HOPAS on OPS-SAT

Final Presentation – 07/12/2022

DEFENCE AND SPACE

AOCS/GNC & Flight Dynamics (TESOA)
On-board Software and Data Processing (TESOD)

Introduction

Project Objective

The goal of this project is to demonstrate the HOPAS (Hybrid Online Policy Adaptation Strategy) in flight. It would be amongst the first experiments on OPS-SAT to perform close loop attitude control, as well as one of the first algorithms running AI in a control system on board of a spacecraft, while also demonstrating the ability to learn on board.

Agenda

I. Context : what is HOPAS, how we got here and how to run it on OPS-SAT
II. Implementation : HOPAS, from simulator to OPS-SAT
III. Flight results : early conclusions from the first in flight tests

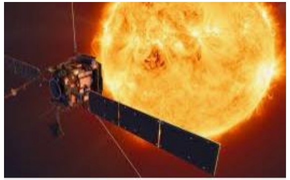
The Team

- Vivien Croes – Project Management
- Pierre Lachevre – Tech Lead and AI Engineering
- Ilke Karsli Terjan – On-board Software Engineering
- Mark Watt – AOCS Senior Specialist
- Maria Carrillo Barrenechea – AI and AOCS Engineer
- Carlos Hervas Garcia – AI Expertise Support

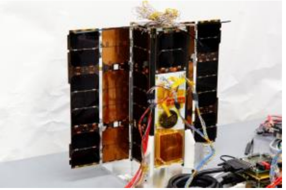
Context

2019
- Start of the GNC-v.AI Project
- It demonstrated the potential of Reinforcement Learning (RL) technologies on AOCS use cases

2021
- Development of the HOPAS algorithm, which uses techniques explored as part of GNC-v.AI in an online manner
- Demonstration of the concept on a Solar Orbiter use case



2022
- Patent submission for the HOPAS algorithm
- **IOD attempt on ESA OPS-SAT**



Next
- Complete performance campaign in flight
- Further R&D to improve learning speed
- Implementation on future missions

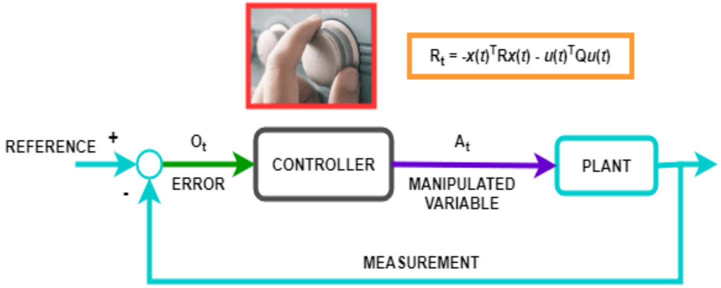
?

HOPAS on OPS-SAT

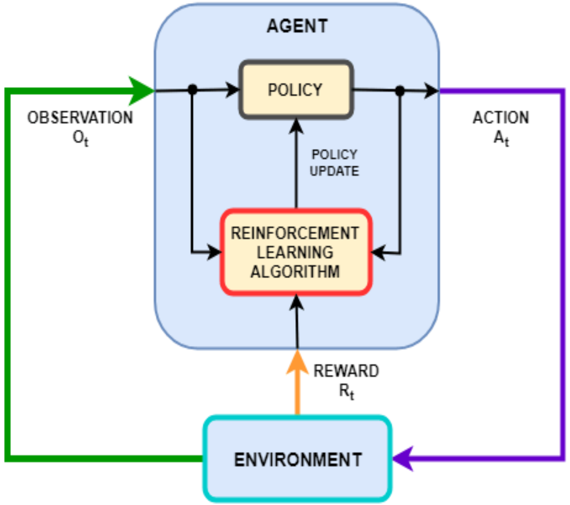
HOPAS description

- HOPAS purpose: Same as ADCS
  - Correct a variable to a desired reference
- HOPAS adds Reinforcement Learning
  - Performance improvement online
  - Adapt to different environments

- Hybrid:
  - Nominal controller
  - + Neural network

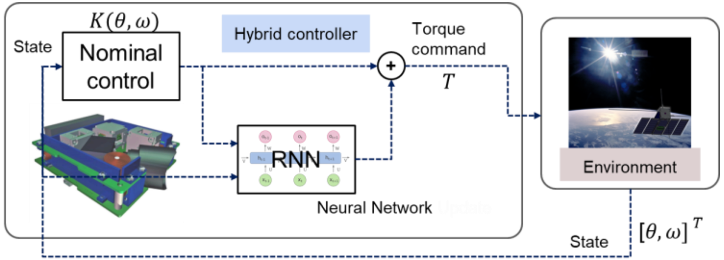


ADCS traditional control loop



Reinforcement Learning

Source: MATLAB



HOPAS on OPS-SAT

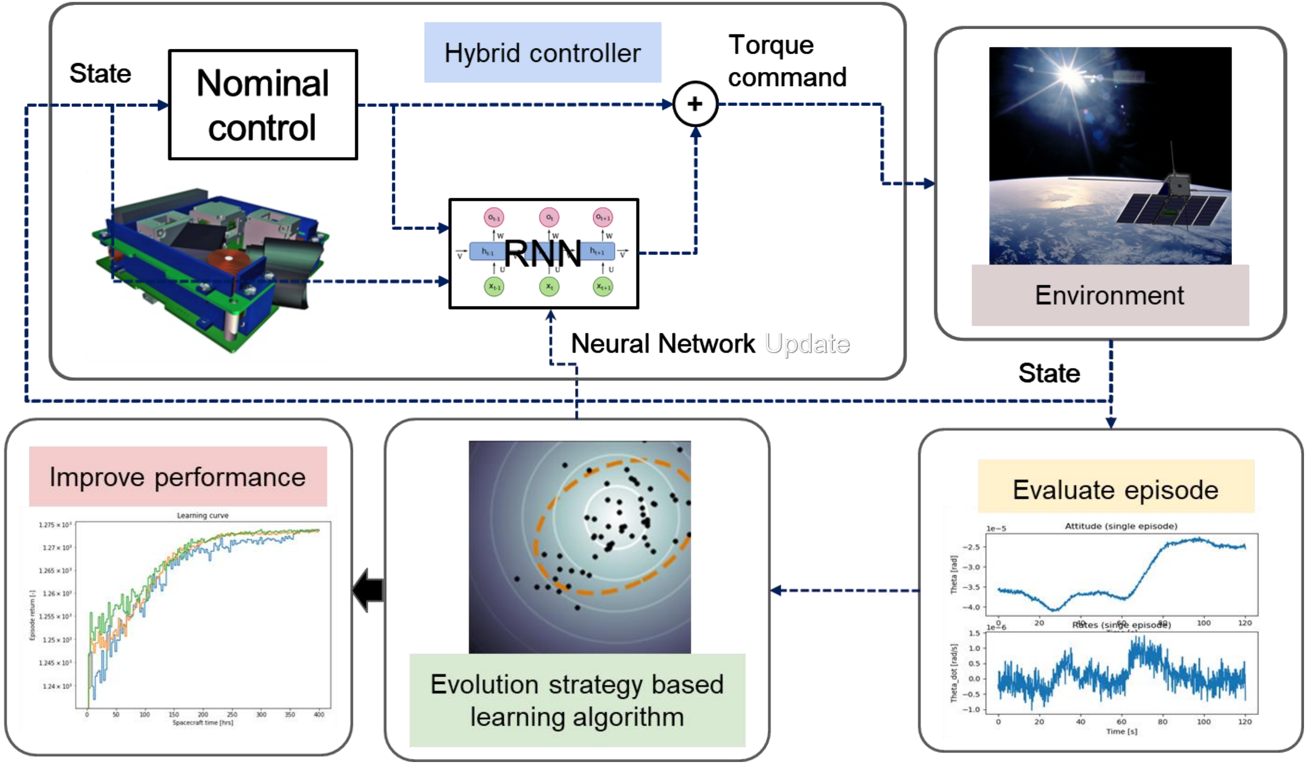
Context and introduction to HOPAS

Implementation on OPS-SAT

Flight Campaigns

Conclusion and way forward

HOPAS description



HOPAS on OPS-SAT

Context and introduction to HOPAS

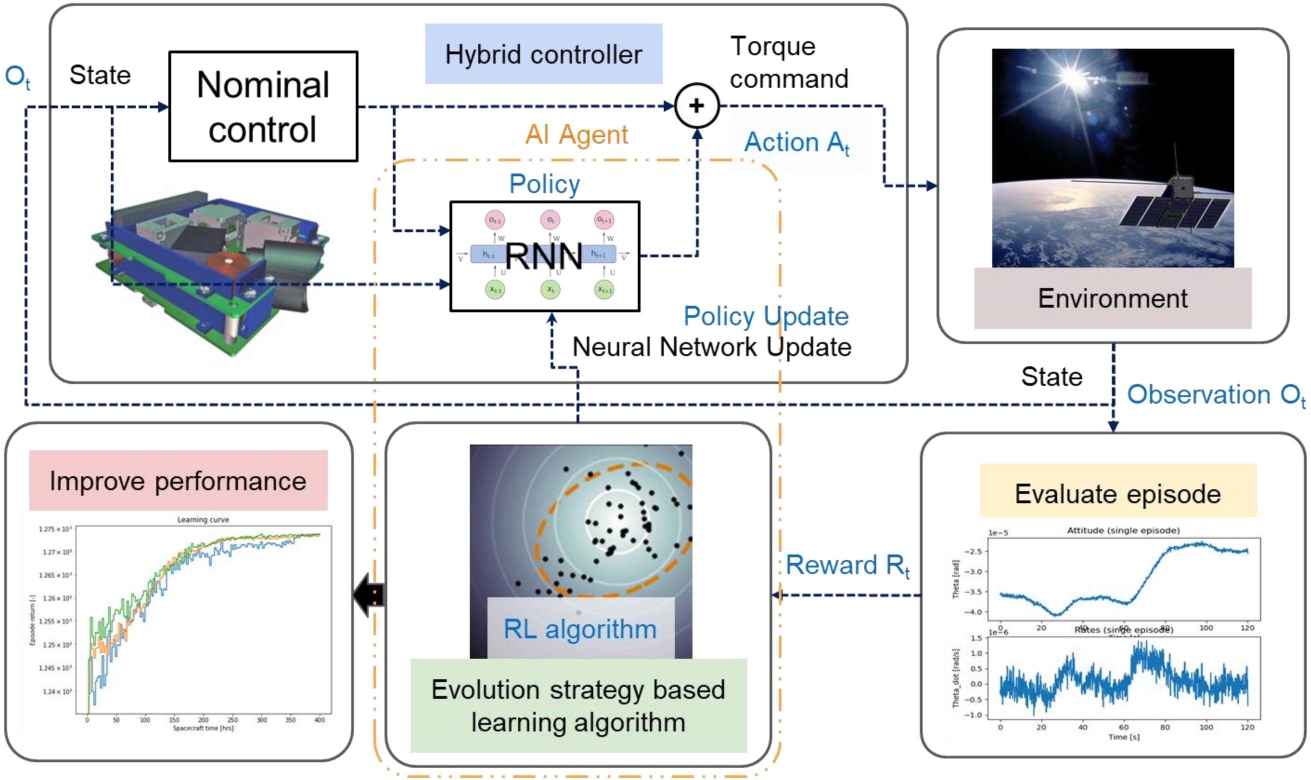
Implementation on OPS-SAT

Flight Campaigns

Conclusion and way forward

AIRBUS

HOPAS description



HOPAS on OPS-SAT

Context and introduction to HOPAS

Implementation on OPS-SAT

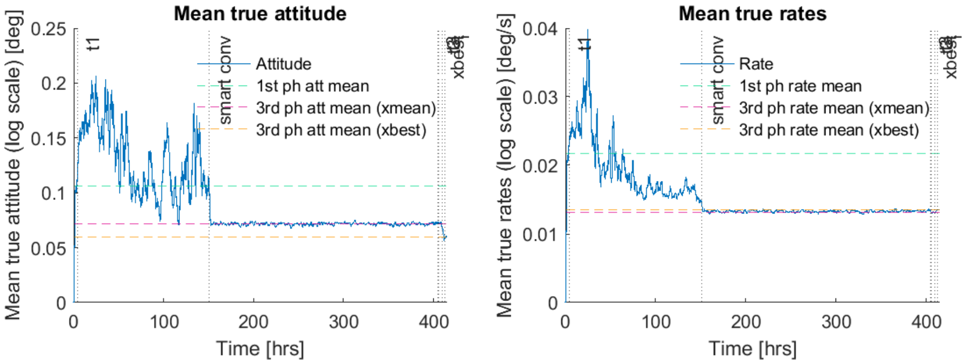
Flight Campaigns

Conclusion and way forward

AIRBUS

HOPAS Tuning for OPS-SAT

- HOPAS is meant to be as generic as possible but still needs some tuning

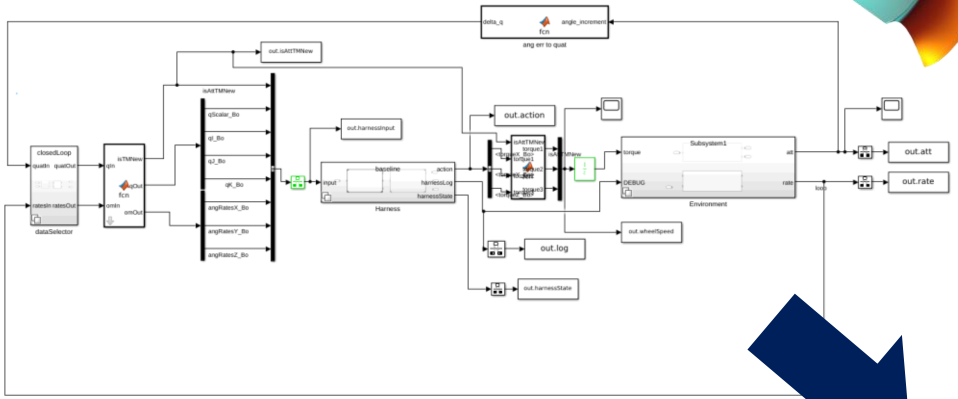


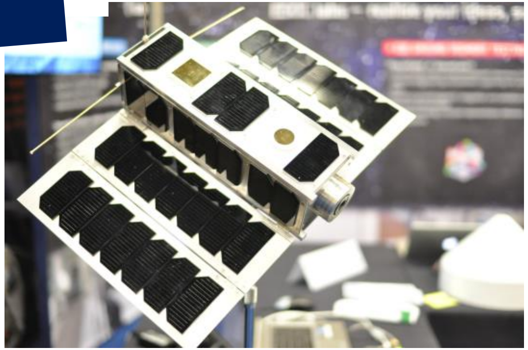
- Simulator OPS-SAT & HOPAS:
  - Dynamics parameters taken from the documentation
  - Wheel speed control loop
  - Time cycle
  - Others: sensors as white noise, delays, 3 axes decoupled, etc.
- Specifities of OPS-SAT:
  - Disturbance torque: coloured noise
  - Missed TMs

HOPAS on OPS-SAT

AIRBUS

From Simulator to OPS-SAT





HOPAS on OPS-SAT

AIRBUS

Test Plan

v1
- Skeleton application, does not include any AOCS software
- Goal : validate application execution and TM interfaces. First evaluation of I2C bus performance (ie interface between SEPP and iADCS)

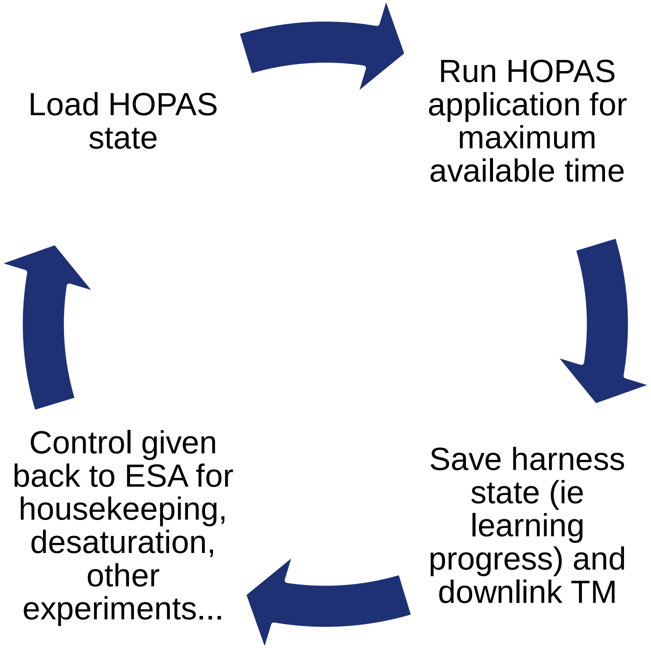
v2ff
- The AOCS code is added in the application and run a feedforward (ff) profile to test telecommands
- Validation of reaction wheel speed commands mapping and polarity

v2fb
- From this point, algorithm is running in close loop
- Validation and evaluation of the baseline controller

v3
- Execution of the application with the entirety of HOPAS
- Validation of proper execution for at least an entire generation, capacity to reload neural network parameters

AI
- Learning campaign executed on OPS-SAT
- Success if learning does occur and final performance is better than baseline (as measured in v2ff)

Quick note on sequenced training

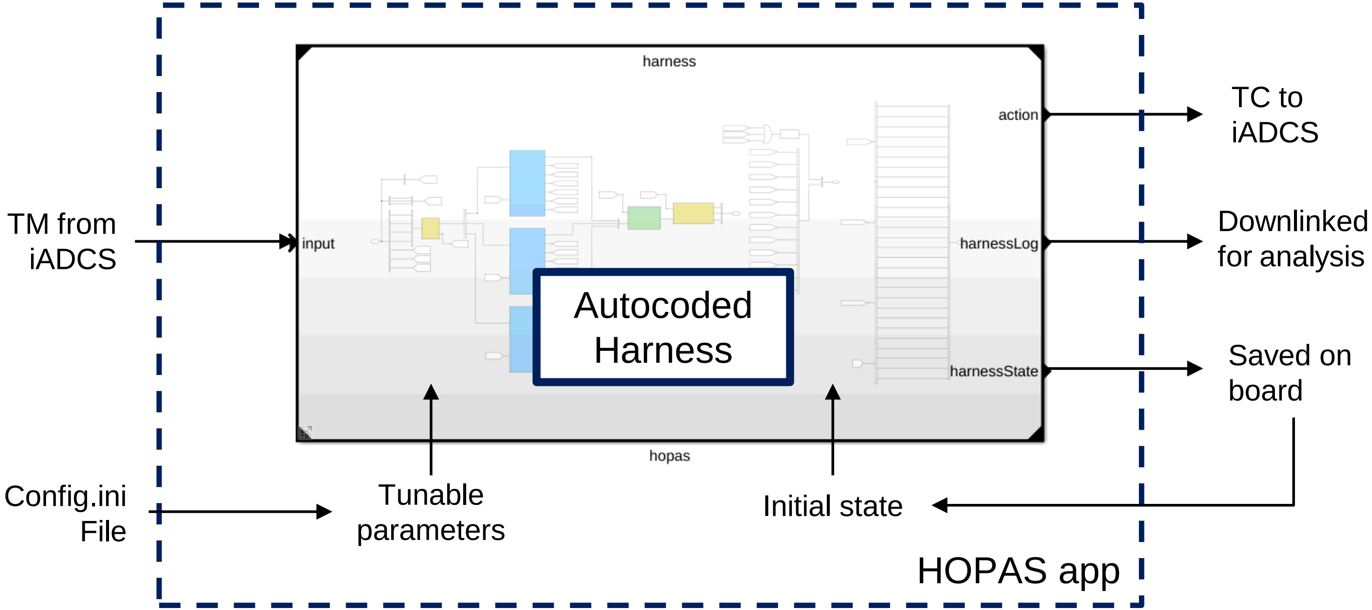


HOPAS on OPS-SAT

AIRBUS

Code generation approach

The chosen HOPAS design requires implementing a whole controller on the SEPP. This is a challenging task on its own, and it was decided to use the state of the art development workflow to be as efficient as possible.
This is achieved by using automatic code generation in MATLAB



AIRBUS

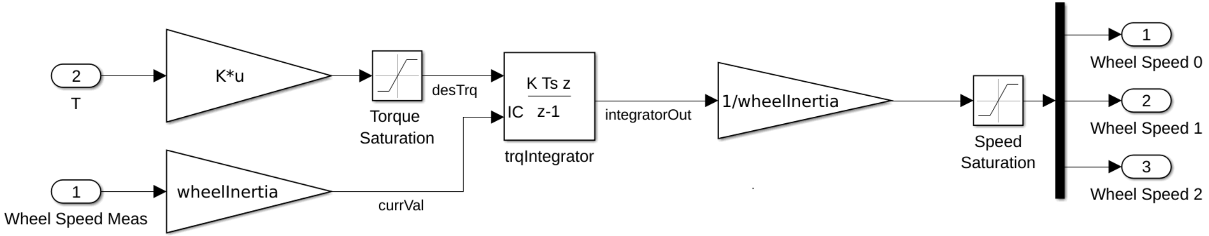
Interfaces and requirements

Requirements for the application from an ADCS perspective

- Harness needs to run at fixed frequency
- An input is expected at every timestep including quaternion an angular rates. The harness has been modified in the recent releases to add some security against failed/missing telemetry calls. See backup slide for me details
- The computed action needs to be passed to iADCS once computed
- The harness state needs to be reliably saved at the end of every generation and the app needs to be able to reload said state to enable sequenced training

Note : it was decided to send reaction wheel command speed. This adds two elements to the overall AOCS chain:

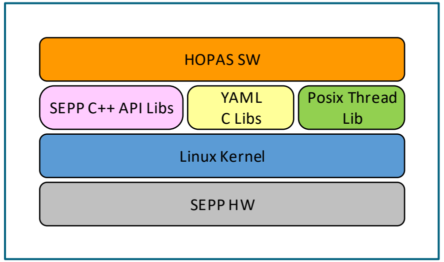
- A reaction wheel speed value needs to be obtained from iADCS at initialization
- Torque commands are converted to wheel speed commands within the harness



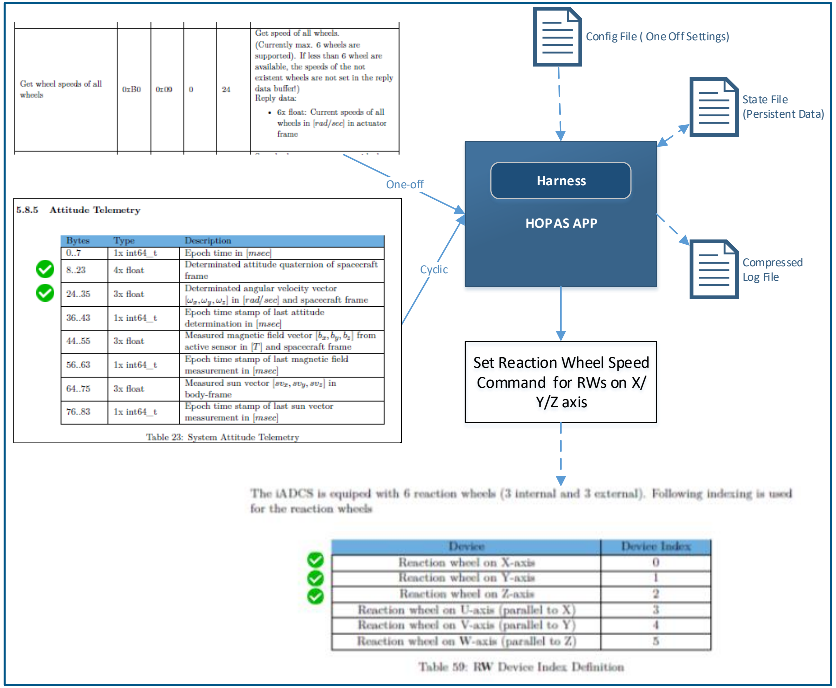
HOPAS on OPS-SAT

AIRBUS

Overall SW architecture



C/C++ Non-NMF application



Get wheel speeds of all wheels	0xB0	0x09	0	24	Get speed of all wheels. (Currently max. 6 wheels are supported). If less than 6 wheel are available, the speeds of the not existent wheels are not set in the reply data buffer!) Reply data: • 6x float: Current speeds of all wheels in [rad/sec] in actuator frame

5.8.5 Attitude Telemetry

Bytes	Type	Description
0..7	1x int64_t	Epoch time in [msec]
8..23	4x float	Determinated attitude quaternion of spacecraft frame
24..35	3x float	Determinated angular velocity vector $[\omega_x, \omega_y, \omega_z]$ in [rad/sec] and spacecraft frame
36..43	1x int64_t	Epoch time stamp of last attitude determination in [msec]
44..55	3x float	Measured magnetic field vector $[b_x, b_y, b_z]$ from active sensor in [T] and spacecraft frame
56..63	1x int64_t	Epoch time stamp of last magnetic field measurement in [msec]
64..75	3x float	Measured sun vector $[sv_x, sv_y, sv_z]$ in body-frame
76..83	1x int64_t	Epoch time stamp of last sun vector measurement in [msec]

Table 23: System Attitude Telemetry

The iADCS is equiped with 6 reaction wheels (3 internal and 3 external). Following indexing is used for the reaction wheels

Device	Device Index
Reaction wheel on X-axis	0
Reaction wheel on Y-axis	1
Reaction wheel on Z-axis	2
Reaction wheel on U-axis (parallel to X)	3
Reaction wheel on V-axis (parallel to Y)	4
Reaction wheel on W-axis (parallel to Z)	5

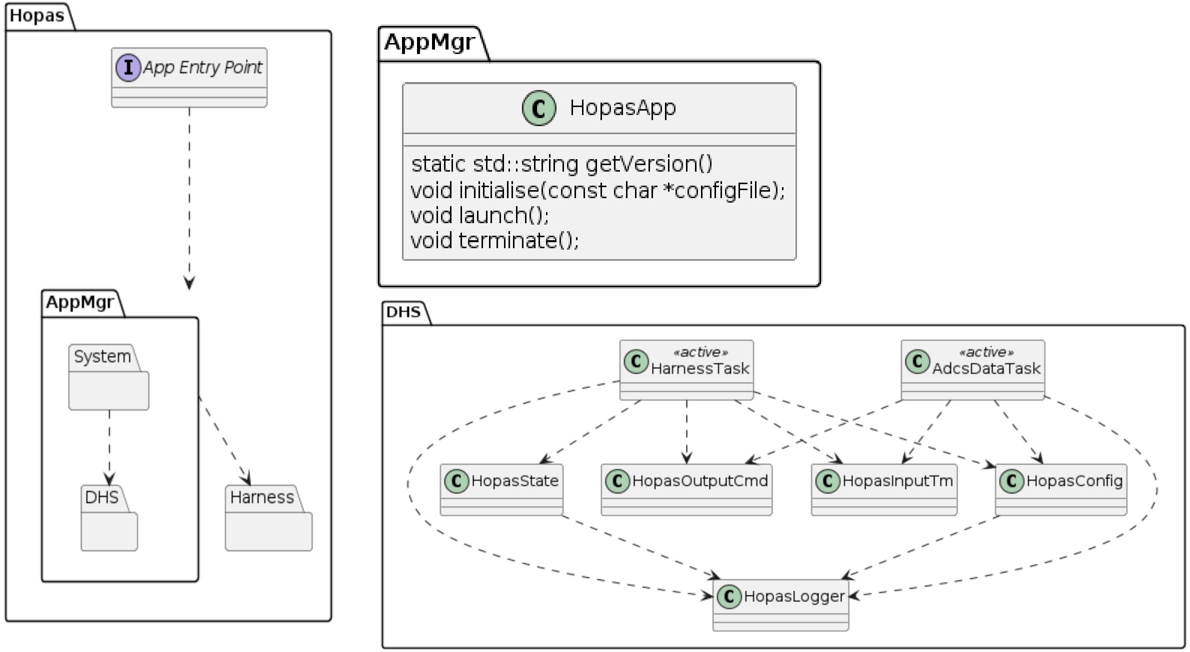
Table 59: RW Device Index Definition

HOPAS on OPS-SAT

- Context and introduction to HOPAS
- **Implementation on OPS-SAT**
- Flight Campaigns
- Conclusion and way forward

AIRBUS

Static SW Architecture



HOPAS on OPS-SAT

Context and introduction to HOPAS

Implementation on OPS-SAT

Flight Campaigns

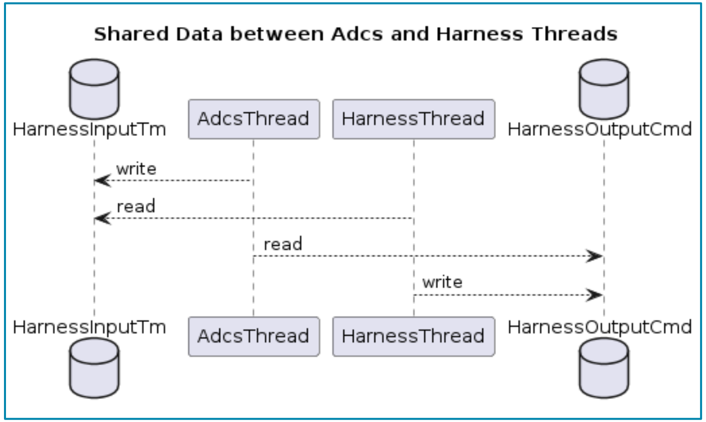
Conclusion and way forward

AIRBUS

Dynamic SW Architecture

- Multi-threaded design to overcome slower and varying rates of acquisition of input Attitude TM
  - Adcs Thread
  - Harness Thread
  - Main Thread

- Protected Shared Data



On-board Test Campaigns Performed

Release Date	Version	EM	FM	EM Test Date	FM Test Date
2022-06-01	v1.0.0	X	-	2022-06-02	
2022-06-07	v1.1.0	X	-	2022-06-07	
2022-06-27	v1.2.0	X	-	2022-06-29	
2022-07-01	v1.3.0	X	-	2022-07-05	
2022-07-27	v1.4.0	X	X	2022-07-28	2022-09-11
2022-10-25	v1.5.0	X	X	2022-10-25	2022-10-28 2022-11-09
2022-10-07	v2.0.0	X	-	2022-11-07	
2022-11-08	v2.1.0	X	-	2022-11-09	
2022-11-09	v2.2.0	X	-	2022-11-09	
2022-11-09	v2.2.1	X	X	2022-11-10	2022-11-25 2022-11-26 2022-11-27
2022-11-30	v2.3.0	-	-	TBD	TBD

HOPAS on OPS-SAT

AIRBUS

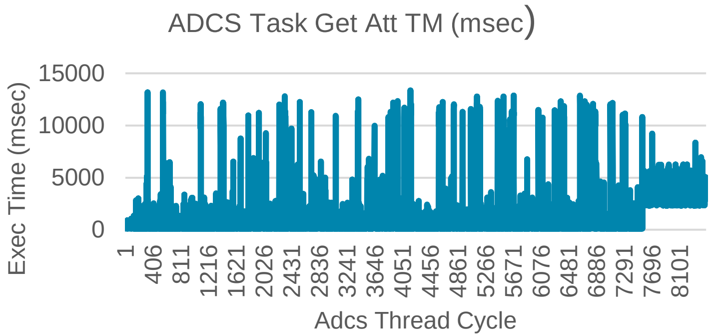
EM tests

- Limitations
  - EM iAdcs Unit not a fully representative of the FM unit
    - No RWs attached to iADCS unit
    - Invalid attitude quaternions received
    - Degraded I2C Bus
- Achievements
  - Demonstrate V1.5.0
    - Programmatic I/F with iADCS unit over SEPP iADCS API
    - Periodic retrieval of the expected TM from iADCS unit over I2C bus
    - Self-terminating application at the end of experiment
    - Compressed Log file size < 10 MiB
  - Demonstrate V2.2.1
    - Same as V1.5.0
  - Measurement of execution time of SEPP iADCS API used cyclically
- Deferred Goals
  - Running Harness baseline controller

AIRBUS

V1 Tests

- V1.4.0 and V1.5.0 tested on FM
- V1.5.0 Experiment Duration : 3 hours

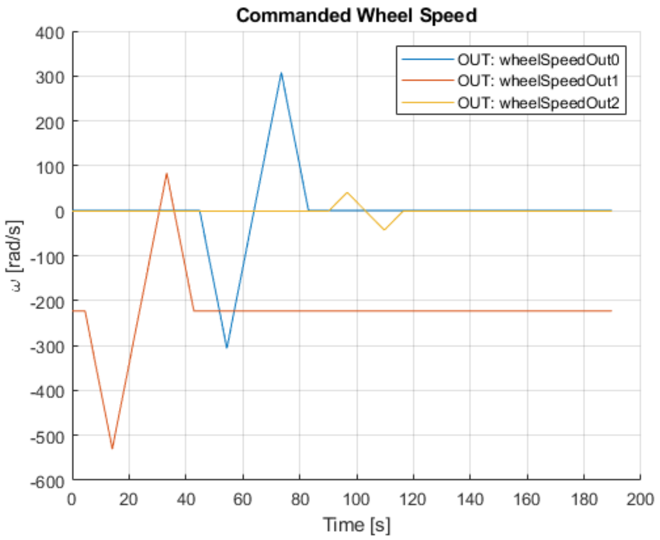


- I2C Bus Performance (Fetch Attitude TM) → Missed TM
  - Average ET(execution time) ~ 1 sec
  - Maximum ET 13.35 sec
  - ET => 72% < 500 millisecond where 16.65% > 2 second
  - 12.64% of the time, SEPP iADCS API failure
  - I2C Bus in FAILED state permanently for the last hour
- Potential Workaround:
  - ↑ Cycle time : 0.5 sec => 1 sec

AIRBUS

V2 test analysis

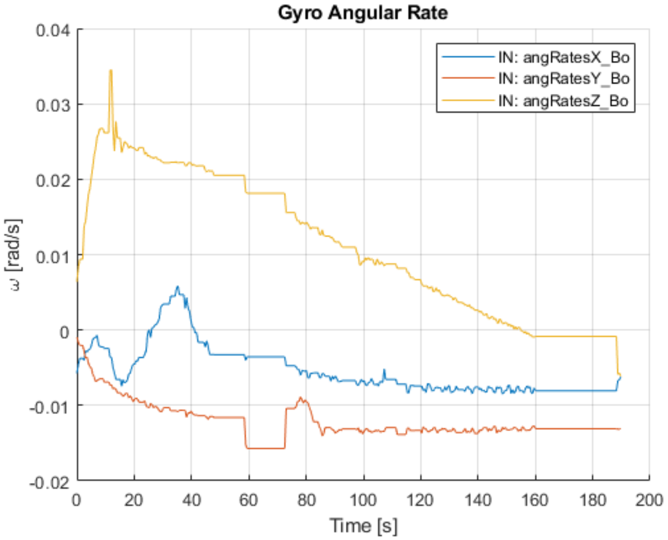
- Open loops test exp172_20221126_1737 performed on the FM
- Sends open loop wheel speed commands to slew OPSSAT about each body axis sequentially
- Expected behavior: slew about x, y then z axis
- Smaller wheel speed command profile used for z-axis slew was used to account for lower z inertia



AIRBUS

V2 test analysis

- Body angular rates shown (gyro)
- Quality of test results not ideal (not ‘clean’)
- Order of slews about x, y then z axis appears to be successful but full confirmation is not possible
- X-axis appears to show most prominent evidence of a slew
  - Slew polarity appears to be reversed w.r.t. expected profile
  - Further quaternion-level assessment will be done to rule out sensor-level polarity error
- Data missing for y-axis slew
- Z-axis slew barely noticable
- Unexplained rate drift observed when not slewing → could be gyro effects or RW spin down
- Additional open loop test is planned to give clearer result and confirm polarity error



HOPAS on OPS-SAT

AIRBUS

Summary

Development progress

- Application structure designed and implemented
- Upgraded to multi threaded design

v1

- Feedforward profile designed and tested in simulator
- Harness encapsulated in application

v2ff

- Close loop controller retuned to latest I2C performance results
- Implemented in application

v2fb

- Autocoded and implemented in application with state reloading
- Retune needed pending v2fb results

v3

- Performance campaigns run in simulator with sequenced training and I2C behavior modelled

AI

In orbit tests

v1

- Run in flight, both single threaded and multithreaded versions

v2ff

- Run in flight
- To be run again with torque commands and lower frequency

v2fb

- Ready to fly once v2ff is ready (same software as v2ff with different configuration)

v3

- Pending

AI

- Removed from project scope

Conclusions and way forward

- **Main achievements:**
  - Execution of the autocoded harness on board of OPS-SAT
  - Successful interface with iADCS
  - Better characterization of the I2C performance
  - Ready to fly HOPAS algorithm
- **Immediate next steps – ideally in the coming weeks depending on OPS-SAT availability**
  - V2ff tests using torque commands, reworked profile and lower frequency
  - V2fb tests
  - V3 tests
- **Long term**
  - Sequenced AI performance campaign
  - Follow up study next year to better qualify algorithm?

Thank you to the whole OPS-SAT team for their support!

AIRBUS

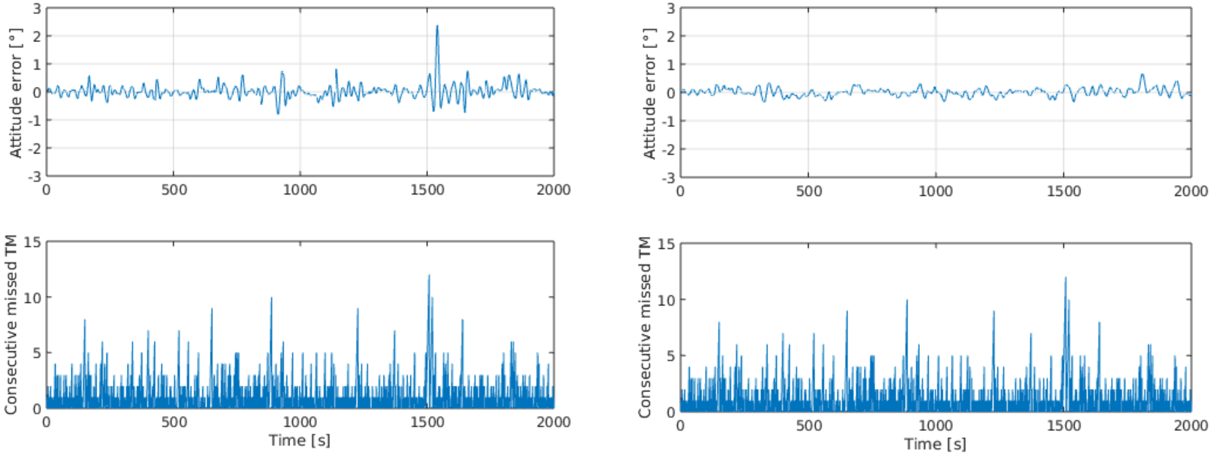
Thank you

AIRBUS

BACKUP : Missed TM management

Early on it the tests it was observed that for some time cycles, a new TM value is not obtained. This may lead to two issues:

- A torque command being integrated over multiple time steps by wheel speed conversion. This is mitigated by forcing torque to 0 when no new TM was received.
- Loss of comparability between learning episodes. This is mitigated by setting reward to 0 on such timesteps and normalizing episode score by number of valid timesteps



Visualisation of the effect of the missed TM mitigation solution

On the left, no mitigation is used. Transients are observed when the numerous consecutive missed TM events occur (for instance around 1500s, when a maximum error of over 2 degrees is seen). On the other hand, once the mitigation solution described above is implemented (see right), the transient behaviour is significantly less pronounced.