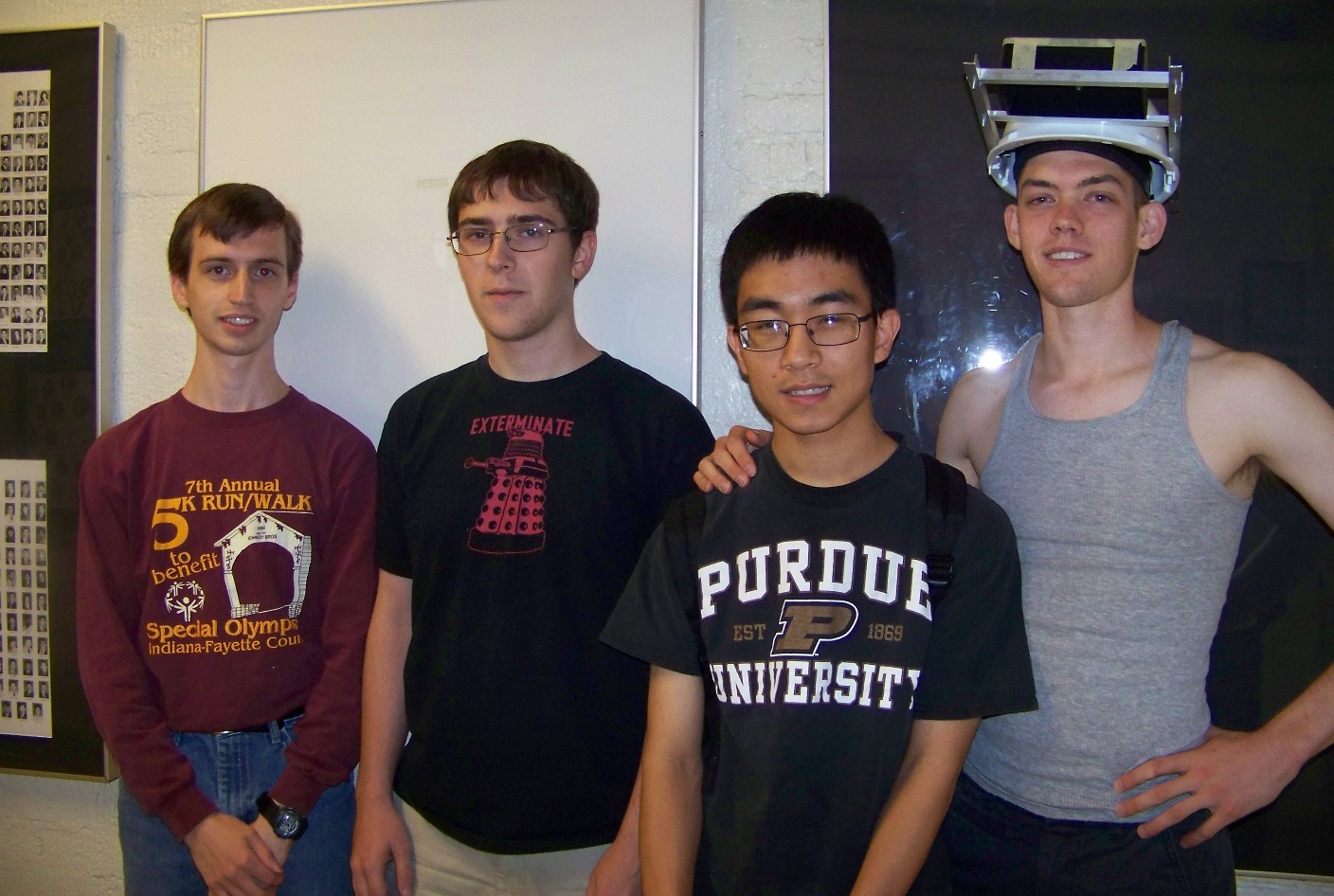
ECE 477 Final Report − Fall 2013

Team #5 − Augmented Reality Headset

****

*From Left: Thor Smith, Stephen Ellis, Stephen Carlson, Alec Green*

Team Members:

#1: \_\_\_Stephen Carlson\_\_\_\_\_\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_12/8\_\_\_

#2: \_\_\_Stephen Ellis\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_12/8\_\_\_

#3: \_\_\_Alec Green\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_12/8\_\_\_

#4: \_\_\_Thor Smith\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_12/8\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| **CRITERION** | **SCORE** | **MPY** | **PTS** |
| Technical content | 0 1 2 3 4 5 6 7 8 9 10 | 3 |  |
| Design documentation | 0 1 2 3 4 5 6 7 8 9 10 | 3 |  |
| Technical writing style | 0 1 2 3 4 5 6 7 8 9 10 | 2 |  |
| Contributions | 0 1 2 3 4 5 6 7 8 9 10 | 1 |  |
| Editing | 0 1 2 3 4 5 6 7 8 9 10 | 1 |  |
| Comments: | | *TOTAL* |  |
|  | | | |

Table of Contents

[Abstract 1](#_Toc374364827)

[1.0 Project Overview and Block Diagram 1](#_Toc374364828)

[2.0 Team Success Criteria and Fulfillment 4](#_Toc374364829)

[3.0 Constraint Analysis and Component Selection 4](#_Toc374364830)

[3.1 Computation Requirements 4](#_Toc374364831)

[3.2 Interface Requirements 5](#_Toc374364832)

[3.2.1 On-Chip Peripheral Requirements 5](#_Toc374364833)

[3.2.2 Off-Chip Peripheral Requirements 6](#_Toc374364834)

[3.2.3 Power Constraints 7](#_Toc374364835)

[3.2.4 Packaging Constraints 7](#_Toc374364836)

[3.2.5 Cost Constraints 8](#_Toc374364837)

[3.3 Component Selection Rationale 8](#_Toc374364838)

[3.3.1 Microcontroller 8](#_Toc374364839)

[3.3.2 Motherboard/Graphics Processing Unit 8](#_Toc374364840)

[3.3.3 Inertial Measurement Unit 9](#_Toc374364841)

[3.3.4 GPS Receiver 9](#_Toc374364842)

[3.3.5 Wireless Communication 10](#_Toc374364843)

[3.3.6 Battery 10](#_Toc374364844)

[4.0 Patent Liability Analysis 11](#_Toc374364845)

[4.1 Introduction 11](#_Toc374364846)

[4.2 Results of Patent and Product Search 11](#_Toc374364847)

[4.3 Analysis of Patent Liability 13](#_Toc374364848)

[4.4 Action Recommended 16](#_Toc374364849)

[4.5 Summary 16](#_Toc374364850)

[5.0 Reliability and Safety Analysis 16](#_Toc374364851)

[5.1 Introduction 16](#_Toc374364852)

[5.2 Reliability Analysis 17](#_Toc374364853)

[5.3 Failure Mode, Effects, and Criticality Analysis (FMECA) 19](#_Toc374364854)

[5.4 Summary 20](#_Toc374364855)

[6.0 Ethical and Environmental Impact Analysis 20](#_Toc374364856)

[6.1 Introduction 20](#_Toc374364857)

[6.2 Environmental Impact Analysis 20](#_Toc374364858)

[6.3 Ethical Challenges 22](#_Toc374364859)

[6.4 Summary 23](#_Toc374364860)

[7.0 Packaging Design Considerations 24](#_Toc374364867)

[7.1 Introduction 24](#_Toc374364868)

[7.2 Commercial Product Packaging 24](#_Toc374364869)

[7.2.1 Product #1 24](#_Toc374364870)

[7.2.2 Product #2 24](#_Toc374364871)

[7.3 Project Packaging Specifications 25](#_Toc374364872)

[7.4 PCB Footprint Layout 26](#_Toc374364873)

[7.5 Summary 27](#_Toc374364874)

[8.0 Schematic Design Considerations 27](#_Toc374364875)

[8.1 Introduction 27](#_Toc374364876)

[8.2 Theory of Operation 27](#_Toc374364877)

[8.3 Hardware Design Narrative 29](#_Toc374364878)

[8.4 Summary 31](#_Toc374364879)

[9.0 PCB Layout Design Considerations 31](#_Toc374364880)

[9.1 PCB Layout Design Considerations – Microcontroller 32](#_Toc374364881)

[9.2 PCB Layout Design Considerations - Power Supply 33](#_Toc374364882)

[10.0 Software Design Considerations 34](#_Toc374364883)

[10.1 Introduction 34](#_Toc374364884)

[10.2 Software Design Considerations 35](#_Toc374364885)

[10.3 Software Design Narrative 39](#_Toc374364886)

[10.4 Summary 41](#_Toc374364887)

[11.0 Version 2 Changes 41](#_Toc374364888)

[12.0 Summary and Conclusions 42](#_Toc374364889)

[13.0 References 43](#_Toc374364890)

[Appendix A: Individual Contributions A-1](#_Toc374364891)

[.1 Contributions of Stephen Carlson: A-1](#_Toc374364892)

[.2 Contributions of Stephen Ellis: A-2](#_Toc374364893)

[.3 Contributions of Alec Green: A-3](#_Toc374364894)

[.4 Contributions of Thor Smith: A-5](#_Toc374364895)

[Appendix B: Packaging B-1](#_Toc374364896)

[Appendix C: Schematic C-1](#_Toc374364897)

[Appendix D: PCB Layout Top and Bottom Copper D-4](#_Toc374364898)

[Appendix E: Parts List Spreadsheet E-1](#_Toc374364899)

[14.0 Qty E-1](#_Toc374364900)

[Appendix F: FMECA Worksheet F-2](#_Toc374364901)

# Abstract

We propose an Augmented Reality Simulator that allows multiple users to interact in a mobile, outdoor environment simulation. A central control unit will coordinate the gameplay while per-player headsets will appropriately overlay game-object pixels on a semi-transparent panel that is suspended in front of the users’ eyes. This product is intended to be used for gaming and other potential simulations that require an augmented environment.

# Project Overview and Block Diagram

Our project consists of two primary components as shown in Figure 1.1: a single immotile central control unit (CCU), and multiple per-user mobile headsets. A user chooses the desired simulation (e.g. game or virtual tour) via the CCU, which is equipped with a keypad and LCD display. The CCU will then wirelessly transmit all simulation-relevant data (e.g. 2D images, 3D models, and if the headsets were equipped with speakers, audio files) to the headsets, which the CCU will later reference by index. The CCU will then signal the start of the simulation to the headsets, and then coordinate game logic throughout the simulation by taking into account the periodic sensor (IMU, GPS) readings returned by each headset. Each headset will also make use of its sensor readings by rendering the image depending on user head orientation, geospatial location, and status in the game/virtual tour.



Figure 1.1: Block diagrams of the CCU (Central Control Unit) and headset respectively.



Figure 1.2 Final CCU Packaging

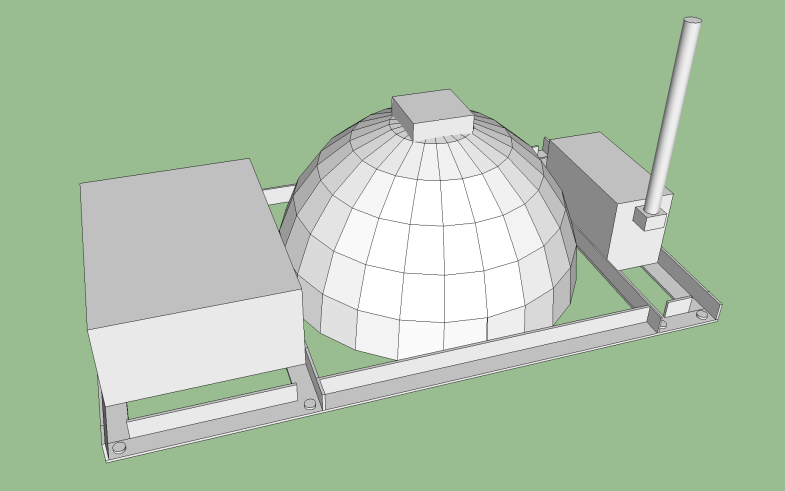


Figure 1.3: CAD Rendering of Headset



Figure 1.4: Final Headset Packaging

# Team Success Criteria and Fulfillment

1. [PASS] An ability to render graphics based on the orientation of the user’s head.
2. [PASS] An ability to render graphics based on the user’s current geospatial location.
3. [PASS] An ability to monitor and display the battery power level of the headset to the user.
4. [PASS] An ability to monitor and display the status and quality of the wireless connection to the central control unit.
5. [PASS] An ability to load headset graphics for a new simulation without re-flashing software on the headset’s microcontroller.

# Constraint Analysis and Component Selection

Augmented reality means that the user will wear the device, so the most important constraints include the packaging, power management, and convenience of the headset. Fluctuating CPU processing demand presents challenges for graphics and microprocessors, but the self-contained nature of the project relaxes requirements on general purpose I/O and isolation/signal drive. The central control unit will perform fewer tasks and thus has limited real-time and power constraints.

## Computation Requirements

The microcontroller component of the headset must handle the wireless communication and all of the sensors installed, as detailed in Table 3.1. The most intensive task of the microcontroller will be decoding and filtering the 9-DOF IMU data describing the orientation of the user’s head. Since this requires floating point math and must run at a constant rate for best results, this task imposes strong real-time constraints. String processing will be required on the GPS data to turn the NMEA strings into latitude and longitude coordinates.

Table 3.1: Computational tasks and real-time constraints for headset and central control unit

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Runs on | Data Rate | Jitter Requirement |
| IMU filtering | Microcontroller | 800 Hz | 5 us |
| GPS string parsing | Microcontroller | 20 Hz | None |
| Wireless communication | Microcontroller | 11.5 KB/s | 1 ms |
| Battery monitoring | Microcontroller | 1 Hz | None |
| Graphics rendering | Headset GPU | 30 Hz | 5 ms |
| Simulation logic | CCU | Varies | Varies |

Wireless communication will require parsing packets received, but data reception rates will be limited to 115200 baud by the wireless module [**6**]. The wireless signal level and battery level will only be checked once per second to save CPU time. Interrupt processing can be reduced due to the DMA support on many microcontrollers [**33**], which can automatically transfer SPI, UART, or I2C data to and from memory and interrupt only when a fixed number of bytes have been processed.

On the dedicated headset GPU board, the only task will be performing GPU rendering. As the GPU is designed for this task, it should be able to render at the required 30 Hz for screen updates. The GPU will receive data over SPI which can be processed in parallel with rendering. The central control unit has to handle only wireless communication and simulation logic, so it has few real-time processing constraints. CPU usage will depend on the complexity of the active simulation.

## Interface Requirements

General purpose I/O requirements are relaxed on both the headset and the central control unit. The motherboards on both units will not interface over their GPIO. Microcontroller pins will also not be used to drive loads and can be placed in reduced drive mode to conserve power. Since all peripherals are supplied with 3.3 V, processors with 3.3 V levels are preferred to satisfy peripheral voltage level constraints. Only the USB charging port will have external connections, minimizing the need for isolation; only a diode is required for electrostatic discharge suppression.

### On-Chip Peripheral Requirements

On the headset, the microcontroller must be able to handle each sensor connection as shown in Figure 1.1 and Table 3.2. The connection to the external GPU will be handled via SPI, requiring five dedicated pins for clock, data in, data out, data ready, and chip select. An I2C bus will be used for both the inertial measurement unit [**34**] and the battery monitor chip [**22**]. Two UART ports will be used for the GPS [**35**] and wireless units; while the GPS port needs only to receive data, the wireless port must be bidirectional, for a total of three additional pins.

Table 3.2: Required on-chip microcontroller peripherals for headset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Device | Peripheral | Special Features | Minimum Speed | Pin Count |
| External GPU | SPI | Master Mode, Chip Select | 800 KHz | 5 |
| GPS | UART |  | 115.2 Kbaud | 1 |
| Battery Monitor | I2C | Open-Drain, Current Sink | 400 KHz | 2 |
| IMU | I2C | Open-Drain, Current Sink | 400 KHz | 2 |
| Wireless | UART | Full Duplex | 115.2 Kbaud | 2 |
| Future Expansion | GPIO | Configurable Data Direction | N/A | 10 |
|  |  |  | **Minimum Pins:** | 22 |

The headset GPU motherboard must have a compatible SPI port to interface with the microcontroller over SPI as described above. It also must support a standard display output such as VGA, composite, or HDMI for rendering to a commercially available screen. A second available standard interface such as USB or UART will also be preferred to allow for future expansion to input devices worn by the user for more advanced simulations.

The central control unit motherboard will require a USB port for a keypad for initial user configuration input. It must have either a dedicated UART capable of 115200 baud or another USB port to connect to the wireless communication device. A standard display output such as VGA, composite, or HDMI will also be required to provide initial user configuration feedback.

### Off-Chip Peripheral Requirements

To satisfy the project criteria, a GPS unit, wireless communication, inertial measurement unit (IMU), and battery monitor chip must be installed on the headset. The GPS unit should be able to function in a difficult landscape with minimal loss of accuracy. GPS update rate is also important to bring real-time screen updates, but time to first fix is not a concern. Almost all GPS units interface over a standard UART interface. The IMU determines the orientation of the user’s head using three 3-axis sensors: a gyro, an accelerometer, and a compass. Low noise and good resolution are important for increased orientation accuracy, and the IMU must interface over SPI or I2C.

Wireless communication keeps the central control unit and the headset connected throughout the simulation. High reliability and low latency are paramount to meet the real-time requirements of the simulation. The wireless device must also transmit at least 6-8 KB per second of data and must be able to report the signal quality and strength. A range of 150 yards line of sight will handle most simulations, but longer ranges, especially near obstructions, will add a variety of usable simulations. Battery monitoring components must provide either an alert pin or preferably an estimate of capacity remaining, but they should not use a current sense resistor as the heat dissipation and quantization error of most coulomb counting approaches would be unacceptable.

For the central control unit, a wireless communication device compatible with the headset and user input devices must be installed. The user input device must be interfaced over an available motherboard USB port and must have a comparable size to the motherboard. As a regular computer keyboard would be bulky, a simple numeric keypad will be adequate.

### Power Constraints

Power will be significantly limited on the headset, as it will run on a lightweight single lithium ion cell worn by the user. The biggest power consumer will be the GPU and display used to render images. Adjustable display brightness will be used to reduce power requirements. The wireless radio and GPS also consume power at a low duty cycle to send and receive data. By comparison, the current draw of the microcontroller, inertial measurement unit (IMU), and battery monitor is negligible. The headset aims to waste no more than 1 watt as heat to increase battery life. Low-dropout regulators are required since no available switch-mode regulator can efficiently convert a battery voltage as low as 3.7 V to the core 3.3 V supply as shown in Table 3.3.

Table 3.3: Power supply constraints and supply rails in use on headset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Supply Rail | Voltage | Current Supplied | Current Use | Peripherals Powered |
| Vcore | 3.3V LDO | 800 mA Low-ripple | 500 mA | GPU Motherboard |
| VMCU | 3.3V LDO | 500 mA | 350 mA | Microcontroller, Wireless, GPS |
| VADC | 3.3V LDO | 50 mA Low-noise | 30 mA | IMU |
| VDISPLAY | 5V Boost | 500 mA | 300 mA | Display, USB |

Power consumption is not a significant factor on the central control unit motherboard, as it will be AC powered via a USB wall wart. Active cooling is inconvenient to install, so the heat dissipation must be low enough to be adequately managed by passive cooling systems.

### Packaging Constraints

The headset is designed to be portable and worn by the user. Of the peripherals in use, the display, inertial measurement unit, and antennas must be placed on the user’s head. An adjustable mount for the half silvered mirror and reflection shield must be provided to allow users with different eyesight to position the device for optimal viewing conditions. The device should be packaged in one unit without exposed wires, as non-technical users should not be expected to plug in parts. Packaging must be able to withstand the vibration and impacts that are inevitable for wearable electronics, and must be able to tolerate outdoor temperatures.

The central control unit has fewer size restrictions as it will remain stationary throughout gameplay. It must be durable enough to withstand outdoor temperatures. Any packaging materials selected must also be compatible with the antenna to avoid signal reflections that could reduce the range of the headsets. The central control unit should have a simple keypad interface to ensure casual users can simply select a simulation and start immediately.

### Cost Constraints

While cost is a consideration when differentiating between similar parts, cost is not an overriding constraint at this time and is secondary to other design constraints. A soft limit of $250/headset and $100/CCU was used to guide parts selection, as competing products are generally very expensive even in quantity. For example, Google Glass [26], a product with higher quality compact optics, is currently priced at $1500, and several other augmented reality products such as ARQuake [42] and CastAR [21] have not left development.

## Component Selection Rationale

### Microcontroller

Processing power, particularly with floating point, is the biggest concern for microcontroller selection. Even with the removal of graphics, high-speed filtering is required for the IMU. Direct memory access peripherals can help ease the data movement strain. Current consumption is secondary as other devices dominate the power use. With these constraints set, only a few choices remain as shown in Table 3.4, all of which meet constraints for on-chip peripherals and pin count:

Table 3.4: Most viable microcontroller choices for headset

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Part | I/O | RAM | Flash | Clock | DMA | Special Features | Cost |
| STM32F405RGT6 | 51 | 192K | 1M | 168MHz | 16 | Floating Point Unit | $11 |
| PIC32MX695F512H | 53 | 128K | 512K | 80MHz | 8 |  | $10 |

The STM32F405 chip was chosen due to the availability of a low cost $15 development board and its very high computational performance, particularly with single cycle floating point operations. This device features extra RAM and processing power for future expansion.

### Motherboard/Graphics Processing Unit

As the GPU motherboard is the largest power consumer on the headset, low power consumption is mandatory. Small size and light weight is also important for a portable device. Since any dedicated GPU vastly outperforms the microcontroller, relative performance is not a concern. The motherboard should also feature a USB port and SPI interface, as this project aims to re-use the same part on the central control unit to reduce development effort.

Table 3.5: Most viable central control unit motherboard/headset GPU choices

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Part | Idle Power | RAM | USB | SPI | Clock | Cost (board) |
| Raspberry Pi Model A | 2 W | 256M | 1 | 1 | 700MHz | $25 |
| BeagleBone Black | 2 W | 512M | 1 | 2 | 1GHz | $45 |
| Intel Atom Board | 12 W | 2G | 4 | 1 | 1.8GHz | $150 |

As shown in Table 3.5, the Raspberry Pi Model A was chosen for its low power consumption, cost, and outstanding community support. Examples exist all over the Internet for handling tasks on the Raspberry Pi, whereas the BeagleBone Black is fairly new. The added power use of an Intel Atom board does not justify increased performance which is unnecessary for this project.

### Inertial Measurement Unit

Due to advancements in MEMS technology, most inertial measurement units provide excellent noise performance and accuracy. Power consumption is not an issue as most units draw little current. A breakout board must be available since most IMUs come in LGA packages.

Table 3.6: Most viable choices for 9-degree of freedom inertial measurement unit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Part | Gyro | Accelerometer | Compass | Resolution | Cost(board) |
| STEVAL-MKI108V2 | L3GD20 | LSM303DLHC | LSM303DLHC | 16/16/12 | $27 |
| MPU-9150 | MPU-9150 | MPU-9150 | MPU-9150 | 16/16/13 | $50 |

As seen in Table 3.6, the slightly increased resolution of the MPU-9150 is offset by its cost and lack of documentation, particularly for the use of its powerful internal filtering algorithms. The cheaper SGS Thomson part was chosen instead.

### GPS Receiver

Past groups often experienced problems with low resolution and slowly updating GPS. The real-time constraints of this project call for a fast and high-resolution receiver module. Power consumption is a concern, but most available modules draw similar amounts of power.

Table 3.7: Most viable choices for GPS receiver, with typical smartphone GPS shown for comparison

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Part | Resolution | Update Rate | Accuracy | Antenna | Power Use | Cost |
| Venus638FLPx | ~0.3m | 20Hz | 2m | External | 29 mA | $50 |
| GP-635T | ~1m | 5Hz | 2m | Integrated | 56 mA | $40 |
| (Smartphone) | ~5m | 1-5Hz | 10m | Integrated | N/A | Varies |

The recently released Venus638FLPx vastly outperforms comparable units due to its excellent resolution and astonishing 20Hz update rate as shown in Table 3.7. Real-world tests confirm the superior signal quality of external antennae. Smartphone data is provided for comparison.

### Wireless Communication

This project initially planned to use XBee wireless radios, as these have the required low latency and acceptable data rates as shown in Table 3.8. In addition, these modules interface easily over UART. Range tests, however, indicate that unrealistically tall antennas are required to obtain the advertised range.

Table 3.8: Most viable wireless communication strategies between headset and central control unit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part | Data Rate | Range (LOS) | Latency | Cost |
| XBee Pro 900HP | 11.5 KB/s | 300 m | 50ms | $40 |
| Bluetooth and Cell Phone Radio | 100+ KB/s | 2000+ m | 500ms | $30 |

Despite the range and data rate advantages of offloading wireless to a cell phone and communicating to the phone over Bluetooth, the latency in tests was shown to be unacceptable. Cell phone signal reliability also left much to be desired, meaning that the XBee Pro will still be used. The low range, especially near obstacles, places emphasis on the signal quality indicator.

### Battery

Battery power keeps each headset running. A lighter weight battery decreases user discomfort, while a higher capacity battery allows longer runtime. Therefore, lithium-ion batteries were the preferred choice, featuring good energy density which allows both constraints to be met. With a target run time of at least 3 hours, the worst case analysis projects needing about 6 Ah of battery capacity.

Two convenient choices were found for the battery, a SparkFun prismatic cell and an Adafruit cylindrical cell pack. Both were 1S3P format devices which combined three smaller cells in parallel with a protection circuit to reduce fault current and prevent overcharging. While the Adafruit battery was slightly cheaper, the SparkFun cell had a much smaller form factor and weighed less, leading to its selection for improved portability.

# Patent Liability Analysis

## Introduction

We are designing an augmented reality headset for gaming and multimedia applications. A stationary central control unit (CCU) equipped with a user interface, processing power, and radio communications will coordinate the application (e.g. game) logic among multiple mobile, user-wearable headsets. The mobile headsets contain multiple intellectual property (IP)-sensitive areas. Namely, the use of an IMU/GPS to determine the orientation/position of a user in a 3D environment for gaming purposes, the use of that orientation/position data to render ‘augmented reality’ images (i.e. pixel overlay on a natural environment view), and projecting those images on a semi-transparent/semi-reflective display that is affixed in front of a user’s field of view all represent areas of potential infringement. (Note that by virtue of our purchase of the Digi Xbee wireless communication devices, we are implicitly licensing the radio IP and therefore are not in danger of infringement of this technology.)

## Results of Patent and Product Search

Relevant published, active patents exist which could potentially overlap with the methods and functionality of our headset in the areas of: using a head-mounted measurement units (e.g. a gyroscope) for outdoor applications, using a semi-transparent/semi-reflective panel in a user’s field of view in order to overlay pixels on the environment, and using a measurement unit to dictate what visual information is overlaid on a such a head-mounted display.

[1] “Head Mounted Information Systems and Related Methods”

*Filed*

Jan 18, 2011

*Abstract*

A head mounted information system comprises a frame configured to be worn by a user, a sensor unit coupled to the frame, and a processor unit coupled to the frame. The processor unit is connected to receive signals from the sensor unit. A display unit may also be coupled to the frame to display the [sensor] parameters to the user.

*Key Claims*

1. A head mounted information system comprising:

a frame configured to be worn by a user;

a sensor unit coupled to the frame, the sensor unit comprising a gyroscope configured to produce angular velocity signals representing a head angular velocity about a generally horizontal z-axis oriented generally perpendicular to a direction the user is facing when the user is wearing the frame; and,

a processor unit coupled to the frame, the processor unit connected to receive signals from the sensor unit,

wherein the processor unit is configured to receive the angular velocity signals from the sensor unit, detect a jump by the user when the head angular velocity indicates upward head tilting exceeding a first jumping angular velocity threshold, and generate a jump output signal indicting one or more jump parameters.

21. A system according to claim 1 wherein the sensor unit comprises a three axis gyroscope configured to produce angular velocity signals about the z-axis, an x-axis and a y-axis, wherein the x-axis and y-axis are generally perpendicular to each other and to the z-axis, and wherein the processor is configured to, after detecting a jump and before detecting a landing, determine rotations about the x-axis, y-axis and z-axis.

22. A system according to claim 21 wherein the processor is configured to display orientation information in substantially real time based on the determined rotations.

[15] “Method and Apparatus for Displaying Images on Reflective Surfaces”

*Filed*

May 3, 2006

*Abstract*

A head mounted display (HMD) is worn on a user's head for displaying an image. A HMD is a personal see-through device designed to view still or video images or data that nonetheless permits the user to view his surroundings.

*Key Claims*

1. A reflective image display comprising:

a reflective, at least semi-transparent lens;

a frame for bearing the lens and configured to be worn by a user so as to place the lens in front of at least one of the user's eyes; and

a display, associated with the frame, for projecting an image onto a surface of the lens facing the user's eye or eyes such that the image is reflected thereon in a manner visible to the user.

2. The display of claim 1 further comprising a sensor for measuring an ambient condition and circuitry for causing the display to project information indicative of the measured condition

[46] “Method, System and Device for Augmented Reality”

*Filed*

March 20, 2002

*Abstract*

A portable electronic device comprises augmented reality viewing apparatus for viewing a real scene and a superimposed computer generated overlay scene. In one embodiment the viewing apparatus comprises a display screen and a semitransparent mirror. The device may be equipped with location determining means, the selection of a displayed image thereby being dependent on the location of the device, whether the images for display are stored locally in the device or transmitted by radio from a remote server. The device may also be equipped with an orientation sensor so that the selection of a displayed images is dependent on orientation of the device.

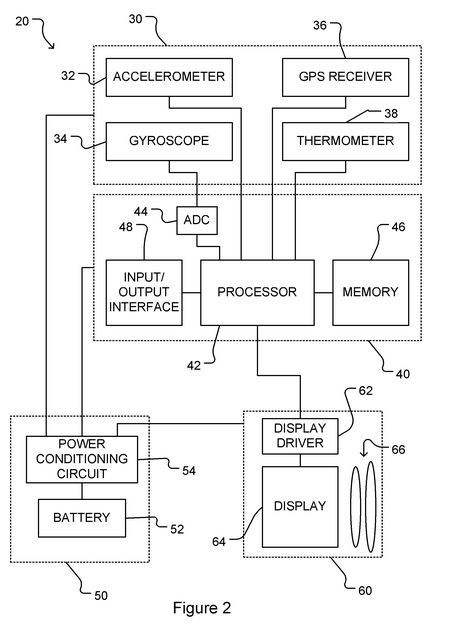
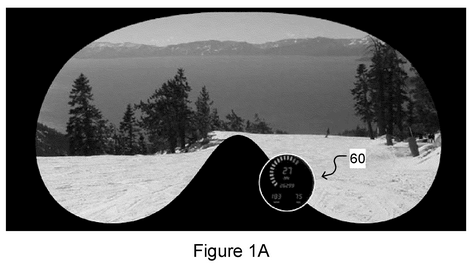
*Key Claims*

1. A method of preparing an overlay scene for display on an augmented reality viewing apparatus, characterised by generating an alignment indicator corresponding to a predetermined element of a real scene for inclusion in the overlay scene, the alignment indicator in use being aligned with the predetermined element of the real scene.

14. A device as claimed in any one of claims 3 to 10, comprising orientation sensing means for generating an indication of orientation, location determining means for generating an indication of location, and storage means wherein the storage means contains a plurality of overlay scenes each corresponding to a different real scene, and selection means for selecting which of the plurality of overlay scenes is displayed, wherein the selection means is responsive to the indications of location and orientation of the portable electronic device.

## Analysis of Patent Liability

[1] “Head Mounted Information Systems and Related Methods”

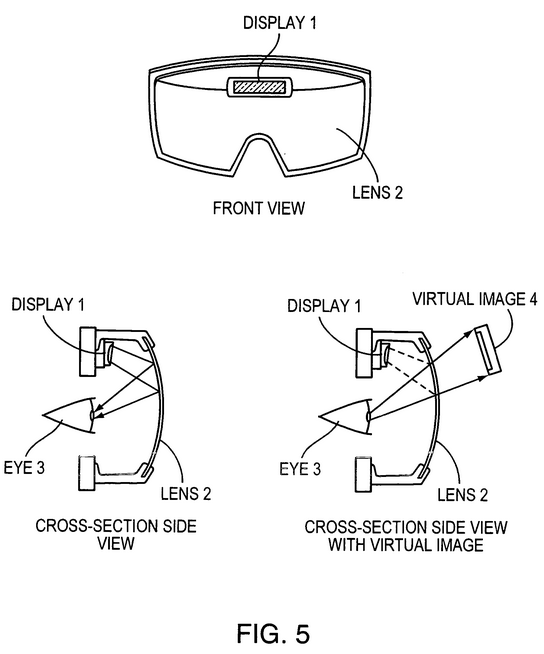


While the block diagram of the patent (Figure 2) is nearly identical to our block diagram, except for replacing a magnetometer sensor with a thermometer sensor, one must investigate the claims in order to determine if our application is truly in violation of this patent.

While the first three parts of claim 1 exactly describe our own application, the fourth component of claim 1 (using the angular velocity signals to determine jumping) differentiate this patent from our own application. This is because while we are sending gyroscope signals to a processor for calculation of orientation angles, we are not using these gyroscope signals for jump determination. Although it is unclear whether or not a claim must be considered in an atomic fashion (that is, must take all or none of the claim sub-components), we will assume that a claim must be considered atomically, and therefore our application does not literally or through doctrine of equivalents conflict with this patent on claim 1, because the intended functionality is distinct (jump detection vs general-purpose head orientation detection).

Claims 21 and 22 pose a similar situation as claim 1. That is, while we do use accelerometers and other IMU sensors to determine user orientation, we are not using this information to determine a user jump. Using the same logic used in the previous paragraph, we will again assume that our application does not conflict with these claims.

[15] “Method and Apparatus for Displaying Images on Reflective Surfaces”



The two and only two claims of this patent are written in very general terms concerning the application of overlaying images on a heads-up display and using complementary sensors to influence the images rendered on the semi-reflective display. Therefore, it seems that we are literally infringing on this patent, or at least come into conflict with the claims in terms of doctrine of equivalents. Our application also utilizes a ‘display’ (specifically, an LCD) for ‘for projecting an image onto a surface of the lens facing the user's eye or eyes such that the image is reflected thereon in a manner visible to the user’ (from claim 1). However, it is debatable whether or not we are infringing on claim 2, as we are not using additional circuitry (e.g. an LED) to display information from the ambient sensors, but rather are using the identical augmented reality image rendering pathway for generating the images based on IMU sensor data.

[46] “Method, System and Device for Augmented Reality”

Although the abstract of this patent sounds highly relevant to our application, reading through the claims (e.g. claim 1), it becomes apparent that this application is dependent on an ‘alignment indicator’ in the real-world environment. This is also known as an augmented reality ‘marker’, in which a sensing device (e.g. camera) processes the real-world visual input to the user, and appropriately aligns a pixel overlay onto that real-world visual input. However, in our application, we do not rely on any external visual input for the generation of our augmented reality image rendering, and instead only rely on orientation and position sensors.

Further claims in the patent (e.g. claim 14), are nullified by the fact that they depend on previous claims which do not apply to our application. That is, although claim 14 speaks of utilizing IMU sensors for influencing the augmented image, this use of IMU sensors is piggybacked on the ‘alignment indicator’ system which our project will not use.

## Action Recommended

[1] “Head Mounted Information Systems and Related Methods”

As determined in section 4.3, we will proceed on the assumption that claim conflicts must be considered atomically. Therefore, because our use of IMU sensing device and an augmented reality display are for a distinct purpose from this application, we are not in infringement of this patent, and no action needs to be taken.

[15] “Method and Apparatus for Displaying Images on Reflective Surfaces”

Because there is not a foreseeable way to modify our project so that it is not in violation of at least claim 1, we would need to obtain a license from the patent holder in order to manufacture and sell our application.

[46] “Method, System and Device for Augmented Reality”

Because we have determined that our application utilizes a significantly different method for augmented image overlay compared to this patent, no action needs to be taken.

## Summary

Multiple relevant patents were considered and analyzed in order to determine whether or not our application infringes on current art published by the US Patent Office. Although two of the patents [1,46] appeared highly relevant based on the patent abstract, it was determined through the claims that our application does not infringe upon them. However, our application does come into conflict with one patent [15] regarding the use of a semi-transparent/semi-reflective material for projecting images in front of a user’s field of view, and therefore we will need to obtain licensing from this patent holder before commercial sale of our project.

# Reliability and Safety Analysis

## Introduction

We propose an Augmented Reality Simulator that allows at least on user to play an electronic game in a mobile, outdoor environment. The device is split into two primary physical parts, but only the headset’s schematic will be considered for safety and reliability in this report. The report will focus on three sections of the headset’s schematic.

The Augmented Reality Simulator design includes four power supplies, a lithium ion battery, battery charging circuitry, a microcontroller, and other components. Failures related to the battery present potential hazards. Other failures will not cause harm to the user, but may render the device inoperable.

## Reliability Analysis

The mean time to failure (MTTF) allows one to compute an estimate of how frequently a part will fail given a certain number of parts in the field. Three components were selected to be measured based on military standards for reliability and safety of electronic equipment. The MCP73831 battery charger, MIC5219 voltage regulator, and STM32F4 microcontroller components were selected. The MCP73831 was selected because it works with the lithium ion battery, which is the most dangerous component on our device. The MIC5219 was selected because it delivers voltage to other components. If it fails then it could ruin those components and generate heat. The STM32F4 was selected because it is the most complex component in the schematic, and it controls the communication between all of the peripheral devices. The analysis for each component is shown in Table 5.1, Table 5.2, and Table 5.3. The Gate/Logic array formula in section 5 from [43] was used for all components.

Some assumptions were made when determining values. For the battery charger and the voltage regulator, the number of transistor was assumed to be a couple hundred. This assumption was based on an example calculation by Novacek. [11][12][13] The TjMax value for each component was calculated based on the power dissipated by the device. For example, the voltage regulator TjMax value was calculated based on the following: the battery’s maximum input voltage of 4.2V [44], the expected output voltage of the regulator at 3.3V, and the maximum current before the regulator fails of 0.5A. [25] The power is calculated as the quantity 4.2V minus 3.3V multiplied by the maximum current. This yields a result of 450mW. The θJC from the regulator datasheet is 60°C/W [25]. This gives a temperature rise of 27°C. This rise was added to 35°C, the T­A for GBE (Ground Benign Environment) [43] to get a value of 62°C, which was rounded up to 70°C.

Table 5.9: MCP73831 Battery charger failure analysis.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | Description | Value | *Comments* |
| C1 | Die complexity | *.20* | *Guess a couple hundred Transistors* |
| πT | Temperature coeff. | *28* | TjMax = 110 deg C |
| C2 | Constant for number of pins | *.0025* | Nonhermetic, 5pins |
| πE | Environmental Constant | *.50* | Ground Benign Environment |
| πL | Learning Factor | *1* | More than 2yrs production |
| πQ | Quality Factor | *10* | Commercial |
| *λ* | *(C1\** πT *+C2\** πE*)*πQπL | *5.60x10-5* | *MTTF = 2.03yrs* |

Table 5.10: MIC5219 Voltage regulator failure analysis.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | Description | Value | *Comments* |
| C1 | Die complexity | *.20* | *Guess a couple hundred Transistors* |
| πT | Temperature coeff. | *2.8* | TjMax = 70 deg C |
| C2 | Constant for number of pins | *.0034* | Nonhermetic, 8pins |
| πE | Environmental Constant | *.50* | Ground Benign Environment |
| πL | Learning Factor | *1* | More than 2yrs production |
| πQ | Quality Factor | *10* | Commercial |
| *λ* | *(C1\** πT *+C2\** πE*)*πQπL | *5.62x10-6* | *MTTF = 20.3yrs* |

Table 5.11: STM32F4 Microcontroller failure analysis.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | *Description* | *Value* | *Comments* |
| C1 | *Die complexity* | *.56* | *32bit Microcontroller* |
| πT | *Temperature coeff.* | *0.6* | TjMax = 70 deg C |
| C2 | *Constant for number of pins* | *.032* | *64 pins* |
| πE | *Environmental Constant* | *.50* | Ground Benign Environment |
| πL | *Learning Factor* | *1* | More than 2yrs production |
| πQ | *Quality Factor* | *10* | Commercial |
| *λ* | *(C1\** πT *+C2\** πE*)*πQπL | *3.52x10-6* | *MTTF = 32.4yrs* |

The part with the least MTTF is the battery charger. The MTTF is low because of its potential for a high operating temperature when the battery’s voltage has sunk to 3.4V. The incoming voltage from USB is 5V. However, battery charging does not occur while the device is being used for a simulation. It is expected that charging will occur overnight because the batteries take a long time to charge. If this device shorts V\_USB to V\_BAT (see Appendix C), the battery protection circuit will prevent overcharging of the battery. The component that is next most likely to fail is the voltage regulator. This device has a higher failure rate because of its temperature coefficient as well, but it is significantly smaller than the battery charger’s temperature coefficient. Increased safety could be achieved by monitoring the temperature of the battery charger through the microcontroller and disconnecting power from the USB connector if the temperature reaches a given threshold. A different modification could measure the current coming out of the battery charger in conjunction with a reading of the battery’s remaining charge to provide a redundant shutoff if the battery charger attempts to overcharge the battery.

## Failure Mode, Effects, and Criticality Analysis (FMECA)

Our design defines three criticality levels which are used in Appendix B for rating specific failure modes: High, Medium, and Low. High criticality indicates that the failure may cause physical harm to the user and possibly death. For this criticality level, *λ* must be less than 10-9. The remaining criticality levels may have *λ* less than 10-6. Medium criticality indicates that the user may experience discomfort while using the device, but no physical injury. This refers specifically to the possibility of displaying images at arbitrary locations, orientations, and zoom levels on the screen. Low criticality indicates that a benign failure of the device occurs such that the user is not harmed. The device may be rendered inoperable.

For the purpose of analysis, the Augmented Reality Simulator schematic was broken into six sections: the microcontroller, debug connectors, battery charging, power supplies, Raspberry Pi GPIO, and sensors. The microcontroller, battery charging, and power supplies blocks are highlighted in Appendix A because they represent the most critical parts of the design in terms of safety. The FMECA charts in appendix F show failure modes for each of the three blocks. The two high criticality failures P1 and B1 are in regards to the power supplies and the battery charging. If any of the power supply regulators fail by shorting to ground and for some reason the battery protection circuit fails as well, then the battery could discharge at an unsafe rate. A similar failure could happen if the battery charger shorted V\_USB to V\_BAT and the battery protection circuit failed. This could result in overcharging the battery, which could cause excessive heat. The failure rates of the regulator and the battery charger from Table 5.1 and Table 5.2 were not found to be sufficient to meet the requirements specified for a high criticality failure. However, these failures will be lower than that of the battery charger and regulator individually because the battery protection circuit must fail as well. Failure M2 is rated as medium because the microcontroller controls what data is sent to the Raspberry Pi for display. If the microcontroller failed in such a way that arbitrary data was sent to the Raspberry Pi, then the user could experience a disorienting display. Failures P2, P3, P4, B2, and M1 are rated low because while the device may cease to function, the user is not harmed in any way. The effect of many of the low criticality failures for the user results in a display that either doesn’t work or doesn’t change. Sensors may be ruined as in P2, P3, and P4 which will make data reported unreliable, but the user will only notice that the device “doesn’t work”.

## Summary

Our proposed project, an augmented reality simulator, aims to display an image to the user that adapts to their position and head orientation. Failure of this device to operate, in most circumstances, will only cause an inconvenience to the user because none of the expected functionality is critical for the user’s wellbeing. If the battery protection circuit fails, then damage to the user could result from failure of the battery charger or any of the voltage regulators. The possibility of this scenario is decreased because of two levels of redundant circuitry.

# Ethical and Environmental Impact Analysis

## Introduction

The augmented reality simulator is a headset with a translucent display and a set of sensors which allows graphics to be displayed based on the user's location and head orientation. A variety of software applications could be written using the hardware, but the current focus is on gaming applications. The electronics and packaging of the headset both present environmental problems in all of the manufacture, use, and disposal of the product. Additionally marketing such a headset faces several ethical challenges with regards to user safety.

## Environmental Impact Analysis

The first and most obvious environmental concern in the environmental impact of the augmented reality simulator is in its manufacture. The major concerns here are the PCB, semiconductor components, the LCD, the packaging, and the battery.

Printed circuit board manufacturing is quite environmentally harmful due to the large quantities of waste chemicals such as etchant, developer, resist, and solvents [9]. Additionally, a large amount of water is used in the manufacturing process to rinse these chemicals from the board during the manufacturing process, but certain steps can be taken by the manufacturer to reduce the amount of water used [45]. Any electronic product using a printed circuit board faces these environmental challenges, but a certain degree of mitigation is possible. The primary way to mitigate the environmental impact of PCB design is to use a smaller PCB, or to use fewer of them. The current design of the augmented reality simulator contains six PCBs due to the use of sensor breakout boards, a motherboard, and a display controller board. Consolidating these into a smaller number of PCBs with a smaller overall footprint could substantially reduce the environmental impact of manufacturing. In addition to reducing PCB size, some care can be taken to select a board manufacturing partner who takes their own precautions to be environmentally friendly as possible themselves.

Manufacturing of the integrated circuits used in the augmented reality simulator also poses an environmental challenge. Much like printed circuit board manufacturing, a large number of highly toxic chemicals is used during the semiconductor manufacturing process. These chemicals include, but are not limited to: acetone, arsenic, benzene, cadmium, hydrochloric acid, and lead [4]. In addition to these chemicals, a typical fab uses 2 million gallons of water and 240,000 kWh of electricity per day [4]. Like printed circuit boards, the use of semiconductors cannot be eliminated from the design of the augmented reality simulator. However, design reevaluation could potentially reduce the amount required and the associated environmental impact. Additionally, a semiconductor manufacturer can be chosen who recycles the needed chemicals and water to the greatest extent possible.

One specific electronic component worthy of environmental note is the LCD panel. The manufacturing process of LCDs, in addition to the usual semiconductor manufacturing concerns, releases a huge amount of greenhouse gasses into the atmosphere. In particular, nitrogen trifluoride and sulphur hexafluoride account for a large fraction of greenhouse effect gasses [7]. A typical LCD fabrication plant, per year, releases >300 tons of NF3 into the atmosphere [7]. Unfortunately all LCD panels are made using this harmful process, so it cannot be avoided. On a positive note, the display used has an LED backlight instead of the common CCFL lamps which contain the very environmentally harmful mercury.

The main materials used in the packaging are ABS, acrylic, and aluminum. ABS and acrylic are oil-based plastics which increases use of fossil fuels [47]. Aluminum is environmentally harmful due to the strip mining process used for the ore, which “nearly always involves some habitat destruction, soil erosion, loss of biodiversity, or water pollution” [48]. The ABS used could potentially be replaced by a biodegradable plant-based plastic such as PLA. The aluminum could probably be completely eliminated from the design by creating a single piece molded headset with integrated electronics enclosures front and rear.

The last environmental manufacturing concern is the battery. The battery used is a lithium polymer battery using a lithium cobalt oxide (LiCoO2) as the electrode. The main concern here is that it contains cobalt, which can cause health problems in the high concentrations found near mining areas [19]. An alternative battery chemistry such as lithium iron phosphate (LiFePO4) would be somewhat more environmentally friendly.

During the use of the augmented reality simulator its only environmental impact is its power usage, which is quite low at ~5W in operation. When powered off, the voltage regulator draws only ~10 microwatts. Thus the environmental impact during use is extremely low.

One other major environmental concern is the lifespan of the product, which will likely be limited by the battery. Lithium cobalt oxide cells are rated, depending on the specific cell for 400-1200 charge cycles [41]. Lithium iron phosphate cells can last up to 2000 charge cycles [8]. The battery is however easily replaceable by the end user without throwing out the entire product.

The final environmental impact to consider is when the product is disposed of. The PCB, semiconductors, and LCD are all non-biodegradable and cannot be recycled, leaving no better way to dispose of them than in a landfill. Due to the modular nature of the wireless and sensor hardware being on separate breakout boards, a particularly enterprising user could reuse those for alternative purposes. As for the packaging, both the aluminum and plastics can be recycled. The lithium battery is considered non-hazardous waste and is also recyclable. Although a large fraction of the product can be recycled, this does not mean that the end users will actually recycle it. Incentives could be provided such a free shipping envelope to send the product to a recycling center once it reaches its end of life, or even a small monetary incentive such as is offered with soda cans and bottles.

## Ethical Challenges

With an augmented reality device, the primary ethical concern is that of user safety. There are several ways in which the use of this headset could injure, or at least increase the risk of injury to, the user.

The most direct way in which the headset could cause user harm is a device failure resulting in a battery fire. Due to redundant protective circuitry, an electrical short resulting in a fire is a very low probability failure mode. Alternately, the battery could catch fire if it were punctured by something, but the battery is enclosed reducing that risk. If the battery were to be punctured, whatever is doing the puncturing is probably a bigger safety hazard than the battery. In spite of the low probability, a warning label or mention in the user manual would be warranted.

Another possibility for injury is the user accidentally running into another person. The current prototype's aluminum rails have sharp corners which could cause very serious injury in this situation. Probably the worst case packaging related injury would be the reflective acrylic sheet shattering and shards getting in the user’s eyes. Switching to an entirely plastic design (if this is structurally feasible) and ensuring that corners are sufficiently rounded instead of sharp can greatly mitigate the risk to others. To reduce the risk of the reflector shattering, a less brittle material such as polycarbonate could be used.

A relatively minor problem that the headset is likely to cause is eye strain. Even with the fresnel lens increasing the apparent distance from the user’s eyes to the display, the effective distance is only around 6”. This makes it impossible to focus on both the display and the user’s surroundings, which will likely force the user to frequently refocus their eyes to switch between the two. This frequent focus switching can cause eye strain and discomfort with extended use. A warning label should tell the user to take relatively frequent breaks when using the product for extended periods to prevent this. Additionally, an improved optics system could greatly reduce the eye strain by increasing the apparent distance to the display.

Augmented reality systems can also indirectly lead to injury in several ways. Like any computer display, triggering of epilepsy is a very real possibility which should be mentioned in a warning label or manual. Because the display is always in the user's field of view, it will partially obstruct the user's view creating a potential safety hazard. The display is partially transparent but especially in dark environments can make it difficult for the user to see their surroundings. If the user is using the headset for a game which involves running, this can increase the risk of tripping or running into things resulting in relatively minor injuries. Far more problematic would be the user attempting to use the headset while engaging in a potentially dangerous activity such as operating a vehicle. While the current software focus is gaming applications, someone could potentially write applications where this usage in a vehicle would make sense. However, usage in a vehicle would likely be rather dangerous and should be discouraged. To mitigate the safety problems with visibility, a warning label can urge caution when using the device. It should also stress not to use the headset when operating a vehicle or other heavy machinery. Additionally, when designing the software care should be taken to obstruct the user’s view as little as possible. Ensuring that stray reflections from anything other than the display are minimized will also mitigate this risk.

## Summary

The augmented reality headset faces numerous challenges of both an environmental and ethical nature. Environmental concerns pop up in the stages of manufacture, use, and disposal of the headset. Manufacturing poses very large environmental problems which can be partially mitigated by both design changes and selection of environmentally conscious suppliers. Usage of the product incurs a very small environmental impact. Disposal of the product is also environmentally unfriendly but can be mitigated to a great degree by recycling if the users care enough to do so. Ethically, user safety is the primary concern. A low probability but high criticality problem would be an electrical fire which should be warned of. Sharp edges are potentially injurious but can be eliminated from the design. Likely the biggest safety concern is the obstruction of the user’s vision by the display resulting in injury from another source.



# Packaging Design Considerations

## Introduction

Our project is a multiuser augmented reality system, consisting of one or more headsets communicating with each other through a central control unit (CCU). Therefore the packaging will consist of two main separate components, that of the headset and the CCU. The headset packaging is particularly important since it is where the vast majority of user interaction will occur. It will need to provide the display overlay on top of the user’s vision, position all antennas such that signal strength is not adversely affected, stay securely on the user’s head, and be reasonably comfortable for extended periods of use. The goals of the CCU packaging are to provide good antenna placement and an input method to configure the software at initial boot. The CCU packaging is much less interesting than the headset and will thus only take up a small fraction of the report.

## Commercial Product Packaging

### Product #1

Epson Moveria BT-100 [10]

The BT-100 is an augmented reality headset resembling a large pair of sunglasses. It uses a pair of projectors mounted in the sides of the frame aimed at pieces of half-mirrored beam splitter glass in front of the user’s eyes. This provides a transparent overlay on normal vision to achieve the augmented reality effect. The primary benefit of this approach is that the user’s normal vision is retained, not blocking out peripheral vision or introducing problems from the camera. The most obvious downside is that due to the display’s transparency, bright ambient light can reduce visibility. Also, if no eye tracking is integrated into the system, the augmented overlay will not line up with the user’s vision if they are not looking straight ahead. Additionally, the optics required for such a system to allow the user’s eyes to simultaneous focus on both the display and background would be extremely difficult and expensive to duplicate. Thus our implementation of the beam splitter optics does have the issue of not being able to focus on the display and background simultaneously.

### Product #2

Vuzix WRAP 1200AR [49]

The WRAP 1200 AR is an augmented reality headset with a package resembling a pair of sunglasses with a pair of cameras mounted on the front. It has a pair of LCD displays with magnifying optics to create the illusion of larger displays placed farther away. One advantage of this approach is a wider field of view than provided by the beam splitter approach. Vuzix claims a field of view equivalent to a 75” diagonal display at a distance of 10’. Applying some simple trigonometry, this works out to a 35 degree diagonal field of view. This is just over 50% higher than the BT-100’s 23 degrees. The ambient brightness problem is also eliminated because the display is not transparent and the glasses format makes it relatively easy to block out ambient light. The simultaneous focusing on overlay and background problem is solved because in this approach they are both on the display. Construction would also be simplified somewhat since there would be no need to support the beam splitter glass hanging down. A disadvantage of this approach is the loss of peripheral vision. Another is the loss of fidelity in vision, as the display cannot match the resolution of human vision and may suffer from camera issues such as white balance and exposure hunting, poor low light performance, lack of dynamic range, and latency. An additional camera problem which this commercial unit does not face but our more budget constrained implementation would is the lack of depth cues from having only one camera. The Raspberry Pi is quite incapable of handing two cameras with acceptable latency, and any system with that sort of processing capability would be well outside our budget. This approach also increases the system’s power consumption, both from powering the camera itself and the additional processing resources required to process the camera feed and copy it into the framebuffer.

## Project Packaging Specifications

For the packaging of the headset, we decided to go with the beam splitter display approach. A hard hat will be used as a base to which components will be mounted. Baseball caps, bicycle helmets, and motorcycle helmets were all considered but found unsuitable for reasons of either sturdiness, bulkiness, or ease of mounting additional hardware. An aluminum frame is bolted on to the hard hat which is used for mounting most of the other hardware. At the front of the head, a plastic enclosure will hold the display such that it is facing down and the bottom of the box is open. The LCD panel is glued down on top of an appropriately sized hole cut in the top of the enclosure. The purpose of this enclosure is to both block ambient light reflections and increase the total distance between the user’s eye and display to make focusing easier. The LCD is of a 16:9 aspect ratio with a 5” diagonal, giving an overall size of 4.3”x2.5”, and so the enclosure will be about 6”x4”x2”. This size of enclosure is conveniently stocked at local electronics stores. This gives a bit of extra room around the edges to prevent unwanted reflections. Mounted to the bottom of the box is a sheet of acrylic with a sheet of reflective window tint film applied. The acrylic is attached to the enclosure with a pair of hinges and to the aluminum rail with a pair of wires looped through holes in the acrylic and around the rails at the other end. This allows the angle of the reflector to be adjusted by changing how much wire loops around the rail. A Fresnel lens is mounted at the bottom of the display enclosure which increases the apparent size and distance to the display as seen by the user. The size increase is substantial but due to the long focal length of the lens, which is approximately one foot, the increase in apparent distance is not very large. This is because the formula for apparent distance of an object from a lens is 1/((1/real distance)-(1/focal length)) which works out to 1/((1/2)-(1/12))=2.5” in our case. A lens with a much shorter focal length would be needed to achieve our desired effect of making the display appear far enough away to be able to simultaneously focus on it and the background. However such a lens would introduce a large amount of barrel distortion which would need to be corrected for in software using an OpenGL shader program.

On the back of the headset, another plastic enclosure will hold the stack of a Raspberry Pi Model A, our custom PCB, and the battery. The antenna for our XBee will protrude from the top of this. The dipole antenna we are using works best when pointing straight up. This positioning of this hardware will help counterbalance the weight of the display assembly in front, making the headset sit more stably and comfortably on the user’s head. There will be cables leading out of this enclosure for video, display power, and the GPS antenna. The GPS antenna is mounted on top of the hat in order to provide a little bit of distance to mitigate any potential interference from other components. This enclosure will measure about 3”x3”x1.5”, with the 3x3 being set by our PCB size and the 1.5” being the approximate depth of the PCB and battery stack if we go with a 6600mAh battery (which should provide about 5 hours of usage).

The CCU packaging will consist of another plastic enclosure with a Pi, XBee explorer module, and graphic display (the same display as the headset uses) sitting inside. On top there will be a keypad for user input. Power will be supplied from an external USB wall wart, as everything can be run off of 5V. The enclosure will be about 4”x6”x2” which leaves just enough room to mount both the number pad and the display on top. The XBee antenna will then stick out the side of the enclosure.

## PCB Footprint Layout

For the footprints of the major components in our design, we were actually left with very little choice in terms of the footprints. The STM32F405RGT6 microcontroller is only available in one package, which is LQFP64. The XBee radio transceiver, STEVAL-MKI108V2 9DOF sensor, and Venus 638FLPx are all mounted on small daughter boards connected to the main board via standard 0.1” pin headers. For the XBee, this is because it is the only package it can be purchased in. The GPS and 9DOF are both available only in LGA packages, which are extremely difficult to solder. A preliminary layout (performed by Stephen Carlson) puts the board size at 3”x3”.

## Summary

The packaging of our augmented reality system will consist of two main components, the headset and CCU. The headset will be using a piece of beam splitter glass to reflect an LCD display and overlay it on the user’s field of view. The display, mounted in an antireflective enclosure, and beam splitter optic assembly will be mounted to the front of the hat, with an adjustable angle for the beam splitter to accommodate different user head sizes. The PCB stack and battery will be placed in a separate enclosure mounted to the rear of the hat in order to balance out the weight. The CCU packaging will be a plastic enclosure with a number pad and LCD mounted on top and antenna sticking out the side.

# Schematic Design Considerations

## Introduction

An augmented reality system with multiple wireless headsets and a central control unit (CCU) is being designed. The headset board comprises a microcontroller and peripherals that serve sensing, wireless communication, and rendering purposes. The CCU, on the other hand, performs processing on a Linux-capable Raspberry Pi, utilizes wireless communication and user interface peripherals. Because the CCU components are primarily a subset of the headset components, the CCU will use the PCB derived from the headset schematic/layout. No exotic circuit techniques have been implemented in the schematic; following the manufacturer’s recommendations with regard to capacitors and resistors is indicated to meet the system’s needs as a whole.

## Theory of Operation

The augmented reality system comprises multiple battery-powered headsets controlled by a microcontroller and one Raspberry Pi, and a single central control unit controlled by one Raspberry Pi.

Upon insertion of a 3.7V lithium ion battery into the headset, or the contact closure of the battery’s on/off switch, the battery’s voltage is connected to the input of 4 power supplies, the input of the battery “fuel gauge”, and the output of the battery charger. A polarized battery header will be used to prevent the physical insertion of the battery in the wrong polarity direction. The single battery powers all components of the headset except for the battery charger, which is optionally powered by an external 5V source in the form of a USB connection. Three of the four power supplies are low dropout (LDO) regulators which altogether supply 3.3V to the IMU sensors, GPS, Raspberry Pi, microcontroller, and XBee radio. The fourth power supply is integrated in the LCD package, and accepts the battery’s 3.7V directly. Once the IMU sensors are powered, they are capable of transmitting raw sensor data to the microcontroller for processing via I2C protocol [34]. The battery fuel gauge will similarly communicate the battery’s charge to the microcontroller via I2C [22]. The Raspberry Pi will communicate with the microcontroller via SPI, and both the GPS and XBee modules will communicate with the microcontroller via the USART protocol [35, 6]. The following table summarizes the communication protocols of microcontroller-connected components:

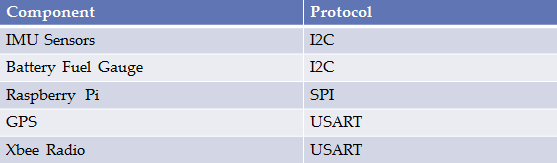


Figure 8.1: Communication Protocols of Electrical Components Used

The microcontroller will process the IMU sensor and GPS data (e.g. Kalman filtering) and battery fuel gauge measurement via software, and hand off this information in a more palatable form to the Raspberry Pi for rendering on an LCD that communicates exclusively with the Raspberry Pi. The microcontroller will also likely send the processed IMU sensor and GPS data to the CCU via the XBee radio. However, this functionality is software-defined. Throughout operation, the microcontroller will also receive instructions from the CCU via the XBee radio and interpret the packets based on software loaded via the 6-pin microcontroller debugger or the USB port. For convenience of mobile code-loading and debugging of the microcontroller, the 6-pin debugger will likely only be used for the required initial code-loading (note the ‘DO NOT STUFF’ directive on the schematic) by physically pressing the debugger connector onto the board for a moment. After the initial code loading, the differential communication lines of the USB port will serve as the debugging path. Note that the USB port serves two uses: regular debugging of the microcontroller, and charging of the headset battery. Also note that the device can be run and charged simultaneously.

The central control unit PCB will be powered by a wall supply. The wall supply directly powers the Raspberry Pi, which in turn powers all other devices on the CCU (XBee radio, user interface LCD/pushbuttons). The schematic and PCB layout are designed in such a way that the central control unit can utilize the headset PCB, even though the CCU uses significantly fewer components.

Indicator LEDs and ‘spare signal’ vias will be important during the debugging process. Currently, only a battery and battery charger LED are included in the schematic. However, once the PCB has been routed with the current components (which have higher priority and less mobility in terms of where they can be placed), more LEDs and signal pad-outs will be added in accessible/visible locations.

Although most components on both the headset and CCU boards could operate from voltage within the range 2.2-3.6V, the GPS module [6] had an operating voltage range of 2.8-3.6. Given that 3.3V is roughly in the center of these ranges, and that 3.3V is a common supply voltage for small components which could be used if we needed to swap parts later, all schematic components on the headset PCB (with exception of the battery charger and LCD power supply) are powered by 3.3V to avoid the need for level translation. Apart from voltage supply considerations, the operating speed and mode of most devices are configured in software. The microcontroller, for example, has a software-configurable operating speed. The 8MHz crystal that supports the microcontroller was chosen among two manufacturer-recommended oscillator circuit configurations in lieu of a lower-frequency option [33]. The higher-frequency base clock speed was chosen in order to allow for a faster maximum clock speed (in this case, 168MHz), which will increase the time in which sensor data can be processed and handed off to the Raspberry Pi for rendering. The additional power that accompanies a faster clock speed are negligible in this case as the power consumption of the microcontroller as a whole (~30mA) will be dwarfed by the power consumption of the Raspberry Pi and headset LCD alone (together, >1A).

## Hardware Design Narrative

The microcontroller used on the headset (Appendix C-2), comprises three primary ports: A, B, and C. Because the oscillator will physically sit near the pad out of port C, and the oscillator creates relatively high noise, it would be desirable to locate all the connections to the microcontroller at ports A and B. However, in order to reduce the distance and complexity of the PCB routing, it eventually became necessary to move some of the component connections to port C. Given that the signals migrated to port C are all digital, the migration is more permissible than if the signals were analog. The following table summarizes which devices are connected to which ports:

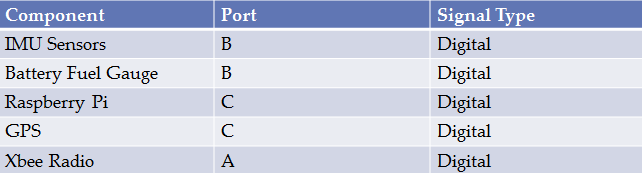


Figure 8.2: Microcontroller Port Assignments

These ports will be used for digital communication protocols as indicated in Figure 8.1. Per the manufacturer’s recommendations, a single 1uF bypass capacitor was placed on the microcontroller’s VBatt pin, 2x 2.2 uF capacitors are placed on the microcontroller’s internal regulator, and 10x .1uF bypass capacitors are applied to the microcontroller’s power pins. Because the manufacturer’s recommendations on schematic and layout configuration of the microcontroller will be followed, it is anticipated that a simple ground pour will appropriately stabilize the power/ground rails, and that no special components or routing considerations will need to be dealt with to avoid, for example, ground loops.

All other components were similarly configured in the schematic and layout according to manufacturer’s recommendations. For example, both the battery charger and fuel gauge (Appendix C-5) use resistor and decoupling capacitor configurations as specified in their respective datasheets for supporting a 3.7V lithium ion rechargeable battery. The bulk 22uF electrolytic capacitors (Appendix C-6) were suggested by the power supply data sheet as well.

On the CCU board, a single wall supply will indirectly power the entire board through the Raspberry Pi. The headset board, on the other hand, uses four power supplies. One power supply is already integrated into the LCD, and therefore must remain. However, components were split among the other three power supplies in consideration of noise and current consumption. The STEVAL board that contains the analog IMU sensors (Appendix C-3) receives a dedicated power supply in order to decouple the IMU sensing from noise caused by radio transmission and graphics rendering in the GPS/Xbee radios and Raspberry Pi respectively. The microcontroller, GPS, Xbee, and Raspberry Pi are split up among two 500mA power supplies as indicated in Figure 8.3 based on their respective current consumption; the Raspberry Pi can consume up to 400mA during peak processing, which is more current than the other three devices combined during peak current draw.

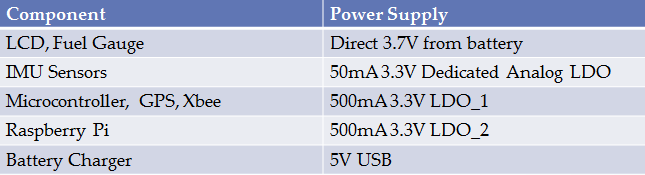


Figure 8.3: Headset Power Supply Distribution

## Summary

The necessary circuitry and support components have been organized in the schematic screenshots (Appendix C) to satisfy the block diagram (Figure 1.1). Logical subsections of the PCB include: microcontroller, IMU sensors, GPS/Xbee radios, power supplies, battery charging/monitoring, and interfacing pads (e.g. Raspberry Pi/LCD headers, debugger port). Manufacturer recommendations were exclusively followed, and common sense was applied where manufacturer recommendations were not available: deciding to power the board with 3.3V power supplies, moving digital communication traces to the microcontroller’s port C despite potential noise emission from the proximal oscillator (Figure 8.2), and dividing up power supplies to accommodate both low noise and high current consumption requirements (Figure 8.3).

# PCB Layout Design Considerations

An investigation of the capabilities of the particular board design manufacturer which will be used for prototype production reveals the minimum limits for PCB design. Advanced Circuits quotes the limits shown in Table 9.1 for standard spec PCBs. [2]. For optimal manufacturing yield, the PCB was routed with wider tracing and spacing and a larger drill size. In addition, common pitfalls such as acute angles and sharp corners were also to be avoided.

Table 9.12: PCB specifications for low-cost "standard spec" boards from Advanced Circuits [2]

|  |  |  |
| --- | --- | --- |
| Metric | Minimum | Chosen for Manufacturability |
| Trace Thickness | 5mil | 10mil |
| Trace Spacing | 5mil | 8mil |
| Drill Diameter | 10mil | 15mil |
| Drill Tolerance | 5mil |  |

The first set of constraints on the PCB design is the form factor and connector placement. To allow the custom PCB to properly mate with the Raspberry Pi, the outer PCB dimensions and 26-pin I/O header placement had to exactly match the motherboard mechanical specifications [31]. The external connectors for USB charging, battery power, and radio antennas must also be facing outwards on the border of the PCB to fit in the target packaging constraints.

To reduce electromagnetic interference (EMI) received, the major radio frequency emitters were placed on the bottom left of the board as far away from the vulnerable inertial measurement unit (IMU) as possible in this design. As the GPS unit has an external antenna [**35**] which moves its noise emission far away from the PCB, the XBee radio and antenna [17] was placed in the far corner of the board with the GPS in the middle. Unintentional EMI emissions were also controlled by placing numerous decoupling capacitors on the microcontroller as recommended by SGS-Thompson [**6**] and locating the external oscillator as close as physically possible to the microcontroller clock pins.

## PCB Layout Design Considerations – Microcontroller

Several of the components of this project, most notably the USB boot loader described in the microcontroller datasheet, depend on a stable clock source. Therefore, an external crystal oscillator in a through-hole package was utilized with the recommended load capacitance of 20pF [36]. As the traces leading to the crystal also add impedance, the EAGLE “*run length-freq-ri*” tool was used to match the oscillator trace lengths within 8mil. The crystal was also placed as close as possible to the microcontroller while still allowing a component-free area around the oscillator to limit noise coupling. Similarly, the same method was used to match the USB data traces to significantly less than the stated 50mil length tolerance [5].

Part placement near the microcontroller was also a major concern. In order for the PCB to be physically possible to route, pin assignments with multiple equivalent options on the microcontroller were chosen to limit the number of crossing signals. Careful effort was also spent in placing parts to minimize the number of traces looping around the microcontroller and interfering with power routing as shown in Figure 9.1. After placement was finalized, the most convenient spare I/O pins were brought out to pads and a spare serial port was connected to an unpopulated header for debugging and future expansion. Two spare LEDs also aid in debugging.

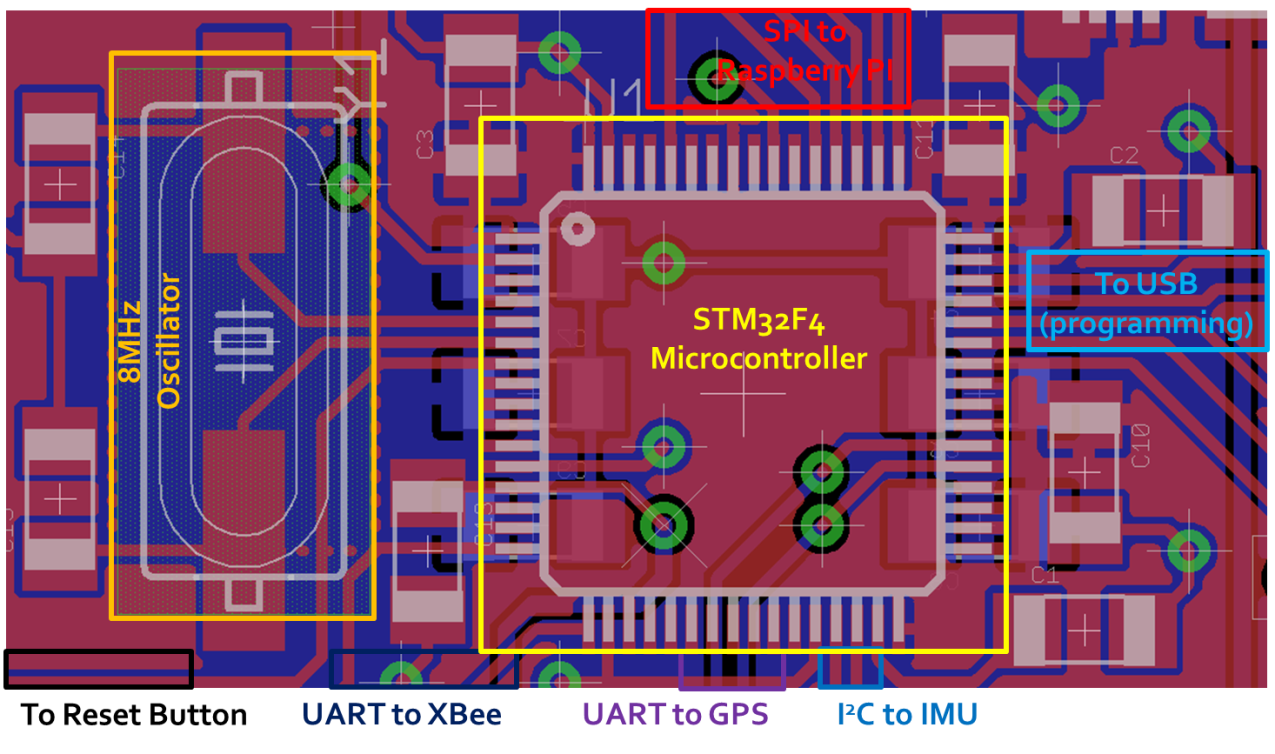


Figure 9.1: Signal routing near microcontroller showing oscillator circuit and decoupling capacitance

To ensure optimal power supply to the microcontroller, a large power and ground plane along with twelve decoupling capacitors were used. One of these capacitors was placed near each power supply pin, with the six remaining capacitors going underneath the microcontroller on the back of the board to keep them as close as possible. Additional 1 uF and 2.2 uF ceramic capacitors were placed according to the manufacturer recommendations to decouple the internal core voltage regulator. The power and ground planes were monitored during routing to ensure that power could come from all four directions of the board to avoid a weakly connected copper pour “island.” As the analog to digital converter on the microcontroller is not used, the routing of the analog supply and analog ground pins was less critical.

## PCB Layout Design Considerations - Power Supply

The power supply design for this board must cope with a large, fluctuating load and power management concerns. Total current could be up to 2 A during radio transmission events as evidenced by design constraint analysis, so power supply traces were widened to 80mil or more to reduce resistive losses. Power and ground planes were emphasized to simplify power routing, and if power needed to switch sides on the board, multiple large vias were used to share the load. The bottom of the board was devoted to a ground plane with few signal traces to reduce the size of noise-inducing ground loops. Power planes were also used to reduce the resistance of power traces while providing thermal dissipation paths for the low-dropout regulator circuits. A sample of the result is shown in Figure 9.2.

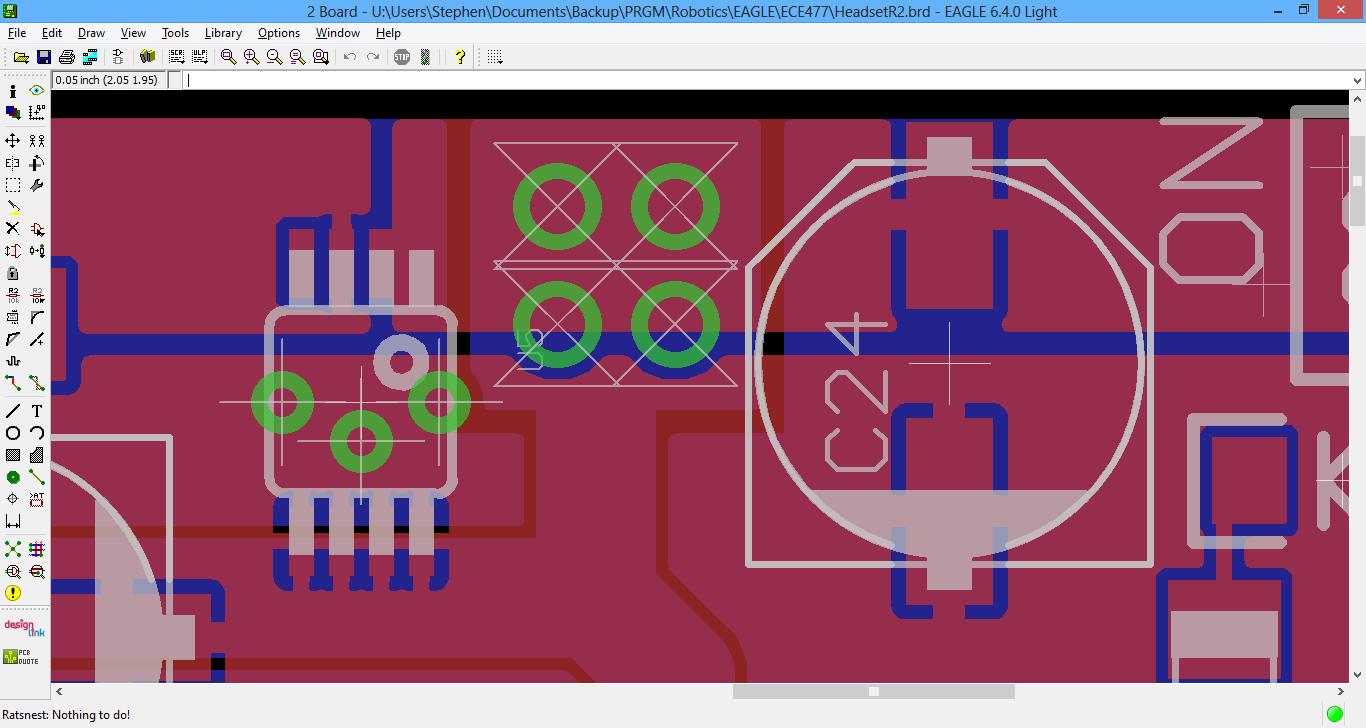


Figure 9.2: Close-up of power supply showing power transfer vias, bulk capacitance, and power plane usage

Bulk capacitors were chosen and placed on the board to handle the large inrush currents of the microcontroller and radios. Solid bulk capacitors with low equivalent series resistance (ESR) were placed at the board input and after each voltage regulator. Additional manufacturer recommended ceramic capacitors for stability were also placed close to each low-dropout regulator [23]. The step-up converter used to generate 5V for the display [29] was located as far away from the rest of the circuit as possible to minimize coupling of switching noise, but its onboard capacitors eliminate the need for additional bulk or decoupling capacitance. Bypass capacitance already used near other parts of the circuit handles power supply decoupling.

To reduce shared impedance noise coupling between the IMU and the digital circuits, a separate low-dropout regulator with outstanding ripple rejection was dedicated for IMU power supply and placed close to it. Although the IMU communicates digital results over I2C, the onboard sensor still uses an analog design where noise could disrupt the computed orientation [**34**]. As this regulator does not have to power other loads, switching noise is reduced, and any noise picked up on the battery supply trace from external sources is not coupled to the sensors.

# Software Design Considerations

## Introduction

We propose an Augmented Reality Simulator that allows at least on user to play an electronic game in a mobile, outdoor environment. The device is split into two primary physical parts, but there are three parts for software. The central control unit, the headset, and the GPU will all have software components. The headset routes information to the GPU and the GPU renders images and objects based on that information. The central control unit tells the headset where images and objects are located and provides a user interface. This report will focus on the software on the headset, but will address the software on all components.

The software design must address how data will flow between all hardware and software components. The headset must accurately transform raw inputs from the sensors into values that can be used meaningfully by other software components. The GPS accuracy poses a challenge for reliably detecting simulation collisions. It also poses a problem for accurately drawing the location of objects for the user. The GPU must accurately draw objects and have a shared method of referring to objects specified by the central control unit. The central control unit must provide a user interface and manage the states of a wireless headset.

## Software Design Considerations

A state machine approach was chosen for the headset versus a polled loop style or purely interrupt driven program. This choice was made because the headset must cycle through states based on wireless data received from the central control unit. The diagram in Figure 1 below shows that states that the headset must go through to successfully demonstrate the required functionality. The headset must begin by broadcasting its presence so that it may be selected for inclusion in the simulation by the central control unit. It must then load static data from the central control unit to the GPU for later rendering. Once this has been completed it may finally enter the simulation phase. Flowcharts describing the operation of the central control unit and the headset main module can be seen in Figures 10.3 and 10.4.

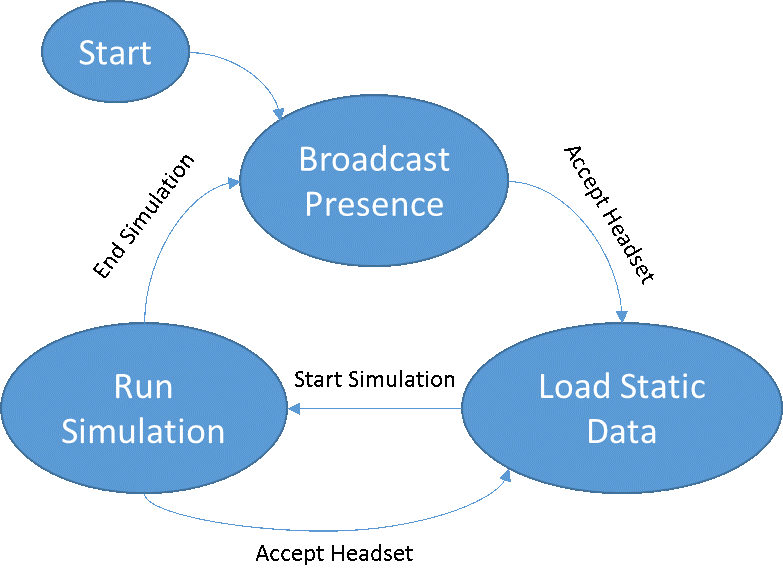


Figure 10.3: State machine for headset operation.

The communication between the central control unit and the XBee [22] is through UART. In the same manner, the communication between the STM32F4 and the XBee is through UART. The communication between the GPU (raspberry pi) and the STM32F4 is through SPI. The GPU will act as the master and the STM32F4 will act as the slave. The IMU communicates to the STM32F4 through I2C clocked at 400 KHZ (fast mode) [40] and the Venus638FLPx GPS Receiver [35] communicates through UART. The baud rate of the XBee modules is set for 56700 on their corresponding UART ports. The baud rate of the GPS is configured for 38400.

The memory configuration of the headset can be seen in the Figure 2. This image was taken from the programming manual for the STM32F4 [39]. SRAM is mapped from 0x20000000 to 0x20020000 for a total of 128K. The stack will be stored at location 0x20020000 because it is a descending stack. Static and global variables will be stored at location 0x20000000. There are currently no plans to use a heap. All of the memory necessary will be determined at compile time for the headset.

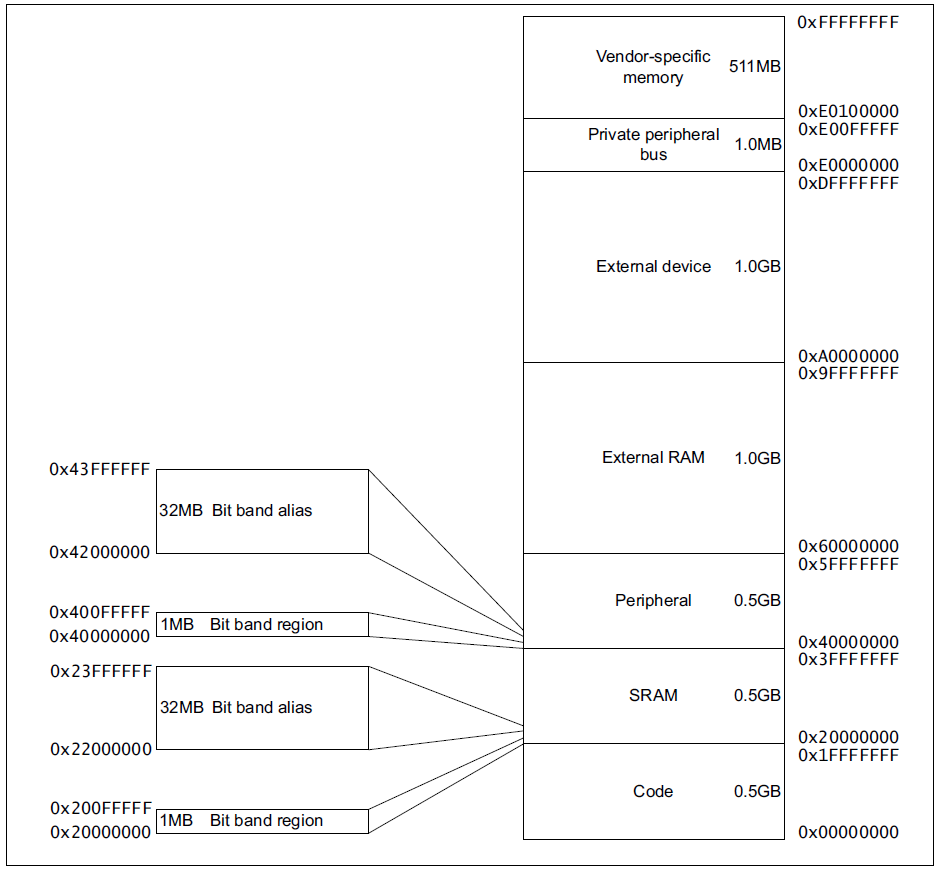


Figure 10.4: Memory map from STM32F4 Manual [39]

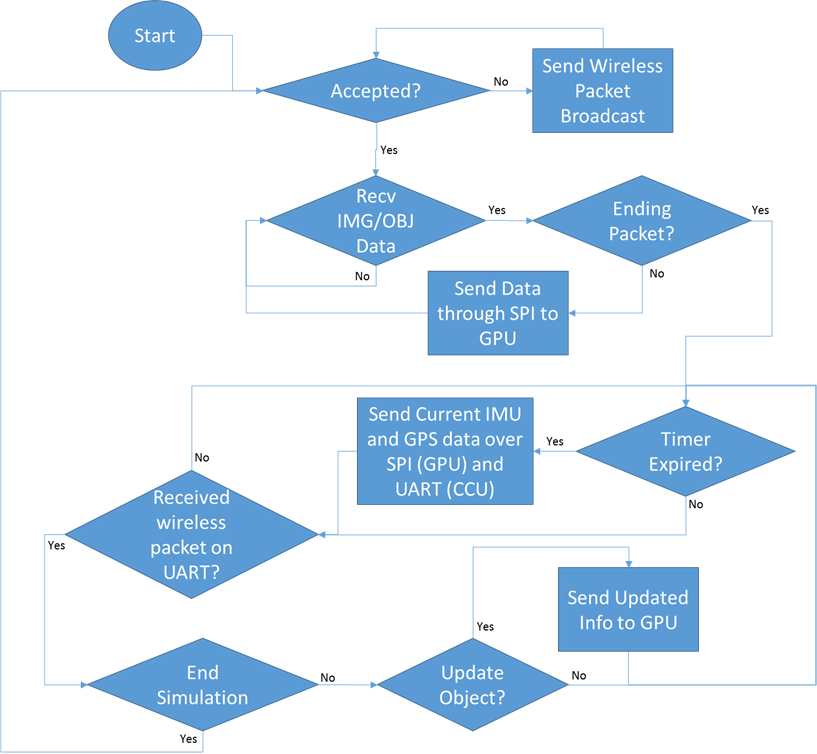


Figure 10.5: Flowchart showing headset main module flow.

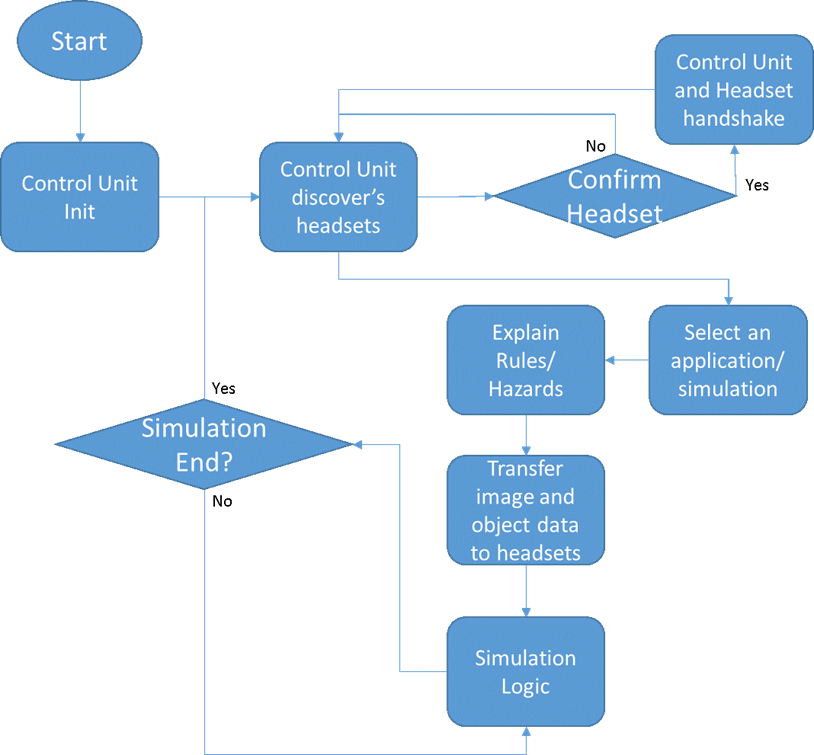


Figure 10.6: Flowchart showing CCU user interface flow.

## Software Design Narrative

Figure 10.5 shows the hierarchical arrangement of the various code modules included in our design. The code modules in our design are: the central control unit user interface, the central control unit simulation, the headset main module, the headset GPS IRQ, the headset IMU IRQ, the headset battery IRQ, the headset XBee IRQ, and the headset GPU.

The central control unit user interface is written in Python Tkinter [30]. Tkinter was chosen because a complex user interface is not needed, and because the author is familiar with the software. The user interface launches on startup of the central control unit and begins looking for available headsets to join in simulations. The user is able to choose available headsets and add them. The simulation allows a user to select a simulation to run. After selecting a simulation, the rules and hazards are explained to the user. The GUI will then launch the simulation and wait for it to end.

The central control unit simulation will first load image and object data to all headsets through wireless communication. It will then proceed to send updates about the position of the image and object data to the user. Updates about the headsets position will be periodically processed and collision detection will be performed to determine simulation events to be triggered. Due to limited accuracy of the GPS, virtual objects will be made large (approximately 2 meters) for the purposes of detecting collisions. Progress has been made in defining the configuration settings for the [XBee](https://github.com/snowpuppy/augreality/tree/master/ccu/test). Most of the work for this module remains.

The headset IMU, XBee, and GPS IRQs pull data from the respective units and pool that data into a FIFO buffer. The headset main module periodically sends data collected for the GPS location, IMU orientation, and battery status to the GPU and to the central control unit. When packets are received, the headset main module transitions between the following states: Broadcast Presence, Load Static Data, and Run Simulation. The Broadcast Presence state causes the headset to repeatedly send out a message to the central control unit advertising the headset's availability. The Load Static Data state will transfer data from the CCU to the GPU for storage. The Run Simulation state will wait for object update messages from the central control unit and send the updates to the GPU. The IMU, GPS, and battery IRQs have been completed. Data has been successfully extracted from all of these sensors through the STM32F4. The code can be found [here](https://github.com/snowpuppy/augreality/tree/master/headset/test). STMicroelectronics libraries were used to extract the data [38].

The GPU software was written for the Raspberry Pi using OpenGL ES 2.0 [27]. There are several major software components running on the Pi: a SPI thread to communicate with the microcontroller, a file loader to dynamically load a simulation from a file, input file handling for debug purposes, rendering loop which actually draws the virtual world. For SPI, an LGPL licensed library called Wiring Pi [14] is used to initialize the Pi’s SPI hardware and read and write bytes. All transfers are initiated by the Pi sending a byte indicating the type of data it wants from the microcontroller: i.e. a file, sensor data, or XBee packet. After this the Pi waits for a response, which starts with a byte indicating the number of bytes in the packet. This is checked against the expected size of the data type requested to detect data corruption. Then the specified number of bytes is read into a buffer and stored into the relevant variables with typecasting. The file loader loads a level file in a simple custom text format specifying all objects in the scene with an ID number, initial location, orientation, visibility flag, and model filename. The game objects are stored into an array of a custom object type holding all of this information. To reduce memory usage and speed up load times, certain common models (like the wall) are only loaded once with each object referencing the same model data in memory. Input handling uses the GPL licensed SDL library [32] to check key presses once every frame and execute the appropriate action, like exiting the application or moving the 3D camera. The rendering loop uses a GPL licensed OpenGL ES 2.0 wrapper library called piNGL [18] and is based on a sample application included with the library. Initially an attempt to write a custom OpenGL engine was made and abandoned due to very high difficulty. A 3D camera is created based on GPS and orientation data read in from the SPI thread. Then the array of game objects is looped through, and if the visibility flag is set it is drawn in the appropriate location. After this, a separate camera is created for 2D drawing. Then 2D icons for the battery and RSSI indicators are drawn to the screen as textured squares.

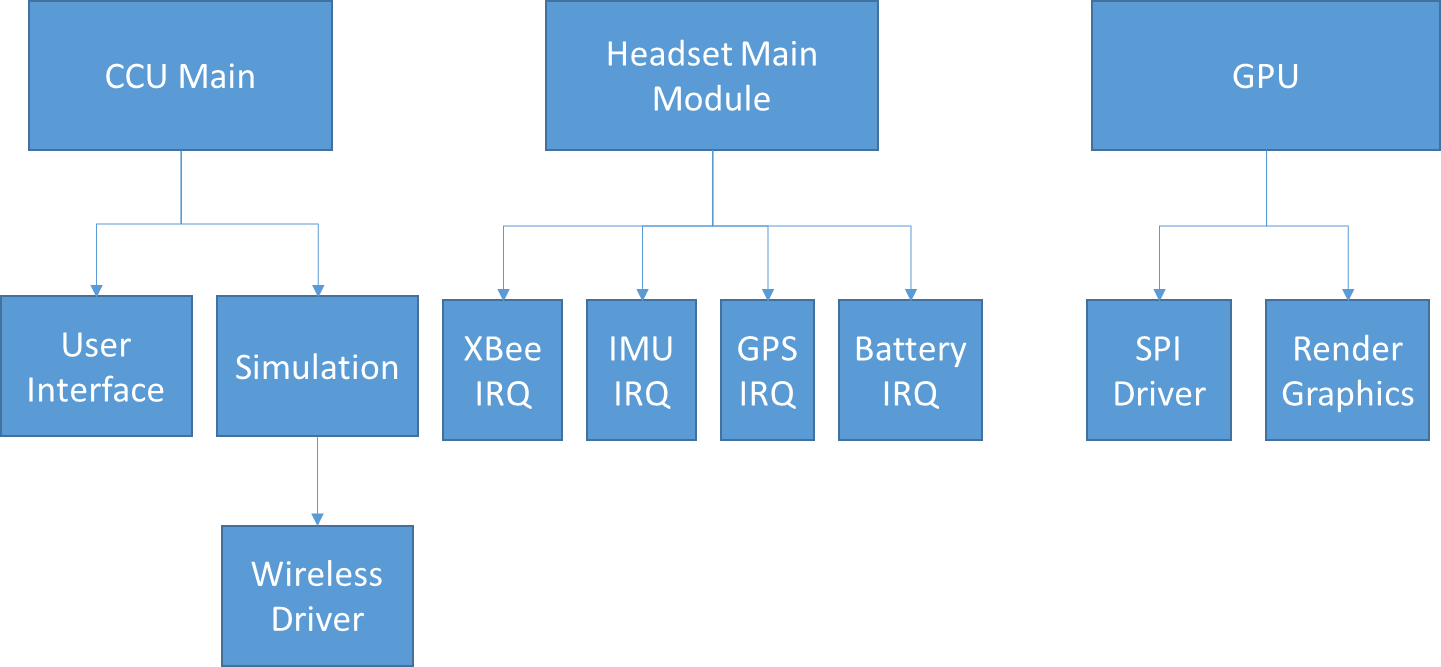


Figure 10.7: Hierarchical diagram showing the relationships between software modules.

## Summary

Our proposed project, an augmented reality simulator, aims to display an image to the user that adapts to their position and head orientation. To accomplish these goals, GPS and IMU data is harvested and routed to the GPU and the central control unit. Updated object position information is routed from the central control unit to the headset to allow objects to change their position as well. A user interface wraps all of the functionality.

# Version 2 Changes

Hardware

* Improve PSU for distinct CCU box (Figure 1.2) due to intermittent power failures. However, our next iteration would likely not utilize a distinct CCU device (see Software changes).

Packaging

* Add at least one general-purpose pushbutton to the headset in order to, for instance, toggle color schemes of a simulation, or trigger a game action like shooting. (Additionally, some form of user input will be required if the CCU functionality is ported to one of the headsets.)
* Assemble at least one more headset (currently only have one working headset).
* Make the SD card slot/connection more physically secure to avoid occasional software ‘lock ups’ due to perturbation to the SD card connection on the headset.
* Make headset packaging more aesthetically pleasing, compact, and sturdy in general.

Software

* Move the CCU functionality from the CCU box to the headset (specifically, on the headset’s Raspberry Pi). This would require some modifications to both the CCU and headset code.
* Add more simulations such as virtual reality tours or additional games (currently only have one ‘Pac-Man’ game emulation).
* Make the CCU logic extensible to the addition of other headsets (currently includes some lack of extensibility due to the fact that we only have interfaced with one headset, and thus were able to ‘get away’ with it).
* Improve the organization and ‘cleanliness’ of the CCU code to improve human readability.
* Improve the IMU filtering on the headset microcontroller (smoother angle updates, lower latency updates).
* Improve the GPU model loader on the headset Raspberry Pi for quicker headset startup.

# Summary and Conclusions

Concerning the engineering aspects of our project, we accomplished what we set out to: PCB design/population/testing, implementing a packet protocol for CCU/headset communication, packaging a headset that reflects an LCD into the user’s field of view, and rendering the LCD output based on IMU/GPS data and game logic (i.e. rendered arbitrary 3D objects in an arbitrary location in 3D space).

Beyond learning the joys of living in the senior design lab, we learned that starting early (in our case, weekly meetings at the beginning of summer 2013) and meeting regularly throughout the semester (at least once a week) prevented us from falling behind. Some of us were also surprised about the apparent lack of major issues or obstacles in implementing our project after we had planned things out (e.g. PCB design and component choice was fine, no major issues with software latency). Individually, we also learned some specific technical details. Steve Ellis learned about object serialization and the difficulty of using OpenGL. Alec Green learned that the most complicated filter design (e.g. Kalman filter) does not necessarily provide better results than a simpler filter (basic low-pass filter). Thor Smith learned about using power tools to manipulate packaging materials and how to get distinct processes to communicate with each other on a microprocessor. Stephen Carlson, on the other hand, did not feel like he learned significant aspects of PCB design and electrical component selection, perhaps because his recent summer internship had him performing a similar task.

# References

[1] Abdollahi et al., “[Head Mounted Information Systems and Related Methods](http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CDkQFjAA&url=http%3A%2F%2Fpatentimages.storage.googleapis.com%2Fpdfs%2FUS20130044043.pdf&ei=wLJzUvfcDaXA2AWf9IGIBQ&usg=AFQjCNFM5zGL8rr4dAYFk0op5QYhD3un3A&sig2=IyQ3GM7FE-4fvkUlQE6bzw&bvm=bv.55819444,d.b2I&cad=rja),” U.S. Patent 0 044 043, Feb 21, 2013.

[2] Advanced Circuits, “Standard Spec vs. Custom Spec PCB”, 4pcb.com, [Online] Available: <http://www.4pcb.com/standard-custom-order-pcbs/> [Accessed Oct. 1, 2013]

[3] Antenna GPS 3V Magnetic Mount SMA [Online] Available: <https://www.sparkfun.com/products/464>

[4] Connexions, “The Environmental Impact of the Manufacturing of Semiconductors” 2009. [Online] Available: <http://cnx.org/content/m14503/latest/>

[5] Cypress Semiconductor Corporation, “AN1168: High-Speed USB PCB Layout Recommendations”, Nov. 2011 <http://www.cypress.com/?docID=32407> [p. 3]

[6] Digi International, “XBee-Pro 900HP,” Xbee Pro 900HP datasheet, Aug. 2013 <http://www.digi.com/pdf/ds_xbeepro900hp.pdf>

[7] Electronics Weekly, “How can manufacture of TFT-LCD displays be eco-friendly?” 2011. [Online] Available: <http://www.electronicsweekly.com/news/manufacturing/how-can-manufacture-of-tft-lcd-displays-be-eco-friendly-2011-11/>

[8] Environmental Graffiti, “The Revolution in Rechargable Batteries: Lithium Iron Vs Lithium Ion” 2010. [Online]. Available: <http://www.environmentalgraffiti.com/energy/news-environmentally-rechargeable-lithium-iron-battery-better-lithium-ion-battery>

[9] Environmental Resource Site for SME Manufacturers, “Overview of Printed Circuit Board Industry” 2007. [Online]. Available: <http://www.sme.greenmfr.org/english/printed/printed.html>

[10] Epson Moveria BT-100 Wearable Display – Product Information [Online]. Available: [http://www.epson.com/cgi-bin/Store/jsp/Product.do?sku=V11H423020#0](http://www.epson.com/cgi-bin/Store/jsp/Product.do?sku=V11H423020%230)

[11] G. Novacek, “Designing for Reliability, Maintainability, and Safety: Part 1---Getting Started,” Circuit Cellar 125, December 2000

[12] G. Novacek, “Designing for Reliability, Maintainability, and Safety: Part 2---Digging Deeper,” Circuit Cellar 126, January 2001

[13] G. Novacek, “Designing for Reliability, Maintainability, and Safety: Part 3---A Sure Thing,” Circuit Cellar 129, April 2001

[14] Gordons Projects, “WiringPi”, 2013 <https://projects.drogon.net/raspberry-pi/wiringpi/>

[15] Grand et al., “[Method and Apparatus for Displaying Images on Reflective Surfaces](http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CC4QFjAA&url=http%3A%2F%2Fpatentimages.storage.googleapis.com%2Fpdfs%2FUS20060250574.pdf&ei=D7RzUpL3JIHN2AX1roGYBQ&usg=AFQjCNHBChgA3ATHF8aifdty0soTwYc0WQ&sig2=BEmFFU9XHpeMj1fnY970aw&bvm=bv.55819444,d.b2I&cad=rja),” U.S. Patent 0 250 574, Nov 9, 2006.

[16] Great Power Battery Co., Ltd, “广州鹏辉电池有限公司”, P-1S5045BC datasheet, Jan. 2012, <http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/BatteryProtection.pdf>

[17] GS-Thomson Microelectronics, “ARM Cortex-M4 32b MCU+FPU”, STM32F405RG datasheet, Sep. 2011 [Revised Jun. 2013] <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037051.pdf>

[18] Jon Macey Computer Animation Pages, “piNGL”, 2013 <http://nccastaff.bmth.ac.uk/jmacey/GraphicsLib/piNGL/index.html>

[19] Lenntech, “Cobalt” 2013. [Online]. Available: <http://www.lenntech.com/periodic/elements/co.htm>

[20] Lithium Ion Battery Pack – 3.7V 6600mAh [Online] Available: <http://www.adafruit.com/products/353>

[21] M. Perenson (2013, May 22) Augmented-Reality Glasses: Startup’s Vision Could Change Gaming [Website] Available: <http://www.technewsdaily.com/18143-augmented-reality-glasses-startups-vision-could-change-gaming.html>

[22] Maxim, “Compact, Low-Cost 1S/2S Fuel Gauges with Low-Battery Alert”, MAX17043 datasheet, Sep. 2009 [Revised Aug. 2012] <http://datasheets.maximintegrated.com/en/ds/MAX17043-MAX17044.pdf>

[23] Micrel Inc., “500 mA-Peak Output LDO Regulator”, MIC5219 datasheet, Jun. 2009 <http://www.micrel.com/_PDF/mic5219.pdf>

[24]Microchip, “Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers”, MCP73831/2 datasheet, Nov. 2012, <http://ww1.microchip.com/downloads/en/DeviceDoc/20001984F.pdf>

[25] Micrel, “500mA-Peak Output LDO Regulator”, MIC5219 datasheet, June 2009, <http://www.micrel.com/_PDF/mic5219.pdf>

[26] Olsson, et al., "Wearable device with input and output structures" U.S. Patent 20130044042, Aug 18, 2011

[27] OpenGL, “OpenGL Software Development Kit”, OpenGL Documentation, 2012 <http://www.opengl.org/sdk/docs/>

[28] PCB Prototypes [Online] Available: <http://www.custompcb.com/faq.htm>

[29] Pololu Robotics and Electronics, “5V Step-Up Voltage Regulator U3V12F5”, pololu.com, [Online] Available: <http://www.pololu.com/catalog/product/2115> [Accessed Sep. 29, 2013]

[30] Python Software Foundation, “Tkinter – Python interface to Tcl/Tk”, Python Documentation, Oct. 2013 <http://docs.python.org/2/library/tkinter.html>

[31] Raspberry Pi Foundation, “Mechanical Profile – 3D models”, raspberrypi.org, [Online] Available: <http://www.raspberrypi.org/phpBB3/viewtopic.php?f=2&t=4402> [Accessed Oct. 3, 2013]

[32] SDL, “About SDL”, 2013 <http://www.libsdl.org/>

[33] SGS-Thomson Microelectronics, “ARM Cortex-M4 32b MCU+FPU”, STM32F405RG datasheet, Sep. 2011 [Revised Jun. 2013] <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037051.pdf>

[34] SGS-Thomson Microelectronics, “L3GD20 and LSM303DLHC 9-axis module for a standard DIL socket”, STEVAL-MK108V2 datasheet, Nov. 2011 <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/DM00041389.pdf>

[35] SkyTraq Technology, “Venus638FLPx GPS Receiver”, Venus638FLPx datasheet, Feb. 2010 [Revised Jan. 2011] <http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/GPS/Venus/638/doc/Venus638FLPx_DS_v07.pdf>

[36] SparkFun Electronics, “Crystal 8MHz”, sparkfun.com, [Online] Available: <https://www.sparkfun.com/products/538> [Accessed Oct. 2, 2013]

[37] STMicroelectronics, “ARM Cortex-M4 32b MCU+FPU,…”, STM32F405xx and STM32F407xx datasheet, June 2013, <http://www.st.com/web/en/resource/technical/document/datasheet/DM00037051.pdf>

[38] STMicroelectronics, “STM32F4 DSP and standard peripherals library, including 82 examples for 26 different peripherals and template project for 5 different IDEs”, STSW-STM32065 peripherals library, Oct. 2013 [http://www.st.com/web/en/catalog/tools/PF257901#](http://www.st.com/web/en/catalog/tools/PF257901)

[39] STMicroelectronics, “STM32F3xxx and STM32F4xxx Cortex-M4 programming manual”, PM0214 Programming manual, Sept. 2012 <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/programming_manual/DM00046982.pdf>

[40] STMicroelectronics, “STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx advanced ARM-based 32-bit MCUs”, RM0090 Reference manual, Sept. 2012 <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/reference_manual/DM00031020.pdf>

[41] ThermoAnalytics, “HEV Vehicle Battery Types” [Online]. Available: <http://www.thermoanalytics.com/support/publications/batterytypesdoc.html>

[42] Thomas, et al. (2002). "First Person Indoor/Outdoor Augmented Reality Application: ARQuake" Personal and Ubiquitous Computing. [Journal]. Available: <http://www.clearchain.com/blog/images/2008/11/Thomas-puc-2002.pdf>

[43] U.S. Department of Defense, “Reliability Prediciton of Electronic Equipment,” Government Printing Office, Washington D.C., Tech. Rep. MIL-HDBK-217F, 1991

[44] Unionfortune, “585460 Li-Polymer Battery Packs Specification.”, 585460 2000mAh datasheet, March 2006, <http://www.sparkfun.com/datasheets/Batteries/UnionBattery-2000mAh.pdf>

[45] University of Minnesota, “Water Conservation Opportunities for a Printed Circuit Board Manufacturer” 2009. [Online] Available: <http://www.mntap.umn.edu/mach/resources/aci-it15.htm>

[46] Valdes et al., “[Method, System and Device for Augmented Reality](http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&ved=0CEQQFjAD&url=http%3A%2F%2Fpatentimages.storage.googleapis.com%2Fpdfs%2FUS20020167536.pdf&ei=K7xzUon6AsrQyAGWuoHQCQ&usg=AFQjCNFo9yFvoYFZqS0BKkAFKc5qTuig7Q&sig2=yslOkg8pP4HNkx1S87AGUg&bvm=bv.55819444,d.aWc),” U.S. Patent 0 167 536, Nov 14, 2002.

[47] Wikipedia, “Acrylonitrile butadiene styrene" 2013. [Online]. Available: <http://en.wikipedia.org/wiki/Acrylonitrile_butadiene_styrene>

[48] World Issues, “Effect of Bauxite Mining on the Environment” 2010. [Online]. Available: <http://www.worldissues360.com/index.php/effect-of-bauxite-mining-on-the-environment-11861/>

[49] Wrap 1200AR | Vuzix [Online]. Available: <http://www.vuzix.com/augmented-reality/products_wrap1200ar.html>

Individual Contributions

Contributions of Stephen Carlson:

Stephen Carlson was the primary hardware contributor to the project, working extensively on the project schematic and PCB. He performed the necessary design constraint analysis, selecting the parts appropriate to the project to meet goals of portability, power consumption, and cost. He also handled the microcontroller development environment and peripheral interfacing drivers, as the foundation of the code running on the embedded system. Stephen also made contributions to the project packaging and did a large fraction of the physical electronic component assembly and soldering onto the custom PCB.

Stephen’s first task was design constraint analysis, as the appropriate parts had to be selected to handle the design requirements of the project specific success criteria. As part of summer planning, research was performed on XBee wireless communication devices and potential choices for the project microcontroller. When the semester began, Stephen shifted to analyzing candidates for the appropriate sensors to be used for measuring the user’s head orientation and geospatial position. Power supply also became a focus at this time, as Stephen worked to ensure that the entire system could be supplied from one lithium-ion battery, while being able to satisfy the criterion of measuring and reporting the battery capacity to the user. Stephen completed the Design Constraint Analysis section of this report, which summarizes the design constraint choices made.

After parts had been selected and ordered, work on connecting them together with a schematic began. Stephen had experience with CadSoft EAGLE from a previous summer internship, so he initially worked on the project schematic. Care was taken to ensure that other group members could easily read the schematic and follow the design process, both to ensure that everyone could understand the project and to allow another member to work on the schematic section of this report. Stephen worked closely with Alec Green during this time to ensure that he could also understand the schematic and communicate the correct insight to other members.

Next, Stephen focused on the PCB design, an important part of the project which left little room for error due to fabrication lead times. With experience from several past PCB designs, Stephen designed the board for manufacturability, reducing the chance of the board house making an unrecoverable error in fabrication. Stephen also paid close attention to impedance matching and noise handling on the PCB to increase the likelihood of success. Stephen made sure to listen to feedback from the course staff and group members during this time. Most of the other considerations taken during this time are described in the PCB Layout Narrative section of this report, which was also written by Stephen. After the PCB was available, Stephen worked on soldering parts, along with incremental testing of the power supply, microcontroller, and peripherals. No modifications to the schematic or circuit board were required for a fully functional design.

Stephen’s last contributions were on the basis of the microcontroller level code. Over the summer, Stephen researched how to set up a development toolchain for the microcontroller used in this project on Windows and Linux. He then helped other team members set up the devices on their own computers to allow anyone to work on the project using the evaluation boards. After the schematic was complete, Stephen worked with evaluation boards for the critical parts in the project, writing and testing the low-level peripheral drivers required for SPI, I2C, and the serial interface to the XBee and GPS.

Contributions of Stephen Ellis:

 Steven’s primary contributions were in the areas of packaging and the rendering code which runs on the Raspberry Pi. Packaging construction was mostly a collaborative effort with Thor which occurred over the course of numerous meetings. The initial prototype was constructed with a baseball cap with project box, display, and reflector glass all attached with masking tape. A few trips were made to hardware and automotive stores with Thor to evaluate possible options for both mounting hardware and reflector solutions. For the mounting hardware the following options were all considered: welding helmets, hard hats, threaded rods, aluminum angle, hinges. For the reflector there were both acrylic and polycarbonate sheets available, and both automotive and home window tint films available to provide the reflective surface. After evaluation the options, a hard hat, acrylic sheet, aluminum angle, bolts, hinges, and reflective home window tint were all purchased. Over the course of several weekends, the final packaging was put together by hacking, drilling, and bolting everything into place which was done mostly by Steven and Thor. Steven also did some research into optics to improve focusing and field of view of the display. He determined that a large fresnel lens with a short focal length would give the desired effect of a display which appears several feet away from the user, allowing focus on the display and background simulteanously. This type of lens would also introduce substantial distortion which could be corrected with software. Unfortunately, such lenses are difficult to find. In the end Steven found and ordered a page sized fresnel lens magnifier with a 12" focal length which was far too long to substantiallly improve the focusing problem but did nicely increase the field of view of the display. This lens was mounted at the bottom of the display enclosure and held in place between aluminum rails with some hot glue. Steven also created the CAD model used for 3D printing the rear electronics enclosure on the headset. After an initial attempt at printing using the printer available in the IEEE office (which Thor and Steven spent some time together attempting to configure properly) which resulted in a severely warped case, Steven revised the model for improved fitment and structual integrity and a second print was made using a printer owned by a fellow ECE477 student, Seth. Steven drilled out the holes for and assembled the CCU enclosure. Steven also dismanted the LCDs which were used and clipped the regulators on them and shorted the needed pads on two of them.

Steven’s other major contribution was writing most of the software stack running on the Raspberry Pi.  The major components of this are the SPI code for communicating with the microcontroller and the 3D rendering code. The SPI code required close collaboration with Thor, who wrote the microcontroller side of the SPI communication. The Raspberry Pi SPI code made use of a library called WiringPi which provides straightforward access to the SPI hardware’s initialization routines and buffered read/write operations. A protocol was devised collaboratively with Thor. Because the Pi’s SPI hardware can only act as a master, the protocol needed to have all transactions initiated by the Pi rather than the microcontroller.  The basic architecture is the Pi sending a byte indicating the type of packet it wants to receive, then waiting for a response from the micro consisting of a leading byte indicating packet size followed by the data packet. The 3D rendering code was based on OpenGL ES 2.0 in order to make use of the Pi’s powerful GPU hardware. A few initial functionality demos of integrating sensor data into rendering were accomplished by modifying example programs from Broadcom included in the default Pi OS image. A fair amount of effort was expended in an attempt to write a custom 3D engine entirely from the ground up using OpenGL directly. This proved to be a larger task than anticipated and a library was sought out to ease development. It turns out that the vast majority of open source 3D rendering libraries will not work on the Raspberry Pi without modification because they either target desktop OpenGL rather than OpenGL ES, or don’t have an easy way of integrating the slightly esoteric initialization routines required to make use of the Broadcom GPU hardware on the Raspberry Pi. The library that ended up using is called piNGL, which is a Pi-specific port of a library developed by a university CS department to help their students learn OpenGL. The final application is based on a sample application included with the library. It adds in SPI code which runs in a separate thread and requests data from the microcontroller periodically. The rendering thread starts out by loading a simulation level file in a custom format designed by Steven which specifies the ID, location, visibility, and model file for each object in the scene. Then during rendering the camera is moved into an appropriate location and orientation based on the GPS and IMU data received from the microcontroller. Then a loop checks the visibility of each object in the scene and draws it if it is set visible.

Contributions of Alec Green:

Alec’s primary contributions were developing a Google Maps parsing script to automatically generate maps for our headset simulations, and developing IMU filtering code to determine user head orientation from IMU sensor (accelerometer, gyroscope, magnetometer) readings. Alec’s other contributions include developing the CCU/Headset packet protocol with Thor.

During the summer, Alec researched various mapping schemes (e.g. Mercator) to understand how the ‘tiles’ of a database like Google Maps or Bing Maps were organized. After understanding this scheme, the next step was to retrieve information from these databases detailing the geospatial coordinates of potential ‘obstacles’ in the game (e.g. buildings, landmarks). However, Alec realized that this sort of detailed information is intentionally not released by the databases and protecting from parsing via sending images and not a parsable language to the client’s map applet. Therefore, Alec implemented image processing functionality in the script to parse the images and determine from the pixel coordinates the corresponding geospatial coordinates of the obstacles. This final script allows one to select an arbitrary geospatial region and be returned coordinates representing all the obstacles in that region.

Alec was also responsible for developing the IMU filtering code on this project. Before implementing a filter, much time was spent researching which filter was most appropriate, and testing those filters on the headset. The Kalman filter is a well-known and popular filter for such applications. However, we noticed that its use of the gyroscope introduced some undesirable drift in the output measurements. The Complementary filter is another popular filter, and its mathematical basis is quite intuitive; however, it requires a large history of measurement data and continuous operation on that data, which is not feasible in our real-time application. Finally, we considered a simple low-pass filter, with filtering characteristics easily identifiable through Matlab. Surprisingly, although this filter does not utilize the high temporal resolution of the gyroscope, it performed acceptably for our application and is quite computationally efficient. It was also easy to adjust this filter to satisfy the latency requirements for our headset due to the simple relationship between non-zero filter coefficients and perceived delay of the output.

Finally, Alec teamed with Thor to develop a packet protocol for communication between the CCU and headset. This involved considering all possible information that might need to be transferred between the two physically separated devices and determining how to transfer this data real-time with a minimal number of bytes sent wirelessly. This included tradeoff consideration of pre-loading data to the headsets before simulation start for indexed reference during the simulation versus sending the explicit data in-simulation (in most cases it made sense to pre-load the data). Additionally, we had to consider to how to monitor the health of network (e.g. wireless connectivity, battery life of headsets) through ‘heartbeat’ packets.

Contributions of Thor Smith:

My contributions to the project vary through a variety of sections. I served as team leader, helped design and build packaging, contributed to software on the headset, and built the software and user interface for the central control unit. Throughout the course of the semester I ordered most of the components used in the design of the project.

In the beginning of the project, one of the primary focuses was to achieve maximum distance for wireless connectivity. I performed several tests with XBee devices to determine their maximum range, which was determined to be about 250 meters. I later contacted DigiKey to find out why the range was significantly less than expected and they informed me that it had to do with height off of the ground.

As the development progressed, a major concern for the practicality of our project was a good choice of packaging. I worked with Steven Ellis and I created paper drawings for the packaging. I contributed to the construction of the first prototype for the headset using a baseball cap. The baseball cap was constructed by cutting out the bill of the hat and taping a black project box on top. A piece of glass was hung from the edge of the project box. I researched alternate helmets that could be used in our design and thought of using a bike helmet and a baseball helmet. The baseball helmet was considered because it had a bill. The initial prototype drove us to travel to hardware stores to find alternative materials that could be used to create a headset that would be considerably sturdier. Steven Ellis and I selected a hard hat, aluminum rods, screws, acrylic, and other parts from Menards. I cut aluminum rods for the headset, drilled holes in the aluminum rods, drilled holes in the hard hat, and filed metal and plastic pieces to remove rough edges. I purchased the hinges from Menards and hung the Acrylic from the headset using masking tape to hold it at an angle. The first piece of acrylic I hung was a piece of acrylic that was foggy and reflected a little bit but didn’t provide much visibility for the background. Later, I selected reflective film from Menards and applied the first coating to a clear piece of acrylic. This considerably improved the view of both the screen and the background. I built a cardboard box for the PCB and raspberry pi to sit in the back of the headset temporarily and worked with Steven and the IEEE Computer Society to get a 3d printed box for the back of the headset. I attached the first printed box to the back of the headset with one screw.

The packaging wound down towards the end of the project and the PCB came in. As these events occurred I pushed harder to complete the software on the headset and CCU. I merged the test code to communicate with the GPS and the IMU onto the headset. I wrote the code to process wireless data on the headset and parse packets. Initially the code didn’t do much. It sent out broadcast wireless packets that were parsed by a test program on the CCU. Later the code was modified to handle receiving all of the wireless packets and to forward these packets through SPI. I designed and implemented the command mode structure for sending data over SPI from the microcontroller to the raspberry pi serving as the GPU. I created a python script to generate C code for the structures representing the packets between the headset and the CCU. The script also generated functions to simplify the process of packing the structures into byte streams to avoid structure padding from the compiler. I created a program on the CCU to monitor the wireless UART port and buffer important information sent from headsets. This program used sockets to communicate with a user interface. I designed the user interface and wrote it using Tkinter. I built an API in python for communicating with the C program to send requests for information retrieved from wireless packets and to send wireless packets, including sending a file. I experienced initial problems sending a file through SPI which ended up being a result of not clearing a buffer. When initially sending SPI data I debugged problems receiving IEEE floats sent through XBee. I wrote another program in python to allow me to test calling all of the python API functions to verify that they worked as expected. The python struct module was used to pack python integers and strings into bytes that could be sent through and interpreted by a C program. I designed the Pacman maze for the Pacman simulation and I worked with Steven Ellis to define a suitable configuration file format for sending data.

Packaging

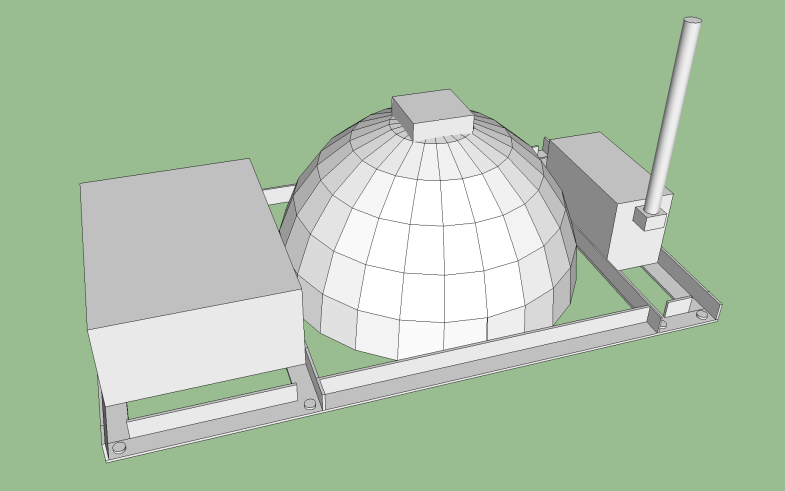


Figure B-1: Headset, ¾ view

Headset Weight Estimate:

|  |  |
| --- | --- |
| Component | Weight |
| Battery | 155g [3] |
| Display | 80g |
| Pi Model A (with SD card) | 35g |
| Hard Hat | 200g |
| XBee + antenna | 40g |
| Custom PCB | 20g |
| Acrylic Sheet | 30g |
| Front LCD enclosure | 120g |
| Rear PCB/battery enclosure | 85g |
| Aluminum rail | 200g |
| GPS Antenna | 50g[4] |
| Total | 1015g |

Headset Packaging Cost:

|  |  |
| --- | --- |
| Component | Cost |
| Hard Hat | $10 |
| Acrylic | $10 |
| Front LCD enclosure | $6.50 |
| Rear PCB/battery enclosure | $10 |
| Reflective film | $20 |
| Aluminum rail | $5 |
| Total | $61.50 |

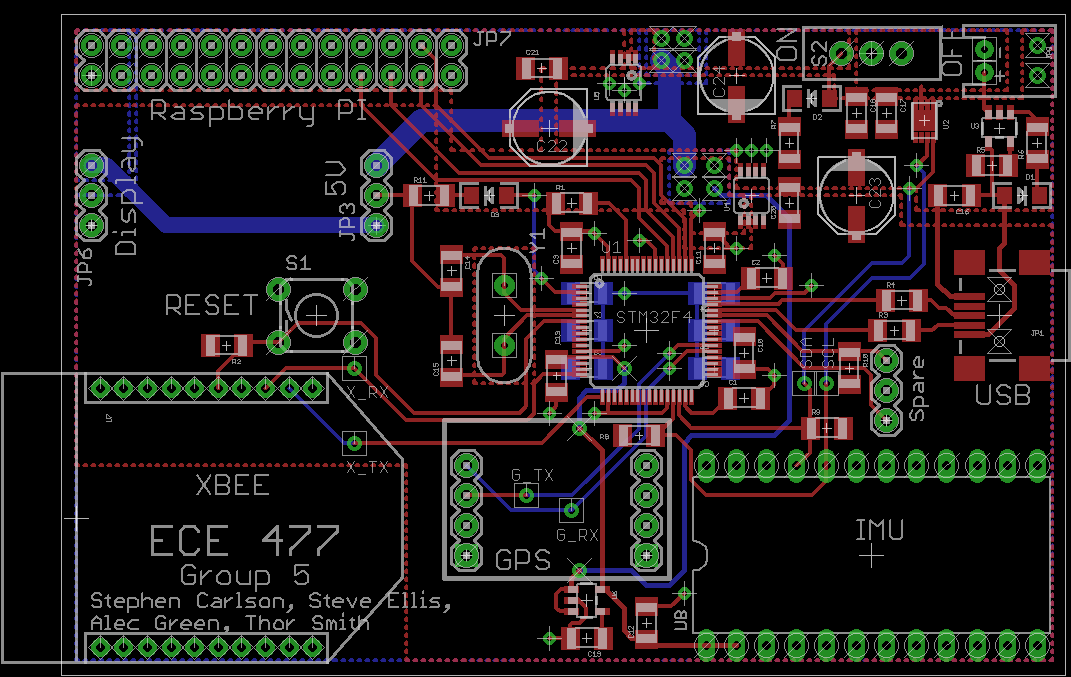
CCU Weight Estimate:

|  |  |
| --- | --- |
| Component | Weight |
| Display | 80g |
| Pi Model A (with SD card) | 35g |
| XBee + antenna | 40g |
| Enclosure | 120g |
| Keypad | 20g |
| Total | 295g |

CCU Packaging Cost:

|  |  |
| --- | --- |
| Component | Cost |
| Enclosure | $6.50 |
| Keypad | $5.00 |
| Total | $11.50 |

Figure B-2: Project Packaging Specifications



Top left: Raspberry Pi GPIO header

Top right: Power Supply

Center: micro

Bottom center: GPS

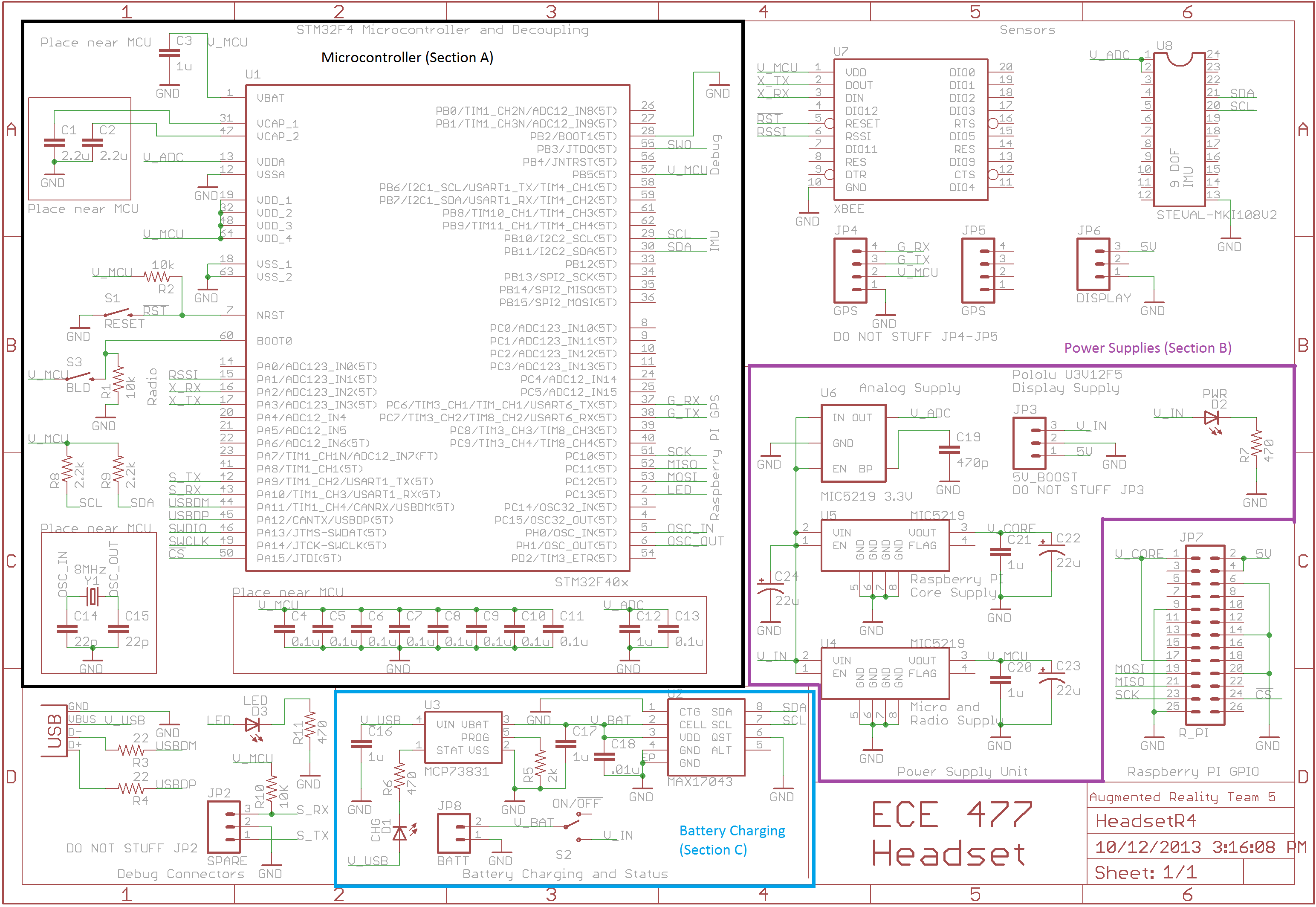
Bottom left: XBee

Bottom right: 9DOF

Figure B-3: PCB Footprint Layout

Schematic

Figure C-1: PCB Complete Schematic



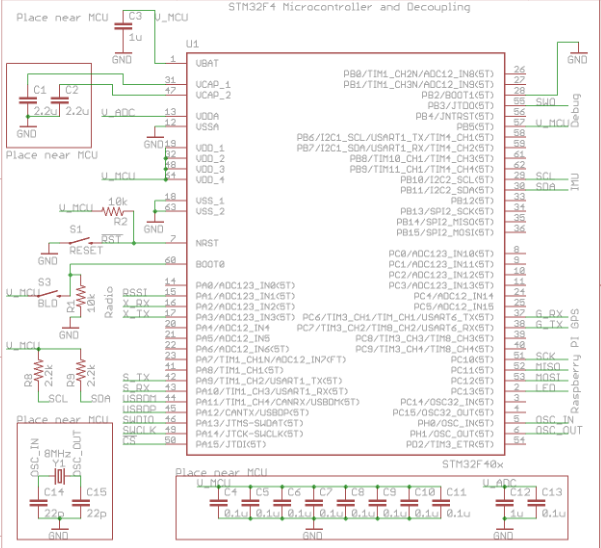


Figure C-2: Microcontroller (STM32 F4)

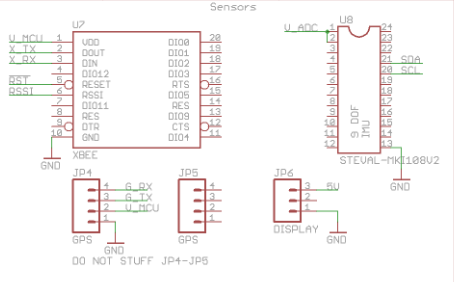


Figure C-3: IMU, XBee, GPS headers, Display headers

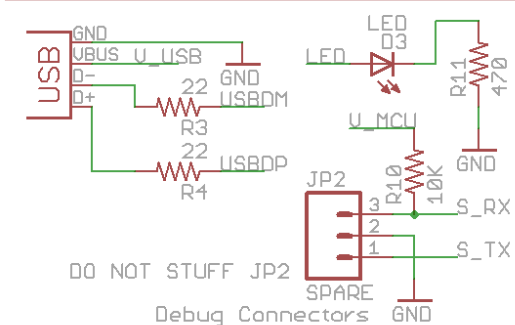


Figure C-4: Headers for USB and Microcontroller Debugger

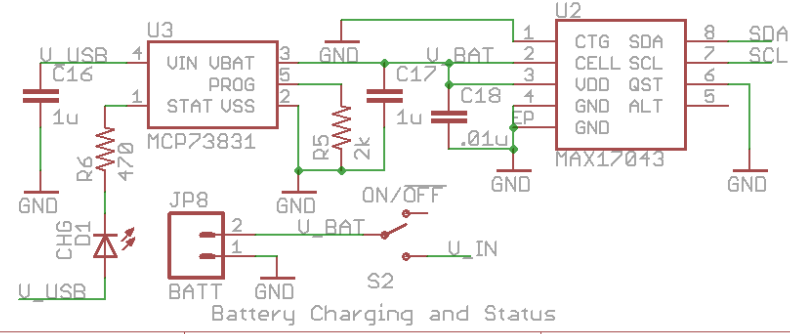


Figure C-5: Li-Ion Battery Charger + Battery “Fuel Gauge”

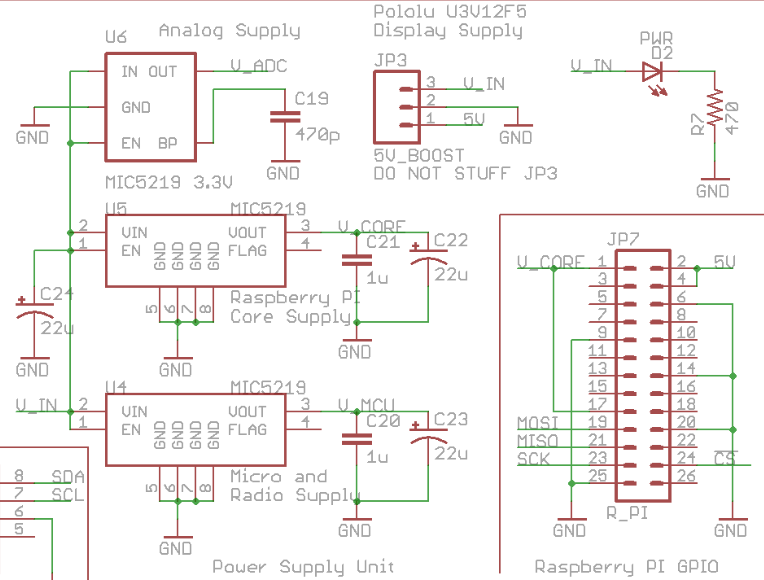


Figure C-6: Power Supplies + Raspberry Pi Header

PCB Layout Top and Bottom Copper

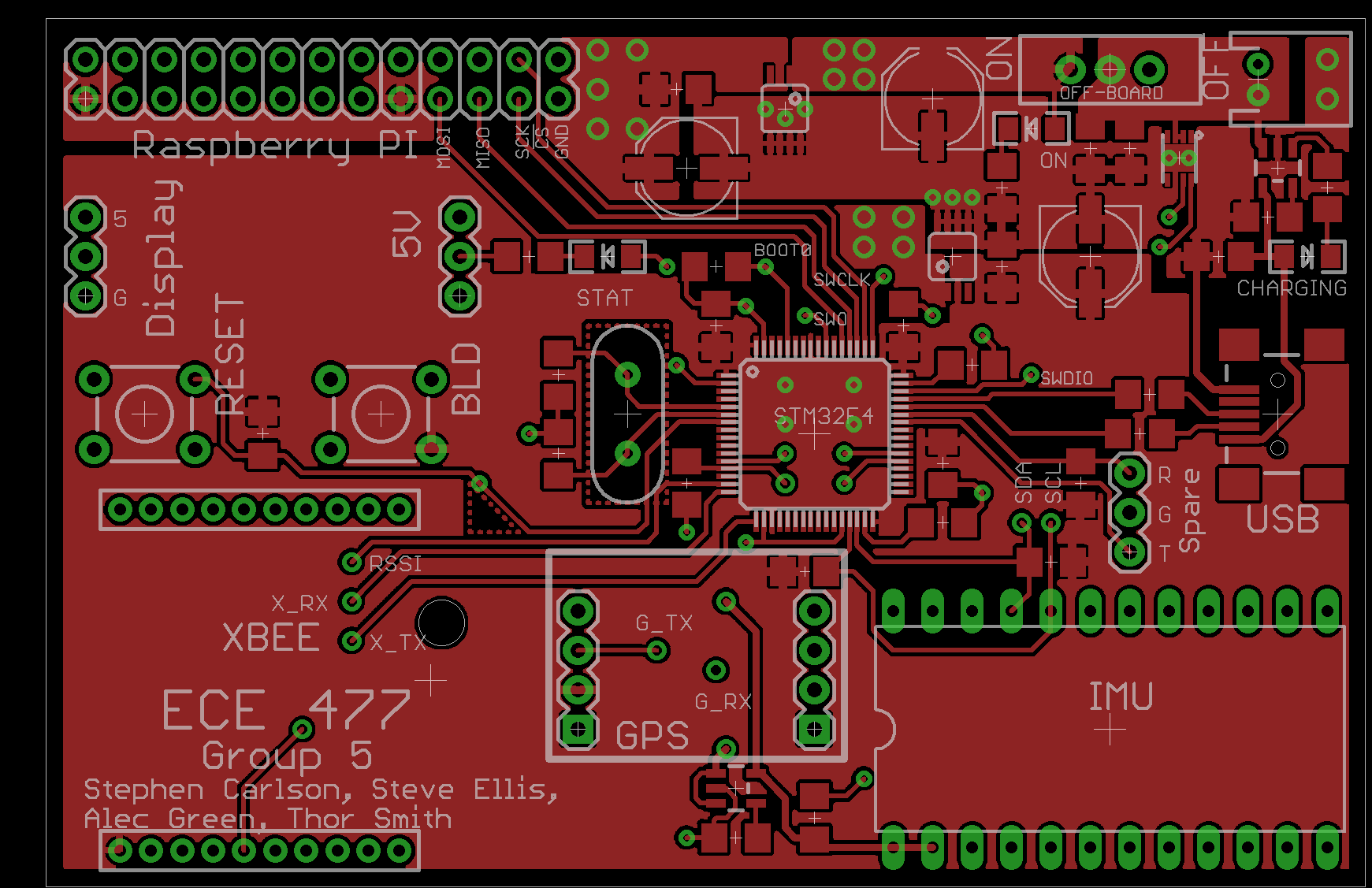


Figure D-1: PCB layout, top copper and silkscreen

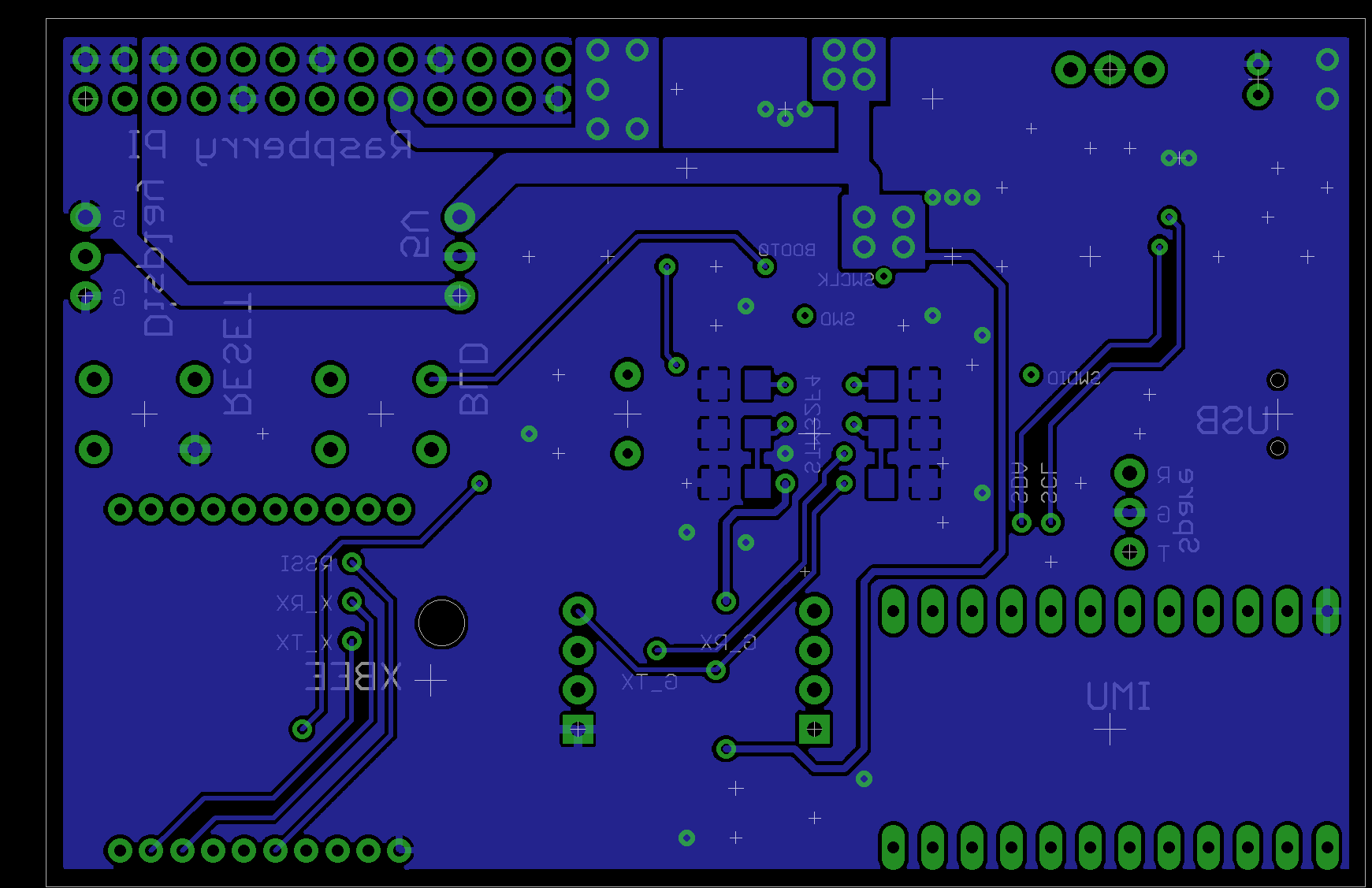


Figure D-2: PCB layout, bottom copper and silkscreen

Parts List Spreadsheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Vendor* | *Manufacturer* | *Part No.* | *Description* | *Unit Cost* | Qty | *Total Cost* |
| Digi-Key | SGS Thomson | STM32F405RGT6 | Microcontroller | 11.45 | 1 | $11.45 |
| Digi-Key | Digi Corporation | XBee Pro 900HP S3B | Wireless Communication Device | 39.00 | 2 | $78.00 |
| Digi-Key | Miscellaneous | N/A | Passive components | 10.00 | 1 | $10.00 |
| Digi-Key | SGS Thomson | STEVAL-MKI108V2 | Inertial Measurement Unit 9-DOF | 27.60 | 1 | $27.60 |
| SparkFun | OnShine | ANT-555 | GPS Antenna RP-SMA | 12.95 | 1 | $12.95 |
| SparkFun | SkyTraq Technology | Venus 638FLPx | Global Positioning System Receiver | 49.95 | 1 | $49.95 |
| Adafruit | Unknown | N/A | Composite Input Display 4.3” | 49.95 | 1 | $49.95 |
| Newark | Raspberry Pi Foundation | Model A | Central Control Unit Motherboard and Headset GPU Motherboard | 25.00 | 2 | $50.00 |
| Newark | Miscellaneous | N/A | Wall Supply/SD cards for Raspberry Pi | 12.00 | 1 | $12.00 |
| Newark | L-Com | HG905RD-RSP | Wireless Antenna | 19.28 | 2 | $38.56 |
| Micrel | Micrel | MIC5216 | Regulator LDO 500mA MSOP-8 | 0.00 | 2 | $0.00 |
| Micrel | Micrel | MIC5219 | Regulator Low Noise LDO SOT-23-5 | 0.00 | 1 | $0.00 |
| Maxim IC | Maxim IC | MAX17043 | Voltage Based Battery Fuel Gauge | 0.00 | 2 | $0.00 |
| Microchip | Microchip | MCP73831 | Linear Charge Management Controller | 0.00 | 1 | $0.00 |
| TOTAL | | **$340.46** |

NOTE: Prices are for one (1) central control unit and one (1) headset.

FMECA Worksheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Power Supplies** | | | | | | |
| **Failure No.** | **Failure Mode** | **Possible Causes** | **Failure Effects** | **Method of Detection** | **Criticality** | **Remarks** |
| P1 | Vin shorts to ground and battery protection circuit fails. | Failure of JP3, U6, U5, U4, Battery protection circuit | Excessive heat generation from power supply. Fire and or battery explosion. | Observation | High | Low probability because of redundant circuitry |
| P2 | V\_ADC > 3.3V | Failure of U6, VIN shorted to V\_ADC | IMU may not report values and may become permanently damaged. | Cannot move “around” virtual objects. | Low |  |
| P3 | V\_CORE > 3.3V | Failure of U5, VIN shorted to V\_CORE | Raspberry Pi may reset repeatedly, work fine, or be permanently damaged. | No display shown. | Low |  |
| P4 | V\_MCU > 3.3V | Failure of U6, VIN shorted to V\_MCU | Micro and Xbee may become permanently damaged. | No display or static display. Objects to not update their location. | Low |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Battery Charging** | | | | | | |
| **Failure No.** | **Failure Mode** | **Possible Causes** | **Failure Effects** | **Method of Detection** | **Criticality** | **Remarks** |
| B1 | V\_USB shorts to V\_BAT and battery protection circuit fails | Failure of U3 | Battery could get overcharged and cause heating or explosion. | Observation | High | Unlikely because two components need to fail. |
| B2 | U2 fails open | Failure of U2 | Reported battery charge incorrect and does not vary with time. | Observation of battery charge indicator. | Low | User unaware of how much battery life is left. Inconvenience |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Microcontroller** | | | | | | |
| **Failure No.** | **Failure Mode** | **Possible Causes** | **Failure Effects** | **Method of Detection** | **Criticality** | **Remarks** |
| M1 | Micro fails to run | Failure of C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, R2, Y1, U1 | Display does not change or no display. | Observation | Low | Device just “doesn’t work” |
| M2 | Micro runs but does not function properly. | Failure of C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, R2, Y1, U1, Software | Unpredictable Behavior: Intermittent display problems or errors in simulation. | Observation | Med | User could be disoriented by moving pictures. |