Rollercoasterometer

From Kwan Hypertext Library

At long last, the completion of my college Senior Project draws near. As I well remember, there was a goal and a mission for that project. The goal was to fulfill a requirement to get my Bachelor of Science degree. The mission, however, was to make a SixDOF IMU which could measure Space Mountain. The goal was achieved, but the project never came close to completing the mission.

That mission is about to be accomplished. The next version of *Accelerate*, the *Rollercoasterometer* has been born.

Contents

[hide]

Mission

The mission is to measure as accurately as possible the track for Space Mountain. Since we will get it for free, we also measure the velocity and accelration profile for the system.

Space Mountain is a difficult thing to measure since it is dark, it is inside, and GPS consequently doesn't work.

The plan then is to use an inertial measurement unit (IMU). Such a device uses suspended proof masses to measure the forces on a vehicle. The IMU must be able to measure accelerations and rotation rates sufficiently accurately to reconstruct the track, and have sufficient dynamic range so as to not saturate.

Resources

Science has marched on from when I was in college. Micro-electro-mechanical systems (MEMS) were just coming into commercial production back in school, and for something like $100 you could get a single-axis accelerometer chip. Now for $19.95 plus shipping and handling you can get A triple-axis accelerometer (http://www.sparkfun.com/commerce/product_info.php?products_id=252) , or under $10 if you are willing to buy in bulk without a breakout board.

More importantly, MEMS rotation rate sensors have become available. These use a vibrating structure to directly measure rotation, rather than the old-style use a gyroscope to measure precession. They cost about three times as much per axis, but it is still feasible.

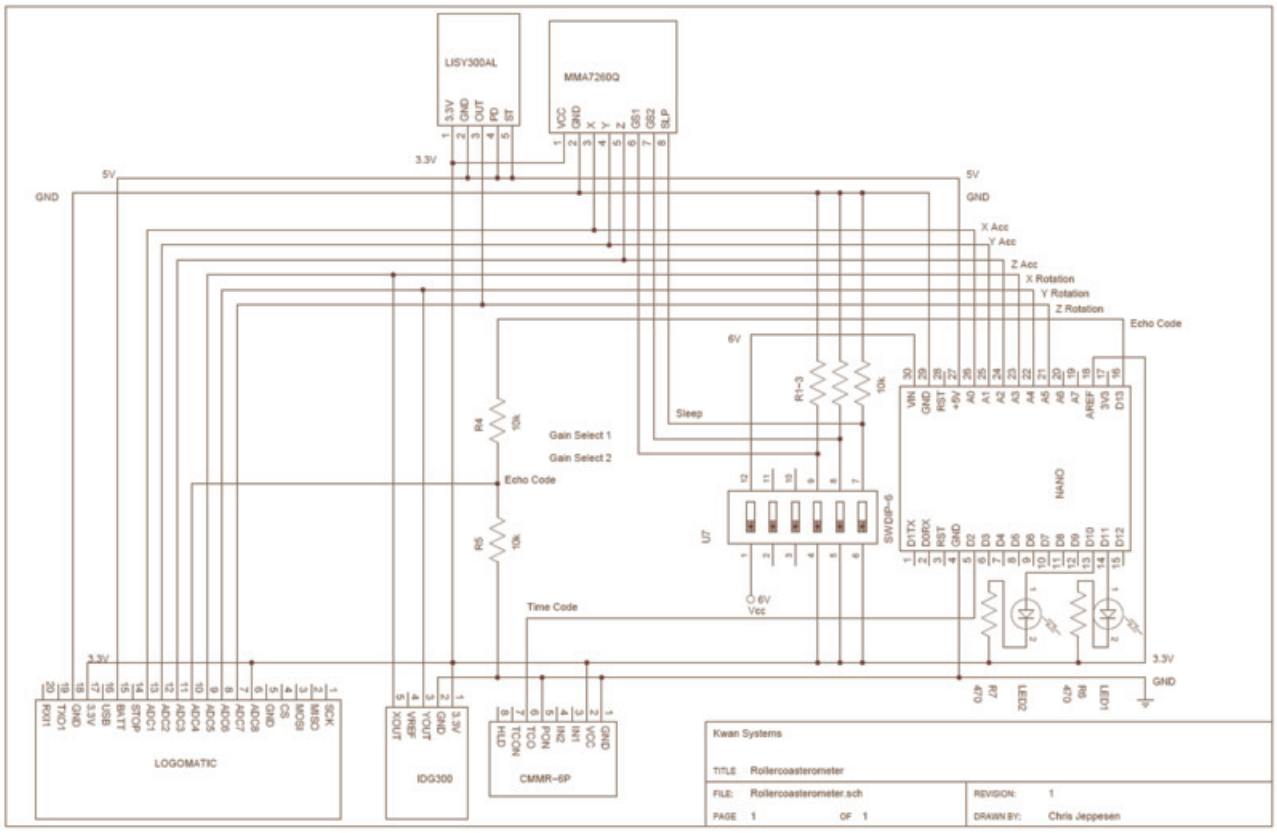
Parts list

- $49.95 - 1 Arduino (http://store.gravitech.us/arduino-nano1.html) Nano (http://gravitech.us/Arduino/UserMan_Arduino_Nano.pdf) microcontroller
- $59.95 - 1 IDG300 (http://www.sparkfun.com/commerce/product_info.php?products_id=698) dual-axis MEMS gyroscope
- $59.95 - 1 Logomatic (http://www.sparkfun.com/commerce/product_info.php?products_id=8627) MicroSD data logger
- $19.95 - 1 MMA7260Q (http://www.sparkfun.com/commerce/product_info.php?products_id=252) tri-axis MEMS accelerometer
- $29.95 - 1 LISY300AL (http://www.sparkfun.com/commerce/product_info.php?products_id=8955) single-axis MEMS gyroscope
- $10.70 - 1 CMMR-6P-60 (http://www.c-maxgroup.com/downloads/getFile.php?id=542&PHPSESSID=976d4279044b6030d34ea14e37a3e98b) WWVB receiver module with ferrite rod antenna
- 1 SanDisk (http://www.sandisk.com/OEM/ProductCatalog(1300)-microSD_Memory_Module.aspx) 128MB MicroSD card
- 1 breadboard
- several resistors
- $9.95 - lots of jumper wires
- 1 USB mini-B cord
- ~~$6.99 - 1 Energizer (http://www.batteries.com/productprofile.asp?appid=201940) six-volt 26Ah lantern battery~~
- 4 Duracell AA 1.5V 2.5Ah cells

I am not costing the things that I already have, sales tax, shipping/handling, and things I bought along with these but didn't use. The total cost then is $247.39. I also swiped the MicroSD card from my GPS and replaced it with a 4GB card.

No one part costs that much, but they add up. The Arduino will be useful in lots of other things, such as Precision.

Theory of Operation



I don't know why they call it that, but such is life. This is basically a text schematic of the system.

The system consists of two microcontrollers, three sensor boards, one time reciever board, and a handful of passive components and LEDs.

Data Logging

The heart of the system is the Logomatic. It is a LPC2148 (http://www.keil.com/dd/chip/3880.htm) microcontroller with a MicroSD card reader, USB Mass Storage interface, and eight A/D inputs, along with some other neat stuff like a 3.3V regulator to power the sensors. When powered by its battery input, it is in 'record' mode, and records the analog inputs in a binary format described below.

The Logomatic analog inputs are connected to:

- Three outputs of the triple axis accelerometer
- Time code signal (described below)
- Three outputs of the rate sensors
- 3.3V regulator output

The Logomatic can run at any voltage from 3.6V to 7.5V, but to simplify the wiring, we run it off of the 5V regulator on the Nano. We use the 3.3V regulator output to run the sensors and time code receiver, as well as provide reference voltages to both itself and the Nano.

As noted above, the Logomatic has its own microcontroller. It has plenty of power to run this entire experiment (12MHz, 512kiB flash, 40kiB sram) and in fact is a bit more powerful than an ATmega168, but it is not an Arduino, and I don't know how to program it. I have all the hardware I need, and the software is freely available, but I haven't learned how to use it. So, I just use the firmware which came with it, which just knows how to log whatever arrives at the ADC inputs.

The Logomatic is configured by way of a text file on its card. It is currently set up to record all eight analog channels. The format is two bytes for each measurement, big endian, padded with six zero bits in the MSB to fill up the two bytes, then two bytes of separator "$$". Since each measurement is 10 bits, the MSB of a measurement can never be more than 0x03, so this separator is out of band.

The data rate is configurable, but is set to 100Hz. Together with the (8x2)+2=18 bytes generated per sample, this gives a data rate of 18x100=1.800kB/s (not kiB). This is 108kB/min and 6.48MB/hr. The card has a usable capacity of 1975255040B (1.83GiB) so it has space theoretically for 1097364s, 18289.4min, 304.82hr, 12.7day.

The writer can write 512B in 42.5ms, or 12047B/s. This means that the sampling rate could theoretically be raised to 669Hz before it was guaranteed to give up. Probably 600Hz is safe.

Time Code

In order to log which recording goes with which Mountain, we need a whole-day automatic logger. The GPSmap provides that. It will track us around the Mountains and time stamp each point. So, if we have a real-time clock signal recorded on the data logger as well, we can cross-correlate the GPSmap data and the Rollercoasterometer data.

The Logomatic has a crystal to run its clock, but no wall-clock capabilities. In order to provide that, we use the Arduino Nano. This microcontroller has 16kiB flash, 1kiB bytes SRAM, and runs at 16MHz. It has 14 digital I/O pins (some specialized) and eight analog pins. It also has a USB interface, but this one is for programming and serial I/O with the host machine.

The primary feature of the Nano is the Arduino IDE which comes with it. This allows for easy programming in C. All of its parts are mostly orthogonal, but some things like the UART are tied to its USB serial interface.

The Nano has the hardware it needs to run an I^2C bus, and I have a MicroSD reader which sits on that bus, but I haven't been able to successfully interface them yet. That, combined with the fact that the Nano doesn't really have enough memory to do the job, means that the Nano and Logomatic complement each other nicely.

So, the Nano pays its way on the rollercoasterometer by reading and interpreting the time code. This code is generated at WWVB, and transmitted by radio to the CMMR-60-P6 time code receiver module. This module is just a radio, so it just outputs the raw time code. The Nano then interprets the time code and keeps enough bookkeeping information such that combined with its own real time clock, it can echo the time code. Why does it echo? So that when there is no WWVB reception, the Nano can keep generating the code itself in the same format as WWVB does.

The time code receiver requires about 2.5kiB, and the time code generator requires about 2.2kiB. So, hopefully the two together will take somewhat less than 5kiB, and will fit within the resources available on the Nano.

WWVB Receiver

The system uses a CMMR-6P time code receiver with attached ferrite rod antenna. Originally, this module was attached directly to the main breadboard. However, that put its antenna in close proximity to the Nano and especially its external LEDs. These parts interfered with the radio and caused it to glitch.

In order to be resistant to these glitches, the software on the Nano only watches for up-transitions at appropriate times (200,500,800±25ms) after the falling edge. If the rising edge is not in one of these windows, the Nano just watches for another fall and rise, and takes the time from the first fall to the last rise. If the time grows to more than 825ms, the Nano gives up on this bit, and resets to watch for the next one.

Also, the receiver module is now on the end of a 1m ribbon cable.

Between these two improvements, I plan to get radio reception all the way out in California. In the event I don't, the Nano will blink a time code starting at 99/001T00:00:00 from when it starts up to when it first gets a complete time frame.

USB serial output

All analog channels are piped to the analog inputs of the Nano as well as the Logomatic. This is done so that when the oscilloscope sketch is running on the Nano, it can see and relay the voltage readings to the host computer.

Power

The Nano has three power inputs:

- 5V from an external regulated supply
- 6-20V from an external unregulated supply, internally regulated down to 5V

- USB power

Now when the Nano is not running on the 5V regulated supply, it uses that pin as an output, providing 5V regulated power (if supply is 6V or more) or less if the input voltage is less. It supplies about 4.8V when running on USB power, and 5V on the nose when running on the 6.4V lantern battery. Either of these voltages is satisfactory input to the Logomatic, so to allow all the power sources to be hard wired.

The Nano also has a 3.3V regulator built into its USB interface, but this is only powered when the Nano is running on USB. So, we can't count on it.

The battery is wired through a switch to the VIN of the Nano. The 5V pin is connected to the 5V bus, and the 3.3V pin is unused. The 5V bus is in turn wired to the BATT input on the Logomatic. When the Nano is running on either USB or battery, power flows from the Nano along the 5V bus to the Logomatic. In turn, when the Logomatic is powered by USB, it puts 4.1V onto the 5V bus, which is just barely enough power to activate the Nano. Interestingly, it seems to power the Nano 3.3V regulator also, which means it is powering the Nano USB interface.

The whole system runs at about 120mA on average when running on battery. The system was originally planned to be run on a 6V lantern battery with a rated capacity of 26Ah, which means that the system could run on battery power for over 200 hours. However, the lantern battery does not fit in the enclosure. So I am trading capacity for convenience, and powering the system with a battery of four AA cells. The system has been tested for 22h20m of continuous running, without the radio.

It appears therefore that the battery has adequate power to remain on during the entire time I am at the Mountains. The battery has a capacity of at least (120mA)(22.3hr)=2.676Ah at 6+volts, for a total energy capacity of 16Whr, or 57kJ (or 14 food calories).

Inertial Sensors

There's not really much to say about the sensors. They are literally black boxes. As discussed above, are all MEMS devices.

All the sensor channels generate an analog output voltage which varies from 0V to near their supply voltage. Zero reading on all sensor channels is represented by an output near half of the supply voltage.

It turns out that the dual axis rate sensor costs a nickel more than two of the single axis sensors I got. But, the single axis sensor's axis is perpendicular to the board, and the dual axis sensor's axes are parallel to the board. This, combined with one fewer board and five fewer pins, is well worth that nickel. It would have been difficult to mount all three single axis sensors mutually perpendicular, since one would be along the terminal rails of the breadboard.

The accelerometer has a gain control. It is adjustable to ±1.5g, ±2.0g, ±4.0g, or ±6.0g, sensitivity. Most civilians start getting scared rather than entertained with G forces of greater than about 3.0, so it is reasonable to assume that the Mountains will stay within that limit, and ±4g is the correct gain setting. However, the sensor seems quite a bit noisier in the per-bit sense at lower gain, so there may not be any *gain* to running at a lower gain. I imagine I will record Space Mountain at least four times, one on each gain setting.

As it turns out, since the three-axis accelerometer actually uses the same proof mass for all three axes, co-locating the sensors is easy. All accelerometers need to be co-located, but the rotation sensors don't need to be, since all points on a rigid body turn at the same rate. As it happens, all sensors are within about an inch of each other, but this is just for board-stuffing convenience.

Wiring and passive components

The system is built onto a single 64x2x5 four-rail breadboard. The board was put together a piece at a time, somewhat by trial and error, with much pulling and re-placing of the boards. Fortunately, none of the boards were damaged in this process. Before the final wiring, enough wires were put in to mark the places of all the boards. Then, all boards were removed and the wiring put into place. Finally, the boards were installed, largely covering the wiring.

Do not trust the photos for a detailed wiring plan. That's what the schematic is for.

There is a block of DIP switches on the board. Combined with a set of pull-down resistors, switches 4-6 are used to supply appropriate logic voltages to the control inputs of the accelerometer. The battery positive terminal is plugged in through switch 1, so it is switchable as needed.

The outputs of the digital logic pins on the Nano are high enough to saturate the ADCs of the Logomatic. When this happens, the signal spills over into other inputs, rendering them useless. For instance, the time code signal cross talks onto at least two of six sensor channels if piped directly in. Therefore, a resistor voltage divider is used to perform level-matching between the two circuits.

The Nano has four LEDs built in, but only one is programmable. So, to indicate time code lock on, two LEDs are needed. These, combined with series current-limiting resistors, are connected between the appropriate logic pins and ground. The programmable orange LED on the Nano is used to echo the time code visually.

User's Guide

Starting the Rollercoasterometer

To start the rollercoasterometer, either turn on switch 1 (which will power the system on battery) or turn off switch 1 and plug a USB cable into the Nano USB port. Due to the power distribution design on the Nano, probably nothing will get hurt if both power supplies are on, but it doesn't make sense to take chances.

Time code reception

Start the Rollercoasterometer with the time code receiver plugged in. If reception is good, LED2 will start blinking received time code immediately, LED1 will come on within 1 minute of startup, and the orange LED on the Nano will start blinking echo code within 2 minutes. If the Nano loses power or is reset, this will need to be repeated. After the orange light is blinking, the time code receiver and two red LEDs may be disconnected.

Calibration

1. Find a corner between a wall and the floor. The floor must be as level as possible, and the wall as close to vertical as possible. Tables are also acceptable, but check the levelness.
2. Set the rollercoasterometer flush with the corner, large flat side down. The Z axis arrow should be pointing down. This gives the Z axis -1g, and the other axes 0g.
3. Slowly rotate the system such that the medium flat side is down and the large flat side is touching the wall. The X axis arrow should be pointing down. Due to the hinges and latches, and the fact that the case is not square in this direction, it will not be possible to put the flat side flat on the floor. This is ok, as long as the edge of the box is flat. This gives the X axis 1g, and the other axes 0g.
4. Slowly rotate the system again such that the short flat side is down, with the large flat side touching the wall again. The Y axis should be pointing down. This gives the Y axis its signal.
5. Swing the box back to level, Z down, large flat side down. Line up an edge against the wall. Bring the box away from the wall, spin the box clockwise once, then push the box back up against the wall. This calibrates the yaw axis.
6. Pick up the box and rotate it around its long axis once (pitch up), then set it down again, flat and flush. This calibrates the pitch axis.
7. Pick up the box again and rotate it around its short axis once to the right. Then set it down again flat and flush. This calibrates the roll axis.

Riding

Try and record the ride with some other sensor with a good clock, like a video camera or GPS. Try and make sure you get a good couple of seconds of complete motionlessness before taking off. Orientation doesn't matter a lot, but try to observe the this side up and this edge forward markings.

Accelerometer sensitivity

The accelerometer may be set to any of four sensitivities:

Sw4 (GS1)	Sw5 (GS2)	Sw6 (Sleep)	Sensitivity
x	x	off	Asleep (0)
off	off	ON	±1.5g
ON	off	ON	±2.0g
off	ON	ON	±4.0g
ON	ON	ON	±6.0g

You should recalibrate the acceleration after each sensitivity change.

Stopping the Rollercoasterometer

To stop the rollercoasterometer, push the STOP button on the Logomatic, then either turn off Switch 1 or unplug the USB power. If you want to restart logging without cycling the power, push reset on the Logomatic.