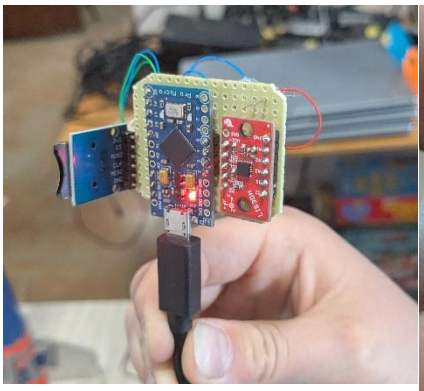
6th-grader gets electronic flight data from a model rocket flight
OR
Alden and the magic rocket flight

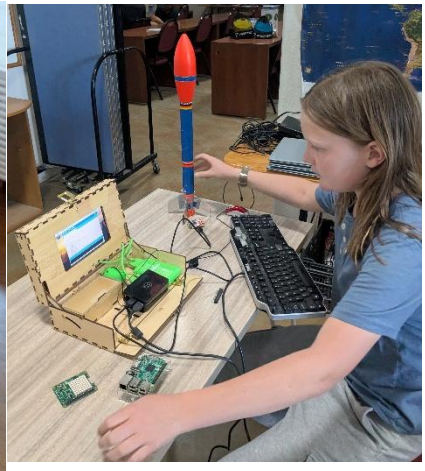
Alden just finished 6th grade at the Las Cruces Academy, a non-profit private school. He just finished flying a model rocket carrying a data module he conceived, designed, built, tested, and retrieved data therefrom. He's liked rockets since kindergarten. He's liked electronics, computers, and programming for years. This last quarter of the school year he chose to take an elective class in rocketry, meeting daily for 35 minutes for 9 weeks. He and teacher Dr. Vince Gutschick had wide-ranging discussions of the physics of rocket flight as well as the history, the actual technology, and the uses of rockets for Earth observations and more.

Alden readily saw that his favorite Arduino microcomputer is light enough to fly on a model rocket. He proposed to build a tiny device using the Arduino Micro Pro (below), greater than life-size) to capture real-time measurements of the forces on an Estes Omloid model rocket. This promised to be 'cool;' could we then measure the height of the flight? Could we see all the details during the flight, to compare with a calculus simulation that Vince devised a number of years ago?

Alden checked for electronic devices that could 'talk' to each other and operate on a single, common battery voltage. He found a source for a tiny, lightweight, and inexpensive ($5) chip, a 3-axis accelerometer. He then figured out how to send the data to a regular SD card as you'll find in cameras, tucked into an equally light, tiny, and inexpensive chip. Alden checked that the total weight was tolerable. He then laid out the 3 chips on a small piece of

perfboard (below). He and Vince checked the wiring plan, then Alden used soldering and wire-wrapping (which he learned in a few minutes) to put it all together. Vince got a small CR23 battery to power it all. The whole business fits nicely into the kind-of-chubby nosecone of the Omloid rocket (also below). Of course, Alden had to write a short and clever program in the C-language to run it all. Much testing ensued, using various computers at hand – a Raspberry Pi that he had assembled earlier, a Chromebook from the school, or his mother's Ma at home; three different operating systems.





Next challenge: How to start the data collection at launch time? The Arduino microcomputer would get a start signal and run the data acquisition program. Alden looked at two tiny radio modules as used in garage-door openers. The receiver inside the rocket would be pulsed by the transmitter operated by hand at launch time. He found them glitchy. He considered a tiny wire from a small battery to the rocket; the blast and pull upon launching would sever the wire and create a signal. Vince's experience kicked in here as he pointed out the unreliability and the delay. Alden came up with the final method: Just before launch he'd connect the battery power to the whole module in the nosecone; the program in the Arduino would wait 30 seconds and start data collection. That gave time to finish attaching the nosecone and then step back to grab the Estes igniter controls. A second person, Vince or Alden's mother, Moire, would call out the 30-second mark and tell Alden to ignite the rocket engine… and begin taking a video of the launch for the record.

Will it fly, literally and figuratively? Will the electronics survive the forces of the launch, the parachute ejection, and landing? Well, not without some failures as classic learning experiences. Pre-flight: Safety. Test the Estes launch controller; it uses 4 AA batteries to send current through a wire loop coated with ignitable powder. The igniter goes into the nozzle of the rocket engine; on heating up it ignites the gunpowder in the engine. Precaution two: Fit ourselves with hard-hat helmets. Test flight one: Boom. The rocket has the parachute stuffed carefully into the body, above the engine. The engine is made to fire for 1.67 seconds, then wait 3 seconds to fire a charge into the rocket body to blow the parachute out. Well, the parachute was just too tightly stuffed. It didn't blow out. Instead,

the rocket body, a cardboard tube, blew apart. Fortunately, it parted at a nice joint. Alden quickly repaired it after the flight. However, the nosecone landed hard, on its own. The only casualty was the SD card that snapped in two. Test flight two: Just test the parachute ejection with more careful stuffing into the rocket. It worked, even if Alden had to ask the school's tallest 8th-grader to retrieve the rocket and parachute because it landed in a tree! Test flight three: The whole system goes up this time. Problem: The launch forces pulled one of the battery wires off the board; the program never started. Test flight four: Alden soldered all the wire-wrapped connections, for strength, and put heavier wire on the battery leads. Ta da! The movie (←) captures the flight, as does a mid-flight still:

It worked. Alden ran joyfully to retrieve the rocket. He opened the nosecone, reached the electronics, and pulled the SD card out to put it in a Chromebook computer.

The data were all there (I used “data” as the plural of “datum)!

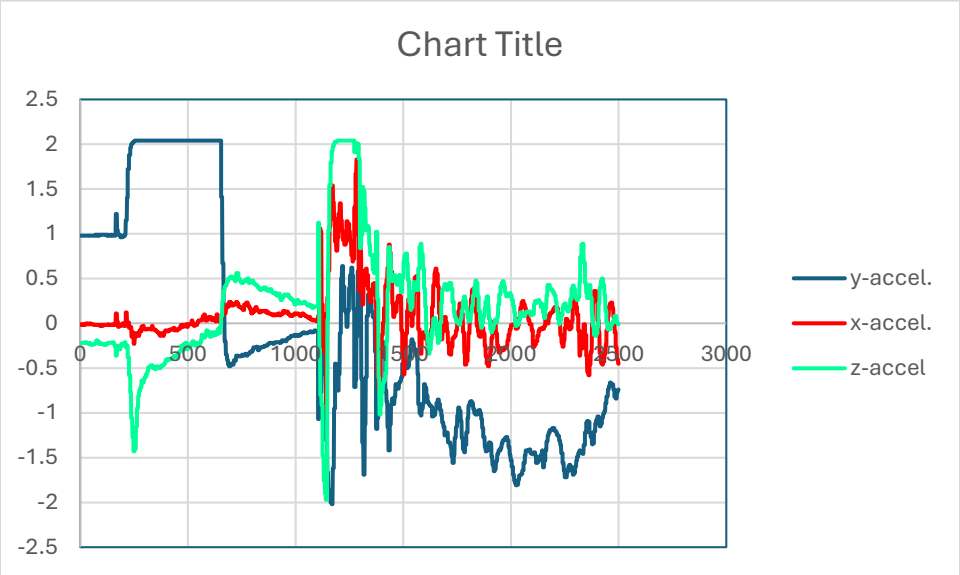
The file is 5000 lines of text:
At 32909:-0.01,0.98,-0.22
/n
At 32915:-0.01,0.98,-0.22
/n
At 32917:-0.01,0.98,-0.22
/n
At 32921:-0.01,0.98,-0.22
/n
At 32925:-0.01,0.98,-0.22
/n
At 32929:-0.01,0.98,-0.22
/n
...
The time in milliseconds comes first, then the acceleration in the x, y, and z axes, all in units of "g's" or surface gravity multiples. The y axis shows the simple force of gravity as the rocket sits there. The acceleration is positive because the accelerometer was pointing down on the y-axis. The x-acceleration is nearly zero, from a good alignment of the rocket. The z-acceleration is slightly negative, as the rocket is tilted a bit off vertical so that it won't drift too far as the wind carries it (and the drag forces vary).

First step: Clean up the data. Get rid of the line-feeds (/n). Separate the x-acceleration value from the time stamp and the colon. Alden did this in the Python language. Vince did it independently using the Vim editor. Here's how the data look, around a second after ignition. The y- or upward acceleration jumps higher, reaching 2 g's by 162 milliseconds after ignition.

At 33718:	-0.03	0.99	-0.21
At 33720:	-0.03	1.14	-0.27
At 33724:	0.12	1.15	-0.27
At 33728:	0.12	1.15	-0.27
At 33732:	0.12	1.15	-0.27
At 33734:	0.12	1.15	-0.27
At 33738:	0.12	1.15	-0.27
At 33742:	0	1.4	-0.25
At 33746:	0	1.4	-0.25
At 33755:	0	1.4	-0.25
At 33759:	0	1.4	-0.25
At 33761:	0	1.72	-0.38
At 33765:	0.01	1.72	-0.38
At 33769:	0.01	1.72	-0.38
At 33773:	0.01	1.72	-0.38
At 33775:	0.01	1.72	-0.38
At 33779:	0.01	1.72	-0.38
At 33783:	-0.05	1.88	-0.58

At 33787:	-0.05	1.88	-0.58
At 33789:	-0.05	1.88	-0.58
At 33794:	-0.05	1.88	-0.58
At 33798:	-0.05	1.88	-0.58
At 33802:	-0.05	1.88	-0.58
At 33804:	-0.04	1.96	-0.77
At 33808:	-0.04	1.96	-0.77
At 33816:	-0.04	1.96	-0.77
At 33820:	-0.04	1.96	-0.77
At 33822:	-0.04	2	-1.06
At 33826:	-0.15	2	-1.06

It's ready to plot! Alden used Google sheets and Vince used Excel:



Very nice!

- The upward or y- acceleration (blue line) keeps a very steady value of 2.-04 g's for 1.767 seconds. Interesting, since the Estes website says that most of their engines have a strong start and then a long finish. Also, the engine burned a bit longer than the 5/3 or 1.67 seconds expected for a C6-3 engine.
- At burnout, acceleration goes negative; aerodynamic drag shows up big time.
- While Vince didn't put a time scale on the graph, the rocket 'coasts' for about 1.4 seconds, with decreasing drag… and then the charge that ejects the parachute really jiggles the accelerometer. It calms down, more or less for another 5 seconds (until the program automatically stops). You can see that the acceleration is now negative; the nosecone is flipped mostly upside down and the accelerometer is 'looking' the other way from when the rocket launched.
- The x-acceleration (red) stays near zero during the engine burn. The rocket is keeping its orientation in the x-direction. The y-acceleration (green) has some

action. The rocket is tilting in the x-direction, in one way during the engine burn and then the other way in the coasting phase.

There's much more to look at in the data. For one, is the altitude reached during engine burn as expected from Vince's simulation? Quick check: At an acceleration, *a*, of 2.04 g's for 1.767 seconds we expect a gain in height of 0.5*2.04*(1.767s)^2, or 31.2 meters (102 feet). Vince's simulation predicted 31.5 meters! Both the rocket flight and the simulation show good science. We'll do more analysis for the final height and such. Congratulations, Alden, on great execution of an ambitious project covering rocket mechanics, electronics, data analysis, and much fun.