

**Final Report for the Educational Toy Known as the Phys-Gun  
for Little Toy Blue**

Project Dates

7 March 2011 to 18 April 2011

No Toys  
—  
for  
Old Men

**Team No Toys for Old Men**

William Josh Billingham

Sangwoo Kim

Avi Maxwell King

Benjamin Otto Sipe

Date Prepared: 18 April 2011

Prepared For

Nikola Tesla

Vice President for Innovation

Little Toy Blue

## EXECUTIVE SUMMARY

The purpose of this report is to describe our experiences, results, and recommendations as a result of the production of the Phys-Gun concept.

As a first entry into the educational toy market, Little Toy Blue asked us to create an idea that we could transform into a working prototype and eventually a marketable product. They required that our concept embodied three goals: educational value, profitability and innovation.

Our first step in response to Little Toy Blue's request was to visit toy stores and examine what types of educational toys were already on the market. After research, we drew a conclusion that led us to a specific toy design. We started working on the prototype after receiving the approval from management. In the midst of the project we gave a progress report that detailed all the changes we made since the inception of assembly.

As a consequence of our market research we identified a specific market segment, 8 to 13 year old males, to whom we thought we could successfully market a new toy. Our proposed toy would incorporate Newtonian physics and the excitement that many children have when playing with toy guns. A month of hard work in the development lab yielded a working prototype that fulfilled almost all of our original objectives.

The final prototype has a multitude of capabilities that make it a desirable product for both Little Toy Blue and its customers. It takes user input from a USB mouse and calculates an appropriate trajectory based on a launch angle between zero and ninety degrees. As the user moves the mouse up and down, a dart gun moves accordingly and the VGA displays data and a parabola that corresponds to the projectile motion of the dart.

With only a few minor changes, the Phys-Gun can be mass-produced as a toy that is fun and affordable. We are confident that our prototype accurately represents the potential of the final product.

The experiences and insights that we gained as a result of this project are invaluable. Over the month of work that we put into this toy, we developed a greater understanding and appreciation of teamwork and engineering principles.

## Table of Contents

1. Introduction.....	1
2. Prototype	
2.1 Design.....	1
2.1.1 Motivation.....	1
2.1.2 Original Scope.....	2
2.2 Development Process.....	2
2.2.1 Difficulties.....	2
2.2.2 Successes.....	3
2.2.3 Changes in Scope.....	5
2.3 Final Prototype.....	6
2.3.1 Limitations.....	6
2.3.2 Capabilities.....	7
2.3.3 Goal Fulfillment.....	7
3. Final Product	
3.1 Increased Memory.....	8
3.2 Improved Ergonomics and Aesthetics.....	8
4. Learning	
4.1 Skills Gained.....	9
4.1.1 Digital Logic Design.....	9
4.1.2 Assembly Language.....	9
4.1.3 Circuits.....	9
4.1.4 Technical Communication.....	9
4.2 Lessons Learned.....	10
4.1.1 Communication.....	10
4.1.2 Division of Labor.....	10
5. Conclusion.....	11
References.....	12
Appendices	
Appendix A – Parabola Calculations.....	13
Appendix B – Circuitry Wiring Diagram.....	14
Appendix C – Budget.....	15

# 1 INTRODUCTION

We, No Toys for Old Men, were assigned the task of creating a prototype of a microprocessor-based educational toy. Little Toy Blue emphasized that we carry out the project with focus on innovation, education, and profitability. To address these concerns, we conducted a research on toys available on the market and formulated an idea for a new educational toy. This idea took shape in the form of the Phys-Gun, which introduces children to Newtonian physics without sacrificing fun. We constructed a working prototype of the Phys-Gun over the past month. This document outlines the process and experience of working as a team on a challenging and exciting project.

## 2 PROTOTYPE

### 2.1 Design

#### 2.1.1 Motivation

Based on the market research and several brainstorming sessions, we formulated a design that would meet Little Toy Blue's demands. The image below, Figure 2.1.1, illustrates the conceptualization of our initial design.



Figure 2.1.1: A Computer-Aided Design rendering of the Phys-Gun

The design, called the Phys-Gun, is a toy gun that can be electronically controlled to adjust its initial firing angle and fire foam darts. The toy gun comes with a visual output (not shown) that shows real-time calculations of the projectile motion, including distance traveled, maximum height, and time of flight.

Specifically, our motivation can be broken down into three pieces. First, we want to differentiate our product from those on the market by incorporating the lessons of Newtonian mechanics into our toy. Second, we want to create a system with which users can interact for intellectual

development and fun. Last but not least, we want to create a simple toy that can be easily mass-produced and sold at an affordable price for great profit.

### *2.1.2 Original Scope*

This section briefly discusses how we originally planned to implement our prototype. First, we planned to build a hardware that could electronically control the gun's angle based on a user input (moving of the USB mouse) and electronically fire the gun (one left click of the USB mouse). The gun was to have 90 degrees of motion along the plane of rotation, that is, the plane perpendicular to the horizontal.

To complement the hardware, our team wanted to develop easy-to-use software that would calculate the trajectory using an equation of a parabola and plot a continuous parabola on a VGA output. We hoped to carry out all of these functions in real-time. Of course, the initial velocity was to be given and constant. Further, we were looking forward to add a Quiz algorithm that would test a user by asking for inputs. Not all, but some of the original scope was implemented with some changes. This is discussed in the following section.

## **2.2 Development Process**

### *2.2.1 Difficulties*

During the development of our prototype, we ran into various technical difficulties that we had to overcome in order to finish the Phys-Gun on time. These challenges stemmed from four distinct sources: the numerical limitations of E100, the limited memory of E100, the simplicity of Assembly Language, and the breakdown of multiple servos.

Soon after beginning work on the large software component for the prototype, we realized that we were very limited in how we could manipulate and store numbers in E100. Specifically, we found out that we could not store in memory any non-integer. This initially hindered us from displaying a decimal-containing number and graphing a parabola on the VGA screen. Both of these functions required a level of numeric precision at least to the tens or hundredths place, so rounding each value to the nearest integer was out of the question. Instead, we came up with the solution of scaling. Scaling stored each value as the decimal-containing number multiplied by a factor of 100 or even 1000, in some cases. By scaling a set of values by a constant factor, we were then able to print a decimal point in the appropriate place and display the correct quantity. For example, the value 12.41 might have been stored as 1,241 in memory, but when displayed on the computer screen appeared as "12.41," the correct value.

Another roadblock that we ran into as a result of the numerical limitations of E100 was the graphing of the parabola. Originally, we had planned on graphing the parabola using the general equation  $y = ax^2 + bx + c$ , where  $a$ ,  $b$ , and  $c$  are constants and  $x$  ranges from 1 to 560. However,  $560^2$  by itself was much too large to store in E100. Compounding the problem was the fact that we had to scale up the coefficients "a" and "b," so the product of the term  $ax^2$  in the general parabola equation was over the numeric limit of 32,000 by approximately a factor of  $10^4$  for some large  $x$ -values. After some deliberation, we came up with a two-part solution. The first part

entailed using the derivative of the general parabola equation to graph the trajectory, since the derivative was only dependent on  $x$  instead of  $x^2$ . The second part was that we would only plot the parabola over 56 pixels and then stretch it out over the whole 560. Essentially, the finished graph was a series of 56 dots, each ten pixels apart, that traced out the flight of a projectile from launch to landing.

Besides the numerical constraints of E100, the size of the memory also became an issue in the development of the Phys-Gun. As we added more and more visual data to the project, we found that we were rapidly approaching the end of E100's memory. Finally, we came to a point where we needed to find another source of memory if we wanted to be able to display a title, credits, or instructions screen to the user. This other source of memory presented itself in the form of SDRAM and an SD Card. By utilizing these new resources, we were able to fulfill our goal of providing the user with friendly and visually stimulating gameplay.

Although the simplicity of the assembly language that we used allowed us to implement many different algorithms, it also limited us to a set of very basic operations. For example, there were no built in sine or cosine functions. As a result, we initially struggled to compute the data that we needed to display for every angle (height, range, and time) even though we had all of the necessary knowledge and could do the calculations on paper. Defining any sort of complex image was also very difficult in assembly language. To overcome the simplicity of the language, we ended up enlisting other software programs like MatLab [MatLab] and Adobe Photoshop [Adobe] to do the heavy lifting for us. These programs allowed us to manipulate data and create pictures in an environment that we were comfortable in, greatly facilitating the development process. After creating a picture or a data set in one of these more powerful programs, we could simply copy and paste it into assembly language to include it in the Phys-Gun.

Although most of our problems came on the software front, we also had some bad experiences with servos, the small motors that moved and fired our dart gun. For the first three weeks of work on the prototype, we worked with some servos that Professor Matt Smith kindly lent us. We planned to use these servos in our final demonstration and physical system. Just four days before our presentation, however, both of the servos that we were using suddenly stopped working for no clear reason. The next day, we rushed to visit a local hobby shop to purchase two new servos to replace the duds. Fortunately, we were able to find exactly what we needed and reconstruct our gun before the deadline.

### 2.2.2 *Successes*

In addition to the difficulties that we encountered, we also had several major successes during work on the Phys-Gun prototype. These milestones in development encouraged our team and brought our initial prototype design one step closer to reality. We had three major successes over our month of work: pulse-width modulation, graphing a parabola, and integration of the system.

One of the first things that we had to address for this project was how we were going to move the dart gun through a range of angles. We decided to use a servo, a small electric motor that we could attach to the side of the gun and control with a Verilog module. To move the servo from

one angle to another, we needed to understand how to implement pulse-width modulation. An example of pulse-width modulation (PWM) is shown below in Figure 2.2.1 [Micro Mouse].

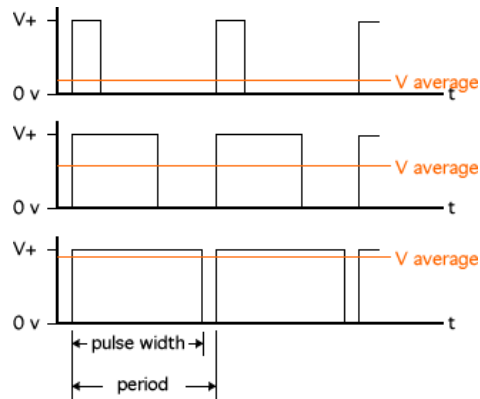


Figure 2.2.1: An example of pulse-width modulation

By varying the width of the pulse, one can easily change the value of “V average” from almost zero to essentially its maximum value. Since a servo changes its position based on the value of V average, it is also dependent on the amount of time that the signal is on for each cycle. After researching the topic online and finding a helpful example [guitarbaka], we were able to nail the implementation of pulse-width modulation in Verilog. By creating a couple of virtual out-ports in the “top.v” file, we were able to give the PWM module an angle calculated in assembly language, which in turn altered V average and moved the servo to the desired position.

Another achievement in the development of the prototype was the graphing of a parabola across the VGA screen. To graph a parabola, we needed to calculate two constants “a” and “b” for every angle between zero and ninety so that we could graph a trajectory defined by the general parabola equation  $y = ax^2 + bx + c$ . Basically, we solved for “a” and “b” in terms of the maximum height and range attained by the projectile, which we could easily calculate using motion equations. For a complete discussion and overview of these calculations, please see Appendix A. Once we had these constants for each angle, we used the derivative of the general parabola equation,  $dy/dx = 2ax + b$ , to plot the parabola. The derivative of a function at a certain point gives the slope at that point, so it is easy to find where the next point on the graph should lie. Using this point-by-point graphing method, we were able to graph a parabola over a maximum of 56 pixels for every angle. Then we “stretched out” these 56 pixels over our entire range of 560 pixels to make an accurate depiction of projectile motion. Figure 2.2.2 displays the parabola that we graphed for an angle of 60 degrees.

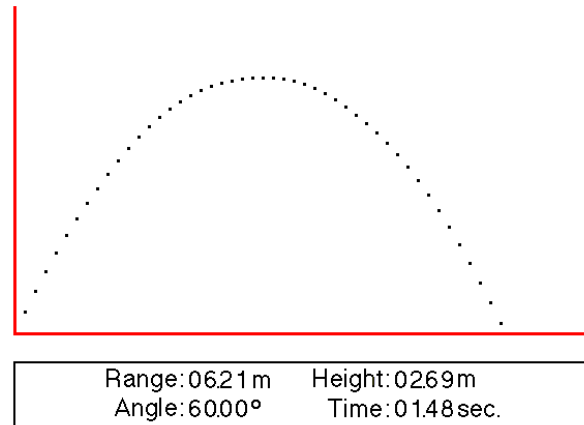


Figure 2.2.2: Parabola Graphed for a Projectile Launched at 60°

After getting the servos to move and a parabola to graph, we successfully integrated both of these functionalities into one cohesive system that updated in “real-time.” At the inception of the project, we were a little worried that the servo would move a noticeable amount of time after the mouse moved up or down and that this “wait time” would detract from the usability of the toy. Once we finally put everything together though, we were excited to see that the response time for the VGA display and the servos was nearly instantaneous. Moving the mouse immediately elicited an appropriate change in position of the servo, changed the graph of the parabola, and displayed new data values for the new angle of the gun. This seamless integration of user, hardware, and software was probably the biggest success of the Phys-Gun. It brought all of the components of the prototype together and allowed a first-time user to easily control the system and benefit from its output.

### 2.2.3 *Changes in Scope*

As a result of the various difficulties and successes that we experienced during the development of the prototype, we made some minor changes in scope to the prototype. We downsized a little bit by deciding to cut the idea of “quiz mode” and graphing the parabola with a series of dots instead of a continuous line. However we also upgraded our scope by adding SDRAM and an SD Card along with multiple splash screens.

In our original scope, we planned on including a “quiz mode” in the prototype. Instead of merely displaying data, quiz mode would prompt a user to guess how far or how high he or she thought a projectile traveled based on a given angle and accompanying trajectory. After some early discussion on the subject, we decided to cut quiz mode from the project scope, mainly because it provided minimal benefit to children. Having not taken any real physics yet, a child between 8 and 13 years old would have no idea how to estimate the height or range of a projectile, quickly turning quiz mode into a guessing game. Also, we thought that there would probably be some mathematical inaccuracy present in the prototype due to the mediocre quality of our dart gun, so the answers to quiz mode would not be physically accurate anyway. We concluded it was undesirable to have a child guess at values that were not truly correct in the first place. Therefore, we decided to drop quiz mode.

Our initial idea for the VGA display included a continuous parabola that defined the trajectory of a projectile launched at a specific angle. After running into problems however, we had to graph the parabola in such a way that it was made up of dots instead of a continuous, curvy line. The specific problems that we ran into are defined in more detail in Section 2.2.1. Despite this setback, the parabola that we ended up being able to graph still satisfied what we initially envisioned. The adjusted parabola lost very little in accuracy and almost nothing in “readability” for most angles between zero and ninety. At the upper end of the spectrum though, the parabolas became a bit unreadable due to the fact that they were made up of only five or six pixels. To accommodate for this we decided to display the parabola corresponding to a launch angle of 83 degrees for any angle between 83 and 90. Despite the fact that our final parabola was discontinuous and slightly constrained, this deviation from our original scope did not have a big negative impact on the prototype. We still succeeded in preserving all of the qualitative learning associated with projectile motion and displaying an aesthetically appealing graph.

An expansion of scope that we had not planned on implementing was the addition of SDRAM and an SD Card. These added memory components allowed us to design multiple splash screens, specifically a title, credits, and instructions screen. Instead of just throwing the user into the gameplay screen as we originally planned on doing, we expanded the scope of the prototype by using SDRAM and an SD Card to display pictures that introduced the user to the game and instructed him or her on how to play.

## 2.3 Final Prototype

### 2.3.1 Limitations

Our final prototype has some limitations. One limitation stems from the limited memory of the E100 processor, which can only store a few significant digits. The limited memory makes calculations with a quadratic equation impossible. Therefore, the toy instead estimates the trajectory based on Euler’s method. This mathematical inaccuracy causes graphical inaccuracy as well; for example, instead of printing a pixel above the one next to it, the E100 might print the pixel at the same level as shown below in Figure 2.3.1.

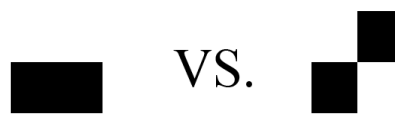


Figure 2.3.1: Pixel discrepancy due to graphical inaccuracy

Because it is impossible to consistently create the result shown on the right of Figure 2.3.1, the toy is programmed to plot the parabola every ten pixels along the horizontal. This process certainly resolves the problem and creates a readily recognizable parabola. The only downside, as shown on figure 3 below, is that the parabola is not fully continuous. The other limitation is partial automation. The prototype cannot cock the gun before it fires, so the gun has to be cocked manually.

### 2.3.2 Capabilities

The prototype also has many capabilities: ninety degree rotation of the gun along the plane of rotation, the electronic firing of foam darts, and real-time virtual display.

The ninety degree rotation is realized with a servomotor; the servo is attached on one side of the gun to rotate the gun. The servo is mounted on a tripod to allow the gun to rotate freely at a higher elevation. Another servo is attached at the bottom of the gun to trigger the firing mechanism; the servo, upon user input, pushes against a small plastic piece of the gun that would fire the foam dart. Lastly, the visual display outputs at real-time a parabola and data of the trajectory based on the current initial angle and velocity as shown below.

### 2.3.3 Goal Fulfillment (*This section seems to lack some technical details*)

We had two goals; the first were the goals set down by Little Toy Blue, and the second were the goals of creating the Phys-Gun prototype. We knew that if we fulfilled all our goals for the Phys-Gun prototype we would be able to also fulfill Little Toy Blue's goals. The goals of the Phys-Gun prototype were to create a physical system that moved the gun up and down and fired it, and to create a virtual display of calculations and the projected trajectory.

In order to complete the physical system we needed to program two servos, one to rotate the gun a full ninety degrees and another that would rotate only about twenty degrees to fire the triggering mechanism. NTFOM's Otto Sipe and Josh Billingham were able to master the circuitry of the servos and the implementation of these servos on the E100 board. They completed the two servo modules in Verilog in the first two weeks and then we put the physical system on hold until the end when we assembled the whole prototype.

The virtual system had many different parts to it. There were a lot of different programs that needed to be written in Assembly language and then all incorporated into one main file. Each member of our team wrote a different part of this main file and then we compiled it all together to render our full virtual system. The final virtual system was able to do calculations and plot a parabola in real time. This was the core of our virtual system.

We were then able to combine both of these systems into a single unit connected through the E100 processor. Overall we completed all the main and important goals we set for ourselves. Through the completion of these goals we also succeeded in completing the goals given to us by Little Toy Blue. Our prototype is a microprocessor based educational toy that is affordable, profitable and innovative.

## 3 FINAL PRODUCT

The marketable version of the toy would be different from the final prototype in several aspects. Below, the discrepancies are discussed in detail.

### 3.1 Increased Memory

The finalized product would have greater memory so that the toy can output more mathematically accurate data and more continuous plots. As discussed previously in Section 2.3.1, the prototype has limited mathematical accuracy because it estimates the trajectory using Euler's Method rather than using an actual equation of a parabola. Increased memory would enable the toy to calculate more accurately with an actual quadratic equation. The increased accuracy would also allow for a more continuous illustration of the parabola. Figure 3.1.1 is an example.

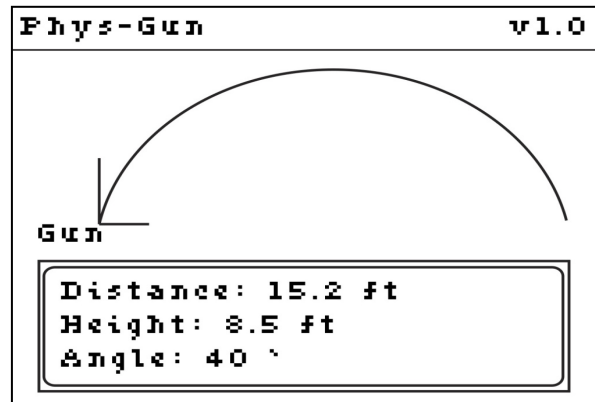


Figure 3.1.1: Final product example screenshot with continuous parabola

As shown above, because of the increased mathematical accuracy, the final product would be able to plot along every horizontal pixel rather than every ten horizontal pixels as does the prototype. The finalized toy would also have an electronic mechanism to complement the manual cocking mechanism mentioned in 2.3.1. In other words, the toy would be able to cock itself automatically.

For aesthetic and ergonomic purposes, the final product would have a joystick like the one pictured in Figure 3.1.2 [Sparkfun] rather than a mouse.



Figure 3.1.2: Final product joystick

The joystick would certainly offer a more user-friendly, presentable interface and greater playability. Also, all internal components such as wires and circuit boards would be hidden within a plastic casing. This would not only make the product more presentable, but also more

durable. Last but not least, all vital components of the toy, including the joystick and the screen, would be integrated into a single unit to make the toy readily portable and easy to use.

## **4 LEARNING**

### **4.1 Skills Gained**

#### *4.1.1 Digital Logic Design*

We gained significant knowledge in logic design by using Verilog hardware description language. The most important component of digital logic design for our project was pulse-width modulation (PWM), which is also discussed in Section 2.2.2. PWM involved using the clock on the Altera DE2 to adjust the pulse width that we gave the servos. As a response to different pulse widths, the servos moved to different angles. We were able to implement pulse-width modulation so successfully that we could move the gun up and down in minute increments. This skill may prove very useful in later EECS classes.

#### *4.1.2 Assembly Language*

We also learned about the Little Toy Blue's assembly language, the ase100. The language offered simple yet fundamental functions such as creation and evaluation of conditionals, arithmetic operations, and creation of functions. Unfortunately, the language did not support built-in libraries as do higher level programming languages and required a very unconventional and unintuitive semantics and syntax. However, our team successfully used the language to program servos and the Altera DE2 board.

#### *4.1.3 Circuits*

A byproduct of working with an electric motor was that we developed a basic knowledge of circuitry. Topics that we encountered include AC to DC power conversion, sharing of ground between circuit elements, and wiring two motors in parallel. For a detailed description and circuit diagram of our complete setup, please see Appendix B.

#### *4.1.4 Technical Communication*

Technical communication was an important component of our learning. Technical communication exposed us to effective and efficient communications methods currently utilized in engineering industries. This helped us to generate important documents such as executive summaries and forewords without sacrificing the overall readability and understandability due to excessive technicality.

## 4.2 Lessons Learned

### 4.2.1 *Communication*

One of the best lessons that we learned as a result of this project was that communication between members is vitally important. We used two types of communication, personal and impersonal, to make sure that everyone in the group was aware of what everyone else was doing and to plan out what work still needed to be done.

Our group communicated impersonally through mediums such as text messaging, online social networking sites, and a site called Dropbox. Dropbox is an online storage system that allowed any member of the group to upload or download files. The website was important because it compiled all of our individual work into one place that could be accessed from any computer at any time of day. We could easily share and edit documents without having to physically meet up. For convenience and flexibility, we also chose to communicate online electronically via the Group feature on the social networking site Facebook. This helped us to coordinate group meetings and to post messages that the whole group could see. As a third method of electronic communication, we used text-messaging to communicate. Texting was the closest thing to talking to someone face-to-face and was often appropriate for urgent questions or conversations that only involved two people.

Although we relied on electronic communication to a degree, group meetings in person were by far the most efficient and productive method of communication. Meetings allowed for fast-paced discussion and brainstorming that really moved the project along in development. They also increased the level of accountability of individual members. Making an excuse for not having something done on time was much easier to do over the Internet than to three other peoples' faces. Generally, we always made big decisions and looked over important documents during a meeting to make sure that everyone understood and supported the final outcome. With everyone in the same place at the same time, we could critique work, explain technical elements, and settle disagreements with a level of ease that was impossible to attain otherwise.

Whether personal or impersonal, communication was important because it prevented redundant work, allowed for individual labor, and provided a platform for questions and help. Without sufficient communication, it is probable that someone would have done work that overlapped with another member's already completed work. Our group avoided this pitfall by establishing clear lines of communication. We could all work individually on different parts of the project because one member always had a good idea of what the other three were doing. However, this does not mean each member always knew exactly *how* to complete his task. At this point, communication became vital to ask for feedback or help from other members.

### 4.2.2 *Division of Labor*

Being able to divide the workload for the prototype in the best way possible was a skill that we developed over the span of the project. As we approached completion, it was apparent that without specialization and abstraction of labor, we never would have come close to finishing the Phys-Gun.

One part of our scheme for dividing work up between group members involved specialization. By making each individual an expert in a certain field, perhaps a field in which he already had experience, we were able to minimize the learning curve for everyone. Instead of making everyone learn everything about the project, we tried to assign a specialized subject area to each member that played to his strengths. For example, Otto Sipe came into the project with an extensive knowledge and interest in graphic design. Accordingly, he completed nearly all of the work that related to displaying visuals on the monitor. This approach to dividing up work streamlined our development process and capitalized on each group members pre-existing skills.

Another technique that we found to be very beneficial was abstraction of labor. Since our project was made up of many different components, it would have been imprudent to try to explain to everyone how each small piece of code worked. By incorporating different levels of abstraction into our project, one member could easily use pieces of code written by another member without necessarily understanding exactly how they functioned. The only knowledge required was the nature of the inputs and outputs of a block of code, no matter how complicated and intricate its inner workings.

## **5 CONCLUSION**

The working prototype is a valid proof of concept for a marketable toy that meets the needs of Little Toy Blue – the Phys-Gun. Our research in the field of current educational toys, experience in producing the prototype, and the feedback we have received to date further strengthen our confidence in this product. Overall, the experience of producing the Phys-Gun prototype has been an educational and intellectually stimulating process for the team members of No Toys for Old Men. We have truly appreciated working for Little Toy Blue and we are looking forward to the launch of the Phys-Gun in the near future.

## REFERENCES

[1] MatLab (v2010a). Distributed and Developed by MathWorks.

[2] Adobe Photoshop CS5. Distributed and Developed by Adobe Systems Inc.

[3] Micro Mouse, Initials. (n.d.). *Dc motors*. Retrieved from <http://www.micromouseinfo.com/introduction/dcmotors.html>

[4] guitarbaka, Initials. (n.d.). *How to create a pulse width modulation in a verilog module to run a motor/servo*. Retrieved from [http://www.ehow.com/how\\_5742129\\_create-module-run-motor-servo.html](http://www.ehow.com/how_5742129_create-module-run-motor-servo.html)

[5] Sparkfun Electronics, Initials. (n.d.). *Arcade joystick*. Retrieved from <http://www.sparkfun.com/products/9136>

## APPENDICES

### Appendix A

#### Parabola Coefficient Calculations

General Equation of a Parabola:  $y = ax^2 + bx + c$

In our case, “c” will always be equal to zero because we want the y-intercept of our projectile-motion trajectory to be zero. Our parabola will also always be opening downward so “a” will be negative. Therefore, our general equation for a parabola is:

$$y = -ax^2 + bx \quad (\text{Eq. 1})$$

We will solve for the coefficients “a” and “b” in terms of the maximum height and range of the projectile, which are easily calculated using equations of motion.

At an x-intercept,

$$0 = -ax^2 + bx$$

$$0 = x(-ax + b)$$

solve for x,  $x = 0, b/a$       $\text{Range} = \frac{b}{a}$  (Eq. 2)

This second x-intercept of  $b/a$  corresponds to the range of the particle. Due to the parabolic nature of the trajectory and the fact that the y-intercept is the origin, the maximum height of the particle occurs at an x-coordinate that is exactly half of the range ( $b/2a$ ). So,

$$\text{Height}_{\max} = -a\left(\frac{b}{2a}\right)^2 + b\left(\frac{b}{2a}\right) = -\frac{b^2}{4a} + \frac{b^2}{2a} = \frac{b^2}{4a}, \text{ solving for } a,$$

$$a = \frac{b^2}{4 \times \text{Height}_{\max}}, \text{ from Eq. 2, } b = a \times \text{Range}$$

Substituting,  $a = \frac{(a \times \text{Range})^2}{4 \times \text{Height}_{\max}}$       $\frac{1}{a} = \frac{(\text{Range})^2}{4 \times \text{Height}_{\max}}$       $a = \frac{4 \times \text{Height}_{\max}}{(\text{Range})^2}$  (Eq. 3)

from Eq. 2,  $a = \frac{b}{\text{Range}}$ , substituting into Eq. 3,  $b = \frac{4 \times \text{Height}_{\max}}{\text{Range}}$  (Eq. 4)

#### Max. Height and Range Calculations

Equations of Motion:

$$V^2 = V_0^2 + 2ax$$

$$V = V_0 + at$$

$$x = V_0t + \frac{1}{2}at^2$$

Division of  $V_0$  into components:

$$V_{0y} = V_0 \sin(\theta)$$

$$V_{0x} = V_0 \cos(\theta)$$

Since the y-component of the velocity is zero at the maximum height of the projectile, we can express the max height as:

$$Height_{\max} = \frac{V_{0y}^2}{2g}, \quad (\text{Eq. 5})$$

To solve for the range, we first need to solve for total time in the air. Since the overall vertical displacement is zero, the initial and final y-components of the velocity are equal and opposite in sign. So,

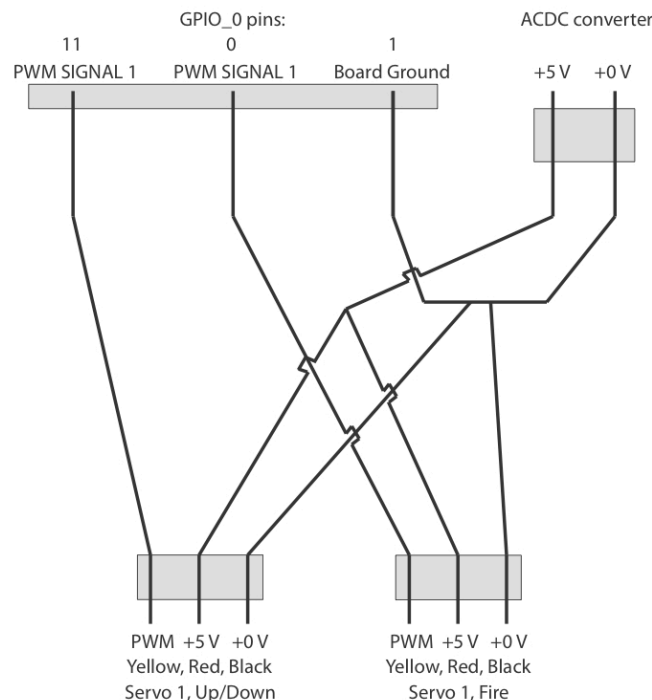
$$t = \frac{2V_{0y}}{g}$$

Because the acceleration in the x-direction is zero, the horizontal displacement (range) is:

$$x = V_{0x}t = V_{0x}\left(\frac{2V_{0y}}{g}\right) \quad (\text{Eq. 6})$$

With the help of MatLab, it is easy to create lists of heights and ranges for every angle between zero and ninety and use these lists to create the “a” and “b” coefficients for every angle.

## Appendix B



This is a simplified version of the wiring used to control the servos from the Altera board and connect them to the power source.

## Appendix C

### Group Spending on Phys-Gun Prototype

Spark Fun Electronics™	
Servo and Battery Pack	\$15.31
Target™	
Tripod	\$8.77
Toys R Us™	
Dart Gun and Darts	\$3.17
Rider's Hobby Store	
New Sevos, Epoxy	\$59.86
Total to Date:	\$87.11
Total Per Member:	\$21.78