DEVELOPMENT OF A SEMI-AUTONOMOUS CONTROL SYSTEM FOR THE UVA SOLAR AIRSHIP *AZTEC*

Andrew Turner – aturner@virginia.edu
UVA Solar Airship Program[1]
University of Virginia, Charlottesville, VA USA

Abstract

The UVA Solar Airship Program is a student run undergraduate program whose goal is to develop a solar powered airship capable of continuous, autonomous flight. This paper presents the design and implementation of a semi-autonomous control system for our Phase II Airship *Aztec*. The control system includes the derived control algorithms, flight hardware, and operating software for the airship.

Nomenclature

F_x, F_y, F_z	=	Forces on the airship in the X,Y, and Z directions on the airship, N
μ_p, μ_s, μ_r	=	Angles of the port, starboard, and rear thruster from pointing down the x-axis, rad
T_p, T_s, T_r	=	Thrust of the port, starboard and rear thrusters, N
d_p, d_s, d_r	=	Distance of the port, starboard and rear thrusters from the center of buoyancy, m
K_p, K_d, K_i	=	PID Controller Gains
ω	=	Rotation Speed of the Propellor (RPM)
θ, ϕ, ψ	=	Pitch, Roll, and Yaw angles of the airship, radians
C_g	=	Center of Gravity
C_b	=	Center of Buoyancy
u, v, w	=	Translation velocity of the airship in the X, Y, and Z directions, m/s
p, q, r	=	Velocity of the airship about the X,Y, and Z axes, rad/s
L, M, N	=	Moments on the airship about the X,Y, and Z axes, N-m
X, Y, Z	=	Primary orthogonal axes of the airship

Introduction

In recent years, there has been an increased interest in autonomous aerial vehicles. Furthermore, airships have increased in popularity, especially for use in surveillance, climate monitoring, communications, and even interplanetary missions [1][2][3].

Current technologies rely on human pilots for airships, which increases cost, while decreasing effectiveness. In order to remedy this problem, autonomous airships must be developed that can operate over long periods of time with a small overhead cost. Several groups, including the University of Virginia (UVA), the University of Stuttgart, and Project AURORA are currently developing these technologies [4][5].

This paper presents the development of a semi-autonomous control system that accepts pilot commands, such as location and attitude, and carries out these commands on a flying airship. In order to attain this functionality, appropriate control algorithms for the UVA Solar Airship *Aztec* are developed. A full hardware command and telemetry system to fly on the airship is described, as well as the software necessary

[1] For more information: http://secap.seas.virginia.edu

to carry out the control of the airship. Finally, the results of this system design and possible future research are presented.

UVA Solar Airship Program

Begun in 1995, the UVA Solar Airship Program is a student run research team at the University of Virginia in Charlottesville, VA USA. The need for the high-altitude, long-term vehicles was recognized, and the team has worked towards advancing the state of Lighter-Than-Air (LTA) technology towards this goal. The program's mission statement characterizes the spirit of the team's goals.

As a team, we will develop a solar-powered airship capable of continuous, autonomous flight.

In order to realize this mission, the development has been divided into 3 phases:

Phase I – *Gain experience building an airship*
Phase II – *Demonstrate controlled flight and solar powered capabilities of an airship*
Phase III – *Complete project objectives with a fully autonomous and continuous airship*

The Phase I airship, *Dunkin*, was a deltoid pumpkinseed shape, 10 meters in length. There were three, X-configuration fins with movable control surfaces, and two stern-mounted motors. (see Figure 1)

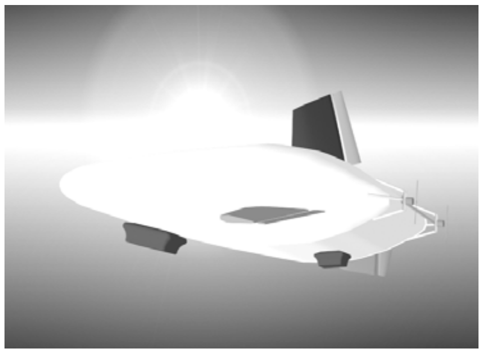


Figure 1: Phase I Airship: *Dunkin*

Dunkin required two pilots due to the fact that a simple Radio Control (RC) system was used. One pilot controlled the three fins, and the other pilot operated the two propulsion units. This proved unduly difficult for simple, controlled flight.

Furthermore, due to undersized fins the ship was unstable in yaw. Therefore, the team moved to a more traditional 'cigar' shape in order to better predict the new airship's flight characteristics.

Phase II built upon the experience gained from *Dunkin*. Better control authority was desired for the system. Therefore, *Aztec*, a 20-meter long, 5-meter wide non-rigid airship, is equipped with two side-mounted motors and a single stern-mounted motor as shown in Figure 2. The side thrusters are independently throttled and can be pitch vectored around 360°. The stern-mounted motor can yaw 90° to either side of aft.

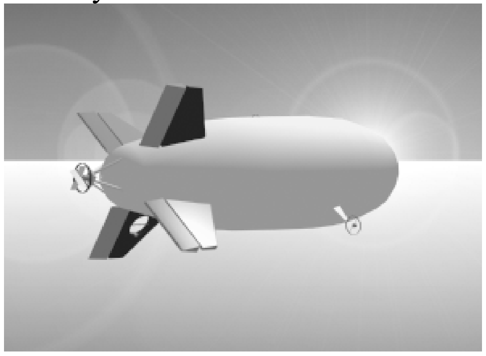


Figure 2: Phase II Airship: *Aztec*

Aztec also has 4 warpable fins, in cruciform shape. These fins are a simple carbon fiber rod, in a U-shape, with nylon fabric stretched over it. As the fin is twisted, it produces control moments. These passive control surfaces are ineffective at low velocities, so the thrusters must be used in this regime. At higher velocities, the fins are more effective.

Following the testing of *Aztec*, implementation of this control system, and successful testing of the solar array, the team will move onto Phase III, dubbed *SunJammer*. This airship will operate at approx 60,000 ft. and provide continual coverage using solar arrays along the top of the ship and batteries.

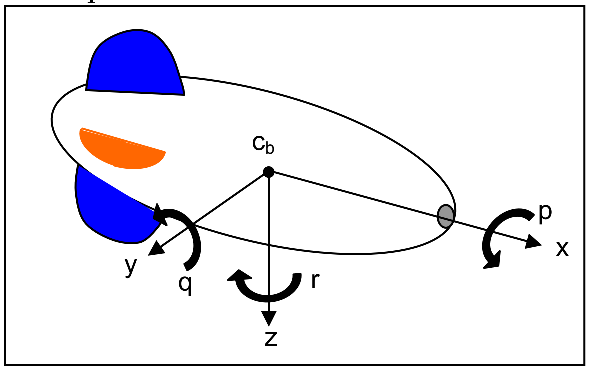


Figure 3: Airship Coordinate Axes

Control System Design

The control system is responsible for the following activities:

- Retrieve user input
- Determine flight trajectory
- Compensate for errors & disturbances
- Map control torques to actuations
- Execute actuation on the airship

Furthermore, the system will be required to maintain position and heading of the airship during maneuvers and station keeping. However, based on *Aztec*'s configuration, pitch at low velocities is not possible. Also, roll is not necessary since the center of gravity (C_g) is underneath the center of volume (C_v), and therefore, neither of these attitude angles will be actively controlled.

Each module is responsible for a certain aspect of the control system (see Figure 4). The *Motion Controller* accepts user input in the form of waypoints and calculate the airship velocities and angles over the entire time of flight. The *Compensator* is a feedback control loop that maps the velocities and angles into necessary torques. Finally, the *Actuation Controller* determines the most efficient method of employing the airship actuators to attain the required torques. The *Sensor* package will be responsible for closing the loop by giving current position, attitude and velocity to the *Motion Controller* and *Compensator*.

Motion Controller

The motion controller must determine the airship's trajectory and flight velocity. A simple way to achieve this is to use a trapezoid method of motion planning.

The user input is in the form of a position, (Latitude, Longitude, Altitude), and attitude, (Roll, Pitch, Yaw). Using the sensor, the current location and orientation of the airship is known. Therefore, determining the change in location is a simple difference.

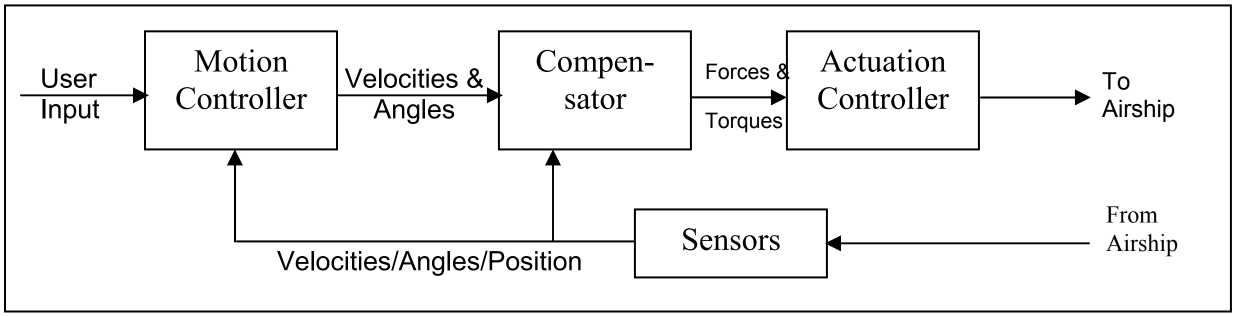


Figure 4: Control System Module Diagram

However, to simplify the trajectory, it is easier to move in a plane. Therefore, the airship will make a yaw angle change to face its destination before translating. (Figure 5)

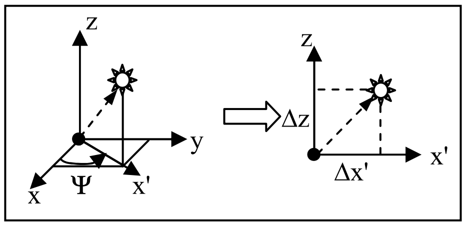


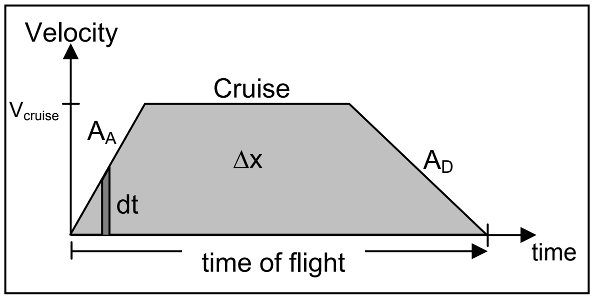
Figure 5: Initial Yaw Change

$$\Psi = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) \qquad (1)$$

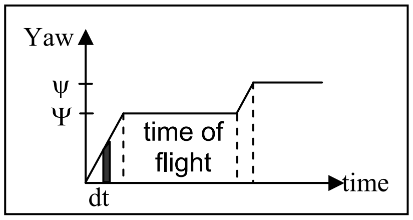
Therefore, the trajectory now only involves a change in the x' direction and the z direction. The motion controller now creates a trapezoid, which rises with a slope equal to the airship's acceleration (A_A), and then maintains a cruise velocity. The trapezoid then decreases with a slope of the airship's deceleration (A_D). The limiting factor on flight time is the distance, which is equal to the are of the trapezoid.

$$\Delta x = \int_0^{t_f} velocity(time) \cdot dt \qquad (2)$$

This creates a trapezoid shape as seen in Figure 7. The value *dt* is then what is passed to the compensator.



However, yaw changes are handled differently. This is because during the translational motion, the heading must be kept. A constant heading being passed to the compensator will correct for any error during flight. Therefore, yaw planning takes on the shape of a tiered equation, where the first tier is the initial yaw motion, meant to simplify the trajectory, and the second yaw change into the final desired heading. (See Figure 6)



Where: Ψ is the initial yaw heading
ψ is the final yaw heading

The final trajectory planning looks like the following: (Figure 8)

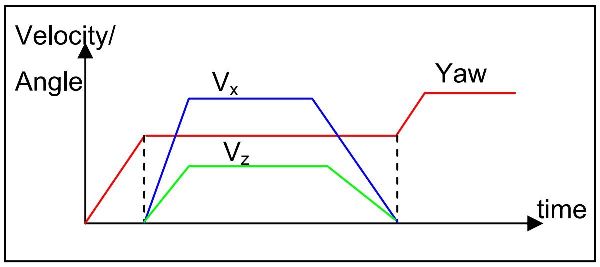


Figure 8: Motion Planning Trajectories

Compensator

The Compensator takes small errors in velocity or yaw angle, and maps these to directional torques in the X, and Z direction and moment in the M direction. A simple PID controller, as seen in Figure 9, is used for all Yaw, and a PI controller for V_x, V_z [6].

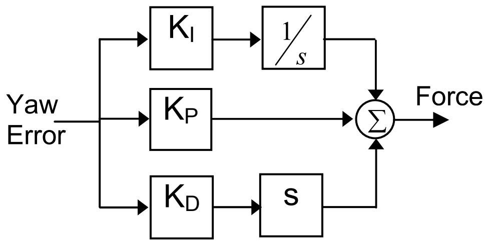


Figure 9: PID Controller

The gains K_P, K_D, and K_I in Figure 9 can be found experimentally and verified using a computer simulation based on the airship models presented in [7]. A model for *Aztec* was developed in Simulink, but was left out of this report for the purpose of brevity.

Actuation Controller

The actuation controller is a more difficult system that maps the directional forces, F_x, F_z, and T_ψ, to appropriate thruster rotation speeds and angles.

Based on a simple model of *Aztec* as seen in Figure 10, the thrusters produce simple forces on the airship at a distance from C_b as indicated on the diagram by the force arrows.

From this diagram, the propulsion actuation functions are derived as follows:

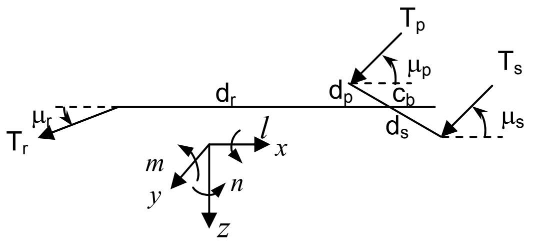


Figure 10: Wireframe Force Diagram of *Aztec*

$$
\begin{aligned}
F_x &= T_p \cdot \cos(\mu_p) + T_s \cdot \cos(\mu_s) + T_r \cdot \cos(\mu_r) \\
F_y &= -T_r \cdot \sin(\mu_r) \\
F_z &= T_p \cdot \sin(\mu_p) + T_s \cdot \sin(\mu_s) \\
L &= d_p \cdot T_p \cdot \sin(\mu_p) - d_s \cdot T_s \cdot \sin(\mu_s) \\
M &= 0 \\
N &= d_p \cdot T_p \cdot \cos(\mu_p) - d_s \cdot T_s \cdot \cos(\mu_s) \\
&\quad - d_r \cdot T_r \cdot \sin(\mu_r)
\end{aligned}
\tag{3}
$$

However, as mentioned above, it is undesirable to control the roll moment, L. It is also not useful to control lateral motion, F_y. Finally, a pitching moment, M, is not possible due to the control configuration.

Therefore, there are 3 equations, F_x, F_z, and N, but 6 unknowns. A system is devised that limits the free variables based on a desired flight mode of the airship. The following 3 constraints limit the free variables:

1) $\mu_r = \frac{1}{2} * \theta_{desired}$
   This allows the stern thruster to contribute to the desired yaw moment, and slowly straighten out as the yaw angle falls to 0.
2) $\mu_p = \mu_s$
   Since it is undesirable to actively control the roll, there would be no efficient configuration in which the side thrusters would not be pointing in the same direction.

3a) $T_r = T_s$ when $\theta < 0°$
   When the airship is turning to port, the port thruster will reduce its thrust, creating a torque moment about the y-axis.

3b) $T_r = T_p$ when $\theta > 0°$
   When the airship is turning to starboard, the starboard thruster will reduce its thrust, creating a torque moment about the y-axis.

These flight techniques were tested out in piloted flight on *Inca*, a quarter scale model of *Aztec*, prior to their implementation on *Aztec*. They allowed for easy, efficient control of the airship during maneuvers.

These constraints allow definite solutions to be found to the equations. However, the three equations are coupled and non-linear, so no closed form solution can be simply found. Therefore, this control system employs the Newton-Raphson method for finding the actuator values. This method uses an initial condition of values to calculate a possible solution point, as well as correction factors for each variable. By iterating, and adding these correction factors, a set of solutions can be found in several (3 or 4) iterations.

In order to find the most efficient actuator configurations, initial guesses must be made within close proximity to the solution.

Therefore, a map of force directions and actuator values was developed.

To summarize the actuation controller, the compensator passes two forces and a torque to the input of the actuation controller. A look-up table of initial values for the six actuators is referenced based on the values of these forces. These actuator values and forces/torque are run through several iterations of the Newton-Raphson method, where solutions are found for the six actuator values.

The output of the actuation controller is fed to the rest of the airship servo controllers, and sensor values are then read and the loop repeats.

Flight Hardware Design & Implementation

The control system relies on the flight hardware for interaction with the rest of the physical airship. The hardware consists of a ground station in which a pilot or user operates the airship by giving it flight commands and waypoints. This information must then be communicated to the airship in its flight, where the onboard hardware interprets the message, and sends out signals to the airship actuators. Furthermore, the onboard system must have a host of sensors that can sense the state of *Aztec* and relay this information to the semi-autonomous software, and the ground station.

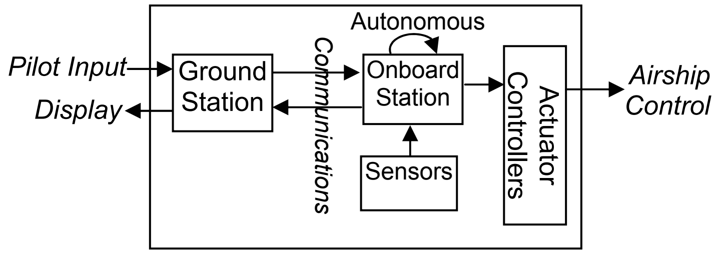


Figure 11: Hardware Module Configuration

Ground Hardware

The ground station must be a simple interface to allow a single pilot or operator to quickly and easily access any function of the airship. Furthermore, the ground station must display all necessary state information to users, and alert them to any potentially hazardous problems.

Since the ground station will move to different locations, a laptop is used. The laptop displays all sensor information to the screen, and makes pilot functions accessible on screen. Waypoints and semi-autonomous control are made available to the user, as well as maps for trajectory planning.

To control the airship's flight during ground operations and flight-testing, a joystick is connected to the laptop. Flight commands are sent to the airship by using the joystick much the same as an airplane would be flown.

Communication between the ground station and onboard system is through a pair of radio modems[2]. These components operate through the RS-232 serial port on a computer, and transmit the data at 900MHz to a range of 20 miles. The data rate is 115.2 kbps, high enough for simple communications and telemetry transmissions. Furthermore, the radio modems employ frequency hopping, which cuts down on prohibitive interference of the airship data while in flight.

Onboard Hardware

The onboard system process all ground messages, control algorithms, and control the airship actuators. The system is a highly modularized, upgradeable embedded system that requires little power, space, and heat dissipation. (See Figure 12)

Communication is carried out by a matching radio modem. This connects to a

[2] Donated by Freewave Technologies, Boulder, CO USA (www.freewave.com)

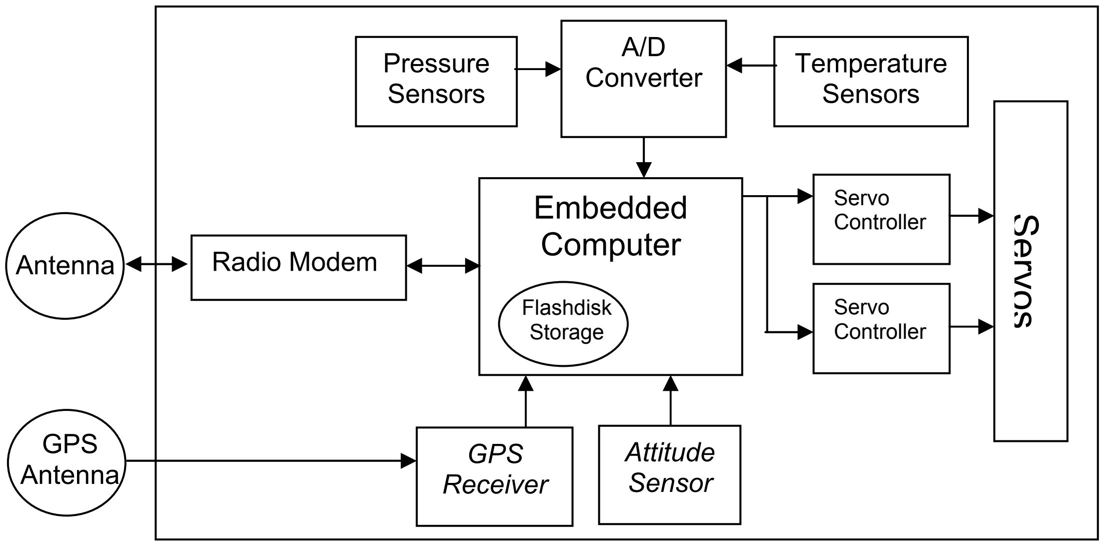


Figure 12: Onboard Hardware Configuration

serial port on the PC-104 based embedded computer[3]. The PC-104 designation[4] is an industry standard for size, and interface of various embedded components. Other modules, such as a power bus and GPS receiver, merely stack on top of the computer, and communicate through the shared bus.

The embedded computer has 6 serial ports, 2 USB ports, an Ethernet port, a PCI bus, a PC-104 bus, and a CompactFlash slot. The system is a Pentium 180 with 32 MB of RAM, and 2MB of FlashROM. This Flash memory holds the Operating System (OS) for the computer. The CompactFlash slot holds a 40MB flash card that stores all of the program information, as well as the data logs.

The computer, being at the heart of the onboard system, also retrieves data from the attitude sensor[5] that measures the airships roll, pitch and yaw. The GPS receiver gives the airship position on the earth with an accuracy of ±10 meters. This location information can be extrapolated for velocity measurements as needed by the *Compensator*. An Analog/Digital Converter polls the voltage varying pressure and temperature sensors that measure the envelope, ballonet, and ambient for airship state information.

The control hardware uses 2 Servo Controllers[6] to operate the airship's actuators. These boards use a PIC chip to convert an 8-bit digital signal from the computer's RS-232 serial port, to a 1-bit Pulse-Width-Modulation (PWM) signal that the airship servos use to change position.

Together, these components provide all of the necessary telemetry and control over the airship that is necessary for controlled, autonomous flight. Furthermore, the system has been built to allow for expansion, such as a high-speed wireless Ethernet, and onboard payload packages, such as weather and climate monitoring equipment, or surveillance cameras.

Flight Software Design & Implementation

The flight software system must properly control all of the hardware components, as well as carry out the control algorithms presented above. The system will also need to be modular and easy to maintain as hardware components and flight goals are changed. For this reason, C++ is used, with

[3] Donated by ZF Microsystems (www.zfmicro.com)
[4] PC-104 Consortium (www.pc104.com)
[5] Donated by Precision Navigation (www.precisionnav.com)
[6] Donated by Pontech (www.pontech.com)

an Object-Oriented Programming (OOP) design architecture.

Each of the blocks in Figure 13 represents a software module, and usually a C++ class.

Operating System

In order to run multi-threaded software with reduced hardware overhead, Linux was chosen as the operating system. Kernels can be stripped down to less than 2 MB of storage space and run with only 4MB of RAM. In addition, the RTLinux module can be loaded into the Kernel to add real-time support. The kernel will reside in the flashdisk storage of the onboard computer. When the OS is booted, a ground user can Telnet into the computer to check system status, upload new software or download flight logs using FTP (File-Transfer-Protocol), and start the software running.

Ground Software

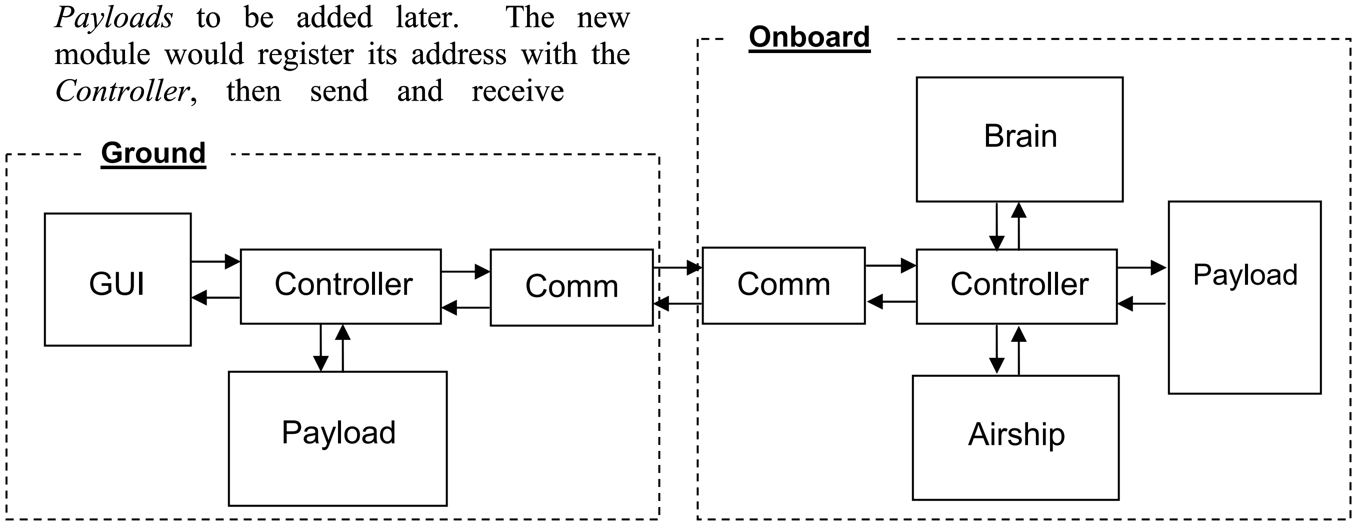
For the ground system, the *GUI* (Graphical User Interface) is responsible for retrieving all user input, as well as displaying the airship sensor information. The *Controller* acts as a router. Messages that it receives are passed onto the next appropriate based on the destination address of the message. This allows external *Payloads* to be added later. The new module would register its address with the *Controller*, then send and receive messages as necessary. The *Comm*, or communications, module is responsible for interfacing with the communications hardware, as well as package the messages in network packets and handle communication errors.

Onboard Software

The onboard software employs the same logic as the ground software. The *Comm* module handles the radio modem hardware interface, and network packets and error checking. It then passes on the message to the *Controller*, which routes the data to the destination module.

The *Airship* module handles all hardware interfaces to the actuators and sensors. This allows for an easy external interface to the system without worrying about the specifics of the airship hardware. Furthermore, the *Airship* module also handles basic motor mixing. This includes interpreting pilot flight commands and mapping them to appropriate actuator commands.

The *Brain* module is where the semi-autonomous software resides. Messages will contain mode and waypoint changes. The *Brain*, after running its control algorithms, will send actuator commands to the *Airship* through the *Controller*. The onboard system runs on two threads, one



thread operating the Brain control algorithms and the other thread handles all of the airship flight activities. A hardware clock is used to ensure regular calculation of the control algorithms without locking system resources.

Brain Module

The control algorithms developed above are calculated in the brain module. Due to the original design of the control system components, the software module system looks very similar (See Figure 14).

The *Motion Planner* retrieves a waypoint from the storage of waypoint locations, creates a trapezoid trajectory map, and sends out the velocity command for each time step. The *Route Line* obtains this information and passes it, as well as the current sensor values to the *PID Controller* module where the PID code calculates the airship forces and moments. Finally, these values are sent by the *Route Line* to the *Actuation Controller*, which uses the lookup tables and Newton-Raphson code to calculate actuator positions. These are sent to a message queue with a destination of the *Airship* module.

Also, the Motion Planner will need to update its motion planning at a frequency much less than that of the control loop frequency due to disturbances in the system such as drift, cross-wind, etc. Future systems could add a Force disturbance module between the *PID Controller* and the *Actuation Controller* to account for the known forces on the airship during flight.

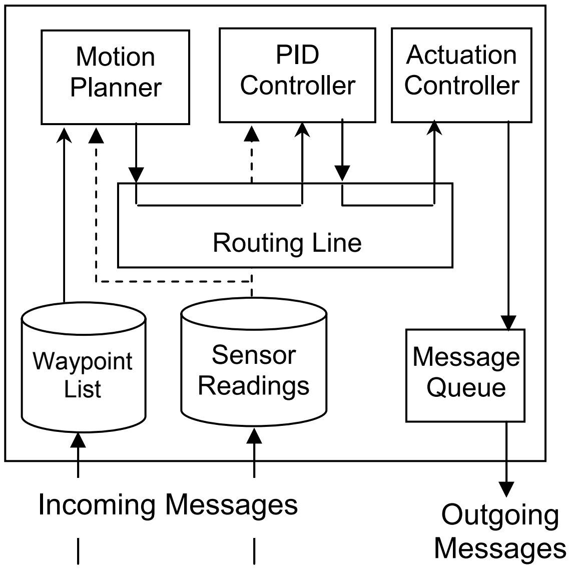


Figure 14: Brain Software Architecture

Flight Testing

The ship is currently being flight tested at our hangar in Charlottesville, VA USA. A ground crew of six line handlers coordinates taxi maneuvers, take-offs, and landings. A Flight Manager is responsible for maintaining the flight protocol as well as directing the ground crew. A single pilot operates the ground control station, with a support member ready to assist the pilot, record all data measurements and debug the system as necessary.

Ground tests, in addition to small-scale model flight tests have demonstrated excellent results with this control system. *Aztec*'s system will be flown through 2000 to test various control configurations and algorithms. An autonomous control system is being developed that will also be tested using this semi-autonomous system as the backbone.

Summary

A semi-autonomous control system for the UVA Solar Airship *Aztec* has been fully developed. The system, when implemented together allows for easy, stable flight of an airship with a human pilot, or through pre-planned waypoint motion. Furthermore, the system allows for easy upgrades and maintenance by later team members. The demonstration of a semi-autonomous airship has large implications on the future of airship design and operations.

Acknowledgement

The author gratefully thanks the UVA Solar Airship Team for all of its hard work in designing, building, flying its airships and for continuing to develop airship technology. Furthermore, many thanks go to Dr. Carl Knospe and Dr. Pradip Sheth for their assistance with difficult problems. And finally, this research would not have been possible without the funding provided by the Raven Society at the University of Virginia.

References

[1] Miller, John A. "Airship platform for high-resolution space technology telescope." *Proc. of SPIE - The International Society for Optical Engineering.* v 2478 1995. Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, 1995, pp.76-88.

[2] Sullivan, Arthur. Turner, William C. "Eye in the sky: airship surveillance." *International Telemetering Conference (Procs)*. v 31 1995. Instrument Society of America, Research Triangle Park, NC, 1995, pp.165-174.

[3] Friedlander, Alan L. "Buoyant Station Mission Concepts for Titan Exploration." Acta Astronautica. v 14 1986, pp.233-242.

[4] Kaempf, B.G., and K.H. Well. "Attitude Control System for a Remotely-Controlled Airship." AIAA, Paper No. 95-1622-CP, 1995, pp.88-98.

[5] Elfes, Alberto. Bueno, Samuel Siqueira. Bergerman, Marcel. Guimaraes Ramos, Josue Jr. "Semi-autonomous robotic airship for environmental monitoring missions" *Proc. of the 1998 IEEE International Conference on Robotics and Automation,* Piscataway, NJ, Vol. 4, 1998, pp.3449-3455.

[6] de Paiva, Ely C., Samuel S. Bueno, and Marcel Bergerman. "A Robust Pitch Attitude Controller for Aurora's Semi-Autonomous Robotic Airship." AIAA, Paper No. 99-3907, 1999, pp.141-8.

[7] Gomes, S. B. V., and Ramos, J. J. G. "Airship Dynamic Modeling for Autonomous Operation." *Proc. of the 1998 IEEE International Conference on Robotics & Automation,* Leuven, Belgium, May 1998, pp.3462-7.

[8] de Paiva, Ely C., Samuel S. Bueno, Sergio B.V. Gomes, et al. "A Control System Development Environment for AURORA's Semi-Autonomous Robotic Airship." *Proc of the 1999 IEEE International Conference on Robotics & Automation*, Detroit, MI, May 1999, pp.2328-35.

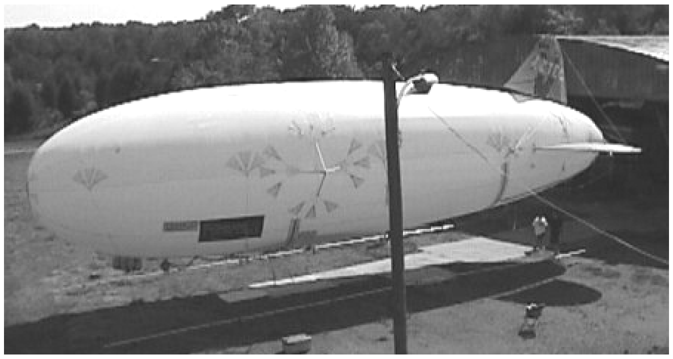


Figure 15: UVA Solar Airship *Aztec* outside hangar in Charlottesville, VA USA