CMPE490/450 FINAL REPORT
DYNAMIC CAMERA STABILIZATION SYSTEM
GROUP 7

DAVID SLOAN
dlsloan@ualberta.ca
REEGAN WOROBEC
rworobec@ualberta.ca

DECLARATION OF ORIGINAL CONTENT

The design elements of this project and report are entirely the original work of the authors and have not been submitted for credit in any other course except as follows:

NMEA Checksum verification functions in C# GPS to Serial routines **[21]**

Slightly modified Parallax standard objects (FullDuplexSerial.spin, FloatMath.spin, Float32Full.spin, Float32A.spin, FloatString.spin)

__________________________________ __________________________________

Reegan Worobec	David Sloan
Apr 12, 2010	Apr 12, 2010

TABLE OF CONTENTS

APPENDICES

ABSTRACT

The goal of this project is to design a gimbal controller that maintains downward direction of a mounted camera given Inertial Measurement Unit (IMU) and GPS inputs, and should be able to hold its angular position relative to ground regardless of the pitch or roll of the aircraft. The system is also expected to return the aircraft's roll and pitch information on command. Provided there is enough design time, the device should have the ability to track a ground based GPS co-ordinate with the camera, or be able to hold the camera at a set angle with respect to the aircraft or ground.

FUNCTIONAL REQUIREMENTS OF PROJECT

While implementing an FPGA, the functional requirements of the project consist of ensuring level operation of the camera with respect to ground as the plane traverses through the air. It should also be able to report current information such as aircraft pitch and roll and position of the hobby servos. Tracking a specific GPS coordinate will be implemented, time permitting:

- Absolute and relative modes of operation
  - Absolute → input angle is with respect to plane
  - Relative → input angle is with respect to ground
- Reports plane roll and pitch information on command
- Can report raw servo position data and IMU readings on command
- **Optional (if design time permits)**
  - Can specify GPS coordinate to track
  - Report plane's current estimated position (debugging mode for point tracking)

Now that the final project is complete, it is safe to say that we were not able to adhere to all of our given requirements going into the project. The biggest requirement we failed to achieve was designing a system utilizing an FPGA. The switch to a Propeller chip was made approximately 3 weeks before the due date of the project due to the uncertainty of completion given our current FPGA difficulties. However, we were able to successfully complete most of the hardware components involved with the FPGA design, such as the servo controller and the interrupt controller, and were able to benefit from this in the fact that this made porting the functionality over to the Propeller design much easier.

As far as core operation goes, the ability to adjust the camera platform at both an absolute and relative angle was successfully completed as envisioned, and the platform adjusts to this angle at a rate of about 50 Hz. As well, through a serial terminal application, we are able to send the device reference vectors in which to stabilize the platform around.

Our original (optional) requirement of GPS point tracking was partially met. We have C++ routines written to parse the correct GPS information, given a valid NMEA sentence. As well, in Propeller's SPIN, we have code written to get these NMEA sentences from a serial connection. In conclusion, the only work missing would be to read the data from GPS into a serial connection, while porting the functionality from the C++ source into the SPIN source.

DESIGN AND DESCRIPTION OF OPERATION

The essential design of this project was to implement *Figure 1* located below:

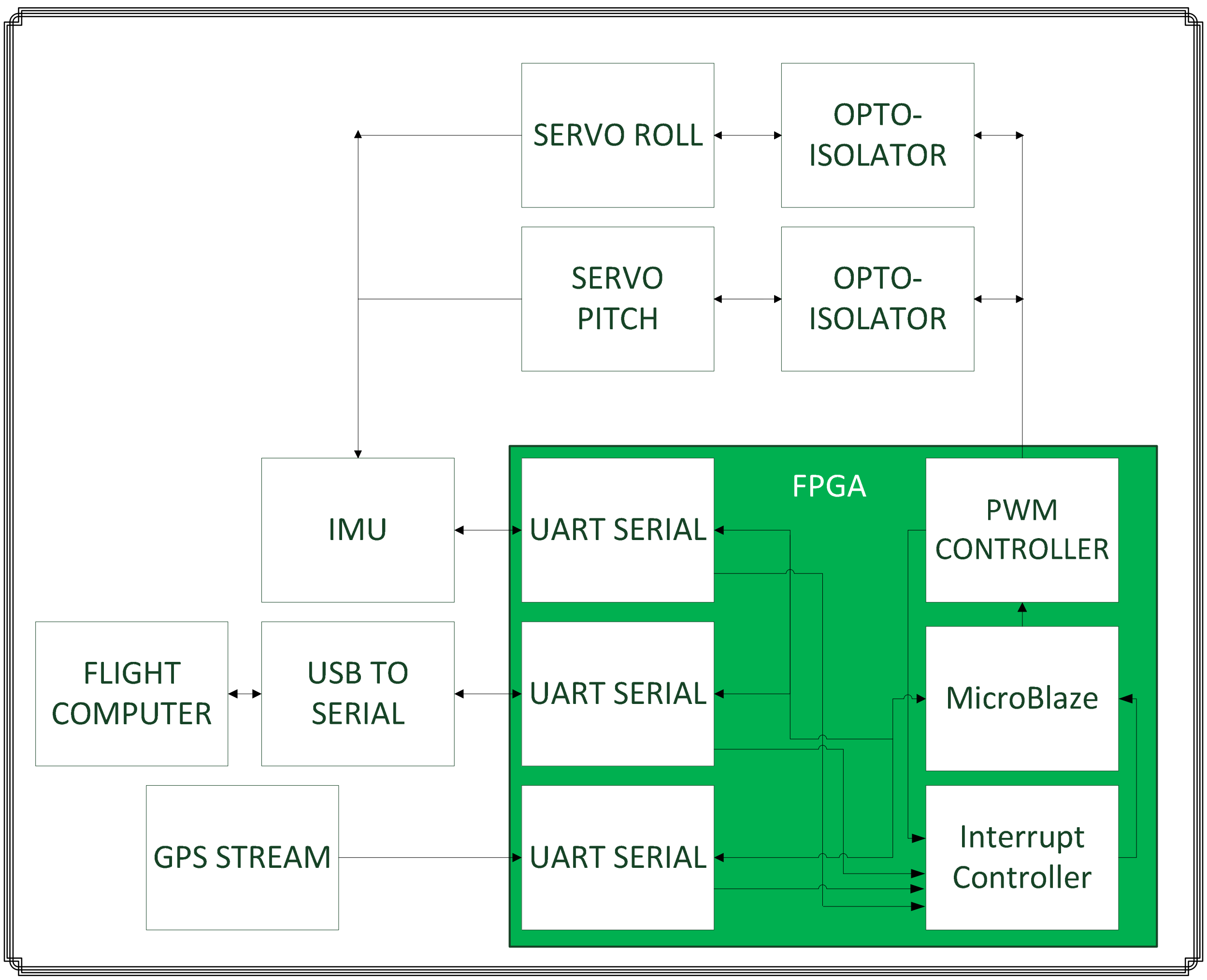

Figure 1 - Data Flow (FPGA) and Block Diagram

However, due to the platform switch mentioned in the previous section, our current design is like the diagram below:

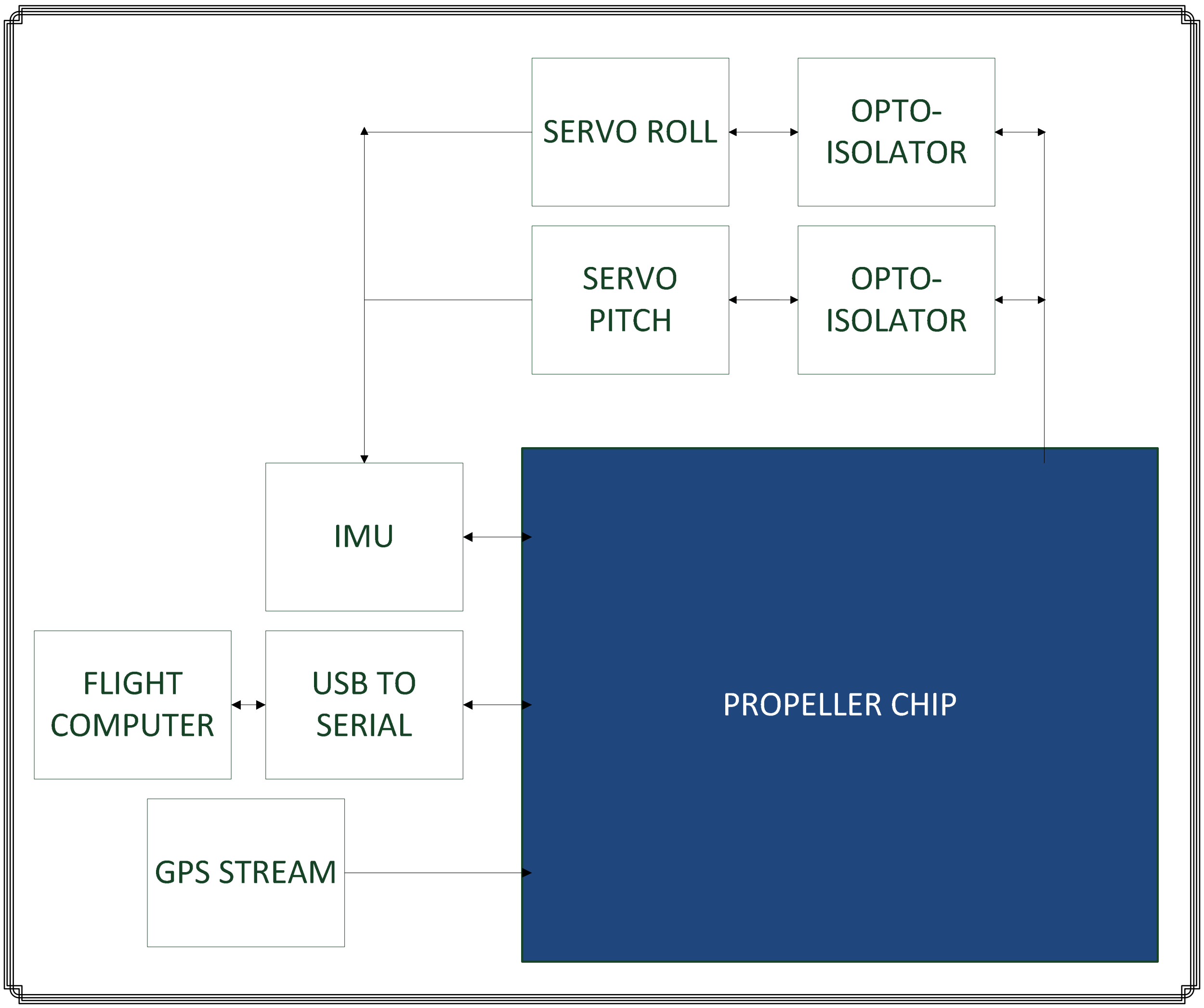

Figure 2 – Data Flow (Propeller) and Block Diagram

Since all of the hardware components from the original FPGA implementation are now taken care of in software (in separate processor cores) on the Propeller chip, the main focus is to follow the code hierarchy in *Appendix D*.

PARTS LIST

- Involved
  - IMU (CHIMU)[10]
  - Spartan 3e starter board[2]
  - Propeller Chip[22]
  - Spartan 3E breakout board[19]
  - Flash memory[14]
  - 50 MHz crystal oscillator
  - USB to serial interface (prop plug)
  - 1.2, 2.5, 3.3 Volt regulators[15, 16, 17]
  - Hirose DF-11 32pin connector x2[18]
  - FTDI FT232RL USB to Serial Chip[8]
  - Resistors
  - Capacitors

DATASHEET

Vital Statistics

Voltage

Servos	5V
IMU	5V
Propeller Chip	3.3V

Current

Servos	100mA (no strain)
	1800mA (w/ platform motion)
IMU	9mA
Propeller Chip	5mA

Calculation Type	**Measured**	**Calculated**
Power		
Servos	N/A	9000 mW
IMU	N/A	45 mW
Propeller	N/A	16.5 mW
		note: calculated from current measurements
Response Time	> 16 ms	20 - 40 ms
	note: 60 calculations / second	

Communication Protocol				
Message Header	**Message Body**	**Terminator**	**Returns**	**Description**
'x'	ASCII Floating Point String	'/r' (0x0d)	N/A	Set the x coordinate of the camera platform orientation vector
'X'	N/A	N/A	ASCII Floating Point String	Get the x coordinate of the camera platform orientation vector
'y'	ASCII Floating Point String	'/r' (0x0d)	N/A	Set the y coordinate of the camera platform orientation vector
'Y'	N/A	N/A	ASCII Floating Point String	Get the y coordinate of the camera platform orientation vector
'z'	ASCII Floating Point String	'/r' (0x0d)	N/A	Set the z coordinate of the camera platform orientation vector
'Z'	N/A	N/A	ASCII Floating Point String	Get the z coordinate of the camera platform orientation vector

SOFTWARE DESIGN

The software operation of our project is detailed in Figure X of *Appendix D (also shown below)*. Essentially, things to note include that our Math library utilizes 2 of the processing cores, due to the fact that there are no floating point operations in hardware or integer multiplication and division.

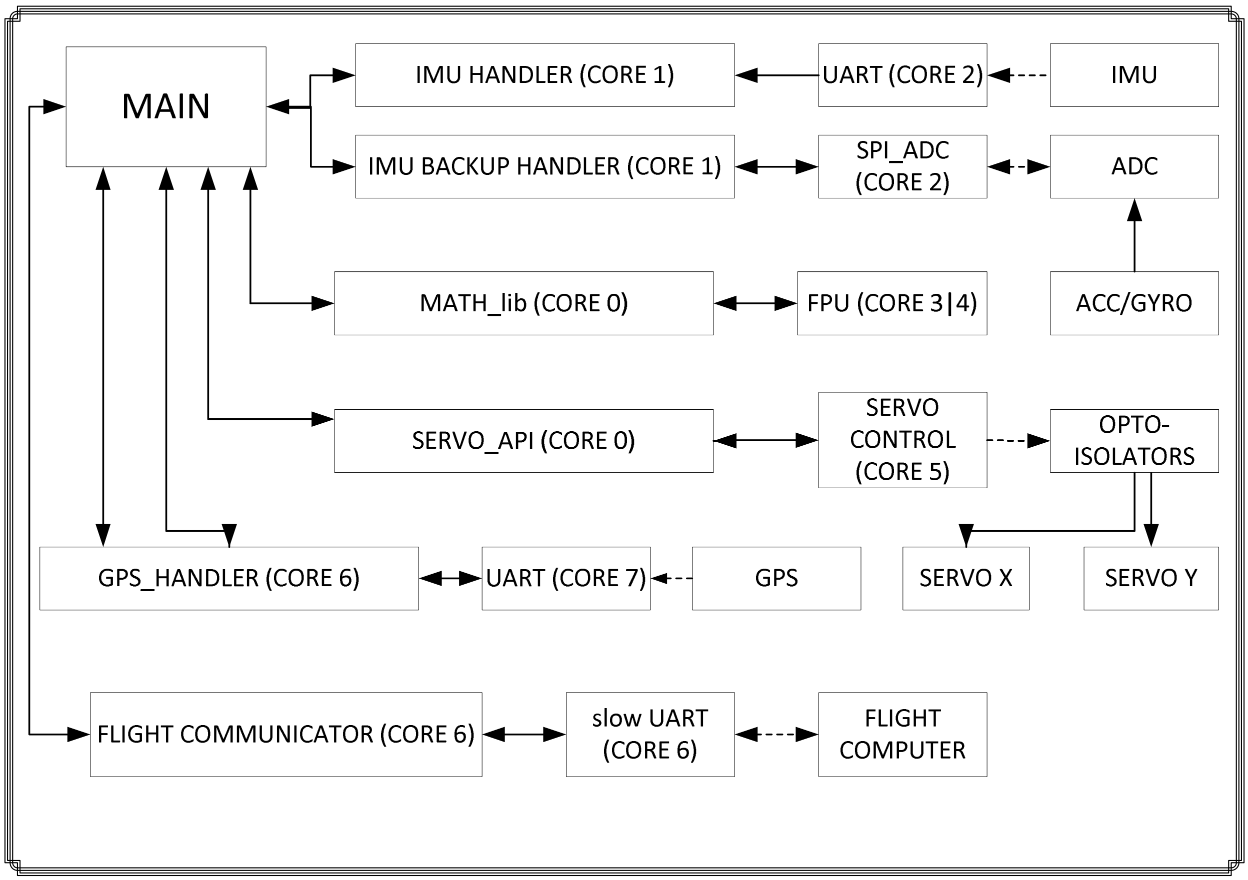

Figure 3 – Software Flow (Propeller/Final Implementation)

Another thing to note is that the current processor usage could be decreased by optimizing some of the communications routines, which can be done a later time.

HARDWARE DESIGN

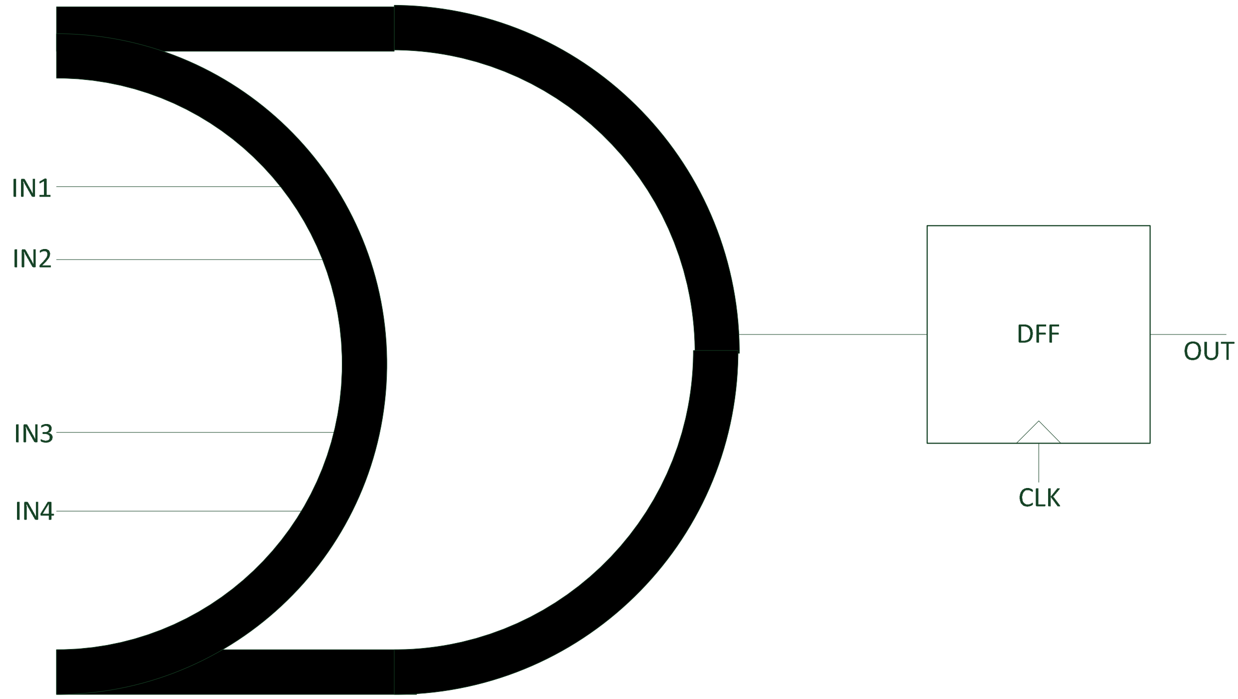

Figure 4 – Interrupt Controller (Hardware)

Above is our interrupt controller, designed in hardware, before the switch over to the Propeller chip. Although no interrupts are currently used in the Propeller design, the above configuration simply triggered an interrupt depending on which input was requesting one during a rising clock edge.

The other major component designed was the PWM controller (no schematic included) which generates specialized pulse width modulation signals for controlling hobby servos. It has programmable limits for minimum/maximum pulse widths and fires an interrupt after the maximum pulse width edge has occurred.

TEST PLAN

Software:

- Get LEDs to blink using EDK on starter board
- Get DB9 RS-232 port communicating to PC (Echo)
- Ramp PWM controller and watch servo movement
- Echo GPS input to computer (Test multiple UART connections)
- Simple application to control servo positions from PC and read IMU/GPS data
- Re-run servo tests after their calibration is finalized
- Manually move gimble and verify camera's direction

Hardware:

- CMOS level UART connection to PC (Echo)
- Ensure PWM is working
  - Hardwire PWM controller to fixed position
  - Write random servo positions
    - Ensure proper operation
- Echo IMU
  - After GPS echo to ensure proper software functionality
  - Test sending commands and receiving results from IMU
- Similar tests to be used after PCB assembly

CITATIONS

[1] SERVO-SIGNAL SPECIFICATIONS
http://www.msu.edu/~markeyna/PWM.pdf
[2] SPARTAN SE3 STARTER BOARD USER GUIDE
http://www.digilentinc.com/Data/Products/S3EBOARD/S3EStarter_ug230.pdf
[3] SPARTAN SE3 STARTER BOARD SCHEMATIC
http://www.digilentinc.com/Data/Products/S3EBOARD/S3E%20Starter_sch.pdf
[4] MICROBLAZE FLASH BOOT LOADER (APP NOTE)
http://www.xilinx.com/support/documentation/application_notes/xapp482.pdf
[5] REMOTE EMBEDDED CORE FIELD PROGRAMMING
http://www.xilinx.com/products/design_resources/config_sol/s3/config_s3e.htm#field
[6] SPARTAN S3 3.3V CONFIGURATION (APP NOTE)
*Could not find link. Email us for direct PDF access
[7] SE3 FAMILY DATASHEET
http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf
[8] SERIAL TO USB DATASHEET
http://www.ftdichip.com/Documents/DataSheets/DS_FT232R.pdf
[9] OPTOISOLATOR DATASHEET
http://pdfdata.datasheetsite.com/web/18848/4N25.pdf
[10] CHIMU USER MANUAL
http://www.ryanmechatronics.com/index_files/ProductDetailCHIMU.htm
[11] USB NODE SCHEMATIC
*Could not find link. Email us for direct PDF access
[12] VENUS GPS DATASHEET
http://www.sparkfun.com/datasheets/GPS/Modules/Skytraq-Venus634FLPx_DS_v051.pdf
[13] SPARTAN S3 SERIES CONFIGURATION USER GUIDE
http://www.digilentinc.com/Data/Products/S3EBOARD/S3EStarter_ug230.pdf
[14] XILINX FLASH MEMORY DATA SHEET
http://www.xilinx.com/support/documentation/data_sheets/ds123.pdf
[15] 1.2V REGULATOR DATASHEET
http://search.digikey.com/scripts/dksearch/dksus.dll?Detail&name=MCP1700T-1202E/TTCT-ND
[16] 2.5V REGULATOR DATASHEET
http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=TC10152.5VCT713CT-ND
[17] 3.3V REGULATOR DATASHEET
http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=TC10553.3VCT713CT-ND
[18] HIROSE DF-11 SEIRES CONNECTOR
http://www.hirose.co.jp/cataloge_hp/e54305002.pdf
[19] SPARTAN S3E BREAKOUT BOARD SCHEMATIC
http://www.sparkfun.com/datasheets/DevTools/FPGA/Spartan_3_Breakout_Schematic.pdf
[20] NMEA GPS STANDARD
http://www.gpsinformation.org/dale/nmea.htm#GGA
[21] NMEA Checksum Calculation
http://webcache.googleusercontent.com/search?q=cache:http://www.codepedia.com/1/Calculating%2Band%2BValidating%2BNMEA%2BChecksums
[22] Propeller Manual
http://www.parallax.com/Portals/0/Downloads/docs/prod/prop/WebPM-v1.1.pdf

APPENDIX A: QUICK START MANUAL

Below is a point form implementation of how to get our project/code up and running again:

- Plug in power to the correct sources
    - 3.3V to the green plug
    - 5V to the red plug, labeled 5V
    - 'Servo Power' to the unlabeled red plug
        - The voltage is specified by the servos used
- Ground accordingly
- Plug flight computer into CMOS-level UART on top of propeller board as labeled
    - If using a serial terminal, xyz set orientation vector
        - E.g. "y 0.7" (enter)
            - Sets y component of camera platform orientation vector to 0.7
    - XYZ get camera platform orientation vector
- Servo calibration
    - Servo_v2.spin
        - Change y/x minf/maxf for angles
        - y/x min/max for min/max servo pulse widths

APPENDIX B: FUTURE WORK

There are quite a few things to add to the functionality of our device. First, the GPS point tracking capability can/should be implemented. This is touched upon a bit in *Appendix D*, but essentially what needs to be done is a port of the C++ GPS functions into the proprietary Propeller language, SPIN. This also includes feeding the velocity data from the GPS to the IMU.

Second, optimizing the mathematical calculations is important. Right now currently it takes approximately 20ms per calculation. This would be done by putting the matrix math into Assembly, and doing away with some of the floating point calculations done in SPIN (which is an interpreted language).

Synchronization of the IMU with the and the servo controller, such that the pulse is starts immediately after the math is completed from the previous IMU update.

Additionally, we could use the IMU in SPI mode, and take the 4 points per servo pulse from the 200 Hz signal to project the expected position of the airframe when the servo is actually updated.

APPENDIX C: HARDWARE DOCUMENTATION

STABILIZATION PLATFORM OPERATION

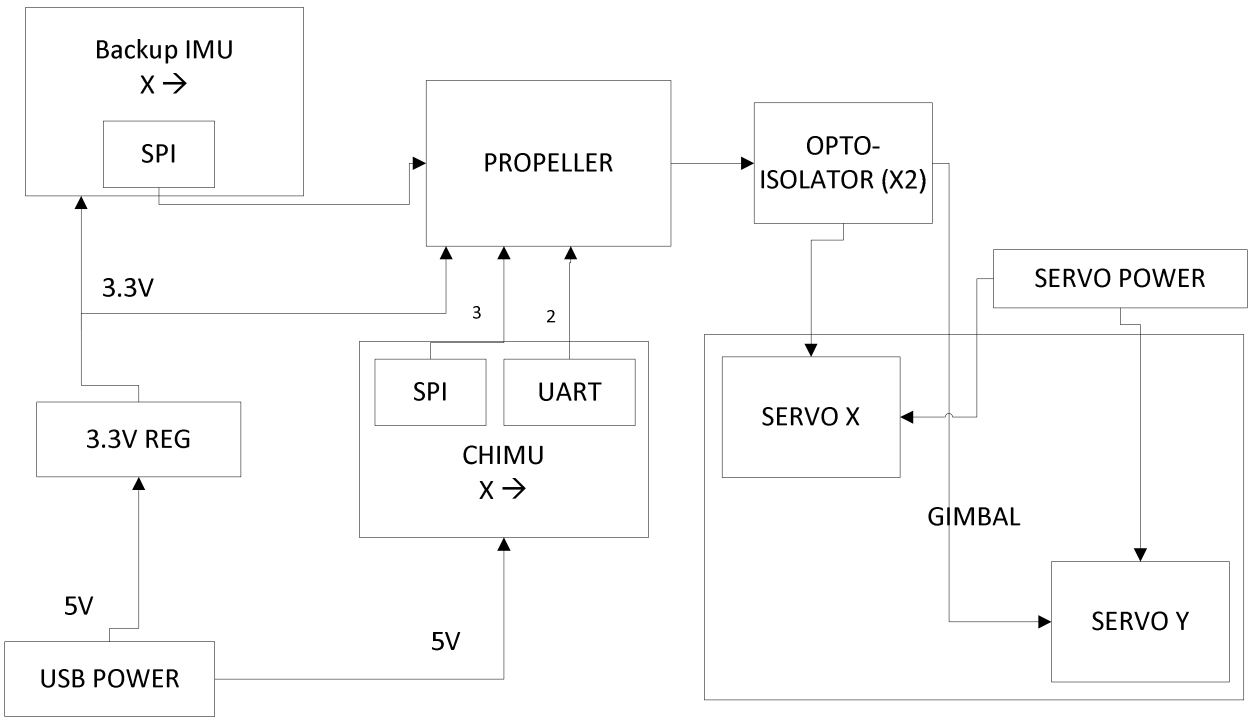

Figure X – Hardware Flow Diagram

Above is a hardware flow representation of our project after the switch to the Propeller from the FPGA.

EAGLE CAD SCHEMATIC (BOARD LAYOUT: INC)

Although this design never made it to fruition, this was our attempt at routing on Eagle CAD with the free version's restrictions. Daniel St. Pierre from UAARG put a lot of work into trying another board layout in Protel's software suite, but recurring problems with hardware components ultimately halted the creation of a PCB.

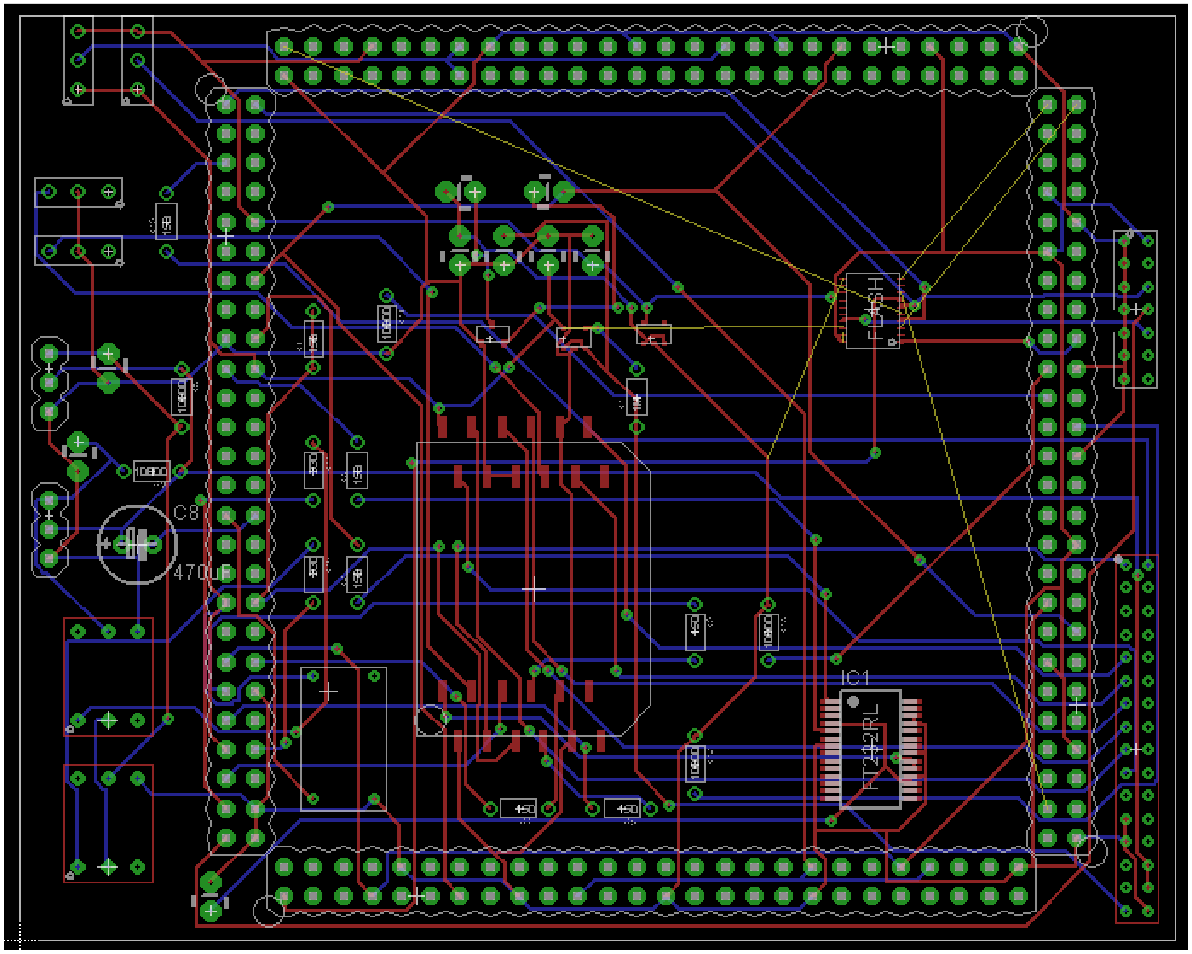

Figure X – Eagle CAD (.brd) Representation

APPENDIX D: SOURCE CODE

SOFTWARE FLOW

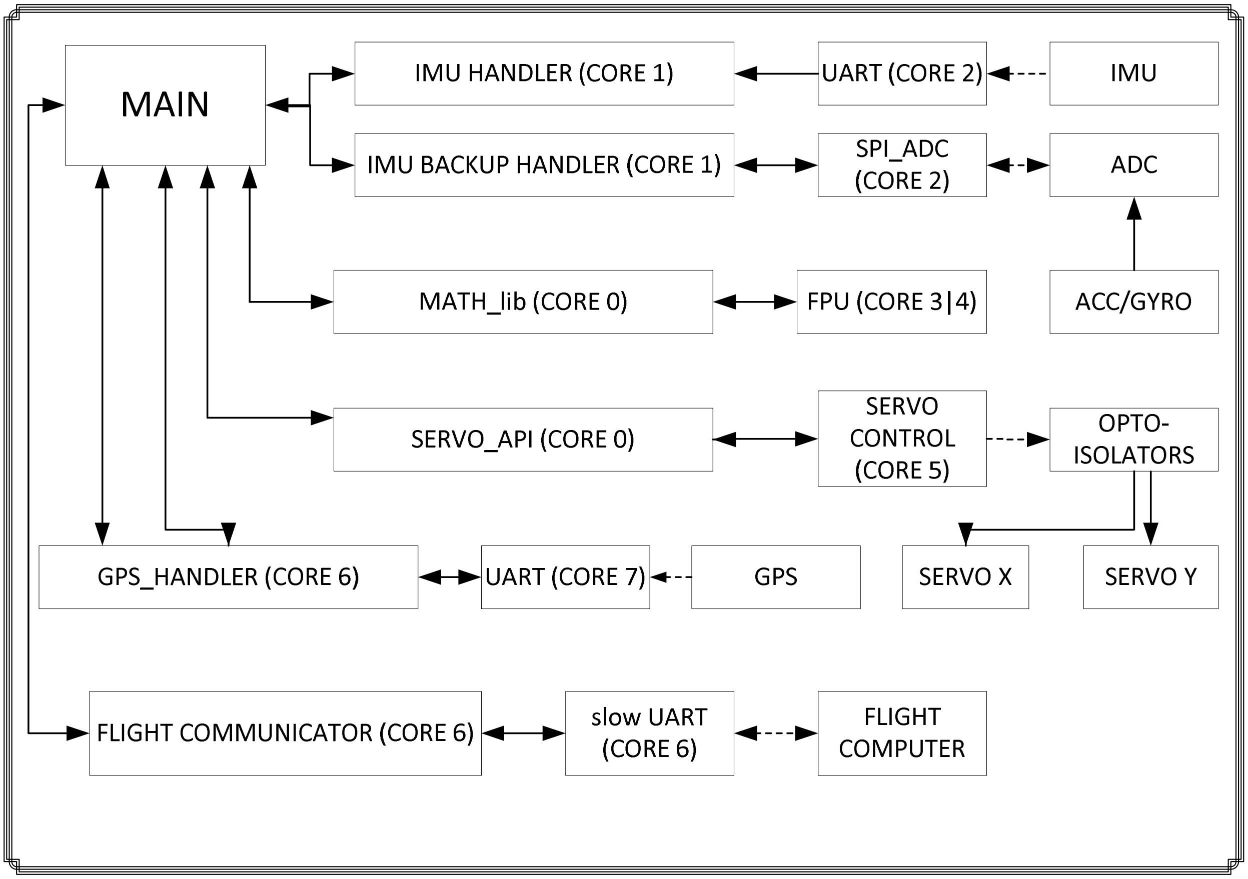

Figure X – Software Flow (Propeller/Final Implementation)

CODE HIERARCHY

Propeller v3

- CameraStabilization.spin
  - Servo_v2.spin
    - FloatMath.spin
  - CHIMU_v1.spin
    - FullDuplexSerial.spin
    - FloatMath.spin
  - IMUMath_v1.spin
    - Float32Full.spin
      - Float32A.spin
  - FullDuplexSerial.spin
  - FloatString.spin
    - FloatMath.spin

Propeller v2

- CameraStabilization.spin
    - Servo_v2.spin
        - FloatMath.spin
    - CHIMU_v1.spin
        - FullDuplexSerial.spin
        - FloatMath.spin
    - IMUMath_v1.spin
        - Float32Full.spin
            - Float32A.spin
    - FullDuplexSerial.spin

Propeller v1

- CameraStabilization.spin
    - Servo_v2.spin
        - FloatMath.spin
    - CHIMU_v1.spin
        - FullDuplexSerial.spin
        - FloatMath.spin
    - IMUMath_v1.spin
        - Float32Full.spin
            - Float32A.spin
    - FullDuplexSerial.spin

Propeller Servo_2_Test2_center

- Servo_2_Test2_center.spin
    - Servo_v2.spin
        - FloatMath.spin
    - FloatMath.spin

Propeller Matrix Math

- IMUMath_test.spin
    - Float32Full.spin
        - Float32A.spin
    - FullDuplexSerial.spin
    - FloatString.spin
        - FloatMath.spin

Propeller CHIMU

- CHIMU_Test.spin
    - FullDuplexSerial.spin
    - FloatString.spin
        - FloatMath.spin
    - CHIMU_v1.spin
        - FullDuplexSerial.spin
        - FloatMath.spin

Propeller BackupIMU.spin

- ProtoBoardBackUpIMUTest.spin
    - FullDuplexSerial.spin
    - ADC_INPUT_DRIVER.spin

Propeller GPS (Not Complete)

- GPS_v1.spin

Xilinx SDK

- main
    - UART
    - SERVO
    - SPI
    - IMU

GPS (C++)

- GPSparser

Test (MicroBlaze)

- TestApp_microblaze.cpp

Test (Xilinx SDK)

- main
    - UART
    - SERVO
    - SPI
    - IMU

*All source code located on .zip in email *

APPENDIX E: TOOLS

The list below yields all of the tools we have used to complete this project:

- Xilinx EDK
- Propeller Tool
- Parallax Serial Terminal
- Visual Studio 2008
- Visual Studio 2010
- Visio (Report Diagrams)