Abstract—The vision of augmented reality is to provide users with relevant information to supplement their normal interactions with their environment. This technology promises to fundamentally improve many common applications that currently require specific devices or cumbersome interactions (e.g., driving directions, searches for local information, information about individuals with whom we interact, etc.). To achieve this vision, however, there are numerous technical challenges that still need to be addressed. For this project, we are developing a system that can perform the key functionalities of an augmented-reality system: (1) sensing a user’s location and direction of view, (2) computing what to display in the user’s field of view, and (3) displaying the visual content without obstructing the user’s view of real objects.

We present a design concept that aims to surpass prior attempts in its simplicity, efficiency, and functionality, to create a truly real-time fusion of analog and digital worlds. Our design consists of a sensor unit, an Intel Atom Processor, and a goggle-based display. The sensor unit obtains the live position and movement data of the users and sends it to the processor via a USB cable. The data is then processed through an integrated GPU and projected through the goggle display.

I. INTRODUCTION

The vision of Augmented Reality (AR), despite all of its allure, has yet to truly be realized. The task of actually producing a wearable AR system, capable of creating an immersive 3-D environment, has proven an enormous challenge to engineers for decades. Based on our research, that challenge can essentially be broken into three primary elements: (1) sensing (2) computation and (3) display. An additional challenge is the choice of the physical apparatus, or medium, through which to display the images. Previous attempts at such designs have been bulky and cumbersome, and not at all practical for any of the aforementioned applications. A truly great design solution requires a lightweight display, comfortable enough to become an extension of the wearer’s body. A portable AR system should ideally be battery-powered, and have a minimal number of components.

Numerous potential applications of the technology exist, in both professional and consumer markets. Personal navigation, gaming, and interior design are just a few areas which could greatly benefit from a more “wearable” solution. Figure 1 shows examples of possible applications for the AR Goggles.

![Figure 1a: Example Application](image1a.png)

![Figure 1b: Example Application](image1b.png)
The design will need to be able to accurately determine the location and facial orientation of the user. Accurate registration and positioning of virtual objects in the real environment requires accurate tracking of the user’s head and sensing the locations of other objects in the environment. According to Azuma [2], “[t]he biggest single obstacle to building effective Augmented Reality systems is the requirement of accurate, long-range sensors and trackers that report the locations of the user and the surrounding objects in the environment.” All head-tracking systems require the combination of multiple sensors to form an accurate picture of the user. More sensors, however, does not ensure higher accuracy. More sensors provide more data, which increases the computational demands of the system. One challenge in designing the sensing subsystem is selecting the best sensors for the application, to give the most accurate estimate of state with the fewest possible components.

The speed of the sensing is another big challenge. Polling every sensor for data, every time, will slow down the system, and would not even be useful. We must determine how to correctly “weight” the relative importance of each sensor’s output at any given time, depending on the present and past state of the system, to form a “good enough” estimate of position and orientation. In other words, a good head-tracking system will be able to distinguish the necessary data from the extraneous. Extraneous data can either mean noisy data, or simply unnecessary data. Some physical attributes, like head tilt, can change quite rapidly, whereas others, such as ambient temperature may not. One would, therefore sample a gyroscope’s measurements much more frequently than a thermometer’s due to one’s inherent instability.

Every system will also have noise. Care must be taken to ensure that all inherent noise, as well as drift (if applicable) is accounted for when filtering the raw binary data provided by the digital head-tracking sensors. Even measurement itself carries inherent noise, which increases with the sampling frequency. The sampling rate of the sensors must be only as high as the computational challenge demands, so as to ensure that the data being received are sufficiently accurate for the system to perform the necessary computation. One part of the “weighting” will be achieved by accurately determining how reliable the sensors are.

The display brings challenges of its own. The display must contain electronics which acquire and process a digital signal, an optics solution for combining light of different wavelengths (different colors), and a method for placing the individual pixels into the display. Aligning virtual and real objects, as well as creating an accurate sense of depth, are two areas where much attention has been devoted in the past (see Drascic and Milgram, [4]). Related to depth is the issue of focus: not only must virtual objects appear correctly sized to the user, they must also be focused at the correct distance. If an object appears blurry when it is only ten feet from the user, and likewise if a faraway object appears too sharp, these will not match the user’s human expectations of perception, and will detract from the immersive experience. The contrast and brightness of real vs. virtual objects must not be so drastically different as to distract the user. Ideally, all visual characteristics of the displayed objects would be matched to those of the real world.

II. DESIGN

A. System Overview

The design tradeoffs we have made between the three primary components will allow for optimum functionality and flexibility of the entire system. Since our prototype will be a portable device suitable for gaming and other real-time simulation, system size and power constraints are rigid and system refresh rate must exceed maximum visual perception limits of the average user. High accuracy and efficiency is required at all stages. A block diagram of our system is shown in Figure 2 below.

![Figure 2: System Block Diagram](image)

B. System Design

1) Sensor Board: We begin at the sensing block of the system. The rest of the design hinges on the accuracy of the sensors. MEMS sensors have reached an unprecedented degree of accuracy. Since our application demands knowledge of position and orientation, we will be using data from a GPS (MN1010), compass & accelerometer (LSM303DLHC), and gyroscope (L3G4200DTR) to construct our estimate of these two attributes. We have selected components which capture data in all three dimensions, with excellent range and sensitivity. Further, with four sensors and essentially only two desired state variables (position and orientation), we can use the data from each sensor multiple times to balance and compensate for the
deficiencies of another. It is then possible to devise an algorithmic “filter” which uses data from multiple sensors to form the best possible estimate of system state.

The gyroscope will be the most frequently updated sensor, since the user’s field of vision changes most drastically with respect to head movements. Digital gyroscopes are notoriously susceptible to “drift” and frequently need to be re-zeroed to a new frame of reference. The magnetic compass is an instrument known to be noisy due to fluctuations in the Earth’s, as well as incidental, magnetic fields, but is not susceptible to drift. We can form an accurate picture of head orientation from an appropriate “blending” or “weighting” of the two sensors (this will be discussed in greater detail in the computational stage). The GPS and the accelerometer will be used to estimate the user’s position. The physical orientation of the sensor board is very important to the design. We will also need to use the accelerometer to stabilize the gyroscope’s reading of head tilt with respect to the normal axis. This will be important to capture because the goggles will tilt with the user’s head, but ideally the visual field should not, so the virtual images must be rotated to negate the tilt of the head.

2) Kalman Filter: The raw sensor data will be fed to an AVR microcontroller, which will process that data to obtain meaningful information by using an estimation filter. The filter, as described by Welch and Bishop [5], is known as the Kalman Filter (KF), essentially a set of mathematical equations used to estimate the state of a process recursively, in a way that minimizes the mean of the squared error. According to Bishop: “the filter… supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown” [5]. Since the filter is entirely software-based, it would be possible to implement it using the Tunnel Creek Board. Previous implementations of the KF, however, have shown that embedded computers perform better at the task since they can maintain fast serial communication with the sensors, and will not be interrupted by background operating system procedures taking control of the CPU unexpectedly [6]. We will instead be processing the data using an on-board 32-bit microcontroller. The basic loop of the filter involves predicting the “current” state based on the previous state, measuring the current state, and forming an estimate of the true current state by combining the calculated estimate and the measured value. A conceptual diagram of our Kalman Filter is shown in Figure 3 and 4.

3) Intel Atom & OpenGL: The processed data will be transmitted to an Atom processor running Windows Embedded 7, which will be programmed using OpenGL to graphically display an image that is relevant to the user’s application. This software solves many computational challenges of graphics rendering such as image scaling and rotation, using a user-friendly interface. The sensor data will be received by OpenGL in a format already suitable for rendering, so that developers need only be familiar with OpenGL to develop their own unique application which displays relevant information for the environment in which it will be used. For our prototype, we will be generating a virtual world suitable for live-action gaming. In principle, however, the actual contents of the graphical display are determined by the application, and the preliminary stages of the system should lose no generality to the specific images we wish to display.

4) Display: The display solution will be a pair of goggles provided by Microvision, Inc., a company that manufactures wearable display systems. The goggles have a VGA input, which will receive visual data from the Tunnel Creek Board running an application using OpenGL. The display uses a laser-
based projector, combiner optics, and MEMS mirrors to generate a full-color, 3D display through the goggle lenses. The technology is also small enough to be housed inside the goggles. With this resource at our disposal, the bulk of our design time will be spent on the sensor board and API development to achieve the most robust platform possible, which will allow for development on a Windows Operating System.

C. Performance Measures

The goal of this system is to provide an immersive AR experience for the user. There are two main criteria that matter in providing such an experience: (1) accuracy of sensing viewing angle and (2) speed of operation under motion. Accuracy is important to ensure that virtual information aligns correctly with real-world objects and speed of operation ensures that there is no perceivable lag between the user’s motion and the display. In addition, there are other criteria such as portability, applicability, and power consumption that affect the performance of the system. The following are some of the measures we will use to judge the performance of the system.

The accuracy of the sensing subsystem is crucial. Accuracy, in this sense, means the relative difference between the user’s actual movements and the estimate provided by the sensing. We also need to know the maximum allowable speed of motion that the sensors can capture. Ideally, we will be able to monitor the user’s location to within two or three feet, and their orientation to within 5°, and update at least twenty times per second. The sensors will not be this accurate on their own. We will test the sensors to determine their level of accuracy by applying a constant acceleration, or a known magnetic field, or rotational speed, and recording the output data to determine the accuracy and precision of a given sensor. Every sensor has a finite range and sensitivity. Once we have determined the accuracy limits of the sensors, we must learn the upper limits of speed at which this accuracy is still achievable. We can use the same testing apparatus to discern their speed limitations. We will need to know the physical sensing limits of the system to determine how frequently to sample and update the Kalman Filter.

A Kalman Filter using appropriate models of system error and measurement error will provide the best possible estimate of the current position and direction of the user. The system and measurement error are provided as inputs to the filter using our knowledge of the sensing subsystem’s capabilities and limits. The effectiveness of the Kalman Filter depends on these parameters, and we will devise a method of verifying our calculations. Using a known speed and trajectory, we can calculate the Root-Mean-Square error between the measured readings from the sensors and our expected values of system state. If the Kalman Filter’s estimate does not match with the expected result, its input parameters can be adjusted to “trust” the accuracy of the measurements more or less, and thus respond to measurements faster or slower, respectively. A similar process is described by Welch and Bishop in [5], who obtained the following plots for two different guesses of measurement error:

![Figure 5: KF response to noisy data, using (a) good and (b) poor estimates of measurement error [5].](image)

In plot (b) on the right, the filter’s level of “trust” in the sensor was too high, and thus it responded too quickly to the raw sensor data. In (a), the filter used a correct guess of sensor precision and thus achieved its desired “smoothing” qualities. We can use the same process, obtaining estimates for position and head angle, each in three dimensions, and plotting the estimates against our predefined trajectory.

There are two subsets of accuracy that will be measured: displacement and rotation. A third parameter, which accounts for the combined effect of displacement and rotation, will also be measured. Three tests were developed by Azuma and Bishop in [2], [5] to measure the relative error of each parameter. Each test is evaluated with no prediction, with prediction but without inertial sensors and with prediction with inertial sensors. The equation used to determine positional error $E_p$ is given by:

$$E_p = |Q_{Actual} - Q_{Predicted}|$$

(1)

The first test, called the “Walkaround”, consists of a user walking around the outer edge of a frame. This data can then be collected and compared to the user’s true movements. The frame in our test will be shaped as a figure-eight to allow for movement in all directions.

In the second test, the user stays in one place while only moving their head. The sensor board will be attached to the side of the head, as it is tilted 90 degrees in each of the yaw, pitch and roll axes. This approach was used by Foxlin [7]. The expression for
angular error $E_A$ is given by:

$$E_A = 2\cos\left[2\pi\left(\frac{\text{Opredicted}}{\text{OActual}}\right)\right] \quad (2)$$

The output of the filter can then be compared to the true values. The third test implemented by Azuma and Bishop is designed to measure the system performance while both position and rotation are changing. The specific details surrounding this test will be decided upon at a later stage in the project and will be similar to the two previously-mentioned approaches.

Another critical performance measure is speed, i.e. the system’s response to the user’s movements, and is defined in units of frames per second (video refresh rate of display goggles). Higher refresh rates allow for faster, less predictable movements. To create a true AR experience, virtual objects must appear to move fluidly across the screen when the user’s direction and/or location are changing.

Consider our system in action. If the user were to spin 360 degrees in two seconds, virtual objects should move smoothly and at the correct speed as they rotated. The sensing system must update quickly to be able to capture the wide array of unpredictable human movements which are possible. If the speed is too slow, the images will change in one abrupt flash. We must determine the maximum rate that the users can change orientation and still perceive surrounding images.

Relative error will be used to measure the performance of our system running at a specific speed. A maximum amount of allowed error will be determined by human perception (how much error becomes noticeable?). Since displacement is, in general, much slower than rotational movement, our measurement of speed will be primarily focused on rotation. That is, if a user can turn at the maximum allowed rate without causing a noticeable on-screen image distortion (maximum allowed error). Knowing the maximum rate at which a human can move their head and still perceive images and the amount of error that becomes noticeable to the user, a purely mechanical test can be created. This test will consist of the same apparatus as above, that can turn about all three axes (yaw, pitch, roll). The apparatus will then be moved 90 degrees in each direction in a controlled fashion, at the maximum speed of human movement. The output of the system can then be compared to the true value. If the error is more than the allowed value, the system speed does not meet the requirement and must be increased.

The computational subsystem performance will be measured by the amount of graphical content it is capable of displaying at a time. The more images we wish to display, the more computation will be required to render those images. We will begin by attempting to generate one basic image and then, once we have achieved this, gradually increase the amount and complexity of visual data, until we have determined the limits of our application. Through its OpenGL Performance Characterization (OPC) group, OpenGL supports a wide array of benchmarking tools which can be used to test the performance of our computational subsystem. Some of these tools measure performance based on the workload of a typical user, including functions such as wireframe modeling, shading, texturing, lighting, blending, inverse kinematics, object creation and manipulation, editing, scene creation, particle tracing, animation and rendering. Others contain tests that measure an application’s performance for I/O-intensive operations, or CPU-intensive operations. These tools will be used to locate the performance “bottlenecks” in our program, and to identify the cause of those bottlenecks. Additionally, Intel’s Graphics Performance Analyzer will be used for these tests, which provides both hardware and software performance metrics.

We plan on testing our developed API with the time we have remaining. We will generate test code and treat the API as a black box. We will create automated test cases, in order to maximize code coverage. We will call various functions and properties described in the API in a test application we design. Finally, we recognize that we will not be able to test all possible scenarios that are possible to use with our API and thus we will focus our testing on the most likely usage scenarios.

D. Design Alternatives

Though we believe in the concept of our design, it is only a concept at this time. Numerous factors, which are out of our control, could negatively impact its likelihood of success. At the very least, however, we must account for those factors and be prepared to implement a plan B in the event that any should present itself as a legitimate concern.

Our biggest concern is that the goggle solution we expect to be provided does not arrive in a timely fashion, not allowing us enough time to tailor our design to their goggles. This is an unlikely scenario, but it is one to keep in mind. If, at the beginning of March, we have not received the goggles, we will begin developing our own optical or video-based solution to the physical display. This could potentially require changing the location and orientation of the sensor board, and could also detract from the overall aesthetic quality of the project.
Our second largest concern is with the construction of a functioning sensor board that can fit inside the goggles. The board must be on the order of 1x2 inches, and must contain the entire Head-Tracking Unit: a GPS and antenna, two digital sensors, and a Microcontroller. Five surface-mounted ICs will introduce some clutter on a PCB of those dimensions and routing a board with so many different signals can be a complex process and those signals could be susceptible to noise, crosstalk, interference, etc. In the case that the prototype does not function properly, we will need to make absolutely sure that our second attempt works. PCB manufacturing is expensive and takes time, so two iterations are really all we can afford if the project is to be completed within budget and in time.

There are environmental factors which could affect the sensor board. These, however, can often be addressed in software. For instance, the magnetometer is susceptible to interference from random, incidental magnetic fields. The GPS needs a strong signal to work, and it rarely gets a strong signal indoors. We will need to test how our system performs without a GPS, and potentially need to power down the GPS unit if the application is to be used indoors. If we find that the antenna simply does not work well when enclosed in the goggles, this will force us to tweak the design. We do not know how the device will perform in inclement weather or rough terrain – something to keep in mind, but not a major concern.

III. MDR Prototype Implementation

A) Sensing Unit Subsystem

1) Choice of Microcontroller

We have chosen a 32-bit microcontroller, the AT32UC3B0256 from Atmel. This is a high-performance, low-power microcontroller. Drawing only 260 mA at 3.3 V DC, this microcontroller allows us to use a single voltage regulator for most of the board. The CPU operating frequency can be set up to 60 MHz. Also, with 256 KB of Flash program memory at our disposal, we will be able to implement a strong prediction/correction algorithm on the microcontroller to give us the best possible estimate of the user’s head position.

We will exploit many of the key features which distinguish the 32-bit series from all others offered by Atmel. These MCUs are preprogrammed with a bootloader which allows the Flash memory to be programmed via USB. This feature makes the board more user-friendly and easy to update by anyone. The architecture of this MCU is quite different from the classic ATMega that we used in ECE 353 projects. Peripherals such as the USB interface can communicate with the data memory without CPU involvement. This allows us to increase the frequency of updates the CPU can receive from the inertial sensors and/or the amount of time the CPU can spend forming its estimate of head position.

2) Overall Circuit Design

The sensor board must be sufficiently powerful to capture the user’s true position, but it must also be small and draw very little power. As such, the task of designing the PCB involves a great deal of thought and care. The GPS antenna must have a very stable ground plane surrounding it so that interfering signals are blocked, and so that the signal of interest does not create noise in the rest of the circuit. The same can be said of the external crystal oscillator, and the microcontroller itself. The use of a USB interface requires some design considerations as well. We have designed a 1.3”x2.4” circuit board in EAGLE, shown in Figure 6, which we have ordered to be fabricated and assembled by a professional board house, since we lack the experience to solder some of the extremely small surface-mount technologies.

![Figure 6: Sensor Board Layout](image)

The entire circuit will be powered by the USB connection, which provides 5V at 500 mA DC power. Every component, with the exception of the GPS (which requires 1.8 V), will run off of a single 3.3 V Linear Low-Dropout Voltage Regulator, and the current consumption of these components has been calculated to be under 300 mA. The only component remaining, the GPS, requires 40 mA at 1.8 V. We calculate the total power dissipation of the board to be 0.787 Watts, with the bulk of this power being consumed by the microcontroller and the GPS.
B. Applications for Demonstration

For the MDR demonstration we will have two applications. The first will be a simple head object that will rotate on a computer screen based on the movement of the sensors connected through USB as seen in Figure 7 below.

![Figure 7: MDR Demonstration Application](image)

This demonstration will show the communication from sensor board to OpenGL, and will show the efficiency of the sensor board with respect to time.

The next demonstration will be a potential final application: personal navigation. We will display a route on the computer screen with random pseudo data being generated to resemble input from the sensor as seen in Figure 8 below.

![Figure 8: Final Application Example](image)

IV. PROJECT MANAGEMNT

A. Team Member Roles

In order to divide responsibility we first had to split the project into specific subsections. These sections include sensor board development, microcontroller programming & Kalman filter implementation, microcontroller & Intel Atom communication, and API & final application development.

The sensor board development involved creating schematics for each component. Then connect those schematics together to make a full sensor board schematic. Then this would need to be translated into PCB manufacturing diagrams. This involved learning EagleCAD software. Next the designs were created and the proper gerber files were generated. The sensor board development was done by both Matt and James.

Next, the microcontroller programming involves receiving the output from the sensors. Then, we will implement the Kalman filter that will filter the data coming in into more meaningful data. Then, the data coming out of the Kalman filter is sent to the Intel Atom board in order to be used as input to OpenGL. The microcontroller programming & Kalman filter implementation is being done by Matt and James.

Next, the communication over USB between the microcontroller and Intel Atom board needs to be implemented. This requires implementing a USB standard for data transmission, and creating a driver for the application on the Atom to receive the data. This will be done by To and Ryan.

Lastly, the API & final application development, involves creating the graphics that will provide an immersive augmented reality experience. This involves creating the API to easily translate the sensor data into graphical rotations and translation. Next, we will create an application that will demonstrate the API and the system as a whole. This will be done by To and Ryan.

B. Outlook on Remaining Tasks of Project

As of now, our sensor board is being manufactured. Again, we face the risk of the board not working completely once we receive it. This could have a significant impact on the final project. We will need to find out what is wrong with the board and order a new one.

Next, we discovered an unforeseen task while beginning the programming of the microcontroller. This task is creating the driver for the microcontroller to communicate with the Intel Atom. We do not believe that this will take up too much time as we have already begun developing the driver.
We are also simultaneously developing the API and final application. We are currently using pseudo data that is generated exactly as our data transmission specification. That way, all that would need to be completed is the communication between the sensor board and Intel Atom.

Overall, we feel to be on track to complete the project by the dates we were given without any changed to out proposed project.

C. Current Timeline for Project

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Microcontroller Programming</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Sensor Board Debugging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Microcontroller Programming</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Application/API Development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Sensor Board Completion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Application Testing/Debugging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Application Completion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. We have already begun the microcontroller programming. Since we have evaluation boards of the microcontroller and some inertial F C sensors at our disposal, we do not need a fully working sensor board to begin developing the program we will use for it. Depending on the amount of existing code we are able to leverage for our project, the programming of the microcontroller could take anywhere from a month to three months. We plan on completing the microcontroller programming by the end of January 2011 at the latest.

2. Once we receive the manufactured PCB we are planning on testing/debugging any problems with the board. We plan on completing the testing by the mid February 2012. This should give us enough time to find any problems and even get another board manufactured if need be.

3. Our second milestone marks the completion of the microcontroller programming.

4. We have already begun creating the final application and API that will run on the Intel Atom using the data from the sensor board. This involves developing a graphics intensive application to be displayed in the goggles. We created data transmission specification in order to begin developing the application using random pseudo data. We will also simultaneously be developing an API which can be used by other developers to access the data from our sensor board and display them on the goggles. We plan to complete these tasks by mid-March 2012.

5. Our third milestone will be reached once we have thoroughly debugged the sensor board design, and have obtained our final revision of the board. We expect to reach this milestone no later mid February, to leave sufficient time to develop the API around our sensing system.

6. The final task on our timeline is the testing and debugging of the API, which we will begin once the sensor board is complete. Since the physical characteristics of the actual sensor board will be different from any devices used for testing, the computational component of the system will need to be tailored to match the specifications of the sensing component.

7. The final milestone of our project is the completion of the API and final application, which will also mark the completion of the project. We therefore aim to reach this milestone by the end of April.

V. SUMMARY AND CONCLUSIONS

Our goal of this project is to advance the development of augmented reality electronics by decreasing the costs associated with creating these devices and availability to a broader set of people than the current augmented reality devices. We plan to create a sensor board that will sense a user’s location and direction, and to develop the communication between a modern day processor and the sensor board in order to develop an immersive, augmented reality experience.

Overall, we believe we are on track to complete the project by the end of April. We ran into some unforeseen problems, but we are working to overcome these problems and mitigate the time and money consequences of those problems. We currently are developing the communication between the sensor board and Intel Atom, and developing the API and final application.

VI. REFERENCES


APPENDIX

A) Application of Math, Science, and Engineering

This capstone project will give us the chance to apply many of the technical skills we have acquired in the past four years. The design of the sensor board circuit has required us to recall many of the topics covered in ECE 211, 212, 314, 323, 324, 333, 353, and 584. We needed to calculate the maximum power dissipation of the entire circuit, the output current demanded of our voltage regulators, and similar circuit design considerations. Our knowledge of electromagnetic fields and waves was definitely useful in designing a PCB with a proper ground plane, well-routed data signals, and strategically-placed components. Computer Systems Lab (ECE 353 and 354) gave us a primer on AVR microcontrollers, and the hardware associated with them such as resets, external oscillators, JTAG programmers, etc. The microcontroller is programmed in C, and OpenGL is programmed in Java, so our knowledge of these two languages from ECE 242 and 353 will be very useful. The head tracking program requires knowledge of classical mechanics to interpret the data from the gyroscope and accelerometer. The display program will require knowledge of optics and lenses. The head tracking program, which implements a Kalman Filter to sense the user’s position, requires matrix computations, the solution of differential equations, and relies on some probability theory.

B) Design and Performance of Experiments, Data Analysis, and Interpretation

So far, we have been working on gathering and filtering sensor data on an Arduino. We have been successful in communicating with all four sensors – the gyroscope, the accelerometer, the compass, and the GPS – simultaneously. We have implemented a Kalman Filter which takes the data from the gyroscope and the accelerometer, and accurately determines the tilt angle of the sensor board in two directions. Since the coordinate system must be fixed and the user’s frame of reference will change dynamically, we need a third sensor to capture the changes in all three directions simultaneously. This will be our next undertaking.

The 32-bit microcontroller requires a more complex programming style and it has taken some time to familiarize ourselves with the code structure. While we wait for our sensor board to be manufactured, we have been using a development board provided by Atmel which uses the same microcontroller we will be using. We have successfully implemented some small programs, and achieved communication over I2C and USART, but when we try to combine the two interfaces in a single program, we run into bugs with the system clock. The I2C interface contains its own clock initialization, which has been conflicting with the system clock of the USART.

From these results, we can safely say that it is possible to communicate with four sensors simultaneously and we feel confident that we will be able to implement a Kalman Filter to obtain the necessary head-tracking data.

C) Design of System, Component, or Process to Meet Desired Needs within Realistic Constraints

The design requirement is a system that can refresh 20 times per second, sensing location with accuracy to within 2 feet, and head angle to within 5°. Our inertial sensors are capable of outputting up to 800 two-byte packets of data per second. This, of course, is more than necessary, especially since the Kalman Filter will take some time to compute a new estimate of state. Indeed, for the sensing block, the Kalman Filter will have the greatest impact on the overall system refresh rate, but it will also allow us to
achieve the second requirement: high accuracy. On the computation end, the graphics rendering will be performed on an Intel Atom, operating at a CPU frequency of 1.66 GHz – fast enough to generate multiple objects in OpenGL within the time constraints. The choice of a USB interface between the sensing and computation blocks will introduce some complexity to the code and the hardware, but it offers a much higher transfer rate than standard serial interfaces.

D) Multidisciplinary Team Functions

- To Chong
  - OpenGL application
  - Website Development
- Matthew Ferrante
  - Sensor Board Circuit Design, Layout
  - Microcontroller Programming
- James Kestyn
  - Kalman Filter Implementation
  - Microcontroller Programming
- Ryan Offir
  - OpenGL application
  - USB Interface

E) Identification, Formulation, and Solution of Engineering Problems

One engineering problem we identified early was the establishment of a standard global coordinate system. Some applications will need to be aware of nearby businesses and restaurants. We need to create a frame of reference for the OpenGL program to interpret the data it receives from the sensing unit. The inertial sensors will determine the change in position and angle, relative to a general 3D coordinate system, while the GPS will give the global coordinates. The OpenGL program must use the global coordinates, along with elevation, to render the correct objects in the display. After doing some research, we found that we are able to obtain path data from Google Earth in KML file format, which provides us with a stable solution to this problem.

F) Understanding of Professional and Ethical Responsibility

We have a professional responsibility to Microvision to uphold the terms they have set in their Non-Disclosure Agreement. At the same time, we must not allow this agreement to compromise our responsibility to divulge information to our faculty reviewers relating to our progress.

Though our applications will not be very ethically sensitive, this device could potentially have enormous ethical implications. If the device became open-source, allowing the general public to write their own OpenGL applications, they could potentially find ways to obtain private data. Consider a program designed to obtain the location of other people wearing the goggles, or simply other people in the user’s circle of acquaintances, and somehow alert the user to their location. Sometimes people do not wish to be found. This could be problematic if there is no mechanism in place to protect their information from being searchable. It is not a major concern, however, as this sort of “friend” location is already possible between users on the same cell phone network, and the persons must deliberately make themselves searchable.

G) Team Communication

Our team communication has been strong thus far. Beginning in the summer, we have held weekly meetings with our project advisor, Professor Wolf, discussing progress and determining the best course of action for the week. Since we often work in pairs, on different aspects of the project, all relevant files, images, articles, programs etc. which we find or develop are synchronized to a Dropbox account, to which we all have 24-hour access. This has been an efficient method of letting other group members know what we are working on, and of coordinating group tasks.

H) Understanding of the impact of engineering solutions in a global, economic, environmental and societal context

I) Application of material acquired outside of course work

As previously stated, PCB design was not part of the general curriculum in the Electrical Engineering program. We taught ourselves how to use the EAGLE software and generate gerber files by consulting many tutorials on the internet, as well as more experienced graduate students at UMass. We also needed to learn about EMC compliance, as well as common design principles for high-speed signal routing, surface-mount antennas, crystal oscillators, and other crucial elements in the system which cannot simply be thrown haphazardly onto the circuit board and expected to work properly. This was a time-intensive process, and required many revisions of the PCB before we felt it was ready to be sent out for manufacture.
J) Knowledge of contemporary issues

AR and Virtual Reality (VR) systems have many potential applications. Surgeons have been shown to benefit greatly from virtual reality simulation training [1], and AR has been suggested in Azuma [2] as a real-time aid to medical professionals to provide instant “x-ray vision” of a patient’s tissues in the examination or operating room. It could be a useful tool to firefighters, instantly providing them with building schematics or other vital information. AR has long been used by the military to superimpose information to pilots through Head-Up Displays [2]. Recently, VR environments have been used in training to simulate battle situations [3]. These simulations have been limited to the computer screen, however, and a 3-D partially-immersive environment would provide a far more realistic experience for trainees.

K) Use of modern engineering techniques and tools

1) Cadsoft EAGLE, a PCB design software suite, was used to design the sensor board

2) AVR Studio 5, a graphical user interface for programming and debugging Atmel microcontrollers, will be used to program the sensor board.

3) USB 2.0, a high-speed data transmission protocol, will be used both to power the sensor board and to interface between the sensor board and the Intel Atom.

4) Tunnel Creek, a development board provided by Intel, will be used to render the graphical display.

5) The goggles to be provided by Microvision are state-of-the-art technology, using lasers and scanning MEMS mirrors to generate images in the wearer’s field of vision.