

2014

AEAS

Active Electronic Assault System

Project Documentation

Group 16
Karena Stout, Ryan Sivek, Alex Balogh
University of Central Florida
4/27/2014



Abstract

Active Electronic Assault System (AEAS) is the senior design project for Group 16 overseen by the College of Engineering and Computer Science at the University of Central Florida. The project aims to provide the proof of concept for an accurate firefight simulation tool that could potentially lead to the replacement of currently used out-of-date laser based systems. AEAS will use various sensors and communications subsystems to determine location and orientation of weapons in the field. This data was communicated to a main server that calculates bullet trajectories and sends accurate hit notifications. System component selection and designs are presented, as well as testing and operation procedures outlined. Project milestones and budget are discussed, and expected uses of the results of this effort are considered.

TABLE OF CONTENTS

1. EXECUTIVE SUMMARY	5
2. BACKGROUND.....	7
2.1. VISION	8
2.2. SPECIFICATIONS	9
3. SYSTEM CONCEPT	10
4. POSITIONING SYSTEM.....	13
4.1. DESCRIPTION	13
4.2. DESIGN APPROACH	14
4.2.1. Trilateration	14
4.2.2. Ultrasonic Beacons	16
4.2.3. Ultrasonic Receivers	19
4.3. PROTOTYPING	20
4.3.1. PCB Fabrication	20
4.4. TESTING	21
4.4.1. Accuracy.....	21
4.4.2. Range.....	23
4.5. RTK GPS ALTERNATIVE SOLUTION	24
5. COMMUNICATIONS	26
5.1. DESCRIPTION	26
5.2. TYPE OF WIRELESS COMMUNICATION	26
5.3. TYPE OF ZIGBEE DEVICE.....	28
5.4. DESIGN APPROACH	30
5.4.1. Configuration and Use	31
5.4.2. Hardware Implementation.....	35
5.5. PROTOTYPING	38
5.6. TESTING	39
6. WEAPON ATTACHMENT	40
6.1. DESCRIPTION	40
6.2. DESIGN APPROACH	41
6.3. ORGANIZATION	41
6.3.1. Microcontroller	42
6.3.2. Orientation Subsystem.....	43
6.3.3. Fire Signal Detection	47
6.3.4. Data Processing	48
6.3.5. Power	49
6.3.6. Circuit Design	50
6.4. PROTOTYPING	51
6.4.1. Printed Circuit Board.....	52
6.4.2. Development Kits.....	53
6.5. TESTING	54
6.5.1. Orientation Accuracy	55
6.5.2. Trigger Sensitivity	57
7. SERVER.....	59
7.1. DESCRIPTION	59
7.2. SERVER HARDWARE	60
7.3. SERVER OPERATING SYSTEM.....	66
7.4. AEAS SOFTWARE.....	68
7.4.1. Data Structures.....	68
7.4.2. Algorithms	70
7.4.3. Visualization.....	75
7.5. TESTING	79
7.5.1. Weapon Orientation Test Data Injection/Procedure	79

7.5.2.	<i>User Position Testing Procedure</i>	79
7.5.3.	<i>Trajectory Determination Test Procedure</i>	80
7.5.4.	<i>Performance Analysis</i>	80
8.	INTEGRATED TESTING	81
8.1.	PERFORMANCE ANALYSIS.....	81
8.2.	SYSTEM ACCURACY.....	81
9.	USING THE SYSTEM	82
9.1.	BEACON SETUP.....	82
9.2.	WEAPON MODULE ATTACHMENT.....	83
9.3.	SERVER OPERATION.....	83
9.4.	SYSTEM CALIBRATION.....	85
10.	EXPANDABILITY	87
10.1.	OPERATIONAL LIMITATIONS.....	87
10.2.	USER IDENTIFICATION.....	87
10.2.1.	<i>Description</i>	88
10.2.2.	<i>Types of User Identification</i>	88
10.2.3.	<i>Type of RFID system</i>	89
10.2.4.	<i>Design</i>	89
10.2.5.	<i>Testing</i>	92
10.3.	BODY ATTACHMENT.....	93
10.3.1.	<i>Description</i>	93
10.3.2.	<i>Design Approach</i>	93
10.3.3.	<i>Body Attachment Microcontroller</i>	94
10.3.4.	<i>Vibration Motor Control</i>	97
10.3.5.	<i>Vibration Motor Circuit</i>	99
10.3.6.	<i>User Identification Circuit</i>	100
10.3.7.	<i>Data Processing</i>	100
10.3.8.	<i>I/O Interface</i>	101
10.4.	PROTOTYPING.....	102
10.4.1.	<i>PCB Fabrication</i>	103
10.5.	TESTING.....	103
10.5.1.	<i>Vibration Motor Response</i>	103
10.5.2.	<i>Communications Performance</i>	104
11.	BUDGET	105
12.	MILESTONES	106
13.	CONCLUSION	107

Table of Figures

FIGURE 1: MILES SIMULATION.....	8
FIGURE 2: AEAS SYSTEM CONCEPT.....	11
FIGURE 3: AEAS BLOCK DIAGRAM.....	12
FIGURE 4: TRILATERATION.....	15
FIGURE 5: BEACON TIMING DIAGRAM.....	16
FIGURE 6: AREAS OF EFFECT FOR THE ORIGINAL BEACON LAYOUT.....	17
FIGURE 7: AREAS OF EFFECT FOR THE IMPLEMENTED BEACON LAYOUT.....	18
FIGURE 9: ULTRASONIC RECEIVER WITH METALLIC CONE ATTACHMENT.....	19
FIGURE 10: STATIONARY POSITIONING TEST EXAMPLE.....	22
FIGURE 11: RADIO FREQUENCY SPECTRUM.....	27
FIGURE 12: UART DATA FLOW FOR THE XBEE-PRO.....	31
FIGURE 13: NETWORK TOPOLOGIES AVAILABLE TO THE XBEE-PRO.....	34
FIGURE 14: XBEE-PRO ANTENNA TYPES.....	36
FIGURE 15: BASE XBEE-PRO ADAPTER BOARD.....	37
FIGURE 16: FIELD XBEE-PRO ADAPTER BOARD.....	38
FIGURE 17: CONCEPT OF WEAPON ATTACHMENT DEVICE WITH AN M16A2.....	40

FIGURE 18: WEAPON ORIENTATION MODULE BLOCK DIAGRAM	41
FIGURE 19: WEAPON ATTACHMENT ORGANIZATION	42
FIGURE 20: INVENSENSE MPU-6050.....	43
FIGURE 21: MPU-6050 PIN ARRANGEMENT	44
FIGURE 22: ORIENTATION MEASUREMENTS.....	46
FIGURE 23: FLEXIFORCE A201 SENSOR. REPRINTED WITH PERMISSION FROM TEKSCAN INC.	48
FIGURE 24: TRIGGER WITH FLEXIFORCE SENSOR ATTACHMENT	48
FIGURE 25: WEAPON ORIENTATION MODULE FINAL CIRCUIT	50
FIGURE 27: MPU-6050 BREAKOUT BOARD	52
FIGURE 28: PCB LAYOUT DIAGRAM.....	52
FIGURE 29: ATMEL DATA VISUALIZER - ACCELEROMETER DATA	53
FIGURE 30: ATMEL DATA VISUALIZER - GYROSCOPE DATA.....	54
FIGURE 31: ANGLE INCLINATION TESTS	55
FIGURE 32: INITIAL AEAS GUI CONCEPT	60
FIGURE 33: OVERHEAD VIEW OF RASPBERRY PI MODEL B SERVER HARDWARE	65
FIGURE 34: SAMPLE UML CLASS DIAGRAM.....	69
FIGURE 35: MAX DISTANCE OF SCENARIO AREA	71
FIGURE 36: BALLISTICS BEHAVIOR MODEL	72
FIGURE 37: TARGET VIEW EXAMPLE	76
FIGURE 38: POSITIONING VIEW EXAMPLE.....	78
FIGURE 39: BEACON PLACEMENT	82
FIGURE 40: INITIAL AEAS WEB GUI (CONNECTION NOT YET ESTABLISHED)	85
FIGURE 41: THE IC ANTENNA AND READER EFFECTIVELY FORM A TRANSFORMER	92
FIGURE 42: TOP VIEW OF BODY ATTACHMENT MICROCONTROLLER SHOWING PIN LOCATIONS.....	95
FIGURE 43: VIBRATION MOTOR HIT REGIONS	98
FIGURE 44: VIBRATION MOTOR CONTROL CIRCUIT.....	99
FIGURE 45: BODY ATTACHMENT CIRCUIT FOR USER IDENTIFICATION	100
FIGURE 46: BODY ATTACHMENT CIRCUIT CONTAINING COMMUNICATIONS DEVICE	101
FIGURE 47: BODY ATTACHMENT CIRCUIT	102

Table of Tables

TABLE 1: SPECIFICATIONS OF THE AEAS SYSTEM	9
TABLE 2: STATIONARY POSITIONING TEST DATA	22
TABLE 3: IN MOTION POSITIONING TEST DATA	23
TABLE 4: MA40S4S DIRECTIVITY IN S.P.L (ESTIMATES).....	24
TABLE 5: MA40S4R DIRECTIVITY IN SENSITIVITY (ESTIMATES)	24
TABLE 6: XBEE-PRO SPECIFICATIONS	30
TABLE 7: MPU-6050 PIN DESCRIPTIONS	44
TABLE 8: POWER REQUIREMENTS OF THE WEAPON ORIENTATION MODULE.....	50
TABLE 9: ORIENTATION ACCURACY TESTS.....	56
TABLE 10: EVALUATION OF TRIGGER PULLS	58
TABLE 11: TYPICAL MEMORY REQUIREMENTS FOR OPERATING SYSTEMS	62
TABLE 12: HARDWARE SPECIFICATION FOR EMBEDDED COMPUTER.....	62
TABLE 13: HARDWARE SPECIFICATION FOR PC	63
TABLE 14: APPROXIMATE TIME REQUIRED FOR ARITHMETIC INSTRUCTIONS.....	64
TABLE 15: HARDWARE SPECIFICATION FOR RASPBERRY PI MODEL B	65
TABLE 16: DEMONSTRATION PC SPECIFICATION	66
TABLE 17: EXAMPLE OPERATING SYSTEMS.....	67
TABLE 18: CONFIGURATION PARAMETERS	86
TABLE 19: ATMEGA328TQFP PINOUT.....	96
TABLE 20: INPUT SELECTION SIGNALS FOR VIBRATION MOTOR CONTROL	98
TABLE 21: COST TO DEVELOP THE SYSTEM	105

Executive Summary

In the Fall 2013 - Spring 2014 Senior Design program as administered by the College of Engineering and Computer Science at the University of Central Florida, Group 16 designed and built AEAS (Active Electronic Assault System), a state-of-the-art, virtual bullet, firefight simulator. The dominant motivation for this effort was the ability to provide more realistic and effective training to the United States Armed Forces. Although laser based training systems have seen extensive capability enhancements over the past decade, they remain expensive, burdensome, and somewhat unrealistic. AEAS shows that a more cost-effective and accurate system is not only desirable, but also achievable with current advances in micro-electromechanical systems (MEMS) and positioning technologies.

AEAS consists of three main parts: the weapon attachment device, main server, and positioning beacons. Each of these units will intercommunicate and provide functionality for virtual feedback stimulated by the actions of real users within the bounds of the playing field. The weapon attachment gathers position and orientation data, which is communicated wirelessly to the server along with 'fire' signals. The server determines 'shot' trajectories based on this information and sends 'hit' signals to the GUI display in the path of the virtual bullet.

The most challenging information to obtain is the accurate position of each weapon. Typical current Global Positioning System (GPS) devices are accurate to within a few meters, which is immensely insufficient for providing accurate 'shot' trajectories to human targets that are typically 1.6 by 0.46 meters¹. An innovative low-cost ultrasonic positioning system has been shown to provide location measurements accurate to within 3cm². This solution is used to produce location data for the AEAS system and provides excellent results within the effective area.

Data communication between each of the system components was also a major consideration for the project effort. A method of communication that is affordable, mid to long-range, and relatively simple to interface with was sought after to provide this capability. After consideration of several possible solutions, using XBee-Pro RF modules by Digi was determined to be a sufficient and practical solution. The units have an effective range of approximately one mile, and are designed for low latency and predictable timing³. The relatively simple interface made the XBee a simple addition to the weapon module, and a USB Adapter Board allows straightforward connectivity to the main server.

Weapon attachment devices were designed and built to measure weapon orientation,

¹ Griggs, J. M. "Human Figure Average Measurements." <http://www.fas.harvard.edu/~loebinfo/loebinfo/Proportions/humanfigure.html>. 2001. Web. 30 Nov. 2013. Dimensions given in English system converted to metric system.

² Bjercknes, J. D., Liu, W., Winfield, A. F., and Melhuish, C. (2007). Low Cost Ultrasonic Positioning System for Mobile Robots. In Wilson, M.S., Labrosse, F., Nehmzow, U., Melhuish, C., and Witkowski, M., editors, *Proceeding of Towards Autonomous Robotic Systems*, pages 107 - 114, Aberystwyth, UK.

³ Digi International, Inc., "XBee®/XBee-PRO® RF Modules Product Manual v1.xEx - 802.15.4 Protocol" 2009.

determine its position in the playing field, respond to trigger pulls, and communicate status to the main server. To manage these tasks, the Atmel ATmega328P PDIP microcontroller was determined to have the necessary capabilities without excessive processing power. The microcontroller interfaces with an Invensense MPU-6050 3-Axis Gyro/Accelerometer Motion Processing Unit, which uses accelerometer and gyroscope devices to determine direction. Trigger pulls are measured by a FlexiForce pressure sensor and 'fire' signals occur when the pressure exceeds a predefined threshold of 4.5lbs. The orientation data, as well as position data from the ultrasonic receiver, and 'fire' signals are communicated to the server using an XBee RF module.

The projection notifies users if they are aiming at a defined target area, or have fired a shot. The projection provides a view of the current intersection of the weapon trajectory with the target (if a shot were to be fired) as well as a secondary view showing the position of the weapon and the direction which it is facing relative to the positioning beacons. For ease of use it was determined that a simple web browser-based GUI would be the most effective. Near pixel accuracy can be achieved for the targeting view assuming that precise measurements have been made, and the system has been appropriately calibrated for the display and deployment area.

At the heart of the AEAS project is the main server, for which it was determined that a Raspberry Pi Model B was an inexpensive and effective choice. The server communicates to the attachment device using an XBee-Pro with USB Adapter Board and controls each of the ultrasonic beacons for unit positioning. Running on Debian, the AEAS software was a custom Java suite programmed specifically for that purpose. The software will perform virtual bullet trajectory calculations and provide a web accessible interactive Graphical User Interface (GUI) and projection with which overseers can monitor firefight progression.

Each of the modules was tested individually and extensively prior to full system testing and final prototyping. The project budget was approximately \$700 and was divided evenly amount the group members. Major milestones have been outlined for the project effort in which extensive prototyping and testing of AEAS was performed.

Incorporating each of the aforementioned modules, AEAS provides the proof of concept of an effective, accurate firefight simulation. The resulting prototype of these efforts may in fact be of use in the development of full military training programs of record. The system may also find uses in commercial gaming and other simulation applications.

1. Background

Many current US Army trainers assert the low fidelity and negative training issues associated with current laser combat training systems. High cost, inaccuracy at considerable distance (due to beam divergence), impenetrability of foliage or other soft obstructions, and low fidelity 'hit' notification mechanisms are some of the more prevalent issues with current laser based systems.

The Multiple Integrated Laser Engagement System (MILES) is the most widely used laser based training system in the world. The system uses laser emitters attached to rifle barrels and laser receptors on soldiers' helmets and harnesses to simulate combat as shown in Figure 1. The US Communication Code Standard for this technology as maintained by PEO-STRI defines that laser pulses encode weapon type, ammunition type, player identification, and weapon/ammunition lethality effects which are then decoded by the receiving sensor.

The MILES system (including the more recent MILES 2000 program) has provided soldiers with a fairly effective means of training for the past 30 years. However, the laser technology is extraordinarily expensive and out of date. Many individuals involved in US Army research and development are confident that modern technology could potentially achieve much more promising results for a tremendously reduced cost. Considering the US government's recent cuts to defense spending, the Army, as well as other defense programs, would greatly benefit from a more cost effective solution.

Beam divergence (especially in direct sunlight) contributes a significant margin of laser signal reception error. A 'shot' from a single rifle at a certain distance can potentially 'hit' multiple targets simultaneously if they are close enough together. The lasers also do not penetrate thick foliage the way an actual bullet would, which means a soldier can essentially be 'invincible' by hiding in a bush or other soft obstacle.

Some advanced MILES systems use a random number roll and casualty probability lookup table to determine 'hit' outcomes, but many systems still use a shuffled deck of 'casualty cards' which are drawn when a 'hit' notification is set off. This mechanism is extremely inaccurate and also contributes to the overall lack of realism in the training simulation.

Due to these negative training issues, many have commented that alternative training mechanisms can and should be explored. A more accurate 'fake bullet' solution is needed to create more effective training scenarios. A more accurate 'hit' penalty notification mechanism also needs to be employed to achieve satisfactory realism in training. With current advances in technology, we were able to build a system that is not only a sufficiently accurate, but also a more cost effective system is an extraordinarily worthwhile endeavor.



Figure 1: Miles Simulation⁴

1.1. Vision

This project was an effort to explore the possibility of providing a low cost, easy to use, high fidelity system that will allow soldiers to train more effectively and in more realistic conditions. Using various combinations of sensors and position tracking mechanisms, we are confident in our ability to develop an effective simulation that could potentially rival the laser systems currently provided. This vision is based on the idea that given accurate enough information about weapon position and orientation in the playing field, ‘shot’ trajectories can be effectively estimated using

⁴ This image is a work of a U.S. Army soldier or employee, taken or made as part of that person's official duties. As a work of the U.S. federal government, the image is in the public domain.

simple Newtonian principles and projectile motion calculations. Then with tracked unit positions, users that lie in the path of the projected bullet motion track are effectively 'hit'. Exact body location and damage of the 'hit' can also be provided if accurate body positioning is tracked.

Using such a system, the realism of firefight simulation training becomes easily enhanced with improvements to technology and the introduction of more efficient and advanced software algorithms. Due to the time and budget constraints of our project, it is unlikely that we was able to provide a complete solution. Our final prototype, however, was ready for expansion to achieve such a solution if the results are satisfactory.

Unfortunately, due to a lack of funding we were forced to simplify our original vision in scope and design. The positioning system and the weapon attachment were reduced. The body attachment was cut completely and instead, a projection of target area and movement tracking were implemented. The identification system used to track statistics of soldiers was also removed from the design.

1.2. Specifications

The desired specifications of the AEAS system are outlined in [Table 1](#).

Minimum virtual bullet accuracy (deviation/distance)	0.2%
Minimum weapon orientation precision deg accuracy = $\tan^{-1}(\text{bullet accuracy}/\text{distance})$.	0.1°
Minimum battery life	2 hours
Minimum data sampling rate	1kHz
Minimum bandwidth	64Kbps
Maximum weight - weapon attachment	5 lbs
Maximum weapon attachment volume	5 in ³
Minimum number of supported players	1
Minimum playing field dimensions	5m x 5m x 5m
Maximum latency of 'shot' signal damage response	0.5s
Maximum latency of GUI display	1s

Table 1: Specifications of the AEAS System

2. System Concept

The implementation of AEAS will consist of three main parts:

1. **Weapon attachment device** - An electronic device that is easily attached to any particular weapon with a trigger, specifically military weapons. This allows soldiers to use their own personal weapons in combat training. The device measures weapon orientation which is communicated to the main server and provides a means of pinpointing the weapon's location and orientation in 3-dimensional space. The weapon attachment runs on battery power. A 'shot' signal is sent to the main server when the user presses the trigger.
2. **Main server** - A distinct hardware and software system that pieces the combat scenario together. When a 'shot' signal is received from a weapon attachment device, data concerning the weapon orientation and position is captured and used to mathematically determine the 'shot' trajectory based on predefined weapon specifications. If the target area is determined to be within the bullet trajectory, impact locations are drawn to the area. This server provides this capability as a front end web accessible GUI that will allow trainers to view the scenario as it progresses in real time.
3. **Positioning Beacons** - The mechanism by which the attachment can pinpoint its location in three dimensional space. Three ultrasonic transmitters are set up at specific points as beacons, and the main server stores and communicates the coordinates of these beacons. This is a reference based positioning system which could be augmented slightly with dead-reckoning via inertial measurement data to provide more accurate measurements.

Each of these devices needs to communicate with each other to transmit various scenario data. Due to the interference that using wires would cause, a wireless system is the most effective solution. The communications and positioning subsystems were designed to be unobtrusive and using various current models allows simple integration.

Each of the aspects of the AEAS system is captured in [Figure 2](#) below. This shows the projection of a virtual target onto a wall in front of the user. The positioning beacons surround the area and the AEAS Server is positioned nearby to control the beacons as well as communicate with the Weapon Attachment. An additional computer connected to the network can also access the server and view all current scenario data in real time.

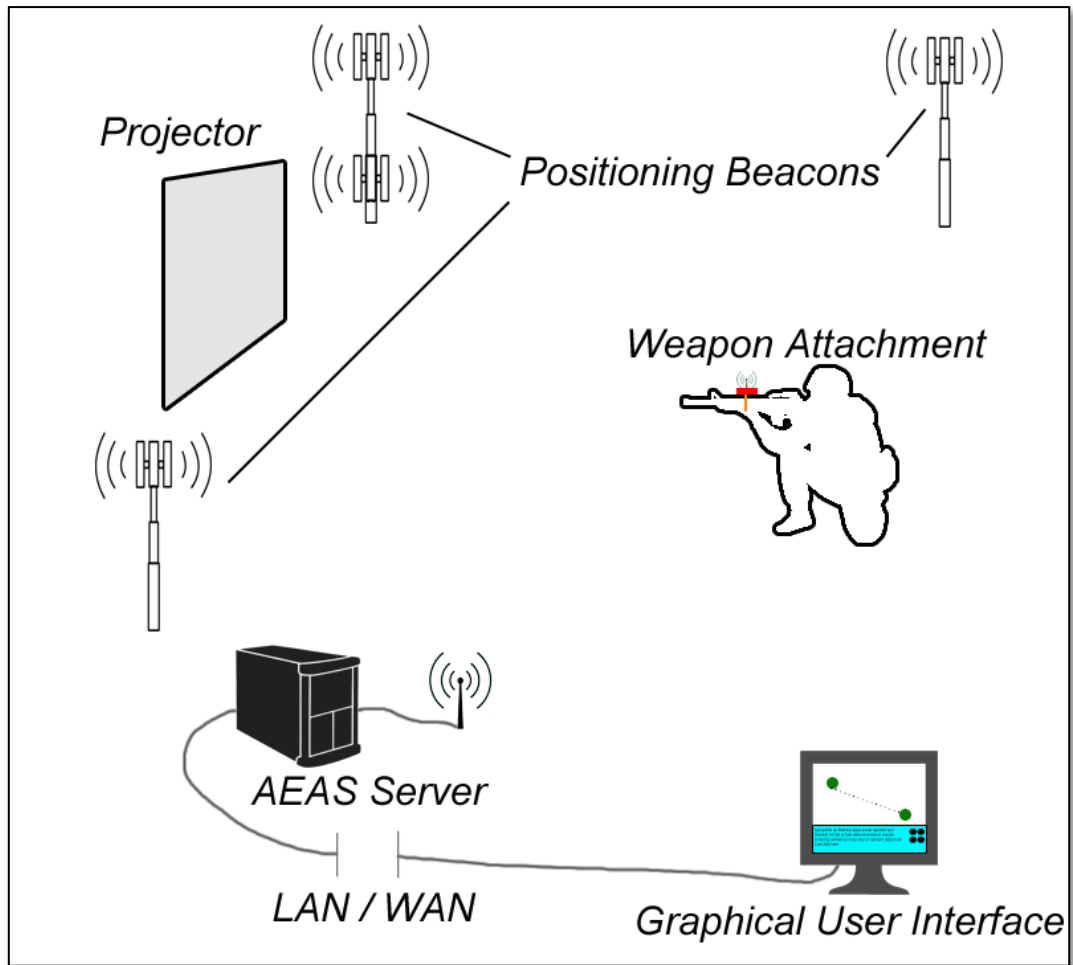


Figure 2: AEAS System Concept

The components of the overall system are captured in the block diagram in Figure 3. and will be explained in detail in subsequent sections. The three main parts are the AEAS Server, the weapon attachment, and the positioning system. The interconnections between each main module are shown with their various sub-modules. Note that a small additional external module is attached to the AEAS server signifying the connection capability to other computers through a web accessible GUI.

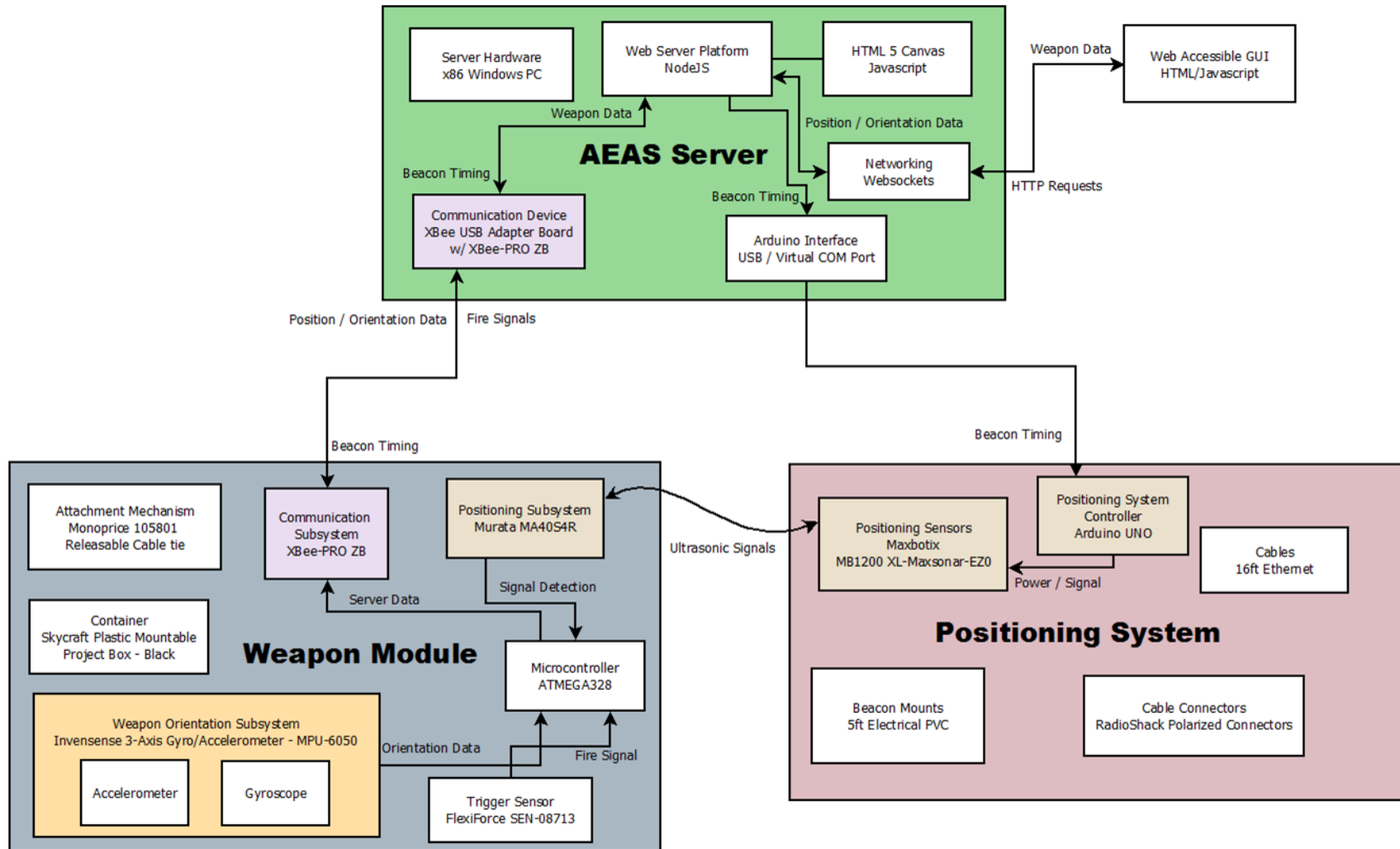


Figure 3: AEAS Block Diagram

3. Positioning System

The positioning system is an essential part to the AEAS system to determine where weapons are in the playing field at any given moment of time, and especially when a virtual bullet is fired. The system needs to be accurate enough to seem realistic compared to current Army training systems.

3.1. Description

In order to obtain accurate bullet trajectory and impact detections, a positioning system with substantial accuracy is required. Many various position measurement solutions were considered. With our desire to achieve the largest outdoor playing field possible, GPS was an obvious initial consideration. However, typical GPS receivers provide a worst case accuracy of 7.8 meters at a 95% confidence level⁵, which is unacceptable for our system. High-quality, centimeter accuracy GPS receivers are still generally prohibitively expensive.

We did locate a relatively low cost RTK GPS receiver project named Piksi that shows promising results and even includes open source software⁶. However, the \$900 price tag was still an insurmountable barrier for our current project. Interested parties should be able to integrate this or similar technology with our prototype without much difficulty. Such an upgrade would be recommended for use in military training.

Therefore, in an effort to keep project costs to a minimum, and still achieve the accuracy and precision required, an ultrasonic active beacon navigation system was determined to be the most effective solution. Active beacon systems allow high sampling rates and reliability with minimal processing. Ultrasonic transmitters and receivers are currently quite inexpensive and typically used for range finding applications in robotics. Sound waves emitted at frequencies upwards of 20 kHz (above the average human hearing range of 15 kHz) are emitted and received, with time delay of arrival used to calculate distances.

Jan Dyre Bjercknes and colleagues from Bristol Robotics Laboratories developed a low cost ultrasonic positioning system for mobile robots and recorded an absolute error of within 3 centimeters⁷. This ultrasonic solution exceeds the needs of our system and cost requirements. Therefore, we settled on adapting a version of their ultrasonic positioning system for the AEAS project.

⁵ National Coordination Office for Space-Based Positioning, Navigation, and Timing. "GPS Accuracy." <http://www.gps.gov/systems/gps/performance/accuracy/>. 18 Sept. 2013. Web. 27 Oct. 2013.

⁶ Swift Navigation Inc. "Piksi: The RTK GPS Receiver." <http://www.kickstarter.com/projects/swiftnav/piksi-the-rtk-gps-receiver>. 2013. Web. 27 Oct. 2013.

⁷ Bjercknes, J. D., Liu, W., Winfield, A. F., and Melhuish, C. (2007). Low Cost Ultrasonic Positioning System for Mobile Robots. In Wilson, M.S., Labrosse, F., Nehmzow, U., Melhuish, C., and Witkowski, M., editors, *Proceeding of Towards Autonomous Robotic Systems*, pages 107 - 114, Aberystwyth, UK.

The system causes sound waves to be emitted from prepositioned beacons at certain time intervals. These sound waves are then picked up by a receiver which calculates the time of arrival of the signal, which can then be used to calculate the distance between the receiver and the beacon. Using four or more beacons, one can use trilateration to pinpoint the receiver's coordinates in three dimensional space. This ultrasonic solution exceeds the needs of our system and cost requirements. Therefore, we have adapted a version of their ultrasonic positioning system design for the AEAS project.

The greatest challenge with using an ultrasonic beacon system was the limited operational range of transmission. Ultrasonic transmitters and receivers have widely varying ranges, and most are used as rangefinders which makes the listed maximum range largely inapplicable to our use. Based on the research done by Bjerknes, a distance of 916 cm is achievable⁸, and we were even able to increase this operational range with additional modifications discussed later.

We decided to use Murata MA40S4S 40 kHz ultrasonic transmitters for the beacons and Murata MA40S4R 40 kHz ultrasonic receivers for the positioning subsystems in the attachment. These units are very cheap, small, and have adequate capabilities. In our subsystem testing we saw promising sensitivity and transmitting characteristics at a 180 degree radius about the direct face which is essential to maintaining positioning coverage throughout the playing field.

3.2. Design Approach

For the design of the ultrasonic positioning system we first present the mathematical implementation of trilateration on which the positioning system is dependent. The positioning beacons and receiver designs will then be discussed thereafter.

3.2.1. Trilateration

Trilateration is determining a position by knowing your distance from at least three known points as shown in Figure 4. In our system, those known points are the ultrasonic beacons. Four ultrasonic beacons are placed along the coordinate axes and provide the required anchor points from which to measure as shown in Figure 4. Aligning these beacons to the axes greatly reduces the amount of computation that the ATmega328 must perform to compute its location. This simplification allows the microcontroller to perform the trilateration algorithm in about 6 - 9 milliseconds with four beacon points.

⁸ Ibid. Page 4.

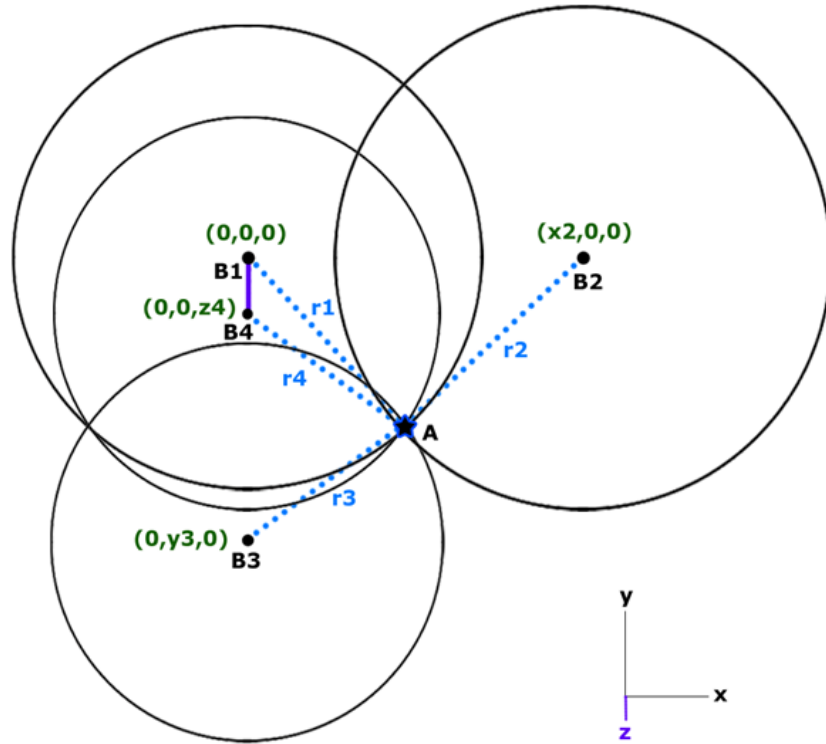


Figure 4: Trilateration

When a signal is transmitted and received from B1, B2, B3, or B4 we then know the distance between the beacons and the unit expressed in Figure 4 as r_1 , r_2 , r_3 , and r_4 respectively. This distance can be expressed as a circle around each beacon. The point at which each of the three circles cross is our position. The area the circles represent are actually spherical areas. Note that the intersection between two spheres results in a circle of possible points. Four points are necessary in order to get resolution of full three dimensional positioning.

We start with the equations for the three circles, assuming B1 is at the origin:

$$\begin{aligned} r_1^2 &= x^2 + y^2 + z^2 \\ r_2^2 &= (x - x_2)^2 + y^2 + z^2 \\ r_3^2 &= x^2 + (y - y_3)^2 + z^2 \\ r_4^2 &= x^2 + y^2 + (z - z_4)^2 \end{aligned}$$

Simplifying these equations we get:

$$\begin{aligned} x &= \frac{r_1^2 - r_2^2 + x_2^2}{2x_2} \\ y &= \frac{r_1^2 - r_3^2 + y_3^2}{2y_3} \\ z &= \frac{r_1^2 - r_4^2 + z_4^2}{2z_4} \end{aligned}$$

3.2.2. Ultrasonic Beacons

In our original design specifications, the ultrasonic beacons are arranged in a pattern, with one beacon acting as the origin for the coordinate frame. The original design is presented here as well as our modifications for use in the final product.

Since the sensitivity of the sensors drops off significantly at frequencies higher or lower than 40 kHz, we determined it was best to have all units operate at the same frequency. Due to this, however, there is no way to distinguish one beacon's signal from another's if they transmit simultaneously. Therefore, we devised a timing scheme which allows each beacon its own firing window as shown below in Figure 5.

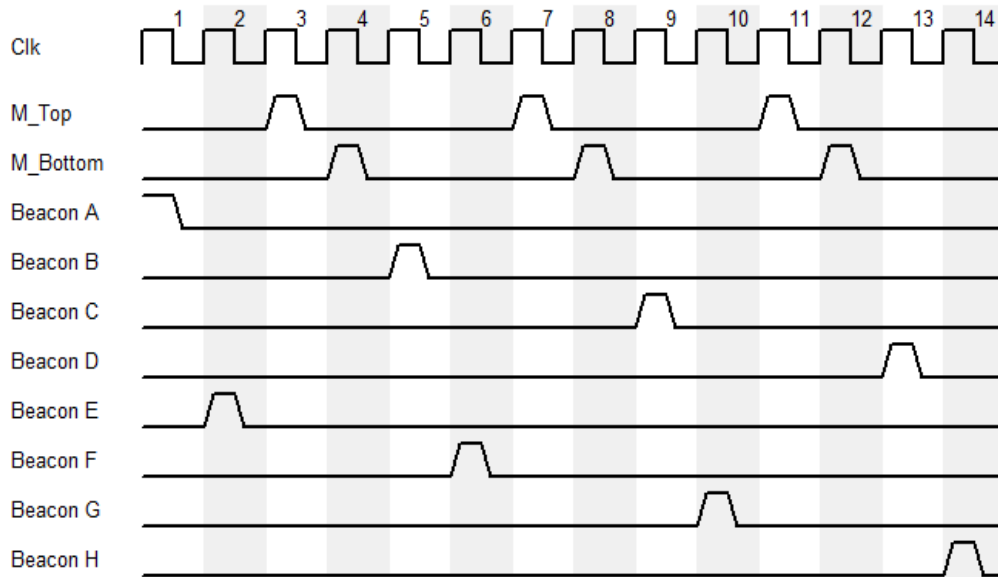


Figure 5: Beacon Timing Diagram

The duration of wait time between beacon transmissions was computed with the following equation:

$$\text{Signal wait time} = \frac{\text{arena maximum distance}}{\text{speed of sound}} + \text{reflection wait time}$$

The reflection wait time is the time allotted to wait for sound reflections in the arena to die down. Note that the speed of sound is dependent on temperature, so the signal wait time also becomes dependent on temperature.

This figure includes estimated areas covered by each beacon using directivity in S.P.L information from the datasheet for the ultrasonic transmitter. Actual transmitter orientation may be changed to allow for more effective coverage in the final prototype.

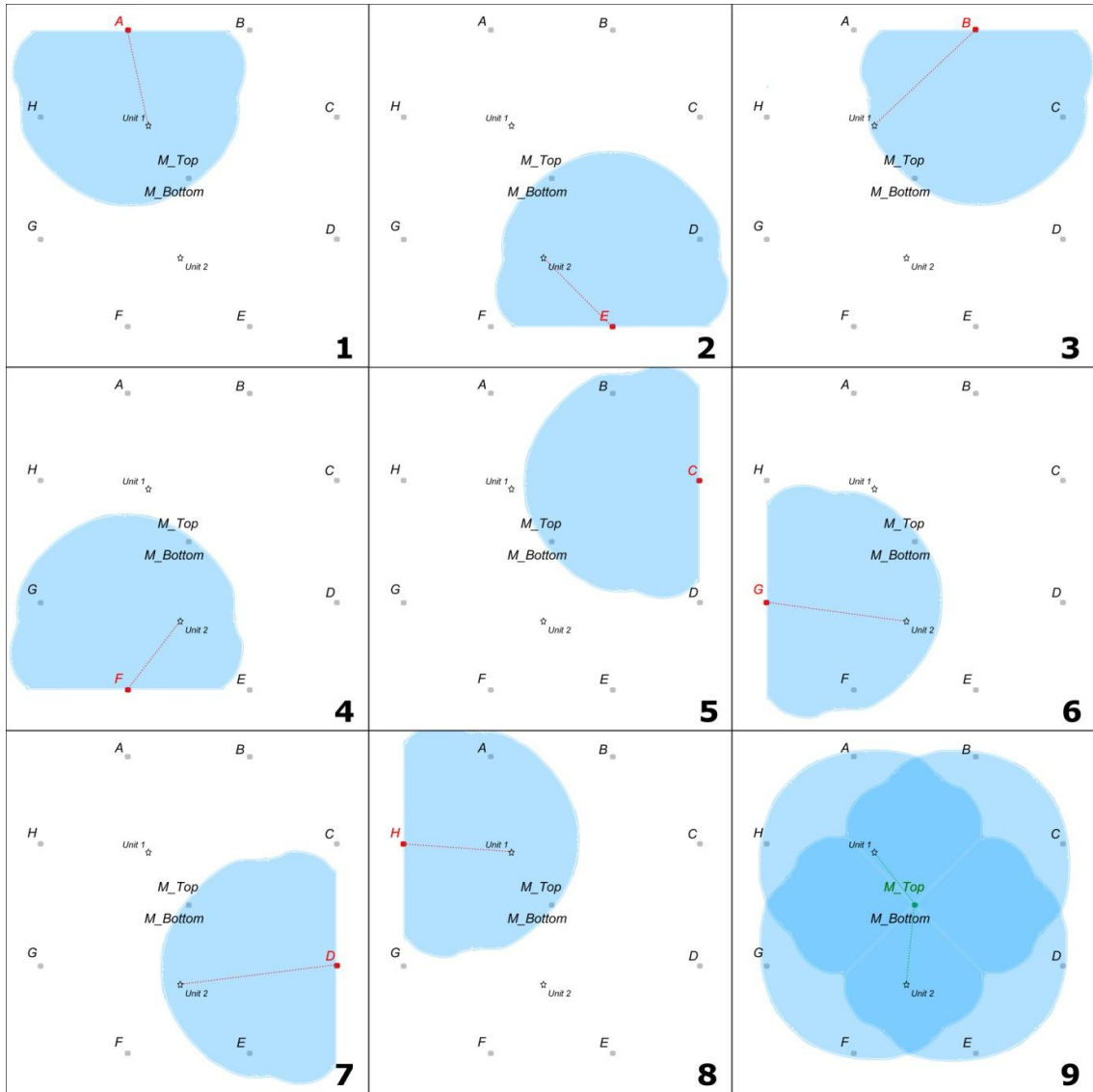


Figure 6: Areas of Effect for the Original Beacon Layout

For the sake of saving space, item 9 in the image is only shown once. This signal pattern is identical for both M_Top and M_Bottom, which refer to the signals transmitted from the center beacon.

Our actual implementation of the beacons arranged them in alignment with the four coordinate axes as presented in the trilateration section of the report. This greatly reduced the cost of the system by requiring less than half of the ultrasonic transmitters of the original system. The effect of this new layout is shown in Figure 7.

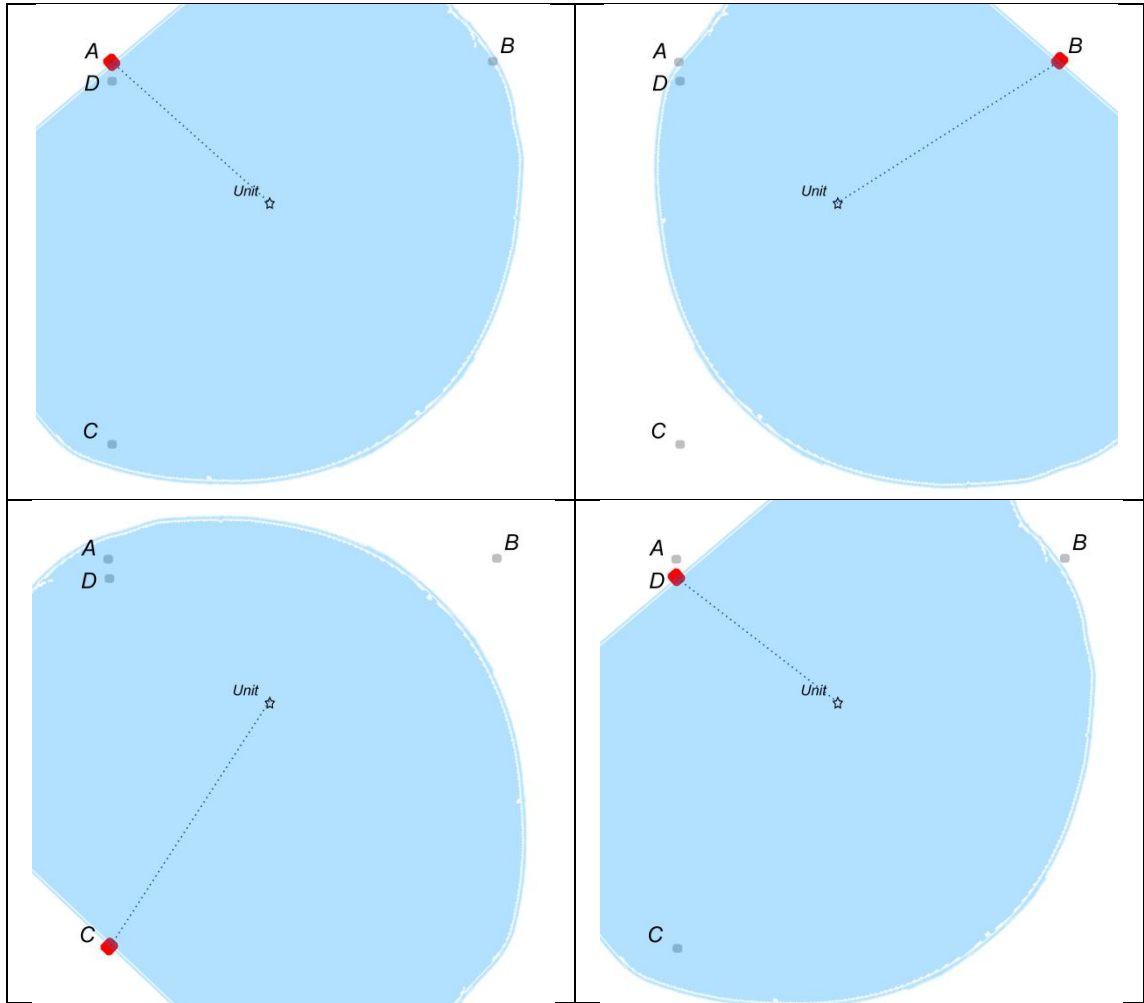


Figure 7: Areas of Effect for the Implemented Beacon Layout

Instead of designing and implementing the ultrasonic transmitters ourselves, we decided to procure four Maxbotix MB1200XL Maxsonar EZO rangefinding devices. These provide full rangefinding capabilities, although we only use them as transmitters of ultrasonic signals. All they need for this to work is a connection to 5v power, ground, and a single pin which activates the device. This pin is internally kept high, so we pull it low with our controller and set it high whenever a signal is to be sent with that beacon.

The beacons are all controlled by an Arduino UNO which is connected directly to the server via a USB port. The Arduino is flashed with a simple program that waits for input from the server. Once this input is received, each of the four beacons are fired in succession with 30ms intervals. Therefore, the firing of each beacon completes in a total of approx. 150ms with the first 30ms lag time.

3.2.3. Ultrasonic Receivers

Schematic diagrams for ultrasonic receivers are also quite prevalent, and very similar. All consist of some type of amplifier circuit, which is needed to pull up the small voltage generated by the receiver as shown in Figure 8. A comparator op-amp IC is then used to send a sufficient signal onward through the relay.

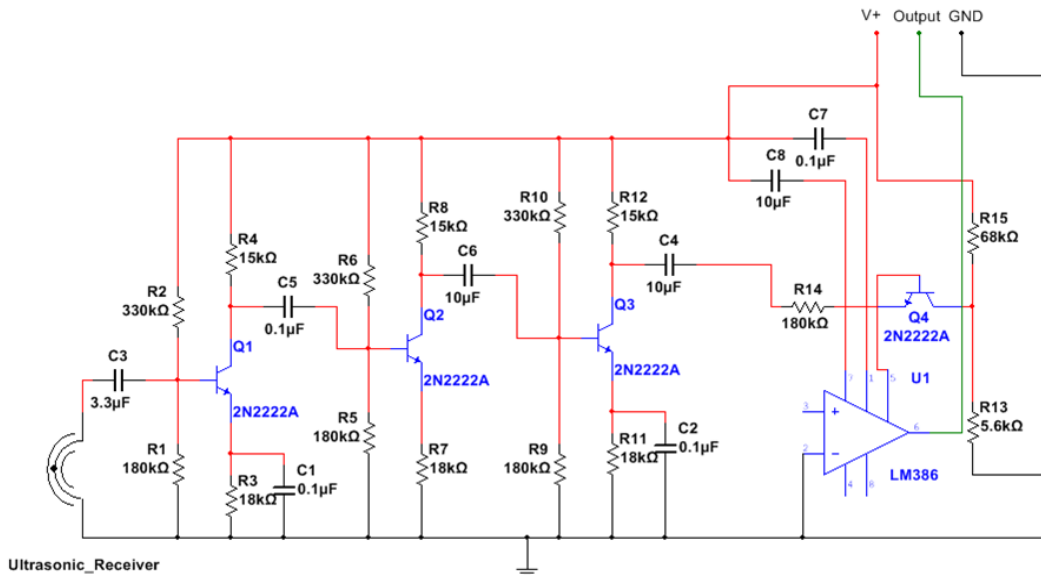


Figure 8: Ultrasonic Receiver Circuit Diagram

When a signal is received, this receiver circuit forwards the signal directly to the attachment unit which will record time of arrival. That metric is then used to determine distances based on the speed of sound, and the distances will then contribute to the calculation of the x, y and z position of the attachment as described in section 4.2.1.

In order to allow the receivers to accept signals from all directions, a small metal cone has been attached to the face of the sensor as shown in Figure 9. This cone significantly increases the overall sensitivity of the receivers.



Figure 9: Ultrasonic Receiver with Metallic Cone Attachment

When an ultrasonic signal travels through the playing field, the cone reflects the sound waves into the ultrasonic receiver. To achieve this, the receiver is placed on the upward side of the weapon attachment so as to be sensitive to sound waves from a 360 degree lateral radius. The metal should be hard and reflective enough to reflect sound waves at near perfection so that no wave momentum is lost at impact to the cone.

For this purpose, we decided to use quite affordable SODIAL 9.5mm metal cone screwback spikes intended for use on studded belts, bracelets etc. The spikes are made of iron with nickel plating and come in inexpensive 50 piece packages. The spikes accomplish the task as effectively as any custom made solution.

3.3. Prototyping

Each of the ultrasonic circuits was assembled onto breadboards and tested extensively before any fabrication onto printed circuit boards was done. The circuits were modified several times to achieve greater sound pressure output from the transmitters, and more sensitivity on the receivers. Since we were not able to achieve the necessary level of accuracy, we elected to pursue a solution that did not involve our own fabrication. We determined that the Maxbotix rangefinders provided a more effective solution. We made minor modifications to the positioning control unit to interface with these devices. Power requirements were fairly similar between these units and our own creations.

3.3.1. PCB Fabrication

Engineeringshock Electronics provides DIY ultrasonic circuit boards at minimal cost⁹. We assessed the viability of using these boards in our final design. However, if the boards did not meet the necessary requirements, we used a reliable PCB manufacturer with accompanying PCB software to produce the circuit boards.

As a contingency plan, we also researched various small effective ultrasonic range finders produced by Maxbotix, particularly the XL-Maxsonar line of products. The XL-Maxsonar EZ/AE0 model in particular has a wide beam and maximum recorded range of approximately 20 feet at 5.0V operation¹⁰. We evaluated the effectiveness of this device for our ultrasonic positioning system. However, these calculations were based on the assumption that the sensors are used for range finding applications and thus were taken by measuring time of arrival after bouncing a signal off an object. In our application, we use a separate unit for transmitting and receiving, which will change the response curve over distance and angle considerably. Therefore, \$\$\$\$\$\$

⁹ Engineeringshock Electronics. "Ultrasonic Kits." <http://www.engineeringshock.com/ultrasonic-kits.html>, 2013. Web. 10 Nov. 2013.

¹⁰ MaxBotix Inc. "XL-MaxSonar[®] - EZ[™] Series High Performance Sonar Range Finder". http://maxbotix.com/documents/XL-MaxSonar-EZ_Datasheet.pdf. XL-MaxSonar[®] - EZ/AE[™] Series. PD1184a. 2012. Web. 24 Nov, 2013.

signal sensitivities \$\$\$\$ these devices.

The Maxbotix devices also come at a considerably higher cost. With a lack of funding, we evaluated the effectiveness of using these devices against our own designs. Using these devices saves on PCB fabrication since they are pre-fabricated. We simply had to integrate them into our attachment unit and ultrasonic beacons using the serial RS-232 interface.

3.4. Testing

Testing of the positioning system consisted of two main aspects: Accuracy and Range. Accuracy was extremely important in providing ‘shot’ precision and realism, and the range allowed us a larger field area which increased the fidelity of the simulation. We wanted each of them to be at a maximum for the final AEAS prototype.

3.4.1. Accuracy

To assess the accuracy of the positioning system, we employed a two part analysis. One stationary, and the second with movement included. This approach allowed us to determine how much the positioning system is affected by motion of the receivers. It was found that accuracy of the system diminishes somewhat during movement, and has an inverse correlation with the velocity of the receiver.

Accuracy of the system was improved with the use of accelerometer data to provide dead-reckoning measurements. This augmentation to the positioning system is explored in section 5.3.4.

The accuracy tests were performed with a Processing script which acted as a simulated server. This program controlled the beacons and communicated with the weapon attachment. Incoming messages from the weapon attachment were logged by the program into a format that was easily readable by Excel. This data was then plotted and analyzed.

3.4.1.1. Stationary Assessment

The stationary test was fairly simple. After the beacons are set up, additional points were mapped on the playing field with measurement tools. Receivers were placed precisely on top of these predetermined points, and a total of 400 position detections and transmissions were recorded. Deltas between the transmitted coordinates and measured coordinates were then computed, as well as the overall error using the Euclidean distance formula:

$$\Delta X = x_t - x_a$$

$$\Delta Y = y_t - y_a$$

$$\Delta Z = z_t - z_a$$

$$\Delta Total = \sqrt{(x_t - x_a)^2 + (y_t - y_a)^2 + (z_t - z_a)^2}$$

From this data we were able to determine system accuracy with a 95% confidence level and a margin of error of about 5%. Graphical data from one of our tests is shown in Figure 10. A summary of the standard deviation and absolute error recorded from the stationary assessment is shown in Table 2.

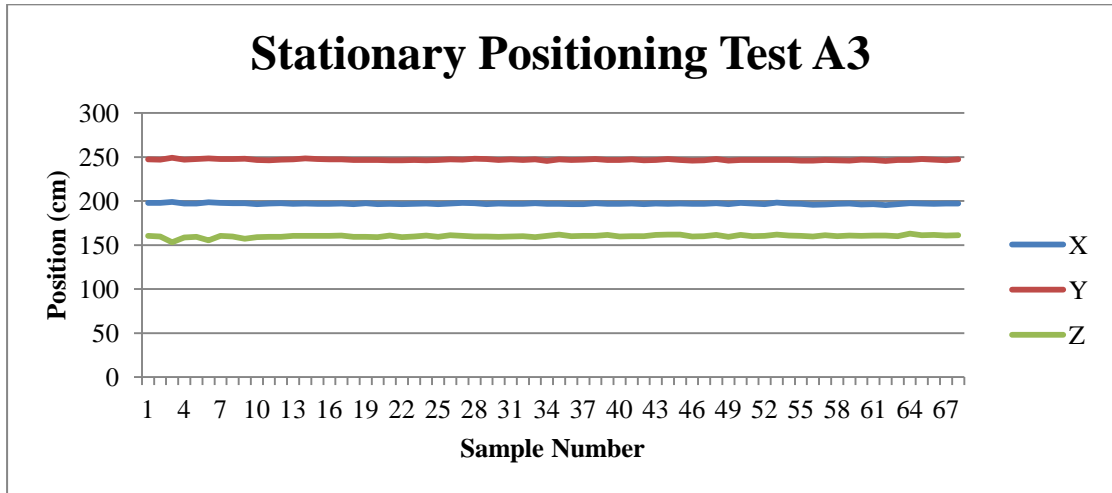


Figure 10: Stationary Positioning Test Example

	X	Y	Z
Standard Deviation	0.748503	0.76409	1.780977
Absolute Error	4.74	5.2	13.53

Table 2: Stationary Positioning Test Data

Note that the standard deviation and error of the Z coordinate axis is considerably higher than X and Y. This is due to the fact that the further the beacons are placed to one another, the more accurate the displacement difference calculation by the microcontroller from the two beacons. Since the fourth beacon on the Z coordinate axis was constrained in distance from the origin, this value is less accurate than the data on the other axes.

3.4.1.2. In Motion Assessment

The in motion test compared transmitted coordinates with expected coordinates calculated using a line defined by two predetermined points used in the stationary assessment. Using an accelerometer to measure the receiver's velocity, we were able

to get an accurate estimate of where the receiver *should* be on the line at any given point in time. A long and physically rigid beam or track was used to ensure the receiver travels as close to the calculated line as possible. Each time a position report was received by the testing software, it used the receivers' current position on the line based on acceleration data to determine the actual position of the unit.

The normalized vector between two measured points, starting at x_0, y_0 and ending at x_1, y_1 was given simply by:

$$u = \frac{(x_0, y_0) - (x_1, y_1)}{\|(x_0, y_0) - (x_1, y_1)\|}$$

We found the distance offset from the starting point by using our acceleration data and elapsed time:

$$d = a * t$$

Then we used the calculated distance and the vector to find the desired point:

$$(x_a, y_a) = (x_0, y_0) + d * u$$

The deltas between these points and the received coordinates were then calculated using the same equations as the stationary assessment. Receivers were propelled at uniform and varying speeds between 1 and 10 meters per second. The data captured from this test is shown in Table 3. Surprisingly, the accuracy of the in motion assessment is not much different from the data captured in the stationary assessment, although it is notably higher.

	X	Y	Z
Standard Deviation	0.943203	0.96153	1.982445
Absolute Error	5.03	5.21	13.81

Table 3: In Motion Positioning Test Data

3.4.2. Range

The operational range of the ultrasonic signals dictates the effective size of the terrain or playing field. Therefore, we needed to have a mechanism for determining the maximum distance for this system so we could arrange our field accordingly. Fortunately, this measurement was fairly straightforward. We simply took the point at which the receiver met the threshold of responsiveness (as outlined in our specifications) at a zero degree angle and that was our maximum range.

Since the Maxbotix XL-Maxsonar EZ/AE0 devices obtained provided at least the minimum range and functionality as defined in our specifications, we used them in the project design due to the substantial relief in fabrication efforts.

The range of ultrasonic transmitters and receivers is directly affected by their attenuation and sound pressure level (SPL), which deteriorates as the angle between receiver face and transmitter face increases. Table 4 shows visual estimates of the directivity in sound pressure level for the MA40S4S ultrasonic transmitter.

Deviation Angle (degrees)	Attenuation (dB)
0	0
15	-1.5
30	-3
45	-6
60	-12
75	-15
90	-20

Table 4: MA40S4S Directivity in S.P.L (Estimates)¹¹

The directivity in sensitivity estimates of the MA40S4R ultrasonic receiver is shown in Table 5.

Deviation Angle (degrees)	Attenuation (dB)
0	0
15	-1.5
30	-4
45	-8
60	-14
75	-20
90	-25

Table 5: MA40S4R Directivity in Sensitivity (Estimates)¹²

Based on these results, we expected experience a significant loss of signal reception at deviation angles larger than 45 degrees. The directivity in sensitivity for the receivers, however, was reversed when the metallic cone was attached. The cone reflects the incoming sounds from all side angles into the device, and is unresponsive to direct sound waves since they are blocked by the large side of the cone. This allows the receiver to accept signals from a 360 degree radius with the receiver pointing upward into the sky. Tests with this configuration were performed extensively to discover the distance at which we have adequate signal reception.

3.5. RTK GPS Alternative Solution

If we had additional funding to pursue the RTK GPS Solution for unit positioning, our

¹¹ Murata Electronics. "MA40S4S: Directivity in S.P.L". http://search.murata.co.jp/Ceramy/CatalogshowpageAction.do?sParam=MA40S&sKno=G010&sTblid=A091&sDirnm=A09X&sFilm=MAS4G04&sType=1&sLang=en&sNHinnm=MA40S4S&sCapt=Directivity_in_S.P.L. 2013. Web. 9 Nov. 2013.

¹² Murata Electronics. "MA40S4R: Directivity in Sensitivity". http://search.murata.co.jp/Ceramy/CatalogshowpageAction.do?sParam=MA40S&sKno=G011&sTblid=A091&sDirnm=A09X&sFilm=MAS4G03&sType=1&sLang=en&sNHinnm=MA40S4R&sCapt=Directivity_in_Sensitivity 2013. Web. 9 Nov. 2013.

maximum field dimensions would have then increased to the maximum range of our communications devices, which would have definitely provided more range than the ultrasonic devices. Since the Piksi receivers are already pre-fabricated devices, we simply needed to interface with them in our attachment module. Testing of the positioning system accuracy will progressed in the same fashion as described in section 3.4.

4. Communications

This section outlines the communication system, which will provide the link between the weapon and body attachments to get the server the information it needs to create the scenario.

4.1. Description

The AEAS system was designed to be used to simulate military combat, which includes a lot of running, twisting, ducking, and jumping. Obviously the AEAS system needed to rely on wireless communication to link the weapon attachment to the server. The server collects the processed orientation, position and trigger pull data from the weapon attachment every millisecond. The server then sends data to the GUI to indicate a 'hit.'

4.2. Type of Wireless Communication

There are many different types of wireless communication available today. The options explored include Bluetooth, Wi-Fi, Infrared, and radio frequency modules.

Radio frequency communication (RF) is the most commonly used source of wireless communications. The radio frequency spectrum consists of signals from extremely low frequency to the terahertz frequencies. The RF devices that are being compared in this section typically operate in the L band (around 2 GHz) and the UHF band (300 MHz to 1000 MHz) as shown in Figure 11. RF communication modules have been in production for a number of years and have developed to the point that they are extremely reliable and use very little power. A number of factors need to be considered when selecting an RF transceiver: power consumption, range, sensitivity, reliability, and transmission rate, security, price and antenna type to name a few. Most of these factors are interrelated and changing one will affect the other. A balance was sought to best meet the needs of the AEAS system.

The infrared communication (IR) is highly dependent on the strength of the LED used for data transmission. The more powerful the LED the stronger and broader the signal, thus increasing the data transmission distance. However, even a very powerful LED will not be sufficient enough to reach 50 feet from source, even with an open line of sight. Another disadvantage with using infrared communications is its susceptibility to interferences from other IR sources, including the sun. For these reasons it was excluded from consideration.

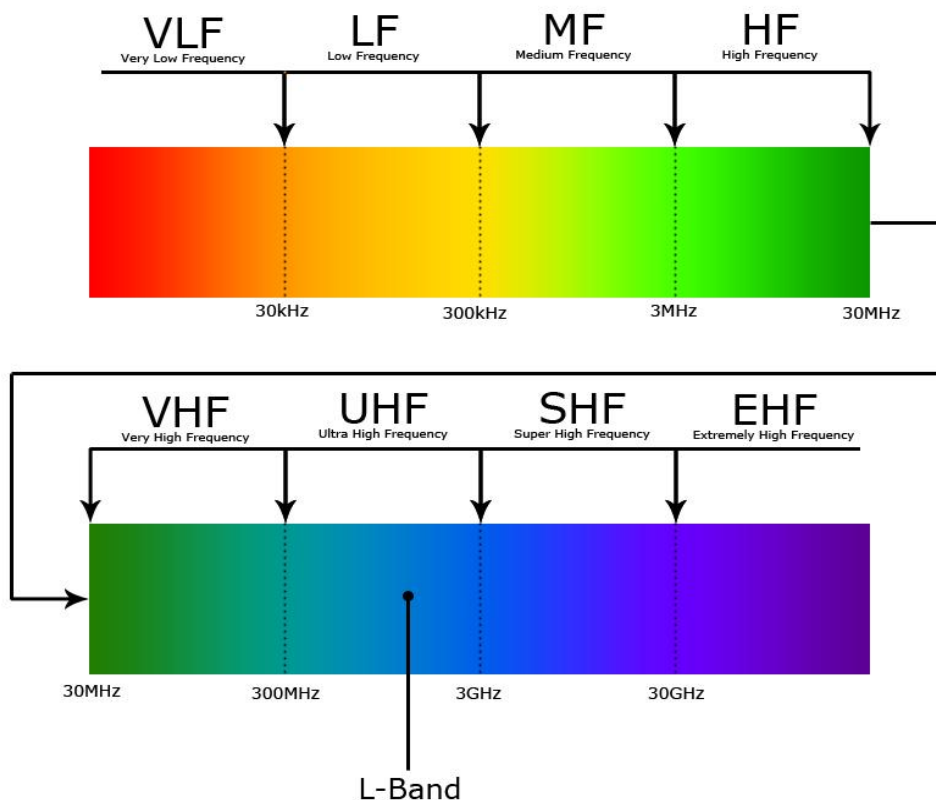


Figure 11: Radio Frequency Spectrum

Bluetooth’s wireless communication was designed to eliminate the use of cords. Bluetooth utilizes the master/slave technology associated with wired communication by connecting all devices to a shared radio channel. All devices on this shared channel share the same clock allowing for the devices to be synchronized. Although, this wireless communication protocol has low power consumption and its maximum range is 300 feet, Bluetooth module prices remain high for now. To keep total costs low Bluetooth technology was excluded from consideration¹³.

Based on our specific needs we narrowed our wireless communications research to RF transceivers. Among the protocols and vendors we decided to look at are: Linx Technologies radio controlled transceivers, Wi-Fi, and Digi international’s XBee-Pro RF modules (ZigBee 802.15.4).

The Linx Technologies long range, 8 button RF transmitter is capable at operating at frequencies of 418 MHz or 433MHz. This specific wireless technology when paired with the RXLC-434-LR series receiver and Holtek HT648L decoder IC is designed to reach a maximum distance of 1000 ft. This 1000 ft. range is based on an open line of sight. Although, this protocol meets our desired range specifications, it is frequency

¹³ “Compare with Other Technologies.” Get Technical. 6 April 2010. Bluetooth 2, October 2010.

sensitive. The Linx protocol only allows for one transmitter to transmit at any given time within the same general air space. Disturbance from another module operating at the same frequency would interfere with the data transmission and the decoder would not recognize a valid transmission, and thus do nothing¹⁴. For this reason it was excluded from consideration.

IEEE 802.11 (b, g, or n), or Wi-Fi, was another RF wireless communication protocol that had to be considered. The Wi-Fi was created to implement a wireless LAN in the 2.4 GHz frequency range. It typically consists of a collection of devices utilizing 802.11 modems that are interconnected via an 802.11 router. This protocol has built-in security and uses spread spectrum signal transmission over numerous frequencies to avoid interference, so in this aspect it meets many of our criteria, but remains expensive to implement. For this reason this protocol was also excluded as a choice.

ZigBee is a data communications protocol that is built on top of the IEEE 802.15.4 standard, designed to address remote monitoring for sensory network applications. This protocol is designed to create ad hoc wireless networks using relatively short range radios which require very small power consumption. In particular ZigBee offers an array of modules that in addition to meeting our specified range; also “access up to 16 different 5 MHz channels within the 2.4 GHz band” and is not expected to interfere with the ultrasonic positioning system. Many of these channels do not overlap those of 802.11 and Wi-Fi technologies, thus greatly limiting interference from other wireless equipment¹⁵. In addition to these advantages ZigBee is capable of connecting serially to the microcontroller on the weapon and body attachments and is easy to include in our design. The only disadvantage to this protocol is its slow data transmission in comparison to other wireless communication platforms. However, for our specific design the speed of transmission is still well within the desired range. For these reasons we elected to use ZigBee for our wireless communication platform.

4.3. Type of Zigbee Device

According to Digi, “XBee modules are embedded solutions providing wireless end-point connectivity to devices. These modules use the IEEE 802.15.4 networking protocol for fast point-to-multipoint or peer-to-peer networking. They are designed for high-throughput applications requiring low latency and predictable communication timing.”¹⁶ Basically XBee is Digi’s own Zigbee based protocol. The XBee Series 1 (also called XBee 802.15.4) is the easiest to work with. They don’t need to be configured, although they can benefit from it. The series 1 is recommended for beginners. For point to point communication these modules work just as well as the Series 2, but without all the extra work. Series 1 and Series 2/2.5/ZB hardware are not compatible.

¹⁴ Linx Technologies Wireless Made Simple, “OEM Long Range Handheld Data Guide Description,” Linx Technologies Inc, 2008. <http://www.linxtechnologies.com/Products/OEM-Products/Long-Range-Handheld-Transmitter/>

¹⁵ C. Diamond, “ZigBee,” 2005, <http://homepage.uab.edu/cdiamond/> Oct 2013.

¹⁶ Digi International Inc, “XBee® 802.15.4,” 2013, <http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module#overview>, Web. Oct 2013.

The XBee Znet 2.5 was formerly the Series 2, but now is retired. Series 2 modules must be configured before they can be used. They can run in a transparent mode or work with API commands, but this all depends on what firmware you configure these with. These also can run in a mesh network, making them highly configurable and flexible modules. It also makes the modules more complicated and difficult to use. These modules are not compatible with the Series 1 modules. These modules are no longer sold but are being replaced with the mostly compatible ZB modules, and were only mentioned for comparison purposes. The ZB is a current Series 2 module. It is basically the Znet2.5 hardware with new firmware, meaning they can also run in a transparent mode or work with API commands and can run in a mesh network. New firmware is readily available and is simple to upgrade. The firmware between the two is not compatible, but is easily interchangeable. These are often call Series 2 modules, and will not work with the Series 1. The 2B is the even more current Series 2 module. These new modules improve on the hardware of the Series 2 modules improving things like power usage. They run the ZB firmware, but because the hardware has been changed they cannot run the Znet2.5 firmware.

The 900MHz modules are technically not a series, but are a family just like the others. The 900s can run two different types of firmware, the DigiMesh firmware and the Point to Multipoint firmware. Digi does sell both modules, having the same hardware, just with different firmware. These modules should be ready to use out of the box, but of course can benefit from all the additional features that can be configured. The XSC is a 900 module that sacrifices data rate for range. The regular 900 modules have a data rate of 156KBps (the others are all around 250Kbps), but the XSC module is only about 10Kbps. On the other hand if a high gain antenna is attached the range of about 6 miles with a regular antenna was increased to about 15 miles. These modules do not require configuration out of the box and have some other differences including a different command set. The XSC S3B is an upgraded version of the XSC modules, which consumes less power than the previous generation despite having a higher selectable transmitting power of 250mW. This higher transmitting power allows for line-of-sight range up to 28 miles with the right antenna. The S3B modules also feature higher-throughput than the previous generation XSC modules.

There are a few differences between the regular X-Bees and the XBee-Pros. The Pros are a bit longer, use more power and cost more money. The greater power means longer range (1 mile instead of 300ft). X-Bees and XBee-Pros can communicated with each other on the same network.

The particular ZigBee device that was used for the AEAS system is the XBee-Pro RF module. The specifications of the XBee-Pro are depicted in Table 6 below.

Specification	Value
Performance	
Indoor/Urban Range	Up to 300 ft. (90 m), up to 200 ft. (60 m) International variant
Outdoor RF line-of-sight Range	Up to 1 mile (1600 m), up to 2500 ft. (750 m) international variant
Transmit Power Output (software selectable)	63mW (18dBm), 10mW (10dBm) for International variant
RF Data Rate	250,000 bps
Serial Interface Data Rate (software selectable)	1200 bps - 250 kbps (non-standard baud rates also supported)
Receiver Sensitivity	-100dBm (1% packet error rate)
Power Requirements	
Supply Voltage	2.8 – 3.4 V
Transmit Current (typical)	250mA (@3.3 V) (150mA for international variant) RPSMA module only: 340mA (@3.3 V) (180mA for international variant)
Idle / Receive Current (typical)	55mA (@ 3.3 V)
Power-down Current	< 10 μ A
General	
Operating Frequency	ISM 2.4 GHz
Dimensions	0.960" x 1.297" (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)
Antenna Options	Integrated Whip, Chip or U.FL Connector, RPSMA Connector
Networking & Security	
Supported Network Topologies	Point-to-point, Point-to-multipoint & Peer-to-peer
Number of Channels (software selectable)	12 Direct Sequence Channels
Addressing Options	PAN ID, Channel and Addresses
Agency Approvals	
United States (FCC Part 15.247)	OUR-XBEEPRO
Industry Canada (IC)	4214A XBEEPRO
Europe (CE)	ETSI (Max. 10dBm transmit power output)
Japan	R201WW08215111 (Max. 10dBm transmit power output)
Australia	C-Tick

Table 6: XBee-Pro Specifications ¹⁷

4.4. Design Approach

The design of the wireless communication system was broken into two parts, including the configuration and use of the digital aspects of the XBee-Pro and the hardware implementation of it into the AEAS system.

¹⁷ Digi International, Inc., "XBee®/XBee-PRO® RF Modules Product Manual v1.xEx - 802.15.4 Protocol" 2009.

4.4.1. Configuration and Use

There are many different ways the XBee-Pro can be used, and each choice results in configuring the module for it. This section follows along with the XBee/XBee-Pro datasheet (in Appendix) for easy reference when implementing. Another consideration for the digital design of the XBee-Pro was how to configure the module. Since the modules on the attachment and the server play different roles in the communication system, their designs will often be considered separately.

4.4.1.1. Serial Interface

The XBee-Pro interfaces through a logic level serial port that directly connects the Universal Asynchronous Receiver Transmitters (UARTs) of the microcontroller and the module when configured with compatible settings. We configured the interface data rate to 57,600 bps by sending a BD command with a 0x06 parameter to each module during setup, and the parity was configured to 8-bit odd parity by sending the NB command with a 2 parameter to each module. This interface data rate configuration satisfies the specification for the bandwidth between the attachment device and the server to be no slower than 50 Kbps. The odd parity configuration adds a needed level of data integrity.

Figure 12 shows the data flow of a UART data packet (0xE8) as it is transmitted through the XBee-Pro. The UART signal remains idle high until a packet of data comes along with a low start bit. The eight data bits are next, starting with the least significant bit and ending with the most significant bit. Then a parity bit calculated by either the microcontroller or the server is sent across. A high stop bit concludes the data packet and the UART signal is again idle high.

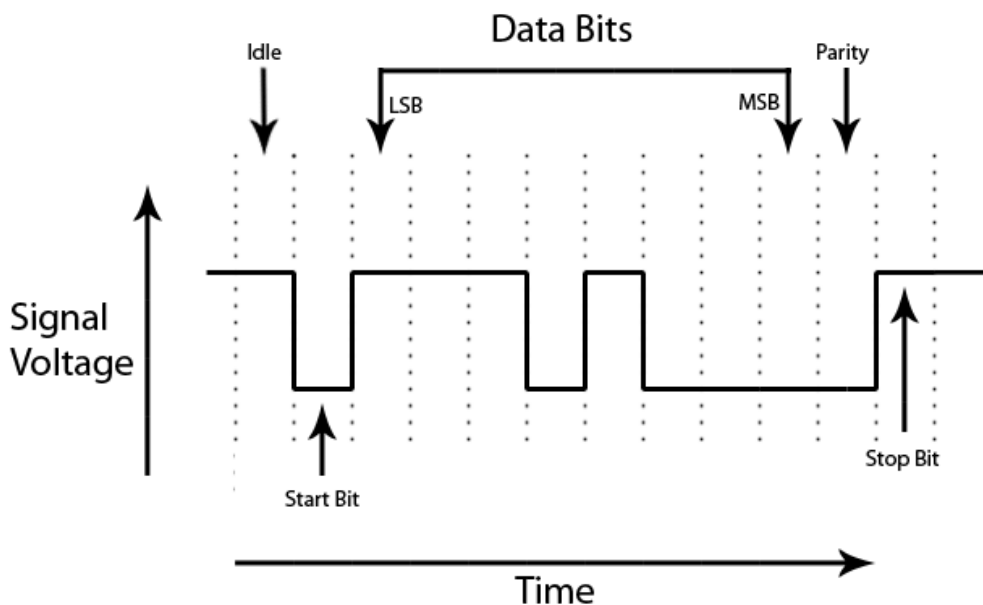


Figure 12: UART Data Flow for the XBee-Pro

4.4.1.2. Module Operation Mode

There are two modes of operation for the XBee-Pro that control how it responds to data received through the UART. Both modes of module operation are used for the AEAS system.

The module on the weapon attachment (field module) uses the Transparent (AT) operation mode, because it is designed for fast data transmitting to another XBee-Pro module. In AT operation the module transmits received UART data through RF immediately after receiving the last character of the data packet, or until the packetization timeout runs out if set. The packetization timeout is the number of character times to wait to transmit received data after receiving the last character of the data packet. In AT operation the receive buffer only has room for one hundred bytes of data, so flow control is often needed when a high packetization timeout is set. We configured the packetization timeout to zero character times for immediate transfer, by sending the RO command with the 0x00 parameter, to minimize delay in data transfer between the attachments and the server and avoid the need for added flow control. In AT operation the module takes the time to enter command mode to change a destination address. The attachment only transmits communication with the server, so the destination address always remains the same after configuration of the field XBee-Pro. Through research of the uses of AT operation, it was found that data is sometimes lost through transmission.¹⁸ As a result a self-made checksum was implemented in the data received by the server. The microcontroller of the body attachments computes an anti-checksum to send at the end of a ‘hit’ message. The server sums the data bits of the entire message, including the anti-checksum, to validate the data was received correctly before sending the ‘hit’ message to the GUI and projection. AT operation is the default mode of operation for the XBee-Pro module, but can also be configured by sending the AP command with the 0x00 parameter.

The module on the server (base module) will use the Application Programming Interface (API) operation mode, because it is designed to control an XBee-Pro or multiple XBee-Pro's. When using API operation the received data packets contain the source address, making it so the base module can easily address and receive from many field modules. The source data also includes other information on the transmitting module, like signal strength, that can be used to determining the integrity of the data received. API operation packetizes the data into frames. API operation implements acknowledgements and retries. The receiving module sends an acknowledgement back to the sender upon receipt. If the transmitting module does not receive an acknowledgment from the receiving module, it will continue to resend the message up to three times. The number of retries is configured to three by default, or by sending a MM command with a 0x00 parameter to enable the 802.15.4 MAC with a Digi header and a RR command with a 0x00 parameter. In API operation an implemented checksum is included in the data frame structure. There is another API operation mode that escapes necessary control characters.

¹⁸ SparkFun Electronics, 2010, <https://forum.sparkfun.com/viewtopic.php?f=13&t=21590>, Web, Oct 2013.

4.4.1.3. Network Topology

The XBee-Pro supports three different network topologies, including point-to-point, star, and mesh. These three networks can be visualized as shown in Figure 13. Point-to-Point is the simplest network topology supported by the XBee-Pro. It establishes a permanent link between two endpoints, ensuring unimpeded communication between them.

Star network topologies are the simplest multi-point networks. Each endpoint is connected to, and communicated directly with, a central hub with a point-to-point connection. A disadvantage of the star topology is that the central hub represents a single point of failure, meaning failure of the central hub renders the network inoperable.

Mesh is a type of network topology where each node must not only capture and transmit its own data, but also serve as a relay for other nodes. A mesh network can be designed using a flooding technique or a routing technique. When using a routing technique, the message is propagated along a path, by hopping from node to node until the destination is reached. To ensure all its paths' availability, a routing network must allow for continuous connections and reconfiguration around broken or blocked paths, using self-healing algorithms. The self-healing capability enables a routing based network to operate when one node breaks down or a connection goes bad. As a result, the network is typically quite reliable, as there is often more than one path between a source and a destination in the network.

While mesh networks may have been suitable for more complicated applications where power is readily available, there are issues with implementing these networks in a battery-powered environment. For the AEAS system to maximize reliability, operational efficiency, and budget a point-to-point network was implemented as the architecture model for the communication system. For scalability, the star network would be used.

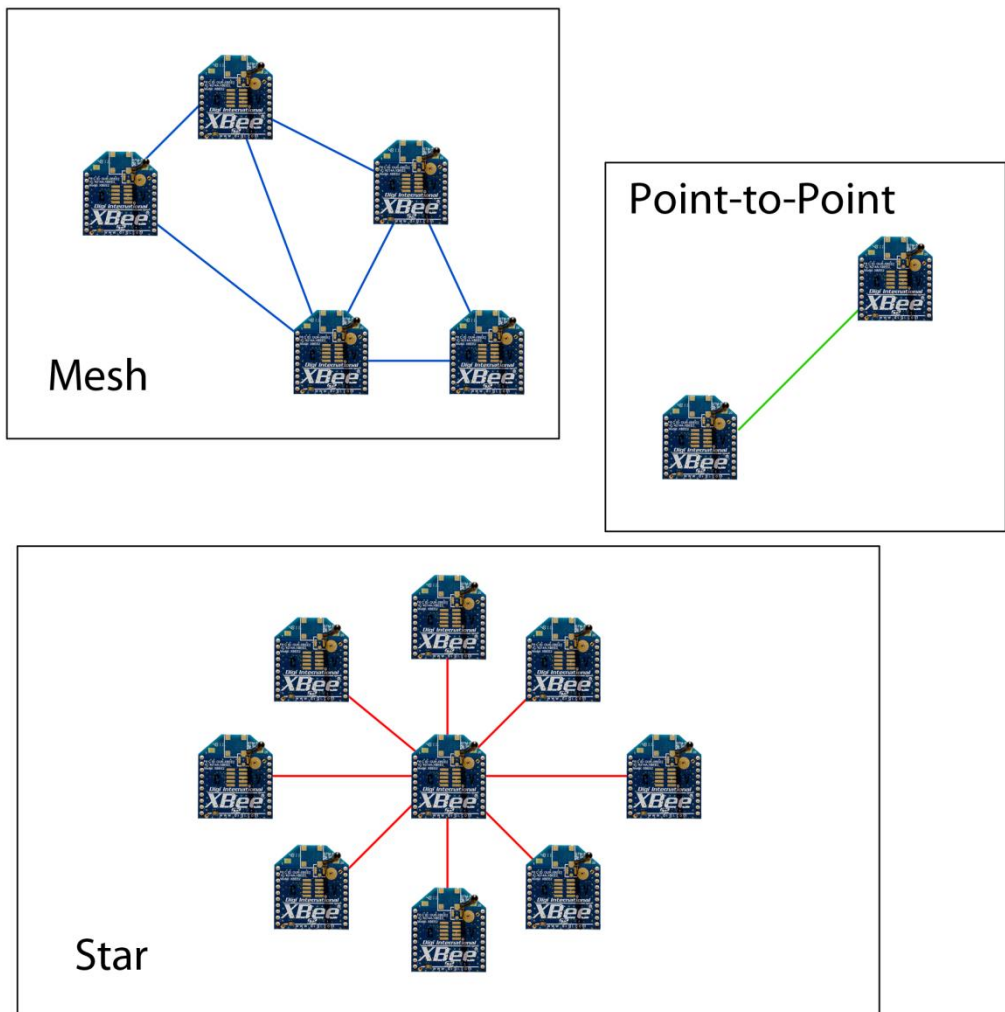


Figure 13: Network Topologies available to the XBee-Pro

4.4.1.4. Addressing

The XBee-Pro has two modes of addressing, including unicast and broadcasting. In unicast mode a message is sent to another particular XBee-Pro, indicated by the destination address. The receiving module sends an acknowledgement back to the sender upon receipt. If the transmitting module does not receive an acknowledgment from the receiving module, it will continue to resend the message up to three times. In broadcast mode DL is set to 0xFFFF and DL is set to 0x0000, resulting in any module within range to receive a transmitted message. There is no acknowledgement sent back and the message is only sent once. Since all of the AEAS system's communication will have an intended destination module, all of the XBee-Pro's will use unicast mode operating. This also increases the reliability of the transmitted RF data to reach its intended destination receiving module.

Each XBee-Pro can be addressed either with its unique factory set 64-bit IEEE serial number or a manually configured 16-bit source address. Though the short 16-bit source address is enough to meet the specification of accommodating simulations with up to twenty users, the AEAS system will use the long 64-bit address to avoid any interference with a possible nearby AEAS simulation or any other RF communication. Each module's MY (16-bit source address) was configured to 0xFFFF to indicate the 64-bit source addresses in SL and SH are to be used as its source address. The microcontroller on the attachment unit is pre-loaded with its assigned base module's source address to be loaded on the XBee-Pro's destination address. During the production of an AEAS system the server reads the base module's address using the SL and SH commands, then sends it to the field module to use as its destination address. To configure the destination address of the field modules the microcontroller sends the DL command with the parameter equal to the SL of the base module and the DH command with the parameter equal to the SH of the base module.

4.4.1.5. Configuration Set-Up

The free X-CTU software was used to configure the base module in the AEAS system. The provided X-CTU software was very easy to use and many tutorials were used for step-by-step set-up and use of the configuration software.

Remote configuration of the field modules is performed by the base module using the X-CTU software; however, it will take much longer to configure upon power-up. Since the field module is designed to always communicate with the same base module, all of the configuration settings were the same each time the AEAS system is used. For these reasons, remote configuration was used for configuring the field modules using the X-CTU software.

4.4.2. Hardware Implementation

Like the digital design, the hardware implementation of the communication system was different for the base module and the field module. Another big consideration is the type of antenna to be used on all of the XBee-Pros in the AEAS system.

4.4.2.1. Antenna Type

The XBee-PRO RF Module accommodates four antenna styles including the Dipole Chip Antenna, the Attached Monopole Whip, the U.FL, and the RPSMA. All of these antennas are omni-directional, which means the signal radiates in all directions. Two of the most popular versions are the PCB chip antenna and wire whip styles shown in Figure 14. The whip antenna has a range advantage over the chip antenna, but only outdoors. The PCB antenna will perform within 5% of the whip antenna, therefore outperforming the chip antenna. The XBee-PRO can achieve more range than the

XBee. The XBee-PRO and XBee both achieve more range outdoors than they do indoors. U.FL and RPSMA are connectors for attaching an external higher-gain antenna, such as a dipole. The RF modem encasement was designed to accommodate the RPSMA antenna option. The wireless communication system implements a PCB chip antenna for the field module to save space in the weapon attachment, since space had become a growing consideration. Although the other antennas have longer ranges the XBee-Pro has enough range using the PCB chip antenna to meet the requirements of the AEAS system. Since space was not a consideration when designing the server, we use the whip antenna to gain some range of communication.

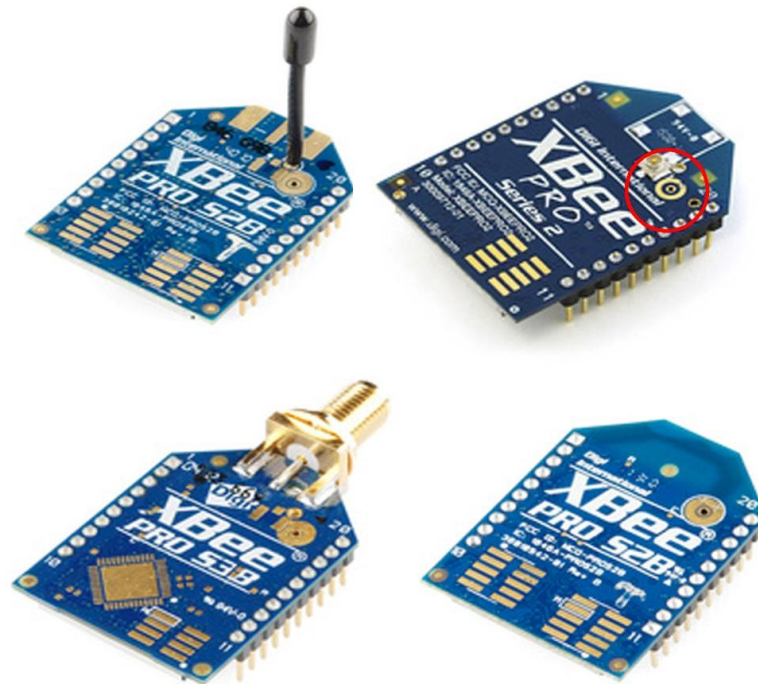


Figure 14: XBee-Pro Antenna Types²⁰

4.4.2.2. Base Module

The base station resides on the XBee USB Adapter, discussed in the Prototyping section, coupled with the server. This adapter incorporates the antenna, power, and EMC filtering circuits. Again, the server does not experience the space constraints of the weapon attachment, leaving this extensive attachment board to be the best choice for the base station. This also adds the indicative LEDs as a permanent debugger when using the AEAS. Additionally, the adapter boards are offered in bulk at a reduced price, making it also cost effective. The XBee-Pro need only be mounted and

²⁰ Images courtesy of Sparkfun Electronics. <https://www.sparkfun.com/products/10418>.
<https://www.sparkfun.com/products/11664>.
<https://www.sparkfun.com/products/10420>.
<https://www.sparkfun.com/products/10421>. Used under the Creative Commons License.

soldered to the adapter board to be ready for production.

The XBee USB Adapter Board, shown in Figure 15 below, was used for interfacing the base module to the AEAS server. This adapter provides “a serial interface using the FTDI USB serial interface chip (FT232RL) and virtual COM port drivers to emulate a serial COM port.”²¹ The XBee USB Adapter came unassembled, so the XBee-Pro needed only to be mounted into it.

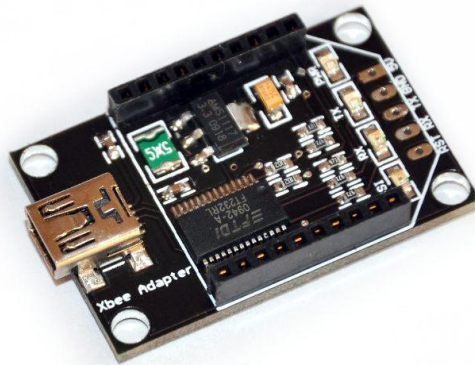


Figure 15: Base XBee-Pro Adapter Board

4.4.2.3. Field Module

For similar reasons the field module resides on an adapter; however, we used a much simpler one to save space in the weapon attachment. Three Parallax adapter boards were considered for prototyping the field module, all which are compatible with the chosen microcontroller.

The basic XBee Adapter Board is sufficient to adapt the XBee-Pro to interface a breadboard; however, it does not add any other features for monitoring or debugging²².

The XBee 5V/3.3V Adapter Board is also sufficient to adapt the XBee-Pro to interface a breadboard, using two eleven-pin headers with two buffered inputs (DIN and RTS). For protection, a diode connected to the Reset pin is connected to a diode to only allow the microcontroller to bring it low. This adapter board also allows for a direct connection between all other XBee-Pro's I/Os and the microcontroller. In addition, it has four different colored LEDs to indicate 3.3V power supply, association status, received signal strength indication (RSSI), and sleep status. The XBee 5V/3.3V Adapter Board comes unassembled, so soldering is required before it

²¹ Parallax Semiconductor Inc., “XBee USB Adapter Board”, 2011, <http://www.parallax.com/product/32400>, Web. Nov. 2013

²² Parallax Semiconductor Inc., “XBee Adapter Board” 2011, <http://www.parallax.com/product/32403> Web. Nov 2013.

can be used for prototyping.²³

The XBee SIP Adapter Board supports all the functions using the same or similar hardware as the XBee 5V/3.3V Adapter Board previously described. Additionally, it had two more indicative LEDs and more conditioned I/Os. The XBee SIP Adapter buffers inputs DIN, RTS, and DTR/Sleep and outputs DOUT, CTS, ASSOC, RSSI. The DOUT, DIN, and RTS pins are easily connected directly to the microcontroller, along with the 5 V supply and ground pins using a dual SIP solderable header. A five-pin female header provides easy interfacing to other important XBee-Pro pins, including ASSOC, RSSI, RST (low), SLEEP, and CTS. The AEAS system is planned to only implement the pins connected through the SIP interface, which makes the XBee SIP Adapter ideal for prototyping the XBee-Pro. If expansion is desired, the only other pins that may be required lie in the female header, which will keep the connections very simple. This adapter has six LEDs for indicating 3.3V supply power, sleep status, association, RSSI, data transmitted to XBee-Pro, and data received from XBee-Pro. From personal field experience in embedded systems, it has been learned that these additional LEDs can make a big difference when prototyping and debugging. Finally, The XBee SIP Adapter Board comes fully assembled, so no soldering is required before it can be used for prototyping.²⁴ For these reasons the XBee SIP Adapter Board was chosen for prototyping the XBee-Pro.

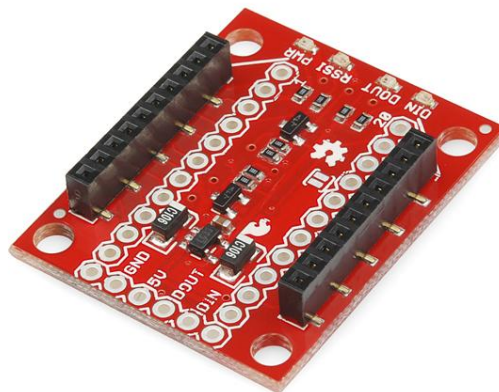


Figure 16: Field XBee-Pro Adapter Board

4.5. Prototyping

The field module was initially prototyped by separately connecting it to a microcontroller using a breadboard. The adapter board was necessary for the XBee-Pro to connect to the standard breadboard, because the pitch between the pins on the XBee-Pro is 2.54 mm and the pitch between the pins on a standard breadboard is 2.0

²³ Parallax Semiconductor Inc., “XBee 5V/3.3V Adapter Board”, 2011, <http://www.parallax.com/product/32401>, Web. Nov. 2013.

²⁴ Parallax Semiconductor Inc., “XBee SPI Adapter Board”, 2011, <http://www.parallax.com/product/32402>, Web. Nov. 2013

mm. The DOUT pin (2) and the DIN (3) of the XBee-Pro were connected to bits 1 and 0, respectively, on the microcontroller. These are the UART inputs and outputs. The VCC pin (1) was connected to the 3.3V power supply of the microcontroller, and the GND pin (10) was connected to the common ground. These were the only pin connections necessary for the initial prototyping of the field module

The base module was prototyped by connecting to a computer using a Windows OS, through a USB adapter board. There was no need for a breadboard in prototyping the base module, because the DOUT, DIN, VCC, and GND pins are connected properly to the computer through the USB interface.

4.6. Testing

Initially each XBee-Pro was tested using the free X-CTU software. The provided X-CTU software was easy to use and many tutorials were available for step-by-step test set-up and testing.²⁹ The free software has an easy-to-use loopback range test built into it for testing the range of the module.

²⁹ Digi International Inc., “Getting Started Guide for the ZigBee RF Module Kit”, 2012.

5. Weapon Attachment

This section outlines the weapon orientation module that can be attached to various weapons to provide information the server needs to create the scenario.

5.1. Description

The weapon attachment is a small unobtrusive 'box' designed to measure the position and orientation of the firearm as shown in Figure 17. This device also sends fire signals when the user presses the trigger. This unit has consistent communication with the main server to send position, orientation, and fire reports with which the system pieces together the scenario as it progresses.



Figure 17: Concept of Weapon Attachment Device with an M16A2

This device consists of the orientation subsystem, positioning subsystem, communications subsystem, and trigger sensing mechanism as shown in Figure 18. The means of attaching the unit to the weapon is a modified, releasable cable tie harness to provide a strong, unmoving attachment while also allowing the user to attach and remove the device with ease. Each of the components of the system were designed to be very modular and easily exchangeable with more current parts. This allows the weapon attachment to be incredibly upgradeable if more accurate positioning mechanisms are to be utilized or more precise inertial sensors are desired for example.

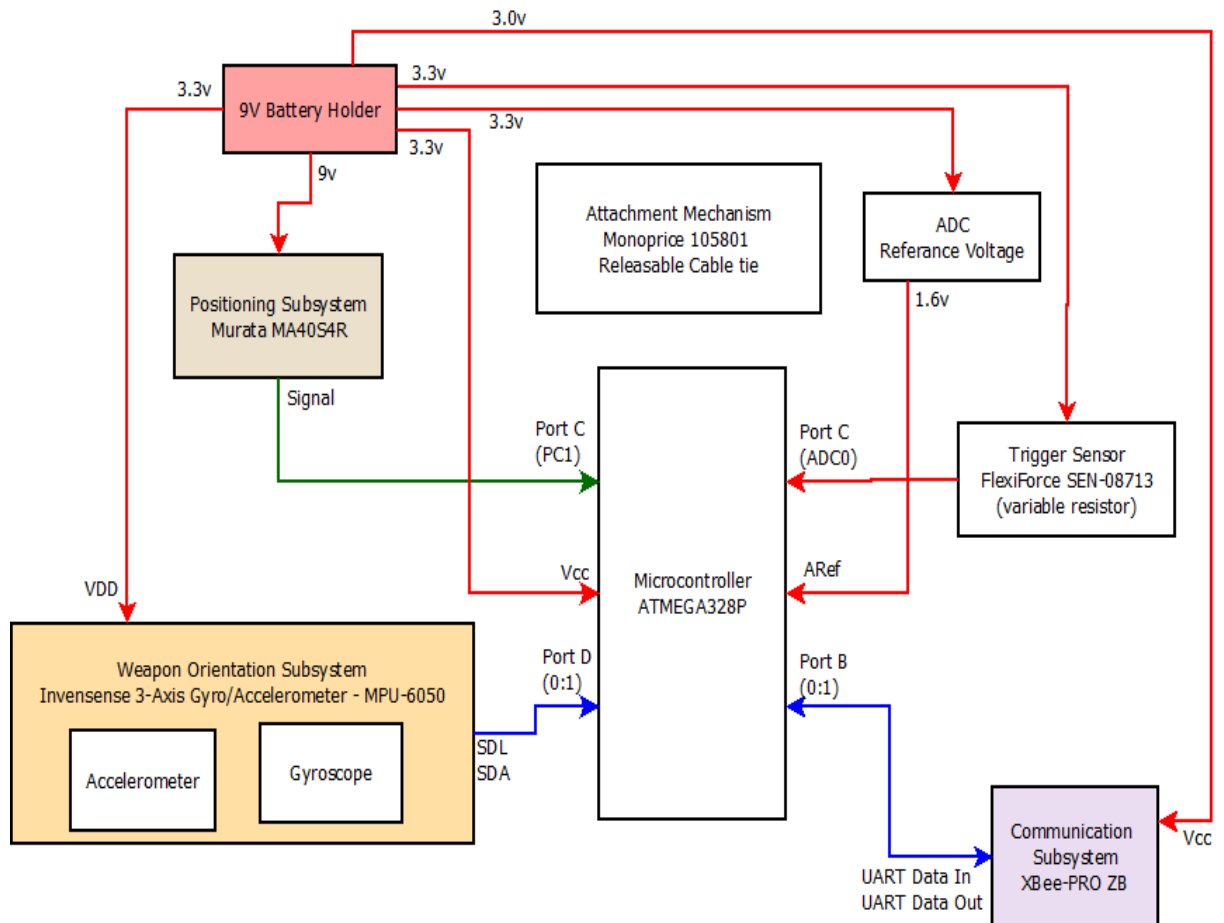


Figure 18: Weapon Orientation Module Block Diagram

5.2. Design Approach

In an attempt to keep our design simple and effective, we pressed forward with our design with the intention of meeting system requirements with the minimum amount of hardware and power requirements. This kept the weapon attachment small and easier to evaluate and modify if needed.

5.3. Organization

The weapon attachment organization is achieved using a simple 5" x 2.5" x 1.75" black project box. A releasable cable tie is attached to the external surface to allow attachment to the side of a weapon. The layout of each of the modules inside of the box is shown in . This organization gives optimal room for replacement of various modules as well as reduces the amount of wire crossing that is necessary to connect each of the devices.

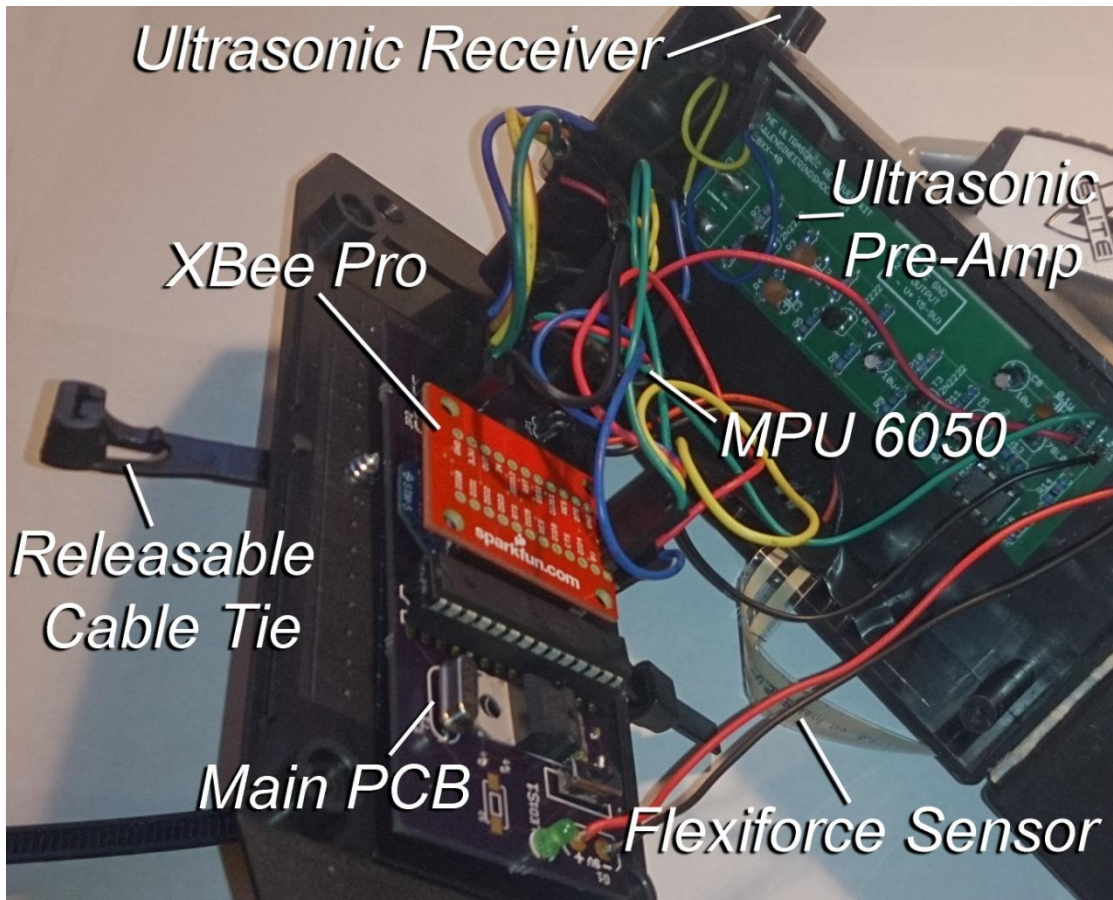


Figure 19: Weapon Attachment Organization

5.3.1. Microcontroller

We considered a number of different microcontrollers for use in our project, but finally determined the ATmega328 to be the best choice for our purposes. The 32K of program space, 20 MHz maximum clock rate, 6 ADC channels, and 23 I/O pins are more than adequate for our purposes³².

The internal clock for this chip runs at 8MHz, which is also more than adequate for our uses. However, the calibration performed in the Atmel factory is made at a fixed operating voltage and precisely 25 degrees Celsius³³. The internal oscillator is affected by operating voltage and temperature, which means in order to get the level of accuracy we need, we needed to connect an external oscillator. It was possible to configure the onboard oscillator to be more accurate however, but it proved more difficult than it was worth, so an external oscillator was the ultimate choice.

³² Atmel Corporation. "8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash". <http://www.atmel.com/Images/doc8161.pdf>. Rev. 8161D-AVR. Oct. 2009. Web. 24 Nov. 2013.

³³ Atmel Corporation. "AVR053: Calibration of the internal RC oscillator." <http://www.atmel.com/Images/doc2555.pdf>. 8-bit AVR Microcontrollers Application Note. Rev. 2555G-AVR. May, 2006. Web. 11 Nov. 2014

5.3.2. Orientation Subsystem

The orientation subsystem consists of an IMU (inertial measurement unit) which uses accelerometer and gyroscope data to make its calculations. This unit resides on the PCB with the microcontroller and provides orientation data that is processed and then forwarded on to the main server.

5.3.2.1. Inertial Measurement Unit

For this project we decided to use the Invensense MPU-6050 3-Axis Gyro/Accelerometer Motion Processing Unit (MPU) due to its low cost, ease of use, and adequate precision. An image of the device is shown in Figure 20. The integrated circuit includes a Digital Motion Processor (DMP) which offloads the computation of motion processing algorithms from the host processor. The chip supports the I²C communication protocol, which will allow for possible expansion of the module's capabilities with additional I²C devices. The device also has an auxiliary serial interface for communicating to an off-chip 3-Axis digital output magnetometer, which was very useful for the precision needed in the orientation calculations.

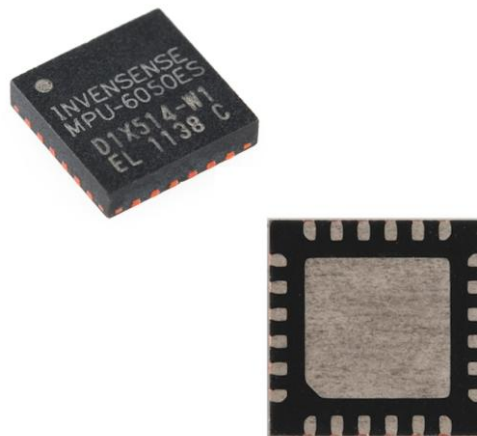


Figure 20: Invensense MPU-6050³⁴

The pin configurations on the chip are outlined in Figure 21. The descriptions of these pins are given in Table 5. Although only a breakout was used in the final design, understanding of these pins were important for us to know the configuration of the device for use.

³⁴ Image courtesy of Sparkfun Electronics. <https://www.sparkfun.com/products/10937>. Used under the Creative Commons License.

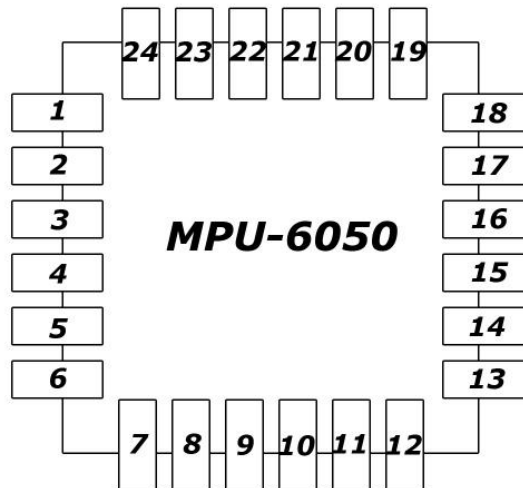


Figure 21: MPU-6050 Pin Arrangement³⁵

Pin #	Name	Description
1	CLKIN	Optional external reference clock input. Connect to GND if unused.
6	AUX_DA	I ² C master serial data for connecting to external sensors.
7	AUX-CL	I ² C master serial clock for connecting to external sensors.
8	VLogic	Digital I/O supply voltage.
9	AD0	I ² C Slave Address LSB.
10	REGOUT	Regulator filter capacitor connection.
11	FSYNC	Frame synchronization digital input. Connect to GND if unused.
12	INT	Interrupt digital output (totem-pole or open-drain).
13	VDD	Power supply voltage and Digital I/O supply voltage.
18	GND	Power supply ground.
19	RESV	Reserved. Do not connect.
20	CPOUT	Charge pump capacitor connection.
21	RESV	Reserved. Do not connect.
22	RESV	Reserved. Do not connect.
23	SCL/SCLK	I ² C serial clock.
24	SDA	I ² C serial data.
2,3,4,5,14,15,16,17	NC	Not internally connected. May be used for PCB trace routing.

Table 7: MPU-6050 Pin Descriptions³⁶

³⁵ Invensense Inc. "MPU-6000 and MPU-6050 Product Specification". <http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf>. Rev. 3.4. 19 Aug. 2013. Web. 26 Nov. 2013.

³⁶ Ibid.

Using the typical operating circuit available in the documentation for this device, the I²C bus has a specific timing characterization. The SCL clock frequency operates at a maximum of 400 kHz. The SCL low period occurs at a minimum of 1.3μs and the high period occurs at 0.6μs. SDA and SCL rise and fall times occur at a minimum of $20+0.1 \cdot C_b$ nanoseconds, where C_b is the capacitive load for each bus line, and a maximum of 300ns. SDA data setup time is 100ns, data valid time is 0.9μs, and data valid acknowledge time is also 0.9μs.

In general, the I²C lines are open-drain and bi-directional. The microcontroller puts the MPU slave address on the bus, and the MPU acknowledges the microcontroller as the master. SDA and SCL lines needed pull-up resistors to VDD. The slave address of the MPU-6050 is b1101005 which is 7 bits long. The LSB of the 7 bit address is determined by the logic level on pin AD0. The devices begin communicating on the bus when the master puts the START condition on the bus, which is a high-to-low transition of the SDA line while SCL is high. The bus is then considered busy until the master puts a STOP condition on the bus, which has a low-to-high transition on the SDA line while SCL is high.

I²C data is defined to be one byte long and there is no restriction to the number of bytes transmitted per call. Each byte must be followed by an acknowledge signal. This signal is generated by the master, and the receiver generates the actual acknowledge signal by bringing SDA to a low state while SCL is high during the acknowledge clock pulse.

After beginning communications, and master sending the 7 bit slave address over the line, the 8th bit indicates read/write (high for read, low for write). The master then releases the SDA line and waits for the acknowledge signal from the slave. Each byte must be followed by an acknowledge signal. Data communication terminates when a STOP signal is sent from the master. Except for start and stop conditions, all SDA changes should take place when SCL is low.

5.3.2.2. Orientation Calculations

A simple but elegant orientation calculation is made by first getting the accelerometer measurements and then making small corrections based on the last orientation and gyroscope data as shown in Figure 22. Although the actual orientation calculation algorithm utilized in the MPU 6050 is not publicly known, this analysis is useful in determining ways to utilize the data received more effectively. The MPU 6050 needs to be programmed every time at system startup.

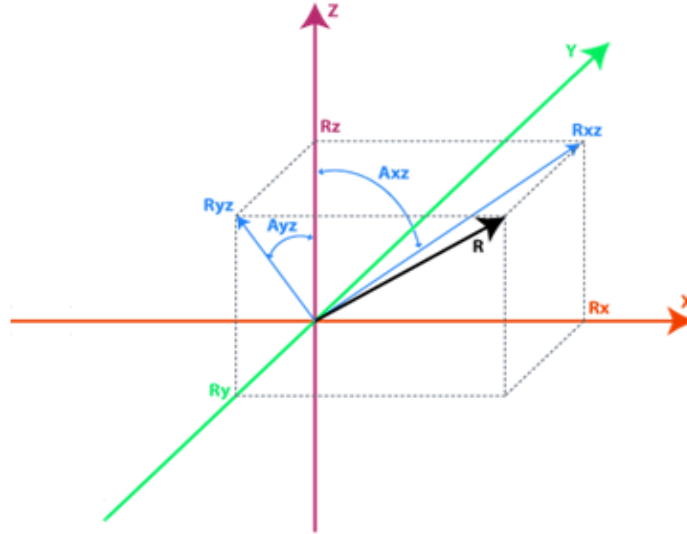


Figure 22: Orientation Measurements

Consider (FIGURE REF) where R is the orientation vector we are trying to obtain. From the right-angle triangle formed by Rz and Rxz, we derive:

$$\tan (Axz) = \frac{Rx}{Rz} \rightarrow Axz = \text{atan2}(Rx, Rz)$$

$$\tan (Ayz) = \frac{Ry}{Rz} \rightarrow Ayz = \text{atan2}(Ry, Rz)$$

We use atan2 because it returns values in the range of $(-\pi, \pi)$ as opposed to $(-\pi/2, \pi/2)$ returned by atan. We then convert the values of Rx, Ry and Rz to angles in the full range of 360 degrees.

We find the previous values of Axz and Ayz to use in the calculation:

$$Axz_{prev} = \text{atan2}(Rx_{prev}, Rz_{prev})$$

$$Ayz_{prev} = \text{atan2}(Ry_{prev}, Rz_{prev})$$

The gyroscope measures the rate of change of the Axz angle. So we estimate the new angles Axz and Ayz using an average rotation rate as follows:

$$Axz = Axz_{prev} + \frac{\text{RateAxz} + \text{RateAxz}_{prev}}{2} * T$$

$$Ayz = Ayz_{prev} + \frac{\text{RateAyz} + \text{RateAyz}_{prev}}{2} * T$$

Now we just calculate Rx_{gyro} and Ry_{gyro} :

$$Rx_{gyro} = \frac{\sin(Axz)}{\sqrt{1 + \cos(Axz)^2 * \tan(Ayz)^2}}$$

$$Ry_{gyro} = \frac{\sin(Ayz)}{\sqrt{1 + \cos(Ayz)^2 * \tan(Axz)^2}}$$

Rz_{gyro} is calculated using Rx_{gyro} and Ry_{gyro} using the Pythagorean Theorem. However, this relationship does not allow us to know if Rz_{gyro} is positive or negative. Therefore, we make the assumption that the sign is equal to the previously derived Rz value.

$$Rz_{gyro} = \text{Sign}(Rz_{prev}) * \sqrt{1 - Rx_{gyro}^2 - Ry_{gyro}^2}$$

$$\text{where } \text{Sign}(Rz_{prev}) = \begin{cases} 1; Rz_{prev} \geq 0 \\ -1; Rz_{prev} < 0 \end{cases}$$

Note that as Rz approaches zero values may overflow and give us bad results since it is used as the reference axis for calculating Axz and Ayz . In our scenario this will occur when the weapon is pointed nearly straight up or down, so we simply set R_{gyro} equal to R_{prev} since it is unlikely that precise orientation measurements was needed in this case.

We use a weighted average between accelerometer data and the calculated gyroscope measurements to get the final estimated values:

$$R_{xyz} = \frac{R_{acc} + R_{gyro} * w}{1 + w}$$

where R_{acc} and R_{gyro} are 3 dimensional vectors. The variable w in the above formula tells us how much we trust the derived gyroscope data over our accelerometer data. According to our consulted literature, values between 5 and 20 will typically give good results³⁷. This vector is then normalized and sent to the server.

5.3.3. Fire Signal Detection

To detect when the user presses the trigger, we used a modified FlexiForce[®] 25lb pressure sensor. The sensor acts as a variable resistor. When pressure is applied to the head section, resistance decreases from infinite to approximately 300K Ω . The sensor itself is flexible and thin, but resistance does not change while flexing.



³⁷ Autodesk Inc. "Accelerometer & Gyro Tutorial." <http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/step3/Combining-the-Accelerometer-and-Gyro/>, 2013. Web. 17 Nov. 2013.

Figure 23: FlexiForce A201 Sensor. Reprinted with permission from Tekscan Inc.

The weapon attachment will interface with this device. Due to its flexible design, it is attached to the trigger with minimal interference to the user as shown in Figure 24.



Figure 24: Trigger with FlexiForce Sensor Attachment

The sensor is attached to wires which lead to the attachment unit. A small threshold pressure of 4.5 lbs. was set after analysis of experimentation and used to determine when to send a 'shot' fired signal. If the trigger is pressed past the threshold pressure, the attachment controller sends the signal to the server as well as up-to-date orientation and position data for processing.

Connections are placed with the comparison voltage and an ADC input on the microcontroller. When the value received by the microcontroller reaches the threshold, the program sends the appropriate signal to the XBee Pro RF module, which then transmits the data to the main server.

The ATmega328 provides a 10-bit ADC, which means that the analog to digital conversion result for a single ended conversion is:

$$result = \frac{V_{in} * 1024}{V_{ref}}$$

5.3.4. Data Processing

The ATmega328 needs to communicate with four external devices for the weapon module: the MPU-6050 Motion Processing Unit (MPU), the FlexiForce sensor, the XBee Pro RF Module, and the ultrasonic receiver.

The MPU conveniently provides its own processor on board which means minimal orientation processing for the host microcontroller. All the microcontroller needs to do is apply the calculations outlined in section 5.3.2 to provide a more accurate reading to the server when needed. This device also simply requires two wire

connections for the I²C interface: serial data (SDA) and serial clock (SCL). The MPU will always operate naturally as a slave device to the microcontroller with a maximum bus speed of 400 kHz.

Accelerometer data may also be used to augment position data by dead reckoning³⁸. Measuring acceleration deviations over time between the last received position update, and the current time can give a more precise location report. Once acceleration data is obtained, it can be integrated to obtain inertial frame velocity and position. Since this data was received at discrete time intervals, velocity and position can be estimated by the following equations:

$$v = v_{prev} + a * T$$

$$R = R_{prev} + v * T$$

where T is the sampling period. Applying this calculation to each of the axes in three dimensional space we get the following:

$$v_x = v_{x\ prev} + a_x * T$$

$$R_x = R_{x\ prev} + v_x * T$$

$$v_y = v_{y\ prev} + a_y * T$$

$$R_y = R_{y\ prev} + v_y * T$$

$$v_z = v_{z\ prev} + a_z * T$$

$$R_z = R_{z\ prev} + v_z * T$$

The initial position R_0 for our uses is given by the most recent position update received by the positioning system. In this way, we reduce error due to sensor inaccuracy. A trajectory error of even a single degree will cause the estimated velocity to be off by more than a 17 meters after 10 seconds³⁹.

5.3.5. Power

Power requirements for each of the components of the Weapon Orientation Module are outlined in Table 6.

Part	Voltage	Clock Rate	Operating Current
ATmega328P (Active Mode)	1.8V	< 1MHz	0.2-0.5mA
ATmega328P (Power-Save Mode)	1.8V	< 1MHz	0.75μA
MPU-6050	2.375-3.46V	1kHz	3.9mA
XBee (Tx/Rx)	2.8-3.4V	-	45-55mA

³⁸ CH Robotics. "Estimating Velocity and Position Using Accelerometers". <http://www.pololu.com/file/0J587/AN-1007-EstimatingVelocityAndPositionUsingAccelerometers.pdf>. AN-1007 rev. 1.0. 19 Oct, 2012. Web. 11 Nov. 2014.

³⁹ Ibid.

5.4. Prototyping

In prototyping our weapon attachment, we mainly used an Arduino UNO, which acted as a simple replacement for our custom PCB during initial testing. Each of the module breakouts were connected to their various inputs on the board where it was available, but there were more required pins than existed on the UNO itself. To overcome this issue, we used a RadioShack perforated circuit board to connect all of the additional wires that were needed by the various devices.

Onto the perforated board was soldered a 5v regulator and several wires leading from the various connections to the regulator. 9v from the battery holder was connected directly to the input of the 5v regulator, with an additional wire from that input to supply voltage to the ultrasonic amplifier circuit since it has its own onboard regulator. Several wires were soldered to the ground pin and the output of the regulator to supply sufficient power to the XBee Pro and to the Arduino UNO.

A prototype of the actual circuit on our PCB was put together on a breadboard to do circuit testing as shown in Figure 25. This implementation was never actually used to do full system testing since moving the breadboard seemed to create errors in the circuit. We assume this is because the wire connections to the breadboard are designed to be motionless.

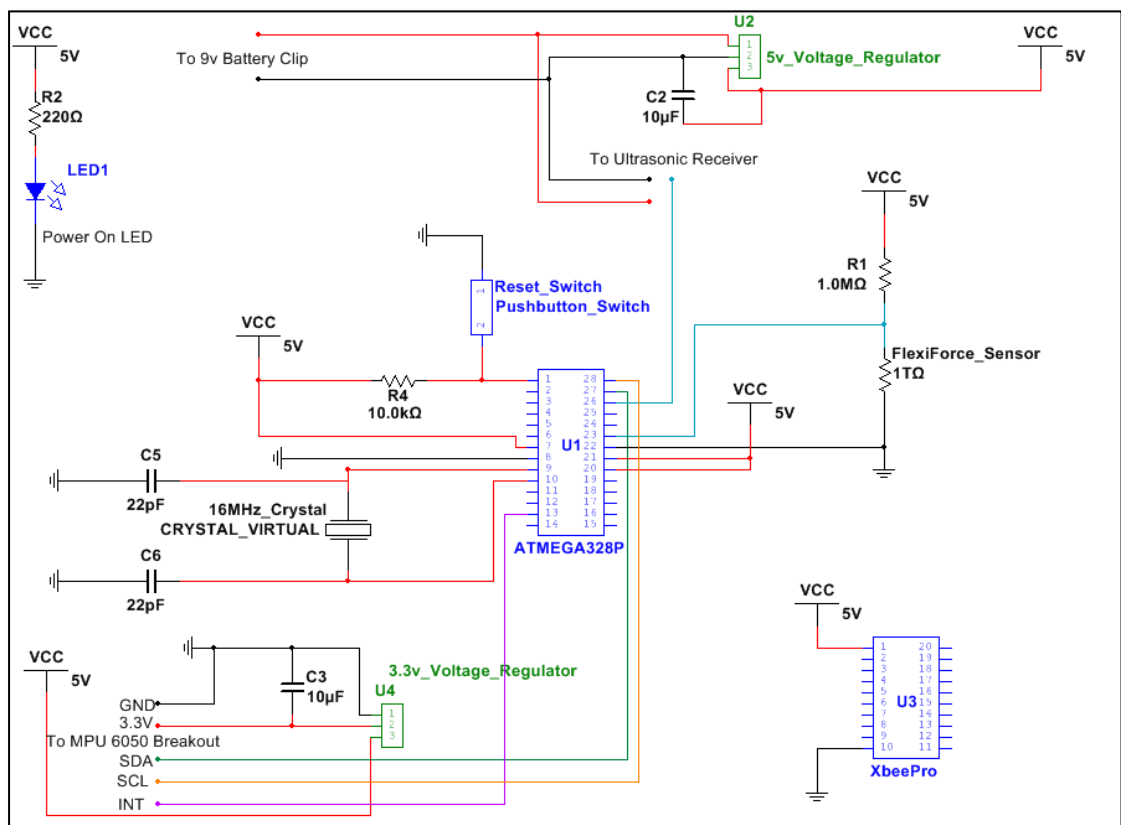


Figure 26: Prototype Circuit Diagram

The weapon attachment was tested extensively with a breadboard design using the breakout board available for the MPU-6050. The original configuration presented

here was tested and modified to determine the optimal solution for use in our final prototype, which was to use a pre-manufactured printed circuit board. This way potential errors could be reduced, and the orientation module itself could be removed and exchanged at will for simple upgradeability or maintenance.

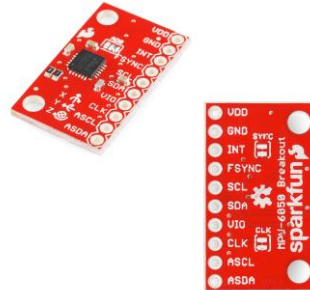


Figure 27: MPU-6050 Breakout Board⁴⁰

5.4.1. Printed Circuit Board

The module's final printed circuit board (PCB) was designed using Eagle, and manufactured by OSH Park services. Our breadboard circuit prototype was tested extensively before the final board was laid out in Eagle. The resulting printed circuit board layout is shown in Figure 28. The space in the center was made to house the Xbee Pro. The layout was designed to provide easily accessible headers to connect each of the additional sub-modules.

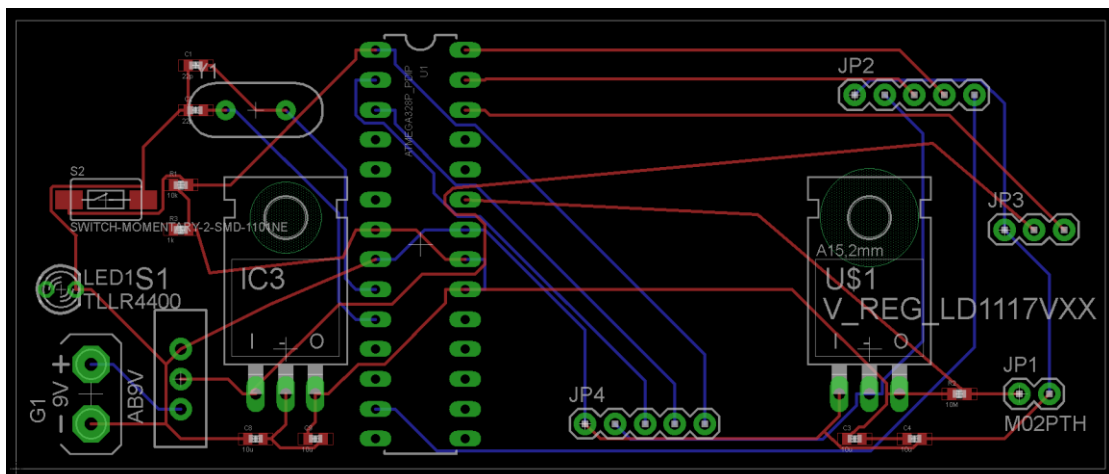


Figure 28: PCB Layout Diagram

Easily seen are the two voltage regulators. The one to the left of the microcontroller marks the 5v regulator which powers the microcontroller, XBee Pro, and flexiforce

⁴⁰ Image courtesy of Sparkfun Electronics. <https://www.sparkfun.com/products/11028>. Used under the Creative Commons License.

sensor. The one far to the right is the 3.3v regulator which delivers power to the MPU6050

5.4.2. Development Kits

To acquaint ourselves with Atmel programming and with accelerometers and gyroscopes, we purchased and evaluated the Atmel UC3-L0 Xplained development board and ATAVRSBIN2 Inertial Two Sensors Xplained attachment. Using this device combination and the Atmel Data Visualization Tool we were able to get a good first insight into microcontroller programming for sensor IC interaction. Visual data from some of our tests are shown in Figure 29 and Figure 30.

Although this development board does not utilize the same inertial sensors that we utilized in our final prototype, the insights gained from programming this device were invaluable in our development. Some of the code was able to be reused from our initial tests and provided a good starting point from which we could continue providing functionality for the system.

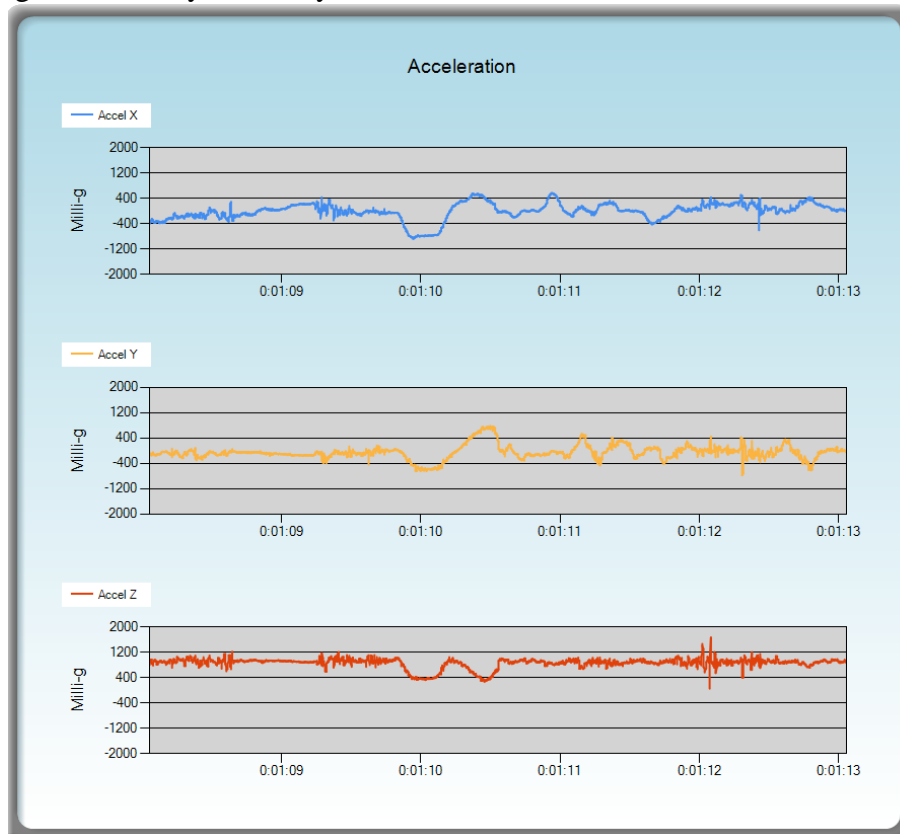


Figure 29: Atmel Data Visualizer - Accelerometer Data



Figure 30: Atmel Data Visualizer - Gyroscope Data

This data could also be logged by the tool and stored locally for further analysis. We used this feature to perform test analyses on received orientation data to determine system accuracy. The results were quite impressive; however, since we are not using the development kit in the actual prototypes of our devices, the results were largely meaningless to our project analysis. However, it did show that precise enough calculations were indeed possible, which means this effort could indeed exceed performance expectations.

We then purchased a breakout board for the Invensense MPU-6050 device so that we could do some initial tests using the same program. The breakout board was connected to a breadboard and our ATmega328P microcontroller which forwarded the measured data to the PC via USB UART simulation.

5.5. Testing

The most important capability of the weapon attachment is the orientation accuracy. Therefore extensive testing was performed to measure and refine this capability. Trigger sensitivity was also tested to assure the unit could effectively replicate the behavior of actual weapon trigger systems.

5.5.1. Orientation Accuracy

To evaluate the orientation precision and accuracy of the attachment, we did an initial assessment with various readings at different angle changes ranging from 15 to 90 degrees as shown in Figure 31.

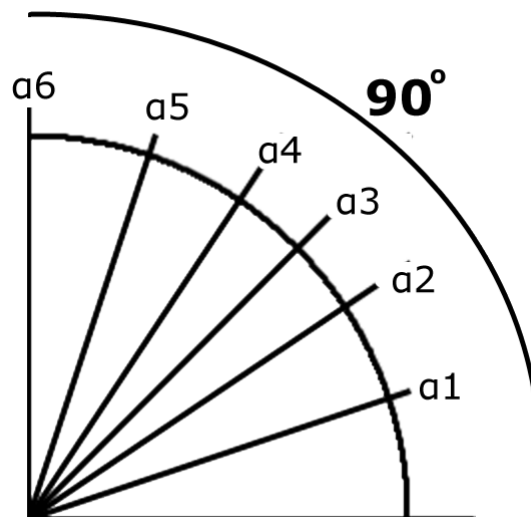


Figure 31: Angle Inclination Tests

The angle changes were made using a steady hinge with extension attached to a hard flat surface such as a table top. The module was fixed to one end of the free motion arm and in a specific orientation based on the axis we wished to measure. Data collection initialized and the arm was inclined to the desired angle at various speeds. The MPU allows the gyroscope to be set to a full-scale range of ± 250 , ± 500 , ± 1000 , or ± 2000 degrees per second (dps), and the accelerometer to be set to $\pm 2g$, $\pm 4g$, $\pm 8g$, or $\pm 16g$.

De Koning and colleagues measured the angular velocity about the elbow joint for three groups of subjects: untrained women, untrained men, and arm trained men such as athletes, weightlifters, bodybuilders, etc. Their data reported a maximum angular velocity of 14.9 ± 1.3 , 16.6 ± 1.5 , and 17.0 ± 1.6 radians per second respectively. Converting these into degrees per second, the trained men data indicates a theoretical maximum of just slightly over 1000 dps. Since we wanted our calculations to be as precise as possible, and increasing the full-scale range of the device results in less precision, so we did consider the ± 2000 dps setting for use in our system since it is highly unlikely that a user was moving at that velocity.

Tests were performed at each scale range up to ± 1000 and at each angle inclination up to 90 degrees. Tests were performed at approximate angular velocities 5, 15, 30 and 300 dps and data was sampled at no less than 1 kHz. The data was logged and analyzed over time to determine accuracy with linear movements before more irregular movements are tested. The 5 and 15 dps results gave us a good indication of how accurate very small precise movements was measured, and the 30 and 300 dps tests provided estimates of larger movement orientation accuracy. A list of test

parameters is defined in Table 9 below.

Test #	Gyro Full-scale Range	Sensitivity (LSB/dps)	Inclination Angle (degrees)	Approximate Angular Velocity (dps)	Approximate Duration
X-1	±250/±500/±1000	131/65.5/32.8	15	5	3s
X-2	±250/±500/±1000	131/65.5/32.8	15	15	1s
X-3	±250/±500/±1000	131/65.5/32.8	15	30	500ms
X-4	±250/±500/±1000	131/65.5/32.8	15	300	50ms
X-5	±250/±500/±1000	131/65.5/32.8	30	5	6s
X-6	±250/±500/±1000	131/65.5/32.8	30	15	2s
X-7	±250/±500/±1000	131/65.5/32.8	30	30	1s
X-8	±250/±500/±1000	131/65.5/32.8	30	300	100ms
X-9	±250/±500/±1000	131/65.5/32.8	45	5	9s
X-10	±250/±500/±1000	131/65.5/32.8	45	15	3s
X-11	±250/±500/±1000	131/65.5/32.8	45	30	1.5s
X-12	±250/±500/±1000	131/65.5/32.8	45	300	150ms
X-13	±250/±500/±1000	131/65.5/32.8	60	5	12s
X-14	±250/±500/±1000	131/65.5/32.8	60	15	4s
X-15	±250/±500/±1000	131/65.5/32.8	60	30	2s
X-16	±250/±500/±1000	131/65.5/32.8	60	300	200ms
X-17	±250/±500/±1000	131/65.5/32.8	75	5	15s
X-18	±250/±500/±1000	131/65.5/32.8	75	15	5s
X-19	±250/±500/±1000	131/65.5/32.8	75	30	2.5s
X-20	±250/±500/±1000	131/65.5/32.8	75	300	250ms
X-21	±250/±500/±1000	131/65.5/32.8	90	5	18s
X-22	±250/±500/±1000	131/65.5/32.8	90	15	6s
X-23	±250/±500/±1000	131/65.5/32.8	90	30	3s
X-24	±250/±500/±1000	131/65.5/32.8	90	300	300ms

Table 9: Orientation Accuracy Tests

The sensitivity of the gyroscope is directly affected by the full-scale range. The MPU

employs 16 bits of analog to digital conversion, which means 2^{16} or 65,536 unique values. Taken over the entire negative to positive range, the sensitivity was calculated as follows:

$$Sensitivity = \frac{2^{16} \text{ LSB}}{\text{Range dps}}$$

So as an example, when the gyroscope is set to a full-scale range of ± 500 , the total range is -500 to +500 which are 1000 units.

$$Sensitivity_{500} = \frac{2^{16}}{1000} = 65.536 \cong 65.5 \frac{\text{LSB}}{\text{dps}}$$

In other words, you get approximately 65 least significant bits for every degree per second of angular velocity. Taken at face value then, when set at a full-scale range of ± 250 , we get about 131 bits per degree per second, which makes our theoretical minimum margin of error for orientation approximately ± 0.008 degrees. The actual margin of error from experimentation is much higher of course, but accelerometer data helps to keep the gyroscope measurement errors to a minimum.

The tests outlined above was also tested with various accelerometer full-scale range settings as well. Extensive testing of this portion of the project was important due to the accuracy requirements of the system to achieve realistic fire capabilities.

5.5.2. Trigger Sensitivity

An evaluation of trigger pulls was conducted by Glockmeister with various added parts. They measured Peak Take-up Pull (PTP), the peak pressure measure to a point before the trigger break, and Trigger Break Pull (TBP), the pressure required to release the firing pin at the final stage of the trigger pull⁴¹. The averages of these measurements (in lbs) are shown in Table 10 below.

The value we are most interested in is the TBP, since that is what ultimately results in a 'shot' being fired. Taking an average of the TBP values obtained by Glockmeister yields 4.875 lbs. An M16 military rifle has a trigger pull of about 5 lbs as well⁴². Therefore, we will have an initial threshold estimate of 5 lbs, but it was adjusted to a lower value of 4.5 lbs for our prototype since we were unable to use actual military weapons in most of our tests and final demonstration.

⁴¹ Glockmeister, LLC. "Evaluation of Trigger Pulls". <http://www.glockmeister.com/evaluationoftriggerpulls.asp>. 2013. Web. 24 Nov. 2013.

⁴² Answers Corporation. "How much pressure is put on the trigger of a gun for it to fire". http://wiki.answers.com/Q/How_much_pressure_is_put_on_the_trigger_of_a_gun_for_it_to_fire#slide2. 2013. Web. 24 Nov. 2013.

Part/s added	PTP	TBP
Stock Configuration	3.0	5.25
Wolff reduced power firing pin spring	2.0	4.0
Glock 3.5 Pound Connector (G35C)	3.0	4.0
Glock 3.5 Pound Connector (G35C) and a WRPFPS	2.0	3.375
G35C and a Glockmeister Heavy Trigger Spring (GHTS)	2.25	3.75
S35C and a GHTS	2.25	3.75
New York Trigger Spring Olive (NYTSO)	4.375	7.625
NYTSO and a WRPFPS	3.125	6.250
G35C and a NYTSO	4.375	6.0
G35C, NYTSO, and a WRPFPS	3.125	4.75

Table 10: Evaluation of Trigger Pulls⁴³

⁴³ Ibid.

6. Server

This section outlines the server that uses the information from the weapon attachment, transmitted through the RF communication system, to create the scenario and provide a visualization of the current state.

6.1. Description

The AEAS system includes a distinct combination of hardware and software for the purpose of piecing together the current state of the combat scenario. Upon certain conditions, such as the notification that a ‘shot’ has been fired by a weapon attachment, the main server acquires orientation and position information for the weapon, and performs precise floating point calculations to determine the trajectory of the fired ‘shot’. The trajectory of the ‘shot’ was calculated based on the firing weapon’s position, and orientation. Using this information, the main server then determines if any target is positioned along the trajectory, and sends a signal to the projected display that has been ‘hit’ or is targeted by the ‘shot’. In addition, the main server also allows trainers to monitor scenario in real-time through a front-end web accessible GUI.

The minimum main server hardware is a low cost Raspberry Pi Model B computer running Debian Linux; however, a server application was developed for any NodeJS capable machine. The main server was running a Linux distribution due to the low system requirements, wide hardware compatibility, and zero cost associated with the operating system. The use of a dedicated Mac or Windows-based machine is also a viable platform for deployment, but was found to be much more processing power than is required, and also much more expensive due to the need for more hardware and operating system licenses. This main server was tasked with receiving weapon state information, providing trajectory calculations, determining if a trajectory intersects with a target, relaying ‘hit’ information to the projection, and providing a web accessible GUI for real-time monitoring of scenarios in progress as shown in Figure 32. As a result, the server must be capable of floating point operations, networking, and communication with the weapon attachment. Additional RF hardware was required to implement communication between the server and weapon attachments and to interface with the server through USB, or serial communications methods.

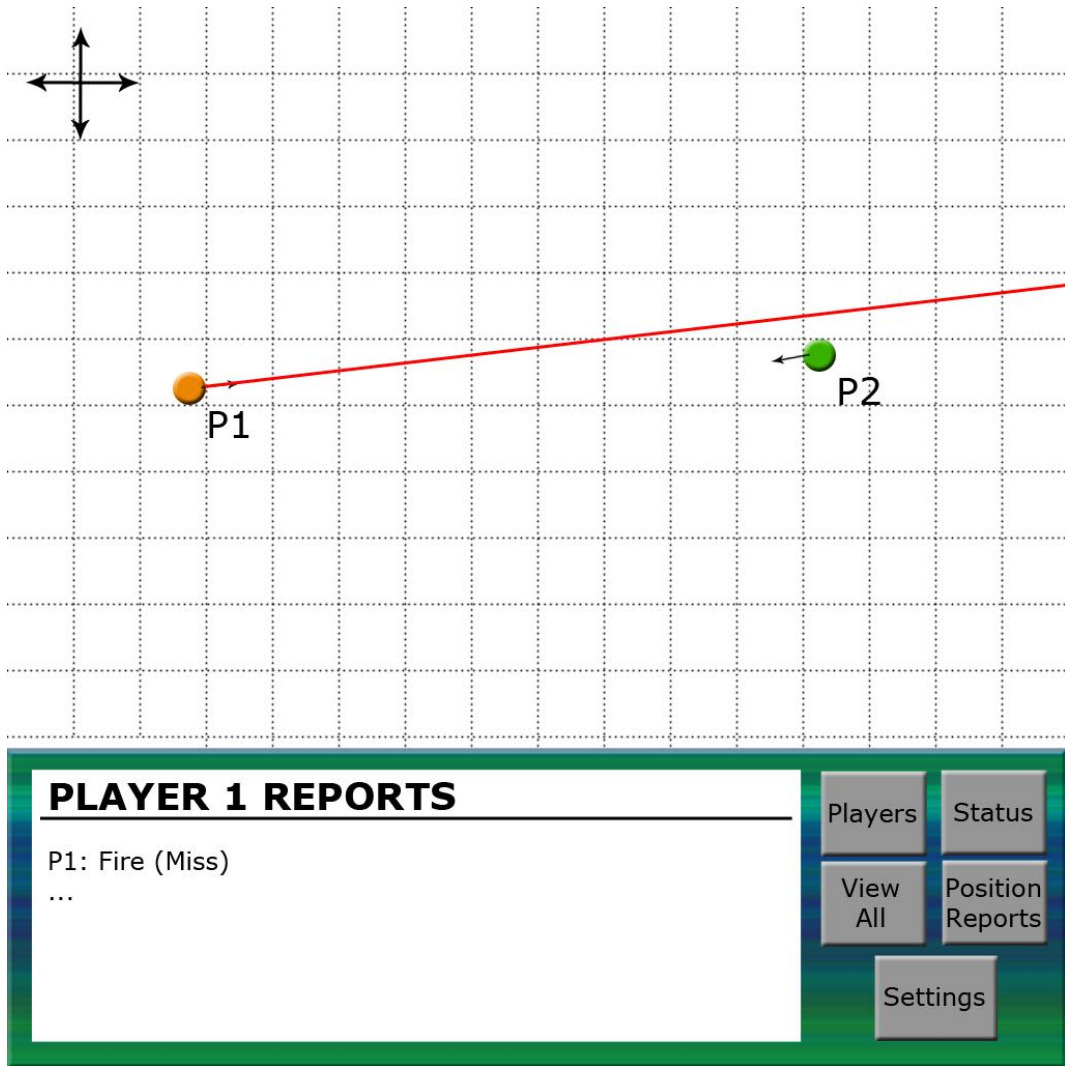


Figure 32: Initial AEAS GUI Concept

The front-end web accessible GUI feature provides trainers and other observers with a clear visual representation of the current scenario by simply opening a web socket connection to a known location (the server) on a local area network. This web functionality allows for real-time viewing of the designated training area, complete with user movements, weapon orientation, and the user's current target while also allowing for scenarios to be monitored from virtually anywhere. Ease of use of the GUI was greatly considered to ensure the usefulness of the system in training situations. The target display and position/orientation display are located within the same web page, and can be accessed by simply scrolling to the desired view (both of which are shown below in section 7.4.3).

6.2. Server Hardware

To ease the communications and computational load on the attachment device, it was decided that a hardware device should exist to perform the trajectory calculations,

'hit' assessments, and aggregation of the attachment data to form the scenario's state. In allocating a dedicated server component to the system, the processing power required by the attachment was minimized considering that those devices need only communicate with a central server, and determine their associated information (position, orientation). By eliminating the need for the attachment device to be able to compute trajectories and possibly determine if the associated target intersects with a trajectory, the frequency of computation was therefore reduced. This reduction allowed for simpler microcontrollers to be used, while also reducing power consumption of devices. Relatively simple, low power attachments were desired to ensure that the form factor of the attachment could be kept to a minimum to reduce the invasiveness of attachment on its user, and are more easily realized by using a system architecture with a central server.

To implement the proposed server component, the hardware had to be able to perform precise trajectory calculations, send information to all attachments, receive information from all attachments, and be accessible via the web. As a result, the selected components had to be capable of high precision floating point operations as outlined in the IEEE 754 standard, be capable of receiving a serial or USB device for communications, and have support for TCP/IP networking. As a result, the vast majority of personal computers available greatly exceed these minimal requirements, thus eliminating the need for specialized server components, such as Intel's Xeon product line, to be used in this implementation.

In the system being developed for demonstration, a complete weapon attachment was built, thus the server hardware had to be capable enough to quickly process any information received, and send the required information with a minimal delay. The system was expected to be highly extensible, so any server hardware implementation suggested was more than adequate to handle the needs of only user. Possible server implementations include hardware based on x86 or ARM architectures, both of which are readily available with floating point support and at wide range of performance options. Both architectures are also compatible with a wide range of operating systems, which allows for an operating system to be selected which either provides the most convenience to the user, or the greatest performance on the available hardware.

For the purpose of this project, an ARM-based implementation was more than adequate to fulfill the needs outlined in the specifications; however, for an extended implementation with a large number of users, an x86 machine would be necessary to provide the desired responsiveness and accuracy from the system due to the availability of higher performance chips.

Memory requirements are small since the system is only required to run an operating system, store location and orientation data for at least two users, compute trajectory and 'hit' results, and serve as a simple web interface for trainers. Some example memory requirements for operating systems are shown in Table 11, and were used as a reference in determining the memory needs of the system.

Operating System	Minimum Requirement	RAM	Disk Requirement	Space
Linux (Command Line Interface) ⁴⁹	64 MB		1 GB	
Linux (GUI)	384 MB ⁵⁰		15-25GB ⁵¹	
Microsoft Windows (GUI) ⁵²	1 GB (32-bit)/2 GB (64-bit)		16 GB	

Table 11: Typical Memory Requirements for Operating Systems

To be a viable option, any hardware configuration chosen needed to meet the appropriate set of minimum requirements, in addition to being able to accommodate the AEAS system's data needs. For the server, two primary options were considered: an embedded solution, and a traditional consumer-range computer. The proposed single board embedded computer under consideration was the Raspberry Pi Model B, which is based on ARM, and the more traditional approach involves either building a low-cost x86 computer, or requiring that users have access to a computer of some minimum specification to be able to implement the AEAS system. Two sample configurations were evaluated for inclusion into the system, and are shown below in Table 12 and Table 13.

Raspberry Pi Model B			
Instruction Set	ARMv6		
Processor	700 MHz ARM1176JZF-S		
Floating Point Unit	Yes		
RAM	512 MB		
Secondary Storage	SD Card (up to 32GB)		
Networking	10/100 Ethernet		
USB ports	2		
		Cost:	\$60

Table 12: Hardware specification for embedded computer

Consumer PC			
Instruction Set	x86		
Processor	AMD or Intel single core (x86) 2 GHz or		

⁴⁹ System requirements for Debian Squeeze operating system. Web: <http://www.debian.org/releases/squeeze/armel/ch03s04.html.en> 23 Nov 2013

⁵⁰ System requirements for Ubuntu operating system from Canonical Ltd. Web: <https://wiki.ubuntu.com/PrecisePangolin/ReleaseNotes/UbuntuDesktop#Installation> Web: 23 Nov 2013

⁵¹ Storage requirements for Ubuntu operating system <https://help.ubuntu.com/community/DiskSpace>

⁵² Windows 7 system requirements from Microsoft. <http://windows.microsoft.com/en-us/windows7/products/system-requirements> Web. 23 Nov 2013.

	higher		
Floating Point Unit	Yes		
RAM	1 GB or more		
Secondary Storage	Hard drive (>= 80GB)		
Networking	10/100 Ethernet		
USB ports	4		
		Approximate Cost:	\$300

Table 13: Hardware specification for PC

In comparing the two possible options (a low cost single-board computer against a low-end consumer PC), it became very clear that both implementations possess a high amount of processing power, possibly much more than is needed for the proposed system. Both machines are also capable of running various Linux distributions with ease; however, there is a large price disparity between the two. Assuming the minimal operating system is selected (as indicated in section 9.3); the choice of hardware became clear: the ARM-based Raspberry Pi.

According to the ARM11 technical reference manual, double-precision floating point add and subtract operations typically execute in eight cycles, while multiply requires nine cycles, and divide requires thirty-three cycles. In this case, execution is defined as the number of cycles required for the resulting data to be available for use in another instruction, thus the maximum wait for a single arithmetic operation is thirty-three cycles⁵³. The processor operates at 700 MHz, giving approximately 700,000,000 cycles per second, thus for computations that use a large number of division operations, the time required for a single division may be approximately

Approximate time to perform FP division =

$$\frac{33 \text{ cycles}}{700,000,000 \text{ cycles/s}} = 4.714 \times 10^{-2} \mu\text{s}$$

assuming that the division occurs atomically (no higher priority operation interrupts execution). When an arithmetic instruction enters execution, a delay may be introduced before another instruction can enter the data path. The delay depends on the type of instruction. For double precision add/subtract operations, the delay is one cycle. Multiply operations, on the other hand, require a delay of two cycles, and division operations need a delay of 29 cycles (which may have an impact on throughput if a very large number of divide operations are encountered). As shown in Table 14 below, the impact of the floating point operations on server performance is very small.

⁵³ ARM1176jzfs Technical Reference Manual, Section 21.11: Execution timing, Page 683
 Web: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0301h/DDI0301H_arm1176jzfs_r0p7_trm.pdf

Arithmetic Operation	Execution period (cycles)	Data path delay (cycles)	Approximate Execution Time
Add/Sub	8	1	$1.143 \times 10^{-2} \mu\text{s}$
Multiply	9	2	$1.286 \times 10^{-2} \mu\text{s}$
Divide	33	29	$4.714 \times 10^{-2} \mu\text{s}$

Table 14: Approximate time required for arithmetic instructions

In considering the performance of the relatively simple ARM processor relative to an x86 implementation, it became very clear that an embedded solution exceeded the required computing power for performing the calculations and running the required programs. In addition, the expected computational overhead for the AEAS software is a small fraction of the ARM processor's power, which also makes the choice well suited to expanded implementations featuring enhanced ballistic models, or multiple users.

It was found that 512MB of RAM and 8GB of secondary storage were sufficient given that the simple configuration will likely be running a minimalist distribution of Linux, and the data sent to/from the attachment should be a small number of bytes, maximum. The code required to make the server front-end web accessible is a Java application, which is easily held in secondary storage along with the Java SE Embedded runtime for ARM (approximately 30MB)⁵⁴. The 700 MHz clock speed of the device also provides for an ample number of clock cycles to handle the required OS, computations, and communications with minimal strain on the hardware.

Another important facet in the hardware consideration was the need for network connectivity. The system must be accessible from the web, so an Ethernet port or Wi-Fi module was required. The Raspberry Pi includes a single RJ-45 port, as well as two USB ports, thus giving users of the system the option of interfacing with the local network using a cable or a wireless connection (if available) through the use of an external USB networking adapter. The Raspberry Pi also contains a set of GPIO (General Purpose I/O) headers which may be configured and used in the event that additional connection options are required for a component. Devices which use a UART, I²C, or SPI interface could be connected to the server through the GPIO pins, which results in compatibility with a wide range of low-level peripherals.

With regard to power consumption, the Raspberry Pi requires a minimal 5V, 700 mA power supply in contrast to the several hundred watt PSUs or expensive batteries required by consumer PCs. The complete server hardware specification is shown below in Table 15, and the device is shown in Figure 33.

Raspberry Pi Model B			
Instruction Set	ARMv6		
Processor	700 MHz ARM1176JZF-S		
GPU	Broadcom VideoCore IV GPU		
Video Output	HDMI		

⁵⁴ <http://www.oracle.com/technetwork/java/embedded/downloads/javase/index.html>

Floating Point Unit	Yes		
RAM	512 MB		
Secondary Storage	SD Card (up to 32GB)	8GB was used.	
Networking	10/100 Ethernet	802.11 Wi-Fi through USB Adapter	
USB 2.0 ports	2		
Power Requirement	5V, 700 mA via MicroUSB	3.3V, 5V Power pins	
Additional Data Communications	UART, I ² C, SPI		
		Cost:	\$60

Table 15: Hardware specification for Raspberry Pi Model B⁵⁵

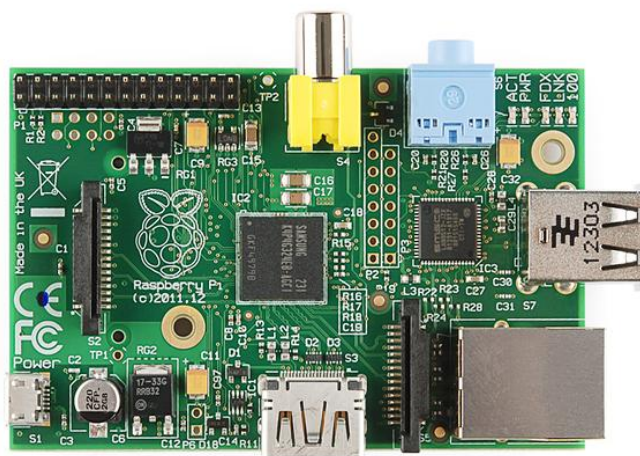


Figure 33: Overhead view of Raspberry Pi Model B server hardware⁵⁶

Should additional computational power be desired, or the system expanded, an alternative x86 system design may be pursued to ensure successful operation.

For the purpose of demonstrating the current weapon attachment and server system, a simple x86 laptop running Microsoft Windows 8.1 was used. The Raspberry Pi was used throughout the development of the server and still works as a viable deployment platform. The laptop solution, however, allows for a more fault-tolerant system which was required due to frequent changes in the area the system was being used in. The Raspberry Pi is recommended for headless operation in a dedicated room, where the positioning beacons and target displays are static.

The system shown below was proven to effectively provide weapon monitoring and integration with required subsystems while maintaining a minimal delay in providing the visualization results.

⁵⁵ Specifications taken from product description. <https://www.sparkfun.com/products/11546>. Reprinted under the Creative Commons License.

⁵⁶ Picture of Raspberry Pi Model B hardware from <https://www.sparkfun.com/products/11546>. Accessed 28 Nov. 2013. Image available under Creative Commons License <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Alienware M11x R1		
Instruction Set	x86	
Processor	1.3 GHz Intel Core 2 Duo SU7300	
GPU	NVIDIA GeForce 335M GT	
Video Output	HDMI/VGA	
Floating Point Unit	Yes	
RAM	8GB	
Secondary Storage	HDD: 250 GB, 7200 RPM SATA II	Less than 1 GB required for AEAS software and required packages
Networking	10/100 Ethernet	802.11n Wi-Fi
USB 2.0 ports	3	
Power Requirement	Battery, or available wall outlet	

Table 16: Demonstration PC specification

6.3. Server Operating System

In order to make the most efficient use of the server hardware, an appropriate operating system was required to develop, test, and run the proposed AEAS software. Now that the server hardware had been selected (see Section 9.2), an operating system which meets the compatibility needs of the Raspberry Pi Model B hardware and the performance criteria necessary to create a responsive firefight simulation had to be found. This section evaluates the software needs of the system, and how well a set of commonly used operating systems meet the required criteria.

A wide variety of operating systems were considered for use in this project, however, the selection of an ARM processor greatly narrowed the list of potential candidates down to a few distributions of Linux. Operating systems such as Microsoft's Windows were initially considered due to their widespread use (and therefore user familiarity), but were eventually ruled out due to a lack of support for the targeted ARMv6 architecture, as well as the high cost of a license. Another possible OS included the Ubuntu Linux distribution, which is available at no cost, and features a simple to use GUI interface, but again the OS did not have the desired hardware compatibility with ARM. To be a viable candidate, the operating system had to have support for the ARMv6 architecture (ARM11 devices), and be able to run on the server hardware defined in Table 13 in Section 9.2.

Another consideration was the distinction between command-line and GUI implementations of the OS. A command-line interface consumes fewer hardware resources (such as RAM and disk space), but is more complicated for users to work with and generally requires a great deal of familiarity with the exact distribution to use effectively. The alternative choice involved choosing a platform with a graphical user interface, which offered a much more familiar window-based approach to

navigation for users, but comes at the cost of performance. One of the goals of the server was to be able to operate as a “headless” unit wherein the system should not require any user interaction after initial setup. The server should not need any user peripherals or a display to output to, and should primarily be interacted with through the web GUI.

The next step was to determine a set of the most suitable operating systems. Given the highly specific choice of server hardware, the number of compatible operating systems was relatively small, yet there still existed a varied set of options due to the ability to choose from many different distributions of Linux. Some options were versions of Linux specifically developed with the Raspberry Pi in mind, while others are simply ARMv6 compatible versions of Linux with the choice of command-line interface or GUI versions. A list of potential choices is shown below in Table 17.

Operating System	Description	System Requirements
Debian 6.0.7	Linux distribution with ARMv6 support.	CLI: RAM: 64MB, HDD: 1 GB GUI: RAM: 128MB, HDD: 5 GB
Arch Linux (ARM)	Linux distribution focused on embedded systems.	Raspberry Pi is a supported device
Raspbian	Debian optimized for the Raspberry Pi	Designed for Raspberry Pi
Pidora	Fedora remixed for the Raspberry Pi	Designed for Raspberry Pi

Table 17: Example Operating Systems

All of the potential solutions were particularly lightweight and consume few computational resources, thus making them ideal for creating a responsive system. The two Linux distributions which are specifically made for the Raspberry Pi (Raspbian and Pidora) were expected to have minimal setup requirements, and a high level of compatibility with the hardware, but the specialization does leave some potential for future software incompatibilities. Debian and Arch Linux, on the other hand, are designed for implementation on a much broader range of devices, so hardware compatibility should not be an issue, though the existing drivers may not make the best use of the hardware. For example, the developers of the Debian operating system suggest that the core version of the OS “will not make best use of the floating point hardware⁵⁷”. Issues such as this are the reason that hardware specific versions of Linux have been developed for the Raspberry Pi, and in this case are recommended over more general distributions of Linux. Given that the Raspbian OS is simply “Debian armhf rebuilt by members of Debian for the RPi’s ARMv6+VFP2 ARM variant,” issues with compatibility of required software packages should be kept to a minimum, and may be an acceptable risk considering the potential increase in floating point performance. For this project, the number of Linux compatible software packages required to be run on the server is very small (no more than three), and thus further minimized the potential for issues to occur.

⁵⁷ Debian Linux operating system support page for Raspberry Pi. <https://wiki.debian.org/RaspberryPi> 30 Oct 2013 Web: 23 Nov 2013

Another consideration relevant to operating system choice was the portability of code written for the platform. For this system, extensibility of the software to support higher numbers of users and more computationally demanding algorithms was highly desired. In order to accomplish this, the AEAS software is able to be implemented on more powerful hardware as necessary to meet performance and user needs. It was therefore important to choose an operating system where the code written for execution can be reused. Given the closely related code bases found in the Debian and Raspbian operating systems, the code written for one should run without issue on the other. Thus, if additional processing power were required, installing Debian on an x86 computer could be accomplished, and the execution of code specific to the AEAS system may occur (assuming that all hardware needs/interfaces are accounted for in the new hardware).

The current build was developed on both Linux and Windows 8.1, and is capable of running on either operating system with approximately equal configuration. The development on Windows occurred primarily to facilitate the need to make rapid changes to the software in the event of a new environment as the system was frequently being moved.

6.4. AEAS Software

In addition to the operating system, the server is also responsible for running software related to the storage of user information, simulation state information, communication, monitoring, and computation for system events.

6.4.1. Data Structures

The AEAS server sends and receives various types of data through communication with the weapon attachment, and users accessing the web GUI. This section covers the types of data being transmitted, their usage, and how the data was stored on the server. A sample UML class diagram modeling the state of the simulation, the users, and the attachments is shown in Figure 34 below. In the end the system was simplified and not all structures shown below were implemented, but the weapon attachment and state information was designed for extension and future integration with the unimplemented information shown which logically extends the current state.

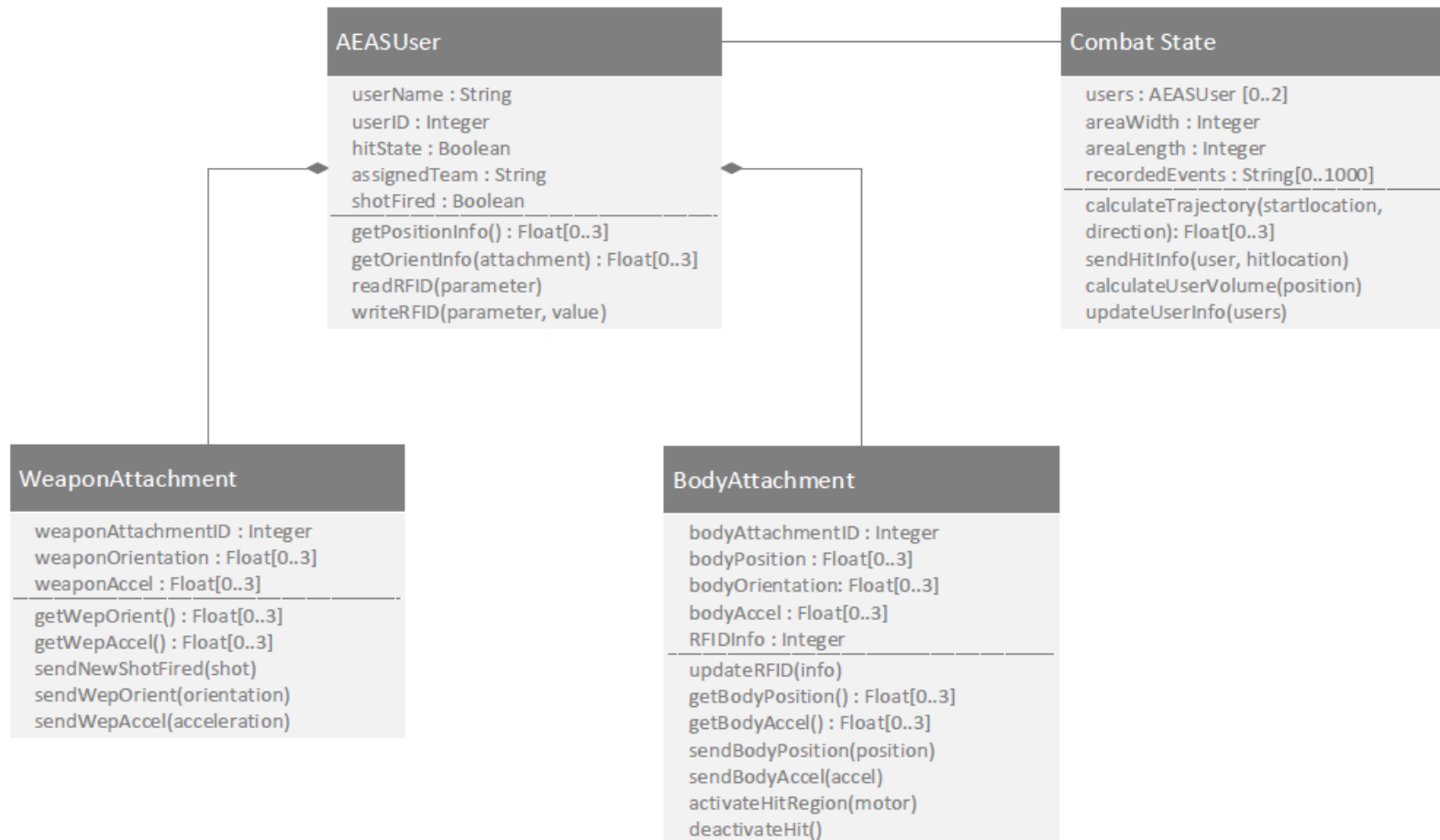


Figure 34: Sample UML class diagram

The weapon attachment data on the server pertains to the orientation and position of the user's weapon measured in a three dimensional space; thus, the two quantities was stored as sets of three floating point values. Orientation was modeled as the change in a set of angles (yaw, pitch, and roll) relative to the standard orientation vector (see weapon attachment section), and resolves to a direction vector.

The server essentially keeps a detailed record of the weapon's position and state information for use in determining the overall state of the simulation. Considering that only one user is under consideration in this system implementation, storage of additional details and information not shown in the class diagram was easily achievable with no impact on system performance.

The data received from the weapon attachment was compiled into a data packet containing the information shown below. This data is the minimum required information to perform the computation of if the user is aiming at the target plane.

{[pitch angle, yaw angle], [positionx, positiony, positionz], trigger state}

6.4.2. Algorithms

The AEAS server implements a large number of algorithms to perform the various tasks appointed to the server, from calculating bullet trajectories to polling the attachments and updating the web GUI's visualization element. The algorithms section describes the basic processes involved in the event of a fired 'shot', a successful 'hit' with a fired 'shot', and the updating of the web GUI after polling each user.

For the AEAS system to accurately simulate the firing of shots, the bullet trajectory calculations performed by the main server had to be physically accurate, and produce results that are within 5 cm of the expected trajectory or impact location when fired from a distance of 5m. Given that the size of the operational area is small, bullet drop due to gravity should be negligible; however, an appropriate model for bullet trajectory using standard firearms was still developed. The calculations were performed using high precision floating point values for user position, weapon orientation, and weapon characteristics (expected mass and velocity of projectile). Consequently, in order to perform such calculations, double-precision floating point operations must be supported by the server hardware, which led to the choice of the Raspberry Pi due to its low cost and ARM processor.

To perform the trajectory calculations, many different approaches were considered. First, the standard kinematic equations taught in physics were considered for use as a ballistic model to determine the bullet trajectories. This model would not have included the effects of mass or form factor of the projectile. Given the small size of the operational area, this simple model could prove adequate in producing reasonably accurate results at a minimal computational cost. There are several issues that may result from using such an algorithm though. For example, the trajectory computation may need to occur in a two dimensional space instead of the desired three dimensional space due to the form of the typical 2D kinematic equations introduced in physics. To perform the calculation, the bullet's acceleration may also need to be assumed to as

zero (velocity is constant over the small operational area). The end result simply models a point to point trajectory analogous to using a laser-based system, which the AEAS system is designed to improve upon. In spite of these issues, the kinematic approach would still have yielded the most practical result. By considering the bullet's trajectory as a straight line vector from the weapon's muzzle to the system's boundary, an accurate flight path was generated for the maximum distance found in the scenario area as shown in Figure 35.

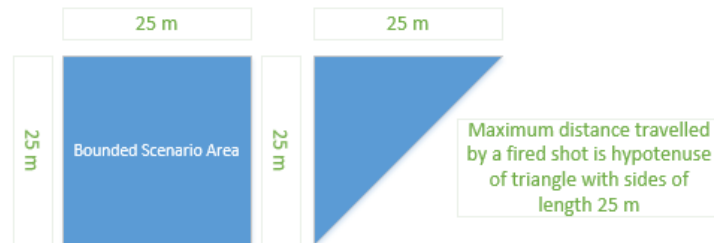


Figure 35: Max Distance of Scenario Area

$$\text{Max distance} = \sqrt{\text{length}^2 * \text{width}^2} \text{ (Pythagorean Theorem)}$$

$$\text{Max distance} = \sqrt{5 \text{ m}^2 * 5 \text{ m}^2} = 35.4 \text{ m}$$

Assuming that the projectile being fired is a 7.62x51mm NATO round with an estimated muzzle velocity of 800 m/s, then the approximate drop in the y-direction of the bullet can be determined by kinematics. First, the time of the bullet's flight is determined over the maximum distance travelled, and is shown below.

$$\Delta t = \frac{\Delta x}{v_x} = \frac{35 \text{ m}}{800 \frac{\text{m}}{\text{s}}} = .0438 \text{ s}$$

Next, the bullet's drop in the y-direction is determined from the equation below.

$$\Delta y = v_y t + \frac{1}{2} * g * t^2 = \frac{9.8 \frac{\text{m}}{\text{s}^2} * .0438 \text{ s}^2}{2} = .0094 \text{ m} = .94 \text{ cm}$$

Now that the approximate drop in the y-direction has been determined, the result is compared against the accuracy requirements. It was suggested that the trajectory calculation must be accurate to within 5 cm, and considering the potential drop due to gravity is less than one-fifth the worst case accuracy at a distance of 10 m farther than the required distance, the kinematic model was actually very viable.

For the kinematic trajectory approach, the result is essentially a point-to-point firing of a bullet due to the small area the system is designed to operate in. The primary computations that occur are the time required for the bullet to exit the playing area and the position of the bullet at a set of discrete time intervals after firing. The server then determines if any target was found to be located along the direction vector over the projectile's flight interval.

A second approach was set out to obtain a set of ordinary differential equations derived from physics research which accurately model the ballistic behavior of a typical small caliber bullet. The desired variables included the initial velocity of the projectile, the angle of the weapon barrel with respect to the coordinate axes, as well as the projectile's mass and response to air resistance as shown in Figure 36. The solution for each equation would then be found or given, and would be used to write code for the server which correctly implements each necessary calculation. In researching various scholarly articles and sources, such information proved to be difficult to find. The computation of a bullet's trajectory in three dimensional spaces was realized to be a very challenging result to calculate due to the sheer number of variables involved. Instead, it became apparent that ballistics calculators were often the primary source of such trajectory information. Typical ballistics calculators generate a table of values or curve indicating how far the bullet has dropped relative to its starting position as it travels. The determination of bullet drop is derived from data tabulated by the bullet's manufacturer, and thus varies from one type of ammunition to another.

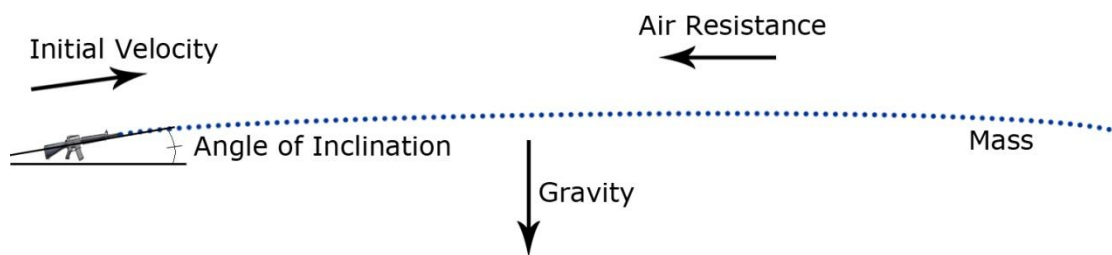


Figure 36: Ballistics Behavior Model

Finally, it appeared that the use of a ballistics modeling program was also a potential option, thus allowing for manufacturer determined characteristics to be used in the determination of an approximate trajectory. If a small number of bullet types were to be used, then information gathered from ballistic calculators provided by munitions and firearm manufacturers can be stored in tables, and the resulting trajectory information used to produce a result within the combat area. The ballistics calculation result would still need to be modified to move the result into the coordinate area of the simulation, and analyzed for any potential 'hits' on targets. One such program that satisfies this is the modeling results is the GNU Exterior Ballistics Computer⁵⁸ and is distributed under the GPLv2 license. Characteristics of the bullet including mass, typical velocity when fired from a weapon and the ballistic coefficient for overcoming air resistance are expected to be known by the calculator. Additional information related to the weapon and bullet type in use would be required depending on the complexity of the model chosen or developed. The ballistics calculator, however, would provide an adequate starting point for determining the behavior of the bullet within our defined playing area. If the kinematic result had been found to be inadequate for the current system, then this solution would have been studied further.

⁵⁸ <http://sourceforge.net/projects/ballisticslib/?source=recommended>

Thankfully, we did not have to go this route. In any case, this model may be viewed as a possible extension of the system should any future versions be developed.

For the demonstrated version of the AEAS system, all simulations took place in very small rooms (typically around 5 meters in length), and thus trajectories were often approximated by nearly instantaneous, point-to-point paths.

Using the information received from the weapon attachment, the current position of the weapon, as well as the orientation angles, are used to determine whether or not a fired shot will intersect with a defined target plane. By specifying the approximate size and location of the projected screen, an accurate target can be created after some calibration and precise measurements. The trajectory computation is performed on each receipt of new data, and draws a target “reticle” to show approximately where the simulated bullet would intersect if it falls within the target area. If the user is pointing the weapon away from the display, then nothing will be drawn to the target display canvas.

For demonstration purposes, it was found that the MPU6050 tended to orient itself about the positive x axis, so by initializing the weapon while it faced the display and assigning the target display on the Y-Z plane, computation of intersections becomes possible. In reality, the location of the display depends upon the positioning of the beacons, so all configurations were designed with this set up in mind.

The weapon direction vector is actually computed using only two of the angles produced by the IMU. By using the generated yaw and pitch angles, the direction vector is computed as follows.

$$\begin{aligned}x &= \cos(\textit{yaw}) * \cos(\textit{pitch}) \\y &= \sin(\textit{yaw}) * \cos(\textit{pitch}) \\z &= \sin(-\textit{pitch})\end{aligned}$$

The negative sign of pitch in the Z computation is simply to correct how the angle is defined by the IMU, such as clockwise or counter-clockwise.

Next, the position and direction vector are used to determine if there exists a point on the target (also known) which falls along the straight-line trajectory.

First, a direction vector which is normal to the target is computed using three known points on the plane. The points can be determined in many ways; however, in this case the length and width of the target screen are simply used to generate three points.

$$\textit{point A} = [\textit{target x length}, 0, 0]$$

$$\textit{point B} = [\textit{target x length}, \textit{target y length}, 0]$$

$$\textit{point C} = [\textit{target x length}, \frac{\textit{target y length}}{2}, \textit{z length}]$$

Now, it should be noted that the coordinate system matters greatly in the determination of the points. In our system, only positive values of x, y, and z are considered. The top of the center beacon serves as the origin of the system, with x and y increasing toward the other two beacons, and z increasing downward (toward the

floor). The use of this coordinate system greatly simplifies the translation from a real-world position relative to the beacons to pixel coordinates which are displayed on the visualization.

Next, two vectors are generated from the three points A, B, and C. By subtracting the corresponding components of the points A and B as well as A and C, two new vectors are produced: one from B to A and one from C to A. By taking the cross product of these two vectors, a new vector (which is perpendicular or normal to the target) is generated.

Using this new vector, an equation can be written which defines the target plane in the form shown below.

$$N_x(x - a_x) + N_y(y - a_y) + N_z(z - a_z) = 0$$

In this case “a” simply represents the point A on the plane.

By using the line between the weapon’s location (a point) and the plane, we can parameterize x, y, and z to allow us to solve for the point at which the line intersections (if it does at all).

$$\begin{aligned} x &= \textit{weapon position}_x + \textit{weapon direction}_x * t \\ y &= \textit{weapon position}_y + \textit{weapon direction}_y * t \\ z &= \textit{weapon position}_z + \textit{weapon direction}_z * t \end{aligned}$$

The parameterized forms of x, y, and z can then be substituted into the equation of the target plane. Thus, the equation can now be solved since there is only one unknown, which is denoted by t. By manipulating the substituted equation, t is computed as indicated below.

$$\begin{aligned} t &= -N_x * \textit{weapon position}_x - N_y * \textit{weapon position}_y - N_z \\ &\quad * \textit{weapon position}_z + N_x * a_x + N_y * a_y + N_z * a_z \\ t &= t / (N_x * \textit{weapon direction}_x + N_y * \textit{weapon direction}_y + N_z \\ &\quad * \textit{weapon direction}_z) \end{aligned}$$

Finally, the point where the trajectory and the target intersect is found by simply solving for x, y, and z with the newly determined value of t. If the point does not fall within the target, then no solution exists, and no target information is conveyed to the user.

6.4.2.1. Networking

In order to make the server front-end web accessible, the NodeJS framework’s networking libraries was used to perform TCP/IP communications. Using the Socket.io package for NodeJS, a WebSocket can be easily configured which sends the most recent data received from the weapon attachment over a port on the local network from the server. A WebSocket provides TCP communications very quickly, introducing no significant delays, and ensuring reliable transmission of the data.

NodeJS is developed with scalability and networking as a primary focus, so the implementation easily allows for many connections to be supported if desired. The current build does not deliver the client over HTTP, thus the user must simply copy the AEAS files over to their computer (all of which take up less than 1 MB), and view the web page in a browser which is connected to the local network. The navigation to the webpage creates the socket connection, and starts sending data as long as the weapon attachment is active. The weapon attachment can also be turned on and off at will as long as the NodeJS server code remains active. Once the weapon attachment becomes active again, data will simply start updating to connected clients.

NodeJS does offer HTTP functionality through the Express module, but was not implemented in the demonstration code due to concerns over network privileges at the demonstration location.

6.4.3. Visualization

The current state of the firefight simulation is provided to trainers and other personnel by navigation to the web accessible GUI. The main server acts as a web server by providing access to the weapon attachment data, which is used to produce an overhead “map” displaying the training simulation in progress, in addition to the current target. The visualization was made using the HTML5 Canvas API due to its simplicity, and standards-compliant implementation on virtually all modern web browsers.

The visualization and other AEAS software are written as a client application, which must be present on the computer of interest to view the attachment data.

The main server receives data from the weapon attachment at regular intervals to obtain up-to-date user information which was used to display the location and orientation of each weapon on the visualization of the current training area.

The map displaying user locations was updated at least eight times per second to ensure that accurate movement and orientation information is available to trainers. The user in the training area is portrayed by a standard icon which shows the direction the user is facing. The visualization also shows when any user has fired a ‘shot’, by extending their trajectory line to the outermost boundary of the playing field (to show the bullet’s approximate flight path). The direction vector which is computed for the trajectory computation is used to generate the weapon direction line and approximate shot trajectory when a shot is fired. This solution allows for the direction vector components to be used by simply multiplying by desired pixel lengths, and adding the results to the user’s current position. Consequently, two points (the weapon position and a point along the trajectory) are connected via a line to show an approximate overhead flight path.

In order to show the current aimed target, as well as the weapon’s position, the location of the shot on the target and the relative position must first be translated to the two dimensional coordinate system of the Canvas to which the visualizations are drawn.

If the weapon is currently being aimed at the target display, then the resulting intersection point is translated to pixel coordinates through the following method.

$$x = \frac{intersection_y}{total\ length_y} * pixels_x$$
$$y = \frac{intersection_z}{total\ length_z} * pixels_y$$

As noted previously, the target was located on the Y-Z plane for the demonstration, which is why only the Y and Z components of the intersection point are used in the translation. The location of the intersection is scaled to the total size of the operating area, and finally scaled back to pixel coordinates which are simply retrieved from the web browser. The pixel coordinates are full screen, and consider only the usable space available to the web browser. A sample figure is shown below, showing approximately where on the target display the weapon's barrel is being pointed.

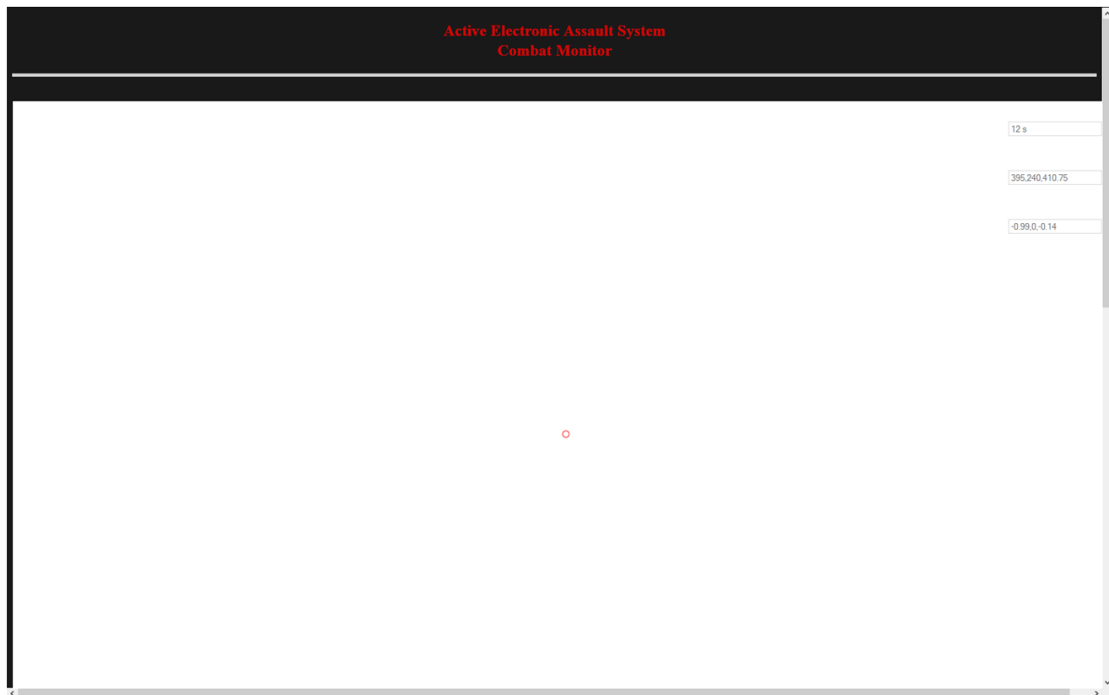


Figure 37: Target View Example

The small red circle in the center of the display is the approximate location of the intersection between the weapon's trajectory and the defined target display. The orientation updates rapidly, so moving the weapon around to target a different location on the display happens with minimal delay. The accuracy of the approximation depends heavily upon the system configuration which is briefly discussed in Section 9 of this document.

Firing information is made available to the server by the weapon attachment, and 'hit' information is determined by the server through trajectory calculations. A hit is shown when the user pulls the trigger and is aiming at the valid target area. If the weapon's

current trajectory does not fall within the designated target and the trigger is pulled, then no state change occurs on the targeting projection. If the user were to pull the trigger given the orientation and position shown above, then a new image would be drawn over the red target circle to indicate the area in which the intersection would have occurred. In a sense, the visualization basically displays a temporary “bullet hole” which has a larger size than the normal targeting circle to indicate the approximate area where a hit would have occurred.

In order to accurately locate the user on the scaled display, a weighted moving average of the last three positions is used to prevent erratic movements introduced by inaccuracy in the positioning system. The position which is displayed through the visualization is determined as follows.

$$\begin{aligned} \text{weightedPosition}_x & \\ &= \text{position}_x * .6 + \text{past position1}_x * .3 + \text{past position2}_x * .1 \end{aligned}$$

$$\begin{aligned} \text{weightedPosition}_y & \\ &= \text{position}_y * .6 + \text{past position1}_y * .3 + \text{past position2}_y * .1 \end{aligned}$$

$$\begin{aligned} \text{weightedPosition}_z & \\ &= \text{position}_z * .6 + \text{past position1}_z * .3 + \text{past position2}_z * .1 \end{aligned}$$

By performing this averaging, the visualization is both more visually appealing (movements are more smooth and natural), while also being more accurate. Only the most recent three positions are taken into consideration simply as a result of the frequency of position updates from the weapon attachment. In a second, the weapon can change in position greatly, thus taking the average of positions over a smaller interval, and updating as frequently as possible gives a better approximation of where the weapon is currently located. If a weapon is removed from the bounded area, then the last position will be displayed until the weapon re-enters the positioning region.

A sample of the weapon positioning view is shown in the figure below.

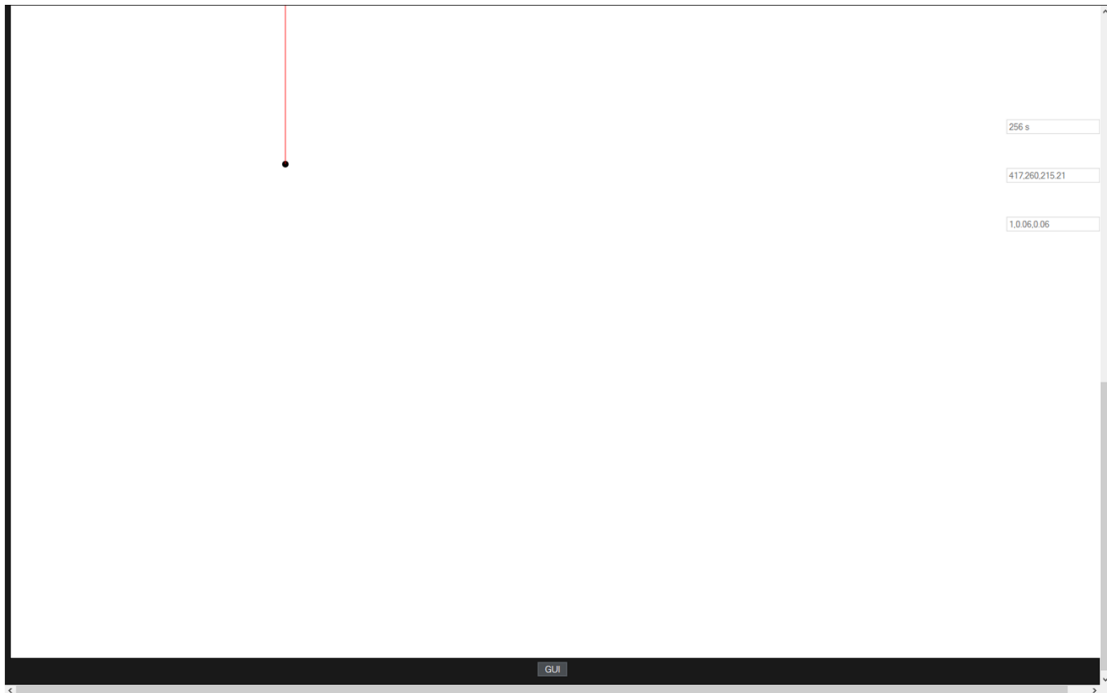


Figure 38: Positioning View Example

The location of the weapon is designated by the black circle, while the current trajectory (direction of the barrel) is indicated by the red line. The position shown is relative to the positioning beacons, and must be configured for the specific properties of the room, such as how the beacons are set up, and where the target display is located.

The current averaged position is also converted from the relative position coordinates to a location on the X-Y plane to give the top-down perspective. The translation to pixel coordinates is very simple, and is shown below.

$$x = \frac{\text{weapon position}_x}{\text{total field length}_x} * \text{pixels}_x$$

$$y = \frac{\text{weapon position}_y}{\text{total field length}_y} * \text{pixels}_y$$

The resulting (x, y) point can then be simply drawn to the Canvas. By keeping the coordinate system we previously defined, we greatly simplify the computation needed to go from a relative position to an on-screen position. This consideration was made to ensure that the system can be made as responsive as possible. More advanced visuals could also be achieved (such as 3D projections), but may introduce a small amount of latency, and were outside the scope of the current project implementation. Future iterations may find use of WebGL as an appropriate graphics API as standards improve on the current web browsers. Assuming that hardware acceleration could be utilized, then the change may further reduce the system latency by offloading some computation to the server's GPU.

6.5. Testing

Testing of the AEAS first took place for each component individually by supplying each device with appropriate data, and analyzing the resulting output from each component. Tests were designed to ensure the optimal and accurate function of each system component through careful analysis of system boundary conditions (such as maximum data rates, positions that are outside of the operational area and environmental factors such as objects within the simulation area which affect measurements).

6.5.1. Weapon Orientation Test Data Injection/Procedure

The weapon attachment measures the orientation and acceleration of the attached firearm to determine where the barrel is pointed in the three dimensional space. Initial testing of the attachment focused on the accuracy of the data obtained from the inertial measurement unit by placing the attachment in different positions and orientations, and then measuring the resulting output. The test data injected was a series of movements relative to a predefined starting position designed to measure the response of the device. First, changes in orientation were measured by moving the attachment about a fixed point in space. The attachment was in a stationary position, and then rotated about the position to produce a new direction vector

$$R_{xyz} = \frac{R_{acc} + R_{gyro} * w}{1 + w}$$

as defined in the weapon attachment section. The resulting direction vector was then compared to the vector physically measured by assuming that the acceleration vector had become zero (the weapon is no longer moving). Additional testing to determine the angle of inclination and orientation during motion also occurred as outlined in the weapon attachment section.

6.5.2. User Position Testing Procedure

User position was determined by the positioning system outlined in Section 4. The user position accuracy was tested by placing a weapon attachment throughout the playing area and measuring the reported positions. The position of the attachment in the testing area was physically measured and compared against the system's result to determine the accuracy of the system. The weapon attachment also needed to be tested for position while in motion, thus necessitating that a sample set of positions be recorded and analyzed for conformance to where the user actually was located in the playing area over the measurement interval. Additionally, tests were performed to ensure that the system could determine when a user has left the scenario area. If a user's body attachment fails to report a signal after a set interval of time, then it is assumed that the user is out of range.

6.5.3. Trajectory Determination Test Procedure

To determine the accuracy of the trajectory calculations, position and orientation data for a weapon was produced and provided to the server. The server then assumes that a 'shot' has been fired from the specified coordinates, and that the simulated projectile will proceed along the direction vector of the barrel. User positions are generated and the server determines if any targets' 'hit' volumes fall along the specified direction vector during the interval over which the simulated projectile is travelling. The tests are then reported back the resulting point at which the projectile exits the playing area, as well as if any simulated targets were 'hit' by the 'shot'. Test data was accurately generated through the use of a simple program written in Java to randomly generate firing positions and user positions both inside and outside of the playing field, and calculate the required trajectories using kinematic calculations. Any computation to be performed on data that lies outside of the standardized area was immediately discarded since the system's focus is on responsive, accurate results within a defined area.

6.5.4. Performance Analysis

Additional testing was performed to verify the responsiveness of the system through the measurement of the time required for positions and orientations to be acquired, and trajectories to be calculated. The time required for each operation to complete had to be minimized to ensure that the system achieves the necessary timing requirements. The time required to poll the weapon attachment for orientation and consolidate the data was measured by having it communicate with a test XBee, where the receiving XBee begins a timer at the start of the test, and measures the amount of time required to compute, send, and receive all orientation data. The same process was repeated for the body attachment positioning systems. The time required for 'hit' notifications to occur will also be measured by randomly selecting a body attachment motor to turn on, and determining the interval required for a user to observe the vibrating motor. If the sum of these measurements results in an interval greater than one second, the performance of the system may need to be reevaluated, or optimizations must occur to the software being run by the components.

7. Integrated Testing

Upon the completion of the server and weapon attachment, testing of the completed system was able to occur. At this stage, all outlined functionality was extensively tested for accuracy and responsiveness to ensure that the goals of the system have been appropriately achieved.

7.1. Performance Analysis

The performance determination for the complete system is a function of the time required for the various system operations to occur. The time required to poll the user positions, obtain the weapon orientations, report a 'shot' fired, calculate the trajectory, determine if any targets have been 'hit', and report a 'hit' will be the longest possible sequence of system events and thus have the greatest impact on the usability of the system. The time to perform each of these operations individually was measured and used to determine if new approaches or optimizations must occur since a long response time will result in confusion of players (since 'hits'/'shots' are being appropriately conveyed to the GUI).

7.2. System Accuracy

The system's accuracy was examined through the careful determination of user positions, weapon orientations, and bullet trajectories in a three dimensional space. In some cases, the positions and orientations was physically measured and compared with the sensor results to determine how far off the system is from the actual results on the playing field. If the results did not meet the requirements outlined in the system specifications, then measures would have been taken to improve the accuracy of the results or the rate at which data is made available to the components. Data needed to be gathered while the attachments are stationary, in motion, inside the bounded area, and outside the bounded area to address the possible issues with system accuracy.

These components are essential to the complete system. Using variations of the tests outlined in previous sections, the entire system as a whole was tested and results recorded. These results directly affected the possibility of this system actually being used in future military training scenarios.

8. Using the System

This section describes how to use the AEAS system in its present state. It presents the necessary prior set up procedures as well as how to initialize the weapon attachment and server. It also presents how to calibrate the system to a target area, presumably provided using a projection or monitor.

8.1. Beacon Setup

Each of the beacons must be placed at corners of the user area. There are three separate beacon poles. The one with two transmitters must always be placed in between the others since the top beacon on this mount is used as the reference point of the coordinate system (0,0). Note that the Weapon Attachment assumes a coordinate system of 5m x 5m x 3m dimensions. Therefore, the outputs of the attachment itself are only exact when the beacons are placed at those distances. When only a larger or smaller location is available, the server will need to transform the coordinates received from the virtual weapon into the actual coordinate system.

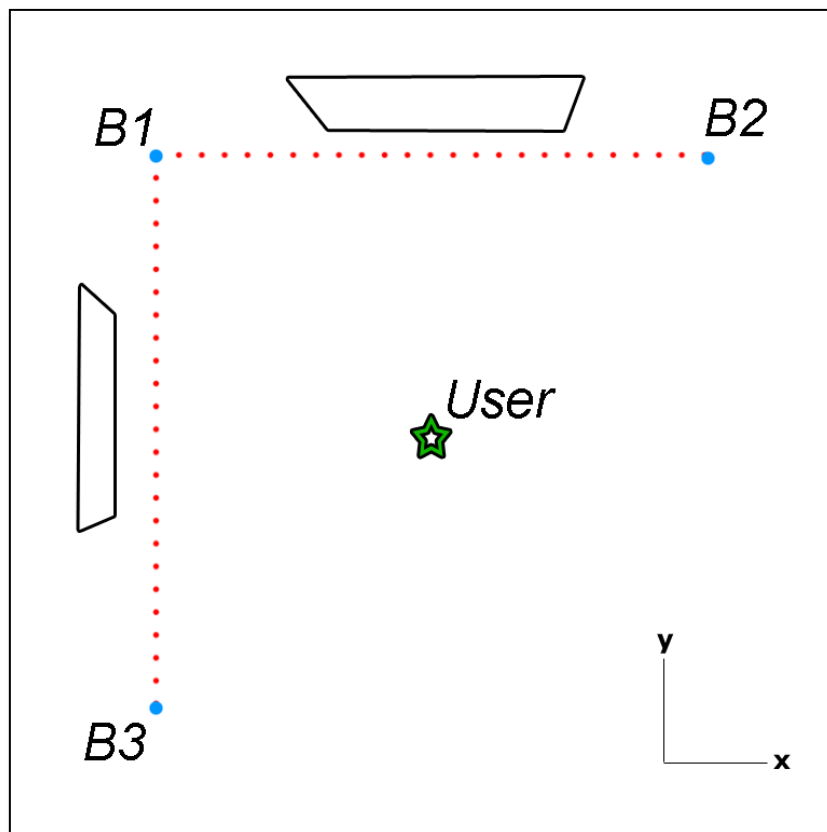


Figure 39: Beacon Placement

When setting up the beacons, effort should be taken to align them as closely as possible as shown in Figure 39. The more exact the right angle between the beacons,

the more accurate the system. Accuracy is also improved by greater distance between the beacons since this allows the microcontroller greater granularity in its time of arrival calculations.

8.2. Weapon Module Attachment

The weapon module itself is very easy to use. It can be attached to any weapon with a maximum diameter of 6 inches. Before fastening, insert a well-charged 9v battery into the battery holder along the side. This can be removed and exchanged while the module is attached, but it is easier to perform while the module is unfastened.

To attach the module:

1. Place along the side of the weapon with the cable tie facing the weapon.
2. Wrap the long end of the cable tie around the weapon.
3. Insert the end of the tie into the receiving slit on the other end.
4. Pull the tie through until module is moderately fastened.
 - a. Note: Don't pull it really tight just yet.
5. Adjust the position of the module such that it is unobtrusive to the trigger and is level with the firearm.
6. Pull the tie through again until the module is securely fastened.

To turn the module on, simply flip the switch on the backside of the battery holder.

8.3. Server Operation

Before the server can begin operation, some installation of software packages is first required. These packages are required to perform the networking functionality, as well as to interface with the communications subsystem and timing microcontroller (both of which are connected to the server via USB). Once set up, the server can be operated by simply entering a command into the NodeJS command prompt. It is assumed that a compatible hardware and operating system are already available.

The first step is to install the appropriate version of NodeJS for your server's operating system from the official link below.

<http://nodejs.org/download/>

For the two server systems which were used in the development, installation was performed on both Raspbian (Debian Linux) and Microsoft Windows 8.1. Installation packages for both of these operating systems are available at the link above, and require no additional configuration.

Once NodeJS has been correctly installed, commands can be run from the preferred command-line utility. The two commands of interest are “node” and “npm.”

In order to run the AEAS server code, a few more packages are required for NodeJS. Descriptions of the packages can be found at the following web pages.

<https://www.npmjs.org/package/serialport>

<http://socket.io/>

<http://expressjs.com/>

Each of these packages can be installed by simply opening the command prompt and typing in the commands shown below. Each is in the form “npm install *modulename*.”

```
npm install serialport
```

```
npm install socket.io
```

```
npm install express
```

Installing the additional NodeJS components requires an internet connection, and may take some time depending on network conditions, and the selected server hardware. Additional installation instructions can also be found at the web sites previously noted.

Once NodeJS has been set up, a new folder called “node_modules” will have been created. This is the location where the AEAS software is to be placed. The location of the folder depends on the operating system, and it is assumed that write privileges are available for the server so that all installations can occur successfully.

The AEASserver.js file will need to be modified to include the appropriate COM port addresses for the Xbee and the timing microcontroller. The addressing of the COM ports also varies with operating system. Under Linux, the address will be something along the lines of

```
/dev/tty...
```

In Windows, the devices can simply be located in the Device Manager, the COM port found, and the parameter “COMX” entered. In this case X simply represents the number of the COM port over which the devices are operating. In the case of our system, the Xbee was located on COM3, and the timing device was located on COM4. The existing code already configures the devices for the appropriate data rate. It is also possible that the system enumerates the devices in this fashion, and thus no additional changes may need to be made.

It is also important to make sure that the local network does not contain any restrictions for communication over port 8000. The AEAS visualization software will need to either be viewed from the server, or another computer which can access port 8000 of the server. Statically allocated IPs are recommended for the server, that way the only configuration that needs to occur will be in the index.html file, where the socket initialization code is. If there are restrictions over the network the server is expected to operate on, please work with the network administrator to determine an

appropriate IP address and TCP port for the server to make the weapon attachment data available through.

Now that everything is configured, the server code can be executed by first navigating to the “node_modules” directory via the command line, and then simply entering the command shown below.

```
node AEASserver.js
```

Leave the command prompt open to have an easy way to determine if any errors have occurred. Once the code has been run, the weapon attachment can be initialized at any time, and the output viewed through the AEAS web page. Once the connection has been established to the server, the Canvas elements which allow for viewing of position and orientation will become full screen. This indicates that the server is running. The initial web page is shown below.

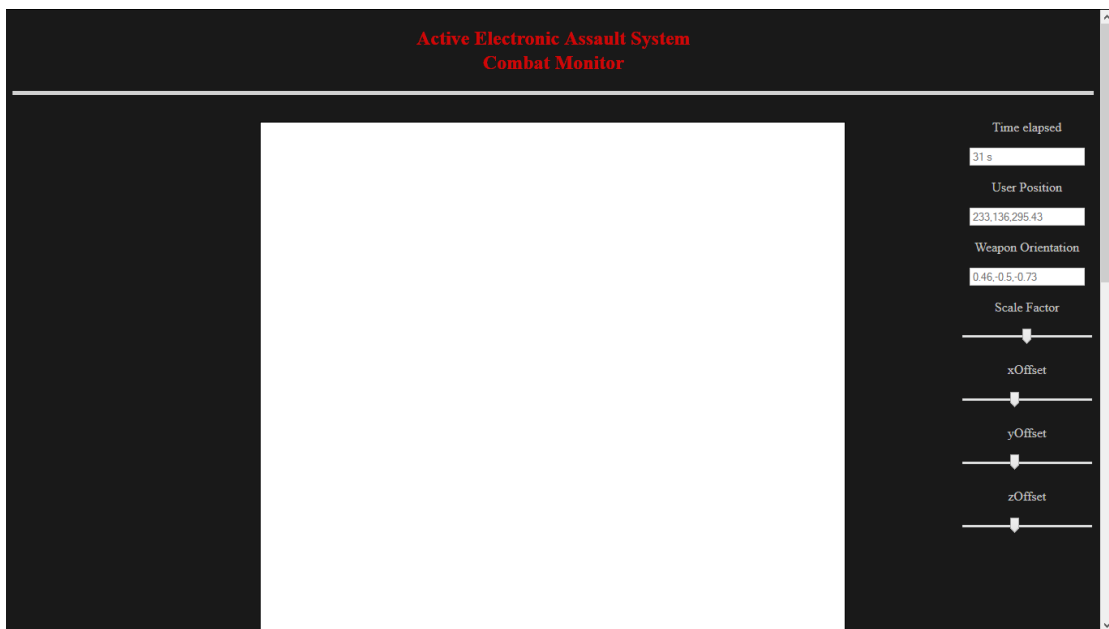


Figure 40: Initial AEAS Web GUI (connection not yet established)

After communication has been established with the server, the weapon attachment can be turned on and off at will. Whenever the attachment is sending data to the server, the result will be displayed in the web GUI. Once data transmission stops, the last state will remain on each display. For the best experience, set the web browser to full screen mode so that the entirety of the display can be used as a simulated target.

8.4. System Calibration

In the figure shown above, it should be noted that there are three fields and four “sliders” on the right side of the web page. This view is for the configuration and verification of the system set up, and can be viewed by toggling the GUI button at the bottom of the web page. The three fields give the current position and weapon orientation data, in addition to a counter to measure the length of time that the

attachment has been operating for. The output shown in these three fields can be used to verify that the system has been configured to a desired level of accuracy.

Configuration of the system takes place by adjusting the sliders. These sliders change parameter values which are important in creating the accuracy required by the system. The description of each slider's function is shown below.

Variable Name	Value Range	Definition
Scale Factor	0.0 to 1.0	The scale factor allows the size of the area in x, y, and z to be configured relative to the maximum dimensions the system supports (490, 490, 360 cm)
xOffset	0 to 250 cm	The xOffset is used to configure the distance that the target begins at begins relative to the origin of the coordinate system in the x direction.
yOffset	0 to 250 cm	The yOffset is used to configure the distance that the target begins at begins relative to the origin of the coordinate system in the y direction.
zOffset	0 to 250 cm	The zOffset is used to configure the distance that the target begins at begins relative to the origin of the coordinate system in the z direction.

Table 18: Configuration Parameters

The configuration depends greatly upon how well spaced the positioning beacons are, in addition to the pixel resolution and physical size of the display. By experimenting with different values for scale, and offset, near pixel-accurate targeting can be achieved. The computation of the correct values of these parameters is not trivial. It is thus advisable to try a few different configurations and determine how each variable impacts the current set up. Over time, a configuration can be achieved where aiming down the weapon's sight will give an accurate dot representation on the screen. In testing, this process took between one and two hours using a standard 42 inch LCD TV. Best results will be achieved purely through experimentation, and user perception of the current accuracy and responsiveness of the set up.

The system is not designed to be frequently moved to new locations as a consequence of this configuration requirement. It is advised that the system be placed in an area where it can remain on, and in place, for the best experience to be achieved.

9. Expandability

The AEAS system has been designed around the idea of future expansions and improvements as noted throughout the document. The system is presently designed for only one player, but is expandable by simply altering software parameters and adding additional attachment hardware up to a certain number of users. Further expansion is also possible through the use of more sophisticated positioning hardware, server hardware improvements, and code optimizations that can be made as necessary.

Certain advances in technology will be particularly applicable to providing increased fidelity for the system. Inertial Measurement Units could be upgraded with relative ease when new technology surfaces that provides increased measurement accuracy. Obtaining accurate positioning of weapons is a difficult problem, and advances in GPS or other systems could provide increased precision which would result in more accurate bullet trajectories and therefore increased ability to train soldiers effectively. The greatest aspect about the expandability of the system is that advances in technology are always taking place, which means the system can only possibly get better over time.

9.1. Operational Limitations

The system should only be operated under the situations and configurations described in this document. Known limitations require that the playing area be completely clear of obstructions due to the sensitivity of the ultrasonic positioning system. In addition, this system implementation for the current project is designed for a maximum of one user, and thus will not be configurable for larger numbers of players.

9.2. User Identification

Knowing your best shooter can be an important piece of knowledge when on the battle field. To accommodate this need, the AEAS system is a training simulation with the ability to record and track statistics on different users. Each user will firstly create an account using the server. Every time the trigger sensor is asserted, the data was paired with a user ID, along with position and orientation data. The server will collect this data through the RF communication, calculate if it was a 'hit' using the algorithms and the position data from the body attachments, and save it all in the user's record. Therefore, in order for the AEAS system to record, the weapon attachment must input a user ID to match with an account.

9.2.1. Description

The user identification system was part of the weapon attachment, and directly connected to the microprocessor. It needs to send and receive serial data, be lightweight, be small, be low power, and not take a lot of processing power. It also should be easy and quick to use when prepping for and during simulation.

9.2.2. Types of User Identification

Three options explored to input the user identification (ID) to the microcontroller for user statistics were a keypad for the user to input an ID code, a barcode scanner, and a Radio Frequency Identification (RFID) system.

A keypad is the simplest method for user identification. A standard 0-9 input keypad is readily available, and easy to interface with. Unfortunately, they are too large and too heavy to fit into our size and weight specifications of the weapon attachments.

Barcodes now come in 2-Dimensional and 3-Dimensional forms. Barcode scanners can be implemented from a small camera, keeping the size and weight added to the weapon attachment minimal and well within the specifications. Once scanned, the bar code must be processed into an ID, requiring more processing power than the selected microcontroller can provide. Using this ID method would over-complicate the AEAS system, and is therefore disregarded as an option.

RFID systems are split into three main categories, including capacitive, inductive, and backscatter coupling. In capacitive coupling RFID, the reader's AC signal is collected by the capacitance between the electrodes of the card and reader and rectified by modulating the load. As a result, capacitive coupling RFID is only useful in a close range of less than 1cm. Smart Cards that are inserted into a reader are an example of capacitive coupling RFID. In inductive coupling RFID, voltage is induced by mutual inductance between the reader's and the card's induction coil antennas to power the chip. When powered it changed the load on the coil antenna to rectify the query signal and return its ID. The induction coils can only induce the supply voltage in a remote range of 1cm to 1m. Inductive coupling RFID can be found in Near Field Communication (NFC) and Low-Frequency (LF) RFID systems. In backscattering coupling RFID, the reader's RF power transmitter signal powers the tag to switch the resistor on the antenna to reflect back a modulated signal containing the ID. Depending on the antenna, shape, and size of the tag, the reader can ID the card for a long range of over a meter.

It was more user-friendly, less time consuming, and more lightweight to use a RFID system to input the identity of the user.

9.2.3. Type of RFID system

The cards used in capacitive coupling RFID are too big to be sticking out of the weapon attachment and the long range of backscatter coupling RFID could result in interfering user IDs from nearby users; therefore, an inductive coupling RFID system wasn't suited the AEAS system. An inductive coupling RFID system consists of an RF module and RFID tags. RFID readers now come in many shapes, sizes, frequency, ranges, and power.

The AEAS system will implement NXP Semiconductor's Mifare RC522 (MFRC522), a popular RDID reader compatible with the ATmega328. Mifare readers perform the complex control algorithms necessary within the RF unit, eliminating additional work for the microcontroller. The MFRC522 is low-power, low-cost, small, and easy-to-use. It can read a card's ID within a range of 10cm, meeting the requirement for quick use.

9.2.4. Design

The design of the AEAS system's RFID system includes the digital and hardware design of the module, along with choosing the ID card for it to communicate with. The digital design of the MFRC522 consists mostly of a choice of interface protocol and the resulting interface configurations. The hardware design consists of power supply design, antenna design, and design of the interface with the microcontroller.

9.2.4.1. RFID Module

The design of the RFID module consists of the digital and physical implementation of the MFRC522 into the weapon attachment.

9.2.4.2. Digital Interface Configuration and Use

The MFRC522 supports three different simple and popular serial interfaces to communicate with the microprocessor at the heart of the weapon attachment.

Serial Peripheral Interface (SPI) is a simple high-speed (up to 10 Mbit/sec) protocol designed for multiple slaves and one master on a PCB. It uses only four connections to the microcontroller, including SCK for self-clocking the communication, MOSI for master out slave in data transmission, MISO for master in slave out data transmission, and SS for slave select when implementing more than one slave. The commands, along with the data and address formats, are fool-proof. Due to the self-clocking attribute to SPI, there is no need for external hardware to connect and communicate with the microcontroller.

UART is a slower (7.2 - 1228.8 Kbit/sec) protocol often used for communication

between parts on different PCBs. UART is only used for peer-to-peer communication, leaving no need for slave configuration or addressing. The clocks on the devices connected through UART must match since does not work with a clock signal. Often an external oscillator is necessary to ensure the clocks are synchronized. The UART also uses only four connections to the microcontroller, including RX for data transmission from the microcontroller to the RFID module, TX for data transmission from the RFID module to the microcontroller, DTRQ for flow control, and MX; although DTRQ and MX are not necessary for the AEAS system's application.

Inter-Integrated Circuit (I²C or I2C) can transfer data at a standard rate of less than 100 KBaud, a fast rate of less than 100 Kbaud, and a high-speed rate of less than 3.4 Mbaud. I2C is intended for communication between a master and more than one slave. The addresses are complicated with concatenated slave and register addresses. This interface protocol uses only two connections to the microcontroller, including SDA for the serial data transmission to and from the microcontroller and SCL for the clock to synchronize the data transfer.

The AEAS system will implement the SPI protocol, due to the unnecessary complications of using UART and I2C protocols, for the serial interface between the RFID system and the microcontroller. There are no configuration settings for using SPI protocol. In addition, The MFRC522 implements Automatic Microcontroller Interface Detection (AMID) by resetting its interface and checking the current host interface type, by sensing the logic levels on the control pins after the reset phase, automatically after performing a power-on or hard reset.⁶¹ AMID is set up using a combination of fixed pin connections.

9.2.4.3. Contactless Analog Interface

The MFRC522 uses a separate contactless serial interface to communicate to the ID card. UART is the only interface protocol option for this communication. The transfer speed ranges from 106 kBaud to 848 kBaud. The format for 106 kBaud communication is simpler than the format used for the higher transfer speeds, while still meeting all the needs and specifications of the RFID system. The UART for the contactless interface to the ID card can be configured for bit or byte oriented framing, parity or no parity, cyclic redundancy check (CRC) or no CRC, etc. The reader will again use AMID to match the UART protocol requirements to those of the contactless RF smart card it is currently communicating with. CRC is error checking code handled within the digital logic of the MFRC522 reader, along with code to manage framing the data. The contactless interface with the card implements a 64-Byte First-In First-Out (FIFO) architecture to manage the streams of data coming in and out of the reader. In addition, interrupts to implement efficient host software are included.

⁶¹ NXP Semiconductors Inc., "MFRC522 Product Data Sheet Rev. 3.6", 14 Dec. 2011, http://www.nxp.com/documents/data_sheet/MFRC522.pdf, Nov. 2013.

9.2.4.4. Hardware Implementation

The AEAS system will use a Mifare RFID Module to implement the RC522 reader, using a SPI digital interface compatible with the microcontroller. The RF module incorporates the antenna topology 1 reference design⁶² from The Antenna Design Guide for MFRC52x, PN51x and PN53x (document number AN1445) application note released by NXP Semiconductors. This design includes the EMC filter and power supply to the antenna and module.

9.2.4.5. ID Card

The MFRC522 Reader is designed to receive signal from ISO/IEC 14443 A/MIFARE compatible contactless cards to render the identification of the current user. There are many ISO/IEC 14443 A/MIFARE compatible cards on the market to choose from. Each card has a unique 32-bit serial number and implements an anti-collision mechanism to support multi-card operations. The IC in the card requires no external power supplied to it in order to read and write at a distance of up to 10cm, regardless of reader. The card operates at 13.56 MHz and the maximum communication speed of the card is 106 KBPS. In addition, the IC in the card comes equipped with an antenna, embedded encryption control logic and communication logic circuit. The data retention period of the card is ten years, it can be rewritten to 100,000 times, and reading from it is limitless.

Contactless RF smart cards are powered by and communicate with the reader via inductive coupling of the reader antenna to the card antenna, as explained above. The two loop antennas effectively form a transformer, as shown in . An alternating magnetic field is produced by sinusoidal current flowing through the reader antenna loop. When the card enters the alternating magnetic field, an alternating current is induced in the card loop antenna. The PICC integrated circuit contains a rectifier and power regulator to convert the AC to DC to power the integrated circuit. The reader amplitude modulates the RF field to send information to the card. The IC contains a demodulator to convert the amplitude modulation to digital signals. The IC also contains a clock extraction circuit that produces a 13.56 MHz digital clock for use within the IC. The data from the reader is clocked in, decoded, and processed by the integrated circuit. The IC communicates with the reader by modulating the loading on the card antenna, which also modulates the load on reader antenna. ISO/IEC 14443 PICCs use an 847.5 kHz subcarrier for load modulation, which allows the reader to filter the subcarrier frequency off the reader antenna and decode the data.

⁶² NXP Semiconductors Inc., “AN1445 Antenna Design Guide for MFRC52x, PN51x and PN53x Rev. 1.2”, 11 Oct. 2010, http://www.nxp.com/products/identification_and_security/nfc_and_reader_ics/nfc_contactless_reader_ics/series/MFRC522.html#documentation Nov. 2013.

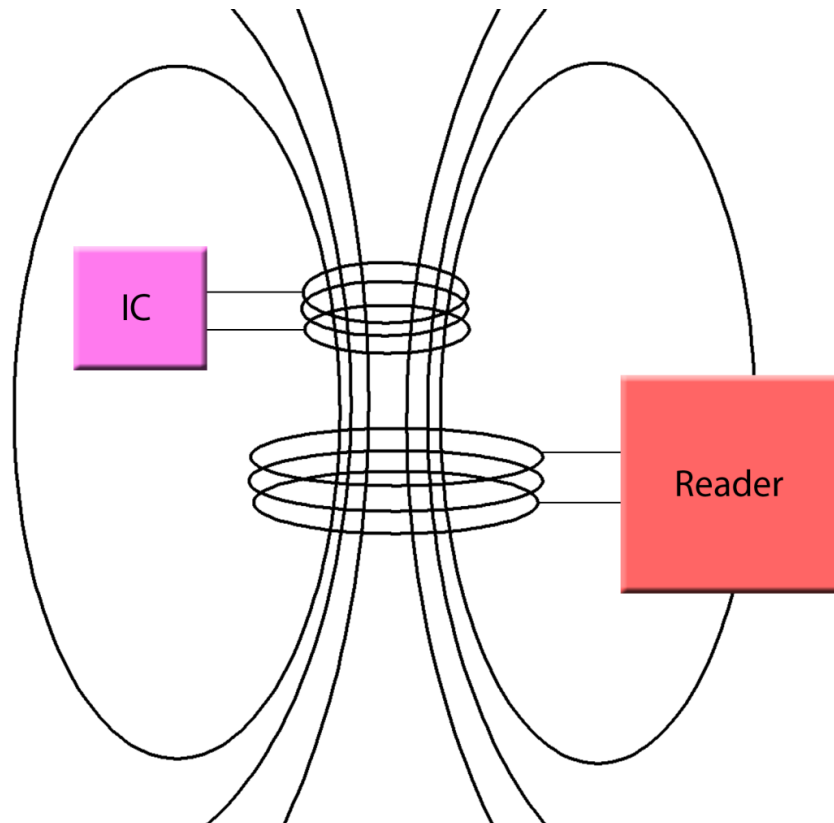


Figure 41: The IC Antenna and Reader Effectively Form a Transformer

9.2.5. Testing

The MFRC522 has the capability to perform a simple digital self-test to validate that once an ID is received it is stored correctly. The test bus is used for tests using the RF transmission between the module and the ID card. A simple code was found that can be loaded onto the microcontroller to perform this test.

9.2.5.1. Range

The test code was run with the ID card held at various measured distances to validate the range and find the maximum distance at which the module will read the card. The experimental distances will range from 1 mm to 12 cm from the reader, with a deviation of 1 mm. The test code was run three times with the ID card held at each distance, for a total of 360 test runs. The range was declared by the furthest distance at which the module receives an ID from the card.

9.2.5.2. Interference

The test code was run with various potential interferences including physical obstructions, other RF transmissions, and other nearby RFID cards. A user might keep

the ID card in a pocket, on a keychain, or in a plastic carrier. All interference tests was performed by running the test code with the intended ID card is held at a distance of 5 cm from the reader. The test was performed using an ID card wrapped in cloth, an ID card in a metal tin, and an ID card in a plastic carrier to simulate these situations, respectively. The RFID system will no doubt be operating among other RF transmissions, including its own RF communication to the server. The test was performed in close proximity (within 1 meter) of an emitting XBee-Pro, Wi-Fi router, and Bluetooth device to check for possible RF interference. The test code was run while a second card is held at distances ranging from 6 cm to 12 cm, with a deviation of 1 cm. Interference was indicated by the module not identifying the correct ID from the card. Each interference test was run three times, per interference, resulting in 42 total interference test runs.

9.3. Body Attachment

This section outlines the body attachment system, which was to be worn by the users to provide information the server needs to create the scenario.

9.3.1. Description

Each user was equipped with a body attachment device which performs user location determination, user orientation (prone, leaning, standing, crouched), and notifies users if they have been ‘hit’ through the use of vibration motors located on at least five ‘hit’ regions. If a user has sustained an injury as determined by the main server, a signal was sent to the user’s body attachment, and the most appropriate vibration motor was triggered, thus indicating a ‘hit’ to the user.

The body attachment sends user position information (as determined by the ultrasonic positioning system) to the main server at regular intervals and upon detecting that a ‘shot’ has been fired to ensure that the most accurate position data is available for ‘hit’ detection. The attachment may also include inertial measurement units (IMUs) for the purpose of determining the velocity at which a user is travelling and the current orientation of a player to further improve ‘hit’ detection accuracy through the ability to predict small changes in movement after a player position has been reported. Any information sent from or received by the attachment was performed using the communications system outlined in section 5.

9.3.2. Design Approach

This section outlines the complete design of the body attachment device, a critical part of the AEAS system.

9.3.3. Body Attachment Microcontroller

The body attachment will make use of a dedicated microcontroller to handle communications with the server, activate/deactivate the vibration motors, determine position results, and process body orientation data. The microcontroller's primary tasks therefore focus on the handling of data for its attached devices. The basic requirements of the microcontroller are listed below.

- Determining when data when must be read from an attached device.
- Read/store data that has been provided by an external component.
- Prepare/store data for an external component.
- Determining when data must be sent to the communications subsystem for transmission.
- Ensuring that any data received is made available to the appropriate component quickly.

To satisfy the data and interface needs of the body attachment's components, the microcontroller will need to provide storage for a small number of bytes (since most data can be discarded after it has been sent or forwarded as it will soon be inaccurate), as well as a number of pins for each required component. The communications system will need to be able to send multiple bytes to and from the body attachment, so hardware support for a communications protocol (such as a UART or I2C interface) is also necessary. The inertial measurement unit was capable of performing the necessary computations through its own on-board processing power, meaning that the microcontroller will only be required to store the resulting data and possibly perform computationally inexpensive transformations to the data. Given these requirements, a simple 8-bit MCU with an operating frequency of approximately 20 to 40 MHz should be appropriate.

For this application, two microcontrollers were found which satisfied the minimum specifications: the Atmel ATmega328 and the Microchip (PIC) 18F2520. The two microcontrollers have approximately the same number of I/O pins, around 2kB of RAM, and support for USART/I2C communications protocols. In addition, both chips consume very little power, making them ideal for use in the attachments which was powered by 9V batteries. The two major differences between the proposed microprocessors are the maximum operating frequency, and cost of each chip. The Microchip device can operate at up to 40 MHz, while the Atmel device was limited to 20 MHz. As a result, the PIC MCU costs nearly twice as much as the Atmel chip, and that is still before any programmers have been acquired. Considering that the computational load on the body attachment is expected to be small, the 20 MHz ATmega328 should provide more than enough processing power. In addition, the IMU used in the attachments is also an Atmel component which uses the same Atmel Studio development environment as the ATmega328. By choosing the Atmel chip, we can minimize costs, reuse a development tool, and also implement similar circuits in both attachments (since a very similar Atmel package was used in the weapon attachment), thus making the ATmega a very practical choice.

Now that the proposed MCU has been established, a package version must be decided upon. The ATmega328 is available in TQFP, DIP, and MLF forms. These versions differ slightly from each other in terms of form factor, pin configuration, and

connection requirements, but are available at virtually the same cost. Considering that the body attachment contains the additional positioning circuit and hardware, it was substantially larger than the weapon attachment. To minimize the size of the body attachment, the surface mounted TQFP package was chosen due to its small, square form factor, and ability to be soldered to a board's surface. The pin positioning may also prove beneficial in the PCB design due to the square pin orientation, as opposed to two long rows of pins. This allows for the components to be positioned on each of the four sides of the MCU instead of being clustered together along two. The pin configuration of the microcontroller is displayed below in and the accompanying . The special pin functions which were used have been listed alongside the corresponding pin.

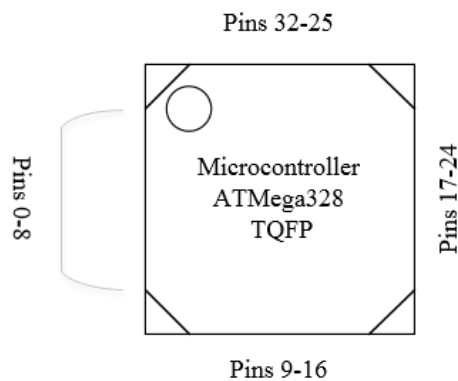


Figure 42: Top view of body attachment microcontroller showing pin locations

PIN	Description
1	PD3
2	PD4 - USART external clock I/O
3	GND
4	VCC
5	GND
6	VCC
7	PB6
8	PB7
9	PD5
10	PD6
11	PD7
12	PB0
13	PB1
14	PB2
15	PB3
16	PB4
17	PB5
18	AVCC
19	ADC6
20	AREF
21	GND
22	ADC7
23	PC0
24	PC1

25	PC2
26	PC3
27	PC4 - SDA (2-wire Serial Bus Data Input/Output Line)
28	PC5 - SCL (2-wire Serial Bus Clock Line)
29	PC6
30	PD0 – USART Data IN (RXD)
31	PD1 – USART Data OUT (TXD)
32	PD2

Table 19: ATMega328TQFP pinout.

The microcontroller will need to interface with the communications system, an inertial measurement unit, the body positioning system’s sensor, the control unit for the vibration motors, and the user identification system. All of these connections were accomplished using the available ports designated as B, C, and D. The designated ports provide twenty-three I/O pins in total, which should be enough to handle all of the devices required. Port D was assigned to the communications system and will perform IO using the UART to exchange data with the XBee-PRO ZB. Port D has designated pins PD4 for USART external clock I/O line, PD0 as USART input pin, and PD1 as USART output pin⁶³, thus for the UART to be used, the communications subsystem must be connected to these three pins. Port B will contain the vibration motor control, and Port C will host the IMU. Additional pins on Port D may also serve as the interface with the user identification system.

Code for the body attachment’s ATMega328 microcontroller was written and uploaded using one of two possible methods. The first option is to use the Atmel Studio environment and a programmer for the device. The second option is such that the bootloader on the device was rewritten to allow for Arduino code to run on the chip⁶⁴. This option allows for the use of an Arduino device to serve as the microcontroller’s programmer, and is actually a slightly less expensive option than the Atmel AVRISP mkII In-System Programmer offered by Atmel. This versatility was another driving factor in the choice of microcontroller since it allows for a greater number of programming languages to be chosen from. Considering that devices such as the XBee are being used, existing Arduino libraries or code may be better suited to interfacing with the device than newly written code, and thus is a consideration being made to ensure optimal performance of the attached devices. Running Arduino code on the device will allow for a higher-level programming language to be used, and will assist in making configuration and overall programming easier; however, if it is found that the Arduino bootloader does not make the best use of the hardware, then the Atmel Studio was used with the default bootloader to run C and assembly language code on the microcontroller.

The communications system was attached to the body microcontroller via the pins PD0, PD1, and PD4 on port D. The XBee is expected to require the use of three pins for the communication between the two devices to occur. Data was transferred to and from the XBee via the USART capability of the hardware. The USART was

⁶³ Pin out, and pin functions taken from Atmel ATMega328 Datasheet. Port D alternate functions, Page 88.
Web: <https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf>

⁶⁴ Modifying the ATMega328 bootloader and uploading Arduino programs. <http://arduino.cc/en/Tutorial/ArduinoToBreadboard>

configured to operate in asynchronous mode (UART) and will thus require pins for TXD and RXD.

The vibration motor control will require the use of three pins on the microcontroller and will occupy the pins [2:0] on port B of the device. The pins was digitally written to one or zero to form the three bit input selection value (ABC) where B0 = C, B1 = B, and B2 = A as requested by the server. Once the pins have been written, the MCU will wait for a set interval before writing a new value to the pins to turn the currently active motor off. To activate a motor, the number of the motor was written as the input selection value to the decoder which acts as the vibration motor control unit. Thus, to activate the motor designated as VM1, the MCU will make the pin B0 go high, while pins B1 and B2 will go low. After a motor has been active, it will remain active for a set interval of approximately five seconds as determined by the timer on the ATMega328. After five seconds has elapsed, the pins was rewritten to the off state ABC = 101.

Any inertial measurement units (Atmel Xplained sensors) was connected on Port C of the microcontroller if additional precision is required. The addition of such sensors allows for greater accuracy in computing position and trajectory, but may be outside the scope of this prototype. Thus, an IMU was added to the body attachment only as necessary. The chosen sensor will require the use of pins PC4 and PC5 due to special I2C functionality where PC4 becomes the SDA (data I/O) line and PC5 becomes the SCL (clock) line.

The RFID system being implemented on the body attachment uses the Serial Peripheral Interface data protocol as outlined in section 6. As a result, a minimum of three pins was required for connecting the MFRC522 component to the body attachment. The required pins relate to the MOSI, MISO, and SCK functionality of the data protocol. The ATMega328 provides special pins for these functions which are located on Port B locations 3, 4, and 5 (or pins 15, 16, and 17) as shown in the table above.

The signal received by the microcontroller from the positioning system is a simple high or low, and thus only a single pin is required for connectivity. The positioning system data was read from Port C pin 1.

For the chosen microcontroller, it is expected that these pin selections was sufficient to ensure the proper operation of the necessary body attachment devices.

9.3.4. Vibration Motor Control

To notify a user that they have sustained an injury, a set of vibration motors was included as part of the body attachment and was divided into ‘hit’ regions. Using the trajectory information, the main server was able to determine approximately where a user has been ‘hit’, and will then send a signal to the body attachment. The body attachment will then activate the appropriate vibration motor to turn on and notify the user. The vibration is expected to be noticeable but not debilitating, and will give the user an idea of where they have been ‘hit’, but not exactly where they have been ‘hit’ due to the need for a very large number of vibration motors. There was a minimum of

five ‘hit’ regions associated with the body attachment, and will notify the users when they have been ‘hit’ in the left or right arm, left or right leg, or torso. This configuration is shown in Figure 43.

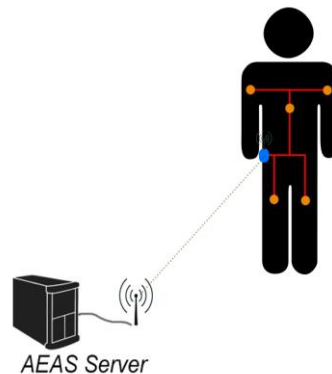


Figure 43: Vibration Motor Hit Regions

Each of the vibration motors must be under the control of the body attachment, and thus a system for controlling the timing and activation of the vibration motors was included. This task was controlled by the body attachment’s microcontroller and a device which has the ability to select among a set of possible outputs based on a provided input value.

The interface between the motor control unit and the vibration motors will likely be an I/O port with at least four lines, and is readily available on the proposed Atmel ATmega328 package as the 8-bit Port B. Given that there will only be five vibration motors and only one motor will need to be active at any given time, a decoder was used to activate the appropriate motor when necessary. Using a 74HC238 3 to 8 Line Decoder requires the availability of three digital pins to switch between any of the five motors (since only one motor should be on at a time). In the circuit containing the decoder, the pin designated G1 should be tied to power (logic one), while the pins $\overline{G2A}$ and $\overline{G2B}$ must be grounded to configure the decoder for the desired input/output combinations⁶⁵. A sample set of input selection signals can be found in below.

G1	$\overline{G2A}$	$\overline{G2B}$	Select Inputs (ABC)	VM 0	VM 1	VM 2	VM 3	VM 4
1	0	0	000	ON	OFF	OFF	OFF	OFF
1	0	0	001	OFF	ON	OFF	OFF	OFF
1	0	0	010	OFF	OFF	ON	OFF	OFF
1	0	0	011	OFF	OFF	OFF	ON	OFF
1	0	0	100	OFF	OFF	OFF	OFF	ON
1	0	0	101	OFF	OFF	OFF	OFF	OFF
1	0	0	110	OFF	OFF	OFF	OFF	OFF
1	0	0	111	OFF	OFF	OFF	OFF	OFF

Table 20: Input selection signals for vibration motor control

⁶⁵ 74HC238 3 to 8 Line Decoder data sheet, page 2.
 Web: <https://www.sparkfun.com/datasheets/Components/General/COM-09577-m74hc238.pdf>

To control the motors from the body attachment's microcontroller, the appropriate values for A, B, and C was written to the pins [2:0] on port B which was directly connected to the decoder inputs.

The vibration motor control will receive instructions to turn on specific motors from the main server via an XBee device using IEEE 802.15.4 wireless. The instructions will then be processed by the ATmega328 on the body attachment, and will resolve to pin values that should be written to port C to form the input values. Once a signal has been received to turn on a vibration motor, the motor in question will remain on for a set time interval of five seconds before shutting off. A timer on the MCU was used to accurately determine when a motor should be shut off. Depending on power constraints, the vibration motors may be multiplexed such that only one may be active at a time, which may introduce a delay between 'hit' notifications if a user is 'hit' in two different regions within a five second interval.

9.3.5. Vibration Motor Circuit

The circuit⁶⁶ shown in the below was used to operate the vibration motors used in the 'hit' notification system. Each of the vibration motors was a 10mm Shaftless Vibration Motor 3.4mm Button Type manufactured by Precision Microdrives. The circuit shown below is designed to ensure stable operation of the motor, and to protect the attached microcontroller from any abrupt changes in voltage that may be produced by the operating motor. As discussed in the previous section, a 74HC238 decoder from ST Microelectronics was used to select from the set of vibration motors, thereby minimizing the number of pins required to connect the five motors.

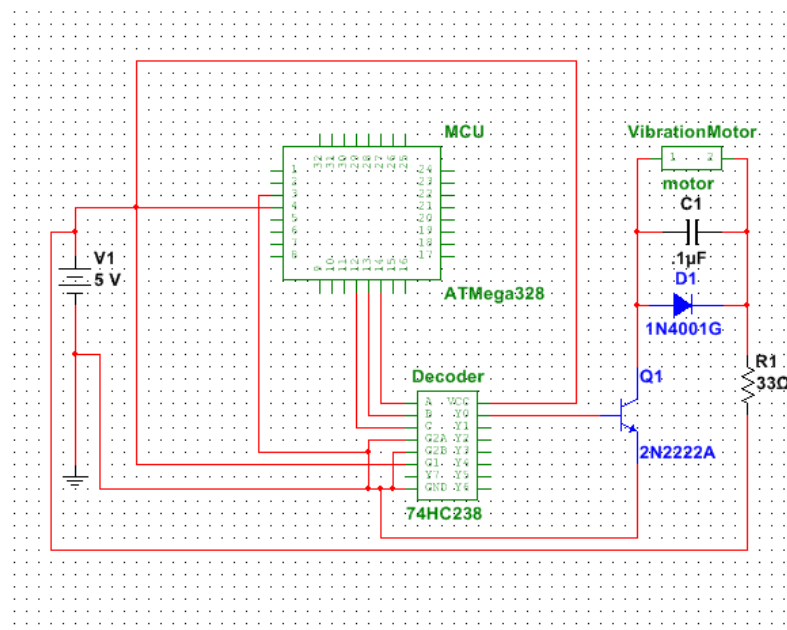


Figure 44: Vibration motor control circuit

⁶⁶ Circuit has been adapted from <http://learningaboutelectronics.com/Articles/Vibration-motor-circuit.php>

9.3.6. User Identification Circuit

The RFID system requires the use of the SPI communications interface to send and receive user data. The configuration requires the use of pins specifically allocated for SPI via the microcontroller, and the connections are shown in below.

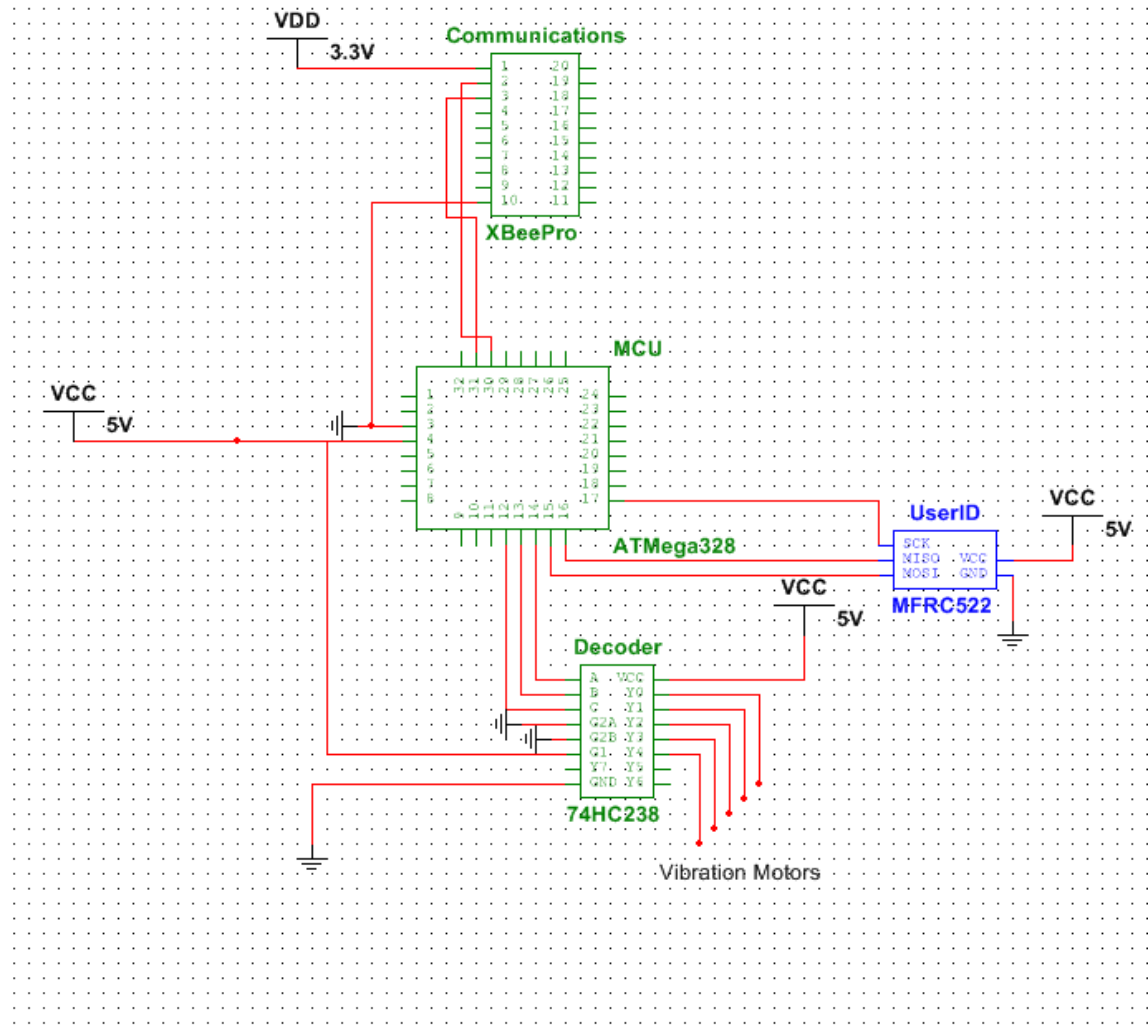


Figure 45: Body attachment circuit for user identification

9.3.7. Data Processing

The body attachment device will need to both generate data and respond to data it has received from the server.

The ultrasonic positioning data was read from the ultrasonic receiver by the body attachment's microcontroller, which will then make the data available to the Xbee RF communications device for transmission to the main server at regular intervals. Measurements are still being performed regarding which voltages correspond to which distance from the ultrasonic transmitters, but the set of voltages read at

specified intervals from the ultrasonic receiver was used to compute the user's location within the bounded playing field as outlined in the ultrasonic positioning system section. The result of the positioning computation will then be forwarded to the main server via the serial connection to the RF communications device.

For any included IMUs, the gyroscope, accelerometer, and magnetometer sensor data was readily available to the body attachment's microcontroller, and requires no additional processing. IMU data will likely be sent via the communications system following the user position data; however, if the load on the server becomes too computationally intensive, user orientation data may be sent less frequently. In addition, the inertial measurement unit's data was processed and determined exactly as described in the weapon attachment section.

9.3.8. I/O Interface

The body attachment must be capable of communication with the main server for the transfer of user position information from the attachment to the server, as well as the transfer of 'hit' information from the server to the attachment. The communications system was connected to the body attachment as shown in below.

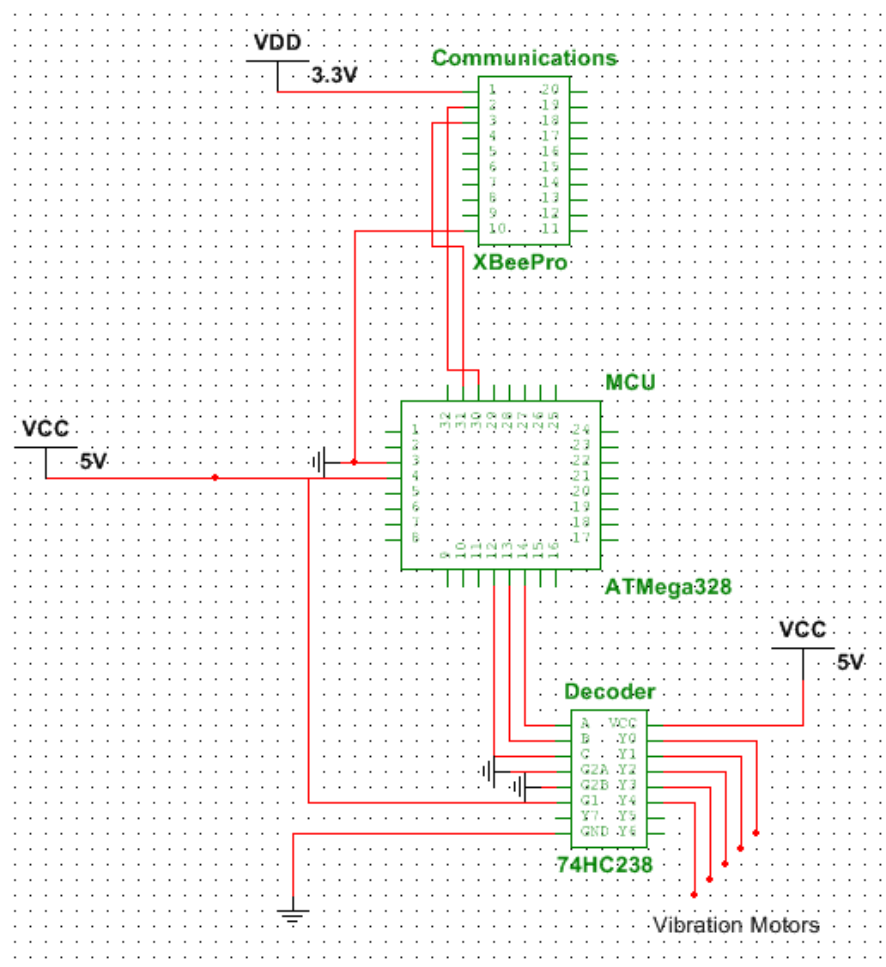


Figure 46: Body attachment circuit containing communications device

9.4. Prototyping

Prototyping of the body attachment was performed by taking the components described above and implementing the complete body attachment circuit shown in section 8.2.5 on a solderless breadboard. The microcontroller will need to be programmed by either an Atmel AVR programmer, or by an Arduino depending on the choice of code being run by the body attachment. Initial testing was performed using code written in the Arduino programming language due to the availability of existing libraries to perform the SPI and UART communications. The ATmega328 was burned with the Arduino bootloader and have code uploaded as necessary to test and run the body attachment components. By using Arduino communications libraries, the need to write code for initializing the hardware was eliminated or minimized allowing for the analysis of data accuracy and throughput to take place immediately. Initial prototyping will primarily focus on ensuring that the positioning system, identification system, communications system, and motor control system work as expected with the microcontroller. At this stage, passing data across the larger AEAS system's components will not be a consideration. The complete body attachment circuit to be implemented during prototyping is displayed in below.

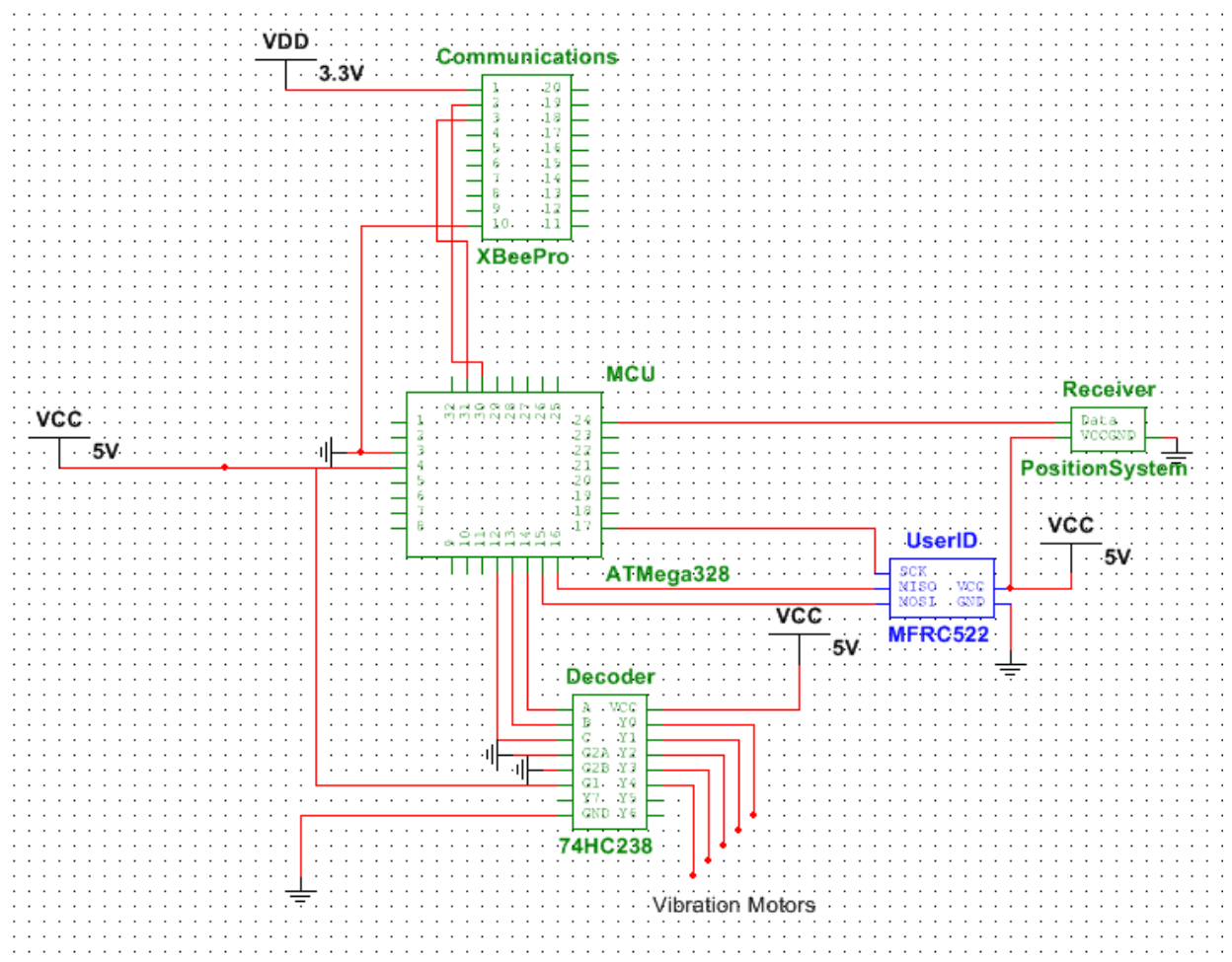


Figure 47: Body attachment circuit

9.4.1. PCB Fabrication

PCB design and fabrication for the body attachment will take place once the final hardware selection has been verified to function correctly, and after any hardware incompatibilities have been resolved. The circuit design will most likely be done using the EAGLE PCB software suite, and manufacturing is expected to be performed by a company such as BatchPCB. PCB Fabrication efforts was immediately following the initial prototyping due to the possibility for manufacturing issues to occur, as well as the required waiting for the finished product to ship (which may take a week or longer).

9.5. Testing

To ensure the correct operation of the body attachment's system of motors and communication capabilities, extensive testing was performed to determine the responsiveness and accuracy of the vibration system, as well as the communications required to activate and deactivate the system.

9.5.1. Vibration Motor Response

Determining the response of the vibration motor system was necessary to determine the effectiveness of using such a mechanism as a notification method. In order to properly test the responsiveness of the vibration motors, extensive calculations of the time required to activate/deactivate the motors as well as the force exerted by the motor over time will also need to be analyzed.

To ensure that the 'hit' notification system is responsive enough that the user knows within a second of being 'hit', the motor control circuit was hooked up to a DC power source, and the time required for the each of the five motors to begin vibrating will attempt to be measured. The decoder select values were hard-coded for the response tests. If the time required to start/stop a motor is found to be too great, the circuits was reevaluated with different components. It is assumed here that the time required by the microcontroller to activate the motor control unit was negligible.

Next, the force exerted by the motor at various voltage levels within the device's operating range will need to be evaluated. The sensation caused by the motor should be noticeable, but not debilitating. The part has a voltage range of approximately 2.5-3.8V⁶⁷, and thus allows for the voltage and various circuit parameters (such as resistor values) to be modified while the most appropriate response is determined. It was important to determine a response that does not result in the motor breaking free from the body attachment, or moving around freely during the combat scenario. It may also be necessary to devise a harness which remains closely attached to the user's body

⁶⁷ 10mm Shaftless Vibration Motor 3.4mm Button Type, Datasheet, voltage range specification.
Web: 28 Nov. 2013

such that the vibration motors remain in positions on the user's body where the response can be clearly felt.

9.5.2. Communications Performance

The performance of the communication subsystem was determined by simply sending as much data between the body attachment and a test XBee as possible. Sample data was generated and stored on the body attachment with the program code. The sample data will then be sent from the body attachment to a computer interfacing via USB with the test XBee. The test XBee will verify the received data and send back response data indicating the success or failure in sending data. Afterward, data was sent from the test XBee to the body attachment to ensure the accuracy of the bidirectional communications. The amount of data sent or received was measured in bytes and performance was determined by calculating the number of correctly transferred bytes over the testing interval.

10. Budget

Below is a table to outline the budget of a prototyping the AEAS system. This includes two weapon attachments, two body attachments, and a server.

Component	Quantity	Cost Per Unit	Total Cost
Development			
Atmel Inertial Two Sensors Xplained development board	1	\$85.54	\$85.54
Microcontroller			
ATMEGA328P with Arduino Bootloader	10	\$0.00	\$0.00
Unit Orientation			
Inertial Measurement Unit	1	\$13.00	\$13.00
Unit Positioning			
MB1200 XL-Maxsonar-EZ0	4	\$46.95	\$187.80
5ft PVC (1 in diameter)	4	\$1.68	\$6.72
4pk Rubber Furniture Pads	2	\$5.75	\$11.50
4 position male connector	4	\$2.64	\$10.56
4 position female connector	4	\$2.64	\$10.56
6 position male connector	1	\$3.17	\$3.17
6 position female connector	1	\$3.17	\$3.17
Power			
9v Battery Holder	1	\$6.95	\$6.95
9v Battery	4	\$5.00	\$20.00
Lithium 9v Battery	1	\$13.00	\$13.00
Vibration Device			
10mm Shaftless Vibration Motor	5	\$5.00	\$25.00
Wireless Communications			
XBee Wireless Pack	1	\$117.00	\$117.00
Trigger Sensor			
Parallax Flexiforce Pressure Sensor	1	\$21.95	\$21.95
PCB Fabrication			
Printed Circuit Boards	3	\$16.68	\$50.04
AMICO 50 pack 16MHZ oscillators	1	\$6.81	\$6.81
5V Voltage Regulators	1	\$5.14	\$5.14
Miscellaneous			
25ft 24 gauge wire	5	\$4.35	\$21.75
Ethernet Cable	1	\$0.00	\$0.00
Arduino UNO	2	\$24.51	\$49.02
Pocket AVR Programmer	1	\$14.95	\$14.95
TOTAL			\$683.63

Table 21: Cost to develop the system

11. Milestones

Below is the outline of project milestones to research, develop, and test the AEAS system. This includes both semesters of senior design.

	Name	Start	Finish	Duration	Qtr 4, 2013				Qtr 1, 2014			Qtr 2, 2014			
					Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May		
1	Deliverables	9/9/13 8:00 AM	4/11/14 5:00 PM	155 days?											
2	Initial Project Document	9/9/13 8:00 AM	9/17/13 5:00 PM	7 days?											
3	Final Research Document	10/4/13 12:00 PM	11/15/13 1:00 PM	30 days?											
4	Critical Design Review	1/6/14 8:00 AM	1/20/14 5:00 PM	11 days?											
5	Conference Paper	3/17/14 8:00 AM	4/11/14 5:00 PM	20 days?											
6	Final Project Document	3/3/14 8:00 AM	4/11/14 5:00 PM	30 days?											
7	Presentation	3/24/14 8:00 AM	4/11/14 5:00 PM	15 days?											
8	Research Phase	8/26/13 8:00 AM	10/28/13 5:00 PM	46 days?											
9	Position System Research	8/26/13 8:00 AM	9/20/13 5:00 PM	20 days?											
10	Orientation System Research	8/26/13 8:00 AM	9/27/13 5:00 PM	25 days?											
11	Communications Research	9/2/13 8:00 AM	10/4/13 5:00 PM	25 days?											
12	Power Research	10/11/13 8:00 AM	10/25/13 5:00 PM	11 days?											
13	Microcontroller Research	9/16/13 8:00 AM	10/25/13 5:00 PM	30 days?											
14	Trigger Sensor Research	9/13/13 8:00 AM	10/4/13 5:00 PM	16 days?											
15	Server Hardware Research	9/2/13 8:00 AM	9/9/13 5:00 PM	6 days?											
16	Server Software Research	9/9/13 8:00 AM	10/11/13 5:00 PM	25 days?											
17	Server Networking Research	9/9/13 8:00 AM	9/20/13 5:00 PM	10 days?											
18	Circuit Design Research	10/14/13 8:00 AM	10/28/13 5:00 PM	11 days?											
19	Design Phase	9/20/13 8:00 AM	4/1/14 5:00 PM	138 days?											
20	Weapon Attachment Design	10/21/13 8:00 AM	11/11/13 5:00 PM	16 days?											
21	Body Attachment Design	10/21/13 8:00 AM	11/11/13 5:00 PM	16 days?											
22	Communication System Design	10/7/13 8:00 AM	10/28/13 5:00 PM	16 days?											
23	Positioning System Design	9/20/13 8:00 AM	10/28/13 5:00 PM	27 days?											
24	Server Software Design	10/14/13 8:00 AM	11/4/13 5:00 PM	16 days?											
25	Web GUI Design	3/10/14 8:00 AM	4/1/14 5:00 PM	17 days?											
26	Acquisition Phase	9/9/13 8:00 AM	11/15/13 5:00 PM	50 days?											
27	Server Hardware Acquisition	9/9/13 8:00 AM	9/16/13 5:00 PM	6 days?											
28	Position System Acquisition	9/27/13 8:00 AM	10/18/13 5:00 PM	16 days?											
29	Orientation System Acquisition	10/4/13 8:00 AM	10/25/13 5:00 PM	16 days?											
30	Communications Acquisition	10/11/13 8:00 AM	11/1/13 5:00 PM	16 days?											
31	Power Acquisition	10/25/13 8:00 AM	11/15/13 5:00 PM	16 days?											
32	Microcontroller Acquisition	10/18/13 8:00 AM	11/8/13 5:00 PM	16 days?											
33	Trigger Sensor Acquisition	10/4/13 8:00 AM	10/25/13 5:00 PM	16 days?											
34	Prototype Phase	12/16/13 8:00 AM	3/19/14 5:00 PM	68 days?											
35	Weapon Attachment Initial Prototyping	12/16/13 8:00 AM	2/3/14 5:00 PM	36 days?											
36	Body Attachment Initial Prototyping	12/16/13 8:00 AM	2/3/14 5:00 PM	36 days?											
37	Weapon Attachment Final Prototyping	3/3/14 8:00 AM	3/19/14 5:00 PM	13 days?											
38	Body Attachment Final Prototyping	3/3/14 8:00 AM	3/19/14 5:00 PM	13 days?											
39	Test Phase	12/16/13 8:00 AM	4/1/14 5:00 PM	77 days?											
40	Position Precision Analysis	12/16/13 8:00 AM	1/27/14 5:00 PM	31 days?											
41	Orientation Precision Analysis	12/16/13 8:00 AM	1/27/14 5:00 PM	31 days?											
42	Weapon Attachment Initial Testing	2/3/14 8:00 AM	2/24/14 5:00 PM	16 days?											
43	Body Attachment Initial Testing	2/3/14 8:00 AM	2/24/14 5:00 PM	16 days?											
44	Weapon Attachment Final Testing	3/19/14 8:00 AM	3/26/14 5:00 PM	6 days?											
45	Body Attachment Final Testing	3/19/14 8:00 AM	3/26/14 5:00 PM	6 days?											
46	Server Testing	1/6/14 8:00 AM	2/10/14 5:00 PM	26 days?											
47	Initial Integrated Testing	2/10/14 8:00 AM	3/3/14 5:00 PM	16 days?											
48	Final Integrated Testing	3/19/14 8:00 AM	4/1/14 5:00 PM	10 days?											
49	Web GUI Testing	3/19/14 8:00 AM	4/1/14 5:00 PM	10 days?											

12. Conclusion

Although laser based training systems have seen extensive capability enhancements over the past decade, they remain expensive, burdensome, and somewhat unrealistic. A more cost-effective and accurate system is not only desirable, but potentially achievable with current advances in microelectronics and inexpensive, accurate positioning technologies.

The design accomplished by this team presents a system which we hope will lead to improvements upon the currently used training systems while offering greater scalability and extensibility options to make the system adaptable to any training scenario. By using readily available components and exploring various design approaches for each system element, we have outlined a set of options which should result in a highly customizable training solution that can be tailored to meet the needs of the trainers. By making changes to the hardware and software components, features such as additional user positioning accuracy, improved system range, more accurate trajectory determinations, and support for many users can be added as necessary. By making the system simple to modify, we expect that the design presented here can be adapted to meet the cost, accuracy, and situational needs for many different training exercises performed by the armed forces.

Over the course of this document, the most practical and least expensive approach has been established, and was the prototype system that is initially produced. This version primarily aims to improve upon the lack of accuracy found in laser based systems, while also greatly reducing costs. The system does have a few drawbacks, such as the need for a playing area free of obstructions, but future iterations can improve upon these shortcomings by simply selecting some of the more advanced hardware configurations. Additionally, it is expected that software improvements and small hardware changes may occur which further enhance the capabilities of the system as development continues. The currently considered system is expected to perform training exercises with at least two users and attain the stated requirements for accuracy and usability.

The system lends itself very well to augmentation and enhancement. As newer technologies are developed that provide superior capabilities, they can be integrated into the AEAS system with relatively little effort due to its use of simple cost-effective components and software. Ultimately, the results of this project will outline what the latest economical technologies can achieve in the way of providing a realistic firefight simulation, and are expected to result in further efforts to provide a superior solution to the future of our Armed Forces.

Appendix A Copyright Permissions

Figure 23: FlexiForce A201 Sensor. Reprinted with permission from Tekscan Inc.

From: Ryan Sivek [rsivek@live.com]
To: support@tekscan.com
Sent: Sat 11/23/2013 2:08 PM
Subject: Image Request

Hello,

I and my team are senior undergraduates working on a senior design project at the University of Central Florida. We are planning to use a FlexiForce A201 pressure sensor for our project and would like permission to use and crop an image on your website for our report. The picture is at the following URL:

http://www.tekscan.com/sites/default/files/A201_allfour_white_0.jpg

We would like to crop the image to one of the four sensors in the image for our report. If you would like more information about our project please don't hesitate to ask.

We greatly appreciate your prompt response.

Thank you,
Ryan Sivek

From: Jeannine Croteau [jcroteau@tekscan.com]
To: Ryan Sivek [rsivek@live.com]
Sent: Mon 11/25/2013 8:43 AM
Subject: FlexiForce image

Hi Ryan

You are welcome to use our image from our website. Could you please just caption the image as a FlexiForce A201 sensor and source Tekscan? Thanks! And I would like to hear more about your project, applications using our sensors are always of interest!

Thanks!

Jeannine Croteau
FlexiForce Marketing Campaign Coordinator
Tekscan, Inc
617.464.4500 Ext 245

Tekscan Confidentiality Notice: This e-mail message, including any attachments, is intended only for the individual or entity to which it is addressed. If you are not the intended recipient, please notify us immediately by replying to this message and then delete this message and its attachments from your system.
