Design of a Stabilized Camera for a UAV

David E Smith

Goal of the Project:

The goal of this project was to create a stabilized camera intended to be mounted on the bottom of Dr. Elkaim's UAV. The system consists of a camera mounted on a two axis gimble. The gimble rotates through angles of roll and pitch in the opposite direction of the aircraft's roll and pitch. This keeps the camera horizontal and pointed at the horizon. Ultimately, the gimble would be capable of being mounted to a flying UAV without compromising the aircraft's structural or electrical integrity while transmitting stabilized video feed back to the ground station.

There are three systems interacting in the stabilized camera: the mechanical structure, control system, and electrical hardware.

Mechanical Structure:

- **Overview**: The gimble allows for rotation about two axes, one for roll and one for pitch. The mechanical structure of the gimble was designed in SolidWorks and the parts were cut using a laser cutter.

- **Using SolidWorks**: You can learn the basics of SolidWorks by watching the "SolidWorks Parts, part 1" and "SolidWorks Assembly, part 2" screencasts at http://www.soe.ucsc.edu/classes/cmpe118/Winter09/

    - Creating tabs:

      Open a sketch, select a plane and create a rectangle (as shown in the screencasts referenced above). Then draw rectangles along the edge(s). Click extrude and select the main body of the rectangle (not the tabs). You can reenter the sketch and alter the tabs by clicking on the plus sign next to "Extrude 1" on the left side of the workspace. Any other tabs or holes you add will automatically be extruded through the material. In order to get tabs to be the right size, you can do two things. First, you can create contruction lines along the inner edges of the tabs and set all these lines equal to one another like Gabe and Max did in the screencasts. Second, you can select each line on the inner edges of the tabs and enter their dimensions directly using the Dimension tool.

    - Setting up Dimension Variables

      This is not described in the screencasts. Setting up a variable allows you to create a design based on a changeable parameter (such as material width). You can create a variable called, say, "MaterialWidth" and assign certain dimensions in your design to equal this variable. You can also set certain dimensions in a design to equal a multiple or complex mathematical expression involving this parameter. To do this, click on the "Equations" button represented by a capital Sigma, and click add. Enter the name and value that you want the variable to have using the following format "VariableName" = X, where X is the value of

the variable. You can also create an equation involving other variables by entering mathematical expressions in place of X. So, for example, if I am using 0.25 inch acrylic as the material for a design but later I might create a prototype with foamcore which has a 0.2045 inch width, then I can add a variable "MaterialWidth" = 0.25. Now that the variable is added, I can set certain dimensions in the model to this variable by creating setting dimension using the dimension tool (as described in the screencast) and opening up the dimension box where you can enter a measurement number. Click on the drop down menu and select "Link Value". In the next box, use the drop down menu under the name heading to find your variable (it should be there if you created the variable correctly). Note: you can also setup variables in for use in an assembly workspace so that dimensions in multiple parts in an assembly will change based on one variable in an assembly workspace. You must be intelligent about linking dimensions to variables if you expect the entire model to change based on that variable. You must make everything that is affected by that variable be linked to the variable directly or through an equation.

- Creating a 2D drawing for exporting to CorelDraw.

  If you plan to cut out your parts using the laser cutter, then you will have to export your design as a 2D drawing because the laser cutter can, obviously, only cut 2D figures. The software that is used on the computer controlling the laser cutter is called CorelDraw and it accepts .dxf files. SolidWorks can export such files. The only trick is with getting the scaling correct. The first step to creating your 2D drawing is to create a new design in SolidWorks and choose "Drawing." Choose a standard sheet size. Click on the plus sign next to the word "Sheet1" in the hierarchy, you should then see the word "Sheet Format 1." Delete this sheet format thing (right-click, delete, confirm). It is just part information that we don't want on the drawing because the laser cutter will cut any shape that is on the drawing and we don't want the laser cutter cutting the part description. Then right click on the drawing workspace and choose properties. Make sure the scale is set to 1:1 on the top left corner of the window. !Note that the scale will change sometimes when you add parts so periodically check to make sure the scale is 1:1! The scale is very important so before you export the finished drawing as a dxf, make sure that the scale is 1:1. You can import parts into the drawing by clicking on the "View Palette" tab (fourth one from the top) and browsing for your parts. Then drag and drop the view of the part that you want onto the drawing remembering that this is the shape that the laser cutter will cut. Note: if you hold down the cntrl key while you move the part across the drawing the piece won't snap to the grid as annoyingly. When the parts are all place in the drawing check to make sure the scale is still 1:1 then click "Save As" and choose the .dxf extension. You will have to transfer the .dxf file via flash drive (or e-mail, etc) to the computer connected to the laser cutter.

- **Design Process/Choices**: Mechanical design is a process which passes through many stages. I began with visualizing a very different gimble then I ended up with. I pictured the gimble being about the size of my hand. I intended the gimble to be mounted to the plane by attaching to its landing gear. I expected to glue the parts together. However, as I progressed, I slowly learned that my design was going to be too big and too heavy. Gabe introduced me to T-slots and I changed from fastening

the assembly using glue to fastening every part together using nuts and bolts via T-slots. By the end, I used both glue and T-slots to obtain the lighter and stronger result. Designing is a iterative process. Below are images of each redesign – changes are made based on fastening methods, materials, and many other factors.

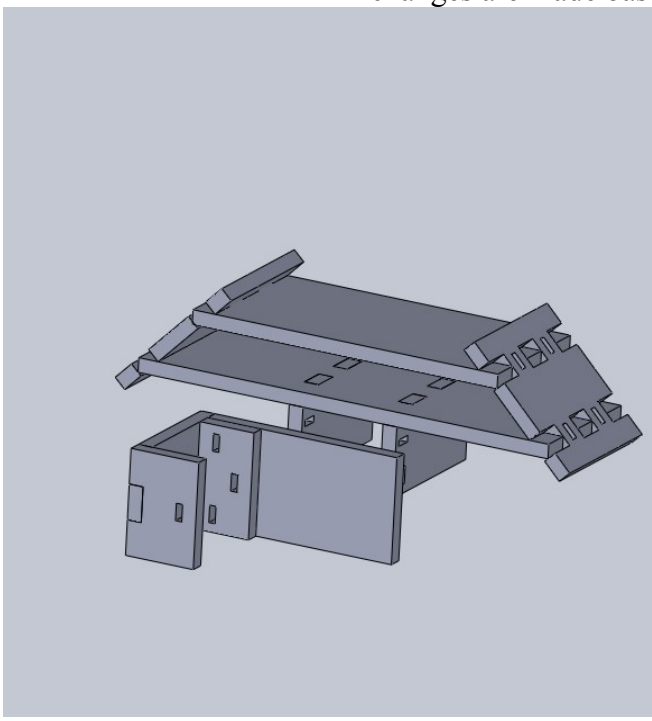


Illustration 1: First Design for FoamCore

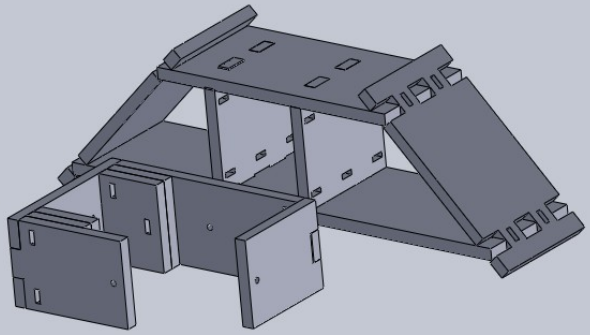


Illustration 2: Second Design: Pitch servo supports redesigned, roll servo mounted in middle of frame rather than underneath.

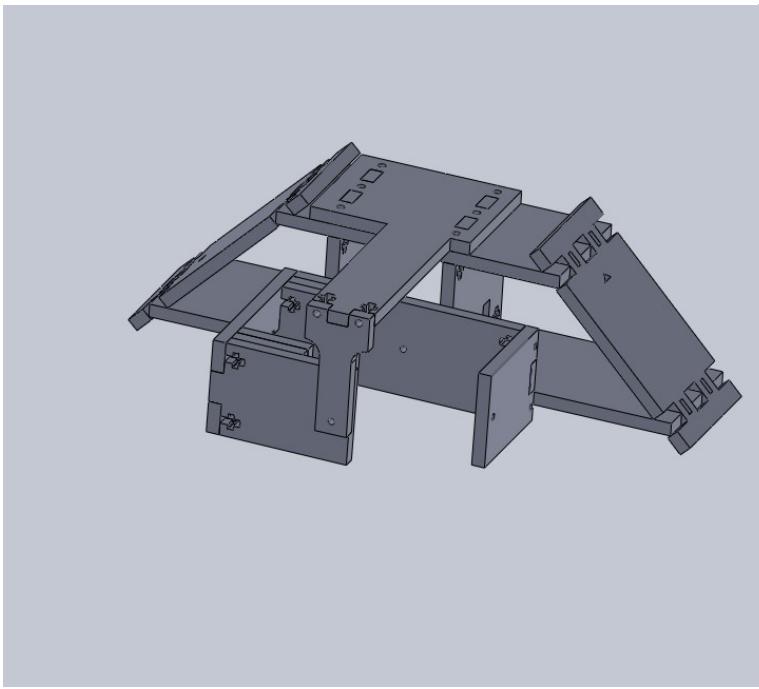
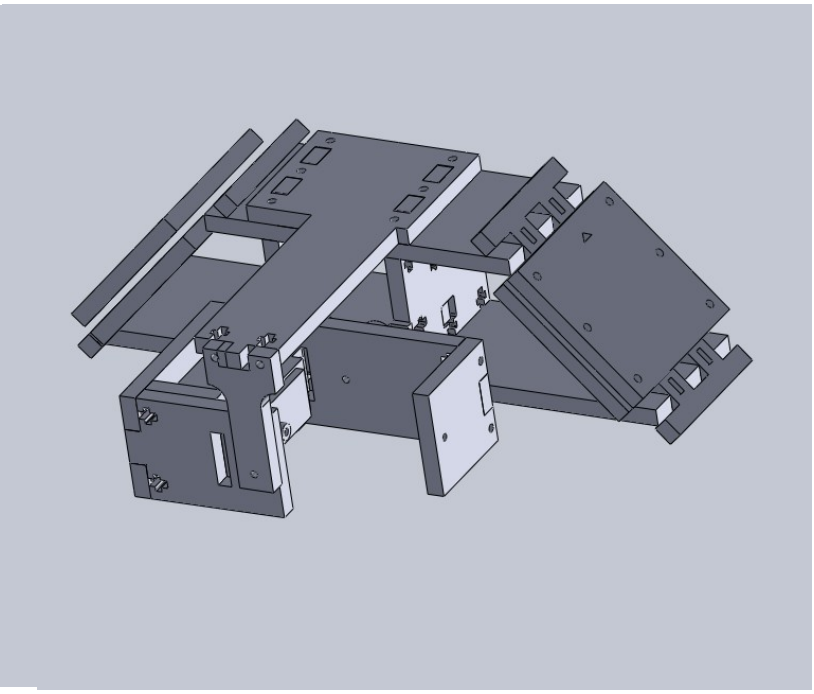


Illustration 4: Third Design: T-slots added, Top support added, holes for the servo wires added.



ıstration 3: Fourth Design: Plates for attaching gimble to ıding gear added, slots for pitch servo inserted, servo models included.

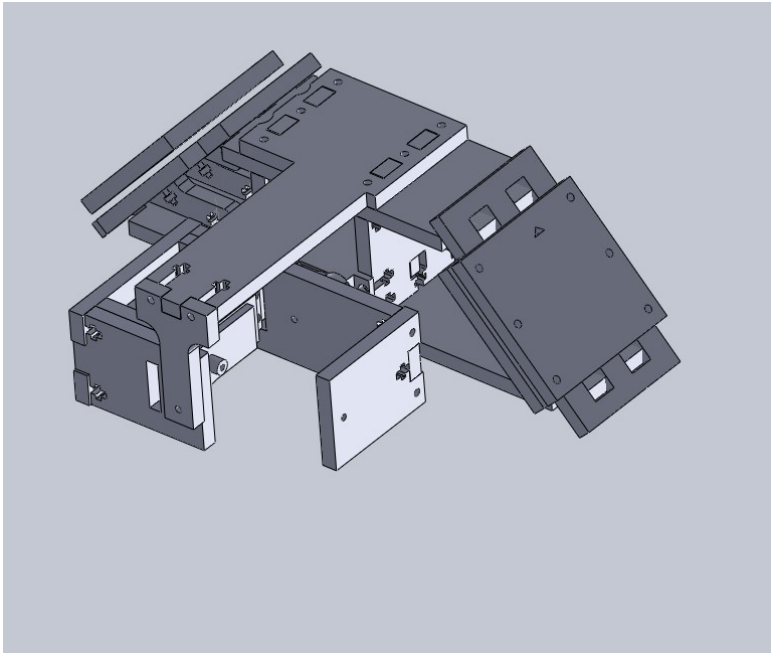


Illustration 5: Fifth Design: Braces added between side, top, and bottom plates for added integrity, many minor adjustments.

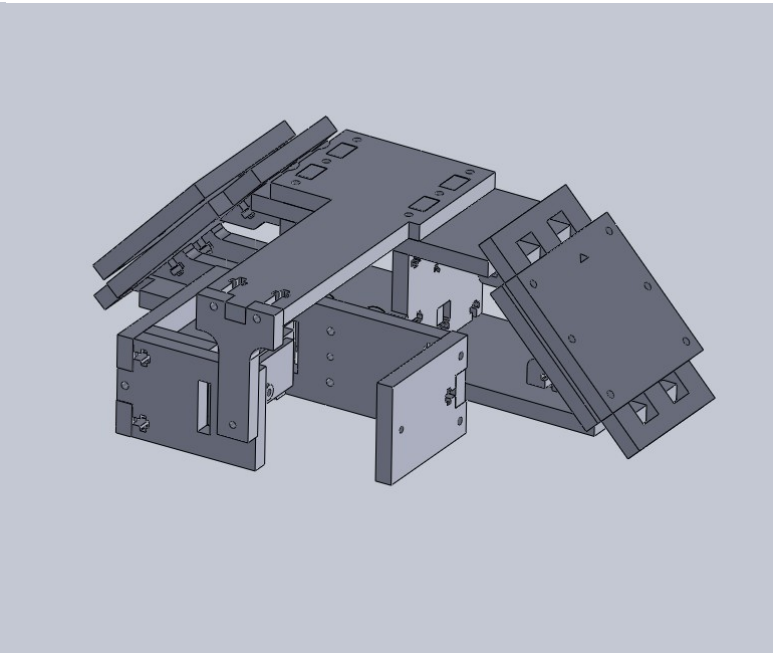


Illustration 6: Sixth DesignA lot of detailed changes, more t-slots added.

At this point, I realized that the gimble was too heavy. Mariano suggested that I remove the T-slots and Gabe asked for a redesign using micro servos. So I redesigned it from scratch, discarding the T-slots, except for attaching the servos,

using micro-servos rather than standard sized, and attaching the gimble using the four screws holding the landing gear to the fuselage rather then attaching it to the legs of the landing gear. The seventh design is shown below:

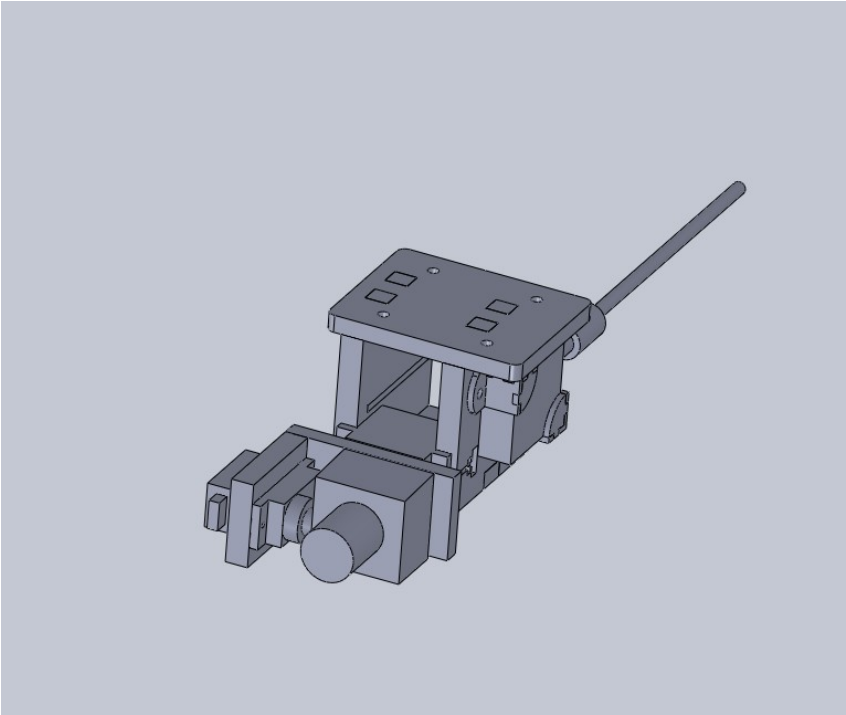


Illustration 7: Seventh Design: Completely redesigned. The long "tail" is the transmitter antenna.

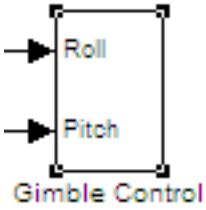
Control System:

- **Overview**: The control system uses a dsPIC33F running an algorithm created using Simulink. The algorithm receives attitude data from gyroscopes and outputs PWM signals to micro servos at the appropriate angles.

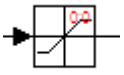
- **Servo Details**: The spec sheet for typical servos can be found by searching for the servo manual of the HS-311 (standard servo) or at this link http://www.hitecrcd.com/product_file/file/44/Servomanual.pdf. Both the standard and micro servos that I used function as described by this spec sheet. Basically, servos are controlled using PWM signals. PWM signals carry information in the duty cycle of a square wave. For servos, this means that you can command the angle of the servo by changing the duty cycle. The complete angle range of a servo is achieved using duty cycles ranging from 4.5% to 10.5%. This can also be expressed as pulse durations of 0.9ms to 2.1ms in a 20ms period square wave.

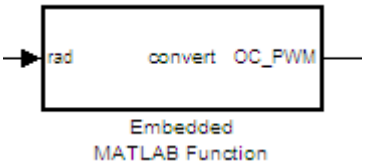
The following information refers to the model "controlMCUSlugsMKII_David2.mdl." Following along through the model may be helpful.

- **Gyroscope data details**: Using the gyroscope data is easy. I used the gyroscope angles by using a bus labeled "Euler_rot" in Mariano's autopilot model called "controlMCUSlugsMKII.mdl". That is, I added my "gimble control" model into his model (drag-and-drop) and used the "Euler_rot" bus as input into my model. After adding my model to his, I saved the new model as "controlMCUSlugsMKII_David2.mdl". I used a demux to separate the bus into three components: Roll, Pitch, and Yaw. I didn't need yaw so I terminated that input (adding a "terminator" block (shown at left) tells the model that you intend to do nothing with this output and prevents compilation errors). The only changes I made to this input was to put them through a 20Hz low-pass filter to attenuate noise and so keep the servos from being jittery. [Note: all blocks can be found in the Simulink Library Browser which you can get by clicking the icon (shown at left) in the Matlab workspace] The filtered roll and pitch angles are then input into the "gimble control" block.



- **"Gimble Control" Block**: In this block is the heart of the control system for the gimble. This is a subsystem block (meaning it contains blocks inside it which were represented as one block for efficiency). There are two inputs into this block: roll and pitch angles (both reported in radians) which come from the gyroscopes. "Gimble Control" takes the angles, applies saturation limits, and converts them to a number which the output compare can turn into a PWM signal with the desired duty cycle. [Note: All non-subsystem blocks have "parameter windows" that you can open by double-clicking them. As we pick apart the "gimble control" subsystem we will look at some of the parameters accessed through these windows.]
  - Negation: You will notice that the Roll input is negated by multiplying it by '-1'. This is because we want the camera to roll the opposite direction of the aircraft in order to keep the camera horizontal. The Pitch input is not negated because the orientation of the pitch servo is such that the angle is already inverted.
  - Saturation: The saturation block (shown at left) prevents the input angles from commanding the servos to an angle which is outside of their range (recall the 4.5% to 10.5% duty cycle restriction) or to prevent the servos from orienting the gimble such that it collides with the aircraft's landing gear or fuselage. The limits in the saturation blocks are determined mostly through experimentation.





  - "Convert Block": In order to create a PWM signal we have to use an output compare block. Inconveniently, the output compare does not accept radians. Rather the output compare accepts a number which determines the duty cycle. Thus we need to convert the radians to such a number. The number range is from 0 to 9999 (actually, I'm unsure about the upper limit called OCmax but I believe it is 9999). So, for example, if I want a 25% duty cycle, then I will input 2500. To command servos, I want a duty cycle ranging from 4.5% to 10.5% so I set the number range from 450 to 1050. [Note: Even though this number range is theoretically correct, after testing the code on the servos I tweaked the number range to maximize the actual angle range of the servos].
    The "convert block" (shown at left) is an embedded Matlab function meaning that it contains custom matlab code. My embedded matlab function converts the angle input from the gyroscopes into a number that the output compare will use to output a PWM with the correct duty cycle. It uses the following equation:

```
OC_PWM = 450*rad + 825
```

I determined this equation through both theoretical and experimental means. I gave the output compare a list of numbers from 450 to 1050 and measured what the resulting angle was on the servos. Fitting a linear function to the plot of these data points gives me this linear equation:

```
OC_PWM = 270*rad + 440
```

I want the servo to be centered (at an angle halfway between the extremes of its angle range) whenever the input angle is 0 (when the plane was level). The half-way/centered point occurs at 750 which is halfway between 450 and 1050. This results in the equation:

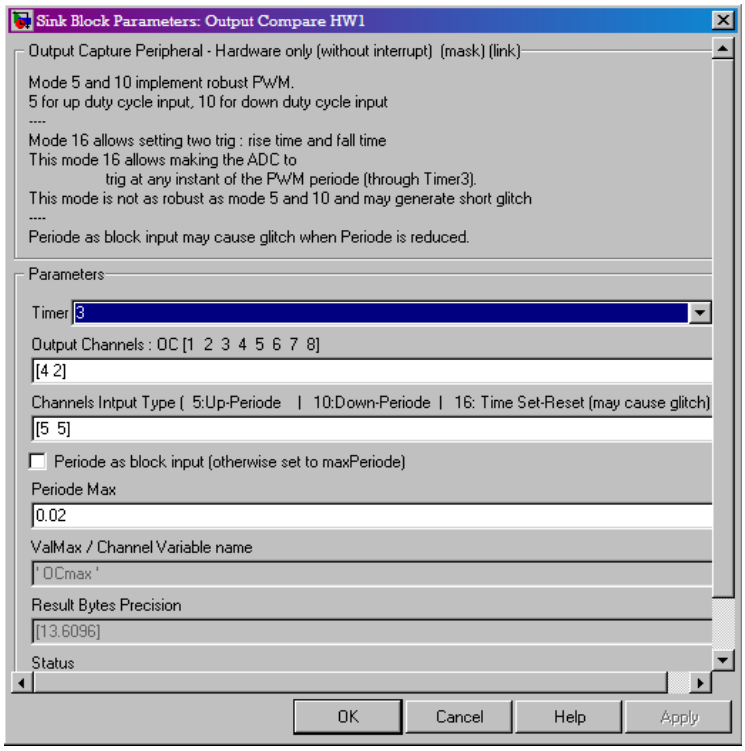
```
OC_PWM = 270*rad + 750
```

However, the theoretical and actual equation differ and so after testing this equation on the servos I obtained this equation:

```
OC_PWM = 450*rad + 825
```

which made the servo horizontal when the angle from the gyros was 0 and made the servo go to the correct angle when the gyros were rolled and pitched.

- Data Type Conversion: Before the number is sent to the output compare, it is converted from a double to a unsigned integer (uint) which is the format required by the output compare.
- Output Compare: I have explained much about the output compare in the preceeding bullets. The output compare outputs a PWM signal with a duty cycle determined from an input number ranging from 0 to 9999. A complete description of the output compare can be found at this link: http://www.kerhuel.eu/wiki/Block/Output_Compare_HW . I used the output channels 1 and 2 for testing from the explorer 16 development board. These channels correspond to RD0 and RD1 on the explorer 16 pinout board (which I obtained from Gabe, also found at this link Part Number: AC164126 - Prototype PICtail Plus Daughter Board). For the model that I programmed on the autopilot, I used output channels 4 and 2 for roll and pitch respectively (see image below). I used these channels because the autopilot harness (the autopilot harness is the mass of wires attached to the autopilot) had connections to those outputs on the autopilot control MCU (there are 2 dsPIC microcontrollers on the autopilot: the control MCU and the sensor MCU).
  I used the “5:Up-Period” option for the 'Channel Input Type' because this means that the number input to the Output Compare will specify how much time during the period that the signal will be high. (Note: because there are two inputs, you will have to enter the Channel Input Type as [5 5])
  Lastly, I set the 'Period Max' to 0.02 because the servos require the square wave to have a period of 20ms.



Electrical Hardware:

- **AVS (Aerial Video System)**: The AVS consists of a camera, a transmitter and a receiver. The transmitter transmits at 1.3GHz and consumes 300mW. Both the camera and the transmitter are powered by 12 volts. The camera sends video to the transmitter which remotely transmits the video data to the receiver at the ground station. The devices were purchased from www.rangevideo.com. Links provided below:
  transmitter (TX-1300-300): http://www.rangevideo.com/index.php?main_page=product_info&cPath=8∏ucts_id=99
  camera (KX171): http://www.rangevideo.com/index.php?main_page=product_info&cPath=6∏ucts_id=96
  receiver (RX-1300-standard): http://www.rangevideo.com/index.php?main_page=product_info&cPath=35_36∏ucts_id=68
  - Choosing this system: I was given a budget of $400. I found this AVS system through a online search for AVS's. However, I found out from Greg Horn that it is cheaper to purchase each item individually then as a system because the system includes certain cables and headers already prepared.
- **PowerBoard**: Power must be routed to the camera, transmitter, and servos on the gimble from a 11.1 volt battery onboard the aircraft. Both the camera and transmitter are 12 +- 10% devices so they can be powered directly from the battery.

However, the servos are controlled from 4.8 to 6 volts and so require that the 11.1 volt power be regulated to 5 volts. I created a schematic for this circuit in Eagle. There is a lite version of Eagle which can be downloaded for free online. This version is limited to creating double layer board but we only need single layer board. After creating the schematic, Max created a board layout from the schematic. Unfortunately, we encountered license issues with the board router software and I ended up making a prototype on perf board. It was important in making this proto board that there be no shorts because it was powered using a Li-polymer battery capable of supplying 29 amps of continuous current. If the battery was shorted, it would quickly overheat and could burst into flames.

Results:

- The gimble maintained constant horizontal (roll) and level (pitch) orientation as the plane (with mounted autopilot) was rolled and pitched during ground testing.
- The video system successfully transmitted video to the ground station.

Future Work:

- To add a third axis (yaw) to the gimble to allow tracking of locations during flight.
- The gimble could be strengthened by adding supports to the opposite sides of the pivots.

Acknowledgements:

I would like to express thanks to...

Mariano Lizarraga – for reliable comments, suggestions, and constructive criticism; for providing detailed guidance through the many problems faced in the project; for keeping the project goal in mind and suggesting the best course of action to accomplish the goal.
Dr. Gabriel Elkaim – for hosting this research and providing a challenging and enjoyable project: also for providing general guidance throughout the project.
Max Negolith – for board layout and routing.
Graig - for assistance with laser cutter
Greg Horn – for offering suggestions as to the design and progress of the gimble.