Simple Software for Prototype Hardware: Using LabView to Enable Algorithm Development for a Prototype GLXP Lunar Hopping Vehicle

Flight Software Workshop 2010

December 9, 2010

Christopher Han
Phillip Cunio
Massachusetts Institute of Technology
Overview

- GLXP and TALARIS Background
- Current and Past Work
- Prototype Software Development
- Modularity
- Software and Avionics Architecture
- GNC Execution
- Summary
Motivation and Background

• Google Lunar X-Prize (GLXP)
  – $30 million competition
  – Go to the Moon, traverse 500 m, send back images and video
  – Next Giant Leaps team

• Terrestrial Artificial Lunar And Reduced gravity Simulator (TALARIS)
  – Collaboration between MIT/Draper Laboratory for the GLXP vehicle
  – Terrestrial 1g testbed for developing guidance, navigation, and control algorithms for remote exploration of lunar and planetary surfaces
Design Overview

• Prototype hopper
  – Create simulated lunar environment
  – Demonstrate stable horizontal hop using impulsive thrusters

• Dual propulsion system
  – 4 x Electric Ducted Fans (EDFs) for gravity offset
  – 8 x Nitrogen Cold Gas thrusters

• National Instruments sbRIO-9642 computer
  • 400MHz PowerPC Processor
  • Xilinx Spartan-3 field programmable gate array (FPGA)
  • 110 3.3V DIO lines
Current Work

[Images of different equipment and experiments]
Prototype Software Development

• **Prototype Software**
  – Test in a safe environment before full flight
  – Useful in academic atmosphere: gain experience
  – Prevent delays in testing; develop flight software in parallel

• **Requirements**
  – Needs to be easy to learn - high student turnover rates
  – Short development cycle
  – Flexible – needs to be used for several different test campaigns

• **LabView graphical programming language**
  – Intuitive interface
  – Makes multi-threaded programming easy
  – Makes serial programming more difficult
  – Easy GUI development
LabVIEW Mapping to Modularity (1)

Functional Module Structure

- Caller routine
  - Module 1a
  - Module 1b
  - Module 2

Physical Module Structure (C/MATLAB/etc.)

```plaintext
%Caller routine
  if X
    ...
  %Module 1a
    for k = 0,n
      ...
      sqrtk = sqrt(k)
      n = n + 1;
  %Module 1b
    ...
```
LabVIEW Mapping to Modularity (2)

Functional Module Structure
- Caller routine
  - Module 1a
  - Module 1b
  - Module 2

Physical Module Structure (LabVIEW)
- IMU Data.Read
  - Number of Elements
  - Timeout (ms)
- Altimeter Data.Read
  - Number of Elements
  - Timeout (ms)
  - Data
  - Elements Remaining
- Elements Remaining
- Data
Modularity

• Labview is inherently suited to modular development
  – Separate program into functional blocks, or modules

• Defining Characteristics
  – Use of exchangeable components that can be easily switched out
  – Each module has little to no dependence on other parts of the system
  – Ability to incorporate a variety of configurations

• Formal Techniques
  – Separate and define functional attributes
    • i.e. sensing, data collection and logging, actuation, timing
  – Define standard interfaces
  – Determine series/parallel execution of modules
  – Map coupling and interaction between components
Modularity on TALARIS

- **Hierarchical structure**
  - FPGA -> PPC -> Flight Laptop
  - Implementation, Execution, and Timing of GNC algorithms

- **Benefits**
  - Increased organization
  - Easier to test and debug
  - Easier to work on with large teams
  - Ability to switch out GNC algorithms
  - Improve flexibility, maintainability, and reusability

- **Detriments**
  - Performance hit?
  - Initial development time?
Modularity on TALARIS

CGS controls and displays

Voltage and pressure sensors

EDF controls and displays
Modularity on TALARIS

Timing & Sensors

CGS control

EDF control
Software & Avionics Architecture

- NI sbRIO-9642 computer
  - FPGA
    - All hardware interfaces through DIO pins
    - Timing on single 40MHz clock
    - Package data with DMA FIFO
  - PowerPC
    - Modular GNC implementation
    - Autocoding GNC from Matlab and/or C
    - Flight Controls
  - Flight Laptop
    - Starts execution
    - Logging

Other sensors
- Altimeter
- IMU
- CGS
- EDFs
- RS-485
- RS-232
- Wireless Communication

FPGA
- Hardware interfaces
- Timing
- Flight controls
- GNC execution
- Communication

PPC
- Start Execution
- Logging
GNC Execution Example

- 5 Hz (200ms) control frequency

Gather data from sensors → Navigation: Determine position & attitude → State estimate → Guidance: Determine target → Control: Actuator Commands

Navigation:
- Determine position & attitude

Guidance:
- Determine target

Control:
- Actuator Commands

State estimate:
- Target Location

GNC Execution Flowchart:
- Gather data from sensors
- Navigation: Determine position & attitude
- Guidance: Determine target
- Control: Actuator Commands

Data Flow:
- Sensor data collection
- Data processing
- State estimation
- Target determination
- Actuator commands
GNC Timeline

- 5Hz control frequency = 200ms "frames"
- Gather 200ms of IMU data, send up in packets
- Process data in chunks, send out commands to be executed on next boundary
- Control cycle delay?
• Splitting of processes between modules enables fine-tuning of timing
  – One piece of information propagates between three modules
  – Each module has a different run time
Summary

• For prototype hardware with an aggressive testing schedule, need prototype software that can keep up
  – No delays in testing
  – Gain experience for flight software

• Labview has enabled TALARIS to test with multiple configurations
  – Manage a flexible, reusable software architecture
  – Maximize the testing opportunities while minimizing development time

• Modularity can help in the development, operation, and maintenance phases
Questions?

TALARIS would like to thank its sponsors: