Control System Infrastructure for the Cal Poly Human Powered Helicopter: Upturn II

Douglas A. Thornber[1] and Samuel J. Wahyou[2]
California Polytechnic University San Luis Obispo, San Luis Obispo, CA, 93407

The Upturn was donated to Cal Poly in October 2012 by Neal Saiki from NTS. Our project was to replicate the old system to decrease the turn-around time in case of a crash during a test flight. The control system for the Upturn II was designed with the same considerations that were used for the Upturn. The result we got was inconclusive for the sensor selection, the new control system is heavier than the old system, and we couldn't validate the system without doing a flight test. For the next iteration if some of the replication requirements are relaxed the system could reduce weight due to an optimal size for the components. The objective of this report is to detail the process and implementation used to arrive at the system as it stands at the end of the 2012-2013 school year, so it may be easily be used by those wishing to continue work on this system.

Nomenclature

A	=	Area
L	=	lift
$\dot{q}$	=	dynamic pressure
RB	=	Red/ Blue Rotor Pair
V	=	voltage (volts)
v	=	velocity (ft/s)
YG	=	Yellow/ Green Rotor Pair
$\ddot{\theta}$	=	angular acceleration

[1] Undergraduate Student, AIAA Member, 1 Grand Ave, San Luis Obispo, CA 93407.
[2] Undergraduate Student, AIAA Member, 1 Grand Ave, San Luis Obispo, CA 93407.

I. Introduction

In 1980 the American Helicopter Society (AHS) established the Igor I. Sikorsky prize for the successful flight of a Human Powered Helicopter (HPH), retaining a current prize purse of $250,000. The a successful flight is defined by the fulfillment of three primary conditions in a single flight:

1. The helicopter must remain aloft for 60 seconds,
2. Reach an altitude of 9.8 ft (3 m), and
3. Remain stationed within a 32.8 m (10 m) squared area.

The first HPH to achieve flight was the DaVinci III from students at Cal Poly: San Luis Obispo in 1989 reaching an altitude of 8" off the ground for a duration of 7.1 s. Following this, a flight the Yuri I from students from Nihon University in Japan flew for 19.46 s at an altitude of 8". Recently the Gamera II from the University of Maryland, and Atlas of AeroVelo have made strides nearing a successful flight.

II. Objectives and Requirements

The scope of this project will cover the implementation of a control system for the Upturn II HPH. The objectives will cover the required performance of the control system, and it's installation as part of the helicopter. The system shall:

– Maintain stability for the two rotor pairs, limiting translational motion vehicle
– Maintain operating capability for more than 60s
– Operate between 3.2 ft and 13.1 ft of altitude
– Integrate with the installation infrastructure of the Upturn I Helicopter
– Evaluate a basic control law for the control of the helicopter

Complete fulfillment of the objectives of this project requires a test flight of the Upturn II helicopter, receiving and recording flight data from the system for a complete stability analysis.

III. Background

In October 2012 Neal Saiki of NTS donated their human powered helicopter the Upturn I, which achieved a two foot altitude lasting for ten seconds in June 2012, to Cal Poly. Our task was to learn how to replicate the helicopter to improve our turnaround time if the helicopter were to crash during a test flight. For the control system Neal told us the Upturn was having a problem with shutting off during the test flight when the Upturn wasn't harnessed in. His only thought on this was that the harness grounded the helicopter and without it static electricity builds up on the wires and turns off the system. Without the control system the helicopter is naturally unstable and any disturbance will cause the helicopter to crash.

Each control system was composed of two sensors, a microprocessor, two servos, and two batteries. The microprocessor was a Maestro written in 4th, a programming language, the sensor was an Infrared (IR) sensor made by Sharp, the servos were made by HiTec, and the batteries were 7.4 V to accommodate the system. We were not able to acquire the code governing the microprocessor and it was written in 4th, we decided not to reuse it. For the sensors Neal strongly recommended the IR sensor over ultrasonic and Inertial Mass Unit (IMU), but we decided to test this for ourselves (except the IMU because they are expensive ~ $3000). Due to the wires being an issue during test flights we decided to go wireless to eliminate this problem, and it would lower mass (the helicopter uses hundreds of feet of wiring)

IV. Part Selection

The main component that was needed for the control system was a microprocessor. The microprocessor is an important component because it is basically our computer that reads in the inputs from the sensors and computes the response to keep the helicopter level. Our microprocessor choice was the Arduino Uno - R3 because it has 16 MHz clock speed, uses a 9V battery, connects with a mini USB, and uses the open source Arduino software library. The 9V battery was nice to have for testing because we didn’t need to recharge them, but they did add weight to the system. The software library was also very useful when we came across a problem because the Arduino community is large enough that someone else had that problem and fixed it.

The next component that we selected was the wireless antenna. The wireless antenna was needed so that one unit could talk to the other. We wanted to go wireless to reduce mass and potentially solve the problem of not working after lift-off. Our wireless antenna pick was the XBee 2mW Wire Antenna – Series 2 because it can easily integrate with the Arduino Uno, has a data rate of 250kbps, and a range of 400ft. The XBee also had a board to easily connect to an Arduino, the XBee Explorer Regulated. Initially we tested this system to make sure that it could transmit and receive through the foam rotors and found that they could talk to each other through the walls of the HPH lab.

One of the more important components was the sensors. The sensors detect the distance away from the ground that each unit is at. With each unit distance we can determine the difference in differences and control the system to try to keep the difference as close to zero as possible. Since the Sikorsky Prize requires that the helicopter reach a ten foot altitude, our sensors need to be able to sense a maximum distance greater than ten feet. With this in mind we wanted to find an ultrasonic sensor and an IR sensor to compare. The ultrasonic sensor we used was the Maxbotix HRLV-EZ4 because it has a range of 1ft to 16 ft and boasts a 0.04 inch accuracy. The IR sensor that we used was the Sharp GP2Y0A710YK0F because it has a range of 3 ft to 18 ft (this is the only IR sensor that we found with this long of a range, most of them had a much smaller maximum range).

The last component that we selected was the servo. The servo is needed to quickly adjust the control surfaces with enough torque to do so. The servo we used was the HiTec HS-5655MH because it can change 60° in 0.09 s, with 194 oz/in of torque, and it has metal gearing. The metal gearing is what drove us to this choice because we wanted a servo that would last and this servo was the cheapest servo with metal gearing. Even though it was the cheapest servo with metal gearing it still had more than enough torque and had a quick angular velocity.

V. Board Design and Construction

One of the requirements the drove much of the design of the control board design was the desire to fit the new control system into the Upturn I helicopter, should the Upturn II be rendered inoperable. The control systems of the Upturn I had many of the components permanently installed with epoxy, meaning in order to examine the system to the some components had to be ripped from their placements. This also prevented components from the Upturn I being reused for the Upturn II system. In order to facilitate flexibility for the system to replace and swap parts without destroying the entire board, all parts mere required to be installed with bolts. Nylon bolts and nuts needed to be used over metal bolts so the bolts do not accidently conduct electricity, and cause the system to break. Since components are bolted on, they can be easily removed and replaced if damaged during flight, or if new better fidelity parts are required. The Upturn I system used an installation board 4" x 4" which screwed into 2 inner wall sections of each rotor with screws. In order to fit the selected components within the required space, the components needed to be stacked vertically. Additionally, on the YG boards require the servo to be placed in the center of the lower board in order to interface with the aileron installations on those rotors. In order to provide space for wire connections the second board level is displaced above the first using 1¼" spacers. This allows the upper side of the upper board to become dedicated space for the large Arduino. The second board is shorter by ½" on each side in order to accommodate from the installation walls. No components can be placed in this portion on the bottom boards. Figure 1 shows the

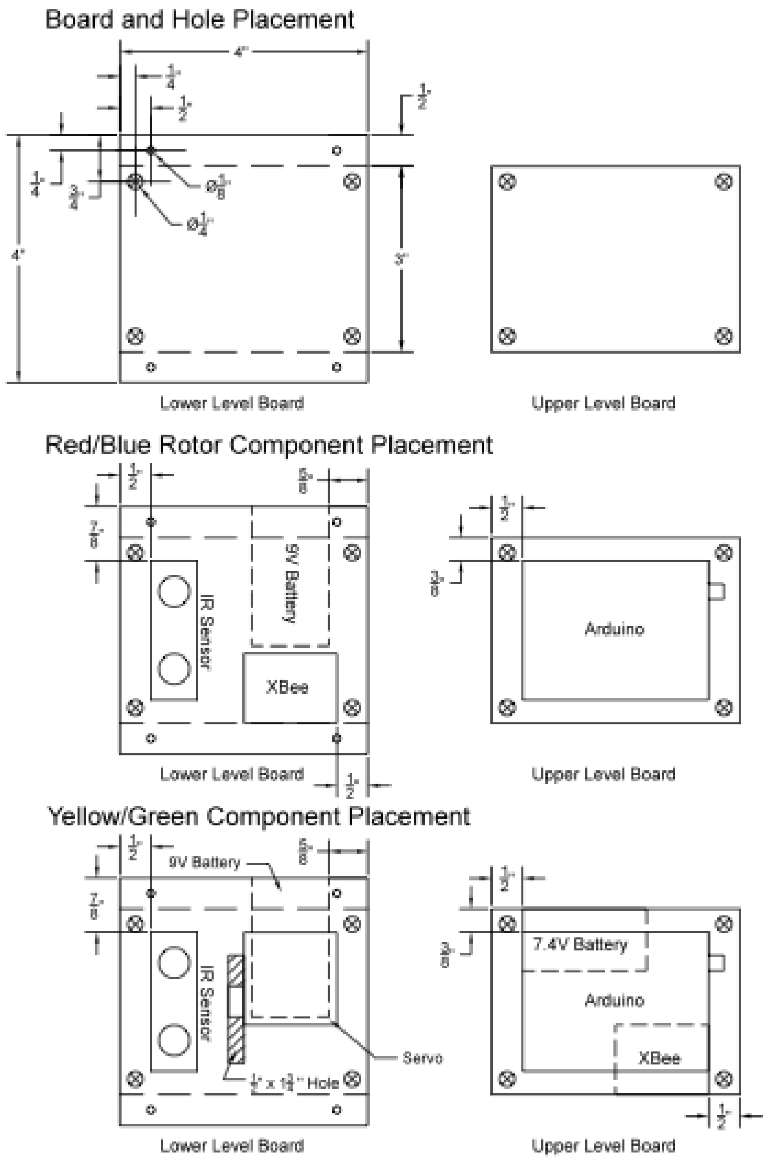

Figure 1. Board design and component placement for both the RB and the YG board sets

configuration of the board in addition to the component placement for each rotor set. The 9V battery that supplies power to the system is placed on the bottom of the lower board so that it may be easily removed and replaced without removing the entire board from the rotor. After adding the vertical layer it was found that on the YG boards there was not enough room to place the XBee next to the servo. Additionally in order to power the servo a 7.4V battery was required. This Lithium Polymer battery is solely responsible for powering the servo. This battery is placed on the bottom of the upper boards. With this placement combination, essential parts retain functionality with the Upturn I, with the implementation of new hardware.

Installation of these boards into each rotor section was achieved with two walls that are secured using epoxy. The majority of the wall is made of spare foam from the rotor project while the mounting, which the screws screw in to, are ¼” x ¼” x 4” wooden blocks. The 4” x 4” hole is cut in the bottom of the rotor, and the walls are constructed by attaching the wooden block to the foam wall section. Next, the top of the wall is attached to the inside of the upper surface of the airfoil while the wooden block is attached to the hole on the lower surface. On the RB rotor pairs the servos are not attached to the boards like on the YG rotor pair. Instead, the servos are attached via two wooden braces. The servos screw into the braces and on aft brace there is a small notch to allow the small extension of the wiring extending from the servo to be removed. With this the servos can be removed and replaced as necessary. The foam must be reasonably protected from shearing sections of the foam out from the surface deflections. To provide this protection, balsa wood plates are attached around the servo installation. These can be removed via the flat screws if access to the servos is required.

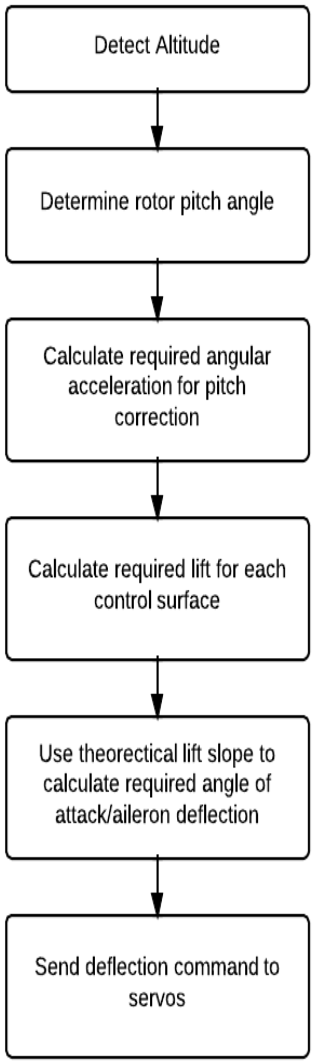
Detect Altitude
Determine rotor pitch angle
Calculate required angular acceleration for pitch correction
Calculate required lift for each control surface
Use theorectical lift slope to calculate required angle of attack/aileron deflection
Send deflection command to servos

Figure 2. Basic Control Scheme

VI. Control Law

A basic exploration of the control law that would govern the deflection of the control surfaces was explored. The governing process of the control system is shown in Fig. 2. The control system calculates the distance between the two rotors, then it calculates the moment required at each wingtip in order to correct for the helicopter pitch. With this the $\ddot{\theta}$ is found, which is then used to calculate the lift required on the lower rotor to correct for the rotor pitch. With the lift required the angle of attack required from the rotor can be calculated from the lift slope curve calculated for each wingtip.

Mark Drela’s Athena Vortex Lattice integrated with MATLAB was used to find the lift slope for each wingtip section. The complete YG pairs were modeled while only the control surfaces were modeled on the RB pairs. These selections reflect the sections that the control system has the ability to utilize for corrections. However, since airfoils sections on this vehicle are moving at different velocities due to the rotation, corrections are applied to the rotor model. Instead of applying corrections to the velocity via dynamic pressure, q, the corrections are applied using scaled corrections to the chord length. The lift slopes for each control surface section are shown in Fig 3.

I. Control Law Implementation

There were multiple paths for designing the control system for the HPH. The finalized design is based on the Arduino programming language, a sub derivative of C. The Arduino code that was used for the human powered helicopter control system available though the Cal Poly HPH team. The following section details specifics of this code and the implementation structure of the control software.

The system is broken up into two different files, the first file, known as the master file and the second file known as the secondary file. Starting with the master file Lines 4 through 26 declare the variables that are being used in the program and any needed packages for supporting devices such as the servos. Line 25 is where the PID control object is declared and initialized. Line 28-36 is where the Arduino initializes itself upon startup. The serial data rate is set along with attaching the servo and initializing more PID values for later use. Line 42 is the start of one of the if statement blocks based on the time that the Arduino is turned on for. A 7 second startup and servo movement is completed before the Arduino continues onward with anything else. During this time the operator needs to verify that the servo moves a full 160-180 degrees, if not then the box needs to be powered down and proper steps for troubleshooting should occur before proceeding with flight. After this servo movement process has occurred then the Arduino continues onward with the regular program, starting on line 63. Line 66 is where the master takes a sensor reading from the IR sensor linked to the analog in pin, A0 on the Arduino and it stores this for later use. Line 69 is the start of the communication code section. Here the master Arduino is waiting for sensor data to be sent from the secondary Arduino box. Once the data is received it proceeds with interpreting the packets from the secondary Arduino. The packet structure for the wireless system that is implemented is a basic bracketed system. The data is sent in the following form: <123.12>, where the floating point value inside is what the secondary Arduino sensor recorded. The communication code then interprets this data while making sure that both flag markers, < and > and located so that the system can parse the inner number and not read an incorrect value in the data stream. This whole process occurs at roughly a few milliseconds with a then built in delay of 20 milliseconds to give the master Arduino time to clear the buffer of any unused data for the next reading. Once this is done the value is stored in memory and the Arduino moves to line 114. Line 114 to line 119 are added for debugging purposes to see what values are sent and received from the secondary Arduino while directly connected to the master Arduino. Line 120 starts the first data calculation portion at time step 1. It then calculates the difference in height measured by the two sensors (master and secondary) and then proceeds to calculate the current angle of the wings. This is stored in memory and a counter is incremented later so that another calculation can be performed at time step 2. Once both angle and delta differences are calculated the difference in angle is calculated at line 142. From there the difference between angles and time are calculated to get theta dot for the system in two different time states. After theta dot is calculated for both time states then theta double is calculated on line 162. This is then used in calculating the lift in the system. Lines 163 to 175 are more testing and debugging lines for the code. Then the calculation for the PID system and lifts starts on line 176. Once the value of alpha2 is determined from the lifting system equations it is fed into the PID controller and the result is computed. This commanded alpha value is then sent to the servo on the master Arduino and passed to the secondary Arduino which commands the reverse angle. The process is repeated by taking measurements again and starting the process all over at line 63. Meanwhile the secondary Arduino is taking measurements and sending them to the master Arduino while listening for a message from the master Arduino so that the servo can be commanded to a specific angle.

II. Results and Discussion

The result of the sensor test was inconclusive and needs to be looked into further. The test was the system on a rigid ten foot board to simulate the wing and placed at five feet in the air. The test was to see the maximum angle that the sensor could handle and the effect of surface finish on detecting range. The ultrasonic performed better at a larger angle and there were no discernible differences between different surfaces, as seen in Tables 1&2, although, the ultrasonic sensor did tend to jump around much more than the IR sensor and would frequently jump up to a large distance (>1000). The IR sensors had a large error but were consistent, as seen in Tables 3&4; leading us to believe it is an error in the program that is sensing the distance. The consistency of IR was promising but further testing needs to be done. The sensors output a voltage that needs to be converted into distance and for the IR sensor that equation is non-linear, so we tried linearizing it to get the distance. The error in this conversion might be the cause of the error in the testing and should be looked into more. Testing was done on different surfaces to see if the reflectivity of the surface made a difference on the detected range. We didn’t know which surface finish we were going to have for the test flight (either very glossy with the gym floor or fairly rough with foam pads). What the data shows that the surface finish did not affect the ultrasonic sensor and while the IR was, but we are unsure about the effect on the IR due to a difference in surface temperatures of the surfaces. The ultrasonic performed better at greater angles, but if the helicopter were to get to fifteen degrees the control system would not be able to fix the problem before crashing so that great of an angle is unnecessary.

Table 1. Ultrasonic Sensor Test Data Run 1

	Rough Concrete			Smooth Concrete			Tarp		
Angle	Actual	Measured	% Diff	Actual	Measured	% Diff	Actual	Measured	% Diff
0°	0’	0.06’	NA	0’	-0.04’	NA	0’	-0.07’	NA
5°	0.80’	0.65’	18.8	0.80’	0.65’	18.8	0.80’	0.65’	18.8
10°	1.64’	1.5’	8.5	1.64’	1.5’	8.5	1.64’	1.7’	13.7
15°	2.55’	2.2’	13.7	2.55’	2.1’	17.6	2.55’	2.2’	13.7

Table 2. Ultrasonic Sensor Test Data Run 2

	Rough Concrete			Smooth Concrete			Tarp		
Angle	Actual	Measured	% Diff	Actual	Measured	% Diff	Actual	Measured	% Diff
0°	0’	-0.04’	NA	0’	0.04’	NA	0’	0’	NA
5°	0.80’	0.6’	25	0.80’	0.65’	18.8	0.80’	0.78’	2.5
10°	1.64’	1.5’	8.5	1.64’	1.6’	2.4	1.64’	1.6’	2.4
15°	2.55’	2.2’	13.7	2.55’	2.2’	13.7	2.55’	2.2’	13.7

Table 3. IR Sensor Test Data Run 1

	Rough Concrete			Smooth Concrete			Tarp		
Angle	Actual	Measured	% Diff	Actual	Measured	% Diff	Actual	Measured	% Diff
0°	0’	0.05’	NA	0’	0.05’	NA	0’	NR	NA
5°	0.80’	0.20’	75	0.80’	0.40’	50	0.80’	NR	NA
10°	1.64’	0.65’	60	1.64’	0.82’	50	1.64’	NR	NA
15°	2.55’	0.92’	64	2.55’	NR	NA	2.55’	NR	NA

Table 4. IR Sensor Test Data Run 2

	Rough Concrete			Smooth Concrete			Tarp		
Angle	Actual	Measured	% Diff	Actual	Measured	% Diff	Actual	Measured	% Diff
0°	0’	0.03’	NA	0’	0.07’	NA	0’	NR	NA
5°	0.80’	0.30’	63	0.80’	0.40’	50	0.80’	NR	NA
10°	1.64’	0.62’	62	1.64’	0.79’	52	1.64’	NR	NA
15°	2.55’	0.79’	70	2.55’	NR	NA	2.55’	NR	NA

The control law has not been tested using the hardware. The change in height should make the servos respond with the lower side increasing the angle of attack. This can be shown using the same test apparatus and measuring the angle of the servo on the lower side (this can be looked at on the computer in real time instead of measuring it with a protractor). The angle response it related to the difference in height and can be adjusted using a different scalar. The one we picked was based on our control law and assumed that the theoretical lift generated was greater than the actual lift, so we increased the angle of attack to compensate for this decrease in lift. This ultimately needs to be tested during a test flight of the entire system to see if the response actually controls the system the way we expect it to. We recommend that a fifth board is set up outside the system to collect and save the data during the test flight so that the data can be analyzed after the flight.

The last measure of merit of our system was the difference in mass from the original system versus the new system. The old system weighed a total of 25.5 oz and the new system weighs a total of 30.4 oz. The new system is heavier than the old system because we needed two batteries to run the system due to the robustness of the microprocessor and the servo. In later iterations of the control system the microprocessor can be replaced with a smaller one, but the servo should remain the same due to its great response time. The Arduino was the chosen microprocessor due to its processing power and ease of programming, but there are smaller Arduinos with fewer connections but similar processing power that could be switched to. The smaller Arduinos could use less power and run off of the servo battery to reduce that mass. Also, if the requirement of fitting on the old system was taken out there would be no need to go to a two story board that we needed to do.

III. Conclusion

The sensor selection came up inconclusive, the servo response needs to be validated using a test flight of the Upturn II, and the control system was heavier than the original system.

Although, the system didn’t really improve upon the old system, the first iteration was meant to replicate the old and was part of the learning experience. Designs take many iterations and we expect this design to go through its fair share. All designs are based on the requirements and the more freedom given to the designers the easier it becomes. The requirement to fit into the old system was almost not met, but was a requirement generated by us. If that requirement was taken away it would allow you to optimize the size and reduce the weight. Analysis could be done on the system using SimMechanics (a MatLab add-on) based on the thesis work of Sean Brown.