Federal Aviation Administration

DOT/FAA/AM-15/18
Office of Aerospace Medicine
Washington, DC 20591

Open-Source Products for a Lighting Experiment Device

Kevin M. Gildea
Nelda Milburn

Civil Aerospace Medical Institute
Federal Aviation Administration
Oklahoma City, OK 73125

October 2015

Final Report

NOTICE

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/AM-15/18		

4. Title and Subtitle	5. Report Date
Open-Source Products for a Lighting Experiment Device	October 2015
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Gildea KM, Milburn N	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
FAA Civil Aerospace Medical Institute P.O. Box 25082 Oklahoma City, OK 73125	11. Contract or Grant No.

12. Sponsoring Agency name and Address	13. Type of Report and Period Covered
Office of Aerospace Medicine Federal Aviation Administration 800 Independence Ave., S.W. Washington, DC 20591	14. Sponsoring Agency Code

15. Supplemental Notes

Work was accomplished under approved task AHRR-521

16. Abstract

The availability of increasingly powerful and versatile open-source software and hardware products continues to open new possibilities for the design and development of experimental devices. The declining cost of many proprietary software and hardware solutions has further increased the options available to researchers. The capabilities of open-source software and microcontrollers were used to construct a device for controlled lighting experiments.

The device was designed to ascertain whether individuals with certain color vision deficiencies were able to discriminate between the red and white lights in fielded systems based on luminous intensity. The device provided the ability to control the timing and duration of LED and incandescent light stimuli presentation, present the experimental sequence and verbal instructions automatically, adjust LED and incandescent luminous intensity, and display LED and incandescent lights with various spectral emissions. The lighting device could easily be adapted for experiments involving flashing or timed presentations of colored lights or the components expanded to study areas such as threshold light perception and visual alerting systems.

17. Key Words	18. Distribution Statement
Color Vision, Colored Signal Lights, Light-Emitting Diodes (LEDs), Vision Research, Open-Source Products	Document is available to the public through the Internet: www.faa.gov/go/oamtechreports

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	26	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Contents

Open Source Products for a Lighting Experiment Device

Open-Source Products for a Lighting Experiment Device

The proliferation of open-source products, specifically software applications and open-source microcontrollers is increasing the flexibility and capabilities in the design and construction of devices for controlled experiments (Christie & Gianaros, 2013; Mathot, Schreij, & Theeuwes, 2012). Previously, many controllers used in experiments required a substantial understanding of software engineering and computer programming. Now, software solutions make programming the device relatively quick and easy (Dixon, 2009; Reimers & Stewart, 2009; Stahl, 2006). Flexibility often increases with open-source software (OSS) solutions (Stoet, 2010; von Bastian, Locher, & Ruflin, 2013), and recent developments in microcontroller design and construction have resulted in products that are obtainable with more modest research budgets. Open source solutions have been used for processing physiological data (Christie & Gianaros, 2013), auditory experiments (Hillenbrand & Gayvert, 2005), and for vision research (Teikari et al., 2012).

A recent research effort at the Civil Aerospace Medical Institute (CAMI) leveraged the capabilities of open-source software and microcontrollers to construct a device for controlled lighting experiments, after searching for, but not finding, an off-the-shelf solution. Colored lights serve many functions in transportation and other fields. Common uses rely on color to convey pertinent safety information such as traffic signals, railroad crossings, road construction signs, and aviation signal lighting (Bullough, Yuan, & Rea, 2007; FAA, 2011). Signal lighting is transitioning from incandescent sources to light emitting diodes (LEDs), creating the research need to understand the ramifications in terms of human visual processing in color vision normal and deficient individuals. The objective of this study was to determine whether pilots, including pilots with color vision deficiencies (CVDs) possessing "Statements of Demonstrated Abilities" (SODAs) for color vision are able to use color-coded LEDs as effectively as they used color-coded incandescent sources in landing systems during their occupational color vision test (OCVT).

The device, constructed for under $1,000 [U.S. Dollar (USD)], fully met the following experimental control requirements.

- Control the timing and duration of LED and incandescent light stimuli presentation
- Present the experimental sequence and verbal instructions automatically
- Adjust LED and incandescent luminous intensity
- Display LED and incandescent lights with various spectral emissions

We previously used a lighting device that presented stimuli of varying chromaticities to evaluate the ability of normal and color deficient participants to identify the color of lights (Milburn & Gildea, 2012). The current device provides additional control over experimental variables. We needed to be able to present stimuli from different light sources, at specified chromaticities and luminous intensities; control the durations of illumination, inter-stimulus intervals, stimulus presentation order; and present participants instructions and auditory stimulus cues. The lighting device could easily be adapted for experiments involving flashing or timed presentations of colored lights or the components expanded to study areas such as threshold light perception and attention-getting mechanisms for lights.

DESIGN AND DEVELOPMENT OF THE DEVICE

Design Specifications

The design process was aided by lessons learned from the design and construction of a manually-controlled precision approach path indicator (PAPI) experimental device (Milburn & Gildea, 2012). PAPI lighting systems present flight crews on final approach to runways with a visual indicator of their height—either above or below—an optimal glide slope. A combination of red and white lights conveys this information to the pilots. Early discussions led us to explore microcontrollers that could control LED luminance using pulse width modulation (PWM). An open-source microcontroller, called an Arduino, provided precise control of the device used in this study and in other experimental efforts (D'Ausilio, 2012). In addition to the Arduino, there are a number of other microcontroller solutions available. Some of these devices are compatible with Arduino hardware and/or software, while others do not assure compatibility with the Arduino but are similar in concept. Some single board computers provide enhanced capabilities to the Arduino or similar devices. Some boards use various versions of Linux and provide additional options for programming languages. Appendix A provides a non-exhaustive list of viable devices, and an Internet search will yield numerous other options.

The devices and associated software span a wide range of prices. The factors of cost and capabilities that addressed our experimental design goals guided our decision to use the Arduino. There are extensive communities of hobbyists and developers using each of these solutions that have populated websites with numerous resources, including sample code, schematics, and lessons learned.

Materials

The key components necessary for this device and relevant to the topic of this paper are the computer, Arduino board, relay module, the light assemblies, and the software for programming and operating the device. We used two other software packages in addition to the Arduino software, Audacity and Gobetwino, to record and play back the audio for the experimental instructions.

Hardware

Computer. A Dell E6400 ATG served as the controller for the Arduino and the platform for experiment audio delivery. The Arduino does not have any specific computer requirements and will run on a Windows®, Linux, or Mac OS X® platform. The software required to program an Arduino is minimal and after uploading the code to the Arduino, generally through a USB cable, all of the processing takes place in the onboard microprocessor.

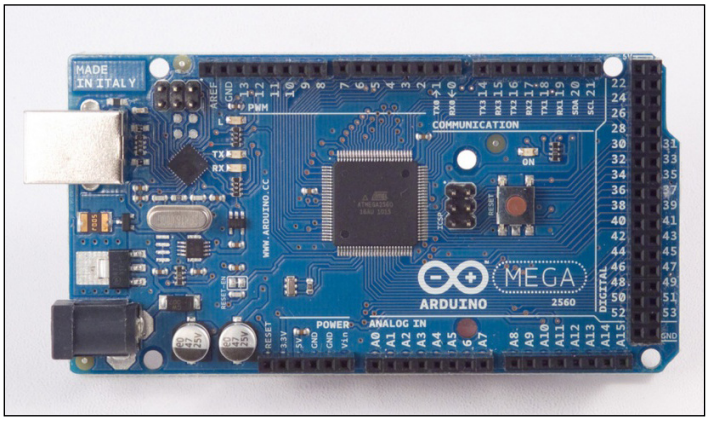


Figure 1. Arduino Mega 2560.

Arduino Mega 2560 microcontroller board. The Mega 2560 Rev 3 board[1] (Fig. 1) provided 54 digital input/output (I/O) pins, including 14 that were capable of PWM. This was a sufficient number to control all of the groupings of LEDs and the incandescent lamps. The LED amperage requirements were 20-30mA (Appendix B), which was within the 40 mA limit of the Arduino digital I/O pins. Additionally, each LED was wired in series with a 220 ohm current limiting resistor that reduced the current drawn by each LED to a maximum of 8.4 mA for the white LEDs and 14.1 mA for the red LEDs.

PWM was used as the primary means of controlling the relative luminous intensity between each of the 16 clusters of 3 LEDs. A 10K ohm 15-Turn PC-Mount Cermet Potentiometer/ Trimmer wired in series with each LED provided fine control of the luminous intensity of the individual LEDs. Each LED was also wired in series with a 220 ohm, ¼ watt resistor to preclude the possibility of a situation where an LED would be presented with no resistance if the potentiometer were to be turned all the way down. A readily available supply of manually adjusted potentiometers influenced their use. The manual adjustment may have added time while calibrating the luminous intensities of the LEDs. Other options that might speed the calibration time and provide more fine control could be digital potentiometers or multi-channel LED drivers.

The Arduino microcontroller boards use an 8-bit, 16 megahertz (MHz) Atmel AVR microprocessor. The microprocessor is programmed using an open-source, Wiring-based language that has similarities with C/C++. A discussion of the programming and the code for this experiment are included in the Software section. The software runs on Windows®, Mac OS X®, and Linux.

Additional boards plugged into the Arduino, generically referred to as *shields,* can provide capabilities such as control of higher voltage or amperage switches, motor control, or for further means of communication. There is a selection of shields available as plans, kits, or completed units. Additional shields can be custom designed and made, as in this research project (Fig. 2), for specialized needs such as controlling the lighting device described in this paper. A custom shield using a commercial off-the-shelf (COTS) Arduino Protoshield as the basis allowed the Arduino to power the LEDs and the relays for the incandescent lights. Arduino Protoshields provide a basic bare circuit board layout for creating custom circuits for use with the Arduino microcontroller board.

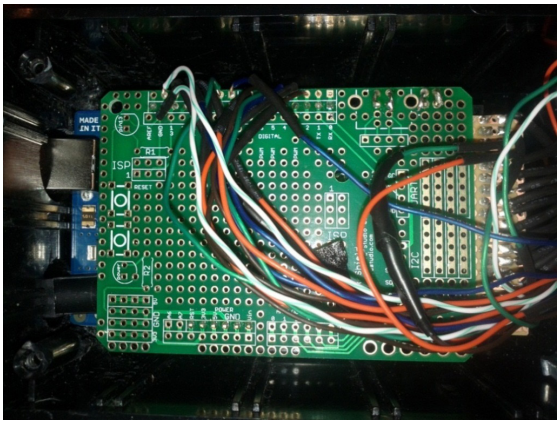


Figure 2. Arduino Protoshield

SainSmart 16-Channel 12V Relay Module for Arduino. A commercial off-the-shelf 16-relay board controlled the incandescent lamps (Fig. 3). A relay board was required to control the incandescent lights because the power requirements of a typical incandescent bulb significantly exceed the power output capabilities of a typical microcontroller. The relay board provided the necessary intermediary between the low power output of the microcontroller and the higher power demands of the incandescent bulbs.

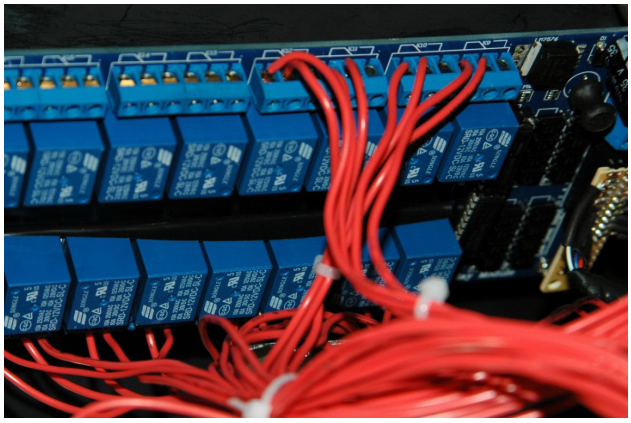


Figure 3. SainSmart 16-Channel 12V Relay Module for Arduino

[1]There are other versions of Arduino boards, and all versions provide digital Input/Output, but not all are capable of PWM. Analog input (6-20Volt) is another available feature. Communication with the Arduino can be accomplished through USB, Bluetooth, Wi-Fi, or other methods. Arduino boards cost less than $100 USD.

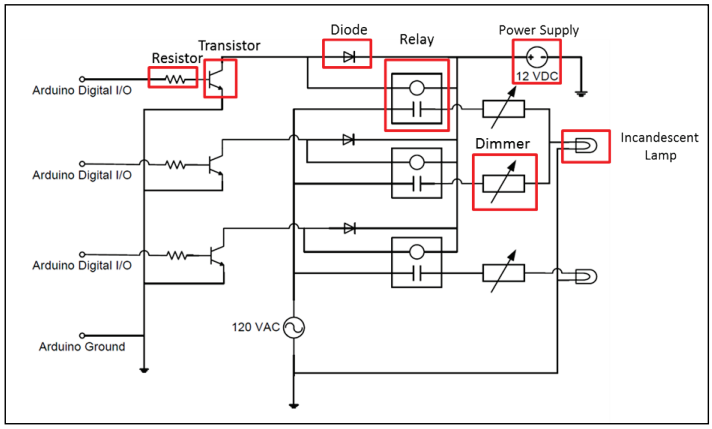


Figure 4. Incandescent lighting control schematic

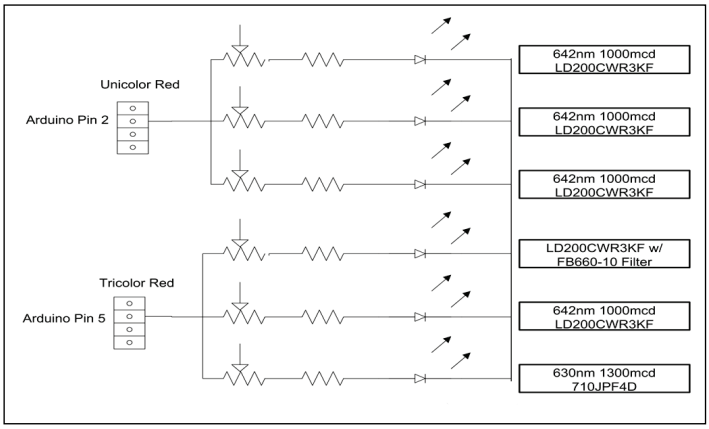


Figure 5. Partial schematic LED lighting control

Although only six incandescent lamps were used (two red, two white, and two green), 12 luminance levels were used. The Arduino board energized the required circuits, each including one of 12 dimmer switches wired in parallel. Availability of analog dimmer switches influenced their use for control of the incandescent bulbs. As with the LEDs, digital control would likely speed the calibration times and facilitate fine-grained adjustments.

The incandescent lighting schematic (Fig. 4) provides a simplified depiction of what is being accomplished with the relay board, with the number of lights reduced for clarity. Additionally, the SainSmart Relay Module contains additional components for purposes (such as voltage regulation) that are beyond the scope of this paper.

When controlling higher-powered devices such as incandescent lights and high-powered LEDs, it is necessary to increase the signal from the Arduino digital I/O pins. Each I/O pin only provides 40 mA and 5V in the "High" state. This is sufficient to operate many lower-powered LEDs such as those used in this experiment. However, a 60W incandescent bulb requires 500 mA and a 120W bulb 1,000 mA. The resistor at the output of the Arduino Digital I/O is a current-limiting resistor placed in the circuit to prevent excessive amperage draws from the Arduino board. The transistor in this circuit acts as a switch to allow the 12VDC supplied to the collector to flow through the transistor to the emitter, thus completing that circuit when the 5V signal from the Ardiuno I/O pin is in the "High" state. The completed 12VDC circuit energizes the coil in the relay, closing the Normally Open switch in the relay and completing the circuit, providing 120VAC to the incandescent lamp.

Light assemblies. The PAPI system lighting that the device was to simulate presents flight crews on final approach to runways with a visual indicator of their height either above or below an optimal glide slope. This is accomplished with a combination of red and white lights at specified relative luminous intensities and wavelengths.[2] The device replicated the relative luminous intensities and wavelengths of halogen incandescent lights used in legacy PAPI lighting systems and the change to LEDs.

Light Emitting Diodes (LEDs). All LED stimuli consisted of 5mm cylindrical LEDs combined in groups of three, as depicted in the simplified schematic (Fig. 5). There were 48 LEDs in total, resulting in 16 LED groupings. Eight of the LED groups were set at equal luminance levels between the red and white LEDs. The other set of eight LED groups presented the white light at twice the luminance level as the red light in accordance with the specifications prescribed for fielded PAPI systems. The objective of the red and white groupings of equal luminous intensity was to ascertain whether individuals with certain color vision deficiencies were able to discriminate between the red and white lights in fielded systems based on luminous intensity, even if they were unable to detect a difference in color.

There were two chromaticity conditions with the LED light source. Half of the LED groupings consisted of unicolor emissions with each of the three LEDs emitting the same wavelength (Fig. 6). The other half of the groupings consisted of LEDs, each emitting a separate wavelength for a tricolor condition. The unicolor white LEDs were 5500 Kelvin (5500K), and the tricolor white LEDs were 3000K, 5500K, and 8000K. Unicolor red LEDs were 642nm, with the tricolor red LEDs being 628nm, 642nm, and 660nm.

LED selection from commercially available sources is problematic in terms of chromaticity, luminance, and distribution pattern. There is a limited selection of chromaticities available from LEDs. For the purposes of this study, it was necessary to have an LED that had a dominant wavelength of 660nm, and the available LEDs with sufficient luminance for our purposes could provide a peak wavelength of 660nm—but the dominant

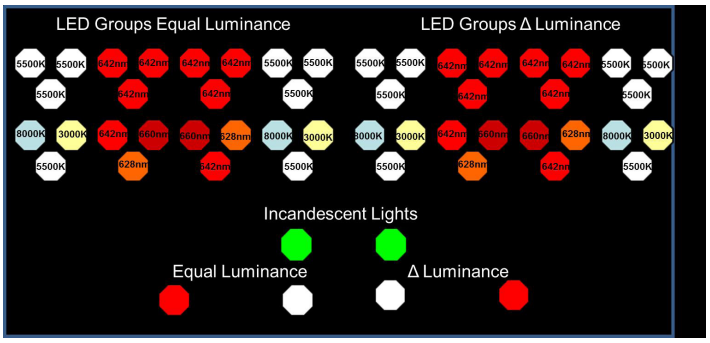


Figure 6. Light positioning within experimental device

[2] The luminous intensity and wavelength of the emissions from PAPI systems are dictated by regulation (FAA, 2011).

wavelength from these LEDs was generally 652nm. This was too close to the 642nm emissions of some of the red LEDs for our experimental purposes. To address this issue, a bandpass filter was selected that would only pass 660nm +/- 2nm, and this was placed in front of a 642nm LED to provide the required emissions.

The luminance intensities of the commercially available LEDs also presented challenges. We were unable to find LEDs with identical luminous intensities for all of the necessary wavelengths. The solution was to select LEDs that provided a higher luminous intensity than was necessary and then adjust those luminous intensities with a combination of current-limiting variable resistors/potentiometers and PWM control from the Arduino.

Beam patterns from LEDs vary widely, based on the lens and construction. To overcome some of these differences, we used only 5mm cylindrical and domed LEDs with similar distribution patterns. There are several typical LED light distribution patterns, including lambertian, batwing, and side emitting. Various lenses can change these distribution patterns. Relatedly, LEDs have viewing angles beyond which the emitted light is no longer visible. For example, an LED with a beam pattern of 90 degrees would be viewable up to 45 degrees off angle from the central axis of the LED (Fig. 7). The relative intensity generally varies as the viewing angle deviates from the central axis.

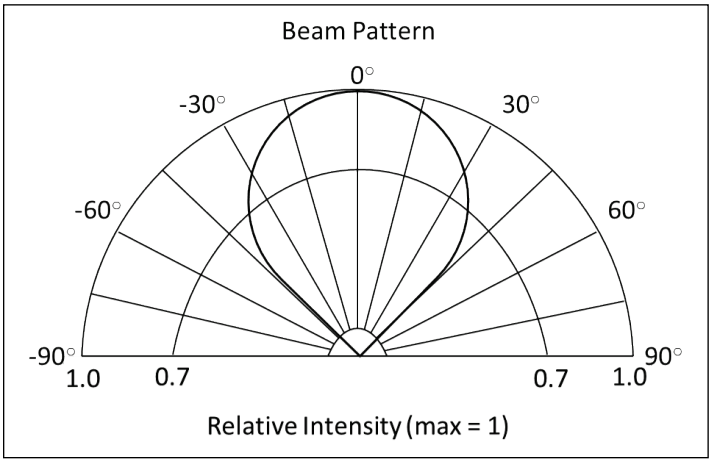


Figure 7. Idealized LED beam pattern

The LEDs selected for the study were of similar, though not identical, viewing angles because of their unavailability with the necessary luminous intensity and chromaticities. Participants viewed the LEDs from positions that varied less than 10 degrees from either side of the central axis of the LEDs obviating the problems associated with off angle viewing. To allow evaluation of potential differences in perception based on viewing angle, we recorded the precise position of each participant. The viewing distance was approximately 12 feet in direct line of sight with the device. Correct identification of the stimuli approached unity regardless of the position of the participants in relationship to the device.

Tests conducted in a darkened room identified sources of stray light, including reflections from within the device and stray emissions reflected through non-illuminated LEDs and their apertures. The solution was to solder the LED groups to black circuit boards and use black bushings to shield them from the output of the other LED groups. Bushings are hollow cylinders that are placed over the LED groups with one open end of the bushing resting against the circuit board and the other against the inside of the enclosure with the opening pointed through the aperture. This prohibits any light emissions in any location other than through the selected aperture.

Incandescent light. Individual aluminum project boxes mounted inside the larger case housed each incandescent lamp (Fig. 8). The project boxes accomplished the same objective as the bushings used with the LEDs in reducing or eliminating light emissions other than from the intended aperture.

Figure 8. Incandescent light source

The only chromaticity condition for the incandescent light source was monochromatic exposure. Given that there is only one general spectrum of emissions for the white and red lights of the legacy PAPI systems, the incandescent filters used in the legacy PAPI systems were obtained from the FAA Depot and used to ensure that the chromaticities met the FAA and International Civil Aviation Organization (ICAO) specifications for aviation red.

Viewing angle. The angular subtense of all stimuli was a critical consideration. Participants viewed the stimuli at the same viewing distance, making the angle subtended on the retina comparable. Because the LEDs were 5mm LEDs in a triangular group, the stimulus was roughly 11mm in diameter, accounting for the slight gap created by mounting tolerances. The apertures for the incandescent lights were also approximately 11mm to provide a comparable angular subtense for both lighting sources.

Software

Besides programming the Arduino Mega 2560 microprocessor, two additional software packages provided additional automation for the experimental protocol. These two software packages were necessary for the recording and playback of the instructions.

Audacity® (http://audacity.sourceforge.net/). Audacity®, a free, digital sound recording and editing application, enabled recording and editing the experimental instructions. Two students at Carnegie Mellon University originally developed this software, and it is under continual development by the open source community. Capabilities include cutting, pasting, and mixing with the ability to work with MP3, WAV, Ogg Vorbis, and AIFF files. Audacity® is available at http://audacity.sourceforge.net. As with the other software packages used in this experiment, Audacity® operates on Windows®, Mac OS X®, and Linux. For Windows 7 and 8 64-bit systems, it is recommended that the system have 4GB of RAM with a 2GHz processor, but only 2GB of RAM and 1GHz processor are required.

Online sources provided files containing some common words and phrases. The .wav files for the stimuli trials (e.g., Trial 1, Trial 2,…etc.) were downloaded from http://evolution.voxeo.com (http://evolution.voxeo.com/library/audio/prompts/numbers/index.jsp).

Gobetwino (http://mikmo.dk/gobetwino.html). Storage of files of the size required for the audio files of the instructions and sample numbers exceeds the capabilities of the Arduino. Additionally, the Arduino does not possess a native ability to play audio files. Therefore, Gobetwino provides a mechanism for the Arduino to communicate with a PC via a USB connection and utilize the storage and speakers available in the PC or external speakers wired through the PC.

Code for the Arduino microprocessor. As mentioned in the *Hardware* section, The Arduino microcontroller board uses an 8-bit, 16 MHz Atmel AVR microprocessor. The microprocessor is programmed using an open-source, Wiring-based language that is similar to C/C++. The software runs on Windows®, Mac OS X®, and Linux.

The full code for this device is contained in Appendix C. For a detailed explanation of programing with the Arduino, see http://arduino.cc/en/Tutorial.

DEVELOPMENT PROCESS

Design-Test-Build Cycle

Because we needed to produce a product with a limited time and budget, we used a cautious symbiosis between design and construction. We tested the implementation of each subcircuit and subassembly in reduced form and then expanded. For instance, a solderless breadboard provided the testing environment for control of the LED circuits with the Arduino and potentiometers (Fig. 9). Early tests involved only single LEDs follow by extension to multiple groups of LEDs with the verification of sections of code and circuit operation. Once a workable means of effectively matching the luminous intensities of various LEDs was derived and controlling their operation within the requirements of the experiment, then the circuits were implemented on the final circuit board.

Lights

The cycle of design, test, modify, retest, and build was used on the incandescent assemblies also. Although the incandescent lights were under analog control for luminance, several iterations were necessary to obtain combinations of bulbs and filters that resulted in the desired luminance and wavelengths. Colored lenses from fielded PAPI systems provided the filtering for the red incandescent lights. However, the transmissivity between the different types of filters was significantly different. The capabilities and perceptual characteristics of another instrument, the Signal Light Gun Test (SLGT) were also incorporated into the device as an extension of this effort. This created the challenge of using the red and white lights from the PAPI experimental device in conjunction with green lights that were specific to the SLGT because the range of luminance levels required for this combination of devices placed several of the incandescent bulbs at both high and low extremes of their viable emission levels. When operating at a low luminance level, the incandescent bulbs tended to begin emitting increasingly in the red portions of the spectrum, which moved their chromaticity outside the boundaries for aviation white.

During early testing, we discovered that even through the limited aperture, it was possible to detect the filament within and markers or texturing on some types of incandescent bulbs. Bulbs in which the filament was not perceivable and there were no markings, patterns, or textures helped reduced extraneous visual cues as potentially confounding factors. Another solution might have been to use diffusers or neutral density filters to mask such a factor; however, with our limited budget and time constraints, the easiest solution was to choose a coated bulb without markings.

When setting the luminance on the incandescent bulbs, there was the well-known challenge of changing intensity as the bulbs increased in operating temperature. Because each light would present only brief, 5-second exposures, it was important to set the intensities when the bulbs had been off for several minutes.

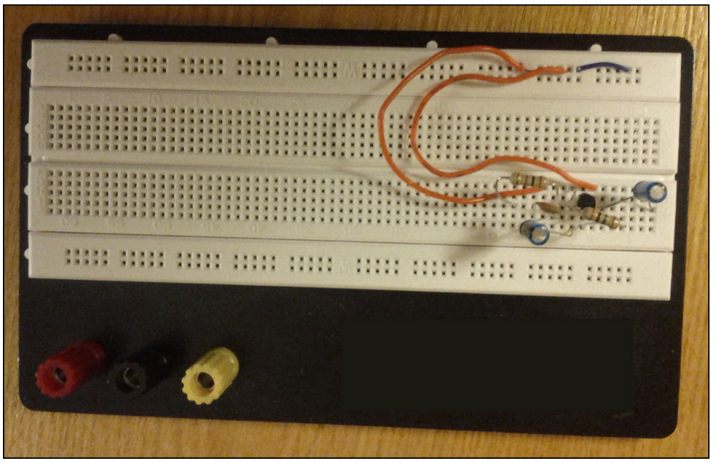


Figure 9. Solderless breadboard or protoboard

With only 14 Arduino pins capable of PWM, we initially thought that another method would be necessary to digitally adjust the luminance, but a brief script (Sample Code 1) was written and tested, which controlled the luminance levels strictly for digital outputs. This code accomplishes the same thing as a PWM-designated pin by defining, specifically, in the code the length of time for the HIGH and LOW states of each pin. For example:

```
Sample Code 1.
int ledPin8 = 8;
// LED connected to digital pin 8
void setup()
{
 pinMode(ledPin8, OUTPUT);
 // sets the digital pin as output
}
void loop()
{
 digitalWrite(ledPin8, HIGH);
 // sets the LED on
 delay(.1);
 digitalWrite(ledPin8, LOW);
 // sets the LED off
 delay(5);
}
```

Another method of controlling luminous intensity with software is through the use of the ShiftPWM library that creates outputs using shift registers. The code and additional information for the use of ShiftPWM is readily available online. There are additional integrated circuit options for control of luminous intensity. Dedicated LED drivers are available, including the LTC3220, TLC5940 16-Channel LED Driver, and LM3409HV, and the MY9221 in conjunction with external current limiting resistors. The ARD127D2P Rainbowduino ATmega328 board is an Arduino compatible board for controlling LED matrices with the use of MY9221 integrated circuits as the modulation controllers.

A SainSmart relay board controlled the incandescent bulbs, although we considered a relay board constructed from discrete components. The cost of a custom relay board, even without considering the labor involved, was significantly more than that of the SainSmart solution. Designers and developers will likely find that to be the case with many devices when weighing options between COTS devices (e.g., microcontroller boards, integrated circuits) and constructing custom devices. However, for those who are unable to find workable solutions commercially, there are often examples of workable circuits online.

Constant current devices and the current limiting variable resistors (potentiometers wired using only two of the leads) were considered for LED luminous intensity control. Several commercial devices are available for constant current control of LEDs; these include the LuxDrive™ 3023 Wired BuckPuck Modules, TLC5940 16-Channel LED Driver, and LM27964 White LED Driver System with I2C Compatible Brightness Control.

The main concern with control of the luminous intensity was stability and repeatability of the stimuli. We were also operating on a very constrained budget thus, we sought a solution that was both acceptable cost effective and that would provide a stable output. We took repeated measures of the output of the LEDs under the control of several mechanisms with an integrating sphere. The luminous intensity was repeatable across multiple days when using the potentiometers. Any future LED equipment in our laboratory will use constant-current devices.

Ultimately, the LED control circuit tests included three sample LEDs, each combined with a potentiometer, 200-ohm resistor, and a digital output from the Arduino Mega. This breadboard implementation vetted the ability to control luminous intensity, timing, and sequence of presentation. Following this test, we constructed the final circuits using perforated circuit boards. Perforated circuit boards facilitate modification of the circuit, if necessary, without the challenges of a custom etched circuit board.

Digital methods were also considered for control of the incandescent bulb luminous intensities. In the final device, triode for alternating current (TRIAC) dimmer switches controlled the incandescent bulbs. The intensities of the bulbs remained constant, with repeated measurements using the integrating sphere. The main drawback of using the potentiometers and dimmer switches for luminous intensity control is the labor-intensive task of manually setting each control.

Because of the deviations in LED output, even from the same manufacturing "bin," each LED had a different value assigned for the PWM control. This required measuring the output with the integrating sphere and manually adjusting the code and/or the individual potentiometer until the requisite luminous intensity was obtained.

Figure 10. Image of PAPI experimental device

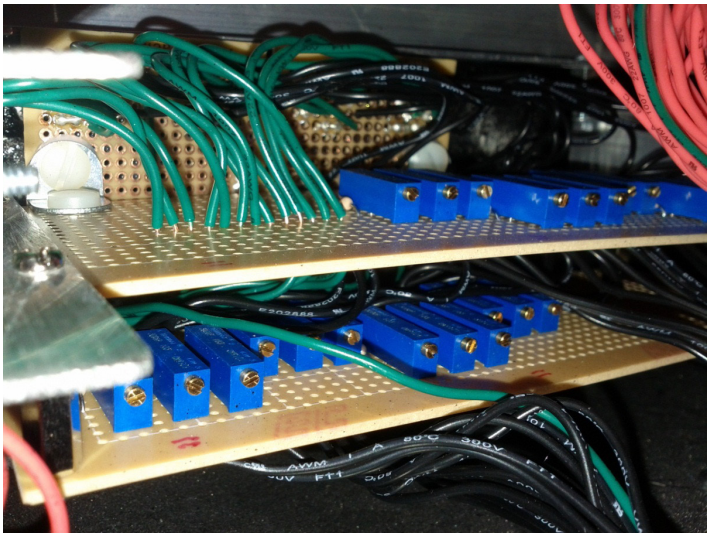


Figure 11. LED assembly mounting

The enclosures for the device yielded their own sets of challenges. Aluminum project boxes enclosed each incandescent bulb assembly, completely shielding each light source from the other bulbs (Fig. 8). Each enclosure had a 0.44-inch diameter aperture. The viewing distance of the participants from 15 feet is approximately equal to viewing a fielded PAPI device from 0.25 km in terms of angular subtense on the retina. Each individual TRIAC dimmer was mounted inside of a project box with the control knob positioned on the back for ease of access. The larger enclosure contained these assemblies, which had six apertures corresponding to the apertures in the light assembly boxes (Fig. 10).

Two of the smaller aluminum project boxes housed the red filters for the incandescent bulbs. The red and green filters for the FALANT test were circular and were mounted in plastic bushings, which were then pressed into the two holes in the middle of the box. The black exterior of the box reduced reflections.

The perforated circuit boards with the LED, 220-ohm resistors, and potentiometers were mounted inside the upper portion of the box using aluminum brackets and machine screws (Fig. 11).

There are additional assembly details beyond the scope of this document. However, a parts list and assembly sketches are included in Appendix D. Additional requests for information can be directed to the first author.

Lessons Learned

Given the time constraints of the project, there was not sufficient time to construct a prototype. The use of detailed engineering drawings avoided some of the challenges that can be encountered during construction. However, a 3D software program is highly advisable in such situations. A detailed 3D model would provide an opportunity to evaluate clearances and positioning. A Computer-Aided Design (CAD) program in association with Computer-Aided Manufacturing (CAM) tools would reduce the likelihood of late modifications and reworks. Some CAD programs are capable of generating an associated parts list that can facilitate the estimation of costs. Some open-source or freeware options for CAD programs are BRL-CAD, FreeCAD, pythonOCC, and the free version of DraftSight. The needs of the researcher generally determine the selection of a CAD program based on software strengths.

For designing and simulating the circuit functionality, there are several versions of SPICE (Simulation Program with Integrated Circuit Emphasis) available including the current version available from the University of California-Berkeley. LTspice™, Multisim™, Ngspice, TINA-TI™, and XSPICE are free and are relatively easy to use. These software tools are useful in evaluating circuit designs prior to moving to a physical breadboard to test circuits.

For determining the physical layout on a printed circuit board (PCB), Fritzing is an open-source software circuit and PCB layout tool designed with the inclusion of Arduino boards in the component library. It provides the ability to design at the schematic level or to use a virtual breadboard with virtual components. This can be particularly useful in gaining familiarization with the physical form of the Arduino boards and associated devices. Fritzing, DesignSpark PCB, ExpressPCB™, and similar software packages provide a manufacturing file (e.g., RS-274 Gerber file format, Excellon format) that can be sent to certain custom PCB manufacturers for drilling and etching at a nominal fee (often <$100) with no limit on the number of boards to order. Manufacturers can deliver the board to the designer in a bare state with just the circuit conduction paths, or traces, and no components soldered to the PCB. Some software packages, including DesignSpark PCB and ExpressPCB™ prepare a list of components for direct order of the discrete components (e.g., resistors, capacitors, ICs). For an additional fee, some companies will deliver the board fully assembled (e.g., Pad2Pad®, Sunstone Circuits®).

Another burgeoning technology that can be useful for building prototype devices is 3D printing, also called additive manufacturing (Pearce, 2012; Zhang, Anzalone, Faria, & Pearce, 2013). This form of manufacturing uses additive or depositional processes of spraying, or printing, layers of material in patterns defined by 3D design software. Many 3D printers can use a wide range of materials, including metals, thermoplastics, and ceramics. There are current development efforts with the goal of making PCBs using 3D printers. This form of printing could also be very useful for manufacturing other components for projects such as this one. For instance, the project enclosure and supporting hardware could be manufactured using 3D printing. The lighting device was relatively amenable to construction within the dimensions of commercially available products. However, a custom printed enclosure and associated peripherals might allow the creation of a system with less wasted internal space or in dimensions and shapes that depart from commercially available solutions. There are many other applications for scientific research for 3D printing, many of which remain unexplored.

Although this was a "clean sheet" design, most of the necessary components were available commercially. There are generally a number of viable options for any given component or subsystem, as well as macro-level design and implementation. In many instances, it is necessary to modify off-the-shelf components for uses other than for those originally intended. For instance, with the baffling for the LEDs, there were several options, including bushings, plastic pipe, etc. Such off-the-shelf devices are generally much less expensive than custom manufacturing or machining of components.

Each circuit and software concept was tested using a breadboard and the appropriate microcontroller or computer prior to construction. Testing the LEDs on a breadboard also provided initial evaluation of luminous intensities. This provided an opportunity to make adjustments and assure that all LED intensities were attainable to match the requirements.

Perforated circuit boards were the basis for all custom circuits. These boards, unlike custom etching, allowed the ability to reposition components, if necessary, with a minimal amount of effort. With custom etching, the circuit and the connections between components are manufactured into the circuit board. This approach requires a finalized circuit design, and any modifications after creating the board are difficult.

The device consisted of several different modules. Each incandescent light was a module capable of operation and replacement as a single unit. The LED lights, Arduino controller box, and relay board were all separate devices with discrete electrical connectors used for each modular component. This allowed the removal of individual segments of the device for modifications without the added complexity of complete disassembly.

Each incandescent light had its own separate sub-enclosure. These sub-enclosures included required filters and TRIAC dimmers. Placing each lighting source, or cluster of sources, in an independent sub-enclosure or module eliminated the need for baffling and provided the flexibility of moving the individual modules, as needed, within the larger enclosure.

We used a steel enclosure for durability, protection of the components, and heat resistance in the case of the incandescent light sources. If steel or aluminum is not an engineering requirement, the use of a plastic enclosure would present a material that is more amenable to drilling, machining, and modifying with commonly available tools. Metal and plastic project boxes are both readily available from electrical and electronics supply houses.

Modifications of plans, hardware, and software are common as with the design of most stimulus-presentation devices for research use, or probably most other unique devices. The challenges are those encountered with any prototype in terms of designing, implementing, testing, and modifying. If the necessary device can be purchased off-the-shelf—buy it; if not, expect to test and modify after the initial design.

CONCLUSIONS

Our goal was to assemble, quickly and inexpensively, a device to simulate current incandescent aviation lighting and proposed lighting with colored LEDs. The proliferation of open-source software and hardware, in addition to freeware, has created many opportunities for developing experimental devices that were not possible, or cost effective, a few years ago. Modest budgets and minimal access to technical support do not present the challenges that one could expect to encounter prior to the introduction of readily available controllers, software packages, and hardware.

The design, construction, and testing of this experimental device was accomplished with only a few weeks of effort for a materials cost of less than $1,000 USD. Commercial-off-the-shelf components, including the Arduino and relay boards, significantly shortened the development time by providing modules that were largely plug-and-play with a minimal amount of software coding and construction of custom hardware interfaces. Online developer communities often provide invaluable support, suggestions, and access to lessons learned with related devices. As science and technology continue to advance, the symbiotic relationship between hardware/software advancements and evolving research methods should open even more avenues for development.

REFERENCES

Bullough, J.D., Yuan, Z., & Rea, M.S. (2007). Perceived brightness of incandescent and LED aviation signal lights. *Aviation, Space, and Environmental Medicine, 78,* 893-900.

Christie, I.C. & Gianaros, P.J. (2013). PhysioScripts: An extensible, open source platform for the processing of physiological data. *Behavior Research Methods, 45,* 125-131. doi:10.3758/s13428-012-0233-x

D'Ausilio, A. (2012). Arduino: A low-cost multipurpose lab equipment. *Behavior Research Methods, 44,* 305-313. doi:10.3758/s13428-011-0163-z

Dixon, P. (2009). A hybrid approach to experimental control. *Behavior Research Methods, 41,* 615-622. doi:10.3758/BRM.41.3.615

Federal Aviation Administration (2011). Advisory Circular 150/5345-28G, Precision Approach Path Indicator (PAPI) systems. Retrieved from http://www.faa.gov/documentLibrary/media/Advisory_Circular/150_5345_28g.pdf

Hillenbrand, J.M. & Gayvert, R.T. (2005). Open source software for experiment design and control. *Journal of Speech, Language, and Hearing Research, 48,* 45–60. doi:10.1044/1092-4388(2005/005)

Mathot, S., Schreij, D., & Theeuwes, J. (2012). OpenSesame: An open-source, graphical experiment builder for the social sciences. *Behavior Research Methods, 44,* 314-324. doi:10.3758/s13428-011-0168-7

Milburn, N., & Gildea, K.M. (2012). Usability of light-emitting diode (LED) Precision Approach Path Indicator (PAPI) simulator by color-deficient and color-normal observers. *Aviation, Space, and Environmental Medicine, 38,* 318.

Pearce, J. (2012). Building research equipment with free, open-source hardware. *Science 337,* 1303-1304. doi:10.1126/science.1228183.

Reimers, S. & Stewart, N. (2009). Using SMS text messaging for teaching and data collection in the behavioral sciences. *Behavior Research Methods, 41,* 675-681. doi:10.3758/BRM.41.3.675

Stahl, C. (2006). Software for generating psychological experiments. *Experimental Psychology, 53,* 218-232. doi:10.1027/1618-3169.53.3.218

Stoet, G. (2010). PsyToolkit: A software package for programming psychological experiments using Linux. *Behavior Research Methods, 42,* 1096-104. doi: 10.3758/BRM.42.4.1096

Teikari, P., Najjar, R.P., Malkki, H., Knoblauch, K., Dumortier, D., Gronfier, C., & Cooper, H.M. (2012). An inexpensive Arduino-based LED stimulator system for vision research. *Journal of Neuroscience Methods, 211,* 227-236. doi: 10.1016/j.jneumeth.2012.09.012

von Bastian, C.C., Locher, A., & Ruflin, M. (2013). Tatool: A Java-based open-source programming framework for psychological studies. *Behavior Research Methods, 45,* 108-115. doi: 10.3758/s13428-012-0224-y

Zhang , C., Anzalone, N.C., Faria, R.P., & Pearce, J.M. (2013). Open-source 3D-printable optics equipment. *PLoS ONE, 8,* e59840. doi:10.1371/journal.pone.0059840

APPENDIX A
Sample of Single Board Contoller Options

Name	Processor	Frequency	Maker	Language
Arduino Uno	ATmega328P	16 MHz	Smart Projects (originally)	C/C++
Arduino Mega2560	Atmega2560	16 MHz	Smart Projects (originally)	C/C++
Arduino Micro	ATmega32u4	16 MHz	Smart Projects (originally)	C/C++
SainSmart UNO	ATmega328	16 MHz	SainSmart	C/C++
Raspduino	ATmega238	16 MHz	Bitwizard	C/C++
Seeeduino v3.0	ATmega328P	16 MHz	SeeedStudio	C/C++
Lightuino	ATmega328P	16 MHz	Toasted Circuits	C/C++
Boarduino	ATmega168/ ATmega328	16 MHz	Adafruit	C/C++
Freeduino USB Mega 2560	Atmega2560	16 MHz	Bhasha Technologies	C/C++
TinyDuino	ATmega328P	8 MHz	TinyCircuits	C/C++
Microchip chipKIT Uno32	PIC32MX320F128	80 MHz	Digilent	C/C++
Netduino Plus 2	STM32F405RG	168 MHZ		.NET Micro Framework
BASIC Stamp 2	2 x ATtiny13		Parallax	PBASIC
Raspberry Pi	ARM1176JZF-S	700 MHz	Raspberry Pi Foundation	Multiple operating system/multiple language
BeagleBone Black	AM3359	1 GHz	TI, Digi-Key, & element14	Multiple operating system/multiple language
UDOO	Freescale Cortex-A9 i.MX.6 & Atmel SAM3X8E	1 GHz & 84 MHz	Aidilab & SECO USA Inc.	Multiple operating system/multiple language

APPENDIX B
LED Specifications

Part Number	Color	Color Temp (K)	λ_D (nm)	Luminous Intensity (mcd)	Viewing Angle	Forward Voltage (Typical)	Max Current (mA) w/220 Ω Resistor
LDF200-XIW-22-LL	White	3000		1000	105	3.2	8.4
710TSW4D	White	5500		1000	60	3.8	5.5
LDF200-0CW-27-LL	White	8000		1100	90	3.2	8.3
710JPF4D	Red		630	1300	60	2.4	11.9
LD200CWR3KF	Red		642	1000	20	1.9	14.1

APPENDIX C
PAPI Device Arduino Code

Code

```
int ledPin1 = 23;
int ledPin2 = 25;
int ledPin3 = 2;
int ledPin4 = 3;
int ledPin5 = 4;
int ledPin6 = 5;
int ledPin7 = 6;
int ledPin8 = 7;
int ledPin9 = 8;
int ledPin10 = 9;
int ledPin11 = 10;
int ledPin12 = 11;
int ledPin13 = 12;
int ledPin14 = 13;
int ledPin15 = 27;
int ledPin16 = 29;

int incPin1 = 31;
int incPin2 = 33;
int incPin3 = 35;
int incPin4 = 37;
int incPin5 = 39;
int incPin6 = 41;
int incPin7 = 43;
int incPin8 = 45;
int incPin9 = 47;
int incPin10 = 49;
int incPin11 = 51;
int incPin12 = 53;



 //sets the digital pins as led
outputs
void setup()
 {
 Serial.begin(9600);
 pinMode(ledPin1, OUTPUT);
 pinMode(ledPin2, OUTPUT);
 pinMode(ledPin3, OUTPUT);
 pinMode(ledPin4, OUTPUT);
 pinMode(ledPin5, OUTPUT);
 pinMode(ledPin6, OUTPUT);
 pinMode(ledPin7, OUTPUT);
 pinMode(ledPin8, OUTPUT);
 pinMode(ledPin9, OUTPUT);
 pinMode(ledPin10,
OUTPUT);
 pinMode(ledPin11,
OUTPUT);
 pinMode(ledPin12,
OUTPUT);
 pinMode(ledPin13,
OUTPUT);
 pinMode(ledPin14,
OUTPUT);
 pinMode(ledPin15,
OUTPUT);
 pinMode(ledPin16,
OUTPUT);

 // set the digital pins as
incandescent outputs
 pinMode(incPin1, OUTPUT);
 pinMode(incPin2, OUTPUT);
 pinMode(incPin3, OUTPUT);
 pinMode(incPin4, OUTPUT);
 pinMode(incPin5, OUTPUT);
 pinMode(incPin6, OUTPUT);
 pinMode(incPin7, OUTPUT);
 pinMode(incPin8, OUTPUT);
 pinMode(incPin9, OUTPUT);
 pinMode(incPin10,
OUTPUT);
 pinMode(incPin11,
OUTPUT);
 pinMode(incPin12,
OUTPUT);
 }

void loop ()
{

// SLGT sequence 1000'

Pause30();
SLGTOV();
Pause30();
SLGT1000();
Pause06();
WAV1();
G1000();
WAV2();
W1000();
WAV3();
G1000();
WAV4();
W1000();
WAV5();
R1000();
WAV6();
R1000();

// SLGT sequence 1500'

SLGT1500();
Pause06();
WAV1();
R1500();
WAV2();
G1500();
WAV3();
G1500();
WAV4();
W1500();
WAV5();
W1500();
```

```
WAV6();
R1500();
PauseInc20();

// Incandescent sequence

INCSAMP();
ExampleInc();
Pause20();
INCTEST();
Pause06();
WAV1();
RE1WE2();
WAV2();
WE2WE8();
WAV3();
RE1RD4();
WAV4();
WED7WD3();
WAV5();
WE2RD4();
WAV6();
RE1RD4();
WAV7();
WE2WD3();
WAV8();
RE1RD4();
WAV9();
WD3RD4();
WAV10();
WED7RD4();
WAV11();
RE1WE8();
WAV12();
WE2WD3();
WAV13();
RE1RD4();
WAV14();
RE1WE2();
WAV15();
WE2WE8();
WAV16();
RE1RD4();
WAV17();
WED7WD3();
WAV18();
WE2RD4();
WAV19();
WE2WD3();
WAV20();
RE1RD4();
WAV21();
WD3RD4();
WAV22();
WED7RD4();
WAV23();
RE1WE8();
WAV24();
WE2WD3();
WAV25();
RE1RD4();
WAV26();
RE1RD4();
PauseInc20();

// led sequence

LEDSAMP();
Example();
Pause20();
LEDTEST();
Pause06();
WAV1();
MW1MW4();
WAV2();
HW13HW16();
WAV3();
HR7HW8();
WAV4();
HW13HW16();
WAV5();
MR2MR3();
WAV6();
MR10MR11();
WAV7();
HW13HR14();
WAV8();
HW5HR6();
WAV9();
HR7HW8();
WAV10();
MR11MW12();
WAV11();
MR3MW4();
WAV12();
MR10MW12();
WAV13();
HW13HW16();
WAV14();
HR14HR15();
WAV15();
HW13HW16();
WAV16();
HW5HW8();
WAV17();
MR10MR11();
WAV18();
MW1MW4();
WAV19();
HR6HW8();
WAV20();
HR14HW16();
WAV21();
MW1MR3();
WAV22();
MW9MR10();
WAV23();
HR15MW16();
WAV24();
HR6HW8();
WAV25();
MW1MR2();
WAV26();
MW1MW4();
WAV27();
HW5HW8();
WAV28();
MW9MR11();
WAV29();
MR2MR3();
WAV30();
MR10MR11();
WAV31();
HW13HR14();
WAV32();
MW1MR3();
WAV33();
MW9MR11();
WAV34();
HR14HR15();
WAV35();
MR3MW4();
WAV36();
HW5HW8();
```

```
WAV37();
HR14HR15();
WAV38();
MW9MW12();
WAV39();
MW1MW4();
WAV40();
MR10MW12();
WAV41();
HW5HR6();
WAV42();
HW13HR15();
WAV43();
MR10MR11();
WAV44();
MW1MR2();
WAV45();
HW13HR15();
WAV46();
HR6HR7();
WAV47();
MR2MR3();
WAV48();
MW9MW12();
WAV49();
HW5HR7();
WAV50();
HW5HW8();
WAV51();
HR15HW16();
WAV52();
HR6HR7();
WAV53();
MR2MR3();
WAV54();
MW9MR10();
WAV55();
HW5HR7();
WAV56();
MW9MW12();
WAV57();
HR14HR15();
WAV58();
MR2MW4();
WAV59();
HR6HR7();
WAV60();
HR14HW16();
WAV61();
MW9MW12();
WAV62();
MR2MW4();
WAV63();
HR6HR7();
WAV64();
MR11MW12();

// Extended pause (2 hours)
at the end of the experiment

End();
}
// SLGT pin timing subroutines

void  G1000(){digitalWrite(incPin6,HIGH);delay(5000);digitalWrite(incPin6,LOW);delay(5000);}
void  R1000(){digitalWrite(incPin11,HIGH);delay(5000);digitalWrite(incPin11,LOW);delay(5000);}
void  W1000(){digitalWrite(incPin9,HIGH);delay(5000);digitalWrite(incPin9,LOW);delay(5000);}
void  G1500(){digitalWrite(incPin5,HIGH);delay(5000);digitalWrite(incPin5,LOW);delay(5000);}
void  R1500(){digitalWrite(incPin12,HIGH);delay(5000);digitalWrite(incPin12,LOW);delay(5000);}
void  W1500(){digitalWrite(incPin10,HIGH);delay(5000);digitalWrite(incPin10,LOW);delay(5000);}
void  Pause30(){digitalWrite(incPin10,LOW);delay(0);digitalWrite(incPin10,LOW);delay(30000);}

// incandescent pin timing subroutines

void
ExampleInc(){digitalWrite(incPin2,HIGH);digitalWrite(incPin3,HIGH);delay(5000);digitalWrite(incPin2,LO
W);digitalWrite(incPin3,LOW);delay(15000);}

void
RE1WE2(){digitalWrite(incPin1,HIGH);digitalWrite(incPin2,HIGH);delay(5000);digitalWrite(incPin1,LOW);
digitalWrite(incPin2,LOW);delay(5000);}
void
WE2WE8(){digitalWrite(incPin2,HIGH);digitalWrite(incPin8,HIGH);delay(5000);digitalWrite(incPin2,LOW)
;digitalWrite(incPin8,LOW);delay(5000);}
void
RE1RD4(){digitalWrite(incPin1,HIGH);digitalWrite(incPin4,HIGH);delay(5000);digitalWrite(incPin1,LOW);
digitalWrite(incPin4,LOW);delay(5000);}
```

```
void
WED7WD3(){digitalWrite(incPin7,HIGH);digitalWrite(incPin3,HIGH);delay(5000);digitalWrite(incPin7,LO
W);digitalWrite(incPin3,LOW);delay(5000);}
void
WE2RD4(){digitalWrite(incPin2,HIGH);digitalWrite(incPin4,HIGH);delay(5000);digitalWrite(incPin2,LOW);
digitalWrite(incPin4,LOW);delay(5000);}
void
WE2WD3(){digitalWrite(incPin2,HIGH);digitalWrite(incPin3,HIGH);delay(5000);digitalWrite(incPin2,LOW)
;digitalWrite(incPin3,LOW);delay(5000);}
void
WD3RD4(){digitalWrite(incPin3,HIGH);digitalWrite(incPin4,HIGH);delay(5000);digitalWrite(incPin3,LOW)
;digitalWrite(incPin4,LOW);delay(5000);}
void
WED7RD4(){digitalWrite(incPin7,HIGH);digitalWrite(incPin4,HIGH);delay(5000);digitalWrite(incPin7,LOW
);digitalWrite(incPin4,LOW);delay(5000);}
void
RE1WE8(){digitalWrite(incPin1,HIGH);digitalWrite(incPin8,HIGH);delay(5000);digitalWrite(incPin1,LOW);
digitalWrite(incPin8,LOW);delay(5000);}
void
PauseInc20(){digitalWrite(incPin1,LOW);digitalWrite(incPin8,LOW);delay(0);digitalWrite(incPin1,LOW);di
gitalWrite(incPin8,LOW);delay(20000);}

// led pin timing subroutines

void
Example(){analogWrite(ledPin9,85);analogWrite(ledPin12,85);delay(5000);analogWrite(ledPin9,0);analo
gWrite(ledPin12,0);delay(15000);}

void
HR14HR15(){analogWrite(ledPin14,255);analogWrite(ledPin15,255);delay(5000);analogWrite(ledPin14,0
);analogWrite(ledPin15,0);delay(5000);}
void
HR14HW16(){analogWrite(ledPin14,255);analogWrite(ledPin16,255);delay(5000);analogWrite(ledPin14,
0);analogWrite(ledPin16,0);delay(5000);}
void
HR15HW16(){analogWrite(ledPin15,255);analogWrite(ledPin16,255);delay(5000);analogWrite(ledPin15,
0);analogWrite(ledPin16,0);delay(5000);}
void
HR15MW16(){analogWrite(ledPin15,255);analogWrite(ledPin16,255);delay(5000);analogWrite(ledPin15,
0);analogWrite(ledPin16,0);delay(5000);}
void
HR6HR7(){analogWrite(ledPin6,175);analogWrite(ledPin7,200);delay(5000);analogWrite(ledPin6,0);anal
ogWrite(ledPin7,0);delay(5000);}
void
HR6HW8(){analogWrite(ledPin6,175);analogWrite(ledPin8,255);delay(5000);analogWrite(ledPin6,0);anal
ogWrite(ledPin8,0);delay(5000);}
```

```
void
HR7HW8(){analogWrite(ledPin7,200);analogWrite(ledPin8,255);delay(5000);analogWrite(ledPin7,0);anal
ogWrite(ledPin8,0);delay(5000);}
void
HW13HR14(){analogWrite(ledPin13,70);analogWrite(ledPin14,255);delay(5000);analogWrite(ledPin13,0)
;analogWrite(ledPin14,0);delay(5000);}
void
HW13HR15(){analogWrite(ledPin13,70);analogWrite(ledPin15,255);delay(5000);analogWrite(ledPin13,0)
;analogWrite(ledPin15,0);delay(5000);}
void
HW13HW16(){analogWrite(ledPin13,70);analogWrite(ledPin16,255);delay(5000);analogWrite(ledPin13,
0);analogWrite(ledPin16,0);delay(5000);}
void
HW5HR6(){analogWrite(ledPin5,140);analogWrite(ledPin6,175);delay(5000);analogWrite(ledPin5,0);anal
ogWrite(ledPin6,0);delay(5000);}
void
HW5HR7(){analogWrite(ledPin5,140);analogWrite(ledPin7,200);delay(5000);analogWrite(ledPin5,0);anal
ogWrite(ledPin7,0);delay(5000);}
void
HW5HW8(){analogWrite(ledPin5,140);analogWrite(ledPin8,255);delay(5000);analogWrite(ledPin5,0);an
alogWrite(ledPin8,0);delay(5000);}
void
MR10MR11(){analogWrite(ledPin10,85);analogWrite(ledPin11,85);delay(5000);analogWrite(ledPin10,0);
analogWrite(ledPin11,0);delay(5000);}
void
MR10MW12(){analogWrite(ledPin10,85);analogWrite(ledPin12,75);delay(5000);analogWrite(ledPin10,0)
;analogWrite(ledPin12,0);delay(5000);}
void
MR11MW12(){analogWrite(ledPin11,85);analogWrite(ledPin12,75);delay(5000);analogWrite(ledPin11,0)
;analogWrite(ledPin12,0);delay(5000);}
void
MR2MR3(){analogWrite(ledPin2,255);analogWrite(ledPin3,255);delay(5000);analogWrite(ledPin2,0);ana
logWrite(ledPin3,0);delay(5000);}
void
MR2MW4(){analogWrite(ledPin2,255);analogWrite(ledPin4,160);delay(5000);analogWrite(ledPin2,0);an
alogWrite(ledPin4,0);delay(5000);}
void
MR3MW4(){analogWrite(ledPin3,255);analogWrite(ledPin4,160);delay(5000);analogWrite(ledPin3,0);an
alogWrite(ledPin4,0);delay(5000);}
void
MW1MR2(){analogWrite(ledPin1,255);analogWrite(ledPin2,255);delay(5000);analogWrite(ledPin1,0);an
alogWrite(ledPin2,0);delay(5000);}
void
MW1MR3(){analogWrite(ledPin1,255);analogWrite(ledPin3,255);delay(5000);analogWrite(ledPin1,0);an
alogWrite(ledPin3,0);delay(5000);}
void
MW1MW4(){analogWrite(ledPin1,255);analogWrite(ledPin4,160);delay(5000);analogWrite(ledPin1,0);a
nalogWrite(ledPin4,0);delay(5000);}
```

```
void
MW9MR10(){analogWrite(ledPin9,85);analogWrite(ledPin10,85);delay(5000);analogWrite(ledPin9,0);an
alogWrite(ledPin10,0);delay(5000);}
void
MW9MR11(){analogWrite(ledPin9,85);analogWrite(ledPin11,85);delay(5000);analogWrite(ledPin9,0);an
alogWrite(ledPin11,0);delay(5000);}
void
MW9MW12(){analogWrite(ledPin9,85);analogWrite(ledPin12,75);delay(5000);analogWrite(ledPin9,0);a
nalogWrite(ledPin12,0);delay(5000);}

// Pause for 20 seconds
void
Pause20(){analogWrite(ledPin9,0);analogWrite(ledPin12,0);delay(0);analogWrite(ledPin9,0);analogWrite
(ledPin12,0);delay(20000);}
void
Pause06(){analogWrite(ledPin9,0);analogWrite(ledPin12,0);delay(0);analogWrite(ledPin9,0);analogWrite
(ledPin12,0);delay(6000);}

// Audio subroutines
void WAV1(){Serial.println("#S|WAV1|[]#");}
void WAV2(){Serial.println("#S|WAV2|[]#");}
void WAV3(){Serial.println("#S|WAV3|[]#");}
void WAV4(){Serial.println("#S|WAV4|[]#");}
void WAV5(){Serial.println("#S|WAV5|[]#");}
void WAV6(){Serial.println("#S|WAV6|[]#");}
void WAV7(){Serial.println("#S|WAV7|[]#");}
void WAV8(){Serial.println("#S|WAV8|[]#");}
void WAV9(){Serial.println("#S|WAV9|[]#");}
void WAV10(){Serial.println("#S|WAV10|[]#");}
void WAV11(){Serial.println("#S|WAV11|[]#");}
void WAV12(){Serial.println("#S|WAV12|[]#");}
void WAV13(){Serial.println("#S|WAV13|[]#");}
void WAV14(){Serial.println("#S|WAV14|[]#");}
void WAV15(){Serial.println("#S|WAV15|[]#");}
void WAV16(){Serial.println("#S|WAV16|[]#");}
void WAV17(){Serial.println("#S|WAV17|[]#");}
void WAV18(){Serial.println("#S|WAV18|[]#");}
void WAV19(){Serial.println("#S|WAV19|[]#");}
void WAV20(){Serial.println("#S|WAV20|[]#");}
void WAV21(){Serial.println("#S|WAV21|[]#");}
void WAV22(){Serial.println("#S|WAV22|[]#");}
void WAV23(){Serial.println("#S|WAV23|[]#");}
void WAV24(){Serial.println("#S|WAV24|[]#");}
void WAV25(){Serial.println("#S|WAV25|[]#");}
void WAV26(){Serial.println("#S|WAV26|[]#");}
void WAV27(){Serial.println("#S|WAV27|[]#");}
void WAV28(){Serial.println("#S|WAV28|[]#");}
void WAV29(){Serial.println("#S|WAV29|[]#");}
```

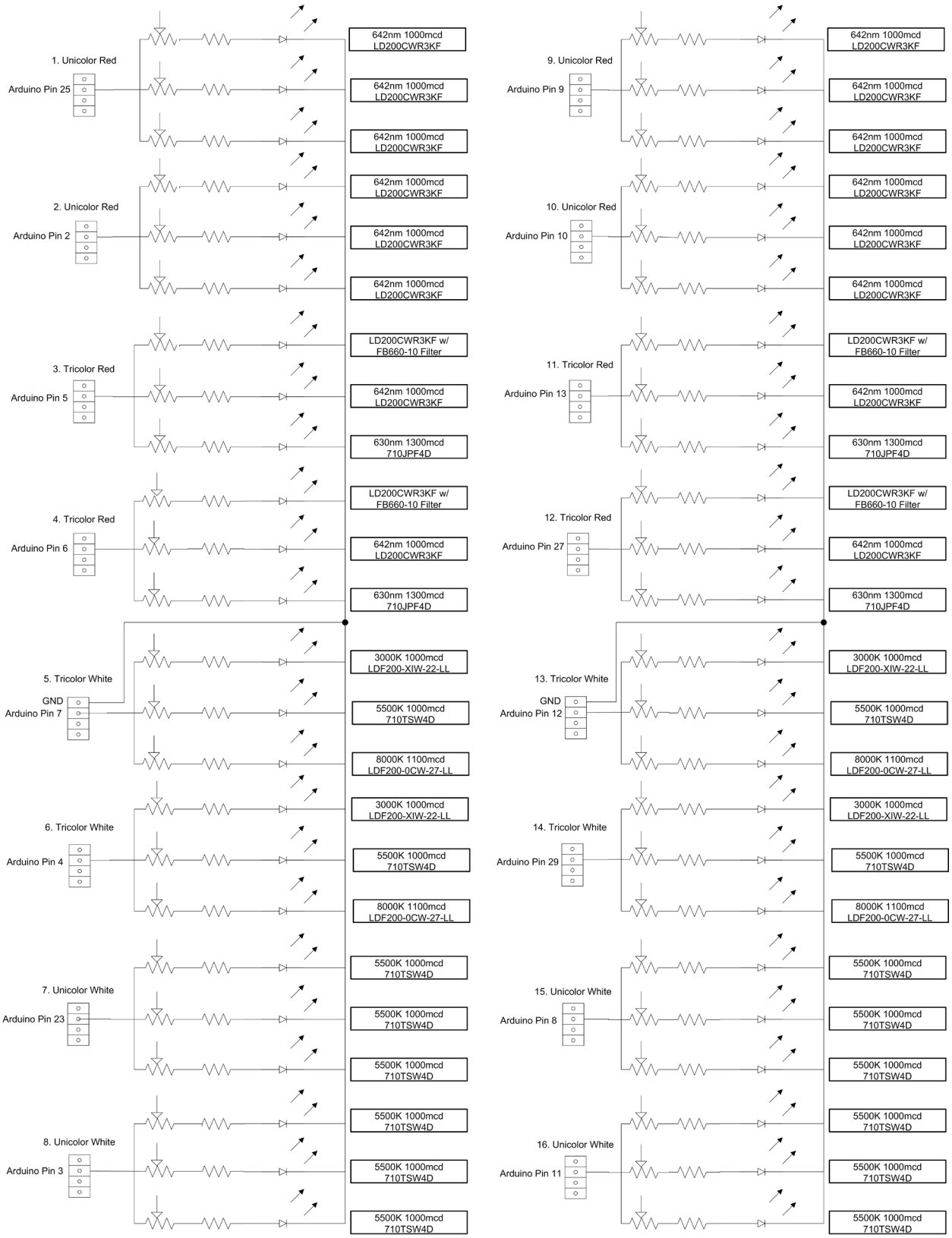
```
void WAV30(){Serial.println("#S|WAV30|[]#");}
void WAV31(){Serial.println("#S|WAV31|[]#");}
void WAV32(){Serial.println("#S|WAV32|[]#");}
void WAV33(){Serial.println("#S|WAV33|[]#");}
void WAV34(){Serial.println("#S|WAV34|[]#");}
void WAV35(){Serial.println("#S|WAV35|[]#");}
void WAV36(){Serial.println("#S|WAV36|[]#");}
void WAV37(){Serial.println("#S|WAV37|[]#");}
void WAV38(){Serial.println("#S|WAV38|[]#");}
void WAV39(){Serial.println("#S|WAV39|[]#");}
void WAV40(){Serial.println("#S|WAV40|[]#");}
void WAV41(){Serial.println("#S|WAV41|[]#");}
void WAV42(){Serial.println("#S|WAV42|[]#");}
void WAV43(){Serial.println("#S|WAV43|[]#");}
void WAV44(){Serial.println("#S|WAV44|[]#");}
void WAV45(){Serial.println("#S|WAV45|[]#");}
void WAV46(){Serial.println("#S|WAV46|[]#");}
void WAV47(){Serial.println("#S|WAV47|[]#");}
void WAV48(){Serial.println("#S|WAV48|[]#");}
void WAV49(){Serial.println("#S|WAV49|[]#");}
void WAV50(){Serial.println("#S|WAV50|[]#");}
void WAV51(){Serial.println("#S|WAV51|[]#");}
void WAV52(){Serial.println("#S|WAV52|[]#");}
void WAV53(){Serial.println("#S|WAV53|[]#");}
void WAV54(){Serial.println("#S|WAV54|[]#");}
void WAV55(){Serial.println("#S|WAV55|[]#");}
void WAV56(){Serial.println("#S|WAV56|[]#");}
void WAV57(){Serial.println("#S|WAV57|[]#");}
void WAV58(){Serial.println("#S|WAV58|[]#");}
void WAV59(){Serial.println("#S|WAV59|[]#");}
void WAV60(){Serial.println("#S|WAV60|[]#");}
void WAV61(){Serial.println("#S|WAV61|[]#");}
void WAV62(){Serial.println("#S|WAV62|[]#");}
void WAV63(){Serial.println("#S|WAV63|[]#");}
void WAV64(){Serial.println("#S|WAV64|[]#");}
void SLGTOV(){Serial.println("#S|SLGTOV|[]#");}
void LEDTEST(){Serial.println("#S|LEDTEST|[]#");}
void INCTEST(){Serial.println("#S|INCTEST|[]#");}
void LEDSAMP(){Serial.println("#S|LEDSAMP|[]#");}
void INCSAMP(){Serial.println("#S|INCSAMP|[]#");}
void SLGT1000(){Serial.println("#S|SLGT1000|[]#");}
void SLGT1500(){Serial.println("#S|SLGT1500|[]#");}

//delay after experiment run
void
End(){analogWrite(ledPin9,0);analogWrite(ledPin12,0);delay(360000);analogWrite(ledPin9,0);analogWri
te(ledPin12,0);delay(360000);}
```

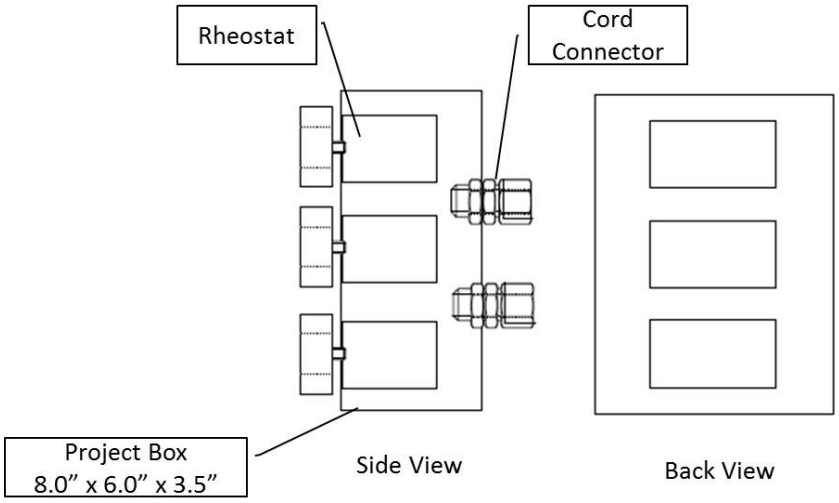
APPENDIX D
Assembly Details
Arduino Pin Assignments

Arduino Pin Assignment	Code	Color
int ledPin1 = 23;	MW1	Unicolor White
int ledPin2 = 25;	MR2	Unicolor Red
int ledPin3 = 2;	MR3	Unicolor Red
int ledPin4 = 3;	MW4	Unicolor White
int ledPin5 = 4;	HW5	Tricolor White
int ledPin6 = 5;	HR6	Tricolor Red
int ledPin7 = 6;	HR7	Tricolor Red
int ledPin8 = 7;	HW8	Tricolor White
int ledPin9 = 8;	MW9	Unicolor White
int ledPin10 = 9;	MR10	Unicolor Red
int ledPin11 = 10;	MR11	Unicolor Red
int ledPin12 = 11;	MW12	Unicolor White
int ledPin13 = 12;	HW13	Tricolor White
int ledPin14 = 13;	HR14	Tricolor Red
int ledPin15 = 27;	HR15	Tricolor Red
int ledPin16 = 29;	HW16	Tricolor White
int incPin1 = 31;	RE1	Incandescent Red
int incPin2 = 33;	WE2	Incandescent White
int incPin3 = 35;	WD3	Incandescent White
int incPin4 = 37;	RD4	Incandescent Red
int incPin5 = 39;	G1500	Incandescent Green
int incPin6 = 41;	G1000	Incandescent Green
int incPin7 = 43;	WED7	Incandescent White
int incPin8 = 45;	WE8	Incandescent White
int incPin9 = 47;	W1000	Incandescent White
int incPin10 = 49;	W1500	Incandescent White
int incPin11 = 51;	R1000	Incandescent Red
int incPin12 = 53;	R1500	Incandescent Red

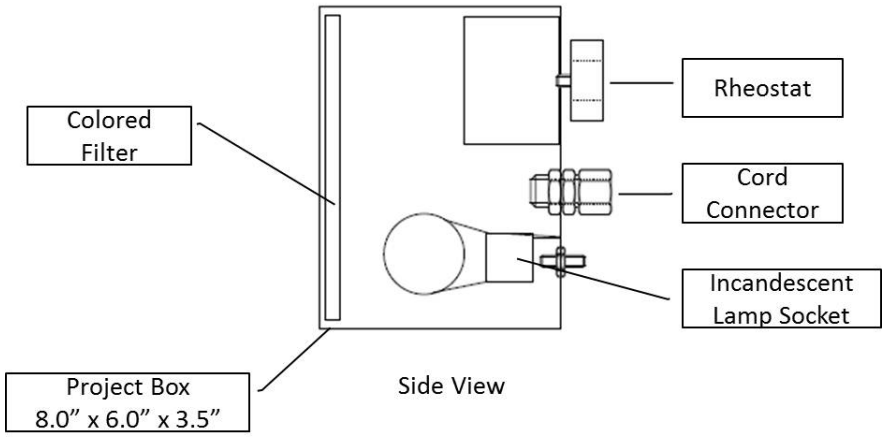
LED Schematic



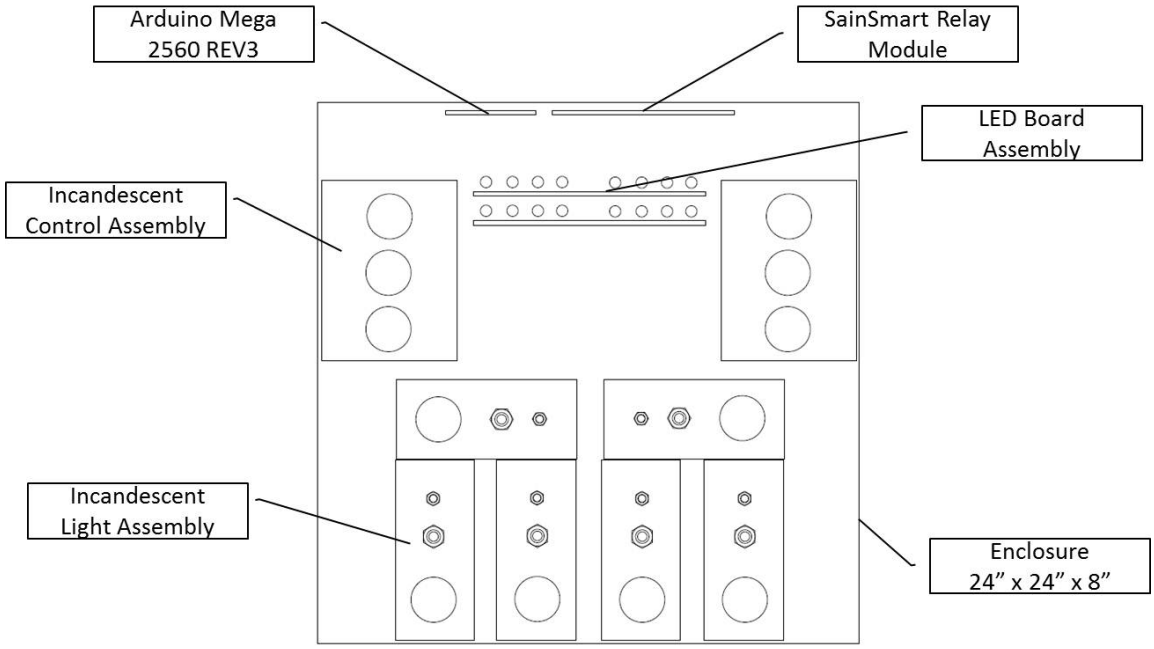
Incandescent Control Assembly



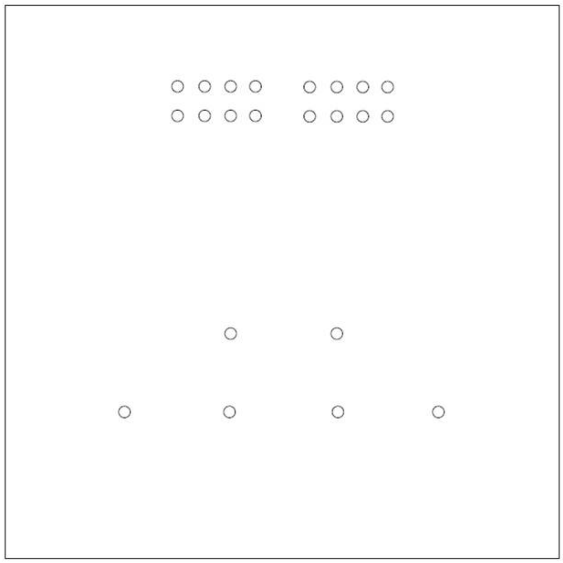
Incandescent Light Assembly



PAPI Device Assembly



Back View



Front View

Parts List

Item	Quantity	Unit Price	Price	Link
Enclosure, 24"x24"x8", Item 6JYW5	1	$ 153.50	$153.50	http://www.grainger.com/Grainger/WIEGMANN-Enclosure-6JYW5?Pid=search
Mini Project Bud Box - 8.00" x 6.00" x 3.50" 28-8579	8	$ 17.49	$139.92	http://electronics.mcmelectronics.com/search.php?Ntt=project+boxes&No=60&Ns=0
AC to DC Wall Adapter 12V 1A	1	$ 14.95	$ 14.95	http://www.jameco.com/webapp/wcs/stores/servlet/Product_10001_10001_170245_-1
Power Cord, 6', AWG 18/3, Detachable	1	$ 4.49	$ 4.49	http://www.jameco.com/webapp/wcs/stores/servlet/Product_10001_10001_161761_-1
IEC Power Receptacle 3-Prong CAT# ACS-48	1	$ 1.25	$ 1.25	http://www.allelectronics.com/make-a-store/item/ACS-48/IEC-POWER-RECEPTACLE/1.html
ARLINGTON CordConnector, .2-.472", 1.963", Nylon	10	$ 2.91	$ 29.10	http://www.grainger.com/Grainger/ARLINGTON-Strain-Relief-Cord-Connector-4JWN3?Pid=search
15-Position Female Solder D-Sub Connector 215FE-ND	2	$ 1.36	$ 2.72	http://www.digikey.com/product-detail/en/171-015-203L001/215FE-ND/858126
15-Position Male Solder D-Sub Connector 215ME-ND	2	$ 1.10	$ 2.20	http://www.digikey.com/product-detail/en/171-015-103L001/215ME-ND/858118
15-Position D-Sub Connector Hood 956-15SPGE-ND	4	$ 0.74	$ 2.96	http://www.digikey.com/product-detail/en/956-015-010R031/956-15SPGE-ND/1632184
Dual Board 213 Holes Model: 276-148	4	$ 1.99	$ 7.96	http://www.radioshack.com/product/index.jsp?productId=2104052
Printed Circuit Board 550 Connect Points Model: 276-170	1	$ 2.99	$ 2.99	http://www.radioshack.com/product/index.jsp?productId=2102846
12-Position European-Style Terminal Strip ED2998-ND	3	$ 1.10	$ 3.30	http://www.digikey.com/product-detail/en/ES0800%2F06DSFB/ED2998-ND/2720745
SYLVANIA 200-Watt A21 Soft White Incandescent Light Bulbs, 3650 Lumens	12	$ 2.46	$ 29.52	http://www.lowes.com/ProductDisplay?partNumber=76508-3-13103&langId=-1&storeId=10151&productId=1100573&catalogId=10051&cmRelshp=req&rel=nofollow&cId=PDIO1
SERVALITE 660-Watt Hard-Wired Lamp Socket	6	$ 2.94	$ 17.64	http://www.lowes.com/pd_75120-37672-884414_4294722560__?productId=3379426&Ns=p_product_qty_sales_dollar|1&pl=1&currentURL=%3FNs%3Dp_product_qty_sales_dollar%7C1&facetInfo=

Item	Quantity	Unit Price	Price	Link
Thorlabs FB660-10 - Ø1" Bandpass Filter, CWL = 660 ± 2 nm	2	$ 84.67	$169.34	http://www.thorlabs.com/thorproduct.cfm?partnumber=FB660-10
Lutron 600-Watt Single Phase Rotary Dimmer Item #: 211635	12	$ 11.96	$143.52	http://www.lowes.com/pd_211635-539-DNG-603PH-DK_4294821967_4294937087_?productId=3363668&Ns=p_product_price|1&pl=1&currentURL=%2Fpl_Dimmers%2B_4294821967_4294937087_%3FNs%3Dp_product_price%7C1%26page%3D7
Metal Standoffs with Screws (4-Pack) Model: 276-195	4	$ 1.99	$ 7.96	http://www.radioshack.com/product/index.jsp?productId=2102848
10k Ohm 3-4 Watt 15 Turn Cermet Potentiometer	48	$ 1.09	$ 52.32	http://www.jameco.com/webapp/wcs/stores/servlet/Product_10001_10001_41822_-1
Heat Shrink Tubing, 1/16" x 4'	1	$ 1.19	$ 1.19	http://www.jameco.com/webapp/wcs/stores/servlet/Product_10001_10001_419127_-1
Heat Shrink Tubing, 1/8" x 4'	1	$ 1.39	$ 1.39	http://www.jameco.com/webapp/wcs/stores/servlet/Product_10001_10001_419160_-1
LDF200-XIW-22-LL 3000K, Cylindrical Body, 1000mcd, 105 Degrees, 20mA	4	$ 0.42	$ 1.68	http://www.ledtronics.com/Products/ProductsDetails.aspx?WP=1794
710TSW4D 5500K, Cylindrical Body, 1000 mcd, 60 Degrees, 30mA	16	$ 67	$ 10.72	http://www.lc-led.com/View/itemNumber/463
LDF200-0CW-27-LL 8000K, Cylindrical Body, 1100mcd, 90 Degrees, 20mA	4	$ 0.68	$ 2.72	http://www.ledtronics.com/Products/ProductsDetails.aspx?WP=1788
710JPF4D, Cylindrical, 630nm, 1,300mcd, 60 Degrees, 30mA	4	$ 0.75	$ 3.00	http://www.lc-led.com/View/itemNumber/467
LD200CWR3KF Domed, 642nm, 1000mcd, 20 Degrees, 30mA	20	$ 0.24	$ 4.80	http://www.ledtronics.com/Products/ProductsDetails.aspx?WP=1181
Arduino Mega 2560 REV3	1	$ 59.99	$ 59.99	http://www.radioshack.com/product/index.jsp?productId=12272877
SainSmart 16-Channel 12V Relay Module	1	$ 28.99	$ 28.99	http://www.sainsmart.com/16-channel-12v-relay-module-for-pic-arm-avr-dsp-arduino-msp430-ttl-logic.html
220-Ohm, 1/4W Axial lead resistor	48	$ 0.0432	$ 2.07	http://www.digikey.com/product-detail/en/CFR-25JB-52-220R/220QBK-ND/1295