JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 1, April 2004

Real-Time System to Control Aircraft Propeller Pitch

Jeffrey C. Nash*
DeVry University, Fort Washington, Pennsylvania 19034
and
Phillip A. Laplante†
Pennsylvania State University, Malvern, Pennsylvania 19355-1443

A novel software system is proposed for the purpose of automatically controlling the propeller pitch of a small single engine aircraft as a function of throttle setting and aircraft altitude. This system decreases single pilot workload considerably during critical periods of flight. After introducing the problem and the proposed solution, the functional requirements are elucidated and a design presented. The design was validated for a Cessna 172RG using a simulator built with the CVI visual programming package and the results are described.

I. Introduction

WITH the 11 September 2001 terrorist attacks on the United States still in the news, major changes are being made to the air transportation system. Some foresee a greater role for small single engine aircraft in short-range business travel. In theory, these aircraft could find a greater role in transporting business and technical personnel distances of several hundred miles in a fast and cost effective manner. There are many airports spread across the United States and if used effectively, businesses could realize significant savings.

Technology used in single engine aircraft is progressing and modern instrumentation provides pilots with visual representation of real-time flight information and significant control automation. But a single pilot in such a setting is still required to interact frequently with the controls. Further simplification of power control is desirable and possible, however, via automatic regulation of propeller pitch.

The contribution of this work is to describe a design for a real-time system that eliminates the need for pilot input to control propeller pitch. Such systems have been developed for experimental amateur built aircraft.[14] These designs, however, employ electrically controlled propellers rather than the more conventional hydraulically actuated ones and such a design may prove less safe in that engine failure might not allow propeller pitch to be minimized, i.e. returned to its safest configuration. Furthermore, the transition from pitch for maximum performance to that of cruise climb is on a timer. That is, after 35 seconds has elapsed since application of full power, propeller pitch is automatically increased. It is our opinion that this is not the safest design because, in this case, propeller pitch change is not associated with any phase of flight.

Our design proposes that a single control will be used to manipulate both engine power and propeller pitch in conjunction with sensor readings. Propeller pitch will be controlled automatically by an embedded real-time system so that pilot input is unnecessary to control this parameter. The result will be reduced pilot workload. The effect is to significantly enhance aircraft performance and flight safety.

The aircraft used to prototype the proposed design was the Cessna 172 RG. It has a retractable landing gear and is fitted with a two-blade variable pitch propeller. This aircraft is equipped with a Lycoming O-360 engine, producing 180 horsepower. The propeller design is an “oil in to increase pitch” design. Oil pressure for the propeller is supplied from the engine crankcase. This is the most commonly used variable pitch propeller on single engine aircraft.

Received 31 July 2003; revision received 6 January 2004; accepted for publication 6 January 2004.
*Assistant Professor, Computer and Information Science Department, 1140 Virginia Drive.
†Associate Professor, Software Engineering, 30 East Swedesford Road.

II. Aircraft Propellers

The purpose of a propeller is to convert the rotational energy of an engine to thrust in a specific direction. Propellers may be either of a fixed or variable pitch design.[2] Each unit typically has between two and six blades.[5] Propeller theory has been well developed for marine[8,1] and for aerodynamic applications,[1] but it is not our intent, nor is it necessary, to review the mathematics here. We simply discuss those operational aspects that are relevant to our objective.

The simplest propellers are the fixed pitch type. That is, their propeller blades cannot be moved to alter pitch, β, at a given time. These propellers exhibit maximum efficiency of 80-85% at a single optimal airspeed. Aircraft designers select β to suit the mission of the aircraft. Propellers with lower values of β give excellent takeoff performance while high values provide better cruise speed. Intermediate β values represent compromise between the two.[11] A very good presentation of considerations for selection of variable pitch propellers exists.[6]

A more efficient propeller employing oil pressure controlled by a governor to rotate propeller blades about their longitudinal axis was pioneered in the 1930s.[13] Since each value of pitch shows peak efficiency proportional to air speed, β can be changed to maintain maximum system efficiency. Once set, a propeller governor adjusts this pitch angle so that a desired rotational speed, RPM, is maintained. For this reason variable pitch propellers are also referred to as constant speed propellers.

Controllable pitch aircraft propellers for single engine aircraft are usually designed so that oil pressure has to be applied to the propeller to increase its pitch. This is a safety feature. Should there be an engine failure, oil pressure will decrease. Centrifugal force would then cause such a propeller to return to its minimum pitch. Aerobatic single engine aircraft, turbine powered aircraft, and multiengine aircraft employ variable pitch propellers of a different design.[6]

In normal operation, the pilot controls propeller pitch via mechanical linkage to a propeller governor. Once set the governor will modulate oil pressure to maintain a desired propeller and engine speed. As odd as it may seem, the instrument that is used to control propeller pitch is the tachometer, which indicates engine rpm. Pitch is set to produce a specific rpm. Should engine power change, propeller pitch will automatically be changed to maintain this rpm.

The power instrument is a pressure gauge whose reading is the absolute pressure in the intake manifold. Readings are given in inches of mercury. When the engine is not running, the manifold pressure gauge reads atmospheric pressure, 29.92 in. At low power settings manifold pressure is typically 12 in. or lower. Conversely, very high power settings result in manifold pressures slightly below atmospheric pressure. Engines with supercharging can show manifold pressures greater than ambient atmospheric. Manifold pressure will decrease with altitude since atmospheric pressure decreases. Turbocharged engines will maintain sea level manifold pressures at altitude.

It is essential that care be taken in the adjustment of propeller pitch. Sudden reduction in β at high power settings can over-speed the engine, whereas increases in power at high β settings can over-torque the engine. Both situations lead to breakage of an engine's internal workings. A rough operating guideline for variable pitch propellers showing the relationship between engine rpm, R (indicating β) and manifold pressure, P_{MP}, expressed in inches of mercury (indicating power) can be deduced:[4]

$$R \geq 100\ (P_{MP}) \tag{1}$$

A more refined, but less flexible, guideline for propeller pitch and power settings is presented by Kershner.[10] Furthermore, a very important rule-of-thumb dictates that when increasing power, changes in propeller pitch will lead those of manifold pressure. When reducing power, manifold pressure should be reduced before propeller pitch.[3]

III. Motivation

During certain aircraft operations, pilot workload can be very high, especially when there is only a single pilot as is most often the case with single prop aircraft. In such cases, it is desirable to offload as much work from the pilot as possible and place it in the hands of automated systems. The following scenario illustrates this point.

A recent hearing of the US House of Representatives Subcommittee on Aviation on Air Traffic Congestion discussion centered on New York City's extremely high volume of air traffic.[15] Testimony indicated that there are 1.4 million aircraft operations* among the three largest Port Authority airports in the area, Kennedy, LaGuardia, and Newark. Typically, these aircraft fly under Instrument Flight Rules (IFR), which involves receiving an IFR

*An aircraft operation is movement of an aircraft that involves a landing or take off.

clearance prior to takeoff. Such a clearance specifies the route and altitudes to be flown. As a flight operates in congested airspace, Air Traffic Controllers (ATC) specifically Departure Control, may find it necessary to amend an IFR clearance.[7] In essence, an aircraft can be assigned a new route and set of altitudes invalidating the clearance received prior to departure. In this case sophisticated flight control systems need to be reprogrammed, literally, on the fly. In a two pilot cockpit reprogramming can easily be done by the non-flying pilot. However, in a single pilot aircraft, a complicated navigation system must be reprogrammed while the pilot flies the aircraft.

Using a popular GPS system, entering a new waypoint for an amended clearance involves pressing a series of buttons to bring up a set of menus on a small screen. The current flight plan must be canceled and new one entered. Once the appropriate menus are viewed, a cursor is moved to get the desired item and then selected with yet another button push. Menus are re-acquired, the appropriate item selected to enter a new flight plan. The waypoint identifier, a 3-4 character alphanumeric string, is entered through a series of knob twists. Once the identifier is entered correctly, another key is pressed to activate this segment of flight. For this one segment alone, there may be 10 button presses and 5-7 knob twists.[9] An amended clearance may have several new segments. Of course, en route and arrival operations tend to be much more routine because there is less congestion and predictable radar vectors are issued. Deviations tend to be weather related and pilot initiated. During the arrival phase, deviations are often handled with radar vectors, i.e. Arrival Control will assign headings and altitudes to specific aircraft. Turns and altitude changes are accomplished with ATC instructions. But in any case, the navigational programming represents a significant workload in a single pilot environment.

All of this navigational activity contributes greatly to the pilot workload at the time when complicated aircraft control is also needed. Hence, a computer controlled pitch adjustment system, which could manage thrust control, would serve to alleviate the pilot of an important responsibility at a time when he or she is highly tasked.

IV. The Proposed System

The design philosophy of the propeller pitch control system is to take advantage of the existing design of the aircraft and to mimic the training of its pilot. The aircraft is equipped with proximity switches that detect the compression of the landing gear as it supports the weight of the aircraft. Landing gear compression due to weight on its wheels is used to indicate an on-ground condition. This reading is used to disable the landing gear retraction mechanism. The proposed design uses this information in conjunction with throttle setting to determine when it is appropriate to conduct a propeller system check.

A. Design Philosophy

Pilots are trained that once airborne, aircraft performance is determined solely by an aircraft's power setting and its bank and pitch attitude.[10] Hence, the proposed design will determine appropriate propeller pitch setting from throttle setting and aircraft attitude. High power settings and lower airspeeds indicate high performance climb. This requires minimum value of β. An aircraft equipped with a controllable pitch propeller should not leave this high performance, or take off, state before reaching at least 500 ft altitude above ground level (AGL). Power settings slightly reduced from maximum and increased airspeed indicate cruise climb. Moderate to high power settings combined with approximately constant altitude are characteristic of level cruising flight. In the proposed design, β will be controlled in each of these cases and in the on-ground propeller system check. These are presented in Tables 1a through 1c. All other circumstances will require that propeller pitch be set at its minimum value.

Table 1a Indications that on-ground system check is appropriate

Purpose:	Check the mechanism that controls propeller pitch to ensure that it is fully responsive. The propeller governor is supplying oil to the propeller.
Procedure:	Cause the propeller governor to send the propeller through its full range of pitch. This is done by increasing the power with the propeller pitch at its minimum value until RPM is 1,800 ±50 then sending the propeller pitch through its full range of travel and returning it to its minimum setting. Cycle time is 2 – 3 seconds. The process is repeated twice.
Sensors Needed:	1. Proximity Switch(es) indicating that aircraft weight is on its wheels (on ground condition). 2. Intermediate Power Setting: Minimum β, RPM = 1,800 ±50. 3. Airspeed should be near 0.
Note:	It is important that during the on-ground condition at power settings other than 1,800 ±50 RPM, β must remain at its minimum value.

Table 1b Indications that cruise climb pitch setting is appropriate

Purpose:	More efficient propeller setting; Better engine cooling; Better forward visibility.
Procedure:	After all obstacles have been cleared (e.g. 500 ft AGL), the pilot maintains full power and uses elevator control to decrease the aircrafts pitch. The aircraft will accelerate to approximately 80-90 knots, manifold pressure is reduced to 25 in. by the pilot after which propeller RPM is adjusted to 2,500.
Sensors Needed:	1. Airspeed reading in the range of 80-90 knots 2. Altitude. 3. Proximity Switch(es) to determine that the aircraft is flying. 4. Manifold Pressure equal to 25-25 1/2”.
Note:	Shortly after take off, aircraft AGL can be determined by subtracting the altitude when weight was relieved from the wheels from the current altitude reading. Terrain makes this method implausible after 2-3 minutes.

Table 1c Indications that level cruise pitch settings are appropriate

Purpose:	Most efficient propeller pitch for the power setting used.
Procedure:	In level flight, propeller pitch is adjusted and maintained so that: 100 x (Engine Manifold Pressure) ≤ Engine RPM
Sensors Needed:	Altimeter (rate of change in altitude should be approximately 0) Power and RPM according to Eq. (1).
Note:	Most pilots fly with slight altitude deviations, the change in altitude cannot be assumed to be precisely 0 during cruise. Once power has been reduced enough that β is at its minimum value, further reduction of power does not affect the value of β.

Table 2 Functional requirements of a real-time system to control propeller pitch

Requirement Number	Requirement Statement
FR01	The engine shall never be operated at a condition of over speed ($R > 2{,}700$)
FR02	The engine shall never be operated at a condition of over torque (See Equation 5)
FR03	At any time, should the engine incur an uncommanded power reduction, propeller pitch shall be returned to its minimum pitch setting since engine oil pressure will be reduced.
FR04	During ground operations, change of propeller pitch shall be enabled only between engine speeds of 1,800 ±50 rpm.
FR05	During the propeller systems check, the propeller shall be sent through its full range of pitch settings. This is 12° to 26$^{1/2}$° at the propeller’s 30 in. station.
FR06	During the takeoff phase of flight, the propeller shall remain at its minimum pitch setting.
FR07	During the climb out phase of flight, the propeller pitch shall remain at it minimum setting between that altitudes of 0-500 ft AGL.
FR08	During the climb out phase of flight and upon attaining 500 ft AGL and upon a power reduction to 25 in. manifold pressure, propeller pitch shall be increased so that the engine speed is 2,500 rpm.
FR09	When reducing power, reduction in power settings shall lead those of propeller pitch.
FR10	When increasing power, increase in propeller pitch setting shall be accomplished before increasing power.
FR11	During the landing phase, the system shall provide the pilot with visual confirmation that the propeller is set to its minimum pitch by way of lighting an LED.

B. Functional Requirements

From the design philosophy just mentioned and the *C-712RG Information Handbook*,[4] a series of functional requirements was developed that must be satisfied by any real-time system designed to control propeller pitch in this application. They are presented as Table 2. The proposed requirements completely define the behavior necessary to attain system control.

C. System Models

Architecture and behavior of the proposed real-time system can be represented in pseudo code, as in Appendix A or diagrammatically. Useful diagrams for representing real-time control systems include Activity Diagrams, Sequence Diagrams and a Finite State Machines. These diagrams, which are described shortly, are members of the Unified Modeling Language (UML) family of languages, which is widely used to model embedded real-time systems.[12]

1. Activity Diagram

Activity diagrams are closely related to the flow chart and are used for the same purpose, that is, to show flow of control. Typically, they are used to model dynamic aspects of a system. However, unlike flow charts, they can model concurrent computational steps and the flow of objects as they move from state to state at different points in the flow of control.

An activity diagram for the proposed propeller control system is presented in Fig. 1. This diagram depicts the system starting operation in an initialization and diagnostic procedure. From this point, it proceeds to ground operations from which it is possible to check propeller performance if power setting falls within the range of 1,800 ±50 RPM while the propeller pitch, β, is at its minimum value. Take off is by application of full power. Note that it is possible to abort the take off and return to ground operations by reducing power to idle.

If the activity diagram proceeds to the next step, proximity switches open upon takeoff, an altimeter reading is taken to determine runway elevation, and airspeed increases. During this portion of the flight, the pilot will accelerate to and maintain a Best Angle of Climb airspeed, Vx of approximately 77 knots. Here, there will be no changes in β until the aircraft is 500 ft above the takeoff elevation.

Once above 500 AGL, the system will increase β so that 2,500 rpm is attained as the pilot reduces manifold pressure (power) to 25 in. In level flight, the altimeter reading will be approximately constant. At this point the system can set power so that engine RPM is equal to 100 times the power setting in inches of mercury as per Eq. (1). From the beginning of take off roll to 500 ft AGL constitutes a critical time, and the system should illuminate an appropriate LED to indicate that β is at its minimum value.

If power setting is sufficiently reduced, β will approach its minimum value. Further reductions in power have no effect upon β. Descent and approach to landing are indicated by lower power settings in flight. Manifold pressure or power settings of less than 12 in. suggests an approach to landing. In this case, a light indicating minimum β should be illuminated.

Note that it is possible to reach previous states through this diagram. For example, if the aircraft is set for descent and the pilot wishes to climb, the aircraft must first enter level cruise to attain this condition. This is exactly how the aircraft is flown.

2. Sequence Diagram

Sequence diagrams are composed of three basic elements – objects, links, and messages. Every object in a sequence diagram has an associated timeline (called a "lifeline"). The lifeline is present whenever the object is active and is represented graphically as a vertical line with logical time traveling down the line. The objects for the sequence diagram are displayed horizontally across the page and are shown staggered down the diagram depending on when they are created.

The sequence diagram for the prop control system, Fig. 2, shows how the system responds to outside stimuli.

In this system, outside stimuli include pilot input through the throttle and sensor readings. Upon start up, the system does a self check and pronounces itself healthy or not. On the ground, the pilot is allowed to do a propeller system check if throttle setting is appropriate. Airborne, the system will adjust β only if appropriate power, airspeed and altitude readings exist. Careful examination of Fig. 2 with knowledge of piloting technique shows that the sequence covers all normal flight modes. Loss of engine power would result in loss of oil pressure. This in turn will cause β to return to its minimum value, just as is required for emergency operations.

3. Finite State Machine

The Finite State Automaton (FSA), Finite State Machine (FSM), or State Transition Diagram (STD) is a formal mathematical model used in the specification and design of a wide range of systems. Strictly speaking, the FSM is

not a member of the UML family. The Harel Statechart, however, which subsumes the FSM is a member. A brief tutorial on system design using Finite State Machines is given in Appendix A.

Intuitively, Finite State Machines rely on the fact that many systems can be represented by a fixed number of unique states. The system may change state depending on time or the occurrence of specific events—a fact that is reflected in the automaton.

Figure 3 represents the proposed system as a Finite State Machine. The corresponding state transition matrix for the proposed system is given in Table 3.

The Finite State Machine is similar to the activity diagram in flow, but represents the system as a series of states. The system can exist in only one state at any given time. Movement between states occurs upon an appropriate event. An event can be considered a change of conditions seen by the system.

The Finite State Machine begins with a power up and diagnostic. Once this is completed, state is changed to ground operation, never to return to its original state until power to the electronics is cycled. From ground operation, the system can freely move between states as external conditions dictate. Note that system can terminate from any state upon power being turned off. Additionally, the system will do self diagnostics during its idle time. Self-testing occurs in all states.

The Finite State Machine is of special interest. Its representation can be implemented directly either in hardware or software as is discussed in Sec. V.

4. Relationship Between Propeller Pitch and Power

During the course of this study, the Cessna 172 RG's Aircraft Information Manual[4] was consulted to determine operational parameters. Multiple linear regression was applied to its tabular cruise performance data.[10] An attempt was made to use this data to predict the best RPM as a function of power setting and altitude. The effort failed since the data was rather evenly distributed through the region of interest. The resulting Coefficient of Correlation was approximately zero, indicating no correlation.

A decision was made to use the general rule for operation of variable pitch propellers. That is manifold pressure must not be greater than 1% of RPM as shown by Eq. (1). This rule was checked against published operating limits. No case exists in which the rule violates published procedures.

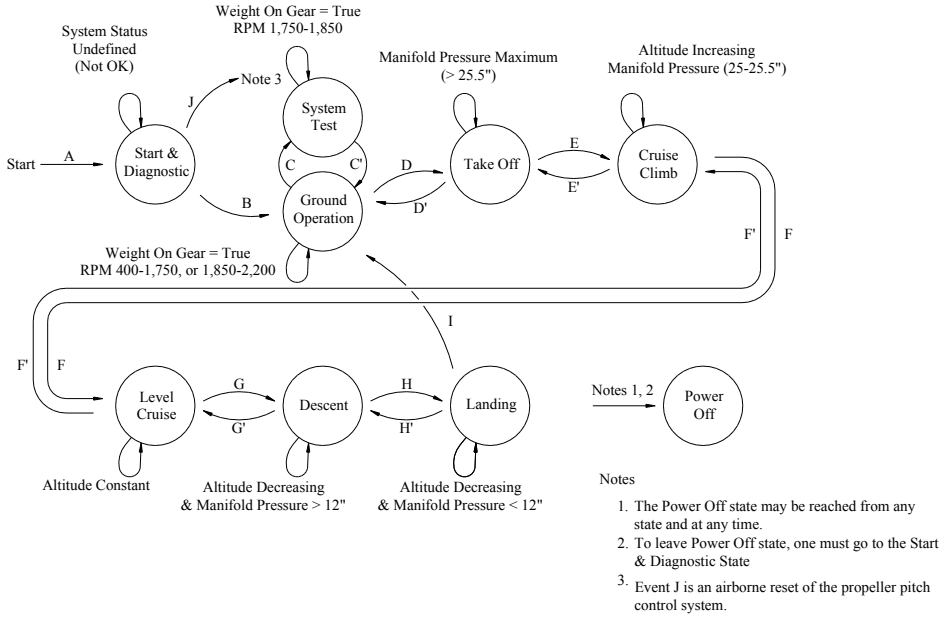

Fig. 3 Partial Finite State Machine depicting the proposed propeller pitch control system.

Table 3 State Matrix for Finite State Machine depicting the proposed propeller pitch control system.

Event	Voltage	Weight on Gear	Airspeed[1]	Sensor Altitude	Manifold Pressure	RPM	Status Discrete
A	≥12 V	True	Undefined	N/A[6]	N/A	N/A	Not Set
A'	<12 V		Undefined	N/A	N/A	N/A	N/A
B	≥12 V	True	Undefined	N/A	N/A	N/A	OK
C	≥12 V	True	Undefined	N/A	N/A	1,750-1,850	OK
C'	≥12 V	True	Undefined	N/A	N/A	<1,750 or 1,850-2200	OK
D	≥12 V	True then False	Undefined - 80	[2]	> 25 1/2''	>2,550	OK
D'	≥12 V	True[3]	N/A	N/A	<12''	<1,000	OK
E	≥12 V	N/A	80-85	> 500 ft AGL & Increasing	25-25 1/2''	2,500	OK
E'	≥12 V	N/A	75-80	> 0 ft AGL & Increasing	25 1/2''	>2,500	OK
F	≥12 V	N/A	> 60	Constant[4]	> 21''	Nominal[5]	OK
F'	≥12 V	N/A	80-85	Increasing	>25	Nominal[5]	OK
G	≥12 V	N/A	N/A	Decreasing	N/A	Nominal[5]	OK
G'	≥12 V	N/A	N/A	Constant[4]	< 21''	Nominal[5]	OK
H	≥12 V	N/A	N/A	Decreasing	<12''	Nominal[5]	OK
H'	≥12 V	N/A	N/A	Decreasing	12-21''	Nominal[5]	OK
I	≥12 V	True	N/A	N/A	N/A	<1,750 or 1,850-2200	OK
J[7]	≥12 V	False	N/A	N/A	N/A	N/A	OK

[1]Undefined airspeed means that airspeed is too low to register (0 to approximately 35 knots).
[2]Record the altitude when Weight on Gear becomes false.
[3]This is an aborted takeoff. Acceleration followed by deceleration on the runway.
[4]Will have to allow some deviation from constant, e.g. ≤±100 ft /min.
[5]RPM specified by Equation 1 or minimum RPM when no further reduction is possible.
[6]N/A indicates a reading that is not necessary to control the system.
[7]Event J is an airborne reset of the propeller pitch control system.

V. Implementation

Our first choice was to implement the system as an embedded software system running in RAM. It became clear that since the solution can be expressed succinctly using FSMs; the proposed system can be implemented using integrated circuits known as a Field Programmable Gate Arrays (FPGA). That is, the Finite State Machine of Fig. 3 and the logic of Tables 1a-1c can used to represent the system as Boolean Logic. This Boolean Logic would be programmed into an FPGA using a high-level circuit description language such as VHDL or Verilog, so that the program logic will be embedded in the integrated circuit. These devices can be mass produced and distributed in production quantities.

In order to test this design, however, our objective was to build a prototype using some sort of software emulation. Upon successful validation of the emulator, then a hardware prototype could be considered.

A. Emulator Requirements

The following statements depict the functional requirements developed during emulator design. Each functional requirement is marked with a Functional Requirement tag and number of the form FR###.

[FR101] The system and the FPGA shall communicate by reading and writing information from and to specific memory locations.

[FR102] Both the emulator and the FPGA shall regularly read appropriate memory locations to determine if new information has been presented by the other system.

[FR103] Both shall write to appropriate memory locations anytime data values have changed. It is anticipated that the emulator user will design test cases to exercise the FPGA. The user will encode parameter settings into a data structure. Each set of parameters will be assigned a duration by the user.
[FR104] The emulator shall read duration and parameter settings from the data structure and write them a specified memory location.
[FR105] These values shall remain unchanged for the duration associated with parameter set.
[FR106] A timer shall be set up and used measure elapsed time since the previous write operation.
[FR107] When the time specified by the value of duration has elapsed, a new set of parameters shall be read and used to overwrite current values.
[FR108] Upon the entry of new values, the timer shall be reset to zero.
[FR109] Values accepted from the data structure shall have minimum and maximum allowable values to be determined by the emulator developer.
[FR110] In the event that values are read from the data structures that are outside of these bounds, the emulator shall have a means of recovering with acceptable values determined by the developer.
[FR111] Any combination of in range values shall be allowed.
[FR112] Upon any change in emulator parameter, the emulator shall write the time, the entire parameter list and the FPGA output (propeller rpm) to a file.
[FR113] This file shall be appended each time new information appears.
[FR114] The emulator shall have a graphical user interface (GUI) front end.
[FR115] The GUI shall display and regularly update the following parameters: Propeller RPM, Propeller Pitch, Engine Power, Altimeter Reading, Airspeed Reading, and Weight on Landing Gear, Aircraft Pitch, and Heading.
[FR116] It shall be possible for the emulator user to change aircraft pitch, airspeed, and engine power setting through the GUI.

B. Emulator Design

The emulation environment used in this project was National Instruments CVI (C Virtual Instruments), which is a visually-oriented, C language programming environment. A CVI panel, which is essentially a GUI, was developed first. Control options were placed upon the CVI panel and callback functions generated. C code to support the behavior of the control was written and inserted as appropriate. Controls used in the emulator are grouped by their type and discussed in detail next.

Sliding Bars

Throttle	Controls engine power and is scaled from 0-100%. Throttle changes are seen as responses on the tachometer (RPM), manifold pressure gauge and airspeed.
Propeller Pitch	Prop Pitch varies the angle of attack of the propeller blades. Propeller pitch on the C-172RG varies from 12-26^{o}. Changes in this control result in RPM, manifold pressure and airspeed changes.
Elevator	This controls the aircraft pitch. Climbing involves pitch up while descending requires pitch down. Inputs vary over the range of -10^{o} to +10^{o}. Inputs are reflected in altitude and airspeed gauges.
Bank	Rolls the aircraft for the purpose of turning. A bank of 7^{o} results in a standard rate turn (3^{o} compass heading change per second). Bank input results in changes to compass heading.

Meters

Bank Angle	A meter much like an inclinometer to indicate direction and magnitude. Reflects the value of the Bank Sliding Bar.

Gauges

Tachometer (RPM)	Reflects throttle and propeller pitch setting. Range is 0-2700 rpm.
Manifold Pressure	Also reflects a combination of throttle and propeller pitch settings. Operating range is 8-25 in. of mercury. When the engine is not running, manifold

	pressure is 29.92 in. of mercury (Atmospheric Pressure).
Airspeed	Supplies indicated airspeed. Indicated airspeed is affected by throttle, propeller pitch, and elevator.
Compass Heading	Indicates the direction in which the aircraft is headed. It functions both in air and on ground. It derives its input solely from the Bank sliding bar.
Altitude	Essentially a function of elevator pitch. However, rate of altitude change is affected by throttle, propeller pitch and elevator pitch. This aircraft has a service ceiling of 15,000 ft. Negative altitudes are not possible.

Toggle Buttons

Engine	Toggles between engine running and engine off. Turning the engine off results in output power being set to zero. This will affect all parameters needing power, e.g. airspeed, altitude, power, rpm.
Brakes	Set/Release. Will stop the aircraft's forward motion on ground. No airborne effect.
Landing Gear	Down/Up. Can be retracted in air only. Doing so results in an increase in airspeed due to reduced drag.
Control Type	Manual/Autopilot. In manual mode, the aircraft is flow from the controls on the panel. In Autopilot, it is flown according to a script.

LEDs

On Ground	Lighted when a squat switch detects weight on the landing gear.
Gear Up	Lighted when a proximity switch indicates that the landing gear has been retracted into its wells.
Brakes Set	Lighted when the breaking level is pulled. In this case, airspeed remains equal to 0.
Engine Operating	Lighted only when the engine is running.
Stall	Lighted when the aircraft is flying and indicated airspeed is less than 42 knots. The aircraft is stalled in the condition. This is a very serious condition in which the aircraft is falling from the sky at 10,000 ft/min. The stall is aggravated by an autorotation equal to 20 rpm. This dangerous condition is known as a spin.
Autopilot	Lighted when the aircraft is being flown according to a script.

Timers

One Second Timer	The time dependent functions of the aircraft (those not on a toggle button) are evaluated and output responses updated once per second.
Two Second Timer	Necessary for bank angles of +/- 2^{o}. Heading will change by 1^{o} every 2 seconds.
Three Second Timer	Necessary for bank angles of +/- 1^{o}. Heading will change by 1^{o} every 3 seconds.

A screen layout was developed. It is presented as Fig. 4. The aircraft can be flown according to a script (Autopilot). Script parameters are supplied through a series of six numeric boxes. If the engine is on in this mode the aircraft takes off and flies the runway heading until it reaches its Turning Altitude. This value is obtained from the first numeric box.

The aircraft continues its climb and turns at standard rate to its initial heading (from a numeric box). Upon reaching the initial heading the climb continues to cruising altitude (also obtained from a numeric box). Length of cruising flight is supplied by the numeric box entitled, Flight Time. After this time has elapsed the aircraft begins its descent to an approach altitude. Accomplishing the descent, it turns to the runway heading. Descent to landing follows.

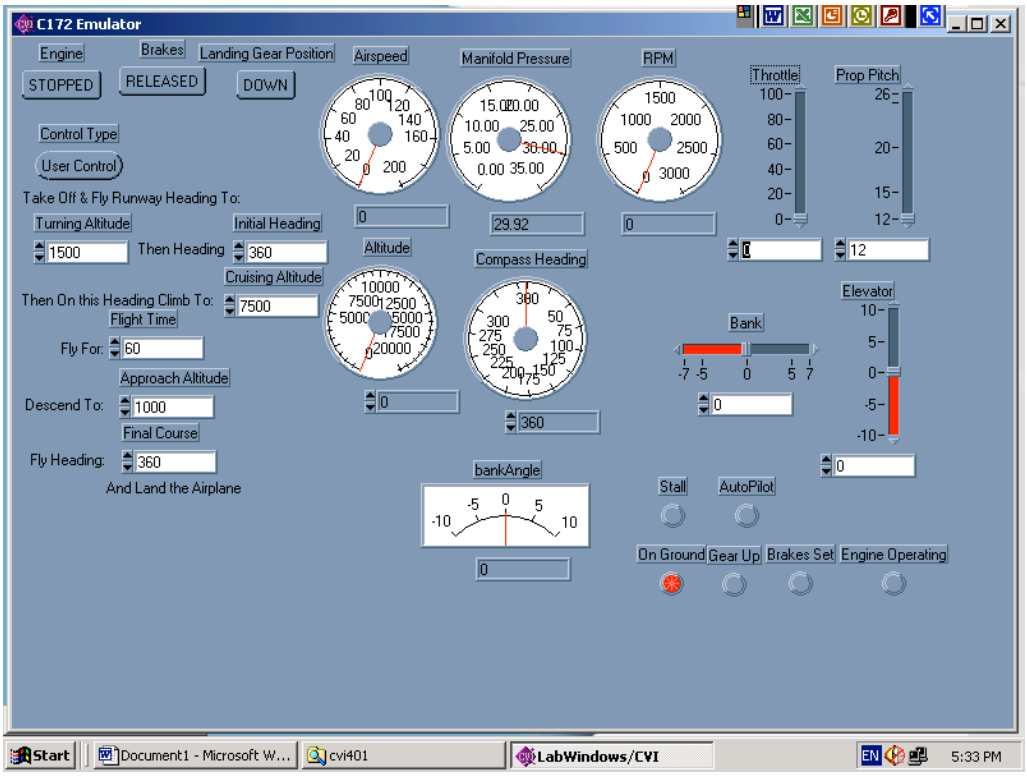

Fig. 4 Screen layout of system emulator.

A set of multimedia files showing the simulator in action (Microsoft® Media Player compatible file) can be found at http://www.personal.psu.edu/faculty/p/a/pal11/resources.htm. The simulation consists of four files. A better understanding of these files can be attained by the reader referring to the screen layout in Fig. 4 prior to viewing a particular video.

Video 1 shows an on-ground test of the propeller governor. The propeller pitch is set to its minimum setting and the throttle adjusted so that engine speed is approximately 1,800 rpm. The system is tested by moving the propeller pitch setting to its maximum then to its minimum (a process termed “cycling the prop”). Note that when this happens, rpm drops radically. This cycling is performed twice.

Video 2 shows that before attaining an altitude of 500 ft AGL, the landing gear is retracted. Upon reaching 500 ft AGL, the throttle is used to reduce manifold pressure to approximately 25 in. of mercury and subsequently, the propeller pitch control is used to reduce engine speed to approximately 2,500 rpm.

Video 3 demonstrates a leveling off in altitude, in this case, at 5,000 ft. Aircraft pitch is reduced to level through use of the elevator control. Manifold pressure is reduced to approximately 23 in. of mercury and engine speed to 2,300 rpm.

Video 4 demonstrates setting the propeller pitch to its minimum setting in a descent configuration prior to landing. Note that the airspeed is slightly decreased as this happens.

C. Testing

Black box testing was employed to evaluate the emulator, that is, test cases represented a set of inputs and expected responses, without regard to the internal workings of the software. The emulator was evaluated in the following operations over the range of possible input values:

1) ground operations
2) ground systems check
3) take off and climb out at rates up to 1200 ft/min
4) turns to heading in level flight and while climbing and descending
5) descent to landing
6) stall characteristics, stall entry, stall recovery

7) spin characteristics
8) flight according to script (autopilot)

In each case the system performed satisfactorily, that is, it met all requirements.

VI. System Limitations

There are three cases in which propeller pitch should be increased from its minimum value. They are the on-ground system check, cruise climb and level cruise. All other ground and flight states should use minimum pitch settings. In all critical phases of flight, an LED should be illuminated to advise the pilot that propeller is in its minimum pitch setting. This is important at takeoff, maximum performance climb, and approach to landing.

Use of this system on single engine aerobatic and turbine aircraft as well as multi-engine aircraft is ill advised. These aircraft employ variable pitch propellers of another design. Furthermore, there are single and multi-engine aircraft whose propellers can adjust β to negative values.[6] The proposed system considers none of these cases. Development of systems to control β for these aircraft should be considered separately.

Finally, the system should be implemented with at least one redundant system for the purpose of enhancing its reliability. There will be error checking during process idle time. When an error is detected, the system will annunciate the failure and switch to single system operation. Service of the malfunctioning hardware system can be performed once the aircraft has been returned to the ground.

VII. Conclusions

We have demonstrated, through a software prototype, that propeller pitch control for a single engine aircraft is feasible. Further, in this application note we have explored various design issues for this novel application using existing techniques and best practices. Our overall approach can, in turn, be adapted to many other avionics applications, and the proposed use of field-programmable gate arrays suggests many other inexpensive and adaptable solutions.

The solution to the prop pitch control problem is relatively simple, but we know of no single prop system in which pitch control is automatic. Moreover, our solution is highly adaptable for use in most other single engine aircraft is trivial since the logic remains unchanged – only new airspeeds and power settings need to be input.

Future work includes constructing a hard prototype using FPGAs and field testing the system.

Appendix A: A Description of the System in Pseudocode

From the requirements in Table 2 and the system behavior outlined in Tables 1a-1c and Figs. 3-5, the pseudocode for the proposed system was developed. This is presented as Table A1.

Table A1. Pseudocode of proposed real-time system to control propeller pitch

```
At system startup, perform system diagnostics on both boards (boards are redundant)

Set Designated_Board to Board#1
Do Forever
        {
        Check Health of Designated_Board
        Switch on Phase of Flight
                {
                //We on the Ground & configured for propeller system check
                CASE 1: Prox Switch = OPEN && Man. Pressure (18-20")
                        IF the prop cycle switch is activated
                                {
                                Over a 2.5 second period, change pitch setting
                                min -> max -> min
                                } end IF the prop cycle switch is activated

                //We have taken off had have 500 ft of altitude or more
                CASE 2: Prox Switch = OPEN && Power = 100% && Altitude >= 500'
                        IF Airspeed = 80-90
                        {
                        Set propeller speed to 2500 RPM
                        Set CRUISE_READY = true
                        } end IF Airspeed = 80-90

                //We are in Level Cruise Flight & want most efficient prop setting
                CASE 3: CRUISE_READY = true && Airspeed > 90
                                && Altitude = Constant && Man. Press > 22"

                        {
                        Set RPM = 100 * Manifold Pressure (inches)
                        } end CRUISE_READY = true && Airspeed > 90, etc

                //Very low power setting suggests an approach to landing
                CASE 4: Man. Pressure < 12"
                        Propeller Pitch Setting to minimum
                        Illuminate Minimum Pitch LED
                Default: All other situations
                        Propeller Pitch Setting to minimum
                }End Switch/Phase of Flight
                Swap the Designated_Board so that the other one can be checked
        }end Do Forever
```

Appendix B: Design Using Finite State Machines[12]

One of the advantages of using Finite State Machines (FSM) in the software requirements specification and later in the software design is that they are easily converted to code and test cases. A Finite State Machine can be specified in diagrammatic, set-theoretic and matrix representations. Finite State Machines lend themselves well to structured implementation, and while it might not seem obvious, also to object-oriented implementation.

To illustrate a high-level design using an FSM, consider a simplified avionics system for a fighter aircraft. The avionics system computer can operate in one of five modes: takeoff (TAK), navigation (NAV), navigation/evasive (NAE), navigation/attack (NAA), landing (LAN). The avionics system computer reacts to various signals received from other computers in the aircraft. These signals consist of: mission assignment (MA), enemy locked-on (LO), target detected (TD), mission complete (MC), enemy evaded (EE), enemy destroyed (ED). The initial state is TAK and the only terminal state is LAN. The system behavior can be described with a case statement or nested if then statements such that given the current state and receipt of a signal, a new state is assigned by the following pseudo-code:

```
TAK:
        If MA then NAV
        Else TAK

NAV:
        If TD then NAA
        If MC then LAN
        If LO then NAE
        Else NAV

NAE:
        If EE then NAA
        Else NAE

NAA:
        If LO then NAE
        If ED then NAV
        Else NAA

LAN:
        Else LAN
```

Pictorially, the Finite State Machine corresponding to this pseudo-code is shown in Fig. B1.

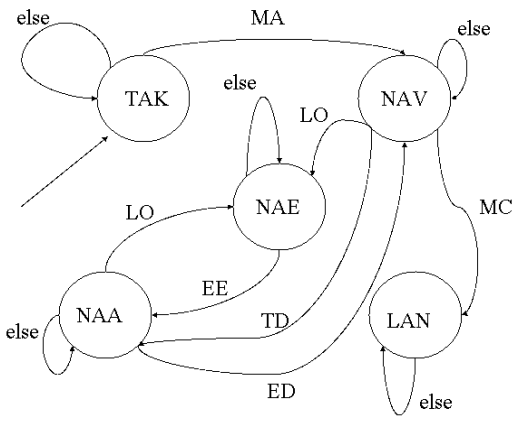

Fig. B1 Finite state machine corresponding to behavior of avionics system.[12]

The tabular representation of the state transition function, which describes the system's high-level behavior is shown in Table B1.

Table B1 Tabular representation of Finite State Machine shown in Fig. B1 (Ref. 12)

	MA	LO	TD	MC	EE	ED
TAK	NAV	TAK	TAK	TAK	TAK	TAK
NAV	NAV	NAE	NAA	LAN	NAV	NAV
NAE	NAE	NAE	NAE	NAE	NAA	NAE
NAA	NAA	NAE	NAA	NAA	NAA	NAV
LAN	LAN	LAN	LAN	LAN	LAN	LAN

Formally, the Finite State Machine can also be represented mathematically by the quintuple**.**

$$M = \{S, i, T, \Sigma, \delta\} \tag{B1}$$

where S is a finite, non-empty set of states, i is the initial state (i is a member of S), T is the set of terminal states $(T \subseteq S)$, Σ is an alphabet of symbols or events used to mark transitions, δ is a transition function that describes the next state of the machine given the current state, and a symbol from the alphabet (an event). That is, $\delta : S \times E \rightarrow S$. In the avionics system example, the set of states, S = {TAK, NAV, NAE, NAA, LAN}, the initial state, i=TAK, the set of terminal states is {LAN}, and the alphabet, Σ ={MA, LO, TD, MC, EE, ED}. The transition function is embodied in the diagram, and can be described formally as a set of triples, but it is usually more convenient to represent the transition function with a transition table, as shown in Table B1.

Any of these forms can be easily transformed into a design working system using the pseudo-code shown in Fig. B2.

```
typedef states: (state1,...,staten);        {n is# of states}
        alphabet: (input1,...,inputn);
        table_row: array [1..n] of states;
procedure move_forward;        {advances FSM one state}
var
        state: states;
        input: alphabet;
        table: array [1..m] of table_row;   {m is the size of the alphabet}
begin
        repeat
          get(input); {read one token from input stream}
          state:=table[ord(input)] [state];     {next state}
          execute_process (state);
          until input = EOF;
end;
```

Fig. B2 Pseudo-code that can implement the behavior of the finite state machine shown in Fig. B1 (Ref. 17).

Each procedure associated with the operational modes can be viewed as executing in one of any number of process states at an instant in time. This functionality can be described by the pseudo-code shown in Fig. B3.

```
Procedure execute_process (state: states);
begin
        case state of
        state 1: process 1;          {execute process 1}
        state 2: process 2;          {execute process 2}

...
        staten: processn;            {execute process n}
end
```

Fig. B3 Finite State Machine code for executing a single operational process in the avionics system. Each process can exist in multiple states, allowing for partitioning of the code into appropriate modules.[17]

The advantage of Finite State Machine design over the case statement, of course, is that the former is more flexible and compact. Moreover, the pseudo-code can easily be coded in any procedural language, or even an object-oriented one.

The object-oriented implementation of the FSM logic is perhaps not obvious, but is equally convenient using one of the design patterns (object-oriented recipes), called STATE, which can be found in Gamma et al.[16] In fact, an excellent description of such an implementation can be found in Martin. A description of a free state machine compiler, which converts a textual state transition table into the classes necessary to implement the STATE pattern can also be found there.[18]

References

[1]Andersen, J. D., *Introduction to Flight,* 3rd ed., McGraw Hill, New York, 1989.

[2]Anonymous, *Pilot's Handbook of Aeronautical Knowledge AC 61-23B*, Federal Aviation Administration. Oklahoma City, OK, Aviation Supplies and Academics, Renton, WA, 1980.

[3]Anonymous, *Flight Training Manual AC 61-21A*, U.S. Government Printing Office, Washington, DC, 1980.

[4]Anonymous, *Information Manual Model 172RG*, Cessna Aircraft Company, Wichita, KS, 1980.

[5]Anonymous, *Private Pilot Manual*, Jeppesen Sanderson Training Products, Englewood, CO, 1997.

[6]Anonymous, "Technical issues involved in propeller system selection for your kitplane," Hartzell Propeller, Inc. Piqua, OH, 2002.

[7]*Aeronautical Information Manual, 2003*, ASA Press, Seattle, WA.

[8]Breslin, J. P., and P. Andersen, *Hydrodynamics of Ship Propellers*, Cambridge Ocean Technology Series 3, Cambridge Univ. Press. Cambridge, England, UK, 1994.

[9]*GNS 430 Users' Manual*, Garmin Avionics, Olathe, KS.

[10]Kershner, W. K., *The Flight Instructor's Manual*, 2nd ed., Iowa State Univ. Press, Ames, IA, 1981.

[11]Laitone, E. V., "Fixed Pitch Propeller Selection for Light Airplanes," *Journal of Aircraft*, Vol. 37, No. 3, 1999, pp. 390-395.

[12]Laplante, P. A., *Real-Time Systems Design and Analysis*, 3rd ed., IEEE Press/John Wiley & Sons, New York, 2003.

[13]Lesley, E. P., "Test of an Adjustable Pitch Model Propeller at Four Blade Settings," TN No. 333. National Advisory Committee for Aeronautics. Washington, DC, 1930.

[14]Martin, D., "Cruising the country with auto power and a really interesting prop" *Kitplanes*, Vol. 19, No. 2, 2002, p.4.

[15]US House of Representatives Subcommittee on Aviation Hearing on Air Traffic Congestion in the New York City Area; 16 June 2001.

[16]Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, New York, 1994.

[17]Laplante, P. A., *Software Engineering for Imaging Systems*, CRC Press, Boca Raton, FL, 2003.

[18]Martin, R. C., *Agile Software Development: Principles, Patterns, and Practices*, Prentice-Hall, Upper Saddle River, NJ, 2003.