A Preliminary Architecture for a Modular and Scalable Edge Computing System for Small Spacecraft

Kelly Williams

Junior Undergraduate Researcher

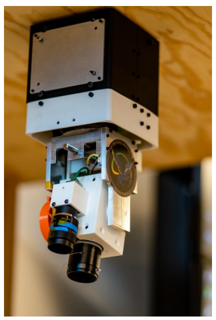
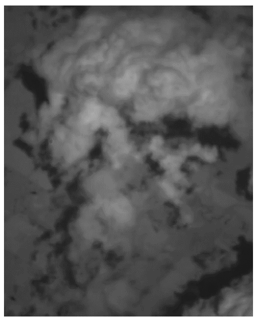
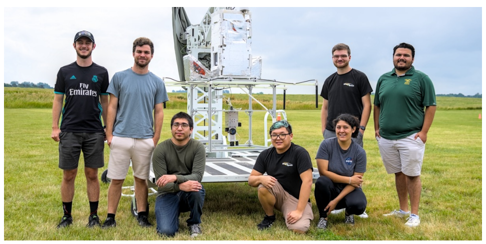
Bronco Space

California State Polytechnic University, Pomona

For Public Release

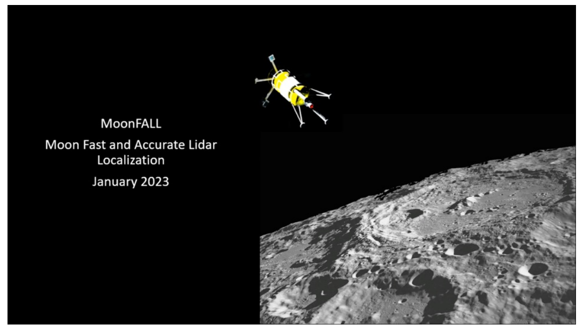
Work Conducted in the Bronco Space Lab

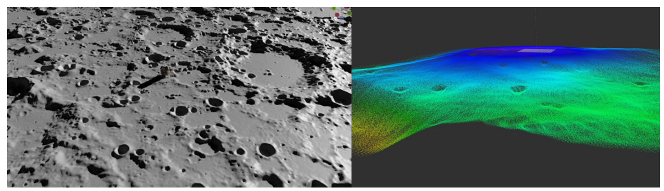
Bronco Ember

- Autonomous Wildfire Detection System
- Computer Vision and Edge Computing for Autonomous Remote Sensing

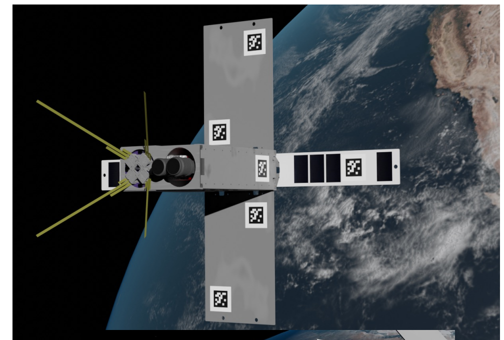
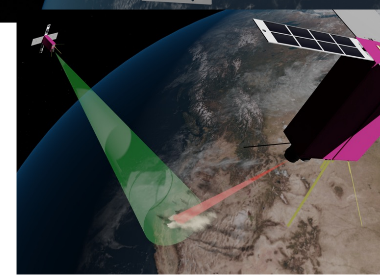
MoonFALL





- Autonomous mapping, precision landing, and hazard avoidance system.
- ML model trained in a digital twin and tested sub orbitally

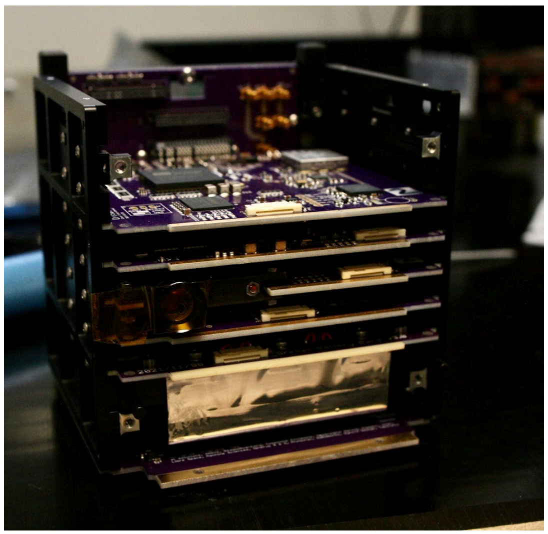
CubeSat Development

- Open-source satellite development and investigations into low cost rapidly deployed systems

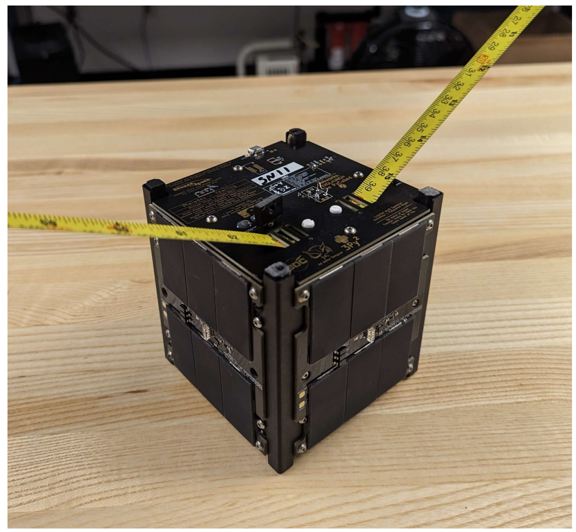
The Three US Open Source CubeSat Platforms

OreSat



- Modular Card Cage System
- OreSat Power Domain and Backplane are unique features
- Resilient but high cost

PyCubed / PROVES



- Single Board Computer architecture
- Minimal overhead is the goal
- Less resilient but very low cost

Artemis CubeSat

- Most traditional CubeSat architecture
- PC104 stackup and Raspberry Pi
- Heavily supported by NASA

SCALES Pedigree: The OreSat C3

Bronco Space

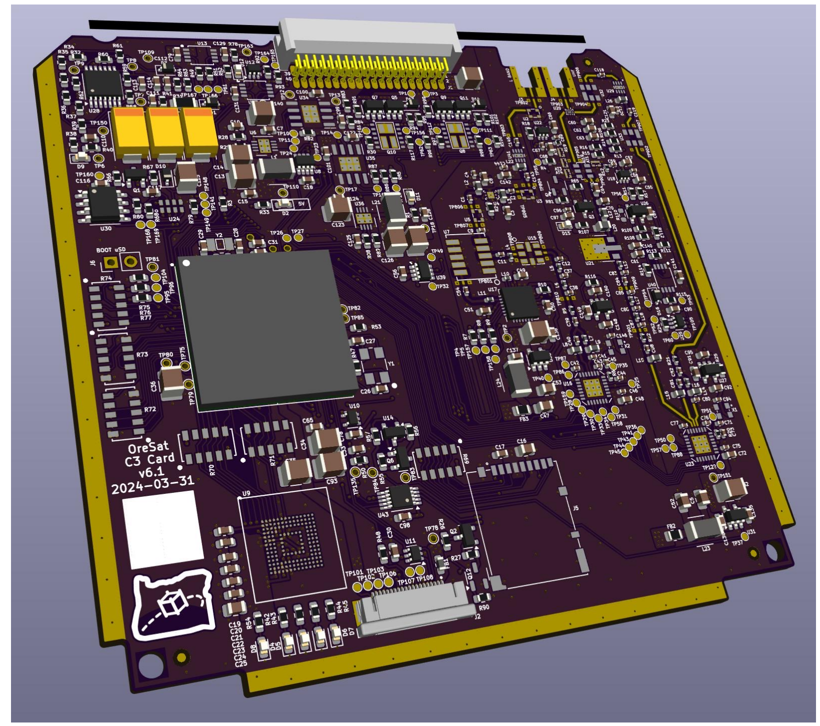
Key Metrics

- Octavo OSD335x-SM SIP AM335X-based Cortex A8 Microprocessor
- TI TLV1042 comparator for radiation tolerant watchdog
- 1 Mbit of FRAM
- 16 GB eMMC flash data storage
- UHF and L-Band Transceivers

Key Features

- Close to a single board solution for satellite command and control.
- Radiation tolerant watchdog scheme enables use of readily available COTS components.
- OreSat Power Domain supports scalability.

The C3



What is SCALES?

Spacecraft

Compartmentalized

Autonomous

Learning and

Edge-Computing

System

- How do you create a platform that people want to use, is adaptable, can auto-regulate different AI models, and allow the AI models to take control of the spacecraft without things going wrong?
- Integrated hardware / software system that creates a reusable environment for the execution and training of Machine Learning (ML) algorithms on-orbit.
- Integration of Commercial Off the Shelf (COTS) edge computing hardware that enhances on-orbit computing and a reusable software architecture that provides reliability and mission assurance.

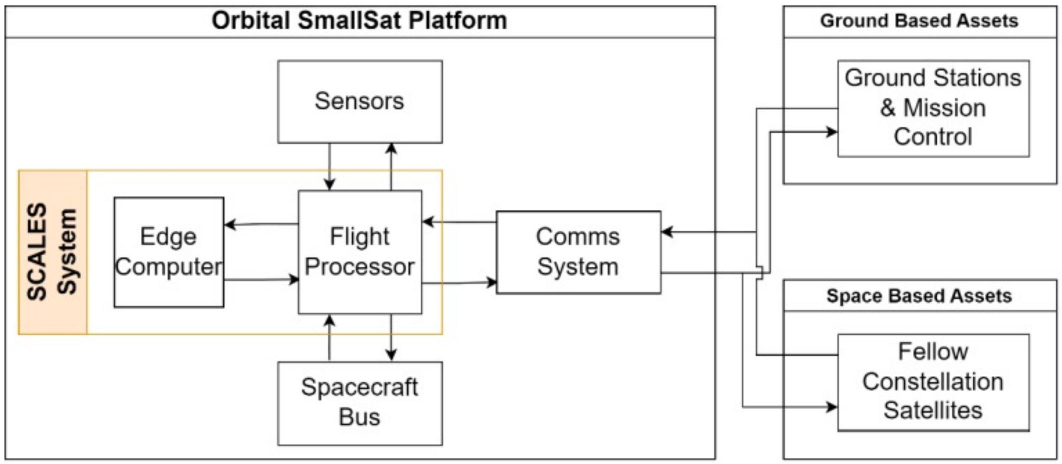
Relevance and Impact to Industry

- Compute performance of flight-ready hardware
- Lack of reliable, reusable software architecture for hardware management
- Physical and fiscal constraints of autonomous functionality in SmallSats
  - Need more cost-effective, compact edge-computing hardware and software solutions
- Most flight software uses FPGA's but they are expensive, difficult and time-consuming to code, and cannot be updated once deployed
  - They are also only good for one specific use, which doesn't hold up well over time or for trying to broaden to different applications and tasks
  - Not very many radiation-safe hardware alternatives
- AI models have only about 85% accuracy on Earth, and in space that drops drastically so development can become unsafe and unreliable in a space environment if the models were given full control.

Potential Use-Cases

- Object Detection
  - Description: Remote sensing satellite using an CV algorithm to process images in real time and make decisions based on image analysis. Real time detection and tracking.
  - Inputs: Images
  - Output: target identification scores, movement requests
  - Control Request: motion of spacecraft to get a better view
  - Risk: Unnecessary motion can be a waste of fuel, extraneous images can be a waste of storage
  - SCALES Response: Feedback on identification scores, limitations to how much it will allow the system to move.
- Navigation
  - Description: Navigation algorithms to use edge-computing to allow use of less-complex sensors. Hazard avoidance, fuel saver, time reducer.
  - Inputs: Altitude sensors and cameras
  - Outputs: state vectors, terrain maps, other navigational aids
  - Control Request: position and velocity of spacecraft
  - Risk: Incorrect algorithm outputs can cause dangerous actions. System lag – vehicle "gets ahead" of the controls
  - SCALES Response: Analyze output history, assess risk, limitations on movement, possibly flag user.

SCALES Architecture

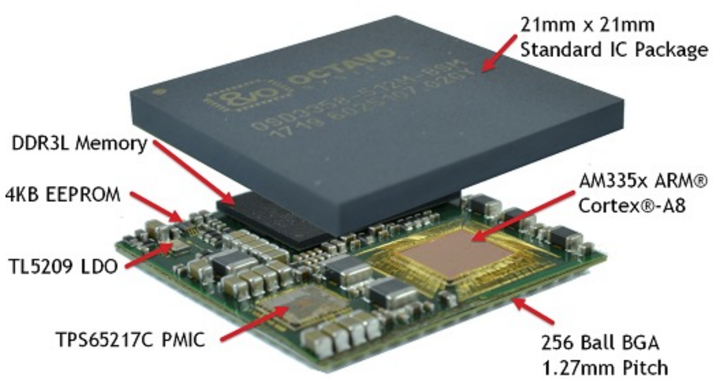


- Two main processor boards will work together:
  - Space qualified flight processor
    - NASA's F Prime software
    - Monitor overall system health, performance, and actions
  - Edge-computing processor
    - Machine Learning models
    - Bridge to F Prime to interface system inputs/outputs

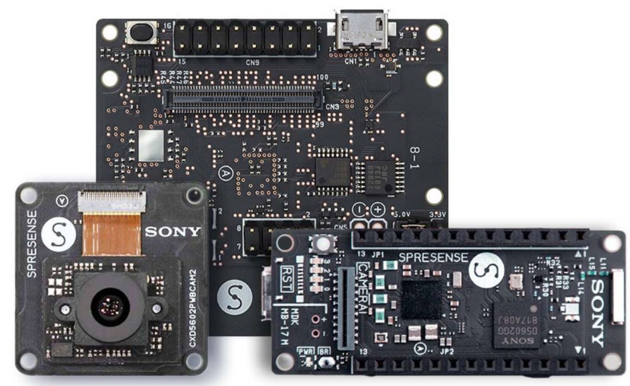
- The goal is to allow the ML/AI models to fully control and monitor the spacecraft
- ML/AI models are not perfect, so allowing full control is an issue
- SCALES will put the models in a "box" to enforce hard bounds to keep the system from completely failing
- Providing these limitations will allow the models to make mistakes and re-train during flight without jeopardizing the health and performance of the system
  - The bridge to F Prime will enforce the "box"

Hardware Considerations: Flight Computer

- Octavo Systems OSD335x-SM
  - Smallest TI AM335x module
  - TI Sitara ARM Cortex-A8 AM335x processor
  - 4KB of EEPROM
  - Used in OreSat C3
  - Used in PocketBeagle

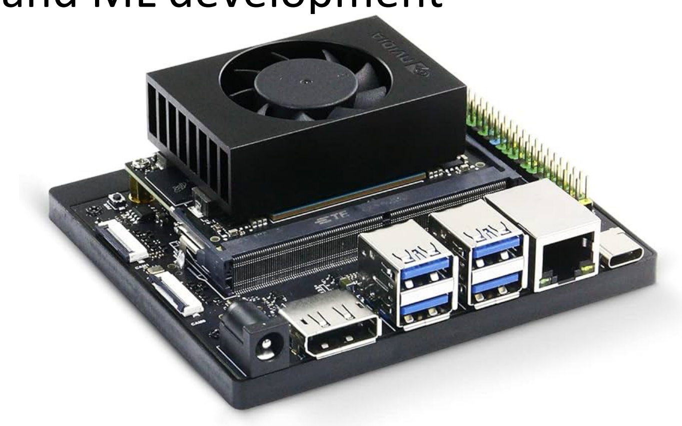


- Sony Spresense
  - ARM Cortex-M4F x 6 Cores
  - High performance
  - Low power consumption
  - 1.5MB of SRAM
  - Fully depleted Silicon chip



Hardware Considerations: Edge-Computer

- NVIDIA Jetson Orin Nano
  - Biggest name in the game
  - 40 TOPS
  - 6-core ARM Cortex-A78AE CPU
  - 8GB 128-bit LPDDR5 memory
  - NVIDIA GPU is commonly used for AI and ML development



- Google Coral Dev Board
  - Used by Stanford University
  - Edge TPU coprocessor
    - 4 trillion operations per second (TOPS)
    - 2 TOPS per Watt
  - Quad-core ARM Cortex-A35 CPU
  - 8GB eMMC

Interfacing and Testing

- How are we going to space-proof our hardware?
- Plan to do a lot of radiation and thermal testing
  - We will probably put a metal box around the components – both for radiation and thermal management
    - Aluminum and stainless steel are common contenders
    - Incorporating tantalum (or other similar metals) into design to protect more vital parts of components
  - Monitoring flipped bits
  - Figure out the life expectancy and how/when they break and under what conditions

- How do we get the two main boards to talk to each other?
  - Running same/similar operating system will make this easier
- How to implement ports between boards to interface other sensors
  - We may design a custom PCB interface board to manage inflow and data streams
    - Build off the concepts used in the open source OreSat C3 board

Project Timeline

- 2 – year project:
  - 1 year of design and integration
  - 1 year of testing and qualification

#	Task Name	Duration	Start	ETA
Complete project execution		730 Days	03-01-24	TBD
1	Preliminary Design	90 Days	03-01-24	05-31-24
1.1	Student Hiring	21 Days	03-01-24	03-21-24
1.2	Requirement Definition	60 Days	03-01-24	04-30-24
1.3	F' Workshop	1 Day	TBD March	TBD March
1.4	Software Architecture	30 Days	04-30-24	05-30-24
1.5	Hardware System Design	30 Days	04-30-24	05-30-24
1.6	Preliminary Design Review	1 Day	05-31-24	05-31-24

2	Critical Design	90 Days	06-01-24	09-01-24
2.1	Key Brassboard Component Procurement	30 Days	06-01-24	07-01-24
2.2	Initial F' Software Topology	15 Days	06-01-24	06-15-24
2.3	Brassboard Component Testing	45 Days	06-15-24	07-30-24
2.4	Initial F' Deployment	30 Days	06-15-24	07-15-24
2.5	Brassboard Prototype Integration	30 Days	07-01-24	07-31-24
2.6	SmallSat Conference	7 Days	08-02-24	08-09-24
2.7	Critical Design Review	2 Days	08-15-24	08-16-24
3	Prototype System Integration	104 Days	09-01-24	12-14-24
3.1	SCALES System Prototype Manufacturing	30 Days	09-01-24	10-01-24
3.2	Prototype F' Software Topology	30 Days	09-01-24	10-01-24
3.3	Hardware Stack Integration	15 Days	09-21-24	10-07-24
3.4	Prototype F' Software Deployment	45 Days	10-01-24	11-15-24
3.5	Hardware Stack Benchmark Testing	30 Days	10-07-24	11-07-24
3.6	SCALES System Prototype Integration	30 Days	11-07-24	12-07-24
3.7	SCALES System Prototype Build Qualification	7 Days	12-07-24	12-14-24
3.8	Winter Margin & Intermission	14 Days	12-21-24	01-07-25

Project Timeline (Cont.)

4	Autonomy Integrations & Qualification Testing	180 Days	01-07-25	06-07-25
4.1	Deployment of Benchmark AI/ML Models	TBD	TBD	TBD
4.2	F' Deployment Bug Fixes and Optimization	TBD	TBD	TBD
4.3	Environmental Testing	TBD	TBD	TBD
5	System Refinement and Application Testing	90 Days	06-01-25	09-01-25
5.1	Hardware / Software System Fixes and Improvement	TBD	TBD	TBD
5.2	Autonomous Detection Test	TBD	TBD	TBD
5.3	Autonomous Tasking Test	TBD	TBD	TBD
5.4	Edge Case Handling Tests	TBD	TBD	TBD
5.5	SmallSat Conference	7 Days	08-01-25	08-07-25

6	Environmental Benchmarking and Application Testing	104 Days	09-01-25	12-14-25
6.1	Shock and Vibration Testing	TBD	TBD	TBD
6.2	Thermal / Vacuum Testing	TBD	TBD	TBD
6.3	System Level Radiation Testing	TBD	TBD	TBD
6.4	Application Stress Testing	TBD	TBD	TBD
7	Autonomy Integrations & Qualification Testing	180 Days	01-07-26	03-01-26
7.1	Flight Test Preperations	TBD	TBD	TBD
7.2	Final Report Drafting	TBD	TBD	TBD
7.3	Results Dissemination and Open Sourcing	TBD	TBD	TBD

Thank you for listening!

Any questions?

kwilliams1@cpp.edu

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration. The material is based upon work supported by NASA under award No 80NSSC24M0003.

References

andrewgreenberg. (n.d.). oresat-c3-hardware. Retrieved from GitHub: https://github.com/oresat/oresat-c3-hardware

BeagleBoard.org Foundation. (n.d.). What is PocketBeagle? Retrieved from beagleboard.org: https://www.beagleboard.org/boards/pocketbeagle-original

Google. (n.d.). Dev Board Mini. Retrieved from Coral: https://coral.ai/products/dev-board-mini/

NASA JPL. (n.d.). F' Flight Software & Embedded Systems Network. Retrieved from NASA - GitHub: https://nasa.github.io/fprime/

NVIDIA. (n.d.). Robotics and Edge Computing. Retrieved from NVIDIA.com: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/

Octavo Systems. (n.d.). OSD335x-SM System-in-Package. Retrieved from Octavo Systems: https://octavosystems.com/octavo_products/osd335x-sm/

PyCubed. (n.d.). Retrieved from PyCubed: https://pycubed.org/

Sony. (n.d.). Spresense. Retrieved from developer.sony.com: https://developer.sony.com/spresense#secondary-menu-desktop