# F Prime to Flight Faster

## Hardware-in-the-Loop Continuous Integration for Accelerated CubeSat Development

**Nate Gay[1], Saidi Adams[1], Michael Pham[2]**

[1]Texas State University · [2]Open Source Space Foundation

## The Problem

*Integration is where flight software teams slow down*

Flight software teams iterate quickly in development and then slow down at integration, because the software must be validated against real hardware that is itself still changing.

Bench testing is manual, intermittent, and happens late. Defects surface days after the commit that caused them, once the context is gone and the change has been built on.

> **Our claim:** Automated testing on real satellite hardware, run on every commit, verifies correctness before merge and lets reviewers judge pending changes on design rather than on whether the code works, increasing team throughput.

**TERMINOLOGY**

**Check**: a single step (lint, unit test, build, etc.) in a CI run.
**Commit**: one saved change to the software.
**Continuous Integration (CI)**: builds and tests every change.
**F Prime (F´)**: NASA JPL's open-source flight software framework.
**GNU debugger (GDB)**: loads and runs code on the satellite.
**Hardware-in-the-Loop (HIL)**: tests that run on satellite hardware.
**PROVES**: an open-source CubeSat kit and program.
**Pull request (PR)**: a proposed change to the software.
**Serial Wire Debug (SWD)**: a direct path to program the satellite.
**YAMCS**: open-source ground software for spacecraft.

## Our Challenge

*The hardware moved while the software was being written*

Over one year the PROVES Kit went through three major and four minor board revisions, plus two revisions of the solar face boards. Every one of them changed something the flight software depended on: pin assignments, device enumeration, power sequencing, which peripherals were even present.

| 9 | 23 | 1,917 | 212 |
|---|---|---|---|
| board revisions | contributors | commits | pull requests |

A team this size, at this rate, cannot re-qualify a satellite by hand every time either side changes. Hardware testing must be automated.

> **Verified every run:** commanding · telemetry · eventing · IMU · thermal management · antenna deployment · real-time clock · filesystem · power management · hardware watchdog

## What Made the Difference

**1. Deterministic hardware state.**
Reformat storage, cycle power, flash over an independent path, address the board by USB ID. Most flakiness was state, not code.

**2. Self-hosted build runner.**
No cache on free cloud runners meant a full submodule re-clone every build. Self-hosted runner saved about 10 minutes per run.

**3. Skeleton cube on standoffs.**
Swapping a part went from disassembling a satellite to unscrewing 4 standoffs. Maintainability of the rig *is* CI uptime.

**4. Flight-like communications and ground software.**
Adding radio tests alongside existing UART tests caught a failure class.

**5. Not every test needs hardware.**
Attitude math, time handling, and frame parsing were pulled out of flight components and unit-tested in the cloud in seconds.

**6. Shift failures earlier in the CI run.**
Encode known hardware failure modes as static checks. A config setting that broke test downlink now fails the build, not HIL.

## Results

*1,917 commits · Aug 2025 – July 2026*

| 132 h | 18 min |
|---|---|
| of hardware test runtime | median commit to HIL verdict |
| **10** | **100%** |
| flight-critical subsystems per run | of merges hardware-validated |

### Hardware testing dominates the pipeline

solid bar = median, pale bar = 90th percentile

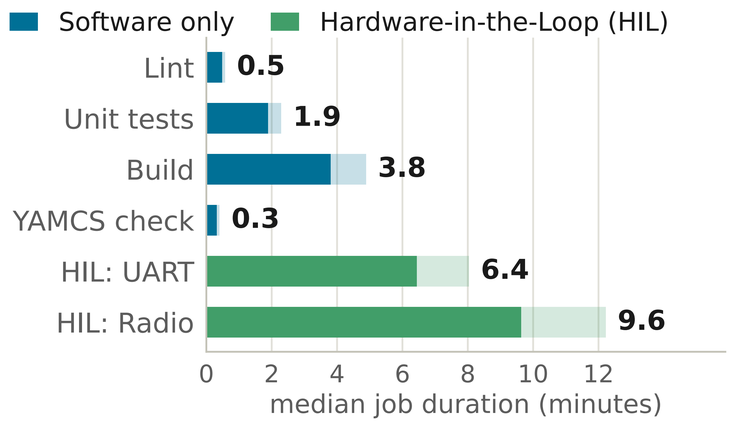



Figure 1. HIL time is 70% of the critical path.

### Testing got more reliable as it scaled

1,559 HIL jobs · Oct 2025 – Jul 2026

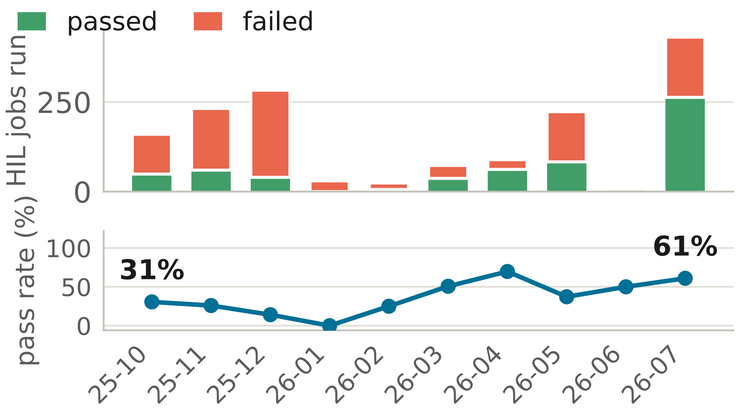



Figure 2. Pass rate rose 31% → 61% while monthly volume tripled.

### Hardware feedback in minutes, not days

317 runs · 2026-05-19 – 2026-08-01

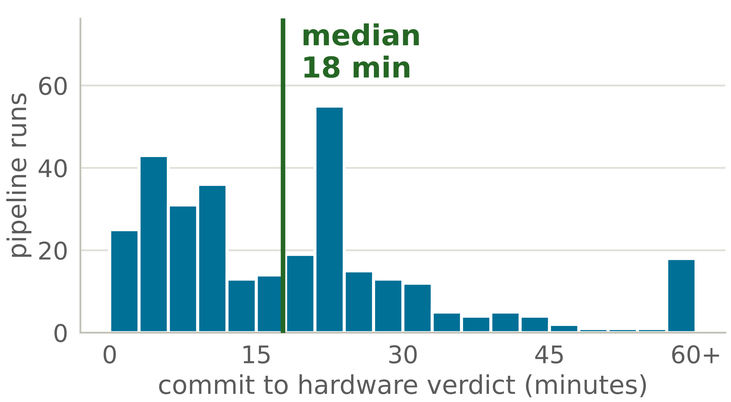
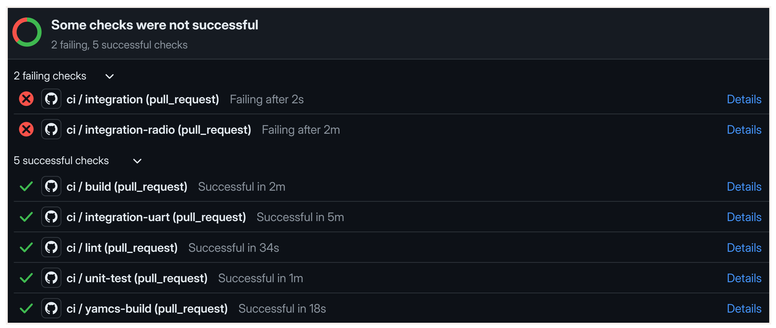



Figure 3. 90% of commits get a HIL verdict within 39 minutes.

## What Broke, and What Fixed It

*Lessons other teams can reuse*

| Symptom | Root cause | Mitigation |
|---|---|---|
| Board unreachable after a bad flash | Reprogramming depended on the software under test | Program satellite hardware over SWD with GDB |
| Watchdog resets flash | Hardware watchdog fired during long software loads | Programmable power supply; sequence power with the load |
| Passes locally, fails in CI | Residual SD card state between runs | Reformat the SD card before each run to remove leftover state |
| Same test, different result per site | Lab-to-lab hardware differences | Tag each test with the hardware it needs; a runner executes only the tests it can support |
| Intermittent, unreproducible failures | Various | Log and archive all telemetry to correlate failures after runs |




CI as a developer sees it: the integration checks run on real hardware, and a failing check blocks the merge.

## Future Work

- **Flat bench layout** replacing the cube form factor, so parts swap without rebuilding a structure
- **Backplane** instead of hand-built per-component wire harnesses
- **Reproducible CI runner setup** via Nix, a bootable image, or Ansible
- **CI run timeouts and queueing** so one hung board or runner cannot monopolize the single physical runner
- **Continue addressing flaky tests** to build trust in the pipeline and keep results reliable

## Build This Yourself

Every piece of hardware and software described here is **open source.**

**Scan QR code** for links to get started as well as a digital copy of this poster and author contact information.

**Acknowledgments.** Dr. Blagoy Rangelov, Evan Jellison, Ines Khouider, Texas State University Department of Physics, Cal Poly Pomona, Open Source Space Foundation, NASA Jet Propulsion Laboratory, and NASA CubeSat Launch Initiative.