

Automated Pilot Control Assistance for a Micro-Scale Helicopter

Parker A. Evans ^{*} and Jeffrey M. Hudson [†] and Collin D. Weber [†]

Cornell University, Ithaca, NY, 14853, USA

Our research group is currently working on methods to electronically control the flight of a small hobby shop helicopter, the EfliteTM Blade CX. This project is highly constrained as it is imperative to work within the constraints of a small weight and power budget due to the scale of the platform. To the stock airframe and components we have added a custom made circuit board containing a series of MEMS sensors, allowing us to determine flight characteristics of the aircraft including, pitch, roll and yaw rates as well as altitude and heading. The TI MSP430 microcontroller is utilized for data synthesis and control signal output to the motors and servos. As the stock helicopter is rather difficult to control for a novice pilot, the group is looking to provide flight assistance aiding the human pilot in such maneuvers as takeoff, altitude maintenance, and heading control. Achieving these objectives requires robust control code, given the constraints of the microprocessor's limited computing power and the resolutions and sampling rates of the sensors. Implementation of multiple control loops has led to the development of heading and altitude control. These basic behaviors can next be utilized jointly to perform more complex maneuvers. This will allow us to investigate advanced flight techniques such as autonomously tracking to a heading, allowing for rudimentary dead reckoning navigation. Since the computing power of the microcontroller is a major limiting factor, these sophisticated maneuvers will be performed through a Linux based Gumstix miniature computer. Other future topics of investigation could include integrating sonar units for obstacle avoidance and a micro scale camera to return live video feeds to the operator.

Nomenclature

a	Net Acceleration
A	Cross-sectional Area
C_d	Coefficient of Drag
$e(t)$	Controller Error Signal Input
F	Force on System
F_D	Drag Force
g	Earth Gravitational Acceleration
K_D	Derivative Gain for PID Controller
K_I	Integral Gain for PID Controller
K_P	Proportional Gain for PID Controller
m	Total Mass of Helicopter
T	Time Between Samples
T_{net}	Net Torque From Both Motors
$u(t)$	PID Control Output
v	Velocity
\textit{Symbol}	
ρ	Density

^{*}Master of Engineering Student, Department of Computer Science, Cornell University, 4130 Upson Hall, Ithaca, NY 14853-7501, AIAA Student Member

[†]Master of Engineering Student, Sibley School of Mechanical and Aerospace Engineering, Cornell University, 105 Upson Hall, Ithaca, NY 14853, AIAA Student Member

I. Introduction

There has been a growing interest in unmanned aerial vehicles (UAVs), especially helicopter based platforms. A great deal of research has been conducted at MIT, Stanford, Carnegie Mellon, as well as numerous other institutions. Most of these vehicles are designed around internal-combustion-powered motors, use commercially available inertial measurement units (IMU), and have scale factors of up to 5 foot blade diameters.¹ In addition to this, the recent unveiling of Sikorsky's counter rotating helicopter, shows growing interest in non-tail rotor vehicles. With recurring deadly crashes of helicopters both in military and commercial use, the need to control flight or aid the pilot is growing. This paper will discuss the development of a control system for the Blade CX, a small scale counter rotating helicopter. The major design constraints of this project are to minimize power, payload as well as price of the system. Keeping these key areas in mind, the goals of the project are to not only establish stable hover, but secondly begin work towards surveillance and advanced flight techniques.

The Blade CX is a 227 gram small hobby shop helicopter. Its wingspan is roughly 35 cm in rotor diameter. There are two motors turning counter rotating blades, instead of the usual tail rotor design most commonly found in helicopter designs. The use of counter rotating blades allows for more stable flight, as the two blades create counteracting torques that, ideally eliminate yaw in the system. The helicopter is mostly made from readily available plastic components and is thus extremely lightweight, it costs less than \$200, and is thus very affordable. Although the Blade CX is inexpensive, the issues that arise with the use of this helicopter are that it has only a 7.4 V battery to power motors, servos and any additional components, and the small scale means that there is only a payload of roughly 90 grams.

The stock helicopter has remained more or less intact with slight modifications to the fuselage, but a small control board was added to incorporate the sensors. The driving factor when selecting the sensors relied on price and power consumption. The various sensors that were incorporated include: a gyroscope, an accelerometer, a magnetometer, sonar and a pressure sensor, as well as the microcontroller. Maintaining the price constraint all electrical components and the board can be purchased for about \$200. At a weight of 20 grams we continued to maintain a relatively safe payload margin. Creating a control board saves considerable money and allows us to maintain a low voltage level and weight. Commercial IMU units cost up to \$16,000, and most have added weight as the design is self contained, and enclosed in a larger package. Since adding a secondary power supply would have been costly for the weight budget the board taps into the stock 7.4 V battery of the helicopter, and careful consideration had to go into not draining the available power output. Additional components had to be added to the helicopter, including a different receiver and sonar. In total the complete system costs about \$400 and adds a 50 gram payload to the helicopter.

The MSP430 microcontroller serves as the basis for the control system, integrates sensor data and generates servo and motor PWM signals. The microcontroller was coded with specific timing frequencies in order to correctly sample each sensor. With its limited computing capabilities the coding of the system needed to limit complex mathematical operations, and thus a simple implementation was necessary. The Gumstix Linux micro computer will be used when trying to perform advanced flight routines and higher level processing that cannot feasibly be performed on the MSP430.

The initial goal for this project was to establish stable hover of the helicopter, controlling both yaw and height. Due to the nature of the counter rotating helicopter these two dynamics are coupled, and considerations for both had to be taken into account in each loop. A MATLAB simulation was created to help tune the height control, and PID control loops were placed on both height and yaw. Height control was designed using a downward facing sonar, and yaw control used both the magnetometer as well as the z-axis gyro. After completion of this, we hope to perform more advanced flight techniques including obstacle avoidance. The group hopes to establish a more intuitive manual control system which will adjust height, orientation and lateral movement. Finally we hope to include video surveillance, using a small wireless video camera. The rest of the paper will detail the developments up to stable hover of the vehicle, including board development, sensor integration and control system implementation.

I.A. Background

The major consideration in this project is that the use of a counter-rotating helicopter will add both simplifications and complexities to the system, compared to the usual tail rotor helicopter. The primary advantage, especially in the small R/C vehicle, is that there is an overall weight reduction, a less mechanically complex system and a smaller moment of inertia, which allows for faster response at the expense of decreased stability.

In the ideal helicopter system, disregarding motor discrepancies, downwash from the blades, and friction, spinning the blades at equal speeds will cause counteracting moments. These moments will be equal in magnitude but opposite in direction, as the blades will have opposite rotations, and thus cancel the inertia and result in solely an upward thrust. In order to establish a positive or negative yaw the rotation rates of the blades are altered to produce a net moment. Throttle considerations need to be taken into account here, as solely removing speed from one blade will cause the desired yaw, but in return will decrease throttle. For this reason the net desired yaw needs to be determined, and then distributed evenly on each blade, as one is increased and one is decreased. This will result in a maintained upward force on the helicopter.

Changes in direction, in both pitch and roll of the helicopter are accomplished by deflection of the thrust vector of the lower blade. This is performed by utilizing a swash plate orientated by two servos which tilt the rotation of the blade, altering the direction of the thrust and thus inducing a pitch or a roll. An upper flybar links to the top blade to increase its inertia, and thus to dampen sudden changes in its position. The increased inertia makes the blade alter its angle at a slower rate and maintain the previous angle of attack of the blade. Without this bar the dynamics of the system would be much more unstable and major excitations to the helicopter could create unwanted movement.

For the project goal of stable hover, with constant height and orientation, several simplifications can be used that deal specifically with the dynamics of the system. Since a counter rotating helicopter has an equilibrium state at hover, with both pitch and roll trimmed correctly, only yaw needs to be altered. Determining a correct mixing of throttle values, depending on the true dynamics of the system was essential to control yaw. Since there were different frictions in the motor blade assemblies and downwash from the top blade, the rotors did not spin at the same speeds with equal power input, for this reason the correct mixing had to be determined before implementing the yaw control. Also, since hover is a relatively simple concept, it can be modeled mathematically. Taking out complex dynamics left two torques that could be coupled together to produce an upward thrust. During the implementation of the controls we utilized PID control over the speeds of the motors to control both the height and yaw. Rough initial gains were determined through knowledge of the maximum angular acceleration of the motors and the moment of inertia about the rotor axis, as well as previous controls experience of the team. Fine tuning proceeded experimentally, resulting in a system that can quickly stabilize perturbations in height and yaw.

I.B. Flight Dynamics

Basic helicopter flight is determined by three characteristics: trim, stability and response. Trim refers to an equilibrium point, such as hover, and the helicopter's ability to maintain this state with static flight controls. Stability is the response of the helicopter when it is being disturbed from a trim state. Some disturbances in this system will not disrupt the stability of the aircraft, and the trim state will be maintained. Finally, the helicopter's response is how it is altered depending on pilot inputs and external disturbances. Response is clearly the most difficult flight characteristic to control, as cross-coupling of axes is inherent in the complex mechanisms of the vehicle design. With regards to hover there are much fewer responses that are introduced into the system since only one axis is being controlled.⁴

To create a system model of the hovering helicopter we can ignore blade flap. Blade flap is another important source of disturbances in the ideal system. This dynamic effect is caused by the increased lift of the blades as they move with the motion of the helicopter causing the blades to "flap." If such an event occurs the airflow can actually become reversed and lead to even greater pitching of the blades, and thus the helicopter leading to an unwanted torque. In stable hover, the angle of the blades should not be altered to the degree that blade flap will become an issue. For this reason we will be ignoring this effect. Another assumption in this system is that we will neglect the aerodynamic coupling from the two counter-rotating blades. There is a downwash from the top blade that will create resistances and complex forcing on the lower blades. For this project this effect was not studied, and was neglected for the model of the system, and the two rotors were summed together into a single force vector. Finally, the last assumption taken in this model is that ground effects were not taken into account. The downwash of both blades causes a cushion of air to be established underneath the helicopter when it is within one blade length from the ground. With these assumptions in mind the system was modeled to show its response to the step input from the microcontroller. The implementation of this system is discussed later in this paper, but the dynamic equations governing it follow.

The z axis forcing on the helicopter can be found using Newton's Second Law, Eq. (1).

$$\sum F = ma \quad (1)$$

Here, F is the force acting on the helicopter, m is the total mass of the helicopter and a , is the acceleration. Taking into account the various forcing on the helicopter, including the net torque from both motors, T_{net} , gravitation forces, g , and drag of the system, the resulting Eq. (2) is found.

$$T_{net} - mg + F_D = ma \quad (2)$$

The drag of the system is found in the term F_D , which is determined in Eq. (3).

$$F_D = \frac{1}{2} \rho v^2 A C_D \quad (3)$$

Here, ρ is the density of air, v is the velocity of the helicopter, A is approximated as the cross sectional area of the helicopter, and C_D is the drag coefficient, which was also approximated as a relatively high value, since testing was not performed on this characteristic. Due to these approximations there are inaccuracies in the system model that were taken into account during implementation.

For the control of the helicopter a basic proportional-integral-derivative (PID) controller was used. The basic formula for a PID controller is found in Eq. (4).

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de}{dt} \quad (4)$$

In this equation, $u(t)$ is the controller output value. The first term is the proportional term, which is simply the error, $e(t)$, multiplied by the proportional gain. The next term is the integral term with the integral gain, multiplied by the integral of the error signal over the time period. The final term is the derivative term, with, the derivative gain multiplying the derivative of the error signal. For a digital system a discrete time PID algorithm can be used; understanding basic calculus principles the integral of the error can be approximated as the sum of the errors times the time period, as seen in Eq. (5).

$$\int_0^t e(\tau) d\tau \approx T \sum_0^t e(t) \quad (5)$$

In our system the time, t , is factored into the integral gain term to form one constant value. In a similar manner using the Fundamental Theorem of Calculus principles the derivative term can be approximated as shown in Eq. (6).

$$\frac{de(t)}{dt} \approx \frac{e(t) - e(t-1)}{T} \quad (6)$$

In this equation the change in subsequent error signals is found and divided by the step time, T . Introducing Eq. (5) and Eq. (6) into the PID controller in Eq. (4), results with the PID controller equation for a digital system, found in Eq. (7).³

$$u(t) = K_P e(t) + K_I T \sum_0^t e(t) + K_D \frac{e(t) - e(t-1)}{T} \quad (7)$$

By implementing this controller into the simulation estimates for the true gains can be obtained, and then manual tuning of the constants in the actual code on the helicopter will allow for the system to be controlled.

II. Platform

The platform on which this project is based consists of a slightly modified E-FliteTMRTF Blade CX helicopter with added electronics, sensors and embedded software. A sensor and control printed circuit board (PCB) with a TI MSP430F2618 microcontroller contains sensors and circuitry for driving motors and servos and embedded software on the microcontroller that samples the sensors and controls the motors and servos on the helicopter. The Castle Creations Berg 4L receiver interfaces with this board and is used to receive servo commands from a transmitter for manual control. In addition, a Gumstix Linux motherboard can be stacked on to the control and sensor PCB for higher level control and data capture.

II.A. Helicopter

The E-FliteTMBlade CX is a micro scale helicopter with counter rotating blades. The stock helicopter with the battery weighs only 227 grams making it a very light weight helicopter. It has an empirically determined payload of only 90 grams. The limited payload capacity of this helicopter is one of the biggest restrictions that we have faced in this project. The control board had to be carefully designed and the sensor array meticulously chosen to avoid going over this weight restriction. Some specifications of the particular helicopter are give in table 1.

Length	400mm
Height	182mm
Main Rotor Diameter	345mm
Weight RTF w/ Battery	227g
Battery	7.4V 800mAh Li-Po
Motor	180 (2 Installed)

Table 1. Blade CX Physical Specifications²

II.B. Control Board and Microcontroller

Our control board was developed at Cornell by our project advisor Robert MacCurdy specifically for this team. It has all of the necessary circuitry to do sensor fusion, radio receiver decoding, data processing and motor and servo control in addition to having provisional circuitry for debugging and connecting to an external Gumstix Linux motherboard. The total weight of the board is 18 grams including on board sensors, keeping our payload weight well within our limit.

The heart of the board is the Texas Instruments MSP430F2618 microcontroller. This is a low power 16 bit microcontroller with a hardware integer multiplier, 116 Kilobytes(KB) of flash memory, 8KB of Random Access Memory (RAM), and 48 general purpose input/output pins (GPIO). The microcontroller can be run at 16 Megahertz with single cycle instructions for a peak performance of 16MIPS (Millions of Instructions Per Second). Other features of this microcontroller that were utilized in this project were that 12bit Successive Approximation Register (SAR) analog to digital converter (ADC) for sampling our analog sensors, a timer with seven capture compare registers for decoding radio receiver input received as a pulse width modulation (PWM) signal and two Universal Serial Communication Interfaces (USCI) for debugging communication and communication with the external Gumstix Linux motherboard. The microcontroller does not have hardware for floating point calculations which means that any floating point calculations would either take hundreds of cycles to execute or would need to be approximated with integer fixed point calculations. This project uses the latter.

The rest of the control board contains circuitry for power management and voltage control for the various components, the surface mount sensors such as the gyroscopes and the accelerometer (and pads for the pressure sensor), through holes for mounting the MicroMag3 magnetometer, a serial communication chip and circuitry for debugging communication to a PC, a low side MOSFET driver chip for modulating the current to the servos and motors from the microcontroller and headers for mounting sonar and the radio receiver. See Fig. (1) for a visualization of the board.

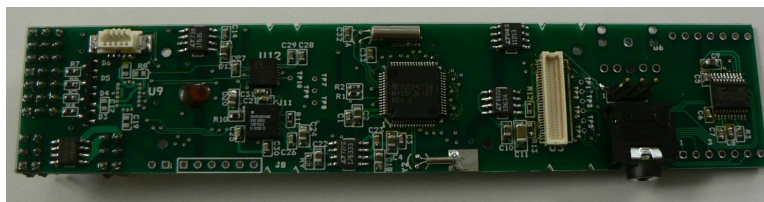


Figure 1. Image of sensor and control board

II.C. Sensors

Our control board supports several different sensors which are useful in sensing state during autonomous flight. These include gyroscopes for determining angular velocity, accelerometers for determining three dimensional acceleration and orientation with respect to gravity, ultrasonic sonar ranging sensors (which are useful in determining short distance altitude as well as detecting obstacles), a magnetometer for steady state heading information, and a pressure sensor for accurate course grained altitude determination. While we have completed the code to interface with all of these sensors except for the pressure sensor, we have only employed one sonar, one gyroscope, and the magnetometer in our control systems. The sensors have various bandwidths, some selectable in circuitry and others fixed. However, because of the limitations of our microcontroller, we are currently sampling all of our sensors at 25Hz. This decision may imply that we need some low pass digital sampling eventually, but we have neglected this so far with few visible negative effects.

II.C.1. Magnetometer

The PNI MicroMag3 is a three axis magnetic field sensor. It comes as a breakout board which attaches to our main board with standoff headers. It contains all the necessary circuitry for magnetic field calculation in three axes using PNI's Magneto-Inductive sensors and an ASIC for measurement and Serial Peripheral Interface (SPI) communication with our microcontroller. It is a popular choice because of its high resolution, minimal noise, and low cost. The choice of this sensor was made early in the design of the control board when few alternatives were available. There are several single IC package 3 axis magnetic sensors and digital compasses (some with tilt compensation) that could provide smaller footprints and thus cut down on the size and weight of future revisions of the board.

II.C.2. Gyroscopes

Two different gyroscopic sensors are used on our control board to obtain angular rate sensing in all three dimensions. This is because there are no affordable small single chip 3 axis solutions. The InvenSense IDG300 is a dual axis in plane MEMS (Micro Electro Mechanical System) gyroscope. This sensor is used to measure the angular rate around the X and Y axes (on our helicopter, these are the roll and pitch rates respectively). The sensor has a range of ± 500 degrees/second (more than adequate enough for our helicopter due to its relative stability in roll and pitch) and outputs a rate proportional voltage for each of the axes. The Analog Devices ADIS16100 is used to measure the angular velocity about the Z axis (our helicopter's yaw rate). This is employed in our Yaw control system as the derivative term of the PID controller to dampen control resonance. It has a range of ± 300 degrees/second and has an SPI interface. Since the yaw rate gyro senses perpendicular to its plane, both of these sensors can sit flat on the board and we still obtain all three axes of measurement.

II.C.3. Accelerometer

The accelerometer that we are using is the Kionix KXR94-2353 3 axis accelerometer. Accelerometers are very popular in many applications from car collision detection and airbag deployment to smart phones for orientation sensing. Because of this, 3 axis chips are cheap, small, and of high quality. The accelerometer has an SPI interface which makes it fairly easy to interface with. In the future we would like to use this sensor for orientation sensing or perhaps motion stabilization.

II.C.4. Sonar

The control board has the ability to interface with up to four LV-MaxSonar-EZ0 sonar modules made by MaxBotix. Currently we are only using one downfacing sonar for altitude sensing. This ultrasonic sonar sensor can detect objects from 0 inches to 254 inches away. However, there is a dead zone between 0 and 6 inches where the sonar reads only 6 inches. This is addressed in our controls appropriately. Also, the sonar requires an initial calibration at power-on. This calibration step requires that it be at least 12 to 14 inches away from the nearest object in its beam. We have had to plan accordingly when powering on the system. The range is fairly accurate and we have been successful using it as the sole sensor for altitude in stable hover.

II.C.5. Pressure Sensor

The pressure sensor that our control board was made to interface with is a VTI Technologies SCP1000 Series absolute pressure sensor. The idea is to use this sensor as a coarse measurement for altitude to back up the altitude sonar measurement. According to their data sheet, we could expect to obtain 10 centimeter resolution were we to incorporate this sensor into our control systems. Since our project so far has been directed at only low altitude applications inside the range of our sonar, we have not incorporated this sensor.

II.D. Software

The software running on the microcontroller must sample all of the necessary sensors, update two PID control systems using the sample data, and then use the control values output from the loop to update the two motor output PWM signals (mixed appropriately accounting for yaw control) as well as update the servo positions, all while decoding input from the radio receiver for implementing manual control signals.

The radio receiver pulse width modulation (PWM) input signal is decoded using an MSP430 timer module and four capture compare registers which are able to automatically store a free running timer count and raise an interrupt. We set this up to happen on both positive and negative edges of the PWM signals and then simply subtract the time when the signal goes low from the time the signal goes high to give us the duty cycle (the period of the PWM signal remains fixed). Since these are standard servo signals, the on time ranges from about 1 millisecond to about 2 milliseconds and thus quick transitions do not occur. The same timer is used to generate the PWM signal outputs for the motor. The problem we have faced with this is that to avoid noticeable oscillation in the stock motor setup, we need a period of about 250 microseconds. This means that there are not enough counts on the timer to time a full on time of the radio receiver input signal. We are currently fabricating a new control board which rectifies this by putting the motors on the other timer PWM output of the microcontroller.

The other hardware timer is currently used to generate the slower PWM signals for setting the positions of the two servos (for pitch and roll). It has a period of 20ms so that the PWM signals that are generated have this period (which is the appropriate period for controlling the servos). It is also used to schedule a task every 40 milliseconds (25Hz) that first samples the X and Y axes of the magnetometer and the Z axis gyroscope, determines heading (algorithm described in the control section), updates the control systems and the servo and motor PWM signals and starts an analog to digital conversion for the altitude sonar reading to be used in the next sample and update period.

The SPI communication with the sensors is performed using GPIO toggling rather than using the built in microcontroller module for SPI communication. One reason for this is because of the general inconsistency of this communication medium. Widths of communication words often differ between sensors so the control board was designed with this in mind. Manual SPI gives much more control over the process but also means slower communication (with the need for manual delays in code) and prevents the microcontroller from doing other tasks while this communication is going on.

The coded control systems are essentially just C structures with three stored constants, a minimum and maximum for the integral windup, and derivative and integral states, as well as an associated function which updates the controller one step and returns a control value. The process is described in some more detail in the controls section below. The code is actually modified from an article on microcontroller PID implementation.⁵ All numbers used in the control system code are in 24.8 fixed point which uses the long integer type (32 bits) and are converted to and from this format when necessary. In this form, 8 bits are allocated to the fraction and 24 to the decimal part of the number. This gives 1/256 resolution which is sufficient for our calculations and provides a drastic speedup in computation over full floating point.

The motor update code simply takes the current radio input PWM values for the pitch and roll, manipulates them slightly and sets the servo PWM duty cycle. It then takes the outputs from the control system update for the throttle and yaw control and mixes these two values to get two PWM output values for the two motors. This mixing essentially changes the yaw value into a range of about -100 to 100, divides this by two, and adds this to one of the motors and subtracts from the other. In terms of compare match values, full throttle for a motor is a count of 499, so we have additional code to make sure this is not exceeded, redistributing the thrust between the two motors to obtain the correct yaw (at the cost of a small amount of thrust).

The software also contains code for using the serial interface between the microcontroller and the PC for debugging, as well as some code to communicate with the Gumstix Linux motherboard when we reach that

stage in the project.

III. Control System

The control of our helicopter is split up into two parts: throttle (or altitude) control, and yaw (or heading) control. We approached the two control systems separately, spending the majority of our time on the more difficult throttle control. The systems are slightly coupled because of the fact that we are using a counter rotating blade helicopter and our mixing algorithm is not perfect, especially at high throttle, but this approach simplified the control design and still led to a combined control system that worked well for stable hover. The control for both throttle and yaw was implemented as a full PID (Proportional Integral Derivative) controller.

III.A. Throttle Control

Throttle is the most difficult (and dangerous) control to implement and test. We thus took a cautious approach to its design and implementation. As a proof of concept, we made a rough simulation to model our system and some of its non-linearities. We then proceeded to implement the control system on the helicopter (making sure to add a kill switch mechanism in the code which could be enabled and disabled from the hand held radio controller).

III.A.1. Simulation

In order to provide ourselves with a general idea of the behavior of the helicopter under the control of a PID altitude controller, our team decided to create a preliminary MATLAB simulation of this system. The simulation consisted of an iterative finite-difference code in which the position, velocity, and acceleration of the system at a given time are used to simultaneously generate an error signal and calculate new values for the dynamic properties of the helicopter. The script operates based on a finite time step, in this case 40 milliseconds, which is the time frame upon which our actual control loop operates as well. This means that the helicopter will continue accelerating at the rate determined at the beginning of a step until the signal is changed again at the end of that step, which agrees with the way in which the physical system operates. The fact that the helicopter behaves in this discretized manner means that a finite difference code will actually simulate the system performance much better than the standard continuous equations used in most control theory. In order to fill out the model further, a quadratic air drag term was added to simulate the resistance of the air against the body of the helicopter.

Of course, this model has an advantage in that the altitudes, velocities, and accelerations are known exactly since they were just numbers in the simulation. In the actual system, altitude and velocity were determined using the ultrasonic range sensor, which provided variable output with some intrinsic error. In order to add to the realism of the simulation and account for some of this error, a noise function in the form of a small randomly generated number was added to the simulated altitude value in order to mimic sonar error. Additionally, the model was executed utilizing a brief time delay (one cycle) between the implementation of the control signal and the reaction of the plant, simulating the time needed for the motors to adjust their speeds to the desired values. Using this model and varying the proportional, integral, and derivative constants, we were able to produce flight patterns such as that shown below in Fig. 2 below.

Figure 2 shows the behavior of the system under PID control as predicted by the model, with a rapid takeoff being followed by several large fluctuations in height before the system settles down after about four seconds. Note that the velocity never remains exactly zero, as the error in the sonar signal continues to require the system to respond. This is also true in the real system, in addition to the presence of other perturbations such as wind gusts. Our model, as described above, provided us with a good basis upon which to determine gain constants which could then be subject to experimental tuning.

III.A.2. Implementation

With the simulation providing a bit of background information, as well as a solid place to start for the tuning of our controller constants, we next implemented full PID control on the altitude using the sonar range values as the input for the error signal. The design of the control loop was quite straightforward and standard. The returned cycle count of the sonar device was converted to a desired height of the helicopter, and the difference

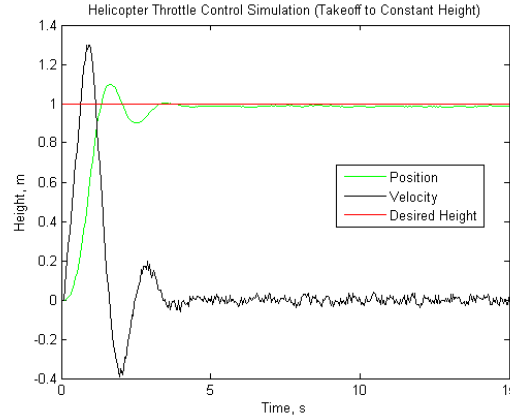


Figure 2. Plot of simulated altitude and vertical velocity behavior

between the current sonar reading and that desired value was used as the basis for the proportional part of the control. These error readings are added to a running sum (limited to within a prespecified range to prevent wind-up of the integral term) and multiplied by the integral constant to yield the integral control. Derivative control proved to be a bit less standard, in that we could not actually differentiate the sonar signal directly since it was highly discretized in 40ms intervals, and we had no other means for measuring the upward velocity of the helicopter. Instead, we used the difference between consecutive sonar readings as the basis for this, which, because the sonar returns values on a fixed time scale, is effectively the same as having a derivative value of change over time. Tuning of the control constants K_P , K_I , and K_D required experimental testing and manual adjustment, as the nature of this complex system made traditional analytical methods impractical.

Before we could begin testing, however, one more issue had to be addressed. Takeoff is a unique situation in the flight of the helicopter, and the model showed that erratic, undesired behavior could occur here due to the implementation of the control. If the system were placed under full altitude control immediately, the instantaneous error signal would be very large, meaning that the motors would immediately spool up to maximum thrust, and the helicopter would take up very rapidly. This could cause physical damage to the motors and connections, and, more troubling, would cause the system to attain a high rate of speed and travel past the desired altitude very quickly. The model showed that such behavior could result in very long settling times. To remedy this, we implemented a two-part ramp control into the code for takeoff. In the first part of this code, the motors are instructed to slowly spool up to the speed that we have determined is necessary for initial liftoff. Then, full control is implemented, but the desired altitude is gradually ramped up from a very low value to the actual hover height that we wish for the system to attain. The result is that the error signal remains very small, with the control slowly “chasing” the specified height as it increases. The implementation of this ramp routine serves to greatly reduce the settling time at the desired altitude and increase the stability of the flight behavior.

III.B. Yaw Control

The second part of our control system, that of yaw control, was approached in a different manner, since we had additional tools at our disposal in the form of more sensors that were able to read useful data. In this case, the control signal is formed based on data from two sensors, with the magnetometer providing the absolute error signal for the proportional and integral parts of the control and the z-axis gyro providing rate information for the derivative portion. Ensuring that these sensors return high-quality values is essential to the performance of the system, and thus proper initialization is necessary. When the board is powered on, the magnetometer works through twenty time steps, taking values for the X and Y components of the field during each increment, and saves the last set of X-Y field strength data as its initial heading value. This ensures that all memory has been cleared and that no artifacts from past use or startup remain. The gyro, similarly, takes initial values for the first twenty sampling periods and keeps the last of these as its calibration value, corresponding to zero rotational velocity.

Getting the data provided from the magnetometer to correspond to an actual compass heading from 0-

355 degrees proved to be difficult due to the limited mathematical capabilities of the microcontroller and its lack of ability to perform hardware floating-point operations. This made the use of a continuous arctangent function impractical. Instead, a vector of values corresponding to the ratios of Y/X in five-degree increments from 0-85 degrees scaled up by 256 was stored in the code (90 degrees is determined to be when X is zero). To find the closest heading angle corresponding to our readings, we first multiply our Y magnetometer reading by 256 (shift left by 8) and then do an integer divide by X (assuming X is not zero). We then use binary search to find the closest value in the vector and then use the index of the value as the index into a lookup table of heading angles between 0 and 85 degrees. Several conditional statements are used to determine, based on the signs of Y and X , which quadrant the heading is in, and this information, combined with the table data of ratios, is used to assign a compass heading between 0 and 355 degrees. The difference between the current heading and the desired heading is used as the basis for the proportional part of the control. Additionally, the system checks to see which direction of rotation is the fastest route back to the desired heading, and switches the sign appropriately to ensure that the system takes the shortest path. The integral portion of the controller continually adds these differences, but is capped at a maximum value to prevent it from dominating the signal. Finally, the differential data comes from the Z -axis gyro, in the form of the difference between the current reading and the calibration zero rotation value. In practice, the result is a smooth, controlled system that is able to find and maintain the desired heading fairly quickly and with low settling times and oscillation.

IV. Conclusion and Future Work

Over the course of this research project, our team has sought to implement autonomous pilot-assistance measures on a stock E-FliteTM Blade CX biaxial model helicopter. A custom microcontroller board was designed and fabricated, featuring a TI MSP430 microcontroller, gyroscope, magnetometer, accelerometer, ultrasonic range sensors, debugging interface, and link to a Gumstix Linux computer. Operator input is obtained via a Castle Creations Berg 4L receiver and mixed with the sensor data by the microcontroller, which then outputs signals to the stock motors and servos for flight control. Early stages of the research work included an extensive amount of effort in integrating the individual sensors, writing code to obtain the relevant data from each, developing calibration routines, and testing the sensors under a wide range of operating conditions to learn their capabilities and behavior. Large-scale modifications also needed to be made to the physical system and airframe to accommodate the added electronics and to ensure proper placement of the sensors. With the system developed and the components integrated, we then moved to implement basic control on the helicopter platform, focusing on stable hover at a desired altitude and heading. PID control loops, based on input from the magnetometer, sonar unit, and gyroscope, were developed to this end. Control gain values were developed based on the findings of a MATLAB model of the altitude control coupled with experimental results. Repeated testing and iteration of the gain values allowed us to produce the desired result of controlled takeoff and stable hover at the desired altitude and heading while retaining the capacity for operator control of the helicopter.

The team has several paths that we wish to pursue in our continuation of this work into the future. In the short term, we desire to bring two new dimensions to the study of the flight in the form of onboard live video and in-flight data logging using the Gumstix computer. We currently have a small camera that can return a wireless video feed at ranges up to several hundred feet, and this can easily be adapted to provide onboard video. Modifications will need to be made to both the camera, so as to allow it to run off the helicopter's power supply, and to the vehicle, to allow for mounting of the camera. We feel that in any application of this research, giving the pilot an in-air perspective adds tremendous value to the system, and thus this proof of concept is a very high priority for us. Secondly, integration of the Gumstix module, with its much larger memory capacity, will allow us to record in real time the entirety of the flight data that passes through the microcontroller in the form of both sensor input and output to the motors and servos. Coupled with observation of the system during test flights, this will provide a powerful tool for tuning control systems, studying the response of the helicopter to disturbances, and viewing the interplay between the pilot input and autonomous control. In the longer term, it is our hope that this information can be used in the development of advance flight maneuvers such as obstacle avoidance and heading-based dead reckoning, with a trend towards increased levels of autonomy and a lessened need for pilot input.

Acknowledgments

We would like to thank our project advisor Professor Ephraim Garcia and our project leader Robert MacCurdy for their assistance and guidance on the project. We would also like to thank Professor Mark Campbell of Cornell University for his helpful suggestions on our control system design and Austin Fang for his insights on the dynamics and control issues of our helicopter. Also, we would like to thank Professor C. Thomas Avedisian for his participation in the initial proposal for the project as well as his active involvement in last year's team. Finally, we would like to thank United Technologies Corporation for their generous funding of this project.

References

- ¹Gavrilets, V., Shterenberg, A., Dahleh, M. A., and Feron, E., "Avionics System for a Small Unmanned Helicopter Performing Aggressive Maneuvers," MIT, Cambridge, MA, 2000.
- ²Hudson, J., Kidd, B., MacCurdy, R., and Miller, S., "Stabilization and Control of a Micro-Scale Helicopter," *AIAA Region I - NE Student Conference*, 2008.
- ³Ogata, Katsuhiko, *Discrete-Time Control Systems*, 2nd ed, Prentice-Hall International, New Jersey, 1995.
- ⁴Padfield, Gareth D., *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling*, American Institute of Aeronautics and Astronautics, Inc., Washington DC, 1996.
- ⁵Wescott, T., "PID Without a PHD", Embedded Systems Programming, <<http://www.embedded.com/2000/0010/0010feat3.htm>>.