# System Design and Controller Development for a Small Unmanned Aerial Vehicle

Muhammad Immad

Professor Meyer Nahon

Bachelor of Engineering

Department of Mechanical Engineering

McGill University

Montreal, Quebec

A thesis submitted to McGill University in partial fulfillment of the requirements of the Undergraduate Honours Program

# Acknowledgments

I would like to express my appreciation and thanks to my supervisor, Professor. Nahon, for giving me this opportunity in the first place. His guidance, expertise, understanding and patience added significantly to my experience. I would also like to thank every single member of the Aerospace-Mechatronics lab for their continuing support.

I would like to extend special thanks to Mr. Waqas Khan for his help in system design and Mr. Eitan Bulka for his assistance in the development of the PID controller. My sincerest thanks go to Bassam-Ul-Haq, Adrian Battiston, Gareth Dicker for sharing their knowledge on the subject matter and positively contributing to my experience.

I would like to thank all my family and friends who supported me throughout my entire life, particularly, Shayan Imran, Vijdan Anjum, Mustafa Aslam, Ahmed Ashraf and Daniyal Taufeeq who have encouraged me throughout time. Lastly, I must acknowledge Arnaud Yoh Massenet-Oshima for his assistance in translating my abstract to french and Andi Rayhan for his advice regarding writing codes.

# Abstract

Small agile UAVs is a category of fixed wing aircraft characterized by high weight to thrust ratios and large control surfaces. They show good potential in the ability to combine speed, range and extreme maneuverability. The goal of the work presented here is to design a system capable of performing autonomous maneuvers and develop a PID controller for executing the maneuvers. The system was successfully designed and tested through piloted flight tests, while the controller was developed and tested using a highly accurate simulator. The controller developed is capable of performing a wide range of maneuvers including level flight, climb to new altitude, 180 degrees turn and a maneuver that executes all of these maneuvers one after another.

# Abrégé

Les véhicules aériens sans-pilotes (UAV) agiles et de petite taille font partie d'une catégorie de véhicules aériens à ailes fixes caractérisée par un ratio poids-poussée élevé et de larges gouvernes. Ces appareils présentent un grand potentiel pour allier une grande maniabilité avec une vitesse importante et une longue distance franchissable. Le but du projet est de créer un système capable de faire des manoeuvres automatiquement en utilisant des régulateurs PID conçus par nos soins. Le système en question est fonctionnel, il a pu être testé durant des phases de tests de pilotages manuels puis automatiques. Quant aux régulateurs PID, ils ont été développés et testés grâce à un simulateur virtuel avancé. Ces derniers sont capables d'exécuter un grand nombre de manoeuvres comme les vols en paliers, les changements d'altitude, les virages à 180 degrés mais aussi la combinaison de ces trois manoeuvres séquentiellement.

# Contents

# Chapter 1

# Introduction

Unmanned aerial vehicles have garnered attention from researchers due to their increasing potential applications. Traditionally, multi-copters were associated with extreme maneuverability while fixed wing UAVs were known for their range and speed. A particular category of UAVs, known as aerobatic UAVs has shown good potential in the ability to combine all three; range, speed and extreme maneuverability. Aerobatic UAVs are characterized by their large control surfaces and high thrust to weight ratios. This enables them to perform maneuvers that cannot be performed by regular fixed wing aircraft. Although aerobatic UAVs have been piloted by RC enthusiasts since a long time, researchers have shown particular interest in implementing autonomous controls on these UAVs. At the Aerospace-Mechatronics Lab at McGill University, one of the goals was to implement closed loop controllers for extreme maneuvers on aerobatic UAVs.

The objective of this work was to upgrade an existing platform at AML and to design a more robust system capable of implementing autonomous controls. This goal was followed by the development and implementation of simple controllers capable of typical maneuvers as well as modifiable to execute extreme ones.

Upgrading the entire system required choosing a suitable air frame, selecting precise sensors and electronics, and picking a suitable computer board. The work on controllers consisted of development and testing of controllers in a simulator and manual gain tuning to improve results. This discussion provides insight to the successful design implementation of the system as well as development and testing of the controllers in a simulator.

However, the controllers were not tested experimentally on the upgraded system during the time span of this work.

# Chapter 2

# Existing Platform

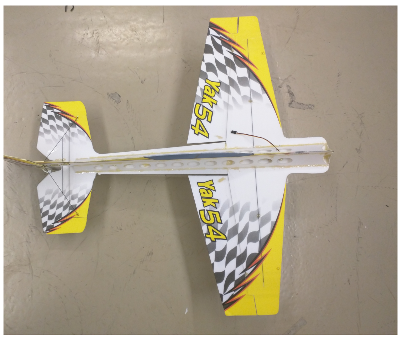
The old system at AML constituted of a Yak-54 air frame and an open-source ArduPilot Mega (APM 2.5) board. The Yak-54 used was manufactured by GreatPlanes and was discontinued during the time span of the project at AML. On the other hand, the APM 2.5 was revised by the manufacturer. The system, in general, proved to be a good starting platform for previous students at AML to test closed loop controls. Additionally, it proved to be a good base for acquiring knowledge and expertise in order to design a new system for the project in discussion.

## 2.1 Air Frame

Successful execution of extreme maneuvers requires a highly stable and light weight air frame. Both these features were offered by GreatPlanes' Yak-54. The Yak-54 was made from Pro Performance foam - GreatPlanes' version of Expanded Polystyrene - and was 3 mm thick. Weighing under 90 grams, the Yak-54 offered great performance for its recommended configuration. However, Yak's performance dropped when it was upgraded to the custom configuration required by AML. Yak-54 fluttered during flight tests resulting in reduced control, besides damaging and deforming the air frame. The flutter would also induce disturbances and inaccuracies in the readings obtained by internal Inertial Measurement Unit (IMU) of the board. Consequently, the YAK-54 required reinforcements for the base model, which were introduced by previous students at AML.

The first step in the larger goal of achieving a new system required manipulating the custom configuration and flight tests performed previously at AML. Therefore, a new base model of Yak-54 was upgraded to resemble the previously enhanced ones. This required structural reinforcements using carbon rods as well as reinforcements on the control surface hinges alongside the relocation and addition of new electronics.

However, there was one noticeable change in the latest Yak-54 configuration: the addition of a multiplexer for closed loop controls. The addition of the multiplexer was due to the lack of on board options for controlling multiple control surfaces via closed loop. The addition of multiplexer introduced instability, necessitating slight redesign and modification of the configuration. Figure 2.1 shows the Yak-54 air frame while the schematic of the multiplexer is given in Figure 2.2.


(a) Base model

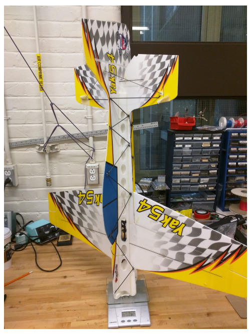

(b) Reinforced model

Figure 2.1

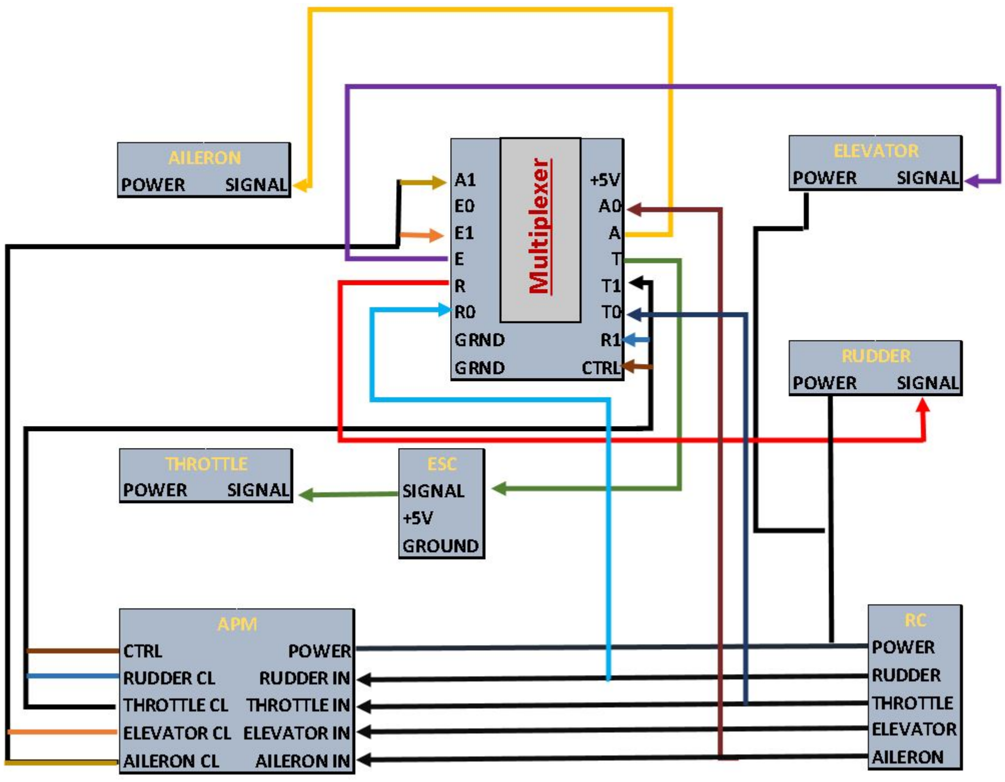


Figure 2.2: Schematic of the multiplexer

## 2.2 ArduPilot Mega (APM)

The ArduPilot Mega uses a modest 8 bit ATmega2560 as the main processor while the ATmega32u4 is tasked to handle USB functionality. It has an internal IMU (MPU-6000) and a barometric pressure sensor. APM also offers ports for an external GPS and compass [4]. APM provides connection ports for multiple motors and servos, however, the configuration at AML was modified to power the motors and servos separately while using the signal and ground pins of the APM. Figure 2.3 displays the ArduPilot Mega used at the Aerospace-Mechatronics Lab.

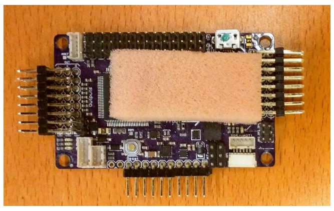


Figure 2.3: ArduPilot Mega

## 2.3 Electronics

The electronics utilized by the old platform included an external MicroStrain IMU (3DM-GX3-25-OEM), a GTPA010 GPS, a MaxSonar-EZ3 ultrasonic sensor, a multiplexer, a power distribution board to power various components and a RC receiver and an external SD card module.

The external IMU was used for tracking the body rate and acceleration of the aircraft, the sonar was used to measure distance from lowest part of the aircraft to the ground, and the SD card module was responsible for logging flight data for post flight analysis. The IMU, GPS and sonar are displayed in Figure 2.4.

(a) IMU

(b) Sonar

(c) GPS

Figure 2.4

## 2.4 Post Flight Data Analysis Code

The flight data is logged in binary format as it is considerably faster than logging in American Standard Code (ASCII). Hence, post processing of the data requires it to be first converted into readable format. This is done via an open-source MATLAB code. However, many times it requires trial and error and modification of the binary data in order to process the data using the code. This is due to system disturbances such as temporary loss of connection between the Arduino board and other sensors or the RC receiver.

## 2.5 Ground Station

The ArduPilot was capable of use with various ground stations, with the most popular one being the Mission Planner. Initially, Mission Planner was used to configure and test the APM board but was aborted afterwards. After initialization, the Arduino environment was used to modify the open-source code for ArduPilot in an effort to develop and implement closed loop controllers. This method was implemented till the end of the life of this platform.

Modifying the code using the Arduino environment necessitated adding in extra effort in order to develop and implement controllers due to compiling or coding errors. Moreover, implementing the use of external modules such as the GPS was generally more demanding as well.

# Chapter 3

# Updated Platform

The increasing interest of researchers and enthusiasts alike in controls of small UAVs has lead to the development of newer, more robust air frames as well as purpose specific computer chips catered for autonomous control. While the Yak-54 powered through Arduino's Ardupilot offered a good initial base, it had slowly become outdated for advanced controls. Hence, the Aerospace-Mechatronics Lab (AML) at McGill University began to explore viable options for both: a new air frame; and a more powerful computer platform.

## 3.1 The New Air Frame

The aging Yak-54 required an upgrade to a more robust 3-D flight capable foam air frame. Fortunately, foam based air frames were widely available from many different companies and distributors such as HobbyZone, ParkZone, West Michigan Park Flyers (WMPF), GreatPlanes, Eflite and many more. With various available options, choosing an air frame was a difficult task, however, AML had the experience of flying the YAK-54 to learn from in order to choose the correct air frame. Two of the major disadvantages of YAK-54 were how easily it was damaged and the difficulty of repairing it. Therefore, the ideal air frame would be made of a different material than Expanded Polystyrene and would offer more resistance to damage.

Surveying different RC lobbies and manufacturers yielded in discovering many different foam types. The major ones were: Elapor, Expanded PolyStyrene, Styrofoam, Expanded

PolyPropylene, Z-Foam and Expanded PolyOlefin. Of them all, research suggested that the most damage resistant was Expanded PolyPropylene (EPP). Hence, AML started to search for 3-D air frames made from EPP. Later, AML keenly looked at air frames from two manufacturers: MS Composit; and West Michigan Park Flyers and finally converged on the McFoamy air frame by West Michigan Park Flyers.

Prior to converging on the McFoamy, another aircraft closely examined was the Yak-55 EPP. However, the final decision was to adopt McFoamy considering it's used by The University of Sherbrooke, which is in collaboration with the Aerospace-Mechatronics Lab at McGill University. Figure 3.1 shows the McFoamy air frame.

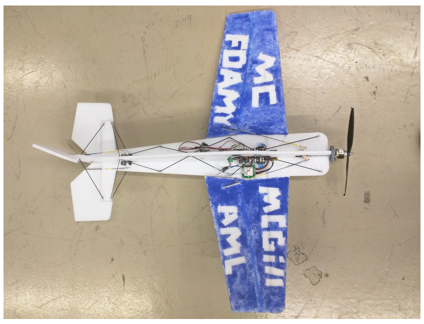


Figure 3.1: McFoamy air frame

## 3.2 Air Frame Difference

The Yak-54 and McFoamy are both capable of 3-D flights and are built for extreme maneuverability. However, their performance varies depending on the purpose they are used for. At AML, the comparison and analysis is based primarily on two sources: feedback from a professional, highly experienced pilot; and how well the aircraft performs experimentally.

The comparison of the two aircraft is based on flutter, weight and structure, alongside the ability of the aircraft to be highly stable during maneuvers. Furthermore, the air frame should offer resistance to impacts and be easily repairable within a short time interval. Throughout this section, it should be kept in mind that the comparison made is between the base foam-only models. Moreover, the enhancements made to the aircraft are based on the requirements of AML and are not necessarily the same as the ones proposed in enhancement kits available from the manufacturers.

McFoamy's 9 mm air frame offers greater resistance to flutter than the YAK's 3 mm air frame. While both the aircraft require carbon rod enhancements, McFoamy offers advantage in two ways. Firstly, McFoamy can be reinforced with fewer carbon rods. Secondly, thinner carbon rods can be used. Hence, McFoamy offers a reduction in the total weight added on the air frame during the enhancement phase.

The base configuration of Yak is approximately 30 grams lighter than that of McFoamy. However, Yak is longer in length while being more prone to flutter. These two properties require the Yak to be reinforced with a greater number of carbon rods, as well as heavier carbon rods, which take away from the advantage of a lighter base air frame.

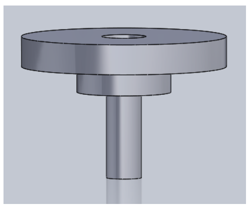
The general structure of the air frame of McFoamy allows for easier sensor mounting and increases the accessibility of the mounted components. Whereas, the Yak has an enclosure above its horizontal axis resulting in difficulty in mounting electronics. However, the Yak's structure offers a motor mount that can be easily adapted to different sized motors, while the standard mount on McFoamy cannot be easily adapted to larger sized motors. Hence, a new motor mount was designed for use with McFoamy, which is shown in Figure 3.2.

McFoamy is made using Expanded Polypropylene (EPP) which offers much more resistance to impacts than the Pro Performance Foam used to construct the YAK. Also, it is easier and less time consuming to repair the McFoamy after a flight crash compared to the Yak.

Finally, the pilot's feedback for McFoamy was very positive as well. These added advantages make McFoamy a more suitable option to use at AML for autonomous controls.

### 3.2.1 Motor Mount

The standard motor mount on the Mcfoamy is designed for much smaller motors than the rimfire 400 used at AML. For this reason, a new motor mount was designed and 3D printed at AML. The motor mount weighs approximately 20 grams and allows for mounting larger motors. Figure 3.2 shows the motor mount used for McFoamy.


(a)

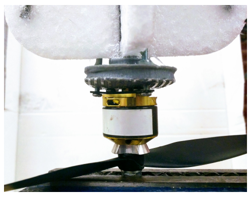

(b)

Figure 3.2: Motor mount for McFoamy

## 3.3 Structural Reinforcement

The McFoamy platform offers much more resistance to flutter compared to the Yak-54. However, structural reinforcements are still required for the air frame. McFoamy is reinforced with 1.5 mm carbon rods both above and below the horizontal plate along the fuselage. The air frame has pre-cut slots for inserting the carbon rods along the length of the top fuselage as well as the horizontal plate. Hence inserting the carbon rods along the top fuselage is slightly easier. Since some components including the battery are also mounted along the bottom fuselage, reinforcements are required along the bottom as well. The bottom fuselage was cut with care to match the top fuselage and have a symmetrical distribution about the horizontal plate as much as possible. Special care was taken while gluing the carbon rods together. Delaying in inserting the carbon rods on either side of the vertical plate of Mcfoamy or unsymmetrical top and bottom fuselage reinforcements can both cause the air frame shape to deform affecting aircraft performance. Figure 3.3 shows the reinforced structure of McFoamy.

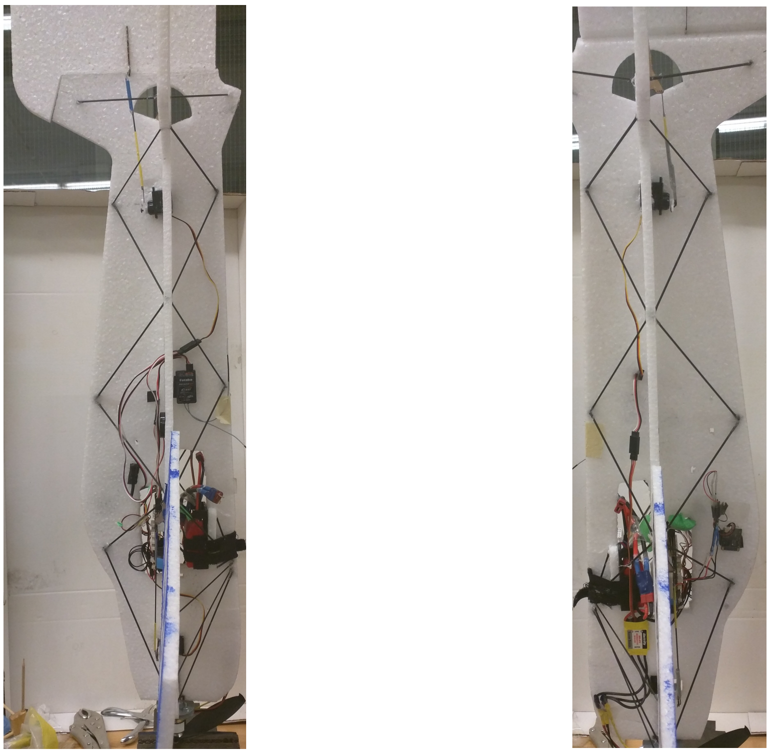

(a) Left Side (b) Right Side

Figure 3.3

## 3.4 Auto Pilot

The end goal of this work was to be able to have a system capable of extreme maneuvers and closed loop controls. Hence, the required auto pilot would need to have the processing power, offer ease in control implementation, be light in weight and at the same time offer a wide range of internal sensors. At AML, the auto pilots were roughly sorted into two categories: open-source auto pilots including the APM, PX4, Navio, Paparazzi UAV, Aero Quad, Pixhawk and a few more; and non open-source ones including the Procerus Kestrel, Cloud Cap Piccolo, MicroPilot MP Series and UNAV 3500.

Naturally, the open-source platforms offered a much more affordable price point generally ranging between \$100 - \$300 compared to the alternative category which started close to \$1000 with some priced upwards of \$20,000. Additionally, some among the latter required documentation to order the auto pilots. However, the non open-source ones generally came equipped with more accurate and advanced internal sensors than the open-source ones.

AML had a positive experience working with open-source auto pilots and decided to continue looking at the same category. One of the major criteria for AML was to have an auto pilot that would allow easy integration of sensors and closed loop controls and also be actively developed. For this reason, AML focused on newer boards and those which did not require an additional external board such as the Arduino or Rasberry Pi. Furthermore, AML preferred boards that offered easy cross platform functionality without limiting itself to any particular environment such as the Linux environment. Therefore, AML was mainly limited to PX4FMU and Pixhawk, with the latter being the final choice since it allowed for running the PX4 Flight Stack firmware as well.

## 3.5 Pixhawk

The auto pilot used for the new platform is the Pixhawk (PX4). Compared to the previously used APM, it offers a faster processor, an extra IMU for increased accuracy and is generally much more user friendly with easily accessible ports for external sensors. Being a newer board, the Pixhawk platform is more actively developed compared to the APM - which is slowly being outdated.

**Hardware**

The Pixhawk Autopilot is a combination of PS4FMU and PX4IO modules and is purpose built for autonomous motion. While the Pixhawk can be utilized for any movable platform, it is most commonly used on small UAVs [2].

Pixhawk uses STM32F427 as its main system-on-chip and houses a 180 MHz ARM Cortex M4 with 256 kb sram. Moreoever, it houses STM32F100 as its fail safe system-on-chip and works with 24 MHz ARM Cortex M3 as the CPU and 8 kb sram [1]. Additionally, Pixhawk is equipped with various on board sensors including two inertial measurement units (IMU), gyroscope, magnetometer and barometric pressure sensor. The Pixhawk also offers ports for external sensors, motor and servo connections and a SD card logger.

Pixhawk also allows for powering the electronics through its power rail, whereas the previous platform required powering the components separately. This allows for significant

(a) Without external case (b) With external case

Figure 3.4

weight reductions as fewer wires are used. Figure 3.4 shows the Pixhawk.

#### On Board Software

At AML, the Pixhawk platform runs the PX4 open source software. The software utilizes the PX4 flight stack and PX4 middleware and runs on top the NuttX operating system. The PX4 flight stack consists of algorithms for guidance, navigation and controls of autonomous vehicles while the PX4 middleware is a collection of device drivers for embedded sensors and a publish-subscribe based middleware that enables the sensors to communicate with the applications running the flight controls.

## 3.6 Sensors and Electronics

Flight tests and the corresponding data analysis are required to evaluate both the system and controller design. The data is then used to modify or redesign the system. The Mcfoamy platform utilizes various sensors for flight data collection to estimate the acceleration, spatial positioning and other states of the aircraft. The data is retrieved from the SD card mounted on the Pixhawk.

### 3.6.1 Internal Sensors

For the flight tests performed by AML, the internal IMUs on board the Pixhawk are used for tracking the body angular rates and acceleration.

Additionally, the internal barometric pressure sensor is used to track the height. While the data in the z coordinate direction can also be taken from the GPS, the pressure sensor produces more accurate results for the purposes the aircraft is used at AML.

### 3.6.2 GPS

AML utilizes 3DR GPS Module (uBlox NEO-7 w/ Compass) to track the spatial positioning of the aircraft as well as the heading. The data collected by the GPS is then used to plot the flight trajectory of the aircraft for the maneuvers. Figure **??** shows the GPS used.

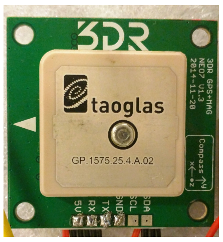
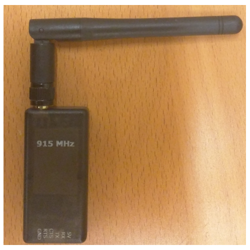


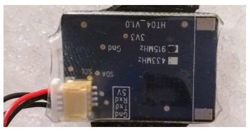
Figure 3.5: GPS

### 3.6.3 Telemetry Link

Hobby King's Micro Telemetry radio Set (915Mhz) is used for live feedback from the aircraft to ground station during flight. This telemetry module offers many advantages. Firstly, the communication link used is the Mavlink Protocol and can be directly used for the Pixhawk via the software QGroundControl. Secondly, it offers an integrated antenna resulting in the external transceiver on the aircraft to weigh 1.6 grams which allows for weight reduction [5]. Figure 3.6 shows the telemetry modules.



(a) Ground module



(b) Aircraft module

Figure 3.6

### 3.6.4 Additional Electronics

AML choose to use the same actuators for the new platform as the ones used for the older one. This decision is credited to their light weight and overall performance. All control surfaces on the aircraft are controlled by Hitec's HS-65HB servos while the Rimfire 400 motor manufactured by Great Planes is used for rotating the 10 x 4.5 S propeller.

The Electronic speed control module chosen is Electrifly's (Great Planes) SS-25 and the overall system is powered using a 3-cell lithium-polymer battery of 850 mAh manufactured by the same company.

## 3.7 Location of the Electronics

For the successful execution and testing of maneuvers, it is extremely important that the aircraft offers high stability at the desired maneuvers and the sensors provide an accurate measurement of the controlled variable. Hence, the mounting and location of the electronics on the aircraft is a crucial aspect for system design.

For optimum performance, the McFoamy air frame user manual recommends a range for the center of gravity (CG) to lie in . Therefore, one of the criteria in mounting the electronics is to ensure that the center of gravity lies within the recommended range. At AML, the center of gravity is made to be exactly in the middle of the range to allow room for shifting components if necessary. However, the need for individual sensors to have a particular orientation for accuracy coupled with the need to maintain zero roll makes the criteria to restrict CG to the middle of the range a challenge.

Furthermore, the mounting of electronics needs to allow for the use of minimum length of wires. Excessive length of wires increase the weight of the aircraft significantly, at the same time increasing difficulty for wire management as they pose a risk to interfering with the control surfaces.

The choice of location is further restricted due to structural reinforcements as the optimum design for reinforcements interferes with control rods in addition to increasing the weight imbalance. This problem is solved on McFoamy by placing the control surface servos much closer to the control surfaces at the tail, while the servo of the ailerons is mounted as recommended by the air frame user guide. Mounting the servos closer to the tail control surfaces also decreases the weight of the carbon control rods as shorter lengths are required.

Another important criteria is the accessibility of the mounted components and their connections. This criteria is particularly important during flight tests, where there could arise a need to either shift the components or replace the connections.

Lastly, the components are further adjusted and the process repeated based on the feedback from the professional pilot after the first flight of the aircraft.

## 3.8 McFoamy 3D Model

Accurate aircraft property inputs to the simulator, such as the CG, require a detailed 3D model of the aircraft. The McFoamy aircraft used at AML was designed using SolidWorks and contains details and locations of the electronics as well. The CAD model of the aircraft was generated using material properties of EPP foam available commercially [9]. The model also utilizes some off-the-shelf component CADs [10], however, their mass properties were verified experimentally at AML. Figures 3.7 and 3.8 show the 3D model of the non-reinforced McFoamy along with the electronic components.

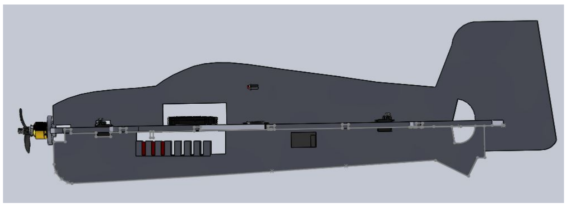


Figure 3.7: Left side

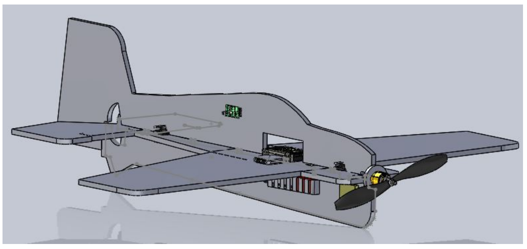


Figure 3.8: Right side

The CAD model can further be used to relocate on board components in an effort to reduce moments of inertia and have a more responsive aircraft. The detailed model can also be used to perform basic stress analysis which would allow AML to reinforce the aircraft more efficiently. Table A.2 in Appendix A shows some of the aircraft properties.

## 3.9 Ground Station

The defining of the orientation of the Pixhawk, calibration of sensors and parameters and any other command or setting execution requires a ground station software. While there are many ground station software available, two full featured desktop based ones were considered for use with the Pixhawk at AML. Namely, QGroundControl (QGC) and MissionPlanner. Both the platforms can run the Pixhawk PX4 Flight Stack,however, QGroundControl utilizes the MAVlink protocol having PX4 Flight stack as its primary focus in addition to offering the flexibility of working with mobile devices. Therefore, QGroundControl was chosen as the ground station for the McFoamy platform.

### 3.9.1 QGroundControl

QGroundControl offers both piloted and autonomous flight control for any MAVlink enabled platform [3]. Perhaps the biggest advantage this ground station provides over the previously used environment is it's user friendly and intuitive interface, removing the need to understand and modify low level codes in order to change settings or execute commands. Moreover, it offers multiple flight modes that can be switched between using the switches on the RC remote controller, allowing for the execution of various autonomous maneuvers during a single flight.

Furthermore, it provides telemetry link support and displays the feedback data in a cockpit-like fashion. At the same time, it provides error messages and important information such as battery power levels. Using QGC, the aircraft can be commanded to perform certain desired maneuvers if power levels drop below a certain percentage. For example, the aircraft can be made to land at its current position or return to its initialization position if the power level drops below 15 percent.

Another of its advantages is that it can be used to define the orientation of the Pixhawk, if different from the default. This allows for the flexibility of mounting the Pixhawk in different ways depending on the type of air frame or structural restrictions.

Compared to the previous set up at AML, using QGC with Pixhawk allows for having multiple flight modes. These flight modes remove the need for a multiplexer altogether allowing for weight reduction as well as easier and more reliable transition between different closed loop maneuvers.

QGC allows for calibrating the radio control through the software. This feature is particularly useful if additional trims are required for control surfaces due to construction errors or accidents. Trims can be copied off from the radio control to QGC, and then further onto the board to level the control surfaces. Copying the trims from the QGC to the board allows for the trims on the radio control to be set back to their centered position, at the same time keeping the control surfaces leveled. Figure 3.9 shows the QGroundControl start up interface.

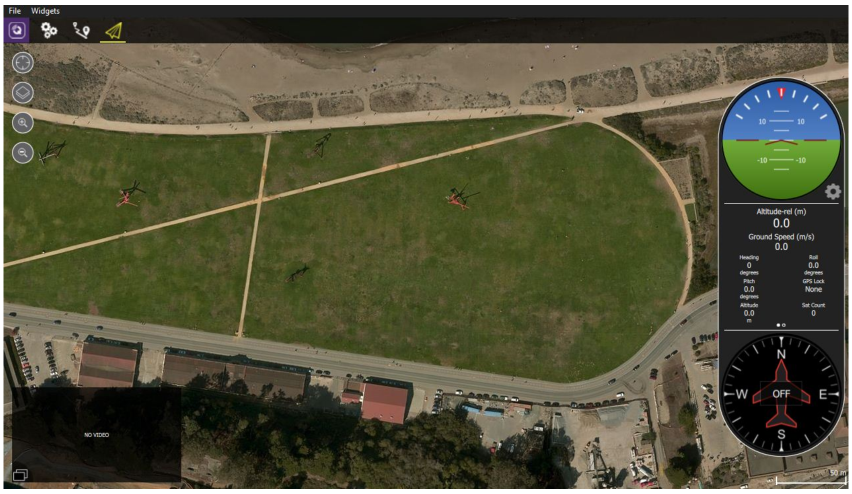


Figure 3.9: QGroundControl

## 3.10 Plotting Software

Plotting software are required for analyzing bench test and flight data. Various open-source scripts and web based softwares exist, however, the two plotting methods used at AML were FlightPlot software and an open-source python script called *sdlog2_dump.py* [7]. While the FlightPlot is generally easy to use and does not require conversion from binary format, it is web based and limited to the pre-programmed options and plots available [6]. Therefore, the python script was used to convert the PX4 Log files to CSV format, and then the results were plotted for analysis. Figure 3.10 shows a flight trajectory generated using the CSV file from the bench test data. The objective of this particular plot was to ensure functionality of the GPS and the trajectory was intended was an approximate circle.

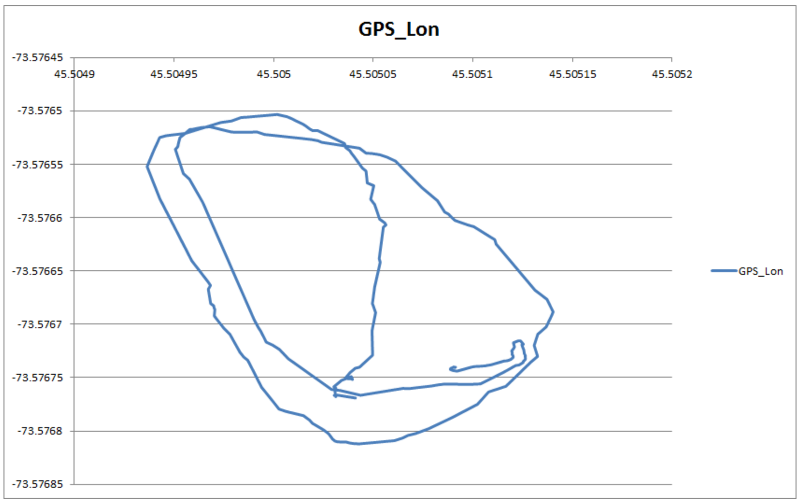


Figure 3.10: Trajectory during bench test

# Chapter 4

# Development and Simulation of a PID controller

One of the goals of this work was to develop a simple PID controller capable of demonstrating typical maneuvers performed by any aircraft. Although the controller can execute a wide range of maneuvers, the ones demonstrated here are level flight, climb to new altitude, 180 degrees turn and a maneuver that enables the aircraft to perform circuits, level flight and climb to new altitude one after another.

The controller presented here can be broken down into two major parts: a flight trajectory generator, which generates the aircraft's reference attitude for a maneuver; and an aircraft surface deflection controller, which computes proportional, integral and derivative responses of the error in attitude to output control surfaces deflections. Figure 4.1 shows the schematic diagram of the controller.

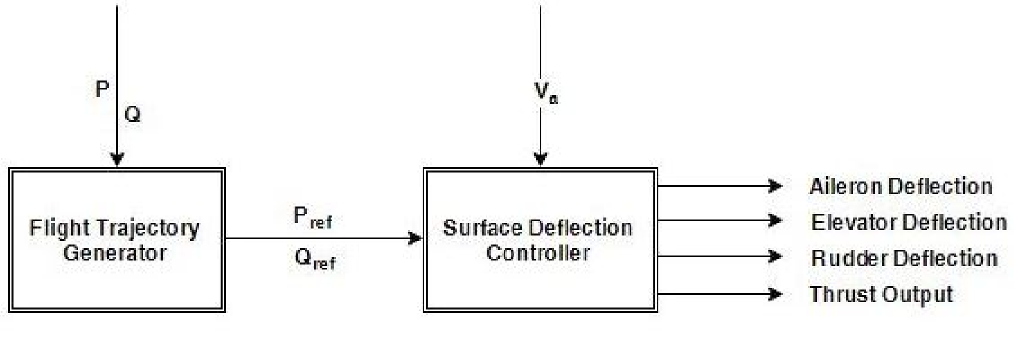


Figure 4.1

P is the position vector of the aircraft, Q represents the Euler angles of the aircraft and $V_a$ is the aircraft velocity.

## 4.1 Flight Trajectory Generator

The flight trajectory generator is tasked to output reference attitudes for a particular maneuver and feed it directly to the surface deflection controller. The generator relies on inputs from Khan's simulator to execute it's functions, namely, the spatial position and the attitude of the aircraft. It uses the spatial positioning of the aircraft to initialize a particular maneuver, then generates the required desired attitude based on the physics of the maneuver and outputs this attitude as a reference to the surface deflection controller.

As an example, when the aircraft crosses a predetermined x-coordinate position, the generator initializes the transition to a 180° turn; as a result, it outputs a reference roll of 20 degrees, and a change in yaw of 180°. Figures 4.2 and 4.3 show the flight trajectory and plots for two 180 degree turns. The green line represent the desired value while the blue line represents the actual ones. In the demonstrated circuit, the aircraft initially stabilizes at an altitude of 0 m with desired roll and yaw of 0°. Once the aircraft has traveled 300 m in the x direction, the flight generator prompts it to change its' roll and yaw angles to 20° and 180° respectively. For the same turn, as the aircraft crosses the 300 m x-coordinate line again, it is prompted to maintain a yaw of 180°, while the desired roll is set to 0°. The aircraft performs a similar turn after traveling 300 m in the negative x direction, except that for this turn, the desired yaw is set to 0°.

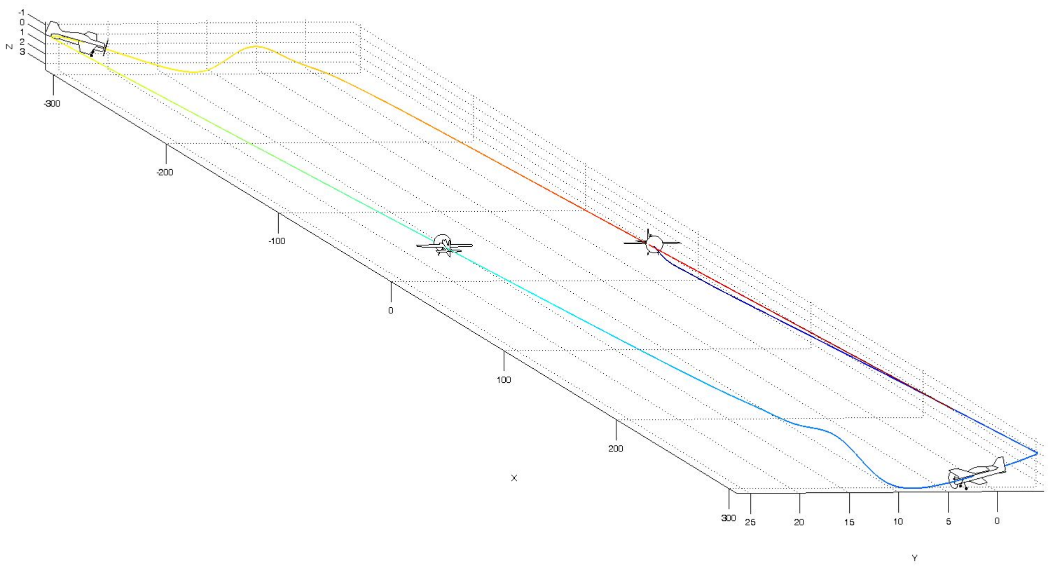


Figure 4.2: Flight trajectory for a circuit.

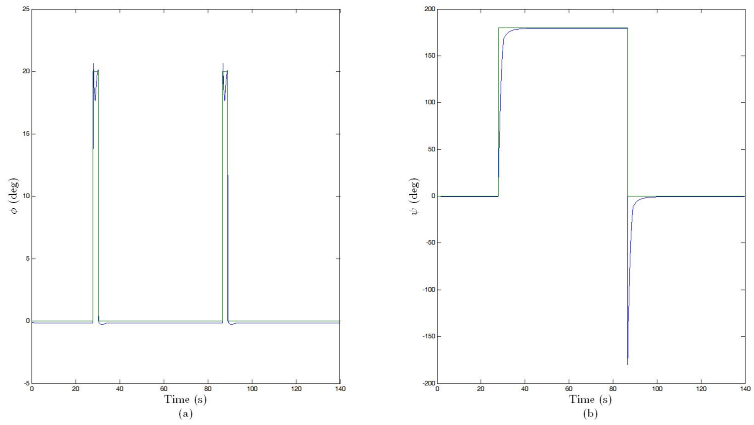


Figure 4.3: Plots for $\phi$ and $\psi$ against time for a circuit.

## 4.2 Surface Deflection Controller

The surface deflection controller takes input from the flight trajectory generator and outputs the surface deflections required for the aircraft to perform the desired maneuver. This controller generates four outputs: the surface deflections in degrees for the ailerons, elevator and rudder; and the output for thrust in revolutions per minute. The thrust output is coupled to control height and airspeed, whereas, the ailerons, elevator and rudder are decoupled and control the roll, pitch and heading respectively. The control logic for the deflection of ailerons, elevator, rudder and the output thrust are given by the equation set 1, preceded by definitions of frequently used variables.

$$\epsilon_a = (\phi_{ref} - \phi)$$

$$\epsilon_e = (z_{ref} - z)$$

$$\epsilon_r = (\psi_{ref} - \psi)$$

$$\epsilon_t = (v_{aref} - v_a)$$

$$\delta_a = K_p \epsilon_a \tag{4.1}$$

$$\delta_e = K_p \epsilon_e + K_d \frac{d}{dt} \epsilon_e + K_i \int \epsilon_e \; dt \tag{4.2}$$

$$\delta_r = K_p \epsilon_r + K_d \frac{d}{dt} \epsilon_r + K_i \int \epsilon_r \; dt \tag{4.3}$$

$$\delta_t = 4600 + K_p(\epsilon_t - k_z \epsilon_e \theta) + K_d \frac{d}{dt}(\epsilon_t - k_z \epsilon_e \theta) + K_i \int (\epsilon_t - k_z \epsilon_e \theta) \; dt \tag{4.4}$$

where $K_p$, $K_i$ and $K_d$ represent the proportional, integral and derivative gains of the controller, and $k_z$ is the reduction factor for $\epsilon_e \theta$. The controller gains used can be found in Table A.1 in Appendix A.

Equation 4.4 has an added term, $k_z \epsilon_e \theta$, which couples the thrust output with the error in altitude. This term improves the performance of the aircraft by increasing or decreasing the thrust output of the aircraft in an effort to propel the aircraft towards the desired altitude faster, consequently reducing the time to reach the altitude.

When the aircraft is below the desired altitude and the pitch up, then the thrust output increases to propel the aircraft towards the altitude, whereas if the aircraft is pitched down, then the thrust reduces in an effort to reduce the magnitude of the drop. Similarly, the thrust output increases when the aircraft is above the desired altitude and pitched down, and decreases when it is above the desired altitude and pitched up. Adding this term effectively reduces the magnitude of oscillations about the desired altitude; depending on the gain $k_z$ and pitch angle $\theta$. It should be noted, however, that this term works on the assumption that $\theta$ is relatively small.

## 4.3 Maneuvers

Khan's simulator uses Great Planes' YAK54 RC airplane as the test platform. Although available commercially, the YAK54 used has been modified for enhanced maneuverability by addition of sensors and structural reinforcements. The physical properties of the aircraft can be found in Table A.3 in Appendix A [8].

All four maneuvers demonstrated in this work have the same initial conditions: spatial positioning of the aircraft at the origin; one non zero velocity of 11.5 m/s in the x direction with respect to the aircraft coordinate axes; and all three angles, roll($\phi$), pitch($\theta$) and yaw($\psi$) of 0°.

### 4.3.1 Level Flight

For level flight, the controller aims to stabilize the aircraft at a desired altitude within a short time span. Initially, the desired altitude is 0 m, which changes to 10 m after aircraft has traveled 800 m in the x direction. The maneuver trajectory and data plots are shown in figures 4.4 and 4.5, and are based on the following set of reference values:

$$\phi_{ref} = 0^\circ$$

$$z_{ref} = \text{-10 m}$$

$$v_{aref} = 11.5 \text{ m/s}$$

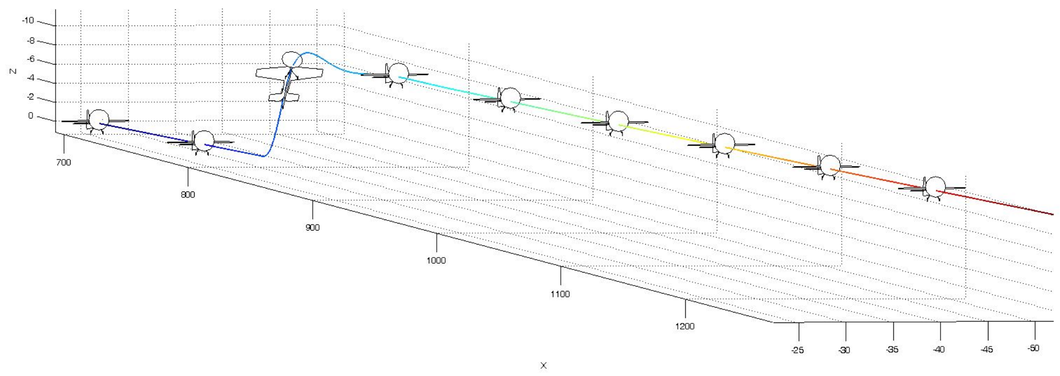


Figure 4.4: Flight trajectory for level flight maneuver.

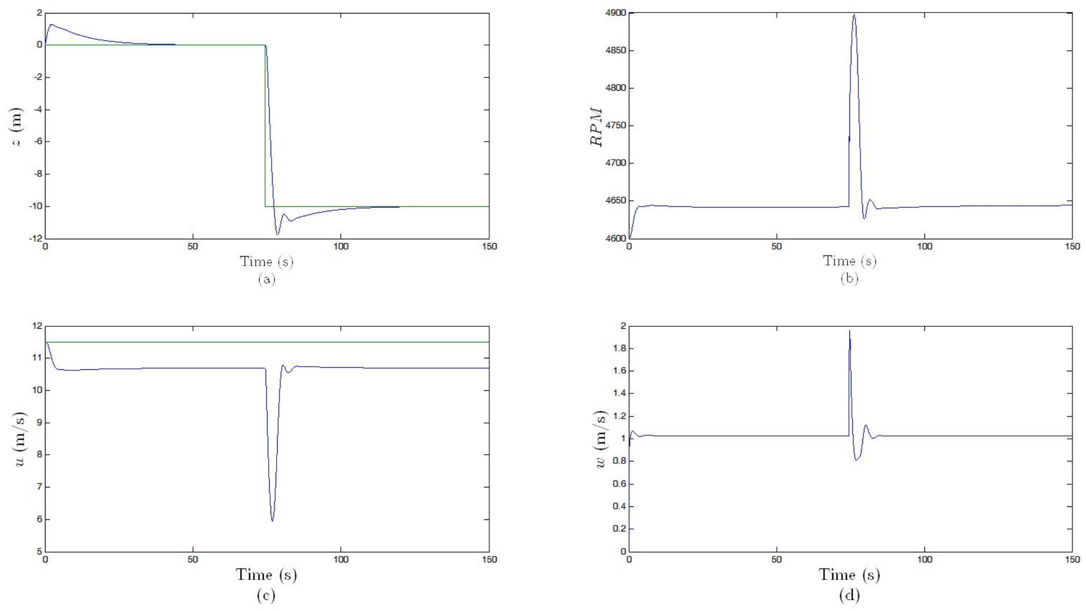


Figure 4.5: Level flight plots for $z$, RPM, $u$ and $w$ against time.

Figure 4.5 (a) shows an undesirable fall of approximately 1.2 m in altitude after the initialization of the maneuver, which results in an increase in $w$. This fall is due to a low base thrust, which quickly rises and stabilizes at approximately 4644 RPM; corresponding to the thrust output required to maintain $u$ at 10.7 m/s. The velocity in the z direction stabilizes around 1 m/s, owing to the aircraft having a constant pitch angle to maintain the altitude. The aircraft stabilizes at a speed of 10.75 m/s at both 0 m and 10 m of altitude, while the reference speed is 11.5 m/s. Hence, 10.75 m/s can be seen as the optimum speed required by the aircraft to maintain level flight. Later, the aircraft is prompted to change altitude to 10 m after 75 seconds of motion, resulting in an increase in both the pitch angle and thrust output to reach the new altitude. Consequently, $u$ decreases momentarily by a large amount and then begins to increase while $w$ increases. This change in altitude causes some oscillations as the aircraft tries to maintain a new equilibrium position at 10 m. However, the controller successfully levels out the oscillations within a time span of 17 seconds.

### 4.3.2 180 Degrees Turn

For a 180° turn, the controller aims to turn the aircraft by 180° after traveling 800 meters in the x direction at an altitude of 0 m. Ideally, the controller is in equilibrium before turning with roll and yaw angles of zero degrees and a reference velocity of 11.5 m/s. The maneuver trajectory and data plots are shown in figures 4.6, 4.7and 4.8, and the turn is defined by the following reference values:

$$\begin{aligned}\phi_{ref} &= 20^\circ \\ \psi_{ref} &= 180^\circ \\ z_{ref} &= 0 \text{ m} \\ v_{aref} &= 11.5 \text{ m/s}\end{aligned}$$

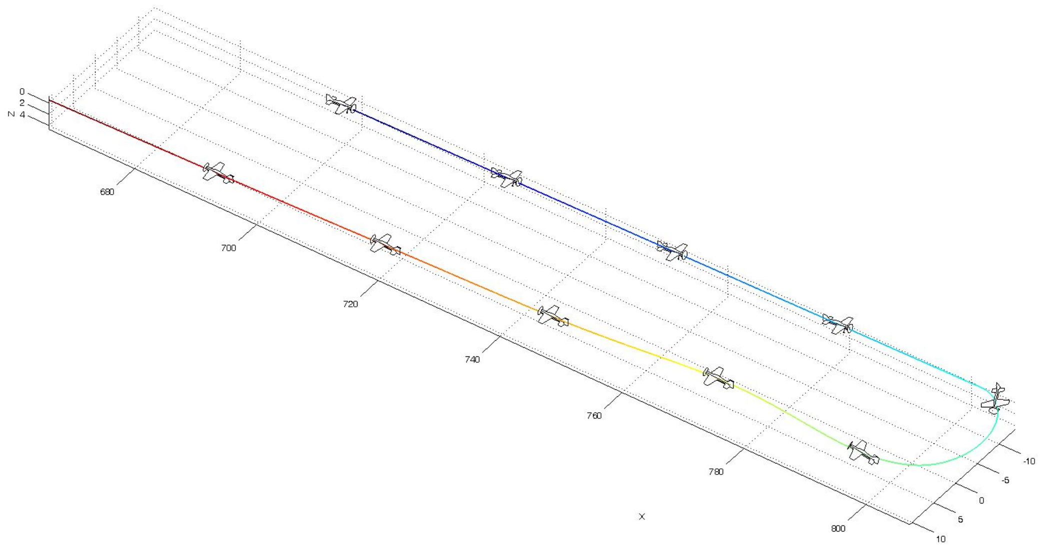


Figure 4.6: Flight trajectory for 180° turn.

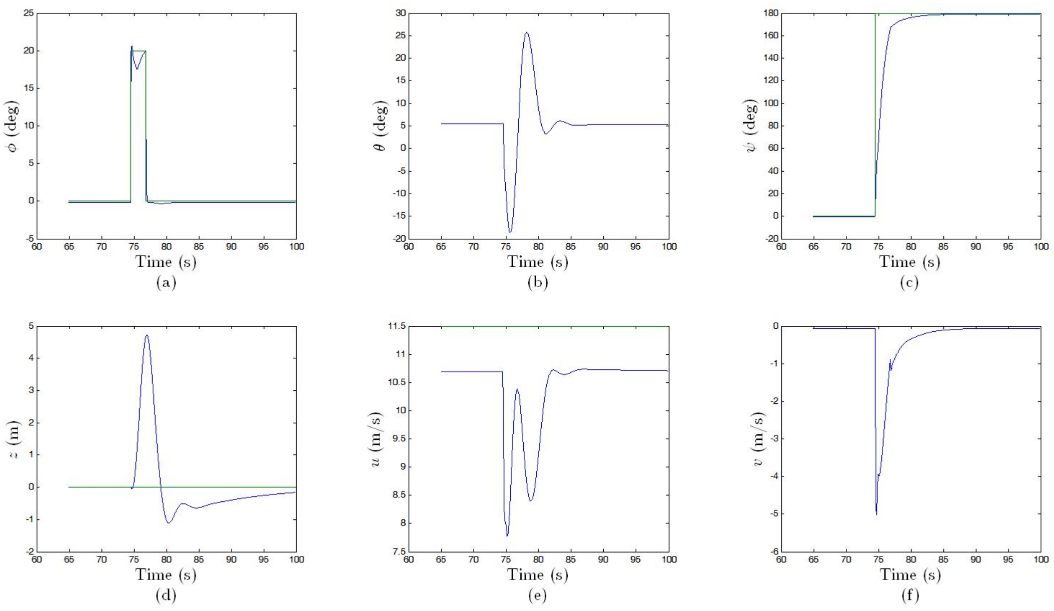


Figure 4.7: 180° turn plots for $\phi$, $\theta$, $\psi$, $z$, $u$ and $v$ against time.

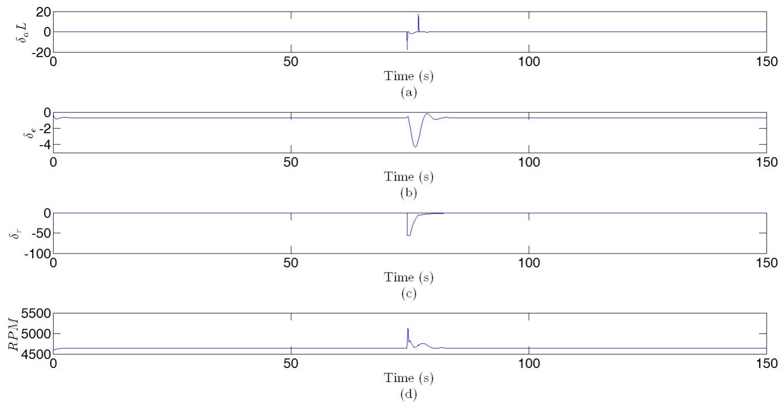


Figure 4.8: Surface deflections

The controller successfully executes a 180° turn in approximately 3 seconds. However, all three plots of $\theta$, $z$ and $u$ can be seen to oscillate for a period of approximately 20 seconds. This is due to the fact that aircraft rolls during turn, which induces a vector in the horizontal direction to maintain the roll, consequently reducing lift. Hence, the altitude drops initially and oscillates before stabilizing at the reference altitude again. The initial drop in $u$ is due to both, the drop in $z$ and an increase in $v$ during turn, whereas the second drop is owing to oscillations in $z$ as the aircraft tries to establish equilibrium at 0 m.

### 4.3.3 Consecutive Maneuvers

This maneuver aims to test the ability of the controller to perform and transition between consecutive maneuvers. The aircraft initially stabilizes at 0 m at cruise conditions, then goes on to perform a circuit at the same height followed by a change in altitude and finally completes the maneuver with a successful circuit at the new altitude. The turns are initiated at a distance of 300 meters from either side of the origin while the transition to the new altitude is triggered at the origin after the first circuit is completed. The maneuver trajectory and data plots are shown in figures 4.9 and 4.10, and the maneuver is defined by the following reference values:

$$\begin{cases} z_{ref} = 0 \text{ m} & \text{First Circuit} \\ z_{ref} = \text{-10 m} & \text{Second Circuit} \end{cases}$$

$$\begin{cases} \psi_{ref} = 180° & \text{x = 300 m} \\ \psi_{ref} = 0° & \text{x = -300 m} \end{cases}$$

$$\phi_{ref} = 20°$$

$$v_{aref} = 11.5 \text{ m/s}$$

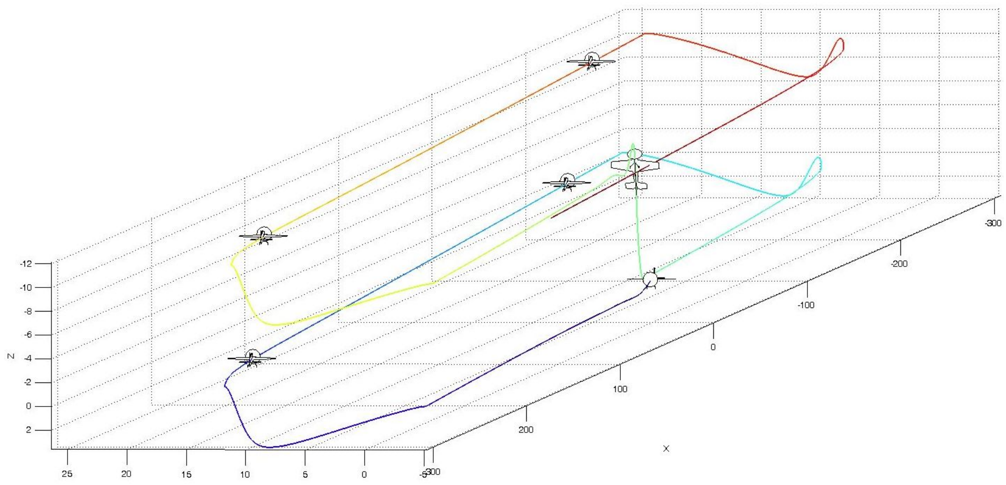


Figure 4.9: Flight trajectory for consecutive maneuvers.

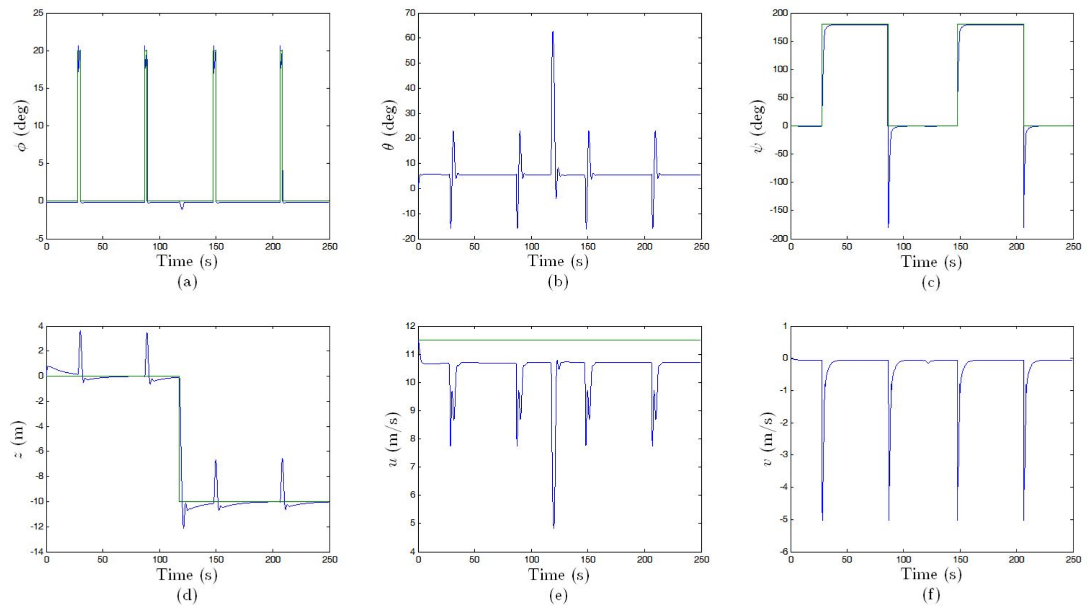


Figure 4.10: Consecutive maneuvers' plots for $\phi$, $\theta$, $\psi$, $z$, $u$ and $v$ against time.

The results for this maneuver are qualitatively similar to those of individual maneuvers, except pieced together. One noticeable difference is the repeated discontinuity found in figure 4.10 (c) at 90 and 207 seconds. This discontinuity is owing to the yaw inherently having a discontinuity where 180 degrees and -180 degrees overlap. However, the controller is programmed to be stable at these conditions, which it demonstrates by successfully completing the maneuver.

# Chapter 5

# Conclusions

The work presented describes the design of a new system capable of performing autonomous flights. During this work, AML successfully upgraded to a new platform and the system was tested through a successful flight test. Furthermore, a PID controller capable of a wide range of maneuvers was developed and tested in a simulator.

In the future, techniques could be studied to mount the electronics more accurately to minimize errors. The simulator results can be further improved as well. For example, the loss of lift caused by the rolling of the aircraft during turn can be minimized. Moreover, The controllers developed during this work could be tested experimentally and modified to perform autonomous maneuvers. Additionally, the euler angles used for the controllers could alternatively be expressed in quaternion in order to avoid discontinuities during extreme maneuvers.

# Appendices

# Appendix A

# Tables

Table A.1: Controller Gains

| Control | $k_p$ | $k_i$ | $k_d$ |
|---|---|---|---|
| Ailerons | -50 | 0 | 0 |
| Elevator | 0.8 | 0.05 | 0.6 |
| Rudder | -35 | -0.01 | -6 |
| Thrust | 50 | .03 | 30 |

Table A.2: Aircraft Properties - McFoamy

| Parameter | Symbol | Value | Unit |
|---|---|---|---|
| Mass | $m$ | .395 | kg |
| Moments of Inertia | $I_x$ | .997 | kg m$^2$ |
|  | $I_y$ | $-.00956$ | kg m$^2$ |
|  | $I_z$ | $-.003430$ | kg m$^2$ |
| Wing Area (over entire span) | $S$ | .38 | m$^2$ |
| Wing Span | $b$ | .8636 | m |
| Propeller Radius | $R$ | .127 | m |

Table A.3: Aircraft Properties - Simulator

| Parameter | Symbol | Value | Unit |
|---|---|---|---|
| Mass | $m$ | .465 | kg |
| Moments of Inertia | $I_x$ | $2.476 \times 10^{-3}$ | kg m$^2$ |
| | $I_y$ | $2.067 \times 10^{-2}$ | kg m$^2$ |
| | $I_z$ | $2.247 \times 10^{-2}$ | kg m$^2$ |
| Non-zero Products of Inertia | $I_{xz}$ | $1.7 \times 10^{-4}$ | kg m$^2$ |
| Wing Area | $S$ | .15 | m$^2$ |
| Wing Span | $b$ | .82 | m |
| Mean Aerodynamic Chord | $\bar{c}$ | .21 | m |
| Control Derivative Coefficients | $C_{l_{\delta_a}}$ | -.0006777 | deg$^{-1}$ |
| | $C_{m_{\delta_e}}$ | -.0117747 | deg$^{-1}$ |
| | $C_{n_{\delta_r}}$ | -.0035663 | deg$^{-1}$ |
| Maximum Aileron Deflection | $\delta_{a_{max}}$ | 52 | deg |
| Maximum Elevator Deflection | $\delta_{e_{max}}$ | 35 | deg |
| Maximum Rudder Deflection | $\delta_{r_{max}}$ | 56 | deg |
| Thruster Constant | $k_t$ | $2.21 \times 10^{-7}$ | $\frac{\text{N}}{\text{RPM}^2}$ |
| Propeller Radius | $R$ | .127 | m |

# Appendix B

# User Manual

## B.1 System Operation Details

### B.1.1 Bypassing the Pixhawk

The Pixhawk supports both manual and autonomous flight. However, bypassing the Pixhawk might be required for troubleshooting or diagnostics. To bypass the Pixhawk in manual mode, the servos can be connected directly to the receiver.

The receiver can be powered through two ways; Pixhawk's power rail; or the ESC. Do not connect any servos directly to the PWM outputs of the radio control receiver if the latter is powered through the RCin port/connection of the Pixhawk. This can damage the Pixhawk. Figure B.1 shows the receiver used by AML.

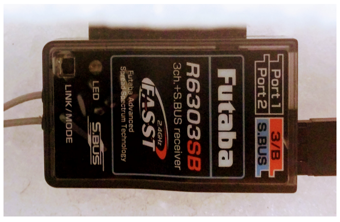


Figure B.1: Onboard Futaba reciever

### B.1.2 External IMU

If required, an external IMU can be plugged into the SPI connection port of the Pixhawk.

### B.1.3 Red Light Error

The red blinking light on Pixhawk is an indication of incorrect calibration of sensors. The sensors would need to be recalibrated.

### B.1.4 Sensor Calibration Override

To bypass the calibration of a sensor, for example the air speed sensor, click on the parameters tab and search for the particular sensor. Override calibration by setting the parameter value to 162128.

### B.1.5 Arming the Pixhawk and Motor

The system needs to be armed twice - through the safety switch and radio controller - before it can be used. To arm the Pixhawk, power the Pixhawk and plug in the safety switch. Wait for the Pixhawk to display a white light before pressing and holding the safety switch for approximately 3 seconds. If done correctly, the Pixhawk will make a positive sound and the control surfaces will deflect to their trimmed states. Within a few seconds, use the radio control remote to push the throttle to its lowest value (down) at the same time pushing the yaw stick to the right. This will cause a distinct beep from the ESC (from the motor). Then center and push the throttle stick upwards to hear a second beep by the ESC. Thereafter, push it all the way down to hear three more beeps. The motor is armed after the three beeps. Pushing the throttle up will cause the motor to spin.

To disarm the motor through the RC controller, push the throttle stick down and the yaw stick all the way to the left. If done successfully, the Pixhawk will make the disarming sound.

Care should be taken while arming the Pixhawk and motor after power up. Waiting for more than a few seconds before arming through the RC controller (after it has been armed through the safety switch) will induce a fail safe that won't allow the motor to be armed until the Pixhawk undergoes a power cycle flush (PCF). A PCF requires all systems to be power off for approximately 5 minutes before they could be powered on again. This can be avoided by arming the Pixhawk through the radio control right after it is armed via the kill switch. Thereafter, it can be disarmed immediately to continue working/inspecting the aircraft and can be armed back when desired without a PCF.

### B.1.6 Reversed Control Surface Deflections

At AML, the surface deflections after calibrating the sensors were reversed. For example, pushing the aileron stick to the right would deflect the left aileron downwards, consequently making the aircraft roll to the left. There were two ways discovered at AML to overcome this problem. It can either be done using the firmware, i.e modifying the code of the firmware and adding a negative sign to what is called a 'mixer' - presumably this way would be required for closed loops - or the controls can be inverted through the RC controller itself. The following steps can be followed to reverse the controls on the Futaba T7C radio controller:

1. Press and hold the mode/page button on the left of the display
2. Use the analog wheel on the right to scroll down to 'REVERSE'
3. Enter the option by pressing the wheel
4. Use the select/cursor option to highlight the desired channel
5. Use the wheel to scroll the channel block into 'REV' option

It should be be noted that if the controls are reversed, then the arming/disarming of the motor from the RC controller is reversed too. For example, if solely the channel corresponding to the yaw was reversed, then arming the Pixhawk would require pushing the throttle stick down and yaw left. Similarly, disarming would required pushing the throttle down and yaw to the right.

### B.1.7 Copying Trims

QGC allows for copying the trims from radio controller to the Pixhawk. This allows for having physical trims on the surface while the ones on the radio controller display are centered. This can be achieved by following the steps:

1. Connect the vehicle to the ground station using the telemetry module
2. Turn on the radio controller and arm the vehicle
3. Set the desired trims using the radio controller
4. Disarm the vehicle using the radio controller
5. Scroll to the 'Radio' tab of QGC
6. Choose the 'copy trim' option and confirm by pressing 'OK.' The Pixhawk will beep twice if the process is successful
7. While the vehicle is disarmed, set the trims to their centered position on the radio controller
8. The trims are now copied. Arming the vehicle will deflect the control surfaces to the trims set in step 3 while the radio controller would display an untrimmed/centered configuration.

### B.1.8 Telemetry Setup

The following steps can be followed to set up the telemetry used at AML:

1. Download and install the 'CP210x USB to UART Bridge VCP Drivers' from the Silicon Labs' website

2. Download and install the 'Virtual COM Port Drivers' (VCP) from FTDI Chip's website
3. Note that steps 2 or 3 might prompt an error message promting 'elevation required' while using windows 7
4. Connect the telemetry ground station module to the computer and the aircraft module to the Pixhawk
5. Identify the COM Port corresponding to the telemetry module
6. Access the COMM LINK tab and click on add to create add a new link
7. Choose the COM port identified in step 5
8. Set the baud rate to 57600
9. Click on connect. The connection should take few seconds
10. Ensure that the USB cable used should allow two way communication. Not all USB cables allow two way communication.

### B.1.9 Preflight Bench Test

The purpose of the bench test is to ensure that all sensors are functional and producing expected results. It also provides the tester a good idea of the error in the data and accuracy of the sensors. While all the sensors are tested before flight, this section provides information on testing the GPS, IMU and barometric pressure sensor.

The GPS test is performed by taking the aircraft outside in one of the fields at McGill. The system is powered up and the Pixhawk LED light observed. Once a solid green light is attained and a GPS is locked on, the aircraft should be made to do a known maneuver. For example, the aircraft can be held level and the tester can make circuits around the field. If the GPS is functional, then the same continuous maneuver could be seen after bench test data analysis. In the case of circuits, the latitude vs longitude plot should display a circuit. Additionally, the approximate latitude and longitude of the field can be found through various software online and can be compared to the values obtained through the GPS.

The barometric sensor test requires marking an object, for example a table, as a reference. The height of the object is measured and the system is made to match that height and held there for a few seconds. This process is repeated for different heights. Ideally, the change in height would be performed one after another to make the post analysis easier. The test data will be used to verify the functionality of the sensor. It should be noted that for the actual flight itself, the barometric pressure should be covered with foam or a sponge to provide more accurate values and minimize noise effects due to prop wash.

The IMU onboard can partially be tested during the sensor calibration phase through QGC. QGroundControl prompts the user to change orientations of the aircraft during the test phase and the interface shows if the aircraft is oriented in the particular way prompted. If the highlighted on screen orientation matches the actual one, then the this particular portion was the test is successful. For additional testing, the aircraft can be made to translate in any one particular coordinate axes, for example along the x axis of the aircraft, while keeping others fixed. If the Pixhawk axes match the aircraft ones exactly, then high acceleration should be seen in the x axis. Constant acceleration around 9.81 m/s should also be seen in the negative z axis assuming the tested configuration had yaw and roll angles of zero degrees.

### B.1.10 Retrieving the PX4 Log Files

PX4 Log files can be retrieved by un-mounting the onboard SD card and inserting into the computer to copy the files to a desired folder. Alternatively, the telemetry link can be utilized. Once the aircraft is connected through the telemetry link, the logs can be displayed by accessing the 'widgets' tab on QGroundControl and pressing on the 'log download' option. The required logs can then be downloaded from the list displayed by the log download window. It should be noted that downloading via the telemetry link is significantly slower than the alternative.

### B.1.11 Post Analysis Python Script

The open-source script used is known as *sdlog2_dump.py*. It converts PX4 log file data to CSV format. Thereon, you can plot graphs and perform analysis of the desired variables. The basic steps for running the scrip in Windows environment are:

1. Perform a custom installation of Python software, for example Python 3.5. When prompted, choose 'install Python as environment variable'

2. Open up the command box. This can be done by searching for 'cmd' in the 'start' menu.

3. Type 'cd' then drag and drop the folder containing the script. Press enter

4. Type 'dir' then press enter. This will display everything in that folder.

5. Copy and past the required log files in .PX4log format in the same folder as the *sdlog2_dump.py*

6. Go to command and type:
   **python sdlog2_dump.py the_log_under_study.px4log** > **this_is_the_new_name**
   then press enter

7. Special care should be taken to the spaces in between the text in step 6

8. The folder containing *sdlog2_dump.py* should now have a new CSV file called **this_is_the_new_name** corresponding to the PX4 log file named **the_log_under_study**

The flight data can be analyzed once the given steps are completed and CSV file retrieved. The CSV file will feature various columns corresponding to different data from each sensor onboard.

# Bibliography

[1] *Pixhawk Hardware*. http://dev.px4.io/hardware-pixhawk.html.

[2] *PX4 Development Guide*. http://dev.px4.io/.

[3] *QGroundControl* http://qgroundcontrol.com/

[4] *APM 2.5 and 2.6.* http://ardupilot.org/copter/docs/common-apm25-and-26-overview.html

[5] *Hobby King Telemetry*. https://hobbyking.com/en-us

[6] *FlightPlot*. https://pixhawk.org/dev/flightplot

[7] *sdlog2*. https://pixhawk.org/firmware/apps/sdlog2

[8] E. Bulka and M.Nahon, *"Autonomous Control for Agile Fixed-Wing UAVs Performing Aerobatic Maneuvers"* To be Submitted, 2017.

[9] *RCFOAM*. www.rcfoam.com

[10] *Pixhawk CAD*. www.grabcad.com