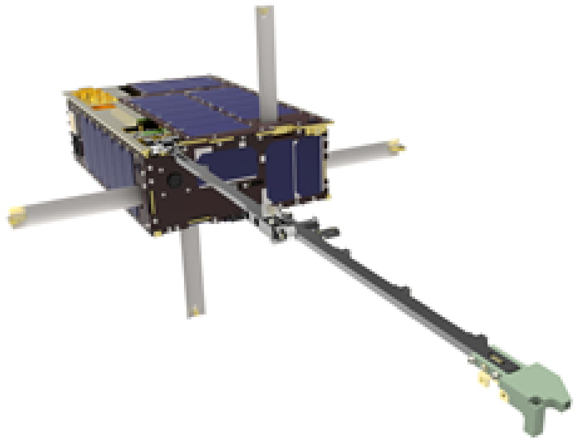
Porting the core Flight System to the Dellingr Cubesat

Alan Cudmore
NASA Goddard Space Flight Center

Agenda

- Background
- Overview of the Dellingr Cubesat
  - Hardware Overview
  - Flight Software Overview
- Simulation on a Budget
- Flight Software Challenges
- Port Details
- Lessons Learned
- Conclusion
- Mission Status



Background

- Dellingr is a Heliophysics 6U Cubesat developed at GSFC
  - "Dellingr" is named after the mythological Norse god of the dawn
- Originally planned as a one year ”skunkworks” effort
  - 1 person/year (1 FTE) for all flight software (FSW) including the Attitude Control System (ACS)
  - Selected Gomsapce Nanomind A712 platform for a 32 bit CPU with minimal power consumption
- After preliminary design review, I volunteered to port the cFS (OSAL/PSP) to the platform
  - After cFS port to a development board, project decided to use the cFS
- Revised plan
  - Original developer continued to work on the ACS
  - I worked on the cFS implementation and created mission app templates
  - Others worked on integrating device code, developing mission specific apps, and developing ground software

Dellingr Cubesat Hardware Overview (1)

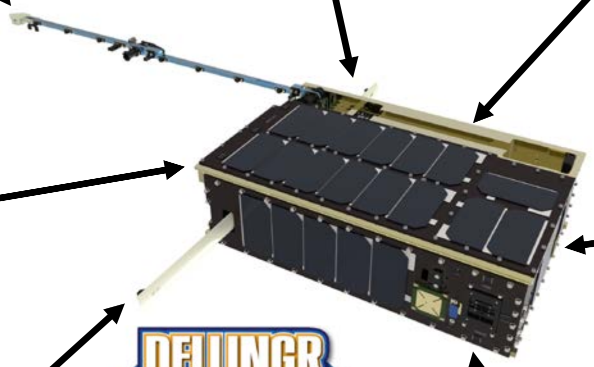
DAGR:
Science
Magnetometers
(Internal and Boom Mounted)

Power:
Clyde Space EPS and Batteries

CPU:
Gomspace Nanomind A712d ARM processor

INMS:
Ion Neutral Mass Spectrometer

Experiments:
Thermal Louvre
Deployment Camera



Comm:
L3 CADET Radio
3 Mbits down/9600 up
4 Gbytes packet Storage
(antenna by home depot)

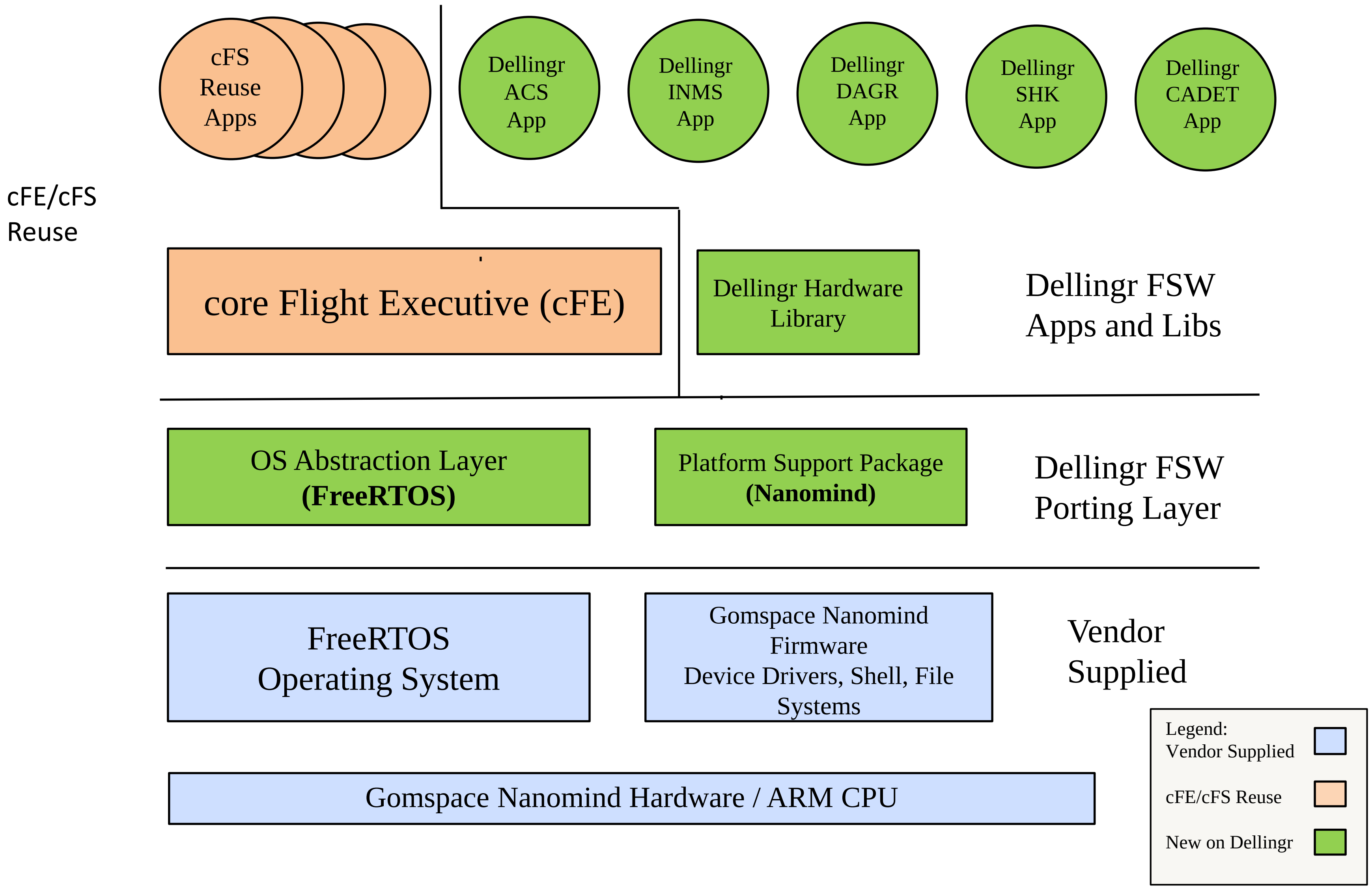
ACS:
Gyro, Fine Sun Sensors, Coarse Sun Sensors, GPS, Magnetometers, Reaction Wheels

Dellingr Facts:
- 6U Cubesat
- Delivered to ISS August 2017
- Deployed from Nanoracks Deployer On November 20, 2017

Dellingr Cubesat Hardware Overview (2)

- Processor Card - Gomspace Nanomind A712d
  - Atmel ARM7 CPU @ 40Mhz
  - 2 Megabytes SRAM
  - 2 x 4MB Flash Memory
  - Micro SD card
  - I2C, SPI, UARTs
  - Real Time Clock
  - Magnetometer, PWM, Analog Inputs

Dellingr Cubesat Flight Software Overview (1)

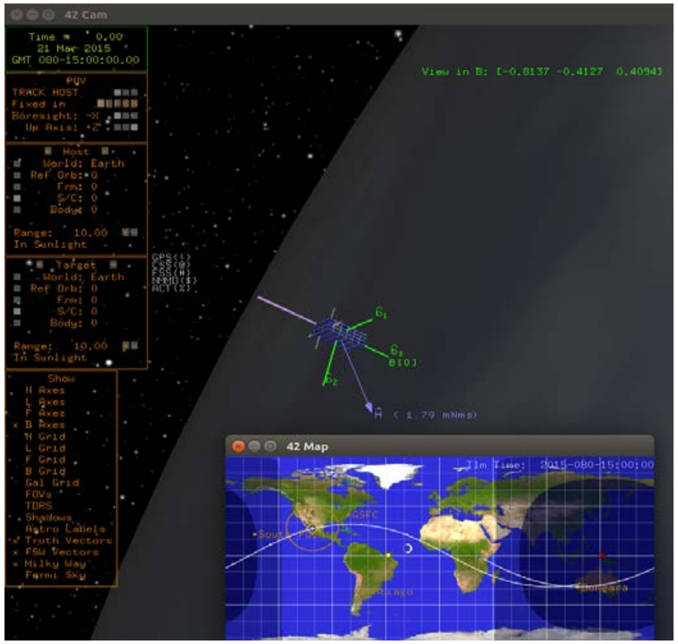

cFS Reuse Apps
Dellingr ACS App
Dellingr INMS App
Dellingr DAGR App
Dellingr SHK App
Dellingr CADET App
cFE/cFS Reuse
core Flight Executive (cFE)
Dellingr Hardware Library
Dellingr FSW Apps and Libs
OS Abstraction Layer (FreeRTOS)
Platform Support Package (Nanomind)
Dellingr FSW Porting Layer
FreeRTOS Operating System
Gomspace Nanomind Firmware Device Drivers, Shell, File Systems
Vendor Supplied
Gomspace Nanomind Hardware / ARM CPU
Legend:
Vendor Supplied 🟦
cFE/cFS Reuse 🟧
New on Dellingr 🟩

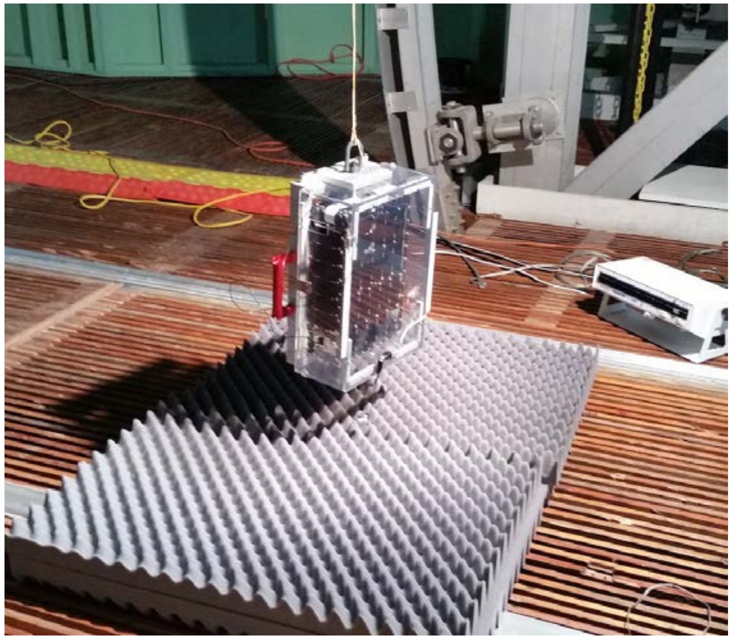
Dellingr Cubesat Flight Software Overview (2)

Vendor Software	**Reuse Software**	**New Software**
FreeRTOS OS	cFE Core	FreeRTOS OSAL Layer
Bootstrap/Startup Code	cFS Scheduler	cFE Platform Support Package
Device Drivers	cFS Stored Commands	Dellingr Hardware Library
File Systems	cFS Health and Safety	DAGR Instrument App
C Library	cFS Checksum	INMS Instrument App
	cFS Limit Checker	CADET Radio App
	cFS File Manager	ACS App
	cFS Memory Dwell	Ephemeris/GPS App
	cFS Memory Manager	Reaction Wheel App
	Static Loader (MMS)	File Downlink App
		File Downlink App
		Spacecraft Housekeeping App

Simulation on a Budget (1)

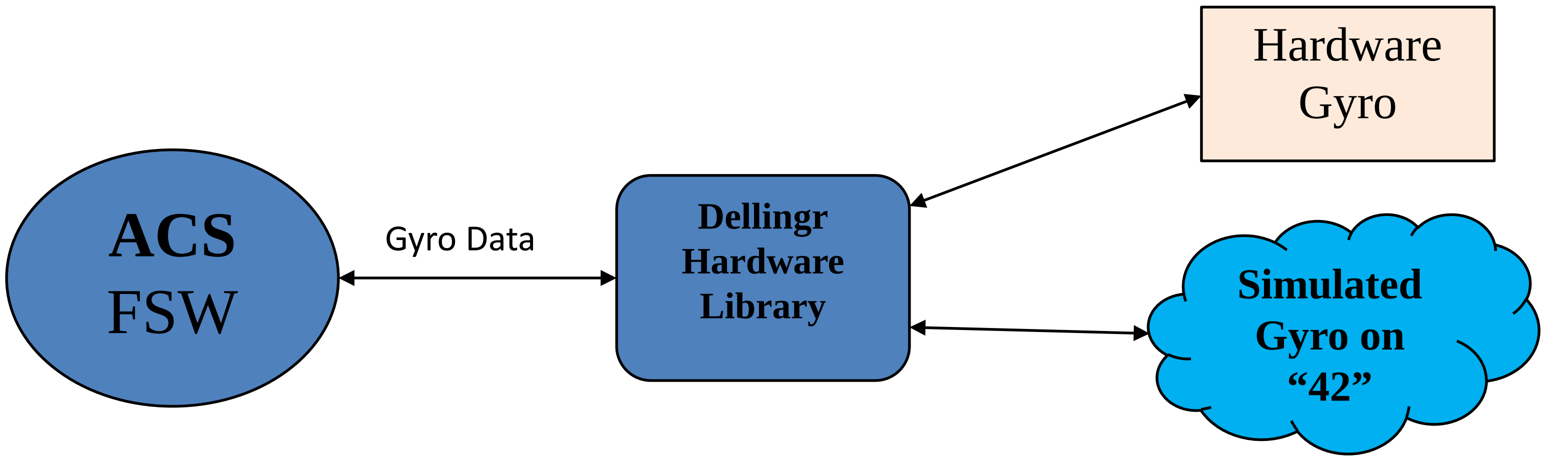
- **Dellingr did not have dedicated dynamic simulator such as the Goddard Dynamic Simulator (GDS) for ACS testing**
- **Simulation is accomplished with a combination of:**
  - **The open source dynamic simulator ”42” developed by GSFC Code 591**
  - **The “Dellingr Hardware Library” with an interface to the ”42” simulator**
- **Most of the ACS app debugging was done on cFS/Linux with 42**
- **When the Satellite was complete, we were able to do a spin test using the reaction wheels and sun sensors**





Simulation on a Budget (2)

- **Dellingr Hardware Library Details**
  - A cFS Library that abstracts all mission specific hardware
    - Focus is on the ACS Hardware
  - Allows FSW to run in the following configurations:
    - On-board computer with real hardware devices
    - On-board computer talking to 42 Simulator over a serial port
    - Linux computer talking to 42 Simulator over TCP/IP Socket
  - The ability to run the Dellingr Flight Software on Linux with the 42 speeds up ACS development and test
  - This model worked well and will be used for future efforts



Flight Software Challenges (1)

- **New Operating System: FreeRTOS**
  - Required a port of the Operating System Abstraction Layer (OSAL)
  - Once the OSAL was working, the most of the cFS just worked
- **On-board computer only has 2 Megabytes of RAM**
  - The cFS code has to run in Flash, since 2 Megabytes of RAM is not enough
  - Some cFS applications had to be left out to save room (CFDP file transfer, Housekeeping)
- **Flash file system is limited**
  - Only holds 64 files, not enough for all of the mission stored command sequences
  - Workaround is to “bundle” multiple stored command sequences into a single file
  - This is a generic solution that could be used on other missions
- **Cadet Radio Subsystem**
  - Half Duplex, not ideal for real time telemetry and commands
  - Very low data rate between on-board computer and radio/recorder limits amount of data spacecraft can produce at one time

Flight Software Challenges (2)

- **Limited Schedule and Budget**
  - The traditional FSW development lifecycle does not work here
  - Framework of the cFE/cFS provided a model to follow for new software, saving us from re-inventing the wheel
- **No ETU or Flat-Sat**
  - There is only one copy of the satellite hardware, and it was not always available for FSW development
  - FSW changes had accumulated right before Thermal Vacuum testing leading to last minute integration bugs and troubleshooting
- **Significant time spent working with device drivers and interfaces**
  - A Typical large mission has a limited set of known interfaces (1553/Spacewire, Serial)
  - Dellingr uses Serial, I2C, SPI, Analog, GPIO, and I2C extended interfaces
  - Multiple problems encountered with I2C and SPI interfaces that just needed more troubleshooting time and effort
  - Engineers end up being multi-discipline (FSW, Hardware, Mechanical, I&T)

Port Details: OS Abstraction Layer

- **Kernel**
  - Basic FreeRTOS port was fairly straightforward
    - All of the OS primitives are present
  - FreeRTOS has a configurable set of priorities (used 256)
  - Using FreeRTOS 8.x
- **File Systems**
  - FreeRTOS does not include a File System
  - Gomspace integrated firmware that included:
    - The “fatfs” open source Filesystem
    - UFFS filesystem for the flash memory
    - A POSIX API Virtual Filesystem Layer (VFS)
- **Network interfaces**
  - FreeRTOS does not include a network stack in the base kernel, but this was not needed for Dellingr
  - The Gomspace Cubesat Space Protocol (CSP) was used to transfer files to the flash over the serial port
- **Dynamic Loader**
  - Used a combination of MMS Static Loader and linking apps directly with the cFE Core and base OS image
  - Most mission apps were loaded from the file system

Port Details: Platform Support Package

- **PSP / Startup code**
  - Wanted to leverage the Gomspace Nanomind firmware
    - Modifications made to facilitate cFS port
    - The original Gomspace Shell was used for a diagnostics console
  - With only 2MB of RAM, the cFE/cFS code had to run in Flash
    - Only selected mission apps could be dynamically loaded to RAM
- **1HZ Time Source**
  - There was no 1hz time source for this system (GPS is not always powered)
  - The FreeRTOS 1000hz timer tick is used as the 1hz source
- **File Systems, Volume Table**
  - The /ram and /boot file systems are directly mapped to the OSAL
  - No path translation is needed
- **Restart Code**
  - Ties to the Gompace firmware reset function
- **Watchdog**
  - Uses the CPU watchdog feature
- **Flash/Nonvolatile access**
  - Due to time and schedule, the flash is readable via memory map for checksum, but not writeable outside of the file system

Lessons Learned (1)

- **Positive cFS Lessons:**
  - The cFS brought a development environment, FSW framework, and process to the project
    - Work could be broken up in to modules or cFS apps for developers to focus on
    - The modularity of an app helped isolate the code
  - The cFS allows you to focus on mission specific code and start to work on that immediately
  - The cFE and cFS functionality added a lot to the mission with little effort
    - FDC, stored commands, Limit Checker, Table Services, saved a lot of time in re-inventing software
  - The cFS cross platform capability allows us to develop and run on desktop Linux, Raspberry Pi, and other targets
    - Very useful for getting high level application logic working
- **Negative cFS Lessons:**
  - The cFS was a poor fit for Gomspace Nanomind processor
  - Experience was used to work around limitations of the platform – This was not a recommended port for cFS beginners
  - cFS experience helped with the selection and configuration of apps – Learning the cFS is beneficial before trying to develop a system

Lessons Learned (2)

- The 42 Simulator interface to the cFS enabled ACS development and debug
  - ACS algorithm checkout would have been very hard without it
- Not having a Flat Sat or second copy of the hardware was difficult
  - Having the cFS and simulation capability from day 1 would ease this problem, but not eliminate it
- Hardware devices were hard to work with, and hardware/software integration was time consuming
  - This was unavoidable no matter what the framework
  - I2C , SPI, UART interfaces are all “Finicky”
  - Used Standalone diagnostic code and even Arduino to test out devices
  - It was often easy to get a device working with a test program by itself but harder when in a multitasking environment with multiple devices operating

Conclusion

- Was the cFS the right choice for the Dellingr Cubesat?
  - Yes : There were compromises, but valuable lessons were learned that can immediately be applied to the next generation of Cubesats
  - We were able to leverage cFS experience and expertise to make up for shortcomings
- What would we do differently if starting over today?
  - If using the cFS, influence the hardware selection to take advantage of an existing port, or at least an easier port
  - Start defining mission specific applications and building a version of the FSW that runs on Linux or similar desktop board
  - Simulate hardware interfaces and integrate 42 simulator from early in the development
  - Consider the ground system earlier

Mission Status

- Dellingr was deployed from the International Space Station (ISS) on November 20, 2017