Space Weather Center

Ryan Laidlaw, Luke Rossel, Sam Harris, Alec Rydeen, Tamaralynn Shute
Department of Computer Science, University of New Hampshire, Durham, NH 03824

Introduction

The UNH Space Weather Center processes real-time satellite data that supports scientific research and forecasting. With the addition of new satellites, such as SWFO-L1, the volume, velocity, and diversity of incoming data continues to increase.

Previous groups focused on real-time DSCOVR ingestion, predictive machine learning models, and interactive dashboards. But integrating SWFO-L1 required:

- Expanded ingestion and preprocessing
- Improved automation and service reliability
- Modernized dependency management
- A scalable architecture for future satellites

By improving the data pipeline, enhancing infrastructure reliability, and modernizing the repository we aim to create a maintainable, extensible foundation for future Space Weather groups to build from.

System Requirements

Functional:

- Ingest real-time satellite data automatically
- Support multiple satellite formats (DSCOVR & SWFO-L1)
- Generate standardized NetCDF outputs
- Provide API-accessible processed datasets
- Support ML-ready data pipelines

Non-Functional:

- High availability and automation (systemd services & timers)
- Robust failure handling and logging
- Modular and maintainable repository structure
- Dependency management and reproducible environments

Stakeholder:

- Researchers require validated, comparable datasets
- Infrastructure teams require reliable, observable services
- Future developers require a documented and scalable design

Design

Modular Separation:

- Data processing logic separated from operational services
- API components isolated from ingestion/transformation logic
- Clear boundaries between ingestion, processing, ML and serving layers

Infrastructure & Automation:

- Background processing managed via systemd services
- Failure recovery and logging mechanisms
- Automated scheduled ingestion

Repository Modernization:

- Transition from legacy setup configuration to pyproject.toml
- Unified dependency management for reproducible env.
- Removal of outdated configuration structures
- Improved documentation and onboarding workflow

NetCDF Framework:

- Implemented proof-of-concept empty NetCDF generation
- Designed schema-driven structure for standardized outputs
- Establishes scalable template system for future satellites

Data Pipeline Flow Chart

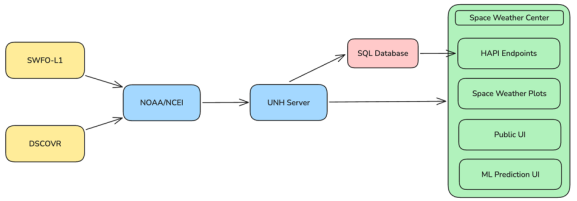

Figure 1. Comprehensive architecture of the space weather data pipeline, showing data acquisition from satellite sources, automated ingestion via systemd services, NetCDF preprocessing and validation, SQL-based storage, machine learning inference, and delivery through HAPI endpoints and user interfaces

End-to-End Data Flow

1. Data Sources:
DSCOVR · SWFO-L1
2. Data Ingestion:
Automated downloads (systemd)
3. Preprocessing & Validation:
NetCDF Generation · Metadata extraction
4. SQL Database:
Archived data storage · Fast access
5. Machine Learning Inference:
Model execution · Predictions
6. Serving & User Access:
HAPI · Dashboards · Research access

Data Access Web Pages

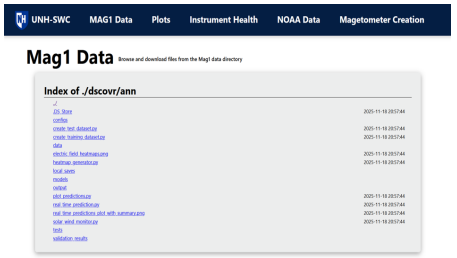

Figure 2. Mag1 Data Directory Page on the Space Weather Center site. Allows users to view and download data from the onboard magnetometer

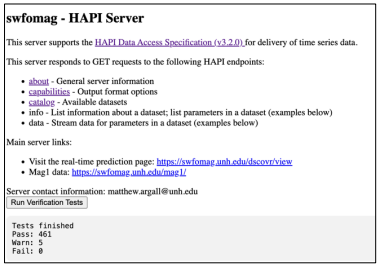

Figure 3. HAPI landing page on the Space Weather Center site. Gives users information about how to query the server for raw magnetometer data

Space Weather Center Web Pages

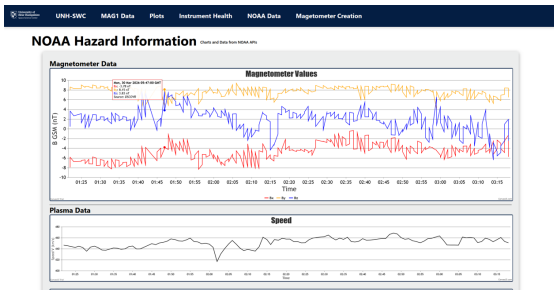

Figure 4. Graphs of Magnetometer data from NOAA APIs, displaying source spacecraft

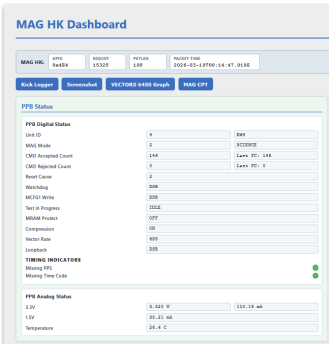

Figure 5. Magnetometer housekeeping data dashboard, which depicts the status of different instruments and telemetry.

Implementation

Heliophysics Data API (HAPI)

- Receives data directly from the SQL database
- Extended from existing Flask routes to provide direct magnetometer data access

Server Reorganization

- Leverages pyproject.toml file for increased portability for data processing logic

User Interface

- Descriptive, public facing Flask/HTML pages
- Magnetometer data analysis, Instrument Health Monitoring, and Data access

Machine Learning Framework

- Uses an artificial neural network (ANN) model to predict non-linear short-term forecasts
- Enhanced with Monte Carlo Dropout for predictive variance estimation

Testing

HAPI

- Open-source unit testing for endpoint functionality
- Test scripts using SQLite for database testing

NetCDF Framework

- Unit testing for verifying dimensions and metadata
- Unit testing for data processing

Machine Learning Framework

- Simulated CSV test and training datasets for model validation
- Evaluated using mean squared error, root mean squared error, and R^2

User Interface / User Experience

- Regular check-in meetings with sponsor and stakeholders
- User experience testing done with peers during development of pages

Evaluation and Conclusions

Our project completed its main success criteria of improving system maturity and user experience. Key evaluation metrics included system reliability, data accessibility, and interface usability. Reliability was improved through consistent pipeline output during data gaps, and data accessibility was improved through HAPI endpoints and structured storage.

User experience was intended to be evaluated through user feedback and usability testing of the updated UI. However, due to time constraints, we could not collect user reviews or quantitative metric. As a result, UI evaluation is currently based on internal validation instead of external feedback.

Overall, our system displays clear improvements in robustness and accessibility, but future work would be focused on collecting user feedback and expanding the interactive UI.

Acknowledgements

We would like to thank:

- Dr. Matthew Argall - sponsor
- David Benedetto - advisor