🚀 Spacecraft Telemetry Analyzer

KVGCE Hackwise 2025
Organized by Sphere Hive
Team Members: Abdhul Azeez, Chethan, Manya, Jithin

---

1. Indrocuction

This project presents a real-time spacecraft telemetry analyzer designed to detect anomalies, generate critical alerts (both audio and visual), and provide advanced 2D and 3D telemetry visualizations. It supports multiple mission profiles (LEO Satellite, Mars Mission, etc.) and provides a simulation environment for testing and validation. Built with Streamlit, Plotly, Matplotlib, and Python, it offers an intuitive dashboard for spacecraft health monitoring.

2. Methodology

- Load telemetry CSV file (or generate simulated data).
- Preprocess data, parse timestamps, and handle missing values.
- Highlight critical telemetry anomalies based on threshold violations.
- Generate real-time visual and audio alerts using pyttsx3.
- Visualize telemetry parameters with 2D line plots and 3D scatter/mesh plots.
- Allow mission-specific threshold selection (e.g., Mars, Moon, Deep Space).
- Export analyzed results into Excel reports and downloadable graphs.
- Display system health status using dynamic gauges and status color codes.

3. Implementation Details

- Frontend: Built entirely with Streamlit for a rapid, beautiful UI.
- Visualization:

  - 2D Line graphs: matplotlib
  - 3D Scatter & Mesh: plotly.express, plotly.graph_objects
- Anomaly Detection: Custom thresholds for each telemetry parameter.
- Audio Alerts: Text-to-speech notifications using pyttsx3.
- Download Options: Excel summary, individual graphs as PNGs.
- Simulation Mode: User-adjustable settings for generating synthetic telemetry data.
- Mission Profiles: Custom threshold loading for real-world spacecraft conditions.

4. Dataset Description

- Format: CSV
- Expected Columns:
  - timestamp (datetime format %d-%m-%Y %H:%M)
  - temperature (°C)
  - pressure (atm)
  - velocity (m/s)
  - battery (%)
  - fuel (%)

Example snippet:

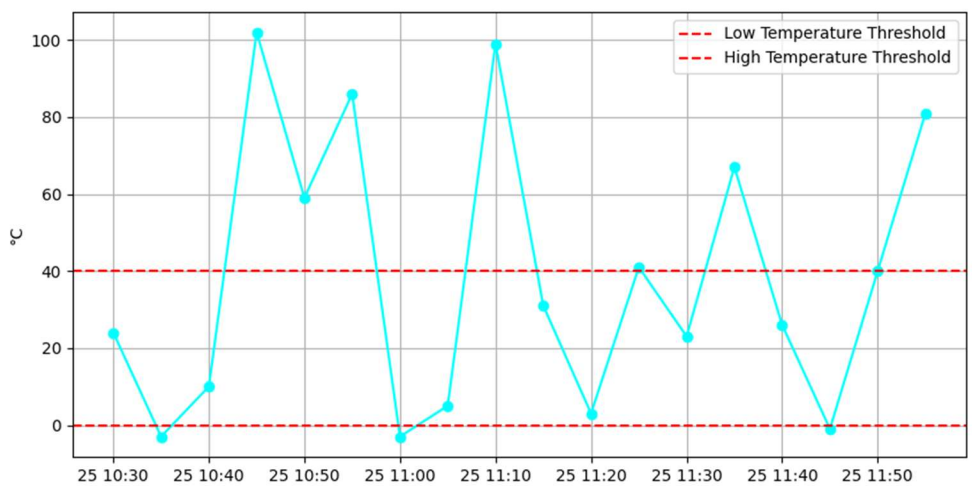
timestamp	temperature	pressure	velocity	battery	fuel
2025-04-25 09:00:00	22.5	1.01	1200	95	80
2025-04-25 10:00:00	23.1	0.98	1250	90	75
2025-04-25 11:00:00	41.5	1.05	1300	85	70

5. Output Format

- Dashboards:
  - 2D graphs for each parameter with anomaly lines.
  - 3D scatter and mesh plots showing telemetry relationships.
  - Styled tables highlighting anomalies.
  - Dynamic system health gauges.

- Export:
  - Downloadable Excel report.
  - Individual graph download as PNG.
- Alerts:
  - Audio announcements of critical anomalies.
  - Visual warning banners inside the app.

	timestamp	temperature	pressure	velocity	battery	fuel
count	18	18	18	18	18	18
mean	2025-04-25 11:12:30	38.3333	0.6056	7320.8889	74	58.7778
min	2025-04-25 10:30:00	-3	0.33	7017	57	0
25%	2025-04-25 10:51:15	6.25	0.4425	7097	67.25	41.5
50%	2025-04-25 11:12:30	28.5	0.6	7249	69	68.5
75%	2025-04-25 11:33:45	65	0.6975	7504.25	84	78.5
max	2025-04-25 11:55:00	102	0.97	7767	100	97
std	None	35.8576	0.2166	255.3053	13.6382	28.3643



6. Challenges Faced and Solutions

- Integrating real-time audio without freezing UI: Used multi-threaded pyttsx3 audio alerts
- Handling missing or wrong timestamp formats: Applied pandas.to_datetime with error coercion
- Streamlit selectbox duplicate key errors: Assigned unique keys to all widgets
- Displaying large 3D plots responsively: Optimized Plotly rendering and

layout management
- Managing mission profile customization: Dynamic threshold override and real-time profile loading

7. Test Cases and Results

- Normal CSV file: Clean telemetry data → Correct graphs and nominal health → Success ✅
- Anomalous CSV file: High temperature (45°C) → Critical alert and audio notification → Success ✅
- Simulated data with 20% anomaly chance: Random synthetic values → 2-4 anomaly alerts generated → Success ✅
- Profile Change to Mars Mission: New thresholds applied → Mars-based telemetry visualization → Success ✅

8. Future Improvements

- Add automatic email notification with SMTP for critical alerts.
- Integrate cloud-based data storage (e.g., Firebase, AWS S3).
- Support live streaming telemetry data over sockets.
- Deploy app to public cloud platforms (like Streamlit Sharing, AWS, or Azure).
- Integrate anomaly detection AI models for smarter predictions.

9. References

- Streamlit Documentation (https://docs.streamlit.io/)
- Matplotlib Documentation (https://matplotlib.org/)
- Plotly Python Graphing Library (https://plotly.com/python/)
- Pyttsx3 (Text-to-Speech) (https://pypi.org/project/pyttsx3/)
- OpenCV and Pandas Libraries (https://pandas.pydata.org/)