GoBosh G700S Flight Simulator

Senior Design I – Fall 2009 – Group 11

Christopher Dlugolinksi, Robert Gysi, Joseph Munera, Lewis Vail

Project Sponsor: Mr. Dave Kotick, Grizzly Aviation

12/14/2009

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.

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

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

This paper describes how each of the features listed above were researched and how they will be implemented, including a budget and a timeline to finishing the simulator for the Sun ‘n Fun airshow and general aviation conference that is to be held at the Lakeland Linder Regional Airport, in Lakeland, FL during the second full week of April. Because of this we have a much earlier deadline compare to other groups, for we must be done by April 1, 2010.

In order to make the simulator as real as possible we needed to pinpoint the parts of the activity of flying the GoBosh that were essential. This is where we gathered our requirements. We got them from our sponsor who is a flight instructor, and some of his fellow aviators. We also had requirements from the hardware and software interfaces that we will be dealing with. The requirements are listed in the requirements chapter in this paper. These requirements led us to our budget, which was originally given to us by the sponsor, and we needed to make that work doing research on many of the components that we were going to use. This is explained further in the budget chapter of this paper. Our budget was also affected by the timeline that we needed to deal with as well as the hardware interfaces that had to be designed. The design of the system is discussed in the design chapter. It lists all the reasons why we decided to create the system the way we are and how we are going to create it. Then we discuss the testing that we plan on doing in order to make the simulation as real as possible. The chapter on testing is very important to us cause it will show us how are planned project is coming together. We have a very tight schedule the next couple of months and we are ready for the challenges as you can see from the research that we did in our trade studies which are listed at the end of this document.

It is going to be a tough couple of months but in the end we will end with greater understanding of our engineering process and also greater knowledge on the art of simulation creation.

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 have been tasked on this project by Mr. Dave Kotick, a local flight instructor based out of the Orlando-Apopka Airport near Apopka, FL who is sponsoring this project through his business Grizzly Aviation.

As defined in our first meeting with Mr. Kotick, the initial purpose of this project is to produce a flight simulator that is not necessarily for flight training, but for demonstrations. In addition to training aspiring Light Sport Aircraft pilots, his business is also a representative of the US importer of the aircraft we are simulating: 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, 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 that is held in Lakeland, FL at the Lakeland Linder Regional Airport. This year the event is being held during the second week of April (4/13-4/18), and as a result he would like to have the simulator at the show. This is ultimately has become the deadline for our project and is where we will present our project to the aviation world.

While making it to Sun ‘n Fun is considered our ultimate objective for this project, we also have several side objectives as well that this project can also 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 is also designed to be taken to a variety of other shows in the future once handed over to Mr. Kotick after the Lakeland show. 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.

Another additional objective for this project is 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 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 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 requirement 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 to be chosen, the requirements for the aircraft model we are to develop for the flight simulator, and the microcontroller 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 is critical to pick the correct flight simulation software package. Currently, there are two competing simulators on the market available to end-users: Microsoft® Flight Simulator X (FSX) and Laminar Research® X-Plane 9.4. To the average end user, they are fairly similar applications, although for our purposes only one really stands out.

X-Plane 9.4 incorporates the most accurate methods of modeling an aircraft in virtual environment by actually taking the shape of the aircraft and model the aircraft through the use of blade element theory. This technique means that the software sections the aircraft model into multiple small “blades” to calculate the forces on these points. This gives a realistic physics model of the aircraft, which means if you model a solid cube with no aerodynamic properties, all it is going to do is sit on the ground. Microsoft FSX takes a different approach and instead of breaking down the aircraft into sections and then modeling it in a physics engine, it receives all of its properties through a configuration file, meaning 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 for down the road 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 the section below.

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	No

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 once 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.

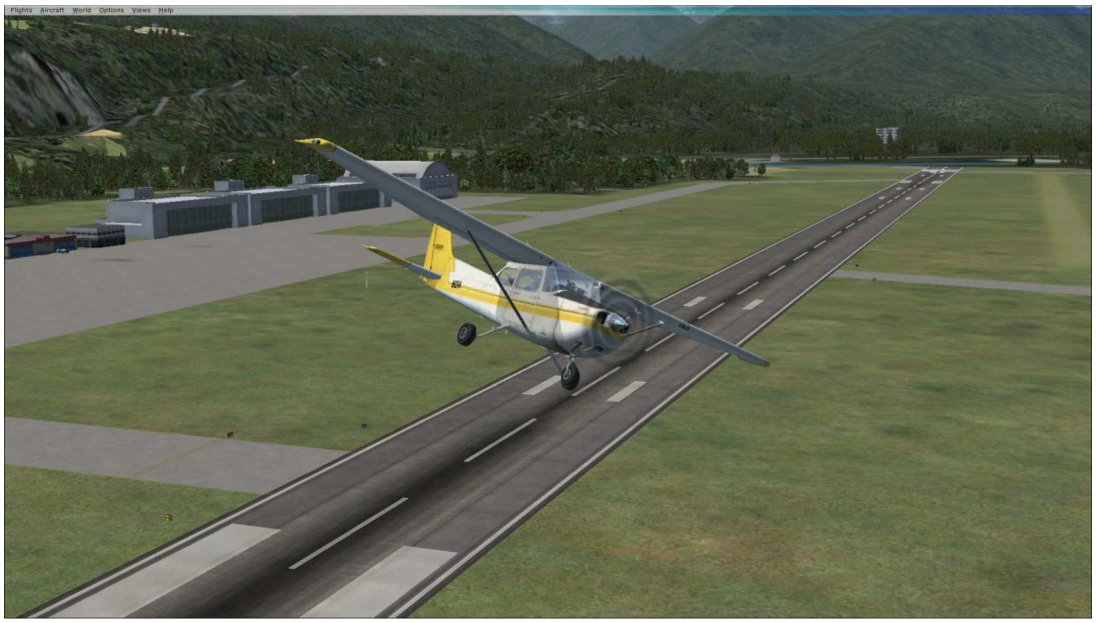


Figure 2-1. Flying a Cessna C172SP over Innsbruck, Austria in Microsoft FSX

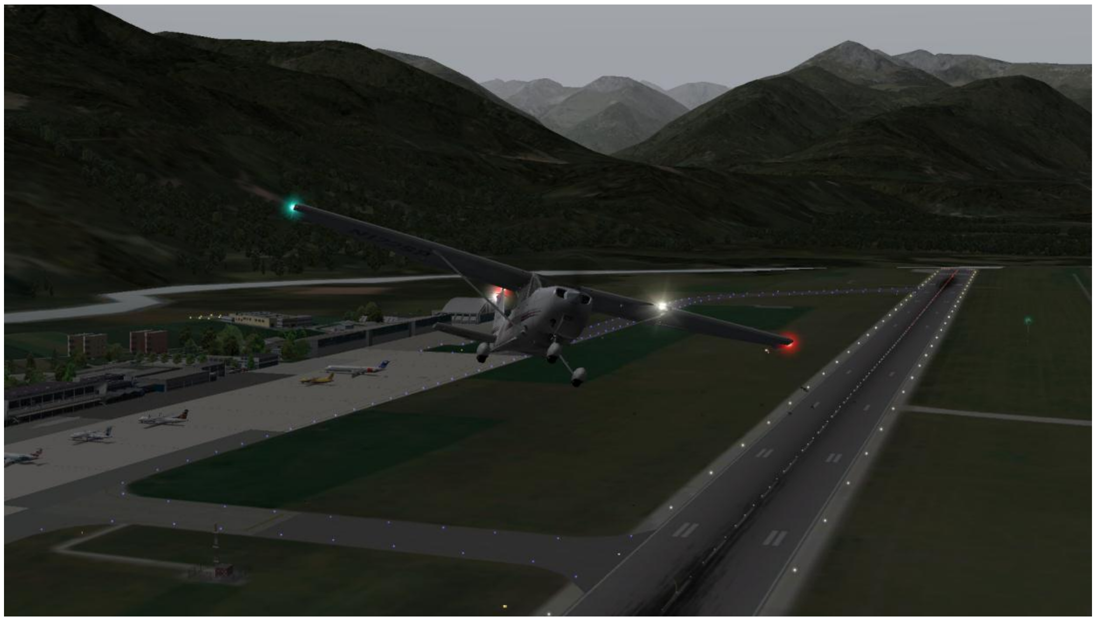


Figure 2-2. Flying a Cessna C172 over Innsbruck, Austria in X-Plane 9.4C

Comparing the two images preceding this paragraph you can notice some interesting differences between the two simulators. However, before We start I should make it clear that in X-Plane 9, the default airport is Innsbruck Kranebitten Airport (LOWI), and is therefore has a higher detailed scenery than many of the airports in the game, whereas in Microsoft Flight simulator it is just another airport in a list of thousands. One thing that is noticeably different between the two simulators is that smoothness of the rendering of the aircraft. Both were ran on the same machine and resolution, but the one in FSX is slightly jagged. Also, while not able to tell form this picture, is that 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. Speaking of the Earth, the 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.

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. X-Plane does address by allowing setting a target frame rate also, 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.

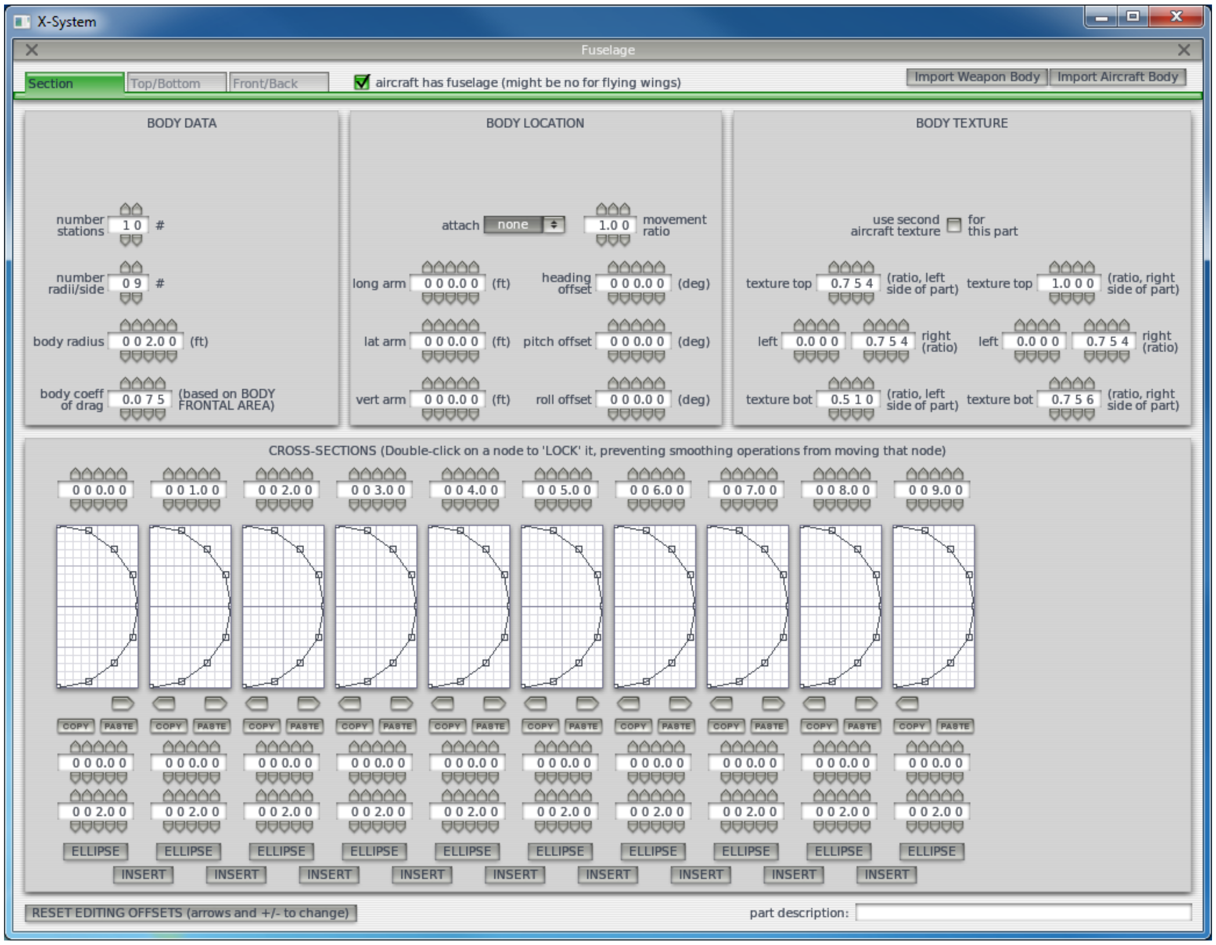
Another point that needs to be addressed is the inclusion of computer controlled or AI aircraft that exist in the simulated environment. Microsoft Flight Simulator has this feature built-in and turned on automatically. These AI-based aircraft fly normal routes and will land, takeoff and even make contact with the AI-based Air Traffic Control. Additionally they are not limited to one type, almost every sigle 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 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 model will need to be developed. 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 where we cover modeling the aircraft using the tools in X-Plane in addition to parametric data we currently have from the manufacturer.



☑ aircraft has fuselage (might be no for flying wings)

number stations 10 #
number radii/side 09 #
body radius 002.00 (ft)
body coeff of drag 0.075 (based on BODY FRONTAL AREA)

attach none
movement ratio 1.00
long arm 000.00 (ft)
heading offset 000.00 (deg)
lat arm 000.00 (ft)
pitch offset 000.00 (deg)
vert arm 000.00 (ft)
roll offset 000.00 (deg)

use second aircraft texture ☐ for this part
texture top 0.754 (ratio, left side of part)
texture top 1.000 (ratio, right side of part)
left 0.000 0.754 right (ratio)
left 0.000 0.754 right (ratio)
texture bot 0.510 (ratio, left side of part)
texture bot 0.756 (ratio, right side of part)

CROSS-SECTIONS (Double-click on a node to 'LOCK' it, preventing smoothing operations from moving that node)

000.00	001.00	002.00	003.00	004.00	005.00	006.00	007.00	008.00	009.00
000.00	000.00	000.00	000.00	000.00	000.00	000.00	000.00	000.00	000.00
002.00	002.00	002.00	002.00	002.00	002.00	002.00	002.00	002.00	002.00

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 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 (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.

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 simulation state and pull out information ranging to which lights are one, 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.

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

Req #	Task	Summary
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 will be 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. Some of the needed info is still not in our possession and will need to be added later, it is not allowed for us to publish the info that we will have unless we own the actual aircraft. Yet, from some of the info we could get from the sponsor and from the brochure that was given to us we can get some of the info needed to create a model of the plane we are trying to simulate.

Getting the information into the actual X-Plane simulation we can use their program Plane-Maker. This program is bundled with the game and has an interface that allows you to input the needed info to get your type of plane up and in the air. As for the actual model of the plane you need to have an OBJ file, and not the known OBJ file that can be output from most modeling software. This OBJ file needs to be of their proprietary type, as explained below.

[1]We will need a 3-D editor capable of saving an object in the X-Plane OBJ format. Note: This is not the same as the Alias OBJ format.
The following file formats can be used to create X-Plane OBJ files:

- 3DS (Autodesk 3D Studio)
- DXF (AutoCAD)
- OBJ (Alias Wavefront)
- AC (AC3D)
- MD2 (Quake model)
- WRL (VRML)
- LWO (LightWave)

[1] (2009, Dec.). Creating Airplanes and ObjectFiles for X-Plane [Online]. Available: http://wiki.x-plane.com/Creating_Airplanes_and_Object_Files_for_X-Plane

- TXT (Milkshape)

With each of these file extensions, users need to convert the objects to ones usable by X-Plane. This is often done by opening the file in AC3D, then using the X-Plane plug-in to export the file as an X-Plane OBJ. In addition, after an object has been created in any of these applications, a tool named ObjConverter can be utilized to convert the file into the appropriate X-Plane OBJ format. This tool is free to download as part of a scenery development pack.[4]

Since the proprietary type may need the purchase of new expensive software, we may need to stick with just the non-physical properties. Although depending on the time it would take to learn we could use Blender as it has a plug-in capable of converting to X-Plane OBJ format, but that will be a decision made once some of the software and hardware are integrated and we know the amount of time that will be available to do this.

The Plane-Maker tool that ships with X-Plane asks for essentially every single bit of dimension and performance data in order to create an accurate 3D model without the use of a 3rd party application. At this moment we currently only have some of the performance data of the aircraft we are simulating, along with basic dimensions. A summary of the data that we currently have to assist with our preliminary modeling is located in Table 2-6 below. This includes information from the GoBosh Aviation sales website in addition to the Airplane Flight Manual for the aircraft as well. However, this is not enough information and in order to model the aircraft we will be receiving the full data from the aircraft manufacturer in the January/February timeframe.

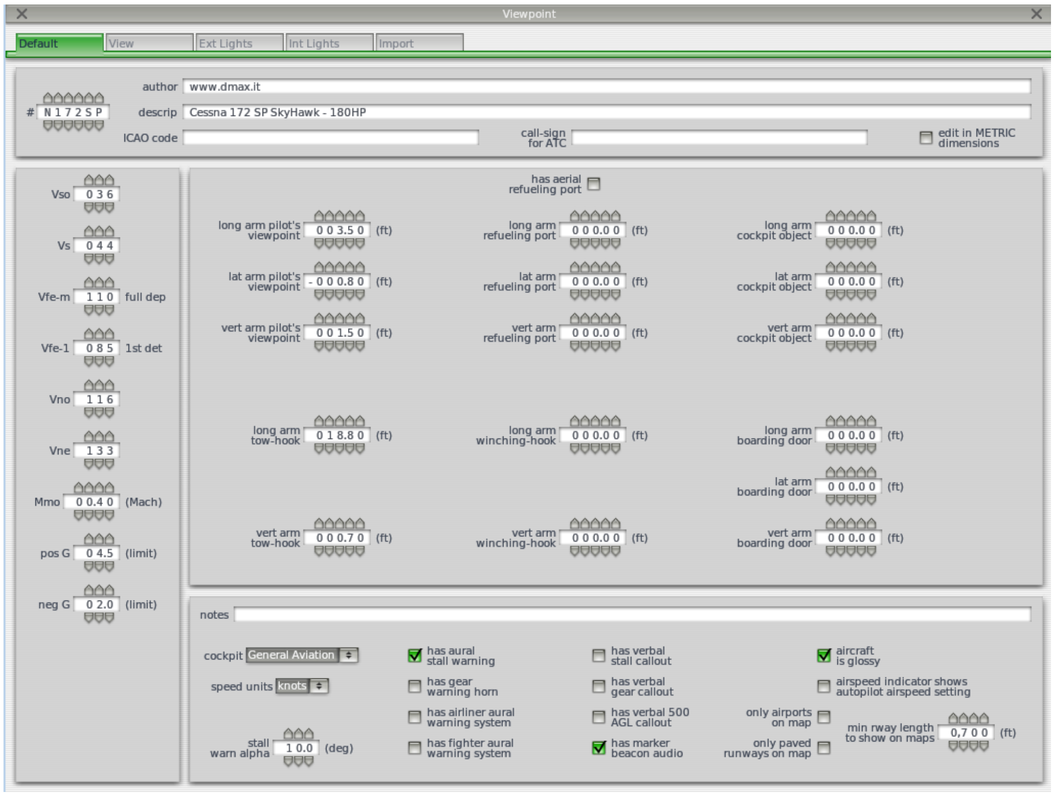
Table 2-6 Summarized Aircraft Data[2,3]

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

[2] *(2009, Nov.). GoBosh G700S Specs [Online]. Available: http://www.gobosh.aero/G700.cfm*
[3] *(2009 Dec.). Airplane Flight Manual Aero AT-4 Light Sport Airplane [Online]. Available: http://www.ussportaircraft.com/uploads/Gobosh_POH_1_.pdf*

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

The figure below shows some of the information that is necessary to fill in to get our aircraft to function correctly. Some of the information is openly available but a lot of the information needs to come from the planes manual in addition to the need to get actual test data of the aircraft. With actual test data from the manufacture we could model the aircraft with even more detail and accuracy compared to the even expanded information we would get from the pilots manual. All together this means that the creation of the model will be a large task on its own. The amount of data that needs to be entered is enormous. The knowledge of aircraft in our group is low and the language of the GUI is not as user friendly as they would like you to think. There are tool tips in the program to guide the user through creation of the model, in addition how to guides available on X-Plane community sites on the internet.



# N172SP
author: www.dmax.it
descrip: Cessna 172 SP SkyHawk - 180HP
ICAO code:
call-sign for ATC:
☐ edit in METRIC dimensions

Vso 036
Vs 044
Vfe-m 110 full dep
Vfe-1 085 1st det
Vno 116
Vne 133
Mmo 00.40 (Mach)
pos G 04.5 (limit)
neg G 02.0 (limit)

☐ has aerial refueling port

long arm pilot's viewpoint 003.50 (ft)
lat arm pilot's viewpoint -000.80 (ft)
vert arm pilot's viewpoint 001.50 (ft)
long arm refueling port 000.00 (ft)
lat arm refueling port 000.00 (ft)
vert arm refueling port 000.00 (ft)
long arm cockpit object 000.00 (ft)
lat arm cockpit object 000.00 (ft)
vert arm cockpit object 000.00 (ft)
long arm tow-hook 018.80 (ft)
vert arm tow-hook 000.70 (ft)
long arm winching-hook 000.00 (ft)
vert arm winching-hook 000.00 (ft)
long arm boarding door 000.00 (ft)
lat arm boarding door 000.00 (ft)
vert arm boarding door 000.00 (ft)

notes:
cockpit: General Aviation
speed units: knots
stall warn alpha 10.0 (deg)
✓ has aural stall warning
☐ has gear warning horn
☐ has airliner aural warning system
☐ has fighter aural warning system
☐ has verbal stall callout
☐ has verbal gear callout
☐ has verbal 500 AGL callout
✓ has marker beacon audio
✓ aircraft is glossy
☐ airspeed indicator shows autopilot airspeed setting
☐ only airports on map
☐ only paved runways on map
min rway length to show on maps 0,700 (ft)

Figure 2-4. Plane-Maker viewpoint setup

Once the model has all of its parameters input then comes the part of the modeling that will need 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. It is important to note that if the exact dimensions are not known for this stage of model development, the editor allows you approximate by changing the shape with your mouse. Figure 2-3 Below shows this ability.

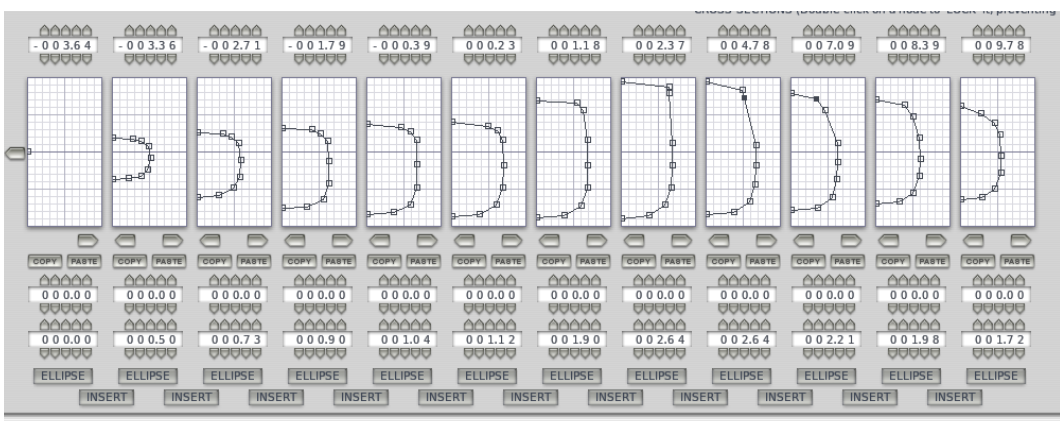
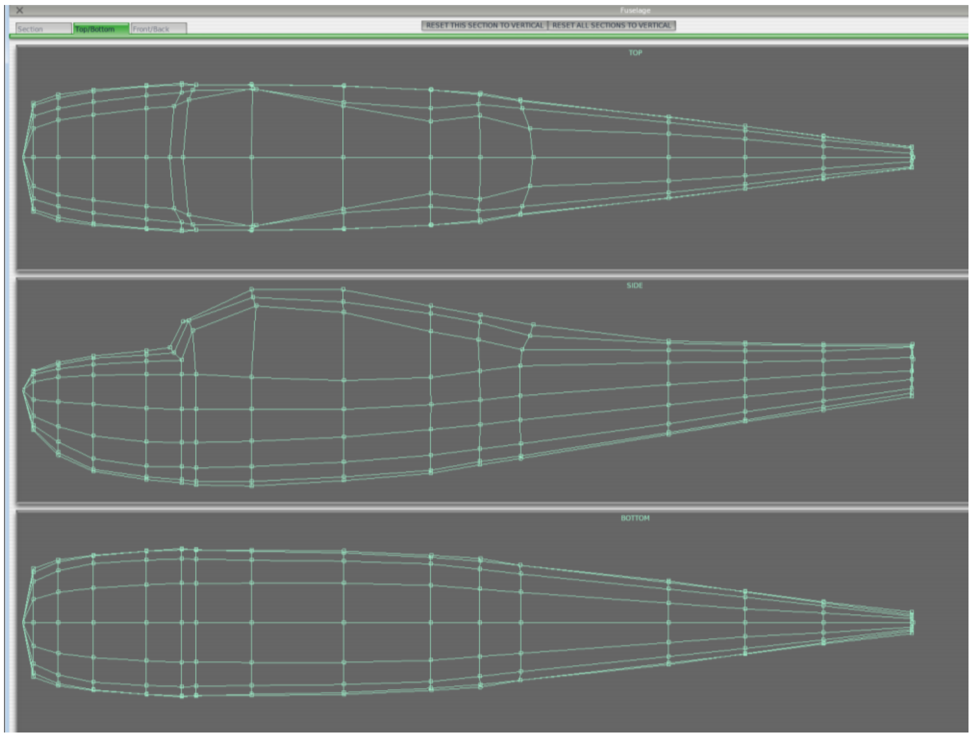


Figure 2-5 Plane Maker Fuselage Editor

The fuselage building allows you to edit your aircraft at different spots along the length of the plane. You can basically get the measurements at each of the set spots on the aircraft and input them in here. You can also just move the points around in the editor until you get the correct look as shown below.


Fuselage
Section
Top/Bottom
Front/Back
RESET THIS SECTION TO VERTICAL
RESET ALL SECTIONS TO VERTICAL
TOP
SIDE
BOTTOM
Figure 2-6 Wireframe representation of a Fuselage in development.

As you can see from the above screen shots the modeling of the aircraft will be a major part of this project. In order for the simulation to actually model the real aircraft the model needs to be almost exactly like the real aircraft.

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. This if our sponsor desires a cockpit model in addition to our exterior model it would be relatively straight forward to create. 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. On the following page is a figure that shows 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 should help with the simulation should we ever need to try and troubleshoot one 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 develop.

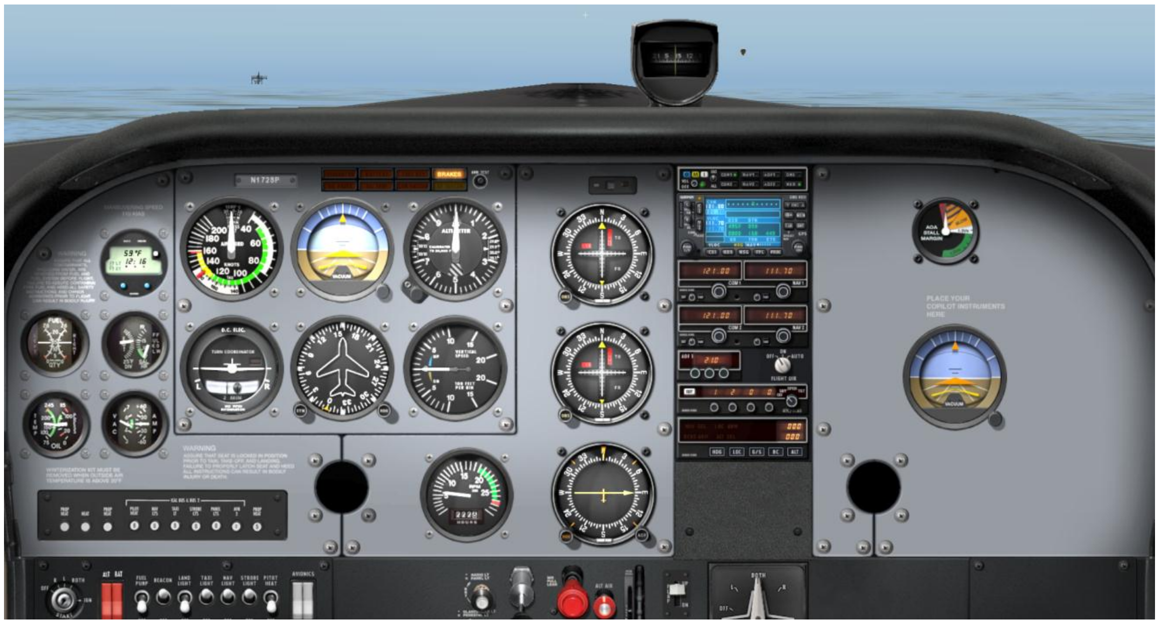


Figure 2-7. Example cockpit showing additional gauges.

2.2.1.3 Microcontroller Software Requirements

The requirements for the microcontroller software are really based on the hardware that will be used. We are using stepper motors and servo motors. The stepper motors need to be updated at a rate that will make the movements look smooth. Depending on the type of stepper motor (we are hoping to use at least 200 steps) there are different number of steps that are taken for a total rotation of the motor. With the 200 steps we can get a 1.8 degree resolution this may not be good enough for smooth rotation. We will need to half step to overcome this and that requires that the microcontroller be able to get information from the host computer and update the motor twice as fast. Twice as fast meaning that we need to update our motor at a certain rate (we are trying to stick with a 33ms rate for ease of use with the X-Plane program) that will keep the motor whether it is the stepper or the servo in the correct position and still look smooth to the observer.

The servo motors need to be updated by a signal for a certain period of time in order to move it. The microcontroller needs to be able to take the value of the gauge and turn this into a time period to be pulsed. The ability to update the servo with pulses of between 1ms and 2ms depending on our desired position will be handled by the USB sending a message to the microcontroller and then the microcontroller will use an A/D converter to setup a 555 timer circuit that can pulse the stepper motor. The circuit still needs to be designed but the theory of how it works is talked about later. The main requirement here is that the data get from the simulation to the USB port and down to the circuitry in less than 20ms in order to keep the servo motor in the correct position. There is more discussion of the actual controlling of the servo in the microcontroller requirements section of this paper.

The software that will be running either X-Plane or FSX will tell us our capabilities. Both packages 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 update faster than this but it would just be the same value sent and that would just cause unneeded traffic. This could slow the code and could take away from other threads that may be operating. So at this rate we need to have a speed of : 1 sec / 30 = 33 ms and we also need to send enough information to update each of the gauges and any of the switches we need to implement. 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 should be 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 that we are going to use will set the type of software requirements that we need also. It will determine the clock frequency of the microcontroller in order to keep up with the frame rate. All microcontrollers today will have no problem keeping up with our needed update rate.

2.2.2 Hardware Requirements

The hardware requirements development includes all of our hardware and 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.

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 need to decide on what the important characteristics of the gauges will be, below is a list of requirements that need to be met for each of the gauges.

Table 2-8 Established Requirements

Req #	Task	Summary
M1	-	USB Controlled
M2	-	20ms refresh rate (minimum)
M3	-	Use less than 5V
M4	-	Minimum 8 I/O Pins for external communications
M5	-	Fit inside of a 2.5” diameter tube
M6	-	Low Cost Microcontroller
M7	-	As self-contained as possible

The gauges that we are trying to duplicate are the six-pack that is located slightly to the left in the photo below in Figure 1. From the requirements we need to make these gauges look and act just like the real gauges would in the actual Bosh aircraft. They also needed to fit in to our budget which after the purchase of a computer with a capable graphics card and the monitors is a very small amount. There of course are the pre-made gauges that have been discussed, but those are expensive, so we need to think of other ways. What we came up with is the handmade gauges discussed in this paper. They are controlled by either a stepper motor or a servo motor depending on what the gauge needs to be implemented. In order to do this we needed to come up with a way to power these gauges as well as control them.

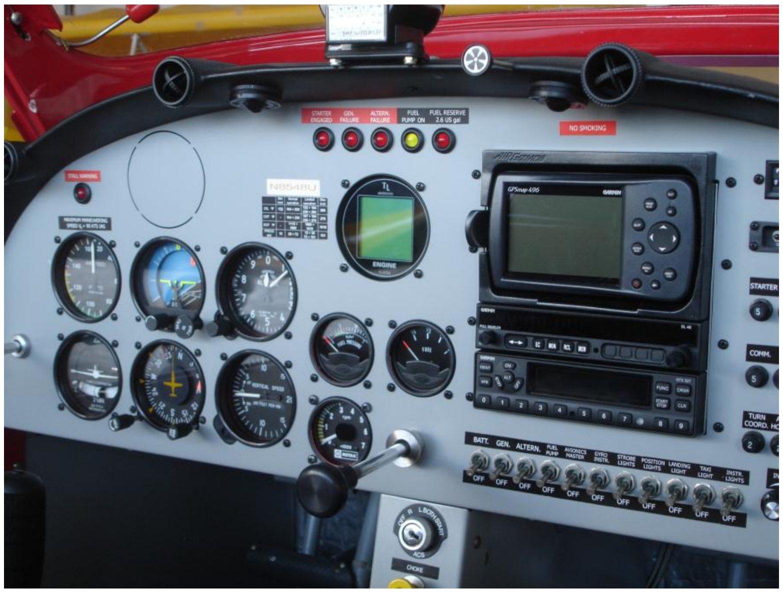


Figure 2-8. A photo of the cockpit in a GoBosh aircraft *Photo by Robert Gysi*

We were given a requirement that everything needs to be connected via USB. With this constraint we will have to find a way to control the motors (both servo and stepper) with a microcontroller of some sort that is able to use the USB protocol. Along with the restriction of speaking USB protocol it needs to fit into the power specs of USB so we either connect an outside power source or find a way to keep all the power ratings under that of the USB spec. USB provides 5 +/- 0.25 volts to power an otherwise unpowered device or charge a battery in a self-powered device[4]. A USB controller or hub is required to power one unit load (100 mA, a low power load). It can optionally power up to 5 unit loads (500 mA, a high power load). This means that we need to keep our needs below 5 volts and 500mA, so devices needing more than 500mA or higher than 5 volts must provide their own power[5].

Looking into the different types of stepper and servo motors we have a wide range to choose from most of which are 12 volt and low amperage. The stepper motor is mostly used in robotics to add some sort of torque (strength) to the

[4] (2009, Nov.). USB as a power source [Online]. Available: http://www.girr.org/mac_stuff/usb_stuff.html
[5] (2009, Nov.). USB pinout and wiring [Online]. Available: http://pinouts.ru/Slots/USB_pinout.shtml

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 will in turn require more current from the power supply. Since we are limited to 500mA we need to pick a motor that can operate at the 500mA divided by the number of coils that could be on at once. This is required for micro stepping of the motor. Microstepping is typically used in applications that require accurate positioning and a fine resolution over a wide range of speeds[6]. 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.

Use of the servo motor to display some of the gauges is useful for gauges that do not need to rotate a full 360 degrees. The servo motor will need 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 will involve a 555 timer, and some sort of Digital to Analog converter. The 555 timer shall 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 chip we will need to get an 8 input D/A that can give out voltages with high resolution (2^8 = 256 values between 0 and 180 degrees).

The design and test phase of the project will determine if we need to add some sort of outside power source to help control the gauges, but we are first trying to put it together using no outside power source. 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 2-7 shows the view of the instrument panel from inside the cockpit. Ideally, this is what the inside of our final cockpit will look like. Unfortunately we do not have the time or resources to model all of these instruments. For the purposes of this document we have broken the instruments into three different categories: essential flight gauges, auxiliary flight gauges, and lights and switches.

[6] (2009, Nov.). Stepper Motors reference guide [Online]. Available: http://ams2000.com/stepping101.html

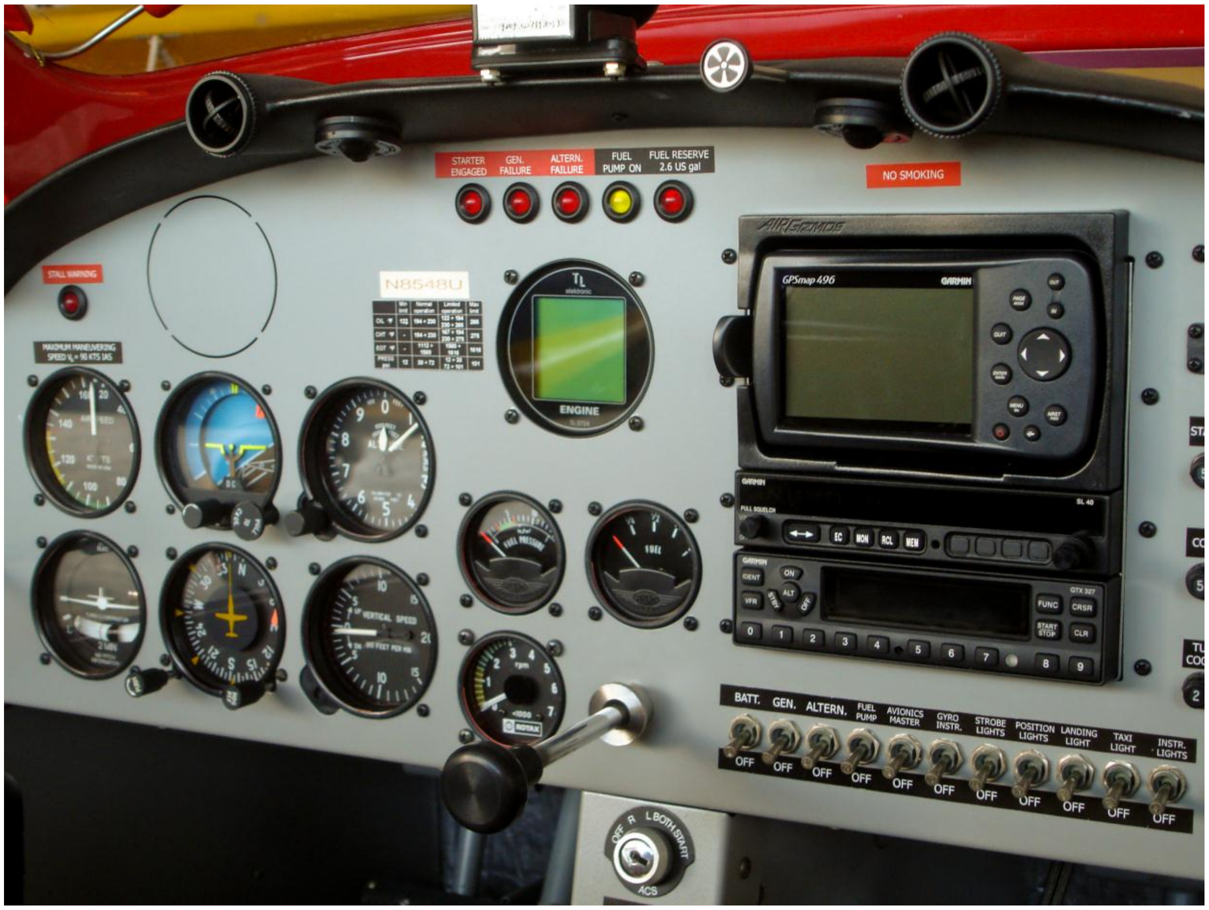


Figure 2-9: The interior view of the instrument panel. *Photo by Robert Gysi.*

The one category of instruments that was essential to properly simulate the aircraft was the standard six-pack of gauges. Figure 2-8 is an enlarged view of these gauges. 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. If we were not able to model any other instruments they would be enough to get the feel of flying a GoBosh. The following are some specific requirements for each gauge:

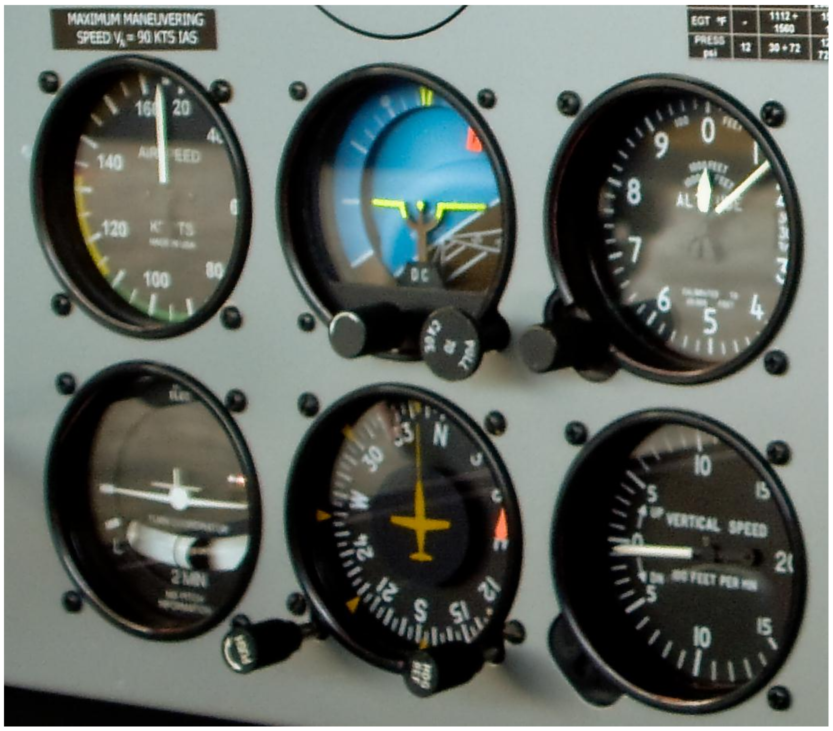


Figure 2-10: This is the standard six-pack of gauges in the GoBosh.
Photo by Robert Gysi.

The airspeed indicator (top left corner of figure 2-8) 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 a the artificial horizon (top middle of figure 2-8) 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 2-8) 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 2-8) 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 2-8. 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[7].

The heading indicator (bottom middle of figure 2-8) is required to turn all the way around (more than 360 degrees) just like the 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 2-8) 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 it’s 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 so and so our model will have to keep up with this. But because this gauge operates at a higher speed, choppiness and lack of precision is less of a concern, as it most likely will not be noticed.

Although the standard six-pack gauges supply the bare minimum to operate the simulation, we will also try to model the rest of the gauges as time permits. These include the floating compass, the fuel pressure gauge, the fuel gauge, and the tachometer. The latter three of these gauges are shown in figure 2-9.

Out of all the optional gauges the floating compass is the most usable to the pilot of our simulation. It performs mostly the same function as the heading indicator except it is a magnetic compass whereas the heading indicator is a gyroscopic gauge that must be manually calibrated before takeoff. Floating compasses come standard in many small aircrafts including the GoBosh that our simulation is based on. This addition would certainly add to the authenticity of our simulator.

[7] (2009, Dec.). Wikipedia Article: Turn Coordinator [Online]. Available: http://en.wikipedia.org/wiki/Turn_coordinator

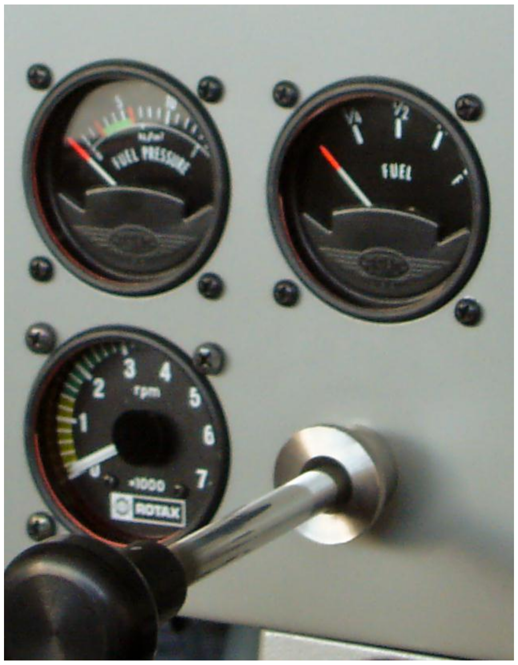


Figure 2-11: This is a close up of the fuel pressure gauge, the fuel gauge, and the tachometer in a real GoBosh (left to right top to bottom). *Photo by Robert Gysi.*

If we finished all the essential gauges and the floating compass and we still had time and resources for another gauge, the next important one would be the tachometer (bottom of figure 2-9). The information that this gauge displays is mostly worthless for the purposes of our simulator so it is most likely that this gauge will not be built. As always, it does add another level of realism to the overall cockpit but its purpose if built would be more along the lines of ascetics flight training. It would need a range of motion of about 270 degrees and be able to display up to 7000 RPMs.

The last two mechanical gauges that come standard on a GoBosh airplane are the fuel pressure gauge and the fuel gauge (top left and right of figure 2-9 respectively). Neither of these are very useful in our simulation as we do not expect to model mechanical problems that would require you to check the fuel pressure gauge nor do we expect pilots to fly long enough to deplete the fuel. The only reason to add these pieces would be to complete the look of the cockpit. It may be completely sufficient to just mount faceplates with fixed needles to accomplish this requirement.

Besides the gauges we also plan on modeling various warning lights and switches in the cockpit to further its look and functionality. To properly model these elements, they must look as similar to their real world counter parts as possible and they must be functional. This means the switches must toggle properly and feed their position into the simulation and the LED warning lights must light up when they receive the appropriate signal from the simulation to a luminance that is bright enough for the pilot to easily see. Figures 2-10 and 2-11 show the various warning lights and switches of the GoBosh cockpit.

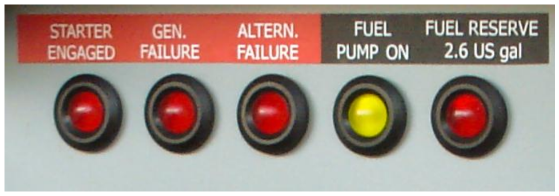


Figure 2-12: Warning lights of the GoBosh cockpit. *Photo by Robert Gysi*.

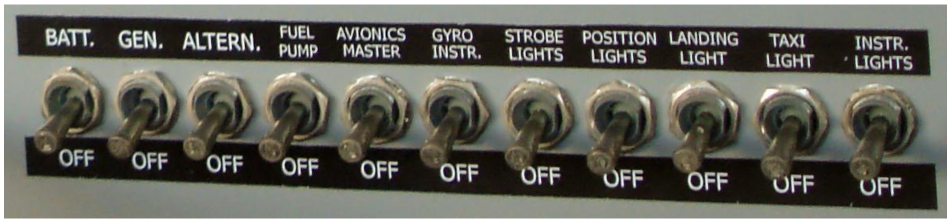


Figure 2-13: Switches of the GoBosh cockpit. *Photo by Robert Gysi.*

2.3 Project Budget

Our project sponsor has established a budget of $1500. After completing the pricing of our components we will probably be over $1500. In the event that the sponsor does not agree to pay for costs over $1500, the members of the group are prepared to meet the additional costs. However, since the majority of the budget for the project is set aside for the computer that the simulator will run on, when it comes time to purchase the components in the spring semester we expect prices to drop on components and where possible will purchase items on sale to minimize our budget. In addition we are also in negotiation with two manufactures of flight instrument kits to see if we can get a discount on the one or two instrument gauges we plan to purchase (Attitude Indicator and Turn Coordinator).

At this point in the design process we have completed our cost analysis and for the components desired our project will cost approximately $1,670.32. It is important note that while we have exceeded our limit of what our sponsor has initially agreed to funding us by $170.32, that does not mean we are over budget. In fact several group members are fine picking up any small cost overruns giving us at least $300 in additional towards our budget, should it be required. In addition, our project sponsor has given us the green light to attract additional sponsorship from various businesses.

Currently we have contacted several kit gauge manufactures and have requested sponsorship in the form of a discount towards the purchase of a gauge. We have heard back from Flight Illusions, but their products, even with a discount, are quite expensive. We have been also attempting to contact SimKits to see if we could arrange a sponsorship with their company as well. Outside of gauges we intend to persue a sponsorship with a computer parts vendor, possibly

NewEgg.com or ZipZoomFly.com. Contact these businesses will occur early in the spring semester when we are finalizing all of our orders for parts and resources. Additionally, in order to entice companies to sponsor a portion of our project, our primary sponsor has ok'd us to offer to put any company that provides a discount or parts to have their logo featured on the aircraft, even during the aviation conference and airshow in Lakeland. Table 2-9 shows our current spending projects.

Table 2-9 Projected Spending and Budget

Item	Part Number	Quantity Required	Unit Cost	Total Cost
X-Plane 9	N/A	1	$29.99	$29.99
Gateway 24" 5ms LCD Display	FHD2401	3	$189.99	$569.97
ATI/XFX ATI Radeon HD 5750 1GB	XFX HD-575X-ZNFC	2	$139.99	$279.98
ASUS Socket AM3 Motherboard	M4A785TD-V EVO	1	$99.99	$99.99
AMD Phenom II X2 Black Edition	HDZ550WFGIBOX	1	$99.99	$99.99
4GB DDR3 1066 RAM Kit	OCZ3G10664GK	1	$72.99	$72.99
Samsung 160GB HDD	HD161HJ	1	$38.99	$38.99
Lite On IDE DVD-Rom Drive	iHDP118-04	1	$17.99	$17.99
1000W ATX Power Supply	SP-585	1	$99.99	$99.99
Mid-Tower ATX Case - Black	313-06-C2228	1	$20.99	$20.99
Atmel USB Microcontroller	AT90USB1287	1	$15.05	$15.05
Slide Potentiometer - 10k, 100mm	N/A	5	$5.00	$25.00
Servo Motor		4	$10.00	$40.00
Simkits Attitude Indicator		1	$154.00	$154.00
FTDI USB Communication Board	FT245BL	3	$30.00	$90.00
USB AB Cord	N/A	3	$2.00	$6.00
Transistors	2N2222	10	$0.29	$2.90
Digital Potentiometer	AD5220	5	$1.00	$5.00
Buffer	CD4049	5	$0.30	$1.50
Notes: **Prices Subject to Change** **Where possible, combo packages of components (for computer) will be purchased to reduce cost**			Total	$1,670.32
			Project Budget	$1,500.00
			Difference	($170.32)

As mentioned before, it is important to note that while the total exceeds the project budget as determined by the sponsor, this does not mean we are necessarily over budget. We will be presenting this information, along with any solicitations we have sent out to other sponsors at the Preliminary Design Review to be held on January 3rd, 2010. It is at that point it may also be determined if our funding is increased. However, this is something that is to not be expected, as with any project and as a result we are continuously working to reduce the cost of the project or find secondary funding sources.

2.4 Project Timeline

As per the requirements of the project, the final product must be ready by April 1, 2010 in order to be on display at the Sun ‘n Fun air show and general aviation conference at the Lakeland Linder Regional Airport in Lakeland, Fl from April 13-18, 2010. Due to this deadline, which is before the established deadline for the course, we will have all work completed by April 1, 2010 so that testing and transport to the show can take place. The design phase will be completed at the conclusion of the fall semester (December 14, 2009). The build phase will begin at the start of the Spring semester on January 11, 2010. Pending parametric data of the aircraft we are simulating, modeling may start during the December-January break along with preliminary software development due to our early completion date. A simplified version of our project schedule is presented below, while the Gantt chart representing all of the tasks of the project can be found in the appendix.

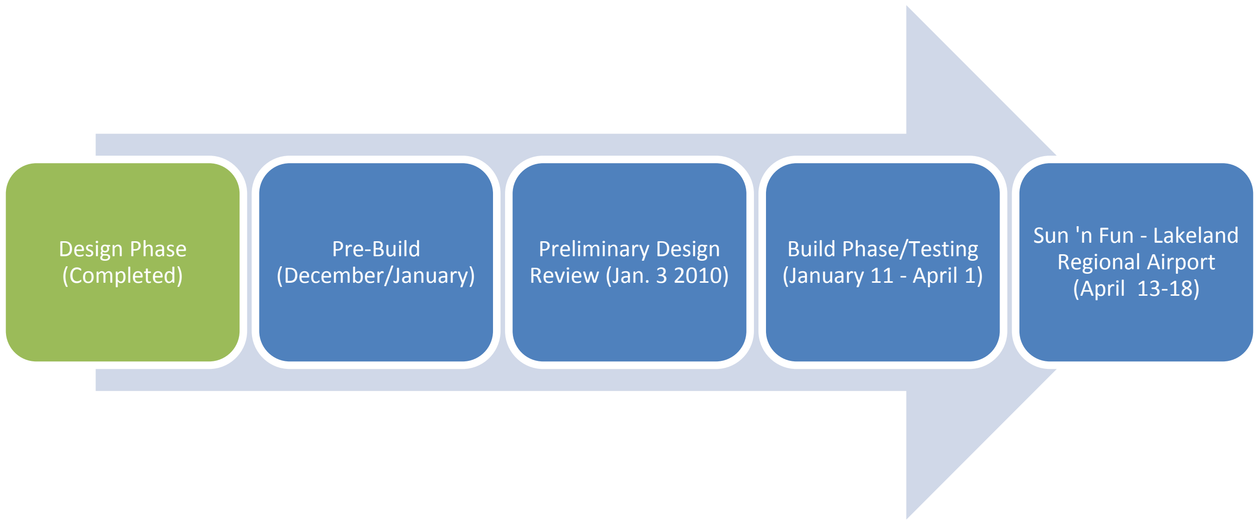

Design Phase (Completed)
Pre-Build (December/January)
Preliminary Design Review (Jan. 3 2010)
Build Phase/Testing (January 11 - April 1)
Sun 'n Fun - Lakeland Regional Airport (April 13-18)

Figure 2-14. Project Schedule

Also, a preliminary design review has been scheduled with our project sponsor for Sunday January, 3 2010. At this date we will present our design findings contained in this document and officially start the build phase with our initial funding. In addition, during a recent meeting it was discussed that we might be traveling to Sebring, Florida to the U.S. Sport Aviation Expo between January 21-24, 2010. We will not be demonstrating our project, as it will not be completed, but would serve as an opportunity to share with individuals our progress, answer questions, and invite them to return to Sun ‘n Fun in Lakeland to see the finished project. Participation in this show will be decided on in January.

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 need 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, that being an extra hurdle it has become a "if we have time OPTION". The system will still have the capability to have the IOS from the simulation software we chose(X-Plane).

The control of the cockpit and the gauges will be done through the USB chips that we have decided upon. The cockpit itself is divided into input devices and passive devices. The input devices are the yoke and pedals along with some of the switches that can be set. The passive devices are the lights that will be lit and the gauges. All of this is covered in the cockpit design section. The block diagram of the design is below.

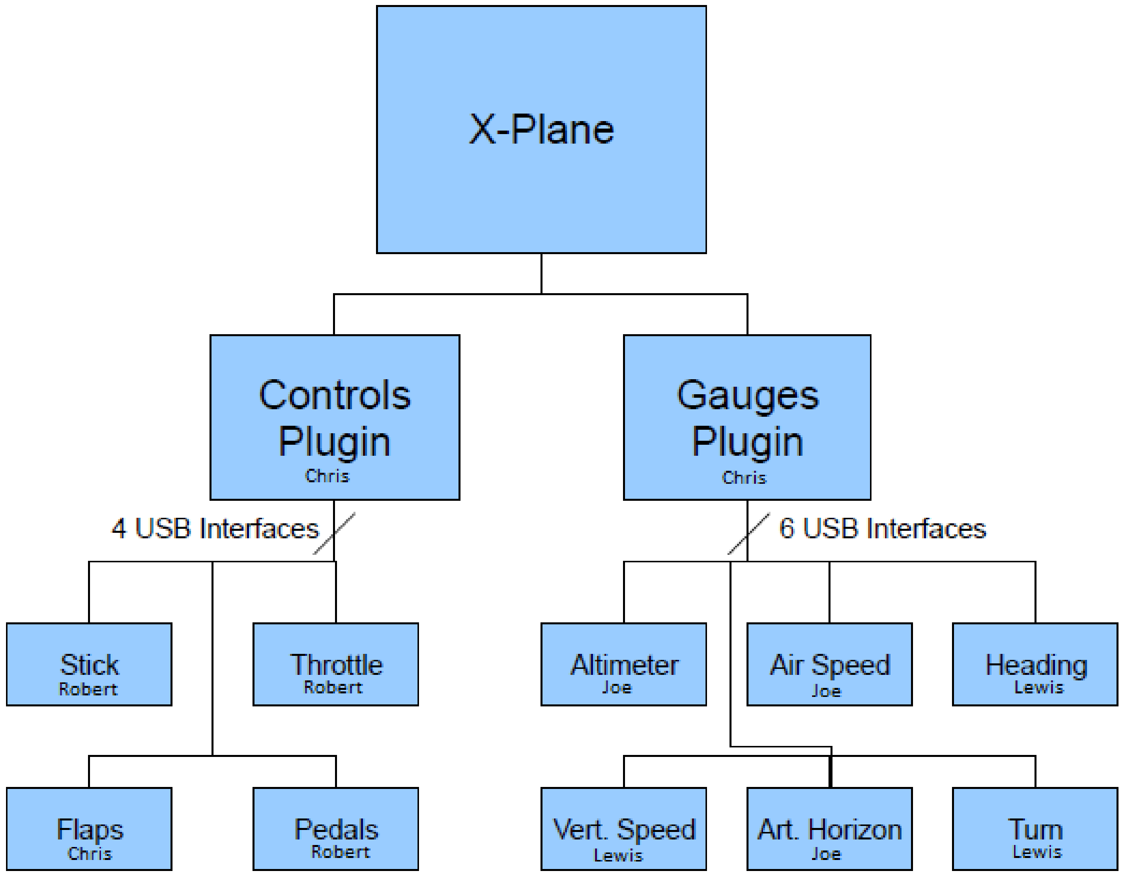


Figure 3-1 Block diagram representing cockpit interfaces and responsible parties.
Diagram by Lewis Vail

One of the major decisions that had to be 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 is the design and construction of the simulators aircraft flight instruments. The flight simulator will consist 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 must be an analog design assembled using either stepper motors or servo 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.

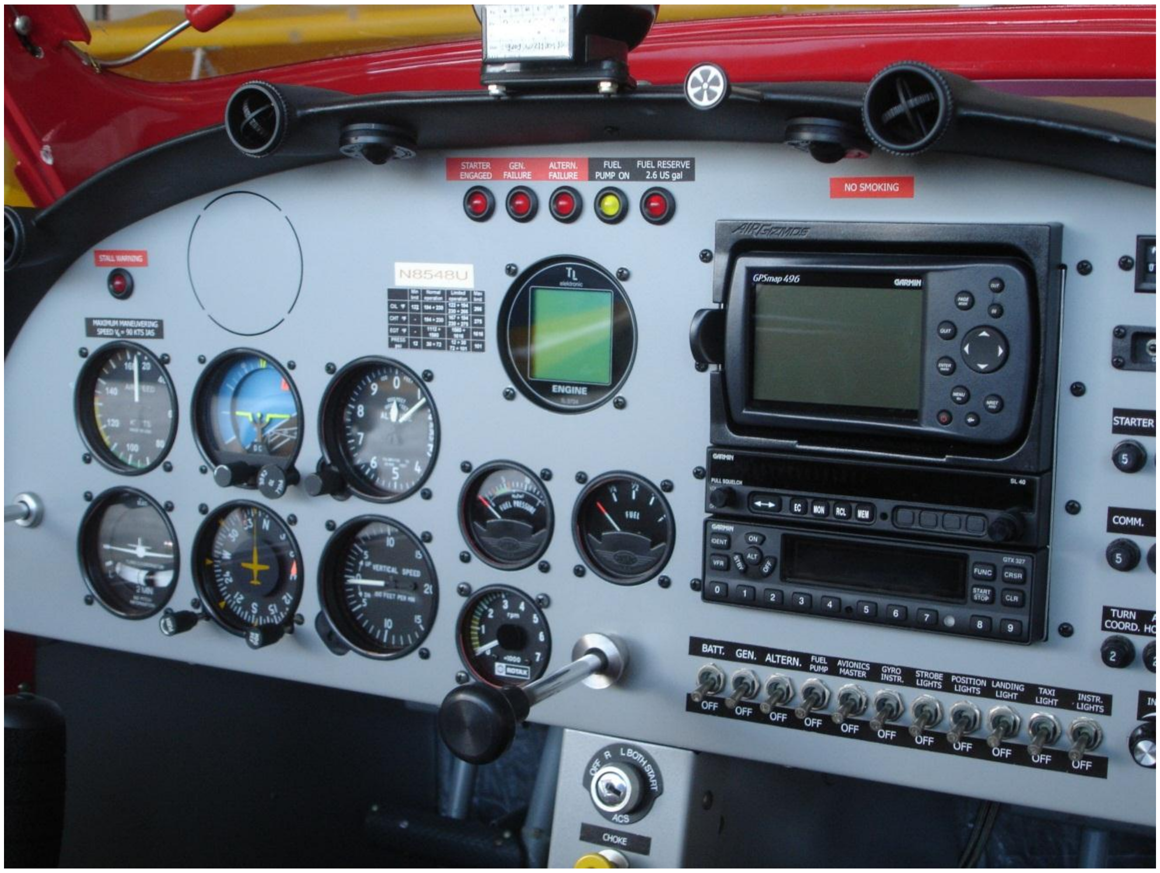


Figure 3-2. Gauges, switches and lights to be implemented. The “Six-pack” gauges are the cluster of six large gauges to the left of the picture. *Photo by Robert Gysi.*

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 to degree of motion was to use stepper motors. A problem occurs 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, while the second a servo motor.

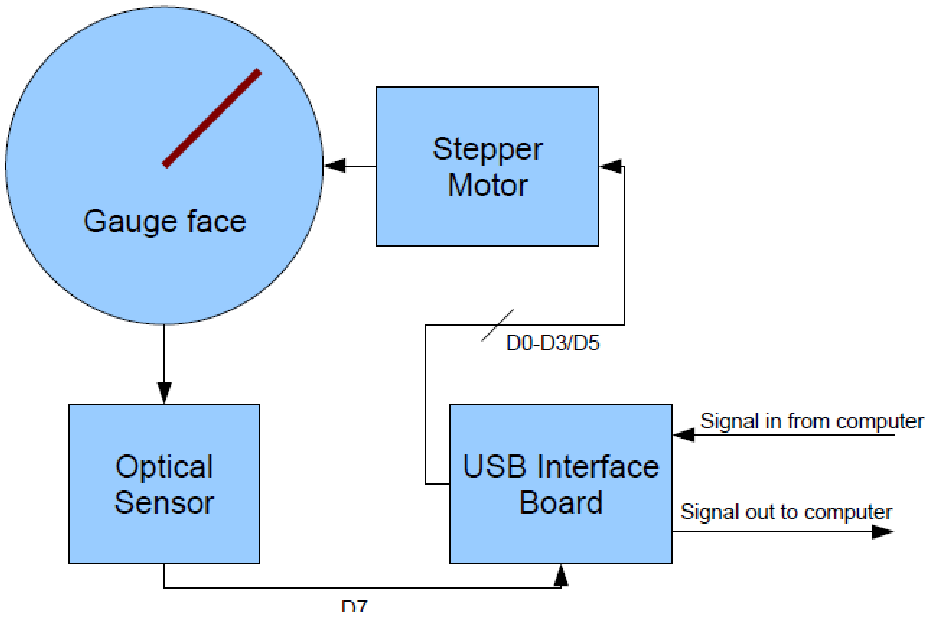


Figure 3-3 Stepper Motor Control Diagram. *Diagram by Lewis Vail*

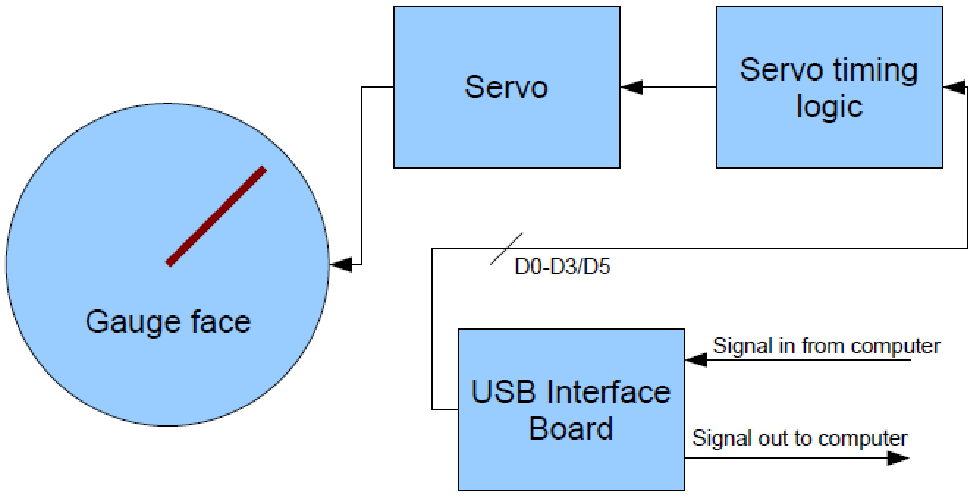


Figure 3-4. Servo Motor Control Diagram. *Diagram by Lewis Vail.*

The aircraft flight instruments will be interfaced with the computer and the flight simulator X-Plane. Each aircraft flight instrument will be 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 will reduce our overall cost to design and build the aircraft flight instruments for our GoBosh 700S flight simulator.

Each aircraft flight instrument will be 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.

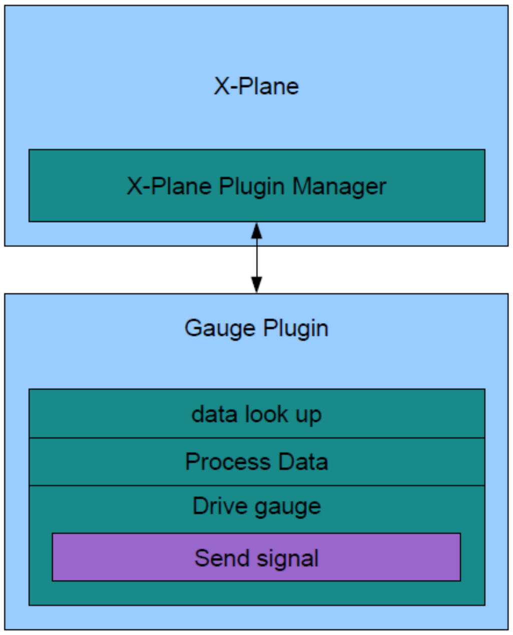


Figure 3-5. Gauge Software Interface. *Diagram by Lewis Vail.*

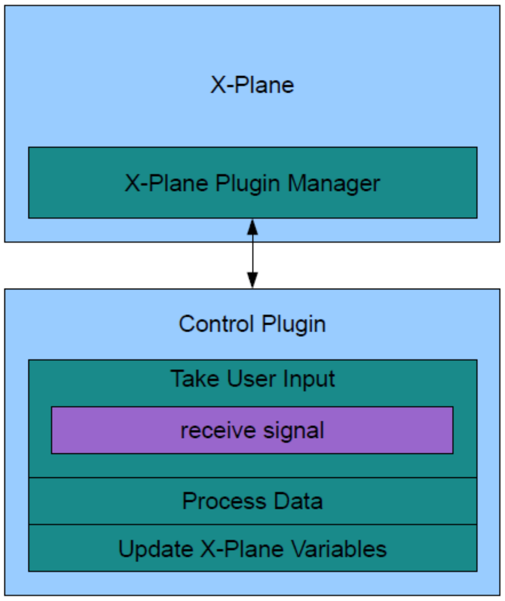


Figure 3-6. Control Software Interface. *Diagram by Lewis Vail.*

The following Figures and flow charts highlight the higher order levels of our design and process.

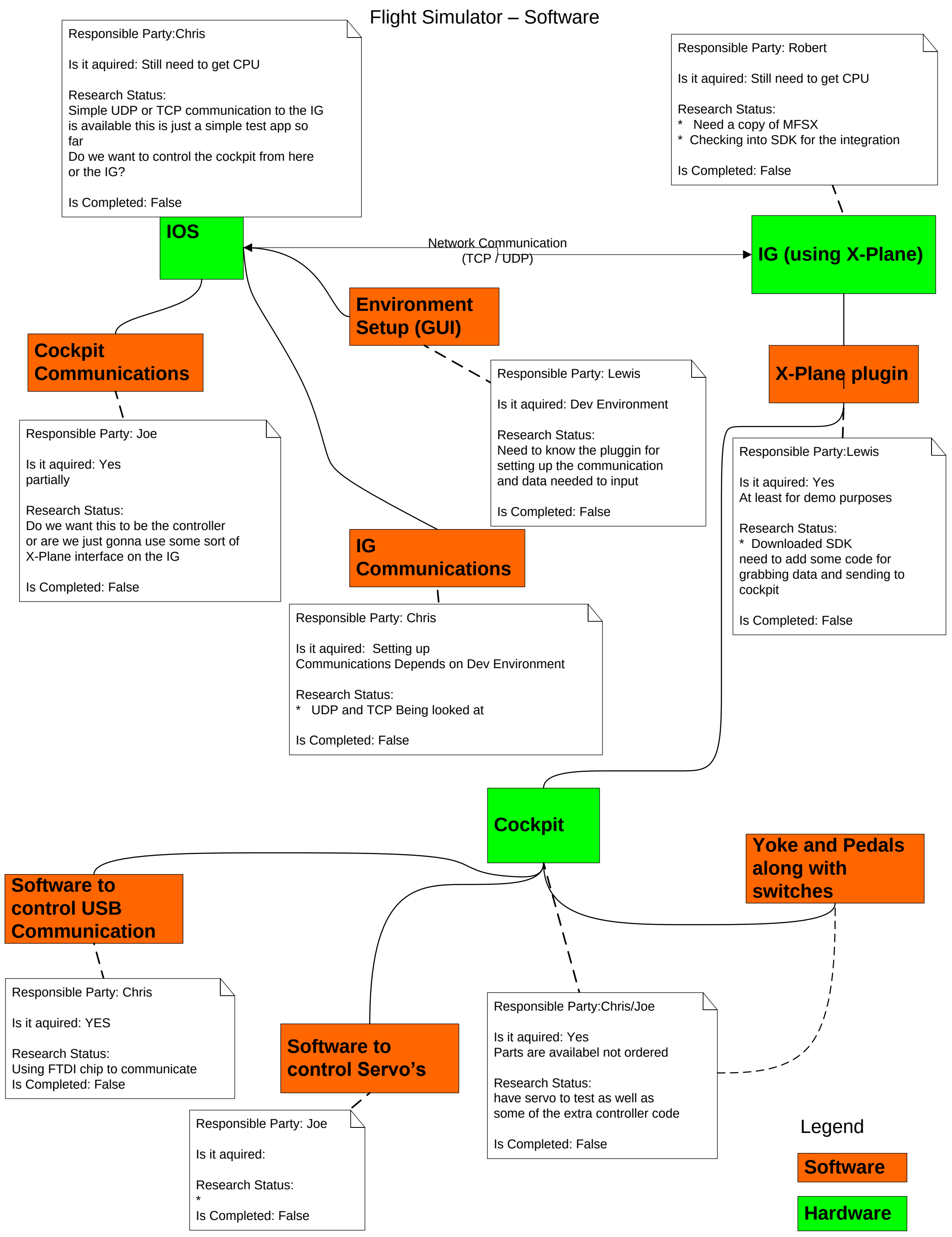

Flight Simulator – Software
Responsible Party:Chris
Is it aquired: Still need to get CPU
Research Status:
Simple UDP or TCP communication to the IG is available this is just a simple test app so far
Do we want to control the cockpit from here or the IG?
Is Completed: False
Responsible Party: Robert
Is it aquired: Still need to get CPU
Research Status:
* Need a copy of MFSX
* Checking into SDK for the integration
Is Completed: False
IOS
Network Communication
(TCP / UDP)
IG (using X-Plane)
Environment Setup (GUI)
Cockpit Communications
X-Plane plugin
Responsible Party: Lewis
Is it aquired: Dev Environment
Research Status:
Need to know the pluggin for setting up the communication and data needed to input
Is Completed: False
Responsible Party: Joe
Is it aquired: Yes
partially
Research Status:
Do we want this to be the controller or are we just gonna use some sort of X-Plane interface on the IG
Is Completed: False
Responsible Party:Lewis
Is it aquired: Yes
At least for demo purposes
Research Status:
* Downloaded SDK
need to add some code for grabbing data and sending to cockpit
Is Completed: False
IG Communications
Responsible Party: Chris
Is it aquired: Setting up Communications Depends on Dev Environment
Research Status:
* UDP and TCP Being looked at
Is Completed: False
Cockpit
Yoke and Pedals along with switches
Software to control USB Communication
Responsible Party: Chris
Is it aquired: YES
Research Status:
Using FTDI chip to communicate
Is Completed: False
Software to control Servo's
Responsible Party:Chris/Joe
Is it aquired: Yes
Parts are availabel not ordered
Research Status:
have servo to test as well as some of the extra controller code
Is Completed: False
Responsible Party: Joe
Is it aquired:
Research Status:
*
Is Completed: False
Legend
Software
Hardware

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

3.2 Microcontroller Design

After some research on the types of microcontrollers that are available with the ability to do the required tasks, we came across 2 different chips at first: (1) Atmel AT89C5131 or AT90USB these are Atmel chips that are able to speak USB with a little bit of coding and little outside connections, they can also control any motor as a person in the group has used the chip before to control stepper motors. The power consumption of these chips is very low and they can run off of the USB reported specs. (2) Pic's PIC18F4550, this is another chip capable of speaking USB and meeting the specs necessary to either run off the USB power or can use an outside source if necessary.

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

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

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

One of the most important parts of the design will be with the power consumption. The USB port on a computer can supply 5 volts and 500mA as we have already stated. The use of this to drive a motor of any kind is pushing the power of the USB port to the limit, not to mention the ability of the motor to push current back at the port which can kill the port all together. So we will need to either use a buffer or some sort of power supply for each controlled gauge.

3.2.1 Implementation of Hardware

The implementation or actual making of the hardware comes from the FTDI FT245BL chip that is 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 comes with a standard crystal, and an EEProm that holds some descriptions that we could use for the gauges. Along with the FTDI chip we need a chip to keep the FTDI chip from being overloaded and we will use the 4050 buffer chip to help drive the small stepper motor.

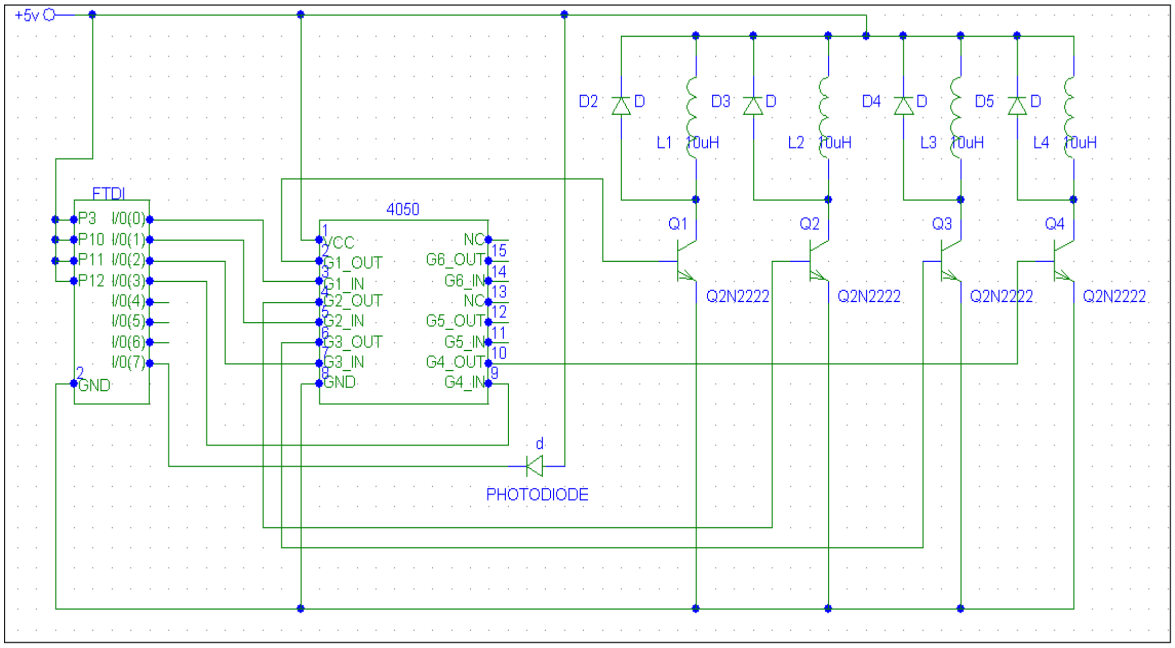


Figure 3-8. Circuit for the stepper motor controls *Created by Chris Dlugolinksi*

This figure shows the connections needed to run each of the coils of the stepper motor and how that might be done. The microcontroller being the FTDI chip and each of the coils needs to be helped on by the transistors. Although most designs call for some sort of MOSFET design this is just preliminary and the actual circuit has not been designed. The diodes are in there to stop the current from coming back and hurting the hardware components. Also note that all of the pins are not displayed at this point only the important parts of the design have been used so far.

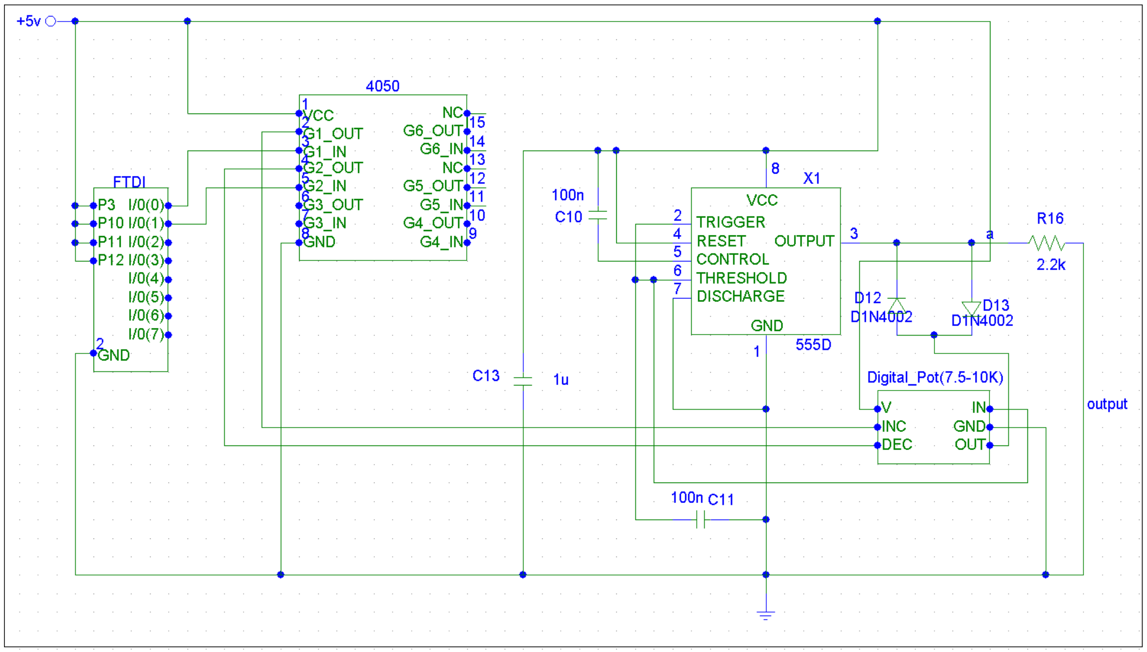


Figure 3-9. Circuit for the Servo motor controls *Created by Chris Dlugolinksi*

The figure 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 has to be decided on still as there are other ways to control the motor. The other way being the PIC microcontroller it is easy to program and will be able to control the servo a lot better and more efficiently than the 555 timer and the digital potentiometer. The ideas are both good and will need to be designed more and tested a lot.

The power supply needs to be able to provide 5 volts for any of the circuitry that we need as well as maybe a 12 volts for the motors that may be used in the design. A simple power supply can be used that would need to be able to give at least 6 amps. That is 1 amp for each of the gauges. We could either design this or use an extra power supply from a computer, or they make molex connectors that use the power supply already in the computer to give outside electronic hookups the 5 and 12 volts they would need.

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 shifts most of the software to the host computer and also shifts where we manage all of our gauges as well.

Although to control the servo motor we will need to have a pulse width change, and to do this with the FTDI chip is difficult, and involves a few other chips that would need to be learned. An answer to this is to use a PIC chip that can be programmed to give out the pulses needed. This chip would just need to know what the output needs to be and it would look up the data to output in the table, then output the correct pulse width. Very simple but depending on the cost of the chip could get expensive.

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 3.3 shows the actual flight instruments that will be simulated for the GoBosh 700S. Figure 3.4 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 3-4 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 some of the ways we plan on building our own for simulation.

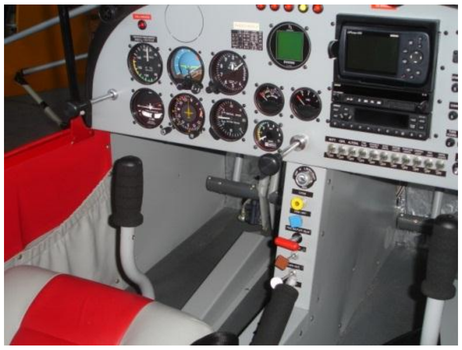


Figure 3-10. Cockpit with gauges to be implemented. *Photo by Robert Gysi*

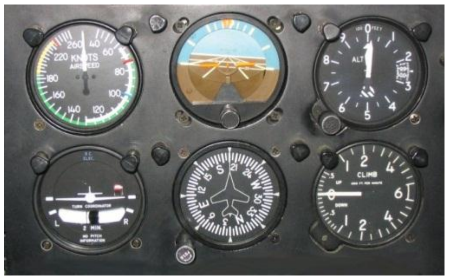


Figure 3-11. Closeup of the standard “Six-Pack” – Wikipedia *used with permission under the GNU License*[8]

[8] *(2009, Dec.). Six Flight Instruments.jpg Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/File:Six_flight_instruments.JPG*

Figure 3-5 shows the basic layout of how each flight instrument will be connected to the game computer. Each flight instrument will be programmed and controlled using a FTDI FT246BM USB chip, which will be connected to a USB hub. The USB hub is connected to one of the game computers USB ports.

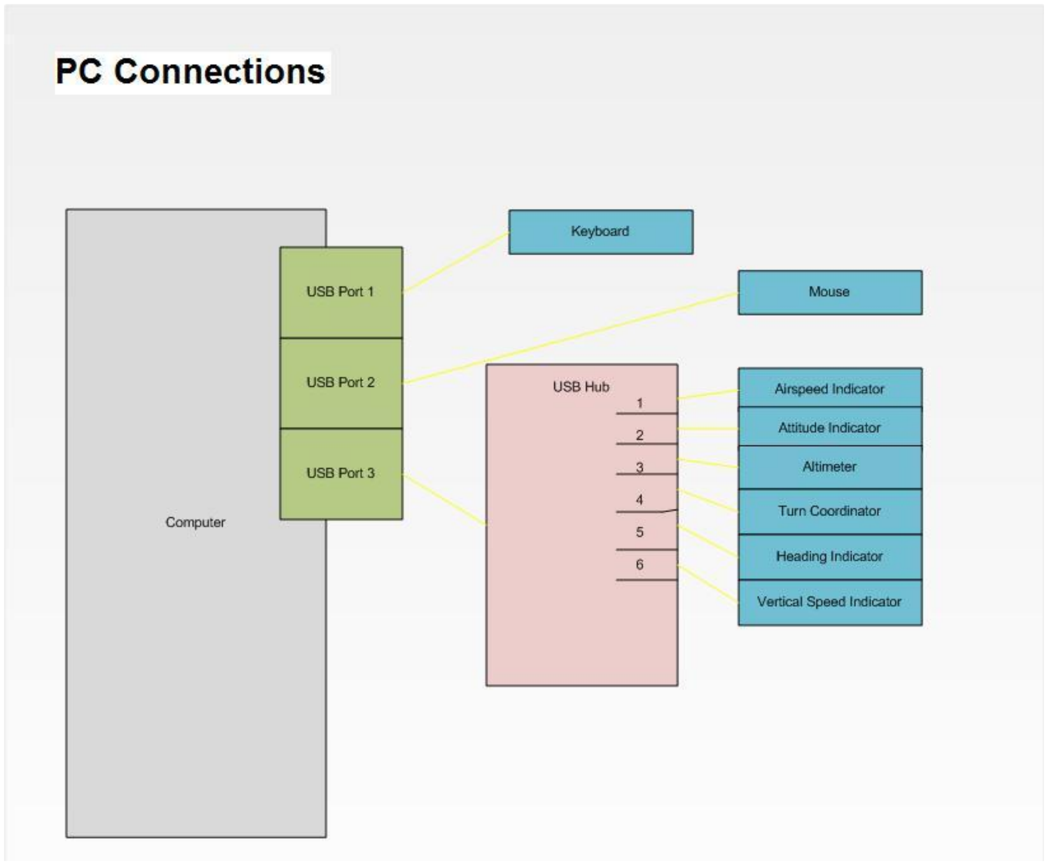


Figure 3-12: Diagram of USB controlled flight instruments showing the connections to the computer. *Created by Joseph Munera*

A mouse and keyboard will also be connected to one of the USB ports on the game computer. This will allow 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 3-6 below shows the back of an aircraft panel with all the USB controlled gauges installed. Each gauge is mounted to the back of the aircraft front panel with the USB cable connection located on the back of the aircraft flight instruments. These USB cables will connect to a USB hub, which is connected to a USB port on the game computer.

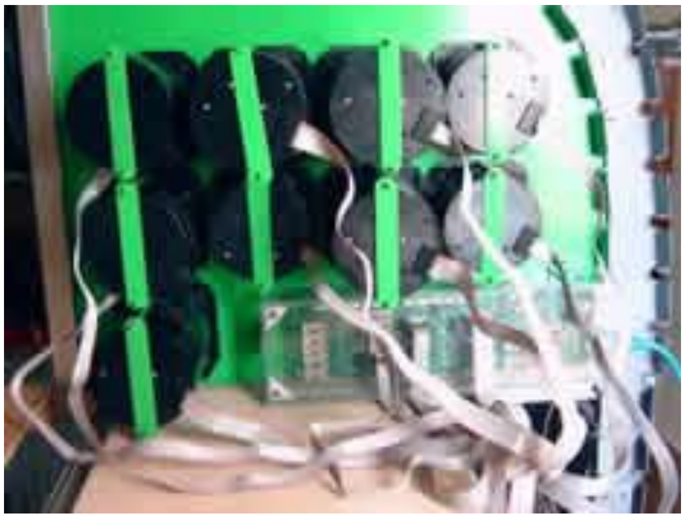


Figure 3-13. Back view of aircraft panel with USB controlled gauges mounted. *Photo used with permission from Jan Verley.* [9]

To accomplish the requirements regarding the flight instruments there are a few options to consider. The options available are to mod a real world gauge to interface with the flight software, buy a simulation gauge kits that come 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 mechanical inner workings are already there so that all we would have to create would be the interface from the computer. Unfortunately, creating this interface is not an easy task. In the aircraft that we are simulating all the gauges we are replicating are either barometric or gyroscopic, neither of which are easily stimulated by a computer. For instance, the airspeed indicator could be stimulated by a variable speed fan blowing into the barometer, but this is neither easier to implement nor more accurate than the other methods we have available. The other problem we have 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 to would be to buy gauge kit 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 are however a few gauges that we will be modeling with these kits just because the mechanics of the gauges are complicated enough to warrant spending the money.

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

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 3-7. 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.

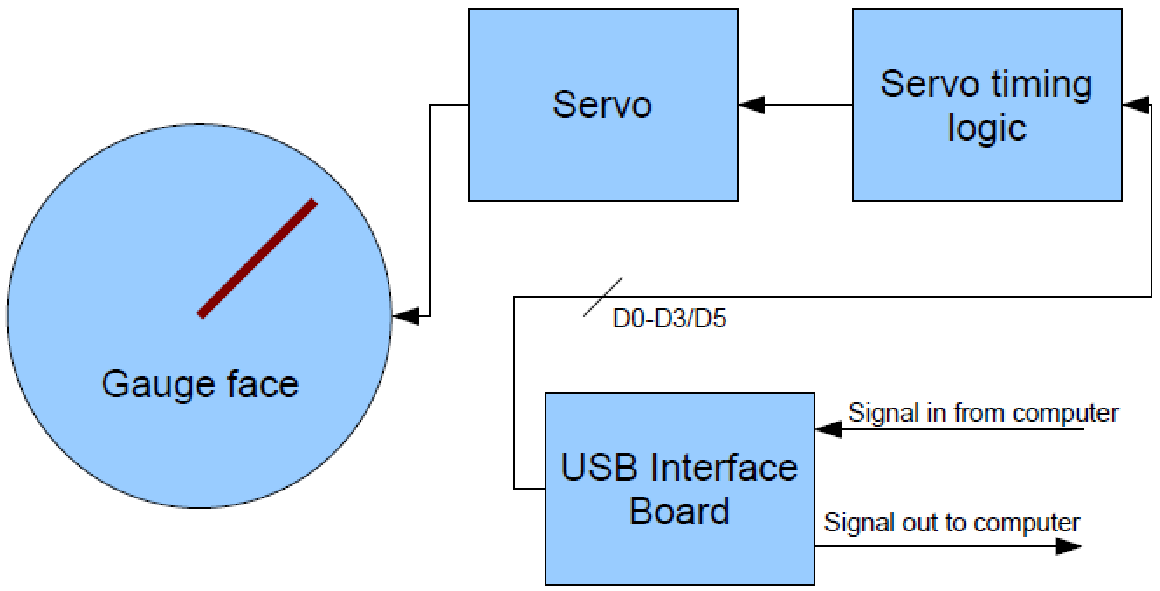


Figure 3-14: Block diagram of servo-based gauge. *Created by Lewis Vail*

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 would use stepper motors that have a step of 1.8 degrees. They can then be 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 would implement an optic sensor to reset the needle to its home position at initialization. This stepper motor based implementation is illustrated in figure 3-8.

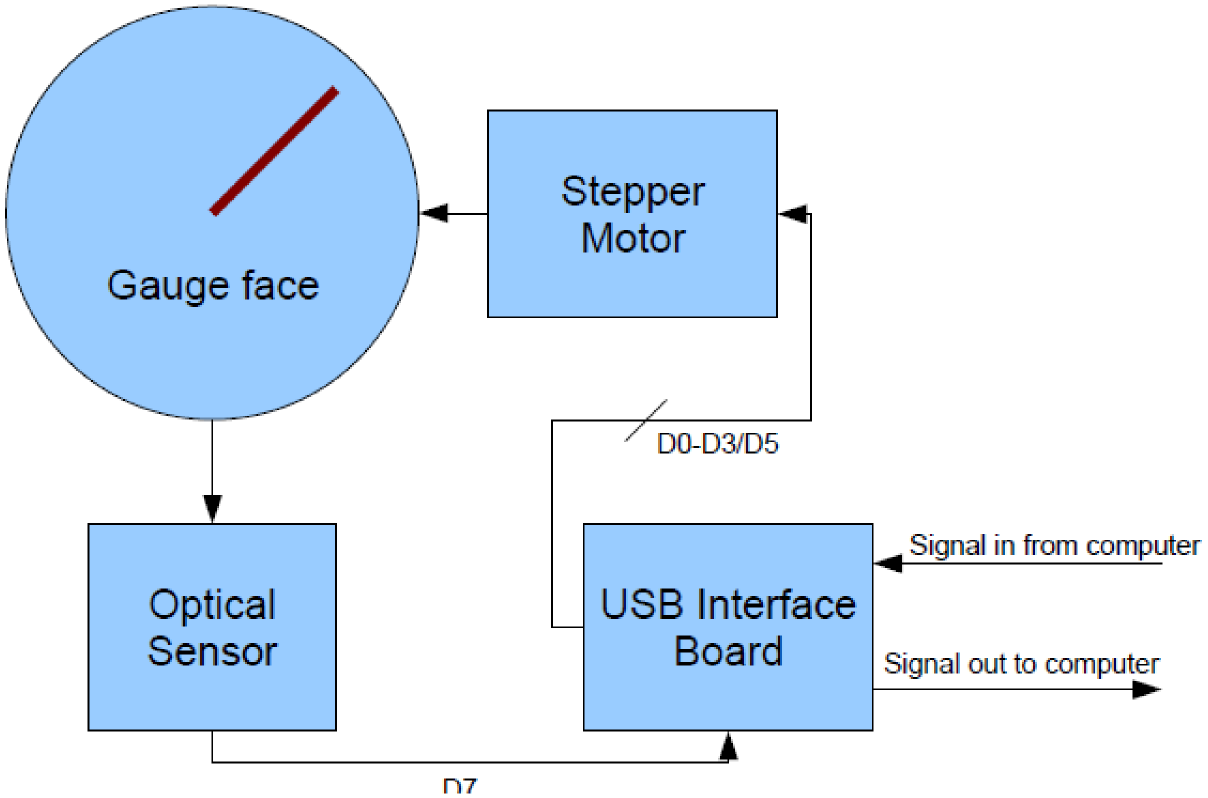


Figure 3-15: Block diagram of stepper motor based gauge. *Created by Lewis Vail*

Given the options above, we will 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 will most likely take much more time to test and refine as the sensor circuit may take a little while to properly calibrate and mount. We chose to model two of the gauges with kit gauges because those two required multiple motors turning independently. Although these two could probably be implemented by using the techniques used for the other ones with some extra components, this seemed to be too difficult mechanically to justify the cost. The following are the details of how each gauge will be modeled.

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 3-9 shows the faceplate of a "three pointer" sensitive aircraft altimeter displaying an altitude of 10,180 ft. While Figure 3-10 shows the approx. dimensions we will use.

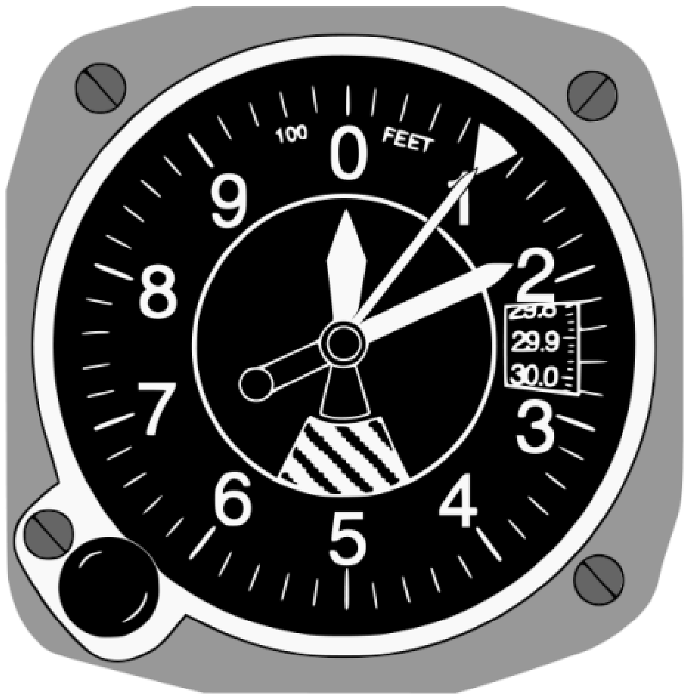


Figure 3-16. Altimeter Face plate, dials, and barometric pressure adjustment knob. *This image has been released into the public domain by its author, Bsayusd at the Wikipedia project.*[10]

[10] (*2009, Nov.). SVG Drawing of Altimeter [Online]. Available: http://en.wikipedia.org/wiki/File:3-Pointer_Altimeter.svg*

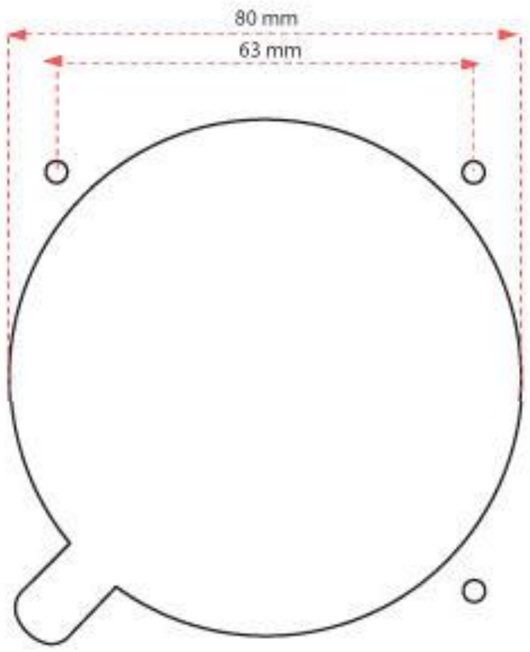


Figure 3-17. Front view measurements of altimeter. *Photo used with permission from Mark Verschaeren*[11].

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

I. Construction of the altimeter will consist of several parts. The gauge will 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.

Multiple motors maybe 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 will be USB controlled by an electronics board, which will deliver direct feedback of the position of each pointer. The small indicator could be controlled by a stepper motor. A second stepper motor can be used to regulate the air pressure scale by using a dial located at the lower left land hand side of the gauge.

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

Simkits.com sells all the necessary components to build a complete and realistic altimeter. The picture below illustrates one way of constructing an altimeter using two servo motors and a stepper motor.

II. The altimeter could be built using 3 stepper motors. There would be a motor to move the 100 ft hand and 1000 ft hand. The third stepper motor can be 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 will 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 could be used to sense when the motor is moved to the start position.

III. A real commercial altimeter could also be used for the flight simulator. It would require accurately generating slow varying pressures within small fractions of a PSI. Figure 3-11 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.

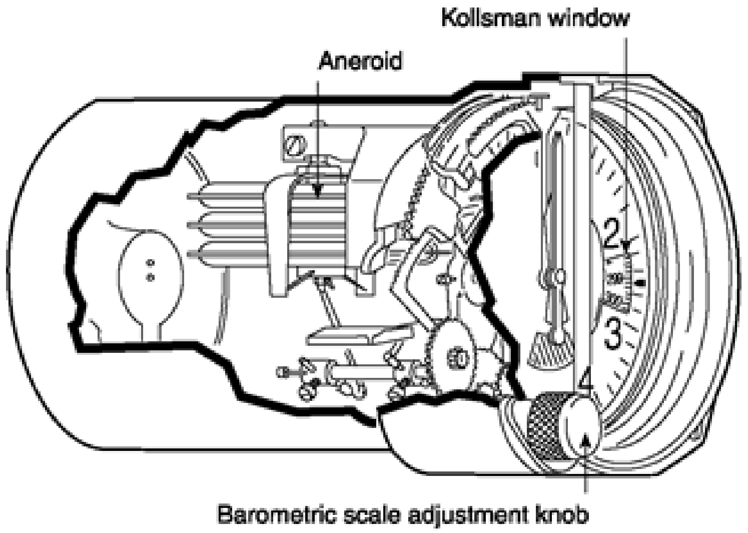


Figure 3-18. Actual altimeter components. *As a work of the U.S. federal government, the image is in the public domain*[12].

[12] (*2009, Nov.). Drawing of Altimeter [Online]. Available: http://en.wikipedia.org/wiki/File:Sens_alt_components.PNG*

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 will be fed with data from X-Plane, simulating the mechanical operation of a pressure fed 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 will be 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 will be developed as a plug in for X-Plane. Figure 3-12 (below) shows an example of an airspeed indicator, similar to the one utilized in the GoBosh G700S, while Figure 3-13 indicates the approximate dimensions and hole patterns we will use to create our simulated version.

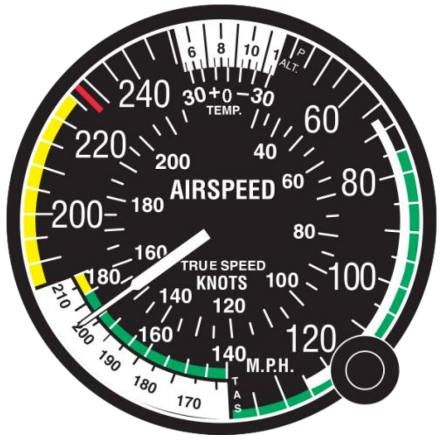


Figure 3-19. Airspeed Indicator Faceplate - Wikipedia. *Used with permission under the GNU Free Documentation License*[13]

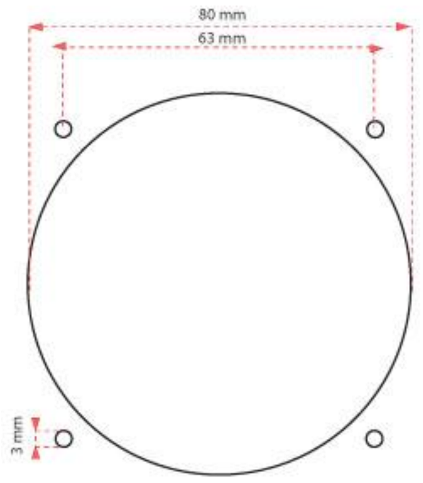


Figure 3-20. Front view measurements of airspeed indicator. *Photo used with permission from Mark Verschaeren*[14].

[13] (*2009, Nov.). Drawing of Airspeed Indicator from FAA Instrument Flying Handbook [Online]. Available: http://en.wikipedia.org/wiki/File:True_airspeed_indicator-FAA.SVG*
[14] For e-mail response granting permission to use, see Appendix C

I. Construction of the airspeed indicator will consist of several parts. The gauge will 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 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 can be 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 3-14 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.

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

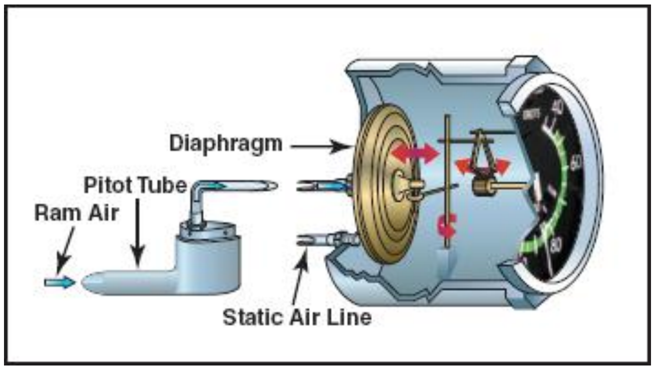


Figure 3-21. Components of an actual airspeed indicator. *As works of the U.S. federal government, all FAA images are in the public domain*.[15]

[15] (*2009, Nov.). Airspeed Indicator Cutaway - Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/File:ASI-operation-FAA.png*

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 3-14 shows the faceplate of a simple vertical speed indicator, similar to the one we will be implementing in our simulator. Figure 3-15 shows the approximate dimensions as well as the whole pattern for the gauge assembly.

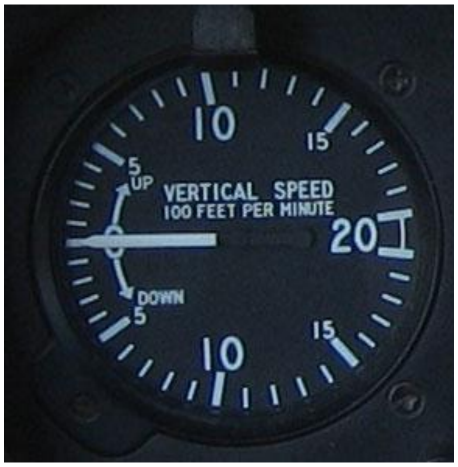


Figure 3-22. Face plate of the vertical speed indicator. *This work has been released into the public domain by the copyright holder, Benet Allen*.[16]

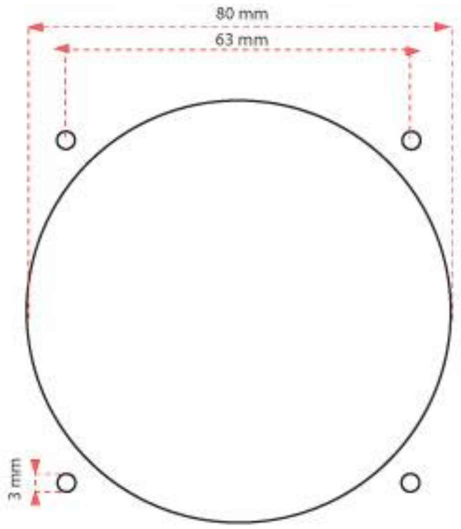


Figure 3-23. Front view measurements of vertical speed indicator. *Photo used with permission from Mark Verschaeren*.[17]

[16] *(2009, Nov.). Vertical Speed Indicator - Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/File:R22-VSI.jpg*

I. Construction of the vertical speed indicator will consist of several parts. The gauge will 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 can be 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 3-16 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.

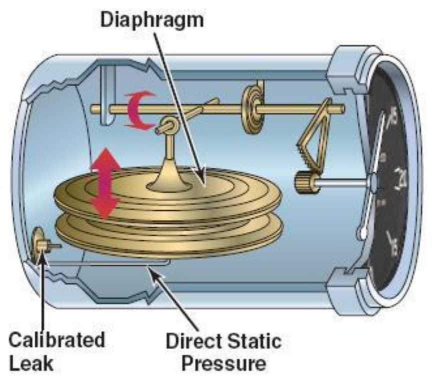


Figure 3-24. Components of an actual vertical speed indicator. *This work is in the public domain in the United States because it is a work of the United States Federal Government*[18]

[17] For e-mail response granting permission to use, see Appendix C
[18] (*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.

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 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 3-17, 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 in of the 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 3-18 represents the approximate hole pattern for the gauge that will be built for use in our simulator cockpit.

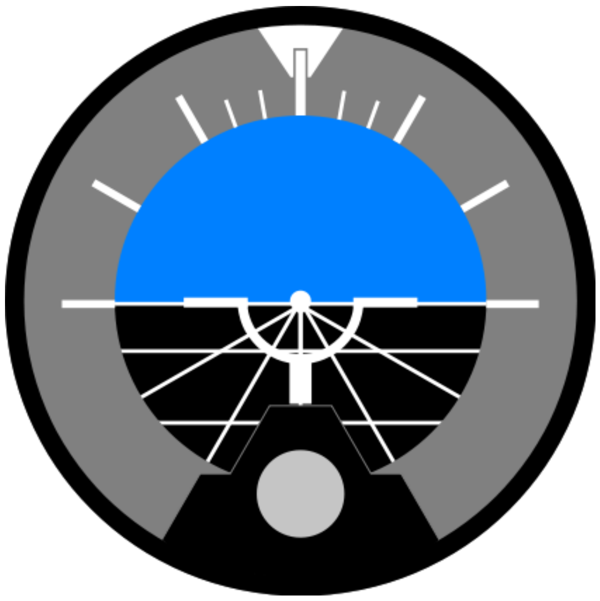


Figure 3-25. Face plate of the attitude indicator. *Used with permission under the GNU Free Documentation License*[19]

[19] *(2009, Nov.). Artificial Horizon - Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/File:Attitude_indicator_level_flight.svg*

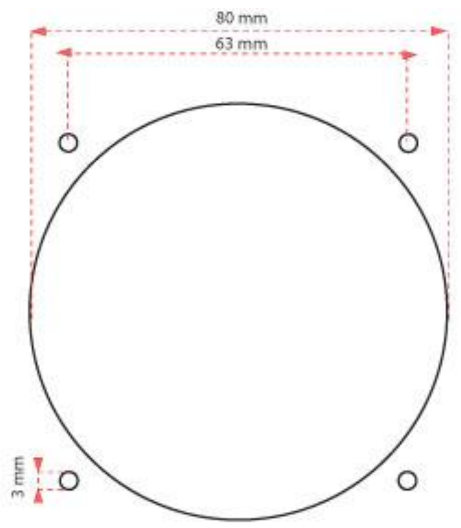


Figure 3-26. Front view measurements of attitude indicator. *Photo used with permission from Mark Verschaeren.*[20]

I. Construction of the attitude indicator will consist 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 most likely cannot 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. Figure 16 shows the inherent complexity involved when trying to use a real aircraft flight instrument in building a flight simulator

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

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.

V. Another option would be 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 3-19 shows an example of a kit version available from Flight Illusion. This (along with a similar one from SimKits) simulate 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.

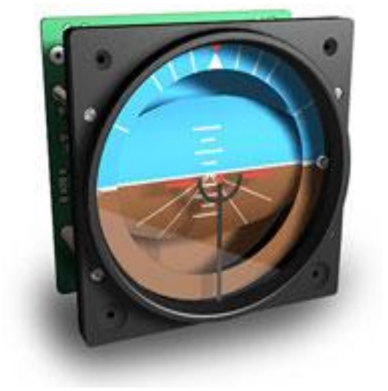


Figure 3-27. Flight Illusion Attitude Indicator Gauge. *Photo used with permission from Mark Verschaeren.*[21]

For the attitude indicator we have decided to go with purchasing a kit gauge versus creating one. This is because of the mechanical complexity involved in building our own. We are currently in discussions with both SimKits and Flight Illusion in regards to additional sponsorship or discounts on their products.

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[22]. If the turn is coordinated than the ball that exists in track in the bottom of the gauge (see Figure 3-20), 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 3-21 shows the approximate dimensions for the mounting and hole patterns for the simulated gauge we will be installing into the GoBosh cockpit.

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

[22]*(2009, Dec.). Turn Coordinator - Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/Turn_coordinator*

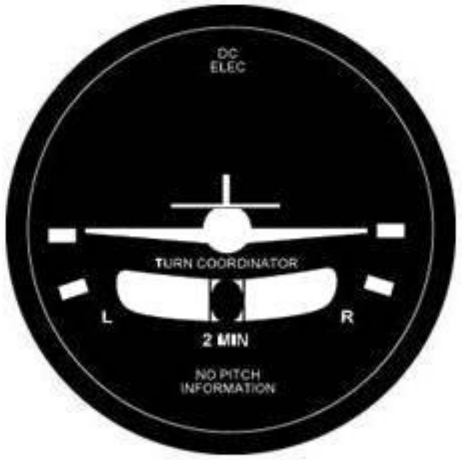


Figure 3-28. Turn Coordinator *Used with permission under the GNU Free Documentation License*[23]

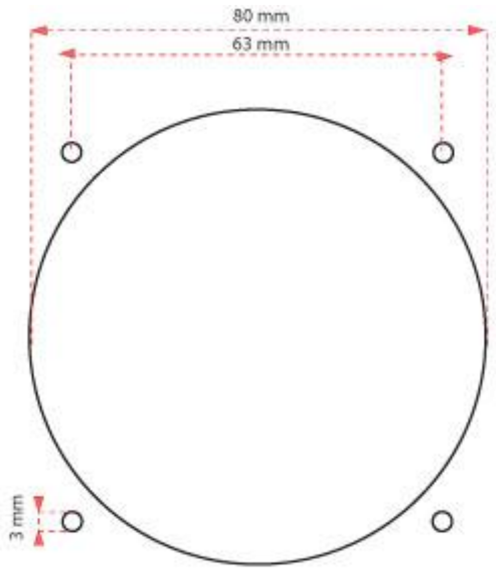


Figure 3-29. Front view measurements of turn coordinator. *Photo used with permission from Mark Verschaeren*.[24]

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.

[23] (*2009, Dec.). Turn Coordinator Drawing - Wikipedia [Online]. Available:*
http://en.wikipedia.org/wiki/File:Turn_indicator.png
[24] For e-mail response granting permission to use, see Appendix C

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 most likely cannot 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 3-22 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 to be removed by disassembling the aircraft flight instrument. Servo motors or stepper motors would then be place inside the aircraft flight instrument.

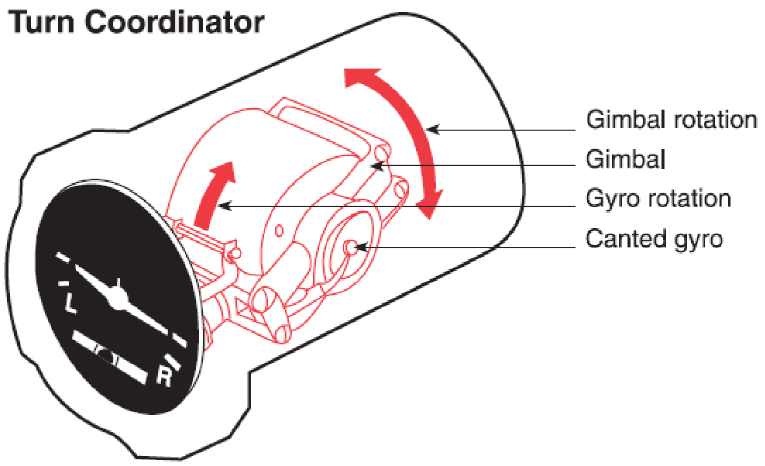


Figure 3-30. Components of an actual turn coordinator. *As a work of the U.S. federal government, the image is in the public domain*.[25]

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 purchasing a kit
gauge versus creating one. This is because of the mechanical complexity involved. We are currently in discussions with both SimKits and Flight Illusion in regards to additional sponsorship or discounts on their products.

[25]*(2009, Dec.). File: Turn_indicators.png - Wikipedia [Online]. http://en.wikipedia.org/wiki/File:Turn_indicators.png*

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 effected by the downward slope of the Earth's magnetic field.[26] Figure 3-23 shows a common design for heading indicator face plates, while Figure 3-24 shows the approximate hole patterns used to design the gauge body, faceplate and mounting.

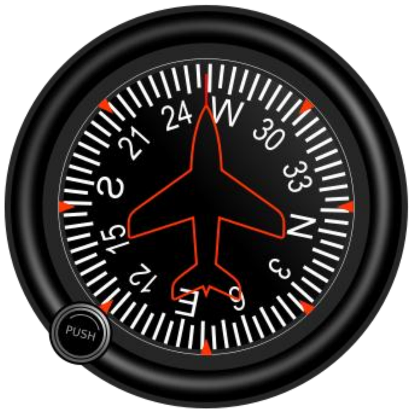


Figure 3-31. Face plate of the heading indicator. *Used with permission under the GNU Free Documentation License*[27]

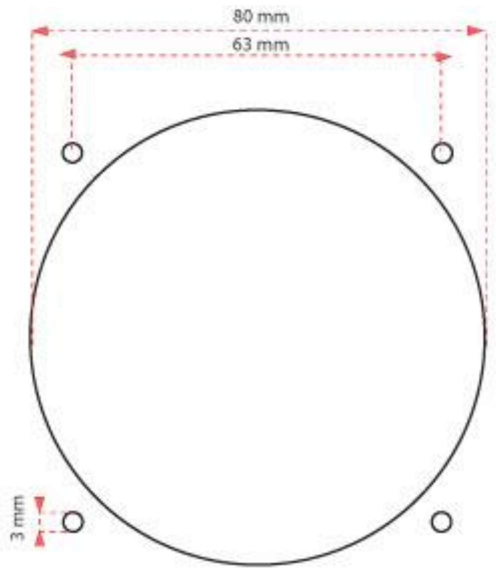


Figure 3-32. Front view measurements of the heading indicator. *Photo used with permission from Mark Verschaeren.* [28]

[26] (*2009, Dec.). Heading Indicator - Wikipedia [Online]. http://en.wikipedia.org/wiki/Heading_indicator*
[27] (*2009, Dec.). Turn Coordinator Drawing - Wikipedia [Online]. Available:* http://en.wikipedia.org/wiki/File:Turn_indicator.png
[28] For e-mail response granting permission to use, see Appendix C

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 3-25 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.

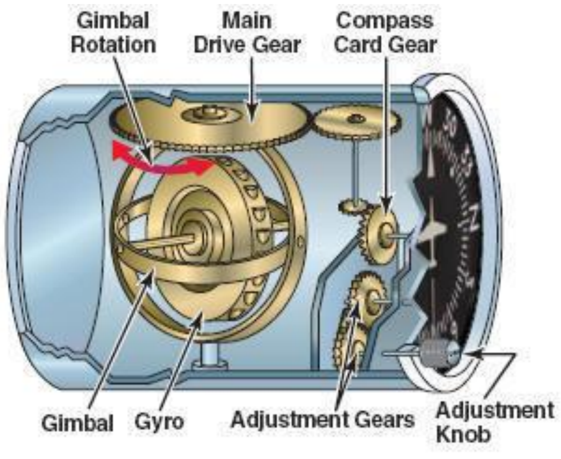


Figure 3-33. Components of an actual heading indicator. *Photo used with permission from Bob Miller*.[29]

For the heading indicator we have selected option 3. This is because stepper motors provide 360 degrees of rotation. In this gauge we definitely need the

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

capability to continuously rotate the instrument if someone was piloting the aircraft in a circle formation.

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 are 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.

Figure 3-26 shows a side view of a heading indicator constructed using a single stepper motor and an optical interrupter.

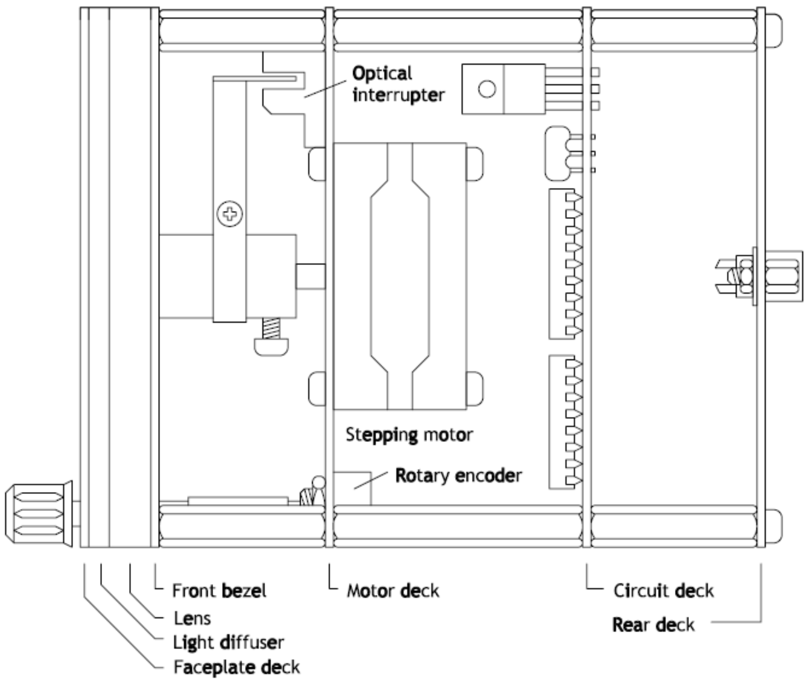


Figure 3-34. Side view of a single stepper motor flight instrument schematic. *Photo used with permission from John M. Powell. Copyright © 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009 by John M. Powell*[30]

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

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.

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.

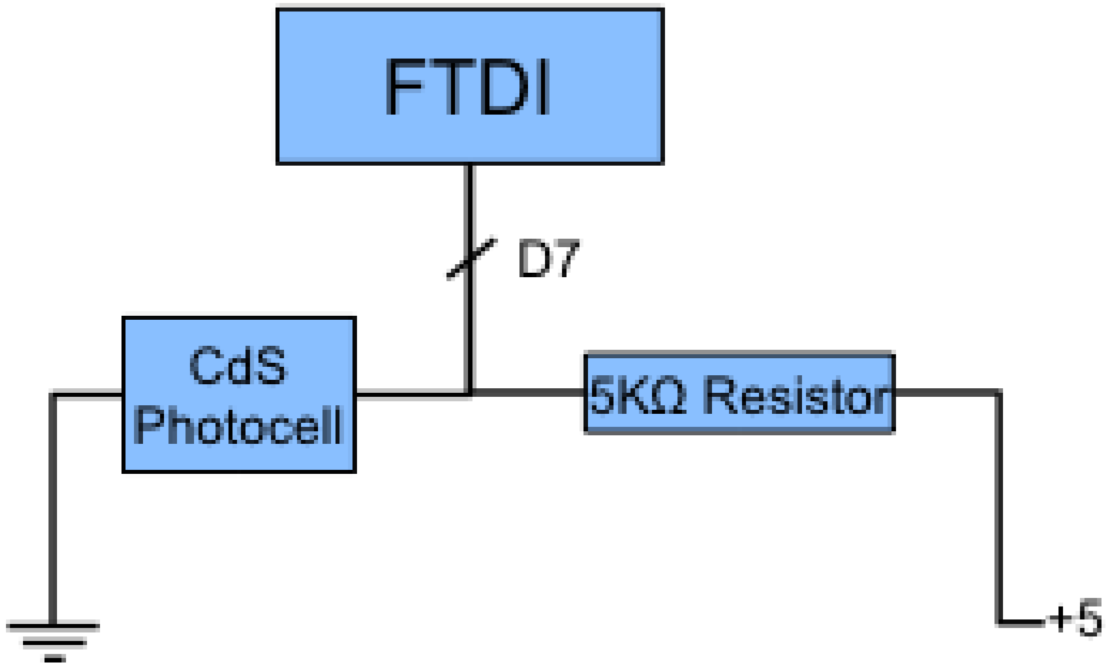


Figure 3-35: block diagram of light sensor used in optical interrupter. *Diagram by Lewis Vail*

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 3-27 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

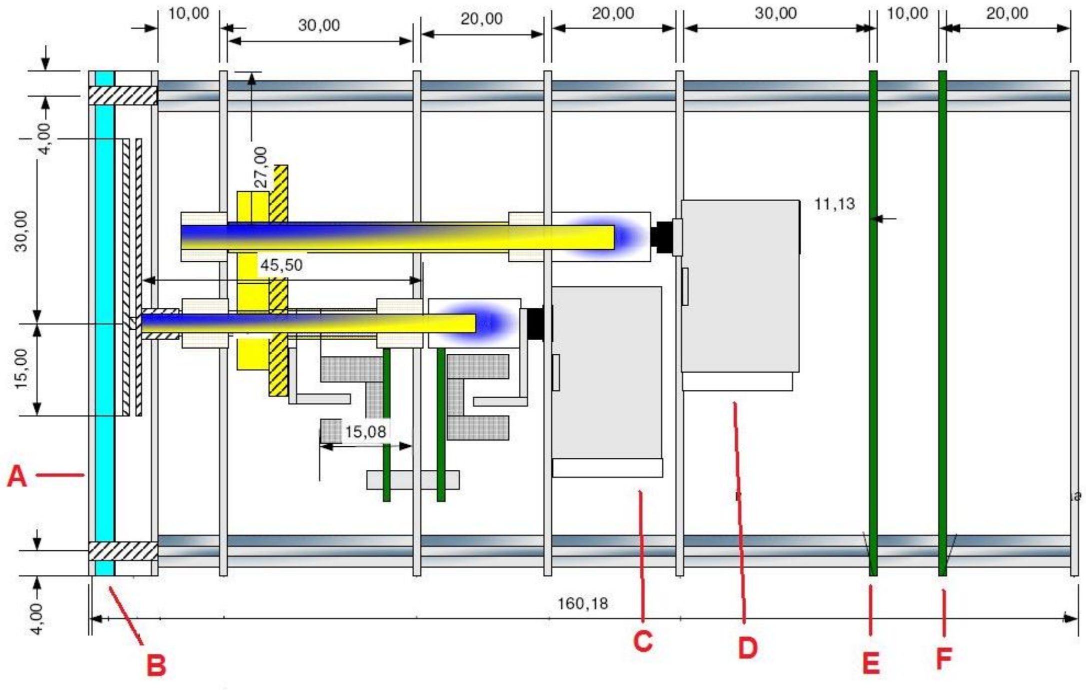


Figure 3-36: Side view of a flight instrument schematic using two stepper motors. *Photo used with permission from Jan Verley*. [31]

The entire aircraft flight instrument structure is made from a variety of components including Aluminum, M3 Studs, and metal tubes with a diameter of 2mm to 5mm. In addition, the glass window of the gauge can be constructed using acrylic or plexi-glass. While Figure 3-27 shows the representation of the gauge in drawing form, Figure 3-28 shows the implementation of the drawing into the basic structure. 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 3-29 shows additional dimensioning for the plexi-glass sheet that forms the windows of the gauge and the hole pattern for the front of the gauge.

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

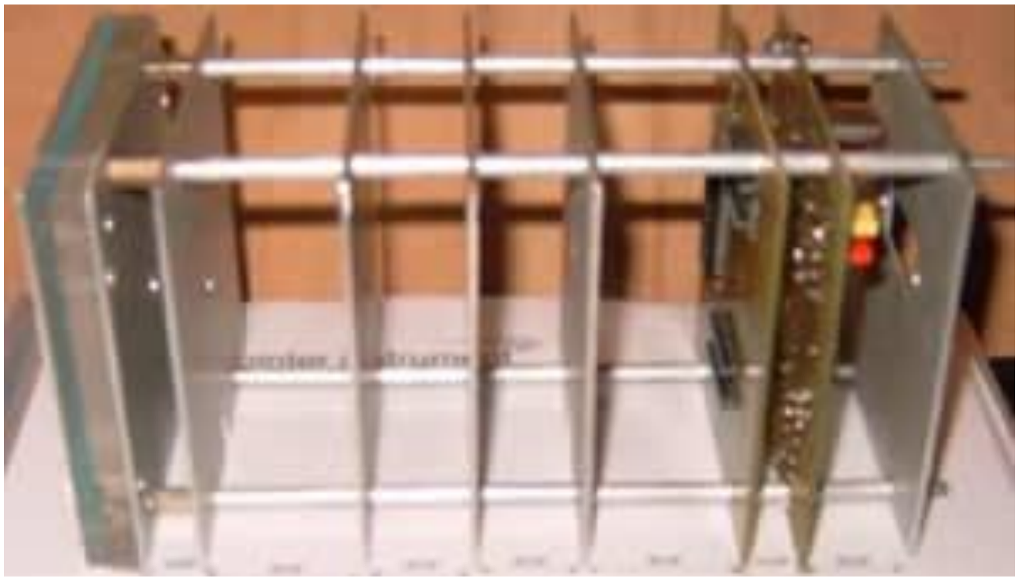


Figure 3-36. Constructed flight instrument. *Photo used with permission from Jan Verley.* [32]

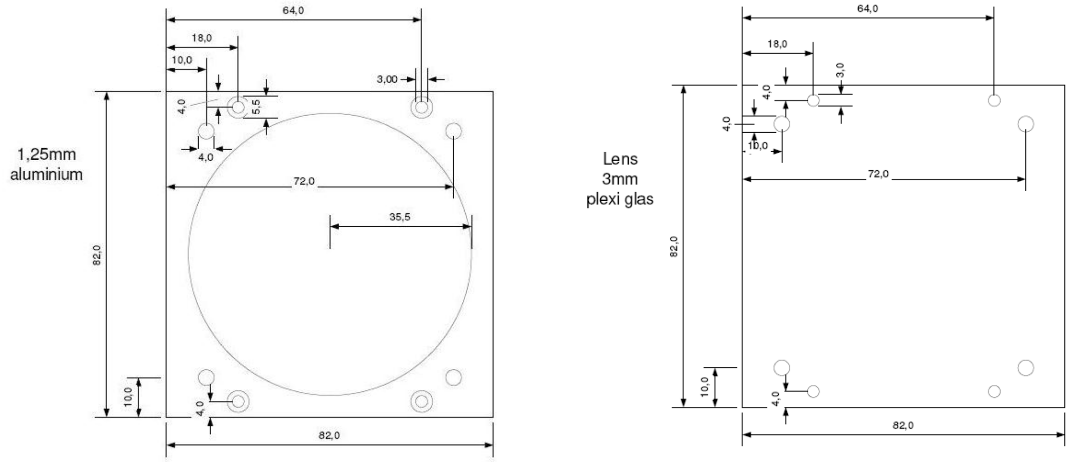


Figure 3-37. Front view measurements for a flight instrument consisting of two stepper motors (left) as well as dimensions for the faceplate (right). *Photo used with permission from Jan Verley.* [33]

Figure 3-38 shows the finished faceplate design that will be mounted to the front of the aircraft flight instrument gauge frame. The faceplate is made using a drawing program then printed out. A 1.25 mm aluminum plate is used to make the needles for the aircraft flight instrument. This aluminum can be purchased from any metal supply shop, or surplus stores such as Skycraft.

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

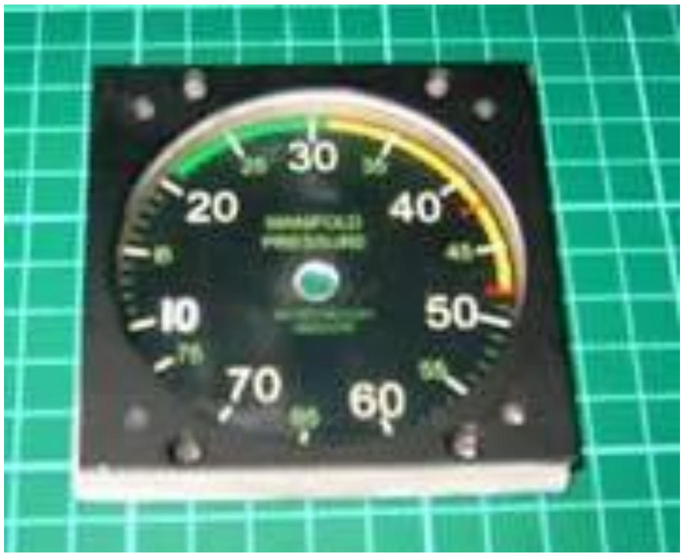


Figure 3-39. An assembled face plate for a flight instrument. *Photo used with permission from Jan Verley*. [34]

The two gears are mounted onto tubes using an epoxy. Ertalon is used for the bushing where the tubes will turn. Epoxy is used to mount the tubes to the aluminum plate. The shafts of each stepper motor are connected to the tubes using a plastic tube. Figure 3-31 shows the overall construction of the flight instrument with the gears and tubes mounted inside of the framework.

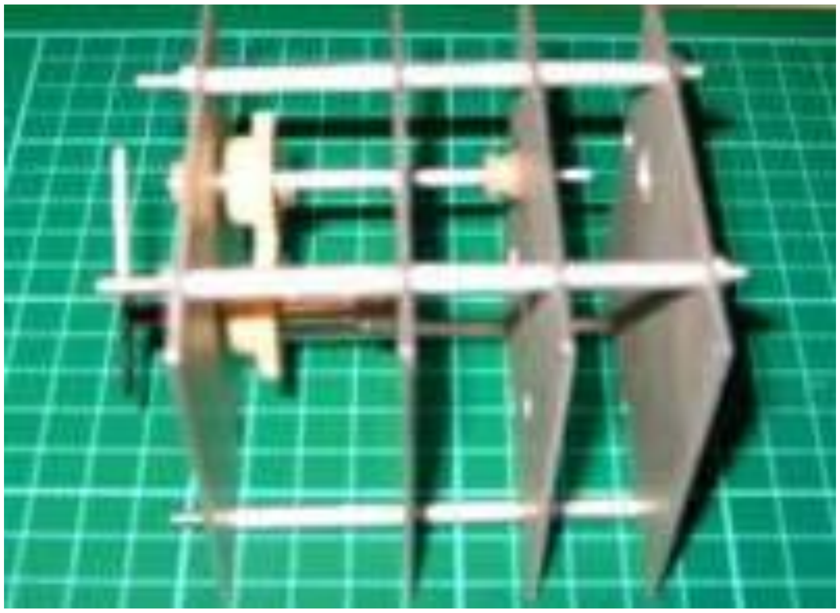


Figure 3-40. Assembled flight instrument housing with gears and tubes mounted. *Photo used with permission from Jan Verley.* [35]

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

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.

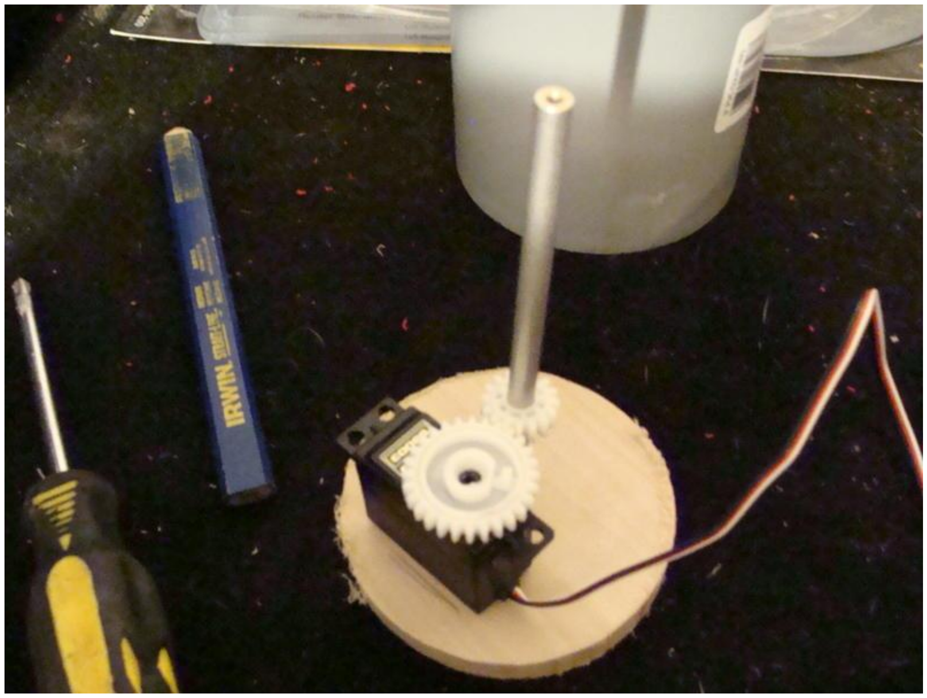


Figure 3-41. Flight instrument assembly consisting of a servo motor, nylon gears, circular wood cutout and aluminum shaft. *Photo by Joseph Munera.*

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 3-32 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.

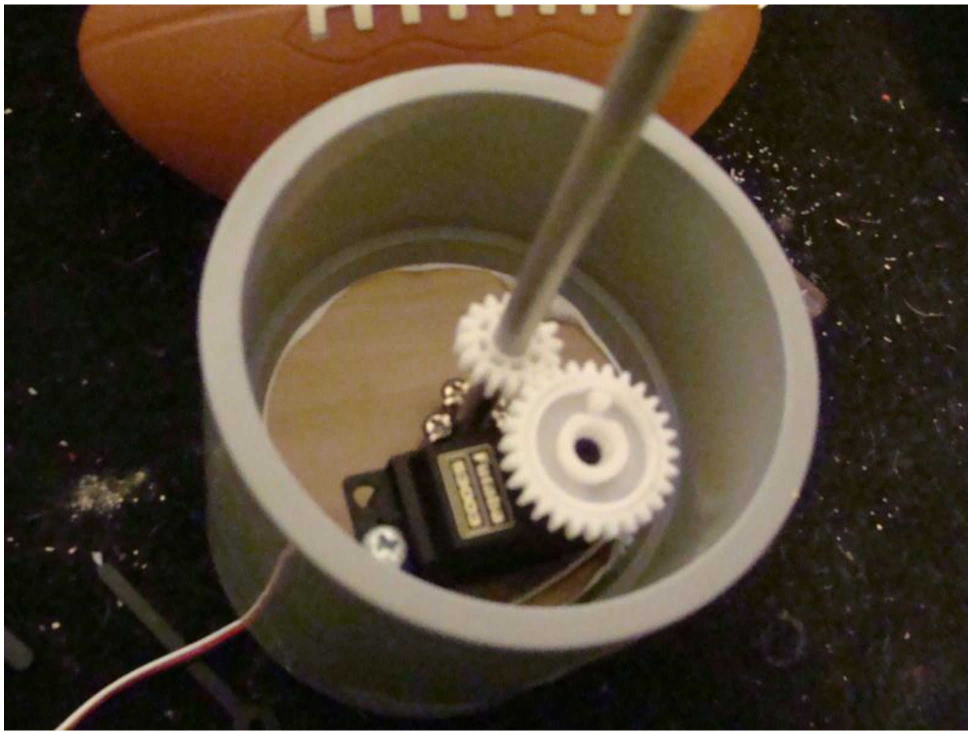


Figure 3-42: Flight instrument assembly glued and sealed to circular pipe. *Photo by Joseph Munera*.

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 3-34 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).

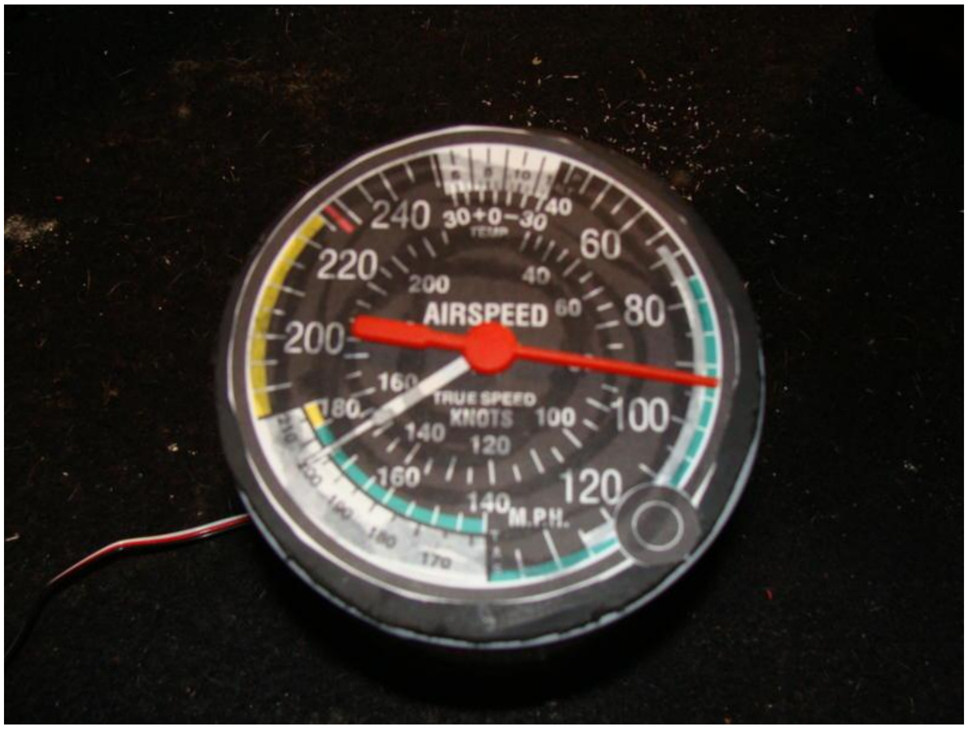


Figure 3-43. Assembled flight instrument with face plate and needle. *Photo by Joseph Munera*

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.

3.4 Flight Control Design

Flight Control Design is where we will design the control systems to relay information to the simulator computer to control the virtual aircraft on the screen. In order to do this we have designed a joystick around the existing stick in the aircraft, rudder pedals utilizing the existing pedals found in the aircraft and a throttle, which will not be included with the aircraft we are receiving, but will need to be built from scratch. The following sections cover each of these systems to be built.

3.4.1 Joystick Design

One of the most critical components of this project is the joystick controller that will be designed for the flight simulator. In most small general aviation aircraft the actual stick is actually connected to the flight surfaces through the use of connecting wires. For this simulator we will need to simulate this same operation digitally and then pass the data into the simulator computer so that the appropriate command is executed on the screen. In order to tackle this problem we need to first understand that in order to create this control stick we will need to work with two directions: the X-axis and the Y-axis. Connected to these two axes are potentiometers which as the stick is moved, change in resistance which allows one to map when at a particular output voltage a certain position has been reached.

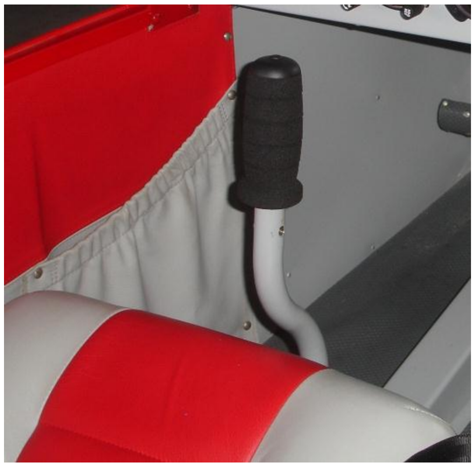


Figure 3-44. Control Stick. *Photo by Robert Gysi*

The image shown in Figure 3-34 is the actual control stick that will be simulated in the GoBosh G700S that will be arriving as part of the airframe that will be utilized on this project. From discussions with the project sponsor, we have determined that both the pilot and co-pilot sticks should be intact with the airframe and that each of the sticks will still be mechanically tied together. In all aircraft the control sticks need to be tied together mechanically (or electronically in fly-by-wire systems on large commercial airliners and military fighters) so that each pilot is able to take command of the aircraft. For the simulator we wanted to make sure that we allowed for this option as well in order to ensure that not only can this be used as demonstrator but also possibly for flight training should that need be established in the future.

In order to handle the design of the stick there are two options readily available for creating the control stick electronically. Option A is to build a custom control utilizing potentiometers in addition to having the mechanical parts fabricated, such as the sliders for each of the axes. This option also requires the use of springs in order to reset the control stick at center. A figure of this option is included below, courtesy of howstuffworks.com in Figure 3-36.

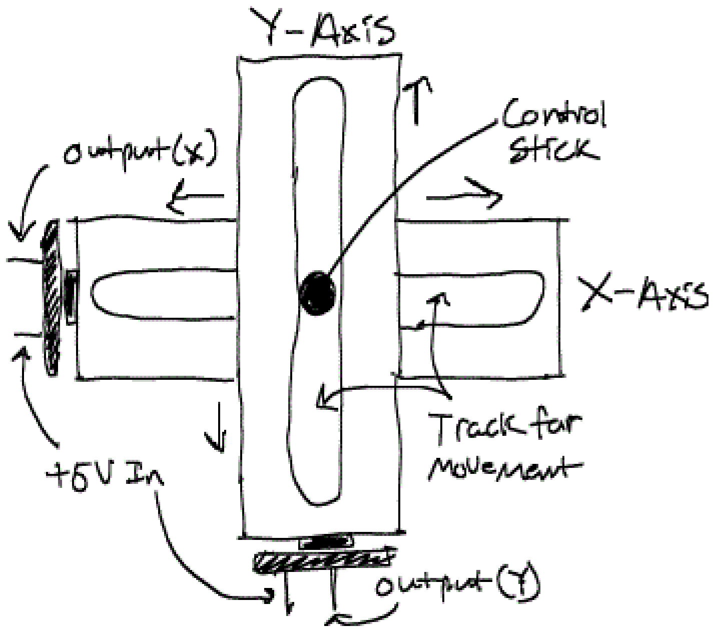


Figure 3-45. Basic Mechanical layout with potentiometers of a joystick. *Drawing by Robert Gysi*

While option A is the most basic method for designing a control stick, it however does have a problem: the need to get the shafts for each of the axes fabricated. In addition, we have no idea about where we would put such a fabricated assembly into the cockpit due to us not having seen any drawings or breakdowns of the aircraft yet. Another option we investigated included using a pre-manufactured joystick controller in order to provide us with the mechanical interface to our electrical side. 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 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[36]. 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), would 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. Due to this, it may have the ability to go 60 degrees in each direction, but it would severely limit how far you could move the stick mechanically. Finally, 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.

[36] (*2009, Nov.). SPC Multicomp STD-2607AR Product specifications and drawing [Online]. Available: http://www.farnell.com/cad/358999.pdf*

Therefore we have investigated one other method to capture our mechanical movement electrically.

Since the first two options we researched ultimately led us back to the drawing board, we started investigating ways to simply retrofit our electrical design to the existing stick. At our last meeting with the project sponsor we requested more information about the mechanical linkages between the stick and the control surfaces on the aircraft. It was stated that, upon arrival the wires leading from the stick to the control surfaces would be intact. Because of this we would be able to tap into the mechanical system. Figure 3-36 (below) shows the mechanical system as described to us by the project sponsor.

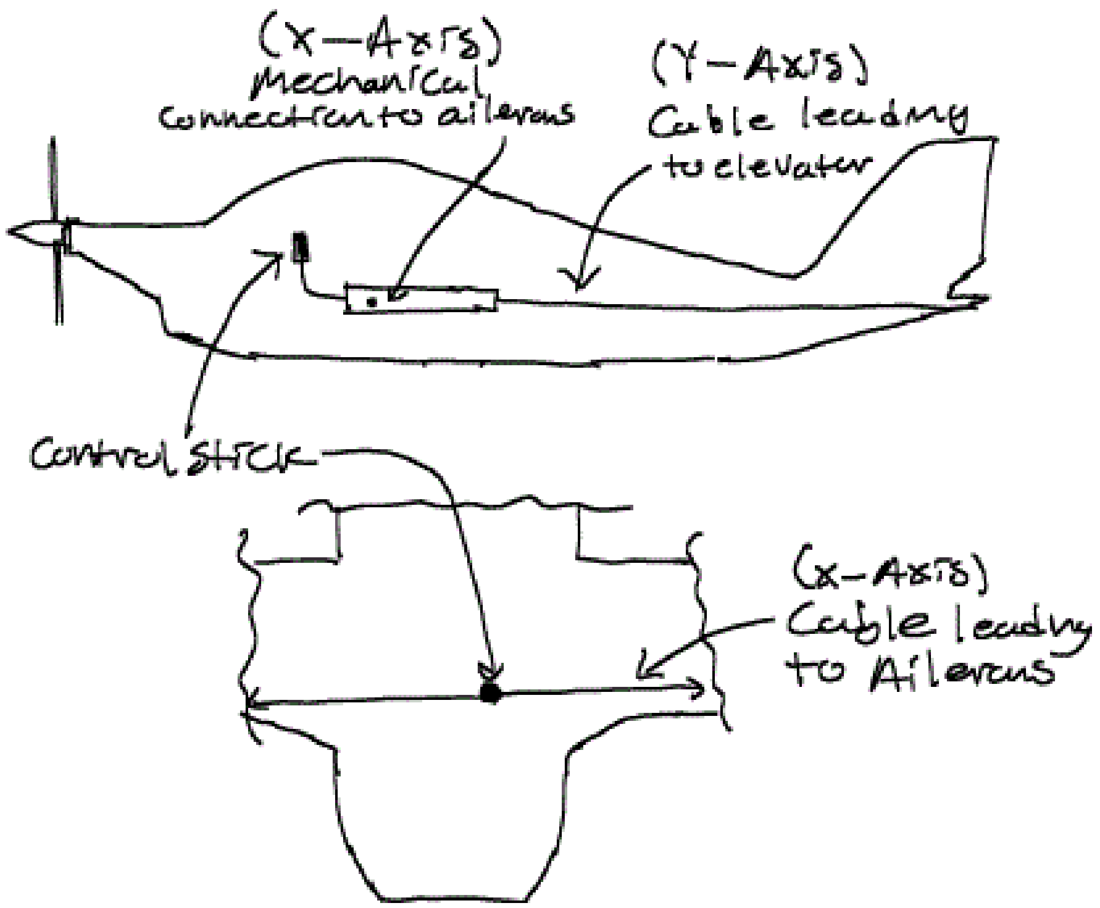

(X-Axis) Mechanical Connection to ailerons
(Y-Axis) Cable leading to elevator
Control stick
(X-Axis) Cable leading to Ailerons

Figure 3-46. Mechanical linkages for control stick. *Drawing by Robert Gysi*

With the mechanical foundation laid for the design of the control stick, we can move on to the implementation of our electrical design. For the control stick (as well as with the other flight controls), the electrical design is very basic outside of the microcontroller. We intend to utilize linear potentiometers for each of our axes. Slide potentiometers are often utilized for sliders in dimmer switches and in industrial control applications in devices such as foot pedals.

For this application we just needed your everyday basic slide potentiometer available from any number of suppliers. However, we have found one particular slide potentiometer that would work perfectly for our application due to the fact that we can modify it to connect to our mechanical control wires. The Bourns PTF01-152A-104B2 is a slide potentiometer that has 100mm of travel and a resistance value of 100kΩ.[37] The reason why we have selected this particular potentiometer is that in the drawing from the manufacturer's datasheet we find that the pole for the slider is made of metal and has a width of .236" at its widest and .157" at its thinnest. Due to this we are able to use a small drill or dremel tool to create a mounting hole in our potentiometer slider to get a hole that is .138" in diameter. This will allow us to use a #6 screw along with a corresponding nut. The drawing below, highlights the modification of the slider arm to accommodate this screw.

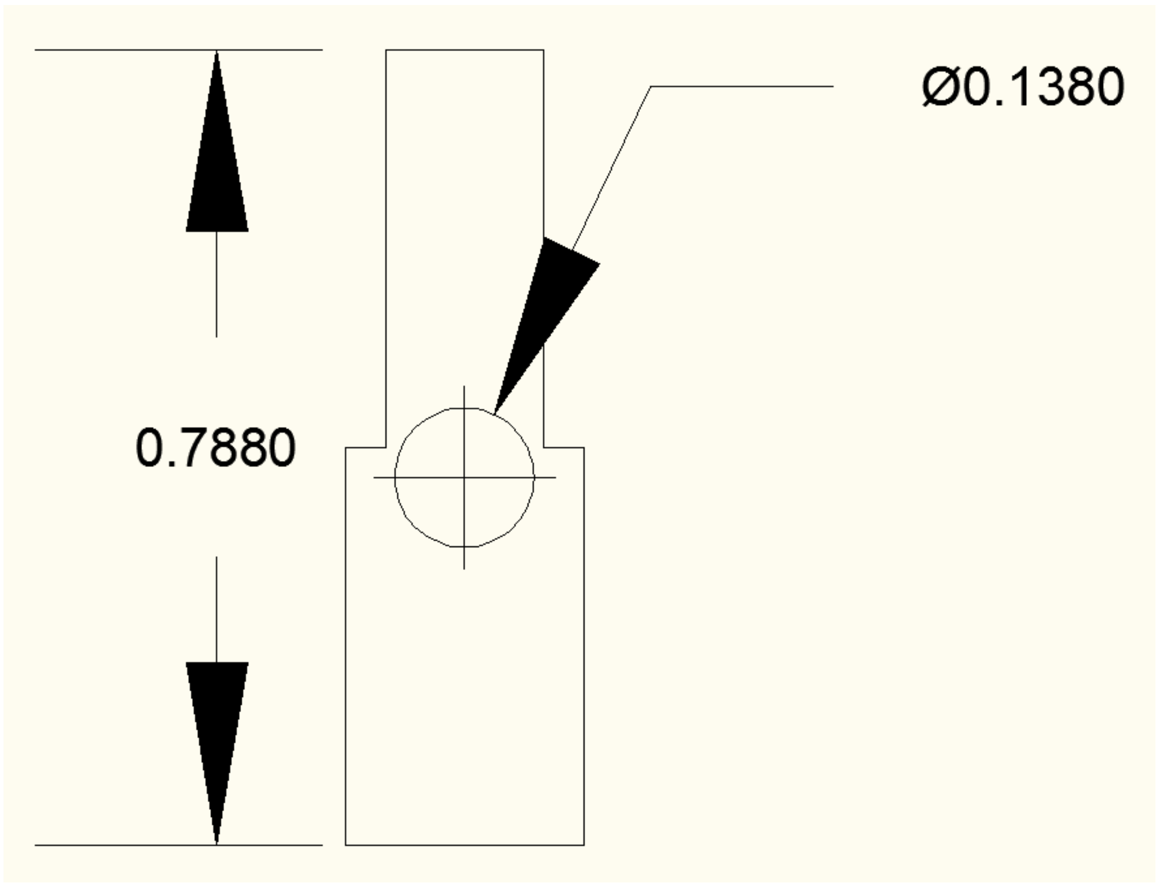


Figure 3-47. Drill Pattern for potentiometer modification. *Drawing by Robert Gysi*

After drilling this hole into the potentiometer, we will then attach a piece of aluminum or other metal angle to form the interface with the cable. We will not need to have pieces fabricated, as we should be able to pick up these small

[37] *(2009, Nov.). Bourns PTF01-152A-104B2 Specifications. [Online]. Available: http://mouser.com/ProductDetail/Bourns/PTF01-152A-104B2/?qs=sGAEpiMZZMurJeyiEZqTmzPfuWnYalgipCgRhXRNuyQ%3d*

angle pieces at surplus stores such as Skycraft. The angle piece will need to have a matching hole as the one in the figure above shows, so that the correct size screw and nut and join the two together. Additionally we require that the angle piece also have a hole for the cable to pass through as well. One such example of the an angle piece to be used that can be found in a surplus store such as Skycraft is show in the figure below. This particular angle bracket has a hole with diameter of approximately .3125". It is however, unknown what the diameter of the cable will be until we receive the cockpit.

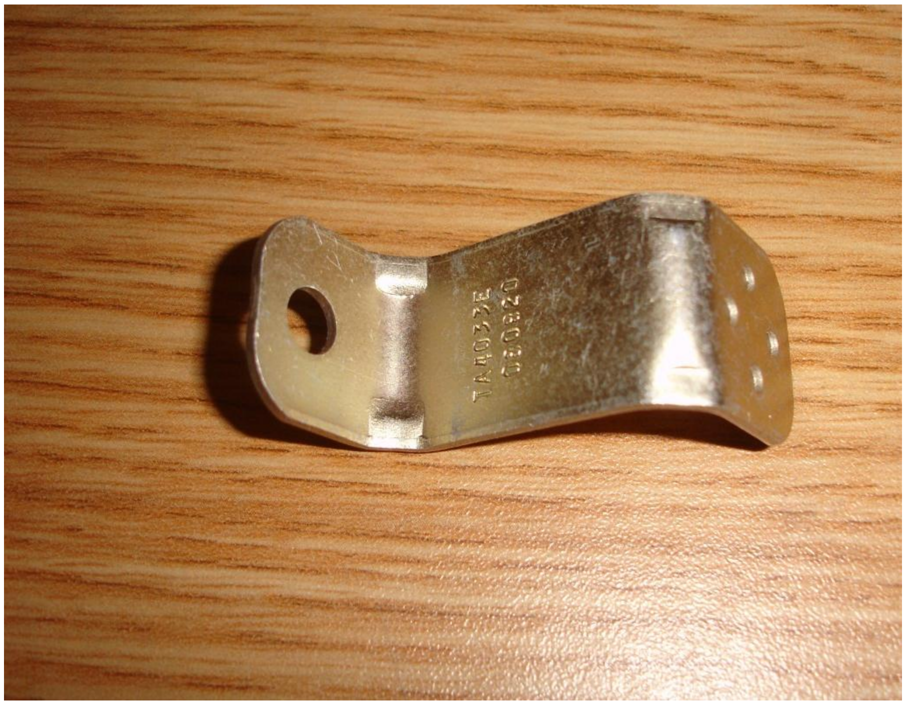


Figure 3-48. Angle bracket for connecting to flight control wire. The connecting screw to the potentiometer would be placed on the solid surface appromatel where the engraved text is located. *Photo by Robert Gysi*

Assembled the interface should allow us to capture all the movements in the cable with no restrictions on the actual movement of the stick. This should allow us to keep the feel of the actual stick in the aircraft while still capturing the analog data that we need to.

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 +12V 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 will use our PTF01-152A-104B2 slide potentiometer after it has been modified to include our connecting angle bracket and wire it up so that Pin 1 is our analog voltage input, pin 2 (wiper) is connected back to pin 1 and our analog output leaves from pin 3.As the potentiometer changes position the resistance will change which will also cause the voltage to drop. This output voltage will then feed into the A/D converter on the AT90USB microcontroller. 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. However, upon further research another chipset was found that would combine both functions into the one. The Atmel AT90USB1287 microcontroller is a USB On-the-go (OTG) microcontroller that has a built in 8 Channel 10-bit A/D converter. The A/D will take in the analog signal from the STD-2607AR Joystick control within 65-260 microseconds and write it into a defined First In, First Out (FIFO) queue. The microcontroller will also be in single conversion mode and the conversion will take place on demand, as the joystick is moved. In addition, once the conversion is complete an interrupt will be generated.

AT90USB1287 Specifications:

- USB On-The-Go (OTG) Microcontroller
- 128-Kbyte self-programming Flash Program Memory
- 8-Kbyte SRAM
- 4-Kbyte EEPROM
- 8 Channel 10-bit A/D-converter
- JTAG interface for on-chip-debug
- Up to 16 MIPS throughput at 16 MHz. 2.7 - 5.5 Volt operation
- Flash (Kbytes) = 128
- EEPROM(Kbytes) = 4
- SRAM (Bytes) = 8192
- Max I/O Pins = 48

Table 3-2 Implementation of Control Stick Analog to Digital

Function	Slider potentiometers	AT90USB1287 Port F
X Axis	Pin 2	Pin 60 (PF1 – ADC1)
Y Axis	Pin 4	Pin 59 (PF2 – ADC2)

The slide potentiometers that are positioned along each axis of movement are tied to the Atmel microcontroller’s analog-to-digital (A/D) converter with the connections listed in Table 3-2 (above). Ultimately there is no digital design necessary as the microcontroller itself is the sole digital component since there is no external A/D or digital buffers. From here software needs to be developed, but that is covered in section 3.4.4 Combined Flight Control Software Design.

3.4.2 Rudder Pedals

In this simulator we are also going to take in the input of the pedals located in the foot wells of the cockpit in order to enhance the simulation experience. In this application, the pedals will be able to be used to steer the rudder on the tail of the aircraft just as a functioning aircraft. Because we want to maintain the feel of the actual pedals we will be utilizing the existing mechanical assembly included in the cockpit we are receiving. We will modify these pedals with rotary potentiometers, and springs (to ensure that the user can simulate the feel of the toe-brakes which will not be implemented) in order to capture the control signal. In reality the design will be very similar to our joystick design in that there will be two axes termed X-Right and X-Left in order to move the rudder to the left or to the right.

Figure 3-49 Pedals on the GoBosh G700S. *Photo by Robert Gysi*

We anticipate little modification will need to be done in order to connect our potentiometers to the pedals in the cockpit. The majority of the modifications to the mechanical assembly of the pedals should only require placing the slide potentiometers on the along the cables that run to the rudder in the vertical stabilizer. These two cables, which can be seen in figure 3-39 as running through the housing in the foot well can be modified in the same fashion as the joystick was. Figure 3-41 shows the mechanical operation and layout of the pedals before modifications. The unknown component is an item that will not be present in the cockpit that we receive.

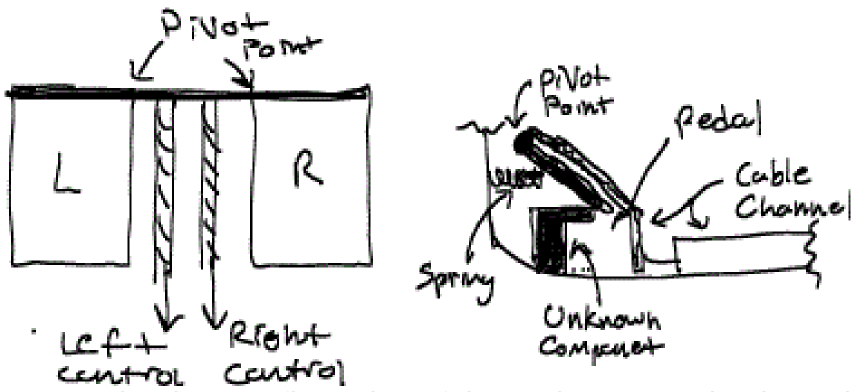


Figure 3-50. Mechanical configuration of the pedals. *Drawing by Robert Gysi.*

3.4.2.1 Rudder Pedal Analog Design

The analog electrical design is even simpler than the design of our joystick. We will receive our +12V DC signal via the computer power supply and put this voltage across the potentiometer. The change is resistance will bring about a change in the voltage which is fed into another set pin on the AT90USB1287 microcontroller.

Since the mechanical linkages are the same as on the control stick, we can take advantage of using that design here as well. We will also utilize the Bourns PTF01-152A-104B2 slide potentiometer with 100mm of travel. We will wire our +12V source from the computer onto pin 1 on the potentiometer with the wiper pin, pin 2, connected to pin 1 and our output on pin 3 will go to the AT90USB microcontroller. We will do this for both our X-Right and X-Left potentiometers.

3.4.2.2 Rudder Pedal Digital Design

Once again, because we are adding on the existing microcontroller being utilized for the yoke control, the design is very similar. The primary difference is that these two potentiometers are being tied to the next two Analog-to-Digital pins on the microcontroller. As before, the digital design is all internal to the microcontroller since the A/D conversion happens inside of the chip. For more information on the development of software for the microcontroller see Section 3.4.4 Combined Flight Control Software Design

Table 3-3 Implementation of Pedals Analog to Digital

Function	**Potentiometer**	**AT90USB1287 Port F**
X-Right	Output	Pin 58 (PF3 – ADC3)
X-Left	Output	Pin 57 (PF4 – ADC4)

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. Electrically it is the same circuit as with the joystick and pedals, except for the throttle we have only the need to have one potentiometer.

Mechanically, it is nowhere near the same as the others. Also, where we are receiving the pedal and stick assemblies with the aircraft cockpit from Europe, we will not be receiving a throttle assembly. Because of this we will need to design a simple throttle assembly using the same principles we developed for the pedals and joystick implementations. Of course instead of having a cable, we will utilize a metal rod, which will ultimately stick out of the flight instrument panel in between the copilot and pilot. Figure 3-41 shows the location of the throttle in the cockpit.

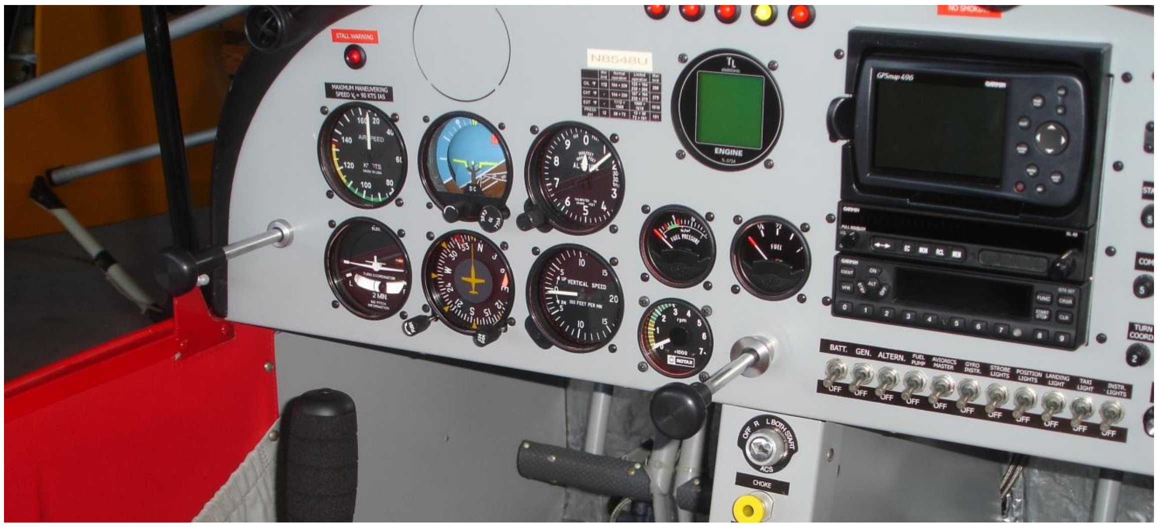


Figure 3-51. Throttle location (black knob pulled out of instrument panel). *Photo by Robert Gysi*

To begin with the mechanical design, we will start off utilizing the same potentiometer modification discussed in section 3.4.1 Joystick Design, specifically in Figures 3-39 and 3-39. The primary difference here is that instead of a cable passing through the bracket pictured in Figure 3-38, we will have a rod that is partially threaded and partially unthreaded with a length of6”. The reason for this is that we will want to secure the end of the rod to our plate so that it does not move free of the potentiometer. The AutoCAD 2D drawing below represents our design for the throttle.

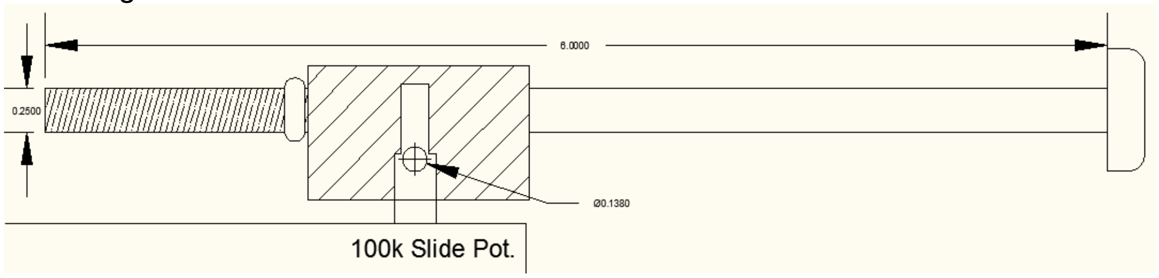


Figure 3-52. Mechanical representation of Throttle Assembly. *Drawing by Robert Gysi*

The components to build should all be able to be purchased from stores such as Skycraft. Should the rod not be available, it can be purchased from McMaster-Carr online. We will attempt to source an authentic throttle knob from an aircraft junkyard or eBay, but if not any circular knob that is black would work. The hatched box in the drawing above represents the piece of angle aluminum that is acting as our potentiometer to rod interface. Like the aforementioned brackets, this can be purchased from Skycraft as well.

3.4.3.1 Throttle Analog Design

The analog design here is the same as with the pedals and joystick, except here we will utilize only one slide potentiometer. We will also use the same input voltage, +12V from the computer power supply tied to Pin 1 on the potentiometer. Pin 2 (wiper) will also be tied to the +12 supply, and just as with before Pin 3 will be our analog voltage output. This will then go the A/D port PF5 on the AT90USB microcontroller.

3.4.2.2 Throttle Digital Design

Once again, because we are adding on the existing microcontroller being utilized for the yoke control, the design is very similar. The primary difference is that we are only feeding one analog voltage into an A/D converter. As before, the digital design is all internal to the microcontroller since the A/D conversion happens inside of the chip. For more information on the development of software for the microcontroller see Section 3.4.4 Combined Flight Control Software Design

Table 3-4 Implementation of Throttle Analog to Digital

Function	**Potentiometer**	**AT90USB1287 Port F**
Throttle	Output	Pin 56 (PF5 – ADC5)

3.4.4 Combined Flight Control Software Design

To interface with the AT90USB1287 we will need a low level section in our code to handle all of the configuration and data gathering. The following is an outline of the responsibilities and processes of our low-level control stick code:

1) Initialization
   a) Initialize the FIFO array
   b) Initialize at least two separate channels on the A to D converter for each axes
   c) Set the appropriate settings for the A to D converter
2) Stall or delay for a certain amount of time *t*
3) Use the following steps to get the location of the control stick:
   a) Send a request to channel 1 to read the data from the A to D
   b) Stall or delay for a certain amount of time *x*
   c) Send a request to channel 2 to read the data from the A to D
4) Process the values before writing to the FIFO array
5) Repeat all the steps from 2 to 4

Figure 33 illustrates this outline. The top figure is a state diagram illustrating the order in which these tasks take place. The bottom figure illustrates what system is involved during each step and also illustrates how long each process takes. This is the model we will be using to write our code.

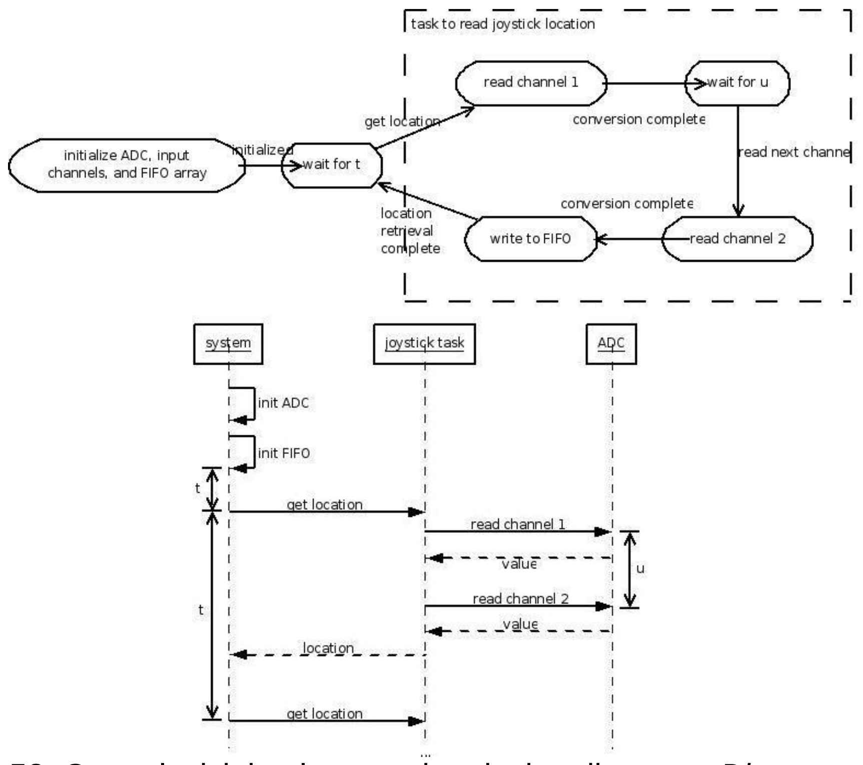


Figure 3-53. Control stick implementation design diagram. *Diagram used with permission from Mantis Cheng.*[38]

3.4.4.1 Control Software

At the higher level, we will have a period task called *read_control_stick*, which will read the control stick values every four ticks. The complete execution of the subsystem can be seen in figure 3-45. This task will be run by our higher level software to find the position of the stick. It is discussed in more detail later on.

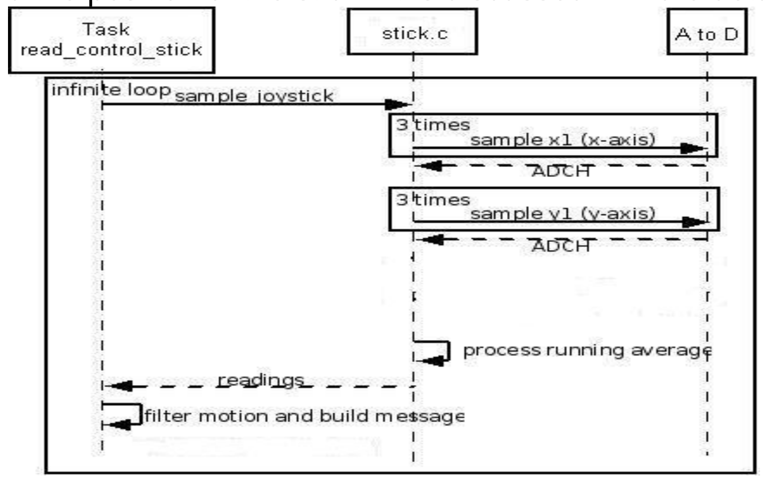


Figure 3-54. Message sequence diagram implementation. *Photo used with permission from Mantis Cheng.*[39]

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

3.4.4.2 Control Stick Driver

ll our low level driver functions for the control stick are in the *control_stick.c* and *control_stick.h* files. The following is an outline of the tasks the driver will be responsible for:

1) Control stick initial settings
   a) The input pins of the control stick (3 pins total)
   b) The settings for the A to D converter
   c) Set the initial settings for arrays running averages
2) The sampling of data from the control stick

In our control stick driver functions we need to initialize the control stick, in order to do this we will need to run a variety initialize the ports on the AT90USB1287 microcontroller. Utilizing the following code snippets used with permission from Mantis Cheng, we are able to do this[40].

First, in *control_strick.c* the pins connected to the control stick have to be set as input pins before data can be read.

```
/* Set up input pins */
DIDR0 = ~PORTF_ENABLE_MASK; // Disable unused ADC input pins
DDRF &= ~PORTF_ENABLE_MASK; // Set low for input
PORTF &= ~PORTF_ENABLE_MASK;  // Write 0 to disable pull-up
resistors
```

Snippet 3-1. Code to set input pins. *used with permission from Mantis Cheng.*[41]

Next, the analog-to-digital converter (ADC) also needs to be configured in *joystick.h* so that capturing each of the three pins can be made cleanly.

```
#define SAMP0 (_BV(REFS0) | _BV(ADLAR) | PINF1) // NOT _BV(PINFn)
#define SAMP1 (_BV(REFS0) | _BV(ADLAR) | PINF2)
#define SAMP2 (_BV(REFS0) | _BV(ADLAR) | PINF4)
```

Snippet 3-2 ADC Configuration. *used with permission from Mantis Cheng.*[42]

Macros are also defined to neatly sample the ADC and wait for the sampling to finish.

```
//turns on AD interrupt enable and AD start capture
#define TAKE_SAMP (_BV(ADEN) | _BV(ADSC))
//checks ADC state register to see if AD start capture has been turned off
```

[39] For e-mail response granting permission to use, see Appendix C
[40] (*2009, Nov.). Designing the Joystick Control CSC460 – Project Five [Online]. Available:* http://webhome.csc.uvic.ca/~mcheng/samples/cpang/joystick.html
[41] For e-mail response granting permission to use, see Appendix C
[42] For e-mail response granting permission to use, see Appendix C

```
yet
#define ADC_NOT_DONE() (ADCSRA & _BV(ADSC))
```

Snippet 3-3 ADC Configuration. *used with permission from Mantis Cheng.*[43]

To read the position of the control stick, the driver uses the *control_stick_sample* function with an array pointer to store the values. The *control_stick_*sample function then calls an internal function *axis_sample* for the two channels X1 and Y1. The A to D is signaled by *axis_sample* to sample a specified pin three times, only returning the value of the third and final sample. Each return value is stored in a temporary array after *axis_sample* is called for the two channels. The return values are then stored into an array which contains previous values. The running average is then calculated by summing all the values and dividing by the array size. The position of the control stick is then communicated as the running average. Using the average position not only avoids choppy output but it also overcomes the problem of over sampling.

The *read_control_stick* task is responsible for retrieving the values of the control stick from the control stick driver and creating a correct message packet based on the values read. The physical center of each axis for the control stick is an area defined by the transfer function. The message packet is activated with the corresponding direction and axis of movement when the control stick is moved out of the center area. This is shown in figure 35.

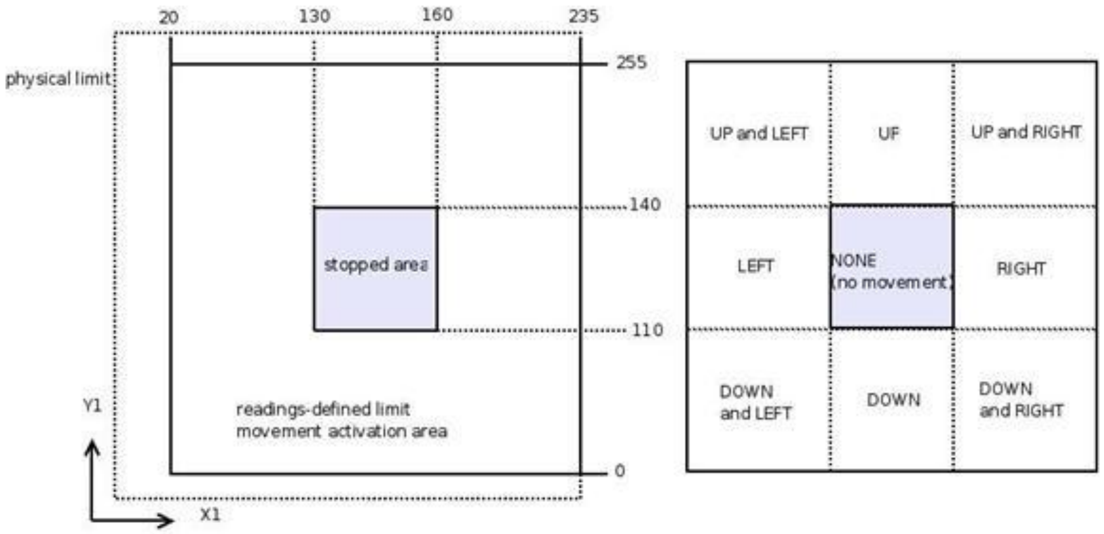


Figure 3-55: Transfer function implementation. *Photo used with permission from Mantis Cheng.*

The message packet is created and the task is completed when the control stick data is retrieved and filtered. The code for the complete control stick task is shown below:

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

```
void read_joystick() {
 uint8_t readings[3];
 for(;;){
  PORTD ^= _BV(PORTD5);
  sample_joystick(readings);
  //Trigger is "up"
  if(readings[2] == 255 && message.trigger == 0) {
   message.trigger = 1;
  } else {
   message.trigger = 0;
  }
  //x-axis is right
  if(readings[0] > 160) {
   message.x_dir = RIGHT;
  }
  //X-axis is left
  else if(readings[0]<130) {
   message.x_dir = LEFT;
  } else {
   message.x_dir = NONE;
  }
  //y-axis is up
  if(readings[1]>140) {
   message.y_dir = UP;
  }
  //y-axis is down
  else if(readings[1] < 110) {
   message.y_dir = DOWN;
  } else {
   message.y_dir = NONE;
  }
  /*Save the "speed", a difference between previous and   current positions
(could be negative currently). At the   moment this is not used by the
output station */
  message.x_speed = x_last_pos - readings[0];
  message.y_speed = y_last_pos - readings[1];
  x_last_pos = readings[0];
  y_last_pos = readings[1];

  Task_Next();
 }
}
```

Snippet 3-4. Control Stick Task. *used with permission from Mantis Cheng.*[44]

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

3.4.5 Combined Flight Control Circuit

The schematic below 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.

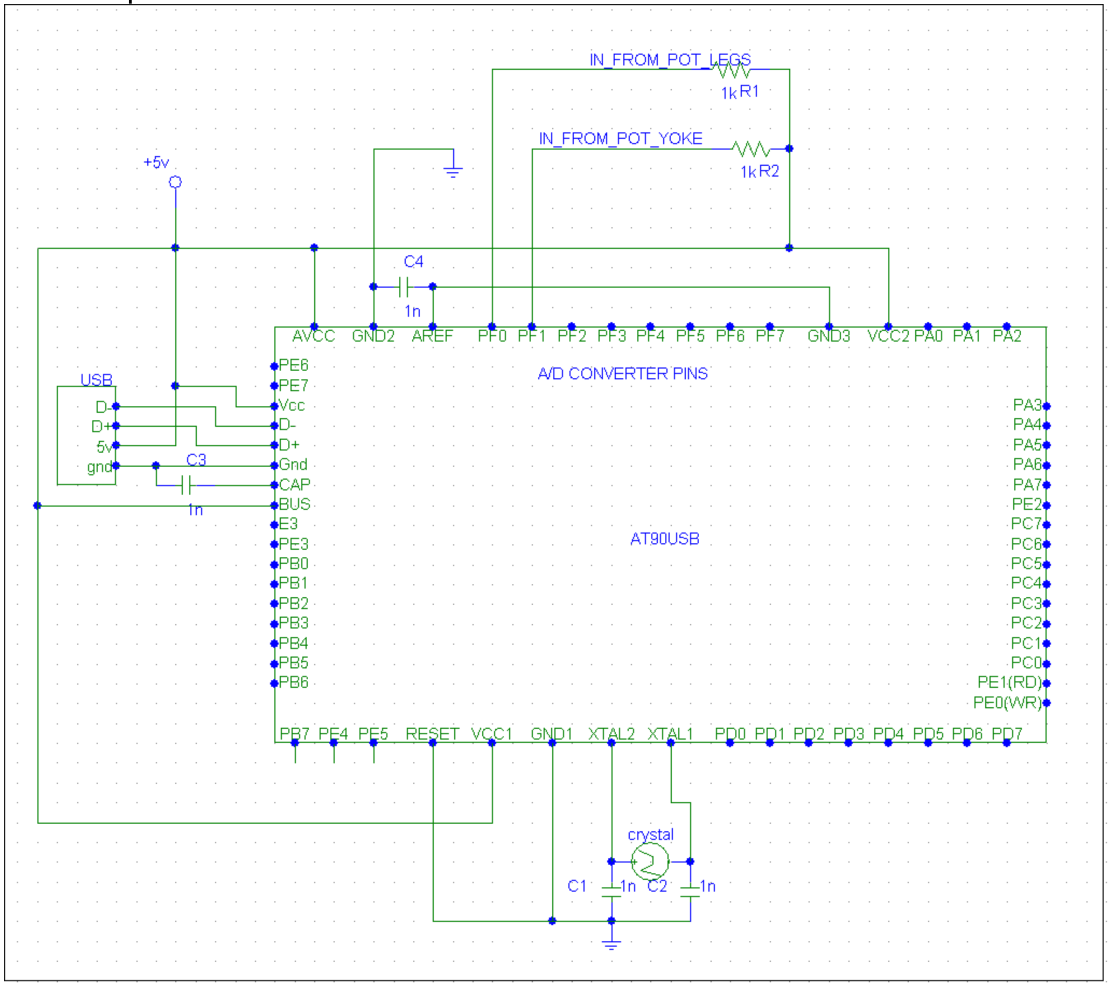

IN_FROM_POT_LEGS
1k R1
IN_FROM_POT_YOKE
1k R2
+5v
C4
1n
AVCC GND2 AREF PF0 PF1 PF2 PF3 PF4 PF5 PF6 PF7 GND3 VCC2 PA0 PA1 PA2
A/D CONVERTER PINS
USB
D-
D+
5v
gnd
C3
1n
PE6 PE7 Vcc D- D+ Gnd CAP BUS E3 PE3 PB0 PB1 PB2 PB3 PB4 PB5 PB6
AT90USB
PA3 PA4 PA5 PA6 PA7 PE2 PC7 PC6 PC5 PC4 PC3 PC2 PC1 PC0 PE1(RD) PE0(WR)
PB7 PE4 PE5 RESET VCC1 GND1 XTAL2 XTAL1 PD0 PD1 PD2 PD3 PD4 PD5 PD6 PD7
crystal
C1 1n C2 1n

Figure 3-56. Circuit implementing control systems. *Created by Chris Dlugolinksi*

Table 3-5 Pin Assignments on the Atmel AT90USB1287 Microcontoller

Function	Slider potentiometers	AT90USB1287 Port F
X Axis	Pin 2	Pin 60 (PF1 – ADC1)
Y Axis	Pin 4	Pin 59 (PF2 – ADC2)
X-Right	Output	Pin 58 (PF3 – ADC3)
X-Left	Output	Pin 57 (PF4 – ADC4)
Throttle	Output	Pin 56 (PF5 – ADC5)

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 purchase and install on their personal computer. This is even more true with X-Plane, because not only does it feature impressive graphics (especially with water and weather effects), but also due to the fact that X-Plane is also a full-fledge aerospace modeling application. Because of this, we need to ensure that our graphics and computational power is as powerful as possible and still within our budget as set by our project sponsor.

The most appropriate place to start the discussion on designing an appropriate computer starts with 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[45]	**AMD Phenom X2 550**[46]
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

[45] *(2009, Nov.). Intel Core 2 Duo E8400 Specifications [Online].* Available: *http://www.newegg.com/Product/Product.aspx?Item=N82E16819115037*

[46] *(2009, Nov.). AMD Phenom X2 550 Black Edition Specifications [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16819103680*

	Intel Core 2 Duo E8400[45]	**AMD Phenom X2 550**[46]
Heatsink Included	Yes	Yes
Price	$167.99	$99.00

So, 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[47].

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[48]. 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.

---

[47] (*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*

[48] (*2009, Dec.). ATI Eyefinity Technology Brief [Online]. Available: http://www.amd.com/us/Documents/ATI_Eyefinity_Technology_Brief.pdf*

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[49]	**XFX HD-575X-ZNFC**[50]
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

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."[51] 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

[49] *(2009, Dec.). EVGA Product Specification Sheet [Online]. Available: http://www.evga.com/products/pdf/01G-P3-N981.pdf*
[50] *(2009, Dec.). XFX 5750 Specifications [Online]. Available: http://www.xfxforce.com/en-us/products/graphiccards/hd%205000series/5750.aspx#2*
[51] *(2009, Dec.). What is CUDA? [Online]. http://www.nvidia.com/object/cuda_what_is.html*

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."[52]

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 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. In addition, the ASUS supplied BIOS leaves open the possibility of unlocking the 3rd and 4th cores as mentioned earlier in the CPU selection. This leaves the possibility of obtaining an even more powerful machine from as little cost as possible. A comparison of the two follows in Table 3-7.

Table 3-7 Motherboard Comparison

	ASUS M4A785TD-V EVO[53]	**MSI 790X-G45[54]**
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

[52] *(2009, Dec.). ATI Stream Technology [Online]. Available: http://www.amd.com/us/products/technologies/stream-technology/Pages/stream-technology.aspx*
[53]*(2009, Dec.). ASUSTeK Computer Inc. M4A785TD-V EVO Specifications [Online]. Available:* http://www.asus.com/product.aspx?P_ID=fcsXWSxnhzZE9rnR
[54]*(2009, Dec.). NewEgg: MSI 790X-G45 Specifications [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16813130249*

	ASUS M4A785TD-V EVO[53]	MSI 790X-G45[54]
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.[55] 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 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. 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

[55] *(2009, Dec.). Western Digital WD1600AAJS Hard Drive Specifications [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16822136075*

Power Supply	EP-1000SC	ePower 1000W SLI Ready ATX PSU

3.5.2 Display Projection

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 went with: 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.

3.5.2.1 Visuals

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 3.4 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).

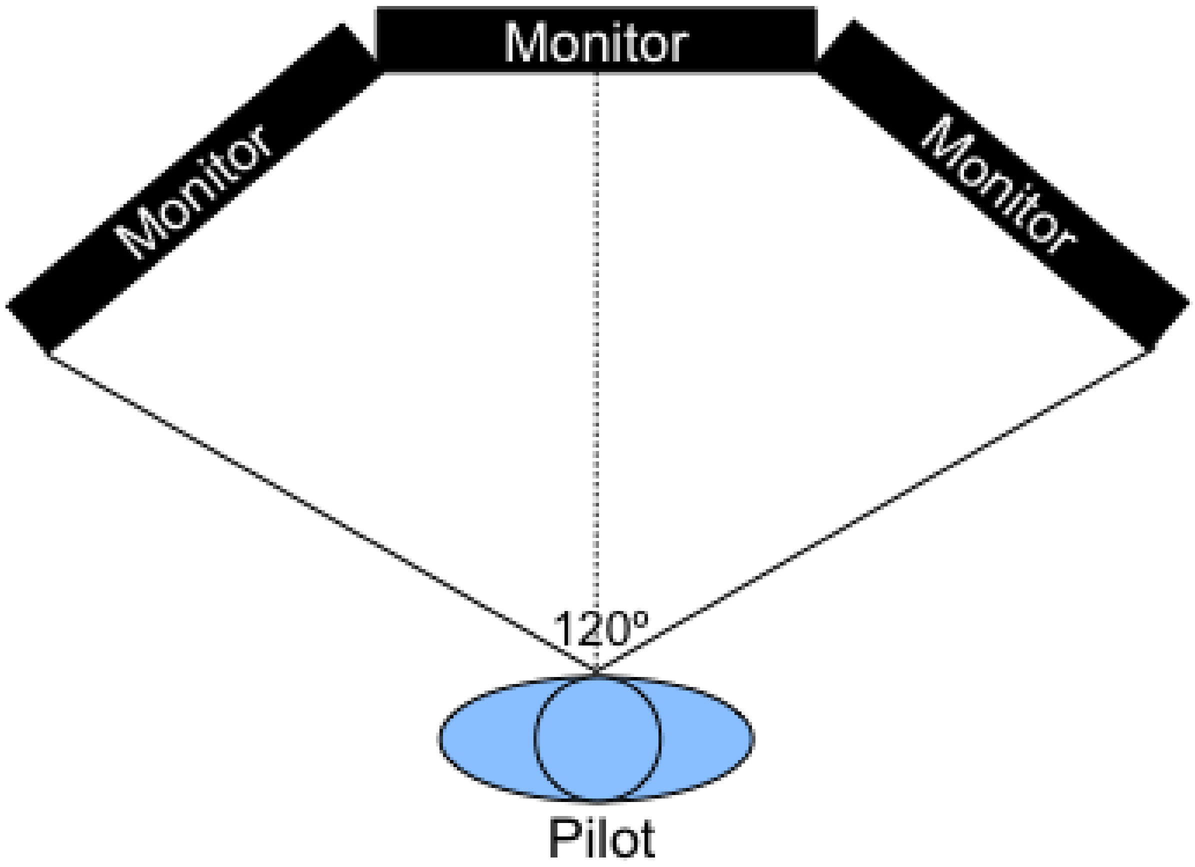


Figure 3.57: Schematic of display configuration. *Diagram by Lewis Vail.*

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

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

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

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 today18. 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 Integration with Cockpit

At the moment we are currently attempting to contact a manufacturer of VESA 100mm monitor arms in attempt to gain additional sponsorship for our project (with our project sponsor's permission) in order to potentially receive 3 monitor arms either for free or with a discount to mount on the top of the cockpit to hold our monitors and to make them fully adjustable. As of the date of submission of this document, we cannot confirm that they will make it into the final iteration of the project. Because of this we have an alternate method for mounting while the aircraft is on display at air shows and conventions.

Our alternate method to mount the displays would be to construct a platform on the above the instrument panel and bolt the monitor stands to the platform. This method would be much less flexible but it would robust so that the monitors should never get knocked out of their proper place. The most cost effective way to construct such a platform would be with a large sheet of aluminum. Two of the sides would be bent at right angles to form a table like structure. These lets will be just long enough mount to the top of the fuselage behind the engine compartment while the middle of the platform sits flat on the middle of the fuselage right above the instruments. We would use six anchor points, two for each leg and two where it sits in the middle. The point of fabricating and mounting this platform is to have a desktop-like surface for the monitors to sit on. Once this is in place it is simple to place the monitors on it and bolt them down while in use with two bolts per monitor through the OEM monitor stand. For transport the monitors can be unbolted and removed as to minimize the risk of them being damaged in transit. Additionally, o prevent any scratching of the aircraft fuselage from the aluminum we will apply a second surface to the bottom of the shelf in addition to painting the shelf in a matching color with the rest of the aircraft. This way we do not damage the simulator body and we keep a consistant theme with the external items.

3.6 Switches

Depending on how far along the project has come in the last few months of creating this simulator we may decide to add some more realistic options. 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 is the panel of switches we will be implementing if time permits. The numbered circles in the left of the picture are not switches, but are fuses and circuit breakers. As part of this project, we will not be simulating these devices.

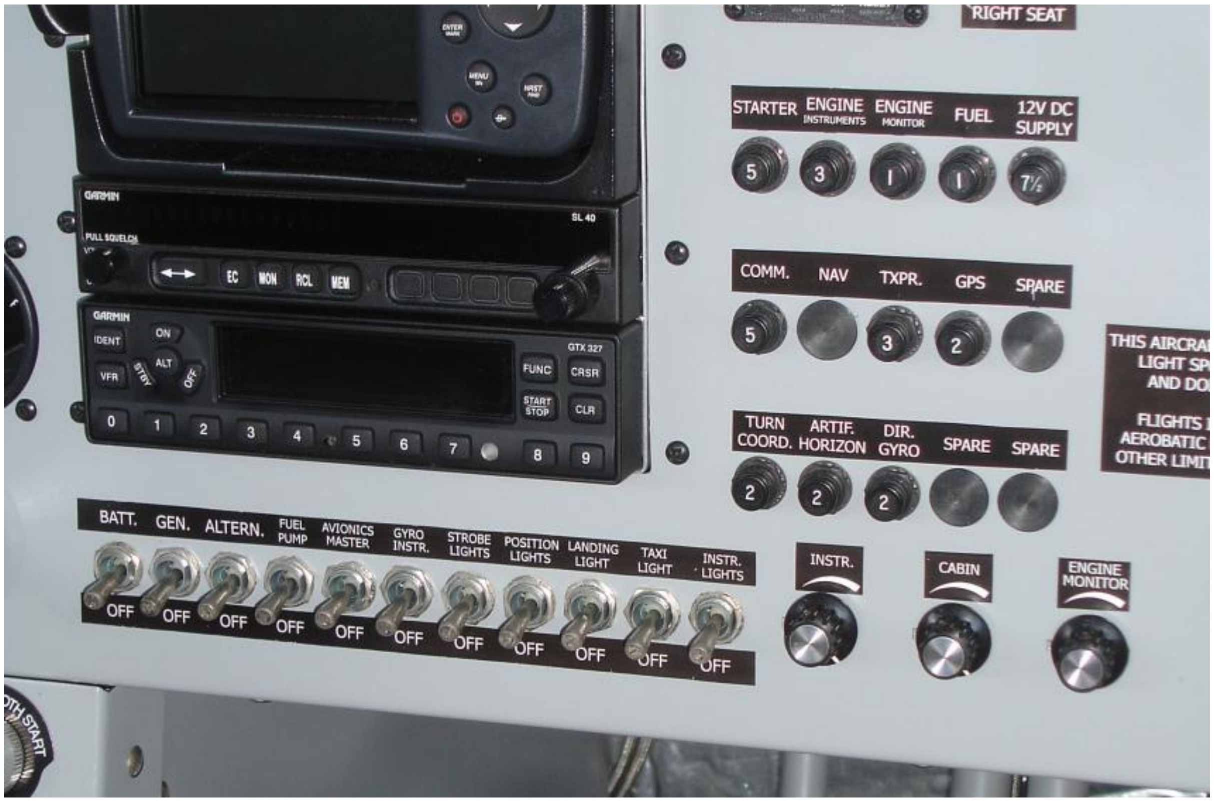


Figure 3-58. Switches and knobs that control the aircraft. *Photo by Robert Gysi*

To integrate these switches we could use some sort of multiplexor/decoder to decide which of the switches are in what position. Each of the channels can be read by the FTDI chip which only has the 8 I/O ports which will be polled by the software. This still meets with our use of the FTDI chip in Bit Bang mode. Although the polling will be easy, the actual use of the info in the simulation has yet to be seen, there is an SDK that allows us to change some of the control of the aircraft in the simulation but the Software has not been bought or developed in yet. However, the SDK variables are relatively straight forward and given time working with the SDK and learning with it, pulling information from the simulator should be relatively straight forward. A simple circuit diagram showing how the switches will be connected will be connected is shown below.

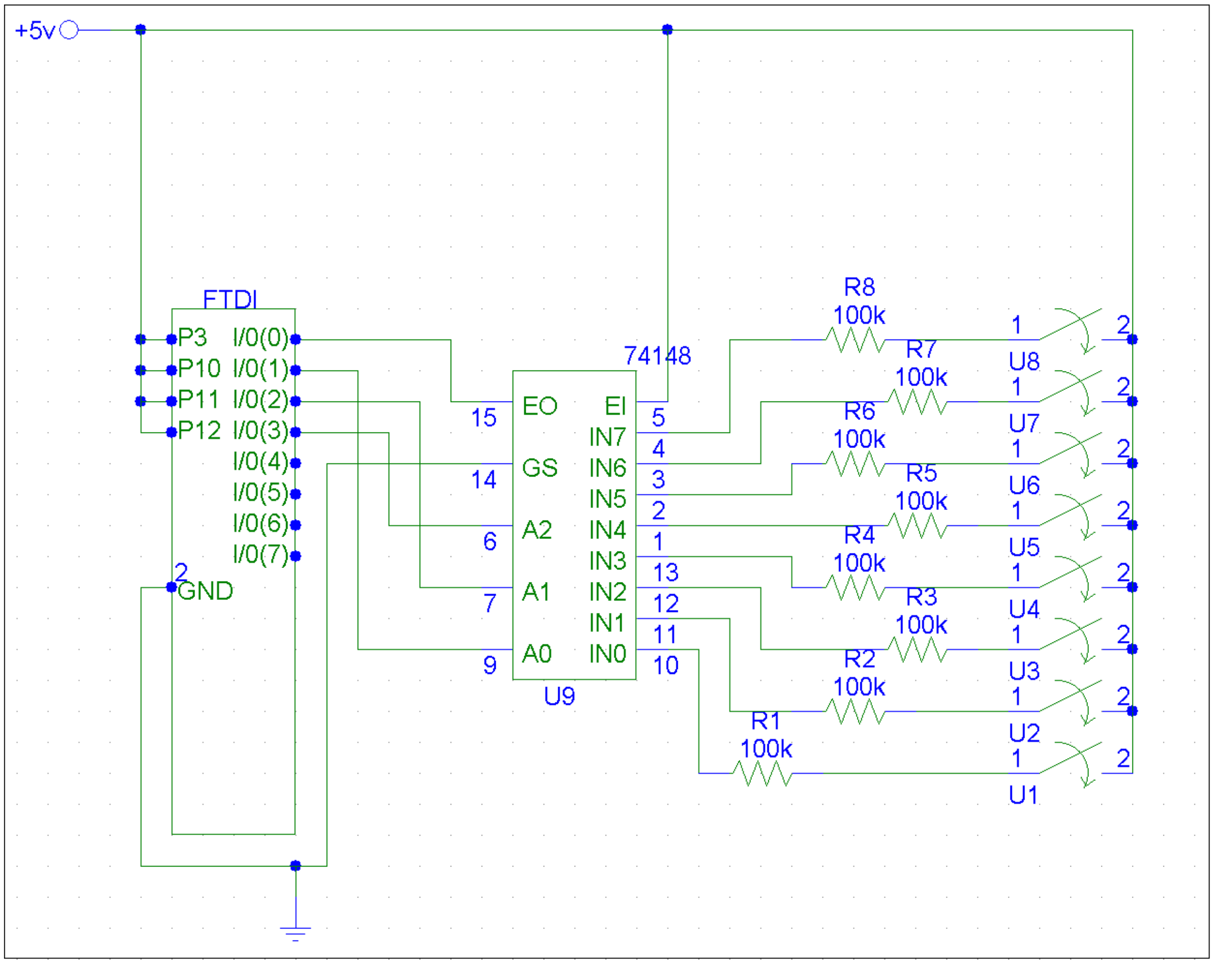


Figure 3-47. Switch circuit. *Created by Chris Dlugolinksi*

In this circuit you can see that the FTDI chip will 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 will work fine.

Each of the switches needs to be connected to a high point or the USB 5 volts or the power supply whichever we chose to go with. There will be a resistor in the circuit so that we don't draw too much current.

The software on the computer side will recognize the chip as the switch 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 will be trying to simulate the lights that are in the dash of the GoBosh aircraft. Depending on the amount of time that we have and the hardware that is available, we will try to implement as many of the indicator lights as possible. The indicator lights will be controlled once again through USB and using the FTDI chip we can 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 will allow 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 will be pulled from variables in X-Plane.

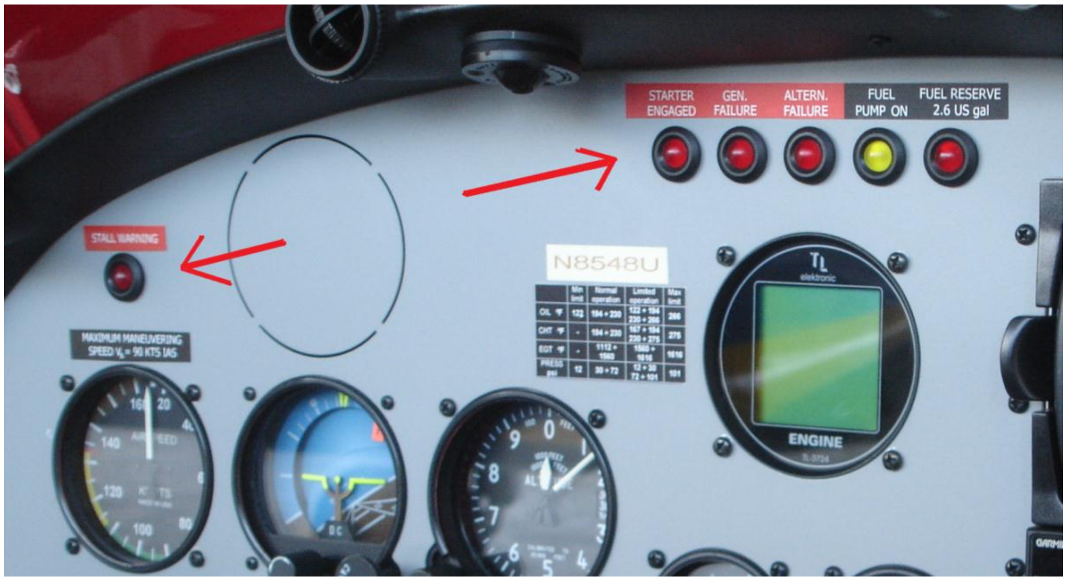


Figure 3-59. Cockpit Indicator lights. Photo by Robert Gysi

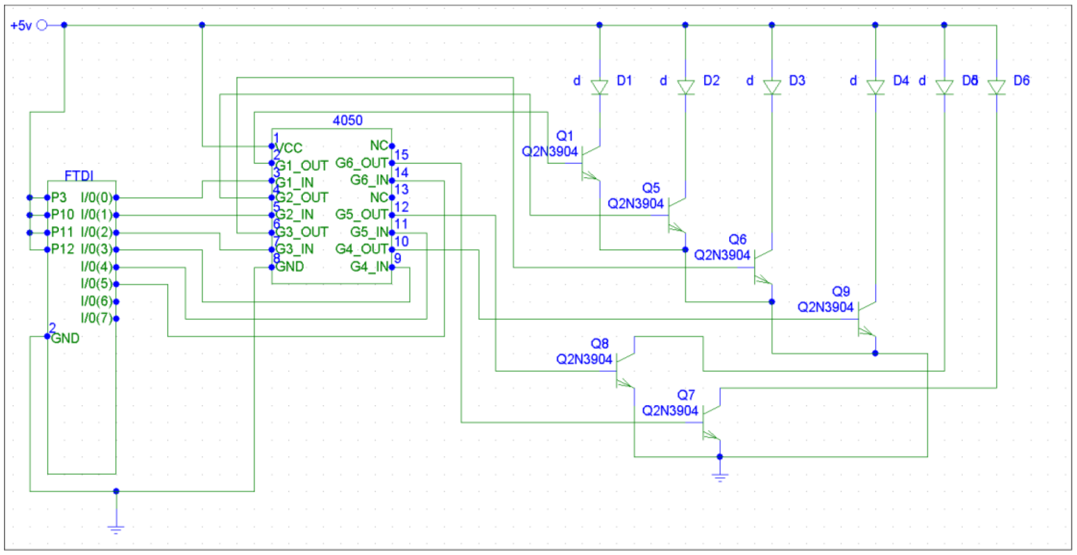


Figure 3-60. Indicator lights circuit diagram. *Created by Chris Dlugolinksi*

The figure above shows one design decision for illuminating the lights in the cockpit. 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 we will be able 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 software will interface with X-Plane via its plug-in API. This API gives us access to all of the functions and variables that we will need. What we need to decide in implementing our interfaces is how many plug-ins to use and how frequently we want to refresh our values.

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

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

After weighing all our options we decided to use two plugins for all of our devices, one for input (all the control devices) and one for output (all of the flight instruments). Figure 3.50 shows a block diagram of our high-level plugin architecture. Each device will plug into the commuter with its own designated USB cord and will be controlled by its respective plugin. The reason we chose to group them in this fashion was because the design for all of the control devices are very similar as is the design for all for all of the flight instruments. Figures 3.51 and 3.52 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 in each plugin.

Figure 3.51 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.[56] The main loop for our control devices will be as follows. For each control, we first sample the input. Most of our controls will give us a 10-bit digital signal, which will need to be truncated to an 8-bit signal so that we can use it in software. Once we have this data we will need to turn it into a format that can be assigned to one of the X-Plane variables according to the plugin API. Finally, once the appropriate data reference is populated with the new value, X-Plane will respond appropriately.

Figure 3.52 shows the interface architecture for the flight instruments. It is very similar to the control interface architecture except it is backwards. With the gauge plugin, the first step in the main loop is to look up the appropriate data reference. Next the data must be translated into something we can use to drive the gauge. Finally the appropriate value is sent out to the gauges, which will turn to display the current instrument readings. In the case of stepper motor based gauges, an aspect of this final step will be a gauge driving loop that steps the motor through the appropriate amount of iterations to get the needle in the right position.

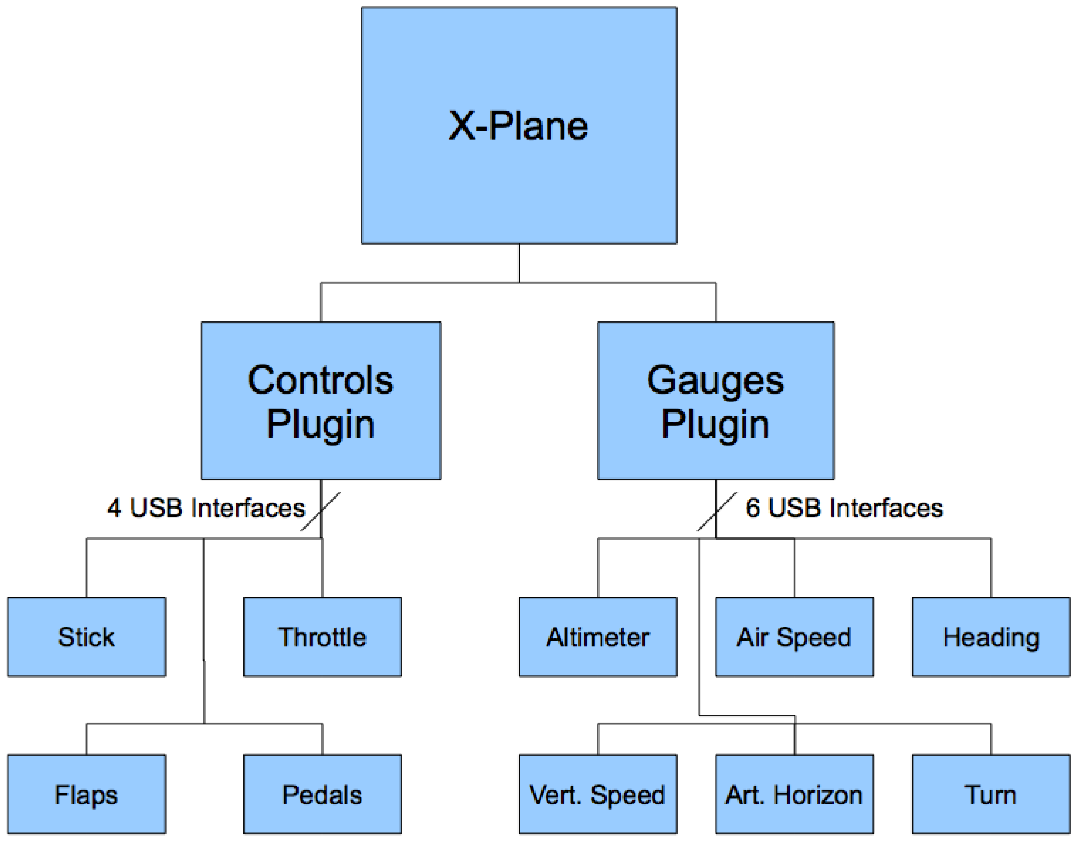
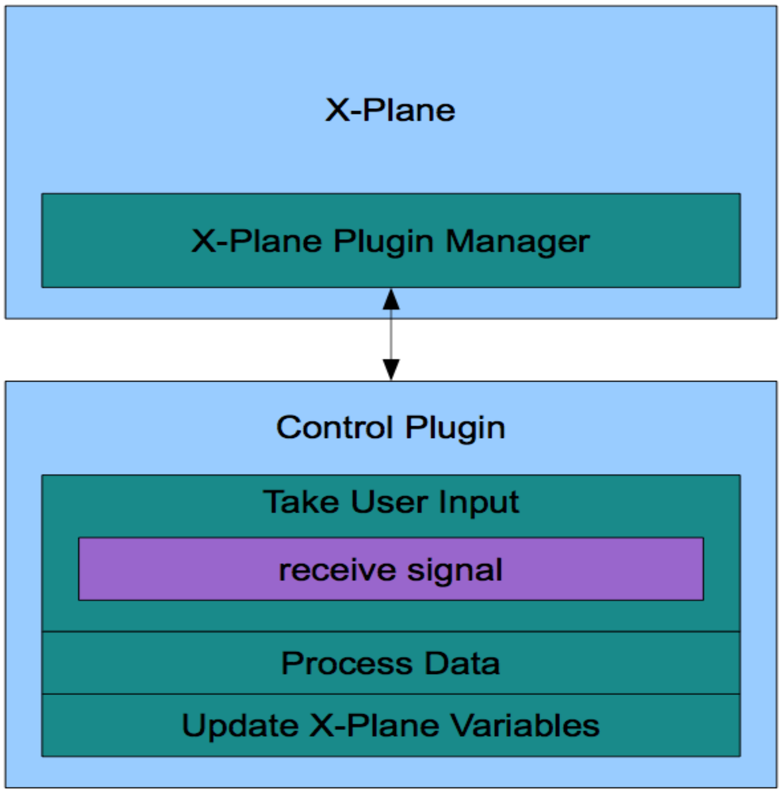


Figure 3.61: Instrument and control architecture. *Diagram by Lewis Vail.*

[56] *(2009, Dec.). X-Plane SDK [Online]. Available: http://www.xsquawkbox.net/xpsdk/mediawiki/Overview*



Figures 3.62: High-resolution control interface. *Diagram by Lewis Vail.*

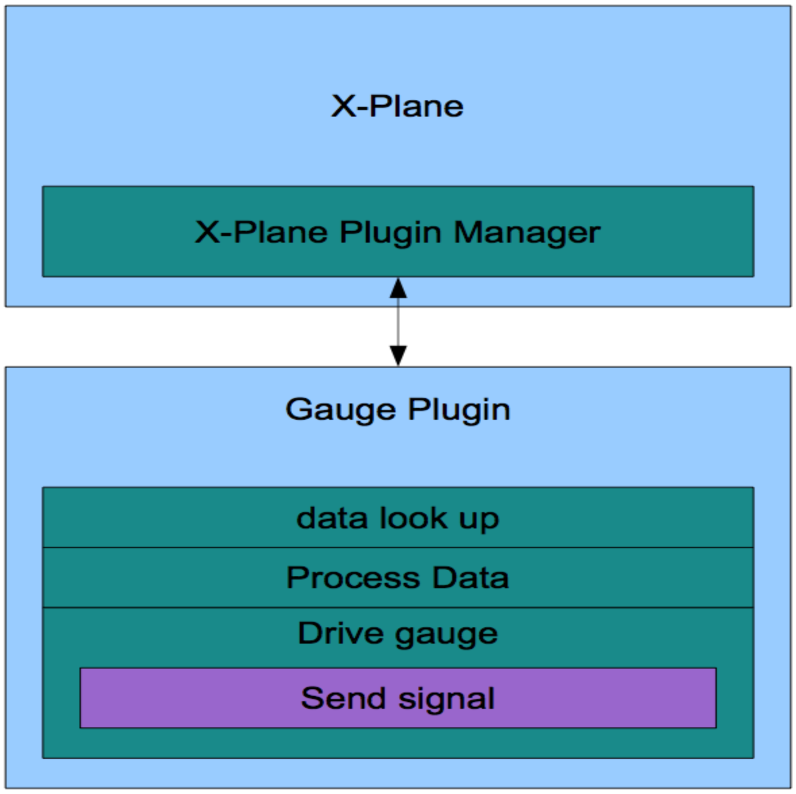


Figure 3.63: High-resolution flight instrument interface. *Diagram by Lewis Vail*

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 are trying to do we need a power supply. The computer will need a power supply that can run the graphics cards and the USB's that we have any normal supply should do. For the external power, the USB hub or any of the other circuitry that is going to be used, there needs to be another 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 design 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.

This will of course put more strain on our computers power supply but there are relatively cheap 1000 watt power supplies that have extra connectors that can supply the extra voltage and current that we will need to draw. For this we intend to utilize the ePower EP-1000SC 1000W power supply that we have included as part of our computer load-out. This power supply has an output power rating of 1000W with six 12V lines rated for 20A each which should be more than enough to power our computer hardware in addition to our gauges, controls, switches and lights. 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[57].

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

[57] (*2009, Dec.). Power Supply Unit Specifications [Online]. Available:*
http://epowertec.com/power_ep-1000p10-t2.html

Since we will only need more than likely 2-3 at most of the 12V lines available on this computer power supply for we should have plenty of voltage and current available to us from the remaining 12V lines to power each of our devices. 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 makes set up much easier. This is especially true at shows where this simulator might be displayed where there is 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 are not responsible for building a second computer to play host to the IOS functions. Instead we will us an existing computer, more than likely 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.

Chapter 4

4.1 Project Implementation Plan

Upon the completion of our Preliminary Design Review (PDR) with the project sponsor on January 3, 2010, the build phase will begin. At this time all design work will have been completed, and the only future design work to be completed will be to correct any deficiencies that arise during testing. While a project schedule has been developed for this phase of the project (and can be found in the appendixes), the purpose of this section is to layout the steps we will take to begin work on the project and ultimately prototype our completed design into a working flight simulator.

4.2 Part and Resource Acquisition

The simulator that we are building is not only a video screen and a joystick, it contains a full mockup of the GoBosh cockpit with real yoke and pedals for control it will also have working gauges and indicator lights, and switches. In order to make all this work we will need to create some of these items from scratch, which means we need to get parts for our design. The next few sections detail that part of our design.

We will order all of our required components from the distributors listed in Table 4-2 in section 4.2.1. Some of these items are stocked normally by the individual companies (such as transistors, potentiometers, and microcontrollers), but some of these items including the kit gauges we will be purchasing have a longer lead time. For the kit gauge, no matter who you order from it must come from The Netherlands, as for whatever reason that is the center of the design and production for simulation gauges available to consumers. As a result these items could take several weeks to arrive after ordering depending on the shipping method chosen. Since we are trying to minimize costs, the cheapest shipping option will be selected for all orders, domestic or internationally. Additionally, any parts that we can source locally we will from shops such as Skycraft or Radio Shack.

4.2.1 List of Required Parts

The following table lists the required parts needed to implement the design contained in Section 3 along with distributors. All components should be able to be ordered online or picked up from local retail stores.

Table 4-1. Required Parts

Item	P/N	Qty. Req'd.	Unit Cost	Total Cost	Vendor
X-Plane 9	N/A	1	$29.99	$29.99	Best Buy
Gateway 24" 5ms LCD Display	FHD2401	3	$189.99	$569.97	NewEgg
ATI/XFX ATI Radeon HD 5750	XFX HD-575X-ZNFC	2	$139.99	$279.98	NewEgg

Item	P/N	Qty. Req'd.	Unit Cost	Total Cost	Vendor
1GB					
ASUS Socket AM3 Motherboard	M4A785TD-V EVO	1	$99.99	$99.99	NewEgg
AMD Phenom II X2 Black Edition	HDZ550WFGIBOX	1	$99.99	$99.99	NewEgg
4GB DDR3 1066 RAM Kit	OCZ3G10664GK	1	$72.99	$72.99	NewEgg
Western Digital 160GB HDD	WD1600AAJS	1	$38.99	$38.99	NewEgg
Lite On IDE DVD-Rom Drive	iHDP118-04	1	$17.99	$17.99	NewEgg
1000W ATX Power Supply	EP-1000SP	1	$99.99	$99.99	NewEgg
Mid-Tower ATX Case - Black	313-06-C2228	1	$20.99	$20.99	NewEgg
Atmel USB Microcontroller	AT90USB1287	1	$15.05	$15.05	DigiKey
Slide Potentiometer - 10k, 100mm	PTF01-152A-104B2	5	$5.52	$27.60	Mouser
Servo Motor	N/A	4	$10.00	$40.00	Cent. FL Hobbies
Simkits Attitude Indicator	N/A	1	$154.00	$154.00	SimKits
FTDI USB Communication Board	FT245BL	3	$30.00	$90.00	DLP Design
USB AB Cord	N/A	3	$2.00	$6.00	
Transistors	2N2222	10	$0.29	$2.90	Digi-Key
Digital Potentiometer	AD5220	5	$1.00	$5.00	Digi-Key
Buffer	CD4049	5	$0.30	$1.50	Digi-Key

Table 4-2. Vendor Contact Information

Vendor	**Web Site**	**Phone Number**
Best Buy	http://www.bestbuy.com	407-482-8099
Digi-Key	http://www.digikey.com	800-344-4539
DLP Design	http://www.dlpdesign.com	469-964-8027
SimKits	http://www.simkits.com	+31 183 56 25 22
NewEgg	http://www.newegg.com	800-390-1119
Cent. Florida Hobbies	http://orlandohobbyshop.com	407-295-9256

Vendor	Web Site	Phone Number
Mouser Electronics	http://www.mouser.com	800-346-6873
McMaster-Carr	http://www.mouser.com	404-629-6500

4.2.1.1 Microcontrollers

We will be getting two different types of microcontroller: The AT90USB1287 which is from Atmel and the FT245BL which is from FTDI. The FTDI chip will be gotten from the company DLPDesign at http://www.dlpdesign.com/. They make premade boards for this chipset, which includes an EEProm and all the external components and connections to hook it right to a circuit and a computer. The Atmel microcontroller can be purchased from a variety of distributors, including Digi-Key, which has a development board for prototyping at a cost of $30.00 and just the chip by itself for $15.05.

4.2.1.2 USB Interface

The FTDI chip mentioned above has the capability to be used as a USB interface by simply attaching the correct pins up to USB. The chip is preprogrammed and able to be recognized by the computer over USB. As with the FTDI chip the AT90USB chip comes with support for the USB interface already in place, but unlike the FTDI chip it will need a little bit of coding to get it up and running. However, there are a plethora of Human Interface Device (HID) development white papers available from Atmel as well as an Atmel supported development community to provide resources.

4.2.1.3 Computer Hardware

We plan to purchase the majority of our computer hardware from Newegg.com. However, when we are ready to purchase our components, we do not have to purchase from Newegg particularly, but will purchase our components from the vendor that has the lowest total cost (factoring in sales tax and shipping if applicable). We also intend to utilize combination specials that online retailers such as Newegg often have in order to get a substantial discount on some of the computer hardware, in addition to regular sales. Additionally, we also will look at alternative options for our components, such as open box or recertified parts from retailers. There is a potential to save a good deal of money if the parts we desire are in one of these categories in addition to several manufacturers such as Dell are known for very inexpensive recertified, large monitors. However, with this said, a sponsorship from a particular store would probably have us purchasing all the components from that business.

4.2.1.4 Pre-Made Flight Instruments

We intend to purchase two of our flight instruments one of two companies, both of which are located in the Netherlands. The first, TRC Simulators d/b/a SimKits, manufacture kit and preassembled instrument assemblies that are available to be purchased and installed in home-built simulator cockpits. We have attempted to reach SimKits regarding an additional sponsorship or a discount as this is an educational project, but to this date have not received word back. The other company that manufactures simulated instruments is known as Flight Illusion. We have contacted them for a discount on their gauges and are in discussions with them. Flight Illusions is the more expensive brand (especially so that they list their prices in Euros compared to SimKits practice of pricing in US Dollars), but may be the better fit. Ultimately the decision with which manufacturer will be decided during a Preliminary Design Review with our Project Sponsor in early January.

4.2.1.5 Additional Parts

Any of the additional parts capacitors, resistors, transistors, and wires will be purchased from the quickest cheapest source. This includes national electronic part distributors such as Mouser, Digi-Key and Allied Electronics as well as local options such as Radio Shack and electronics surplus stores such as Skycraft in Winter Park or AstroToo in Melbourne for readily available parts. Additional supplies in personal electronics kits may also be utilized.

4.3 Build Phase

The build phase for this project will commence after our Preliminary Design Review with the project sponsor, which is tentatively schedules for January 3, 2010. After this meeting, we will make any changes to our design as requested by our sponsor and start immediately building our components. 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 the figure below.

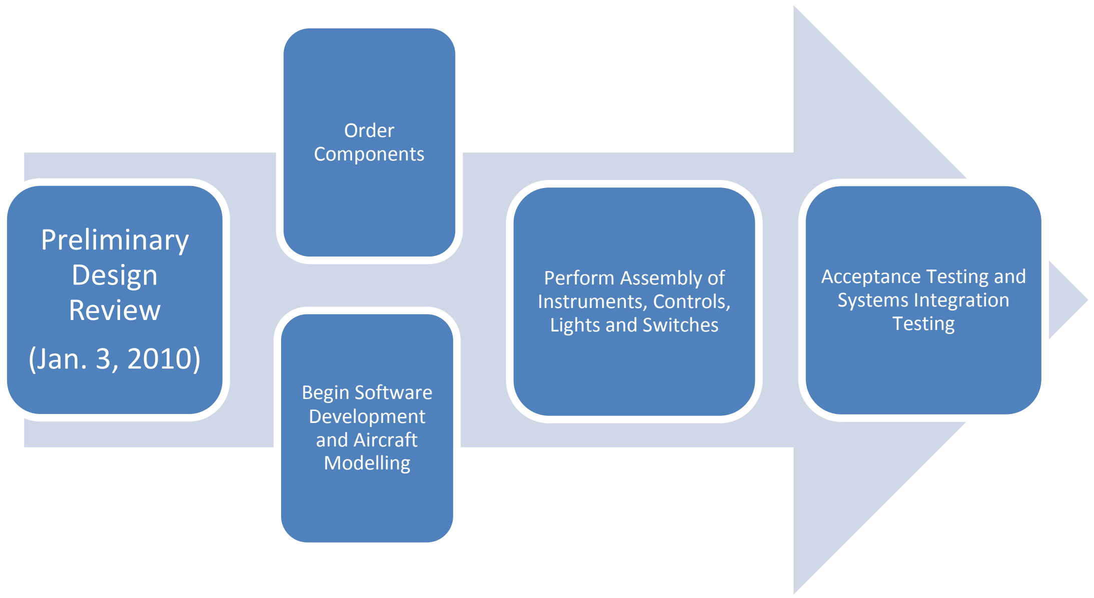

Preliminary Design Review (Jan. 3, 2010)
Order Components
Begin Software Development and Aircraft Modelling
Perform Assembly of Instruments, Controls, Lights and Switches
Acceptance Testing and Systems Integration Testing

Figure 4-1 Overview of the Build Phase

4.3.2 Flight Instrument Assembly

The assembly of our simulated flight instruments will take place upon completion of the Preliminary Design Review and we have received our initial components. The target date for beginning to assemble the gauges is currently January 27, 2010 which by then not only should we have our components to assemble the gauges, but also have any code on the microcontroller compiled and programmed into the EEPROMs of the FTDI chipsets. We also suspect that we will have a beta of our X-Plane Plug-in and we will be able to troubleshoot any issues we may have with our software or our hardware as we build.

As we assemble them we will take all necessary precautions regarding ESD and personal safety. The gauges will be constructed in the Senior Design Laboratory in Engineering I, with a minimum of two group members working at the same time. This is not only to ensure our safety but to also ensure the safety of the items purchased by our project sponsor. Also, the attitude indicator and turn coordinator gauges are being purchased as kits. These gauges will be shipped from the Netherlands and could take 20+ days to arrive. Therefore, these will be the last two gauges to be assembled. Once the aircraft arrives from Europe we will remove the instrument panel from the cockpit and bring it back to the UCF Campus, so that we can assemble the entire panel in the senior design laboratory versus having to drive to Apopka. Upon completion of the instrument panel (including the switches and lights in Section 4.3.4), the panel will be returned to the Grizzly Aviation Hanger to be installed in the cockpit. We expect that the building of gauges will take until February 19, 2010 which at that time we will be able to test our gauges by the procedures established in Section 5.4.2. We anticipate the reinstallation of the instrument panel to occur before the week of spring break. This will be the last major component to be installed outside of the computer monitor support. Upon completion of this assembly and installation, the Integrated Systems Testing can begin.

4.3.3 Flight Control Assembly

Unlike the flight instruments we have designed and will construct in the spring semester, the flight control assemblies will not start until after the arrival of the aircraft from Europe arrives. Also due to the fact that we are merely retrofitting electrical components to the existing mechanical assemblies present in the aircraft, the work will more than likely take place at the Grizzly Aviation Hanger at the Orlando-Apopka Airport. It is due to this that assembly will take place when the project sponsor has availability.

While we may not be able to fully implement our flight controls in due to the delayed start, it does not mean that we will simply not start implementing the design at the beginning of the Spring semester. To the contrary, since the electrical components are nothing more than slide potentiometers, we can verify the electrical design in the senior design lab before attempting the installation. At this same time we will be also working on the software required for the Atmel AT90USB1287 microcontroller along with a plug-in (if X-Plane requires this additional step, although it should be recognized by Windows and X-Plane as a game controller), so that we can verify the software implementation and so that when the cockpit does arrive, we will be able to install the components in a short amount of time. Our schedule, located in Appendix B, has the flight controls being installed in the aircraft around February 24, 2010.

4.3.4 Indicator and Switch Assembly

The indicator light board and switch board will be two relatively easy boards assemblies to build once the build phase has commenced. These components have also been labeled as an optional requirement for this project. For this reason, and due to the simple nature of these circuits, we will perform construction on these two assemblies last. This should occur in the two weeks before spring break, and once completed the panel will be installed in the cockpit as previously mentioned.

Chapter 5

5.1 Overview

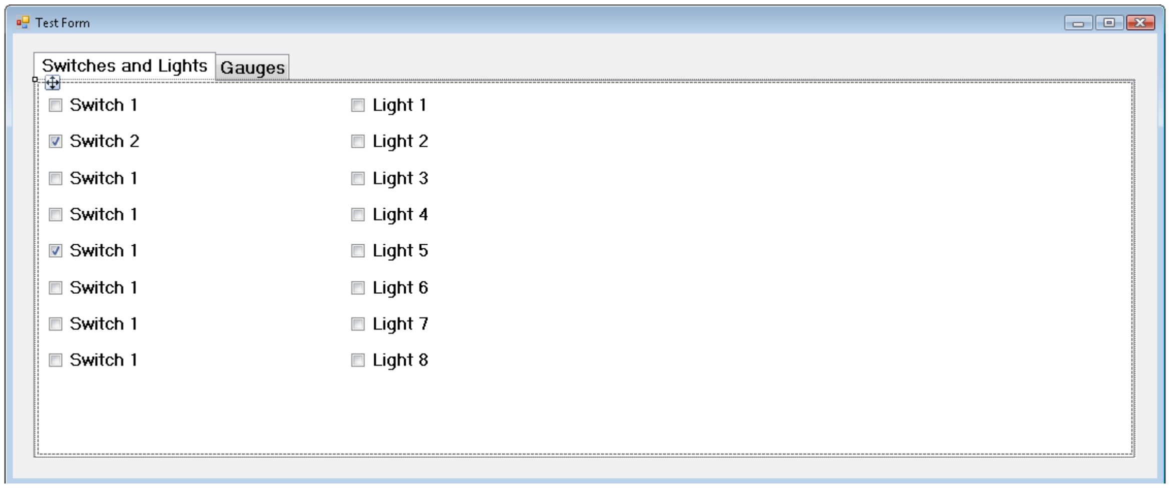
This section contains all of our test procedures for testing the individual components to be installed in the aircraft cockpit as well as the final test procedure to ensure that the system as a whole works correctly. This is critical as our project ultimately will wind up in the hands of our project sponsor, and this an excellent method for us to perform quality control on our components. This should also remove any guesswork should a component fail at the presentation at Sun ‘n Fun in Lakeland or during the in-class presentations. In addition to test procedures, 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.

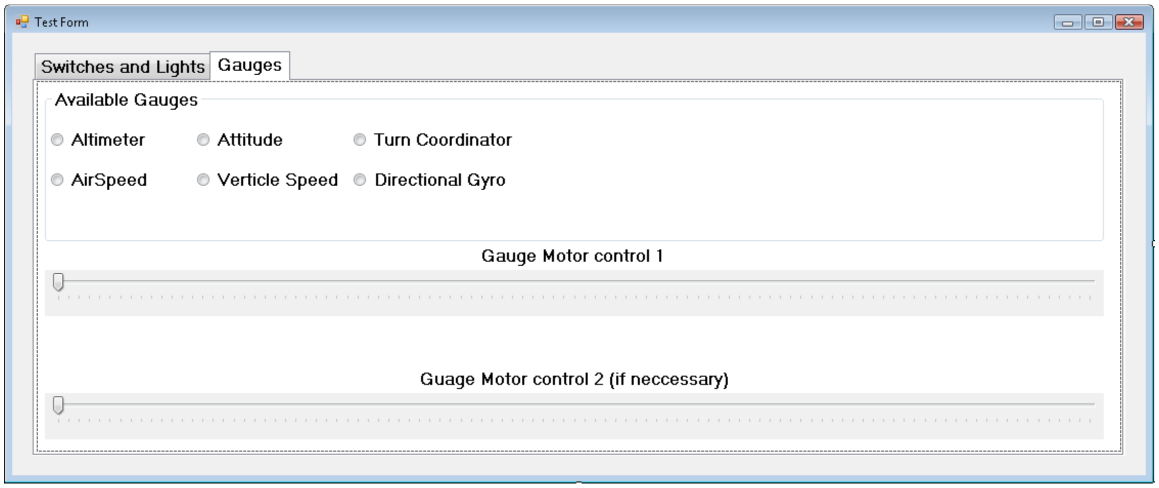
- PC running Windows 7 Professional.
- Latest version of X-Plane (currently version 9.4).
- Digital Multimeter (to troubleshoot any electrical issues that may arise).
- Computer screwdriver set (for making adjustments to mechanical components if necissary).
- Second computer (such as a laptop) for running 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. The two figures below represent the screens of this testing application.



Switch 1 | Light 1
[✓] Switch 2 | Light 2
Switch 1 | Light 3
Switch 1 | Light 4
[✓] Switch 1 | Light 5
Switch 1 | Light 6
Switch 1 | Light 7
Switch 1 | Light 8

Figure 5-1. Switches and Light Test Tab



Available Gauges

Altimeter · Attitude · Turn Coordinator
AirSpeed · Verticle Speed · Directional Gyro

Gauge Motor control 1

Guage Motor control 2 (if neccessary)

Figure 5-2. Gauge/Motor Test Tab

5.3 Test Locations

Testing will occur at several locations during the build and test phase of this project. All acceptance testing will be completed in the senior design lab in room ENGR 456. This includes the testing of our parts as we receive them and as we finish constructing individual parts. The exception to this will be the flight controls, as we will not have the cockpit in the senior design lab. Those tests along with the integrated systems testing will take place at the Grizzly Aviation Hanger located at the Orlando-Apopka Airport in Apopka, FL. Some additional testing may be completed at other locations as necessary. However, this will be minimized as the senior design lab and the Grizzly Aviation hanger are more appropriate test sites for components and for the overall simulator.

5.4 Acceptance Testing

The purpose of the acceptance testing is to verify that as we complete building each component we can 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 cockpit of the GoBosh G700S. This is 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. However, note some devices such as the joystick and the pedals will be tested while installed in the aircraft. This is due to the mechanical nature of these components, and that these components will be pre-installed in the cockpit minus our electrical modifications.

5.4.1 Part Testing

This section is to verify that the critical components that will be installed in our instruments and controls perform as specified from the manufacturer. We will require this of our most critical components: the microcontrollers, motors, and lights and switches. The microcontrollers need to be tested before they are installed onto a board as a faulty chip will not only cost us the price of the chip, but also the cost the board and other associated components installed on the board.The procedures for each of these parts are located in the following sections.

5.4.1.1 Microcontroller

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.	

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.

5.4.2.1 Airspeed Indicator

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 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

5.4.2.2 Altimeter

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. 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

5.4.2.3 Attitude Indicator

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

No.	Testing Action	Result
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 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

5.4.2.4 Turn Coordinator

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

5.4.2.5 Heading Indicator

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 accordance with the schematic diagram.	P / F
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	

No.	Testing Action	Result
	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

5.4.2.6 Vertical Speed Indicator

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 accordance with the schematic diagram.	P / F
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

5.4.3 Flight Controls

This section of the acceptance testing will cover the testing of our flight controls to be installed in our cockpit. Each step must result in a "pass" with any deficiencies noted for correction. It is suggested that before and after transport of the simulator to both Sun 'n Fun and the University of Central Florida that these procedures are checked again to ensure that there was no damage in transport and that any repairs that need to take place occur before presenting the simulator. Should a flight control not pass inspection the deficiency will need to be corrected or part redesigned before the simulator can enter the integrated systems testing phase

5.4.3.1 Joystick

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 are soldered correctly and that the wires lead to the correct pins on the AT90USB microcontroller 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	

No.	Testing Action	Result
	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

5.4.3.2 Throttle

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 AT90USB microcontroller board 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	

No.	Testing Action	Result
	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
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

5.4.3.3 Pedals

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 pins on the AT90USB microcontroller board 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

5.4.4 Cockpit Switch and Indicator Circuit Testing

The switches and indicator lamps circuits will be tested to the same level as all other flight instruments and controls. 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. Nonfunctional switches will be noted, so that they can be excluded from testing. It is expected each indicator lamp will be implemented and can be tested. Due to the possibility of a lamp burning out it is recommended that this procedure be performed before transport to Sun ‘n Fun and to the University of Central Florida in April.

5.4.4.1 Indicator Lamps

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

5.4.4.2 Switches

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.	

No.	Testing Action	Result
	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

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 ultimately the Sun 'n Fun convention in April 2010. In the integrated systems testing, each component will be tested individually in a large scale test event using the acceptance test procedures. It is recommended that for this test there is one individual testing the components, one verifying that the information on the screen of the simulator computer matches the physical components and that the Instructor Operator Station running on the networked computer matches the other two as well.

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	

No.	Testing Action	Result
	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. Immediately upon boot a batch file should run prompting X-Plane to start. If X-Plane does not start automatically, verify that the batch file was set to run on boot. 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 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	

No.	Testing Action	Result
	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.	Perform System Shut down. Exit X-Plane to the Windows desktop. At this point the batch file used to launch X-Plane will begin a system shut down. Verify that the system shuts down with no issues.	P / F
6.	Restart the system. Perform steps 3 through 5 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
7.	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

5.6 Prototype Use Cases

The simulator being developed as part of this project will ultimately be used as a demonstrator at the Sun ‘n Fun airshow and aviation conference at Lakeland Linder Regional Airport in April 2010. This simulator will 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 will 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 is 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. There is no additional design work to be done in order to implement this usage scenario.

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. The idea is that a complete function simulator will prove to the individual that it is relatively easy to get started.

It really comes down to is that anywhere you want to demonstrate either the qualities of the aircraft for sales purposes or to introduce an individual to flight training it really comes down to the fact that this simulator is really a demonstrator. In any situation where Grizzly Aviation would like to hold a demonstration this prototype would do well given their goals and objectives for the end use of this project.

5.7 Requirements Verification

After completing the test procedures and certifying that our project was built to our specifications and schematics, we need 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 will be verified that every requirement has been met. If any of the requirements has not been meet (outside of an optional requirement), then corrective actions will need to take place before the simulator is taken to Lakeland for Sun ‘n Fun or presented at the University of Central Florida for the end of semester presentations and grading. 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

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.	
S1	A	Realistic Scenery: scenery has a natural feel and does not look jaded or ragged. Terrain meshes are of high enough	

Req. #	Sub. Req.	Requirement Description	Result
		resolution to navigate from the air.	
S1	B	Inclusion of Airports Worldwide: Ensure a wide variety of airports are installed.	
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.	
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.	
S3	-	Model Entertainment Aspects	
S3	A	Weather Effects: Ability to have a wide range of weather scenarios in X-Plane including rain, snow, wind, sheer effects, turbulence, lightning and strong waves in the water.	
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.	
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	
S3	D	Ability to create custom scenarios/missions: X-Plane has tools to create and save custom missions.	
S3	E.	AI Aircraft also utilizing airspace and airports: Available via 3rd party plug-ins and custom development using the SDK.	
S4	-	Aircraft Model	
S4	A	Aircraft Exterior Model: Complete and generated via the Plane-Maker tool.	
S4	B	Model parametric data: Data received and implemented from the aircraft manufacturer or other source	
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.	

Req. #	Sub. Req.	Requirement Description	Result
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 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.	
S6	A	FAA Certification – Optional Requirement: Ability to be implemented with a $500 key. The ability to is the requirement, not the implementation.	
S7	-	Ability to interface controls/flight instruments: SDK to write control plug-ins for flight instruments and flight control s	
S8	-	Ability to interact with an Instructor Operator Station: Includes built in IOS or 3rd party applications.	

5.7.2 Hardware Requirement Verification

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.	
C2	-	120 degree field of view: Ability to in X-Plane as well as with chosen graphics adapters and monitors.	
C2	A	3 LCD monitors: Must be no smaller than 24” and secured to the fuselage of the aircraft.	
C2	B	Graphics Card/Adapter: Powerful enough to output required resolution to 3 monitors with a resolution of approximately 1920x3240.	
C3	-.	2Ghz 64-bit CPU (minimum): Established through X-Plane requirements.	

Req. #	Sub. Req.	Requirement Description	Result
C4	-	4GB RAM: Sufficient memory to run both Windows 7 Professional as well as the flight simulator.	
C5	-	120GB Hard Drive (minimum): X-Plane requires around 72GB for a full install, and Windows 7 requires 20GB. 160GB recommended.	
M1	-	USB Controlled: Has USB on the chip with little development required to implement computer communications	
M2	-	20ms refresh rate (minimum)	
M3	-	Use less than 5V to power the actual chip. Devices connected to the chip may use other values.	
M4	-	Minimum 8 I/O Pins for external communications	
M5	-	Fit inside of a 2.5" diameter tube: For the aircraft gauges and alongside the flight controls.	
M6	-	Low Cost Microcontroller: Including not only the chip but also the development board.	
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.	
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	
F2	-	Realistic flight instruments and controls: Flight controls are to be original, instruments should be as close to original manufacture specification as possible.	
F2	A	Gauges: Standard Six-Pack has been implemented - Altimeter, Airspeed Indicator, Attitude Indicator, Turn Coordinator, Heading Indicator, Vertical Speed Indicator. Ensure each gauge	

Req. #	**Sub. Req.**	**Requirement Description**	**Result**
		matches or closely matches the actual gauge utilized in the G700S cockpit.	
F2	B	Flight Controls (Stick, Pedals, Throttle): Using existing controls from the GoBosh G700S to preserve realistic look and feel.	

Chapter 6
6.1 Summary
At this point in the project we have finalized all our design work but have yet start any production. Within the next few weeks we will begin to write our software and order our parts for 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 or 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. Although we know the logical flow of our interface software, we still need to began writing the actual code. Example code is available on the SDK website and should be a great resource in the upcoming weeks.

On the hardware end, we have looked at various parts for various applications and have decided which parts we want to use for what components. We've decided to use FTDI boards for all the gauges and other simple devices and to use AT90USB1287 microcontrollers for the control devices that generate an analog signal. We have already tested and written basic board interface code for the FTDI boards and will be ordering some AT90USB1287 microcontrollers soon to begin testing them. For the gauges we have explored every conceivable implementation and decided to use stepper motors exclusively (with the exception of the two kit gauges we will be using). Furthermore, we have already laid out a high level circuit design for our stepper motor gauges. We still need to acquire the proper motors and assemble and test them and we plan on accomplishing this at the first of the year.

We have complete every step of the design phase and will be plowing forward with the build phase during the break. We fully expect to meet our April 1 deadline.

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 parts
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
2.	Detailed Realistic Scenery	1.A	Yes	Yes[1]
2a.	Accurately detailed major cities and	1.A	Yes	No

No.	Item/Description	Req. No.	FSX	X-Plane 9
	landmarks			
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]

No.	Item/Description	Req. No.	FSX	X-Plane 9
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.

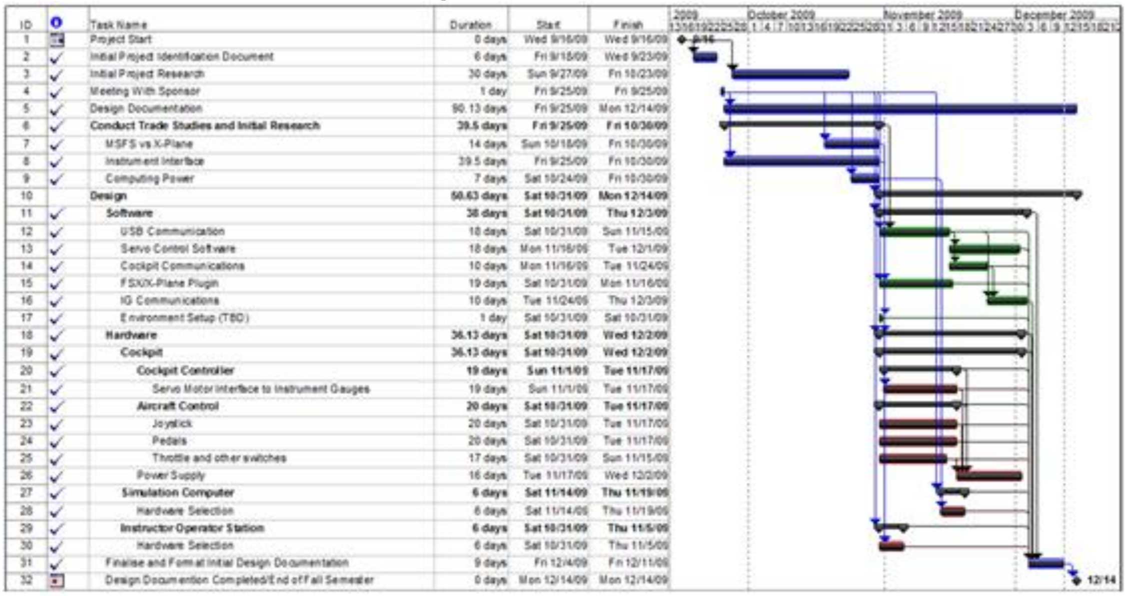
Summary:

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.

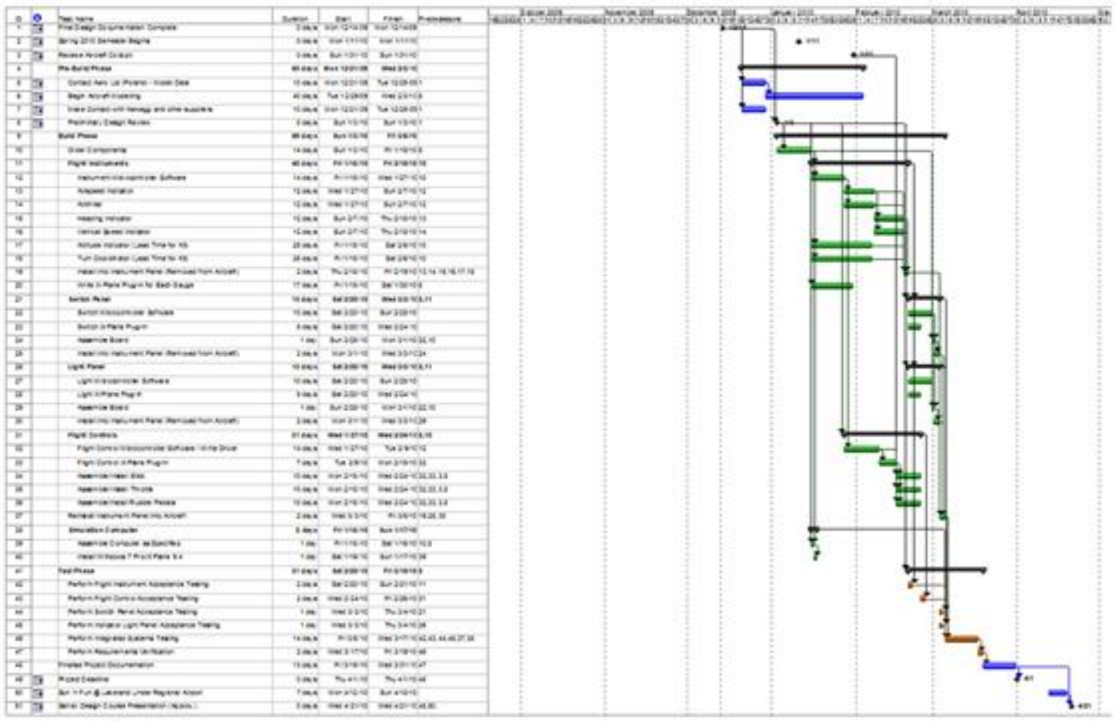
Appendix B: Project Schedules and Monthly Status Reports

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

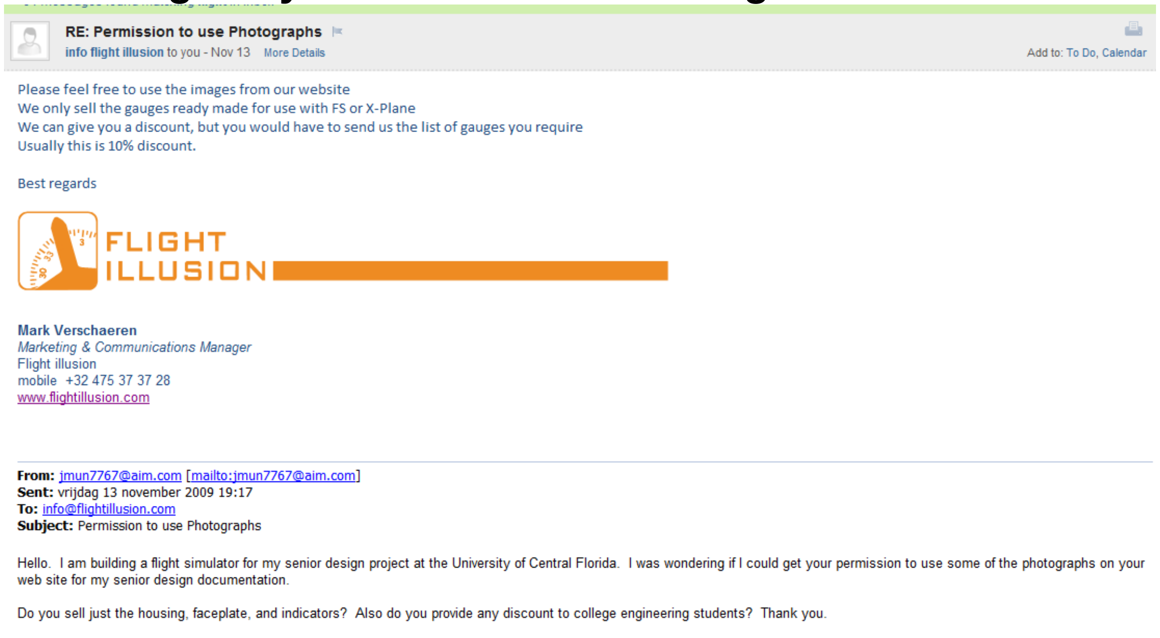
Project Files Delivered:
Budget

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

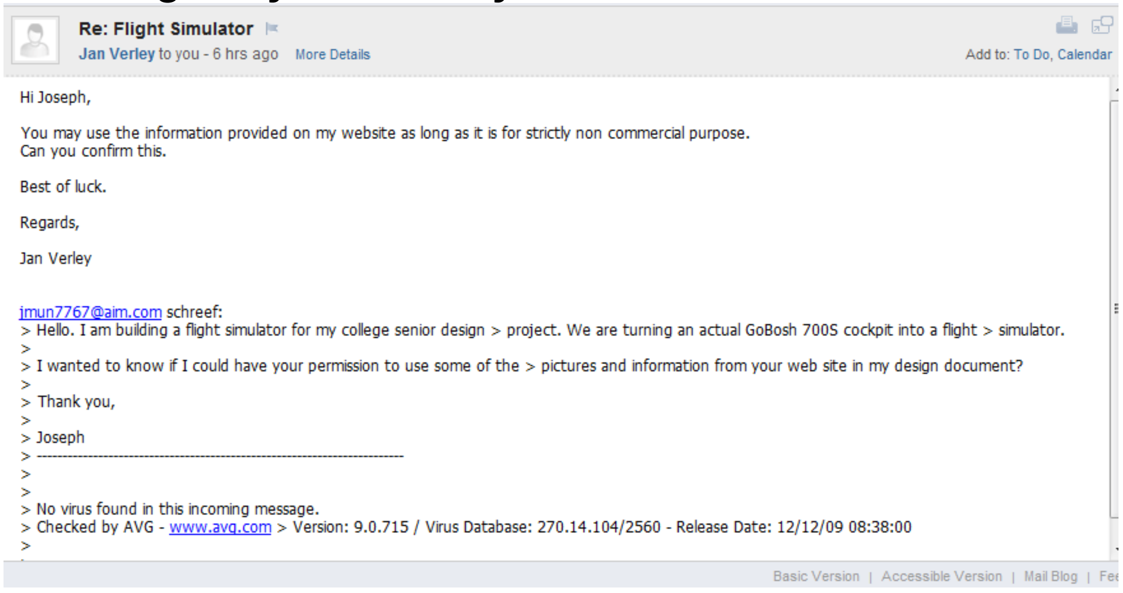
Hello. I am building a flight simulator for my senior design project at the University of Central Florida. I was wondering if I could get your permission to use some of the photographs on your web site for my senior design documentation.

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 3-17
B) Figure 3-20
C) Figure 3-23
D) Figure 3-26
E) Figure 3-27
F) Figure 3-29
G) Figure 3-32

C.2 Images by Jan Verley



Re: Flight Simulator

Jan Verley to you - 6 hrs ago More Details

Add to: To Do, Calendar

Hi Joseph,

You may use the information provided on my website as long as it is for strictly non commercial purpose.
Can you confirm this.

Best of luck.

Regards,

Jan Verley

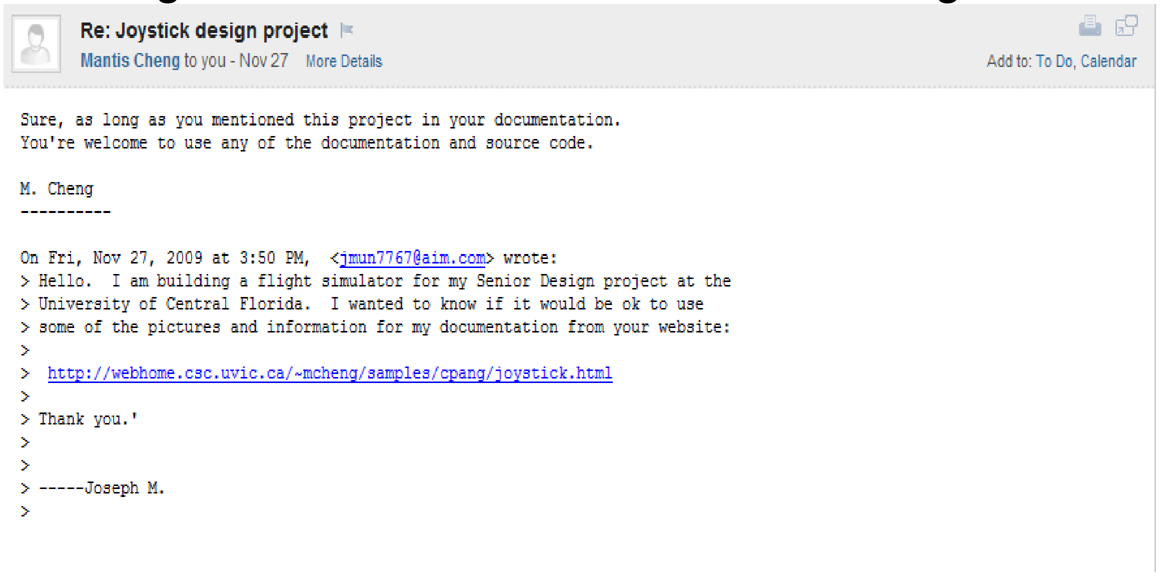
jmun7767@aim.com schreef:
> Hello. I am building a flight simulator for my college senior design > project. We are turning an actual GoBosh 700S cockpit into a flight > simulator.
>
> I wanted to know if I could have your permission to use some of the > pictures and information from your web site in my design document?
>
> Thank you,
>
> Joseph
> --
>
>
> No virus found in this incoming message.
> Checked by AVG - www.avg.com > Version: 9.0.715 / Virus Database: 270.14.104/2560 - Release Date: 12/12/09 08:38:00
>

Basic Version | Accessible Version | Mail Blog | Fee

Site:
This reference is used with permission for the following figures:

A) Figure 3-13
B) Figure 3-35
C) Figure 3-36
D) Figure 3-37
E) Figure 3-39
F) Figure 3-40

C.3 Images and Information from Mantis Cheng


Re: Joystick design project
Mantis Cheng to you - Nov 27 More Details
Add to: To Do, Calendar

```
Sure, as long as you mentioned this project in your documentation.
You're welcome to use any of the documentation and source code.

M. Cheng
----------

On Fri, Nov 27, 2009 at 3:50 PM,  <jmun7767@aim.com> wrote:
> Hello.  I am building a flight simulator for my Senior Design project at the
> University of Central Florida.  I wanted to know if it would be ok to use
> some of the pictures and information for my documentation from your website:
>
>  http://webhome.csc.uvic.ca/~mcheng/samples/cpang/joystick.html
>
> Thank you.'
>
>
> -----Joseph M.
>
```

Site: http://webhome.csc.uvic.ca/~mcheng/samples/cpang/joystick.html
This reference is used with permission for the following figures: (software stuff)

A) Figure 3-53
B) Figure 3-54
C) Figure 3-55

C.4 Information from Mike's Flight Deck

Site: http://www.mikesflightdeck.com/
If, after reading this, you are still interested, please be aware that the contents of this site are protected by copyright (copyright © 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009 by John M. Powell). Nonetheless, you may copy this material subject to these three conditions: (1) the copyright notice is copied and presented along with the material, (2) the copy is used for non-commercial purposes, and (3) the source of the material is properly credited.

This reference is used with permission for the following figures:

A) Figure 3-34

C.5 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 3-23

C.6 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. 3-16^	http://en.wikipedia.org/wiki/File:3-Pointer_Altimeter.svg
C)	Fig. 3-18+	http://en.wikipedia.org/wiki/File:Sens_alt_components.PNG
D)	Fig. 3-19*	http://en.wikipedia.org/wiki/File:True_airspeed_indicator-FAA.SVG
E)	Fig. 3-21+	http://en.wikipedia.org/wiki/File:ASI-operation-FAA.png
F)	Fig. 3-22^	http://en.wikipedia.org/wiki/File:R22-VSI.jpg
G)	Fig. 3-24+	http://en.wikipedia.org/wiki/File:Faa_vertical_air_speed.JPG
H)	Fig. 3-25*	http://en.wikipedia.org/wiki/File:Attitude_indicator_level_flight.svg
I)	Fig. 3-28*	http://en.wikipedia.org/wiki/File:Turn_indicator.png
J)	Fig. 3-30+	http://en.wikipedia.org/wiki/File:Turn_indicators.png
K)	Fig. 3-21*	http://en.wikipedia.org/wiki/File:Heading_indicator.svg