# CubeSat ADCS OBC Challenge – Full Documentation

## 1. Overview

This document summarizes the complete workflow for the CubeSat ADCS On-Board Computer (OBC) Challenge, including bug fixing, code integration with cFS, fault injection, testing, Docker setup, and submission preparation.

## 2. Task Objectives

1. **Bug Fixing**
   * Correct C++ source files.
   * Ensure compilation without errors.
   * Maintain core logic.
2. **Integration**
   * Integrate corrected code into cFS.
   * Connect outputs to telemetry Python script.
3. **Fault Injection**
   * Introduce controlled faults.
   * Demonstrate threshold breach in video.

## 3. Files Corrected

* adcs\_controller.cpp
* microcontroller.cpp
* microcontroller\_wrapper\_c.cpp
* pid\_controller.cpp
* adcs\_test\_harness.cpp

## 4. Bug Fixing & Compilation Steps

1. Copied corrected source files into fsw/src.
2. Fixed header inclusion issues and paths.
3. Ensured SensorData\_t and ActuatorCommands\_t types were used consistently.
4. Updated microcontroller.cpp to use proper member variable names.
5. Changed arrays to std::array<float, 3> for gyro\_rates\_, magnetometer\_, wheel\_torques\_, magnetorquer\_ to allow .data() usage.
6. Updated adcs\_test\_harness.cpp to match corrected structures.
7. Compiled using:

g++ -std=c++17 -I./inc -I./src \

src/adcs\_controller.cpp \

src/microcontroller.cpp \

src/microcontroller\_wrapper\_c.cpp \

src/pid\_controller.cpp \

adcs\_test\_harness.cpp \

-o adcs\_test\_harness

## 5. Integration with cFS

1. Built the cFS system in the cFS/build directory.
2. Verified integration by compiling space\_adcs\_app and linking with cFS headers.
3. Connected telemetry outputs to telemetry\_bridge.py.

Run command for integrated test:

./adcs\_test\_harness | python3 /home/osk/telemetry\_bridge.py

## 6. Fault Injection

* Introduced a controlled gyro fault at cycle 10 in adcs\_test\_harness.cpp.
* Verified fault detection and status change.
* Video demonstration captured showing fault injection.

## 7. Docker Image Creation

1. **Dockerfile:**

FROM ubuntu:22.04

RUN apt-get update && apt-get install -y \

build-essential \

g++ \

python3 \

python3-pip \

&& rm -rf /var/lib/apt/lists/\*

COPY . /workspace

WORKDIR /workspace/fsw

RUN g++ -std=c++17 -I./inc -I./src \

src/adcs\_controller.cpp \

src/microcontroller.cpp \

src/microcontroller\_wrapper\_c.cpp \

src/pid\_controller.cpp \

adcs\_test\_harness.cpp \

-o adcs\_test\_harness

CMD ["bash"]

1. Build image:

docker build -t adcs\_cfs\_sim:latest .

1. Run container:

docker run -it adcs\_cfs\_sim:latest

cd fsw

./adcs\_test\_harness | python3 /workspace/telemetry\_bridge.py

## 8. GitHub Repository Structure

* src/ : Corrected source files.
* inc/ : Header files.
* adcs\_test\_harness.cpp : Simulation harness.
* telemetry\_bridge.py : Telemetry output.
* DOC.md : Documentation.
* Dockerfile : Build environment for container.
* README.md : Project instructions and build/run steps.

## 9. Submission Components

1. GitHub repository with source files and documentation.
2. Docker image (adcs\_cfs\_sim:latest) containing compiled system.
3. Demo video showcasing:
   * Integrated system running.
   * Fault injection and threshold breach demonstration.

## 10. Summary of Steps Carried Out

1. Corrected header includes and type inconsistencies.
2. Fixed variable naming issues in microcontroller.cpp.
3. Changed array implementations to std::array for .data() usage.
4. Created adcs\_test\_harness.cpp for simulation.
5. Integrated corrected files with cFS.
6. Connected simulation output to telemetry\_bridge.py.
7. Implemented controlled fault injection.
8. Compiled and verified system.
9. Created Docker image for reproducible environment.
10. Recorded demo video for submission.

This document represents a full consolidated workflow from bug fixing to demo preparation.