FutureHand: The Use of an Inertial Measurement Unit as a USB Device

Raphaël Townshend

1 Abstract

FutureHand is a next-generation device that uses sensors hooked up to an Arduino Duemilanove to measure absolute tilt, position, velocity, and acceleration of a user’s hand. In turn, these gestures are mapped to specific keyboard and mouse commands via a serial port interface that allows for the control of virtually any digital device. Arduino programs were written to capture the raw data under a number of different test cases. The theory and mathematics for converting from the nine degrees of measurement from the sensors to yaw, pitch, and roll was developed. A MATLAB program was written to test the algorithms and convert the raw data; the program successfully tracked movements of the device within five degrees of precision. The code was then ported to Java so that it could be integrated into the Advanced Topics in Computer Science project Moonweasel for debugging and display purposes.

2 Introduction

With the invention of computers, humanity was ushered into a new era of prosperity and efficiency. The power of computing spread across the world, and soon all were benefiting from the nearly limitless possibilities of the digital world. Unfortunately, this newfound strength came at a price. As humans became increasingly intertwined with the multitude of devices that surrounded them, they began to lose touch with the physical world, which is where humans actually reside.

Thus, if it were possible to bring the physical world into the digital world, we could combine efficiency with the world we are used to interacting with and would greatly improve the lives of the millions who

This paper was written for Dr. James Dann’s Applied Science Research class in the spring of 2010.
use this technology in their everyday lives. The hand-controlled next-generation mouse that I seek to create is a first attempt at achieving this fantastic goal, and if it succeeds I will perhaps build upon it in order to help advance research in the field. The inspiration for this project came quite suddenly, while I was viewing Pranav Mistry’s 2009 talk at TED, where he displayed his revolutionary Sixth Sense device. [1] The device allowed the user to bring the digital world wherever he went, and I immediately realized that the potential behind this idea of fusion was great indeed.

For one such as myself who has always been fascinated by the world of computing yet bothered by its disconnection from reality, this newly formed field presented the ideal opportunity to research and develop new technology that would allow for the merging of the digital and physical worlds. Perhaps my work in this field will spark the interest of others, and maybe even lead to a discovery that will bring reality and computing closer than ever before.

I therefore sought to create a hand-controlled next-generation mouse that would be a first step in allowing such fusion to occur. More specifically, I attempted to build a device comprised of gyroscopes, accelerometers, and magnetometers that was hooked up to an Arduino board and that will run software that conforms to the USB protocol. This in turn will allow the user to control any device using USB ports with simple gestures of the hand, thus giving previously abstract actions actual meaning and rooting them in the physical world. The user’s movements can be mapped to virtually any command, and thus the customization and applications of such a device would be nearly infinite.

Through this project I learned how to design embedded systems that are compatible with modern-day electronics. I have also learned how to use and write programs for the Arduino and how to receive and process analog and digital inputs that I obtained to give the desired outputs. In addition, I am now able to use the MATLAB program as I needed it to run the multitude of tests necessary to perfect my complementary filter. In the near future I will research how to create a device
that complies with the USB protocol. And last but not least, I will learn how to properly package and design my product so that it could be used with a minimum of a hassle.

3 History

Surprisingly, little work has been done in the field to date. The most publicized forays into this realm were attempts to make the use of the computer more intuitive, whether via touch screens [2], trackpoints [3], or innovative mice [4, 5], but these still require users to learn an abstract interface, and there remains a degree of removal, albeit a diminished one.

In addition, there have been attempts at making controllers that resemble real-world objects. These controllers, which are more in line with my current project, keep track of the various actions the user performs upon them, and those actions are then transmitted and executed upon a digital object similar to its physical counterpart. A prime example of this can be found in arcades, where many shooter-type games have the players wield controller-weapons that allow them to control the identical in-game counterpart seamlessly. [6] Unfortunately, the use of these controllers is limited as they can only simulate one object, and thus impose physical limitations on the digital world. Nintendo has also released its own foray into the field with the Wii. Though it exploits a user’s gestures, its use and potential remain somewhat narrow. [7]

The only work that I have encountered that actually uses the user’s hands as the controller for the digital world was in fact my original source of inspiration for this project: Mistry’s Sixth Sense. [8] In the case of this device, the limitations of the physical world are done away with since we no longer rely on the intermediate controller-object, yet the control of the digital world becomes more intuitive as it uses the tools we have had practice with since the beginning of our lives.

However, I believe the lack of work in the field in fact shows how much room exists for innovation. Very little has yet been tried, and notions of what is acceptable and what is not have not been set in stone for this domain.
In addition, the extremely positive reception of Mistry’s device suggests the receptiveness of the world to innovation in the field. [9, 10] The general public and the market itself are just beginning to realize that it is possible to do away with the layer of abstraction that has been forced upon us ever since we began using digital devices.

4 Theory

4.1 The Big Picture

The FutureHand is comprised of a three-axis accelerometer, two gyroscopes, and a magnetometer (collectively called an Inertial Measurement Unit), that are hooked up to an Arduino Duemilanove, which is in turn plugged into a computer’s USB port. The role of the sensors is to determine the position, tilt, velocity, and acceleration of the user’s gestures.

In order to do so, all six degrees of freedom of a hand’s motion must be accounted for. Thus the need for all four devices, where the ADXL345 outputs linear acceleration in all three directions, the LPR530AL gives the angular velocity in two directions (pitch and roll), the LY530AL obtains the angular velocity in the remaining dimension (yaw), and the HMC5843 measures the strength of the magnetic field.

The two gyroscopes used will measure the angular velocity in three dimensions, from which we can derive angular acceleration and integrate to obtain a very rough estimate of absolute tilt. We then use the accelerometer-based estimate of pitch and roll to calibrate the gyroscopes’ readings. Finally, the magnetometer allows us to adjust the yaw measurements using a complementary filter.

The magnetometer and accelerometer are useful for measuring tilt in static situations as we need only to map their values in the local coordinate system to the expected values in the global coordinate system. In a dynamic situation, however, the readings from the gyroscope are needed to subtract the effects of gravity from the accelerometer’s readings and obtain the real linear acceleration. In addition, the magnetometer and accelerometer contain imprecisions when subjected to rapid changes, thus the gyroscope readings are needed in order to provide live tracking of these sudden modifications in positioning and tilt.
In short, the Arduino is constantly reading in data from four sensors and needs to find the optimal solution to an over-constrained six-variable system. The Duemilanové will then interpret these six degrees of freedom and use the readings to simulate keyboard and mouse commands by conforming to the USB protocol for such devices.

4.2 The Coordinate System

When dealing with sensors whose orientations are subject to constant changes, it is important that we define not only a global coordinate system but also a local one, as it is in the latter that the raw sensor data will be given while external forces such as gravity will be given in the global coordinate system, and we will thus need to transform points from one coordinate system to the other.

We accordingly define a global coordinate system (X, Y, Z) with origin located at an arbitrary point on the user’s desk. The Z-axis is up, the X-axis is lined up with the magnetic north, and the Y-axis runs perpendicular to the XZ-plane, heading west.

In addition, we define the local coordinate system (x, y, z) with origin located at the center of the accelerometer and whose orientation is determined by the chip’s labels.

Figure 1: Tilt angles as defined in 4.2 [11]
Now that we have these two systems, we need to define a way to transform a point from one system to the other. The beta version of the FutureHand will only use tilt to control digital devices. Therefore the problem boils down to properly rotating the coordinate systems. Since three degrees of freedom are associated with rotation, the transformation procedure will be done in three steps, whose order is chosen arbitrarily but, once determined, should be kept consistent. Let us assume that we are transforming a point in the global system to the appropriate position in the local system.

By convention, we first perform the rotation around the Z-axis using the right-hand rule. The amount the XY-plane is rotated by is defined as yaw (expressed as $\psi$, in degrees). Next we perform the rotation around the once-rotated Y-axis using the right-hand rule. The magnitude of this transformation is defined as pitch (expressed as $\theta$, in degrees). And finally we perform the rotation around the twice-rotated X-axis using the right-hand rule. The magnitude of this transformation is defined as roll (expressed as $\phi$, in degrees). Mathematically, this can be written as follows:

$$
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} = R_z(\phi) R_y(\theta) R_x(\psi) \begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
$$

Since we are dealing with a three-axis coordinate system, each of these rotations comes in the form of a 3x3 matrix, which we sequentially apply to the 3x1 position matrix (with the rightmost transformation matrix applied first). Here are the definitions of these transformation matrices [12]:

$$
R_z(\phi) = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \phi & -\sin \phi \\
0 & \sin \phi & \cos \phi
\end{bmatrix} \quad R_y(\theta) = \begin{bmatrix}
\cos \theta & 0 & \sin \theta \\
0 & 1 & 0 \\
-\sin \theta & 0 & \cos \theta
\end{bmatrix} \quad R_x(\psi) = \begin{bmatrix}
\cos \psi & -\sin \psi & 0 \\
\sin \psi & \cos \psi & 0 \\
0 & 0 & 1
\end{bmatrix}
$$

With these matrices we now have the capacity to rotate a three-dimensional vector in the global coordinate system to its respective position in the local coordinate system. This has many uses as far as FutureHand is concerned, most notably the ability to subtract the impact of
gravity from the accelerometer readings so that we obtain the device’s true acceleration. This is done as follows:

In the global (X, Y, Z) coordinate system, the acceleration due to gravity can be written as a vector of the form (0, 0, -g). Thus, by applying the three transformation matrices, we obtain the effect of gravity, as a 3-D vector, in the local coordinate system:

\[
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} = R_x(\phi) R_y(\theta) R_z(\psi) \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = R_x(\phi) R_z(\theta) \begin{bmatrix} g \sin(\theta) \\ 0 \\ -g \cos(\theta) \end{bmatrix} = \begin{bmatrix} -g \sin(\theta) \\ g \sin(\phi) \cos(\theta) \\ -g \cos(\phi) \cos(\theta) \end{bmatrix}
\]

And finally we obtain the true acceleration \( \left( \frac{d^2x}{dt^2}, \frac{d^2y}{dt^2}, \frac{d^2z}{dt^2} \right) \) of the device by subtracting the effects of gravity in the local coordinate system from the accelerometer’s readings \( (A_x, A_y, A_z) \):

\[
\begin{bmatrix}
\frac{d^2x}{dt^2} \\
\frac{d^2y}{dt^2} \\
\frac{d^2z}{dt^2}
\end{bmatrix} = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} - \begin{bmatrix} -g \sin(\theta) \\ g \sin(\phi) \cos(\theta) \\ -g \cos(\phi) \cos(\theta) \end{bmatrix}
\]

Thus, by knowing the yaw, pitch, and roll of the device (obtained by integrating the gyroscope’s output and calibrating using the accelerometer and magnetometer values), we can correct the accelerometer’s readings by subtracting out the net effect of gravity in the local coordinate system.

The same set of transformation matrices also provide the procedure necessary to transform a point in the local coordinate system to the global coordinate system. If we apply the same transformation matrices in the reverse order, using the negated values of the yaw, pitch, and roll, we should be able to work backwards and obtain the global coordinates of a point from its local coordinates. This makes sense as it is essentially the global-to-local procedure in reverse. Mathematically:

\[
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} = R_z(-\phi) R_y(-\theta) R_x(-\psi) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}
\]
We can then use this set of transformation matrices to convert sensor readings into the global coordinate system, thus allowing us to compute the tilt via the steps shown in Section 4.6.

4.3 LPR530AL and LY530AL Gyroscopes

The LPR530AL measures angular rate along the pitch and roll axes, whereas the LY530AL measures angular rate along the yaw axis. [13, 14] They are both composed of a micromachined structure containing a gyroscope that is permanently oscillating at high speeds. The gyroscope is free to react to changes in angular velocity in any direction, and thus, due to conservation of angular momentum, it will resist changes in its orientation. The resistive force can then be measured and the angular velocity thereby deduced and outputted as a voltage. During my experimentation I discovered that the gyro breakout boards include a high-pass filter on the output, making it more difficult to get readings that can be easily integrated.

4.4 ADXL345 Accelerometer

The ADXL345 Accelerometer measures static and dynamic acceleration in all three axes. It is essentially comprised of a micromachined structure suspended over a polysilicon wafer and differential capacitors. [15] One plate of each capacitor is attached to the suspended structure and the other is independently fixed. Thus the deflection of the structure changes the distance between the capacitors’ plates, modifying the capacitance. A circuit within the chip measures the difference between these capacitances and outputs an amplitude proportional to the acceleration. This amplitude is converted into a digital signal and sent to the Arduino using an I2C interface.

4.5 HMC5843 Magnetometer

The HMC5843 is designed to detect small magnetic fields along all three axes through the use of magneto-resistive sensors. [16] An increase in the magnetic field lowers the electrical resistance of conducting materials present within the sensors, and this resistance can in turn be measured by running a current through the element and measuring
and amplifying the resulting voltage across the conductor. [17] However, since the decrease in electrical resistance is only proportional to the perpendicular component of the magnetic field vector projected onto the current vector, we can detect the magnetic field in all three directions by placing conducting materials along each axis and measuring the resistance of each of them.

4.6 Combining Sensor Data to Compute Yaw, Pitch, Roll

As discussed beforehand, all three sensor types help determine the absolute tilt of the device, as they can each be used to compute a unique estimate of the angles or their derivatives. Once these estimates have been obtained, we need to combine them to obtain highly accurate measurements of roll, pitch, and yaw as a function of time. We are essentially dealing with a system consisting of nine degrees of measurement and three degrees of freedom, and we need to find the best possible solution for the system of equations.

There are a few different ways of approaching this problem:

- Integrate the gyroscope outputs. This is simple but only uses three degrees of measurement and would drift as the gyroscopes contain high-pass filters, which remove any constant angular changes.

- Use accelerometer readings to obtain an estimate of roll and pitch and the magnetometer data for an estimate of yaw. This would be sensitive to linear acceleration, however, and would also be very noisy. High frequency (fast) movements would also be very difficult to track.

- Apparently the optimal way to do this is with the Kalman filter. It uses all the inputs to minimize the noise in the estimates. However, this primarily relies on magic (also known as very complex mathematics).

- Finally, the solution used for the FutureHand is to mix the high-frequency gyro outputs with the drift-free accelerometer and magnetometer outputs in order to obtain a fairly noise-free and accurate estimate of yaw, pitch, and roll. This is also known as a complementary filter. [18]
We first obtain an estimate of pitch and roll using the accelerometer readings. These are given in the local coordinate system as \((A_x, A_y, A_z)\) and, when the device is at rest, correspond to the force of gravity, given as \((0, 0, -g)\) in the global coordinate system. Thus, we want to find the roll, pitch, and yaw such that:

\[
\begin{bmatrix}
    A_x \\
    A_y \\
    A_z 
\end{bmatrix}
= \begin{bmatrix}
    -g \sin(\theta) \\
    g \sin(\phi) \cos(\theta) \\
    -g \cos(\phi) \cos(\theta) 
\end{bmatrix}
\]

Note that the accelerometer readings do not depend on the yaw of the device. From the above we can calculate the pitch and roll as follows:

\[
\begin{aligned}
\theta_A &= -\sin^{-1}\left(\frac{A_x}{g}\right) \\
\phi_A &= -\tan^{-1}\left(\frac{A_y}{A_z}\right)
\end{aligned}
\]

In the above calculations, the roll can vary between \(-\pi\) and \(+\pi\), so we have to use the signs of \(A_x\) and \(A_y\) to determine the quadrant of the arctan.

If the device is not at rest, the accelerometer readings will add up to less than or more than \(g\), which will cause the above equation to break down. Yet we can partially compensate for this error by scaling the readings so that they add up to \(g\). The error caused by interpreting acceleration as gravitational pull is then minimized via the complementary filter, as will be discussed later in this section.

From the magnetometer readings \((M_x, M_y, M_z)\) we can estimate the yaw, \(\psi_M\). If the sensor is not pitched or rolled, this is straightforward using just \(M_x\) and \(M_y\):

\[
\begin{aligned}
M_x &= M_0 \cos(\psi) \\
M_y &= M_0 \sin(\psi)
\end{aligned} \Rightarrow \quad \psi_M = \tan^{-1}\left(\frac{M_y}{M_x}\right)
\]
However, if the device is pitched or rolled, the z component of the magnetic field due to inclination will confound the directional measurement. [See Section 4.7.] We can therefore use the pitch and roll estimates to tilt-compensate the magnetometer readings so that the above equations can be used. The tilt compensation is essentially putting the magnetometer readings through the local-to-global transform without the final yaw transform.

\[
\begin{bmatrix}
M'_x \\
M'_y \\
M'_z
\end{bmatrix} = R_y(-\theta) R_x(-\phi)
\begin{bmatrix}
M_x \\
M_y \\
M_z
\end{bmatrix}
\]

\[
\psi_m = \tan^{-1}\left(\frac{M'_y}{M'_x}\right)
\]

Similarly, the gyroscope readings (G_x, G_y, G_z) are in the local coordinate system and need to be converted to derivatives of the yaw, pitch, and roll: [19]

\[
\begin{bmatrix}
\frac{d\phi}{dt} \\
\frac{d\theta}{dt} \\
\frac{d\psi}{dt}
\end{bmatrix} =
\begin{bmatrix}
1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\
0 & \cos\phi & -\sin\phi \\
0 & \sin\phi\sec\theta & \cos\phi\sec\theta
\end{bmatrix}
\begin{bmatrix}
G_x \\
G_y \\
G_z
\end{bmatrix}
\]

At this point we have estimates of the yaw, pitch, and roll that are accurate in the long-term and estimates of their rates of change that capture rapid changes. These are combined using a complementary filter one time step at a time to form our final estimates of \(\phi(t), \theta(t),\) and \(\psi(t):\)

\[
\phi(t) = 0.98\left(\phi(t-\Delta t) + \frac{d\phi}{dt}\Delta t\right) + 0.02\phi_m(t)
\]

\[
\theta(t) = 0.98\left(\theta(t-\Delta t) + \frac{d\theta}{dt}\Delta t\right) + 0.02\theta_m(t)
\]

\[
\psi(t) = 0.98\left(\psi(t-\Delta t) + \frac{d\psi}{dt}\Delta t\right) + 0.02\psi_m(t)
\]
4.7 Inclination of the Earth’s Magnetic Field

Figure 2: A graphical representation of the Earth’s magnetic field. Frame A shows the overall field lines, whereas Frame B shows the direction of the magnetic field at different latitudes along the Earth’s surface. [20]

Unlike gravity, which we used in Section 4.6 to obtain an accelerometer-based estimate of tilt, the strength and direction of a magnetic field vary widely for different regions of the Earth. [21] This can be problematic as we use the magnetic field’s vector in global coordinates to obtain a magnetometer-based estimate of yaw. Though the magnitude of the field can easily be calculated from the magnetometer readings,
finding the direction proves to be harder as the Earth’s magnetic field lines are not parallel to the Earth’s surface, as shown in Figure 2. Even though the field lines continue to point in a northward direction, we observe that as we approach the magnetic north the inclination of the magnetic field increases, augmenting the vertical component of the field in our global coordinates while decreasing the horizontal component. Therefore, every time the FutureHand is used, the inclination of the magnetic field must be recalculated so that our computations remain properly calibrated. (This is possible as we have an over-constrained system.) Once the magnetic field is obtained, however, we can assume it is constant as the user will probably not move enough during the time of use such that the field experiences a significant change in direction. As a final note, one significant problem encountered is that any transient magnetic field will confound the magnetometer readings and yield erroneous results in the yaw calculations.

5 Design

The FutureHand is a small device that can be worn on a hand, either as a glove or with an attaching strap that detects the movement of the hand. Any motion, tilt or acceleration of the hand can be measured and converted into commands for the host computer using algorithms that I will design. In this way, intuitive gestures can be used to control various pieces of computer software. For example, windows could be moved around by swiping, viewing direction in a game could be controlled by hand tilt, or a document could be scrolled by sharply rotating your hand.
Figure 3: Breadboard view of ADXL345 Accelerometer, LRP530AL Gyroscope, LY530AL Gyroscope, and HMC5843 Magnetometer wired up to Arduino Duemilanove. This setup allows for the running of the sample (source code available online at http://roundtable.menloschool.org) and allows for the reading of the accelerometer’s, magnetometer’s, and gyroscopes’ output.
Figure 4: Schematic view of Breadboard shown in Figure 3. The Arduino's Analog inputs 4 and 5 are being used for digital I2C communications with A4 being the data line and A5 being the clock line. The resistors are in place in order to prevent the short circuit that would otherwise occur if A4 and A5 were directly linked to the voltage source. Both the HMC5843 and ADXL345 are using I2C communication and are hooked in parallel to the A4 and A5 ports. The gyroscopes are connected to regular analog ports.
6 Results

6.1 The Big Picture

So far I have been able to hook up an accelerometer, two gyroscopes, and a magnetometer to the Arduino and nine degrees of measurement of data are successfully being read out. I have written an Arduino test program using the Wire library that sends the accelerometer readings to a Processing program that constructs a graphical display of the data and writes it to a text file. I then wrote a program in MATLAB that successfully calculated roll, pitch, and yaw from these text files. Finally, I wrote a Java program that interfaces with the Advanced Topics in Computer Science project “Moonweasel” by controlling a starship with dynamically calculated Euler angles. In the process I have learned how to use the Arduino and its various features effectively, how to read and comprehend a component’s data sheets and schematics efficiently, how to set up an I2C wiring interface, how to wield the power behind Processing and MATLAB applications, and how to properly read data off a serial port.

6.2 Developing in MATLAB

To correctly develop the mathematics needed to get a smooth and accurate output, I ran several tests and captured the raw data into files as mentioned in Section 4.2. The test cases that I ran include the following:

1. Pause 2 s, Rotate 90° around Y-axis, pause 2 s, return, pause 2 s.
2. Pause 2 s, Rotate 90° around X-axis, pause 2 s, return,
   Pause 2 s, Rotate 90° around Y-axis, pause 2 s, return,
   Pause 2 s, rotate 90° around Z-axis, pause 2 s, return, pause 2 s.
3. Pause 2 s, Rotate 90° around Y-axis, pause 2 s,
   Rotate 90° around X-axis, pause 2 s,
   Rotate 90° around Z-axis, pause 2 s,
   Rotate -90° around X-axis (back to start), pause 2 s.
4. Pause 2 s, Rotate 360° around Z-axis, pause 2 s.
5. No movement for 10 s.
The data from these were loaded into MATLAB and processed with a program I wrote that implements the transformations, tilt compensation, complementary filter, and drift compensation. (See Appendix 9.3.) This program successfully uses nine degrees of measurement to calculate a fairly accurate and rapid estimate of the yaw, pitch, and roll (discussed more in depth in Section 4.6).

For example, Figure 5 shows the raw data and outputs from a run of Test #4. The top panel shows the raw data from the accelerometer. The values remain constant, with a small amount of noise. The x and y values are near zero, and the z value reflects the gravitational force acting straight down as though the device were being accelerated upward at one g.

Panel 2 shows the gyroscope raw data. The x and y values remain relatively constant since it was not being rotated in those directions, but the z values show a rotation averaging 20 degrees per second. The noisiness reflects the inconsistency of moving it by hand.

Panel 3 displays the normalized magnetometer data. Even though the device was lying flat, there is a z component to the readings due to the inclination of the Earth’s magnetic field. In fact, the inclination was calculated from these measurements to be 52° with a field strength of 0.38 Gauss. This compares reasonably with the values from NOAA of 61° and 0.49 Gauss. [22] The next panel shows the tilt-compensated values. In this test the tilt compensation has no effect since the device was lying flat.

Panel 5 shows the roll, pitch, and yaw derived from the accelerometer and magnetometer. It clearly shows that the pitch and roll remain near zero and the yaw went through a full rotation of 360°. Since the yaw and roll angles are constrained to be between -180° and +180°, there is a jump at around 10 seconds. The noise on the roll and pitch is due to bumpiness (linear acceleration) while moving the device.
Panel 6 shows the final yaw, pitch, and roll output from the complementary filter. The transient noise shown in Panel 5 has been suppressed since the gyro did not indicate such large changes in the roll and pitch. Otherwise it tracks the previous panel. The final panel shows the difference between the two prior panels.

Similar outputs for the other test cases are included in Appendix C. In all cases the final estimates of roll, pitch, and yaw track the expected values well. The plots also show the effect of passing +/- 180° in the roll or yaw, resulting in sharp discontinuities. In addition, they show the effects of pitching up to +90°. At that point there are multiple possible values for the roll and yaw. Furthermore, when the device is pitched past vertical, the pitch starts decreasing again and the roll and yaw suddenly jump by 180°. These are all artifacts of representing the orientation using yaw, pitch, and roll and do not reflect a large jump in the actual orientation.
Figure 5: MATLAB Plots for Test Case #4
6.3 Building an Arduino-Computer Interface

Though the Arduino is responsible for the reception and compilation of the sensors’ outputs, it cannot act as a stand-alone device for a couple of reasons. For one, the FutureHand is intended to function in conjunction with a computer that the user desires to control, and thus from a purely design standpoint it must be able to transmit data to the host device. However, the Arduino is also difficult to program and might not have enough processing speed to compute the six degrees of freedom desired from the nine degrees of measurement, and so must be able to send data to a processing application on a more powerful device (in this case, the host).

In order to communicate with the host device the Arduino transmits its serial output to a USB interface, which allows the computer to read the data. It then outputs the data via a virtual serial port that any computer software can read by accessing the /dev folder.

The first program used was Processing, and though it was too low-level to easily implement a 6-dof calculator, it was able to display the sensor data graphically in a manner that allowed for some initial debugging.

The next step was to construct an actual interpreter for the sensor data that would output the yaw, pitch, and roll of the FutureHand. MATLAB, a very robust system for these types of tasks, was first employed, and a high number of flaws were eliminated thanks to a series of standardized tests that were then analyzed within the program. However, a fundamental flaw of this system was that it could not calculate the Euler angles in real time, and instead read them in from a pre-generated text file.

Thus, I decided to port the now mostly bug-free MATLAB code to an environment more suited for reading in data live from a serial port. I ended up selecting Java as it had that capability as well as being very familiar to me already. However, doing so was quite difficult as I had built my code around a text file input format, where all the data is already known, and I had to port the code to a loop-based structure more adapted to real-time processing.
Fortunately, the attempt was successful, and I was able to write a small
display program that accomplished much the same tasks as the original Processing one with the addition of the Euler angles.

The last step was to integrate the obtained Euler angles into a demonstration program. I chose the Advanced Topics in Computer Science’s 3-D space flight simulator named “Moonweasel.” The Java 6-dof calculator was routinely polled by the main Moonweasel program for the current orientation of the FutureHand, and the results were applied to the orientation of the in-game vessel shown in Figure 6.

![Advanced Topics in Computer Science project Moonweasel being controlled by the FutureHand](image)

**Figure 6**: Advanced Topics in Computer Science project Moonweasel being controlled by the FutureHand

### 6.4 Packaging

The FutureHand’s current packaging consists of a water bottle containing Arduino Duemilanove and the various sensors. The USB wire feeds through the mouth of the bottle. This format allows for easy manipula-
tion of the device while minimizing the likelihood of damage. However, it is difficult to access the device and it is quite a bulky design.

Future iterations will probably consist of much smaller metal or plastic boxes containing the Arduino and its sensors so that it takes up less space and is easier to manipulate. It will also be much easier to access the wirings.

7 The Next Step

The next step is to package the FutureHand into a more compressed shell in order to facilitate its use. Afterwards I will modify my code so that it also includes translation calculations. Once that is done I can begin mapping a variety of gestures to keyboard and mouse commands. Another feature that I would like to implement is wireless transmission of the sensor data to the computers via X-Bees, but I lack the proper components. I will also do extensive debugging and testing to ensure that the system functions properly and as desired. And finally, if time permits, I will design a circuit board and package the components into a small unit.

The main obstacle so far has definitely been figuring out all the math behind the transformations, tilt compensation, and complementary filter implemented in MATLAB. Going forward, I believe that debugging the massive amounts of code will be quite difficult since the math behind this is not trivial and the mistakes are often not obvious.

8 Acknowledgements

I would like to thank my father for his contributions to this project in the form of a student edition of MATLAB, various circuit building equipment, and his critical reading of the manuscript.
9 Appendices

9.1 Appendix A: Parts List

The following parts are available from Sparkfun.com (already obtained):

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
<th>Use</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple Axis Accelerometer Breakout - ADXL345</td>
<td>See theory section</td>
<td>Measures tilt and linear acceleration</td>
<td>$27.95</td>
</tr>
<tr>
<td>Gyro Breakout Board - LY530AL - 300°/s</td>
<td>See theory section</td>
<td>Measures angular velocity in terms of yaw</td>
<td>$19.95</td>
</tr>
<tr>
<td>Gyro Breakout Board - LPR530AL Dual 300°/s</td>
<td>See theory section</td>
<td>Measures angular velocity in terms of pitch and roll</td>
<td>$29.95</td>
</tr>
<tr>
<td>Arduino USB Board</td>
<td>Arduino Duemilanove</td>
<td>Interfaces between the sensors and the device employing FutureHand</td>
<td>$29.95</td>
</tr>
<tr>
<td>Triple Axis Magnetometer Breakout - HMC5843</td>
<td>See theory section</td>
<td>Measures magnetic field</td>
<td>$49.95</td>
</tr>
</tbody>
</table>

9.2 Appendix B: Source Code

Source code for the accelerometer/gyroscope test program, processing display program, MATLAB text file input code, MATLAB text file loop code, and Java dynamic loop code can be found online at http://roundtable.menloschool.org.
9.3 Appendix C: Test Plots from MATLAB

Figure 7: MATLAB plots for Test Case #1: Pause 2 s, rotate 90° around Y-axis, pause 2 s, return, pause 2 s
Figure 8: MATLAB plots for Test Case #2: Pause 2 s, rotate 90° around X-axis, pause 2 s, return, pause 2 s, rotate 90° around Y-axis, pause 2 s, return, pause 2 s, rotate 90° around Z-axis, pause 2 s, return, pause 2 s.
Figure 9: MATLAB plots for Test Case #3: Pause 2 s, rotate $90^\circ$ around Y-axis, pause 2 s, rotate $90^\circ$ around X-axis, pause 2 s, rotate $90^\circ$ around Z-axis, pause 2 s, rotate $-90^\circ$ around X-axis (back to start), pause 2 s.
Figure 10: MATLAB plots for Test Case #4: Pause 2 s, rotate 360° around Z-axis, pause 2 s.
10 Citations


[8] Pranav Mistry: The thrilling potential of SixthSense technology.


[13] LY530AL datasheet,

[14] LPR530AL datasheet,


[16] HMC5843 datasheet,

[17] Wikipedia: Magnetoresistance,
http://en.wikipedia.org/wiki/Magnetoresistance


[19] Term Project 2: Electronic Directional Gyro MAE 5483,
http://www.caselab.okstate.edu/ocharle/projects/directionalgyro.pdf


[21] Flight Instruments - Level 3 - Magnetic Compass,
http://www.allstar.fiu.edu/aero/MagComp.htm

[22] National Geophysical Data Center,
http://www.ngdc.noaa.gov/geomag/magfield.shtml