command and data handling subsystem

Abhijit Pamarty and suraj s prakash

# introduction

This report seeks to develop a model for the command and data handling subsystem, and lays out design considerations and component selection for the same. The intention of writing this report is to educate current members on all the intricacies of the subsystem, as well as to familiarize future members who will take part in this project.

# Design considerations

## Resilience

Resilience, as applied to any autonomous machine functioning in a hazardous environment where maintenance is impossible (in this case, deep space), implies an ability to recover from catastrophic breakdowns by giving the system work-arounds at the time of development.

## Reliability

Reliability implies the system is capable of self-diagnosing critical errors and redundant circuitry, which, at the point of failure, allows the system to prevent any mission interruptions needing operator intervention.

As a general rule of thumb, resiliency is easier to implement in a system as compared to reliability in terms of developmental costs.

## Single points of failure

A single point of failure is a part of a system that, if it fails, will stop the entire system from functioning. Systems can be made robust by adding multiple redundancies at every single point of failure. In general, the lesser number of SPoFs a system has, the lesser the chance of a breakdown causing an early mission abortion.

In continuing sections, a few SPoFs have been identified, and a few ideas have been put forth to increase resiliency and prevent mission failures.

## Mechanical resilience

As a satellite is expected to face high G-forces at launch, as well as rotational accelerations and periods of intense vibrations, it is imperative that the electrical connections are made with lockable connectors and loose cabling is avoided to as much of an extent as possible.   
  
Extreme temperatures (-200°C to +150°C) encountered in space also demand that the controller electronics be stable even in wide fluctuations in temperature. Fortunately, small satellites in low earth orbits stay in a lower range of temperatures, allowing the use of standard COTS components.

## Electronic resilience

The biggest challenge in CubeSat resilience comes from cosmic particles. Collisions with electronic circuits can create problems which last through the entirety of the mission. A memory cell may be bit-flipped, or stuck in a state permanently.

While radiation hardening is possible, the cost of buying radiation hardened electronics goes above budget. It then becomes a necessity that other methods be used in order to make the CubeSat recoverable in case of failure.

## processor selection

The microcontroller chip of the CubeSat is an obvious single-point-of-failure. Keeping the budget in mind, it is suggested that the microcontroller needn’t be space-qualified. This shortens the mission duration, but at the same time increases the chance of project completion.

An 8-bit MCU, while cheap, is unlikely to have enough processing power to perform all processes required for optimal performance. A multi-core, very-high-performance chip isn’t recommended either, as the smaller circuit elements reduce electronic resilience.   
  
It is also required that the MCU be low-power, to prevent catastrophic mission failure from battery depletion. Since it is likely that the peripherals of the CubeSat will use SPI, I2C or UART communication protocols, the microcontroller must have all three capabilities.

To avoid the creation of more single points of failure, it is also recommended that each peripheral be given its own separate channel, in the case of I2C or SPI, which can handle multiple peripherals on a single bus.

## software resilience

While most microcontrollers have in-built watchdog timers to force a reset if the processor clock stops, additional infrastructure will be required in the case of system crashes where the processor clock doesn’t stop. In these cases, a system where an external supervisor device requests a programmed “heartbeat pulse”, where the device will force a reset if it does not receive one, can be implemented.

In cases where a sensor or other peripheral becomes unresponsive and the microcontroller gets stuck in a loop repeatedly requesting data, but still sends the heartbeat pulse to the supervisor device, additional infrastructure will need to be put in place to reset the microcontroller.

A timer chip which forced a reset every 25 hours was implemented in *Dellingr*, regardless of system functioning and health. A similar system can be put in place, with additional housekeeping data being sent to the ground station, where an operator can shut down or request a reset of sensors in the case of failure.

It might also be possible that multiple crashes occur one after another, in the case of which a system reset might not fix the problem. In this situation, the satellite must recognise that there is a firmware bug, and wait for further instruction.   
  
However, these errors have the highest amount of likelihood of ending the mission, as the radio-in window for most CubeSats is in the order of a few minutes, making large firmware updates impossible.

## routines

The command and data handling subsystem will most likely require multiple modes of operation, with each mode corresponding to a specific function or action.

The main functions which require controller supervision or input are listed as follows:

* Solar panel deployment
* Radio antenna deployment
* Sensor activation and data collection
* Scientific data transmission
* Housekeeping data transmission
* EPS supervision
* “Listening mode”, where the satellite listens for instructions, in the case of functional failure or other special circumstances
* ACDS control
* Data storage

In addition to these functions, there is also the need for a “Safe mode”, to which the OBC can switch to in the case of low power, or other peripheral failure. The safe operations mode will reroute power from the peripherals to essential systems, in order to ensure that critical systems don’t fail in the case of an emergency.

For timing abstraction, maintainability, modularity, improved efficiency and code reuse, it has been decided that an RTOS will be used. At the time of writing this document, FreeRTOS running on an STM32 chip has been decided.

## rough roadmap [not to be in the final document]

Buy nucleo board -> mess around with nucleo board-> use FreeRTOS on nucleo board-> Write program architecture -> write program in stm32cubeide -> suraj writes program using makefiles, abhijit does pcb design of OBC, heartbeat pulse monitor and analog reset timer

## obc hardware

# Sources

Dellingr: Reliability lessons learned from on-orbit, Larry Kepko, Luis Santos, Chuck Clagett, Behnam Azimi, Dean Chai, Alan Cudmore, Scott Starin, James Marshall, John Lucas

Radiation Effects on Electronics 101: Simple Concepts and New Challenges, Kenneth A. LaBel, NEPP, NASA