Reliable and Self-Configurable Flight Software Architecture for CubeSats

Te-Chuan Huang, Cheng-Ting Wu, and Jyh-Ching Juang

Department of Electrical Engineering

National Cheng Kung University, Tainan, Taiwan

Photo by NASA (July, 2014)

Outline

- Introduction
- PHOENIX Nanosatellite
- Flight Software Design
- Resilient Software Design
- Testing and Verification
- Conclusion

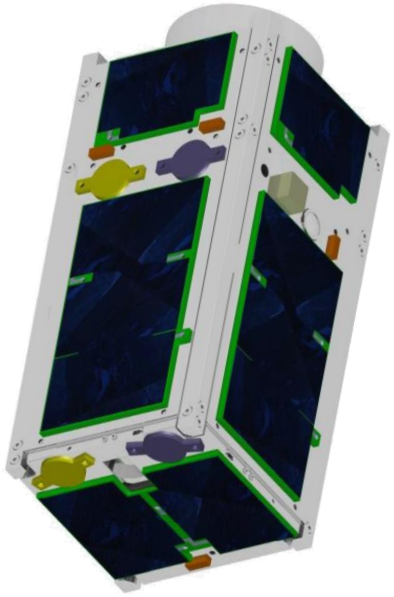
Outline

- **Introduction**
- PHOENIX Nanosatellite
- Flight Software Design
- Resilient Software Design
- Testing and Verification
- Conclusion

Introduction

- CubeSat
  - Standardized nanosatellite
  - Inexpensive cost and shorter development time
  - Commercial Off-The-Shelf (COTS) products available

- On Board Data Handling (OBDH) Subsystem
  - Hardware - On Board Computer (OBC)
  - Software - Flight Software (FSW)



Introduction

- Flight Software (FSW)
  - Limited resource
  - Unpredictable space environment
  - Responsible for many task
  - Hard to modify after launch

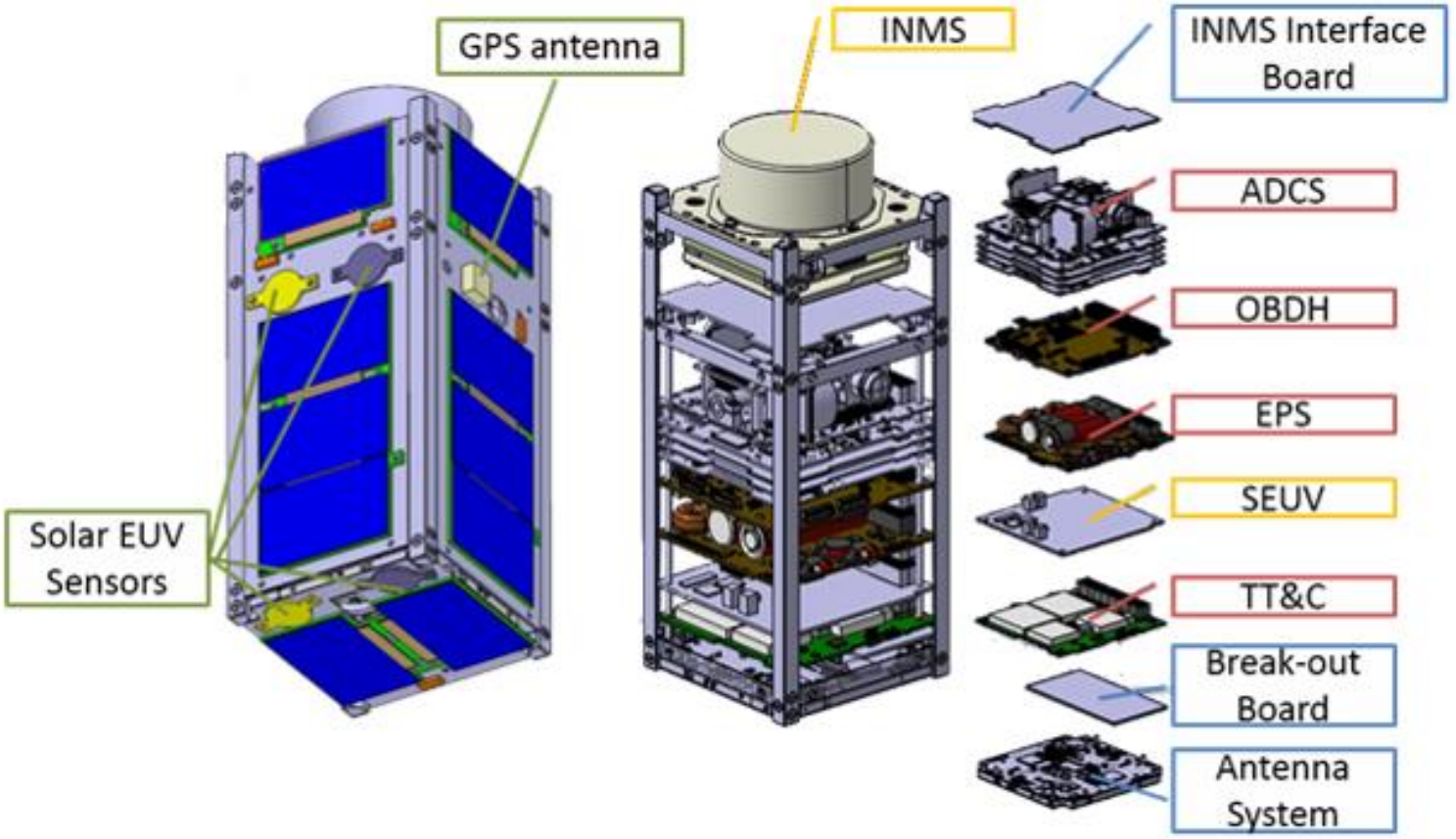
- Several techniques of design and implementation of reliable FSW are presented.

Outline

- Introduction
- **PHOENIX Nanosatellite**
- Flight Software Design
- Resilient Software Design
- Testing and Verification
- Conclusion

PHOENIX Nanosatellite

- PHOENIX is one of the CubeSat in QB50 project.



PHOENIX Nanosatellite

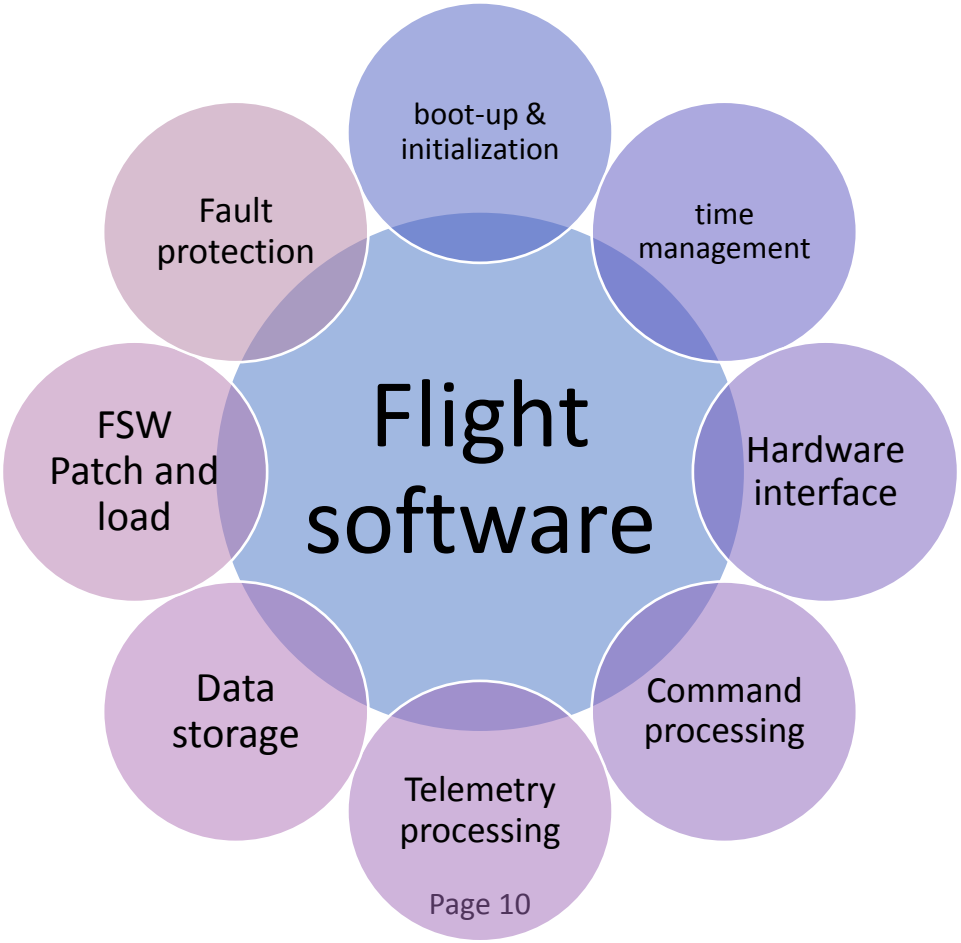
- QB50
  - EU's FP7 project managed by VKI
  - Carrying out atmospheric research within the lower thermosphere
- PHOENIX
  - Subsystem: EPS, OBDH, COM, ADCS, ANTS & GPSR.
  - Science payload: INMS and SEUV
- OBDH
  - GomSpace NanoMind A712D
  - FreeRTOS operating system

Outline

- Introduction
- PHOENIX Nanosatellite
- **Flight Software Design**
- Resilient Software Design
- Testing and Verification
- Conclusion

Flight Software Design
- Overview -

- Flight software is embedded in the On Board Data Handling (OBDH) subsystem in CubeSat and serves as a brain of the satellite.

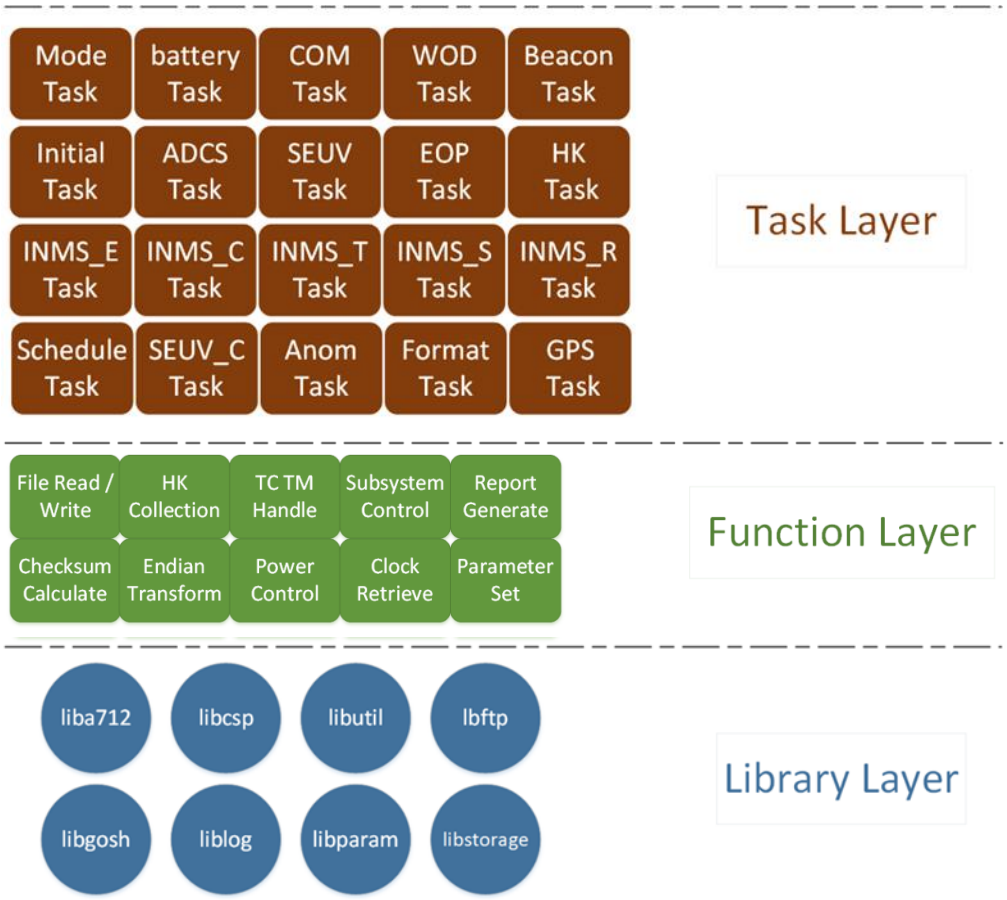


Flight Software Design
- Requirements -

- By analyzing the requirements, a system view from top to down helps the development of software.
  - Functional Requirements
  - Interface Requirements
  - Operational Requirements
  - Software Reliability Requirements
  - Software Safety Requirements

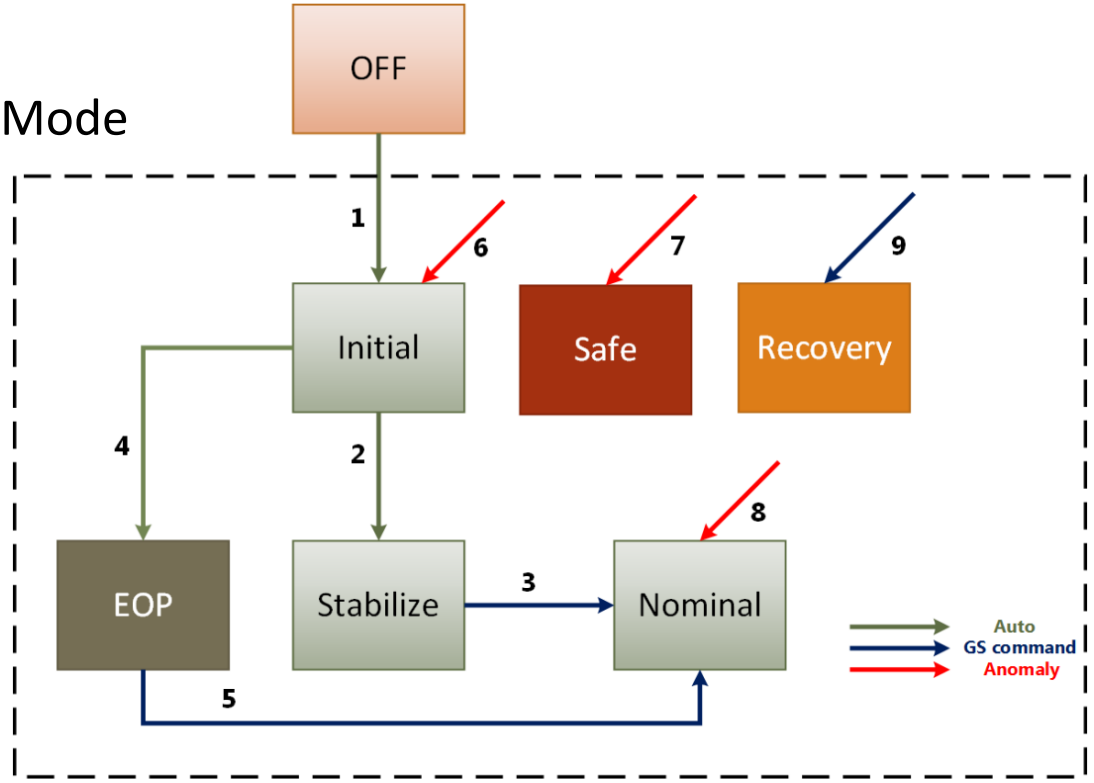
Flight Software Design
- Architecture -

- Three layered architectures



Flight Software Design
- Operation Mode -

- Mode Transition
  - Initial Mode
  - Early Orbit Phase (EOP) Mode
  - Stabilize Mode
  - Nominal Mode
  - Safe Mode
  - Recovery Mode



AUTO: flight software trigger

COM: telecommand trigger

Anomaly: anomaly trigger

Outline

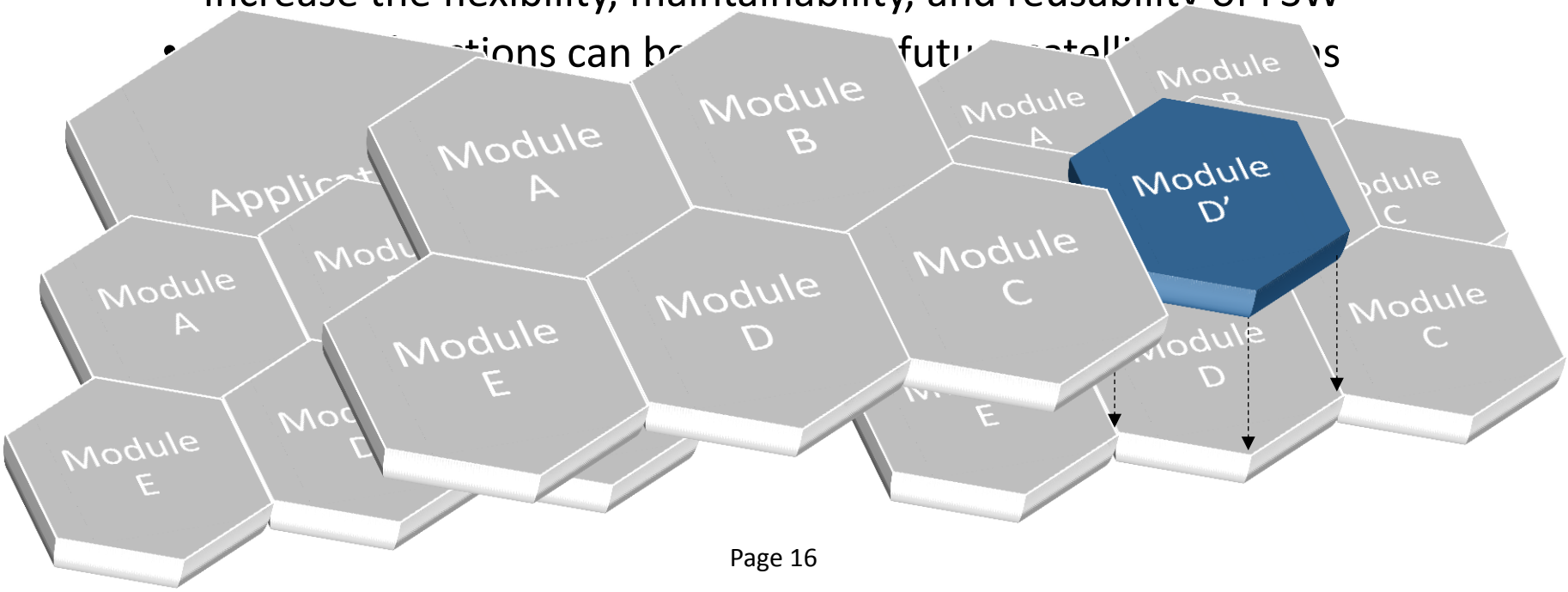
- Introduction
- PHOENIX Nanosatellite
- Flight Software Design
- **Resilient Software Design**
- Testing and Verification
- Conclusion

Resilient Software Design

- Several method proposed to increase the reliability of the FSW
  - Modular Programming
  - Redundancy Memory Design
  - Anomaly Handling
  - Self-Configurable Architecture
  - On Board Scheduling Management
  - In Orbit Software Update

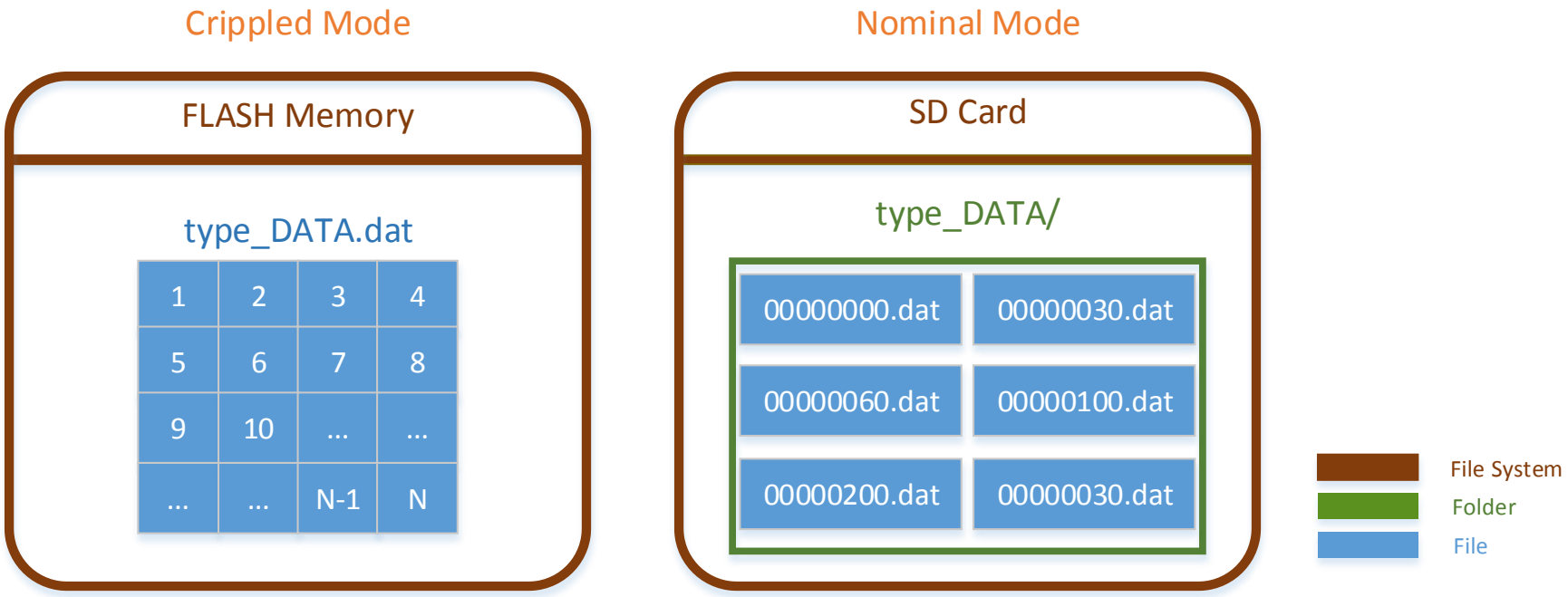
Resilient Software Design
- Modular Programming -

- Features
  - Divide an application into several modules
  - Independent, interchangeable modules
  - Plug and play idea
  - Increase the flexibility, maintainability, and reusability of FSW
  - [illegible]ions can b[illegible] futu[illegible]s



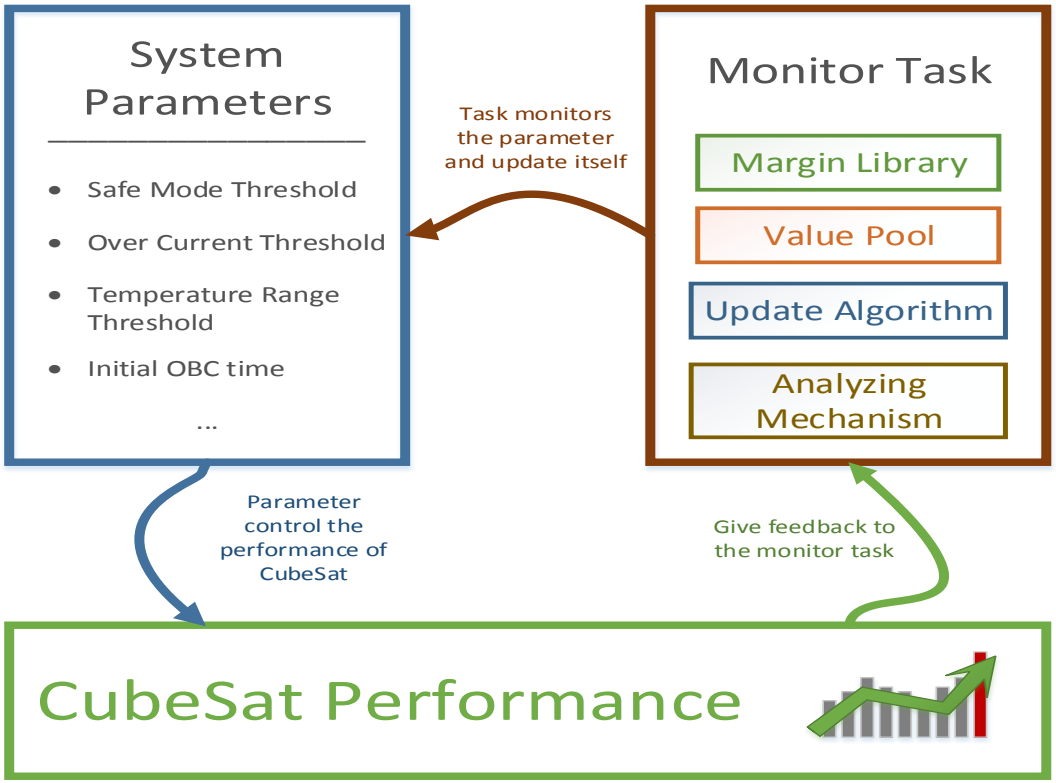
Resilient Software Design
- Redundancy Memory -

- SD card partition – /sd0 & /sd1
- Crippled mode
  - Backup solution
  - Use flash memory instead
  - Limited storage place



Resilient Software Design
- Self Configurable Architecture -

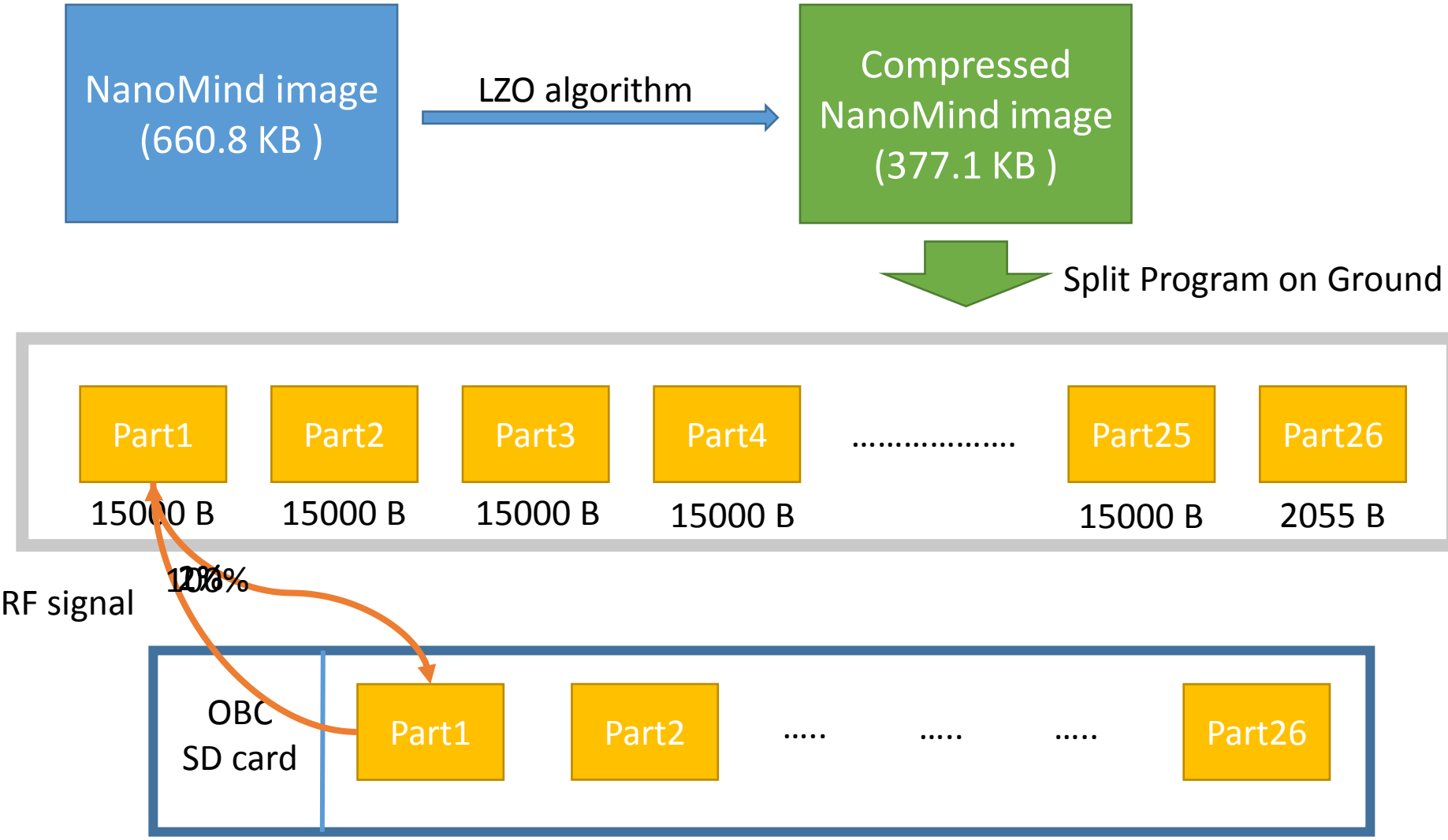
- The behavior of the FSW is influenced by some system parameters.
- Tune/adjust system parameters depending on the situation.



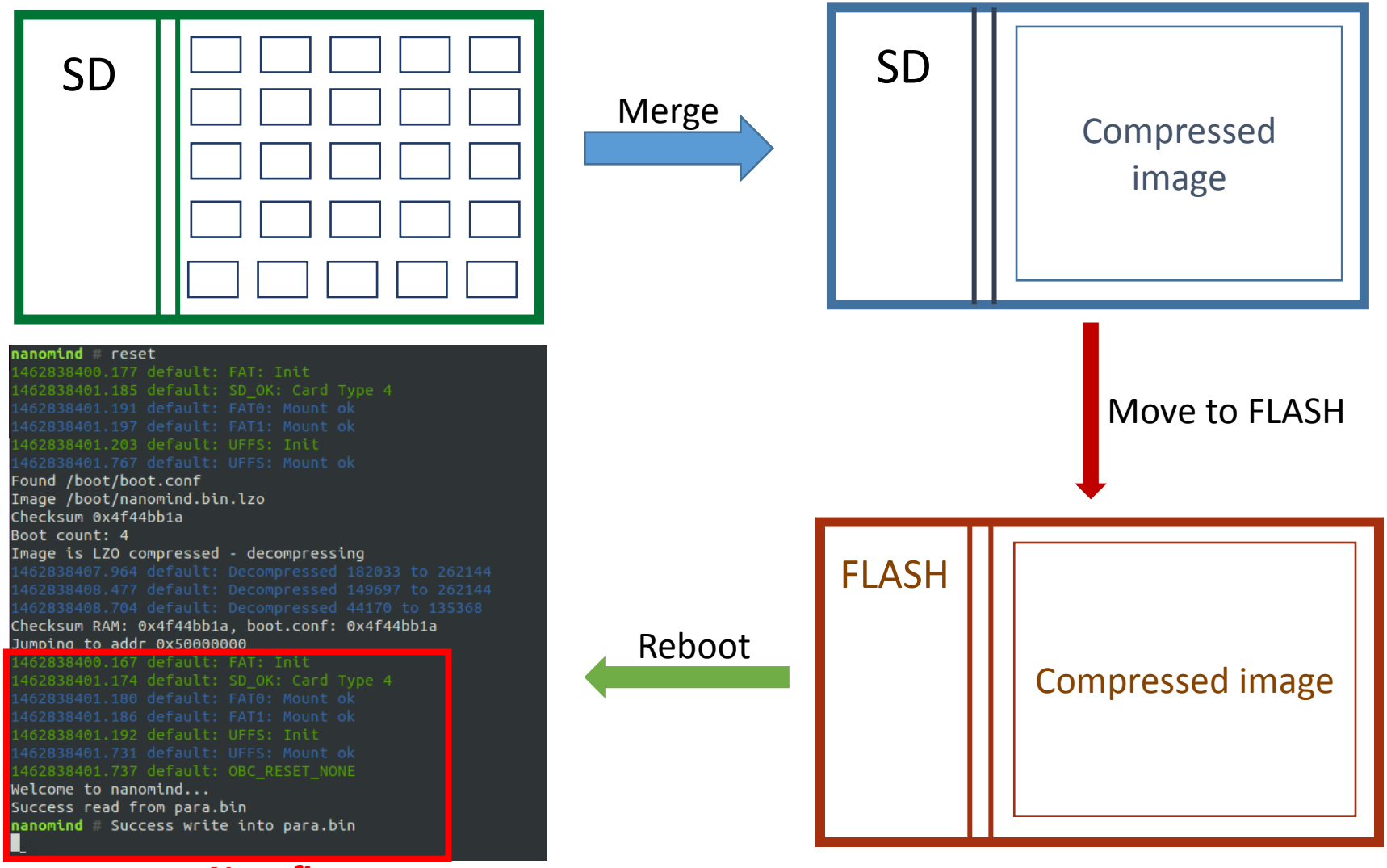
Resilient Software Design
- In Orbit Software Update -

- Some unpredictable situation occur in space, it may cause the fatal error in FSW.
- In case that some major issues happened in flight software, a backup solution is proposed by uploading new firmware on OBC.
- All the step must be done by telecommand from GS.

Resilient Software Design
- In Orbit Software Update -



Resilient Software Design
- In Orbit Software Update -



Resilient Software Design
- In Orbit Software Update -

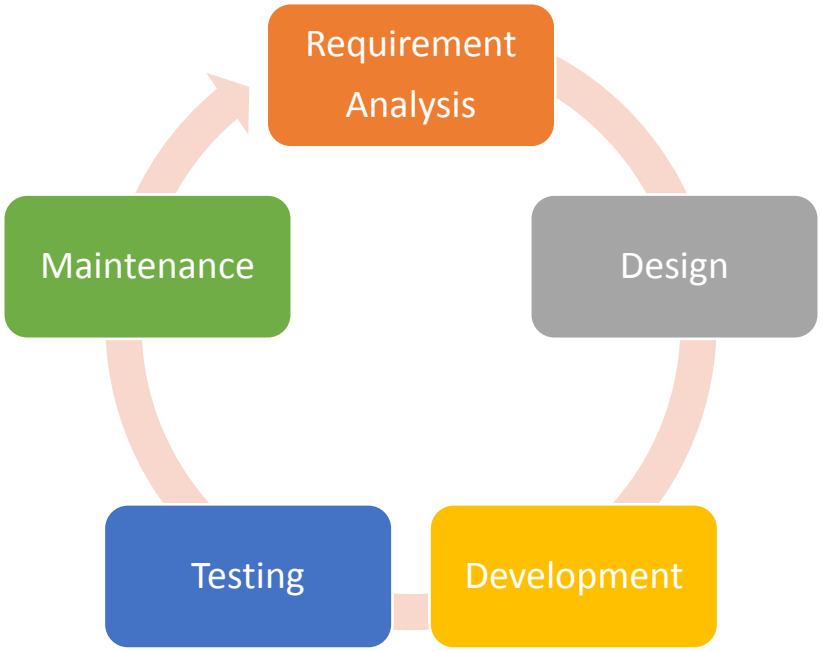
- Results:
  - each part will be uploaded within 180 seconds
  - More than 20 contacts are needed.
  - Successfully update the flight software all by GS

Outline

- Introduction
- PHOENIX Nanosatellite
- Flight Software Design
- Resilient Software Design
- **Testing and Verification**
- Conclusion

Testing and Verification

- Several tools are used to conduct the testing campaign and verification process.
  - Discover errors or defects in FSW.
  - Verify that if the design meets the requirements.
  - Evaluate the performance of FSW.

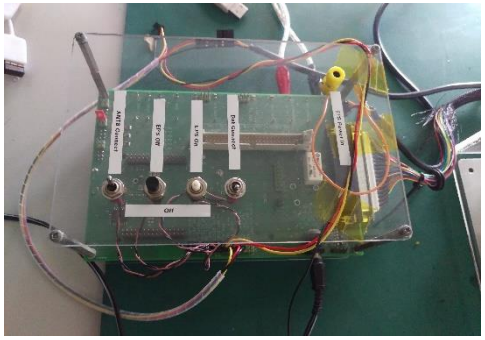


Testing and Verification
- Test Configuration Setup -

- Software Debug Interface
  - Self defined command
  - GOSH interface

```
finalCheck          PHOENIX: finalCheck
gpstest             PHOENIX: gpstest
ch                  PHOENIX: ch [ON(1), OFF(0)]
pc                  PHOENIX: pc [sub] [ON(1), OFF(0)]
epss                PHOENIX: epss []
ff                  PHOENIX: first flight switch
mSFD                PHOENIX: mSFD
ic                  PHOENIX: ic
isn                 PHOENIX: isn [buffer]
mcs                 PHOENIX: mcs [ON = 1 / OFF = 0]
inmsR               PHOENIX: inms
inms                PHOENIX: inms <cmd1> <cmd2> <cmd3> <cmd4> ....
i2c                 PHOENIX: i2c <node> <rx>  will have <para> *N byte ?
inmss               PHOENIX: inmss [ON = 1 / OFF = 0]
cm                  PHOENIX: cm
adcss               PHOENIX: adcss [ON = 1 / OFF = 0]
seuvs               PHOENIX: seuvs [ON = 1 / OFF = 0]
tele                PHOENIX: tele [ON = 1 / OFF = 0]
jt                  PHOENIX: jt [sec]
ir                  PHOENIX: inms script read
ct                  PHOENIX: simulate receiving a uplink command and execute it
T_test              PHOENIX: Activate/OFF Thermal Task,switch 1=on, 0 =off
shutdown_tm         PHOENIX: change transceiver standby mode
parawrite           PHOENIX: write para setting in FS
pararead            PHOENIX: read on board parameter setting in FS
paradelete          PHOENIX: delete parameters.bin
T_data_del          PHOENIX: delete t_obc.bin t_inms.bin
jump_mode           PHOENIX: jump_mode [mode] // 0=safe mode, 2=adcs mode,3=payload mode
idleunlock          PHOENIX: skip idle 30m step
testmode            PHOENIX: enter  testmode, please reboot satellite if want to leave this mode
seuvwrite           PHOENIX: configure SEUV
seuvread            PHOENIX: Take data from a configured Channel
comhk               PHOENIX: retrive com hk data
comhk2              PHOENIX: retrive com transmitter state
```

- Tools
  - EGSE – ISIS Generic Interface System
  - I²C protocol analyzer
  - STM32F429 Discovery Board



Testing and Verification
- Test Campaign -

- Reference Functional Test
  - Basic Functional Test
- Environmental Test
  - Data Recording
- End to End Communication Test
  - Uplink large file, downlink large file
- 32 Hour Test
  - Continues operation, simulate contacting, mode transition
- Anomaly Handling
  - Handle anomaly during the mission
- Mission Verification
  - Focus on operation of each mode
- Telecommand Test
  - All telecommands (169/169) have been implemented and tested.

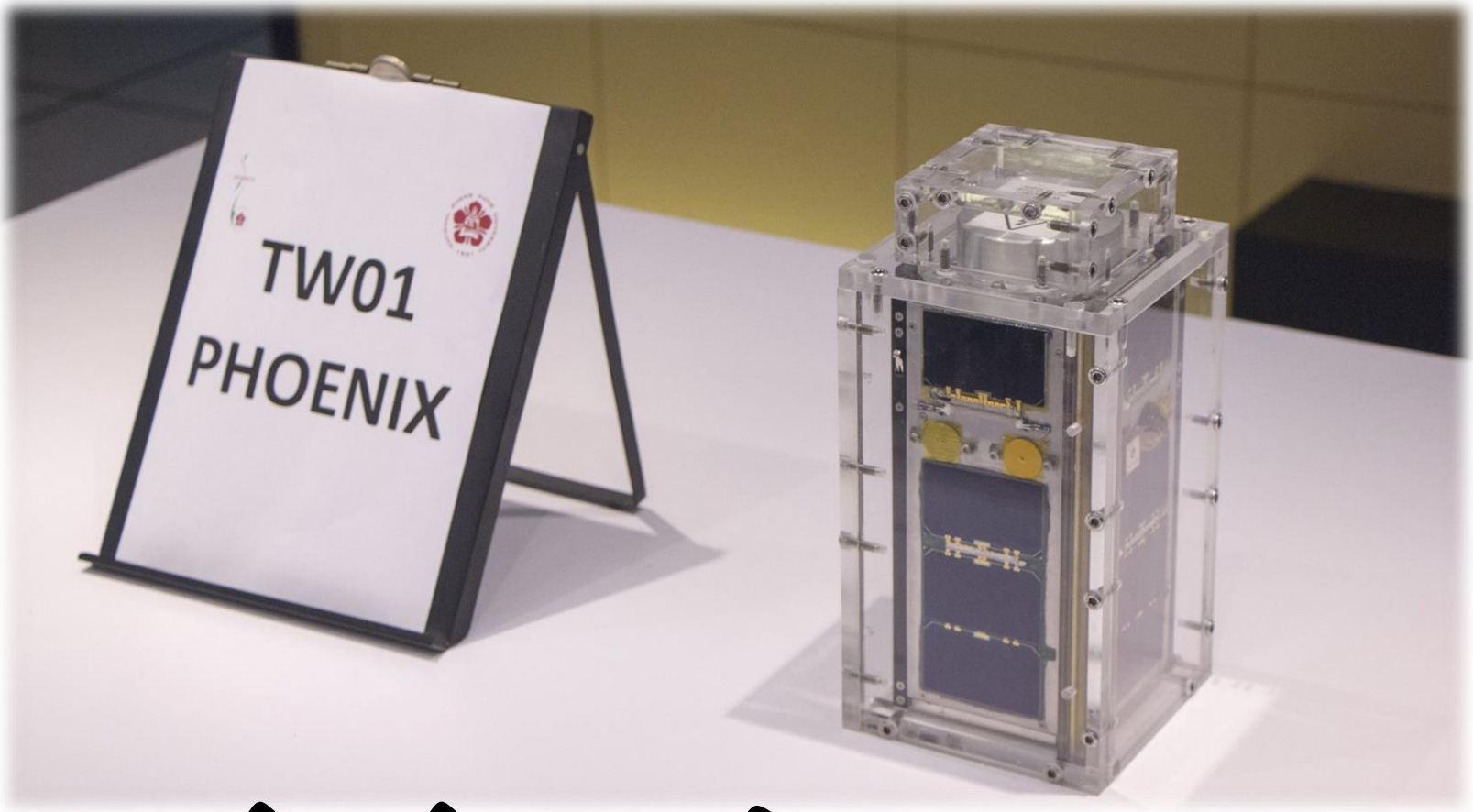
Outline

- Introduction
- PHOENIX Nanosatellite
- Flight Software Design
- Resilient Software Design
- Testing and Verification
- **Conclusion**

Conclusion

- Flight Software
  - A comprehensive software architecture is designed and implemented.
  - A few methods for increasing the reliability of FSW are proposed.

- Testing and Verification
  - Several testing campaigns are conducted and passed successfully.
  - Some anomaly are detected, and update the related action in FSW.



Thank You for attention !