



S4 GUIDE

[http:// s4.sonoma.edu/guide.html](http://s4.sonoma.edu/guide.html)



Table of Content

1	Introduction – About This Guide
3	1 Background Information - Rocketry
3	1.1 Historical Overview of Rocketry
5	1.2 Rocketry for the Secondary School Classroom
5	1.3 Model Rocketry vs. High Powered Rocketry
7	2 Background Information - Ballooning
7	2.1 Historical Overview of Ballooning
8	2.2 Ballooning for the Secondary School Classroom
9	3 Background Information – Electronics, Soldering and Computer Programming
9	3.1 Basics of Electronic Circuits and Measurements
9	3.1.1 Voltage, Current and Resistance
10	3.1.2 Combining Components in Circuits
11	3.1.3 Measuring Resistance, Voltage and Current in a Circuit
12	3.1.4 Analog and Digital Signals
12	3.2 How to Solder
13	3.3 Basics of Computer Programming
14	3.3.1 The Programming Process
14	3.3.2 Programming Tools
16	4 Introduction to the S4 Payload
16	4.1 Base Payload Components
16	4.1.1 Flight Board
16	4.1.2 Arduino
17	4.1.3 Wi-Fly Shield
17	4.1.4 Open Log SD Writer
17	4.1.5 GPS Receiver
17	4.1.6 Duck Antenna
17	4.1.7 Logic Level Converter
17	4.1.8 Voltage Regulator
18	4.2 Sensor Components
18	4.2.1 Accelerometer
18	4.2.2 Magnetometer
18	4.2.3 Barometric Pressure/Temperature Sensor
18	4.2.4 Humidity/Temperature Sensor
19	5 Building and Testing the S4 Payload
19	5.1 Assembling the Flight Board
19	5.1.1 LEDs
20	5.1.2 Capacitors
20	5.1.3 Power Jack

20	5.1.4 Resistors
21	5.1.5 Making and Installing Male Header Pins
21	5.1.6 Logic Level Converter
22	5.1.7 Voltage Regulator
22	5.1.8 Female Header Pins
23	5.1.9 Arduino/WiFly Header Pins
23	5.1.10 Attach Stand-Offs
23	5.1.11 Complete Pre-Assembled Flight Board
24	5.1.12 Put Header Pins On Sensors
25	5.2 Powering The S4 Payload
25	5.2.1 Powering the Payload During Development
25	5.2.2 Powering the Payload During Field Testing
25	5.2.3 Powering the Payload During Flight
26	5.3 Testing The Assembled Flight Board
26	5.3.1 Barometric Pressure/Temperature Sensor
26	5.3.2 Accelerometer
27	5.3.3 Humidity/Temperature Sensor
27	5.3.4 Magnetometer
27	5.3.5 Open Log
28	5.3.6 Arduino/ WiFly Shield
28	5.3.7 GPS Connector
29	6 Programming the Arduino
29	6.1 Obtaining Arduino and S4 Software
29	6.2 Basic Arduino Programming Commands and Punctuation Marks
30	6.3 Communicating with the Arduino
30	6.4 Programming the Arduino to Collect Data
31	6.4.1 Writing a Sketch to Turn On and Off an LED Connected to the Arduino
32	6.4.2 Printing to the Serial Monitor
33	6.5 Coding and Testing Base Flight Board Sensors
33	6.5.1 GPS Receiver Code and Test
35	6.5.2 Open Log Code and Test
36	6.5.3 WiFly Code and Test
41	6.6 Coding and Testing Sensors
41	6.6.1 Accelerometer Code and Test
42	6.6.2 Magnetometer Code and Test
43	6.6.3 Barometric Pressure/Temperature Sensor Code and Test
44	6.6.4 Humidity/Temperature Sensor Code and Test
45	6.6.5 Testing All Sensors
46	7 Customizing the S4 Payload Design
46	7.1 Designing Experiments With the S4 Flight Board
46	7.1.1 Size, Mass and Power for the S4 Payload
46	7.1.2 Standard Measurements With the S4 Flight Board and Sensors
48	7.1.3 Other Possible Measurements for Advanced Students
48	7.1.4 Rocket or Balloon?

49	7.2 Rocket Payload Requirements
50	7.3 Balloon Payload Requirements
50	7.4 Flight Project Documentation Requirements
50	7.4.1 Design
51	7.4.2 Experimental Validation
51	7.4.3 Procedures
51	7.4.4 Flight Log
51	7.5 At the Launch Site
52	8 Experimental Data
52	8.1 Acquiring and Accessing Payload Data
52	8.1.1 Acquiring the Payload Data
52	8.1.2 Accessing the Data
53	8.2 Analyzing Payload Data
53	8.2.1 Displaying and Analyzing Payload Data (Excel)
54	8.2.2 Experimental Uncertainties
54	8.2.3 Mapping the Data (Google Earth)
55	8.2.4 Comparing Measurements to Predictions
56	8.3 Presenting Results
56	8.3.1 Lab Report
56	8.3.2 Presentation Poster
56	8.3.3 Presentation Talk
56	8.3.4. Beyond the Experiment
57	Appendix A – Standards Alignment Information
60	Appendix B – NASA-Funded Programs
61	Appendix C – References and Additional Resources
63	Appendix D – Glossary
67	Appendix E – Electrical Schematic for Flight Board
69	Appendix F – Preferred Parts List
71	Appendix G – Parts and Tools
72	Appendix H – Completed Labeled Flight Board



Introduction – About This Guide

Approximately 7000 American students build model rockets each year for Team America Rocketry Challenge competitions, typically launching eggs to altitudes around 500-700 feet. Secondary students also participate in local tethered weather balloon events, which can gently loft experiments to altitudes as high as 1000 feet. These two types of competitions offer great opportunities to develop student STEM skills by building, designing and analyzing data from scientific payloads. Most of these experiments and payloads, however, do not fly modern electronics, nor do they allow the ability to communicate with ground stations in real time. This educator guide has been designed to help you and your students to achieve the next level in payload design, data acquisition and analysis while building exciting projects that can be integrated into your curriculum, or used in informal educational settings. The Arduino-based flight boards that we have designed will allow your students to customize their experiments, and will help them to develop important and valuable skills in electronics, computer programming, data analysis, problem solving and project planning.

The **S4 project** is a partnership between the SSU Education and Public Outreach (E/PO) group, the Association of Experimental Rocketry of the Pacific (AeroPac) prefecture of the Tripoli Rocketry Association, and the Endeavour Institute. This guide, the flight board designs, and Arduino software have been developed at Sonoma State University (SSU) under the direction of Professor Lynn Cominsky through NASA grant NNX12AB97G. Additional major contributors to this project include SSU staff members Kevin John and Logan Hill, SSU student Kevin Zack, AeroPac members Ken Biba and Tony Alcocer, and the Endeavour Institute's Director Steve Klierer. We also acknowledge assistance from Rebecca Salvemini, Lauryn Loudermilk, and Jeffrey Camerino in testing the guide and payloads. Layout and illustrations in this guide are by Aurore Simonnet.

What's in the Guide?

This guide includes :

- an overview of the S4 project
- historical background information about rocketry and ballooning
- information about different classes of rocketry and engines;
- brief introductions to building and testing digital electronic circuits, soldering, and computer programming;
- a description of the functionality of each part used to build the payload;
- a step-by-step guide that illustrates how to build and test the payload using our flight board design and parts on our preferred parts list;
- instructions for programming the payload and the optional sensors;
- information about customizing the payload to perform experiments,
- a brief summary of how to access, analyze and present the experimental data.

Also included is information about Standards Alignment for middle and high-school curricula, a list of other NASA programs for you and your students, references and additional resources, a glossary, complete electrical schematics and ordering information for the preferred parts list.

Overview of the S4 Project

I. Introduction

Students are presented a lecture or assigned reading on the S4 project as well as writing up their planned experiment with the sensor(s) needed.

- Materials: S4 Guide
- Time: 1 - 3 hours

II. Constructing the Base

The basic components of the payload need to be soldered together either by the students in class or the educator beforehand.

- Materials & Instructions: See 3.2 How to Solder and 5.1 Assembling the Flight Board.
- Time: 2 - 5 hours

III. Programming

Students learn to write their sensors' computer code and upload the code to the payload.

- Materials & Instructions: See 3.3 Basics of Computer Programming and Chapter 6 Programming the Arduino
- Time: 1 - 5 hours

IV. Testing & Trouble Shooting

Students add their sensors to the base payload, test their payloads and make sure they are taking accurate data.

- Materials & Instructions: See 6.5 Coding and Testing Base Flight Board Sensors and 6.6 Coding and Testing Sensors
- Time: 1 - 2 hours

V. Launching the Experiment

Student acquire flight data, conducting their planned experiment with the payload

- Materials: See 7 Customizing the S4 Payload Design and 8.1 Acquiring and Accessing Payload data.
- Time: varies greatly. The launch time may essentially take no time if the students' payloads are delivered to other fliers and the data are accessible via the internet. Or it could take up to several days if the students are traveling to a remote launch site. For on-campus launch sites (balloons and/or rockets) it is estimated to take 1 - 3 hours depending on the total number of launches for the day.

VI. Data Analysis & Presentation

Students retrieve and analyze their data as well as present their experimental findings.

- Materials: See 8 Experimental Data.
- Time: 1 - 3 hours

Project Total Time: 6 - 18 hours + flight time.

1 Background Information - Rocketry



1.1 Historical Overview of Rocketry

One of the earliest descriptions of rocket-like devices comes from *The Life of Apollonius of Tyana*, by Philostratus, circa 200 CE, where legendary Greeks, in attempting to conquer “sages” living in India, are “driven off by rockets of fire” and thunderbolts which were hurled obliquely from above and fell upon their armor.”

The first known rockets were fireworks created by the Chinese for festivals and celebrations possibly as early as the first century of the Common Era. Yet, their first recorded use as a weapon was at the battle of Kai-keng in 1232. The Chinese fought the Mongols and used rocket-arrows to help repel the invaders. It only took a few decades for the Mongols to then introduce gunpowder to Western Europe where the weaponization really took off.

The English philosopher, Roger Bacon (1214 – 1294) was one of the first Europeans to write about gunpowder in his books. In the centuries that followed, gunpowder was used almost exclusively in warfare and it wasn’t until the 16th and 17th centuries that accounts of its use for explorative endeavors begin to slowly surface. In the 1500s a Chinese legend emerged that told the tale of Wan Hu, who attached many rockets to a special chair to launch himself to great heights. In one version of the legend he made it to the Moon.

In 1650, the Polish-Lithuanian nobleman Kazimierz Siemienowicz (c. 1600 – c. 1651) published *Artis Magnae Artilleriae pars prima* (Great Art of Artillery, the First Part); a guidebook on the uses of gunpowder that ended up being used by militaries across Europe. In it, Siemienowicz writes “I...thought myself obliged...to instruct you in the due and proper Methods of preparing [rockets]; to shew [sic] you their different Forms and Construction; and to display the particular Use they are of.”

Thirty-seven years later, Sir Isaac Newton (1642 – 1727) would publish his *Philosophiæ Naturalis Principia Mathematica* in which he outlined, among other things, the laws

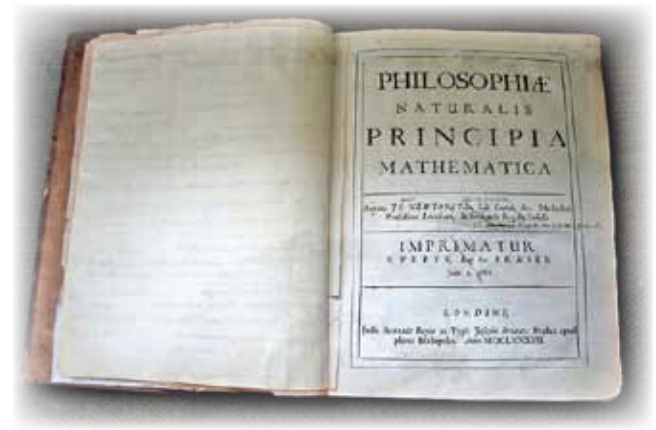


Figure 1.1 Newton's *Philosophiæ Naturalis Principia Mathematica*

of motion and gravitation. Built upon ideas put forth by Galileo Galilei (1564 – 1642), these natural laws would become fundamental concepts for rocketry.

Throughout the Enlightenment rocketry remained largely within the purview of the military. At the end of the 18th Century, Tipu Sultan's army of the Kingdom of Mysore, saw the first recorded to use rockets with iron fuselages during the Mysore War against the British. Captured rockets were sent back to England where Sir William Congreve (1742 – 1814) developed what became known as Congreve Rockets which, while still highly inaccurate, were able to withstand greater pressure due to the metal fuselage and therefore attain higher thrusts and greater distances.

One of the first recorded instances of the use of non-military payloads is by the Italian artist and rocket enthusiast, Claude Ruggieri (c. 1800) who reportedly was able to launch small animals in 1806 and by 1830 he successfully launched a sheep up to 600 feet and landed it safely via parachute.

At the end of the Enlightenment rockets entered literary history in the 1814 poem “Defense of Fort McHenry” by Francis Scott Key (1779 – 1843), which would go on to become the lyrics of United States of America's national anthem:

¹“rockets of fire” being a translation of the ancient Greek nominalization of the verb “to eject forcefully.”

Greek Legend	Chinese Fireworks	R. Bacon Gunpowder literature	Wan Hu chair launch legend	K. Siemienowicz Gunpowder guide book	Isaac Newton Laws of Motion
200 CE	1232	1240	1500	1650	1687

*O say can you see by the dawn's early light,
 What so proudly we hailed at the twilight's last gleaming,
 Whose broad stripes and bright stars through the perilous fight,
 O'er the ramparts we watched, were so gallantly streaming?
 And the rockets' red glare, the bombs bursting in air,
 Gave proof through the night that our flag was still there;
 O say does that star-spangled banner yet wave,
 O'er the land of the free and the home of the brave?*

The rockets that Key mentions in the poem were British fired Congreve rockets developed only a decade prior to their use against U.S. Fort McHenry.

In 1865 Jules Verne (1828 – 1905) wrote *From the Earth to the Moon*. The spacecraft of the story was not launched on a rocket, but instead by cannon. However, to land on the moon the craft used rockets to help in its descent.

At the turn of the last century, Konstantin Tsiolkovsky (1857 – 1935) independently derived the “ideal rocket equation” which is also called the Tsiolkovsky Rocket Equation. This equation allows for the calculation of a rockets change in velocity as its mass decreases. Tsiolkovsky is considered to be the founding father of rocketry.

Figure 1.2 Tsiolkovsky Rocket Equation

$$\Delta v = v_e \ln \frac{m_0}{m_1}$$

From Tsiolkovsky's work on rocketry both Hermann Oberth (1894 – 1989) and Robert H. Goddard (1882 – 1945) emerged as pioneers of rocketry. In 1922, Oberth published *Die Rakete zu den Planetenräumen* (By Rocket into Planetary Space) which would inspire the creation of the Verein für Raumschiffahrt (VfR, Society for Space Travel) in Germany.

Robert H. Goddard would be the first to create and launch liquid-fueled rockets. His first flight took place in 1926 in Auburn Massachusetts. Goddard would end up launching dozens of these rockets over the years until 1941.

Goddard was severely underfunded, especially in the early years of his design and testing. Meanwhile back in Germany, around 1930 a student named Wernher Von Braun (1912 – 1977) had joined the VfR. And despite the disbanding of the VfR, Von Braun had clearly shown great aptitude in understanding rocketry as he had worked with Hermann Oberth in some of Germany's first liquid-fuel tests. Von Braun is a controversial figure due to his membership in the Nazi party

and the fact that he worked and built rockets for the Third Reich during World War II. The German military was keenly interested in rocketry and all civilian rocketry programs were banned and/or subsumed into the military.

German scientists, including Von Braun, consulted and used Goddard's research in order to develop their own rockets. Before the war there was some correspondence on rocket science between Goddard and German scientists.

By the end of World War II, Von Braun had overseen the creation of liquid-fueled rockets, the V-2, that could reach London and Antwerp, reached heights of 80 km, and could be guided with rudimentary radio (wireless) signals.

Wernher Von Braun surrendered to the Allies and was brought to the United States after the war where he continued to work on rockets; first for the U.S. Army expanding upon the V-2 rockets from Fort Bliss, Texas and White Sands, New Mexico, then, during the 1950s, as a promoter of manned space flights that included promotion of the idea of manned travel to the Moon.

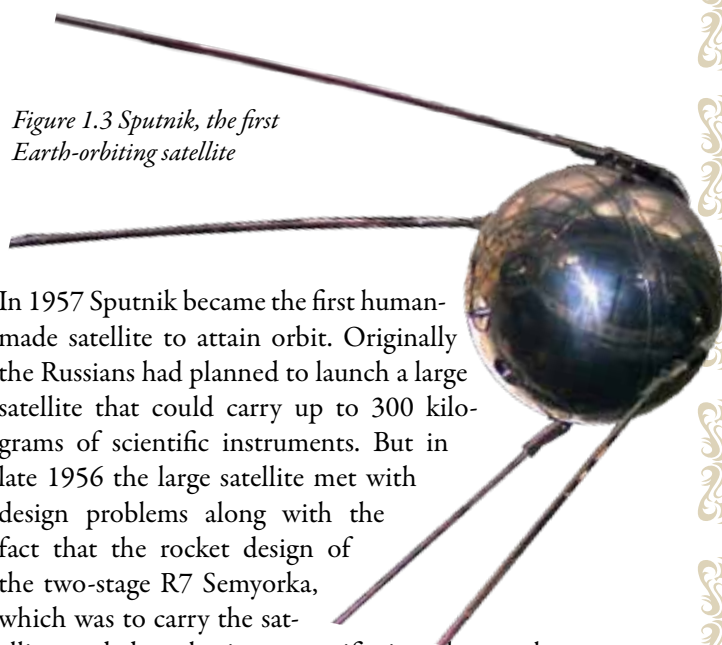


Figure 1.3 Sputnik, the first Earth-orbiting satellite

In 1957 Sputnik became the first human-made satellite to attain orbit. Originally the Russians had planned to launch a large satellite that could carry up to 300 kilograms of scientific instruments. But in late 1956 the large satellite met with design problems along with the fact that the rocket design of the two-stage R7 Semyorka, which was to carry the satellite, ended up having a specific impulse nearly 2% lower than originally projected. Because of these issues the Russians decided to switch from the original heavy and complex payload to the lighter, 84 kg., Sputnik.

It was also in 1957 that the U.S. National Association of Rocketry (NAR) was founded to encourage and support sport rocketry. Founded in 1957, NAR is the oldest and largest sport rocketry association in the world, with over

Congreve Rockets	F.S. Key Poem (National Anthem)	C. Ruggieri Sheep safe launch	J. Verne From Earth to the Moon	Tsiolkovsky Rocket Equation
1800	1814	1830	1865	1900



4500 members and 120 affiliated clubs across the U.S. In addition, NAR members conduct educational outreach activities that annually expose thousands of students to “rocket science” in a safe and exciting hands-on way

As the Space Race heightened, the United States established NASA (National Aeronautics & Space Administration) in 1958. Von Braun was transferred to the new institution where he was able to continue working on the Saturn rocket, one of the crowning achievements of the early NASA program.

By mid-year 1960, Von Braun became NASA’s first Director. The U.S.S.R. was also heavily invested in continuing the Space Race, and on April 12th, 1961, the cosmonaut Yuri Gagarin (1934 – 1968), became the first human being in space. He was launched from a next-generation Semyorka; the Vostok 8K72K rocket.

In 1964, the Tripoli Rocketry Association (TRA) was founded in Pittsburgh, Pennsylvania. Like NAR, TRA organizes and promotes sport and science-based rocketry.

1.2 Rocketry for the Secondary School Classroom

NASA has developed several rocketry guides for use in the secondary school classroom. Links to these resources are given in Appendix B. Although these guides and tutorials provide an excellent survey of rocket science and technology, they do not address the design, construction and analysis of data from a small scientific payload, which are the primary goals of this guide.

The projects described in this guide can be used to advance the skills of students that have already participated in model rocketry competitions such as Team America Rocketry Challenge (<http://www.rocketcontest.org>). TARC is a nation-wide program that uses student-designed model rockets, with limited altitudes and payload capabilities. Started in 2003 to celebrate the centennial of flight, TARC is currently supported by NASA, the Department of Defense, and the American Association of Physics Teachers, as well as by a multitude of industry sponsors and NAR. The annual TARC competitions encourage registrants nation-wide and typically draw 7000 students in approximately 600 teams from across the United States. The final TARC competition features the top 100 teams as judged by scores achieved in local competitions run by NAR, and takes place in Virginia. A typical TARC competition objective (e.g., the 2011 contest) is to design and build a safe and stable model rocket flight vehicle and use it to lift a fragile payload (one raw hen's egg) to an altitude of exactly 750 feet, with a total flight duration between 40 and 45 seconds, then return this payload safely and undamaged using a 15-inch parachute.

1.3 Model Rocketry vs. High Powered Rocketry

Table 1 defines the range of total impulse levels for the different classes of rocket motors that will be discussed in this guide. (Each subsequent letter increases the thrust range by a factor of two.) Commercially available motors for model rocketry (as small as ¼ A up to class G) can be purchased at local hobby stores and/or online: many localities allow model rocket launches without special clearances. However, for larger motors (class H or higher), one must be certified by one of the national rocketry organizations such as NAR or TRA. Certification then enables individuals to purchase high-powered motors for launches at sanctioned events. Such high-powered rocketry (HPR) events require clearance (called “waivers”) from the Federal Aviation Administration, and there are many complex safety requirements for conducting HPR launches. We therefore recom-

The Apollo program was a NASA program with the goal of placing humans on the moon. Using the Saturn V rocket, NASA put the first humans, Neil Armstrong (1930 -2012) and Buzz Aldrin (b. 1930), on the moon in 1969 (Michael Collins [b.1930] remained in the orbiting command module as pilot) with the Apollo 11 mission. This was the first time human beings had been to any non-terrestrial world in person, and is considered one of the crowning achievements of humanity. Since that time, NASA has continued to launch humans and satellites into orbit for scientific endeavors. There is also a huge commercial satellite presence, with over 3000 working satellites still orbiting the Earth, and probably four times that many that have ceased operations or have re-entered the atmosphere.

Figure 1.4 NASA Taurus rocket



Goddard & Von Braun Liquid fueled Rocket	W. Von Braun promote manned space flights	Sputnik Russian Satellite	NASA established	Vostok 8K72K rocket 1st man in space	Saturn V rocket starts Apollo Program
1941	1950	1957	1958	1961	1969

Table 1: Definitions of Impulse Classes for Commercial Rocket Motors

Class	Total Impulse (Metric Standard)	
A	1.26-2.5 N·s	Non-certified
B	2.51-5.0 N·s	
C	5.01-10.0 N·s	
D	10.01-20.0 N·s	
E	20.01-40.0 N·s	
F	40.01-80.0 N·s	
G	80.01-160.0 N·s	
H	160.01-320 N·s	HPR certification required
I	320.01-640 N·s	
J	640.01-1,280 N·s	
K	1,280.01-2,560 N·s	

mend that any educators who are considering flying student payloads on HPRs contact your local NAR or TRA chapter (see Appendix C), and partner with experienced HPR flyers. There are NAR and TRA chapters world-wide, and members are enthusiastic about sharing their knowledge and love of rocketry with students of all ages.

Commercial rocket motors are further described by a code similar to the following: J350-6W. In this example, the initial letter (J) indicates the total impulse range of the rocket (320.1 – 1280 N·sec), the number after the letter indicates average thrust (e.g., 350 N), the number after the dash (6) indicates the delay between when the motor fuel burns out and the ejection charge is ignited (assumes the motor kit comes with a delay grain, which is burned after the initial motor fuel.); and the additional letter at the end (W) (if present) indicates the color of smoke produced by the motor (e.g., white). Vendors often use variations of this code, but the base letter-number format is typically consistent. Figure 1.5 shows the thrust profile of an AeroTech J350 motor. The area under the curve integrates to a total impulse in the J range, and the total impulse divided by the average thrust gives you a rough indication of how long the main motor fuel will burn. Information on rocket competitions and organizations is provided in Appendix C.

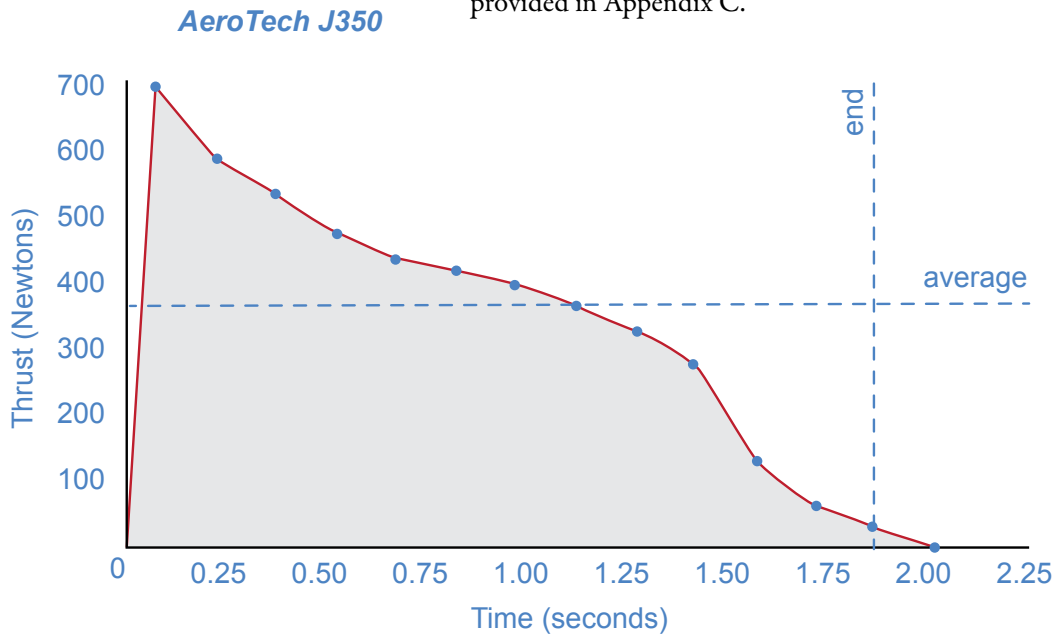


Figure 1.5: The thrust (force in N) vs. time for the J350 motor made by AeroTech. The average thrust is about 350N, and the total impulse is about 700 N·sec.



Figure 1.6 Looking up to the future - SpaceX Falcon 9

2 Background Information - Ballooning

2.1 Historical Overview of Ballooning

In about 200 CE, the Chinese developed the flying lantern: paper lanterns that ascend into the sky like miniature hot-air balloons, for use as military signals at first and then as celebratory decorations still in use today.



Figure 2.1 *Bellifortis*

It is thought that lighter than air objects akin to the flying lanterns were also making their way into Europe through Mongolian influence. In the 1405 manuscript, *Bellifortis* by Konrad Kyeser, descriptions are given of fire breathing dragons fueled by oil-lamps that are used by warriors on horseback

While it can easily be debated as to whether the image shows a kite, stylized banner-flag, or some other non-lighter-than-air object, Kyeser's description of their use of oil-lamps, the one-handed delicate use of the object by the horseman in the image, and the era in which he is writing make it plausible that more than just gunpowder had made its way from China to Europe.

The year 1783 proved to be a monumental year for balloon flight. The Montgolfier brothers (Joseph 1740 – 1810, and Jacques 1745 – 1799) realized that small bags, when held over a fire, would float in the air. Expanding to bigger and bigger experiments, they built a silk and paper balloon that, on June 4th, 1783, rose to over six thousand feet. Their balloon design was named after the brothers.

Less than 4 months later, in September the Montgolfier brothers flew various animals for Louis XVI and Marie Antionette. Then in November, the first manned flight occurred as Pilatre de Rozier (1754 – 1785) and Marquis d'Arlandes (1742 – 1809) took to the air in an untethered Montgolfier balloon for which Benjamin Franklin was in attendance.

Jacques Charles (1746 – 1823) and Nicolas Robert (1760 – 1820) then made the first hydrogen balloon launch on December 1st, 1783 with Charles providing the hydrogen (having taken five days to fill the balloon) and Robert supplying the rubber coated silk. From 1783 onward, ballooning would develop, expand, and see myriad innovations including renowned scientist, Michael Faraday's (1791 – 1867) inventing of the modern rubber balloon. Throughout the 1800s, ballooning continued, especially with manned vehicles for pleasure and leisure. The century also saw the development of the steerable and independently powered lighter-than-air craft.

During 1911-1913, an important physics experiment was conducted using balloon-based measurements: Austrian Physicist Victor Hess took electroscopes aloft in a balloon to accurately measure the levels of radiation as a function of altitude. He made repeated and systematic measurements at great risk to his own health, as he accompanied his instrumentation to altitudes over 5 km. The increased radiation levels at high altitudes that he measured was the first confirmed evidence for the existence of charged particles arising from outside the Earth's atmosphere; these particles are now known as cosmic rays. For his work, Hess was awarded the 1936 Nobel Prize in Physics.

During World War I, hydrogen-filled tethered balloons were used to conduct surveillance operations at altitudes over 3000 feet. As hydrogen is flammable, many hundreds of balloons were lost and passengers in the balloon gondolas were often forced to escape by parachute. After World War I, most balloons have been filled with the much safer gas, Helium. Balloon use for spotting submarines was important during World War II. After the war, balloon use increased as balloons were used by various government agencies as well as companies like General Mills in the late 1940s and 1950s.

In 1960 the National Center for Atmospheric Research (NCAR) was established which ushered in the era of high altitude balloons carrying scientific payloads during the late 1960s and early 1970s. From the 1970s onward, scientific high altitude ballooning has continued to provide

Chinese floating lanterns	Bellifortis K. Kyeser	Isaac Newton Laws of Motion	Montgolfier Brothers' balloon flights	1st Steerable Balloon Faraday's rubber balloons
200 CE	1405	1687	1783	1824 1852

important data regarding our planet and the universe in which it spins. As materials, instrument packages, and power supplies have improved so has the success rate of balloon missions. In 1981, a mission from Greenville, South Carolina, flew for over 40 hours, showing that extended duration flights were possible.

In the 1990s and 2000s, many balloon flight experiments were dedicated to studying the cosmic microwave background, most notably BOOMERanG. Balloons that study the microwave background are often lofted from Antarctica, to provide a quiet background signal against which tiny fluctuations in the cosmological background temperature can be more easily measured.



Figure 2.2 High altitude balloon

The National Balloon Launching Facility (NBLF) maintains launch sites in Palestine, Texas, and at Ft. Sumner, New Mexico for scientific use. Helium-filled balloons routinely loft scientific experiments to over 100,000 feet from these locations. Today, standard NASA scientific balloons are constructed of very thin polyethylene film, similar to that used for plastic bags. The balloons are open to the atmosphere at the bottom to equalize the internal pressure with the surroundings. These very large balloons can carry a payload weighing as much as 3,600 kilograms up to altitudes over 40 kilometers, and can stay there for up to two weeks. Ultra-long duration balloons of a different design are being used for flights of up to 100 days, with payload retrieval by parachute once the experiments are completed.

2.2 Ballooning for the Secondary School Classroom

At least once a year, the Endeavour Institute organizes a “Balloon Fest” in which teachers and teams of three to six students are invited to launch helium-filled, tethered balloons with student-designed instrumentation. The students work for weeks to design their experiments, which measure different characteristics of the atmosphere, or test alternative engineering designs to drop eggs or obtain video. Launches typically begin in the early morning hours, followed by analysis and presentation of the data. In contrast to high-powered rocketry, balloon launches are easy to stage, and have much less stringent safety regulations. If your balloon launch site is located more than five miles from any airports and is clear of hazards (such as power lines), and your balloon is filled with less than 120 cu. ft. of Helium, then it can be tethered up to 1000 feet without FAA approval.

The NASA/Wallops Balloon Experience for Educators (WBEE) program encourages high school teachers to learn how to build payloads for high-altitude balloon flights. WBEE uses the platform developed for the High Altitude Student Project (HASP) on balloons



Figures 2.3 and 2.4: Educators work with student teams to loft tethered weather balloons at Balloon Fest 2013.

which loft to altitudes of up to 116,000 feet. Launches occur in Ft. Sumner, New Mexico. Since 2004 HASP has flown payloads built by undergraduate students from across the United States, and this opportunity is now being extended to high school teachers and students.

Hess' electrosopes		WWI Military Reconnaissance	WWII Military Reconnaissance		NCAR established	Scientific Balloon boom
1911	1914	1918	1939	1945	1960	1970 - Present

3 Background Information

Electronics, Soldering and Computer Programming

3.1 Basics of Electronic Circuits and Measurements

3.1.1 Voltage, Current and Resistance

In order to build and test the S4 payload, it is important to have a basic understanding of the flow of electric currents over wires through systems of resistors, capacitors, transistors and other payload components. This will help you effectively integrate the various many components that comprise the payload.

The most basic properties one can measure in an electric circuit are voltage (V), resistance (R) and current (I). When learning about these properties, it's often useful to employ an analogy of water moving through a pipe. This analogy is illustrated in Figure 3.1 below.

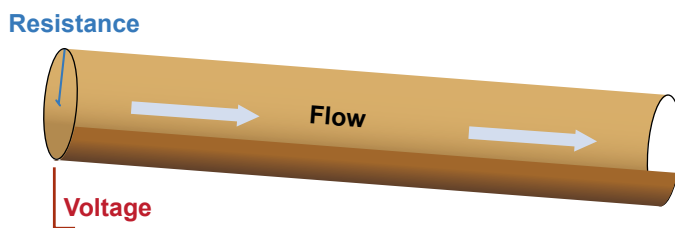


Figure 3.1 The pipe analogy for voltage, resistance and current

The amount of water flowing through the pipe is analogous to the current flowing through the wire. The difference in height between one end of the pipe and the other is analogous to the voltage, and without a difference in height, no water will flow. The radius of the pipe is analogous to the resistance, with a larger pipe having lower resistance. The lower the resistance, the larger the pipe, and the more easily water can flow through it.

Current (I):

Current is a measure of the flow of electrons through the circuit. A large current means that many electrons are flowing through the circuit, and a small current means that few electrons are flowing through the circuit. Larger currents provide greater amounts of energy that can be used to work with the circuit electronics. Current is measured in the unit of Amperes or 'Amps.' In the S4 payload, typical values for the current will be read out in milliamps (mA); 1 mA is equal to 1/1000 of 1 Amp.

Voltage (V):

Voltage is a measure of the difference in electrical potential energy between two points in a circuit. In the presence of a potential difference, electrons will flow and a current will be present. Larger potential differences generate higher currents. Voltage is measured in the unit of Volts.

Batteries:

A battery is a device which can turn chemical energy into a potential difference (voltage) which then drives an electric current. The S4 payload is powered by a single 9V battery, and on the flight board, we use two different voltage levels: 3.3 and 5 V.

Resistance (R):

Electrical resistance is a property of a material that dissipates current. Lower resistances allow greater current flow, while higher resistances impede the flow of current. A material with very low resistance is called a 'conductor,' while a material with very high resistance is called an 'insulator.' Resistance is measured in the unit of Ohms (abbreviated by the Greek letter Omega, Ω). Typical resistances in the S4 payload range from tens to thousands of Ohms.

Identifying Resistors using the Color Code:

In order to determine the value of a resistor, engineers use a color code. Engineers do this instead of writing the value of the resistor on the device in regular text because resistors can be very small, and reading and writing on such small devices can be difficult. Each resistor will have at least 3 color bands around it. Each color corresponds to a number, and these numbers get plugged into an equation as shown in Figure 3.2.

Color	1 st number	2 nd number	3 rd number	Multiplier
Black	0	0	0	1
Brown	1	1	1	10Ω
Red	2	2	2	100Ω
Orange	3	3	3	1,000Ω
Yellow	4	4	4	10,000Ω
Green	5	5	5	100,000Ω
Blue	6	6	6	1,000,000Ω
Violet	7	7	7	10,000,000Ω
Grey	8	8	8	
White	9	9	9	

Example: Let's identify the resistance of the resistor in Figure 3.3:
Starting on the left, we have a brown band which represents the number 1.

$$X = 1$$

Next we have a black band which represents 0.

$$Y = 10$$

Finally, the third band, the multiplier, is orange indicating that we must multiply what we have by 1,000.

$$Z = 10 \times 1,000$$

This resistor has a value of 10,000 Ω which can also be expressed as 10 kΩ.

3.1.2 Combining Components in Circuits

The relationship between Voltage, Current and Resistance is defined by Ohm's Law which states that the voltage (V) in a circuit is directly proportional to the electric current (I) and the resistance (R), or $V = IR$

Since most circuits contain more than a single component, there are additional rules that define how these components add together in different combinations. There are two basic architectures for combining circuit components: series, and parallel.

Adding Resistors in Series:

With resistors "in series" as shown in Figure 3.4, the same current flows through each in turn. The current, therefore, experiences a total resistance that is the sum of the individual resistor values.

To find the total resistance we add the resistance from individual resistors:

$$R_1 + R_2 + R_3 = R_{\text{total}}$$

Adding Resistors in Parallel:

When the current is divided and flows through each resistor simultaneously as shown in Figure 3.5, the resistors are said to be 'in parallel.' In this case, the size of the "pipe" that the current flows through has increased compared to that for a single resistor; therefore the total resistance has decreased.

To find the (inverse of the) total resistance for parallel resistors, the inverse resistance values are summed:

$$1/R_1 + 1/R_2 + 1/R_3 = 1/R_{\text{total}}$$

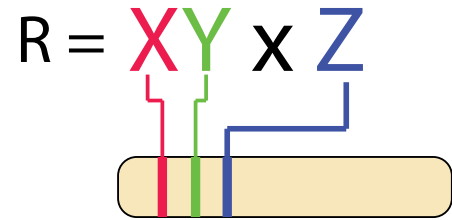


Figure 3.2: The resistor color code

The first two bands, X and Y, represent the first two digits of the resistance, and they are multiplied by the third band, Z, to get the total resistance of the resistor. You may notice that some resistors have a fourth and occasionally a fifth band. If there is a gold or silver band at one end of the resistor, then start at the other end to read its bands. Gold and silver can be either a multiplier band, a tolerance band, or both. They will never be at the beginning of a resistor's bands.



Figure 3.3 A sample resistor with colored bands

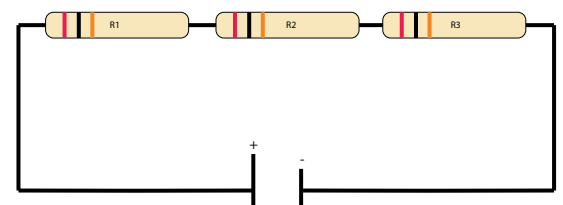


Figure 3.4: Three resistors in a circuit in series with a battery

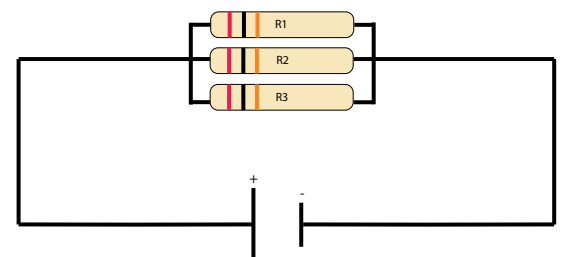


Figure 3.5: Three resistors in a circuit in parallel with a battery

Capacitance:

Capacitance is the ability of an electronic component to store electrical charge. Any component that is capable of storing charge is called a capacitor. Capacitance is measured in the unit of Farads (F). A capacitance of 1 F results from a current of 1 A flowing for 1 second through a potential difference of 1 V across the capacitor. This is an extremely large unit, so capacitances in the S4 payload are typically measured in microFarads (μF), where 1 μF is equal to 10^{-6} F.

Capacitors can be combined together into circuits in much the same way that resistors can, however the rules that define how their capacitance adds over series vs. parallel configurations is opposite that of resistance.

Adding Capacitors in Parallel:

For capacitors in parallel, the potential difference (voltage) across each capacitor in the circuit is the same, thereby increasing the total ability of the circuit element to store charge. Parallel capacitors are shown in Figure 3.6.

The total capacitance is therefore found by adding the individual values of the capacitance:

$$C_1 + C_2 + C_3 = C_{\text{total}}$$

Adding Capacitors in Series:

For capacitors in series, the same current flows through all the capacitors. Each capacitor stores the same amount of charge, regardless of its capacitance rating. This is because the charge on the connected sides of two adjacent capacitors must be equal. Capacitors in series are shown in Figure 3.7.

With this arrangement, the overall capacitance of the circuit is lowered, and the (inverse of the) total capacitance is found by adding the inverse of the individual capacitances.

$$1/C_1 + 1/C_2 + 1/C_3 = 1/C_{\text{total}}$$

3.1.3 Measuring Resistance, Voltage and Current in a Circuit

In order to measure voltage, resistance and current values in a circuit, you will use a digital multi-meter, sometimes called a “VOM” (or Volt-Ohm) meter. These meters have two probes (usually red and black), and different settings and input ports that are used to for measure the different electrical properties.

Measuring Resistance:

1. Make sure the black lead is plugged in the port labeled “COM”
2. Make sure the red lead is plugged into the port labeled $V\Omega\text{mA}$
3. Turn the dial to a reading that is labeled “ Ω ”.
4. Attach the probes of your multi-meter across the component, as shown in Figure 3.8.

Measuring Voltage:

To measure the potential difference (sometimes called “voltage drop”) across a circuit component:

1. Make sure the black lead is plugged in the port labeled “COM”
2. Make sure the red lead is plugged into the port labeled $V\Omega\text{mA}$
3. Turn the dial to a reading that is labeled “DCV”. Note: Meters usually measure two different types of voltage: direct and alternating. The direct voltage will either have a bar (and not a wavy line) on top of it, or will be labeled DCV. If there is a label that says “ACV” or that has a wavy line on top of it, this is NOT the setting to use. Also note: there may be more than one setting for direct voltage. Choose the range that is closest to the voltage that you expect to measure.

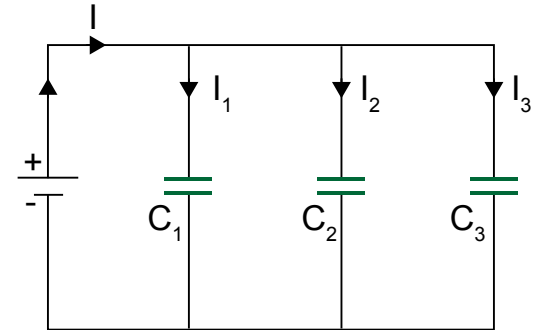


Figure 3.6: Three capacitors in a circuit in parallel with a battery

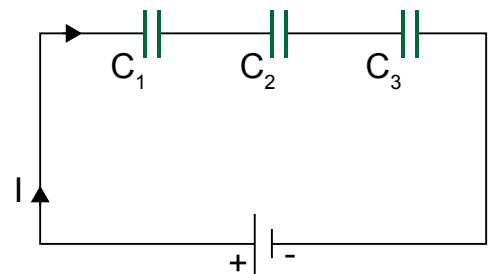


Figure 3.7: Three capacitors in a circuit in series with a battery

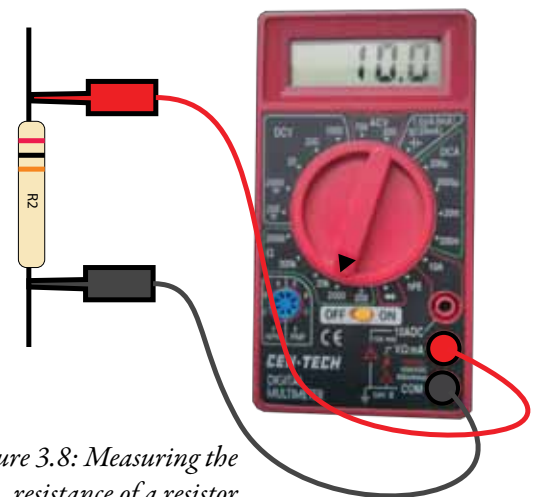


Figure 3.8: Measuring the resistance of a resistor

4. Attach the probes of your multi-meter in parallel with the component, as shown in Figure 3.9 and read the value output on the meter's display screen.

Measuring Current:

To measure the current that is flowing through a specific location in a circuit:

1. Make sure the black lead is plugged in the port labeled "COM"
2. Make sure the red lead is plugged into the port labeled "A" or "mA" (depending on how much current you expect. If you are not sure, use "A" first, but be aware that "A" is a very large current value.)
3. Turn the dial to a reading that is labeled "DCA."
4. Attach the probes of your multimeter as shown in Figure 3.10. Notice that the circuit is broken so that current flows through the multimeter in series with the circuit.

WARNING: The multimeter should never be left in "Current mode". In this mode it can accidentally create a dead short and blow out the meter. This might happen if, for example, you try to measure voltage while set to "current mode".

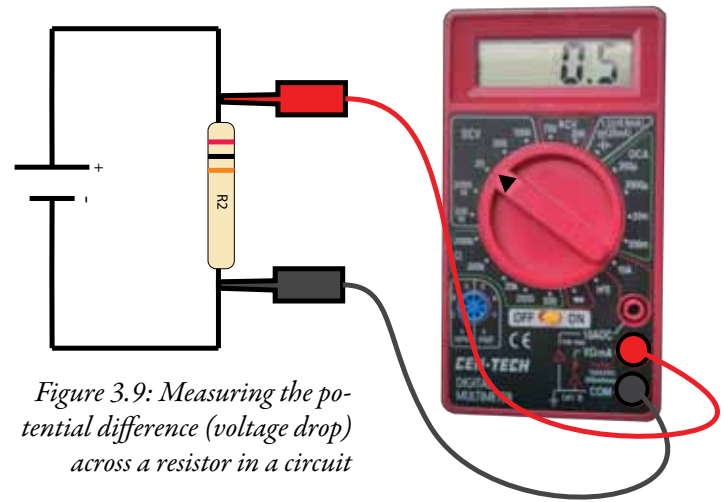


Figure 3.9: Measuring the potential difference (voltage drop) across a resistor in a circuit

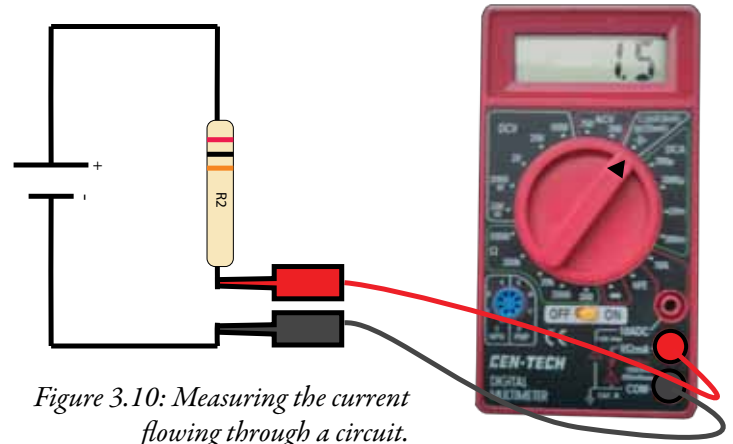


Figure 3.10: Measuring the current flowing through a circuit.

3.1.4 Analog and Digital Signals

The S4 payload includes some components (such as resistors and capacitors) that are usually used in analog circuits: these are circuits that have continuous, usually smoothly varying values for voltage and current. However, most of the components in the S4 payload have digital readouts: for these components, only two values of voltage are measured. Voltage values are near the ground level or 0 volts are interpreted as the digital bit "0" and higher voltage values, such as 3.3 V, are interpreted as the digital bit "1".

3.2 How to Solder

To build the S4 payload, either you or your students will need to solder some parts (a few components and some "header pins") to the main payload board. High school students should be able to do this soldering with proper supervision. However, you may want to solder the components onto flight boards for younger students. In either case, you will need the following tools, which are shown in Figure 3.11:

- A pair of safety goggles
- Electrical Tape
- Helping hands (something to hold the board still while you are soldering). You can also use vises or clamps
- Solder wire
- Brass sponge
- A soldering iron
- Regular Pliers
- Needle nose pliers
- Small flat head screwdriver
- Wire cutters

Soldering is *dangerous*. The iron and solder can reach temperatures in excess of 300°F. Only experienced or supervised students should be allowed to solder.



Figure 3.11: Soldering tools

When soldering, remember to always wear safety goggles. Additionally, make sure you've read the user's manual for your soldering iron for specific information related to its use.

Some soldering irons will need to be tinned before their first use. To tin a new tip:

1. Use alcohol to clean all the oils off the new tip.
2. Do not heat the tip until ready to tin.
3. Apply fresh solder and flux as soon as the tip gets hot.
4. Quickly and lightly swipe the tip on a damp sponge to spread solder and to refresh the tip periodically while in use.

Assuming you have a tinned soldering iron, turn it on so that it is hot enough when you wish to begin soldering.

1. Take the soldering wire and unravel about 6-10 inches of it so that you have plenty of slack.
2. Take the soldering iron out of its holder and touch the tip of it on the brass sponge to be sure it is clean and ready.
3. Hold the soldering iron with one hand up against the side of the lead as close to the board as you can. You will want to hold it at an angle as shown in Figure 12.
4. On the other side of the lead you will hold the tip of the soldering wire. As you do this, you will see the wire starting to melt and it will form a small pyramid shape of solder around your lead. As soon as you see this happen, remove the wire first and then the iron.
5. Place the iron back in its holder and look closely to be sure you've made a good connection.
6. If the instructions say to cut the leads, then use the wire cutters to clip the leads as close to the board as possible.

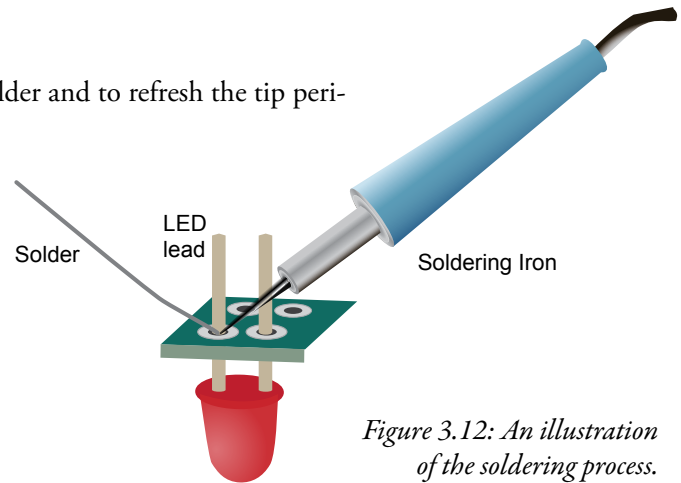


Figure 3.12: An illustration of the soldering process.

3.3 Basics of Computer Programming

Computer programming is the method by which the computer is given a set of instructions that tell it what to do. You can probably think of many examples of different types of computer programs, e.g., an internet browser, a word processor, or a video game. The common element connecting these different types of software is a set of instructions that the computer follows to make each program do its specific functions. This process may seem esoteric, or even mystical at first; but once you understand how to program, you'll find no end to the kinds of tasks you can accomplish with just a few simple tools.

Before you start to learn about programming, it might be useful to illustrate some of the meaningful differences between asking a computer to perform a task and asking a person to perform a task. Our brains are very good at inferring missing details; for instance you might ask a group of students to "finish up," and because of the circumstances (what class it is, the last assignment given, etc.) they will be able to understand, or at least have a good idea, as to what is being asked of them whether it is to finish cleaning their workspace or finish the quiz because the time is almost over. However, for a computer program, precise instructions must be clearly defined by the programmer. The computer is not able to infer meaning and cannot fill in gaps in the information that it uses to execute programs.

There are many different programming languages and a discussion of the advantages and disadvantages of each are outside the scope of this guide. In the S4 payload, the computer that is used is a microprocessor chip known as an Arduino. The Arduino microprocessor uses a programming language known as "Processing" (similar to the "C" language) to read the sensors and other payload components that generate and provide data in your rocket or balloon experiment.

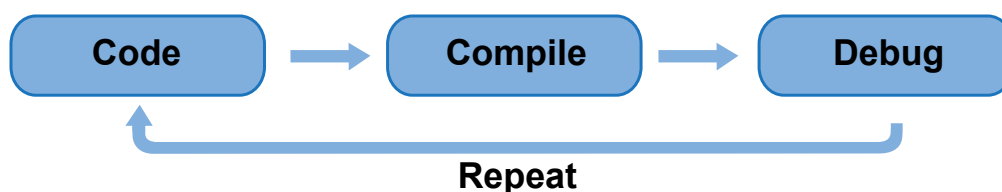


Figure 3.13: The iterative process used in developing a computer program

The process of writing a computer program is iterative; this means that we repeat a simple set of steps until we get the result that we want. For this project, the programming steps are: code, compile, and debug. These steps are then repeated until the program functions correctly, as illustrated in Figure 3.13 below.

If you've ever seen a computer programmer at work, either in real life or even in the movies; you'll typically see them writing some text with various characters and punctuation. Maybe something that looks a little like Figure 3.14.

Compile

In order for a computer to execute instructions (such as those shown in the figure above), they must first be translated from human readable source code into ‘machine readable’ code. The process that does this translation is called *compiling*. Compiled code looks something like the (hexadecimal) digits shown in Figure 3.15.

The translation process is usually performed by a program called the ‘compiler.’ As long as the source code has been written properly, it will compile into machine instructions when asked to do so. If there are errors in the grammar, logic or syntax of the source code, error messages will be generated, the compiling process will be interrupted, and the programmer will need to go back to the source code to find and fix the errors and then try the compilation step again.

Once our program has been coded and compiled, it is executed to test that it functions correctly. This is often the most time consuming part of the programming process. It is relatively easy to write source code that will compile and execute, but that does not insure that the program will function correctly. For example, the output may not be in the desired format, the data may not read out properly, or the entire system may not function. Any type of error in functionality will require the programmer to repeat the steps in this process until the system functions optimally. This may take many iterations of the process to achieve.

Some of the most basic building blocks of any program are variables. Variables are the program's way to reserve locations in the computer's memory into which values can be stored for later use. The concept of variables in computer programs is similar to that of variables in mathematics. In programming, however, variables usually have to be defined with a certain 'type'. This tells the computer how much memory to allocate for this variable. For example, a large number, like the longitude in a GPS location, needs more memory space than a smaller number, like an integer.

A function is a piece of code that accomplishes a single task. In mathematics, a function can have one or more mathematical inputs, to produce a single output number. In programming, the inputs and outputs of a function are not only limited

```

1
2
3 int main(void)
4 {
5     int n;
6     int i;
7     int current;
8     int next;
9     int twoaway;
10    printf("How many Fibonacci numbers do you want to compute? ");
11    scanf("%d", &n);
12    if (n<=0)
13        printf("The number should be positive.\n");
14    else
15    {
16        printf("\n\n\tI \t Fibonacci(I) \n\n");
17        next = current = i;
18        for (i=1; i<=n; i++)
19        {
20            printf("\t%d \t %d\n", i, current);
21            twoaway = current+next;
22            current = next;
23            next = twoaway;
24        }
25    }

```

Figure 3.14: Example of some human readable source code

8802	0000	5802	0000	2402	0000	a800	0000
1c01	0000	a001	0000	2c00	0000	0100	0000
006f	e20a	a56b	cd01	6072	0000	a800	0000
3100	0000	4d00	6900	6300	7200	6f00	7300
6f00	6600	7400	2e00	5600	6900	7300	7500
6100	6c00	5300	7400	7500	6400	6900	6f00
2e00	5400	6f00	6f00	6c00	7300	2e00	4f00
6600	6600	6900	6300	6500	2e00	4400	6500
6200	7500	6700	6700	6500	7200	2e00	6400
6c00	6c00	0000	0200	0100	0000	000a	c9b1
a96b	cd01	503e	0300	1c01	0000	2e00	0000
4d00	6900	6300	7200	6f00	7300	6f00	6600
7400	2e00	5600	6900	7300	7500	6100	6c00
5300	7400	7500	6400	6900	6f00	2e00	4400
6900	6100	6700	6e00	6f00	7300	7400	6900
6300	7300	2e00	4300	6f00	6d00	6d00	6f00
6e00	2e00	6400	6c00	6c00	0000	0100	0000
000a	c9b1	a96b	cd01	7802	0300	a001	0000
3600	0000	4d00	6900	6300	7200	6f00	7300
6f00	6600	7400	2e00	5600	6900	7300	7500
6100	6c00	5300	7400	7500	6400	6900	6f00
2e00	4400	6500	6200	7500	6700	6700	6500
7200	2e00	5700	7000	6600	5400	7200	6500
6500	5600	6900	7300	7500	6100	6c00	6900
7e00	5500	7300	2e00	6400	6e00	6e00	0000

Figure 3.15: An example of (hexadecimal) machine code

to numbers. You may put in a number and get out a text string. You may put in a text string and get back nothing (but some meaningful process happens inside the function). Even the largest and most complicated programs are constructed from the simple machinery of functions.

Integrated Development Environment (IDE)

An Integrated Development Environment (IDE) is a software package that incorporates all of the necessary tools to help you develop your software. To program the S4 payload, you will be using the Arduino IDE (see Chapter 6). Typical IDE tools include:

Code Editor

It's entirely possible to construct even the most complicated program using nothing but a simple text editor like Notepad or TextEdit. However, most programmers recognize the benefit of writing their code using a text editor that is specifically designed for the task of coding. Certain features like syntax highlighting, which color codes your text to make key words stand out, and auto-completion, which makes suggestions on what it thinks you are trying to write, can make the process of writing your code much faster, more reliable, and more efficient.

Compiler

The compiler is the program that turns your source code into machine code. For the most part, you won't need to dig too deeply into how this program functions; but you'll be using it almost constantly as you build your programs and it's important to know that it's there.

Debugger

After building your program, the debugger listens to your program and detects any errors that might occur. It presents these errors to you, and provides useful information about what caused the errors that will help you in correcting them.

Libraries

Not every piece of code needs to be written from scratch. Many times it is advantageous to let the programmers who have tackled a given program write the code before you do the work. This is where the concept of importing libraries comes into play. Let's imagine that during the construction of some program we need to take the sine of some angle. Of course, the computer doesn't intrinsically understand what the sine function means. We could explain it to the computer by expanding out a complicated numerical expression, however, a much simpler option would be to use an existing piece of code that already does this. We simply import a Math library that includes the sine function, and use the existing code to save ourselves a lot of time and trouble. For the S4 payload, we have developed libraries that include the software needed to read data from various components. You will learn how to use these in Chapter 6.

4 Introduction to the S4 Payload

The S4 payload was designed mostly from ‘off the shelf’ components available online and/or at various electronics retailers. Its design emphasizes a simple, reliable, and flexible platform that students can use without needing to learn how to solder; and with a size and weight appropriate for flying on either a high powered rocket or balloon gondola. The payload has two classes of components: base components and sensor components. The base components are required in order for the payload to complete its most basic functions and must be present on all payloads. The sensor components are flexible; your students will be free to select sensors from the preferred parts list (Appendix F) that meet the needs of a particular experiment or to use any other available sensors if they wish to write their own programs to read the output data. The following section describes the components on the preferred parts list.

4.1 Base Payload Components

The base payload components include the flight (printed circuit) board, the Arduino microprocessor, Wi-Fly shield, Open Log SD Writer, GPS receiver, logic level converter, voltage regulator, and 2.4 GHz Duck antenna.

4.1.1 Flight Board

The flight board is a Printed Circuit Board (PCB) that functions as the backbone of the entire payload. It provides the physical superstructure that keeps all the components firmly in place. It also provides pathways for both power and information to travel between components; this is accomplished through a system of wires printed right onto the board called ‘traces.’ The flight board is the only component in the S4 Payload that is not off the shelf. The S4 team designed these boards, and they were manufactured specifically for this project. The boards may require additional assembly by the instructor before they can be given to (younger) students. Figure 4.1 shows the flight board before the additional assembly described in Chapter 5.

4.1.2 Arduino

The Arduino is the main microprocessor. It is responsible for collecting data from the various components, formatting that data, and sending it to the Wi-Fi and SD card devices for transmission and storage. All the code that operates the payload will be executed by the Arduino. It contains a USB port for communicating with your computer and receiving new programming. It has many pins for communicating with other devices, and when it is attached to the flight board, these pins are routed to the proper payload components. When attached to the flight board, it will also receive power from the board’s power supply. Figure 4.2 shows the Arduino Uno-R3 chosen for this project.

The Arduino Uno has 14 digital input/output pins, 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It can be easily connected to a computer with a USB cable or powered it with either a AC-to-DC adapter or battery.

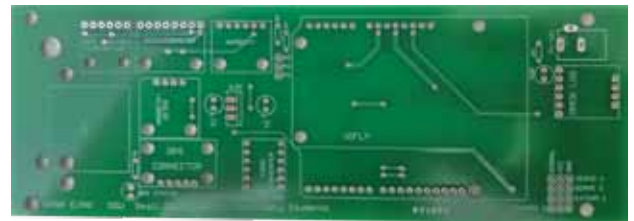


Figure 4.1: The top figure shows the top side of the board, and the bottom figure shows the reverse side of the board.

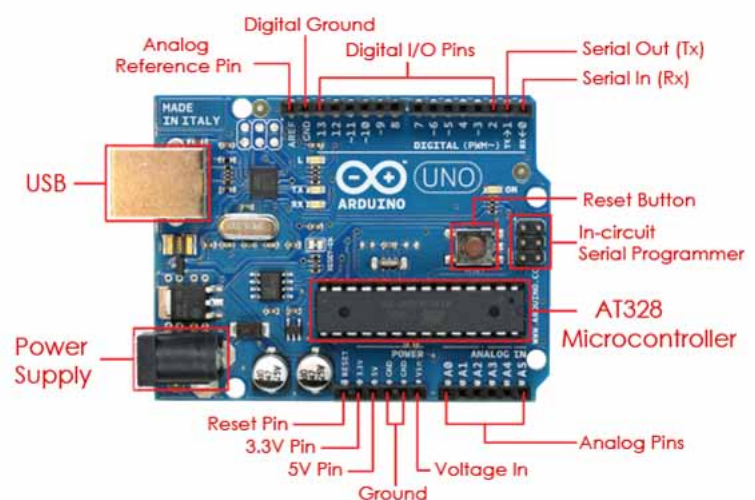


Figure 4.2: The Arduino Uno-R3

4.1.3 Wi-Fly Shield

The Wi-Fly chip allows the payload to communicate in real time with a remote base station using 802.11g wireless network protocol. This is the same type of wireless communication you might use at home or in the classroom to provide internet access to your computers or smart phones. This chip provides two-way communication between the ground station and the payload in flight (on either a rocket or a balloon). The Wi-Fly chip is mounted on a 'shield' which can be attached directly to the Arduino, and is therefore about the same size as the Arduino. Figure 4.3 shows the Wi-Fly chip mounted on its shield.



Figure 4.3: Wi-Fly chip on its shield

4.1.4 Open Log SD Writer

The Open Log chip stores data from the sensors onto a Micro-SD card (the type used on many smart phones). This ensures that the data will be saved even if the ground station temporarily loses the Wi-Fi connection with the payload. It also allows students to get useful data from the payload if they have not yet integrated the Wi-Fly card, or do not have a local Wi-Fi network over which to communicate in their classroom. Please note that reading the data from a Micro-SD card requires either a computer with a built in SD card reader, or an external USB Micro-SD card reader. Figure 4.4 shows the Open Log chip (right) and an SD card (Left).

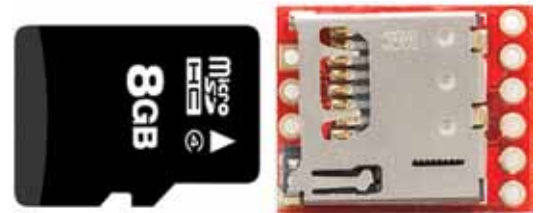


Figure 4.4: SD Card (left) and Open Log SD Writer (right)

4.1.5 GPS Receiver

The GPS (Global Positioning System) chip provides the three dimensional location (Altitude, Latitude, Longitude) of the S4 payload as a function of time. These data are stored with the data from experimental sensors so that they can be correlated to return scientific results. Figure 4.5 shows the GPS receiver; the black cylinder is its helical antenna.



Figure 4.5: GPS Receiver with helical antenna

4.1.6 Duck Antenna

The S4 payload uses a 2.4 GHz antenna to communicate through the Wi-Fly chip to the ground station computer. This "Duck" antenna is shown in Figure 4.6.



Figure 4.6: Duck 2.4GHz antenna

4.1.7 Logic Level Converter

The logic level converter is an electrical device that converts the high (9V) voltage from the battery to the 3.3 V level needed by the Arduino and the other flight board components. It is shown in Figure 4.7.



Figure 4.7: Logic Level converter

4.1.8 Voltage Regulator

In order to ensure that the voltage to the Arduino does not exceed 3.3V, the flight board includes a voltage regulator, which is shown in Figure 4.8.



Figure 4.8: Voltage Regulator

4.2 Sensor Components

There are many different choices for sensors that students can use to perform individual experiments. The S4 team has provided library routines that can be used to support the following sensors: accelerometer, magnetometer, barometric pressure/temperature sensor, and the humidity/temperature sensor.

4.2.1 Accelerometer

The accelerometer sensor measures how fast the payload is accelerating in any direction, returning values for each of the X, Y and Z axes. It measures static acceleration due to the Earth's gravitational force, as well as acceleration resulting from motion (units in g's). These data can be used to determine useful information about the flight path of the payload or to provide the orientation of the payload. Although it only provides relative information, the accelerometer does not require GPS signals from external satellites. The shield is labeled with a diagram depicting the actual orientation of the X, Y, and Z axes. Its performance is limited to 16 times the g-force from the Earth's surface, which may be exceeded in some high-powered rocket launches. Figure 4.9 shows the accelerometer sensor.



Figure 4.9:
The accelerometer

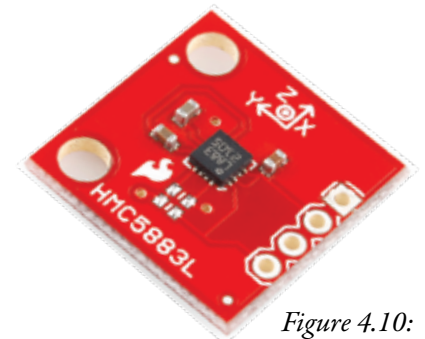


Figure 4.10:
The magnetometer

4.2.2 Magnetometer

The magnetometer measures the strength of magnetic fields along 3 axes, returning the strength of the magnetic field in each of the X, Y and Z axes (units in mG). The shield is labeled with a diagram depicting the actual orientation of the X, Y, and Z axes. Figure 4.10 shows the magnetometer sensor.

4.2.3 Barometric Pressure/Temperature Sensor

The barometric pressure and temperature sensors are integrated into a single device which returns two values, one for pressure and one for temperature. This barometric pressure sensor covers a range of pressures from 300-1100 hPa with accuracy down to 0.03 hPa (1 hPa = 100 Pascals, and 1 Pascal = 1 N/m²). The temperature is recorded in Celsius. Figure 4.11 shows this sensor.



Figure 4.11:
The barometric pressure/
temperature sensor

4.2.4 Humidity/Temperature Sensor

The humidity sensor measures relative humidity (RH) with temperature compensation. The device also consists of a standalone temperature sensor output, which is measured in Celsius. Figure 4.12 shows this sensor.

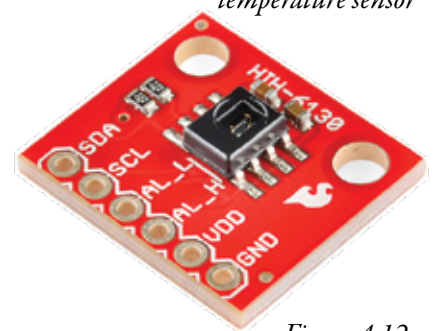


Figure 4.12:
The humidity/temperature sensor

5 Building and Testing the S4 Payload

5.1 Assembling the Flight Board

As mentioned previously, the flight boards can either be pre-assembled by the instructor or by (more advanced) students. In this section, we provide an illustrated guide to how to do this assembly.

The first thing to note is that all the pieces you will be attaching to your board have wires that come out from them and go through designated holes which are labeled on the flight board. These wires are called leads.

Here are the basic steps to follow with each piece:

- Identify the part you wish to attach and be sure you place the leads of the part through the holes in the board in the appropriate orientation. Most are labeled and can be easily seen. It is important that the leads come out the back of the board and not the front. All the components EXCEPT THE ARDUINO mount to the front of the board. Each part has an individualized, labeled space and holes for its leads.
- Using electrical tape, secure the part (LED, capacitor, etc.) to the board; be careful to make sure it is straight and flat up against the board.
- Turn the board upside down so the leads are facing up and place the board in the helping hands or clamps to hold it still.
- When instructions call for you to “cut the leads” unless otherwise noted, you should clip the leads as they extend out from the solder, usually on the back of the flight board.

5.1.1 LEDs

Part A – Red LED (GPS)

The first item we will attach is the red LED: the side view of the LED is shown in Figure 5.1 and the top view is shown in Figure 5.2.

Figure 5.2: Top view of the red LED – the cathode (flat) side is on the right.

Notice that the leads on this LED are uneven. The longer lead is the anode (positive side), while the shorter one is the cathode (negative side). The cathode side is flat when viewed from the top (see Figure 5.2).

On your flight board, where it says GPS Status (Figure 5.3), you will see that one side of the circle is flat (the top side in the Figure). You will want to match the flat side of the LED with the flat side drawn on your board. ***It is crucial that you put the LED in the board in the right direction.*** After soldering this part onto the flight board, cut the leads.

When the board is completed, this LED will go from blinking to a steady glow which indicates that the GPS signal has been locked (stably acquired from the GPS satellites).

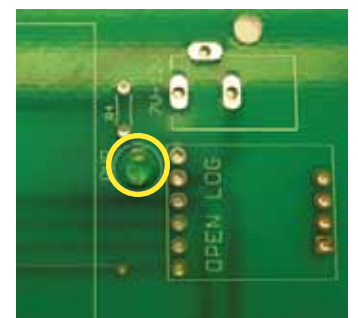
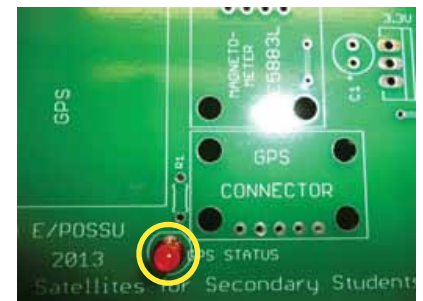
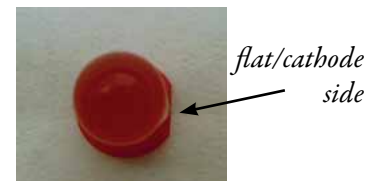
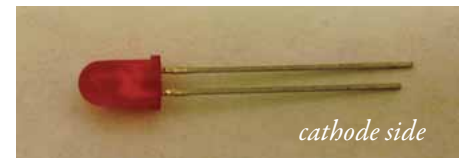
Figure 5.3: Red LED secured into flight board

Part B – Green LED (Power)

The green LED is indicated by the arrow in Figure 5.4. As shown in the Figure it goes in the spot labeled power on the board. ***Remember to be sure to line up the flat side of the LED with the flat side on the drawing as you did with the red LED.*** After soldering this part onto the board, **cut the leads**. When the board is completed, this LED will shine green when there is power to the Open Log chip.

Figure 5.4: Green LED secured into flight board

Figure 5.1: Side view of a red light-emitting diode (LED): the shorter lead is the cathode (negative side).



5.1.2 Capacitors

Now we will solder on the capacitors. They are located near the square labeled “logic level converter” and are called C1 and C2. The capacitors are exactly the same so it does not matter which one you solder on first. However, the direction in which they are placed into the holes in the board does matter. It may be difficult to see, but there is writing on a white band on one side of the capacitor. *You want the writing to face away from the labels that say “C1” or “C2”.* Both capacitors need to be facing the same way. In Figure 5.5, you can see the two capacitors secured to the board. After soldering these parts onto the board, **cut the leads**.

Figure 5.5: Capacitors C1 and C2 secured to the flight board

5.1.3 Power Jack

Next, we will install the power jack. It goes in the space labeled “7V-12V” which is adjacent to the space designated as “Open Log”. You will want the “hole” facing out of the board as shown in Figure 5.6. After soldering this part onto the board, **cut the leads**.

Figure 5.6: Power jack secured to flight board

At this point, you should have 5 parts secured to your board. They are the red LED (GPS), the green LED (power), both of the capacitors (C1 and C2), and the power jack.

5.1.4 Resistors

Next we will be adding on the four resistors.

a) The first one has resistance equal to 62Ω (measure to be sure). You will be placing its leads into the holes labeled R1 next to the red LED. The resistor can be put into place in either direction, however be sure you are placing it in as flat as possible and tape it down before soldering. After soldering this part onto the board, **cut the leads**.

Figure 5.7: GPS resistor close-up

b) The “R4” resistor has resistance equal to 402Ω . It will be placed in the holes labeled “R4” right next to the power jack as shown in Figure 5.8. Again, the resistor’s direction does not matter. The only thing you must do is try to push it as flat against the board as possible. After soldering this part onto the board, **cut the leads**.

Figure 5.8: Power resistor close-up

c) The next resistor, which has a resistance equal to $2.2k\Omega$, will be placed into holes labeled R3. After soldering this part to the board, **cut the leads**.

d) The last resistor, which has a resistance equal to $2.2k\Omega$, will be placed into holes labeled R2. After soldering this part to the board, **cut the leads**.

Figure 5.9: Magnetometer resistors close-up

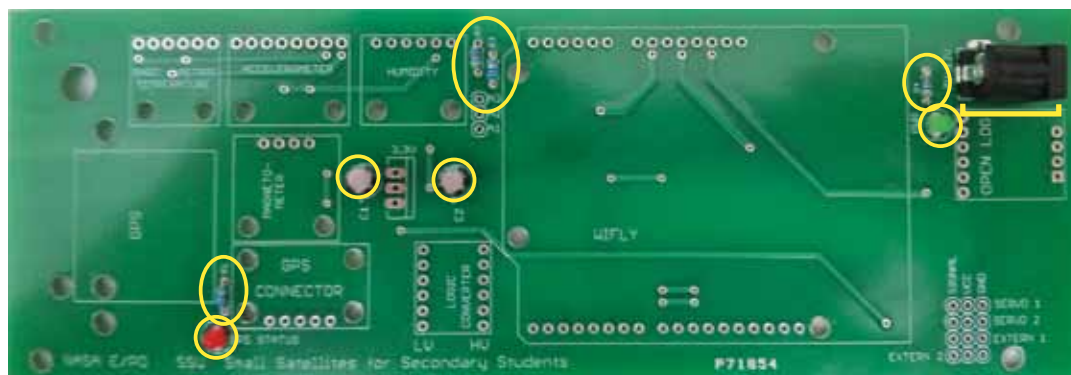


Figure 5.10: Flight board including both LEDs, the power jack, all resistors, and both capacitors.

5.1.5 Making and Installing Male Header Pins

A- Header Pin Strip

In your kit, you should see that you have long strips of black header pins. They are shown in Figure 5.11.

Use pliers to hold the number of pins you wish to use and then break that amount off of the long chain.

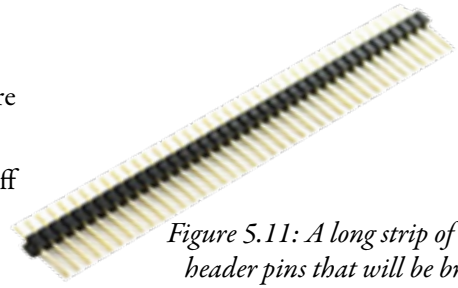


Figure 5.11: A long strip of male header pins that will be broken into smaller pieces

B- Capacitor Header pins

For this step, break off a piece with three pins. You should place the header pins into the part of the board labeled “A1, A2, A3” that is near capacitor 2. When placing the header pins into the board, you want the longer ends facing up on the top of the board. Figure 5.12 shows a bird’s eye view and Figure 5.13 shows a side view of how this part should look on your board. After soldering this part onto the board, **cut the leads**.

Figure 5.12: Black header pins for A1, A2, A3 (top)

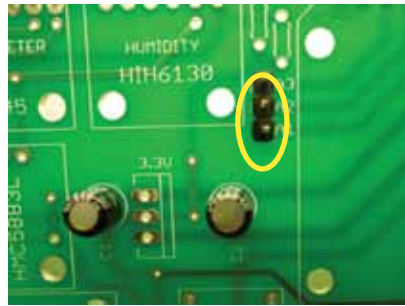


Figure 5.13: Black header pins for A1, A2, A3 (side)

C- External Controller Header Pins (Servo 1, Servo 2, Extern 1, Extern 2)

In this step you will use the same type of pins as in the last step. Use pliers to break off four more sections from the long strip of header pins: each section should have three pins. They will go on the same side of the board as the power jack in the slots labeled, “Servo 1”, “Servo 2”, and “Extern 1”, and “Extern 2” as shown in Figure 5.14. *It is important at this step to be sure you have the longer leads sticking out on top of the board, not underneath.* Also, it is easiest to solder these parts if you do all of them together in one step. You will want to tape the pieces as shown in Figure 5.15, i.e., parallel to the long side of the board. If you tape them the other way, they might be soldered unevenly. After soldering these parts onto the board, **cut the leads**.

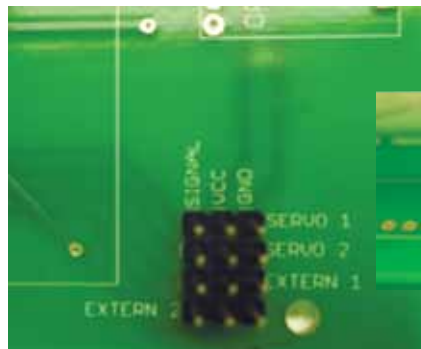


Figure 5.14: Four Sets of three Header Pins



Figure 5.15: How to tape header pins to ensure even solder joints

D - Logic Level Converter Header Pins

The next parts to be soldered are two sets of six male header pins for the logic level converter. These are placed into the space labeled “Logic Converter” that is below the two capacitors, as shown in Figure 5.16. Unlike the other male header pins you have used so far, these should be oriented with the long pins facing down. After soldering these parts onto the board, cut the leads.

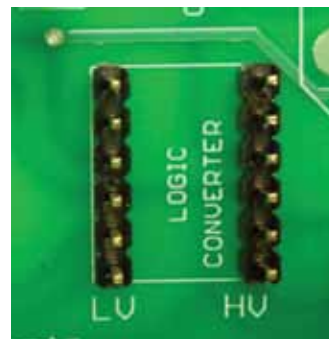
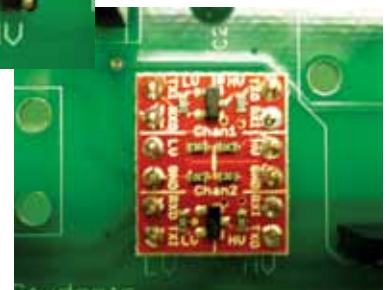


Figure 5.16: The two sets of six header pins for the logic level converter

5.1.6 Logic Level Converter

Slide the logic level converter over the header pins and solder the device into place, as shown in Figure 5.17. The LV and HV sides are marked on the Logic Level Converter and should match the labeling on the board.

Figure 5.17: The logic level converter



5.1.7 Voltage Regulator:

Next we will assemble the voltage regulator. Apply a thin coating of the thermal compound to the voltage regulator, then press on the mica insulator as shown in the Figure below:

Line the holes up with that of the heat sink, and fasten all three components together by putting the nylon screw through the hole and fastening it with the nut as shown in Figure 5.18 (left). Finally, the voltage regulator is soldered into place between the two capacitors, in the spot labeled “3.3 V.” Be sure to orient the voltage regulator as shown in Figure 5.18 (right). After soldering this part onto the board, **cut the leads**.

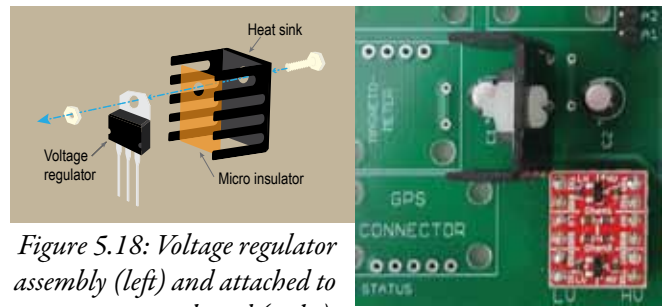


Figure 5.18: Voltage regulator assembly (left) and attached to board (right).

5.1.8 Female Header Pins

a) Barometric Pressure/Temperature Pins:

Place one of the 6-pin female header pins in the box marked “BAROMETRIC TEMPERATURE” as shown in the image below. Solder this part onto the board and then **cut the leads**.



Figure 5.19: Barometric header pins attached

b) Accelerometer Pins:

Place one of the 8-pin female header pins in the box marked “ACCELEROMETER” as shown in the image below. Solder this part onto the board and then **cut the leads**.



Figure 5.20: Accelerometer header pins attached

c) Humidity/Temperature Pins:

Place one of the 6-pin female header pins in the box marked “HUMIDITY” as shown in the image below. Solder this part onto the board and then **cut the leads**.



Figure 5.21 Humidity header pins attached

d) Open Log Pins:

Start with one of the 8-pin female header and snap it in half to give you two 4-pin female headers (save one for the Magnetometer in section e). Place both your 4-pin connector and 6-pin headers in the appropriate holes in the box marked “OPEN LOG.” Solder the parts onto the board and then **cut the leads**.



Figure 5.22: Open Log header pins attached

e) Magnetometer pins:

Place the remaining 4-pin female header from section d in the appropriate holes in the box marked “MAGNETOMETER.” Solder this part onto the board and then **cut the leads**.



Figure 5.23: Magnetometer header pins attached

f) GPS Pins:

Start with a 6-pin female header and snap off 1 pin to leave yourself with a 5-pin female header. Place the 5-pin header in the appropriate holes in the box marked “GPS CONNECTOR”. Solder this part onto the board and then **cut the leads**.

Figure 5.24: GPS header pins attached



5.1.9 Arduino/WiFly Header Pins

This final step attaches the female header pins that are used to attach both the Arduino and the WiFly shield. Since we are attaching devices to both the top and bottom of this particular set of header pins, the leads must NOT be cut.

Along the top of the box marker “WIFLY” place a 6-pin female header (left) and an 8-pin female header (right). Along the bottom of the box marked “WIFLY” place an 8-pin female header (left) and a 10-pin female header (right). Solder these parts into place. After soldering, **DO NOT CUT THE LEADS**.



Figure 5.25: Arduino/WiFly Header pins



Figure 5.26: the underside of the flight board showing the uncut Arduino header pins.

5.1.10 Attach Stand-Offs

Included with your kit is a collection of nylon stand-offs that attach to the board and keep the sensors securely fastened during flight. At this point, it is recommended that you attach the stand-offs to the board for any sensors you intend to use. If you know you are NOT going to use a particular sensor, you can feel free to not attach those stand-offs. The standoffs are attached to the flight board with a single screw through the bottom. A second screw through the top will be left off until the sensors are attached. Use Figure 5.27 to help you attach your standoffs to the correct holes only.



Figure 5.27: Stand-offs

5.1.11 Complete Pre-Assembled Flight Board

Figures 5.28 and 5.29 show the top and bottom views (respectively) of the complete pre-assembled flight board.

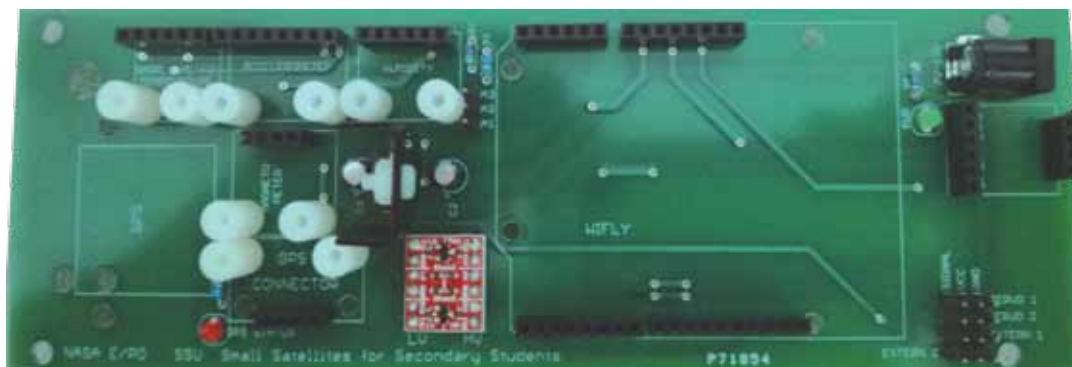


Figure 5.28: Complete Flight Board (top)

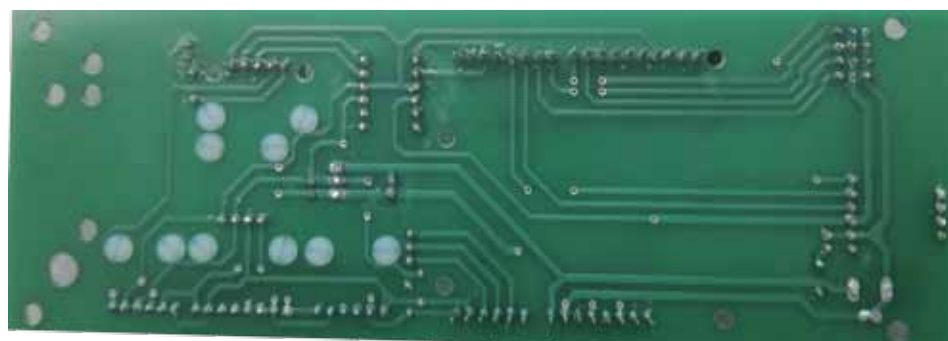
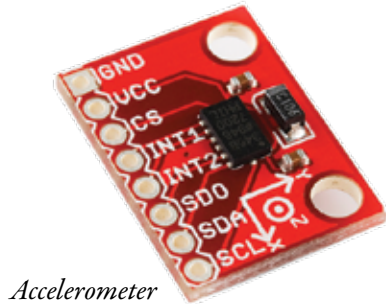


Figure 5.29: Complete Flight Board (bottom)

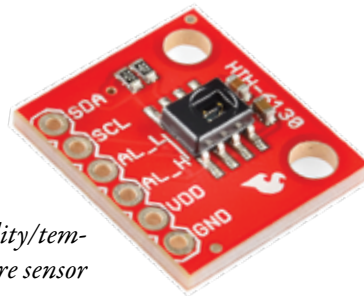
5.1.12 Put Header Pins on Sensors:

You may have noticed by now that the various sensors that snap onto the completed S4 flight board do not arrive from the manufacturer with the headers pins attached. After completing the S4 flight board and getting some experience soldering, you should have little trouble attaching the header pins to the various sensors.

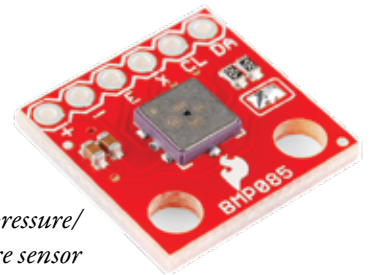
Accelerometer, Humidity, Barometric Pressure and Temperature Sensors:



Accelerometer



Humidity/temperature sensor



Barometric pressure/temperature sensor

Each of these chips has a single row of 6 male header pins to attach. Snap off a section of 6 male header pins, and slide them through the row of holes along the side of the chip with the long side of the pin facing down and the small side of the pin facing up through the hole. Solder these in place.

Magnetometer:

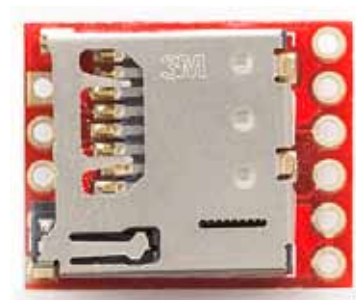
This chip has a single row of 4 male header pins to attach. Snap off a section of 4 male header pins, and slide them through the row of holes along the side of the chip with the long side of the pin facing down and the small side of the pin facing up through the hole. Solder these in place.



The magnetometer

Open Log:

This chip has one row of 6 male header pins, and one row of 4 male header pins to attach. First, snap off a section of 6 male header pins, and slide them through the rows of holes along the side of the chip with the long side of the pin facing down and the small side of the pin facing up through the hole. Solder these in place. Next snap off a section of 4 male header pins and slide them through the rows of holes along the opposite side of the chip with the long side of the pin facing down and the small side of the pin facing up through the hole. You can solder three of these in place, however the fourth will be unreachable because of the apparatus which holds the SD card in place. That's ok because these four pins do not relay any power or data and are for structural purposes only.



Open Log

WiFly:

This chip has a two rows of 6 male header pins, and two rows of 8 male header pins to attach. Snap off two sections of 6 male header pins, and two sections of 8 male header pins and slide them through the rows of holes along each side of the chip with the long side of the pin facing down and the small side of the pin facing up through the hole. Solder these in place.



WiFly

5.2 Powering the S4 Payload

The S4 Payload is capable of being powered in a couple of different ways. A 9V power supply (either battery or wall adapter) can be attached to the barrel jack at the back of the flight board. Alternatively, the payload can also be powered by the 5V signal provided by the USB cable. The Arduino steps the voltage down from 9V to 5V.

5.2.1 Powering the Payload During Development

During the development phase, the S4 Payload will be attached to your programming computer by the USB cable. The USB port is on the Arduino, which is attached to the bottom of the payload. This cable is needed to load new versions of the Arduino sketches onto the board as you refine your program. This cable also provides enough power to run the entire S4 Payload.

Fig 5.30 – Payload attached to computer over USB.

Note: older machines that have USB 1.0 ports will not provide enough power to run the payload. If this is the case, you may need to purchase a wall adapter to provide the needed power. You will still be able to load the new sketches over USB 1.0.

5.2.2 Powering the Payload During Field Testing

After you have your payload assembled and programmed, you will likely want to do some field testing, where you detach the payload from your machine and let it collect data and transmit it wirelessly to your server, or just write it to the SD card. During this phase, you can power the device by attaching a 9V battery to the 9V barrel jack using the 9V barrel jack adapter.

Fig 5.31 – Payload attached to 9V.

Fig 5.32 – Payload attached to Wall Adapter.

Note: The S4 Payload kit does NOT include the wall adapter; but should you decide this meets your needs best, it can be purchased inexpensively here:

<https://www.sparkfun.com/products/298>

5.2.3 Powering the Payload During Flight

During flight, the only power option is the 9V battery adapter. Of course, the battery itself needs to be physically secured for the flight. Figures 5.33 and 5.34 illustrate two options for securing the battery.

1) Secure the battery directly to the inside of the rocket or balloon gondola



Fig 5.33 Battery secured to rocket or balloon gondola

2) Secure the battery to the Flight Board using a battery harness



Fig 5.34 – Battery harness

Once the board has power, the logic level converter will step down the voltage from 5V to 3.3V, and the Arduino has the capability to communicate with all of the devices. Most of the other base and sensor component devices operate at 3.3 volts.

Important warning: some 3.3V devices (and especially the GPS receiver) are very sensitive to over voltage. If you are using an Arduino that operates at 5V, and it sends a 5V signal over its data lines, you can destroy a 3.3V device. The S4 flight board ensures the correct voltages for all the supported devices and sensors. If you are testing devices on a breadboard; be sure that the devices are being powered with the correct voltage or they will be destroyed.

5.3 Testing the Assembled Flight Board

Configure Multimeter:

First, look at the three holes on the bottom right corner of the multimeter. The red cable should be plugged into the second hole labeled “VΩmA”. The black cable should be plugged into the last hole labeled “COM”. The dial should be turned to the location labeled ‘20’ in the DCV box. The device should be powered on.

Figure 5.35: Multimeter with probes attached for reading voltage



Cut two pieces of wire about three inches long and strip both ends of each one. Wrap one end of each wire around the metal tip of each multimeter cable (red and black) and secure it with electrical tape. Now the end of each cable will have an exposed end of stripped wire to insert into the pin holes on the flight board.

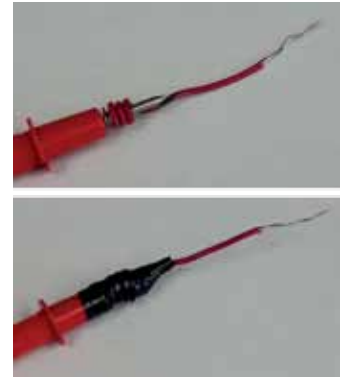


Figure 5.36: Multimeter probes modified to fit into pin holes

Supply power to the board by plugging in either a 9V battery or the wall adapter to the 9V barrel jack. **Do not test the voltages with the power supplied to the board by the USB connection to the Arduino.** Make sure the multimeter is on and configured as shown in Figure 5.35 above. We will now step through each device on the Flight Board and test to make sure that power is being supplied to that device using the multimeter.

5.3.1 Barometric Pressure/Temperature Sensor:

Power will be the first pin hole from the left, and ground will be the second hole. You should measure around 3.3V

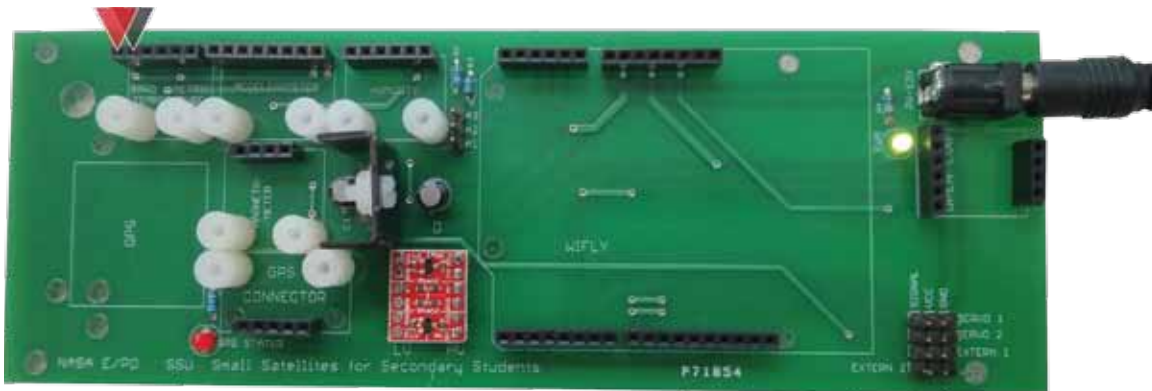


Figure 5.37: Testing the power to the barometric pressure/temperature sensor pins

5.3.2 Accelerometer:

Ground will be the first pin hole from the right, and power will be the second hole. You should measure around 3.3V

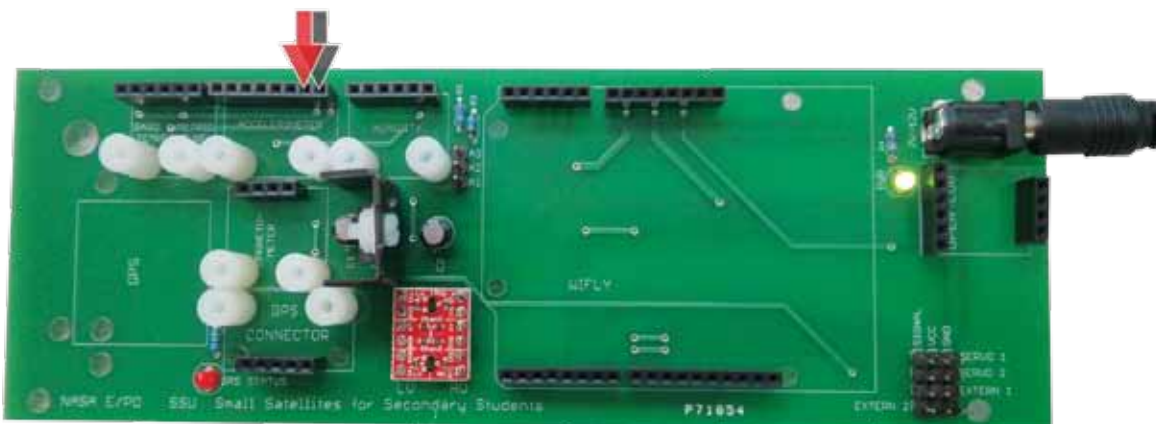


Figure 5.38: Testing the power to the accelerometer sensor pins

5.3.3 Humidity /Temperature Sensor:

Ground will be the first pin hole from the left, and power will be the second hole.
You should measure around 3.3V

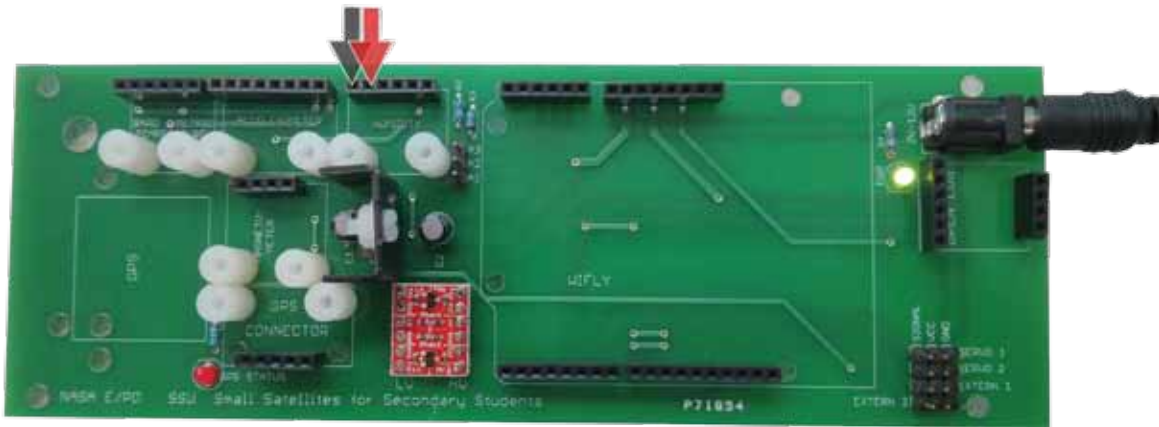


Figure 5.39: Testing the power to the humidity/temperature sensor pins

5.3.4 Magnetometer:

Ground will be the first pin hole from the left, and power will be the second hole.
You should measure around 3.3V



Figure 5.40: Testing the power to the magnetometer sensor pins

5.3.5 Open Log:

Ground will be the second pin hole from the top of the left vertical strip of header pins, and power will be the third hole down on that same strip.

You should measure around 3.3V



Figure 5.41: Testing the power to the Open Log pins

5.3.6 Arduino/ WiFly Shield:

The top side contains two strips of female header pins, the left strip with six pins and the right strip with eight pins. Power will be the first pin from the left on the right-hand strip of headers. Ground will be the second pin from the left on the right-hand strip of headers. Note: The pins on the WiFly Shield and Arduino share these pins.

The multimeter should read around 9V.

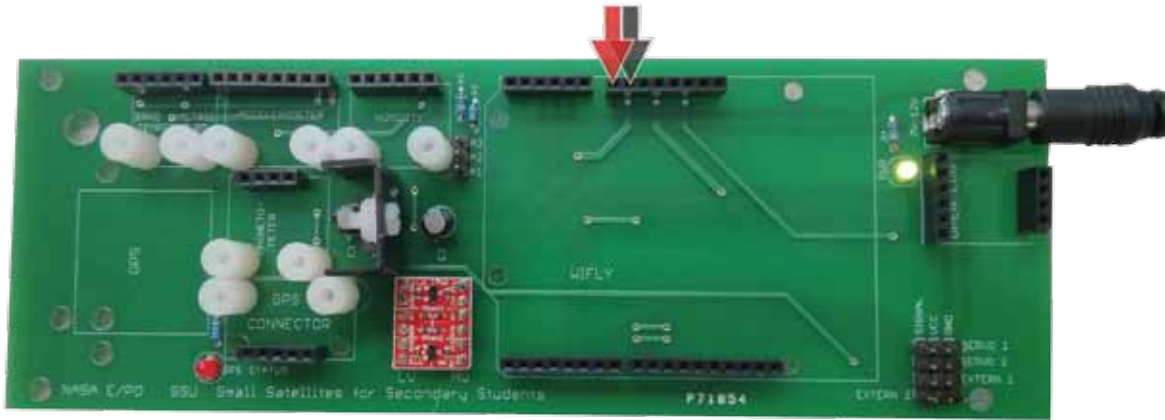


Figure 5.42: Testing the power to the Arduino/WiFly shared pins

5.3.7 GPS Connector

Ground will be the second pin from the left, power will be the first pin from the left.

You should measure around 3.3V



Figure 5.43: Testing the power to the GPS pins

If all of your power readings check out correctly, then congratulations! You have succeeded in correctly building and soldering the flight board. In the next chapter you will learn how to load code into the Arduino to run the board components.



6 Programming the Arduino

In Section 3.3.2, we discussed the advantages of using an Integrated Development Environment (IDE) to develop and debug computer programs. We also discussed libraries of code that the S4 team has developed to help you to program your flight board so that it can acquire data from the sensors that are supported by the S4 team (Section 4.2) and that are listed in the Preferred Parts List (Appendix F). In this chapter, we provide detailed instructions to help you learn how to program the Arduino to acquire and telemeter (send) data using the on-board Wi-Fly device.

6.1 Obtaining Arduino and S4 Software

Step 1 – Download the S4 Workspace from the S4 website (http://s4.sonoma.edu/?page_id=167). Be sure to select the version of the S4 Workspace that corresponds to your operating system.

Step 2 – Extract the S4 Workspace (.zip) file that you downloaded. The location where you extract this file will be where all your files and resources for the S4 project will live, including any programs you write yourself. It's recommended you place this file on your desktop, or in a location where you can easily access it.

6.2 Basic Arduino Programming Commands and Punctuation Marks

The following commands and punctuation marks are commonly used in the Arduino computer language (which is called “Processing”).

Curly Brackets {}: are used to indicate the beginning and end of a function.

Semicolon: A semicolon is used to end a statement and separate elements of the code.

Setup(): Code written in the setup function will be executed once and only once when the program first starts.

Loop(): Code written in the loop function will be executed over and over again for as long as the device is running. This function is really the core of your program.

Void: this word often appears before the titles of functions such as setup and loop. In the Processing language, when you define a function, you must also define the type of variable that it will return. Using the void command simply means that the function will not return any type of value.

Comments: Any text written after double forward slashes // will be ignored. Comments are used to write notes to yourself as you write your program. These types of notes are very helpful when you need to troubleshoot your code, as they help you to recall the purpose of the code that you originally wrote.

#define: This command defines a variable to make our code cleaner and easier to write and read. It tells the Arduino to use the new value every time it sees the variable name. For example: #define (LED, 13) will write the number 13 every time it sees the variable name “LED” in the code.

PinMode(____): Tells Arduino how to configure a certain pin. First, in the parentheses, comes the number of the pin you wish to specify followed by the mode of the pin. For example: pinMode(LED, OUTPUT) will tell Arduino to make the LED pin an output. pinMode is a function. The information inside the parentheses are arguments.

DigitalWrite(____): This turns on or off any pin that is an output. The first value in the parentheses, specifies which pin; that argument is followed by either “HIGH” (on), or “LOW”(off). For example: digitalWrite(LED, HIGH) will turn the LED on.

Delay(____): the delay function tells the Arduino to do nothing for a specified number of milliseconds. For example: delay(1000) will tell the Arduino to wait and do nothing for 1000 milliseconds or 1 second.

6.3 Communicating with the Arduino

The Arduino has several methods (“protocols”) that it uses to communicate with external devices. Each peripheral device has its own communication method that is defined by the manufacturer of the device. In this section, we will describe three communications protocols used by S4 base and sensor components.

Serial Communication

The most common communications method, and the one that is used to load sketches onto the Arduino, is the Serial Interface. Serial communications only require two pins to be connected: transmit (TX) and receive (RX). These pins use the exact same serial port as the USB cable and will interfere with one another. When using the hardware Serial interface, only one device can be connected to these pins (RX = Pin 0 and TX = pin 1). If you have a second device (e.g. a sensor) that is connected to these pins on the Arduino, and you try to reprogram it, you will fail because the new program tries to load onto the Arduino using these same pins. To alleviate this problem, you can either unplug the peripheral device while you upload of the new program, or you can use the Software Serial command. The Software Serial command allows you to use two of the digital I/O pins (Pins 2-13) to perform serial communications. This method has some data speed limitations, but is useful for most serial devices.

Synchronous Serial Data Protocol (SPI) Communication

The Arduino also supports SPI communication, Synchronous Serial Data Protocol, which requires four pins. This type of communication allows multiple devices to be controlled on the same set of pins. The master device (Arduino) alerts the slave device (the peripheral) when it would like to receive data. The alert is accomplished by raising the voltage on the slave select pin to HIGH, which allows the slave device to accept commands. Selecting which device you want to talk to allows for multiple devices to share the same data pins. This type of communication only works over short distances, so the peripheral device needs to be located adjacent to the Arduino on the flight board. In the S4 payload, the WiFly uses SPI communication., and needs to have priority over any other SPI device. If your code interrupts communications with the WiFly to talk to another device on the SPI network, an error will occur. This is a rare problem but will prevent the real-time telemetry from functioning correctly.

I2C Communication

The last and most common communication method for sensors is TWI or I2C communications. This protocol only uses two wires, a clock line (SCL) and a data line (SDA). Each device has its own unique 7 bit address which is in hexadecimal (base 16). For example, the barometric sensor may have the address 0x77, where the 0x specifies that it is a hexadecimal number.

6.4 Programming the Arduino to Collect Data

With an understanding of the basics of coding (see Section 3.3), and simple commands and syntax in the Processing language (Section 6.2), you can now begin programming the Arduino to collect data and/or to control various functions by turning on and off different pins. To do this, you will write small Arduino computer programs, which are typically called “sketches.”

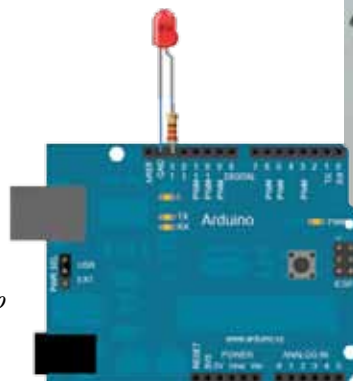


Figure 6.1: The Arduino attached to a red LED with a 220Ω resistor.



Figure 6.2 (above): An image of the Arduino connected to a computer with a USB cable

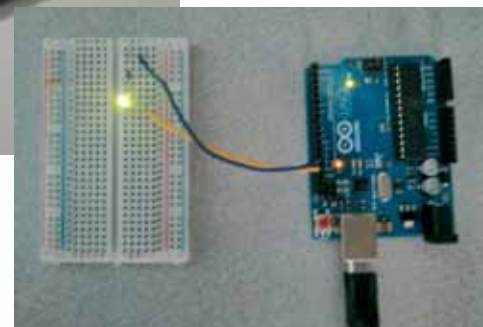


Figure 6.3 (below): An image of the completed circuit in a breadboard.

The Arduino programming cycle consists of the following steps:

Step 1: Plug your Arduino board into the USB port on your computer using a standard USB cable

Step 2: Start the Arduino IDE and write a sketch that will bring your board to life

Step 3: Compile that sketch into a binary file

Step 4: Upload the binary file to the Arduino through the USB connection and wait a few seconds for the program to start.

Step 5: Watch and enjoy as the Arduino executes the sketch

6.4.1 Writing a Sketch to Turn On and Off an LED Connected to the Arduino:

The blinking LED test is the first simple activity. If you aren't familiar with LEDs yet, it's important to note that the longer lead on the LED is the positive side; this lead should be connected to Pin 13 on the Arduino. The shorter lead is the negative side, and that should be connected to the pin marked ground (GND) on the Arduino. If you are not familiar with using circuitry breadboards (as shown in Figure 6.3 image) see Appendix C for some suggested resources.

Open the Arduino IDE and load this code automatically by selecting [File](#) ▶ [Examples](#) ▶ [Basics](#) ▶ [Blink](#)

[Code]

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
  */


// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;


// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```


Figure 6.4: The Arduino “Blink” sketch that turns an LED on and off every second


Along the top of the Arduino IDE, you should see a series of icons. Let's take a moment to look at what these do.


 **Verify** – This tells the Arduino IDE to compile the sketch we have created. If we have made any mistakes while writing our code, the compiler will give us error messages. The messages can be used to identify and correct the mistakes we've made so our code will execute properly. This button will not send the compiled code to the Arduino.

 **Upload** – This button tells the Arduino to first compile the sketch we have created, and then upload it to the Arduino. This does everything the verify button does, but it does send the compiled code to the Arduino. Obviously, the upload button will return an error message if the Arduino is not connected to the computer with a USB cable.

 **New** – Opens a new blank Arduino sketch.

 **Open** – Open an existing Arduino sketch.

 **Save** – Save the current Arduino sketch. Pro tip: Save early, save often.

 **Serial Monitor** – This opens the Serial Monitor which allows you to get feedback from the Arduino as it is running. Note: starting the Serial Monitor will reset/restart the sketch currently loaded in the Arduino.

Now, click the upload icon  to send your blink sketch to the Arduino and see what happens!

Now that the sketch is running on your Arduino, let's take a moment to examine what's really happening. The Blink sketch first defines the LED variable as the number 13. Next, it tells the Arduino to set the LED pin (which is number 13) as an output pin. It tells it to do this just once in the setup function. The program then begins its loop function. First it tells Arduino to turn the LED pin on (HIGH). Then it tells the Arduino to do nothing for 1 second, using the `delay(1000)` command. Next it tells Arduino to turn the LED pin off (LOW). Finally it tells Arduino to do nothing for another 1 second. It will repeat this loop over and over until the programmer stops it. This will cause the physical LED that is connected to pin 13, to blink on and off with 1 second in between each blink.

6.4.2 Printing to the Serial Monitor

When programming the Arduino, it's often important for the Arduino to send information to the computer you are using to program it, and display that information on the computer screen so that you can see what is happening. We do this through use of the Serial Monitor, which is a component built into the Arduino IDE. Let's take a look at a simple sketch that will write a short message to the serial port, and then display it on your computer screen. The sketch below will simply write the text "Hello World" every second. This is a standard beginning program to write when you are learning a new computer language.

Step 1 - Open the Arduino IDE and load this code automatically by selecting [File ▶ Sketchbook ▶ S4_ SerialMonitor](#)

```
[Code]

void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.println("Hello World!");
  delay(1000);
}
```

Figure 6.5: The Arduino "Hello World" sketch that writes to the computer screen

Looking at the `setup()` function, you can see that it calls a function called `Serial.begin()` and passes a value of 9600. This is called the 'baud rate' and is a measure of how quickly bits of information are being sent between the Arduino and the computer; for most applications in our program, 9600 is a good value for the baud rate.

Looking at the `loop()` function, there is a new word, the `Serial.println()` function, which is used to write a string of text to the serial port. After this, the program delays for 1 second before repeating this action.

Step 2 – Upload and run the code; you probably won’t see anything happen. In order to see the text “Hello World”, you first need to open up the Serial Monitor from the Arduino IDE by clicking on Tools ► Serial Monitor, or clicking on the Serial Monitor icon. You should now see the string “Hello World” being written to your Serial Monitor once every second.

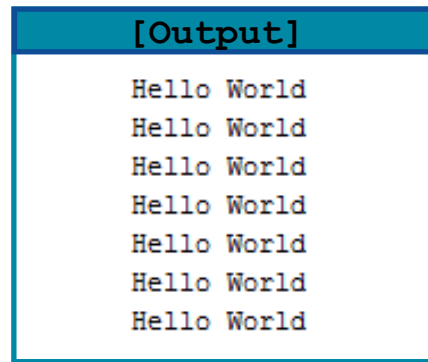


Figure 6.6: Output of “Hello World” sketch on the computer screen

6.5 Coding and Testing Base Flight Board Sensors

The Arduino collects data through electrical and communications interfaces with the other devices on the flight board. (The three communications protocols used by the Arduino were discussed in Section 6.3 above.) Each device has its own requirements, e.g., different communications protocols, specific calibrations, or different ways to interpret the output data values. Fortunately, much of the code needed to make the S4 devices work has been written for you, and is included in the S4 library.

In order to use any of the S4 sensors with your flight board, you must first complete the following tasks:

- 1) Assembly of the flight board including the desired device (Chapter 5)
- 2) Downloading and extracting the S4 workspace (Section 6.1)

The following sub-sections contain detailed instructions on testing each of the devices that comprise the complete S4 payload. For each device, an image will be shown that indicates which devices should be attached. For **ALL** of the examples included, **the Arduino should be attached to the back of the board** as shown in Figure 6.7.



Figure 6.7 The Arduino attached to the S4 flight board

Note: For the remainder of Chapter 6, the subsections need not be completed in the order presented. Each device can be tested completely independently of all other devices (except the Arduino). If you are not using a specific sensor, you can safely skip that subsection.

6.5.1 GPS Receiver Code and Test

The GPS device allows you to know the location of your payload in 3D space at all times. These data are likely to be critical for most experiments. The GPS data will also provide you with a timestamp for time series data. The GPS device requires the ability to lock onto GPS satellites to provide data.

Step 1: Attach the GPS device to the flight board as shown in Figure 6.8.



Figure 6.8: The GPS receiver attached to the S4 flight board

Step 2: Open the Arduino IDE and load this code automatically by selecting **File ▶ Sketchbook ▶ S4_GPS**

```
[Code]

/** Librarys of Code for the project **/
#include <S4GPS.h>
#include <SoftwareSerial.h>

/****Sensor Variables ****/
#define BUFFSIZE 90
char gps[BUFFSIZE]; // variable to store the gps data

S4GPS S4GPS;
void setup()
{
    Serial.begin(9600);
    S4GPS.begin(9600); // initiate the GPS
}

void loop()
{
    if(S4GPS.getGPS(gps)) // wait until we get GPS data
    {
        Serial.println(gps); // print the gps data
    }
}
```

Figure 6.9: GPS sketch for Arduino

Step 3: Upload the sketch to the Arduino and then open the Serial Monitor to watch the data streaming in from the device. The data should look similar to those shown below:

```
[Output]

$GPGGA,210044.00,3820.39934,N,12240.61207,W,1,06,1.78,55.0,M,-29.2,M,,*5B
$GPGGA,210045.00,3820.39922,N,12240.61234,W,1,06,1.78,55.9,M,-29.2,M,,*54
$GPGGA,210046.00,3820.39839,N,12240.61260,W,1,06,1.78,56.4,M,-29.2,M,,*53
$GPGGA,210047.00,3820.39784,N,12240.61293,W,1,06,1.78,56.4,M,-29.2,M,,*57
$GPGGA,210048.00,3820.39738,N,12240.61346,W,1,06,1.78,56.4,M,-29.2,M,,*56
$GPGGA,210049.00,3820.39682,N,12240.61368,W,1,06,1.78,56.9,M,-29.2,M,,*56
$GPGGA,210050.00,3820.39660,N,12240.61340,W,1,06,1.78,57.1,M,-29.2,M,,*51
$GPGGA,210051.00,3820.39646,N,12240.61307,W,1,05,5.23,56.9,M,-29.2,M,,*57
```

Figure 6.10: Example GPS data when the device has satellite lock

The GPS receiver needs to be able to lock onto the network of Earth-orbiting GPS satellites in order to get a fix on its position. Depending on the construction and location of your classroom, you may or may not be able to get a GPS lock inside your classroom. If you don't have a lock, your output will look something like this:

```
[Output]

$GPGGA,,,,,0,00,99.99,,,,,*48
$GPGGA,,,,,0,00,99.99,,,,,*48
$GPGGA,,,,,0,00,99.99,,,,,*48
$GPGGA,,,,,0,00,99.99,,,,,*48
$GPGGA,,,,,0,00,99.99,,,,,*48
$GPGGA,,,,,0,00,99.99,,,,,*48
$GPGGA,,,,,0,00,99.99,,,,,*48
$GPGGA,,,,,0,00,99.99,,,,,*48
$GPGGA,,,,,0,00,99.99,,,,,*48
```

Figure 6.11: Example GPS data when the device does not have satellite lock

Getting this type of data with missing values for the position is enough to confirm that your GPS unit has been attached and coded correctly. However, it may be more interesting if you are able to find a location where the GPS receiver can lock onto the satellites. This may require going outside.

6.5.2 Open Log Code and Test

The Open Log allows the payload to record data even if it loses its Wi-Fi signal. In order to read these data, you must have a computer capable of reading the microSD cards that are used by the Open Log. Most computers will NOT have a microSD reader; but many will have a standard SD card reader. The microSD cards included in the preferred parts list include an adapter that lets them fit into a standard SD card reader. If your computer does not have an SD card reader, an external (USB) SD card reader can be purchased fairly inexpensively.

Step 1: Attach the Open Log device to the flight board, and insert the microSD card into the Open Log as shown in Figure 6.12.

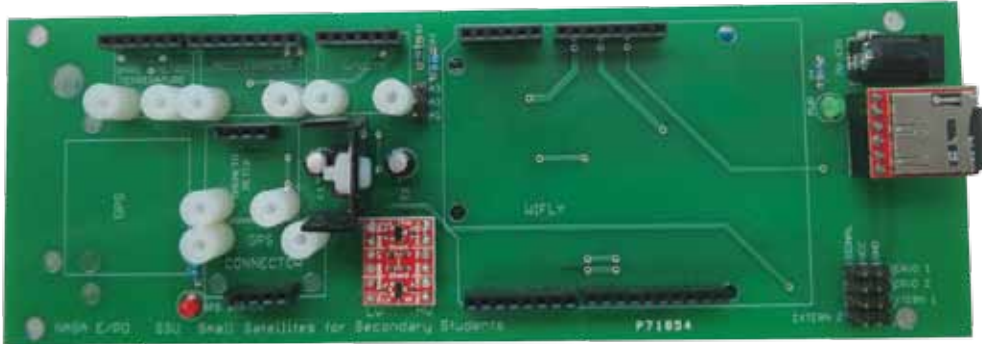


Figure 6.12: Open Log attached to flight board.

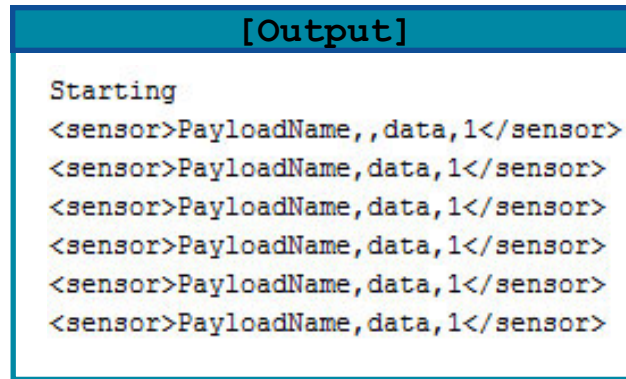
Step 2: Open the Arduino IDE and load this code automatically by selecting **File** ► **Sketchbook** ► **S4_OpenLog**

[Code]

```
/** Librarys of Code for the project **/  
#include <S4.h>  
#include <S4Sensor.h>  
#include <SoftwareSerial.h>  
#include <Wire.h>  
#include <WiFly.h>  
  
S4 S4;  
  
void setup()  
{  
  Serial.begin(9600);  
  S4.useWiFi(false);  
  S4.useGPS(false);  
  S4.begin("PayloadName","DeviceTesting");  
}  
  
void loop()  
{  
  long data = 1;  
  S4.addData("data",data);  
  S4.writeData();  
  delay(1000);  
}
```

Figure 6.13: Open Log Arduino sketch

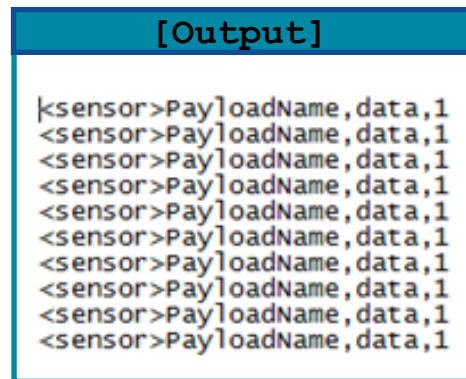
Step 3: Upload the sketch to the Arduino and then open the Serial Monitor to watch the data streaming in from the device. Figure 6.14 shows a generic data stream. The actual data stream will depend on your choice of sensor.



```
Starting
<sensor>PayloadName,,data,1</sensor>
<sensor>PayloadName,data,1</sensor>
<sensor>PayloadName,data,1</sensor>
<sensor>PayloadName,data,1</sensor>
<sensor>PayloadName,data,1</sensor>
<sensor>PayloadName,data,1</sensor>
```

Figure 6.14: Generic example of data from the Open Log device

Step 4: Power down the payload, then remove the microSD card from the payload, place it in the standard SD card adapter, and place that adapter inside your computer's SD card reader. Once your computer recognizes the new device, you should be able to open the files (LogXXXXXX.txt) written to the SD card in a plain text format. Figure 6.15 shows the plain text output from the SD card for a generic sensor.



```
|<sensor>PayloadName,data,1
|<sensor>PayloadName,data,1
|<sensor>PayloadName,data,1
|<sensor>PayloadName,data,1
|<sensor>PayloadName,data,1
|<sensor>PayloadName,data,1
|<sensor>PayloadName,data,1
|<sensor>PayloadName,data,1
|<sensor>PayloadName,data,1
|<sensor>PayloadName,data,1
```

Figure 6.15: Generic data output from the SD card in plain text format

Notes: In later chapters, we will show you how to take the raw data from the SD card and import it into programs like Google Earth or Excel to plot values or look at geospatial data in 3D. For now, if you have the data, you can be assured that your devices are working properly. Every time you start your Arduino, it should create a new file.

6.5.3 WiFly Code and Test

The WiFly device allows the payload to communicate wirelessly with the ground station. The other devices we have tested on the payload so far required only the payload and a computer to program it, but testing the WiFly device requires a more complicated procedure. Here's a brief overview of what needs to be done to ensure the WiFly is communicating properly:

- A. Establish a wireless network
- B. Setup your payload hardware
- C. Run the S4_WiFlySetup sketch to program your WiFly card for your network
- D. Run The S4 Server Program On Your Computer
- E. Run the S4_WiFly sketch to test that everything is working properly

A - Establishing a Wireless Network:

It's possible that your school or institution may already have a wireless network established. If you already have a wireless network, and that network requires a password for security reasons, be sure to contact your institution's network administrator and get that password before continuing.

If your institution does not have an existing wireless network, or the existing network is otherwise unsuitable, you may need to establish a wireless network of your own.

Purchasing a Wireless Router:

Wireless routers are widely available and fairly inexpensive. They can be purchased online or at electronics stores for as little as \$30. Wireless routers communicate over different bands, which are labeled with different letters. The WiFly device requires either 802.11b and/or 802.11g. The newest routers (as of the writing of this guide) will probably be 802.11n, but should be backward compatible with 802.11b/g devices. Check to make sure that 802.11 b or g bands are available before purchasing a router, and then follow the manufacturer's instructions to establish your network.

SSID and Network Key:

Once the network is established, be sure to write down two important pieces of information about it to be used in subsequent sections:

SSID – This is the Network's name, which is typically chosen during the process of establishing your network.

Network Key – This is the password that is required to log into the network. If security is not an issue, you may leave the network unsecured: in this case, no password is required and the network key is not important.

Finding The IP Address Of Your Server

In order for the Arduino to send data to the server, it has to know the server's location on the network. Network locations are denoted with a number called the Internet Protocol or 'IP' address. This is a series of numbers that is unique for every machine on the network. Just like a postal address tells the postal service how to route a letter to your home, the IP address tells the network how to route information to your computer. Once your server is connected to the wireless network, follow the directions below to find your IP Address:

PC:

- a. Open the "Start" menu and click the "Control Panel" link. Click the "Network and Internet" section.
- b. Click on "Network and Sharing Center." Click the "View status" link in the "Connection" section.
- c. Click "Details" to see more information. In most cases, your IP address is listed next to "IPv4 IP Address."

Mac:

- a. Click the Apple logo in the upper left corner of the menu bar at the top of the screen.
- b. Click "System Preferences..."
- c. Click "Network" in the "Internet & Wireless" field. Your Mac's IP address is displayed under "Status."

Note: Normally, when a computer connects to a network, it is assigned an IP Address by the router. For the computer to function as the S4 Server, you may want to manually assign that value so it does not change. Follow the instructions on the Wiki page at <http://s4.sonoma.edu>. Always unplug the power from the flight board before connecting or disconnecting any of the other components on the board.

B – Setup Your Payload Hardware

Attach the WiFly shield to the flight board as shown in Figure 6.16



Figure 6.16: WiFly shield attached to the flight board

C - Running The Wifly_setup Sketch to Program Your WiFly Card for Your Network

The next step is to give the WiFly card the information it needs to connect to the network. This only needs to be done once, as the data will remain in the WiFly card's memory until it is overridden by rerunning the sketch. If at any time the payload needs to be given new network information (e.g., on the day of launch if you connect to a new network), this sketch will need to be run again.

Open the Arduino IDE and load this code automatically by selecting [File ▶ Sketchbook ▶ S4_WiFlySetup](#)

[Code]

```
#include <S4Wi-FiSetup.h>
#include "WiFly.h"

S4Wi-FiSetup S4Wi-FiSetup;

const char *device_ip_address = "192.168.1.2";
const char *server_ip_address = "192.168.1.66";
const char *wep_key = "0000000000000000";

void setup()
{
    Serial.begin(9600);

    S4Wi-FiSetup.begin(); // initiates
    S4Wi-FiSetup.setDCHP("1"); // 0 turns off DHCP; 1 turns on DHCP
    S4Wi-FiSetup.setIpGateway("192.168.1.1"); // sets gateway ip addresss
    S4Wi-FiSetup.setLocalPort("2000"); // sets port of that device connects too
    S4Wi-FiSetup.setWlanAuth("1"); // sets to WEP-128 key
    S4Wi-FiSetup.setDataRate("0"); // lowest data rate 1 M/sS for max range
    S4Wi-FiSetup.setDeviceName("SSU-01"); // device ID
    S4Wi-FiSetup.setRouterKey(wep_key); // WEP key
    S4Wi-FiSetup.setIpAddress(device_ip_address); // ip address of devices
    S4Wi-FiSetup.setHostAddresss(server_ip_address); // ip of computer that recives TCP or UDP connection
    S4Wi-FiSetup.setBroadcastAddress(server_ip_address); // ip of computer that recives broadcast message
    S4Wi-FiSetup.setBordcastInterval("0x1"); // broadcast every 2 seconda minumm
    S4Wi-FiSetup.setExternalAntenna("1"); //external antennae 1, chip antenna 0
    S4Wi-FiSetup.setFlushSize("1420"); // largest size for best TCP so 1420bytes at 9600 buad
    S4Wi-FiSetup.setProto("1"); //TCP proto 1; UDP proto 0
    S4Wi-FiSetup.setChannel("0"); // scans for the channel if 0; else 1-13 fixed
    S4Wi-FiSetup.save();
}

void loop()
{
    while(SpiSerial.available() > 0)
    {
        Serial.write(SpiSerial.read());
    }
    if(Serial.available())
    {
        SpiSerial.write(Serial.read());
    }
}
```

Figure 6.17: Arduino sketch that configures the WiFly device

There are several substitutions you must make to the code for proper function on your particular wireless network:

- Replace the *Server_IPAddress variable with your server's IP address (from part A).
- Replace the *NetworkKey variable with your networks security key (if you are using a secure network).

When you upload this sketch to the Arduino and open the Serial Monitor, you should get something similar to the output shown in Figure 6.18.

Seeing this type of output indicates that you have successfully loaded the information about your wireless network to the WiFly card. Later we will try actually connecting and sending data over the network.

[Output]

```
Setup 2.23, 04-26-2011 on 131C11
<2.23>
set ip dhcp 0
AOK
<2.23> set ip gateway 192.168.1.1
AOK
<2.23> set ip localport 2000
AOK
<2.23> set wlan auth 1
AOK
<2.23> set wlan rate 0
AOK
<2.23> set wlan key ac8afe4e5fb3931264f5df2f6f
AOK
<2.23> set ip address 192.168.1.3
AOK
<2.23> set ip host 192.168.1.2
AOK
<2.23> set broadcast address 192.168.1.2
AOK
<2.23> set broadcast interval 0x1
AOK
<2.23> set wlan ext_antenna 1
AOK
<2.23> set comm size 1420
AOK
<2.23> set ip protocol 1
AOK
<2.23> set wlan channel 0
AOK
<2.23> save
Storing in config
<2.23>
```

Figure 6.18: Output from the WiFly setup sketch

D - Running The S4 Server Program On Your Computer

The Arduino is designed to send data in real time to a server program, which saves the data to a database. This server program is written in a language called Java. You will need to install Java onto your computer for the program to run. Even if you already have Java installed, you may want to run through these instructions to ensure that you have the latest version so that the program will run. Follow the instructions on the Wiki page at: <http://s4.sonoma.edu/>.

Now that you have Java installed, you should be able to run the Server Program by opening the file LOCATION/TestingServer.jar from the S4 software package. This is a stripped down version of the server application that does not save the data to a database, but displays it on the screen. For instructions on setting up the standard Server for an actual launch see the Wiki page at: <http://s4.sonoma.edu/>.

Note: Firewall software on your machine can easily prevent the Server from being able to communicate with the payload. If you are having trouble communicating with the payload, try disabling your computer's Firewall.

E - Run the S4_WiFly Sketch to Test that Everything is Working Properly

Now we will run a sketch that connects to the network and sends a test message to our server. Open the Arduino IDE and load this code automatically by selecting [File ▶ Sketchbook ▶ S4_WiFly](#)

```
[Code]

/** Librarys of Code for the project **/
#include <S4.h>
#include <S4Sensor.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <WiFly.h>

S4 S4;

void setup()
{
  Serial.begin(9600);
  S4.useWiFi(true);
  S4.useGPS(false);
  S4.begin("PayloadName","NetworkName");
}

void loop()
{
  long data = 1;
  S4.addData("data",data);
  S4.writeData();
  delay(1000);
}
```

Figure 6.19: WiFly sketch that connects to the network

Substitute in the name you wish to give your payload, and the name you assigned to your network where you see "PayloadName" and "NetworkName."

Before uploading this sketch to the Arduino, double check and make sure that the TestingServer application is running on your server (see Part B). When you upload this sketch to the Arduino, the output should appear on the Testing Server (as shown in Figure 6.20) and Serial Monitor (as shown in Figure 6.21).

```
[Output]

Terminal Started...
Payload Connected!
Sensor Data Received !
Sensor Data Received !
Sensor Data Received !
Sensor Data Received !
Sensor Data Received !
Sensor Data Received !
Sensor Data Received !
Sensor Data Received !
Sensor Data Received !
```

Figure 6.20: Output from WiFly on the Testing Server

```
[Output]

|<sensor>PayloadName,data,1
<sensor>PayloadName,data,1
<sensor>PayloadName,data,1
<sensor>PayloadName,data,1
<sensor>PayloadName,data,1
<sensor>PayloadName,data,1
<sensor>PayloadName,data,1
<sensor>PayloadName,data,1
<sensor>PayloadName,data,1
<sensor>PayloadName,data,1
```

Figure 6.21: Output shown on the Serial Monitor

6.6 Coding and Testing Sensors

Now that you have assembled, coded and tested the base components on the flight board, the next step is to get the payload to take measurements using a particular sensor. To do this, we use code that reads the current data value from the sensor, and writes that value to the serial buffer to be read on our computer. Of course, the flight version of the code writes the data values to both the Wi-Fi card and to the SD card; but for now we want to simply test that the values are being read in the laboratory. This is usually called a “ground systems test.”

For all the examples below, the instructions for testing the devices assume that they are mounted on the flight board. However, there is no reason that students could not do this testing with the components mounted on breadboards connected to the Arduino with jumper wires (if the flight board is not available or you do not wish to use it). In each example, we will assume that the Arduino and the device we are testing are the only items attached to the flight board. The code will work the same regardless of what devices (other than those being mentioned in the instructions) may or may not be attached to the board.

6.6.1 Accelerometer Code and Test

In order to program and test the accelerometer:

Step 1: Attach the accelerometer to the flight board as shown in Figure 6.22.



Figure 6.22: Accelerometer attached to the flight board

Step 2: Open the Arduino IDE and load this code automatically by selecting [File](#) ► [Sketchbook](#) ► [S4_Accelerometer](#)

```
[Code]

/* *** DISPLAYS OF CODE FOR THE PROJECT *** */
#include <S4.h>
#include <S4Sensor.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <WiFly.h>

S4Sensor S4Sensor;

/* ****Sensor Variables **** */
double accelX;
double accelY;
double accelZ;

void setup()
{
  Serial.begin(9600);
  S4Sensor.startAccelerometer(); // initiate the accelerometer
}

void loop()
{
  S4Sensor.getAccelerometer(accelX, accelY, accelZ); // fill the variables with new data
  Serial.print("Accel X: ");
  Serial.print(accelX);
  Serial.print("    Accel Y: ");
  Serial.print(accelY);
  Serial.print("    Accel Z: ");
  Serial.println(accelZ);
  delay(100); // delay 100 ms
}
```

Figure 6.23: Arduino sketch that runs the accelerometer sensor

Step 3: Upload the sketch to the Arduino and then open the Serial Monitor to watch the data streaming in from the device. The data should look similar to those shown in Figure 6.24.

The units of acceleration are g's, where 1 g is the acceleration due to gravity you experience at the Earth's surface. You may notice that even if the device is sitting flat on the table it will still measure the acceleration due to gravity. You should also notice that even if the device is not moving, there are some small fluctuations in the value that it reads. Try moving the device around (but remember that it is still connected to your computer) and watching the values it displays on the Serial Monitor.

6.6.2 Magnetometer Code and Test

In order to program and test the magnetometer:

Step 1: Attach the magnetometer to the flight board as shown in Figure 6.25.

Step 2: Open the Arduino IDE and load this code automatically by selecting **File** ► **Sketchbook** ► **S4_Magnetometer**

[Output]		
Accel X: -0.01g	Accel Y: -0.61g	Accel Z: 0.74g
Accel X: -0.02g	Accel Y: -0.62g	Accel Z: 0.75g
Accel X: -0.02g	Accel Y: -0.62g	Accel Z: 0.75g
Accel X: -0.02g	Accel Y: -0.62g	Accel Z: 0.74g
Accel X: -0.01g	Accel Y: -0.61g	Accel Z: 0.74g
Accel X: -0.01g	Accel Y: -0.61g	Accel Z: 0.75g
Accel X: -0.02g	Accel Y: -0.61g	Accel Z: 0.75g
Accel X: -0.01g	Accel Y: -0.62g	Accel Z: 0.74g

Figure 6.24: Sample data from the accelerometer sensor



Figure 6.25: Magnetometer attached to the flight board

[Code]
<pre>/** Librarys of Code for the project **/ #include <S4.h> #include <S4Sensor.h> #include <SoftwareSerial.h> #include <Wire.h> #include <WiFly.h> S4Sensor S4Sensor; /*Note the readout returns 10x microTesla ****Sensor Variables ****/ int magX; int magY; int magZ; void setup() { Serial.begin(9600); S4Sensor.startMagnetometer(); // initiate the Magnetometer Serial.println('start'); } void loop() { S4Sensor.getMagnetometer(magX,magY,magZ); // fill the variables with new data Serial.print("Mag X: "); Serial.print(magX); Serial.print(" x10^-7 Tesla Mag Y: "); Serial.print(magY); Serial.print(" x10^-7 Tesla Mag Z: "); Serial.print(magZ); Serial.println(" x10^-7 Tesla"); }</pre>

Figure 6.26: Arduino sketch for the magnetometer

Step 3: Upload the sketch to the Arduino and then open the Serial Monitor to watch the data streaming in from the device. The data should look similar to those shown in Figure 6.27.

[Output]					
Mag X: -403	x10 ⁻⁷ Tesla	Mag Y: 27	x10 ⁻⁷ Tesla	Mag Z: 26	x10 ⁻⁷ Tesla
Mag X: -402	x10 ⁻⁷ Tesla	Mag Y: 30	x10 ⁻⁷ Tesla	Mag Z: 23	x10 ⁻⁷ Tesla
Mag X: -401	x10 ⁻⁷ Tesla	Mag Y: 29	x10 ⁻⁷ Tesla	Mag Z: 25	x10 ⁻⁷ Tesla
Mag X: -400	x10 ⁻⁷ Tesla	Mag Y: 27	x10 ⁻⁷ Tesla	Mag Z: 25	x10 ⁻⁷ Tesla
Mag X: -403	x10 ⁻⁷ Tesla	Mag Y: 29	x10 ⁻⁷ Tesla	Mag Z: 26	x10 ⁻⁷ Tesla
Mag X: -403	x10 ⁻⁷ Tesla	Mag Y: 29	x10 ⁻⁷ Tesla	Mag Z: 24	x10 ⁻⁷ Tesla
Mag X: -405	x10 ⁻⁷ Tesla	Mag Y: 28	x10 ⁻⁷ Tesla	Mag Z: 24	x10 ⁻⁷ Tesla
Mag X: -402	x10 ⁻⁷ Tesla	Mag Y: 27	x10 ⁻⁷ Tesla	Mag Z: 26	x10 ⁻⁷ Tesla
Mag X: -407	x10 ⁻⁷ Tesla	Mag Y: 24	x10 ⁻⁷ Tesla	Mag Z: 24	x10 ⁻⁷ Tesla
Mag X: -402	x10 ⁻⁷ Tesla	Mag Y: 30	x10 ⁻⁷ Tesla	Mag Z: 25	x10 ⁻⁷ Tesla

Figure 6.27: Sample data from the magnetometer

The units of magnetic field are 10⁻⁷ Teslas where 1 Tesla = 10⁴ Gauss, and the total magnetic field is roughly 0.5 Gauss at the Earth's surface.

6.6.3 Barometric Pressure/Temperature Sensor Code and Test

In order to program and test the barometric pressure/temperature sensor:

Step 1: Attach the barometric pressure/temperature sensor to the flight board as shown in Figure 6.28

Step 2: Open the Arduino IDE and load this code automatically by selecting **File** ► **Sketchbook** ► **S4_Barometric**



Figure 6.28: Barometric pressure/temperature sensor attached to flight board

```

[Code]

/** Librarys of Code for the project **/
#include <S4.h>
#include <S4Sensor.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <WiFly.h>

S4Sensor S4Sensor;

/****Sensor Variables ****/
double temperature;
long pressure;

void setup()
{
  Serial.begin(9600);
  S4Sensor.startBarometric();
}

void loop()
{
  S4Sensor.getBarometric(pressure,temperature);
  Serial.print("Pressure: ");
  Serial.print(pressure);
  Serial.print("    Temperature: ");
  Serial.println(temperature);
  delay(500);
}

```

Figure 6.29: Arduino sketch for the barometric pressure/temperature sensor

Step 3: Upload the sketch to the Arduino and then open the Serial Monitor to watch the data streaming in from the device. The data should look similar to those shown below:

[Output]	
Pressure: 100602 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100602 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100598 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100599 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100595 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100601 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100602 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100611 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100599 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100606 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100604 Pascals	Temperature: 24.00 degrees Celsius
Pressure: 100598 Pascals	Temperature: 24.00 degrees Celsius

Figure 6.30: Sample data from the barometric pressure/temperature sensor

Note that the pressure readings are given Pascals (where 1 Pascal = 1 Newton/m²). Earth's standard atmospheric pressure at the surface is 101,325 Pa. The temperature readings are in degrees Celsius.

6.6.4 Humidity/Temperature Sensor Code and Test

In order to program and test the humidity/temperature sensor:

Step 1: Attach the humidity/temperature sensor to the flight board as shown in Figure 6.31

Step 2: Open the Arduino IDE and load this code automatically by selecting [File](#) ▶ [Sketchbook](#) ▶ [S4_Humidity](#)

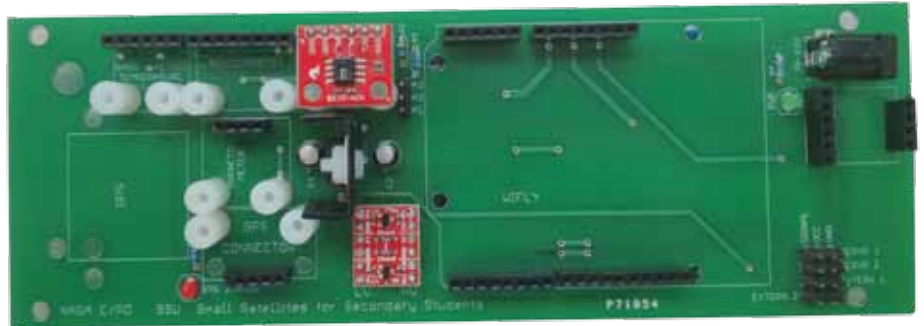


Figure 6.31: Humidity/temperature sensor attached to flight board

[Code]
<pre> /** Librarys of Code for the project **/ #include <S4.h> #include <S4Sensor.h> #include <SoftwareSerial.h> #include <Wire.h> #include <WiFly.h> S4Sensor S4Sensor; /****Sensor Variables ****/ double humidity; double temperature; void setup() { Serial.begin(9600); S4Sensor.startHumidity();// initiate the Humidity detector } void loop() { S4Sensor.getHumidity(humidity,temperature); // fill the variables with new data Serial.print("Humidity: "); Serial.print(humidity); Serial.print(" Temperature: "); Serial.println(temperature); delay(500); // delay 500 ms (0.5 seconds) } </pre>

Figure 6.32: Arduino sketch for the humidity/temperature sensor

Step 3: Upload the sketch to the Arduino and then open the Serial Monitor to watch the data streaming in from the device, as shown in Figure 6.33.

[Output]	
Humidity: 61.93 relative %	Temperature: 23.66 degrees Celsius
Humidity: 61.93 relative %	Temperature: 23.66 degrees Celsius
Humidity: 61.93 relative %	Temperature: 23.68 degrees Celsius
Humidity: 61.93 relative %	Temperature: 23.69 degrees Celsius
Humidity: 61.93 relative %	Temperature: 23.68 degrees Celsius
Humidity: 61.93 relative %	Temperature: 23.69 degrees Celsius
Humidity: 61.93 relative %	Temperature: 23.69 degrees Celsius
Humidity: 61.90 relative %	Temperature: 23.69 degrees Celsius
Humidity: 61.90 relative %	Temperature: 23.69 degrees Celsius
Humidity: 61.88 relative %	Temperature: 23.69 degrees Celsius
Humidity: 61.88 relative %	Temperature: 23.69 degrees Celsius
Humidity: 61.86 relative %	Temperature: 23.69 degrees Celsius
Humidity: 61.86 relative %	Temperature: 23.69 degrees Celsius

Figure 6.33: Sample data from the humidity/temperature sensor

Note that the humidity readings are relative, and are given in percentages from 10-90%, while the temperature readings range from 5-50 degrees Celsius.

6.6.5 Testing All Sensors

Now that you have tested the sensors individually, we can run the code that tests them all together. This is the code that you will actually run when you are using your payload to acquire data in the field. Load the S4_AllSensors Sketch: [File](#) ► [Sketchbook](#) ► [S4_AllSensors](#)

Note that in the Setup() function, by default, useWiFi and useGPS are both set the false. If you are planning on using either of these features , make sure that you set them to true. Also note that by default, the payload name and network name in the S4.being() function are set to default values, you will need to put in the correct values, as you did in section 6.5.3E (WiFi testing) for these properties.

Figure 6.34: Sample data from all sensors

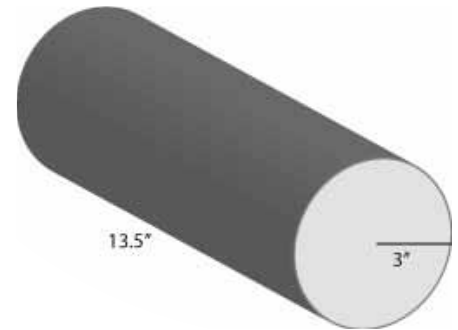
[Code]
<pre> ****Sensor Variables *** double temperature; long pressure; double humidity; double temperature2; double accelX; double accelY; double accelZ; int magX; int magY; int magZ; void setup() { Serial.begin(9600); S4.useWiFi(false); S4.useGPS(false); S4.begin("SSU-01","DeviceTesting"); S4Sensor.startBarometric(); S4Sensor.startHumidity(); S4Sensor.startAccelerometer(); S4Sensor.startMagnetometer(); } void loop() { S4Sensor.getBarometric(pressure,temperature); S4Sensor.getHumidity(humidity,temperature2); S4Sensor.getAccelerometer(accelX,accelY,accelZ); S4Sensor.getMagnetometer(magX,magY,magZ); S4.addData("Baro",pressure); S4.addData("Temp",temperature,2); S4.addData("Hum", humidity,2); S4.addData("Temp2",temperature2,2); S4.addData("AcelX",accelX,4); S4.addData("AcelY",accelY,4); S4.addData("AcelZ",accelZ,4); S4.addData("MagX",magX); S4.addData("MagY",magY); S4.addData("MagZ",magZ); S4.writeData(); } </pre>

7 Customizing the S4 Payload Design

7.1 Designing Experiments With the S4 Flight Board

7.1.1 Size, Mass and Power for the S4 Payload

The S4 payload that is described in this guide fits within a cylindrical payload bay of radius 3" and height 13.5". Power is provided by a 9-V battery. The mass of the payload is a maximum of 350 g, including all four optional sensors (accelerometer, barometric pressure/temperature, humidity/temperature and magnetometer and the 9-V battery).



7.1.2 Standard Measurements With the S4 Flight Board and Sensors

The first step for each group of students is to decide what type of experiment to perform. Most students will not have clear ideas as what types of measurements are possible, and the difficulty level of the experiment that is proposed. The following examples are all good starting points from which to base their experiments. Below we provide a (non-exhaustive) list of experimental quantities that may be readily measured by the sensors and/or base

Warning!
Experiments that are intended to be flammable, explosive, or otherwise harmful to persons, property, or flora and fauna, are *not allowed*.

components on the S4 flight board, in combination with the software included in the S4 Arduino library files. The flight board has been designed to be easily extended to work with additional sensors as well as to control a servo for mechanical operations. Many other sensors and operations are available for Arduinos, and the Arduino community has developed software that can be adapted to these uses. See Appendix C for additional Arduino resources.

Acceleration

Acceleration can be experimentally determined with data from the on-board GPS chip or from the accelerometer sensor. The GPS chip returns position vs. time data, which can be analyzed to compute acceleration. The accelerometer returns three-axis values for the acceleration in units of the local "g" (9.8 m/sec²) force on the Earth's surface. The total acceleration is given by:

$$a_{\text{tot}} = (a_x^2 + a_y^2 + a_z^2)^{1/2}$$

Possible experiments:

- Measure and plot acceleration vs. time
- Detect engine cutoff in a rocket flight
- Compare payload acceleration calculated from GPS data to the readouts from the accelerometer

Altitude

Altitude can be measured using data from the on-board GPS chip or from the barometric pressure/temperature sensor. Tests with the GPS chip have shown that it is accurate within 10 meters when determining altitude. The barometric pressure sensor returns values in Pascals (Pa), which can be referenced to standard atmospheric pressure (101325 Pa, where 1 Pa = 1 N/m²). For every 300 meters (\approx 1,000 feet) one ascends, the atmospheric pressure decreases by about 4%.

Possible experiments:

- Measure and plot altitude vs. time
- Compare altitude values from the GPS chip to the readouts from the barometric pressure sensor
- Compare altitude values from the barometric pressure sensor to the altitude measured by triangulating the balloon position
- Measure and plot other quantities such as temperature, magnetic field, etc. vs. altitude

Humidity

Relative humidity is measured using the humidity/temperature sensor. The relative humidity values are given in percentages from 10-90%. Recalibration of the sensor can be achieved by simply leaving the sensor at room temperature in ambient conditions for at least five hours. The temperature sensor is built in to allow the humidity sensor to correctly compensate for changing temperatures, while also providing an independent measurement of the temperature.

Possible experiments:

- Determine if there is a change in humidity between different times of the day, a month, a year, etc.
- Measure and plot relative humidity vs. temperature
- Measure and plot relative humidity vs. altitude

Magnetic Field

The magnetometer sensor measures the magnetic field strength, in mG (milli Gauss, $10 \text{ mG} = 1 \mu\text{T}$) in three dimensions. Magnetometer data can be combined with accelerometer data to provide the orientation of the experiment with respect to magnetic north. Magnetometer data can be combined with GPS data to provide the magnetic field orientation and strength at different positions on the Earth.

Possible experiments:

- Measurements of Earth's magnetic field strength or orientation at different ground positions
- Measurements of the Earth's magnetic field at different times of day, month, year (then correlate with solar activity)
- Measurement of the orientation of the experiment with respect to magnetic north

Position

The on-board GPS chip provides longitude, latitude and altitude as a function of time.

The readouts occur every 1 second and are in the following format (See Section 6.5.1):

\$GPGGA, 210044.00, 3820.39934, N, 12240.61207, W, 1, 06, 1.78, 55.0, M, -29.2, M,, %5B

In the above string, we use the following entries:

- 210044.00 – this is the UTC time in HHMMSS.SS. The time is therefore: 21 hours, 00 minutes, and 44.00 seconds.
- 3820.39934, N – this is the longitude in DDMM.MMMMM. The longitude is therefore: 38 degrees, 20.39934 minutes North.
- 12240.61207, W – this is the latitude in DDDMM.MMMMM. The latitude is therefore: 122 degrees, 40.61207 minutes West.
- 55.0, M – this is the altitude above mean sea level in meters.

Possible experiments:

- Independent verification of the rocket apogee altitude compared to the rocket software simulation's estimates
- Determine wind velocity at various altitudes
- Plot the GPS position and altitude as a function of time
- Compare the GPS altitude to the altitude from the barometric pressure sensor

Pressure

The barometric pressure sensor returns values in Pascals (Pa), which can be referenced to standard atmospheric pressure (101325 Pa, where $1 \text{ Pa} = 1 \text{ N/m}^2$). The barometric pressure sensor corrects for temperature, and can also be used to determine altitude (see above). The sensor returns an independent temperature measurement.

Possible Experiments:

- Measure and plot the pressure vs. time
- Measure and plot the pressure vs. temperature
- Use pressure to compute altitude and compare to altitude determined using other means

Temperature

Both the barometric pressure and humidity sensors have built-in, independent temperature readouts. Temperature is easy to measure but may not change very much throughout the short duration of a rocket flight.

Possible experiments:

- Measure and plot temperature vs. time
- Measure and plot temperature vs. altitude
- Measure and plot temperature vs. pressure
- Measure and plot temperature vs. altitude

7.1.3 Other Possible Measurements for Advanced Students

There is a wide variety of other sensors available that can be interfaced to the S4 flight board and read out by Arduinos. In this section we provide some additional examples for more advanced students to consider.

Biological Experiments

Due to the high accelerations experienced on rockets, launching live vertebrate animals is prohibited by rocketry organizations. It is also not a good idea to conduct experiments with the goal of injuring or killing the specimens. There are some balloon experiments where gentle observations of insects or other types of creatures are acceptable, however.

Cosmic Rays

Cosmic rays are charged, high-velocity particles that emanate from outside the Earth's atmosphere. Often the primary cosmic rays interact with the atmosphere to create secondary cosmic rays. The flux of cosmic rays increases with altitude, as was discovered in early balloon flights by Victor Hess (see Section 2.1). The size of rocket payloads and short duration of rocket flights provide a challenge for detection of cosmic rays – hence, balloon payloads are better suited for this type of measurement. Geiger counters are one type of cosmic ray detector, that will also detect electromagnetic radiation, such as X-rays or gamma rays.

Gases

Different gas sensors available include hydrogen, carbon monoxide, alcohol gas and methane.

Particulates

Particulate detectors can be used to measure dust in the air or other types of macroscopic pollutants. One possible detector uses an infrared LED to scatter off the dust in the air with a phototransistor as the sensor.

Servo-Controlled Experiments

Although the S4 flight board includes the controls for a servo, the design of experiments that use this device to actuate a mechanical release or other sensor are beyond the scope of this project.

7.1.4 Rocket or Balloon?

There are several benefits and constraints that differentiate rocket and balloon-based experiments. For rocket payloads, the mass of the payload is limited, as is the volume within which the payload can be contained. High-powered rockets need certified flyers to purchase the motors, and special waivers from the FAA to conduct launch events. Additionally, the short duration of a rocket's flight limits the amount of data that experiments can acquire.

Balloons provide for longer duration flights, and greater payload masses and volumes. Tethered balloons provide repeated measurements of atmospheric conditions at specific, predetermined altitudes, or as a function of altitude up to the length of the tether (1,000 ft. in most cases). Typical payload mass limits are 4 kg. Meteorological balloons require approximately 100 to 150 ft³ of helium to lift a 1.4 to 2.75 kg (3 to 6 lbs) payload.

There are two sizes of meteorological balloons that are in standard use for student experiments: 300-g balloons, which have a standard inflation diameter of 2.4 m (8 ft) and a burst diameter of 3.3 m (11 ft); and 200-g balloons, which have a standard inflation diameter of 2.0 m (6.5 ft) and a burst diameter of 2.4 m (8 ft). The 200-g balloons are better suited for low-weight gondolas and low-altitude experiments.

It is also possible to do untethered meteorological balloon launches. Untethered balloons remain airborne for a long time, which allows for larger data sets gathered per experiment. However, it is often difficult to retrieve the payloads as the on-board GPS may not be received by the ground station throughout the duration of the flight. Tracking the payload using Ham radio frequencies will extend the range and increase the chance of finding the payload. Even so, there remains the possibility of having to hike into wilderness areas in order to retrieve one's experiment, once it lands.

Some pros and cons of rocket vs. balloon experiments are summarized in Table 7.1 below.

Table 7.1: Rockets and Balloons: Pros and Cons			
Balloons (tethered)		Rockets	
Pro	Con	Pro	Con
Payload dimensions are not limited, only mass.	Payload mass limited to 5.44 kg (12 lb or 4 kg without FAA approval) where no single piece has a mass over 2.72 kg (6 lb).	Payload mass only limited by motor and rocket size, but volume is limited.	Need certified flyers and approved locations for high power flights.
Relatively inexpensive vehicle costs. Reusable parts.			High-powered rockets and motors are fairly expensive
Easy recovery.	Maximum altitude limited to 1000 ft.	Maximum altitude limited only by motor size.	Possibility of difficult recovery.
Rarely experiences catastrophic failure.			Slight but real possibility of catastrophic failure.
Multiple flights can be done using a single balloon.	Helium supply is uncertain and may be costly.		
Variable flight durations and altitude profiles.			

7.2 Rocket Payload Requirements

The S4 payload will easily fit within a 4-inch diameter rocket, but may also fit in a 3-inch diameter rocket, if there is adequate space in the payload bay. The payload contains an antenna which optimally should extend away from the flight board, but can be folded back over on itself if the length is a problem. In this configuration the payload could be fit into a cylindrical volume with a 3 inch radius and 8 inch length. The two antenna configurations are shown in Figure 7.1 below.

The payload is designed to be mounted in a bay that has rails running along the sides as shown in Figure 7.2 below. The spacing on the rail should be about 3 inches. This configuration allows the flight board to be tied securely to the rails using zip-ties through the holes in the four corners of the board. Alternatively, for 3-inch diameter rockets, the payload can be securely wrapped in foam and placed in the payload bay for launch.

Figure 7.1: The antenna extended away from the flight board (top), and folded back into a more compact configuration (bottom).

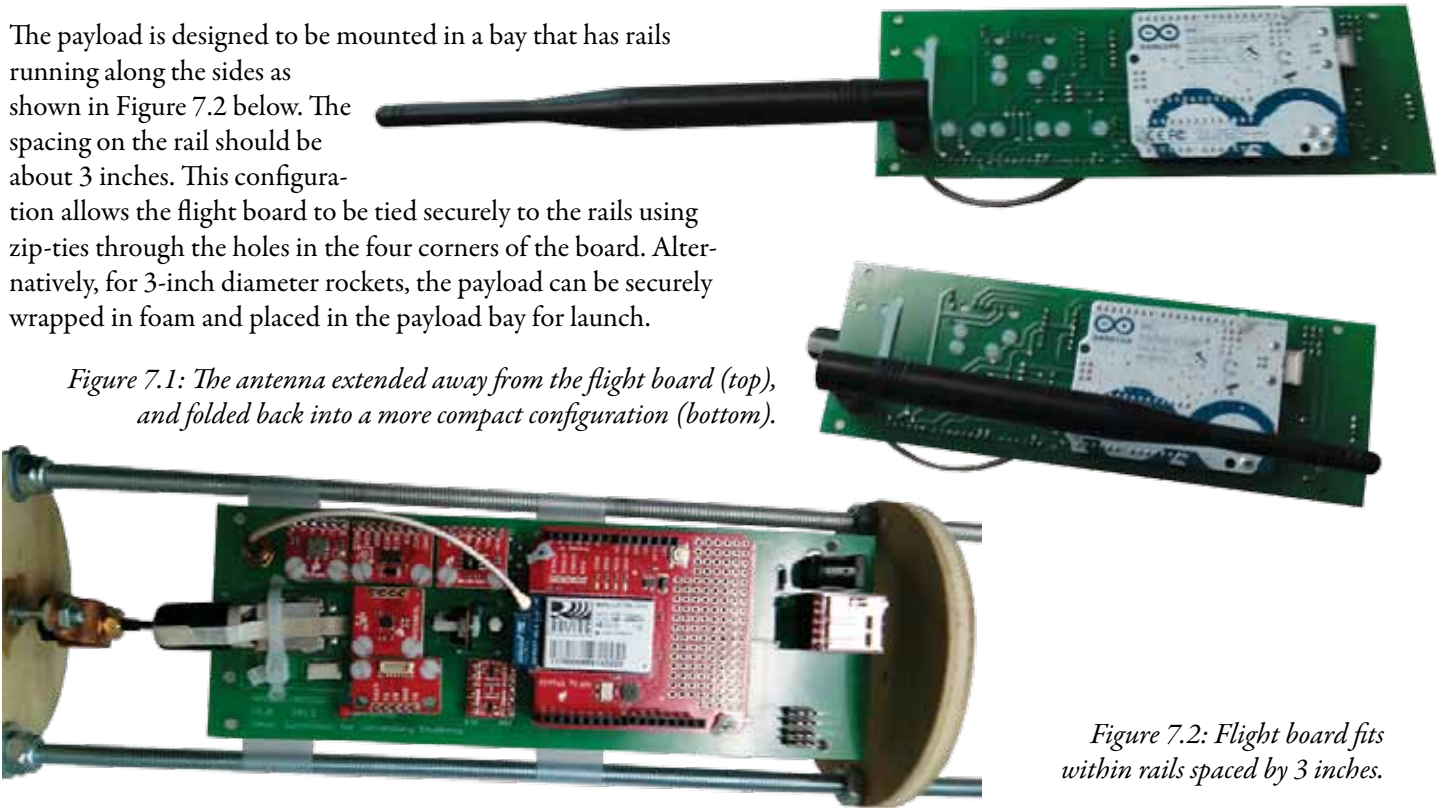


Figure 7.2: Flight board fits within rails spaced by 3 inches.

Other Considerations for Rocket Launches:

- The payload is not ejected from the airframe and does not come down on its own chute. The airframe should therefore be designed so that the payload bay comes down at a reasonably slow speed in order to protect the payload on landing.
- Since the payload communicates wirelessly with the ground station, carbon fiber bodies will interfere with the signal and cannot be used.
- The payload does not need to expose any sensors or the antenna to the outside air (at this time). It also does not need any type of external power holes through which to flip switches. The device is low power, and can be communicated with wirelessly to confirm proper functionality before launch.
- A bulk head of some kind should be used to protect the payload from the separation charge.

For a rocket motor with a given thrust, lighter payloads will reach higher altitudes. A good rule is that the maximum total weight of the rocket (including motor and payload) should not exceed a 5:1 thrust to weight ratio. Check with the high-powered rocket flyer to make sure that the 350 g (maximum) S4 payload mass has been taken into account in the simulations of the rocket flight. Also, the location of the payload should be accurately specified in calculations of the rocket's center of gravity.

7.3 Balloon Payload Requirements

The carrier for a balloon payload is known as a “gondola.” Students will need to design and build gondolas to carry their customized S4 payloads. Gondolas serve as protection for the payload against direct contact with the balloon's tether (and possibly the balloon itself if wind speeds become severe).

The gondola can be attached to the S4 payload through holes in the four corners of the flight board with zip-ties. The gondola is then attached to the balloon with a D-ring, or a carabiner, which in turn is tied securely to the rope attached to the balloon.

When constructing the gondola, students should use sturdy materials while trying to minimize the weight. The gondola should form a box or cage around the payload to ensure that it can't “swing” in the wind where it risks being slapped against the tethered line or the base of the balloon.

Special FAA Clearance is Not Needed for Tethered Balloon Launches as Long as:

- the launch site is farther than 5 miles from the nearest airport
- permission of land owner has been granted
- the balloon diameters are less than six feet (120 cu. ft. gas volume)
- payloads (including the gondola & tether) weigh less than 4 kg (8.8 lb)
- the maximum tether length is less than 1000 ft AGL (above ground level)

7.4 Flight Project Documentation Requirements

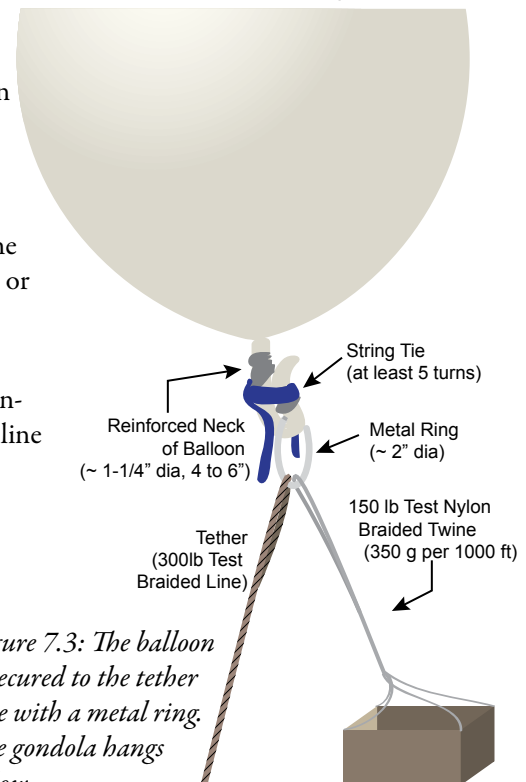
NASA flight projects typically include a series of reviews that ensure that the experiment meets its design requirements, can be safely flown and recovered, will withstand the rigors of launch and the conditions aloft, and will return data that will accomplish the experimental goals. Additionally, these tasks must all be accomplished within the constraints of budget and schedule. Depending on the complexity of the project, these reviews can include: Preliminary Design Review, Critical Design Review, Flight Readiness Review. In Appendix B, we provide a link to the NASA Student Launch Initiative website which includes the requirements for PDR, CDR and FRR. Take into account the launch date, projected weather and wind speeds. Wind greater than 5 mph will typically limit flights.

For S4 student programs, you will probably want to simplify this formal process, while retaining some of the more important elements. In Appendix B, we provide a link to the Balloon Fest website which has model documents for students to use to describe their Designs, Experimental Validation, Procedures, Flight Log, and final Presentations. In the following sections, we follow this model for Flight Project documentation requirements.

7.4.1 Design

The design document should include:

- a brief summary or proposal of the experiment
- the type of launch vehicle to be used
- at least one hypothesis to be tested



The instructor should approve the experimental design before proceeding to the next step.

- the proposed method to test the hypothesis
- a list of parts, including approximate sizes and weights,
- (optional) an outline for any new Arduino software that may be needed

7.4.2 Experimental Validation

In this phase, the students should indicate the procedure they will use to test their experiment on the ground, in order to show that the sensors are working properly, and are capable of obtaining the data required to test their hypothesis.

Students should be taking notes in lab notebooks at this phase; documenting their hypotheses, their expected results, all materials used, their planned procedure, and then recording the results of the experimental validation (ground system) tests.

For rocket flights, students should work with flyers to simulate the flight using a program such as RockSim that includes the model of the rocket that will be used, the motor size, ejection delay, and correct center of gravity including the payload.

For balloon flights, students should include the specifics of the gondola design to be used, and describe the expected balloon altitude profile as a function of time.

These plans should be updated if the payload changes. Changes in design are common but it is important to document the changes as they occur in the lab notebook, so that accurate information is available later for the data analysis phase.

7.4.3 Procedures

The procedure document should include the purpose of the experiment, a complete list of the equipment to be used, a description of the payload, including mass, the steps needed to fly the payload, to obtain data from the payload, and to analyze data from the payload.

7.4.4 Flight Log

The flight log includes the detailed notes taken during the flight. It can be as simple as checking off the steps in the procedure document, or it can be more complicated as the procedures evolve to reflect the reality of the flight dynamics.

7.5 At the Launch Site

Sponsoring organizations have rules and regulations at their launch sites, whether for balloons or rockets. Rocket safety regulations are strongly enforced, especially with high-powered rocketry. Make sure you and your students understand these regulations before arriving at the site.

It is also very important for the payload to be checked one final time before launch. Optimally, the checklist should be reviewed three times; once before it leaves for the launch site, once upon arrival to the site, and lastly right before the payload is to be launched. Having one or two students visually check the payload as another one reads off the checklist will help to ensure that they remember important things like their tools and fresh power supplies. The list below is a good starting point for the S4 base payload:

- Acceleration test has been passed
- Fresh Batteries
- PCB lead connections checked
- Power (red) LED is active
- Test Data have been acquired and saved from the WiFi (if used)
- Test Data have been saved and retrieved from the SD card (if used)
- Zip-ties are securely fastened to avionics bay/gondola
- Zip-ties are securely fastened to payload

Students should also use a stopwatch or a clock, synchronized with the payload being turned on for the experiment. A student should keep track of when events happen such as; at what time the payload launched, when the payload hit apogee and is starting to come down, etc. This will help students identify where, in their data, they should start looking for the data they want.

Payload recovery for rockets can be a difficult and time consuming process depending on where the rocket finally lands. Educators should supervise students if they take part in recovering the rocket as it will often require walking through fields, hills, and even in-use ranchland. Depending on wind conditions, the size of the parachute, and the height of apogee of the rocket, the payload can end up being several thousand feet to a mile or more away from the launch site.

Tethered balloons are far easier to recover as the students merely have to reel in the tether and the payload comes right back to them.

In either case, students should check over their payload when they recover it to ensure that no damage has occurred.

8 Experimental Data



8.1 Acquiring and Accessing Payload Data

8.1.1 Acquiring the Payload Data

The Virtual Classroom

The Virtual Classroom (VC) is a converted mobile television van that has the capability to stream WiFi telemetry data from the Payload through satellites to the Internet (and then to your classroom) in near real-time. It is ideally suited for live telemetry download from remote locations such as high-powered rocketry launch sites (where normal Internet service is not readily available). The VC also runs video cameras which record the launches, and can serve as the interface to the internet for other on-site computers. All the data sent through the VC from the launch site flow through the S4 server at Sonoma State University to other Internet clients, such as classrooms. Figure 8.1 below illustrates the network setup using the VC.

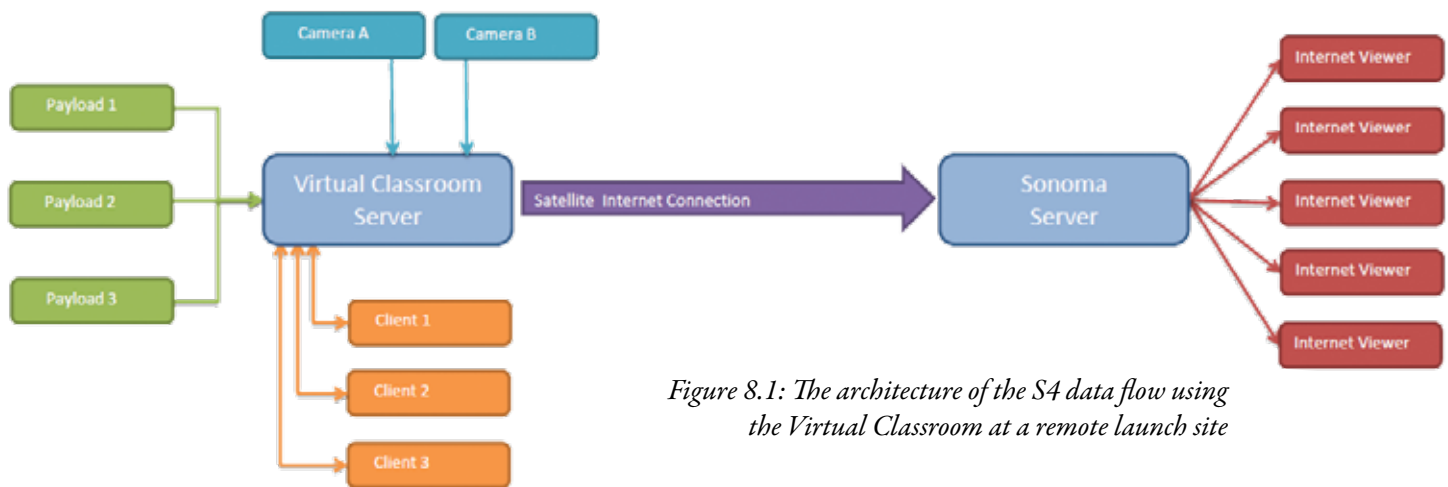


Figure 8.1: The architecture of the S4 data flow using the Virtual Classroom at a remote launch site

On-Site WiFi

The on-board WiFly chip combined with a local, off-the-shelf router at the launch site, allows WiFi enabled computers to receive the real-time data from the operable payloads without the VC. This has been tested with tethered balloon flights up to 1000 feet, and rocket flights up to ~3,000 ft. This allows teachers and students to easily test their payloads on balloons or in the classroom.

8.1.2 Accessing the Data

The data obtained through the Virtual classroom are saved in a MySQL database on the S4 Server at Sonoma State University.

Server Information

- Standard IP Address: 130.157.169.39
- Standard Port Number: 2003

If you have used a local router for the data, it will end up on your computer in a MySQL database that you can set up using the provided software.

In either case, once the experiment is over the data can be accessed through the use of the S4DataExporter tool which allows the raw text file of data to be converted to either a .csv or a .kml file.

Open the S4DataExporter tool on your computer. Once it is open, select “Import,” then “From File,” and then navigate to the folder where the data txt file is stored. Or, select “Import,” the “From Database,” and in the window that pops up enter in the IP Address and Port of the server (see below) and then select your payload’s data file from the available list of data files.

Once the file is imported into the S4DataExporter you will now need to export it as one of two different file types: .csv or .kml.

CSV File

A .csv file is a “comma-separate values” file. In the .csv format, data are recorded with commas separating the different values. For example, the GPS data are written as:

\$GPGGA, 210044.00, 3820.39934, N, 12240.61207, ...

So after each data point there is a comma that indicates a new field of information. In order to analyze the data, they must first be imported into a spreadsheet program.

Microsoft Excel and OpenOffice Calc programs will recognize the .csv file format. This chapter assumes the use of Excel (although the two are very similar). You can double click on the .csv file and Excel will open it. Each column will have its data, listed in chronological order and will include header row titles.

KML File

A .kml file is a “keyhole markup language” file. This is the type of file format that is used in geographic-map software such as Google Earth. The .kml file includes information, such as latitude, longitude, and altitude, which are graphically represented using (for example) Google Earth software.

8.2 Analyzing Payload Data

Either before or after the data are in the spreadsheet, students may wish to remove (delete) the beginning and/or ending data of the file because it does not correspond to the actual beginning and/or ending time of the experiment. For example, the time from turning on the payload to the actual rocket launch may be several to tens of minutes while waiting for other rockets launch, and the payload will continue taking data while waiting for the launch to occur. A rocket waiting on the launch pad with a GPS lock for 8 minutes, will generate nearly 500 lines of data which may not be particularly interesting for the experiment. Additionally, until the payload is turned off, it will continue to transmit data which may not be very interesting (once the final GPS location of the payload has been obtained). Alternatively, you can use a spreadsheet to subtract the actual starting time from all the data points, so that the graph will display time equal to zero when the experiment actually began. (This is what has been done in Figure 8.2 below.)

Once data are imported into a spreadsheet program, students will be able to analyze the data by graphing them and determining if the data fit a trendline, using graphs made as “scatter plots.” Note that current versions of Open Office Calc are only able to perform linear fitting trendlines (though there may be add ons that have been independently created to increase the functionality of this software). Microsoft Excel is able to fit linear, polynomial, as well as logarithmic trendlines to the data.

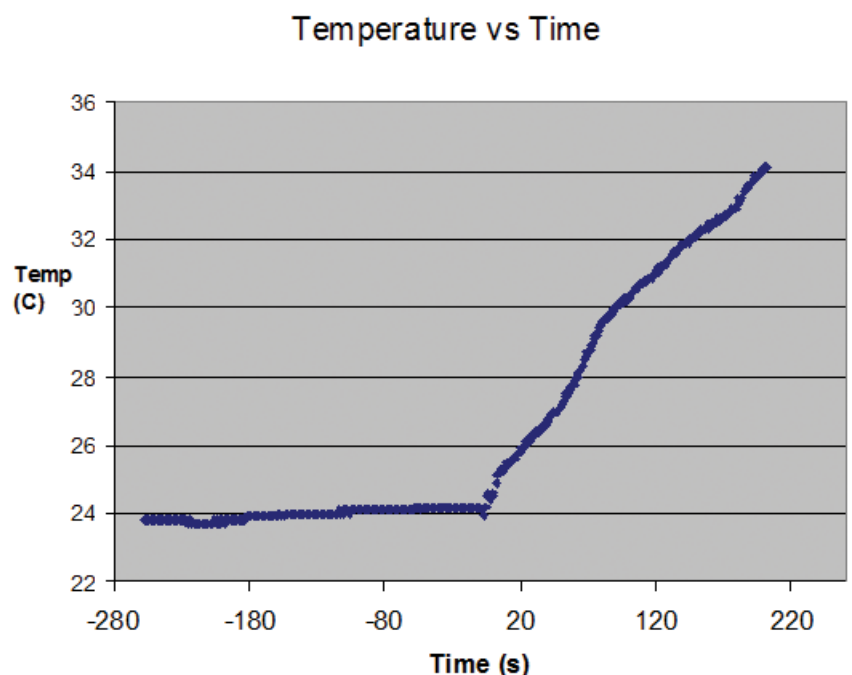
8.2.1 Displaying and Analyzing Payload Data (Excel)

To graph the data, first select the “chart wizard,” icon from the toolbar in Excel, or select “Insert,” then “Chart” from the top menu. Under most circumstances, students will want to use the XY (scatter) chart type that does not use lines to connect the data points.

Next, students will want to select the Data Range or Series that they will be displaying. Most of the time, students will be comparing two values and so it is best to manually select a series which will require them to delete the initial Data Range selected by default.

Once the students have selected their Y and X data series, make sure to title the graph and label the axes (including units used). After finishing the chart procedure, you should see something similar to the temperature vs. time graph shown in Figure 8.2.

Figure 8.2: Sample data from the payload's temperature sensor as a function of time. The time data points have been adjusted so that the actual experiment started at zero seconds.



It is important to have the students remember that when analyzing their data, they should reference not only specific values but the data points for those values as well. For example, in the chart above, students could write either;

The data prior to 0 seconds occurred before the experiment began. After 0 Seconds is when the experiment started.

Or

The payload was turned on at -280 seconds (23.8 C) and the sensors began taking data despite the GPS not having a time or location lock with orbiting satellites. At approximately -60 seconds (24.1 C) the payload entered the elevator where it was transported to the roof. At approximately -10 seconds (24.75 C) the payload exited the building and was exposed to the ambient outside temperature.

A trendline can be added to a graph by clicking on the chart, selecting “Chart” from the top menu, then “Add Trendline.” Select the best fit trend/regression type. Note, for Excel, under options, you can select it to display the equation for the trendline on the chart. See the additional resources in Appendix C for more information about displaying data using Excel or Open Office Calc.

In addition to graphing the experimental data, students can determine the range of the data values, the maximum, minimum, and average (if this is an interesting physical quantity) as well as calculating the total flight time, the experimental flight time (e.g. if data were only intended to be collected while in parachute descent then students will need to identify the time of apogee in their data) and can add labels identifying interesting experimental occurrences on the chart.

8.2.2 Experimental Uncertainties

Students should think about and/or calculate the experimental uncertainties in gathering the payload data. Depending on the level of skill and background of the students, they could think about qualitative issues such as “how accurate are these data given that the Payload only records a data point every 1 second” or “how well can we measure temperature if the thermometer readings are only recorded in increments of 0.01 degrees C” etc. Or more advanced students can use the data to calculate standard deviations or describe the goodness of fit for various trendlines.

Group Name: _____ Date: _____

Student Names: _____

Experiment Name: _____

Sensor Used: _____

Data Name (temp, altitude, acceleration, etc.): _____

Experiment start time: _____

Experiment end time: _____

Average data value (if relevant): _____

Maximum data value: _____

Minimum data value: _____

Trendline equation: _____

►students insert graph of data here

Data Basic Worksheet Example

If a standard error is assigned to each value (for example, 0.005 degrees in the temperature example above), error bars can be included in the graphs by right clicking (Windows) on the data in the graph created in Excel. Select “Format Data Series,” then select the “X” or “Y Error Bars” tabs and fill in the appropriate information.

8.2.3 Mapping the Data (Google Earth)

The data can be imported into Google Earth (freeware) and viewed three-dimensionally. This can be done immediately after importing the data in the S4DataExporter tool. Select export as a .kml file and then open that newly created .kml file with Google Earth.

Within Google Earth you can rotate around and see the flight path of the payload.

Figure 8.3: Actual path from GPS positions of the payload during a test walk around the SSU Schulz Information Center (home of the S4 team)



Figure 8.4: GPS readouts during a high-powered rocket flight, Snow Ranch, April 6th, 2013

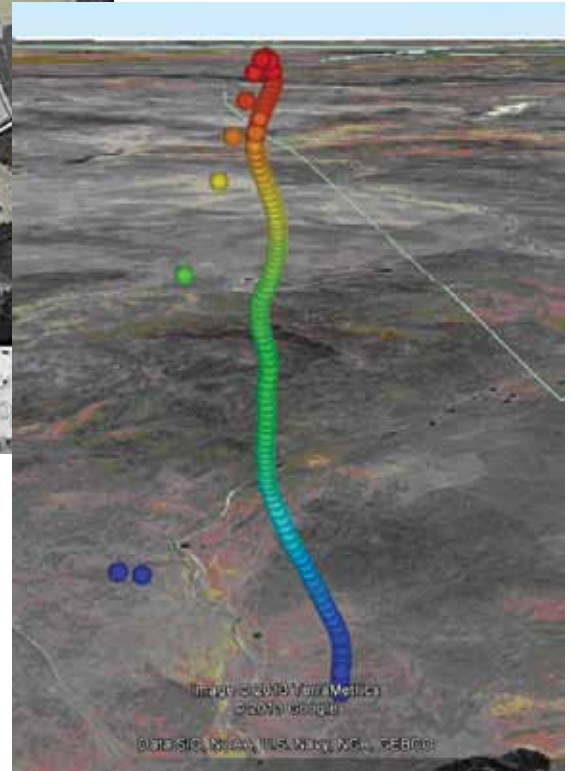


Figure 8.5: GPS flight path for a tethered balloon flight at the Tobin James Winery, Paso Robles, May 5th, 2012

8.2.4 Comparing Measurements to Predictions

Once the flight data have been analyzed students will need to compare the measured data with the estimated results that were hypothesized in their initial proposal.

During this step students will need to determine when their predictions aligned with the experimental data and where there were deviations. Where/when the data do not match the predictions, students may provide new hypotheses to account for the disparities. Some common reasons why the data may not agree with the predictions include:

- A variable was not accounted for; e.g. in using the barometric sensor to estimate altitude the students did not take into account that it was an exceptionally warm or cold day.
- A mechanical/engineering design issue; e.g. in decorating their payload with decals, students accidentally covered or impeded a sensor or sensors.
- Sensor failure, either failure to receive power during flight or an actual failure of the sensor itself. Each sensor has an accompanying data sheet which can be used to see if it is operating within working parameters after the flight if sensor failure is thought to be an issue.

Students may wish to come up with proposed adjustments to their experiment or alternate hypotheses for future study.

8.3 Presenting Results

Students are encouraged to follow at least one of the following three methods in presenting their results:

1. A lab report
2. A presentation poster
3. A slide-show presentation

8.3.1 Lab Report

The Lab report may adhere to the following format or to something similar as required by the educator.

Abstract – General summary of the purpose, hypothesis and results.

Introduction – Reasons why the students decided to explore this area of science and the results from other studies that the students researched.

Procedure – How the students decided to collect the data, what sensors, balloon or rocket, etc.

Experiment – Detailed information on the time, date, location, and weather during the experiment as well as a description of the flight; was it standard with no issues? Did a component of the delivery vehicle fail?

Results – Include here any prominent graphs of the data, as well as the information detailed in the Data Basics Worksheet.

Conclusion – Thoughtful summary comparing the expected results to those obtained.

Further Study – Possible issues that they may have encountered, and how to better construct future experiments for the same studies.

Notes – Included in the notes section are the Data Basics, Design, Procedure, Experimental Validation, Flight Log, and Pre-flight Checklist worksheets.

8.3.2 Presentation Poster

The presentation poster is essentially the Lab Report distilled down to one, large (usually at least 36" x 36") page. Students should include the main points/topic sentences from each of the sections in the lab report (the Introduction, Procedure, etc.) as well as the best (clearest, most accurate, etc.) data and graphs acquired.

8.3.3 Presentation Talk

Students may prepare the results of their experiment as a short (5 – 15 minute) talk. The talk should include the initial hypothesis, experimental design, the results of the experiment, and a comparison of the predictions to the results. If time/space allows, the talk could also include a discussion of any roadblocks and/or problems encountered and if/how the problems were resolved.

8.3.4. Beyond the Experiment

Students may also wish to develop a looking-forward proposal for their experiments (as outlined in their Further Study section of their Lab Report) that will allow them, or future students, to focus on acquiring more accurate and precise data. In creating these Further Study write-ups, educators are encouraged to keep them on hand for future students to work on as their own experiments so long as the base Further Study write-up is clear and makes for a viable, second-generation, experiment. Students should also be encouraged to use previous payloads as additional data collectors or as sounding boards to which they can compare their own payload's data.

Appendix A - Standards Alignment Information

1. Background

Content	NSES Grades 5-12	NCTM Grades 6-12	STL Grades 6-12
Introduction to NASA balloon and rocket launch programs	History and Nature of Science: Science as a Human Endeavor, Nature of Scientific Knowledge, Historical		Technology & Society: Students will develop an understanding of the influence of technology on history.
Newton's Laws (Review of 2D motion equations/graphs/plots)	Physical Science: Motions & Forces	Algebra: Represent and analyze mathematical situations and structures using algebraic symbols	
Defining Design Requirements	Science in Personal and Social Perspectives : Science and technology in society	Data Analysis & Probability: Formulate questions that can be addressed with data	Abilities for a Technological World: Students will develop the abilities to apply the design process

2. Essentials of Payload Design

Rockets or Balloons? (Trajectories, orbits, flight characteristics)	Physical Science: Motions & Forces	Algebra: Represent and analyze mathematical situations using algebraic symbols	The Designed World: Students will develop an understanding of and be able to select and use transportation technologies.
Altimeters (Barometric pressure)	Physical Science: Motions and Forces	Measurement: Understand measurable attributes of objects and the units, systems and processes of measurement	Abilities for a Technological World: Students will develop the abilities to use and maintain technological products and systems
GPS (GP Satellite System, Correcting the clock, GPS receivers, Position calculation)s	Physical Science: Motions and Forces	Geometry: Specify locations and describe spatial relationships using coordinate geometry and other representational systems	Abilities for a Technological World: Students will develop the abilities to use and maintain technological products and systems
Introduction to the Arduino Microcontroller (Circuits, Digital logic)	Science & Technology: Abilities of Technological Design	Algebra: Represent and analyze mathematical situations using algebraic symbols	Abilities for a Technological World: Students will develop the abilities to use and maintain technological products and systems.
Sensors (A/D conversion, Analog devices, Digital devices)	Science & Technology Abilities of Technological Design	Measurement: Understand measurable attributes of objects and the units, systems and processes of measurement	Abilities for a Technological World: Students will develop the abilities to use and maintain technological products and systems.

3) Payload Construction

Content	NSES Grades 5-12	NCTM Grades 6-12	STL Grades 6-12
Introduction to Basic Electronics	Science & Technology: Understanding about science and technology		Design: Students will develop an understanding of engineering design
Breadboarding with Arduino	Science & Technology: Abilities of Technological Design		Design: Students will develop an understanding of the role of troubleshooting, research and development, invention and innovation, and experimentation in problem solving.
End to End Testing	Science & Technology: Abilities of Technological Design		Design: Students will develop an understanding of the role of troubleshooting, research and development, invention and innovation, and experimentation in problem solving.

4) Acquiring and Displaying Payload Data in the Lab

Introduction to Programming			Design: Students will develop an understanding of engineering design.
Arduino Programming Environment (Procedural programming, Logic)	Science & Technology: Abilities of Technological Design		Abilities for a Technological World: Students will develop the abilities to use and maintain technological products and systems.
Telemetry (radio transmission, data rate, bandwidth, USB, WiFi)	Physical Science: Interactions of Energy and Matter, Transfer of Energy	Representation: Use representations to model and interpret physical phenomena	The Designed World: Students will develop an understanding of and be able to select and use information and communication technologies
Video (pixels, sampling, data rate, bandwidth, resolution)	Science & Technology: Abilities of Technological Design	Representation: Use representations to model and interpret physical phenomena	The Designed World: Students will develop an understanding of and be able to select and use information and communication technologies
Graphical Data Display		Algebra: Use mathematical models to represent and understand quantitative relationships	The Designed World: Students will develop an understanding of and be able to select and use information and communication technologies.

5) Testing the Payload

Simulating the Flight Environment	Science & Technology: Abilities of Technological Design	Algebra: Use mathematical models to represent and understand quantitative relationships	Design: Students will develop an understanding of the role of troubleshooting, research and development, invention and innovation, and experimentation in problem solving.
Troubleshooting	Science & Technology: Abilities of Technological Design	Measurement: Apply appropriate techniques, tools, and formulas to determine measurements	Design: Students will develop an understanding of the role of troubleshooting, research and development, invention and innovation, and experimentation in problem solving.
Analyzing Test Data	Science & Technology: Abilities of Technological Design	Data Analysis & Probability: Formulate questions that can be addressed with data and collect, organize, and display relevant data to answer them	Design: Students will develop an understanding of the role of troubleshooting, research and development, invention and innovation, and experimentation in problem solving.

6) Integration with Flight Vehicle

Content	NSES Grades 5-12	NCTM Grades 6-12	STL Grades 6-12
Flight Readiness Review	Science & Technology: Abilities of Technological Design		Abilities for a Technological World: Students will develop the abilities to apply the design process.
Rocket Integration and Test	Science & Technology: Understanding about science and technology		The Designed World: Students will develop an understanding of and be able to select and use transportation technologies.
Balloon Integration and Test	Physical Science: Interactions of Energy and Matter, Transfer of Energy		The Designed World: Students will develop an understanding of and be able to select and use transportation technologies.

7) Flight Data Acquisition

Determining the Flight Trajectory		Geometry: Specify locations and describe spatial relationships using coordinate geometry and other representational systems	The Designed World: Students will develop an understanding of and be able to select and use transportation technologies.
Payload Recovery	Science & Technology: Abilities of Technological Design		Abilities for a Technological World: Students will develop the abilities to use and maintain technological products and systems
Safety Considerations	Science in Personal & Social Perspectives: Personal and community health		Technology & Society: Students will develop an understanding of the effects of technology on the environment.

8) Scientific Results

Analyzing Flight Data	Science as Inquiry: Abilities necessary to do scientific inquiry	Measurement: Apply appropriate techniques, tools, and formulas to determine measurements Data Analysis & Probability: Formulate questions that can be addressed with data and collect, organize, and display relevant data to answer them	Abilities for a Technological World: Students will develop the abilities to assess the impact of products and system.
Comparing Measurements to Requirements	Science & Technology: Abilities of Technological Design	Algebra: Use mathematical models to represent and understand quantitative relationships, Analyze change in various contexts	Abilities for a Technological World: Students will develop the abilities to assess the impact of products and system.
Presenting Results	Science as Inquiry: Abilities necessary to do scientific inquiry	Process: Communication	Abilities for a Technological World: Students will develop the abilities to assess the impact of products and system.
Planning for design revisions	Science as Inquiry: Abilities necessary to do scientific inquiry	Data Analysis & Probability: Develop and evaluate inferences and predictions that are based on data	Design: Students will develop an understanding of engineering design.



Appendix B - NASA-funded programs

S4 Partners:

AERO Institute: <http://www.aeroi.org/>

Association of Experimental Rocketry of the Pacific (AeroPac): <http://www.aeropac.org>

Dryden Flight Research Center: <http://www.nasa.gov/centers/dryden/home/index.html>

Education and Public Outreach at Sonoma State University: <http://epo.sonoma.edu/>

Endeavor Institute: <http://endeavourinstitute.org/>

NASA-Sponsored Student flight programs:

Balloon Fest: <http://scipp.ucsc.edu/outreach/BF/balloon.html>

CubeSat Launch Initiative: http://www.nasa.gov/directorates/somd/home/CubeSats_initiative.html

NASA Student Launch Initiative:

http://www.nasa.gov/offices/education/programs/descriptions/Student_Launch_Initiative.html

NASA University Student Launch Initiative:

http://www.nasa.gov/offices/education/programs/descriptions/University_Student_Launch_Initiative.html

Rockets for Schools program: <http://www.rockets4schools.org>

Teachers in Space: <http://tis.spacefrontier.org/>

Team America Rocket Challenge: <http://www.rocketcontest.org/>

Appendix C - References and additional resources

Books and Educator Guides

- 1405 manuscript, *Bellifortis* by Konrad Kyeser: “fire breathing dragons fueled by oil-lamps used by warriors on horseback” image http://commons.wikimedia.org/wiki/File:Konrad_Kyeser,_Bellifortis,_Clm_30150,_Tafel_21,_Blatt_91v.jpg
- *Handbook of Model Rocketry*, G. Hary Stine and Bill Stine. ISBN: 9780471472421
- *Modern High-Power Rocketry 2*, Mark Cenepa. ISBN: 1412058104
- *NASA Ares: Launch and Propulsion Educator Guide* EP-2009-02-32-MSFC http://www.nasa.gov/pdf/340827main_Ares_Educator_Guide.pdf
- *NASA Adventures in Rocket Science* EG-2007-12-179-MSFC: http://www.nasa.gov/pdf/265386main_Adventures_In_Rocket_Science.pdf
- *NASA Rockets Educators Guide* EG-2003-01-108-HQ: http://quest.nasa.gov/space/teachers/rockets/58269main_Rockets.Guide.pdf
- *NASA Rockets Educators Guide* EG-2008-05-060-KSC: http://er.jsc.nasa.gov/seb/main_Rockets.Guide.pdf
- *Secondary school educator's guide* - suggestions for simple balloon payloads, by James Dann. <http://scipp.ucsc.edu/outreach/balloon/guide/guide.html>

Balloons

- Balloon purchases: <http://www.scientificsales.com/balloons.htm>
- 300-lb Dacron kite line purchases: <http://www.intothewind.com>
- Near Space Ventures (free online flight prediction and balloon ascent calculator): <http://nearspaceventures.com/>

Launch Event Institutions, Clubs, and Competitions

- A Rocket Launch for International Student Satellites (ARLISS): <http://www.arliss.org/>
- Association of Experimental Rocketry of the Pacific (AeroPac): <http://www.aeropac.org>
- Balloon Fest <http://scipp.ucsc.edu/outreach/BF/balloon.html>
- Battle of the Rockets: <http://www.rocketbattle.org/Main.html>
- CanSat competition: <http://www.cansatcompetition.com/>
- National Association of Rocketry: (NAR): <http://www.nar.org/>
- Netherlands CanSat competition: <http://www.tudelft.nl/live/pagina.jsp?id=65137fd3-91aa-47ec-8a12-b0572af157b4&lang=en>
- Norwegian CanSat competition: <http://www.narom.no/artikkel.php?aid=2&bid=56&oid=944>
- Rocketry Organization of California: <http://rocstock.org/>
- Spanish (“International”) CanSat Competition: <http://cansat.leem.es>
- Team America Rocket Challenge: <http://www.rocketcontest.org/>
- Tripoli Rocketry Association (TRA): <http://www.tripoli.org/>

Payload Parts

(see Appendix F, Preferred Parts List for individual links to each part of the payload)

- Advanced Circuits: <http://www.4pcb.com/>
- Harbor Freight Tools: <http://www.harborfreight.com>
- Mouser Electronics: <http://www.mouser.com/>
- Sparkfun Electronics, Inc: <https://www.sparkfun.com/>

Rocket Parts and Software

- Estes Rockets Free Online Educational Materials: <http://www2.estesrockets.com/cgi-bin/WEDU100P.pgm>
- Public Missiles, LTD. (rocket airframes): <http://www.publicmissiles.com/>
- RockSim Software from Apogee: http://www.apogeerockets.com/Rocket_Software
- What's Up Hobbies: <http://stores.whatsuphobby.com>

Arduino Resources

- Arduino home page: <http://www.arduino.cc/>
- Arduino playground: <http://playground.arduino.cc/>
- *Arduino Cookbook* by Michael Margolis, ISBN 978-0-596-80247-9
- *Getting Started with Arduino* by Massimo Banzi, ISBN 978-0-596-15551

Tutorials

- Beginner's Guides to Rockets, NASA Glenn Research Center <http://exploration.grc.nasa.gov/education/rocket/bgmr.html>
- Tony Alcocer Rocket & Motor building (includes fishing videos as well): <https://www.youtube.com/user/tfish38>
- Working With Breadboards: <http://www.instructables.com/id/Breadboard-How-To/>
<http://www.youtube.com/watch?v=oiqNaSPTI7w> (part 1)
<http://www.youtube.com/watch?v=Mq9XMNsoAd8> (part 2)
- Using Microsoft Excel
(Excel 2013): <http://www.gcflearnfree.org/excel2013>
(circa 2009 versions of Excel): <http://sunburst.usd.edu/~bwjames/tut/excel/>
<http://www.youtube.com/watch?v=8L1OVkw2ZQ8>
- Using OpenOffice Calc
http://spreadsheets.about.com/od/otherspreadsheets/ss/080616_24_calc.htm
<http://www.youtube.com/watch?v=fT7KdzwAnr4>

Other-Education

- International Technology Educators Association (2007), *Standards for Technological Literacy: Content for the Study of Technology*, <http://www.iteaconnect.org/TAA/PDFs/xstnd.pdf>
- Katchi, L., Pearson, G., and Feder, M., Editors; (2009) Committee on K-12 Engineering Education; National Academy of Engineering and National Research Council, *Engineering in K-12 Education: Understanding the Status and Improving the Prospects*. Nakasuka, Shinichi, Nakamura, Yuya, Funase, Ryu, Nagai, Masaki, & Kawashima, Rei
- NASA Education Recommendation Report, 2011:
http://www.nasa.gov/pdf/536766main_NASA-Education-Recommendation-Report-v15-web.pdf
- National Council of Teachers of Mathematics, Principles and Standards for School Mathematics:
<http://www.nctm.org/standards/content.aspx?id=4294967312>
- National Research Council Committee on NASA's Suborbital Research Capabilities (2010), *Revitalizing NASA's Suborbital Program: Advancing Science, Driving Innovation, and Developing a Workforce*, National Academies Press:
http://www.nap.edu/catalog.php?record_id=12862
- National Research Council (1996), *National Science Education Standards*, National Academies Press:
http://www.nap.edu/catalog.php?record_id=4962
- Next Generation Science Standards: <http://www.nextgenscience.org/>



Appendix D - Glossary

.csv – comma separated values (a type of data file)

.kml – a type of Google Earth data file

#define – a coding command that defines a variable

A

abstract – the general summary of a lab report

acceleration – the rate of change of velocity

accelerometer sensor – a sensor that detects the payload's acceleration

Ampere (A) – a unit of measurement for electric current, 6.241×10^{18} electrons or one coulomb per second

analog – a smooth or continuous voltage and current, as opposed to digital

Arduino – the microprocessor that is used to run the S4 experiment

average thrust – the average force, in Newtons, of a rocket motor

B

balloon – a helium filled elastic device used to carry an experiment into the atmosphere

barometric pressure sensor – a sensor that detects changes in atmospheric (air) pressure

base components – the parts of the S4 payload that are essential to the payload. This list does not include specific experiment sensors

battery – the power source for the S4 payload

binary – the number system that computers use, comprised of 0s and 1s (base 2)

C

capacitance – the ability of an electronic component to store electrical charge, measured in Farads

capacitors – an electronic device that stores electrical charge

circuits – a closed electrical system used to power and regulate other electronic device

clock line (SCL) – a hexadecimal port used in the TWI or I2C communications protocol

code – the text within a program that determines how a program will work

comments – non-code text within a program to help clarify the code for the human programmer(s)

compile – taking code and turning it into the digital 1s and 0s that can be read by a computer

computer programming – the method by which a computer is given a set of instructions that tell it what to do

conclusion – lab report section summarizing the experimental results, compared to the predictions

Curly Brackets {} – start and stop indicators signifying a function in the Arduino computer code

current – the flow of electrons through a circuit, measured in Amperes (A)

D

debug – the process of testing and removing errors from code

data line (SDA) – a hexadecimal port used in the TWI or I2C communications protocol

delay(____) – a function that tells the Arduino to do nothing for a specified number of milliseconds

digital – on (1) or off (0) states for voltage and current, as opposed to analog
digitalWrite(____,____) – a function that turns on or off any pin that is an output

E

experiment – lab report section that includes the detailed information about the experiment

F

F – see Farads.

Farads (F) – the unit of capacitance equal to ampere seconds per volt

flight board – the printed circuit board that also provides structural integrity for the payload

function – a section of code that performs a specific action or task

further study – lab report section that describes insights for further exploration

G

Global Positioning System – a network of Earth-orbiting satellites that help a device determine its location on Earth

gondola – the protective structure in which a payload is placed for balloon flights

GPS – see Global Positioning System

H

hexadecimal – a number system that uses 0-9 and the letters A-F to represent 10-15 (base 16)

high-powered rocketry (HPR) – rocketry that uses motors above class G and require certification from either the National Association of Rocketry (NAR) or the Tripoli Rocketry Association (TRA)

HPR – see high-powered rocketry

humidity sensor – a sensor that detects moisture

I

I/O pins – an Input/Output pin

I2C communications – a type of protocol that uses hexadecimal code

IDE – see Integrated Development Environment

Integrated Development Environment (IDE) – a software package that incorporates all of the necessary tools to help you develop your software

introduction – lab report section that explains the motivation for the experiment

L

LED – Light Emitting Diode

lead – the exposed conductive end of a part that connects to an electrical device

logic level converter – the electronic device in the S4 payload that safely steps down a higher voltage input to a lower voltage output

loop() – a function that is executed repeatedly while the program is running

M

magnetometer sensor – a sensor that can detect Earth's and other magnetic fields

Micro-SD card – one of the many sizes of SD cards used to store data for many electronic devices, including the S4 payload

microprocessor – the essential core hardware of a computer

motors – the casing and fuel used in high-powered rocketry

MySQL database – a type of database that allows for data storage and retrieval.

N

NAR – see National Association of Rocketry

National Association of Rocketry (NAR) – a rocketry club that hosts launch-events across the United States, was founded in 1957, and offers certification for high powered rocketry.

Network Key – this is the password that is required to log into your network. If security is not an issue for you, you may decide to leave the network unsecured, meaning no password is required and the network key is not important.

notes –lab report section that includes worksheets documenting all the work done to develop and run the experiment.

O

Ohm's Law – that electrical current is equal to the quotient of voltage and resistance

Ohm (Ω) – the SI unit of electrical resistance

Open Log chip – the chip that records the S4 payload data on the Micro-SD card which the chip also houses.

P

parallel – an arrangement of circuit parts that have equal voltages in each parallel segment

payload – the scientific experiment that is housed by the rocket or balloon gondola.

PCB – See Printed Circuit Board.

pin – in electronics, the conductive, pin-like, material that connects one device or component to another.

pinMode(__, __) – a function that directs the Arduino to configure a pin in a specified way.

Printed Circuit Board (PCB) – the structurally significant component of a device that also electrically connects other devices to each other.

procedure – the section in a lab report that describes how data were collected.

Processing – the programming language used to program Arduinos.

programming language – a way to communicate human intent to electronic devices through an intermediate artificial language.

protocols – the method by which the Arduino communicates with another device.

R

resistance – the difficulty that a electrical current has in flowing through a material, measured in the SI units of Ohms (Ω).

resistor – an electrical component that introduces a specified amount of electrical resistance in a circuit.

results –lab report section in which the experimental findings are presented.

rocket – a vehicle that carries its own propellant, the expulsion of which gives the vehicle thrust.

RX pin – in Serial communications, the receiving pin

S

S4 – Small Satellites for Secondary Students (this project)

S4 payload – the payload that this guide describes

S4 Sketchbook – the library of pre-written sketches for the S4 payload and its optional sensors

SCL – see clock line

SD card – Secure Digital card, used to store data.

SDA – see data line

semicolon – a semicolon (;) is used to end a statement and separate elements of the Arduino's computer code

sensor components – these are the devices that gather data about a physical quantity like temperature or pressure

serial communications – a form of device protocol that only uses RX and TX pins

serial Interface – the physical interface of pins that allows for serial communication

series – when electrical current flows through the same components (i.e. resistors) in turn, or one at a time

setup() – a function that executes, or runs, once and only once when the device is first powered on

shield – boards that are made in order to plug directly into Arduino PCBs

sketch – a small Arduino computer program

solder – both the act of connecting electrical pins through molten metal and the wire metal used in connecting the pins

source code – the original computer code for a device or program

SPI communication – see Synchronous Serial Data Protocol

SSID – the Network's name, typically chosen during the process of establishing a network

Synchronous Serial Data Protocol (SPI) – a computer protocol that allows multiple devices to be controlled over the same set of pins

T

temperature sensor – a sensor that detects the temperature

tethered balloons – helium filled industrial balloons that are tied to extended ropes so as to keep them local

total impulse – the average thrust (in Newtons) per second of a rocket motor.

TRA – See Tripoli Rocketry Association

Tripoli Rocketry Association (TRA) – international rocketry organization that hosts launch-events across the United States, was founded in 1964, and offers certification for high powered rocketry

TWI communication – a common protocol that uses only two wires, the SCL and SDA

TX Pin – in Serial communications, the transmitting pin.

U

USB – universal serial bus, a standard connection for electronic devices

V

variables – a way for a computer program to reserve memory locations where values can be stored for later use

VC – see Virtual Classroom

Virtual Classroom (VC) – the Virtual Classroom is a mobile system that allows for launch site telemetry and video to be fed in near real-time to the internet through a satellite connection

void – a function command that prohibits that function from returning any values

voltage – the difference in electrical potential energy between two points in a circuit

voltage regulator – an electronic device that limits the amount of voltage that passes through it

Volt (V) – the SI unit for voltage

W

Wi-Fly chip – the electronic device that allows the S4 payload to communicate over the 802.11g wireless network protocol

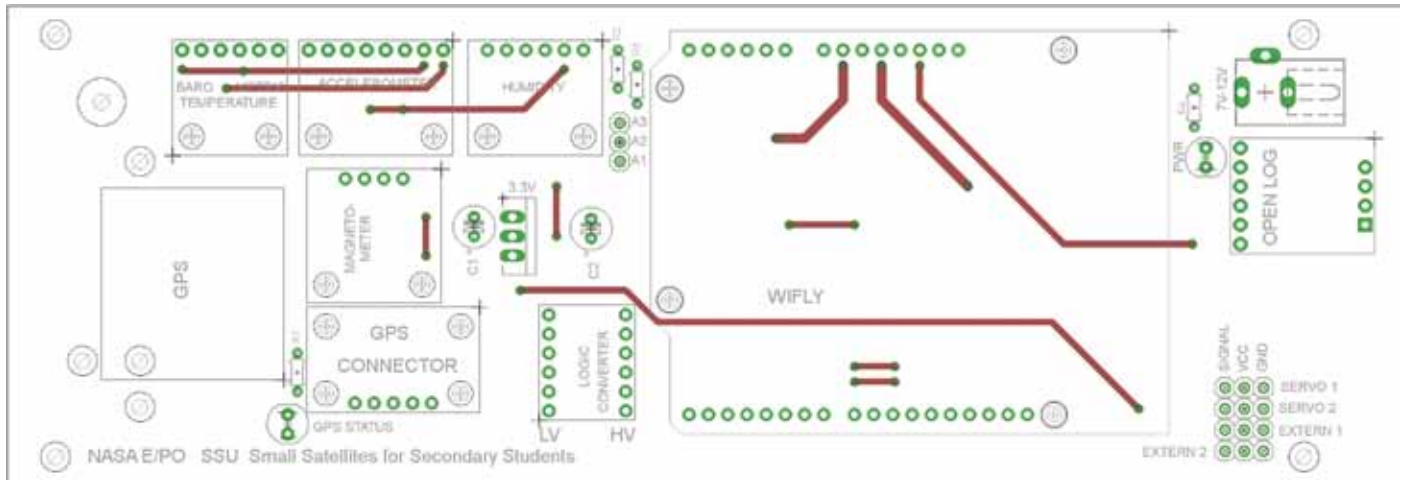
X

XAMPP – A program that will run a MySQL database on your server

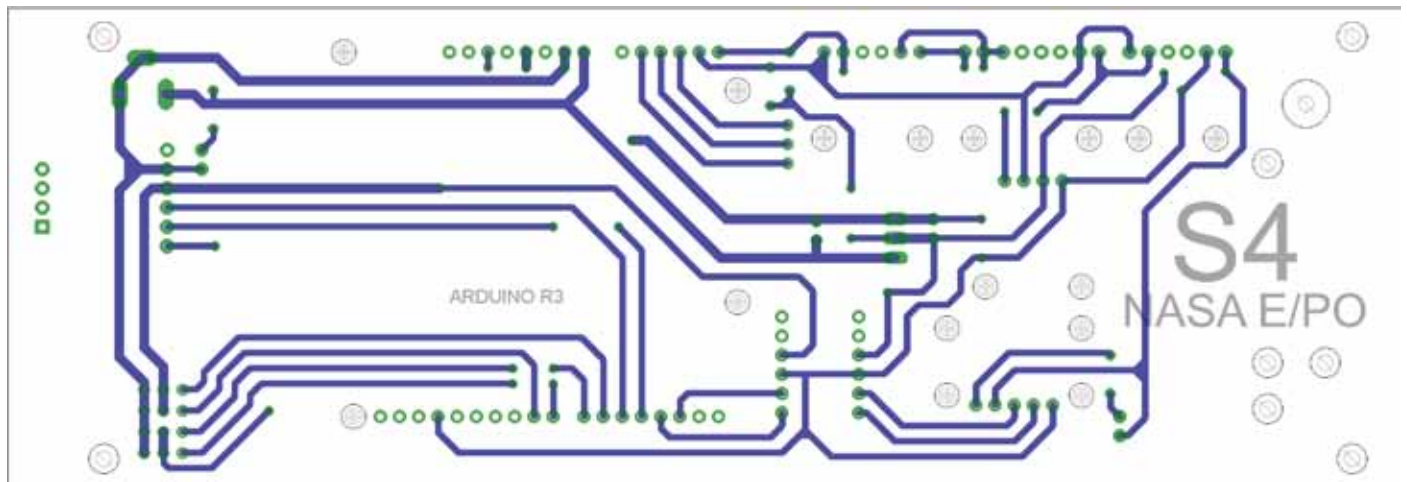
Appendix E - Electrical Schematic for Flight Board



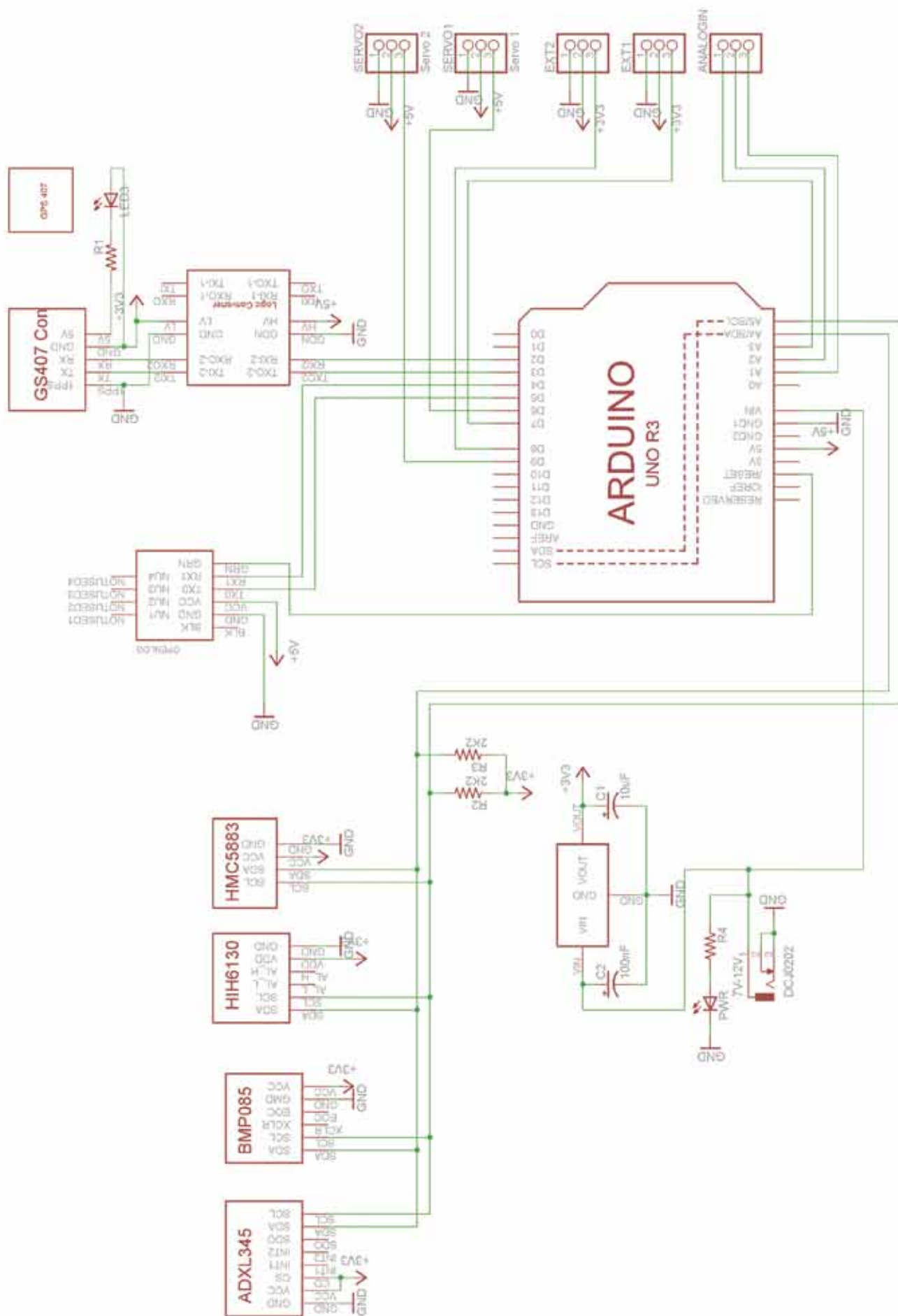
Top



Bottom



Circuit



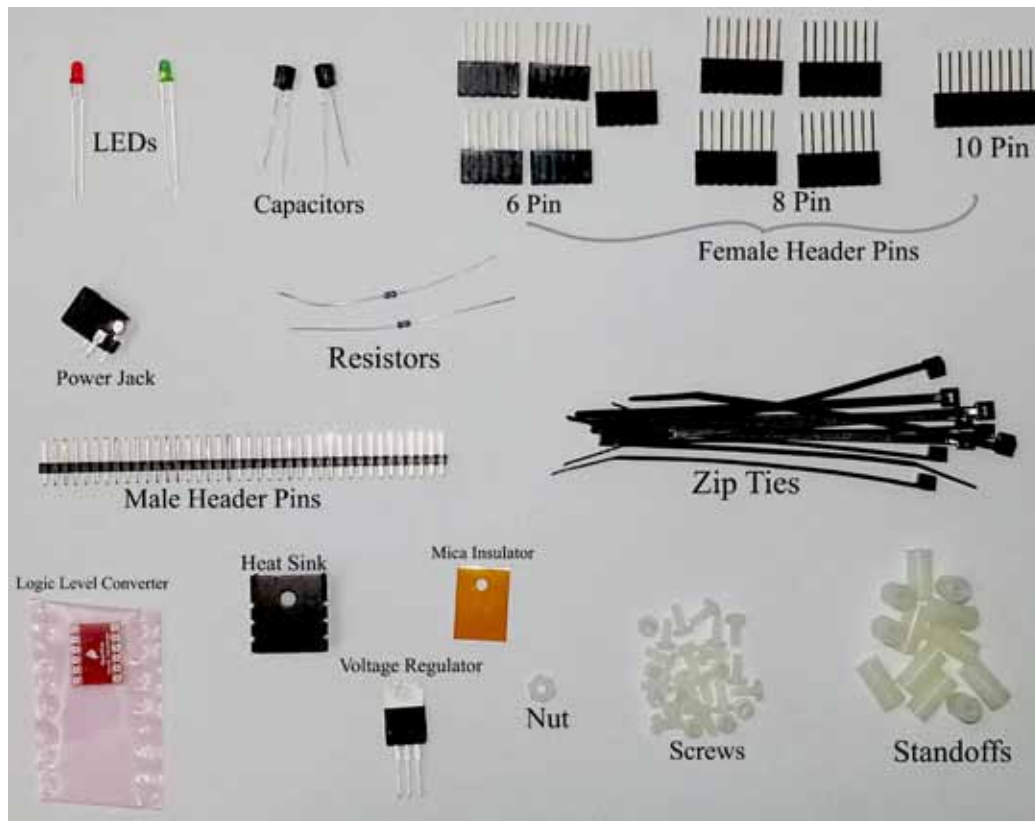
Appendix F - Preferred Parts List



Part	Description	QTY	Price	URL
Main Board				
PCB	Custom Printed Circuit Board	1	6.12	http://www.4pcb.com
Standoff	1/2" Standoff nylon 4-40 threaded	20	0.21	http://www.mouser.com/Search/ProductDetail.aspx?R=561-TSP3virtualkey56100000virtualkey561-TSP3
Standoff	1/4" Nylon 4-40 bolt	41	0.07	http://www.mouser.com/Search/ProductDetail.aspx?R=561-P440.25virtualkey56100000virtualkey561-P440.25
Nut	4-40 nut	1	0.19	http://www.mouser.com/ProductDetail/Eagle-Plastic-Devices/561-G440/?qs=sGAEpiMZZMsqI59i2oRciqPBwuKPLJRKPAZ4dfvAsE%3d
LED	LED - Basic Red 3mm	1	0.35	https://www.sparkfun.com/products/533
LED	LED - Basic Green 3mm	1	0.35	https://www.sparkfun.com/products/9650
Resistor	62 Ohm Resister	1	0.15	http://www.mouser.com/ProductDetail/Xicon/270-62-RC/?qs=sGAEpiMZZMu61qfTUdNbGyNKiyP0gIG%252bq2TM%26V%26bw%3d
Resistor	402 Ohm Resister	1	0.15	http://www.mouser.com/ProductDetail/Xicon/270-402-RC/?qs=sGAEpiMZZMu61qfTUdNbG9rLbwLhtOUu2uk7%2fA7LGi4%3d
Header	Arduino Stackable Header - 6 Pin	5	0.5	https://www.sparkfun.com/products/9280
Header	Break Away Headers - Straight	3	1.5	https://www.sparkfun.com/products/116
Jack	DC Barrel Jack Adapter - Breadboard Compatible	1	0.95	https://www.sparkfun.com/products/10811
Jack Plug	9V to Barrel Jack Adapter	1	2.95	https://www.sparkfun.com/products/9518
CAP	Electrolytic Decoupling Capacitors - 10µF/25V	2	0.45	https://www.sparkfun.com/products/523
Converter	Logic Level Converter	1	1.95	https://www.sparkfun.com/products/8745
Power Supply	Wall Adapter Power Supply - 9VDC 650mA	1	5.95	https://www.sparkfun.com/products/298
Heatsink	Heat Sink	1	0.86	https://www.sparkfun.com/products/121
Base Payload Components				
Processor	Arduino Uno - R3	1	29.95	https://www.sparkfun.com/products/11021
Wifi	WiFly Shield	1	69.95	https://www.sparkfun.com/products/9954
Antenna	2.4GHz Duck Antenna RP-SMA - Large	1	9.95	https://www.sparkfun.com/products/558
WiFi cable	Interface Cable RP-SMA to U.FL	1	4.95	https://www.sparkfun.com/products/662
GPS	50 Channel GS407 Helical GPS Receiver	1	89.95	https://www.sparkfun.com/products/11466
GPS Connector	EM-406 Connector Breakout	1	4.95	https://www.sparkfun.com/products/10402
GPS Cable	Interface Cable for EM401 and EM406	1	1.95	https://www.sparkfun.com/products/574
Logger	OpenLog	1	24.95	https://www.sparkfun.com/products/9530
SD Card	Flash Memory - microSD 1GB	1	6.95	https://www.sparkfun.com/products/8163

Part	Description	QTY	Price	URL
Expanded Payload Components				
Sensor	Triple Axis Magnetometer Breakout - MAG3110	1	14.95	https://www.sparkfun.com/products/10530
Sensor	HIH6130 Humidity Sensor Breakout	1	29.95	https://www.sparkfun.com/products/11295
Sensor	Barometric Pressure Sensor - BMP085 Breakout	1	19.95	https://www.sparkfun.com/products/11282
Sensor	Triple Axis Accelerometer Breakout - ADXL345	1	27.95	https://www.sparkfun.com/products/9836
Supplies and Tools				
Soldering Iron	Soldering Iron	1	3.99	http://www.harborfreight.com/30-watt-lightweight-soldering-iron-69060.html
Multimeter	Multimeter	1	3.99	http://www.harborfreight.com/7-function-digital-multimeter-92020.html
Solder	Solder	1	2.99	http://www.harborfreight.com/lead-free-rosin-core-solder-69378.html
Helping Hands	Helping Hands	1	2.99	http://www.harborfreight.com/helping-hands-319.html
Adhesive	Heat Sink Compound	1	1.95	https://www.sparkfun.com/products/9599

Appendix G - Parts and Tools



Appendix H - Completed Labeled Flight Board

