.

In the future, it is recommended that high altitude testing be performed to determine the actual success of the solution.

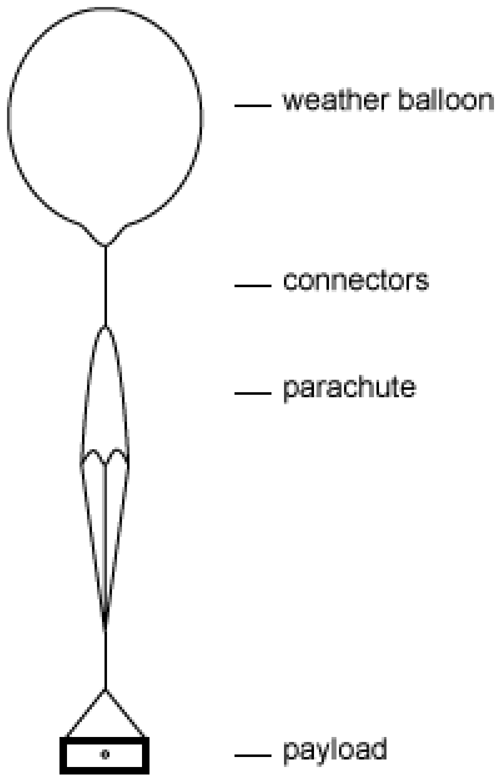
0. Introduction

0.1 The Problem

On a weekly basis, the National Institute of Water and Atmospheric Research, NIWA, launches weather balloons for atmospheric research purposes. NIWA collects a variety of data, such as ozone concentration, wind speed, altitude and temperature. In order to acquire these data samples, electronic payloads known as radiosondes and ozonesondes are attached to the weather balloons. These electronic payloads cost approximately NZ$800 per launch. The balloons ascend at a rate of five meters per second, and reach an altitude of approximately thirty kilometres (above sea level) before the gas inside expands to the point where the balloons burst. This results in a typical ascent time of one hour and forty minutes.

Currently when the balloons burst, the on-board electronics are left to fall back to the Earth's surface, where they are either damaged or lost. The payloads are fitted with parachutes, however, these often fail to deploy, or in the case where they do deploy, are carried even further away from the launch location by the wind. Figure 1 shows a diagrammatic representation of the current recovery solution. In an attempt to recover payloads, the packages carry a reward notice of NZ$60 for anyone who finds, and returns NIWA's electronics. The current method results in 20% of launched payloads being returned.

Figure 1. Representation of the current recovery solution.



NIWA has requested that a solution be developed to return the electronic payloads to the launch location at Lauder, Otago. NIWA have stated that if a feasible solution existed, they would be able to launch more expensive payload electronics without the risk of them being damaged or lost.

0.2 Previous Research and Development

Others have attempted to develop solutions, for NIWA in the past, however, these have not been able to operate at high altitudes (up to thirty kilometres) and have required NIWA to commit with large start-up costs, in the order of NZ$1m. For example, a fixed wing recovery solution, the GPS Boomerang DataBird, was developed by Synco Reynders [22], however, this was not pursued by NIWA. Despite not being an ideal solution for NIWA, the GPS Boomerang is a platform which can be used for carrying or recovering payloads for atmospheric research.

Internationally, self-recovering payload research has been performed by universities such as Oregon State University [23]. Oregon State University implemented a controllable parachute design which was tested at altitudes up to twenty-five kilometres. From both Synco Reynders' and Oregon State University's research, it was identified that one of the major challenges was achieving penetration into the wind. This leaves the problem of developing a solution which is able to operate at high altitude, while solving issues surrounding wind penetration.

0.3 Aims and Design Specifications

The team involved with this project was approached by NIWA to develop a self-recovering payload which could operate at up to thirty kilometres in altitude. NIWA specified that an ideal solution would be able to return payloads to within a ten kilometre radius of Lauder, Otago - the launch base of interest. NIWA stated that the electronics must be in a reusable state upon recovery. Despite being interested in the recovery of payloads launched from Lauder, NIWA stated that the solution would be used at other launch sites around New Zealand.

In addition to NIWA's requirements, the team decided that the solution should also be accessible using only off-the-shelf components, and have a total materials cost of less than NZ$1000. To start the project, the team was given a budget of NZ$500 from the Department of Electrical and Electronics Engineering at the University of Canterbury. In addition to this, NIWA indicated that they would provide additional funds as necessary.

This report discusses process undertaken to design a self-recovering solution which could operate at up to thirty kilometres. The following sections discuss the choice of recovery solution, airframe choice, _ and legal matters by Brodie Greenfield; feasibility and statistical modelling by Sam Stephenson; system hardware and software choices by Thomas Shortt; and controller tuning and return to launch algorithm design by Callum White.

Chapter 1. Airframe Selection

Brodie Greenfield

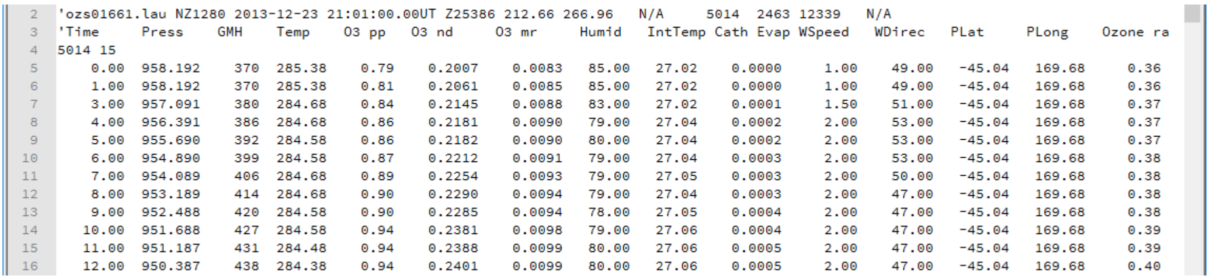
1.1 Theory of operation

This section describes the theory behind the techniques used to identify, then select an appropriate airframe for the task. First, some initial modelling to determine the problem that was being faced. Following that, aerodynamic theory behind three different types of descent control. This then follows on to high altitude operations, and the uniqueness of the environment that is at 30km.

1.1.2 Initial modelling

In order to get an idea of the size of the problem faced, close examination of a large dataset of previous NIWA launches was undertaken. This will be taken much further in the next chapter, however for the purposes of choosing an airframe, a simple plot of all burst points was sufficient.

All balloons from the last 30 years have been tracked using Global Positioning System (GPS). This data was then sent to NIWA and logged into a file. Figure 1 shows an example that a python script could interpret.



```
'ozs01661.lau NZ1280 2013-12-23 21:01:00.00UT Z25386 212.66 266.96    N/A      5014   2463 12339    N/A
'Time       Press     GMH     Temp     O3 pp    O3 nd       O3 mr       Humid    IntTemp Cath Evap WSpeed    WDirec     PLat      PLong     Ozone ra
5014 15
     0.00  958.192     370  285.38     0.79    0.2007      0.0083     85.00     27.02      0.0000       1.00      49.00    -45.04    169.68      0.36
     1.00  958.192     370  285.38     0.81    0.2061      0.0085     85.00     27.02      0.0000       1.00      49.00    -45.04    169.68      0.36
     3.00  957.091     380  284.68     0.84    0.2145      0.0088     83.00     27.02      0.0001       1.50      51.00    -45.04    169.68      0.37
     4.00  956.391     386  284.68     0.86    0.2181      0.0090     79.00     27.04      0.0002       2.00      53.00    -45.04    169.68      0.37
     5.00  955.690     392  284.58     0.86    0.2182      0.0090     80.00     27.04      0.0002       2.00      53.00    -45.04    169.68      0.37
     6.00  954.890     399  284.58     0.87    0.2212      0.0091     79.00     27.04      0.0003       2.00      53.00    -45.04    169.68      0.38
     7.00  954.089     406  284.68     0.89    0.2254      0.0093     79.00     27.05      0.0003       2.00      50.00    -45.04    169.68      0.38
     8.00  953.189     414  284.68     0.90    0.2290      0.0094     79.00     27.04      0.0003       2.00      47.00    -45.04    169.68      0.38
     9.00  952.488     420  284.58     0.90    0.2285      0.0094     78.00     27.05      0.0004       2.00      47.00    -45.04    169.68      0.38
    10.00  951.688     427  284.58     0.94    0.2381      0.0098     79.00     27.06      0.0004       2.00      47.00    -45.04    169.68      0.39
    11.00  951.187     431  284.48     0.94    0.2388      0.0099     80.00     27.06      0.0005       2.00      47.00    -45.04    169.68      0.39
    12.00  950.387     438  284.38     0.94    0.2401      0.0099     80.00     27.06      0.0005       2.00      47.00    -45.04    169.68      0.40
```

Figure 1 – an example log file from one of NIWAs launches.

I wrote a simple python script to analyse these log files, and provide the GPS co-ordinate of the last point on the datasheet. The data stops logging after the balloon bursts, so the last point on the log file is an acceptable way to retrieve the burst point.

The main theory behind this program is the haversine formula. This formula is used to calculate the distance between two GPS points [1]. Equation 1 shows the haversine formula.

$$d = 2r \arcsin\left(\sqrt{sin^2\left(\frac{\emptyset_1 - \emptyset_2}{2}\right) + cos(\emptyset_1)\, cos(\emptyset_2)\, sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Equation 1

Where:

- r = the radius of the earth in km; 6378.1km
- ϕ_1 & ϕ_2 are the latitudes of point 1 and point 2 respectively
- λ_1 & λ_2 are the longitudes of point 1 and point 1 respectively

While this model is slightly inaccurate due to the inconsistencies of the radius of the earth, for the distances that are being travelled being close in relation to the initial point, as well as very high accuracy results being unnecessary, the effects of the changing radius of the earth can be ignored.

A good target for the solution when this model was created was 50km. This is why the .kml output file was pre-programmed with four potential landing cones spaced around the Central Otago / Southern Canterbury region, as this would cover a large area, and a large number of the previous launches would have been caught by this range.

1.1.3 Aerodynamic Theory

1.1.3.1 Balloons

The first of three ideas for controlling the descent of the payload was using a balloon method. This method would allow for the packages to be safely returned to earth, instead of simply crashing back and either being damaged or destroyed.

As there is no propulsion on a balloon lowered package, it would be entirely at the mercy of the wind. Figure 2, shown below, shows the predicted path of a balloon controlled descent.

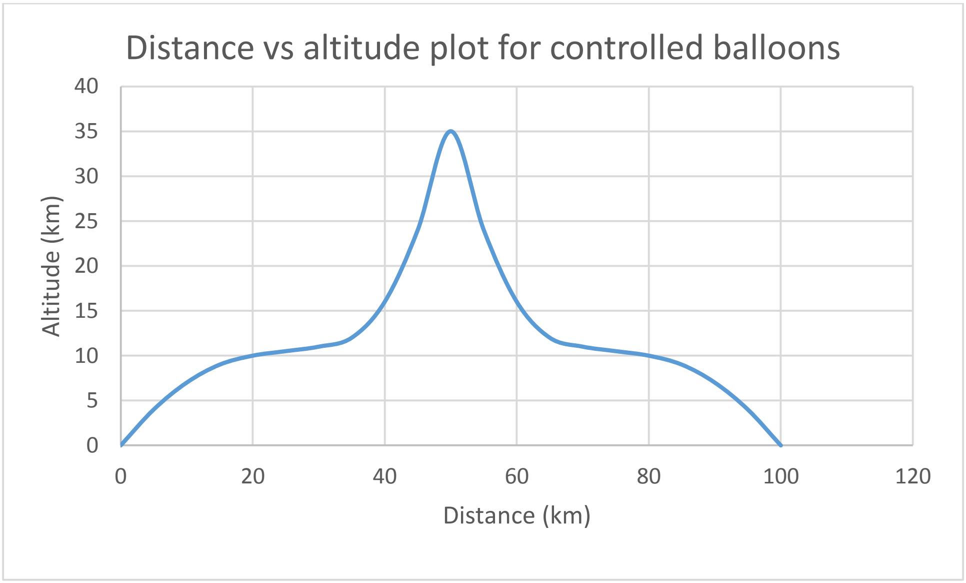


Figure 2 – Distance vs altitude plot for controlled balloons.

As Figure 2 shows, the initial payload is taken in a certain direction by the wind. As it would spend a similar time in the air on the way down, as on the way up, the balloon would experience approximately the same wind profile on the way down, as on the way up. This means that the balloon would be carried further away from the launch site, and would quite likely end up in the ocean.

The scale in Figure 2 is fairly arbitrary. The dependence between the final distance travelled, and the distance travelled at the bursting point is double the bursting point distance. With the average distance that the balloons have been travelling from NIWAs dataset calculated with the python script mentioned above being 84km, this translates to an average distance travelled away from Lauder being 168km.

1.1.3.2 Parachutes

The next idea was to use a series or system of parachutes to control and direct the descent. The theory is that the parachutes are controlled by two strings, the length of which are controlled from within the payload. This was not the conventional circular parachute that the current packages are fitted with. The parachute investigated was more like that of a skydiver; one that can be controlled. Figure 3 below shows an example parachute.

In order to control the direction, there would need to be a GPS and a compass on board. These would determine the altitude and the direction that the package was heading, and where it needed to go.

Figure 3 – An example parachute.

For the control of the parachute lines themselves, there were two different ideas. The first was to use a pair of servos to individually control the length of the lines. The other option was to use a pair of stepper motors to control the length.

Servos are not particularly useful in this application, as they have a limited range, and if the range was to be increased using a gearing system the power, and therefore size requirements would increase beyond the range that was desirable.

This left the stepper motor as the system to control the length of the parachute lines. There would need to be a controller to keep track of the number of forwards and backwards turns that the motors had turned through, as this would be the length of the lines that would be fed to the flight controller to determine the next control inputs.

The next issue surrounding parachutes is that of their capability to penetrate into the wind, and get back to the starting point. While this would be an acceptable solution on a clear, calm day, the winds above Lauder, Central Otago reach 150m/s in the Jetstream. There are also wind gusts, and other non-laminar winds, which would very effectively cause the parachute to collapse on itself, and the payload would lose height so that it would be unrecoverable.

1.1.3.3 Fixed Wing Aircraft

For a fixed wing aircraft there are several differences when compared with the other two methods of recovery identified above. The first is the wind. When it comes to fixed wing aircraft, the actual windspeed does not matter, it will just have an impact on the groundspeed, not the operation of the aircraft itself.

The aircraft doesn't see wind, it only sees air, and its velocity into the air. In terms of the lift that the aircraft generates, it is all about the wind moving over the wings, rather than the aircrafts velocity relative to the ground.

A fixed wing aircraft also performs better in wind gusts. Even a side gust, which is less likely in the Jetstream, will only cause the aircraft to tumble. A properly tuned control system will prevent the aircraft from remaining in an out of control mode of flight.

When it comes to penetration into the wind, the fixed wing aircraft has the highest speed of the three options detailed here. This means that it will be best able to penetrate into the high winds that are faced in the Jetstream.

Weight is an important factor when working with any type of fixed wing aircraft. However, as the aircraft here operates from a starting altitude of 30km, weight is not an issue. As Equation 1 shows below, potential energy is related to mass. As such, as mass increases, so does the potential energy, so the actual distance that the aircraft can fly is not affected by the increase in mass.

$$E_{P_{Grav}} = mg\Delta h$$

Equation 1

As Equation 2 shows below, gravity is as good as constant. [2]

$$\frac{g_1}{g_2} = \left(\frac{r_2}{r_1}\right)^2$$

Equation 2

Doing the math with $r_1 = 6371$, $r_2 = 6401$, and $g_1 = 9.81$, gives a value for g_2 to be 9.72 m/s^2. Also, the height is constant, so the potential energy is directly proportional to the mass of the aircraft. This is helpful, as the aircraft can be loaded to the maximum weight possible to ensure the best possible penetration into the wind.

Having extra weight also helps with stability, as it is harder to throw the aircraft off balance if it weighs more, as there will be more inertia to overcome. This also means that the control system had to be capable of controlling the extra weight.

1.1.4 High Altitude Operations

As Figure 4 shows on the next page, there is very little atmosphere at 30km. This means that there is very little for the control system to act with. This means that there needed to be a way to guarantee stability at this point until such a point on the descent that the control system is able to kick in and guide the solution back to the recovery point.

All that a solution had to do was keep the solution from being completely out of control. For the case of the fixed wing aircraft, the tail would need to be kept up, with the nose pointing back towards earth. There would also need to be some type of system to control this, as it would not be necessary for the entire descent.

In order to implement this, a simple form of dragging something behind the aircraft would be sufficient at keeping the aircraft pointing at the earth. Spinning at this point in the drop would be unhelpful, as the control system would be correcting this instead of flying towards its target. However, it is not an uncontrollable situation, and this is an acceptable solution.

There is the possibility of simply dragging the burst balloon behind would work. However, some careful thought, and upon advice from NIWA that the balloons do not tend to burst at any specified altitude, or in any predictable fashion, something else would need to be implemented.

A big issue with stability is where the centre of gravity is. Having the CoG in the wrong place for a fixed wing aircraft, either too far forwards, or too far backwards, will contribute to an unstable solution. Careful placement of the internal electronics is crucial to keeping the moment of inertia, as well as the solution in the most stable position possible.

In order to model the air density with varying altitude, the following equation is used:

$$P_{alt} = P_{MSL}\left(1 + \frac{L_b}{T_b}(h - h_{MSL})\right)^{-\frac{g_0 M}{RL_{MSL}}} \qquad \text{Equation 3}$$

- P_{MSL} = Standard pressure at sea level = 101.35 hPa
- T_{MSL} = Standard temperature at sea level = 293.15 °K
- L_{MSL} = Standard temperature lapse rate = -0.0065°K/m
- h_{MSL} = Height at mean sea level = 0m
- h = Height above sea level (m)
- R = Universal gas constant = 8.31432 $\frac{N.m}{mol.K}$
- g_0 = Gravitational acceleration constant – 9.80665 m/s^2
- M = Molar mass of earths air = 0.0289644 kg/mol

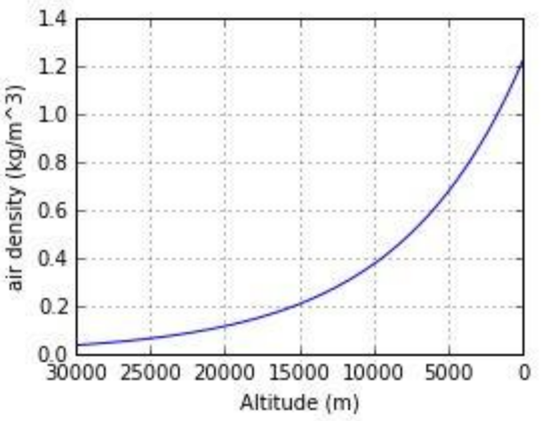


Figure 4 – Air density vs altitude above sea level

Equation 3, and therefore Figure 4 use a model which doesn't take into account temperature as it should. There are some layers in the atmosphere where the temperature is constant [3]. The effects of this however are small, as the change in $L_{MSL}$is a small change (ignoring the divide by 0 issue), and it is only in effect for small parts of the ascent. Most of the ascent does indeed conform to the -0.0065 co-efficient used.

Close examination of Figure 4 shows that it is much shallower close to the x-axis. This shows that the air density increases reasonably slowly throughout the first portion of the descent. As such, there is definitely a case for a stability control system.

A final point is that of potential energy. As Equation 4 shows, there is a certain amount of potential energy associated with the high altitude starting point. Equation 4 shows the amount of potential energy that would be gained by adding a battery to make the plane up to the desired weight, instead of using inert ballast.

$$E_{p_{Chem}} = 3600 * V * Capacity$$

Equation 4

This equation can be combined with Equation 4 to compare the two forms of potential energy that can be consumed by the plane.

1.2 Method

1.2.1 Civil Aviation Authority

Early on in this project it was identified that there would be issues surrounding the legality of any solution that we came up with. These legal issues came about due to the nature of any solution proposed. Instead of just falling through controlled airspace, like the current packages when they return to earth, any solution that we proposed would be considered to be flying through the controlled airspace, which the Civil Aviation Authority (CAA) tend to want to know about.

With this in mind, we brought in Frank Usmar, who is a Senior Technical Specialist from the Special Flight Operations and Recreational Aviation Unit of the CAA New Zealand. He is currently working with the regulations governing the increasing use of UAVs in New Zealand's airspace.

With Franks knowledge of the new part 101 [4] and part 102 [5], we were able to determine the steps that would need to be taken in order to actually get permission from Airways, the governing body that controls the airspace in New Zealand. These steps involved making a case to Airways to acquire permission to use the solution.

One of the main purposes for meeting with Frank was to determine if there was anything special that would need to be built in to the design that would cause us to drastically rethink the design. There was only the possibility to include a transponder on board, which wouldn't have any effect on the design itself, but did impose a restriction on the space that was needed in the eventual solution in order to accommodate all of the required electronics.

1.2.2 Airframe Selection

With any solution that was created there were several criterion that needed to be met. These are, in order of importance:

- Weight
- Penetration
- Payload space
- Stability

In order to make a decision about which type of descent method to use, comparing the proposed solutions, the balloons, parachutes, and fixed wing airframe was necessary. The three types of descent were compared using the above criterion, and the decision was made to use a fixed wing aircraft as the solution moving forwards. This decision was made as it outperformed both the balloon and the parachute method on every measure defined above.

As such, the fixed wing aircraft will be examined in further detail, as there is further decisions to be made before the final solution is determined.

1.2.2.1 Fixed Wing Aircraft

Before the airframe was selected, I had a choice to make with regards to where to source the airframe from. The University has a CNC mill that could be used to create multiple airframes at low cost, not mattering if they were broken during the course of the project.

However, as I'm not a mechanical engineer, and at the time didn't know a lot about aircraft and aerofoil design, and in the interest of time and simplicity, I decided that an off the shelf model would be the best place to source the airframe moving forwards. Off the shelf gave the best performance for the time and money that was available at the time of the aircraft selection, as the off the shelf models had already been designed and tested to ensure the best operation.

The other choice that I made before the aircraft was selected was that of the material that it would be constructed from. As the airframe would be repeatedly landed into often very different landscapes, something that could be repaired easily, and without adding weight would be an advantage. This meant that some type of foam would be the preferable material for the construction of the aircraft.

There are three types of fixed wing aircraft that I identified as possibilities for the final solution. The generic types of these aircraft are:

- A flying wing
- A specialised FPV payload carrying frame
- A standard fuselage airframe

Figure 5 below shows examples of each of the three airframes specified above.

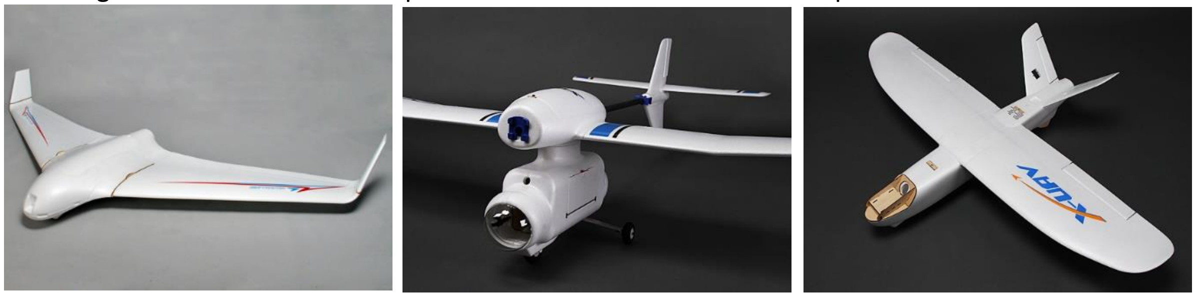


Figure 5 – The flying wing The FPV Payload carrier the standard fuselage

These three airframes were compared using the following four criterion, in order of importance once again:

- Weight
- Penetration
- Payload space
- Stability

In order to appropriately compare these three airframes, the internet was the best choice for data. As these are all off the shelf hobby grade airframes, there is plenty of user accounts as to how the planes react in all types of weather scenarios. These were an excellent place to start, as they provided real world data and feedback on how each airframe performed.

The three models above were all compared according to the criterion above, as well as with not just user reports, but the expertise and knowledge of Associate Professor Keith Alexander, from the University of Canterbury. He provided a wealth of knowledge and expertise from many years in the remote control flying industry.

He also mentioned that the amount of potential energy at 30km should be plenty in order to successfully glide back to the required landing point in Lauder. This will be examined further in the results section.

1.2.3 Testing

In order for there to be sufficient data with which to complete modelling of the problem, significant and thorough testing needed to be completed.

This also needed to occur to test the autopilot and airframe itself. In order to complete the testing, there were several matters that had to be addressed. Most of these surrounded the logistics of the testing. There were several fields that were able to be used, and the actual field used depended on the height restrictions of the individual field. In order to get the required data for the feasibility and modelling section, the plane was flown manually through several manoeuvres while information about the flight was being logged using the on-board logging facilities of the Pixhawk flight controller.

Once the tests were completed, the data was transferred from the Pixhawk back to a computer for analysis.

The battery that was used for the testing was unlikely to be the battery that would be used in the final solution.

For each days testing, there was a list of specific tests that were desired to complete for either general controls testing, or data logging tests. These test plans were created to ensure that all of the desired tests for the day were completed. There was a couple of days where the tests were cut short due to failures in the airframe (it crashed).

1.2.4 High Altitude Operations

As mentioned above, some type of stability assurance was required for the first part of the drop from 30km. This is to ensure stability, or, more accurately, to ensure that the aircraft is in a mode of flight that is recoverable. Catastrophic rolls and spins are undesirable, and would often be unrecoverable.

To prevent this, some type of stability measure would need to be undertaken. To ensure the successful operation of this, a controlled release mechanism such as the one in Figure 6 below would be housed in the tail of the plane. As the string that attaches the balloon would cause havoc for the aircraft during its descent, I have designed the controlled release to be in two stages.

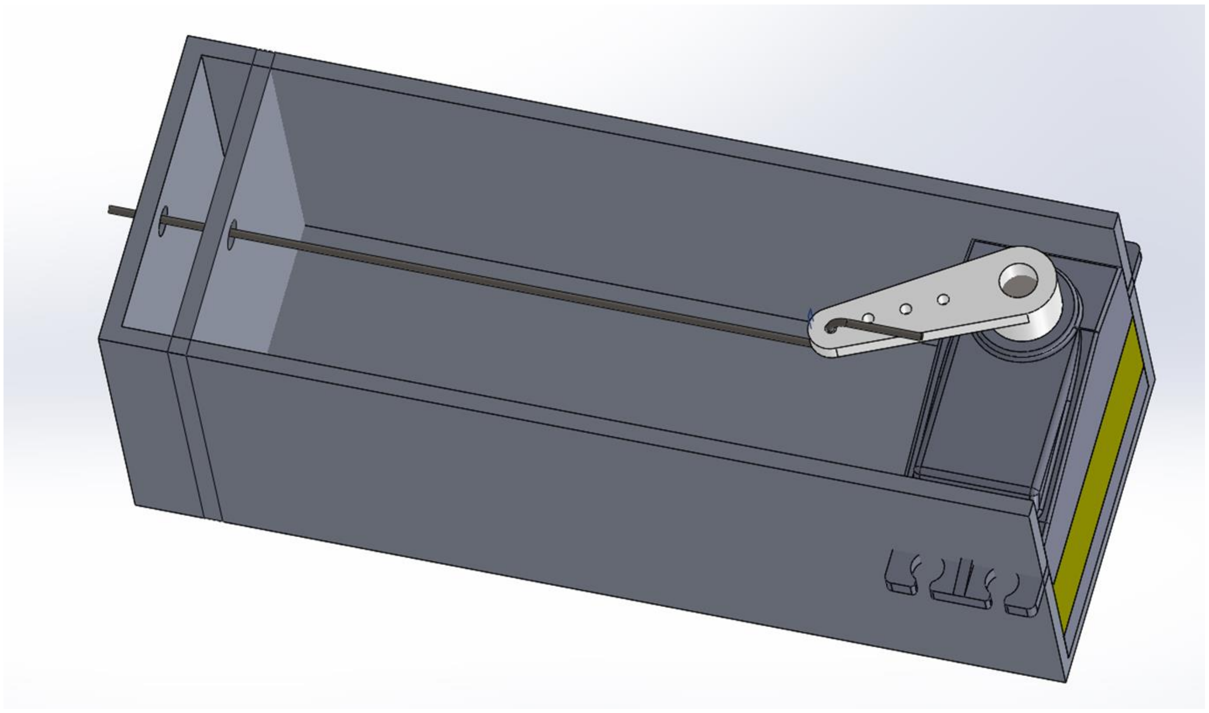


Figure 6 – The controlled release mechanism.

The balloon string would cause problems due to the unpredictable nature of the popping of the balloon. We can see from the dataset mentioned above that the balloons all pop somewhere between 30 and 35km high. Also, as there may be bits of balloon left on the string, consistency would be difficult to ensure.

To circumvent this, the two-stage controlled release mechanism will first jettison the balloon, and simultaneously release a control mechanism. This will stay trailing the aircraft until such a time as the aircraft has gained stability, and the second part of the mechanism will release the control components to fall back to earth. In order to ensure they will not be polluting the environment, they will be constructed of a biodegradable material.

The controlled release is controlled with a servo, and a length of wire. This is to be housed in a lightweight enclosure. In order to perform the releasing of each stage, the servo moves to a known position, and the hook that is holding the aircraft to the balloon will be released. At the secondary altitude, the control components will be released by moving the servo further, and unhooking the control components.

These operations will happen at pre-defined altitudes. The initial drop will happen at 30km, as all functioning balloons pop above this point. In order to ensure that the aircraft does jettison the balloon, the flight controller will also release the balloon if the altitude is determined to be falling, which would indicate that the balloon has popped prematurely. The second part is yet to be determined, but based on the air density theory, the control components should be released at about 15km above sea level.

In order to determine whether or not having power would be a good idea, the potential energy of the aircraft at 30km was compared with the potential energy of a battery, in this case a LiPo battery. In order to determine the battery capacity, the weight limit needed to be identified. As the solution with all of the required gear on board weighed 1.2kg, the weight for a battery could be set at 800g. The biggest battery that would stay under this weight limit was a 10,000mAh 4S battery.

With the equations that were defined in the previous section, the two potential energies could be calculated, and compared to see if it was worth the extra expenditure required to purchase such a large battery.

1.3 Results

1.3.1 Initial Modelling

While the python script I wrote does not take into account all of the data that was provided, it takes enough sample points to provide adequate statistics for the application that I was using it for. More detailed statistical analysis and modelling are provided in the next sections.

The script provided two main outputs. The first was a Google Earth .kml file, which graphically plotted the burst points of all of the balloon launches since 2007. This gave a visual representation of the scale of the problem that we were facing, and gave an idea of the sort of distances that would be required to be covered by our final solution. Figure 7 shows the resulting output.

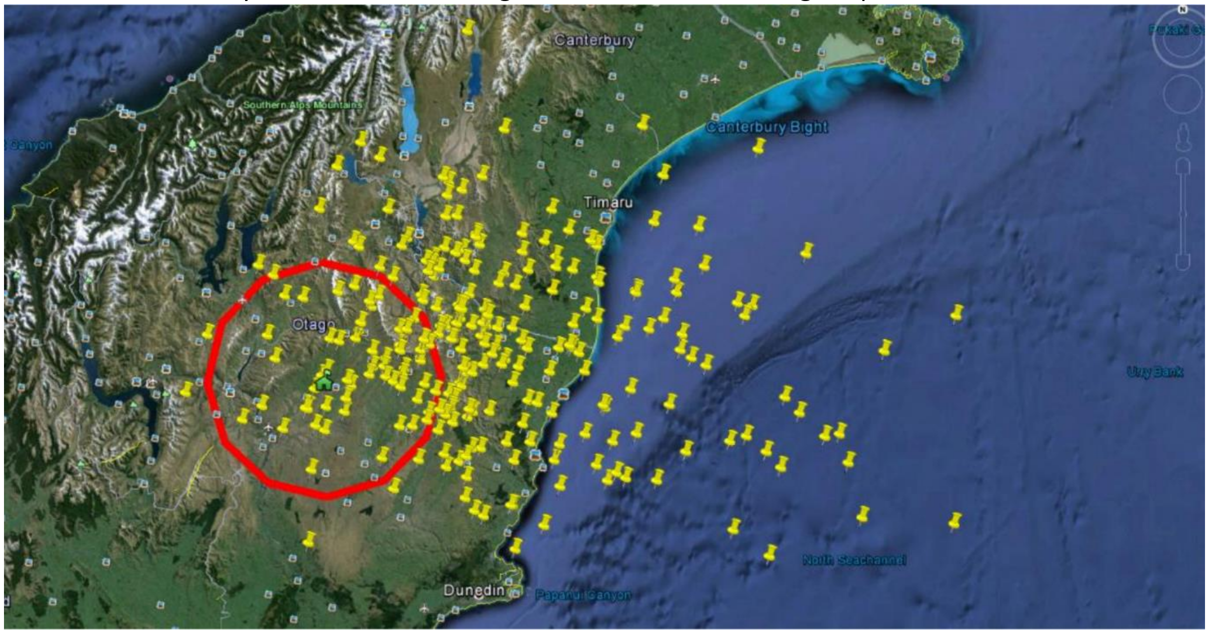


Figure 7 – The initial output for the problem definition.

The other part of the program did a statistical analysis on the data points using the haversine formula for calculating the distance between two GPS points. The program reported that 24% of the balloons burst within the 50km red ring in Figure 7, with the average distance travelled being 86km. There are obviously some that travel an extremely long way from the launch site, and these have no chance of ever being recovered, so any solution that we came up with was not to be launched when conditions indicated that the package would travel this far.

A good proportion (~45%) of the flights burst between 50 and 100km from the launch site, giving us the target distance that our solution would have to make it back in order to recover itself.

1.3.2 Civil Aviation Authority

There are four possible outcomes from a case made to Airways for permission to actually use the solution in controlled airspace.

The current height limit for Unmanned Aerial Vehicles (UAVs) is 400ft above ground. As the solution will be travelling much higher than that, permission needs to be sought from Airways.

The four outcomes are:

- An absolute no
- Only in a restricted corridor
- With a NOTAM
- No change

Obviously, the first of those is an undesired outcome, and to try and prevent this, the best possible case will need to be put forward to Airways by NIWA for permission.

The second option, flying in a restricted corridor is also an undesired outcome. It is also unlikely, as there are commercial airliners that fly in that area, servicing Queenstown, Dunedin, and Invercargill. It is also a lot of effort to go to for a weekly launch.

The third option is a much better option. A NOTAM, or NOtice To AirMen, simply states to all aircraft in the area that there is an unmanned aerial vehicle that will be flying in the area, and it would be a good idea not to hit it.

The final option is no change at all. As NIWA currently has packages of similar size falling through the same airspace with no need to issue a NOTAM or do anything with regards to the airspace that they are falling through, it would be the best outcome from Airways.

One of the biggest factors that might help a case to Airways would be the inclusion of a transponder. These pieces of equipment are very expensive, so NIWA would want to be confident that the solution would be capable of returning it without damage. A transponder signals information about its position and heading to the control towers around the country. This allows Air Traffic Control to know where it is, and provide directions to other aircraft flying in the region.

1.3.3 Airframe Selection and Electronics Fitting

With the two alternate forms of returning the payload to earth, the balloons and the parachutes being rejected in the previous stages, the focus then shifted to the fixed wing aircraft. There are many types to choose from, and so in order to make the decision, I drew up a list of criterion. To re-iterate, these are, in order of importance:

- Weight
- Payload space
- Penetration
- Stability

To choose from the three fixed wing aircraft, I used the following evaluation matrix shown in Table 1 below.

Aircraft	**Weight (0.4)**	**Payload Space (0.3)**	**Penetration (0.2)**	**Stability (0.1)**	**Totals**
Flying Wing	8	5	7	2	**6.3**
FPV frame	2	10	4	2	**4.8**
Standard	10	10	8	7	**9.3**

Table 1 – Evaluation matrix for the type of fixed wing aircraft.

With the airframe selected as the mini-talon, manufactured by x-craft in China, proper testing and other associated parts of the project could get underway. One of these such parts was the fitment of the electronics inside the fuselage.

The main part of the autopilot hardware is the Pixhawk flight controller. About the size of a credit card, but much thicker, it sits in the middle of the fuselage, with servo leads all running to it. Other parts of the electronics associated with the running of the plane are the ESC and the battery. The ESC is kept on the outside of the plane to keep it cool while it is running. The battery, in order to keep the Centre of Gravity in the correct place, in this case on the central spar that runs through the wings.

As there will be more electronics loaded into the rear compartment of the fuselage, the battery will be increasing in size in part to keep the CoG in the correct place. This is detailed further in the next part. In the rear of the fuselage there will be two parts.

The first is the controlled release mechanism from the method section, along with the associated control apparatus. The other part is the actual ozone sensing equipment. This sits horizontally in line with the plane as the plane will be vertical during its ascent.

The radiosonde equipment sits in front of the Pixhawk. After this, there is very little space left inside the fuselage, and the total weight comes in at 2kg.

1.3.4 Testing

With each of the different tests that were performed, the information that was returned increased. To get the dataset as large as possible, to accommodate for many different scenarios, the aircraft was flown in different ways, such as:

- In a straight line
- In a straight line with no power
- Up and down in an oscillatory fashion
- Side to side in an oscillatory fashion

These tests, coupled with the data from the GPS and the airspeed sensor allowed for a simple model of the plane to be created. With the wind very laminar, and tests performed both into and with the wind, although there was very little wind that day provided an excellent form around which to build a model.

The straight line test was the main test for this purpose, as it provided the cleanest data logs to use for further modelling.

The no-power test was to test the glide slope, with this figure, it is one of the factors in determining how far the plane might fly, given a starting altitude, and average wind speed.

The two oscillatory tests were more for testing the aircraft. For the control system on board the flight controller, being able to respond quickly is the best way of gaining control of an uncontrolled flight. Using these tests, the Pixhawk was able to be configured to the best response possible.

1.3.5 Battery

As mentioned before, the inclusion of a battery is a question of potential energy. With the potential energy of the solution being dependent on the height it starts at, and its mass, the question was, would a battery significantly add to this figure.

To start with, the potential energy at 30km. For a solution weighing 2kg, the potential energy is defined as $E_{P_{Grav}} = mg\Delta h$. Running this with g = 9.81 m/s^2, Δh as 30000m, and m as 2kg gives a value for the potential energy from gravity as 588,600J.

This needs to be compared with the potential energy of a battery. Using the heaviest battery that the weight budget will allow, this gives a 4S, 10,000mAh battery. With the equation for energy defined as $E_{p_{Chem}} = 3600 * V * Capacity$, V being 14.8V, the nominal voltage of a 4S battery. This gives a value for the potential chemical energy as 532,800J. Comparing these two values gives an increase in potential energy of 90% on the original gravitational potential energy. This tells me that including the battery and power system is absolutely worth including.

1.3.6 High Altitude Operations

As we know from the provided dataset, the balloons generally only burst above 30km. To keep with simplicity, the flight controller will release the balloon at 30km.

One of the large parts of the high altitude operations is the issue of stability. To circumvent this issue, a streamer will be used to create drag and keep the aircrafts nose pointing towards earth. This will ensure that the only manner of unstable flight will be spinning around the central axis. As there will still be airflow over the control surfaces, the control system will be able to gain control of the aircraft, and begin the flight back towards the launch site.

When the flight controller detects that the altitude has reached a secondary altitude, about 15km, and, more importantly, it has control of the aircraft, it will release the streamer before it starts the flight back to the launch location.

1.4 Discussion

1.4.1 Aims

The main goal of this project was to design and build a solution to safely return a payload of atmospheric sensing equipment back to its launch location.

In order to achieve this, this part of the project started off with some basic statistical analysis, which would be taken further later. This was achieved by analysing the large dataset that NIWA provided of the launches since 1986. For the purposes of the initial statistical analysis however, only the data since 2007 was reviewed. With the scale of the problem defined, the selection of the method by which the problem would be solved could begin.

In order to make the correct decision about which type of airframe to choose, the CAA needed to be contacted to determine whether or not there should be any special requirements that would need to be considered when designing the final solution. Choosing the aircraft type was the next item to be completed. As neither I, nor any of the group had much aeronautical experience, the project supervisor was the go-to person for identifying some of the different solution types.

There was also the testing that needed to be done. As there are various height limits for UAVs in a legal sense, testing was performed at lower altitudes, with results to be interpreted along with the feasibility and modelling data to get an idea of the behaviour at the high altitudes that the solution will be facing.

Finally, there was the matter of high altitude to contend with. Measures needed to be taken to ensure stability for the first part of the descent, as the aircraft needed to be in a controllable state, although not necessarily controlled at such a time that it came into enough atmosphere to begin to fly.

1.4.2 Iterations / Relate Results to Theory

This section re-iterates the ideas and thinking at each stage of the project. It also relates the results and decisions made during the course of the project back to the theories explained in section 1.1.

1.4.2.1 CAA

At the start of the project, it was identified that the CAA would play a part in the final design of the solution. To determine just how much of a part that would be we met with a representative from them to get an idea of just how much the design might have to change.
In the end, the CAA and legal issues were kept in mind, but ultimately determined that it would be NIWAs responsibility to ensure compliance with the rules surrounding UAVs. NIWA was also a good choice, as they already have a good relationship with Airways, so could use that to their advantage when making the case to use our solution.

1.4.2.2 Airframe Selection

The first iteration was the decision of the type of craft that would return the payload to earth. With the three options being the balloons, parachutes, and the fixed wing aircraft, the choice needed to be made as to which one of these three methods were the best for the application at hand. The thinking at this point was that simpler is always better, and that the controlled parachutes would be the way forward, as it was obvious that the balloons would be incapable of making any headway into the wind that had taken it so far away to start with.

Talking with a couple of people with knowledge in the aeronautical field, the thinking that parachutes were simple and easy to use turned out to be not true. Also, when it came to parachutes, there were almost guaranteed to be control issues, not just from the start in low atmospheric conditions, but also in the high winds faced in the Jetstream on descent. These winds would likely hamper the control, and also have a large effect on the penetration, and therefore the overall distance that they would be able to cover.

So, the fixed wing aircraft was moved to as the preferred option moving forwards. The next decision that had to be made was which type of airframe would be best for the task. In order to decide that, I compared three different types of craft against four criterion; weight, penetration, payload space, and stability.

The weight was considered the most important, as there were two forms of restriction, and one big benefit from having a weight of 2kg. The restrictions of course come from the balloons that are used to lift the aircraft; they can only lift a payload of 2kg. The other restriction comes from the CAA, whose tougher rules come into play when the flying weight of the solution rose above 4kg. The benefit of having the solution as heavy as possible comes in the form of penetration. As the theory has shown, energy is proportional to mass for a fixed altitude, so the heavier the object is, the more energy it will have. This becomes useful when it is time to penetrate into the wind during the descent.

The next option to choose from was the ability to penetrate into the wind. As mentioned above, this was partly dependant on the weight of the aircraft itself, however the differences in each of the designs made each aircrafts ability to penetrate different. Again, working from advice from knowledgeable people in the aeronautical area, each of the airframes was compared against the others to determine which was best.

Thirdly was the payload space. Any airframe needed to have a large enough payload space to house all of the required electronics for the atmospheric research, as well as the electronics that were necessary for the automatic retrieval of the aircraft. This was a relatively easy parameter to measure, and erring on the side of caution, a larger volume than strictly necessary was chosen, in order to safeguard against future upgrades to the hardware, and the atmospheric payload.

Finally, the stability. The ideal airframe would be unconditionally stable, however this is a very difficult feat to achieve, especially in the low air densities of 30km. With this in mind, the thinking changed to a more realistic easy to get to an unconditionally stable mode of flight. This made the choice easier as with some simple stability measures, the chosen aircraft could be made unconditionally controllable state. While that state may not be controlled, it is a simple matter for the control system in the autopilot to gain control of the aircraft.

1.4.2.3 High Altitude Operations

As has been explained in previous sections, stability and control at high altitudes is the biggest issue that we faced. If the aircraft is in an uncontrollable state when it reaches the required air density or speed with which to fly, it will behave no differently to the current packages that are launched, i.e. a crash landing will be inevitable.

To ensure that this does not ensue, a control system in the form of a streamer will be deployed to keep the aircraft pointing in the correct orientation. There is some debate about the altitude that the streamer will need to be released. It all comes down to how much drag there is from the streamer, and therefore how much this slows the aircraft down at the top of the altitude scale. Further research into this is necessary to determine when the autopilot should jettison the streamer.

1.4.3 How do Results Achieve Aims

The legalities of the solution have been addressed by talking with the CAA, and determining what there needed to be in order to achieve legality. While this is in the most part up to NIWA to implement, there have been changes to the design to accommodate the possible need to include items such as a transponder. These changes have been mostly physical, with keeping the weight down below the CAAs limit of 4kg to prevent more serious rules applying, as well as selecting a large enough airframe to house any potential extra electronics.

When it came to choosing an airframe, the mini-talon that was chosen as it outperformed the other two options in all of the criterion that were determined that the solution must have. Throughout the testing process, the mini-talon has been ratified as the right choice for the trials and tribulations that it will face during the descent and associated flight back to the launch location.

The main part of the ratification process was the testing. Testing the aircraft through many different manoeuvres, in many different weather conditions was the ideal test for the aircraft itself. The testing process also covered stability, as well as data logging for the feasibility and modelling section.

In terms of the high altitude stability options, there were precious few. The streamer idea, while a simple one, was the only concept we came up with that had the ability to prevent the aircraft ending up in an uncontrollable state. As such, it is the idea that has been implemented.

Finally, the battery. When I was informed that the potential energy of the aircraft at 30km would exceed the potential energy of a battery, I was dubious. Running the numbers with simple equations revealed that I was correct in thinking that that quote didn’t hold up, with the potential energy of the battery adding a further 90% to the potential energy due to the starting height. With heavier planes that have less battery capacity per flying kilogram this might be correct, but for our aircraft, having the battery on board adds significant range.

Chapter 2. Feasibility and statistical modelling

Sam Stephenson

At the commencement of the project, very little was available in the form of a clear image of the conditions in which the craft might operate. Many questions had emerged with regard to what kind of airframe might be suitable, whether carrying a motor would be beneficial, and most fundamentally what proportion of NIWA payloads could be recovered.

New Zealand's location and lack of surrounding land-mass is such that its atmospheric wind conditions are persistently harsh. This is an issue that has ultimately led to other attempts at producing a self-recovering payload being inoperable for NIWA. An analysis of these conditions, and how they affect NIWA's high altitude balloons allows us to predict the range of conditions in which our solution must operate.

An analysis of the expected operating conditions would useful insight into what strategies a successful craft might employ to optimize performance. The development of a flight model to reflect the expected behaviour would also be useful to determine the expected benefit the solution would provide. With a flight model and an understanding of how our chosen airframe behaves aerodynamically at low altitudes, we were able to extrapolate its behaviour at higher altitudes, and as result estimate how far it will be able to travel from a 30km release point. Estimates were produced for both for scenarios when the airframe was only gliding as well as when it was carrying a motor to utilise at lower altitudes.

Given that NIWA's ideal outcome is a solution that will return to within a 10km radius of the point of launch in Lauder, Otago, it can be reasonably assumed that the airframe will be traveling undoubtedly into the wind for the duration of the return journey because it will by flying in the opposite direction to the balloons drift. This dictates that a model used to predict the solutions range would have to both consider what the ideal travel distance of the airframe might be under perfect conditions without wind, and to what degree an oncoming wind would curb this range.

In following the following chapter, the selection process of a suitable flight model is discussed as well as how it operates. In the methods section, presented first is an analysis of New Zealand wind conditions, and how they affect balloon flight. Second is the method used to identify the parameters required to model the airframe, how the airframe is expected to behave at higher altitudes with an oncoming wind, and how flight performance can be improved with the use of a motor.

2.1 Theory of Operation

Flight models themselves are very well understood, and packages such as XPLANE are often used to simulate flight for aerial vehicle research [6] [7]. The parameter set required, however, to develop an accurate flight model for this package is extensive. Using a standard built-in airframe model simply would not have been representative of the flight characteristics of an extremely small airframe such as the mini-Talon and unfortunately parameter set's for small airframes are not readily available. As result a model would have to be developed using parameters that could be directly measured from the recorded flight characteristics of the craft.

Wind tunnel experiments were also considered to measure flight parameters, but would have occupied too great a portion of the project timeframe. Instead, the Pixhawk flight controller provided extensive logging capability of inertial measurement unit, GPS, magnetometer and pitot-static airspeed measurements, which initially thought to be adequate to identify the necessary flight parameters. It was later found that this route required assumptions to be made that would reduce the accuracy of the model.

In the interest of limiting the number of parameters to be identified, the flight model used should aim to be as simple as possible. As this analysis aimed to solely estimate the expected travel distance of the craft, it was only necessary to model flight in two dimensions, so the sideslip behaviour of the characteristics could be ignored. In addition, as the flight from thirty kilometres will take a time in the order of half an hour to complete, a large portion of flight can be reasonably expected to occur at steady state, which makes it reasonable to also ignore the crafts inertial qualities.

This reduces the modelled set of forces to two aerodynamic forces, (lift and drag), and gravity. Aerodynamic forces are be defined as below (equation 1), where ρ is density of air at the given altitude (kgM^{-3}), v is the magnitude of the airspeed of the craft (ms^{-1}), A is the area exposed (m^2) and C_{aero} is an aerodynamic coefficient, which can be determined experimentally. The force of gravity is defined, for the sake of consistency (equation 2), where m is the mass of the airframe (kg), and g is the gravitational constant.

$$F_{aero} = \frac{1}{2}\rho v^2 A C_{aero} \tag{1}$$

$$F_{grav} = mg \tag{2}$$

As such, the parameter set required to model the three aforementioned forces has been reduced to three values: the coefficient of lift (C_{lift}), the coefficient of drag (C_{drag}) and the mass of the plane (m). In the context of an airframe, the three forces interact as shown below in figure 8.

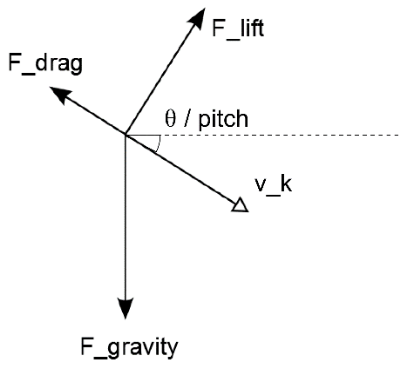


Figure 8: The flight model used

Using two dimensional vector space, the above system can be represented as shown below in equations 3, 4 and 5. Where R is the rotation matrix describing the aircrafts angle of incidence($R = \begin{bmatrix} \cos\theta & -sin\theta \\ sin\,\theta & \cos\theta \end{bmatrix}$, where θ is angle of incidence.)

$$F_{aero} = \frac{1}{2}\rho v^2 \begin{bmatrix} C_{drag} \\ C_{lift} \end{bmatrix} \tag{3}$$

$$F_{grav} = R \begin{bmatrix} 0 \\ -mg \end{bmatrix} \tag{4}$$

$$F_{aero} + F_{grav} = 0 \quad * (ma = 0) \tag{5}$$

The above system can be represented as shown in equations 6 and 7, to analytically define the descent of an airframe at steady state. These can be rearranged to define the glide slope, and velocity of the airframe.

$$mg\ cos\theta = \frac{1}{2}\rho v^2 C_{drag} \tag{6}$$

$$mg\ sin\theta = \frac{1}{2}\rho v^2 C_{lift} \tag{7}$$

$$\theta = \ tan^{-1}\left(\frac{C_{lift}}{C_{drag}}\right) \tag{8}$$

$$v = \ \sqrt{\frac{1}{\rho} \times \frac{2mg}{(C_{lift} cos\theta + C_{drag}\, sin\,\theta)}} \tag{9}$$

Steady state descent slope, θ is defined in terms of the aerodynamic coefficients of the craft. Steady state velocity, v is defined as a function of the mass of the craft, air density and the aerodynamic coefficients. This has interesting implications. The typical approach to aircraft design is the minimization of weight, because generally aircraft must ascend without assistance. In this case however, the craft is being carried to altitude, so this is not a concern. Instead increasing the weight of the craft, and making it travel faster will make it penetrate the wind better, and as a result, travel further.

The above system dictates that the velocity of a particular airframe is given as a function of its total mass and air density. The glide slope, or distance travelled for every meter of altitude lost will be constant. This is slightly counter intuitive, as in higher altitudes with lower air densities, one would expect the plane to fall rather than fly. However this is simply a matter of velocity. i.e. the plane will fall, for a short while, but the minimal air densities also mean very little drag, which enable the plane to reachi very high velocities, at which stage enough air will travel across he wings for it to fly, despite the low air density.

It is worth noting that the lift and drag coefficients described above actually vary with the aircrafts angle of attack, which describes how an aircraft is oriented relative to it's angle of incidence, as shown in Figure 9. A typical relationship between angle of attack and the lift coefficient is shown in figure 10. Understanding how controlling the angle of attack of a given airframe essentially allows for an evaluation of how controlling the airframe could affect its performance. This is because steady state angle of attack can be controlled with an aircraft's elevators.

In general, an aircraft flying with a higher angle of attack will have a slower steady state velocity, but also a shallower glide slope, whereas flying with a sub-zero angle of attack results in a steeper glide slope and a higher steady state velocity, i.e. a dive. If this relationship could be measured, it would allow for an investigation into how an airframes decent could be optimised with control input.

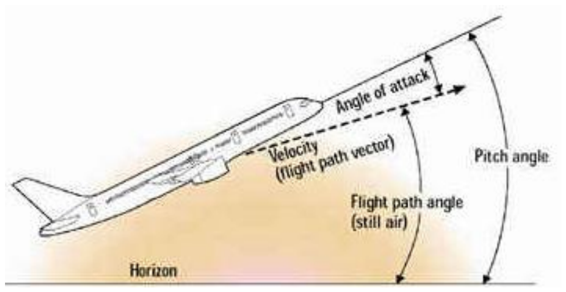


Figure 9: Angle of attack is the difference between the pitch angle of the airframe, and its actual angle of flight

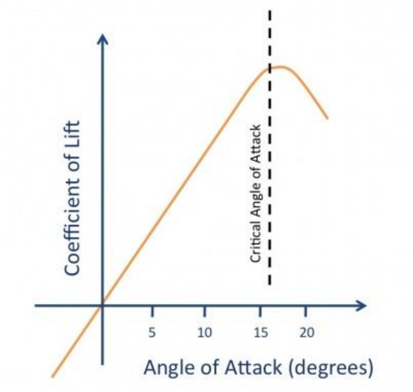


Figure 10: How the coefficient of lift varies with angle of attack. The relationship is linear until the stall angle, where the aircraft cannot operate

2.2 Method / Results

First an analysis was conducted into the distribution of wind speeds at various altitudes, and how NIWA weather balloons are affected thereof. Later into the project, the chosen airframe had been assembled, and flight data became available to which the previously discussed flight model could be fitted. This enabled us to make observations about how the best performance could be derived from an airframe and what proportion of payloads are expected to be recoverable.

2.2.1 Data analysis

To investigate the expected operating conditions, it was first important to obtain a suitable source of data. Organizations such as the NOAA provide large datasets of forecasted weather information [8], including wind speed at various altitudes. NIWA, however, requires clear skies, and calm conditions so that the balloons may be launched safely, Data collected in this means is, therefore, not necessarily representative of these suitable conditions.

Through correspondence with NIWA it was found that they were able to provide a very large dataset containing the measurements recorded from 1653 balloon launches since the year 2000. This provided a more precise image of the expected operating conditions, as it captures the subset of conditions in which NIWA launches their balloons. These logs include measurements of: wind-speed, GPS position, GPS and barometric altitude, temperature, humidity, internal temperature and Ozone.

As an initial foray into the data set, an investigation was made into the drift behaviour of the NIWA balloons. The GPS measurements made in balloon flight can be seen in Figure 11. From this data several basic observations were made; The trending directions of travel, to the North-west and South-west have very important implications for feasibility of using multiple secondary recovery sites. Furthermore, the variance in travel path indicates that gradual changes in wind direction at higher altitudes are a common occurrence. Most significantly, this data demonstrates the sheer distance to which many of the NIWA balloons travel, as much as 200km and it becomes apparent that a 100% recovery of payloads simply isn't realistic.

In the interest of providing a more comprehensive understanding of the required travel distance for a successful solution the distribution of these balloon burst distances was investigated. Shown in Figure 12 is the cumulative distribution function of NIWA balloon displacement at the time of bursting. This demonstrates that even with a craft capable of traveling 70km into the wind, only 50% NIWA payloads could be recovered. This does not even consider that a payload would be only be launched with some safety-margin to ensure that the more expensive payload itself would be able to return to the point of launch.

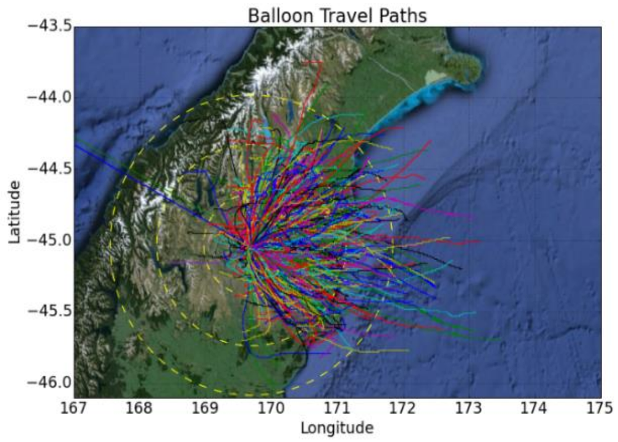


Figure 11: NIWA balloon travel paths, where the yellow rings represent a 50km radius from the point of launch in Lauder, Otago

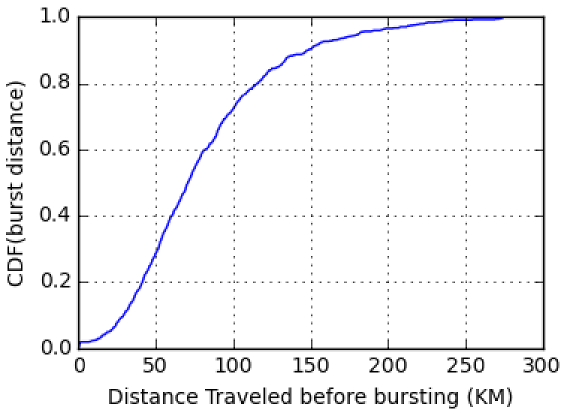


Figure 12: Cumulative distribution function of balloon bursting distance

Next investigation was made into the nature of the wind patterns causing the NIWA balloon travel characteristics, to understand how these would in turn effect the flight of an aircraft. The envelope of recorded conditions can be seen below in figure 13, where all measured wind speed profiles have been superimposed. It is evident that two maxima can occur, the first at around 12km altitude, and the second occurring at higher altitudes, higher than the maximum altitude reached by the balloons. Figure 14 shows a subset of the measured wind speed profiles, demonstrating the two distinct situations.

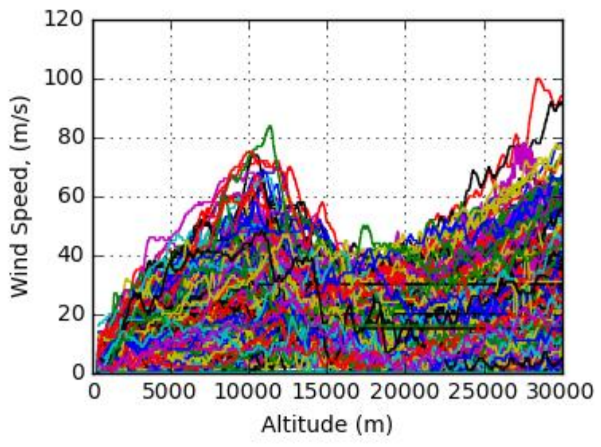


Fig 13: The wind conditions experience by NIWA balloons at various altitudes

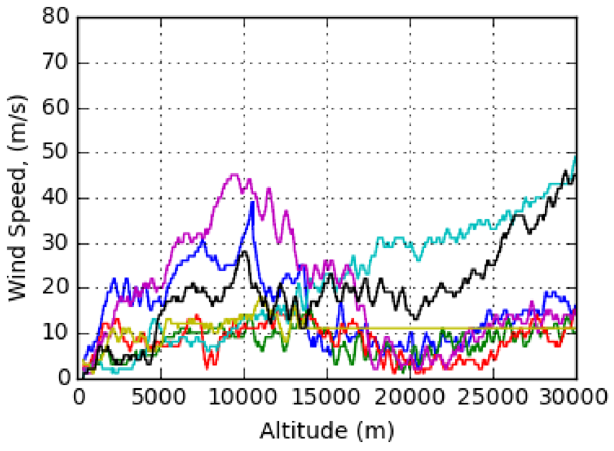


Fig. 14: A subset of the above dataset, showing the distant maxima occurring both at 12km and higher

Figure 13, however only gives an impression of the range of conditions which might be experienced, so a further investigation was made into the distribution of the wind speed data. This was done by separating the data set into sections of altitude. Within these sections the distribution was investigated by finding the maximum recorded value within every fifth percentile. These can be seen in figure 15, where each solid line represents the maximum value within each 10th percentile envelope, and similarly for every dotted line, the 5th percentile. From this it can be observed that the lower maxima is much more frequent, but the second maxima is significantly more variant.

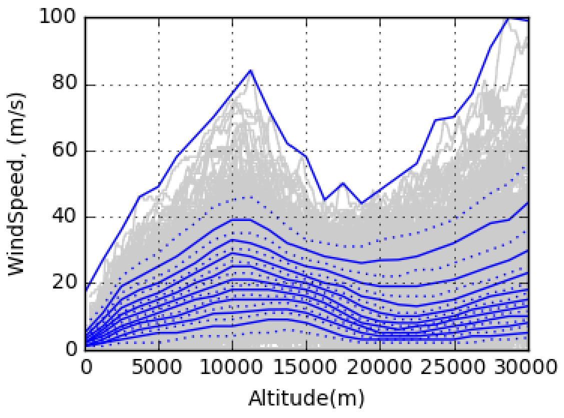


Figure 15: Wind speed distribution at various altitudes

Next the relationship between balloon drift and wind conditions was investigated. Figure 16 shows balloon wind drift as a function of average wind speed, for the entire dataset of NIWA balloon launches. The use of average wind speed to represent the diversity of wind conditions must be considered. In this case, as the ascent rate of the balloons is linear, as can be seen in Fig 17, average wind speed is a good representation of conditions, because the balloon is equally effected by wind speed at all altitude layers.

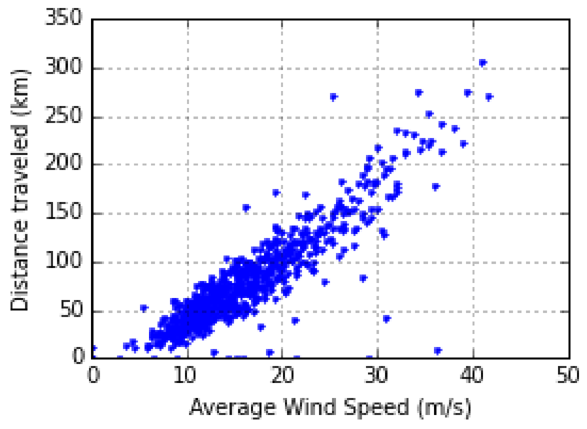


Figure 16: Relating wind speed with balloon drift

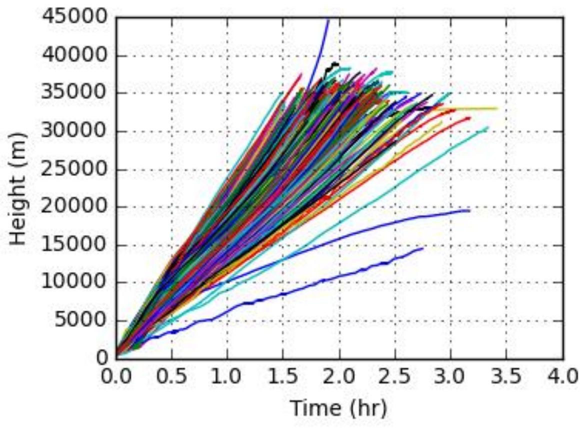


Figure 17 Balloon ascent

While the data is strongly correlated as would be expected, the variance can be traced back to several sources. Not only do the balloons vary in the altitude at which they burst, there is also inconsistency in the rate of ascent, as can be seen in Figure 17. That is to say that some balloons are exposed to the wind for a greater duration than others, and as result drift further than a different balloon might for the same set of conditions. Variance in wind direction also contributes to the spread, as can be seen the previous Figure 11, where the travel paths of the balloons can be seen to change as they ascend.

Given that the ascent rate of the balloons is to some extent controllable by NIWA, investigation was made into how each of the two sources of variance contributed to the overall variance. This might indicate if a possible benefit could be derived by NIWA modifying their launch procedure. This is to say that if less variant ascent rate could be attained, would the use of a self-recovering payload suddenly become more beneficial, either because it would become easier to predict a balloons drift, and/or because the balloons could be better controlled not to drift as far.

For each balloon launch, the variance in wind direction can be accounted for by integrating measured wind speed as a function of height for the full ascent period. Because ascent rate is linear, variance in ascent time can be very simply accounted for by scaling travel distance with the factor to which ascent time differs from some ideal ascent time. That is if ascent takes two hours, and the ideal ascent time is 1.5 hours, the travel distance can be scaled to ¾ of its original magnitude.

Figure 18 shows the effect of correcting for wind direction. As might be expected this shifts the distribution upward, Figure 19 shows the effect of correcting for NIWA's ideal ascent time of 5 m/s (a total ascent time of an hour and forty minutes). It can be seen that this has produced a more significant reduction in the variance of the dataset, and also made the slope more favourable. If both corrections are applied the relationship becomes perfectly linear but interestingly does not reduce the variance to a great extent. This indicates that inconstant wind direction is a regular occurrence, and that the performance of the solution could potentially be improved if the deployment procedure of the balloons could be modified for a more consistent ascent rate.

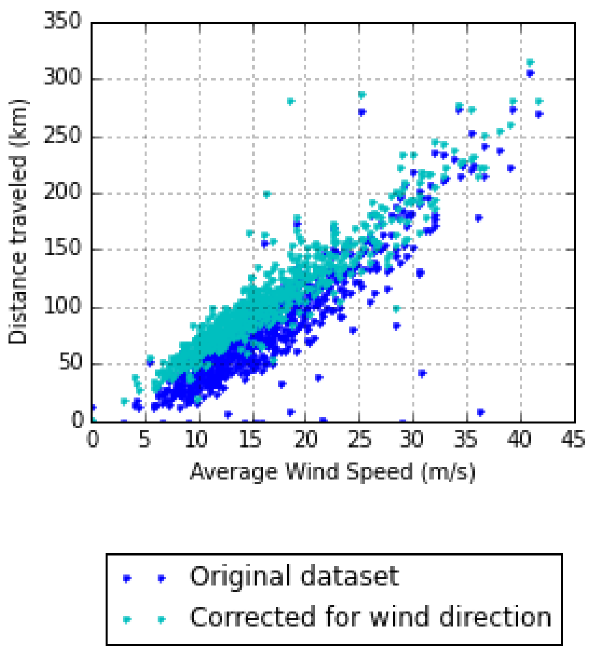


Figure 18: The effect of correcting for wind direction on the distrobution of balloon drift.

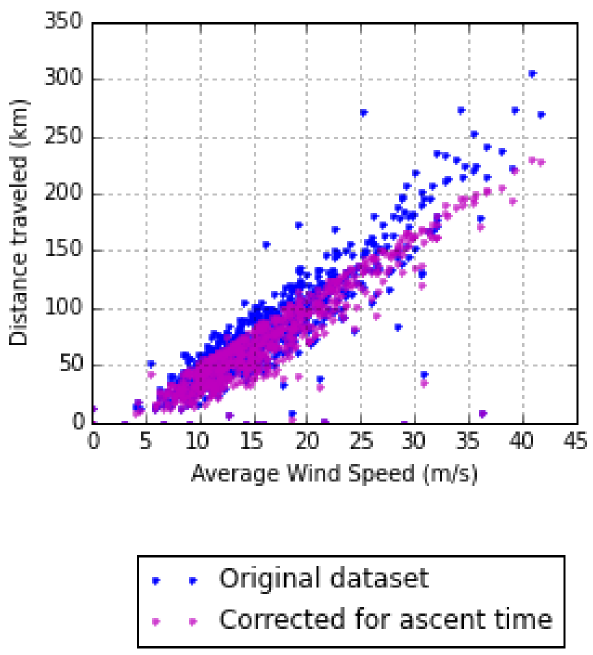


Figure 19: Similarly to figure 11, the effect of varying ascent time was also investigated, a clear wost case relationship emerges

2.2.2 Parameter Identification

To estimate the behaviour of the aforementioned flight model across the whole range of altitudes, the lift and drag coefficients had to be identified from the flight data obtained with the mini-talon airframe. Figure 20 shows measured position of the airframe over the course of a 15 minute flight.

It was discovered that the measurement of angle of attack with only airspeed, IMU and GPS measurements was in-fact non trivial, and required either the use of wind speed estimation [9] [10], or an inertial model of the craft [11], which was not feasible within the scope of the project. Wind tunnel tests had previously been considered, but would have required too significant a time investment.

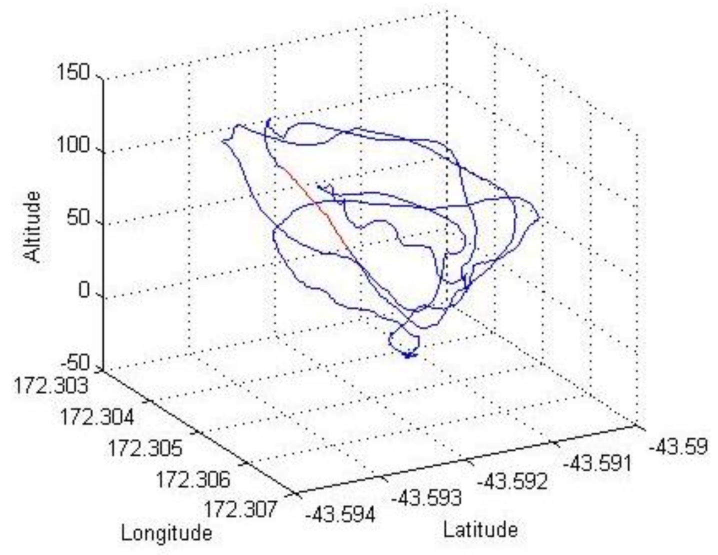


Figure 20: The recorded flight data, with the section used for parameter identification marked in red.

These descents were experienced by the plane at very close to steady state. No more than 10 degrees of attitude variation were experienced over the course of the decent. The airframe was not completely at steady state however as it experienced a slight deceleration along the primary axis of the airframe. This was, however, easily accounted for by fitting the model for the measured deceleration, as shown in equations 10 and 11.

$$ma = \frac{1}{2}\rho v^2 C_{drag} - \ mg\ cos\theta \qquad (10)$$

$$0 = \frac{1}{2}\rho v^2 C_{lift} \ = \ mg\ sin\theta \qquad (11)$$

The measured coefficients were found as below,

$$C_{drag} = \ 0.0212$$

$$C_{lift} = 0.0625$$

2.2.3 Flight modelling

Using the measured aerodynamic coefficients, the expected behaviour of the airframe at higher altitudes can be extrapolated. Forward simulating the steady state flight model from an altitude of 30km predicts 87km of lateral travel as shown in Figure 22, and flight velocities of 133 m/s at high altitudes down to 21 m/s at sea level as shown in Figure 21.

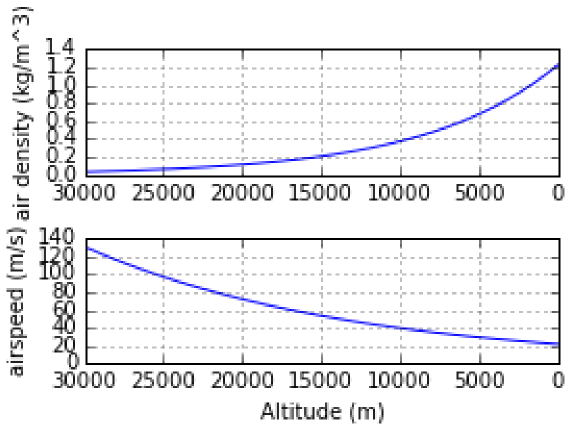


Figure 21: The relationship between air density and steady state airspeed, with both shown as functions of altitude.

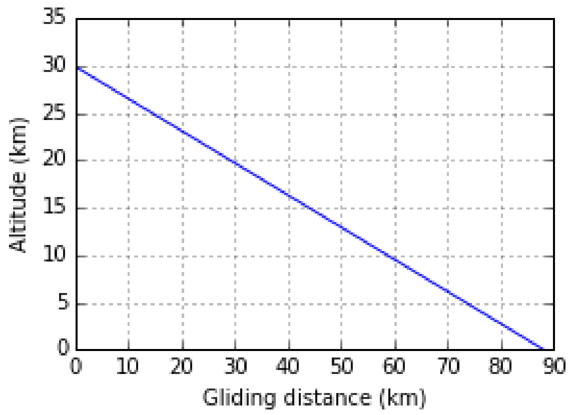


Figure 1: The descent profile of the airframe; constant as would be expected with the model used.

This model does assume that the craft will be travelling at full steady state flying velocity from the moment the balloon burst, which an unrealistic simplification. It is not possible to extend the flight model to emulate an acceleration period, because throughout an acceleration period the craft would transition from being oriented directly downward until the steady state angle of incidence was reached. Modelling this would not only require knowledge of the aircrafts moment of inertia but also how it's aerodynamic behaviour changes with angle of attack, as this also would vary through the acceleration period.

A slightly simpler way to approach this problem is to articulate the model, into an acceleration period, and a flight period. Within the acceleration period the effects of lift are ignored, and the craft is modelled approaching terminal velocity, falling vertically. Once a steady state velocity is reached, which constitutes a velocity great enough to sustain flight, the original model can be substituted. This is a slightly conservative approach to modelling, as a more gradual transition from falling into flying would result in the craft not losing as much altitude during the initial falling period. This behaviour is shown in Figure 23.

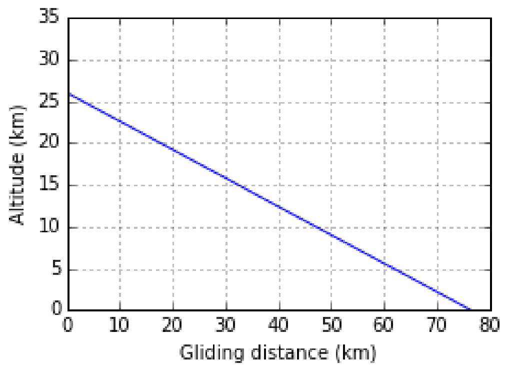


Figure 223: An articulated model of airframe descent, modelling an initial acceleration period before flight begins.

Still this model only provides an estimate of how far the aircraft would travel in ideal conditions. To develop this model such that it may be used to predict the proportion of NIWA balloons would be recoverable, it must also account for the effects of wind.

With a known wind profile (wind speed as a function of altitude, such as those measured by NIWA), wind drift can be determined by numerically integrating wind speed at each point in altitude for the time at which the craft will be there. I.E. if it is expected that, while flying, the aircraft would take 120 seconds to descend from 500 meters to 400 meters, and wind speed at this altitude is 15 m/s, then the wind drift experience during this period of descent can be estimated as 15 (m/s) x 120(s) = 1800(m). This assumes that the measured wind speeds act only horizontally. Performing an integration, as above for the full range of altitude from 30km, the total expected wind drift can be calculated for a full descent.

To evaluate the worst possible case, where the aircraft must fly directly into the wind for the full duration of the return flight, the integrated wind drift is subtracted from the ideal modelled glide distance. This procedure was performed for the full set of wind conditions measured by NIWA, the results of this are shown in Figure 24, where each point is an expected gliding distance under a particular set of conditions.

The variance in the points occurs because of the aircrafts expected variance in velocity as a function of airspeed. Essentially this is because for two different wind speed profiles with the same average wind speed, if the maxima occurs at a higher altitudes, total integrated wind drift will be less than if the maxima occurred at lower altitudes, simply because the aircraft is traveling faster at higher altitudes, and as results spends less time there.

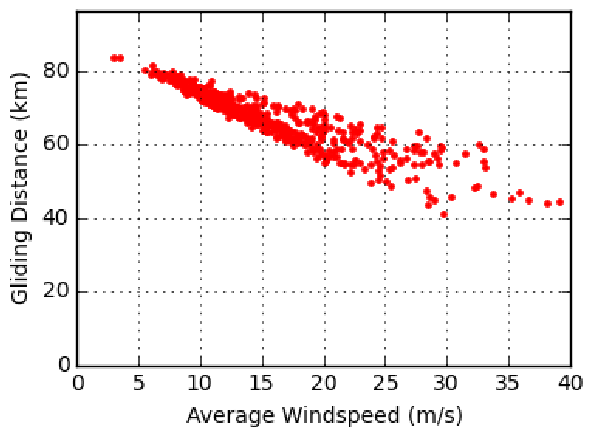


Figure 24: Estimated flight behavior as effected by a headwind. In this case a headwind is modelled using the wind speed profiles collected by NIWA

2.2.4 Simulating powered flight

As previously observed, the situation in which the solution must operate has some interesting implications. The best wind resilience is derived from a craft capable of traveling as fast as possible, with minimal drag. This implies ideal performance can be derived from making the craft as dense as possible. The size and weight of the airframe are constrained by the both the size of the Ozonesonde payload which must be fitted to the airframe, and the weight restrictions imposed by the CAA unmanned Arial vehicle rules, which essentially limit combined balloon-airframe weight to 4kg. Instead of carrying ballast, it makes sense to carry useful weight, which supports the usage of a motor.

Using the recorded flight data, which measures steady state flight velocity and current draw, the characteristics of level flight can be derived. Thus the flight model can be articulated, to model both a period of gliding, down to a set altitude and a period of level, powered flight. The results of this model can be seen below in Figure 25. For an 8 Amp hour batter, a flight time of 40 minutes can be expected. Depending on wind conditions this could allow the craft to travel as much as 40 kilometres beyond its gliding distance.

Comparing this model to the original, after accounting for the expected headwind, which can be seen in Figure 26, adding the power train shifts the distribution as would be expected. Interestingly a slight increase in variance occurs, this is because now the particular wind speed at the powered flight altitude now has a significant effect on the flight performance, and this significance is not accounted for in the average wind speed.

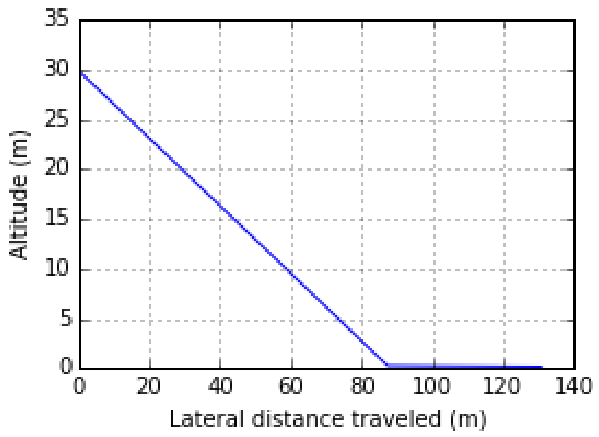


Figure 25: The descent profile of the powered airframe

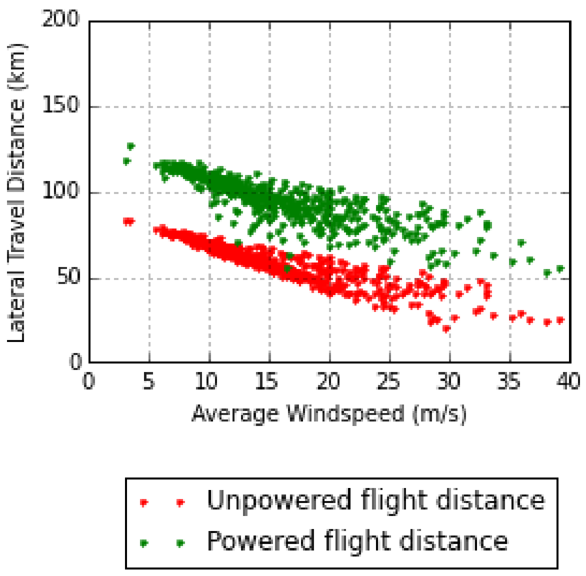


Figure 26: The powered flight model (in green) as effected by wind, compared with the unpowered (in red).

2.4 Discussion

With the ability to estimate the travel distance of our solution it is possible to work out what proportion of NIWA payloads would be recoverable. This was done by estimating what proportion of previous NIWA launches would have been recoverable, had the solution been available. Two approaches were considered for this, the first being to evaluate on a case by case basis, for each NIWA log, and the associated wind speed measurements, however this would not necessarily been indicative of actual recovery rate, as it does not account for the safety margin NIWA would use when considering the launch of the more advanced payload.

A more valuable approach is to generalize, across all recorded data, and find the point at which the safe return of the airframe becomes uncertain. I.e. where wind conditions are such that the worst case expected balloon drift is equal to the expected travel distance of the airframe. Predicted conditions are available to NIWA at the time of launch, and are already used to predict the travel path of the balloons. NIWA quoted these predictions to be reliable.

Figure 27 shows the superimposition of expected balloon drift, and airframe travel distance, for a range of expected wind conditions, as a function of average wind speed. It can be seen that the two data series first intersect with average wind speeds of 12 m/s.   The cumulative distribution function of average wind speeds measured in the NIWA data also be seen in Figure 28 which indicates that average wind speeds equal or less than 12 m/s occur 28% percent of the time. So it is reasonable to say that NIWA could expect to recover around 28% of payloads with the developed airframe.

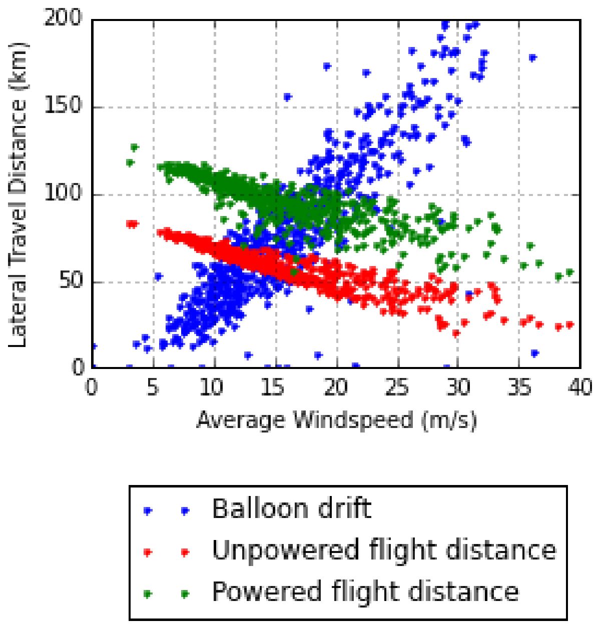


Figure 27: Intersecting expected balloon drift distance, and modelled flight distance both in terms of wind conditions allows for the limit of operating conditions to be determined.

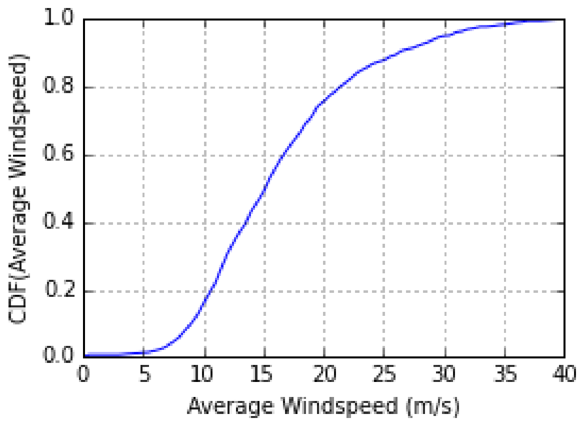


Figure28: The distribution of average wind speeds measured in NIWA balloon launches can be used to determine what proportion of payloads are recoverable

The flight prediction only models the behaviour of an uncontrolled airframe at steady state angle of attack, it is well known that an increased angle of attack an improved glide slope can be attained, this reduces airspeed however so flying as such may make the craft more susceptible to the oncoming winds. Of course a controllable craft could also fly with below steady-state angle of attack, essentially in a dive, which could be used to quickly traverse altitude sections of high wind speed. This begs the question of what benefit could be derived from controlling the aircraft to perform and optimal descent? Previous attempts at self-recoverable payloads have not utilized pitch control, so this presents a promising, and not yet investigated approach.

It is also interesting to consider the potential benefit of a more controlled balloon ascent, it is not well understood how easily this could be attained, but as can be seen in Figure 29, this could increase the limit of operating conditions closer to 14 m/s and could potentially offer an improvement in recovery rate.

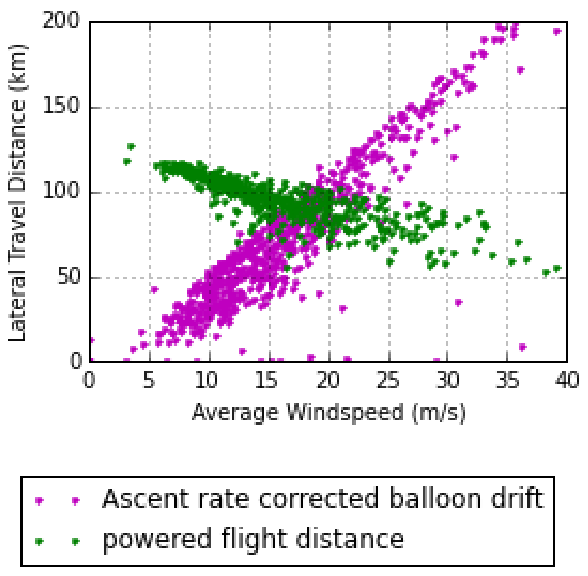


Figure 29: Payload recovery where balloon ascent rate is exactly 5 m/s

Model inaccuracies / Improving estimates

The inability to measure angle of attack limited the accuracy to which the aerodynamic coefficients could be measured. Ongoing study would benefit from a wind tunnel experiments to more accurately measure these confidents so that a more precise estimate of airframe gliding distance could be produced.

There are several flaws in using a steady state flight model such as the one used here. The use of this model essentially assumes that from the moment the balloon bursts, the airframe will be travelling at full velocity. As the gliding velocities at this altitude are the highest that would be experienced for the entire flight, the model will be providing a generous estimate of gliding distance.

However it also assumes the craft is completely uncontrolled. The optimal glide slope of a craft is actually experience with an angle of attack around 5-15 degrees, though this results in a lower airspeed. A very interesting study would be to investigate the optimal descent trajectory, which would aim to optimize travel distance based on current wind speed. Essentially this would look at how the craft might be extend its travel distance by exploiting zones of low wind speed and maximize gliding distance, while minimizing the adverse effects of high wind speed zones, by plunging to lower, less detrimental zones.

Chapter 3. Flight controller Selection, and configuration

Thomas Shortt

3.1 Theory of Operation

3.1.1 UAV Autopilots

Unmanned aerial vehicles (UAVs) are often controlled through the use of autopilots to allow them to become completely autonomous. Autonomous flight allows a UAV to be operated on a completely pre-determined route with no manual intervention, thus allowing tasks to be completed without needing any human resources.

In the case of this project, the National Institute of Water and Atmospheric Research (NIWA) wanted to retrieve electronic equipment used to measure ozone and atmospheric conditions that are attached to weather balloons. The idea was to attach the UAV solution to the weather balloon with the measurement electronics encased inside it. The UAV would, after measurement was completed, detach from the balloon and safely navigate itself to—or as close to—the launch location as possible for recovery and reuse. This task required a completely autonomous UAV as the aircraft would be completely out for line of sight for the vast majority of the descent. Furthermore, this allowed NIWA to launch without having to employ a qualified pilot for each weather balloon release.

UAV autopilots have two main sections. These sections are the autopilot hardware and the autopilot software. The components that make up the autopilot hardware are a flight controller and supporting additional modules such as airspeed sensors, GPS and telemetry radio equipment. The autopilot software is loaded onto the flight controller and is responsible for interpreting inputs such as those from the GPS and airspeed module to produce the output adjustments that control the aircraft. These outputs control the servos on the control surfaces of the aircraft. The control surfaces on a basic fixed wing UAV consist of two ailerons, a rudder and an elevator (figure 1).

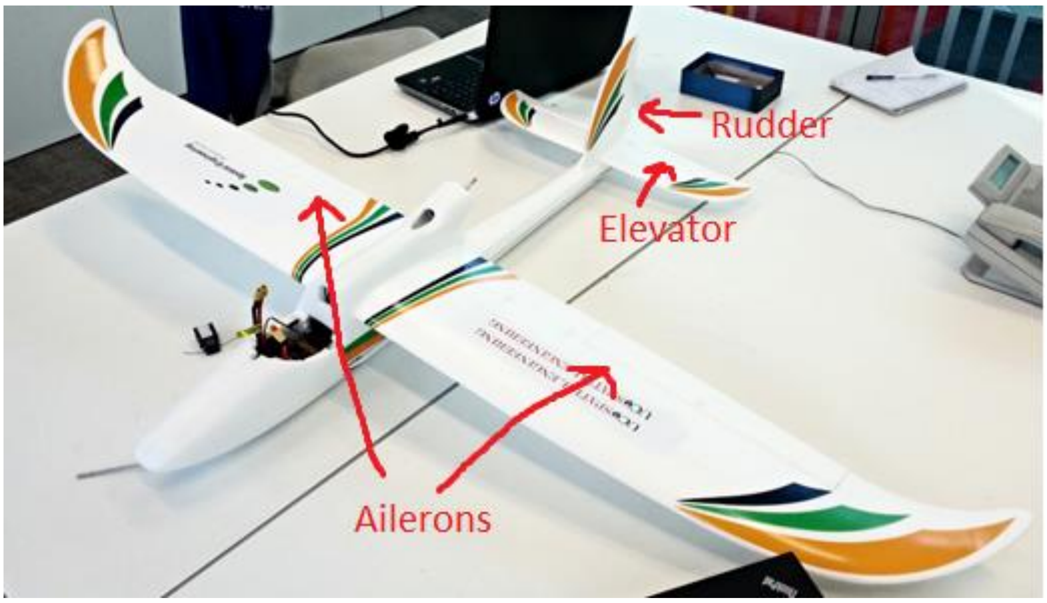


Figure 303: Control surface of Bixler 2 aircraft.

3.1.2 NIWA Motivation

The NIWA launches weather balloons on a weekly basis from their Lauder Atmospheric Research Station in Otago. The balloons are carrying a payload of ozone sensors. NIWA also sends frost-point hygrometer packages monthly. Every unrecovered payload costs NIWA approximately NZ$1000. The vast majority of payloads that NIWA releases are unrecovered as the only current method of recovery is a message on the payload asking for its return, which will result in a small monetary reward.

Allowing NIWA to be more efficient with their atmospheric research budget was the overall goal of this project. If NIWA can expect the safe return of the sensing equipment using our solution it will allow for more expensive equipment to be deployed. That in turn would allow for better, or more, data collection to occur, which for NIWA—as a research organisation—is optimal.

3.1.3 Potential Solutions

Regardless of the type of UAV selected as the solution vehicle—from controlled parachute to fixed wing aircraft—an autopilot system will be needed to control the descent.

3.2 Method

To recover the weather balloon payload an autopilot system would be used to control the return of the payload carrying fixed wing aircraft. An autopilot system can be broken down into two sections: 1) the hardware component, and 2) the software component. The autopilot system is required to control the aircraft during the entire descent to allow for safe recovery of the payload. This means the autopilot needs to be able to use GPS waypoint navigation to follow a predetermined guided path from the point of release from the balloon back to the recovery location (i.e.: original launch site). The autopilot also needs to be able to control the solution without any manual human intervention.

The autopilot system needs to be compatible with additional modules to assist with the overall functionality of the solution. These modules are a GPS module (unless one is included within the autopilot), an airspeed sensor and telemetry radio equipment. The telemetry radio equipment will not be utilised for controlling the solution when finally operational, but it will be used for control of the aircraft throughout the testing phase.

3.2.1 Autopilot Hardware Selection

The next step in selecting the autopilot was to select the autopilot hardware to be used. In this project the hardware and software components of the autopilot system were chosen in parallel, but for the purposes of this report the hardware selection is discussed first with the software selection following.

When selecting the autopilot hardware the main factors taken into account were:

- Price;
- Computational power;
- Flight logging ability, and
- Support.

With these requirements in mind three potential flight controllers were identified. They were the Dragon Circuits Sparky Flight Controller, the 3DRobotics APM 2.6, and the 3DRobotics Pixhawk. These flight controllers will be analyzed and compared in the following section.

3.2.1.1 Sparky

The Sparky flight controller [12] is very basic and cost-effective coming in at US$70. This flight controller comes with an onboard GPS which is capable of basic GPS navigational functions such as holding position, returning to home and waypoint navigation. The Sparky is Mavlink compatible for the use of telemetry radio equipment for locational information and navigation. This flight controller is very compact with a total weight of just 6.9g.

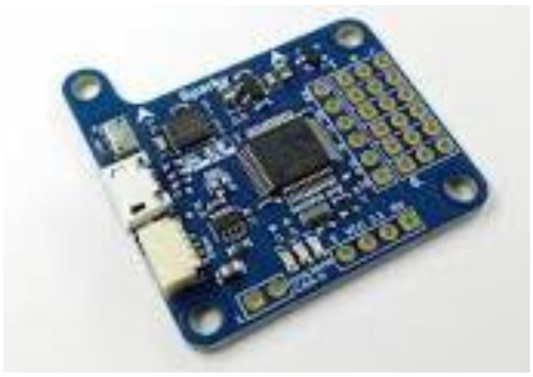


Figure 31: Sparky flight controller.

The Sparky flight controller has quite limited computational ability and no ability to log flights. It also has very limited technical support as it is aimed at consumers with prior UAV building knowledge.

3.2.1.2 APM 2.6

The APM is a completely open source autopilot system that allows the user to turn any fixed, rotary wing or multirotor aircraft into a fully autonomous vehicle [13]. The APM 2.6 is manufactured by 3DRobotics, a US based UAV company. This type of flight controller is an older model which operates using Atmel's ATMEGA2560 and ATMEGA32U-2 chips for processing, accompanied by a 4MB dataflash chip. The APM costs US$160.

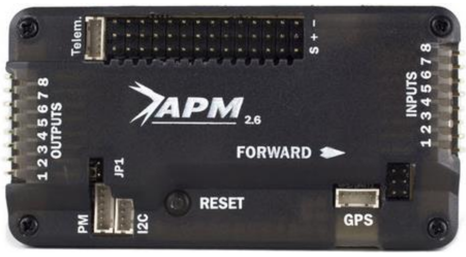


Figure 42: APM 2.6 manufactured by 3DR Robotics.

Currently the APM struggles to run modern autopilot software using essentially 100% of its computational resources. This means that if any major changes are made to the chosen autopilot software there is potential that the APM 2.6 will no longer be able to run the software.

3.2.1.3 Pixhawk

The Pixhawk is also manufactured by 3DRobotics. This flight controller has far superior computational power with its 32 bit ARM Cortex M4 processor. This processor is capable of running a NuttX RTOS (Real time operating system) [14]. Along with the main Cortex M4 processor, the Pixhawk has a 32 bit STM32F103 failsafe co-processor which can take control of the aircraft if a 'catastrophic' failure occurs. This feature makes the Pixhawk more likely to pass some of the Civil

Aviation Authority (CAA) regulations. There is also a micro SD card slot in the Pixhawk which allows for logging flight data during testing. The Pixhawk costs US$200.

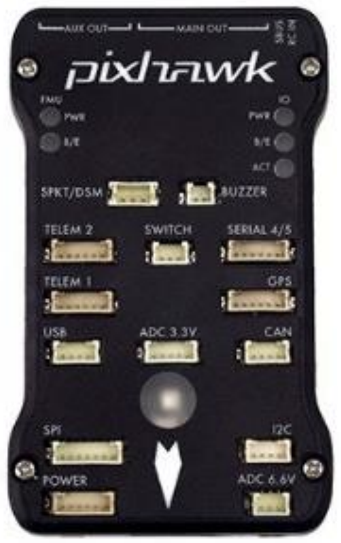


Figure 33: Pixhawk manufactured by 3DRobotics.

All three packages require additional sensors regardless of which is chosen. These additional sensors consist of a GPS and compass unit, an airspeed sensor and a set of telemetry radios for communication with a ground station.

In the end the Pixhawk was chosen as the hardware component of the autopilot system. The choice was based on the fact that for a US$40 increase in price over the APM 2.6, much was gained in the way of extra features. The ability to log flight data will be invaluable during the coming phases of the project. Also, the significant increase in the processing power provided by the Pixhawk will allow a much more customized solution. The Sparky controller was determined to be too minimalistic in its capabilities and features for this project.

3.2.2 Additional Hardware Modules

Alongside the Pixhawk package, a GPS & compass module, airspeed sensor, and telemetry radios needed to be identified and purchased to complete the hardware for the autopilot. As all of these components need to be compatible, it was decided to purchase them all from the same retailer.

The GPS & compass module chosen for the recovery system was the 3DR GPS Ublox Compass which comes at a price of US$90 [15].This module has a small form factor and is very light weight which makes it a good fit for the project. With an update rate of 5Hz it will be able to accurately determine and log its position.

Figure 34: 3DR uBlox GPS and Compass Kit [15].

The next item was an air speed sensor. The selected model was the Pixhawk airspeed sensor kit at a price of US$55 [16]. This sensor was chosen as it is compatible with the Pixhawk package. It can also measure up to 360km/h winds. In addition to measuring wind speed, the sensor will also measure temperature allowing the true airspeed to be calculated.

Figure 35:Pixhawk Airspeed Sensor Kit [16].

The final additional hardware module was a telemetry radio set. The set chosen was the 3DR Radio 433 MHz at a cost of US$50 [17]. This radio was selected primarily due to the fact it is compatible with the Pixhawk package.

Figure 36: 3DR Telemetry Radio 433 MHz [17].

3.2.3 Software Selection

The next step was to choose the autopilot software to run on the Pixhawk flight controller. The main requirement for the autopilot software was for it to have good GPS waypoint navigational capability and well-documented support. The requirement for well-documented support was to allow for the autopilot system to be setup and configured quickly, thus allowing other group members to continue with their parts of the project.

With these requirements in mind, three possible autopilot software packages were identified. These packages were: *OpenPilot*, *ArduPilot* and *Tau Labs*.

3.2.3.1 OpenPilot

OpenPilot [18] is an autopilot software package designed for the CC3D range of flight controller boards. This autopilot excels at acrobatic operation of UAVs, while lacking advanced GPS navigational capabilities. For this project acrobatic flying was not required, whereas GPS navigation was paramount. This implied that *OpenPilot* was not a suitable software package for the project.

3.2.3.2 ArduPilot

The next software package investigated was *ArduPilot* [19]. *ArduPilot* is an open source autopilot that comes in a variety of versions to cover the many different types of UAV. There is a version for remote controlled cars called the *ArduRover*, one for multi-rotor aircraft called *ArduCopter* (with many subversions for the varying multirotor setups), and finally *ArduPlane* which could be used in this project. *ArduPilot* is sponsored by 3DRobotics, the manufacture of the APM 2.6 and the Pixhawk.

ArduPilot excels at GPS waypoint navigation which is crucial to the success of this project. *ArduPilot* is accompanied by ground station software called *Mission Planner* [20]. *Mission Planner* is used to create flight plans, and then loads the flight plans onto the Pixhawk, this can be done either through a USB connection or a telemetry radio connection. *Mission Planner* also allows the solution to be monitored in flight using a telemetry radio connection.

Mission Planner is also used for programming the flight controller’s parameters such as PID control gains, type of airframe (such as V-tail, powered or unpowered), and inversion of control surfaces. This allows for in-depth manipulation of the selected flight controller to gain maximum performance.

3.2.3.3 Tau Labs

The final software package investigated was *Tau Labs* [21]. *Tau Labs* started as an offshoot of the *ArduPilot* software package and is mainly intended for use by University lecturers and students developing open source hardware. The main disadvantage of *Tau Labs* is the lack of support and documentation for use when operating a flying vehicle. This is due to the fact it is primarily used to test open source hardware.

ArduPilot was chosen for this project principally due to the Spatial Engineering Research Centre at the University of Canterbury having already used it successfully and, therefore, having some experience with both the hardware and software components.

3.2.4 Testing

Once the hardware was purchased from 3DRobotics it was assembled. A basic block diagram can be seen below.

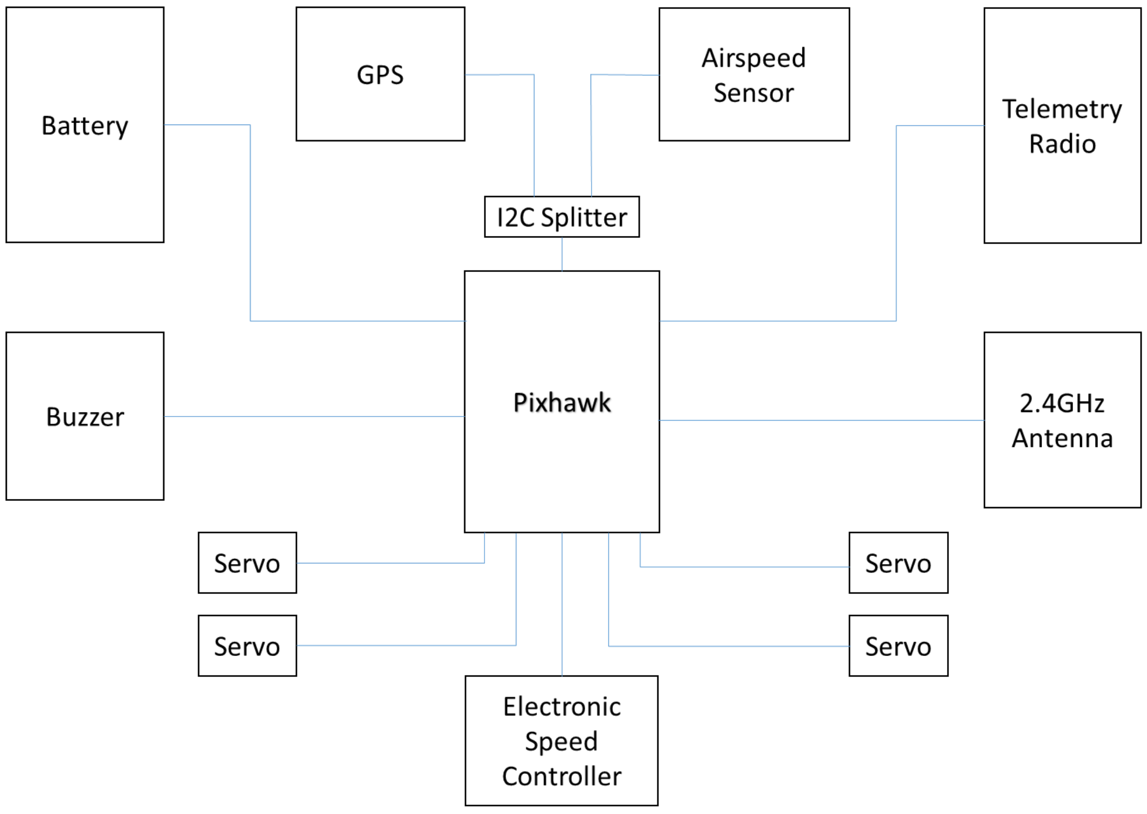


Figure 37: Basic block diagram of final electronics configuration.

The above diagram (figure 8) shows how all of the components connect to the Pixhawk, which acts as the master node in the system. The four servos shown in the diagram are connected to the two ailerons, the rudder and the elevator (figure 9).

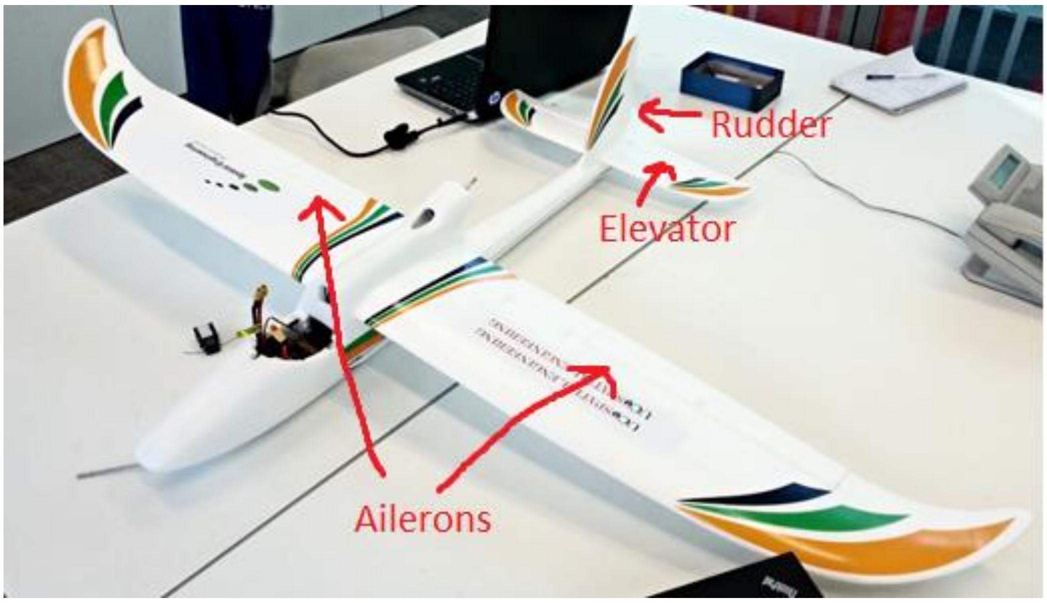


Figure 38: Bixler aircraft with autopilot electronics installed.

After the electronics were assembled, the first step was to load *ArduPilot* onto the Pixhawk. This was done using a USB connection from a ground station. In this case the ground station was a laptop running *Mission Planner* software. Once *ArduPlane* was installed on the Pixhawk a telemetry radio connection was attempted. Initially this connection failed as the bitrate on *Mission Planner* was still set at the required USB bit rate, this had to be lowered to 57600 before a connection was achievable.

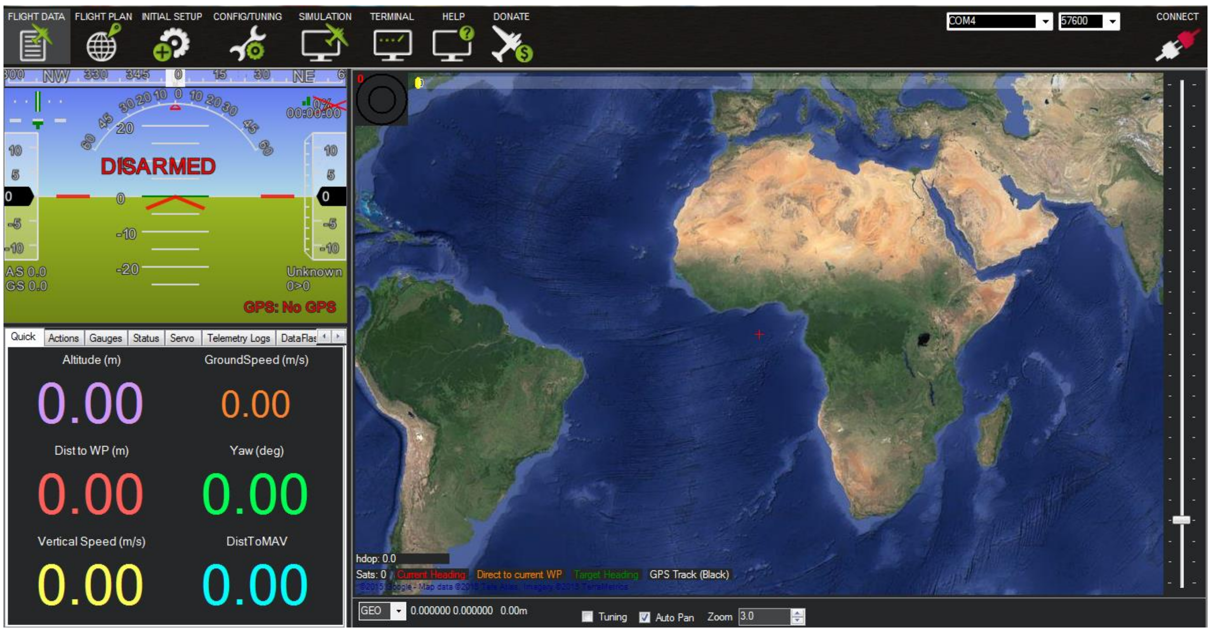


Figure 39: Home screen of Mission Planner.

Once the telemetry connection was achieved it was time to calibrate the GPS and Accelerometer units. This was achieved by following two calibration wizards on *Mission Planner*. Initially these tests continually failed. It was determined that this was due to the electronics being assembled outside of the airframe and not fixed in place. The Pixhawk requires that it and the GPS module be in constant alignment throughout both of these calibrations. This was achieved by mounting all of the hardware inside of the Bixler airframe as seen in figure X.

3.2.4.1 Accelerometer Calibration

To calibrate the accelerometer the accelerometer wizard was started in *Mission Planner*. This wizard then asked for the aircraft to be orientated on all of its different planes of orientation, such as nose down, tail down and so on. Once this test was passed and a message of "Good Accel Health" was returned by *Mission Planner* the next step was to calibrate the GPS.

3.2.4.2 GPS Calibration

Calibration of the GPS was achieved using the GPS calibration wizard in *Mission Planner*. This process was slightly more complicated than the accelerometer calibration. The aircraft had to be rotated around its axis until predetermined points in the form of small white boxes shown in *Mission Planner* were all hit.

Once the hardware was mounted in the Bixler the GPS calibration tests were still failing. Initially the cause of these failures was unknown, but on further inspection of the Bixler airframe it was discovered there were magnets used to hold the cover down. These magnets were in such close proximity they were disrupting the tests. Once the magnets were removed the next attempted GPS calibration was successful with *Mission Planner* returning "Good GPS Health" thus informing the team of a successful calibration.

3.2.4.3 Controller Calibration

Once the accelerometer and GPS were calibrated, the next step was calibrating the radio controller. The radio controller was used to give manual control of the aircraft while flying. While the final solution doesn't require a radio controller, testing would not have been able to be completed without the radio controller. This was due to the fact that the planes flying capabilities had to be tested before the autopilot was tested. This was done to ensure it would not crash due to a fault caused by the aircraft and not the autopilot.

Calibrating the controller again employed the use of a wizard contained in *Mission Planner*. Once the wizard was started and a telemetry link was achieved between the Pixhawk and the ground station a 2.4GHz link needed to be achieved between the controller and the Pixhawk. This was achieved through using a 2.4GHz Spektrum antenna and a paired radio controller. Once this connection was achieved the next step was to flick all of the switches, buttons and control switches to their maximum and minimums to inform *Mission Planner* on the expected PPM signal magnitudes. This then allowed for *Mission Planner* to map the full range of servo control to the full range of motion of the radio controller.

Figure 40: Radio controller used for this project.

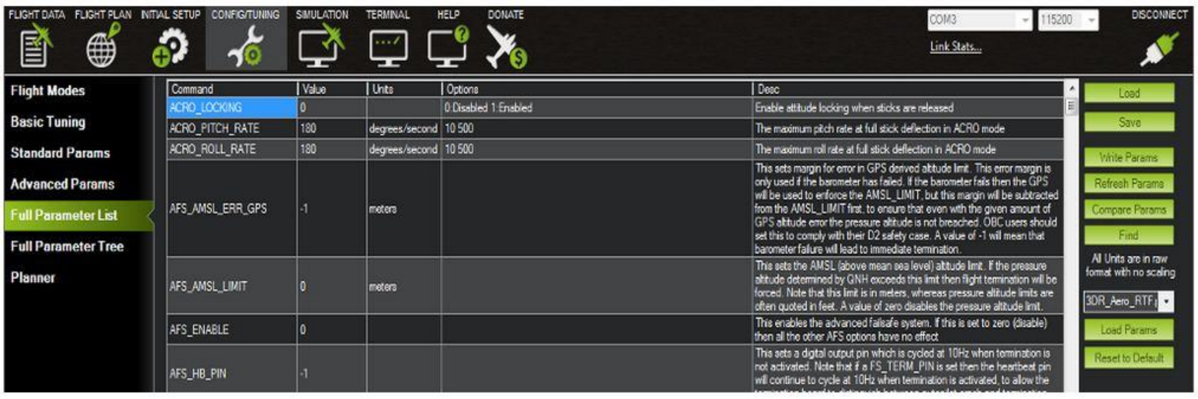
One of the three position switches on the radio controller was used to switch the Pixhawk from different flying modes such as “manual mode” where the pilot had full control to “automatic mode” where the Pixhawk was in full control of the aircraft.

3.2.5 Test flights

3.2.5.1 First test flight

Once all of the calibration was completed a basic test flight was then planned at Ilam fields. The main objective of this test was to make sure the aircraft was in working order and to complete a very basic test of the autopilot. The test planned for the autopilot was for the aircraft to circle around a single waypoint above the field.

Once the team had arrived at the field and was ready to conduct the test, the test pilot informed us that the control surfaces were all inverted. This meant that when a correction was made using either the radio controller or the autopilot it would increase the error, not correct it. To fix this a parameter was found in the configuration file on the Pixhawk that would invert all of the control surface signals. Thus giving the correct response, this change was made through *Mission Planner* then the new configuration file was loaded onto the Pixhawk.



Command	Value	Units	Options	Desc
ACRO_LOCKING	0		0:Disabled 1:Enabled	Enable attitude locking when sticks are released
ACRO_PITCH_RATE	180	degrees/second	10 500	The maximum pitch rate at full stick deflection in ACRO mode
ACRO_ROLL_RATE	180	degrees/second	10 500	The maximum roll rate at full stick deflection in ACRO mode
AFS_AMSL_ERR_GPS	-1	meters		This sets margin for error in GPS derived altitude limit. This error margin is only used if the barometer has failed. If the barometer fails then the GPS will be used to enforce the AMSL_LIMIT, but this margin will be subtracted from the AMSL_LIMIT first, to ensure that even with the given amount of GPS altitude error the pressure altitude is not breached. OBC users should set this to comply with their D2 safety case. A value of -1 will mean that barometer failure will lead to immediate termination.
AFS_AMSL_LIMIT	0	meters		This sets the AMSL (above mean sea level) altitude limit. If the pressure altitude determined by QNH exceeds this limit then flight termination will be forced. Note that this limit is in meters, whereas pressure altitude limits are often quoted in feet. A value of zero disables the pressure altitude limit.
AFS_ENABLE	0			This enables the advanced failsafe system. If this is set to zero (disable) then all the other AFS options have no effect
AFS_HB_PIN	-1			This sets a digital output pin which is cycled at 10Hz when termination is not activated. Note that if a FS_TERM_PIN is set then the heartbeat pin will continue to cycle at 10Hz when termination is activated, to allow the

Figure 41: Screenshot of the configuration file in Mission Planner.

Switching airframes

Once a few test flights were completed concluding that the autopilot system was working, it was time to move the hardware from our trainer aircraft—the Bixler—to our final aircraft a Mini Talon. When the hardware was changed across a few parameters in the configuration file needed to be changed as the trainer airframe had both a rudder and elevator whereas the Mini Talon has a v-tail which consists of two "ruddervators." To adjust for this the signals for the rudder and elevator are now mixed and supplied to the two ruddervators.

3.2.5.2 Additional test flights

Once the initial test flight of the Mini Talon was completed a series of additional test flights were completed to learn more about the behaviour of the autopilot. These tests determined such features as: automatic take-offs, automatic landings and extensive waypoint navigation.

3.2.5.3 Automatic take-offs

During testing it was discovered that the Pixhawk was capable of preforming automatic take-offs. These take-offs were achieved by throwing the aircraft level while having the Pixhawk set into automatic mode. While these take-offs will not be used in the final solution they were used extensively through testing.

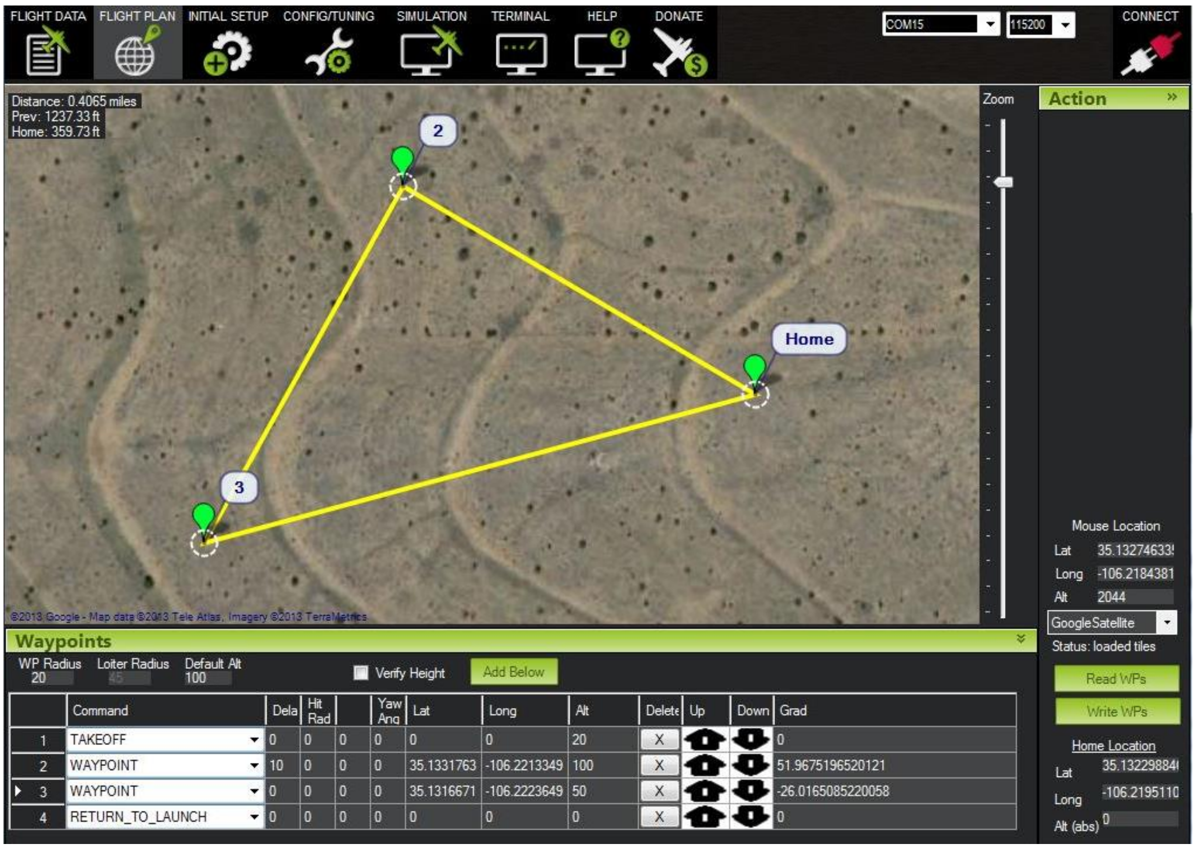
3.2.5.4 Automatic landings

Along with automatic take-offs, the Pixhawk is capable of automatic landings. Automatic landings allowed for the aircraft to complete an entire test flight in automatic mode with no intervention by a human pilot, which satisfied the requirement for the autopilot system to control the aircraft without any manual intervention. The ability to complete automatic landings will be discussed further in the next section.

3.2.5.5 Waypoint Navigation

Throughout testing many different waypoint navigation modes were tested such as loitering around pre-set waypoints, completing pre-set flight plans involving a take-off, landing and multiple

GPS waypoints. The aircraft was also flown into unstable modes of flight such as spins and being inverted then switched into automatic mode proving the Pixhawk could recover from unstable flight.



Distance: 0.4065 miles
Prev: 1237.33 ft
Home: 359.73 ft

Mouse Location
Lat 35.13274633!
Long -106.2184381
Alt 2044

GoogleSatellite
Status: loaded tiles

Waypoints

WP Radius 20 | Loiter Radius 45 | Default Alt 100 | Verify Height

	Command	Dela	Hit Rad		Yaw Ang	Lat	Long	Alt	Delete	Up	Down	Grad
1	TAKEOFF	0	0	0	0	0	0	20	X			0
2	WAYPOINT	10	0	0	0	35.1331763	-106.2213349	100	X			51.9675196520121
3	WAYPOINT	0	0	0	0	35.1316671	-106.2223649	50	X			-26.0165085220058
4	RETURN_TO_LAUNCH	0	0	0	0	0	0	0	X			0

Home Location
Lat 35.13229884
Long -106.2195110
Alt (abs) 0

Figure 42: Screenshot of a basic sample flight plan in Mission Planner.

This extensive testing allowed the project group to conclude that the 3DR Pixhawk would be able to complete all necessary components of a drop from 30km above the earth's surface without any need for manual intervention.

3.3 Results

3.3.1 Autopilot hardware selection

The main requirements determined for the autopilot hardware selection were as follows:

- Price;
- Computational power;
- Flight logging ability, and
- Support.

The Pixhawk flight controller was determined to be the best fit for this project due to the following reasons.

3.3.1.1 Price

While the Pixhawk was the most expensive of the three controllers at US$200, the additional features this controller provided significantly compensated for the increase in price. This controller was still within the budget for the project.

3.3.1.2 Computational power

The Pixhawk has by far the most computational power of all three controllers. The 32 bit cortex M4 CPU will allow the Pixhawk to run complex UAV autopilot softwares such as ArduPilot with ease. There is also room for UAV autopilots to improve without having to worry about replacing the flight controller due to the CPU resources being exhausted.

3.3.1.3 Flight logging ability

As the Pixhawk was the only controller with an on-board micro SD card slot to allow for data logging, it was an obvious choice when concerning this requirement.

3.3.1.4 Support

The Sparky flight controller is a hobbyist controller thus has limited support when compared to the two commercial flight controllers in the form of the APM 2.6 and the Pixhawk manufactured by 3DRobotics. Either the APM 2.6 or the Pixhawk would have been suitable selections when considering this requirement.

As can be seen by the above sections, the Pixhawk flight controller was determined to be the best fit for this project.

3.3.2 Autopilot software selection

Of the three autopilot software packages, *Open Pilot*, *Tau Labs* and *ArduPilot*, *ArduPilot* was selected. *ArduPilot* was selected as it excelled at GPS waypoint navigation and had in-depth support. *Open Pilot* was not chosen as it is mainly focused on acrobatics, rather than navigation, which is not needed for this project. *Tau Labs* could have been used, as it is very similar to *ArduPilot*, but the lack of support would have been detrimental to the project. Also as *ArduPilot* had been previously used in the department additional assistance was achieved through access to various staff members.

3.4 Discussion

3.4.1 Aims

The aim of this project was to supply NIWA with a system to recover their electronic weather balloon payloads. To allow for this a UAV was determined to be the best option. The UAV needed to be completely autonomous, thus an autopilot system was needed. The autopilot system would need to consist of a flight controller, additional modules and an autopilot software package.

3.4.2 Flight controller

While the Pixhawk flight controller was determined to be the best fit of the three controllers investigated, there is potential for another flight controller on the market to be a better fit for this system. With that said, the Pixhawk was proven through testing to be able to accomplish all of the required tasks for the flight controller.

3.4.3 Software package

As with the Pixhawk, there is potential that a better autopilot software package exists somewhere in the market place. Again with that said, the combination of the *ArduPilot* software package and the Pixhawk flight controller gave an autopilot system that met all of the requirements imposed on this project.

3.4.4 Testing

Throughout extensive testing of the autopilot system it was discovered that the combination of a 3DR Pixhawk and *ArduPilot* were ideal for the project. The autopilot hardware and additional modules allowed for all of the required tasks to be completed. The *ArduPilot* software package allowed for extensive GPS waypoint navigation. The combination also allowed for stable flight to be achieved from an unstable mode of flight.

While there was only one iteration of autopilot system for this project, it was installed and tested on two different airframes. The first of these was a test aircraft in the form of a Bixler 2, and the second airframe was the one selected for the project solution, a Mini Talon.

3.4.5 How results achieve aims

NIWA required an autopilot system that would be able to recover their electronic payload. The system needed to be able to return the payload as close to the point of launch as possible. Through the use of a Pixhawk flight controller and *ArduPilot* autopilot software package and autopilot system was developed to fly a Mini Talon aircraft without any manual intervention. Throughout extensive testing it was determined that this combination of software and hardware were able to complete all required tasks at a low altitude. The autopilot system was able to function without any manual intervention, complete complex GPS waypoint navigation and recover from unstable modes of flight. The next step is to test the system at a higher altitude. It is anticipated that the system will perform as intended during these higher altitude tests.

Chapter 4. Controllability and Secondary recovery locations

Callum white

4.1 Theory of Operation

4.1.1 PID Control

It is possible that at high altitudes, the Mini Talon may be in an unstable mode of flight. This is because at high altitudes, the density of air decreases. Since the airframe requires air to flow over its control surfaces to provide the necessary forces for flying, at higher altitudes, it is possible that the Mini Talon will be out of control. The unstable modes of flight could include tumbling or corkscrew-like behaviour.

As the airframe falls from 30 kilometres, it will pick up speed, approaching terminal velocity. The airframe will also experience increasing air density on descent. As the airspeed of the Mini Talon increases, and the air density of the environment increases, there will be more air flowing over the airframe's control surfaces. Eventually, the air flow will be enough for the airframe to become controllable. This is where the control system of the on-board Pixhawk flight controller is able to influence the nature of the Mini Talon's flight.

In order to correct any unstable modes of flight, the Pixhawk utilises separate PID controllers to control the airframe about the pitch and roll flight axes. It is necessary for the control gains about these axes to be tuned to enable the flight controller to correct unstable flight modes, thus allowing the Mini Talon to navigate to predetermined GPS waypoints. The Pixhawk utilises yaw damping for control about the yaw axis.

PID control tuning is a common, and challenging aspect of many engineering problems. In control theory, a controller can be represented as an error equation:

$$u(t) = K_p e(t) + K_i(t) \int_0^t e(\tau) d\tau + K_d \frac{de}{dt}$$

Where:

- $u(t)$ is the controller output which varies with time
- $e(t)$ is the time dependent error between some desired value and some measured value
- K_p is the proportional error coefficient
- K_i is the coefficient for the integral of the error
- K_d is the coefficient for the derivative of the error

In the case of the pitch controller, the error is represented as $e_p(t) = P_{desired}(t) - P_{actual}(t)$, where:

- $P_{desired}(t)$ is the desired angle about the pitch axis
- $P_{actual}(t)$ is the current, measured angle about the pitch axis

The goal of the flight controller, $u(t)$, is to minimise the error, $e(t)$, such that the actual values of pitch and roll converge on the desired values of pitch and roll.

Ideally, a PID controller output would exactly track a desired input. For example, if the input was a step, then the response would be an identical step. However, in reality, this can be difficult to achieve. The three control parameters; K_p, K_i and K_d each influence the controller response in different ways. Figure 43 shows a variety of response variables which depend on a controller's gains.

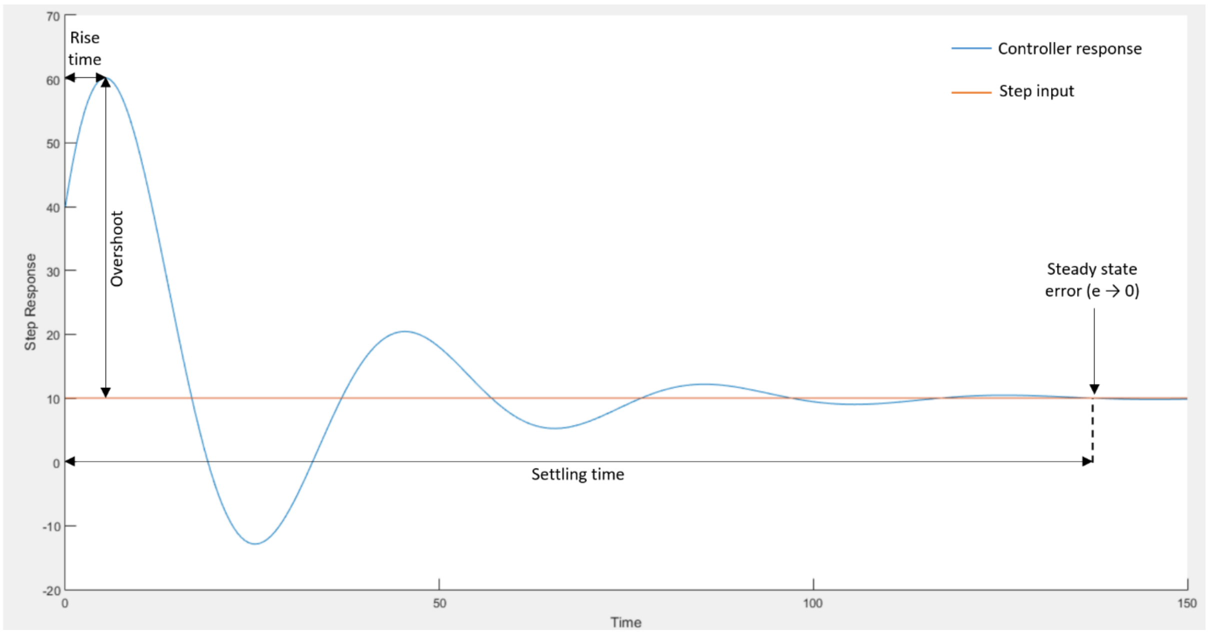


Figure 43. Response variables illustrated for a step input (magnitude = 10 units).

Proportional control gain directly scales the output by the size of the error between the desired and actual values. An increase in proportional gain affects the controller response by decreasing the rise time; increasing the overshoot; decreasing the error in the steady state; and can degrade the stability of the controller.

Integral control gain scales the output as a proportion of the integral of the input error over time. An increase in integral gain affects the controller response by decreasing the rise time; increasing the overshoot; increasing the settling time; eliminating the steady state error; and can also degrade the stability of the controller.

Derivative control gain scales the output as a proportion of the derivative of the input error over time. An increase in derivative gain affects the controller by decreasing the amount of overshoot; decreasing the settling time; and can improve the controller stability if the derivative gain is small. Derivative control is sensitive to noise, which causes steep spikes in the input error. Steep spikes cause the derivative of the error to dominate the controller output and can cause controller instability.

There are many methods to tune the control parameters of a PID controller. Manual tuning (a form of heuristic tuning) is performed by adjusting the proportional, integral and derivative control gains for a PID controlled system, then observing or simulating the system's response to step inputs. Manual tuning is commonly used for complex, or unidentified systems.

One such heuristic approach to PID controller tuning is the Ziegler-Nichols method. This method can be used to provide an "aggressive" controller, with a quarter wave decay response. A quarter wave response is characterised by a decaying overshoot in which subsequent peaks are a quarter the size of the previous peak (see figure 44).

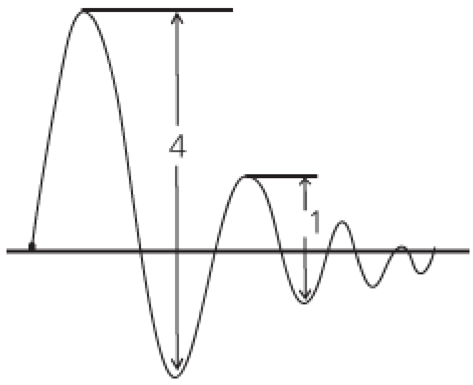


Figure 44. Example of quarter wave decay.

The Ziegler-Nichols method can be used as a first approximation for a set of heuristic control gains. The results may then be manually tuned since for some applications, aggressive control and quarter wave overshoot can be inappropriate. The Ziegler-Nichols algorithm is depicted below in figure 45.

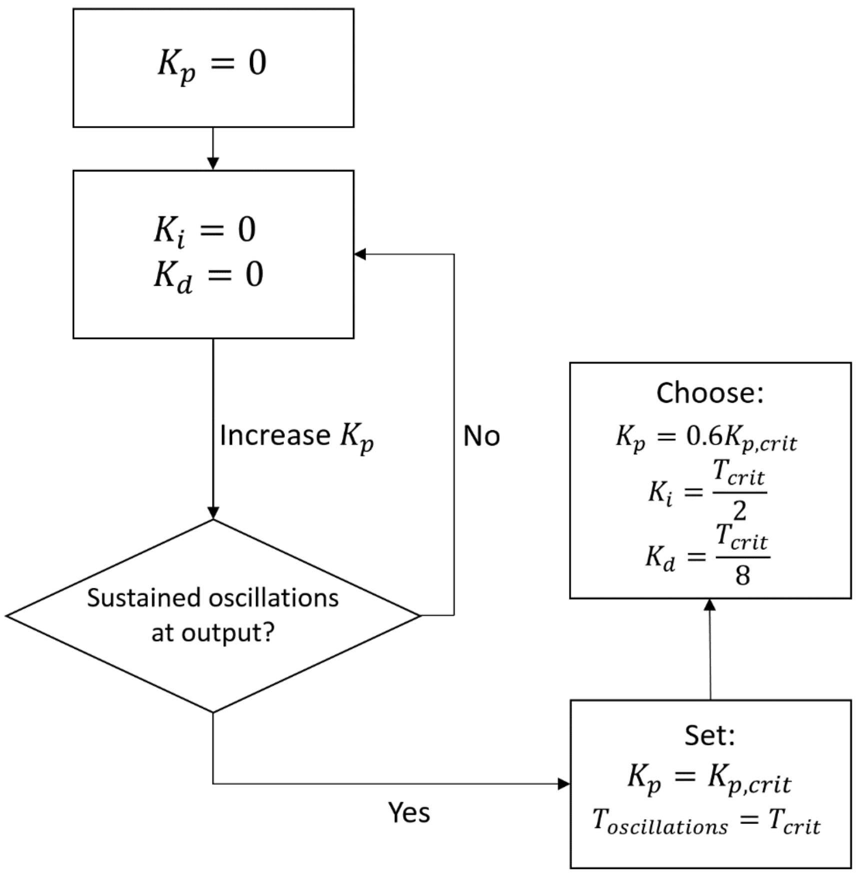

$K_p = 0$
$K_i = 0$
$K_d = 0$
Increase K_p
No
Sustained oscillations at output?
Yes
Set:
$K_p = K_{p,crit}$
$T_{oscillations} = T_{crit}$
Choose:
$K_p = 0.6K_{p,crit}$
$K_i = \frac{T_{crit}}{2}$
$K_d = \frac{T_{crit}}{8}$

Figure 45. The Ziegler-Nichols PID control tuning algorithm.

Depending on the availability, or identifiability, of system information, PID control gains may be determined analytically. Usually information, such as a model of a whole system, is required in order to find an analytical control solution. The performance of the analytical control parameters are then dependent on the accuracy of the system model. Figure 46 shows a general control loop, in which the plant must be identified in order to approximate appropriate control parameters. Depending on the complexity of the system, there may be multiple plants (processes) or disturbances to model.

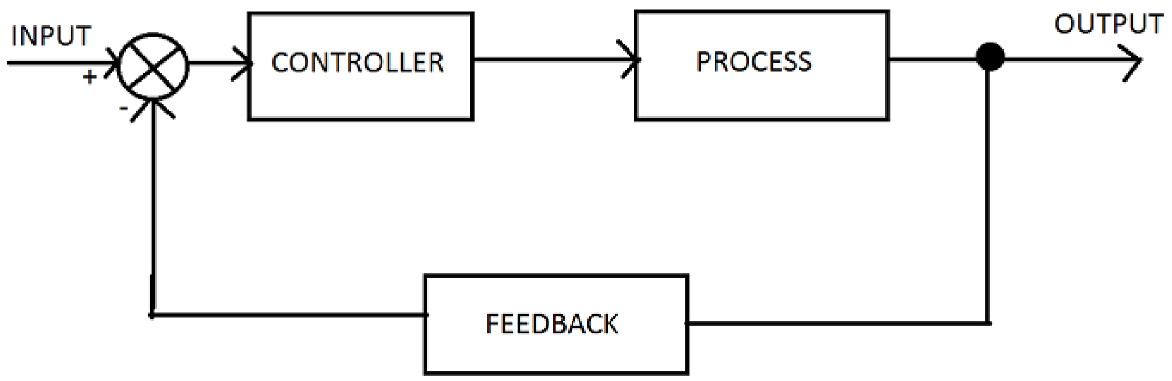


Figure 46. General plant-controller block diagram [24].

PID tuning can also be performed through the use of PID tuning software, or by simulating the controlled system in environments such as MathWork's Simulink package. Such methods require information about the closed loop control system, such as the transfer functions of plants present in the closed loop. In some cases, plant (or process) information may be available either from design parameters, or through the use of system identification, thus enabling the use of PID tuning by simulation or tuning software. If plant information is unknown, or is not easy to identify, a heuristic approach may be preferable.

The Pixhawk's control loop for the yaw axis is characterised by a yaw damper and utilises the parameters $yaw_{slip}, yaw_{damp}, yaw_{roll}$ and $K_{i,yaw}$ to tune an integral controller. A similar method to the Ziegler-Nichols tuning method is used for tuning the Pixhawk's yaw damper [25]. The method involves setting yaw_{slip}, yaw_{damp} and $K_{i,yaw}$ to zero and yaw_{roll} to one. yaw_{damp} is then slowly increased until the tail of the airframe attempts to "wag" from one side to the other. yaw_{damp} is then set to half of this value. yaw_{roll} must then be increased until the nose of the aircraft does not yaw in favour of the inside or outside of a turn. The yaw_{slip} and $K_{i,yaw}$ gains only need tuning if the aircraft experiences lateral acceleration when rolling.

4.1.2 Return to Launch Operations

According to feasibility studies and statistical modelling (see chapter 2), it is not always possible to recover NIWA's payload electronics to within ten kilometres of their launch base at Lauder, Otago. Therefore, it was necessary to design safety features into the system to ensure that the on-board electronics are recoverable, despite not being returned to within ten kilometres of the launch location.

A safety net algorithm was designed and implemented to enable the flight controller to choose between secondary landing zones, such as farms or radio flyer's clubs, as arranged by NIWA. The safety net algorithm operates to ensure that the payload electronics are always recoverable. The glide slope of the aircraft is a necessary parameter in the success of the safety net algorithm.

4.2 Method

4.2.1 Controller Tuning

In order to tune the Pixhawk's on-board controllers using tuning software or simulation, it would have been necessary to have a model to represent the response of the Mini Talon airframe about the pitch, roll and yaw axes. Since I do not have a background in modelling aerodynamic systems, this approach to determining appropriate control gains was discarded. Since an analytical approach also requires a model encompassing the Mini Talon's pitch, roll and yaw responses, this was also discarded.

In an attempt to shortcut the gain tuning process, I first researched control gains used by others on Mini Talon aircraft. The researched control gains were $K_{p,pitch} = 0.81, K_{i,pitch} = 0.10, K_{d,pitch} = 0.07, K_{p,roll} = 1.62, K_{i,roll} = 0.10, K_{d,roll} = 0.14, K_{i,yaw} = 0, yaw_{slip} = 0, yaw_{damp} = 0$ and $yaw_{roll} = 1.0$. The flight controller was then tested for its ability to control the aircraft from an unstable mode of flight. To perform this test, the Mini Talon was manually flown into a vertical stall, then switched to autopilot mode. The ideal result would have consisted of the aircraft correcting the unstable, corkscrew-like behaviour induced by the stall, thus returning to a stable, and controlled, mode of flight. The actual result showed that the researched control gains were inappropriate, as the aircraft continued the corkscrew behaviour before crashing. Since the researched control gains were deemed inappropriate, I decided to use a manual, heuristic approach.

Firstly, to find an initial set of appropriate PID control gains (for pitch and roll), I utilised the Ziegler-Nichols method of control gain tuning. To do this, the PID control gains for the pitch and roll axes were set to zero. The reaction of the aircraft's control surfaces, namely the ailerons and ruddervators (rudders and elevators) shown in figure 47, was observed while the proportional control gains about each axis were slowly increased. Once the control surfaces were steadily oscillating, the proportional gains were set to 0.6 times their current value. Since the system response was being observed by eye, due to the lack of a graphical representation, it was impractical to approximate the frequency of the control surface oscillations. As a result, the integral and derivative control gains were slowly increased to lower the amplitude of the control surface oscillations.

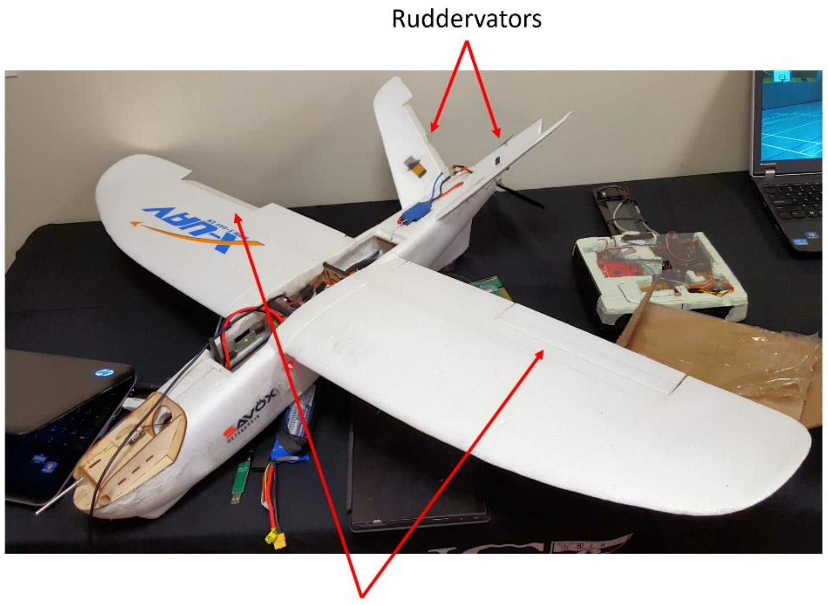


Figure 47. The Mini Talon airframe showing ailerons and ruddervators.

To tune the yaw damper, I used the yaw damper tuning method described in section 4.2.1. It was discovered that the researched control gains for the yaw damper were already suitable for the Mini Talon airframe.

The control gains were then tuned while in flight to reduce the visible overshoot and response time of the aircraft when given a manual control input about the pitch, roll and yaw axes. Utilising knowledge of the impact of proportional, integral, and derivative control gains (see section 4.1.1 on PID controllers), the response of the aircraft was manually tuned. During tuning of the aircraft's control gains, the flight response to manual inputs was observed until it was fast and had invisible overshoot to the human eye. It was difficult to determine whether or not there was steady state error present in the final set of tuned control gains, as the desired angle was unmeasurable from the hand-held radio controller, and the steady state angle of the aircraft in flight was unmeasurable by eye.

4.2.2 Safety Net Algorithm Design

The purpose of the safety net algorithm was to provide a means of selecting a secondary landing zone if the primary landing zone, at Lauder, could not be reached. In order to achieve this, the safety net algorithm first deals with the case in which the primary landing zone is achievable. Secondary landing zones could be sites, such as farms or radio flyer's clubs, at which NIWA acquires permission to land the solution. These sites should be chosen to create a landing constellation in the balloon flight vicinity. Figure 48 shows an example secondary landing zone constellation for a case where a weather balloon bursts up to 50km from Lauder in the South-East direction.

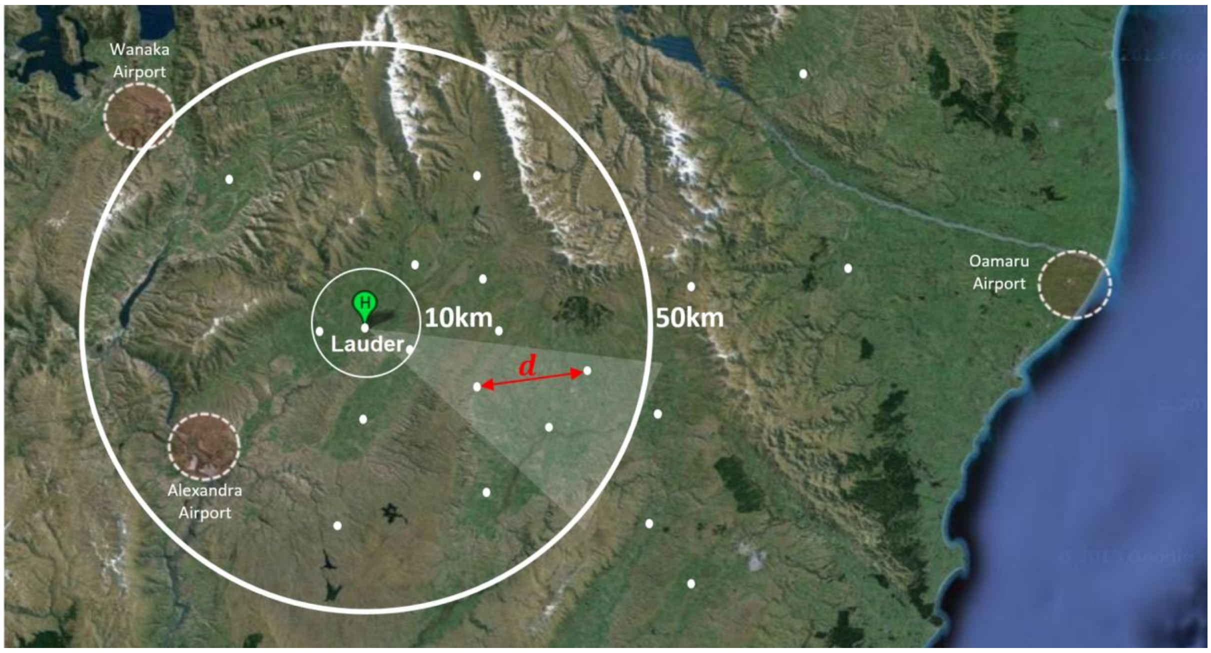


Figure 48. Example landing constellation (white points) for estimated balloon drift in the S-E direction (grey triangle).

For the case shown in figure 48, the safety net algorithm would be able to choose from any of the indicated secondary landing zones based on the shortest distance from the Mini Talon's current location to a secondary site. Changing the course destination from the primary landing zone at Lauder to a secondary landing zone is only performed when a specific criterion is achieved. Figure 49 shows a block diagram of how the safety net algorithm operates. The safety net algorithm is also explained below.

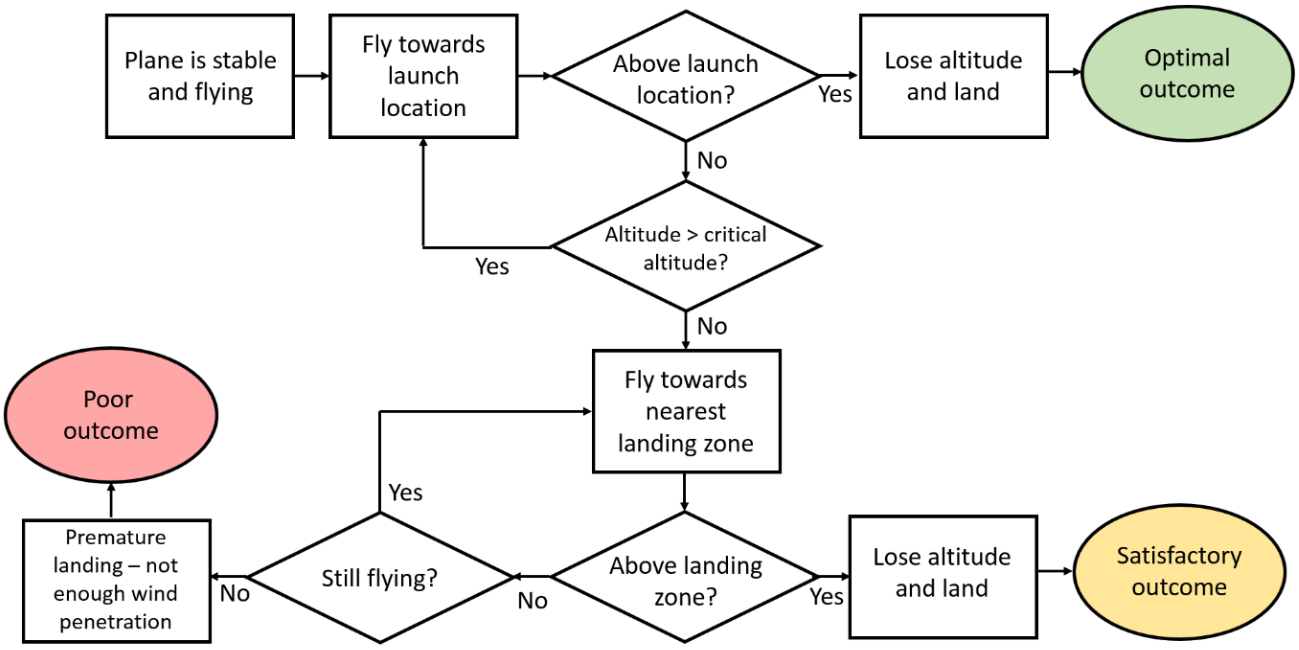


Figure 49. Block diagram of the safety net algorithm.

Once the Mini Talon is stable and flying, as a result of pitch, roll and yaw control, the aircraft navigates towards the launch location. While in flight, GPS latitude, longitude and altitude data is used to determine whether or not the solution has reached the launch location. If the Mini Talon is above the launch location, the aircraft is allowed to lose altitude and land. This is classified as the optimal outcome as the solution will have landed within ten kilometres of Lauder. In order to avoid triggering the critical altitude criterion, mentioned later, the aircraft is not allowed to navigate directly to the ground, but first must achieve a target zone in airspace above the launch location, at an altitude greater than the critical altitude.

Until the aircraft reaches the target zone above the primary landing zone, the solution continuously checks whether its current altitude is greater than a critical altitude. The critical altitude is determine based on the glide slope of the aircraft, and is such that the solution will always be able to land at a secondary landing zone if the primary landing zone is unachievable.

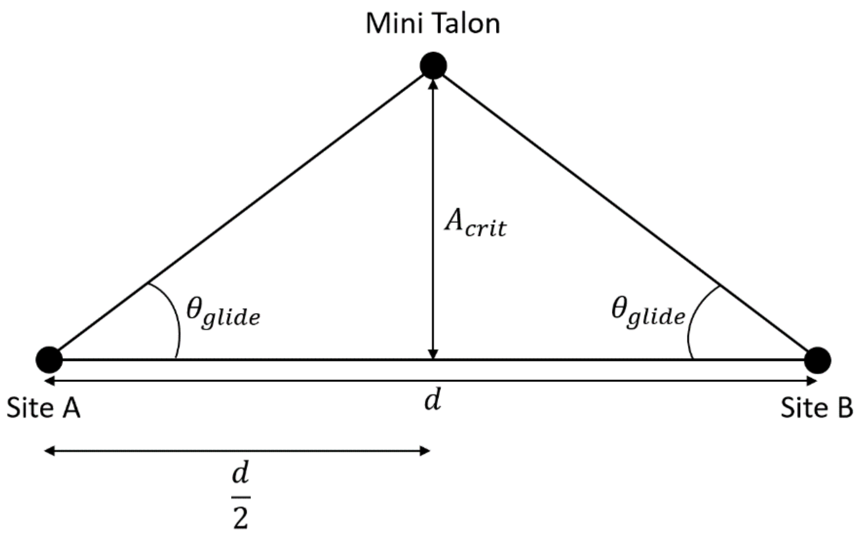


Figure 50. Pictorial representation of the critical altitude.

The critical altitude can be determined using figure 50, where:

- d is the greatest distance between two adjacent landing zones on the return flight path (with no other landing zones in between). For the example case in figure 48, the distance, d, is shown in red
- θ_{glide} is the angle formed between the glide path and the ground
- A_{crit} is the critical altitude

From figure 50, the critical altitude can be calculated by

$$A_{crit} = \frac{d}{2}\tan(\theta_{glide})$$

which ensures that when the Mini Talon is equidistant from two landing zones, it has enough altitude to enable landing at either site. Thus, the solution will always be able to fly to a landing zone if the primary zone is unreachable. This relies on NIWA arranging enough secondary landing zones in the expected balloon drift region (see figure 11 in section 2.2). It is also assumes that the solution would not be launched in conditions where the Mini Talon would not be able to fly far enough to reach any landing zones.

If the Mini Talon's altitude is greater than the critical altitude, then it continues to fly towards the primary landing zone. If the altitude falls below the critical altitude, the solution changes course and heads for the nearest secondary landing zone, based on GPS latitude and longitude. If the Mini

Talon detects that it is above a secondary landing zone, then it loses altitude and lands. This is classified as a satisfactory outcome, as the payload will not be within the desired ten kilometre radius of Lauder. Despite not being fully recovered, knowing that the solution has landed at a secondary landing zone makes the payload recoverable. The solution could then be recovered by requesting the land owner of the secondary site to mail the payload back to Lauder.

Furthermore, if the Mini Talon lands at a secondary, or primary, landing zone, it is known that the payload electronics are undamaged. This is because each landing zone can be customised to have its own descent path which is necessary to avoid obstacles, such as hedgerows, fences and buildings. It should be noted, that NIWA has contact with their own on-board electronics in order to determine the landing location of the solution. In the case where contact is lost with the payload, secondary landing zone owners of likely landing zones (along the return flight path) can be contacted to determine where the aircraft has landed.

If the Mini Talon is unable to land at a secondary waypoint, it is concluded that sufficient wind penetration could not be achieved. This case is included for completeness and in theory should not occur, unless the solution is launched in unrecoverable conditions, such as those indicated in section 2.3. The outcome of this case is classified as poor, since the Mini Talon would not have landed at a designated landing zone, therefore may have crashed into an unknown obstacle.

The algorithm is implemented by uploading a formatted data matrix to the Pixhawk flight controller, via the flight planning software package, Mission Planner. The formatted data matrix columns contain information such as action flags, latitude, longitude, altitude and other action associated parameters, for example loiter radius. Each row of the data matrix constitutes one action taken by the flight controller. The data matrix can be used to implement conditional logic, and thus implement the aforementioned safety net algorithm, by setting the action flag parameters appropriately. A diagrammatic representation of the formatted data matrix is shown in figure 51.

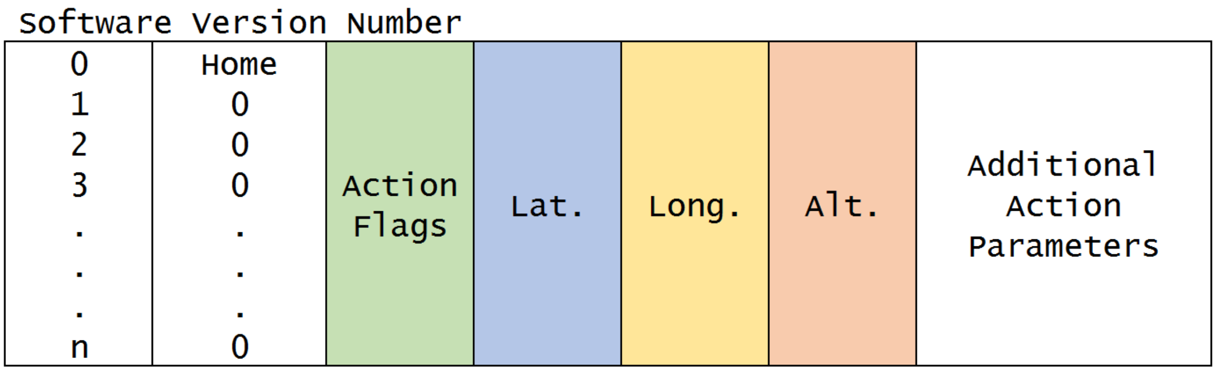


Figure 51. Data matrix format for flight plan implementation.

The first column of the matrix denotes the index of each action taken by the flight controller. The first row of the data matrix is reserved specifically for identifying the latitude, longitude and altitude of the "home" or launch location. The action flags column contains numbers associated with specific flight actions. Depending on the value of the action flag, the flight controller may attempt to take off, land and loiter at a designated latitude and longitude. Action flags can also be set to jump to different rows of the data matrix when specific conditions are met.

Finally, I implemented code (appendix 1) using Matlab which either generates a new flight plan using the Lauder coordinates as a home location, or takes the latitude, longitude and altitude values for a secondary landing zones and appends them to an existing flight plan. The code outputs a matrix which can be saved as a ".txt" file and loaded onto the Pixhawk's memory. It should be noted, that the code in appendix 1 has no knowledge of the actual landing zone, therefore it may be necessary to manually adjust the approach angle of the aircraft when landing to avoid obstacles present at the site.

4.3 Results

4.3.1 Controller Tuning

To test the control gains, the aircraft was manually manoeuvred into unstable flight modes before being switch into autopilot mode. The behaviour of the aircraft was then observed as the flight controller attempted to correct the unstable flight, utilising the new control gains.

Unlike the researched set of control gains from section 4.2.1, the manually tuned control gains enabled the flight controller to correct itself from a variety of stall cases, inducing both tumbling and corkscrew behaviour. The results were recorded in video format and show that the aircraft can successfully correct itself from unstable modes of flight. Since the flight controller does not record the response of its pitch, roll and yaw controllers, it was difficult to gain measurements that quantify the response of the system in flight. An estimate was made that the tuned flight controller could correct a tumbling mode of flight to a stable flight mode within approximately five meters of falling. This however, is a crude human estimate. Figure 52 shows snap shots of the aircraft correcting itself from an unstable mode of flight, filmed from the ground. Overall, the final set of PID and yaw damper control gains were $K_{p,pitch} = 1.30, K_{i,pitch} = 0.10, K_{d,pitch} = 0.02, K_{p,roll} = 1.5, K_{i,roll} = 0.20, K_{d,roll} = 0.03, K_{i,yaw} = 0, yaw_{slip} = 0, yaw_{damp} = 0$ and $yaw_{roll} = 1.0$.

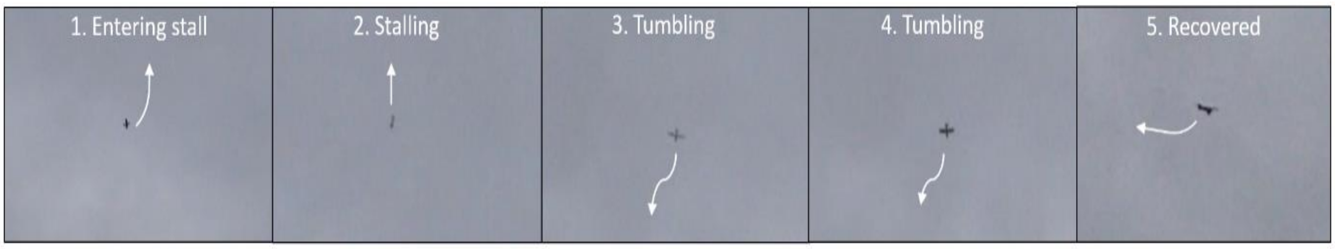


Figure 52. Snapshots of the controller correcting unstable flight.

4.3.2 Safety Net Algorithm

The safety net algorithm was tested on a small scale, at low altitude, due to restrictions on unmanned aerial vehicle flight. By implementing the algorithm on the Pixhawk flight controller, it was found that the algorithm operated as expected, according to the description in section 4.2.2. Since the vertical accuracy of the u-Blox GPS module not trusted (see section 3.2.2), and that testing occurred at low altitude, a manual switch was used to simulate the critical altitude criterion. This was done since it was difficult to determine the true altitude of the aircraft operating within a low flight ceiling and unknown altitude accuracy. When the switch was turned on, the flight controller correctly determined the nearest secondary landing zone to its current position and performed a landing. In cases where the switch was not turned on, the aircraft performed expected landings at a designated primary landing zone.

4.4. Discussion

4.4.1 Aims

To enable recovery of NIWA's electronic payloads, it was necessary to tune the on-board pitch, roll and yaw controllers of the Pixhawk flight controller. Tuning the control systems allows the Mini Talon to fly in a stable manner, and recover from potentially unstable modes of flight. These unstable flight modes may occur with high wind speeds and low air density. The worst case unstable flight modes would involve tumbling or corkscrew behaviour, while lesser cases include simply falling. It is unlikely that the Pixhawk controllers will be able to control the Mini Talon's flight immediately when the weather balloon bursts. This is due to low air density at high altitudes (figure 53) which results is low air flow over the Mini Talon's control surfaces, thus preventing controllable flight.

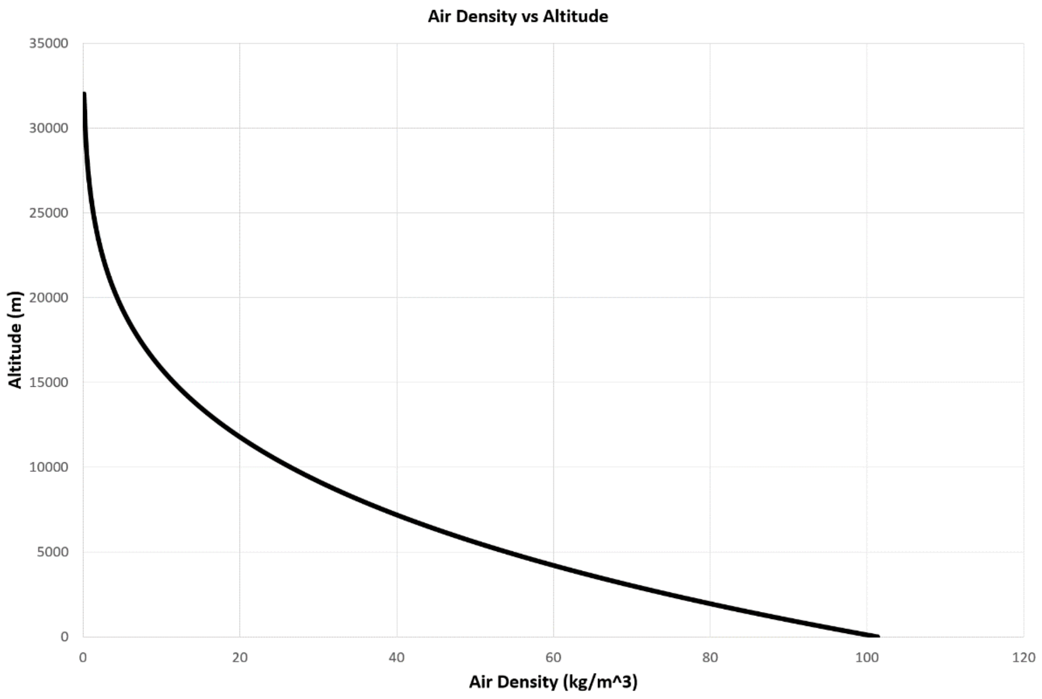


Figure 53. Air density versus altitude curve.

As the Mini Talon falls, picking up speed and entering higher air density space, the air flow over its control surfaces will increase, eventually making the system controllable. The control systems must then be able to stabilise the airframe and navigate the aircraft towards the primary launch location, at Lauder.

To increase recoverability of NIWA's electronic payloads, the flight controller follows a safety net algorithm on the return flight. The purpose of the safety net algorithm is to provide a means of safely landing the Mini Talon airframe, thus protecting the on-board electronics, in cases where it is impossible to return to the launch location. The safety net algorithm provides decision making logic, allowing the flight controller to switch to the nearest possible secondary landing zone when necessary. Secondary landing zones may include, but are not limited to, sites such as farms or radio flyer's clubs on the expected flight path which NIWA has acquired permission to land at.

The ultimate goal behind tuning the control systems and implementing a safety net algorithm is to ensure, and increase, the recoverability NIWA's payload electronics. In cases where the Mini Talon lands outside of the ten kilometre radius around Lauder, which NIWA specified as the maximum ideal landing distance, NIWA can arrange the return of their electronics, via postage, by contacting the owners of secondary landing zones.

4.4.2 Iterations/Relate Results and Theory

During the process of tuning the controllers of the Pixhawk flight controller, there were variety of control parameter iterations. Before control tuning was considered on the Mini Talon airframe, a "test bench" airframe, known as the Bixler 2, figure 54, was used to set-up the Pixhawk. The initial control parameters for the Bixler were deemed unstable, which was determined though test flying. The first test flight in automatic mode proved that the control parameters were unstable, since the flight controller was unable to control the airframe without veering towards the ground.

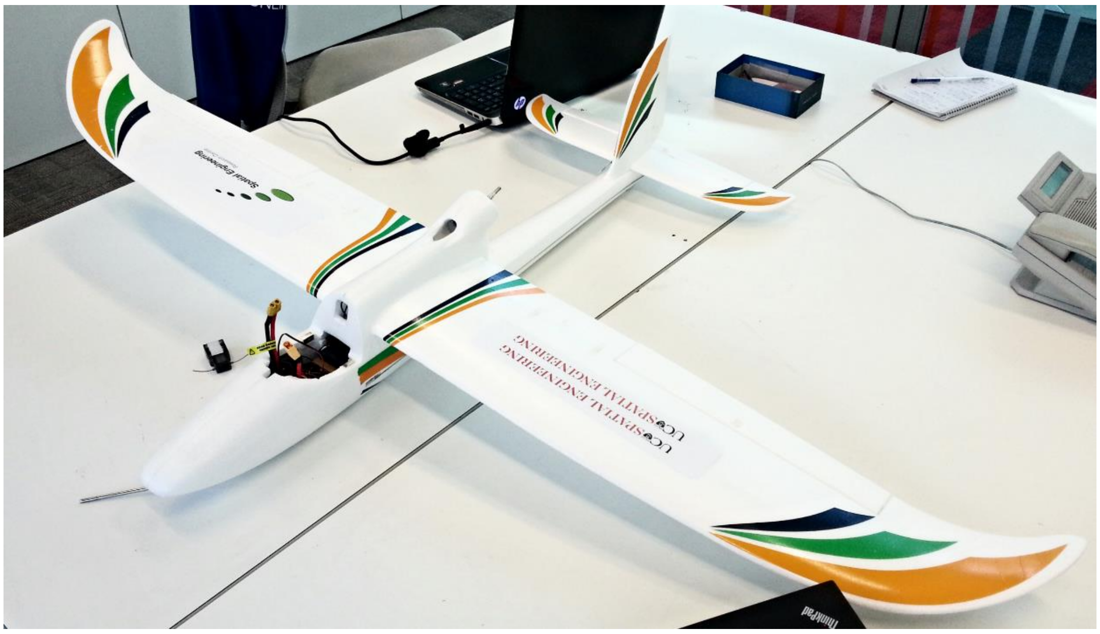


Figure 54. The Bixler 2 "test bench" airframe.

The control parameters were then updated by researching control values that had been used on Bixler airframes in the past. The set of parameters determined by research were appropriate for the testing performed on the Bixler frame.

When the Pixhawk flight controller was transferred to the Mini Talon airframe, the control parameters had to be readjusted. As mentioned in section 4.2.1, a set of researched parameters were unsuccessful at controlling the Mini Talon airframe. The controllers were initially tuned using the Zeigler-Nichols tuning approach. The control parameters were then adjusted manually through an iterative process. With the theoretical knowledge that increasing proportional control decreases rise time and increases overshoot of a controller's response; that increasing integral control eliminates steady state error, decreases rise time and increases overshoot; and that increasing derivative control decreases overshoot and settling time, an educated trial and error process was undertaken.

4.4.3 How do the results achieve the aims

As described in section 4.3.1, the final set of control parameters for the Mini Talon airframe are successful at correcting a variety of unstable flight modes. The safety net algorithm is also operable, and allows the Mini Talon to choose to land at a secondary landing zone as necessary.

These results are a step towards achieving the goal of ensuring, and increasing, the recoverability of NIWA's payload electronics. The tuned controllers ensure that the aircraft is able to recover to stability during the initial falling phase at high altitude. The controllers also enable the Mini Talon to navigate, and land at, various landing zones. The safety net algorithm provides a backup in-case the Mini Talon's wind penetration is insufficient to return to the launch site. This helps to increase recoverability of NIWA's payloads, by ensuring that the aircraft a variety of landing zones to safely land without the risk of hitting obstacles. The safety net means that no matter where the solution lands, it will always be nearby people in contact with NIWA, thus can easily be returned via postage.

The benefit of the safety net algorithm can be quantified by figure 55. By assuming that the safety net algorithm increases the radius in which a solution can be considered recovered around Lauder, the probability of recovery can be plotted against landing constellation radius. The landing constellation radius is the radius of an assumed circular space around Lauder in which secondary landing zones exist. The landing zones are assumed to be located such that recovery within the circular space is guaranteed. In essence, the safety net algorithm reduces the distance to be travelled between a balloon burst location and a landing zone, whilst still being recoverable.

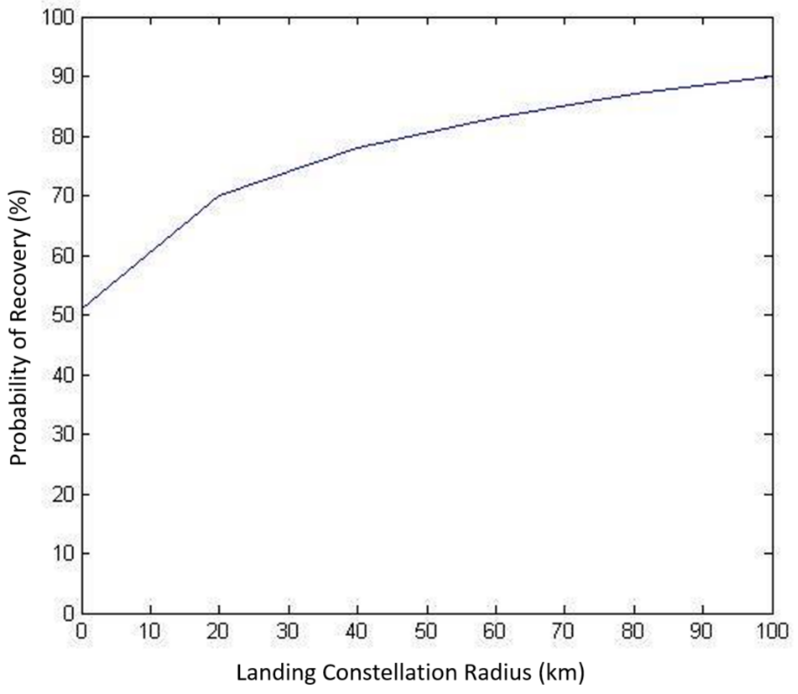


Figure 55. Probability of recovery with increasing landing constellation radius.

The figure above is based on the modelling performed in section 2.3, specifically how figure 25 changes as the distance travelled by the solution decreases (as the landing constellation radius increases). This is then related to the cumulative distribution function, figure 28, by plotting the probabilities with respect to landing constellation radius instead of wind speed.

From figure 55, it can be seen that on average, fifty percent of payloads can be recovered without the need for the safety net algorithm. As the landing constellation radius increases, the safety net algorithm enables the probability of recovery to increase. For example, a landing constellation, with secondary landing zones up to fifty kilometres of Lauder, results in a recovery probability of eighty percent.

5. Conclusion

An analysis was conducted into the expected atmospheric operating conditions and the drift behaviour of NIWA weather balloons. A flight model was developed to emulate the expected behaviour of a small airframe in a flight from 30km. Parameter Identification was used to fit the flight model to data recorded in flight testing, however this required an assumption of angle of attack. The fitted flight model was used to estimate what gliding distance could be expected, from a release point of 30 kilometres, and was extended to investigate how this could be extended with the use of a motor. It was also observed, that loading the airframe to increase its weight would increase its flying speed, making it more resilient to the detrimental effects of flying into a headwind, which supported the use of a motor and large battery.

Based on the expected travel distance of the mini-Talon airframe with a motor, it was found that an expected recovery rate of 30% of NIWA payloads could be achieved with an automatic recovery vehicle. This followed an assumption that NIWA will only launch the more expensive payload in conditions where successful recovery can be deemed likely. However the use of secondary recovery points close to lauder would allow NIWA to launch the self-recovering payload with less certainty of recovery to the point-of-launch, which would increase the expected recovery rate.

An investigation was made into what variety of airframe could be deemed suitable for application as a self-recovering payload, and whether an existing hobby grade airframe would be suitable. The main considerations were having a fuselage that could house all of the sounding equipment currently used on NIWA's balloons as well as the required control system. It was also determined that an airframe with a tail-plane would be beneficial as the tail fins would provide stability at high altitudes, where the low air density is such that controlling the craft becomes difficult. With the identification that oncoming winds would also be a significant challenge to the operability of a self-recovering payload, it was determined that minimizing drag would be an important factor. One of the primary points of consideration related to the CAA unmanned aerial vehicle rules [ref], which require that an unmanned balloon launch must weigh less than 4kg to be launched without a notice to all airmen, which is an important requirement for NIWA, as the exact launch time for their balloons cannot be reliably predicted. With the weight of the weather balloon, the airframe weight is limited to 2kg.

From the flight tests conducted it was found that the mini-Talon airframe performed extremely well, and under automatic control was very stable. The airframe also had plenty of space for the required electronic payload and even with a motor and battery, meets the CAA weight requirements. The mini-Talon airframe is a highly recommended candidate for the continued development self-recovering payload.

The team also investigated the use of readily available open-source flight controllers for the self-recovering payload. It was found that the PIXHAWK flight controller, with several peripherals could control the craft extremely reliably at low altitudes. With a comprehensive tuning process it was found that the controller was capable of recovering the airframe from unstable modes of flight, which is a significant finding for justifying the reliable operation of the solution from a legal standpoint. The PIXHAWK could also perform recovery-site prioritization, which would enable the use of multiple recovery sites, and potentially drastically increase the expected recovery rate of NIWA payloads.

Out investigation has found that readily available componentry can be used to produce an airframe that shows promise as a high altitude recovery vehicle, and that 30% to 50% of NIWA payloads are expected to be recoverable with such a vehicle.

6. Sustainability Analysis

This project was created with sustainability in mind. A big part of the operations of any company today is sustainable operations. With NIWA, the current methods for launching radiosondes and ozonesondes are unsustainable, costing NIWA large amounts of money, and polluting the environment with unrecovered electronics. Our solution solves these issues by returning the ozonesondes and other such equipment to the earth safely, and to a known location. Firstly, a triple bottom line analysis will be performed, followed by a life cycle analysis of the solution as it currently stands.

Triple Bottom Line Analysis

This analysis will look into the three issues that form the triple bottom line. Economic, environmental, and finally social impacts will be investigated, and the solution that has been developed will be compared against the current method that NIWA uses for their payloads.

Economic

The current system that NIWA uses to launch and recover their radiosonde packages consists of around $800 worth of sensors, radios, and other electronics. With only 20% of these packages being returned to the NIWA station in Lauder, these launches are very expensive, as there is usually nothing wrong with the actual electronics, and they can be reused.

With this in mind, the solution that we developed was limited to $1000, to keep the number of packages recovered in order to break even to a minimum. The following graph shows the number of packages recovered vs the amount saved. As shown, the number of packages successfully recovered in order to break even is just two. This means that the target of a cheap recovery system has been met, graphically shown in Figure 56 below.

In addition to this, NIWA will be able to launch more expensive payloads regularly, if the solution's actual recovery rate is sufficient enough to break even with each solution purchased. This cost saving comes in the form of less units that will need to be manufactured for NIWA. Figure 56 shows a cash flow diagram based on the number of successful payload recoveries.

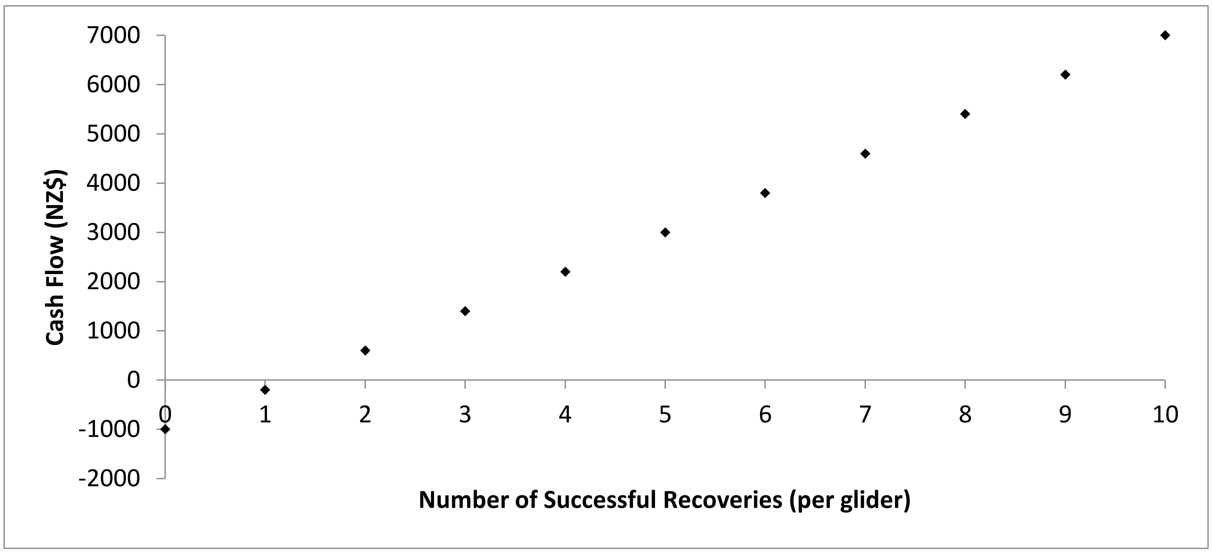


Figure 56 - Cash flow directly based on number of successful payload recoveries.

Environmental

The main part of the environmental analysis is the actual package itself. No part of the current packages launched are biodegradable. This results in polystyrene packages along with metals, epoxies and battery chemicals (lithium polymers)being released into the environment.

A significant proportion of launched payloads currently fall into the ocean, which is an undesirable result, especially with the materials that these packages contain. The rest end up on land, albeit in an unknown location. This means that and wildlife could come into contact with hazardous substances, causing damage to their health.

Furthermore, this results in a further economic benefit for NIWA due to a reduction in manufacturing costs. With less radiosondes and other such electronics being required, the amount of hazardous waste from the manufacturing processes is greatly reduced. This also applies to the LiPo batteries, as using rechargeable batteries will significantly reduce the number of single use batteries ending up in landfills, or the environment.

Social

The final part of the triple bottom line analysis is the social aspects. The current solution is marked with wording to indicate that it is not a bomb and requests that the finder return payloads to NIWA. This is because a payload was once mistakenly identified as an explosive by a member of the public. Although there is a reward for doing so, package return places an unnecessary burden on the people that find launched payloads. Our solution will remove the need to do this.

Another aspect to the social side is closely related to the environmental issues. NIWA is a crown research institute, and so it can come under scrutiny by taxpayers and politicians alike. With our solution however, it can be noted by NIWA that they are making efforts to become more sustainable, not just with taxpayers money, but with regards to the environment also.

Furthermore, with more expensive, and different hardware likely to be launched on a regular basis, conditional to high altitude success, NIWA and other institutes will be able to collect more information about the atmosphere and this will give a better understanding of how the environment is changing over time.

Life cycle analysis

There are of course various materials and electronics associated with our design and solution. The airframe is made of expanded polyolefin (EPO) foam, which is recyclable. The airframe will have a much shorter lifespan than that of the electronics that run the plane. A 10,000mAh battery that provides the power for the last part of the journey has the next shortest lifespan, at somewhere up to 500 charge/recharge cycles.

Otherwise, barring some type of premature failure, the other electronics should last for 10 years. After that, they can be recycled at an electronics recycling facility.

All of the components that make up the plane are recyclable to some extent. Another point to note is that the plane is a fully modular solution, meaning that if there is a failure of any component, it can be replaced without needing another complete system.

7. Recommendations/Implications/Applications

Since the aircraft has only been testing at low altitudes, it is now necessary to perform higher altitude testing. An initial recommended starting point for higher altitude testing starting from 1km, as a small scale simulation of a thirty kilometre drop. NIWA own and operate a one kilometre tethered balloon rig which can be used to perform these tests. This would also be beneficial in testing the release mechanism of the solution.

Providing that one kilometre testing is successful, thirty kilometre full altitude tests could be performed. This would involve attaching the solution to an untethered weather balloon, installing a working radiosonde and ozonesonde into the solution and creating a flight plan with a constellation of secondary landing locations built in. This test would be used to determine some key unknowns about the solution. These unknowns include how the electronics will react in both the temperature encountered at 30km and the speeds at which the solution will reach during the low air density portion of the descent.

The safety net algorithm could be refined in future by the addition of extra hardware. In particular, range finding equipment on the front and bottom of the airframe would enable the solution to detect obstacles when landing. This would enable the solution to automatically avoid collisions, and could reduce the need for secondary landing zones. Furthermore, GPS tracking hardware could be fitted into the solution to provide a means of locating the payload if secondary landing zones become redundant.

References

[1] F. Ivis, “Calculating Geographic Distance: Concepts and Methods,” Canadian Institute for Health Information, Toronto, 2006.

[2] “Gravity Changes with Altitude,” [Online]. Available: https://www.mansfieldct.org/Schools/MMS/staff/hand/lawsgravaltitude.htm. [Accessed 16 10 2015].

[3] M. Cavcar, “The International Standard Atmosphere,” [Online]. Available: http://home.anadolu.edu.tr/~mcavcar/common/ISAweb.pdf. [Accessed 15 10 2015].

[4] Civil Aviation Authority NZ, “Part 101,” 24 09 2015. [Online]. Available: https://www.caa.govt.nz/rules/Rule_Consolidations/Part_101_Consolidation.pdf. [Accessed 16 10 2015].

[5] Civil Aviation Authority NZ, “Part 102,” 24 09 2015. [Online]. Available: https://www.caa.govt.nz/rules/Rule_Consolidations/Part_102_Consolidation.pdf. [Accessed 16 10 2015].

[6] R. Garcia and L. Barnes, “Multi-UAV Simulator Utilizing X-Plane,” *Published online,* vol. Journal of Intelligent and Robotic Systems, no. 57, pp. 393-406, 2010.

[7] L. Oliveira, R. Ribeiro and M. F. Neusa, “UAV Autopilot Controllers Test Platform,” *40th ASEE/IEEE Frontiers in Education Conference,* 2010.

[8] N.-D. Access, NOAA, 2015. [Online]. Available: http://www.ncdc.noaa.gov/data-access. [Accessed 15 10 2015].

[9] T. A. Johansen, A. Cristofaro, K. Sørensen, J. M. Hansen and T. I. Fossen, “On estimation of wind velocity, angle-of-attack and sideslip,” Centre for Autonomous Marine Operations and Systems, Department, Trondheim, Norway, 2015.

[10] J. W. Langelaan, N. Alley and J. Neidhoefer, “Wind Field Estimation for Small Unmanned Aerial,” *AIAA Guidance, Navigation and Control Conference, Toronto, Canada,* p. 21, 2010.

[11] J. Perry, D. A. Mohamed, B. Johnson and D. R. Lind, “Estimating Angle of Attack and Sideslip Under High Dynamics on Small UAVs,” *ION GNSS 21st. International Technical Meeting of the,* 208.

[12] Dragon Circuits, “Sparky Flight Controller,” Dragon Circuits, 2015. [Online]. Available: http://dragoncircuits.com/shop/sparky/. [Accessed 15 10 2015].

[13] 3D Robotics, “APM 2.6+,” 3D Robotics, [Online]. Available: https://store.3drobotics.com/products/apm-2-6-kit-1?taxon_id=42. [Accessed 13 05 2015].

[14] 3D Robotics, “3DR Pixhawk,” 3D Robotics, [Online]. Available: https://store.3drobotics.com/products/3dr-pixhawk. [Accessed 13 05 2015].

[15] 3D Robotics, “3DR uBlox GPS with Compass,” 3D Robotics, [Online]. Available: https://store.3drobotics.com/products/3dr-gps-ublox-with-compass. [Accessed 13 05 2015].

[16] 3D Robotics, “Pixhwak Airspeed Sensor Kit,” 3D Robotics, [Online]. Available: https://store.3drobotics.com/products/pixhawk-airspeed-sensor-kit. [Accessed 13 05 2015].

[17] 3D Robotics, “3DR Radio Set,” 3D Robotics, [Online]. Available: https://store.3drobotics.com/products/3dr-radio-set. [Accessed 13 05 2015].

[18] openpilot, “OpenPilot.org,” [Online]. Available: https://www.openpilot.org/. [Accessed 14 10 2015].

[19] APM Plane, “Plane | Fixed-wing aircraft UAV,” 3D Robotics, [Online]. Available: http://plane.ardupilot.com/. [Accessed 16 10 2015].

[20] ardupilot, “Mission Planner | Ground Station,” 3D Robotics, [Online]. Available: http://planner.ardupilot.com/. [Accessed 15 10 2015].

[21] TAU LABS, “Tau Labs,” Tau Labs, 2014. [Online]. Available: http://taulabs.org/. [Accessed 17 10 2015].

[22] GPS Boomering Ltd., “About Us,” GPS Boomering Ltd., 2005. [Online]. Available: http://www.gpsboomerang.com/content/view/19/33/. [Accessed 2015].

[23] J. Schepige, J. Tate and W. Schoenfeld, “Creation of Western Oregon University Weather Ballooning Program: GPS Tracking, Heating Circuitry, and Balloon Filling Proceedure,” Oregon State University, Corvallis, 2013.

[24] G. Selvaraj, “Temperature Controlled System,” Engineers Garage, 2012. [Online]. Available: http://www.engineersgarage.com/contribution/experts/temperature-controlled-system. [Accessed 2015].

[25] 3DRobotics, “Roll, Pitch and Yaw Controller Tuning,” 3DRobotics, [Online]. Available: http://plane.ardupilot.com/wiki/roll-pitch-controller-tuning/. [Accessed 2015].

Appendices

```python
"""
SONDE.PY

Created on Thu Apr 09 09:54:00 2015

@author: Sam

 Visualise NIWA flight data
"""



from logs import *
from functions import *
import numpy as np
import matplotlib.pyplot as plt
from scipy.misc import imread
import csv

import matplotlib


from figformat import *


prefix = "sonde_fig."

plt.close('all')



#           GLOBALS
log_files =  [ f for f in listdir('DATA/') if isfile(join('DATA/',f)) ]



# Look at what data is present in the logs
def logInfo():
    keyOccurences = {}
    files = [ f for f in listdir('DATA/') if isfile(join('DATA/',f)) ]
    print len(files), " log files available"
    for file in files:
        for i, line in enumerate(open('DATA/' + file)):
            if i == 2:
                for item in readKeys(line):
                    if item in keyOccurences:
                        keyOccurences[item] += 1
                    else:
                        keyOccurences[item] = 1
    outKeys = sorted(keyOccurences.keys())
    for outKey in outKeys:
        print outKey, '\t', keyOccurences[outKey]



# plot height versus time
def heightTime(logs):
    fig = plt.figure(figsize = plotSize, dpi = plotdpi)
    for log in [logs[i] for i in range(len(logs)) if i != 105]:     # comprehension
 gets rid of dodgy one.
        plt.plot(np.divide(log['Time'],3600), log['GMH'])
    plt.axis([0, 16000/3600, 0, 45000])
    plt.ylabel('Height (m)')
    plt.xlabel('Time (hr)')
#    plt.title('Balloon Ascent')
    plt.grid('on')
    plt.tight_layout()
    plt.show()
    fig.savefig('./Figures/' + prefix + '1.1. height-time.png')



# plot distance versus time
def distanceTime(logs):
    fig = plt.figure(figsize = plotSize, dpi = plotdpi)
    for i, log in enumerate(logs):
        if 'PLat' in log:
            #create displacement vector from coords
            displacement= LatLon2Dis(log)
            plt.plot(np.divide(log['Time'], 3600), clean(displacement))
    #print max_dist
    plt.axis([0, 3.5, 0, 350])
#    plt.title('Balloon Displacement')
    plt.xlabel('Time (hr)')
    plt.ylabel('Displacement (Km)')
    plt.grid('on')
    plt.show()
    plt.tight_layout()
    fig.savefig('./Figures/' + prefix + '1.2. distance-time.png')

def heightDistance(logs):
    fig = plt.figure(figsize = plotSize, dpi = plotdpi)
    for i, log in enumerate(logs):
        if 'PLat' in log:
            #create displacement vector from coords
            displacement= LatLon2Dis(log)
            plt.plot(clean(displacement), log['GMH'])
#    plt.title('Balloon Ascent Profile')
    plt.ylabel('Height (m)')
    plt.xlabel('Horizontal Displacement, (Km)')
    fig.savefig('./Figures/' + prefix + '1.3. height-distance.png')

# plot windspeed as a function of altitude
def windAltitude(logs):
    fig = plt.figure(figsize = plotSize, dpi = plotdpi)
    for log in logs:
        if 'WSpeed' in log and 'GMH' in log:
            plt.plot(log['GMH'], clean(cleand(log['WSpeed'], 3), 100), 'b', alpha = 0.03)
#    plt.title('Windspeed')
    plt.xlabel('Altitude (m)')
    plt.ylabel('Wind Speed, (m/s)')
    plt.grid('on')
    plt.tight_layout()
    plt.axis([0, 30000, 0, 120])
    fig.savefig('./Figures/' + prefix + '4. windAltitude.png')


# plot windspeed as a function of altitude
def windAltitudeSUBSET(logs):
    fig = plt.figure(figsize = plotSize, dpi = plotdpi)
    for log in logs[12::80]:
        if 'WSpeed' in log and 'GMH' in log:
            plt.plot(log['GMH'], clean(cleand(log['WSpeed'], 3), 100))
#    plt.title('Windspeed')
    plt.xlabel('Altitude (m)')
    plt.ylabel('Wind Speed, (m/s)')
    plt.tight_layout()
```

```python
        plt.grid('on')
        plt.axis([0, 30000, 0, 120])
        plt.ylim([0, 80])
        fig.savefig('./Figures/' + prefix + '4. windAltitudeSUBSET.png')



    # plot the direction of wind travel asa function of altitude
    # NOTE: demonstrates very little in the way of a trend whether there is indeed
    # one their or not
    def windDIR(logs):
        fig = plt.figure(figsize = plotSize, dpi = plotdpi)
        for log in logs:
            if 'WDirec' in log:
                plt.plot(clean(log['GMH']), clean(log['WDirec']), 'b')
        plt.show()
        fig.savefig('./Figures/' + prefix + '-1. windDir.png')



    # plot the travel paths of the balloons
    def latlon(logs, plotPaths = False, plotEndpoints = True):
        fig = plt.figure(figsize = plotSize, dpi = plotdpi)

        #background image
        img = imread("./Figures/GPS_back.png")
        plt.imshow(img,zorder=0, extent = [162.98, 177.63, -46.65, -42.33], aspect = 'auto')

        for log in logs:
            if 'PLat' in log:
                if plotPaths:
                    plt.plot(clean(log['PLong']), clean(log['PLat']), '')
                if plotEndpoints:
                    plt.plot(clean(log['PLong'])[-1], clean(log['PLat'])[-1], 'w.')

        plt.axis([167, 174, -46.1, -43.5])
        plt.xlabel('Longitude')
        plt.ylabel('Latitude')
        plt.grid('on')
        plt.tight_layout()

        if plotPaths:
    #        plt.title('NIWA Balloon Burst Points')
            fig.savefig('./Figures/' + prefix + '1.5. paths.png')
        else:
    #        plt.title('NIWA Balloon Burst Points')
            fig.savefig('./Figures/' + prefix + '1.5. endpoints.png')


    def latlonPaths(logs):
        latlon(logs, plotPaths = True, plotEndpoints = False)



    # plot distance traveled versus  average windspeed experience over a launch

    def avewindDistTraveled(logs):
        fig = plt.figure(figsize = plotSize, dpi = plotdpi)
        max_dist = []
        wspeed_ave = []
        for i, log in enumerate(logs):
            if 'PLat' in log and 'WSpeed' in log:
                displacement = LatLon2Dis(log)
                displacement = clean(displacement)
                max_dist.append(max(displacement))
                wspeed_ave.append(sum(clean(log['WSpeed']))/len(log['WSpeed']))

        plt.plot(wspeed_ave, max_dist, '.')
    #    plt.title('Relating Distance Traveled with Average Windspeed')
    #    plt.title('NIWA Balloon Wind Drift')
        plt.ylabel('Distance traveled (km)')
        plt.xlabel('Average Wind Speed (m/s)')
        plt.axis([0, 50, 0, 350])
        plt.grid('on')
        plt.tight_layout()

        #trendline
    #    d = np.arange(0,50,1)
    #
    ##    a = 200.0/25.5
    #    a = 7.233
    ##    b = -13
    #    b = 0
    #    p = d*a + b
    #    trend = plt.plot(d, p, 'g')
    #    p = d*100/20
    #    plt.plot(d, p, 'r')


    #    plt.legend(trend, ['y = 7.843x'], 'upper left')
    #

        plt.savefig('./Figures/' + prefix + '2.0. avewind-distTraveled.png')



    # plot distance traveled versus  average windspeed experience over a launch
    def avewindDistTraveled2(logs):
        max_dist = []
        wspeed_ave = []
        for i, log in enumerate(logs):
            if 'PLat' in log and 'WSpeed' in log:
                displacement = LatLon2Dis(log)
                displacement = clean(displacement)
                max_dist.append(max(displacement))
                wspeed_ave.append(sum(clean(log['WSpeed']))/len(log['WSpeed']))



        fig = plt.figure(figsize = np.multiply(plotSize, [1, 1.5]), dpi = plotdpi)
    #    avewindDistTraveled(logs)
        int_dist = []
        int_dist_corrected = []
        wspeed_ave = []
        wspeed_ave_corrected = []
        for i, log in enumerate(logs[::]):
            if 'PLat' in log and 'WSpeed' in log:
                int_dist.append(np.trapz(clean(log['WSpeed']), log['Time']))
                int_dist_corrected.append(np.trapz(clean(log['WSpeed']), log['Time'])/(log['Time'][-1]/(1.6*3600)))
                wspeed_ave.append(sum(np.abs(clean(log['WSpeed'])))/len(log['WSpeed']))#*log['Time'][-1]/(2*3600))


        ax = plt.subplot2grid((3, 1), (0, 0), rowspan = 2)
```

```
        plt.plot(wspeed_ave, max_dist, '.', axes = ax)#, alpha = 0.25)
        plt.plot(wspeed_ave, np.array(int_dist)/1e3, 'c.', axes = ax)#, alpha = 0.25)
    #    plt.plot(wspeed_ave, np.array(int_dist_corrected)/1e3, 'c.')
        plt.grid('on')
    #    plt.title('Relating Distance Traveled with Average Windspeed')
        plt.ylabel('Distance traveled (km)')
        plt.xlabel('Average Wind Speed (m/s)')
        art = []
    #    lgd =plt.legend(['measured displacement', 'Corrected for wind direction', 'corrected for ascent rate'],bbox_to_anchor=(1, 1.1))
        lgd =plt.legend(['Original dataset', 'Corrected for wind direction'],bbox_to_anchor=(1, -0.3))
        art.append(lgd)
        plt.tight_layout()
    #    plt.axis([0, 50, 0, 350])

    #    z = np.polyfit(wspeed_ave, np.array(int_dist)/1e3, 1)
    #    p = np.poly1d(z)
    #
    #    trend = plt.plot(wspeed_ave, p(wspeed_ave))
    #    plt.legend(trend, ['y = %.3fx + %.3f'%(z[0], z[1])], 'upper left')

        fig.savefig('./Figures/' + prefix + '2.0. avewind-distTraveled (wind dir corrected normalised).png', additional_artists = art)



    # plot distance traveled versus  average windspeed experience over a launch

    def avewindDistTraveled3(logs):
        fig = plt.figure(figsize = np.multiply(plotSize, [1, 1.5]), dpi = plotdpi)

        max_dist = []
        max_dist_corrected = []
        wspeed_ave = []
        for i, log in enumerate(logs):
            if 'PLat' in log and 'WSpeed' in log:
                displacement = LatLon2Dis(log)
                displacement = clean(displacement)
                max_dist.append(max(displacement))
                max_dist_corrected.append(max(displacement)/(log['Time'][-1]/(1.6*3600)))
                wspeed_ave.append(sum(clean(log['WSpeed']))/len(log['WSpeed']))




    #    plt.title('Relating Distance Traveled with Average Windspeed')
    #    plt.title('NIWA Balloon Wind Drift')
        plt.ylabel('Distance traveled (km)')
        plt.xlabel('Average Wind Speed (m/s)')
        plt.axis([0, 50, 0, 350])
        plt.grid('on')




        # account wind variation
        int_dist = []
        int_dist_corrected = []
        wspeed_ave = []
        wspeed_ave_corrected = []
        for i, log in enumerate(logs[::]):
            if 'PLat' in log and 'WSpeed' in log:
                int_dist.append(np.trapz(clean(log['WSpeed']), log['Time']))
                int_dist_corrected.append(np.trapz(clean(log['WSpeed']), log['Time'])/(log['Time'][-1]/(1.6*3600)))
                wspeed_ave.append(sum(np.abs(clean(log['WSpeed'])))/len(log['WSpeed']))#*log['Time'][-1]/(2*3600))


        ax = plt.subplot2grid((3, 1), (0, 0), rowspan = 2)
        plt.plot(wspeed_ave, max_dist, '.', axes = ax)
    #    plt.plot(wspeed_ave, np.array(int_dist)/1e3, 'r.', axes = ax)
        plt.plot(wspeed_ave, max_dist_corrected, 'm.', axes = ax, alpha = 0.75)
    #    plt.plot(wspeed_ave, np.array(int_dist_corrected)/1e3, 'g.', axes = ax)
        plt.grid('on')
    #    plt.title('Relating Distance Traveled with Average Windspeed')
        plt.ylabel('Distance traveled (km)')
        plt.xlabel('Average Wind Speed (m/s)')
        art = []
    #    lgd =plt.legend(['measured displacement', 'Corrected for wind direction', 'corrected for ascent rate'],bbox_to_anchor=(1, 1.1))
        lgd =plt.legend(['Original dataset', 'Corrected for ascent time'],bbox_to_anchor=(1, -0.3))
        art.append(lgd)
        plt.tight_layout()
    #    plt.axis([0, 50, 0, 350])

    #    z = np.polyfit(wspeed_ave, np.array(int_dist)/1e3, 1)
    #    p = np.poly1d(z)
    #
    #    trend = plt.plot(wspeed_ave, p(wspeed_ave))
    #    plt.legend(trend, ['y = %.3fx + %.3f'%(z[0], z[1])], 'upper left')

        fig.savefig('./Figures/' + prefix + '2.0. avewind-distTraveled (ascent time normalised).png', additional_artists = art)



    # genrate 10th percentile envelopes for windspeed plot as a means to profile worst possible expected wind conditions
    def windAltStat(logs):
        print 'windaltsat'

        fig = plt.figure(figsize = plotSize, dpi = plotdpi)

        bin_width = 1250
        bin_sep = bin_width
        bins = [[] for i in range(25)]


        for log in logs:
            if 'WSpeed' in log and 'GMH' in log:
                # add it to the plot
                plt.plot(log['GMH'], clean(cleand(log['WSpeed'], 5)), '0.8')

                #put data points in bins
                for i in range(len(log['GMH'])):
                    for j in range(0,25):
                        #print j*bin_width-bin_sep/2
                        if log['GMH'][i] < j*bin_width+bin_sep/2:
                            bins[j].append(log['WSpeed'][i])
                            break

        # find average wnidspeed in each bin
        # (not related to the enevelopes, this origionally plotted average windspeed)
```

```
#    averages = []
#    for bin_ in bins:
#        if bin_:
#            averages.append(float(sum(bin_))/len(bin_))

#    plot average
#    print len(averages)
#    plt.plot(range(0, bin_width*len(averages), bin_width), averages, 'r')


    # find envolopes for every 5th percentile
    max_envolopes = [[] for i in range(20)]
    ave_envolopes = [[] for i in range(20)]
    for bin in bins:
        if bin:
            for i in range(20):
              max_envolopes[i].append(np.max(sorted(bin)[0:int((i+1)*float(len(bin)/20.0))]))
              ave_envolopes[i].append(np.average(sorted(bin)[int((i)*float(len(bin)/20.0)):int((i+1)*float(len(bin)/20.0))]))




    for i, env in enumerate(max_envolopes):
        if i%2 == 0: # Plot 5th percentile envelopes as dotted
            pa = plt.plot(range(0, bin_width*len(env), bin_width), env, 'b:')
        else:
            pb = plt.plot(range(0, bin_width*len(env), bin_width),env, 'b')


#    pa.set_label('5th percentile')
#    pb.set_label('10th percentile')
#
#    plt.legend()


#    plt.title('Windspeed Distribution (envelopes)')
    plt.xlabel('Altitude(m)')
    plt.ylabel('WindSpeed, (m/s)')
    plt.grid('on')
    plt.tight_layout()

    plt.axis([0, 30000, 0, 100])

    fig.savefig('./Figures/' + prefix + '2.2. windStat.png')

    with open('max_envelopes.csv', 'wb') as envelopes_out:
        writer = csv.writer(envelopes_out)
        for envolope in max_envolopes:
            writer.writerow(envolope)

    with open('ave_envelopes.csv', 'wb') as envelopes_out:
        writer = csv.writer(envelopes_out)
        for envolope in ave_envolopes:
            writer.writerow(envolope)






# Generate a cumulatie distrobution function of travel distance for data set

def distanceCDF(logs):
```

```python
        fig = plt.figure(figsize = plotSize, dpi = plotdpi)
        max_dist = []
        log_nums = []
        wind_speeds = []
        for i, log in enumerate(logs):
            if 'PLat' in log:
                #create displacement vector from coords
                displacement = LatLon2Dis(log)
                displacement = clean(displacement)
                max_dist.append(max(displacement))
                log_nums.append(i)
                wind_speeds.append(log['WSpeed'])
        # do statistical analysis on max dist
        cdf = []
        # for all values of D, find what percentage of balloons travel a distance
        # less than D.
        D_range = range(0,275)
        for D in D_range:
            c = 0
            for dist in sorted(max_dist):
                if dist < D:
                    c += 1
                else:
                    break
            cdf.append(float(c)/len(max_dist))

        plt.plot(np.array(D_range), cdf)
    #    plt.title('Cumulative distrobution function of burst displacement')
        plt.xlabel('Distance Traveled before bursting (KM)')
        plt.ylabel('CDF(burst distance)')
        plt.grid('on')
        plt.tight_layout()
    #    plt.xlim2([0, 275])
        fig.savefig('./Figures/' + prefix + '2.0 distCDF.png')


    # Generate a cumulatie distrobution function of travel distance for data set

    def avewindspeedCDF(logs):
        fig = plt.figure(figsize = plotSize, dpi = plotdpi)
    #    distanceCDF(logs)
        max_dist = []
        log_nums = []
        ave_winds = []
        for i, log in enumerate(logs):
            if 'WSpeed' in log:

                log_nums.append(i)
                ave_winds.append(np.mean(clean(log['WSpeed'])))

        # do statistical analysis on max dist
        cdf = []
        D_range = np.arange(0,40,0.5)
        for D in D_range: # Still calling it D, but don't need to go as far
            c = 0
            for wsp in sorted(ave_winds):
                if wsp < D:
                    c += 1
                else:
                    break
            cdf.append(float(c)/len(ave_winds))

        plt.plot(np.array(D_range), cdf)
    #    plt.title('Cumulative Distrobution Function of Average Windspeed (NIWA Data)')
        plt.xlabel('Average Windspeed (m/s)')
        plt.ylabel('CDF(Average Windspeed)')
        plt.grid('on')
        plt.tight_layout()
        fig.savefig('./Figures/' + prefix + '2.0 avewindCDF.png')




    # Run plots with every log file
    # sequentially plots chunks of the total set of logs to stop overflows
    def plotAll(logs):
        heightTime(logs)
        distanceTime(logs)
    #    heightDistance(logs)
        windAltitude(logs)
        windAltitudeSUBSET(logs)
        windAltStat(logs)
    #    windDIR(logs)
        latlon(logs)
    #    latlonPaths(logs)
        avewindDistTraveled(logs)
    #    windAltStat2(logs)
        distanceCDF(logs)



    if __name__ == '__main__':
        print 'sonde.py'
        if not 'logs' in  locals():
            logs = readAllLogs()

    #    windAltitude(logs)

    #    latlon(logs)
    #    latlonPaths(logs)

    #    avewindspeedCDF(logs)
    #    avewindspeedCDF(logs)
    #    distanceCDF(logs)

    #    avewindDistTraveled(logs)
        avewindDistTraveled2(logs)
        avewindDistTraveled3(logs)

    #    windAltitudeSUBSET(logs)
    #
    #    heightTime(logs)
    #    windAltStat2(logs)

    #    windAltStat(logs)


    #    plotAll(logs)
    #    print 'Completed'


    #else:
    #    if not 'logs' in  locals():
    #        logs = readAllLogs()
```

```
% Identify Coefs.m
% derive aerodynamic coefficients assuming a fixed angle of attack.

clear
% clc
close all


% Flight data log file
load('2015-08-14 09-21-58.log-348556.mat')

T0 = AHR2(1, 2);
lat = timeseries(AHR2(:, 7), AHR2(:, 2)-T0);
lat.TimeInfo.Units = 'milliseconds';

lon = timeseries(AHR2(:, 8), AHR2(:, 2)-T0);
lon.TimeInfo.Units = 'milliseconds';

Galt = timeseries(AHR2(:, 6), AHR2(:, 2)-T0);
Galt.TimeInfo.Units = 'milliseconds';


roll = timeseries(AHR2(:, 3), AHR2(:, 2)-T0);
roll.TimeInfo.Units = 'milliseconds';

pitch = timeseries(AHR2(:, 4), AHR2(:, 2)-T0);
pitch.TimeInfo.Units = 'milliseconds';

yaw = timeseries(AHR2(:, 5), AHR2(:, 2)-T0);
yaw.TimeInfo.Units = 'milliseconds';

Aspd = timeseries(NTUN(:, 8), NTUN(:, 2)-NTUN(1, 2));
Aspd.TimeInfo.Units = 'milliseconds';
Aspd = resample(Aspd, lat.Time);

Gspd = timeseries(GPS(:, 11), GPS(:, 3)-GPS(1, 3));
Gspd.TimeInfo.Units = 'milliseconds';
Gspd = resample(Gspd, lat.Time);

% look at how changing the assumption of angle of attack changes
% the calculated coefficients
AOAs = [10, 0];

for s = 1:2

if s == 1
    ind = find(Galt.Time > 3.585e5 & Galt.Time < 3.615e5);
    start = ind(1)+19
    stop = ind(end)-3
else

    start = 4697+39;
    stop = 4776-20-12;

end

u_x = Aspd.Data(start:stop);
v_g = Gspd.Data(start:stop);
```

```
    T = AHR2(start:stop-1, 2)-AHR2(start, 2);

    T = T/1000;
    z = Galt.Data(start:stop-1);
    for i = start+1:stop
        x(i-start) = pos2dist(lat.Data(i), lon.Data(start),
lat.Data(start), lon.Data(start), 1)'*1e3;
        y(i-start) = pos2dist(lat.Data(start), lon.Data(i),
lat.Data(start), lon.Data(start), 1)'*1e3;
    end

    a = roll.Data(start:stop);
    b = pitch.Data(start:stop)
    c = yaw.Data(start:stop);

    x = x';
    y = y';

    dx = gradient(x, T);
    dy = gradient(y, T);
    dz = gradient(z, T);

    g = 9.80665;    % gravitational acceleration (m/s2)

    theta = b;
    v_k = u_x;
    aoa = AOAs(s);

    theta = deg2rad(mean(theta));       % pitch

    v_k = mean(v_k/cos(deg2rad(aoa)));  % Kinetic speed


    M = 1.460;    % measured mass of the plane at time of flight, not the
2kg that flight will be later modelled with

    R = [cos(theta), -sin(theta); sin(theta), cos(theta)]; % Rotation
matrix
    Fg = R*[0; g*M];

    SigF = Fg + M*[mean(diff(u_x)); 0];

    % Aerodynamic forces = 1/2 * rho * v^2 * A * [Cd; Cl]

    rho_0 = 1.2250;  % air density at sea level

    C(s, :) = [1 0;0 1]*SigF * 2 / rho_0 / norm(v_k)^2 / 1; % aerodynamic
coeffieicnets for reference area of 1

    E = C(s, 2)/C(s, 1);

    aoa

    end

    % mean(C)
```

C

```python
# -*- coding: utf-8 -*-
"""
STEADY_STATE_FLIGHT.PY

Created on Thu Sep 03 11:56:21 2015

@author: Sam

    Use the measured aerodynamic coefficients to model flight
    unnaffected by wind
"""


import matplotlib

from functions import *
import numpy as np
import matplotlib.pyplot as plt


from figformat import *


prefix = 'fly_norm.'

m2k = 3.6

rho_0 = 1.2250 # air density at sea level
T_0 = 288.15 # temperature at sea level
g = 9.80665 # Acceleration due to gravity
M = 0.0289644 # Molar mass of air on earth (Kg/mol)
R = 8.31432 # Universal gas constant for air

# Airframe parameters
m = 2 # Mass (kg)

#for CdAd, ClAl in [[0.0212, 0.0625], [0.090, 0.0674], [0.0428, 0.0549]]:

if __name__ == '__main__':
    CdAd = 0.0212
    ClAl = 0.0625

#
#CdAd = 0.0212
#ClAl = 0.0625


# Simulate accross a range of altitudes
h_max = 30E3
h_step = 100
h = h_max - np.arange(-0.0000001, h_max, h_step)


# Air density model
rho = rho_0*np.exp(-g*M*h/R/T_0)


# estimate flight speed

#glide slope
theta = np.arctan(CdAd/ClAl)

K = ClAl*np.cos(theta) + CdAd*np.sin(theta)
```

```python

a = np.sqrt(2*m*g/rho/K) # airspeed


################    AIR DENSITY AND VELOCITY IN TERMS OF ALTITUDE


fig = plt.figure(1, figsize = plotSize, dpi = plotdpi)
plt.subplot(2, 1, 1)
#plt.title('Airspeed / Air Density Relationship')
plt.plot(h, rho)
plt.ylabel('air density (kg/m^3)')

plt.xlim([0, h_max])
plt.gca().invert_xaxis()
plt.gca().set_ylim(bottom = 0)
plt.grid('on')


plt.subplot(2, 1, 2)
plt.plot(h, a)
plt.ylabel('airspeed (m/s)')
plt.xlabel('Altitude (m)')

plt.xlim([0, h_max])
plt.gca().invert_xaxis()
plt.gca().set_ylim(bottom = 0)
plt.grid('on')
plt.tight_layout()
fig.savefig('./Figures/' + prefix + '10.0. velocity-air density.png')

plt.show()

################     LARGER PLOT JUST OF AIR DENSITY FOR BRODIE

fig = plt.figure(figsize = plotSize, dpi = plotdpi)
plt.plot(h, rho)
plt.ylabel('air density (kg/m^3)')

plt.xlim([0, h_max])
plt.gca().invert_xaxis()
plt.gca().set_ylim(bottom = 0)
plt.grid('on')
plt.ylabel('air density (kg/m^3)')

plt.xlabel('Altitude (m)')


fig.savefig('./Figures/' + prefix + '10.1. air density.png')
```

```python
# -*- coding: utf-8 -*-
"""
HEADWIND_FLIGHT_UNCONTROLLED.PY

Created on Thu Sep 03 12:27:22 2015

@author: Sam

Use the flight model in steady_state_filght.py and extend it to model the
detremental effects of flying into a headwind

"""

from functions import *
import numpy as np
import matplotlib.pyplot as plt
import matplotlib


plt.close('all')

# Lift and drag coefficients calculated in 'identify_coefs.m'
CdAd = 0.0212
ClAl = 0.0625

# An extension of steady_state_flight.py, has dependant variables
from steady_state_flight import *
from sonde import *


from figformat import *

prefix = 'fly_wind.'




envelopes = np.transpose(loadEnvelopes('wind.csv'))
envelopes = [envelopes[i] for i in range(len(envelopes)) if len(envelopes[i]) >= 30
1]
#get rid of the dodgy ones (should have been gotten rid of allready)
envelopes = [envelopes[i] for i in range(len(envelopes)) if np.mean(envelopes[i]) >
 0]


# airspeed at each point in altitude is defiend by the model,the vertical
# component of this can be used to find the time taken to traverse a step in altitu
de
t_H = h_step /  vsp
t_H = [0]+t_H[:]

# The above does not consider the effects of an initial acceleraion
# period, 'Falling.py' estimates the time taken to reach a steady state velocity
# The expected horizontal component of airspeed is set to zero for the
# acceleration period
import Falling
T_acel, vsp_acel = Falling.model_acceleration(CdAd)
# override the first 500m of decent times to simulate the acceleration period
t_H[0:len(T_acel)] = T_acel
gsp[0:len(T_acel)] = 0
vsp[0:len(T_acel)] = vsp_acel

# find distance traveled at each altitude step
d_H = np.multiply(t_H, gsp)
d_H[0] = 0
d_H[1] = 0
```

```python

# find distance carried at each altitude step due to wind drift
# essentially wind drift can be found by integrating windspeed accroos each
# altitude step for the time that would be spent there
d_H_w  = []
for i, env in enumerate(envelopes):
    if len(env) >= 301:
        d_H_w.append(np.multiply(t_H, env[:len(t_H)][::-1]))


##################      ALTITUDE AND LATERAL TRAVEL / TIME

fig = plt.figure(figsize = plotSize, dpi = plotdpi)

plt.subplot(2, 1, 1)
plt.plot(np.cumsum(t_H)/60, h/1e3)
plt.ylabel('Altitude (km)')
plt.grid('on')

plt.subplot(2, 1, 2)
plt.plot(np.cumsum(t_H)/60, np.cumsum(d_H)/1e3)
plt.ylabel('Horizontal \n displacement (km)')
plt.xlabel('time (m)')

plt.grid('on')
fig.savefig('./Figures/' + prefix + '10.1. decent profile.png')


##################      ALTITUDE / LATERAL TRAVEL

fig = plt.figure(figsize = plotSize, dpi = plotdpi)

plt.plot(np.cumsum(d_H)/1e3, h/1e3)
plt.xlabel('Gliding distance (km)')
plt.ylabel('Altitude (km)')

plt.tight_layout()
plt.grid('on')
fig.savefig('./Figures/' + prefix + '10.2. decent slope.png')


##################      INTEGRATED WIND DRIFT PROFILES

fig = plt.figure(figsize = plotSize, dpi = plotdpi)
plt.plot(np.transpose([np.cumsum(d_H_w[i]) for i in range(len(d_H_w))])[::-1], h[::-1])
plt.xlabel('Integrated Distance due to Wind Drift (m)')
plt.ylabel('Altitude (m)')

plt.ylim([0, h_max])
plt.gca().invert_xaxis()
#plt.plot(h, np.transpose(d_H_w))
plt.title('Integrated Wind Drift')
plt.grid('on')
fig.savefig('./Figures/' + prefix + '10.3. integrated distances.png')



##################      TRAVEL AS EFFECTED BY WIND DRIFT

# average windspeed
ave_wind = np.mean([envelopes[i][:len(t_H)] for i in range(len(envelopes))], 1)

#avewindDistTraveled(logs)
fig = plt.figure(figsize = plotSize, dpi = plotdpi)
```

```
        plt.plot(ave_wind, np.subtract(sum(d_H), sum(np.transpose(d_H_w)))/1E3, '.r'
)
        #plt.plot(0, sum(d_H)/1e3, 'or')
        #plt.title('')
        plt.xlabel('Average Windspeed (m/s)')
        plt.ylabel('Gliding Distance (km)')
        #plt.title('Gliding Distance as Affected by Wind')

        plt.ylim([0, sum(d_H)/1e3*1.1])
        plt.xlim([0, 40])
        plt.grid('on')


        fig.savefig('./Figures/3.5 Range.png')
        plt.tight_layout()

        fig.savefig('./Figures/' + prefix + '10.4. intersection.png')
```

```python
# -*- coding: utf-8 -*-
"""
HEADWIND_FLIGHT_UNCONTROLLED_POWERED.PY

Extend headwind_flight_uncontrolled.py to also consider the benefit of using
a motor

Created on Thu Sep 03 12:27:22 2015

@author: Sam
"""

import matplotlib

from functions import *
import numpy as np
import matplotlib.pyplot as plt

from sonde import *

from figformat import *

# prefix used for saving plots
prefix = 'fly_pwr.'

# Measured windspeed envelopes
envelopes = np.transpose(loadEnvelopes('wind.csv'))
envelopes = [envelopes[i] for i in range(len(envelopes)) if len(envelopes[i]) >= 30
1]
#get rid of the dodgy ones (should have been gotten rid of allready)
envelopes = [envelopes[i] for i in range(len(envelopes)) if np.mean(envelopes[i]) >
 0]




# Add the balloon drift patterns to fig 1, done outsi
#avewindDistTraveled(logs)


# loop over coeffients calculated for different assumptions of angle of attack
# to investigate the overall effect
for CdAd, ClAl in [[0.0212, 0.0625], [0.0190, 0.0560]]:


    # run the steady state flight model
    # Simulate accross a range of altitudes
    h_max = 30E3
    h_step = 100
    h = h_max - np.arange(-0.0000001, h_max, h_step)

    # Air density model
    rho = rho_0*np.exp(-g*M*h/R/T_0)


    #glide slope
    theta = np.arctan(1/(ClAl/CdAd))

    K = ClAl*np.cos(theta) + CdAd*np.sin(theta)

    a = np.sqrt(2*m*g/rho/K) # airspeed

    vsp = a*np.sin(theta)
    gsp = a*np.cos(theta)


    # find time taken to traverse each altitude step using vertical component of airspeed at given altitude
    t_H = h_step /  vsp
    t_H = [0]+t_H[:]


    # The above does not consider the effects of an initial acceleraion
    # period, 'Falling.py' estimates the time taken to reach a steady state velocity
    # The expected horizontal component of airspeed is set to zero for the
    # acceleration period
    import Falling
    T_acel, vsp_acel = Falling.model_acceleration(CdAd)
    # override the first 500m of decent times to simulate the acceleration period
    t_H[0:len(T_acel)] = T_acel
    gsp[0:len(T_acel)] = 0
    vsp[0:len(T_acel)] = vsp_acel

    t_H_o = list(t_H)
    gsp_o = gsp[:]

    t_H[-3] = 40*60 # overide the time taken to traverse from 300m to 200m to simulate a period of powered flight.
    gsp[-3] = 18 # overide airspeed as well, as we'll be going slower


    # find distance traveled at each altitude step
    d_H = np.multiply(t_H, gsp)
    d_H[0] = 0
    d_H[1] = 0

    d_H_o = np.multiply(t_H_o, gsp_o)


    # find distance carried at each altitude step due to wind drift
    d_H_w  = []
    for i, env in enumerate(envelopes):
        if len(env) >= 301:
            d_H_w.append(np.multiply(t_H, env[:len(t_H)][::-1]))


    # For the unpowered model, wind drift is different because it doesnt spend
    # a long period at one low altitude
    d_H_w_o  = []
    for i, env in enumerate(envelopes):
        if len(env) >= 301:
            d_H_w_o.append(np.multiply(t_H_o, env[:len(t_H)][::-1]))



    ###############         ALT/DIST over TIME

    fig = plt.figure(3, figsize = plotSize, dpi = plotdpi)
    plt.subplot(2, 1, 1)
    plt.title('Powered Descent Profile')
    plt.plot(np.cumsum(t_H)/60, h)
    plt.ylabel('Altitude (m)')
    plt.grid('on')
    plt.subplot(2, 1, 2)
    plt.plot(np.cumsum(t_H)/60, np.cumsum(d_H))
    plt.ylabel('Lateral distance traveled (m)')
    plt.xlabel('time (m)')
    plt.grid('on')
#    fig.savefig('./Figures/' + prefix + '10.1. powered decent profile.png')


    ###############         ALT over DIST


    fig = plt.figure(4, figsize = plotSize, dpi = plotdpi)
#    plt.title('Decent Profile (altitude / distance)')
    plt.plot(np.cumsum(d_H)/1e3, h/1e3)
    plt.xlabel('Lateral distance traveled (m)')
    plt.ylabel('Altitude (m)')
    plt.grid('on')
    plt.tight_layout()
    fig.savefig('./Figures/' + prefix + '10.2. decent slope.png')


    ###############         Integrated wind drift


#    fig = plt.figure(figsize = plotSize, dpi = plotdpi)
#    plt.plot(np.transpose([np.cumsum(d_H_w[i]) for i in range(len(d_H_w))])[::-1], h[::-1])
#    plt.xlabel('integrated distance due to wind drift (m)')
#    plt.ylabel('Altitude (m)')
#
#    plt.ylim([0, h_max])
#    plt.gca().invert_xaxis()
#    #plt.plot(h, np.transpose(d_H_w))
#    plt.grid('on')
#    fig.savefig('./Figures/' + prefix + '10.3. integrated distances.png')

    #avewindDistTraveled2(logs)


    ###############    Powered vs unpowered flight as effected by windspeed



    max_dist_corrected = []
    wspeed_ave = []
    for i, log in enumerate(logs):
        if 'PLat' in log and 'WSpeed' in log:
            displacement = LatLon2Dis(log)
            displacement = clean(displacement)
            max_dist_corrected.append(max(displacement)/(log['Time'][-1]/(1.6*3600)))
            wspeed_ave.append(sum(clean(log['WSpeed']))/len(log['WSpeed']))




    # average windspeed
    ave_wind = np.mean([envelopes[i][:len(t_H)] for i in range(len(envelopes))], 1)

    # Exponentially wieghted average windspeed
    #exp_weights = 1/0.63212/len(t_H) * np.exp(-np.divide(np.arange(0, len(t_H)), float(len(t_H))))
    #ave_wind =  sum(np.multiply(exp_weights, [envelopes[i][:len(t_H)] for i in range(len(envelopes))]))
    #avewindDistTraveled(logs)


```

```python
        fig = plt.figure(1, figsize = plotSize, dpi = plotdpi)

    #    plt.plot(ave_wind, np.subtract(sum(d_H_o), sum(np.transpose(d_H_w_o)))/1E3, '.r') # add the original set as well


        plt.plot(wspeed_ave, max_dist_corrected, 'm.')         # ascent rate adjusted balloon drift

        plt.plot(ave_wind, np.subtract(sum(d_H), sum(np.transpose(d_H_w)))/1E3, '.g')
        #
    #    plt.plot(ave_wind, np.subtract(sum(d_H_o), sum(np.transpose(d_H_w_o)))[:len(t_H)]/1E3, '.r') # add the original set as well
        #
        #plt.plot(ave_wind, np.subtract(sum(d_H), sum(np.transpose(d_H_w)))[:len(t_H)]

        #plt.plot(0, sum(d_H)/1e3, 'or')
        plt.hold('on')
        #plt.title('')
        plt.xlabel('Average Windspeed (m/s)')
        plt.ylabel('Lateral Travel Distance (km)')
    #    lgd =plt.legend(['Balloon drift', 'Unpowered flight distance', 'Powered flight distance'],bbox_to_anchor=(1, -0.3))
    #    lgd =plt.legend(['Unpowered flight distance', 'Powered flight distance'],bbox_to_anchor=(1, -0.3))
        lgd =plt.legend(['Balloon drift for 5m/s ascent', 'Powered flight distance'],bbox_to_anchor=(1, -0.3))


        plt.tight_layout()
        plt.grid('on')

        plt.xlim([0, 40])
        plt.ylim([0, 200])

        fig.savefig('./Figures/' + prefix + '10.4. intersection.png')
```

```python
# -*- coding: utf-8 -*-
"""
FALLING.PY

Model the initial acceleration period of an airframe with drag and gravity, as
if the wings don't exist.

This is done using time stepping, but results are converted for time in terms
of hight so they can be used with the rest of the model


Created on Sun Oct 18 10:42:16 2015

@author: Sam
"""

import numpy as np

def model_acceleration(CdAd):

    rho_0 = 1.2250 # air density at sea level
    T_0 = 288.15 # temperature at sea level
    g = 9.80665 # Acceleration due to gravity
    M = 0.0289644 # Molar mass of air on earth (Kg/mol)
    R = 8.31432 # Universal gas constant for air


    # Air density model

    m = 2


    h = 30e3;

    V = []
    H = []
    T = []

    dt = 0.1;
    Ts = np.arange(0, 25*60, dt)


    v = 0;
    for t in Ts:
        rho = rho_0*np.exp(-g*M*h/R/T_0)
        fdrag = 1.0/2*rho*(v**2)*CdAd;
        a = (m*g - fdrag)/m

        v += a*dt
        h = h - v*dt
        V.append(v)
        H.append(h)
        T.append(t)
        if h <= 0:
            break




    # find the times where the model crosses 100 meter steps, these times will be u
sed to model the acceleraion of the flighth model
    h_lim = 29e3
    T_steps = []
    v_steps = []
    for i in range(len(H)):
        if H[i] <= h_lim:
            T_steps.append(T[i])
            v_steps.append(V[i])
            h_lim -= 100
            if H[i] <= 25e3:
                break

    return np.diff([0] + T_steps), v_steps

if __name__ == '__main__':
    CdAd = 0.0212
    T_steps, v_steps = model_acceleration(CdAd)

```

Flight plan generation code in data matrix format

```
function [flight_plan] = flightPlan(LZs, flight_plan)
  %Author: Callum White
  %Date: 13/8/2015
  %Returns a flight plan based on an input matrix of lat, long and
  %altitude values for secondary landing zones. An nx12 flight plan
  %matrix may be passed as the second input to have the first input
  %appended in the correct format.
  dim = size(LZs);
  for i = 1:dim(1)
    if (nargin < 2) && (i == 1)
      flight_plan = add_LZ(LZs(i,1), LZs(i,2), LZs(i,3));
    else
      flight_plan = add_LZ(LZs(i,1), LZs(i,2), LZs(i,3), flight_plan);
    end
    i = i + 1;
  end
end


function [flight_plan] = add_LZ(lat, long, alt, flight_plan)
  %Author: Callum White
  %Date: 13/8/2015
  %Appends a new landing zone to a flight plan based on the latitude,
  %longitude and altitude. If no flight plan is passed in, a new plan is
  %created. The flight plan is returned by this function, as well as
  %saved in a text file (which can be read in mission planner).

  SoftwareVer = 'QGC WPL 110';
```

```
    %Define home location: Lauder, Otago.
    %Defines landing routine to land at the point of launch. Does not
    %account for obstructions such as trees or fences.
    home(1,:) = [0 1 0 16 0 0 0 0 -45.040136 169.688333 100 1];
    home(2,:) = [1 0 3 16 0 0 0 0 -45.040136 169.688333 1000 1];
    home(3,:) = [2 0 3 31 0 0 0 0 -45.040136 169.688333 50 1];
    home(4,:) = [3 0 3 21 0 0 0 0 -45.040136 169.688333 1 1];

    if nargin < 4
        flight_plan = home;
    end

    dim = size(flight_plan);
    i = dim(1);

    %Append new landing zone to the flight plan:
    %Note: This method does not account for landing approach. Obstructions,
    %such as trees or fences, must be avoided by manual waypoint input.
    newLZ(1,:) = [i 0 3 189 0 0 0 0 lat long 1000+alt 1];
    newLZ(2,:) = [i+1 0 3 16 0 0 0 0 lat long 1000+alt 1];
    newLZ(3,:) = [i+2 0 3 31 0 0 0 0 lat long 50+alt 1];
    newLZ(4,:) = [i+3 0 3 21 0 0 0 0 lat long alt 1];

    flight_plan = [flight_plan; newLZ];
end
```