

ATTITUDE-BASED AIR-START IGNITION CONTROL SYSTEM FOR HIGH-POWER ROCKETRY

Frank B. Hermes, NAR 88930 TRA 12122 L3 and William Premerlani, PhD - Jan 26th, 2011

Background

When I first got back into model rocketry (January 2009), I was enamored with the high power rockets I witnessed at my first launch. However, I was less impressed by the 2-stage rockets that I saw. Most of them either did not ignite the second stage, or they flew in crazy arcs or blew up or did not fly at all. Most of the more successful ones were the smaller, sport-rocket types that seemed to work okay using simple instant, motor-to-motor ignition.

Somewhere around the time I was working on the build of my Level 2 project and reading like mad to try and assimilate all that I could to be sure things worked correctly, I came across G. Henry and Bill Steins' book, *Handbook of Model Rocketry*. In it they discussed several aspects of staging, but again I did not pay much attention at the time, nor did I when, sometime later, I read Mark Canepa's excellent book *Modern High-Powered Rocketry 2*, only glancing at the sections on staging.

One day when I was searching for some information in Tim Van Milligan's *Apogee Newsletter* archives, I stumbled across an article on staging. In it Tim discussed how to achieve maximum altitude. He was discussing a *coasting period* between booster burn out (BBO) and sustainer ignition...hmmm...what he was saying contrasted with what I remembered reading in Steins' book. I remembered vaguely that Stein advocated performing sustainer ignition right after BBO, in this way you would be coupling the maximum velocity of the booster driven rocket with the maximum velocity of the sustainer rocket in order to achieve maximum altitude. However, when I went back and reread Stein, I realized he qualified his statement and indicated he was "neglecting aerodynamic drag". I was a bit confused at this point so I went back and reread Canepa – he drew the distinction of attempting to achieve maximum velocity versus maximum altitude and that you needed two very different approaches.

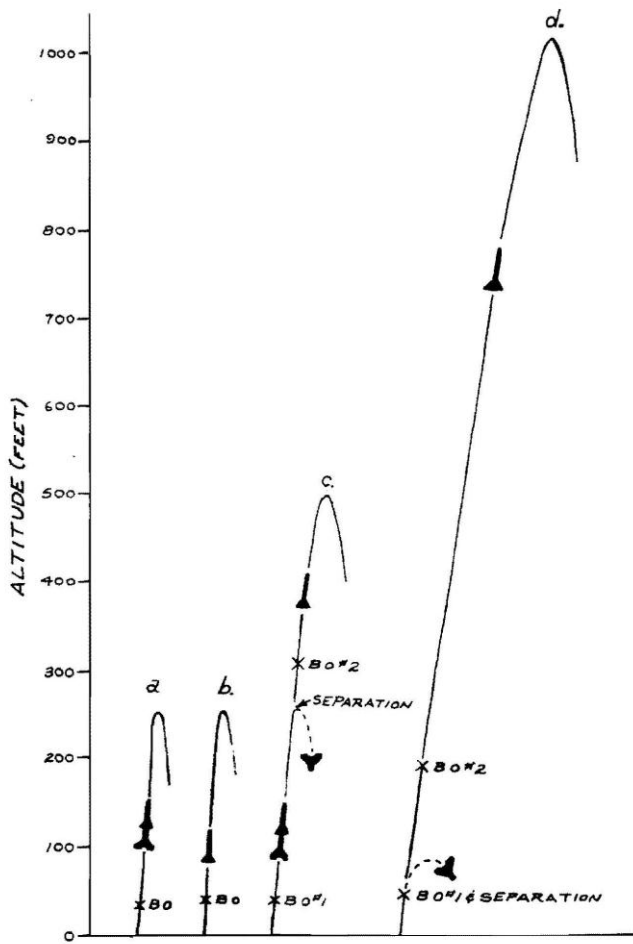


FIGURE 11-3 Series staging at various points in a model rocket flight. (a) Lower stage only ignites. (b) Top stage only. (c) Model stages at peak altitude of lower-stage flight. (d) Model stages at lower-stage burnout (BO) point and therefore at maximum-stage velocity.

Graphic from: *Handbook of Model Rocketry*

Van Milligan pointed out that Stein's instant ignition would achieve maximum altitude only if you were in a vacuum since parasitic drag is induced by our atmosphere. In fact, such drag is proportional to the *square* of the velocity, so the higher velocity gained by instant coupling is precisely the wrong approach to maximize altitude. What Canepa, Van Milligan and others point out is that *minimizing* velocity, and therefore drag, will generally achieve the higher altitude for the same total amount of thrust.

My simulations in RockSim show an average increase in altitude of about 30+% when the coast period is maximized verses igniting the sustainer motor immediately after booster motor burnout. That is a significant difference.

Since I was developing this urge to go high and realizing that staging was one way to accomplish this, I started thinking more about the coast period between sustainer ignition and BBO. I was intrigued. Just

how would you go about maximizing the coast period? After a successful Level 3 certification (June 2009), I began to wonder just where to go next. Maybe staging could prove interesting – it obviously was very challenging.

From what I could determine, most high power staging ignition is done with timers. But that seemed rather crude to me. If my goal is to achieve maximum altitude, how much time should I dial in to maximize the coast period? I could guess at how much time to allow after booster burn out. Or I could run simulations in a program such as Apogee's *RockSim*, or, and probably the most expensive route, I could just experiment with test flights. However, to really attempt to maximize the coast period I would have to have something better than a good guess. Even if I were able to perform a very accurate estimate of the coast period, it would still be assuming the actual flight went exactly according to my assumptions. How would I allow for such things as inconsistent motor burns, erratic flight patterns, the weather and other assorted conditions that I could not reliably predict?

I thought about current day electronic altimeters. Prior to altimeters being adapted for use in high powered model rockets, most deployment was also based on timers, either the relatively simple motor ejection charge delay used as the igniter, or mechanical or electric timers. The basic problem with timers is that they are not dynamic in nature. That is, if everything goes according to plan in regards to a flight's performance, then the timer works well. However, as we have often seen out on the launch site, things do not always go according to plan – maybe seldom is a better word! For example, if a motor's burn is shorter than expected, or produces less thrust than expected, apogee will occur sooner than expected. Any delay that was designed into a timer to deploy at apogee will be later than desired and deployment will not occur at the slowest point in the rocket's flight, but rather at some higher speed as it is accelerating on its downward path. However, if an altimeter was used in that same scenario, it would be looking for the apogee event in real-time, based upon its sensor readings, not a prescribed time – it is able to *dynamically* adjust the deployment point to suit the actual flight conditions incurred.

I wondered if I would be able to develop some sort of sensor system that was analogous to the altimeter to dynamically maximize the coast interval. Could I find something to replace a simple timer for staged ignition?

The Problem

So what factors or issues would I need to consider in order to replace the staging timer? What is it about the rocket's flight that characterizes an ideal coast period anyway? Just how would you describe to someone what has to happen between booster burnout (BBO) and a point in the flight path that would be considered as the longest possible coasting interval that would result in the rocket achieving maximum altitude?

In the ideal world, after BBO, your sustainer rocket would fly perfectly vertical all the way to what would be apogee, but, just prior to it "falling over", you would ignite an "instant on" motor that would come up to pressure immediately and send the sustainer hurtling upward.

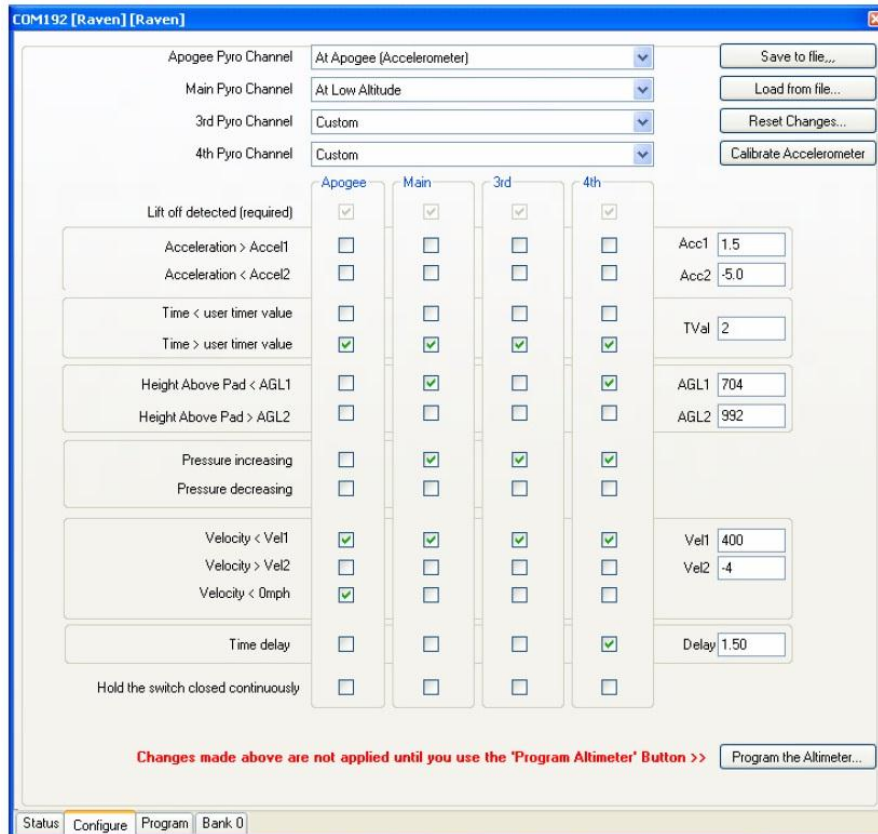
From a more realistic point-of-view, we want to initiate sustainer ignition just at the point that we have enough time to bring the motor up to pressure while the rocket is still flying fast enough to maintain its “relatively” vertical attitude. So, we need to be able to monitor both velocity and verticality on a real time basis and make decisions based upon that information.

Aside from the fact that seldom does a rocket fly vertical all the way to apogee, we never know for sure ahead of time how long it will take a rocket to reach apogee due to variances in real performance versus specifications or simulations, or the current weather, booster motor performance, etc. If we could characterize and determine or predict what that value was however, such a predictive apogee sensor system would offer a *dynamic* trigger point, analogous to an altimeter.

Looking at it from the velocity aspect, if we knew that our magic rocket would fly vertical all the way to apogee, i.e., where it reaches zero velocity, we could pick a velocity point, e.g., decreasing through 200 MPH, prior to that which allowed us to pressurize the motor in time before the rocket arrived at apogee. Many of the popular commercial altimeters, or *flight computers*, out there today have the ability, either through on board accelerometers and their micro-controllers, to be calculating velocity in real-time. So, if we knew we would stay vertical throughout the coast period, we could fairly easily determine through flight test trial-and-error or simulation, a velocity-based ignition trigger point.

Verticality

Adrian Adamson at Featherweight Altimeters has provided some access to verticality and velocity as a trigger with both his Parrot altimeter and the newer Raven. With the Parrot or Raven you can access a parameter in the Featherweight Interface Program (FIP) “decreasing through nnn MPH” as an event to control the output of any of the output/pyro channels...



Featherweight Altimeters - Raven™

Additionally, flight computers such as the Raven can offer a few more indicators that the rocket is probably still flying upward. Equipped with an accelerometer and timer, as well as a barometer, such flight computers can assure that the pressure is still decreasing (lesser air pressure as you ascend) and that the rocket is at a predictable/configurable minimum altitude after a certain amount of time – all indications that the rocket is still ascending. Assuming that the flight is perfectly vertical all the way to apogee, use of one of these “vertical-check” flight computers is basically all you need.

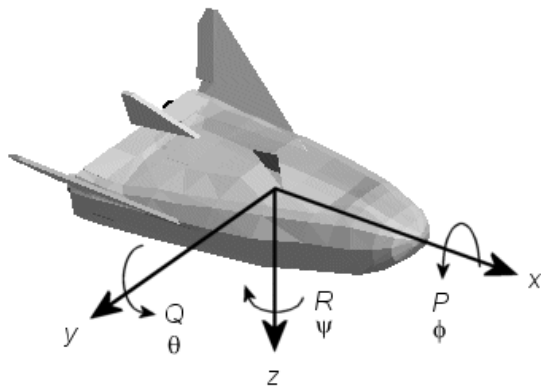
However, a more likely scenario is that the rocket will not fly perfectly straight and will be arcing over to some degree or another. What is missing from the vertical-check flight computer is a monitoring of the angle off the vertical that the rocket is experiencing.

The rocket may still be ascending per the vertical-check parameters, but do you want to trigger ignition if the rocket is flying at an angle of say, 30 degrees off vertical? What about 45 - or even 60? High altitude recovery is tough enough without setting yourself up for a very long retrieval or loss of the rocket, or even worse it occurred to me about this time, a flight into the crowd. To be in even a more dynamic triggering environment, we need to be able to measure the angle off vertical, or tilt, in conjunction with the other information before deciding to trigger sustainer ignition.

Tilt

How do you determine tilt? Or maybe better – what is tilt as it applies to a rocket.

Envision some sort of tilt detectors placed in the avionics bay of a rocket sitting on the launch pad. You are facing the rocket, which is perfectly vertical, and looking straight ahead to the north. Though there are no pre-defined absolute model rocket coordinate reference systems, for purposes of illustration, we will refer to the axis that runs through the rocket from south to north as the X-axis, and the one that runs through it from east to west as the Y-axis. The Z-axis of the rocket runs through the middle of it from top to bottom. This will be our rocket's X, Y, and Z coordinate frame of reference, and it is for our purposes defined with respect to the earth. So, with our rocket sitting vertical on the pad, we are arbitrarily defining that the rocket's and the earth's two reference systems coincide, and tilt would be a deviation from one or more of those axis.



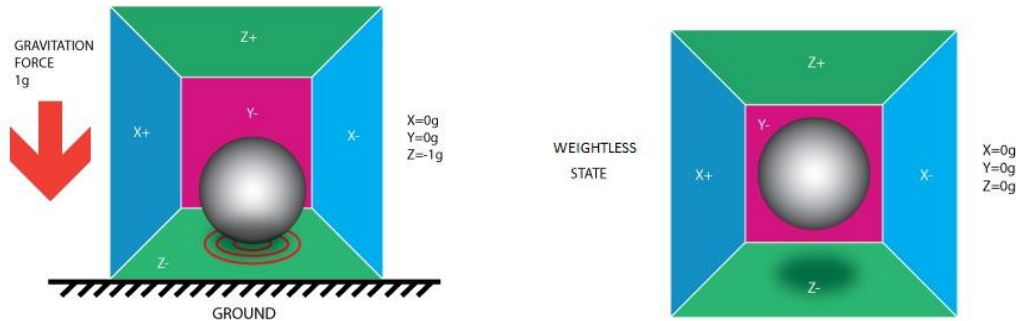
Aircraft Coordinate System

Initially I considered mechanical tilt meters, such as you might find on “tilt-over” shipping containers to record any damage or adverse handling during the shipping process or maybe recreational vehicle levelers. A company named Rieker is one manufacturer of such devices. I ordered a couple of their tilt meters to trigger an output whenever they exceeded 15 degrees of tilt. I placed two of the one-axis sensors one atop the other and rotated one by 90 degrees. So now I had a two-axis, X and Y tilt meter that gave me an output signal whenever the tilt exceeded the 15 degrees in any direction.

Gravity

I discussed my plans with a physicist friend of mine. I explained to him what I was doing. He asked me to explain the specific technology that the Rieker units were based upon. I did not know exactly so I went to the Rieker web site and deduced (never confirmed) that the devices contained a upward-pointed curved tube running along the orientation axis with some viscous substance to dampen out the travel of a ball that would travel inside the tube and cause something to trip once the prescribed angle was

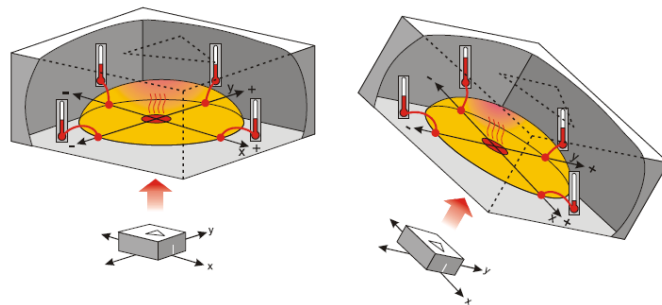
reached. He thought about things for a while and made me realize that during the period of interest, i.e., coasting, the sensors would be experiencing negative gravity - as soon as booster burnout occurs, the positive g force the rocket was under quickly switches to negative g. The ball then, rather than staying in the bottom of the tube, as it should, would be floating up (think of yourself floating in a falling elevator whose cable had become unattached) and very prone to travel up the curved tube as soon as the slightest tilt occurred. That is when a serious recognition on my part of the role of gravity began to enter the picture.



I began searching for something more “solid-state” and discovered that you can derive a tilt sensor from an accelerometer. I obtained a Memsic 2125 2-axis (X and Y) accelerometer that used a temperature differential inside a closed chamber to sense changes in gravity to its orientation. I started thinking that maybe this was the answer to the concerns about the mechanical “rolling balls in a tube”.

Theory of Operation

The MX2125 has a chamber of gas with a heating element in the center and four temperature sensors around its edge. When the accelerometer is level, the hot gas pocket rises to the top-center of the chamber, and all the sensors will measure the same temperature.



By tilting the accelerometer, the hot gas will collect closer to some of temperature sensors. By comparing the sensor temperatures, both static acceleration (gravity and tilt) and dynamic acceleration (like taking a ride in a car) can be detected. The MX2125 converts the temperature measurements into signals (pulse durations) that are easy for microcontrollers to measure and decipher.

MicroControllers

While I was identifying a sensor to use, I knew that eventually I would have to be able to use the output of whatever sensors I decided on. I came across the Parallax web site and saw that they offered a tutorial starter kit for microcontrollers. A microcontroller is a device that includes a microprocessor (a small version of the microprocessor in your home or office computer) along with other embedded peripheral devices such as input and output ports, power regulators, and memory.

It was not too much of a stretch to become familiar with microcontrollers and the analog to digital conversion necessary to work with certain sensors. The parallax course was very good and I also took a Saturday morning microcontroller course at the local junior college to get a more in depth experience.

Eventually, I assembled the MX2125 circuit, connected it to the microcontroller, did a bit of programming, fired it up and it worked great. I set it up to trigger an output whenever the absolute value of the tilt angle exceeded 15 degrees on either the X or Y-axis.

Gravity, Another Warning

About this same time, I was at a ROC launch in Lucerne Valley prepping my 2-stage for a test flight (February 2010). A fellow wandered over and watched while I was assembling one of the Roush Tech CD3 CO2 charges I was using for deployment. I mentioned that I was investigating maximizing the coast interval and told him about my Rieker units and the Memsic accelerometer-based tilt meter I had recently created. He paused for a moment and shook his head. I asked him what was wrong. His reply was that any sensor that is affected by gravity would not work. Later, when discussing the topic again with my physicist friend, he had come to the same conclusion.

The problem gets back to the two different frames of reference I mentioned previously – the rocket's coordinates and the earth's coordinates. When the rocket is sitting on the pad vertical, the two systems are aligned. It is the earth's frame of reference that is important, since it is in that frame of reference that tilt has its basis – i.e., the 15 degrees of tilt that is of interest is in the earth's frame of reference relative to the Z axis (verticality).

The Rieker unit and accelerometers, such as the Memsic, by definition sense gravity. Acceleration sensed while the rocket is on the pad and acceleration due to the rockets movement in flight are indistinguishable to the sensor. Once the rocket leaves the pad it loses its reference to earth. If it has a sensor on board that can sense the gravity vector that it felt while it was sitting on the pad, it would still know whether it was vertical or not. However, once the motor ignites and the rocket lifts off, it is under a g force that exceeds 1 g (earth's gravity). Any sensor that is monitoring gravity will lose any sense of the earth's pull and its associated reference – the sensor cannot distinguish the difference between gravity and acceleration.

Therefore the tilt in the earth's frame of reference cannot be measured any longer. And, the converse is true – when the rocket reaches motor burnout the rocket is weightless for an instant and begins to experience negative g, the same phenomenon occurs but in the reverse. Unfortunately, even my thermal-based Memsic accelerometer was affected in this way.

Alternatives

So now what? What other sensors could I try that would not be affected by gravity?



Transolve Flux Capacitor

Magnetometers. I knew a bit about magnetometers and had seen some ads for the Transolve Flux Capacitor that relied on a magnetometer to sense apogee. My physicist friend was very familiar with magnetometers and after realizing the issue with accelerometers, had come to the conclusion that magnetometers might be an ideal solution. The magnetic flux lines that extend from the earth are very stable and fairly easy to read. The problem with them is that they are very weak in magnitude and subject to interference from any nearby metallic influences – such as a long metal launch rail!

Anything that can influence the magnetic reading can usually be calibrated out if its influence is constant – e.g., if you house the sensor in an avionics bay with steel connecting rods, the magnetic distortion caused by the rods can be deducted from the reading effectively canceling out the rods' influence - they always stay in the same orientation relative to the magnetometer. However, the launch rail is there while the rocket is sitting on the pad, but once it is launched its influence goes away. You could after all, calibrate the sensor away from the pad just prior to launch, load the rocket on the rail, then when it lifted off the calibration you did would take effect. However, this seemed troublesome and I discounted them.

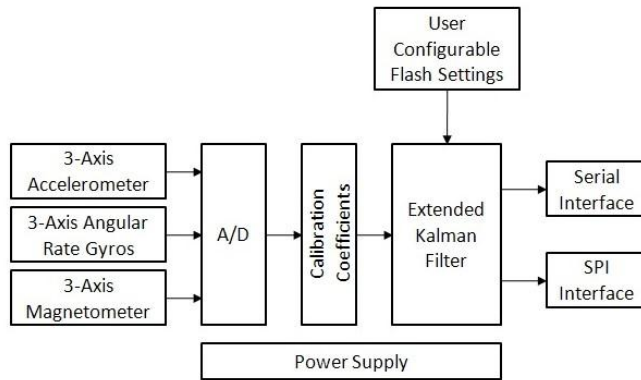
GPS. GPS sensors also came to mind, but the resolution afforded does not really match up to the requirements for a rocket moving so quickly in such a small relative flight path footprint. I discounted use of a GPS sensor fairly quickly.

Thermopiles. I came across a reference to “thermopiles” being used to sense the horizon by contrasting the sky's IR signature with the earth's. I eventually was drawn away from using the thermopiles and never really explored them for a couple of reasons. For one, it did not seem that the thermopile sensors would give me the accuracy or preciseness of angle that I wanted. Additionally, it was recognized that the thermopile sensors did not work very well under a cloudy sky.

Gyros. One of the references I saw on the internet while searching about thermopiles mentioned something called an IMU to replace the thermopile sensor. More research.

An IMU is an inertial measurement unit. It usually uses combinations of sensors to determine changes in a vehicles orientation. The sensors it uses are usually some combination of accelerometers, gyroscopes or magnetometers. Using an IMU, you can transform the vehicles orientation to the earth's orientation and you will have an AHRS or attitude/heading reference system. Maybe I could adapt an IMU to develop my tiltmeter, after all, I noticed that the output of the some of the IMU sensor systems I was researching output the rate of rotation and what are called Euler angles or vector references. These

angles were developed chiefly from the internal gyros. After discovering these all-in-one devices, I started thinking it might be a lot easier to just bite the bullet and order one that output angles already but the devices were priced on the order of several hundreds to thousands of dollars.



Continuing to display my ignorance, I thought I should be able to come up with what I needed relatively easily using a couple of gyros. I did not need all the sophisticated sensor systems offered. All I needed were angles. Gyros were designed to sense rotation and be integrated into angles, right? I should be able to grab a gyro, do a bit of coding, and develop my own tiltmeter! Seemed easy enough and the new MEMS-type gyros were not terribly expensive.

Unfortunately, as I later learned, standalone gyros inherently have drift and are not useable as such. And, just how does one go about turning the output of a rate-of-rotation sensor into an angle. I pushed on.

I began by reading the specification sheets that were on the web store at SparkFun. SparkFun is a wonderful site for experimenters and those dabbling in electronics, and microcontrollers and sensors in particular. I picked a couple of likely candidates and began working on the code to use the gyros with a microcontroller and to integrate the output of the gyros over time to develop an output in angles.

I learned about gyros and their characteristics which influence their output, such as bias, reference voltage, saturation, resolution, sensitivity and sample rate. Once all of those aspects are dealt with in the programming, I then learned to convert the sampled data into angles by integrating many samples over time.

After much trial and error (lots of both!), I eventually I was able to harness it all and had a gyro-based tiltmeter. Unfortunately, the drift issue became readily apparent. The angles would wander in spite of the fact that my system was at a stable point.

William Premerlani

While doing research on IMU's, I kept seeing references to the UAV Development Board (UDB) and a guy named William Premerlani. Apparently, Bill was challenged by his son one day a few years ago to build

an autopilot after they saw a UAV unmanned autonomous vehicle) make a Transatlantic crossing. Though not expecting to cross the Atlantic, Bill took on the challenge and began to develop his own autopilot board and software.

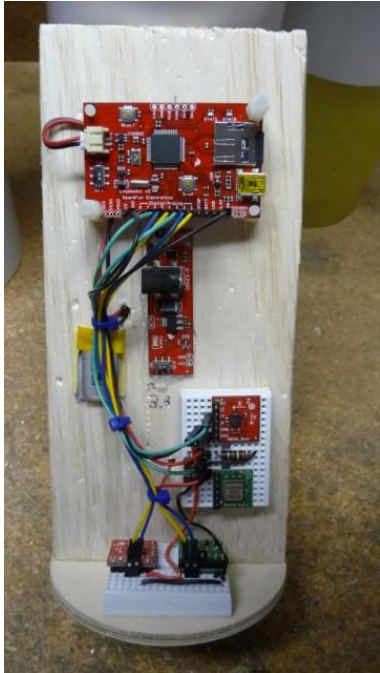


William Premerlani's UAV Development Board (UDB)

I contacted Bill one day and told him of my project – he showed immediate interest and offered to help. Now that I fumbled my way and learned enough to actually do rotational rate integration and derive angles, I spent time discussing the more rocket-specific details of my project with Bill. It appeared that indeed, maybe with Bill's help I could adapt the UDB and its built-in IMU for my tiltmeter. I ordered one of his UDB's and ordered a couple of books on the C programming language – the coding system that Bill used on the UDB.

Initial Gyro Flight Testing

While waiting for my UDB to show up, I wanted to determine if the gyro sensors I was using were capable of performing while under the stress of the g load experienced during a high-powered rocket flight.



Gyro Test Fixture w/Logomatic Data Logger

I designed a little fixture that I could fly in a “mule” rocket that would simultaneously record the output of the two different gyros I had selected during a test flight. I used another device from SparkFun, a Logomatic SD card-based data logger. With the Logomatic, I could connect both of the gyros simultaneously through two of its eight ADC channels, then download the data after the flight and compare the outputs.

I flew the gyros a couple of times (March 2010). The results indicated there was not a substantial difference between the performances of the two designs.

However, something I had not considered became apparent as I studied the data. Though I knew that the rocket flight path had a continuous arc from my observation, the angles I had plotted grew as expected, but then diminished further along the flight up to apogee. Hmmm, that was odd. It was also not consistent with my observation that the rocket continued to arc over at a fairly steady state all the way up. What did that mean? Then it dawned on me. The rocket had rolled during its ascent to apogee and this threw the angle calculation off.

So, after some thought I decided I would need to add a third gyro to monitoring the Z-axis and then integrate all three to allow for the roll aspect.

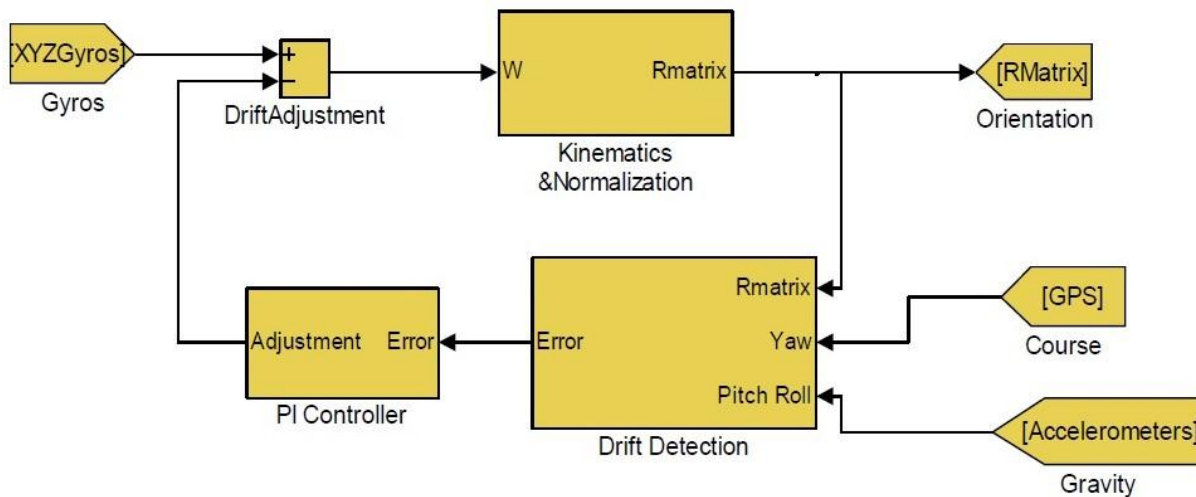
I was anxious to throw this line of modified integration calculations thinking over to Bill and see what he thought. I wanted to see how tough this new direction I found myself headed in was going to be. I sent Bill an email, describing my view and asked if it was indeed a relatively straightforward linear problem as I perceived, or if it were subtler than that. Disappointingly to me because of the implications to my project, Bill answered that it was indeed more subtle.

Fortunately though, Bill also said that he had something he thought would help. Thinking back now, he was probably grinning a bit as he wrote that, having been through the learning curve I was on many years ago.

Direction Cosine Matrix - DCM

The Kalman filter is probably the most common approach to use for coordinate transformations. Another approach to the problem is the rotation matrix, or direction cosine matrix (DCM). The heart of the UDB's attitude orientation system is the DCM, which Bill developed and adapted a few years ago with help from Paul Bizard (interestingly, a similar approach was being independently developed by Robert Mahoney about the same time – see reference section).

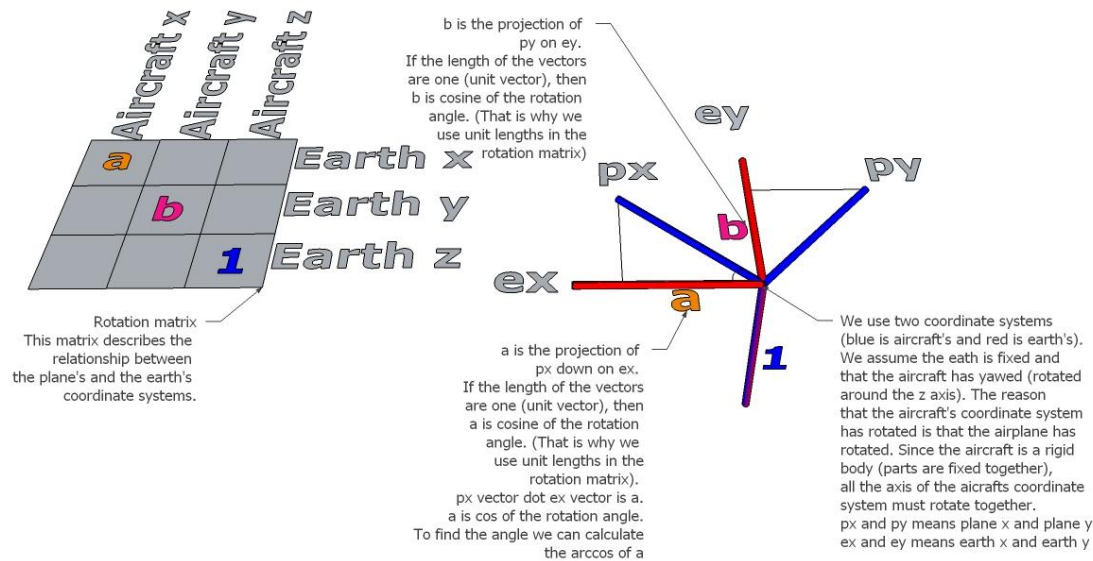
The DCM is another mathematical construct to keep track of attitude and orientation, but rather than being a general approach like the Kalman filter, it is designed specifically for tracking rotation. Because it is specific to the task, it is very effective and efficient. Therefore, the computations are fewer and easier for the microcontroller to process. There is extensive background material included in the reference section at the end of the article on Kalman filters and the DCM. Another advantage is that the DCM is not subject to some “singularity” solutions that cause some implementations of the Kalman filter to be somewhat troublesome.



DCM Block Diagram

In his reply to me, Bill offered that obtaining the X and Y angles I wanted from the existing calculations being done by the DCM-based UDB firmware was straightforward and that he could help me easily

derive them. I worked with Bill and revised the UDB firmware accordingly. Soon I had a working gyro-based X & Y axes tilt meter!! After nine months of work, I had a solution (April 2010).



After working with me to adapt the firmware, Bill offered another insight. He made me realize that by monitoring X and Y for plus and minus tilt (i.e., right-left, front-back), I would actually be creating a “boxy” envelope rather than a true uniform tilt meter. What I needed was something that monitored the *tilt orientation off the Z-axis*, rather than monitoring X and Y. If I could do that, I would indeed have a true tilt meter with a cone-shaped Z-axis envelope instead of the boxy-shaped X and Y envelope. A few more emails and programming and I had a working, gyro-based, true cone-shaped Z-axis tilt monitor!

So now I had a device that would allow me to interface with other ignition-initiating rocket electronics such as flight computers, altimeters or timers. I could effectively put my tiltmeter between the device calling for ignition and the igniter. If the rocket's angle has gotten too far past vertical when ignition was being called, I could preclude ignition and have the rocket just return for another attempt.

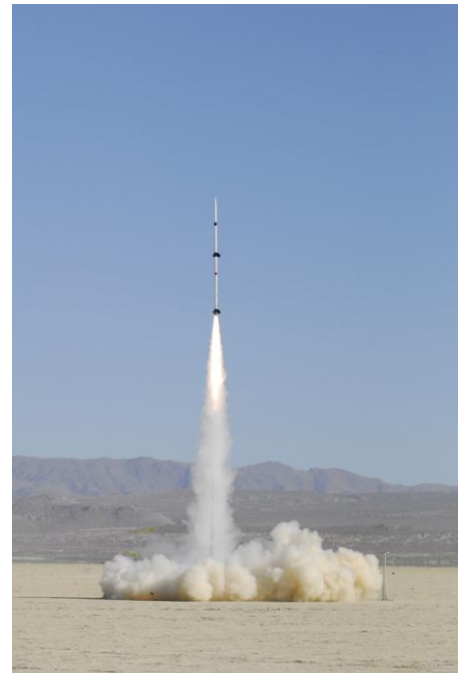
Goal achieved - plus!

This new “tiltometer” (til-tom'-e-ter) could indeed be used to satisfy my initial desire to maximize the coast period in an effort to maximize altitude. As a *tremendous* side-benefit, and one not initially on my radar at the start of my journey, is that in the event of an adverse attitude flight where the tilt angle is

significantly off-axis and poses a safety or rocket destruction threat, this “ignition control system” will inhibit ignition!

Flights

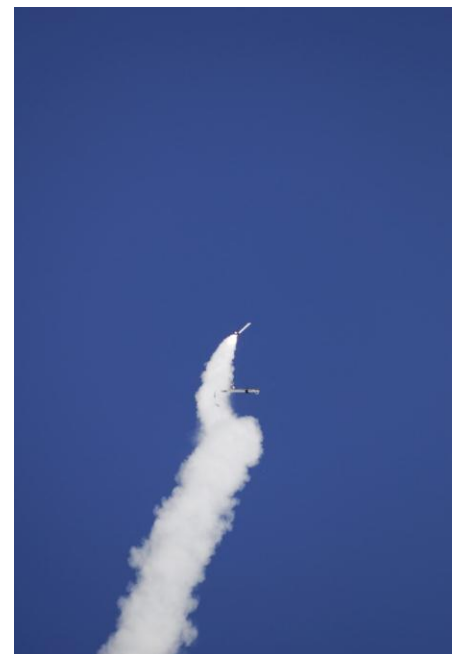
I flew the tiltometer at BALLS 19 in September of 2010. The rocket I flew was named Hermes 8, Mark III, and it housed the prototype tiltometer and received launch and ignition triggers from a Raven2 flight computer. The booster contained a Cesaroni N5800 C-Star and the sustainer a Cesaroni M2020 I-Max. My RockSim simulation predicted a 50K'+ flight.



Liftoff was great, but shortly afterwards the rocket started arcing over, finally breaking up at about 2,000'.

Though the flight was an obvious disappointment, the tiltometer wound up saving my sustainer motor to fly another day. In addition to the expense of the motor reload kit (~\$370 retail) that was saved, the end result was much safer than it could have been if the motor had been tied to a straight timer for ignition. With the adverse event happening so close to the ground and the wild nature of the airframe sections at that point, motor ignition could have been a serious threat to the spectators.

On a better note, later in the year, I altered the motor configuration to allow for the lesser waiver height allowed at Plaster City, CA for the annual Plaster Blaster event held in



November of 2010. With a ceiling of 25K', I used the saved BALLS M2020 sustainer motor in the booster and put an L995 in the sustainer, with an expectation of exceeding 20K' if all went according to plan.



The boost went as expected. Once again, a Raven2 was monitoring various aspects of the flight. When it saw the other the pre-programmed conditions go "true", as it decelerated through 400 fps (another trigger condition), it sent an ignition signal to the tiltometer. Since the tilt of 9 degrees was well within the 20 degree envelope for which I had it configured, when it received the ignition signal from the Raven2, the tiltometer in turn sent an ignition trigger signal to the analog input channel on a G-Wiz HCX. The HCX then fired one of its pyro channels and the sustainer's L995 ignited. Hermes 8, Mark II, reached an altitude of 22,299' with a coast time between booster burn out and sustainer ignition of about 11 seconds.

So, two different experiences, but in both cases the tiltometer worked to perfection - very satisfying results and a nice payoff for all the months of effort.

Cost

Quite a bit of expense was undertaken over the several months of development to research all of the alternatives described above. Building a one-off tiltometer as described and incorporating a modified UDB and the components necessary to sense and react to a call for ignition might run a few hundred

dollars. Further development however could allow all the components to be integrated onto one printed circuit board which would allow significant cost reduction.

References

“A Guide To Using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications”, Starlino, <http://starlino.com>

J. Geen, D. Krakauer, “New iMEMS® Angular-Rate-Sensing Gyroscope”, ADI Micromachined Products Division, Analog Devices www.analog.com

W. Premerlani, P. Bizard, “Direction Cosine Matrix IMU: Theory”, May, 2009

R. Mahoney, T. Hamel, J. Pflimlin “Nonlinear Complementary Filters on the Special Orthogonal Group”, IEEE Transactions On Automatic Control, Vol. 53, No. 5, June 2008

ROCKETILOMETER™ - www.RocketElectronics.com