

# **Effect of sensor sampling frequency (10-100 Hz) on Apogee estimation accuracy in a model rocket flight.**

SRI HARSHA VIBHAS VARIGONDA

## **Abstract**

Student rocketeers like me are mostly excited about achieving the maximum altitude also referred to as Apogee for the model rocket launches. Finding low cost flight computers as alternatives to measure the Apogee got me interested and I embarked on this study to investigate how sensor sampling frequency affects the Apogee calculation. I built a custom flight computer using Adafruit Feather RP2040 Adalogger, a BMP388 barometer, and an LSM6DSO32 accelerometer and identified 10, 50 and 100 Hz as sampling frequencies to calculate the Apogee and bring out the impact of these sensor sampling frequencies on the Apogee estimation. I conducted a single flight of Estes Lil' Spite on AeroTech G74W-4 motor and configured the custom flight computer to log the data at 100 Hz. The data samples for 50 Hz and 10 Hz were derived from data log by down sampling. The Apogee calculation was derived from both the barometer and accelerometer observations with Jolly Logic AltimeterTwo measurements as independent check. I started with a hypothesis that the higher sampling frequencies would improve the apogee calculations significantly as I felt that the more readings, the better the Apogee estimate. However, to my surprise, I found that the barometer apogee calculations were barely impacted by the sampling frequencies as the ten 10 Hz estimates were within 0.3 m of the 100 Hz recordings. In case of accelerometer, while the average apogee was not impacted by the sampling frequencies, the range of possible values grew from 12 m at 50 Hz to 66 m at 10 Hz and it also impacted peak acceleration as lower sampling rates gave lower peak. Hence I concluded that the lower sampling rates impacted the precision more than the accuracy. I deduced that 20-50 Hz sampling frequency should be sufficient for barometer apogee while accelerometer needs 100 Hz sampling frequency for optimal results. I was glad to find that the barometer apogee reading at 100 Hz was within 1% of the AltimeterTwo reading.

## Introduction

The Apogee is the highest altitude that a model rocket can reach. The apogee is what the student rocketeer cares so much about; they care so much about the apogee because the apogee is tested in many competitions and research projects alike. In my experience with the Apogee, I have loved the thrill of building new rocket models that accommodate new motors, just so that I could have my rocket reach new apogees that I haven't reached before. But to reach new apogees, we need to be able to calculate accurately, which is also something many students worry about. For calculating apogee, we need to use accurate sensors.

However, there are many problems that come with calculating the apogee, but there are also some fixes as well: Most students who use sensors are of high cost, so low-cost sensors that use Arduino IDE and development kits are used to tackle that problem (Telaak & De Backer, 2023); as stated before, many students who are interested in rockets and hence participate in competitions such as the American rocketry challenge focus on reaching the target altitude. However, there has been a problem with sensors and how they sample apogee. Vandersteen et al. (2018) found that navigation must handle sensor uncertainties, so they made practical use of the Kalman filtering by combining the global navigation satellite system.

Sensors such as inertial sensors, which are the common norm for the sensors used today, are based on the MEMS technology, for MEMS sensors detect an external physical or environmental condition and transform that mechanical movement into a readable electrical output through a standard process (Shaeffer, 2013). MEMS barometers, although widely used, have their readings affected by sensitivity, accuracy, variability and the environment (Manivannan et al., 2020).

The early sampling theory states that the most obvious way to determine the distortion of telegraph signals is to calculate the transients of the telegraph system (Nyquist, 1928). Nyquist instead used the

alternative steady-state approach to discover that to transmit independent signals through a channel without interference, the max signalling rate can be at most twice that of the bandwidth of the channel. Shannon (1949) was able to use this theory in the reverse direction by proving that to capture a signal, a sample had to be faster than twice the highest frequency, and this is the sampling theorem. This implies that the '100 Hz' we will be discussing through this paper allows us to capture signal frequencies below 50 Hz. And hence, through the sampling theorem, we can infer that a 10 Hz rate could miss fast changes within the burn phase, such as ignition spikes and burnout. The need for the sampling rate is not just limited to model rockets, however, as we see its effects in other areas as well, such as navigation.

Sampling rates are used in integration methods seen in measuring inertial navigation: fast motion handled by high-speed updates feeding the slower ones (Savage, 1998). When acceleration data contains noise, numerical double integration acts as an amplifier of that noise, resulting in the final velocity and position calculations to be far apart from that of reality (Thong et al., 2004).

Low-cost flight computers often choose or are set up with a rate due to hardware limits or habit, such as the Featherweight Blue Jay, which has a sampling rate of 50 Hz (Contrail Rockets, n.d.). These flight computers have said rates based on hardware for the following reasons: Flight computers depend on a constant time step in order to calculate the apogee. If the time step varies, it will break down the derivatives and integrals used for said calculation, resulting in guidance errors. Software-driven loops can also shut down if the CPU is focused on multiple secondary tasks; lastly, fixed sampling rates dictate exact memory sizes, SPI/I<sup>2</sup>C bus transaction schedules and telemetry bandwidth so that the system never runs out of RAM or drops packets mid-burn. The motor used for this experiment, the Aerotech G74W-4, has a burn time of approximately 1.1 seconds (ThrustCurve.org, n.d.). During this time, around 10 readings would occur in one second, while 100 Hz would mean that there would be 100 readings that will occur for one second. Stepping out of the explanation for a bit, the reason for the many readings is that, with few readings, there could be data loss at the critical moments. This would affect the apogee accuracy, skewing our results or, worse, making them invalid. To my knowledge, few studies have measured the effect of sampling rate on apogee accuracy for model rocket flight computers, which this paper addresses.

This paper investigates the “effect of sensor sampling frequency (10 - 100 Hz) on Apogee estimation accuracy in a model rocket flight”, where I tried out and tested how changing the sampling rates affected the calculation of the apogee during a model rocket flight. To test this, I set up a flight computer to sample at 100 Hz and mounted it inside my model rocket for only one flight; afterwards, I used a program, `analyze_flight.py` (Varigonda, 2026), to calculate what the apogee would be if the sampling rate was 50 Hz and then what it might be if the sampling rate were 10 Hz. Finally, I compared the results to the apogee recorded by Jolly Logic ‘AltimeterTwo’ (LOC Precision, n.d.), which was also mounted in the model rocket. This process tested my hypothesis that "higher sampling rates will be able to record the apogee much better than slower sampling rates." This is because higher sampling rates will record the apogee more frequently than that of slower sampling rates.

## Methods

The key objective of this research is to measure the "effect of sensor sampling frequency on Apogee estimation accuracy”. In order to achieve the objective, the following factors have been identified in this experiment:

1. Independent Variable: The sampling rate, which was varied from 10 Hz to 50 Hz to 100 Hz.
2. Dependent Variable: The Apogee calculated by the barometer and the accelerometer present in the custom microcontroller at varying sampling rates and their error or gaps when compared with the AltimeterTwo apogee calculations.
3. Controlled variables: Using a single source of data from a single flight with the same rocket, same flight computer and sensors, I logged the data at 100 Hz and simulated the other sampling rates (50 Hz and 10 Hz) by keeping every 2nd and 10th reading, respectively.
4. The reference measurement was the Jolly Logic AltimeterTwo.

I did not choose separate flights for each of the sampling frequencies, as each flight is subject to multiple confounding variables such as wind speed, wind direction, and motor variation, all of which will affect the apogee. This would make it difficult to isolate the effect of sampling frequency on the apogee estimation. Instead, I opted for a single flight to record data at the highest frequency (100 Hz) and derived the lower frequencies from the sensor output. All three sampling rates come from the same flight conditions, so differences come only from the sampling frequencies.

The rocket used is the Estes 9737 Pro Series II Lil' Spite kit (Estes Rockets, n.d.), which has a length of 131.6 cm and a diameter of 5.08 cm. Estes lists a mass of the empty rocket of 292.0 g and my rocket with AltimeterTwo and recovery wadding weighed 355 g. After including a flight computer mass of about 25 g and motor weighing 87 g (ThrustCurve.org, n.d.), the final liftoff mass was roughly 467 g. The motor used for the flight is an Aerotech G74W-4 29mm \* 83mm (Composite Propellant Model Rocket Motor). The number of the vent holes is 4, and their sizes are roughly 0.419 cm each, and they are placed evenly on 4 sides of the upper tube of the rocket, right below the nose cone of the rocket where both the altimeter and flight computer are placed. The stability margin of the model rocket is 9.24 cal.

Tables 1 and 2 show the details of the materials used for building the model rocket.

**Table 1:**

*Rocket Components*

| S.no | Name                                    | Manufacturer | Qty |
|------|-----------------------------------------|--------------|-----|
| 1    | Estes 9737 Pro Series II Lil' Spite kit | Estes        | 1   |
| 2    | Jolly Logic AltimeterTwo                | Jolly Logic  | 1   |

|   |                                |          |        |
|---|--------------------------------|----------|--------|
| 3 | Pro Series II Recovery Wadding | Estes    | 1      |
| 4 | Aerotech G74W-4                | Aerotech | 1 pack |

Here is the table with the parts of the flight computer I built:

**Table 2:**

*Flight Computer Components and Key Specifications*

| No. | Name                                                                                            | Supplier     | Adafruit # | Qty | Key Specifications                                                                                                                                                             |
|-----|-------------------------------------------------------------------------------------------------|--------------|------------|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | Adafruit Feather<br>RP2040 Adalogger –<br>8MB Flash with<br>microSD Card –<br>STEMMA QT / Qwiic | Adafruit.com | 5980       | 1   | RP2040 dual-core ARM Cortex-M0+ at ~133 MHz; 264 KB RAM; 8 MB flash; onboard microSD slot for data logging; built-in LiPo charger; ~5.6 g<br><br>(Adafruit Industries, n.d.-b) |
| 2   | Adafruit BMP388 –<br>Precision Barometric<br>Pressure and Altimeter –<br>STEMMA QT / Qwiic      | Adafruit.com | 3966       | 1   | Bosch BMP388 sensor; relative accuracy $\pm 8$ Pa ( $\approx \pm 0.66$ m altitude); pressure range 300–1250 hPa; output data rate up to 200 Hz; I <sup>2</sup> C/SPI           |

|   |                                                                                   |              |      |        |                                                                                                                                                                           |
|---|-----------------------------------------------------------------------------------|--------------|------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   |                                                                                   |              |      |        | (Adafruit Industries, n.d.-a; Bosch Sensortec, 2020)                                                                                                                      |
| 3 | Adafruit LSM6DSO32<br>6-DoF Accelerometer<br>and Gyroscope –<br>STEMMA QT / Qwiic | Adafruit.com | 4692 | 1      | Accelerometer $\pm 4/8/16/32$ g; gyroscope $\pm 125$ to $\pm 2000$ °/s; accelerometer data rate 1.6 Hz–6.7 kHz; I <sup>2</sup> C/SPI<br><br>(Adafruit Industries, n.d.-c) |
| 4 | Lithium Ion Polymer<br>Battery – 3.7V 400mAh<br>(JST-PH connector)                | Adafruit.com | 3898 | 1      | 3.7 V, 400 mAh; $\sim 36 \times 17 \times 7.8$ mm;<br>8.2 g<br><br>(Adafruit Industries, n.d.-d)                                                                          |
| 5 | STEMMA QT / Qwiic<br>JST SH 4-pin Cable –<br>100 mm Long                          | Adafruit.com | 4210 | 2      | 100 mm, 4-pin JST SH; carries I <sup>2</sup> C data<br>and power between the boards                                                                                       |
| 6 | SanDisk Ultra<br>microSDHC 32GB<br>100MB/s Class 10<br>UHS-I                      | Amazon.com   | N/A  | 1      | 32 GB; up to 100 MB/s read; stores<br>flight data logs                                                                                                                    |
| 7 | Mounting Materials:<br><br>Adhesive Foam Padding<br>Sheet,                        | Amazon.com   | N/A  | 1 pack | Secure the flight computer in the<br>payload bay and cushion it against<br>vibration                                                                                      |

|  |                     |  |  |  |  |
|--|---------------------|--|--|--|--|
|  | Velcro brand strips |  |  |  |  |
|  | Cable zip ties      |  |  |  |  |

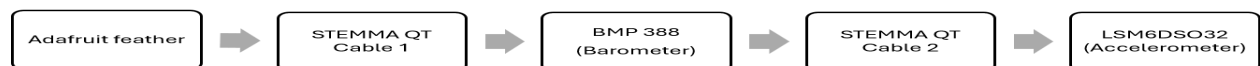
The accelerometer was configured to a  $\pm 32g$  range with an internal output data rate of 416 Hz and logged at 100 Hz; sampling internally at a higher rate than the logging rate to reduce aliasing in the recorded data. The barometer IIR filter was disabled so the data would remain raw, since filtering would smooth the measurements and hide the effect under study and the barometer was configured to no pressure and temperature oversampling.

The reference instrument was a Jolly Logic AltimeterTwo mounted in the same payload bay as the flight computer. According to the manufacturer, it samples pressure 25 times per second and acceleration over 250 times per second (LOC Precision, n.d.).

Using the STEMMA QT cables, the boards were connected in the following order:

**Figure 1:**

*Wiring Order of the Flight Computer Components*



To mount the microcontroller into the inside of the upper tube: The boards were mounted on a cardboard sled using foam, velcro and zipties, with all three boards and the battery on one strip; then I attached the top of the sled to the same nose cone loop with a ziptie, so the sled hangs straight down from the nose cone, inline with the rocket; Then, it was made snug in the tube by wrapping it in a band of foam around the sled so it slides into the body tube with a light friction fit; The AltimeterTwo was attached onto the same sled with a ziptie or Velcro; now both instruments sit in the same spot and read the same air.

**Figure 2:**

*Flight Computer Mounted on the Cardboard Sled*

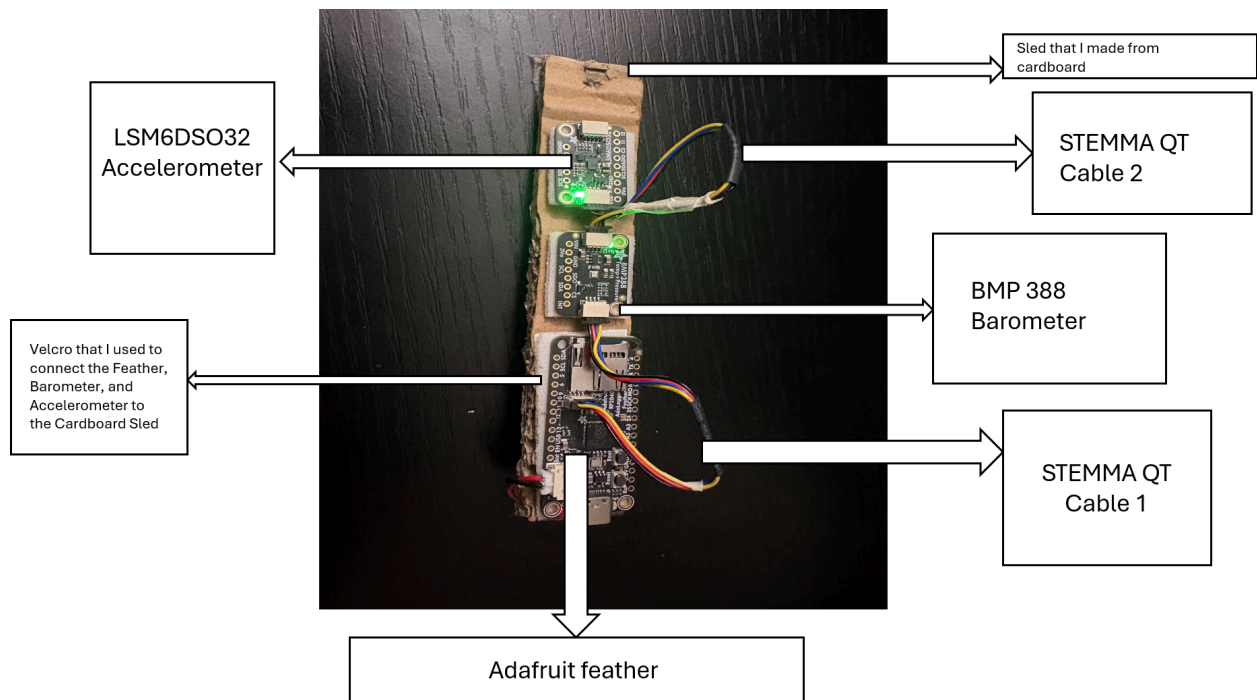
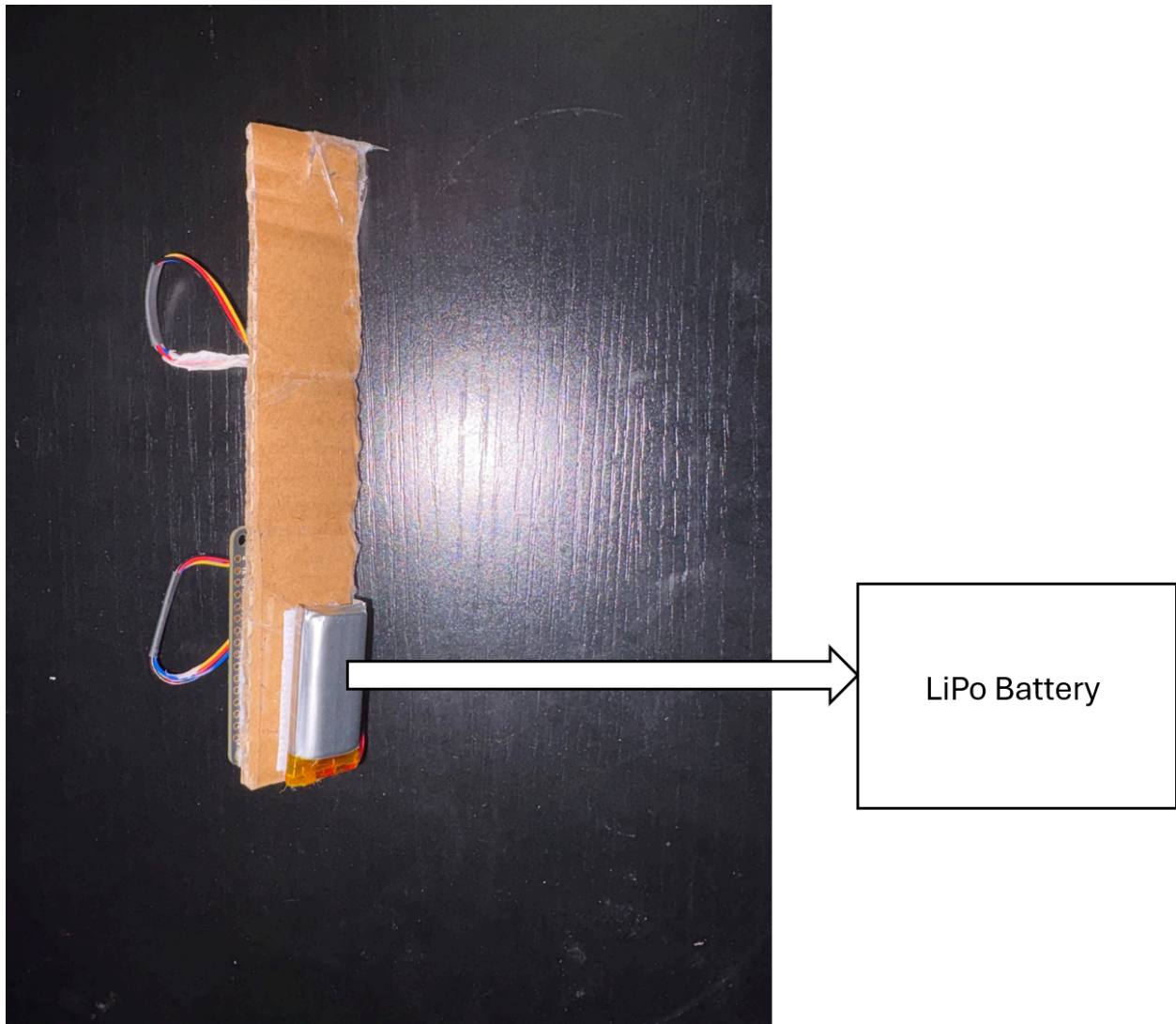


**Figure 3:**

*Flight Computer LiPo Battery Mounted on the Back Side of the Cardboard Sled*



The programming environment used for this is Arduino IDE 2 with the Philhower RP2040 Board package. The libraries used were the Adafruit BMP3XX, Adafruit LSM6DS, and SDFat (Adafruit fork).

To test whether the sensors were working, I used an Arduino file called 'i2c scan', which checks if the sensors are connected. The STEMMA QT cables have 4 wires: SDA, SCL, power and ground. The Adalogger feather is the controller, and the sensors are the ones that respond when called. Each sensor has a fixed address. The BMP388 barometer has 0x77, and the LSM6DSO32 accelerometer has 0x6A. The result shows if each sensor correctly responded or not. To make sure the readings made sense, I used the sensor test Arduino file. This file, when uploaded and run by the Adalogger, starts the sensors at basic settings used in flights: serial speed at 115200 baud, accelerometer range from -32 g to 32 g, and gyroscope range, accelerometer & gyro internal rate, barometer oversampling, barometer IIR filter, barometer internal rate, and I2C speed are set to the default of the sensor. Then the test reads the values for each repeatedly and then prints the values live to the serial monitor in Arduino IDE at 115200 baud. Nothing is saved to the SD card.

On the date of the flight, I had everything prepared. To make sure that the flight computer would take readings from the flight, I inserted the micro-SD card into the Adalogger feather and uploaded the flight\_logger.ino file the day before the flight. If the feather had a slow red-light blink on it, then I meant that it was executing the flight logger file and it was recording. On the day of the flight, I made sure to have the altimeter ready. I used a zip-tie to connect both the altimeter and the flight computer to the underside of the nose cone. To record the observable conditions of the launch, I brought a flight log I printed the day before and recorded the launch. After the flight and once I recovered the rocket, I turned off the altimeter because it sustains the log of every flight until you make it ready for a new flight. I then unplugged the LiPo battery which was connected to the Adalogger feather. To transfer the flight readings recorded by the flight computer, I plugged the microSD into the SD Card Reader, which gave the .csv file named LOG000.CSV immediately. This raw flight log is published in the GitHub repository as flight1\_raw\_100Hz.csv (Varigonda, 2026). The printed flight log that I brought on the day of the launch had the following: Weather conditions such as wind speed, wind direction and temperature; flight observations such as launch time, weathercocking, tilt and parachute deployment; and altimeter readings.

In order to fly at the launch, as per NAR and ROCI procedures, I was required to fill out a flight card and show that I am a junior member, all of which I completed.

Once the flight is completed, data analysis of the flight readings done by the flight computer must be completed. For this I used a Python script called `analyze_flight.py` and ran it on my laptop. For a ground reference, I averaged the readings 3 s to 0.5 s before launch, while the rocket sits still on the pad. This helps to give the ground pressure, which was set to 99,727 Pa (also called the zero altitude), and serves to give the at-rest accelerometer reading, which was set to 9.69 m/s<sup>2</sup>. To make sure the acceleration is only considered during launch, I only used reading which had a reading of above 3 g for 5 readings in a row. Since bumps from placing the rocket on the launchpad while the flight computer is active can spike above 3 g for 1 or 2 readings, a real motor keeps pushing for the whole burn. Once the acceleration drops below the at-rest 1 g reading, I understood that the burn phase of the rocket ended because after the burn phases, the rocket stops pushing and air drag slows the rocket. To indicate the ejection occurred, I only regarded the first jolt above 20 g for at least 0.5 seconds after the burnout. Because the parachute deployment creates a sharp jolt, it is fair to take 20 g and to skip any brief shaking right at burnout; I only deemed the readings that occurred for at least 0.5 seconds. One problem with the barometer is that each time the Adalogger feather asks for a reading from the barometer, the library returns a reading that is 10 ms old. This is fixed by the script giving the pressure value's timestamp when loading the log so that the barometer lines up with the accelerometer. To calculate the 50 Hz and the 10 Hz readings, the `analyze_flight.py` python script keeps every second row for the 50 Hz and keeps every 10<sup>th</sup> row for the 10 Hz. For calculating the altitude, I used the formula (Bosch Sensortec, 2013) derived from the U.S Standard Atmosphere (National Oceanic and Atmospheric Administration [NOAA] et al., 1976):

$$h = 44330 \left[ 1 - \left( \frac{p}{p_0} \right)^{0.1903} \right]$$

During the flight, some readings can be a noise spike, like  $\pm 0.5$  jitter, which is why fitting a curve through many readings near the top averages out the noise and finds the true peak. Also, wherever the slower logger's readings land is luck. Testing all of them shows the full spread and a fair average. For the

Barometer Apogee, a parabola is fitted to samples within  $\pm 1$  s of the highest reading. For the Accelerometer Apogee, we first subtract 1 g, and then we integrate it twice (Thong et al., 2004). Since acceleration is the double derivative of displacement, double-integrating acceleration gives up the displacement.

$$a(t) = \frac{d^2h}{dt^2}$$

*This then implies, through some more calculus,*

$$h(t) = \int_0^t \int_0^\tau a(s) ds d\tau$$

However, wider time steps miss fast changes in the burn, and integrating the acceleration twice will allow these errors to add up, which will cause the height error to grow. If the ejection occurs while the rocket is climbing to the apogee, the script will stop integrating at the ejection and add the remaining coast height. The climb will be mistaken for the ejection charge during the premature ejection, so integrating it would add false speed and ruin every later height, and after ejection the “up” axis no longer points up. At this the velocity head, found from  $\frac{v^2}{2g}$ , will be added to the height, but drag slows the rocket down faster, so this equation ignores drag, making the estimate too high.

Finally, the script also calculated the peak acceleration and the speed and altitude at burnout. The script reads the raw flight log (LOG000.csv) and saves every individual result to apogee\_results.csv.

## Results

The experiment took place on October 17<sup>th</sup>, 2026, at the ROCI event handled by the NAR at the AMA site in Muncie, Indiana. Only one flight was conducted because I thought it best to conduct only one due to the varying weather conditions. The rocket used was an Estes Pro Series II Lil' Spite (designed with co-partnership between Estes and Xyla Foxlin). The motor used for the flight was an Aerotech G74W-4, which has an engine burn time of approximately 1.1 secs (ThrustCurve.org, n.d.). From the LOG000.CSV file, which I obtained through the flight\_logger.ino that was uploaded into the microSD card. I found 364,599 rows of data collected over 61 mins, because the rocket stayed for a good chunk of the reading due to two misfires taking place that did not affect the change in altitude that was being measured by the flight computer. The readings occurred within a gap of no more than 14 ms, since it was being recorded at 100 Hz. In the file, the launch occurred as  $t = 2779.40$  s after the readings started, and the engine burn time came to 1.11 secs while the ejection was about 478 m, as reported by the AltimeterTwo, from the ground at  $t = 6.62$  secs after launch, all of which was recorded by the custom flight computer and analysed by analyze\_flight.py (Varigonda, 2026). The peak acceleration came to 17.0 g, which was measured by the analyze\_flight.py script after the flight, while the Jolly Logic AltimeterTwo measured it as 16.5 g. As per analyze\_flight.py the average acceleration during burn time was 14.1 g, and the burnout speed was 142.5 m/s. And finally, the barometer apogee after launch was reached after 8.4 s after launch, and the ejection of the parachute at 1.8 s before the apogee was reached.

Through analyze\_flight.py, I was able to get the following readings: (Note: since the 50 Hz readings were analysed through picking every 2<sup>nd</sup> row, starting from a range of 0 to 9, and the 10 Hz readings were analysed through picking every 10<sup>th</sup> row, starting from a range of 0 to 9, for the 50 Hz and the 10 Hz, the values appear in the form of ranges.)

1. Barometer Apogee:
  - a. 100 Hz: 503.9m

- b. 50 Hz: 504.0 m, 503.9 m
- c. 10 Hz: 504.0 m, 503.7 m, 503.9 m, 504.2 m, 503.9 m, 503.9 m, 503.9 m, 503.6 m, 504.0 m, 504.1 m
- 2. Accelerometer Apogee:
  - a. 100 Hz: 489.9 m
  - b. 50 Hz: 483.8 m, 495.9 m
  - c. 10 Hz: 483.9 m, 529.9 m, 501.3 m, 505.0 m, 505.1 m, 491.8 m, 464.1 m, 479.1 m, 466.2 m, 475.7 m
- 3. Peak Acceleration capture:
  - a. 100 Hz: 17.0 g
  - b. 50 Hz: 16.7 g, 17.0 g
  - c. 10 Hz: 16.4 g, 16.5 g, 16.7 g, 16.7 g, 16.5 g, 16.5 g, 16.4 g, 17.0 g, 16.6 g, 16.6 g
- 4. Burnout Speed:
  - a. 100 Hz: 142.5 m/s
  - b. 50 Hz: 142.3 m/s, 142.6 m/s
  - c. 10 Hz: 140.8 m/s, 144.0 m/s, 144.4 m/s, 143.8 m/s, 142.5 m/s, 140.5 m/s, 137.5 m/s, 138.5 m/s, 137.1 m/s, 136.8 m/s

The best number that summarises the 10 Hz result for the accelerometer apogee is the 490.2 m because it is the mean of the data values. Since the 10 Hz consists of a range of values from 464.1 m to 529.9 m, it is best if we take the mean of the range of values.

The readings from AltimeterTwo, the flight computer and the open rocket simulation prediction (OpenRocket, 2025) of the rocket's flight path and stats are compared in Table 3:

**Table 3:**

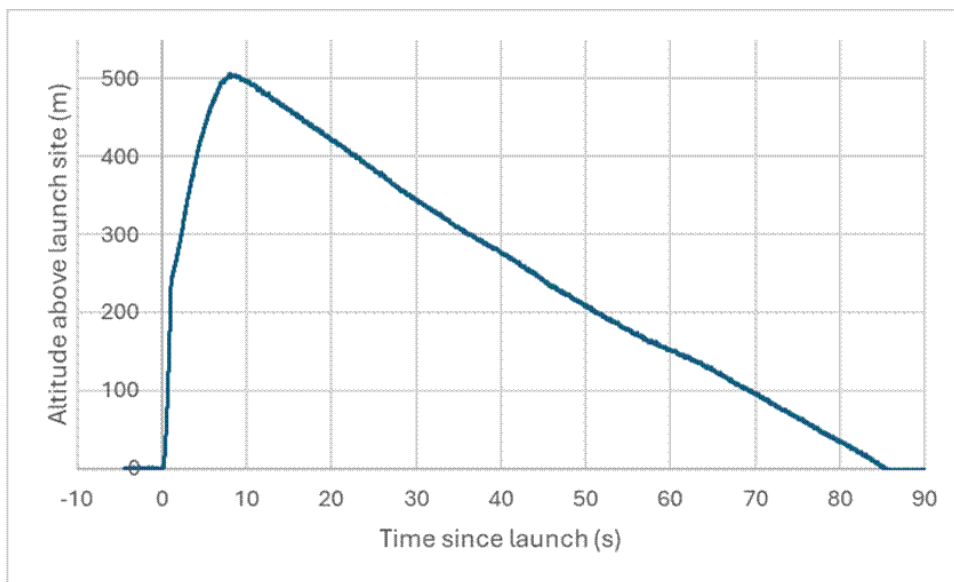
*Comparison of Flight Computer (100 Hz) vs. AltimeterTwo vs. OpenRocket Prediction*

| Measurement           | Flight Computer (100 Hz) | AltimeterTwo | Open Rocket Prediction |
|-----------------------|--------------------------|--------------|------------------------|
| Apogee (m)            | 503.9 m                  | 500 m        | 411 m                  |
| Top speed (m/s)       | 142.5 m/s                | 155 m/s      | 159 m/s                |
| Peak acceleration (g) | 17 g                     | 16.5 g       | 19.37 g                |

The LOG000.csv file was generated by the data logged by the flight computer, and I converted the LOG000.csv file from a .csv file to an .xlsx file, and I created the below graph depicted in Figure 4.

**Figure 4:**

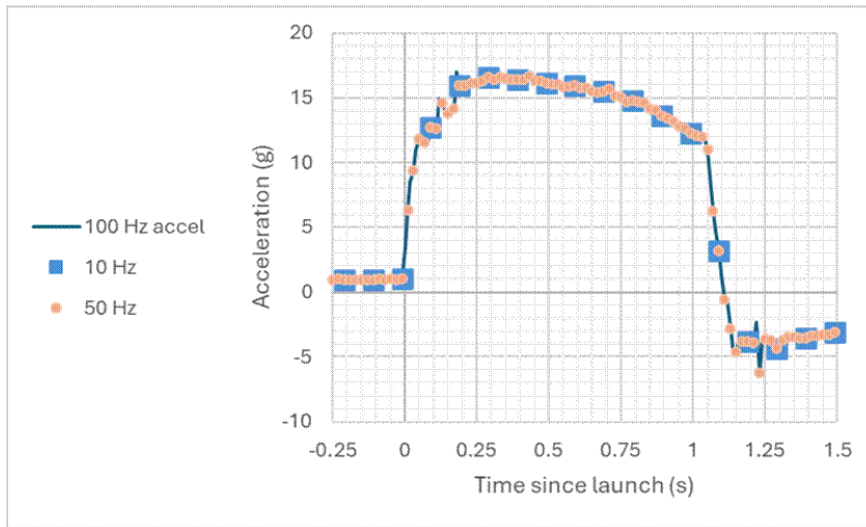
*Altitude Above Launch Site (m) vs. Time Since Launch (s)*



The altitude above the launch site is the same as the barometric altitude. It's also worth noting that the early steep shows the burn-phase error.

**Figure 5:**

*Acceleration During Motor Burn at 100, 50, and 10 Hz*

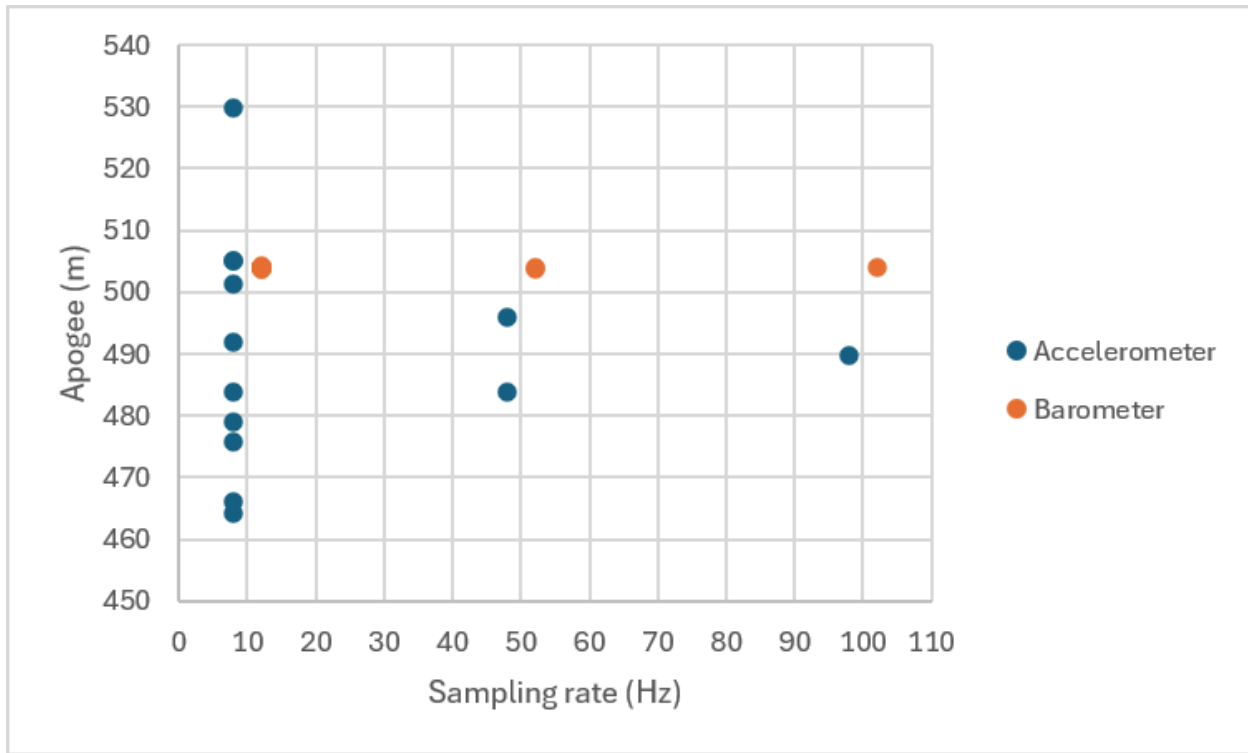


In Figure 5, the 50 Hz and the 10 Hz points show one possible down-sampled version each.

Figure 6 plots every individual apogee estimate at each sampling rate.

**Figure 6:**

*Apogee Estimates for Each Start Row at Each Sampling Rate*



*Note.* Each dot is one possible down-sampled version (one start row), with 2 versions at 50 Hz and 10 at 10 Hz, and the dots are shifted slightly sideways so they don't overlap.

## Discussion

Let's come back to my hypothesis, which was "Higher sampling rates will be able to record the apogee much better than slower sampling rates." This result mostly did not support my hypothesis because the accelerometer apogee barely changed between sampling rates (489.9 m at 100 Hz vs 490.2 m at 10 Hz), but the range of possible results grew from 12 m at 50 Hz to 66 m at 10 Hz, while the sampling rate made

almost no difference to the barometer apogee, as all ten 10 Hz estimates were within 0.3 m of the 100 Hz value. The thing that surprised me was that I thought that the 100 Hz would beat the 50 Hz and the 10 Hz by a landslide across both barometer apogee and accelerometer apogee, since more readings seemed like they should always be better; however, there is only a marginal difference in the average accelerometer apogee and barely any difference in the average barometer apogee. Now when we compare the Jolly Logic AltimeterTwo with the 100 Hz flight computer I built, the AltimeterTwo acts as an independent check on whether the flight computer reads sensibly. Moreover, AltimeterTwo readings are not absolute truth because it samples pressure at 25 Hz.

Near the apogee the rocket is slow, which causes the barometer reading for the altitude to slowly change, with even the 10 Hz readings catching the apogee. Since acceleration is the double derivative of distance, i.e., to get the distance, we need to integrate twice (Thong et al., 2002). This means that the accelerometer apogee depends on integrating twice and would have brought missed/extra samples during fast changes into height (Thong et al., 2004).

One thing to notice is that as the sampling rate decreases, while the average nearly stays the same for all the sampling rates, the ranges of the possible apogee predictions grow (see Figure 6).

**Figure 7:**

*Apogee, Peak Acceleration and Burnout Speed at Each Sampling Rate (analyze\_flight.py Output)*

```
File: LOG000.CSV  rows: 364599
Launch t = 2779.40 s  burn time = 1.11 s  ejection 6.62 s after launch
Up axis: ax_mps2 (sign +1); 1 g on the pad read as 9.69 m/s^2
```

| Rate   | Baro apogee (m)     | Accel apogee (m)    | Peak accel (g)   | Burnout speed (m/s) |
|--------|---------------------|---------------------|------------------|---------------------|
| 100 Hz | 503.9 (503.9-503.9) | 489.9 (489.9-489.9) | 17.0 (17.0-17.0) | 142.5 (142.5-142.5) |
| 50 Hz  | 503.9 (503.9-504.0) | 489.9 (483.8-495.9) | 16.8 (16.7-17.0) | 142.5 (142.3-142.6) |
| 10 Hz  | 503.9 (503.6-504.2) | 490.2 (464.1-529.9) | 16.6 (16.4-17.0) | 140.6 (136.8-144.4) |

Numbers in brackets = lowest-highest over every possible starting row.  
Every individual result was saved to apogee\_results.csv

As seen in Figure 7, which is the output of the analyze\_flight.py (Varigonda, 2026) script, we can see that the accelerometer apogee ranges have increased dramatically from the 50 Hz sample to the 10 Hz sample.

A possible cause for this might be that for the 10 Hz, the time steps are ten times wider. The time step is the time difference between two consecutive readings taken. Since the time steps occur when we take the reciprocal of the sampling rate, the time step of 10 Hz is therefore  $1/10 = 0.1$  s (100 ms), which may be the cause of the wide range during a burn phase where the acceleration is high. As for why the average stayed the same, using Figure 7 to ascertain the possible values, we get that the average stayed the same because each 10 Hz version uses different readings so that there is an overestimation or underestimation, and these errors cancel each other when averaged. However, the peak acceleration was the exception since a slower rate can miss the true peak, and hence it dropped from 17.0 g to 16.6 g. Still, a real 10 Hz flight computer would only get one of these versions, so its apogee could be off by up to 40 m depending on when its readings happened to fall.

So, lowering the sampling rate did not make the apogee estimate biased, and hence the accuracy stayed the same.

The barometer has a slow signal, so the theorem is easily satisfied. The barometer apogee comes from the top of the flight. From Figure 4, the top takes roughly 2-10 seconds to change from climbing to falling. Converting that into frequency, it gives us a range of 0.1-0.5 Hz. The accelerometer, however, is a fast signal, so during the burn, the acceleration changes very quickly. This violates the theorem at low rates (Nyquist, 1928). Sampling rate is a trade-off in other fields too (Khan et al., 2016).

So, for a student choosing a sampling rate, the student should sample the barometer at about 20-50 Hz and the accelerometer at 100 Hz.

Through the LOG000.CSV file, the barometer altitude was 241 m at burnout vs. the 82 m at burnout from the accelerometer. The likely cause could be that the fast airflow over the vent holes lowers pressure inside the tube, which reads as 'higher'. The reason why these matter for the research question is because the barometer is unreliable during rapid acceleration at any sampling rate, which means the accelerometer

is the right sensor for the burn phase. So, combining the barometer and the accelerometer helps cover each other's weaknesses (Sabatini & Genovese, 2014).

From the table where I showed the top speed from the flight computer and the AltimeterTwo, the top speed calculated by the flight computer at 100 Hz was 142.5 m/s, while the top speed calculated by the AltimeterTwo was 155 m/s. This difference in the readings results in the flight computer readings for top speed being 8% less than the AltimeterTwo. A possible cause for this is that when the sensor was placed on the launchpad, it calculated that the acceleration due to gravity value (g value) was at  $9.69 \text{ m/s}^2$  instead of the  $9.81 \text{ m/s}^2$ , but this calibration offset is only a part of the gap. The altimeter's speed method isn't published and may be barometer-based, which would read high during the burn. Another possible explanation could be weathercocking or tilt of the rocket. Similarly, the accelerometer apogee is 2% less than the apogee calculated by the AltimeterTwo, with the accel apogee being at 489.9 m vs the 500 m apogee calculated by the AltimeterTwo. Ejection occurred 1.8 s before apogee, and the G74W-4 delay (4 s) is shorter than the Open Rocket optimum (7.07 s), as shown in the open rocket simulation results in Figure 8.

**Figure 8:**

*OpenRocket Simulation Results*

| W...  | Name         | Configuration | Velocity off rod | Apogee | Velocity at depl... | Optimum delay | Max. velocity | Max. accelerat... | Time to apogee | Flight time | Ground hit velo... |
|-------|--------------|---------------|------------------|--------|---------------------|---------------|---------------|-------------------|----------------|-------------|--------------------|
| ✓ 1 ⚠ | Simulation 2 | [G74W-4]      | 17.4 m/s         | 411 m  | 33.2 m/s            | 7.07 s        | 159 m/s       | 190 m/s²          | 6.13 s         | 64.7 s      | 7.24 m/s           |

Because of early ejection, accel apogee = altitude at ejection +  $v^2/2g$ . As said before in the method section, the coast step ignores drag, adding about 6 m too much at 100 Hz. The ejection also came early, at 1.8 seconds before the apogee, so the accelerometer couldn't be measured to the top. At 10 Hz, the integration errors in speed get squared in the coast step, widening the spread.

OpenRocket predicted 411 m vs 500 m measured by the altimeter. This could have been caused either by the mass or CG settings that did not match the real rocket, the surface finish setting, motor variation or default weather and launch conditions.

Now which of these affects my conclusion? Top speed isn't what we are comparing, so it is irrelevant to our conclusion. It is best if we split the ones that do affect my conclusion into two groups: factors which change the conclusion and factors that shift the numbers but not the comparison.

Factors which change the conclusion: one flight, thinned data vs. real slower hardware, early ejection amplifying the 10 Hz spread, and 100 Hz being treated as the reference.

Factors that shift the numbers but not the comparison: calibration, tilt, temperature, the OpenRocket prediction and AltimeterTwo rounding.

But if I were to fly the experiment once again, I would address these where possible.

## Limitations

However, there are limitations to this research/experiment:

1. Due to flying only once, I have limited the number of trials I can use to confirmably state that the conclusion that we have reached is always true.
2. Decimation imitates slow hardware, and real slow hardware may filter differently.
3. As per my design, I only used one rocket, one sensor set and one motor type, which doesn't account for the fact that other rockets, where some have a surface that creates more laminar or turbulent flow of air than mine, could change, even slightly, the data points recovered by the flight computer. Also using a motor like the G74W-4, which is not a high-powered motor, and taking out any better comparison that I can make between high-, mid- and low-powered rockets.
4. I kept every 10th reading without filtering first, which may have let fast vibration show up as false slow changes.
5. AltimeterTwo samples pressure at 25 Hz and may smooth its data, so differences of a few metres may fall within this error.

## Conclusion

Coming back to the objective of the paper, which is “Effect of sensor sampling frequency (10-100 Hz) on Apogee estimation accuracy in a model rocket flight”, my results, as stated before, mostly did not agree with my hypothesis: the higher rates were much more consistent for the accelerometer, while the barometer was barely affected. The sampling rate had very little effect on the barometric apogee; all ten 10 Hz estimates were within 0.3 m of the 100 Hz value because altitude changes slowly near apogee. However, the sampling rate had a large effect on the accelerometer apogee, which had a range of 12 m at 50 Hz and 66 m at 10 Hz, leading me to the conclusion that lower sampling rates lower precision, not accuracy, in the case of the accelerometer apogee. During the burn phase, the barometer was unreliable because of pressure effects, so the fast accelerometer sampling is more important in this phase. The accelerometer integrates twice through a fast burn; this causes the errors to build up alongside the fact that wider time steps miss rapid changes. At 100 Hz, my flight computer agreed with the AltimeterTwo within about 1% on barometer apogee (503.9 m vs. 500 m) and within 0.5 g on peak acceleration.

Based on this flight, for the barometric apogee, use around 20-50 Hz, but for the accelerometer-based apogee, use at least 100 Hz (especially during the burn).

Looking towards the future, I believe it is better to include multiple flights, log natively at each sampling rate rather than downsampling, use a GPS or second reference altimeter, and use a motor with a longer delay to avoid early ejection, as well as combine barometer and accelerometer data through sensor fusion.

## Code and Data Availability

I have uploaded all the code and data used in this study and made them publicly available at <https://github.com/sriharshavarigonda-coder/sampling-rate-study> (Version 1.2; Varigonda, 2026) under

the MIT License. Included in the GitHub repository are the flight logger firmware (flight\_logger.ino), the sensor test and I<sup>2</sup>C scan sketches as well as the analysis script (analyze\_flight.py), the raw 100 Hz flight log (flight1\_raw\_100Hz.csv, originally LOG000.CSV), and the Excel spreadsheet that was used to create the figures.

## Acknowledgements

I thank those who helped me conduct my experiment, such as my parents, who bought the materials for me, and the ROCI members at the NAR launch site in Muncie, Indiana, who allowed me to launch the model rocket I built.

## AI Use

The flight firmware code and analysis code were written with the assistance of Claude (Anthropic). AI was also used for feedback on the structure and wording of this paper; all the text was written by me. I designed the experiment, built and flew the rocket and flight computer, tested the code, and interpreted the results.

## References

Adafruit Industries. (n.d.-a). *Adafruit BMP388 – Precision barometric pressure and altimeter* [Product page]. Retrieved October 15, 2026, from <https://www.adafruit.com/product/3966>

Adafruit Industries. (n.d.-b). *Adafruit Feather RP2040 Adalogger – 8MB flash with microSD card – STEMMMA QT / Qwiic* [Product page]. Retrieved October 15, 2026, from <https://www.adafruit.com/product/5980>

Adafruit Industries. (n.d.-c). *Adafruit LSM6DSO32 6-DoF accelerometer and gyroscope – STEMMMA QT / Qwiic* [Product page]. Retrieved October 15, 2026, from <https://www.adafruit.com/product/4692>

Adafruit Industries. (n.d.-d). *Lithium ion polymer battery – 3.7V 400mAh* [Product page]. Retrieved October 25, 2026, from <https://www.adafruit.com/product/3898>

Bosch Sensortec. (2020). *BMP388 digital pressure sensor* (Document No. BST-BMP388-DS001, Rev. 1.7) [Datasheet]. <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp388-ds001.pdf>

Bosch Sensortec. (2013). *BMP180 digital pressure sensor* (Document No. BST-BMP180-DS000-09, Rev. 2.5) [Datasheet]. <https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf>

Contrail Rockets. (n.d.). *Featherweight Blue Jay altimeter* [Product page]. Retrieved October 25, 2026, from <https://contrailrockets.com/product/featherweight-blue-jay-altimeter>

Estes Rockets. (n.d.). *Estes Pro Series II Lil' Spite flying model rocket kit* [Product page]. Retrieved October 15, 2026, from <https://estesrockets.com/products/lil-spite>

Khan, A., Hammerla, N., Mellor, S., & Plötz, T. (2016). Optimising sampling rates for accelerometer-based human activity recognition. *Pattern Recognition Letters*, 73, 33–40. <https://doi.org/10.1016/j.patrec.2016.01.001>

LOC Precision. (n.d.). *Jolly Logic AltimeterTwo* [Product page]. Retrieved October 15, 2026, from <https://locprecision.com/products/jolly-logic-altimetertwo>

Manivannan, A., Chin, W. C. B., Barrat, A., & Bouffanais, R. (2020). On the challenges and potential of using barometric sensors to track human activity. *Sensors*, 20(23), Article 6786. <https://doi.org/10.3390/s20236786>

National Oceanic and Atmospheric Administration, National Aeronautics and Space Administration, & United States Air Force. (1976). *U.S. standard atmosphere, 1976* (NASA-TM-X-74335). <https://ntrs.nasa.gov/citations/19770009539>

Nyquist, H. (1928). Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2), 617–644. <https://doi.org/10.1109/T-AIEE.1928.5055024>

OpenRocket. (2025). *OpenRocket* (Version 24.12) [Computer software]. <https://openrocket.info/>

Sabatini, A., & Genovese, V. (2014). A sensor fusion method for tracking vertical velocity and height based on inertial and barometric altimeter measurements. *Sensors*, 14(8), 13324–13347. <https://doi.org/10.3390/s140813324>

Savage, P. G. (1998). Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms. *Journal of Guidance, Control, and Dynamics*, 21(1), 19–28. <https://doi.org/10.2514/2.4228>

Shaeffer, D. K. (2013). MEMS inertial sensors: A tutorial overview. *IEEE Communications Magazine*, 51(4), 100–109. <https://doi.org/10.1109/MCOM.2013.6495768>

Shannon, C. E. (1949). Communication in the presence of noise. *Proceedings of the IRE*, 37(1), 10–21. <https://doi.org/10.1109/JRPROC.1949.232969>

Telaak, J. T., & De Backer, W. (2023). Designing an Arduino-based rocket flight computer for embedded systems education. In *AIAA Regional Student Conferences*. American Institute of Aeronautics and Astronautics. <https://doi.org/10.2514/6.2023-71965>

Thong, Y. K., Woolfson, M. S., Crowe, J. A., Hayes-Gill, B. R., & Challis, R. E. (2002).

Dependence of inertial measurements of distance on accelerometer noise. *Measurement Science and Technology*, 13(8), 1163–1172. <https://doi.org/10.1088/0957-0233/13/8/301>

Thong, Y. K., Woolfson, M. S., Crowe, J. A., Hayes-Gill, B. R., & Jones, D. A. (2004).

Numerical double integration of acceleration measurements in noise. *Measurement*, 36(1), 73–92. <https://doi.org/10.1016/j.measurement.2004.04.005>

ThrustCurve.org. (n.d.). *AeroTech G74W*. Retrieved October 25, 2026, from

<https://www.thrustcurve.org/motors/AeroTech/G74W/>

Vandersteen, J., Bennani, S., & Roux, C. (2018). Robust rocket navigation with sensor uncertainties: Vega launcher application. *Journal of Spacecraft and Rockets*, 55(1), 153–166.

<https://doi.org/10.2514/1.A33884>

Varigonda, S. H. V. (2026). *Sampling-rate study* (Version 1.2) [Computer software]. GitHub.

<https://github.com/sriharshavarigonda-coder/sampling-rate-study>