0.0.1 Flight Software Architecture

Many Cubesat missions choose a layered architecture for the onboard software development [1][2][3][4][5]. This layered structure is described by Johl and Lightsey et al in the document "a reusable command and data handling system for university cubesat missions" [6]. Johl and Lightsey devided the onboard architecture into four layers. In common with the rest of the architecture layer description in other documents, the layer have a top down approach. The lower layers have closer relationship with the onboard hardware; The higher layer is close to application that controls the logic on the flight software. The four layers that Johl and Lightsey proposed are listed from bottom as below:

- Device Interface Layer
  Johl and Lightsey proposed that the bottom of the architecture to be the device interface layer. Device interface layer contain the device drivers that implement the protocols for communication between the OBC and other spacecraft hardware. These device drives enable the upper layers uses communication protocols such as I2C, SPI, and GPIO as needed. A similar architecture is proposed by Mitchel, C et al from KySat-2 [5]. However, in the book "Onboard Computers, Onboard Software and Satellite Operations" [7], Eickhoff named this layer "operating system ad hardware interface layer". He proposed to also include the BIOS and the realtime operating system in the layer. This is more similar to the library structure that ISIS provided for MIST mission.

- High-Level Subsystem Interface Layer
  This layer contains the components that are used to provide the functionality from various onboard subsystems and devices of the spacecraft. The complete functionality of the subsystems and devices should be contained in this layer with appropiate name for usability. The examples including COM function, EPS functions, and payload functions. In the KySat-2 documentation, Mitchel,C et all composed this layer with similar API to access external support devices on the software level. In the KySat-2 mission, this layer and the bottom layer are composed as hardware access layer.

- Software Process Layer
  Johl and Lightsey briefly mentioned software process layer in the document. The authors proposed this layer to provide the processes and the means for storage of mission data, health data, and generated command and error message. The components shares similarity to the KySat architecture. However, in the KySat documentation, it is proposed as distributed kernel layer. The primary feature of the distributed kernel layer is to monitor the subsystem modules. Besides the monitor feature, the error handling, ground command interface are also mentioned as features on the distributed kernel layer.

- Application Layer
  Most documents unanomously named the top layer of the flight software application layer [1][2][3][4][5][6][7]. The application layer includes the main flight software application that controls the subsystems and the payload to implement the mission [5][7]. Due to the variaty of payload and subsystem of each mission, this layer is unique for every mission [6].

0.0.2 Onboard Tasks and Scheduling

A task on a RTOS is a small program in its own right [8]. It is a basic building block of RTOS software [9]. Many onboard software have a centralized task structure [4][10].

Dabrowski from the ION satellite mission describes the central piece of software as application manager [11]. This is considered a robust and intelligent version of the scheduler. On the ION satellite mission, the application manager is able to start and stop applications as specified in configuration file. It also performs system recovery. On AAU-cubesat however, the centralized task is refered as the supervisor [12]. The supervisor performs as the organizer of the satellite software system. The functionality of the supervisor is routing information between other tasks and ensure the healthiness of the mission.

As stated in the Chapter 0.0.1, application layer have various components that controls the onboard subsystems and payloads. For each application component, AAU-cubesat also created the coresponding task or tasks. For example, a specilized task PCU(power control unit) is created to control the power hardware and level of activity. As another example for this matter, the CubeSat mission implemented a COM task for communication downlink from satellite to earth, and another

task COMRX for the communication from the earth to satellite.

The scheduling onboard can be handled by a task as the ION satellite [11]. It can also be handled with a scheduling cycle as the CryoSat mission [7]. The onboard cycle is devided into different subcycles, each with a time duration. A certain set of software managers or handlers are triggered during a subcycle. KySat-2 performs a different task scheduling due to the implementation of non-preemptive RTOS [5]. On KySat-2, a watchdog timer(WDT) is implemented. WDT is a piece of hardware that sends a reset signal to the CPU after a preset period of time. The software needs to perform a kick subroutine to reset the timer to prevent the reseting action. In KySat-2 mission, each task meets the requirement of a critical section no longer than the watch dog timer(WDT) period. The WDT kick subroutine can only be called by the scheduler, thus prevents any hung task that occupies the CPU.

0.0.3 I2C interface

References

[1] Adnane Addaim, Abdelhaq Kherras, and El Bachir Zantou. *Design of Low-cost Telecommunications CubeSat-class Spacecraft*. INTECH Open Access Publisher, 2010.

[2] Artur Scholz. *Command and data handling system design for the COMPASS-1 Picosatellite*. PhD thesis, University of Applied Sciences Aachen.

[3] California Polytechnic State University-San Luis Obispo Greg D. Manyak. *Fault Tolerant and Flexible CubeSat Software Architecture*. PhD thesis, California Polytechnic State University, San Luis Obispo, 2011.

[4] S F Sabri, S S Yuhaniz, and K Kamardin. *Designing a Low Cost CubeSat's Command And Data Handling Subsystem Kit*. PhD thesis, 2006.

[5] C Mitchell, J Rexroat, S A Rawashdeh, and J Lumpp. Development of a modular command and data handling architecture for the KySat-2 CubeSat. In *2014 IEEE Aerospace Conference*, pages 1–11, March 2014.

[6] S Johl, E Glenn Lightsey, S M Horton, and G R Anandayuvaraj. A reusable command and data handling system for university cubesat missions. In *2014 IEEE Aerospace Conference*, pages 1–13, March 2014.

[7] Jens Eickhoff. *Onboard Computers, Onboard Software and Satellite Operations: An Introduction*. Springer Aerospace Technology. Springer Berlin Heidelberg, 2012.

[8] Richard Barry. *Mastering the FreeRTOS^{TM} Real Time Kernel, A Hands-On Tutorial Guide*. Real Time Engineers Ltd, 2016.

[9] Tasks and states - RTOS. `https://sites.google.com/site/rtosmifmim/home/tasks-and-states`. Accessed: 2017-5-27.

[10] D Schor, J Scowcroft, C Nichols, and W Kinsner. A command and data handling unit for pico-satellite missions. In *2009 Canadian Conference on Electrical and Computer Engineering*, pages 874–879, May 2009.

[11] Michael J Dabrowski. *The design of a software system for a small space satellite*. PhD thesis, University of Illinois at Urbana-Champaign, 2003.

[12] Aalborg University. *Documentation of AAU-Cubesat On Board Computer Software*, 2002.

1 Appendix

Observera. Nedanstående rubriker är förslag på bilagor som kommer att ingå i den färdiga rapporten. Författarna skriver rapporten i Latex och har ännu inte hittat ett ordentligt sätt att ge en bifogad PDF en korrekt rubrik, därför blir det en tom sida innan en bifogad PDF. De bifogade PDF som finns i dagsläget är uppdragsbeskrivning och projektdefinition.

Appendix A Kravdokument

Appendix B Arkitekturdokument

Appendix C Clickstreams

Appendix D Uppdragsbeskrivning

Appendix E Projektdefinition