CubeSat
Flight Control Software
Senior Project - 2013
Dan Turner
Colin Myers
Project Description

- Flight Control Software for Vermont Techs CubeSat
- Scheduled to launch in September of 2013
- **Purpose:** Test onboard subsystems & a SPARK implementation of GEONS
- **Future Goal:** Send a CubeSat to the moon to either orbit or land on it.
- The completed CubeSat design will be used to replicate additional CubeSats for future missions.
Student Involvement

2011 (Summer) – Jeremy Audet & Matt Ward started the flight control software
2012 (Summer) – Dan Turner picked up where Matt and Jeremy left off with the flight control software
2012 (Fall) – Michael Collins and Colin Myers started working on the flight control software
2013 (Spring) – India Beauregard helped develop the IMU board
Other Acknowledgements

Carl Brandon - Principal Investigator

Peter Chapin - Software Director

Carl Wolf - Helped with designing the IMU board

Oliver Piluski (LED Dynamics) - Helped with the development and printing of the IMU board

Barry Trutor (Microstrain) - Helped with IMU integration

Rob Devarney - Is currently assisting with the Ground Control
What is a CubeSat?

"CubeSats are a class of research spacecraft called nanosatellites. The cube-shaped satellites are approximately four inches long, have a volume of about one quart and weigh about 3 pounds."

--NASA
Hardware Requirements

Hardware consist of:
• Camera
• Inertial Measurement Unit (IMU)
• GPS
• Power Supply
• Radio
• MSP430 processor
• SD Card
• Antenna
Software Requirements

- The **Antenna** must be deployed no sooner than 45 minutes after the CubeSat is ejected from the rocket.
- The **Radio** must send a beacon every minute if there is no communication with the Ground Control.
- The **File System** must store telemetry data, error logs, and images on a SD card.
- The **Camera**, **GPS** and **IMU** must save telemetry data to the file system.
- The **Power Supply** must report battery status when requested.
- All Flight Control Software must be written in **SPARK/Ada**.
What is SPARK/Ada?

“SPARK is a formally-defined computer programming language based on Ada, intended to secure and support the development of high integrity software used in applications where predictable and highly reliable operation is essential”

-- Wikipedia (SPARK programming language)
Why Use SPARK?

• If the software fails, we will lose the satellite

• Ada offers a greatly improved probability of error-free software when compared with C. Most other CubeSat projects use the C Programming Language.

• SPARK uses static analysis to prove our code free of run-time errors.
package body Time

-- own State is Timer, Time_Since_Initialization, Last_Lookup_Time;

is

Timer : Timer_Type := Super_Loop;

Time_Since_Initialization : Time_Type := 0;

Last_Lookup_Time : Time_Type := 0;

procedure Restart_Timer

-- global out Timer, Time_Since_Initialization, Last_Lookup_Time;
-- derives Timer from &
-- Time_Since_Initialization from &
-- Last_Lookup_Time from ;

is

begin
    Timer := Super_Loop;
    Time_Since_Initialization := 0;
    Last_Lookup_Time := 0;
end Restart_Timer;

procedure Sleep(Millisecond_Count : in Utility.Millisecond_Count_Type)

-- global in out Time_Since_Initialization, Utility.Hardware;
-- derives Time_Since_Initialization from Millisecond_Count, Time_Since_Initialization &
-- Utility.Hardware from Millisecond_Count, Utility.Hardware;

is

begin
    Time_Since_Initialization := Time_Since_Initialization + Time_Type(Millisecond_Count);
    Utility.Sleep(Millisecond_Count);
end Sleep;

procedure Get_Time_Milliseconds( Time_Milliseconds : out Time_Type)
Software Development Process

- There is no Ada compiler for our Texas Instruments MSP430 series processor

- In order to resolve this dilemma we must convert our Ada code to ANSI C with AdaMagic.
Software Development Process

- Software is written in SPARK/Ada using AdaCore’s GNAT Programming Studio (GPS)

- Checked with the Praxis High Integrity Systems’ SPARK Toolset

- Compiled and checked with AdaCore’s GNAT Pro compiler

- Sofcheck’s AdaMagic compiler front end is used to produce ANSI C as the intermediate code

- Rowley’s Crossworks C cross compiler for Texas Instruments’ MSP430 CPU produces the object code
Software Development Process

1. Code in SPARK/Ada with GPS
   - Spark Errors
     - Examine with Spark
     - Other Errors
       - Compile in GPS
2. MSP430 object code
   - Compile C in Crossworks
     - C intermediate code
       - Compile in AdaMagic
System Design
System Design

Antenna → Main
Camera → Main
Inertial Measurement Unit → Main
File System → Main
Power Supply → Main
GPS → Main
Radio → Main
Commander → Main
System Design
Subsystems
Superloop/Subsystem Interaction
Superloop/Subsystem Interaction

Diagram:

- Superloop
- Sub Systems
- Commander
- File System
- Radio
Superloop/Subsystem Interaction
Superloop/Subsystem Interaction
Superloop/Subsystem Interaction
Superloop/Subsystem Interaction
Superloop/Subsystem Interaction
Superloop/Subsystem Interaction
Custom IMU Board

Components:
- IMU
- Camera (with camera shutoff circuitry)
- GPS Antenna
- Hysteresis Rods
- Antenna I\(^2\)C lines
- 6 headers of LEDs
Software Completion

Diagram:
- Antenna
- Camera
- Inertial Measurement Unit
- File System
- Main
- Radio
- Power Supply
- GPS
- Commander
Software Completion

[Diagram showing interactions between components such as Antenna, Camera, Inertial Measurement Unit, File System, Main, Radio, Power Supply, GPS, and Commander]
Spark Examinable Software
Current Status

• The CubeSat hardware is complete and has passed the ShakeN Bake

• Other than the GPS and some miscellaneous protocol procedures, the flight control system is done, although it could use more Spark testing.

• We're currently working on setting up a ground control station for our launch at the end of September.
Questions?

Dan Turner
dturner34@gmail.com
Colin Myers cmyers@vtc.edu