**CubeSat Onboard Computer Integration Report**

**Introduction**

This report outlines the step-by-step process followed to compile, debug, and integrate the MySensorApp into the Core Flight System (cFS) for the CubeSat program using the provided Docker environment and challenge setup. It also covers the telemetry handling and system behavior validation.

**Step 1: Setup Environment and Docker Container**

* Installed Docker on the host machine.
* Pulled the challenge Docker image using:

text

docker pull hk2989441/adcs\_sim:test2

* Launched the container with required privileges:

text

docker run -it --cap-add=sys\_nice --cap-add=ipc\_lock --cap-add=SYS\_RESOURCE \

--ulimit rtprio=99 --ulimit memlock=-1 --ulimit nice=-20 \

--sysctl fs.mqueue.msg\_max=256 --sysctl fs.mqueue.msgsize\_max=65536 \

--user root --name adcs\_sim hk2989441/adcs\_sim:test2

**Step 2: Building the cFS System**

* Accessed the container shell.
* Changed directory to the cFS folder (ensuring proper capitalization):

text

cd /home/osk/cFS

* Verified presence of the Makefile.
* Ran the build with:

text

make

* Build completed successfully, compiling cFS core and all apps including MySensorApp.

**Step 3: Creating Startup Script**

* Noticed missing startup script cfe\_es\_startup.scr in /cf.
* Created the /cf directory:

text

mkdir /cf

* Created startup script to launch MySensorApp automatically:

text

cat <<EOF > /cf/cfe\_es\_startup.scr

# cFE ES Startup Script

APP MySensorApp

EOF

**Step 4: Running the cFS Core Executable**

* Located the core executable at:

text

/home/osk/cFS/build/i686-linux-gnu/default\_cpu1/cpu1/core-cpu1

* Ran the executable. It launched and initialized the cFS runtime.
* Observed expected continuous running behavior (running operationally, awaiting commands and telemetry).

**Step 5: Telemetry Handling**

* Searched for telemetry Python scripts inside container, found:
  + /home/osk/telemetry\_bridge.py
  + /home/osk/cFS/tools/cFS-GroundSystem/Subsystems/tlmGUI/UiGenerictelemetrydialog.py
* Ran telemetry script to listen and process cFS telemetry data, enabling data monitoring during runtime.

**Step 6: Fault Injection and System Response**

* Observed system state transitions from normal to warning to altitude failure during testing.
* This behavior is expected during fault injection, demonstrating:
  + Detection of anomalies
  + Issuance of warnings
  + Entry into failure modes for robust fault management

**Python Telemetry Bridge**

File: telemetry\_bridge.py

Functionality:

* Sends CCSDS-formatted telemetry packets via UDP to 127.0.0.1:1234.
* Simulates:
  + Gyroscope rates (x, y, z)
  + Altitude decay
  + Drop calculation
* Updates telemetry status based on drop thresholds:
  + NORMAL → drop < 0.8 × threshold
  + WARNING → drop ≥ 0.8 × threshold
  + ALTITUDE\_FAIL → drop ≥ threshold

Modifications Made:

1. Altitude decay rate adjusted to ensure 1 km drop within 360 seconds.
2. Automatic termination added when ALTITUDE\_FAIL is reached.

Resulting Output Example:

0s Alt: 359.997km Drop: 0.00km Status: NORMAL

1s Alt: 359.994km Drop: 0.01km Status: NORMAL

...

360s Alt: 359.0km Drop: 1.00km Status: ALTITUDE\_FAIL

* Simulation stops automatically at ALTITUDE\_FAIL

**Summary**

* Successfully set up, built, and launched the cFS environment with integrated MySensorApp in the provided Docker container.
* Created necessary startup configurations to automate app launch.
* Ran and monitored telemetry to validate system health and functionality.
* Demonstrated the system’s dynamic fault detection and response through state transitions, fulfilling challenge objectives.

**Appendix: Key Commands Used**

bash

docker pull hk2989441/adcs\_sim:test2

docker run -it --cap-add=sys\_nice --cap-add=ipc\_lock --cap-add=SYS\_RESOURCE \

--ulimit rtprio=99 --ulimit memlock=-1 --ulimit nice=-20 \

--sysctl fs.mqueue.msg\_max=256 --sysctl fs.mqueue.msgsize\_max=65536 \

--user root --name adcs\_sim hk2989441/adcs\_sim:test2

cd /home/osk/cFS

make

mkdir /cf

cat <<EOF > /cf/cfe\_es\_startup.scr

# cFE ES Startup Script

APP MySensorApp

EOF

/home/osk/cFS/build/i686-linux-gnu/default\_cpu1/cpu1/core-cpu1

python3 /home/osk/telemetry\_bridge.py