



University of Kentucky  
UKnowledge

---

Theses and Dissertations--Electrical and  
Computer Engineering

Electrical and Computer Engineering

---

2013

## PROVIDING A PERSISTENT SPACE PLUG-AND-PLAY AVIONICS NETWORK ON THE INTERNATIONAL SPACE STATION

Zachary A. Jacobs  
*University of Kentucky*, [zachjacobs@gmail.com](mailto:zachjacobs@gmail.com)

[Right click to open a feedback form in a new tab to let us know how this document benefits you.](#)

---

### Recommended Citation

Jacobs, Zachary A., "PROVIDING A PERSISTENT SPACE PLUG-AND-PLAY AVIONICS NETWORK ON THE INTERNATIONAL SPACE STATION" (2013). *Theses and Dissertations--Electrical and Computer Engineering*. 16.  
[https://uknowledge.uky.edu/ece\\_etds/16](https://uknowledge.uky.edu/ece_etds/16)

This Master's Thesis is brought to you for free and open access by the Electrical and Computer Engineering at UKnowledge. It has been accepted for inclusion in Theses and Dissertations--Electrical and Computer Engineering by an authorized administrator of UKnowledge. For more information, please contact [UKnowledge@lsv.uky.edu](mailto:UKnowledge@lsv.uky.edu).

## **STUDENT AGREEMENT:**

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained and attached hereto needed written permission statements(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine).

I hereby grant to The University of Kentucky and its agents the non-exclusive license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless a preapproved embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

## **REVIEW, APPROVAL AND ACCEPTANCE**

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's dissertation including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Zachary A. Jacobs, Student

Dr. James Lumpp, Jr., Major Professor

Dr. Zhi David Chen, Director of Graduate Studies

PROVIDING A PERSISTENT SPACE PLUG-AND-PLAY AVIONICS NETWORK ON THE  
INTERNATIONAL SPACE STATION

---

THESIS

---

A thesis submitted in partial fulfillment of the  
requirements for the degree of Master of Science in  
Electrical Engineering in the College of Engineering  
at the University of Kentucky

By

Zachary Andrew Jacobs

Lexington, Kentucky

Director: Dr. James Lumpp Jr, Professor of Electrical Engineering

Lexington, KY

2013

Copyright © Zachary Andrew Jacobs 2013

## ABSTRACT OF THESIS

### PROVIDING A PERSISTENT SPACE PLUG-AND-PLAY AVIONICS NETWORK ON THE INTERNATIONAL SPACE STATION

The CubeLab is a new payload standard that greatly improves access to the International Space Station (ISS) for small, rapid turn-around microgravity experiments. CubeLabs are small (less than 16"x8"x4" and under 10kg) modular payloads that interface with the NanoRacks Platform aboard the ISS. CubeLabs receive power from the station and transfer data using the standard terrestrial plug-and-play Universal Serial Bus (USB). The Space Plug-and-play Avionics (SPA) architecture is a modular technology for spacecraft that provides an infrastructure for modular satellite components to reduce the time to orbit and development costs for satellites. This paper describes the development of a bus capable of interfacing SPA-1 payloads in the CubeLab form-factor aboard the ISS. This CubeLab also provides the "discover and join" functionality that is necessary for a SPA-1 network of devices. This will ultimately provide persistent SPA capabilities on the ISS which will allow users to send SPA-1 devices to orbit for on-the-fly installation by astronauts.

KEYWORDS: CubeSat, CubeLab, NanoRacks, Plug and Play, Distributed Computing

---

Zachary A. Jacobs

---

March 18, 2013

PROVIDING A PERSISTENT SPACE PLUG-AND-PLAY AVIONICS NETWORK ON THE  
INTERNATIONAL SPACE STATION

By

Zachary Andrew Jacobs

James E. Lumpp, Jr.

---

Director of Thesis

Zhi David Chen

---

Director of Graduate Studies

March 18, 2013

---

To my parents for always believing in me and providing me with the inspiration to partake in this adventure.

## ACKNOWLEDGEMENTS

I would like to thank Kentucky Space for taking the initiative to start an aerospace industry in my backyard. They have helped put Kentucky on the map in the aerospace community.

I would like to thank to the current and past members of the Space Systems Lab that I had the pleasure of working with. I wish all of them the very best of luck in all of their future endeavors. Specifically, thanks to the Space Systems Lab members, Trevor Fenwick, Max Bezold, and Chris Mitchell. Their assistance in helping me understand SPA and how to adapt it to the 8051 in a very short period of time was impressive and much appreciated. Additionally, I would like to thank Samir Rawashdeh. His guidance during writing process of my publication and this thesis was very valuable and his assistance will not soon be forgotten.

I would like to also thank Brian Zufelt at COSMIAC. Through his patience in meetings and the ground work that he laid out, I was able to quickly adapt SPA for our application.

Lastly, I would like to extend a special thanks to my graduate advisor, Dr. James Lumpp. My excitement about space and my career path have changed significantly since I first started in the Space Systems Lab. I will be forever grateful for the opportunities that he provided me.

# TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b> .....	<b>IV</b>
<b>TABLE OF CONTENTS</b> .....	<b>V</b>
<b>LIST OF FIGURES</b> .....	<b>VII</b>
<b>1 INTRODUCTION</b> .....	<b>1</b>
1.1 CUBESATS.....	2
1.2 NANORACKS AND CUBELAB RESEARCH.....	3
1.3 THE SPA STANDARD.....	4
1.4 PROBLEM STATEMENT .....	7
<b>2 BACKGROUND</b> .....	<b>8</b>
2.1 CUBESAT STANDARD.....	8
2.1.1 Success of the CubeSat Standard.....	10
2.1.2 Barriers to CubeSat Standard .....	11
2.2 THE NANORACKS PLATFORM AND THE CUBELAB STANDARD .....	13
2.2.1 NanoRacks Platform .....	13
2.2.2 The CubeLab Standard.....	16
2.2.3 Real-time Operations.....	17
2.2.4 Current Status.....	19
2.2.5 Microgravity Research on ISS.....	20
2.2.6 Comparison of CubeSat and CubeLab Platforms.....	21
2.2.7 Alternative technologies available .....	22
2.3 SPA ARCHITECTURE.....	23
2.3.1 SDM.....	24
2.3.2 ASIM .....	26
2.3.3 XTEDS .....	27
2.3.4 Current adoption of SPA .....	28
<b>3 AMESLAB BUS</b> .....	<b>31</b>
3.1 SYSTEM REQUIREMENTS.....	31
3.1.1 CubeLab Specification and NanoRack requirements .....	32
3.1.2 Interface Requirements.....	32
3.1.3 Functional Requirements .....	32
3.2 HARDWARE .....	33
3.3 SOFTWARE.....	36



<b>4</b>	<b>SPA-1 NETWORK ON A CUBELAB.....</b>	<b>39</b>
4.1	SOFTWARE.....	39
4.1.1	SDM-lite functionality .....	40
4.1.2	AmesLab software .....	48
4.1.3	Code Compliance .....	50
4.2	HARDWARE .....	51
4.2.1	I2C isolators.....	51
4.2.2	Physical connector.....	52
4.3	OPERATIONS FOR INSTALLING NEW SPA DEVICES .....	52
4.3.1	Pre-Launch .....	53
4.3.2	Post-Launch.....	53
4.4	8051-BASED ASIM.....	56
<b>5</b>	<b>RESULTS.....</b>	<b>57</b>
5.1.1	Network enumeration.....	57
5.1.2	Reporting temperature.....	58
5.1.3	ASIM re-enumeration.....	59
5.1.4	Multiple ASIM re-enumeration .....	59
5.1.5	ASIMs changing addresses.....	60
5.1.6	ASIM powering off for extended period of time .....	61
5.1.7	High Round Robin Count .....	61
5.2	ASIM RESULTS.....	62
<b>6</b>	<b>CONCLUSION .....</b>	<b>64</b>
	<b>APPENDICES .....</b>	<b>66</b>
APPENDIX A	SOFTWARE FLOWCHART FOR SPALAB .....	66
APPENDIX B	SPA-1 NETWORK TRAFFIC FOR NETWORK ENUMERATION .....	76
APPENDIX C	SPA-1 NETWORK TRAFFIC FOR TEMPERATURE CONTROLLED LED .....	78
APPENDIX D	SPA-1 NETWORK TRAFFIC FOR ASIM RE-ENUMERATION .....	79
APPENDIX E	SPA-1 NETWORK TRAFFIC FOR MULTIPLE ASIM RE-ENUMERATION .....	80
APPENDIX F	SPA-1 NETWORK TRAFFIC FOR ASIMs SWITCHING ADDRESSES.....	81
APPENDIX G	SPA-1 NETWORK TRAFFIC FOR ASIM POWERING OFF FOR EXTENDED PERIOD OF TIME.....	82
APPENDIX H	SPA-1 NETWORK TRAFFIC FOR ROUND ROBIN COUNT GREATER THAN 100.....	83
	<b>LIST OF ACRONYMS.....</b>	<b>84</b>
	<b>REFERENCES.....</b>	<b>87</b>
	<b>VITA.....</b>	<b>90</b>

## LIST OF FIGURES

<b>Figure 1 - 1U CubeSat, KYSat-1 by Kentucky Space</b>	<b>2</b>
<b>Figure 2 - NanoRacks and CubeLabs exploded view</b>	<b>3</b>
<b>Figure 3 - P-POD with four 1U CubeSats during integration for the ELaNa-I mission</b>	<b>9</b>
<b>Figure 4 - CubeSat organizational structure by Cal Poly</b>	<b>10</b>
<b>Figure 5 - P-POD location on Taurus XL - credit: NASA</b>	<b>13</b>
<b>Figure 6 - NanoRacks Platform and CubeLab exploded view</b>	<b>15</b>
<b>Figure 7 - Data path from CubeLab payload to CubeLab developer</b>	<b>19</b>
<b>Figure 8 - Shannon Walker after installation of NanoRacks Platform-2</b>	<b>20</b>
<b>Figure 9 - Effect of microgravity (right) on combustion process (credit: NASA)</b>	<b>21</b>
<b>Figure 10 - Pyramid comparing devices and data rates in SPA devices [1]</b>	<b>24</b>
<b>Figure 11 - SPA network showing use of hub/router [1]</b>	<b>24</b>
<b>Figure 12 - SPA Architecture showing SDM [1]</b>	<b>25</b>
<b>Figure 13 - ASIM provides an interface to the SPA network for legacy devices [1]</b>	<b>26</b>
<b>Figure 14 - Example of a simple xTEDS for a temperature sensor [1]</b>	<b>27</b>
<b>Figure 15 - PnP component layout inside QuadSat-PnP [19]</b>	<b>29</b>
<b>Figure 16 - Block diagram of the AmesLab electrical interfaces</b>	<b>33</b>
<b>Figure 17- Exploded view of AmesLab enclosure</b>	<b>35</b>
<b>Figure 18 - Rev0 and Rev1 of AmesLab Development board</b>	<b>35</b>
<b>Figure 19 - Populated flight board for AmesLab</b>	<b>36</b>
<b>Figure 20 - Top level software flowchart for AmesLab software</b>	<b>38</b>
<b>Figure 21 - Software layering diagram for SPALab</b>	<b>40</b>
<b>Figure 22 - Top-level software flowchart for SDM-lite functionality</b>	<b>41</b>
<b>Figure 23 - Structures used in SDM-Lite for data storage</b>	<b>41</b>
<b>Figure 24 - Round Robin used to communicate with SPA-1 ASIMs</b>	<b>44</b>
<b>Figure 25 - 50-pin header connection for SPA Devices</b>	<b>52</b>
<b>Figure 26 - Operations for installing new SPA-1 device into the SPALab</b>	<b>55</b>
<b>Figure 27 - Black Box test setup for SPALab</b>	<b>57</b>
<b>Figure 28 - CubeFlow kit from COSMIAC with AmesLab attached</b>	<b>62</b>
<b>Figure 29 - SPA Application showing Temperature data from 8051-based ASIM</b>	<b>63</b>

# 1 Introduction

The Space Age began in 1957 when the Soviet Union launched Earth's first artificial satellite, Sputnik-1. Besides the political advantages, the technology in Sputnik-1 also provided valuable data about Earth's upper atmosphere and the ionosphere [1]. The launch of Sputnik-1 set the stage for a number of scientific advancements in the name of space. Since then, many different types of satellites have been developed including: weather satellites, scientific research satellites, navigation satellites, communications satellites, cosmic observation satellites and military satellites.

The monetary and schedule costs associated with building, launching and maintaining a satellite are high [2]. Differences in satellite mission objectives typically prohibit assembly-line style production that would help ease the cost and schedule constraints. However, technological advancements made in the way of miniaturization of electronics and the development of Commercial Off-The-Shelf (COTS) components can also help reduce the cost of satellite missions. The maturity of a new COTS component is evaluated by using a measure called the Technology Readiness Level (TRL) [3]. Once the TRL reaches a sufficient level, the COTS part can reliably be integrated into a subsystem on a satellite.

COTS parts and the miniaturization of electronics on satellites are achieved through the use of the following standards: CubeSats, CubeLabs and Space Plug-and-Play Avionics (SPA). The introduction section further expands on those standards and provides a motivation for the thesis work that utilizes those standards to reduce barriers associated with space research and provide an easier method to increase the TRL of COTS components.

## 1.1 CubeSats

In 1999, the CubeSat program was created as an effort to reduce the cost and development time, increase accessibility to space, and sustain frequent launches of student satellites. In simple terms, this standard defined a 1 Unit (1U) CubeSat to be a 10 cm x 10 cm x 10cm cube-sized satellite that weighs less than 1.33 kg [1]. An example CubeSat, Kentucky Space's KYSat-1, can be seen in Figure 1 [2].

Using a standard size and weight for a satellite enables a common launching mechanism, the Poly Picosatellite Orbital Deployer (P-POD), to launch CubeSats from a wide variety of launch vehicles. This satellite standard has been growing in popularity since its inception and has been utilized by industry and government agencies.



**Figure 1 - 1U CubeSat, KYSat-1 by Kentucky Space**

CubeSats are typically composed of COTS parts and have the same subsystems as much larger satellites. Examples of systems on a CubeSat include: power generation and storage, radio communication, satellite bus, payload, thermal, gyros, imaging systems, structure, and attitude determination and control. Typical CubeSat missions range from a 1U size (10cm x 10cm x 10cm) to a 3U size (10cm x 10cm x

30cm). CubeSat mission objectives have included: technology demonstrations, Earth remote sensing, biological experimentation, and the study of the cosmos. Some of the better-known CubeSat missions are as follows: GeneSat-1, NanoSailD, PHARMASat, RAX, O/OREOS [3] [4].

The success of the CubeSat form factor can be attributed to the standardized launching platform which allows any CubeSat to be exchanged with one another if the need arises. The familiarity of the CubeSat standard has been leveraged during the creation of a similar standard for research on the ISS called CubeLabs which utilize the NanoRacks facility on the ISS.

## 1.2 NanoRacks and CubeLab Research

Use of the existing infrastructure on the ISS for microgravity research was susceptible to budget overruns and extended schedules due to the barriers that were in place. In 2009, NanoRacks began development of a facility that would make an effort to reduce those barriers. To do this, NanoRacks set out to provide a common interface to the ISS for experiments. The NanoRacks platform leverages the familiarity of the CubeSat standard as described in Paragraph 1.1 to bring CubeSat style research to the ISS, called CubeLabs [5].

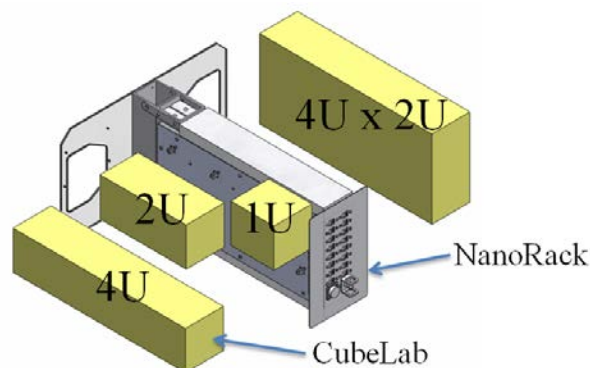


Figure 2 - NanoRacks and CubeLabs exploded view

The NanoRack platform serves as an interface to the ISS and provides a mechanical attachment point, power and data transfer for CubeLab modules. CubeLabs can be flown to and from the ISS on a variety of manned and unmanned vehicles to support inexpensive, repeatable accesses for small payloads. Once aboard the ISS, CubeLabs are installed onto the NanoRacks facility. As seen in Figure 2, each NanoRack facility is capable of hosting 16 CubeLabs. The CubeLab standard was developed which defines the form-factor electrical, mechanical and data interface requirements for a CubeLab [6].

The University of Kentucky Space Systems Laboratory (UK SSL) leveraged this research standard with an aim to further extend the capabilities and developed a bus for a CubeLab payload called AmesLab bus. The aim of the AmesLab bus is to adapt NASA Ames payloads to the NanoRack platform by creating a CubeLab that conforms to the NASA Ames Research Center standard payload interface. Additionally, the AmesLab bus provides additional power and data storage capacity for experiments. SPA capabilities are also being added to the NanoRacks platform through the use of the SPALab [7].

### **1.3 The SPA standard**

Air Force Research Laboratory (AFRL) has led an effort to bring a PnP-based system called SPA to space to facilitate rapid development and building of spacecraft. This effort has been shown to significantly reduce the complexity of spacecraft design and reduce the time to orbit as well. The success of this effort is due to the fact that the SPA architecture implements a self-organizing network of devices where components are self-describing and attached to a standardized data and power bus [1] [2].

SPA was developed in an effort to reduce the complexity of the connection between avionics components on spacecraft. Ultimately, this led to the development of a

Plug-and-Play (PnP)-style architecture that defines hardware and software connections and the interactions between the parts in the system. Due to the large breadth of complexity of avionics used in spacecraft, data rates and power requirements for components vary greatly. This has led to the creation of four different SPA interfaces. Starting with the most complex and power-capable, the SPA interfaces are: SPA-O, SPA-S, SPA-U and SPA-1. The interfaces are based on optical, spacewire, USB and I2C networks, respectively. The latter of the four interfaces, SPA-1 is typically used on small spacecraft due to power availability and low data rates for devices.

The SPA architecture is such that a SPA middleware component called the SDM provides a service that allows SPA devices to be discovered and join the SPA network. The SDM can loosely be referred to as the “traffic cop” that discovers new devices on a network and records the capabilities of the new device in a data registry so existing SPA devices can have access to the data that is available from the new device. Currently, the SDM is only compiled to run on Linux and VxWorks operating systems [1]. On the most complex SPA layers, SPA-O, SPA-S, and SPA-U, the power requirements of running a Linux or VxWorks operating system (OS) are typically justified as these systems have more power available. SPA-1 based systems are typically used on small satellites, namely CubeSats that do not have the luxury of a large power budget. Through the work performed by this thesis and a collaborative effort between the SSL at The University of Kentucky and The Configurable Space Microsystems Innovations and Applications Center (COSMIAC) at The University of New Mexico, a “lite” version of the SDM is developed. The new version of the SDM, called SDM-Lite, specifically targets limited resource SPA-1 based networks.

Additionally, COTS devices exist that facilitate rapid integration of avionics, subsystems and payloads onto a SPA network through a standardized interface called an Appliqué Sensor Interface Module (ASIM) [3]. The ASIM works as a

“bridge” between non-SPA devices and the SPA network. COTS ASIMs are very beneficial to the overarching goals of AFRL as this makes it much easier for device makers to make their devices SPA-compliant. However, different variants of ASIMs can be used or even developed by anyone. Through the work in this thesis, an 8051-based ASIM is also developed as an additional option in this ecosystem of ASIMs.

SPA has proven to be beneficial for some, but is difficult for researchers who don't have access to a satellite to operate their device. The obvious budget and manifestation requirements just to operate a SPA sensor in microgravity need not be listed here. Given that SPA isn't widely adopted at this point, it would prove to be rather difficult to get a launch just to test a new SPA device. Sensor designers that only need to operate their device in microgravity could take advantage of the microgravity environment on the ISS. This environment could be made even easier if a SPA interface existed on the ISS. With the launch and operation of ISS experiments now being sponsored by a NASA and Center for the Advancement of Science in Space (CASIS) partnership, this will be a very attractive option for low budget missions allowing developers to easily operate their SPA-1 based experiments and payloads aboard the ISS.

The CubeSat and CubeLab standards provide access to the microgravity environment through the utilization of standardized platforms. The SPA architecture provides a method to quickly reduce spacecraft design and reduce the time to orbit. The problem statement of this thesis details a simple, low-power Satellite Data Model (SDM) for SPA-1 based networks which facilitates easier access to the microgravity environment by providing a persistent on-orbit interface for SPA-1 based devices on the ISS.



## 1.4 Problem Statement

This thesis focuses on extending the functionality of the AmesLab bus to create a persistent SDM-Lite for SPA-1 networks on the ISS and to create an 8051-based ASIM. SDM-Lite functionality is added to the NanoRacks facility by utilizing the existing AmesLab bus. This work enables SPA-1 device designers to quickly and inexpensively operate their devices on a SPA-1 network in microgravity. The SPA configuration for the AmesLab bus will herein be referred to as the SPALab. The ASIM that is developed allows SPA device designers to add their non-SPA devices to a SPA-1 network by using the ASIM as a “bridge”. This further extends the ASIMs which are available on the market. The software that is developed through this thesis work is verified through the code review processes and extensive testing.

The remainder of this thesis is organized as follows. The second chapter, the Background, provides an in-depth explanation of the CubeSat standard, the NanoRacks platform, the CubeLab standard and the SPA architecture. The third chapter, AmesLab bus, contains a detailed design description of the AmesLab bus created by the SSL. This bus is included as a separate chapter as some of the design decisions were made with the SPALab in mind. The fourth chapter, SPA-1 network on a CubeLab, details the efforts of this thesis which increases the capabilities of the AmesLab bus by adding an SDM-Lite to accept SPA-1 devices. The fifth chapter, Results, details the results of this thesis. The final chapter, Conclusion, provides an analysis of the results of this work and provides a prospect for future research that can be performed.

## 2 Background

This chapter introduces the CubeSat standard that changes the way space research is performed at the university level over the past decade. The NanoRacks Platform and CubeLab standard are also discussed as a platform that is attempting to achieve CubeSat-style success for ISS payloads. Finally, the SPA standard is introduced as a tool for rapid and affordable design and integration of spacecraft components.

### 2.1 CubeSat Standard

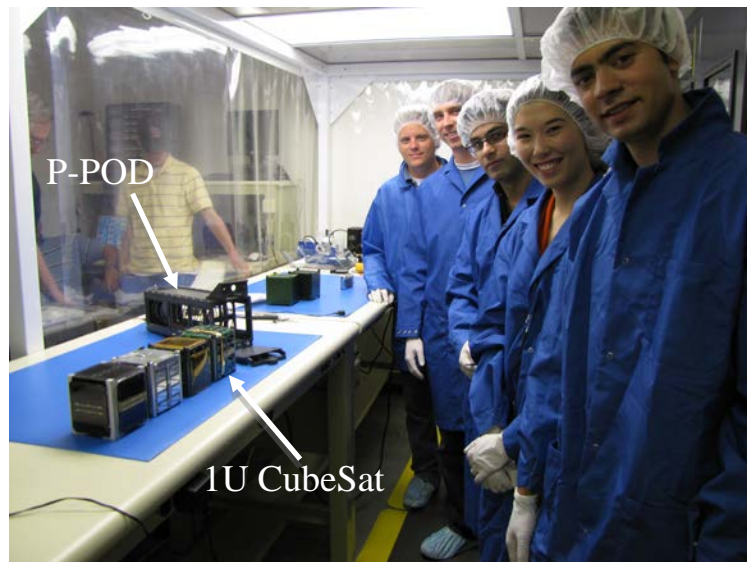
During the late 1990s, only a few undergraduate university aerospace programs around the country were developing small satellites. Unfortunately, the high launch costs and lengthy mission timeline put building a small satellite out of the reach of other undergraduate programs. Stanford University's Space Systems Development Lab worked with California Polytechnic State University (Cal Poly) to define the CubeSat standard. The standard was based on the picosatellites that were developed for Stanford's Orbiting PicoSat Launcher (OPAL) satellite. The CubeSat standard was created as an effort to reduce the cost and development time, increase accessibility to space, and sustain frequent launches of student satellites. The CubeSat standard design was detailed in the form of a document called the CubeSat Design Spec (CDS) so other universities could build CubeSats [4].

In its simplest form, the current CDS defines a 1U CubeSat as a 10cm cube with a weight of up to 1.33 kg. 1U CubeSats can be combined or stacked to create a 2U (10cm x 10cm x 20 cm) or a 3U (10cm x 10cm x 30 cm). The CDS goes further to describe the following types of requirements: general, safety, material, mechanical, electrical and testing [4]. An example of a 1U CubeSat can be seen in Figure 1.

Building a CubeSat to the CDS facilitates a common launching mechanism, the P-POD, to launch a CubeSat from a launch vehicle (LV) into orbit. The P-POD was developed

by Cal Poly to ensure the safety of the CubeSat and to protect the LV, the primary payload, and other CubeSats [4]. In this model, CubeSats only need to be compatible with the P-POD rather than being compatible with many different launch vehicles. Since its inception, the CubeSat Program at Cal Poly has worked extensively with LV providers to make LVs “CubeSat capable” by installing a P-POD onboard. CubeSats are swappable with one another between LVs with minimal concern about integration issues arising. The flexibility of this standard has proven to be the key to its success [5].

The P-POD with four 1U CubeSats can be seen in Figure 3. The figure shows three CubeSats for the Educational Launch of NanoSatellites (ELaNa)-1 mission with one CubeSat that is a backup in case a primary CubeSat isn’t ready. All CubeSats shown were designed to the CDS which facilitates ease of configurability.



**Figure 3 - P-POD with four 1U CubeSats during integration for the ELaNa-I mission**

Figure 4 represents the organizational structure that Cal Poly has setup to integrate CubeSats onto LVs [5]. Cal Poly is the only interface between CubeSat developers and launch providers and licensing agencies. This greatly simplifies the manifestation process of getting a CubeSat to orbit. Section 2.2 shows that

NanoRacks LLC attempted to follow a similar organizational structure for CubeLabs by taking the place of Cal Poly.

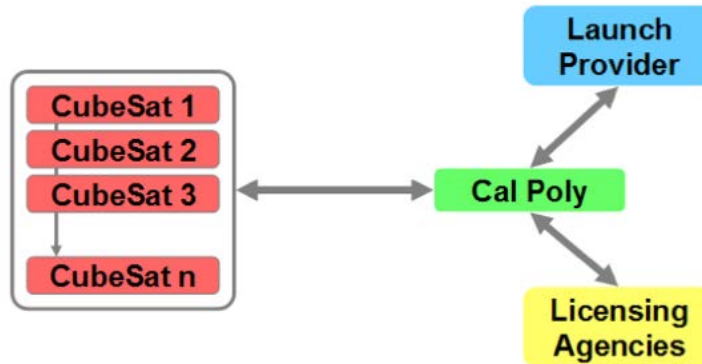


Figure 4 - CubeSat organizational structure by Cal Poly

### 2.1.1 Success of the CubeSat Standard

Since the definition of the CubeSat standard in 2000, the momentum of the CubeSat launches has grown significantly. What started out as a standard to help develop university aerospace programs and train their students has evolved into a program that is well established with multiple industry organizations developing CubeSat launchers and CubeSats and also providing the community with multiple launch vehicles for launch opportunities. Analysts have even stated that they are confident that CubeSats are a long-term trend with revolutionary implications for some sectors of the space industry [6].

The driving force of creating the standard was to define a common interface between a CubeSat and a deployment mechanism for deployment of CubeSats from a LV. This common interface, the P-POD, opens the door for standardized COTS parts to be built, which in turn, reduces the development cost and schedule for a satellite. The CubeSat's success can be attributed to how effective the P-POD has been in getting the satellites to orbit. Some of the benefits of the P-POD have been

documented to include: decrease launch costs, simplifies interaction between multiple developers and launch provider, flexibility in mounting, and flexibility in access to space [5].

In 2008, NASA Launch Services Provider (LSP) recognized the benefits of providing rideshares for educational institutions for CubeSats. They created the CubeSat Launch Initiative (CSLI) as an effort to bring P-PODs onto previously planned missions as auxiliary payloads so CubeSats can be launched after the satellite on the primary mission has been released in space. Since the inception of the CSLI, three missions, ELaNa-I, ELaNa-III, and ELaNa-VI have been launched with a total of 68 CubeSats accepted for current and future launches. LSP is currently developing a launch vehicle that will place CubeSats as the primary payload to orbit. The Nano-Launcher is scheduled to be completed FY2014 [7].

As previously mentioned, a standard deployment interface has facilitated the appearance of COTS CubeSat parts. This further decreases the cost of CubeSat parts for individual institutions and increases the reliability of parts as COTS parts can be tested on a wide variety of platforms and applications. One popular COTS provider, Pumpkin Inc., aims to provide an all-in-one kit for building a CubeSat [8].

Even with all of the advancements that the CubeSat program has made with providing many launch opportunities, a lot of barriers still exist [9]. These barriers are discussed in the next section.

### **2.1.2 Barriers to CubeSat Standard**

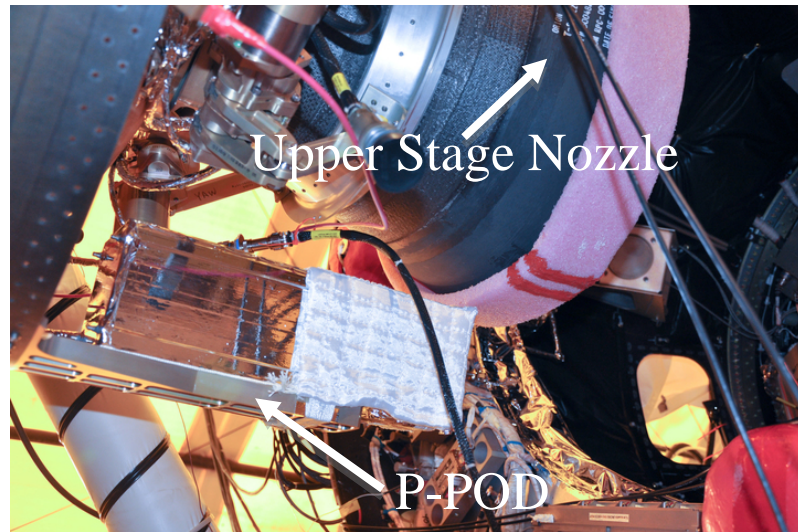
The CDS rigidly defines mass, size and material construction requirements for a CubeSat to conform to the P-POD. While this simplifies the manifestation process, it limits the types of payloads satellite designers can build. To date, the largest CubeSat that has been launched has been a 3U [6]. If a researcher wants to fly a

science instrument which slightly exceeds the dimensions of the CubeSat standard, this currently isn't possible. Work is being done to increase the launch capacity of CubeSats to 6U by ABC/NPSCuL, CRS and ORS/CubeStack, but these systems are still waiting to be launched [6] [10] [7]. The CubeLab standard, which is described later, is much more flexible and allows for larger payloads.

Anomalies due to the harsh launch and space environments are still fairly high. Factors that could be an issue include: radiation, structure/launch interface, thermal, communications, power, Central Processing Unit (CPU) or even launch failure. As of mid-2011, Universities that have not been designated by their government as a national center for spacecraft engineering research and development have experienced a 20% launch rate failure [6]. With the launch failure of ELaNa-1, it is apparent that even the CSLI isn't immune to launch problems. Given time however, this trend will get better as the launch and space environments for CubeSats are better understood.

Since CubeSats are often a secondary payload, the P-POD can be placed in a rather undesirable location. This was the case with the CubeSats that were part of NASA's ELaNa-I mission. The P-POD containing the ELaNa-I CubeSats was placed near the upper stage nozzle of the Taurus XL launch vehicle as shown in Figure 5. Unfortunately, the vibration levels at that location for the Taurus XL were not characterized before launch.

Vibration levels of P-PODs in LVs vary drastically due to different mounting locations and rocket behaviors making it difficult to define qualification requirements that encompass all available LVs. This causes CubeSat designers to over-design their CubeSats which could increase cost and schedule [9].



**Figure 5 - P-POD location on Taurus XL - credit: NASA**

Finally, since the CubeSats are free-flying, the designers are also concerned with power generation, a radio link to a terrestrial location, no return of science and withstanding the harsh environment of space. All of these items can increase budget and schedule and even put microgravity research out of the reach of some organizations. The NanoRack standard detailed in the next section aims to overcome some of the barriers listed here.

## **2.2 The NanoRacks Platform and the CubeLab Standard**

This section details the motivations behind the creation of the NanoRacks platform and describes the CubeLab standard. Real-time operations are also discussed which details how CubeLabs are operated on-orbit on the ISS.

### **2.2.1 NanoRacks Platform**

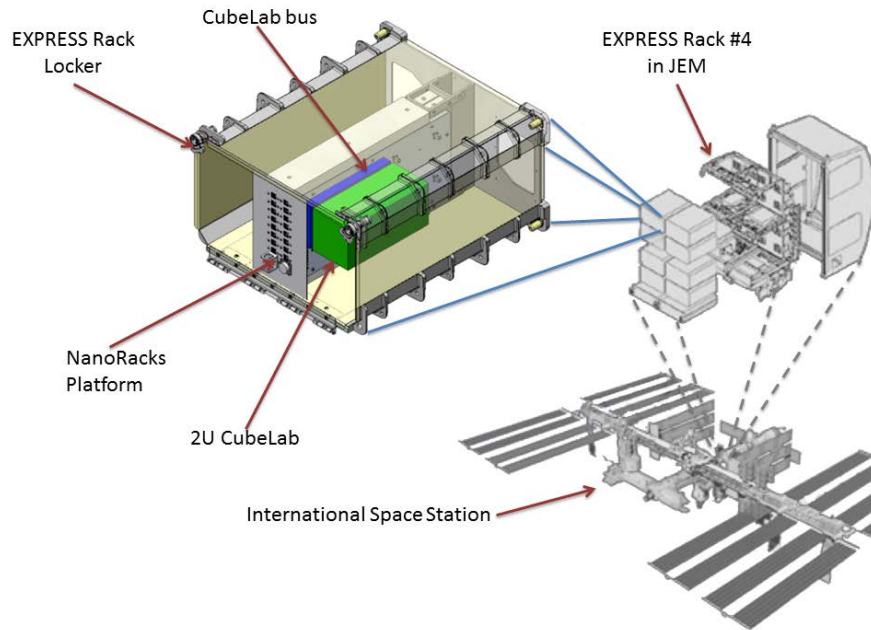
In 2005, in an effort to utilize the ISS to its full capability, the Congress designated the US segment of the ISS as a National Laboratory [11]. The purpose of this was to increase the utilization of the ISS by other federal entities and the private sector.

Unfortunately, the standard process to initiate research on the ISS was still a 20 month long process before an experiment can be launched. This can be a hindrance to researchers with tight schedules and limited budgets.

To help reduce this barrier, a “lean” payload integration process was introduced [12]. Under this new process, payloads undergo significantly shorter engineering verification via “Ship and Shoot” testing. This process determines testing requirements on a per-payload basis. NanoRacks LLC recognized this as a market for providing educational and commercial customers with rapid, repeatable access to the microgravity environment on the ISS. To take advantage of this new access, NanoRacks LLC signed an SAA with NASA in September 2009. As part of the SAA, NASA provides on orbit support and limited launch opportunities for NanoRack payloads [13].

To provide this access to customers, NanoRacks LLC needed to enable customers with an easy method of interfacing with the existing infrastructure on the ISS. The CubeSat form factor was recognized as a new industry standard that a growing number of researchers were becoming familiar with. Familiarity with the form factor was leveraged to develop a new standard for ISS payloads called the CubeLab. To make the new CubeLab standard compatible with the existing ISS infrastructure, the NanoRack platform was developed.





**Figure 6 - NanoRacks Platform and CubeLab exploded view**

The NanoRacks Platform serves as the interface between individual CubeLabs and the ISS, providing mechanical mounting points and electrical connections for power and data connectivity. A NanoRacks Platform is installed inside an Expediting the Process of Experiments to the Space Station (EXPRESS) rack locker that is housed in an EXPRESS rack. Figure 6 shows an exploded view of the installation configuration of a CubeLab and the NanoRacks platform onboard the ISS.

Each NanoRacks platform can accommodate 16U worth of CubeLab payloads in any configuration. CubeLabs of any size (1U, 2U, 3U, 4U, etc.) can be installed onto a NanoRacks platform. An exploded view of example configurations can be seen in Figure 2. The figure shows 15U worth of CubeLabs. The front panel of the NanoRack that is visible to the astronauts contains 16 USB type B connectors, a standard ISS power connector, an LED and a circuit breaker. The 16 USB connectors on the front panel provide the astronaut with a direct data connection between the CubeLabs that are installed on the NanoRack and an EXPRESS rack laptop computer (ELC). To command CubeLabs or to download experimental results, the ELC is

connected to the appropriate USB connector. The standard ISS power connector on the front panel connects to the 28V power available on ISS. The NanoRacks platform provides voltage step-down, distribution, filtering and isolation of the 5V connections to each of the 16 USB ports on the side of the NanoRack. The LED on the front panel provides a visual indication that astronauts can view to inform them that 28V is being supplied to the NanoRack Platform. The circuit breaker provides over current protection and a means to switch the power on and off [14].

In the fall of 2009, NanoRacks LLC partnered with Kentucky Space and the SSL at the University of Kentucky to design, build, and launch the first two NanoRacks Platforms and the first four CubeLabs. The first platform was on orbit and operational by April 2010. This short timeline was because NanoRacks took advantage of the “Ship-and-Shoot” payload testing process as mentioned previously. Further information about the current status of the NanoRacks Platform and CubeLabs can be found in Section 2.2.4.

### **2.2.2 The CubeLab Standard**

As CubeSats must conform to the CDS to be compatible with a P-POD, CubeLab developers must design their payload to conform to the “Interface Control Document Between CubeLab Modules and the NanoRacks Platform” (herein referred to as the CubeLab ICD) [15]. This allows their CubeLab to successfully interface with the infrastructure that exists on the ISS.

The CubeLab ICD allows CubeLab payloads to be heavier than a 1U CubeSat at 1.33 kg and defines a 1U to be larger than the CubeSat size of 10cm x 10cm x 10cm. It allows researchers to operate space hardware on-orbit without having an extensive background in power generation and radio communications. The main requirements which affect the work of this thesis include but are not limited to the following:

- Power supplied to CubeLab via type B USB port allocates 2W per USB port at 5 VDC .
- Data connectivity from CubeLab to ELC via type B USB port
- Communication shall only occur when initiated by an astronaut using the data cable to connect the ELC to the corresponding port on the NanoRacks Platform.
- CubeLab must contain a USB mass storage device which is accessible to astronauts for file transfer to the ELC.
- The CubeLab module/ELC interface shall not require any drivers to be loaded onto the ELC to operate nominally

The complete CubeLab ICD can be found on the UK SSL's website [15].

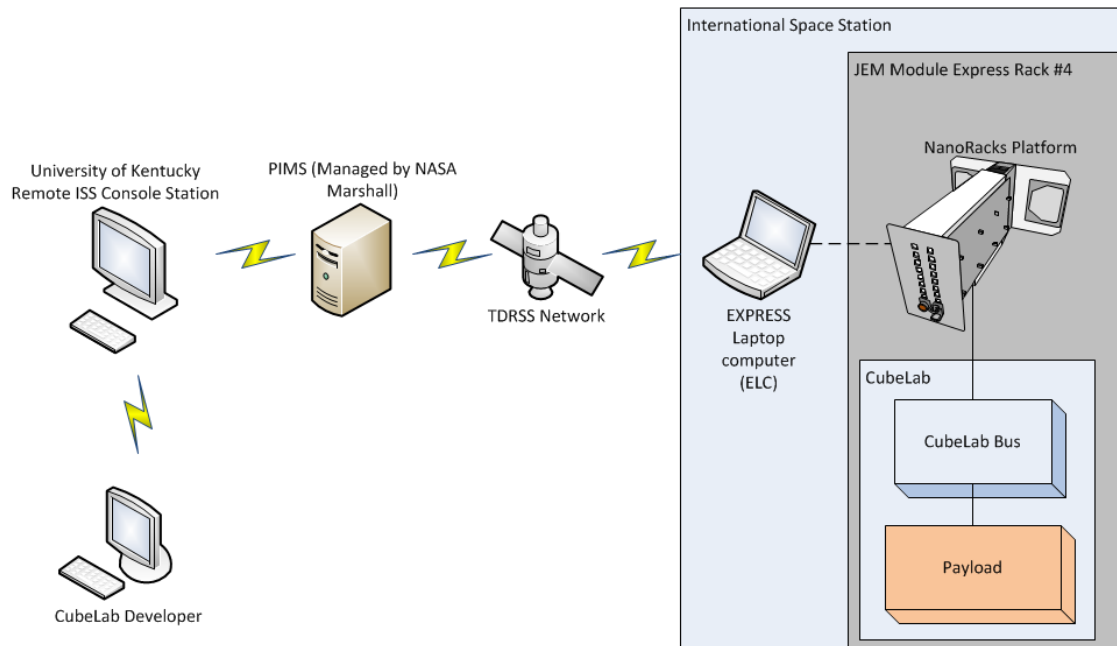
As seen above, the CubeLab standard requires that data stored on a CubeLab must be accessible through the USB connection on the NanoRacks front panel via the USB Mass Storage Device standard. This requires the implementation of a file system and a USB mass storage device stack. This is typically beyond the capability of some researchers. This issue is one of the main motivations behind the AmesLab bus which is further detailed in Section 3.

### **2.2.3 Real-time Operations**

The SSL coordinates real-time operations of CubeLabs, including installation, activation, data transfer (upload and download) and deactivation, aboard the ISS with the Huntsville Operations Support Center (HOSC) at NASA Marshall Space Flight Center (MSFC). This is coordinated through a fully authorized remote console station that is physically located at the SSL. This operations center consists of a secure operation console tied into NASA voice loops, real-time astronaut and ground systems scheduling systems, procedure development and viewing tools, real-time telemetry feeds and live high-definition video feeds from the ISS [14]. Due to

astronauts' schedule being very time critical, the SSL is required to provide console support during all real-time operations onboard the ISS. The SSL works with CubeLab developers to provide astronauts with assistance if any issues arise during operations.

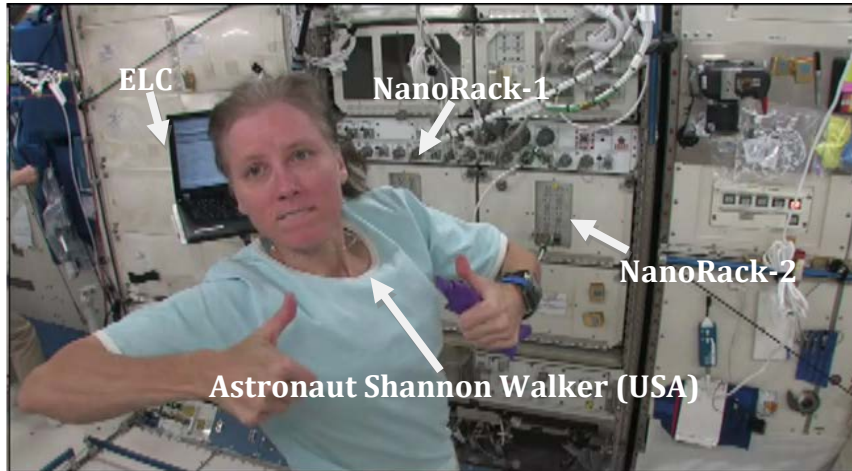
Through the coordinated effort of the CubeLab developer, SSL, NanoRacks and NASA, the most important CubeLab operation, a data transfer, can occur. The data path to/from CubeLab developers and a CubeLab payload aboard the ISS can be seen in Figure 7. Experimental data is generated on the CubeLab payload and is collected and stored by the AmesLab. Astronaut time is scheduled for data collection and the astronaut initiates data collection by plugging a USB cable from the ELC to the appropriate USB plug on the front panel of the NanoRacks Platform. Experimental data is transferred from the AmesLab to the ELC. The Payload Rack Officer (PRO) at HOSC downlinks the experimental data from the ELC, through the Tracking and Data Relay Satellite System (TDRSS) network, and into Principal Investigator Microgravity Services (PIMS). The SSL uses a secure connection to connect to PIMS to transfer the experimental data to their remote console station. The files are then securely transferred to the CubeLab developers. This procedure can be carried out in reverse if CubeLab developers wish to upload information to the AmesLab to command their payload [14]. This will be expanded further below in Section 4.3.



**Figure 7 - Data path from CubeLab payload to CubeLab developer**

## 2.2.4 Current Status

As of the date of this publication, there are two NanoRacks platforms installed on the ISS, currently operating researchers' payloads. NanoRacks Platform-1 was flown to orbit on Space Transportation System (STS)-131 and installed on July 12<sup>th</sup>, 2010 and NanoRacks Platform-2 was flown to orbit on STS-132 and installed on August 23<sup>rd</sup>, 2010. Astronaut Shannon Walker can be seen in Figure 8 post installation of NanoRack Platform-2. Both platforms can be seen in the image over her left shoulder. The ELC, which is used during commanding and downloading experimental results, can be seen over her right shoulder.

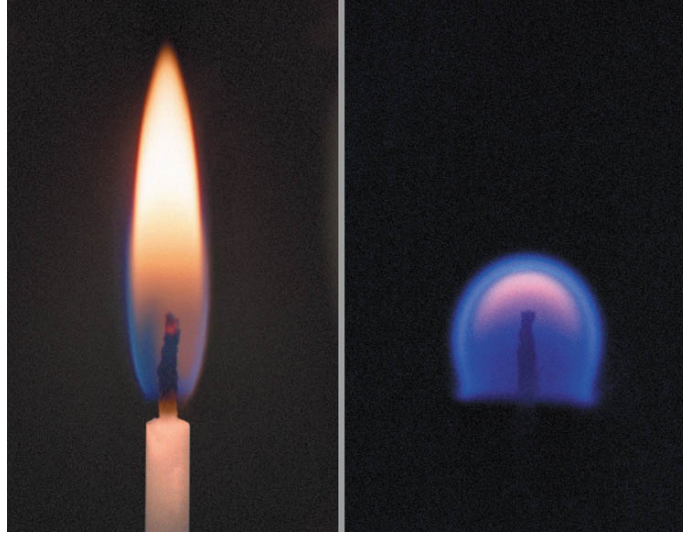


**Figure 8 - Shannon Walker after installation of NanoRacks Platform-2**

There have been five CubeLabs that have been designed and sent to orbit. The UK SSL has flown the following CubeLabs: Flash Incident Radiation Susceptibility Test Lab (FIRSTLab), CubeLab-2, CubeLab-3, and CubeLab-4. These were 1U CubeLabs which performed hardware radiation susceptibility experiments and tested the operation of the NanoRacks platforms. In addition, on orbit support has been provided for payloads from developers at a variety of universities and research institutes.

### **2.2.5 Microgravity Research on ISS**

Access to microgravity has provided a unique research opportunity to scientists over the years. Biological, chemical and physical systems that react in a known way terrestrially, can behave completely different in microgravity. The CubeLab form factor aims to make the microgravity environment more accessible to scientists to perform groundbreaking research. Popular areas of microgravity research include: microgravity biotechnology, microgravity fluid physics, microgravity materials science, microgravity combustion science (seen in Figure 9), microgravity fundamental physics, acceleration measurement program, advanced technology development program, and the microgravity glovebox flight program [16].



**Figure 9 - Effect of microgravity (right) on combustion process (credit: NASA)**

### **2.2.6 Comparison of CubeSat and CubeLab Platforms**

The initial motivations of the architecture of the NanoRacks/CubeLab platform is to leverage the successes of the CubeSat listed in Section 2.1.1 and attempt to overcome the deficiencies listed in Section 2.1.2 by providing a reliable interface to the ISS.

The paradigm of launch and operation of a CubeSat and a CubeLab are a bit different, but the successes of the CubeLab standard can be replicated. Launch costs are being kept in the range of CubeSat launches. The interaction between CubeLab developers and the launch provider is being simplified by NanoRacks [13]. NanoRack's customers have access to a wide variety of launch vehicles that take cargo to and from the ISS. The CubeLab standard is very flexible in terms of mounting for transfer to the ISS. CubeLab modules are shuttled to the ISS using soft stow bags which provide a very safe ride compared to typical satellite launches. NanoRacks can also provide flexible access to space as it allows upmass on a wide variety of launch vehicles.

A very popular CubeSat COTS architecture is the CubeSat Kit (CSK) sold by Pumpkin Inc. It was recently reported that shipments of the CSK to the CubeSat community is over 200 units [8]. The CSK features a USB type-B interface for additional power and data communication during development. The AmesLab bus is equivalent to the CSK for the CubeLab platform.

Popularly used power and data interfaces of a CubeSat were also brought to the CubeLab standard. Since this interface is familiar among so many international CubeSat teams, this interface was chosen for the power and data transfer for the CubeLab standard. This is an attempt to leverage the familiarity of the connections that exists out in the community. The difference between the CubeLab and the CubeSat is that the CubeSat only uses those connections during development whereas the CubeLab utilizes the USB connection throughout the entire mission timeline. As mentioned in Section 2.2.2, the data connection on the USB interface of the CubeLab must conform to the USB Mass Storage Device class.

The strict physical requirements of the CubeSat standard set the stage for COTS parts to be developed and adopted by organizations that don't have the knowledge or time to develop their own components. COTS parts provide a good starting architecture for CubeSat designers to easily and reliably build a CubeSat bus and interface a payload with it. The aim of the SSL AmesLab bus is to replicate same type of success that Pumpkin Inc. is seeing with the CSK. If this type of product can be replicated for the CubeLab architecture, CubeLab developers will have a significantly easier path to get their payload on the ISS to begin testing.

### **2.2.7 Alternative technologies available**

Currently, The CubeLab kit by Pumpkin Inc. is the only COTS product on the market that duplicates some of the functionality of the AmesLab bus [22]. The CubeLab kit provides a PIC based system to provide USB On-The-Go (OTG) functionality which allows the NanoLab to act as a host for slave USB devices to be attached. This in-turn



requires the researcher to plug in a USB device or to add an SD breakout board for storage capacity.

## **2.3 SPA Architecture**

As mentioned in Section 1.3, the SPA architecture was created by AFRL as an effort to bring a PnP-based system to the satellite ecosystem. The SPA architecture defines the following: *SPA components, SPA interfaces, ASIMs, SPA Networks, SPA systems, SPA middleware, Ontology and System Conventions* [1]. SPA hardware is referred to as a device or a component and SPA software is referred to as an application. The properties of devices and applications are described in the eXtensible Transducer Electronic DataSheet (xTEDS). SPA interfaces define the physical layer between devices and “handshaking” protocols that must be followed to facilitate data exchange between devices. The purpose of the ASIM is to provide an interface between non-SPA compliant spacecraft components and the SPA network. It generally contains a processor that handles the translations and memory storage necessary to store the xTEDS. ASIMs are described in more detail in Section 2.3.2. More information about xTEDS can be found in Section 2.3.3. SPA components are hardwired together through the use of a hub or router to form a SPA network. The physical layer of SPA networks are defined by the SPA interface being used, SPA-O, SPA-S, SPA-U or SPA-1. As shown in Figure 10, SPA-O has the fewest number of devices but also the fastest data rates, with SPA-1 being on the opposite end of the spectrum. SPA middleware is defined as a software component on the SPA network which operates the “discovery and join” mechanism for new devices. This software is typically referred to as the SDM and is described more extensively in Section 2.3.1. When designing a PnP system, all devices must share and understand a common “machine language”. The SPA architecture provides this common language by using a common ontology and system conventions. More information about how a common ontology is used in SPA can be found in Section 2.3.3.

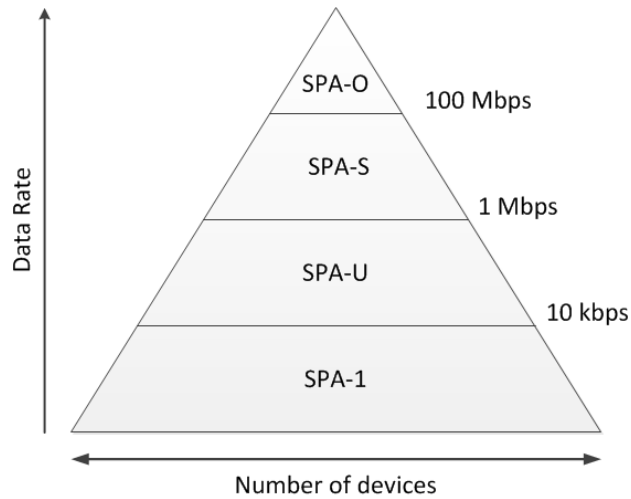


Figure 10 - Pyramid comparing devices and data rates in SPA devices [1]

A SPA system is a network of SPA components. Networks of differing SPA interfaces can be created through the use of a SPA bridge. The SPA network can consist of an entire spacecraft platform or just a subset of a larger non-SPA network. An example SPA network topology can be seen in Figure 11. All boxes, A-F can represent SPA endpoint components.

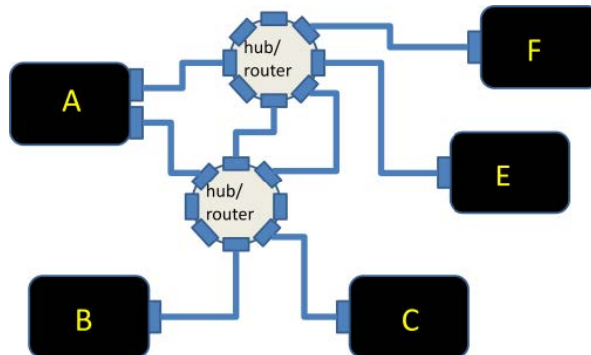


Figure 11 - SPA network showing use of hub/router [1]

### 2.3.1 SDM

As previously mentioned, the SDM is a piece of software that provides a set of services that allow SPA devices to discover and join a SPA network. The SDM exists

in a SPA network topology as an endpoint as shown in Figure 11 as any one of the blocks labeled A-F. An overview of the SPA architecture with the pieces that have been described in the paragraphs above can be seen in Figure 12. SPA Devices are shown on the bottom of the figure and are labeled “Camera”, “Thermometer”, etc. The SDM and interior processes are shown in the yellow box, labeled “Satellite Data Model”. Elements in the top two levels in the figure, “Applications” and “Mission code/scripts”, use data from the SPA devices to carry out the mission goals. The main objective of the SDM in this form is to provide a path between “consumers” of information with “producers” of information.

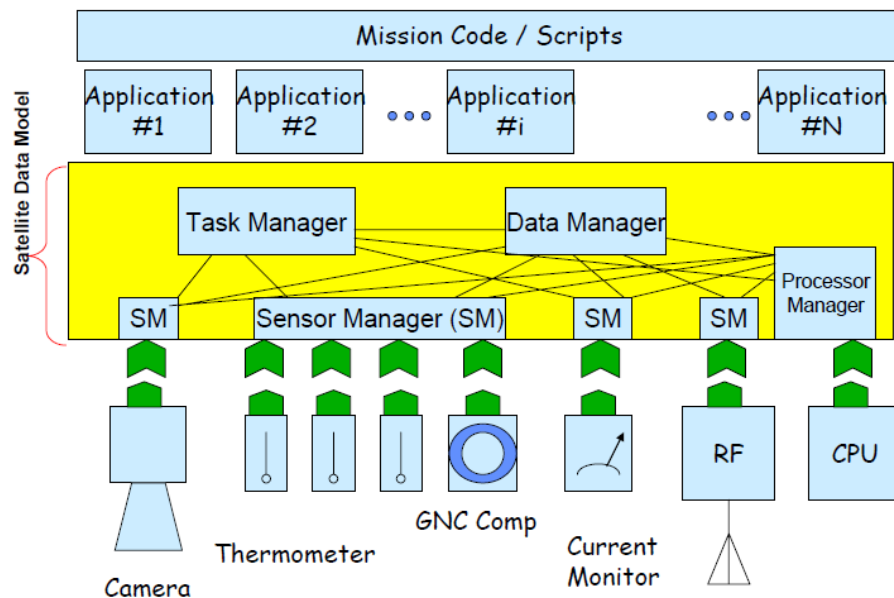


Figure 12 - SPA Architecture showing SDM [1]

Currently, the SDM is only compiled to operate on the Linux and VxWorks operating systems. Linux and VxWorks operating systems require complex processing architectures which aren't suited well for low power electronics. This version of the SDM is incredibly robust and provides many services which are far and above what is necessary to operate a SPA-1 network effectively. The main goal of this thesis is to remove unnecessary features from the SDM to create a new version called SDM-

Lite in order to operate a SPA-1 network of devices using an 8-bit microcontroller [1] [2].

### 2.3.2 ASIM

Non-SPA satellite components cannot be added to a SPA network without having an explicit interface which follows the SPA architecture. To simplify the addition of these components to the SPA network, the concept of an ASIM was created as an interface device. Figure 13 shows how a legacy device can be connected to a SPA network through the use of an ASIM. The ASIM contains data storage which contains the capabilities of the non-SPA component. These capabilities are programmed into storage using an eXtensible Markup Language (XML) format that is easily understood by the SPA network called XTEDS. The ASIM is also programmed to be able to interrogate the device through a customized interface [1]. There are a few 3<sup>rd</sup> party ASIMs available on the market for purchase. Part of the work performed in this thesis was to add the 8051 architecture to the ASIM ecosystem.

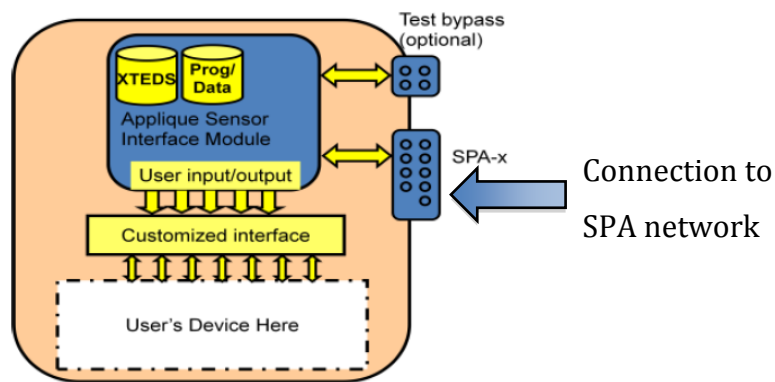


Figure 13 - ASIM provides an interface to the SPA network for legacy devices [1]

### 2.3.3 XTEDS

All components in the SPA architecture must share a common “machine language” to be able to communicate effectively. This common language must also use common terms to describe the same type of device. For example, all of the different names of a temperature sensor cannot possibly be understood by the SDM. Therefore, all temperature sensors must always be referred to as “temperatureSensor”. Terms that are used in the common “machine language” are stored in the Common Data Dictionary (CDD).

To describe the capabilities of devices and applications, the terms in the CDD are compiled together in an XML format called an xTEDS. The xTEDS format was based on the already existing IEEE 1451.4 standard for Transducer Electronic Data Sheets (TEDS) [17]. The TEDS standard defined protocol and interface for analog transducers to communicate digital communication with other TED sensors. An example xTEDS for a temperature sensor can be seen in Figure 14 [1].

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <xTEDS name="ExampleDevice" version="1.0">
3   <Device name="ExampleDevice" kind="temperatureSensor" />
4   <Interface name="ExampleInterface" id="1" >
5     <Variable name="celsius" kind="temperature" format="FLOAT32" />
6     <Notification>
7       <DataMsg name="GetTemperature" id="1" msgArrival="PERIODIC" msgRate="1.00" >
8         <VariableRef name="celsius" />
9       </DataMsg>
10    </Notification>
11    <Command>
12      <CommandMsg name="ToggleLED" id="2" />
13    </Command>
14  </Interface>
15 </xTEDS>
```

**Figure 14 - Example of a simple xTEDS for a temperature sensor [1]**

The next section presents information on the types of missions and companies that are actually using SPA technology. This includes both launched and yet to be launched missions.

#### 2.3.4 Current adoption of SPA

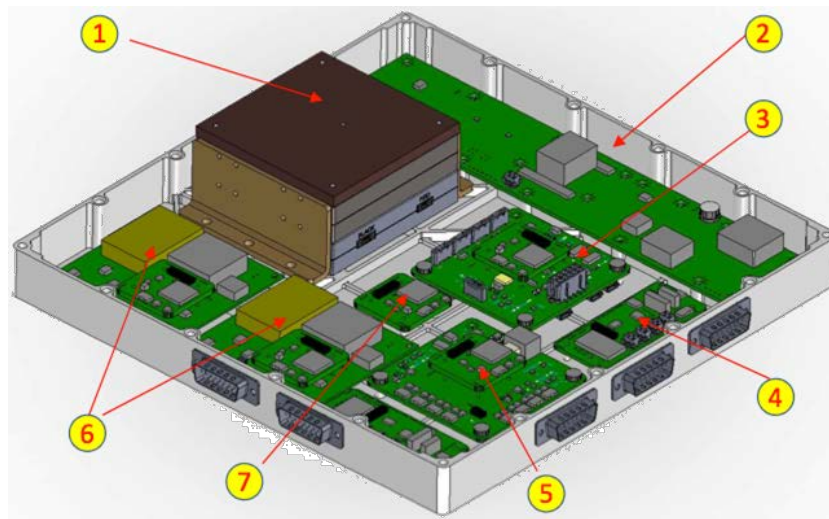
To date, only a few spacecraft have taken advantage of SPA technology. The first SPA based satellite to be designed was Plug-and-Play Satellite (PnPSat). It was designed to be constructed extremely rapidly using the design concepts of SPA that have been previously discussed in this thesis. As of 2007, the construction of PnPSat was assigned to the Responsive Space Testbed at AFRL's Space Vehicles Directorate [17]. Unfortunately, PnPSat was never launched.

Rapid prototyped Mems Propulsion And Radiation Test CUBEflow SATellite (RAMPART CUBESAT) is a CubeSat that contains an additional secondary circuit board with the intent of gathering radiation performance statistics on three types of SPA-1 PnP modules. There are three ASIMS onboard, two radiation hardened ASIMS, one made in the US and one made in Sweden, and one commercially available PIC. The orbit of this CubeSat is such that the apogee is 1200km and should provide an adequate radiation dose to test the radiation susceptibility of the ASIMS onboard [18]. RAMPART is in the process of being manifested on the Space Test Program (STP) to be launched in 2013.

QuadSat-PnP is a nanosatellite under development by a coordinated effort between University of Applied Sciences Bremen, OHB System, ÅAC Microtec in Sweden and the AFRL in the United States. This nanosatellite is the first in its class to be based completely on a PnP model, specifically SPA. Many PnP devices were developed specifically for this satellite and that helped facilitate a development of ASIM equivalent devices called a Remote Transceiver Unit (RTU). The nano-RTU and the  $\mu$ RTU are devices developed by ÅAC Microtec that enable non-US based space organizations to take advantage of the SPA PnP network. The nano-RTU is a SPA-1 interface module. It provides the same base functionality that an ASIM would provide for a SPA-1 network, and more importantly it provides a radiation-hardened architecture. The  $\mu$ RTU provides much more functionality over the nano-

RTU. It is capable of SPA-1, SPA-U, and SPA-S and carries much more processing power.

Figure 15 provides the physical layout of the PnP components inside the QuadSat-PnP. The hardware as labeled in the figure is as follows: 1. TDRS software-definable radio with SPA-U interface 2. Main power distribution unit 3. Distributed power control unit with four SPA-1 ports 4. Inertial measurement unit SPA-U node with  $\mu$ RTU interface 5. Distributed power control unit with four SPA-U ports. 6. Miniaturized rad-hard point of load SPA-1 node with nano RTU interface. 7. Nano-RTU [19]. From the captions mention above, it is clear that this satellite is packed with PnP technology. As of the publication of this thesis, there is no firm launch date for QuadSat-PnP.



**Figure 15 - PnP component layout inside QuadSat-PnP [19]**

COSMIAC is currently building a CubeSat, named Trailblazer-1, which serves as a proof-of-concept for SPA technology. This satellite is based on the SPA-1 standard and works to adapt existing COTS technologies to the SPA network. This will prove the feasibility of adapting commonly used satellite components to operate on the SPA network. The overall mission timeline of the satellite is to go from design to

delivery in under a year. This will validate the reduced schedule claims that are part of the SPA paradigm. A SPA-1 ASIM is being developed and is based on a Peripheral Interface Controller (PIC) microcontroller. The C&DH uses Pumpkin Inc's CubeSat kit and runs an SDM-Lite. The SDM-Lite handles component discovery, component registration, data centric queries, time distribution, and internal systems health monitoring and status reporting [20]. There are two SPA based payloads on the satellite, a dosimeter and a rapid prototyped Inertial Measurement Unit (IMU). Trailblazer is currently manifested on the ELaNa IV mission and is scheduled to be launched in 2012 [21].

COSMIAC's satellite architecture for Trailblazer-1 is very similar to the overall goals of what this thesis is trying to provide, an SDM that will run on a low resource processor and support a SPA-1 network of devices. In addition to this, COSMIAC's expertise in SPA due to their role of educating the space community by using the CubeFlow kit made the collaboration with them for this thesis work an easy decision.



## 3 AmesLab Bus

This chapter details the design description of the AmesLab bus and the motivations behind the bus. This work was not directly a part of the work in this thesis, but decisions were made in the design process which made it easy to make compatible with SPA. These decisions are reflected in the system requirements section. Applications using the AmesLab bus and alternative technologies that provide similar features as the AmesLab bus are also discussed.

### 3.1 System Requirements

The concept of the original design description of the AmesLab was to provide an interface for generic payloads to use the NanoRacks platform in the CubeLab form factor without having to comply with the USB mass storage device requirement. In 2010, the SSL had the opportunity to work with NASA Ames Research Center (ARC) to provide an interface to the NanoRack for their 2U CubeSat payload called MisST. The MisST mission was a free flyer 2U CubeSat payload which studied the effects of microgravity on *C. Elegans*. In addition to providing an interface to the NanoRack, this mission required the use of a high power imager and had strict thermal constraints. This led to the development of an upgraded power system for the AmesLab which is capable of charging a Nickel-metal Hydride (NiMH) battery pack that can be used to provide higher instantaneous power than the power limitations listed in the CubeLab ICD. To fulfill the very specific temperature requirements, the SSL designed a thermal control system which involved using peltier coolers to remove heat from the MisST payload so the experiment could achieve a target temperature.

In addition to the design requirements for AmesLab that support the NASA Ames payload in the CubeLab form factor, there are additional requirements which facilitate support for SPA-1 based payloads and several derived requirements from

the CubeLab Interface Control Document, the USB standard, and the SPA-1 standard. The requirements are summarized below and the following sections discuss the solutions to the major challenges associated with the requirements.

### **3.1.1 CubeLab Specification and NanoRack requirements**

1. The CubeLab and its payload shall consume less than 400mA continuous per USB port at 5Volts DC from the connection to the NanoRack.
2. The AmesLab shall appear as a USB Mass Storage Device (MSD) and adhere to the File Allocation Table (FAT) file system when a laptop is connected to the NanoRack front panel.

### **3.1.2 Interface Requirements**

3. The AmesLab shall provide a mechanical attachment point for payloads.
4. The AmesLab shall not interrupt USB enumeration during astronaut data transfer.
5. The AmesLab shall not interrupt payload operation (generic or SPA devices) during astronaut data collection.
6. The AmesLab shall provide an interface which could be used as a SPA-1 network interface for a SPA-1 device.

### **3.1.3 Functional Requirements**

7. The AmesLab shall be capable of interrogating generic payloads (possibly SPA-1 devices) to obtain data from the payload and log it for later downlink.
8. The AmesLab shall be capable of accepting uploaded command files from the astronaut interface.

### 3.2 Hardware

The hardware design for the AmesLab can be split into the Command and Data Handling (C&DH) and Electrical Power System (EPS) components. The development of the C&DH component of the AmesLab started in the fall of 2010 with the development of the requirements listed in Section 3.1. In an effort to fulfill these requirements, an implementation was designed which is represented by the block diagram shown in Figure 16.

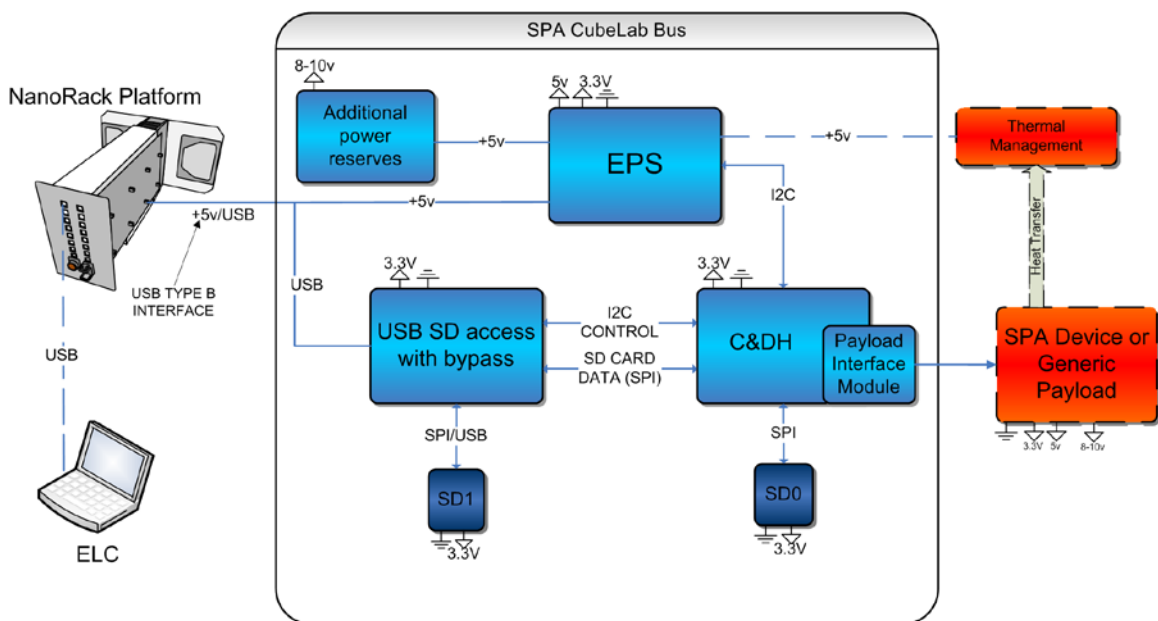
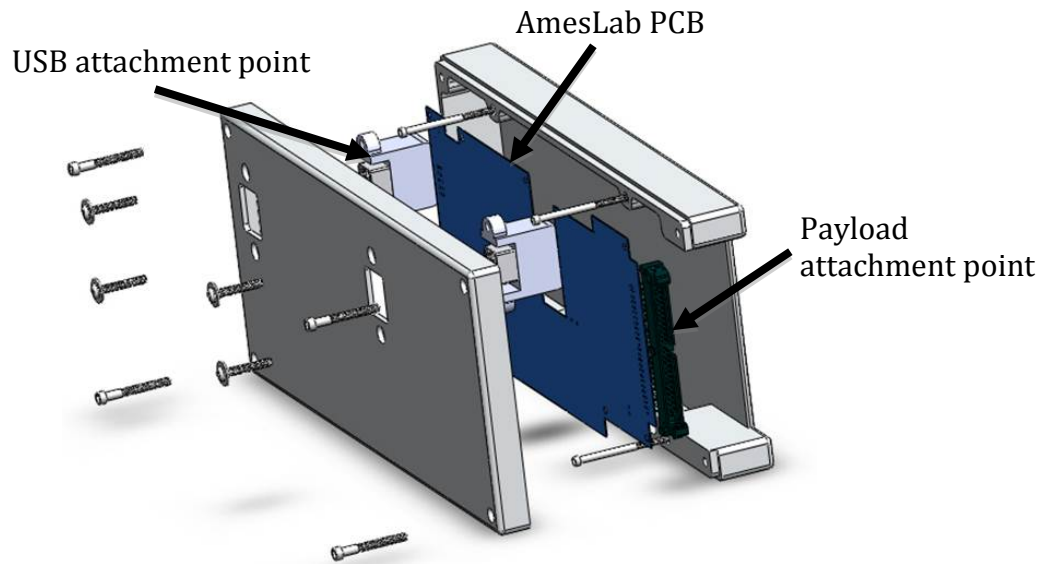


Figure 16 - Block diagram of the AmesLab electrical interfaces

The Silicon Labs C8051F120 was chosen as the main processor to control the C&DH block of the AmesLab design. To interface with the SD card used as the MSD in requirement 2, the microcontroller has to use a FAT filesystem. A filesystem was chosen (more in section 3.3) that runs on the 8051 using very little overhead and requiring little coding effort on the part of the SSL. Requirements 4 and 5 require an element to monitor the astronaut activity and modify the system based on what the

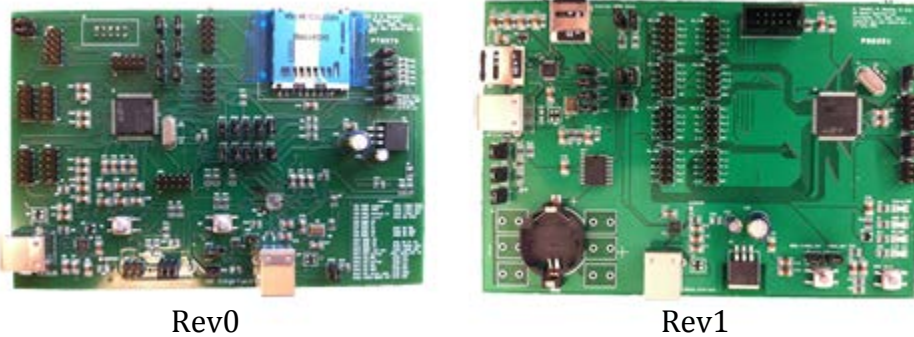
astronaut is doing at any given time. The monitoring and reaction to that activity is also provided by the 8051F120. The 8051 has many different communication peripherals (Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), Universal Asynchronous Receiver/Transmitter (UART)) to interface to generic payloads, including SPA devices. Additionally, General Purpose Input/Output (GPIO) pins are available which could theoretically be used to interface with any number of devices.

The AmesLab must also provide a mechanical attachment to the NanoRack Platform, per requirement 3. Given the power requirements and the 2U size of the original NASA ARC payload, a 2U size was chosen for the bus. This facilitates an additional 400 mA at 5VDC power and two mechanical attachment points through the form of two USB type-B plugs which stick out of the side panel of the NanoRack Platform. Since the force of gravity can be viewed as negligible on the ISS, the friction of the USB connection between the NanoRack Platform and the AmesLab are enough to hold the CubeLab in place. This has been tested numerous times on orbit with no adverse side effects. A mechanical enclosure has been designed which encloses the AmesLab board and provides mechanical attachment points for SPA-1 payloads. An exploded view for the design of the enclosure can be seen in Figure 17 below.



**Figure 17- Exploded view of AmesLab enclosure**

The hardware design for the AmesLab board was created using an iterative design approach through a collaborative effort with all of the students in the SSL. This design process led to the creation of 3 different revisions of the AmesLab board. While in the design and prototyping phase, the nomenclature for the board was “CubeLab development board”. This went through two different versions, Rev0 and Rev1 as shown in Figure 18.



**Figure 18 - Rev0 and Rev1 of AmesLab Development board**

Once the components were chosen, I developed the schematics and board layout that were used for the Rev0 board. Numerous headers and jumpers were added

into the circuit for development purposes. A quick board layout was generated and fabricated.

After the completion of the power board design and the hardware design was proven feasible with thorough testing, I started the process for designing the final board revision, a flight board, as shown in Figure 19. Several students in the SSL took over and finished the layout effort. The flight board removes all of the components that were required for testing and integrated the power board into the design of the AmesLab. The flight board is very reliable as the design was thoroughly tested on the two versions of the development board.

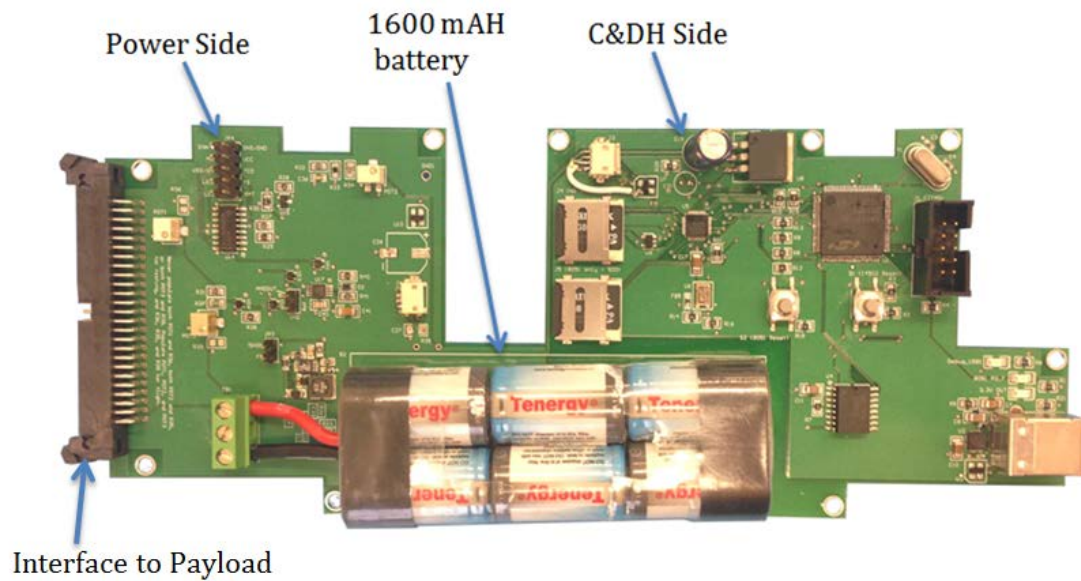


Figure 19 - Populated flight board for AmesLab

### 3.3 Software

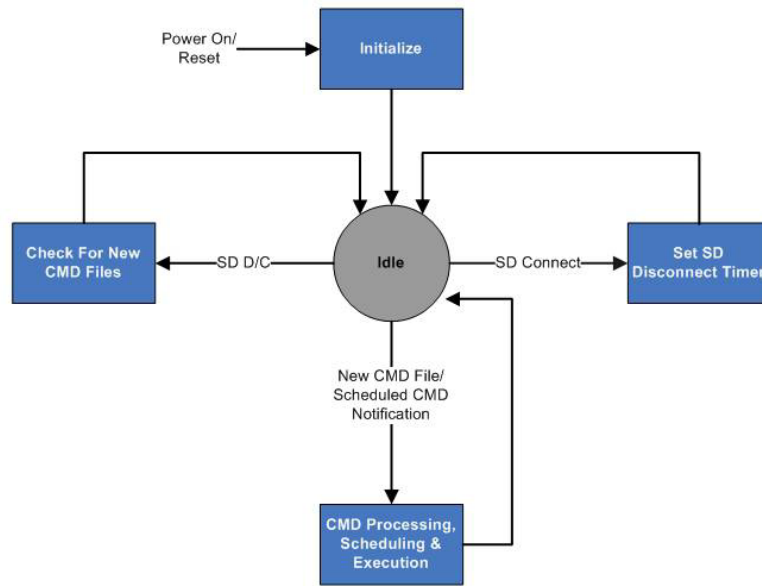
The initial software effort was centered on writing software that would allow a free-flying satellite, Microsatellite in-situ Space Technology (MisST), from NASA Ames to be operated within the NanoRacks facility. This effort included writing software for

the onboard 8051F120 that will satisfy the requirements listed in Section 3.1 above. This software effort includes:

- Drivers for using I2C, UART, and SPI (req 6 and 7)
- A file system implementation (req 2)
- Libraries which allow the use of SD cards (req 5)
- Software architecture to read uploaded command files (req 8)
- Storage of experimental data (req 7)

The top-level software flowchart that was developed to allow commanding of the AmesLab from a terrestrial location (see Figure 7) can be seen in Figure 20 below. When an astronaut plugs the ELC into the NanoRack, the AmesLab provides an interface to the SD1 card for the astronaut. To do this, the AmesLab must comply with the USB Mass Storage device class and the SD1 card has previously been formatted with the FAT16 file system. This allows the astronaut to read and write to the card without the need for special drivers for the ELC. Since SD1 has been formatted with the FAT16 file system, the only way for the 8051F120 to read or write data to the card is to do so through a file system. File system drivers had to be developed for the 8051F120 in order to easily write and read to the SD1 card without worrying about data corruption.

The SSL researched various file systems (FATFS, Secure Digital Fat16 Driver, smxFS, and an application note from Silicon Labs) to use on the 8051F120 to interact with SD1. Ultimately, due to cost and limited functionality of other file systems, FATFS was chosen. FATFS provided the highest amount of file system calls to the 8051 and was still within the budget (free!).



**Figure 20 - Top level software flowchart for AmesLab software**

Per requirement 8, the AmesLab software also must be capable of accepting upload scripts from the astronaut and transferring them to the AmesLab. This could include scripts that will command the payload or firmware updates to the AmesLab software. The AmesLab monitors astronaut interaction to react when an astronaut plugs the ELC into the front panel. Due to the persistent power on the USB interface, the AmesLab must detect a data transfer initiation by an astronaut using the NanoRack USB data lines to fulfill requirement 4. In order to not require special software or driver installation on the ELC, the AmesLab is expected to halt experimental data recording and respond to astronaut interaction and appear as a USB Mass Storage Device to the ELC. This is done by monitoring the state of the USB device that is plugged into the CubeLab through the NanoRack. The device can either be in 'suspend' or 'resume' status. If there is any change in this state (ie an astronaut plugs or unplugs the USB cord), the MAX14502 sends an interrupt to the C&DH. The C&DH responds by switching the MAX14502 to card reader or pass through mode to allow the appropriate device to interface SD1.



## 4 SPA-1 network on a CubeLab

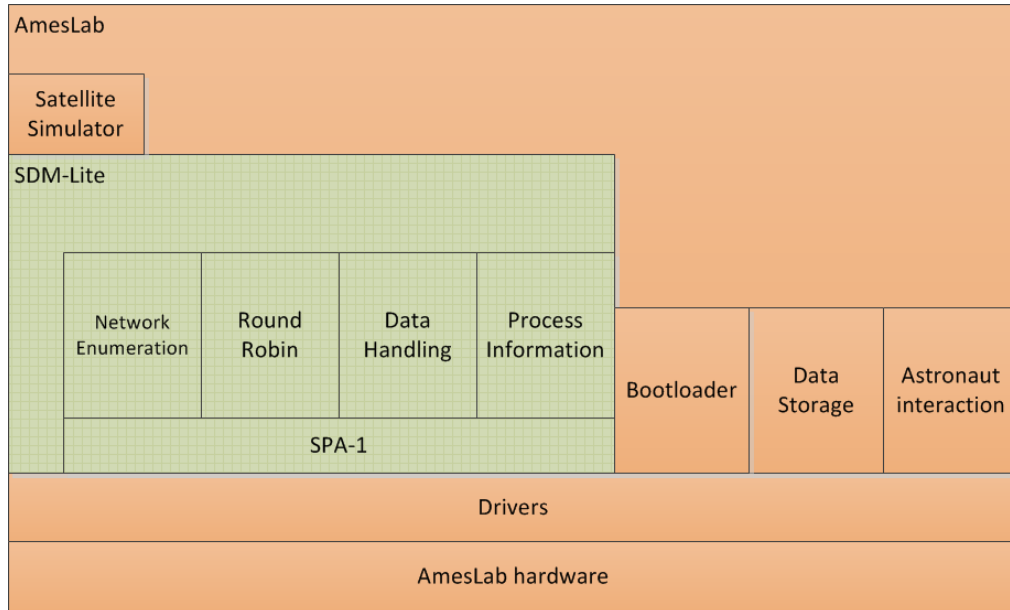
This chapter describes the effort of adding SPA-1 capabilities to the AmesLab by providing SDM-lite functionality to the C8051F120 microprocessor on the AmesLab is detailed. Hardware is also added to the AmesLab that will facilitate the safe addition of I2C devices using an external interface. Finally, an operation plan is developed which should be followed to get new SPA devices to be installed in the NanoRacks AmesLab facility. The AmesLab with SPA-1 capabilities is herein referred to as SPALab.

### 4.1 Software

COSMIAC is developing a CubeSat called Trailblazer-1 that was mentioned previously in this thesis [22]. This CubeSat aims to bring SDM functionality for SPA-1 networks to a low resource, low power ARM microprocessor. This modified SDM for low resource embedded systems is called SDM-lite. The available resources for the 8051 on the AmesLab and the Advanced RISC Machines (ARM) chip in the Trailblazer-1 CubeSat are very similar. In way of this, the design of the SDM-lite that runs on both platforms is nearly identical. The software flow chart for the SPALab, in 0, was developed through an effort with COSMIAC. After the flowchart was developed, the software for both platforms was developed independently.

To provide a SPA-1 network in the CubeLab form factor, the software for the SPALab took two major efforts (SDM-lite and AmesLab software) and merged them into a common software package. These two efforts are clearly delineated in the software flow charts in 0 and in Figure 21. The blocks pertaining to the SDM-lite are shown in green. The blocks pertaining to the AmesLab functionality are shown in orange. This thesis work specifically provides the software functionality for the SDM-lite blocks while providing an integration plan for the AmesLab blocks.

The next two major sections, Section 4.1.1 and 4.1.2, discuss SDM-lite functionality and the AmesLab software, respectively.



**Figure 21 - Software layering diagram for SPALab**

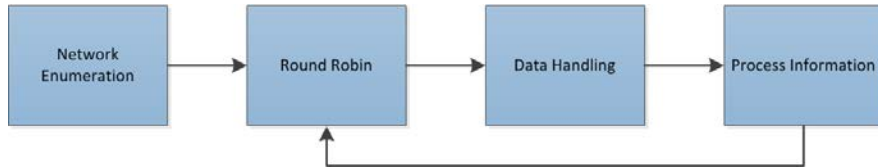
The software for the SDM-lite was written in the C programming language using  $\mu$ Vision 4 Integrated Development Environment (IDE) for the development environment. Tortoise Subversion (SVN) was used as a repository to keep track of software changes and to ripple those changes out to the entire software team.

#### **4.1.1 SDM-lite functionality**

The main purpose of SDM-lite is to provide the mechanism referred to as “discovery and join” for SPA-1 devices. The purpose of this mechanism is to detect the existence of new components on the SPA network and provide the ability to query the “services” provided by these components (as described by the XTEDS) [1].

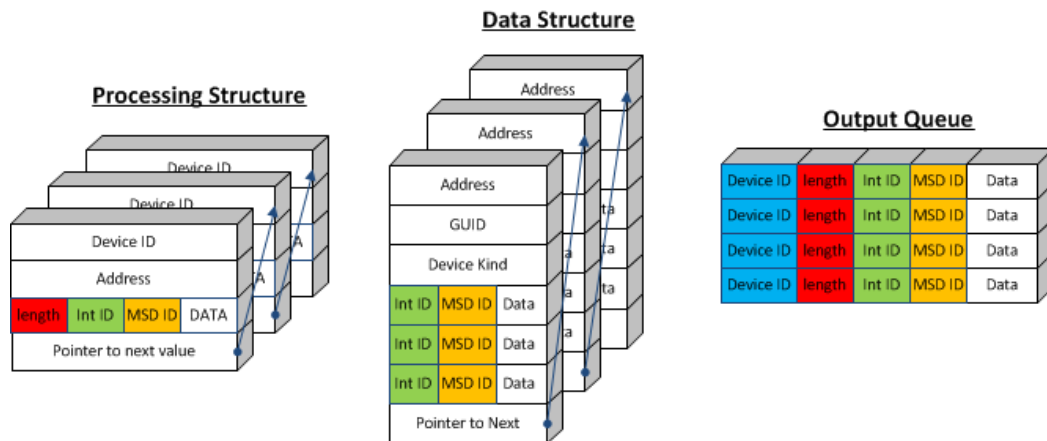
The SDM-lite functionality can be broken up into three main components (round robin, data handling, and processing of information) and two main responsibilities

for each component (controlling data on the bus and providing plug and play mechanisms for new devices). The high-level software flow for SDM-lite can be seen in Figure 22 and a more in-depth flowchart can be seen in 0.



**Figure 22 - Top-level software flowchart for SDM-lite functionality**

The three components mentioned use three different data structures to pass information to/from ASIMs and store data about the ASIMs on the C&DH. These three data structures are the processing structure, the data structure, and the output structure. The processing structure is a buffer for information that is read from each ASIM. The data structure is the data storage on the C&DH that corresponds to the xTEDS from each ASIM. This is also intended to store the current state of the device upon the last read cycle. The output structure is a buffer for information that will be written to ASIMs on the next round robin. These three structures can be seen in Figure 23.



**Figure 23 - Structures used in SDM-Lite for data storage**

In a normal SDM environment, these structures would be dynamic and would get created and destroyed as necessary. Unfortunately, dynamic memory allocation is heavily dependent on an OS as it is responsible for allocating space based on memory that is available in the system. In the case of SPALab, there is no OS running. Therefore, dynamic memory allocation would indeed produce undesirable results including unintended overwritten data. For this reason, the xTEDS for each SPA device is stored on the firmware (FW) of the C&DH itself and all structures for storing data are statically created at compile time. When a SPA device is plugged in, the xTEDS is retrieved and compared with the xTEDS that already exists on the C&DH. This ensures that the SPA device uses the appropriate entry in the data structures.

#### **4.1.1.1 Network Enumeration**

The network enumeration functionality in the SPALab serves the following purposes: registering SPA devices on the network when they come online and maintaining the data structures with the current I2C address associated with the device.

The software flowchart for the network enumeration process can be seen in 0. The “not enumerated” branch goes through the process of registering the SPA device on the network with the SDM-lite. The registration process is a series of sending and receiving commands to/from an ASIM to interrogate such things as the Global Unique Identifier (GUID), the version of software, the status of the ASIM, and the xTEDS. Once all enumeration data are read from the ASIM, the appropriate values are stored into the data structure for that ASIM. The GUID is a unique identifier associated with ASIM and is hardcoded into the ASIM code. It is used by the SDM-Lite to reference and contact each device.

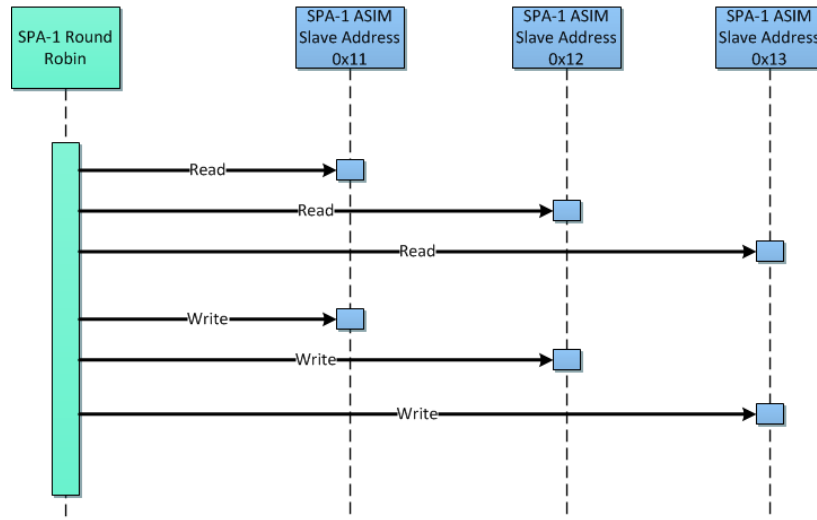
Parsing the xTEDS would allow the SDM-lite to discover the capabilities of the ASIM on the fly. Initially, the SPALab will not allow parsing of the xTEDS because is not

implemented in this version of the software. In the meantime, the xTEDS from the ASIM is compared with the xTEDS for that device that is stored on the SDM-lite. This ensures that the SDM-lite has the capabilities through the appropriate applications for nominal operation for the ASIMS that are plugged into the network.

A re-enumeration process is also called every 100 round robin cycles to ensure the network addresses are up-to-date. This process starts at the lowest I2C address, retrieves the GUID from the ASIM, updates the data structure corresponding to that GUID with the current I2C address. Cycling through all registered I2C addresses completes the process.

#### **4.1.1.2 Round Robin**

The SPA-1 protocol states that SPA-1 ASIMs shall be written to and read from in the manner shown in Figure 24. The sequential writing and reading from ASIMs is a process called Round Robin and is designed to give the ASIMs enough time to prepare data between the write command and the read command. Due to the nature of I2C, each ASIM is read from and then written to in that order. This process ensures the correct devices are read from and allows adjusting to be made before the write cycle if necessary.



**Figure 24 - Round Robin used to communicate with SPA-1 ASIMs**

#### 4.1.1.2.1 Controlling data on the bus

The round robin is responsible for reading and writing data and commands to and from ASIMs that are on the bus. The software flow for the round robin can be seen on the round robin page in 0. By following that software flow, it is shown that the round robin process starts by sending a read command to the first ASIM that is at I2C address 0x11. The data structures are searched to find a device with the I2C address of 0x11. Assuming the device is found, the I2C bus is read at that address and the data are stored into a processing structure tagged with the current Device ID. Assuming normal operation, after reading the data from that ASIM, the first decision in the flowchart, “data structure contains device at that address” will be true. Abnormal operation of the bus is discussed in section 4.1.1.2.2.

The slave address is then incremented and the process is repeated until all ASIMs are read and their corresponding processing structures are updated. Following the completion of a read from all available ASIMs, the data structures for the ASIMs are realigned with the processing structures associated with the ASIMs. The write cycle of the round robin is discussed in Section 4.1.1.3.

#### 4.1.1.2.2 PnP mechanisms for devices

The round robin is also responsible for searching for new devices on the bus. The previous section mentioned the round robin functionality during normal operation but this section covers how the round robin behaves when an ASIM goes off line and comes back online. When this occurs, the address resolution process for the ASIM (discussed in Section 4.4) may cause the ASIM to obtain a different address than it had before it went offline. The slave address stored in the data structure for that ASIM is now incorrect and needs to be updated. It is critical to the operation of the SDM-lite that the data structures for each device always display the current slave address of that ASIM.

The “no” branch of the “Data structure contains devices at that address” decision block handles the abnormal operation as mentioned above. This requests the GUID from the unknown ASIM and searches through all of the data structures to find a match. If a match is found, the data structure is updated with the current address of the ASIM and all available data from an ASIM are subscribed to. This subscription request triggers the ASIM to start outputting data messages to the bus to be read by the SDM-lite. If a GUID match isn’t found, the enumeration process is executed and the ASIM is registered on the SPA-1 network as a new device.

#### **4.1.1.3 Data Handling**

##### 4.1.1.3.1 Controlling data on the bus

The data handling slide in 0 is the central point at which all information passed is located so that the data to be processed and commands to be passed can be housed for future operations. The data handling procedure starts out by updating the highest address that was read from on the previous round robin read cycle. Since the round robin read cycle temporarily stores the received information into a

processing structure, updating the highest address is a matter of searching through the processing structures and finding the highest address.

Next, the process starts writing packets that are in the output queue to the SPA-1 bus. The first packet in the output queue is read and the device ID from that packet is found in the data structures. The I2C slave address associated with that Device ID is stored and the packet is sent to that I2C address. If the address isn't found, the packet isn't written to the network. The process repeats until all packets in the output queue are serviced. The output queue is then cleaned to ensure future output packets will be outputted.

Finally, a round robin counter is incremented. If this counter exceeds 100, the bus is re-enumerated by calling the network enumeration operation. The reasons for this are explained in the next section.

#### 4.1.1.3.2 PnP mechanisms for devices

The data handler provides PnP mechanism for devices in two ways, using Device ID for the output queue and re-enumerating the bus after 100 round robin calls. By using the Device ID only for the packets in the output queue, this allows SPA devices to have different I2C network address in the event of a power cycle of the device. If a SPA device goes offline and online the I2C address for that SPA device will automatically be updated in the data structure for that device. Since the data handler retrieves the I2C address from the data structure, this ensures the packets in the output queue will always have the most recent I2C address for each SPA device. Additionally, in the case that two ASIMS go offline and trade I2C network addresses, the existing software flow has no way to handle that. The re-enumeration of the bus every 100 round robin cycles will ensure the data structures get realigned if I2C addresses get swapped.



#### 4.1.1.4 Processing information

The information processing component could be thought of as the application layer of SDM-lite. Strictly speaking, it isn't part of the SDM-lite layer but a layer that resides on top of it as shown in Figure 12. It is assumed that every ASIM has an application associated with it, which facilitates processing. These applications are represented on the information processing page of 0 as "Device X Application". These applications will be unique to each SDM-lite configuration and can be changed by using the FW updater as mentioned in the next section. Following the processing of information by each of the applications, the information on the bus is either written to an ASIM data structure, sent to the output structure to be written to an ASIM on the next round robin or both.

##### 4.1.1.4.1 Controlling data on the bus

The information processing component of software is responsible for operating on the information that is passed on the bus. As seen in the information processing page in 0, the information process component starts by reading the first entry from the processing structure. The first process is routed to the appropriate application by referring to the "Device ID" of that process. The application performs analysis or processing on the data and then has the option to output a message to another module or to store the data in the data structure of another ASIM. After the first process is read and completed, the procedure is repeated until all processes in the processing structure are read and processed.

##### 4.1.1.4.2 PnP mechanisms for devices

Since the applications output to a device ID and not to a slave address directly, this preserves the plug-and-play nature of the SPA Network. This ensures that information will be routed to the proper endpoint and covers the case of an ASIM going through re-enumeration in the middle of the round robin.

### **4.1.2 AmesLab software**

The SPALab also contains software components that are derived from the AmesLab software. The primary purpose of this software is to fulfill the software requirements that are listed in Section 3.1. This software can be split into four main components: data storage, bootloader, interaction with the astronaut through the NanoRack, and the satellite simulator. The AmesLab software isn't completely integrated into the SPALab software project files as the main focus of this thesis is to provide the SDM-lite functionality to the AmesLab and to provide a plan for AmesLab integration.

#### **4.1.2.1 Data Storage**

As seen in the Figure 16, the two main data storage places that are available are SD1 and SD0. In the SPALab configuration, SD1 is primarily used for direct interaction with the astronaut. This interaction includes: uploading new FW to the C&DH, uploading new satellite simulator scripts, retrieving previously recorded SPA-1 network data and permanent storage of experimental data. SD0 is used for temporary data storage which includes: temporary storage of new FW, temporary storage of new satellite simulator scripts, temporary storage of experimental data that are generated during astronaut interaction. The latter case is used if an astronaut has an ELC plugged into the front panel and the SPA-1 payload is still generating data. These data will be stored onto the SD0 card until the astronaut unplugs the ELC from the NanoRack. Following the unplugging action, the information on SD0 is transferred to SD1 for more permanent storage.

#### **4.1.2.2 Bootloader**

Since SPA devices can be reconfigurable on-orbit, the application layer (as shown in Figure 12) also has to be updated. Reprogramming the application layer allows the SSL to repurpose the SPALab to take advantage of the new configuration. To update

the application layer and the data structures associated with a new SPA device, a bootloader is used to reprogram the C&DH on the SPALab. Code can be sent from the users' computer, through the data path shown in Figure 7 and uploaded to the SD1 card on the AmesLab. Once the astronaut finishes the data transfer to the SD1 card, the ELC is disconnected from the NanoRack. When this occurs, the new firmware is transferred to the SD0 card. Upon the next reboot, the firmware version on SD0 is checked and if new, is re-flashed to the C&DH. The bootloader process is detailed further in 0 on the FW updater page.

#### **4.1.2.3 Interaction with astronaut through the NanoRack**

The SPALab requirements are very similar to the requirements for the AmesLab for astronaut interaction. Astronaut interaction with data stored on the AmesLab is covered in section 3.3 and will not be repeated here. However, this software will allow astronauts to copy new firmware and software simulator code to the SPALab. Additionally, the astronaut can retrieve xTEDS and experimental data. The USB interrupt slide of the software flowchart in 0 details the astronaut interactions.

#### **4.1.2.4 Satellite simulator**

If a customer wants to operate their SPA-1 device in microgravity, they have the following options: install it in an existing SPA-1 satellite with other SPA-1 devices or install it in the SPALab which can simulate data from any number of SPA-1 devices. Either way, the SPA-1 device in question will not "know" the difference between which platform it has been installed in.

The satellite simulator mode is a configuration which makes the SPALab appear as a complex SPA-1 network to a SPA-1 device under test. The satellite simulator mode will use two separate components. First, data structures for a dummy SPA device can be programmed into the firmware alongside the data structures for the real SPA devices. Second, data associated with a dummy SPA device can be injected into the

processing structures during the “Network Simulator” block on the Overview slide of 0. The placement of this block allows the processing structure to be filled with dummy data before the “processing information” section and allows data to be placed into the output structure before the next write cycle of the round robin.

### **4.1.3 Code Compliance**

A very large amount of coordination and effort has to be completed to get this product to operate on orbit. The software on the SPALab has to be thoroughly tested as to instill confidence that the project will operate nominally once on orbit. To ensure this high level of confidence, the SSL has developed a coding standard and a peer code review process. The SPALab project was susceptible to those standards and will be described below. This software has been through a rigorous code review process to ensure efficient, reliable and robust embedded software was written.

#### **4.1.3.1 SSL Coding Standard**

The SSL as a whole has a history of writing in-depth software for a wide variety of embedded applications. The structure of the lab is such that students can come and go during the duration of a software project. In order to ensure consistent coding practices throughout the entire project from inception to completion, an SSL coding standard was completed. It is especially important to write code that is easily understood by all lab members to facilitate modification if necessary.

The SSL coding standard resides in an online medium known as a wiki which is easily accessible to all lab members. The coding standard was originally developed for the KYSat-1 project and has slowly evolved to fit the changing needs of the lab. It is a merged standard based on Motor Industry Software Reliability Association (MISRA) C version 2004, The Firmware Development Standard by Jack Ganssle, and a master’s thesis on Fault Tolerant and Flexible CubeSat Software Architecture by

Greg D. Manyak at California Polytechnic State University [23] [24] [25]. The SPALab software was coded to the SSL coding standards and the code was verified during the peer code review process.

#### **4.1.3.2 Peer Code Reviews**

Code reviews are also an important part of the software coding process. Code reviews are a great way to identify bugs, encourage collaboration and keep code more maintainable. Two resources for performing effective peer code reviews were identified by the SSL: “Effective Code Reviews without the Pain” [26] and “11 Best Practices for Peer Code Review” [27]. These procedures were analyzed and followed closely for the code review of the SPALab project. Overall, the peer code reviews were very successful in identifying potential bugs and getting the entire software team more familiar with the SPALab code.

## **4.2 Hardware**

There were very minimal hardware changes that were made to the AmesLab board to adapt it to become a SPALab. An I2C isolator and various connection interfaces were added to the AmesLab to make it easier and safer to connect SPA-based payloads.

### **4.2.1 I2C isolators**

Typically I2C isolation can be difficult due to the bidirectional nature of I2C operation. Most I2C isolation circuitry is unidirectional and as such is not suited for the SPALab project. The Silicon Labs Si8400 bi-directional I2C isolation chip was chosen to provide isolation in this environment.

I2C isolation provides the following benefits: signal integrity is preserved when the signal is carried over a long distance and allows for payloads with different I2C bus voltages to be connected to the SPALab.

#### **4.2.2 Physical connector**

With the addition of I2C isolators, a physical connection was also added to the AmesLab to allow a wider variety of payloads to be connected. The AmesLab relies on a Samtec Incorporated 50-pin header connector, model number EHT-125-01-S-D-RA, to interface to payloads. While the 50-pin header is an interface option, it limits the types of devices that can connect to the AmesLab. To accommodate the usage of the 50-pin connector by SPA devices, the same signal and ground lines were also routed to the 50-pin connector. The 50-pin connector can be seen in Figure 25 below.



**Figure 25 - 50-pin header connection for SPA Devices**

#### **4.3 Operations for installing new SPA devices**

This section describes the operational processes for installing a new SPA-1 device in the SPALab as detailed in Figure 26. This scenario assumes that the SPALab is

already on orbit and installed into the NanoRacks platform. An example of a SPA-1 temperature sensor being developed by a payload developer is used so the reader can follow the processes a bit easier.

#### **4.3.1 Pre-Launch**

The payload developer creates a temperature sensor that is fully compliant with the SPA-1 network protocol. This includes the development of an xTEDS. They contact the SSL and inform them they want to use the SPALab to test their SPA-1 temperature sensor. The SSL works with the payload developer to create an application that will run on the SPALab C&DH in coordination with the SDM-Lite to test all of the functionality of the temperature sensor as well as log the results.

The SSL manually extracts the capabilities of the temperatures sensor by analyzing the xTEDS and placing relevant information into data structures on the SPALab software image. The xTEDS data structures, the xTEDS itself, and the application needed to use the temperature sensor are used to generate a FW update for the SPALab C&DH that is on orbit. The software is sent to NASA to be loaded onto the SD1 card on the SPALab by using the data path shown in Figure 7. The SSL works with NASA to transport the temperature sensor to the ISS to be installed on the NanoRacks. This process is detailed in section 2.2.3.

#### **4.3.2 Post-Launch**

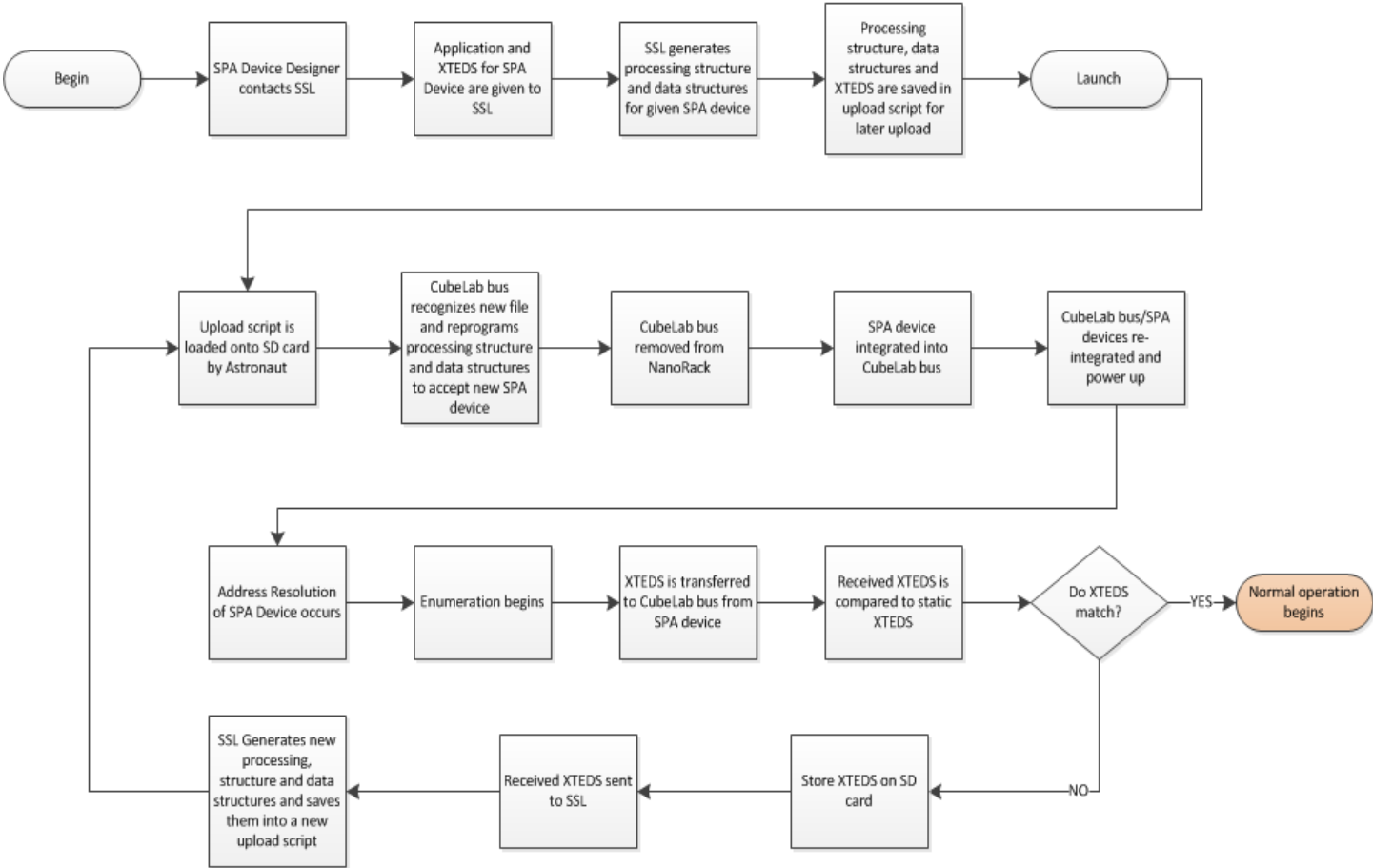
Following launch of the temperature sensor, the SSL works with an astronaut on the ISS to upgrade the SPALab to use the new temperature sensor. This process is started by dragging the FW from the ELC file system onto the SPALab SD card, SD1. When the data transfer is complete, the astronaut performs a hard reboot of the SPALab by disconnecting power and reapplying power. Upon boot up, the SPALab SW checks the appropriate folder and recognizes a new FW version is available. All processes are halted until the FW update has been completed. Upon the completion

of the FW update, the astronaut removes the SPALab from the NanoRack, installs the temperature sensor by connecting it to the “Interface to payload” connector as seen in Figure 19.

When the final configuration has been completed, the SPALab with attached temperature sensor is attached to the NanoRack. Upon boot up and enumeration, the xTEDS for the sensor is transferred to the SPALab. The xTEDS from the sensor is compared to the xTEDS that was loaded onto the C&DH during the FW update process. If the xTEDS do not match, it is assumed that the FW is incorrect or the wrong SPA sensor was plugged into the SPALab. The xTEDS from the temperature sensor is stored on the SD card for later retrieval by an astronaut and analysis by the SSL and the payload developer on the ground. If a FW update is needed, the update process is repeated. If the xTEDS match, normal operation begins. A flowchart showing the entire process described in this section can be seen in Figure 26. Alternatively, the software image can be loaded onto the SPALab and the SPA-1 device is installed during the pre-launch phase. This is the preferred method, as it requires less coordination with NASA for using astronaut time and allows the system to be tested before launch to ensure proper operation.



**Operational flow – Installing new SPA-1 device on SPALab**



55

**Figure 26 - Operations for installing new SPA-1 device into the SPALab**

#### 4.4 8051-based ASIM

During the process of designing and testing the SPALab, the use of an ASIM based on existing architectures would have severely limited the testing and analysis processes of the SPA-1 network. The SSL didn't have the microprocessor development kits required to properly run the existing ASIM code. The AFRL has an online resource for the SPA standard which is managed by SDL at Utah State University [28]. Software for an Atmel-based ASIM was downloaded and analyzed as a starting point for writing custom code for an ASIM.

After much work, the Atmel-based ASIM code was ported to a Silicon Labs C8051F120 microprocessor. The coding effort to build an ASIM that was based on the 8051-architecture was started in early 2012. This porting effort was mainly done by me but others in the lab also helped. The 8051-ASIM efforts allowed a lot of flexibility when testing the SPA-1 network and also verified that the system was worked as expected. The 8051-based ASIM that was developed has the same functionality as the ASIM as it is described in section 2.3.2.

All SPA-1 ASIMs conform to the hardware and protocol requirements contained in the "Designing a SPA-1 ASIM" document that is found on SDL website for SPA [28]. According to the document, a SPA-1 ASIM must respond to three phases: Address Resolution Protocol (ARP), enumeration and the round robin. The address resolution is a method that ASIMs use to obtain their I2C slave address. When an ASIM comes online, it starts at the lowest possible address of 0x11 and tries to send a message to that address. If a Negative Acknowledge (NACK) is received, it assumes that address. If an Acknowledge (ACK) is received, the address is incremented by one and the contact is retried until an available address is found. Enumeration and the round robin have been covered extensively in this thesis and will not be repeated here.

## 5 Results

This chapter details the results and testing procedures for the SDM-lite and the 8051-based ASIM in a laboratory environment. There are many different ways to test a system of such complexity. I used a black box style testing methodology to demonstrate the functionality of the SDM-lite on the SPALab. Figure 27 shows the test setup that was used in the lab environment for testing. The general idea behind the test setup was to have one ASIM (Temp ASIM) report the processor temperature once per second to the SDM-lite. The SDM-lite used that information and through the use of an application, outputted a message to toggle an LED on the second ASIM (LED ASIM) if the temperature went over 40C. The I2C network traffic was observed throughout the entire power up and operation sequence including the enumeration process and all blocks in the top-level flow chart as see in Figure 22. This traffic is annotated for clarity to the reader and include in the appendices.

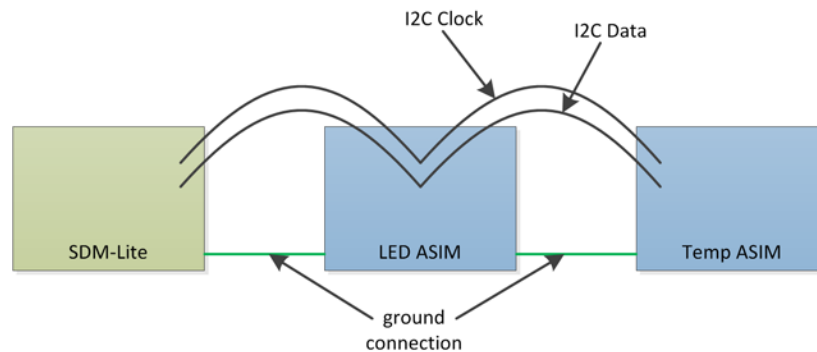


Figure 27 - Black Box test setup for SPALab

### 5.1.1 Network enumeration

The network enumeration phase is the primary process for discovering the capabilities of each ASIM and registering it with the SDM-lite. Appendix B includes

the I2C network traffic that was generated during the ARP phase and network enumeration for each ASIM.

The first 3 lines in Appendix B show the ARP for both ASIMS. The Temp ASIM came online and searched for the lowest available I2C address by probing address 0x11. Since there was no device at 0x11, the packet received a NACK and the Temp ASIM started using 0x11 as its' slave address. Next, the LED ASIM came online and went through the same process. When address 0x11 was probed, the Temp Sensor ASIM responded and the LED ASIM moved on to the next slave address, 0x12. When probed, 0x12 responded with a NACK so the LED ASIM assumed 0x12 as its' slave address.

Next, the enumeration process is shown which retrieved the GUID, status, version and the xTEDS from each ASIM. Packets have been omitted for the sake of brevity. After the enumeration process was completed for each ASIM, all available data messages were subscribed to by sending commands to each ASIM.

### **5.1.2 Reporting temperature**

The main functionality of this thesis is proven by the I2C traffic shown in Appendix C. This traffic was recorded after the ARP and network enumeration process were completed. It represents the I2C bus output generated by looping through software as represented by the top-level software flowchart shown in Figure 22 a total of eleven times.

During the observation, I used a temperature gun to heat the processor temperature on the Temp ASIM. This was done to trigger the operation of the LED at above 40C. The Temp ASIM at slave address 0x11 was read and the processor temperature was reported as: 0x48, 0xE1, 0x07, and 0x42. The bytes read from the Temp ASIM were reported in little endian format and represent a floating-point number. When

converted to decimal, this value represents a temperature of 33.97 C. This value was sent to the application for the Temp ASIM and analyzed. Since the temperature was below 40 C, a message was sent to the LED ASIM at address 0x12 to ensure the LED was off by sending byte 0xAA. This process was repeated until the floating-point value received from Temp ASIM went above 40 C. The third temperature packet from the Temp ASIM that is shown indicates that the temperature rose to 43.85 C. Analysis of this temperature shows the application for the Temp ASIM sent a message to the LED ASIM to turn the LED on by sending the byte 0xFF. Expected operation of the LED on the LED ASIM was observed visually.

### **5.1.3 ASIM re-enumeration**

To test the plug-and-play aspect of the SPALab, the test setup was powered up and allowed to reach a steady state. The Temp ASIM was unplugged and plugged back in and the I2C network traffic was observed to see if the system re-enumerated the Temp ASIM and started reporting the correct command to the LED ASIM. The network traffic from the test can be seen in Appendix D.

The network traffic shows the Temp ASIM (0x11) was initially unresponsive and the LED ASIM (0x12) was alive and reporting 0xFF FF FF. The Temp ASIM was plugged back in around 11:21:08 and the ASIM was re-enumerated. Next, the SDM-lite re-subscribed to the data messages and the Temp ASIM resumed reporting the temperature data.

### **5.1.4 Multiple ASIM re-enumeration**

The SPALab needs to handle the case where both ASIMs power cycle. The test setup was placed in the initial configuration and powered up. The network traffic was analyzed and allowed to reach a steady state. Both ASIMs were power cycled as shown in Appendix E. The ASIMs were powered up in the same order as they were powered on during the initial testing cycle (Temp ASIM first). ARP for the LED ASIM

was observed in the network traffic. Following the ARP of the LED ASIM, network enumeration and subscription to data messages for each ASIM was observed.

#### **5.1.5 ASIMs changing addresses**

In addition to handling multiple ASIMs power cycling and resuming operation with the same I2C slave address, the SDM-lite also needs to be able to handle the case where ASIMs come back online in a different order and switch I2C addresses. Network analysis for this case is shown in Appendix F.

The I2C traffic showing the re-enumeration of each ASIM was analyzed. The startup packets corresponding to the network enumeration were omitted. The network analysis shows the Temp ASIM responded at address 0x11. Analyzing the packets that were received indicates this. The values “0xC3, 0xF5, 0xCC and 0x41” represent a floating-point temperature value in little endian notation. Converted to decimal, the bytes represent a temperature of 25.62 C. Additionally, the LED ASIM was observed at address 0x12 by observing an LED off command that was sent to that address.

The ASIMs were powered off and allowed to resume power in an order that allowed them to assume different slave addresses. Upon power up, the LED ASIM was assigned address 0x11, enumerated during the round robin phase, and any available data messages were subscribed to. A message that was left in the output queue from before the power cycle was sent to the LED ASIM at 0x11 indicating that the SDM-lite registered the new address. Next, the Temp ASIM was powered up and the ARP phase was initiated. Upon completion of the ARP, the Temp ASIM was assigned address 0x12 and the enumeration process was initiated. Finally, a temperature message was read from the Temp ASIM at address 0x12 indicating the SDM-lite registered the new address of the Temp ASIM.

### **5.1.6 ASIM powering off for extended period of time**

There was also a need to test the functionality of the SDM-lite if an ASIM was powered off for an extended period of time. In terms of this system, an extended period of time was chosen as 90 seconds. At an average of 16 messages per second, the SDM-lite sends roughly 1,440 messages during this time. This allowed the round robin count to be reset 14 times causing an enumeration of the bus each time. By analyzing the network traffic in Appendix G, an ASIM is shown to come back online after an extended period of time.

The I2C traffic shows that the Temp ASIM was slave address 0x11 and output temperature data just before it went offline. 90 seconds passed with the ASIM offline. Upon power up, the ASIM was immediately enumerated during the round robin phase and message subscriptions were resumed. Finally, the Temp ASIM output a temperature value after a second passed indicating nominal operation resumed.

### **5.1.7 High Round Robin Count**

There is also a mechanism built into the SDM-lite which causes the system to re-enumerate all ASIMs after a certain amount of round robins have passed. This allows the system to recalibrate itself if an error goes undetected. For the purposes of the SPALab application, the upper round robin count was set at 100. The test results show the re-enumeration process was initiated after 100 round robins have occurred and nominal operation was resumed. Appendix H shows the I2C network traffic for this analysis. The 100<sup>th</sup> round robin was observed with the re-enumeration of each ASIM following. Data subscriptions were not sent in this case as the ASIMs didn't power cycle since the last enumeration and the last subscriptions were still valid. This was verified by observing normal operation after the enumeration process.

## 5.2 ASIM Results

To test the functionality of the 8051-based ASIM that were created, they were tested using two different methods. First, they were used in all of the tests scenarios listed in Section 5. This shows that the 8051-based ASIMs are working correctly for the SDM-lite application so those results will not be repeated here. Second, the Temp ASIM was tested with the SDM to ensure proper operation.

Testing the 8051-based ASIM with an SDM was achieved by using the SPA training kit called CubeFlow by COSMIAC [29]. The CubeFlow kit uses a Linux-based Gumstix controller to provide a full SDM for SPA-U and SPA-1 networks. The test setup that was used can be seen in Figure 28. Through use of the CubeFlow kit and the online resources at SDL mentioned in section 4.4, a test application was developed that tested the full functionality of the 8051-based ASIM. An xTEDS was created using the online tools and loaded onto the Temp ASIM.

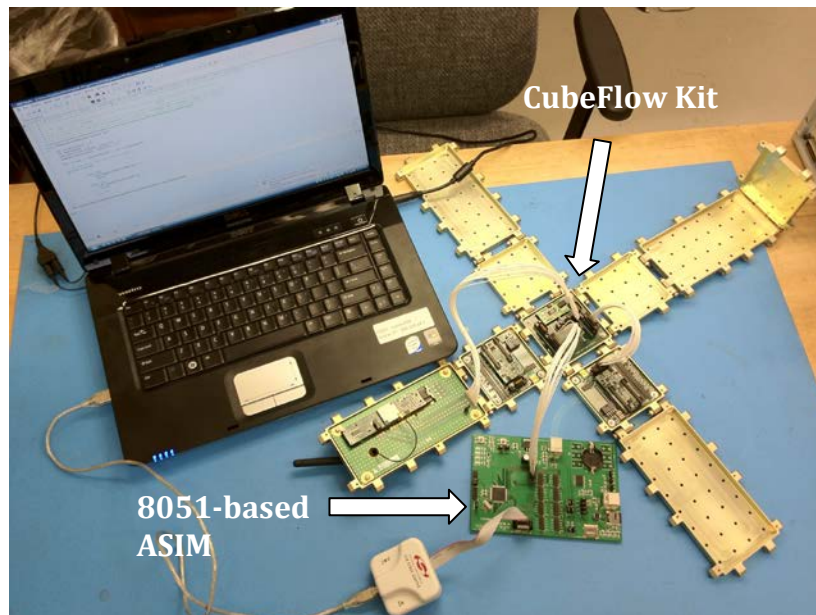
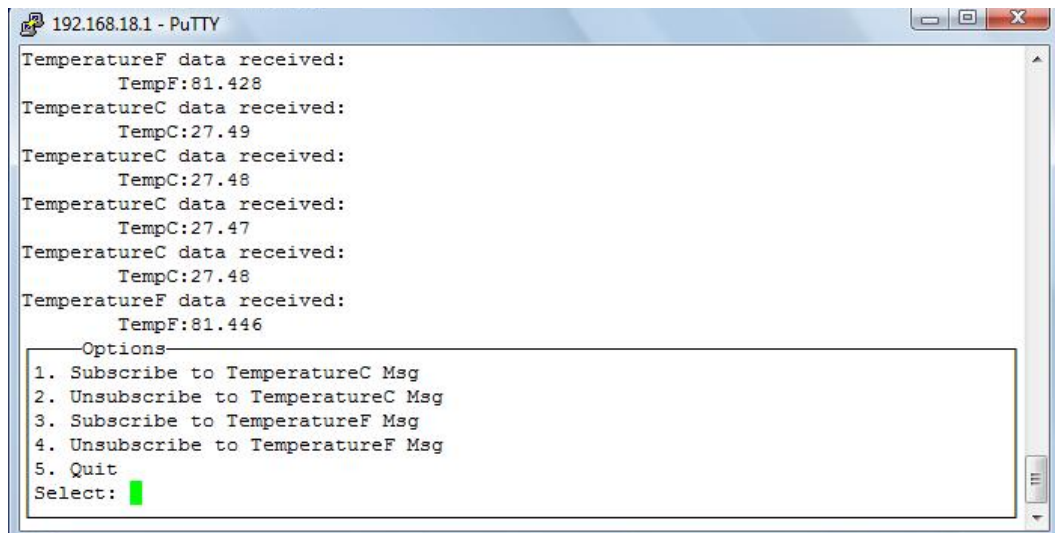


Figure 28 - CubeFlow kit from COSMIAC with AmesLab attached



During the enumeration process, the xTEDS was sent to the CubeFlow Kit using the SPA-1 protocol and data structures were created on the SDM that represented the capabilities of the Temp ASIM. An application to test the capabilities of the temperature sensor was created using the online tools and used on the Gumstix. The application was used to subscribe to available temperature data on the Temp ASIM which reported the processor temperature in Fahrenheit and Celsius. The output of the Temp ASIM that was received by the SDM was passed onto the application running on the Gumstix. A floating-point representation of this data output is shown in Figure 29.



```
192.168.18.1 - PuTTY
TemperatureF data received:
  TempF:81.428
TemperatureC data received:
  TempC:27.49
TemperatureC data received:
  TempC:27.48
TemperatureC data received:
  TempC:27.47
TemperatureC data received:
  TempC:27.48
TemperatureF data received:
  TempF:81.446
Options
1. Subscribe to TemperatureC Msg
2. Unsubscribe to TemperatureC Msg
3. Subscribe to TemperatureF Msg
4. Unsubscribe to TemperatureF Msg
5. Quit
Select: █
```

**Figure 29 - SPA Application showing Temperature data from 8051-based ASIM**

## 6 Conclusion

This chapter provides a summary of the contributions that this thesis has made in addition to providing a prospect for future research. This thesis set out to analyze several existing standards in the small satellite community and merge them together by developing a common platform. This work involved research into the current state of the following platforms: CubeSat standard, NanoRacks, CubeLabs, SPA, and the AmesLab by the SSL. It was recognized that these platforms could be merged together to provide a testing platform on the ISS that will facilitate microgravity operation of SPA devices. The beginning stages of the development of this thesis was detailed in paper published at the 2012 Infotech@Aerospace conference entitled “A SPA-1 Enabled Plug-and-Play CubeLab for ISS Payloads”.

This thesis detailed the development of a software component called SDM-lite that facilitates the operation of SPA-1 devices on low-resource applications, specifically the SPALab on the ISS. This functionality was provided through a collaborative effort with COSMAIC at the University of New Mexico and with the help of other students in the SSL. Specifically, embedded C code was developed which runs on the SPALab and provides the SDM-lite capabilities as defined in Section 4.1.1. To show this functionality, an SDM-lite capable of operating 2 ASIMs which allow passing SPA-1 messages from one ASIM to another was demonstrated. The software blocks that were developed can be seen in Figure 21. The network traffic for the SDM-lite setup was analyzed and detailed in Chapter 5 and the appendices. An overall software flow chart for the SPALab showing the integration of SDM-lite into the AmesLab software architecture was developed and can be seen in 0.

Additionally, software for an 8051-based ASIM was developed and tested. This further expands the ecosystem of architectures that can be used as an ASIM for SPA-1 devices.

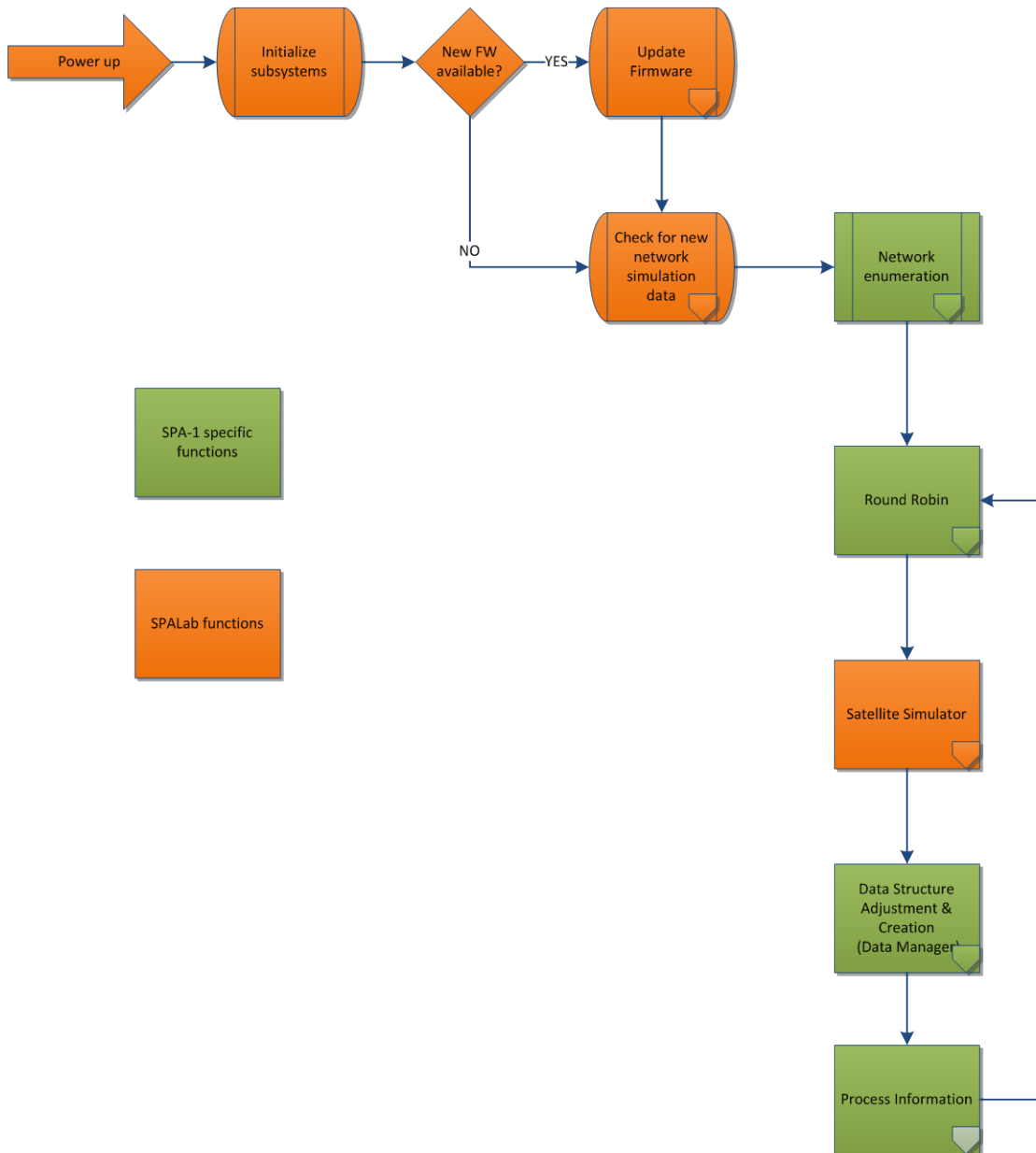
The SPALab has taken a significant step towards getting to orbit through the many advances made by this thesis, but there are still tasks that must be completed before the final step can be accomplished. Currently the data structures on the SDM-lite have to be programmed with the capabilities of each ASIM. Further collaboration with COSMIAC will enable parsing of XTEDS and population of the data structures during the enumeration process. Next, the AmesLab and SDM-lite code exist in the same project file in  $\mu$ Vision but are yet to be fully integrated together. Currently the API for all of the necessary AmesLab software components exists but they lack full integration into the SDM-lite code. The integration will be complete when the entire flow chart shown in 0 is fully coded.

Beginning in the summer of 2012, Kentucky Space (which includes SSL at the University of KY and Morehead State University) started building a CubeSat called KYSat-2. This CubeSat utilizes the SiLabs C8051F120 as the main processor for the C&DH and several SiLabs C8051F912s for an interface between the C&DH and devices on the network. This architecture allows the C8051F120 to provide some of the same functionality as the SDM-lite and the C8051F912s will provide functionality similar to the 8051-based ASIM as described in this thesis. Many of the concepts discussed in this thesis were applied to the architecture for KYSat-2.

# Appendices

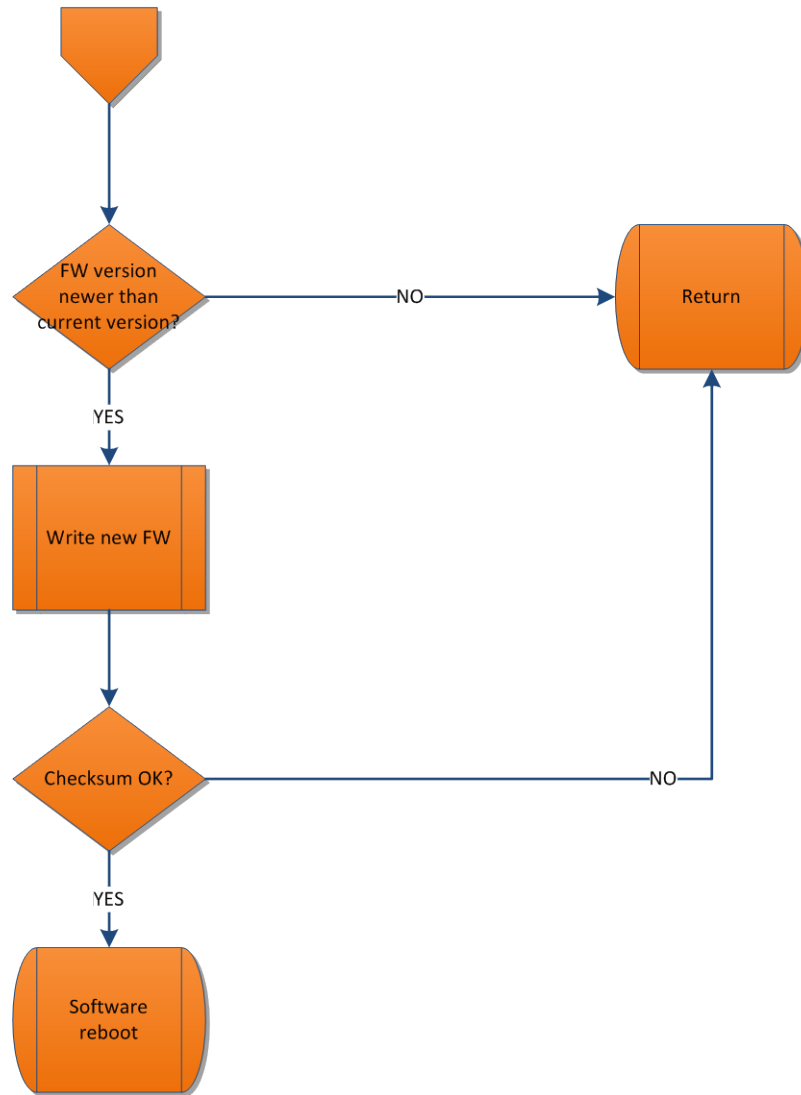
## Appendix A Software flowchart for SPALab

### SPALab - Overview



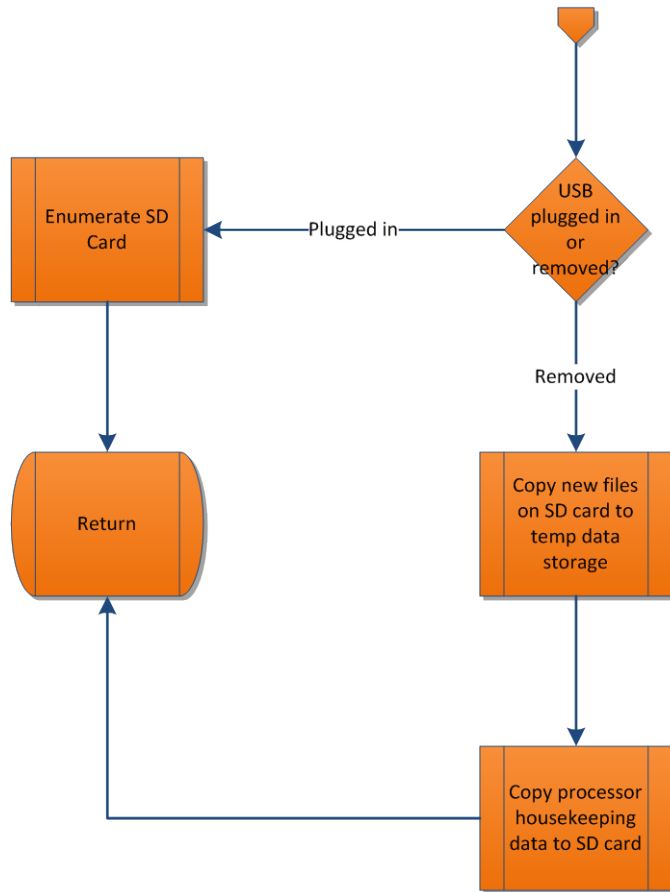
## FW updater (SPA-FU)

If different SPA devices are plugged into the SPALab while on orbit, the SPALab will need to be "reprogrammed" to interact with the new configuration. This process is used to retrieve a new FW update file from an astronaut and reprogram the C&DH.



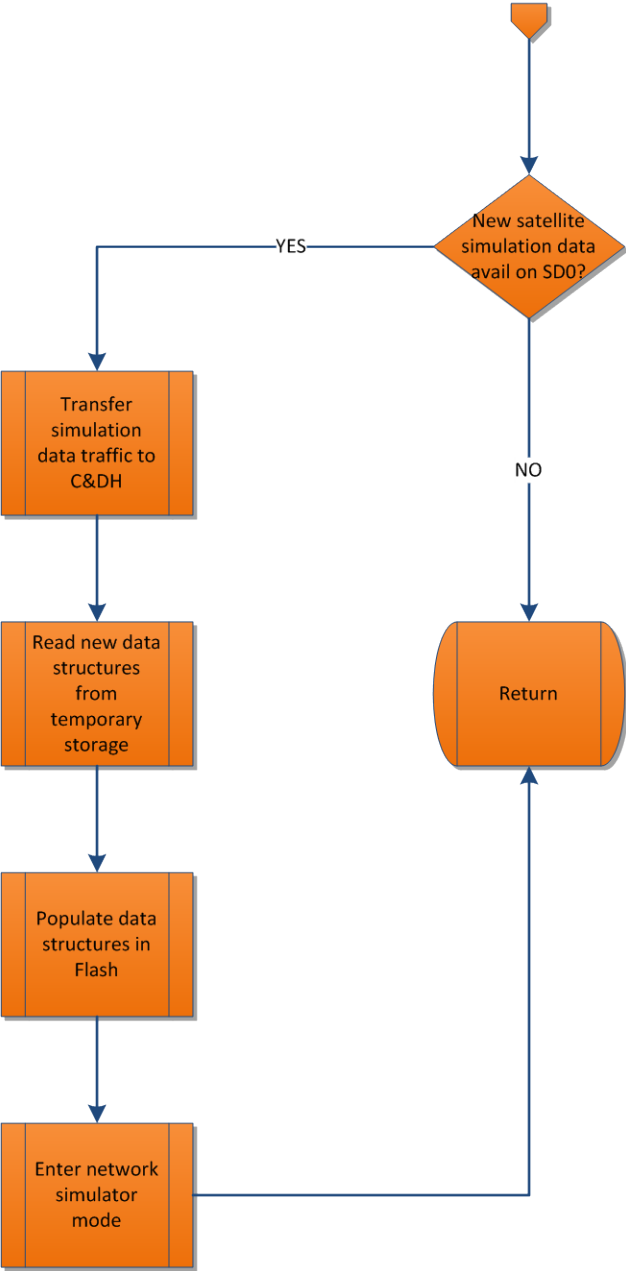
# USB interrupt

This interrupt is serviced if there is a change in the USBSR register on the MAX14502. A change in the status occurs if the USB connection changes between suspend or resume modes. This is a good way to test if a USB device has been plugged in suspend -> resume or if a USB device is inactive resume -> suspend. This assumes that the MAX14502 is in Card Reader mode by default.

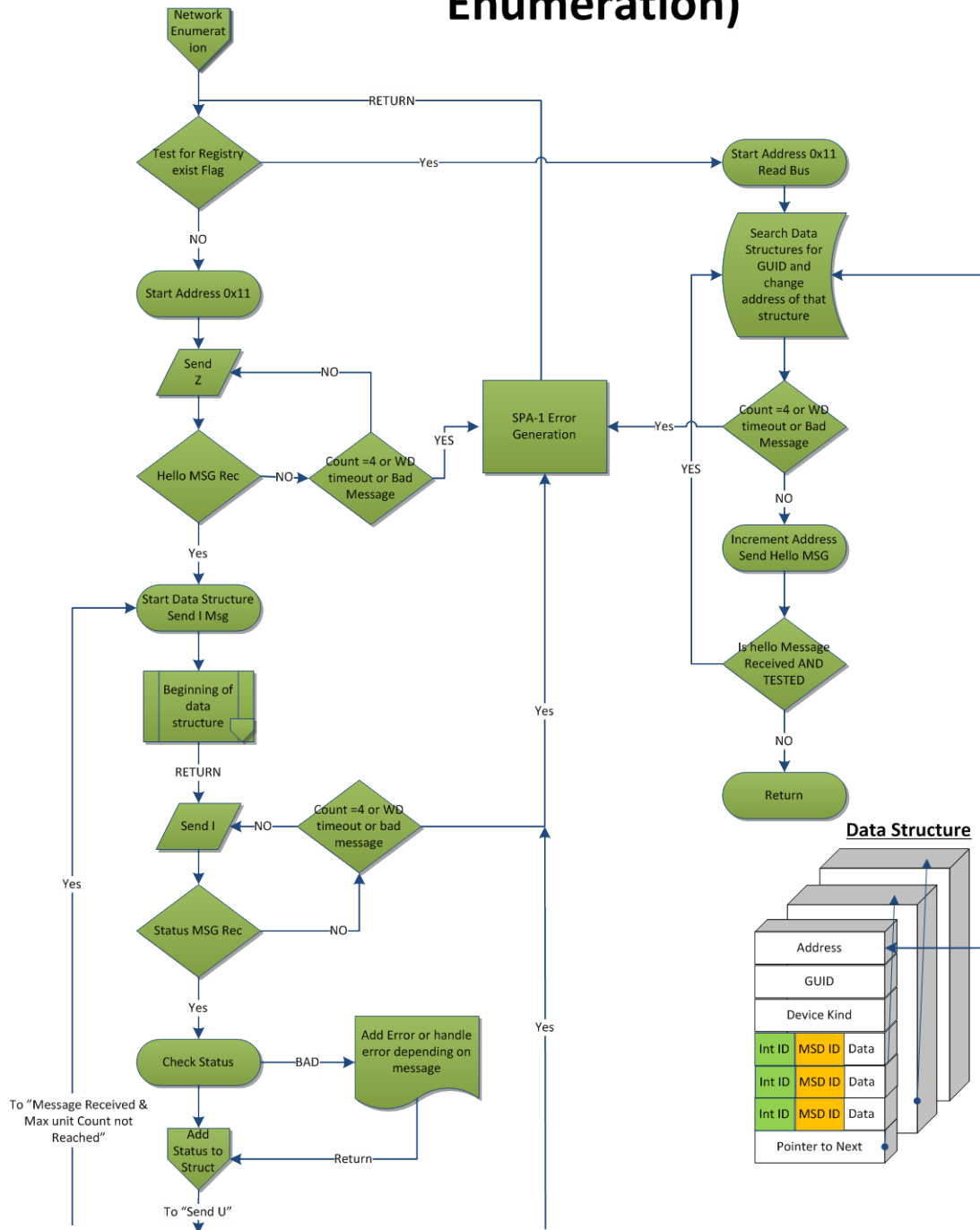


# Load new config script

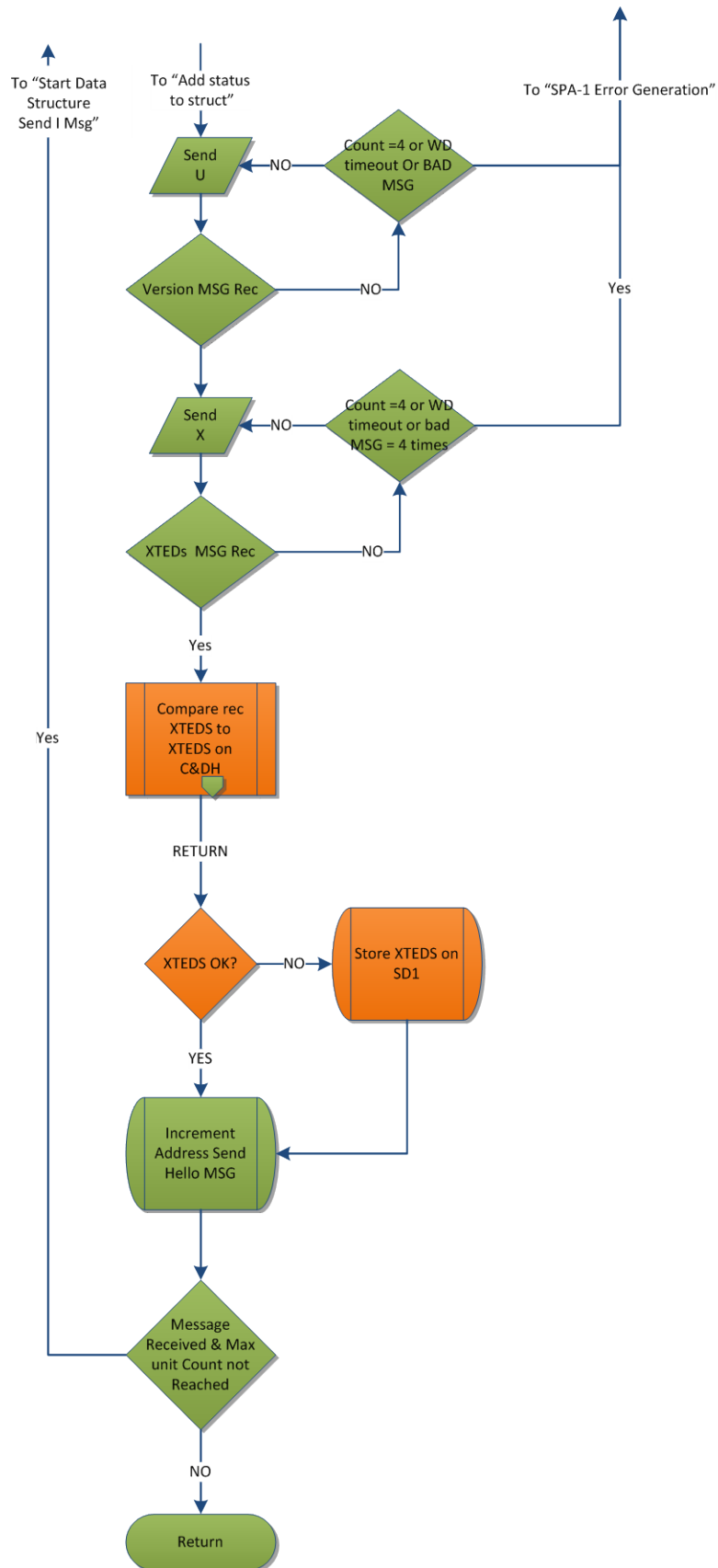
This process will read the processes and data structures necessary to operate new devices on a SPA-1 network. This is useful if the SPA network is reconfigured on orbit. The SPALab will need to be “reprogrammed” to interact with the new configuration



# MiniPnP Data Handler (Network Enumeration)



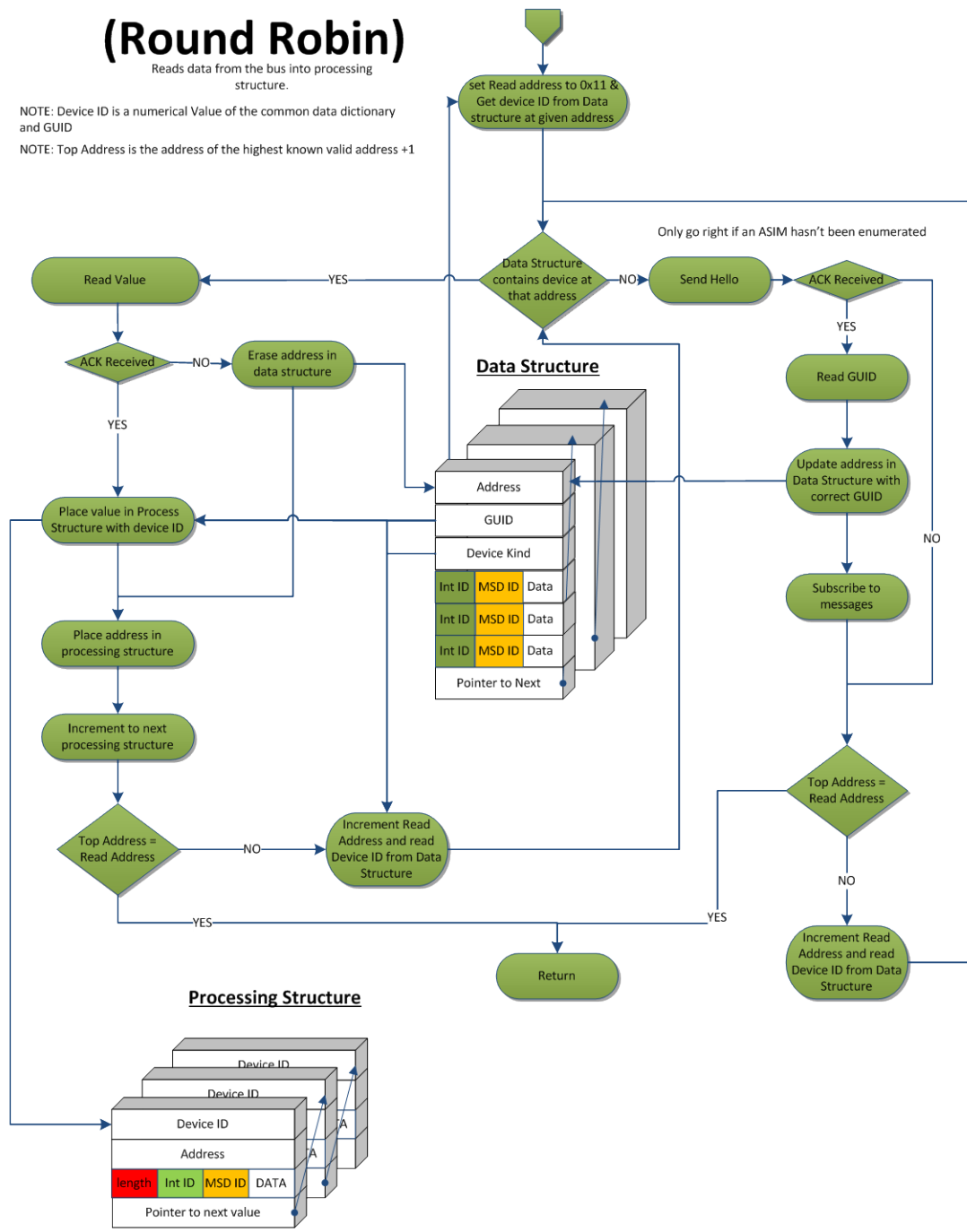




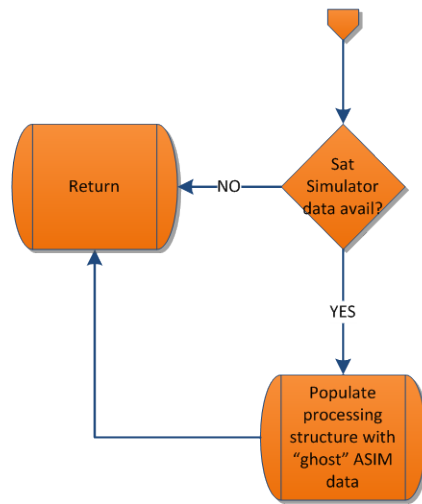
# MiniPnP Data Handler (Round Robin)

Reads data from the bus into processing structure.

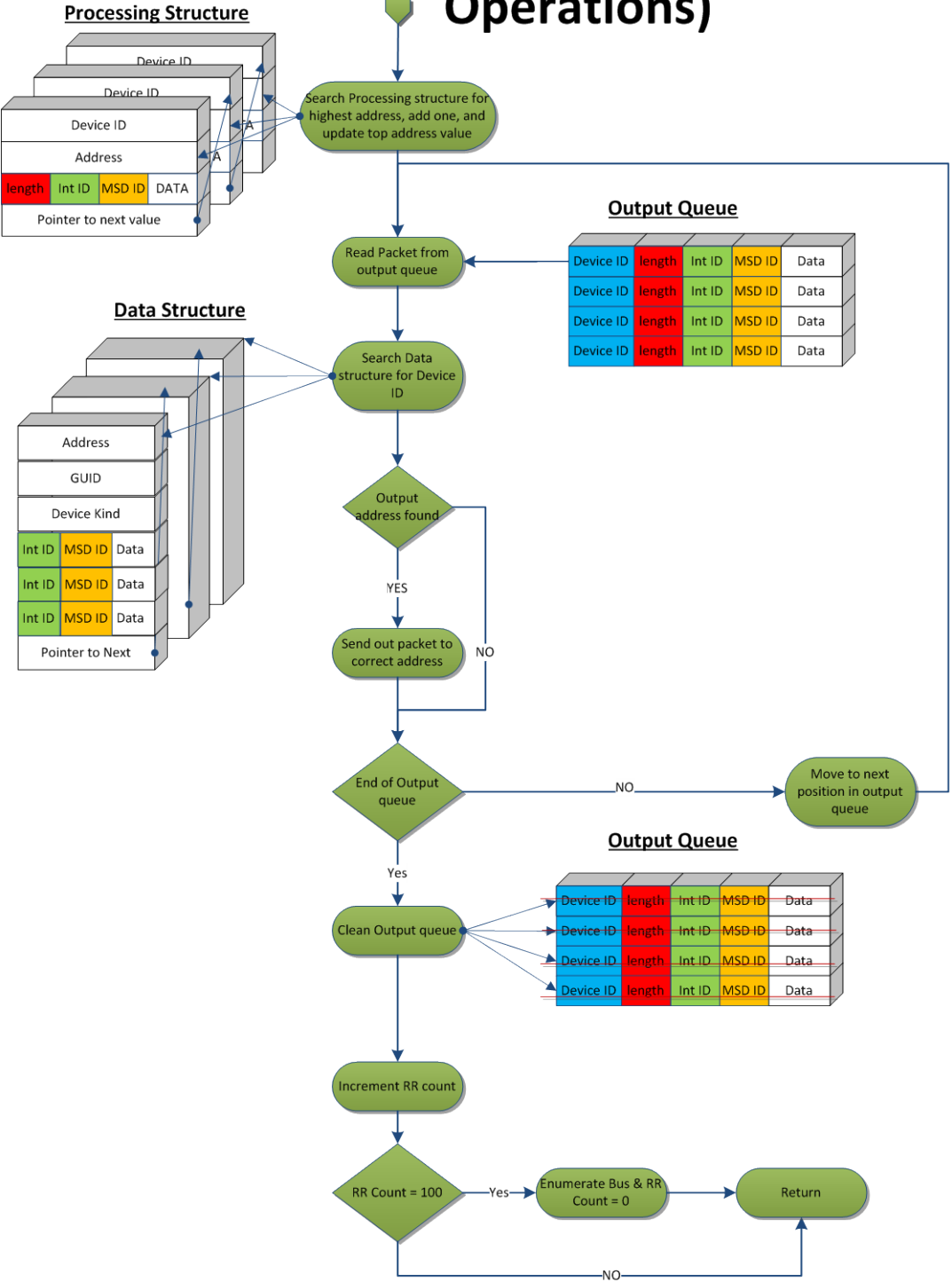
NOTE: Device ID is a numerical Value of the common data dictionary and GUID  
NOTE: Top Address is the address of the highest known valid address +1



# Satellite Simulator



# MiniPnP Data Handler (Data Registry Operations)





## Appendix B SPA-1 Network traffic for network enumeration

```
(2012-12-11 13:33:07) [S] <11:w>* [P] ← ARP for first ASIM
(2012-12-11 13:33:09) [S] <11:w> 57 00 04 8b d0 38 35 [P]
(2012-12-11 13:33:09) [S] <12:w>* [P] ← ARP for second ASIM
```

---

### ASIM 1 Enumeration

---

```
(2012-12-11 13:33:11) [S] <11:w> 5a 00 00 [P] ← "Enumerate" message
(2012-12-11 13:33:11) [S] <11:r> 48 04 00* [P] ← "Hello" header
(2012-12-11 13:33:11) [S] <11:r> d2 02 96 49* [P] ← "Hello" payload
(2012-12-11 13:33:11) [S] <11:w> 49 00 00 [P] ← "Initialize" message
(2012-12-11 13:33:12) [S] <11:r> 53 01 00* [P] ← "Status" header
(2012-12-11 13:33:12) [S] <11:r> 10* [P] ← "Status" payload
(2012-12-11 13:33:12) [S] <11:w> 55 00 00 [P] ← "Request Version" message
(2012-12-11 13:33:12) [S] <11:r> 4b 01 00* [P] ← "Version" header
(2012-12-11 13:33:12) [S] <11:r> 00* [P] ← "Version" payload
(2012-12-11 13:33:12) [S] <11:w> 58 00 00 [P] ← "Request XTEDS" message
(2012-12-11 13:33:12) [S] <11:r> 4a 13 04* [P] ← "XTEDS" header
(2012-12-11 13:33:12) [S] <11:r> 3c 3f 78 6d 6c 20 76 65 72 73 69 6f 6e
3d ← Begin XTEDS
(2012-12-11 13:33:12) 22 31 2e 30 22 20 65 6e 63 6f 64 69 6e 67 3d 22
(2012-12-11 13:33:12) 75 74 66 2d 38 22 20 3f 3e 0a 3c 78 54 45 44 53
(2012-12-11 13:33:12) 20 78 6d 6c 6e 73 3d 22 68 74 74 70 3a 2f 2f 77
(2012-12-11 13:33:12) 77 77 2e 69 6e 74 65 72 66 61 63 65 63 6f 6e 74
```

<Packets omitted>

```
(2012-12-11 13:33:13) [S] <11:r> 4a 13 04* [P]
(2012-12-11 13:33:13) [S] <11:r> 69 63 61 74 69 6f 6e 3e 0a 09 3c 2f 49
6e
(2012-12-11 13:33:13) 74 65 72 66 61 63 65 3e 0a 3c 2f 78 54 45 44 53
(2012-12-11 13:33:13) 3e* [P] ← End XTEDS
```

---

### ASIM 2 enumeration

---

```
(2012-12-11 13:33:13) [S] <12:w> 5a 00 00 [P] ← "Enumerate" message
(2012-12-11 13:33:13) [S] <12:r> 48 04 00* [P] ← "Hello" header
(2012-12-11 13:33:13) [S] <12:r> 35 38 d0 8b* [P] ← "Hello" payload
(2012-12-11 13:33:13) [S] <12:w> 49 00 00 [P] ← "Initialize" message
(2012-12-11 13:33:13) [S] <12:r> 53 01 00* [P] ← "Status" header
(2012-12-11 13:33:13) [S] <12:r> 10* [P] ← "Status" message
(2012-12-11 13:33:13) [S] <12:w> 55 00 00 [P] ← "Request Version" message
(2012-12-11 13:33:13) [S] <12:r> 4b 01 00* [P] ← "Version" header
(2012-12-11 13:33:13) [S] <12:r> 00* [P] ← "Version" payload
(2012-12-11 13:33:13) [S] <12:w> 58 00 00 [P] ← "Request XTEDS" message
(2012-12-11 13:33:13) [S] <12:r> 4a 7f 02* [P] ← "XTEDS" header
(2012-12-11 13:33:13) [S] <12:r> 3c 3f 78 6d 6c 20 76 65 72 73 69 6f 6e
3d
(2012-12-11 13:33:13) 22 31 2e 30 22 20 65 6e 63 6f 64 69 6e 67 3d 22
(2012-12-11 13:33:13) 75 74 66 2d 38 22 20 3f 3e 0a 3c 78 54 45 44 53
(2012-12-11 13:33:13) 20 78 6d 6c 6e 73 3d 22 68 74 74 70 3a 2f 2f 77
(2012-12-11 13:33:13) 77 77 2e 69 6e 74 65 72 66 61 63 65 63 6f 6e 74
(2012-12-11 13:33:13) 72 6f 6c 2e 63 6f 6d 2f 53 50 41 2f 78 54 45 44
(2012-12-11 13:33:13) 53 22 20 78 6d 6c 6e 73 3a 78 73 69 3d 22 68 74
```

<Packets omitted>

```
(2012-12-11 13:33:13) [S] <12:r> 4a 7f 02* [P]
(2012-12-11 13:33:14) [S] <12:r> 6d 6d 61 6e 64 3e 0a 09 09 3c 43 6f 6d
6d
(2012-12-11 13:33:14) 61 6e 64 3e 0a 09 09 09 3c 43 6f 6d 6d 61 6e 64
(2012-12-11 13:33:14) 4d 73 67 20 6e 61 6d 65 3d 22 4c 45 44 5f 4f 46
(2012-12-11 13:33:14) 46 22 20 64 65 73 63 72 69 70 74 69 6f 6e 3d 22
(2012-12-11 13:33:14) 55 73 65 64 20 74 6f 20 74 75 72 6e 20 74 68 65
(2012-12-11 13:33:14) 20 4c 45 44 20 4f 46 46 22 20 69 64 3d 22 32 22
(2012-12-11 13:33:14) 20 2f 3e 0a 09 09 3c 2f 43 6f 6d 6d 61 6e 64 3e
(2012-12-11 13:33:14) 0a 09 3c 2f 49 6e 74 65 72 66 61 63 65 3e 0a 3c
(2012-12-11 13:33:14) 2f 78 54 45 44 53 3e* [P] ← End XTEDS
(2012-12-11 13:33:14) [S] <13:w>* [P]
(2012-12-11 13:33:14) [S] <11:w> 4d 02 00 01 01 [P]
(2012-12-11 13:33:14) [S] <11:w> 4d 02 00 01 02 [P]
(2012-12-11 13:33:14) [S] <12:w> 4d 02 00 01 01 [P]
(2012-12-11 13:33:14) [S] <12:w> 4d 02 00 01 02 [P]
```



---

End Network Enumeration

---

## Appendix C SPA-1 Network traffic for temperature controlled LED

Begin Round

---

```
(2012-12-11 15:45:14) [S] <11:r> 44 06 00* [P] ← Temp Data 33.97C
(2012-12-11 15:45:14) [S] <11:r> 01 01 48 e1 07 42* [P]
(2012-12-11 15:45:14) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:14) [S] <13:w>* [P]
(2012-12-11 15:45:15) [S] <13:w>* [P]
(2012-12-11 15:45:15) [S] <11:r> ff ff ff* [P]
(2012-12-11 15:45:15) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:15) [S] <13:w>* [P]
(2012-12-11 15:45:15) [S] <13:w>* [P]
(2012-12-11 15:45:15) [S] <12:w> 56 03 00 01 01 aa [P] ← LED OFF
(2012-12-11 15:45:15) [S] <11:r> ff ff ff* [P]
(2012-12-11 15:45:15) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:15) [S] <13:w>* [P]
(2012-12-11 15:45:15) [S] <13:w>* [P]
(2012-12-11 15:45:15) [S] <11:r> ff ff ff* [P]
(2012-12-11 15:45:15) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:15) [S] <13:w>* [P]
(2012-12-11 15:45:15) [S] <13:w>* [P]
(2012-12-11 15:45:15) [S] <11:r> 44 06 00* [P] ← Temp Data 39.14C
(2012-12-11 15:45:15) [S] <11:r> 01 01 5c 8f 1c 42* [P]
(2012-12-11 15:45:15) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:15) [S] <13:w>* [P]
(2012-12-11 15:45:16) [S] <13:w>* [P]
(2012-12-11 15:45:16) [S] <11:r> ff ff ff* [P]
(2012-12-11 15:45:16) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:16) [S] <13:w>* [P]
(2012-12-11 15:45:16) [S] <13:w>* [P]
(2012-12-11 15:45:16) [S] <12:w> 56 03 00 01 01 aa [P] ← LED OFF
(2012-12-11 15:45:16) [S] <11:r> ff ff ff* [P]
(2012-12-11 15:45:16) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:16) [S] <13:w>* [P]
(2012-12-11 15:45:16) [S] <13:w>* [P]
(2012-12-11 15:45:16) [S] <11:r> ff ff ff* [P]
(2012-12-11 15:45:16) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:16) [S] <13:w>* [P]
(2012-12-11 15:45:16) [S] <13:w>* [P]
(2012-12-11 15:45:16) [S] <11:r> 44 06 00* [P] ← Temp Data 43.85C
(2012-12-11 15:45:16) [S] <11:r> 01 01 66 66 2f 42* [P]
(2012-12-11 15:45:16) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:16) [S] <13:w>* [P]
(2012-12-11 15:45:17) [S] <13:w>* [P]
(2012-12-11 15:45:17) [S] <11:r> ff ff ff* [P]
(2012-12-11 15:45:17) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:17) [S] <13:w>* [P]
(2012-12-11 15:45:17) [S] <13:w>* [P]
(2012-12-11 15:45:17) [S] <12:w> 56 03 00 01 01 ff [P] ← LED ON
(2012-12-11 15:45:17) [S] <11:r> ff ff ff* [P]
(2012-12-11 15:45:17) [S] <12:r> ff ff ff* [P]
(2012-12-11 15:45:17) [S] <13:w>* [P]
(2012-12-11 15:45:17) [S] <13:w>* [P]
```



## Appendix D SPA-1 Network traffic for ASIM Re-Enumeration

<startup packets omitted>

```
(2012-12-12 11:21:07) [S] <11:w>* [P] ← Temp ASIM unresponsive
(2012-12-12 11:21:07) [S] <11:w>* [P]
(2012-12-12 11:21:07) [S] <12:r> ff ff ff* [P]
(2012-12-12 11:21:07) [S] <13:w>* [P]
(2012-12-12 11:21:07) [S] <13:w>* [P]
(2012-12-12 11:21:07) [S] <11:w>* [P]
(2012-12-12 11:21:07) [S] <11:w>* [P]
(2012-12-12 11:21:07) [S] <12:r> ff ff ff* [P]
(2012-12-12 11:21:07) [S] <13:w>* [P]
(2012-12-12 11:21:08) [S] <11:w>* [P]
(2012-12-12 11:21:08) [S] <13:w>* [P] ← Temp ASIM back online, re-enumerate
(2012-12-12 11:21:08) [S] <11:w> 5a 00 00 [P]
(2012-12-12 11:21:08) [S] <11:r> 48 04 00* [P]
(2012-12-12 11:21:08) [S] <11:r> d2 02 96 49* [P]
(2012-12-12 11:21:08) [S] <11:w> 4d 02 00 01 01 [P] ← resubscribe to
(2012-12-12 11:21:08) [S] <11:w> 4d 02 00 01 02 [P] ← messages
(2012-12-12 11:21:08) [S] <12:r> ff ff ff* [P]
(2012-12-12 11:21:08) [S] <13:w>* [P]
(2012-12-12 11:21:08) [S] <13:w>* [P]
(2012-12-12 11:21:08) [S] <11:r> ff ff ff* [P]
(2012-12-12 11:21:08) [S] <12:r> ff ff ff* [P]
(2012-12-12 11:21:08) [S] <13:w>* [P]
(2012-12-12 11:21:08) [S] <13:w>* [P]
(2012-12-12 11:21:08) [S] <11:r> ff ff ff* [P]
(2012-12-12 11:21:08) [S] <12:r> ff ff ff* [P]
(2012-12-12 11:21:08) [S] <13:w>* [P]
(2012-12-12 11:21:09) [S] <13:w>* [P]
(2012-12-12 11:21:09) [S] <11:r> 44 06 00* [P] ← Temp Data 23.86
(2012-12-12 11:21:09) [S] <11:r> 01 01 48 e1 be 41* [P]
(2012-12-12 11:21:09) [S] <12:r> ff ff ff* [P]
(2012-12-12 11:21:09) [S] <13:w>* [P]
(2012-12-12 11:21:09) [S] <13:w>* [P]
(2012-12-12 11:21:09) [S] <11:r> ff ff ff* [P]
(2012-12-12 11:21:09) [S] <12:r> ff ff ff* [P]
(2012-12-12 11:21:09) [S] <13:w>* [P]
(2012-12-12 11:21:09) [S] <13:w>* [P]
(2012-12-12 11:21:09) [S] <12:w> 56 03 00 01 01 aa [P] ← LED OFF
```

## Appendix E SPA-1 Network traffic for multiple ASIM Re-Enumeration

<startup packets omitted>

```
(2012-12-12 12:38:18) [S] <11:w>* [P] ← TEMP ASIM and LED ASIM unresponsive
(2012-12-12 12:38:18) [S] <11:w>* [P]
(2012-12-12 12:38:18) [S] <11:w>* [P]
(2012-12-12 12:38:18) [S] <12:w>* [P]
```

<ASIMS RESUME POWER>

```
(2012-12-12 12:38:19) [S] <12:w>* [P]
(2012-12-12 12:38:19) [S] <13:w>* [P]
(2012-12-12 12:38:19) [S] <11:w> 57 00 04 8b d0 38 35 [P] ← ARP for LED ASIM
(2012-12-12 12:38:19) [S] <12:w>* [P]
(2012-12-12 12:38:19) [S] <13:w>* [P]
(2012-12-12 12:38:19) [S] <7f:w>* [P]
(2012-12-12 12:38:19) [S] <11:w> 5a 00 00 [P] ← re-enumeration for TEMP ASIM
(2012-12-12 12:38:19) [S] <11:r> 48 04 00* [P] ← re-enumeration for TEMP ASIM
(2012-12-12 12:38:19) [S] <11:r> d2 02 96 49* [P]
(2012-12-12 12:38:19) [S] <11:w> 4d 02 00 01 01 [P] ← subscribe to messages
(2012-12-12 12:38:19) [S] <11:w> 4d 02 00 01 02 [P] ← subscribe to messages
(2012-12-12 12:38:19) [S] <12:w> 5a 00 00 [P] ← re-enumeration for LED ASIM
(2012-12-12 12:38:19) [S] <12:r> 48 04 00* [P] ← re-enumeration for LED ASIM
(2012-12-12 12:38:19) [S] <12:r> 35 38 d0 8b* [P]
(2012-12-12 12:38:19) [S] <12:w> 4d 02 00 01 01 [P] ← subscribe to messages
(2012-12-12 12:38:19) [S] <12:w> 4d 02 00 01 02 [P] ← subscribe to messages
(2012-12-12 12:38:20) [S] <13:w>* [P]
(2012-12-12 12:38:20) [S] <13:w>* [P]
(2012-12-12 12:38:20) [S] <12:w> 56 03 00 01 01 aa [P] ← resume normal operation
(2012-12-12 12:38:20) [S] <11:r> 44 06 00* [P]
(2012-12-12 12:38:20) [S] <11:r> 01 01 14 ae c1 41* [P]
(2012-12-12 12:38:20) [S] <12:r> ff ff ff* [P]
(2012-12-12 12:38:20) [S] <13:w>* [P]
(2012-12-12 12:38:20) [S] <13:w>* [P]
(2012-12-12 12:38:20) [S] <11:r> ff ff ff* [P]
(2012-12-12 12:38:20) [S] <12:r> ff ff ff* [P]
(2012-12-12 12:38:20) [S] <13:w>* [P]
(2012-12-12 12:38:20) [S] <13:w>* [P]
(2012-12-12 12:38:20) [S] <12:w> 56 03 00 01 01 aa [P]
(2012-12-12 12:38:20) [S] <11:r> 44 06 00* [P]
```

## Appendix F SPA-1 Network traffic for ASIMs switching addresses

**<startup packets omitted>**

(2012-12-12 12:55:10) [S] <11:r> 44 06 00\* [P] ← TEMP ASIM at 0x11

(2012-12-12 12:55:10) [S] <11:r> 01 01 c3 f5 cc 41\* [P]

(2012-12-12 12:55:10) [S] <12:r> ff ff ff\* [P]

(2012-12-12 12:55:10) [S] <13:w>\* [P]

(2012-12-12 12:55:10) [S] <13:w>\* [P]

(2012-12-12 12:55:10) [S] <11:r> ff ff ff\* [P]

(2012-12-12 12:55:10) [S] <12:r> ff ff ff\* [P]

(2012-12-12 12:55:10) [S] <13:w>\* [P]

(2012-12-12 12:55:10) [S] <13:w>\* [P]

(2012-12-12 12:55:11) [S] <12:w> 56 03 00 01 01 aa [P] ← LED ASIM at 0x12

**<Power Cycle ASIMs>**

**<packets omitted>**

(2012-12-12 12:55:13) [S] <11:w> 5a 00 00 [P] ← re-enumeration for LED ASIM

(2012-12-12 12:55:13) [S] <11:r> 48 04 00\* [P]

(2012-12-12 12:55:13) [S] <11:r> 35 38 d0 8b\* [P]

(2012-12-12 12:55:13) [S] <11:w> 4d 02 00 01 01 [P] ← subscribe to messages

(2012-12-12 12:55:13) [S] <11:w> 4d 02 00 01 02 [P]

(2012-12-12 12:55:13) [S] <12:w>\* [P]

(2012-12-12 12:55:13) [S] <12:w>\* [P]

(2012-12-12 12:55:13) [S] <13:w>\* [P]

(2012-12-12 12:55:13) [S] <13:w>\* [P]

(2012-12-12 12:55:13) [S] <11:w> 57 00 04 49 96 02 d2 [P]

(2012-12-12 12:55:13) [S] <12:w>\* [P]

(2012-12-12 12:55:14) [S] <11:w> 56 03 00 01 01 aa [P] ← LED ASIM at 0x11

(2012-12-12 12:55:14) [S] <11:r> ff ff ff\* [P]

(2012-12-12 12:55:14) [S] <12:w> 5a 00 00 [P] ← re-enumeration for TEMP ASIM

(2012-12-12 12:55:14) [S] <12:r> 48 04 00\* [P]

(2012-12-12 12:55:14) [S] <12:r> d2 02 96 49\* [P]

(2012-12-12 12:55:14) [S] <12:w> 4d 02 00 01 01 [P] ← subscribe to messages

(2012-12-12 12:55:14) [S] <12:w> 4d 02 00 01 02 [P]

(2012-12-12 12:55:14) [S] <13:w>\* [P]

(2012-12-12 12:55:14) [S] <13:w>\* [P]

(2012-12-12 12:55:14) [S] <11:r> ff ff ff\* [P]

(2012-12-12 12:55:14) [S] <12:r> ff ff ff\* [P]

(2012-12-12 12:55:14) [S] <13:w>\* [P]

(2012-12-12 12:55:14) [S] <13:w>\* [P]

(2012-12-12 12:55:14) [S] <11:r> ff ff ff\* [P]

(2012-12-12 12:55:14) [S] <12:r> ff ff ff\* [P]

(2012-12-12 12:55:14) [S] <13:w>\* [P]

(2012-12-12 12:55:14) [S] <13:w>\* [P]

(2012-12-12 12:55:15) [S] <11:r> ff ff ff\* [P]

(2012-12-12 12:55:15) [S] <12:r> 44 06 00\* [P] ← TEMP ASIM at 0x12

(2012-12-12 12:55:15) [S] <12:r> 01 01 ae 47 c3 41\* [P]

(2012-12-12 12:55:15) [S] <13:w>\* [P]

(2012-12-12 12:55:15) [S] <13:w>\* [P]

(2012-12-12 12:55:15) [S] <11:r> ff ff ff\* [P]

(2012-12-12 12:55:15) [S] <12:r> ff ff ff\* [P]

## Appendix G SPA-1 Network traffic for ASIM powering off for extended period of time

<startup packets omitted>

```
(2012-12-12 13:04:44) [S] <11:r> 44 06 00* [P] ← TEMP ASIM at 0x11
(2012-12-12 13:04:44) [S] <11:r> 01 01 e1 7a cc 41* [P]
(2012-12-12 13:04:44) [S] <12:r> ff ff ff* [P]
(2012-12-12 13:04:44) [S] <13:w>* [P]
(2012-12-12 13:04:44) [S] <13:w>* [P]
(2012-12-12 13:04:44) [S] <11:r>* ff* [P] ← TEMP offline
```

<packets omitted...1.5 minutes pass>

```
(2012-12-12 13:06:13) [S] <11:w>* [P]
(2012-12-12 13:06:13) [S] <11:w> 5a 00 00 [P] ← re-enumeration for TEMP ASIM
(2012-12-12 13:06:13) [S] <11:r> 48 04 00* [P]
(2012-12-12 13:06:13) [S] <11:r> d2 02 96 49* [P]
(2012-12-12 13:06:13) [S] <11:w> 4d 02 00 01 01 [P] ← subscribe to messages
(2012-12-12 13:06:14) [S] <11:w> 4d 02 00 01 02 [P]
(2012-12-12 13:06:14) [S] <12:r> ff ff ff* [P]
(2012-12-12 13:06:14) [S] <13:w>* [P]
(2012-12-12 13:06:14) [S] <13:w>* [P]
(2012-12-12 13:06:14) [S] <11:r> ff ff ff* [P]
(2012-12-12 13:06:14) [S] <12:r> ff ff ff* [P]
(2012-12-12 13:06:14) [S] <13:w>* [P]
(2012-12-12 13:06:14) [S] <13:w>* [P]
(2012-12-12 13:06:14) [S] <11:r> ff ff ff* [P]
(2012-12-12 13:06:14) [S] <12:r> ff ff ff* [P]
(2012-12-12 13:06:14) [S] <13:w>* [P]
(2012-12-12 13:06:14) [S] <13:w>* [P]
(2012-12-12 13:06:14) [S] <11:r> 44 06 00* [P] ← TEMP ASIM at 0x11
(2012-12-12 13:06:15) [S] <11:r> 01 01 7b 14 be 41* [P]
(2012-12-12 13:06:15) [S] <12:r> ff ff ff* [P]
(2012-12-12 13:06:15) [S] <13:w>* [P]
(2012-12-12 13:06:15) [S] <13:w>* [P]
```

## Appendix H SPA-1 Network traffic for Round Robin Count greater than 100

<startup packets omitted>

```
(2012-12-12 12:41:16) [S] <11:r> ff ff ff* [P]
(2012-12-12 12:41:16) [S] <12:r> ff ff ff* [P]
(2012-12-12 12:41:16) [S] <13:w>* [P]
(2012-12-12 12:41:16) [S] <13:w>* [P] ← 100th Round Robin
(2012-12-12 12:41:16) [S] <11:w> 5a 00 00 [P] → Re-enumerate TEMP ASIM
(2012-12-12 12:41:16) [S] <11:r> 48 04 00* [P] →
(2012-12-12 12:41:16) [S] <11:r> d2 02 96 49* [P]
(2012-12-12 12:41:16) [S] <12:w> 5a 00 00 [P] → Re-enumerate LED ASIM
(2012-12-12 12:41:16) [S] <12:r> 48 04 00* [P] →
(2012-12-12 12:41:16) [S] <12:r> 35 38 d0 8b* [P]
(2012-12-12 12:41:16) [S] <13:w>* [P]
(2012-12-12 12:41:17) [S] <13:w>* [P]
(2012-12-12 12:41:17) [S] <13:w>* [P]
(2012-12-12 12:41:17) [S] <13:w>* [P]
(2012-12-12 12:41:17) [S] <11:r> 44 06 00* [P] ← resume normal operation
(2012-12-12 12:41:17) [S] <11:r> 01 01 d7 a3 cc 41* [P]
(2012-12-12 12:41:17) [S] <12:r> ff ff ff* [P]
(2012-12-12 12:41:17) [S] <13:w>* [P]
(2012-12-12 12:41:17) [S] <13:w>* [P]
(2012-12-12 12:41:17) [S] <11:r> ff ff ff* [P]
(2012-12-12 12:41:17) [S] <12:r> ff ff ff* [P]
(2012-12-12 12:41:17) [S] <13:w>* [P]
(2012-12-12 12:41:17) [S] <13:w>* [P]
(2012-12-12 12:41:17) [S] <12:w> 56 03 00 01 01 aa [P]
```

## List of Acronyms

1U	1 Unit
ACK	Acknowledge
AFRL	Air Force Research Lab
API	Application Programming Interface
ARC	Ames Research Center
ARP	Address Resolution Protocol
ARM	Advanced RISC Machines
ASIM	Appliqué Sensor Interface Modules
C&DH	Command and Data Handling
Cal Poly	California Polytechnic State University
CDD	Common Data Dictionary
CDS	CubeSat Design Spec
CSLI	CubeSat Launch Initiative
COSMIAC	Configurable Space Microsystems Innovations and Applications Center
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CSK	CubeSat Kit
ELaNa	Educational Launch of Nanosatellites
ELC	EXPRESS Rack Laptop Computer
EPS	Electrical Power System
EXPRESS	EXpediting the PROcess of Experiments to the Space Station
FAT	File Allocation Table
FIRSTLab	Flash Incident Radiation Susceptibility Test Lab
FW	Firmware
GPIO	General Purpose Input-Output
GUID	Global Unique IDentifier
HOSC	Huntsville Operations Support Center

I2C	Inter-Integrated Circuit
ICD	Interface Control Document
IDE	Integrated Development Environment
IDIQ	Indefinite Delivery Indefinite Quantity
IMU	Inertial Measurement Unit
ISS	International Space Station
LSP	Launch Services Provider
LV	Launch Vehicle
MISRA	Motor Industry Software Reliability Association
MisST	Microsatellite in-situ Space Technology
MSD	Mass Storage Device
MSFC	Marshall Space Flight Center
NACK	Negatively Acknowledge
NASA	National Aeronautics and Space Administration
NiMH	Nickel-metal Hydride
OPAL	Orbiting PicoSat Launcher
OS	Operating System
OTG	On-The-Go
P-POD	Poly Picosatellite Orbital Deployer
PIC	Peripheral Interface Controller
PIMS	Principal Investigator Microgravity Services
PnP	Plug-and-Play
PnPSat	Plug-and-Play Satellite
PRO	Payload Rack Officer
RAMPART CUBESAT	Rapid prototyped Mems Propulsion And Radiation Test CUBEflow SATellite
RTC	Real-Time Clock
RTU	Remote Transceiver Unit
SAA	Space Act Agreement
SD	Secure Digital

SDM	Satellite Data Model
SPA	Space Plug-and-Play Avionics
SPI	Serial Peripheral Interface
SSL	Space Systems Lab
STP	Space Test Program
STS	Space Transportation System
SVN	Subversion
TDRSS	Tracking and Data Relay Satellite System
UART	Universal Asynchronous Receiver/Transmitter
UK	University of Kentucky
USB	Universal Serial Bus
XML	eXtensible Markup Language
xTEDS	Extensible Transducer Electronic DataSheet



## References

- [1] James Lyke et al., "A Plug-and-play Approach Based on the I2C Standard," in *24th Annual Small Satellite Conference*, Logan, UT, 2010, pp. 1-13.
- [2] Jim Lyke, Don Fronterhouse, Scott Cannon, Denise Lanza, and Wheaton Byers, "Space Plug-and-Play Avionics," in *3rd Responsive Space Conference*, Los Angeles, CA, 2005, p. 12.
- [3] (2012, February) AAC Microtech Website. [Online].  
<http://pnp.aacmicrotec.com/index.php/aac-products.html>
- [4] The CubeSat Program. CubeSat Design Specification. [Online].  
[http://cubesat.org/images/developers/cds\\_rev12.pdf](http://cubesat.org/images/developers/cds_rev12.pdf)
- [5] Alexander Chin et al., "Standardization Promotes Flexibility: A Review of CubeSats' Success," in *AIAA/6th Responsive Space Conference*, Los Angeles, 2008, pp. 1-9.
- [6] Michael Swartwout, "Attack of the CubeSats: A Statistical Look," in *25th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, 2011, pp. 1-15.
- [7] Garrett Lee Skrobot and Roland Coelho, "ELaNa - Educational Launch of Nanosatellite; Providing Routing RideShare Opportunities," in *26th Annual Conference on SmallSats*, Logan, UT, 2012, pp. 1-6.
- [8] Andrew E Kalman. (2011, August) CubeSat. [Online].  
[http://cubesat.org/images/2011\\_Summer\\_Workshop/pumpkin\\_csdwlu\\_2011-2.pdf](http://cubesat.org/images/2011_Summer_Workshop/pumpkin_csdwlu_2011-2.pdf)
- [9] Jordi Puig-Suari et al., "Recent CubeSat Launch Experiences on U.S. Launch Vehicles," in *25th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, 2011, pp. 1-6.
- [10] Ryan Hevner, Jordi Puig-Suari, and Robert Twiggs, "An Advanced Standard for CubeSats," in *25th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, 2011, pp. 1-12.
- [11] (2005) NASA official website. [Online].  
[http://www.nasa.gov/pdf/181091main\\_1-Public%20Law%20109-155.pdf](http://www.nasa.gov/pdf/181091main_1-Public%20Law%20109-155.pdf)
- [12] ISS National Laborator Office. (2010, June) Payload Developers and Principal investigators Payload Planning, Integration and Operations Primer. [Online].  
[http://www.nasa.gov/pdf/501115main\\_ISS\\_Payload\\_Integration\\_Process\\_Primer\\_final\\_submission\\_baseline.pdf](http://www.nasa.gov/pdf/501115main_ISS_Payload_Integration_Process_Primer_final_submission_baseline.pdf)
- [13] NASA Space Act Agreement with NanoRacks. [Online].  
[http://www.nasa.gov/pdf/387938main\\_SAA\\_SOMD\\_6355\\_Nanoracks\\_ISS\\_National\\_Lab.pdf](http://www.nasa.gov/pdf/387938main_SAA_SOMD_6355_Nanoracks_ISS_National_Lab.pdf)
- [14] James E. Lumpp Jr, Daniel M. Erb, Twyman S. Clements, Jason T. Rexroat, and Michael D. Johnson, "The CubeLab Standard for Improved Access to the

- International Space Station," *IEEE*, 2011.
- [15] Zach Jacobs and Max Bezold. (2011) University of Kentucky Space Systems Laboratory. [Online]. <http://ssl.engineering.uky.edu/files/2011/01/8400-NRP-ICD-1.pdf>
  - [16] (2012, February) NASA Marshall Space Flight Center Microgravity Research Program Website. [Online]. <http://www.nasa.gov/centers/marshall/news/background/facts/microgravity.html>
  - [17] Don Fronterhouse and Jim Lyke, "Plug-and-Play Satellite (PnPSat)," in *First Annual International SpaceWire Conference*, 2007, pp. 1-11.
  - [18] Gilbert Moore et al., "3D Printing and MEMS Propulsion for the RAMPART 2U CUBESAT," in *24th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, 2010, p. 10.
  - [19] Fredrik C. Bruhn et al., "QuadSat/PnP: A Space-Plug-and-play Architecture (SPA) Compliant Nanosatellite," in *Infotech@Aerospace*, St. Louis, 2011, p. 12.
  - [20] Craig J. Kief and K Brian Zufelt, "Trailblazer: Proof of Concept CubeSat Mission for SPA-1," in *AIAA Infotech*, St. Louis, 2011, pp. 1-7.
  - [21] (2012, February) Cosmiac Web site. [Online]. <http://www.cosmiac.org/trailblazer.html>
  - [22] (2012, February) CubeSat Kit latest news 2011. [Online]. <http://www.cubesatkit.com/content/news-2011.html>
  - [23] (2007, November) Guidelines for the Use of the C Language in Critical Systems. PDF, ISBN 978-0-9524156-7-1. [Online]. <http://193.35.217.33/Publications/tabid/57/Default.aspx#label-c2>
  - [24] Jack G. Ganssle. (2012, December) A Firmware Development Standard. [Online]. [www.ganssle.com/misc/fsm.doc](http://www.ganssle.com/misc/fsm.doc)
  - [25] Greg D. Manyak, Fault Tolerant and Flexible CubeSat Software Architecture, 2011, Master's Thesis at California Polytechnic State University.
  - [26] Robert Bogue. (2006, January) Effective Code Reviews Without the Pain. [Online]. <http://www.developer.com/tech/article.php/3579756/Effective-Code-Reviews-Without-the-Pain.htm>
  - [27] SmartBear. (December, 2012) 11 Best Practices for Peer Code Review. [Online]. <http://smartbear.com/SmartBear/media/pdfs/WP-CC-11-Best-Practices-of-Peer-Code-Review.pdf>
  - [28] Space Dynamics Lab. (2012, December) PnP Space Software Group. [Online]. <https://pnpsoftware.sdl.usu.edu/>
  - [29] COSMIAC. (2012, December) CubeFlow Training. [Online]. <http://www.cosmiac.org/cubeflow-training.html>
  - [30] Daniel et al. Erb, "Kentucky Space: A Multi-University Small Satellite Enterprise," *23rd Annual AIAA/USU Conference on Small Satellites*, 2009.
  - [31] (1999) Universal Serial Bus Web site. [Online]. [http://www.usb.org/developers/devclass\\_docs/usbmassbulk\\_10.pdf](http://www.usb.org/developers/devclass_docs/usbmassbulk_10.pdf)

- [32] (2010) Ducommun Inc. Web site. [Online]. <http://phx.corporate-ir.net/phoenix.zhtml?c=70735&p=irol-newsArticle&ID=1380115&highlight=>
- [33] Zachary Jacobs, Samir A. Rawashdeh, and James E. Lumppp Jr, "A SPA-1 Enabled Plug-and-Play CubeLab for ISS Payloads," in *Infotech@Aerospace*, Garden Grove, CA, 2012, pp. 1-10.

## VITA

Zachary Andrew Jacobs

### Education:

Bachelor of Science in Electrical Engineering, minor in Computer Science  
University of Kentucky, May 2007

### Experience:

#### **Research Assistant**

August 2009 - current  
Space Systems Laboratory, University of Kentucky  
Lexington, KY.

#### **Student Internship**

July 2011 – August 2011  
Small Spacecraft and Payload Technologies division, NASA Ames Research Center  
Moffett Field, CA

#### **Graduate Research Assistant**

May 2010 - August 2010  
Space Data Systems division, Los Alamos National Lab (LANL)  
Los Alamos, NM

### Publication:

Zachary Jacobs, Samir A. Rawashdeh, and James E. Lumpp Jr, "A SPA-1 Enabled Plug-and-Play CubeLab for ISS Payloads," in Infotech@Aerospace, Garden Grove, CA, 2012, pp. 1-10.

### Presentations:

**Infotech@Aerospace March 2012**  
**CubeSat Summer Workshop, Summer 2010**