Helicopter Automation with Control Systems (HAWCS©)
School of Engineering, Burnaby, B.C., V5A 1S6
arg-heli@sfu.ca

December 20th, 2001

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, BC
V5A 1S6

Re: ENSC 340 Process Report for and Automated Hovering Helicopter

Dear Dr. Rawicz:

We've attached our Process Report for our ENSC340 project, "Autonomous Hovering Helicopter", along with this letter. The Process Report contains a brief overview of the project system and a discussion of the evolved configuration with considerations to the problems that occurred in the development process. Further, the document will provide some discussions of changes that would make the automated helicopter flight more reliable and an overview of future project considerations.

The Our HAWCS© team contains a talented and cohesive group of engineers – Shahin Roboubi and Jimmy Tsai, two embedded system programmers / Linux specialists, Michael Adachi and Michael Mierau, two control system engineers, and Neil Patzwald, the hardware specialist. Details of the group and the project can be viewed through the project website and its links at www.sfu.ca/~arg/hawcs/index.htm. Furthermore, if you have any questions or suggestions related to the project for us, please feel free to send an email to our team at arg-heli@sfu.ca.

Sincerely,

Jimmy Tsai
Project Manager,
HAWCS, Inc.

Enclosure: Design Process Report for an Automated Hovering Helicopter

Helicopter Automation with Control Systems (HAWCS©)
School of Engineering, Burnaby, B.C., V5A 1S6
arg-heli@sfu.ca

Process Report for an Automated Hovering Helicopter

Project Team: Michael Adachi
Mike Mierau
Neil Patzwald
Shahin Roboubi
Jimmy Tsai

Contact Personnel: Jimmy Tsai
jtsai@sfu.ca

Project Website: www.sfu.ca/~arg/hawcs/index.htm

Submitted to: Andrew Rawicz
Date: December 20, 2001

Table of Contents

Table of Figures and Tables

1. Introduction

With the advent of global positioning systems and continued miniaturization of electronic components, computers of smaller payloads may be integrated as control systems on miniature aircraft. These miniaturized systems have recently provided video links to police and fire services to provide search and rescue assistance without the endangerment of human lives in the process. Further, recent developments in World politics may have increased the significance in the development and marketability of the small autonomous controlled aerial vehicle. Moreover, advances in navigation systems and feedback control are now being integrated by the Aerial Robotics Group (ARG) from Simon Fraser University School of Engineering Science in this course project, to provide the first step in the autonomous flight of a small unmanned helicopter.

The development of the autonomous control of an aerial vehicle will consist of three stages, of which we hope to cover the first two in this project and the final stage in the coming spring of 2002. In the first stage of development, our project group will produce an aerial model that will react to sensor feedback when tilted and compensate with rotor corrections in a static testing mode. This testing will assure that the hardware and software are working in unison to provide the required functionality and reliability to ensure that there will not be any damage to our helicopter in flight tests. Therefore, successes in the initial stage will lead to quicker second stage development, consisting of flight-testing to demonstrate the finely tuned features resulting from the static testing procedures. In the spring of 2002, the third stage of development will consist of a programmed autonomous flight with no user input.

This document is provided to detail the Process Report for an Automated Hovering Helicopter. The Process Report contains a brief overview of the project system and a discussion of the evolved configuration with considerations to the problems that occurred in the development process. Further, the document will provide some discussions of changes that would make the automated helicopter flight more reliable and an overview of future project considerations.

2. System Overview

Our project is centered on a series of control systems performing the adjustments required to maintain helicopter's stability. The control systems allow a higher-level application to "command" the helicopter using these control loops. Using this method, we hope to eventually achieve complete autonomous flight. For the scope of this project, the goal will only be to maintain an autonomous hover, which will be a starting point for future additions such as forward flight and surveillance set point flight.

A laptop computer, connected to the GPS base station is used as the base station, which will have a software interface to control the helicopter through a wireless Ethernet connection. The laptop computer collects GPS data and sends it to the helicopter computer. The helicopter also has an onboard GPS remote station, which also collects GPS data. The process of using both sets of data from the two stations is called Differential GPS, and will allow better accuracy of XYZ coordinates positioning of the helicopter. Further, an override board enables the manually controlled flight through a RC radio and enables the switch to computer control when appropriate. The computer control reads GPS data from the two GPS stations and additional information in the form of pitch (forward/backward angle), roll (left to right angle) and yaw data (compass angle) from the Tokin sensor mounted onboard the helicopter. Further, the Precision Navigation TCM2 sensor supplies addition movement information. The accumulation of positioning data will enable the computer control to compensate for deviations in coordinates, or deviations in helicopter angles through specially designed application software. Flight compensations are attained through the use of servos that provide mechanical actuation to control the aileron (affects pitch angle), elevator (affects roll angle) and collective pitch (affects height) movements on the in-flight helicopter. Another pair of servos provides the mechanical actuation to control an onboard governor, and the yaw servo that keeps the helicopter pointed in one direction by the control of an onboard gyro.

Figure 1 details the overall flow of data and control throughout the system. The sensors are read through digital ports on the computer and then use control loops to determine the proper compensation factors required to stabilize the helicopter. Compensation signals are then sent to electronic servos that physically change the appropriate onboard devices.

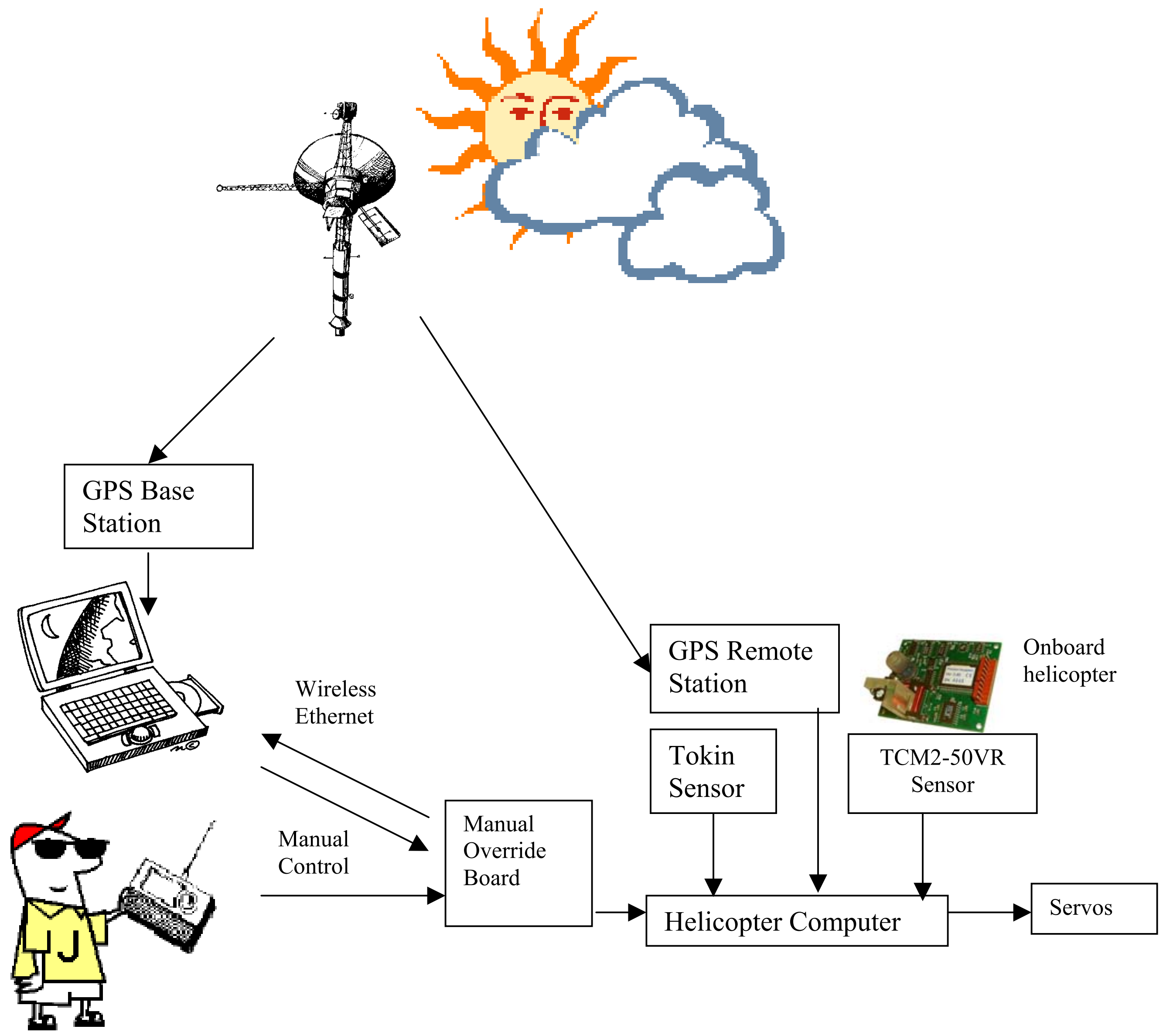

Figure 1: System Overview

Furthermore, Figure 2 details the sub-system integration and the information flow through the processing unit. The constant flow of information from the sensors to the servo is required to maintain a robust control system.

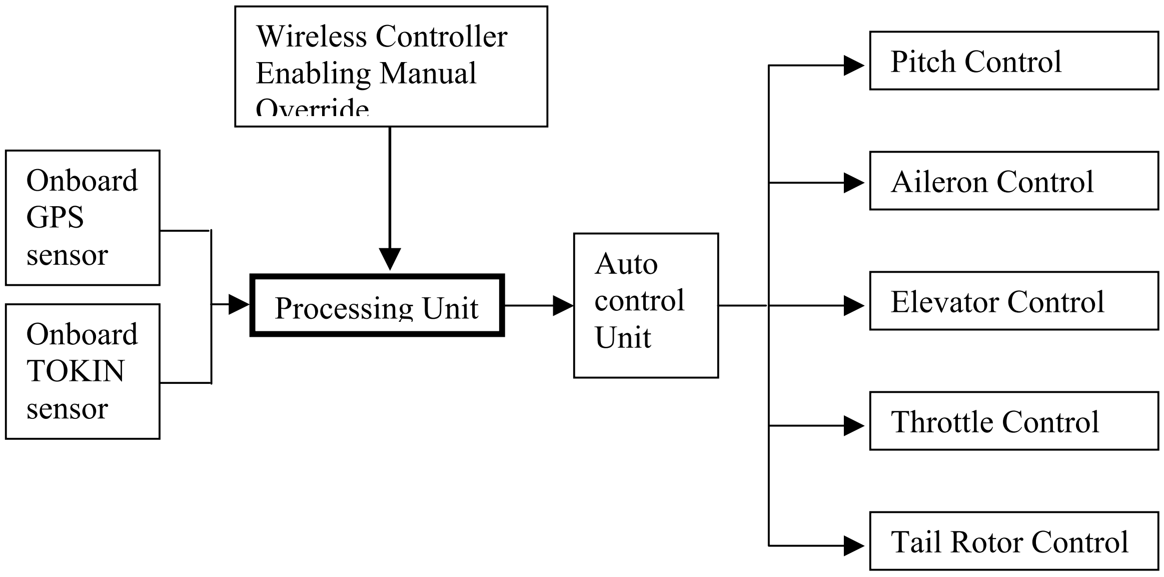

Figure 2: Component Sub-system Integration

3. Technical Implementations

3.1 Hardware

3.1.1 The Helicopter

The backbone of the Automated Helicopter is the TSK Mystar 60 model helicopter. The TSK Mystar 60 is a lightweight RC controlled model helicopter powered by a 2-cycle nitro-methane fueled motor, thereby, giving it a high power to weight ratio. Figure 3 shows the TSK helicopter in a static position with the hardware configurations mounted and the rotor blades spread in the flight position. Further, Figure 4 details the onboard hardware configuration necessary for automated flight capabilities.

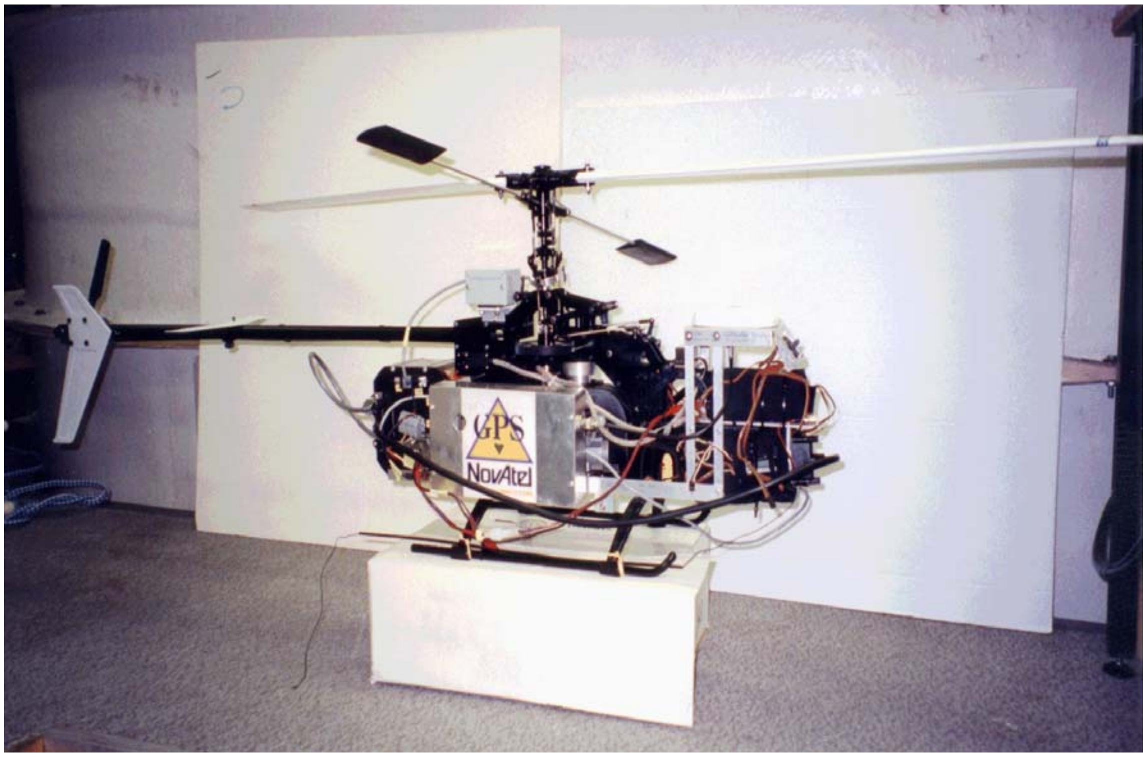

Figure 3: TSK Mystar 60 Helicopter

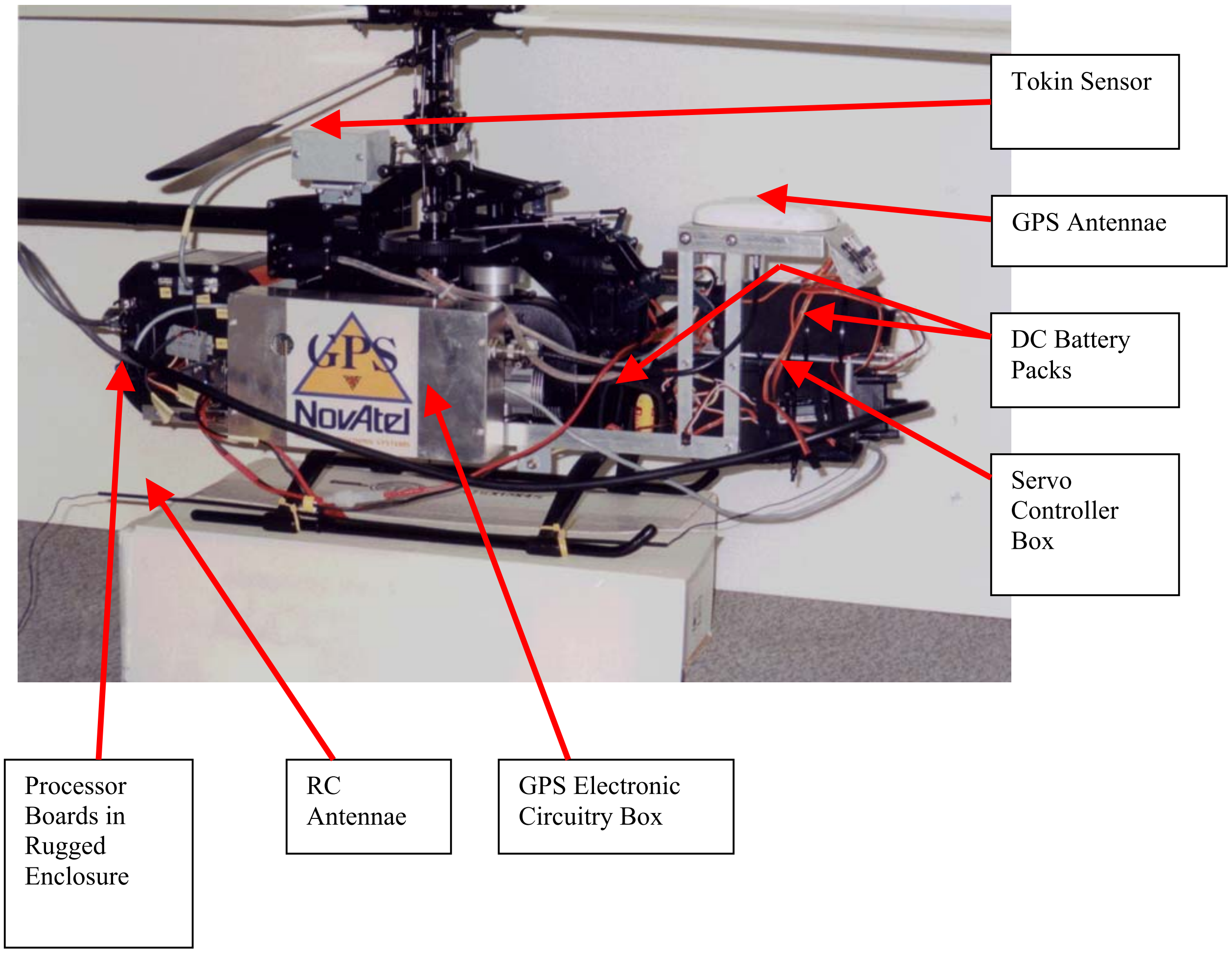

Figure 4: System Component Placement on Helicopter

3.1.2 The Static Testing Table

The system testing can be accomplished in a static method on a specially designed tilting table. The table has been designed to enable tilt and roll of the helicopter by the manipulation of several planes (as seen in Figure 5). Therefore, the static testing table enables testing of simultaneous tilt and roll compensations made by the processing unit through servo movements and movements of connected structures (see Figure 6).

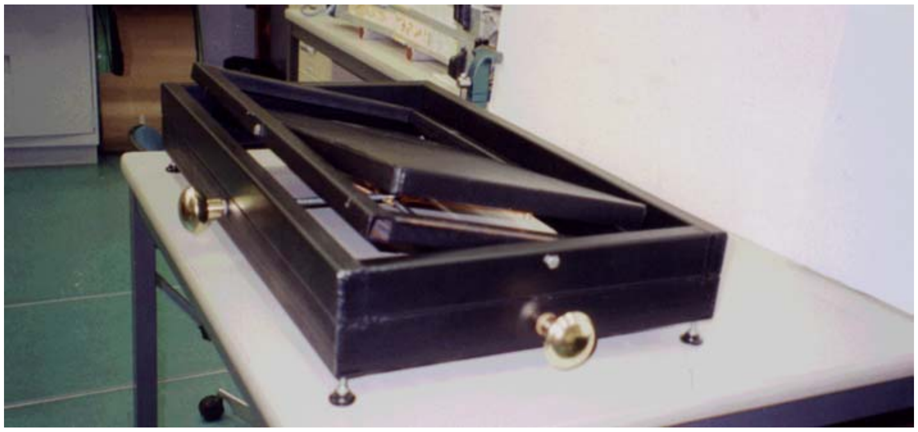

Figure 5: Multi-plane Testing Table

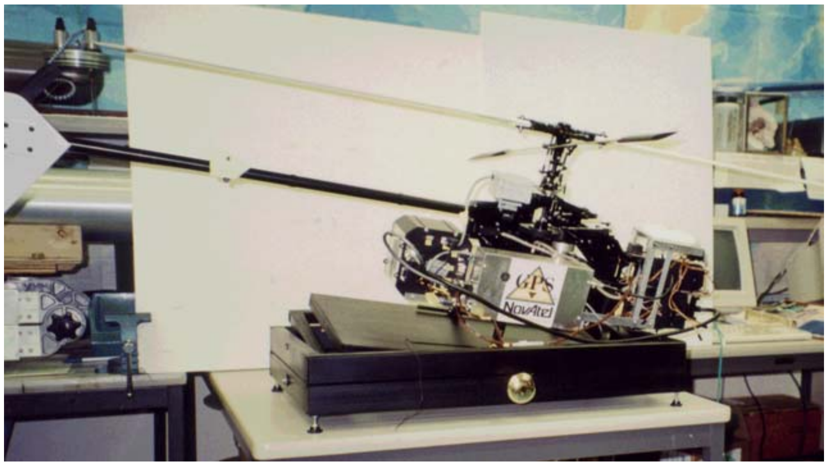

Figure 6: Multi-plane Testing Table in a Forward Roll Tilt Configuration

3.2 Software

The configuration of the control software that was designed to control the flight of the Automated Helicopter is shown in Figure 5. Each of the elements is critical in the flight of the helicopter and the dependency of each element is detailed in the flow chart. The project operating system is Linux with Real Time Extension Module. Linux has the advantage of being a stable, free, open (source code available) and is widely supported on Internet.

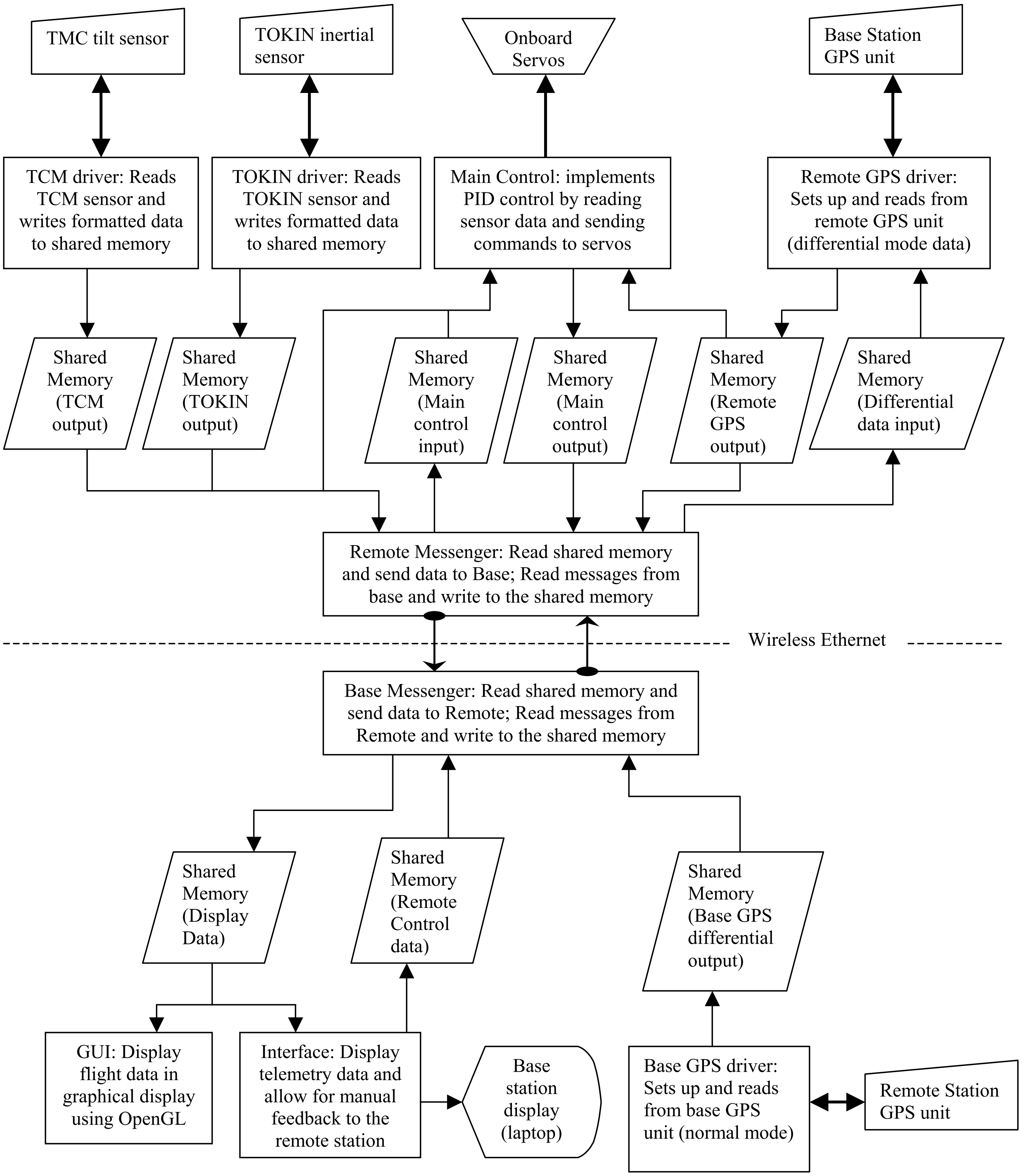
TMC tilt sensor
TOKIN inertial sensor
Onboard Servos
Base Station GPS unit
TCM driver: Reads TCM sensor and writes formatted data to shared memory
TOKIN driver: Reads TOKIN sensor and writes formatted data to shared memory
Main Control: implements PID control by reading sensor data and sending commands to servos
Remote GPS driver: Sets up and reads from remote GPS unit (differential mode data)
Shared Memory (TCM output)
Shared Memory (TOKIN output)
Shared Memory (Main control input)
Shared Memory (Main control output)
Shared Memory (Remote GPS output)
Shared Memory (Differential data input)
Remote Messenger: Read shared memory and send data to Base; Read messages from base and write to the shared memory
Wireless Ethernet
Base Messenger: Read shared memory and send data to Remote; Read messages from Remote and write to the shared memory
Shared Memory (Display Data)
Shared Memory (Remote Control data)
Shared Memory (Base GPS differential output)
GUI: Display flight data in graphical display using OpenGL
Interface: Display telemetry data and allow for manual feedback to the remote station
Base station display (laptop)
Base GPS driver: Sets up and reads from base GPS unit (normal mode)
Remote Station GPS unit

Figure 6: System Software Configuration

3.2.1 TOKIN Driver

The TOKIN driver is responsible for reading raw data from the USB port, which is connected to the TOKIN sensor, and calculating or formatting it into integer type data. This data includes fields for roll, pitch, change in roll, change in pitch, and change in yaw. After the data is processed, it is written to shared memory where is can be read by other processes.

3.2.2 TCM Driver

The TCM Driver is responsible for reading raw data from the serial port connection of the TCM sensor. The TCM driver program calculates or formats the data stream into integer type data. This data includes fields for roll, pitch, yaw, and temperature. Errors from the TCM sensor are also read, checked validity and recorded. Wherever possible, invalid data is corrected or ignored, if correction is not possible from the data supplied. Further, after processing, the correct data is written to shared memory where it can also be read by other processes.

3.2.3 Main Control

All of the automated control is handled through this main control process. It reads sensor data, performs calculations to determine the various gains for controlling the helicopter, and sends the acquired data to both the servos and the shared memory so that the base station can continue constant monitoring. The program calculations involve determining the PID gains using the sensor data, the desired location and height values input from the sensors. Further, many error checking and limiting functions are also implemented in this software process for safety and reliability reasons.

3.2.4 Operator Interface

The operator interface is displayed on the base station laptop as an advanced text based user interface. It displays all telemetry data from the sensors and main control. Further, it allows for the dynamic modification of desired position values that are sent back to the remote station and PID constants for changes in desired amounts of servo outputs.

3.2.5 Graphical User Interface

A Graphical User Interface (GUI) was implemented to display the current state of the helicopter on the screen. The program was written in C++, with the graphics drawn using OpenGL. Since the GUI is loaded in a separate process, it can be minimized or repositioned anywhere on top of the telemetry display, however, it is initially located in the top right corner of the screen. The GUI displays the helicopter orientation (roll, pitch, yaw), compass heading, and temperature. A frame rate counter is also displayed so we can monitor the performance when all of the processes run simultaneously. Currently, on our 300 MHz notebook computer we are achieving almost 3 frames per second.

3.2.6 The GPS Driver

The GPS driver enables position coordinate data to be read from the Novatel GPS stations. Further, the driver enables communication to each base unit to change the required configuration settings and operating mode, thereby easily enabling the exploration of the full functionality of the GPS equipment capabilities.

3.2.7 User Data Protocol

Two User Data Protocol (UTP) processes are responsible for all of the wireless communications, one running on the base station laptop called Base UDP and one running on the remote station helicopter computer called Remote UDP. Each process is dedicated to receiving and sending wireless Ethernet packets while reading and writing this data to shared memory. The present Base UDP configuration acts as the clock for the sending and receiving processes, using a 0.25sec timer to send the periodic packets. The Remote UDP process waits for to receive this packet and then sends the Remote station data to the Base UDP process. There are other methods of doing this such as using the external timer such as using the GPS Base Station as the timer when it updates differential GPS data every 1 sec. However, to avoid the dependency on the GPS Base Station, we used a timer in the Base UDP process.

Furthermore, as previously illustrated in Figure 5, the Base UDP process reads shared memory from the “Remote Control” and “Differential GPS” Shared Memory every 0.25 sec, and then appends this data together into one packet and sends it off to Remote UDP. Once Remote UDP receives this data, it will split the received information back into two parts, “Remote Control” and “Differential GPS” data, and writes them to “Main Control In” and “Differential GPS” shared memory. Immediately afterwards, the “TCM”, “TOKIN”, “Remote GPS” and “Main

Control Out" Data are all read from shared memory and appended together and sent out to Base UDP. Once Base UDP receives this data, it will be written to "Display" shared memory without splitting the data up. Further, the whole cycle repeats on the next timer count.

3.3 Control

The control model consists of five control loops, the pitch angle of the helicopter (circular path of rotor blades), the roll angle, the left/right position, forward/back position and vertical position. A simplified hover model that neglected the drag forces on the helicopter tail rotor and fin was configured from several complex algorithms listed in our references. Our control program implemented a version of the V.Gavrilets configuration for the XYZ control models, and the Kadmiry model for the roll and pitch. The XYZ helicopter dynamics measurements were made for our RC Helicopter whereas the roll pitch helicopter dynamics were based on another RC helicopter very similar to ours. The control loops have been designed and implemented as shown in the Figure 7 and 8. In the control loop, we assumed that the helicopter fuselage to be parallel to the disk (circular path of the rotor blades) to make the model simpler.

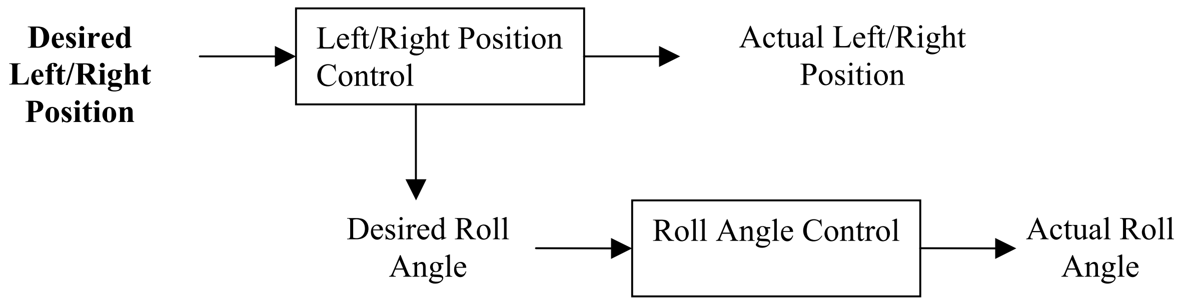
Desired Left/Right Position
Left/Right Position Control
Actual Left/Right Position
Desired Roll Angle
Roll Angle Control
Actual Roll Angle

Figure 7: Roll Model Overview

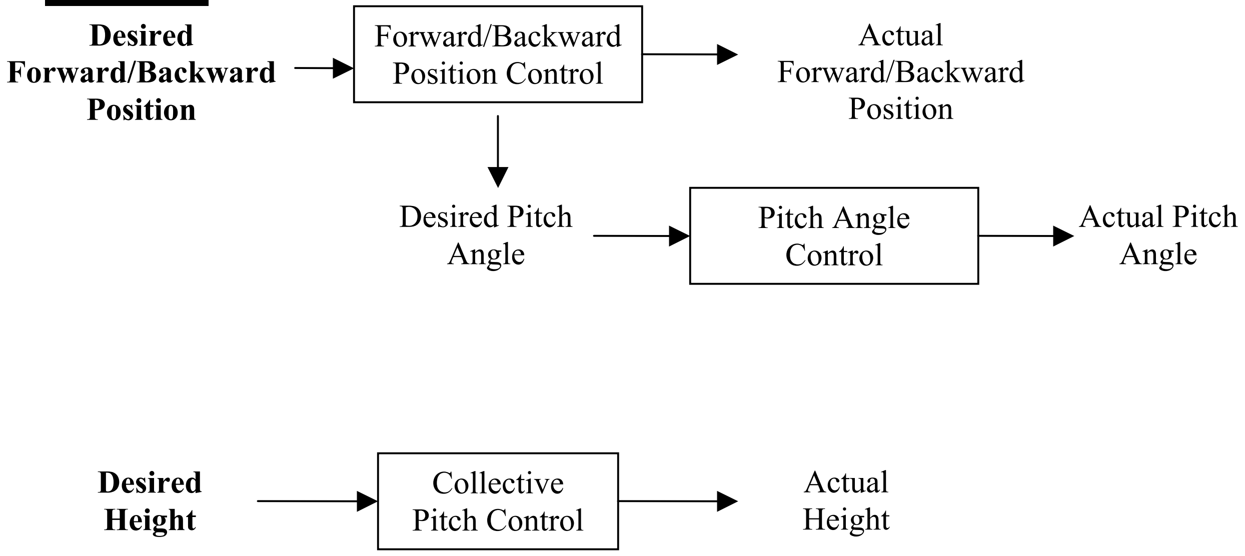

Figure 8: Pitch and Collective Control Model Overview

4. Problems Encountered

4.1 Hardware

4.1.1 Hardware Failures

The TCM fluid sensor, which Pavel Haintz acquired two years ago for the SFU aerial robotic group, had performed steadily during the period. Due to its data reliability compared to the TOKIN3D sensor, HAWCS group member decided to use it as our primary inertia sensor input. Unfortunately, it stopped functioning three weeks before we were schedule to do our demo. There was panic around our team because the sensor replacement cost was 400 dollars Cdn to replace and just happened to fail near the American Thanksgiving holiday which resulting in a considerable waiting time for the replacement to be shipped. However, through the diagnostic help of the TCM support personnel, we believe that RS232 chip on board may have been damaged and upon repair will provide a spare unit. Furthermore, our group received unexpected funding from the International competition authority due our last year’s performance that helped to fund the sensor replacement.

Upon testing, we found that one of the pairs of GPS antenna and receiver was also broken at the start of the project. It was very odd because we didn’t anticipate having both the antenna and receiver broken at the same time. Since the GPS equipment is on loan and is very valuable, Novatel wanted considerable data confirmation before they would replace the faulty equipment. Therefore, it took weeks of testing for us to confirm that they were indeed both faulty. Further, our speculation leads us to suspect that they might have been mishandled by Shipping company while in route to the summer 2001 competition site in Washington DC. Fortunately, we had contacts with Novatel GPS and they gave us temporary loans so that we can complete our project. As a major aerial robotics competition, Novatel have now fixed our receiver for no cost, however, another company makes the antenna and we will have to pay a bill of 379 dollars Cdn to have that replaced.

4.1.2 Inclement Weather

Most of the difficulties in our sensor testing were due to foul weather. Our GPS required the coverage of 6 or more satellites that was only attainable with receiver antennae mounted in open spaces. However, every time we tested the GPS outside, we ended up spending a lot of time because some of the required element failed to function properly (e.g. power, wireless Ethernet, or programming). Further, the cold resulted in a loss of motivation and resulted in a loss of productivity. Eventually, we decided to use the balcony above the pit to be our testing spot, since that place has accessible power and limited, but usable GPS satellite coverage. Most importantly, our team can stay in the comfort of a warm and dry environment, which enables us to work for several hours without freezing ourselves to death. Therefore, on the same day that we picked the balcony, we were able to attain 2 cm of differential GPS precision after several hours of tweaking. Moreover, if we had to do it again, we would test everything possible inside before considering taking any of the testing equipment outdoor. If we had done that right from the beginning, we might save ourselves weeks of testing time and frustration.

4.2 Software

4.2.1 Software Engineering

An important consideration in the project review is the timing of the software engineering. The software planning and design was not done until late in the project because our project group initially believed that other hardware and control implementations should have been started so that the necessary software design would be more clearly defined. However, we neglected to develop an overall implementation plan and kept working on individual components of software. This led to the need for reworking much of the software in the last few weeks to develop a workable system implementation. Further, another thing that might have helped organize the project and speed things up was an implementation of a software version control. This concept would have reduced the confusion when many people worked on a single piece of code and may have reduced code loss. Moreover, bad software engineering led to the need for much of the software to be reworked in the last few weeks.

4.2.2 System Damage

Our project computers were left hooked up to the SFU network and experience some damage by an unscrupulous hacker. Therefore, all of our program files had to be checked for damage and the main program has to be reloaded to ensure that there were no lasting computer problems. Furthermore, the evaluation led to a project downtime of one week before we could continue to forge ahead with project development.

4.2.3 Control System

The main difficulty we encountered while creating our MATLAB Simulink RC Helicopter Control simulation was the complexity of the helicopter dynamics. Therefore, we used a simplified hover model and neglected the drag forces on the tail rotor and fin to eliminate an overly complex algorithm, but provide a workable model. We used elements of V.Gavrilets article for the XYZ control models and the Kadmiry article for the roll and pitch models. The XYZ helicopter dynamics measurements were made for our RC Helicopter whereas the roll pitch helicopter dynamics were based on another RC helicopter very similar to ours.

5. What Our Group Would Have Done Differently

5.1 Jimmy Tsai, (Embedded System Programmer)

I learned the advantages of having diverse team talent. We have people who have control system, embedded programming, openGL programming, and hardware experience. Each one of us has been able to contribute on our specialty and lift the project together.

Technical wise, I learned about Linux administration in areas of networking and system configuration. By writing communication programs between helicopter computer and my laptop, I learned about UDP programming and share memory usage. Furthermore, I also learned about the elements of interface programming. Last but not least, I wrote our driver for Novatel differential GPS set and thus gained experience interfacing and operating with GPS hardware.

Because two of our sensor broke down, I had to contact customer support from both companies. I worked with those people to determine the fault in our hardware and eventually sent it back to the company for replacement. Although not technical, it is a valuable experience because of its significance to our project success. We are lucky enough to have both sensors replaced just in time for our project demo.

5.2 Shahin Roboubi, (Embedded System Programmer)

I learned a lot about Linux and how to work within the unprotected environment that it provides. I also gained experience with coding and low-level system call functions. Most importantly I learned that the best way to relieve stress and maintain healthy team dynamics is to take some time out once in a while and play a game or do something non-work related with the team members.

5.3 Michael Adachi, (Control System Engineer)

From the experience I obtained during this project, I learned that it’s a very good idea to have backup hardware. When our critical sensors, TCM and the GPS base station did not work, we noticed that there were a lot of dependencies between different parts of our project. The GPS base station not working which meant we could not test whether the GPS driver worked, which meant we postponed our position control testing and position data sending. I believe a flow diagram showing the tasks that were dependent on which other tasks would be a very useful.

I also better realized that being able to work well with the group members is a very critical part of not just how well the project turns out, but also how willing you will be to continue on with the same project after due dates.

5.4 Michael Mierau, (Control System Engineer)

During this semester, I worked cohesively with a relatively large group of people with various backgrounds to achieve an ambitious task. I believe that through effective planning and coordination with group members, we were able to be both productive and efficient, thereby enabling us to finish on schedule. I was surprised by the amount of skills I applied to this project that I learned in other courses. In particular, I found MACM equations could actually be applied practically, in the case of differentiation in the PID loop, and cubic spline interpolation for throttle calibration. In addition, ENSC 383 proved to be a useful course, since it helped with modeling the control system. Furthermore, ENSC 488 helped with OpenGL programming also.

Some knowledge I gained from this project; are an increased understanding of helicopters and their operation, how to model dynamics equations using Simulink, and familiarity with the Linux operating system and shared memory implementation.

5.5 Neil Patzwald, (Hardware Specialist)

In this project, I learned that software configurations in a multiprocess control system could become large and hard to follow. Further, I have greater respect for those that can wade through the masses of code lines and find trouble spots.

Furthermore, I learned how to compile and format large documents. Since I have had limited experience with aspects of computer generated document, the experience gained though the document requirements of this course will prove invaluable in latter years.

6. Future Developments

6.1 Throttle / Collective Pitch Control

Under manual control the throttle is calculated automatically. However, when the helicopter is under computer control, the throttle curve must be explicitly calibrated as a function of collective pitch. In order to do this, we must take five measured points and construct a natural cubic spline that interpolates the data. We chose a spline as opposed to a single degree four polynomial because it is smooth function and monotonically increasing for our data points, whereas the polynomial oscillates between the points. Using numerical analysis, the system of equations can be solved to yield the coefficients of the spline. Hence, in the main control program, whenever a new collective pitch is sent to the servos, we can send the calculated throttle value to its corresponding servo.

6.2 Forward Vision Capable Camera Mount

The requirements of future competitions will require the integration of forward vision for the helicopter to complete the required tasks. Therefore, a camera mount has been designed and needs to be built in the coming months. The new design provides linear Radio Controlled movement with the signal being processed by the current processor configuration in the helicopter. The design of the camera mount is shown in Figure 8.

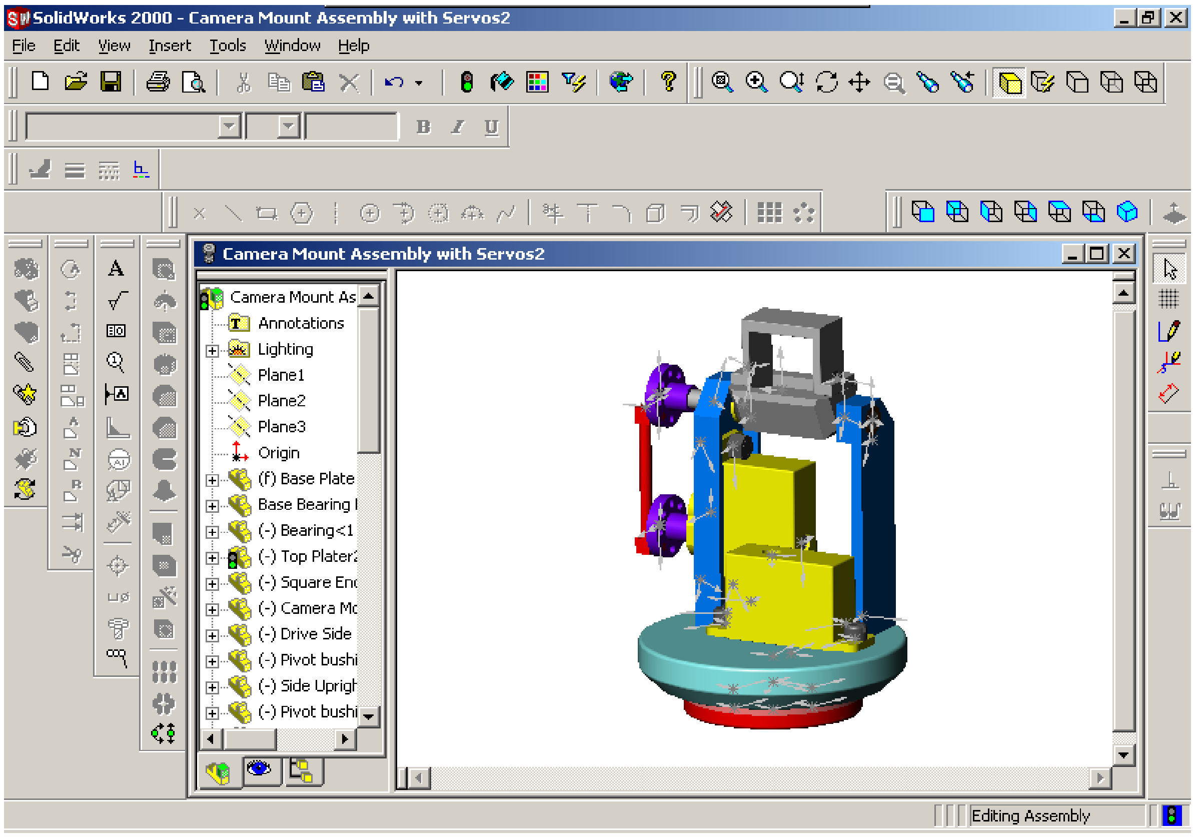

Figure 9: Forward Vision Camera Mount with 2 Servos

6.3 TOKIN Driver Development

In our present configuration, the TOKIN3D sensor is not a major part of our flight control sensor input. This is due to the large amount of time that would be required to develop a complex mathematical processing algorithm to extract precise and meaningful data. Fortunately, TOKIN is one of our sponsoring companies and has showed interest in helping to a proper Linux sensor driver necessary for future project implementation. Further, they have agreed to write a module doing signal processing to complement our existing Linux driver.

Moreover, from the interest and demonstration of the aerial robotics project last week, the TOKIN3D sensor inventor promised that he would send us the next generation sensor hardware, which is due next April. Therefore, we may be able to use the new hardware and software configurations for the July International Aerial Robotics Competition.

6.4 Forward Flight

After HAWCS corp. achieves the ability to hover autonomously, members of the HAWCS corp. and the rest of the SFU Aerial Robotics Team will use the result of the project as a basis for more sophisticated flight capability – most importantly the transition from hovering to forward flight. Since the scheduled timeline for hovering is January 2002 and the next aerial robotic competition is schedule in July 2002, six addition months will be available for future development. New modeling, control system tuning parameters and even new hardware will possibly be applied.

6.5 New Override Board

Currently, the development HAWCS corp. will be based on the initial Servo Override board, developed for year 2000 International Arial Robotics competition. An improved version of the board is currently under development by Lawrence Harris, as a member of the SFU Aerial Robotics Group. The new board performs the same functionality as the old board; however, it has the addition capability of reading the input to the servos. This additional function is significant because we will than able to read a helicopter pilot's input and apply Ziegler-Nichols tuning on our PID regulators.

7. References

A.R.S. Bramwell, Helicopter Dynamics. Wiley and Sons, Inc. 1976.

V.Gavrilets, et al. Nonlinear Model for a Small-Size Acrobatic Helicopter, American Institute of Aeronautics and Astronautics Paper, MA 02139. 2001.

B. Kadmiry, Fuzzy Control of an Autonomous Helicopter, IEEE, 2001