Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2009
August 30 - September 2, 2009, San Diego, California, USA

DETC2009-86547

LOW COST RAPIDLY RECONFIGURABLE UAV AUTOPILOT FOR RESEARCH AND DEVELOPMENT OF GUIDANCE, NAVIGATION AND CONTROL ALGORITHMS

M. I. Lizarraga*, **G. H. Elkaim, G. M. Horn & R. Curry**
Autonomous Systems Laboratory
Jack Baskin School of Engineering
University of California
Santa Cruz, California 95064
Email: malife@soe.ucsc.edu

V. Dobrokhodov & I. Kaminer

Unmanned Systems Laboratory
Dept. of Mech. & Astronautical Eng.
Naval Postgraduate School
Monterey, California, 93940

ABSTRACT

This paper presents the development and preliminary results of a rapidly reconfigurable autopilot for small Unmanned Aerial Vehicles. The autopilot presented differs from current commercial and open source autopilots mainly as it has been designed to: **(i)** be easily reprogrammable via Simulink (models are directly transferred to the autopilot through the Real-Time Workshop's code-generation capability); **(ii)** decouple the traditional tasks of attitude estimation/navigation and flight control by using two Digital Signal Controllers (one for each task) interconnected via a Serial Peripheral Interface; and, **(iii)** being able to interact directly with Simulink as a Hardware-in-the-Loop simulator. This work details each of the main components of the autopilot and its ground control station software. Preliminary results for sensor calibration, Hardware-in-the-loop, ground and flight tests are presented.

1 Introduction

Unmanned Aerial Vehicles (UAVs) have seen a remarked increase in usage in recent years, partially fueled by the dramatic increase in performance to cost/size/power ratios of microelectronics that make up the core avionics. Additionally, the appearance of 16-bit and 32-bit microcontroller units (MCUs) and digital signal controllers (DSCs) have made it possible to put sophisticated onboard processing within the power and payload budgets. This new generation of MCUs and DSCs are not only more computationally efficient, but also consume less power, making them ideal for battery operated systems such as small UAVs. Another key factor in the growth of UAV usage has been the availability of off-the-shelf components that are easily integrated into the avionics. These components have allowed many universities to establish well-equipped UAV laboratories [1] to conduct research in diverse topics in image processing and controls.

Although UAVs were once thought to be strictly military tools, they have become formidable tools in forest fire monitoring, border surveillance, whale and other marine mammal tracking, power-line verification, and search and rescue missions in disaster areas [2]. These and many other applications have gained the interest of researchers to employ UAVs as research platforms. Currently UAVs are being actively used to conduct research in fault-tolerance [3] [4], tracking [5] [6], path following [7] [8], and cooperative control [9] [10] to name but a few.

1.1 Evolution of Autopilot Technology

The history of UAV autopilot development is tightly linked to the historical evolution of aviation. Although controversial in its first appearance, the first functionally complete autopilot design of a manned aircraft was demonstrated in 1914 by Lawrence Sperry with a gyroscopic stabilizer. The first demonstration caused a great deal of excitement, with headlines such as: "Long-

*Address all correspondence to this author.

Copyright © 2009 by ASME

sought aeroplane stabilizer invented at last; Lawrence B. Sperry performs seemingly impossible feats by aid of his father's gyroscope device" [11]. As the first "autopilot" functionality, it was only capable of straight-and-level flight and correcting bearings, but this already significantly reduced the pilot workload [12]. Advancements in airplane technology and onboard instrumentation were driven primarily by military needs. This resulted in an *explosion* of autopilot technologies; by World War II most of the industrialized powers had built aircraft capable of simple autonomous flight and attack against heavily guarded targets [12]. Designed primarily for precision bombing, the autopilot held bombers on the exact course determined by the navigator. The autopilot during the course of the bombing-run had complete control over the aircraft and attained a precision beyond human capability. This precisely guided airplane provided an accuracy of ballistic delivery from high enough altitudes to avoid ground fire. Although first built to assist humans, the autopilots quickly succeeded not only in advancing navigation, but in eliminating the pilot altogether. By removing the pilot from potential harm, the autopilots evolved into a new class of fully autonomous weapons [13].

Inertial navigation expertise [14] was developed over the second part of 20th century, and led to the realization that **(i)** the airplane state's information degrades with time regardless of motion; **(ii)** the equipment is bulky and expensive; **(iii)** initial alignment is necessary; and **(iv)** the accuracy of navigation information depends on the vehicle maneuver. As an improvement to complement the inertial sensors, a military experimental program in the 1970s led to the development of the Global Positioning System (GPS). By the early 1990s, GPS was fully deployed and became globally operational (and now has widespread civilian use). This blend of advances in inertial navigation and GPS technology accompanied with the revolutionary progress in microelectronics, materials, and power is what makes the miniature autopilot designs feasible today. In the most general sense the autopilot is used to stabilize the UAV and responds to ground station *mid-level* commands such as turn-rate or pitch, or *high-level* commands such as waypoint tracking.

There exists a wide variety of commercial miniature autopilots. Piccolo [15], MicroPilot [16] and Kestrel [17] are the most visible in the US market. However, each day brings a new design to the market. In spite of this, practically all the autpilots share the same functionality of waypoint navigation and stabilization inherited from manned aircraft operation. Indeed, even though there are no limitations from a human onboard and in spite of the advances in Guidance Navigation & Control (GNC) technologies the vast bulk of the autopilots implement simple coordinated control of a UAV. Instrumented with a set of sensors all of them deliver a complete ready-to-use (but prohibitive to modify) package. On the other hand there are many Open Source options replicating many if not all of the features available in their commercial counterparts. The Paparazzi autopilot [18] which uses a set of thermopyles (infrared sensors) for attitude determination has shown to be a very viable alternative to its commercial counterparts, without using a full Inertial Measurement Unit (IMU).

1.2 Motivation for an R&D Autopilot

While the systems available today (commercial and open-source) are very capable and provide a useful platform for UAV flights, they are all difficult to modify. This is one of the driving forces behind this autopilot design. Research and Development (R&D) labs seeking to advance the state of the art require an autopilot that is easy to modify to suit their research objectives. Currently, these labs face two main options: **(i)** Augment the fixed set of autopilot commands available to the end user with their proposed control system [19] or **(ii)** invest a considerable amount of time and resources in modifying the autopilot's source code, hardware, or both (if available) to suit their needs.

Though option **(i)** has been shown to work in some cases, it is not ideal since it forces the control engineer to design any algorithm as an *outer* (and thus slower) loop instead of an integral part of the autopilot control system; and there are certain cases, such as multiple failure tolerance, where outer loop *augmentation* is not feasible. Option **(ii)** is more promising in the long run, but the learning curve is steep and requires in-house expertise in either hardware design (if extending the hardware) or in a high level C or C++ programming (if modifying the firmware). Further, even if these changes are possible one is then tied to the existing hardware.

To overcome these limitations this work presents an autopilot that has been designed from the beginning to be easily reprogrammable using MATLAB Simulink's Real Time Workshop code generation tool. By harnessing the power of automatic code generation, anything from a simple change in controller architecture to a complete redesign can be achieved without the need of directly writing source code. It is widely believed that automatically generated code is neither as reliable nor as efficient as handwritten code, however, these beliefs are outdated. Automotive companies have used code generation to reduce development and deployment time by up to 75% [20] [21]. In 1998 Toyota presented results at the Global Automotive Engineering Seminar showing that code generated by the Real-Time Workshop is only 5% larger and 15% slower than handwritten code. And there are reports of code generation being employed in safety-critical systems such as industrial PLCs [22], navigation and flight control systems [23] and experimental hypersonic scramjets [24].

Although the Real Time Workshop is able to generate generic C code, specific features of a given processor are only available through what in Simulink lingo is called an Embedded Target (ET). This ET makes available, via a Simulink library, a set of blocks to access the hardware peripherals. It also facilitates the compiling and linking process by providing hooks to the compiler from within Simulink. One can create

Copyright © 2009 by ASME

the Simulink model, generate code, and compile directly within Simulink. There are several commercial alternatives that target different MCUs and DSCs. The ET employed by the autopilot presented here is that of Ref. [25].

Another important resource for any autopilot research is a Hardware-In-the-Loop (HIL) simulator [26]. This software allows the end user to conduct overall system check, training, and, in the R&D case, algorithm verification and validation without the need to fly. Although the simulators shipped with currently available autopilots work well in general, they are quite limited once one tries to simulate complicated phenomena or use a different aircraft model. For example, simulating a component failure, a combination of failures or employing a different plant model from the one pre-programmed is difficult (and often not possible). To overcome this, the autopilot presented in this work uses Simulink as an HIL simulator. In general, a Simulink model is used to send synthetic sensor data to the autopilot and receive back control commands. This setup offers flexibility not previously available and, provided that a plant model obtained by system identification is accurate, reduces the amount of parameter tuning when going from HIL to real flight implementation.

Lastly, most UAV autopilots available today have somewhat similar architecture: on the firmware side they provide stability augmentation, and waypoint navigation, as previously discussed; on the hardware side, most of them use the classical sensor suite: Inertial Measurement Unit (IMU), GPS, pressure sensors, and a radio downlink for a ground station. The architecture implemented in the autopilot presented herein differs from existing architectures by dividing the regular tasks performed by the autopilot among two processing units. This provides a higher processing power that allows for implementation of complex control algorithms. Although the use of two processing units in an autopilot has been done before [27] the architecture presented here differs by the way the autopilot tasks are assigned. By having two processing units, one for attitude estimation/navigation and the other for control, researchers can focus on either one of these topics while leaving the other unchanged eliminating the risk of mutual disruption.

The rest of this paper is organized as follows: Section 2 presents in detail the main components of the developed autopilot, namely, the hardware design, software architecture, communications protocol, ground control station and HIL simulator. Section 3 presents preliminary results obtained from the sensor calibration, ground and flight tests. Finally, Section 4 concludes the work and points the next steps in the development of the autopilot.

2 Autopilot Design

Although it is tacitly assumed that the design and development of an autopilot is a solved problem, to the authors' knowledge, there is very little written in the literature of how to go about it from an applied point of view. Much has been written about the individual components but it is an undeniable fact that integrating those components in an organized and workable fashion requires a significant systems engineering effort. This integration requires interdisciplinary knowledge of diverse topics such as aeronautical, electrical, computer, software, and mechanical engineering. Due to this interdisciplinary complexity, algorithm design and the consequent embedded implementation is very error prone. By using automatic code generation this newly designed autopilot enables one to advance in the objective of the UAV based research by streamlining the onboard algorithm design and its embedded-code transition. The following subsections describe this architecture and details how each of the components interact. Where applicable, we provide the driving factors that were taken into account during the design phase of each subsystem.

2.1 Hardware Design

The goal was to develop a complete architecture for an autopilot for small UAVs that had enough processing power for moderately complicated control tasks and at the same time was easily and rapidly reprogrammable using advanced capabilities of Simulink. The proposed computational architecture physically decouples sensor integration and attitude estimation/navigation from the control algorithm and communications by using two dsPIC33 DSCs [28] interconnected via a high-speed Serial Peripheral Interface (SPI) bus. The autopilot has been designed to be modular and extendable in order to accommodate a rich sensor and peripheral suite as the need arises.

The main subsystems and their principal tasks are (see Figure 1):

1. A Sensor DSC used to read sensor data, and perform attitude estimation/navigation.
2. A Control DSC used to, based on the computed attitude, generate control signals to the control surfaces' actuators and report all the telemetry to the ground control station.
3. A Complete three-axis IMU: 3 MEMS Accelerometers, 3 MEMS angular rate sensors, and 3 magnetometers.
4. Differential and absolute pressure sensors.
5. A thermistor for temperature compensation.
6. Controller Area Network (CAN) connectivity for optional daughterboards.
7. A 5 Hz. update-rate serial GPS.
8. A serial port to read incoming data from the HIL Simulator (see Section 2.5).
9. A multiplexor used to selectively send commands to the actuators either from the autopilot itself or from an external radio control (RC) receiver.
10. Ten Pulse Width Modulation (PWM) output channels for RC servo actuators.

Copyright © 2009 by ASME

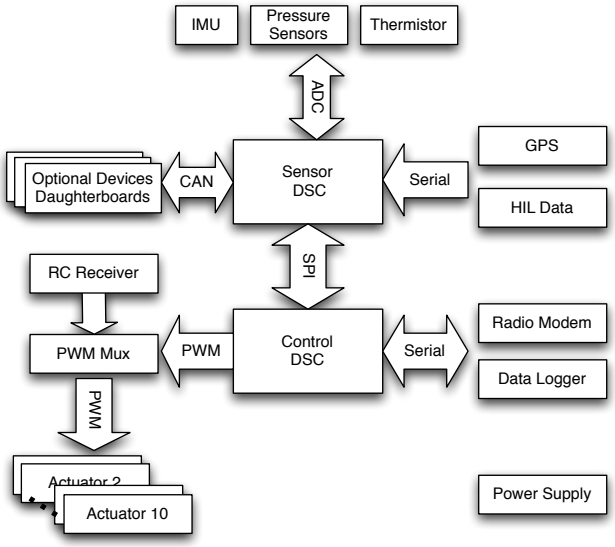

Figure 1: Autopilot Block Diagram

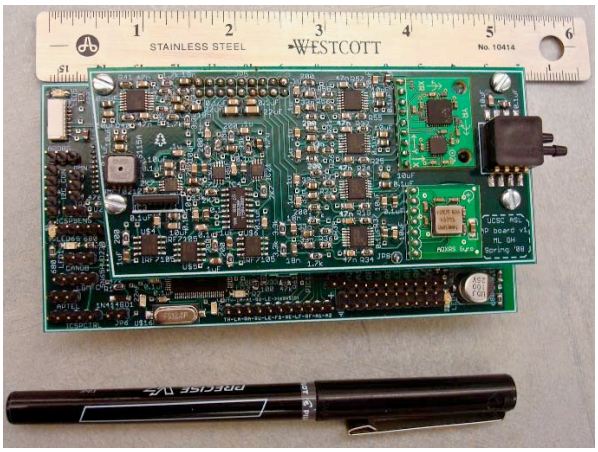

Figure 2: Autopilot Hardware

11. Communications protocol input/output via serial to a radio modem.
12. Data logging to an internal SD or MMC card.
13. Power supply for all the components on board.

Because of the space constraint of fitting into an RC airplane, the Printed Circuit Board (PCB) was split into a digital motherboard and an analog daughterboard (see Figure 2). The analog daughterboard consists of the sensor suite and an array of analog filters. Each sensor signal is filtered through a second-order analog Bessel filter implemented with a Sallen-Key topology [29].

Low-noise mixed-signal techniques were rigorously utilized to minimize sensor noise. The PCB was designed with four copper layers, with dedicated ground and power planes. Analog and digital power were supplied by separate fixed linear regulators, and the analog regulator was given an additional capacitor-inductor-capacitor (CLC) Π filter [29] to suppress high frequency digital switching transients. Careful attention was paid to the power plane layout with respect to current return paths and avoiding ground loops. A single-point tie in the ground plane was utilized to isolate the noisy servo power from the rest of the board. Every analog and power IC was bypassed with parallel 10μF/0.1μF ceramic chip capacitors. Power ICs received additional large electrolytic capacitors. Digital ICs were bypassed with a 0.1μF ceramic chip capacitors, with parallel larger ceramic chip capacitors when warranted. As much as possible, digital power and signal traces were not routed close to analog traces and the Analog to Digital Converter (ADC) section of the sensor DSC.

2.2 Firmware Architecture

One of the driving design specifications was to implement a library of data sources and sinks that interacted with the hardware and produced ready to use data. In that way, for instance, reading, parsing, and decoding the GPS data stream, required nothing more than dragging a source from the library. The output of this block contains all the available GPS data to be used throughout the Simulink model.

Since each of the DSCs has different tasks the firmware architecture is slightly different. Each of these is described as follows:

2.2.1 Sensor DSC Contains two main sources which are mutually exclusive (see Figure 3). It can either receive data from all the sensor suite onboard or from the HIL simulator (see Section 2.5). This data goes through a calibration and temperature compensation processes. At this point all the data from the sensors is in meaningful units and is ready to be used by the attitude and position estimation block. The estimated attitude and position (and sensor biases if available) are packed in the communications protocol (see Section 2.3) and sent to the control DSC via SPI.

2.2.2 Control DSC Also contains two main sources (see Figure 4). It receives sensor data, computed attitude and position from the sensor DSC; and control commands and configuration data from the ground control station (see Section 2.4). Both of these sources turn the data to a protocol decoder to convert the data streams into usable data by the rest of the blocks. With the data in the right format, the control block computes the control commands to each of the control surfaces based on the commands received from the ground control station. Control

Copyright © 2009 by ASME

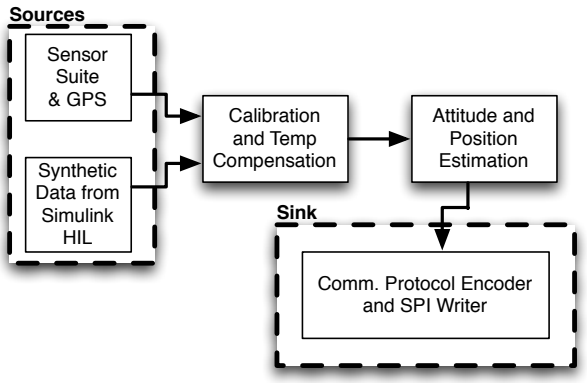

Figure 3: Sensor DSC Firmware Architecture

surface commands as well as all the data received from the sensor DSC is encoded in the communications protocol and sent to the ground control station for display and logging purposes.

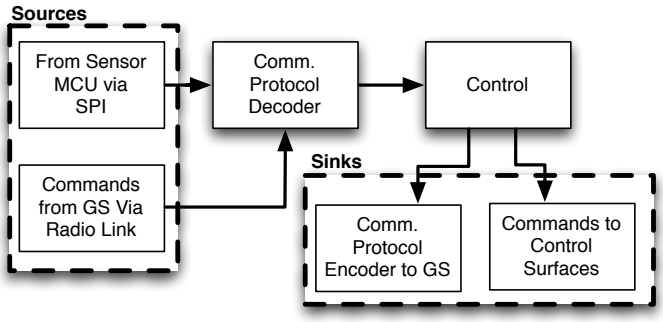

Figure 4: Control DSC Firmware Architecture

2.3 Communications Protocol

The complete autopilot system generates about 30 different meaningful variables. Attitude, position, sensor biases, temperature, absolute and differential pressure, control surface commands, and many more are generated on every cycle. These values are not only used by the autopilot but also sent to the ground control station and logged to facilitate debugging and post-flight analysis. To do so, a communications protocol assembles the data into coherent sets and calculates a checksum which is appended to the message. This becomes even more relevant when the communication is asynchronous, such as in the case of the sensor DSC to control DSC inter-processor communication.

The autopilot presented here implements a very simple, yet effective binary communications protocol where each message contains a four-byte header and a three byte trailer (see Figure 5). The header consists of two begin-of-message indicators ($@); a message identifier, which specifies the type of message, and the message length. The trailer contains two end-of-message indicators (*@) and the XOR checksum, which is computed from the message identifier to the last end-of-message byte.

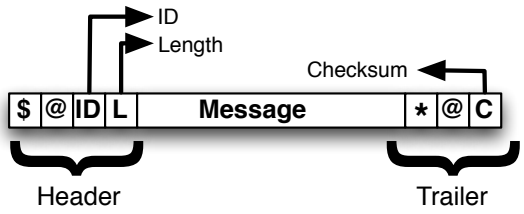

Figure 5: Communications Protocol Message Structure

Currently there are a total of 28 different messages used to pass information between the sensor DSC, the control DSC and the ground station.

2.4 Ground Control Station

The ground control station is an application that runs on any PC (see Figure 11 in Section 3.3) and reads the communications protocol stream coming from the autopilot over a serial port. The purpose of the ground control station is to provide the operator with an interface to:

Display Data. Receive and display information from the sensors and the onboard calculations such as: attitude, position, control surface commands, pressure, etc.
Display the Trajectory. Send the received position to Google Earth for display of the UAV's trajectory.
Command the UAV. Send either low-level commands such as turn-rate, or high-level commands such as way-point tracking.
Tune the Gains. Modify the gains for the various control loops on board.
Configure Way-points. Capture and update the waypoints directly from Google Earth and upload them to the autopilot.
Configure the HIL Simulator. Configuration of the simulation PC's IP address and UDP ports (see Section 2.5).
Log Data. As the incoming data stream is decoded for display purposes a Matlab-ready log file for post-flight analysis is generated.

Copyright © 2009 by ASME

2.5 Hardware-in-the-loop Simulator

As previously discussed, in a general purpose autopilot, the HIL simulator is generally used for operator training and overall system check. In the autopilot presented here the HIL simulator can be used for those purposes, but is more flexible than that. Since Simulink is *"an environment for multidomain simulation and Model-Based Design for dynamic systems"* [30] it is an ideal candidate for a simulation engine. By using Simulink as the simulation engine, the user is free to modify or analyze any aspect of it, such as sensor noise, environment, plant model, engine model, etc. and do so in a more intuitive way by the use of blocks instead of source code editing. This also makes it easy to introduce failures or check the autopilot's performance in extreme conditions.

The HIL simulator has three main components (see Figure 6):

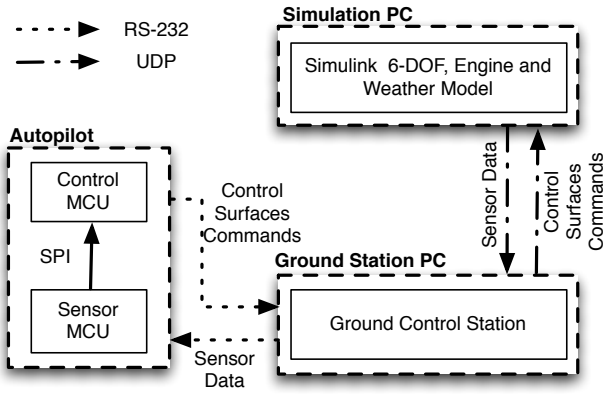

Figure 6: HIL Simulator Architecture

Simulation PC. It runs, inside Simulink, a six-degree-of-freedom (6-DOF) model of the UAV, an engine and a weather model. The 6-DOF model responds according to the control surface commands received from the autopilot and in turn generates synthetic sensor data which is sent to the ground control station.

Ground Station PC. It runs the ground control station and is responsible for routing the synthetic sensor data to the autopilot and the control surface commands to the 6-DOF model. This PC may or may not be the same as the simulation PC.

Autopilot. The autopilot, once put in HIL mode by the ground control station, it switches the source of sensor data from the onboard sensor suite to synthetic data received from ground station. The autopilot operates exactly as if it was on a real flight.

Note in Figure 6 that the Simulink model and the ground control station communicate via User Datagram Protocol (UDP) packets. This allows for a flexible setup where both can run on the same PC, or if more processing is required, they can be on two separate computers on the same network. Also, even though sensor data is being sent to the autopilot, true attitude and position is always known via the 6-DOF model and can be compared to those computed by the autopilot to assess its performance.

3 Results

3.1 Sensor Calibration

Being real physical devices that convert a physical phenomena to electrical signals, the sensors require calibration to remove their temperature dependance as well as correct for scale factor and bias errors. Note that bias drift can still occur and is removed during the attitude estimation process.

In order to calibrate the temperature variation of the sensors, the autopilot was first chilled and then brought up to high temperature while monitoring the sensors. A linear fit was used to correct the sensors. As expected, some sensors showed a very clear dependence on temperature, while others showed little to no dependence. Figure 7 shows the result of a full temperature run on the autopilot. The first row shows the reading (in counts) of the temperature sensor. Rows two and three show the raw readings and the computed values after temperature compensation for the Y-axis gyro and the static pressure sensor.

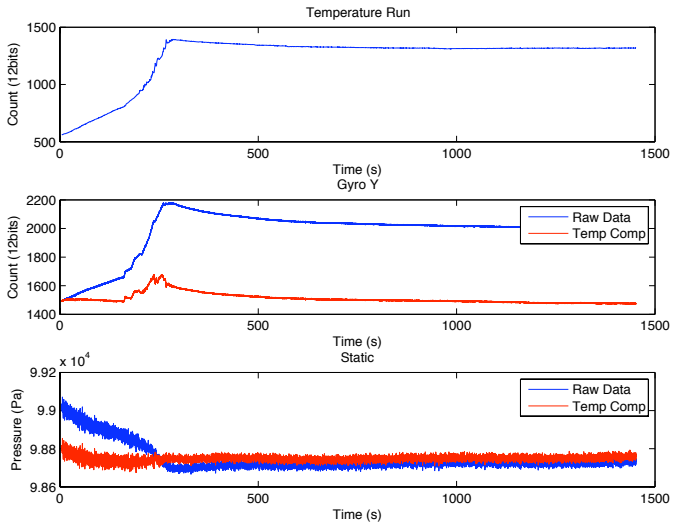

Figure 7: Gyro Y and Static Pressure Pre- and Post- Temperature Compensation

For the gyro calibration, the autopilot was placed on a rate table along with a KVH DSP-3000 high quality Fiber Optic Gyro

Copyright © 2009 by ASME

(FOG) [31] that was assumed producing true data. A series of rotations on the table were used while recording the data from both the FOG and the autopilot (see Figure 8). By matching the data, a bias and scale factor was obtained for each axis.

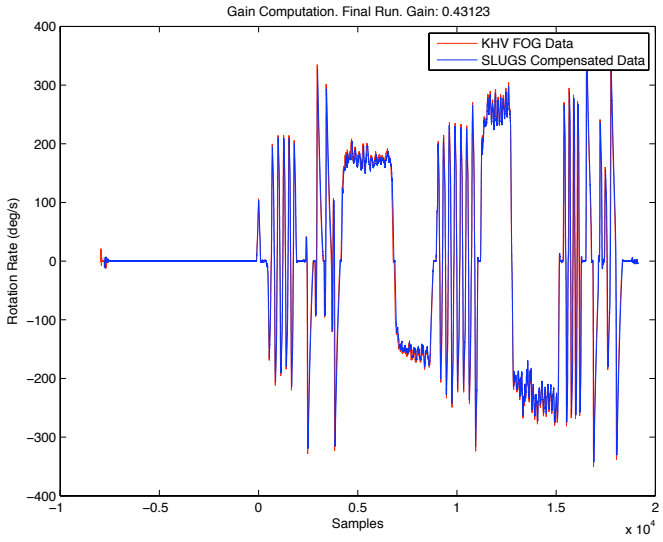

Figure 8: Gyro X Resulting Calibration Compared to the Results of the KVH Fiber Optic Gyro

Lastly, in order to calibrate the three-axis magnetometers and accelerometers we use the non-linear two-step method as described in [32]. This method calculates the scale factors, biases, and non-orthogonality factors for all three axes by tumbling the sensor triad. Figure 9 shows the pre- and post-calibration sphere generated by plotting each axis on a 3-D plot.

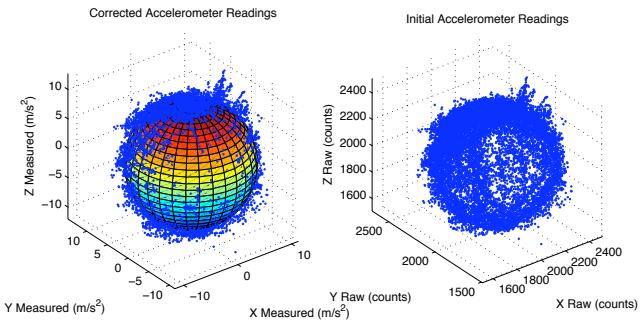

Figure 9: Accelerometer Raw Data and Resulting Calibration

3.2 Attitude Estimation

The attitude estimation currently implemented in the autopilot is a Kalman filter which estimates the gyro biases $\vec{\omega}_b$ and the rotation error between the expected quaternion $q_{(-)}$ and the true attitude quaternion q. The rotation error is expressed as the error quaternion q_e [33] given by:

$$q_e = [1 \; \varepsilon_1 \; \varepsilon_2 \; \varepsilon_3]^T, \tag{1}$$

where

$$q \equiv q_{(-)} \otimes q_e. \tag{2}$$

The time derivative of the attitude quaternion $\dot{q}$ is a function of the quaternion q and the body rotation rates $\vec{\omega}$ as described in [34]. The time derivative of the attitude quaternion is integrated to compute the expected attitude quaternion $q_{(-)}$ using $\vec{\omega} = \vec{\omega}_{meas} - \vec{\omega}_b$, where $\vec{\omega}_{meas}$ is the angular rate sensor measurement of the body rotation.

The state $[\varepsilon_1 \;\; \varepsilon_2 \;\; \varepsilon_3 \;\; \omega_{b1} \;\; \omega_{b2} \;\; \omega_{b3}]^T$ evolves according to

$$\vec{\varepsilon}_{k+1} = \vec{\varepsilon}_k + \frac{\Delta t}{2}(\vec{\omega}_{meas_k} - \vec{\omega}_{b_k}) \tag{3}$$

$$\vec{\omega}_{b_k+1} = \vec{\omega}_{b_k}, \tag{4}$$

so the linearized state transition matrix is

$$\Phi = \begin{pmatrix} I^{3\times3} & -\frac{\Delta t}{2} I^{3\times3} \\ 0^{3\times3} & I^{3\times3} \end{pmatrix}. \tag{5}$$

The true attitude quaternion measurement q_m is calculated by measuring the magnetic field and acceleration in the body frame from the onboard sensors. The translational acceleration is calculated by differencing GPS coordinates and subtracted from the measured acceleration. From these measurement vectors, the measurement quaternion is computed using vector-matching [33].

By assuming that q_m is directly measured, the Kalman filter sensor measurement matrix H is noticeably simplified to $H = [I^{3\times3} \;\; 0^{3\times3}]$.

Equation (5), H, q, and q_m from [33] are used to formulate a standard Kalman Filter, yielding the best estimates of q_e and $\vec{\omega}_b$. The propagated attitude quaternion q is updated using (2), and the next time step the process starts over using the updated $\vec{\omega}_b$ for the integration.

This algorithm has been shown to work well in simulations, numerous ground tests, and preliminary flight data . Post-flight analysis of the collected data during the initial flight tests is currently underway to tune the filter to improved its performance as

Copyright © 2009 by ASME

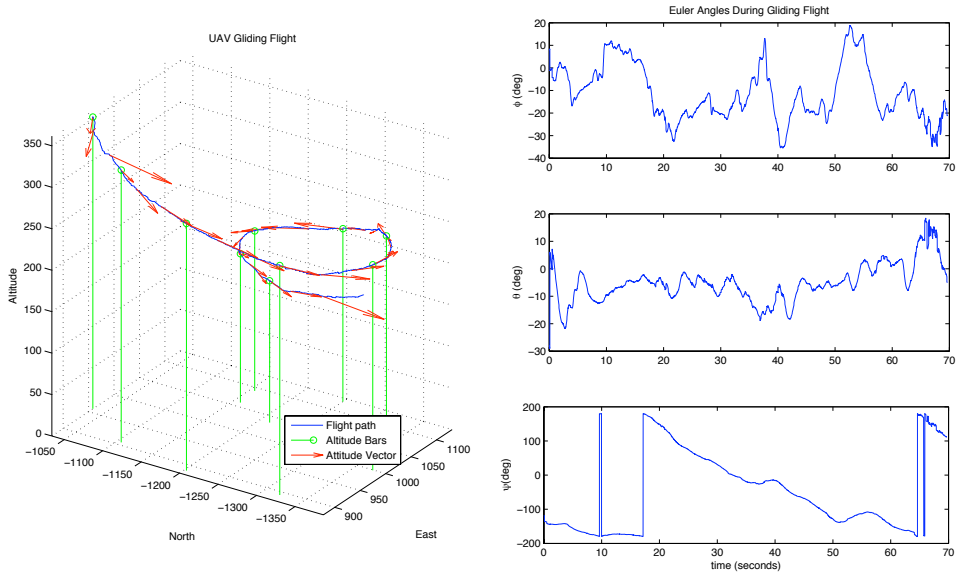

Figure 10: Attitude Estimation Results for Gliding Flight

it was severely affected by the vibrations produced by the UAV motor. Figure 10 shows the performance of the attitude estimation filter during gliding flight with motor idle.

3.3 Car Test Results

The purpose of performing car tests was two-fold. On one hand it was a full system check to verify that the communications protocol and the firmware were able to interact without a problem. On the other hand it was the first "true" test of the attitude estimation. In order to achieve this, the autopilot was fixed to a car dashboard and driven around UCSC's campus while the ground station logged data. Figure 11 shows the ground control station displaying in realtime the autopilot data and a plot (in red) of moving trail composed of the last 30 computed positions.

3.4 Hardware-in-the-loop Results

The HIL simulator makes use of the 6-DOF UAV model reported in [35] and [4]. The implemented Simulink model makes it extremely simple to receive control commands from the the autopilot and send synthetic sensor data back to the AP, therefore closing the feedback loop. The developed library for the HIL simulator consists of: **(i)** a source block (shown in orange in Figure 12) which receives commands for up to ten control surfaces; and **(ii)** a sink block (shown in green in Figure 12) which sends out all the synthetic sensor data. Experimental results show an approximately 3 sample (30 ms) round-trip time delay.

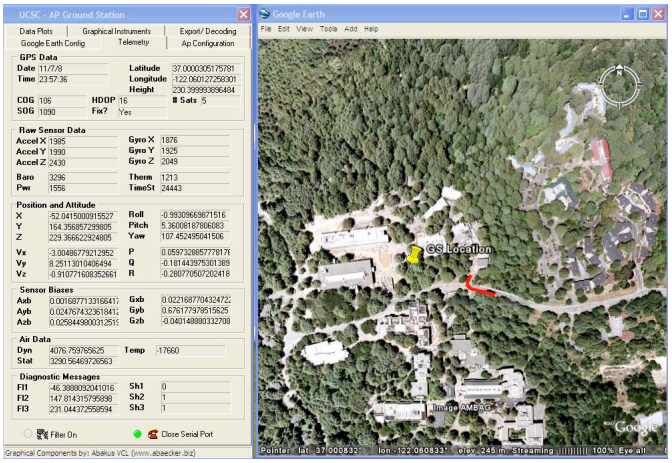

Figure 11: Ground Control Station and Google Earth Displaying the Vehicles Trajectory and Communications Protocol Data

3.5 Preliminary Flight Results

Once simulation, HIL and ground tests showed that there was no apparent problem in the autopilot, it was installed in the airframe for further flight testing. The airframe used for testing was a Multiplex Mentor electric RC aircraft (see Figure 13).

In the same manner as in the car tests, the first flight tests had the purpose of performing a full system check and asses the

Copyright © 2009 by ASME

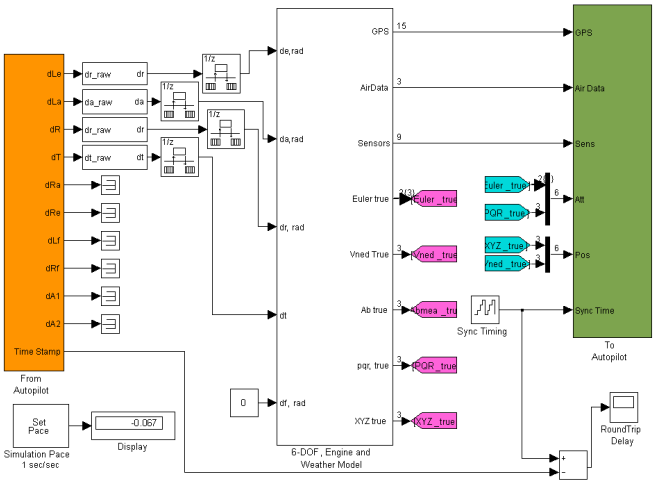

Figure 12: HIL Simulator Simulink Model

Figure 13: Multiplex Mentor Airframe with Autopilot

performance of the attitude estimation algorithm.

From the first three flights the following was confirmed:

1. The communications protocol provided a reliable way of exchanging information among all the components of the system.
2. The ground station software successfully displayed all the autopilot telemetry, logged the state of all the variables and interacted with Google Earth for display purposes.
3. The SD data logger was not able to keep up with the all the data being generated by the autopilot. This resulted in the data only being logged on the ground and not on the autopilot.
4. Due to a poor design of the autopilot mount on the airframe, the motor vibration had a severe impact on the sensor readings thus noticeably degrading the attitude estimation performance while the motor was running. This was further confirmed when analyzing flight data when the motor was off and the UAV was simply gliding.

Unfortunately, during the last flight test a failure in one of the main switching regulators in the autopilot made the autopilot loose communication with the RC pilot control and ended in crash with the subsequent airframe loss. Work is currently underway to assemble a new airframe and flight tests are expected to occur before the end of spring.

4 Conclusions and Next Steps

This paper presents a new autopilot specifically developed for R&D labs to be easily and rapidly reconfigured and reprogrammed via MATLAB Simulink without the need of high-level (C/C++) language programming. This autopilot includes the hardware, the Simulink block libraries to interact with the different system components, the ground control station, and the HIL simulator. The preliminary HIL-, ground-, and flight tests show this autopilot to be a viable platform for UAV R&D.

Future work will include: **(i)** rebuild the airframe for more rigorous flight testing; **(ii)** an improved motor mount to better isolate the sensor suite from the motor vibration; **(iii)** a better performing attitude estimator; **(iv)** implementing classical control loops as well as modifying the inner loops to include a more robust failure-tolerant scheme. Furthermore, a new hardware revision is currently in progress to reduce the hardware cost, volume, power, and using superior sensors. Finally all the components of the autopilot (software and hardware) will be released during spring 2009 for the open public under the MIT Open Source license and made available at the project's website [36] .

Acknowledgments

The authors would like to thank the Mexican National Science and Technology Council (CONACyT) for partially funding this project. Partial funding was also received from the National Science Foundation grant #0521675.

REFERENCES

[1] V. Dobrokhodov, O. Yakimenko, K. Jones, I. Kaminer, E. Bourakov, I. Kitsios, and M. I. Lizarraga, "New generation of rapid flight test prototyping system for small unmanned air vehicles," *AIAA Modeling and Simulation Technologies Conference Proceedings*, 2007.

[2] K. Nonami, "Prospect and recent research & development for civil use of autonomous unmanned aircraft as UAV and MAV," vol. 1 of *Journal of System Design and Dynamics*, 2007.

[3] D. Vos, "Fault tolerant automatic control system utilizing analytic redundancy," 2000. US Patent 6,085,127.

[4] V. Dobrokhodov, I. Kitsios, I. Kaminer, K. Jones, E. Xargay, N. Hovakimyan, C. Cao, M. Lizarraga, I. Gregory, "Flight validation of metrics driven $\mathcal{L}_1$ adaptive control," in

Copyright © 2009 by ASME

AIAA Guidance, Navigation and Control Conference and Exhibit, 2008.

[5] B. Bethke, M. Valenti, and J. How, "Cooperative Vision Based Estimation and Tracking Using Multiple UAVs," *Lecture Notes in Control and Information Sciences*, vol. 369, p. 179, 2007.

[6] U. Zengin and A. Dogan, "Real-Time Target Tracking for Autonomous UAVs in Adversarial Environments: A Gradient Search Algorithm," *IEEE Transactions On Robotics*, vol. 23, no. 2, p. 294, 2007.

[7] D. Nelson, D. Barber, T. McLain, and R. Beard, "Vector Field Path Following for Miniature Air Vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 3, p. 519, 2007.

[8] D. Jung, J. Ratti, and P. Tsiotras, "Real-time Implementation and Validation of a New Hierarchical Path Planning Scheme of UAVs via Hardware-in-the-Loop Simulation," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1-3, pp. 163–181, 2009.

[9] I. Kaminer, O. Yakimenko, A. Pascoal, and R. Ghabcheloo, "Path generation, path following and coordinated control for time critical missions of multiple UAVs," *American Control Conference*, pp. 4906 – 4913, June 2006.

[10] I. Kaminer, O. Yakimenko, V. Dobrokhodov, A. Pascoal, N. Hovakimyan, C. Cao, A. Young, and V. Patel, "Coordinated path following for time-critical missions of multiple UAVs via $\mathcal{L}_1$ adaptive output feedback controllers," *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2007.

[11] H. Hoeber, "Long-Sought Aeroplane Stabilizer," *The New York Times magazine*, p. SM9, 1914.

[12] A. P. Stephen Van Dulken, *Inventing the 20th Century: 100 Inventions That Shaped the World from the Airplane to the Zipper*. NYU Press, 2002. ISBN 0814788122, 9780814788127.

[13] D. Baker, *The Rocket: The History and Development of Rocket and Missile Technology*. Taylor & Francis, 1978. ISBN 0904568105, 9780904568103.

[14] A. Chatfield, *Fundamentals of High Accuracy Inertial Navigation*. Aiaa, 1997.

[15] B. Vaglienti, R. Hoag, and M. Niculescu, *Piccolo System User's Guide*. Cloud Cap Technology, Hood River Oregon, 2005.

[16] MicroPilot, "Micropilot: World leader in miniature uav autopilots." http://www.micropilot.com, 2009.

[17] Procerus Technologies, "Kestrel autopilot." http://www.procerusuav.com/productsKestrelAutopilot.php, 2009.

[18] P. Brisset, A. Drouin, M. Gorraz, P. Huard, and J. Tyler, "The Paparazzi Solution," *2nd US-European Competition and Workshop on Micro Air Vehicles*, November 2006.

[19] I. Kaminer, A. Pascoal, E. Xargay, C. Cao, N. Hovakimyan, and V. Dobrokhodov, "3D Path Following for Small UAVs using Commercial Autopilots augmented by $\mathcal{L}_1$ Adaptive Control." Submitted to *Journal of Guidance, Control and Dynamics*, 2008.

[20] The Mathworks, Inc., "Land Rover Vehicles Achieve EPA Certification with MathWorks Tools for Embedded Code Generation and add2 Target Hardware." User Story, 2008.

[21] The Mathworks, Inc., "Daimler-Chrysler Designs Cruise Controller for Mercedes-Benz Trucks Using MathWorks Tools." User Story, 2004.

[22] M. Schwarz, H. Sheng, A. Sheleh, and J. Boercsoek, "Matlab® / simulink® generated source code for safety related systems," *Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on*, pp. 1058–1063, 31 2008-April 4 2008.

[23] V. Dobrokhodov and M. Lizarraga, "Developing Serial Communication Interfaces for Rapid Prototyping of Navigation and Control Tasks," *AIAA*, vol. 6099, p. 2005, 2005.

[24] The Mathworks, Inc., "NASA's X-43A Scramjet Achieves Record-Breaking Mach 10 Speed Using MathWorks Tools for Model-Based Design." User Story, 2005.

[25] L. Kerhuel, "Matlab-simulink device driver blockset for PIC/dsPIC microcontrollers." http://www.kerhuel.eu/wiki/index.php5.

[26] M. Gomez, "Hardware-in-the-loop simulation," *Embedded Systems Programming*, vol. 14, December 2001.

[27] R. Klenke, "Development of a Novel, Two-Processor Architecture for a Small UAV Autopilot System." Research Agreement No. W911NF-05-1-0324 Final Report, 2006.

[28] Microchip Technologies, Chandler, AZ, *dsPIC33F Family Reference Manual*, 2008.

[29] P. Horowitz and W. Hill, *The Art of Electronics*. Cambridge University Press, 1989.

[30] The Mathworks, Inc., *Simulink User's Guide*. Natick, MA, 2008.

[31] KVH Industries Inc., "Kvh dsp-3000 high performance, singel axis fiber optic gyro." Product Brochure.

[32] G. H. Elkaim and C. C. Foster, "Extension of a nonlinear, two-step calibration methodology to include non-orthogonal sensor axes," *IEEE Transactions on Aerospace Electronic Systems*, vol. 43, no. 4, 2007.

[33] D. Gebre-Egziabher and G. Elkaim, "MAV attitude determination by vector matching," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 44, pp. 1012–1028, July 2008.

[34] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*. 2nd ed., 1992.

[35] M. I. Lizarraga, "Autonomous landing system for a UAV," Master's thesis, Naval Postgraduate School, Monterey, CA, USA., March 2004.

[36] U. of California Santa Cruz, "SLUGS: Santa Cruz Low Cost UAV GNC System." http://slugsuav.soe.ucsc.edu, 2009.

Copyright © 2009 by ASME