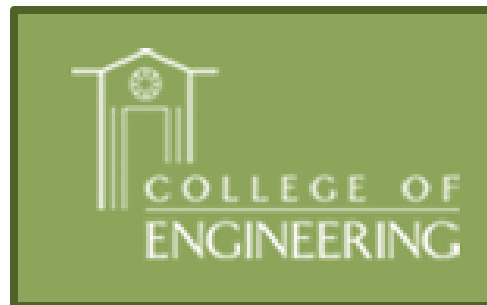


MSU ECE 480

Design Team 3

Final Report
4/23/2014

Ballistic Chronograph



Sponsor: Brian Wright
Facilitator: Dr. Nihar Mahapatra

James Cracchiolo, Nicholas Mancuso, Steven Kanitz,
Madi Kassymbekov, Xuming Zhang

Executive Summary

The ballistic chronograph that our team designed implemented a laser detection scheme to accurately report projectile velocity. The design uses two detection screens: parallel planes that are orthogonal to the path of the projectile. The screens use reverse-biased light emitting diodes with line lasers projected onto them to detect projectiles passing through each screen. The detection logic was handled by a microcontroller, initially the MSP430 but later on the Arduino Due to gather data and calculate results. The prototype described herein will consist of the following dimensions: two 14 inch x 14 inch square screens connected with 30 inches of depth in between them. Wireless data transmission was also implemented as a convenience for the user so all data from our prototype will be reported back over Bluetooth where they can control and save chronograph data. The ability to receive this data can be had with an Android smart phone or other Android device. Additionally, our prototype reports the temperature and relative humidity of the environment with each projectile it detects. These environmental variables were part of the initial specifications required by the sponsor. The data we collected shows that our chronograph accurately reports velocities up to 70 feet per second, however the specifications required accuracy up to 3500 feet per second. The team decided not to validate high speed projectiles due to the risk to the prototype. Mathematically, the chronograph's microcontroller can handle up to 12000 feet per second, and there is no indication that it cannot detect such speeds, other than the lack of opportunity to test it. Therefore, the project was a successful failure, implementing some but not all specifications, and only partially validating the chronograph.

Acknowledgements

Brian Wright

Our sponsor supplied the problem and technical specifications of the project. He was willing help us come up with ideas for the project and was easy to contact anytime we had an update or needed clarification. Brian also helped the team from his position in the ECE Shop, letting us test different components, and providing us with the resources we needed to complete the project.

Dr. Nihar Mahapatra

Dr. Mahapatra was a great facilitator. He met with us every week and made sure the team stuck to our schedule. He helped us solve system level problems when we transitioned from the MSP430 to the Arduino microcontroller. He asked us early into the project to figure out a validation scheme and suggested a radar gun or competing chronograph. He helped us prepare our presentations and how to avoid common pitfalls.

Dr. Gregory Wierzba

Dr. Wierzba helped the team solve an analog problem when we were having difficulties isolating our components. We used his suggestions in the final chronograph, implementing op-amps to separate the laser screens from the microcontroller.

Gregg Mulder

Helped us with our PCB design, and made sure the PCBs were always ready for our design.

Dr. Timothy Grotjohn and Dr. Lalita Udpa

Dr. Grotjohn and Dr. Udpa ran the ECE 480 class, checking our progress at various stages of development, and giving us important lectures on teamwork and power supply design.

Roxanne Peacock

Roxanne ordered all of our parts on time, and helped us track down a critical Arduino shipment at the end of the semester after our microcontroller had burned out. Thanks for your help!

Table of Contents

<u>Chapter 1 – Introduction and Background</u>	4
<u>Chapter 2 – Exploring the Solution Space</u>	9
<u>Chapter 3 – Technical Description of Work Performed</u>	15
<u>Chapter 4 –Test data with Proof of Functional Design</u>	34
<u>Chapter 5 – Final Conclusion and Future Design Improvements</u>	43
<u>Appendices</u>	47

Chapter 1 - Introduction and Background

A ballistic chronograph is a device used to measure the velocity of projectiles. Our chronograph is designed to assess the velocity of projectiles mid-flight and report the readings back to the user. In order to achieve these measurements, chronographs have to implement a detection scheme to determine when the projectiles have entered the area of interest and calculate distance traveled divided by time. Our solution is able to accomplish this detection through line lasers and reverse-biased LEDs as emitters and receivers, respectively. Methods of detection by current chronographs include infrared light detectors and photo detectors.

The chronographs available for purchase vary considerably in price. The higher end chronographs are very sophisticated and report velocity with high precision and accuracy. The less expensive chronographs are less accurate with many weaknesses. One of the main objectives of this project was to explore different detection schemes and create a chronograph that was more reliable than the consumer grade chronographs. Examples of current market chronographs are as follows:



Figure 1. Caldwell Ballistic Precision Chronograph

The Caldwell Ballistic Precision chronograph, as shown in Figure 1, is priced at \$99 and uses photo-detectors to determine when the projectile has crossed each screen. This chronograph cannot function indoors or in the dark because it relies on ambient sunlight. When a projectile passes through the screen as shown in Figure 1, the chronograph detects the projectile due to the change in luminescence across its photo-detectors. This chronograph can report velocity back to the user over a

physical wired connection to a smart phone or can be displayed on the local monochrome LCD screen. It can read between 5 and 9,999 feet per second. [3]



Figure 2. CED M2 Chronograph Set

The CED M2 Chronograph Set, as shown in Figure 2, can report velocities between 50 and 7,000 feet per second. This chronograph is priced at \$199 and uses photo-detectors and ambient light to determine projectile velocity. These photo-detectors become inaccurate in darker conditions. It has built-in storage to save up to 1000 recordings, and can perform calculations for minimum, maximum, and average velocities. This chronograph can be connected to a computer to access all of this data by supporting USB capability. [4]



*Figure 3. The Pact Professional XP
Chronograph w/ IR Sensors*

The Pact Professional XP Chronograph, as shown in Figure 3, costs \$269.99 and comes with infrared sensors to detect passing projectiles. This chronograph reports data based on user input on peripheral terminal. This chronograph can only detect bullets. By telling the chronograph what kind of bullet you are firing it will report an accurate velocity. It maintains a database of “bullet coefficients” that contain bullet-specific drag over time. It can report data usage on the LCD terminal and has a built in printer to report recorded velocities. This printout includes additional information such as environmental pressure and temperature. The infrared sensors allow the chronograph to work during cloudy days, but extremely bright conditions will give the sensors false detections. By removing the infrared sensors, the chronograph will use photo-detectors instead which allows it to work in sunny conditions with ambient light, but will struggle in overcast. Due to the process of manually having to change the sensing mode of the chronograph as well as a lack of remote data reporting, this chronograph requires a lot of physical interaction with the user during its operation. [9]

The primary goal of this project was to create a low-cost, low-power ballistic chronograph that can report accurate velocities regardless of environmental conditions. The prototype also had to be constructed from materials that would be less expensive than those currently available which were achieved by exploring the use of inexpensive hardware components as well as open-source solutions. A secondary goal was to determine the effectiveness of lasers as a detection alternative to photo-detectors and infrared sensors. Strengths of the proposed design include:

Effective in dynamic lighting conditions

The commercial chronographs struggle to report velocities accurately in non-ideal conditions. By choosing a detection scheme separate from, or in addition to, photo-detectors or infrared sensors, we can alleviate the need for a static ambient light. Our chronograph was designed to be used outdoors, indoors, and with or without ambient lighting. Chapter 2 of this report explains the solution space we explored such as using laser based detection and/or ultrasonic sensors both of which remove the requirement for ambient light.

Wireless Communication

Current budget chronographs are clunky because they are best used when the projectile is mid-flight, but the velocity is displayed on the chronograph itself. Our chronograph should report data back

to the user wirelessly so that he/she can continue chronograph operation without any need for physical interaction.

Easily modified

By making the detection and calculation phases of the chronograph modular, either part can be modified or improved without adjusting the other. These can be attained through either changes in the physical design or software. This gives the prototype flexibility and allows it to reach a broad market.

Any solution that branches away from photo-detectors would be unique to the market. Should this design succeed, using low-cost and open-source solutions, it would be easier and more effective for potential chronograph customers to implement our design. This project would have immediate application in the Do-it-yourself community, but with a custom smart phone application and additional modification with a faster microcontroller, this could have application in the academic and consumer worlds.

The total budget to build a chronograph described in this report would be \$190. A full explanation of our budget, including expenses on experimental components and test equipment, can be found in Chapter 5.

Chapter 2 – Exploring the Solution Space

The Critical Customer Requirements for the ballistic chronograph were established in the first meeting with our sponsor. The following list is organized into primary and secondary specifications for the prototype chronograph.

Primary Specifications:

- *Projectile Velocity* → The Chronograph should be able to detect and accurately report velocities up to 3500 feet per second.
- *Wireless Reporting* → The user should be able to receive data from the chronograph without having to interact with it.
- *Data Storage* → The user should be able to access and store the data acquired by the chronograph.

Secondary Specifications:

- *Environment Variables* → The chronograph should record environment variables relevant to projectile flight such as: temperature, humidity, pressure and altitude.
- *Detection Scheme* → Explore the effectiveness of a laser-based detection system
- *Portability* → The chronograph should be portable and have a dedicated power supply

The chronograph is divided into two primary functions: the analog hardware detection scheme and the reactive software on the microcontroller. Additional secondary functions such as peripheral sensor data and user-readable output is controlled by hardware waiting for commands from the microcontroller. The FAST Diagram below, Figure 4, explains the relationships between these primary and secondary functions.

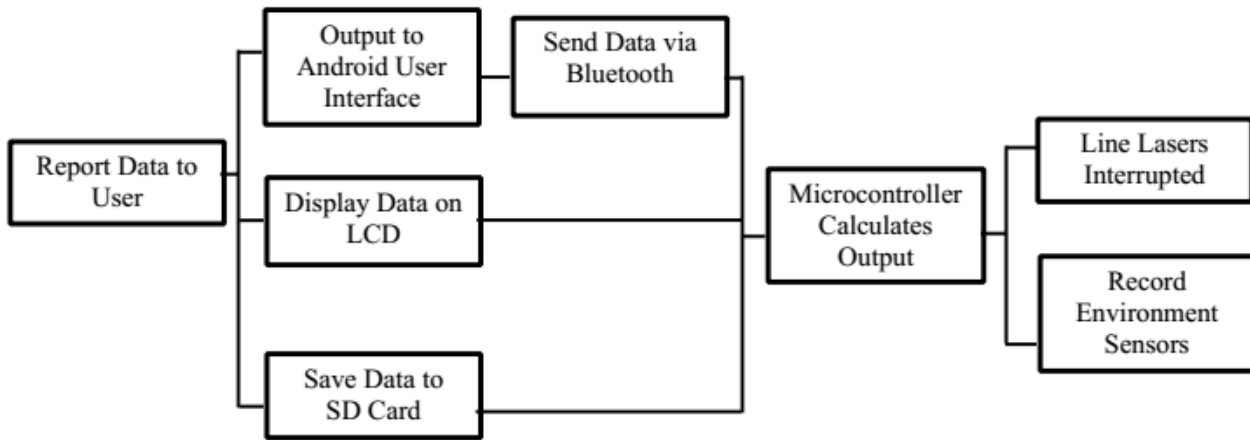


Figure 4. FAST Diagram

Any solution to this problem would need to implement screens. A screen is defined as a 3-D plane orthogonal to both the ground and the projectile of interest. Figure 5 below shows the side view of a two screen chronograph system.

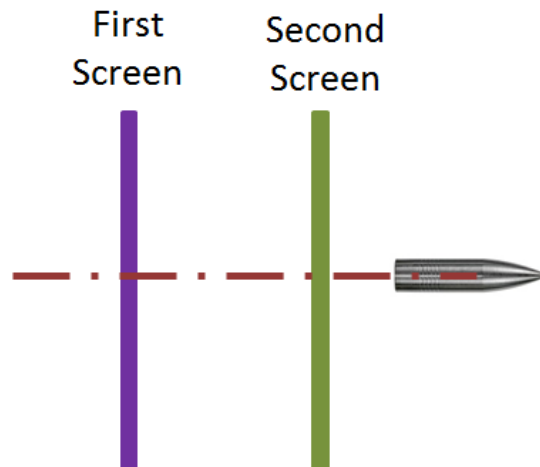


Figure 5. Two Screen Detection System

Based on the FAST diagram and customer specifications, there were three solutions that rose to the surface: Infrared sensors, ultrasonic sensors, and laser-based detection.

Infrared Sensors

The main advantage to using infrared sensors would make the design closer to the commercial products already on the market. This design could be improved in other areas to give the user of the device greater feedback of surrounding conditions, in addition to basic ballistic information, such as: temperature, humidity, altitude, and pressure. This information could then be sent wirelessly to a smartphone via Bluetooth for future reference. The concepts in this design not specific to detection could be applied to any chronograph system.

Ultrasonic Sensors – Doppler Effect

A second potential design scheme could use ultrasonic sensors spread out over a large area to track the projectiles via the Doppler Effect. With this method a radar signal at a certain known frequency would be transmitted over an area and reflected off the projectile. The returned signal would then be examined to determine the change in frequency compared to the original signal. Knowing the change in frequencies would allow for the calculation of the projectile velocity. The advantage to this design is that the projectile would not need to be fired through such a small area as in the previously discussed designs. The disadvantage is that this design would be much harder to implement since it would exist on a much larger scale. It would not reflect the portability specification and would deviate from the two screen system. It would also require more sensors to implement and more complex software. Depending on how large an area was being monitored, it is possible that multiple microcontrollers would need to be used, introducing timing and synchronization as a potential hazard. Prototyping the Doppler Effect system would be a much larger challenge, and a much greater investment than an infrared solution.

Laser Detection

A laser detection scheme would comprise of lasers as an emitter and reverse-biased LEDs as a receiver. When the laser was interrupted by a projectile, the LEDs would no longer be receiving a signal, and the semiconductor inside would not produce a voltage. Blocking an LED is this

In order to determine the best solution, we ranked each according to cost, complexity, experimental time, reliability, and fulfillment of critical specifications. The table below shows the results of our solution matrix.

Design	Cost	Complexity	Experimental Time	Reliability	Critical Specifications	Overall Rank
Line Lasers	1	1	1	1	1	1
IR Sensor	1	1	1	2	2	2
Doppler Effect	2	2	2	3	2	3

Table 1. Solution Matrix

After determining that using laser based detection would be the best solution for our given criteria, we broke down different potential laser systems. Researching laser detection and reverse-biased LEDs, we came across a do-it-yourself laser alarm system designed for use with the Texas Instruments MSP430 Launchpad. [7]

Initially, the plan was to extend this analog laser detection system into multiple lasers and LED pairs to create a detection grid. After researching the cost of lasers, the team determined that a solution requiring a large amount of lasers would be too expensive, and fail to meet the cost requirement of the project. The next idea was to pass the light through a lens to diffuse the light and cover multiple LEDs with a single laser. After discussing this possibility with our sponsor, it was suggested that we look into line-lasers: point lasers manufactured with a divergent lens. This led to line-lasers being incorporated into the design solution where a single laser would power multiple LEDs. Figure 6 shows the relationship between a point laser and a lens to create a line laser, and Figure 7 shows a line shining across a human hand.

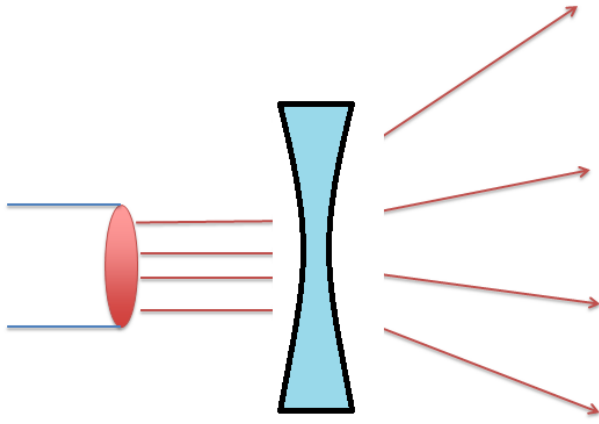


Figure 6. Divergent Lens



Figure 7. Human Hand and Line Laser

After receiving the initial line lasers for experimenting, the team tested the line lasers against a bank of reverse-biased LEDs on the oscilloscope to verify the validity of the solution. The oscilloscope could differentiate between blocked and unblocked laser readings, so the team continued to implement the laser based detection scheme.

The initial estimate of the solution was \$175. The table below shows the predicted breakdown of costs for the prototype.

	Estimated Budget		
	Part	Quantity	Cost
	Line Laser	2	\$20.00
	Sensors*		\$30.00
	SD card	1	\$10.00
	LCD Screen for microcontroller	1	\$10.00
	Projectiles for Testing		\$20.00
	Screen Frames		\$30.00
	LEDs		\$5.00
	Additional Unknown Expenses		\$50.00
Total:			\$175.00

Table 2. Estimated Budget

Figure 8 below shows the Gantt chart used to plan and schedule the semesters' workload. Every person was given technical responsibilities. These technical roles can be found in more detail under Appendix 1 of this report.

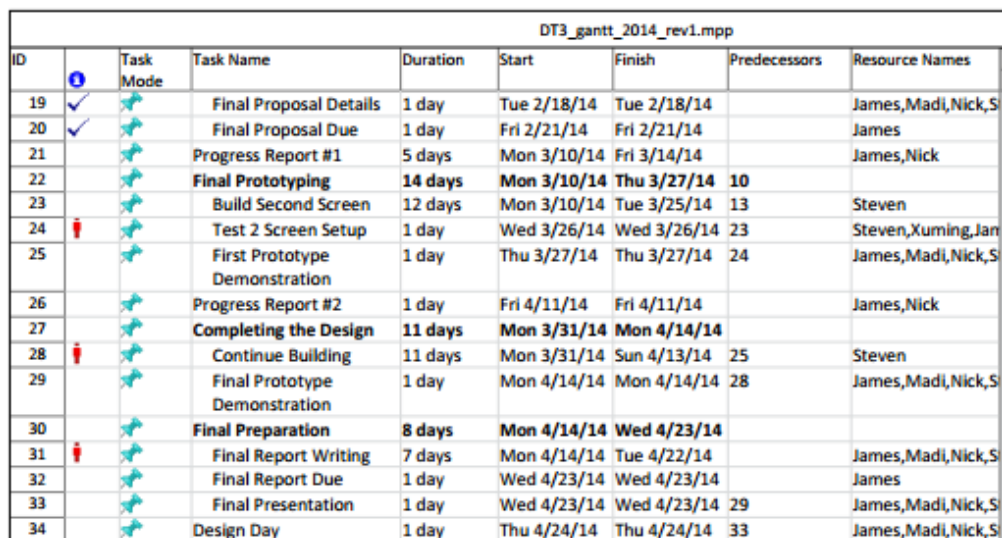
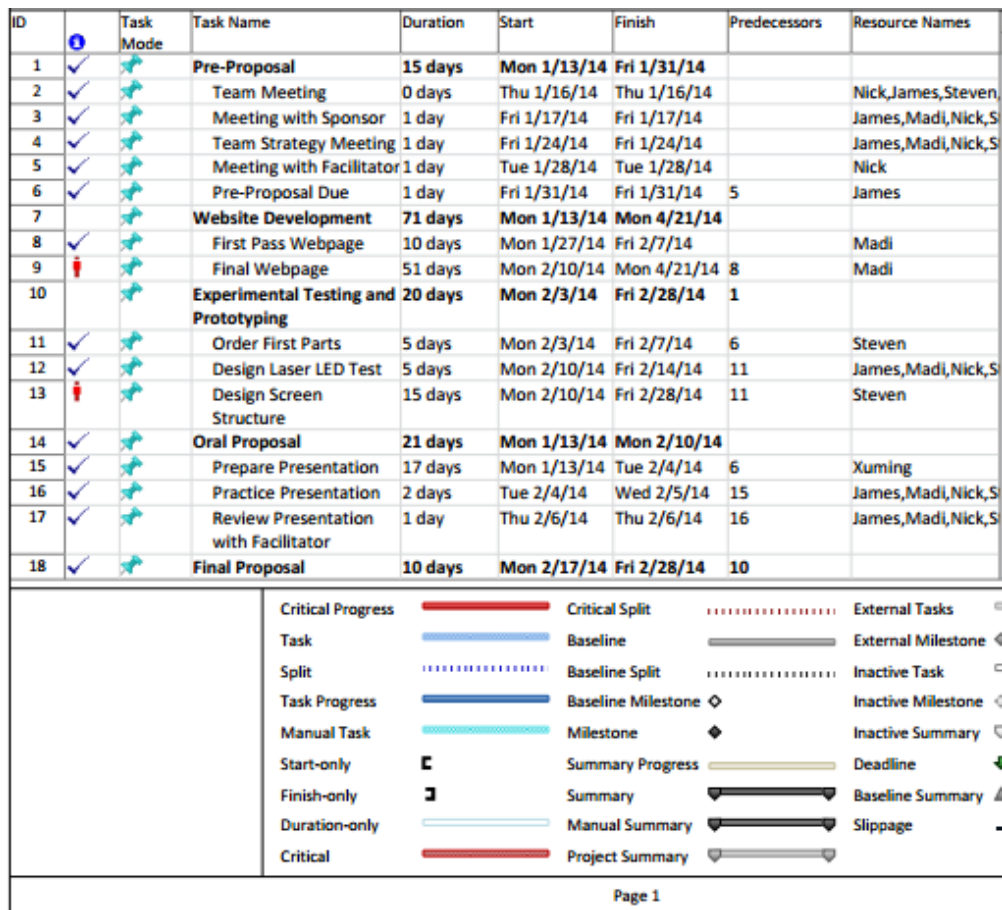


Figure 8. Gantt Chart

Chapter 3 – Technical Description of work performed

The prototype chronograph went through many phases of development and testing including the testing of two different microcontrollers and a variety of peripheral hardware such as sensors. The design effort included: analog development, digital hardware development, software development paradigms, and the evolution of each subsystem.

Choosing the MSP430

In order for the chronograph to perform its calculations, we initially chose the Texas Instruments MSP430 Launchpad to be used in our design. The MSP430 also provided us with an analog-to-digital converter which was instrumental in implementing an analog detection system. We chose this due to the familiarity of working with the MSP430 in the ECE 480 lab as well as the example alarm system we discovered during the initial research phase along with copious online software examples. [8] Figure 9 below shows the TI MSP430 Launchpad.



*Figure 9. TI MSP430
Microcontroller*

Prototyping and Verifying the Laser Screen

The first analog design decision the team made was to use line lasers and reverse-biased LEDs configured as emitters and receivers, respectively, to create effective laser screens that work in any lighting conditions. The line lasers were powered off of 5 Volts at 2 milliamps. The LED banks were constructed from many reverse-biased LEDs in series, put in parallel with 32 M Ω of resistance. The first iterations of development were on a protoboard, seen in Figure 10 below.

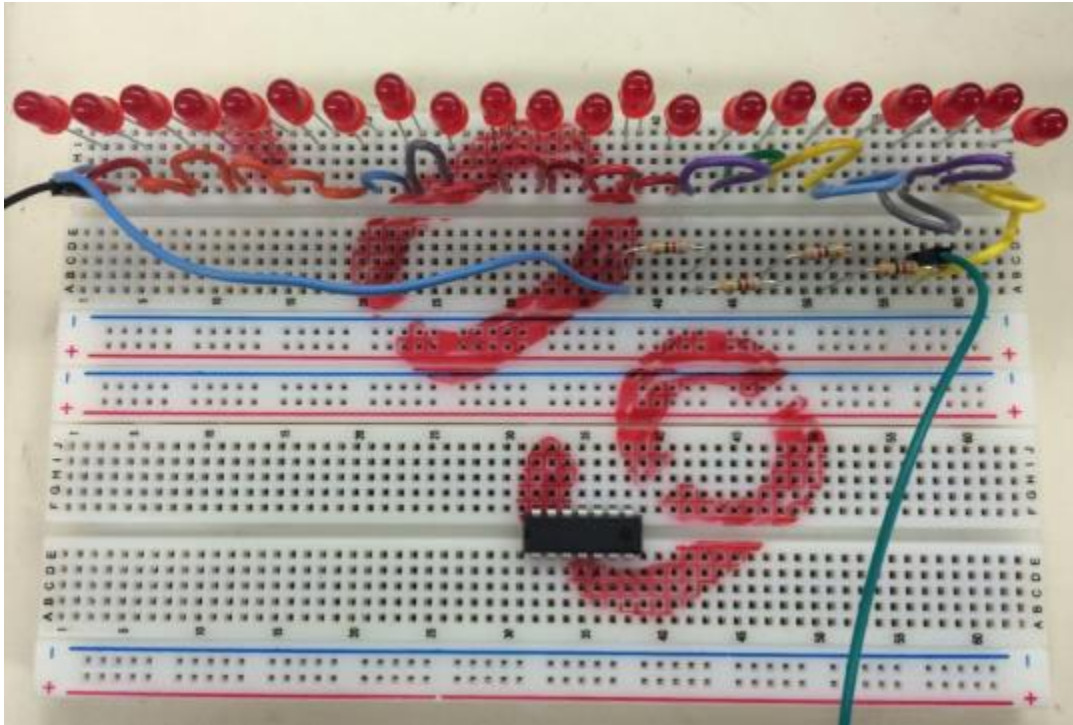


Figure 10. Protoboard Prototype LED Bank

After creating a protoboard version of a screen, the team attached them to the MSP430 microcontroller to determine whether or not the on-chip analog-to-digital converter could detect the difference between laser-active and laser-interrupted circuits. The lasers powered the semiconductors within the LEDs to generate a voltage and that signal was being sent into the microcontroller. We used our hands to block the laser which lowers the amount of voltage being generated by the LEDs and wrote a program to control an on-chip LED to verify that the microcontroller could detect the difference between the blocked laser and unblocked laser states. Figure 11 below shows the code for this initial, single screen protoboard test.

```

1/*
2 * main.c
3 *
4 * Created on: Feb 26, 2014
5 * Author: mancuso8
6 */
7
8#include <msp430g2553.h>
9volatile long voltageRaw;
10double convVoltage;
11
12void main(void)
13{
14    WDTCTL = WDTPW + WDTHOLD;
15    BCSCTL2 = BIT5 + BIT4 + BIT2 + BIT1; //MCLK and SMCLK
16    BCSCTL3 = BIT5; //ACLK
17    ADC10CTL1 = 0x0004; //select A0 input channel and repeat single channel
18    ADC10AE0 = BIT0; // enable A0
19    ADC10CTL0 = ADC10ON; //set reference voltage, turn on and setup ADC10 and MSC
20    _delay_cycles(5); //delay 5 cycles
21    ADC10CTL0 |= ENC + ADC10SC; //enable ADC10 and start conversion
22
23    //enable pldir
24    P1DIR = BIT6; // set the green LED to OUTPUT DIRECTION
25    while(1)
26    {
27        voltageRaw = ADC10MEM; //get the voltage
28        convVoltage = ((double)voltageRaw/440) * 8; //convert to range from 0 to 2.5 volts
29
30        if (convVoltage > 700) {
31            P1OUT = BIT6;
32        }
33        else {
34            P1OUT = ~BIT6;
35        }
36    }
37}
38
39
40

```

Figure 11. MSP430 Detection Code

Extending the above test, the microcontroller was programmed with a two-loop state machine to verify that the microcontroller could determine that a single screen had been interrupted twice. This experiment allowed us to simulate the final prototype functionality without having to construct a second screen. Figure 13 below shows the single screen, double interruption state machine code.

```

1 #include <msp430g2553.h>
2
3 /* mancuso8 - work for DT3 ECE480
4  * main.c
5  */
6
7
8 #define LED0 BIT0//red
9 #define LED1 BIT6//green
10
11 unsigned int value=0;
12 float laser=0;
13 float input_voltage = 0;
14
15 void ConfigureAdc(void)
16 {
17     /* Configure ADC Channel */
18     ADC10CTL1 = INCH_5; // Channel 5, ADC10CLK/4
19     ADC10CTL0 = SREF_1 + ADC10SHT_3 + ADC10ON + ADC10IE + REFON + REF2_5V; //VCC & VSS as reference
20     ADC10AEO |= BITS; //P1.5 ADC option
21 }
22
23
24 void main(void)
25 {
26     WDTCTL = WDTPW + WDTHOLD; // Stop WDT
27     BCSCCTL1 = CALBC1_16MHZ; // Set range
28     DCOCTL = CALDCO_16MHZ;
29     P1DIR |= LED0 + LED1;
30     P1SEL |= BITS; //ADC Input pin P1.5
31     P1OUT &= ~(LED0 + LED1);
32
33     ConfigureAdc();
34
35     //__enable_interrupt(); // Enable interrupts.
36     __delay_cycles(20000);
37     int i = 0;
38     for (i = 0; i < 100; i++) {
39         ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
40         value += ADC10MEM;
41         __delay_cycles(20000);
42     }
43     value=0;
44     for (i = 0; i < 10; i++) {
45         __delay_cycles(10000);
46         ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
47         value += ADC10MEM;
48     }
49     laser = (float)(value/10)*2.5/1023;
50     //first screen
51     while(1)
52     {
53
54         __delay_cycles(2000); // Wait for ADC Ref to settle
55         ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
56         value = ADC10MEM;
57         input_voltage = (float)value*2.5/1023;
58         if (input_voltage > /*laser-.21*/.9) //laser is on the LEDS //1.37
59         {
60             P1OUT &= ~LED1;
61             P1OUT |= LED0;
62         }
63         else //laser has been blocked
64         {
65
66             P1OUT &= ~LED0;
67             P1OUT |= LED1;
68
69             // time stamp
70             __delay_cycles(10000);
71             break;
72         }
73     }
74     for (i = 0; i < 40; i++) {
75         ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
76         value = ADC10MEM;
77         __delay_cycles(200000);
78     }
79     //second screen
80     while(1)
81     {
82         __delay_cycles(2000); // Wait for ADC Ref to settle
83         ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
84         value = ADC10MEM;
85         input_voltage = (float)value*2.5/1023;
86         if (input_voltage > .9) //laser is on the LEDS //.6
87         {
88             P1OUT &= ~LED0;
89             P1OUT |= LED1;
90         }
91         else //laser has been blocked
92         {
93             // time stamp
94             __delay_cycles(1000);
95             break;
96         }
97     }
98     while(1)
99     {
100         P1OUT ^= LED1;
101         __delay_cycles(16000000/4);
102     }
103 }

```

Figure 12. MSP430 Two Screen Detection

Finalizing the Laser Screens

After successfully verifying that detection was possible with the laser screen concept, we transitioned from protoboard testing to PCB testing. The PCBs let us place more LEDs in a smaller area, increasing the resolution of detection. This first testing was done on a small PCB strip at a length of about 2.5 inches. Figure 13 below shows the first pass PCB schematic for the laser screens, balanced with half the prototype resistive load, 16.4 M Ω .

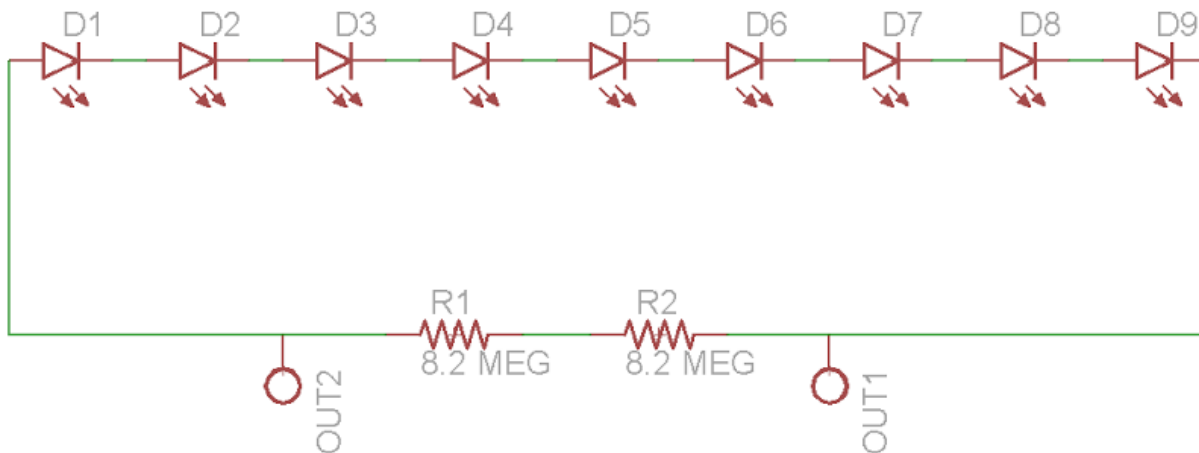


Figure 13. 1st PCB Schematic

Figure 14 below shows the PCB layout equivalent of the first pass PCB LED screen.

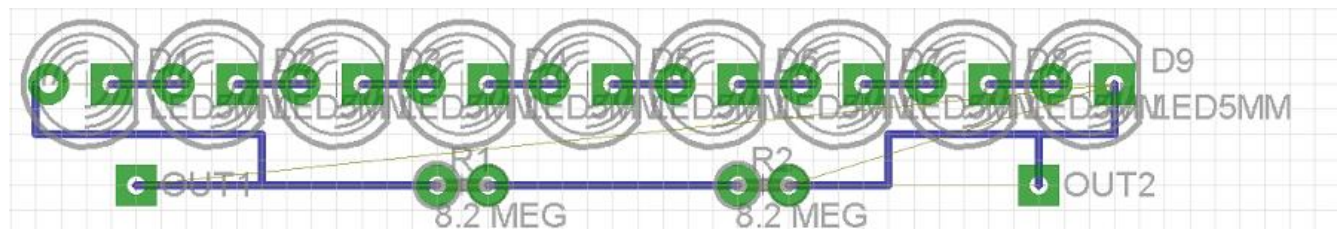


Figure 14. 1st PCB Layout

The final version of the laser screens used bigger PCBs at about 5.75 inches so that each complete laser screen could be created from two PCB LED banks. The figures below show the PCB designs, and the final prototype screens.

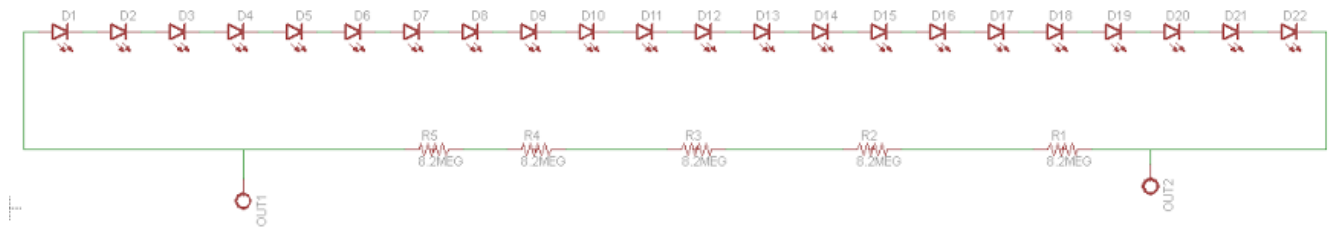


Figure 15. Final PCB Schematic

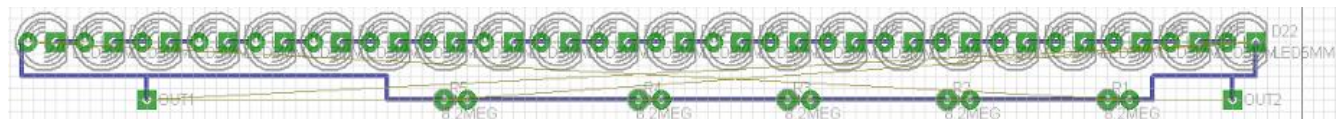


Figure 16. Final PCB Layout

Bluetooth Module

Bluetooth was explored as a way for wirelessly sending the data to the user. The ITEAD Bluetooth booster pack was an affordable solution that was compatible with the Texas Instruments MSP430 Launchpad. This sensor was configured by looking at the schematic and connecting the appropriate pins. It was designed to use the TX and RX serial communication pins with the MSP430 to send and receive data. On the user end, an Android application receives the Bluetooth messages and

displays the chronograph data to the user. The following figures, 17 and 18, show the ITEAD Bluetooth Booster pack and its pin configuration diagram, respectively. [6]

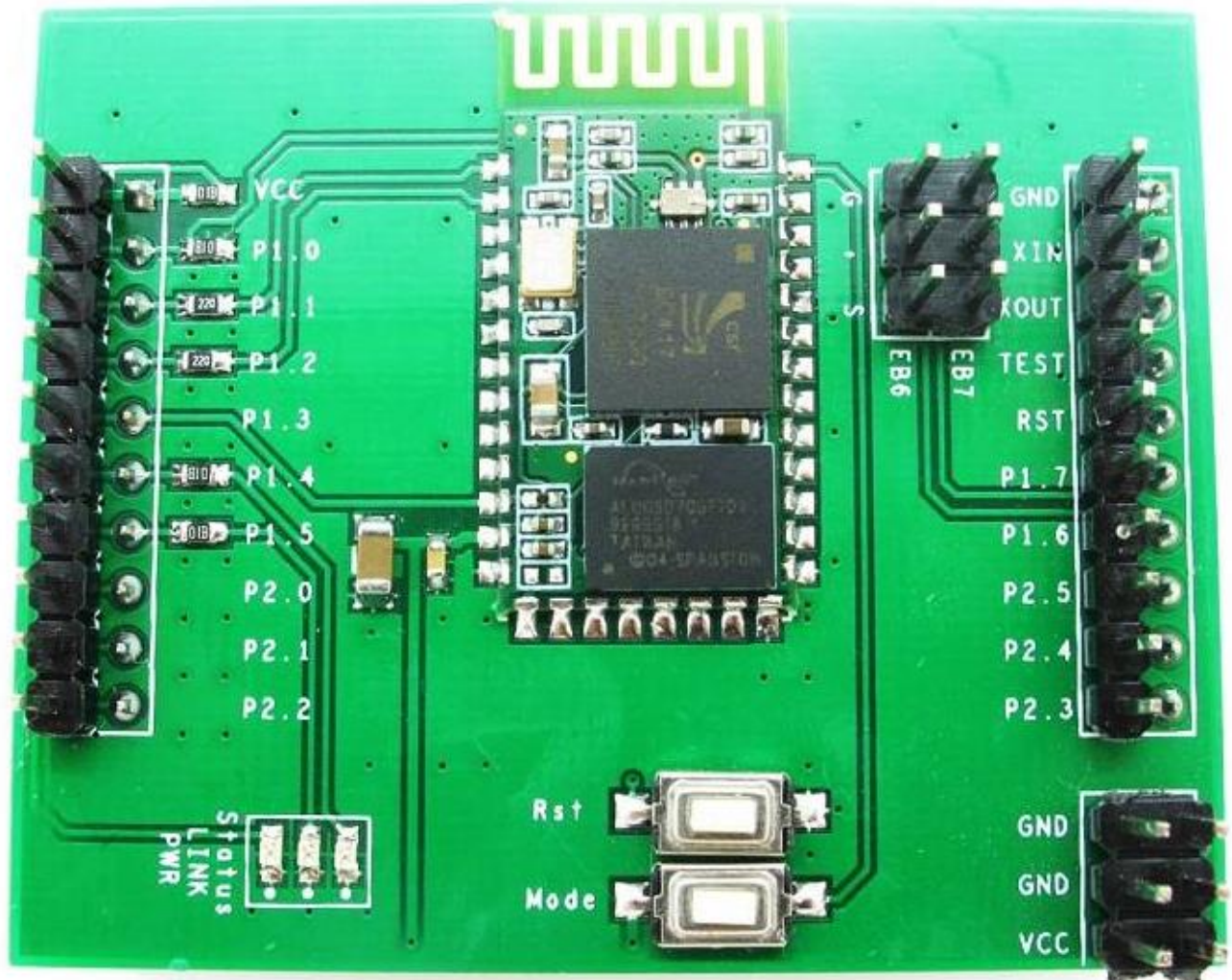


Figure 17. ITEAD Bluetooth Booster pack



MODE	P1.0
TX	P1.1
RX	P1.2
BT_Ret	P1.3
Status	P1.4
Link	P1.5
EB6	P1.6
EB7	P1.7

Selecting the sensors

Two sensors were selected for use with the chronograph: a temperature and humidity sensor, and an altitude and pressure sensor. These sensors were designed for an Arduino. Using the datasheet, we were able to figure out how many I/O pins the sensors would use on the MSP430. Software controlling these sensors was not created until later in the chronograph's development. The following figures are the temperature and humidity sensor, and the air pressure and altitude pressure.



Figure 19. Altitude and Pressure Sensor



Figure 20. Temperature and Humidity Sensor

LCD Screen

An LCD Booster pack by RobG was selected because it was compatible with the MSP430 and was supported by a custom programming library. Shortly after connecting this screen to the microcontroller we realized that it required the majority of the programmable input/output pins of the MSP430. Between this screen and the aforementioned sensors, we realized the MSP430 would not have the analog or digital I/O we need. This was the first major design mistake we made, not properly investigating our total microcontroller capability and hardware choices. Figure 21 below shows the LCD screen we purchased for use with the MSP430. [10]

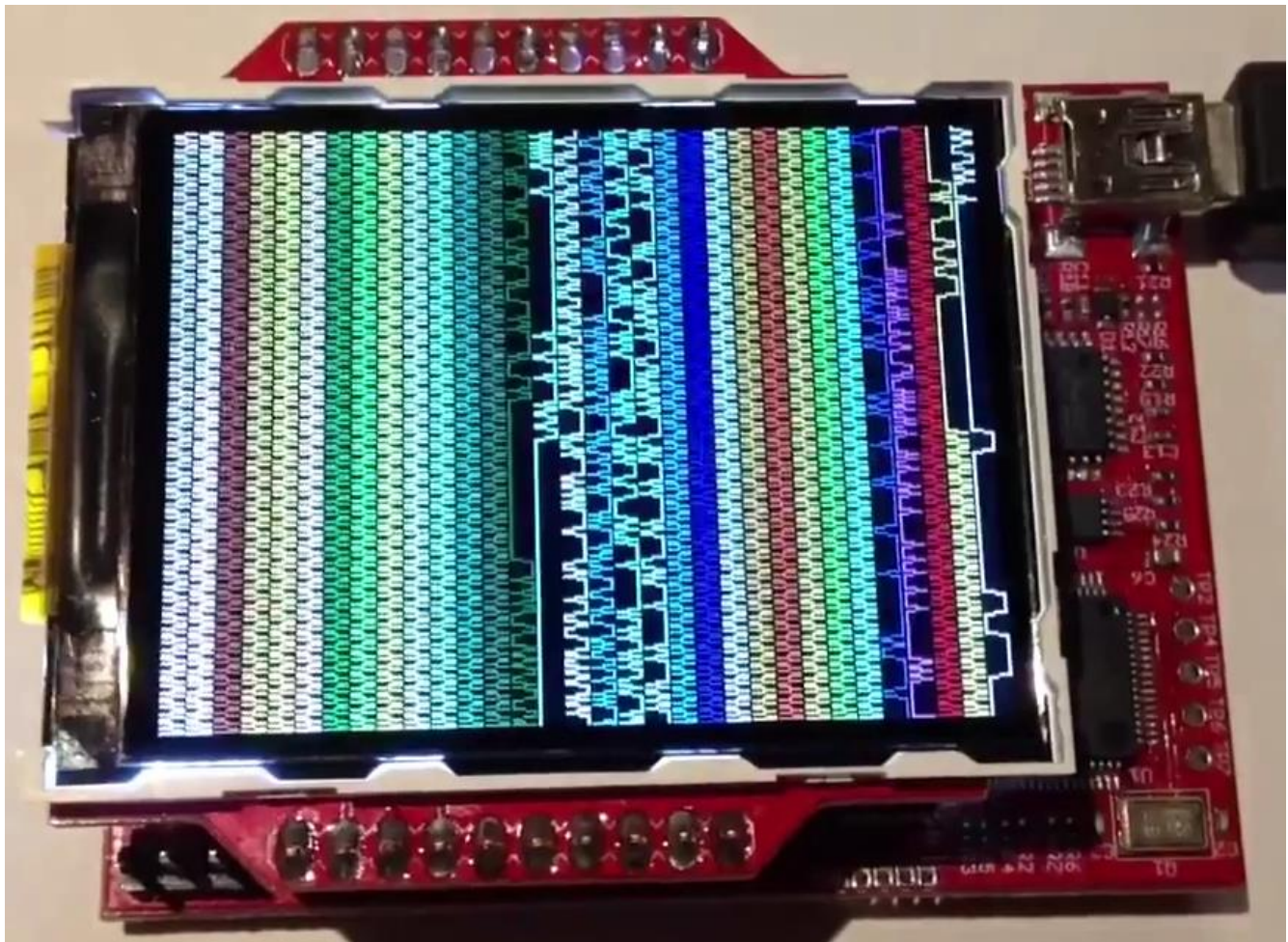


Figure 21. LCD screen for MSP430

Transition to Arduino Due Microcontroller

After realizing that the MSP430 would not supply us with enough pins to connect all of our peripheral hardware, we switched microcontrollers to the Arduino Due. This change provided us with additional advantages: faster on-chip processor, increased number of programmable input/output pins, and immediate connectivity to our sensors. The only immediate disadvantage was losing the LCD screen that was designed for the MSP430. We selected the Due instead of other microcontrollers because it has an 84 MHz clock speed instead of the standard 16 MHz found on the MSP430 and other similarly priced Arduino microcontrollers. Figure 22 below shows the Arduino Due microcontroller. [2]

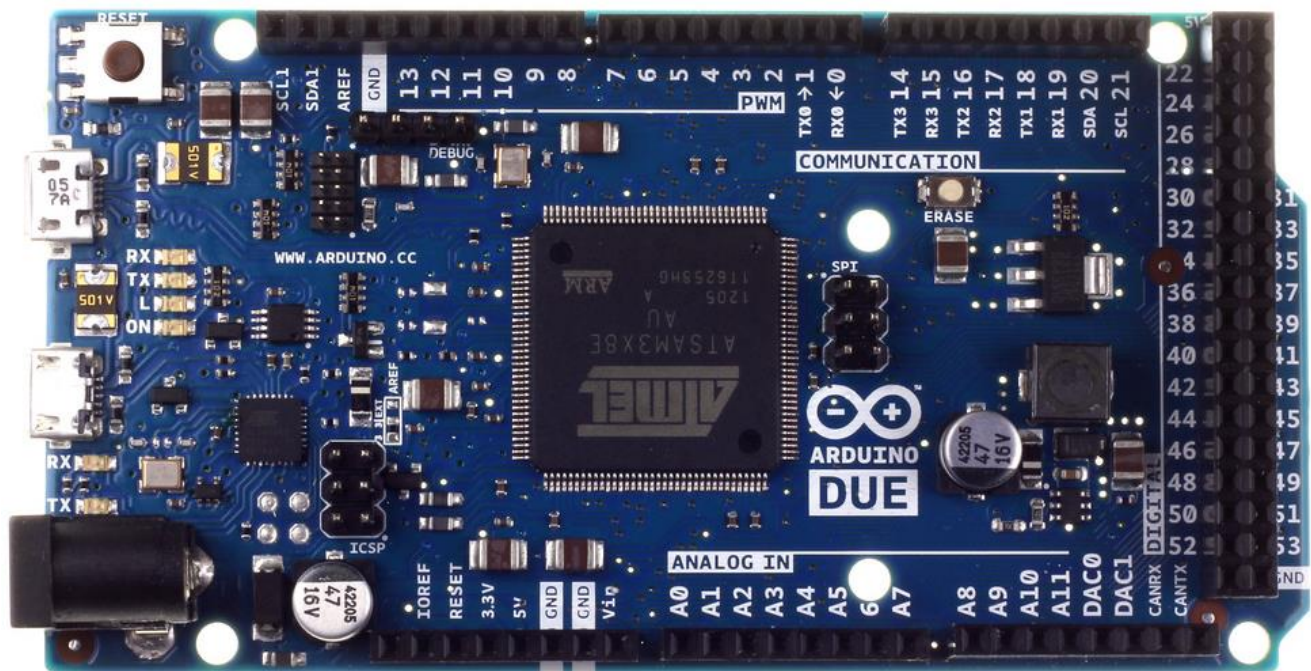


Figure 22. Arduino Due

Dedicated Power Supply

One of the primary specifications for the chronograph was portability. While waiting for the Arduino to ship, the team designed a PCB for a dedicated power supply based on the power supply we constructed in the ECE 480 labs. This power supply gave us outputs of 9 V, 5 V, and 3.3 V. The following figures, 23 and 24, are of the PCB schematic of the power supply and its layout, respectively.

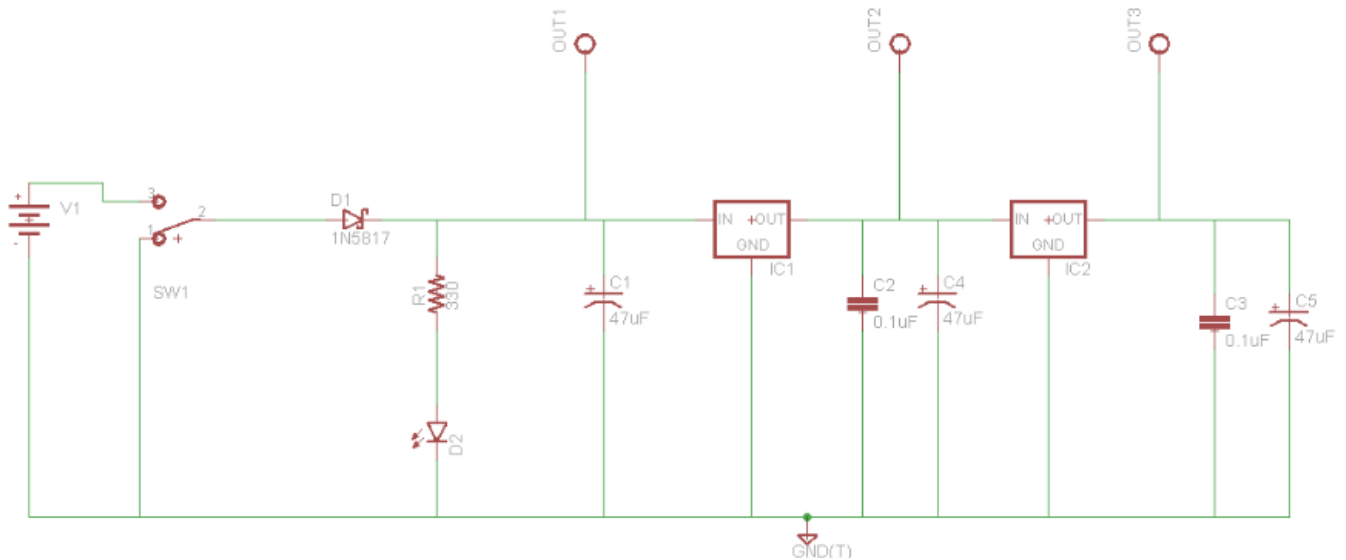


Figure 23. PCB Power Supply Schematic

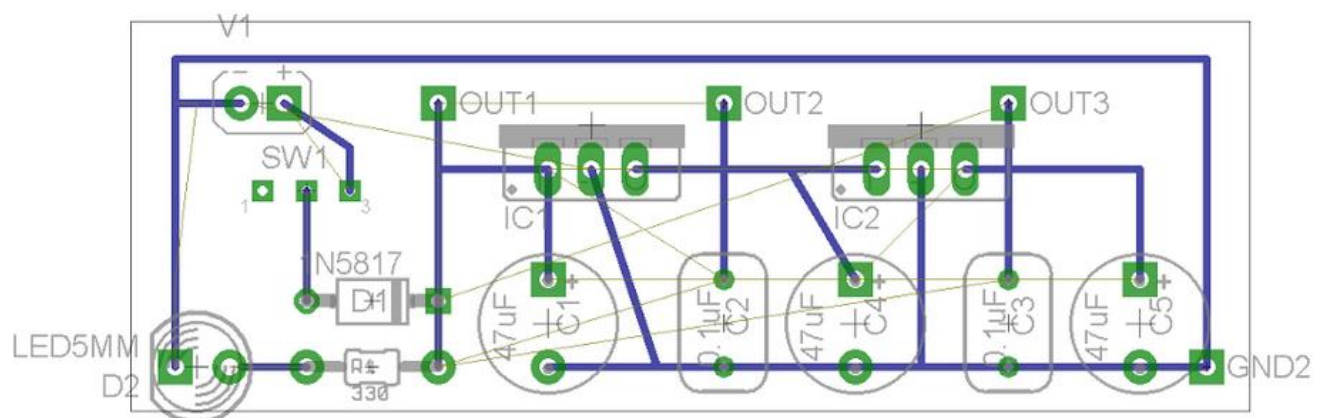
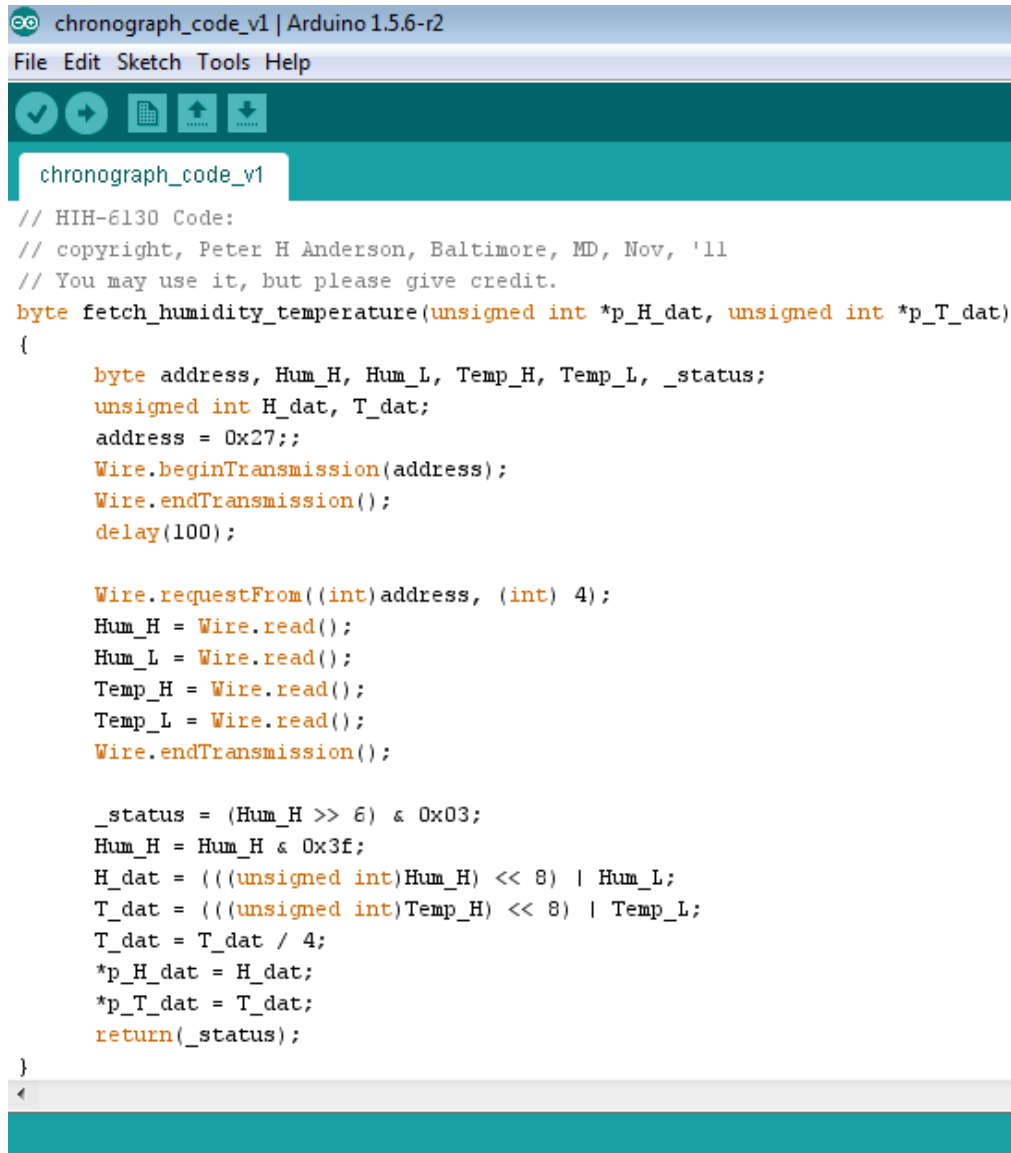


Figure 24. PCB Power Supply Layout

Humidity and Temperature Sensor

After the switch to the Arduino Due, code for the temperature and humidity sensor was readily available online. A temperature and humidity function by Peter H Anderson was selected, and is shown in Figure 25 below. [5] We verified the accuracy by polling the sensor outdoors, and checking the local weather data for accuracy. [11]



```

chronograph_code_v1 | Arduino 1.5.6-r2
File Edit Sketch Tools Help

chronograph_code_v1

// HIH-6130 Code:
// copyright, Peter H Anderson, Baltimore, MD, Nov, '11
// You may use it, but please give credit.
byte fetch_humidity_temperature(unsigned int *p_H_dat, unsigned int *p_T_dat)
{
    byte address, Hum_H, Hum_L, Temp_H, Temp_L, _status;
    unsigned int H_dat, T_dat;
    address = 0x27;;
    Wire.beginTransmission(address);
    Wire.endTransmission();
    delay(100);

    Wire.requestFrom((int)address, (int) 4);
    Hum_H = Wire.read();
    Hum_L = Wire.read();
    Temp_H = Wire.read();
    Temp_L = Wire.read();
    Wire.endTransmission();

    _status = (Hum_H >> 6) & 0x03;
    Hum_H = Hum_H & 0x3f;
    H_dat = (((unsigned int)Hum_H) << 8) | Hum_L;
    T_dat = (((unsigned int)Temp_H) << 8) | Temp_L;
    T_dat = T_dat / 4;
    *p_H_dat = H_dat;
    *p_T_dat = T_dat;
    return(_status);
}

```

Figure 25. Arduino Temperature and Humidity Code

New LCD screen for Arduino

A new LCD screen had to be ordered for the Arduino since the first one we ordered was designed specifically for the MSP430. By configuring the correct pins, the LCD screen was connected to the Arduino. Figure 26 below shows the screen displaying demo text from the Arduino test code provided by Adafruit, the manufacturer. [1]

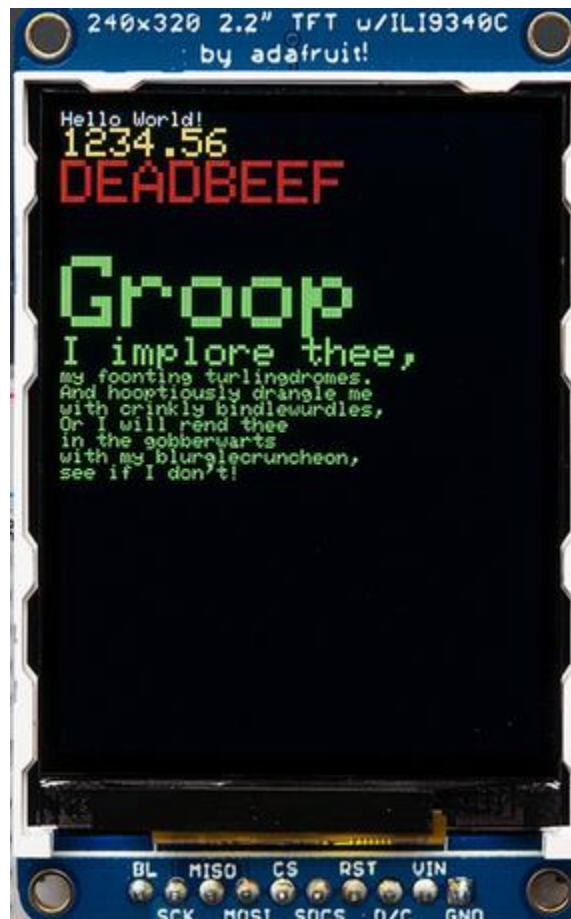


Figure 26. Arduino LCD Display

The First Due Dies

After a long period of testing with the Arduino Due being powered by a wall wart, the Arduino's voltage regulator failed. This led to the Arduino overheating and burning out. Our immediate decision was to order two additional Arduino Dues, and to investigate how we fried the microcontroller. Connecting the wall wart to a voltmeter revealed the wall wart power supply rated at 9 V DC was actually outputting a 14.5 V DC signal. That is outside the effective operating range of the Arduino Due: 7-12 V. It was clear that without additional voltage regulators between the wall wart and

the Arduino, the wall wart should not be used. At this point we abandoned the wall wart completely in favor of the portable power supply we discussed earlier. All further chronograph testing would be powered by batteries reflecting the final chronograph prototype.

Improving the Analog Design

After receiving our new Dues, development was able to continue. We created two complete PCB LED screens and connected them to the Arduino Due. Testing two screens at once demonstrated a new problem: when the voltage of one screen dropped, the other screen dropped as well. This led to instantaneous screen detections and impossible projectile velocities beyond 10,000 feet per second for slow objects like a paper ball. At first glance, the circuit we were using had a direct route to ground for each screen and should have been producing independent voltage levels.

We consulted Dr. Gregory Wierzba on our screen voltage problem to see if he could suggest a way to stabilize the voltage levels. He recommended we use op-amps to isolate each LED bank. By incorporating this feature into our design, we successfully isolated each LED bank and the microcontroller as three separate systems while preserving the voltage detections.

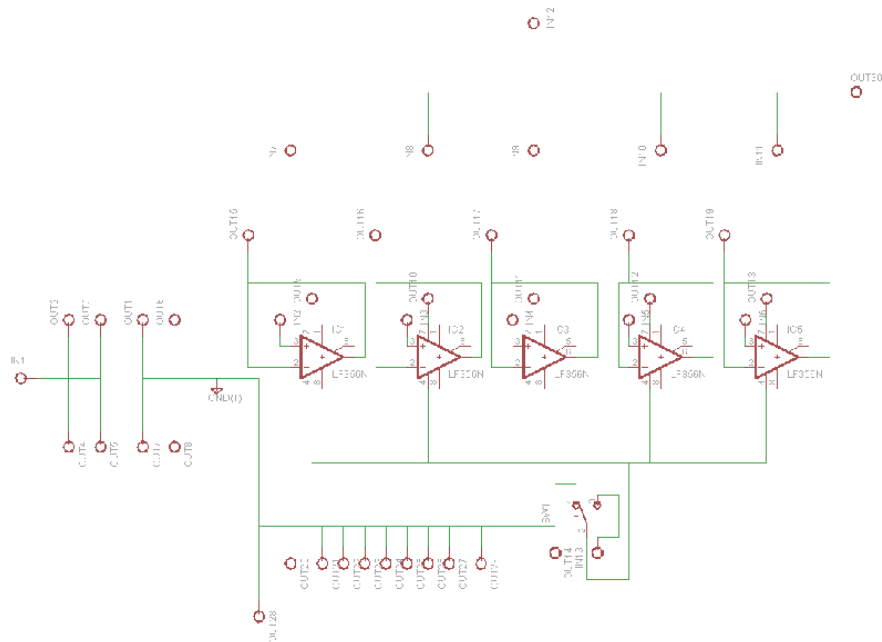


Figure 27. OP-AMP Schematic

After verifying the isolated circuits on a protoboard, we made this into a PCB. The layout can be seen in Figure 26 below.

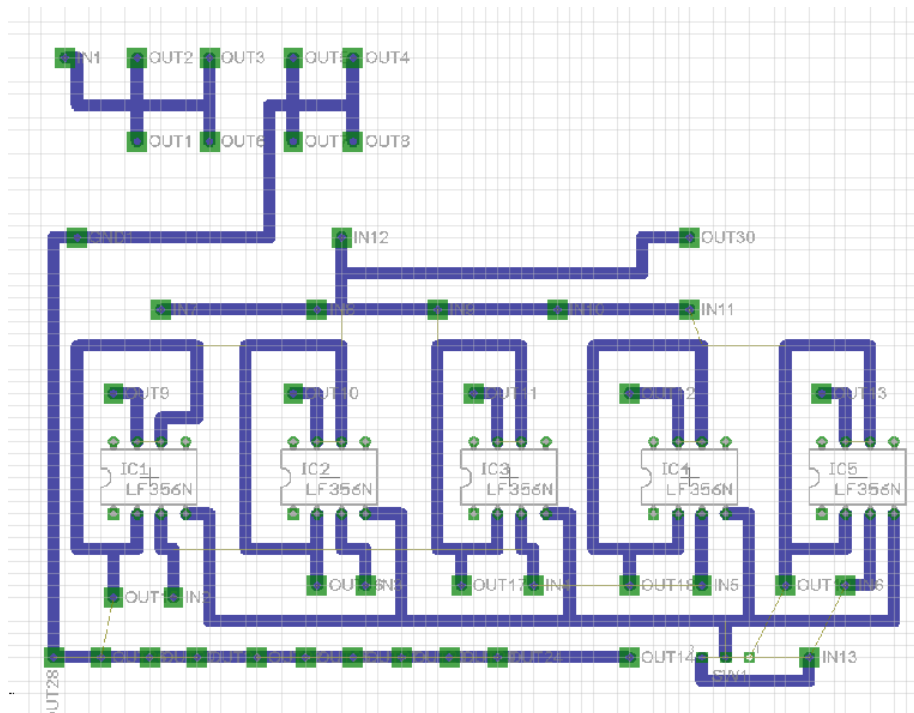


Figure 28. OP-AMP Layout

The second Due Dies

After another long night of testing, another Arduino dies. The reason for this Due overheating was never determined. The on-chip Atmel processor was hot the touch every time the Due was powered, and it stopped accepting or running code. At this point we transitioned to our 3rd and final Arduino Due.

A Line Laser Dies

Through all the testing, one of the lasers had dimmed and needed to be replaced. We noticed that the voltage levels that the screen was normally generating had dropped considerably. This limited the number of lasers we had ordered from 4 to 3, and had to commit to a single laser per screen design. There was not enough time left in the semester to order a replacement laser. This single laser design considerably limited the region of detection, which is discussed in the project capabilities section of chapter 4.

Final Software Development

The microcontroller takes in several samples of voltage readings, calculates an average, and then uses the result to establish a baseline for detection. After these initial values are calculated, the microcontroller enters a loop averaging 10 voltage readings from the first screen with a 50 microsecond delay between each. When it drops below 95 percent of the calculated baseline, excluding ambient light detected by an LED bank ambient light sensor, the microcontroller will report that a projectile has been detected and records the first timestamp. It then enters a second loop averaging 3 voltage readings from the second screen with a 50 microsecond delay between each. When that screen drops below 95 percent of the calculated baseline, the microcontroller records the second timestamp. Figure 29 below shows the two-loop code used to perform detections.



```
chronograph_code_v1 | Arduino 1.5.6-r2
File Edit Sketch Tools Help

chronograph_code_v1 $
// -- Detection code

//time variables
long time1 = 0;
long time2 = 0;

float threshold1 = (float)(led_screen1_min-led_ambient)*95.0/100.0;
float threshold2 = (float)(led_screen2_min-led_ambient)*95.0/100.0;

//screen1
while (1) {
  sensorValue = averagePin(A0, 10);
  if ( (sensorValue-led_ambient) <= threshold1 ) {
    time1 = micros();
    break;
  }
} //end while

//screen2
while (1) {
  sensorValue = averagePin(A1, 3);
  if ( (sensorValue-led_ambient) <= threshold2 ) {
    time2 = micros();
    break;
  }
} //end while
```

Figure 29. Arduino Screen Detection Code

By dividing the total distance across our chronograph, 29.5625 inches, by the time the projectile took to cross both screens, the Arduino calculate the velocity. At this point, it fetches the correct temperature and humidity and reports all three values to the user over Bluetooth. The complete code for the Arduino can be found in Appendix 2.

Physical Construction

The frame was constructed out of aluminum with steel brackets. The brackets are attached to the frame with steel screws for maximum longevity. Each screen is about 14 inches by 14 inches with the depth of the frame between screens about 30 inches. PCBs were used to consolidate the circuits and they were housed in a wood box attached to the side of the chronograph.

Chapter 4 – Data and Validation

The final chronograph prototype did not meet every specification outlined by our sponsor. This chapter details the successes and failures of the project, as well as verification of the functional specifications and overview of the final product capability.

Completed Specifications

The specifications that our project successfully implemented were the laser detection scheme and wireless Bluetooth communication of data. Two specifications that were partially successful were accurate reporting of velocity and collection of environment variables. Velocity accuracy was only verified for speeds up to 70 feet per second and cannot be guaranteed for higher speeds up to 3500 feet per second. The subsection of this chapter titled “Test Plan and Validation” illustrates the testing done to verify that low-speed projectiles can be accurately detected. As for environment variables, only temperature and humidity were implemented in the chronograph’s data collection scheme.

Incomplete Specifications

The altitude and air pressure sensor was not implemented in the final prototype. The code that the sensor came with did not run correctly, and upon further testing, the sensor appeared dead on arrival. Since this was not a critical feature for velocity detection the team decided not to order another sensor.

The initial benchmark was to construct a chronograph that could record velocities up to 3500 feet per second. In theory the framework of our chronograph can report these velocities as the calculations are done in the software. The total software delay introduced by the microcontroller was 650 microseconds. The chronograph can handle up to 11300 feet per second, as demonstrated by modifying the ambient light sensor to create instantaneous screen detections. The team decided the prototype was too valuable to be tested against high speed projectiles. Any collision at that speed would destroy the prototype and the project, and was not worth the risk. Modifying the ambient light sensor was the best way to validate maximum chronograph projectile velocity, and is detailed in the subsection “Test Plan and Validation”.

At the end of the project the LCD screen was removed from the design. Since the Bluetooth connection was working and it could display wirelessly to an Android device, the need for the LCD seemed redundant. By removing the LCD screen we were able to decrease the total power needs of the prototype, expanding upon the low-power and portability specifications of the project.

The data storage specification was not directly implemented by our team. The open source BlueTerm program running on an Android phone can record BlueTooth logs, and therefore storage is possible in that way. The team did begin development on a custom Android application, but as design day approached it was more important to collect data and finalize the working portions of our design. In future designs, it would be ideal to be recording data on an SD card directly in the chronograph, or have a custom Android application to store the data in a more accessible way.

Test Plan and Validation

Each of the individual components and subsystems of the chronograph were tested as they were developed, and are detailed in the previous chapter. This subsection is dedicated to testing the complete system, and the data collected verifying our conclusions.

We tested our chronograph indoors in two environmental extremes: high ambient light, and complete darkness. 100 samples in each environment were collected, and the resulting data is shown below. In order to test our system, we had to use a safe projectile of a known speed. We decided upon using a Nerf gun as it was safe to use indoors, and the projectiles can launch consistently within a certain velocity range. The data collected is shown in figures 30 and 31 below.

Complete Darkness – Round 1

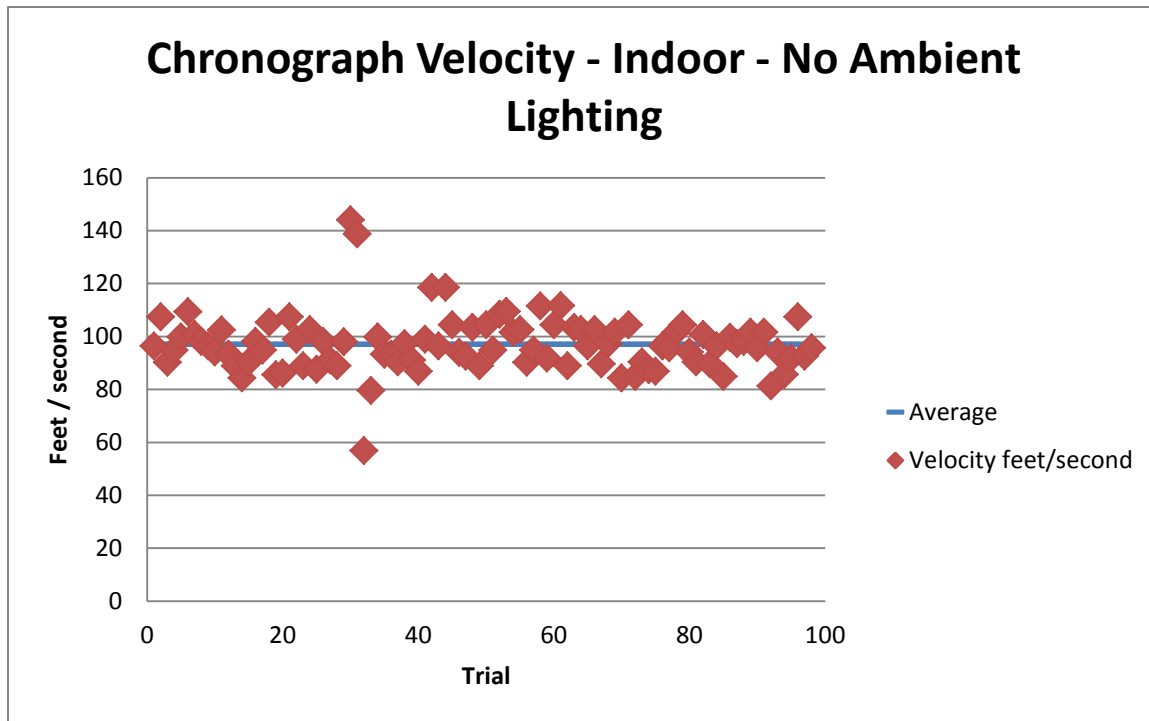


Figure 30. Chronograph Test Dark - Round 1

High Ambient Light – Round 1

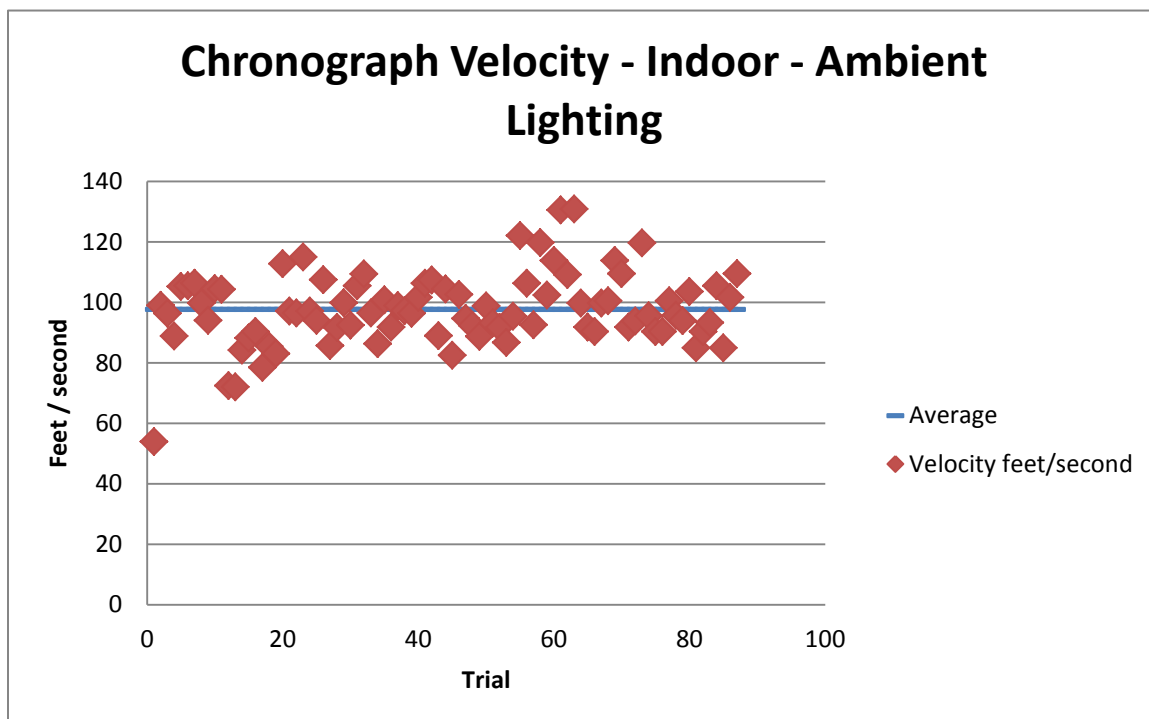


Figure 31. Chronograph Test Light - Round 1

This data showed that the projectile was moving around 97 feet per second on average in both light and darkness, which seemed extremely fast. In order to verify the results of our chronograph, we had to come up with a solution to verify the speed of the Nerf projectile. The first way we verified the speed of the projectile was by launching the projectile through the Caldwell Ballistic Precision Chronograph. Our tests showed the purchased chronograph reading values between 50 and 70 feet per second. It was very difficult to get more than a few readings off the chronograph indoors. Additionally, we believed The Caldwell Ballistic chronograph to be faulty on arrival as we could only successfully get it to run when hooked up to a computer running through the factory use port. This prohibited us from being able to test it outside where it is designed to collect data. Without enough time to return the product, we chose to use it indoors as best we could to get an estimated expected range of Nerf projectile velocity.

Since those velocities were reported by a purchased chronograph in non-ideal conditions, we recorded the sound of the projectile traveling over a known distance and hitting an object. By comparing the time difference in the sounds of the projectile launching and the projectile hitting the object, we determined that the projectile was going 62.5 feet per second in the first 30 inches of flight. Based on these values, and the few successful data points we collected from the purchased chronograph, we knew that our chronograph should be reporting values between 50 and 70 feet per second, averaging around 62.5 feet per second. We expected to have data fill out that entire range due to the inconsistency of Nerf projectiles. The Figure below shows our audio-video analysis of the projectile in Camtasia Studio.

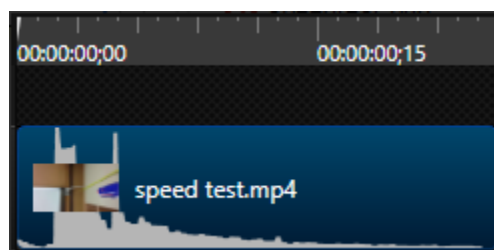
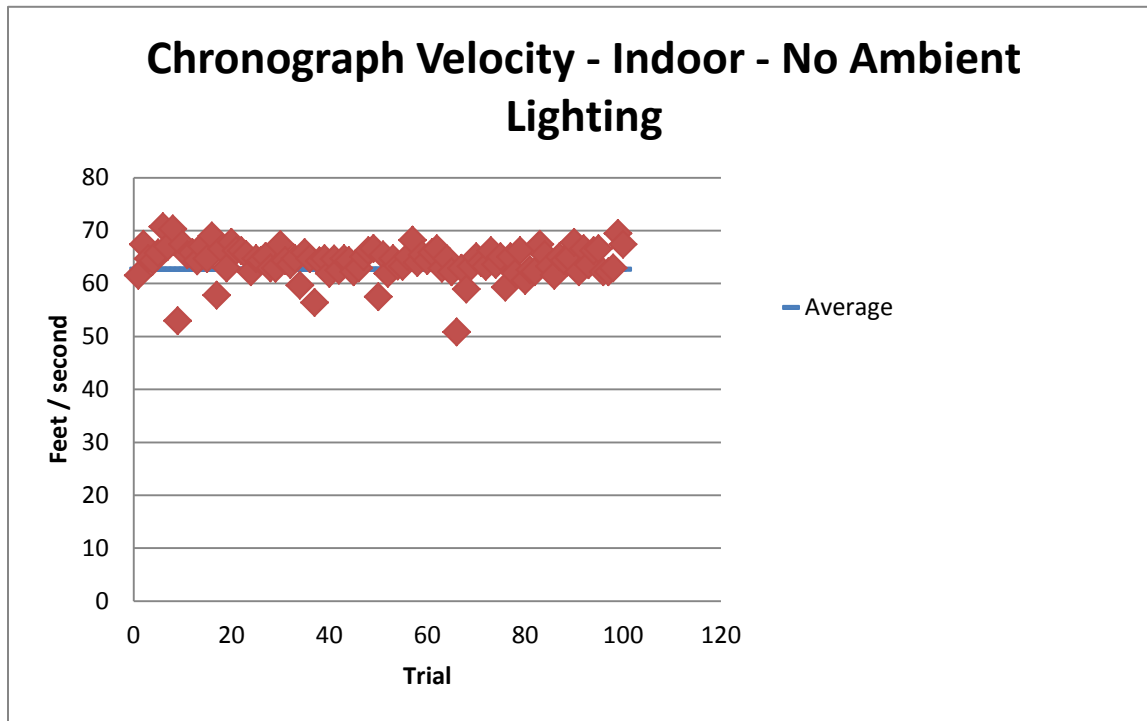
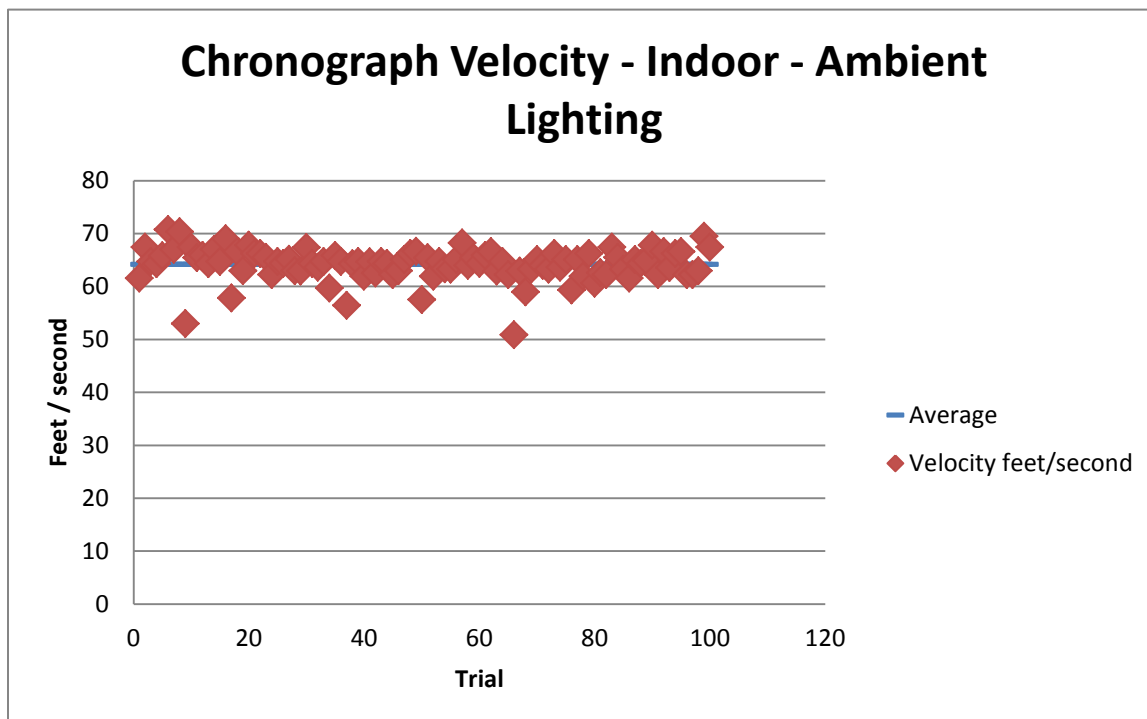


Figure 32. Nerf Projectile Analysis

Based on the previous tests, it was clear that our chronograph was reporting velocities inaccurately. We discovered while testing the chronograph again that the LED sighting on the Nerf gun we were using the test shines a large ring around the gun. After we noticed the extra LED light hitting our screens, we stopped using the LED sight to aim. The velocities reported by our

chronograph were now around 60 feet per second. It was obvious that the extra red LED light was triggering the second screen early. We redid our testing, this time with the red LED sight on the Nerf gun off. This time the average speed in the dark was 62.6 feet per second, and in the light was 64.11 feet per second. The data plots are shown below.

Complete Darkness – Round 2*Figure 33. Chronograph Test Dark - Round 2**High Ambient Light – Round 2**Figure 34. Chronograph Test Light - Round 2*

One of the main problems with the Nerf gun is its inaccuracy. It is difficult to properly aim the gun without the LED sight, but we fixed that by mounting the gun in a known spot. The Figure Below shows our mounted setup for testing.

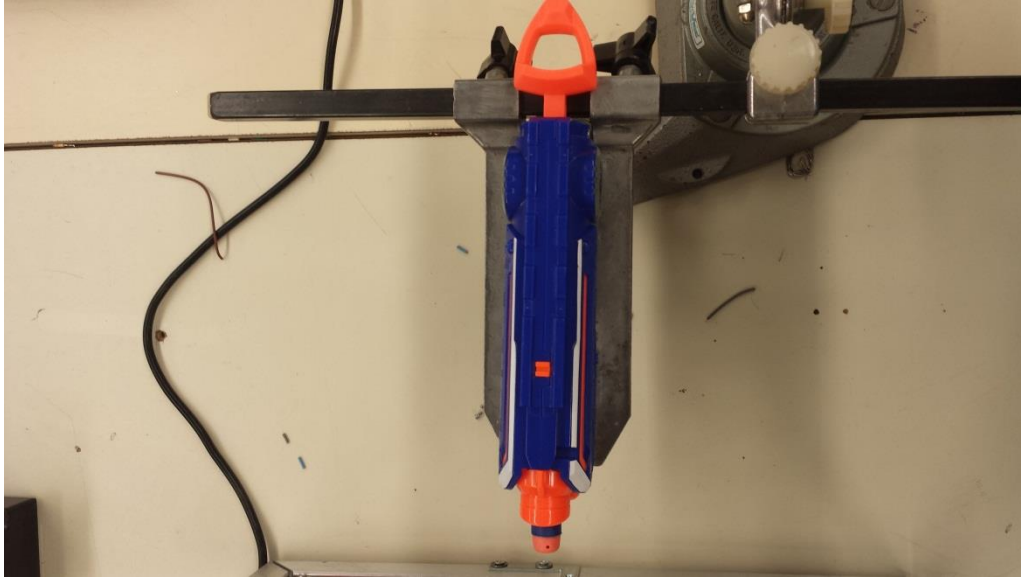


Figure 35. Nerf Gun Mount

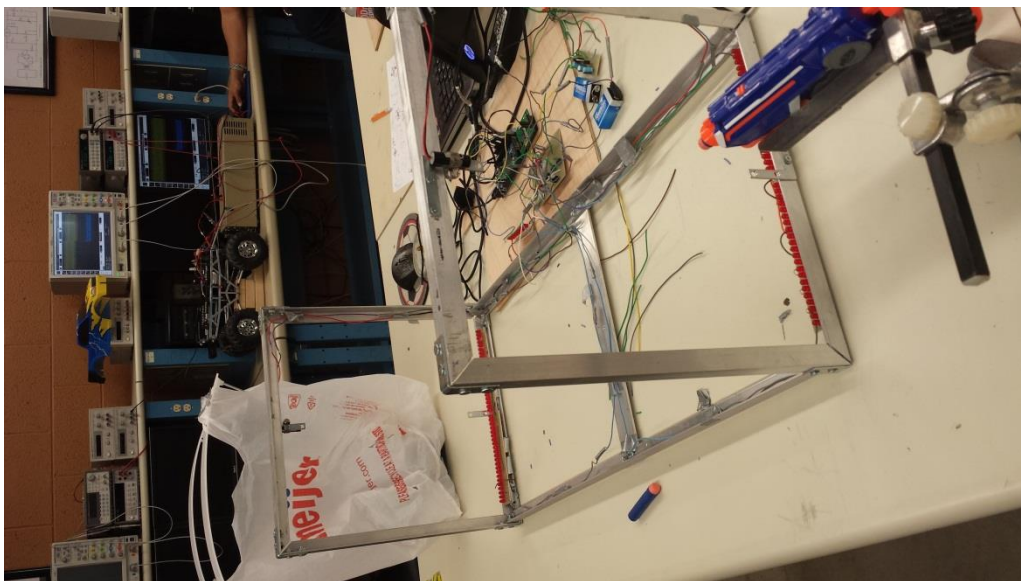


Figure 36. Chronograph Test Setup

This inaccuracy with the Nerf projectile was the primary reason we did not test against higher velocity projectiles: we did not want to damage the prototype. After completing the design the team determined it was too great a risk.

Product Capability and known issues

The inherent problem with our laser detection screens is the reverse-biased LEDs in series. When only one or two LEDs are blocked from receiving the laser, surrounding LEDs will charge them, and the voltage drop will not be significant enough to detect a projectile. Figure 37 below illustrates the charging problem and how the LEDs will remain lit despite the projectile interrupting the laser.

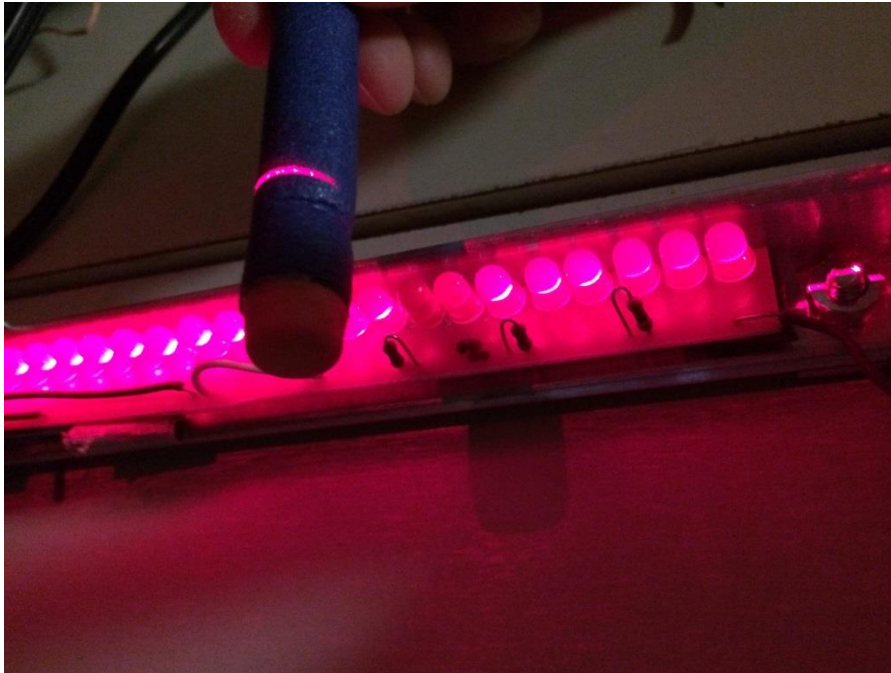


Figure 37. Blocked LED Being Charged

Another problem with our detection screens is the placement of the line laser. The effective detection area is illustrated in the Figure 38 below.

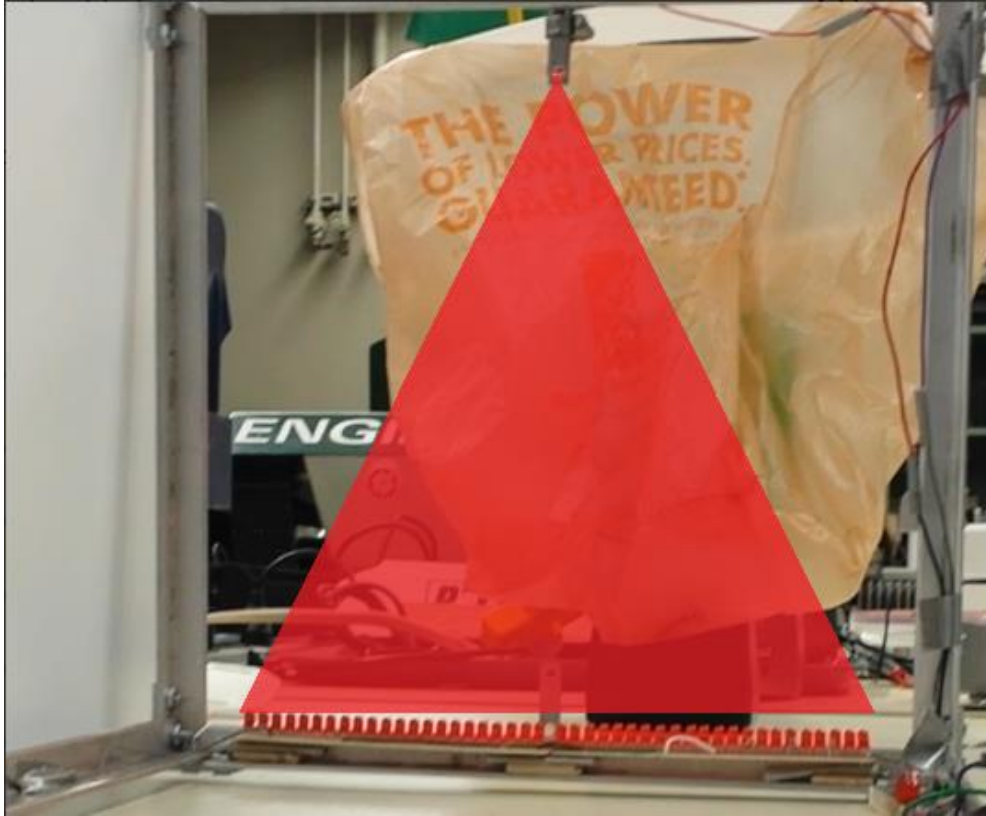


Figure 38. Effect Detection Area

As can be inferred from Figure 38, the effective detection area is much smaller than the square area of the frame. This is an unfortunate design flaw resulting from one of our lasers dying, as detailed in chapter 3, and implementing only one laser per screen.

Another problem is extra LED interference. The bad data we showed in Chapter 4 shows just how drastic an effect extra environment LEDs can have as an effect on our chronograph. Implementing LED shields to protect them from additional light sources would be an effective solution.

The subsection “Future Design Improvements” in Chapter 5 discusses design changes that would solve these problems and successfully fulfill the incomplete specifications.

Chapter 5 – Final Conclusion and Future Design Improvements

Summary

The prototype chronograph successfully explores the effectiveness of a laser based detection scheme. The chronograph can detect low-velocity projectiles with high accuracy, and can theoretically detect high-velocity projectiles. In comparison to other inexpensive chronographs, our chronograph can function indoors, outdoors, and in any lighting conditions. As detailed throughout this report, the design proves that a laser detection scheme is both affordable and practical, and could be applied to improved future designs. Figure 39 below shows the finish prototype chronograph.

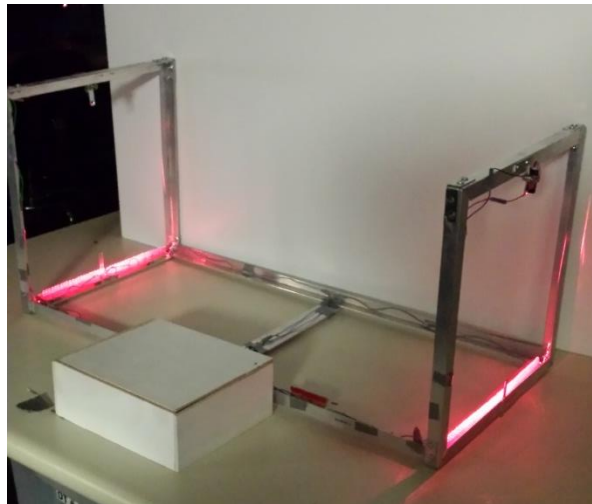


Figure 39. Finished Chronograph

Future Design Improvements

Since the region of detection of the laser is a fraction of what was necessary, the first thing to improve would be to increase the area of detection. We were originally going to implement a few more lasers but due to lasers burning out, we did not implement multiple lasers per screen. The current design can be easily tweaked by adding more lasers per screen.

Another way to improve the design is to orient the LED banks in a different manner. Instead of having the laser shine down upon the LEDs, an implementation from the sides could be explored. By supplanting the current LED configuration with one from the side working in conjunction with the previous solution, this should increase the region of detection.

In addition to improving the effective area, a more sophisticated microcontroller could also be explored such as the Texas Instruments Beaglebone. By using a more sophisticated microcontroller, it will offer the developers a higher sampling rate and can increase the accuracy in triggering the detection of the projectile. The Beaglebone is more expensive than the Arduino Due, but significantly more powerful. New power concerns for battery life and portability would have to be considered.

Also, since we did not complete the sensor specifications for the air pressure and altitude sensors, implementing these sensors as well as other sensors could be an inexpensive improvement to the current design to give the user even more data about the environment.

Our project successfully implemented laser based detection using series connected reverse-biased LEDs. An improvement to this analog system would be to connect each of these LEDs in parallel to remove the charging problem described in Chapter 4. By using a current sensor instead of voltage readings, you could detect whether or not each and LED has been interrupted.

If the entire project was to be reconstructed from scratch, very radical design changes could be made. The analog laser detection system could be extended to include the detection itself, rather than supplying a voltage reading to a microcontroller. By using biased opto-isolators on each LED bank, the voltage readings could be converted into a digital 1 or 0 before the microcontroller received the signal. Now with digital I/O, the microcontroller could have significant speedup, no longer wasting clock cycles on sampling, analog-to-digital conversion, mathematical calculations, and logic comparisons. Each of those would be replaced with an analog system that reduced the input signal to a 1 or 0 based on a given threshold. That threshold could be found experimentally by changing the bias on the opto-isolators. If a new design team was to tackle this project, this very analog driven solution would be recommended as a strong starting point.

Final Cost of Prototype

The complete budget spent on the development of the chronograph is listed in Table 3 shown below.

Final Budget			
	Part	Qty	Cost
	Line Laser Diode - 5mW 650nm Red	2	\$17.90
	Humidity and Temperature Sensor - HIH6130 Breakout	1	\$23.96
	Altitude/Pressure Sensor - MPL3115A2 Breakout	1	\$11.96
	Color LCD Booster Pack (320x240 with microSD card socket)	1	\$16.00
	LED BAR GRAPH 10-SEGMENT RED	15	\$15.36
	Standard LEDs - Through Hole RED DIFFUSED	200	\$15.80
	Line Laser Diode - 5mW 650nm Red	2	\$17.90
	Development Boards & Kits - ARM ARDUINO DUE	1	\$49.99
	2.2" 18-bit color TFT LCD display with microSD card breakout - ILI9340	1	\$24.95
	Caldwell Ballistic Precision Chronograph 720001	1	\$91.99
	ARDUINO A000062 DEV BRD, SAM3X8E ARM Cortex-M3, ARDUINO DUE	2	\$99.98
Total:			\$385.79

Table 3. Complete Experimental Budget

The budget for a single unit chronograph, without any experimental costs, as detailed in this report is listed in Table 4 shown below.

	Final Budget Per Unit		
	Part	Qty	Cost
	Line Laser Diode - 5mW 650nm Red	2	\$17.90
	Humidity and Temperature Sensor - HIH6130 Breakout	1	\$23.96
	Standard LEDs - Through Hole RED DIFFUSED	200	\$15.80
	Development Boards & Kits - ARM ARDUINO DUE	1	\$49.99
	2.2" 18-bit color TFT LCD display with microSD card breakout - ILI9340	1	\$24.95
	ITEAD Bluetooth BoosterPack	1	\$14.00
Total:			\$146.60

Table 4. Chronograph Budget Per Unit

Appendix 1 – Technical Roles and Responsibilities

James Cracchiolo



James served as the team's document preparation manager. For the design of the project, he assisted in the development of the software portion of the project. This included mainly programming both the MSP430 and Arduino Due in a pair programming software development setting with Nick. Pair programming is an agile software development technique where two programmers will work together at one computer. In this type of setting, we were able to solve numerous programming flaws such as implementing the capturing of the voltage from the analog-to-digital converter. He also worked on some Android application development and discovered an open-source application that could connect to the Arduino. Using the Android application developer's website, he worked on developing his own application that would feature a more specific user interface. The interface was eventually scrapped since the open-source application, Blueterm, was able to achieve connectivity to the Arduino easily and output what was necessary for the basic functions of the project. In order to program the sensors, the switch to an Arduino made it easier to program because they were originally designed to be run on an Arduino thus eliminating the need to figure out how to convert the code for the MSP430. He configured the test code that the temperature and humidity sensors came with to assure that they functioned properly. He also assisted in soldering together LEDs to the PCBs as many were manufactured for testing purposes. As part of the requirements of the course, he also aided in presentation preparation and technical documentation as well as editing over all of the documentation.

Xuming Zhang

Xuming's technical role in this project was designing the velocity detecting system as well as hardware installation and testing. He made sure that the LED semiconductor's wavelength range would match the laser that is used in the detection system. Also, because LEDs were used in the detection system, the LED semiconductor works as a reverse-biased diode. He verified that orienting the LED as reverse-bias diode could produce enough voltage to be detected. He also designed all of the PCBs used in the making of the project. After communicating with the group, he drew the basic schematic according to the circuit that was first prototyped via a protoboard. By using the Eagle software to design the schematic of each LED bank with resistor he next had to design the layout in order to fabricate the PCB in the ECE shop. During the designing, he communicated with the team to verify the design specifications, and also communicated with the ECE shop staff to confirm the size of the PCB and the components specifications. He also designed the power supply PCB board which supplies power to the project. The power supply's schematic was based on Dr. Wierzba's schematics which were provided in the ECE 480 lab. In addition, he changed the circuit to providing the isolated ground in the power supply. The technical role that he played in this project included testing circuits and verifying the outcome results. After we found solutions for preventing voltage changes in the ground, we kept testing the circuit until we gained the complete isolated ground. Xuming designed the circuit with wire bus which was used for organizing the circuit wires. Since the design software does not include the component schematic that we used in our project, he found out there are some other components that have different functionality and could be applied into the PCB design. Furthermore, he soldered many components to each PCB and organized the wires to connect to the microcontroller. In order to organize components in our project, he constructed a wood box to house the PCBs. Xuming also aided in the testing of the chronograph.

Steve Kanitz

Steve was the team lab coordinator and mainly contributed to hardware portion of the project. He was a major contributor to designing and testing the analog circuit components used in the project. This required building, testing, and troubleshooting circuits to be used in the project on proto-boards to ensure that they would work properly in the final design, as well as soldering together the final circuits when their printed circuit boards were completed. He built the structural components of the chronograph and mounted the circuit elements to the structure. As lab coordinator, he was responsible for ordering the components the team would be using in the project through the ECE ordering service and retrieving the orders when they were received. This meant ensuring that the parts ordered would be compatible with each other and be sufficient for the project to function as required by the sponsor's specifications. Steve's tasks were not limited to the hardware development of the project; he also contributed by interacting with the sponsor, and the development of the technical documents and presentations that were turned in throughout the course of semester.

Madi Kassymbekov

Madi was the team webmaster by keeping the team's website up-to-date with the team's written reports and presentations. Moreover, as a webmaster he was responsible in designing the team's website that met usability and design requirements of the project. Considering the project portion of the class, he took part in designing and testing the analog circuit components used in the project and helped in designing the microcontroller software used in the project. As part of this process, it required building and testing analog circuits with lab hardware such as an oscilloscope, multimeter and function generator. In addition, he assisted in designing PCB layouts of analog circuits using the Eagle software and soldering analog parts such as resistors and wires to the manufactured PCBs. Apart from the actual project, Madi took part in interactions with the sponsor, and also helped in designing technical documentation and presentations throughout the semester.

Nicholas Mancuso

Nick was primarily responsible for the software side of project. He worked closely with James to design and implement the code for the MSP430 and Arduino microcontrollers. Early in the semester he analyzed the analog-to-digital conversion process to determine how to set the software thresholds on the screen detection loops. Nick was responsible for converting the MSP430 detection code over to Arduino code when the team changed microcontrollers. He worked closely with Xuming, Steve, and Madi on the hardware design, and came up with the Ambient Light sensor solution to stabilize voltage reads in variable indoor lighting conditions. Additionally, he worked closely with Xuming and Steve to implement the hardware circuit isolation op-amps that Dr. Wierzba proposed to improve the detection scheme. Nick and James tried many different software screen detection solutions including multiple-sample averaging, high delay samples without any averaging and rapid voltage polling. Nick also designed the test plan for the chronograph, and came up with the idea for the sound-based projectile velocity validation. He implemented code to support LCD screens on both microcontrollers. The final chronograph did not end up using an LCD screen and instead completely transitioned to a full Bluetooth implementation. The Bluetooth communication over serial the Arduino serial ports was designed in conjunction with James, who was implementing a custom Android application. That development application was deemed redundant after discovering the open source application BlueTerm, and Nick and James verified that the serial communication on the Arduino could be displayed through this application.

Appendix 2 – Literature and website references

- [1] Adafruit LCD Screen
<http://www.adafruit.com/products/1480>
- [2] Arduino Due
<http://arduino.cc/en/Main/ArduinoBoardDue>
- [3] Caldwell Chronograph
<http://www.battenfeldtechnologies.com/caldwell/catalog.asp?product=ballistic-precision-chronograph>
- [4] CED M2 Chronograph
<http://www.cedhk.com/shop/products/CED-M2-Chronograph-Set.html>
- [5] Humidity Sensor Code
<http://www.phanderson.com/arduino/hih6130.html>
- [6] ITEAD Studio Bluetooth BoosterPack and Pinout
<http://imall.iteadstudio.com/bt-boosterpack.html>
- [7] MSP430 Laser Security
<http://forum.43oh.com/topic/1386-laser-security-v2-msp430/>
- [8] MSP430 Example Wiki
http://processors.wiki.ti.com/index.php/MSP430_LaunchPad_Tutorials
- [9] PACT Pro Chronograph
http://pact.com/index.php?option=com_content&task=view&id=19&Itemid=66
- [10] RobG LCD Screen
http://store.43oh.com/index.php?route=product/product&product_id=80
- [11] Weather.com
<http://www.weather.com/>

Appendix 3 and beyond – Detailed technical attachments –

Complete Chronograph Data – Feet/Second

Light Samples		Dark Samples
61.53		63.24
67.35		65.05
64.65		65.29
64.26		65.75
65.74		62.9
70.7		66.96
66.93		64.31
70.26		64.31
52.93		62.89
67.35		56.35
65.41		61.53
65.8		62.84
64.26		58.67
66.95		60.87
64.66		61.2
68.96		61.86
57.76		60.87
66.56		63.59
62.89		65.8
67.72		53.43
66.18		63.92
66.18		61.53
65.42		62.55
62.2		63.93
64.67		63.94
64.31		61.52
65.05		64.32
62.89		64.67
62.83		63.59
67.3		59.6
64.29		58.98
63.59		60.54
64.67		63.56
59.62		65.42
65.79		58.98
64.67		61.2
56.35		58.98
64.26		64.31
64.67		65.05
61.86		60.22

64.62		62.89
62.49		62.55
64.68		62.87
64.31		64.31
62.2		63.24
62.88		56.33
65.04		61.86
66.17		58.98
66.57		62.54
57.48		64.29
65.41		58.06
61.86		65.8
64.63		64.32
63.24		62.52
63.19		59.29
65.04		63.95
68.16		58.38
63.92		58.98
65.41		63.24
64.28		63.59
65.79		64.67
66.56		60.87
62.86		64.67
64.67		62.2
62.19		60.88
50.8		62.55
62.83		60.55
58.94		62.89
63.21		63.24
65.03		60.23
63.94		65.42
63.23		66.96
66.17		62.89
63.59		63.24
65.01		62.19
59.29		68.16
65.02		59.6
61.86		67.36
66.18		65.42
60.54		62.89
62.54		63.24
62.17		63.57
67.35		66.54
65.42		63.24
63.23		63.55
61.52		58.98
65.01		59.91

64.31		62.54
64.85		62.52
67.74		63.95
62.2		63.59
66.56		63.59
63.51		63.24
66.14		62.55
66.56		64.64
62.17		63.24
62.2		63.59
62.89		65.02
69.41		64.32
67.35		65.03

Complete Arduino Code

```
//includes and function definitions
#include <Wire.h> //I2C library

byte fetch_humidity_temperature(unsigned int *p_Humidity, unsigned int *p_Temperature);

//global variables
int led_screen1_min = 0; //screen 1 min
int led_screen2_min = 0; //screen 2 min
int led_ambient = 0; //average ambient value
int sensorValue = 0; //current sensor reading
unsigned long microseconds = 0; //timing in microseconds

//initilialize the minimum voltage on a pin
int initializeMin(int pin) {
  int sensor = analogRead(pin); //read a value
  delayMicroseconds(500); //delay
  int minimum = 1025; //minimum starts at 1025
  for (int i=0; i<500; i++) {
    sensor = analogRead(pin);
    delayMicroseconds(50); //50 micro delay
    if (sensor < minimum) {
      minimum = sensor;
    }
  }
  //end for
  return minimum;
}

//average an analog input by averaging 200 samples with 500 microsecond delay
int averagePin(int pin, int samples) {
  int sensor = analogRead(pin); //read a value
  delayMicroseconds(50); //delay
```

```

int total = 0; //total sum is 0
sensor = 0; //reset recording
//take x samples and total them
for (int i=0; i<samples; i++) {
  sensor = analogRead(pin);
  delayMicroseconds(50);
  total += sensor;
} //end for
return (total/samples);
}

```

// the setup routine runs once when you press reset:

```

void setup() {
  //debug serial prints
  Serial.begin(9600);
  Wire.begin();
  pinMode(4, OUTPUT);
  digitalWrite(4, HIGH); // this turns on the HH3610
  delay(1000);
  Serial.println();
  Serial.println(">>>>>>>>>>>>>>>>>>>>>>>>>"); // just to be sure things are working
}

```

// the loop routine runs over and over again forever:

```

void loop() {

  sensorValue = 0;
  led_screen1_min = initializeMin(A0); //get the min value at screen1
  led_screen2_min = initializeMin(A1); //get the min value at screen2
  led_ambient = averagePin(A2, 200); //get average ambient reading

  Serial.println();
  Serial.println("--Initializing--");
  Serial.println(" >>>> Debug Variable Start <<<< ");
  Serial.println(led_screen1_min);
  Serial.println(led_screen2_min);
  Serial.println(led_ambient);
  Serial.println(" >>>> Debug Variable End <<<< ");

  delay(50); //delay for 50 milisecond before detection is good to go again
  Serial.println("--Ready for Detection!--");

  // -- Detection code

  //time variables

```

```
long time1 = 0;
long time2 = 0;
long time3 = 0;

float threshold1 = (float)(led_screen1_min-led_ambient)*95.0/100.0;
float threshold2 = (float)(led_screen2_min-led_ambient)*95.0/100.0;

//screen1
while (1) {
  sensorValue = averagePin(A0, 10);
  if ( (sensorValue-led_ambient) <= threshold1 ) {
    time1 = micros();
    break;
  }
} //end while

//screen2
while (1) {
  sensorValue = averagePin(A1, 3);
  if ( (sensorValue-led_ambient) <= threshold2 ) {
    time2 = micros();
    break;
  }
} //end while

//calculate time
long totalTime = time2-time1;

//fetch humidity and temperature readings
unsigned int H_dat, T_dat;
float RH, T_C;
byte response = fetch_humidity_temperature(&H_dat, &T_dat);
RH = (float) H_dat * 6.10e-3;
T_C = (float) T_dat * 1.007e-2 - 40.0;

//report velocity based on known distance (approximately 30 inches)
float distance = (29.5625/12.0); //in inches, change numerator to get correct amount
Serial.print(" Velocity: ");
float velocity = (distance/(float)totalTime ) * (float)1000000;
Serial.print(velocity);
Serial.println(" ft/s");
Serial.print("Humidity: ");
Serial.println(RH);
Serial.print("Temperature(c): ");
Serial.println(T_C);
delay(2000); //wait 2 second after reading

}
```

```
// Illustrates how to measure relative humidity and temperature.
// HIH-6130 Code:
// copyright, Peter H Anderson, Baltimore, MD, Nov, '11
// You may use it, but please give credit.
byte fetch_humidity_temperature(unsigned int *p_H_dat, unsigned int *p_T_dat)
{
    byte address, Hum_H, Hum_L, Temp_H, Temp_L, _status;
    unsigned int H_dat, T_dat;
    address = 0x27;;
    Wire.beginTransmission(address);
    Wire.endTransmission();
    delay(100);

    Wire.requestFrom((int)address, (int) 4);
    Hum_H = Wire.read();
    Hum_L = Wire.read();
    Temp_H = Wire.read();
    Temp_L = Wire.read();
    Wire.endTransmission();

    _status = (Hum_H >> 6) & 0x03;
    Hum_H = Hum_H & 0x3f;
    H_dat = (((unsigned int)Hum_H) << 8) | Hum_L;
    T_dat = (((unsigned int)Temp_H) << 8) | Temp_L;
    T_dat = T_dat / 4;
    *p_H_dat = H_dat;
    *p_T_dat = T_dat;
    return(_status);
}
```