The Internet of Satellites: A C&DH Software Framework Based on Open Internet Standards and Software

Jesse Coffey
Smallsat 2016
Logan, Utah
Capable Cubesats

- **Powerful Processors with Small Form Factors**
  - Beaglebone Black, Raspberry Pi, etc.

- **Ethernet Connectivity**
  - Internal IP network

- **Open Source tools**
  - Massive user base
  - Exponentially increases speed of development

- **You can now “Fly your Laptop”**
“Under the Hood” - System Architecture
System Architecture

- **Linux Base**
  - Custom linux build tailored to the Beaglebone
  - Familiar platform for developers
  - Massive user base to provide open source tools and support
System Architecture

- **Core Bus Firmware**
  - Hardware Interface
    - “bent-pipe”
  - CCSDS Spacepacket standard
    - Ubiquitous in current flight software and groundstations
  - Lowest level interface
  - High TRL base
System Architecture

● **Python Library**
  ○ Open Source - On our GitHub
  ○ Library for interfacing with the Core Firmware
    ■ CCSDS Spacepackets
  ○ Can be used directly to build Python payload applications
    ■ event-driven applications
    ■ service based
System Architecture

- **HTTP server**
  - High Level
  - Internet standards:
    - ReSTful API
    - JSON data format
  - Exposes most common Core functionality

- **Advantages to Payload Developer**
  - Familiar protocol
  - Wide native or library HTTP support:
    - Java, Python, JavaScript, C/C++, C#, ...
  - API retains backward-compatibility through hardware upgrades
  - Data interchange is standard & intuitive (JSON)
System Architecture

- **Multiple Levels of Abstraction**
  - Low level control and direct access to the Core
  - High level interface for ease of access and development
Example Payload: Antarctic Observation

- **Mission:**
  - Support antarctic base research with aerial footage at high temporal resolution.

- **Tasks:**
  - Take pictures of area around arctic base
  - Process pictures during orbit to measure ice
  - Downlink data & satellite health when passing over ground stations in lower latitudes
Example Payload: Design

- **Hardware:**
  - Camera
  - Linux Computer for controlling camera and image processing
    - Raspberry Pi, Beaglebone Black, NVidia Tegra
  - SSD Drive for photo storage

- **Software:**
  - Pick your favorite language…. Let’s say C++
  - Open HTTP Library for communication with the Flight Computer:
    - libcurl - [https://curl.haxx.se/libcurl/](https://curl.haxx.se/libcurl/)
  - Open Source JSON library:
    - JSON for modern C++ - [https://github.com/nlohmann/json](https://github.com/nlohmann/json)
  - Open Source library for image processing:
    - OpenCV - [http://opencv.org/](http://opencv.org/)
Example Payload: Payload <-- Bus

- Check for pass over antarctic to take photos:
  - `GET /gps/state`
    - Response: `{lat: 84, long: -44, altitude: 235}`

- Request to point camera at targets:
  - `POST /adcs/attitude`

- Send data to flight computer for downlink on next pass:
  - `PUT /payload-01/telemetry-downlink`
    - Body: `{measurement_1: 344, measurement_2: 874, ...}`

- Check for commands from ground to downlink raw data:
  - `GET /payload-01/commands`

- Send raw files to be downlinked:
  - `PUT /payload-01/data-downlink`
Data and Commands are human readable
Pumpkin Mission Architecture

- User Programs
  - HTTP
  - Ground HTTP Server
    - UDP/IP
    - Ground Station
      - RF
      - Flight Radio
    - Flight HTTP Server
      - UDP/IP
      - Payload
        - HTTP
Example Payload: Ground Station <---> Ground Applications

- **Same Interface on Ground**
  - Payload data pulled into cloud database for analysis
  - Command/control remote from ground stations
  - Phone alerts about satellite health, pass information, etc. sent to dish operators & mission engineers.

- **Design an App for your satellite**
Development Status

- **Beta Release Ready:**
  - Core Bus Firmware
  - Python Libraries
  - HTTP/ReST Server Bus - Payload

- **In early development:**
  - Ground station HTTP server
Conclusion

- Powerful flight computers for Cubesats are here
- Adopt web standards for easy integration
- Modular and Open Source Systems speed up development
Thanks...

- Shaun Houlihan
- Austin Small
- James Womack
- Devlyn Nelson
Questions?

Diagram:
- Ethernet Switch
  - Payload
  - Payload
  - SW Payload
    - Flight Computer
  - Serial Payload
  - Radio
- User Software
- ReST API
- HTTP Server
- Python Library
- Core Firmware
- Linux

Graphical elements: arrows indicating connections between components.
Goals

● **Reliable**
  ○ Leverage as much flight heritage as possible
  ○ Compartmentalize mission specific code

● **Capable**
  ○ Fully utilize all onboard hardware
  ○ Allow access to all levels of the system

● **Compatible**
  ○ Fits into current systems seamlessly
  ○ Use widely adopted hardware/software standards

● **User friendly**
  ○ Provide multiple levels of abstraction
  ○ Enable everyone from novice to veteran developers
  ○ Open source
Typical Mission Architecture

User Programs → Manual Handoff → Command and Control Software → Ground Station

Payload → Serial, I2C, etc... → Flight Software

Flight Software → Serial, I2C, etc... → Flight Radio

RF
Pumpkin Mission Architecture