ONBOARD DATA HANDLING FOR CUBE SATELLITE

by

Christopher Bechard

Joyce Lee

Steven Terryberry

(advisor: Dr. Jayshri Sabarinathan)

ECE 4415 Computer Engineering Design Project

Final Report

Submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Engineering Science

Department of Electrical and Computer Engineering
Western University
London, Ontario, Canada

April 15, 2019

Abstract

The onboard data handling system is the central control system in a cube satellite which is responsible for processing and relaying information between the subsystems. It is essential in ensuring the operation of the cube satellite, as well as achieving the cube satellite's research purpose which is to capture and send VR images to the ground station. There are two main deliverables in this project: functional design and research documents that will be carried forward in the cube satellite project and a working prototype which will demonstrate the computer system's essential functionality. A V-model engineering methodology was employed due to the collaboration required with other subsystem teams. The Endurosat Onboard Computer with FreeRTOS was chosen as the final recommendation for the cube satellite moving forward. This choice is validated through the design and testing of a prototype built using the Nucleo-F429ZI which shares the same ARM Cortex M4 processor as the Endurosat Onboard Computer. The completion of this project gives confidence to the success of Western University's cube satellite project which will be launched and deployed from the International Space Station in 2022 in partnership with Nunavut Arctic College and the Canadian Space Agency.

Contribution of the Team Members

Each team member was responsible for attending weekly meetings with the project manager and other multidisciplinary teams, providing updates, collaborating and communicating with other subsystem teams to ensure compatibility, and working on assigned individual tasks that are decided on by the team.

Joyce is a fourth year computer engineering student specializing in electronic devices. She completed a year long internship at Shopify where she worked as a solutions engineering intern as well as a software developer intern. Joyce was responsible for collecting and maintaining the list of known data interfaces from the other subsystem teams, guiding the team through the V-model engineering methodology, as well as developing the prototype concept and testing and validation strategy.

Chris is a fourth year computer engineering student specializing in software systems. Chris has a deep knowledge of embedded systems was responsible for conducting detailed research into the hardware requirements of the project, and assessing the requirements against the possible C&DH hardware options. Chris also selected the prototype hardware based on his findings and recommended the open-source software tools used in the project.

Steven is a fourth year computer engineering student specializing in software systems. Steven was responsible for conducting detailed research into operating systems.

The following is a list of the tasks completed during the project. The original timeline was created with an approximate 30 week (~8 month) timeline in mind. This timeline and the team's progress is illustrated in the Gantt Chart in the Appendices.

Task	Duration (Weeks)
Research Phase - Acquire data from previous research teams (transition) - Research strengths and weaknesses of existing CubeSat computer systems - Understand challenges faced by previous research teams - Define the scope of project	3

Exploration and Design Phase
- Define objectives and constraints
- Identify margins for power budgeting
- Identify considerations/connections required to interface with other subsystems (mounting, power, mass etc)
  - Specify a communication interface for transmitting digital application data from satellite to ground station
  - Specify application data interfaces for each subsystem
  - Specify power interfaces for each subsystem
  - Telecommand interface development
  - Data budgeting
- Mid-term progress report
	6

Build Phase - Assess feasibility of building prototype on Endurosat board. Otherwise, make arrangements to prototype on the alternative C&DH board - Identify criteria for success and create tests for each subsystem component which will be used during the testing phase - Identify methods to demonstrate the system (prototype hardware, testing simulations) - Prioritize subsystem tasks that rank critical functions of the computer in relation to the overall mission of the CubeSat (ie. memory preservation, transmitting real-time data, attitude determination and control) - Request and purchase prototype parts - Application modelling for essential functionality (the exact data formats/messages from each of the subsystems may be "black boxes") - Confirm mass budget for structural team (with basic mounting details) - Confirm thermal budgeting for electrical and structural teams	10
Testing and Iteration Phase - Test each individual subsystem component and identify successes and failures - Make adjustments to fix failures or make recommendations for future teams based on lessons learned from the failures - Demonstrate essential research functionality with prototype	8
Project Deliverables and Presentations - Design demonstration - Design presentation - Western CPSX Space Day presentation - Final report and reflection	3

Acknowledgements

This project was made possible by the faculty, staff and postdoctoral students from the Centre for Planetary Science and Exploration (CPSX) at Western University. In particular, Dr. Jayshri Sabarinathan (Principle Investigator and Capstone Project Advisor), Dr. Matthew Cross (Project Manager), Dr. Matthew Bourassa (Deputy Project Manager), Dr. Ken McIsaac (Co-Investigator), Alexis Pascual (Graduate Student Advisor), and Kelsey Doerksen (Graduate Student Advisor) have been instrumental to the successful organization of 10 engineering capstone teams across 5 different disciplines (electrical, computer, software, mechatronics, mechanical) who worked collaboratively on the subcomponents of the cube satellite project. Their perspective, guidance and feedback were essential to the success of this project. This project was funded by the Canadian Space Agency who also sponsored the space engineering and science education of participating students through educational resources and webinars. It has been a privilege to participate in this experiential learning experience and contribute to Western University's first cube satellite research project.

Nomenclature

No specific symbols requiring explanation were used in this report. All technical terms beyond the scope of general knowledge are defined in the glossary or the context mentioned.

Glossary

ADCS	Attitude Determination and Control System
C&DH	Command and Data Handling
CAN	Controller Area Network
COTS	Consumer Off-the-shelf
EPS	Electric Power System
I²C	Inter-Integrated Circuit
IoT	Internet of Things
IPC	Inter-Process Communication
ISR	Interrupt Service Routine
LVDS	Low-Voltage Differential Signalling serial protocol
MCU	Multipoint Control Unit
OBC	Onboard Computer
PAP	Password Authentication Protocol
PPP	Point-to-Point Protocol
RTOS	Real-Time Operating System
RS-xxx	Recommended Standards, equivalent to ANSI/EIA/TIA specifications
S-band	2 to 4 Ghz frequency band
SBC	Single Board Computer
SNR	Signal to noise ratio
SPI	Serial Peripheral Interface
UHF	300 Mhz to 1 Ghz frequency band
USB	Universal Serial Bus

Table of Contents

List of Figures/Tables

1. Introduction/Background

1.1 Problem Statement

The cube satellite requires a central control system which can process and relay information between subsystems for safety and operation of the cube satellite, as well as to achieve the cube satellite's research purpose which is to communicate image and location data to the ground station. The preliminary hardware components chosen by the other subsystems are made by different manufacturers and use different hardware communication protocols which have different data rates and interrupts. An appropriate command and data handling (C&DH) board must be chosen which can handle the different types of serial data. The embedded operating system must react to interrupts given by the other subsystems by processing serial data and relaying relevant information to other subsystems.

1.2 Background Information/Detailed Literature Review

This literature review aims to compare and contrast the operating system, communication, and application design strategies of the cube satellite industry in the context of an academic project. First, the operating systems are evaluated on the merits of licensing, kernel behaviour, and ease of use or community support. Then the relevant communication protocols to this report will be compared on their features and context of use. Finally, the traditional master-slave is contrasted with distributed multiprocessor application architecture.

1.2.1 Operating Systems

An operating system must be utilized to provide software functionality to the hardware device. There are a wide variety of viable OS choices, of which the most appropriate will be discussed. Windows and VxWorks are proprietary software, while FreeRTOS, Linux, and KubOS are open source. The industry is showing a general trend towards more open source, with an increase in market share and energized community interest [1]. VxWorks and FreeRTOS are inherently RTOS, while Windows, Linux, and KubOS can be modified to incorporate real-time functionality.

Windows uses a micro kernel setup, which when compared to the kernel of Linux, takes less space at the expense of being less efficient [2]. To employ this in a real-time, deterministic environment, Windows CE [3] can be utilized. A unique feature of Windows CE is a variable clock tick [4], which allows for interrupt ticks to be appropriately adjusted, improving operating system efficiency. Due to acceptable but not ideal latency time, it would be considered soft real-time [5]. A hard real-time system requires much stricter timing restrictions that Windows CE does not meet. In recent years, Microsoft has transitioned towards a Windows 10 IoT [3] in order to bring similar functionality to a product more closely connected to their mainline software. However, not all features are carried over, causing the need for third-party extensions to maintain RTOS requirements.

Linux offers more customization and reliability than Windows, but is much more complex as a result [2]. It has a very large support base of hardware architecture [4], and through its widely adopted nature can be considered very reliable. However, it is limited only to 32-bit and 64-bit processors [6]. This ensures the system will be much more complex, and thus more difficult to design. By using a time-sharing scheduling algorithm to fairly distribute processing time, tasks may need to wait almost one second [7], making it incompatible with time sensitive situations. During operation, a kernel mode system call cannot be interrupted, forcing real-time processes to wait. Linux has the potential to be used as a RTOS through such modified versions including RTLinux. A real-time kernel runs the regular kernel, and the real-time kernel is in charge of interrupts ensuring that the regular kernel will not interrupt running processes unless it is deemed appropriate.

KubOS combines a modified Linux kernel with subsystem APIs and critical software services and protocols [8]. The product provides a similar experience to Linux, but with the addition of a framework to more easily utilize it for CubeSat launches as well as a tailored customer support service.

VxWorks has many similarities to RTLinux, but with an improved interrupt latency speed derived from using only one kernel instead of two [7]. Interrupt efficiency is improved through different mechanisms used by VxWorks. Firstly, ISR are kept in a special context eliminating the need for context switching. Second, when not running a mission critical task, interrupts will cause the current task and subsequent lower priority interrupts to be delegated to a work queue that immediately resumes the task after the interrupt finishes.

FreeRTOS has support for much smaller processors than RTLinux, but as a tradeoff, has more limited processing power. While RTLinux provides more scalability, only FreeRTOS can be scaled down to utilize 8-bit and 16-bit processors [9]. Being smaller, it only consists of basic scheduling, IPC, and synchronization semaphores [9], while RTLinux provides the much greater functionality that comes with Linux. Task scheduling is available in RTOS, with FreeRTOS specifically using either cooperative or preemptive scheduling. Both options make use of common scheduling features such as tasks, queues, and priorities [10], making them very similar but with one important difference. Cooperative scheduling will run tasks until they explicitly give back control [10], while preemptive scheduling introduces interrupts that allow for control to be forcibly taken back. Preemptive scheduling can be utilized under certain circumstances to ensure that mission critical issues will be much more quickly addressed. However, it comes at the cost of increased overhead due to processes switching between running and ready states [11].

1.2.2 Communications

Transceivers	
Device	Description
UART	Universal asynchronous transmitter/receiver. Baud rates for these are variable, but often limit data rates far below the theoretical limits of the cabling from SNR alone. Modern standard data rates include 9600, 14400, 19200, 38400, 57600, 115200, 128000 and 256000 baud.
USART	Universal synchronous/asynchronous transmitter/receiver. A UART with an extra line for a clock, enabling a higher baud rate from clock synchronization such as with SPI. The clock line may be used in asynchronous serial specifications to enable additional features such as flow control. Relevant examples include: RS-232 (CTS - clear to send on Rx, RTS - request to send on Tx), and RS-485 (DE - driver enable).

Table 1. *A comparison of UART and USART Transceivers. CAN and USB transceivers have been excluded.*

Cable Specifications (Electrical and SNR Excluded)				
Specification	Transmitters	Receivers or Transceivers	Clocked (Synchronous)	Direction
I²C [12]	Multi-master	7-bit or 10-bit address with practical limitations	True	Half duplex
SPI	Single master	Single slave (or multiple slaves each with a separate select)	True	Full duplex
CAN	Multi-master	Practical limit (data rate is inversely proportional to number of nodes)	False (higher layer arbitration defined)	Half duplex

RS-232	Single master	Single slave	False	Full duplex
RS-422	Single master	10	False	Full duplex Half duplex
RS-485	Multi-master	32	False	Half duplex
USB [13]	Single master	Single slave	False (higher layer arbitration defined)	Half duplex (1.x, 2.0, 3.x) Full duplex (3.x)

Table 2. *A comparison of network capabilities for a relevant subset of cabling specifications.*

Data link, Network, and Transport Protocols

Aside from the physical layer specifications mentioned above, cube satellites may use different data link layer protocols and above to improve performance of the network or point-to-point connections. The relevant protocol design literature may be explored with three contexts: one, for a packet radio connection, two, for a method of operating peripherals, and last, in the context of a distributed application on a network.

First, a protocol for a packet radio connection, in the context of a cube satellite, is often a point-to-point connection. Another possible type of connection would be a point-to-multipoint connection, where, for example, multiple ground stations share overlapping access to a single satellite. A point-to-multipoint connection may require time division (TDMA) or carrier sensing (CSMA/CA) protocols to share a single band. Additionally, the popularity of amateur radio makes software and hardware using the AX.25 protocol easy to acquire and use, making this an attractive option to some cube satellite projects with these requirements operating in amateur radio bands. However, with a serial or point-to-point connection, the address and access management information may be removed. This leads to protocols such as PPP with PAP or KISS. These protocols aim to contain the least information possible as they only have a single destination with no extra information required. There is some variance in features such as forward error correction, authentication, and flow control. There is a space domain specific serial protocol, SpaceWire, with many features, however at this time it is not applicable to the project.

Overall, the protocols familiar for their usage in terrestrial networks are often more than what is necessary for a packet radio connection with a single satellite.

Second, the data link layer protocols for the peripherals may be similar or identical to those used for the satellite to ground station link, with the exception that there may be more value in a protocol that is not connectionless or just point-to-point. CAN, I²C, and USB define multiple layers of the OSI model above physical, so may be excluded. This leaves SPI, RS-232, RS-422, and RS-485 defined only as a physical layer. The data link layer protocols for each of these, in the case of a cube satellite, will often be defined by the peripheral vendor. For RS-422, one might want to ensure that the higher level protocols can address multiple receivers. For RS-485, one might want to negotiate the driver of the line, such as a CSMA or TDMA protocol.

Last, as will be explored in the following section, the master-slave architecture such as with a peripheral is not the only model for communicating between subsystems across multiple processors. In this case, one may wish to use an embedded TCP/IP implementation, such as uIP, lwIP, or CSP. The ideal physical layer for these communications is typically the CAN bus, which is present on many COTS components.

1.2.3 Application Architecture

Common application architectures must be considered. The client-server model is often used in distributed systems, where multiple independent entities are connected through a network [23]. Depending on the system, the client and server will have different amounts of responsibility. Generally, the client will initiate a request and display the selected data, while the server responds to concurrent requests and transmits appropriate responses [24]. In contrast, the master-slave model is frequently used in embedded systems. Slave processes are responsible for the collection of data, while the master process communicates with external elements, calculates collected data, and coordinates all subordinate slave processes [23]. This paradigm is useful for real-time systems since the master must request packet transmission from each slave [25]. This

ensures that transmission will always occur in a predictable manner determined by centralized processing.

1.3 Project Objectives

The objective of this project is to design a computer system which will be responsible for processing image data captured by the cube satellite camera, communicate with the ground station, and interface with the components designed by the electrical power, communications and structural teams. The ADCS subsystem is currently being prototyped as a separate project, but in the future the ADCS subsystem will be implemented on the C&DH board. Two methods of communication are currently being investigated by the other multidisciplinary teams: S-Band (spirit) and UHF (opportunity). The onboard computing system will need to be compatible with both systems. This will require selecting a piece of hardware and a compatible operating system that can withstand space conditions and have the computing power necessary to buffer and transmit compressed images.

This project is the first of three computer capstone projects which will culminate in a final design which will be launched into orbit in 2022. This project will define the preliminary design of the computer system onboard the CubeSat. The goal for this phase of the computer system cube satellite project is to produce two deliverables: functional design and research documents that will be used for future projects and a working prototype which will demonstrate the computer system's essential functionality.

The more specific goals of this year's preliminary design of the cube satellite's onboard computer system are as follows:

1. Demonstrate the suitability of a C&DH board for both S-band and UHF designs
2. Evaluate an operating system to meet project requirements
3. Evaluate software packages and tools that will ease development
4. Develop a testing and verification strategy for the prototype
5. Validate the hardware/software recommendations by building a prototype which will demonstrate the cube satellite's main research function

This project has two main deliverables: functional design and research documents that will be carried forward in the cube satellite project and a working prototype which will demonstrate the computer system's essential functionality. The scope of the project is reasonable considering the 8 month project timeline, technical skills of the team members, their ability to communicate progress and concerns with other subsystem teams, and the state of development of the computer subsystem in relation to other subsystems.

2. Design Approach

2.1 Concept Generation

First, to establish a rationale for concept generation, the current constraints of the project are listed under the following subheadings: functional subsystem interfaces and data interfaces.

The onboard computer must be able to support the data requirements for all subsystems, with possibly different options for Spirit and Opportunity if they have different system requirements. Although it may be possible to purchase a C&DH board which has the sufficient interfaces for all of these different protocols, each of these protocols have different interrupts and data rates which can present synchronization complications such as race conditions, deadlocks and resource starvation when accounting for operating system and application constraints.

2.1.1 Constraints

A. Subsystem Functional Overview

The functional diagram below outlines the basic functions of each subsystem and how they will interact with the onboard computer.

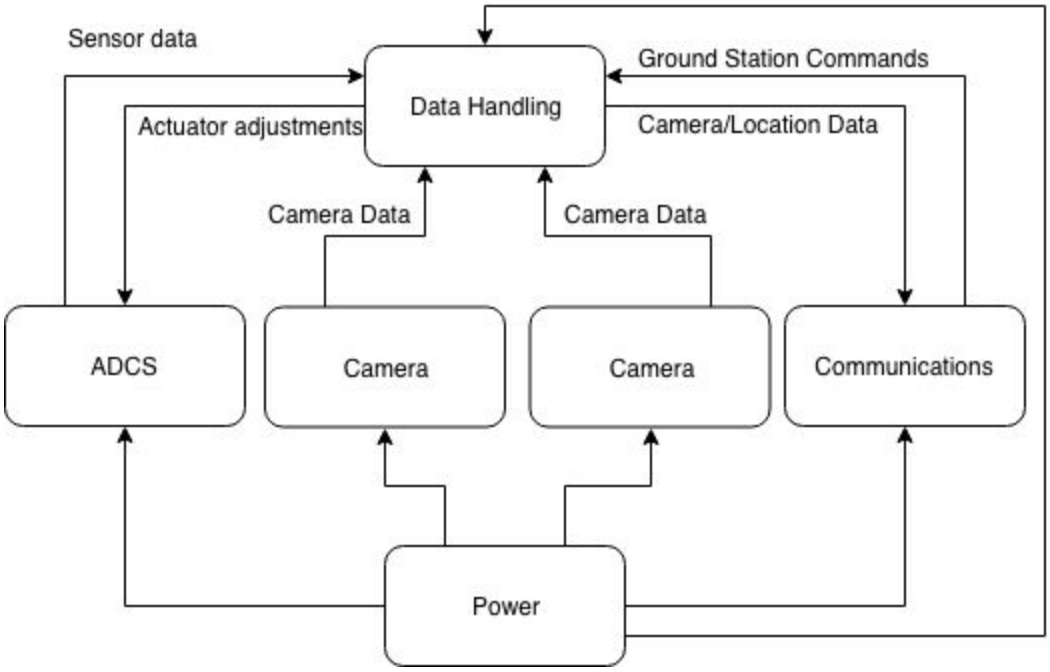


Figure 1. *A high level diagram describing the relationships between subsystems.*

B. Data Interfaces

Preliminary Component Choices From Other Subsystems				
Subsystem	Device Name	Vendor	Data interface(s)	Quantity
S-Band Communication (Spirit)	S-Band Patch Antenna	Endurosat [14]	N/A	1
	S-band Transceiver (Recent Choice)	Satlab [15]	CAN + RS422 (satellite bus), Ethernet	1

UHF Communication (Opportunity)	UHF Transceiver Type II	Endurosat [16]	UART, RS-485 (opt), I²C, CAN (opt)	1
Cameras	VR Camera	Canadensys [17]	RS-422, USB	2
ADCS	Sun Sensor (NCSS-SA05)	New Space [18]	Analog (5 channels)	3
	Precision Navigation & Pointing 3 axis Gyroscope Evaluation Board (400046-0300)	Silicon Sensing [19]	3x SPI	1
	Magnetorquer Rod (NCTR-M002)	NewSpace [20]	N/A (PWM)	3
	3-Axis Digital Magnetometer IC	Rohm [21]	I²C	1

Table 3. *A list of preliminary peripherals and SBCs from other subsystems.*

C. Canadensys Interface

The details of the camera interface are not completely clear. Connectionless operation of the cameras seems unlikely, therefore concept one will be used for the purposes of this report.

Concept 1: power, commands and responses Tx/Rx over USB, image data Rx over RS-422

Concept 2: power, image data Tx/Rx over usb, connectionless commands over RS-422

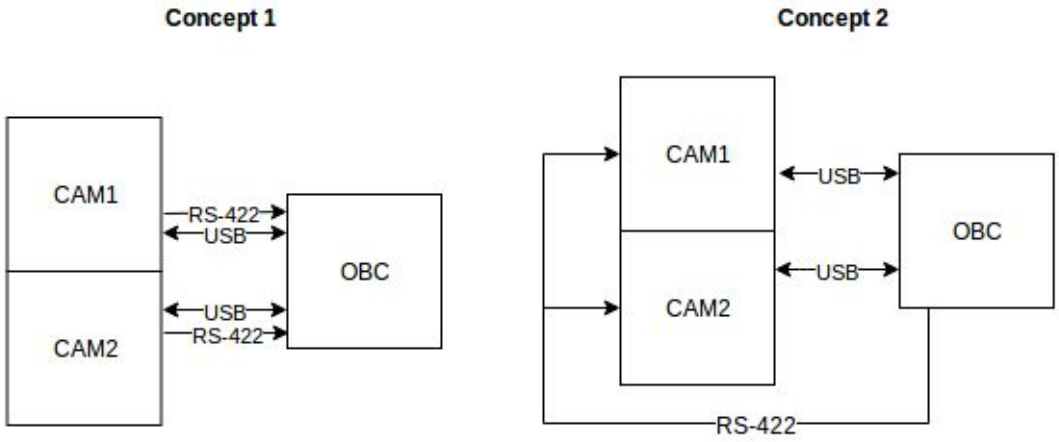


Figure 2. *Two possible interfacing configurations between the Canadensys camera and onboard computer subsystems, given the details provided by Canadensys.*

Given that there are several different systems that the onboard computer must interface with, there are two possible approaches to the hardware selection and design: selecting an onboard computer which has all the necessary interfaces to receive data in the board's default form then designing the operating system implementation around this, or unifying the communication protocols to simplify the operating system design.

The first approach is common in modern computers such as a laptop: an operating system accepts many different kinds of hardware inputs and has complex procedures which prevent concurrency problems such as race conditions and deadlocks. However, this may require complex operating system implementation that is beyond the skill level of this team. The

problem with the second approach is that if this requires additional hardware, the system may not fit in the design created by the structural design team. Also, it may not be possible as the interfaces on the hardware chosen by other groups cannot be changed.

There are two general approaches to developing an embedded operating system: adapting an existing operating system for an embedded application, and designing and implementing an operating system intended solely for embedded use. For this project we considered higher level operating systems that would allow us to write code using languages with more pre-built functionality such as Android, as well as FreeRTOS which would require lower level C programming.

2.1.2 Concepts

A. C&DH COTS Evaluation

Three options for space-rated onboard computers made by reputable manufacturers are shown below in Table 4.

	Available Interfaces	Quantity
Endurosat Cubesat Onboard Computer https://www.endurosat.com/products/cubesat-onboard-computer-obc/ A relevant expert of the Endurosat data sheet can be found in the Appendix. Cost: $3625 USD	I²C	2
	SPI	2
	USART	2
	UART	2
	USB	1
	CAN (optional)	1
	PWM	3 channels
ISIS Onboard Computer https://www.isispace.nl/product/on-board-computer/ Cost: $5046 USD	I²C	1
	SPI	8
	UART	2 (1x RS232 + 1x RS485/422/232)
	ADC	8 channel, 10 bit

	PWM	6
	GPIO Pins	Available
	USB (host or device)	1
	MicroSD	2x 2GB
CubeSpace Cube Computer https://www.cubesatshop.com/product/cube-computer/ Cost: $5161 USD	I²C	2
	CAN	1
	ADC	4*
	UART	1*
	SPI	1*
	PWM	4*
	MicroSD	2GB

* *"Piggyback" pins exist but an external board must implement the driving components necessary*

Table 4. *Options being considered for space-rated onboard computers made by reputable manufacturers.*

2.2 Concept Evaluation and Selection

Although a higher level operating system may allow us to develop more quickly, the memory and base processing power required to run a system like Android outweighed the benefits. In contrast, a completely custom operating system did not fit within the time constraint of the project, as we would need to account for extensive testing for reliability issues as well as the expected learning curve.

The performance of competing candidate concepts is evaluated by means of the appropriate methodology that includes higher-order estimates of the physical behavior, whenever possible orr applicable. The selection of the best and most viable concept is based on technically sound engineering principles.

During the research phase and design and exploration phase of the project, several options for C&DH hardware and operating systems were identified. Selecting the correct combination of hardware and software were extremely important to the success of the cube satellite project moving forward. In order to confidently defend the group's decision, the following two decision matrices were constructed. Each deciding factor is given a weight from 1 to 5 (5 being most important), and each vendor is given a score from 1 to 5 on how well it meets expectations for a given factor.

Decision Matrix for C&DH Hardware				
	Weight of Score	Endurosat	ISIS	CubeSpace
Contains necessary peripherals to connect to other subsystems	(5)	5	3	3
Contains pre-installed drivers for serial protocols	(3)	5	3	3
Ease of software development (contains pre-installed configurable OS)	(3)	5	5	3
Cost	(4)	5	3	3
Customer Support	(1)	5	4	4
Total Score	**80**	80	61	49

Figure 3. *Decision matrix for C&DH hardware*

Decision Matrix for Operating Systems						
	Weight of Score	Windows	VxWorks	Linux	KubOS	FreeRTOS
Scalability	(1)	3	3	2	2	4
Interrupt Latency	(3)	2	4	3	3	5
Complexity	(4)	3	4	1	2	4
Real-Time Capabilities	(5)	3	5	3	3	5
Memory	(2)	1	3	2	2	5
Cost	(4)	3	1	5	5	5
Customer Support	(1)	5	3	5	4	3
Total	**100**	55	69	59	62	93

Figure 4. *Decision matrix for C&DH Operating Systems*

The total weighted score from these two decision matrices show that the Endurosat cube computer with FreeRTOS is evidently the best choice for the project moving forward.

3. Design Analysis

3.1 Engineering Techniques/Software Tools

The project prioritized using proven open source or off-the-shelf software/hardware components. The team used GitHub as a version control tool to manage the development of the operating system. The team also contributed to the overall cube satellite project's GitHub Wiki which contains important information that other subsystem teams can reference, and also used Slack to communicate with each other as well as the other subsystem teams.

The team followed the V-model development life cycle engineering methodology [22], with an emphasis on top down requirements, system specification, development, and subsystem testing

with a plan for integration and deployment. This methodology was selected due to the collaboration required between subsystem teams. The requirements from each subsystem team is reliant on the others, so constant iteration and communication must be made, which the V-model development life cycle is well suited for. The V-model has four verification phases: requirement analysis, system design, architecture design and module design, followed by the coding phase. During the requirement analysis phase, the team researched the research goals of the cube satellite project and defined project objectives and criteria for success. During the system design phase, the peripherals of the other subsystems were gathered in order to proceed with the selection of the C&DH hardware. This phase was the longest phase, since there were many variables to be accounted for, and it was dependent on other subsystem teams who needed time researching and gathering requirements for their own projects prior to selecting their hardware components. Then during the architectural design phase, a high level design diagram of how the onboard computer would interact with other subsystems was created, along with a more detailed diagram which included the corresponding peripherals. These diagrams guided the module design phase in which more low level design occurred, which included the planning of operating system tasks.

During the module design phase and the coding phase, a prototype was developed which would validate the recommendations made by the team based on their work during the previous research phases. A validation and testing strategy was also developed during this phase to define the scope of the prototype. GitHub was used during the coding phase to allow multiple team members to work on different parts of the project at once without overwriting each other's work. GitHub provides version control and branch tools which make collaborating a smooth process. Furthermore, prior to merging one's work into the master branch, other team members will take time to review the code and look for potential bugs or flaws. Once the code is approved, the code

will be merged into the production code. The project concluded following this phase. System testing and acceptance testing requires the completion of other subsystem projects and is therefore out of the scope of this year's project.

3.2 Complete Analysis/Calculations

The design analysis for this portion of the project primarily involves validating that the hardware features, interfaces, and software platform could feasibly meet the known requirements.

The primary hardware feature of concern, in this case, are the variable clock rates and associated power consumption. Endurosat has performed some measurements of the current draw under different clock rates, and even with some ADCS and other peripherals attached or enabled. Unfortunately, a conclusive analysis will depend on further implementation, however, assuming the worst case, the C&DH processor is not the primary power concern for the satellite. Measuring the power consumption of an running application is possible, and some example code for this will be referenced in the project GitHub. In addition to operating in the 16Mhz lowest clock rate, the FreeRTOS tick may be disabled for further power reduction. There is some uncertainty in what the power consumption of the serial interfaces, particularly the CAN bus, will be.

The primary hardware interfaces of concern are the serial buses between subsystems. Out of the requirements, the most resource intensive will be the transmission of images. Although there is some uncertainty about the data rate of the camera to C&DH connections, it is certainly going to be lower than the practical data rate of the transmission to the ground station(s), and higher if the USB is power + data instead of power only. The ARM Cortex M4 supports an SD card, of which 4GiB would be easily addressible with the 32 bit architecture. This exceeds what would be needed to buffer images between the cameras and the transceiver, with each image taking only 0.9MB. The cameras have their own external storage. The UART interface is capable of 115200 baud, which would mean that a 0.9MB image would be transmitted in 62.5s. The data rate over SPI to the sd card and latency of reading from the UART buffer into memory are not a

bottleneck in comparison to the UART rate. Therefore, the data rate from the cameras, even in the worst case over UART, will allow plenty of time to buffer the images for transmission. The CAN bus is much faster than UART, approximately 10Kbps to 1 Mbps, with the criteria that each transceiver must operate at the same frequency. This is primarily a concern for power consumption, and will not limit the bandwidth to the transceiver.

The software interface of concern again relates the transmission. The limitation on packet size is 100 bytes. The packetization process will involve both software packages and hardware from the Endurosat board. On the prototype, a similar packetization process may be simulated by using the CRC module in addition to the ethernet port, however, this fell outside of the feasible scope of the prototype. In UHF, the packet structure is likely to be an AX.25 packet wrapping an embedded TCP or UDP packet. The S-band packet may be similar due to the popularity of AX.25, or could be a more lightweight point-to-point protocol. AX.25 has a packet size of 24 bytes total: 2 bytes for CRC-16 + 2 bytes for 2 flags + 1 byte control + 1 byte protocol id + 16 bytes for address. A minimal implementation of TCP offered by Endurosat, CSP, takes 2 bytes. Therefore the remaining data payload of a 100 byte packet is 74 bytes. 0.9MB/74 bytes is 12163 packets, or 1216300 bytes, discounting any additional file transfer or command protocol overhead. This leads to a transmission time of 507 seconds per image at the 19.2 kbps data rate stated for the Endurosat UHF transceiver.

Overall, the power consumption, memory, and data packetization constraints have all been met with the proposed software and hardware. There is some uncertainty about the power consumption of the serial interfaces, the scheduling constraints on CPU time imposed by the ADCS, and the final camera payload interfaces. However, this is suitable for the original design objectives of a software and hardware platform, not a full application analysis.

4. Results and Validation

4.1 Prototype Concept

There are two major deliverables for this capstone project: research and initial groundwork which will be used in the final cube satellite project, and a prototype which demonstrates the functionality of the OBC and how it will interact with other subsystems.

The major requirement for the first deliverable is to select an onboard computer which has the necessary data interfaces and power requirements to be compatible with the communications, electric power, and attitude determination and control subsystems and the Canadensys VR camera. The team is confident that the Endurosat C&DH board is the best possible option available for the long term success of the cube satellite project, given factors such as cost, compatibility with other subsystems, and customer support. A decision will be made by the Project Manager for which communication method will be used in the final design; the board we have chosen accommodates both those options.

In order to demonstrate how the onboard computer will interact with other subsystems, two Nucleo-F429ZI boards are used to send and receive mock data to and from the onboard computer. The prototype consists of three modules: the camera, onboard computer, and communications, as shown in Figure 5. The data interface for the Canadensys camera is RS-422 over UART and the data interface for the communications subsystem is the CAN buses.

The other subsystems have been omitted from the prototype for simplicity. The interaction between the camera and communications subsystem is expected to be the most resource intensive task. By demonstrating a successful implementation of this task, it can be assumed that the processor will have the necessary power, memory and processing power to manage the remaining subsystem interactions. The prototype will demonstrate the board's ability to receive serial data from the camera and reformat it into packets for the transceiver. Other tasks will be structured in a similar way, where information from one subsystem is parsed from a serial input, then reformatted and output to another subsystem.

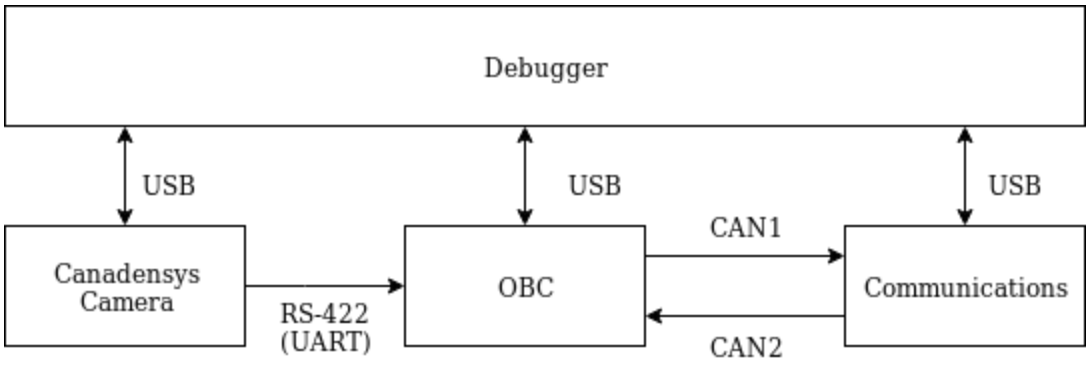


Figure 5. *Module design overview diagram*

4.2 Prototype Fabrication and Cost Breakdown

The Endurosat OBC uses an ARM Cortex M4 or M7 architecture. A software prototype was implemented on an ARM Cortex M4 based MCU. The Nucleo-F429ZI was chosen for the prototype hardware because it was a cost-effective option with a similar processor, data interfaces, and operating system as the Endurosat OBC. Alternative prototyping hardware was also considered as shown in Table 5. The board itself however, is not an appropriate recommendation for the final design because its components would not survive the harsh space environment. The cost of the prototype and the software tools used is shown in Table 6 and Table 7. The prototype cost is well within the capstone project budget, which allocated $75 per group member ($225 total).

The Endurosat datasheet states that the maximum frequency rates of these available cores are 180 MHz and 216 MHz respectively. If the prototype board contains a floating point unit (M4F/M7F), usage should be avoided with compiler options. Unfortunately, imitating the exact pins of vendor buses or the PC/104 mechanical form factor will be outside of the scope of this prototype. However, it should be possible to prototype using a CAN bus, along with USB, I2C, USB, and UART.

Option	Item	Cost
1	Endurosat OBC with pre-installed FreeRTOS	$3625 USD
2	180 MHz Cortex M4	Low: $30: Nucleo-F429ZI, High: $80: EA-QSB-016 (incorrect clock rate, more peripherals)
3	216 MHz Cortex M7	Low: $20: NUCLEO-F767Z, High: $70: MIMXRT1020-EVK (incorrect clock rate, more peripherals)

Table 5. *Hardware prototype options*

Part	Unit Cost	Quantity	Total Cost
Nucleo-F429ZI	~$30	3	$90
Miscellaneous cables	~$3	3	$9

Table 6. *Prototype hardware cost breakdown*

Software development tools	Cost	License
Git with Github	$0	Git uses GNU GPL v2 and GNU LGPL v2.1 GitHub is a proprietary service
UML with Microsoft Visio	$0 (Western license)	Proprietary
Eclipse IDE	$0	Open Source
GNU ARM Embedded Toolchain	$0	Open Source
GNU MCU Eclipse OpenOCD	$0	Open Source

Table 7. *Software tools cost breakdown*

4.3 Development Environment Setup

The team chose to use the open source Eclipse IDE with the GNU ARM embedded toolchain plugin and OpenOCD for the development of the prototype. This development environment is the same across MacOS and Windows which allowed us to collaborate seamlessly when developing on the Nucleo-F429ZI. The GNU ARM embedded toolchain is an Eclipse plugin containing a set of open source tools that make it easy to program microcontrollers with the Arm Cortex-M and Cortex-R processors. OpenOCD is a debug environment that works on both MacOS and Windows. A complete guide to setting up the GNU ARM embedded toolchain with OpenOCD by software engineer Lutong Zhang [26] was followed. Finally, the semihosting GDB debugger was used to connect the code being run on the Nucleo-F429ZI to the debugging console. A complete guide to setting up the GDB debugger by software engineer Thomas Vitale [27] was followed. The User Manual (UM 1725) by STMicroelectronics [26] was the primary reference manual used to develop the prototype on the Nucleo-F249ZI.

4.4 Testing Strategy/Validation Protocols

The goal of the prototype is to validate the choice of processor which is used in the Endurosat OBC and Nucleo-F429ZI. In order to validate this, the cube satellite's main research function, which is to send image data to the ground station, is implemented. This is task will be the most resource intensive task in the onboard computer. The successful implementation of this task will show that the cube satellite's main research function can be fulfilled using the chosen hardware with its available power, memory, and processing power. The CPU usage of this task will show the chosen hardware will have sufficient power, memory and processing power to run the other maintenance tasks which will be less complex and demand fewer resources. This validation phase is known as Integration Testing in the V-model development life cycle.

The following testing strategy is used to validate the recommendation of the Endurosat OBC:

- Test 1: 13.5MB image can be loaded to the C&DH board. If possible, we will obtain a VR image from Canadensys. Otherwise, any 13.5MB file should suffice. This test will validate the board's compatibility with the Canadensys camera subsystem.
- Test 2: The image is framed as CSP or IP to be compatible with both Endurosat and Satlab packet assemblers. This test will validate that the supporting libraries for this processor can adequately support the data reformatting needs of the project.
- Test 3: Packets can be loaded to the "transceiver" module and the full image can be reconstructed. This test will validate the board's compatibility with the communications subsystem.
- Test 4: The OBC can resend packets if a packet is failed, and the image can still be reconstructed. This test will validate the OBC subsystem's ability to verify the successful transmission of a packet, and take action if necessary.

4.5 Final Results and Validation

Test 1 Results

This test successfully passed, with some limitations. The team was able to implement a mechanism which loaded an image from the camera prototype module to the onboard computer prototype module over UART. This validates the board's compatibility with the camera subsystem, which has an RS-422 serial data interface which can be implemented by UART.

A limitation of this test is that the exact specifications of the Canadensys camera subsystem were unable to be obtained through the Project Manager. The group tried to model the prototype as close as possible to the expected system components, but the only details available about the Canadensys camera subsystem were that a compressed VR image which was 0.9MB in size (13.5MB when uncompressed) would be transmitted over RS-422. More specific details about the image would have allowed the group to better demonstrate how the onboard computer would parse and read the file. Despite these lack of details however, the group's successful implementation of an established line of communication between the camera prototype module

and onboard computer prototype module demonstrate and validate the compatibility between the camera and onboard computer.

Test 2 Results

This test failed due to a lack of software documentation for the Nucleo-F429ZI. Although the hardware components of the Nucleo-F429ZI and Endurosat OBC are very similar, the team learned that the board support packages are quite different. The Endurosat OBC offers a much more complete board support package which makes configuration and application programming straightforward. In comparison, the open source software package provided by Nucleo-F429ZI for the STM32 processor requires developers to program directly on the hardware abstraction layer. Programming directly on the hardware abstraction layer is not ideal because it requires immense attention to detail and deep knowledge of the board's hardware. In order to be successful, the team would have required advanced debugging tools and most importantly detailed documentation. Although the team was unable to successfully complete this test, they learned that the more powerful board support package provided by Endurosat in addition to their excellent customer service will allow future groups who will program the board to be successful. Given this knowledge, the team is hopeful that the Endurosat OBC will still be a viable option for the cube satellite.

Test 3 Results

This test successfully passed, with some limitations. Given that the previous test had failed and TCP/UDP packets were not able to be created, the team decided to move forward by implementing a modified design shown in Figure 6, which could still serve to validate the onboard computer's compatibility with the communications subsystem. In the modified design, the onboard computer would take the original image received from the camera prototype module and split it into 100-byte data packets, which is the size of the downlink packets expected by the transceiver. These packets would then be sent over UART, which is an alternative data interface available on the UHF and S-Band communications subsystems.

Ideally, the team wanted to create TCP/UDP packets and send them to the communications prototype module over a CAN bus because in this design, the packets would be addressed which would make it easier to identify failed packets. Overall however, the packet transmission mechanism implemented by the team validates the onboard computer compatibility with the communications subsystem.

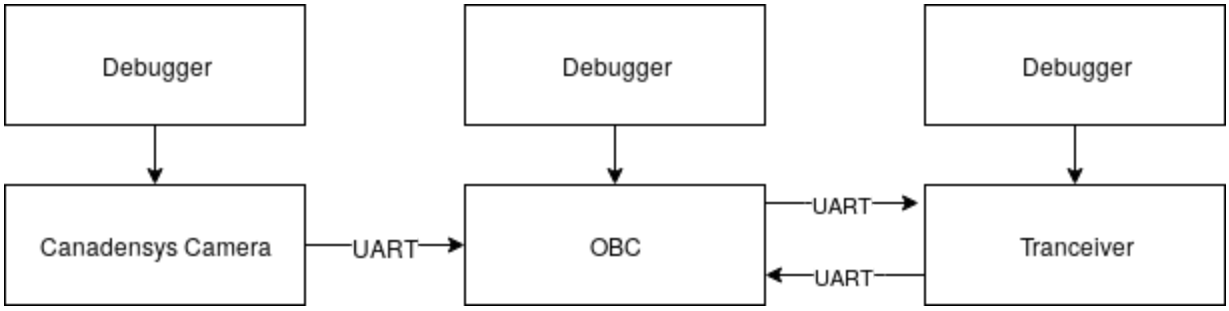


Figure 6. Modified *module design overview diagram*

Test 4 Results

This test successfully passed, with some limitations. The team was able to successfully implement an additional line of communication between the transceiver and the onboard computer prototype module, as shown in Figure 6. Each time a packet was sent by the onboard computer, the onboard computer would wait for an acknowledgement packet from the transceiver. If the message in the acknowledgement packet was "OK," the next packet would be sent. Otherwise, the onboard computer would print the index of the failed packet in the debugger console and attempt to send the packet a second time. If the second attempt at the transmission was unsuccessful as well, the onboard computer would simply log the failure in the debugger console and continue transmitting the remaining packets. This way, the onboard computer would not be stuck in a state of constant failed transmission attempts.

The main limitation of this test is the lack of knowledge surrounding the exact specifications of the transceiver acknowledgement packets. However, the main mechanism developed by the team could be easily modified to meet different packet specifications. Thus, the OBC subsystem's

ability to verify the successful transmission of a packet was validated, giving confidence to the recommendation for the Endurosat OBC.

5. Conclusions, Future Work and Recommendations

5.1 Conclusions

Overall, the objectives for the preliminary design phase have been completed to the degree that available requirements and time have allowed. The design tasks, in review, were: first, to find command and data handling hardware that would be suitable for both UHF and S-Band designs, second, to evaluate an operating system suitable for the software platform, third, to evaluate tools and libraries necessary for development, fourth, to prototype the software platform necessary to perform essential command and data tasks, and last, to demonstrate a method of verifying the prototype. The final recommendation has two parts: the hardware and software platform.

For the hardware platform, the Endurosat OBC provides the necessary CAN bus to communicate with the EPS and transceiver. Additionally, it has the necessary interfaces to handle the ADCS sun sensors, gyroscope, magnetometer, and magnetorquers. The H1 and H2 stack connectors, along with PC/104 form factor, meet the mechanical requirements and dimensions for Cubesats. The Canadensys Cameras will require the Type II Endurosat OBC, with one of the two USB slaves being implemented on the protoboard area. The RS-422 data interface is unspecified but was assumed to be UART, this will need to be clarified before purchase. The Cortex M4 option should be adequate for this project, as there are not specific data handling tasks that would require the floating point unit on the M7.

For the software platform, FreeRTOS should provide the necessary features of deterministic memory allocation and scheduling, a low memory footprint, and free licensing. The open development toolchain with GNU ARM tools, OpenOCD, and GDB is available for any Cortex-M core. These tools have the best chance of being compatible with any configuration tools or sample projects that Endurosat has available. In addition, the Endurosat board support package should provide some ports of essential libraries, likely using different or lower level

peripheral libraries than the ST HAL in the prototype. However, all code from the prototype is portable to the STM32F427. The ST software license should be acquired due to its extensive use in relevant, helpful tutorials.

The chosen C&DH board will support either the UHF or S-band communications boards, represented by the Opportunity and Spirit teams respectively. The deciding factors should ultimately be left to other subsystem requirements such as ground station availability, power generation, or pointing accuracy. The recommendation to purchase an Endurosat M4 OBC Type II and implement the platform on FreeRTOS meets the preliminary design objective of a computing platform, but is underspecified in addressing the final functional requirements of the project.

5.2 Future Work and Recommendations

Work on this subsystem will likely be continued by another computer or mechatronics capstone team (who have been working on the ADCS subsystem this year). In either case, the progress made on the ADCS subsystem will be merged with this subsystem, and their code will need to be rewritten in embedded C to be able to be loaded onto the board. There will be some continuity in the supervisory role as this project progresses to the next phase in the design process, however, some suggested tasks for future work are listed below:

1. Implement and measure CPU time of C&DH tasks (ADCS, telemetry, transmission)
2. Prioritize and measure C&DH tasks with variable power consumption or eclipse modes
3. Verify UDP transmission over unreliable network conditions in a point-to-point (S-band) or multipoint (potentially UHF) connection
4. Implement a secure interface for commands and remote bootloading, licensing the ability to encrypt commands if operating in the amateur UHF band

Additionally, there are some recommendations based on difficulties encountered when prototyping. It should be encouraged that groups in the future become acquainted with the Endurosat board support package as early as possible to circumvent any issues regarding its

implementation. Due to its relevance, the embedded systems course should be taken prior to fourth year to better prepare students for working in this type of environment, if possible. It is also recommended to attempt to get programming resources from the Endurosat manufacturer. They provide knowledgeable and responsive customer support, and can help to provide cubesat-specific code to help get started. To aid in transitioning this project to the next development team, a complete documentation is to be shared on GitHub. This would include the lessons learned through experimenting with the ARM Cortex M processor, recommendations for implementation of the software, and an overview and setup procedure for the development environment.

6. References

[1] E. Brown, "Survey shows Linux and FreeRTOS out front in embedded tech", *LinuxGizmos*, 2017. [Online]. Available: http://linuxgizmos.com/survey-shows-linux-and-freertos-out-front-in-embedded-tech/. [Accessed: 14-Jan-2019].

[2] "Difference between Linux and Windows Operating System (with Comparison Chart) - Tech Differences", *Tech Differences*, 2017. [Online]. Available: https://techdifferences.com/difference-between-linux-and-windows-operating-system.html. [Accessed: 14-Jan-2019].

[3] "From Windows CE to an IoT RTOS", *Insight.tech*, 2018. [Online]. Available: https://www.insight.tech/transport/from-windows-ce-to-an-iot-rtos. [Accessed: 14-Jan-2019].

[4] R. Aroca and G. Caurin, "A Real Time Operating Systems (RTOS) Comparison", *ResearchGate*, 2019. [Online]. Available: https://www.researchgate.net/publication/228940960_A_Real_Time_Operating_Systems_RTOS_Comparison. [Accessed: 14-Jan-2019].

[5] S. Seo, J. Kim and S. Kim, "An Analysis of Embedded Operating Systems: Windows CE, Linux, VxWorks, uC/OS-II, and OSEK/VDX", *Ripublication.com*, 2017. [Online]. Available: https://www.ripublication.com/ijaer17/ijaerv12n18_113.pdf. [Accessed: 14-Jan-2019].

[6] H. Leppinen, "Current use of Linux in spacecraft flight software", *ResearchGate*, 2017. [Online]. Available: https://www.researchgate.net/publication/321788741_Current_use_of_linux_in_spacecraft_flight_software. [Accessed: 14-Jan-2019].

[7] E. Kaltea and M. Johansson, "Comparison of RTLinux and VxWorks with special interests in interrupts", *Citeseerx*. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.4796&rep=rep1&type=pdf. [Accessed: 14-Jan-2019].

[8] "KubOS | Kubos", *Kubos*. [Online]. Available: https://www.kubos.com/kubos/. [Accessed: 14-Jan-2019].

[9] K. Andersson and R. Andersson, "A comparison between FreeRTOS and RTLinux in embedded real-time systems", *SemanticScholar*, 2005. [Online]. Available: https://pdfs.semanticscholar.org/0b0d/ce57cf932da05273653eac853286b79b85e4.pdf. [Accessed: 14-Jan-2019].

[10] D. S, "What are “co-operative” and “pre-emptive” scheduling algorithms?", *Rapita Systems*, 2013. [Online]. Available: https://www.rapitasystems.com/blog/cooperative-and-preemptive-scheduling-algorithms. [Accessed: 14-Jan-2019].

[11] "Difference Between Preemptive and Non-Preemptive Scheduling in OS", *TechDifferences*, 2016. [Online]. Available: https://techdifferences.com/difference-between-preemptive-and-non-preemptive-scheduling-in-os.html. [Accessed: 14-Jan-2019].

[12] “UM10204 I²C-bus specification and user manual,” NXP, N.V., 2014 [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf. [Accessed: 14-Jan-2019].

[13] “Document Library | USB-IF.” [Online]. Available: https://www.usb.org/documents. [Accessed: 14-Jan-2019].

[14] “Endurosat Patch Antenna Data Sheet,” Endurosat, Jan. 2019 [Online]. Available: https://github.com/cubesat-project/CubeSat. [Accessed: 14-Jan-2019].

[15] “Satlab SRS-3 Datasheet Revision 1.1,” Satlab [Online]. Available: https://www.satlab.com/resources/SLDS-SRS3-1.1.pdf. [Accessed: 14-Jan-2019].

[16] “Endurosat User Manual: UHF Transceiver Type II,” Endurosat, Jan. 2019 [Online]. Available: https://cdn4.endurosat.com/modules-datasheets/UHF_type_II_User_Manual_Rev_1.5.pdf.

[Accessed: 14-Jan-2019].

[17] “Canadensys VR Camera,” Cross M., Jan. 2019 [Online]. Available: https://github.com/cubesat-project/CubeSat/wiki/Canadensys-VR-Camera. [Accessed: 14-Jan-2019].

[18] “NewSpace Sun Sensor Data Sheet,” NewSpace, Jan. 2019 [Online]. Available: http://www.newspacesystems.com/wp-content/uploads/2018/02/NewSpace-Sun-Sensor_6a.pdf. [Accessed: 14-Jan-2019].

[19] “Precision Navigation and Pointing Gyroscope,” Silicon Sensing, Jan. 2019 [Online]. Available: https://www.siliconsensing.com/media/1160/crm200-00-0100-132_rev_9.pdf**.** [Accessed: 14-Jan-2019].

[20] “Magnetorquer Rod,” NewSpace, Jan 2019 [Online]. Available: https://www.cubesatshop.com/wp-content/uploads/2016/06/NewSpace-Magnetorquer-Rod_7b.pdf. [Accessed: 14-Jan-2019].

[21] “3-Axis Magnetorquer IC Datasheet,” ROHM Semiconductor, Jan 2019 [Online]. Available: https://www.rohm.com/datasheet/BM1422AGMV/bm1422agmv-e. [Accessed: 14-Jan-2019].

[22]N. B. Ruparelia, “Software development lifecycle models,” *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 3, p. 8, May 2010. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1764810.1764814 [Accessed: 26-Oct-2018].

[23] I. Sommerville, “Software Engineering”, *Pearson Education*, 2006. [Online]. Available: https://edisciplinas.usp.br/pluginfile.php/2150022/mod_resource/content/1/1429431793.203Software%20Engineering%20by%20Somerville.pdf. [Accessed: 14-Mar-2019].

[24] “Understanding Distributed Systems", *Oracle*, 2019. [Online]. Available: https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SD173/ch1.htm#toc009. [Accessed: 14-Mar-2019].

[25] Altium Designer, "Important Considerations in Your Embedded System's Master-Slave Communication Model", *Altium*, 2017. [Online]. Available: https://resources.altium.com/pcb-design-blog/important-considerations-in-your-embedded-systems-master-slave-communication-model. [Accessed: 14-Mar-2019].

[26] "User Manual (UM1725)." STMicroelectronics. Description of STM32F4 HAL and LL Drivers. February 2017. Accessed March 20, 2019. https://www.st.com/content/ccc/resource/technical/document/user_manual/2f/71/ba/b8/75/54/47/cf/DM00105879.pdf/files/DM00105879.pdf/jcr:content/translations/en.DM00105879.pdf.

[26] Zhang, Lutong, and Lutong Zhang. "Develop STM32F4 Discovery(Cortex M4) with Eclipse on Mac OS _ PART 1." Medium. October 18, 2017. Accessed April 15, 2019. https://medium.com/@zlt1213/develop-stm32f4-discovery-board-on-mac-os-part-1-dca644816e5.

[27] Thomas Vitale. "How to Setup Gdb and Eclipse to Debug C Files on MacOS Sierra." Thomas Vitale. October 17, 2018. Accessed April 15, 2019. https://www.thomasvitale.com/how-to-setup-gdb-and-eclipse-to-debug-c-files-on-macos-sierra/.

7. Appendices

Endurosat OBC Data Sheet (Excerpt from User Manual)

The full user manual is provided in the zip file submitted with this report. The user manual/data sheet is not publically available through the Endurosat website and was obtained through Giuseppe Sisinni, a customer support agent (sisinni@endurosat.com).

3 DESCRIPTION

The EnduroSat onboard computer type II is a low power consumption and high performance computing platform for nanosatellites, fully compatible with the CubeSat standard. It is based on ARM Cortex M4 with frequency rate up to 180MHz or optionally on ARM Cortex M7 processor with frequency rate up to 216 MHz. It comes with integrated double redundancy sensors: 3-Axis accelerometers and compass. PWM drivers for magnetorquers and inputs for sun sensors, temperature sensors and gyroscope allow the implementation of the attitude determination and control systems.

Customized interfaces and connectors allow high flexibility of the unit.

A Protoboard area on the PCB allows high customization ideal for test bed and fast prototyping. It is possible to connect additional PCB through connectors and mounting holes and to integrate easily additional sensors and chips such as atomic clocks, GPS receiver and so on.

4 PRODUCT PERFORMANCE AND PROPERTIES

- ARM Cortex M4/M7 processor;
- Frequency rate: up to 180 MHz for M4, up to 216 MHz for M7;
- 2MB Program Memory Size; 256kB RAM for M4, 2MB RAM for M7; 2048kB flash memory;
- MicroSD card slot;
- Integrated double redundancy sensors: 3-axis accelerometer and compass;
- 3x PWM drivers for magnetorquers;
- 6x analog inputs for sun sensor;
- 6x external temperature sensors can be connected;
- Three external gyroscope can be connected
- Interfaces: CAN, 2x USART, UART, 2x I2C, 2x SPI, USB (VCP);
- Real Time Clock
- Flexible frequency eco-mode;
- Weight: 58 g.
- 1Gbit Serial NOR Flash Memory
- 64Mbit Static RAM (Optional)
- Connector for antenna deployment
- ProtoBoard area for easy connection of payload and access to main power and communication busses.

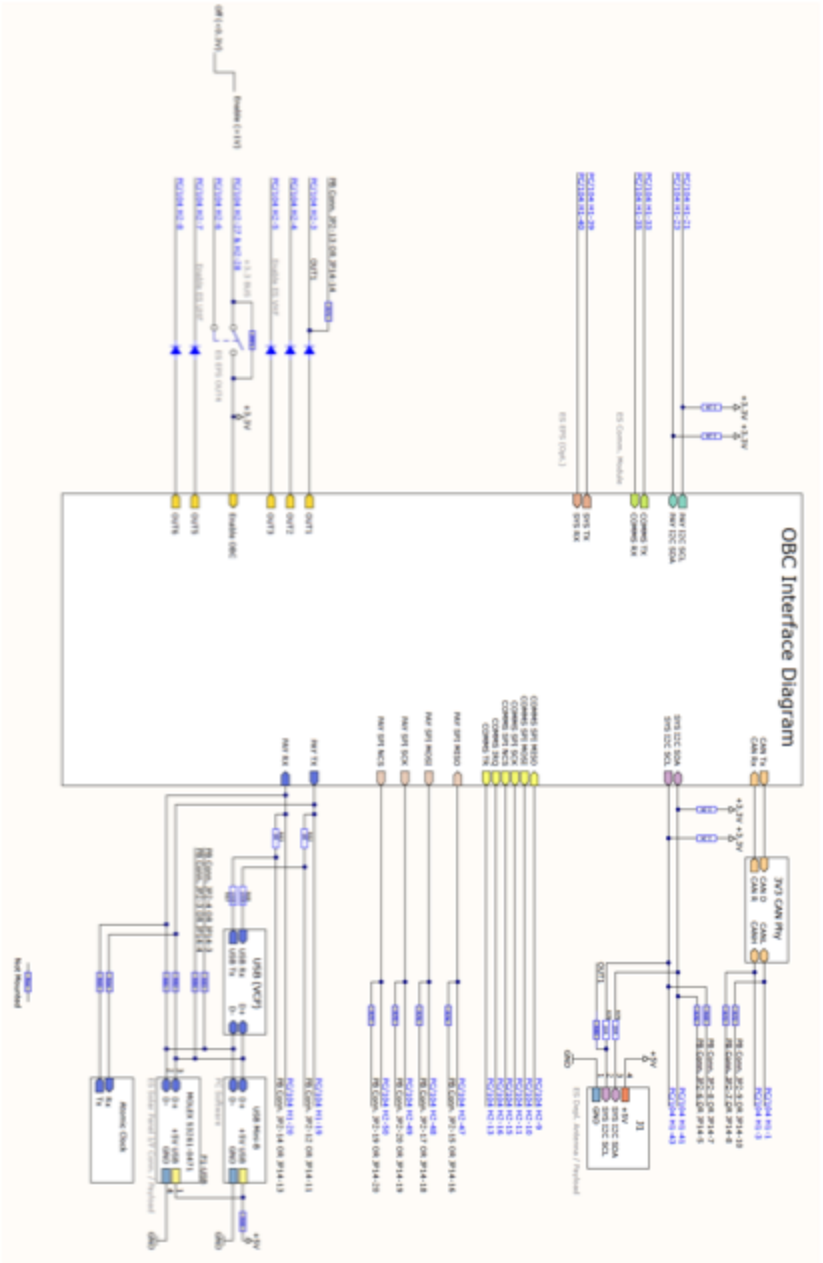
5 ELECTRICAL CHARACTERISTIC

Parameter	Unit	Condition	Min	Typ	Max
Supply voltage	V		3	3.3	3.6
Supply current	mA	STM32F427 @185Mhz		104	123
	mA	STM32F427 @120Mhz		58	72
	mA	STM32F427 @60Mhz		30	38
	mA	STM32F427 @16Mhz		13	27
	µA	3-Axis Accelerometer – Normal Mode[1]	200		400
	µA	3-Axis Accelerometer – Low Power Mode[1]	8		12
	µA	3-Axis Accelerometer – Power Down Mode[1]	0.1		2
	µA	3-Axis Digital Compass – Power Down Mode[2]		1	
	µA	3-Axis Digital Compass - Measurement Mode[2] – Low Power Mode		40	
		3-Axis Digital Compass - Measurement Mode[2] – High Resolution Mode		280	
	mA	Ext. 64M-bit Static RAM (Opt.), F = 18Mhz		45	55
	mA	Ext. 64M-bit Static RAM (Opt.), F = 1Mhz		7.5	9
	µA	Ext. 64M-bit Static RAM (Opt.), Stand-By Mode		8	48
	mA	Ext. 1Gbit NOR Flash Memory Operational Mode @108Mhz (fast-read extended I/O)		4	15
	mA	Ext. 1Gbit NOR Flash Memory Operational Mode @54Mhz (fast-read extended I/O)		6	6
	mA	Ext. 1Gbit NOR Flash Memory Operational Mode @108Mhz (fast-read dual I/O)			18
	mA	Ext. 1Gbit NOR Flash Memory Operational Mode @108Mhz (Operating current (fast-read quad I/O)			20
	µA	Ext. 1Gbit NOR Flash Memory Operational Mode Stand by Mode			200
Bi-directional PWM Outputs	mA	@3.3V			3000
Operating Temperature	°C		-30		85
Storage Temperature	°C			25	

[1] - Current consumption is for one 3-Axis Accelerometer. The OBC has two identical sensors on the same location, but on opposite sides of the PCB.

[2] - Current consumption is for one 3-Axis Digital Compass. The OBC has two identical sensors on the same location, but on opposite sides of the PCB

6 INTERFACE DIAGRAM



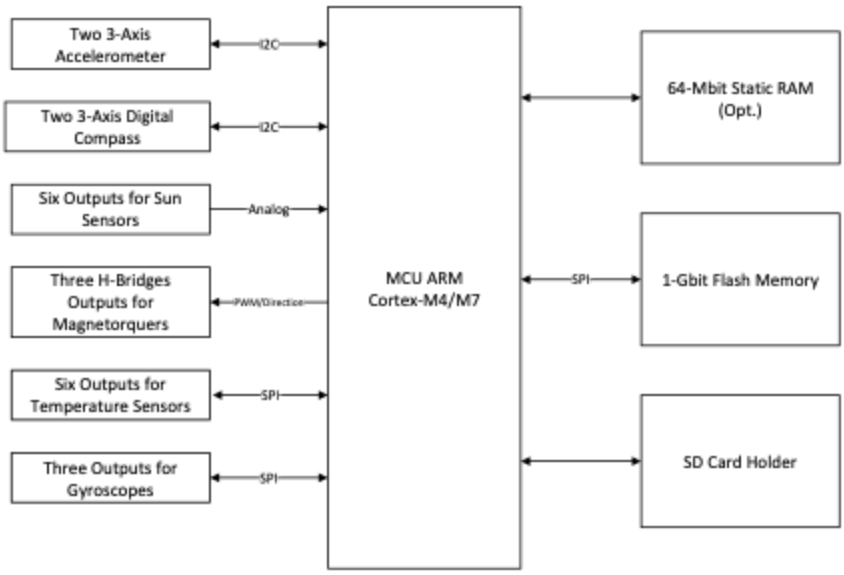
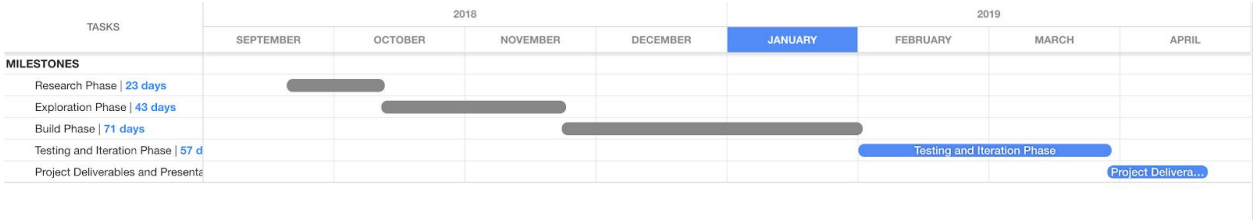


Figure 2 – OBC microcontroller periphery

Gantt Chart (Original)



Gantt Chart (Revised)