MEMS Sensor Characterization

6 DOF
David Bonner
Gilbert Gede
Jihwi Kim

Introduction

Knowledge of the motion and position of a dynamic system is important for analysis and control tasks. Many dynamic systems can't have these states measured externally. Thus, onboard sensing is required. An inertial measurement unit (IMU) is commonly used in these situations. Onboard sensors are used to measure accelerations, angular rates, magnetic fields, GPS signals, etc. from which a state estimate can be made. Commercial systems can be quite expensive, easily costing thousands of dollars for relatively low-grade units. A goal for the Sports Biomechanics Lab is to use an IMU to measure roll angle for a bicycle. A different solution is currently in use, but progress was made on developing a cheaper, open-source IMU which could be used in the future by our group and others.

An open-source design is important for a number of reasons, but the main reasons are improved development and accessibility. Allowing many people to examine the source code behind a project can result in better, faster, cleaner code. Accessibility is also important; spending thousands of dollars on an IMU is not feasible for many people doing experiments with dynamics. We feel it would be unacceptable that the cost of such a device be a barrier for allowing others to learn and research. By allowing the filter's code to be open source and based around cheap hardware (below $100 dollars), it could allow many more people to acquire data about their dynamic systems.

The Extended Kalman Filter seems to be the standard method used to fuse sensor data, and a working filter has been developed. Input parameters to the filter include sensor characteristics such as noise and bias drift, on top of calibrated sensor readings. Testing of a complete IMU would be impractical without a multi-axis rate table and a significant amount of time "tuning" filter parameters, so it was decided to limit this project to characterizing the sensor parameters required by the filter.

The Kalman Filter in its linear form (when using accurate parameters) is in fact the optimal estimator of a system with white gaussian noise. Being based off of both system dynamics and statistics, parameters from each domain are required. This is the reason for collecting data about the bias drift and sensor noise. More can be read about this at the website for the filter's code: https://github.com/gilbertgede/KalmanFilterIMU .

To accomplish this, the accelerometer and the angular rate sensor in a cheap commercial sensor stick were characterized with both static and dynamic tests. Dynamic testing was accomplished using computer control of laboratory equipment, implementing control of a shaker to test the accelerometer, and a rate table to test the angular rate sensor. The Horsley lab will also benefit from our project, as no computer control of the rate table had been performed prior to our project. The Labview software developed by our group will aid in future characterization of gyroscopes fabricated in our lab, as it will help in characterizing the sensitivity and bandwidth of these sensors.

Methods

Static testing of the accelerometer and angular rate sensor were performed to determine the accelerometer's noise, bias offset, bias drift and sensitivity (as the accelerometer is subject to a gravitational acceleration under static loading). Static testing also determined the angular rate sensor's noise, bias offset, and bias drift. The sensitivity of the angular rate sensor needed to be determined using the rate table, so it was exposed to an angular velocity. The frequency response of the accelerometer was measured using the shaker.

One of the goals is for the sensor to be open-source, so for data acquisition Matlab and LabVIEW are not acceptable. Python was chosen as an open-source alternative, as there are a number of numerical and scientific libraries being developed for it. The microcontroller used was an Arduino open-source microcontroller based off an Atmel 8-bit processor. Both the pcb layout, code, and the libraries behind the code are freely accessible.

The sensors used were an Analog Devices ADXL345 accelerometer and an Invensense ITG-3200 angular rate sensor. These were connected to the microcontroller with an I2C bus, which uses two wires for data transmission, one clock and one signal, and can have a large number of devices on one bus. Sensor datasheets list the registers that need to set, queried, and read from.

Wireless communication was determined to be necessary, both for the end result of a wireless IMU and for ease of data acquisition from the angular rate sensors when placed on a rate table. Wireless communication was provided by XBee radios, which self-associate to provide a wireless serial bridge. (Figure 1)

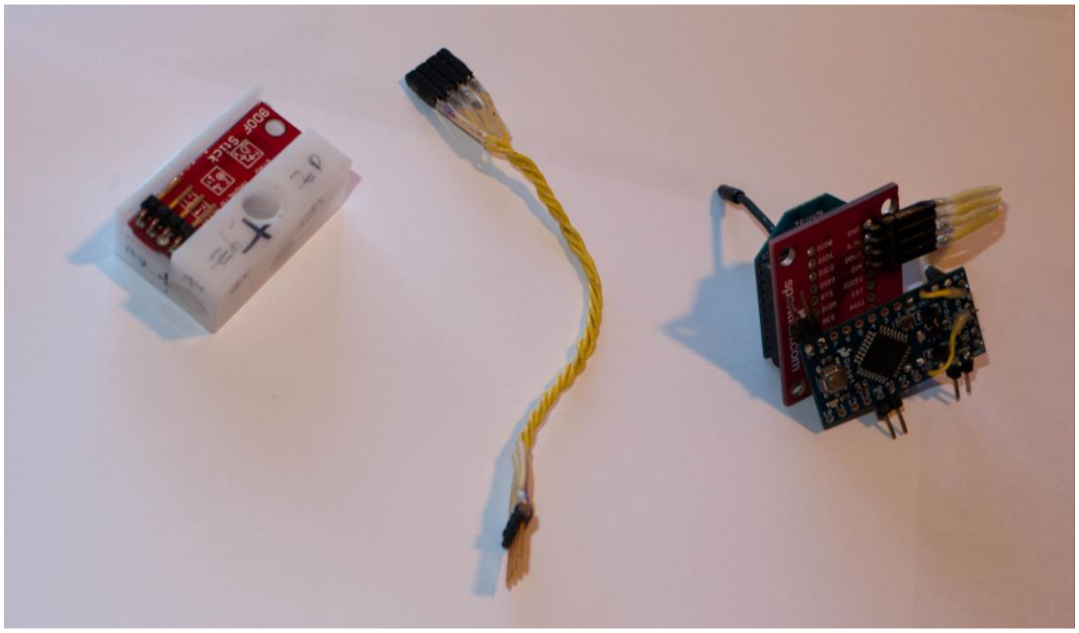


Figure 1, Sensor stick in the acetal block enclosure and the wireless transmitter

It should be noted that USB was used when possible to attempt to avoid potential data loss issues. Data was sent in binary form over the serial link to maximize throughput. The Arduino and computer use different methods of storing negative numbers (regarding two's complement); thus, a line of code to convert the values was necessary.

After the data was acquired from the accelerometer and angular rate sensor during both the static and dynamic tests, the data was processed by assuming the signal from each sensor as:

$$y = c * u + (b_0 + b_1 * t)$$

Where:

y is the sensor output
c is the gain
u is the input (acceleration, rate, etc.)
b_0 is a static signal offset, or bias
b_1 is the bias drift
t is time

To estimate these unknown values, a least squares fit is performed on the sensor output. The least squares fit estimates *x* in *Ax*=*b*. The structure we used is:

$$\begin{bmatrix} u_1 & 1 & t_1 \\ u_2 & 1 & t_2 \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{Bmatrix} c \\ b_0 \\ b_1 \end{Bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix}$$

The column of 1's is needed to find b_0. From here, the least square fit is performed and the necessary coefficients found. The b_0 value is thrown away, as it would change from session to session and the value will be estimated as part of the filter. In addition, once the least squares fit is complete, a standard deviation value can be calculated, which is another parameter required for the filter. The standard deviation is calculated with the fitted line being the signal average and the actual sensor reading as the signal with random noise. Noise and drift data was averaged between all runs for each channel independently, as some sensors claimed different noise values for different axes.

For the precise static and dynamic test to characterize the sensors, an acetal block was machined to allow precision orienting of the sensor block. (Figure 1) Because the sensor stick should be able to be placed in six different orientations, which are +*x*, -*x*, +*y*, -*y*, +*z*, and –z, for the tests, the block was machined to a shape of a right rectangular prism. Also, the block has three through holes so that it can be fixed to an instrument with a bolt in six different orientations. The sensor stick was glued to the enclosure block to secure it.

Static testing was accomplished by labeling the sensor to have three main orientations, *x*, *y* and *z*. It was then placed on a leveled surface with each axis both up and down, for a total of 6 tests. It was necessary to test with each axis both up and down in order for the least squares fit to be able to calculate both the gain and static bias accurately. The sensors were left in each orientation for 20 minutes; this length provided enough data with which we could estimate the bias drift.

Accelerometer testing

In order to characterize the ADXL345 digital accelerometer on the sensor stick, the static acceleration of gravity and the dynamic acceleration of motion were measured by the accelerometer. The microcontroller connected to the sensor stick acquired the signal from the sensor and sent the data to a computer over USB. A python program then read the binary data and wrote it to a file, where it could be subsequently analyzed.

For the dynamic test, the frequency response of the accelerometer was measured for each axis individually, with the test setup shown in Figure 2. A DUNEGAN/ENDEVCO 84501 M100 shaker was used with only one vertical axis of vibrating motion and a maximum frequency of 1000 Hz. An Agilent 33120A waveform generator inputs a time varying voltage to a Labworks Inc. pa-119 power amplifier which then amplifies the waveform generator's signal before sending it to the shaker. As shown in Figure 3, the sensor block was mounted on the shaker's mounting post with a bolt, and the sensor was connected to the microcontroller as in the static test. The accelerometer on the shaker was compared against a reference accelerometer which was already calibrated, so the actual acceleration the sensor was experiencing could be determined. The reference accelerometer used was an ISOTRON PE accelerometer, and was installed on the shaker mount along with the actual sensor. The reference sensor is operated with a 4 mA current source, a DYTRAN 4114B1. The current source acquires a voltage from the accelerometer, and sends this signal to an oscilloscope. In order to show and obtain the reference sensor's signal, a Tektronix 3014 oscilloscope was connected to the current source generator.

In order to automatically control the waveform generator and acquire the reference sensor data from the oscilloscope, Labview software was used. The waveform generator and the oscilloscope were connected to the computer with a general purpose interface bus (GPIB). The National Instruments PCMCIA-GPIB card for a laptop computer provided a direct connection to the GPIB instruments. Because the GPIB connector is stackable, it was possible for one computer (with only one PCMCIA-GPIB card) to control the waveform generator and the oscilloscope at the same time.

The primary purpose of our Accelerometer Testing VI is to make data acquisition from the ADXL345 digital accelerometer and from the reference sensor easier. Our VI is designed to control the shaker frequency and amplitude, as well as acquire data from the reference sensor on the oscilloscope. Our VI does not perform data acquisition from the ADXL345 digital accelerometer, as this is accomplished using python, as mentioned earlier. The Accelerometer Testing VI was developed from examples found in the instrument drivers, as we modified the examples and included them as sub-VI's in our program. The front panel and the block diagram are shown in Figure 4 – 7. The front panel consists of two parts; the left side is for the waveform generator and the right side is the graph panel for the oscilloscope. As seen on the block diagram in Figure 5, a for-loop is executed depending on the initial frequency, the final frequency, and the frequency interval inputs, and a sequential process is performed for each loop. This is shown below:

1) Our VI sends the signal to the waveform generator.
2) It then delays the oscilloscope reading for the reference accelerometer for 500 ms to get a steady state signal.
3) Next it reads the reference accelerometer signal from the oscilloscope
4) The VI then delays the next frequency change by 4.5 seconds to let the microcontroller read the ADXL345 digital accelerometer's signal for 5 seconds (only three periods of the input signal are sent to the oscilloscope and written to a file with the TDMS format, which Excel can read with the TDMS plug-in provided by National Instruments.)

After successful control of the two instruments was achieved, the test was performed at various frequencies starting from 15 Hz to 400 Hz for 5 seconds each, all started with one click on the LabVIEW VI. This frequency sweep test was completed once for each axis. A sine wave was selected for the input signal, and the amplitude of the input signal was kept low in order to prevent damage to the shaker.

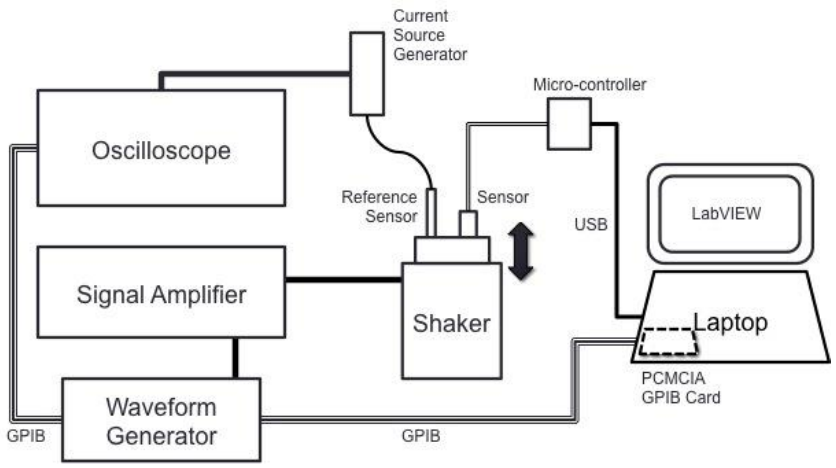


Figure 2, Accelerometer testing schematic

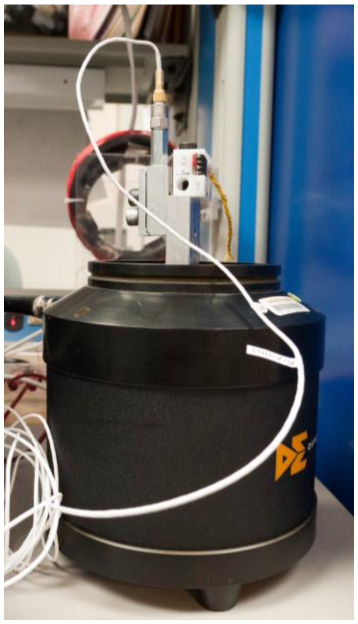
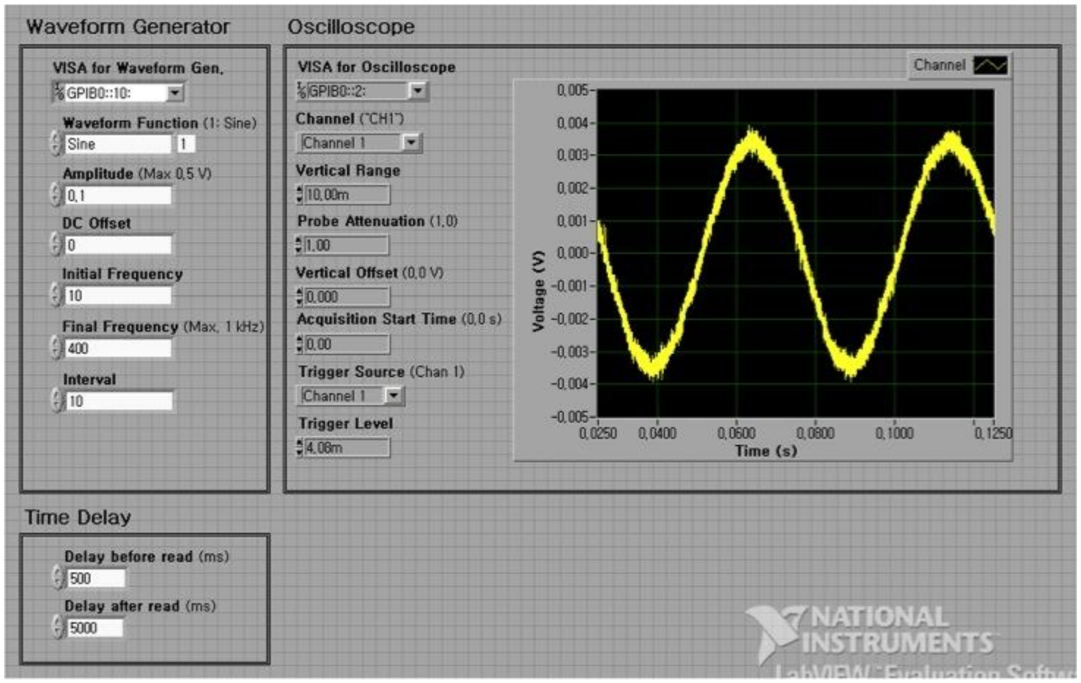


Figure 3, Sensor stick and the reference sensor on the shaker



Waveform Generator

VISA for Waveform Gen. GPIB0::10:

Waveform Function (1: Sine) Sine 1

Amplitude (Max 0.5 V) 0.1

DC Offset 0

Initial Frequency 10

Final Frequency (Max. 1 kHz) 400

Interval 10

Oscilloscope

VISA for Oscilloscope GPIB0::2:

Channel ("CH1") Channel 1

Vertical Range 10.00m

Probe Attenuation (1.0) 1.00

Vertical Offset (0.0 V) 0.000

Acquisition Start Time (0.0 s) 0.00

Trigger Source (Chan 1) Channel 1

Trigger Level 4.08m

Time Delay

Delay before read (ms) 500

Delay after read (ms) 5000

Figure 4, Front Panel of Accelerometer Testing VI

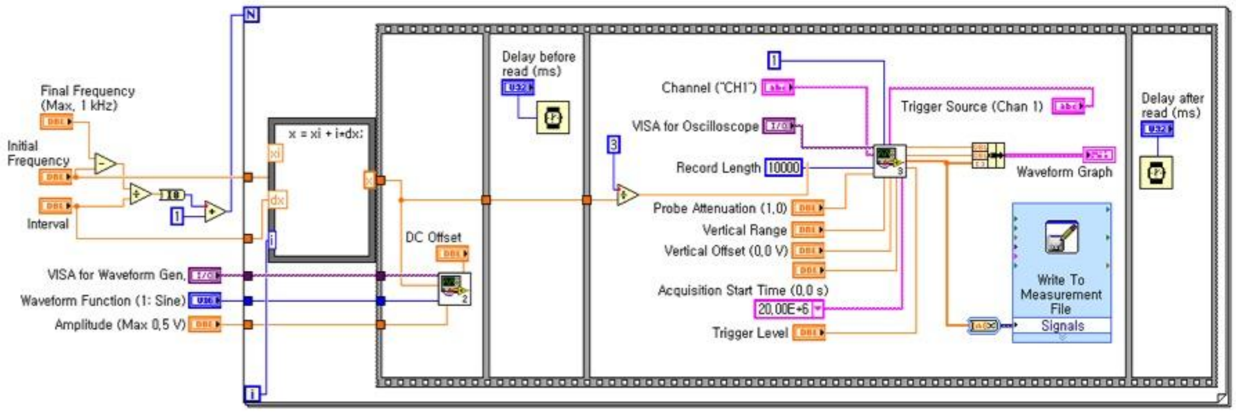


Figure 5, Main Block diagram for Accelerometer Testing VI

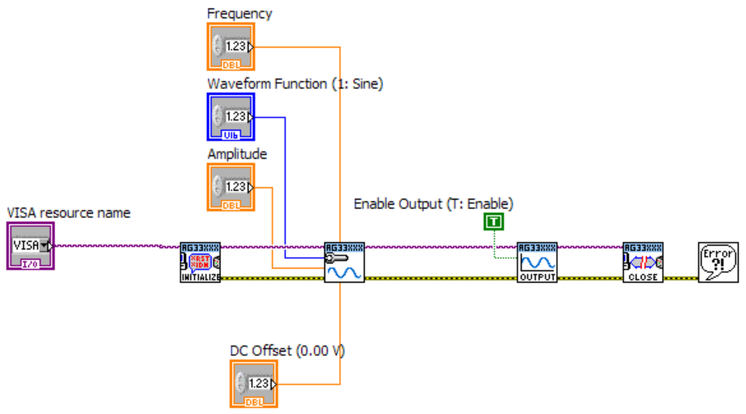


Figure 6, Sub-VI for the Waveform Generator Control

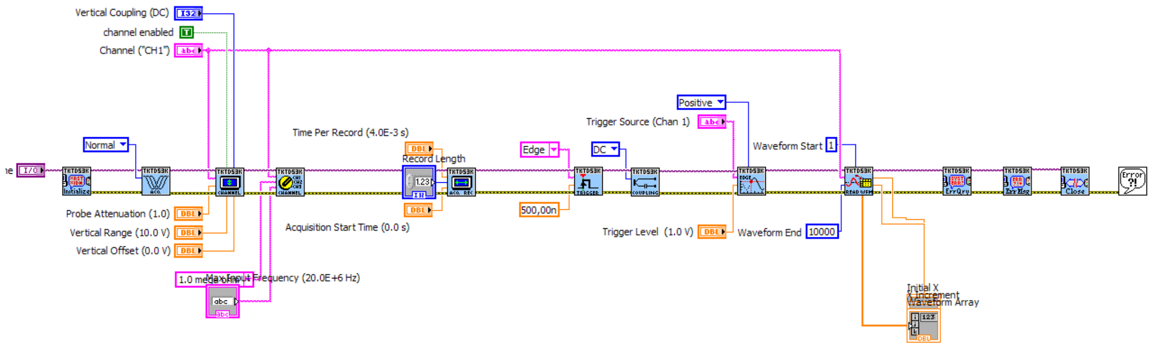


Figure 7, Sub-VI for the Oscilloscope Control

Angular Rate Sensor Testing

Both the static and dynamic performance characteristics of the angular rate sensor are important, just as for the accelerometer. To measure static characteristics such as offset bias, noise, and drift, the angular rate sensor was positioned in 6 different orientations, in exactly the same manner as for the accelerometer characterization. The main dynamic characteristic of interest is the sensor's sensitivity to angular velocity.

In order to characterize this sensitivity, an “Ideal Aerosmith 1291 BR single axis automatic positioning and rate table system,” was used. The angular rate table has a maximum angular velocity of 500 degrees/second. The speed of the rate table is controlled by an accompanying controller. This is accomplished with an encoder on the rate table, which enables the controller to know the position of the rate table, and thus adjust the appropriate parameters to control it. Figure 8 shows the rate table with its controller beneath it. In order to operate the rate table, ASCII commands must be issued from a computer, connected to the controller through an RS-232 Serial Port. In order to facilitate the communication process, Labview was used to communicate with the rate table, using VIs designed to send and receive ASCII commands through a serial port. This is shown in figure 9 below:

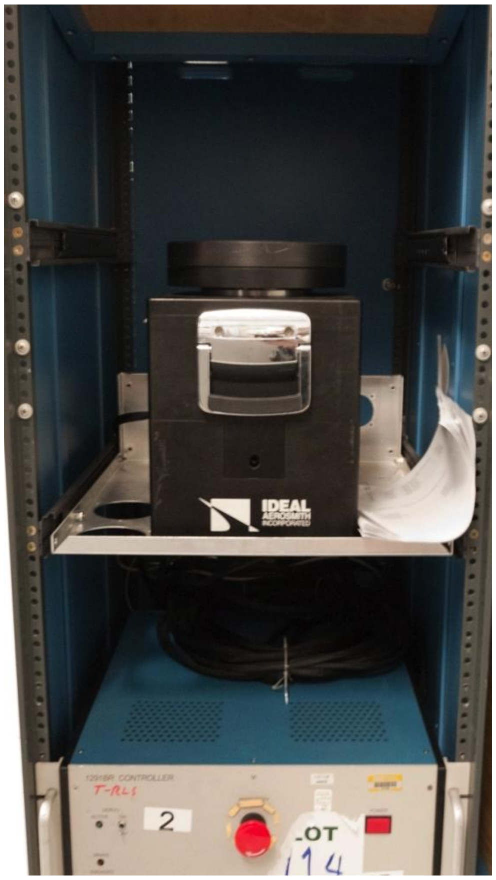


Figure 8, Rate Table Setup

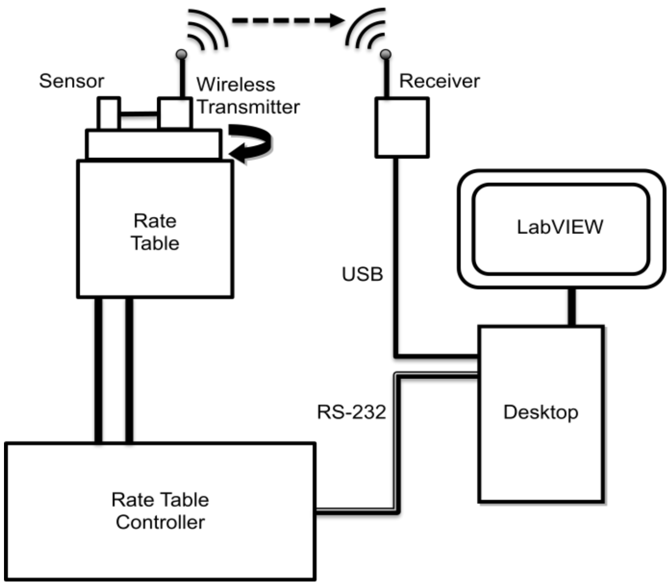
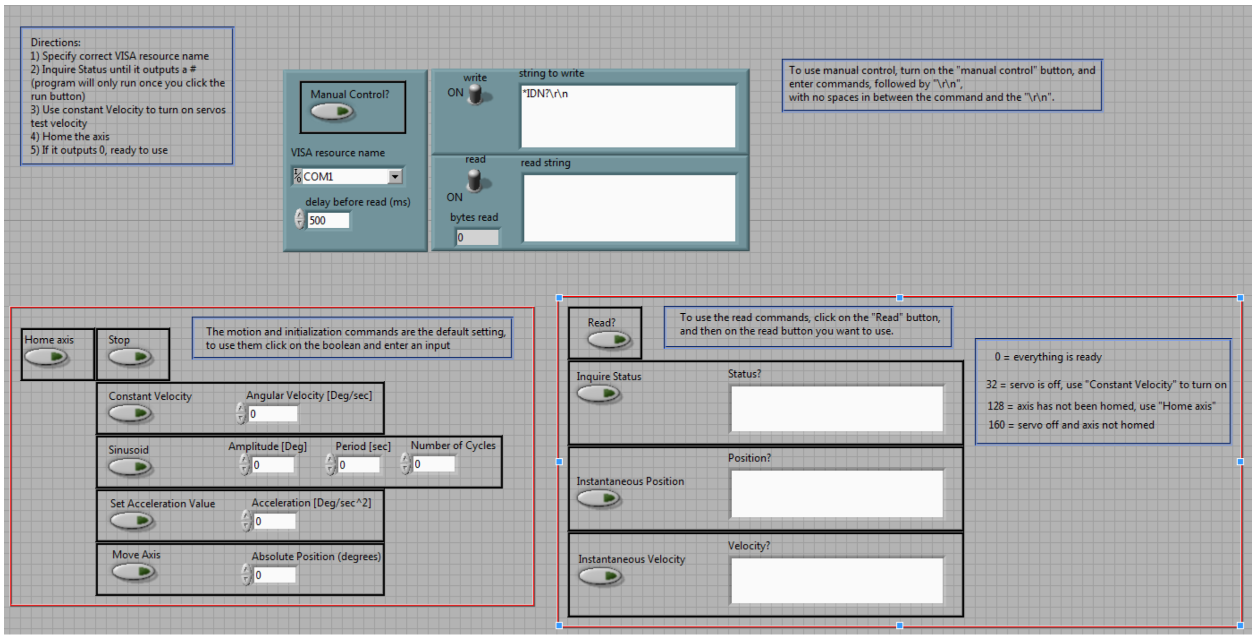


Figure 9, Schematic of Angular Rate Sensor Testing

Typically, ASCII commands are issued to the controller as a string of uppercase characters, sometimes followed by a numerical value, and are always terminated by the “\r\n” carriage return & newline characters. For example, “JOG30\r\n” would command the rate table to spin at 30 degrees per second, “STO\r\n” issues a command to stop the rate table. Information can also be read from the rate table, as one can inquire about the rate table’s velocity by using the command, “PVE\r\n”, which returns the angular velocity in degrees per second. Also, if troubleshooting the rate table, one can issue the command, “STA\r\n”, which yields a number in binary corresponding to a particular set of issues that could be affecting the rate table’s performance, such as: servo is off, or brake is engaged. The front panel for the Rate Table Control VI is shown below, as well as the associated block diagram structures:



Directions:
1) Specify correct VISA resource name
2) Inquire Status until it outputs a #
(program will only run once you click the run button)
3) Use constant Velocity to turn on servos test velocity
4) Home the axis
5) If it outputs 0, ready to use

Manual Control?

VISA resource name: COM1

delay before read (ms): 500

write: ON

string to write: *IDN?\r\n

read: ON

read string

bytes read: 0

To use manual control, turn on the "manual control" button, and enter commands, followed by "\r\n", with no spaces in between the command and the "\r\n".

Home axis

Stop

The motion and initialization commands are the default setting, to use them click on the boolean and enter an input

Constant Velocity — Angular Velocity [Deg/sec]: 0

Sinusoid — Amplitude [Deg]: 0 — Period [sec]: 0 — Number of Cycles: 0

Set Acceleration Value — Acceleration [Deg/sec^2]: 0

Move Axis — Absolute Position (degrees): 0

Read?

To use the read commands, click on the "Read" button, and then on the read button you want to use.

0 = everything is ready

32 = servo is off, use "Constant Velocity" to turn on

128 = axis has not been homed, use "Home axis"

160 = servo off and axis not homed

Inquire Status — Status?

Instantaneous Position — Position?

Instantaneous Velocity — Velocity?

Figure 10, Front Panel of Rate Table VI

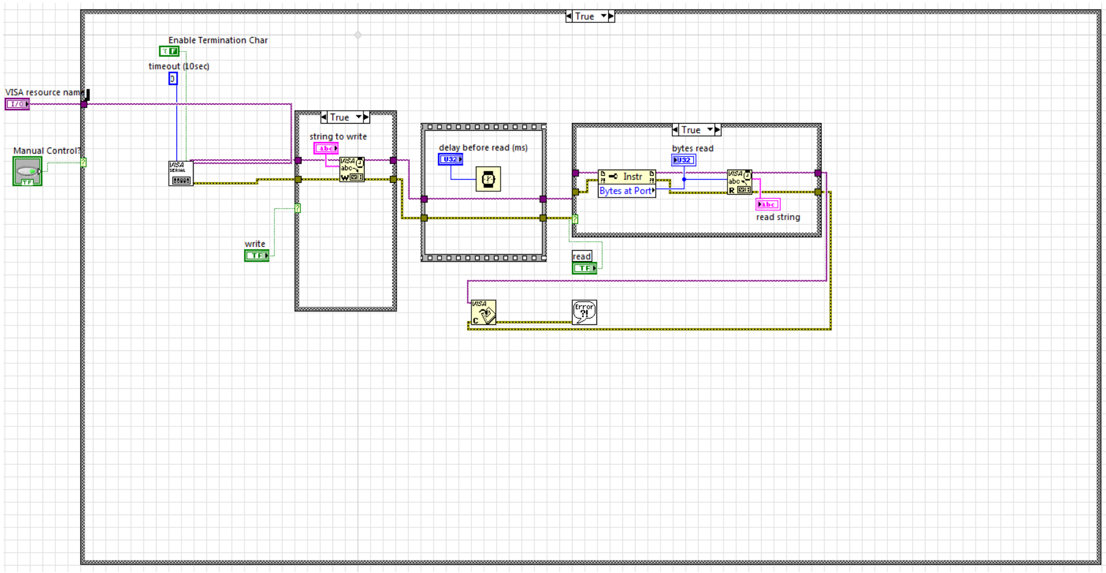


Figure 11, Block Diagram of Manual Control

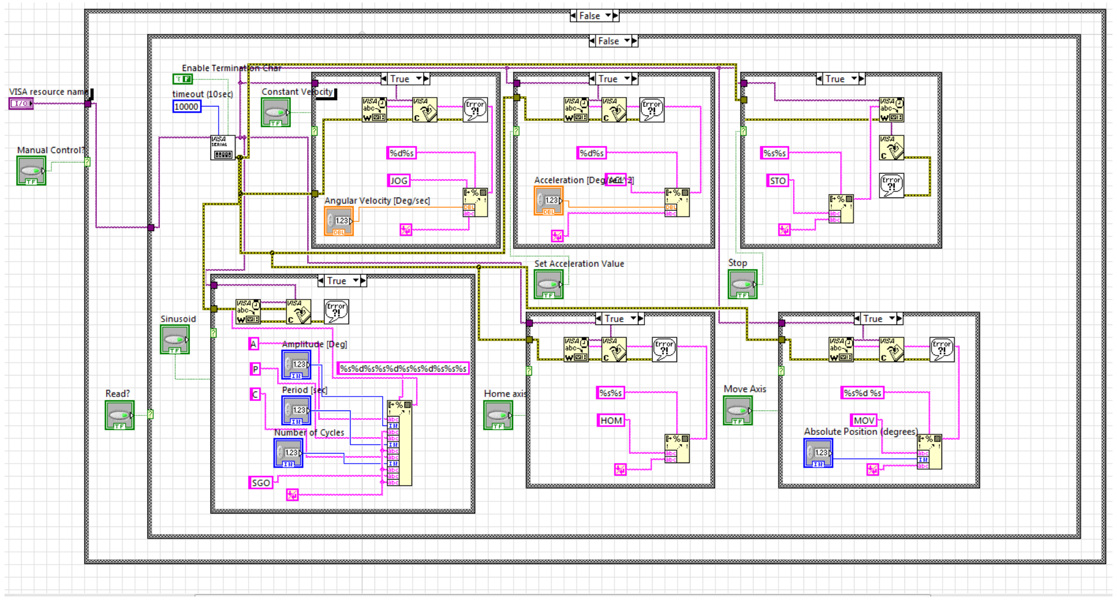


Figure 12, Block Diagram of Automatic Motion Commands

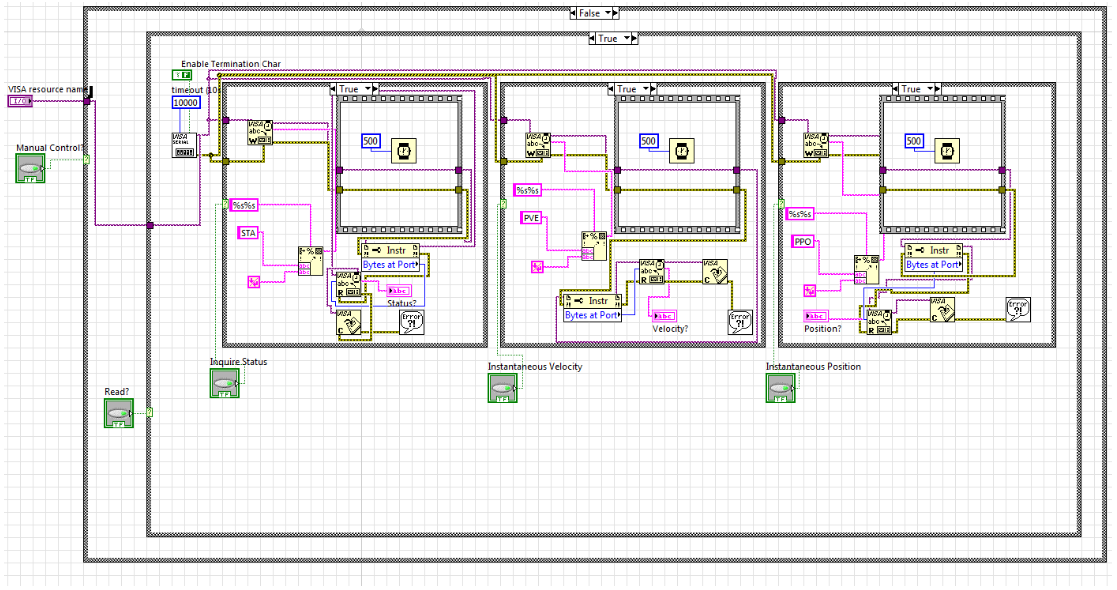


Figure 13, Block Diagram of Automatic Read Commands

The Rate Table Control VI contains three primary sections: Manual Control, Automatic Motion Commands, and Automatic Read Commands. The Manual Control Section of our VI can be seen as the Blue Box near the upper portion of our front panel, and its associated block diagram can be viewed in Figure 11. The Automatic Motion Commands can be seen in the bottom left corner of the front panel, encased in red, and in figure 12, which shows its block diagram. The Automatic Read Commands can be seen in the bottom right corner of the front panel, and its block diagram is shown in figure 13. In determining which of these sections to execute, our program contains two primary case structures. The first evaluates whether to execute the Manual Control section or the Automatic Command Sections. This is accomplished by the “Manual Control?” button, which executes the Manual Control section if it evaluates true, or one of the Automatic Commands if it evaluates false. The second primary case structure discriminates between the Automatic Motion Commands and the Automatic Read Commands, and is evaluated based on the “Read?” button. When the read button evaluates true, the Automatic Read Commands is executed, and if it evaluates false, the Automatic Motion Commands is executed.

If one desires to issue ASCII commands to the rate table manually, the Manual Control section can be used. It is primarily useful when troubleshooting the device, as it enables the user to input all of the ASCII commands and thus determine and fix problems with the rate table. The Manual Control section, shown if figure 11, contains the sub VI Open Serial, which first opens the serial port, enabling communication with the rate table. To write commands to the rate table, the Serial Write VI is used, which has a control on the front panel enabling the user to input a string which gets sent to the rate table. An event structure is used to create a delay before the Serial Read command is used, which then returns information from the rate table, such as its status or velocity, depending on what write command was issued. The Close Serial VI then closes the serial port, so that the user must run the program again to issue a new command. In developing our Labview Program, we used the “Basic Serial Read and Write” VI as a model, and made a few modifications to it in the manual control section of our program.

Much more significant modifications were made in the Automatic Motion Command sections, as we made a number of improvements to enable the user to input commands to the rate table more easily. The Automatic Motion Commands section, shown in figure 12, again contains the Open Serial VI, but then branches off into 6 different case structures, each corresponding to the buttons in the appropriate section of the front panel. When the user clicks on the appropriate button and enters numerical values (if applicable), the case structure is executed to format the correct string to output to the Serial Write VI, which sends the command to the rate table. To format the correct string, the Format String VI was used. One can specify the order in which to enter parameters, which can be constants or controls. For example, for the constant velocity command: JOG30\r\n, the input string is the constant “JOG”, and the next string to append to this is the decimal integer: 30, followed by the \r\n carriage return & newline characters. The format string command needs to place these strings in the correct order as well as get the spacing between them correct. Problems associated with this will be described in the discussion section. Some of the motion commands include: spinning at a constant velocity (“Constant Velocity”), accelerating at a constant rate (“Set Acceleration Value” followed by “Constant Velocity”), moving to a certain position (“Move”), and oscillating at a particular amplitude and period for a specified number of cycles (“Sinusoid”).

The Automatic Read Command section, figure 13, issues commonly used inquiry commands to the rate table using the Serial Write VI, and displays the results in the white box below. Each case structure uses the same functionality as the manual control section, except the correct string is again sent to the rate table automatically using the format string VI. Status inquiries can be made, as well as inquiries regarding the instantaneous axis position and velocity.

Results

Control of the rate table and the shaker were both effective. The rate table controller is accurate in both the velocity and positioning control, as determined from inquiries regarding its velocity and position. Velocity and position errors never exceeded more than half of one percent of the input values. Thus, we can conclude that the data we received from the rate table is very accurate.

Successful instrument control was also achieved for the accelerometer testing with the shaker. The waveform generator sent the signal into the shaker, and the shaker vibrated with an increasing frequency in steps lasting 5 seconds. The graph panel of the oscilloscope controller showed the same response signal of the reference sensor shown in the panel of the oscilloscope.

Data acquisition was also successful for the accelerometer and angular rate sensor, both for the direct wiring to the sensor during the shaker testing, as well as for the wireless transmission during the rate table testing. The static testing data (20min @ 50hz, 6 tests) was put though the least squares fit discussed previously. For the accelerometer this gave sensor noise, drift, and the calibration matrix. For the angular rate sensor this gave sensor noise and drift. Figures 14 & 15 below shows sample data from each sensor for 2 runs: with the X axis up & down and all three channels for each sensor.

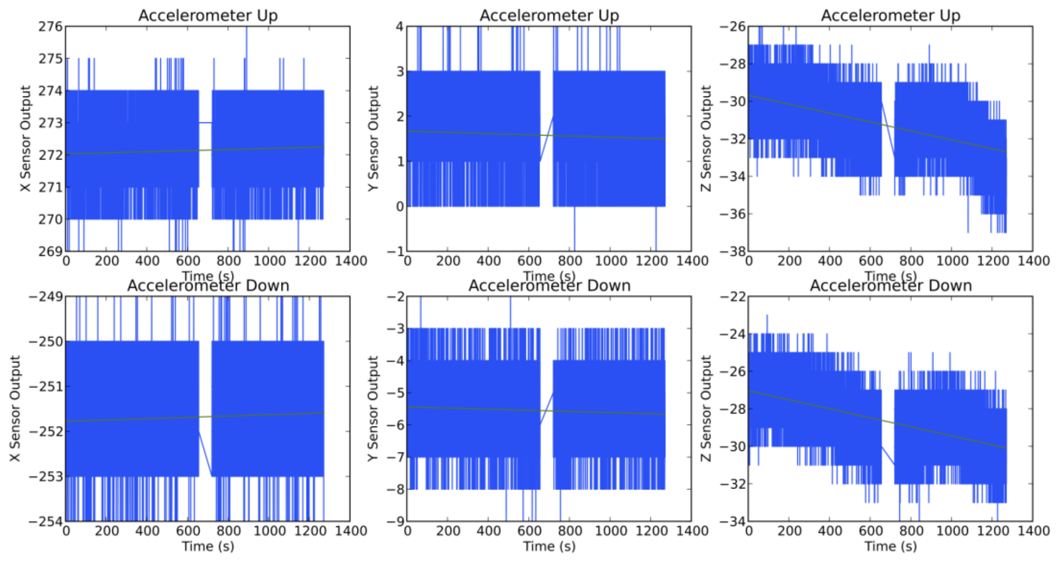


Figure 14. Accelerometer Static Test Data

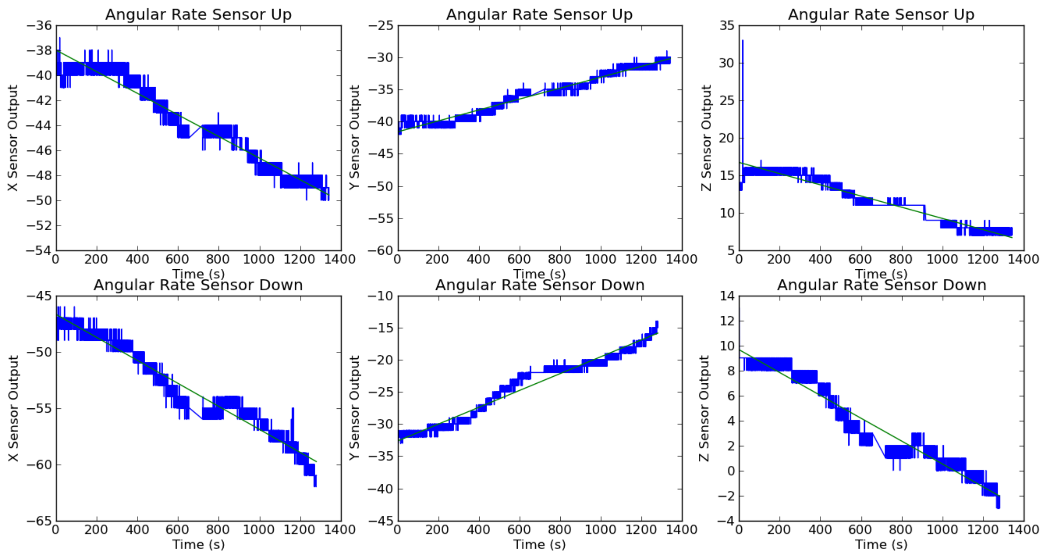


Figure 15. Angular Rate Sensor Static Test Data

The least squares fit gave the following parameters for the static measurements (lsb = least significant bit (defined as the bit position corresponding to the units value in an integer in binary form), m = meters, s = seconds):

Accelerometer

Sensitivity Matrix Format	xx yx zx	xy yy zy	xz yz zz
Sensitivity matrix values [lsb/(m⁄s^2)]	26.72 0.36 -0.13	0.36 -26.7 0.15	-0.29 -0.18 -25.62
	x	y	z
Drift [(m⁄s^2)/s]	5.56E-06	4.01E-06	7.84E-05
Noise standard deviation [m⁄s^2]	0.027	0.026	0.042

Table 1, Sensitivity Matrix, Noise, and Drift for Accelerometer

Angular Rate Sensor

	x	y	z
Drift [lsb/s]	8.01E-03	8.18E-03	6.27E-03
Noise standard deviation [lsb]	1.22	1.062	0.96

Table 2. Drift and Noise for Angular Rate Sensor

The negative values listed in the sensitivity matrix were due to the internal sensor axis conventions being different than what we chose as x, y, and z. After conducting the angular rate sensor tests on the rate table, data was again fitted with least squares for the following parameters and yielded (Figure 16 below shows recorded sensor data):

Angular Rate Sensor Sensitivity Format	xx	xy	xz
	yx	yy	yz
	zx	zy	zz
Sensitivity values [lsb/(deg/sec)]	-14.4	-0.0867	-0.00756
	0.0304	14.4	0.0386
	-0.106	-0.272	14.5

Table 3. Sensitivity Matrix for Angular Rate Sensor

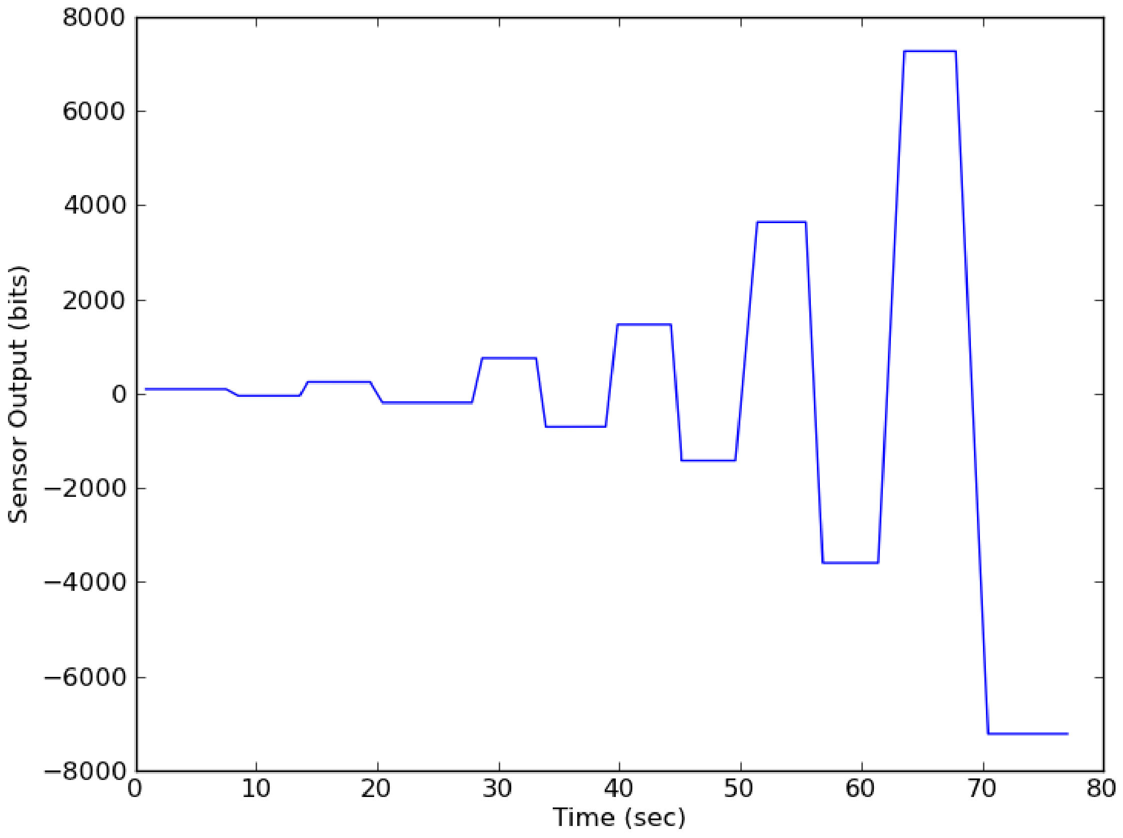


Figure 16. Angular Rate Sensor Dynamic Test Data

It was discovered that the sensor/microcontroller combination could not provide samples faster than around 300 Hz. However, tt was decided to perform the high-frequency accelerometer testing anyways (200 & 400 Hz), as the signal amplitude was the target data, and aliasing would not change the amplitude in this situation. The reference accelerometer signal amplitude was observed on the oscilloscope for comparison to the sensor output. The sensitivity matrix was used to scale the accelerometer signal, which was then compared to the reference accelerometer to get the gain. This was then put on a bode plot to determine the low pass filter cutoff frequency. Figure 17 below shows qualitatively that this is a low pass filter. In this figure, the y-axis is the gain between sensed value and the reference value.

$$Gain = log\left\{\frac{Sensed\,Value}{Reference\,Value}\right\}$$

Frequency [Hz]	x	y	z
15	-15	-19	-18
50	-18	-21	-21
200	-26	-31	-28
400	-39	-38	-36

Table 4, Accelerometer Dynamic Test Data

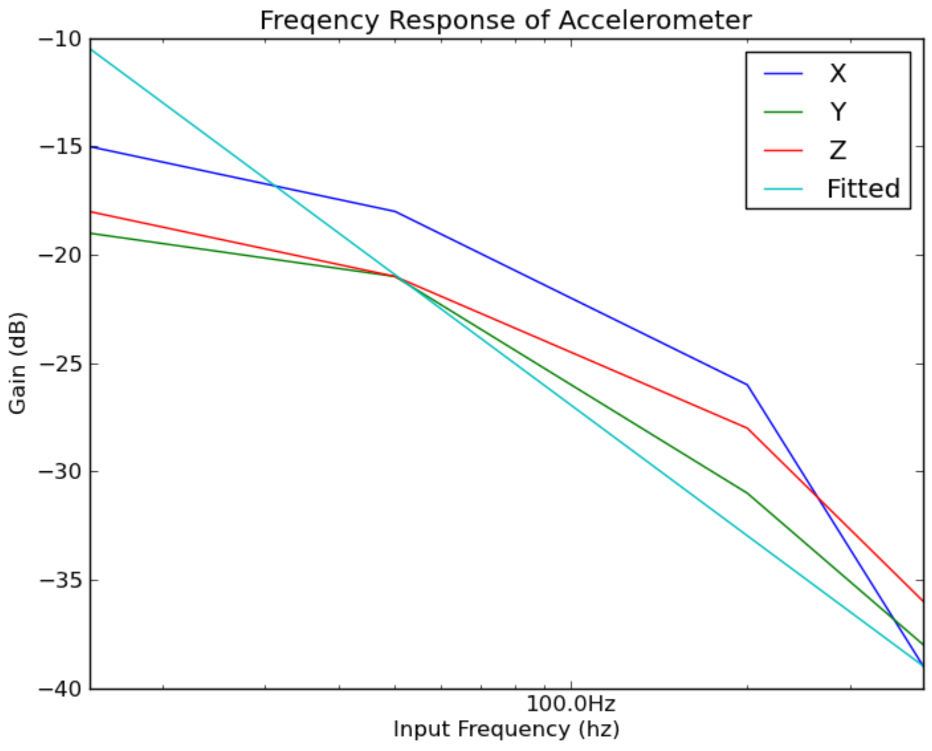


Figure 17. Accelerometer Dynamic Test Data

A line with a 20dB/decade slope was fitted to the data to estimate the crossover frequency with good results, indicating a 1st order low-pass filter. This showed a cutoff frequency of 4.5 Hz when the high frequency asymptote is traced back to the y axis. It should be noted that visual inspection shows that the signal is unlikely to reach the x-axis at 5 Hz. In response to this data, further testing was performed as low as 4 Hz. Unfortunately, the reference accelerometer signal was not as clean at the lower frequencies; also, it would seem we are missing some sort of gain in the reference accelerometer, as at very low frequencies it does not go to zero, but hovers around -20 dB.

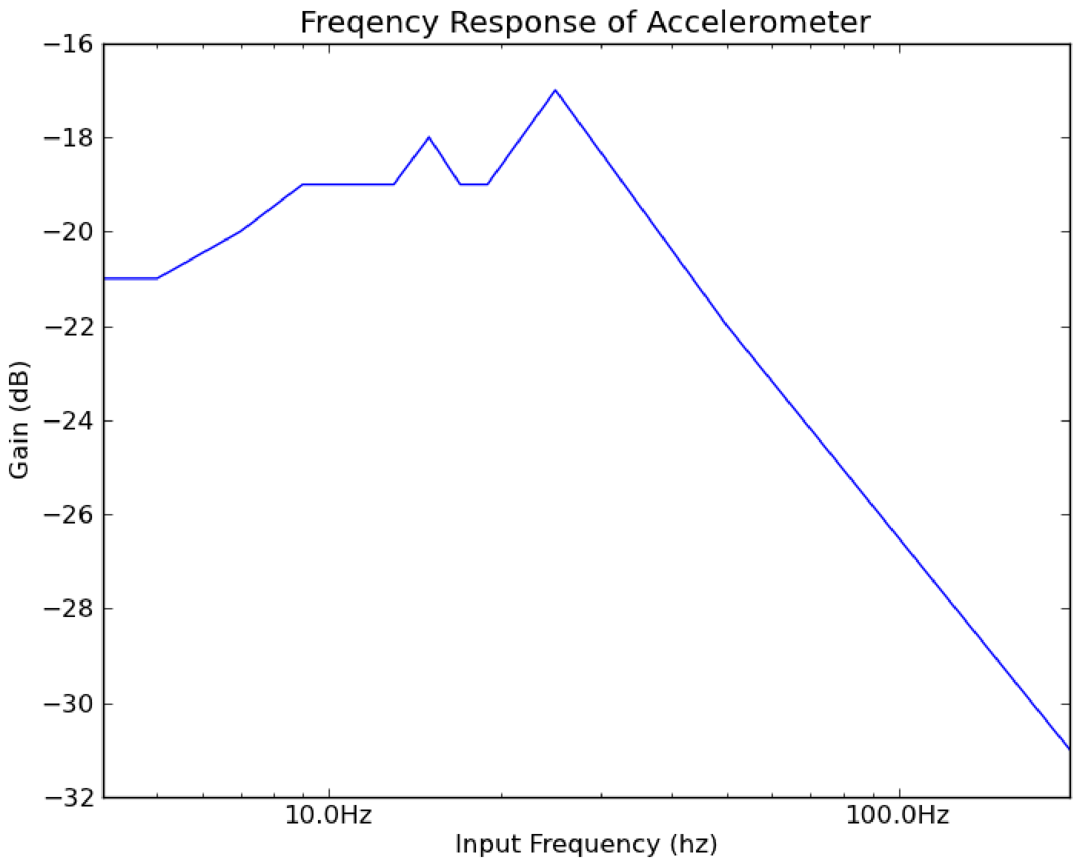
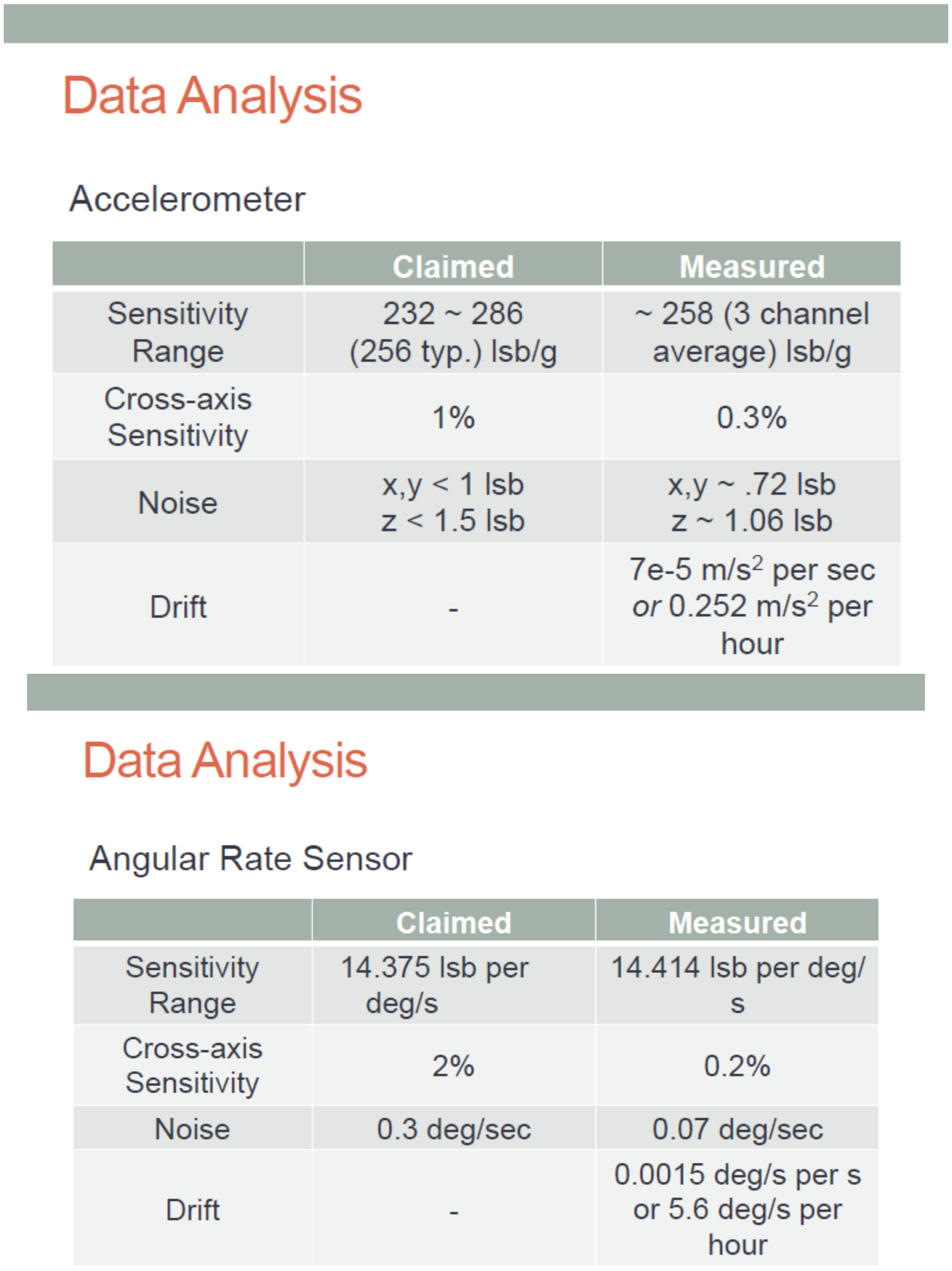


Figure 18. Further Dynamic Accelerometer Testing

Visual inspection shows the peak response to be -17dB at 25Hz; -3dB from this point is around 35Hz. Therefore, we feel confident in saying that the bandwidth is less than 35Hz for this sensor.

<u>Discussion</u>

As this project is a smaller part of developing an IMU, a few topics need to relate back to that level. While we had access to sophisticated lab calibration equipment, not everyone who could potentially use this will. A comparison between the nominal datasheet values and the recorded values was necessary then. Figure 19 below provides this comparison.



Data Analysis

Accelerometer

	Claimed	Measured
Sensitivity Range	232 ~ 286 (256 typ.) lsb/g	~ 258 (3 channel average) lsb/g
Cross-axis Sensitivity	1%	0.3%
Noise	x,y < 1 lsb z < 1.5 lsb	x,y ~ .72 lsb z ~ 1.06 lsb
Drift	-	7e-5 m/s^2 per sec *or* 0.252 m/s^2 per hour

Data Analysis

Angular Rate Sensor

	Claimed	Measured
Sensitivity Range	14.375 lsb per deg/s	14.414 lsb per deg/s
Cross-axis Sensitivity	2%	0.2%
Noise	0.3 deg/sec	0.07 deg/sec
Drift	-	0.0015 deg/s per s or 5.6 deg/s per hour

Figure 19, Comparison Specs

It can be seen that most of the data sheet values can be trusted. The values not given and with large ranges (accelerometer sensitivity and drift values) fortunately can be tested without any specialized equipment. The sensitivity measured by our group turned out to be very close to the values claimed in the data sheet (the sensitivity for the accelerometer shown was converted to g's). The cross axis sensitivities and the noise were also lower than the values claimed in the data sheet, which is good. The drift for the accelerometer was low, and high for the angular rate sensor, as we expected. Determination of these values is important, as they are parameters to the Kalman filter. However, there is some flexibility in acceptable magnitudes. Hopefully, these values will be acceptable.

It was not possible for us to measure the bandwidth of the angular rate sensor for a number of reasons. It was decided that this would be a riskier test, as sinusoidal motion had not yet been fully explored on the rate table and it is quite heavy. Additional work would have been required as well in securing the sensor and microcontroller to the rate table.

Another difficulty was encountered in developing the Rate Table Control VI. Using the Format String VI to concatenate the correct string to send to the Serial Write VI was difficult. We were initially formatting the string to end in the characters: \r\n. This led to the rate table not accepting our input commands. We had to adjust the string to contain the carriage return and newline characters.

As seen in the results, the accelerometer was able to have its bandwidth tested. It was found to be very low. From the datasheets, we were not able to find any onboard low-pass filter parameters that could change. This leaves it ambiguous as to whether the low-pass behavior is from onboard filtering (after sampling) or mechanical limitations. The accelerometer is meant for use in consumer electronics though, where a high bandwidth is not necessary, so the accelerometer could very likely be functioning as intended by the manufacturer.
Originally, a smaller microcontroller was going to be used. Difficulty was encountered in using it wirelessly though. It was found the onboard voltage regulator could not provide sufficient current to the sensors, processor, and wireless radio; a larger microcontroller was used instead.

A magnetometer was also going to be calibrated; this would provide additional information for the Kalman filter to use when estimating orientation. Unfortunately, however, communications with it stopped working early on in the project. It was decided to focus instead on the remaining sensors and developing VI's for the lab equipment. This also led to our team name changing from 9 DOF to 6 DOF.

Colophon

Our project to characterize a MEMS sensor and utilize control of laboratory equipment both furthered our understanding of control and data acquisition techniques and fulfilled all of the course objectives.

As mentioned in the proposal evaluation parameters, the first indicator of our project's success was the ability to control both the rate table and the shaker setup (waveform generator and oscilloscope). Both of these succeeded, as we were not only able to control all of their parameters of interest, but were also able to develop Labview software to make data acquisition and control of them easy. For example, our programs enable us to make frequency sweeps to the shaker and capture data from the oscilloscope for each frequency input, as well as easily implement control of the rate table and determine angular velocity from the encoders. We also learned about wireless data acquisition, as well as the necessary hardware and software needed to transmit data wirelessly. Some important processes to take into consideration were the sampling frequency limitations that existed when we were characterizing the accelerometer (we saw aliasing of signals higher than 150 Hz, although this did not affect our measurements, as we were only interested in the magnitude at a particular frequency).

We also gained valuable knowledge into how to characterize sensors, learning more about noise, sensitivity, and frequency characterization. Because calibration of the actual sensor was necessary, we learned about how to characterize a sensor from the ground up, which is useful in terms of being able to actually test a sensor that is fabricated here at UC Davis. We utilized GPIB cables to connect our computer to both the waveform generator and oscilloscope, and used a serial cable to control the rate table, which enabled us to better understand how to interface computers with laboratory equipment. We learned about these various laboratory devices in more detail, including how to use the shaker and its power amplifier properly, as well as using a reference accelerometer to determine the actual acceleration the accelerometer was subjected to. As mentioned earlier, the rate table has not been previously used in the Horsley lab; thus, our lab will benefit from our efforts in controlling the rate table.

Finally, we learned more about both the graphical programming language Labview, as it took a significant effort to develop the correct string concatenation format to control the rate table, and the correct structure to both control the shaker and acquire data from the oscilloscope. The programming language Python was also used to acquire and process the data from the sensor, helping us improve our understanding of both of these programming techniques. As mentioned, we used basic data analysis in fitting a line to the data to determine the drift and sensitivity, and furthered our understanding of frequency analysis in characterizing the accelerometer's bandwidth.

Overall, this project was beneficial to all of us, as we now know much more about sensor characterization and control than when we started. It was also beneficial to the Horsley lab because we implemented control of the rate table, as well as to the Hubbard lab, as their sensor is now calibrated and ready to be used in a real application.