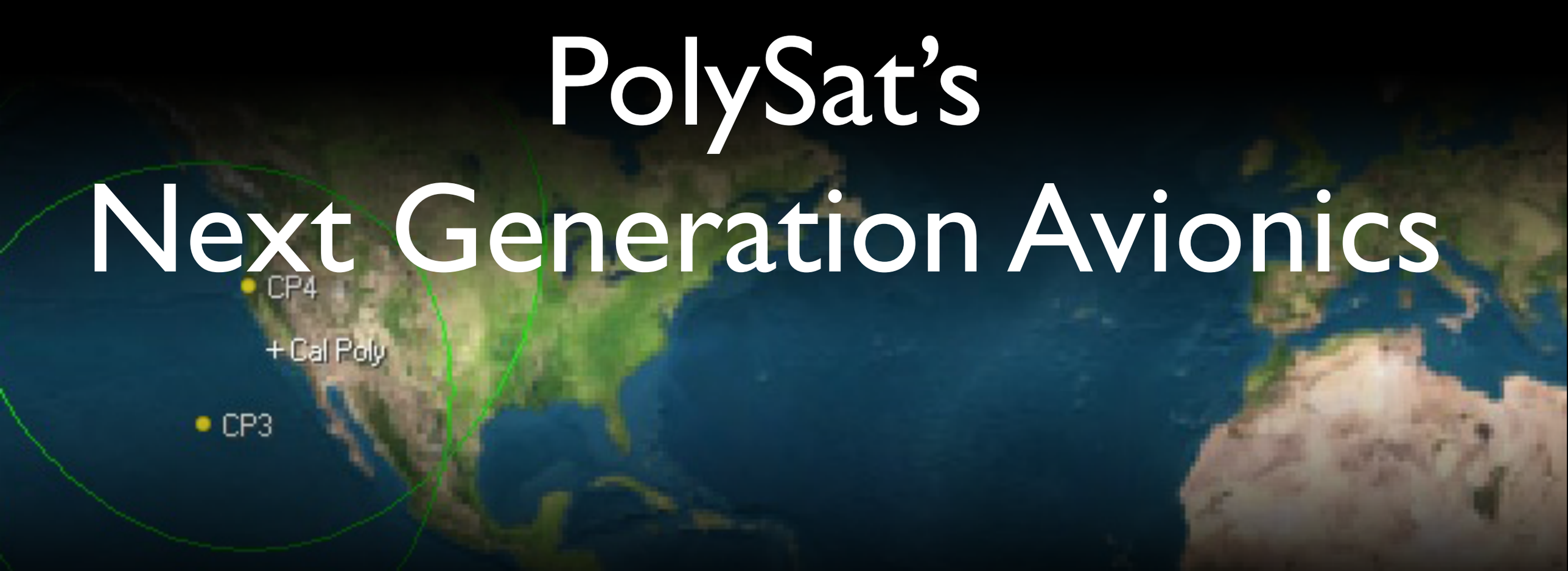
PolySat's Next Generation Avionics



Presented By:

Greg Manyak

Avionics Software Project Lead

PolySat California Polytechnic State University Satellite Project

Cal Poly

CubeSats Are Evolving

- Cal Poly and the CubeSat community is moving past educational experiments
- Now that we have a significant launch history, CubeSats are being looked at for more significant and involved missions
  - The most exciting part is these more complex missions are actually being funded, too!
- However, payload volume and computing requirements are exceeding that of the typical CubeSat bus
- New bus design approach can help solve these problems

Avionics Discussion

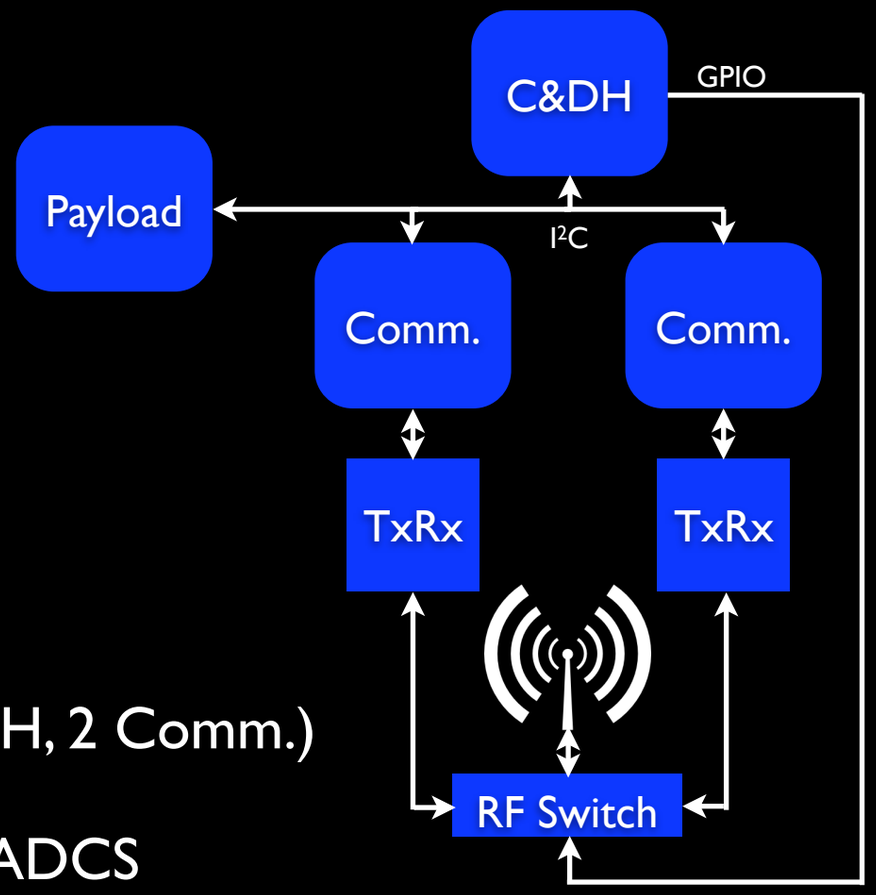
- General Avionics Requirements
- PolySat's Legacy Bus Design
- Legacy Bus Results
- Avionics Design Principles
- New Avionics Design
- Avionics Comparison
- Fault Tolerance

Avionics Requirements

- Minimize volume
- Reasonable amount of power consumption based on performance
- Modular, reusable software
- Enables quick mission turn-around time

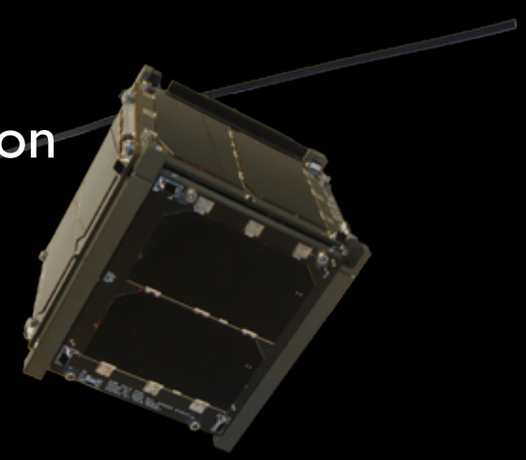
Legacy Bus Design

- Fault Tolerant Architecture:
  - Redundant communication subsystems
  - “Smart Fuse” custom circuit for latchup protection
  - External watchdog chip for each controller
- 3 PIC18 4MHz single-purpose microcontrollers (1 C&DH, 2 Comm.)
  - Full-custom software with telemetry collection and ADCS
- Typical payload requires own controller



Legacy Bus Results

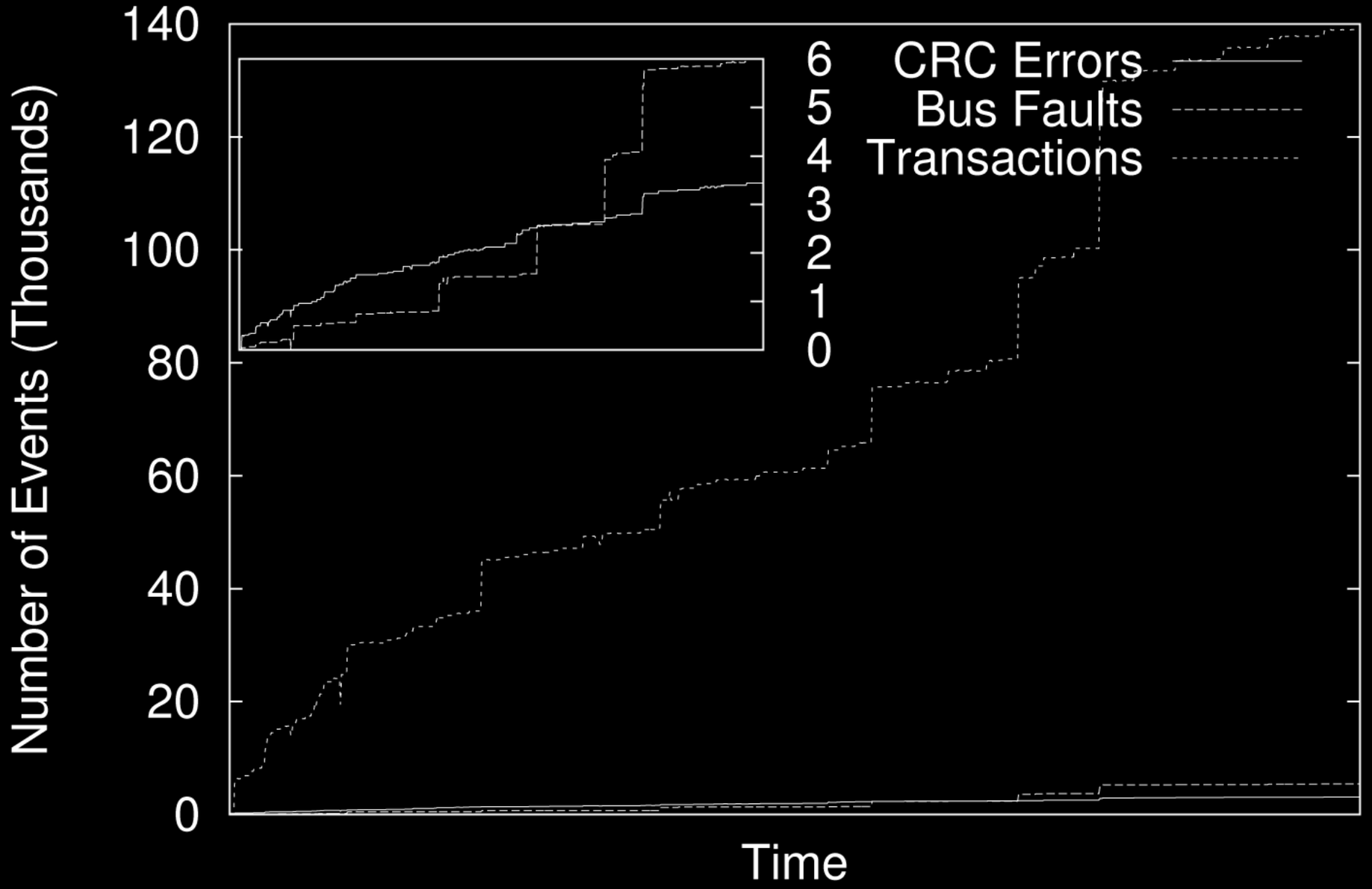
- CP3 -- Successful launch, DNEPR 2 in April 2007
  - Basic bus operations successful, but receive sensitivity limited commands
- CP4 -- Successful launch, DNEPR 2 in April 2007
  - C&DH failure soon after reaching orbit, only comm. processor commands functional
- CP6 -- Successful launch, Minotaur I in May 2009
  - Refly of CP3 with improved receive sensitivity
  - Greater operational success; validated de-tumbling algorithm
  - Unfortunately, same result as CP4: C&DH failed, ending mission



Failure Investigation

- The common believed cause for the CP4 failure was the I2C bus between the comm. and C&DH microcontrollers
  - On CP6, I2C telemetry was added, in attempt to validate this assumption
  - A CRC byte was also added to comm. to C&DH messages for transaction failure detection
- CP6 telemetry showed a constant ~8% I2C bus transaction failure rate
  - Suggests I2C is not necessarily the source of blame
  - The fact that the comm. and C&DH link failed again though does suggest this single point of failure is a significant problem

CP6 I2C Telemetry



I2C bus event data from CP6. Inset in upper left shows only the CRC error and bus fault event counters. Time progresses left-to-right, but not proportionally due to periods of time when no data was downlinked.

Lessons Learned

- Always implement a simple reset command!
- Hardware redundancy may not always provide sufficient fault tolerance
- I2C may not be the sole source of bus failure
- Gather as much telemetry available, even if not obviously usable

Avionics Design Principles

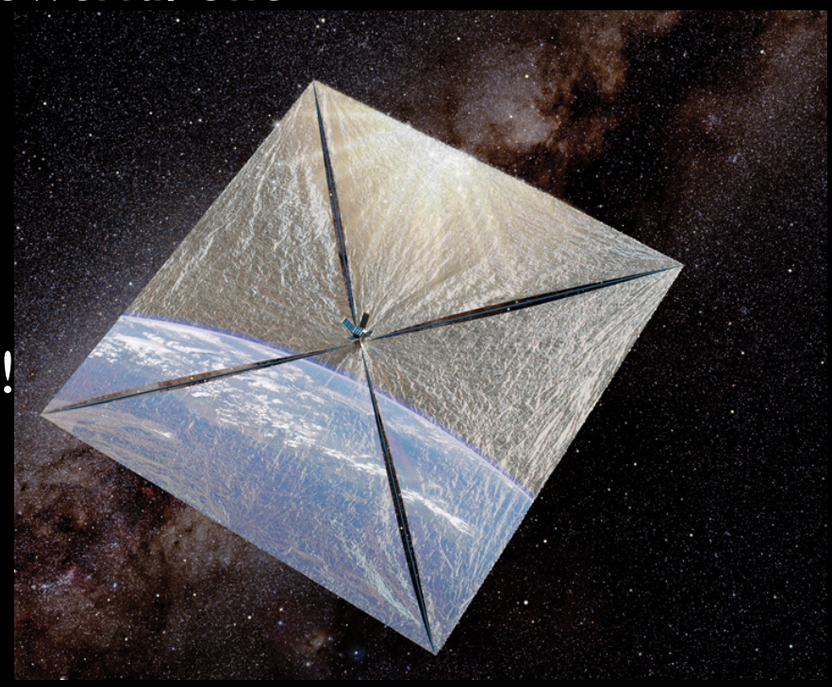
- Prefer software to hardware modularity
  - "Easier" to throw hardware at the problem
  - Reduces available payload volume
- Minimize hardware redundancy
  - Too expensive: adds power, volume, price and complexity
  - Not enough operational history to identify where it's needed most
  - Cost of a single CubeSat encourages redundancy through backup flight units
    - e.g. CP3 and CP6, CP2 and CP4

Change Payload Interface Philosophy

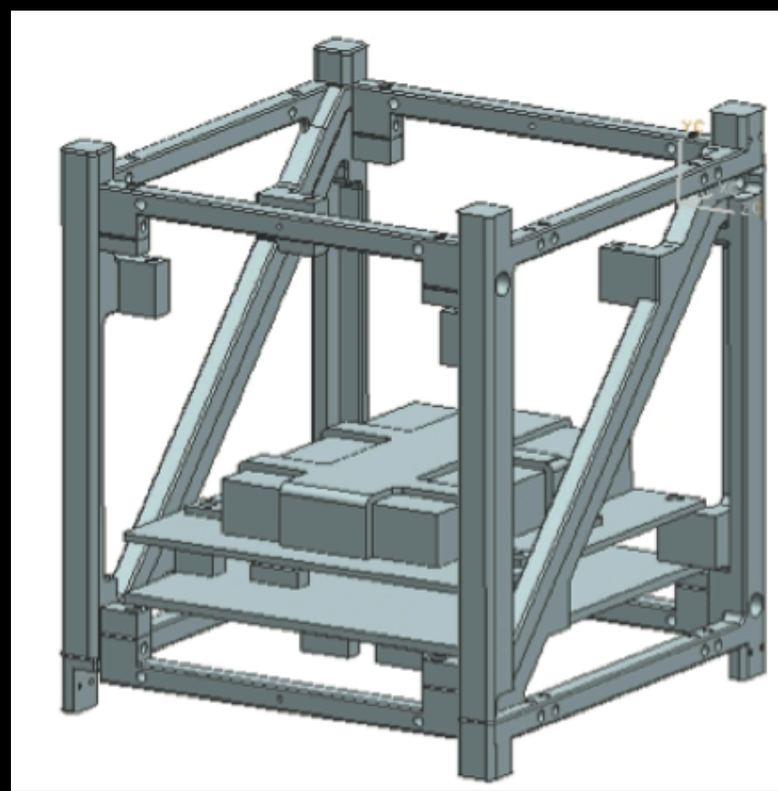
- Eliminate payload controllers!
  - Most designs include a controller for the payload
    - Appears to be the "clean" option, but adds substantial complexity!
    - Custom interface protocol typically required and code/hardware rarely reusable
  - Instead, payload can utilize a variety of direct interfaces from C&DH controller
    - Similarly simple hardware design, but minimizes interfacing requirements
    - Tradeoff is complexity of main controller is increased

New Avionics Design

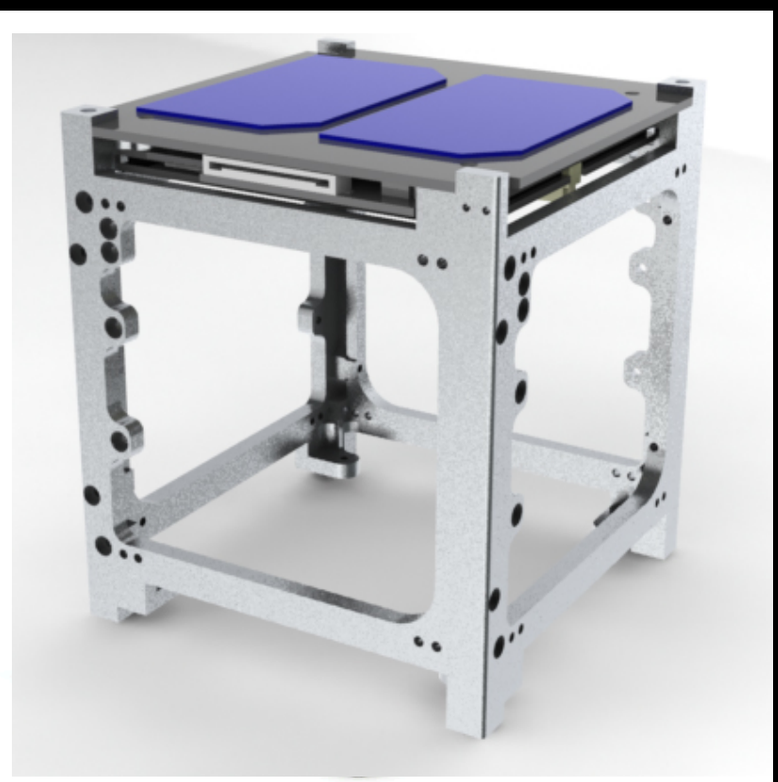
- Culmination of design principles, requirements, and past experience
- Basis for two current missions: (3U) LightSail, and (1U) CP7
- Consolidation of standard bus electronics (EPS and C&DH) onto one board
- 2 daughtercard expansion slots for transceiver(s) and/or mission specific electronics
- Eliminates 2 of 3 processors, but necessitates more powerful one
- C&DH built around 32bit ARM9-based Atmel chip
  - Supported by SDRAM and NAND flash
- Resulting design only consumes approximately .1 liter!



Design Comparison



1st Gen Design



2nd Gen Design

Revision	Processor	Clock (MHz)	Volatile Memory	Non-Volatile Memory	OS	Volume	Idle Power (W)
1st	3x PIC18	4	3.75 KB	256 KB	Custom	.25L	0.2
2nd	AT91SAM9	400	64 MB	528 MB	Linux	.1L*	0.34

*without batteries

New Avionics Software

- Significant software architecture change: new microprocessor runs full Linux!
  - Instant learning curve reduction thanks to Cal Poly Linux coursework
  - Large amount of pre-existing software to leverage for new design
- Custom distribution of Linux via use of open source tool Buildroot
- Each primary function relegated to a process
  - e.g. Beacon, Data Logging, System Management, S/W Watchdog, Communications
- Each process utilizes a common library to provide frequently used capabilities
  - Event scheduling, .config files, debug interface, inter-process communication, etc.
- Drivers implemented in both user space and kernel

New Avionics Fault Tolerance

- Primary mechanism to recover from single event upsets: reset satellite
  - Hardware + software watchdogs, and long-term hardware counter
- Radiation tolerant phase change memory for multiple basic software images
  - Custom bootstrapper to verify integrity of image before boot
- NAND Flash ECC for file system error correction and bad block detection
  - “Free” from existing Linux driver and file system code!
- Robust message checksums for IPCs and commands

Implementation Progress

- Revision 2 of hardware ordered last week
  - Revision 1 functional w/ wire mods and minor component additions
- Majority of low-level software functionality has been demonstrated
- First release of primary software libraries in April, 2011
- Engineering model to be completed by June, 2011

Conclusion

- We see this avionics system as an enabling technology for CubeSats
- Software modularity can be a wonderful thing
  - However, needs to be considered at the beginning of the design!
- PolySat's software team is interested in discussing Linux designs in detail

Questions?