GoBosh G700S Flight Simulator

Senior Design II – Spring 2010 – Group 11

Christopher Dlugolinksi, Robert Gysi, Joseph Munera, Lewis Vail

Project Sponsor: Mr. Dave Kotick, Grizzly Aviation

5/3/2010

Table of Contents

Chapter 1
1.1 Executive Summary

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.

Originally, an actual aircraft fuselage of this aircraft type was going to be part of our design. The fuselage would have come equipped with all the working control inputs (pedals, stick, and throttle) as well the instrument panel for our simulated instruments. Unfortunately, due to supplier issues between the factory (Aero Sp. z o.o.) and the US importer (GoBosh Aviation), we did not receive a cockpit as intended. Although the cockpit was not received, GoBosh did come through and deliver us an instrument panel cutout and several gauges to use for parts to add what realism we could to our simulator. Even with this limitations, our goal was to still to make an as realistic as possible simulator with the materials and resources we had available to us.

Originally one of the key features for this simulator was the actual use of the aircraft's original flight controls. However, with above mentioned supply issue, we instead were tasked with implementing our electronics design to test rigs to validate our design work. The electronics design did not change at all and still utilized potentiometers (slide and turn) coupled with Analog-to-Digital Converters feeding into one of our FTDI USB communication chips. From a mechanical perspective, we needed to design in a short amount of time our controls for implementation with our simulator. The implementation of these controls is discussed in a later chapter. The other half of the physical implementation requirements of this simulator included the design and construction of a set of simulated "six-pack" flight instruments. This includes the airspeed indicator, turn coordinator, vertical speed indicator, altimeter, artificial horizon and directional gyro (compass). These will all take their values from X-Plane and will display the same as if they were the simulated gauges on the computer screen.

Another aspect of our original plan was to implement a visual projection system along with computer hardware to power our simulator. This was going to be accomplished through the use of three 24" monitors to give the user a 120° field of view out of the cockpit. Since the actual cockpit was not to arrive in time, a decision was reached with our sponsor to not take the simulator to Sun 'n Fun in Lakeland and as a result to not purchase the associated computer components at this time. Even with this limitation we were able to demonstrate the ability of the software to output 120° field of view onto a single 24" monitor. Additionally, for our demonstration, we utilized our primary development machine to power all

of our simulated controls, instruments and visual output without any performance or other issues.

Additionally, another 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. These databases are all included in X-Plane, although for higher realism 3rd-party scenery can be purchased to increase realism of the local area. All of the features listed above will give a good simulation of the GoBosh 700s that will give the user a better understanding of this aircraft reacts inflight and the ease at which one can fly this aircraft.

This paper describes how each of the features listed above were researched how they were implemented, and the results of our testing. Also we will cover some administrative information, including our budget and our original project schedule predictions. Additionally, in the Appendices of this paper you can find several of our early project deliverables and research information.

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. From several meetings with our sponsor and two of his fellow aviators we gathered and formulated our project requirements. This allowed us to develop requirements for our hardware and software components that we needed to interface with one another. These requirements are listed in the requirements chapter in this paper and are broken down by functional area. Also located in that table is the status of the implementation of our project requirements, since due to some decisions between us and our sponsor, several requirements were not met. From our requirements we formulated our budget. We were given an initial budget limitation of $1500 by our sponsor, which would not have been enough for the project if we had purchased the computer. Since we did not purchase a computer our budget came in at under half of the original. This is explained further in the budget chapter of this paper. Additionally, the design of our individual of components is discussed in the chapter on design. It lists all the reasons why we decided to create the system the way we did as well as provide the basics for implementing our designs. After discussing our design we will discuss the implementation of our designs including the testing of our system to ensure that it meets our project requirements.

Overall, the project was a tough challenge, but we feel that we all now have a greater understanding of the engineering process as well as effort that it takes to create a functioning simulator. The effort required many hours of work in the senior design lab, but overall it is worth it when in the end a functional product works during the project demonstration.

Chapter 2
2.1 Project Objective

The objective of this project is simple: to build a simulator around a GoBosh G700S/Aero AT-4 Light Sport Aircraft. We were tasked on this project by Mr. Dave Kotick, a local flight instructor based out of the Orlando-Apopka Airport (X04) near Apopka, FL who has sponsored this project through his business Grizzly Aviation.

As defined in our first meeting with Mr. Kotick, the initial purpose of this project was to produce a flight simulator that is not necessarily for flight training, but for demonstrations of this aircraft. In addition to training of aspiring Light Sport Aircraft pilots, his business is also a representative of the US importer of the aircraft we simulated: GoBosh Aviation (the aircraft are manufactured in Poland by a company known as Aero Sp. z o. o. as the Aero AT-4). Because of this, he frequently travels to airshows and general aviation conferences to demonstrate the aircraft and find potential buyers or those who are interested in potentially earning their LSA pilot's license.

One of these events in which he participates in with GoBosh Aviation, is the Sun 'n Fun airshow and general aviation conference which was held in Lakeland, FL at the Lakeland Linder Regional Airport. This year the event was held during the second week of April (4/13-4/18), and we were originally to be part of the exhibits with our simulator at the show. Unfortunately, due to not receiving our cockpit from Aero in Poland, a decision was made to not demonstrate at the airshow.

While making it to Sun 'n Fun was considered our ultimate objective for this project, we also had several side objectives as well that this project needed to meet. In addition to being developed for demonstrations at aviation shows where the individuals in attendance are familiar with aircraft or at least flying one, it was also designed to be taken to a variety of other shows in the future once handed over to Mr. Kotick. One example would be the Orlando Home and Boat show, where people who may have never considered becoming a light sport aircraft pilot or purchasing a light sport aircraft could be exposed. This serves the purpose of education, as many people assume that they could never fly due to the fact that flight lessons are expensive and time consuming, which is the opposite of the aircraft we are simulating. Because of this, the purpose at these shows is to show the relative ease that exists to pilot one of these aircraft. Although we were unable to get the cockpit in time, we were able to complete all of the necessary software and hardware without it and this can be put into the real cockpit once obtained. This would allow our sponsor to ultimately meet these original goals.

Another additional objective for this project was for it to potentially be used in ground based flight training. Pilots are allowed to use a limited number of ground based flight simulator training in lieu of actual time in the cockpit. With this in mind we have kept this as an open option through the development of our

requirements and through our design. In fact as later discussed in Section 2.2.1.1, we see that it actually is as simple as plugging in a special USB key into a computer running the simulation software (X-Plane). While providing the key is not within the scope of this project, the ability to do this allows our sponsor to add to the simulator once we have finished.

2.2 Specifications/Requirements

We have broken down our requirements into two parts; our software requirements and our hardware requirements. In each subsection, we break down the development of our requirements and explain why we chose a particular option over the other and ultimately which devices or software were ultimately the one that met our requirements.

2.2.1 Software Requirements

The software requirements will be broken down into three sections: the requirements for the flight simulator, the requirements for the aircraft model we developed for the flight simulator, and the microcontroller/FTDI chip control software requirements.

2.2.1.1 Flight Simulator Requirements

In order to be able to realistically portray the GoBosh G700S/Aero AT-4 in a virtual environment it was 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 the previously mentioned cube would be able to fly with the proper variables.

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.

While X-Plane takes a victory when it comes to modeling, it does not when it comes to scenery. Scenery in FSX is much more detailed including airport terminals, landmarks, towers, and major population centers. X-Plane 9.4 does

not include any of these and instead uses random auto-generated scenery to populate the world. While a city such as Apopka does not need major detail, cities like New York City miss all the important landmarks a pilot would use to fly. However, this feature is not a primary requirement and is considered part of the "entertainment-value" of the simulator and where the need arises for detailed local scenery it can be developed or purchased from third-party developers.

There is also the possibility, down the road, that this flight simulator could be used for ground based flight training. Currently only X-Plane is certified by the FAA when coupled with a $500 USB key, which guarantees frame rates and output data. However, the consumer version does allow you to set a frame rate limit and it will scale the simulation graphics settings in order to match this rate. For the purposes of this simulator, a minimum of 30 frames per second was deemed necessary. In addition, this is part of the commitment Laminar Research has made to the X-Plane family including the fact that there are regular updates of the software. This is compared to FSX which as of January 2009, has had a stop in development of future versions due to the closing of the Microsoft ACES studio.

All of these items together show that for this simulator, the use of X-Plane 9.4 would be most advantageous to use. A further exploration of the requirements and results of a side-by-side comparison lie in Table 2-1 below with explanations given in the proceeding paragraphs.

Table 2-1 Environmental Aspects

No.	Item/Description	Req. No.	FSX	X-Plane 9
1.	Inclusion of Majority of Airports Worldwide	S1.B	Yes	Yes
2.	Detailed Realistic Scenery	S1.A	Yes	Yes
2a.	Accurately detailed major cities and landmarks	S1.A	Yes	No
3.	Realistic Weather Conditions	S3.A	Yes	Yes
3a.	Real-World Weather	S3.A	Yes	Yes
4.	AI Aircraft in the virtual world	S3.E	Yes	No
5.	Deliver a constant 30 FPS	S6	No	Yes

While comparing the environment simulated in both of the software options we find that on the surface the two seem similar. They both include a large number of airports worldwide (X-Plane even includes a few that FSX omits), but the major difference is that in X-Plane airports are just runways, taxiways, and aprons. There are no buildings on airport property at any airports in the simulator, not even at airports such as John F. Kennedy International Airport (KJFK) in New York City or Orlando International Airport (KMCO). Microsoft's Flight Simulator does have these major airports accurately modeled and where there isn't an actual model, automatically generated buildings are displayed along with other

support buildings. However, while this is a feature that is nice to have, the whole purpose is actually to have the plane flying, not taxiing to a commercial aircraft gate at a terminal. At the same time, Microsoft FSX also includes more detailed scenery overall. In order to provide some realism, the modelers at Microsoft decided to model major landmarks and major population centers. That means when flying over Disney World you fly over the EPCOT attraction Spaceship Earth or if flying over New York City, the skyline of Manhattan is present. Now, this isn't to say that X-Plane does not have decent scenery installed. In fact in some areas it does appear to have a decent level of detail, however most areas do appear to be just randomly generated entirely. When you realize that X-Plane is not an entertainment simulator like Microsoft FSX, you can see that why Laminar Research spent more time on the aircraft physics modeling instead of providing great details to look at. Additionally one can supplement the default scenery (of either simulator) by purchasing 3rd party packages or creating your own. A comparison of scenery in each simulator is in the following figures (figure 2-1 and 2-2).

Figure 2-1. Flying a Cessna C172SP over Innsbruck, Austria in Microsoft FSX

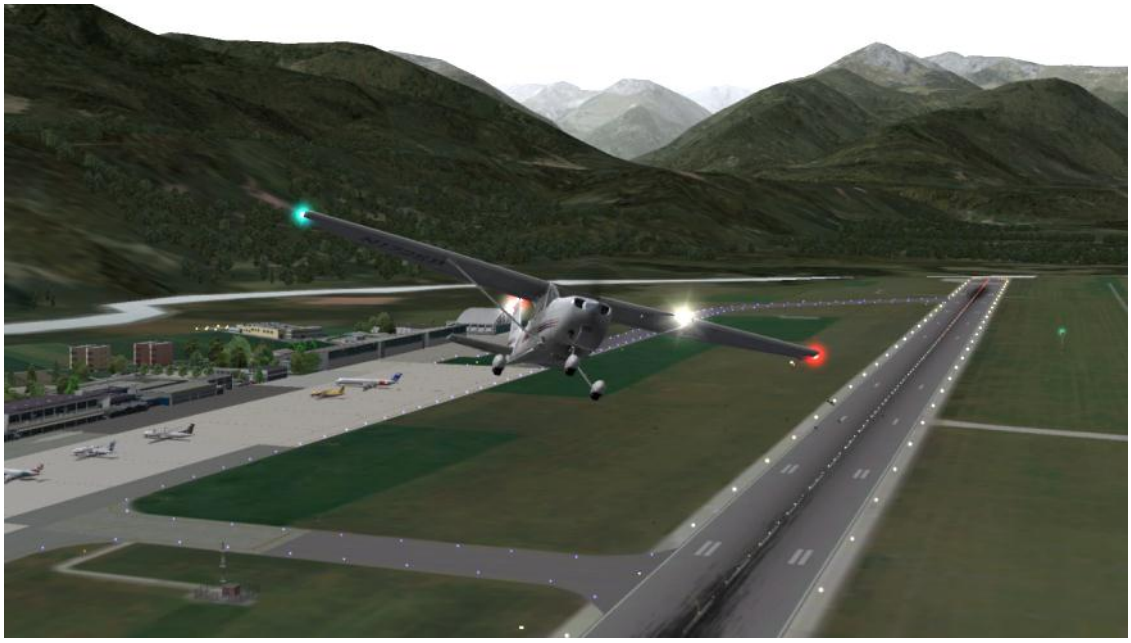


Figure 2-2. Flying a Cessna C172 over Innsbruck, Austria in X-Plane 9.4C

Comparing the two images preceding this paragraph (Figure 2-1 and 2-2) you can notice some interesting differences between the two simulators. However, before we start, we should make it clear that in X-Plane 9, the default airport is Innsbruck Kranebitten Airport (LOWI), and is therefore has higher detailed scenery than many of the airports in the game, whereas in Microsoft Flight simulator it is just another airport from a list of thousands. One thing that is noticeably different between the two simulators is that smoothness of the rendering of the aircraft. Both simulation packages ran on the same machine and resolution, but the one in FSX is slightly jagged. Also, while not able to tell form this picture, X-Plane supports curved runways, which this airport has (runways typically are not a 0% gradient), whereas in FSX, it's a flat straight line. Also speaking of airport surface areas, the taxiways in X-Plane are also of a higher detail where in FSX they just intersect the runway as a opposed to having some curve into it. Terrain data in either X-Plane or FSX appears identically the same (there were no missing or added terrain features), so there is no differentiation in that department. Render distance is essentially the same, but as the terrain fades off into the distance FSX does a better job of blending the horizon and the sky. If you notice in Figure 2-2 the mountain in the distance appears to be on a boundary of different shades of grey in X-Plane.

Out of this table, one requirement is much more important, especially if this simulator is to ever be used for ground based flight training: the ability to deliver a constant frame rate of 30 frames per second (FPS). In FSX you are able to set a target frame rate, but unfortunately this target is just a way for you to compare the output frame rate and the ideal, so that you can adjust the graphics settings yourself on the computer. Unfortunately, this also means at times the system can become slow and as a result the simulation will not feel as real at all. X-

Plane does address this by allowing the user to set a target frame rate, but unlike FSX, the software will actually scale the graphics settings of the game to match the target. In addition to this feature X-Plane also has the option to purchase a $500 USB key that allows for the simulator to be considered FAA accredited through the guarantee that the output frame rate will not drop below 30 FPS. For the purpose of this project, this key will not be purchased.

Another point that needs to be addressed is the inclusion of computer controlled or AI aircraft that exist in the simulated environment. Microsoft Flight Simulator has this feature built-in and turned on automatically. These AI-based aircraft fly normal routes and will land, takeoff and even make contact with the AI-based Air Traffic Control. Additionally they are not limited to one type, almost every single flyable aircraft in the game can be found in the skies on an AI flight path including some that are not available to the user. This ensures a decent mix of air traffic that adds to the realism. On top of all this, for users that seek true realism, many users in the FSX community have generated their own flight plan files. This allows the addition of the schedules of entire airlines or flights around an airport. Additionally this also means that it is relatively straight forward to create custom flight plans.

X-Plane however, lacks built-in AI-based aircraft support. This however does not mean that you cannot have computer controlled aircraft sharing the airspace with the user, it just means that like everything else with X-Plane a plug-in has to be developed. Luckily there exist several plug-ins already available to download for free. FSImp is one such plug-in that has spanned many versions of X-Plane and allows a user to import flight plans from Microsoft Flight Simulator in to X-Plane. The beauty about this solution is that an X-Plane user can utilize all of the flight plans developed by the Flight Simulator community, which far outnumbers the available flight plan databases that are available to the X-Plane Community. There are other plug-ins as well for X-Plane for AI aircraft, including one that is nothing more than a flight recorder that replays your past flights as computer controlled flights.

Table 2-2 Aircraft Modeling

No.	Item/Description	Req. No.	FSX	X-Plane 9
1.	Included 3D Model Generator	S4.A	No	Yes
2.	Ability to change aircraft parametric data on the fly	S4.B	Yes	Yes

In order to deliver an accurate simulation of the aircraft, a detailed and realistic model was developed to the best of our abilities. Each of the simulators utilize two different methods to model aircraft, with FSX requiring the use of a 3rd party 3D modeling software such as 3ds Studio Max. In addition, once the model is generated in the software, one must then create an aircraft.cfg file which specifies the model properties. As mentioned earlier this creates the possibility for generating a model that does not meet the flight characteristics. X-Plane

utilizes an included model generator that allows us to build an accurate model without utilizing expensive software. Pictured below is how one builds a fuselage with the editor; in addition you can edit all the other features of the aircraft including avionics and engines. A more specific discussion on the development of requirements for the aircraft model can be found in the next section, Section 2.2.1.2 Aircraft Model Requirements is where we cover modeling the aircraft using the tools in X-Plane in addition to parametric data we currently have from the manufacturer. Table 2-2 above summarizes this information.

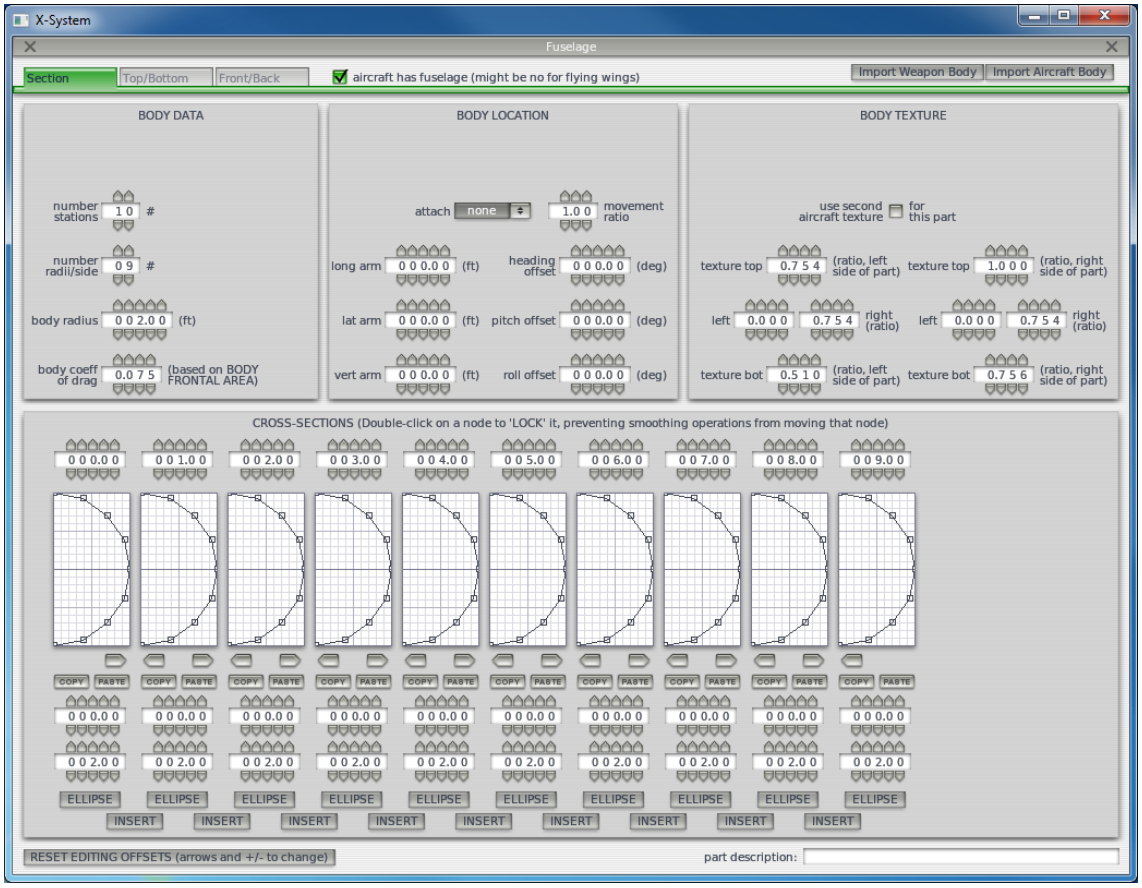


Figure 2-3. X-Plane Model Editor running in Windows

Table 2-3 Entertainment Features

No.	Item/Description	Req. No.	FSX	X-Plane 9
1.	Detailed Crash Effects	S3.B	No	Yes
2.	Multiplayer Support	S5.A	Yes	Yes
3.	Aircraft Sounds	S3.C	Yes	Yes
4.	Ability to create custom scenarios/missions	S3.D	Yes	Yes
5.	Built-in Instructor Operator Station (IOS)	S8	No	Yes

By default in both of the flight simulators, when the aircraft crashes or the airframe is overstressed due to physical factors, the flight ends with the aircraft stuck in that position; either struck the ground or featured overstressed conditions. However, X-Plane allows for the removal of flight surfaces if the aircraft goes past over-speed and over-G thresholds as well as the flaps and gear doors when over-Vfe (Velocity flap extended) thresholds have been passed. FSX has overstress indicators in addition to crash detection, but they are not nearly as extensive as in X-Plane.

As for multiplayer support FSX utilizes the GameSpy matchmaking service for multiplayer sessions across the internet, but also supports direct connections utilizing Microsoft DirectPlay for computers on the same local area network. X-Plane also allows for direct connections over a local network. For each of the simulators the multiplayer connectivity options allow us to also integrate an Instructor Operator Station (IOS) to remotely control aspects of the simulator. For FSX one would need to have to write additional software, and with X-Plane this feature is built in and would just require an additional installation of X-Plane. Alternatively, we are also able to utilize the variables presented in the X-Plane SDK and create our own IOS application. This would allow us to customize the interface to our needs or provide different interfaces for different usage scenarios. This way there could be one IOS interface for public demonstrations and one for actual flight training, should it be used for that. The “entertainment” features test results are given in Table 2-3 on the previous page.

Table 2-4 Simulator to External Flight Instruments/Controls Communication

No.	Item/Description	Req. No.	FSX	X-Plane 9
1.	Protocol/API to interface with flight simulator software	S2.A	Yes	Yes

FSX allows for two methods of interfacing with simulated flight controls and instruments: the SimConnect API and the legacy FSUIPC interface from previous versions of Flight Simulator, but still supported. X-Plane also has an API available in order to develop plug-ins for the software. This allows us to develop .dll and .exe files to facilitate the data flow between the software and our hardware. The API for either flight simulator allows access to nearly all of the internal variables used in the simulators. This allows us to dig into the simulation state and pull out information ranging to which lights are on, is a switch on or off, to changing the weather, changing aircraft position, and of course simply flying the aircraft. This allows us to write a plug-in for X-Plane or an application for FSX that allows us to do nearly everything. Due to this we will be able to interface with each of our gauges, our indicator lights, switches and our flight control systems.

Our overall requirements list is presented the Table 2-5 below. This incorporates all the requirements that were derived in the preceding paragraphs.

Table 2-5 Simulator Requirements

Req #	Task	Summary
S1	-	Realistic look and feel
S1	A	Realistic Scenery
S1	B	Inclusion of Airports Worldwide
S2	-	Ability to change environmental factors dynamically
S2	A	Ability to interface hardware with software via API
S3	-	Model Entertainment Aspects
S3	A	Weather Effects
S3	B	Crash Effects
S3	C	Sounds
S3	D	Ability to create custom scenarios/missions
S3	E	AI Aircraft also utilizing airspace and airports
S4	-	Aircraft Model
S4	A	Aircraft Exterior Model
S4	B	Model parametric data
S5	-	Ability to interface with other Flight Sim/X-plane games
S5	A	Native Multiplayer Support
S6	-	Guaranteed minimum 30 FPS
S6	A	FAA Certification - Optional Requirement
S7	-	Ability to interface controls/flight instruments
S8	-	Ability to interact with an Instructor Operator Station

2.2.1.2 Aircraft Model Requirements

The requirements that we needed in our simulation for the aircraft model have to do with the actual aircraft and how we can get its physical characteristics into the X-Plane editor. While we were able to get a good amount of information from the manufacturer (Aero) and the Importer (GoBosh) on the aircraft model specifications, some of the needed info was not able to be obtained. We also contacted the creator of X-Plane and some of the modeling that we wanted to do was not in the current release of X-Plane and was planned for a later time. Yet, from some of the info we could get from the sponsor and from the brochure that was given to us we got some of the info needed to create a model of the plane we are trying to simulate. We also used the planes manual that we were able to get from an online site (I don't think it should of been published but we found it).

In order to get the information into the actual X-Plane simulation we needed to use the included application Plane-Maker. This program is bundled with the game and has an interface that allows you to input the various parameters of an aircraft needed info to build a model and make it fly. As for the actual model of the plane you need to have an .acf file and this is what is created by the Plane-Maker software.

We tried to create the aircraft outside of the editor and ran into many difficulties. The drawings for the model needed to be exported in the form of an .obj file and then loaded into sketch-up and then could be exported as an X-Plane format but this proved to not work or be unworkable in the time that we had so we needed to go back to the Plane-Maker software for the solution that we eventually went with.

Given the aircraft specifications in Table 2-6 below, we used this information to develop our aircraft model, and also developed an airfoil for the wings using publically available wind tunnel testing data. A further discussion of the design and implementation of the model will follow in the design chapter.

Table 2-6 Summarized Aircraft Data[1,2]

Wingspan	27'4"
Height	7'4"
Fuselage Length	20'6"
Width (At Cabin)	41"
Prop. Diameter	5'8"
Lifting Area	122.7 ft^2
Wing Profile	NACA 4415 mod.
Empty Weight	820 lbs.
Maximum Weight	1320 lbs.
Maximum Cruise Speed	116 ktas.
Stall Speed / Minimum landing speed (V_{s0})	39 kts.
Stall Speed / Minimum steady flight speed (V_{s1})	44 kts.
Normal Operating Speed	110 kts.
Never Exceed Speed	129 kts.
Maneuvering Speed	90 kts.
Service Ceiling	13,200 msl
Sea Level Climb Rate	850 fpm.
Maximum Range	360 nm.
Minimum Take-off distance	380 ft.
Minimum Landing Distance	656 ft.
Wheel Track	7.42 ft

Figure 2-4 below shows one of the screens in the plane maker software. This shows some of the parametric data input for the plane maker and gives an idea of what we had to work with. While X-Plane makes the creation of a aircraft model relatively easy compared with Microsoft FSX, it is still a very challenging task. With not a single group member having experience with 3D modeling or aerospace engineering or the basics of aircraft operation, there is a limitation on our abilities to create 100% accurate model. The GUI of the plane maker was

[1] *(2009, Nov.). GoBosh G700S Specs [Online]. Available: http://www.gobosh.aero/G700.cfm*
[2] *(2009 Dec.). Airplane Flight Manual Aero AT-4 Light Sport Airplane [Online]. Available: http://www.ussportaircraft.com/uploads/Gobosh_POH_1_.pdf*

also fairly complex and lacked a variety of features that would have made the creation of a flight model much easier.

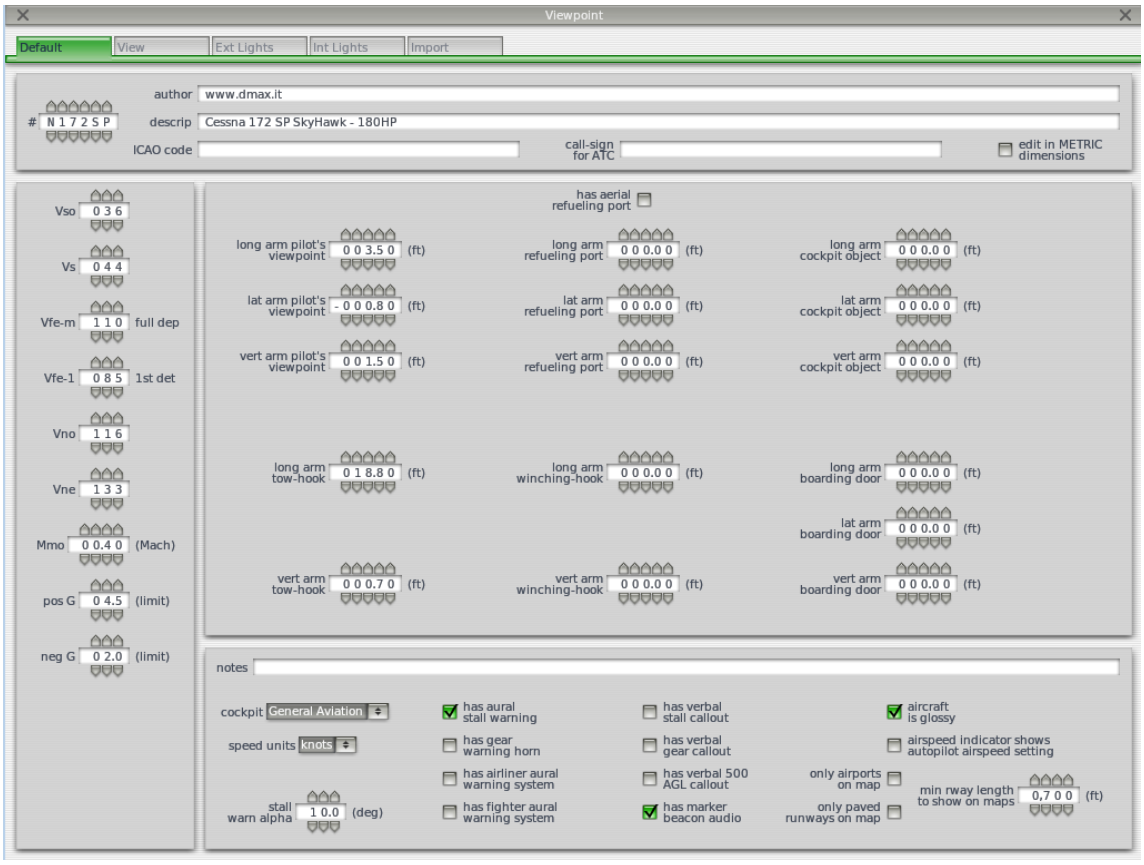


Figure 2-4. Plane-Maker viewpoint setup

Once the model had all of its parameters input then came the part of the modeling that needed some sort of artistic capability. The Plane-Maker tool also includes a basic model editor that allows you to change the fuselage to the correct shape it also gives you the options for each of the wings and nose of the aircraft. A screenshot below in Figure 2-6 and 2-6, while not representing the model that we developed shows process for creating the fuselage of the aircraft. In Figure 2-6 the wireframe representation of the model is manipulated by pulling on the points indicated. These can be stretched in any direction and all of the three views will be updated. Additionally more sections can be added from the default to increase the ability to create smooth edges. Another plus is that we can place an image behind the wireframe to trace our fuselage shape.

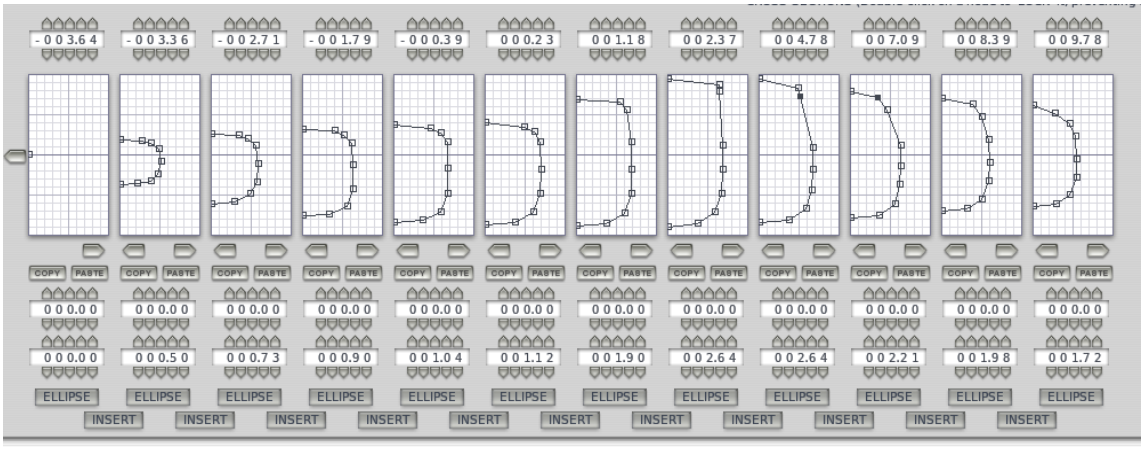


Figure 2-5 Plane Maker Fuselage Editor

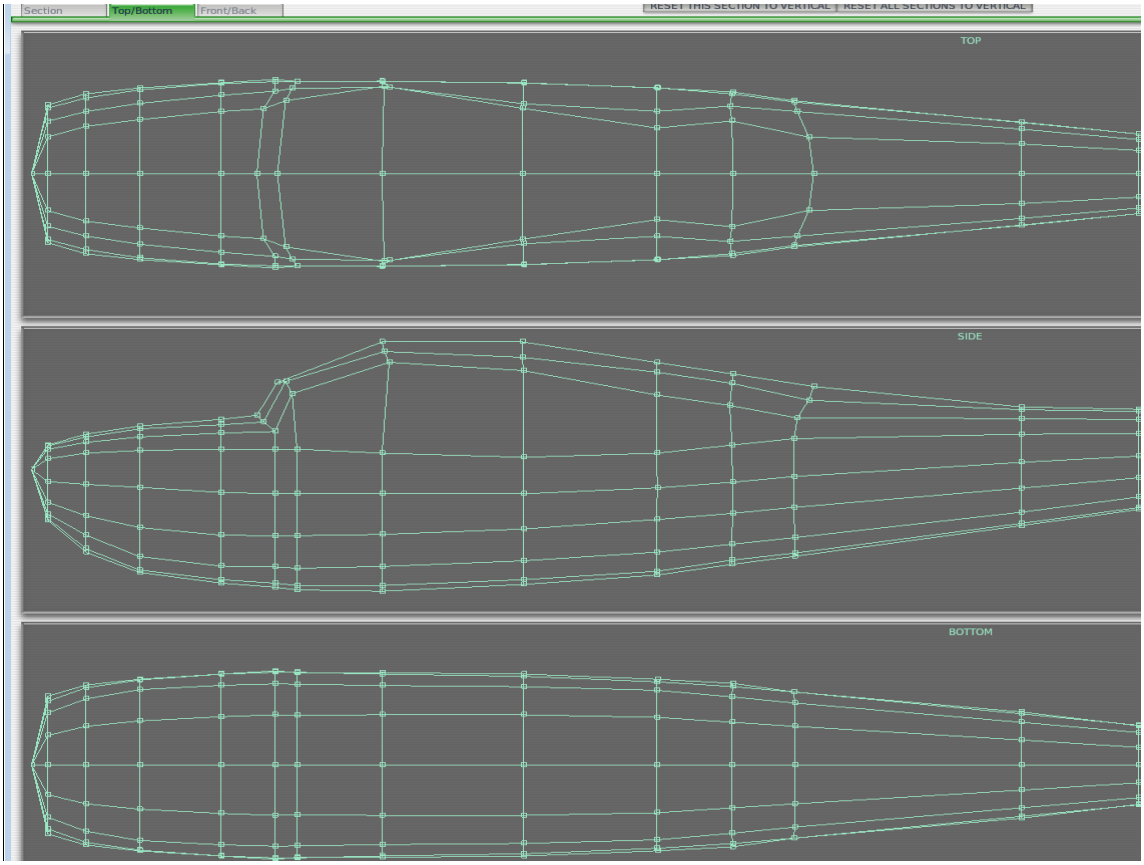


Figure 2-6 Wireframe representation of a Fuselage in development.

Another part of the plane-maker software that will allow us to interact with the simulation correctly is the fact that you can actually model the aircrafts systems and inputs. This includes being able to customize an existing cockpit panel or creating your own panel and even the flight electrical systems. All of the gauges can be displayed and indicator lights can be switched on and off based on the data that is given to the simulation. In Figure 2-7 we show a default cockpit from a Cessna C172 that has been modified to include some additional gauges

utilizing the Plane-Maker tool. They are located on the right side of the cockpit. This is helpful with the testing of our gauges, since the output to the in game gauges is the same as we will be outputting to our hardware via the plug-in we developed. Figure 2-7 shows the layout of an example cockpit built with the cockpit editor that is part of the Model Editor software.

Figure 2-7. Example cockpit showing additional gauges.

2.2.1.3 Microcontroller Software Requirements

The requirements for the microcontroller software were really based on the hardware that ended up using to implement our design. We are used stepper motors and A/D converters to design the gauges and the controls. The stepper motors needed to be updated at a rate that will make the movements look smooth. This led to a decision on the motor being a stepper motor, that and the fact that we needed to continually go in 360 degree circles. We used all the same stepper motors with 200 full steps per revolution. With the 200 steps we can get a 1.8 degree resolution this may not be good enough for smooth rotation. We needed to half step to overcome this and that required that the microcontroller/FTDI chip be able to get information from the host computer and update the motor twice as fast in order to get smooth looking steps.

The servo motors we looked at needed to be updated by a signal for a certain period of time in order to move it. This would of been fine except that when that design idea came before our sponsor, he didn't like it and that led to us picking the stepper motors as they could be controlled easily through the FTDI chips. There is more discussion of the actual controlling of the stepper motors in the microcontroller requirements section of this paper.

The software that we are running X-Plane tells us our capabilities. The plugin allows us to dictate the update rate and should update their graphics at a rate of 30 frames per second minimum. This gives us our update rate for sending signals to the gauges. We could of updated faster than this but it would just be the same value sent and that would just cause unneeded traffic. That could of slowed down the code and could take away from other threads that were operating. So at that rate we needed to have a speed of : 1 sec / 30 = 33 ms and we also needed to send enough information to update each of the gauges we implemented. For the basic six pack we have 8 bits x 6 gauges which is the same as 6 bytes that need to be updated every 33ms that gives us a speed of 6 bytes x 30 = 180 bytes/sec this was easy to keep up with over USB speeds as USB 1 was 12 Mbit/s. There will be room to expand the gauges and input devices as needed.

The microcontroller/FTDI chip we used set the type of software requirements that we needed also. It determined the how we needed to update the gauges as well as how we could get our information from our controls.

2.2.2 Hardware Requirements

The hardware requirements development includes all of our hardware and physical assemblies that will need to be created. This includes our simulator PC that will run X-Plane, the requirements for the microcontroller (outside of software issues), requirements for our aircraft instruments, and requirements for our flight controls. While we did not receive our cockpit or demonstrate at Sun ‘n Fun as intended, some of these requirements were not implemented (especially for the computer setup). As these were requirements developed for our original design, we will retain these discussions at the end of this paper show the status of each of the requirements and which ones we met or did not meet and why.

2.2.2.1 The Game PC

In order to meet our performance requirements for the flight simulator of providing a constant frame rate while maintaining detailed graphics we have established a baseline for the simulator computer that goes above and beyond the system requirements listed by Laminar Research for X-Plane 9. Not only will this give us room to play with the graphics settings, but will allow for the computer to be used for years to come as newer versions of software is released. In Table 2-6, the minimum requirements for X-Plane are listed, while our suggested requirements are listed in Table 2-7 in order to deliver excellent graphics and performance.

Table 2-6 X-Plane 9.4 Minimum Requirements

Operating System	Windows XP/Vista/7, Linux, MacOS
RAM	1 GB
CPU Speed	2.0 GHz
HDD Space	60 GB
Video Card	64 MB

Table 2-7 Established Requirements

Req #	Task	Summary
C1	-	USB ports for Flight Controls and Instruments
C2	-	120 Degree Field of View
C2	A	Three LCD Monitors
C2	B	Graphics Card/External Device to output required resolution
C3	-	2GHz 64-bit CPU (minimum)
C4	-	4GB of RAM
C5	-	120GB Hard Drive (minimum)

2.2.2.2 Microcontroller Requirements

In order to make the gauges we needed to decide on what the important characteristics of the gauges will be, below is a list of requirements that needed to be met for each of the gauges. These are laid out in Table 2-8 below.

Table 2-8 Established Requirements

Req #	Task	Summary
M1	-	USB Controlled
M2	-	Use less than 5V
M3	-	Minimum 8 I/O Pins for external communications
M4	-	Fit inside of a 3.24”x3.24” footprint
M5	-	Low Cost Microcontroller
M6	-	As self-contained as possible

The gauges that we duplicated are the six-pack that is located slightly to the left in the photo in Figure 2-8 (next page). From the requirements we needed to make these gauges look and act just like the real gauges would in the actual Bosh aircraft. We also need to make them react as the gauges in the X-Plane game, this makes them as real as the model in the simulation. They also needed to fit in to our budget was a very small amount. There of course are the pre-made gauges that were discussed, but those are expensive, so we needed to think of other ways. What we came up with is the handmade gauges discussed in this paper, and we also got lucky when we talked to the people at GoBosh, and they sent us some used gauges to take apart and alter for use in our simulator. The gauges are controlled by a stepper motor. In order to do this we needed to come up with a way to power these gauges as well as control them.

Figure 2-8. A photo of the cockpit in a GoBosh aircraft. *Photo by Robert Gysi*

We were given a requirement that everything needed to be connected via USB. With this constraint we had to find a way to control the motors with the USB protocol. Along with the restriction of speaking USB protocol it needed to fit into the power specs of USB so we chose to connect an outside power source. This allowed us to control a stronger stepping motor and also not worry about meeting the requirements for the USB protocol, which only allowed for 5 volts at 100mA at startup and 500mA during peak running of all coils of the motor[3].

Looking into the different types of stepper motors we to go with a 12 volt and low amperage motor that used to drive 5.25" floppy drives. The stepper motor is mostly used in robotics to add an amount of torque (strength) to the limbs. The stepper motor requires sometimes turning on more than one coil inside the motor in order to get the correct amount of movement. This required more current from the power supply. Since we are using a computer power supply the amount of current and voltage required is less than the voltage and current the power supply supplies. Micro-stepping is typically used in applications that require accurate positioning and a fine resolution over a wide range of speeds[4]. Although the microstepping of the motors was not implemented and is left for future expansion of the project if it is to be included within the scope of a future upgrade work on our system. Stepper motors have the capability to run at lower currents since the current is what controls the motors torque or holding power, and we only need to hold a small pointing device.

---

[3] (2009, Nov.). USB as a power source [Online]. Available: http://www.girr.org/mac_stuff/usb_stuff.html

[4] (2009, Nov.). Stepper Motors reference guide [Online]. Available: http://ams2000.com/stepping101.html

Use of the servo needed some sort of extra timing circuit in order to give the servo motor the pulse widths it desires to run correctly. The initial design of the controller for the servo involved a 555 timer, and some sort of Digital to Analog converter. The 555 timer was to be used to give us our pulses of the different lengths, depending on the analog values that are received from the D/A converter. In order to get the most from our FTDI chip we needed to get an eight input D/A that can give out voltages with high resolution (2^8 = 256 values between 0 and 180 degrees). This was ruled out when our sponsor, who has an electrical engineering background, decided that the 555 timer circuit couldn't give us the most stable of time pulses to keep everything accurate.

The design and test phase of the project will determined we needed to add some sort of outside power source to help control the gauges, but we first tried to put it together using no outside power source. Unfortunately, this led to the devices disappearing during operation due to the power supplied was too low for the device to function properly. A diagram of the circuits can be found below in Chapter 3.

2.2.2.3 Flight Instrumentation Requirements

The flight instruments are one of the most important elements regarding the authentication of the simulation. For this reason we had some very strict requirements regarding the instruments. First of all, it was asked that we use mechanical, heads down gauges for all the instruments we were modeling. In the simulation world, many times the instrument panel is modeled using LCD screens displaying virtual gauges and this functionality is even built into the simulation software. The problem with virtual gauges is that you don’t get the look and feel of the cockpit like you do with mechanical gauges. Figure 2-9 shows the view of the instrument panel from inside the cockpit. Figure 2-10 shows our simulator’s instrument panel. As you can see the look of the simulated gauges are almost identical to those of the actual gauges. Unfortunately we did not have the time or resources to model all of the Gobosh instruments instruments so we just modeled the essential flight gauges as seen in Figure 2-10.

Figure 2-9: The interior view of the instrument panel. *Photo by Robert Gysi.*

Figure 2-10: Our simulator's instrument panel. *Photo by Lewis Vail*

The instruments that were essential to properly simulate the aircraft were the standard six-pack of gauges. Figure 2-11 is an enlarged view of these gauges from the actual aircraft. They include (from left to right, top to bottom) the

airspeed indicator, the attitude indicator, the altimeter, the turn coordinator, the heading indicator, and the vertical speed indicator. These are the gauges that are essential to successfully fly and navigate a plane. Although we were not able to model any other instruments these were enough to get the feel of flying a GoBosh. The following are some specific requirements for each gauge:

Figure 2-11: This is the standard six-pack of gauges in the GoBosh.
Photo by Robert Gysi.

The airspeed indicator (top left corner of figure 2-11) requires the ability to record up to 160 KTS as indicated on the faceplate. This requires almost a 360-degree range of motion. Among the six-pack gauges, this one requires one of the faster moving needles but still needs to support small fluctuations in airspeed without looking choppy.

The attitude indicator, also known as the artificial horizon (top middle of figure 2-11) is one of the more complicated gauges. It is required to turn all the way around (more than 360 degrees) and part of the face must slide up and down to indicate whether the plane is nose up or nose down respectively. The speed and precision needed for this gauge is comparable to that of the airspeed indicator.

The altimeter (top right corner of figure 2-10) requires the ability to record up to 10,000 feet above sea level as indicated on the faceplate. The altimeter has two arms like a clock; the long arm (corresponding to the minute hand of a clock) represents hundreds of feet above sea level. This arm will need to go all the way around up to ten times. The shorter arm (corresponding to the hour hand of a clock) represents thousands of feet above sea level. This gauge will move at a fairly fast rate, especially during dive maneuvers, and therefore the gauge we build must turn the needles fast enough to replicate this real worlds speed. The requirement for smooth movements also persists with this gauge but precision is not as critical as with the slower moving gauges.

The turn coordinator (bottom left corner of figure 2-11) is another more complicated gauge, similar to the attitude indicator. It consists of two components, a plane shaped needle that indicates the bank of the plane during a turn and a small ball in a tube (similar to a bubble level) that indicates the slip and skid. Both components require the least range of motion and therefore every move they make must be as smooth as possible. This gauge operates at a moderate speed that is far less critical than some of the other gauges. To optimize authenticity, this gauge must have four tick marks as shown in figure 2-11. The top two tick marks represent no bank and the bottom two marks represent a turn in which the heading change is three degrees per second. These two bottom marks are now at the 2 minute marks because it takes two minutes to do a full 360 at this bank[5].

The heading indicator (bottom middle of figure 2-11) is required to turn all the way around (more than 360 degrees) just as with the attitude indicator and altimeter. This gauge is probably the slowest turning of all the gauges. This means that any choppy movement would be magnified.

The vertical speed indicator (bottom left corner of figure 2-11) is required to have a range of motion of 360 degrees. Unlike the altimeter, the attitude indicator, and the heading indicator, this gauge is not required to turn more than 360 degrees. To match its real life counterpart, our gauge must also have a range of ±2000 feet per minute. This is the fastest of all the six-pack gauges and our model will have to replicate this speed. But because this gauge operates at a higher speed, choppiness and lack of precision is less of a concern, as it is hardly noticable.

2.3 Project Budget

Our project sponsor has established a budget of $1500 during our initial discussions. Due to the previously mentioned issue with not receiving a cockpit we went through a major design revision. This affected our budget in that we wound up being significantly under our original budget due to not purchasing several components. For this section we will discuss our actual planned budget in addition to our actual spending after design changes.

For the original design effort we anticipated that our budget would need to be expanded to cover the costs of the simulation computer and monitors. Up to the moment the decision was made to alter our project due to the lack of a cockpit, we had an anticipated cost of approximately $1700. In the event that our sponsor would have not agreed to pay for costs over $1500, the members of the group were prepared to meet the additional costs required to implement this design. Our original budget can be found in the appendices of this document for reference.

[5] (2009, Dec.). Wikipedia Article: Turn Coordinator [Online]. Available: http://en.wikipedia.org/wiki/Turn_coordinator

Once the decision was made to forego the purchase of a computer, we knew that we would not have an issue meeting our requirement of keeping spending under $1500. However, since to go to that amount under the circumstances would not be in good taste, we attempted to keep our spending to approximately within the bounds of what we felt our original non-computer component cost would be. As a result we made an attempt to spend approximately $500 on components. We found this difficult to keep given several last minute purchases including a powered USB Hub. In the end our spending represented a total of $626.36 or roughly $100 more than our anticipated spending. Table 2-9 below represents our actual spending upon completion of spring 2010 semester. All costs are included, although may be listed under general categories for small items such as screws would be under Misc. Hardware. Quantities on items may be smaller than required for project completion due to getting some components from other sources for free or by using parts already possessed by a group member.

Table 2-9 Expenses

Item	Part Number	Quantity Required	Unit Cost	Total Cost
USB Communication Board	FTDI245BL	10	$30.00	$300.00
IC Sockets (Assorted)	Various	-	$21.72	$21.72
PCB Boards (Small)		4	$1.99	$7.96
Wire		3	$5.99	$17.97
Transistor	2N3904	4	$0.79	$3.16
Diodes	1N4003	-	$6.75	$6.75
Spacers (Assorted Lengths)	N/A	-	$28.20	$28.20
Stepper Motors		8	$5.00	$40.00
Terminal Blocks	N/A	36	$0.30	$10.80
PCB Boards (Large)	N/A	7	$2.50	$17.50
Buffer Chip	CD4050	6	$0.35	$2.10
Comparator IC	LM741CN	5	$0.25	$1.25
A/D Converter	ADC0804LCN	3	$2.50	$7.50
Powered USB Hub		1	$49.99	$49.99
USB Cables	Various	9	$5.00	$45.00
Slide Potentiometers	RA6020F-10-20D1-B10K	3	$2.12	$6.36
Molex Connectors	Various	-	$20.36	$20.36
Epoxy Putty	N/A	1	$3.97	$3.97
3/8 x 0.035 Aluminum Tube	N/A	1	$4.78	$4.78
Misc. Hardware	N/A	-	$30.42	$30.42
470-Ohm Resistors		3	$0.99	$2.97
Thin Aluminum	N/A	1	$2.38	$2.38
			Total	$631.14
			Project Budget	$1,500.00
			Difference	$868.85

2.4 Project Timeline

The original deadline of the project was determined by the date of Sun 'n Fun which meant that we needed to be done around April 1, 2010 in order to complete testing and transport the simulator to Lakeland. However, with that event not occurring, the deadline became the day of our presentation on April 21, 2010. Originally, we had wanted to adhere to our original schedule, but unfortunately due to uncertainties early in the semester we found ourselves getting behind our schedule and as a result did not meet an April 1 deadline. We did however, get all of our components working by the new deadline of April 21. Figure 2-14 shows our original project schedule in a simplified manner while Figure 2-15 shows approximate dates for when certain phases and aspects of the project were completed. Also, in Appendix B at the end of this documentation, one can find our original schedules from our fall semester documentation.

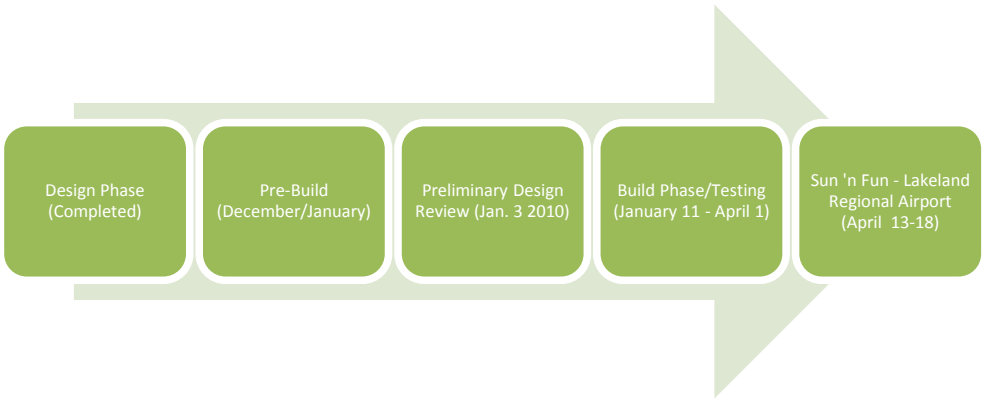


Figure 2-14. Original Project Schedule

What cannot be seen from the milestones listed in Figure 2-15 is that while the build phase officially started on January 11 (start of the Spring 2010 semester), is that no actual building started at this time. It was at the beginning at this time we built our first test gauge to validate our design and from this we determined that we needed to have our metal gauge decks fabricate by a machine shop. Gauge construction was placed on hold for a few weeks during this time in order for fabrication to be complete. Additionally, the issues with the cockpit arrival kept us from being able to start on working on controls, until we received word that we would not receive it in time. However, with these few issues, we still completed with enough time to test all of our components.

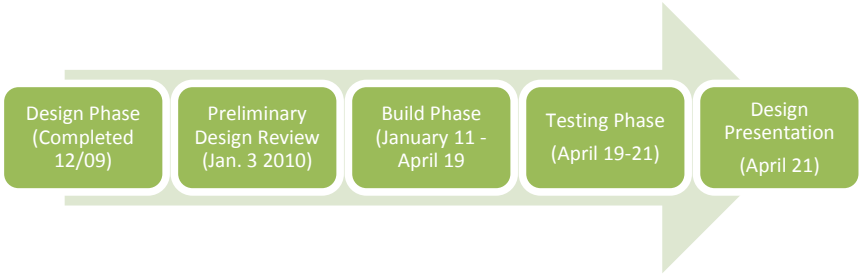


Figure 2-15. Actual Project Completion Milestones

Chapter 3
3.1 Design Summary

In Designing the System we needed to know which of the Flight simulators we were going to use. As well as what type of gauges we were going to implement and also what other switches and knobs needed to be able to interact with our simulation. For the initial design of this system we came up with a design plan that included a separate IOS station to control the setup of the simulation. Since that became an extra hurdle it was an "if we have time OPTION" and eventually never came to fruition. The system still has the capability to have the IOS from the simulation software we chose(X-Plane).

The control of the cockpit and the gauges is done through the USB chips that we have decided upon. The flight controls are divided into input devices and passive devices. The input devices are the yoke and pedals along with. The passive devices are the lights that are lit and the gauges. All of this is covered in the cockpit design section. The block diagram of the design is below in Figure 3-1.

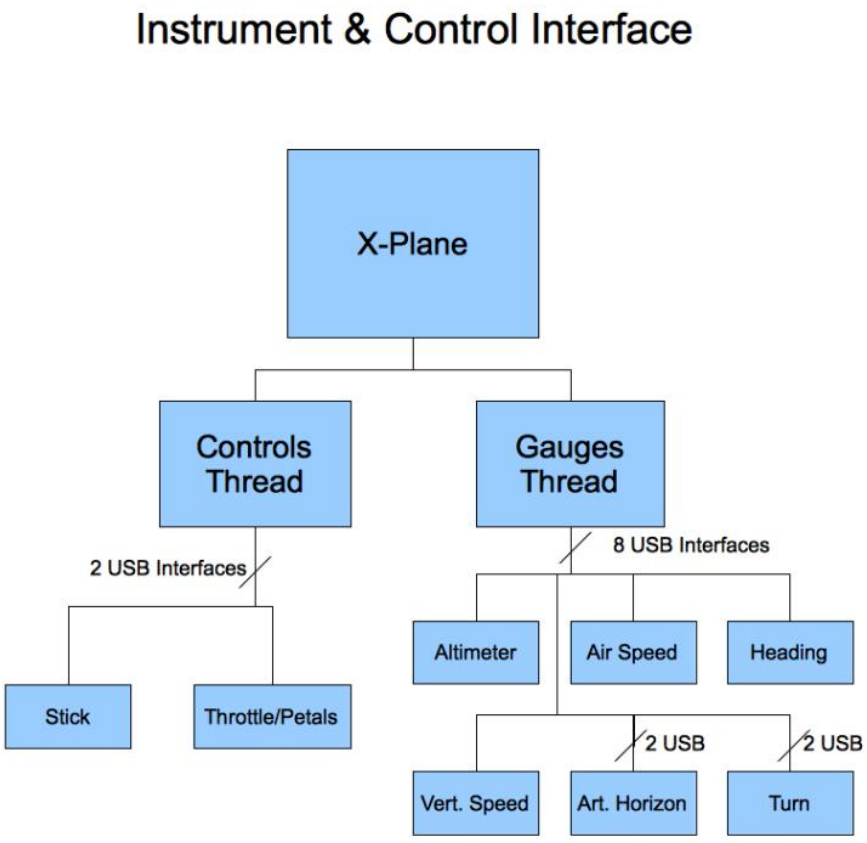


Figure 3-1 Block diagram representing cockpit interfaces and responsible parties.
Diagram by Lewis Vail

One of the major decisions was made regarding the design of the GoBosh 700S flight simulator was which flight simulator software to use. We had two options available to us as to which commercially available flight simulator software we could use. The first option was Microsoft FSX. Microsoft's flight simulator is the oldest and most established flight simulator of the two. We liked the fact that the community and resources available for the Microsoft flight simulator series was very vast and highly accessible. Unfortunately for the flight simulator community Microsoft decided in January 2009, to close both Ensemble Studios and ACES Game Studio due to a process of ongoing job cuts due to financial crisis and restructuring of their game studios. This became a factor in deciding which flight simulator software to use for this project.

The second option was X-Plane. X-Plane is the newest and least established of the two flight simulators. The X-Plane community and resources are not as vast and content rich when compared to Microsoft's. This was one of our biggest concerns when we were considering using X-Plane for our GoBosh 700S flight simulator. It turns out that X-Plane was not that different from Microsoft FSX in terms of our integration needs and requirements for the flight simulator. X-Plane also has a plugin architecture that allows users to create their own modules, extending the functionality of the software.

One unique feature that really stood out with X-Plane was the Plane-Maker Software. Plane maker is included with the purchase of the X-Plane software and allows users to build their own aircraft models. What is really remarkable about this is that there is no extra cost, unlike Microsoft FSX which requires the use of expensive 3rd party applications. Additionally, the method at which these models are simulated in the environment turned out to be a differentiating factor. X-Plane distinguishes itself by implementing a concept known as blade element theory. With plane maker you are able to build and model any aircraft using blade element theory. This feature will greatly simplify the design of the aircraft modeling and the aircraft flight dynamics. In the end we decided to use X-Plane as the visuals for the GoBosh 700S flight simulator.

One of the biggest components of the flight simulator was the design and construction of the simulators aircraft flight instruments. The flight simulator consists of the traditional six-pack of flight instruments. The traditional six-pack consists of the altimeter, attitude indicator, airspeed indicator, heading indicator, turn indicator, and the vertical speed indicator. All the aircraft flight instruments for this simulator are analog designed and assembled using stepper motors. As we researched the functionality and mechanical operation of each aircraft flight instrument we began to narrow down the way we were going to go about designing them.

Figure 3-2. Gauges to be implemented. The “Six-pack” gauges are the cluster of six large gauges to the left of the picture. *Photo by Robert Gysi.*

When designing the aircraft flight instruments we realized we needed 360 degrees of motion for most of the needles on the instruments. The best way to achieve this degree of motion was to use stepper motors. A problem occurred when using stepper motors as they have no unique home position. We solved this problem by using an optical sensor to establish the zero position. When the optical sensor is interrupted this signals to the computer that the needle is in the home position.

We also considered using servo motors in our design of the simulators aircraft flight instruments. Most servos unfortunately only have a range of motion of 180 degrees. This limits our ability to turn the needle on certain aircraft flight instruments the full 360 degrees required. There exist several ways to get around this limitation. A few that we explored consisted of modifying a servo by removing the mechanical stopper as well as a few other modifications, or by buying a servo motor capable of rotating 360 degrees. The following figures highlight the basic operation of the two types of motors applied to the gauges. The first shows operation of a stepper motor (figure 3-3), while the second a servo motor (figure 3-4).

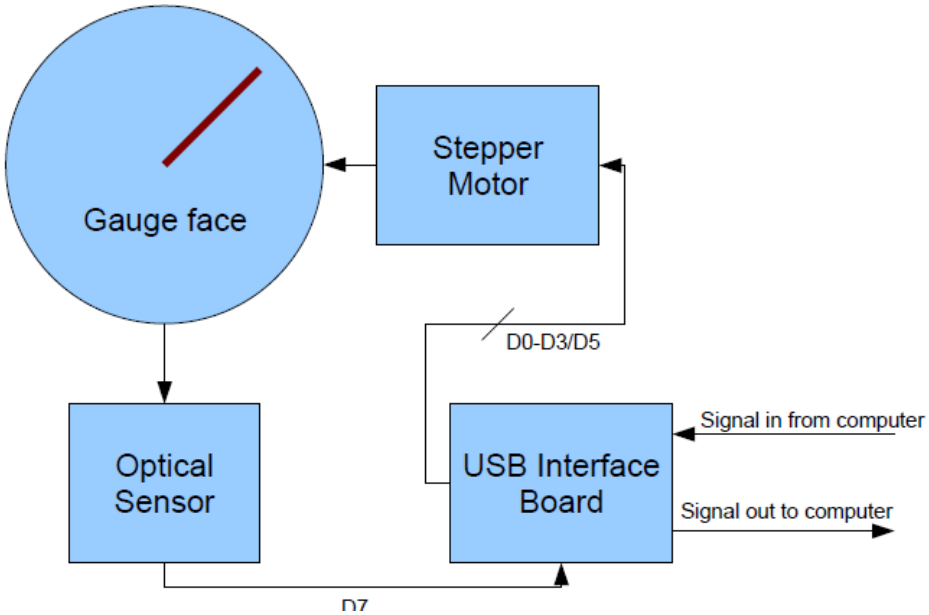


Figure 3-3 Stepper Motor Control Diagram. *Diagram by Lewis Vail*

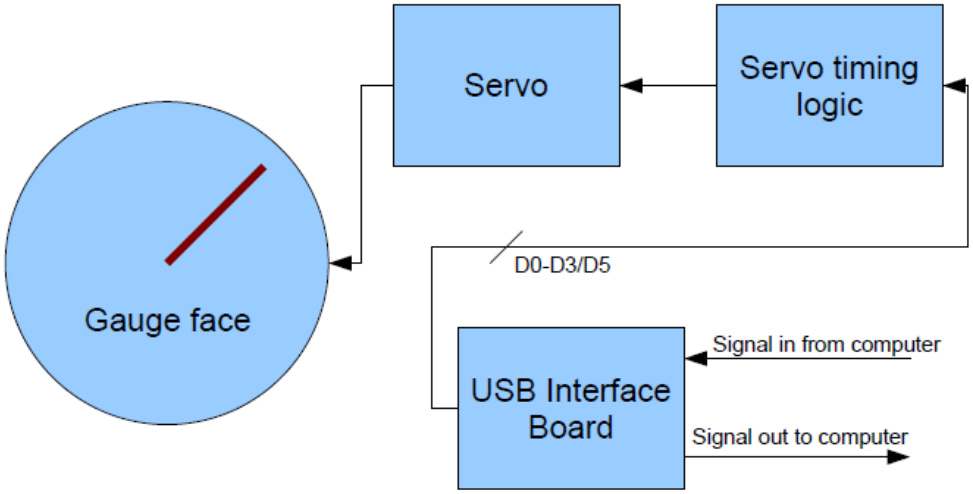


Figure 3-4. Servo Motor Control Diagram. *Diagram by Lewis Vail.*

The aircraft flight instruments are interfaced with the computer and the flight simulator X-Plane. Each aircraft flight instrument is controlled with the FTDI chip. The FTDI chip was chosen because the cost of each chip is only $5.00. The low price of the chip reduced our overall cost to design and build the aircraft flight instruments for our GoBosh 700S flight simulator. Figure 3-3 and Figure 3-4 show the control diagrams for each type of motor that was considered.

Each aircraft flight instrument is connected to the computer through a USB connection to a USB hub. The USB hub is in turn connected to the USB port on the computer. This is where our communication with the computer takes place. However, the previously mentioned FTDI chipset does not handle any processing of our data; it merely passes it over the USB to the desktop computer. This is excellent because we are able to do all of the programming in C++ and on the computer side, meaning that we will be able to write plug-ins for our instrument panels. The two flow charts below represent the relation of the Plug-ins to the X-Plane software and hardware. Figure 3-5 shows the relation for flight instruments while Figure 3-6 shows the relation for flight controls.

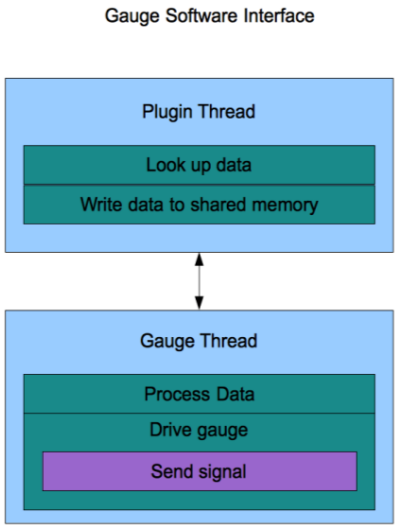


Figure 3-5. Gauge Software Interface. *Diagram by Lewis Vail.*

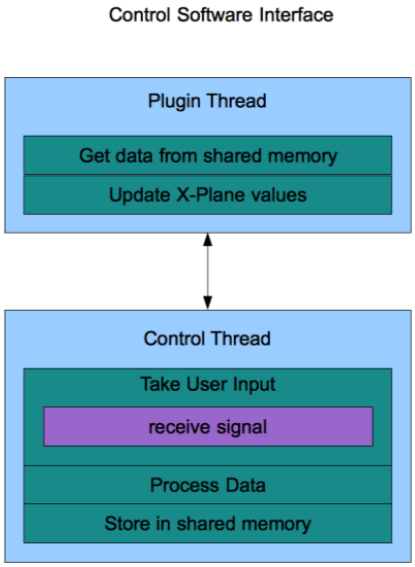


Figure 3-6. Control Software Interface. *Diagram by Lewis Vail.*

The following block diagram in figure 3-7 highlights the higher order levels of our design and process. The chart also assigns block responsibility and completion status. Being at the end of this project, all blocks are complete with the exception of the IOS. This is due to the decision to not take the simulator to Sun 'n Fun. However, X-Plane does have the IOS function built in.

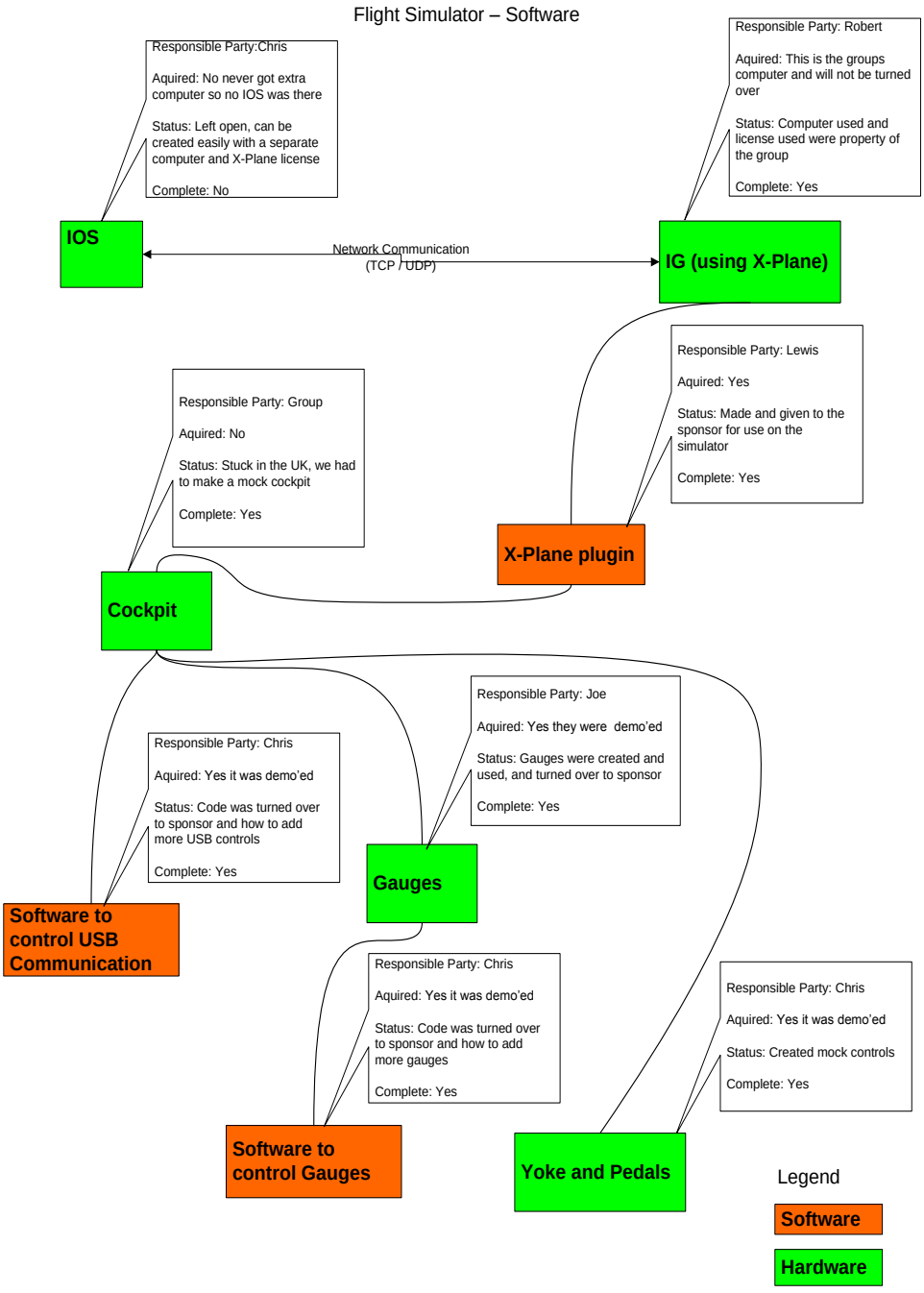


Figure 3-7 Software Flow Chart. *Created by Chris Dlugolinksi*

3.2 Microcontroller Design

After some research on the types of microcontrollers that were available with the ability to do the required tasks, we decided to go with the FTDI chip that has the capability to be used in bit bang mode which allows for control of individual I/O ports.

The 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. Yet, the added cost saved us some time. This gave us the ability to not really worry about USB protocol and it is able to remove our need for a microcontroller at least for the gauges we have implemented. The implementation only called for controlling of a few I/O ports, this was done with any I/O ports and the FTDI chip in its special mode was 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.

We used 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 contains many different types of gauge control capability and adding a new gauge only take setting the PID or VID to that type of gauge, and you 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. This cost for a premade board for the FTDI chip makes it cost around $25 per board which is slightly higher than predicted.

One of the most important parts of the design was 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 went with an external power supply (computer power supply) using Molex connectors for each controlled gauge.

3.2.1 Implementation of Hardware

The implementation or actual making of the hardware came from the FTDI FT245BL chip that was able to use the USB signal and allow the use of the 8 I/O ports right through the computer. We ordered the DLP-USB245M chip as a dev board for trying out our design. The chip came with a standard crystal, and an

EEProm that holds some descriptions that we used for naming the gauges. Along with the FTDI chip we needed a chip to keep the FTDI chip from being overloaded and we will use the 4050 buffer chip to help drive the transistors that drive the small stepper motor.

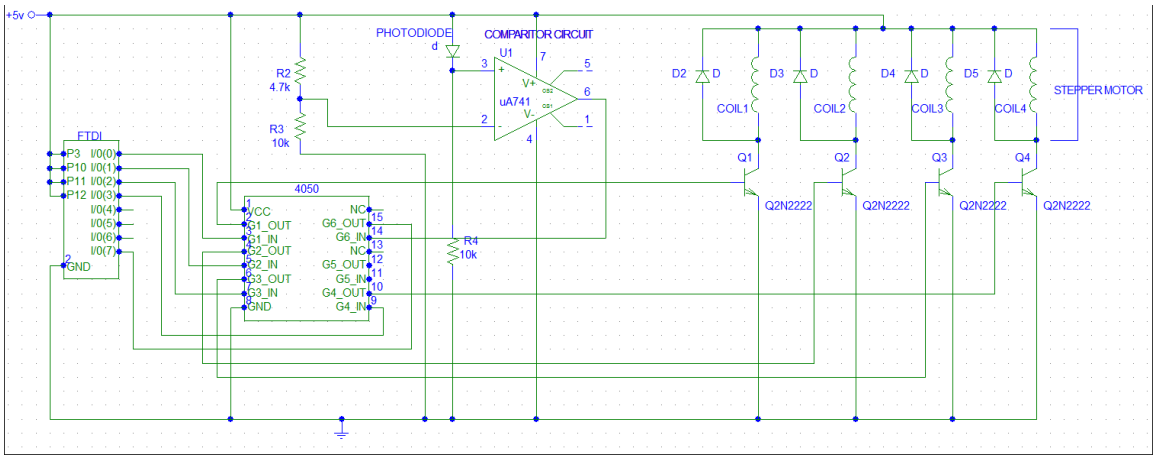


Figure 3-8. Circuit for the stepper motor controls *Created by Chris Dlugolinksi*

Figure 3-8 shows the connections needed to run each of the coils of the stepper motor and how the circuit is designed. The microcontroller being the FTDI chip and each of the coils needed to be turned on by the transistors. The diodes are in the design to stop the motor kickback current from coming back and hurting any of the hardware components.

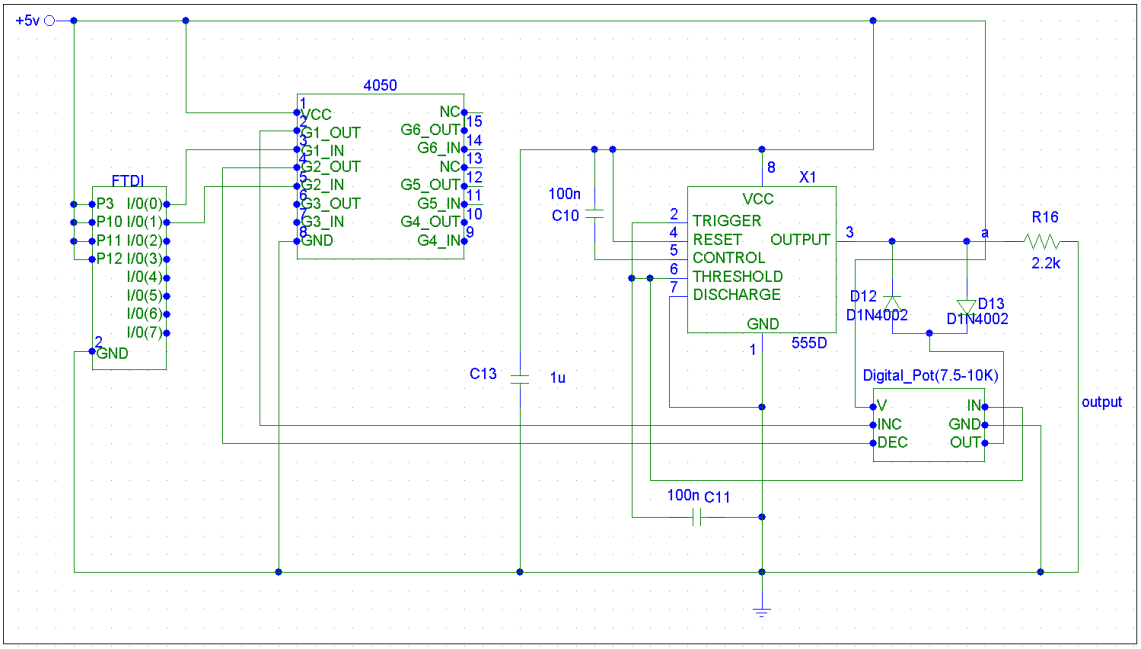


Figure 3-9. Circuit for the Servo motor controls *Created by Chris Dlugolinksi*

Figure 3-9 above shows the controls for the servo motor, the 555 timer is used to generate the pulse widths needed to control the position of the motor.

Depending on the needed position of the motor we are turning on a certain amount of resistance in the digital pot chip that will change the pulse width of the 555 timer. We are keeping the frequency at a frequency that will allow the pulses to be output every 20ms.

The circuit shown above was vetoed by the sponsor based on his knowledge that the signal from the 555 timer will not be able to give the correct pulse length repeatedly ("it is not that accurate").

The power supply needed to be able to provide 5 volts for any of the circuitry that we need as well as 12 volts for the motors that were used in the design. A simple power supply was used that needed to be able to give at least 6 amps. That is 1 amp for each of the gauges. We used an extra power supply from a computer, and Molex connectors. This gave us our 5v and 12 volts needed to drive the circuit.

3.2.2 Embedded Software

The embedded software has been minimized since we are using the special (Bit Bang) mode on the FTDI chip that allows for direct use of the I/O ports. The amount of embedded software just comes down to programming each of the gauges so that they are recognized as different gauges and can be recognized if unplugged and re-plugged in. This shifted most of the software to the host computer and also shifted where we manage all of our gauges as well. With this type of design we needed to come up with a way to get our modular design. We have a write-up that we will give the end-user on how to go about implementing our design for future expansion of the gauges or controls.

3.3 Flight Instrument Design

There are six main flight instruments, as pictured above, to be designed and simulated for our senior design project. Figure 3.10 shows the actual flight instruments that have been simulated for the GoBosh 700S. Figure 3.11 shows a close up of the traditional six pack of flight instruments arranged in a “basic T” that are very similar to the flight instruments contained in the GoBosh 700S and many traditional aircraft to date. The names of the flight instruments in Figure 3-11 starting from the top are airspeed indicator, attitude indicator, altimeter, turn coordinator, heading indicator, and vertical speed indicator. In this section we will describe the name and function of each flight instrument, how it is constructed, and the way we built and constructed our own for simulation.

Figure 3-10. Cockpit with gauges to be implemented. *Photo by Robert Gysi*

Figure 3-11. Closeup of the standard “Six-Pack” – Wikipedia *used with permission under the GNU License*[6]

Figure 3-12 shows the basic layout of how each flight instrument is connected to the game computer. Each flight instrument is programmed and controlled using a FTDI FT246BM USB chip, which is connected to an externally powered USB hub. The USB hub is in turn connected to one of the free USB ports on the computer.

[6] (*2009, Dec.). Six Flight Instruments.jpg Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/File:Six_flight_instruments.JPG*

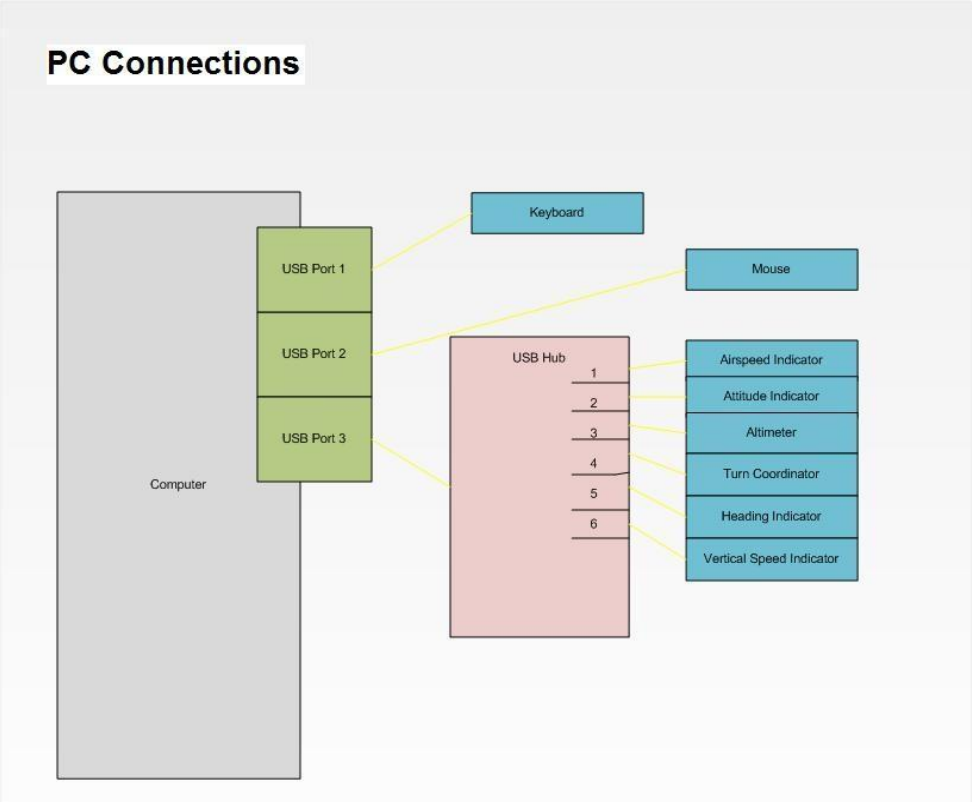


Figure 3-12: Diagram of USB controlled flight instruments showing the connections to the computer. *Created by Joseph Munera*

Additionally a mouse and keyboard are connected to USB ports on the game computer. This allows the operator of the simulator to adjust the settings of the X-Plane flight simulator as well as troubleshoot programming issues affecting the operation of the flight simulator's flight instruments.

Figure 3-13 below shows the back of the actual aircraft panel we used with most of the USB controlled gauges installed. Each gauge is mounted to the back of the aircraft panel with the USB cable connection located on the back of the aircraft flight instruments. These USB cables are connected to a USB hub, which is then connected to a USB port on the game computer.

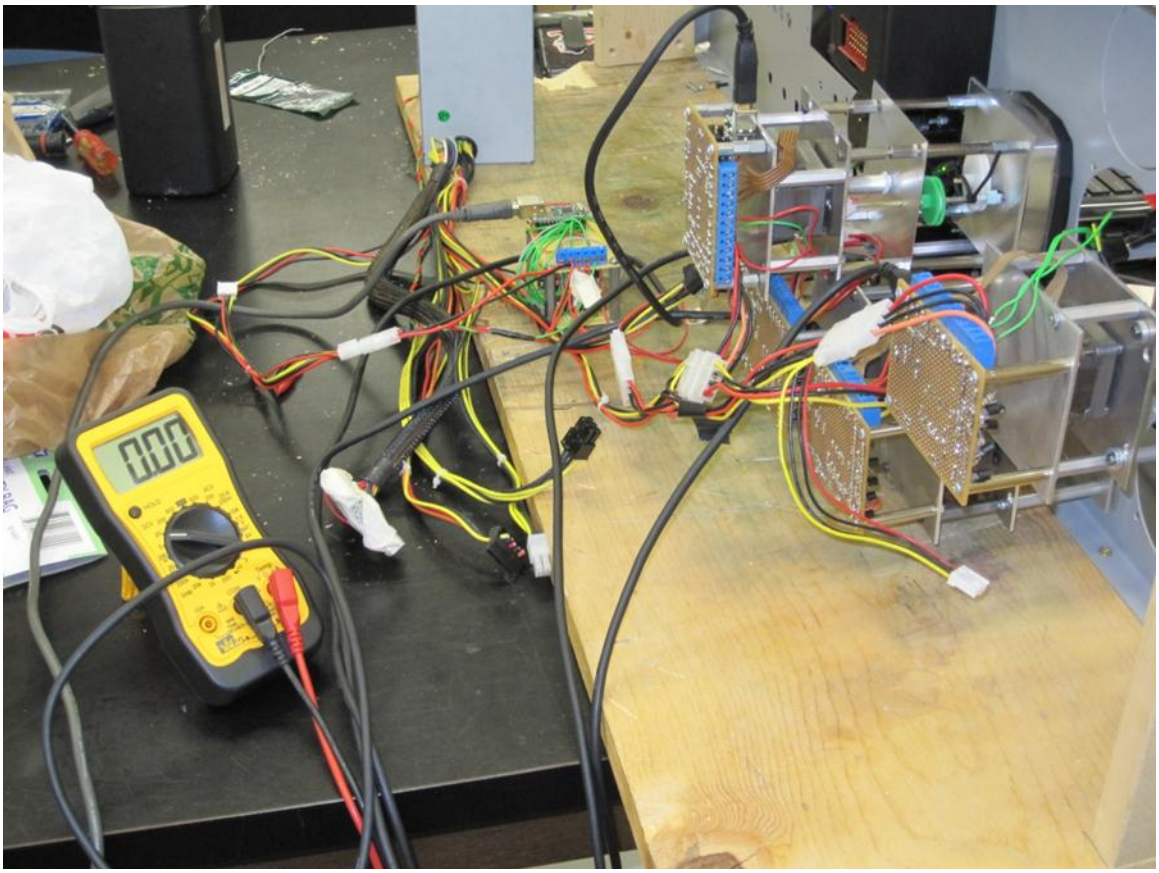


Figure 3-13. Back view of the aircraft panel received from Dave Graham of GoBosh Aviation with USB controlled gauges mounted. *Photo used with permission from Joseph Munera.*

To accomplish the requirements regarding the flight instruments there were a few options to consider. The options available were to mod a real world gauge to interface with the flight software, buy a simulation gauge kit that comes with all the parts pre-manufactured, build servo based gauges, or build stepper motor based gauges.

The best part of modifying real world gauges is that you get the most realistic look, as the gauges are in fact real. The other benefit of using real aircraft gauges is that all the faceplates, needles, glass coverings and in some cases the mechanical inner workings are already there so that all we would have to create would be the interface with the computer. Unfortunately, creating this interface was not an easy task. In the aircraft that we simulated all the gauges we are replicating are either barometric or gyroscopic, neither of which are easily simulated by a computer. For instance, the airspeed indicator can be simulated by a variable speed fan blowing into the barometer, but this was neither easier to implement nor more accurate than the other methods we have available. The other problem we had with real world gauges is the price. After calling a few airplane junkyards, like the one in Groveland, FL, we found that to buy salvaged

gauges would be over $100 per gauge. This is much higher than some of the other implementations and therefore was ruled out as an option.

The easiest option would have been to buy gauge kits from a supplier like SimKits. These kits usually come with all the parts needed pre-manufactured and ready for assembly. The only development work that would be required is to write the USB drivers for each gauge. The downfall of this implementation is the cost. Each kit costs well above $150. There were however a few gauges that we initially considered modeling with these kits just because the mechanics of the gauges were complicated enough to warrant spending the money.

Servo-based gauges are probably the most common gauge implementation in the flight simulator community. In this implementation, the needle of the gauge is turned by a servo, which is driven by a microcontroller. Servo-based gauges are fairly low cost and easy to design. The microcontroller sends a pulse to the servo and depending on the width of the pulse; the needle will turn to the appropriate angle. There are however a few limitations with servo-based gauges. The first problem we ran into in our design was that we planned to interface all of the gauges with an FTDI USB chip using bit-bang mode. Unfortunately, this interface did not allow us to send pulses to the servo with the accuracy we needed to implement the gauge smoothly. Fortunately we were able to design a timing circuit to interface the FTDI chip with the servo. This implementation is illustrated in figure 3-14. The other limitation that we ran into is that a servo only has a certain range of motion. For most gauges we can gear the servo so that it turns as far as we wish, but two of the gauges must be able to turn all the way around multiple times and would therefore require another implementation.

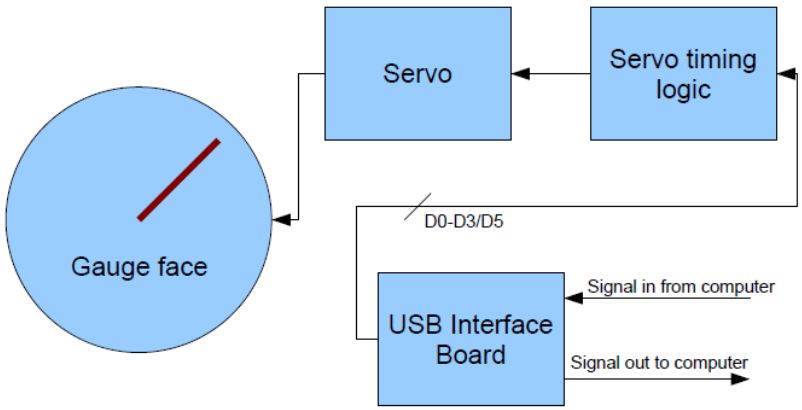


Figure 3-14: Block diagram of servo-based gauge. *Created by Lewis Vail*

The last implementation we had to choose from was a stepper-based gauge. Stepper based gauges are similar to servo based gauges except that they use stepper motors instead of servos. They are comparable in terms of cost and ease of implementation but they don't suffer from the two limitations servo based motors do. Instead of taking in a timed pulse like servos, they take in a 5 bit

digital input meaning that it can be interfaced directly with the FTDI chip. Also, stepper motors will turn indefinitely in either direction. They do however have a few limitations of their own. The first being that when you step the motor, it will turn the needle a discrete amount which can look choppy if the steps of the pulse are too large. To overcome this limitation, we used stepper motors that have a step of 1.8 degrees. They are then half-stepped to maximize fidelity. The other limitation that stepper motors have is that they support no option to supply feedback to the computer regarding the location of the needle. To overcome this we implemented an optic sensor to reset the needle to its home position at initialization. This stepper motor based implementation is illustrated in figure 3-15.

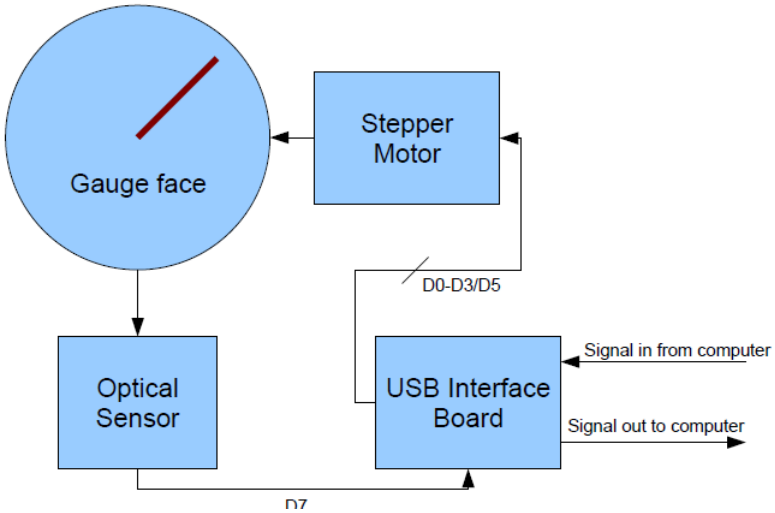


Figure 3-15: Block diagram of stepper motor based gauge. *Created by Lewis Vail*

Given the options above, we initially were going to be using two kit gauges, and four stepper motor gauges. Although servo gauges are low cost and fairly easy to implement, all of our gauges require fairly high range of motion that is not available with standard servos. The fact that the location of the needle is always known when using servos makes this the easiest one to code because there is no initialization needed but this implementation falls short of meeting the requirements for our gauges. Although stepper motor gauges are not quite as simple as servo-based gauges they are not too much harder to implement. They did take much more time to test and refine as the sensor circuit took a little while to properly calibrate and mount. We initially decided to model two of the gauges with kit gauges because those two required multiple motors turning independently. Although these two could have been implemented by using the techniques used for the other ones with some extra components, this seemed at first to be too difficult mechanically to justify the cost.

After considering all of these options we found out through contacting GoBosh Aviation that the Vice President of the company, Dave Graham, had set aside for us five of the six traditional flight instruments and a aircraft panel that we were able to use to house the simulated instruments. The five instruments that we

received from Dave Graham included the altimeter, attitude indicator, turn coordinator, airspeed indicator, and vertical airspeed. Figure 3-16 shows some of the instruments we received from GoBosh Aviation. We were allowed and given permission to take apart every flight instrument and use any of the parts that were useful for our flight simulator. The fact that we were able to use real flight instruments to build our simulated gauges helped to give our flight simulator the highest realism that we could hope to achieve. The only flight instrument that we did not obtain was the heading indicator. Fortunately this flight instrument was one of the simpler gauges to build and simulate. We were able to build and construct the heading indicator from scratch.

Figure 3-16. Instruments as received from GoBosh. *Photo by Joseph Munera*

3.3.1 Air Pressure Sensing Instruments

The air pressure sensing instruments are a collection of instruments that exist in the “six-pack” that on an actual aircraft generate their data from taking measurements based on air pressure. For our purposes this is just grouping these common gauges together, as we will not be utilizing any air pressure systems. These gauges include the Altimeter, the Airspeed Indicator, and the Vertical Speed Indicator.

3.3.1.1 Altimeter

The altimeter is used to measure altitude above a reference level, which is usually set at sea-level. This is done by measuring the local air pressure. Figure 3-17 shows the faceplate of a “three pointer” sensitive aircraft altimeter displaying an altitude of 10,180 ft. while Figure 3-18 shows the autocad drawings with the dimensions we used. For the altimeter we only implemented two pointers to indicate tens and thousands of feet.

Figure 3-17. Altimeter Face plate, dials, and barometric pressure adjustment knob. *This image has been released into the public domain by its author, Bsayusd at the Wikipedia project.*[7]

[7] (*2009, Nov.). SVG Drawing of Altimeter [Online]. Available: http://en.wikipedia.org/wiki/File:3-Pointer_Altimeter.svg*

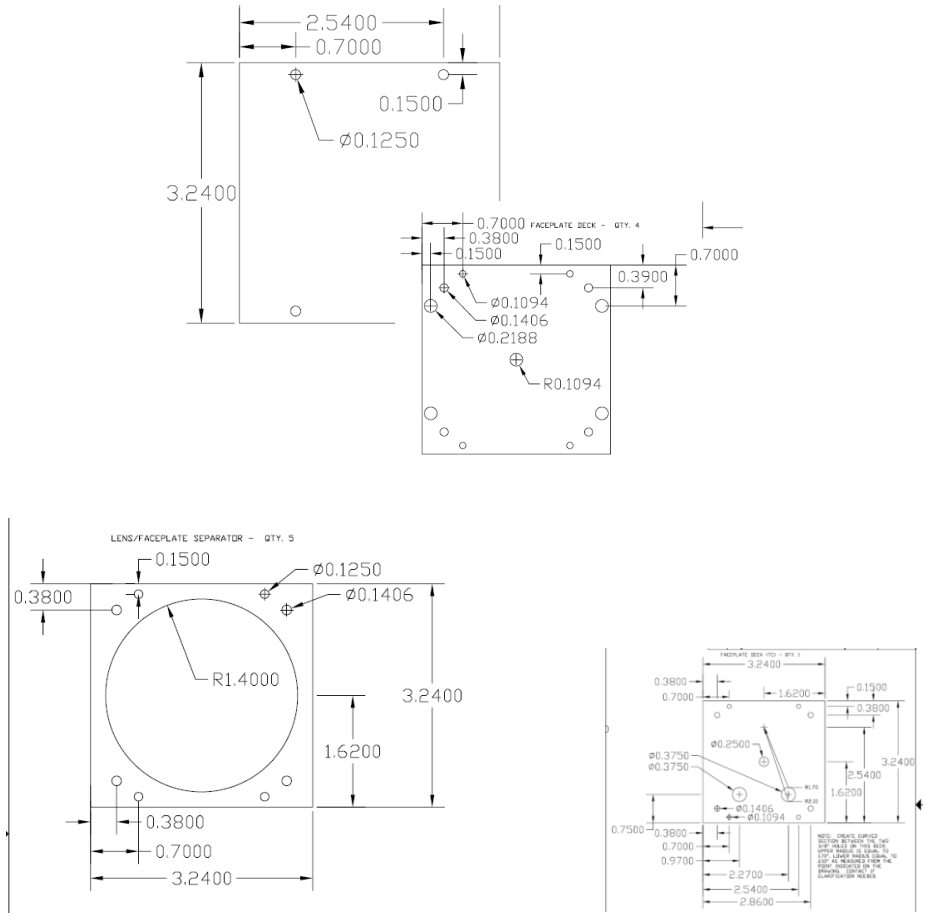


Figure 3-18. Autocad drawings with the measurements of the altimeter sheet aluminum decks. *Drawings by Robert Gysi*.

We had three options for constructing this gauge in order to be implemented in the cockpit. Of the following three gauges we have selected option two and three. This is because the stepper motors provided us with 360 degrees of rotation, which is necessary should the plane ever go above 1000 feet (this will happen with 100% certainty).

Three options that were considered:

I. Construction of the altimeter would have consisted of several parts. The gauge would be controlled by servo or stepper motors. A servo motor design would have required either the use of a 360 degree capable servo or the modification of a standard servo of 180 degrees. A standard servo could have been modified by either modifying the internal structure or by pairing a set of gears together with the proper gear ratio to spin a shaft at least 360 degrees.

Multiple motors may have been required to control the altimeter. The small hand and large hand of the altimeter could be controlled by a modified servo made to

run continuously. Each pointer would be USB controlled by an electronics board, which would l deliver direct feedback of the position of each pointer. The small indicator could have been controlled by a stepper motor. A second stepper motor could be used to regulate the air pressure scale by using a dial located at the lower left land hand side of the gauge.

II. The altimeter could have been built using 3 stepper motors. There would have been a motor to move the 100 ft hand and 1000 ft hand. The third stepper motor could have been used for the plate linked to a potentiometer to adjust the settings for the barometric pressure. The pointers, small indicator, and dial for the barometric pressure would be USB controlled by an electronics board, which will deliver direct feedback of the position and setting.

A problem arises when using stepper motors. There is no way to know when the shaft is positioned at zero or home. To determine where the starting point or zero is an optical sensor is used to sense when the motor is moved to the start position.

III. A real commercial altimeter could be used for the flight simulator. It required accurately generating slow varying pressures within small fractions of a PSI. Figure 3-19 shows the inherent complexity involved when trying to use a real aircraft flight instrument in building a flight simulator cockpit. The mechanical parts of the aircraft flight instrument are removed by disassembling the aircraft flight instrument. Stepper motors are then placed inside the aircraft flight instrument.

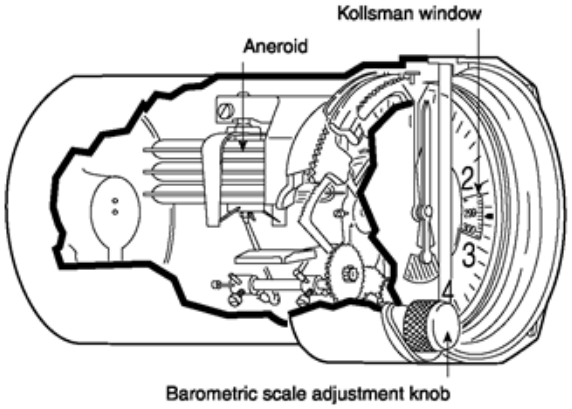


Figure 3-19. Actual altimeter components. *As a work of the U.S. federal government, the image is in the public domain*[8].

We were only able to utilize the faceplates, needles, and glass frame of the real altimeter we obtained from GoBosh Aviation to build our simulated altimeter. These components were added to the faceplate deck of the layered sheet metal

[8] (*2009, Nov.). Drawing of Altimeter [Online]. Available: http://en.wikipedia.org/wiki/File:Sens_alt_components.PNG*

and hex spacer flight instrument design. A single stepper motor was mounted in the center of the motor deck and used to turn the needles of the altimeter. A piece of 3/8" round aluminum tube was first mounted to the shaft of the stepper motor. Next brass tubing of 1/16" was slipped inside brass tubing of 3/32" and each was allowed to spin freely within each other. To simulate the altimeter needle readings 48 pitch gears of 12, 24, 48, and 60 teeth were used to spin the shafts at a 1:10 ratio to indicate the 100 and 1000 feet readings.

3.3.1.2 Airspeed Indicator

The airspeed indicator measures the indicator airspeed of an aircraft via a probe on the fuselage. In our application, our airspeed indicator is fed data from X-Plane, simulating the mechanical operation of a pressure driven system. In an actual gauge, the speed indicated is relative to the surrounding air by measuring the ram-air pressure in the aircraft's pitot tube. As with our other instruments this is powered over USB and based on the FTDI chipset previously mentioned in our microcontroller design section. All of the code required to drive the motor is developed as a plug in for X-Plane. Figure 3-20 (below) shows an example of an airspeed indicator, similar to the one utilized in the GoBosh G700S, while Figure 3-21 shows the autocad drawings with the dimensions we used.

Figure 3-20. Airspeed Indicator Faceplate - Wikipedia. *Used with permission under the GNU Free Documentation License*[9]

[9] (*2009, Nov.). Drawing of Airspeed Indicator from FAA Instrument Flying Handbook [Online]. Available: http://en.wikipedia.org/wiki/File:True_airspeed_indicator-FAA.SVG*

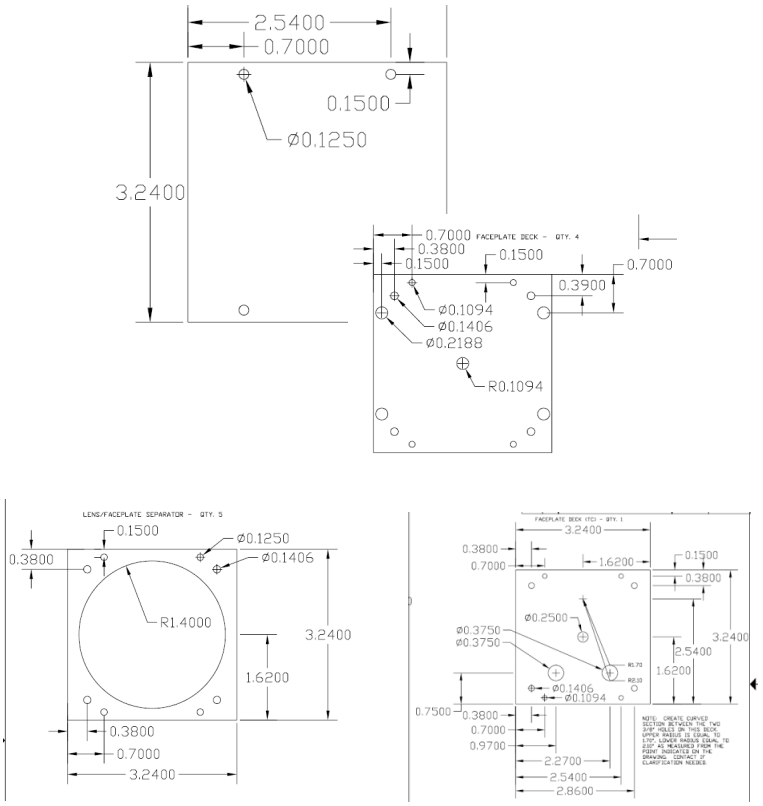


Figure 3-21. Autocad drawings with the measurements of the airspeed indicator sheet aluminum decks. *Drawings by Robert Gysi*.

Four options that were considered:

I. Construction of the airspeed indicator would consist of several parts. The gauge would be controlled by servo or stepper motors. A servo motor design would require either the use of a 360 degree capable servo or the modification of a standard servo of 180 degrees. A standard servo could be modified by either modifying the internal structure or by pairing a set of gears together with the proper gear ratio to spin a shaft at least 360 degrees.

II. The airspeed indicator only has a single pointer to control. A single servo motor capable of 360 degrees, a modified 180 degree servo motor, or a 180 degree standard single servo motor paired with a set of gears could have been used to control the pointer on the airspeed indicator.

III. A single stepper motor capable of turning 360 degrees could be used instead of a single servo motor to control the pointer of the vertical speed indicator. A problem arises when using stepper motors. There is no way to know when the shaft is positioned at zero or home. To determine where the starting point or zero is an optical sensor could be used to sense when the motor is moved to the start position.

IV. A real commercial airspeed indicator could also be used for the flight simulator. It would require accurately generating slow varying pressures within small fractions of a PSI. Figure 3-22 shows the inherent complexity involved when trying to use a real aircraft flight instrument in building a flight simulator cockpit. The mechanical parts of the aircraft flight instrument would have needed to be removed by disassembling the aircraft flight instrument. Servo motors or stepper motors would then be placed inside the aircraft flight instrument.

For the airspeed indicator we have selected option three and four. This is because stepper motors provided 360 degrees of rotation. While our airspeed indicator never had to go around more than once, we do need to ensure that the extreme values can be represented. Servo motors unfortunately lacked resolution at high angles.

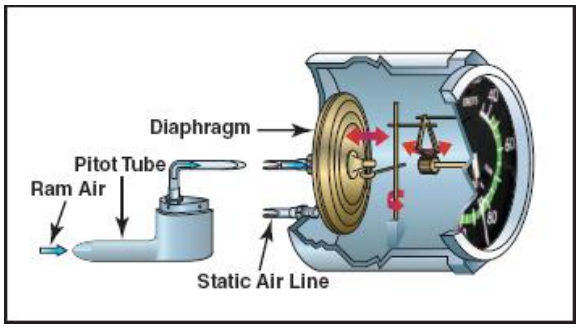


Figure 3-22. Components of an actual airspeed indicator. *As works of the U.S. federal government, all FAA images are in the public domain*.[10]

We were only able to utilize the faceplate, needle, and glass frame of the real airspeed indicator we obtained from GoBosh Aviation to build our simulated airspeed indicator. These components were added to the faceplate deck of the layered sheet metal and hex spacer flight instrument design. A single stepper motor was mounted in the center of the motor deck and used to turn the needle of the airspeed indicator. A piece of 3/8” round aluminum tube was mounted to the shaft of the stepper motor to extend the shaft length to the faceplate.

3.3.1.3 Vertical Speed Indicator

The vertical speed indicator measures the speed at which an aircraft rises and falls - its vertical speed. If the nose is banked upward and the vertical speed drops starts to decrease for example, this would indicate that the aircraft has stalled, or lost lift. Also, for example if the nose is banked downward, the vertical speed indicator would now turn counter-clockwise to provide the vertical speed as you decrease in altitude (and increase in indicator airspeed as well). Figure 3-23 shows the faceplate of a simple vertical speed indicator, similar to the one we

[10] (*2009, Nov.). Airspeed Indicator Cutaway - Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/File:ASI-operation-FAA.png*

implemented in our simulator. Figure 3-24 shows the autocad drawings with the dimensions we used.

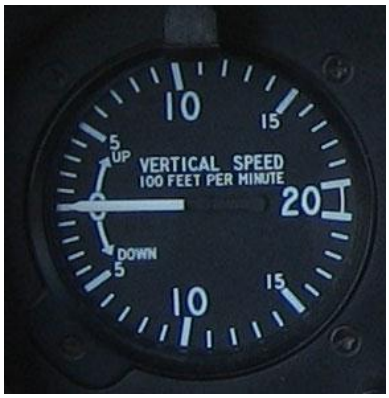
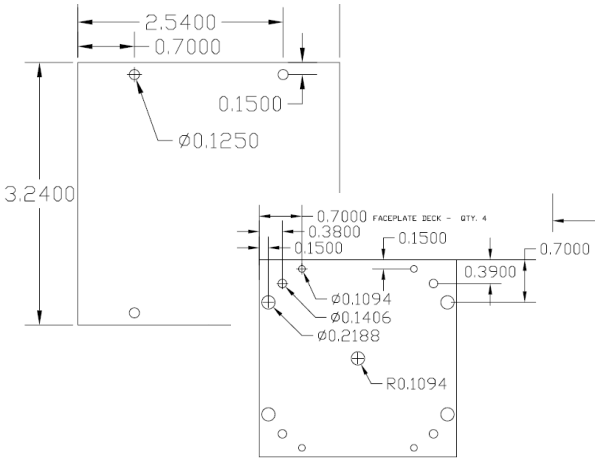


Figure 3-23. Face plate of the vertical speed indicator. *This work has been released into the public domain by the copyright holder, Benet Allen*.[11]



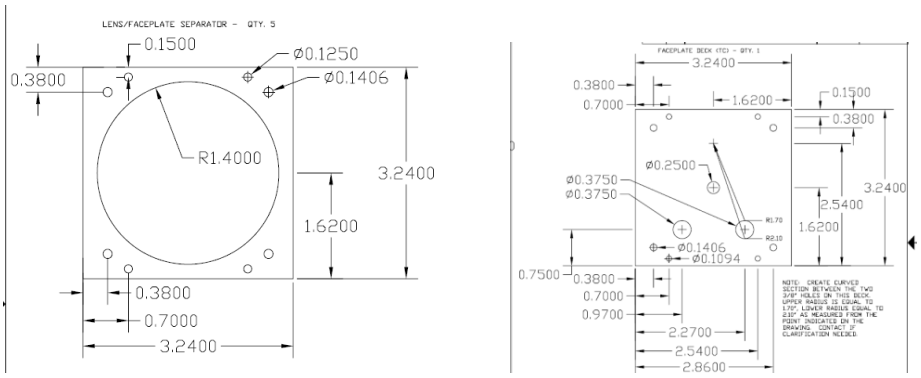


Figure 3-24. Autocad drawings with the measurements of the vertical speed indicator sheet aluminum decks. *Drawings by Robert Gysi*

[11] (*2009, Nov.). Vertical Speed Indicator - Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/File:R22-VSI.jpg*

Four options that were considered:

I. Construction of the vertical speed indicator would have consisted of several parts. The gauge would be controlled by servo or stepper motors. A servo motor design will require either the use of a 360 degree capable servo or the modification of a standard servo of 180 degrees. A standard servo can be modified by either modifying the internal structure or by pairing a set of gears together with the proper gear ratio to spin a shaft at least 360 degrees.

II. The vertical speed indicator only has a single pointer to control. A single servo motor capable of 360 degrees, a modified 180 degree servo motor, or a 180 degree standard single servo motor paired with a set of gears could have been used to control the pointer on the vertical speed indicator.

III. A single stepper motor capable of turning 360 degrees could be used instead of a single servo motor to control the pointer of the vertical speed indicator. A problem arises when using stepper motors. There is no way to know when the shaft is positioned at zero or home. To determine where the starting point or zero is an optical sensor could be used to sense when the motor is moved to the start position.

IV. A real commercial vertical speed indicator could also be used for the flight simulator. It would require accurately generating slow varying pressures within small fractions of a PSI. Figure 3-25 shows the inherent complexity involved when trying to use a real aircraft flight instrument in building a flight simulator cockpit. The mechanical parts of the aircraft flight instrument would have to be removed by disassembling the aircraft flight instrument. Servo motors or stepper motors would then be place inside the aircraft flight instrument.

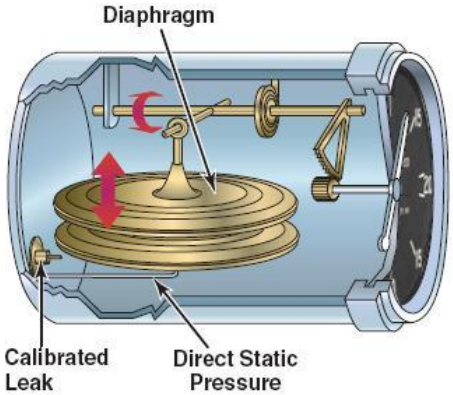


Figure 3-25. Components of an actual vertical speed indicator. *This work is in the public domain in the United States because it is a work of the United States Federal Government*[12]

[12] (*2009, Nov.). Vertical Speed Indicator (FAA) - Wikipedia [Online]. http://en.wikipedia.org/wiki/File:Faa_vertical_air_speed.JPG*

For the vertical speed indicator we have selected option three. This is because stepper motors provide 360 degrees of rotation. Again, just like the airspeed indicator, we will never need to go beyond just short of 360 degrees, but since we will need to cover large angles of rotation, only a stepper motor can provide us with the resolution we require.

We were only able to utilize the faceplate, needle, and glass frame of the real vertical speed indicator we obtained from GoBosh Aviation to build our simulated vertical speed indicator. These components were added to the faceplate deck of the layered sheet metal and hex spacer flight instrument design. A single stepper motor was mounted in the center of the motor deck and used to turn the needle of the vertical speed indicator. A piece of 3/8" round aluminum tube was mounted to the shaft of the stepper motor to extend the shaft length to the faceplate.

3.3.2 Gyroscopic Instruments

The instruments in this category all are based on gyroscopes. They help the pilot determine the position and status of the aircraft in flight. While during day time flying a pilot may be able to determine if his wings are level or if he is at level flight (or climbing or descending), but in times of low light levels, it may be impossible to see the ground or may become disoriented and not know which way is up, down, left, or right.

3.3.2.1 Attitude Indicator (Artificial Horizon)

The attitude indicator displays the aircraft's orientation relative to the earth. As seen in Figure 3-26, we see that the gauge has two different colored areas: one blue to represent the sky and one black (in most case this colored brown) to represent the earth. The hatch indicates the attitude of the aircraft. This gauge does not solely work in an up and down fashion. Since it is gyroscope based on an actual aircraft, it also rotates to the left and the right indicating the bank or roll of the aircraft. Figure 3-27 shows the autocad drawings with the dimensions we would have used had we not received an actual attitude indicator.

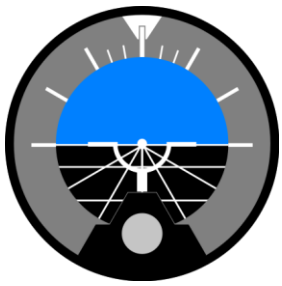


Figure 3-26. Face plate of the attitude indicator. *Used with permission under the GNU Free Documentation License*[13]

[13] (*2009, Nov.). Artificial Horizon - Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/File:Attitude_indicator_level_flight.svg*

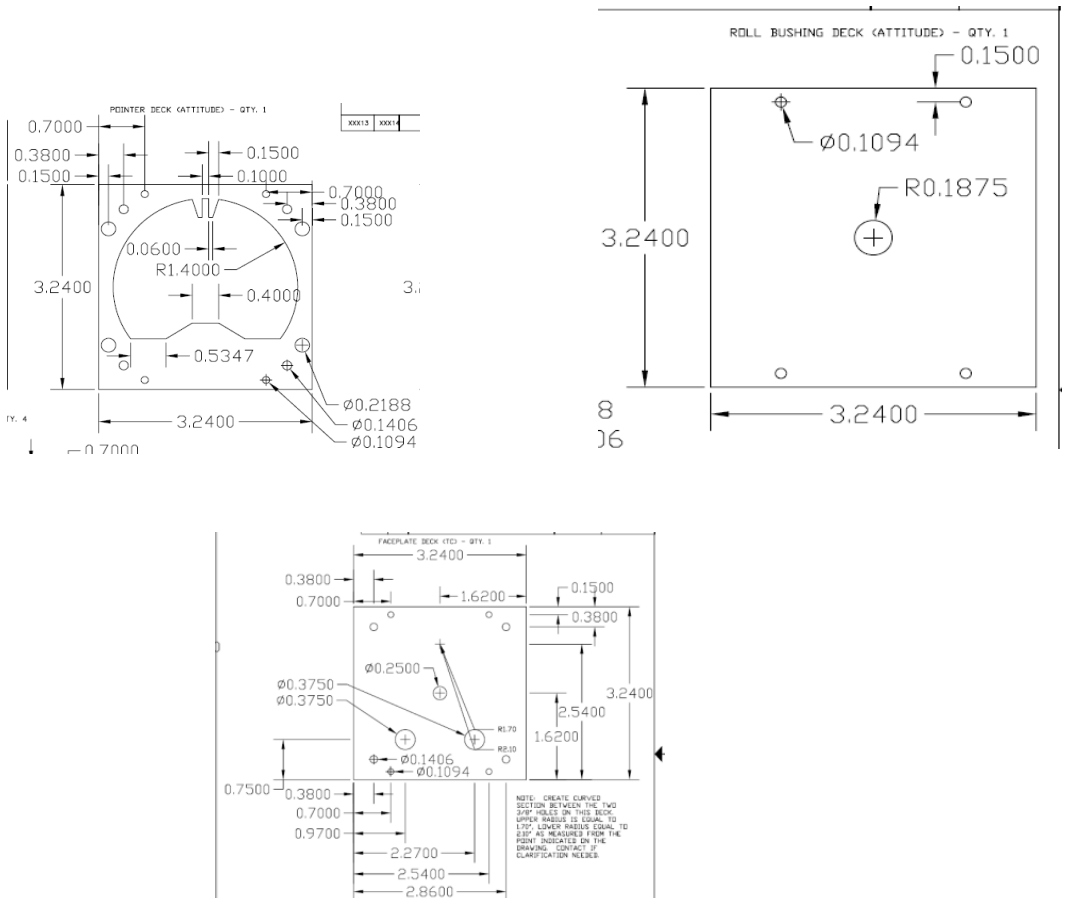


Figure 3-27. Autocad drawings with the measurements of the attitude indicator sheet aluminum decks. *Drawings by Robert Gysi*

Five options that were considered:

I. Construction of the attitude indicator consists of several parts. The gauge will be controlled by servo or stepper motors. A servo motor design may require either the use of a 360 degree capable servo or the modification of a standard servo of 180 degrees if a roll indication of 360 degrees is desired. A standard servo can be modified by either modifying the internal structure or by pairing a set of gears together with the proper gear ratio to spin at least 360 degrees.

II. Two standard 180 degree servo motors can be used if the desired roll indication does not require 360 degrees. The two servo motors will drive the scales which are able to turn left or right, as well as move up and down. One of the servo motors will control upward and downward motions. The second servo motor will be used to control the turning motion. With this design the roll indication will have a maximum at 95 degrees to the left and at 95 degrees to the right. A centrally located dial underneath the attitude indicator will show proper indication of the horizon.

III. A pair of stepper motors could be used instead of servo motors to control the scales. A problem arises when using stepper motors. There is no way to know where the scales are positioned. To determine where the starting point or zero is an optical sensor could be used to sense when the motor is moved to the start position.

IV. A real commercial attitude indicator could be used for this simulator. The attitude indicator incorporates a gyro which is designed so the indicators do not move. The instrument housing bolted to the aircraft is what moves around the indicator. The mechanical parts of the aircraft flight instrument would have to be removed by disassembling the aircraft flight instrument. Servo motors or stepper motors would then be place inside the aircraft flight instrument.

V. Another option would have been to buy a simulated instrument kit for this type of flight instrument. This could be purchased from SimKits.com or from Flight Illusion, both of which are companies located in the Netherlands. Figure 3-28 shows an example of a kit version available from Flight Illusion. This (along with a similar one from SimKits) simulates the motions of the gyroscope ball by utilizing a moving plate. It should also be noted that the SimKits gauge, while it has a X-Plane plug-in available it is only sold for "professional use" and is priced accordingly at $2000. The Flight Illusion gauge however includes a free X-Plane plug-in, so there would be no additional costs or development required to implement.

Figure 3-28. Flight Illusion Attitude Indicator Gauge. *Photo used with permission from Mark Verschaeren.*[14]

For the attitude indicator we chose option three and four. The attitude indicator was one of our toughest and challenging flight instruments to simulate. We were fortunate enough to get a real attitude indicator from GoBosh Aviation. The attitude indicator was taken apart and the gyro was removed. A stepper motor was mounted on the inside of the yoke to drive the pitch indicator. A second stepper motor was mounted on the back to control the roll indicator. Figure 3-29

[14] For e-mail response granting permission to use, see Appendix C

on the following page shows the setup of the actual attitude indicator and the mounted stepper motors.

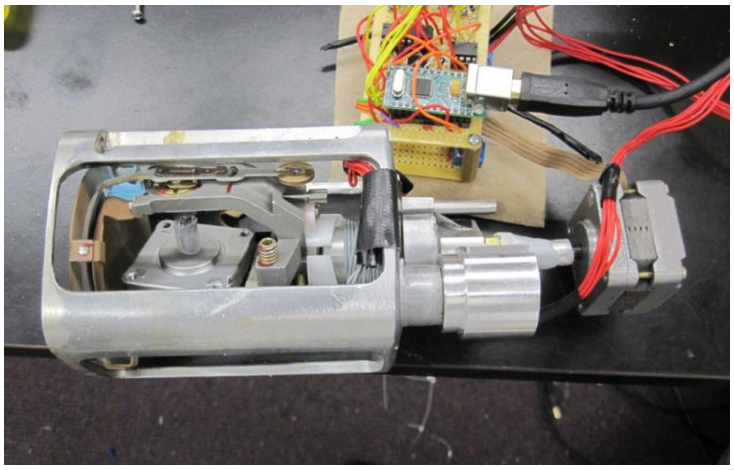


Figure 3-29. Actual Attitude Indicator during modification process. *Photo by Lewis Vail*

3.3.2.2 Turn Coordinator

The turn coordinator provides to the pilot information about the yaw, roll and coordination of the turn being performed[15]. If the turn is coordinated than the ball that exists in track in the bottom of the gauge (see Figure 3-30), will remain in between the two black lines. If this ball moves to the left section of the track it is known as skid and to the right section of the track it is known as slip (these are both when the aircraft is making a turn to the right). Figure 3-31 shows the autocad drawings with the dimensions we used.

Figure 3-30. Turn Coordinator *Used with permission under the GNU Free Documentation License*[16]

[15](*2009, Dec.). Turn Coordinator - Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/Turn_coordinator*

[16] (*2009, Dec.). Turn Coordinator Drawing - Wikipedia [Online]. Available:* http://en.wikipedia.org/wiki/File:Turn_indicator.png

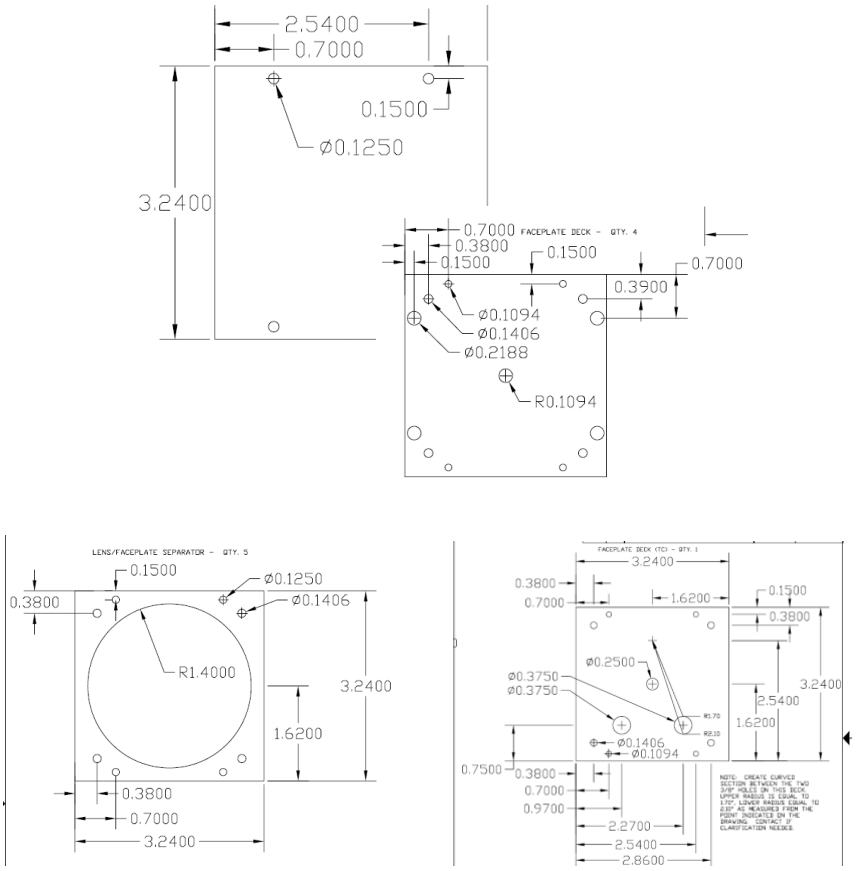


Figure 3-31. Autocad drawings with the measurements of the turn coordinator sheet aluminum decks. *Drawings by Robert Gysi*

Five options that were considered:

I. Construction of the turn coordinator will consist of several parts. The gauge will be controlled by servo or stepper motors. A servo motor design will not require the use of a 360 degree capable servo or the modification of a standard servo of 180 degrees.

II. The turn coordinator can be built using two standard 180 degree servo motors. The first servo motor will control the aircrafts turn rate. The second servo motor will be used to control the slip indication.

III. A pair of stepper motors could be used instead of servo motors to control the turn rate and slip indication. A problem arises when using stepper motors. There is no way to know where the turn rate and slip indication are positioned. To determine where the starting point or zero is an optical sensor could be used to sense when the motor is moved to the start position.

IV. A real commercial turn coordinator could be used for this simulator. The turn coordinator incorporates a gyro which is designed so the indicators do not move.

The instrument housing bolted to the aircraft is what moves around the indicator. Figure 3-32 shows the inherent complexity involved when trying to use a real aircraft flight instrument in building a flight simulator cockpit. The mechanical parts of the aircraft flight instrument would have to be removed by disassembling the aircraft flight instrument. Servo motors or stepper motors would then be place inside the aircraft flight instrument.

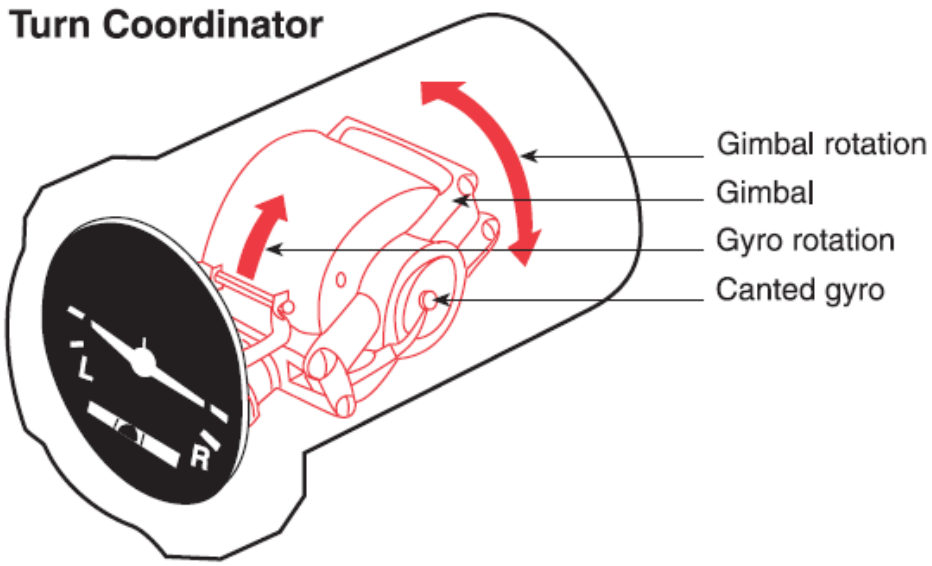


Figure 3-32. Components of an actual turn coordinator. *As a work of the U.S. federal government, the image is in the public domain*.[17]

V. Another option would be to buy a simulated instrument kit for this type of flight instrument. Like with the Attitude indicator, there are two manufacturers of this gauge, SimKits and Flight Illusion. SimKits has one with an existing faceplate, while Flight Illusion does not have this particular one. If we are to order this gauge versus building it, we would need to order from SimKits.

For the turn coordinator we have decided to go with options three and four. We were only able to utilize the faceplate, airplane, and glass frame of the real turn coordinator we obtained from GoBosh Aviation to build our simulated turn coordinator. These components were added to the faceplate deck of the layered sheet metal and hex spacer flight instrument design. Two stepper motors were mounted side by side on the motor deck offset from the center. The airplane was fastened to brass hobby tubing through the center of the faceplate with a gear fastened on the opposite end of the tube. Another gear was fastened to one of the stepper motor shafts and allowed to mesh with the brass tubing in order to spin the airplane clockwise or counter clockwise. The second stepper motor had a piece of 3/8” round aluminum tube attached to its shaft in order to extend it. On

[17] *(2009, Dec.). File: Turn_indicators.png - Wikipedia [Online]. http://en.wikipedia.org/wiki/File:Turn_indicators.png*

the tip of the aluminum shaft was a black flag that was used to indicate the slip or skid ball.

3.3.2.3 Heading Indicator

The heading indicator in the standard instrument gauge setup on an aircraft functions as a compass pointing in heading of travel of the aircraft. This however does not function exactly like a wet compass as it is not affected by the downward slope of the Earth's magnetic field.[18] Figure 3-33 shows a common design for heading indicator face plates, while Figure 3-34 shows the autocad drawings with the dimensions we used.

Figure 3-33. Face plate of the heading indicator. *Used with permission under the GNU Free Documentation License*[19]

[18] (*2009, Dec.). Heading Indicator - Wikipedia [Online]. http://en.wikipedia.org/wiki/Heading_indicator*

[19] (*2009, Dec.). Heading Indicator Drawing - Wikipedia [Online]. Available:* http://en.wikipedia.org/wiki/File:Heading_Indicator.png

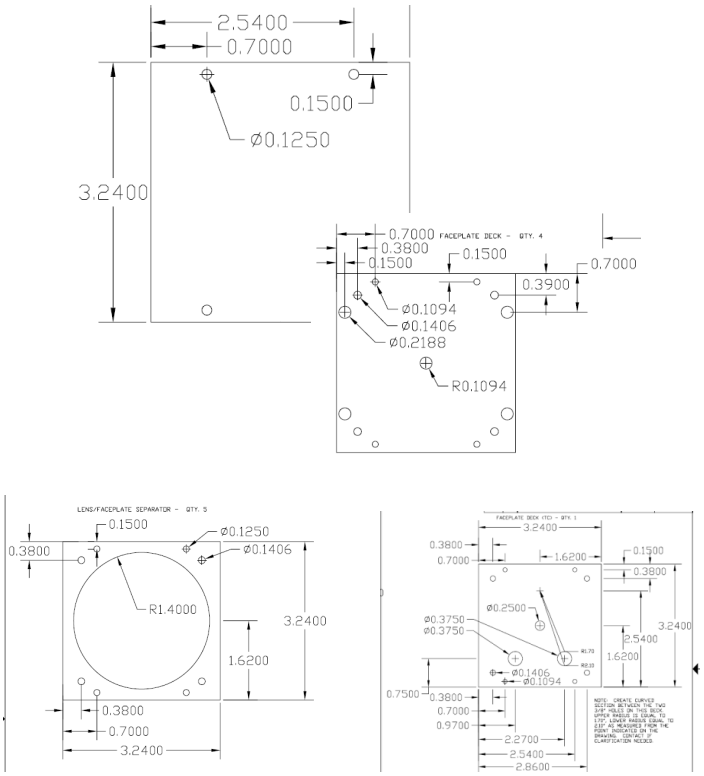


Figure 3-34. Autocad drawings with the measurements of the heading indicator sheet aluminum decks. *Drawings by Robert Gysi*

Four options that were considered:

I. Construction of the heading indicator will consist of several parts. The gauge will be controlled by a servo or stepper motor. A servo motor design will require either the use of a 360 degree capable servo or the modification of a standard servo of 180 degrees. A standard servo can be modified by either modifying the internal structure or by pairing a set of gears together with the proper gear ratio to spin a shaft at least 360 degrees.

II. A single modified servo motor can be used to turn right or left. The heading indicator may also have two dials for this aircraft. The left dial will indicate the position of the gyro compass. The right dial will be used to adjust the heading bug to the proper heading for use with auto pilot.

III. A single stepper motor could be used instead of a servo motor to control the turn from right or left. A problem arises when using stepper motors. There is no way to know were the heading is positioned. To determine where the starting point or zero is an optical sensor could be used to sense when the motor is moved to the start position.

IV. A real commercial heading indicator most likely cannot be used for this simulator. The heading indicator incorporates a gyro which is designed so the indicators do not move. The instrument housing bolted to the aircraft is what

moves around the indicator. Figure 3-35 shows the inherent complexity involved when trying to use a real aircraft flight instrument in building a flight simulator cockpit. The mechanical parts of the aircraft flight instrument would have to be removed by disassembling the aircraft flight instrument. Servo motors or stepper motors would then be place inside the aircraft flight instrument.

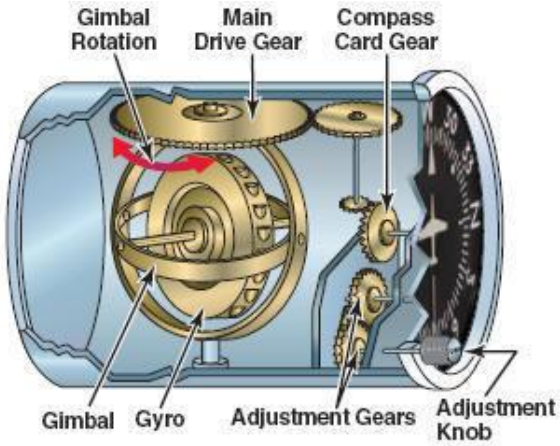


Figure 3-35. Components of an actual heading indicator. *Photo used with permission from Bob Miller*. [20]

For the heading indicator we selected option three. This is because stepper motors provide 360 degrees of rotation. In this gauge we needed the capability to continuously rotate the instrument if someone was piloting the aircraft in a circle formation.

The head indicator is the one gauge we did not receive from GoBosh Aviation. Fortunately it was not an extremely difficult gauge to build and construct. A single stepper motor is used to spin the compass rose of the heading indicator. A 3/8" round aluminum tube is attached to the shaft of the stepper motor to extend the length. The compass rose is attached to the end of this tube. A square cutout of clear acrylic with an airplane marking is placed above the compass rose to indicate the heading of the aircraft. Figure 3-36 shows the heading indicator without the lens.

[20] For e-mail response granting permission to use, see Appendix C

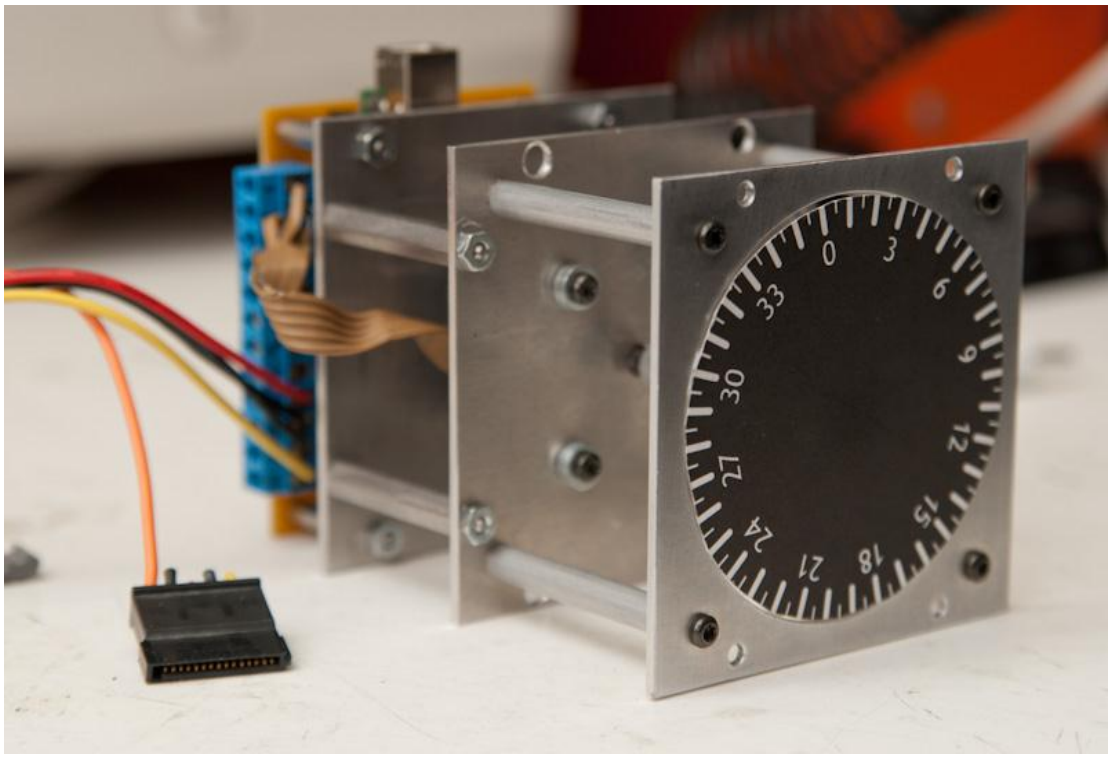


Figure 3-36. Completed Heading Indicator (without lens faceplate). *Photo by Lewis Vail*

3.3.3 Gauge Design

This section details the construction of the flight instruments including the housing, motors, faceplates, and needles. Due to the similar nature of many of these gauges, we were able to use a common design and then only make slight adjustments to take care of any differing features.

3.3.3.1 Stepper Motor Design

This design takes advantage of the two strengths involved when using a stepper motor for a simulated aircraft flight instrument or gauge. The motor can rotate continually because there is no mechanical stop. There is also quite a bit of torque at your disposal. When using stepper motors to design the aircraft flight instrument an optical interrupter must be incorporated into the design. An optical interrupter will be used to sense the zero position during power up and execution of the reset command.

The stepping motor is centered and mounted directly behind the faceplate assembly. For this type of aircraft flight instrument there is a rotary encoder which is mounted in the lower corner of the motor deck. The optical interrupter is located directly above the stepper motor. The optical interrupter is mounted on the decks front surface with the leads pointing toward the circuit deck. Just behind the motor deck is the circuit deck which is followed by the rear deck.

There is no unique starting position for the stepper motor. When powered up the stepper motor shaft is in an unknown rotational position.

A 400 step per revolution stepper motor is used in this design. To boost the number of steps to 800 the stepper motor is half stepped.

The difference between the motor positioned desired and the current position is calculated with each pass through the interrupt service routine. The code executes a return from interrupt when a difference of zero indicates no need for movement.

3.3.3.2 Flag Interrupter

The flag interrupter consists of an L-Shaped thin piece of sheet metal. A machine screw and nut hold the flag interrupter in place. The flag interrupter is bent on the outer end so that as the needle or faceplate rotates, the flag interrupter will pass through the gap for the optical interrupter. The flag interrupter can be constructed out of any rigid opaque material or aluminum.

3.3.3.3 Motor Deck

The rotary encoder, stepper motor, and optical interrupter are supported by the motor deck. The stepper motor is mounted on the decks rear surface with the motor shaft facing forward. The stepper motor has no particular up or down orientation. Figure 3-37 shows a picture of the motor deck on the left.

Figure 3-37. The motor deck is pictured on the left. *Photo used with permission from Joseph Munera.*

Just above the stepper motor is where the optical interrupter is mounted on the front surface of the motor deck.

To determine the best direction of movement a difference not equal to zero is used. This requires more than just observing the sign of the difference. Ideally we want to move in the direction that will minimize the number of steps required. Even though both are positive differences movement from 0 to 1 is in one direction while movement from 0 to 799 should go the other way.

By looking at the sign of the difference a direction is set, forward movement is implied with a positive difference. Then the number of half steps equivalent to half a rotation is compared to the magnitude of the difference. The direction flag is complemented if the magnitude is larger. It is then shorter to go the other way.

3.3.3.4 Optical Interrupter

The optical interrupter is used to during initialization to tell the gauge when the needle is in the home position. It consists of two components, a sender and a receiver.

The receiver is a light sensor that goes to 0 when it senses light. It consists of a P4537 CdS photocell connected to ground and a 5KΩ resistor connected to VCC wired in series. The voltage is taken between them and fed into the FTDI chip at D7 (see figure 3-27). When no light is hitting the photocell its resistance is much higher than 5KΩ and the D7 goes high. When light hits the photocell its resistance is much lower than 5KΩ and D7 goes low. The 5KΩ value mentioned above is just a theoretical value arrived at by examining the P4537 datasheet and it may change once we begin testing. A simplified block diagram of this system is located below in figure 3-35.

The sending component is simply an LED. It emits light that it picked up by the photocell. During initialization the LED is lit. When the needle finally gets around to the home position, the flag interrupter blocks the LED from lighting the photocell and the signal to stop is sent to FTDI chip.

3.3.3.5 Mechanical Construction

Figure 3-36 shows the internal structure of a dual needle aircraft flight instrument, along with dimensions required to build the gauge. The measurements are in mm and the following labeled parts are:

A) Face Plate
B) Lens
C) Motor 1
D) Motor 2
E) Printed Circuit Board 1
F) Printed Circuit Board 2

The entire aircraft flight instrument structure is made from a variety of components including Aluminum, and metal tubes with a diameter of 2mm to 5mm. In addition, the glass window of the gauge isconstructed using acrylic or plexi-glass. Figure 3-38 show the implementation of the autocad drawings into the basic structure and foundation of our simulated flight instruments As you can see the aluminum rods connect the front and rear ends of the gauges with the sheets of aluminum providing stability to the whole assembly including a place to mount components to.

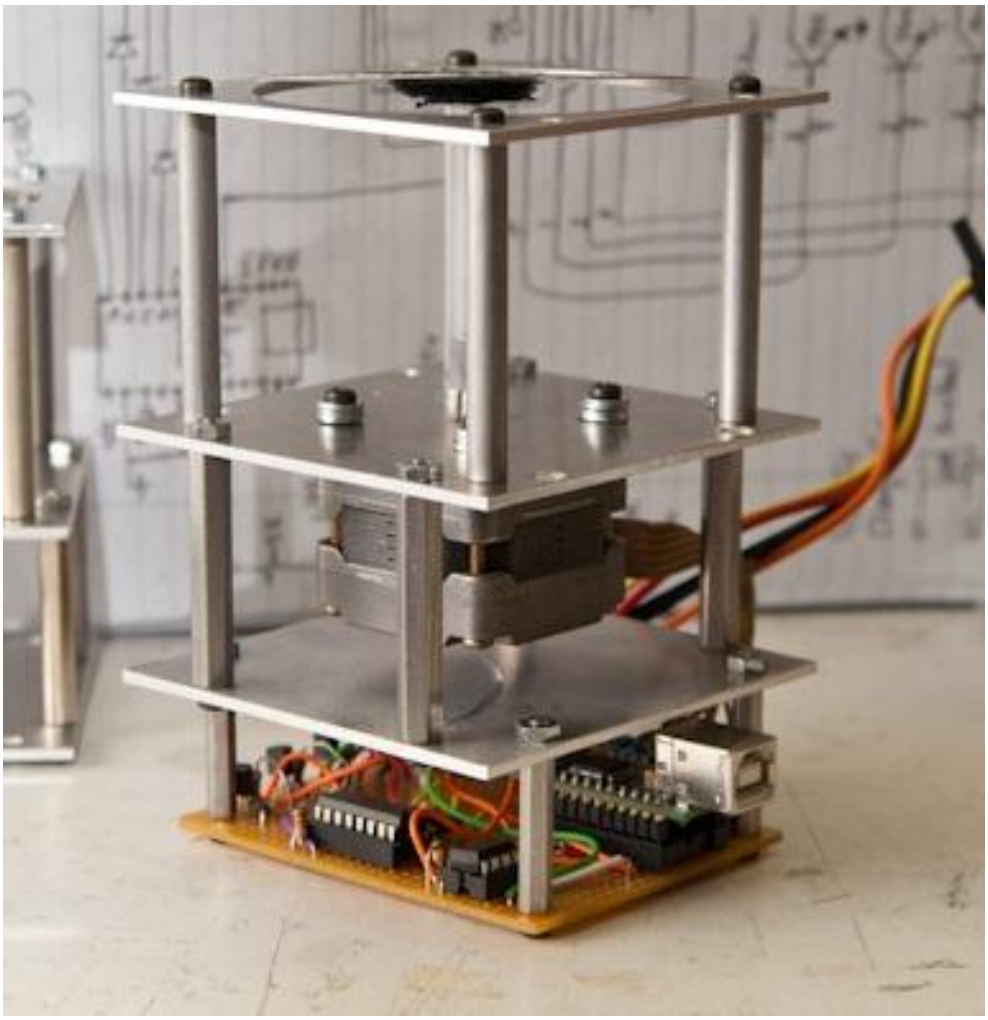


Figure 3-38. Constructed flight instrument. *Photo used with permission from Lewis Vail.*

Figure 3-39 shows some of the finished faceplate designs that were mounted to the front of the aircraft flight instrument gauge frames. The faceplates are all from the actual aircraft flight instruments we received from GoBosh Aviation except for the heading indicator. All the needles were also removed from the actual flight instruments and utilized in the simulated aircraft instruments.

Figure 3-39. Assembled face plates for a various flight instruments. *Photo used with permission from Lewis Vail*

3.3.3.6 Prototype Gauge

In order to validate the research we performed, we built a prototype airspeed indicator around a Futaba S3003 servo motor purchased from Central Florida Hobbies. Using the function generator in the Senior Design Lab, we were able to perform a test of the circuitry. Other components used in the construction of the prototype consist of plywood, nylon gears (from a clock kit), aluminum shafts, circular pipe, screws, and epoxy. Figure 3-40 shows this prototype during testing in the Senior Design lab.

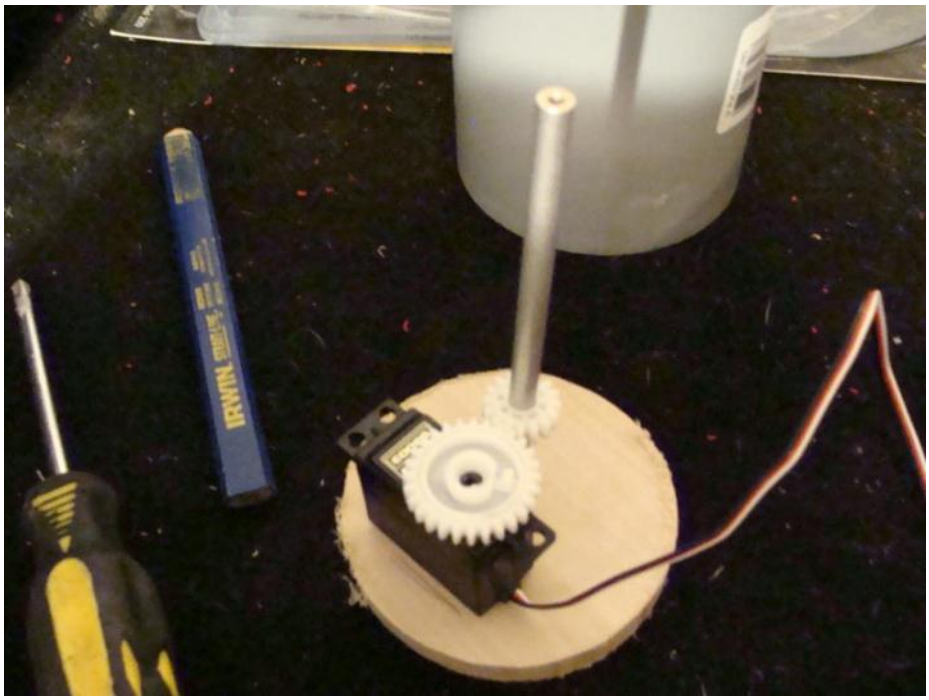


Figure 3-40. Flight instrument assembly consisting of a servo motor, nylon gears, circular wood cutout and aluminum shaft. *Photo by Joseph Munera.*

A circular saw was used to make a circular cutout from the plywood. The circular cutout is used to mount the servo motor and shaft. The aluminum shaft is positioned in the center of the circular cutout. A power drill is used to drill out a small section of the wood so that the aluminum shaft can sit in the center of the circular cutout and rotate freely.

The nylon gears are drilled out so that the smaller gear would be able to fit onto the aluminum shaft and the larger gear would be able to be mounted on the Futaba servo motor shaft.

Figure 3-41 shows the aluminum shaft with the gear in place and also the Futaba servo motor with the larger gear mounted on the top of the shaft. The circular wood cutout is now glued and sealed to the back of the circular pipe. The Futaba servo motor is glued and two screws are used to mount it to the circular wood cutout. The aluminum shaft sits inside its drilled fitting. Three screws are used to hold the aluminum shaft in place preventing it from slipping out of its setting.

The wires for the Futaba servo motor hang out the back of the circular wood cutout. The Futaba S3003 servo motor has three colored wires. The black wire is the ground, the red wire is for the power and the white wire is for the PWM signal.

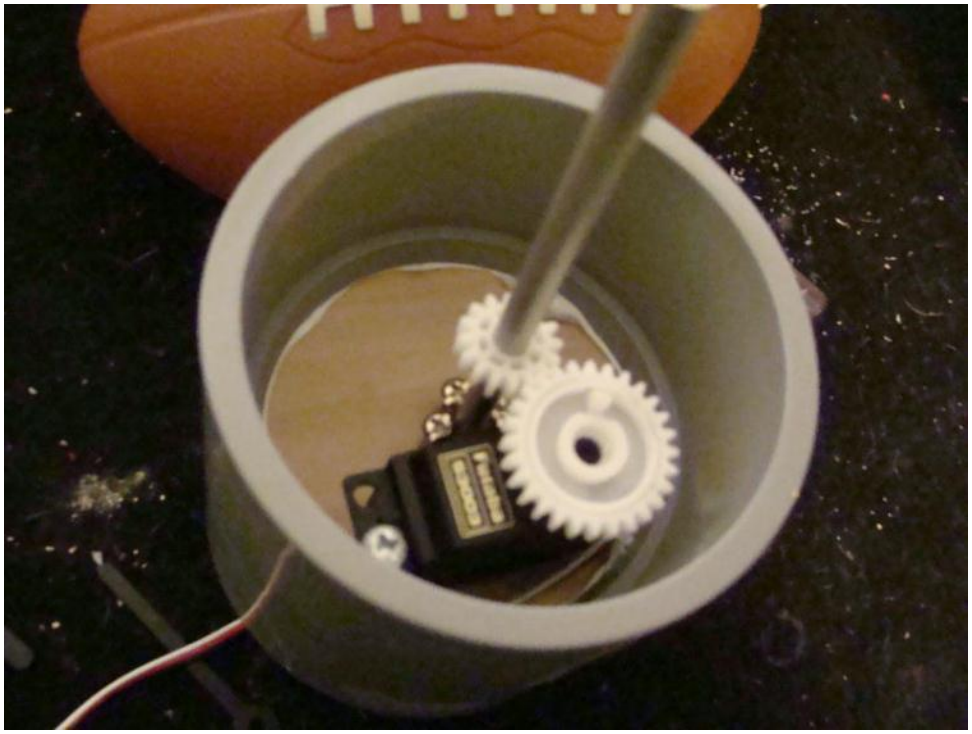


Figure 3-41. Flight instrument assembly glued and sealed to circular pipe. *Photo by Joseph Munera*.

Table 3-1. Futaba S3003 Specifications

Specification	Value
Speed	0.23 sec/60° @ 4.8V 0.19 sec/60° @ 6V
Torque	44 oz-in (3.2 kg-cm) @ 4.8V 57 oz-in (4.1 kg-cm) @ 6V
Dimensions	1.6" x 0.8 x 1.4" (40 x 20 x 36mm) w/o output shaft
Weight	1.3oz (37g)
Connector	"J" type with approx. 5" lead
Cost	$10.99

Figure 3-43 shows the flight instrument aircraft gauge finished product. A hacksaw was used to cut the aluminum shaft to the correct size. The faceplate consists of a printed airspeed indicator face glued to a circular cardboard cutout. This was then glues to the circular pipe. The needle comes from a build your own clock set kit that was purchased from Skycraft. It sits inside a hole in the center of the aluminum shaft. The gears have 28 teeth on the idler gear and 16 on the pinion. If we divide the idler gear teeth count by the pinion gear teeth count, we find the gear ratio for the setup. For this gauge the gear ratio was calculated to be 1.75:1. Additionally the gear was tested in the Senior Design lab where we were able to successfully turn the shaft with the servo motor. It appears however that the Futaba servo does not possess the right response curve in terms of the rotation angle, therefore causing problems with gauges that require extreme movements of the gears (such as an airspeed indicator or altitude indicator).

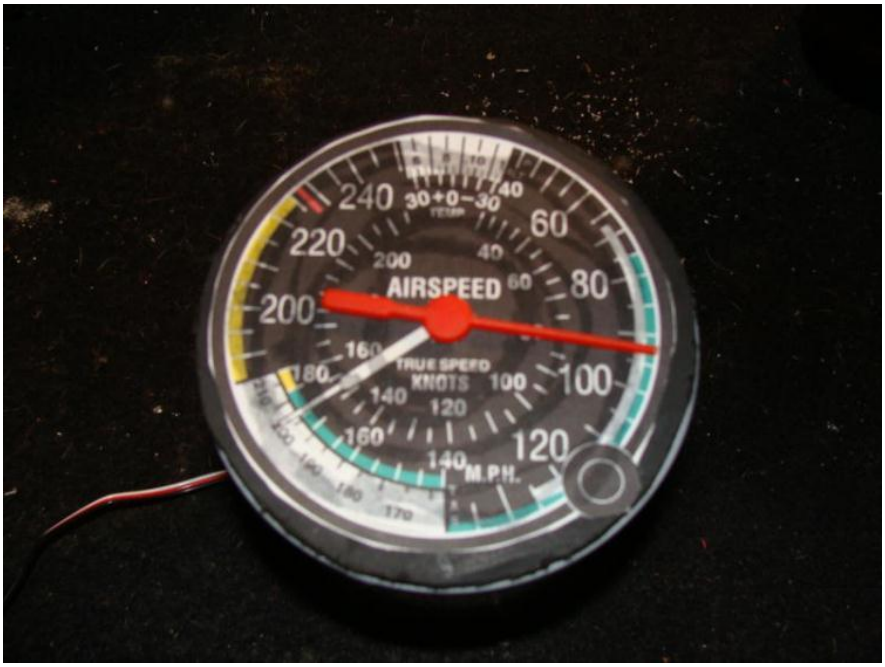


Figure 3-42. Assembled flight instrument with face plate and needle. *Photo by Joseph Munera*

Since our testing of a prototype using a Futuaba motor did not work as expected, we have ultimately decided to not use it. Because the issue lies not just with the Futaba motor, but with servo motors in general, we will utilize a stepper motor in the gauges instead. The stepper motor we intend to use the Mineba SMT-112 available at allelectronics.com. It features 48 steps per revolution with a movement of 7.5 degrees per step. Testing with this motor will take place, to confirm if this a suitable replacement for the Futaba servo motor.

The second prototype we built was designed around the multiple layered deck concept and was closer to what we actually built for our simulated flight instrumentation.

Figure 3-43 shows the prototype for the heading indicator which later became the basis for all of our simulated flight instruments. This design consists of the clear acrylic lens with the airplane markings. This lens is supported by #4-40 thread female to female hex spacers which connect to a piece of sheet aluminum for the motor deck. In between these two pieces is the compass rose which in turn is connected to the shaft of the stepper motor.

Figure 3-43. Prototype 2 *Photo by Joseph Munera*

3.4 Flight Control Design

In this section we will discuss the design of our flight control interfaces. Originally, the plan was to implement our designs into the cockpit that we were slated to receive. Due to not receiving the cockpit, we still needed to provide flight controls for the user, and a decision was made with our sponsor to build

test rigs for the controls that would be used for input for the demonstration in addition to validating our electrical designs. We originally were designing a joystick around the existing stick in the aircraft, rudder pedals utilizing the existing pedals found in the aircraft and a throttle, but since no aircraft arrived we needed to change our design implementation. Since we want preserve a record of our original designs so that they could be possibly implemented by our project sponsor (if the cockpit arrives at a later date), we will present these alongside temporary controls that we built from scratch.

3.4.1 Joystick Design

One of the most critical components of this project was the joystick controller that was 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 push-pull rods. For this simulator we needed to simulate this same operation digitally and then passes the data into the simulator computer so that the appropriate command is executed on the screen. In order to tackle this problem we needed 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.

Figure 3-44. Control Stick. *Photo by Robert Gysi*

The image shown in Figure 3-44 is the actual control stick in a GoBosh G700S that would have been part of the airframe that was to be utilized on this project. From discussions with our project sponsor, we determined that since the cockpit was not being received with the control devices, we would need to build a basic joystick for inputs and testing. We decided to come up with a very similar design from the one we first came up with using the two slide pots one for x-axis and one for y-axis. It uses a plunger for control and return to center.

During our research in Fall 2009, two methods were initially researched for implementation for our joystick. The first proposed design we brought to our sponsor had us utilizing a joystick controller that we could purchase from a number of electrical component distributors. An example of this option would be the SPC Multicomp STD-2607AR joystick controller available from Newark.com. This joystick controller has all of the mechanical and electrical connections and can simply be retrofitted to the end of a control stick. Mechanically it has the ability to rotate 60° in each direction with a minimum required operating force, but only has a rated lifetime of 300,000 cycles[21]. Additionally the size is relatively small, which at first seemed perfect since we did not know where we would find room to mount this device. However, it having a short stature (1.29" is the length of the joystick knob), was deemed to be impractical to use for this application, because we need to be able to have a wide range of deflection in our controls to mimic the actual feel of the stick in the aircraft (approximately 8" at the user end). Additionally, it is a fairly expensive part for a fairly simple task, costing $67 with a $20 handling fee as it must be shipped from the United Kingdom. Therefore after discussing options with our sponsor we decided to take a much cheaper and basic approach.

Our finalized approach included using 60mm slide potentiometers that we would attach to the push-pull rods and cables that exist inside of the fuselage. While we did not receive the cockpit, we were still able to use this design electrically with our test rig setup. The only difference between integrating with the cockpit controls and building our own is simply how the slide potentiometers are connected to the controls. To keep a record of our original design efforts, Figure 3-45 highlights the locations of the push-pull rods and cables that we would have interfaced our controls to.

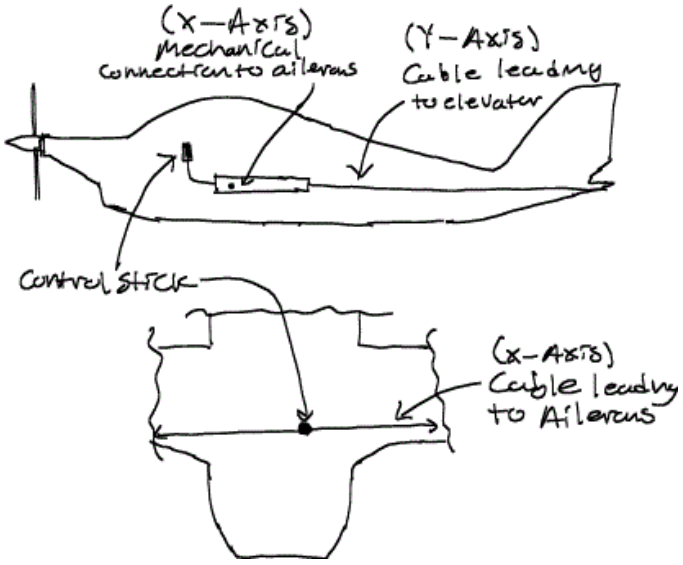


Figure 3-45. Mechanical linkages for control stick. *Drawing by Robert Gysi*

[21] (*2009, Nov.). SPC Multicomp STD-2607AR Product specifications and drawing [Online]. Available: http://www.farnell.com/cad/358999.pdf*

For our implemented design for our demonstration, we needed to build out of wood, every aspect of the joystick. To do this we started with a sheet of plywood which we cut slots for our slide potentiometers to be held in place with screws. Attached to this base was also stand which held our stick and created a pivot point for the stick (fashioned out of a used sink plunger). At the bottom of the stick we connected strips of metal to the each of the X and Y axis potentiometers.

In our initial testing of our joystick we found that there was no return to center capability like the stick in the aircraft would have. To correct this issue, we used the rubber end of the plunger to accomplish this. All we needed to do was simply cut a hole in the center and slide over our shaft. Additionally we made some slits to the rubber plunger base to free up the motion a little bit. This was done after testing showed the plunger by itself was possibly too stiff to correctly get our range of motion. Additionally since we wanted to make sure we had the correct throw of the stick, we measured the throw in our sponsor's GoBosh to a distance of 8". To implement this in our simulated stick, we simply measured out 8" of throw on a table edge and cut our stick to the length that would give us our distance. As a result the stick is a bit on the low side to the user, but still perfectly usable. Figure 3-46 shows our overall construction of the joystick test rig.

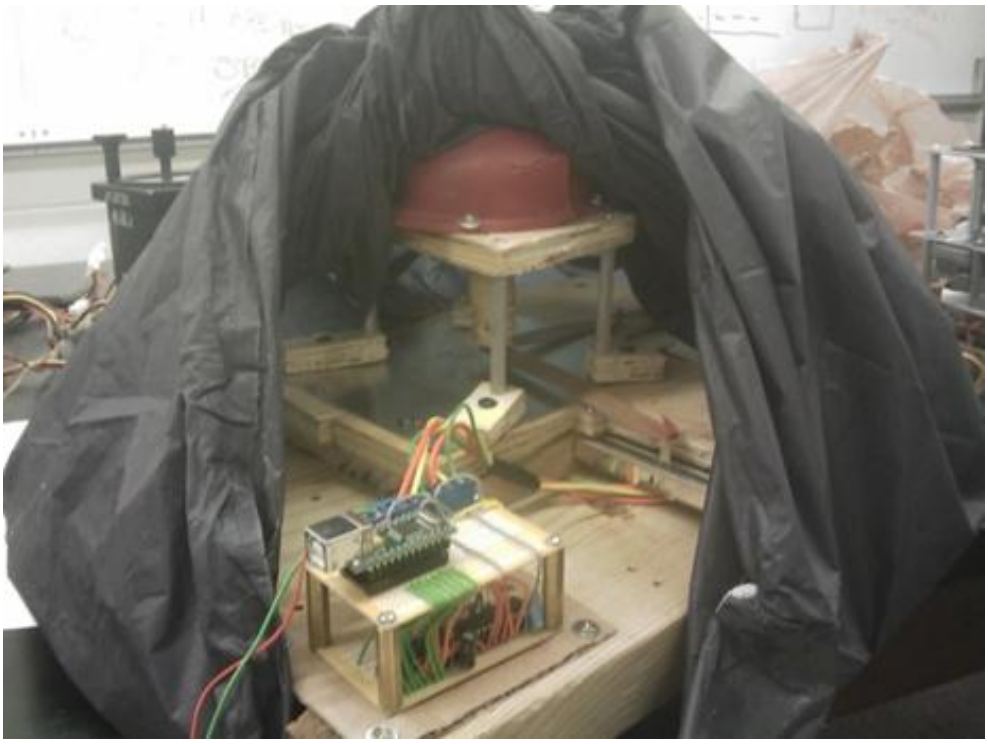


Figure 3-46. Joystick implementation

For this application we just needed your everyday basic slide potentiometer available from any number of suppliers.

3.4.1.2 Joystick Analog Design

Without going into how a potentiometer functions, the analog side of the design is quite straight forward. Utilizing a +5V supply from the computer power supply, we can then find what the corresponding voltage is across the separate X and Y axes. With these varying voltages, we can take the analog output of the joystick control and then feed the result into an analog to digital (A/D) converter. We used our RA6020F-10-20D1-B10K slide potentiometer to get our voltage divider circuit to register a voltage between 0 and 5 volts. This output voltage will then feed into the A/D converter which is then fed into the FTDI chip. Once it enters the A/D it is now a digital design problem.

3.4.1.3 Joystick Digital Design

As the control stick must ultimately be connected to the USB port of a computer, there is a digital component to the design of this device. For this purpose, there were two options to consider for this design. The first option reviewed included using an A/D converter IC chip tied to the inputs on the FTDI USB interface chip that is being utilized for our gauge design. This depending on the number of controls connected to the FTDI chip is fed into the USB chip based on the chip select pin on each of the A/D that are connected to the controls. This value is then read in and processed by the software and is fed into the X-Plane system.

Table 3-2 Implementation of Control Stick Analog to Digital

Function	Slider potentiometers	FTDI chip select
X Axis	Output (Pin 3)	0
Y Axis	Output (Pin 3)	1

The slide potentiometers that are positioned along each axis of movement are tied to the analog-to-digital (A/D) converter with the connections listed in Table 3-2 (above). The software handles the change in values that are needed to take the A/D value to the correct input value; this is handled in the Config.ini file.

3.4.2 Rudder Pedals

In this simulator we needed to include the ability to use rudder pedals as a basic flight control device and to enhance the simulation experience. Like with the stick, due to not receiving the cockpit meant that we would not have actual pedals to interface with. Additionally, we found that we would need to change every aspect of our design aside from the electrical components in order to implement this requirement.

For our original design we wanted to match the feel of the pedals in the aircraft as close as possible. This meant imply interfacing a rod to the back of one of the pedals in the footwell to a slide potentiometer behind the engine firewall. This

would allow have allowed us to keep the mechanical interfaces and linkages intact and maintain the “feel”. In addition to this, we should note that from the beginning, the requirement for toe brakes were not required nor planned to be implemented. We also did not implement this as an additional feature when we needed to build our own pedals. Figure 3-47 shows the pedals as they exist in the aircraft and what we needed to replicate.

Figure 3-47 Pedals on the GoBosh G700S. *Photo by Robert Gysi*

Using purchased lumber and scrap pieces from the senior design lab we set out to create our rudder pedal test rig so that we could validate our electrical design efforts and provide input during the demonstration phase of the project. The design was kept inherently simple just like with all of our other control interfaces. The biggest difference here between our other controls is that for our test rig we did not utilize a slide potentiometer, instead using a rotary potentiometer for measurement. We opted for this as we wanted to have a single pivot point so that the pedals moved together. To achieve this we just needed to build a simple frame with two pedals and a rod sticking out of the front middle. Given the short time frame that we had to develop a solution (approx. 3 weeks before the project deadline is when the final determination was made that cockpit would not arrive in time), we sought to keep everything simple in comparison to a design that would have utilized a slide pot in the middle. This would have required more moving parts would have more than likely increased the cost.

During our testing we found that our first revision had a design flaw in that the motion was not very smooth at all due to the wooden frame rubbing across the plywood base. To rectify this issue, we purchased a lazy susan from Home Depot that elevated our pedal frame off of the base and provided a smooth motion. This resulted in more favorable results although occasionally the gears on our potentiometer slip for an unknown reason. As a result, when first using the simulator, it is recommended to use the test application to find the midpoint of the potentiometer and then set the pedals so that they are aligned correctly. Figure 3-50 below shows a close up of the gears on the shaft of the rotary potentiometer.

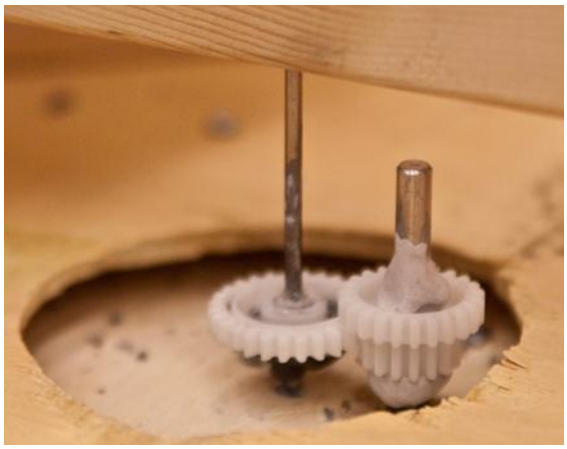


Figure 3-48. Mechanical configuration of the pedals. *Photo by Lewis Vail.*

In addition to the basic frame of the pedals we needed to place a location for the user to place their feet during use. We also wanted to ensure that we had somewhat of the feel of actually using the pedals in the aircraft. In order to achieve this we simply cut two pieces left over from building the frame of the pedals and attached each to the from using door hinges. This allowed the user to either have their feet at an angle up on the pedals (provided they did not push too hard as it possibly would have adverse effects on the gear shafts) or in a position with their heels on the ground and their feet on the pedals. Together with the stick the instruments were secured to another piece of plywood so that each device would retain proper distance apart in addition to keep the user from pushing the pedals away during use. Figure 3-49 below shows this configuration.

Figure 3-49. Pedals and Stick combined. *Photo by Lewis Vail*

3.4.2.1 Rudder Pedal Analog Design

As mentioned in the previous section we did not utilize our 60mm 10kΩ slide potentiometers as originally intended. Instead we utilized a 10kΩ linear rotary potentiometer for our data measurement. The decision was made from a purely mechanical standpoint and can be reviewed in the section preceding this one. Just as with the joystick control, we utilized the +5V supply from the computer ATX power supply to find what the corresponding voltage is across our potentiometer and then fed the resulting voltage into an A/D on the control circuit reserved for the throttle and pedals. We would read in a value between 0 and 5 volts just as with the joystick and once this voltage is passed through the A/D it becomes a digital design problem, which we cover in the next section.

3.4.2.2 Rudder Pedal Digital Design

As the pedal assembly must ultimately be connected to the USB port of a computer, there is a digital component to the design of this device. For this purpose, there were two options to consider for this design. The first option reviewed included using an A/D converter IC chip tied to the inputs on the FTDI USB interface chip that is being utilized for our gauge design. The second option was to use a microcontroller (specifically the Atmel AT90USB1287) which had onboard USB support and A/D pins. Ultimately the decision was made to utilize two FTDI chips between our controls with our throttle and pedals using one control circuit. For our throttle/pedals design we utilized the chip select pin on each of the A/D that is connected to our controls. This is to ensure that we select the right input device for taking in values into X-Plane. Table 3-3 below shows the chip select value for selecting the A/D assigned to the pedals.

Table 3-3 Implementation of Pedals Analog to Digital

Function	Potentiometer	FTDI chip select
Pedals	Output (Pin 3)	0

3.4.3 Throttle

The theme with the controls for this simulator is that in each section the design becomes simpler and simpler. For our throttle control all we need to implement is a simple slide potentiometer which should have a travel length as close to 100mm in order to get as much travel as possible. From an electrical standpoint it is the same as our other control inputs.

Mechanically, it is nowhere near the same as the others. This was also the only control device that we were not slated to receive from Aero as part of the cockpit (pedals and stick were to be in place). As a result we did not need to make any design changes to our throttle assembly. To implement our throttle we opted to use the same slide potentiometer as with our other control devices, meaning we used the Alpha (Taiwan) RA6020F-10-20D1-B10K. This potentiometer did meet the requirement of being 100mm of travel, but unfortunately due to costs of some

100mm slide potentiometers and the lower cost ones being not expected in before project completion, the decision was made to utilize this slide potentiometer. As a result we are about an inch short on the throw of the throttle, but this does not impede operation of the device. Figure 3-50 shows the location of the throttle inside of the cockpit of our sponsor's GoBosh G700S.

Figure 3-50. Throttle location (black knob pulled out of instrument panel). *Photo by Robert Gysi*

To begin with the mechanical design we will start off discussing how we need to modify our slide potentiometer to be able to interface with our throttle rod. The reason we picked this slide potentiometer outside of it meeting our electrical requirements was the fact that the slider on it would be able to be modified for use on any of our control devices. That that end, the dimensions from the slide are .197" at its widest and .787" total height. This gives us adequate space to drill a hole for a #4 screw into. Figure 3-51 shows an approximation of the tab and the modification made to it.

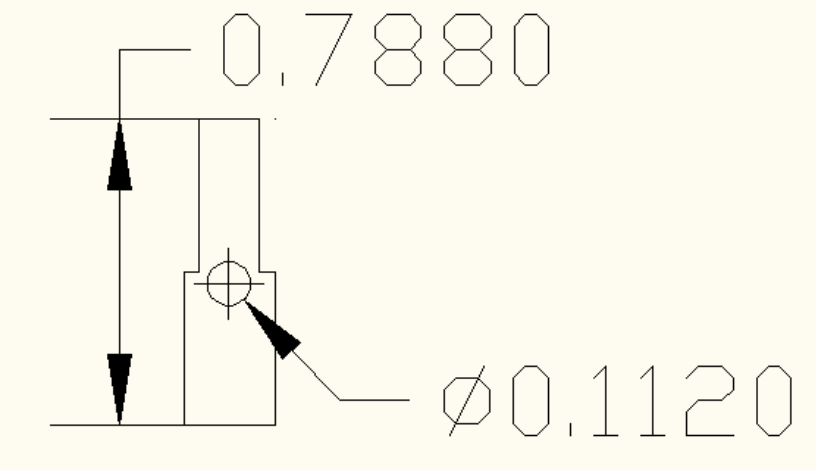


Figure 3-51. Drill Pattern for potentiometer modification. *Drawing by Robert Gysi*

After modifying the potentiometer tab, we can discuss how to connect the throttle rod. To achieve this we knew we wanted to secure a small piece of angled metal to the slider. Through searching through the scrap metal bins at SkyCraft, we discovered several small pieces that were strong, lightweight, cheap, and small enough to fit in the space required. It was important to use a piece that was angled so that we could have a flat surface against the potentiometer and forward facing flat surface for the throttle rod to mount to. To build our throttle rod we started with a 6-32 threaded rod that was also purchased from skycraft. This gave us a satisfactory length with the slide potentiometer we used (would have needed a longer rod if we had been able to secure a 100mm potentiometer). Since the rod itself is of a small diameter, we wanted to increase the shaft diameter by using the same aluminum tube we purchased originally for our gauge construction. This was cut and filed down until we had just enough of the 4" threaded rod showing to secure our knob to (just a standard wooden cabinet knob purchased from Home Depot). The AutoCAD 2D drawing in Figure 3-52 represents the design of our throttle.

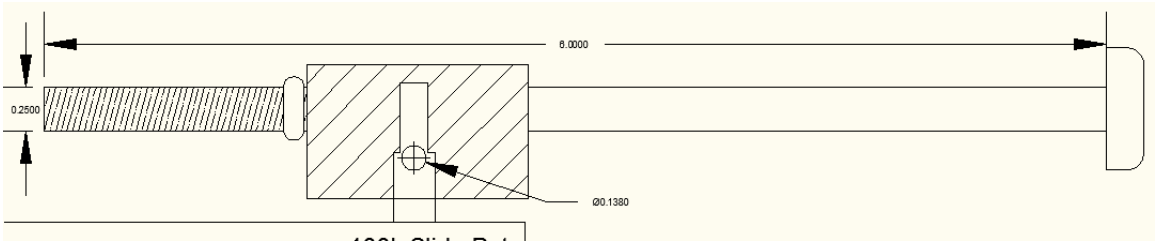


Figure 3-52. Mechanical representation of Throttle Assembly. *Drawing by Robert Gysi*

Additionally, we needed to be able to lift the assembly located behind the panel so as to secure it in place and keep the throttle from moving around. To achieve this we used more angle brackets purchased from SkyCraft (although with only a single bend) to form the legs of our throttle stand. After drilling a hole large enough for a #6 screw (diameter of approx. 0.13"), we used another 4" rod and secured on each end with two nuts on both sides of each bracket. The inner nut is your standard 6-32 nut, while the one on the outside is a nylon insert lock nut. This helps us keep our rod from rotating once in place. It is also important to note that since we wanted to run the rod across the bottom of the slide potentiometer between the connection pins, we needed to ensure that we kept the rod from shorting anything. To ensure this, we wrapped the rod in electrical tape and then inserted the slide potentiometer over top. The tape itself held the potentiometer fairly snug, but for extra security we utilized zip ties on both the front and back of the throttle (so as to not impede movement).

Figure 3-53. Angle bracket for connecting to slide potentiometer. *Photo by Robert Gysi*

Through our testing we discovered issues with the potentiometer wanting to lean towards the right when a user pulls out on the throttle. To alleviate this we added an extra angle bracket supporting it from being able to rotate about the rod. Also, we found that the throttle assembly had issues of wanting to travel left and right within the hole. Since we were not able to get the correct nut that the throttle would have normally exited the panel through, we devised a solution using the thin aluminum that we used for several purposes in our gauge construction. We cut a thin strip that was secured to the base of the front throttle assembly. This helped restricted our throttle from moving left and right. Although this was our solution, we would recommend that the appropriate sized nut be located and installed to maximize the left-right travel restriction. The completed throttle assembly installed in the instrument panel is located in Figure 3-54 below.

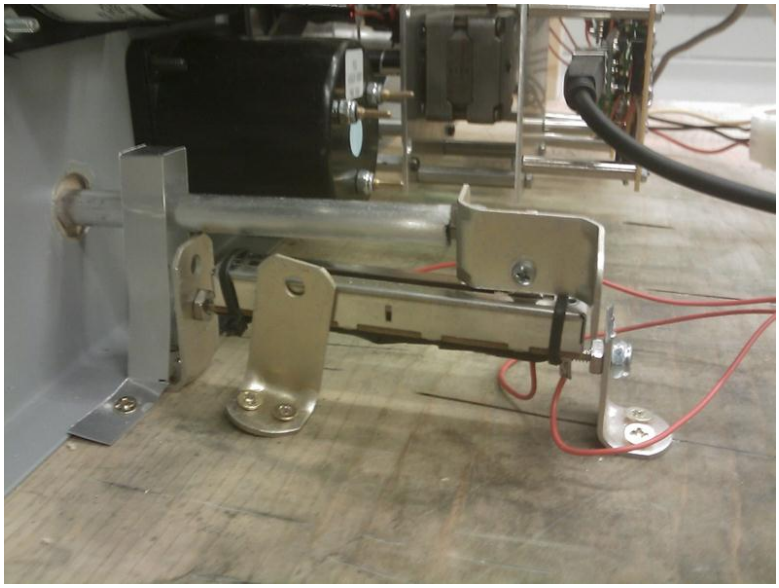


Figure 3-54. Completed throttle assembly. *Photo by Robert Gysi*

3.4.3.1 Throttle Analog Design

The analog electronics design on the throttle is the same as with all the other control circuits. Using the 10kΩ slide potentiometer we used an input voltage of +5V from the ATX power supply and connected in such a fashion that Pin 1 was input, pin 2 was the wiper (also tied to input) and pin 3 was our output pin, so that our voltage divider circuit would register a voltage between 0 and 5 volts. This output voltage will then feed into the A/D converter which is then fed into the FTDI chip. Once it enters the A/D it is now a digital design problem.

3.4.2.2 Throttle Digital Design

As the throttle must ultimately be connected to the USB port of a computer, there is a digital component to the design of this device. For this purpose, there were two options to consider for this design. The first option reviewed included using an A/D converter IC chip tied to the inputs on the FTDI USB interface chip that is being utilized for our gauge design. The second option was to use a microcontroller (specifically the Atmel AT90USB1287) which had onboard USB support and A/D pins. Ultimately the decision was made to utilize two FTDI chips between our controls with our throttle and pedals using one control circuit. For our throttle/pedals design we utilized the chip select pin on each of the A/D that is connected to our controls. This is to ensure that we select the right input device for taking in values into X-Plane.

Table 3-4 Implementation of Throttle Analog to Digital

Function	Potentiometer	FTDI chip select
Throttle	Output (Pin 3)	1

3.4.5 Combined Flight Control Circuit

The schematic below in figure 3-55 represents the overall circuitry for the implementation of the flight controls utilizing the pins on A/D ports mentioned in the previous sections tied to a potentiometer.

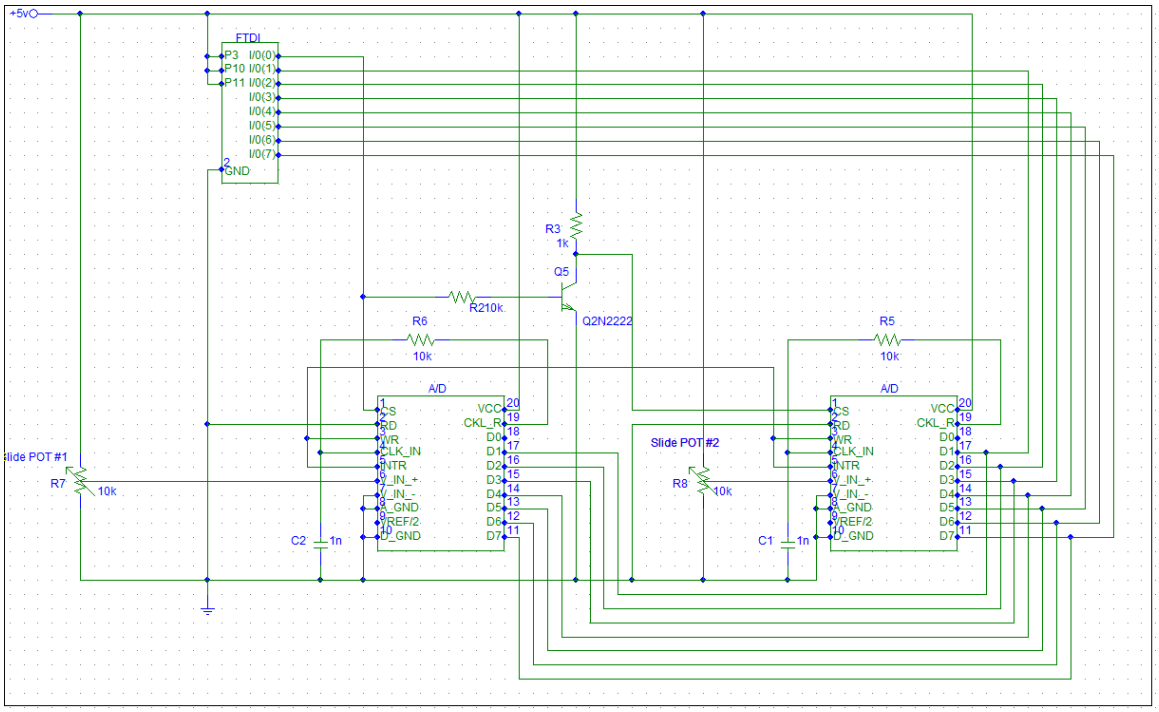


Figure 3-55. Circuit implementing control systems. *Created by Chris Dlugolinksi*

3.5 Computer Hardware Selection

Computer selection is a probably the most important task on this project due to the fact that a full computer needs to be assembled so that we are able to meet our requirements that were discussed earlier in this documentation to provide the most realistic and smooth experience while maintaining a low cost. This is not always easy to do and in section 3.5.1 a discussion on the choice of various components along with trade-offs are presented.

3.5.1 Computer Hardware

Flight simulators are notorious for being some of the most graphics intensive applications that a consumer can install on their personal computer. This is even truer with X-Plane, because not only does it feature impressive graphics, but also due to the fact that X-Plane is also a full-fledge aerospace modeling application. Because of this we needed to ensure that we had hardware powerful enough to support the application. In this section we will cover both the original computer selection that was to power the simulator before the decision to not display at Sun 'n Fun was made (Section 3.5.1.1) and the computer specifications for the system we used for our demonstrations (Section 3.5.1.2). This is presented in this fashion to preserve our original efforts for future completion by our sponsor.

3.5.1.1 Original Design Effort

Starting with our original computer design effort the most appropriate place to start our discussion is on CPU selection. When it comes to manufacturers, there are only two (Intel and AMD), and along with that there is a fierce debate

between enthusiasts over which one provides superior performance. Ignoring the recommendations of those individuals we set out to find the lowest-cost, highest performing CPU available from each manufacturer. For comparison, the Intel family selected would be the Core 2 Duo whereas the AMD family selected for comparison would be the AMD Phenom II family. Each of these processor families are dual core, x86-64 processors and are priced relatively the same. Upon searching various computer part distributors online we settled on two processors the Intel Core 2 Duo E8400 and the AMD Phenom X2 550. A comparison the specifications are listed below in table 3-5.

Table 3-5. Comparison of Intel and AMD CPU options.

	Intel Core 2 Duo E8400[22]	**AMD Phenom X2 550**[23]
Clock Speed	3.0 GHz	3.1 Ghz
FSB (Intel) / HT (AMD)	1333 MHz	4000 Mhz
Socket Type	LGA 775	AM3
L2 Cache	6 MB	2x 512kB (1024kB total)
L3 Cache	N/A	6 MB
64-bit	Yes	Yes
Manufacturing Process	45 nm	45 nm
Voltage	0.85-1.3625V	0.85-1.425V
Heatsink Included	Yes	Yes
Price	$167.99	$99.00

Here we have two very similar CPUs that are matched in almost every specification, but one of them, the $99 AMD Phenom X2 550 is around $70 cheaper than the most equivalent Intel-manufactured CPU. Since we are attempting at all costs to create the most powerful machine for the lowest cost, it makes perfect sense to choose the AMD Phenom CPU over the Intel. Additionally, while this was not a factor in determining the CPU, it appears AMD has given the enthusiast community a gift with the release of this particular model. It turns out that the Callisto-based Phenom X2 processors are really quad core chips with just two of the cores disabled and are extremely receptive to overclocking to upwards of 4 GHz. This means that for $99 we could upgrade our CPU using a fairly simple process to unlock the remaining two cores (requires no hardware modification) and up our clock speed to a high value in the end giving us the performance of a nearly $200 AMD Phenom X4 or an Intel Core 2 Quad[24].

[22] *(2009, Nov.). Intel Core 2 Duo E8400 Specifications [Online].* Available: *http://www.newegg.com/Product/Product.aspx?Item=N82E16819115037*

[23] *(2009, Nov.). AMD Phenom X2 550 Black Edition Specifications [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16819103680*

[24] (*2009, Dec.). AMD Phenom X2 550 Review – Unlocking Blocked Cores [Online].* Available: *http://www.xbitlabs.com/articles/cpu/display/phenom-athlon-ii-x2_15.html*

The next most important piece of computer hardware to be installed in the simulator computer is the graphics card. Just like in the CPU industry there are two primary manufacturers of chipsets: ATI and NVIDIA. In order to stay within our project budget we would have to neglect the most recent, high-end cards from these manufacturers. This unfortunately means that we would not be able to purchase a card that has ATI's new Eyefinity™ technology. This technology allows for a maximum resolution of 8192x8192, but at the same time also requires the use of a monitor that includes a display port. There is also a limitation currently where if one wanted to use three monitors utilizing the DVI connections, only two would be able to be utilized even with two cards running due to a technical limitation[25]. This technical limitation could possibly corrected by the completion of the project, however it would still be possible to run the card in CrossFireX mode without using the Eyefinity support to span the three displays. Currently large monitors, such as those in the neighborhood of 24" are still quite expensive, so we would need to utilize two ATI based cards. If the display limitation is addressed or the cost of display-port equipped monitors decreases in cost, then this may be a suitable graphics solution. Now, in order to select the graphics card to be utilized in our simulator PC, we have chosen two similarly priced graphics cards. One is based on an NVIDIA chipset while one will be based on an ATI chipset. They both should have fairly comparable specifications and performance given the rivalry between the two companies. The comparison between the ATI and NVIDIA based chipsets are in table 3-6 located below.

Table 3-6. Comparison of Graphics Card Options

	EVGA 01G-P3-N981-TR[26]	**XFX HD-575X-ZNFC[27]**
GPU Family	NVIDIA GeForce 9800 GT	ATI Radeon HD 5750
Core Clock	600 MHz	700 MHz
Shader Clock	1500 MHz	Unknown
Stream Processors	112	720
Memory Clock	1800 MHz	1150 MHz
Memory Size	1 GB	1 GB
Memory Interface / Type	256-bit GDDR3	128-bit GDDR5
SLI / CrossFire Support?	Yes (SLI)	Yes (CrossFireX)
Price	$139.99	$139.99

[25] (*2009, Dec.). ATI Eyefinity Technology Brief [Online]. Available: http://www.amd.com/us/Documents/ATI_Eyefinity_Technology_Brief.pdf*
[26] (*2009, Dec.). EVGA Product Specification Sheet [Online]. Available: http://www.evga.com/products/pdf/01G-P3-N981.pdf*
[27] *(2009, Dec.). XFX 5750 Specifications [Online]. Available: http://www.xfxforce.com/en-us/products/graphiccards/hd%205000series/5750.aspx#2*

These two cards are exactly the same price at Newegg.com and while they have some similarities, such as memory size, relatively close GPU clock speeds, and the ability to be linked to another graphics card to increase graphics processing power, they are very much different cards. The ATI-based card for example has a lowe memory clock rate than the 9800, but many more stream processors. Now there may be a difference in how ATI versus NVIDIA calculates the stream processors on the chip, but there is not a way to know without diving into what is more than likely ATI-proprietary information.

Moving on, let's discuss some of added features each card brings to the table. The NVIDIA based card is a Direct X 10/OpenGL 3.0 based card that includes support for NVIDIA PhysX (enhanced physics processing), and support for NVIDIA CUDA, which is a "general purpose parallel computing architecture that leverages the parallel compute engine in NVIDIA graphics processing units to solve many complex computation problems in a fraction of the time required on a CPU."[28] These two features however will be of no assistance to us in running the flight simulator, for one X-Plane is not optimized for the PhysX architecture and we will not be writing any CUDA based applications.

Likewise, the ATI Radeon HD 5750 also comes with "value-added" features as well. However, it is important to first note that this card is not only DirectX 11, but is also an OpenGL 3.1 card, meaning that it is compliant with the latest revision of each graphics rendering architecture and being the most up-to-date card between the two. Additional the GPU on the 5750 includes support for the previously mentioned ATI Eyefinity technology and a technology known as ATI Stream.[11] The Eyefinity technology was described on the previous page, and ATI describes the Steam technology as "enable AMD graphics processors (GPUs), working in concert with the system's central processors (CPUs), to accelerate enabled applications beyond traditional graphics and video processing."[29]

With all of the information about each of the graphics processing units taken into account, we must come to a conclusion about which to pick for inclusion in the simulation computer. The ATI Radeon HD 5750 from XFX is the one that has been selected. It has a slight edge over the NVIDIA-based card in the raw specifications, but the real selling point has been the inclusion of the Eyefinity technology. While the Eyefinity technology may have some limitations currently, this project will ultimately have a much longer life beyond the end of the Spring Semester and it is important that we design the computer powering the simulator to be ready for future technologies and future capabilities.

The selection of the motherboard is something that either happens first and then you build your computer around it or you go in the opposite direction and find the

---

[28] *(2009, Dec.). What is CUDA? [Online]. http://www.nvidia.com/object/cuda_what_is.html*

[29] (*2009, Dec.). ATI Stream Technology [Online]. Available: http://www.amd.com/us/products/technologies/stream-technology/Pages/stream-technology.aspx*

components you wish to use and find a board that will meet your specifications. Performing a search on newegg.com returned several boards, but two stood out from the rest. The boards, one manufactured by ASUS and the other by MSI are both fairly comparable boards with nearly the same specifications at the same price. However, there is one major difference. The ASUS board allows for 16GB of DDR3 RAM to be installed while the MSI board only allows for 8GB of DDR3 RAM A comparison of the two follows in Table 3-7.

Table 3-7 Motherboard Comparison

	ASUS M4A785TD-V EVO[30]	**MSI 790X-G45**[31]
Socket	AM3	AM3
Chipset	AMD 785G/SB710	AMD 790X/SB710
Memory	4x DDR3 DIMM Max. 16GB	4x DDR3 DIMM 8GB
Expansion Slots	2x PCIe x16, 1x PCIe x1, 3xPCI	2x PCIe x16, 2x PCIe x1, 2xPCI
CrossFireX Support	Yes	Yes
Onboard Audio	Yes	Yes
USB Ports	12	6
Form Factor	ATX	ATX
Phenom X2 Unlock	Possible	Not Possible
Price	$99.99	$99.99

With the CPU, graphics cards and motherboard selected, we can select our remaining components to round out our computer build. For RAM, we have decided to go above the minimum requirements for X-Plane (set at 1GB) and Windows 7 minimum (also 1GB) and go with a 4GB DDR3 dual channel kit running at the DDR3 1066 speed (PC3 8500). This should be sufficient for the simulator, although if more memory is desired, the motherboard will allow up to 16GB total to be installed (64-Bit Windows 7 is required for this).

For drive selection it was incredibly straight forward. For the hard drive we calculated that total space required by an Installation of Windows 7 Professional and X-Plane 9.4 would utilize roughly 100GB of capacity. Since we do not need a very large drive due to the computer's specialization, a 160GB Serial-ATA drive with a 8MB cache and a 4.2ms average latency from Western Digital was selected.[32] In reality when it comes time to purchase any drive as long as it meets or exceeds the same specifications could be purchased. This will allow us to procure the cheapest drive and potentially cut our spending some. In addition, the same situation exists for the DVD-ROM drive. Since almost all DVD-ROM

[30](*2009, Dec.). ASUSTeK Computer Inc. M4A785TD-V EVO Specifications [Online]. Available:* http://www.asus.com/product.aspx?P_ID=fcsXWSxnhzZE9rnR

[31](*2009, Dec.). NewEgg: MSI 790X-G45 Specifications [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16813130249*

[32] *(2009, Dec.). Western Digital WD1600AAJS Hard Drive Specifications [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16822136075*

drives are essentially the same (they all read DVD and CDs) and almost all have a read speed of around 18x, we are again able to go with the cheapest possible drive available to us. The Lite-On iHDP118-08 meets this requirement and only costs under $20.

All of the components will be fitted into an case that meets the ATX specification. We have chosen the Linkworld 313-06-C2228 available from Newegg.com for $20.99. It is a very simple case that can hold our ATX motherboard, includes 3 mounting locations for fans and provides enough space for all of our drives. Also, since the case manufacturer is not a critical requirement (only that we have a case for the computer), this could change when it comes time to purchase components in the spring. Table 3-8 lists the complete specifications of our desired computer configuration.

Table 3-8. Complete Simulation Computer Specifications

	Part Number	**Description**
CPU	HDZ550WFGIBOX	AMD Phenom X2 550 @ 3.1 GHz
Graphics Card	HD-575X-ZNFC	XFX ATI Radeon HD 5750
Motherboard	M4A785TD-V EVO	ASUS AM3 ATX Motherboard
RAM	OCZ3G10664GK	OCZ 4GB (2x2GB) DDR3 1066 Kit
Hard Drive	WD1600AAJS	Western Digital 160GB
DVD Drive	iHDP118-04	Lite-On 18X DVD-ROM Drive - OEM
Case	313-06-C2228	Black ATX Tower
Power Supply	EP-1000SC	ePower 1000W SLI Ready ATX PSU

3.5.1.2 Demonstration Hardware

While the preceding section discussed our original hardware design under the circumstances of receiving a cockpit to integrate all of our systems with this section deals with the reality of not receiving our cockpit and what we did to ensure that we still had the abilities to run X-Plane sufficiently.

From discussions with our sponsor, it was decided that for the demonstrations of our project at the end of EEL 4915 that we should use our development machine to power our graphics as well as all of our controls and instruments. Since the bulk of development took place using Chris' laptop we set out to test his machine to ensure that it would be able to handle all of the demands of X-Plane. We should note here that also due to not going to Sun 'n Fun meant that we would not be purchasing monitors and would not need to implement 3 monitors tied to the VGA output on the laptop. Instead we made the decision to just output to one 24" monitor owned by a group monitor and having X-Plane output a 120-degree field of view onto the single monitor.

From our testing on the laptop, we noticed no issues with graphics performance and as a result decided to use this computer for our demonstration. One limiting factor that did arise from the use of the laptop however was the number of USB ports. With only four available ports on the computer we knew that we would need a USB hub to handle all of our controls and instruments. We started out by first utilizing a single powered USB hub that had 7 available ports onboard. Through testing our USB hub during our integrated systems testing, we found that often when a seventh device was connected to the hub, we would experience issues. USB should be able to handle 128 devices, so the cause of this issue is unknown. As a work around we use a separate four port unpowered USB hub connected to a separate USB port on the computer. This alleviated our issue and allowed for all devices to work flawlessly. A full table below lists the specifications of the machine we used during our demonstrations.

Table 3-9. Complete Simulation Computer Specifications

	Description
CPU	Intel Core 2 Duo 2.4 GHz
Graphics Card	NVIDIA GeForce 280
RAM	6 GB DDR
Hard Drive	500 GB
OS	Microsoft® Windows® 7

3.5.2 Display Projection

Just as with the computer hardware selection, we will cover both our original design centered on receiving the cockpit in addition to covering what we did when the decision was made to not purchase a computer. Section 3.5.2.1 contains our original design work, so that it can be implemented by our sponsor at a later date. Section 3.5.2.2 covers what we did to have a working component for our demonstrations.

3.5.2.1 Original Design Effort

Display projection really comes down to two options. The first option was to utilize a DLP projector for producing our visuals onto a drop screen in front of the simulator. Unfortunately this presents several issues. To meet the requirement of a 120-degree field of view given to us by our sponsor, we would need to use multiple projectors, but each time the simulator would be set up the placing of projectors would need to be calibrated and the whole set up would be very cumbersome. Another concern was the effectiveness of projectors at an outdoor event. We would probably have to implement some kind of tenting to keep the light level within the projectors' operational range. This of course would also hide our simulator from plane sight hindering our sponsor from luring in on-lookers at various shows. The ultimate factor that led us away from this option was the cost. If we had chosen this form of display our entire budget would be spent on just the projectors.

Our second option is the one we recommend for implementation at a future date by our project sponsor: LCD Monitors. First of all the cost of LCD monitors has been driven down enormously over the last few years; a 24" monitor can now be purchased for under $200. In addition the setup is incredibly easy in that you just need to place each monitor next to the other. Also, LCD monitors are much easier to see than projectors when operating in very bright environments. The primary operation area of this simulator will be at air shows and other aviation gatherings so this was a major concern.

In designing our display system we needed to know what size monitors we need to purchase in order to produce a field of view of 120°. To achieve this we have established a set of formulas using basic trigonometry to calculate the required monitor size for any given viewing distance. Figure 3.56 shows the monitor configuration that meets our 120-degree field of view requirement. The dotted line in the middle represents the distance used in equation 3-1. *Monitor size* in this equation refers to the diagonal of the screen. This is because that is the dimension manufactures use to market their product. *K* in this set of equations represents the proportionality constant used to calculate a monitor's diagonal from their width. It is included in the formula set to illustrate the path that was taken to get our monitor size to distance proportionality constant (this constant comes out to be about 0.8352).

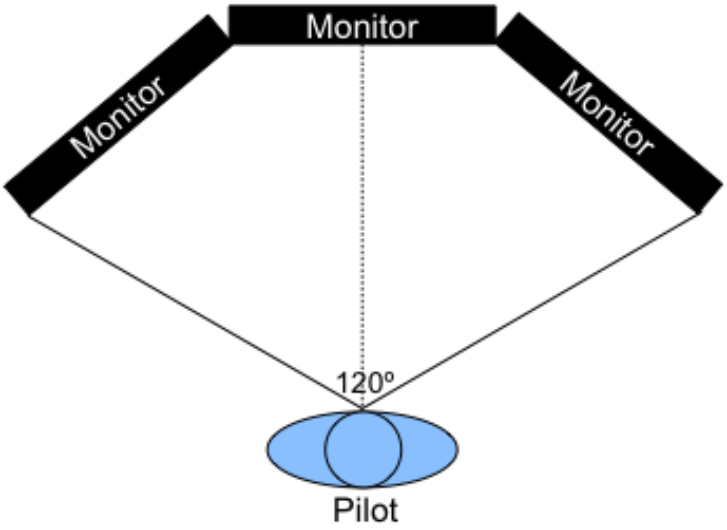


Figure 3.56: Schematic of display configuration. *Diagram by Lewis Vail.*

$$K = \frac{\sqrt{9^2 + 16^2}}{16} \quad (Eq.1)$$

$$constant = 2K \tan 20° = \frac{\tan 20° \sqrt{9^2 + 16^2}}{8} \quad (Eq.2)$$

$$monitor\ size = (distance)(constant) \quad (Eq.3)$$

Equation 3-1. Monitor Distance Formulas

From these three equations we can solve for our constant value and then multiply that by the distance the individual in the cockpit will sit from the monitors. For this we know that the distance should be about two and a half feet or 30 inches. Plugging this value into Equation 3 above gives us a value of 25". Now this is the complete diagonal of the monitor we have found, including the frame. In order to find the monitor size that we need to purchase, we need to simply subtract approximately 1" total (the frame of the many LCD monitors is around 0.5" wide) to come to the conclusion that we need three 24" monitors to give us our 120° field of view.

With our monitor sized now defined, we set about to located an adequate monitor for our needs. We ultimately found the Gateway FHD2401 on sale at Newegg.com for only $189.99. The monitor has a native resolution at 1920x1200 with a maximum viewing angle of 160° (Horizontal and Vertical) in addition a 5ms response time and a 2000:1 contrast ratio; all common traits of lower cost LCD monitors today[18]. We also expect that monitor prices will continue to drop as they have for the past few years, and should a better deal come along, say potentially a refurbished monitor from a major manufacturer that meets or exceeds the specifications on the FHD2401, we will consider purchasing that instead in order to decrease the overall cost of our project.

3.5.2.2 Demonstration Hardware

For our demonstration we still needed to provide a visual projection to the end user separate from our test machine. The reason for this is that we wanted to not utilize the screen built-in to the laptop as it was only 17" and we would have not been able to place it in an acceptable location without either blocking instruments or at a bad angle resulting in a sub-par performance. To remedy this we utilized a single 24" monitor (Asus VW246H) owned by a group member. This was placed in the center on top of our instrument panel frame and angled slightly downwards towards the user. This allowed us to give the reviewers some resemblance to how if we had three of these monitors the space footprint that would have been required. Additionally, we selected 120-degree as our output range in X-Plane which was in turn displayed on our single 24" monitor. Figure 3-57 below shows the configuration used for our demonstration.

Figure 3-57. Monitor placement for demonstration. *Photo by Robert Gysi*

3.6 Switches

The topic of implementing switches will be briefly covered, as this was not a specified requirement of this project and was not implemented. However, as we did complete the design work for this we have maintained the section on how to implement this for future reference for our sponsor, should the decision be made to at a later date. In the GoBosh there is a row of switches that light up when on and that allow you to control different functions on the aircraft. This includes switches to control the various lights on the aircraft in addition to the ability to start the aircraft. Pictured below (Figure 3-58) is the panel of switches that could be implemented using this electrical design. The numbered circles in the left of the picture are not switches, but are fuses and circuit breakers. This design discussion does not cover implementing these.

Figure 3-58. Switches and knobs that control the aircraft. *Photo by Robert Gysi*

To integrate these switches we could have used some sort of multiplexer or decoder to decide which of the switches is in what position. Each of the channels could have been read by the FTDI chip which only has the 8 I/O ports which is polled by the software. Pulling information from the simulator is relatively straight forward; a simple circuit diagram showing how the switches could be connected is shown below. This would allow any other developer to put this into the code and implement this.

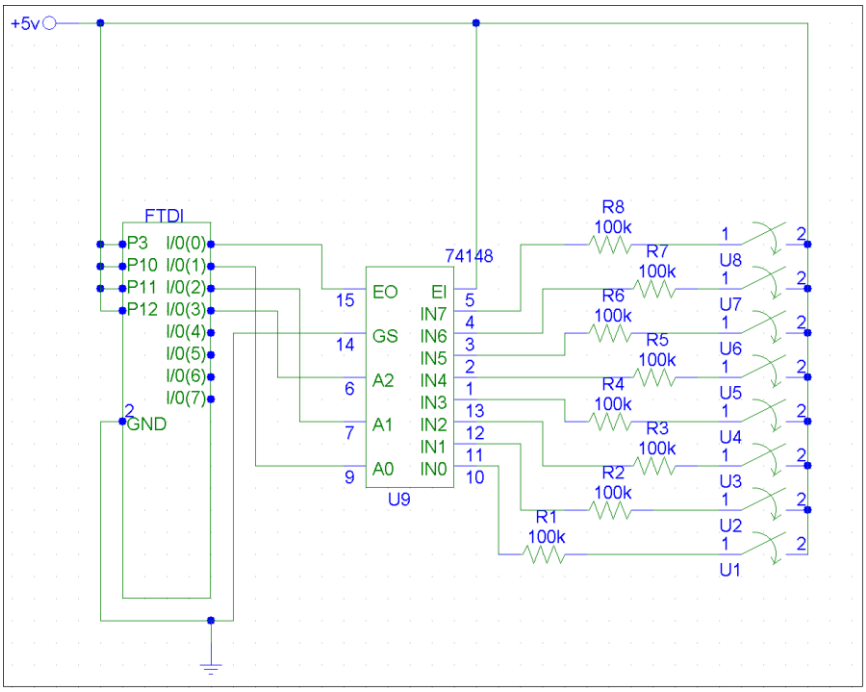


Figure 3-59. Switch circuit. *Created by Chris Dlugolinksi*

In this circuit in Figure 3-59 you can see that the FTDI chip would be connected to the Decoder, and a couple of the pins are used to select the channel that is allowed to come across. It is an 8-1 decoder; it will take 3 pins for selection and then one pin to read the switch. This can be done for 2 separate decoders allowing input for 16 different switches, and since switches are not really time needy they don't need to be polled very quickly and this circuit should work fine.

Each of the switches would need to be connected to a high point or the USB 5 volts or the power supply whichever we chose to go with. There should be a resistor in the circuit so that we don't draw too much current. The software on the computer side should recognize the chip as the control chip and then be setup to poll 2 switches at once using the same code sent out to each of the decoders. This info will then be read in over the USB and applied in the simulation.

3.7 Panel Indicator Lights

Similar to the switches we didn't have time to implement this additional feature. Although, we did receive lights from GoBosh in case we were able to find time to

implement this feature, we were not able to implement them. However, we did populate the panel with the lights we were given and the electrical design in this section is presented here for future use by our project sponsor. Just as with all the other devices controlled by the simulator, the indicator lights will be controlled once again through USB and using the FTDI chip we could use simple transistors and resistors to turn on and off the lights whenever needed. The nice part about using the FTDI chip is that once the port is high or low on the chip it will stay that way acting just like a switch. This allows the indicator light to stay lit or not lit whenever we need it to be. A circuit of the lights is shown on the next page as well as the actual indicator lights to be implemented along with their labels in the actual aircraft below. The data to drive these lights should be pulled from variables in X-Plane.

Figure 3-60. *Cockpit Indicator lights. Photo by Robert Gysi*

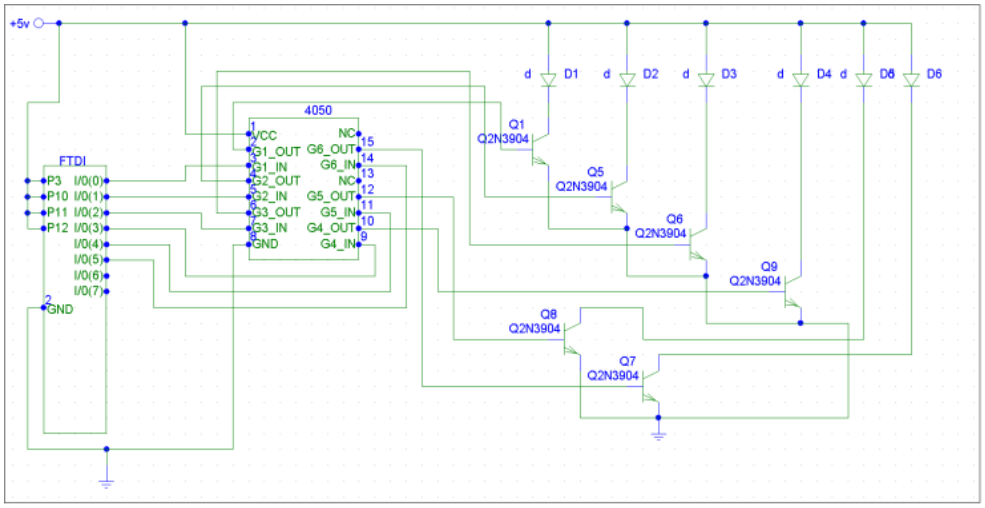


Figure 3-61. Indicator lights circuit diagram. *Created by Chris Dlugolinksi*

The schematic in Figure 3-61 (on the previous page) shows how the design for illuminating the lights in the cockpit could of been implemented. The other way that can be used depends on the sinking capabilities of the buffer. If it can light up all the lights and still function it may be possible to remove the transistors and just use resistors.

3.8 Flight Instrument and Control Interface Design

The core of our simulator will be X-Plane's simulation software. Therefore, all of our softwareinterfaces with X-Plane via its plug-in API. This API gives us access to all of the functions and variables that we needed.

The implementation of the plugin has three main parts. (1) The TimeProcessing part, this is the part of the plugin that has the main parts of the plugin callbacks needed in X-Plane. (2) The devices have two separate threads, one for the Controls, this updates the controls for the aircraft. (3) Then the final part of the plugin is the control of the Gauges. Each of these will be discussed in detail later in this section.
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.

Figure 3.62 shows a block diagram of our high-level plugin architecture. Each device will plug into the computer with its own designated USB cord and will be controlled by the 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. Figures 3.62 and 3.63 show a higher resolution diagram of the control and flight instrument interfaces respectively. Because all of the hardware interfaces into the simulation fall into one of these two categories, much of the code is reused for each gauge.

Figure 3.63 shows interface architecture for the control devices. 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.[33] The main loop for our control devices will be as follows. Setup each of the Devices connected to the computer. Find out how many devices are attached and initialize them using the .ini file that is used to set parameters for the controls or gauges. The two threads that need to be started by the plugin are then kicked off.

[33] (*2009, Dec.). X-Plane SDK [Online]. Available: http://www.xsquawkbox.net/xpsdk/mediawiki/Overview*

The controls thread is used to allow us to eliminate any waiting period to update X-Plane when the controls are being monitored. Most of our controls will give us a 8-bit digital signal, which will need to be truncated to an 7-bit signal so that we can use it in software. This is only for controls that have multiple controls linked to the FTDI board. 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. There will be a variable sized array in the plugin that is filled with the data for the controls that are connected so that we can just read these values in the plugin for use in X-Plane. Finally, once the appropriate data reference is populated with the new value, X-Plane will respond appropriately.

Figure 3.64 shows the interface architecture for the flight instruments. It is very similar to the control interface architecture except it is backwards. The first step in the main loop once again is to look up the appropriate data reference, this is found in the .ini file. The gauge thread is started and in this thread you have the data that needs to be set getting passed in from the array that carries all the connected devices. 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.

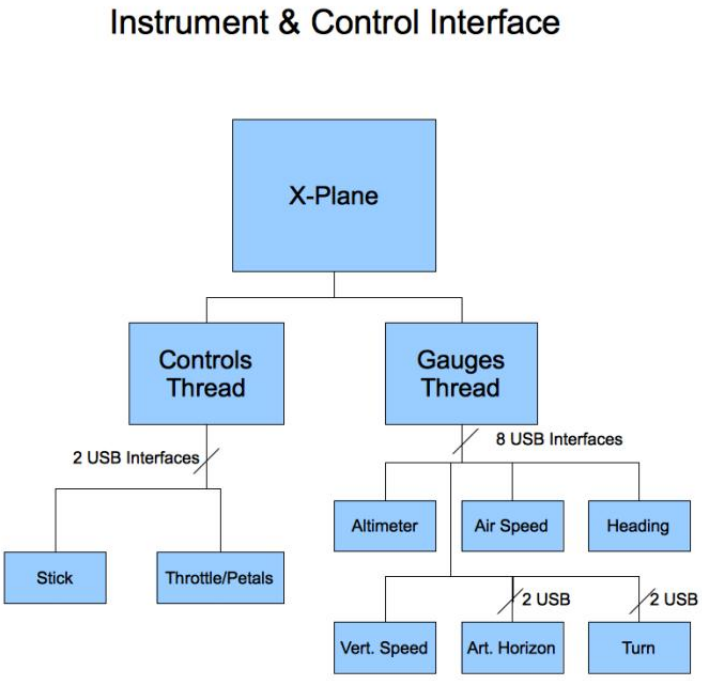
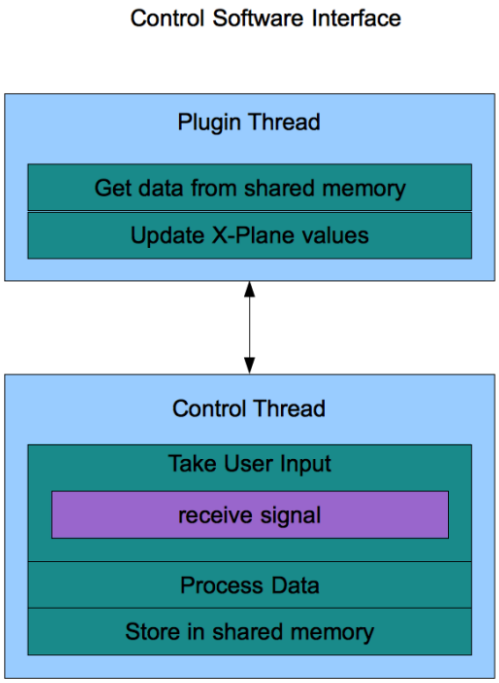


Figure 3.62. Instrument and control architecture. *Diagram by Lewis Vail.*



Figures 3.63. High-resolution control interface. *Diagram by Lewis Vail.*

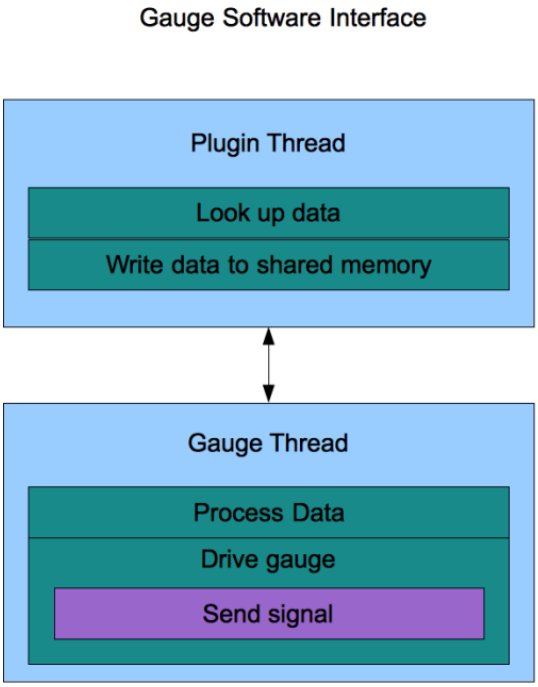


Figure 3.64. High-resolution flight instrument interface. *Diagram by Lewis Vail*

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.

3.9 Power Supply

No matter what we were doing we needed a power supply. Since we didn't get the computer that we planned and had to use a laptop, we just used a computer power supply, for the external power, the USB hub and any of the other circuitry that is was used, used this power supply.

3.9.1 Peripheral Devices Power Supply

The peripherals will need at most a 12 volt supply to run the motors in the gauges. We could of designed a power supply for each of the circuits and use this supply for each of the needed devices. Another alternative is to use a molex pass-through card that can give you the same voltages from the computer supply on the outside of the computer. It can be found here performance-pcs.com for $4.00 this is cheaper than building our own and will give us a steady 12 or 5 volts to use. Since we didn't get the computer we were not able to get the pass through cards and that can be added later, we had to go with our separate power supply for the simulator.

The power supply that we decided to use has an output power rating of 1000W with six 12V lines rated for 20A each which should be more than enough to power our gauges and controls. All that needs to be added are some extenders and splitters for the molex connectors. The following table is a summary of the power supply output as indicated by the manufacture and on the side label of the power supply[34].

Table 3-9 Power Supply Ratings

VDC_{out}	+3.3V	+5V	+12V	+12V	+12V	+12V	+12V	+12V	-12V	-5V
$I_{max,out}$	28A	28A	20A	20A	20A	20A	20A	20A	0.8A	6.5A
$I_{min,out}$	0.3A	0.3A	0.5A	0.5A	0.5A	0.5A	0.5A	0.5A	0.1A	0.1A

Since we will only need at most 1 amp total current draw on each of the gauges which is well below the power ratings of the power supply. In addition this will give us a safety feature in that we are reducing the number of devices to be plugged into a single wall receptacle which makes set up much easier. This is especially true at shows where this simulator might be displayed, where there is

[34] (*2009, Dec.). Power Supply Unit Specifications [Online]. Available:* http://epowertec.com/power_ep-1000p10-t2.html

limited availability of power receptacles available for exhibitor use, such as outdoor areas at airports during airshows.

3.10 Remote Instructor Operator Station

One of the feature requests of our project sponsor was to have potential ability to implement a remote Instructor Operator Station (IOS) in order to dynamically change flight simulator characteristics. First it should be noted that as part of the scope of this project we were not responsible for building a second computer to play host to the IOS functions. Instead the use of an existing computer, like a laptop or netbook, to run the IOS functions for us.

X-Plane 9.4 provides many angles of attack for providing an IOS to the simulator user. The first option that X-Plane provides is to simply draw the IOS on a secondary monitor. This is inconvenient due to the fact that it will obstruct the view of the individual flying an aircraft in the simulator and give away anything the instructor may try to throw at the pilot. Luckily X-Plane provides another to interface with an IOS console. Using the local network and either TCP/IP or UDP, we can either write a custom application or simply purchase another copy of X-Plane. The beauty of purchasing a second copy of X-Plane is that it already has the IOS console built in and all we have to do is simply connect to the host (simulator computer) machine. From there the instructor can change weather effects, add flocks of birds in the air or deer running across a runway, change the aircraft position or speed, and also add other aircraft operating in the proximity of the piloted aircraft. In addition the instructor can view the aircraft gauges as well. For the purposes of this project, utilizing the built in X-Plane IOS over a local network is the most efficient use of resources and keeps the project sponsor from being locked into a custom application.

3.11 Aircraft Model

One of the major requirements of this project was to build an accurate as possible flight model for the GoBosh G700S. In section 2.2.1.2 we discuss the requirements that we developed for our model. For example Table 2-6 on page 12 covers all of the basic aircraft data available from the GoBosh Flight Manual. However, when it comes to developing the model, this information while helpful is only a part of the required information to complete the task. Section 2.2.1.2 also covers additional information on the background on the operation of the Plane Maker tool. This will not be discussed in this section.

It should be noted now, that none of the group members had any experience working with any sort of 3D modeling software or have any strong background in flight physics. As a result, we were only able to make our model to the best of our abilities given our limited knowledge and experience. Still, we were able to generate a model that bore a strong resemblance to the actual aircraft (Figure 3-65 below).

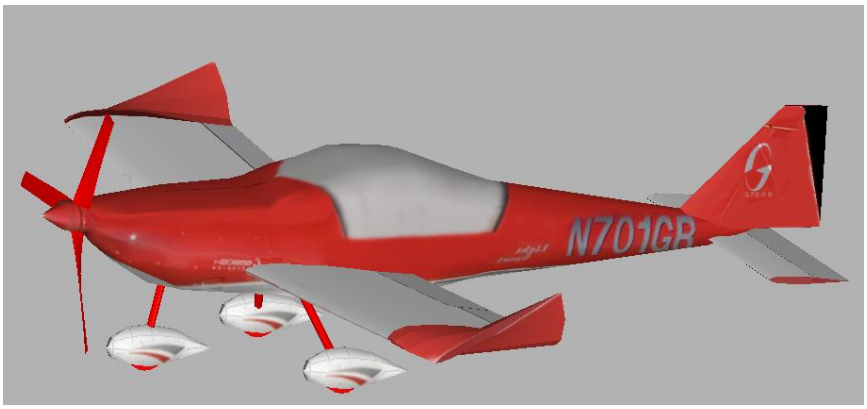


Figure 3-65. Finished Aircraft Model. *Image by Robert Gysi*

3.11.1 Model Generation

Generating the model was a fairly difficult process, although was made much simpler thanks to the included Plane Maker that ships with X-Plane 9. In order to generate our basic fuselage shape we utilized dimension drawings obtained from the manufacturer. We contact one of Aero's design engineers who was more than willing to provide us with this information. From these drawings we were able to trace our fuselage shape into the plane maker. While the drawings we used were dimensioned, it is possible that our fuselage is not the exact length of the actual aircraft; however by using the same drawings for each background image, we are confident that it is at least to scale (and should still be fairly close to the full size). This process is shown in Figure 3-66 below.

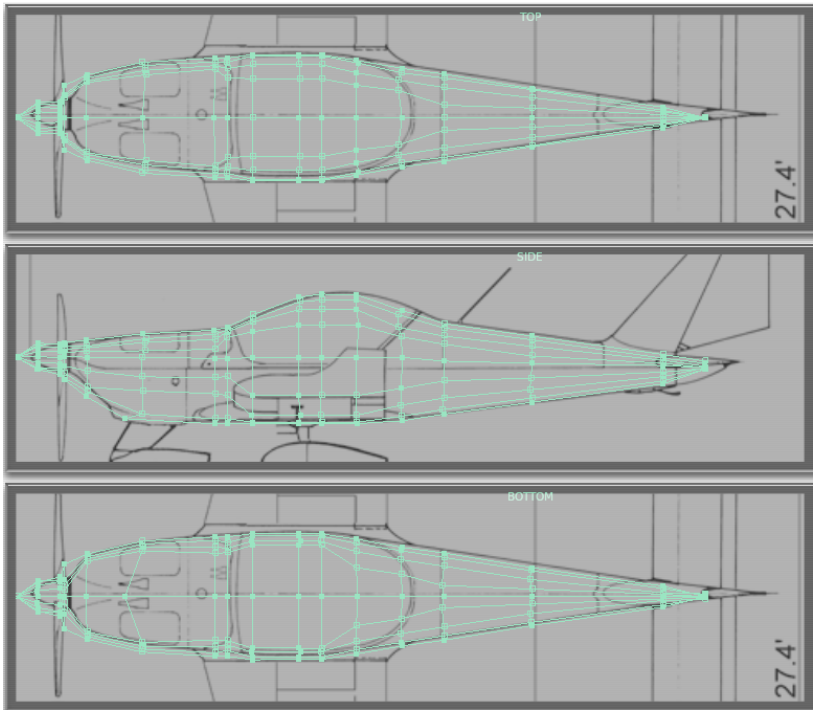


Figure 3-66. Wireframe model traced over dimensioned drawings. *Image by Robert Gysi*

The wings were much easier to implement, as all that really needs to be done is to specify the length of the wings and place the control surfaces in the correct location. Due to our simulator not having controls for the flaps, this feature was not implemented. However, the ailerons are required to the fly the aircraft and they were placed that the location specified in our dimensioned drawings. Also in terms of control geometry, we needed to be able to specify the chord ratio of the ailerons. Using our drawings again we found that the ratio was approximately 0.20 of the total surface area of the wing. In order to create the winglets on the tip of the wings we simply created a new wing section and tweaked the incidence values in the plane maker until we had a similar shape.

The horizontal stabilizer is where we first ran into problems when modeling the aircraft. Due to a limitation in the Plane Maker software we are not able to change the point about which it rotates. In the actual GoBosh it pivots from the leading edge of the stabilizer, but in Plane Maker it must pivot from the center of the control surface. This is a limitation that according to the creator of X-Plane will be corrected with a future release. We still attempted to ensure that the proper control reactions were the same, so utilizing flight manual we input values of 20° for the upper range of motion and 10° for the downward range of motion. Similarly, the vertical stabilizer was not able to be accurately modeled in Plane Maker as well. There is not an easy way to change the base of the rudder so that it is angled upwards. We even contacted Austin Meyer, the creator of X-Plane to see if he had any ideas for implementing this, and admitted that there was not a good method to do this within Plane Maker. As a result we attempted to maintain the other aspects of the shape of the stabilizer, so to as hopefully maintain as close as possibly flight characteristics. Figure 3-67 below shows a comparison of this section of the aircraft in both real life and in our model.

Figure 3-67. Actual versus Model. *Photo by Lewis Vail, Model by Robert Gysi*

Another issue we had with the model stems from specifying the engine. We attempted to utilize the specs for the Rotax 912ULS, the engine that is shipped with the GoBosh, but this unfortunately would not provide enough power to allow the plane to take off. This was probably due to an issue from modeling our

aircraft, as there may be something that does not match the GoBosh's flight profile at all. As a result we had to customize our engine specifications, using other LSA models as a starting point. Ultimately, we did find engine specs that would allow the plane to take off and fly, although it is possible to go slightly faster than maximum speed that the GoBosh is rated for. Table 3-10 below highlights these differences.

Table 3-10. Engine Specifications

	Rotax 912ULS[35]	**Actually Used**
Horsepower	98.5 hp	180 hp
Redline RPM	5800 RPM	2700 RPM
Idle RPM	1400 RPM	500 RPM
Continuous RPM	5500 RPM	2500 RPM

3.11.2 Airfoil

One of the particular aspects we wanted to attempt to accurately model was the flight physics. To do this we needed to create an airfoil for the NACA 4415 wing profile that this aircraft uses for its wings. While we lack the basic aerospace principles to fully understand this process, the steps for creating the airfoil were covered extensively on the X-Plane community message boards. To start, we got the polar coordinate data file from the University of Illinois at Urbana-Champaign Applied Aerodynamics Group for the wing profile on their public website. From there we needed to utilize an application known as javafoil to convert this file into a format that X-Plane can understand. This is a fairly time consuming process as one must wait for all the calculations to be completed and then remove extra information from the .afl file it generates.

[35] *(2009 Dec.). Airplane Flight Manual Aero AT-4 Light Sport Airplane [Online]. Available: http://www.ussportaircraft.com/uploads/Gobosh_POH_1_.pdf*

Chapter 4

4.1 Project Implementation

Following the completion of our Preliminary Design Review with our sponsor on January 3, 2010 we started our build phase. All of the design work at this point was considered completed and we were still slated to receive a cockpit to integrate at this time. In mid-March, we were told that we would not be receiving the cockpit as planned and which necessitated several design changes. This included building our own joystick and pedals and not procuring the computer we would have powered our simulator with. Upon demonstrating our project and presenting to the review committee our project has been completed with all the hardware and software being handed over to our project sponsor.

4.2 List of Required Parts

The following table (Table 4-1) lists the required parts needed to implement the design contained in Section 3 along with distributors in Table 4-2.

Table 4-1. Required Parts

Item	P/N	Qty. Req'd.	Vendor
USB Communication Board	FTDI245BL	10	Saelig
IC Sockets (Assorted)	Various	-	SkyCraft/Radio Shack
PCB Boards (Small)		4	Radio Shack
Wire		3	Radio Shack
Transistor	2N3904	36	Radio Shack
Diodes	1N4003	36	Radio Shack
Spacers (Assorted Lengths)	N/A	-	SkyCraft
Stepper Motors		8	RoboKits World
Terminal Blocks	N/A	36	SkyCraft/Radio Shack
PCB Boards (Large)	N/A	7	SkyCraft
Buffer Chip	CD4050	6	Futurlec
Comparator IC	LM741CN	5	Futurlec
A/D Converter	ADC0804LCN	3	Futurlec
Powered USB Hub		1	Best Buy
USB Cables	Various	9	Big Lots
Slide Potentiometers	RA6020F-10-20D1-B10K	3	Mouser
Molex Connectors	Various	-	Radio Shack
Epoxy Putty	N/A	1	Home Depot
3/8 x 0.035 Aluminum Tube	N/A	1	Cent. FL Hobbies
Misc. Hardware	N/A	-	SkyCraft/Home Depot
470-Ohm Resistors		3	Radio Shack

Table 4-2. Vendor Contact Information

Vendor	Web Site	Phone Number
Best Buy	http://www.bestbuy.com	407-482-8099
Radio Shack	http://www.radioshack.com	800-843-7422
DLP Design	http://www.dlpdesign.com	469-964-8027
Cent. Florida Hobbies	http://orlandohobbyshop.com	407-295-9256
Home Depot	http://www.homedepot.com	321-235-3600
Mouser Electronics	http://www.mouser.com	800-346-6873
Futurlec	http://www.futurlec.com	None
Saelig	http://www.saelig.com	585-385-1750
Robokits World	http://www.robokitsworld.com	None
Big Lots	http://www.biglots.com	407-380-3755

4.3 Build Phase

The build phase for this project commenced after our Preliminary Design Review with the project sponsor, which was completed on January 3, 2010. A full schedule representing our build phase as displayed in a Gantt chart can be found in Appendix B. A simplified view of the progression through this phase is presented in figure 4-1 below.

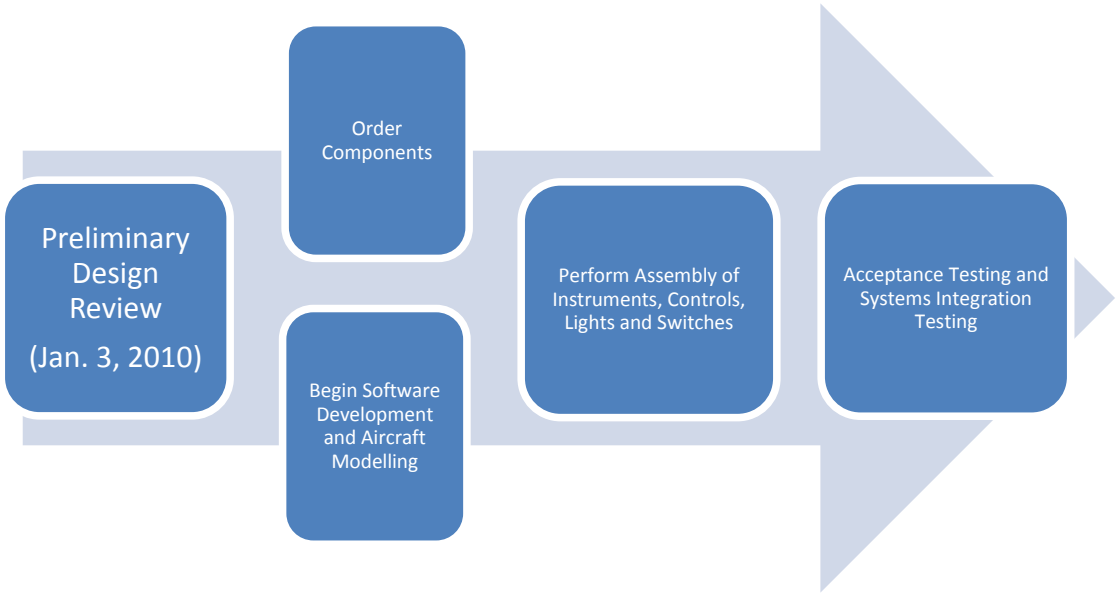


Figure 4-1 Overview of the Build Phase

4.3.1 Flight Instrument Assembly

The assembly of our simulated flight instruments occurred immediately after completion of our design review with our sponsor. We began to prototype our boards at this point and working out issues with the circuits. During this time we also attempted fabrication of our mounting decks, and dertermined through this exercise that having parts fabricated was the best option. Our original schedule

had us finishing our flight instrument build phase by the first week of March 2010. Unfortunately, due to a variety of factors this milestone was not reached. This was due to the fabrication time of our cut aluminum. Additionally we had to work out issues with gearing and other mechanical related issues. We did complete the construction of each component however within an updated schedule, which left us with plenty of time for testing of our project. All the work on the flight instruments occurred in the senior design lab at UCF.

4.3.2 Flight Control Assembly

Flight control assembly occurred, unfortunately at the last minute of the build phase. The reason behind this was up until mid-March we were still assuming that we would be receiving a cockpit to integrate with. As a result, while we had our electronics finished and tested using rotary potentiometers, we were behind schedule. The design of mechanical interfaces was not part of the original scope of this project and as a result we had to focus efforts on additional designs. Fortunately, we were able to get this completed before our testing phase commenced. For more information on the design of the controls, refer to Chapter 2.

4.3.3 Indicator and Switch Assembly

No work was completed on the indicator lights or swtiches. This was due to the lack of time for completing the project and in addition to not demonstrating at Sun ‘n Fun. These were never required components of the simulator, but features we designed in case we had the time available.

Chapter 5
5.1 Overview

This section contains all of our test procedures for testing the individual components to be installed in the aircraft cockpit as well as the final test procedure to ensure that the system as a whole works correctly. This is critical as our project ultimately will wind up in the hands of our project sponsor, and this an excellent method for us to perform quality control on our components. In addition to test procedures and results, this section also includes the usage cases as well as the requirements verification.

5.2 Required Test Equipment

In order to perform the testing the acceptance testing in section 5.3, some equipment will be needed. The list below includes the required items.

- PC running Windows 7 Professional.
- Latest version of X-Plane (currently version 9.4).
- Digital Multimeter (to troubleshoot any electrical issues that may arise).
- Computer screwdriver set (for making adjustments to mechanical components if necessary).
- Second computer (such as a laptop) for running the Instructor Operator Station (IOS) during integrated systems testing.
- Test application for light, switches and motor control testing
- USB Cables
- USB Hub
- Oscilloscope for troubleshooting issues with motor control.

In addition to this test equipment will have written a test program that we can use to test the indicator lights, switches, and the gauges. There will be the ability to turn on and off the lights, test the response of the switches (when the switch is thrown, the checkbox will become selected), and a tab that will allow us to test each of the motor control circuits for the gauges. In regards to the gauge test, there will be a slider control with a range representing 0-100%. For the gauges that need to continue to rotate it will only rotate one full revolution. In addition this will allow us to verify that a microcontroller is in 100% working order before we create any boards, and thus help eliminate the chance of a possible expensive mistake from occurring. Figure 5-1 on the next page highlights the test application.

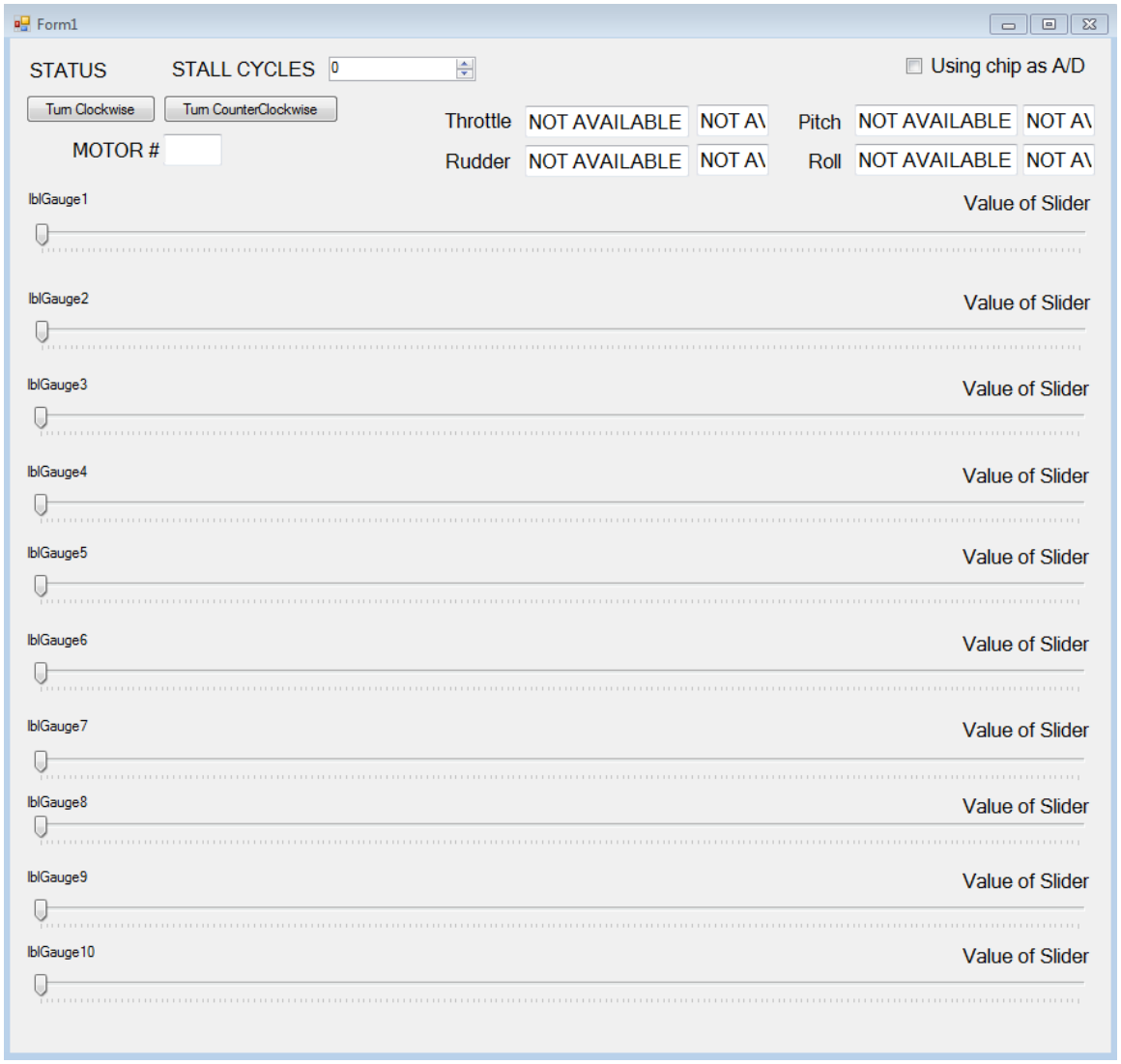


Figure 5-1. Gauge/Motor TestForm

5.3 Test Locations

All testing occurred in the senior design lab in room ENGR 456 on the UCF campus. This included the testing of our parts as we receive them as well as our integrated systems testing. Originally, the integrated systems testing was to take place at the hanger of our sponsor. This was due to this being the location of the stored cockpit, if it had arrived from Poland.

5.4 Acceptance Testing

The purpose of the acceptance testing is to verify that as we completed building each component we could immediately verify if the component is working 100% according to our specifications and requirements or if there are deficiencies that need to be corrected before we install the component into the instrument panel. This was our way of performing quality control on our components, so that once we install a component, we should not need to replace it due to a failure.

Controls are to be tested through the use of their test rig assemblies built specifically for the demonstrations.

5.4.1 Part Testing

This section is to verify that the critical components that were to be installed in our instruments and controls perform as specified from the manufacturer. We will require this of our most critical component: the FTDI USB communication board. This board will need to be tested before any are installed onto a board as a faulty chip will not only cost us the price of the chip, but also the time it takes to receive a replacement component. The procedures for this follows in section 5.4.1.1.

5.4.1.1 Microcontroller

The purpose of the microcontroller test is to verify that the part is received in working order. If the test results in any failures, a new part will need to be ordered or other corrective actions. This will be tested using the test software installed on the test computer.

No.	Testing Action	Result
1.	The microcontroller is internal to the Gauges that are being used in order to test it you must plug it into the computer through the USB port. Computer should recognize the device	**P** / F
2.	In the Application there is a Test Tab open it you will find a list with all the connected gauges. Make sure your gauge is in the list.	**P** / F
3.	Depending on the gauge you will be able to test the max and the min of the gauge. Slide the bar between max and min and the gauge should move with the slider.	**P** / F
4.	Repeat steps 1-3 for each gauge / motor	-
5.	We will now verify the operation of the switches and lights through the microcontroller to ensure that we have no defective parts. Disconnect the microcontroller responsible for the gauge tested in the previous step and connect the switches and lights up individually up to the microcontroller.	**P** / F
6.	The microcontroller is externally connected to all the switches and lights in order to test this controller connect the controller up to the computer through the USB port it should be recognized	**P** / F
7.	In the Application there is a Test Tab open it you will find a list with all the connected gauges. You will see a section for the switches and lights. You should see the current state of all of the switches and lights connected to that particular device	**P** / F
8.	Depending on the switch you are testing you will see the	

	switch change in the test program as well.	**P** / F
9.	Do this for all the switches and lights that are being connected.	**N/A**
10.	Overall Result	**Pass**

From the completion of the above test procedure, we were able to verify that each of the FTDI chips was working upon arrival from the distributor. In order to knot waste paper, the cumulative results of the testing for the entire batch of development boards was recorded the table above. Step 9, while included in the procedure was not tested as lights and switches were not implemented as part of this project.

5.4.2 Flight Instruments

This section of the acceptance testing will cover the flight instruments or gauges to be installed in our cockpit. Gauges to the tested will include the airspeed indicator, altimeter, attitude indicator, turn coordinator, heading indicator and the vertical speed indicator. Success will be determined if all of the test steps results in a "pass". Any failures will need to be corrected before being installed in the instrument panel. If necessary a redesign will occur, if successive fails are generated by the component in question. Additionally, any comments regarding the test events are included in paragraphs following the result tables.

5.4.2.1 Airspeed Indicator

The purpose of this test is to verify that the assembled component has been properly manufactured. If the test results in any failures, a replacement parts will need to be ordered or other corrective actions performed. This will be tested using X-Plane on a test computer.

No.	Testing Action	Result
1.	Perform Visual Inspection of the airspeed indicator. **WARNING:** Ensure airspeed indicator is disconnected from the USB Port and that the device is not powered. 1. Ensure that all contacts are soldered properly. 2. Verify that the indicator motor is clean of and there are no obstructions to the movement of the gauge. 3. Verify wiring to/from the FTDI USB controller is in accordance with the schematic diagram.	**P** / F
2.	Plug in the airspeed indicator into a free USB port on the simulation computer. Verify that the computer recognizes the device.	**P** / F
3.	Perform operational testing utilizing X-Plane 9. 1. Launch X-Plane and set up with an aircraft on a runway idling. Ensure the throttle is set to zero. 2. Ensure you are in the cockpit view in X-Plane. We will	

No.	Testing Action	Result
	want to verify that the same indicated airspeed is displayed on the virtual instrument on the screen and our simulated instrument. 3. First release the aircraft brake by pressing the B key on the keyboard. Then using the throttle control increase the power to at least 40 kts. Verify that the physical gauge matches the airspeed indicated in X-Plane. Verify that the gauge moves at the same rate as indicated on the screen. 4. Bring the aircraft to a halt. Verify that the gauge returns to zero. If it does not return to zero, note where it stops. This is important as we will need to potentially adjust the calibration of the gauge if it does not return to zero. 5. Repeat steps 3 and 4. Ensure that the data again matches on both the screen and on the physical gauge installed in the cockpit.	**P** / F
4.	Overall Result	**Pass**

5.4.2.2 Altimeter

The purpose of this test is to verify that the assembled component has been properly manufactured. If the test results in any failures, a replacement parts will need to be ordered or other corrective actions performed. This will be tested using X-Plane on a test computer.

No.	Testing Action	Result
1.	Perform Visual Inspection of the altimeter. **WARNING:** Ensure altimeter is disconnected from the USB Port and that the device is not powered. 1. Ensure that all contacts are soldered properly. 2. Verify that the indicator motor is clean of and there are no obstructions to the gauge movement. 3. Verify wiring to/from the FTDI USB controller is in accordance with the schematic diagram.	**P** / F
2.	Plug in the altimeter into a free USB port on the simulation computer. Verify that the computer recognizes the device.	**P** / F
3.	Perform operational testing utilizing X-Plane 9. 1. Launch X-Plane and set up with an aircraft on a runway idling. Ensure the throttle is set to zero. 2. Ensure you are in the cockpit view in X-Plane. We will want to verify that the same altitude is displayed on the virtual instrument on the screen and our simulated instrument.	

No.	Testing Action	Result
	3. First release the aircraft brake by pressing the B key on the keyboard. Then climb to an altitude of 900ft above sea level. Verify that the physical gauge matches the altitude in X-Plane. Verify that the gauge moves at the same rate as indicated on the screen. 4. Now climb to a level of 2300 feet above sea level. With this increase in altitude the thousands hand on the gauge should move. Verify that the altitude matches the result displayed in X-Plane. If the thousands hand is not correct, check the gearing of the motor. 5. To ensure that we can roll back, decrease the altitude to 500 feet above sea level. Verify that the physical gauge matches the value given on the virtual gauge in the simulation software.	**P** / F
4.	Overall Result	**Pass (With conditions)**

With the altimeter, everything from an electrical standpoint works as designed. However, from testing we discovered that the gearing we utilized eventually causes the gauge to be off at high altitudes. We can partially correct this by adjusting the barometric pressure in X-Plane on the virtual gauge. In order fully fix this new gears would need to be installed. Unfortunately, this was discovered late in the process, as this gauge took a very long time to construct.

5.4.2.3 Attitude Indicator

The purpose of this test is to verify that the assembled component has been properly manufactured. If the test results in any failures, a replacement parts will need to be ordered or other corrective actions performed. This will be tested using X-Plane on a test computer.

No.	Testing Action	Result
1.	Perform Visual Inspection of the attitude indicator. **WARNING:** Ensure attitude indicator is disconnected from the USB Port and that the device is not powered. 1. Ensure that all contacts are soldered properly. 2. Verify that the attitude indicator was constructed in accordance with the manufacturer's specifications. Verify mechanical assembly and electrical schematic. 3. Verify that the indicator motor is clean of and there are no obstructions to the movement of the gauge.	**P** / F
2.	Plug in the airspeed indicator into a free USB port on the simulation computer. Verify that the computer recognizes	**P** / F

No.	Testing Action	Result
	the device.	
3.	Perform operational testing utilizing X-Plane 9. 1. Launch X-Plane and set up with an aircraft on a runway idling. Ensure the throttle is set to zero. 2. Ensure you are in the cockpit view in X-Plane. We will want to verify that the same position is indicated on the screen and with our simulated gauge. 3. First release the aircraft brake by pressing the B key on the keyboard. Take-off and then climb to any altitude. As you are climbing the attitude indicator should indicate that the plane is at an increased pitch (in the blue region). Verify that the same level is indicated on the physical gauge and in X-Plane. 4. Put the aircraft into level flight. Verify that the attitude indicator rests on the line representing the horizon (between the blue and brown sections). 5. Roll the wings to the left and to the right. Verify that the result on the gauge matches the movement of the aircraft on the screen and the gauge on the screen. 6. Put the aircraft nose down. The attitude indicator should roll forward into the lower half of the gauge (brown section) as you head towards the ground.	**P** / F
4.	Overall Result	**Pass**

5.4.2.4 Turn Coordinator

The purpose of this test is to verify that the assembled component has been properly manufactured. If the test results in any failures, a replacement parts will need to be ordered or other corrective actions performed. This will be tested using X-Plane on a test computer.

No.	Testing Action	Result
1.	Perform Visual Inspection of the turn coordinator. **WARNING:** Ensure turn coordinator is disconnected from the USB Port and that the device is not powered. 1. Ensure that all contacts are soldered properly. 2. Verify that the turn coordinator was constructed in accordance with the manufacturer's specifications. Verify mechanical assembly and electrical schematic. 3. Verify that the indicator motor is clean of and there are no obstructions to the movement of the gauge.	**P** / F
2.	Plug in the turn coordinator into a free USB port on the simulation computer. Verify that the computer recognizes the device.	**P** / F
3.	Perform operational testing utilizing X-Plane 9.	

	1. Launch X-Plane and set up an aircraft on a runway idling. Ensure that the throttle is set to zero and you are in the virtual cockpit view. 2. Take-off and climb to 4000 feet above sea level. Keep the aircraft in level flight. At this point the aircraft pictured on the turn coordinator should be level and the ball below the aircraft on the instrument should be in the center. 3. Next make a turn to the right. The turn coordinator should match bank angle of the aircraft or in other words the right wing should be dipped to the right as indicated by the instrument. The ball should also move towards the right. Verify that the turn indicator in the virtual cockpit matches the result on the physical gauge. 4. Next make a turn to the left. The turn coordinator should match bank angle of the aircraft or in other words the right wing should be dipped to the left as indicated by the instrument. The ball should also move towards the left. Verify that the turn indicator in the virtual cockpit matches the result on the physical gauge.	**P** / F
4.	Overall Result	**Pass**

During testing our turn coordinator worked flawlessly. The bank angle of the aircraft was reported accurately and the ball worked as well during our testing and up to the day of the demonstration. Unfortunately, we did fry the FTDI chip that controlled the motor for the ball, possibly during our first demonstration attempt. As a result, this instrument will require the purchase of a new FTDI chip to be restored to full functionality.

5.4.2.5 Heading Indicator

The purpose of this test is to verify that the assembled component has been properly manufactured. If the test results in any failures, a replacement parts will need to be ordered or other corrective actions performed. This will be tested using X-Plane on a test computer.

No.	Testing Action	Result
1.	Perform Visual Inspection of the Heading indicator. **WARNING:** Ensure heading indicator is disconnected from the USB Port and that the device is not powered. 1. Ensure that all contacts are soldered properly. 2. Verify that the indicator motor is clean of and there are no obstructions to the gauge movement. 3. Verify wiring to/from the FTDI USB controller is in	**P** / F

No.	Testing Action	Result
	accordance with the schematic diagram.	
2.	Plug in the heading indicator into a free USB port on the simulation computer. Verify that the computer recognizes the device.	**P** / F
3.	Perform operational testing utilizing X-Plane 9. 1. Launch X-Plane and set up with an aircraft on a runway idling. Ensure the throttle is set to zero. 2. Ensure you are in the cockpit view in X-Plane. We will want to verify that the same position is indicated on the screen and with our simulated gauge. 3. First note the direction indicated on the physical gauge while on the runway. Verify that this matches with the heading indicator in the virtual cockpit. 4. Release the brake by pressing the B key on the keyboard, take-off and climb to any altitude. Once at an appropriate altitude turn to a heading of 330 degrees. Verify that the physical gauge moves smoothly in the correct direction to 330 degrees and matches the movement of the virtual gauge. 5. Put the aircraft back into level flight. Next perform a 360 degree turn to the right. Verify that the indicator goes around the full 360 degrees back to a heading of 330 degrees. Resume a forward heading and continue level flight. 6. Repeat part 5, but instead of turning to the right as stated, make a turn to the left. Verify that gauge works correctly and that you have returned to a heading of 330 degrees.	**P** / F
4.	Overall Result	**Pass**

5.4.2.6 Vertical Speed Indicator

The purpose of this test is to verify that the assembled component has been properly manufactured. If the test results in any failures, a replacement parts will need to be ordered or other corrective actions performed. This will be tested using X-Plane on a test computer.

No.	Testing Action	Result
1.	Perform Visual Inspection of the vertical speed indicator. **WARNING:** Ensure vertical speed indicator is disconnected from the USB Port and that the device is not powered. 1. Ensure that all contacts are soldered properly. 2. Verify that the indicator motor is clean of and there are no obstructions to the gauge movement. 3. Verify wiring to/from the FTDI USB controller is in	**P** / F

No.	Testing Action	Result
	accordance with the schematic diagram.	
2.	Plug in the vertical speed indicator into a free USB port on the simulation computer. Verify that the computer recognizes the device.	**P** / F
3.	Perform operational testing utilizing X-Plane 9. 1. Launch X-Plane and set up with an aircraft on a runway idling. Ensure the throttle is set to zero. 2. Ensure you are in the cockpit view in X-Plane. We will want to verify that the same position is indicated on the screen and with our simulated gauge. 3. Release the brake by pressing the B key on the keyboard, take-off and climb to any altitude. As you climb you should see the vertical speed indicator move in a clockwise fashion. Ensure that the movement mimics the virtual gauge on the screen. 4. Pitch the aircraft nose as far back as possible, putting the aircraft into a stall. Right before the stall the gauge should go no further than the established maximum on the gauge. 5. Recover from the stall (return to level flight) and pitch the nose towards the ground. The vertical speed indicator should now move in the counter-clockwise direction. Verify that this matches the gauge on the screen.	**P** / F
4.	Overall Result	**Pass**

5.4.3 Flight Controls

This section of the acceptance testing will cover the testing of our flight controls to that were installed in our instrument panel. Each step must result in a “pass” with any deficiencies noted for correction. Each individual component should pass before being installed to the instrument panel and before integrated system testing. Any comments about the testing follows the result tables in each section.

5.4.3.1 Joystick

The purpose of this test is to verify that the assembled component had been properly manufactured. If the test results in any failures, replacement parts will need to be ordered or other corrective actions performed. This was tested using X-Plane on a test computer.

No.	Testing Action	Result
1.	Perform Visual Inspection of Joystick Control. **WARNING:** Ensure joystick control is disconnected from the USB Port and that the device is not powered. 1. Ensure contacts on each of the slide potentiometers	

No.	Testing Action	Result
	are soldered correctly and that the wires lead to the correct pins on the A/D Converter board as specified on the schematic. 2. Ensure the entire yoke mechanical assembly including the wires leading to the slide potentiometers is connected and that there is no restriction in the movement of the stick.	**P** / F
2.	Plug in the joystick control into a free USB port on the simulation computer. Verify that the computer recognizes the device.	**P** / F
3.	Perform operational testing utilizing the Windows Control Panel. 1. In Windows 7 click Start → Control Panel → Devices and Printers → Right click on the icon associated with the yoke → Click Properties → Click on the Test Tab. 2. This is built in Windows Test utility for game controller and joysticks. First move the joystick in the positive X direction and then to the negative X direction. The crosshair should move up and then down. 3. Next test the Y-axis in the same fashion. Moving the stick to the left should move the crosshair to the left and moving the stick to the right should move the crosshair to the right.	**P** / F
4.	If the joystick passed the previous test, then we may verify that it works accordingly in X-Plane 9.4. First launch X-Plane and set up with an aircraft on a runway. 1. First release the brake on the keyboard (if enabled) by pressing the B key. Then using the throttle control increase the throttle until the RPM gauge in X-Plane moves and the aircraft moves down the runway. 2. Pull back on the stick when V_1 speed has been achieved. Ensure that the aircraft rotates off of the runway. Note if the aircraft is slow to respond to the joystick control. 3. Once airborne move the yoke in the direction of all four axes. Ensure that the response on the screen matches both the direction and the speed at which the yoke was moved.	**P** / F
5.	Return the joystick to center. It should stay in the center without moving in any direction.	**P** / F
6.	With the aircraft still in flight, verify that the rudder pedals move accordingly. Ensure that when pressing on the correct pedal that the aircraft moves in the same direction	**P** / F
7.	Overall Result	**Pass**

5.4.3.2 Throttle

The purpose of this test is to verify that the assembled component has been properly manufactured. If the test results in any failures, a replacement parts will need to be ordered or other corrective actions performed. This was tested using X-Plane on a test computer.

No.	Testing Action	Result
1.	Perform Visual Inspection of Throttle Control. **WARNING:** Ensure throttle control is disconnected from the USB Port and that the device is not powered. 1. Ensure contacts on the slide potentiometer are soldered correctly and that the wires lead to the correct pin on the A/D Converter board responsible for the throttle and pedals as specified on the schematic. 2. Ensure the entire throttle mechanical assembly including the wires leading to the slide potentiometers is connected and that there is no restriction in the movement of the throttle	**P** / F
2.	Plug in the throttle control into a free USB port on the simulation computer. Verify that the computer recognizes the device.	**P** / F
3.	Perform operational testing utilizing the Windows Control Panel. 1. In Windows 7 click Start → Control Panel → Devices and Printers → Right click on the icon associated with the yoke → Click Properties → Click on the Test Tab. 2. This is built in Windows Test utility for game controller and joysticks. To test our throttle, simply move the throttle out. The bar labeled 'slider' should move along with the throttle.	**P** / F
4.	If the throttle passed the previous test, then we may verify that it works accordingly in X-Plane 9.4. First launch X-Plane and set up with an aircraft on a runway. 1. First release the brake on the keyboard (if enabled) by pressing the B key. Set the throttle for full throttle and take off. Verify that virtual throttle position on the screen is roughly the same as the physical throttle. 2. Increase and decrease speed with the throttle while in level flight. Verify that it response on the screen matches the physical input.	**P** / F
5.	With the aircraft still in flight, verify that the rudder pedals move accordingly. Ensure that when pressing on the correct pedal that the aircraft moves in the same direction	**P** / F
6.	Overall Result	**Pass**

5.4.3.3 Pedals

The purpose of this test is to verify that the assembled component has been properly manufactured. If the test results in any failures, a replacement parts will need to be ordered or other corrective actions performed. This was tested using X-Plane on a test computer.

No.	Testing Action	Result
1.	Perform Visual Inspection of foot pedals. **WARNING:** Ensure pedals are disconnected from the USB Port and that the device is not powered. 1. Ensure contacts on the slide potentiometer are soldered correctly and that the wires lead to the correct pin on the A/D Converter board responsible for the throttle and pedals as specified on the schematic. 2. Ensure the entire throttle mechanical assembly including the wires leading to the slide potentiometers is connected and that there is no restriction in the movement of the pedals.	**P** / F
2.	Plug in the pedals into a free USB port on the simulation computer. Verify that the computer recognizes the device.	**P** / F
3.	Perform operational testing using X-Plane 9.4. 1. First release the brake on the keyboard (if enabled) by pressing the B key and then proceed to take off. 2. Once in the air, use the rudder pedals to change the position of the rudder on the tail of the aircraft. This is best observed when flying in chase view. Ensure that both the left and right pedals cause the correct change in direction of the aircraft on the screen.	**P** / F
4.	Overall Result	**Pass**

5.4.4 Cockpit Switch and Indicator Circuit Testing

The switches and indicator lamps circuits would have been tested to the same level as all other flight instruments and controls, if they had been implemented. The indicator lamps provide secondary information to the pilot and the indicator switches provide additional input commands, including turning on and off exterior strobe lights to the pilot. This was established as an optional requirement for the project. Not all of the switches may be functional, as implementation is solely up to our sponsor after handing over the project. Nonfunctional switches will be noted, so that they can be excluded from testing.

5.4.4.1 Indicator Lamps

The purpose of this test is to verify that the assembled component has been properly manufactured. If the test results in any failures, a replacement parts will need to be ordered or other corrective actions performed. This should be tested using X-Plane on a test computer. For the scope of this project, this component

was not implemented and therefore did not require testing. These procedures are included in this document for the reference of our project sponsor and/or future groups that may work on this simulator.

No.	Testing Action	Result
1.	Perform Visual Inspection of indicator lamps. **WARNING:** Ensure indicator lamp control board is disconnected from the USB Port and that the device is not powered. 1. Ensure of each LED is connected to the board correctly and that the overall circuit matches the board schematic.	P / F
2.	Plug in the indicator light control board into a free USB port on the simulation computer. Verify that the computer recognizes the device.	P / F
3.	Perform operational testing using X-Plane 9.4. 1. First start by setting up the aircraft so that there is only 2 gallons of fuel available. This should trigger the low fuel light. 2. Turn off the engine to the aircraft. Reconfigure the aircraft to have a higher amount of fuel. Start the aircraft using the keyboard command CTRL-1. The starter engaged light should come on as the engine starts. 3. To check if the generator failed indicator works properly, use the cockpit of the Cessna C172SP and toggle off the battery switch. This should cause the light to turn on. 4. Locate a fuel pump in the virtual cockpit. Click your mouse so that the switch is on. The light on the board should turn on. 5. Next, select an aircraft and take-off. Achieve level flight and a steady airspeed. Pitch the nose of the aircraft up quickly until the aircraft loses lift and the stall light turns on. This light should extinguish once the aircraft has achieved lift again.	P / F
4.	Overall Result	

5.4.4.2 Switches

The purpose of this test is to verify that the assembled component has been properly manufactured. If the test results in any failures, a replacement parts will need to be ordered or other corrective actions performed. This should be tested using X-Plane on a test computer. For the scope of this project, this component was not implemented and therefore did not require testing. These procedures

are included in this document for the reference of our project sponsor and/or future groups that may work on this simulator.

No.	Testing Action	Result
1.	Perform Visual Inspection of switches. **WARNING:** Ensure switch control board is disconnected from the USB Port and that the device is not powered. 1. Ensure of each switch is connected to the board correctly and that the overall circuit matches the board schematic. 2. Ensure that each switch is in the off position.	P / F
2.	Plug in the switch control board into a free USB port on the simulation computer. Verify that the computer recognizes the device.	P / F
3.	Perform operational testing using X-Plane 9.4. 1. First start by setting up an aircraft on a runway with the virtual cockpit open. 2. Taking the switch that is desired to be tested and switch it into the on position. Verify that the switch in the virtual cockpit has moved to the on position as well. 3. For each switch implemented repeat step 2, until all implemented switches have been placed into the on position. 4. Next start turning off the switches one by one, ensuring that the result on the screen mimics the physical switch. 5. Repeat steps 2-4 once more to verify that the switch circuit is still functional after one full operational cycle.	P / F
4.	Overall Result	

5.5 Integrated Systems Testing

The purpose of integrated systems testing is to validate the install of the components as a whole and ensure that each system works together in a combined environment. This represents the final phase of testing before the project can be considered complete and allows any issues to be corrected before the project deadline and demonstration. In the integrated systems testing, each component was tested individually in a large scale test event using the acceptance test procedures. During the testing we had one individual operating the simulator, one individual ensuring that the data from X-Plane matched our physical gauges, and another individual keeping track of everything from the test computer screen.

No.	Testing Action	Result
1.	Perform Visual Inspection of cockpit. **WARNING:** Ensure that power is disconnected to all of the electrical components, including the computer, before performing the inspection. 1. Ensure that the mechanical components of the stick, pedals and throttle are all free of obstructions and that the electrical components have been properly installed in accordance with the schematic. 2. Verify that each of the gauges has been installed in the proper location. Check the mechanical connections on the motors to the gauge faces for any obstructions or misconnections. Verify that the electrical layout matches the appropriate schematic drawing. 3. Verify that the indicator lights have been installed in the instrument panel correctly. Verify wiring to the electrical schematic. 4. Verify that the indicator switches have been installed in the instrument panel correctly. Ensure that each one is seated properly with no movement of the switch housing when the switch is used. Verify the electrical connections with the schematic diagram. 5. Verify that the cockpit is clean of any debris. 6. Verify that the monitors are secured to the top of the cockpit.	**P** / F
2.	Plug in the power supply to the computer, the individual power supplies for the monitors, and any other required power supplies to a 115VAC, 60Hz receptacle. Plug in all USB cables into an empty USB port on one of the USB Hubs.	**P** / F
3.	Perform system start up. 1. Press the power button on the computer. The computer will boot into Microsoft® Windows 7 Professional. After Windows starts, double click on the X-Plane icon on the desktop. 2. X-Plane by default will load to the default aircraft and default airport. Select the airport KMCO – Orlando International Airport and select the GoBosh G700S aircraft model. 3. On our second computer launch X-Plane and connect to the IP address of the simulation computer. Open the Instructor Operator Station (IOS) window. We will use this to assist in verifying data output over the established network connection	**P** / F
4.	Perform Flight testing. This procedure will make reference to	

No.	Testing Action	Result
	our previous test procedures for the individual components. The goal here is to operate the aircraft under normal flying conditions while a second group member verifies that each component is working. The following procedures do not need to be followed in a specific order, as long as each step is verified. While each procedure is being verified, ensure that the same result is being displayed on the physical gauge in the cockpit, the virtual gauge in X-Plane, in addition to the data matching on the Instructor Operator Station computer as well. If there is a mismatch in the data being displayed on one of the computers or the physical gauge perform troubleshooting to determine which device is reporting the incorrect information to the user. 1. Verify Operation of the gauges. Perform the following sections from the acceptance testing to verify the install of each gauge. a. 5.3.2.1 Airspeed Indicator b. 5.3.2.2 Altimeter c. 5.3.2.3 Attitude Indicator d. 5.3.2.5 Heading Indicator e. 5.3.2.6 Vertical Speed Indicator 2. Verify Operation of the gauges. Perform the following sections from the acceptance testing to verify the install of each flight control. a. 5.3.3.1 Joystick b. 5.3.3.2 Pedals c. 5.3.3.3 Throttle 3. Verify the operation of indicator lamps. Perform the following sections from the acceptance testing to verify the install of each lamp. a. 5.3.4.1 Indicator Lamps 4. Verify the operation of the switches. Perform the following sections from the acceptance testing to verify the install of each switch. a. 5.3.4.2 Switches	**P** / F
5.	Restart X-Plane. Perform steps 3 through 4 again. Ensure that the gauges, switches, lights, and controls still work the same without needing calibration. If any gauges appear to not reset to zero, take note of which need adjustments along with the ones that reset with no issues.	**P** / F
6.	If the system performs with no issues on the second system run, then we can consider the system as having been certified in working order and built to our specifications and design.	**P** / F

At the completion of our integrated systems testing, all systems were functioning as they should and with the limitation on the instruments as noted in the acceptance testing section. Any issues that have arisen since testing are not covered in the above results.

5.6 Prototype Use Cases

The simulator being developed as part of this project was slated to be ultimately used as a demonstrator at the Sun 'n Fun airshow and aviation conference at Lakeland Linder Regional Airport in April 2010. Unfortunately, due to cockpit not arriving this use was never realized. Should the cockpit have arrived this simulator would have be used by the aircraft manufacturer to give prospective buyers seat time in a very realistic simulation of the actual aircraft. In this capacity it would have also be utilized to take those prospective customers and show how relatively easy (compared to other general aviation aircraft) that the aircraft is to fly. This was meant to assist in the selling of flight instruction courses for the actual aircraft.

The second usage scenario for our prototype is as a ground based instruction simulator. In the configuration being developed as part of this project it has the ability to give a new student basic lessons in aircraft control before setting off in the actual aircraft. However, those hours will not be able to be logged as flight time, due to the simulator not being FAA Certified. In order to achieve certification, the optional $500 USB key from Laminar Research would need to be purchased. This allows the student pilot to log up to ten hours of ground based training towards the completion of their sport aviation license. Although, we do not have the actual cockpit, our desktop simulator could possibly be used for this scenario. All that would need to be done is to procure the computer components to build the simulation computer and the three screens for the 120-degree field of view.

Beyond the scope of our efforts, is the use of this simulator at future airshows and general aviation conventions after the prototype has been turned back over to Mr. Kotick and Grizzly Aviation. It has been mentioned that one of the second type of events this would be taken to gatherings and trade shows such as the Orlando Home and Boat show. At this type of show, the goal would be to introduce individuals to the aircraft and flying in general. This use scenario is dependent on our sponsor receiving the cockpit from the manufacturer and moving our completed panel into the cockpit.

While our prototype did not wind up finished as designed, all of the components have been built and are working. It is now up to our sponsor to utilize this simulator as he sees fit for his business and expand upon the features we were not able to implement due to time or the cockpit not arriving.

5.7 Requirements Verification

After completing the test procedures and certifying that our project was built to our specifications and schematics, we needed to perform a requirements verification to ensure that each requirement we developed in Chapter 2 has been implemented. The requirements have been broken into two tables: hardware requirements and software requirements. In this final check of the simulator we have tallied what we successfully met, what have we partially met, and what requirements were not met. Most requirements were met overall, however as seen in the following sections there are some requirements that were either partially met or not met at all. The majority of these cases are the result of the cockpit not arriving. This has also been established as a requirement by our project sponsor in order to have traceability of the implementation of our requirements and that each component has been tested and found to be in good working order.

5.7.1 Software Requirement Verification

The following table determines compliance with the established requirements from the beginning of this document. All of these requirements were met through the selection and purchase of X-Plane as our simulation software.

Req. #	Sub. Req.	Requirement Description	Result
S1	-	Realistic Look and Feel: The virtual simulation environment mimics the look and feel of the real world as close as possible. This not only includes visual effects, but also how physics are applied to the environment.	Met
S1	A	Realistic Scenery: scenery has a natural feel and does not look jaded or ragged. Terrain meshes are of high enough resolution to navigate from the air.	Met
S1	B	Inclusion of Airports Worldwide: Ensure a wide variety of airports are installed.	Met
S2	-	Ability to change environmental factors dynamically: Using the X-Plane IOS screen or from the weather and time/season options in the menu bar.	Met
S2	A	Ability to Interface Hardware with software via API: Inclusion of X-Plane SDK to develop plug-ins to interface with gauges, controllers as well as other computers and data types.	Met
S3	-	Model Entertainment Aspects	Met
S3	A	Weather Effects: Ability to have a wide	

Req. #	Sub. Req.	Requirement Description	Result
		range of weather scenarios in X-Plane including rain, snow, wind, sheer effects, turbulence, lightning and strong waves in the water.	Met
S3	B	Crash Effects: When the aircraft is overstressed, or flies into the earth effects are generated by X-Plane end the simulation is ended.	Met
S3	C	Sounds: Realistic prop sounds. Either using default audio in X-Plane from a similar propeller driven aircraft or recorded sounds of an actual GoBosh G700S	Met
S3	D	Ability to create custom scenarios/missions: X-Plane has tools to create and save custom missions.	Met
S3	E.	AI Aircraft also utilizing airspace and airports: Available via 3rd party plug-ins and custom development using the SDK.	Met
S4	-	Aircraft Model	Met
S4	A	Aircraft Exterior Model: Complete and generated via the Plane-Maker tool.	Met
S4	B	Model parametric data: Data received and implemented from the aircraft manufacturer or other source	Met
S5	-	Ability to interface with other Flight Sim/X-plane games: X-Plane has built in multiplayer as well as the ability to interface with other simulators with an appropriate plug-in.	Met
S5	A	Native Multiplayer Support: Support over TCP/IP and UDP protocols included for an enhanced simulation experience through multiplayer gaming or through the use of	Met

Req. #	Sub. Req.	Requirement Description	Result
		an Instructor Operator Station.	
S6	-	Guaranteed minimum 30 FPS: Set in rendering options; ensure graphics settings are not overset so that there is no error while the simulator is launching that it is reducing graphics settings to maintain performance.	Met
S6	A	FAA Certification – Optional Requirement: Ability to be implemented with a $500 key. The ability to is the requirement, not the implementation.	Met
S7	-	Ability to interface controls/flight instruments: SDK to write control plug-ins for flight instruments and flight control s	Met
S8	-	Ability to interact with an Instructor Operator Station: Includes built in IOS or 3rd party applications.	Met

5.7.2 Hardware Requirement Verification

The following table determines compliance with the established hardware requirements from the beginning of this document. Several of these requirements were not met due to the cockpit not arriving and the subsequent decision to not build a computer as a result of not going to Sun ‘n Fun in Lakeland, FL. All of our primary requirements were met however for implementing controls and gauges.

Req. #	Sub. Req.	Requirement Description	Result
C1	-	USB interface for controls & gages: Motherboard provides enough free USB ports for all of the flight controls and instruments or requires the use of a USB hub.	Met
C2	-	120 degree field of view: Ability to in X-Plane as well as with chosen graphics adapters and monitors.	Partial

Req. #	Sub. Req.	Requirement Description	Result
C2	A	3 LCD monitors: Must be no smaller than 24” and secured to the fuselage of the aircraft.	No
C2	B	Graphics Card/Adapter: Powerful enough to output required resolution to 3 monitors with a resolution of approximately 1920x3240.	No
C3	-.	2Ghz 64-bit CPU (minimum): Established through X-Plane requirements.	No
C4	-	4GB RAM: Sufficient memory to run both Windows 7 Professional as well as the flight simulator.	No
C5	-	120GB Hard Drive (minimum): X-Plane requires around 72GB for a full install, and Windows 7 requires 20GB. 160GB recommended.	No
M1	-	USB Controlled: Has USB on the chip with little development required to implement computer communications	Met
M2	-	20ms refresh rate (minimum)	Met
M3	-	Use less than 5V to power the actual chip. Devices connected to the chip may use other values.	Met
M4	-	Minimum 8 I/O Pins for external communications	Met
M5	-	Fit inside of a 3.24”x3.24” profile.: For the aircraft gauges and alongside the flight controls.	Met
M6	-	Low Cost Microcontroller: Including not only the chip but also the development board.	Met

Req. #	Sub. Req.	Requirement Description	Result
M7	-	As self-contained as possible: Does not require any complex circuitry or boards to be manufactured outside of very simple boards that can be manufactured in ENGR 456.	Met
F1	-	Motor to drive flight instruments: Use of servo and stepper motors to drive flight instruments. Must be able to complete a turn of over 360 degrees for the altimeter and heading indicator. Other gauges need only to travel less than 360 degrees	Met
F2	-	Realistic flight instruments and controls: Flight controls are to be original, instruments should be as close to original manufacture specification as possible.	Partial
F2	A	Gauges: Standard Six-Pack has been implemented - Altimeter, Airspeed Indicator, Attitude Indicator, Turn Coordinator, Heading Indicator, Vertical Speed Indicator. Ensure each gauge matches or closely matches the actual gauge utilized in the G700S cockpit.	Met
F2	B	Flight Controls (Stick, Pedals, Throttle): Using existing controls from the GoBosh G700S to preserve realistic look and feel.	Met

Requirements F2 and F2A were both recorded to have partial compliance with our stated project requirements. In the case of requirement F2, this is due to the end result of our gearing being off on the altimeter and the FTDI chip that control the turn coordinator ball having fried. As a result, these requirements have been mostly fulfilled, but will need work after the project is handed over to our project sponsor to fine tune the results.

Chapter 6
6.1 User Manual
In this chapter we will discuss the proper operation of our flight simulator. We will start with the basic operation and cover some troubleshooting procedures if something unexpected occurs during use. This user manual assumes that the user is familiar with the Windows operating system and flight simulators.

6.2 Setup and Basic Operation
With this section we will describe how to setup the simulator for a first run. To start navigate Windows Explorer to your X-Plane directory. This is commonly found in C:\Program Files\ but could be located elsewhere depending on your setup. From here verify that you have the following directories:

- …\X-Plane 9\Aircraft\General Aviation\GoBosh
- …\X-Plane 9\Airfoils\NACA_4415.afl
- ....\X-Plane 9\Resources\plugins\GaugeTest.xpl
- ....\X-Plane\config.ini

Also verify that X-Plane 9.4 is the version that is installed. This software has not been tested with version 9.5 which was released in March 2010.
After verifying that you have the appropriate plug in and aircraft data installed, proceed to plug the seven port USB hub into one free USB port on the simulation computer. Plug the 4 port hub also into a free USB port on the simulation computer. At this point you should hear multiple audible alerts that Windows has detected new hardware. Each of the instruments and controls can be verified in the device manager of Windows.

After plugging in all of the devices, double click on the X-Plane icon located on the desktop. This will launch X-Plane. While the loading screen appears on your monitor, you should see each of the six simulated instruments move as they initialize. If a gauge does not move, proceed to section 6.3 for troubleshooting information.

Once X-Plane has loaded, you should be on a runway with the GoBosh loaded on the screen. If the GoBosh is not the aircraft on the screen then the GoBosh will need to be selected from the Aircraft menu on the top of the screen. It can be found in the category “General Aviation,”

Before flying, it is suggested that you ensure your control devices are calibrated. With the pedals, the gears can slip when extreme forces are applied during use. This causes the center to not be correct. Using the chase view of the aircraft, ensure that the rudder pedals are forward facing to you and that the rudder of the aircraft is in the 0° position. Additionally, take the time to move the joystick to ensure that the ailerons and elevator moves as expected.

At this point one can switch to the cockpit view by hitting “A” on the keyboard or remain in the chase view (to return to the chase view from the cockpit, hit the “W”

key). At this point you may release the brakes with the “B” key and increase the throttle. To increase the throttle, one just needs to push in on the rod that is located in the center of the instrument panel. To decrease throttle, simply pull back on the rod. Note: the throttle may get caught as you pull out on a zip tie; this results in not fully decreasing your throttle. Simply pull up and back until you reach full stop.

Flying in the aircraft is fairly straight forward and works just the same as in an aircraft or with any other flight simulator. To increase altitude pull back on the stick. To decrease altitude, push forward on the stick. To go left, pull the stick to the left and to go right, pull the stick to the right. The rudder pedals have been designed to help add realism to the simulator. In order to use place your feet firmly on the foot rests or place your heel on the ground and toes on the foot rest. The pedals are attached with hinges so either operation will work. To turn the rudder to the left, push on the pedals with your right foot. To turn right, push on the pedals with your left foot. It may take a while to get the hang of the operation, but once successful, flying the plane will actually be easier.

Since the majority of simulator functions at this point are functions of X-Plane, please see the X-Plane User Manual at http://wiki.x-plane.com/Category:X-Plane_Desktop_Manual. This will cover all aspects of the simulator software.

6.3 Troubleshooting

At some point during the operation of the simulator, a component may fail or produce undesired results. This section will cover the steps to recover from these failures.

6.3.1 Inoperative Gauge

It is possible that a gauge may not properly work during simulator usage. This could be caused by a variety of factors including Windows not recognizing the device properly. The steps below should correct this issue.

1. Check the Config.ini file to see if the gauge is named correctly
2. Check to make sure the gauges and controls show up in the Device Manager Correctly
    a. Right Click on My Computer and select Manage, then select Device Manager
    b. Click on the USB and you should see all the connected gauges and controls
    c. Right Click on the gauge or control in question and select properties
    d. Select the Advanced tab
        i. You see that the VCP drivers are deselected
    e. If you don't see Advanced tab then you need to uninstall and reinstall the device, making sure the VCP drivers is deselected

6.3.2 Gauge does not Initialize Properly

If the gauge does not initialize properly on startup, the first thing to check is to ensure that the LED for the light sensor is on. If it is on ensure that it hits the light sensor. It is possible that during transport that these two could become misaligned and cause undesirable results.

For the attitude (artificial horizon), if it does not return to a perfect center, it is due to another issue. In this gauge there is no LED for calibration. Instead the issue is that the wire that goes to the stepper motor will become too tight around the shaft. Simply manually adjusting the shaft and rotating until it lines up rectifies this issue.

6.3.4 Control Device is not Recognized

If a control is not functioning in X-Plane, first verify that it is plugged into a USB port. If it is plugged in, follow the procedure below to troubleshoot the control.

1. Check the Config.ini file to see if the gauge is named correctly
2. Check to make sure the gauges and controls show up in the Device Manager Correctly
    a. Right Click on My Computer and select Manage, then select Device Manager
    b. Click on the USB and you should see all the connected gauges and controls
    c. Right Click on the gauge or control in question and select properties
    d. Select the Advanced tab
        i. You see that the VCP drivers are deselected
    e. If you don't see Advanced tab then you need to uninstall and reinstall the device, making sure the VCP drivers is deselected

6.4 FTDI Chip Programming

This section will explain how to install the drivers for a device that you wish to connect that uses the FTDI chip and how to program the EEProm on the chip to make the device whatever you want it to be. This will allow you to use the device with the GoBosh Simulator. Prior to doing this you should have:

- An idea of the gauge or control that you are creating
- FTDI Drivers can be gotten at the Code Site through a versioning software (tortoise SVN) or here http://www.ftdichip.com/Drivers/D2XX.htm
- A board wired up for connection to the computer(USB A/B cable)

Procedure:

1. Make sure the board is at least wired up to drive the FTDI chip.
2.

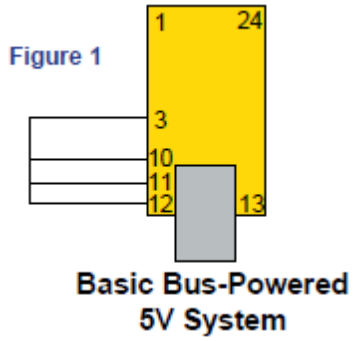


Figure 6-1. Basic Bus Powered 5V System. *Image by Chris Dlugolinski*

3. Connect the USB A/B cable to the chip and to the computer, since it is self powered as shown you don't need any external power.
4. In the FTDI Drivers folder you need to open the D:\FTDI Drivers\CDM 2.06.00 WHQL Certified folder
    a. You will see two files that you need to edit
    b. Ftdiport.inf and ftdibus.inf

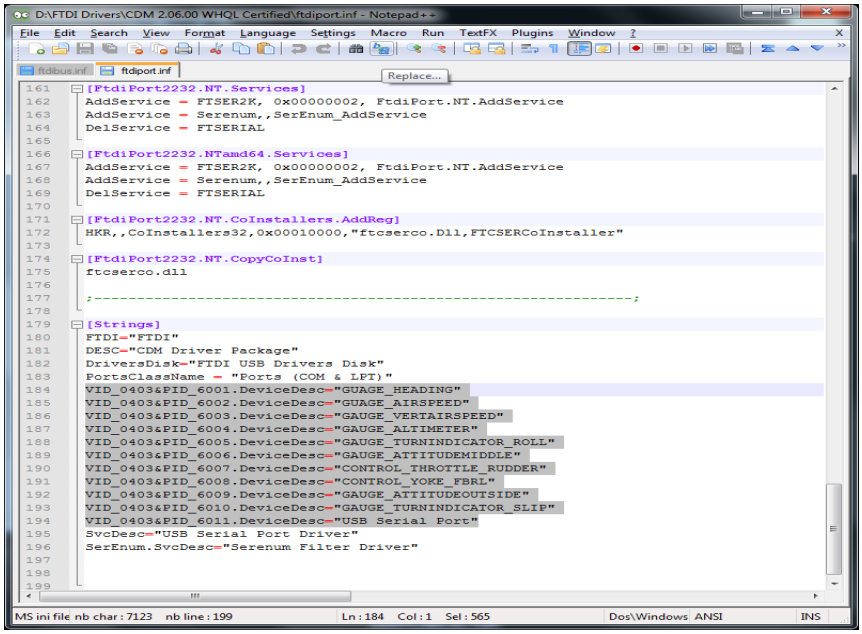


Figure 6-2. FTDIport.inf. *Screenshot by Chris Dlugolinski*

    c. The highlighted section above shows the section in ftdiport that you need to edit
        i. You need to add a new PID for your device and a name following the GAUGE_<name> or CONTROL_<name>

convention, this is for the plugin to work with the gauge or control

d. Do the same for the ftdibus.inf file as shown below (following the same naming convention)

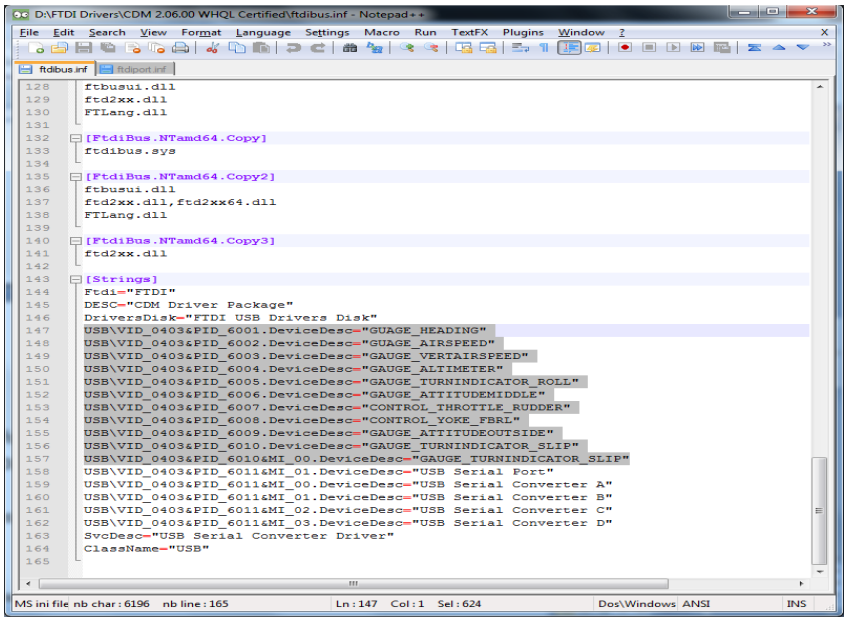


Figure 6-3. FTDIbus.inf *Screenshot by Chris Dlugolinski.*

e. There is one last file that needs to be edited before you can begin and that is the Config.ini file

f. Follow the other data format and fill in the info needed to make your control or gauge work correctly

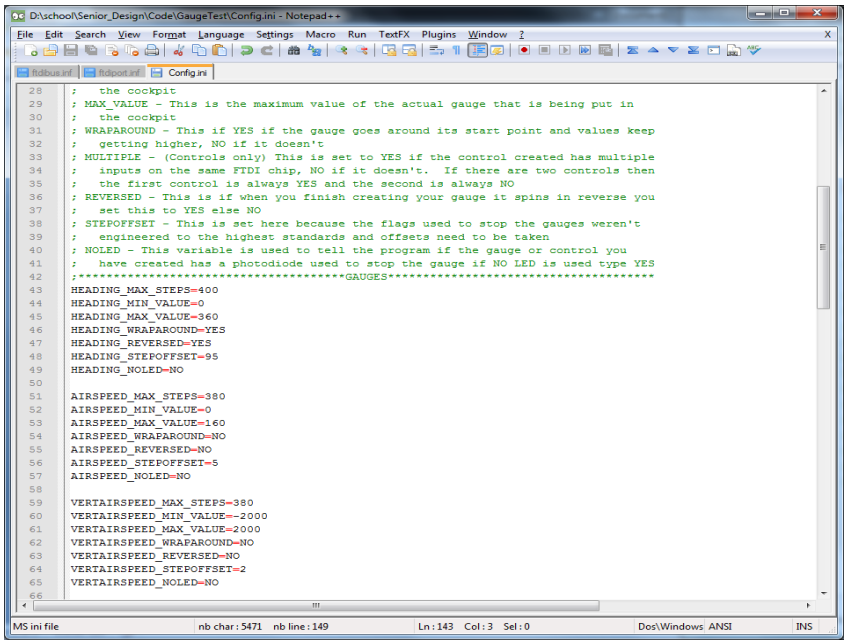


Figure 6-4. Config.ini file. *Screenshot by Chris Dlugolinski*

5. Plug in the FTDI chip
6. You will see the found new hardware window open cancel it and open the Device Manager
    a. Start->Right Click on My Computer and select Manage
    b. Click on Device Manager

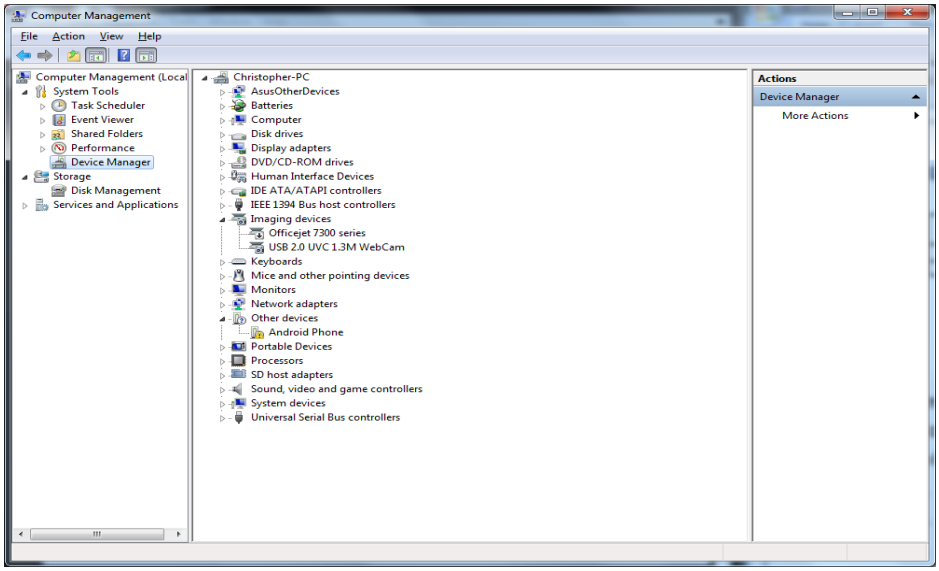


Figure 6-5. Device Manager. *Screenshot by Chris Dlugolinski*

    c. You will see a USB to Serial device with a Yellow Exclamation.
    d. Install the FTDI Driver, Right Click on the device and select Update driver
    e. You will need to find the directory where you saved the FTDI drivers and point to it
    f. Click continue anyway
    g. Once the drivers are installed you need to uninstall the VCP drivers
    h. Then unplug the USB and re-plug the device in, if it comes up again without the yellow exclamation you can continue if not reinstall the driver and uninstall the VCP
    i. Once installed you can change the device to what you have entered into the other files above using the FTD2XX.exe serializer program

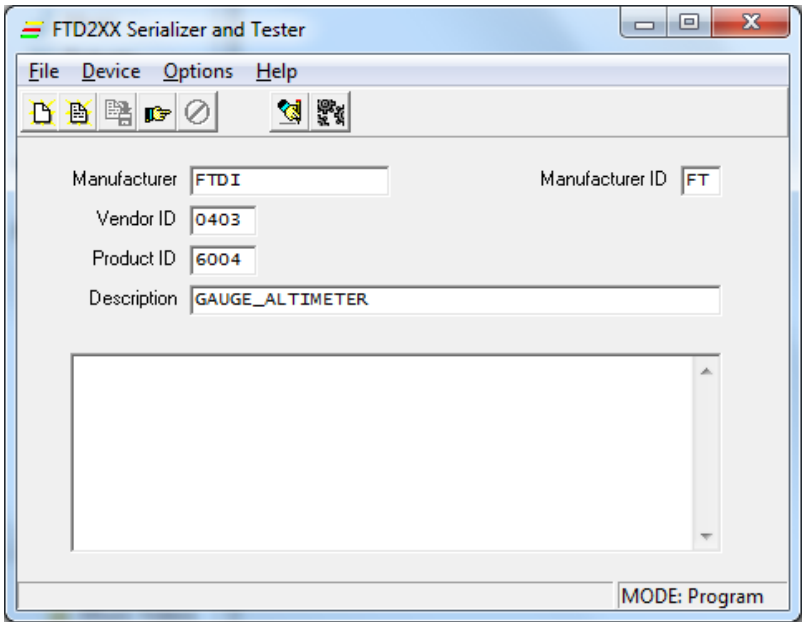


Figure 6-6. FTD2XX Serializer Program

j. First you need to enter the info as seen above just changing the description to your corrected name, and changing the Product ID to your corrected ID.

k. Then you will need to select the Advanced setup button  this will pop up a window you just need to select OK to

l. Then you will see that the other buttons are enabled press the save button 

m. Then you can program the chip using the Program button 

n. Now press the test button it will probably fail but that is fine 

o. Unplug the USB and re-plug in the device and see if the new name shows in the Device Manager. If it does OK, if not you need to Right Click on the device and Uninstall it then Scan for new hardware and you should find it. If Not then you have to check your .inf files to see if the PID and the Description you typed match in that file.

Upon completion of the above procedure, the new FTDI chip will have been programmed. At this point you can use the chip for the purpose you have set it up for.

Chapter 7
7.1 Summary

Our project has been completed according to our design specifications and we have ceased all further production. All the software and parts have been integrated and tested completing the test and production phase of the project. The following is where we stand with each component of the project.

The first part of the semester we spent all of our time doing expensive trade studies to decide on various implementations to peruse. For our simulation software trade study we chose X-Plane over Microsoft Flight Simulator because it better met our need overall. Once we picked this platform we did extensive research into the X-Plane SDK and decided on the best way to interface with the simulation software. We figured out the logical flow of our interface software, and have completed and tested the actual code. Example code available on the X-Plane SDK website was a great resource in the design and implementation of our software.

On the hardware end, we had looked at various parts for various applications and decided on the parts we wanted to use and for what components. We've used FTDI boards for all the gauges and other simple devices as well as for the control devices that generate an analog signal. We have completed writing and testing all the interface code for the FTDI boards. For the gauges we had explored every conceivable implementation and went with stepper motors exclusively for all six. For the electronics we successfully implemented our high level circuit design for our stepper motor gauges. Finally we acquired the proper motors and assembled and tested them successfully for all six gauges.

We have successfully completed every step of the design phase and the build phase and our project has been handed over to our project sponsor.

Appendix A: Trade Studies

A.1 Microcontroller Trade Study

Table A-1 Microcontroller Comparison

Microcontroller	Atmel AT89C5131	PIC18F4550	FTDI FT245BM
Dev Board	futurlec.com	futurlec.com	FTDI
Cost	Dev Board $35.90 Chip $10.11	Dev Board $46.90 Chip $14.99	Dev Board $30 Chip $5.00
Usb driver	http://www.atmel.com/dyn/resources/prod_documents/doc7646.pdf	CDC Firmware http://www.microchipc.com/sourcecode/index.php#pic18f4550usb http://microcontrollershop.com/product_info.php?products_id=2125	Free from FTDI to download and no programming on chip unless really necessary. Www.futurlec.com
speed	24 MHz	48 MHz	USB 1.1 or USB 2.0 (compatible)
Examples	Come with dev board	Come with dev board http://www.create.ucsb.edu/~dano/CUI/ http://www.edaboard.com/ftopic313796.html	http://electronicdesign.com/Articles/Index.cfm?AD=1&ArticleID=16125
Memory	32k	32k	External EEPROM
Memory RAM	1k	2k	
I/O	34	35	8 pin
Languages	c, assembler	c, c++, assembler	any
Power Needed	3.0V to 3.6V 30 mA Max Operating Current	3.3V detached 25mA	All usb self contained may need to do something for control of external
Thoughts			
Atmel AT89C5131	This seems to be a better choice all around including the fact that we could have pre made USB communications cutting out some of the hassle of that. After speaking with Dr. Richie and discussions with the rest of the group this option is there only if we need to actually do some programming on chip that is greater than necessary to make the hardware work		
PIC18F4550	Most hobbyists and a lot of projects on the net use this controller, which means we will have many examples to use or go from		
FT245BM	This is a chip that requires no extra programming on chip for the USB communication. It can be used in conjunction with the other chips or on its own in a speacial mode that allows for direct transfer of the info from the cpu to the ports on the chip….		
Conclusion:			
Needing to know the devices and gauges so we know what ouputs and ports need to support. We have been looking into different servo motors, Joe bought a small servo that seems to have the ratings needed to run off of USB power alone, that will be easily interfaceable with the FT245BM USB chip. That will solve some of the problems with some of the gauges as well as allow for feedback of the position, we could also gear these motors to get the full rotation that is necessary. The other gauges that need to be continuous are a little different and will need to use steppers if possible to find a mini stepper at the ratings that we need. I have been looking and have found a few but the ratings are right at the cutoff for the power consumption of the USB. This will take a little more research but it should be possible if not we could always use a separate power supply for each of the gauges, either way the gauges can be driven by the simple FT245BM chip and circuit that is necessary to make it work found on the FTDI Website....			

A.2 Flight Simulator Trade Study

Table A-1 Environmental Aspects

No.	Item/Description	Req. No.	FSX	X-Plane 9
1.	Inclusion of Majority of Airports Worldwide	1.B	Yes	Yes

No.	Item/Description	Req. No.	FSX	X-Plane 9
2.	Detailed Realistic Scenery	1.A	Yes	Yes[1]
2a.	Accurately detailed major cities and landmarks	1.A	Yes	No
3.	Realistic Weather Conditions	3.A	Yes	Yes
3a.	Real-World Weather	3.A	Yes	Yes
4.	AI Aircraft in the virtual world	3.E	Yes	No
5.	Deliver a constant 30 FPS	6	No[2]	No[2]

Notes:

1. X-Plane 9 does include the majority of airports worldwide including a few obscure airports that are not found in Microsoft Flight Simulator, but otherwise each largely has the same facilities available. Microsoft Flight Simulator however does include a higher detail of scenery of individual airports by ensuring that beacons, buildings and fueling stations are located at each facility. X-Plane 9 has only the runways and taxiways in the scenery, lacking any structures – even at major airports such as KJFK or KMCO.
2. The retail versions of FSX and X-Plane do not include any guarantees for being able to reach 30 FPS. This requirement can be achieved by purchasing sufficient computer hardware and optimizing the setting of the software package. FSX will allow you to set a target frame rate in the display options, but this will not change the display settings to deliver the required rate. X-Plane 9 also allows you to set a target frame as well as ensuring that the target frame rate is reached by changing the graphics settings on the fly. In addition to this one can purchase a USB key that brings the software into FAA compliance at a price of $500 (if to be used for flight training).

Table A-2 Aircraft Modeling

No.	Item/Description	Req. No.	FSX	X-Plane 9
1.	Included 3D Model Generator	4.A	No[1]	Yes
2.	Ability to change aircraft parametric data on the fly	4.B	Yes[2]	Yes

Notes:

1. FSX requires the use of an outside modeling program such as 3ds Studio Max to generate a 3D model of the aircraft. This opens up to the possibility that the model could look one way and have the flight characteristics of an aircraft that does not resemble that particular design.
2. Requires editing the aircraft.cfg file in a text editor, but allows you to change all of the aircraft variables.

Table A-3 Entertainment Features

No.	Item/Description	Req. No.	FSX	X-Plane 9
1.	Detailed Crash Effects	3.B	No[1]	Yes[1]
2.	Multiplayer Support	5.A	Yes[2]	Yes[2]
3.	Aircraft Sounds	3.C	Yes	Yes
4.	Ability to create custom scenarios/missions	3.D	Yes	Yes
5.	Built-in Instructor Operator Station (IOS)	8	No[3]	Yes[3]

Notes:

1. By Default in both of the flight simulators, when the aircraft crashes or the airframe is overstressed due to physical factors, the flight ends with the aircraft stuck in the position that the either struck the ground or featured overstressed conditions. However, X-Plane allows for the removal of flight surfaces if the aircraft goes past over-speed and over-G thresholds as well as the flaps and gear doors when over-Vfe thresholds have been passed. [X-Plane]
2. Microsoft Flight Simulator utilizes the GameSpy matchmaking service for multiplayer sessions across the internet but also supports direct connections across computers on the same LAN. X-Plane has support for local networking built-in.
3. X-Plane features a built in IOS that can be projected to a secondary monitor or can be utilized across the network with a different computer running a separate copy of X-Plane. Microsoft Flight Simulator does not have this feature built in and would require an additional application to be developed for this functionality to exist.

Table A-4 Simulator to External Flight Instruments/Controls Communication

No.	Item/Description	Req. No.	FSX	X-Plane 9
1.	Protocol/API to interface with flight simulator software	2.A	Yes[1]	Yes[1]

Notes:

1. FSX allows for two methods of interfacing with simulated flight controls and instruments: the SimConnect API and the legacy FSUIPC interface. X-Plane 9.4 utilizes plug-ins based on .dll files to communicate between the software and other applications and external instruments/controls.

Summary:

While Microsoft Flight Simulator X wins in regards to the default scenery included with the software and the number of resources available on the internet it is also unfortunately no longer being developed by Microsoft with no time frame for when a new version would be released, if ever. X-Plane 9 however released version 9.40 recently with no indication that development will stop soon. In addition, X-Plane models the aircraft more realistically, and includes the model

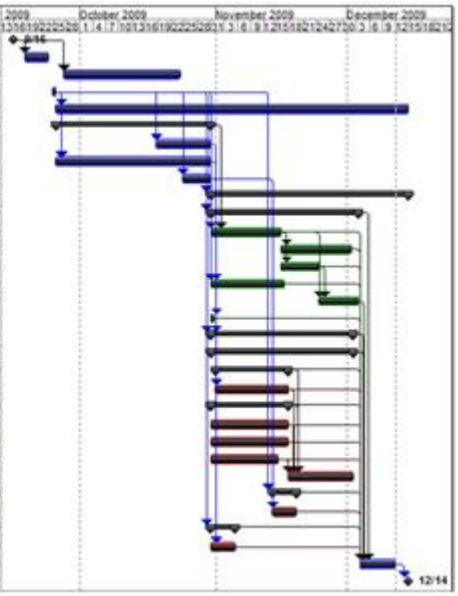
generator to develop an airframe to fly in the software compared to FSX which requires the use of an expensive 3rd party 3D modeling package. X-Plane also includes a few more effects in-terms of crashes, but lacks detailed scenery. Any areas that would need detailed scenery would need to be modeled or purchased as a add-on from a 3rd party developer. Finally there is the aspect if this simulator were to ever be used for ground based training, the only option to allow for this would be to use X-Plane after purchase of a $500 USB license key which unlocks the ability for it to be FAA Certified.

Recommendation:
Use Laminar Research X-Plane 9.4 for the graphics software to power the G700S Cockpit flight simulator.

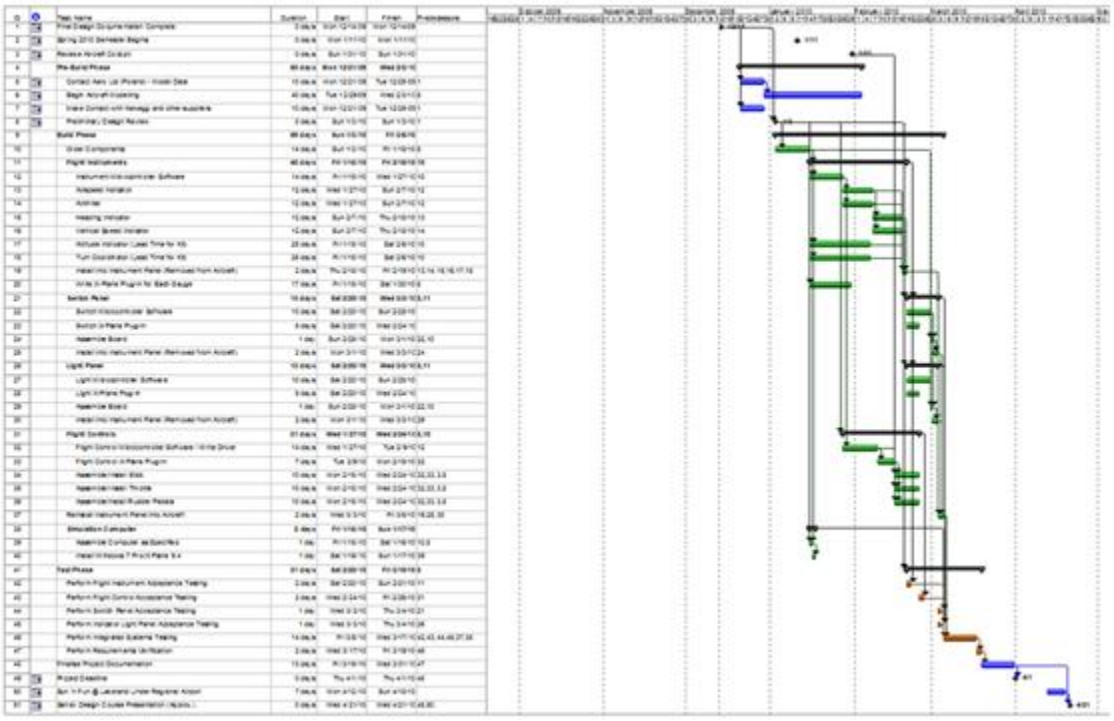
Appendix B: Project Schedules and Fall Semester Monthly Status Reports

B.1 Fall Semester Project Schedule

ID	Task Name	Duration	Start	Finish
1	Project Start	0 days	Wed 9/16/09	Wed 9/16/09
2	Initial Project Identification Document	6 days	Fri 9/18/09	Wed 9/23/09
3	Initial Project Research	30 days	Sun 9/27/09	Fri 10/23/09
4	Meeting With Sponsor	1 day	Fri 9/25/09	Fri 9/25/09
5	Design Documentation	90.13 days	Fri 9/25/09	Mon 12/14/09
6	**Conduct Trade Studies and Initial Research**	**39.5 days**	**Fri 9/25/09**	**Fri 10/30/09**
7	MSFS vs X-Plane	14 days	Sun 10/18/09	Fri 10/30/09
8	Instrument Interface	39.5 days	Fri 9/25/09	Fri 10/30/09
9	Computing Power	7 days	Sat 10/24/09	Fri 10/30/09
10	**Design**	**50.63 days**	**Sat 10/31/09**	**Mon 12/14/09**
11	**Software**	**38 days**	**Sat 10/31/09**	**Thu 12/3/09**
12	USB Communication	18 days	Sat 10/31/09	Sun 11/15/09
13	Servo Control Software	18 days	Mon 11/16/09	Tue 12/1/09
14	Cockpit Communications	10 days	Mon 11/16/09	Tue 11/24/09
15	FSX/X-Plane Plugin	19 days	Sat 10/31/09	Mon 11/16/09
16	IG Communications	10 days	Tue 11/24/09	Thu 12/3/09
17	Environment Setup (TBD)	1 day	Sat 10/31/09	Sat 10/31/09
18	**Hardware**	**36.13 days**	**Sat 10/31/09**	**Wed 12/2/09**
19	**Cockpit**	**36.13 days**	**Sat 10/31/09**	**Wed 12/2/09**
20	**Cockpit Controller**	**19 days**	**Sun 11/1/09**	**Tue 11/17/09**
21	Servo Motor Interface to Instrument Gauges	19 days	Sun 11/1/09	Tue 11/17/09
22	**Aircraft Control**	**20 days**	**Sat 10/31/09**	**Tue 11/17/09**
23	Joystick	20 days	Sat 10/31/09	Tue 11/17/09
24	Pedals	20 days	Sat 10/31/09	Tue 11/17/09
25	Throttle and other switches	17 days	Sat 10/31/09	Sun 11/15/09
26	Power Supply	16 days	Tue 11/17/09	Wed 12/2/09
27	**Simulation Computer**	**6 days**	**Sat 11/14/09**	**Thu 11/19/09**
28	Hardware Selection	6 days	Sat 11/14/09	Thu 11/19/09
29	**Instructor Operator Station**	**6 days**	**Sat 10/31/09**	**Thu 11/5/09**
30	Hardware Selection	6 days	Sat 10/31/09	Thu 11/5/09
31	Finalise and Format Initial Design Documentation	9 days	Fri 12/4/09	Fri 12/11/09
32	Design Documention Completed/End of Fall Semester	0 days	Mon 12/14/09	Mon 12/14/09



B.2 Spring Semester Project Schedule



B.3 October – Monthly Status Report

Period Covered:
1 Oct. 2009 – 31 Oct. 2009

Project Progression:
Upon reaching October 31 the trade study and requirements development phases have been completed. Trade studies are being presented at the first meeting of November along with our formal recommendations for design. In addition, the design phase has begun with USB communication and microcontroller interface being worked on.

Project Expenditures:

- Project Funds: $0
- Personal Funds: ~$50
    - Purchased a copy of X-Plane, servo motor and USB communications chip for evaluation.

Project Files Delivered:
FSX vs. X-Plane Trade Study
Microcontroller/USB Implementation Trade Study
Hardware Trade Study

Project Items to be Completed:

- Design of Flight Instruments
    - Microcontrollers/USB interface
    - Servo Motors
    - Required software on simulator PC
- Design of Flight Controls
    - USB interface
    - Throttle
    - Yoke
    - Pedals
- Other Electrical Design
    - Lights/Switches
    - Power Supply
- Design of Aircraft Model – Need Parametric Data
- Mounting Design for Monitors and Computer Hardware

Design Documentation

B.4 November – Monthly Status Report

Period Covered:
1 Nov. 2009 – 30 Nov. 2009

Project Progression:

Upon reaching November 30, 2009 we have completed major areas of the project design. The majority of the flight instruments design has been completed, although we are still attempting to contact simkits in regards to a discount on their pre-built gauges. Part selection for all of the major components, including the computer has been completed and has been rolled into our projected budget. At this time we are working on pulling together our project documentation and taking care of the remaining design tasks. We have also successfully tested the FTDI chipset that we intend to use for the aircraft gauges.

Project Expenditures:

- Project Funds: $0
- Personal Funds (this month): $0
- Personal Funds (project total): ~$50

Project Files Delivered:

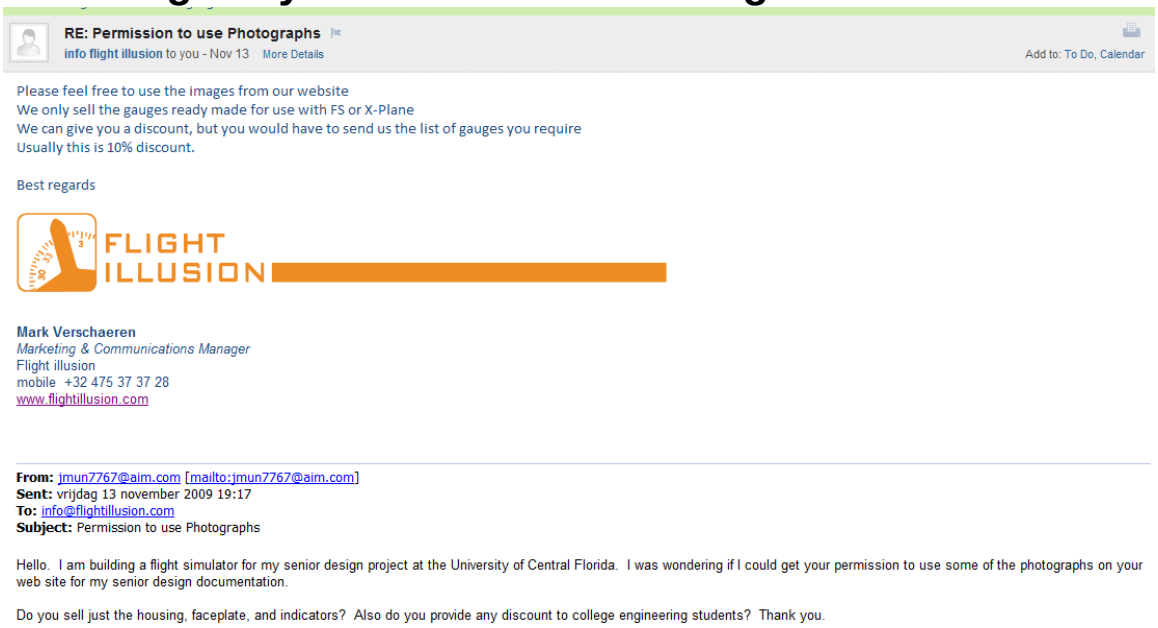
Budget

Project Status:

- Design of Flight Instruments
  - Microcontrollers/USB interface (Complete)
  - Servo Motors(Complete)
  - Required software on simulator PC (Complete)
  - Mechanical design (In Progress)
- Design of Flight Controls
  - USB interface (Complete)
  - Throttle (Complete)
  - Yoke (Complete)
  - Pedals (In Progress)
- Other Electrical Design
  - Lights/Switches (Complete)
  - Power Supply (Complete)
- Design of Aircraft Model – Have aircraft manual, other sources of performance data?
- Mounting Design for Monitors and Computer Hardware (In Progress)
- Design Documentation (In Progress)

Appendix C: Permissions to use Protected Materials

C.1 Images by Mark Verschaeren/Flight Illusion



RE: Permission to use Photographs
info flight illusion to you - Nov 13 More Details
Add to: To Do, Calendar

Please feel free to use the images from our website
We only sell the gauges ready made for use with FS or X-Plane
We can give you a discount, but you would have to send us the list of gauges you require
Usually this is 10% discount.

Best regards

FLIGHT ILLUSION

Mark Verschaeren
Marketing & Communications Manager
Flight illusion
mobile +32 475 37 37 28
www.flightillusion.com

From: jmun7767@aim.com [mailto:jmun7767@aim.com]
Sent: vrijdag 13 november 2009 19:17
To: info@flightillusion.com
Subject: Permission to use Photographs

Hello. I am building a flight simulator for my senior design project at the University of Central Florida. I was wondering if I could get your permission to use some of the photographs on your web site for my senior design documentation.

Do you sell just the housing, faceplate, and indicators? Also do you provide any discount to college engineering students? Thank you.

Site: http://www.flightillusion.com
This reference is used with permission for the following figures:

A) Figure 3-28

C.2 Information from Bob Miller

Reply: OTA is NOT copyrighted. Please pass along anything you like. Credit back to OTA would be appreciated.
-- Bob Miller

Site: http://overtheairwaves.com/vol3-46.jpg
This reference is used with permission for the following figures:

A) Figure 3-35

C.3 Wikipedia

Images taken from Wikipedia fall into three categories: Licensed under the GNU Free documentation License (denoted with a *), A work of a Federal Agency of the United States Government covered by Title 17, Chapter 1, Section 105 of the US Code (denoted by a +), or with no copyright claimed by the author (denoted by a ^).
For the following figures:

A) Fig. 3-11^ http://en.wikipedia.org/wiki/File:Six_flight_instruments.JPG
B) Fig. 3-17* http://en.wikipedia.org/wiki/File:3-Pointer_Altimeter.svg
C) Fig. 3-19+ http://en.wikipedia.org/wiki/File:Sens_alt_components.PNG
D) Fig. 3-20* http://en.wikipedia.org/wiki/File:True_airspeed_indicator-FAA.SVG

E) Fig. 3-22+ http://en.wikipedia.org/wiki/File:ASI-operation-FAA.png
F) Fig. 3-23^ http://en.wikipedia.org/wiki/File:R22-VSI.jpg
G) Fig. 3-25+ http://en.wikipedia.org/wiki/File:Faa_vertical_air_speed.JPG
H) Fig. 3-26* http://en.wikipedia.org/wiki/File:Attitude_indicator_level_flight.svg
I) Fig. 3-30* http://en.wikipedia.org/wiki/File:Turn_indicator.png
J) Fig. 3-32+ http://en.wikipedia.org/wiki/File:Turn_indicators.png
K) Fig. 3-33 * http://en.wikipedia.org/wiki/File:Heading_indicator.png