**Final Report**

*Document Number: 3*

*Date: 28, April Thursday, 2016*

***Pocket Pew Laser Tag***

*Karishma Hirani*

*Robert DeRosa*

*Anthony Russo*

*Brian Kancylarz*

Professor Yu

Computer Science & Software Engineering Department

Monmouth University

West Long Branch, NJ 07764-1898

*Table of Contents*

[Acknowledgements 4](#_Toc449608502)

[1 Introduction 4](#_Toc449608503)

[2 Literature Review 5](#_Toc449608504)

[2.1 Introduction 5](#_Toc449608505)

[2.1.1 Overview 5](#_Toc449608506)

[2.1.2 Development 6](#_Toc449608507)

[2.2 Background and Motivation 7](#_Toc449608508)

[2.2.1 Most Relevant Sources 7](#_Toc449608509)

[2.2.2 Research 7](#_Toc449608510)

[2.3 Project Division 8](#_Toc449608511)

[2.3.1 Hardware 8](#_Toc449608512)

[2.3.2 Software 10](#_Toc449608513)

[2.3.3 Communication 13](#_Toc449608514)

[2.4 Discussion and Future Work 14](#_Toc449608515)

[2.4.1 Findings and Data 14](#_Toc449608516)

[2.4.2 Future Research 15](#_Toc449608517)

[2.5 Conclusion 15](#_Toc449608518)

[3 Design 15](#_Toc449608519)

[3.1 Scope 15](#_Toc449608520)

[3.1.1 Identification 15](#_Toc449608521)

[3.1.2 System Overview 16](#_Toc449608522)

[3.2 Requirements 17](#_Toc449608523)

[3.2.1 Required States and Modes 17](#_Toc449608524)

[3.2.2 Adaptation Requirements 17](#_Toc449608525)

[3.2.3 Safety Requirements 17](#_Toc449608526)

[3.2.4 Security and Privacy Requirements 17](#_Toc449608527)

[3.2.5 Software Item Environment Requirements 17](#_Toc449608528)

[3.2.6 Packaging Requirements 18](#_Toc449608529)

[3.2.7 Hardware Requirements 18](#_Toc449608530)

[3.3 Software Design 21](#_Toc449608531)

[3.3.1 Initial Class/Activity Diagram 21](#_Toc449608532)

[3.3.2 Final Class/Activity Diagram 22](#_Toc449608533)

[4 Implementation 23](#_Toc449608534)

[4.1 Hardware 23](#_Toc449608535)

[4.2 Software 23](#_Toc449608536)

[4.3 Team Dynamic 30](#_Toc449608537)

[5 Conclusion 31](#_Toc449608538)

[5.1 Key Developments 31](#_Toc449608539)

[5.2 Market Counterparts and Contributions 32](#_Toc449608540)

[5.3 Vision 32](#_Toc449608541)

[Bibliography 33](#_Toc449608542)

[Individual Contributions 36](#_Toc449608543)

# Acknowledgements

Monmouth University

Adafruit

Surplusshed

Arduino

Android

Radioshack

Spinning Designs Incorporated in Asbury, NJ

thingiverse

# 1 Introduction

Our Senior Project idea is called *Pocket 'Pew' Portable Laser Tag.* We will create a functioning laser tag application that can be used with a modern android smartphone. This system will be an improvement on laser tag systems that use guns or extra handheld devices and require special arenas. The smartphone application is the interface of the system and it will work with a real-world attachment that we will design mainly using an Arduino module and a Bluetooth module. Although the extra hardware is still necessary, it will be greatly enhanced by the versatility of using a phone application. This is due to the fact that our culture highly values smartphones and those who have them tend to take them everywhere. Most of our effort will be centered around the app and its features.

A user must initiate the app on their phone while having the module attached to their phone in order to "tag" someone. This module will communicate with the app via Bluetooth. The user can point their phone at an opponent in battling and press the corresponding button on the app to “shoot” a laser at them. If this infrared laser hits the target which is the other user's phone, and they have the system installed as well, their application will register a strike. This record will be saved in the target's application as the strike counts against them and in the player's application as their strikes count against the enemy. It will increment every time that user manages to tag their target with their laser. Once that target has been tagged a total of 5 times, their app will make a noise signifying the laser has been disarmed. This means the match is over.

The design of this project includes two main components: the application and the real world attachment. The application will be made for android using the Java programming language and environment. We will create two attachments that each include the Arduino module, one Bluetooth module, an infrared laser, and a receiver. Once we have made the application we will test it out in the real-world using two android phone. Using practical trials, we can refine the device and the app so that it functions in a simple way. Our design will be agile in nature, focusing on basic function and streamlining the necessary requirements. The system will be highly marketable to all ages, perhaps with a sharper focus on the sci-fi community as many people find laser tag matches to be enjoyable. However, our device will allow these matches to be played virtually anywhere at any time with any other user in the system.

# 2 Literature Review

## 2.1 Introduction

### 2.1.1 Overview

We are investigating the development of modern laser tag systems. Our aim is to create a fully functioning system which can be implemented with android smartphones running OS 4.4 and higher. Over the last 30 years, laser tag systems have been a well-recognized part of our cultural entertainment. However, they were originally developed as combat systems for the United States Army (MILES Tag System). Over the course of consumer development, laser tag has been modernized. The same gameplay structures still exist, with traditional games such as 3 vs 3 or capture the flag, etc.

It has become widely popular such that there are full-blown arenas dedicated to it. In the United States alone there are even conventions where users can gather to play. By implementing this system with a smartphone, we will remove the traditional weapon and replace it with something that almost everyone already owns. We make the experience much more user-friendly and customizable. It will also allow users to initiate laser tag with any other user currently in range. The physical aspect of this game will be completely updated while the original game structures will be retained.

### 2.1.2 Development

Laser Tag was originally founded by the company Worlds of Wonder in 1986 (http://www.lasertagmuseum.com/brand/l-p/lazer-tag). They created a game called “Lazer Tag” which utilized plastic weapons and armor to simulate realistic battle. Their original equipment consisted of “a pistol, a ‘StarSensor’, a belt and holster for the pistol, and a harness which the sensor could be attached to” (http://www.lasertagmuseum.com/brand/l-p/lazer-tag). The aim in the past was to make the game more like realistic combat which is why guns are usually involved. Other systems such as DIY MilesTag were also developed around the same time, using the same weapon-sensor-infrared system (https://en.wikipedia.org/wiki/Laser\_tag#Outdoor\_equipmen[t](https://en.wikipedia.org/wiki/Laser_tag#Outdoor_equipment)).

When developing a laser tag system there are two main ways to achieve communication. A developer may use radio frequencies, or Infrared. The greatest example of RF Laser Tag is the Lazer Runner laser tag system. The system consists of an RF circuit board and internal antennae for data transmission (http://www.lazerrunner.com/How-Laser-Tag-Works.html). However, infrared systems are the more widely used equipment in Laser Tag games. This system utilizes two laser beams. One is the infrared beam, which carries data. This IR beam travels alongside the aesthetic laser beam. The Lazer Runner website outlines some flaws of using infrared such as, “They cannot be focused to pinpoint accuracy thereby creating beams that vary from a few inches to several feet in diameter. The beams can become unaligned which can cause targeting errors. Stray signals could occur which can cause players to be deactivated by IR that has bounced off walls or ceilings” (http://www.lazerrunner.com/How-Laser-Tag-Works.html).

Regardless of which system is implemented, laser tag can take place indoors or outdoors. Indoor arenas are extremely popular, usually with stable power dependencies for each weapon. There are also small, low-power residential systems which are completely integrated within toy laser tag guns. In the present day there are DIY systems such as DuinoTag, utilizing an Arduino circuit board, and various electronic materials (http://www.instructables.com/id/Duino-Tagger/). These DIY systems don’t have a lot of complex game capability and are simply used for shooting matches. However, our design will build on this by incorporating one of the most powerful tools in our world today, the smartphone. We will take the hardware and place it into a well-designed attachment. We will also incorporate a phone application that can have infinite possibilities for game expansion.

## 2.2 Background and Motivation

### 2.2.1 Most Relevant Sources

We began by researching products that didn’t require a large arena. There were many hits for DIY laser tag. One of the most crucial resources for creating the foundation of our project are instructable tutorials for forming the original DuinoTag system (<http://www.instructables.com/id/Duino-Tagger/>). Apart from required materials, we learned that most people have made attachments which still keep the concept of laser gun. There’s a lot of focus on making it still feel like a traditional gun battle. The noises are built in and typically unchangeable. Even if phones are somehow incorporated, they are made to fit into the hardware. What we decided it to form the hardware around the phone instead. We compared this list of parts with another list to determine the most fundamental pieces (<http://www.lasertagparts.com/mtdesign.htm>).

### 2.2.2 Research

Upon discovering the Arduino system, we were driven to research the Arduino website for the most convenient model. In comparing the builds, we selected one of the more full-sized beginner models, the Arduino Uno. Arduino boards function as tools of interpretation. They are how virtual development can be translated into physical development. A basic overview of their features includes the ability to “read inputs such as light on a sensor, a finger on a button, or a message of some kind. They can turn this reading and turn it into an output such as activating a motor, turning on an LED, or publishing something online” (https://www.arduino.cc/en/Guide/Introduction). The Arduino, like most development materials, comes with its own Integrated Development environment. We researched the IDE which must be used in connection with the board and discovered that it can be used to create code in several popular languages. We selected C and C++ due to group experience.

Concerning the app, we knew that Android would be the best system to work with because our group has more experience with Android development. It is a more feasible goal that can be achieved within the allocated project time. To accommodate a user-friendly interface, we decided to design and implement our own android app. Due to group experience again, we selected Android Studio as our IDE for creating the app.

In order to get the app to communicate to the most central portion of hardware, the Arduino, we decided to involve Bluetooth. It is a widely known wireless communication device which many of our common devices rely on today. It is certainly an essential part of smartphones. Wireless connectivity is hugely important in communicating between two weapons as well. For a match to take place there has to be two users whose apps present information that comes from the two attachments (sender and receiver). From there, we researched different Bluetooth modules. We selected the Adafruit Bluefruit LE UART Friend based on the protocol of communication as well as the size and experience-level required to implement it with an app (https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend).

Our initial goal was to create a very simple easy device to put together. Therefore, we would learn more about the product and could advance design and development to construct a more efficient system if necessary. In order to contain all of the hardware, we considered 3D printing the attachment. This would be more desirable because we could engineer it to be almost universal if possible. Upon completion of research, we divided our project into three main parts; Hardware, Software, and Communication. We also came up with basic requirements. The app must be able to shoot a signal at another phone and that phone must be able to receive that signal. The strike should be recorded on the app for both phones. And the app should have settings for how many points determine a match. A more refined list of requirements will be available in our Project design.

## 2.3 Project Division

### 2.3.1 Hardware

#### 2.3.1.1 Smartphone

In our project, functional requirements will be incorporated into our Smartphone App. The Smartphone itself must be able to run Android 4.3 and higher. Effectively, any model Android smartphone made in the last 5 years will be able to run the application. Our team will be using two undecided android smartphones in our testing.

#### 2.3.1.2 Arduino Uno

The central piece of hardware in our attachment is the Arduino Uno. This is a microcontroller based off the ATmega48A chip (https://www.arduino.cc/en/Main/ArduinoBoardUno). It has 14 digital input/output pins and 6 analog inputs. The Uno runs on 5 volts, with a built in power regulator. It is 68.6mm by 53.4 mm which is critical for our project as we need small pieces to fit inside the attachment. During research we selected this board over others due to how easy implementation will be. The Arduino Uno is easy to maintain and reprogram which will be useful during our refining process.

#### 2.3.1.3 Bluefruit LE UART

The Arduino Uno will be connected with the Adafruit Bluefruit LE UART Friend. This module has a 5V input and 3.3V voltage regulation (https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend). The most important ports of the Bluetooth are the Rx and Tx ports which can be used for receiving and transmitting messages, respectively. These ports utilized the ASCII character set. The Bluetooth will be the middle-man in our communications between the Arduino controller and the Smartphone’s Bluetooth.

#### 2.3.1.4 The Laser

The Arduino Uno will also be connected with an Infrared LED emitter and Infrared receiver which handles the data transmission of successful strikes in the game. The emitter is essentially the shooter. It will be very bright, running at 1.6 volts. The emission will be at a 940nm wavelength. The emitter will be collimated through a small acrylic magnifying glass of about 20 mm. This will create the beam which will be sent through a small PVC tube. However, because this is an LED, the beam will very safe. Ordinary lasers such as those found in pointers, are harmful to the eye. One of the advantages of our product is the safety aspect of this laser. On the receiving end, there will be an IR receiver that runs at 3-5 volts. It will have programmable threshold sensitivity which allows us to make the game more realistic. This means that you only hit things which you are aiming directly at. It will read in the 940nm wavelength. The attachment which contains the Arduino, the Bluefruit module, and these other pieces of hardware will be 3D printed as a nylon based clip-on with an accessible compartment for the power source, a 9-volt battery.

### 2.3.2 Software

#### 2.3.2.1 Environment

We will use two open-source IDEs for project development. Our primary IDE for this assignment will be Android Studio. Android Studio was chosen as it is the current standard for the creation of Android applications. Furthermore, our team has a strong background in the use of Android Studio. The Arduino IDE will be used to generate the interface for the Arduino Uno. As our microcontroller, we will need to teach the board what certain signals mean, and how to interpret them. That way we can translate all virtual actions into physical ones and vice versa.

#### 2.3.2.2 Languages

The Android Studio uses Java; thus our application will be programmed in Java. We will use Object-Oriented Programming to create panels for the application. Each screen will have devoted functions but only one screen is viewed on the phone at any given time. The Arduino components will utilize C/C++ functions. Our Arduino components will extend the capabilities of our android application to send information through a focused IR beam. Some phones already have this capacity; however, as IR Emitters and Receivers must be paired to the same frequency to exchange information, we will forego any built in IR capacity in favor of a standard pair.

#### 2.3.2.3 Our application

##### 2.3.2.3.1 Basics of the Application:

The phone with the application will be paired via Bluetooth to one’s own personal IR receiver (The Target) and IR emitter (The Blaster). The IR Blaster is attached to the user’s phone and controlled via our application through Bluetooth signals. Our application will also handle setting up of communications between every attachment’s Bluetooth. There is a lot of overhead on the part of the user’s smartphone but with newer operating systems this is not much of an issue. Using a button press on the phone’s screen; each press will transmit an “attack” beam, activate a color LED, play a sound and vibrate the phone. The data sent by the IR beam will carry “shot information” to another person’s app if their target is hit. This information sent by the app will likely include damage, the source, and other information necessary for gameplay.

The IR receiver will receive Infrared laser information fired from other players. Shots received from other players will have their information send back to the app via Bluetooth. The application will handle that data in order to perform functions such as deduct lives, take damage, vibrate the phone, disable one’s blaster or otherwise handle the information appropriately. Data Transfer is enabled through Pulse-Length Modulation. This means a lens-focused LED will flash at a frequency, which can only be received by a receiver of the same frequency. The LED will turn on and off at this speed which helps to differentiate the signal from the IR light of the sun. A header signal will flash will prepare an enemy system. This will protect the system from perceiving any stray signals as a shot. The bits following the header will provide information. Variable LED pulses will be interpreted as bits by our software and in turn our app.

##### 2.3.2.3.2 Application Features

As previously mentioned, our application coordinates with an Arduino IR Emitter and IR Receiver to enable Android phones to support a laser tag game between two or more players. These are the planned features that we have for our initial user interface. We have a list of events which will be handled by our programming.

Interface

1. Users will be able to initiate pairing with their module via Bluetooth connection signal symbols
2. Users will be able to see a total number of lives on themselves during the match, otherwise known as their own death rate.
3. The settings page of our app will interact with the settings page of the android smartphone to turn on the Bluetooth.
4. There will be a timeout function, displayed during the match which activates based on how many shots were used. This simulates running out of rounds in real-life. The timeout is basically a timer to keep track until the weapon has been reloaded and can fire lasers again. When this timeout is active, the shooting system will not work, but the application will still register received attacks.

Event-Handling

1. Performs functions based on shots fired and received (Vibration, LEDs, Life reduction, etc.)
2. Coordinates with an Arduino IR Emitter and IR Receiver to enable Android phones to play laser tag.
3. Basic “Free play” mode.

Gameplay

1. The basic mode of the game will be a “free play”.
2. Free play involves no joining or exiting of a game mode.
3. All players set to free play begin with 5 lives.
4. A free for all where any number of players can battle each other.
5. Each player’s app will accept any shots received from any other player and deduct lives accordingly.
6. Running out of lives will disable one’s firing capability until they leave the match.
7. Free play will be used as a test bed to develop future gameplay modes.

#### 2.3.2.4 Challenges

We hope to implement new game modes and themes. Testing is set to be conducted halfway through the product in order to gather data on refinement processes. We also face the challenge of streamlining user-interface to try and compete with apps already on the market. However, we will create our interface using our own designs. As always, user error is a possible variable which must be accounted for in programming, so we will be taking on this challenge as well.

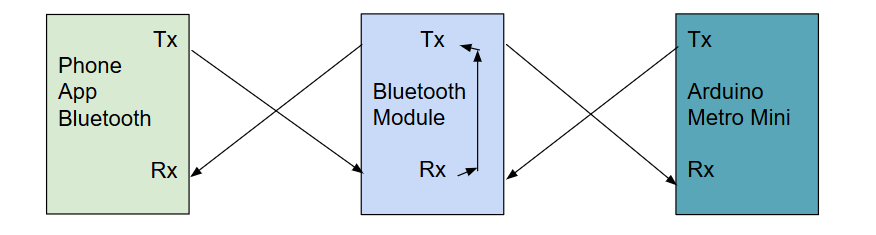
### 2.3.3 Communication

Using open source code from the Bluetooth Adafruit website we will enable the Bluetooth to take in communication from the Arduino and also relay it to the Bluetooth in our own smartphone. This portion of programming will be carried out in Android Studio. On our Bluefruit module, we will be utilizing the two main ports TX and RX. TX is for transmitting, and Rx is for receiving. The protocols of communication that are used in Bluetooth devices today are UART and SPI. We will be using UART serial communication. UART stands for Universal Asynchronous Receive/Transmit, which allows one controller to receive and others to transmit to it. This is the perfect protocol for laser tag because there will be multiple users transmitting. Compared with the USB protocol, UART is more cost effective and requires less software overhead.

In our first phase of communication testing, we will send one character from our Android application to the Arduino and have it turn our LED on. In terms of travel, the Bluetooth in our phone transmits the message to the Bluefruit module, which receives it on its Rx port. It then transmits this message to the Arduino. The Arduino reads the message in on its serial port. It has been coded to interpret it as a signal to switch the light on. Thus in android studio we will need to code TX and RX methods (<http://www.tutorialspoint.com/android/android_bluetooth.htm>). Since Bluetooth is so widely used, if a consumer buys a new device they can expect it to be compatible with any other Bluetooth device they probably already own. This means our product will pair with updated systems without much maintenance on our part.

It is simple to use because we will simply pair the phone with the Laser Tag attachment via the internal Bluetooth modules. Usually this involves going into the Bluetooth settings of one device and pushing a button on the other. We will have to code the Arduino to send signals to the Bluetooth to enable pairing. We selected this method because this is easier than setting up communication via Wi-Fi, which often involves a middle device, and having to connect to it with both devices and entering a secure password.

It is easier to stay connected than with just Infrared, which requires that the two devices stay pointed at each other. With Bluetooth, the two devices can stay connected as long as they are within range.



## 2.4 Discussion and Future Work

### 2.4.1 Findings and Data

    Based on our findings, we have concluded that our project will be an efficient rebuild of the original laser tag system. Our updates will be some of the largest to be added to the timeline of laser tag development. We have researched how the original system was created. Based on this research we considered residential systems. Using smaller scale laser tag, we discovered that the Arduino is highly recommended for the kind of project we are attempting. So we have chosen materials based on a condensed list of parts that many laser tag systems used. In our design decisions, we chose to build the physical system by focusing on how it can work with a smartphone instead of on its own. We considered this to be one of our most important choices because our product should be portable and appeal to the average consumer. By adapting the technology of similar systems that were made before ours, we will create a hand-held laser-tag application with a physical component to make our app more realistic. With the hardware available to us, we will need to develop a user-friendly experience. We anticipate our greatest challenge will be getting multiple players to be able to interact smoothly. However, we will undergo testing using only two users and from there work out the biggest issues so that we can consider potential solutions earlier in the development process.

### 2.4.2 Future Research

    We hope to make our product more energy efficient and compact. We will also implement the more complex features in the app once we get it to function as a basic hit counter. One of the features we would like to implement is a range view to see how far a target is from you by looking at the screen as you point your phone at them. Once we have working hardware we will refine it focusing on user experience again. We may build a tag team system that can call nearby users for support using location data from google. Features that can be used to make the app more customizable would be adding a color-picker for the LED laser. This may be done after the project is finished.

## 2.5 Conclusion

    Pocket Pew Laser Tag will enable users to have engaging laser tag battles with each other at any given time. We are utilizing a variety of open source resources. Our chosen Integrated Development environments will assist in making our own programs and encoding our own interfacing between our hardware and our application. With a strong grasp of fundamental requirements, we will create a simple but effective system that can be targeted at any age group and audience of smartphone users. We will remove traditional weapons from our laser tag system in order to make the game friendlier. However, we will reuse the same gameplay modes which many laser tag consumers have become used to. This ensures our interface will fold into original games while still embodying modern entertainment.

# 3 Design

## 3.1 Scope

### 3.1.1 Identification

The purpose of the Pocket “Pew” Portable Laser Tag System is to allow users to be able to easily play laser tag as long as they have a special set of hardware and a phone with an application to connect to said hardware installed.  Normally, laser tag must be played in a special facility, with expensive equipment.  This equipment usually includes a laser “gun” that fires a powerful laser, and a special vest that protects the wearer from the radiation that can be emitted from the lasers.  Players shoot other players with the laser gun, aiming for targets that are on other players’ vests.  They rack up points if they hit these targets, and lose points if their targets are hit by other players.

The Pocket “Pew” Portable Laser Tag System is a set of software and hardware that when used together will allow users to play a mobile version of laser tag.  The “software” refers to a mobile application for the Android operating system that will connect with hardware to allow users to play laser tag on their phones.  The “hardware” refers to a physical module, which the software connects to that will give the phone extended capabilities that will enable it to be used as a laser tag gun.

This release will be version 1.0, the first release of this software.

### 3.1.2 System Overview

The Pocket “Pew” Portable Laser Tag System consists of two parts, the software and hardware. Therefore, the breakdown below will be separated into software and hardware sections.

The hardware module will allow the user to shoot a safe laser at another player who has a connected, powered-on module. The system will be lightweight. Our application will take up as little resources as possible to decrease Android Operating System overhead. The module will be battery-powered with its own individual battery to enforce the policy of not taking up Android Operating System resources.

The application will begin by showing users instructions.  It will then go to a connection screen, which will enable the application to communicate with the Bluetooth module via UART.  After connecting, the play screen will display, with a large “fire” button as well as ways to change the settings. If the fire button is pushed, the phone sends an ‘f’ to the Bluetooth, then to the Arduino, which then pulses the IR LED.  If the IR receiver is hit, then the Arduino sends an ‘r’ to the Bluetooth, and then to the phone, which deducts a point from the users score.

## 3.2 Requirements

### 3.2.1 Required States and Modes

The application will consist of the following states.  It will be Idle when it is either not yet begun the connection process to the module or is connected to the module, but not receiving or firing.  It will be Ready when it is searching for the module via UART connection.  It will be Active when it is connected and either firing or receiving fire.

### 3.2.2 Adaptation Requirements

Installation will require the user to have an Internet connection only to download the application.  The module will come with a clip to allow it to be used universally across all Android phones.  The module will not be compatible with tablets or computers as they are too big for the clip.

### 3.2.3 Safety Requirements

The concept did not call for protection from whatever is fired from the module to ensure the portability of the game, so using lasers had to be ruled out.  Infrared is being used instead.  The game will be as safe as if someone pointed a remote control at another and pressed a button on it.

### 3.2.4 Security and Privacy Requirements

As this application does not connect to the Internet in any way, there are no security and privacy issue.  The only permission the user will have to give is to allow the application to be installed on their phone. It will be a closed-system application.

### 3.2.5 Software Item Environment Requirements

This application was developed and tested on Android devices running version 4.4.2 and later.  As such, it is recommended to use a device running at least Android 4.4.2.  The application will not run on smart watches, smart televisions, game consoles, Google Glass, or in car entertainment systems running the Android operating system.  It will be guaranteed to run on any other Android device running at least version 4.4.2, but the module will not be compatible with tablets or computers due to the size of such devices.  Therefore, this application is recommended only for smartphones and MP3 players. Android Studio and Arduino Sketch Builder required.

### 3.2.6 Packaging Requirements

The application will be distributed for free in the Google Play Store.  The description of the app will state while it is a free download, it will require the user to purchase the module, and a link will be included that will take the user to an online commerce site where the module can be purchased.  The module will come packaged in a standard box that is expected of most hardware.

### 3.2.7 Hardware Requirements

#### 3.2.7.1 Materials for Laser Module

1. Super-bright 5mm IR LED Transmitter - 940nm
   1. 50-ohm resistor
2. IR (Infrared) Receiver Sensor - TSOP38238
3. Bluefruit LE UART Friend - Bluetooth Low Energy (BLE)
4. Small Acrylic Magnifying Glass (20mm)
5. 3D printed plastic nylon case with lense tube
6. Insulated conductive wires
7. Glue gun with glue stick
8. 3D printer
9. 9-volt battery
10. Switch
11. Battery connector plastic circuitry piece
12. Color LED
    1. 330 Ohm resistor
13. Arduino Uno or Metro Mini

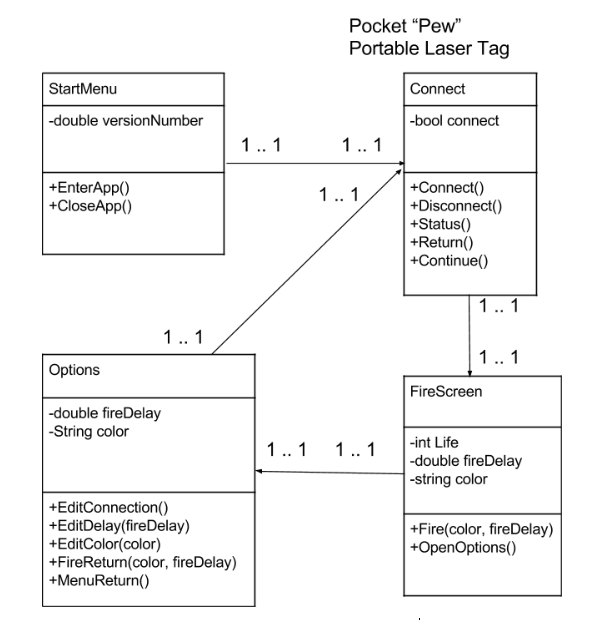
#### 3.2.7.2 Diagram

#### Copy of Android Lazer Tag Design Layout.png3.2.7.3 Key

1. RXI - This is the UART Receive pin *into* the breakout (MCU --> Bluefruit LE). This has a logic level shifter on it, you can use 3-5V logic.
2. TXO - This is the UART Transmit pin *out* of the breakout (Bluefruit LE --> MCU), it's at 3.3V logic level.
3. CTS - Clear to Send hardware flow control pin *into the* breakout (MCU --> Bluefruit LE). Use this pin to tell the Bluefruit that it can send data back to the microcontroller over the TXO pin. This pin is pulled high by default and must be set to ground in order to enable data transfer out! If you do not need hardware flow control, tie this pin to ground it is a level shifted pin, you can use 3-5V logic.
4. RTS - Read to Send flow control pin *out of* the module (Bluefruit LE --> MCU). This pin will be low when its fine to send data to the Bluefruit. In general, at 9600 bauds we haven't seen a need for this pin, but you can watch it for full flow control! This pin is 3.3V out.
5. MOD: Mode Selection. The Bluefruit has two modes, Command and Data. You can keep this pin disconnected, and use the slide switch to select the mode. Or, you can control the mode by setting this pin voltage, it will override the switch setting!  High = Command Mode, Low = UART/DATA mode. This pin is level shifted; you can use 3-5V logic.
6. 5mm Infrared LED, 940nm wavelength (most common), 20-degree beam width, 100 mA continuous, 1000 mA pulse, and 1.6V forward voltage. The LED will be pulsed at 38KHz and focused through a 20mm magnifying glass lens.
7. IR sensor tuned to 38KHz. Left pin is signal, middle pin is ground, right pin is 5V power.
8. A 9 Volt battery will be used to power the Arduino Uno and then the power will be distributed from there to the external sensor and modules.

## 3.3 Software Design

### 3.3.1 Initial Class/Activity Diagram

Each activity corresponds to a screen in our app once the app is open.

### 3.3.2 Final Class/Activity Diagram

Included in corresponding file due to oversized property: pocketPewPortableLaserTagUML.pdf

Details: Vector Imaging. Be advised this file may take a long time to render. However, zoom functions can be used to remedy legibility and clarity issues.

# 4 Implementation

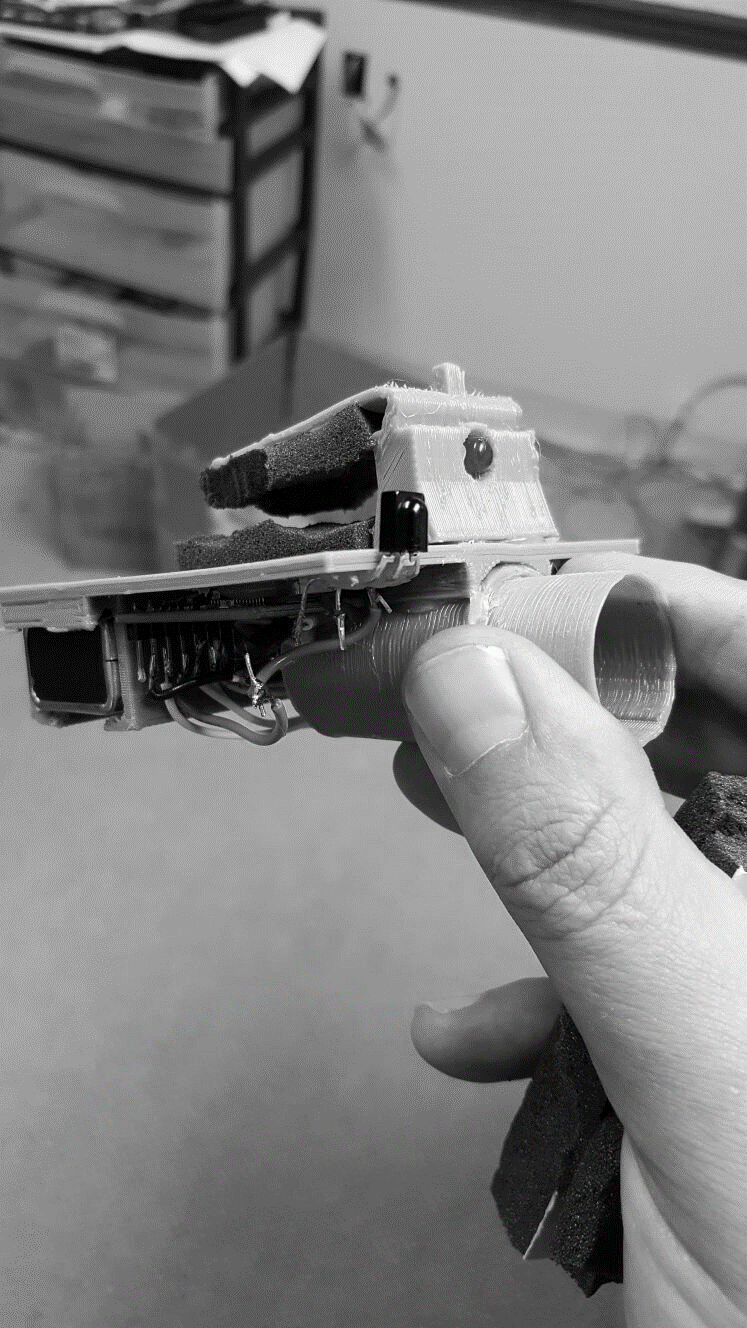
## 4.1 Hardware

Included in corresponding file: Lazer\_Tag\_BlueTooth\_dataMode.ino

Details: All lines are commented with functionality. Lines that our team edited, have # symbols in the line.

Lines of Code inclusive of open-sourced material: 189

Module pictured below:



## 4.2 Software

Included in corresponding file directory: PocketPewPortableLaserTag.zip

Details: All lines are commented with functionality. Lines that our team edited, have # symbols in the line.

Lines of Code inclusive of open-sourced material: 12,915

Files used in our project: README, the following list.

Important Notes: Some open-sourced code is unnecessarily reliant on unused open source files (in our application purpose). However, in our future upgrades to this product, we will continue with a long term optimization process that will include removing these disused files.

Required Files List: Pocket Pew:

Folder res:

    Folder drawable: //# (Number is () means different versions of same image, resizes)

                 //# (Lots of holdovers/optimization necessary in version 2)

        adaflower.png (2)

        ic\_info\_outline\_menu.png(2)

        ic\_settings\_white\_24dp.png(2)

        mqtt\_connected.png(2)

        mqtt\_connecting1.png(2)

        mqtt\_connecting2.png(2)

        mqtt\_connecting3.png(2)

        mqtt\_disconnected.png(2)

        placeholderredbutton.png

        roundedborders\_blue.xml

        roundedborders\_gray.xml

        signalstrength0.png(2)

        signalstrength1.png(2)

        signalstrength2.png(2)

        signalstrength3.png(2)

        signalstrength4.png(2)

Folder layout:

    activity\_color\_picker.xml

    activity\_connect.xml

    activity\_main.xml

    activity\_mqttsettings.xml

    activity\_mqttsettingscodereader.xml

    activity\_options.xml

    activity\_splash.xml

    activity\_uart.xml

    fragment\_ibeacon.xml

    fragment\_uribeacon.xml

    layout\_scan\_item\_child.xml

    layout\_scan\_item\_title.xml

    layout\_uart\_datachunkitem.xml

Folder menu:

    menu\_color\_picker.xml

    menu\_main.xml

    menu\_uart.xml

Folder: mipmap

    ic\_launcher.png(5)

Folder: raw //#(original file names edited in order to fit with Android sound name guidelines).

     //#(ripped from halo 3 sounds; downloaded off of soundfxcenter.com, placeholder sounds)

     //#(edited using mp3cut.net)

    haloempty.wave

    halogrenadedie.wav

    halohit.wav

    halopew.wav

Folder:values

    strings.xml (Lots of defunct strings, need to clean.)

Folder:xml

    preferences.xml

Folder: Manifests

    Android Manifest

Folder: com.laserhawk.pocketpewportablelasertag

    Connect

    MainActivity

    Options

    Splash

    Folder: app

        ColorPickerActivity (Currently disused but depended upon)

        IBeaconFragment

        UartActivity

        UartDataChunk

        UartInterfaceActivity

        URIBeaconFragment

        UriBeaconUtils

        Folder:settings

            MqttUartSettingsActivity

            MqttUartSettingsCodeReaderActivity (Perhaps?)

            PreferencesFragment

            SettingsActivity

        Folder:shortnener

            BitlyShortenerAsyncTask

            ShortenerAsyncTask

    Folder: ble

        BleDevicesScanner

        BleGattExecutor

        BleManager

        BleUtils

        KnownUUIDs

        StandardUUIDs

    Folder: mqtt

        MqttManager

        MqttSettings

    Folder: ui.utils

        DialogUtils

        ExpandableHeightExpandableListView

        ExpandableHeightListView

Folder: dFULibrary //# (More research required, currently disused but depended upon)

    Folder:manifests

        AndroidManifest.xml

    Folder:java

        Folder:no.nordicsemi.android

            Folder:dfu

                Folder:exception

                    DeviceDisconnectedException

                    DfuException

                    HexFileValidationException

                    RemoteDfuException

                    UnknownResponseException

                    UploadAbortedException

                Folder:manifest

                    FileInfo

                    InitPacketData

                    Manifest

                    ManifestFile

                    SoftDeviceBootloaderFileInto

                ArchiveInputStream

                DfuBaseService

                DfuSettingsConstants

                HexInputStream

            Folder:error

                GattError

    Folder:res

        Folder:drawable

            ic\_action\_notify\_cancel.png(2)

            ic\_stat\_notify\_dfu.png(2)

        Folder:values

            strings.xml

## 4.3 Team Dynamic

We implemented a Scrum style team dynamic consisting of traditional scrum meetings. Our methodology for this project was completely Agile in nature. There were virtually four specialised roles however, we implemented pair/group programming during every debugging session even if it was not directly associated with individual specialty. In addition to agile development, rotating leadership was implemented. In each portion of the project schedule, focus on a specific part of the product was headed by the developer whose specialty was associated with it. Associated roles are as follows:

Project Manager: Played the role of end user and client. Assisted in development of hardware code and configuration of hardware components. Stimulated team development. If an individual had a problem, they could question their peer developers for the answer.

Software Developer: In charge of developing the application. Connected user interface to back-end functions. Back-end functions interact with hardware unit to create a cohesive experience.

Hardware Developer: Assembler of hardware and encoded the Arduino controller with interpretation code to receive communication from our software. Responsible for peer work with Software Developers to synchronize user experience effects.

Documenter: Scribe for all project processes including but not limited to: developments of all aspects of the physical product and team interaction.

# 5 Conclusion

## 5.1 Key Developments

We coded our app interface and implemented interaction with the Arduino controller. Adafruit provided open source code for both the Arduino and the Bluetooth module connection. We built upon these codes to support the functions necessary for laser tag. The development of the interpretation in both Android and Arduino, allows seamless communication between the phone and our hardware unit. This code then be updated and compiled with very little cost to our company, by utilizing the existing Bluetooth pairing. Our product also consists of a unique case which was 3D printed from separate atomic open-sourced parts. It has sturdy construction and a universal fit for all smartphones, though Android is currently our only available operating system source. Our application features a very simple design to maximize perfect functionality of basic functions such as our live loss counter. We created a project which allows spontaneous gameplay. We have set up a foundation on which to build new modes. Our future developments will include a web developed portion to contain ranking and battle information, code to support battle over Wi-Fi, new battle modes, and a slimmer design with even stronger components concealed in a case with interchangeable battery compartment.

## 5.2 Market Counterparts and Contributions

Our most prominent competitor is Laser AppTag developed on both Android and iOS by John Atherton. This product includes a bulky plastic gun which connects to the phone and uses a matching app for gun sights and battle mode counters. It is very similar to our app in these ways, however it does not promote spontaneous battle and it draws more power and resources on the smartphone being used. Our product is lightweight in both software and hardware components which allows for longer playing time. Our product is laser safe and doesn’t require firing projectiles at another user. It is designed for simplicity which means the functions are not very complex. Our product utilizes up to date technology and offers a range of cost cutting measures that allow our prices to be scaled down so we can compete with current market conditions. Our app won’t run on older kernels which stimulates purchase of newer models and grows the economy for smartphones. We have a wealth of opportunity for marketable extras which can also generate revenue. We enable family friendly competitions that appeal to a large range of users of all ages and backgrounds. In an age where the modern smartphone can isolate its user, our application and module will bring people together in a social aspect. The effects of our development are the same in both a global and local scale because they draw on basic common features of human psychology. Globally, laser tag is incredibly popular. In the United States, it has been popular for over 30 years.

## 5.3 Vision

Due to the invaluable quality of smartphones as all-in-one devices, our app builds on versatility of the smartphone even with the addition of our module. By using a phone, less extra hardware is needed. We also promote non-violent gaming by taking away the traditional ‘gun’ used in older versions of laser tag. The application can be marketed to anyone using a new smartphone for Android Operating System. We have future plans to contribute to iOS development. We view our product as a successful, stress-relieving device that is a lot easier to use than many of its predecessors. We felt success in creating a clean user experience. Agile development promotes the streamlining of basic functions in user interaction especially. We felt we chose the right path even if it meant less detailed documents. What we plan to improve upon is our understanding of open source code, not only the ones utilized in this project. We plan to improve upon code documentation skills and optimize our project so it only features the files we absolutely need. We will continue work even after the submission of this project, because there is a lot that we can learn from implementing all the proposed features.

# Bibliography

Note: Formatted in APA.

Arduino. (2016). *Arduino*. Retrieved 28 April, 2016, from https://www.arduino.cc/en/Main/ArduinoBoardUno

* + The Arduino Uno description

Arduino. (2016). *Arduino*. Retrieved 28 April, 2016, from https://www.arduino.cc/en/Guide/Introduction

* + Open source Arduino Board description

Arduino. (2016). *Arduino*. Retrieved 28 April, 2016, from https://www.arduino.cc/en/Guide/Environment

* + Arduino IDE description

Android. (2016). *Android*. Retrieved 28 April, 2016, from http://developer.android.com/tools/studio/index.html

* + Android Studio IDE description

Adafruit. (2016). *Adafruit*. Retrieved 28 April, 2016, from https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend

* + Bluefruit Module description

Tutorialspoint.com. (2016). *TutorialsPoint*. Retrieved 28 April, 2016, from http://www.tutorialspoint.com/android/android\_bluetooth.htm

* + Tutorial on making the Bluetooth communicate to Android

Arduino. (2016). *Arduino*. Retrieved 28 April, 2016, from http://forum.Arduino.cc/index.php?topic=2554.0

* + Insight on IR data transfer

Guthrie. (2016). *Lasertagmuseum*. Retrieved 28 April, 2016, from http://www.lasertagmuseum.com/brand/l-p/lazer-tag

* + Examples of present laser tag systems

Lazerrunner.com. (2016). *LaZer Runner Laser Tag Systems*. Retrieved 28 April, 2016, from http://www.lazerrunner.com/How-Laser-Tag-Works.html

* + How laser tag works

Autodesk, Inc. (2016). *Instructables*. Retrieved 28 April, 2016, from http://www.instructables.com/id/Duino-Tagger/

* + Instructable for DuinoTag

Robertson. (2010). *Lasertagparts*. Retrieved 28 April, 2016, from http://www.lasertagparts.com/mtdesign.htm

* + Parts for Laser Tag

Bennett. (2008). *Caltech*. Retrieved 28 April, 2016, from http://alumnus.caltech.edu/~leif/infratag/lens\_choice.html

* + Lense Design

Makerbot® industries, llc. (2016). *Thingiverse*. Retrieved 28 April, 2016, from http://www.thingiverse.com

* + 3D printed base compiled design from a public learning domain

All sound bites for laser tag are the property of Halo 3 plasma gun developed by Microsoft.  
Sound effects © 2016 Microsoft

Retrieved from soundfxcenter.com

# Individual Contributions

Robert DeRosa - Hardware Developer

1. The Idea
2. Oral literature presentation - Hardware section
3. Design plan documentation - Hardware section
4. Hardware requirements - Decided what was necessary to complete the project
5. Hardware controller functionality code - Responsible for the code in the Lazer\_Tag\_BlueTooth\_dataMode.ino file
6. Debugging hardware code and module - Proper interpretation of data passed from the app. Proper execution of hardware features. Timing the sending of different lights and signals.
7. Hardware component diagram - Designing and connecting all hardware components together.
8. Materials provision - Acquired all wires, chips, adhesives, led and ir components
9. Assembly of hardware - Assembled all pieces according to Hardware component layout
10. Final project presentation - Responsible for the following slides: More Technology Used, 3D printing, Following through…, Implementation, Problems in our Hardware, Solutions in our Hardware
11. Arduino sketch documentation

Anthony Russo - Software Developer

1. The Idea
2. Oral literature presentation - Software section
3. Design plan documentation - Application section
4. Application User Experience design - Responsible for how the interface looks to the User, all imaging not associated with Adafruit, and how the app functions from the perspective of an end user.
5. Software interface functionality code - Responsible for the visual representation and responsive portion of the app.
6. Software back-end functionality code - Responsible for the connectivity, sounds, and live calculation in the app. Files used, README, lines of code, comments within code.
7. Materials provision - Provided use of personal Android phone for debugging
8. Debugging software code - Responsible for streamlining and optimizing all code for easily replicable interaction.
9. Final project presentation - Responsible for the following slides: Target Audience, Problems in our Application, Solutions in our Application, Demonstration, Key Developments
10. Android App documentation

Karishma Hirani - Project Leader, Assistant Hardware Developer

1. The Idea
2. Project proposal documentation
3. Weekly progress documentation
4. Oral literature presentation - Overview and Development section
5. Oral literature documentation - Utilized information from the presentation slides of every group member. Responsible for writing the entire document.
6. Design plan documentation - Project schedule.
7. Debugging hardware code and module - Proper interpretation of data passed from the app. Assisted in debugging hardware DFU reset.
8. Materials provision - Purchased lenses, and foam material to grip the clip to the phone. Provided use of personal Android phone for debugging
9. Debugging software code - Assisted in debugging Bluetooth connectivity.
10. Final project presentation - Responsible for the following slides: Our Team, Project Summary, Market Counterparts, Contributions, Conclusion, Credits and Acknowledgements, References
11. Final project documentation - Utilized information from the presentation slides of every group member. Responsible for writing the entire document.

Brian Kancylarz - Assistant Software Developer

1. The idea
2. Oral literature presentation - Communication section
3. Design plan documentation - Software section class diagram
4. Software requirements - Complete application requirements Software Requirements Specification format.
5. Application User Experience design
6. Software interface functionality code - Responsible for the visual representation
7. Materials provision - Provided use of personal Android phone for debugging.
8. Final project presentation - Responsible for the following slides: What we anticipated learning, Design and Design choices, Hardware Design (visual from Robert), Software Design, Technology Used.

X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_               Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_               Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_               Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_               Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_