GoBosh G700S Flight Simulator

Christopher Dlugolinksi, Robert Gysi, Joseph Munera, Lewis Vail

School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — This paper will discuss the design and construction of the GoBosh 700S flight simulator. The GoBosh 700S flight simulator is designed to provide an realistic as possible and fun simulated flight experience for every experienced and aspiring pilot. The emphasis and objective of this project is to interface various electrical devices, stepper motors, and slide potentiometers to the X-Plane 9 simulator. This simulator consists of six analog flight instruments and flight controls similar to what is found in the actual GoBosh 700S aircraft. Our project utilizes the capabilities of the USB interface, logical design, and software programming.**

***Index Terms* — Flight simulation, systems integration, USB device development, Aerospace Simulation, Aircraft instrumentation, Stepper Motors, Simulation Software.**

I. Introduction

The scope of this project has been to develop a cockpit-based flight simulator based on the popular GoBosh G700S. This is an US import variant of the Polish-built Aero AT-4 Light Sport Aircraft. The work is being performed for a local flight instructor, Mr. Dave Kotick on behalf of his business Grizzly Aviation as well as GoBosh.

With the objective defined, the purpose of the simulator is to be utilized in two possible roles. The first is for it to be utilized as an instrument through which sales of this aircraft can be increased, by providing an interactive environment which the plane's ease of use can be experienced firsthand. The second role for which this is intended to be used is to increase the client base for Grizzly Aviation, the flight instruction business Mr. Kotick runs. Here the simulator would again demonstrate the ease of flying for the LSA category of aircraft and act as a vehicle which to inform potential student-pilots the short amount of flight training an LSA license requires. The third possible use is for it to be used in ground-based flight training scenarios. This however will require the system to be tested and coupled with a USB key from Laminar Research, the makers of *X-Plane*.

This project is being funded by the project sponsor in its entirety and will be delivered to the sponsor at the completion of the course. In addition, all documentation and source code files in addition to compiled programs will be included.

II. Specification and Requirements

There are two common commercially available flight simulation applications available to consumers: Microsoft Flight Simulator X and Laminar Research X-Plane 9.

When the idea of creating a flight simulator came up as a topic for a senior design project it sounded like a fun project that could have many different types of challenges. A simulator is an imitation of something real, and the simulator that we were asked to build was for a real product, the GoBosh 700s aircraft. The aircraft is used for training students on how to fly and the simulation would make that task easier, and also make the student a little more comfortable with his/her ability as a pilot before they actually fly the real aircraft.

The task sounded like fun and to add to the fun there was going to be an actual aircraft fuselage that was going to be part of our design. The fuselage will come equipped with all the working parts and have room for all the gauges that are to be simulated in the aircraft. The main idea for our senior design project is to create a simulation out of this GoBosh 700s aircraft making it as realistic as possible.

Some of the key features of the simulation are the actual use of the aircrafts original flight controls. We will be using the flight controls that come in the aircraft and just mounting them with sensors so that we can measure the inputs and use them in our simulation software. There is also going to interactive switches and gauges that will allow the user to get the actual interface that you would see if you were really flying the GoBosh. It will also include the use of three screens to give the user the feeling of the 120 degree field of view that you would have if you were flying the actual aircraft. An added feature of the simulation is that we will include databases for the local airports of the surrounding areas so that pilots form this area can notice landmarks while flying the simulator. All of the features listed above will give a good simulation of the GoBosh 700s that will give the user a better understanding of how the actual plane will react when in the air.

This paper describes how each of the features listed above were researched and how they will be implemented, including a budget and a timeline to finishing the simulator for the Sun „n Fun airshow and general aviation

conference that is to be held at the Lakeland Linder Regional Airport, in Lakeland, FL during the second full week of April. Unfortunately, due to production issues at the Aero Sp.zoo plant in Poland, we were not able to receive our cockpit before the end of the semester. Therefore, the requirement to go to Sun 'n Fun as part of the GoBosh Aviation exhibit was dropped by our project sponsor.

In order to make the simulator as real as possible we needed to pinpoint the parts of the activity of flying the GoBosh that were essential. This is where we gathered our requirements. We held several meetings with our sponsor and his flight instructor colleagues where we developed a requirements matrix for all of our hardware and software. The requirements are listed in the requirements chapter in this paper. These requirements led us to our budget, which was originally given to us by the sponsor, and we needed to make that work doing research on many of the components that we were going to use. Our budget was also affected by the timeline that we needed to deal with as well as the hardware interfaces that had to be designed. The design of the system is discussed in the design section. It lists all the reasons why we decided to create the system the way we are and how we are going to create it. We were held to a very tight schedule, and outside of the cockpit not arriving we were able to meet most of our scheduled objectives.

As we are now at the conclusion phase of our project, we have come to have a greater understanding of the challenges in designing, building and implementing a flight simulator. Each of us has expanded our knowledge in software, electronics, mechanical systems, in addition to 3D modeling skills that were necessary to accurately model the GoBosh G700S for use in X-Plane.

III. Application Components

A. X-Plane 9

In order to be able to realistically portray the GoBosh G700S/Aero AT-4 in a virtual environment it is critical to pick the correct flight simulation software package. Currently, there are two competing simulators on the market available to end-users: Microsoft® Flight Simulator X (FSX) and Laminar Research® X-Plane 9.4. To the average end user, they are fairly similar applications, although for our purposes only one really stands out.

X-Plane 9.4 incorporates the most accurate methods of modeling an aircraft in virtual environment by actually taking the shape of the aircraft and model the aircraft through the use of blade element theory. This technique means that the software sections the aircraft model into multiple small "blades" to calculate the forces on these points. This gives a realistic physics model of the aircraft, which means if you model a solid cube with no aerodynamic properties, all it is going to do is sit on the ground. Microsoft FSX takes a different approach and instead of breaking down the aircraft into sections and then modeling it in a physics engine, it receives all of its properties through a configuration file, meaning that one could make a very un-aerodynamic shape (the cube mentioned earlier) fly through the air at whatever speed as long as the proper variables were set in the aircraft configuration file.

X-Plane also includes a model editor in order to create aircraft that will fly in the game. FSX does not include this feature and requires expensive third-party applications in addition to manually editing a configuration file.

Additionally, the requirement for a realistic looking environment was also added by our project sponsor. While accurate flight models are important, it was also deemed important that the simulation be visually appealing due to its projected use as a demonstration device. Both flight simulators have similar levels of graphic details, with each having a leg up on the other. Microsoft FSX for example includes major landmarks in major cities, so places like Amway Arena, the SunTrust Building downtown or the EPCOT ball are included in the scenary. Also with FSX, all airports have modeled terminals with the majors ones being relatively accurate. X-Plane 9.4 has neither of these level of details, but it does have more detailed airfields leading it to be the better selection as a demonstration device and as a possible flight training device.

B. FTDI USB Chip

The FTDI chip has the capability to interface with any microcontroller, being the USB portion of the communication for the microcontroller without the need to code the interface for USB communications. This was great but added to the cost of the gauges. After even more research on the FTDI chip it supposedly has a special mode that allows you to directly control the I/O pins on the FTDI chip. This would give us the ability to not really worry about USB protocol and it is able to remove our need for a microcontroller. It can remove our microcontroller if you look at what the microcontroller was going to do for us, it was going to just control the gauges (stepper and servo) for us using its output pins. This is simply a turning on and off of a few port lines or sending a pulse of a certain length. This can be done with any I/O ports and the FTDI chip in its special mode should be able to do this. It can read and write to the lines directly giving us the ability to write the control for the device right into our software.

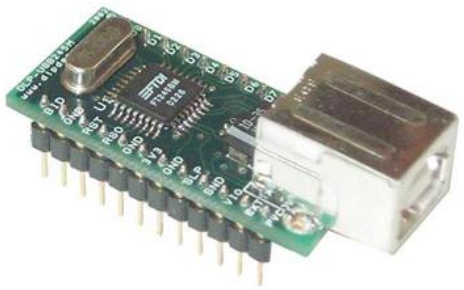

Fig. 1. FTDI Development board that was selected and used for the USB communication chipset for all of our I/O needs.

TABLE I
SUMMARY OF FTDI SPECIFICATIONS

Microcontroller	FTDI FT245BM
Dev Board	FTDI
Cost	Dev Board $30 Chip $5.00
Usb driver	Free from FTDI to download and no programming on chip unless really necessary. Www.futurlec.com
speed	USB 1.1 or USB 2.0 (compatible)
Examples	http://electronicdesign.com/Articles/Index.cfm?AD=1&ArticleID=16125
Memory	External EEPROM
Memory RAM	
I/O	8 pin
Languages	any
Voltage and Current Ratings	All usb self contained may need to do something for control of external parts

We will be using the FT245BL chip it has 8 I/O pins and direct connection capable to talk over the USB line as needed. It also has the capability to have EEProm connected up to the chip. This will allow us or anyone who wants to make a gauge to make one and we could easily identify them through a description or a PID or VID that we will assign. Our program will contain many different types of gauge control capability and adding a new gauge will only take setting the PID or VID to that type of gauge and we will have control for that gauge based on the real gauges activity, and it will be given data from the simulation that is running it.

The cost of the FTDI chip will also reduce our overall cost of the gauges. During the initial research a microcontroller was thought to be needed and they cost $10 to $15 dollars for the chips we were looking at. The FTDI chip is only $5 a chip and needs only a few external components.

One of the most important parts of the design will be with the power consumption. The USB port on a computer can supply 5 volts and 500mA as we have already stated. The use of this to drive a motor of any kind is pushing the power of the USB port to the limit, not to mention the ability of the motor to push current back at the port which can kill the port all together. So we will need to either use a buffer or some sort of power supply for each controlled gauge. Equation 1 below shows our required data speed based on our motor selection while equation 2 shows the max motor speed that the FTDI chip will be capable of controlling, given its max transfer rate.

$$200\frac{steps}{rev} * \frac{8\ bits}{.5\ step} = 3200\ bits/rev \quad (1)$$

and

$$\frac{8{,}000\frac{kb}{s}}{3.2\frac{kb}{rev}} = 2500\frac{rev}{s} \quad (2)$$

C. Stepper Motors

We choose the RKI-1128 stepper motor to drive all the flight instruments. This is a low torque small size stepper motor with a five wire interface. The stepper motor can be used in unipolar & bipolar mode for small applications.

Features

1) 1.8 degree step angle – 200 steps per revolution
2) Holes on four corners for mounting
3) 12 V, 150 ma
4) Weighs 160 gm
5) Size 38x38x30 mm

The more important feature listed above, to us, is the 1.8 degree step angle that this motor supports. Additionally we can do half steps at 0.9 degrees giving us 400 half steps per revolution. This ultimately gives us finer control over the position of our instruments. Unfortunately this also leads to the appearance that under some circumstances an instrument may appear choppy. However, this was the best motor that could be acquired within our budget.

D. Flight Instruments

The core of our simulator will be X-Plane's simulation software. Therefore, all of our software will interface with X-Plane via its plug-in API. This API gives us access to all of the functions and variables that we will need. What we need to decide in implementing our interfaces is how many

plug-ins to use and how frequently we want to refresh our values.

With regards to how many plug-ins to use, at one end of the spectrum we could have a separate plug-in for every interface as to allow maximum parallelism. This would make each plug-in a lot simpler and since a lot of the interfaces are very similar, this may simplify the whole design. It would also make the design more modular so that future development could be done without having to change old code. The problem of having so many different plugins is that it is likely that they would all be competing for the same USB hardware. Also, although it would be nice to be able to add or remove certain hardware components by just adding or removing their plugins, this may be more cumbersome in the long run than just having one plugin.

By interfacing with all the gauges using a single plug-in we have a little more control and can step through all of the interfaces in series. This way, whenever it is time to refresh the data on the I/O devices, we can make sure it all happens at the same time. You could also keep the desired modularity by allowing the user to configure which devices to use or by having the software sense all appropriate I/O devices at initialization. Also, by using one global plugin, it makes it much easier to share and recycle code.

After weighing all our options we decided to use two plugins for all of our devices, one for input (all the control devices) and one for output (all of the flight instruments). Each device will plug into the commuter with its own designated USB cord and will be controlled by its respective plugin. The reason we chose to group them in this fashion was because the design for all of the control devices are very similar as is the design for all for all of the flight instruments. Because all of the hardware interfaces into the simulation fall into one of these two categories, much of the code is reused in each plugin. As with all of our I/O devices, the controls will be integrating with X-plane via the X-Plane Plugin Manager. The plugin manager is a dynamically linked library that handles all the communication between the plugins and X-plane.56 The main loop for our control devices will be as follows. For each control, we first sample the input. Most of our controls will give us a 10-bit digital signal, which will need to be truncated to an 8-bit signal so that we can use it in software. Once we have this data we will need to turn it into a format that can be assigned to one of the X-Plane variables according to the plugin API. Finally, once the appropriate data reference is populated with the new value, X-Plane will respond appropriately.

It is very similar to the control interface architecture except it is backwards. With the gauge plugin, the first step in the main loop is to look up the appropriate data reference. Next the data must be translated into something we can use to drive the gauge. Finally the appropriate value is sent out to the gauges, which will turn to display the current instrument readings. In the case of stepper motor based gauges, an aspect of this final step will be a gauge driving loop that steps the motor through the appropriate amount of iterations to get the needle in the right position.

The other major consideration with regards to how we integrate our I/O devices into X-Plane is how often we update the X-Plane values. In the case of the instruments, this would be moving the needle into the right location, and in the case of the controls, this would be updating the variables as the user moves the controls. With regards to the instruments, the limiting factor here would most likely be the speed of the motors. Too fast and the needles may jump around. Too slow and the needles movements may be too choppy. A good starting point would be thirty times a second to correspond with the frame rate but testing will have to be done to optimize it. With regards to the controls, the limiting factor would most likely be the human element. Once again, it would probably be best to start with the frame rate and increase the loop time until it is optimal.

E. Flight Controls

One of the most critical components of this project is the joystick controller that will be designed for the flight simulator. In most small general aviation aircraft the actual stick is actually connected to the flight surfaces through the use of connecting wires or push-pull rods. For this simulator we will need to simulate this same operation digitally and then pass the data into the simulator computer so that the appropriate command is executed on the screen. In order to tackle this problem we need to first understand that in order to create this control stick we will need to work with two directions: the X-axis and the Y-axis. Connected to these two axes are potentiometers which as the stick is moved, change in resistance which allows one to map when at a particular output voltage a certain position has been reached. The original implementation for this was going to include the use of the stick in the aircraft. However, due to the cockpit not arriving we have built a test rig to validate our electrical design.

We also need to provide input from the rudder pedals that would normally be in the foot well of the cockpit. In this application, the pedals will be able to be used to steer the rudder on the tail of the aircraft just as a functioning aircraft. Again, a test rig will be constructed using wood to validate our electrical design. In the pedal test rig we will have a single slide potentiometer located between the left and right pedals. The slide pot will be connected to a

lever bar that is connected to the pedals, where once one pedal is pushed forward the other is pushed back. As this motion occurs, the lever arm will cause the slide pot to move forwards or backwards giving us a value which has been mapped into our X-Plane plug in.

The final flight control to be discussed is the throttle. The throttle is the simplest to design from a mechanical standpoint as all we need is a rod connected to our slide pot arm. We have accomplished this by attaching a bracket to the slide pot which is connected to a rod that extends through our instrument panel. A mechanical drawing of the throttle is shown in the figure below.

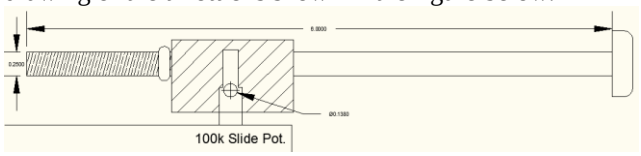

Fig. 2. Mechanical drawing of Throttle implementation.

F. Power Supply

The power needed to run one stepper motor is shown in the equation below:

$$12\ volt * 150\ mA = 1.8\ W \quad (3)$$

As we are using 8 stepper motors we have calculated that we need a total power requirement of 14.4W. For our power supply we have used a standard computer ATX power supply to power our motors. This allowed us to save time on development and building a power supply, when the ATX power supply provides us with everything we need: 12V DC, 5V DC, and multiple leads for us to connect our boards to. This also allows us to maintain our goal of expandability. When the end user wants to add an additional gauge or other control to the simulator all that will be required is a free connector on the ATX power supply and the purchase of an inexpensive 4-pin molex connector. Indicator lights and slide pot control will run within the USB power rating limits.

G. Software

The development for this project is as modular as possible for future development. One plug-in is used bot for input (controls) and output instruments). The plug-ins are completely parametric driven. To make the simulation as realistic as possible we aimed for 30 fps for both sampling inputs and driving outputs.

The plug-in communicates with X-Plane through the X-Plane Plug-in Manager (XPLM) via the plug-in API. For the controls the position is read, translated to X-Plane Value, and a new value is written back through XPLM. The flight instruments read the X-Plane values, translate these values to number of steps, and then step through the difference.

All the software is written in the C++ programming language. This is one of the programming languages supported by the X-Plane SDK. Internal plugin logic is all implemented in one thread to avoid any read/write hazards within the XPLM.

Conversion between X-Plane values and number of steps is done using stored minimum and maximum values for each device. This data is stored in a flat config file. One config file is used per FTDI chip. The control plug-in and instrument plug-in use much of the same code to ease development.

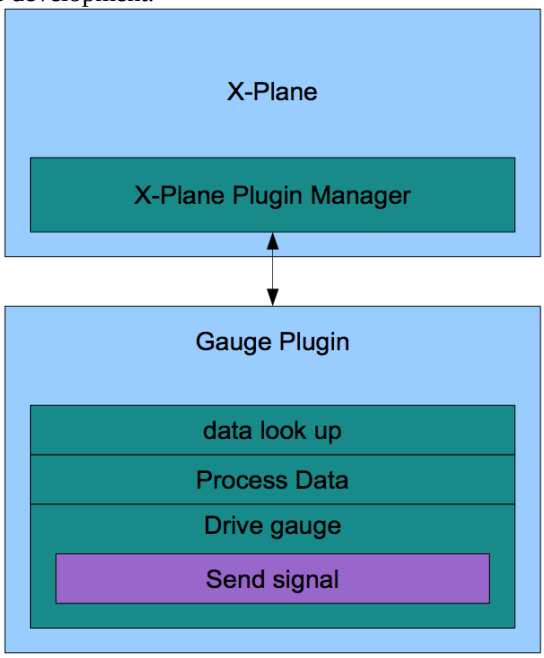

Fig. 3. Gauge Architecture Diagram

IV. Design And Implementation

The Design of the boards was simple for both the control and gauge boards use a USB interface (FTDI chip) with 8 I/O pins that can control the motors that are connected to them. The control circuitry for the motors is buffered by the CD4050 chip in order to make the signal have a solid on off input. The driver is using 4 NPN transistors with Diode protection from the motor. There is a photo diode circuit that is used to control the position of the motor upon start up. This is fed through a comparator to determine when the photo diode is sensing no light and will turn on the 8^{th} pin that we are checking with the USB chip.

The control circuit is controlled with again the 8^{th} I/O pin to change which of the A/D converters are being used. Then the USB reads the input and can set the information in the plug-in for X-Plane as desired through a simple configuration file.

The Implementation of the circuit boards came from the fact that our sponsor didn't want us to use printed circuit boards in the design as testing would prove that our design may need to be changed and that would take a while if that happened, and using prototype boards allowed us to change the circuit on the fly until it met our needs. There were sockets used in case any of the components went bad during testing and we needed to replace them.

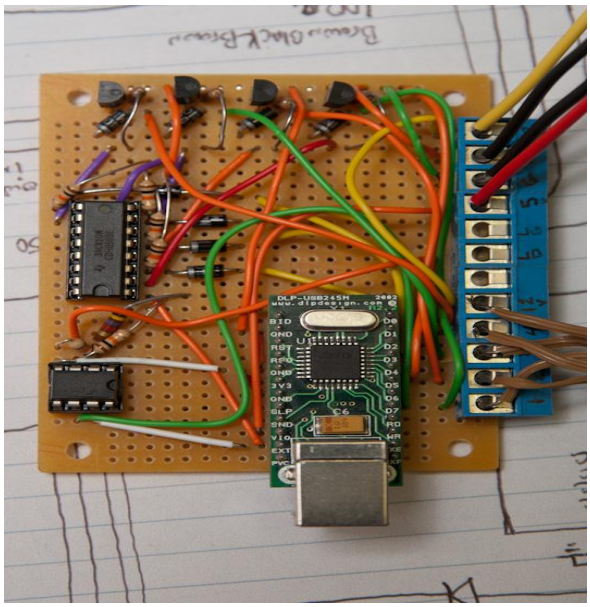

Fig. 4. Completed circuit board for Flight Instruments.

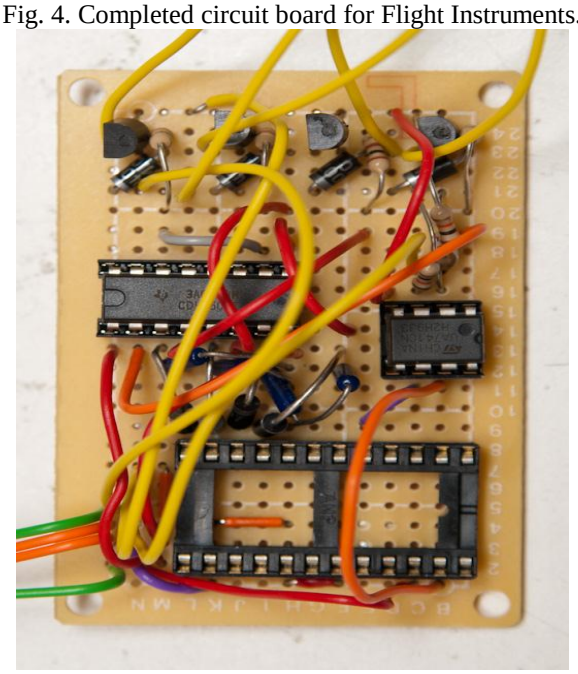

Fig. 5 Completed Circuit board for Flight Instruments (Initial Revision)

Outside of the electrical implementation of the boards, for each of the instruments, there was much to be done for the mechanical aspects. When we purchased our stepper motors we were not aware that the shafts were not slotted or designed to secure an extension to. We first needed to overcome this obstacle in order to extend that shaft of the gauge to the height of the faceplate deck. To do this we used a hollow nylon tube cut into section about .25-.3" long and then slide on the shaft of the motor. This was a very snug fit and held well. After fitting our nylon tubing we used sandpaper to remove some of the nylon so that a aluminum tube which had been cut to the appropriate length could securely fit over top. This created our shaft to the faceplate. The only detail remaining was the connection of the shaft to the needle. This is delicate as the connection for the needles is a very small piece of metal that is connected to a fine gear in the actual gauge. We determined that we would use an epoxy to secure our needle and heading indicator head to the shaft securely.

After experimenting with several liquid based epoxies, we found that LOCTITE Epoxy Putty was the best solution to filling the top of the shaft. It has a setting time of 5 minutes, which is enough to position our needle into the putty. Although it is supposed to cure in 30 minutes, we found that the conditions of the senior design lab caused it to not fully cure during that time and came to the conclusion that the safest bet for ensuring that it was finished was to let each gauge set for 24 hours after assembling. This putty also hardens to the point where it is possible to drill into it should the need arise.

Moving on to the structural aspects of the gauges, lets first review the major components common to the majority of the gauges:

1) At least one 200 step/rev, 1.8 degrees per step unipolar stepper motor (Turn coordinator and Artifical Horizon require two due to each being gyro-based instruments.
2) Sheet Aluminum
3) Clear Acrylic
4) OWCP 4537 CDS Photocell
5) FTDI FT245BL USB Communication Board

Each instrument aside from artificial horizon used each of these components assembled on a variety of decks to form our gauges. Starting at the top and working our way down we have the Faceplate deck with Lens Separator then to the motor deck with houses the motor and then finally to the circuit deck which provides a standoff for the circuitry and protects our components from being disconnected. Additionally it also provides separation and protection of the stepper motor and ensures that the should a motor fail, any damage would not occur to the deck outside of possible electrical damage.

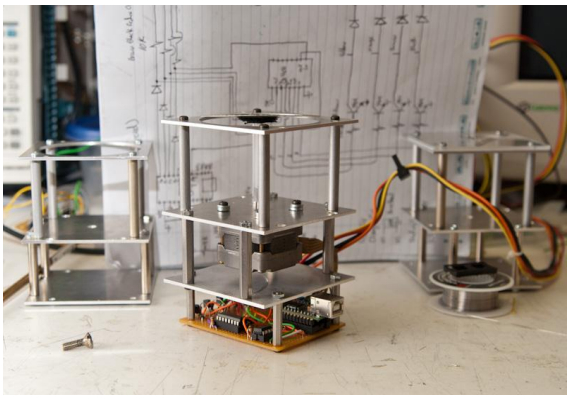

Fig. 6. Side view showing the various decks while three gauges (heading indicator in front) are under assembly.

Precision was a key requirement in building the decks that make up our gauges. We first attempted to cut our own aluminum, but this resulted in pieces that looked chewed up. Also we had the problem of getting our drill holes to line up every time despite using a drill press and securing our components. Therefore we contacted an individual with access to a machine shop. We supplied the materials and the individual fabricated all of the components that we needed for our gauges free of charge. This saved us a significant amount of money as we were quoted rates in the ball park of $35/hr for the use of the UCF CECS Machine Shop and the UCF Physics Department Machine Shop.

While it would be impractical to put all of the drawings required to manufacture these parts, we have included two scaled-down versions of the two most common decks, the motor deck and faceplate/lens separator deck in figure 8.

This assembly/manufacturing process held true for four of our gauges (Airspeed, Vertical Speed, Altimeter, and Heading), the turn coordinator and artificial horizon were different beasts to tame. For each of these gauges operated based on an internal gyro and the turn coordinator also has a liquid based level on its faceplate. In order to implement these gauges, two stepper motors are required.

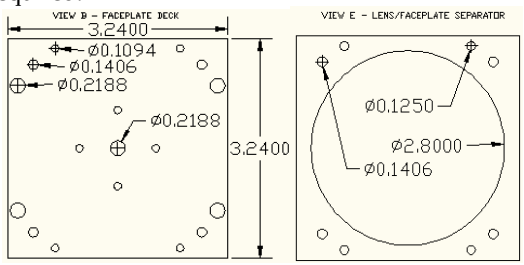

Fig. 7. Common instrument deck fabrication drawings.

Of the two, the Turn Coordinator is the simplest, due to the ability to actually reuse some of the design for the other gauges. In order to start, we modified the motor deck by adding a second motor. Each are placed next to each other with one shaft on the lower half of the deck and another on the upper half. In order to reach the indicator that needs to be in the center of the instrument faceplate, we have used gears to bridge the distance between the two. For the liquid-based level we have simply attached a black flag to the motor shaft and it moves in the space cut out for that instrument. The turn coordinator will require the use of two FTDI boards to control each motor.

Fig. 8. Turn Coordinator Faceplate

The artificial horizon is the most challenging. There are several kinks to work out, but we expect that we will be able to modify the existing instrument that we received by replacing the gyros with our stepper motors. This method has been used before to modify an existing artificial horizon and as a result we will be following these plans to achieve a result. As for the electronics to control this instrument we will require two control boards each with separate FTDI chips. As a result in our configuration file we will need to treat each motor as a separate gauge.

V. Conclusion

The design and prototyping of the GoBosh 700S simulator came along as well as any design process could be expected to go. Due to the fact that the aircraft fuselage did not arrive as expected some major changes were necessary to the design developed earlier in the year. This resulted in us not being able to interface with actual flight controls and the resulting development of test rigs to validate our electrical design. We were able to fortunately able to receive a instrument panel cutout in addition to actual gauges that we were able to use for the faceplates of our simulated instruments. Additionally, due to the cockpit not arriving and the goal of going to Sun ‘n Fun, was dropped, The electrical and computer engineering aspects of this project came along smoothly and were amongst the easier aspects of developing this project and resulted with all devices and designs working as expected.

Some of the most difficult aspects of this project turned out to be the mechanical engineering considerations involved in building six aircraft flight instruments and the flight controls stick and pedals.

ACKNOWLEDGEMENT

We would like to thank Dave Kotick of Grizzly Aviation, and Dave Graham of GoBosh Aviation for their partnership in this project, Dr. Samuel Richie for his coordination of the senior design courses and guidance during this process, as well as Dr. Ladislau Bölöni, Dr. Zhihua Qu and Dr. David Workman for their time to be able to sit on the project review committee for our design.

PROJECT ENGINEERS

CHRISTOPHER DLUGOLINKSI is a senior at the University of Central Florida that will graduate with a B.S. in Computer Engineering (May 2010). He also has a degree in Electrical Engineering Technology from the University of Cincinnati. He is now working for a simulation company and plans on continuing to forward his career after graduation.

ROBERT GYSI is a senior at the University of Central Florida and will graduate in May 2010 with a B.S. in Electrical Engineering. He is currently employed by ARINC Engineering Services, LLC in Panama City Beach, FL as an Engineering Intern and plans to continue his career in a Electrical or Systems Engineering role in support of DoD programs.

JOSEPH MUNERA is a senior engineering student at the University of Central Florida. He will graduate in May 2010 with a degree in Computer Engineering (B.S.Cp.E). He plans to continue his education in the field of computer engineering, as well as beginning a career in industry.

LEWIS VAIL is graduating from the University of Central Florida with a degree in Computer engineering in May of 2010. He currently works at PEO STRI as systems engineering co-op student and will be pursuing a career in software development upon graduation.

REFERENCES

1. [1] [DLP-USB245M User Manual] DLP-USB245M USB to FIFO Parallel Interface Module Online. <http://www.ftdichip.com/Documents/DataSheets/DLP/dlp-usb245m13.pdf >

2. [X-Plane forum] Home of the X-Plane flightsim community. Online. <http://x-plane.org/>

3. [X-Plane SDK] X-Plane SDK. Online. <http://www.xsquawkbox.net/xpsdk/mediawiki/Main_Page>

4. [Simkits] Simkits Flight Simulator Hardware. Online. <http://www.simkits.com/>

5. [Flight Illusion] Flight IllusionOnline. <http://www.flightillusion.com/>

6. [Flight Instruments] Flight Instruments *Wikipedia* Online.<http://en.wikipedia.org/wiki/Flight_instruments>