# Programmable Power Strip

**Jesse Goodall**
**E90 2007**
**Advisor: Bruce Maxwell**

# Table of Contents

# Table of Figures

## Abstract

This project focuses on the construction of a programmable power strip directed towards consumer use. A programmable power strip looks and functions like a typical consumer power strip, but it adds the ability to control its sockets individually as a function of time. Benefits include the reduction of everyday power consumption, entertainment such as choreographed lighting or equipment, home security, and automation. The project is composed of three major phases: physical construction of a simple model involving relays controlled through a serial or parallel connection with a computer, software implementation and application development, and the implementation of a remote physical connection and web interface. In conjunction with the aforementioned design process the device itself is composed of three conceptual design components: the physical electronic assembly based on solid state relays, a computer acting as a controller, and the interface module which is responsible for delivering commands from the controller to the electronic assembly. Phase one focuses on implementing a parallel connection for the interface module while phase three upgraded that connection to an Ethernet connection. Phase two provides the intermediate step of designing software for the controller. The project was implemented slightly ahead of schedule with a prototype cost of about $350. The cost of manufacture is projected to be between $80 and $120, just above the cost of a high end power strip.

# Introduction

## *Proposal Background*

The ability to control a conventional source of AC power as a function of time using automated means is not a new concept. Many consumer devices include electronics to perform this function for purposes of energy conservation or security. Examples include computers which shutdown after lack of usage and monitors which sleep. Industrial appliances and institutional buildings use this ability extensively to conserve power after work hours, turn on outdoor lighting at dusk, and control HVAC equipment. On a consumer level, the availability of a device to provide this functionality common to institutions is not widely available. The proposed project intends to create a device to control a conventional source of AC power as a function of time using automated means for consumer usage.

## *Project Features*

There are several advantages to implementing the project. Advantages include modularity of design, cost, and application. Modularity of design is the product of the conceptual framework for the design of the device. The device can be divided into three components: a physical power strip, a controller providing a human interface using a computer, and an interface between the two. Physical construction of the power strip can be constructed independently of the interface and software deployment. The interface and software can be developed in stages of increasing complexity ensuring completion of a functional device by the project deadline.

The cost of the project was relatively low due to the proposed budget specification set out in the project proposal. Because the device is intended to be a member of the family of consumer power strips, the manufactured cost of the device should be within or marginally above the price range of upper end power strips. The estimated cost of a manufactured version of the programmable power strip is about $80-$120 based on relay boards already on the market. The actual cost of prototyping and design of the device came out to about $350 with major expenses including the cost of 6 relays and a SitePlayer development kit used in the final phase of the project.

## *Use*

Because of the general form of the device based on simple power strips already in wide circulation, the device will be applicable to a wide range of uses. Applications include controlling lighting, especially when a house is vacant to simulate occupancy and avoid robbery of the house, and controlling lighting for entertainment such as synchronizing it with music or switching a set of multicolored lights. The device could also be used to switch on power for electronic equipment at regular intervals such as radios, computers (for use with remote boot), speaker systems, battery chargers, fans or portable HVAC equipment, dehumidifiers, or kitchen appliances. It could be used to save energy during the night when devices are not being used, to control things remotely, and for security to stop people from turning on equipment outside of business hours.

## *Project Design*

The device is based on two conceptual parts with a modular interface. The first is a physical electrical assembly supplying conventional 120V AC power to all attached devices. The second is a programmable interface (also referred to as a controller in this report) to control the 1st part as a function of time (see Figure 1). The most common power distribution device in homes today is probably the power strip. Adding programmability to a power strip ensures the features desirable in a power strip are inherent in the proposed device. Computer control was selected as the programmable interface due to wide availability, power, and flexibility. Computers are common, can be mobile as laptops, provide the option of web control, and cater to more sophisticated control using visual or script based control (for scheduling and automation). Another option of control includes an onboard PIC chip or other microprocessor with buttons or a display. The interface between the computer and the device can come in numerous forms providing flexibility in design and project implementation. The simplest interface available will be used for the initial part of the project to ensure a working solution within the timeframe of the project. A parallel cable was selected as the initial interface. Because a parallel cable is a direct connection from a computer to the device usually no more than 6 feet long and parallel ports are no longer widely available (they have been replaced by USB), a better interface was needed for the final design of the device.

A solution to the direct connection problem is tackled in the final part of the project: creating a remote connection. This can come in many forms which were researched as part of the project. One proposal is to use an infrared transmitter and receiver like a TV remote. The problem with this is obstruction of the signal, which restricts placement of the device. A wireless transmitter and receiver is an alternate solution solving the obstruction issue. Another idea is to use an onboard chip to store program functionality and a few buttons to run or terminate the program. Data can be uploaded to the chip using the already implemented parallel connection. A more involved solution, the one selected for phase 3 design, uses a web interface for programmability from any internet connected computer. There are at least two methods to implement this. One method uses a computer permanently attached to the device using the usual serial or parallel direct connection. The computer would host a webpage and provide a web connection. The alternate solution is to use a microprocessor acting as a stand-alone web server with Ethernet connection. The final idea is to make the device wireless using a wireless hub connected to the stand-alone web-serving chip.

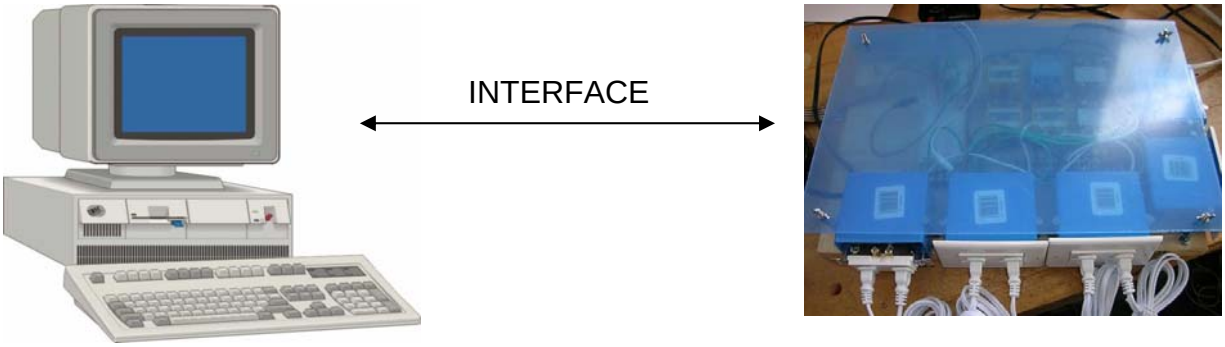


**Figure 1 - Project Concept**

## *Project Management*

Effective project management was one of the most influential and successful components of the project. A three phase design process consisting of physical assembly, software implementation, and a stand-alone implementation was developed to complement the three primary modular conceptual components. Physical assembly and testing was performed first so that the next phase, software development, would be able to be done with the device present for testing.

Completing the physical assembly and accompanying software for direct control over parallel afforded a few critical benefits despite the obvious issue that using a parallel connection is not a satisfactory final interface implementation, as mentioned in project design above. The key benefit of implementing a parallel connection before moving on to more advanced technologies is iterative design. By the time the first two phases were finished, a fully functional prototype was complete. Lights could be synchronized to music, remote control was possible with a directly connected computer acting as a web server, and demos to prospective engineering students, friends, and party venues were possible long before completion of the project. Iterative design also lent itself very well to testing and planning. Many parts within each phase were also developed iteratively, ensuring testing and debugging at each step in the project.

Phase three was implemented as an addition to what was developed in the first two phases. This ensures that, at any time, the user of the prototype can choose to use either parallel or Ethernet connections. For project management, this also ensures that, at a minimum, the device would work through parallel regardless of the stage of development of phase 3 by the conclusion of the project.

As a result of efficient scheduling and hard work the device was built in time to be featured in the campus-wide I20 Paces party on April 13 with great success.

# Phase 1: Physical Assembly

## *Prototype Physical Construction*

Physical prototype design focused on safety features, layout of components, and a design for ease of modification and addition later on in the project. Plywood was selected as the base due to ease of use, and Plexiglas shielding has been used to box off the components. One side of the device was left with free space and no Plexiglas cover for use during phase three, whose space requirements at the time of construction were not well known. Threaded rod was chosen to secure the Plexiglas cover. With a combination of nuts and wing nuts, the cover could be raised to a variable height and easily removed and replaced so that accessing the electrical components would be easy even after completion of the prototype. All components were secured with a combination of screws and nails. The power supply cable entering at the corner of the box was secured to ensure accidental tugs on the cable would not destroy the device. The device is shown below in Figure 2.

**Figure 2 - Device**

## *5V Logic*

5V Logic is responsible for taking control signals from the interface module and appropriately controlling the 120V AC circuitry through solid state relays. There are 6 separate circuits of components connected in series (see Figure 4), each one a connection

from a shared hex inverter, through an LED providing status, to the control side of a relay. Power to the circuitry comes from a 5V DC power supply, as required by the hex inverter. A breadboard and 22 gauge wiring was chosen for the initial prototype design rather than a custom designed printed circuit board for ease of use and experimentation. The breadboard also allowed phase 3 planning to occur at the beginning of phase 3 rather than during physical construction. A picture of the resulting electronic circuitry is below in Figure 3.

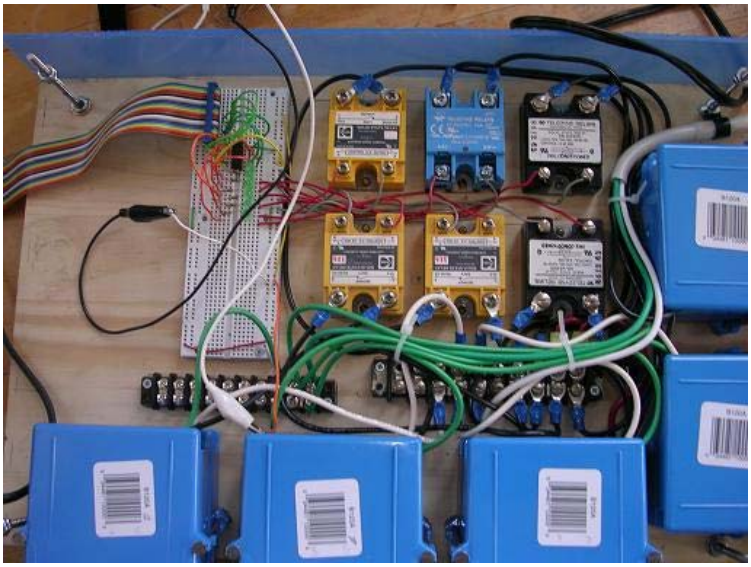

**Figure 3 - Device Electronics**

## *120V AC Output Circuit*

The 120V circuitry is controlled by the relays and provides power to connected devices. Power enters through the hot wiring in the power supply cable, passes through the power switch provided for safety and convenience, is switched by the relays, then passed on to the connected devices, and returns through the neutral cabling forming a complete circuit (see Figure 4). Ground cabling is connected to all power sockets and the power switch and also connects to the logic to provide a common reference point. The 120V power is entirely isolated from the 5V logic by optoisolators in the relays. For the sockets, ends of power cables and standard home power sockets were considered. The power sockets were chosen because, while larger, they look and feel more professional and provided the learning experience of wiring a house. The 120V wiring is 16 gauge wiring as required by code.

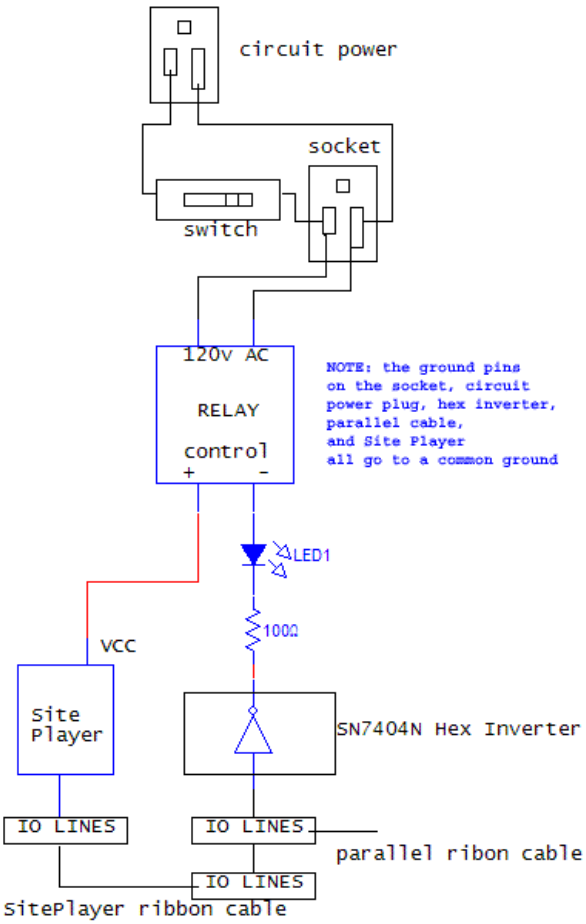


**Figure 4 - One of 6 Identical Circuits in the Device**

## *Safety*

Safety is a large consideration of the project due to live 120V circuitry. Both human and equipment safety were considered. Equipment safety involved protection between the device and the controller on the 5V logic side, and the device and building circuit on the 120V side of the device. For 5V logic, op amps and optoisolators were considered. Op amps were selected in the form of a hex inverter. All logic wiring is routed through the inverter before contacting the interface to avoid damage to a computer's parallel port or the onboard microprocessor added in phase 3. A 20 amp fuse acts to shield the building circuit and the device from each other in the case of any faulty wiring. This could prevent the tripping of a building's circuit breaker. If the device were to be manufactured the fuse would be replaced with a surge grade breaker.

For human safety when handling the prototype, Plexiglas shielding prevents spills or hands from touching the circuitry. A power switch only controlling the 120V side of the device allows for easy shutoff in emergencies and testing with the use of the LED status indicators without actually powering any attached devices. Placing the 120V hot wiring in the center of the device and using standard code-compliant power sockets also add safety. Another unimplemented addition would be a ground fault interrupter for human safety when working with the prototype.

# Phase 2: Software Implementation

## *Windows Localhost*

This implementation is by far the simplest. Because Windows (as of XP) no longer supports direct parallel port driver control, the use of a control library is necessary. In this case, inpout32.dll is the standard. Before any programming was considered a standard parallel port testing program was used to ensure communication between the PC and the device was functioning, as shown in Figure 5. A simple C program uses this library to allow a user to dictate which data pins in the parallel port register are on via an 8-bit number (ie 0 all off, 255 all on). A second version was written allowing the user to toggle the sockets with keys 1-6 on the keyboard in real time.

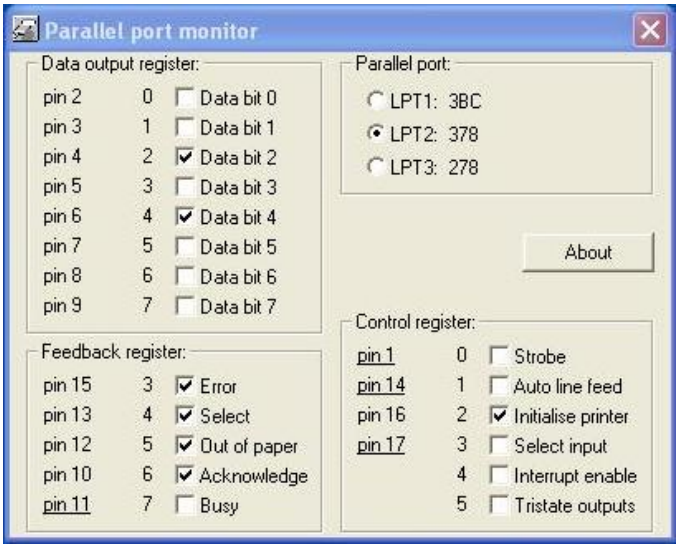


**Figure 5 - Localhost Parallel Port Testing**

## *Windows Web Interface*

Once localhost control has been established, remote control can be provided through a web interface (Figure 6). To do this, the combination of a web server on the attached computer, PHP or cgi-bin scripts, and possibly Ajax (for real-time update through the use of JavaScript) are required. The user visits the PHP page and selects which sockets he wants to activate. PHP in turn calls the command line arguments developed for localhost control.

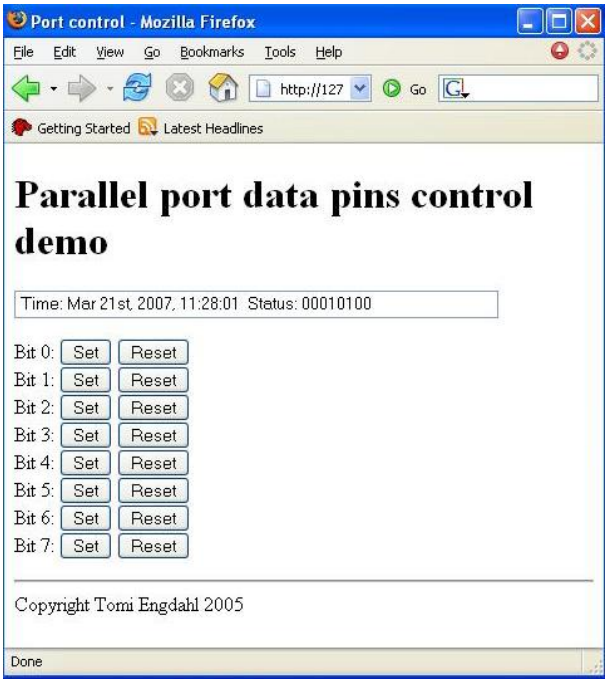


**Figure 6 - PHP Page for Controlling the Device Remotely**

## *Linux Localhost*

Local parallel port control in Linux differs from Windows because direct control can be achieved (given root privileges for the code). There are two different methods, one using seek and fopen and the other using ioperm. However, both lead to the same result: the ability to provide the system with an 8-bit number (in hex) as a command line argument for controlling the data lines in the parallel port register.

## *Linux Web Interface*

This is nearly identical as that for Windows except for the inherent ability to avoid the cumbersome Windows visual interface on the server. Unfortunately the setup is generally more tedious due to permissions. Regardless, the result is the same: Apache, PHP, Ajax.

## *Lights Synchronized to Music*

With some Winamp magic and its wonderful plugin interface, power sockets can be made to switch in time to music. This utilizes the parallel port control library for the local host, spectral analysis (ie fast Fourier transform for retrieving a frequency spectrum from music) provided by Winamp, and a custom plugin (see Figure 8). The plugin

creates upper and lower bounds around frequency ranges and a threshold within each set of bounds. When the amplitude of the sound signal at a frequency within one of these ranges exceeds the threshold for that range, the corresponding power sockets are triggered. The cumulative result from the threshold testing (ie the total of which thresholds have been exceeded) is considered to be a state which indicates which sockets on the device should be activated during the current program cycle. The process occurs once per cycle, but cycles occur many times a second. In the current implementation there are low, mid, and high ranges which can be modified by the user via the keyboard.

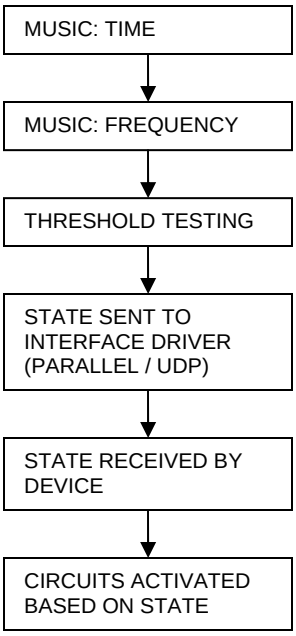


**Figure 7 - Winamp Flowchart**

As indicated in Figure 7, the entire cycle process can be broken down into 6 conceptual steps. First, Winamp retrieves the music as a function of time and then, through Fourier analysis, renders a frequency spectrum at that point in time. Threshold testing is then employed by the plugin and the resulting state information is sent to the appropriate interface driver. The interface driver sends the state to the device which activates the correct circuitry.

For incandescent lighting, the quick cycling of lights on and off creates the effect of frequency modulation. This is a similar effect harnessed by dimmers to adjust the intensity of lights. The net result is that by adjusting the thresholds, the lighting intensity and duration can be changed easily allowing a dark or bright mood for a party. While incandescent lights tend to blur the fast pulses of power generated from this application LED strip lights do not. Nonetheless, this is a negligible problem for the viewer as the human eye tends to blur the result itself.

Additional features can be easily added in the source code. More frequency ranges could be added until all the relays are used instead of just 3. Light pulses, especially for low frequencies, could be longer and less frequent by adding a few simple counters in the code so that when the low frequency is activated it remains active for

several cycles before it is tested again. Finally, some sort of weighted average could be employed such as a median filter for more intuitive light patterns (reminiscent of Karl Petre's E90).

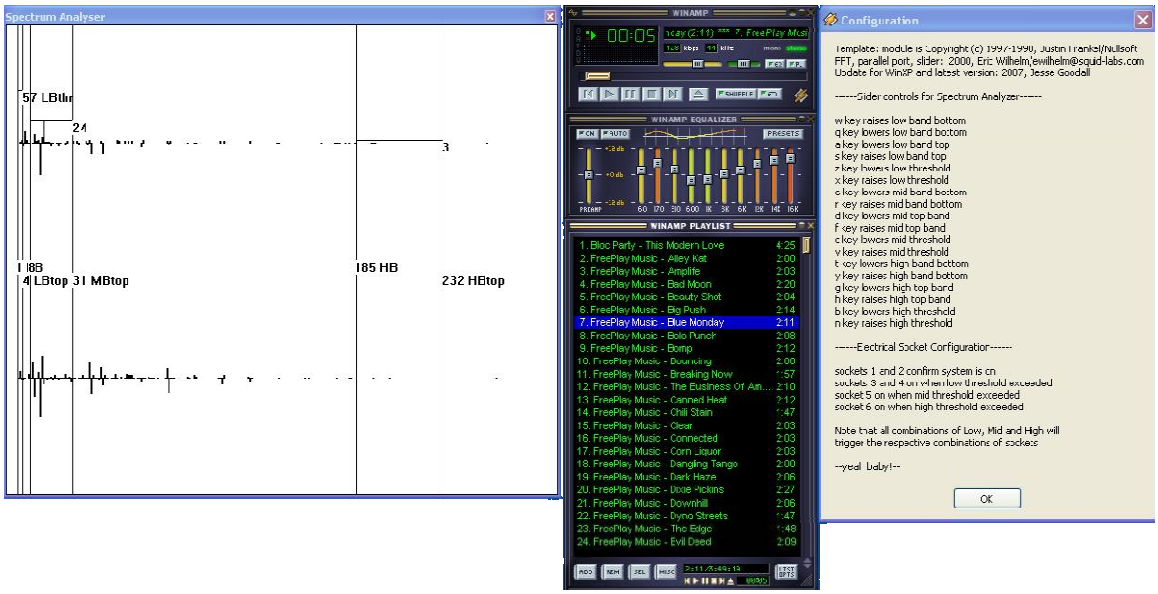


**Figure 8 - Winamp Application with Plugin**

# Phase 3: Stand-alone Interface

## *Computer Network Based*

Selection of a stand-alone implementation for the interface module presented a challenge due to the dearth of options available. As discussed in the introduction, options included the use of an infrared transmitter and receiver like a TV remote, a wireless transmitter and receiver, an onboard chip to store program functionality and a few buttons to run or terminate the program, and a web interface for programmability from any internet connected computer. The web interface is the chosen third phase implementation. This requires a microprocessor acting as a stand-alone web server with Ethernet connection. Picoweb and Siteplayer were considered as two proprietary options for the microprocessor-based web server, with Siteplayer (Figure 9) as the final pick. Creating a custom built solution was not considered due to infeasibility given the constraints of the project.

Using a computer network based solution offers a variety of advantages. Computer networking protocols are in widespread use and, with connection to the internet, can allow communication between a computer acting as a controller to the device from anywhere in the world. Harnessing computer networking also allows the use of already built networking infrastructure including security protocols, access to mobile devices including PDAs and browser capable cell phones, and wireless networking, eliminating the need for a custom built wireless solution. If no computer network is available, communication between the device and a laptop computer is still possible through the use of a crossover cable (a direct connection between the computer and the device up to 100m long before attenuation occurs) or wireless. Wireless can be implemented by plugging the device directly into a Linksys wireless router and using a laptop with a wireless card hence allowing wireless control with nothing more than a wireless card and the router, which are both portable.

**Figure 9 – SitePlayer (Size of a Quarter)**

## *Technical Difficulties Using SitePlayer*

Using a proprietary solution like Siteplayer created a variety of technical challenges. First, Siteplayer was not well documented, nor supported. Many users complain of the lack of support on forums and firmware upgrades. Secondly, the use of io pins (input-output pins) on the board was not easily discovered. Initially it was perceived that SitePlayer had to communicate through a serial port commonly used to communicate with device microprocessors. Successfully, but unnecessarily (and not included in the final design), a serial to 8 bit discrete output circuit was researched as a solution. Finally, the closed source proprietary firmware limits innovation. Combined with the lack of support, the result is that DHCP and password protection do not function on the device.

## *Connecting to the Device*

Because the device requires an IP address for network communication, assigning one and knowing the current address is necessary to coordinate communication between the controlling computer and the device. Originally during development the IP address had been hard coded into code developed on the controller. On the device the only way to change the IP was through a serial connection. There were two methods developed to make configuring an IP for the device possible so that it could be used portably.

The first method is dynamic assignment of the IP address via a DHCP server. The method for this involves discovering the subnet on which the device will be operating, plugging in the device, scanning the subnet and using ARP to find the device based on its MAC Address (see Figure 10). The only flaw in the method is that DHCP is not functional as of the latest firmware update for SitePlayer.

The second method, which is the current one, used static assignment of an IP address. The first step is to discover the subnet on which the device will operate. This can be done by asking the network administrator or connecting a laptop to the Ethernet jack the device will be connected through and ANDing its IP address and subnet mask to obtain the subnet. Next the device should be given either a static IP assigned by the

network administrator or one with the same subnet just discovered. The third step is to connect to the device directly via a crossover cable (see Figure 11). The laptop will need to be assigned an IP address manually with the same subnet as that of the SitePlayer's initial IP (this would be indicated on the device). From the laptop, the webpage can be accessed through the device's initial IP and on the page a new IP address can be assigned (the one intended for use). Without shutting off the device, it should be connected to the wall jack with a standard Cat5 Ethernet cable. Finally, the device should now be accessible on the web (or local computer network).

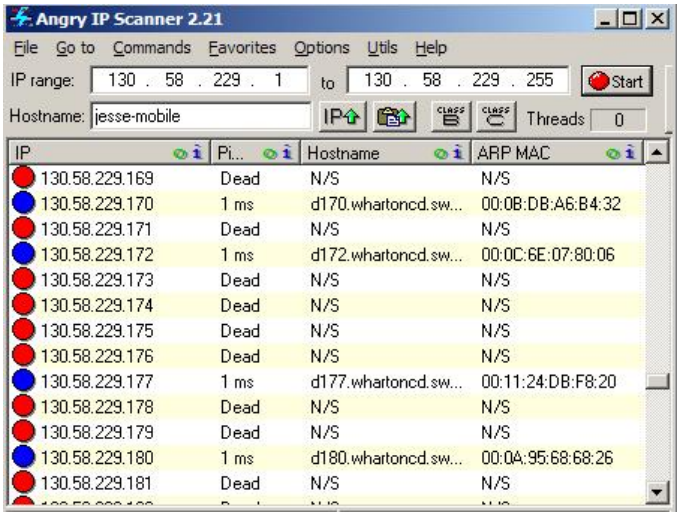


**Figure 10 - IP Scanner with ARP MAC Lookup**

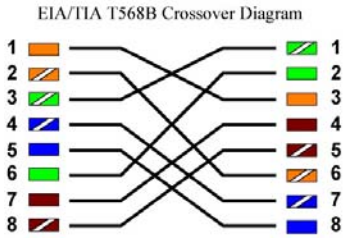


**Figure 11 - Crossover Cable Wiring**

## *Interactive Web Interface*

The device can be used and configured over the web by typing in its IP address into a web browser. The primary function of the custom built webpage is real time interactive control of the device (see Figure 12). There is a checkbox for each of the 6 sockets on the device. Refreshing the page will provide the latest status with checked boxes indicating sockets which are on. The moment a checkbox is clicked the state of the corresponding socket will toggle. This form of control uses SitePlayer's proprietary cgi-bin-like coding. Using JavaScript allowed additional features including an all on and all off button.

There are two methods by which data is transmitted between the client computer and the SitePlayer server (Figure 13). New state data is transmitted to the server from the client through the use of a hyperlink or html form. A checkbox is an example of a form

element. On submission of the form, the state is transmitted to the server which activates the corresponding circuitry. Form submission requires that all form data is sent simultaneously through the use of a submit button. On the interactive page there is no submit button because a user should not have to check a box and then click a form button. It is much more responsive to send the state data directly when the box is checked. The technique used to accomplish this is encapsulation of the checkbox form element in a hyperlink so that the hyperlink sends the data – a trick developed specifically for this project. State data is returned to the client immediately once the hyperlink has been activated in the form of a page refresh, thus ensuring the client is always updated whenever it has any indication of a state change.


**Figure 12 - Interactive Interface**

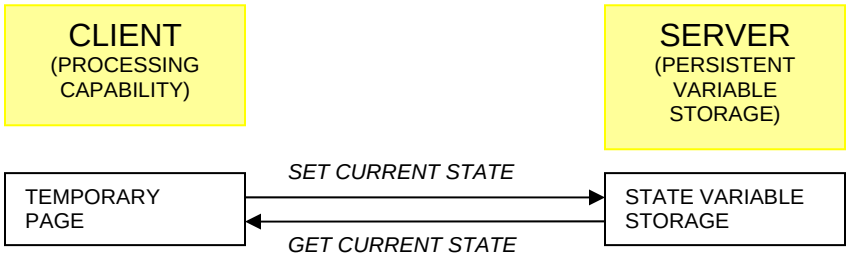


**Figure 13 - Client-Server Setup**

### *Remote Music over UDP*

The most popular application of this device has been the synchronization of lights with music across a computer network. The result is the ability to play music on a laptop connected wirelessly to the campus network and the programmable power strip located anywhere else on campus will sync lights to that music. This feature of the device was used with great success for the campus-wide I20 Paces party on April 13th. The device was connected to an Ethernet jack in the party venue. All lights were routed to the device and a laptop connected to the sound system wirelessly relayed state information for the lighting to the device. A demonstration of the device was given to incoming freshmen on Ride the Tide weekend for the Engineering view book (Figure 14).

The feature uses the same Winamp plugin developed in the software implementation phase except that instead of control bytes being sent to parallel device drivers, they are now wrapped in UDP packets and sent to the device's IP address as specified by the user.

There are at least two advantages of this application over other forms of remote synchronization of lights and music. First, creating a plugin for Winamp as opposed to an independent application provides access to Winamp's excellent music processing features whose production would not be possible during the scope of this project. The second advantage relates to the use of UDP. There are two commonly used protocols for encapsulating network data: TCP and UDP. While UDP is less reliable, it is less bandwidth intensive. For this application where data is sent many times a second, bandwidth is more important than ensuring every single packet reaches the destination (if one packet drops no one will notice). In this case, the plugin is sending one byte of information in a UDP packet which is about the smallest possible transmission across a network.

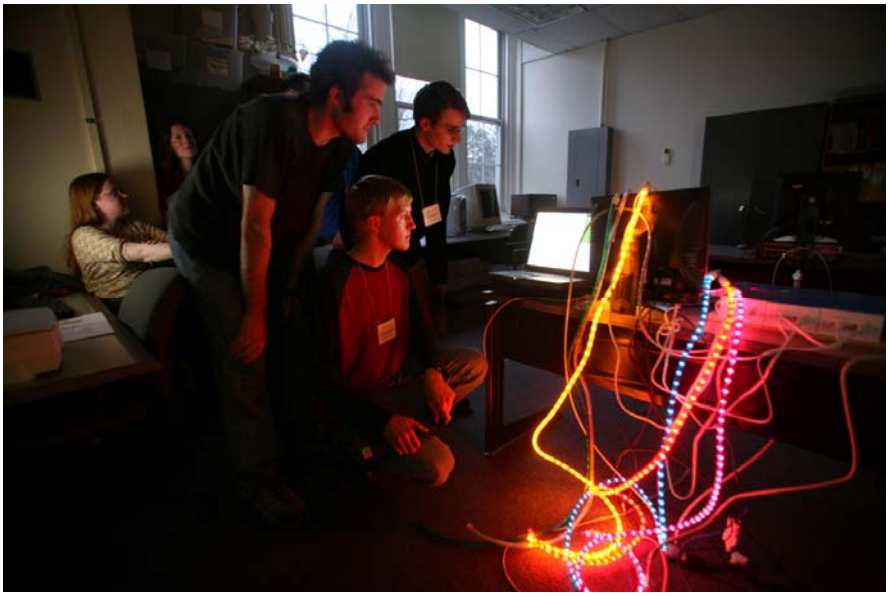

**Figure 14 - Dancing Lights Demo for Ride the Tide**

## *Scheduling*

Scheduling attached devices to go on and off at set intervals (a sequence) is one of the primary intended uses of the programmable power strip. Three methods were considered to achieve scheduling: scripting by the user which would then be interpreted by a C program or PIC chip, a C or C++-based program which would communicate with the device over UDP, and a JavaScript application. The later was chosen because the code is entirely contained on the device without the need for server-side processing thus avoiding dependence on a PIC chip while allowing easy access to the application over the web from multiple clients.

The microprocessor used for the SitePlayer web server provides 48KB flash and virtually no onboard processing for applications. Because the controlling computer is very powerful relative to the device, processing should ideally be done on the computer (the client side). JavaScript is the most commonly used web technology for client side processing. For the application developed during the project, JavaScript was used to time events. It sent timestamp and state information to the SitePlayer server. When the local page was refreshed to reflect the current state of the device, JavaScript would retrieve the current state and time information from the SitePlayer to continue the scheduling application. In this way, the client and server would continuously update each other allowing reliable scheduling.

The client-server scheduling application uses JavaScript's processing and SitePlayer's persistent variable storage to allow multiple clients to run different parts of the same schedule independently. A schedule consists of a series of events, with each event having a state and time as indicated in Figure 15. When the application is started, timestamp and state information (provided by the user) is sent to the SitePlayer server. The server keeps the time and state data until the application concludes or has been terminated. A client can then request this data as part of a page refresh and compare event times to the current time retrieved from the client computer clock. When the computer clock reaches an event time, the client commands SitePlayer to assume the new state. Because it is not important that the same client retrieves the time and state data from the server to initiate an event, multiple clients can be used at different times during the same sequence to initiate events, assuming their clocks are synchronized. Also, because the only data a client needs to transmit to the server is the initial time and state data and then, later, only single commands to assume the new state when an event is reached, a client only needs to have the application running at the beginning and then only a few seconds before and after an event occurs. The net result is that a user could commence a sequence with a client, shut the client down, and then hours later, just before the next event, open the application page again from another client. Thus, the application can be very convenient for users without a dedicated client.

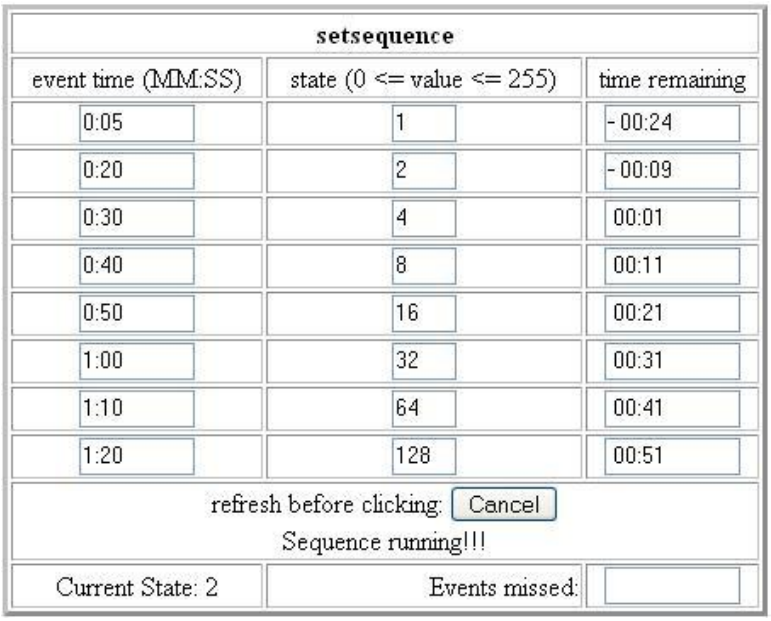


**Figure 15 - Sequencer Application**

Figure 16 shows how the sequencer application works in detail. The diagram is laid out to emphasize the interaction between the client and server. As mentioned, only the client can initiate events and process data. The server's purpose is to receive imitation commands and provide persistent event data storage so that multiple clients may be used and the application can run even when not open on a client. The first part of the diagram simply indicates that the client can request two different web pages, the usual simple interactive page, or the much slower page that includes the sequencing application (slower due to the large amount of code required for the application). The sequencer application starts with the 'wait for set' state. Once the set button has been pushed, the program will initiate by starting a timer and counting down for all events. The set button is replaced with the cancel button which, if pressed, will stop the timer and delete all event data on the server. When the first state is reached the client sends time and state information for all intended events to be stored on the server for the duration of the sequence. It also sends the server a command to change the state to match that of the first event. Sending the data requires a refresh of the webpage at which point all data on the client is erased and replaced with the event data which was just sent to the server. The client then looks at the next intended event to see if it has already passed. Missed events can occur if the application is closed during the time of event initiation, if there is a network error, or if the page is refreshing (networks are complicated so expecting failed communication between client and server is important). If an event is missed the client informs the user. Either way, timing continues until the next event at which point the new state is initiated. At the end of the sequence all server-side data is reset and the application terminates, ready to start again should set be pressed. If the application is cyclic (a user option not implemented due to project time constraints), the application would continue indefinitely until cancelled.

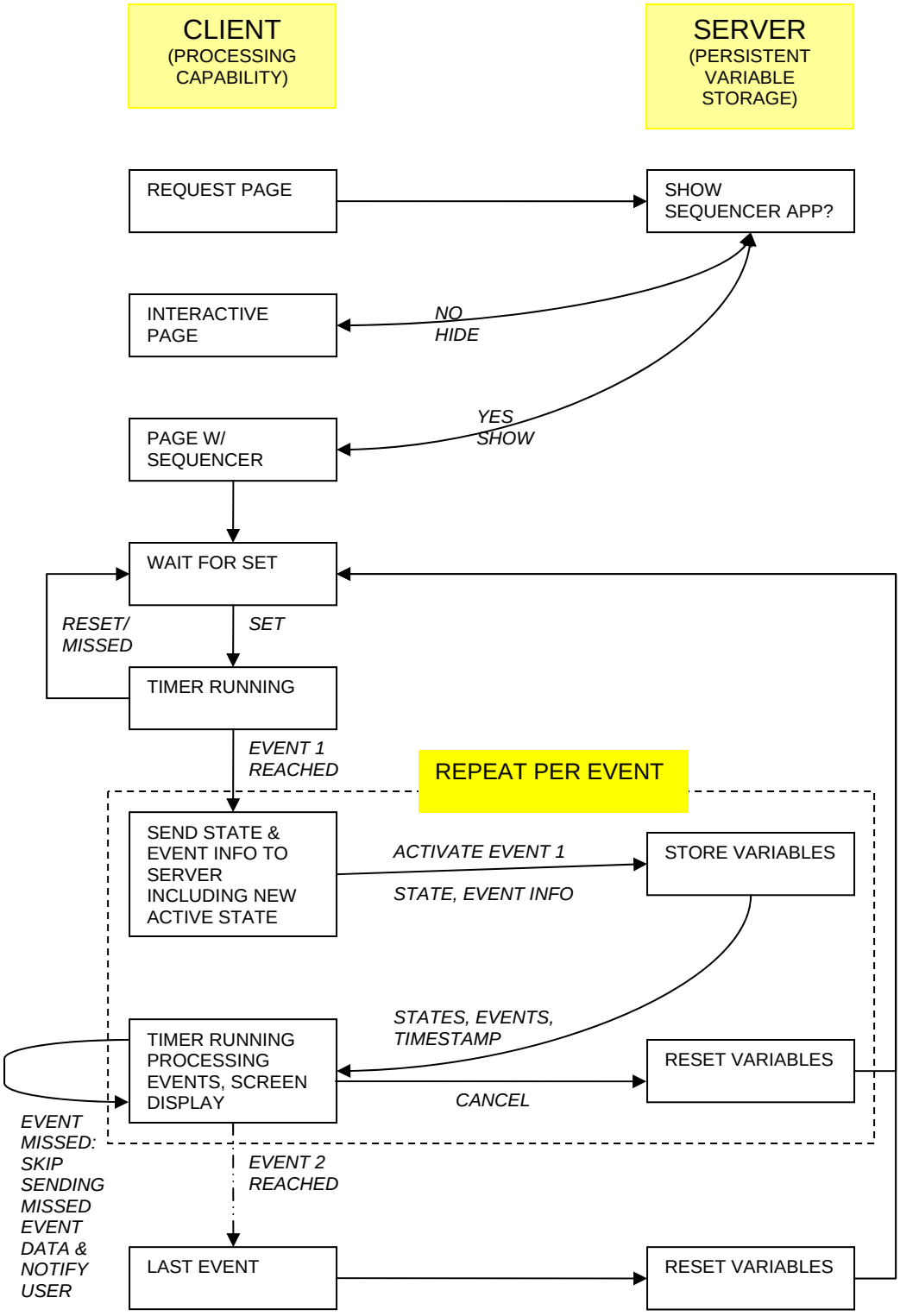


**Figure 16 - Sequencer Flowchart**

### *PowerPoint*

Like synchronized lighting using Winamp, there are a host of specialized applications for a programmable power strip. One of the most interesting, especially for use with E90 presentations based on slideshows, is the control of lighting and appliances from within PowerPoint. With this technology it's possible to make lighting shows and control fans on slide transitions or through interactive elements such as buttons. PowerPoint's VBA (Visual Basic) provides the bridge between the slideshow and device driver software for the programmable power strip (Figure 18). With VBA macros and add-ins user actions can be detected and state commands sent to a UDP client or parallel port driver through a shell call (Figure 17). Both methods are identical from VBA's point of view. If using UDP, however, both state information and an IP address must be specified. VBA provides the added flexibility of timing. Using a timer based on the CPU ticks allows a sequence of state changes to occur on a slide transition.

A demonstration of this application was given during E90 presentations. One demo had a plane flying through a thunderstorm. Audio was compiled from sound effects and a plane was flown across the screen with PowerPoint's animation feature set. Lights around the room were timed to flash as thunder struck while a fan began whirring when the plan passed by the screen. Other uses include museum exhibits or stage sets. For an exhibit, lights illuminating various pieces of artwork could be activated as a slideshow discussed the history of the pieces one at a time. For the theatre, the backdrop could be a colorful PowerPoint presentation and lighting, fans, and smoke machines could be activated during the performance in sync with the backdrop slideshow.

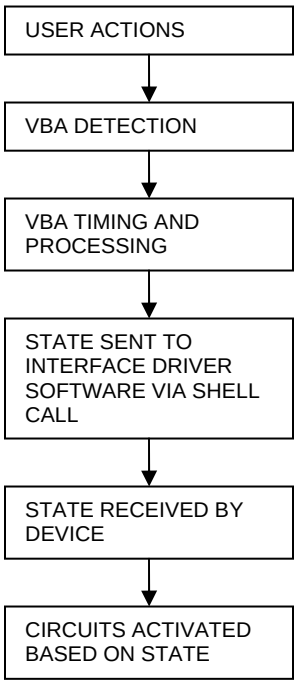


**Figure 17 - PowerPoint Flowchart**

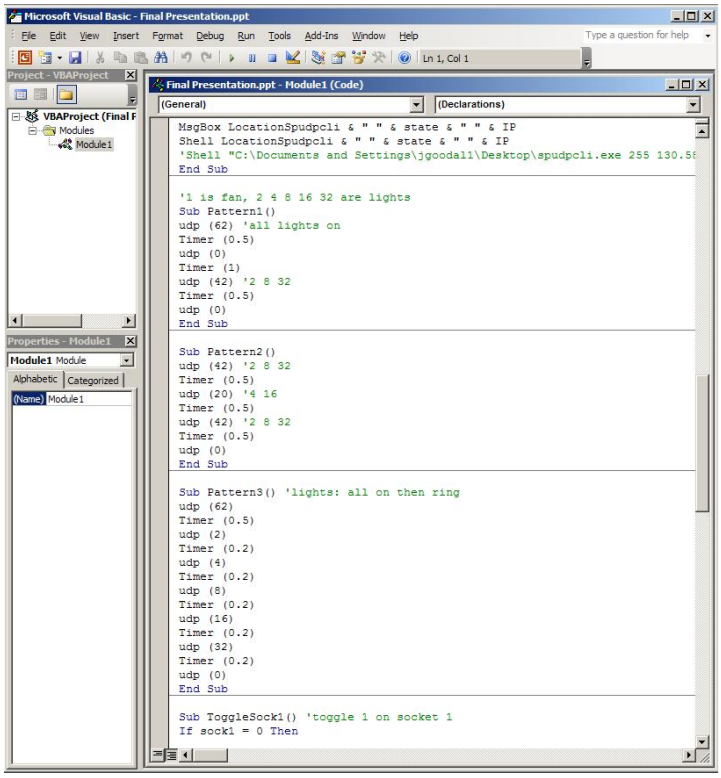


**Figure 18 - VBA Coding**

# Further Development

## *Scalability*

One of the advantages of a power strip-like device is that, if manufactured, each could act independently or in coordination with one another providing an easy method to scale control of appliances. Applications of scalability include control of large theatrical performances, lighting shows, or coordinated control of computer power cycles for offices. One could, for example, control the lighting for multiple parties around campus simultaneously. There are two methods considered for scalability, a master/slave configuration, and direct control of devices from the controlling computer. In the master/slave configuration, a client would provide commands to the master. The master would, in turn, forward those commands to slaves over UDP. The client could communicate with the master through UDP or on a webpage. Direct control of multiple devices is simpler. The same procedure used for remote music could be employed, but UDP state packets would be sent to multiple devices instead of only one.

## *Security*

While security has not been a part of the development for the project, it is a major consideration. Reliable control over the source of power for appliances is critical. There are always manual overrides on circuits in buildings. The thought of someone else gaining control over the appliances in one's home is not friendly and may be a source of reservation over the adoption of a programmable power strip. Unfortunately, password protection based on current internet standards was not possible during development due to the lack of support from SitePlayer. The development of a password page is still possible, but passwords would be passed over the Ethernet using cleartext unless a PIC chip attached to the SitePlayer could encrypt the communication. Hence, anyone using a packet sniffer like Ethereal would be able to crack the password page. The best solution to the issue is to put the device on its own VLAN (virtual local area network) with the controller. Operation outside of the local network could be securely achieved using VPN (virtual private network). Neither of these technologies are common on the home network, but are widely used in institutional and corporate settings. For instance,

Swarthmore College uses VLANs to secure the blackboard system for the administration of the meal plan.

### *PIC Chip Backend*

Using a PIC chip in coordination with SitePlayer presents new opportunities for server side processing. This could replace the use of JavaScript for scheduling allowing the device to use preprogrammed routines and do all scheduling independent of a controller. It could allow onboard specialized applications such as DSP for music processing. The drawback is the increased expense and so a version of the programmable power strip featuring a PIC chip would have to be considered the "pro" version.

### *Computer Vision, Infrared, and Other Indirect Control*

For security and automation, computer vision, infrared, and other sensors could be used in coordination with the device. For instance a webcam attached to another SitePlayer could detect intruders and relay commands to the device to activate lights etc. Another interesting application involves the use of Paul Azunre's E90 project for making things go on off when a finger or hand enters a region of the sensor pad. This could be good for presentations – wave your arms and things activate.

### *Further Software Development*

Arcade games present an opportunity for further software development. Instead of expensive specialized setups, just use a computer game with a connected programmable power strip. When things are clicked or you enter a region in a game devices including fans, motors, and lights are activated for an interactive experience. Another area for further development includes adding a more user friendly interface to the Winamp plugin including output to more than 3 channels. A computer application dedicated to lighting and device scheduling for theatre is another exciting possibility. Finally, a central server based application for control of the corporate environment especially including powering on and off computers and equipment to save energy would be practical.

### *USB Connection*

USB is the replacement for the parallel and serial connection. The advantage of its use is that it is found on all computers today. The disadvantage and reason it has not been implemented in this project is that it is a direct connection, forcing the controller to be within limited distance of the device, and because it transmits and receives serially, a chip would be needed on the device to buffer the data flow.

## Acknowledgements

- Bruce Maxwell - Advisor
- Ed Jaoudi - Electrical construction, parts, computing support
- Grant 'Smitty' Smith - Physical construction
- Cheever - Proposal
- All those sources and resources on the web and otherwise which I have built this project upon.

# Bibliography

Engdahl, T. (1996-2006). "Parallel port interfacing made easy: Simple circuits and programs to show how to use PC parallel port output capabilities." from http://www.epanorama.net/circuits/parallel_output.html.

Logix4u.net. "Parallel port Interfacing Tutorial." from http://www.logix4u.net/parallelport1.htm.

Morgan (2007). "Timecli."

NetMedia "SitePlayer."

Sreejith, N. (2006). "Stepper motor driver for your Linux Computer." Linux Gazette(122).

Wilhelm, E. J. Halloween Haunted House Controller. Make Magazie. **3:** 89-95.