Developing Avionics Hardware and Software for Rocket Engine Testing

Bryce Aberg
Kennedy Space Center
Major: Electrical Engineering
NIFS Summer Session
Date: 25 July 2014

Developing Avionics Hardware and Software for Rocket Engine Testing

Bryce Aberg[1]
Western Kentucky University, Bowling Green, KY, 42101

Abstract

My summer was spent working as an intern at Kennedy Space Center in the Propulsion Avionics Branch of the NASA Engineering Directorate Avionics Division. The work that I was involved with was part of Rocket University's Project Neo, a small scale liquid rocket engine test bed. I began by learning about the layout of Neo in order to more fully understand what was required of me. I then developed software in LabView to gather and scale data from two flowmeters and integrated that code into the main control software. Next, I developed more LabView code to control an igniter circuit and integrated that into the main software, as well. Throughout the internship, I performed work that mechanics and technicians would do in order to maintain and assemble the engine.

Nomenclature

V	=	Volts
A	=	Amperes
Hz	=	Hertz
f	=	frequency
GPM	=	gallons per minute
LOX	=	Liquid Oxygen
LCH_4	=	Liquid Methane
GN_2	=	Gaseous Nitrogen
GHe	=	Gaseous Helium
SSME	=	Space Shuttle Main Engine
RIO	=	Reprogrammable Input/Output
SDP	=	Software Development Plan
SRS	=	Software Requirements Specification

I. Introduction

My internship project was providing support for the avionics hardware and software in Project Neo. Project Neo is the Propulsion Systems Test Bed which provides systems engineering experience with a liquid propulsion rocket and is a part of Rocket University at Kennedy Space Center. The Rocket U program aims to train and maintain flight systems engineering skills through hands-on projects and is comprised of numerous disciplines and projects including the Propulsion Systems Test Bed. The components of Neo include: a metal test skid capable of withstanding 6,000 lbs of thrust, an electronics cabinet that houses engine and test skid controls, the engine that was originally obtained from Armadillo Aerospace for a different project, and a cryogenic fluids system that delivers fuel and oxidizer to the engine. In the months leading up to my internship, the Neo team was planning on integrating electronics hardware with the control software so as to begin testing the avionics. My work directly supported this goal and consisted of several duties. First, I was tasked with learning about the layout of the engine and its schematic in order to better understand the system as a whole which also meant that I had to have some understanding of how liquid rockets work in general. Second, I worked with interfacing the engine hardware components with LabView for the purpose of sensor characterization and engine control. Lastly, I supported the development of required documentation for the software that I was involved with as well as other required documentation for the project.

[1] NASA NIFS Intern, Propulsion Avionics NE-A5, Kennedy Space Center, Western Kentucky University.

II. Rocket Engine Control 101

Coming from an electrical engineering background, my knowledge of rocketry was limited. Therefore, many of the concepts that I learned about while working for NASA at KSC were new concepts to me. In order to more fully understand the requirements for the software that I was going to write, I needed to have some understanding of how liquid rocket engines operate and how they are controlled. In this section, I will summarize what I learned about these topics and how they are applied in Project Neo.

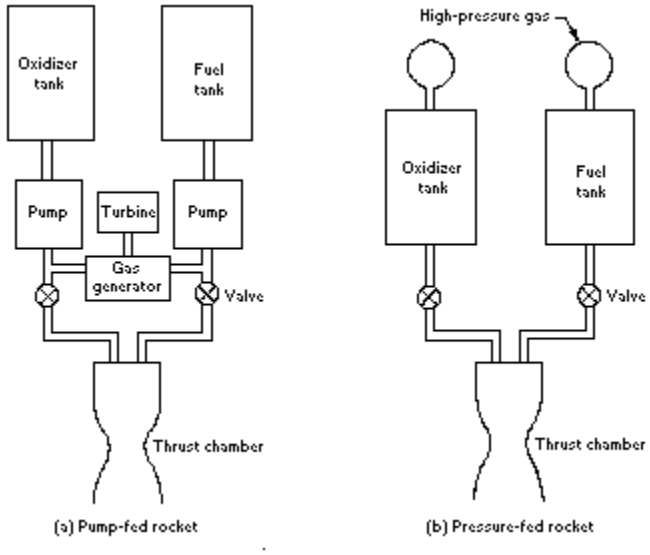

Figure 1. Liquid Rocket fuel delivery systems

A. Fuel Systems

Although many types of fuel can be used for a liquid rocket engine, most fuels require an oxidizer in the form of liquid oxygen (LOX) to combust. Neo uses liquid methane (LCH_4) as its fuel source and both it and the LOX are fed to the thrust chamber via a pressurized system. Larger rocket engines, such as the Space Shuttle Main Engine (SSME), require pumps[1] to transfer commodities to the thrust chamber. The difference between these two systems can be seen in Fig. 1 with the pressure-fed rocket being simpler than the pump-fed rocket. So b) in Fig. 1 can be seen as a much simplified version of the Neo engine schematic. The way that the mixture of the fuel and oxidizer is controlled is through the use of valve actuators on the engine. Simply put, when the fuel valve is open, fuel is flowing to the engine.

B. "Chilldown" and Purging

A "chilldown" prior to a rocket's ignition[2] must be performed to ensure that the two commodities are liquids when they reach the combustion chamber. Because the condensation point for LOX is -297° F and LCH_4 is -259° F, the pipes that they flow through must be cold enough so they do not cause flash evaporation of the propellants upon contact with the engine propellant lines. Therefore, the fuel and oxidizer inlet valves are opened during engine pre-start operations to allow time to chill the lines. As the liquids slowly fill the pipes up to the main control valves, they naturally evaporate into gases upon contact with the ambient propellant lines, which causes the commodities to evaporate through bleed valves prior to starting the engine. Eventually, the only molecules in the pipes after chilldown should be the liquid commodities. On Neo, prior to chilldown, the fuel and oxidizer lines must be purged in order to ensure that the lines are not contaminated with molecules that could react with either the liquid methane or LOX. The fuel and oxidizer lines are flushed with either gaseous nitrogen (GN_2) or gaseous helium (GHe), both inert commodities, in order to make sure that the lines are clean and dry. After the lines are flushed, there is only inert purge gas in the lines, which are non-reactive with oxygen and methane. If the lines had some foreign molecules in them that were not gaseous at extremely cold temperatures, they could freeze due to the propellant commodities' cryogenic temperatures and cause formation of ice that could block the lines and create a dangerous pressure buildup.

C. Neo Engine Layout

One of the future goals of Project Neo is to eventually attach the engine to a flight vehicle. Because of this goal, the electronics on Neo are split into two groups: test skid controls and engine controls. Both groups consist of the same types of components including a National Instruments Compact RIO, actuated valves, solenoid valves, pressure sensors, and temperature sensors. However, the two groups have different purposes; the test skid controls exist to provide power to the entire skid and to control distribution of ground supplied propellants to the engine, while the engine controls exist to control the performance of the engine. Because I was an intern in the propulsion avionics branch, I worked solely with the avionics group of electronics.

For each commodity line, there are three types of valves that control where the commodity goes in the engine. These are the main valves, bleed valves, and igniter valves. The main valves are linearly actuated which allow them

to be anywhere between 0% and 100% open, much like a garden hose nozzle, and they control how much of commodity flows into the engine's thrust chamber when the rocket is firing. The bleed valves are controlled by a solenoid and allow gas to escape into the atmosphere when the engine is purged and chilled prior to engine start. The igniter valves are also controlled by solenoids and they allow propellant to flow into a chamber that serves as a pilot light for the engine. In addition to the valves, the sensors that are attached to each line include: two thermocouples, one pressure sensor, and one flowmeter. The purpose of the pressure sensor and thermocouples is to monitor the status of the engine's fuel and oxidizer lines in order to characterize engine performance and detect any hazardous situations. As an example, a hazardous situation could occur if one of the pipes starts leaking LCH_4 which could then ignite and cause a fire. The purpose of the flowmeter is to characterize the relationship between the main valve's actuator position and flow rate. Because the main valve can be partially open, the rate of flow through the pipes will be variable therefore, the flowmeter helps us know what the flow rate will be when the valve is open to a certain percentage. I will talk about the flowmeter in a later section.

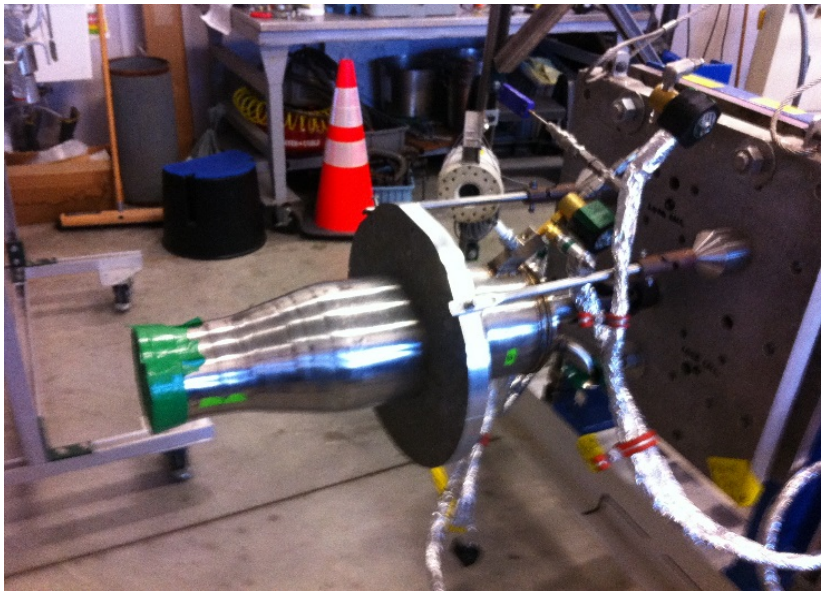

Figure 2. Disassembled Neo flight engine

Finally, attached to the actual nozzle of the engine will be two actuators, nine thermocouples, a pressure sensor, and an igniter. Actuators will eventually replace the two rods that are attached on either side of the nozzle shown in Fig. 2 and they will control gimballing. The pressure sensors and thermocouples will monitor the combustion pressure and wall temperature of the nozzle, much like the ones on the propellant lines. Lastly, the igniter consists of a sparkplug and a small, enclosed chamber that the igniter valves feed into that will ignite the commodities to start the engine.

III. Engine Interface in LabView

A. Engine Controller Overview and Difficulties

Attached to the skid is an electrical box that contains the majority of the electronics for Neo. Power for the box when Neo is being tested outside will come from power supplies located in a mobile trailer, but while in the garage Neo is powered by a power supply plugged into a wall outlet. Included in the list of electronics are two Compact RIO (cRIO) systems made by National Instruments: one for control over the test skid and one for control over the engine. The engine cRIO, which is the RIO that I worked with for this internship, has eight slots for modules that have functions like analog voltage I/O, digital I/O, current measurement, and thermocouple measurement. All engine monitoring devices like the flowmeters, thermocouples, pressure sensors and controls like the solenoid or actuator valves eventually interact with this cRIO. My experience with National Instruments hardware prior to this internship was limited to the myDAQ and myRIO devices whose target audiences were students and universities. The difference between these devices and the cRIO is that the cRIO consists of an embedded controller, programmable FPGA, and the I/O modules mentioned above while the other two have all their hardware embedded onto the same board. In addition to the embedded nature of the latter, NI simplified the procedure to interface with them in Labview because they are targeted at students and universities.

Because the myDAQ did not have a programmable FPGA and I never used the FPGA on the myRIO before, I had some initial difficulty when trying to program the cRIO. I learned through analyzing the existing Labview code that if I want to send/receive data to/from the I/O modules on the cRIO, there needs to be two virtual instruments (VI's) running on the cRIO: one on the FPGA to interface with the I/O modules and one on the cRIO controller to interface with the FPGA. Prior to this revelation, I hit my first snag because I was under the impression that a VI on the FPGA alone would be able to perform the tasks that I intended for it to perform. After compiling some code that

I had written for the FPGA to test the flowmeters, an error was encountered that I had never seen before. In simple terms the clock on the FPGA was not giving the code enough time to execute each cycle, so my initial fix for the problem was to decrease the clock rate from 80 MHz to 40 MHz. Unfortunately, this fix was only a temporary one because the code I had written to test the flowmeters existed in a vacuum. In reality, that code needed to be integrated into the larger main LabView project and the FPGA VI for that project ran at 80 MHz. I realized that my VI would not work with the existing FPGA VI because that VI was used only for sending/receiving data to/from the I/O modules. That explained why the clock was originally set to 80 MHz. With this knowledge, I was able to modify to the existing code for my own purposes.

B. Flowmeters

The Neo team acquired two off-the-shelf flowmeters to measure commodity flow. The purpose of the flowmeters is to compare the percentage that the main valves are open with the amount of commodity that are flowing through them. The flowmeters consist of two parts: a turbine flowmeter that outputs an analog voltage and a microprocessor that converts the flowmeter output into a linearized 4-20 mA output. The microprocessor is unique for this project because it outputs a current rather than a voltage. Therefore, the current has to be measured with a cRIO module specifically for current measurement as opposed to an analog voltage input module. Because the microprocessor is part of a loop powered transmitter, there aren't any ground pins connected to it. Rather, the negative end of the power source serves as ground and the chassis of the microprocessor and flowmeter are connected to that.

The flowmeter turbine is connected to a magnetic pickup coil that sends pulses to the microcontroller which then measures the frequency of those pulses. From there, the microcontroller converts the frequency into a linear 4-20 mA signal. This characterization means that if 4 mA is measured, the flow through the flowmeter is 0 GPM. The max flow rate, however, depends on the flowmeter being used; the fuel flowmeter has a max flow rate of 130 GPm while the oxidizer flowmeter has a max flow rate of 90 GPM. Regardless of the flowmeter used, 20 mA output from the microprocessor corresponds to its respective flowmeter's maximum flow rate. But what if the current is in between 4 mA and 20 mA? Fortunately, the manufacturer provides an equation to convert from current to GPM in their user manual. The microcontroller uses the following equations to convert frequency to flowrate to current.

$$Flow\ Rate = \frac{f}{Kfactor} \times 60 \text{ (Ref. 3)} \tag{1}$$

The Kfactor in the above equation is a value used to scale the frequency of the pickup pulses to a flow rate. The physical units for the value are pulses/gallon and it is calculated by running a gallon of water through each flowmeter during calibration. After calculating the flow rate of the liquid, the microprocessor then uses the following equation to convert to a current:

$$Current = 4\ mA + \left(16\ mA\ \times \frac{Flow\ Rate}{Max\ Flow\ Rate}\right) \text{ (Ref. 3)} \tag{2}$$

By moving some of the terms around, I came to an equation that can convert a 4-20 mA current to a flow rate measured in gallons per minute:

$$Flow\ Rate = (Current - 4\ mA) \times \frac{Max\ Flow\ Rate}{16\ mA} \tag{3}$$

Equation 3 is obviously what I used to scale the output from the microprocessor in LabView. From working with the flowmeters and discussing them with my mentor, I learned that data scaling is incredibly important. If a piece of raw data is scaled incorrectly or not at all, then it is almost useless and should not be taken into consideration for control purposes. On the other hand, a piece of data can be scaled correctly but if it cannot be retrieved in a timely manner, then it, too, is useless. I thought that I encountered this issue because the user manual for the microprocessors stated that their response time was 2 seconds, which is too slow to control Neo's engine. I called the manufacturer to confirm that that information was indeed correct and they actually told me that there was a misprint in the user manual that came with the flowmeters. In reality, the response time is $1/8^{th}$ of a second, which proves that investigating how a piece of hardware works is worth the time to understand its specific applications.

C. Igniter Circuit Control

The purpose of the igniter circuit is to cause a spark plug in the combustion chamber of the engine to spark at a certain frequency. Although I was not involved in creating the circuit itself, I was given its schematic and the circuit had initially had three analog voltage inputs and two analog voltage outputs. The first two inputs armed and fired the igniter, while the third input allowed for the frequency of the spark to be set in software. The timing for the sparks were controlled by a 555-timer, and the only caveat was that the timer had to be isolated from the rest of the circuit. This lead to a problem because the rest of the circuit, along with all the other electronics on the engine, were connected to the same ground. If the systems engineers wanted to keep the ability to set the igniter frequency in software, the solution discussed would have been to obtain an entirely new cRIO and power supply just for the timing circuit. The practical solution, and the one that was eventually agreed upon by the electronics and avionics teams, was to replace the input with a potentiometer. This solution did two things: (1) it removed the need for the frequency to be set in software and (2) it still allowed for the frequency to be adjusted. The outputs of the circuit just measure the current flowing through the spark plug as well as the voltage across the battery that powers it.

IV. Physical Work on Project Neo

Project Neo is really about the hardware that it is made up of. Like any other project, many man hours have to be spent turning wrenches, drilling holes, and soldering wires in order to get all of the components fastened together. I performed work that technicians and mechanics perform on a daily basis. When the other interns and I arrived for the summer, the Neo team had actually already started to disassemble the engine for cleaning. Although I did not get a chance to disassemble the parts that were already on the engine and test skid, there were plenty of other things to take apart. For instance, there were several metal boxes containing electronics and small pumps that needed to be salvaged for parts, so I worked with some of the other interns to take those parts out and sort them for later use. I also assisted one of the engineers in splicing wires connected to an actuator and I drilled holes in thick metal plates that are used to keep the skid in place.

The microprocessor for the flowmeter was ordered with a blast-proof chassis which protects the electronics inside and allows the microprocessor to be mounted to the test skid. Figure 5 shows a temporary mounting and wiring configuration for the fuel line flowmeter and its microcontroller. Holes had to be drilled in order to mount the chassis to the skid and, since there no pipes to connect to, the flowmeter was initially just fastened to the skid using zip-ties. 22-18 gauge wire was used to temporarily connect the two for testing purposes, but permanent wiring harnesses were eventually installed by the electrical team.

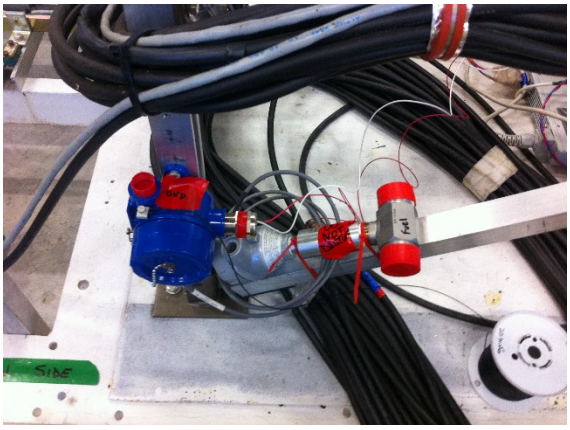

Figure 3. Flowmeter microcontroller chassis (blue) installed with temporary wiring to the flowmeter (silver)

V. Software Documentation

One of the areas that I worked in that I had a limited amount of experience in was that of formal software documentation. Although I have completed documentation for projects that I have done in school, I never had to strictly structure them or go into as much detail as I did for Neo. Through working on the documentation for this project, I learned that documenting the work that I do is critical in avoiding mistakes and working efficiently.

A. Software Development Plan (SDP)

Many pieces of software have an SDP that outlines the process that is used to develop them. Because I did not have any prior knowledge of the software development process at NASA, my mentor gave me a different project's SDP to read over to understand its format and content that I would need to include in the Neo SDP. However, because the other project and Neo are quite different from each other, the development process for Neo had to be different from what I was reading but I was still able to identify sections that could be adapted to Neo's SDP. Fortunately, there was also a very rough draft of an SDP that had already been drawn up for Neo. I eventually met with one of the software engineers and we went through the rough draft and added and deleted different portions of the document which left us with a baseline document draft that then went on to get reviewed by safety personnel.

B. Software Requirements Specification (SRS)

Of the two documents that I worked on, the SRS for the avionics was the more technical of the two. The purpose of the document is to spell out exactly what the systems engineers need the associated software to do for them. Because Rocket U is not as structured as other KSC programs, the Neo avionics team was allowed to develop the software and its requirements in parallel to expedite the development process. According to my mentor, though, this approach is not often the route that is taken. My mentor gave me the RS-25 rocket engine's SRS to read over to become familiar with the types of requirements that the Neo avionics would need. While I was reading the document, I identified sections that Neo would need, like sensor data scaling, and took note of the type of verbiage used in them to serve as a reference point for the Neo SRS. As I progressed through the internship, I became more familiar with the avionics such as the flowmeters and the igniter circuit. With the help of manufacturer component datasheets and user manuals, I was able to contribute to the NEO SRS by adding scaling requirements.

VI. Conclusion

Spending the summer working at the Kennedy Space Center has been the most educational experience I have ever had. Not only did I get to develop my LabView skills by interfacing with the various sensors on Project Neo, but I also got to interact with personnel in a real working environment. Working beside NASA employees has shown me how engineers work and develop technologies in the real world, which is something that cannot be taught easily in the classroom.

Acknowledgments

I want to thank my mentor, Ken Jacobs, for providing insight and guidance on the projects that I worked on. I would also like to thank the NASA Kentucky Space Grant for the funding of this internship opportunity.

References

[1]NASA, MSFC, "Space Shuttle Main Engine (SSME) Enhancements," FS-2002-03-60-MSFC, 2002.

[2]Banica, M., and Wisse, M., "Dynamic Chilldown Model of a Liquid Hydrogen Feed System," *Journal of Propulsion and Power*, Vol. 29, No. 4, 2013, pp. 975.

[3]*Model: CAT1 Microprocessor Controlled Two-Wire 4-20mA Loop Powered Transmitter*, Hoffer Flow Controls, Inc., Elizabeth City, 2014, Sec. 3.

[4]*NI 9219 Opearting Instructions and Specifications*, National Instruments Corp., Hungary, 2009, pp. 7.