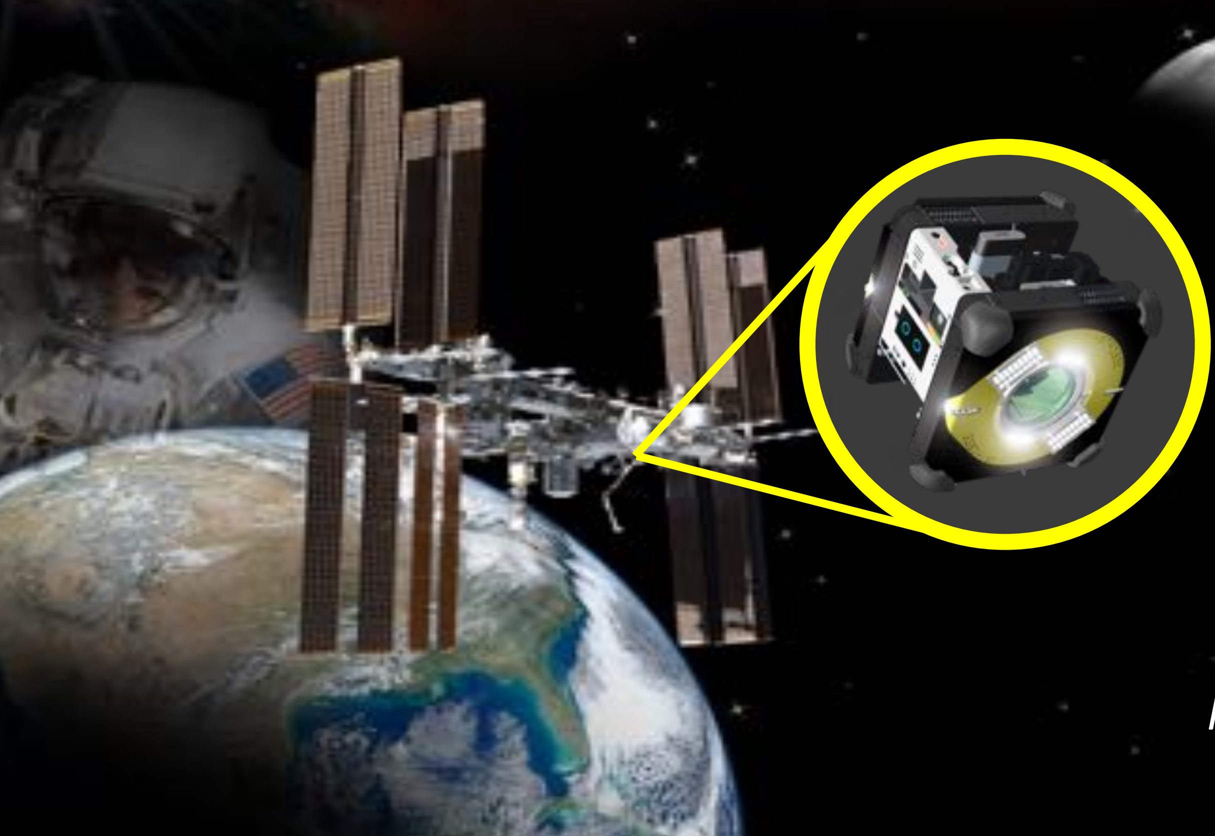
Astrobee Robot Software: Enabling Mobile Autonomy on the ISS



Lorenzo Flückiger

Brian Coltin

Intelligent Robotics Group
NASA Ames Research Center


ISS ROBOTIC FREE FLYER
ASTROBEE


AERCam Sprint
Smart SPHERES
PSA

MIT
NASA ARC
AURORA DARPA
SPHERES


To HAB


ASTROBEE

Astrobee Project Elements

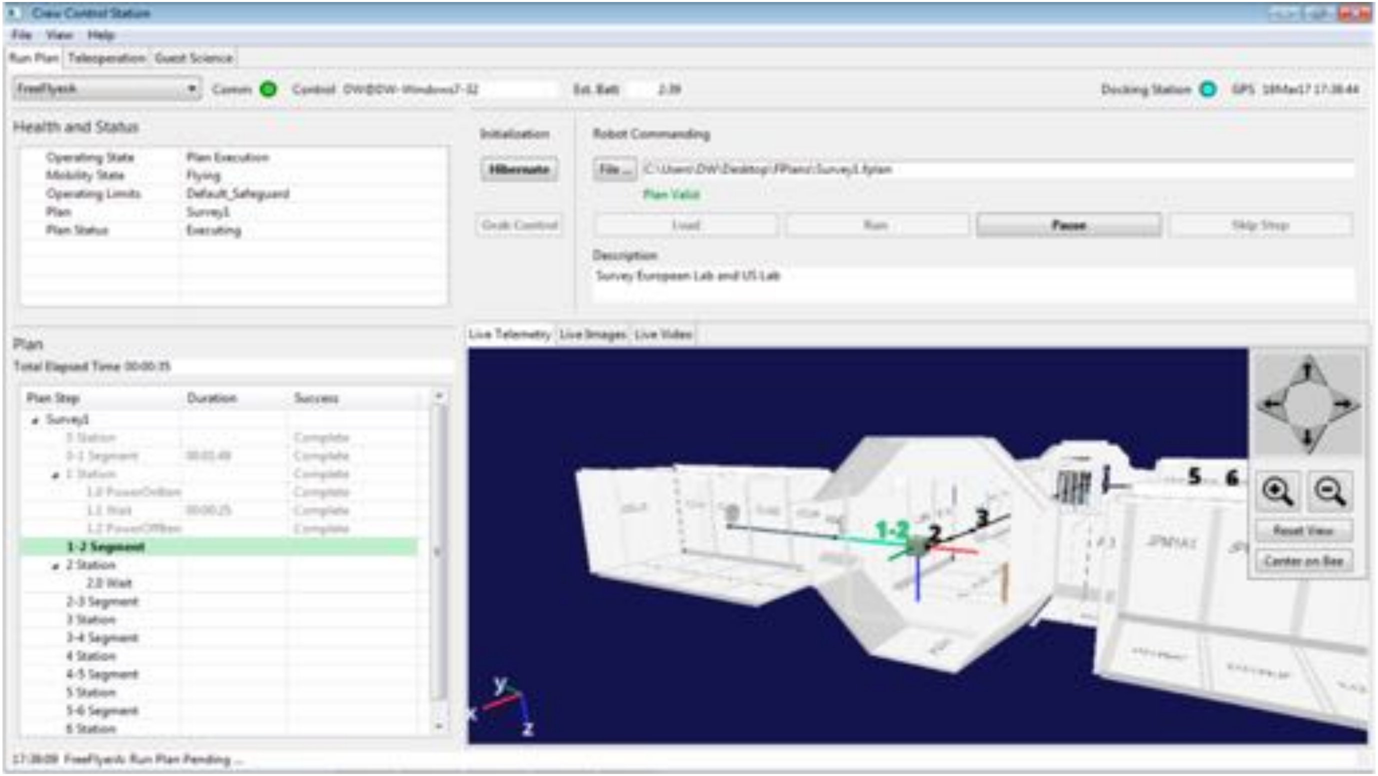
- Astrobee
  - Free Flyer
  - Dock
  - Ground Data System

Engineering Tools

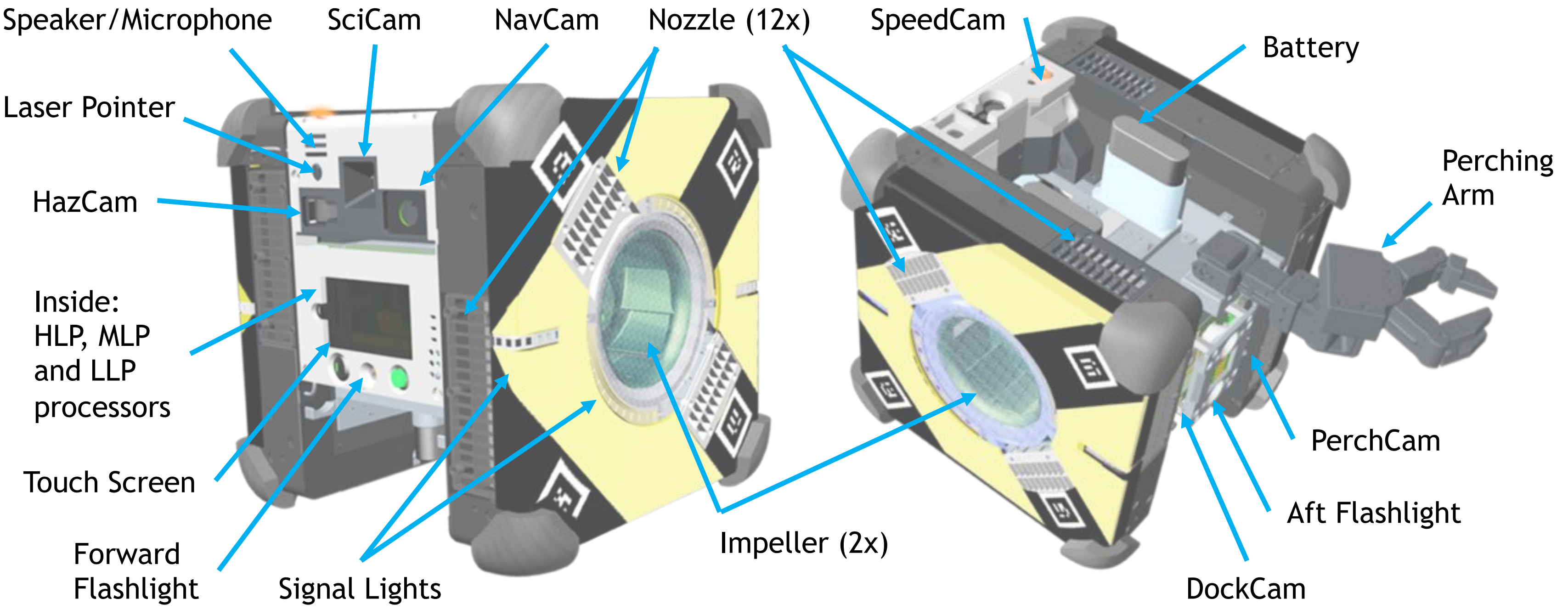
Users Development Testbed (Simulator)

Astrobee Robot Software

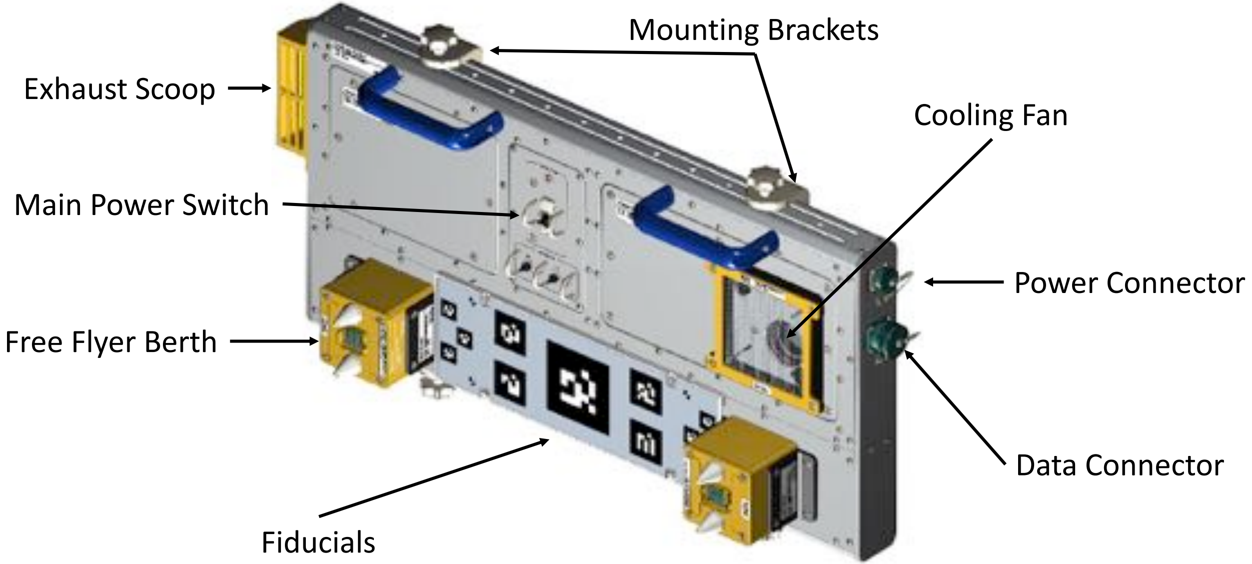
Astrobee Control Station



Astrobee Free-Flyer Hardware



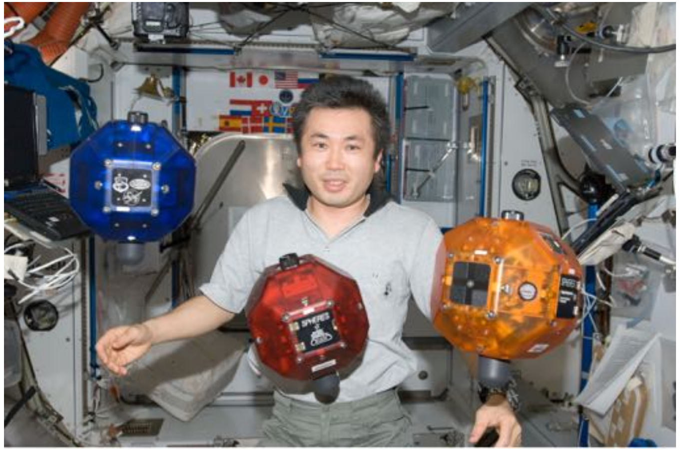
Docking Station



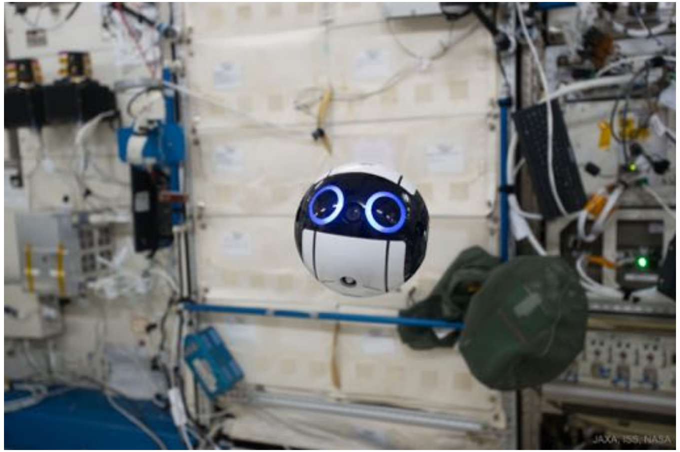
- Berths for 2 free flyers
- Provides power and Ethernet
- Provide wired update path
- Fiducials used for visual servoing to autonomously dock
- Magnets provide retention force

Current IVA Free Flyers

- SPHERES (NASA) – launched 2006
  - Highly successful research platform used for many guest science experiments
  - Astrobee will replace SPHERES, managed by the same facility team
- Int-Ball (JAXA) – launched 2017
  - Successful experiment in building an IVA free flyer with a rapid development cycle (18 months)
  - Small size (15 cm diameter) enabled by JAXA's miniaturized all-in-one CPU / IMU / 3-axis reaction wheel module
  - Joint activities between Int-Ball and Astrobee may be possible
- CIMON (DLR) – Demo in November 2018
  - Enable research on AI for human-robot interaction
  - International cooperation – CIMON will share from the pool of batteries that Astrobee qualified for ISS



SPHERES



Int-Ball

The Astrobee Platform

- Research Platform: Enabling research partners to conduct scientific experiments in micro-gravity by developing their own software and/or hardware payloads.
- Autonomous Surveyor: Carrying payloads to perform spatial surveys of the environment. For example, an assessment of air quality or noise levels.
- Mobile Camera: Permitting ground controllers to monitor crew operation with a high quality video stream.

Software Features

- Localize throughout the U.S. Orbital Segment of the ISS without extra infrastructure.
- Precisely plan and execute motions without collision.
- Provide control and monitoring from the ground with resilience to communication loss.
- Support multiple control modes, including remote teleoperation, autonomous plan execution and on- board control by guest science (external researchers) software.
- Autonomously dock for battery recharging and wired communications.
- Autonomously perch on handrails to conserve energy while providing pan/tilt camera functionality.
- Manage guest science software, hardware payloads, and user interface components.

Software Challenges

- The unconventional location of and limited space within the ISS preclude localization techniques that exploit beacons, satellite navigation systems or Earth's gravity and magnetic fields.
- The ISS provides high bandwidth communications, but has frequent signal loss and high latency.
- The software must maximize reliability to minimize crew interventions.
- Astrobee robots are not serviceable by an expert on orbit, and thus the system needs to support completely remote maintenance and introspection.

Vision based localization with offline map generation

Powerful, space tested communication framework (DDS)

Proven solutions + Testing

Remote firmware and software update capabilities

Astrobee Robot Software classification

Class C,
non-safety critical

- Subject to NPR **7150.2B** (NASA Software Engineering Requirements)
- Software Development Plan and Processes required

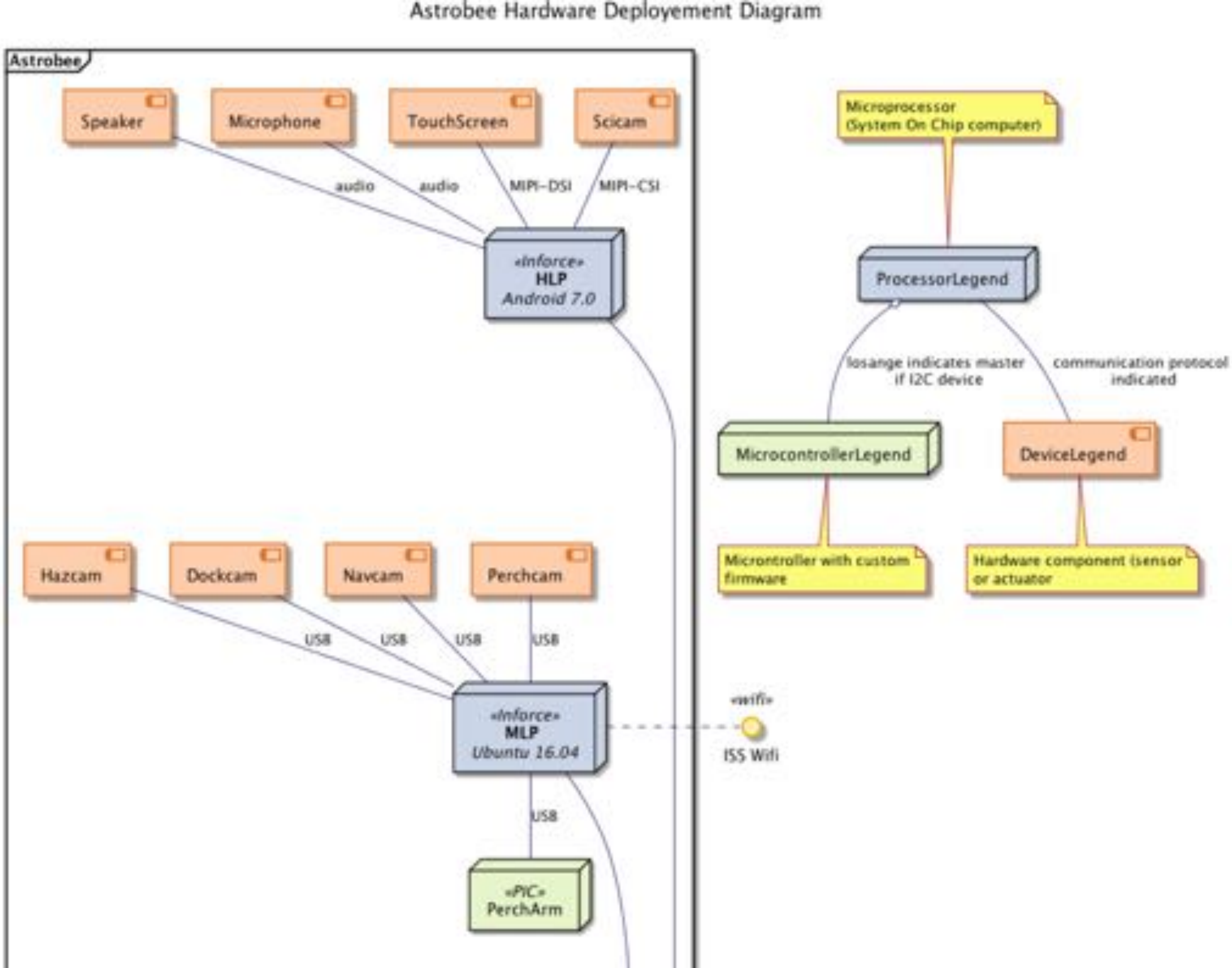
Class	Description
A	Human Rated Space Software Systems
B	Non-Human Space Rated Software Systems or Large Scale Aeronautics Vehicles
C	Mission Support Software or Aeronautic Vehicles, or Major Engineering/Research Facility Software
D	Basic Science/Engineering Design and Research and Technology Software
E	Design Concept and Research and Technology Software
F, G, H	General Purpose Computing / Desktop …

Hardware computation architecture

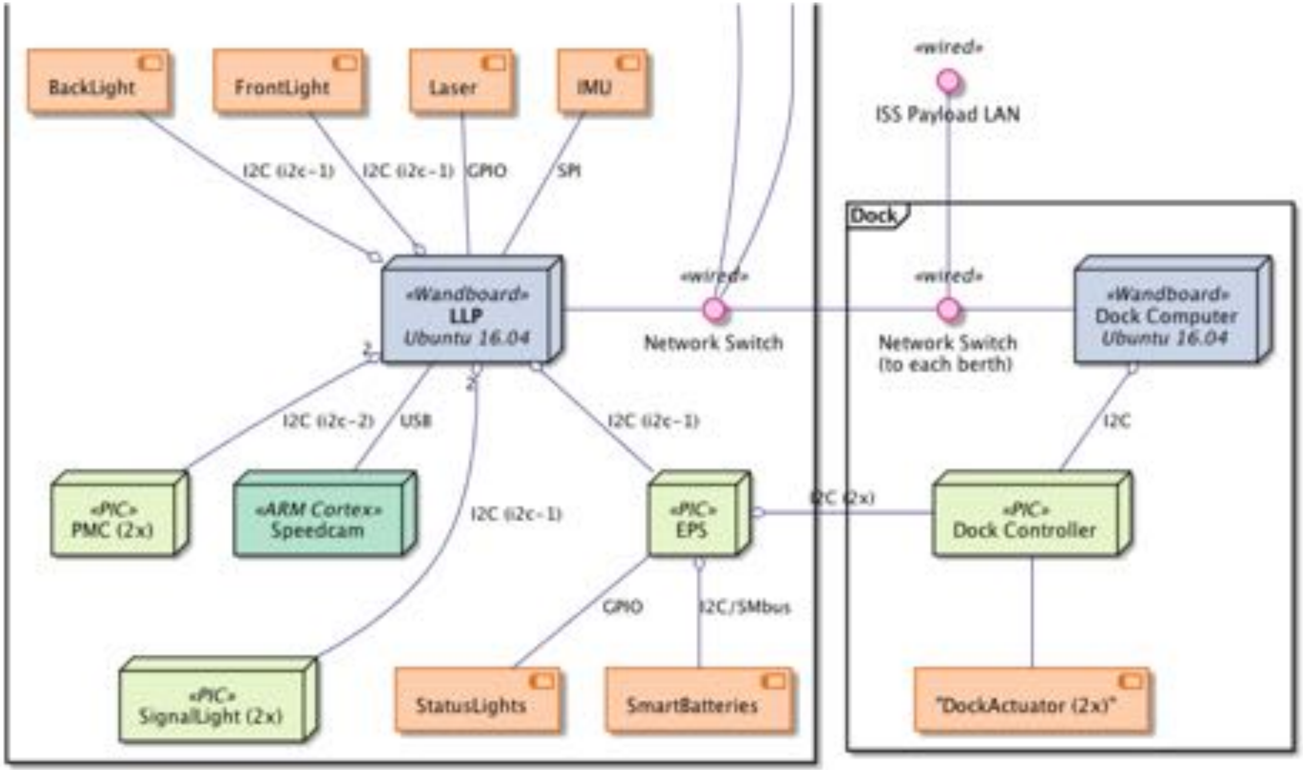
100 Mbps Network Switch

Name	Denomination	Processor	OS	Purpose
HLP	High Level Processor	Inforce 6601 Qualcomm Snapdragon 820	Android 7.1	HD Streaming, audio, touchscreen Guest Science Applications
MLP	Mid Level Processor	Inforce 6501 Qualcomm Snapdragon 805	Ubuntu 16.04	Vision Processing, Mobility, Communications and Management
LLP (and Dock)	Low Level Processor	Wandboard i.MX6 Dual ARM Cortex A9	Ubuntu 16.04	Localization (EKF only), propulsion control and power management

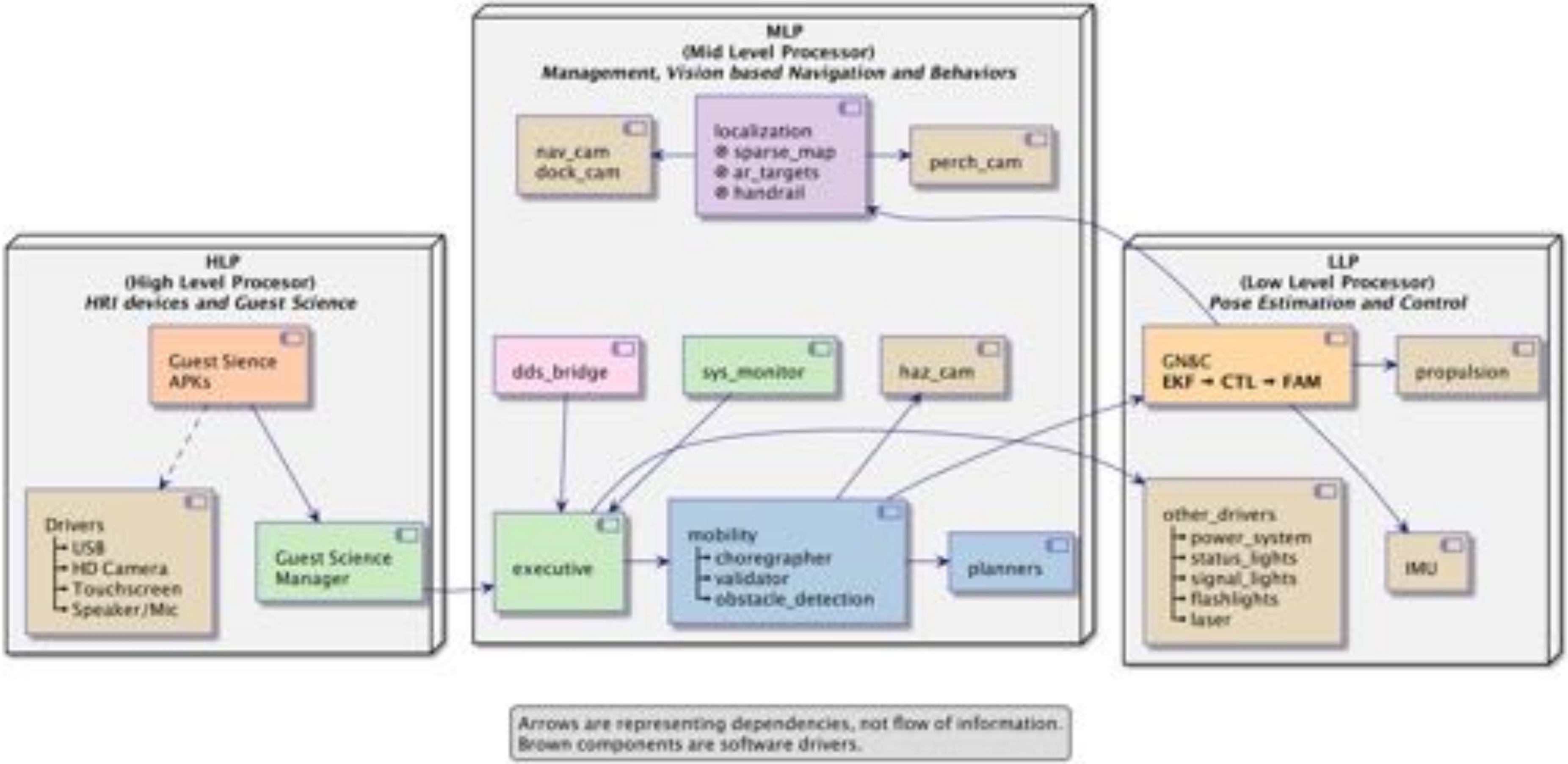
7 microcontrollers with custom firmware +
6 microcontrollers with COTS firmware


Astrobee Hardware Deployement Diagram
Astrobee
Speaker
Microphone
TouchScreen
Scicam
audio
audio
MIPI-DSI
MIPI-CSI
«Inforce»
HLP
Android 7.0
Hazcam
Dockcam
Navcam
Perchcam
USB
USB
USB
USB
«Inforce»
MLP
Ubuntu 16.04
USB
«PIC»
PerchArm
«wifi»
ISS Wifi
Microprocessor
(System On Chip computer)
ProcessorLegend
losange indicates master
if I2C device
communication protocol
indicated
MicrocontrollerLegend
DeviceLegend
Microntroller with custom
firmware
Hardware component (sensor
or actuator

Hardware deployment cont.



Astrobee Robot Software Overview



Software Components

- Communication Framework (ROS + DDS based)
- Simulator
- Localization
- Mobility
- Propulsion Control
- Management (Executive, Faults, DDS Bridge, Guest Science Management)
- Platform Management and development tools

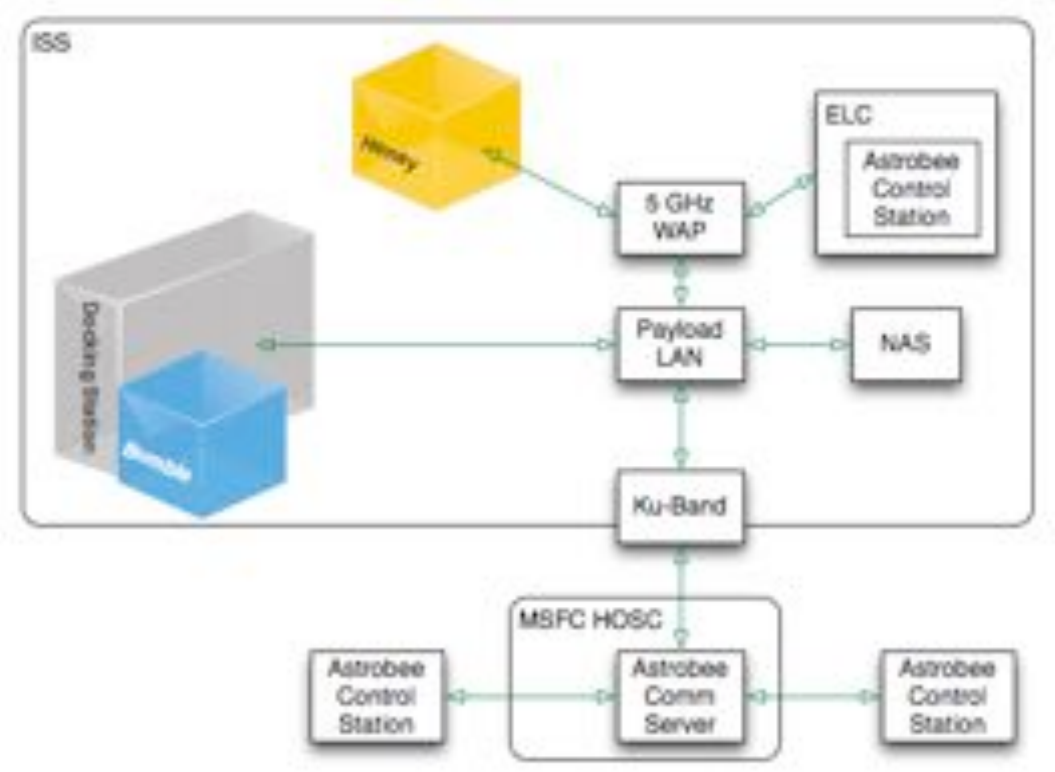
Number or ROS nodes for a simulation: ~36
Number of ROS nodes on Astrobee: ~48
(not counting HLP)

COCOMO (organic, C++ part) -> 126 man-month

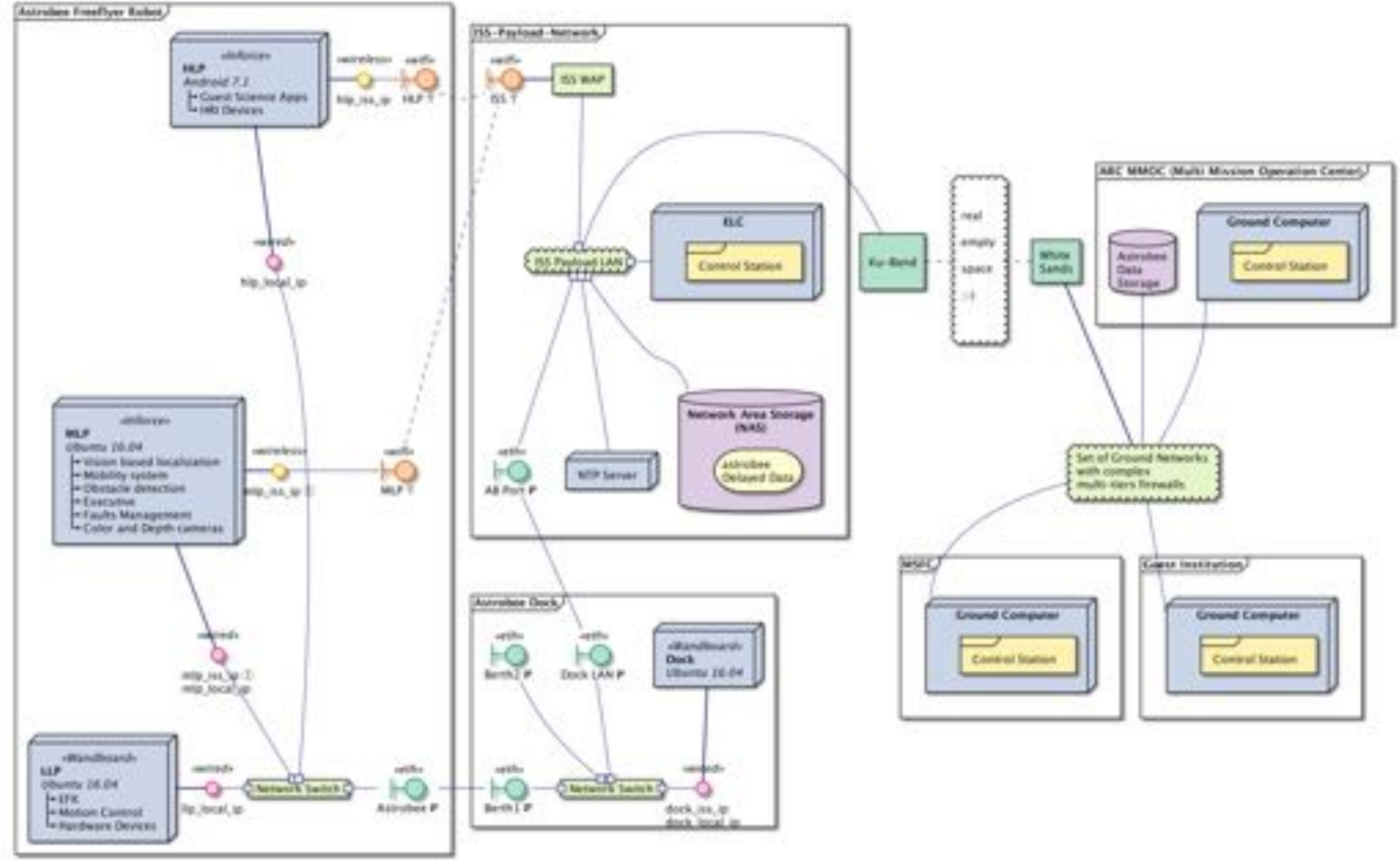
Language	files	comment	code
C++	263	11078	45023
Lua and config	76	2179	14380
C/C++ Header	158	4232	8209
Python	58	1798	6706
CMake	150	3631	5624
IDL MSG SRV ACTION	148	0	5139
XML	163	309	4271
Java	70	1764	3822
Bourne (Again) Shell	83	853	2522
SUM:	1182	25948	96228

Communications

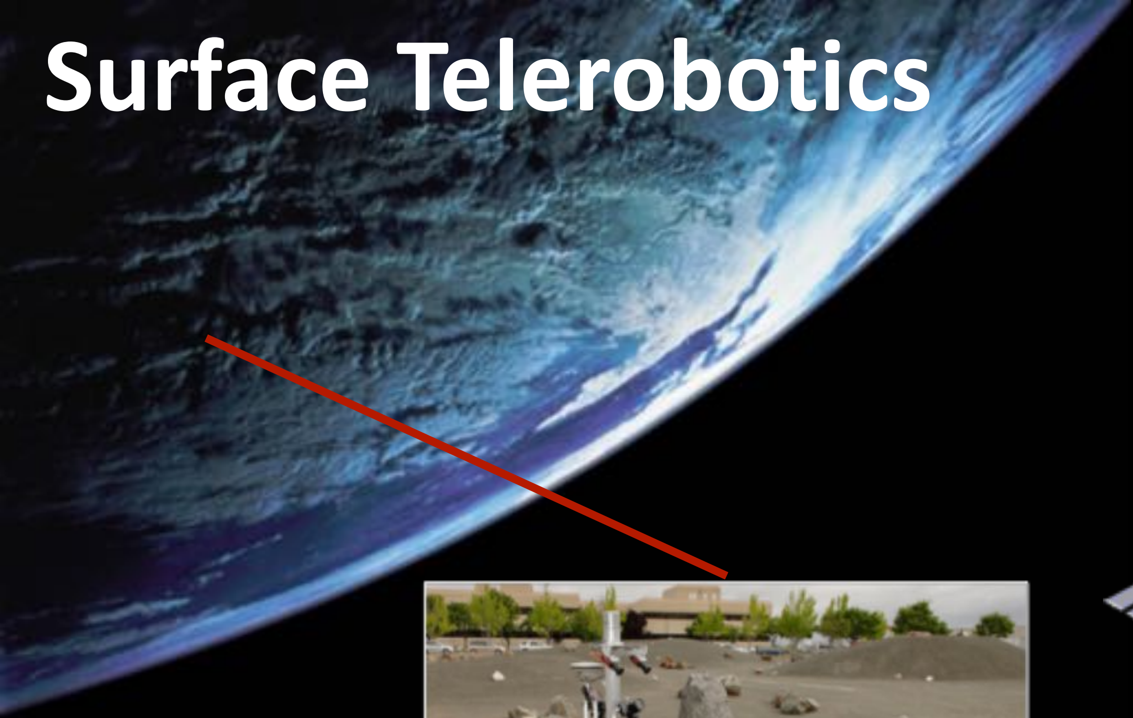
- Communicates through ISS WiFi when flying
- Single telemetry/video stream to ground
- Multiple ground stations can connect through server
- Large file transfers and software updates through Ethernet on the dock



Astrobee Network Deployement Diagram



Surface Telerobotics



Development and Testing of a Crew Controlled Planetary Rover System

Luca Parmitano and K10 Red

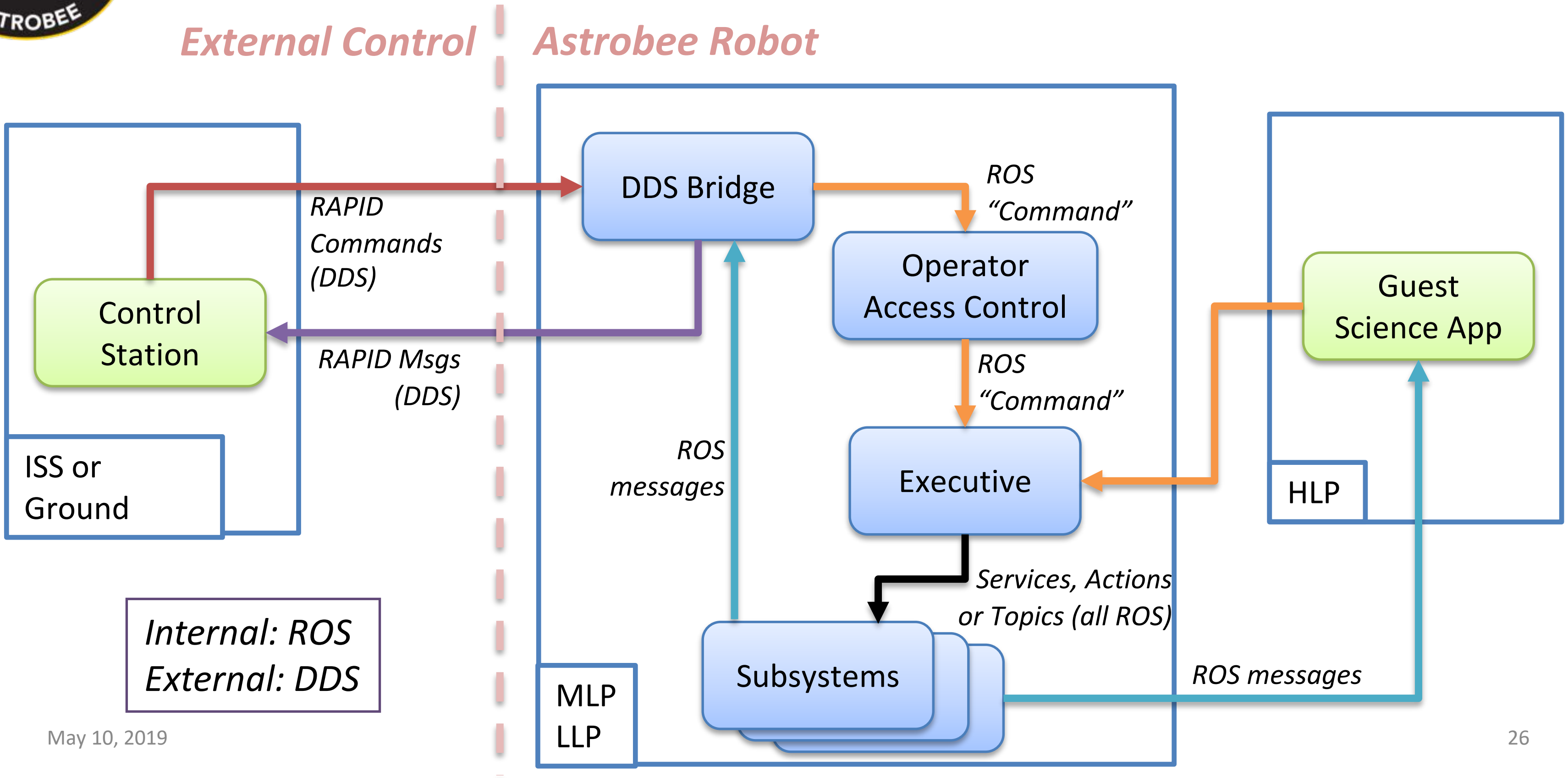
Communication Frameworks

- Key factors for ROS selection (vs. CFE):
  - Messages definition and serialization support
  - Better service isolation
  - Documentation & Support
  - Library of Robotics Algorithms Available
- Key factors for DDS + RAPID
  - Multiple Configurable Quality Of Service (QoS)
  - ISS Tested + Heritage from SmartSpheres

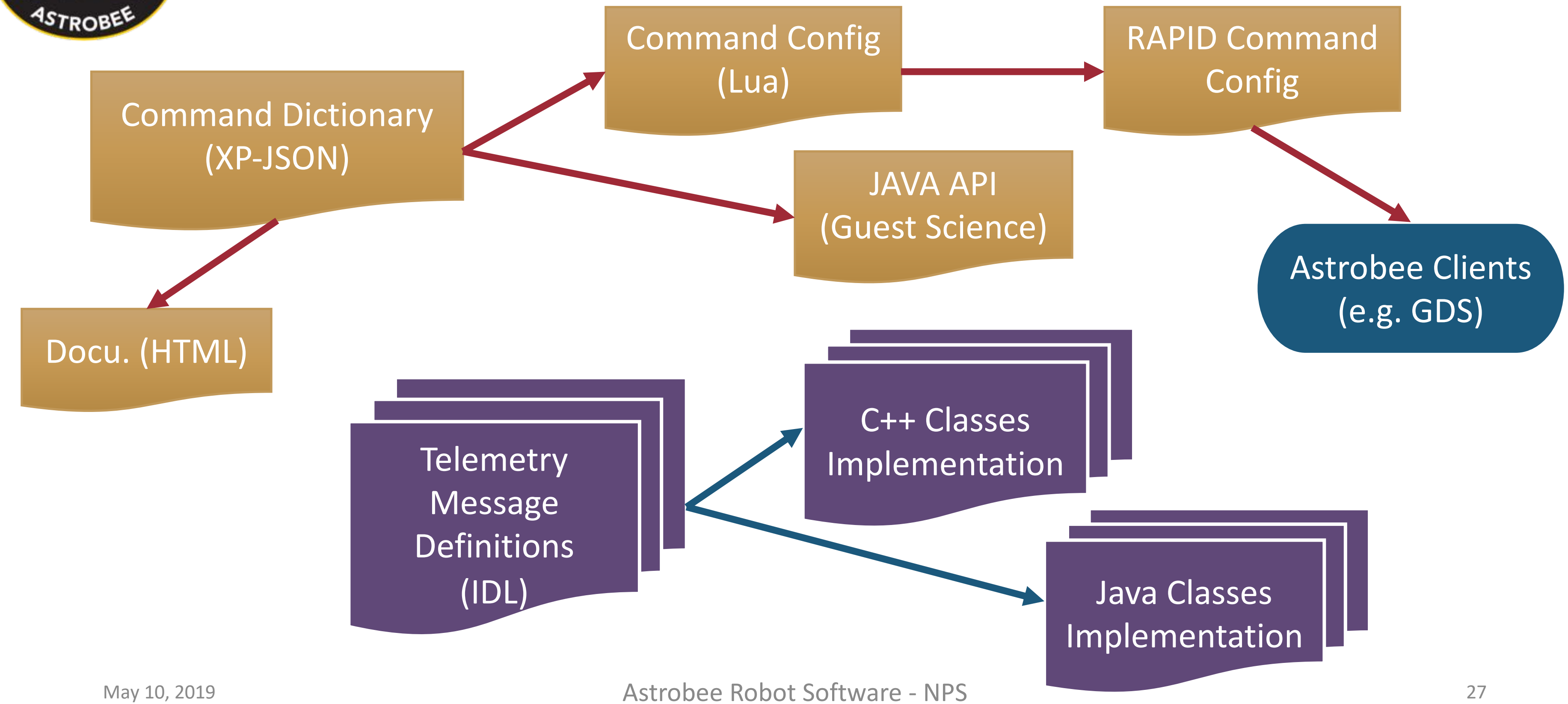
Candidates
Common Flight Executive (CFE)
Robotic Operating System (ROS)
Mobile Robot Programing Toolkit (MRPT)
Joint Architecture for Unmanned Systems (JAUS)
IRG RoverSW (SORA + **RAPID**)
Data Distribution Service (DDS)

Selected solution is hybrid of:
- ROS for onboard messaging
- DDS for remote communications

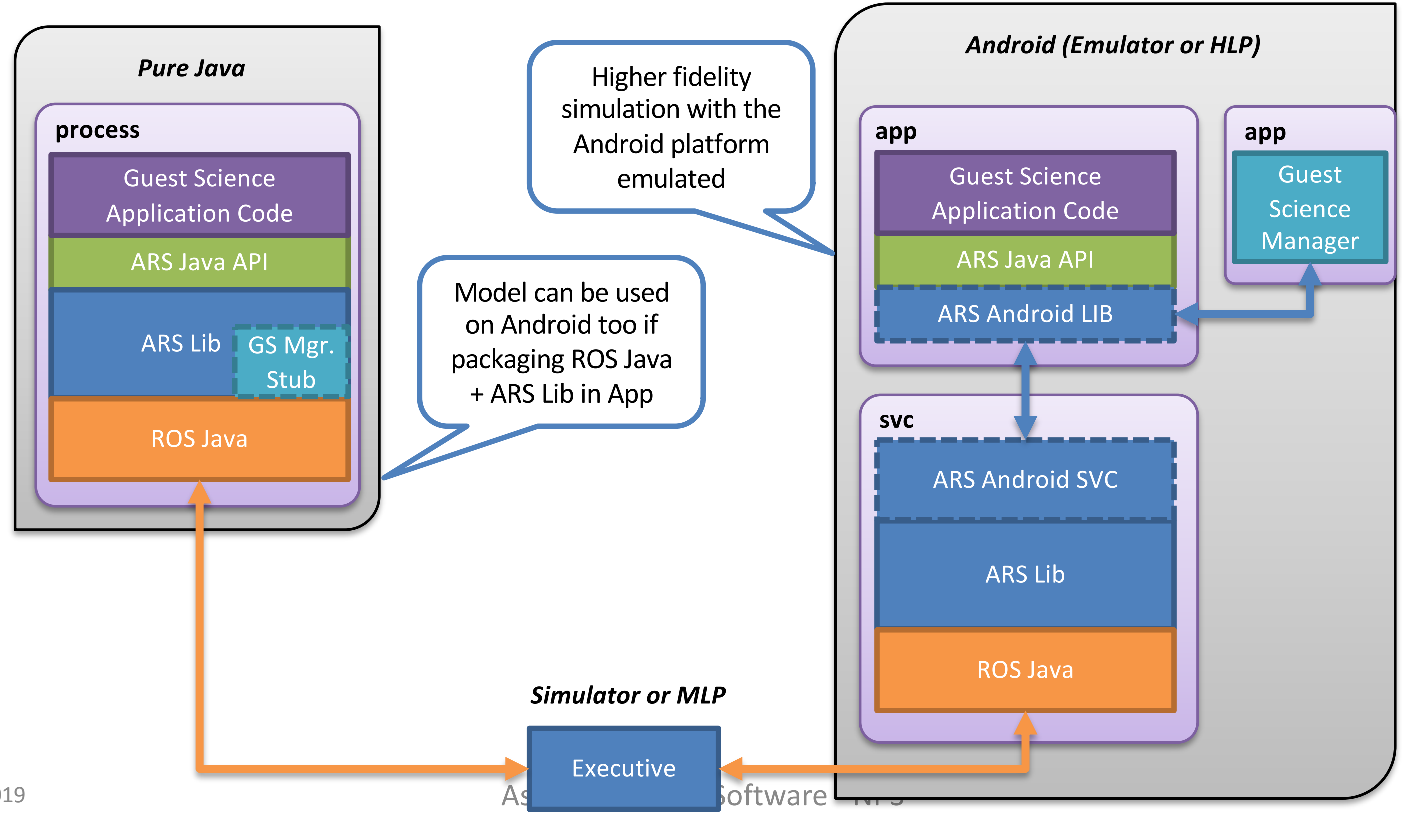
Dual Middleware – Unified API



Commands and Telemetry

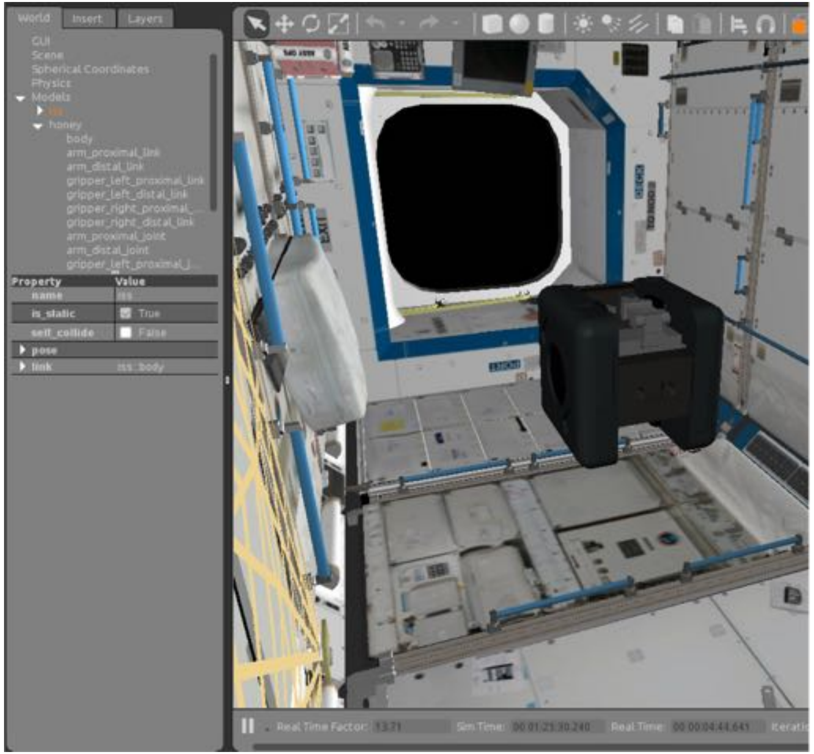


Guest Science Implementation



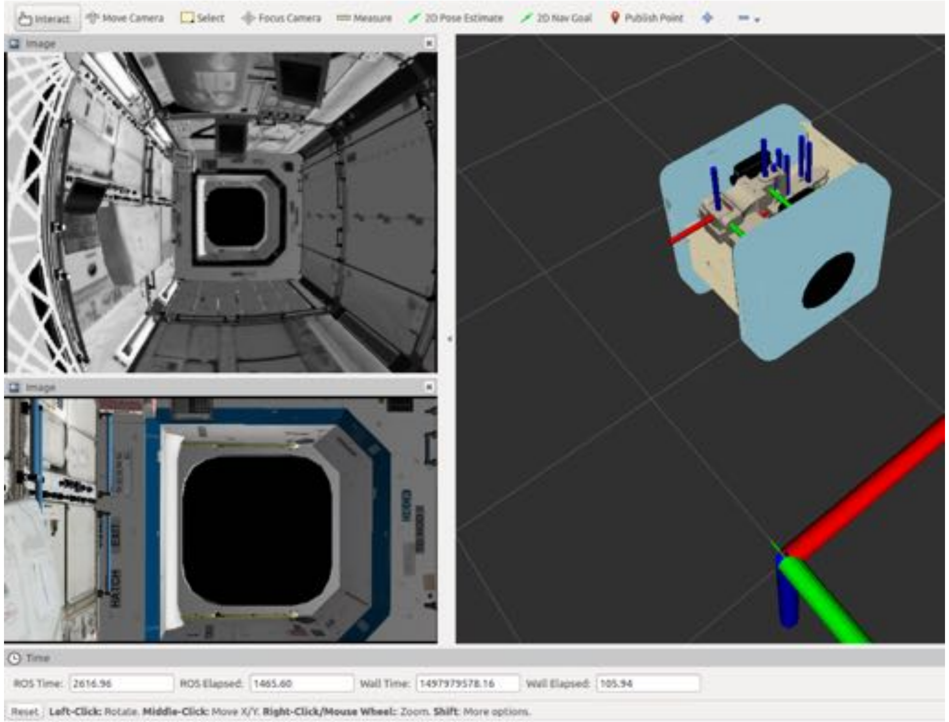
Astrobee Robot Software and ROS

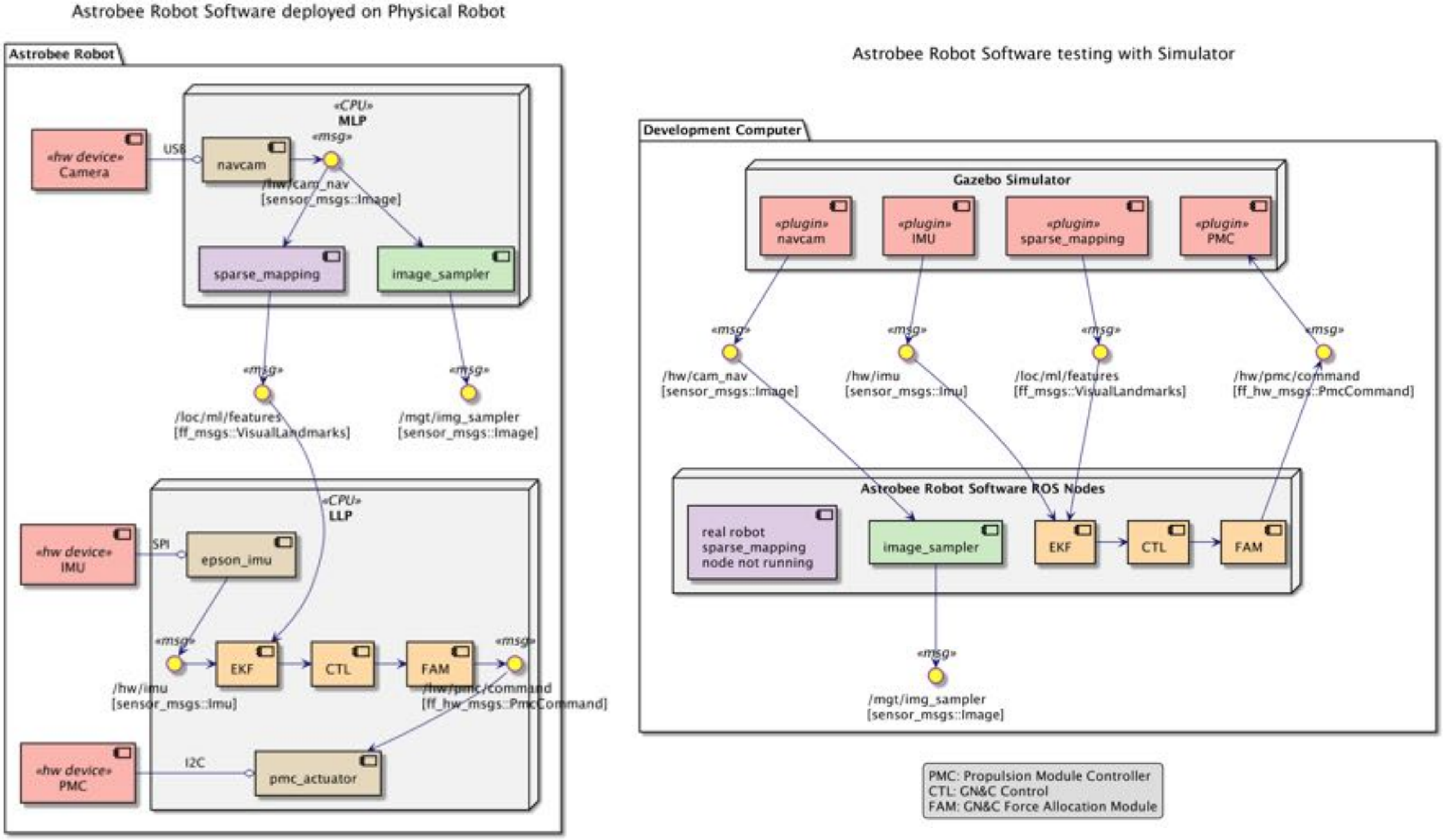
- Astrobee Robot Software makes extensive use of the open-source Robot Operating System (ROS):
  - Communication framework linking all “nodes” running on the target platform
  - Try to maximize the re-use of existing ROS messages benefit from existing ROS packages
  - Use ROS introspections tools to rapid debugging
  - Use ROS facilities to record/replay/analyze data
  - Use some ROS/Gazebo components for the simulator
  - Popular framework for guest scientists



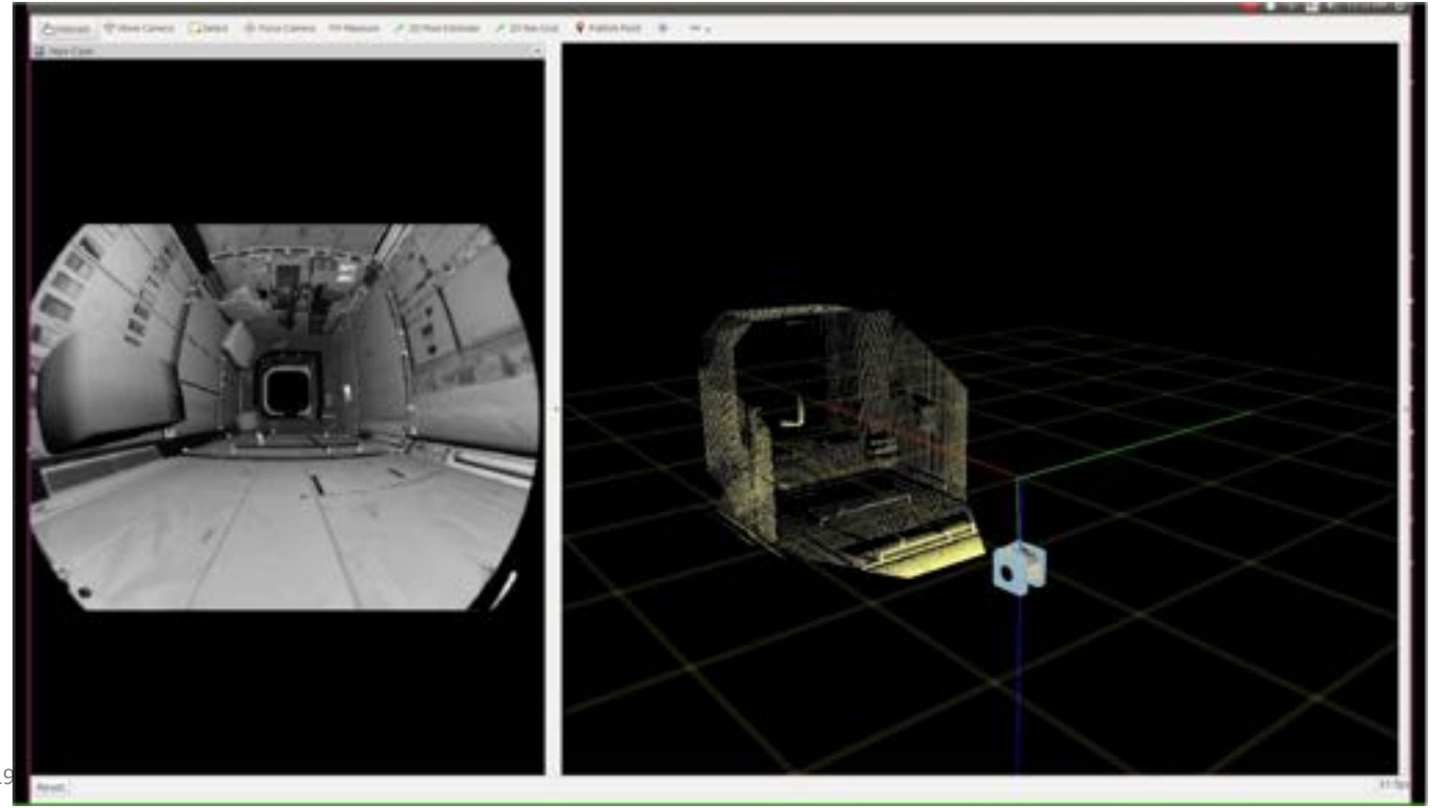
Simulator

- Gazebo based dynamics model, imagers, arm, lights and ISS model
- Custom propulsion system and some localization sensors
- Can run all nodes on desktop or some nodes on target development board
- Can run real time, or up to 10x speed on desktop with good graphic card



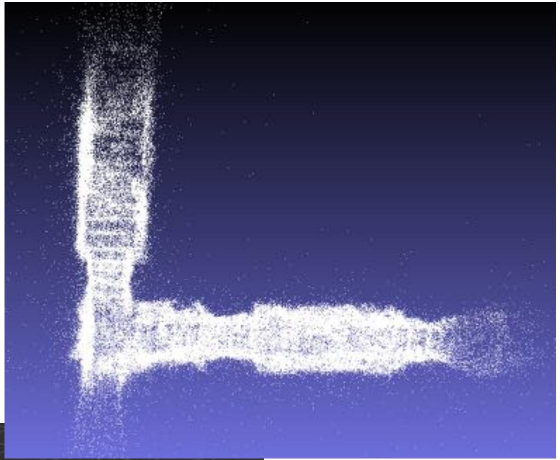

Astrobee Robot Software deployed on Physical Robot
Astrobee Robot
«hw device» Camera
USB
«CPU» MLP
navcam
«msg»
/hw/cam_nav [sensor_msgs::Image]
sparse_mapping
image_sampler
«msg»
/loc/ml/features [ff_msgs::VisualLandmarks]
«msg»
/mgt/img_sampler [sensor_msgs::Image]
«CPU» LLP
«hw device» IMU
SPI
epson_imu
«msg»
/hw/imu [sensor_msgs::Imu]
EKF
CTL
FAM
«msg»
/hw/pmc/command [ff_hw_msgs::PmcCommand]
«hw device» PMC
I2C
pmc_actuator
Astrobee Robot Software testing with Simulator
Development Computer
Gazebo Simulator
«plugin» navcam
«plugin» IMU
«plugin» sparse_mapping
«plugin» PMC
«msg»
/hw/cam_nav [sensor_msgs::Image]
«msg»
/hw/imu [sensor_msgs::Imu]
«msg»
/loc/ml/features [ff_msgs::VisualLandmarks]
«msg»
/hw/pmc/command [ff_hw_msgs::PmcCommand]
Astrobee Robot Software ROS Nodes
real robot sparse_mapping node not running
image_sampler
EKF
CTL
FAM
«msg»
/mgt/img_sampler [sensor_msgs::Image]
PMC: Propulsion Module Controller
CTL: GN&C Control
FAM: GN&C Force Allocation Module

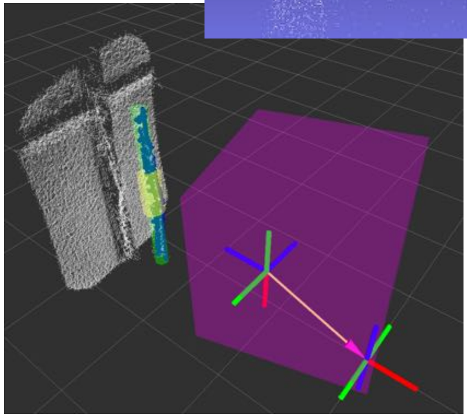
Simulation usage example: Octomap for Collision Detection

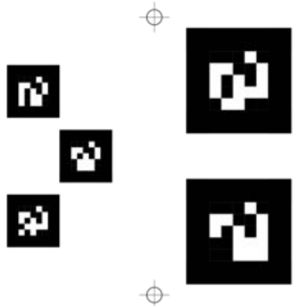


Localization

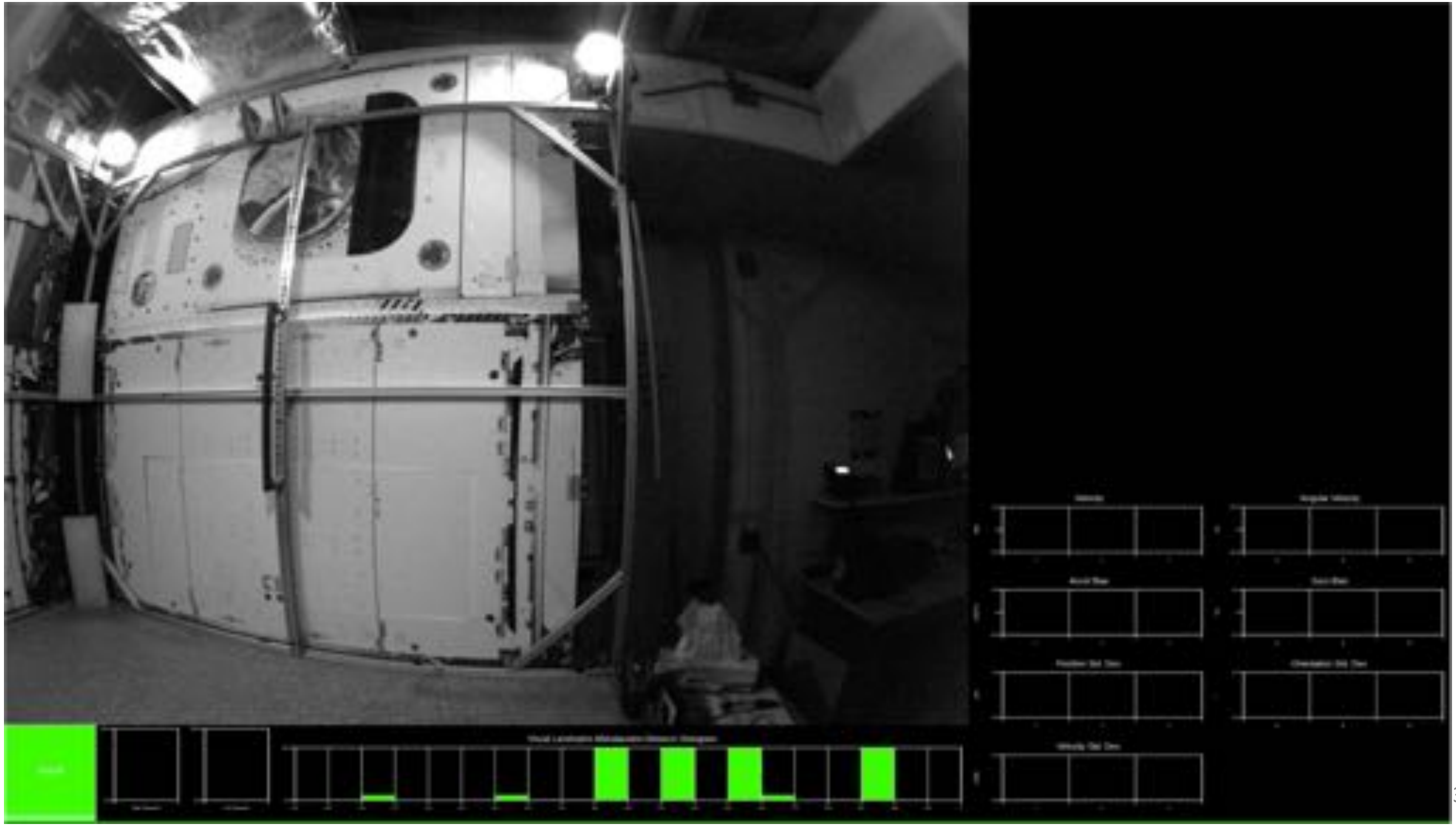
- Four vision modules send observations to the Pose Estimator
  - Sparse Mapping : runs for regular navigation, provides absolute position within the ISS map (~5cm)
  - Visual Odometry: provides velocity and maintain position when no features are available
  - Handrail Detector: only runs for perching
  - AR Tags: only runs for docking (~1cm)





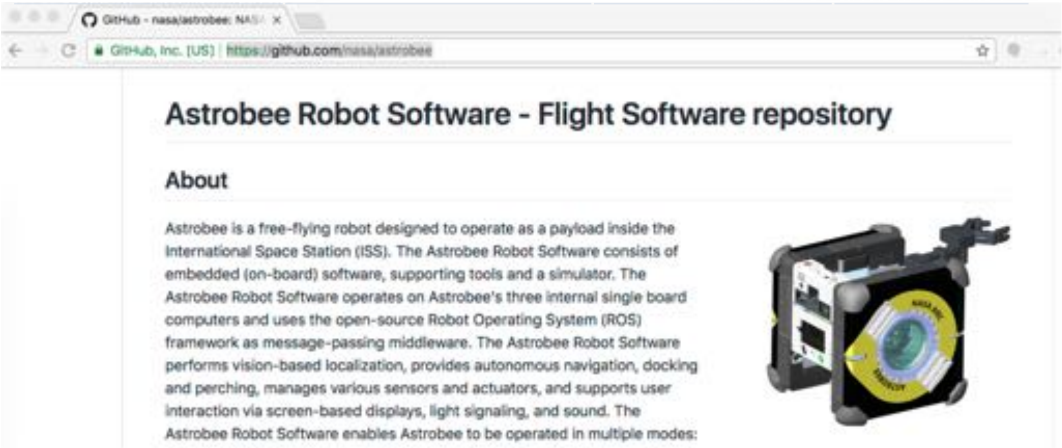


Localization in action

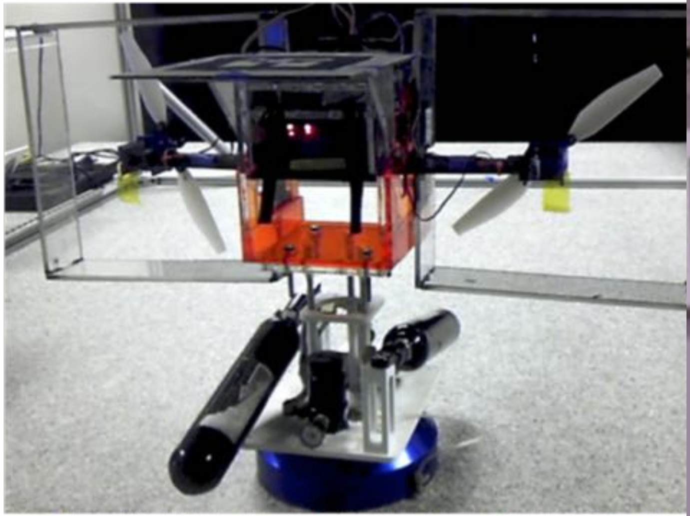


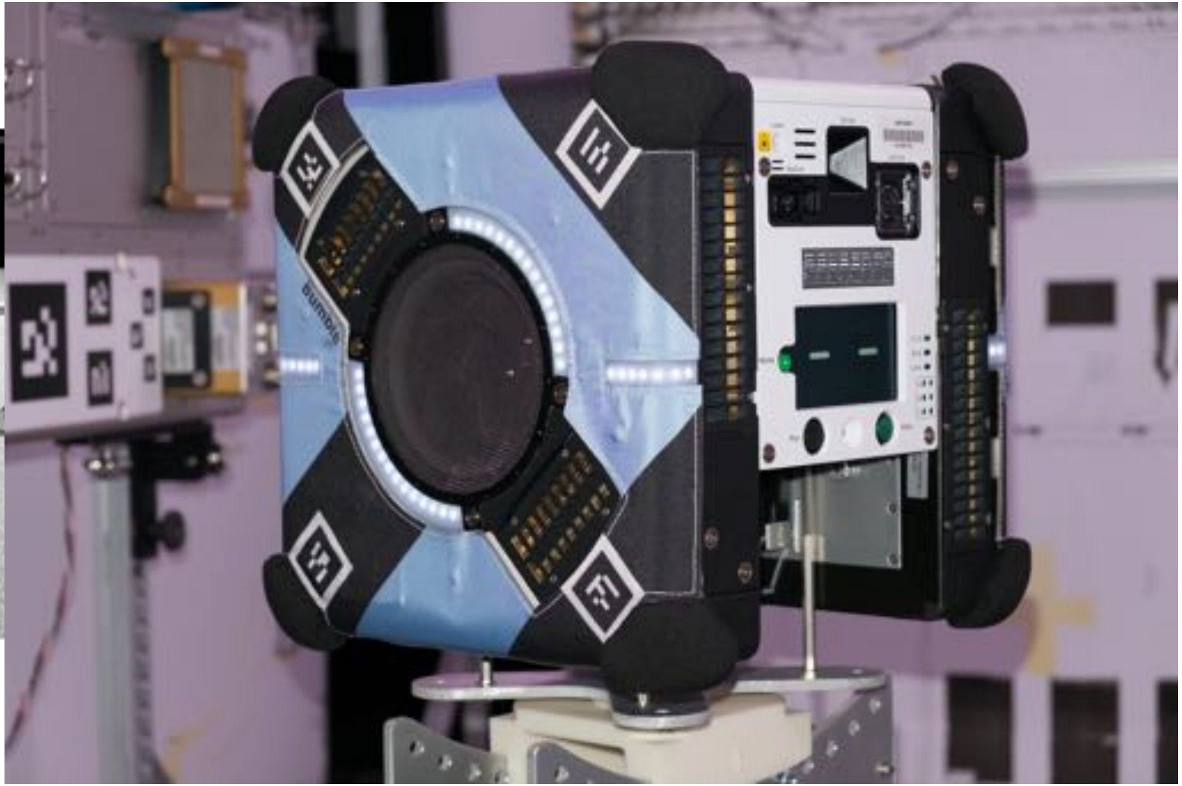
Some Lessons Learned

- Balancing research with space qualified deliverables
- Hardware constraints / development cycle
- Software optimization
- Open Source



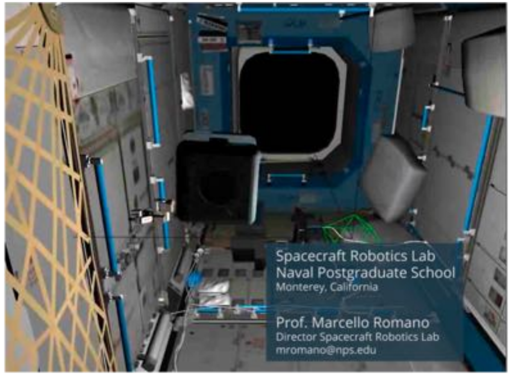
Astrobee Evolution





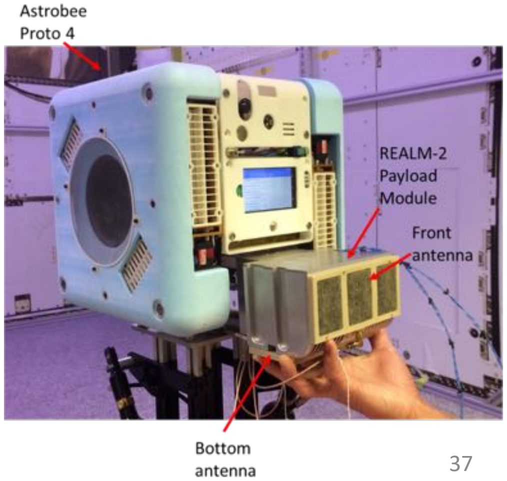
Users Case Study

ZeroRobotis @ MIT:
Pure simulation



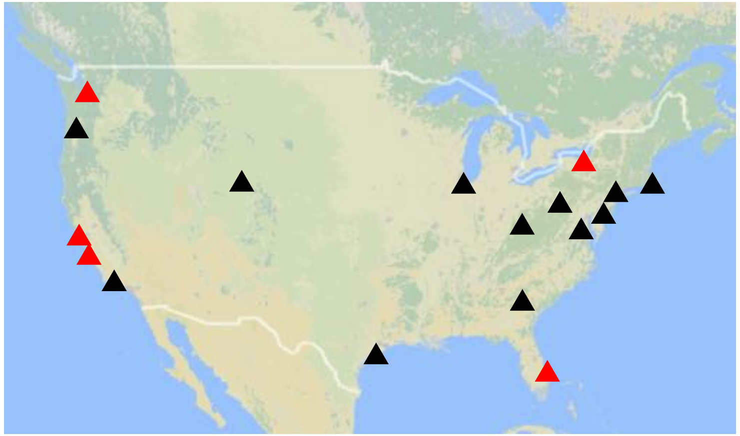
Space Lab (NPS):
Control algorithms

REALM 2 (JSC):
Hardware payload

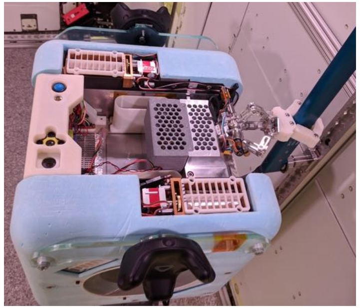


Astrobee Research Community

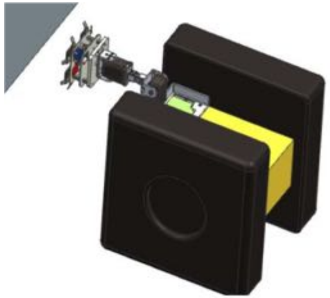
- CASIS
  - MIT (Zero Robotics), Airbus
- ECF
  - Georgia Tech, CU Boulder, Brown
- ESI
  - IIT, **Columbia**, **Stanford**, UMD
- NSTRF
  - Stanford, U-Penn, USC, MIT (2), UCSD, Tufts , Oregon St., Cornell
- SBIR/STTR
  - **FIT**, Honeybee, **Tethers Unlimited**, Altius, Energid
- Other
  - REALM-2 (AES), MIST (NASA CIF), **NPS** (Navy), MIT/JPL (SSTP)



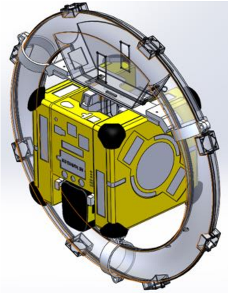
Guest Science Concepts



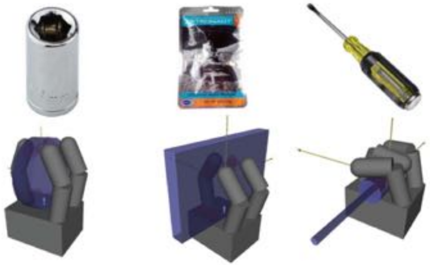
Prototype Astrobee arm based on Canfield joint, enabling new motions (**Tethers**)



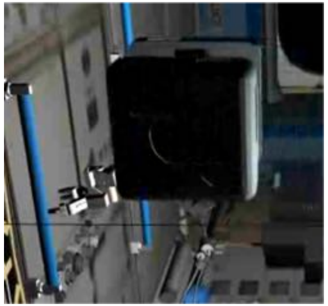
Gripper concept based on gecko-like adhesives (**Stanford**)



Adapting the RINGS magnetic propulsion payload to Astrobee (**FIT**)



Improving gripper dexterity without increasing actuator count (**Columbia**)

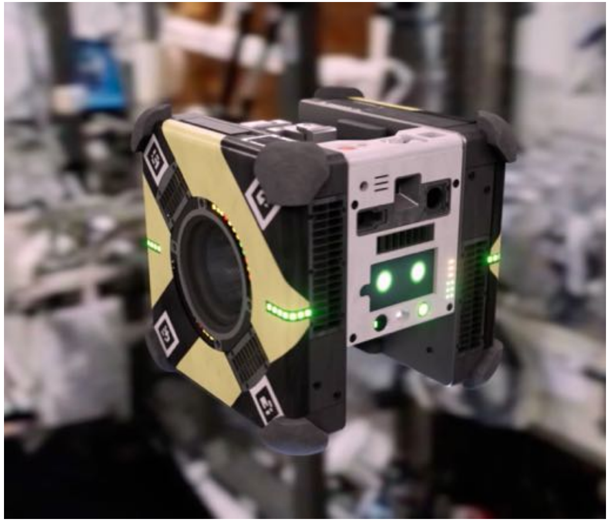


Arm grasping controller developed using Astrobee open source simulator (**NPS**)

and many more…

Conclusion

- The Astrobee Robot Software manages a powerful and complex hardware platform
- Dual Middleware approach (DDS + ROS)
- Cell phone technologies + modern software libraries in space
- Open Source release under Apache-2 license
- Astrobee first autonomous flight in 2 weeks!



Next

- It is really the start of cool experiments!
- Demonstrate autonomous flying
- Improve reliability and algorithms
- Support research projects

Questions?

https://www.nasa.gov/astrobee
https://github.com/nasa/astrobee

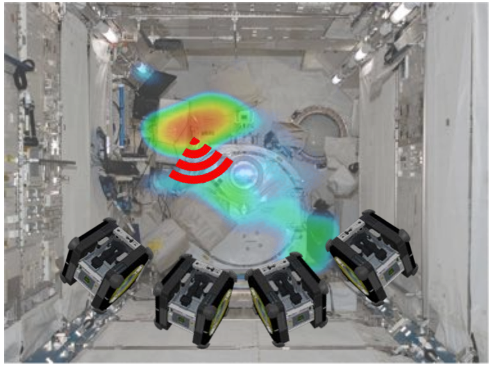
Lorenzo.Fluckiger@nasa.gov

Acknowledgments

- Funded by:
  - NASA Game Changing Development Program (Space Technology Mission Directorate)
  - ISS SPHERES Facility (Human Exploration and Operations Mission Directorate)
- Thanks go to:
  - ISS SPHERES Team
  - ISS Payloads Office
  - JSC Flight Operations Directorate
  - ISS Avionics and Software
  - Advanced Exploration Systems Program

Future Applications

- Astrobee will help prove out the concept of "Caretaker Robots" for future exploration architectures
- Allows monitoring, maintenance and repair of a facility before and between crews
  - Gateway may be crewed just six weeks per year!
  - Critical need to care for spacecraft when crew are not present
- Inspection functions can include:
  - Spot checks
  - Surveys
  - Automated change detection and trending
  - Localizing problems
- With dexterous robotic manipulation capabilities, future tasks could include:
  - Maintenance
  - Repair
  - Cargo transfer

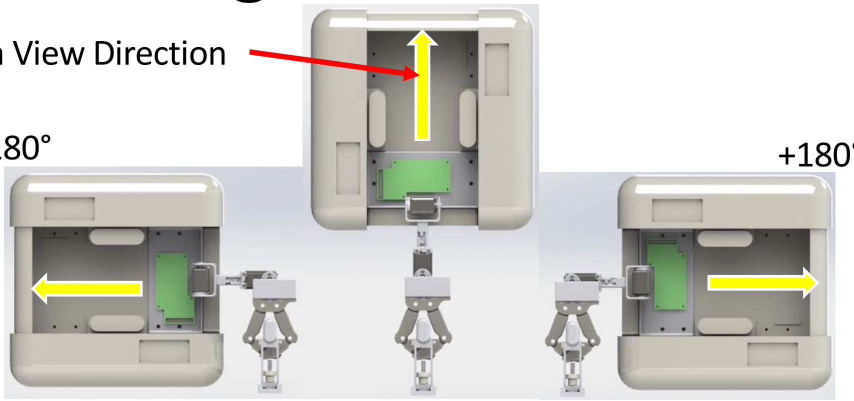


Isolating faults: Ultrasonic leak detection

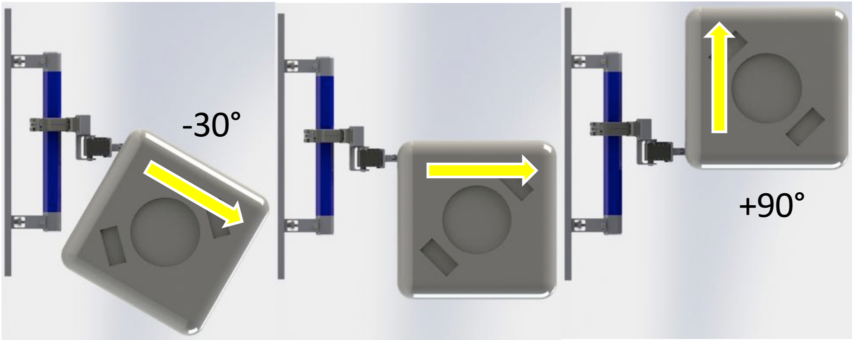
Off-load routine astronaut tasks: Robotic cargo transfer

System Description: Perching Arm

- Designed to grasp handrails
- Stows completely in payload bay
- Acts as a pan-tilt unit while perched
- Flexible and back-drivable
- May be perched manually



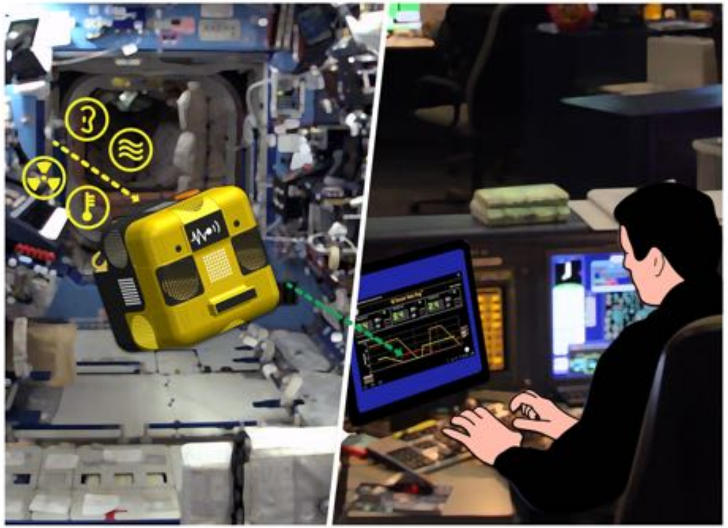
Astrobee Perching Arm pan range



Astrobee Perching Arm tilt range

System Description: Ground Data System

- Astrobee Control Station
  - Sortie planning tool
  - Execution monitoring
    - Live telemetry
    - Image and video streams
    - 3D virtual display
  - Supervisory control (run plans or single commands)
  - Typically used by ground operators
- Crew Control Station (used rarely) runs on an EXPRESS Laptop Computer (ELC)
- Server for archiving and distributing Astrobee data
- Suite of engineering tools to support maintenance and software upgrades



Control Centers

- Astrobee can be operated from almost anywhere
  - Flight controllers at Mission Control Center (JSC)
  - Payload controllers at Payload Operations Integration Center (MSFC)
  - Guest scientists at Multi-Mission Operations Center (ARC) or home institutions
- Provides operators with a mobile camera for improved ground situation awareness during crew activities
  - Optimize viewing angles using the pan/tilt or by relocating Astrobee
- Supervisory control means 100% of operator’s attention is not required

Camera Scenario: OSO observes crew maintenance task

- Schedule Astrobee activity
- Use Plan Editor to create 1) a plan that moves Astrobee to crew activity site, and 2) a plan that returns Astrobee to the Dock
- Shortly before crew activity, execute 1st plan
- At start of crew activity, switch to Teleoperate to begin streaming HD video and adjust pan and tilt
- If crew blocks camera view, teleoperate Astrobee to unperch, fly to new handrail, re-perch.
- During LOS, Astrobee will continue to record video
- At conclusion of crew activity, end HD video streaming and execute 2nd plan to return to dock
- Once Astrobee is docked, if desired, downlink recorded video file.

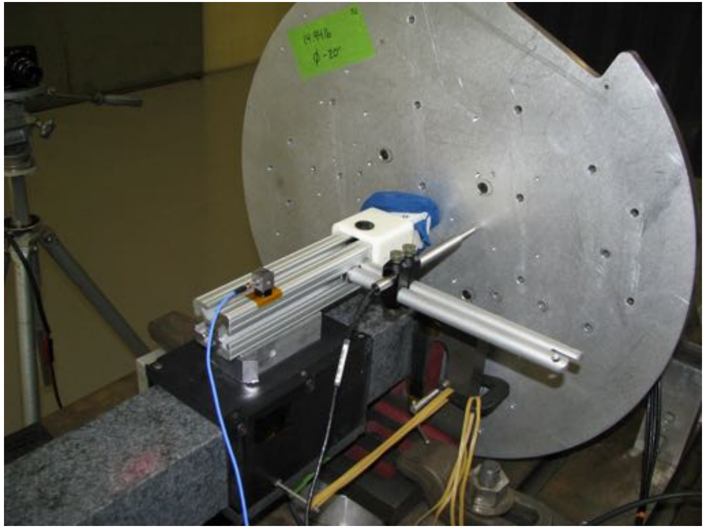
Concept of Astrobee perching for crew activity documentation

Other Operational Considerations

- 3 Astrobees will be on orbit, but only 2 Docking Station berths are available
  - Third free flyer will be stowed and will require crew to charge and install batteries before use
- Multiple free flyer operations
  - Each Astrobee accepts commands from only one Control Station at a time
  - Any Control Station may monitor telemetry from multiple Astrobees
  - Allows operators to watch for interference between multiple Astrobee activities
- ISS operators must schedule use of Astrobee with the Astrobee Facility

Challenges: Safety

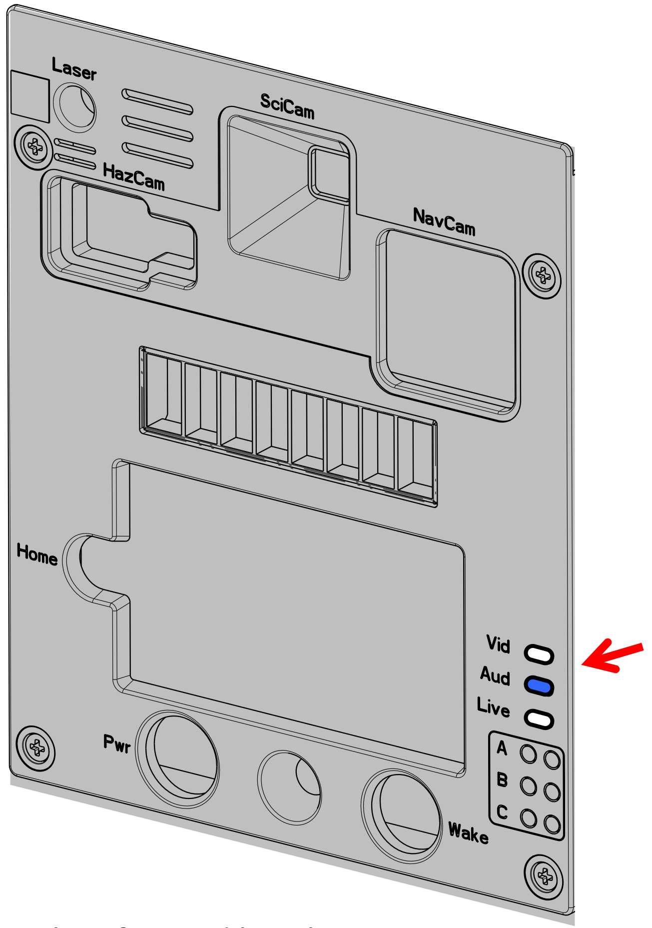
- Unique collision hazards: Crew can move faster than Astrobee can move out of the way
- Mitigations
    - Light (low mass, ~10 kg)
    - Slow (max speed 0.5 m/s)
    - Soft (corner bumpers and foam padding)
    - Signal lights/noise when entering hatchway
    - Keep crew aware through operational techniques
        - Daily Plan
        - Daily conferences
        - CapCom calls as needed
- Screens cover intakes
- Grills cover nozzle flaps



Bumper collision test rig

Challenges: Privacy

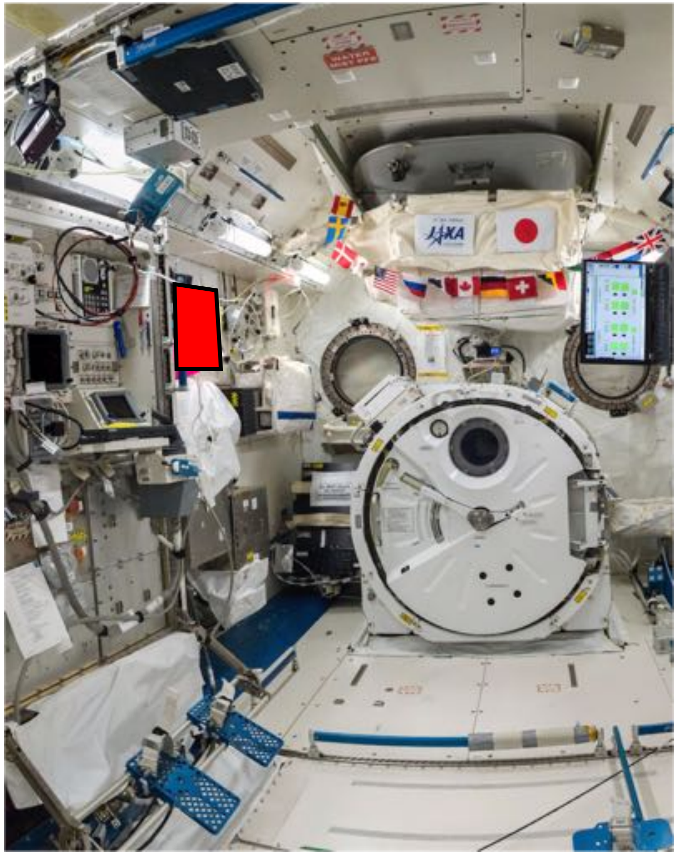
- Some cameras always on whenever Astrobee is operating
  - Privacy status LEDs on forward and aft faces indicate when cameras or mic are on and/or streaming
- Crew actually most concerned about live audio
  - In addition to privacy status LEDs, signal lights on left/right Prop Modules will shine blue when mic is on
- Keep-Out Zones (KOZ) can be used to keep Astrobee out of areas where:
  - A crew member is exercising
  - A medical experiment is in progress
  - A sensitive payload is operating
  - An exhaust vent creates fast-moving air that might blow Astrobee off course



Astrobee forward bezel privacy status LEDs

Challenges: Placement

- Difficult to find a “permanent” location for the Docking Station
  - Occupies significant space
  - Want to avoid high traffic areas
  - Anticipated service life until 2024, will last through many changes to ISS
- Lesson learned: expect to be moved, and be flexible
  - Dock design now has many mounting configurations with adjustable brackets, based on both seat track and hook-and-loop
  - Accommodates many possible mounting locations
- Initial location: JAXA has agreed to host the Astrobee dock in the JEM-Pressurized Module Port Endcone, Aft



Initial dock location in red, JPM1A7