# ONBOARD DATA HANDLING FOR CUBE SATELLITE

by

Christopher Bechard

Joyce Lee

Steven Terryberry

(advisor: Dr. Jayshri Sabarinathan)

ECE 4415 Computer Engineering Design Project

**Progress Report**

Department of Electrical and Computer Engineering

Western University

London, Ontario, Canada

**January 14, 2019**

## Table of Contents

**1. Introduction/Background** **3**
1.1 Problem statement 3
1.2 Literature Review 3
1.2.1 Operating Systems 3
1.2.2 Communication 5
1.2.3 Application Architecture 8
1.3 Project Objectives 8
**2. Design Approach** **9**
2.1 Concept Generation 9
2.1.1 Constraints 10
A. Subsystem Functional Overview 10
B. Data Interfaces 10
C. Canadensys Interface 11
2.1.2 Concepts 13
A. C&DH COTS Evaluation 13
2.2 Concept Evaluation and Selection 14
**3. Preliminary Approach** **16**
3.1 Engineering Techniques/Software Tools 16
3.2 Preliminary analysis 18
**4. Prototype Validation Plan** **18**
4.1 Prototype concept 18
4.2 Budget/parts list details 19
**5. Team Member Contribution** **20**
**6. References** **23**
**7. Gantt Charts** **27**

## Glossary

| | |
|---|---|
| ADCS | Attitude Determination and Control System |
| C&DH | Command and Data Handling |
| CAN | Controller Area Network |
| COTS | Consumer Off-the-shelf |
| $I^2C$ | 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 |
| 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 |

# 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 survival 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 (CD&H) 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 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, second, the relevant communication protocols to this report will be compared on their features and context of use, and last, 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 keep 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 Communication

| 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). |

**Fig 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<br>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)<br>Full duplex (3.x) |

**Fig 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, $I^2C$, 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 CubeSat camera, communicate with the ground station, and interface with the components designed by the electrical power, communications and structural teams. This project is one of three capstone projects which will culminate in a final design which will be launched into orbit in 2021. 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 and it will define the preliminary design of the computer system onboard the CubeSat. The goal for this phase of the computer system CubeSat 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 with both S-band and UHF designs
2. Evaluate an operating system to meet requirements
3. Evaluate software packages and tools that will ease development
4. Prototype preliminary command and data tasks
5. Prototype a preliminary secure telecommand interface for satellite operation
6. Demonstrate a testing and verification strategy with the prototype

## 2. Design Approach

### 2.1 Concept Generation

First, to establish a rationale for concept generation, the current constraints of the project, although not entirely fixed, 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 is not intended to replace power, data, or application interface diagrams. It is primarily to outline the basic functions of each subsystem.

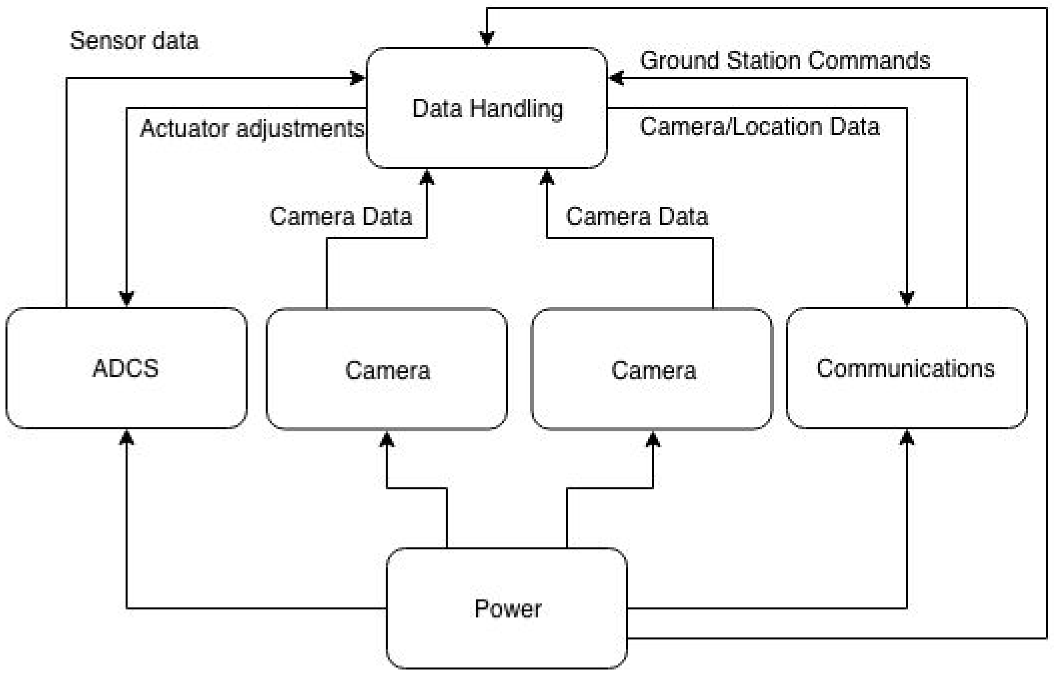


**Fig 3.** *A high level diagram describing the relationships between subsystems.*

#### 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 |

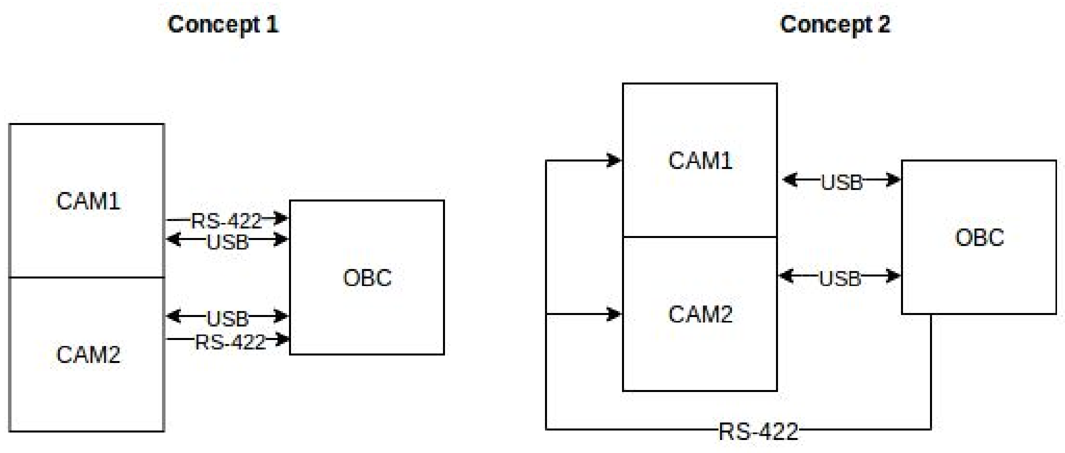
**Fig 4.** *A list of preliminary peripherals and SBCs from other subsystems.*

## C. Canadensys Interface

The details of the camera interface is not yet 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



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.

| | Available Interfaces | Quantity |
|---|---|---|
| Endurosat Cubesat Onboard Computer https://www.endurosat.com/products/cubesat-onboard-computer-obc/ 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 | I²C | 2 |

| | | |
|---|---|---|
| https://www.cubesatshop.com/product/cube-computer/<br><br>Cost: $5161 USD | 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*

**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/applicable. The selection of the "best"/preferred concept is based on technically sound engineering principles.

In the past 4 months the group has progressed through the research phase and design and exploration phase of the project. Several options for CD&H 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 |

| 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 |

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. Preliminary Approach

### 3.1 Engineering Techniques/Software Tools

The project will prioritize using proven open source or off-the-shelf software/hardware components. The team will use Github as a version control tool to manage the development of the operating system. The team will also contribute to the overall cube satellite project's Github Wiki which will contain important information that other subsystem teams can reference, and Slack to communicate with each other as well as the other subsystem teams. The team will also use Unified Modeling Language (UML) diagrams to visualize and organize use cases and system activities.

The engineering methodology used will be similar to a waterfall or V-model development life cycle [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.

The team is currently near the end of the module design phase and is entering the coding phase, in which the tasks which define the interactions between subsystems will be implemented in the prototype. Once the coding phase has begun the team will simultaneously write and perform unit tests as they implement each task. Github will be 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. This is likely where this project will conclude, since integration 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 Preliminary analysis

The team is confident that the Endurosat CD&H 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 as shown in the Decision Matrix in Section 2.2. The board contains sufficient peripherals for the other subsystems and comes from a reputable manufacturer. It includes a software development kit which includes FreeRTOS pre-installed, support documentation, microcontroller interface driver libraries, and sample code. Similarly, FreeRTOS is the best possible option due to its well supported, powerful, lightweight library as shown in the Decision Matrix in Section 2.2.

However, this may not be the board which the team will build this year's system on because it may not arrive in time for this year's capstone project. The selection and purchase of the board has been delayed because the purchase is contingent on the selection of other parts by the other subsystem groups. As a result, the prototype which will demonstrate the board's essential functionality will be built on an alternate board.

## 4. Prototype Validation Plan

### 4.1 Prototype concept

The endurosat OBC uses an ARM Cortex M4 or M7 architecture. The team will not have access to flight hardware or any prototype board from Endurosat, given the weight of purchasing decisions and large manufacturer lead times. It is fair to state at this point that a software prototype should be implemented on ARM Cortex M4 or M7 based MCU. 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.

We are still evaluating which cost effective ARM Cortex M4 and M7 boards that meet the clock rate requirements.

The following is a preliminary list of functions will be implemented using the board's software development kit.

| **Commands** | | |
|---|---|---|
| **Subsystem** | **Command** | **Conditions** |
| Communications | Adjust magnetorquer | Receive command from ground station |
| | Send location data to ground station | Receive request from ground station Periodic |
| | Send camera data to ground station | Receive request from ground station Periodic |
| | Turn off power | Receive request from ground station |
| Camera | Load images into memory for transmission | Periodic Receive request from ground station |
| ADCS | ADCS will be implemented in a future capstone project. It is currently being prototyped by a mechatronics capstone team using a different CD&H board. | |

**4.2 Budget/parts list details**

The grant acquired for the project according to the proposal is $250,000 with $50,000 going to educational outreach activities, leaving $200,000 for launch and some unknown amount for CubeSat development. The following is an updated list of parts and software tools and their associated cost:

| Option | Item | Cost |
|---|---|---|
| 1 | Endurosat OBC with pre-installed FreeRTOS | $3625 USD |
| 2 | 180 MHz Cortex M4 | Low: $30: Teensy 3.6,<br>High: $80: EA-QSB-016 (incorrect clock rate, more peripherals) |
| 3 | 216 MHz Cortex M7 | Low: $20: NUCLEO-F767Z,<br>High: $70: MIMXRT1020-EVK (incorrect clock rate, more peripherals) |
|  | Additional Cabling/Components | - |

| Software development tools | Cost | License |
|---|---|---|
| Git with Github | $0 | Git uses GNU GPL v2 and GNU LGPL v2.1<br>Github is a proprietary service |
| UML with Microsoft Visio | $0 (Western license) | Proprietary |

## 5. Team Member Contribution

Each team member is 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. Chris and Steven are fourth year computer engineering students specializing in software systems.

The following is a list of anticipated tasks required to complete the project with approximate time estimates. This timeline was created with an approximate 30 week (~8 month) timeline in mind. The team is on schedule with the research, exploration and design of the project, however as they enter the build phase they encounter several major roadblocks which they predict will take time away from the building, testing and iteration phase. These roadblocks include:

- Although our research shows that the Endurosat onboard computer is the best hardware option to support the cube satellite (regardless of whether spirit or opportunity is chosen next year), we are hesitant to buy it because it is a large purchase and requires complete commitment from the other subsystems, who have expressed wavering commitment to their chosen components. Furthermore, once the purchase is made by the Project Manager, it may take several weeks to arrive. This leaves us unsure of the hardware that we can use to prototype this particular capstone project with.
- There is a lack of mock data to simulate interactions between subsystems. It is unclear what the incoming data will look like since the data is coming from many different serial protocols and possibly from different manufacturers.

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

| | |
|---|---|
| **Exploration and Design Phase**<br>- Define objectives and constraints<br>- Identify margins for power budgeting<br>- Identify considerations/connections required to interface with other subsystems (mounting, power, mass etc)<br>    - Specify a communication interface for transmitting digital application data from satellite to ground station<br>    - Specify application data interfaces for each subsystem<br>    - Specify power interfaces for each subsystem<br>    - Specify thermal interfaces for each subsystem<br>    - Telecommand interface development<br>    - Data budgeting<br>- Mid-term progress report | 6 |
| **Build Phase**<br>- Assess feasibility of building prototype on Endurosat board. Otherwise, make arrangements to prototype on the alternative CD&H board<br>- Identify criteria for success and create tests for each subsystem component which will be used during the testing phase<br>- Identify methods to demonstrate the system (prototype hardware, testing simulations)<br>- 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)<br>- Request and purchase parts (after Project Manager Matt Cross submits an application for satellite license in January) | 10 |

| | |
|---|---|
| - Application modelling for essential functionality (some components might be “black boxes” such as power management and the actual ADCS functionality)<br>- Confirm mass budget for structural team (with basic mounting details)<br>- Thermal budgeting | |
| **Testing and Iteration Phase**<br>- Test each individual subsystem component and identify successes and failures<br>- Make adjustments to fix failures<br>- Simulate whole mission (day in the life testing) [4]<br>- Demonstrate essential functionality with a prototype | 8 |
| **Project Deliverables and Presentations**<br>- Design demonstration<br>- Design presentation<br>- Western CPSX Space Day presentation<br>- Final report and reflection | 3 |

## 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].

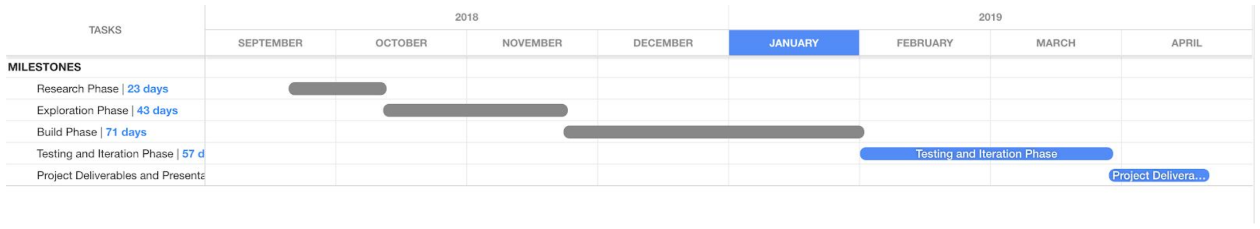
[22]N. B. Ruparelia, “Software development lifecycle models,” *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 3, p. 8, May 2010. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1764810.1764814 [Accessed: 26-Oct-2018].

[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-Jan-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-Jan-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-Jan-2019].

## 7. Gantt Charts


TASKS
2018
2019
SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER
JANUARY
FEBRUARY
MARCH
APRIL
MILESTONES
Research Phase | 23 days
Exploration Phase | 43 days
Build Phase | 71 days
Testing and Iteration Phase | 57 d
Project Deliverables and Presenta
Testing and Iteration Phase
Project Delivera...