Group 11

GOBOSH G700S FLIGHT SIMULATOR

Chris Dlugolinski, Robert Gysi, Joseph Munera, Lewis Vail

Sponsored by: Mr. Dave Kotick, Grizzly Aviation

Motivation

- Create a realistic cockpit-based flight simulator that can be used by our project sponsor to increase sales of both the aircraft and of his flight instruction business.

Background (Aircraft)

- The GoBosh G700S is a US-spec version of the Polish-built Aero AT-3 and is considered to be in the Light Sport Aircraft (LSA) segment, which requires very little pilot training compared to other small, single engine aircraft.

Background (Project)

- To build a fully-integrated simulated cockpit environment which includes:
  - Standard Six-Pack gauges (Airspeed, VSI, Altimeter, Heading, Attitude and Turn Coordinator
  - Flight Controls (Pedal, Stick, Throttle)
  - Projection system
- Integrate the above systems into an actual cockpit being shipped to our sponsor from Poland (did not arrive)

Outline

- Simulator Software
  - Aircraft Model
- Computer Hardware
- SDK Software
- Instrument/Control Software
- Flight Controls
- Flight Instruments
- Administrative Information

Simulator Software

Flight Simulator Requirements

- Needs to be a low-cost software package
- Allow us the ability to interface with custom instruments and controls
- Provide a realistic environment
- Guarantee 30 Frames Per Second
- Method for creating/importing a custom aircraft model
- Two that meet these requirements:
  - Microsoft Flight Sim X (FSX)
  - Laminar Research X-Plane 9

Microsoft FSX Pros

- The most popular desktop based flight simulator available on the market; large community of add-on developers
- Uses the FSUIPC and the SimConnect API for interfacing custom devices into the simulator
- Inclusion of many worldwide airports and accurate detailed scenery in large cities Computer-controlled (AI) based aircraft populate airspace automatically
- Cost: $30

Microsoft FSX Cons

- We can't deliver a guarantee a minimum of 30 FPS. A target can be set but the game will not auto-adjust settings to maintain the frame rate.
- No included model editor – require expensive 3rd party modeling tools.
- No built in Instructor Operator Station (IOS) functionality out of the box. Requires additional development.
- No longer in development – Microsoft closed the ACES studio in Jan. 2009.

X-Plane 9

- Decided to go with this software instead.
- While there is not as an extensive community of add-on developers, the SDK documentation is very thorough – everything is written as a plugin for the simulator software.
- Includes a built in model editor, meaning no need to purchase additional software.
- Can maintain a frame rate of 30 FPS
- With additional thumb stick from Laminar Research the simulator software can become FAA Certified and used for ground based training.
- Cost: $30

FSX/X-Plane Graphics Comparison

◄ FSX

▼ X-Plane 9

Both screenshots are of a Cessna C172 over Innsbruck, Austria.

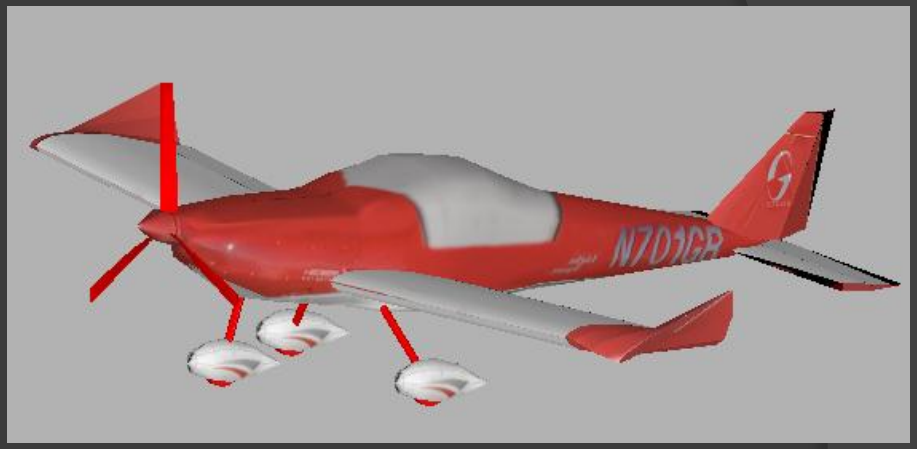
Aircraft Model

Aircraft Model Requirements

- Model must match the actual Aircraft
  - Physics
    - NACA 4415 Wing Profile
  - Look
  - Flight Control
- Build in the included X-Plane Plane Maker

Aircraft Model

- Used X-Plane Plane-Maker
- Traced fuselage shapes using scaled drawings from Aero
- Some limitations with the Horizontal and Vertical Stabilizers
- Issue with the ROTAX 912ULS Engine Specifications - using engine specs from another LSA aircraft



Aircraft Model (Limitations)

- Due to certain aspects, the model generated for this project is not 100% accurate. This is partially due to software limitations and skill limitations.
- Vertical and Horizontal Stabilizers do not function as they do on the actual aircraft. The X-Plane Plane Maker lacks the ability to change the pivot point of the horizontal stabilizer or the lower cord at a higher slope from the top. Both of these are features to be added in later versions of the software according to Laminar Research.

Airfoil

- In order to create a more accurate model of the aircraft, we needed to create an airfoil to attach to our wing model.
- Using data from the UIUC Applied Aerodynamics Group, we took the coordinate data file for the NACA 4415 wing profile and then used JavaFoil to create an X-Plane compatible .afl file.

Simulator Computer

Simulator PC Requirements

- CPU: 2GHz
- RAM: 4GB
- HDD: 120GB
- Graphics Card(s) powerful enough to output 120-Degree Simulated field of view on three monitors (Software requires minimum of 64MB onboard graphics RAM)
- Monitor: 24” or larger to created desired FOV (using equations above)

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

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

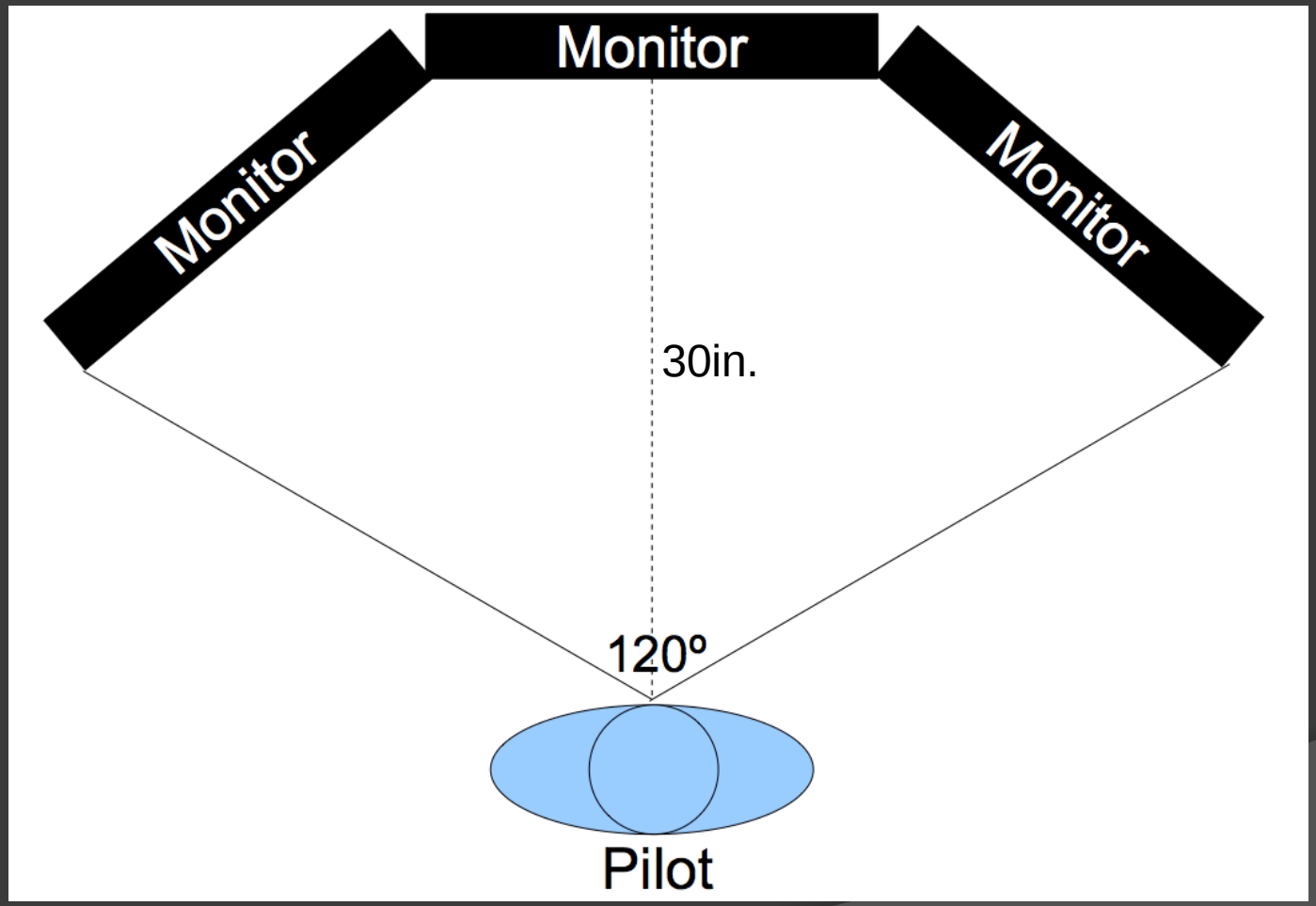
$$monitor\ size = (distance)(constant) \quad (Eq. 3)$$

Simulator PC Specifications

- Total Cost would be roughly $1200 to outfit the entire computer system, even using fairly low-cost components.
- Computer not actually purchased due to lack of cockpit/decision to not display at Sun ‘n Fun.

Item	Description
CPU	AMD Phenom X2 550 @ 3.1 GHz
GPU (x2)	ATI Radeon 5750 1GB RAM onboard each
Motherboard	ASUS M4A785TD-V EVO
HDD	160GB
DVD-ROM	Yes
Power Supply	1000W ATX
Monitors (x3) – 24”	Gateway FHD2402

Monitor Configuration

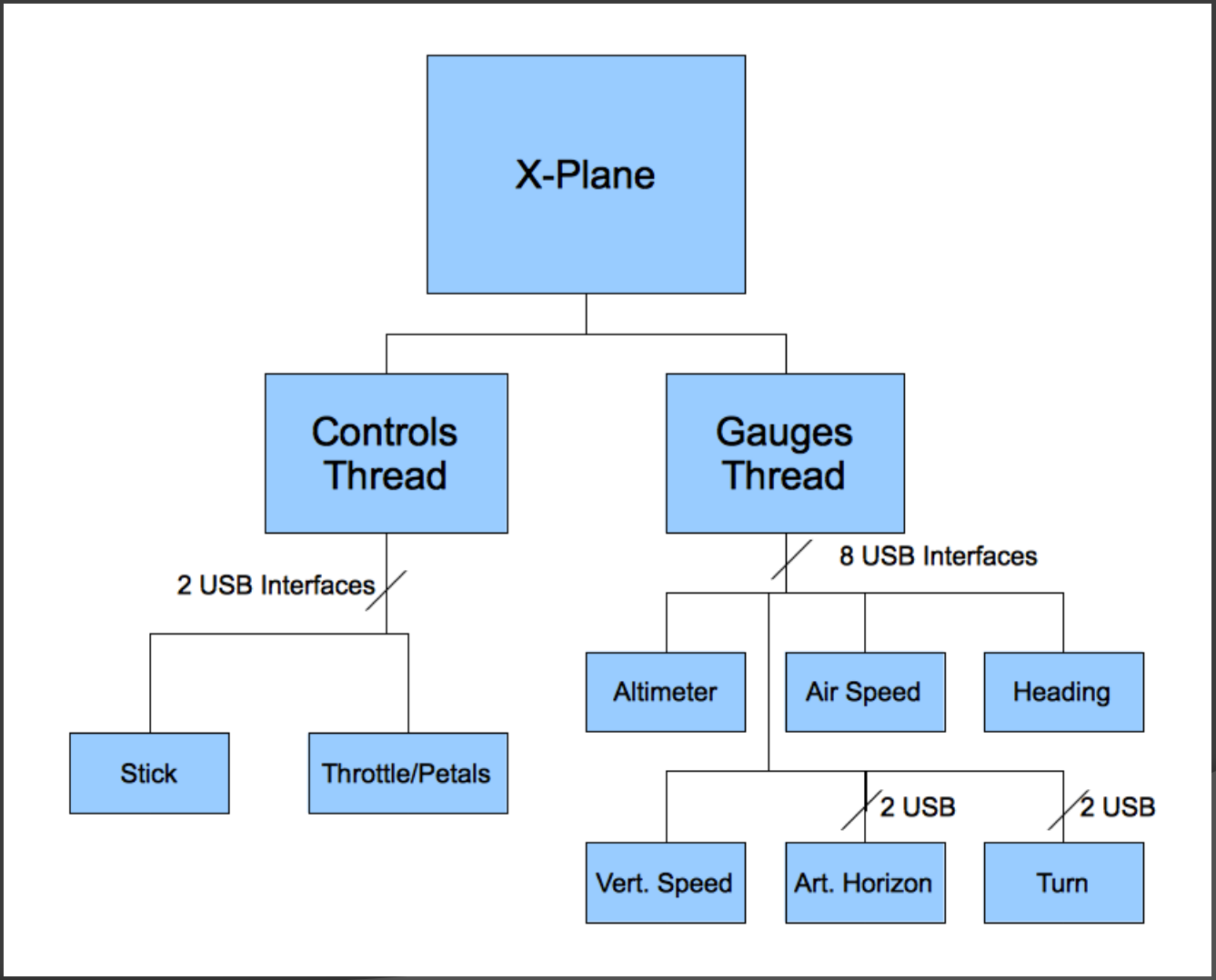


SDK Software

Plugin Requirements

- Wanted our development to be as modular as possible for future development
  - Plugins are completely data driven
- Need to be as realistic as possible
  - Want to match the 30fps we are getting from the graphics
  - Sampling input once every 15ms
  - Writing to gauges every frame

High-level Plugin Architecture

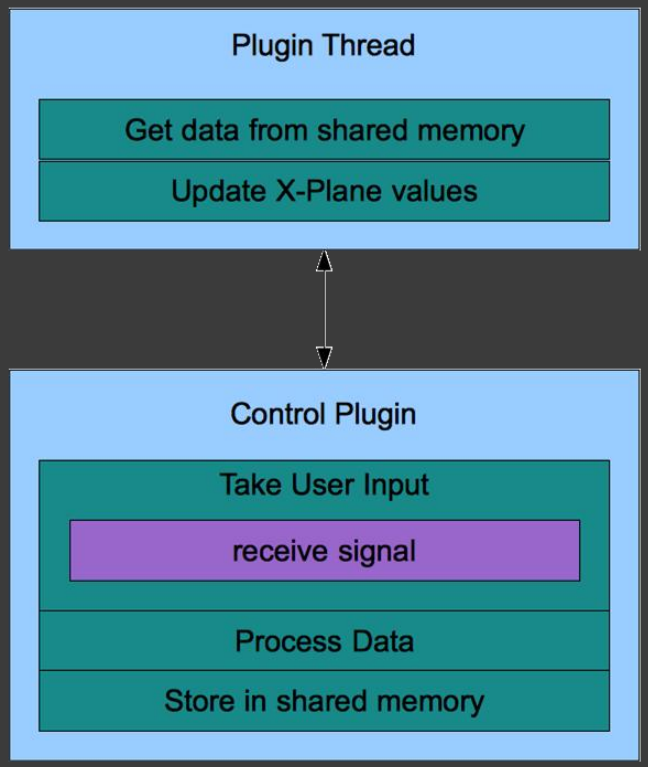


Plugin Design

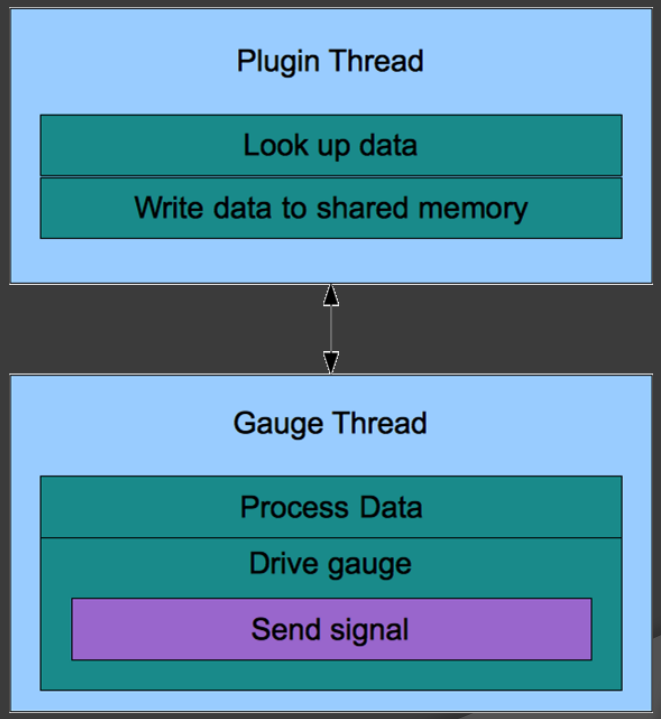
- Plugin Uses three threads:
  - Main Thread:
    - Manages initialization, X-Plane interface, and destructions
  - Controls Thread:
    - Read position of controls, translate to X-Plane Value, and write new value to shared memory for main thread
  - Instruments:
    - Read X-Plane values from shared memory, translate to number of steps, and step.

Plugin Design (Continued)

- Control Thread Architecture



- Gauge Thread Architecture

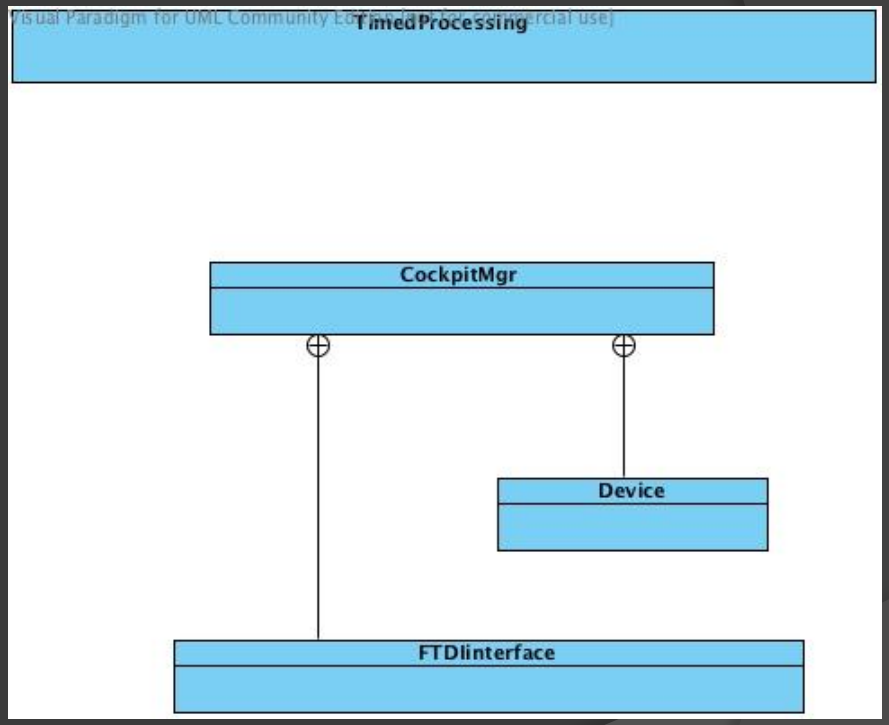


Plugin Implementation

- Writing everything in C++ because this what the X-Plane SDK supports
- X-Plane is cross-platform but our code is written for a windows environment
- Conversions between X-Plane values and number of steps will all be done using stored minimum and maximum values for each device
  - This data will be stored in config.ini

Class Diagram

- Four classes:
  - **TimeProcessing** interfaces with X-Plane
  - **DeviceMgr** is the container class for FTDIinterface and Device
  - **FTDIinterface** interfaces with the FTDI chips
  - **Device** stores the FTDI data



Instrument/Control Software

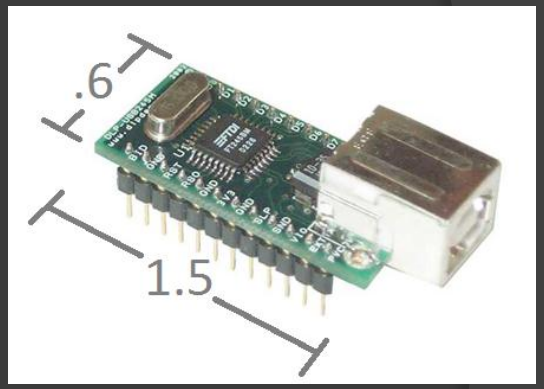
Control Design Decisions/Requirements

- Data Speed
  - Must use USB for communications
  - Smooth gauge motion
- Modularity
  - Adding gauges
  - Adding controls

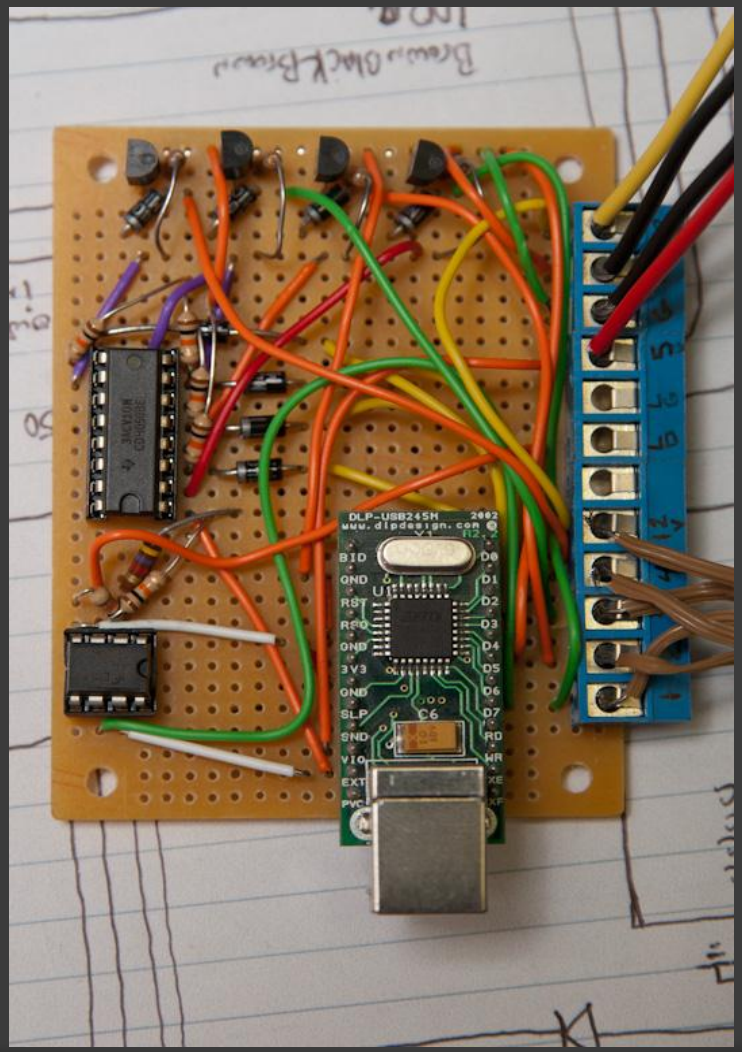
Control Requirements

- Need to support USB
- Control stepper motors and switches (if time permits)
- Work with A/D converters
- Use less than 5v and
- < 100mA at startup and
- < 500mA fully functioning
- Fit in a 3.25 inch square
- At least 8 I/O pins

FT245BM



Benefits of FTDI chip



- No need for a microcontroller (but can use one if needed)
- Simple circuit design
- Only need to program on the computer. (only one language)
- Allows for Modular Code

More Benefits

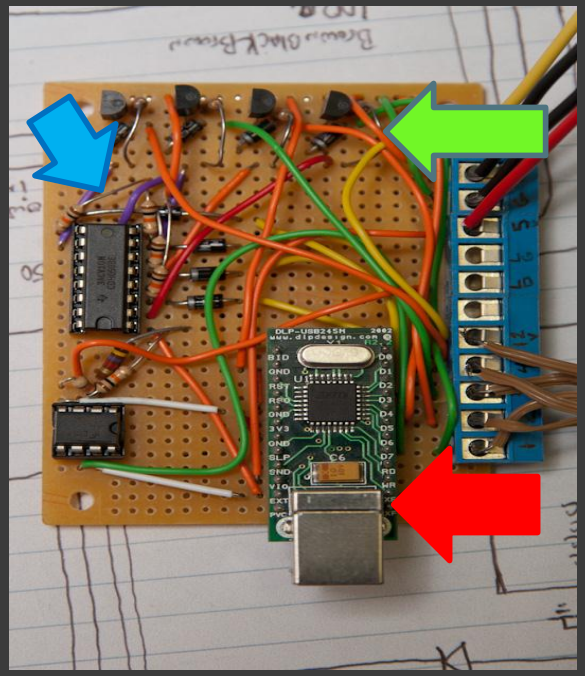
- Data Speed Calculation

Actual Motor speed FTDI chip

$$200\frac{step}{rev} * 2\frac{ms}{step} = 400\ ms/rev \qquad \frac{1}{\frac{.4s}{rev}} = 2.5\ rev/s$$

- More than fast enough for any gauge that we created
- If needed we could control two motors with one FTDI chip

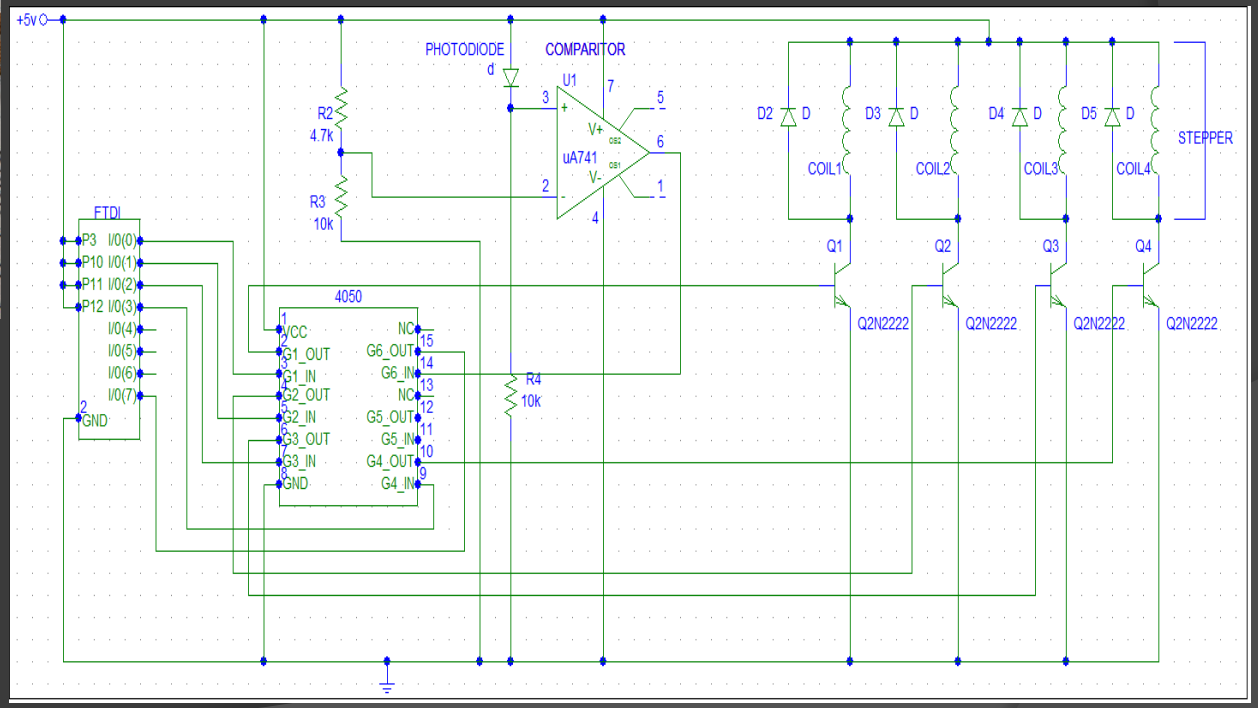
Completed Circuit (Gauges)



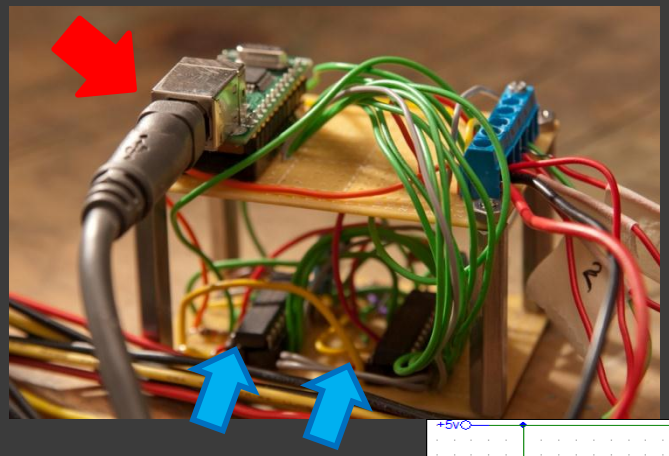
FTDI Chip

Buffer 4050

Motor Control

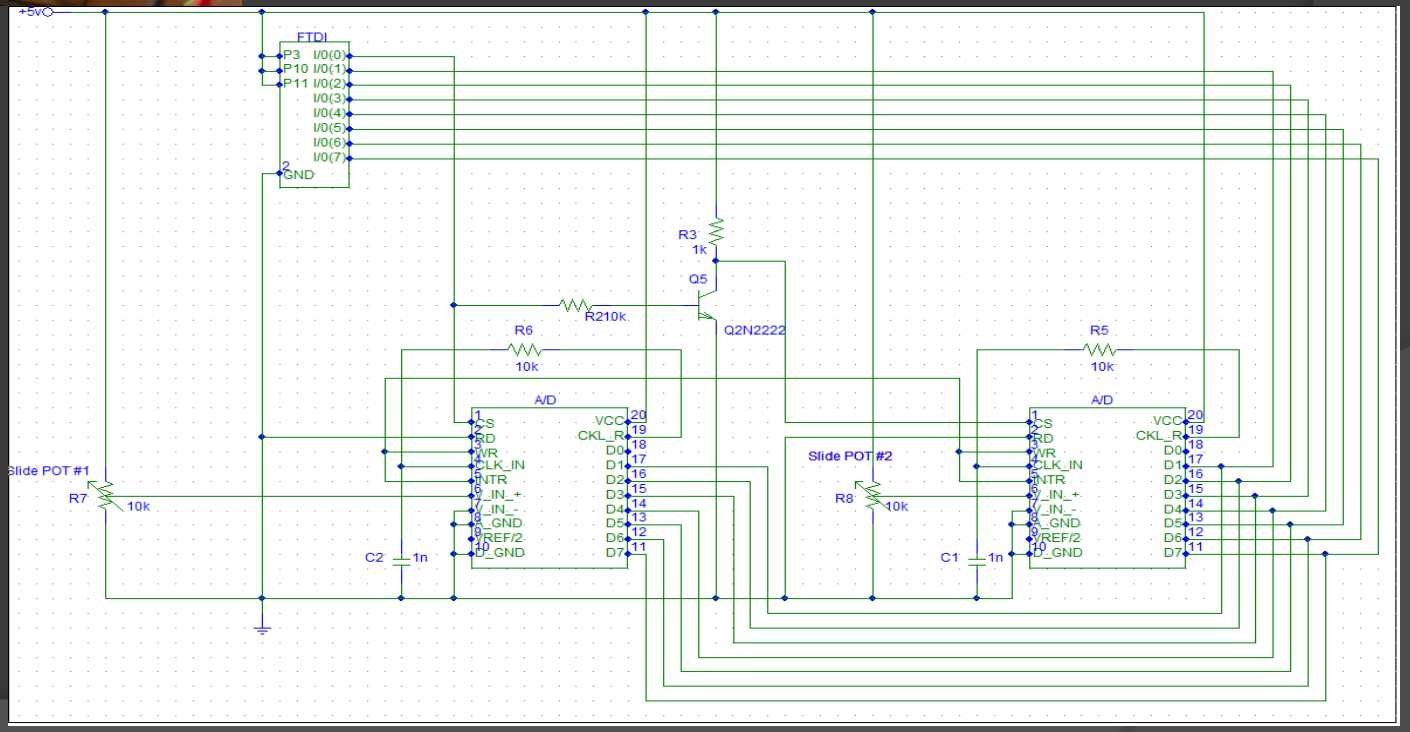


Completed Circuit (Controls)



FTDI Chip

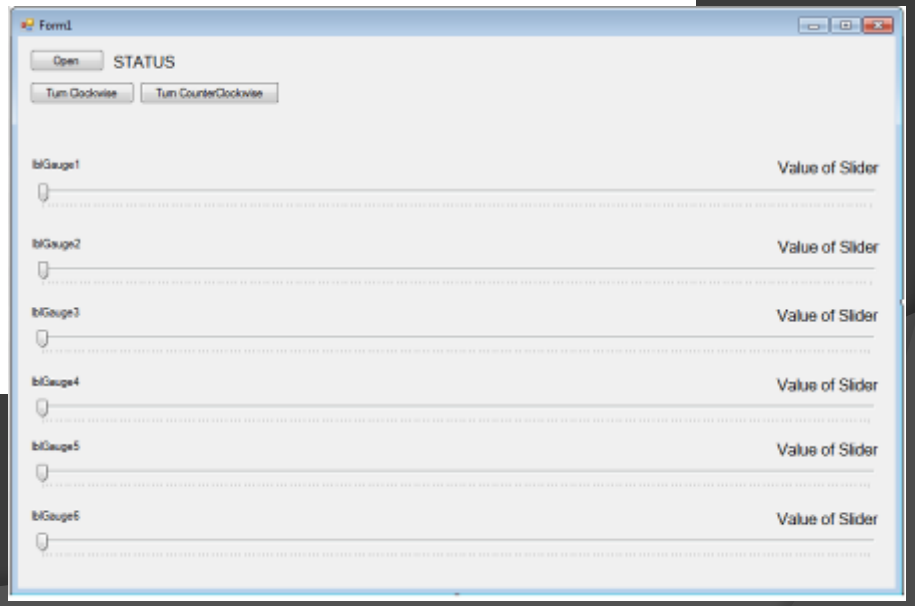
ADC



Implementation

```
private: System::Void gaugeSlider_Scroll(System::Object^  sender, System::EventArgs^  e)
{
	lblValue->Text = gaugeSlider->Value.ToString();
	if(previousValue < gaugeSlider->Value)
	{
		if(m_obUSBComm->GetAvailableGaugesList()[0].type == "FTDI Chip")
			m_obUSBComm->GetAvailableGaugesList()[0].status = m_obUSBComm->MoveClockwise(m_obUSBComm->FindHandle(0));
		else
			m_obUSBComm->GetAvailableGaugesList()[0].status = m_obUSBComm->MoveClockwise(m_obUSBComm->FindHandle(0));
	}
	else if(previousValue > gaugeSlider->Value)
	{
		if(m_obUSBComm->GetAvailableGaugesList()[0].type == "FTDI Chip")
			m_obUSBComm->GetAvailableGaugesList()[0].status = m_obUSBComm->MoveCounterClockwise(m_obUSBComm->FindHandle(0));
		else
			m_obUSBComm->GetAvailableGaugesList()[0].status = m_obUSBComm->MoveCounterClockwise(m_obUSBComm->FindHandle(0));
	}
	previousValue = gaugeSlider->Value;
	if(gaugeSlider2->Enabled && gaugeSlider2->Visible)
	{
		gaugeSlider2->Value = previousValue;
	}
}

FT_STATUS USBComm::Open(int deviceNumber, FT_HANDLE* handle)
{
	FT_STATUS ret;
	ret = FT_Open(deviceNumber, handle);
	if(ret != FT_OK)
	{
		// FT_Open failed
		ShowError("Open Faileed");
	}
	return ret;
}
```



Flight Controls

Control Design

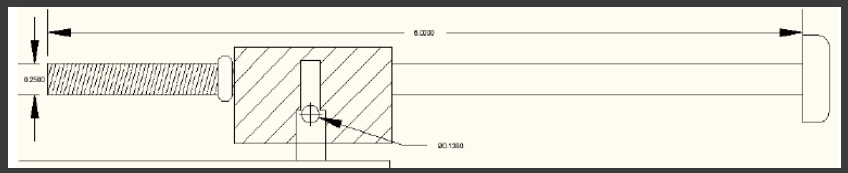
- For implementation in the actual cockpit, the design for all three would have been very similar – just a 10k-ohm slide pot attached to the push-pull rods, pedals, and throttle shaft.
- The throttle design was not affected by the lack of a cockpit – mechanically or electrically.
- For the joystick and pedals our electrical design remains unchanged, however we needed to build test rigs – resulting in new mechanical design.

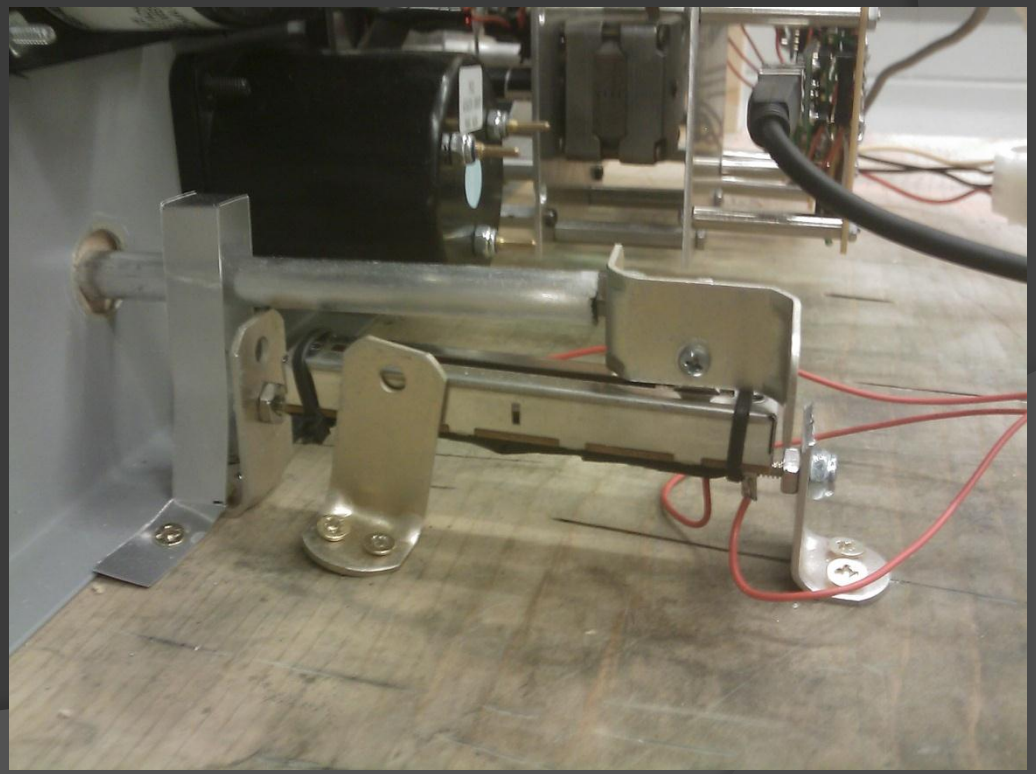
Flight Controls

- No real cockpit or original controls
- Controls therefore needed to be built (stick, pedals, throttle)

Mechanical Design (Throttle)

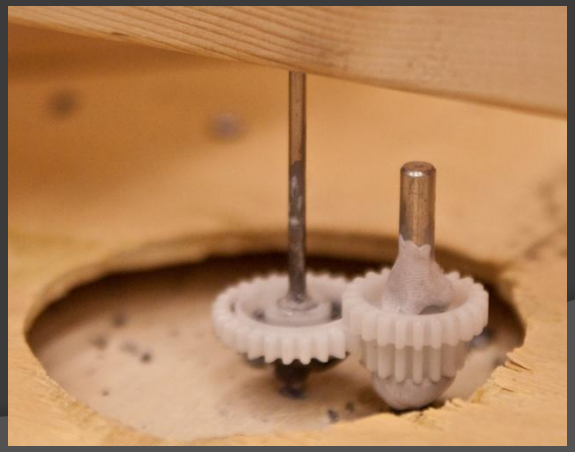
- Throttle is very basic: A threaded rod is connected to the slider on a 10k Slide Pot and secured to the platform.





Stick/Pedals (Mechanical)

- Originally we planned to connect to the existing mechanical linkages in the cockpit.
- Since it did not arrive we needed to build test rigs to validate our electrical design and provide input to the simulator.



Cockpit Overview

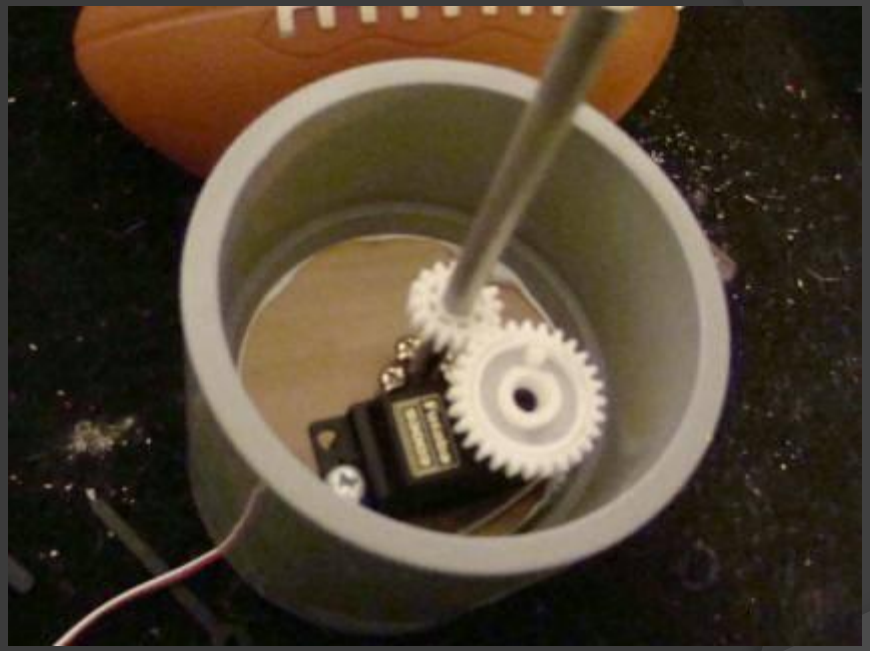
- Cockpit view showing instruments to be implemented (in addition to other items outside of the scope of this project).

Motor Selection

- Two options: Servo or stepper motor
- Servos: Uses Pulse Width Modulation, use of a 555-timer circuit cannot give precise control over motor position.
- Steppers: Allows us to step through our rotations with no limit on number rotations, very inexpensive

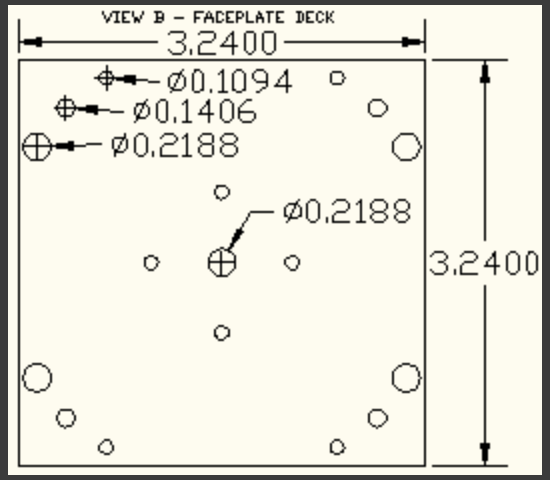
Servo Motor Prototype

- 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)

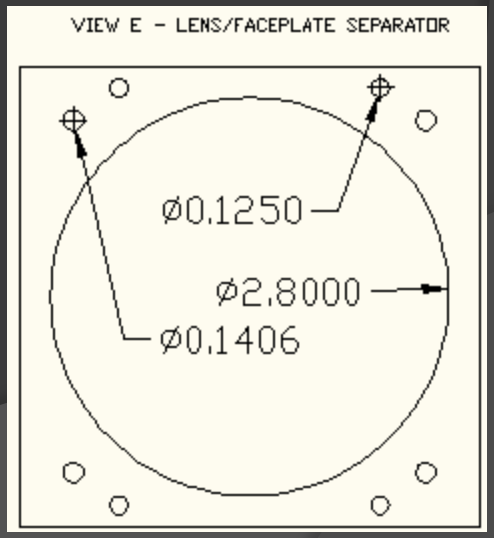


Common Materials

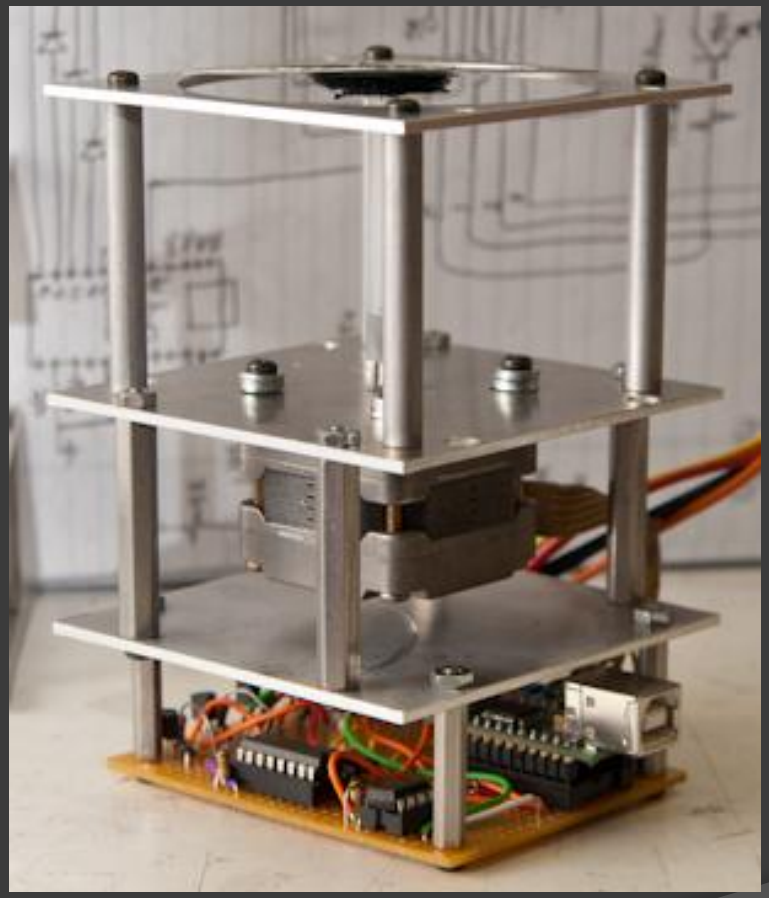
- 200 step/rev, 1.8 degrees per step, unipolar stepper motor
- .050” sheet aluminum
- .125” clear acrylic sheet
- OWCP 4537 CDS Photocell
- FTDI FT245BL USB Communication Board
- #4-40 Hex Spacers
- #4-40 Screws



Two of the decks that are common to many gauges



Basic Gauge Structure



Airspeed Indicator

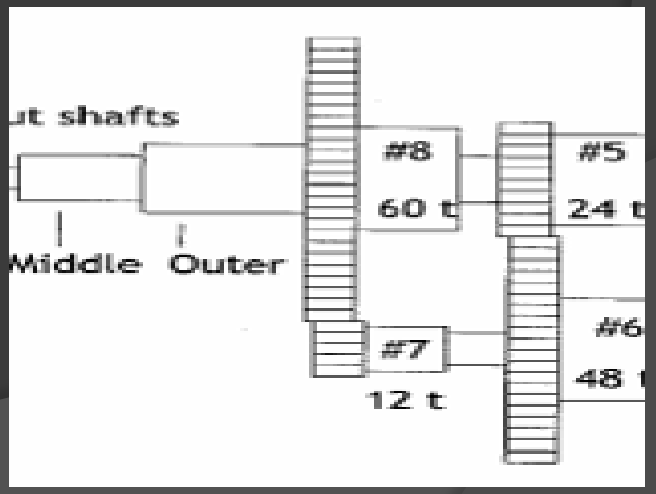
- Displays airspeed of the simulated aircraft.
- Due to the nature of the GoBosh, our displayed airspeed range will be 0-160 Knots.
- Requires nearly 360-Deg. range of motion from a single stepper motor.

Vertical Speed Indicator

- Displays vertical speed of the simulated aircraft.
- Displayed output range is 0 to +/-20
- Requires nearly 360-Deg. range of motion from a single stepper motor.

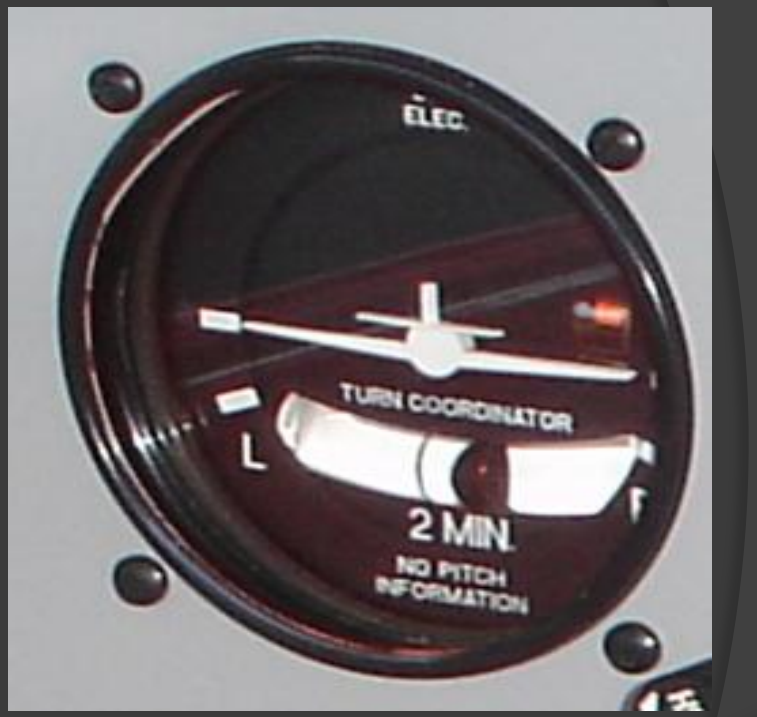
Altimeter

- Measures the Altitude of an object above a fixed level
- Displayed output range is 100 and 1000 feet
- Requires nearly 360-Deg. range of motion from a single stepper motor.
- Requires the gearing of the shaft to accommodate the dual needles representing 100 ft and 1000ft increases.
- 1:10 Gear ratio is required.



Turn Coordinator

- Displays the rate of yaw (turn), roll, and the coordination of the turn.
- Requires two stepper motors. One for the level indicator and one for the ball.
- The wings on the level indicator are limited to +/- 90-Deg.
- The ball moves moves within 50-Deg. in the ball track.



Attitude Indicator

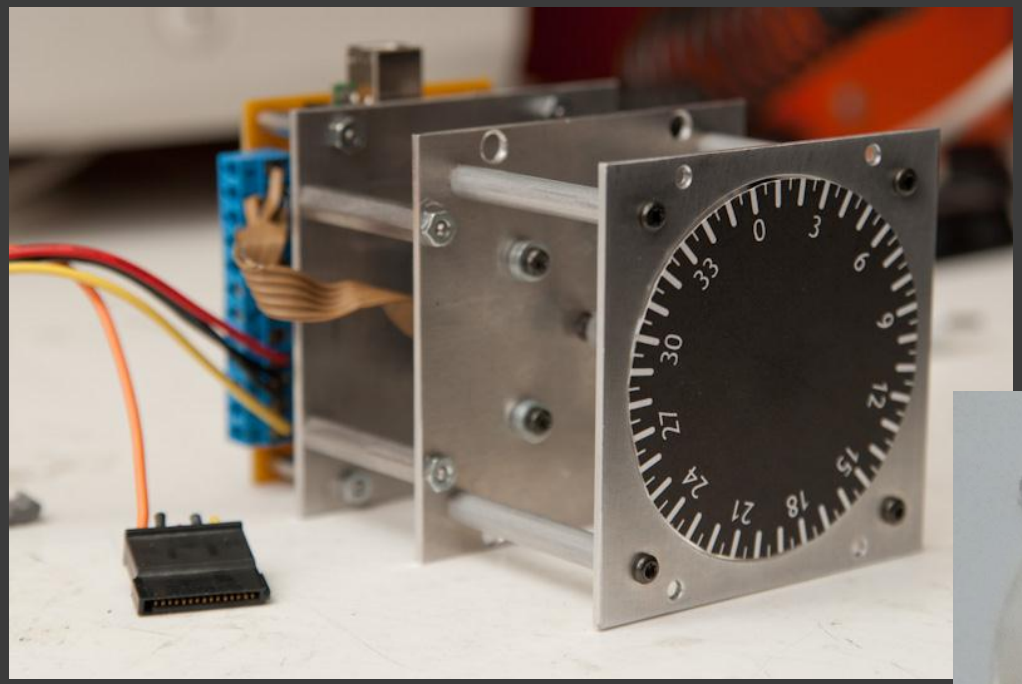
- Displays aircraft relative to the horizon.
- Gyroscopic instrument – modifed actual instrument by replacing gyros with stepper motors.
- Requires two stepper motors, one to indicate pitch and one to indicate roll.

Heading Indicator

- Displays aircraft heading (compass)
- Requires 360-Deg. of movement with no mechanical stops (flying in a circle) - stepper motor

Prototype Build

Final Product



Requirements

Requirements - Sim Software

- The selection of X-Plane met all of our requirements for a flight simulator application.

Req. #	Sub. Req.	Requirement Description	Result
S1	-	Realistic Look and Feel.	MET
S1	A	Realistic Scenery	MET
S1	B	Inclusion of Airports Worldwide.	MET
S2	-	Ability to change environmental factors dynamically.	MET
S2	A	Ability to Interface Hardware with software via API.	MET
S3	-	Model Entertainment Aspects	MET
S3	A	Weather Effects.	MET
S3	B	Crash Effects.	MET
S3	C	Sounds: Realistic prop sounds.	MET
S3	D	Ability to create custom scenarios/missions	MET
S3	E.	AI Aircraft also utilizing airspace and airports.	MET
S4	-	Aircraft Model	MET
S4	A	Included Model Editor to create 3D Model	MET
S4	B	Ability to create flight model with parametric data	MET
S5	-	Ability to interface with other Flight Sim/X-plane games	MET
S5	A	Native Multiplayer Support	MET
S6	-	Guaranteed minimum 30 FPS	MET
S6	A	Ability for software to be FAA Certified	MET
S7	-	Ability to interface controls/flight instruments	MET
S8	-	Ability to interact with an IOS	MET

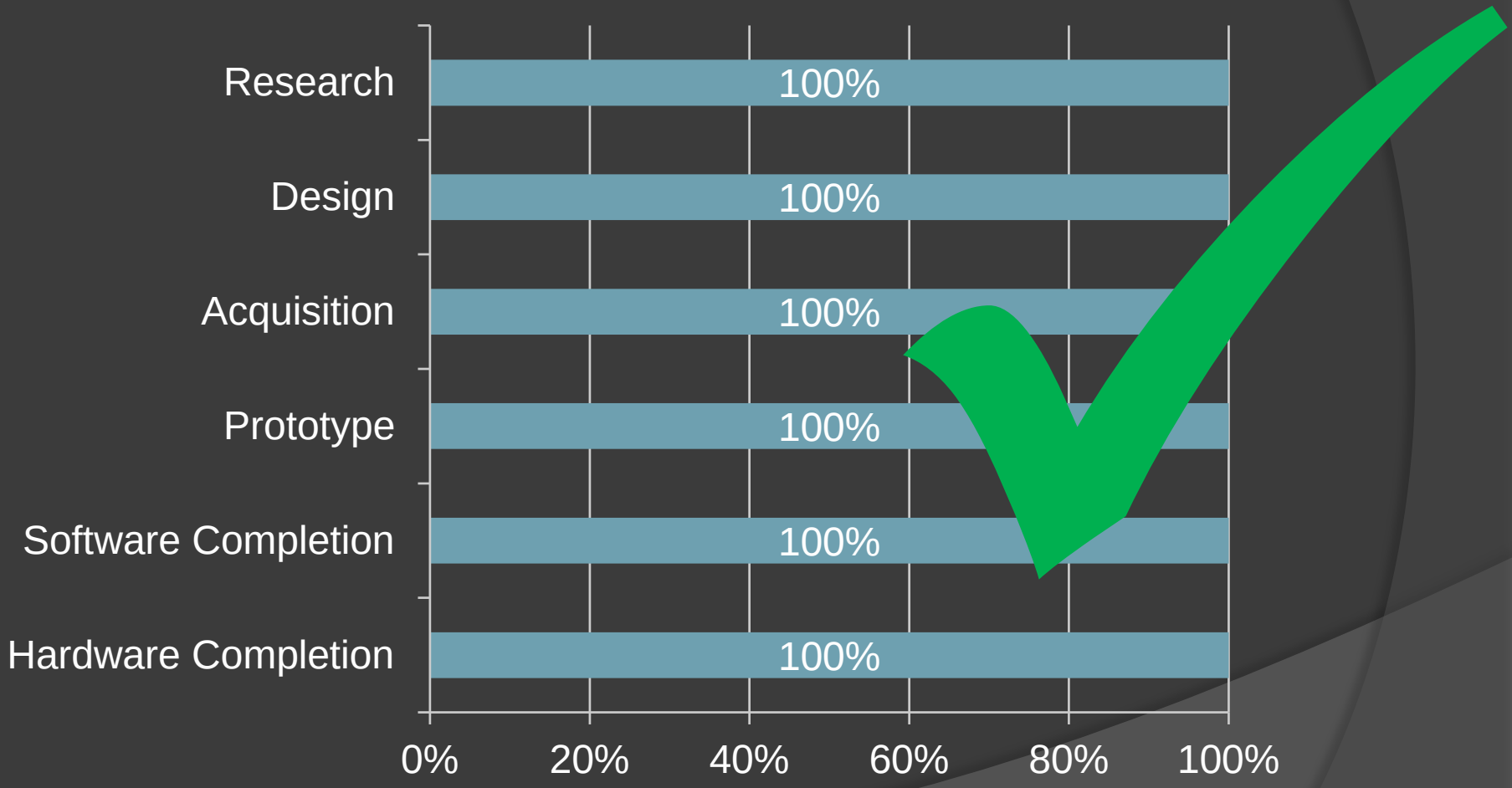
Requirements - Hardware

- The decision to not demonstrate at Sun ‘n Fun due to the lack of a cockpit from Aero/GoBosh is the cause for requirements not being met.

Req. #	Sub. Req.	Requirement Description	Result
C1	-	USB interface for controls & gages	MET
C2	-	120 degree field of view	PARTIAL
C2	A	3 LCD monitors	NO
C2	B	Graphics Card/Adapter capable to power three monitors	NO
C3	-.	2Ghz 64-bit CPU (minimum):	NO
C4	-	4GB RAM	NO
C5	-	120GB Hard Drive (minimum)	NO
M1	-	USB Controlled	MET
M2	-	20ms refresh rate (minimum)	MET
M3	-	Use less than 5V to power the actual chip.	MET
M4	-	Minimum 8 I/O Pins for external communications	MET
M5	-	Fit inside of a 3.24”x3.24” profile.	MET
M6	-	Low Cost Microcontroller or USB communication development board	MET
M7	-	As self-contained as possible: Does not require any complex circuitry or boards.	MET
F1	-	Motor to drive flight instruments: Must be able to complete a turn of over 360 degrees for the altimeter and heading indicator.	MET
F2	-	Realistic flight instruments and controls	MET
F2	A	Gauges: Standard Six-Pack has been implemented - Altimeter, Airspeed Indicator, Attitude Indicator, Turn Coordinator, Heading Indicator, Vertical Speed Indicator.	PARTIAL
F2	B	Flight Controls (Stick, Pedals, Throttle)	MET

Administrative Information

Progress


Research
Design
Acquisition
Prototype
Software Completion
Hardware Completion
100%
100%
100%
100%
100%
100%
0%
20%
40%
60%
80%
100%

Spending

Item	Part Number	Quantity Required	Unit Cost	Total Cost
FTDI USB Communication Dev Board	FT245BM	10	$30.00	$300.00
8 Pin IC Sockets – 2pk		4	$0.48	$1.92
2N3904 Transistor	2N3904	4	$0.79	$3.16
Diodes	1N4003	5	-	$6.75
Powered USB Hub		1	$49.99	$49.99
Wire		3	5.99	$17.97
Spacers		-	-	$28.20
Terminal Blocks		-	-	$10.80
Large PCB Boards		7	$2.50	$17.50
Stepper Motors		8	$5.00	$40.00
Assorted IC Sockets		-	-	$19.80
Buffer IC	CD4050	6	$0.35	$2.10
Comparator	LM741CN	5	$0.25	$1.25
A/D Converter	ADC0804LCN	3	$2.50	$7.50
USB Cables		9	$5.00	$45.00
Misc. Hardware		-	-	$36.77
3/8x0.035 Aluminum Tube		1	$4.78	$4.78
Resistors		3	$0.99	$2.97
10k-ohm Slide Pots		3	$2.12	$6.36
Small PCB Boards		4	$1.99	$7.96
Connectors		-	-	$20.36
			Total	$631.14
			Project Budget	$1500.00
			Difference	$868.86

Questions/Demo