ROB599/AEROSP740 M-Fly PADA GPS-Enabled Controls System Overview

Avinash Kumar
Aerospace Engineering
University of Michigan
Ann Arbor, MI
avikumar@umich.edu

Raphael Rosal
Aerospace Engineering
University of Michigan
Ann Arbor, MI
rosalrj@umich.edu

Erik Rutyna
Aerospace Engineering
University of Michigan
Ann Arbor, MI
erutyna@umich.edu

Venkat Subramaniam
Aerospace Engineering
University of Michigan
Ann Arbor, MI
vrsub@umich.edu

I. Introduction

One objective of the University of Michigan student project team M-Fly is to develop a lightweight (total weight less than one pound total) fixed-wing plane, known as a Powered Autonomous Delivery Aircraft (PADA). The PADA is designed to carry a payload before autonomously landing inside of a directed target on the ground. While M-Fly has developed the PADA full-scale model, the controller platform and documentation to enable autonomous flight and landing has not been shown to be effective at meeting the objectives. The authors of this paper were responsible for the design, implementation, and testing of the controller platform and documentation of their methodology and results. This report outlines the design justification of the PADA half-scale model, the wind tunnel airframe characterization of the PADA half-scale model, the theory and implementation of the controls system from provided texts, and the development of the onboard electronics and sensing array (the controller platform). The results, as well as limitations and errors of the implementation, are discussed in subsequent sections, and the future work is described in Section IX.

II. Powered Autonomous Delivery Aircraft Overview

The PADA is a small, powered, autonomously controlled aircraft that will be dropped from a larger "mothership", with the goal of delivering a fixed payload to an identified target zone on the ground. According to Society of Automotive Engineers (SAE) Aero Design competition rules, it must be dropped at an altitude of less than 50 ft and at least 100 ft away from the center-point GPS location of the target area. The PADA is configured in a delta wing configuration with elevon control surfaces. The delta wing enables a large wing area with a short wing span, which allows for high-lift scenarios with a short wing bending moment. The elevons allows for two axis of control - pitch and roll - simultaneously; this gives the PADA the ability to have control over two axes with two servos, instead of the tradition three servo control system for pitch and roll. The PADA has a shallow payload bay as a fuselage that can hold a fixed amount of payload. The aircraft's mass is capped at 16 ounces. Table I shows the full mass budget of the PADA, highlighting that the electronics system must weight 3.6 ounces or less. It shall be constructed of laser-cut craft ply wood, and glued together. The skin of the wing will be made of UltraKote bonded to the wooden planform. Figure 1 shows a picture of the half-scale model of the vehicle. One strategic goal of an open source design is to minimize the total weight of the control system as much as possible, which will increase payload capacity while maintaining the 16 ounce total mass. Currently, M-Fly has designed a simple manual control system with a commercial off-the-shelf flight controller for manual flight, but has not designed an open-source flight controller for this application. Our task is to develop a lightweight hardware system capable of all position, velocity, and attitude estimation of the aircraft, and write flight controller code that can use theses sensors to autonomously fly the aircraft from the drop location to a landing zone on the ground. M-Fly has performed manual flight tests of the vehicle and has determined that it should not have a pitch angle of $\pm 7\,\mathrm{deg}$, or a roll angle of more than $15\,\mathrm{deg}$ in either direction.

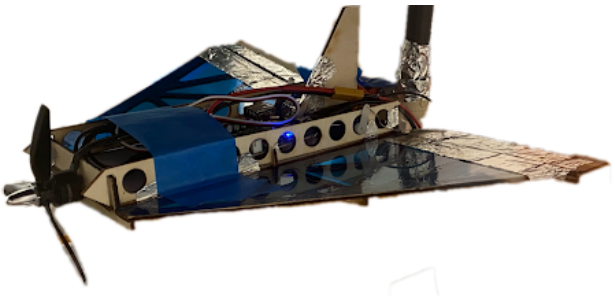

Fig. 1. 3D view of PADA scale model

III. Wind Tunnel Testing

A. Scale Model Design

The original PADA design provided by M-Fly was too large to fit in the wind tunnel. In order to build a model that would still produce meaningful results, the model was scaled via Reynolds number scaling. This process involved picking a scale factor to shrink the size of the PADA, and then varying the air speeds of the wind tunnel to hold the Reynolds number and advance ratio of the vehicle constant. A code was written to perform these calculations to find a model that would be

TABLE I
PADA MASS BUDGET TOTAL TO 16 OZ.

Component	Allocated Mass [oz.]
Wings	3.28
Fuselage	1.17
Elevons	0.4
Propulsion	1.3
Electronics	**3.6**
Empty Weight Estimate	10.57
Roughness Factor	1.05x
Final Empty Weight	11.10
Payload	4.90
Loaded PADA Weight	16

large enough such that it was easy to handle, but small enough to fit within the 2' x 2' wind tunnel. The result is a model with a scale factor of exactly one-half.

B. Scale Model Manufacturing

The scale model is almost an exact one-half scale version of the actual PADA with minor modifications to maintain the same material thickness as the actual PADA for strength and rigidity. In order to maximize testing time spent in the wind tunnel, the half-scale PADA was manufactured using M-Fly's original PADA drawings, scaled down 1:2, and modified to accommodate the laser-cut material thickness. The main structural parts were laser cut from 1/8" Birch Aircraft Plywood using University of Michigan Aerospace Engineering Professor Peter Washabaugh's laser cutter in the Aerospace 205 Class Laboratory. The laser cut parts were then assembled and affixed using superglue and the wings were covered in UltraKote. Aluminum tape was used to provide additional reinforcement to glued joints. It was also used for the elevon hinge. For the powerplant, a BX1306 2300KV BLDC motor was used along with a 4" 3-bladed propeller which, despite not being an exact 1:2 scale representation of the actual PADA's powerplant, was accounted for in a scaling script for testing. Finally, servos were mounted inboard the fuselage walls to control the elevon deflection and maintain the streamlined aerodynamics of the delta wing.

C. Testing Procedure

In an effort to properly tune the gains for the flight controller, the stability and control derivatives needed for the PADA needed to be found. In order to find the derivatives, the half-scale design was manufactured and a rigid rod was mounted behind the rudder to allow it to connect to the load cell for the wind tunnel. A diagram of the experimental setup can be seen in Figure 2, and a picture of the PADA during the testing process can be seen in 3.

The general testing procedure is as follows:

1) The wind tunnel load cells are calibrated, the PADA is mounted to the load cells, and the motor and BeagleBone Blue used to control the motor are wired and powered via external power supplies.
2) The PADA is given a pitch angle and elevon deflection angle in accordance to which condition is being tested.

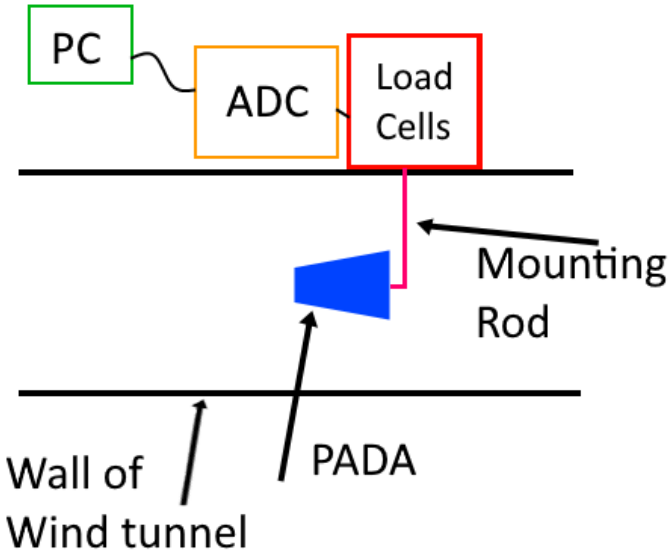

Fig. 2. A basic diagram showing the experimental setup for finding derivatives of the PADA in the wind tunnel.

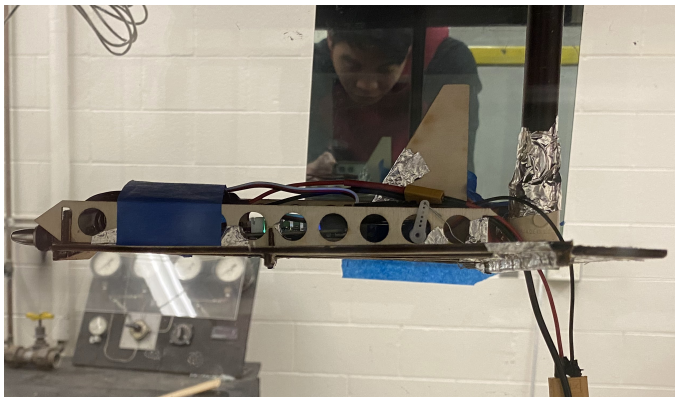

Fig. 3. The half-scale PADA model in the 2' x 2' wind tunnel.

3) The wind tunnel is adjusted to produce the appropriate air speed to match the Reynolds number scaling condition.
4) The flight condition is trimmed - the motor is either sped up or down using the rc_pilot library function "rc_test_escs" to a zero thrust/drag condition.
5) The data is recorded for this scenario is collected.

At this point the scenario is changed and the process is repeated. In total there are 3 air speeds, 3 pitch angles, and 3 elevon deflection angles. This produces a 3x3x3 testing matrix for a total of 27 unique flight conditions.

D. Testing Results Analysis

The testing campaign in the wind tunnel was unsuccessful. While data was acquired for all 27 test configurations, delays in manufacturing and testing forced us to be over time and unable to get meaningful results from the data. This in part because the data given by the 2' x 2' wind tunnel is only in 3-axis, and in order to compute all derivatives we needed a 6-

axis load cell system, but also due to the aforementioned time delays. A successful testing campaign would have included access to a 6-axis load cell system that would allow result in data to compute all stability and control derivatives. This data could then be used with a model of the PADA to produce the gains for our controller. However, without this data there was no need for a model, so instead the gains are tuned experimentally by flying the PADA and observing how it reacts and changing the gains in the controller accordingly.

IV. Controls Theory

A. Control Problem

The PADA is released from the "mothership" at a maximum altitude of 50 feet and a minimum distance of 100 feet from the target landing zone. In order to design a controller that can tackle the entire descent, the PADA's descent was split into two separate control stages: turning and constant-heading descent. When the PADA is released from the mothership, the PADA will likely not be following the exact heading to necessitate a straight descent to the target zone, so there needs to be controls designed to calculate the heading to the target and command the PADA to turn to that heading. Next, the PADA will need to have controller capabilities to establish its downward pitch angle to safely descend from the maximum height of 50 feet and reach the target zone at zero altitude accurately. Figure 4 illustrates the control problem in a piecewise setup.

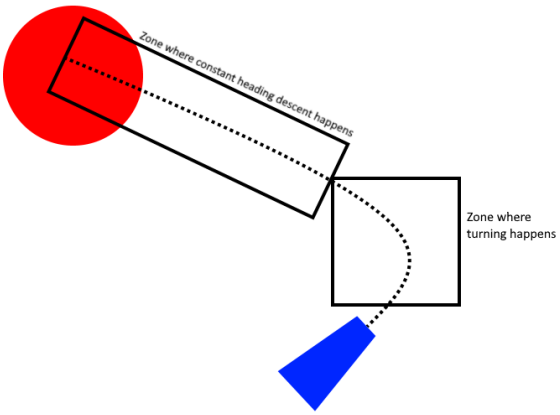

Fig. 4. The control problem can be viewed as 2 sequential, individual control problems: turning and descent.

B. Control Theory

The theory behind this problem can be broken down into two parts: a course hold problem, and a altitude descent problem.

1) Course Hold Control: The PADA only needs to track and hold a given set command as the vehicle is designed to turn once, and then hold that heading afterwards. This scenario is an example of course hold control. This type of control uses a set of k_i and k_p values such that the state can asymptotically track the set command given to the system; a block diagram representation of this can be seen in Figure 5.

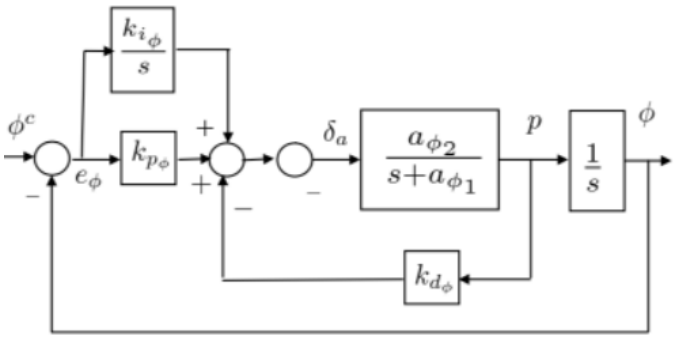

Fig. 5. A block diagram showing the entire closed inner and outer outer loops for course hold problems [1].

2) Altitude Descent Control: Altitude descent works similar to course hold by also choosing a set of k_i and k_p values such that the altitude component of the state vector asymptotically tracks the set command; a block diagram representation of this can be seen in Figure 6.

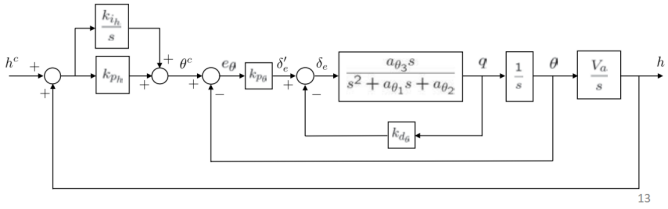

Fig. 6. A block diagram showing the entire closed inner and outer outer loops for altitude descent problems [1].

V. Control Implementation

Both problems described above are relatively standard controls problems in theory, without accounting for wind effects and other disturbances or biases. The methodologies of both problems used simple trigonometry to find the desired heading angles to command to the PADA.

A. Implementation for Turning

In order to turn the PADA after being released from the "mothership", the PADA needs to be able to determine its current heading and its target heading, and find the error between these two headings. This error between them is the amount the PADA needs to turn in order to be on a straight-line flight path to the target zone.

In order to do this, the PADA calculates an instantaneous averaged current velocity vector using its current GPS location at time t, and the GPS location of where it just was at time $t - dt$, and dividing this difference by time period dt between when those two position measurements were taken. This process can be applied to both the latitude and longitude coordinates to give both x-velocities and y-velocities for the

current and desired velocity headings. Using these velocities we can find the angle with respect to some origin on the Latitude-Longitude plane. The angles that these two headings have themselves are not needed, but the difference between them is important as this is the error in our state that is fed into the outer-loop controller.

$$\dot{x}_{current} = \frac{Latitude_t - Latitude_{t-dt}}{dt} \quad (1)$$

$$\dot{y}_{current} = \frac{Longitude_t - Longitude_{t-dt}}{dt} \quad (2)$$

$$\dot{x}_{wanted} = \frac{Latitude_t - Latitude_{wanted}}{dt} \quad (3)$$

$$\dot{y}_{wanted} = \frac{Longitude_t - Longitude_{wanted}}{dt} \quad (4)$$

$$\theta_{error} = arctan\left(\frac{\dot{y}_{current}}{\dot{x}_{current}}\right) - arctan\left(\frac{\dot{y}_{wanted}}{\dot{x}_{wanted}}\right) \quad (5)$$

This error angle is then fed into a PI outer-loop function which processes the error into a roll angle. Since M-Fly has limited the maximum roll angle to $\pm 15°$, the roll angle is capped at this value. Should the controller command a roll angle larger than $\pm 15°$, it will maximize the input to the inner-loop controller to this value, which prevents the PADA from commanding a turn faster than the vehicle is able to perform. The roll angle is then fed into the inner-loop controller code, authored by Derrick Yeo, where it maintains a course hold and turns the PADA. This process runs continuously to correct the current heading to the appropriate needed heading until the PADA is pointing towards the target zone.

B. Implementation for Descent

Now that the heading of the PADA points towards the target zone, the PADA needs to descend at the appropriate rate to land within the target zone. Using the GPS data of the PADA, the state error is set to the current altitude of the vehicle, as any non-zero altitude is automatically considered error. This error value at a given timestep is fed into the PI outer-loop function, which processes the error to determine the current pitch angle based on previous data. This pitch angle, also limited by the predetermined $\pm 10°$to avoid pitching past the limits of the PADA, is fed into the inner-loop controller code. This inner loop controller commands the elevon deflection angles to meet the computed pitch angle to slowly descend the aircraft towards the target altitude. It should be noted that this controller will only work if the aircraft is at a constant airspeed, which requires throttle control and an airspeed sensor. This capability is outlined in the future work section.

VI. Electronics and Sensing Equipment

We designed an electronics and power system that is extremely lightweight while still providing all the necessary sensors to estimate the aircraft's position and attitude. The controller is built on an Arduino Nano platform and can output servo commands to the two servos controlling the elevons. Figure 7 shows a block diagram of the electronics system,

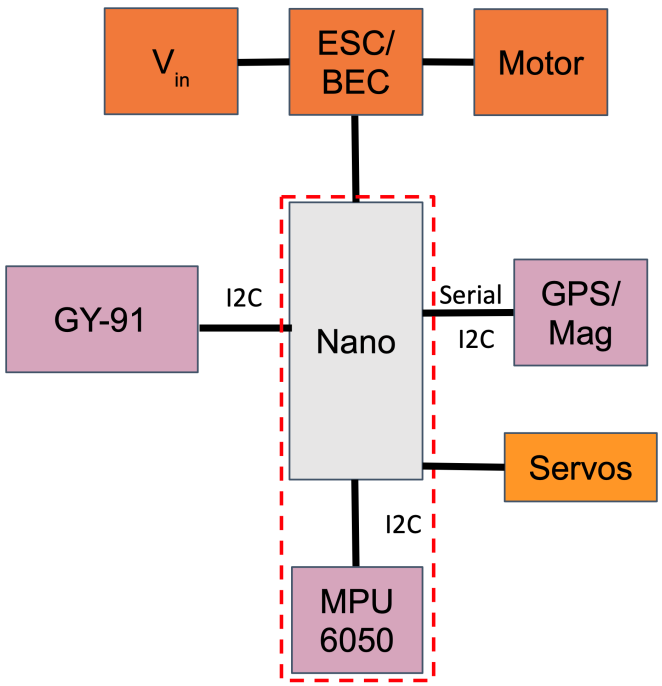

Fig. 7. Block diagram of electronics shows sensors, power system, and connections

which is powered by a 11.1 V 3-cell Lithium-Polymer battery. The battery powers the motor directly through the electronic speed controller (ESC), and powers the Arduino through the built-in battery eliminator circuit (BEC) on the ESC. The BEC provides a 5 volt supply to the Arduino Nano and servos. The total mass of the system is 1.2 ounces, which falls well under the 3.6 ounce mass limit given by M-Fly. This allows for a much larger portion of the vehicle to be allocated to the payload, which drives scoring decisions. The dashed red outline in Figure 7 show the sensors integrated into Derrick Yeo's original controller. All other components were added by the team to meet the functionality needed for our use-case. Figure 8 shows Arduino board setup and the senors we added.

A. Microcontroller and Sensors

As mentioned previously, the team built the flight controller based on a fixed wing controller platform created by Derrick Yeo. This controller was built on an Arduino Nano microcontroller with a custom built daughter board to allow for easy component integration. It uses an MPU-6050 accelerometer and rate gyroscope unit to assist in attitude state estimation. The MPU-6050 handles all measurement required for attitude state estimation. We added the following sensors to enable the controller to estimate the position of the aircraft:

- Beitian BN-880 GPS and Magnetometer Module: This module incorporates a GPS sensor and 3-axis magnetometer. We interface to the GPS sensor over an serial connection, which gets the raw GPS data packets. We use the TinyGPS+ Arduino library to read and parse

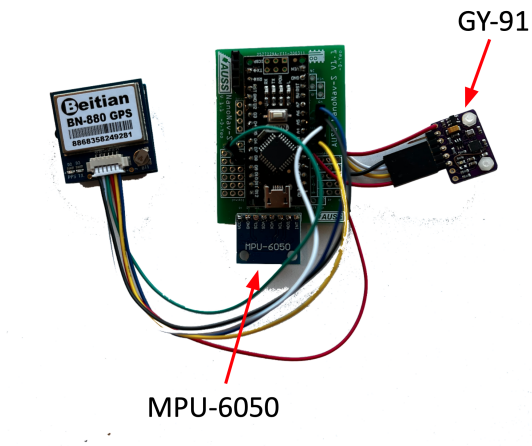

Fig. 8. Microcontroller shows GPS, MPU, and barometer board connected to custom daughterboard

the raw GPS data. This allowed for simple, but rather slow interpretation of GPS data. Using TinyGPS+ we were able to directly measure the aircraft's latitude and longitude, as well as measure its heading. One challenge with integrating this GPS module over a serial interface came from the availability of a single serial port on the Arduino Nano. This serial port is connected to the USB port of the Arduino, which we used to output controller values to the serial monitor. In basic testing of the GPS, we could the SoftwareSerial package to simulate a hardware serial connection, but this was not possible in the controller implementation. The inner loop code provided to us does not work when using SoftwareSerial. We remedied this in the final implementation by using the single dedicated hardware serial port for the GPS, but only connecting the GPS after the code of uploaded to the microcontroller. This allowed us to use the same serial port for interfacing with the computer and t he GPS. The downside of is that we cannot test the GPS over a serial connection using the controller code.

- GY-91 MPU/Magnetometer and Barometer Module: We used this module solely for the BME-280 static pressure sensor. We interfaced with the sensor using an Adafruit BME-2820 library over an I2C connection. We calibrated the altitude of the aircraft based on the barometer value at ground altitude and used a built in function to estimate aircraft altitude. This allowed for easy estimation of aircraft altitude, which can then be passed to the controller.

VII. Testing Plans

Due to weather constraints and the inability for M-Fly to test the "mothership", we plan to test our control system on the flight vehicle over the summer during M-Fly test flights. M-Fly plans to perform a summer test campaign where they will test the manual and autonomous flight control capabilities of the PADA and evaluates the feasibility of the control system for competition. This summer test campaign will allow for proper design reviews to take place, which the goal of flying the PADA with the our newly developed control system for the 2023 competition year.

VIII. Future Work

Based on this initial 7-week project, we believe the following modifications should be made to the controller to improve its performance:

- Airspeed and throttle control: The current design of the PADA does not make it feasible to add a pitot probe for airspeed measurement since the aircraft is extremely light and airflow from the propeller affects a large portion of the wing span. Due to this, we were not able to add an airspeed controller in a position that would allow for accurate airspeed estimation. Once the airspeed sensor is attached, we can then write a short airspeed control algorithm that will maintain the airspeed of the vehicle so that pitch and altitude can be better controlled. Once the flight controller is able to maintain the vehicle's airspeed, the altitude descent logic will allow for proper descent.
- Path planning: The current controller implementation simply reduces altitude and position error, which is reasonable for an initial prototype of the control system. However, to enable more accurate control of the aircraft, we can implement a path planner which can dynamically generate waypoints between the drop location and landing zone that can more accurately be followed by the vehicle. This will allow for the vehicle to safely land at a shallow glide scope instead of a potential harsh belly land.
- Overshoot prevention and correction: The controller does not have the ability to properly correct itself if the PADA 'misses' the target landing shoot due to overshoot. One way to prevent this is to write logic that can recognize a missed landing attempt and take the aircraft to a new point via Dubin's path and attempt to land again. We plan to give this recommendation to M-Fly as they further develop the control system.
- Use of numerical methods to develop plant model for gain-tuning: M-Fly has the ability to design the hardware and software required to autonomously land the PADA. In its current incarnation, M-Fly must experimentally fly the aircraft to determine control gains, which risks damaging the vehicle and is time-consuming. One major effort in the coming months is to develop proper stability analysis methods using Vortex Lattice Method (VLM) tools. These tools can quickly generate stability derivatives of any design and can be used to develop a plant model in Simulink for the aircraft. M-Fly can add the two implemented control systems to the plant model to simulate landings, which allows for gain-tuning through simulation.
- Use of magnetometer to compute aircraft heading: We have implemented a GPS-approach to estimate the heading of the flight vehicle. One option to improve the accuracy of this is to use the built-in magnetometer in either

the GY-91 board or the GPS. Since the aircraft flies in an outdoor environment, it will be able to measure heading based on Earth's magnetic field instead of potentially noisy GPS sensor data. However, the changing magnetic field from the nearby motor could affect the readings from the magnetometer.

IX. Conclusion

The main goal of this project was to develop a control system for the Powered Autonomous Delivery Aircraft (PADA) fixed-wing plane designed by the M-Fly student project team. The researchers were tasked with designing and manufacturing a half-scale model PADA to be able to test the aerodynamic properties of the vehicle inside the 2' x 2' wind tunnel. The PADA was constructed out of laser-cut ply wood, glued together with wood glue, and the wings were bonded using UltraKote. To test the half-scale PADA, wind speed, pitch angle, and elevon deflection angle were all varied and each configuration was saved in a dataset. This dataset requires additional time to be processed due to unforeseen delays in manufacturing and testing. In addition, the data obtained in the wind tunnel is only given in 3-axis, though to obtain stability and control derivatives, we require a 6-axis load cell system. From this 6-axis data we could establish a computational plant model which can then be used to tune the controller gains. Instead, we experimentally tuned the gains by flying the PADA and observing the stability in-flight.

To develop the controller code for autonomous turning and descent to the target zone, the researchers used the Course Hold Control and Altitude Descent Control control diagrams provided in Chapter 6 of Beard and McLain [1]. The controller was designed to segment the turning and descent control problems into two independent problems. To implement the theories, we used the GPS position data and trigonometry to determine the optimal pitch and roll angles for the elevons, while minding the angular saturation limits for the aircraft stated by by M-Fly. From here, the angles were fed into an inner-loop controlelr made by Yeo to command the servo motors and deflect the elevons by an angle to induce the appropriate pitch or roll angles.

The electronics and power system was designed with all of the necessary sensors to estimate position and altitude of the PADA. The controller can output servo commands to deflect the elevons. The total mass of the electronics system is 1.2 oz, which is significantly lower than the maximum mass of 3.6 oz required by M-Fly, so there is more payload mass available. The following sensors were added to enable the controller to estimate the position and altitude of the aircraft: GPS, magnetometer module, MPU, and barometer module.

The schedule of the testing and modifications did not line up with the available time to work on the PADA, so future testing and design plans needed to be specified for other M-Fly team members to use in the future. Due to weather constraints, the PADA will be tested over the summer during M-Fly testing sessions. The manual and autonomous flight capabilities of the PADA will be measured to allow for design reviews to be completed, with the goal of finalizing design prior to the 2023 competition for M-Fly. There are a few modifications to the controller that will improve the performance. The first modification is to add an airspeed controller to the full-scale model to estimate the airspeed accurately. Users will be able to develop a control algorithm to maintain the PADA's airspeed, and make it easier to descend properly. The second modification is to implement a path planner to dynamically generate waypoints between the drop location and landing zone for the PADA to follow. The current implementation is a greedy algorithm to reduce altitude and position error to get the PADA to the target zone, but a path planner may be better for the PADA to safely land instead of a harsh belly landing. The third modification is to add logic to the controller that can recognize an overshoot trajectory and reset the PADA to a new starting point via a Dubins path trajectory and attempt to land again. Currently, the controller cannot correct if the PADA misses the landing zone. The fourth modification is to use numerical methods to develop a plant model for tuning the controller gains. The last proposed modification is to use the magnetometer for heading measurement instead of using multiple GPS longitude and latitude points.

Acknowledgments

We would like to thank Professors Peter Gaskell and Derrick Yeo for their work in developing and teaching this course, and a secondary acknowledgement to Derrick Yeo for his help in providing both flight controller hardware and control loop code that was used in this project. Additionally, we would like to extend our gratitude to Akshay Mathur, who provided quick feedback and mentorship as our graduate instructor.

References

[1] Beard, Randal W. and McLain, Timothy W., "Small unmanned aircraft theory and practice." Princeton University Press, Princeton, N.J., 2012.