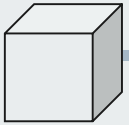
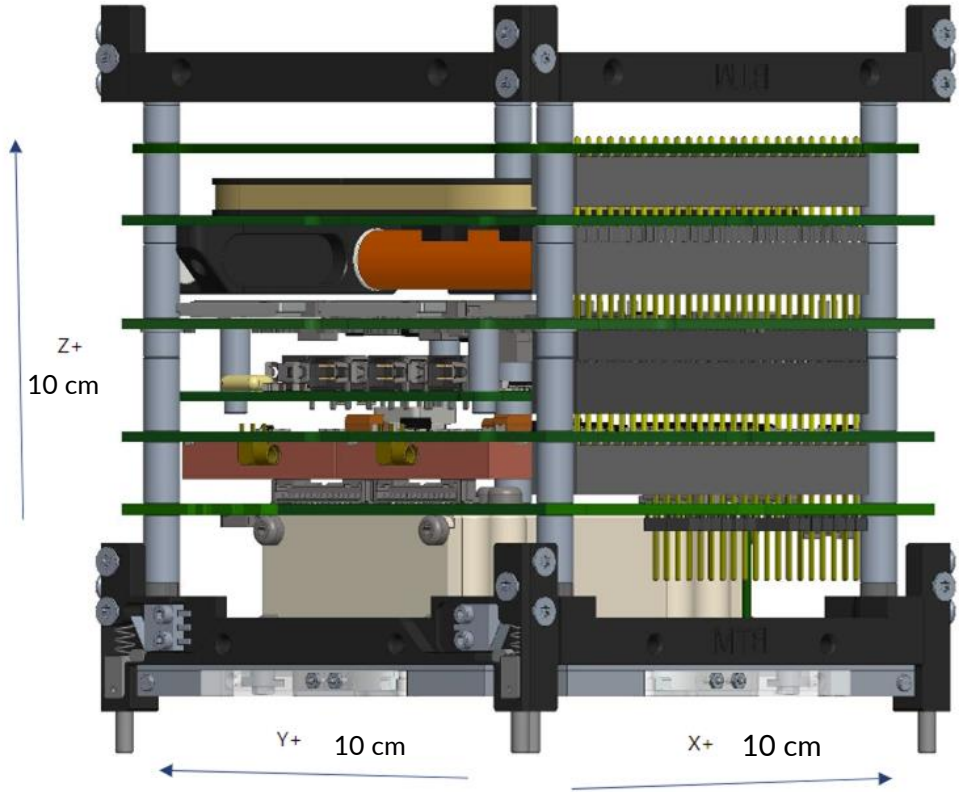
Developing Satellite Software

Jose Pastrana, Anas Matar, Marouf Paul, Zheng Yu Wong

Mission



- *RHOK-Sat is a 1U CubeSat*
- *Characterize the performance and degradation of perovskite cells*
- *Develop a fault tolerant software*



payload

3-axis magnetorquer

on-board computer

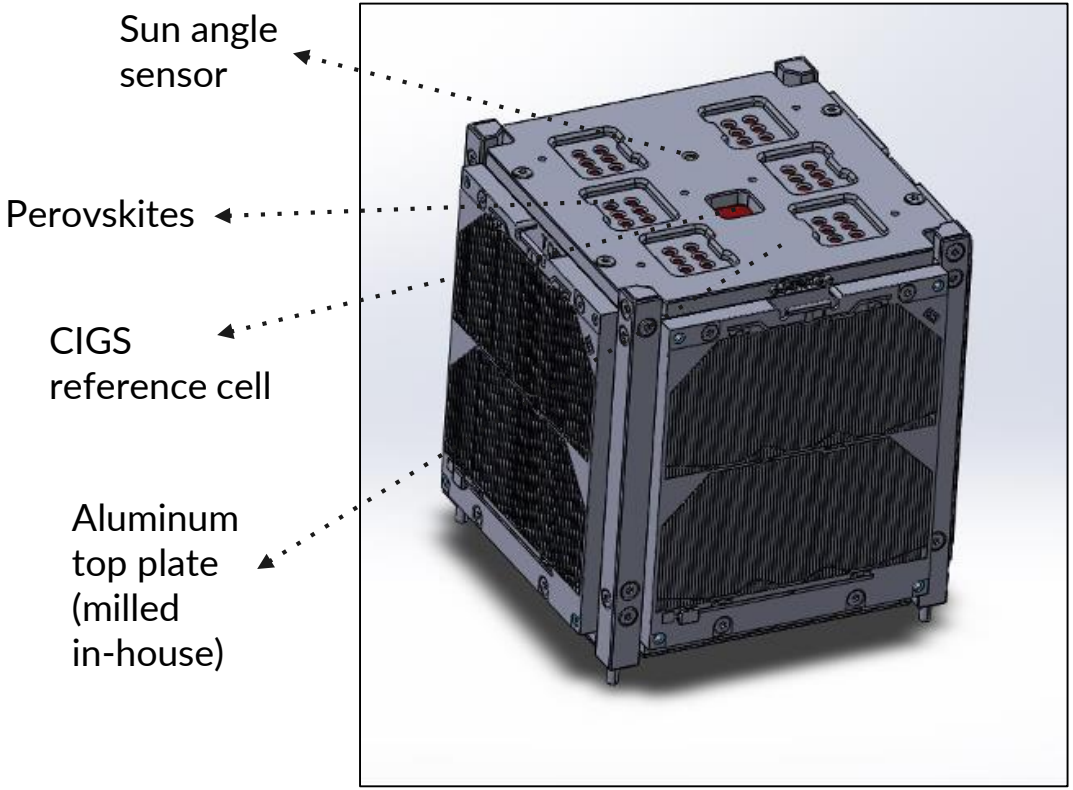
transceiver

electrical power system

antenna

Satellite Subsystems

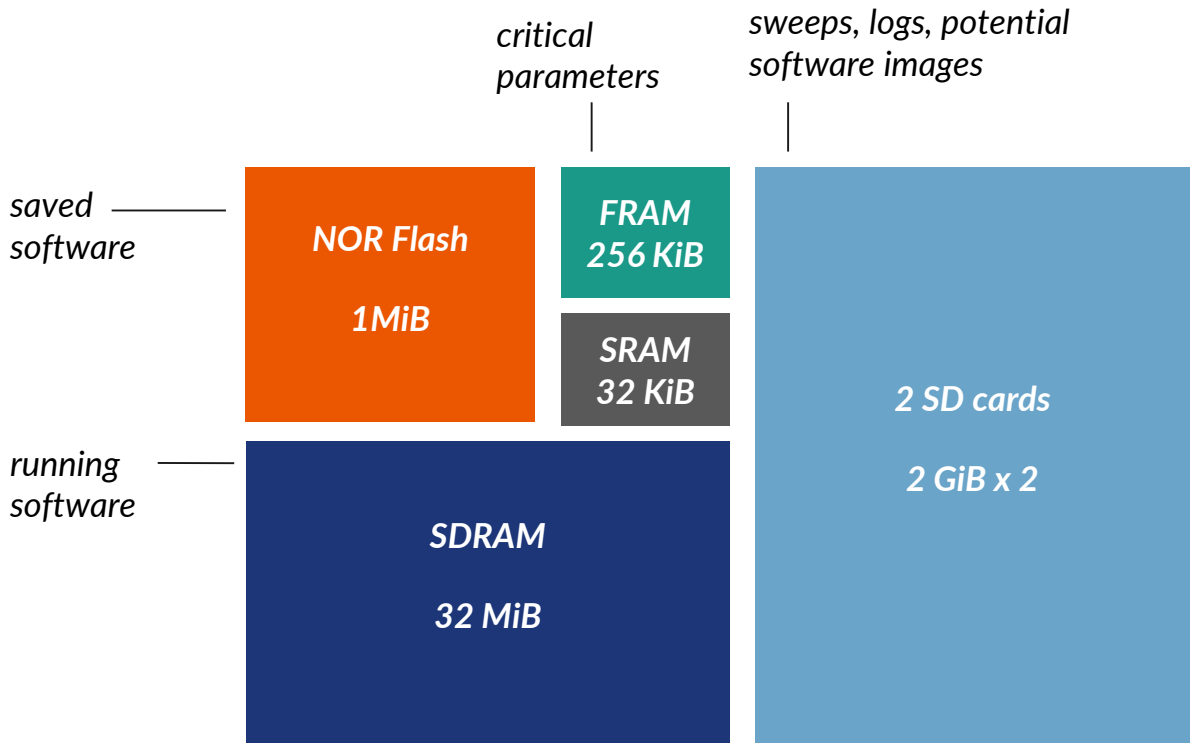
Payload: Top Plate



Hardware Constraints

- *Memory constraints*
- *Power budget constraints*
- *Experimental requirements*
- *Data budget constraints*

Memory Constraints



Chosen based on:

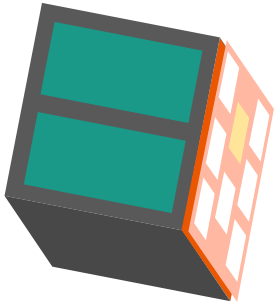
Volatile and non-volatile

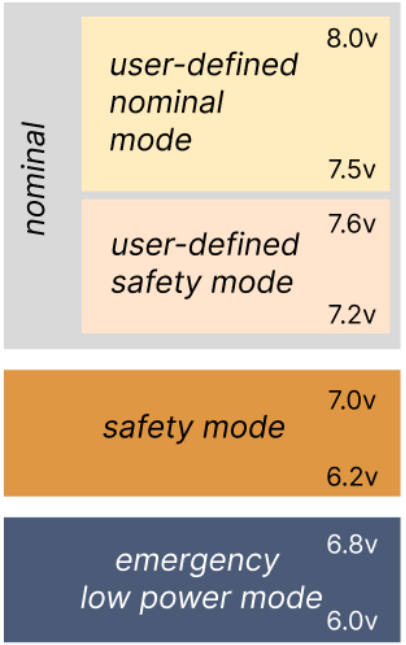
Radiation hardiness

Space

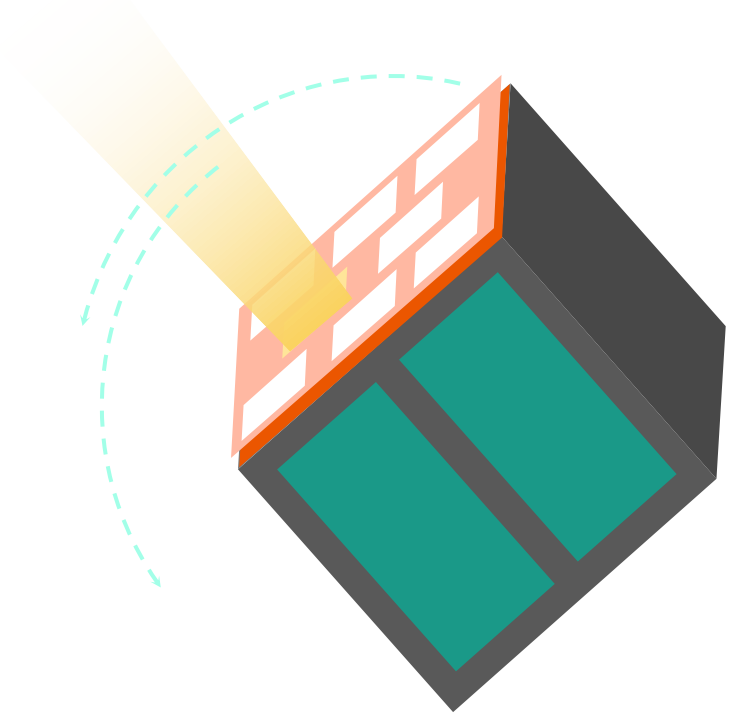
Power Levels

- Power levels are monitored in a cyclic fashion
- Need to guarantee enough power to complete the upcoming task
- Low power mode interruption midway through processes can corrupt data and generate spurious errors



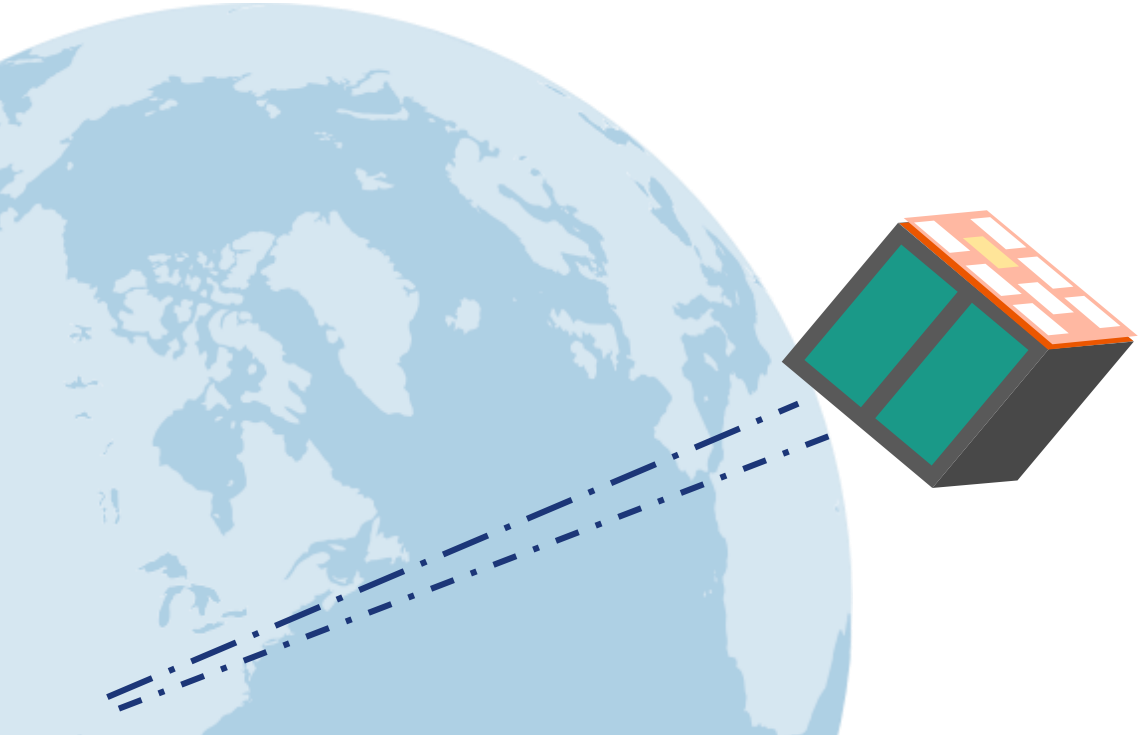


Experimental Requirement



- Solar cells must be held at near constant illumination to be measured accurately
- Reduce tumbling rate to 1°/s overall
- Sun angle must be within 35°
- Periodic check to determine whether to take a sweep
- Measure temperature before and after sweep

Communication



- The satellite is capable of performing 42 measurement procedures (sweeps) per orbit
- Additional logs and diagnostic files
- Only capable of transmitting 28 sweeps per pass over our ground station

Radio

- Using the amateur radio bands
- Implementing a **transponder** over the weekends to contribute to the ham radio community

- SatNOGS helps gather data through participating ground stations
- In return, we built a dedicated ground station that is always online for open use

Software Decisions

Minimal dependencies

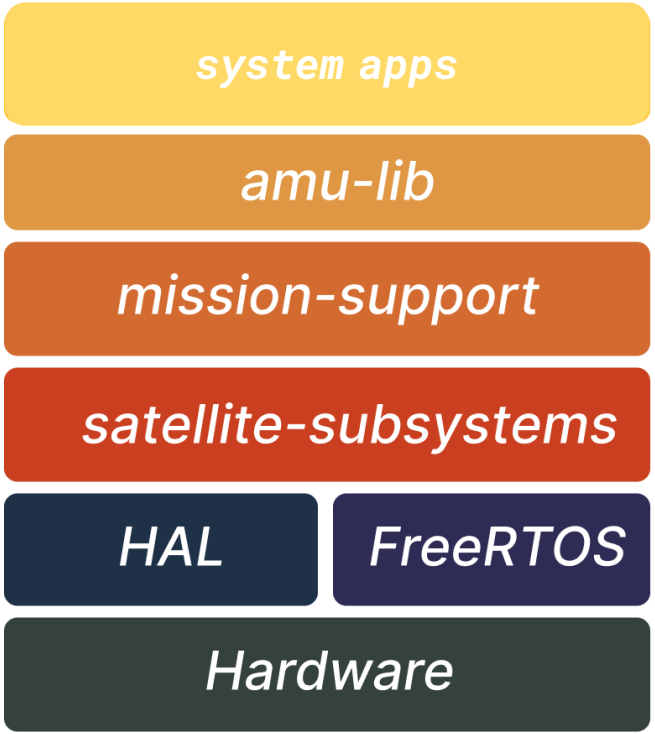
Static memory

Cooperative multitasking

Cyclic execution pipeline

Architecture

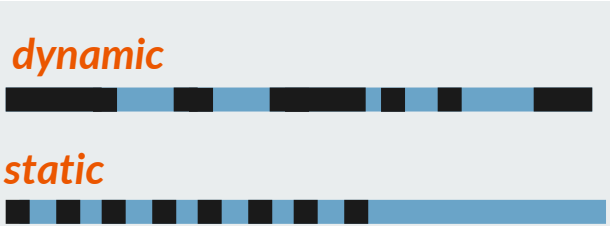
- 3rd party software
- Keeping software size small is critical
- Only for accessing low level hardware
  - Hardware abstraction layers
  - FreeRTOS
  - AMU library



Error mitigation

Statically allocated memory

- Prevents memory fragmentation bugs
- Frees unused memory

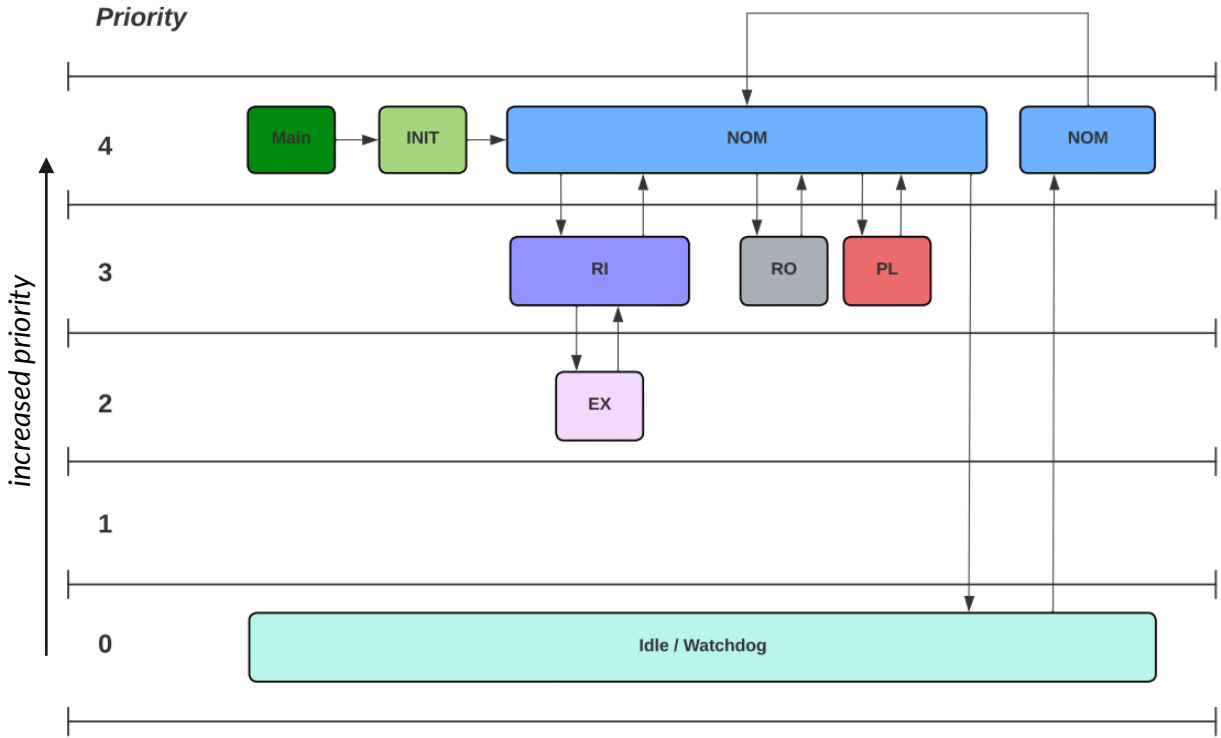


Cooperative scheduling

- Prevents collisions and conflicts between tasks
- Predeterministic
- Easier to implement and debug
- Leads to a cyclic execution pipeline

Cyclic Execution Pipeline

- Allows periodic health checks
- Ensures the experiment is run appropriately
- Avoids deadlocks and task starvation



Fault Tolerance

Commands

Direct control over the subsystems from the ground station

Ability to alter the configuration and parameters of the satellite

Update

Handling runtime bugs

Testing

Rigorous unit and integration tests

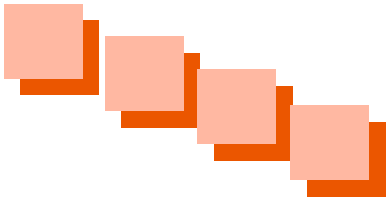
Logs

Save events for future diagnosis

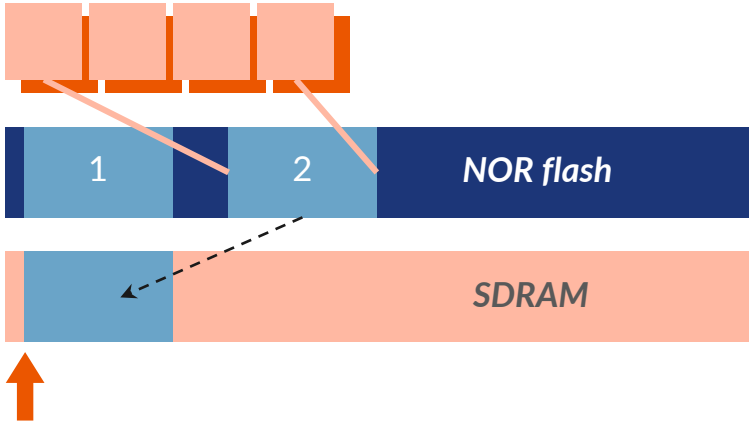
Updating Software

Low bitrate data transfer

Software uploaded in **packets**



After stitching it's checked with **cyclic redundancy checks**



Update process

- Packages are stored on the stack
- Written to NOR flash next to the 1st image
- Copied from NOR flash onto SDRAM
- Image selection is determined by parameter in FRAM

Bootloader

- Cornerstone of safe software updates
  - Can’t just overwrite the only program we have. What if it accidentally fails midway?
  - Write elsewhere (in NOR flash or SD) and boot from there next time
  - Require a second-stage bootloader
- Ultimate fail-safe
  - Comes at the cost of long transfer and writing times
- Everyone recommends it
  - No one tells you how to do it!

Developing Satellite Software

Jose Pastrana, Anas Matar, Marouf Paul, Zheng Yu Wong