Department of Computer Science
Texas State University

April 23rd, 2025
CS 4395.251

Texas State's First Satellite Mission: An Independent Study in Pleiades Maia's Software Development

Author

Taylor Gilg

vzu3@txstate.edu

Independent Study Advisor

Dr. Mylene Queiroz de Farias

mylene@txstate.edu

Abstract

I joined Texas State University's Space Lab software development team in September 2024 for the university's first satellite mission, Pleiades Maia. Pleiades is an open-source, collaborative project between six universities that aims to construct, communicate with, and control small, Earth-orbiting satellites with more accessible and cost-effective embedded measuring devices. The long-term goal is to combine extensive documentation, software tools, and small satellite engineering materials into an educational kit. We hope that our contributions to the Pleiades project will help to cultivate an educational tool and community that enables high schools and colleges without established space labs to gain hands-on experience with space mission development, satellite development, and a finished, space-ready satellite in a matter of months.

I have been working alongside six other undergraduate students to build upon and enhance a Python-based open-source flight software called PROVES (Pleiades Rapid Orbital Verification Experimental System) Kit CircuitPy. Due to the collaborative and exploratory nature of the project, our tasks have been incredibly varied. Our team's contributions to the CircuitPy ProvesKit software include: infrastructure and tooling development, advancing of architecture and code quality, testing, firmware and embedded functionality development, programmatic shifts and strategic decision making, ground systems and communications, and project outreach.

My major contributions this semester have been introducing PyTest unit tests to the code base to validate software functionalities, refactoring non-volatile memory (NVM) usage for storage of important flags beyond satellite reset, and readapting old SD Card functionality code into our current repo for file system development [10]. My next tasks will be finishing generating thorough and expansive documentation for our code base, and designing live data presentations for our developing ground station software.

1 Introduction

Pleiades 5 is currently one of the most active, open source small satellite projects in the research space [5]. The collaborative and educational nature of the mission has informed the choice to make the design not only straightforward and easy to work with, but as modular and customizable as possible [6]. While we are developing the software and hardware foundation that will be shipped on every kit, the payload, various measuring devices and sensors, will be able to be easily switched out and replaced, making missions with the kit endlessly customizable to different research questions and able to claim valuable novelty, something really important when it comes to landing grants, research opportunities, and further experiences in the field and industry.

With the help of the Pleiades project and similar collaborations of its kind, the construction cost of small satellite CubeSats has been reduced from approximately $50,000 to around $3,000 on the Pleiades project, although the goal is to go even lower to $1000 [6,3]. Despite construction not being the largest cost involved in a satellite mission– that would be the launch provider and environmental testing – Pleiades 5's launches and preliminary tests will be covered by NASA's CubeSat Launch Initiative (CSLI) [4].

Pleiades Maia's launch will have Texas State's satellite intercept with the International Space Station (ISS). Here, the astronauts on board will eject the craft from a specialized launcher built into the space station to deliver the satellite to its final destination in low earth orbit [11,3]. Pleiades Maia's satellite is a 10x10x10cm cube called a CubeSat. For its size, it is referred to as a 1U, larger satellites being multiple U large [3]. Inside the aluminum frame sits a flight controller board, battery board, and payload of sensors and measurement devices. The satellite's antenna attaches to the external flight board and each side is made up of solar panel boards, one side also including a camera [6].

Texas State University's satellite mission, Pleiades Maia, is specifically focused on using more accessible, cost effective embedded measurement devices to control the movement of a satellite. The goal is to be able to change the attitude, or where a particular face on the satellite is facing, with only a gyroscope, magnetometer, light sensors, and a magnetorquer. Two of the most commonly used solutions to determine a satellite's orientation and location in space are both incredibly expensive monetarily and resource wise. The solutions include: taking a picture of either the earth or nearby stars and using an intensive algorithm to calculate where the craft is, or have the accessibility to communicate with a network of other satellites in order to figure out a craft's spacial data. Using the rotational speed data from a gyroscope, the directional data of the magnetic field of the earth and the craft via the magnetometer, the direction of the sun via light sensors, and being able to apply force onto the satellite via the magnetorquers, we wish to direct the motion of and change the attitude of the spacecraft with more accessible, cost effective embedded measurement devices than the current alternatives.

2 Approach/ Methodology

The Texas State University software development team has significantly advanced the Proves Kit CircuitPy code base by implementing software infrastructure tools, enhancing code architecture, introducing software testing, initiating ground systems development, engaging in community outreach, and making strategic technical decisions.

In early December of 2024, the Texas State University team switched from developing a version of the Proves Kit with C++ based open source flight software, F Prime (F'). The software lacked a significant development community or much support past its documentation so the team decided to join Pleiades' collaborators in further advancing the other software version of the Proves Kit using python based open source software CircuitPy.

2.1 Software Infrastructure and Tooling

At the time, the code base lacked a solid foundation for code development, building, and deployment. Seeking to streamline the development and user experience, the Texas State team integrated various new tools including continuous integration (CI) pipelines for automated testing on code changes, automated code reformatting with pre-commit hooks, introduced Intellisense for CircuitPython code, created a pull request template for organization of contributions, locked

dependencies for consistency, and streamlined setup of development environment and dependencies via Make tools.

2.2 Architecture and Code Quality Improvements

With the introduction of new development infrastructure, the team moved to improve the code base's structure, maintainability, and robustness. Evaluation of existing functionality implementation was soon conducted and extensive code cleanup was performed to remove inefficiencies and evaluate areas in need of refactoring. Serviceability and durability was increased by reducing cyclomatic complexity (decreasing number of nested code paths), adding comprehensive type-hinting, and adopting a dependency injection pattern (separating service construction from applications of use) into the code base's design. To document critical design decisions and manage hardware initialization failures systematically, an Architectural Decision Record (ADR) has recently been introduced. Additionally, variable management has become more flexible through the creation of a centralized configuration file, replacing previously hardcoded satellite parameters and enabling easier adjustments for mission-specific needs. Taking from the team's time learning about F Prime's structured approach to hardware abstraction, a similar strategy has been adopted to enhance hardware accessibility. The recent release of flight controller board version five in March 2025, has accelerated modularization of the code base: separating development spaces to maintain simultaneous support of versions four and five and isolating the customizable payload layer for end-users.

2.3 Testing, Documentation, and Quality Assurance

To ensure the software functions reliably and as intended, the team has implemented comprehensive testing and quality assurance practices. Pytest unit testing has been introduced across the code base, currently achieving approximately 50% code coverage. Additionally, to strengthen system dependability, validation checks for expected ranges and data types have been recently integrated into the satellite's configuration variables. Robust software monitoring and debugging capabilities are supported by a structured logger, which categorizes runtime events and errors by severity level and provides clear, readable, color-coded alerts significantly improving issue detection and resolution processes. Finally, comprehensive software documentation generation has been initiated to facilitate effective onboarding and maintain continuity in future development efforts.

2.4 Firmware and Embedded Functionalities

The team made several hands-on contributions to on-board firmware and device-level programming this semester. Multiple essential firmware features were reintroduced and expanded including SD card functionalities for file system development, asyncio for concurrency and task scheduling, and bitflag storage and manipulation in non-volatile memory (NVM). To expedite development cycles and facilitate more efficient testing workflows, streamlined procedures were implemented for rapidly setting board time configuration and quickly onboarding firmware onto flight controller boards. These advancements significantly improved the efficiency and practicality of the team's embedded software development processes. Additionally, the V.4 flight controller board PROVES Kit firmware was recently officially adopted into upstream CircuitPython, making the team's improvements broadly accessible within the open-source community [1].

2.5 Ground Systems and Communications

Most recently, the team has begun foundational work on ground station software for the Pleiades project and Texas State University's future space missions. The first steps of refactoring radio communication tests between boards is now underway. Additionally, many team members, including myself, have completed certification for amateur radio licenses to operate in the specific frequency bands the satellite will communicate in.

2.6 Outreach and Team Building

Beyond technical contributions, the team has accomplished significant work in outreach, culture building, and team management. Team identity was reinforced through the creation of the Pleiades Maia mission patch and by sharing the project's progress at the PyTexas conference in April 2025. Additionally, team cohesion and knowledge transfer were enhanced by conducting technical interviews, onboarding new members effectively, and facilitating weekly demonstrations showcasing both individual accomplishments and collective team milestones. These consistent interactions have fostered an inclusive and collaborative environment, strengthening the team's overall effectiveness and sense of community.

2.7 Personal Contributions

2.7.1 Introducing Pytest Unit Testing to Repo

My development journey for Proves Kit CircuitPy began back in December 2024 by introducing the first software tests to the repository. At the time, I was still familiarizing myself with the code base and python testing conventions. I began by identifying detumbling functionality in the satellite functions.py file (in the pysquared directory) as a strong candidate for initial testing, due to its minimal direct hardware interaction. Originally, I used Python's built-in unittest library, wanting to avoid adding external dependencies. However, following feedback from our team lead, Nate Gay, I transitioned to Pytest for its efficiency and advanced features:

- Faster execution and simpler syntax
- Support for fixtures that streamline setup and teardown of test environments
- Enables parallel and concurrent test execution, improving feedback cycles

When a satellite is released into its final orbit, it typically begins spinning or “tumbling” out of control due to forces exerted during deployment. Before normal operations can begin, this rotational motion needs to be stabilized, a process known as "detumbling." During detumbling, measurements of magnetic field strengths and angular velocities along the satellite’s x, y, and z axes (obtained using onboard magnetometers and gyroscopes respectively) are utilized to determine the required force to reduce the craft’s angular velocity to near zero. Magnetic torque coils, known as magnetorquers, are used to generate controlled magnetic fields that interact with Earth's magnetic field and apply precise torque to stabilize the satellite’s orientation.

How do_dotumble() works: (previously in functions.py of pysquared)

- Collects real time sensor data of the x, y, z axes from the Inertial Measurement Unit’s (IMU) magnetometer and gyroscope.
- Angular velocity values less than 0.01 are set to 0 to avoid calculations based on noise.
- The magnetic dipole moment (the required strength and orientation of magnetic torque) is then calculated by calling detumble.py’s magnetorquer_dipole() (takes in a list -data[]- storing the magnetometer -data[1]- and gyroscope -data[0]- tuples).
- In detumble.py, magnetorquer_dipole() calls upon dot_product() to multiply tuples (in this case just squaring the magnetic field data) and x_product() to produce a cross product of tuples (magnetic field and angular velocity) to make its calculation.
- Do_detumble then uses this calculation to inform the actuators of the

different faces of the satellite to perform the necessary forces.

In testing, I tried to use values that could be realistically seen from the magnetometer and gyroscope while also accounting for edge cases such as zero, negative, or very large values [2]. Additionally, I integrated the code base's existing Make tools to automate testing on command "Make test". The experience pushed me to learn more testing techniques, tools, and methodologies such as setting up testing environments via fixtures and experimenting with mocking to get around hardware calls. While other tests I developed for functions.py were left behind due to restructuring, it was still an incredibly useful learning experience that helped familiarize me with the repository's interacting parts and served as a useful reference for my other team members beginning to add tests to their work.

2.7.2 Bit Flag NVM Refactor

Before the introduction of SD card functionalities, storing information across satellite reboots relied on a limited amount of non-volatile memory (NVM) stored on the flight controller board. The memory is utilized in the form of an array of bytes that is used to store various single bit flags that indicate critical status information including: burn wire success, occurrence of shutdown, brownout, or soft reboot, and FSK radio frequency activation. After identifying inefficiencies in how these flags were defined and manipulated, I refactored the system by implementing a Flag class that streamlined access to and control of individual bits. Each Flag is defined by its byte index (_index), bit position within that byte (_bit), a shared byte array/NVM (_datastore), and a derived bitmask (_bit_mask). The design supports two key methods:

- get() accesses a specific byte of the array via the index and performs a bitwise AND operation with the bitmask to read the value of a single specified bit. True is returned if the bit is 1, and false if 0.
- toggle() takes in a boolean value to set a specific bit to either 1 or 0. If the value passed in is true, a bitwise OR is performed with the bitmask on the specified byte to turn the bit to 1. Passing in false will invert the bitmask and perform bitwise AND on the specified byte to produce a 0 bit.

To verify the correctness and robustness of this new system, I developed a suite of unit tests using PyTest. These tests validate the flag initialization, correct behavior of the get() and toggle() methods under normal and edge-case conditions (e.g., first and last bits in a byte), and the persistence of flag states. This testing ensures that each bit flag behaves predictably and that the new system remains maintainable and safe for mission-critical uses, such as detecting whether critical startup conditions or hardware faults have occurred.

2.7.3 Reintroducing SD Card Utilities

As part of the team's plans to expand onboard data storage capabilities, I took on the task of reintegrating legacy SD card functionality into the current repository. This involved adding clear comments and documentation throughout the code, applying type hinting to all functions to improve readability and maintainability, and implementing type checks and exception handling to catch method misuse and improve runtime stability.

What the SD card module supports:

- SD Card Initialization and Mounting: Sets up SPI bus and a FAT file system and /sd directory on the SD card.
- File Operations: Read, write, append, replace, insert and copy files.

- File System Management: Create unique file names, manage and print out directories and their contents, and delete files and directories.
- USB Drive Mode Toggle: Modify a JSON config to enable/disable USB write access.

The future intentions with establishing a file system are to eventually be able to save logs of system health, events, telemetry, etc. These logs will be transmitted to ground stations during downlink windows. Ideally, automatic cleanup operations will be able to preserve memory space for core functions without manual intervention. Ultimately, this file system will allow the satellite to retain critical data across reboots, support autonomous fault tracking, and provide a transparency of operations to ground station operators.

2.7.4 In Progress Work and Next Tasks

I am currently focused on consolidating the pysquared directory, the high level interface layer of the project. This documentation aims to clearly explain the purpose and functionality of key modules, methods, and components critical to satellite operations. To support this effort, I am using a tool called MkDocs, a static site generator, to build a browsable, locally launchable HTML website. The site will eventually be hosted via Github Pages to allow automatic regeneration of documentation upon any changes to the code base. As development on the ground station software begins, I plan to integrate Grafana, a powerful data visualization platform, to display and monitor up to date satellite data intercepted by the lab's ground station. Lastly, I am finalizing preparations to send the Pleiades Maia mission patches, which I designed, to production - an effort that will help promote our team identity and mission visibility.

3 Results/Outcomes

The Texas State software development team has made extensive contributions to the Pleiades Maia mission, resulting in significant improvements to software infrastructure, testing reliability, and system-level functionality. These contributions have set up the flight software for successful long-term maintainability while also strengthening the collaborative and educational foundation of the project.

Key Outcomes:

- Improved Development Infrastructure:
  - Established continuous integration pipelines, pre-commit hooks, and streamlined Make tooling to automate testing and accelerate development cycles.
  - Integrated dependency locking and Intellisense support to enhance consistency and developer productivity.
- Enhanced Code Architecture and Modularity:
  - Refactored repository architecture to follow modern design patterns (ex. dependency injection, ADRs).
  - Modularized the codebase to support version-specific development (V4 and V5), and isolated the customizable payload layer for future kit users.
- Robust Testing and Quality Assurance:
  - Introduced PyTest-based unit testing with approximately 50% code coverage.
  - Implemented validation checks for satellite configuration inputs and established a structured logging system to track runtime events and anomalies.
- Embedded Functionality Expansion:

    - Refactored and reintroduced key features including SD card file system support, asyncio multiprocessing, and NVM-based bit flag tracking.
    - Streamlined firmware onboarding its and contributed Texas State's CircuitPy firmware version to the official upstream CircuitPython project, broadening its community impact.
- Ground Systems and Communications:
    - Initiated ground station development, including amateur radio licensing and early radio communication testing between flight boards.
- Outreach and Culture Building:
    - Developed and presented at the PyTexas conference, designed and prepared Pleiades Maia mission patches for production, and supported new team members through interviews and weekly demonstrations.
- Personal Contributions:
    - Introduced Pytest tools, refactored the NVM bit flag system with test coverage, reintroduced SD card utilities with documentation and type safety, and began building MkDocs-based user documentation.

4 Conclusion/Future Works

This independent study has been a deep dive into satellite systems development through Texas State University's involvement in the Pleiades Maia mission. The project provided the opportunity to contribute meaningfully to the creation of open-source, accessible satellite flight software and work that supports a broader mission to make space research approachable for educational institutions with limited resources.

Through collaborative development with my peers, we improved the modularity, reliability, and maintainability of the Proves Kit CircuitPy flight software. My personal contributions– establishing the project's initial unit testing strategy, refactoring the non-volatile memory bit flag system, reintroducing SD card functionality, and beginning user-facing documentation and data visualization tools –have all helped solidify the software foundation ahead of Pleiades Maia's launch.

These contributions not only granted me experience with embedded systems, Python development, and software architecture, but also gave me valuable insight into working on a large, distributed, open-source project. Collaborating with students and researchers across universities, I learned how to communicate technical decisions clearly and contribute to a living codebase with mission-critical applications.

Looking forward, I will continue expanding our documentation and user interface tools. The upcoming ground station software will incorporate data visualization with Grafana, enabling up to date monitoring and telemetry tracking during flight. With a launch scheduled for early 2026 and our lab anxious to start on a new 2U satellite mission of our own, our team is preparing to complete core development by the end of summer 2025. Additionally, the mission patch I designed will soon be produced to celebrate and represent our team's work.

Pleiades Maia is only the beginning. The tools, systems, and lessons learned here are forming the foundation for Texas State's continued presence in space exploration and I am excited to be part of its next chapter.

Acknowledgments

I wish to thank my fellow members of the Texas State University Space Lab software development team for all their collaboration, support, and hard work on this project. I am incredibly grateful for the opportunity to get to learn from you all and work by your side.

It has to be acknowledged the incredible contribution of time, knowledge, and enthusiasm to this project from the entire lab. I truly reflect every day on how lucky I am to be included in this group of extraordinary individuals.

Finally, I would also like to thank Dr. Mylene Queiroz de Farias for her mentorship and assistance through this independent study.

Space Lab Directors:

Dr. Blagoy Rangelov

rangelov@txstate.edu

Professor Evan Jellison

egjellison@txstate.edu

Space Lab Software Team Lead:

Nate Gay

nategay@txstate.edu

References

[1] “Circuitpython.” *PROVES Kit RP2040 V4 Download*, circuitpython.org/board/proveskit_rp2040_v4/. Accessed 22 Apr. 2025.

[2] “Created Unit Tests for Detumble.Py by TaylorGilg · Pull Request #57 · Proveskit/Pysquared.” *GitHub*, Pleiades 5 PROVES Kit, github.com/proveskit/pysquared/pull/57. Accessed 22 Apr. 2025.

[3] Karaliunaite, Vaida. “CubeSat 101: The Comprehensive Guide to Understanding Satellite Technology.” *NanoAvionics*, 3 Oct. 2023, nanoavionics.com/blog/cubesat-101-the-comprehensive-guide-to-understanding-satellite-technology/.

[4] “NASA Selects New Round of Candidates for CubeSat Missions to Station.” *NASA*, NASA, 26 Mar. 2024, www.nasa.gov/centers-and-facilities/kennedy/nasa-selects-new-round-of-candidates-for-cubesat-missions-to-station/.

[5] Pham, Michael. “The Proves CubeSat Kit.” *GitHub*, Pleiades 5, github.com/proveskit. Accessed 22 Apr. 2025.

[6] Pham, Michael. “Welcome to the Proves Kit!” *Welcome - The PROVES Kit Documentation*, Pleiades 5, docs.proveskit.space/en/latest/. Accessed 22 Apr. 2025.

[7] “Proveskit/Circuitpython_rp2040_v4: Circuitpython Flight Software for the Proves Kit RP2040 V4 Flight Controller Board.” *GitHub*, Pleiades 5 PROVES Kit, github.com/proveskit/CircuitPython_RP2040_v4. Accessed 22 Apr. 2025.

[8] “Proveskit/Circuitpython_rp2040_v5: Circuitpython Flight Software for the Proves Kit RP2040 V5 Flight Controller Board.” *GitHub*, Pleiades 5 PROVES Kit, github.com/proveskit/CircuitPython_RP2040_v5. Accessed 22 Apr. 2025.

[9] “Proveskit/Pysquared: Pysquared Flight Software Library for the Proves Kit.” *GitHub*, Pleiades 5 PROVES Kit, github.com/proveskit/pysquared. Accessed 22 Apr. 2025.

[10] “Pull Requests · Proveskit/Pysquared.” *GitHub*, Pleiades 5 PROVES Kit, github.com/proveskit/pysquared/pulls?q=is%3Apr%2Bauthor%3ATaylorGilg%2Bis%3Aclosed. Accessed 22 Apr. 2025.

[11] “Watch Shoebox-Sized Cubesats Deploy from ISS in These Amazing Views.” *YouTube*, VideoFromSpace, 28 June 2024, www.youtube.com/watch?v=s8hhvvOqmmk.