Auto Code Generation for Simulink-Based Attitude Determination Control System

First Jose Molina[1]

NASA Marshall Space Flight Center, Huntsville, Alabama, 35808

This paper details the work done to auto generate C code from a Simulink-Based Attitude Determination Control System (ADCS) to be used in target platforms. NASA Marshall Engineers have developed an ADCS Simulink simulation to be used as a component for the flight software of a satellite. This generated code can be used for carrying out Hardware in the loop testing of components for a satellite in a convenient manner with easily tunable parameters. Due to the nature of the embedded hardware components such as microcontrollers, this simulation code cannot be used directly, as it is, on the target platform and must first be converted into C code; this process is known as auto code generation. In order to generate C code from this simulation; it must be modified to follow specific standards set in place by the auto code generation process. Some of these modifications include changing certain simulation models into their atomic representations which can bring new complications into the simulation. The execution order of these models can change based on these modifications. Great care must be taken in order to maintain a working simulation that can also be used for auto code generation. After modifying the ADCS simulation for the auto code generation process, it is shown that the difference between the output data of the former and that of the latter is between acceptable bounds. Thus, it can be said that the process is a success since all the output requirements are met. Based on these results, it can be argued that this generated C code can be effectively used by any desired platform as long as it follows the specific memory requirements established in the Simulink Model.

[1] José Molina, EV-41, Address/Mail Stop, and AIAA Member Grade (if any) for first author.

Nomenclature

(Nomenclature entries should have the units identified)

ADCS = Attitude Determination Control System

HWL = Hardware in the loop

FSW = Flight Software

NASA = National Aeronautics and Space Administration

I. Introduction

THIS document gives a general overview on the work done at NASA Marshall, between the period of January 15th, 2012 and April 28th 2012; on the Simulink Attitude Determination Control System (ADCS) auto code generation process. Topics covered include: project objectives, tools used, simulation output, necessary modifications to the simulation for the code generation process, test procedures for the auto generated code, problems encountered and how to mitigate these and finally conclusions on the process.

II. Project Objectives

There are two primary objectives for this project; the first is to generate portable 'C' from an Attitude Determination Control System (ADCS) Simulation developed in MATLAB (Ref. [1]) and Simulink (Ref. [2]) such as to be able to test it in any C capable platform. The secondary objective is to prepare laboratory that can use this simulation software and two computers to create a HWL environment in order to simulate the flight of a nano-satellite around the Earth.

III. Project Plan

In order to carry the project objectives some steps must be followed. These are outlined below:

1) Generate 'C' code from the ADCS Simulink Subsystem and port it to the flight computer.
2) Generate 'C' code from the Sensor Models and Satellite Dynamics Subsystems and port them to the rack computer.
3) Establish the HWL between the rack computer and the flight computer.
4) Run the code over a set amount of steps in both the simulation and HWL and log the output data of each to two separate files.

5) Compare both files by doing overlay plots and error plots.

IV. Attitude Determination Control System Simulation

This project is based off of the simulation off of an Attitude Determination Control System (ADCS) develop by EV-41 and EV-42 branches at NASA Marshall Space Flight Center. It consists of three Subsystems: Sensors Models, ADCS FSW, and Satellite Dynamics. This simulation is used to determine the dynamics of a satellite as it is orbiting around the Earth.

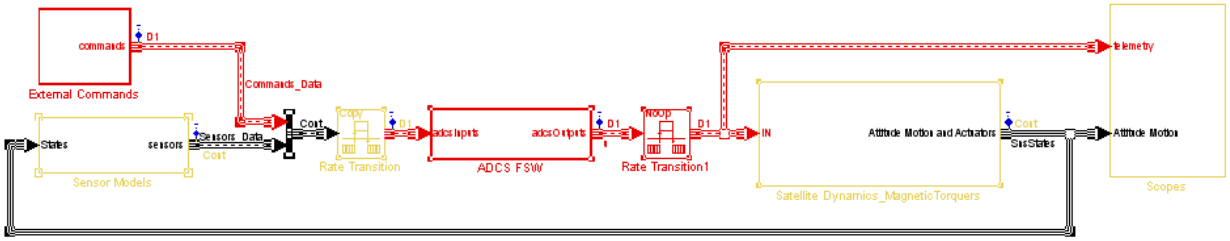

Fig. 1: CubeSat Sim Block Diagram

V. CubeSat Overview

A CubeSat is a "10 cm cube with a mass of up to 1.33 kg. (Ref. [5])". The goal of a CubeSat is to "provide access to space for small payloads. (Ref. [5])". CubeSats are launched in rockets within a deployer known as a P-POD. (Ref. [5]).

Fig. 2: CubeSat Structure

After the auto-code generation is finished the ‘C’ code will be compiled and ported to the flight computer of the CubeSat. The flight computer used for this project is a dsPIC33F256GP710 (Ref. [7]) microcontroller from Microchip. This microcontroller has 256 Kbytes of ROM memory and 30 Kbytes of RAM memory and is programmed in ANSI 'C' language. The ADCS FSW code must fit within these memories constraints in order to be executed successfully.

VI. SPRITE Laboratory Overview

The Small Projects Rapid Integration Test Environment (SPRITE) Laboratory was recently established to work on projects such as CubeSats. There are various components available for prototyping as well as two computers and three monitors to write and test software.

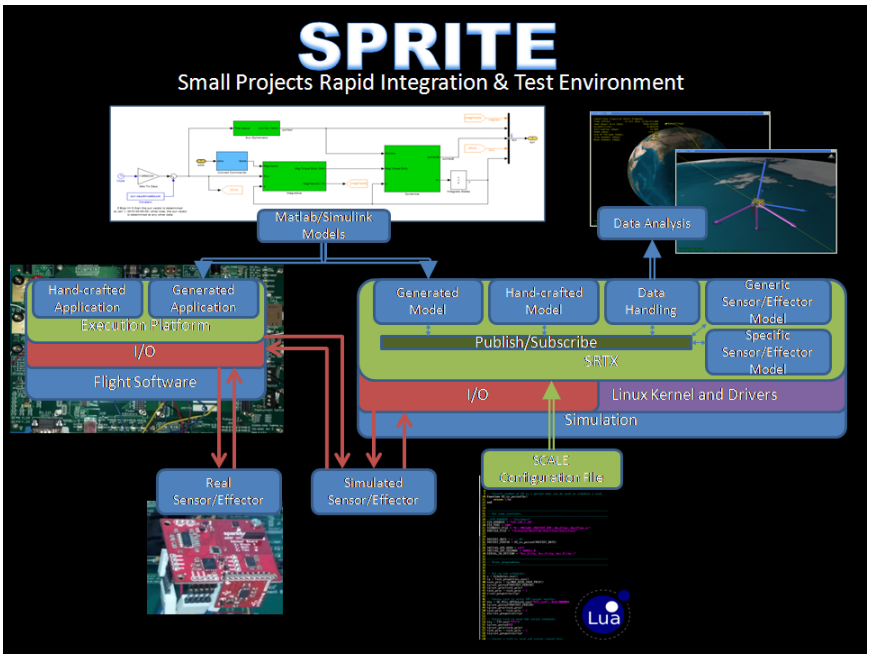

Fig. 3: SPRITE Laboratory Overview

VII. Hardware in the loop simulation

The Hardware in the loop Simulation consist of the Rack Computer running the Sensor Models and Satellite Dynamics, the CubeSats Flight Computer running the ADCS FSW, and the Monitor PC showing the output via a serial port or command line terminal.

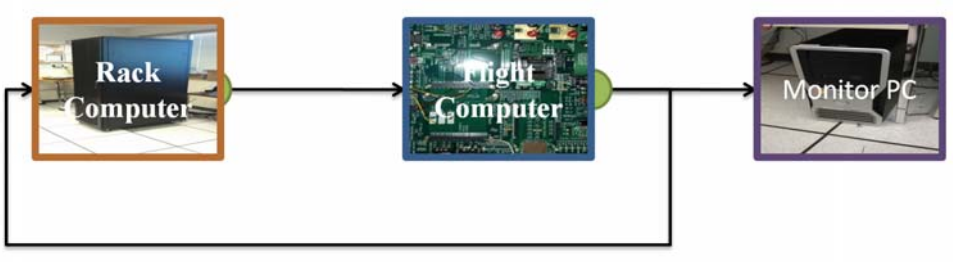

Fig. 4: Hardware in the loop simulation

This Hardware in the loop is the final desired result from the code generation process and can be used to test how the generated code will work once ported to real embedded hardware such as the Flight Computer.

VIII. Development Software and Hardware

In order to carry out and test the code generation process various software and hardware tools were required. These tools are listed and described below:

- MATLAB (Ref. [1]): is Software Package to program scripts to carry out complex mathematical situations and is widely used in NASA for Guidance and Control (GNC) software development.
- Simulink (Ref. [2]): is a sub-component of MATLAB (Ref. [1]) while bringing a simple user interface to connect blocks known as ‘Subsystems’ in order to create a ‘Model’ that will carry out specified math and logic calculations.
- Dev-C++ (Ref. [3]): is an Integrated Development Environment (IDE) for programming in C and C++
- Pumpkin CubeSat Kit: is a commercial Development Board for rapid integration of CubeSat satellites.
- Microhard Trans-Receiver: is a radio used to send and receive telemetry data to and from the CubeSat.

Fig. 5: Microhard Trans-Receiver

IX. Code Generation Procedure

The code generation procedure requires some preparation in order to work correctly. The following steps must be carried out before the process begins:

1. Create and test any given Simulink (Ref. [2]) simulation. Try to make sure to avoid as many warnings as possible
2. Determine which subsystems, (Simulink (Ref. [2]) simulation components), must be generated as reusable functions and convert these to atomic subsystems.
3. Re-test the Simulink (Ref. [2]) simulation and make sure the output is the same.

In order to convert Subsystems into their atomic representations the 'Treat as atomic unit' option must be checked in the Subsystem Parameters:

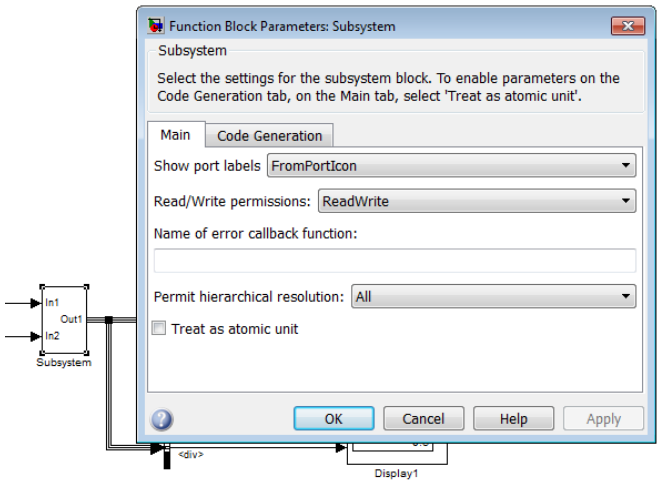

Fig. 6: Subsystem Parameters - Main

After the subsystem is converted to its atomic representation it can then be packaged as a 'C' function by going to the Code Generation Tab of the Subsystem Parameters and modifying the Function Packaging, Function Name, and File Name Options as necessary:

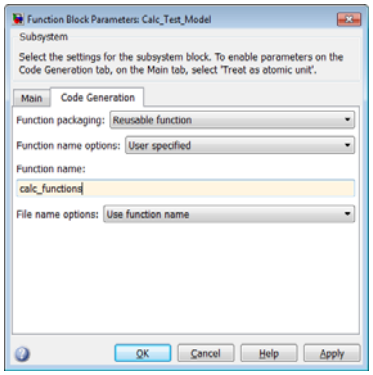

Fig. 7: Subsystem Parameters – Code Generation

Once the simulation is ready for auto-code generation the process can be started by going to the configuration parameters of the current model and clicking on generate code. Alternatively a Subsystem can also be auto-code generated by right clicking on it and selecting Code Generation > Build Subsystem:

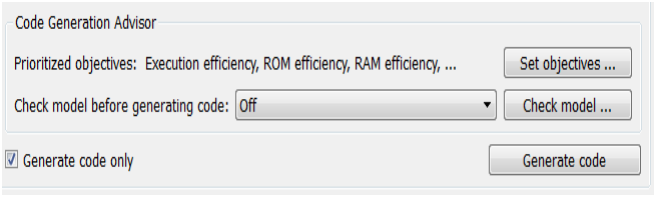

Fig. 8: Configuration Parameters

Once the code generation begins Simulink will show warnings based on the incorrectly selected Configuration Parameter options. Once these warnings are fixed or established as non-problematic the code generation process may start by clicking Build on the Variable Packaging Pop-up window:

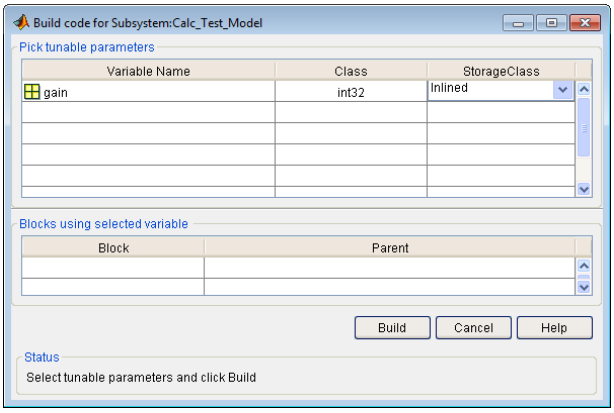

Fig. 9: Variable Packaging Window

If the process is successful the Code Generation Report window should be shown:

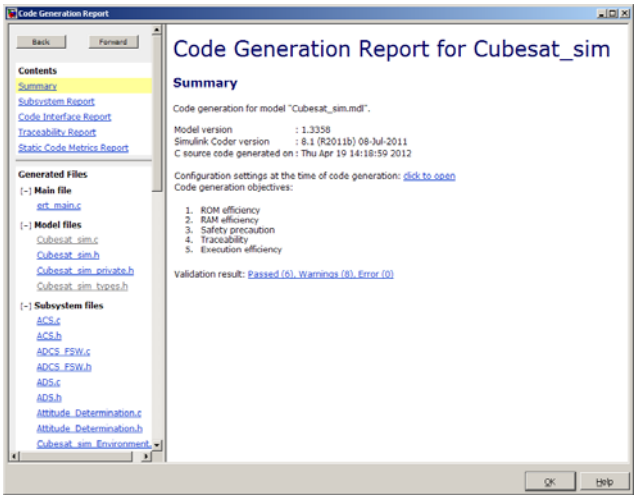

Fig. 10: Code Generation Report

This report contains various tools such as Code Traceability segments to compare generated code to the actual simulation blocks, warnings and errors encountered during the code generation process, summary of the process, etc. This Code Generation Report is generated as a set of portable HTML files which can be viewed in any web browser without the need for MATLAB (Ref. [1]).

In order to test the generated code the source files must be compiled into an executable. This can be done with any Integrated Development Environment (IDE) that can compile C code; one such IDE is Dev-C++ (Ref. [3]). Once the code is compiled it can be tested by viewing its output and comparing it to that of the Simulink model from which it was generated:

The generated code will always contain the following components which correspond to the Inputs, Processes and Outputs of the given subsystem or model:

- Model_Name_Step: This is the primary function for the simulation. Every time this function is called it's the equivalent of running the generated Subsystem for one step in the Simulink simulation.
- Model_Name_U: These _U variables/structures are the Inputs to the simulation or subsystem and they are to be set before calling the _Step function.
- Model_Name_Y: These _Y variables/structures are the Outputs of the simulation or subsystem and they are set after calling the _Step function.

X. Output Data Comparison

In order to compare the output of both the simulated and generated code two ".txt" files were created with the output data of the ADCS FSW Subsystem. A script was written in MATLAB (Ref. [1]) that would take the data from the two files and compare them and plot them and their differences. These plots are finally generated as MATLAB figures which can be seen below:

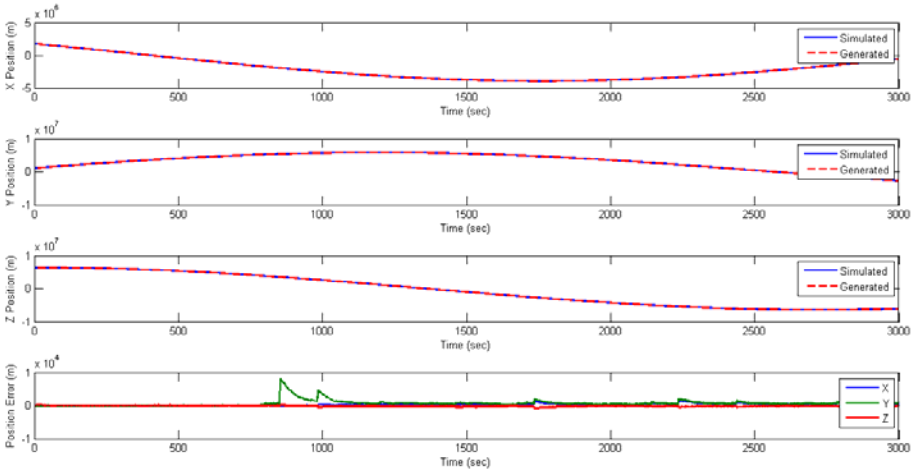

Figure 2: Satellite Position

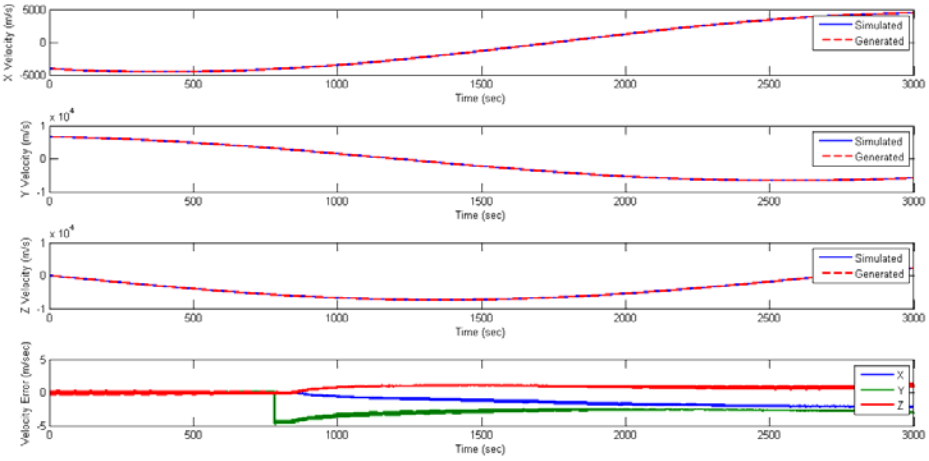

Figure 3: Satellite Velocity

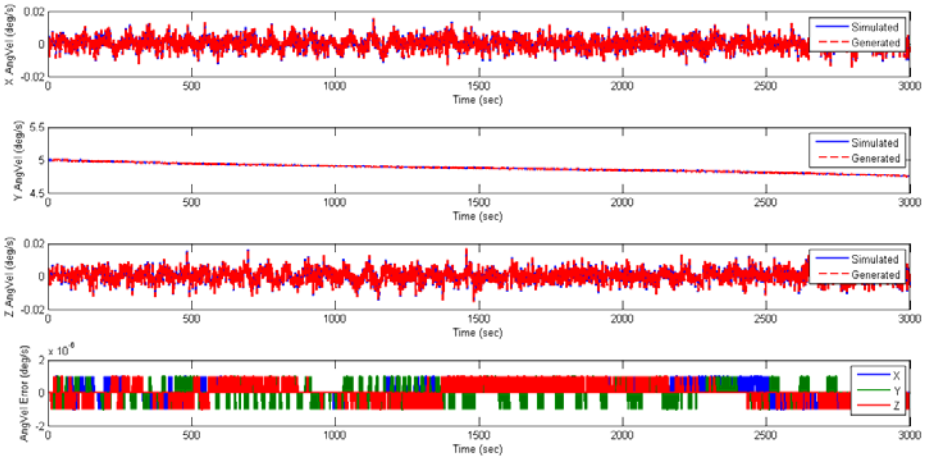

Fig. 15: Satellite Angular Velocity

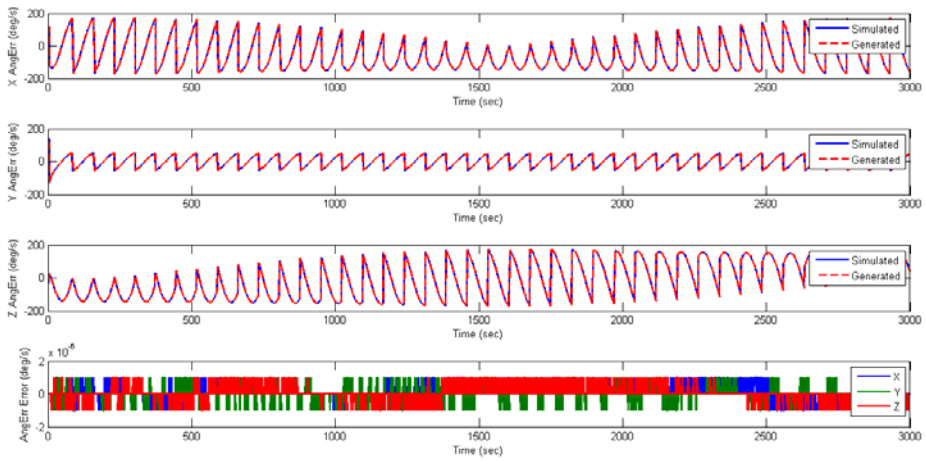

Fig. 16: Satellite Angular Errors

XI. Visual Output of Satellite Dynamics

Visual output of the satellite data can be seen through the AGI Satellite Toolkit (STK) software (Ref. [4]). This tool can be used to determine how the actual satellite motion should look once the satellite is in orbit and can be used to easily verify if the attitude of the satellite is as expected:

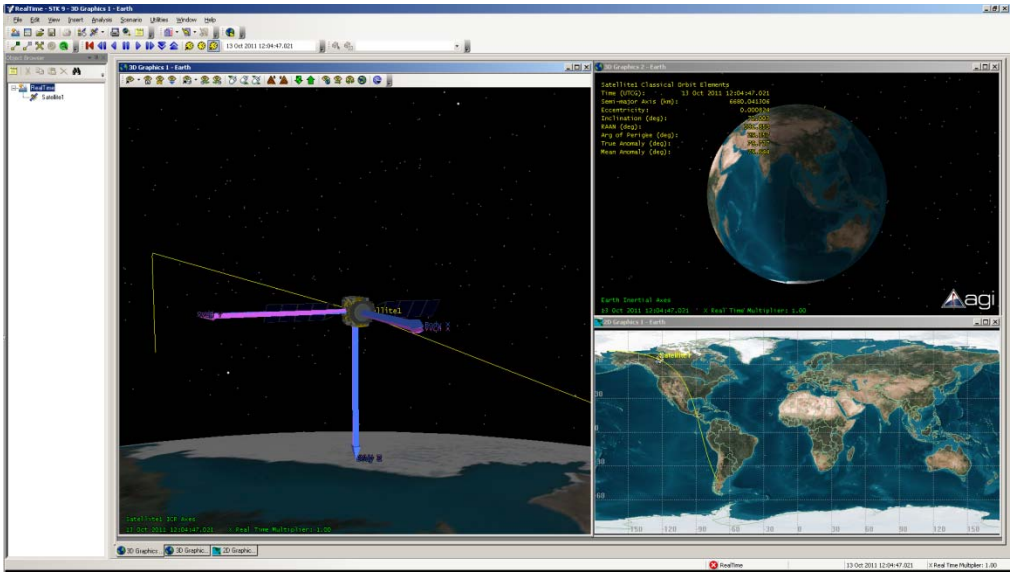

Fig. 17: STK Output

XII. Conclusion

The code generation process was a success based on the difference of the output data between the simulation and the generated being between acceptable bounds. It is noted that the output generated with the embedded hardware differs based on the platform difference and the use of a serial channel for data transmission. A possible fix to this problem is to use binary or raw data and establish a standard for data input and output on the embedded hardware and the dynamics computer. Future improvements include changing the output of the data to actual peripheral signals in order to simulate real sensors more effectively, add a GUI to the simulation, use the Pan Tilt and Roll arm to simulate the movement of the satellite around the orbit of the Earth and finally to test this procedure using different microcontrollers or microprocessors such as the GUMSTIX.

Acknowledgments

- *Puerto Rico NASA Space Grant*
- *NASA Marshall Space Flight Center*
- *Daniel Heater*
- *Dr. Pedro Capo Lugo and his family*
- *Devon Sanders*

- *Evan Anzalone*
- *John Rakoczy*
- *Bill Hopkins*

References

Computer Software

[1] MATLAB, The MathWorks, Inc., Software Package, Ver. R2011b, Natick, Massachusetts, 2011.

[2] Simulink, The MathWorks, Inc., Software Package, Ver. R2011b, Natick, Massachusetts, 2011.

[3] Dev-C++, Bloodshed Software, Software Package, Ver. 5, 2005.

[4] Satellite Toolkit, Analytical Graphics, Inc., Software Packer, Ver. 9.2.3, Exton, Pennsylvania, 2011

Publications

[5] CubeSat Design Specification Rev. 12. California State Polytechnic University. Retrieved 2010-10-16.