

TITLE

Autonomous Soaring with Thermal Energy Extraction

PROJECT MENTORS

Jeffrey Koessler, Andrew Dudar

FACULTY ADVISORS

Hermann Fasel, Ricardo Sanfelice

A handwritten signature in black ink, appearing to read 'P. Tindall', is written over the text 'TEAM MEMBERS'.

TEAM MEMBERS

Phillip Tindall, Joel Mueeting, Maira Garcia, Nicholas Griffis, Matthew Ashton

DATE

May 9, 2014

Table of Contents

- Abstract (pg. 3)
- Introduction (pg. 4)
- Relevant Literature and Theory (pg. 5)
- Thermal Characterization and Theory (pg. 6)
- Platform
 - o Airframe (pg. 9)
 - o Hardware (pg. 11)
 - o Software (pg. 14)
 - Mission Planner (pg. 14)
 - Arduino (pg. 16)
 - o Sink Polar (pg. 17)
- Control Logic
 - o Control Logic Theory and Alternative Proposed Algorithms (pg. 18)
 - o Thermal Decision Logic (pg. 21)
 - o Arduino Implementation of Control Logic (pg. 23)
- Controller Simulations (pg. 26)
- Testing (pg. 29)
- Conclusion (pg. 32)
- Appendix (pg. 34)
- References (pg. 36)
- Layout of Individual Responsibilities (pg. 37)

Abstract

Unmanned Aerial Vehicles (UAVs) are rapidly becoming an integral part of everyday life. Whether for military surveillance, personal entertainment, or commercial transportation, each UAV is limited in flight time by the amount of fuel it can carry or the power its batteries can hold. This project sought to break that boundary by allowing a gliding UAV to autonomously make use of the natural energy of thermals: rising pockets of air that form over warm patches of ground. Birds and manned gliders have already been making use of this energy for years by a process called thermalling, in which they are able to gain altitude by circling around the center of a thermal and rising with the surrounding air. By altering the autopilot code of a typical UAV glider, this project was able to achieve autonomous thermalling in both simulated and actual flight tests, and achieved more than triple the plane's natural gliding flight time.

Introduction

Thermalling is a process which has been used for millennia by birds, but humans have only been studying thermals for a relatively short time. Human studies of thermals began in April 1883, when John Strutt (Lord Rayleigh) noticed that birds could fly upward without ever flapping their wings [Reference 1]. Based on this observation, he postulated that the air around them likely had some non-horizontal component to its velocity. Lord Rayleigh was correct, as was later confirmed by manned glider pilots, who have since used thermals to gain altitude and increase their flight times. Of course, manned gliders and birds have intuitive senses and variometers that are not reasonably applicable to a small unmanned glider. The initial difficulty of this project was to try to match these instinctive and natural senses with digital ones, so that the UAV could determine when it was in a thermal and initiate a protocol to circle the thermal's center and gain altitude. While thermal imaging initially seemed to be an effective method of locating thermal pockets, this was soon found to require too heavy of hardware to be feasible. Instead, the UAV must rely on data obtained from the on-board GPS unit and pitot-static tube to determine whether or not it is in a thermal. After that, the autopilot system must set a bank angle to circle around the thermal and gain altitude. Finally, the UAV must set some criteria for when to exit thermalling flight and continue with the mission.

Formulating these three components—determining the presence of a thermal, circling the thermal, and evaluating when to leave—into a useable code that could be implemented into the existing Arduino autopilot software was the primary focus of this project. The final results were that the code was implemented into the UAV, autonomous thermalling was seen in test flights, and the flight time was tripled when compared to natural, non-thermallng flight under the same conditions.

Relevant Literature and Theory

As one of the goals is to identify, characterize, and utilize thermals, it is important to be able to understand what the atmosphere is doing and how to measure it, as well as how best to write algorithms to control the aircraft. A number of articles were used to gain this understanding, including Michael J. Allen's "Updraft Model for Development of Autonomous Soaring Uninhabited Air Vehicles" [Reference 2] and Jason A. Kyle's "Optimal Soaring by a Small Autonomous Glider" [Reference 3]. These papers focused on the common goal of extracting energy from the environment in the form of extended flight time. Kyle's paper focused on a way to develop algorithms using knowledge of the environment's wind energy to enable an aircraft to constantly optimize its trajectory during flight. Due to the constant change in the atmosphere, these algorithms have to constantly adapt to environmental change as it happens. Kyle modeled the velocity gradient from the center of a thermal at a single altitude, which is essential for creating a successful simulation. The book "Aircraft Control and Simulation" [Reference 4] by Stevens and Lewis was used heavily for the formulation of the six degree-of-freedom aircraft model developed in this project.

Also, due to the limited scope of this project, last year's project's work on thermal characterization was used so as to enable this year's project to focus on its main goals. That thermal characterization drew heavily from Allen's "Updraft Model for Development of Autonomous Soaring Uninhabited Air Vehicles". Allen extensively studied the behavior of the convective mixing layer in the atmosphere, and through experimentation developed a mathematical model to describe how the shape of a thermal changes with altitude.

Thermal Characterization and Theory

A thermal is a rising section of air, often in the shape of a column, induced by heat fluctuations on the terrain from sun radiation. As the air near the heated ground gains energy, its density decreases, causing it to rise up from the ground. Certain terrain features cause thermals more easily than others, such as plowed fields and large sections of pavement, due to their ability to absorb heat. If there is enough moisture in the atmosphere, the air will expand and condense as it rises, which will sometimes result in the formation of a cumulus cloud. For this reason, manned and human guided gliders will often look for cumulus clouds as they come with a high chance of indicating the presence of a thermal. For unmanned systems, however, these visual indicators are difficult to utilize without complicated algorithms and heavy sensor equipment. The only easily accessible information for detecting thermals is the air's direct effect on the vehicle's motion and flight for maximizing lift and increases in altitude.

By the conservation of mass principle, rising air in a closed system means there must also be an equal amount of sinking air in the same general area. In a thermal, the air has the highest rise rate at the center, with the rise rate decreasing as the distance from the center increases. This results in an induced bank, due to the non-uniform lift distribution across the wings of the aircraft. At a certain distance from the center, the rise rate will drop below zero, and there will be a region of sinking air surrounding the region of rising air. This region of sinking of air is always present in a thermal.

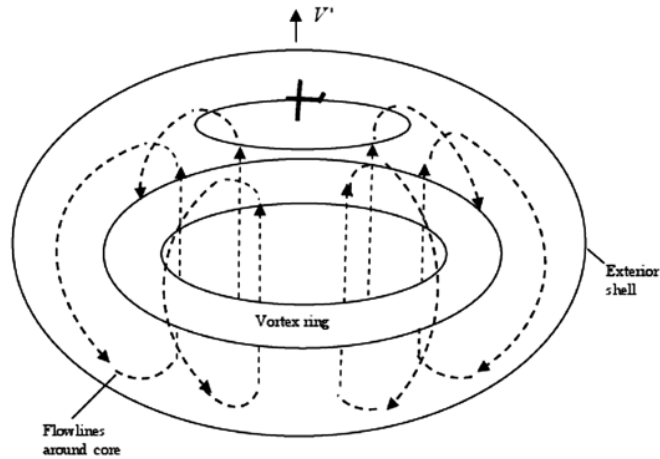


Figure 1 - Internal Structure of a Thermal

In order to characterize a thermal, it is necessary to understand what happens inside a slice of air as it moves up the thermal. As a section of air moves up a thermal, it undergoes a number of changes. As the altitude within a thermal increases, the rise rate of the air also increases. This means that as a plane rises within a thermal, its upward velocity should increase as its altitude in the thermal increases. Next, the rise rate of the air becomes more uniform, which means that as it gains altitude, an aircraft should be able to decrease its bank angle and still maintain the same rise rate. Also, as altitude increases, a thermal will begin to expand and increase in radius, and it often ends in a cumulus cloud (when it does not, it is called a “blue thermal”). A very important behavior of thermals to note is that they drift, they do not always stay in the same place, which is an important consideration when trying to track and utilize a thermal to gain altitude.

While complicated, it is important to take all of these behaviors into consideration when trying to develop algorithms for utilization of thermals. Previously, thermal drifting and expansion were not taken into account for soaring purposes. However, to achieve maximum lift

and altitude gain it would be necessary to develop an algorithm that can account for these changing factors.

Platform

Airframe

The glider used for this project was the Hobby King Fox 2.32 m RC Motor Glider, which is a scale model of an aerobatic sailplane. The aircraft is made from expanded polyolefin (EPO) foam and has two carbon fiber rods and an alloy joiner on the main wing.



Figure 2 - Hobby King Fox 2.32 m RC Motor Glider [Reference 10]

During the first test flight, the center of gravity (CG) was checked to ensure the components inside the glider were properly placed in order to avoid tail- or nose-heaviness. After the preliminary check, the center of gravity was adjusted by moving the hardware around and is now located 40 mm back from the wing leading edge.

Before each test flight, the aircraft was leveled on the ground and calibrated and then take off was initiated in stabilized mode. During the first test flight, the aircraft took off in manual mode to verify operation but every flight since has been in the “stabilized mode,” which provides some autonomous control to level the flight. In the second semester, the aircraft no longer had to

be leveled on the ground and calibrated because it was found that the new code implemented into the aircraft calibrated the aircraft automatically when it was powered on.

During a later test flight, the motor got shorted out due to a poor design in the stock motor mount. The design allows the motor to spin in its mount, and this can lead to excess strain on the wires, causing them to short out. Fortunately, a spare motor was readily available. A camera was also added to the aircraft to provide visual data to accompany the numerical data.

Throughout the latter half of the project, the only change made to the aircraft was the addition of tape on each wing to secure it to the fuselage since the screws connecting the wings had gotten loose. The aircraft has always had a pitot-static tube installed but the functionality was never verified until the spring of 2014. During the last flight, the OSB board that added the data overlay onto the video footage was removed.

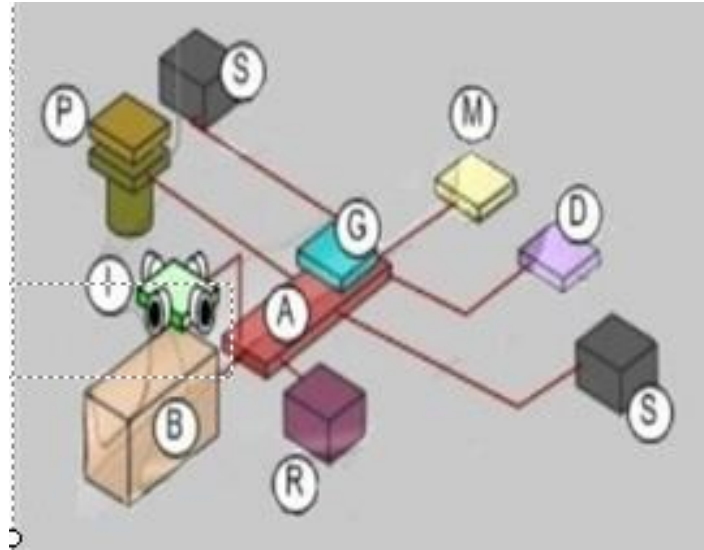
Key Specifications

Wingspan:	2.32 m (91.3 inch)
Length:	1.29 m (50.8 inch)
Weight:	Originally 1150 g (2.5 lbm), with additions 1516 g (3.3 lbm)
Controls:	Ailerons (2X), Elevator, Throttle, Rudder

See Appendix A for more involved and detailed specifications for the aircraft.

The electrical components of the platform are as follows:

- 3DR 915 MHz Telemetry Modules
- 540TVL SONY CCD Chipset Color Board Video Camera
- Single 3S 2100 mAh LiPo Battery supplying ALL aircraft power:
- 3DR Power Module supplying 5 VDC to APM / RC Rx / GPS / Telemetry
- Castle Creations 10 A BEC supplying 5 VDC to Servos (output plugged into APM output ch8)



- (A) Autopilot: 3DR APM 2.5+
- (B) Battery: 3S LiPo, 2200 mA-hr
- (D) Telemetry: 3DR 915 MHz
- (G) GPS Receiver: 3DR uBlox LEA-6
- (M) Electronic Speed Control (ESC)
- (R) RC Receiver: Spektrum AR6210 2.4 GHz
- (S) Servos: 2X Aileron, Rudder, Elevator
- (P) Altimeter & Airspeed (Pitot / Static)

Figure 3 - Diagram of GC-Fox' electrical system [Reference 11]

Hardware

The airframe has been equipped with several pieces of hardware to accomplish the mission that were not originally implemented into the production design. The main piece is the APM module for the autopilot needed for autonomous flight. We are using the APM 2.5 that was used at the end of last year and was already set up and functioning properly.

The APM 2.5 board comes from the factory already soldered and in an enclosure, ready to have the chosen firmware loaded by the Mission Planner (Mission Planner software will be discussed in more detail in the Mission Planner subsection).

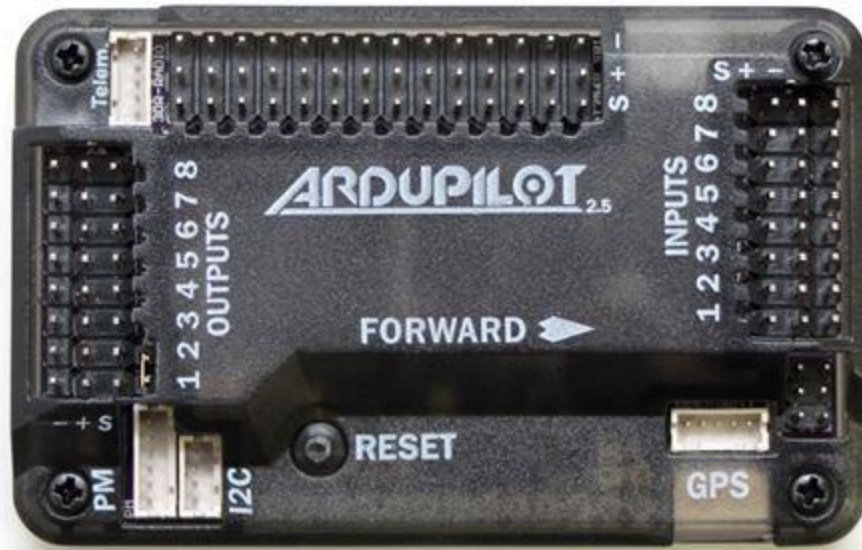


Figure 4 – Ardupilot Board [Reference 6]

Connected to the APM module is a GPS module, which collects all the data and sends it to the module. We are using the GPS unit that was used last year which was also already installed and functioning properly. This miniaturized ceramic GPS patch antenna is based on smart XtremeGain™ technology. It is mounted via pin and double-sided adhesive and has been tuned as optimal solution for small UAVs.



Figure 5 - 4mm thick GPS Patch Antenna

There is also a pitot-static tube installed on the right wing that was installed last year and remains on the airframe. The pitot tube was connected and calibrated so that more accurate airspeed data could be obtained. This data is not currently being used for the thermal finding

code but is being used in data analysis of test flights. In the future it would be ideal to use this airspeed data in some sort of total energy calculation for thermal finding.

Along with the APM module is a ground module with USB connection. This plugs straight into the ground station computer and is the means to send and receive information to the aircraft wirelessly. This is very crucial for being able to change mission waypoints and flight modes during flight tests. It is also how the ground crew can observe the aircraft without being in visual range. It works in conjunction with the "Air" module, offering a 2-way half duplex wireless communication system. It is based on the 3DR Radio system and is 100% compatible.



Figure 6 – Ground Module for Transmitting from Mission Planner to G-C Fox [Reference 9]

The computer used as the ground station is an HP Pavilion g7 17.3" Notebook PC purchased during the first weeks of the project. The mission planner software along with Google Earth maps and analyzing software such as Matlab run very efficiently on the Windows 8 platform. The ground station computer has the following specifications.

- Product numbers: g7-2282nr or g7-2281nr

- 17.3" diagonal HD BrightView LED-backlit display
- Windows 8 operating system
- AMD Quad-Core A8-4500M accelerated processor
- 8GB DDR3 memory
- AMD Radeon HD discrete-class graphics
- 1TB hard drive capacity
- HP ProtectSmart hard drive protection
- Altec Lansing dual speakers with Dolby Advanced Audio
- SuperMulti DVD burner

Software

The autopilot firmware installed and running on the aircraft is ARDUPilot version 2.78b, which comes with a fixed-wing autopilot called APM:Plane. The free APM:Plane (formerly ArduPlane) firmware running on the APM autopilot gives any fixed-wing aircraft full autonomous capability. APM:Plane provides advanced functions such as support for hundreds of three-dimensional waypoints, automatic take-off and landing, and sophisticated mission planning and camera controls. It works with the desktop Mission Planner ground station software to form the complete software platform for this project.

Mission Planner

Mission operations are done through a software interface called Mission Planner. The entire package allows the user to set a custom series of waypoints, view real-time data of the plane's state, and mathematically and graphically evaluate the telemetry data after the flight.

Mission Planner is a free downloadable program that works in conjunction with the APM autopilot system that we have installed on our aircraft.



Figure 7 – Mission Planner [Reference 8]

The Mission Planner version and build number that we are using for our project is version 1.2.95 build 1.1.5150.11972. The Mission Planner, created by Michael Osborne, is the default software to interact with an Arduino autopilot board. Its features include:

- Point-and-click waypoint entry, using Google Maps.
- Select mission commands from drop-down menus
- Download mission log files and analyze them
- Configure APM settings for your airframe
- Interface with a PC flight simulator to create a full hardware-in-the-loop UAV simulator.
- See the output from APM's serial terminal

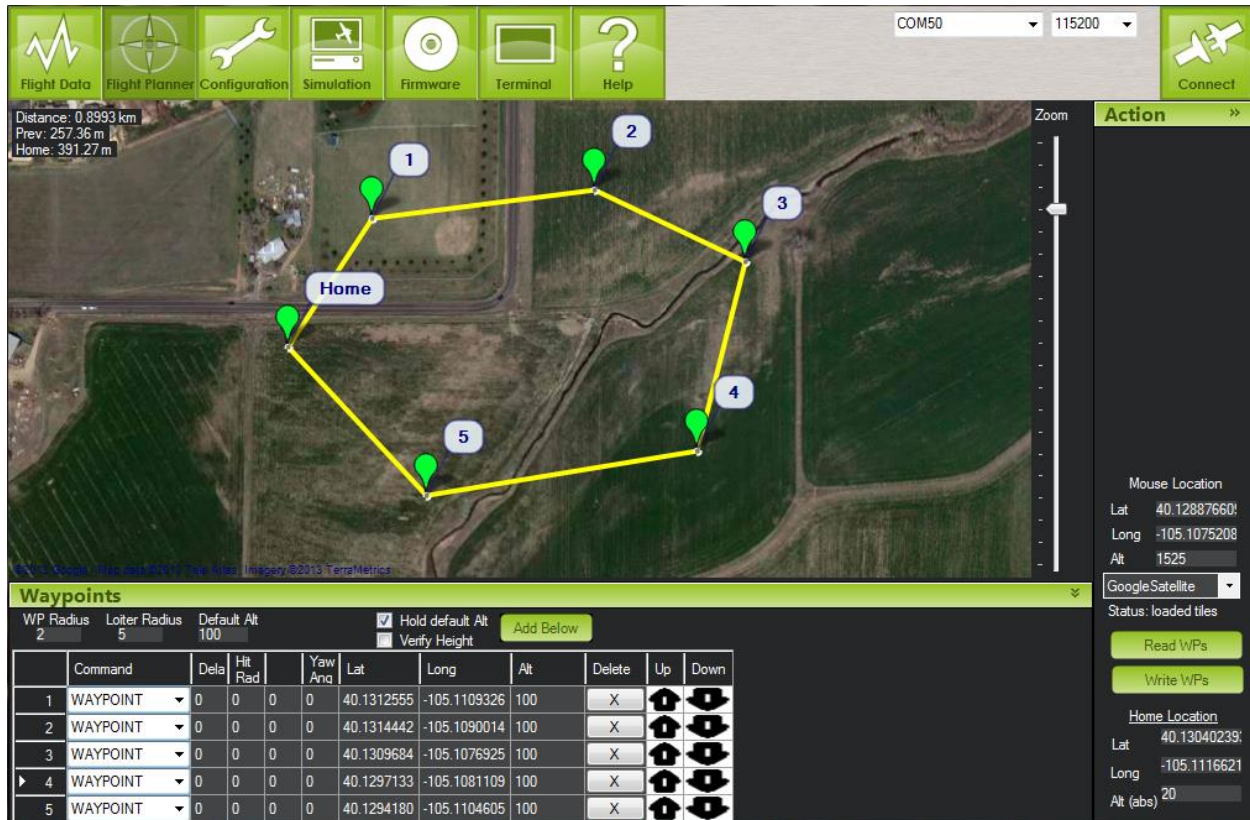


Figure 8 – Waypoint Navigation Screen in Mission Planner

Arduino

The APM:Plane autopilot is programmed in a language called Arduino. The Arduino Software is free, open source, and available for Windows, Mac OS, and Linux. Familiarity with Arduino’s coding structure was necessary to modify the APM autopilot code and implement our thermal searching mode of flight. Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring, which is roughly similar to C) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash, Processing, MaxMSP).

Sink Polar

The glide ratio of an aircraft is an important parameter to consider when designing the controller for an autonomous glider. Developing a sink polar assists in determining the optimal speed at which to fly when not in a thermal and in calculating the expected sink at any speed. Throughout this year a large amount of flight data has been collected and analyzed to produce the following sink polar data. Each data point was calculated by comparing the altitude given by the glider's GPS over certain time intervals to the glider's airspeed. The glider must be flying as level as possible while maintaining a constant speed. Much of the data was slightly noisy and wind perturbations caused the altitude data to vary nonlinearly with time, though for time intervals of approximately 30 to 20 seconds the trend was mostly linear. The data was put through an infinite impulse filter (which can be seen in the Thermal Decision Logic Section) in order to reduce the noise, and the following Sink Polar was obtained.

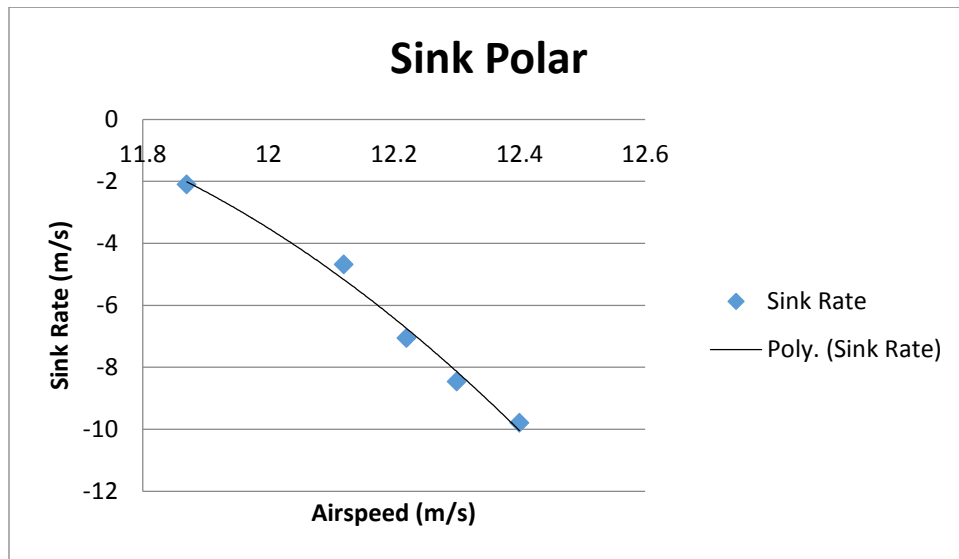


Figure 9 – Sink Polar for G-C Fox

Control Logic

Control Logic Theory and Alternative Proposed Algorithms

The object of this control logic was to get the glider to identify when it is in a thermal, execute a flight maneuver to characterize the thermal, and then find an optimal bank angle to maximize its rise rate. The most important part of a thermal is the fact that it increases lift and causes the plane to rise. However, just because the plane is rising for a short length of time does not consistently indicate the presence of a thermal, so it is necessary to have more checks than just an increase in lift to verify that the plane is actually in a thermal, which is necessary to counterbalance the cost of the plane turning and potentially losing altitude. Two important characteristics of a thermal are that it has a down draft surrounding its rising air center, and it induces a bank angle away from its center unless the plane is flying directly along the velocity gradient of the thermal. Using these pieces of information, it is possible to create a control logic that will allow a reasonable level of surety for determining whether the aircraft is in fact in a thermal or not. This control logic is demonstrated in the following figure:

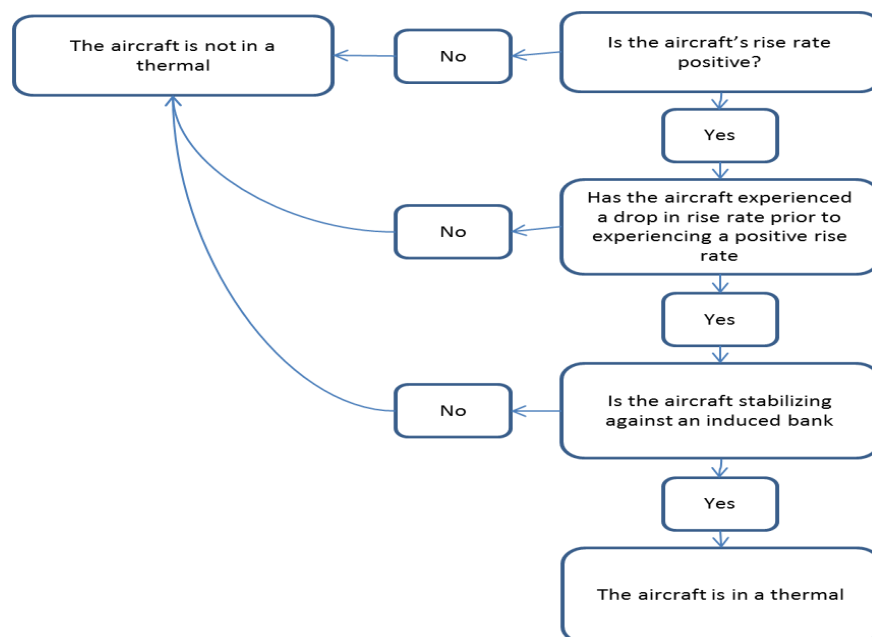


Figure 10 – First Potential Control Logic Structure

The other important problem to solve is the characterization of the thermal and finding the optimal bank angle. For characterizing the thermal, it is necessary to know two pieces of information: the strength of the velocity field as the aircraft gets closer to the thermal, and the actual geometric center of the thermal. The demonstration of this process is shown in Figure 11 below:

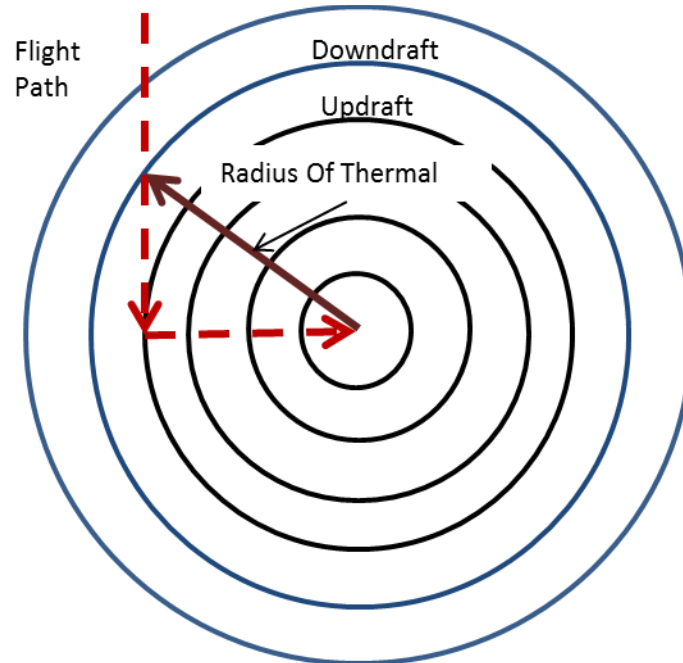


Figure 11 – Aircraft Entering a Thermal

The aircraft is in straight and level flight as it is determining whether it is in a thermal. This algorithm saves a certain amount of its attitude motion, so once it determines it is in a thermal, it determines the starting point of the increase in lift when it enters the circle. Then, as it is flying it monitors the second derivative of its roll rate. When the second derivative becomes negative, it marks the point at which it is negative and banks so as to fly up the velocity gradient of the thermal towards the center. It then monitors the second derivative of its vertical position, and when that value goes to zero and is negative, it marks that point as the center of the thermal. Using the information just calculated, the aircraft can calculate the radius of the thermal using

the Pythagorean Theorem. This information, along with the catalogued rise rate data, is enough to characterize the thermal. In order to calculate the optimal circling bank angle, the aircraft will optimize the following equation:

$$f = (\text{rise rate}) * (\cos(\theta))$$

Optimizing this equation will yield the optimal bank angle to circle within the thermal. The aircraft will then execute this bank angle, which will allow it to achieve the maximum rise rate within the thermal. The control logic for this process is illustrated in the following figure

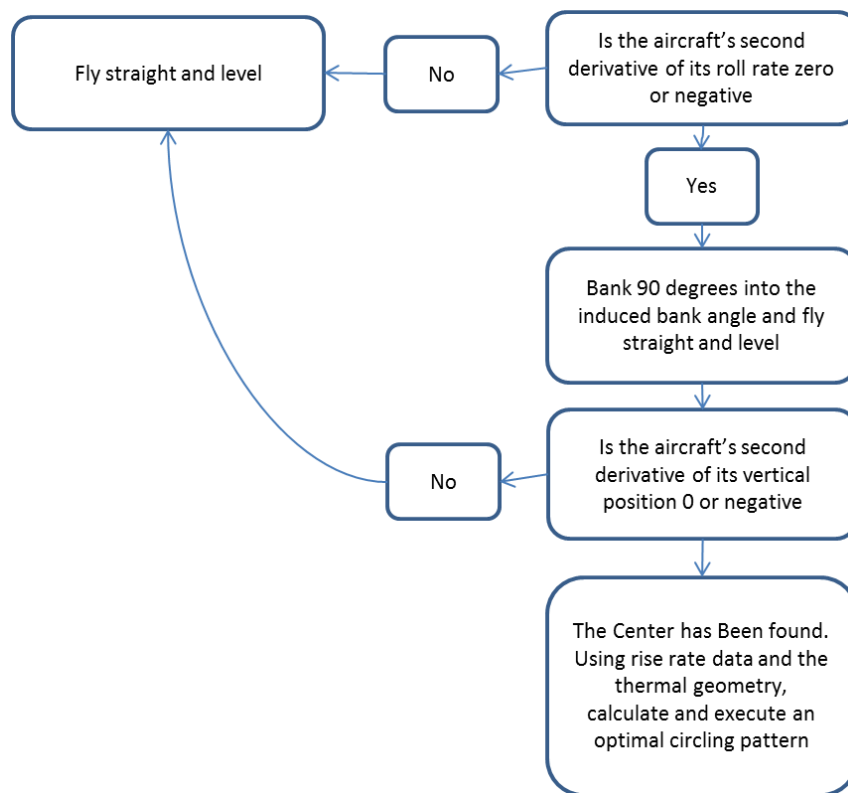


Figure 12 – Second Potential Control Logic Structure

During simulation, it was discovered that this algorithm was overly idealistic, and only allows for a small portion of thermalling scenarios to be properly analyzed. Also, when it was decided that the project called for an imbedded, efficient control logic loop, it became apparent that this loop would not work as it was currently conceived. This is due to the fact that this control logic requires an extensive amount of real-time data storage and analysis, which is both

very difficult to implement into a high frequency looped auto-pilot, and extremely computationally expensive. However, it is believed that this logic at its core is sound in theory and should be revisited in future projects and investigated further.

Thermal Decision Logic

The primary object of this portion of the control logic deals with the plane detecting whether or not it is in a thermal and taking the appropriate response. While there were a number of methods considered, the method was eventually dictated by the available variables and data given from Arduino software itself, which ended up being the vertical component of the airspeed given from GPS/barometer data. The control logic that was actually implemented is shown in Figures 13 and 14.

This control logic consists of two main parts, the algorithm loop deciding whether or not to switch from waypoint navigation to thermal navigation and the desired bank angle, and the loop analyzing the incoming vertical velocity data and giving the necessary information to the outer loop to decide what action to take. The inner loop uses two main components: an IIR (infinite impulse filter) to filter the data and dampen random variations and value fluctuations, and a counter to determine whether the vertical velocity has been above or below zero for long enough to start or stop thermalling, respectively. An infinite impulse filter is a filter that takes a running average over a sequence of points, but its unique construction allows for the previous data to all be stored in one variable, making it ideal for embedded loops. The infinite impulse filter takes the form of the following equation:

$$\bar{x} = \alpha * x_{new} + (1 - \alpha) * \bar{x}_{old}$$

In this equation, \bar{x} is the current average of the data, \bar{x}_{new} is the current value in the loop, \bar{x}_{old} is the previously calculated average, and α is the filter coefficient. In this filter, α is the value that is used to fine tune the filtering, for example, a 0.1 value of α would represent the equivalent of a 10 data point average once 10 samples were taken, and a .01 value would represent a 100 data point average. This filter was a key component in making the data being fed into the algorithm usable for analysis. Before filtering, the data was far too noisy to be usable.

The second part of the algorithm is a counter that requires there to be five (or any other desired value) filtered values showing vertical airspeed above zero before the outer loop is told to enter thermalling, and three (or any other desired value) consecutive filtered values of vertical airspeed below zero before the outer loop is told to return to waypoint navigation.

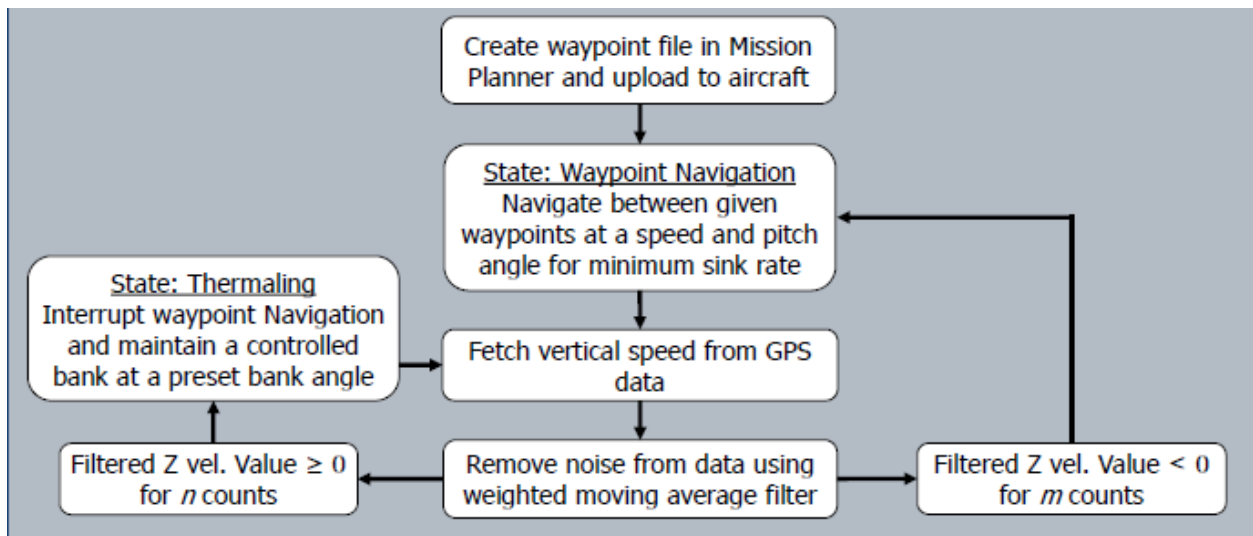


Figure 13 – Final Implemented Control Logic Structure

Figure 14 shows how the control logic was implanted into the main control loop of the ArduPilot software. In this configuration, the thermalling controller and filter are used in every instance of the main loop. Climb velocity is fetched from the GPS/barometer and used to determine whether or not to increment the count variable of the current state. If a set count threshold has been reached, then the state will switch. There are only two states: Waypoint

Navigation and Thermalling. In the latter state, a bank angle override command is executed and the ArduPilot’s PID bank angle controller keeps the aircraft at the constant bank given. In the Waypoint Navigation state, the controller autopilot is given full control of the aircraft to fly the plane from waypoint to waypoint.

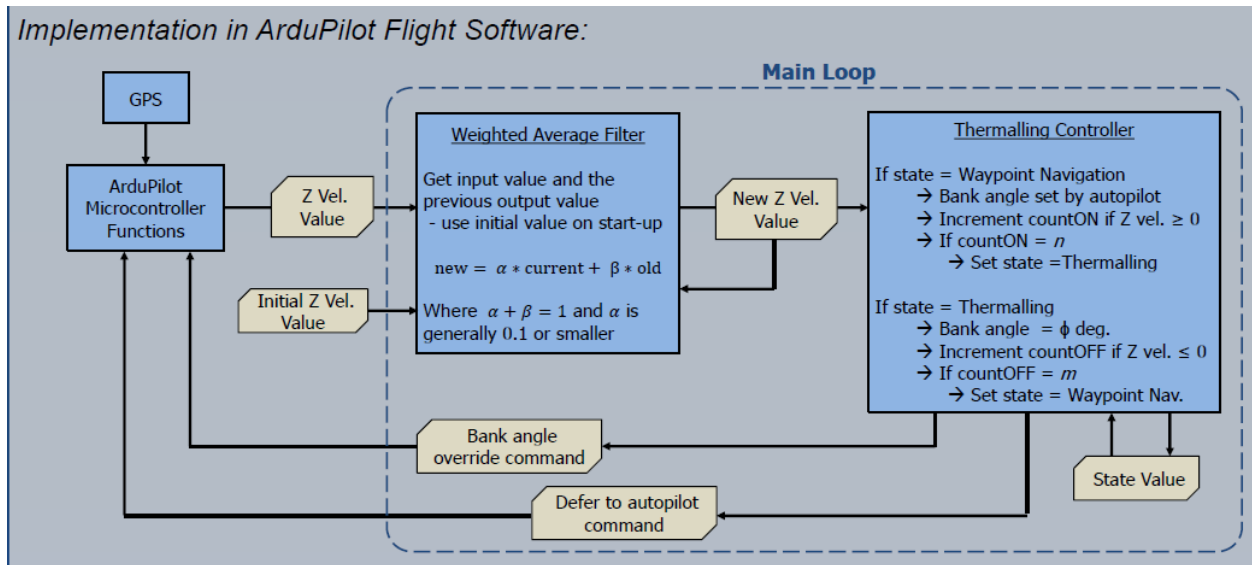


Figure 14 – Control Logic Implementation Structure

Arduino Implementation of Control Logic

The primary challenge in implementing the above control algorithm into an actual autopilot was to try to find where we could add to the core logic without having to change or remove the existing code, since we had it working well with our airframe. We felt that this would give us a better chance of keeping the stabilization and waypoint finding portions of our flight working properly while only adjusting our thermal-finding portion of our autopilot.

In order to do this, we had to find where in the code we could add our thermal finding without interrupting any other procedures. We chose to implement our thermal finding in the “auto” mode of the already existing code so that we wouldn’t have to switch modes ourselves or have to create a completely new mode. The “auto” mode in the code, which autonomously

follows waypoints from a mission plan uploaded to the plane prior or during flight, is actually rather simple in the outermost loop of the code. If the plane is in “auto”, and is not taking off or landing, it runs three functions. The three functions are to calculate the desired pitch, calculate the desired roll, and calculate the desired throttle.

Since the basic algorithm we used was to bank to a given angle when vertical velocity was sensed, we chose to put our thermal code in the function calculating roll. Inside this function, the code acquires data from the GPS and compares that to the next waypoint in order to determine which direction to turn and at what rate. Here we added an if statement to determine whether or not we had positive vertical airspeed. If we did, then we would initiate a right bank to 24 degrees, if we did not, the code would continue as usual to acquire the waypoint and make the appropriate turn.

The strategy that we used worked well except that the GPS data was very sensitive and resulted in a very noisy response. This caused our plane to go in and out of thermal mode constantly from the fact that it was taking live data and could not determine the trend but only the instantaneous situation. In effect, the noise in the data caused the plane to switch rapidly in and out of thermalling mode. This is when we decided to filter the data and require consecutive values in order to change modes. All of this was done inside of the function for calculating roll and after our if statement determining whether a thermal was present or not.

The following code is the actual code used during the last test flight. This is only the part of the code that was put in the function to calculate the desired roll. Other smaller portions of code we added in different areas in order to initialize variables and calibrate the autopilot.

```
static void calc_nav_roll()
{
```

```

// Added 4/30/2014

fresh = g_gps->velocity_down();

newz = 0.1*fresh+0.9*old;

old = newz;

if (thermal==0) {

    nav_roll_cd = nav_controller->nav_roll_cd();

    nav_roll_cd = constrain_int32(nav_roll_cd, -roll_limit_cd, roll_limit_cd);

if (newz<=0) {

    countON = countON+1;}

if (countON == 5) {

    thermal=1;}

countOFF = 0;

}

if (thermal==1) {

    nav_roll_cd = 2500;

if (newz>0) {

    countOFF = countOFF+1;}

if (countOFF == 5) {

    thermal=0;}

countON = 0;

}

}

```

Controller Simulations

The main controller logic used to determine if the aircraft has encountered a thermal or not was simulated by coding the algorithm in MATLAB and feeding in raw flight data from previous test flights. This had the benefit of being able to test the control logic as if it was actually flying, allowing us to be able to fully test the software without having to conduct numerous test flights. The results of the test are shown in Figure (D).

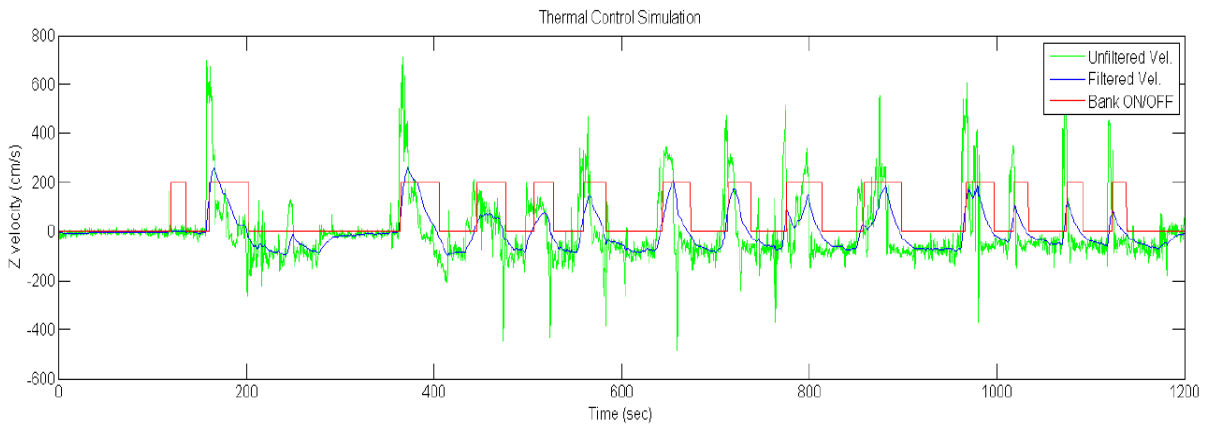


Figure 15 – Simulated Control Response Applied to Actual Flight Data

This controller depends heavily on three variables: the coefficient used in the filter, the “on” count variable threshold, and the “off” count variable threshold. As the filtering coefficient is lowered, the incoming velocity data will become more dampened. Therefore, it is important to find a correct value that will filter out the noise that could cause the controller to malfunction but also provide realistic data. Data that is ‘smoothed’ out too much will cause the controller to spend longer times banking and not entering a bank. Therefore, this variable has an effect on the glider’s reaction time to encountering thermals. The count thresholds have a similar effect as they determine when the aircraft is allowed to switch between states. The simulation allowed for fine tuning of these variables so that the controller could be optimized for flight. Figure 15 shows the final result of simulation in which the controller tells the plane to bank, shown when the red

square wave is ‘up’, when it is expected to do so. In this case, it is desired to have the aircraft bank when its climb rate is greater than zero.

A preexisting 6 DOF mathematical model of the Fox Glider was modified to reflect changes made to the aircraft and updated sink polar data. The model was used in a simulation that tested a novel thermal-centering algorithm that would allow the glider to find a thermal’s center and navigate around it. This is believed to be accomplished by exploiting the observation that the updraft velocity within thermals increases as the plane moves radially inward toward the center. Therefore, by measuring the aircraft’s rate of change in climb or sink, the flight computer should be able to determine if the glider’s trajectory will take it out of the thermal or not. This idea is illustrated in Figure 16. Several iterations of this controller were tested using this simulation, some of which focused on keeping the bank angle constant after the center of the thermal had been found. The simulation showed that the algorithms were only marginally successful and that further refinement was needed. Therefore, integration of the thermal-centering algorithm was deferred until further testing can be done and the first thermalling controller build was implemented and tested in flight.

This simulation ultimately served as a very useful test bed that allowed us to detect modes of failure in these more complex control algorithms. Figures 17 and 18 demonstrate the plotted output from one of the control algorithms tested in this simulation.

Controller adjusts the glider's bank angle to achieve optimum trajectory.

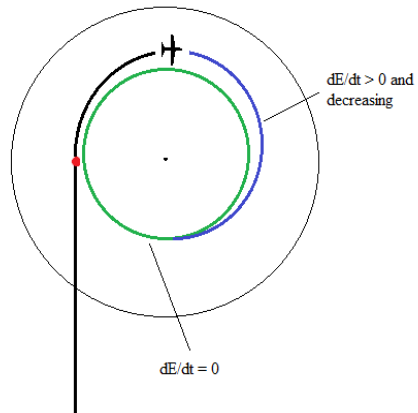


Figure 16 – Hypothetical Flight Diagram for Potential Control Logic Structure

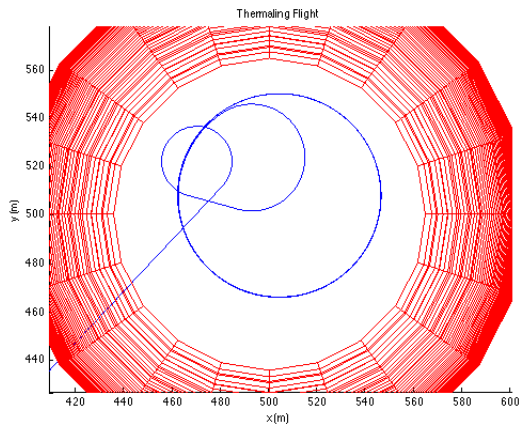


Figure 17 – Simulated Results, XY-Plane

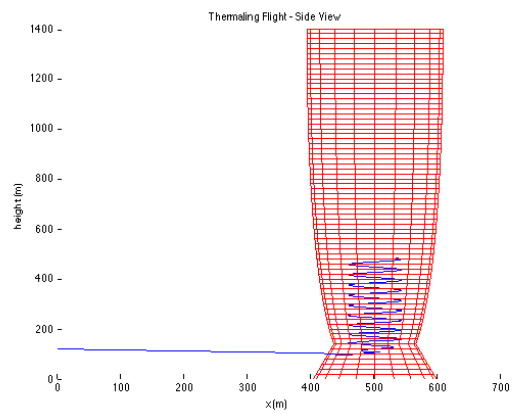


Figure 18 – Simulated Results, XZ-Plane

Testing

Throughout the course of this project, the capabilities of the plane and thermalling algorithm were assessed in flight tests that occurred at the El Tiro Glideport in Marana, Arizona. Tests were carried out to evaluate the characteristics of the plane (such as in the sink polar shown in the Platform section), the capabilities of the Mission Planner software (such as running an autonomous “racetrack” flight between a series of waypoints), and finally the efficiency of the thermalling algorithm. Before each flight, the Fox glider was calibrated for stability and linked telemetrically with the ground station, running Mission Planner software. By having the ground station run Mission Planner throughout the flight, flight data could be viewed in real-time to observe sink/climb rates, bank angles, and flight modes (among other pieces of data) on the ground station screen.

The testing procedure was as follows: after calibrating the plane and setting up the ground station, the plane would be hand-launched in Ardupilot’s “stabilize” mode, using the propeller to gain altitude in this pilot-controlled flight mode. After a reasonable height was achieved (typically 150m above ground level, but standardized at 100m for comparing thermalling flights), the propeller was powered off, and the plane was put in “auto” mode. This is the flight mode where the plane would follow the waypoint set dictated by the Mission Planner program unless overridden by the thermalling segment of the code. The plane would be allowed to continue in autonomous flight until it either reached too low an altitude or came too close to some other obstacle. Once a low altitude was reached (usually about 10m), the pilot (Jeffrey Koessler, who was also our project mentor) would initiate “stabilize” mode once again to take control of the plane and rise to a reasonable altitude. Then the process would be repeated until the test flight was finished.

The results of these flight tests were obtained from .tlog files of the data collected by the glider in flight. After flight, these files can be analyzed in Mission Planner's software to graph any value Mission Planner tracks versus time. Several of these flight tests verified thermalling flight, showing the characteristic 24 degree bank to the right which was the programmed thermal-centering protocol. Figure 19 shows the results of our final test flight, both in terms of an altitude versus time graph and a graph of filtered airspeed versus time. This filtered airspeed value is the same as the value seen by the autopilot, after undergoing the filter mentioned in the Control Logic section. After 5 counts of positive filtered airspeed, the autopilot switches into thermalling mode, which is shown by the red square wave on the same graph. The high value of this square wave indicates a time when the program is in thermalling mode, the low value indicates non-thermalling waypoint mode.

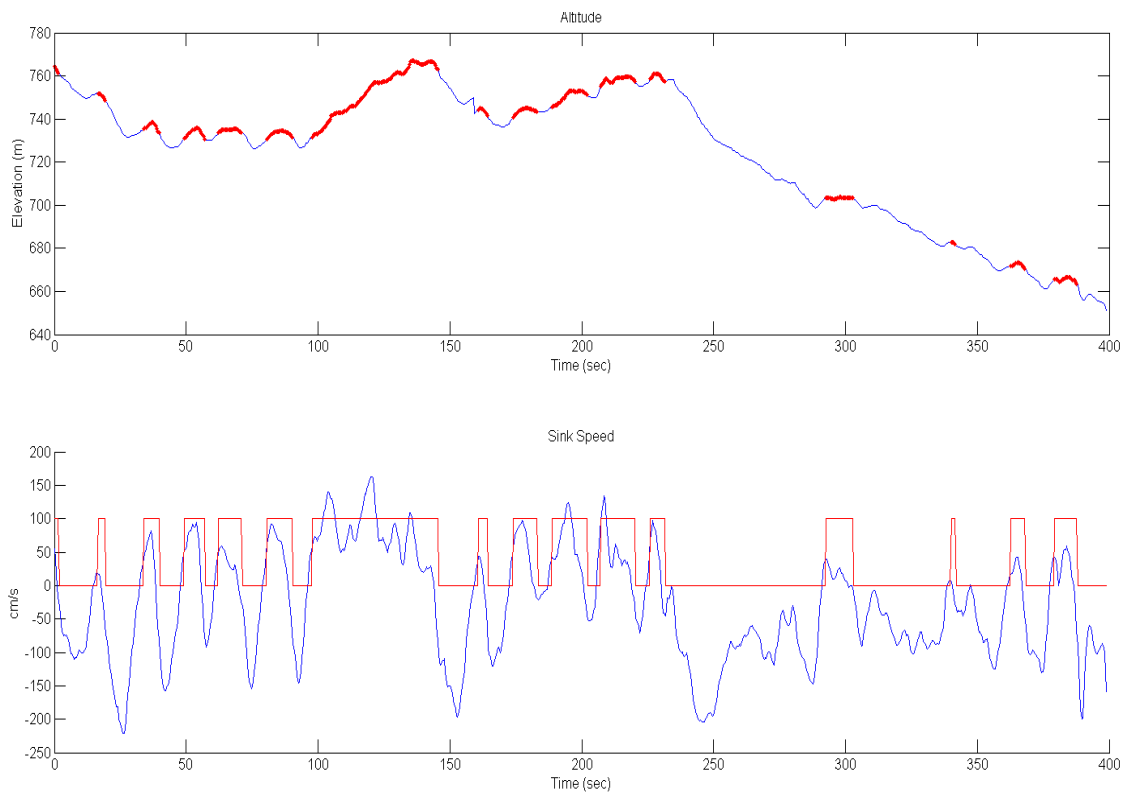


Figure 19 – Final Flight Test Results

The two key features to the altitude versus time graph in Figure 19 are the red sections (which indicate that the plane was thermalling) and the overall time scale. The red sections visibly maintain or even rise in altitude, indicating that the plane is truly drawing potential energy from the environment. The longest of these sections is roughly 50s long, during which time the glider gains about 30m in altitude without ever using the propeller. The overall time scale shows that the flight as a whole lasted 398s. When compared to other flights with the same initial altitude of 100m above ground (about 660m above sea level), we see that flights without the thermalling algorithm typically saw flight times around 120s.

Conclusion

The simulated and actual flight tests showed that the autopilot algorithm developed applies an induced bank angle to achieve thermalling flight when it enters a pocket of air with sufficient thermal energy. Based on the results of the final flight test, we see that rise rates of roughly 0.75m/s can be expected in a good thermal, compared to a sink rate of 1.5m/s for the same plane outside of a thermal. The benefits are apparent in the flight test results: the gliding flight time between propeller bursts is 3.32 times the flight time achieved without a thermalling algorithm. That means that the plane can go three times longer without the use of its propeller, which in theory more than doubles the energy efficiency of the plane.

Ideally, the flight time and efficiency could be increased even further by applying a more sophisticated thermal model or centering algorithm. The program could develop a GPS map of where it believes the thermal center to be, for example, and update that coordinate based on readings of upward wind velocity. Improvements such as these would increase the computation time required to obtain a bank roll command, thus potentially slowing the autopilot and making it “sluggish.” However, it is more likely that this computation time increase would be negligible, and the increased sophistication of the model would improve the efficiency in spite of the mild cost.

The other issue with a more complex algorithm would be the loss of universality. The great value of this program comes from the fact that it involves no values which are specific to the G-C Fox glider. In theory, then, the same program could be run on any other UAV running an Arduino autopilot. For any light UAV in reasonable weather conditions for thermalling, then, this program can be expected to double, if not triple its flight time between propeller bursts, and

increase its overall energy efficiency by more than 50%, allowing the UAV to save in energy costs and to stay on its mission longer.

Appendix

Appendix A

Measurements of the GC-FOX

Masses (in grams)

Wings (both)	600
Body	786
Battery	194
Total	1580

Fundamental Lengths (in cm)

Full Body Length	129
Span (tip to tip)	232
Nose to LE	40.3
LE to CG	4.0
Nose to CG	44.3

Horizontal Tail (lengths in cm, areas in cm²)

Elevator Span	54.0
Elevator Chord	3.6 - 4.4 - 3.6 (trapezoidal + an extra bit on either end of the stabilizer having spanwise length 2.5 cm)
Stabilizer Span (discluding elevator)	48.4
Stabilizer Chord (discluding elevator)	4.4 - 7.1 - 4.4 (trapezoidal)
Elevator Area	234.0
Stabilizer Area	278.3
Total Horizontal Tail Area	512.3
Aspect Ratio	5.69

Vertical Tail (lengths in cm, areas in cm²)

Rudder Chord (Root)	12.4
Rudder Chord (Tip)	6.1
Rudder Span (Vertical Dimension)	21.5
Total Vertical Tail Chord (Root)	20.5
Total Vertical Tail Chord (Tip)	10.0
Total Vertical Tail Span (Vertical Dimension)	24.5
Rudder Area	203.5
Stabilizer Area	124.4
Total Vertical Stabilizer Area	327.9

Aspect Ratio	1.41
--------------	------

Wings (in cm, areas in cm², refers to single wing if not otherwise specified)

Shape of each wing is assumed to be a swept trapezoid with a small triangle attached to the tip. C(tip) refers to the flight-direction dimension of the end of the trapezoid, and h(triangle) is the measurement of the triangle perpendicular to that. C(extend) is the theoretical chord you would get if you extended the wing trapezoid to the centerline of the plane. C(root) is the actual chord observed at the connection of the wing and fuselage). The same labels refer to the spanwise dimensions.

Span (extend to root)	6.1
Span (root to tip)	105.2
h (triangle)	4.7
Span (tip to tip)	232
C(extend)	23.2
C(root)	22.5
C(tip)	10.0
Total Wing Area (both wings)	3900
Area Ratio	13.8
Aileron Span	55.8 (measured parallel to Trailing Edge)
Aileron Chord	3.7
Aileron Area (both ailerons)	412.9
Spanwise Distance from tip to center of Ail	38.0
Span Dist from center of Ail to centerline	73.3
Span (Wing CM to root)	43.8
C(Wing CM to LE at the same spanwise point)	6.0
Span(Wing CM to centerline)	49.9

Fuselage (in cm, areas in cm², we assume an elliptical cross-section for calculating area)

Max Perimeter around Fuselage	54.0
Vertical dimension at largest cross-section	21.0
Horizontal dimension at largest cross-section	12.0
Max Frontal Area around Fuselage	198

Measured by Matthew Ashton and Jeffrey Koessler, November 27, 2013

References

1. Taylor, Colin G. "Lord Rayleigh." *Lord Rayleigh*. N.p., n.d. Web. 05 Dec. 2013.
2. Allen, Michael J. *Updraft Model for Development of Autonomous Uninhabited Air Vehicles*. Rep. N.p.: n.p., n.d. Print.
3. Kyle, Jason A. *Optimal Soaring by a Small Autonomous Glider*. Rep. N.p.: n.p., n.d. Print.
4. Stevens, Brian L., and Frank L. Lewis. *Aircraft Control and Simulation*. New York: Wiley, 1992. Print.
5. Naseem Akhtar, Alastair K. Cooke, and James F. Whidborne. "Positioning Algorithm for Autonomous Thermal Soaring." Cranfield University, Cranfield, England. MK430AL, United Kingdom, *Journal of Aircraft* Vol. 49, No. 2, 2012.
6. "Arduino - HomePage." *Arduino - HomePage*. N.p., n.d. Web. 03 Dec. 2013.
7. "Home." *ArduPlane*. N.p., n.d. Web. 05 Dec. 2013.
8. "Home." *Mission Planner*. N.p., n.d. Web. 05 Dec. 2013.
9. "Taoglas : The Antenna Solution Provider for M2M, GPS, GSM, WiFi, Public Safety, PIFA, PCB, CDMA, WCDMA, UMTS." *Taoglas : The Antenna Solution Provider for M2M, GPS, GSM, WiFi, Public Safety, PIFA, PCB, CDMA, WCDMA, UMTS*. N.p., n.d. Web. 05 Dec. 2013.
10. "Fox EPO Glider 2.32M (91.4in) Plug-n-Fly (v2) (EU Warehouse)." *HobbyKing Store*. N.p., n.d. Web. 06 Dec. 2013.
11. "HobbyKing Fox EPO Glider 2.32M - RC Groups." *RC Groups RSS*. N.p., n.d. Web. 06 Dec. 2013.

Layout of Individual Responsibilities

Matthew Ashton – Team Leader, Data Analysis Specialist

Nicholas Griffis – Autopilot Integration Subteam, Numerical Methods and Controls
Specialist

Joel Muetting – Autopilot Integration Subteam, Programming Specialist

Maira Garcia – Platform Hardware Subteam, Hardware and Device Specialist

Phillip Tindall – Platform Hardware Subteam, Flight Control and Navigation Specialist