Drift Compensated Inertial Position Sensor for Healthcare Patient Monitoring

by

David Lee Nelson

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2005

Author .....
Department of Electrical Engineering and Computer Science
August 16, 2005

Certified by .....
Dr. William J. Long
Principal Research Associate
Thesis Supervisor

Accepted b ....
v ...nith
Chairman, Department Committee on Graduate Students

Drift Compensated Inertial Position Sensor for Healthcare Patient Monitoring

by

David Lee Nelson

Submitted to the Department of Electrical Engineering and Computer Science
on August 16, 2005, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In order to provide more effective health care, especially to the elderly, we must enable the physician to monitor the patient outside of the clinic or hospital. A patient's activities are a critical indicator of his or her well-being, and the physician must have an un-intrusive and inexpensive means of monitoring patient activity. The objective of this project was to design and construct a low-cost, low-power, six degree-of-freedom inertial activity monitor that can be used with a portable computer. In this thesis, I describe the design and implementation of a such a monitor that can communicate using several popular peripheral bus protocols. I describe a simple attitude estimation filter and give a qualitative assessment of its performance.

Thesis Supervisor: Dr. William J. Long
Title: Principal Research Associate

Acknowledgments

I would like to thank my advisor for supporting my work on this project, and for allowing me the space to persue my own wild ideas along the way. I would also like to thank Jack Memishian of Analog Devices for introducing me to the joys of inertial sensors, for donating a few of them to the cause, and for giving me some tips on how to work with them effectively. Finally, I would like to thank my father for instilling in me a curiosity about all things technical, and for providing a goal toward which to strive.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Vision

In order to provide more effective health care, especially to the elderly, we must enable the physician to monitor the patient outside of the clinic or hospital. A patient's activities are a critical indicator of his or her well-being, and the physician must have an un-intrusive and inexpensive means of monitoring patient activity. A measure of a patient's motion, position, and orientation provides valuable data, which when combined with post-processing algorithms allow a physician to determine the patient's actions and search for abnormalities that might indicate a health problem.

The Clinical Decision Making group at MIT CSAIL is dedicated to exploring and furthering the application of technology and AI to clinical situations. Because of the importance of the medical field and its need for fast and accurate information, the group also focuses on the gathering, availability, security and use of medical information. The group is currently pursuing a project to allow doctors to monitor a patient at home over an extended period. The hope is to detect changes in behavior patterns by using a handheld computer as a mobile sensor platform to record and analyze a patient's movements. The system will then determine the patient's activities and watch for signs of illness or improvement and for abnormalities that may need further investigation. As part of its task, the system needs to be able to determine the patient's motion within a wide range of operating environments (i.e. home, outside,

car, etc.). The iPaq handheld computer currently supports a GPS extension and some limited inertial sensor extensions; however there are currently no inertial six degree-of-freedom position sensors for the iPaq.

The objective of this project was to design and construct a low-cost, low-power six degree-of-freedom inertial sensor that could be used with the iPaq portable computer.

1.2 Organization

First, I discuss the requirements of a patient activity monitor, and then mention other work related to inertial activity monitoring. Chapter 2 briefly examines three types of monitoring systems: GPS-based, acoustic, and inertial and then describes the operation of the inertial sensors used for this project. In Chapter 3, I explore the mathematics of representing position and orientation, and conclude that a quaternion representation best meets the needs of the project. Chapter 4 describes a simple attitude estimation filter and its sub-components, and Chapter 5 lays out the design of the monitor hardware. In Chapter 6, I discuss how the device met the design goals, and give a qualitative assessment of the performance of the attitude estimation filter. Chapter 7 concludes by exploring some applications of this activity monitor, and lists the contributions of this project.

1.3 Requirements

The project requires a low-cost, low-power, inertial six degree-of-freedom sensor for use with a portable computer. Current implementations of inertial position sensors require costly, high-precision devices to compensate for the drift that occurs when performing open-loop integration on the sensor data. I have instead used low-cost devices and leveraged advances in MEMs technology to obtain data that, although not perfect, is nonetheless useful for monitoring patient health. Just last year, Analog Devices developed a low-cost, dual-axis accelerometer that remains an order of magnitude more stable over temperature fluctuations than their previous models [1].

As recently as 2002, researchers carefully studied human motion in an effort to determine how patients move when they lose their balance [12]. Such information about the way humans move, both while healthy and when ill, combined with inertial activity measurements, may allow doctors to diagnose patients and monitor their recovery.

1.4 Related Work

Sabelman has used a human mounted accelerometer system to build inertial models of human fall patterns with the hope of recognizing pre-fall situations and improving diagnosis and treatment of balance disorders. He chose inertial sensors for their ability to be used anywhere, and uses them to analyze movement patterns in stroke victims and joint replacement patients. He also uses his system to analyze tremors in Parkinson's patients to determine how thy respond to treatment. He has also incorporated a mechanism to provide real-time feedback to the patient. However, he uses only accelerometers, and positions them at twelve locations along the body, with wires running to a dedicated, proprietary handheld computer, which captures and processes the data [12]. This custom fabricated computer adds cost, and requires the real-time analysis software to be written in the language of the proprietary system. Although his work will likely provide the basis for analyzing the output of inertial activity monitors, his device does not capture gyro data, and is not suitable for every-day use.

Bachmann, et al, developed the quaternion based filter used in this project. Its purpose was to insert humans into artificial environments and to track robot joint angles. They recognize the benefit of inertial sensors as an infrastructure-free sensing solution, and recommend using multiple six degree-of-freedom sensors on the human or robot to be tracked. They used low-cost, off-the-shelf components, but did not intend their device to be small or portable; it is wired directly into a PCI data acquisition card in a PC. They did not envision the sensor as a device for every-day use, but as part of a body suit to be worn for short periods in a virtual environment [3].

Marins, et al, have developed a process model and Kalman filter for fusing sensor data from accelerometers, gyroscopes and magnetic field sensors. They, too, recognize the utility of inertial sensors and the superiority of a quaternion-based representation. By using a filter like that described in [3], they have succeeded in designing a linear Kalman filter for orientation estimation. Unfortunately, the majority of their work is theoretical and done only in simulation. The brief trial performed on real data showed deviations in quaternion components of up to 15% [10].

In [9], Luinge explores various combinations of inertial sensors for measuring human arm orientation. He recognizes the value of inertial sensors for the measurement of human ambulatory motion, and develops models of arm movement and signal error generation. He also develops a theoretical model for a single-mass inertial six degree-of-freedom sensor. However, he focuses his attention on joint orientation measurement rather than activity monitoring, and gives no explicit thought to device cost.

Researchers at NovAtel and Honeywell examine ways of integrating GPS and inertial measurement units for general tracking purposes in [6]. They have achieved sub-meter resolution of position tracking between GPS updates, but their target applications are automotive and aerospace. As such, they rely on extremely expensive devices, which are not portable. Their objective is sensor fusion rather than healthcare.

Veltink, et al, compare the utility of a six degree-of-freedom inertial sensor to that of the human vestibular system in [13]. They constructed and tested a single-mass, tri-axial accelerometer, which they use to show that angular rate data alone are insufficient for human orientation tracking. They designed their accelerometer to be small and portable, but low cost and low power were not explicit design objectives.

Chapter 2

Types of Monitors

2.1 GPS-based

Activity monitors based on the Global Positioning System (GPS) provide coarse position information with accuracy of several meters. A system called Differential GPS (DGPS) exists, and can provide higher resolution by using ground based transmitters to correct errors in the GPS transmissions. A system using GPS can track position anywhere in the world and requires no extra infrastructure. Unfortunately, a GPS receiver cannot get a signal indoors, and nearby buildings and atmospheric conditions can severely degrade GPS signal integrity. Finally, the resolution of a GPS-based system limits is utility to tracking a person's general location (i.e., home, supermarket, restaurant) as opposed to his or her individual bodily movements.

2.2 Acoustic

Acoustic motion trackers measure the time-of-flight of acoustic (usually ultrasound) signals between beacons with known locations and the transceiver to be tracked. Based on the time-of-flight to at least three beacons and the local speed of sound, the receiver can determine the distance to each of the beacons and triangulate its location. This, of course, requires the receiver to have line-of-sight to at least three beacons at all times. Using ultrasonic beacons in a patient's house presents additional problems

because of reflections, multi-path signals, and the possibility of beacons being bumped or moved. Recent work in the Networks and Mobile Systems group at MIT CSAIL allow the beacons to dynamically determine their relative position as they are moved as long as at least three remain stationary [11].

2.3 Inertial

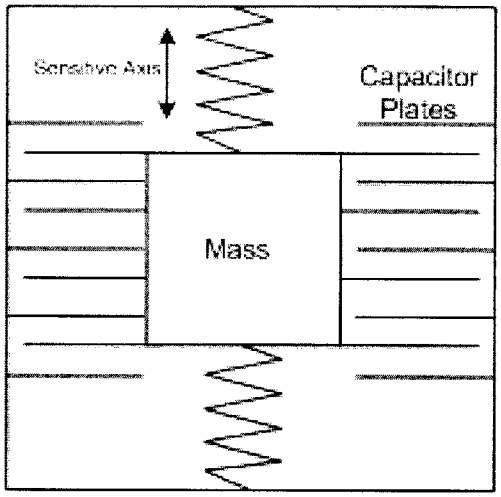
Inertial tracking systems measure the forces that the object being tracked exerts on the tracking device. By applying Newtonian mechanics and the equations of rigid-body motion, the tracker can estimate the trajectory followed by the tracker. Typically, the tracker uses three accelerometers to measure lateral accelerations, and three angular rate sensors to measure rotations about all three axes. The angular rate sensors determine the orientation of the device so that the acceleration due to gravity can be subtracted from the acceleration measurement. The major benefit of the inertial system is that it does not require any extra infrastructure, so it works in any environment without any special preparation. It provides local resolution of a few centimeters, although its global accuracy will drift with time. Indeed, drift is often seen as a major shortcoming of inertial systems. Error introduced by noise in the sensors accumulates over time and distorts the global position and orientation estimate. However, for the purpose of monitoring the activity of a patient, the inertial system provides fine grained local information that can be later applied to models of patient activity to determine behavior and, most importantly, deviations from normal behavior.

2.3.1 Inertial Sensors

Accelerometers

Advances in semiconductor fabrication technology have led to the development of Micro Electro-Mechanical Systems (MEMS) which are silicon chips with moving parts. The basic operation of an accelerometer is as a spring-mass system. A set of springs

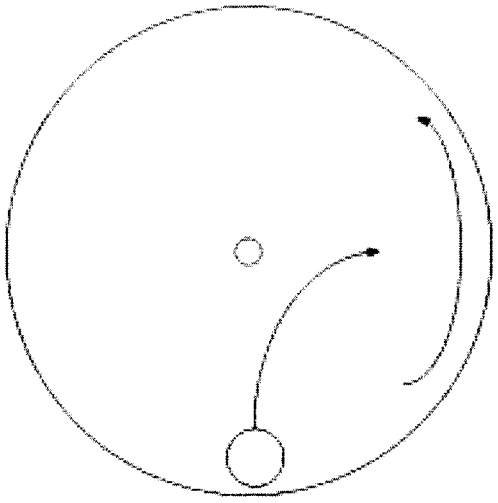
Figure 2-1: An accelerometer spring-mass system



suspends a mass on the chip (see Figure 2-1). As the device accelerates, the inertia of the mass causes the springs to stretch and compress until the spring-force equals the applied force and the mass accelerates with the device. The acceleration of the device is proportional to the deflection of the springs. The devices used employ a differential capacitor to measure the mass displacement. As the mass moves, it changes the distance between the plates of a parallel plate capacitor, which induces a differential capacitance in the system. Sensitive signal conditioning circuitry then amplifies and filters the signal, producing an analog voltage proportional to the acceleration.

By placing springs on each side of a massive block, acceleration can be measured along all three spatial dimensions. For manufacturing reasons, it is much simpler to only place springs on the sides of the mass and not on the top and bottom, yielding a two-axis accelerometer rather than a three-axis one. This requires the use of two separate devices to span the acceleration space, and introduces problems of device alignment.

Figure 2-2: Example situation demonstrating the Coriolis force



Gyroscopic Angular Rate Sensors

There are three basic types of gyroscopic angular rate sensors: spinning mass gyroscopes, laser ring gyroscopes, and vibrating mass gyroscopes. Spinning mass gyroscopes, when rotated about an axis other than the spin axis, exert a torque perpendicular to both the spin axis and the axis of rotation. This torque is proportional to angular velocity and can be measured by the displacement of springs. Laser ring gyroscopes operate by splitting a laser beam and sending it in opposite directions around a circular path. The beams are then recombined and measured by a detector. If there is no rotation about the sensitive axis, then the path lengths will be the same, and the beams will interfere constructively at the detector. If there is rotation about the sensitive axis, then the path lengths will differ, and there will be some destructive interference at the detector. The amplitude of the signal at the detector depends on the phase shift of the beams, which depends on the angular velocity. Neither the spinning mass gyroscope nor the laser ring gyroscope is suitable for use as a personal activity monitor because they are too large and quite expensive.

By contrast, vibrating mass gyroscopes are both small and inexpensive. They operate by setting a mass in motion perpendicular to the sensitive axis, and measuring the Coriolis acceleration of the mass. The Coriolis force, like the centrifugal force, is

a fictitious force which appears as the effect of inertia in rotating frames of reference. Imagine standing on the edge of a spinning merry-go-round facing the center and trying to throw a ball to your friend standing opposite you. If you throw the ball straight across the center of rotation, that is, directly to your friend, it will appear, from your rotating frame of reference, to curve to the side, depending on which direction the merry-go-round is spinning (see Figure 2-2). From the ground's stationary reference frame, of course, the ball appears to travel straight, and you appear to rotate to the side. This is the Coriolis Effect: from the rotating frame of reference, it appears as though a force - the Coriolis force - has accelerated your ball perpendicular to both the axis of rotation and to the velocity of the ball. The vibrating mass gyroscope sets a mass oscillating perpendicular to the sensitive axis, like throwing the ball back and forth. When the sensor rotates, the mass accelerates perpendicular to both the axis of oscillation and the sensitive axis, with force given by:

$$F_c = 2m \cdot v \cdot \omega$$

Where m is the mass of the block, v is the velocity of the mass along the axis of oscillation, and ω is the angular velocity about the sensitive axis. This acceleration is then measured by springs and differential capacitors as with the accelerometers.

Chapter 3

Representation

Scientists and engineers working on aircraft navigation, computer graphics, robotics, and many other fields have needed to describe and compute rotations and orientation in three dimensions. From this need have arisen two popular schemes for representing rotations in three dimensions: Euler Angles, and Quaternions. In choosing a representation, one must remember that it should be a medium for efficient computation, and it represents an ontological commitment, defining how the designer thinks about the world, and what can be said about it, and what it can say [4].

3.1 Euler Angles

The Euler method involves applying an ordered set of rotations about three perpendicular axes to the world coordinate frame to bring it into alignment with the body coordinate frame. Although there are twelve possibilities for these rotations, the most commonly used angles are azimuth (ψ), elevation (θ), and roll (ϕ) [5].

3.1.1 Rotation

Euler showed that any rotation about any axis can be broken down into three successive rotations about three perpendicular axes. When those axes are coincident with the coordinate axes, the rotations are known as Euler Angles, and can take on the

following ranges of values:

$$\psi = \pm\pi \quad \theta = \pm\tfrac{\pi}{2} \quad \phi = \pm\pi$$

Thus, any rotation can be represented by three rotation matrices, one about each axis: $R = R_z R_y R_x$, where the first rotation is associated with the rightmost matrix. The Euler Angle rotation matrices are therefore:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Because it is often necessary to perform translations as well as rotations, many applications use homogeneous coordinates and homogeneous rotation matrices.

3.1.2 Body Rates

The rotation rates returned by a tracking device in its own coordinate frame are referred to as body rates: roll (p), pitch(q), and yaw (r).

$$B_\omega \quad = \quad [pqr]^{\mathrm{T}}$$

Body rates are presented in the body coordinate frame, and must not be confused with Euler rates $(\dot{\phi}, \dot{\theta}, \dot{\phi})$, which are in the world coordinate frame. To convert body rates to Euler rates, one must perform the appropriate intermediate rotations. The

rotation rates in world coordinates are

$$W_\omega = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + R_z \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_z R_y \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix}$$

So the rotation in body coordinates is given by

$$B_\omega = [R_z R_y R_x]^{\mathrm{T}} W_\omega$$

From the definition of the rotation matrices,

$$B_\omega = \dot{\psi} \begin{bmatrix} -\sin(\theta) \\ \sin(\theta)\cos(\theta) \\ \cos(\phi)\cos(\theta) \end{bmatrix} + \dot{\theta} \begin{bmatrix} 0 \\ \cos(\phi) \\ -\sin(\phi) \end{bmatrix} + \dot{\phi} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Solving for the Euler angle rates gives,

$$\begin{aligned} \dot{\theta} &= q\cos(\phi) - r\sin(\phi) \\ \dot{\psi} &= r\sec(\theta)\cos(\phi) + q\sec(\theta)\sin(\phi) \\ \dot{\phi} &= p + r\tan(\theta)\cos(\phi) + q\tan(\theta)\sin(\phi) \end{aligned}$$

These equations are also known as the gimbal rate equations [5].

3.1.3 Gimbal Lock

A brief inspection of the gimbal rate equations reveals that, when the elevation goes through vertical ($\theta = \pm\frac{\pi}{2}$), the secant and tangent functions become undefined. Indeed, when the system is implemented numerically, which is the case for almost all practical systems, as θ approaches $\pm\frac{\pi}{2}$, the secant and tangent functions will likely overflow the system's numerical precision.

Physically, when the elevation approaches vertical, the roll axis becomes coincident with the azimuth axis. When the system is implemented mechanically, with rotating concentric rings (gimbals), it may be impossible to separate the roll and azimuth rings once elevation has gone through vertical. This situation is called "gimbal lock." In situations such as the Apollo missions, the navigation computer would issue a warning whenever the spacecraft maneuvered too close to the gimbal lock singularity. The MIT Instrumentation Laboratory proposed the use of a fourth redundant gimbal to prevent gimbal lock [7]. Other solutions used in computer simulations involve hacks to the code to prevent division by zero, usually limiting the range through which objects may be tracked.

3.2 Quaternions

3.2.1 Definitions

Quaternions are an extension of complex numbers, first described by Sir William Hamilton, which form a normed division algebra over the real numbers. In addition to the root imaginary number, i, used in complex algebra, quaternions add j, and k, which satisfy the following rules:

$$\begin{aligned} i^2 = j^2 = k^2 &= ijk = -1 \\ ij = k & \qquad ji = -k \\ jk = i & \qquad kj = -i \\ ki = j & \qquad ik = -j \end{aligned}$$

Quaternions form a vector space over $\mathbb{R}^4$, where every quaternion can be expressed as a linear combination of the four basis quaternions: $q = a + bi + cj + dk$. Quaternion addition, subtraction and scalar multiplication are performed as with vectors. Indeed, quaternions can be thought of, and are often written as, a real scalar part and an

imaginary vector part: (a, v).

The quaternion conjugate is defined as $q^* = (a, -v)$, and the quaternion norm is defined as $|q|^2 = qq^* = a^2 + |v|^2$.

3.2.2 Manipulation

It should be immediately apparent from the table of quaternion basis products that quaternion multiplication is non-commutative. It is, however associative, and is completely defined by the above table:

Let,

$$\begin{aligned} q_1 &= a_1 + b_1 i + c_1 j + d_1 k \\ q_2 &= a_2 + b_2 i + c_2 j + d_2 k \end{aligned}$$

Then the quaternion product is

$$\begin{aligned} q1q2 &= (a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2) + \\ &\quad (b_1 a_2 + a_1 b_2 + d_1 c_2 + c_1 d_2) i + \\ &\quad (c_1 a_2 + d_1 b_2 + a_1 c_2 - b_1 d_2) j + \\ &\quad (d_1 a_2 - c_1 b_2 + b_1 c_2 + a_1 d_2) k \end{aligned}$$

or, in vector notation:

$$q_1 q_2 = (a_1 a_2 - v_1 \bullet v_2, a_1 v_2 + a_2 v_1 + v_1 \times v_2)$$

The presence of the vector cross product indicates also that the quaternion product is non-commutative.

It is clear from the definition of the quaternion norm that the product of a quater-

nion and its conjugate always has a zero vector component, so

$$\frac{qq^*}{\text{norm}(q)} = 1$$

which implies that every non-zero quaternion has an inverse:

$$q^{-1} = \frac{q^*}{\text{norm}(q)}$$

Note that in the special case of a unit quaternion, $q^{-1} = q^*$. This fact will become important when using quaternions to represent rotations.

From the definition of the quaternion norm, it follows that the norm of a product is the product of the norms,

$$\begin{aligned}
q_1 q_2 &= (a_1 a_2 - v_1 \bullet v_2, a_1 v_2 + a_2 v_1 + v_1 \times v_2) \\
|q_1 q_2|^2 &= (a_1 a_2 - v_1 \bullet v_2)^2 + (a_1 v_2 + a_2 v_1 + v_1 \times v_2) \bullet (a_1 v_2 + a_2 v_1 + v_1 \times v_2) \\
&= a_1^2 a_2^2 - 2 a_1 a_2 v_1 \bullet v_2 + v_1^2 v_2^2 \cos^2(\theta) + \\
&\quad a_1^2 v_2^2 + a_1 a_2 v_1 \bullet v_2 + 0 + \\
&\quad a_1 a_2 v_1 \bullet v_2 + a_2^2 v_1^2 + 0 + \\
&\quad 0 + 0 + v_1^2 v_2^2 \sin^2(\theta) \\
&= a_1^2 a_2^2 + a_1^2 v_2^2 + a_2^2 v_1^2 + v_1^2 v_2^2 \\
&= (a_1^2 + v_1^2)(a_2^2 + v_2^2) \\
|q_1 q_2|^2 &= |q_1|^2 |q_2|^2
\end{aligned}$$

For the purposes of interoperation with vectors in $\mathbb{R}^3$, a vector may be thought of as a purely imaginary quaternion with a zero scalar part. Similarly, a scalar may be thought of as real quaternion with a zero imaginary vector part.

3.2.3 Orientation Representation

For rotations in two dimensions, it is possible to use just complex multiplication by $e^{i\theta}$. For quaternions, however, multiplication of a purely imaginary quaternion by another nontrivial quaternion, in general, does not yield a purely imaginary quaternion (that is, a vector). However, the scalar part of the resulting quaternion can be canceled by post-multiplying by the quaternion inverse. So,

$$v' = qvq^{-1}$$

is purely imaginary. Further, if q is a unit quaternion (that is, $|q|^2 = 1$) then the inverse is just the conjugate, and the norm of the resulting vector is the same as the norm of the original vector:

$$|v'|^2 = |qvq^*|^2 = |q|^2|v|^2|q^*|^2 = |v|^2$$

This is actually a requirement, because vector length is invariant under rotation.

According to Euler, any sequence of rotations about different axes can be described by a single rotation about an inclined axis. The quaternion representing a rotation of θ about an axis $\hat{u}$ is,

$$q = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{u}\right)$$

To apply the rotation represented by the quaternion q to the vector v, simply compute, $v' = qvq^*$. This operation can be thought of as a rotation in four dimensions followed by a counter-rotation that brings the vector back into $\mathbb{R}^3$ [8]. Every rotation has two representations in quaternion space: $v' = qvq^* = (-q)v(-q^*)$.

3.2.4 Quaternion Calculus

One useful feature of using quaternions to represent orientation is that it can be easily differentiated. In order to track orientation with an inertial sensor, it is necessary to determine the derivative of orientation, $\dot{q}$, in terms of the body rates (p, q, r). For

small θ,

$$\cos\left(\frac{\theta}{2}\right) \approx 1, \qquad \sin\left(\frac{\theta}{2}\right) \approx \frac{\theta}{2}$$

so,

$$q = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{u}\right) \approx \left(1, \hat{u}\frac{\theta}{2}\right)$$

Taking differentials,

$$dq = \left(0, \frac{1}{2}\hat{u}\dot{\theta}dt\right)$$

where $\hat{u}\dot{\theta}$ represents the angular rate of $\dot{\theta}$ about the axis $\hat{u}$, therefore,

$$\begin{aligned} \hat{u}\dot{\theta} &= (p, q, r) \\ \dot{q} &= \frac{1}{2}(0, p, q, r) = \frac{1}{2}B_\omega \end{aligned}$$

If q_1 is an initial orientation (in world coordinates), and q_2 is a second rotation in body coordinates, then the net rotation is $q_3 = q_1 q_2$, so

$$\dot{q}_3 = \dot{q}_1 q_2 + q_1 \dot{q}_2 = q_1 \dot{q}_2 = \frac{1}{2} q_1 B_\omega$$

In general [5],

$$\dot{q} = \frac{1}{2} q B_\omega$$

Using Euler integration, this formula yields a smooth rotation and avoids the "branch cut" problem of Euler angles. After every complete rotation, the quaternion will switch to its negative [8].

3.3 Reasons for Using Quaternions

There are both advantages and disadvantages to using quaternions. First, there is no singularity associated with quaternions; every orientation can be tracked. Second, quaternions provide a smooth tracking, avoiding the "branch cut" problem of Euler angles. Furthermore, it takes fewer operations to multiply two quaternions than it does to multiply two 3x3 matrices. Finally, quaternions can be computed directly from measured values without needing to compute trigonometric functions. Numerical truncation may cause the product of many orthonormal matrices not to be orthonormal, and although the product of many unit quaternions may not be a unit quaternion, it is trivial to find the nearest unit quaternion, but quite difficult to find the nearest orthonormal rotation matrix [8].

There are also several difficulties with quaternions, but most only arise in computer graphics situations. First, a rotation of 0 is very different from a rotation of 2π or 4π, however these are all represented by the same quaternion. Second, quaternion rotation is isotropic, and interpolation depends only upon the relation between the initial and final quaternions. If one were interpolating a camera position, it would be desirable to have the camera always upright, but this notion does not easily fit into quaternion algebra. For the purposes of an activity monitor, however, quaternions are superior to Euler Angles.

Chapter 4

Attitude Estimation Filter

The attitude estimation filter has three main parts: the Orientation Estimator, the Displacement Estimator, and the Bias Estimator.

4.1 Orientation Estimator

The Orientation Estimator attempts to maintain a quaternion describing the heading of the device. It accomplishes this by performing Euler integration on the estimated rate quaternion and comparing that against the estimated acceleration vector (see Figure 4-1).

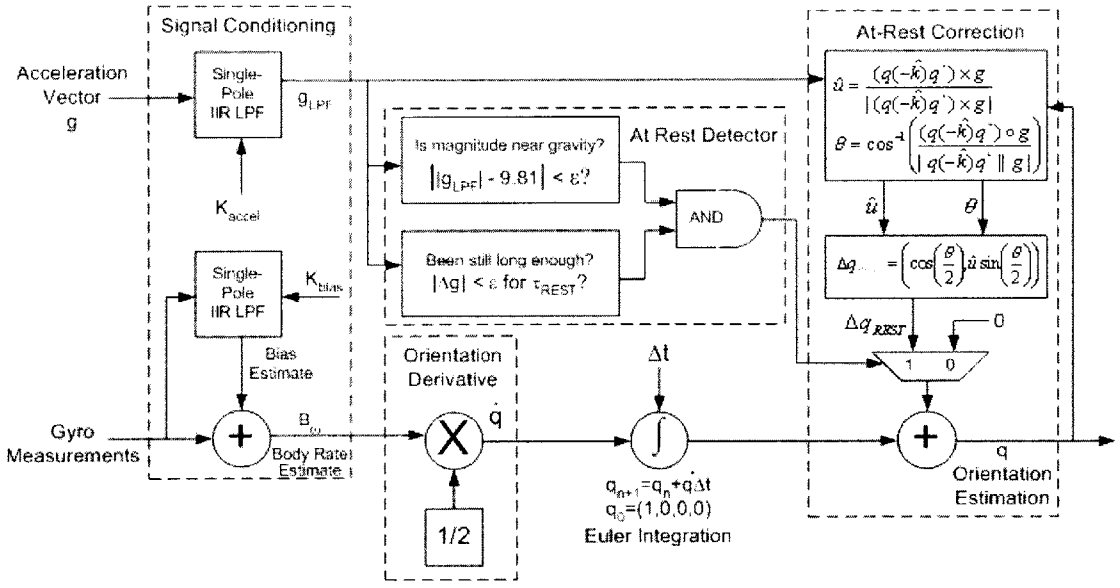
The conditioned output of the angular rate gyros, $B_\omega = (0, p, q, r)$, is integrated into the existing orientation estimate via the following update rule:

$$q_{n+1} = q_n + \frac{1}{2} q B_\omega \Delta t$$

where $\frac{1}{2} q B_\omega$ is the derivative of orientation, $\dot{q}$. The new orientation quaternion must then be normailzed to a unit quaternion.

If the magnitude of the acceleration vector is equal to the acceleration due to gravity, and it has been stable for a specified period of time, τ_{rest}, then the filter assumes that the device is at rest, and updates the orientation quaternion using the direction of the acceleration vector as "down." To accomplish this, the estimator

Figure 4-1: Block Diagram of the Orientation Estimator



constructs a correction quaternion which represents a rotation equal to the angle between the current heading, $q(-\hat{k})q^*$, and the estimated acceleration direction, g, about an axis perpendicular to both:

$$\begin{aligned} \Delta q_{rest} &= \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{u}\right), \text{ where} \\ \hat{u} &= \frac{(q(-\hat{k})q^*) \times g}{|(q-(\hat{k})q^*) \times g|} \\ \theta &= \arccos\left(\frac{g \bullet (q(-\hat{k})q^*)}{(|g||q(-\hat{k})q^*|}\right) \end{aligned}$$

where $\hat{k} = (0,0,1)^{\mathrm{T}}$. The correction quaternion is then incorporated into the orientation estimate via the relation $q = \Delta q_{rest} q_{estimated}$.

To simplify the computation, $\sin\left(\frac{\theta}{2}\right)$ and $\cos\left(\frac{\theta}{2}\right)$ are computed using the half-angle identities,

$$\sin\left(\frac{\theta}{2}\right) = \sqrt{\frac{1-\cos(\theta)}{2}}$$

$$\cos\left(\frac{\theta}{2}\right) = \sqrt{\frac{1+\cos(\theta)}{2}}$$

4.2 Displacement Estimator

The Displacement Estimator maintains the current position and velocity of the device in world coordinates, p. When the device is considered moving, the estimator rotates the measured acceleration vector into world coordinates, and subtracts the acceleration due to gravity:

$$a = q^* g q - \begin{pmatrix} 0 \\ 0 \\ -9.8066\frac{m}{s^2} \end{pmatrix}$$

It then updates the velocity and position vectors using Euler integration:

$$v_{n+1} = v_n + a\Delta t$$

$$p_{n+1} = p_n + v\Delta t$$

4.3 Bias Estimator

The output of the angular rate sensors is an analog voltage between 0 and 5 volts. When the device is stationary, each gyroscope reads about 2.5 volts, but this value is device specific, and can vary with temperature. Because this variation in the bias voltage occurs slowly, the bias estimator averages the first few hundred samples when the device is calibrated, and then tracks the bias over time with a very long time constant low-pass filter. The filter is implemented as a single pole IIR filter with a time constant, k_{bias}, that can be adjusted based on operating conditions and expected maneuver noise. Each of the gyroscopes has an onboard temperature sensor which can

also be used to adjust the bias estimate based on empirically measured temperature dependence curves.

Chapter 5

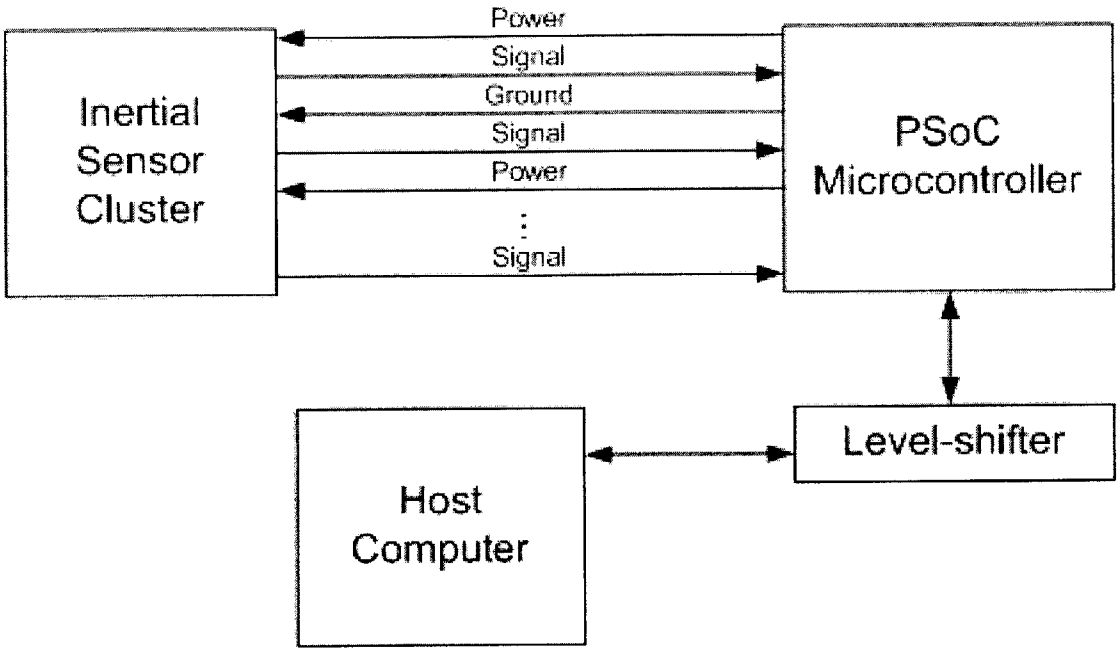
Hardware Design

The physical device measures accelerations and rotations, and sends them to a microcontroller for initial processing. The microcontroller then constructs communication packets and sends them to the host computer, which performs logging and final processing. Figure 5-1 shows an overview of the signal flow. The communication link between the microcontroller and the host computer is currently configured as RS-232 serial, but can, with a little effort, also be configured as USB, I2C or IrDA.

5.1 Mechanical Design

The physical sensor uses two Analog Devices Dual-Axis iMEMS®high-precision accelerometers (ADXL203), and two Analog Devices iMEMS®angular rate sensors (ADXRS150) mounted in a clear acrylic form. The accelerometers are capable of measuring accelerations of $\pm 1.7g$ at a resolution of $1\text{m}g$, and the gyroscopes are capable of measuring rotation rates of up to 150 degrees/s at a resolution of 2.22 arcminutes per second. Each sensor is mounted on a dedicated PCB provided by Analog Devices containing passive components for signal conditioning and power filtration. The form is constructed from 3/8" and 1/8" cast acrylic cut to fit the PCBs on a 120W laser cutter, and assembled with machine screws (see Figure 5-2 and Figure B-1). A 20-conductor ribbon cable supplies power to the sensor and returns the output signals.

Figure 5-1: Signal flow block diagram

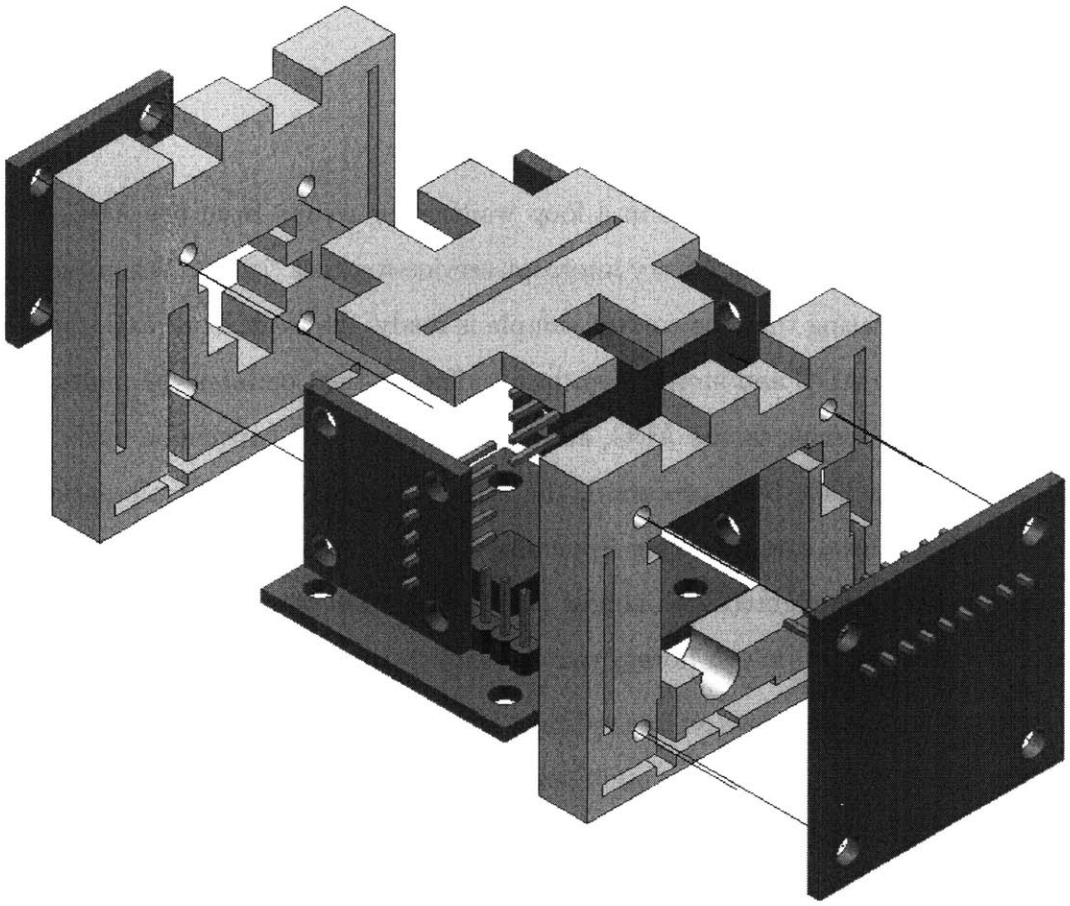


5.2 Electrical Design

The sensor has nine analog outputs: three accelerometer axes (the fourth redundant axis is unused), three gyroscope axes, and three temperature sensors - one for each gyroscope. There are five redundant power lines and five redundant ground lines interleaved between the nine signal lines to improve noise immunity and reduce cross-talk (see Figure 5-1).

The microcontroller used to capture the signals from the sensors and perform low-level processing is a Cypress Microsystems Programmable System on a Chip (PSoC). The PSoC used (CY8C27443) is a mixed signal array that includes an 8-bit Harvard architecture MCU, 8 reconfigurable digital blocks and 8 reconfigurable analog blocks. The analog blocks are configured as two 4-to-1 analog multiplexers, and a triple-input 13-bit integrating analog-to-digital converter (ADC). The two analog multiplexers each read four input signals and sequentially present them to two of the ADC inputs. Because of on-chip routing constraints, the ninth signal must be fed directly into the third input of the ADC. The digital blocks have been configured as three 8-

Figure 5-2: CAD model of the inertial sensor cluster



bit counters (one for each ADC input), one 16-bit pulse-width modulator for ADC timing, and one full-duplex UART for serial communication with the host computer (see Figure 5-3). The analog signals enter at the lower left side of the diagram and enter the analog multiplexers and switched-capacitor blocks that make up the ADC. The top set of blocks contains the counters for the ADC, its PWM time base, and the UART in the lower right. The UART uses an external MAX233CPP level-shifter from Maxim to convert its TTL serial signals to/from RS-232.

5.3 Firmware

The firmware on the PSoC sits in a loop waiting for results from the ADC, which does all of its processing in library interrupt-service-routines (ISRs). When the ADC sets a flag indicating that the current sample is ready, the main loop reads the three values from the ADC and stores them in a buffer. It then updates the multiplexers to present new signals to the ADC, and starts a new ADC conversion cycle. Once all nine samples have been processed, the firmware constructs a sample packet and sends it to the host computer (see Appendix C). The major benefit of using a PSoC is that it can be configured with almost arbitrary combinations of peripherals, which are all controlled by built-in library routines called from C.

Because the system uses an integrating ADC, the conversion time varies depending on the signal level. When the input voltage is +5V, conversion takes 14,972 CPU cycles, but when the input voltage is 0 volts, conversion takes only 7,868 CPU cycles. In order to provide a time base, the PSoC's sleep timer has been used to increment a counter value at 512 Hz. This counter is then included in the sample packet. To achieve the maximum sample rate, the main loop sends sample packets as soon as they are ready. With the MCU running at 24 MHz, the PSoC sends about 30 sample packets per second. This is adequate because analyses of human motion have shown that the dominant spectral power of normal human movement is limited to about 10 Hz [2].

Figure 5-3: On-chip signal flow diagram for PSoC microcontroller

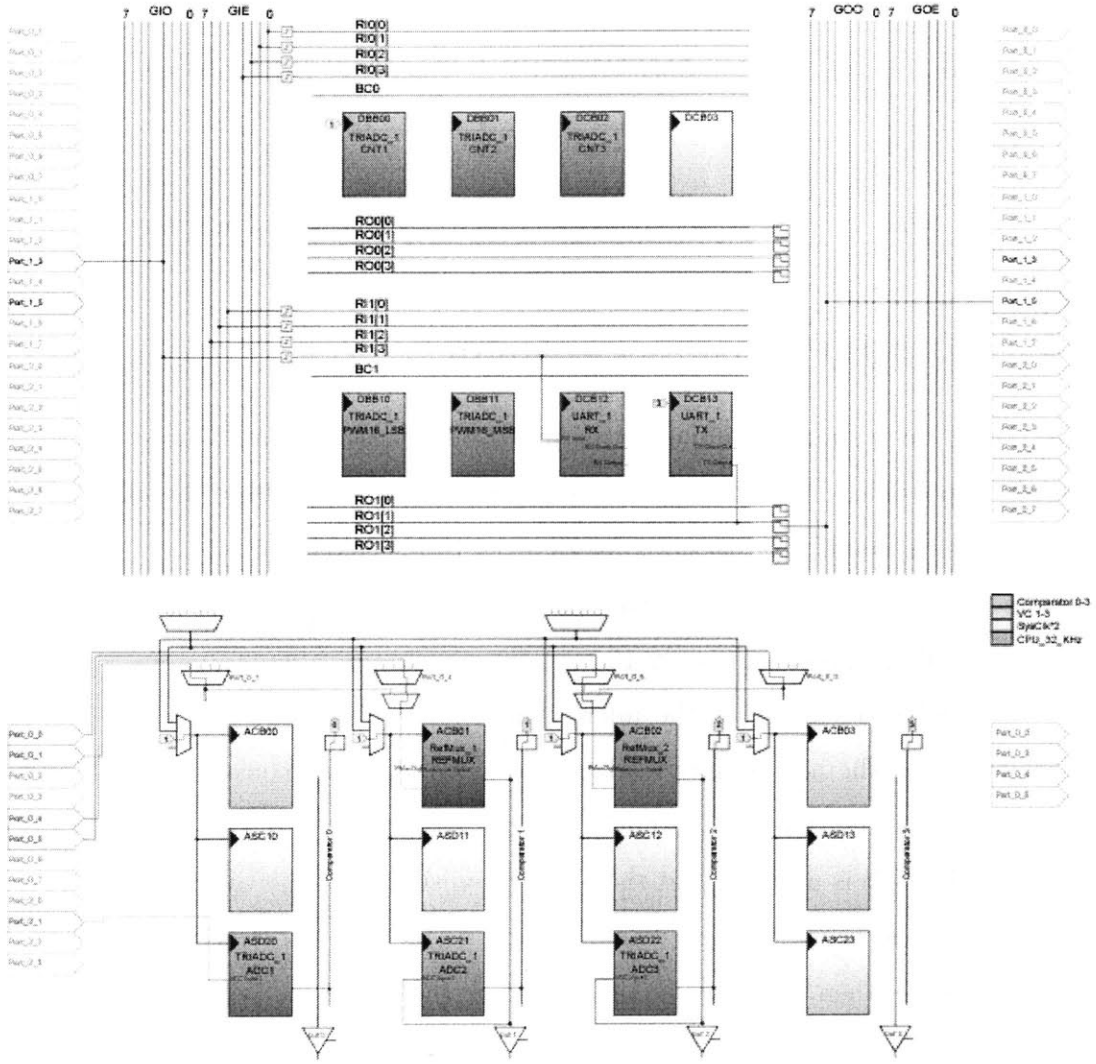


Table 5.1: Sample Packet Format

Field	Size	Values	Description
Time	2	0x0000-0xFFFF	512 Hz Oscillator count
Yaw	2	0x0000-0x1FFF	Yaw rate
Z-Accel	2	0x0000-0x1FFF	Acceleration along z-axis
Y-Accel	2	0x0000-0x1FFF	Acceleration along y-axis
Temp (Pitch)	2	0x0000-0x1FFF	Temperature of pitch sensor
Roll	2	0x0000-0x1FFF	Roll rate
Temp (Roll)	2	0x0000-0x1FFF	Temperature of roll sensor
X-Accel	2	0x0000-0x1FFF	Acceleration along x-axis
Pitch	2	0x0000-0x1FFF	Pitch rate
Temp (Yaw)	2	0x0000-0x1FFF	Temperature of yaw sensor
Sync	2	0xFFFF	Synchronization word

5.4 Communication Protocol

The system uses a simple communication protocol, consisting of a 22-byte packet sent as often as possible. Therefore, the host computer must be capable of handling about 660 bytes per second. The packet consists of 11 16-bit fields (see 5.1).

The software must detect when the time counter overflows and behave accordingly. Because the data samples are only 13-bits, the maximum value of any measurement field is 0x1FFF. This way, if the micro controller and the host computer become unsynchronized, the host simply reads single bytes until two consecutive bytes read 0xFF. This ensures that the host will resynchronize with a data loss of only two sample packets. It is unlikely that the host will misinterpret the timer field as a sync field because the host notices a framing error when the sync field is too small. If that is the case, then the host should encounter the next sync field before the next timer field. If the host does manage to resynchronize on the timer field, then the next packet will immediately flag another framing error, with only one corrupted packet.

The MATLAB scripts used to log and process the sample packets may be found in Appendix C.

Chapter 6

Results

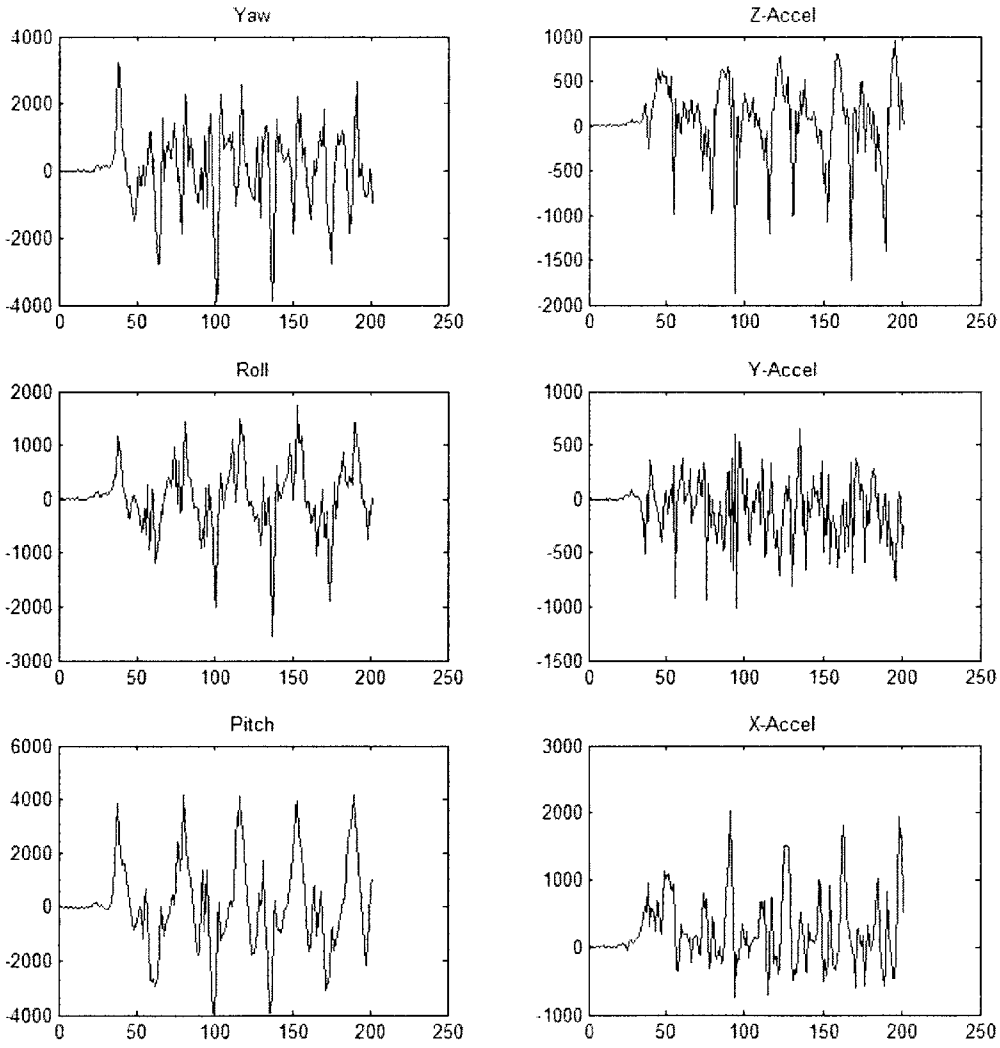
6.1 Design Goals

The design goals of the device were that it should be portable, low-cost, low-power, and provide a rich data stream. The sensor cluster itself is a cube approximately 1.5" on a side, making it about half the size of an ordinary cellular telephone. The microcontroller, level-shifter and interconnect occupy about six square inches of PCB surface, which would fit easily onto the back of a PDA or handheld computer. The device clearly meets the portability design goal.

The gyroscopes are the most expensive piece of hardware, costing about $30 each, followed by the accelerometers, at about $12 each. The microcontroller and level-shifter together cost about $10, and the cost of the passive components and structural materials is vanishingly small. All of the prices listed assume quantities of one, so the total cost, in parts, of the device is less than $125. Other commercial sensors sell for not less than $500, some much more, so the device also meets the low-cost requirement.

The microcontroller consumes most of the power used by the device. The accelerometers each draw less than 1mA at 5V, and the gyroscopes each draw about 6mA at 5V. The total current consumption at 5V was measured as 65mA, meaning that the total power usage is about 1/3W, less than the display on most handheld computers.

Figure 6-1: Raw traces of a person walking



Finally, the data rate is high enough to capture every-day human movement. Because the spectral power of human movement is band-limited to below about 10 Hz, the Nyquist rate for sampling human motion is 20 Hz. The activity monitor samples at 30 Hz.

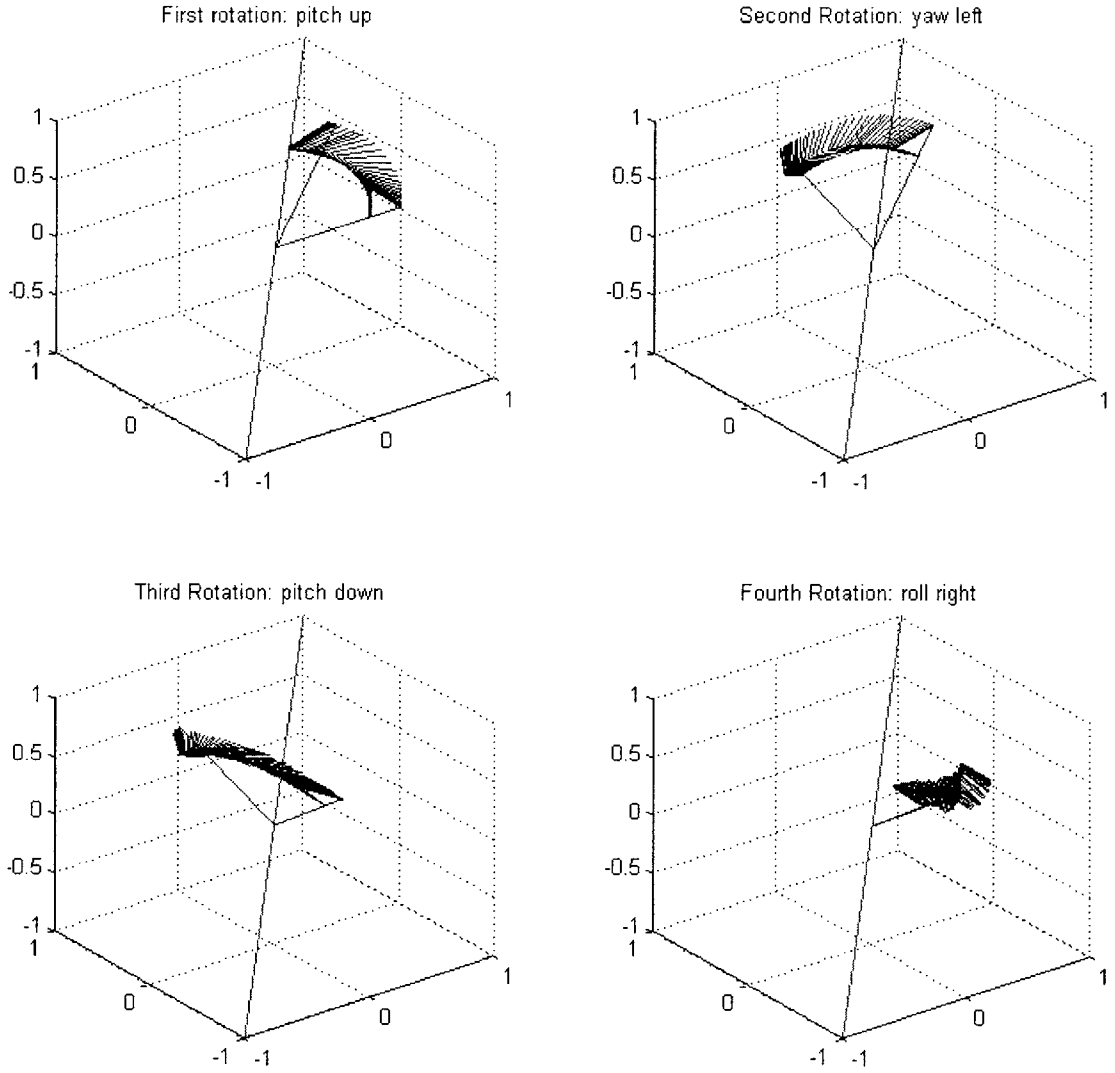
Figure 6-1 shows the raw traces (with gravity subtracted) from a person walking. By examining the pitch rate, one can easily see that the figure depicts five steps, and by noting that four steps occur over about 150 samples, one can conclude that the step period is just longer than a second, meaning that this was a slow, deliberate walk. The rounded tops and the sharp bottoms of the z-axis accelerometer trace hint at the impact force of the foot falls. Clearly the monitor provides a rich data set.

6.2 Attitude Estimation

The relatively simple design of the attitude estimation filter worked fairly well for maintaining relative orientation, but poorly for maintaining relative position. Small errors in orientation were corrected by the direction of the gravity vector when the device was at rest, however even very small errors in alignment between the estimated gravity vector and the true gravity vector caused the position estimate to diverge rapidly. Perhaps a more complex filter, like that in [10], and an explicit, empirical model of human dynamics would improve performance of the position estimator.

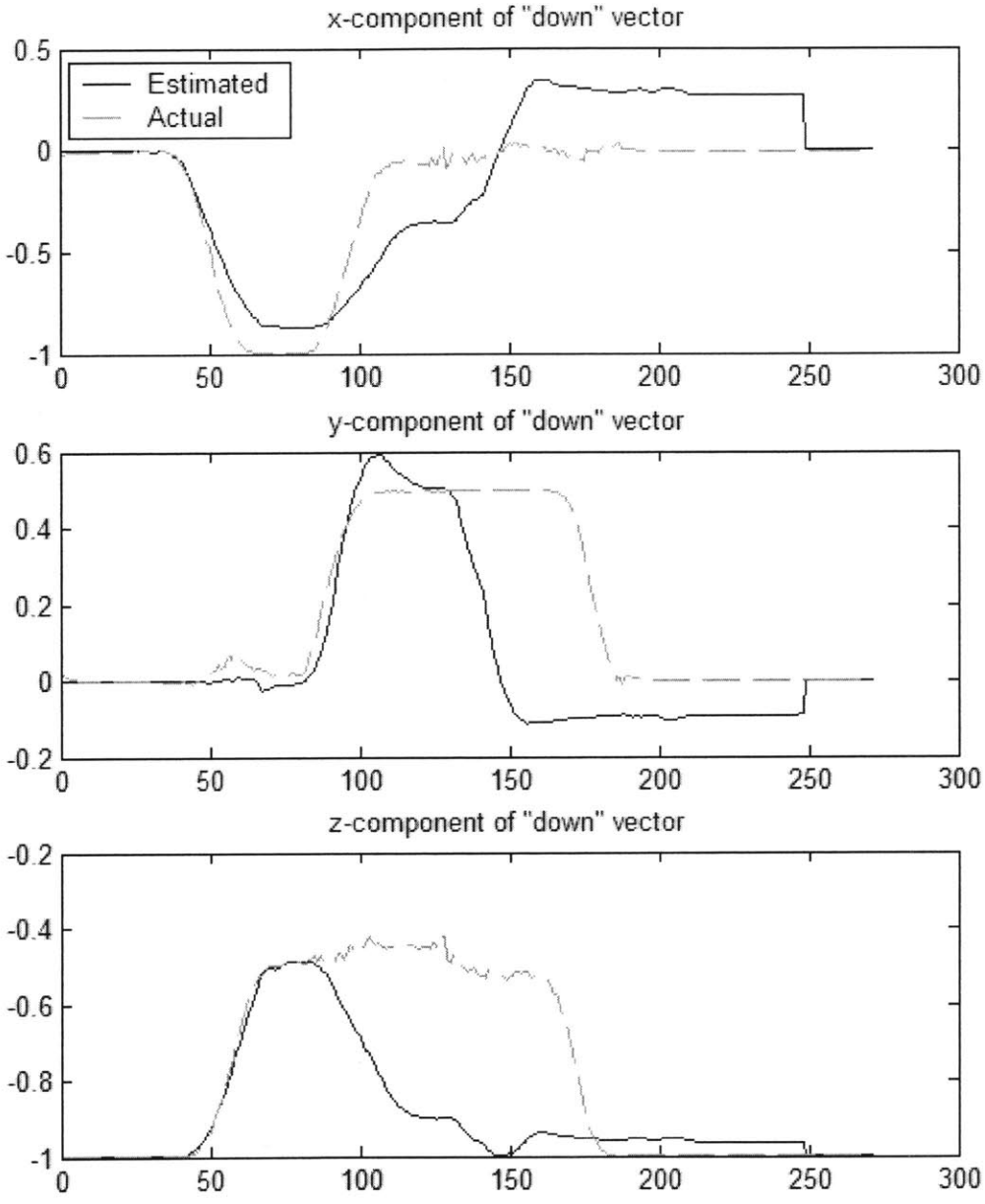
Figure 6-2 shows the estimated attitude as the device was moved through four rotations, returning it to its starting orientation. The figure depicts arrows pointing in the directions that the device was facing at each sample point, with the second point of the arrow indicating the "top" of the device. The main shafts of the arrows have been removed (except the first and last in each series) for clarity. The rotations were made by hand, so the axes of rotation do not coincide exactly with the coordinate axes, and the angles are not all 90 degrees. Notice that near the end of the fourth rotation, the trail moves sharply upward. This is the corrective action of the accelerometers when the device comes to rest. Because it was set down at the end, the final orientation is very close to the initial orientation. Figure 6-3 shows the same sequence of motions

Figure 6-2: Traces of the estimated orientation as the device moves through four consecutive rotations.



represented as the components of a "down" vector. That is, with one degree-of-freedom removed from the quaternion. Notice again the corrective action toward the end of the trace.

Figure 6-3: Traces of the estimated "down" vector as the device moves through four consecutive rotations.



Chapter 7

Conclusion

7.1 Applications and Future Work

There are many applications for a low-cost, low-power, six degree-of-freedom activity monitor. By building models of both normal and abnormal human movement, doctors can examine movement data from patients in a non-laboratory setting and determine if abnormalities exist, their severity, and their likely causes. By analyzing a patient's gait, for example, doctors can see how a patient is recovering from knee or hip replacement surgery, or if the patient has a ruptured disk, causing him or her to stoop or limp. After diagnosis, the doctor can monitor the data to determine the extent to which the patient is responding to treatment.

Patients' movement patterns in a laboratory setting will likely differ from those in a normal home setting. Indeed, many patients, especially elderly ones, may not notice these differences, and so will be unable to report them to their doctor. This 6-DOF activity monitor provides a low-cost solution to the problem of inaccurate or incomplete patient reporting. In addition to post processing of the data, doctors can drop in filters on the host computer that will screen for pathological movement patterns that the patient should avoid. Such filters can provide immediate feedback to the patient about ways he or she should move for treatment of a particular disorder, or simply warn if the patient seems in danger of falling.

At a higher level, by using more sophisticated attitude estimation filters, like those

described in [3, 10], and by integrating the monitor with GPS and/or a few acoustic beacons, it may be possible for doctors to recognize lifestyle patterns and suggest changes. For example, by examining the sensor data, it may become apparent that a patient is spending an excessive amount of time standing in a particular place, like a kitchen sink, or sitting in a particular chair, or going up and down a set of steps many times. Patients used to living in a particular way, especially elderly patients, may not realize that some of their health problems stem from unhealthy lifestyle patterns.

7.2 Contributions

With this project, I have

- designed and constructed an inertial six degree-of-freedom activity monitor,
- designed and constructed an interface for the monitor that can communicate over serial, USB, and I2C,
- ensured ease of use and portability by using parts available in small packages,
- provided a low-cost solution consisting of total parts costs under $200,
- produced a low-power device consuming under 70mA at 5V,
- implemented a relatively simple real-time attitude estimation filter for use with the monitor,
- provided a qualitative analysis of the filter's performance.

It is clear that, for the purposes of human tracking, better models of human motion, more sophisticated filter algorithms, and additional modes of tracking are needed to achieve long term tracking performance and maximum patient benefit. Nonetheless, the existing system provides a rich data stream, a flexible interface, and a cost-effective platform on which to base future research.

Appendix A

Photos

Figure A-1: Sensor Cluster

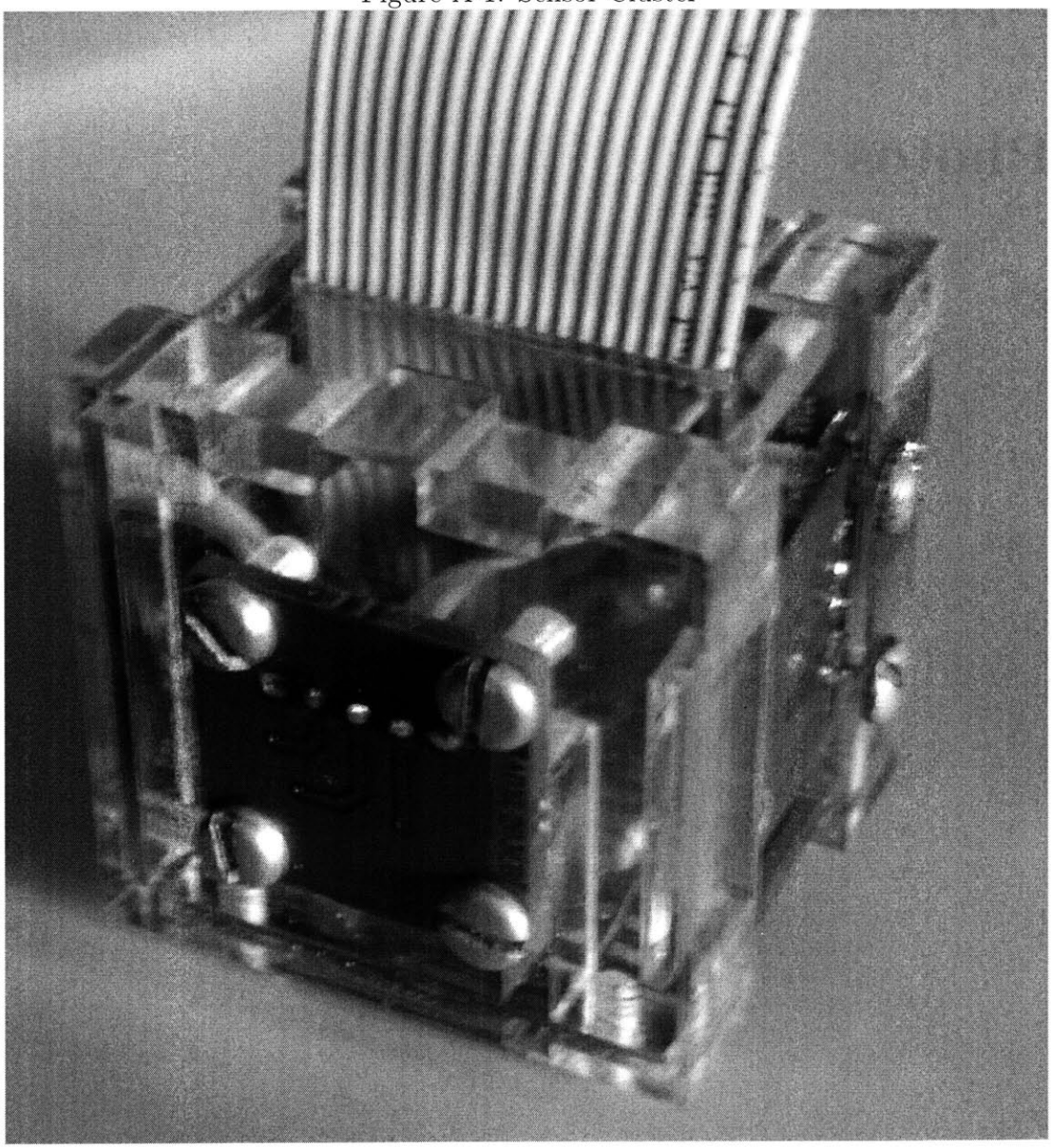
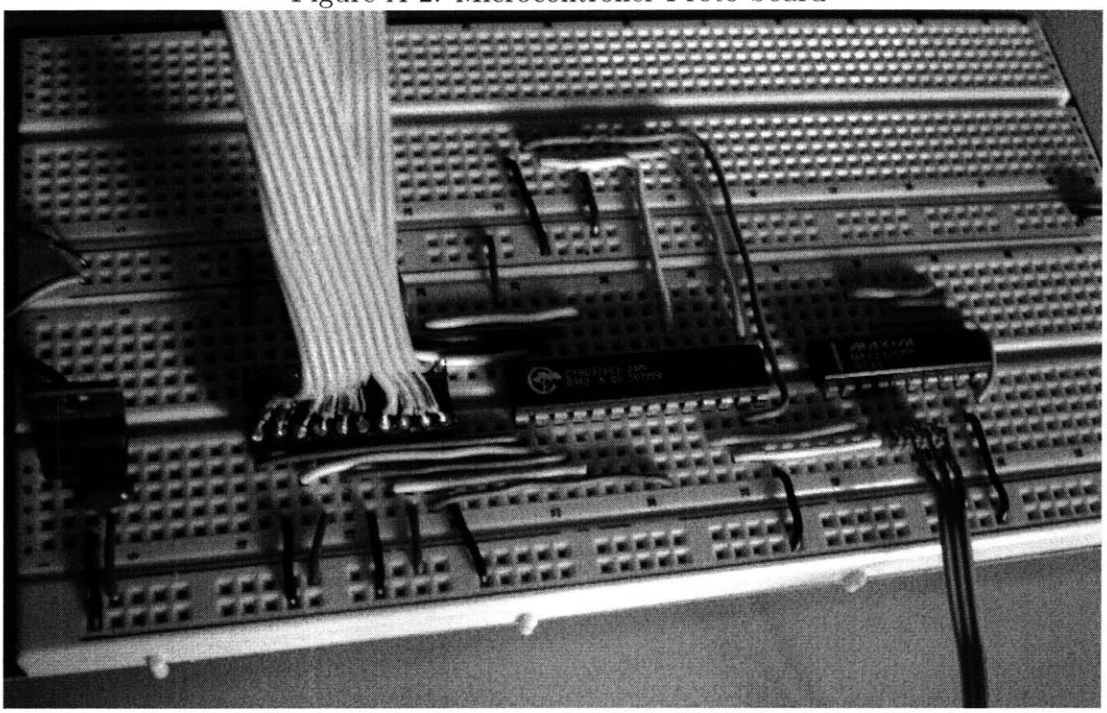


Figure A-2: Microcontroller Proto-board



Appendix B

Drawings

Figure B-1: Sensor Mounting Tool Paths

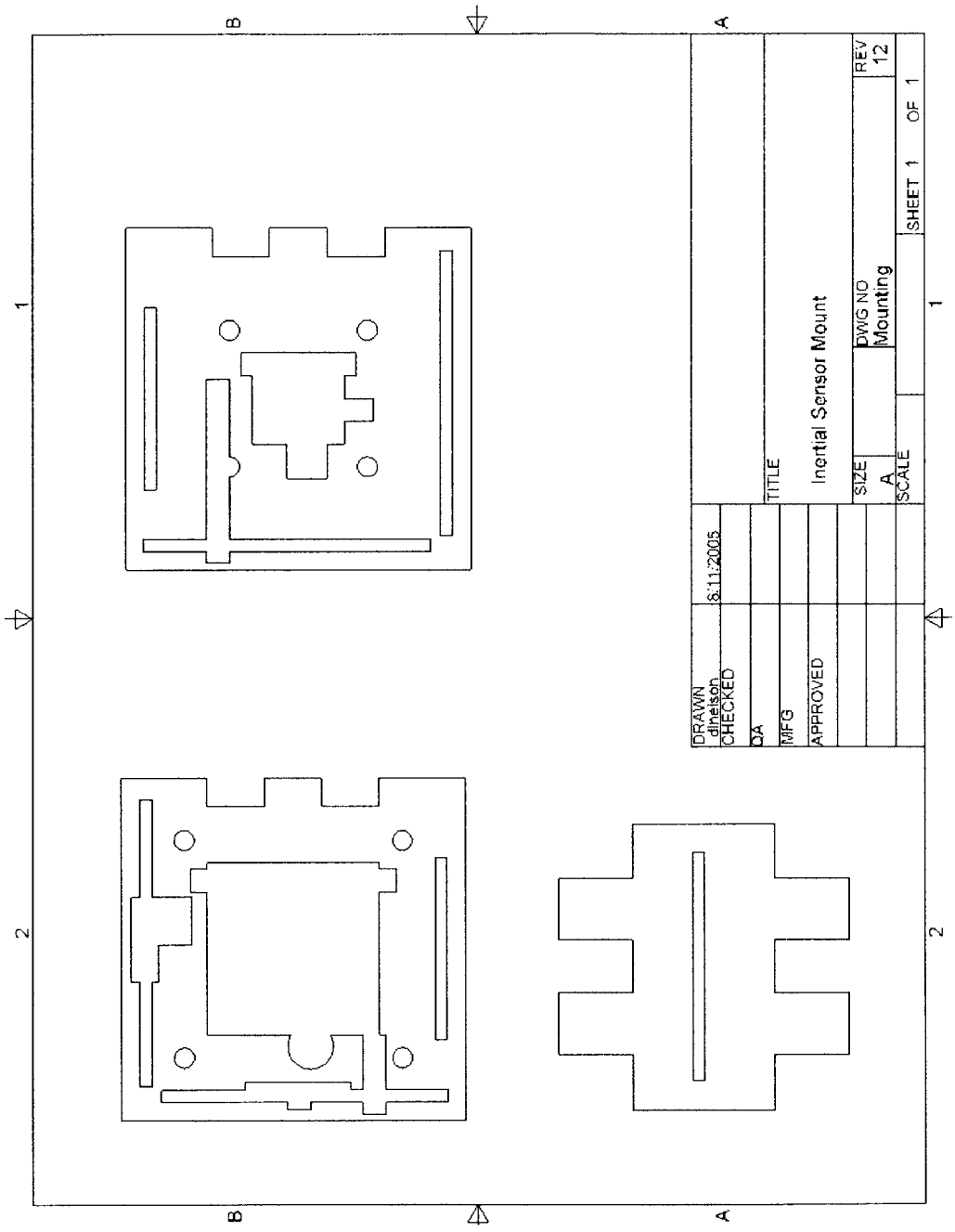
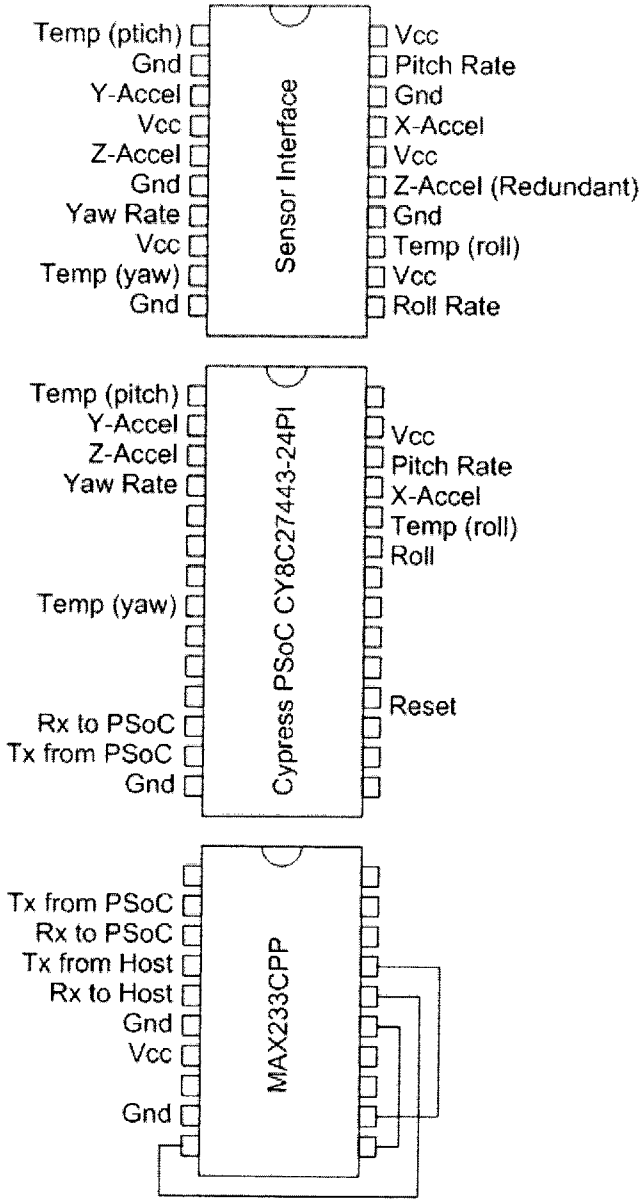


Figure B-2: Device Pinout and Schematic



Appendix C

Code

C.1 Firmware: main.c

```
//-------------------------------------------------------
// C main line
//-------------------------------------------------------

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

INT sample[9];
extern unsigned int  sleepcounter;

void init() {
//      UART_1_DisableInt();
        UART_1_Start(UART_1_PARITY_NONE);

        M8C_EnableGInt;

        RefMux_1_Start(RefMux_1_HIGHPOWER);  // Only necessary if selecting AGND
        RefMux_1_RefSelect(RefMux_1_PMUXOUT);

        RefMux_2_Start(RefMux_2_HIGHPOWER);  // Only necessary if selecting AGND
        RefMux_2_RefSelect(RefMux_2_PMUXOUT);
```

```
        AMUX4_1_Start();  // Unnecessary
        AMUX4_1_InputSelect(0);

        AMUX4_2_Start();  // Unnecessary
        AMUX4_2_InputSelect(0);

        TRIADC_1_SetResolution(13);
        TRIADC_1_Start(TRIADC_1_HIGHPOWER);
        TRIADC_1_GetSamples(0);
}

void clear_sample() {
        BYTE i;
        for(i = 0; i < 9; i++) {
                sample[i] = 0x8000;
        }
        //sleepcounter = 0;
}

BYTE check_sample_ready() {
        BYTE i;
        for(i = 0; i < 9; i++) {
                if(sample[i] == 0x8000) return 0;
        }
        return 1;
}

void SendInt(INT i) { // Big Endian
        UART_1_SendData((BYTE)((i>>8) & 0xFF));
        while( !( UART_1_bReadTxStatus() & UART_1_TX_BUFFER_EMPTY ) );
        UART_1_SendData((BYTE)(i & 0xFF));
        while( !( UART_1_bReadTxStatus() & UART_1_TX_BUFFER_EMPTY ) );
}

void send_sample_ascii() {
```

```
        BYTE i;
        UART_1_PutSHexInt(sleepcounter);
        UART_1_PutChar(' ');
        for(i = 0; i < 9; i++) {
                UART_1_PutSHexInt(sample[i]);
                UART_1_PutChar(' ');
        }
        UART_1_PutCRLF();
}

void send_sample() {
        BYTE i;
        SendInt(sleepcounter);
        for(i = 0; i < 9; i++) {
                SendInt(sample[i]);
        }
        SendInt(0xFFFF);
}

void main()
{
        INT data;

        BYTE port1 = 0;
        BYTE port2 = 0;

        init();
        clear_sample();
        sleepcounter = 0;
        INT_MSK0|=0x40; // enable sleep interrupts. 512Hz clock.

        for(;;) {

                if(TRIADC_1_fIsDataAvailable()) {
                        data =  (TRIADC_1_iGetData3() & 0x1FFF);
                        sample[port1] = data;
```

```
                port1 = (port1+1) % 4;
                AMUX4_1_InputSelect(port1);
                data =  (TRIADC_1_iGetData2() & 0x1FFF);
                sample[4+port2] = data;
                port2 = (port2+1) % 4;
                AMUX4_2_InputSelect(port2);
                data =  (TRIADC_1_iGetData1() & 0x1FFF);                    100
                sample[8] = data;
                TRIADC_1_ClearFlag();
            }

            if(check_sample_ready()) {
                send_sample();
                clear_sample();
            }
        }
                                                                            110
}
```

C.2 Filter: capture.m

```
% Capture.m

% Set-up COM Port
s = serial('COM6');    % Replace with actual port
s.BaudRate = 115200;
s.ByteOrder = 'bigEndian';
s.BytesAvailableFcn = @process_data;
s.BytesAvailableFcnCount = 22;
s.BytesAvailableFcnMode = 'byte';
s.InputBufferSize = 64;

% Clean up workspace
clear global A;
clear global q;

% Declare globals
global A;
global o_save;  %saved arrows
global q;
global vel;
global pos;
global arrow;
global w_bias;
global t_old;
global g_old;
global lpf_accel;
global still_count;

% Initialize globals
A = zeros(11,1);
t = 0;
q = [0 0 0 1]';
vel = [0 0 0]';
pos = [0 0 0]';
```

```
arrow = [0 1 .75 .75; 0 0 0 0; 0 0 .25 0];
w_bias = [0 0 0 -1]';  % -1 used for calibration counter
t_old = 0;
g_old = 0;
lpf_accel = [0 0 0]';
still_count = 0;                                                    40
o_save(:,:,1) = arrow;

% Set up the display
set(gcf, 'renderer', 'opengl');
pause(.05)

% Open the serial port
fopen(s);

% Go!                                                               50
% Don't forget to fclose(s) when you're done
```

C.3 Filter: process_data.m

```
function process_data(obj, event)

global A;
global o_save;
global q;
global vel;
global pos;
global arrow;
global w_bias;
global t_old;
global g_old;
global lpf_accel;
global still_count;

tau_rest = 30;
k_bias = 0.9995;
k_accel = .7;
a_x_offset = 0;
a_y_offset = -0.0575;
a_z_offset = -0.0400;
a_x_scale = 2.1280;
a_y_scale = 2.1045;
a_z_scale = 2.0900;

% Read packet
v = fread(obj,11,'uint16');

% Check for sync word
if v(11) == 65535

    % Re-order fields for convenience
    v = [v(1); v(2); v(6); v(9); v(3); v(4); v(8); v(5); v(7); v(10); v(11)];
    %   time   gyro  gyro  gyro  accel accel accel temp  temp  temp   sync
    %   512Hz  Yaw   Roll  Pitch   Z     Y     X   Pitch Roll  Yaw    FFFF
```

```
% Append packet to log
A(:,end+1)=v;

% If we're in calibration mode,
if w_bias(4) < 0

    %Accumulate gyro and accel data
    w_bias = w_bias + [v(3)  v(4) v(2) 0]';

    g = [v(7)-4096 v(6)-4096 v(5)-4096]'*0.0012207;
    g = (g + [a_x_offset a_y_offset a_z_offset]')./[a_x_scale a_y_scale a_z_scale]';
    lpf_accel = lpf_accel + g;

    % Update scratch counter variable
    w_bias(4) = w_bias(4) - 1;

    % If this is the last iteration,
    if w_bias(4) < -1000

        % Reset counter for use
        w_bias(4) = 0;

        % Build averages
        w_bias = w_bias/1000/(1-k_bias);
        lpf_accel = lpf_accel/1000/(1-k_accel);

        % Initialize variables
        t_old = v(1);
    end

    % Status
    %   Reordered packet
    subplot(1,2,1), bar(v);
    %   Progress of accumulators (dona at about 2000)
    subplot(1,2,2), bar(w_bias*(1-k_bias));
```

```
    return;
end

% Calculate time base
if t_old > v(1)
    t_old = t_old - 65536;
end
delta_t = (v(1) - t_old)/512;
t_old = v(1);

% Condition accel data
%   Build vector
my_g = [v(7) v(6) v(5)]';
%   Adjust with empirical constants, and convert units to g's
g = [v(7)-4096 v(6)-4096 v(5)-4096]'*0.0012207;
g = (g + [a_x_offset a_y_offset a_z_offset]')./[a_x_scale a_y_scale a_z_scale]';
%   Update low-pass filter
lpf_accel = g+lpf_accel*k_accel;
g_lpf = lpf_accel*(1-k_accel);

% Condition gyro data
%   Build and adjust vector, convert units to rad/s
w = ([v(3)  v(4) v(2) 0]' - w_bias*(1-k_bias))*-0.07324/180*pi/2;
%   Update bias low-pass filter
w_bias = [v(3)  v(4) v(2) 0]' + w_bias*k_bias;

% Integrate quaternion derivative
q = qnorm(q + .5*qmult(q, w)*delta_t);

% Rotate arrow graphic for display
o = qvrot(q, arrow);
o_save(:,:,end+1) = o;

% Accelerometer Correction Step
%   Initialize variables
q_correct = [0 0 0 1];
```

```
%   Update still count if acceleration vector is stable
if norm(g_old - g) < .01
    still_count = still_count + 1;                                    110
else
    still_count = 0;
end
g_old = g;

%   Check if acceleration vector could be just gravity
if (.98 < norm(g_lpf)).*(norm(g_lpf) < 1.01).*(still_count > tau_rest)

    % Construct axis for correction rotation
    rotz = qvrot(conj(q), [0 0 -1]');                                 120
    u_hat = cross(rotz, g_lpf);
    u_hat = u_hat / norm(u_hat);

    % Get angle for correction rotation
    cos_theta = sum(g_lpf .* rotz) / norm(rotz) / norm(g_lpf);

    % Build correction quaternion
    q_correct = [u_hat'*sqrt((1-cos_theta)/2) sqrt((1+cos_theta)/2)]';

    % Apply correction quaternion                                     130
    q = qmult(q_correct, q);
else
    % Update velocity estimate
    vel = vel + qvrot(qconj(q), g_lpf)-[0 0 -1]'*delta_t*9.81;
end

% Update position estimate
pos = pos + vel*delta_t;

% Display                                                           140
%   Arrow indicating orientation estimate
subplot(2,2,2), plot3(o(1,:), o(2,:), o(3,:), [-1.1 1.1], [-1.1 1.1], [-1.1 1.1]);
```

```
    grid on;

    %   Velocity estimate
    subplot(2,2,3), plot3([0 vel(1)], [0 vel(2)], [0 vel(3)], [-1.1 1.1], [-1.1 1.1], [-1.1 1.1]);
    grid on;

    % Position extimate
    subplot(2,2,4), plot3([0 pos(1)], [0 pos(2)], [0 pos(3)], [-1.1 1.1], [-1.1 1.1], [-1.1 1.1]);
    grid on;
    drawnow;
else
    %disp 'Framing error.'
    b1 = (v(11) & 255);
    b0 = 0;
    while (b1~=255)+(b0~=255)
        b1 = b0;
        b0 = fread(obj, 1, 'uint8');
    end
    %disp 'Ok.'
end
```

Bibliography

[1] Analog Devices, Norwood, MA. *ADXL103/ADXL203: Low Cost 1.7g Single/Dual Axis Accelerometer Data Sheet, Rev. 0*, 2004.

[2] C. Angeloni, P. O. Riley, and D. E. Krebs. Frequency content of whole body gait kinematic data. *IEEE Transaction on Rehabilitation Engineering*, 2(1):40–46, March 1994.

[3] E.R. Bachmann, I. Duman, U.Y. Usta, R.B. McGhee, X.P. Yun, and M.J. Zyda. Orientation tracking for humans and robots using inertial sensors. In *International Symposium on Computational Intelligence in Robotics and Automation*, 1999.

[4] R. Davis, H. Shrobe, and P. Szolovits. What is a knowledge representation? *AI Magazine*, 14(1):17–33, 1993.

[5] Ildeniz Duman. Design, implementation, and testing of a real-time software system for a quaternion-based attitude estimation filter. Master's thesis, Naval Postgraduate School, Monterey, CA, March 1999.

[6] T. Ford, J. Hamilton amd M. Bobye, J. Morrison, and B. Kolak. Mems inertial on an rtk gps receiver: Integration options and test results. In *Proceesings of ION NTM '03*, Anaheim, CA, 2003.

[7] David Hoag. Apollo guidance and navigation considerations of apollo imu gimbal lock. MIT Instrumentation Laboratory Document E-1344, April 1963.

[8] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society*, 4(4):629–642, April 1987.

[9] H.J. Luinge. *Inertial Sensing of Human Movement.* PhD thesis, University of Twente, 2002.

[10] Joao Luís Marins, Xiaoping Yun, Eric R. Bachmann, Robert B. McGhee, and Michael J. Zyda. An extended kalman filter for quaternion based orientation estimation using marg sensors. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii, USA, October 2001.

[11] Nissanka B. Priyantha, Hari Balakrishnan, Erik Demaine, and Seth Teller. Anchor-free distributed localization in sensor networks. Technical Report 892, MIT Laboratory for Computer Science, April 2003.

[12] E. E. Sabelman. Accelerometric human body motion analysis using a wearable computer/recorder. Presentation to Larry Rubenstein, MD, GRECC Clinical Director, Sepulveda VAMC, in conjunction with RESNA 1999 Annual Conference, Long Beach, CA, June 1999.

[13] P. H. Veltink, H. J. Luinge, B. J. Kooi, C. T. M. Baten, P. Slycke, W. Olthuis, and P. Bergveld. The artificial vestibular system - design of a tri-axial inertial sensor system and its application in the study of human movement. In J. Duysens, B. Smits-Engelman, and H. Kingma, editors, *Control of Posture and Gait*, pages 894–899, Maastricht, 2001. Proc. ISPG Conf.