Design Report: Spacecraft Subsystem Simulation

• Introduction

This simulation models a spacecraft operating in Low Earth Orbit (LEO) using object-oriented programming in Python. The simulation includes key aerospace-like subsystems and tasks such as tacking and controlling its altitude, managing power implementing solar input and battery consumption, thermal camera as payload operation and status communication as well as handling unexpected scenarios such as eclipse, energy constraints and gyroscopic drift.

• Subsystem Models

- Attitude Control System:

Tracks spacecraft orientation and corrects itself when anomalies detected.

- Power System:

Manages and choses between energy generated from solar input and batteries, tracks battery charge and discharge. It also simulates battery response to eclipse conditions, limiting the power input to only batteries given the case. The discharge increases if the fan for the payload is turned on.

- Payload Camera:

Simulates a scientific payload that detects temperature and power levels. Includes logic for overheating cases and cooling using a power consuming fan.

- Communications System:

Models a basic communication system that automatically shuts down during low battery to preserve energy. Just stays transmitting or not.

- Event Manager:

Handles random anomaly production and time-dependent events like eclipse periods. Implements recovery responses.

- Spacecraft:

Aggregates and integrates all subsystems. Runs the simulation loop, calls updates, and handles reporting seen in terminal.

• Object-Oriented Programming Principles

- Abstraction:

All subsystems inherit from a shared abstract base class `Subsystem`, ensuring a consistent interface (“update()” and ”status()”).

- Encapsulation:

Subsystem states such as “battery_level”, “temperature”, and “orientation” are private to their classes. They are only modified through defined methods to preserve integrity of main code.

- Inheritance:

Concrete classes like `AttitudeControlSystem` and `PowerSystem` are extensions to the `Subsystem` base class in order to reuse common structure and behavior.

- Polymorphism:

All subsystems implement the same interface (`update()`, `status()`), which allows for uniform and simplified control in the main loop regardless of the subsystem called.

• Anomaly Handling and Recovery Logic

The “EventManager” periodically injects random anomalies and simulates somewhat realistic conditions:

- Eclipse Events:

Solar power is cut off for a defined period. Battery discharges more to support the subsystems. If battery level drops below 20 Wh, the spacecraft adjusts its orientation to maximize solar input when eclipse is over. If battery levels are too low, communications and payload subsystems are shut down to maintain other power and altitude systems running.

- Gyro Drift:

Randomly applied “weight” to the attitude system. It can be automatically corrected on the next update cycle by stabilization logic which is assumed immediately corrected.

- Payload Overheat:

Random thermal anomalies raise basal temperature. A fan is activated if overheating is detected, which increases power consumption but gradually lowers temperature.

- Battery-Based Emergency Shutdowns:

- Battery < 5 Wh: CommSystem shuts down.
- Battery < 3 Wh: PayloadCamera shuts down.