A user interactive calibration program for an object tracking system using a triaxial accelerometer

2007

Richard A. Elliott

University of Central Florida, elliottra2001@yahoo.com

Find similar works at: https://stars.library.ucf.edu/honorstheses1990-2015

University of Central Florida Libraries http://library.ucf.edu

Part of the Computer Engineering Commons

Recommended Citation

https://stars.library.ucf.edu/honorstheses1990-2015/688

This Open Access is brought to you for free and open access by STARS. It has been accepted for inclusion in HIM 1990-2015 by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
A USER INTERACTIVE CALIBRATION PROGRAM FOR
AN OBJECT TRACKING SYSTEM USING A TRIAXIAL
ACCELEROMETER

by

RICHARD A. ELLIOTT

A thesis submitted in partial fulfillment of the requirements
for the Honors in the Major Program in Computer Engineering
in the School of Electrical Engineering and Computer Science
and in The Burnett Honors College
at the University of Central Florida
Orlando, Florida.

Fall Term 2007

Thesis Chair: Dr. Chris Bauer
ABSTRACT

A major method in object tracking systems and other inertial measurement devices revolves around the use of one, two, or three axis accelerometers. A leader in the field such devices is Microstrain Incorporated. They have developed a three axis accelerometer that uses a three axis magnetic sensor array to compute the pitch, roll, and yaw of a compact inertial measurement unit.

In researching such devices, it became apparent that data collected using such units is extremely sensitive both to local magnetic fields and human interactions with the devices. It is therefore of great importance to ensure the device or devices are properly calibrated. In the construction of an effective calibration program, it is necessary to measure and zero out even minor discrepancies, as even small misalignments have deleterious effects on device performance.
DEDICATIONS

To my parents, for their support now and always.

To Frank Barclay, my respected teacher.

And to Guy Kinney for assistance in writing this thesis. Thank you for an excellent education in the composition of the English language.
ACKNOWLEDGEMENTS

First and foremost, thank you to my committee members. Especially my mentor Dr. Chris Bauer. You fostered this project from inception to completion and this thesis would not have succeeded without your invaluable guidance and support.

Thank you also to Grissel Guzman-David, the skilled and supportive administrative assistant of the School of Electrical Engineering and Computer Science.

And thank you to Barry Trutor and Michael Robinson of Microstrain Corporation for cooperation.
TABLE OF CONTENTS

Introduction ..................................................................................................................... 1

Chapter 1: Inertial Measurement Systems .............................................................. 4

1.1 Overview .............................................................................................................. 4
1.2 Accelerometers ............................................................................................... 5
1.3 Accelerometer Calibration .................................................................................. 11
1.4 Magnetometers ............................................................................................... 16
1.5 Gyroscopes ....................................................................................................... 22
1.6 MEMS .............................................................................................................. 23

Chapter 2: The 3DM-M Device .................................................................................. 26

2.1 Overview ......................................................................................................... 26
2.2 System Requirements ..................................................................................... 27
2.3 3DM Family of Devices ............................................................................... 28
2.4 Data Acquisition ............................................................................................. 35
2.5 Microstrain® Software Versions ..................................................................... 36
2.5.1 Overview ..................................................................................................... 36
2.5.2 MS DOS® Software .................................................................................. 37
2.5.3 MS Windows® Software ........................................................................... 37
2.6 Device Communication Protocols .................................................................... 43

Chapter 3: Microstrain's Calibration Program ..................................................... 48

3.1 Configuration Routine .................................................................................... 48
3.2 Device Calibration .......................................................................................... 50

Chapter 4: My Calibration Program ......................................................................... 54

4.1 Overview ......................................................................................................... 54
4.2 Calibration Routine ......................................................................................... 55
4.3 User Interface .................................................................................................. 58
4.4 Conclusions ..................................................................................................... 60

APPENDIX A ............................................................................................................. 66
APPENDIX B ............................................................................................................. 68
APPENDIX C ............................................................................................................. 70
APPENDIX D ............................................................................................................. 87
List of References ...................................................................................................... 89
Table of Figures

Figure 1: 3DM-M Device ......................................................................................................................... 1
Figure 2: 3DM Angles Mode .................................................................................................................... 40
Figure 3: 3DM-M Configuration Screen .................................................................................................. 42
Figure 4: 3DM Calibration Screen .......................................................................................................... 51
Figure 5: My GUI Layout .......................................................................................................................... 56
Table of Equations

Equation 1: Accelerometer Linearity .............................................................................. 7
Equation 2: Accelerometer Sensitivity .......................................................................... 7
Equation 3: 3DM Output Conversion Formula ............................................................. 9
Equation 4: ADXL202E Sensitivity ............................................................................... 11
Equation 5: AN3447 Acceleration ............................................................................... 12
Equation 6: AN3447 Acceleration Error ...................................................................... 12
Equation 7: Temperature Coefficient Offset ................................................................ 13
Equation 8: AN3447 Measured Acceleration ............................................................... 15
Equation 9: Magnetometer Bridge Output ................................................................... 19
Introduction

The purpose of this research project was to create a software program using the 3DM-M® inertial measurement device manufactured by Microstrain®, Inc. to create a user-friendly, interactive calibration program. The end use of the calibration program was to allow students the opportunity for practical experience using concepts explained in an engineering course on numerical methods. The 3DM® does have its’ own calibration program, but it does not show any details of the process of the calibration of the device. The user interactive calibration program I created for the device uses a combination of the 3DM software and my own software. The students will be able to write their own programs using my software to study serial communication and numerical integration. I feel that my step-by-step interactive, visual calibration routine allows the students the opportunity to interact with actual hardware, and not just a computer simulation.
Computer simulations allow the exploration of a great breadth of devices and processes. Nevertheless, there is no substitute for learning on real devices that students will encounter in industry.

My program takes the user through the calibration of the 3DM®'s accelerometers and magnetic sensors (magnetometers). The students explore the gains, offsets, sensitivity, and errors encountered, when working with an inertial measurement system. Using a series of well-defined questions and results, the user of my software zeros the x, y, and z axis of both types of sensors. After doing this calibration, the user can experiment with measurements from the three axes of the device, which are known as the pitch, roll, and yaw axis. The data can be analyzed as angular measurements, or as digitized bit values representing acceleration and magnetic field measurements.

An important point in using this device is that it is older technology, which is known as a legacy device. Legacy devices often have either software, hardware, or both software and hardware interaction problems with newer operating systems and peripheral technologies. In using this device, several “problems” have become apparent. The most significant of these problems has to do with the device’s communication protocols. In the approximately ten years since the device was manufactured, hardware communications with external devices are now handled through a high speed universal serial bus (USB®). The 3DM-M® device was designed to use RS232 via a “native” serial communication port. I define a native port as is a port directly connected to the computer’s motherboard. The machine I used for development of my calibration program does not have a traditional DB9 serial communication port. I attempted communication using RS232 with a DB9 to USB® converter. The results were
unacceptable. The 3DM software had numerous interaction problems. I blame these problems on the fact that USB® samples at a much faster rate than RS232 using traditional ports. I believe that the communication protocol differences in these two technologies are responsible for the intermittent sampling problems.
1.1 Overview

Inertial measurement systems are primarily used in inertial guidance systems for vehicles or other moving objects, such as biomedical engineering applications that study movement of limbs, etc. The focus placed on inertial measurement systems for this project was concentrated on systems with devices that had three degrees of freedom (DOF). The 3DM-M® device manufactured by Microstrain®, Inc. has three DOF because of its three linear acceleration sensors and tri-axial magnetometer (magnetic sensor). Most new devices have six DOF, because of the addition of gyroscope (gyro) sensors. The 3DM-M measures acceleration on three axes—x, y, and z. These axes are generally referred to as the pitch, yaw and roll axes, respectively. The 3DM-M also contains three magnetic sensors that measure the strength of earth’s magnetic field along each sensitive axis, which is the axis of measurement.

The error associated with using accelerometers and magnetometers increases due to the use of previous measurements being used to produce new measurements. The error produced in this type of feedback system is handled by applying offsets to the measurements. The magnetometers also reduce the error from the accelerometers. In later versions of the 3DM®, gyroscopes and temperature sensors were added to increase the accuracy of the device. Temperature and the construction of the device enclosure itself are the predominate sources of error. In order to handle temperature errors, it is necessary to use appropriate heat sinks, vents, etc. Due to the nature of the
accelerometers and magnetometers inside the 3DM®, it is necessary to construct the enclosure of non-ferrous metals such as aluminum or brass, and plastics.

1.2 Accelerometers

Originally accelerometers used quartz crystal or a moving coil to measure acceleration. [8] There are six common types of accelerometers in use today. They are capacitive, piezoelectric, piezoresistive, hall effect, magnetoresistive, and heat transfer sensors. One of the more common accelerometers used today is a suspended-mass type sensor. Suspended-mass accelerometers are characterized as solid-state devices that are about as big as a transistor. [6] A key feature is their ability to respond to DC (static) accelerations. One type of suspended-mass accelerometer of interest uses capacitive sensing to produce an electrical signal proportional to the displacement. My main use for the data from the accelerometers is to find the displacement. Once I have the displacement, it is relatively direct to determine how close the data for each axis is to the origin. The 3DM-M® uses capacitive acceleration sensors. Capacitive accelerometers are great for low g applications. Piezoelectric sensors use a crystal mounted to a mass. These accelerometers are better utilized in more rugged applications. The voltage returned is then converted to acceleration. A major problem with piezoelectric accelerometers is the need to maintain ‘high insulation resistance’ and low noise in the cabling between the accelerometer and the charge amplifier. [8] Piezoresistive accelerometers use a beam or micromachined device, whose resistance changes with acceleration. Hall effect sensors convert motion into an electrical signal by detecting
changing magnetic fields. A magnetoresistive accelerometer measures material resistivity changes in the presence of a magnetic field. Lastly, when heat transfer is used to detect acceleration, the location of the heated mass is tracked during acceleration by temperature sensors.

A common use of accelerometers is to measure tilt. Tilt is more commonly a concern, when using inclinometers. It is, however, also critical in calculating the offsets necessary to properly calibrate most accelerometers. The effect of tilt on the output of DC accelerometers is significant enough to warrant an extra sensor or other means dedicated to removing the induced error. The 3DM has on-board tilt sensors that correct for deviations from zero when the device is on a level surface. [2] The method of mounting a device also causes errors easily handled by tilt sensors. According to research done by Texas Instruments on their own devices, they found out that one degree of tilt in the 0g reference position creates an output error equivalent to ten degrees of tilt at the +/- 1g position. [6] The 1g force is due to the surface on which the device is resting; pushing up on the accelerometer with a force equal to the weight of the accelerometer. In other words, at 0g orientation, a change in tilt of one degree causes 57 times larger change in sensor output versus what it would be at +/-1g. TI uses a convention for their devices in which the device experiences +/- 1g, when the accelerometer is pointed up or down, and laying on a flat surface, respectively. Their devices use the 0g reference point for when the accelerometer is on its side. No device can in reality have 0g acceleration. The point is simply used as a means of reference in calculations.
In measuring the acceleration of an object, the need for accuracy is paramount. It is essential in order to calibrate an acceleration sensor. In making the calibration program, I had to be sure that the values I was getting were reasonably correct. I found two formulas that will prove very useful in determining the accuracy of the measurements from the 3DM® device. Both formulas were very important as a part of the new calibration program. The first formula is a linearity test that determines the maximum deviation of the calibration curve from a straight line. [6] It uses the output voltages from the accelerometers at the three reference points (0g, 1g, -1g).

**Equation 1: Accelerometer Linearity**
\[
Y = V_{\text{out,}0g} - \frac{1}{2}(V_{\text{out,}+1g} + V_{\text{out,-1g}})
\]

In the above equation, \(V_{\text{out,}+1g}\) is the output voltage at the +1g reference and \(V_{\text{out,-1g}}\) is the output voltage at the -1g reference. Another important formula that measures how much the output of a sensor changes, as the input acceleration changes, is called the sensitivity formula. [6] It is measured in volts per g, where ‘g’ is the gravitational acceleration. The sensitivity is the change in output voltage per change in gravitational acceleration.

**Equation 2: Accelerometer Sensitivity: ‘g’ = (V_{\text{out,}+1g} - V_{\text{out,-1g}}) / 2g**

In the above equation, \(V_{\text{out,}+1g}\) is the output voltage at the +1g reference and \(V_{\text{out,-1g}}\) is the output voltage at the -1g reference. According to TI®, some typical applications for their accelerometers are vibration detection, tilt/roll, impact detection, and vehicle skid detection.

There are several important factors that should be monitored to ensure proper readings from the accelerometers. First, the maximum deviation of the accelerometer output voltage from a ‘best fit straight line’ fitted to a plot of acceleration versus output...
voltage, or nonlinearity should be calculated as a percentage of the full-scale output voltage at 5g. Also, as mentioned before, resonance should be a factor of consideration. Another error to be mindful of is sensor offset drift. Whenever a dc accelerometer is used there will always be some temperature drift at the 0g offset level. An excellent means of compensation is to add a temperature sensor. It is also very important, when choosing an accelerometer is the resolution. The resolution of an accelerometer is the lowest g-level the accelerometer is capable of measuring. It is determined by both the device noise and bandwidth of measurement. The lowest level does not include the 0g level, which is primarily a reference level that one can use to measure the output of the accelerometer, when it is not in motion. As for the noise “floor”, this includes the ambient background noise, as well as that of the accelerometer. The level of this “noise floor” is directly proportional to that of the bandwidth of measurement. The lower the bandwidth; the better the signal to noise ratio of the measurement and increase of resolution. The bandwidth of the accelerometer can easily be reduced by adding a low or band pass filtering method. Microstrain® uses a successive weighted average infinite impulse response filter that can be adjusted based on the number of samples used to calculate the average.

The one property that makes a DC accelerometer so important, especially in my application, is that the sensor can measure 0Hz—static position. In order to properly calibrate the 3DM®, the device had to be rotated at least 360 degrees about each axis. The device had to be able to lie motionless at the origin in order to calculate the gains and offsets. The offsets can then be subtracted out, when it is certain that the device is at its default position. The device will then be rotated and the results recorded into data files. I
have created several solid calibration tests that will run off of these data files. After running the calibration program, the accelerometers registered very little movement. Due to the nature of the device, and the materials from which it was constructed, there will always be a little jitter registered by the accelerometers.

The 3DM-M® device uses three capacitive, solid-state DC accelerometers. The acceleration data is returned from the device in one of two modes—polled or continuous. The most useful mode for a calibration routine is continuous mode. The 3DM® can output either angles or bits, when in continuous mode. The angles are the pitch, yaw, and roll. In bits mode, the device sends either separately or together, the raw data from the accelerometers and magnetometers. The data is transmitted from the analog to digital (A/D) converter. In other words, data processing within the device is analog, but the output is digitized. The accelerometer data is first calculated using voltages, and then returned in a digital representation of accelerations and magnetic field values. [2] For the data to be useable, a conversion formula must be applied to the data. The conversion formula takes the value of the gains, offsets, and applied forces; and uses it to calculate adjusted data values.

Equation 3: 3DM Output Conversion Formula: Acceleration = \(2 \times (output - offset) / gain\)

In the conversion formula, output is the raw output from the device. After the conversion formula was applied, the acceleration was determined for the x, y, and z axis in g’s, where a ‘g’ is the acceleration of gravity at sea level. The value is 32.2 ft/s\(^2\) or 9.81 m/s\(^2\) depending on whether you are using English standard or metric units, respectively. The applied forces, g’s, are of no practical uses without the application of the conversion formula.
The accelerometer used in the 3DM-M® is the ADXL05JH® single chip accelerometer made by Analog Devices. [16] The 3DM-M® contains two of these devices. The x and y-axis values are calculated using one chip. The z-axis data is calculated using a separate chip. The manufacturer of the ADXL05 family of devices, which includes the ADXL05JH®, can measure accelerations with full-scale ranges of +/-5g to +/-1g or less. [16] These sensors are designed to measure accelerations provided by an applied force. In the case of the 3DM®, the applied force will be a human hand turning the device about its three axes. When the device is positioned relative to earth’s gravity, the pitch and roll axis accelerometers should experience an acceleration of 0g. The roll sensor should experience a -1g acceleration. Acceleration is measured along the accelerometers sensitive axis. The sensitive axis for these accelerometers is perpendicular to the transverse z axis. Based on information from the datasheet, “an ideal sensor will react to forces along or at angles to its sensitive axis, but will reject signals from its various transverse axes—those exactly ninety degrees from the sensitive axis. If acceleration is acting in any other direction upon the sensor other than the sensitive axis, the output from the sensor will have reduced amplitude. The manufacturer does state that “small errors in alignment have a negligible effect on the output signal.” A common source of error for these accelerometers is resonance of the mounting fixture. [7] The data sheet gives as an example a circuit board that the accelerometers mounts to may have resonant frequencies in the same range as the signals of interest. The suggested solution is to dampen the resonances by mounting the sensors near a mounting post or by adding extra screws to hold the board in place. [12] I don’t see this as relevant to the purposes of this project in that the device containing the sensors will not be mounted.
Any resonance should be negligible. Further, in the calibration situation, the accelerations experienced by the device will be very small and mostly between 0 and -1g. Therefore, the internal signal conditioning circuitry should be able to handle any ambiguities. Nevertheless, the device will be tested along all three axes to verify that all data is reasonably accurate.

1.3 Accelerometer Calibration

The most generic method for calibrating accelerometers is known as flip calibration. It is the most direct method for calibrating accelerometers using just earth’s gravity. For low g applications the force of gravity is the most stable, accurate, and convenient acceleration reference available. \[13\] The easiest place to use as a reference is the 0g position. At the 0g point, you orient the device parallel to the earth’s surface, and read the output. For a more accurate reading, measurements should be taking at the +/- 1g reference points. The +1g reference is when the device is laying right side up on a flat surface. The -1g reference is when the device is laying upside down on a flat surface. Using measurements at these two points, you can determine the sensitivity. In order to calibrate the accelerometers, orient the accelerometer’s measurement axis parallel to the earth’s surface. Let the +1g reading equal A, and the -1g reading equal B.

Equation 4: \[\text{ADXL202E Sensitivity} = \frac{[A - B]}{(2 \times g)}\]

The only major caveat in using this approach is that you must be very careful about the local value of gravity, as the output readings are based solely upon it.
A similar calibration method to flip calibration is that known as auto-zero calibration. The technique is based on the sampling of the output voltage of the accelerometer at 0g, which is known as the zero reference. The value at the zero reference is known as the offset voltage. Offset errors can occur device to device based on offset variations from trim errors, mechanical stresses from the package and mounting, shifts due to temperature, and due to aging. [13] Offset error can be very significant in that a tilt reading on a flat surface can be off by as much as twelve degrees, experimentally. Using auto-zero calibration, theses errors can be greatly reduced. Note, however, that the correction of error is limited by the resolution of the A/D converter. It is explained in the article that “a two point acceleration calibration can be performed to accurately determine the sensitivity and get rid of the offset calibration errors.” [13] In order to determine the acceleration, you need to be given or calculate the sensitivity and offset data for the given device.

**Equation 5: AN3447 Acceleration: Acceleration(g) = (V_{out} - V_{off}) / S**

In the acceleration equation above, \( V_{out} \) is the output voltage, \( V_{off} \) is the offset voltage, and \( S \) is the sensitivity. If an offset calibration error, \( \Delta V_{off} \), is introduced, then there will be a corresponding error in the accelerometer output.

**Equation 6: AN3447 Acceleration Error: g + \Delta g = [V_{out} - (V_{off} + \Delta V_{off})] / S**

In the above equation: \( V_{out} \) refers to the output voltage, \( V_{off} \) refers to the offset voltage, and \( S \) is the sensitivity. The change in acceleration is then added to the experimentally determined value of acceleration. Mounting and orientation seem to be the most common offset errors resulting in calibration errors. [7] It is important to check
the minimum, maximum, and expected offset values on the accelerometer datasheet before proceeding with calibration. Temperature also commonly leads to offset errors, which are due to the Temperature Coefficient of Offset (TCO). [13] The TCO parameter is the rate of change of the offset, when the sensor is subjected to temperature.

**Equation 7: Temperature Coefficient Offset: TCO = (ΔV_{off}/ΔT)**

In the equation, ΔV_{off} is the change in the offset voltage and ΔT is the change in temperature. There are several means commonly used to calibrate the accelerometer and calculate the offset voltage. They include a manual, full range calibration along all three axis with respect to the 0g reference; a simple three axis calibration routine using the 0g reference; a freefall calibration technique; a calibration along the x and y axis based on the 0g reference, and for the z axis at the +1g reference.

An interesting technique is a three axis, full range, manual calibration technique that uses the 0g reference point, an accurate 0g reference offset value must be taken. [13] If the reference is off by even a little bit, it can render the test invalid and useless. Therefore, the tester must be certain that the device is on a level surface. It is also just as critical if not more so to ensure a level surface, when taking measurements with a +/- 1g reference. In addition to a table that is not level, the user must be aware of minor g forces due to the packaging of the device and device shifts. A very simple method to obtain an accurate 0g reference reading involves simply rotating the device from +1g through -1g. The maximum output voltage value will be at the +1g reference, and the minimum offset voltage will be at the -1g reference point. In order to calculate the sensitivity, you must assume that the sensitivity is symmetric from zero to positive 1g and zero to negative 1g. After this assumption is made, the sensitivity is calculated simply by averaging the two
symmetrical sensitivity values. After you know the sensitivity, the 0g offset can be calculated by adding the sensitivity to the minimum offset voltage \( V_{\text{off,min}} \), or by subtracting the value of the sensitivity from the maximum output voltage, which was calculated at the \(+1\,g\) reference point. It is recommended that after you do the previous calculations you then place the part level at the 0g reference and double check the value. The observed value and the calculated value should be very close. In order to calibrate all three accelerometers, you must do the calculations three times for each axis. In a simpler method of the 0g reference calculation, you assume that the device is on a level surface. The next step then would be to turn the device ninety degrees once to go from the \( x \) and \( y \) axis values at 0g to the value of \( z \) at 0g. The simple 0g calibration provides a quick solution, but at the expense of accuracy. [13]

A quite different approach uses freefall to obtain the measurements needed for calibration. The freefall method is outside the realm of possibility due to the risk to the device. Nevertheless in the freefall calibration, all three axes are calibrated at the 0g reference at once, while the device is falling. The time savings is a major advantage, but this calibration technique is subject to error in that the device may rotate while in freefall, which will add a force to the device. I consider this technique to risky for the delicate device that I am calibrating. It is, however, helpful on a theoretical level. Simple calibration of \( x \) and \( y \) at 0g, and \( z \) at \(+1\,g\): A second method of simple calibration is to calibrate the \( x \) and \( y \) axis offsets at the 0g reference, while calibrating the \( z \) axis offset at the \(+1\,g\) reference. The process for this technique is to place the device on a level surface, so that \( x \) is at 0g, \( y \) is at 0g, and \( z \) is at \(+1\,g\). The offset values are recorded, while the device is lying in this position. Using this method, the \( x \) and \( y \) values are fairly accurate,
but the value for the z axis would include errors, since it was not recorded at 0g. In order to rectify the error in the z output, the known sensitivity would have to be subtracted from the +1g value. In doing this, an assumed value of the offset at 0g is calculated for the z output voltage. The major advantage of this method is that the device does not have to be moved, which saves time. However, the downside is that it is the least accurate way to calibrate the 0g offsets.

As noted in AN3447©, an auto-zero calibration is a calculation at the 0g reference. In order to do an automatic calibration instead of one of the manual calibrations, you need a microcontroller with an acceptable A/D converter resolution. [12] The auto-calibration technique does reduce offset errors, but an error is introduced due to the resolution of the A/D converter. The only major limiting factor in this calibration technique is that of the converter resolution. The technique will also work using the alternate methods, where the z offset is calculated using the +1g reference. The offset and acceleration values are then used to calculate the measured acceleration.

**Equation 8: AN3447 Measured Acceleration:** \( A_{\text{meas}} = \frac{[CA - CZ_{\text{off}}]}{S} \)

In the above equation: CA is the current acceleration, CZ_{off} is the current zero offset for whichever axis you are calibration, and S is the sensitivity for the axis. An important thing to keep in mind, when using this technique is that it must be performed as often as possible in order to dynamically compensate for system offset errors. Note, the TCO offset error and the calibration error will not be corrected if the sensor later experiences a wide temperature range or offset shift until the calibration is redone. The auto-zero calibration technique is best used, when the acceleration sensors are interfaced with a microcontroller. If the simple 0g x, 0g y, +1g z technique is used, the measured
acceleration uses a slightly different equation. Although these techniques are very accurate and simple to test, I did not use them, since I do not have access to raw currents or voltages. However, they provide a clearer explanation of how accelerometers and their components function.

### 1.4 Magnetometers

Magnetic sensors (magnetometers) detect the presence of ferrous metal objects. The objects change the magnetic flux through the magnetometer coils, which generates a voltage at the coils terminals. The magnetometers on the 3DM compensate for errors due to tilt, and ensure the accuracy of the accelerometers.

The 3DM uses the HMC1021® and the HMC1022® magnetic sensors manufactured by the Honeywell Corporation. The HMC1021® is a one axis magnetometer. The HMC1022® is a two axis magnetometer. The HMC1021® and HMC1022® are used in combination to provide magnetic sensing along all three axes. [15] Honeywell also manufactures a three axis device, the HMC1023®. The HMC1023® serves the same purpose, and performs the same operations as the HMC1021® and HMC1022®. The designers of the 3DM-M®, however, chose to use the two chip solution. The HMC1021® measure ambient and applied magnetic field values along the x and y axes; and the HMC1022® measures the same for the single z-axis. Whether you are using the two separate sensors or the combine sensor, the Honeywell chips provide three axis sensing in a small, solid state, cost-effective package.
They offer a wide field range of up to +/- 6 gauss. For comparison, the earth’s magnetic field is 0.5 gauss.

The Honeywell sensors also use a unique, patented process of offset and set/reset straps instead of wrapping the sensors in metal coils. [15] The set/reset straps reduce the effects of several common errors, which include temperature drift, loss of signal output, and non-linearity errors. Signal loss is a common error resulting from the presence of high magnetic fields. As long as the device is kept away from strong magnets and ferrous objects, this should not be a problem. [10] The best materials to use in the construction of an enclosure for the sensors are one made of aluminum, brass, or plastics. Honeywell’s offset straps provide a unique protection against hard iron distortion among other things. The combination of the one and two axis chips reduces board assembly costs, improves reliability, and ruggedness. [15] The solid state nature of these devices also eliminates common errors associated with physical components used on traditional magnetic sensors. The two chips use a four element wheatstone bridge configuration to convert magnetic fields to a differential output voltage that is capable of sensing magnetic fields, as low as 30 micro-gauss.

The Honeywell sensors are of the ‘magneto-resistive’ type. The simple resistive nature of the sensors allows them to measure any ambient or applied magnetic field in the direction of the sensitive axis with only a small supply voltage. The bridge circuit, which is the heart of this type of magnetic sensor, has in addition two on-chip magnetically coupled straps. They are the set/reset strap, and the offset strap. The straps are unique in their application. They are made from a nickel-iron thin film placed on a silicon wafer. In the presence of an applied magnetic field, a change in the resistance in the bridge
circuit causes a corresponding change in the voltage output. When an external magnetic field is applied normal to the side of the film, the internal magnetization vector rotates and changes angles. The vector realignment is what causes the resistance change followed by the change in output voltage. The change in resistance is directly related to the angle of current flow and the magnetization vector.

In production, the Honeywell magnetic sensors are set to a preferred field direction. The field is set to one direction along the length of the film, which allows the maximum change in resistance for an applied field. The sensor offset strap affords the user several options, when measuring a magnetic field with a DC current being driven through it. First, an unwanted magnetic field, such as a local ambient field, can be subtracted out using the offset strap. Second, the bridge offset can be set to zero. The bridge output can also drive the offset strap in order to cancel out the field being measured in a closed loop configuration. Lastly, the bridge gain can be auto-calibrated in the system. According to information found in the device datasheet from Honeywell, the set/reset (S/R) strap can be pulsed with a high current to force the sensor to operate in high sensitivity mode, flip the polarity of the output response curve, and be cycled during normal operation to improve linearity and reduce cross-axis and temperature effects.

The offset in the horizontal direction is called the external offset. It may be due to a nearby ferrous object or unwanted magnetic field that is interfering with the applied field attempting to be measured. The external offset error can be easily fixed by applying a DC current in the offset strap, which will adjust the offset to zero. Another solution to this problem is to shield the sensor using plastic or a non-ferrous metal, such as aluminum or brass. Since I cannot affect the magnetic sensors without placing an
extremely strong magnet next to the 3DM, which would actually put the device in a mode
counter to what I need, my only choice is to shield the sensors. If the sensors are
functioning correctly, the output voltages should resemble a sine wave. The ultimate
need here in this situation is that the offset from the magnetic sensors be set to zero, or as
close as possible to zero. Another major problem for the magnetic sensors is noise. The
noise density curve for the typical magneto-resistive (MR) sensor should have a slope
that is the inverse of the operating frequency with a corner frequency near 10Hz that goes
down to no less than 3.8 nV/(Hz⁻¹/₂). The acceptable amount of noise in most situations
is approximately the above, which is typically referred to as white noise. The typical
bridge resistance should be close to 850 ohms. According to the manufacturer, if the MR
sensor stays in a fixed position, then the effect of the device using the sensor on the
earth’s magnetic field can be approximated by a shift or offset. If the value of this shift
can be determined, then it can be compensated for by applying an equal and opposite
field using the offset strap. When analyzing the applied magnetic field is important to
keep in mind that the field measured by the sensor is a combination of the ambient field,
as well as the offset field.

Many unwanted effects can also be eliminated by using the set/reset (S/R) strap. Among the effects that can be eliminated are temperature drift, non-linearity errors, cross-axis effects, and loss of signal output due to the presence a high magnetic field. The S/R strap can be reset by a pulse of equal and opposite current through the S/R pins. It can be set by driving a current pulse from the S/R+ pin to the S/R- pin. Offset and temperature effects can also be canceled out by using the following equation.

**Equation 9: Magnetometer Bridge Output:** \( \frac{V_{\text{out(set)}} - V_{\text{out(reset)}}}{2} \)
In the equation, $V_{out(set)}$ is the initial output voltage and $V_{out(reset)}$ is the output voltage after the pulse. Unlike the offset strap, the S/R strap is magnetically coupled to the MR sensor in the cross-axis or insensitive direction. Once the sensor is set or reset, low noise and high sensitivity field measurements can be made. When the MR sensors are placed near a ‘disturbing’ magnetic field, the sensor elements are ‘broken up’, that leads to a degradation of sensor sensitivity. A current pulse larger than the minimum specification current sent through the S/R strap will generate a strong enough magnetic field to realign the magnetic domains in one direction. A negative pulse would orient the magnetic field in the opposite direction, and change the polarity of the sensor outputs. The magnitude of the S/R current pulse is dependent upon the magnetic noise sensitivity of the system. Although I cannot directly affect the current through the bridge circuit, I can manipulate the ambient magnetic field in an attempt to reduce it. Then, the total field would be comprised in majority by the desired applied field. The circuit that generates the S/R pulse should be located close to the MR sensor. [15] The 3DM-M® device is approximately two inches by three and a half inches, which means signals do not have to travel far, which satisfies the previous condition. After opening up the 3DM®, by inspection, all electrical connections on the circuit board are in excellent condition. They are well laid out and without scratches or other undesirable embellishments. The magnetic sensors if properly aligned can retain their configuration for years.

I also checked into Honeywell’s single-chip magnetic sensing unit. The main advantage of having all three axes on one chip is a greater sensitivity to surrounding magnetic fields. The combination of all three sensors in an orthogonal configuration allows sensing down to 85 micro-gauss, which is more than twice as sensitive as the two
chip package. The applications of this chip include uses in navigation systems, attitude reference, traffic detection, and medical devices. The HMC1023® sensor has three wheatstone bridge circuits that in combination measure magnetic fields for both field strength and direction. According to the device datasheet, “the sensor’s elements convert any incident magnetic field in each element’s sensitive axis direction to a differential voltage output.” (Honeywell HMC1023) Like its two chip counterpart, this device has two magnetically coupled strips. The offset strap is a “spiral of metallization” that couples each sensor element’s sensitive axis. In this design, there is one strap per bridge. Each offset strap measures nominally 50 ohms. The straps easily handle currents to “buck or boost” fields through the +/- 6 gauss measurement range. In most applications using this sensor device, the offset strap is not used. The set/reset (S/R) strap is used to realign the magnetic fields. [14] A major difference in this configuration is that the S/R strap couples the sensor elements along the easy axis, which is perpendicular to the sensitive axis. There are three straps with one for each sensor that are paralleled together, which allows for operation at low voltages. Unlike the offset straps, the S/R straps are not optional. They must be used to periodically condition the magnetic domains of the magneto-resistive (MR) elements for the most accurate and efficient performance. The noise density for this configuration is approximately 50 nV per Hz\(^{1/2}\). In a last note, in situations where power is at a premium, the sensor bridge and support electronics can be switched off between magnetic field measurements. In my construction of a calibration routine, either chip package would work equally well. Overall, I am not as concerned about the accuracy of the magnetic sensors, as I am about the acceleration sensors. The magnetic sensors if properly aligned can retain their configuration for years.
1.5 Gyroscopes

Although not found in the 3DM-M®, gyroscopes greatly increase the accuracy of inertial measurement devices. For example, the 3DM-GX1® has a temperature compensated gyroscope that combines the outputs from three angular rate gyro (gyroscopes) with three orthogonal DC accelerometers and three orthogonal magnetometers. [9] According to an article on machinedesign.com, “the embedded microprocessor [on the 3DM-GX1®] contains a unique programmable, complimentary filtering algorithm, which blends the static and dynamic outputs of the sensors to provide stabilized pitch, roll, and yaw measurements under dynamic and static conditions.” The sensor provides an orientation matrix to measure over a range of 360 degrees. The gyroscopes used in the 3DM® devices are vibrating gyros that utilize micro electronic mechanical system (MEMS) technology and usually, a vibrating, quartz tuning fork to measure the force. [1] When rotated, the element in this device is subjected to the Coriolis Effect, which causes secondary vibration orthogonal to the original vibrating direction. Using this secondary vibration, the gyroscope can detect the ‘rate of turn’.

The main reason for having the gyroscopes in the first place is to measure angular rate or orientation about a given directional vector. The gyroscope on the 3DM-GX1® uses a multi-axis gyro that provides measurements in three orthogonal directions. [1] The two most common, basic gyroscopes in use are rate and rate integrating. Rate gyroscopes are classified as single degree-of-freedom (SDOF) devices. These devices use an elastic restraint of the spin axis about the output axis. Rate-integrating gyros are also SDOF devices but instead use a primarily viscous restraint of the spin axis about the output axis. [11] Most gyroscopes in use today are rate gyros. Another category of gyroscopes,
optical gyros, use the repeated reflection of a laser ray inside the enclosure. The two main types of optical gyroscopes are ring laser gyros (RLF) and fiber optic gyros (FOG). RLF gyros achieve laser reflection by the use of mirrors inside the enclosure. FOG devices use a coil of optical fiber to reflect the laser inside the enclosure. Spinning mass gyros (SMG) use a steadily moving mass with a free moving axis, known as a gimbal. When a gyroscope of this type is tilted, motion along the angle of the rotating mass indicates that the angle has moved.

Gyroscopes come in three axis types; uniaxial, biaxial, and triaxial. [11] Uniaxial gyros measure the angular rate around a single axis, while biaxial gyros measure the rate around two orthogonal axes. The 3DM-GX1® uses a three axis gyroscope that measures the rate around three orthogonal axes. When speaking about the angular rate, I am actually referring to three separate properties—angular rate change, linearity, and bandwidth. Angular rate range is the maximum rotary rate. Angular linearity, which is also known as rotary axis linearity, is measured over an operating temperature range, as a percentage of the full scale. (GlobalSpec, globalspec.com) Lastly, angular bandwidth is the frequency range over which the gyroscope meets its accuracy specifications before the occurrence of ‘roll off’. Usually only the high frequency roll of point at 3db is included, because most gyroscopes are capable of a DC response.

1.6 MEMS

The 3DM® device uses a revolutionary new technology known as micro electronic mechanical system (MEMS). The MEMS system is comprised of a micro
inertial measurement system (MIMS). MEMS refers to a type of system that makes use of small, three dimensional structures with devices that require areas as small as micrometers. A more specific form of this technology that is relevant to inertial measurement systems is known as a micro inertial measurement system (MIMS). [5] “A MIMS system integrates measurement, sampling, quantization, coding/compression, and storage.” (Dong and Zhang, 16) A MIMS system has many applications in the measurement of motion in rockets, missiles, and vehicles. In using a MIMS system, the primary areas of consideration when choosing the right system for the application include volume, cost, and reliability. One of the best results of using this type of technology is the reduction in scale factor. Other advantages include, but are not limited to high reliability and level of integration and production volume. One such important observation is that the data accumulated is stored in RAM in real-time, and can later be stored on a computer system. [5] The MIMS used in their analysis used micro-machined silicon transducers, an ASIC with self-adapting collection and storage control functions, and power control. “System modeling and analysis showed that the transducer and memory determine the size of the entire measurement system, especially when the data size is very large.” (Dong and Zhang, 18) In a MIMS system, the accelerometers measure force in G’s through piezoresistance. In cases with extremely large data size or very large accelerations, no material exists for data acquisition that can sustain this force. A means still in testing uses a special buffering technology known as “explosion cushioning.”

The 3DM-M® uses MEMS based capacitive DC accelerometers to measure the acceleration in the x, y, and z directions (triaxial). Accelerometers are commonly used in
combination with gyroscopes in inertial measurement systems, such as the later 3DM devices, like the 3DM-GX1® and 3DM-GX2®. [1] In my case, it is used to track changes in position based on the rotation of the device over 360 degrees over all three axis. In searching for a quality MEMS system, one should look at the amplitude range, frequency range, and examine the conditions in which the accelerometer would be used. The main values important to my project would be those of the amplitude. From the amplitude, I get the magnitude and direction of the acceleration force in g’s. Accelerometers that output analog data typically do so by transmitting the voltage, current, or frequency. The 3DM outputs the data as a digitized representation of raw voltage coming of the device’s A/D converter. [2]
Chapter 2: The 3DM-M Device

2.1 Overview

The 3DM-M® is Microstrain®'s first device in the area of inclinometers and other inertial measurement systems. It is now out of production, but has some novel features that made it quite useful in its day. The 3DM® device uses an orthogonal array of magnetometers (magnetic sensors), and accelerometers (acceleration sensors) to measure the roll, pitch, and yaw angles of inertial movement. [1] The device is capable of ±180° of yaw and pitch, and ±70° of roll. These three angles provide a compass heading with three degrees of freedom that can be represented graphically on a three dimensional plot. The yaw measurement is calculated by using the Earth’s magnetic field. The measurement compensates for tilt errors using the accelerometers. (3DM User Manual, 3) Any errors in the yaw output due to the roll and pitch of the device are handled internally using embedded hardware algorithms. The three axis orientation sensor that comprises this device can be programmed to output either raw data from the sensors or angles of roll, pitch, and yaw that are generated in either degrees or as ranges of bits. In order to obtain accurate data, care must be taken when using the device. The earth’s magnetic field is easily distorted by ferromagnetic objects. [10] These erroneous results can be avoided by placing the device in or around objects that do not affect earth’s magnetic field, such as aluminum, brass, or plastics. The device I am using is encased in a plastic box. Therefore, the device should operate properly. However, to be certain, in addition to the calibration of the accelerometers, the magnetometers should also be tested.
for accuracy. Microstrain® has already designed calibration and configuration routines for both the accelerometers and magnetometers. I used a mix of Microstrain®'s methods along with my own code to enable the users in this case the students, the ability to test for and ensure the accuracy of the data they retrieve from the device.

The input and output to and from the device is transmitted serially using the RS-232 standard for single units, or the RS-485 standard, when multiple devices are linked together on a common bus. I only needed one device. Therefore, I used only the RS-232 standard. The device is externally powered with a 9 volt AC adapter. It communicates with the computer using RS232 protocols via a DB9 serial interface cable. If the computer running the software does not have a native com port, that is a port directly or indirectly connected to the motherboard, a serial to USB® adapter will work. However, there are some errors in using a USB® adapter. These errors are mainly caused by the rate of transfer between RS232 and USB®. If you are using a native com port, the device will be on com one. If you are using a serial to USB® adapter, you will have to check the Windows Device Manager to determine the port the device is using. Due to the nature of this type of device, you must add the hardware using the Add Hardware Wizard. The device is not plug-in-play compatible. When you add the hardware to the system, Windows will notify you as to which port it chose to place the device.

2.2 System Requirements

The computer the 3DM® is attached to must be running the Microsoft Windows 2000® or Microsoft Windows XP® operating systems. Also, the computer must have a
DB9 serial port to allow the device to communicate with the computer. A DB9 to USB® adapter will also work. The device is externally powered by a standard 6V DC adapter. Since technical support and parts are no longer available for the device, it becomes very difficult and sometimes problematic to create or modify the software or hardware of the 3DM®. Despite these limitations, the device can still be of use to students taking a course that utilizes the 3DM®. In utilizing a device such as the one I am using, care must be taken to assure proper functionality of the software. If this care is not taken, the device may not properly function, and could even be damaged. I have found some technical details that must be addressed in relation to the 3DM® device and my software calibration program. One such major issue that I have already discovered is that the device’s magnetometers might need to be recalibrated prior to the calibration of the accelerometers due to affects on the device from ferromagnetic objects.

### 2.3 3DM Family of Devices

The 3DM-DH® is the micro-miniature counterpart to the 3DM-M®. [1] It has the same sensor configuration, which uses three accelerometers, three magnetometers, and a user programmable adaptive infinite impulse response filter. The weighted moving average used in both this and the 3DM-M® device provide more accurate data by giving more weight to newer samples. The only downfall of this method is that errors in measurements are propagated through future calculations; especially in using the Euler integration technique. [1] The 3DM-DH®’s sensor range is equivalent to its larger counter part. The 3DM-DH® can also measure +/- 180 degrees of yaw and pitch, and +/-
70 degrees of roll. It also uses orthogonal arrays of accelerometers and magnetometers in order to measure the three angles over a wide angular range. The 3DM-DH® device still uses a serial port that returns analog data; which can return values as raw magnetic field and accelerometer outputs, or computed angles of pitch, roll, and yaw. The 3DM-DH® can also be used alone or in programmable arrays in a daisy chain configuration using RS-485 multi-drop protocols. Applications of the 3DM-DH® include but are not limited to navigation, bio-medical engineering, and robotics.

The 3DM-G® is different from the 3DM-M® and 3DM-DH® in that it uses a three axis gyroscope. The output can be computed using Euler angles, Quaternions, or by a 3x3 rotation matrix that is also called a coordinate transformation matrix. A downside to the 3DM-G® device is that it is limited to +/- 90 degrees pitch due to a mathematical singularity produced using Euler angles. [1] The 3DM-G® also uses RS-232 and RS-485 in their respective applications. RS-232 is the most reliable and convenient method for serial communication, when using a single device of this type. The RS-485 mode allows communication of multiple data on the same data bus. The half-duplex configuration used in this mode does require additional structuring of the host’s computer software since multiple 3DM®’s can not communicate with each other at the same time. It is not possible for the devices to both transmit and receive data. Data transmission by the 3DM-G® is controlled by one or more single byte commands. The 3DM-G®’s on-board processor operates with a 0.0065536 second clock tick interval. (Microstrain®, www.microstrain.com) According to Microstrain®, “the processor continuously reads the raw sensor inputs, scales them into physical units, performs gyro-stabilization, and (if
requested by a user issued command) generates an estimate of its orientation. This device uses its own internal clock to read, scale, and stabilize sensor outputs.

The 3DM-G® like the 3DM-M® and 3DM-DH® can output data continuously (continuous mode) or by user request (polled mode). In continuous mode, the data arrives at the host computer in a continuous stream at the maximum possible rate at uniformly spaced time intervals. Therefore, the host computer must be able to properly buffer and interpret the data. There are no gaps in data transmission in continuous mode. The same rules apply to the 3DM-M®, which I am using for this project. As with the 3DM-M, this device always powers-up in polled mode. If the user desires it to start-up in continuous mode; the user must set the desired value on the EEPROM. Unlike the 3DM-M®, the 3DM-G® can operate in a combination of polled and continuous mode. It does this by interleaving the continuous commands with the polled commands. At the completion of a calculation cycle, the 3DM-G® transmits the response to continuous mode commands followed by the responses to polled commands. The two packets are transmitted during the same calculation cycle. The calculation of the Euler angles by the 3DM-G is virtually identical to the methods used by the 3DM-M®. The gyroscope does not play a role in the calculation of the role, pitch, and yaw that is determined in this device. The only values used in the calculation are those from the accelerometer and magnetometer sensors.

The orientation scheme in both devices is with respect to the fixed earth by the “ZYX” or aircraft coordinate system. [1] In this system, X points north, y points east, and Z points down in the direction of earth’s gravity. Both devices are exposed to the same linear accelerations that produce magnetic interference and sensor artifacts. The
roll and yaw angles have a range of -32768 bits to +32767 bits, which represents -180 degrees to +180 degrees; and a pitch with range -16384 bits to +16383 bits that represents an angular measurement of between +/- 90 degrees. In order to obtain angles in degrees, when doing calculations, the user must multiply by a factor of 360 divided by 65536.

The 3DM-G® can output in one of the following modes: RawMag, RawAccel, RawAngRate, MagField, Accel, AngRate, StabMagField, StabAccel, and CompAngRate. (Microstrain, www.microstrain.com) RawMag transmits the three raw voltages of the three axis magnetometer in terms of A/D converter codes. RawAccel returns the raw output voltages for the 3-axis accelerometer. RawAngRate returns the raw voltage outputs from the three axis rate gyroscope. According to Microstrain® documentation, MagField sends back a vector quantifying the direction and magnitude of the instantaneously measured magnetic field that the 3DM® is exposed to during operation. Accel returns a vector, as well, that quantifies the direction and magnitude of the instantaneously measured acceleration values. The values returned by AngRate are in the form of a vector that quantifies the rate of rotation of the 3DM-G®. StabMagField is the gyroscopically stabilized analog of the Accel vector. Finally, CompAngRate is the bias compensated analog representation of the AngRate vector. The means and methods of the 3DM-G® are so important to my project due to the lack of documentation for the 3DM-M® due to its age. The 3DM-G® operates very similar with respect to the magnetometers and accelerometers, as the 3DM-M®. The extra functionality explained for the 3DM-G® though in applicable to my device provides not only an understanding of an alternative inertial measurement system, but also clarifies information on certain topics not present in the 3DM-M® documentation. In other words, using the
documentation for the 3DM-G®, I am able to work backwards to the functionality of the 3DM-M®.

The 3DM-GX1® is the second generation of the 3DM-G®. It combines three angular rate gyros with three orthogonal magnetometers, a multiplexer, 16 bit A/D converter, and embedded microcontroller, to output its orientation in dynamic and static environments. (Microstrain, www.microstrain.com) The device is capable of on-board processing and filtering of accelerometer, gyroscope, and magnetometer outputs. The device operates in a 360 degree range on all axes. The orientation information can be read out using Euler angles, quaternion, and matrix formats. The output is in digital serial; and is fully temperature compensated. As in the other devices, a network of 3DM-GX1®’s can be created to aid orientation calculations. There are several upgrades found in this device versus the one I am using. The most noticeable differences are the gyroscope and embedded microcontroller. The applications are quite similar; they include navigation, biomedical applications, and robotic devices. The calibration system for the 3DM-GX1® is quite unique to Microstrain®’s other devices. The routine for this device performs a robotic calibration of all modules, software correction for sensor misalignment, software compensation for bias and sensitivity for all nine sensors over the full temperature range, software correction for the gyroscopes, and a hard and soft iron field calibration. [1]

The main functionality of the 3DM-GX1® is its utilization of triaxial gyros to track dynamic orientation. It also uses triaxial DC accelerometers and triaxial magnetometers to track static orientation. A unique aspect of the 3DM-GX1® is that it allows the user to optimize the data output rate by adjusting the number of data quantities
delivered per second. According to Microstrain®, the 3DM-GX1® device provides a fast response even in a system with vibration and quick movements, while eliminating drift. The stabilized output is provided to the user by way of an easy to use digital format. The device can also output in analog for the Euler angles if needed by the user. Further, the device has wireless networking capability using the Bluetooth standard. The wireless connectivity provides less cable hassles and greater data security.

The 3DM-GX2® is a gyroscope enhanced inertial measurement system that utilizes miniature MEMS sensor technology. The device uses four different types of sensors. The device uses triaxial accelerometers, triaxial gyroscopes, triaxial magnetometers, and temperature sensors. The 3DM-GX2® outputs data by several means, which include calibrated inertial measurements of acceleration, angular rate, and magnetic field or deltaAngle and deltaVelocity vectors and computed orientation estimates of pitch and roll or rotation matrix. [1] The angular rate quantities are corrected for G-sensitivity and scale factor non-linearity. All measurements are also corrected for temperature errors and sensor misalignment. The device has several options for communications with the host computer. The 3DM-GX2® can send data using RS-232, RS-422, USB 2.0®, and wireless transceiver that uses standard 802.15.4. However, no matter what communication standard is used, the host computer always views the incoming data as serial communication. The 3DM-GX2® is an improvement over earlier 3DM’s with the addition of six independent A/D converters, which allow all sensors to be sampled simultaneously with the best possible time integration. Not unlike the 3DM-GX1®, the 3DM-GX2® allows for polled, continuous, or a combination of polled and
continuous communication of commands. The applications involve a wide variety of inertial navigation systems, biomedical animation, and platform stabilization.

The 3DM Inertia-Link® sensor measurement unit utilizes high performance MEMS technology with a triaxial accelerometer, triaxial gyroscope, temperature sensor, and on-board processor that combines and interprets all the data from all of the sensors. It is calibrated for sensor misalignment, gyro G-sensitivity, and gyro-scale factor non-linearity. The outputs can be in Euler angles, rotation matrix, deltaAngle and deltaVelocity, acceleration, and angular rate vectors. [1] The communications interface hardware is contained in a separate module to allow for easy user customization. There are currently modules available for a wireless transceiver, USB 2.0®, RS-232, and RS-422. It is ideal for robotic, unmanned navigation, platform stabilization, and INS and GPS location tracking. The device samples data simultaneously, which allows for improved time integration performance. The system architecture was carefully designed to avoid common errors, such as hysteresis induced by temperature changes and sensitivity to supply voltage variations. The major difference between this and earlier 3DM® devices is the lack of magnetometers. The lack of magnetic sensors is compensated for by the angular rate gyroscopes, temperature sensors, and independent A/D converters that allow for simultaneous data sampling.

The 3DM FAS-A® is a one or two axis inclinometer that is best suited for heavy equipment applications, such as leveling, drilling, and container handling. Nevertheless, it is part of the 3DM® inertial measurement family. Unlike the other 3DM® devices, it only has two DC accelerometers. The two accelerometers can be used separately or together for varying applications. [1] If one accelerometer is used, the device has a 360
degree range of measurement. If both accelerometers are used, it has a range of operation of +/- 70 degrees. As with the 3DM-M® device that I am using, the FAS-A® has a user programmable infinite impulse response filter. It averages $2^x$ data points with a successive weighted average that favors newer data. Not only does this device include the standard A/D converter, it has a D/A converter that provides an analog voltage linearly proportional to the inclination with the analog output proportional to angles.

The 3DM FAS-G® includes the functionality of the FAS-A® along with an angular rate gyroscope. The on-board microcontroller utilizes a 12 bit A/D converter along with a D/A converter to according to Microstrain® provide an analog voltage linearly proportional to inclination in dynamic, as well as static environments. The device operates over the full 360 degree range of angular motion. The 3DM FAS-G® is much more versatile than the 3DM FAS-A® not only because of the addition of the gyroscope, but also due to the additional digital serial output capability. [1] The gyroscope tracks dynamic angular position, while the two DC accelerometers track static angular position. It can be programmed to output compensated angles or raw sensor data. The raw sensor data output feature is a key part of the use of the original 3DM® device in my design. It allows me to utilize the data from the device in the most convenient manner.

2.4 Data Acquisition

The 3DM-M® can sample in two modes. In continuous mode, data can be retrieved in angles or bits mode. In polled mode, the device returns various data on
requested based on a series of commands. While in polled mode, the device will respond only when it receives a command to transmit data. In working with the device, I have determined that the continuous sampling mode works the best. In continuous mode, the device is constantly sending data until the buffer reaches its limit. The buffer size can be increased. The RS-232 asynchronous communication parameters are 9600 baud, no parity, eight data bits, and one stop bit. The data is returned to the computer in packets once the communication has been synchronized at both ends.

2.5 Microstrain® Software Versions

2.5.1 Overview

The 3DM-M® device is compatible with three operating systems—Microsoft DOS, Microsoft Windows 2000®, and Microsoft Windows XP®. Further, it is compatible with three Microstrain® software versions: 3.0.0, 4.0.3, and 4.0.6. Version 3.0.0 is an MS DOS® program, which can run on Windows 2000® or Windows XP® using a DOS® emulator. A DOS® emulator is built into the Windows XP® operating system. The DOS® program used in conjunction with my calibration program was compiled by Pedro Claudio of Lockheed Martin. The device software made by Microstrain® has been tested on Windows 2000® and Windows XP®. The device was not tested on the new Microsoft Windows Vista®, or any other operating system. In the construction of the user-interactive calibration program for UCF, Microstrain®’s version 4.0.6 software was used operating on Microsoft Windows XP®.
2.5.2 MS DOS® Software

The Microsoft DOS® program for the 3DM® was used in part to verify the data obtained through the use of Microstrain®’s version 4.0.6. The DOS® program is designed to mimic Microstrain®’s version 3.0.0 MS Windows® software. The main functions of the program are to compute the acceleration and magnetic sensors values, and the angle measurements of pitch, roll, and yaw. More specifically, option one continuously outputs the pitch, roll, and yaw angles. Option two continuously outputs the raw bit values for the three accelerometer and three magnetometer readings. The output is given as ax, ay, az, mx, my, and mz; where ‘a’ represents the acceleration sensors and ‘m’ represents the magnetic sensors. The third option uses the polled mode of the device. It also outputs the accelerometer data, magnetometer data, and angles. The last measurement mode outputs the three angles in bit representation. There are two other options that allow the user to adjust the “PC Parameters” and the sensor configuration parameters. The first allows configuration of the gains, offsets, and filter parameters. There is also an option provided to restore the factory defaults for these values. The PC parameters option lets the user toggle between serial communication ports one and two.

2.5.3 MS Windows® Software

I used Microstrain®’s version 4.0.6 software on both MS Windows 2000® and MS Windows XP® in the development of the calibration program. For most of the product development, I used the version 4.0.3 software on MS Windows XP®. The 4.0.3 version is identical to the 4.0.6 software with the exception that the 4.0.6 software allows
you to select any serial com port. The 4.0.3 software limits you to com ports one through four. The reason this is an issue has to do with the fact that I was using a USB® to serial adapter. Microsoft Windows® places what are called USB® virtual serial ports on com port numbers five or higher. Hence the 4.0.6 software became the best option. In later development, I switch over to a PC running MS Windows 2000® with a native com port. Using this other machine, I no longer needed the adapter. The elimination of the adapter dramatically increased the performance of the 3DM® software. More specifically, it reduced errors in sampling data. The current version of the software that Microstrain® ships with its 3DM-M® and 3DM-G® devices is version 5.0.3. However, this software is incompatible with the 3DM-M®. As mentioned, due to communication problems I wound up using a computer with a native com port. Since most of my development was done with software version 4.0.6, I continued to use that software version even though I only needed com port one.

Microstrain® software version 4.0.3 and 4.0.6 both contain the following functionality. The software versions both launch with a main screen depicting a 3DM-M® device, a file, and help menu. The following are the options under the help menu: About—gives general program information; Help—directs the user to the separate help documentation. The help file is not part of the software program. In order to start collecting data, the user must choose the com port to which the device is attached. Under the file menu, select Comm. Port, and a box appears that lets the user select the desired com port. When using a computer with a native com port, the operating system will always default to com one—unless your computer has multiple serial ports. If you are using a USB® to serial converter or a USB® docking station serial port, the port is
randomly selected by the operating system. Therefore, when using a USB® adapter, the user must check the Device Manager® to find out what port the operating system selected. A warning is needed if a USB® adapter is used. Due to the fast speeds at which USB® runs, there is a potential conflict with the older, slower transmission RS232 protocols. In constructing the calibration program, I decided not to use a USB® adapter—primarily due to this problem. The device has three sampling modes: continuous bits, continuous angles, and polled. [3] Due to an irresolvable problem with Microstrain®’s software, the program can only run on continuous bits mode. There is a menu that allows users to change the sampling mode. However, doing so crashes the program. After selecting the com port, the device is initialized. However, due to another error in the device’s software, the user must under the file menu un-check, and then re-check the Initialize Connection menu item in order to collect data.

After you initialize the connection, two other menus are enabled—display and tools. If the connection was established properly, there is a check mark next to Comm. port and Initialize Connection under the File menu. Under the display menu, the user can choose Dials(angles), Gauges(bits), Graph(angles or bits), and Instruments (angles). Under the tools menu, you can choose to calibrate or configure the device. ‘Angles’ returns measurements in degrees. ‘Bits’ returns data in digitized values that require a conversion formula to use the data values. [3] In choosing the Dials(angles) display, you are shown three dials(0 to 360 degrees pitch, 0 to 360 degrees yaw, and -70 to 70 degrees roll. When in dials display mode, you can tare (zero) the values—sampling must be in progress to tare the sensors. Under the data menu, you can choose to save the current
values, sample, or return to the previous screen. When you click sample, the needles start bouncing around.

In order to save the data from a sampling session, you must first click save under the data menu prior to clicking sample. After you have the data you want just click sample again to stop the sampling process. Now you can open the file containing the sampled data in notepad or excel. The data is stored in a comma separated values file with the extension .csv. The file contains the date the file was created, the time the file was created, and four columns containing from left to right (time, pitch, yaw, and roll). Time is measured in seconds. The file is most easily viewed in Microsoft Excel®. The .csv can and should be saved as a .xls file in order to work best with Matlab.

Under the gauges display, you can see the raw bit values for the accelerometers (x, y, and z axis) and the magnetometers (x, y, and z axis). Under the graph display, you
are shown an x/y graph of angles vs. seconds, a numerical display of the pitch, roll, and yaw angles, and a numerical display of the x, y, and z axis bit values for both the accelerometers and magnetometers. In graph mode, you can create data files for the angles or bits values, but you cannot save the displayed graph. It is only for real-time viewing. However, you can save the data values to a .csv file, and create a graph using Microsoft Excel®, or a similar spreadsheet program. The angle and bits values on the graph produced by the 3DM® software are color coded to match the tabulated values on the left side of the screen.

Using the instruments display, you are shown the heading (yaw) on a compass rose, the angles in text boxes on the left side of the screen, and the attitude (pitch and roll) in an airplane cockpit view in the middle of the screen. In the data file, the measurements are displayed in degrees. As in all of the other data files created using this mode, the table lists the pitch, yaw, and roll for the duration of the sampling time in seconds.

Under normal operation, there are two other sub-menus under the tools menu on the main screen. They allow the user to calibrate the device and configure device parameters. Under the configure menu, you have the option to set the values for the digital filter (magnetometers ad accelerometers), orientation mode (swap axis), and tare (pitch tare, yaw tare, roll tare). On this screen, you can read the values currently stored on the device and/or write new values to the device; this is done under the tools menu by choosing either read or write. On the calibrate screen, you can read or write the values for the current bit of the accelerometers or magnetometers(x, y, and z axis), and set the highest and lowest bit values for the (x, y, and z) axis of the magnetometers and accelerometers. The user can also set the gain and offset values for the (x, y, and z) axis.
of the accelerometers and magnetometers. The option is also given to individually select
by check boxes for reading from or writing to the device.

![Config Sensor]

**Figure 3: 3DM-M Configuration Screen**
Courtesy of Microstrain, Inc. (www.microstrain.com)

The software can also operate in a special factory mode. In order to run the
device in this mode, a ‘-device’ switch must be added to the target path of the program.
The target path gives the location, where the program executable is stored on the
computer. If the target path were c:\program files\3dm\3dm.exe, the path would have to
be changed to c:\program files\3dm\3dm.exe -factory. The main changes are in the Tools
menu on the main screen. The changes occur in the configuration menu, and a new menu
referred to as ‘linearization’. I was unable to use the device linearization program due to
an error in the 3DM software. If the program worked, it would allow you to view the
deviation of the pitch and yaw angles from true angles. ‘True angles’ is just a fancy term
for angles from zero to 360 degrees. The 3DM® plots the measured values against the
‘true’ values to determine the deviation percentage. [4] If this program worked, it would
have allowed me to determine better the accuracy of the measurements. It was not, however, critical to the creation of my calibration program. The main addition to the configuration program created by Microstrain® is the ability to view and adjust the gain and offset values for the accelerometer and magnetometer data along the x, y, and z axes. There are also thirteen other settings that the user can read from or write to the device. The first six settings adjust various parameters of the device filters. They are the static and dynamic filter coefficients, filter sensitivity, and on/off setting for the adaptive filter. The x, y, and z values from the digital to analog converter (DAC) can be read. Lastly, the user can turn on/off angles bits mode, analog mode, yaw look-up, and pitch look-up.

2.6 Device Communication Protocols

The 3DM-M® device primarily uses the RS232 communication standard. When dealing with more than one 3DM® in a closed loop network, the RS485 standard is used to communicate with the devices. [2] For the purpose of writing a calibration program, only one device is needed. Therefore, this project only uses the RS232 standard. The device is designed to operate in either continuous or polled sampling methods. It is highly recommended by the manufacturer to use the polled sampling method. In tests done by Microstrain®, data received through polled sampling was of much higher accuracy than that received through continuous sampling. [4] Unfortunately, the 3DM® software for an unknown reason will not operate in polled mode.

In sampling data, I had to use the continuous sampling method. I was able to retrieve data as either angles or digitized representations of the accelerometer or magnetometer data in bits mode. An analog to digital converter is used to retrieve the
data from the device. Using continuous mode, the data is returned in predefined packets.

In using the continuous sampling mode, the device will be sending data packets to the computer on a non-stop basis. In order to retrieve and send data in this manner requires that the device and the computer be synchronized in order to prevent loss of data. The first continuous sampling mode, which is known as continuous angles mode sends the most and least significant bits of the computed values for roll, pitch, and yaw. These are the accelerometer readings for the x, y, and z axis, respectively. These angles are transmitted in degrees by sending the ASCII character ‘w’ (77h) to the device. The other mode known as continuous bits mode sends a thirteen byte packet containing the most significant bits (MSB) and least significant bits (LSB) of both of the three acceleration sensors (accelerometers) and the three magnetic sensors (magnetometers). When sending the ASCII character ‘u’ (75h), this data is received in order of x, y, and z axis of the magnetometers followed by the accelerometers. The first byte in the packet is a constant value of 255, as is the first byte in the packet for the continuous angles mode. In continuous mode, the data will continue to be sent until a different character is transmitted.

Even though I only used the continuous sampling mode, it is worthwhile to explain the polled sampling mode. The device is placed in polled sampling mode by sending the ASCII character ‘t’ (74h). The device will remain in this mode until a different character is received by the device. While in polled mode, the device will only respond when it receives one of seven commands, and its’ response will be a fixed number of characters. The broadcast command is the only exception. The broadcast mode is only used when multiple devices are connected via a common bus. The
command format is a one byte packet made up of seven bits. The highest bit is always one. Bits six through four are used to select the desired command. The lower four bits contain the command destination address. The command data packet is a zero followed by seven bits of data. When the data is received, a packet is sent back to notify the sender of whether the data arrived correctly or not. If the data was received correctly, a packet is sent back with upper bits 01000 followed by a three bit response message. If there was an error, from high to low, the packet contains 0110 followed by a four bit error message.

The device can send data back in one of four formats. If the device receives the command one (001), the device returns both the magnetometer and accelerometer data. If the data is received correctly, the value 41h is returned. If there is a problem with the data, the device reports an error by responding with 6xh, where ‘x’ is an error code. The data returned when this command is received is twelve bytes consisting of the MSB followed by the LSB for the x, y, and z axis, respectively of the magnetometer followed by the accelerometer data. In order to receive only the magnetometer data, the command two (010) should be sent to the device. The device will return six bytes of data MSB followed by LSB in the order x followed by y followed by z axis. The caller will receive the response 42h if there were no problems and 6xh if there was an error. The accelerometer data is retrieved by sending the command three (011). The data is sent in the same order as for the magnetometers. The only difference is the response from the device is 43h followed by six bytes of data. Again, if there is an error, 6xh is returned. In order to simply return the roll, pitch, and yaw measurements send the command four (100). The device will return the MSB followed by the LSB for the roll, pitch, and yaw
respectively. The response from the device to denote this action is 44h for data or 6xh if there was an error. The commands five (101) and six (110) are not used, and command zero (000) is reserved. All data returned by the device is calculated in sixteen bit words by multiplying the MSB by 256 and then adding to it the LSB. \[2\] In order to get the response in radians, divide the sixteen bit word by 10430.

In order to read data from the device, such as the configuration data, bytes one and two of the command data must be 66h and the address were the data is stored. The device will respond with the value at the specified memory location by returning two bytes of data MSB followed by LSB. The data stored in the EEPROM is non-volatile. When writing to the device, six bytes of command data are sent. The first two bytes are 65h and 71h, respectively. The third byte contains the address were the data will be written. The fourth and fifth bytes contain the data to be written MSB followed by LSB. The last byte sent is AAh. The device responds with the programmed value. The value is the MSB of the data followed by the LSB of the data.

In my software I used continuous bits mode, because I need a real time or almost real time data stream with which the students can track the incoming data. Real time data is important, because unlike with static data files, the data is always changing. However, due to problems inherent to the device, I had to create data files in case the device has difficulty. In my decision between using continuous angles or continuous bits mode, the primary deciding factor for me had to do with the means of calibration. I could have written a program that used angles or velocity data to determine if the device is calibrated. The advantage of continuous angles mode is the calibration process is much more visually understandable. For example, in one of the 3DM's software modes, the
data is displayed on gauges that range from zero to 360 degrees. Even the untrained eye can tell very quickly if the physical movement of the device matches what is represented on the gauges. In order to use the data, the computer has to know when to start and stop reading the data in order to operate on that data. There are several methods for determining when the wanted amount of data has been retrieved; among them are using a timer, measuring packet length or tracking the data byte by byte as it comes in looking for the value 255d to determine where a packet starts. As I parsed through the data, I collected the data values in between the 255d packet header. The data packets do not have a stop bit, therefore, whenever you see the value 255d it is the start of a new packet of data. The six bytes after that number will always represent the data values. In order to get the actual acceleration value, for example, you have to apply the device output conversion formula as stated in equation three on page twelve. The offset and gain can be determined by querying the device. However, due to technical difficulties, I had to retrieve the data by running Microstrain®'s software. The technical difficulties will be elaborated upon in discussion of the calibration routines.
Chapter 3: Microstrain’s Calibration Program

3.1 Configuration Routine

The 3DM-M® device configuration program applies to both the accelerometer and magnetometer data. The 3DM®’s configuration file allows users to adjust the sampling rate and filter style via GUI window. In order to use the software by Microstrain®, you must first configure the device settings. If you want to zero all settings, you can ‘tare’ the device’s internal sensors. The user can adjust the settings for filtering, tares, orientation, and the gain and offset for each axis of both the accelerometers and the magnetometers. The first step in the configuration is to check the current configuration values displayed on the computer screen. If you want to modify any item on the configuration, you must use the write function, which will send all fields from the configuration screen to the device. If the device does not output as expected, then the user should run a calibration program.

Although not usually necessary, the user can alter some of the device’s internal parameters. In order to write data to the device outside of Microstrain®’s software, you must issue a six byte command data packet to the device. [2] The first and second bytes are always 65h and 71h, respectively. The third byte contains the address in memory where you are sending the data. The fourth and fifth bytes contain the data to be written with the MSB followed by the LSB. The last byte of the packet is always AAh, when writing to the device. The user can also read back data stored on the device. It is a good idea to retrieve this data and examine it thoroughly before altering any parameters. The
read operation follows a simple two byte command data packet. When the user sends 66h followed by the address of the data they wish to retrieve, the device will send back two bytes of data in the form MSB followed by LSB. The normal axis convention for the device has the telephony connector at the top of the board. [1] In this configuration, the yaw rotates 360 degrees about z-axis, when following the standard x-y-z-axis convention. The pitch angle rotates 360 degrees about the x-axis. The roll angle ranges from zero to seventy degrees rotating about the y-axis. If the swap axis setting is set to one, the pitch and roll axis swap with one another. The result of this operation is a ninety degree rotation of the device orientation. Under the normal convention, gravity is acting downward on the z-axis.

The 3DM-M® uses a successive weighted average digital infinite impulse response (IIR) filter. The average is determined by weighting the last set of data greater than previous sets of data. The number of data points assigned to the filter within a range of zero to five sets the number of samples based on $2^x$ successive samples, where $x$ is the value of the number of data points. [3] The device can use either standard or adaptive filtering. When in adaptive filtering mode, the device uses static or dynamic filtering based on the rapidity of the movement of the device. The user can put the device into adaptive filtering mode by setting the adaptive filtering value to one. The filter type and sensitivity can also be set individually for the magnetometers and the accelerometers. The sensitivity in this case refers to how many data bits change during the sampling time. If a large change in bits occurs, the device treats the situation as if it had been rapidly moved from one position to another. In this case the 3DM® uses dynamic filtering. The user can set sensitivity values for both static and dynamic filtering. The data is set by
assigning four values, which represent static filtering for both the accelerometers and magnetometers, as well as values for dynamic filtering of both sensors. If the user sets the static and dynamic filter values the same, while adaptive filtering is enabled, it negates the adaptive filtering scheme. Adaptive filtering can improve accuracy by allowing the device to determine the rate of movement and hence the best type of filtering. [1] The samples corresponding to the number of data points sampled can range from one to fifteen.

3.2 Device Calibration

Due to various reasons, both the accelerometer and the magnetometer data should be checked prior to using the device. Normally, you just need to calibrate the accelerometers. However, if the device has come in contact, or is believed to have come in contact with any ferromagnetic objects, the magnetometers (magnetic sensors) should be recalibrated, as well. The 3DM® device is very sensitive to surrounding materials. For this reason, the device should be encased in a non-ferromagnetic material, such as plastic or anodized aluminum. The presence of metals, like steel, distorts the earth’s magnetic field with respect to the device. [1] The 3DM’s accelerometers are calibrated with respect to the magnetometers. Therefore, it is necessary to recalibrate both the accelerometers and magnetometers at the same time. If the device must be near ferromagnetic materials, recalibration will negate the effects of the metallic substrate as long as the 3DM® does not move relative to the substrate. Typically the accelerometers
should not require recalibration, because the earth’s gravitational field is the same
everywhere, and cannot be easily distorted.

Figure 4: 3DM Calibration Screen
Courtesy of Microstrain, Inc. (www.microstrain.com)

Microstrain® Inc. recommends the following procedure for recalibrating the
device. First, rotate the 3DM® in its mounting location, which allows the device to
search for minimums and maximums of the magnetic and gravitational fields from which
the calibration coefficients (offsets and gains) are determined. Before calibrating the
accelerometers, it is usually necessary to recalibrate the magnetometers. Magnetometer
calibration is a two step process. First, the device should be rotated in the horizontal
plane. The x and y magnetic coefficients are then written to the device. Next, the device
should be rotated ninety degrees; this will determine the value for the z sensor. The z
magnetic sensor should be calibrated through the same field as the x and y sensors. In the
manufacturer’s software, this is accomplished simply by selecting check boxes and
clicking the write button. When acting outside the manufacturer’s program, you must
write the gains and offsets separately to their appropriate locations on the EEPROM's memory. When rotating the device, be sure to keep the device in either the horizontal or vertical plane, as appropriate. When calibrating the x and y magnetic sensors, be sure to rotate the device through 360 degrees at least once. Before moving the device for the next step in the calibration, write the data values to the device's memory. If programming through Microstrain®'s software, only check the magnetometer x and y values check boxes. Four values will be written to the device. They are the gain, offset, and minimum and maximum range. Next, place the device on its back, rotate, and check the 'mz' check box to set the z-axis magnetic sensor value. After the new magnetometers values have been checked for accuracy, the three accelerometer sensors need to be tested for accuracy, as well.

Using the software provided by Microstrain®, the accelerometers are calibrated by rotating around all three axes through at least 360 degrees. The check boxes for 'ax', 'ay', and 'az', which represent the three accelerometer sensors, are then checked and the 'write' button is clicked. The procedure described above will in and of itself not work without the company's software. Therefore, the calibration program that I am developing will use a series of numerical integrations operating on raw accelerometer data outputted by the A/D converter in combination with rotating the device to calibrate the acceleration sensors. One option available to assist in the accelerometer calibration is the tare command, which when applied to the device's sensors resets all values to zero; and then based on that data calculates an appropriate offset for each sensor. The tare command is very useful in that it supplies values that are used to adjust the axis measurements, which allows for the accurate sampling of data. Even though the device's acceleration sensors
may be properly calibrated, the device can still be a little bit off. Using the tare command rectifies these small errors. The acceleration values are then put into a conversion formula, which gives the output of the accelerations in g’s.
Chapter 4: My Calibration Program

4.1 Overview

My software reads in values from the 3DM®'s digital to analog converter (DAC). From this data, the conversion formula listed on page twelve in equation three provided by Microstrain® is used to convert the data into accelerations measured in g’s. Bits_output is the values returned from the DAC. The acceleration values are then integrated twice using Forward Euler integration to determine the displacement values. The closer the displacement values are to zero, the more accurate the calibration. The acceleration values are also used to ensure accuracy by measuring how close the values are to the default position. The default position is where the device is at 0g for the x and y axis, and -1g for the z-axis. [1] The -1g reference for the z-axis is an industry standard for inertial measurement devices. My software also reads the internally calculated pitch, roll, and yaw angles from the 3DM®, which allows the user to quickly determine the state of the device’s sensors. The software sold with the 3DM® was used to determine the gain and offset values for each sensor. The 3DM® software was also used to create data files for four states—angles mode still, angles mode rotating, bits mode still, and bits mode rotating. The data files provide an accurate, and easy-to-use method of data analysis. The calibration of the magnetometers has been eliminated from this routine due to the lack of required data from the maker of the device. It should, however, not cause any significant aberrations in the results. The device will operate separately from the manufacturer’s software, and communicate with the computer using RS232 standard
communication protocols via a standard DB9 serial port. Communication via a USB® to Serial converter was attempted, but performed poorly due to the difference in transfer rates between the two standards.

4.2 Calibration Routine

The main purpose of the software that I created was to calibrate the 3DM® inertial measurement device. Due to device limitations, the calibration routine requires the 3DM® software to obtain data files, which are then processed by my software. The first step in my calibration routine is to determine the accuracy of the current device measurements. A good means of determining how well the device is performing is to analyze the device using data retrieved from the 3DM®, while it is in ‘angles’ mode. The ‘angles’ data is returned in degrees. The software measures pitch, roll, and yaw angles. The pitch axis corresponds to the x axis. The roll axis corresponds to the y axis. The yaw axis corresponds to the z axis. In order to get the best idea of how the device is operating using this mode is to take measurements at rest, and while the device is being rotated 360 degrees about each axis. I held the device in its default position, which is with the front of the device facing me, and the telephony connection at the top. For user reference, an image on the front of the device shows the user how to hold it. With the device in this position, I collected data for approximately 30 seconds, which gives me about 300 data values. The data is then saved to a Microsoft Excel® file, and imported into a Matlab m-file. The user of the program then observes how close the measurements are to zero. The second test is to observe the data, while the device is being rotated about
each axis. The pitch and roll angles should oscillate between zero and 360 degrees. The yaw measurements should be between approximately -70 and 70 degrees. Further, the pitch and roll graphs should contain a sine wave at some point during the measurement cycle. The yaw graph will resemble a cosine curve with the top half of the wave clipped.

![Figure 5: My GUI Layout](image)

The majority of the calibration routine focuses on the 3DM® software's 'bits' mode. Data is again collected for both 'still' and rotating states. First, data is collected and analyzed for still mode. As in the angles mode testing, plots are made for displacement and acceleration. Those plots are then analyzed to determine if the values are close to zero. The acceleration graph is observed first to see how much jitter there was in the measurements. The device is not physically moving in the test. However, there is some jitter in the sensors that registers some movement on the graphs. The
displacement plots are then analyzed to see how far the device moved away from zero. Ideally, there should be no registered movement, and the displacement would register a constant zero. Since this is not so, the device must be 'calibrated' to get those values as close as possible to zero. After the device is properly adjusted, accurate measurements can be taken, while the device is in motion.

The 3DM® has two settings that can be altered by the user—gains and offsets. The manipulation of these two values is the heart of my calibration routine. The 3DM® software comes with default values for the gains and offsets for each axis of each sensor. I am only interested in the output of the acceleration sensors. The user is given the default values, and plots the distance, velocity, and acceleration for each axis. The user is also given a plot of the values returned from the DAC. The values from the DAC represent the raw digitized output data from the device. It is based on these values that adjustments can be made to the gains and offsets in order to make the 3DM®'s measurements more accurate. Once the measurements from the three axes have been obtained along with the corresponding gains and offsets; the data can be fed into a formula provided by Microstrain® in order to convert the raw bits outputted by the device into gravitational accelerations. The acceleration is proportional to the offset, and inversely proportional to the gain. Based on the conversion formula, the higher the gain; the smaller the acceleration. As for the offsets, a smaller value is better. Overall, the closer the offsets are to the average values of the output present in the DAC graphs, the lower the accelerations. The smaller the acceleration, or the slower the device is rotated; the more accurate the measurements. I suggest in my routine that the user first adjust the offsets to ½ the average output value from the DAC for each respective axis. The
displacements, velocities, and accelerations for the three axes are then plotted again and analyzed. A noticeable improvement should have occurred in the displacement plots. In my tests, more adjustments were needed. In my routine, I then instruct the user to increase the gain for each axis by a factor of ten, and observe the results. It takes a rather large increase in gain to make noticeable improvement. Therefore, I suggest adjusting the offsets before altering the gains. The user keeps adjusting the gains and offsets until the plots show a relatively nominal displacement for each axis. At this point, the user is instructed to apply the modified gains and offsets to the data file containing measurements of rotating the device about each axis a full 360 degrees. As with the angles plots, the plots obtained in bits mode should also show a sinusoidal peak and trough for the x and y axis, and two troughs for the z axis. The plots in this case go from -1 to 1 for all three axes. It is important no matter which sampling mode the device is in to obtain measurements by rotating each axis through at least 360 degrees.

4.3 User Interface

The user interface for my 3DM® calibration program has two components. The first involves using Microstrain®'s 3DM® software version 4.0.6. The second component uses Matlab to retrieve information from the user to guide decisions made by the software. In this approach, I am using Matlab®'s GUIDE® graphical user interface (GUI) development environment. The major advantage of using GUIDE® is that you can visually see the interactions the user will have with the program. GUIDE® also auto
generates code for standard GUI operations, such as code to enable a drop down menu. It is up to the programmer to configure the generic code to fit the application.

In order to develop the user interactive graphical user interface (GUI), I used GUIDE® to construct the layout and actions of my GUI. The program comes with three templates, or you can create one of your own. Since none of the templates fit my requirements, I created one of my own. [18] The purpose of the GUI is primarily to show the user the plots of acceleration, distance, etc., as run through the various parts of the calibration routine. The GUI has been laid out such that the user can adjust the gains and offsets; and then view the effects of their changes on various plots. For example, from a drop down menu, the user can select an acceleration plot, which will display the data obtained by running Microstrain®’s software. Due to the fact that I can not directly access or write the 3DM device, the calibration program will have to be a combination of Microstrain®’s program and my own code. In analyzing the plots through the GUI, the user will be able to refine them by adjusting the gains and offsets. The user can then apply their changes to a new data set, and observe how well their changes affected the 3DM®’s output.

In more detail, my layout contained the following: a plot, six data entry boxes for the gains and offsets, an update plot button, a drop down menu, and two update buttons for the gains and offsets. The drop-down menu included plots for the following: angles mode still, angles mode rotating, DAC output, bits mode still (x-axis), bits mode still (y-axis), bits mode still (z-axis), bits mode rotating (x-axis), bits mode rotating (y-axis), and bits mode rotating (z-axis). In order to view a particular plot, the user selects it from the drop-down menu, and clicks on the Update Plot button. The software will use stored data
to construct the plot. The plots should be adjusted by altering the values in the input boxes, and clicking the Update Offsets and Update Gains buttons. However, in the event the buttons do not work, the gain and offset values can be changed in the code for the GUI. The GUI can then be launched again to view the updated plots. I am currently working on a fix for the data entry code.

4.4 Conclusions

I began this project over the past summer. At the conclusion of the summer term, I had created a proposal that laid of the shell of the work that lie ahead. I had been tasked to use the 3DM-M® device manufactured by Microstrain, Inc. to create a user-interactive calibration program for the devices' three accelerometers. In the coming months, I learned what a challenge this project would be by the time it was finished. The first major hurdle in the project came in the device itself, which I eventually found out was over seven years old, and no longer supported by the company. I attempted to get a newer device from the company, but it was at too high of a cost. Therefore, I began software development using whatever documentation I could find on the company’s website and other sites online. I decided early on based on the intended use of my work to develop the software in Matlab®. It turned out over the coming months that this was a great decision. The plethora of documentation available with the Matlab® software made code development much easier. On the other end, I found a technical support specialist at Microstrain® who has been instrumental in the success of this project.
I first wish to go into detail on the 3DM® device and its software. First and foremost, the age of the device was a deciding factor in many of my decisions. The device itself was designed to communicate with a PC using the RS232 standard via a conventional DB9 serial port interface. The device software that I was using was made to run on MS DOS® and MS Windows 98®. I discovered relatively quickly that the software was also compatible on MS Windows 2000® and MS Windows XP®. I did not test the device on any other operating systems including MS Windows Vista®.

In the beginning of the project, I was using the company’s version 4.03 software. The major complication with this software version for me was that it was coded to only operate on com ports one, two, three, and four. I was also using an MS DOS® program created by Pedro Claudio of Lockheed Martin as a test bench. The MS DOS® program, which was based on Microstrain®’s version 3.0.1 software was further restricted to com ports one or two. The major problem I had was that my laptop did not have a native com port. I then purchased a USB® to DB9 converter. However, to my dismay, MS Windows XP® placed the adapted on com port eight or higher. I also attempted using a docking port, but MS Windows XP® placed that serial port on com seven. Due to these communication issues, I switch development over to an older laptop that had a native com port. The operating system on this machine was, however, MS Windows 2000®. I proceeded with development on the MS Windows 2000® laptop, and simply changed my hardware requirements.

I initially developed the program as a simple Matlab® script file, which allowed me to experiment with Matlab®’s serial communication protocols and their interaction with the 3DM® device. I soon discovered that I could not access the device’s EEPROM,
which complicated the data retrieval and parameter process. I contacted Microstrain®, and found a very helpful customer support person, Barry Trutor, who without his help, this project would have been dead before it could really get started. Even though the device was no longer supported and documentation was scarce, he offered to examine the device and search for any documentation. I sent the device up to Vermont, and was given an updated software version and several key documentation resources. After inspecting the device, he found that it could operate on a new software version—4.0.6. The major advantage of this software version was that it allowed me to select any comport I wanted. He also provided a software help file and device user manual. Upon this windfall, I switched development back to my MS Windows XP® machine using the USB® to DB9 converter. I made this decision due to the fact that its eventual use would most probably be a part of a course. Using the newer software would eliminate two major problems for the end user. The device could now operate using a USB® port with an adapter, any comport, and the MS Windows XP® operating system. It was also beneficial to my code development in that I could use a newer version of Matlab®, and a faster processor. My old laptop was very slow, and I could not install the new Matlab software, since Matlab now sends the software out on a DVD. My laptop did not have a DVD drive. The newer Matlab version also allowed me to take advantage of certain Matlab® toolboxes that expedited my software development. I used the MS Excel® and Data Acquisition Toolboxes. The GUIDE® GUI development environment was also much improved on the newer Matlab® version—Release 2006a. In developing the data files, I ran into an interesting error, when sampling the data. I discovered the problem had to do with the speed of the data transfer. It turned out that the data transfer rate from/to the USB
connection was too fast for the RS232 communication format that the 3DM used. The error was so severe, that I had to switch back to the MS Windows 2000® computer to obtain the data files.

In code development, I also discovered two other significant problems. First, since I could not directly access the device memory, I would need to obtain data files instead of streaming data in real-time from the device. The situation was further complicated by the fact that Matlab samples data in integer time values. (Matlab, www.mathworks.com) In one second, the 3DM® device outputs about twenty-five data values. I discovered later on in development that using the Excel Link toolbox, I could read in data from the 3DM® into Matlab®, and save it in a comma separated value file (.csv). The .csv file could then be opened in MS Excel®, and be saved as a .xls file. After saving the data in standard MS Excel® format, I was then able to parse the data column by column into Matlab® variables. The other option would have been to use the comma separated value files. I chose to use the .xls files to streamline code development.

I had intended to use the serial communication protocols in Matlab to process the data from the 3DM®. Due however to the problems aforementioned, I had to use stored data files. The only need for communication with the device was then limited to creation of the data files. Another complication to software development dealt with the obtaining of certain parameters and variables from the 3DM®’s internal memory. I was unable to either read or write device parameters using Matlab®’s external interface routines. At this point it became necessary to develop the project using both my software and Microstrain®’s software. I would obtain the data and parameters from the device, and then use them in my code. Even though I could not use the serial communication
routines in Matlab®, I still developed a serial communication program to demonstrate how the software would have worked if I had been able to retrieval the needed information from the 3DM® device. I also developed a Matlab® script that demonstrates the use of data files, etc. for calibration. Lastly, I developed a Matlab® GUI file. The GUIDE® program provided code for all of the GUI features. It was up to the user to program the ‘callbacks’ that handled the details of the GUI, such as how many items were in the drop down menu and what the GUI did when one of them was selected. I had two major problems with the GUIDE® program that have hindered development. First, the routines developed for user data entry are not working. Second, the GUI figure files have a tendency to get corrupted for an unknown reason. When this occurs, development must be restarted from scratch, or put back together piece by piece. I had the corruption problem occur twice in development. The problem is so severe that it crashes the entire Matlab program. I contacted Matlab®, and they have no solution for that bug at this time. The second time the problem occurred was after I had the program 95% complete. Luckily, however, I cut and pasted the code out of the GUI m-file into notepad before it crashed Matlab. I then painstakingly rebuilt my GUI line-by-line.

In conclusion, I feel that I gained skills that I could not have obtained in any other setting. My understanding of Matlab® has been greatly increased by this project. I also learned a great deal about the construction and use of accelerometers and magnetometers in inertial measurement devices. Even though, the development was plagued with numerous obstacles some of which could not be fixed, I learned how to overcome those obstacles. I hope that this research is continued. I feel that with a newer device, such as the 3DM-GX1® or 3DM-GX2®, excellent uses at UCF can be found for this and other
software implementations. The 3DM® is both an excellent learning tool and a great device for on-going research projects in the College of Engineering and Computer Science.
APPENDIX A

Calibration Procedure
1) Plot graphs for each axis in Angles Mode (Still)
2) Displacement, velocity, and acceleration values should be almost zero
3) If not, observe how far away they are from zero degrees
4) Plot graphs for each axis in Angles Mode (Rotating)
5) Pitch and Roll should at some point go from -360 to +360 degrees
6) Yaw should at some point go from -70 to 0 and 0 to +70 degrees
7) Plot graphs for each axis in Bits Mode (Still)
8) Displacement, velocity, and acceleration values should be almost zero
9) If not, observe how far away they are from zero
10) Take note of the default values for the gains and offsets for each axis
11) If the acceleration is greater than zero the device needs adjustment
12) If acceleration is greater than zero, plot graph for DAC Output
13) Adjust offsets to \( \frac{1}{2} \) the average value of the DAC output for their respective axis
14) Plot the graph for each axis in Bits Mode (still) again
15) Observe the displacement
16) If the displacement is greater than zero, try increasing the gain by a factor of ten
17) Plot the graph for Bits Mode (still) again
18) Repeat steps 13 to 17 until displacement graphs are approximately zero
19) Plot acceleration graphs in Bits Mode (Rotating)—use the new gains & offsets
20) X- and Y-axis plots should have a sinusoidal segment that goes from -1 to 1
21) For the z-axis, the plot looks like a clipped cosine function & goes from -1 to 1
APPENDIX B

Serial Communication Test Program
% This file tests serial communication between Matlab and the 3DM-M device.  
% The initial connections are made in Matlab through Matlab Serial I/O  
% functions and External Interface routines and protocols. This program  
% was used to create the data files used with my calibration routine. I  
% could not read or write the 3DM's internal memory. However, using the  
% 3DM software I was able to create data files that mimicked real time  
% data. If I had had access to the device EEPROM, I would have used the  
% methods below to acquire the data, and calibrate the device.

s = serial('COM8'); %create serial port object linked to port COM 8
set (s,'BaudRate',9600);
set (s,'Parity','none');
set (s,'StopBits',1);
set (s,'DataBits',8);
set (s,'ReadAsyncMode','continuous');

fopen(s); % open serial connection
s.InputBufferSize = 1000; %set buffer to 1000 bytes or 1MB
s.BytesAvailableFcnMode = 'byte'; %use # bytes left as terminator
s.BytesAvailableFcnCount = 1; % stop collection when only 1 byte remains
% note BytesAvailableFcnCount must be >= 1. It can't be zero.

fwrite(s,'u'); % 'u' = continuous bits mode
data_cbits = fread(s);
fwrite(s,'w'); % 'w' = continuous angles mode
data_cangles = fread(s);
fwrite(s,'t'); % 't' = polled mode
fwrite(s,'3'); % '3' = return accelerometer data (command 3)
data_p3 = fread(s);
fclose(s); % close serial connection
delete(s); % delete serial port object

clear s;
APPENDIX C

GUI Calibration Program
function varargout = my_gui6(varargin)
% *MY_GUI6 M-file for my_gui6.fig
% *MY_GUI6, by itself, creates a new MY_GUI6 or raises the existing
% singleton.
% *H = MY_GUI6 returns the handle to a new MY_GUI6 or the handle to
% the existing singleton*.
% *MY_GUI6('CALLBACK', hObject, eventdata, handles,...) calls the local
% function named CALLBACK in MY_GUI6.M with the given input arguments.
% *MY_GUI6('Property', 'Value',...) creates a new MY_GUI6 or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before my_gui6_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to my_gui6_OpeningFcn via varargin.
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
% *See Appendix A for more information
% *Last Modified by GUIDE v2.5 15-Nov-2007 18:00:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @my_gui2_OpeningFcn, ...
    'gui_OutputFcn', @my_gui2_OutputFcn, ...
    'gui_LayoutFcn', [], ..., ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before my_gui6 is made visible.
function my_gui2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to my_gui2 (see VARARGIN)

% Choose default command line output for my_gui2
handles.output = hObject;

% Update handles structure

NUMERIC = xlsread('c:\temp\angles_still');
time = NUMERIC(:,1);
pitch = NUMERIC(:,2); % x-axis acceleration
roll = NUMERIC(:,3); % y-axis acceleration
yaw = NUMERIC(:,4); % z-axis acceleration

t(1) = 0;
p(1) = 0;
r(1) = 0;
y(1) = 0;

for m = 1:450
    t(m+1) = time(m);
    p(m+1) = pitch(m);
    r(m+1) = roll(m);
    y(m+1) = yaw(m);
end

% This sets up the initial plot - only do when we are invisible
% so window can get raised using my_gui6.

if strcmp(get(hObject,'Visible'),'off')
    a1 = subplot(3, 1, I ,'Parent',handles.Plots);
    plot(a1,t,p,'k.:');
    xlabel('Time (sec)','FontSize',11)
    ylabel('PITCH (deg)','FontSize',11)

    a2 = subplot(3,1,2,'Parent',handles.Plots);
    plot(a2,t,r,'k.:');
    xlabel('Time (sec)','FontSize',11)
    ylabel('ROLL (deg)','FontSize',11)

    a3 = subplot(3,1,3,'Parent',handles.Plots);
    plot(a3,t,y,'k.:');
    xlabel('Time (sec)','FontSize',11)
    ylabel('YAW (deg)','FontSize',11)
end

% UIWAIT makes my_gui6 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = my_gui6_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in UpdatePlot.
function UpdatePlot_Callback(hObject, eventdata, handles)
% hObject handle to UpdatePlot (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

popup_sel_index = get(handles.popupmenu1, 'Value');
if popup_sel_index == 1

NUMERIC = xlsread('c:\temp\angles_still');
{\text{time}} = NUMERIC(:,1);
pitch = NUMERIC(:,2); \% x-axis acceleration
roll = NUMERIC(:,3); \% y-axis acceleration
yaw = NUMERIC(:,4); \% z-axis acceleration

t(1) = 0;
p(1) = 0;
r(1) = 0;
y(1) = 0;

for m = 1:450
\text{t}(m+1) = \text{time}(m);
p(m+1) = \text{pitch}(m);
r(m+1) = \text{roll}(m);
y(m+1) = \text{yaw}(m);
end

a1 = subplot(3,1,1,'Parent',handles.Plots);
plot(a1,t,p,'k.:');
xlabel('Time (sec)','FontSize',11)
ylabel('PITCH (deg)','FontSize',11)

a2 = subplot(3,1,2,'Parent',handles.Plots);
plot(a2,t,r,'k.:');
xlabel('Time (sec)','FontSize',11)
ylabel('ROLL (deg)', 'FontSize', 11)

a3 = subplot(3, 1, 3, 'Parent', handles.Plots);
plot(a3, t, y, 'k.:');
xlabel('Time (sec)', 'FontSize', 11)
ylabel('YAW (deg)', 'FontSize', 11)

elseif popup_sel_index == 2

NUMERIC = xlsread('c:\temp\angles_moving');
time = NUMERIC(:, 1);
pitch = NUMERIC(:, 2); % x-axis acceleration
roll = NUMERIC(:, 3); % y-axis acceleration
yaw = NUMERIC(:, 4); % z-axis acceleration

t(1) = 0;
p(1) = 0;
r(1) = 0;
y(1) = 0;

for m = 1:500
    t(m+1) = time(m);
    p(m+1) = pitch(m);
    r(m+1) = roll(m);
    y(m+1) = yaw(m);
end

a4 = subplot(3, 1, 1, 'Parent', handles.Plots);
plot(a4, t, p, 'k.:');
xlabel('Time (sec)', 'FontSize', 11)
ylabel('PITCH (deg)', 'FontSize', 11)

a5 = subplot(3, 1, 2, 'Parent', handles.Plots);
plot(a5, t, r, 'k.:');
xlabel('Time (sec)', 'FontSize', 11)
ylabel('ROLL (deg)', 'FontSize', 11)

a6 = subplot(3, 1, 3, 'Parent', handles.Plots);
plot(a6, t, y, 'k.:');
xlabel('Time (sec)', 'FontSize', 11)
ylabel('YAW (deg)', 'FontSize', 11)

elseif popup_sel_index == 3

NUMERIC = xlsread('c:\temp\bits_still');
ax = NUMERIC(:,2); % x-axis acceleration
ay = NUMERIC(:,3); % y-axis acceleration
az = NUMERIC(:,4); % z-axis acceleration

time = NUMERIC(:,1);
t(1) = 0;
b_bits_output_x(1) = 0;
b_bits_output_y(1) = 0;
b_bits_output_z(1) = 0;

T = 0.1; % Time step

for m = 1:600
    t(m+1) = time(m);
    b_bits_output_x(m+1) = ax(m);
    b_bits_output_y(m+1) = ay(m);
    b_bits_output_z(m+1) = az(m);
end

elseif popup_sel_index == 4

NUMERIC = xlsread('c:\temp\bits_still');
ax = NUMERIC(:,2); % x-axis acceleration
ay = NUMERIC(:,3); % y-axis acceleration
az = NUMERIC(:,4); % z-axis acceleration

xOffset = -89;
ay_offset = 59;
az_offset = 110;
ax_gain = 2434;
ay_gain = 2408;
az_gain = 2489;

time = NUMERIC(:,1);
t(1) = 0;
T = 0.1; % Time step

for m = 1:600
    t(m+1) = time(m);
end

vx(1) = 0;
sx(1) = 0;
vy(1) = 0;
sy(1) = 0;
vz(1) = 0;
sz(1) = 0;

for n=1:600
    accel_x(n+1) = [2 * (ax(n) - xOffset)] / ax_gain;
    vx(n+1) = vx(n) + (T*accel_x(n));
    sx(n+1) = sx(n) + (T*vx(n));
end

for n=1:600
    accel_y(n+1) = [2 * (ay(n) - ay_offset)] / ay_gain;
    vy(n+1) = vy(n) + (T*accel_y(n));
    sy(n+1) = sy(n) + (T*vy(n));
end

for n=1:600
    accel_z(n+1) = [2 * (az(n) - az_offset)] / az_gain;
    vz(n+1) = vz(n) + (T*accel_z(n));
    sz(n+1) = sz(n) + (T*vz(n));
end

a10 = subplot(3,1,1,'Parent',handles.Plots);
xlabel('Time (sec)','FontSize',11)
ylabel('Displacement', 'FontSize', 11)
title('Bits Mode: Device at Rest (X-Axis)','FontSize',14)
plot(a10,t,sx,'k.:')

a11 = subplot(3,1,2,'Parent',handles.Plots);
xlabel('Time (sec)','FontSize',11)
ylabel('Velocity', 'FontSize', 11)
plot(a11,t,vx,'k.:')

76
al2 = subplot(3,1,3,'Parent',handles.Plots);
xlabel('Time (sec)','FontSize',11)
ylabel('Acceleration','FontSize',11)
plot(al2,t,accel_x,'k:');

elseif popup_sel_index == 5

NUMERIC = xlsread('c:\temp\bits_still');
ax = NUMERIC(:,2); % x-axis acceleration
ay = NUMERIC(:,3); % y-axis acceleration
az = NUMERIC(:,4); % z-axis acceleration

ax_offset = -89;
ay_offset = 59;
az_offset = 110;
ax_gain = 2434;
ay_gain = 2408;
az_gain = 2489;

time = NUMERIC(:,1);
t(1) = 0;
T = 0.1; % Time step

for m = 1:600
    t(m+1) = time(m);
end

vx(1) = 0;
sx(1) = 0;
vx(1) = 0;
sy(1) = 0;
vz(1) = 0;
sz(1) = 0;

for n=1:600
    accel_y(n+1) = [2 * (ay(n) - ay_offset)] / ay_gain;
vyy(n+1) = vy(n) + (T*accel_y(n));
syy(n+1) = sy(n) + (T*vy(n));
end

a13 = subplot(3,1,1,'Parent',handles.Plots);
xlabel('Time (sec)','FontSize',11)
ylabel('Displacement','FontSize',11)
title('Bits Mode: Device at Rest (Y-Axis)','FontSize',14)
plot(a13,t,SY,'k:');
elseif popup_sel_index == 6

NUMERIC = xlsread('c:\temp\bits_still');
ax = NUMERIC(:,2); % x-axis acceleration
ay = NUMERIC(:,3); % y-axis acceleration
az = NUMERIC(:,4); % z-axis acceleration

ax_offset = -89;
ay_offset = 59;
az_offset = 110;
ax_gain = 2434;
ay_gain = 2408;
az_gain = 2489;

ax = NUMERIC(:,1);
t(1) = 0;
T = 0.1; % Time step

for m = 1:600
    t(m+1) = time(m);
end

vx(1) = 0;
sx(1) = 0;
vy(1) = 0;
sy(1) = 0;
vz(1) = 0;
sz(1) = 0;

for n=1:600
    accel_z(n+1) = [2 * (az(n) - az_offset)] / az_gain;
vz(n+1) = vz(n) + (T*accel_z(n));
sz(n+1) = sz(n) + (T*vz(n));
end
a16 = subplot(3,1,1,'Parent',handles.Plots);
xlabel('Time (sec)','FontSize',11)
ylabel('Displacement','FontSize',11)
title('Bits Mode: Device at Rest (Z-Axis)','FontSize',14)
plot(a16,t,sz,'k:.')

a17 = subplot(3,1,2,'Parent',handles.Plots);
xlabel('Time (sec)','FontSize',11)
ylabel('Velocity','FontSize',11)
plot(a17,t,vz,'k:.')

a18 = subplot(3,1,3,'Parent',handles.Plots);
xlabel('Time (sec)','FontSize',11)
ylabel('Acceleration','FontSize',11)
plot(a18,t,accel_z,'k:.')

elseif popup_sel_index == 7

NUMERIC = xlsread('c:\temp\bits_moving');
ax = NUMERIC(:,2); % x-axis acceleration
ay = NUMERIC(:,3); % y-axis acceleration
az = NUMERIC(:,4); % z-axis acceleration

ax_offset = -89;
ay_offset = 59;
az_offset = 110;
ax_gain = 2434;
ay_gain = 2408;
az_gain = 2489;

time = NUMERIC(:,1);
t(1) = 0;
T = 0.1; % Time step

for m = 1:690
    t(m+1) = time(m);
end

vx(1) = 0;
sx(1) = 0;
vx(1) = 0;
sy(1) = 0;
vz(1) = 0;
sz(1) = 0;

for n=1:690
accel_x(n+1) = [2 * (ax(n) - ax_offset)] / ax_gain;
vx(n+1) = vx(n) + (T*accel_x(n));
sx(n+1) = sx(n) + (T*vx(n));
end

a19 = subplot(3,1,1,'Parent',handles.Plots);
xlabel('Time (sec)', 'FontSize', 11)
ylabel('Displacement', 'FontSize', 11)
title('Bits Mode: Device Rotating (X-Axis)','FontSize',14)
plot(a19,t,sx, 'k.: ')

a20 = subplot(3,1,2,'Parent',handles.Plots);
xlabel('Time (sec)', 'FontSize', 11)
ylabel('Velocity', 'FontSize', 11)
plot(a20,t,vx, 'k.: ')

a21 = subplot(3,1,3,'Parent',handles.Plots);
xlabel('Time (sec)', 'FontSize', 11)
ylabel('Acceleration', 'FontSize', 11)
plot(a21,t,accel_x, 'k.: ')

elseif popup_sel_index == 8
    NUMERIC = xlsread('c:\temp\bits_moving');
    ax = NUMERIC(:,2); % x-axis acceleration
    ay = NUMERIC(:,3); % y-axis acceleration
    az = NUMERIC(:,4); % z-axis acceleration
    ax_offset = -89;
    ay_offset = 59;
    az_offset = 110;
    ax_gain = 2434;
    ay_gain = 2408;
    az_gain = 2489;
    time = NUMERIC(:,1);
    t(1) = 0;
    T = 0.1; % Time step
    for m = 1:690
        t(m+1) = time(m);
    end
    vx(1) = 0;
    sx(1) = 0;
    vy(1) = 0;
sy(1) = 0;
vz(1) = 0;
sz(1) = 0;

for n=1:690
    accel_y(n+1) = [2 * (ay(n) - ay_offset)] / ay_gain;
    vy(n+1) = vy(n) + (T*accel_y(n));
    sy(n+1) = sy(n) + (T*vy(n));
end

elseif popup_sel_index == 9

    NUMERIC = xlsread('c:\temp\bits_moving');
    ax = NUMERIC(:,2);  % x-axis acceleration
    ay = NUMERIC(:,3);  % y-axis acceleration
    az = NUMERIC(:,4);  % z-axis acceleration

    ax_offset = -89;
    ay_offset = 59;
    az_offset = 110;
    ax_gain = 2434;
    ay_gain = 2408;
    az_gain = 2489;

    time = NUMERIC(:,1);
    t(1) = 0;
    T = 0.1;  % Time step

    for m = 1:690
        t(m+1) = time(m);
vx(1) = 0;
sx(1) = 0;
vz(1) = 0;
sz(1) = 0;
vy(1) = 0;
sy(1) = 0;

for n=1:690
    accel_z(n+1) = [2 * (az(n) - az_offset)] / az_gain;
    vz(n+1) = vz(n) + (T*accel_z(n));
    sz(n+1) = sz(n) + (T*vz(n));
end

a25 = subplot(3,1,1,'Parent',handles.Plots);
xlabel('Time (sec)','FontSize',11)
ylabel('Displacement','FontSize',11)
title('Bits Mode: Device Rotating (Z-Axis)','FontSize',14)
plot(a25,t,sz,'k.:')

a26 = subplot(3,1,2,'Parent',handles.Plots);
xlabel('Time (sec)','FontSize',11)
ylabel('Velocity','FontSize',11)
plot(a26,t,vz,'k.:')

a27 = subplot(3,1,3,'Parent',handles.Plots);
xlabel('Time (sec)','FontSize',11)
ylabel('Acceleration','FontSize',11)
plot(a27,t,accel_z,'k.:')

else
end

% function FileMenu_Callback(hObject, eventdata, handles)
% hObject    handle to FileMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% function OpenMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to OpenMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file = uigetfile('* .fig');
if ~isequal(file, 0)
    open(file);
end

% --------------------------------------------------------------------
function PrintMenultem_Callback(hObject, eventdata, handles)
% hObject    handle to PrintMenultem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
printdlg(handles.figure1)

% --------------------------------------------------------------------
function CloseMenultem_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenultem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
selection = questdlg(['Close ' get(handles.figure1,'Name') '?'], ...
    ['Close ' get(handles.figure1,'Name') ' ... '],...
    'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end

delete(handles.figure1)

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

set(hObject, 'String', {'angles_still', 'angles_rotating', ...
'DAC Output', 'bits_still_x', 'bits_still_y', 'bits_still_z',
'bits_rotating_x', 'bits_rotating_y', 'bits_rotating_z'});

function xOffset_Callback(hObject, eventdata, handles)
% hObject    handle to xOffset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if isnan(user_entry)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return
end

% --- Executes during object creation, after setting all properties.
function xOffset_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xOffset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
xOffset = str2double(get(hObject,'string'));
xOffset = 25;

function yOffset_Callback(hObject, eventdata, handles)
% hObject    handle to zOffset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
yOffset_input = str2double(get(hObject,'string'));

if isnan(yOffset_input)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return
end

% --- Executes during object creation, after setting all properties.
function yOffset_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zOffset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

function zOffset_Callback(hObject, eventdata, handles)
% hObject    handle to zOffset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
zOffset_input = str2double(get(hObject,'string'));

if isnan(zOffset_input)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return
end

% --- Executes during object creation, after setting all properties.
function zOffset_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zOffset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcnS called

function xGain_Callback(hObject, eventdata, handles)
% hObject    handle to xGain (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

xGain_input = str2double(get(hObject,'string'));
if isnan(xGain_input)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return
end

% --- Executes during object creation, after setting all properties.
function xGain_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xGain (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

function yGain_Callback(hObject, eventdata, handles)
% hObject    handle to zGain (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

yGain_input = str2double(get(hObject,'string'));
if isnan(yGain_input)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return
end

% --- Executes during object creation, after setting all properties.
function yGain_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zGain (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

85
% handles empty - handles not created until after all CreateFcns called

function zGain_Callback(hObject, eventdata, handles)
% hObject handle to zGain (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

zGain_input = str2double(get(hObject,'string'));

if isnan(zGain_input)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return
end

% --- Executes during object creation, after setting all properties.
function zGain_CreateFcn(hObject, eventdata, handles)
% hObject handle to zGain (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% --- Executes on button press in UpdateOffsets.
function UpdateOffsets_Callback(hObject, eventdata, handles)
% hObject handle to UpdateOffsets (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

ax_offset = -89;
ay_offset = 59;
az_offset = 110;

% --- Executes on button press in UpdateGains.
function UpdateGains_Callback(hObject, eventdata, handles)
% hObject handle to UpdateGains (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

ax_gain = 2434;
ay_gain = 2408;
az_gain = 2489;
APPENDIX D

Image Usage Approvals
Hi Richard,

It is OK for you to use images from our website etc, if our products are shown in a good light throughout the report. I would ask that you state that the images are courtesy of MicroStrain Inc.

Best Regards

Mike

Mike Robinson
V.P. Sales & Marketing
MicroStrain Inc
310 Hurricane Lane, Unit 4
Williston, VT 05495
Tel: +1-802-862-6629
Fax: +1-802-863-4093
Email: mirobinson@microstrain.com
Web: www.microstrain.com

Confidentiality Notice: This email and its attachments (if any) may contain confidential information of the sender which is legally privileged. The information is intended only for the use by the direct addressees of the original sender of this email. If you are not an intended recipient of the original sender (or responsible for delivering the message to such person), you are hereby notified that any review, disclosure, copying, distribution or the taking of any action in reliance of the contents of and attachments to this email is strictly prohibited. We do not waive
List of References


