Ember
Group May 13-14

Stephen Brossart
Mike Carter
Eric Greenley
John Ryan
Matt Boyce

April 29, 2013

Advisor: Dr. Phillip Jones
Client: Troy Benjegerdes

Abstract

This document gives a detailed specification of a Power over Ethernet LED lighting controller. It includes design considerations, requirements, system components, testing results, and a user manual.

Contents

1 Overview

1.1 Problem Statement

There is a growing market for alternatives to incandescent lighting, both residentially and commercially. LED lighting is emerging as an alternative that is energy efficient and cost-effective. Companies have developed LED lighting solutions using a variety of strategies including Philips, Insteon, NuLEDs, LumenCache, and others. These solutions have disadvantages, some of which include proprietary bulbs, separate, non-standard cables for control and power, or proprietary control systems.

1.2 Purpose

The purpose of Ember is to create an LED lighting controller that uses industry standard cabling, non-proprietary LED bulbs, and integrates with any available home automation solution.

1.3 Sample Installation

Figure 1 demonstrates the intended use-case of Ember. The Ember lighting controller accepts network commands and power over a single CAT6 cable, and outputs 12 volts of DC power to an LED bulb.

2 System Design

2.1 Requirements

The light controller must power one or more LED lights with the following requirements.

2.1.1 Functional Requirements

Conform to the IEEE PoE+ Standard The lighting controller must support being powered by a typical powered network switch. The IEEE 802.3at standard must be supported; 42.5V - 57V DC input must be tolerated.

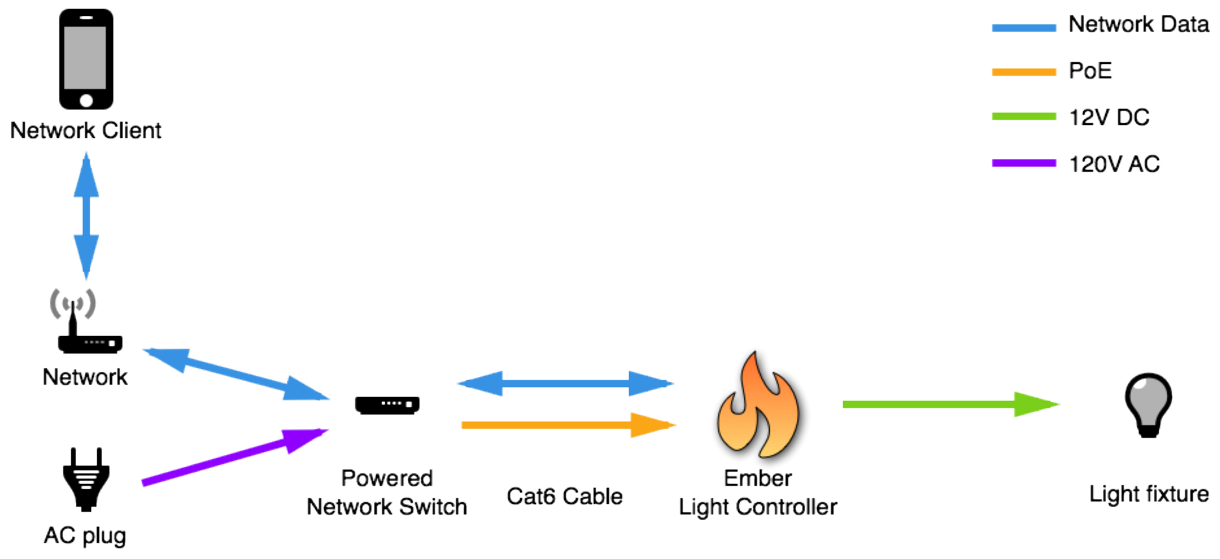


Figure 1: Installation diagram. More examples are shown in Appendix A.

Provide power to LEDs Output from the lighting controller must power standard LEDs. Typical LED bulbs are rated for 12V DC. Input and output cables must not require an electrician to install, therefore they must be supplied with no more than 60 volts at 100VA, to meet 2012 NEC class two wiring standards.

Network Control Communication over TCP must be supported. Commands to enable, disable, dim, set the color, and check the light's status must be supported.

Use common cable and ports Uncommon cables and ports are not to be used. Common cables are defined as inexpensive (under 300 USD for 1000 feet) and widely available (available in online stores).

Discovered over the network The lighting controller will be assigned an IP by a router or DHCP server. Broadcast packets must be sent from the lighting controller to allow other network devices to discover it.

2.1.2 Non Functional Requirements

High speed The time between sending the command and the action should be tolerable (under 100ms).

Easy to install Installation will not require extensive knowledge of the product.

Works with a variety of 3rd party LED bulbs Support for non-proprietary LED bulbs requires a 12V output from the lighting controller.

A network protocol A network protocol must be used to allow interfacing with the lighting controller. A network protocol allows for integration with 3rd party home automation systems to interface with the lighting controller.

2.2 System Analysis

2.2.1 System Topology

Multiple possible topologies were considered for the Lighting Controller, primarily mesh networks and tree networks.

Partially connected mesh networks are desirable because the ability to add additional Embers by connecting them anywhere in the network improves the installation process. However, it becomes difficult to ensure that the cables connecting the nodes will be able to support the power levels necessary to supply power to all nodes. This problem becomes similar to a power grid distribution problem, requiring resources beyond the scope of this project.

Tree networks were chosen because of their extensive use in existing Ethernet networks. CAT5e/CAT6 cable is currently installed in many homes and offices using this topology. A powered network switch is added just before the Light Controller in the tree, adding power for the Light Controller using the data provided by the Ethernet network, while simultaneously minimizing the distance needed for the network to carry power.

2.2.2 Cabling / Interface

A related decision to the topology was the cabling and interface to use. The initial proposal for the project used InfiniBand to connect the nodes. InfiniBand is a protocol commonly used in cluster computing as a high speed interconnect for communication between cluster nodes. It is connected in a mesh, which is desirable for the wide variety of ways the network can be connected. This would work very well for sending control messages. Unfortunately, power distribution through an arbitrary mesh is a very difficult problem to solve, as described in the previous section. Its cabling and controllers are much less common than Ethernet and much more expensive.

Another choice used by several competing LED lighting solutions is a simple two wire cable for power. The downside of this is that it almost always requires new cabling to be run, which is at best inconvenient in existing buildings, and it requires power and data to be separated into separate cables (or data to be sent wirelessly).

Power over Ethernet+ was chosen because it uses the same cable (CAT5e or CAT6) as extremely common and familiar Ethernet networks, and the wide availability of powered network switches. It also allows power and control data to be sent using a single cable.

2.2.3 Light Dimming

The two methods for light dimming considered were constant current dimming and PWM dimming. Constant current dimming avoids the potential problem of a noticeable flicker that PWM can exhibit at low frequencies. However, constant current dimming causes significant nonlinear brightness and color response, whereas PWM remains linear in both aspects. PWM dimming was ultimately selected because it is supported directly by the microprocessor, and because of its linearity in brightness and color response. To remove the possibility of noticeable flickering, a PWM frequency of 500Hz is used, which exceeds the rate of perception of the human eye. (Aimtec 2011)

2.2.4 LEDs

LED bulbs powered by 12V DC are supported. This allows for maximum flexibility in bulb types and styles relative to many other LED lighting solutions, many of which require the use of proprietary LED bulbs.

2.2.5 Microcontroller

Several options were considered for managing network communication and controlling the LEDs. Among these were an FPGA, a Raspberry Pi using an ARM processor, and an Arduino using an ATmega328p. Implementing a networking stack in hardware is possible, but ultimately an FPGA was not chosen due to complexity and time requirements.

The Arduino was chosen over the Raspberry Pi because of its open source hardware.

2.2.6 Network Protocol

There were two choices to make in selecting a network protocol: the control protocol, and the discovery protocol. TCP was chosen for the control protocol because of its wide usage and support. It was chosen over UDP because of its error detection and guaranteed delivery. Guaranteed delivery and error detection make TCP slower than UDP, but the efficiency of the control protocol makes the speed difference negligible. Bonjour/mDNS was chosen as a discovery protocol using a GPL licensed Bonjour library for Arduino.

2.3 Components

The following three components are used in the light controller's design.

A power over Ethernet module is used to communicate with the powered sourcing equipment, (PSE) or powered network switch. It announces that it is able to accept power. It is able to separate power and data.

The microcontroller's responsibility is to manage network discovery, receive and parse network commands, and output the appropriate PWM signals to the amplifier.

The amplifiers accept PWM signals from the microcontroller and amplify those to produce a large signal (12V swing) replica of the small (5V swing) PWM signal.

3 Detailed Design

3.1 Input / Output Specification

3.1.1 System Level I/O

Input: 1 RJ45 jack from a powered network switch containing both power and control data, per the IEEE 802.3at (PoE+) specification.

Output: 1 screw terminal sourcing 12V DC to a LED light bulb. See figure 2 for a digital render.

3.1.2 Component Level I/O

PoE Module

Input: 1 RJ45 jack from a powered network switch containing both power and control data, per the IEEE 802.3at (PoE+) specification.

Output: 12V DC at up to 25.5W.

Microcontroller

Input: TCP/IP network data containing commands for the controller.

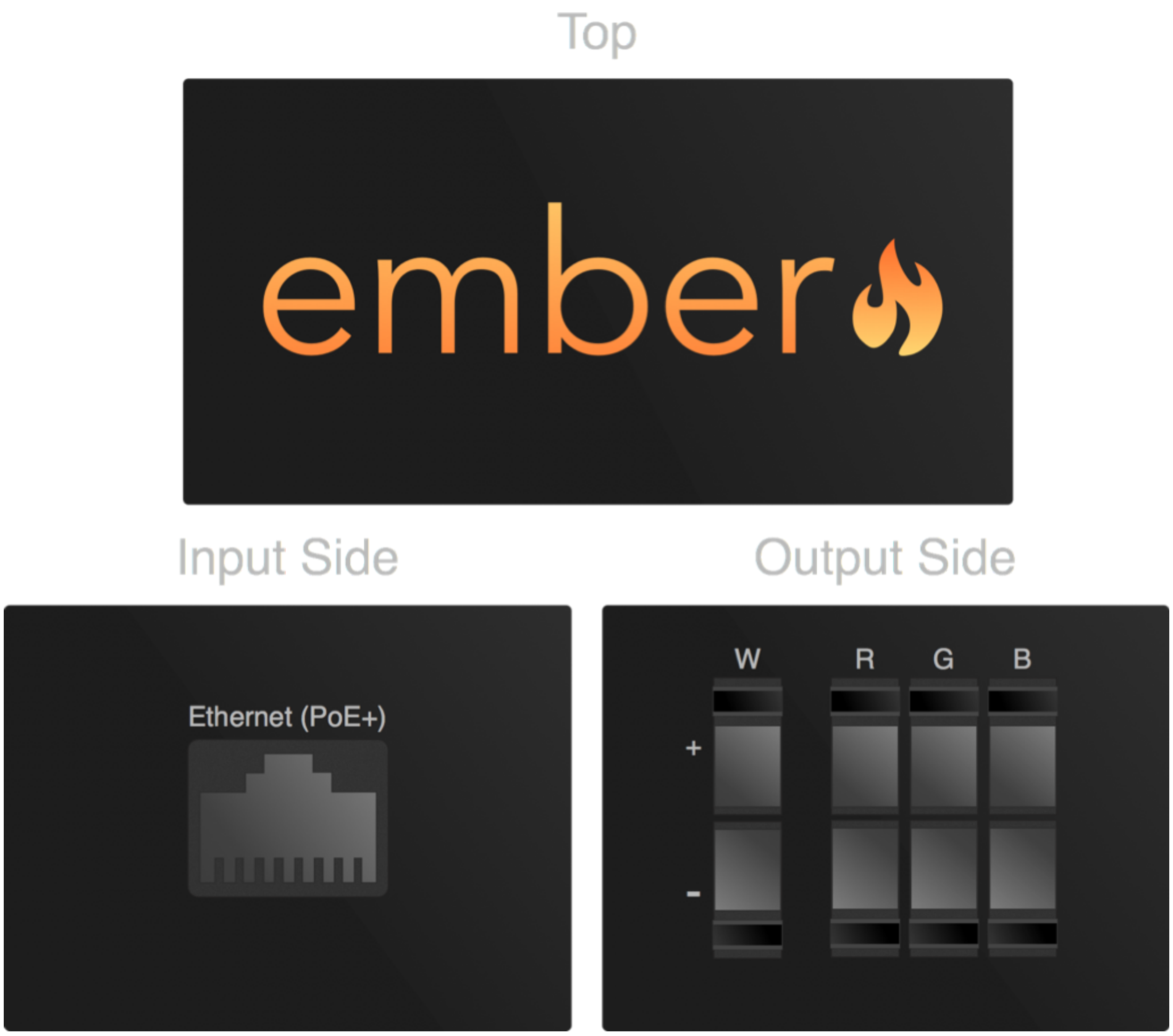


Figure 2: Digital render of Ember's system level I/O

Output: A PWM signal corresponding to the light level. mDNS discovery packets to allow detection on the network by other devices.

Amplifier

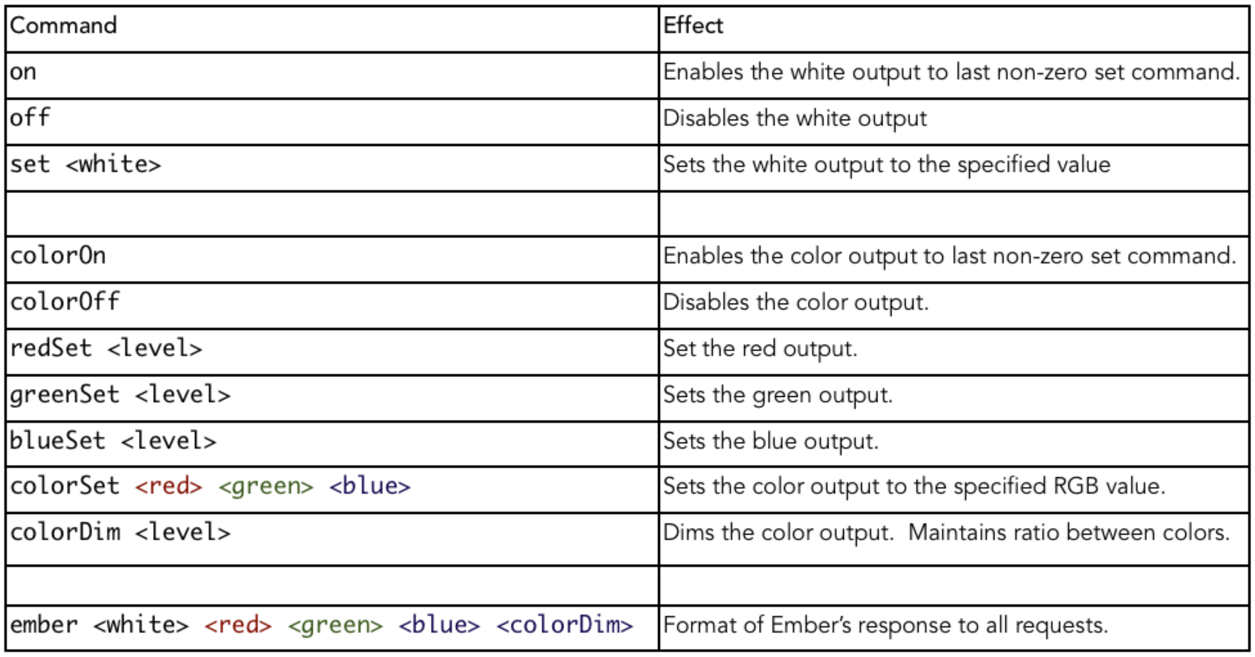
Input: A PWM signal from the microcontroller.

Output: an amplified version of the signal strong enough to power a 12V LED bulb.

3.2 User Interface Specification

Users can interface with the Light Controller using TCP sockets provided by most programming languages, netcat, and similar utilities. The syntax for communicating with the Controller is described below.

3.2.1 Network Protocol



Command	Effect
`on`	Enables the white output to last non-zero set command.
`off`	Disables the white output
`set <white>`	Sets the white output to the specified value
`colorOn`	Enables the color output to last non-zero set command.
`colorOff`	Disables the color output.
`redSet <level>`	Set the red output.
`greenSet <level>`	Sets the green output.
`blueSet <level>`	Sets the blue output.
`colorSet <red> <green> <blue>`	Sets the color output to the specified RGB value.
`colorDim <level>`	Dims the color output. Maintains ratio between colors.
`ember <white> <red> <green> <blue> <colorDim>`	Format of Ember's response to all requests.

Figure 3: Network API

All commands are terminated by a newline character. All commands will receive a response of the form `level 255 255 255\n` where `\n` is a newline character. The response gives the current RGB value of the light as in the "set" command. (This example response would indicate that the light is white at maximum brightness). See figure 3 for a full description.

3.3 Hardware Specification

The hardware components of each Ember include a SilverTel Ag5100 PoE Module, an ATMega328p microprocessor, a WIZnet W5100 Ethernet controller, and four IRF510 MOSFETs. The Silvertel Ag5100 provides hardware support so that the Ember is recognized as a PoE compliant device, and performs voltage conversion to provide 12V DC to the load when supplied with 42.5V - 57V DC by the PoE power source.

The ATMega328p microprocessor manages network commands, sends mDNS discovery packets, and produces the PWM lighting control signals.The WIZnet W5100 Ethernet controller implements a full Ethernet stack. Lastly, the MOSFETs produce a large signal (12V swing) replica of the small (5V swing) PWM signal in order to drive the LEDs. See figure 4 for an visual description.

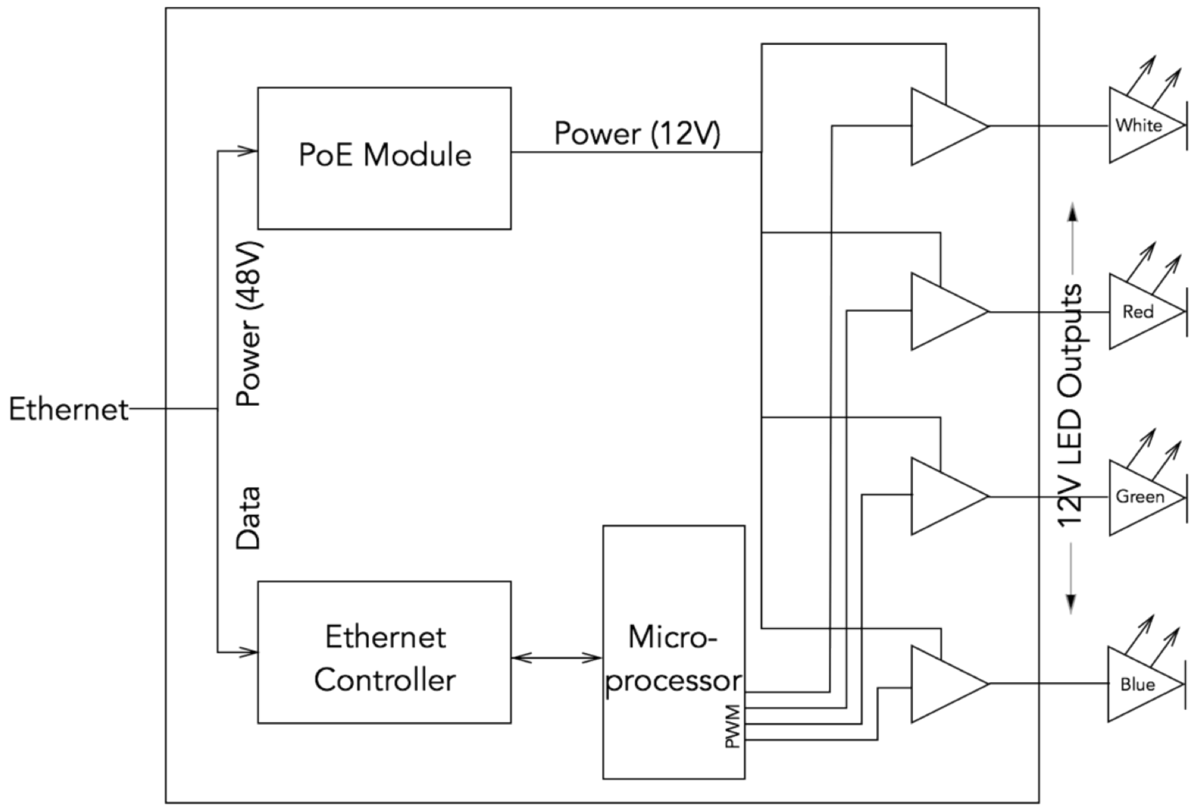


Figure 4: Ember Hardware diagram

3.4 Software Specification

The microcontroller runs a C++ project that implements a TCP server for receiving network commands as specified in the Network Protocol section above. The code also provides network discovery using mDNS and controls the PWM outputs. This is done in a single process, allowing for one socket connection at a time. When a TCP packet is received, it is parsed, interpreted, and the resulting PWM pin is set appropriately.

3.4.1 Software Libraries

Arduino libraries include a WIZnet W5100 Ethernet driver, which is used to host an Ethernet server.

Ethernet Bonjour library for Arduino, written by Georg Kaindl (Kaindl 2006) implements mDNS discovery protocol for Arduino, which allows for Embers to broadcast their IP and other network information to any network device listening for mDNS packets.

3.5 Test Specification

3.5.1 Hardware Testing

The SilverTel PoE module was tested with a powered Ethernet network switch. When connected, the SilverTel successfully classified itself as a PoE compliant device, and was supplied with 42.5V - 57V DC by the switch. The DC-DC converter also performed as desired, and reduced the high supply voltage to the 12V DC at up to 25.5W, required by the load components.

After it was shown that the SilverTel was capable of being identified as a PoE compliant device, and was able to output 12V DC at up to 25.5W, an Arduino Ethernet was connected to it as a load. The 12V DC supply correctly powered the Arduino.

IRF510 MOSFETs are used to amplify the PWM digital output from the ATMega328p to drive the LEDs. These transistors amplifiers were tested by loading them with LEDs, and driving them with a PWM signal from the ATMega328p. The MOSFETs demonstrated proper functionality by producing a large signal (12V swing) replica of the small (5V swing) PWM signal.

To verify the entire hardware system performance, all individual test structures were combined together into a single prototype device. A powered Ethernet switch was connected to the

SilverTel module through standard CAT6 network cable, and provided the PoE supply voltage. The SilverTel successfully accepted the PoE supply, and produced 12V DC, which was capable of powering an Arduino Ethernet, and four LEDs. While in this complete configuration, the Arduino continued to drive the LEDs with appropriate brightness, via the IRF510 MOSFETs.

3.5.2 Software Testing

Software was tested by sending TCP commands from a client program. This client program verified the lighting controller's response and allowed for direct input from the user. Low-voltage LEDs were used to test without a PoE module or amplifier.

4 Hardware Implementation

The final product is a single circuit board containing the entire system. That is, the RJ45 jack, SilverTel Ag5100 PoE module, WIZnet W5100 Ethernet Controller, ATmega328p microcontroller, IRF510 MOSFETS, and wire output terminals together in a clean and compact structure. Although the initial design relied on an Arduino Ethernet and external connections to the other components, a single board design was preferred so that high volume production cost could be minimized.

Finally, two solutions have been designed. The first is a PCB design (figure 5) that implements the entire system on a single board, just as intended. The single board solution will achieve the desired outcome of minimizing production costs, because individual components can be purchased at a low price, and unused components of the standalone devices can be removed completely. The second is a secondary board that contains the system without the Arduino Ethernet included. This secondary board is intended to be an accessory to an Arduino Ethernet, and attaches to it in the same manner as other available Arduino shields. While this second solution introduces more cost, it may be attractive to users who already use Arduinos, and who enjoy the freedom of a modular system.

5 Instruction Manual

5.1 Overview

This manual will guide you in the setup of a network used to power and control Ember LED light controllers. Make sure you have all the necessary components before beginning

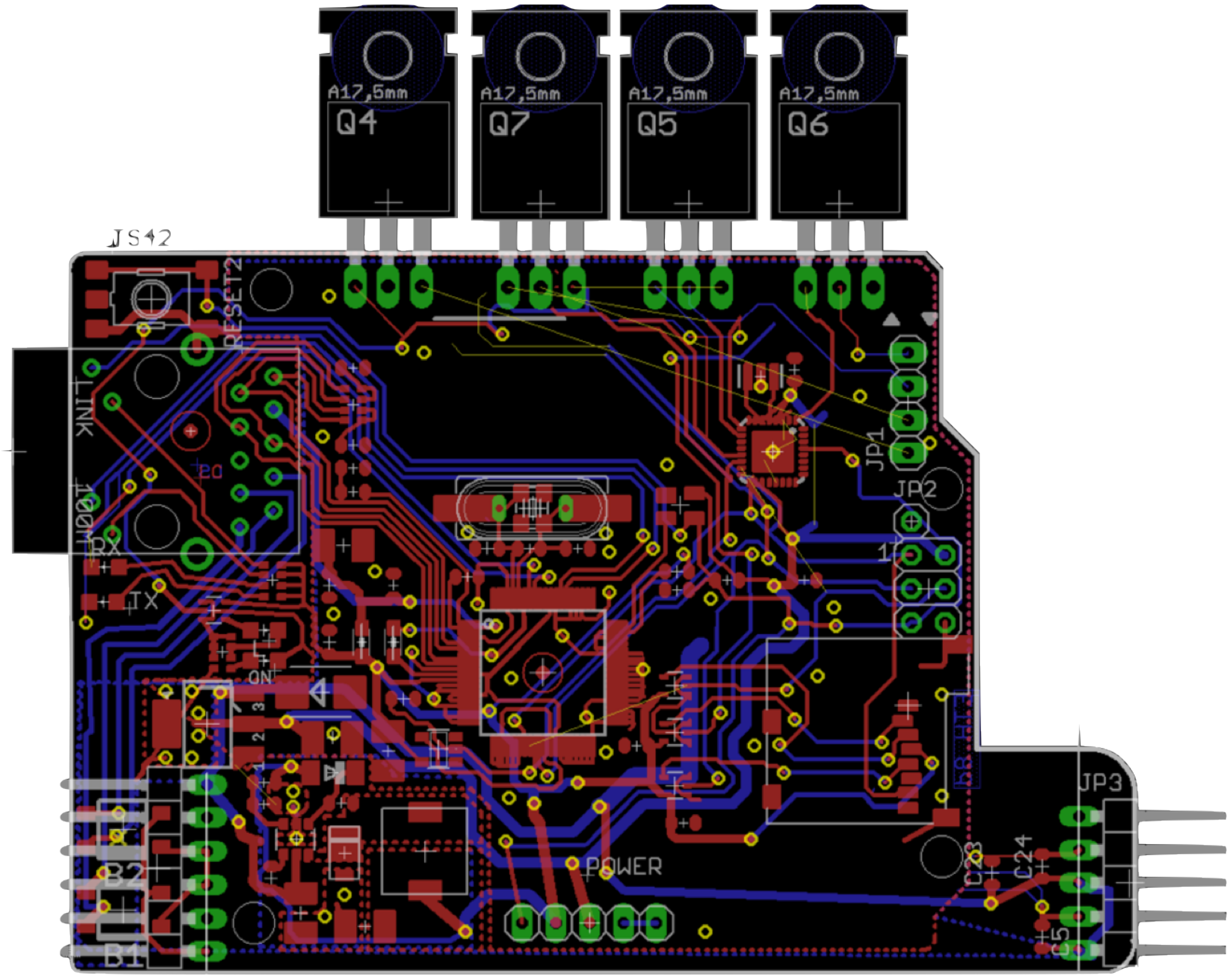


Figure 5: a render of the Ember printed circuit board. Certain components like the USB header were removed.

installation. This manual assumes a working knowledge of Ethernet networks, PoE, and UNIX command line tools. If you are unfamiliar with these tools, consult a local Home Automation Integrator for assistance with your Ember.

Your new Ember is just one part of an Ethernet network. Ember can be added to an existing network, or you can create a completely new network. To show a complete configuration, this manual assumes that a completely new network is being constructed. Of course, if this isn't the case in your particular setup, feel free to skip the unnecessary steps.

The second section explains how to upload new firmware to an Ember. If you have an Ember with firmware already installed or don't want to update, continue to Section 3.

Please Note: This manual is not intended to exemplify all setups and configurations in which Ember can be used and covers only a simple setup for testing and debugging. Ember's TCP Network API is flexible and can be used in applications on mobile devices, advanced home automation control systems, and in any other device that has TCP sockets.

5.2 Required Materials

Depending on how you plan to integrate Ember into your network, you may be able to reuse many of the components below from your existing network.

- Ember Light Controller (1 or more)
- WiFi Router with DHCP enabled (1)
- PoE+ (IEEE 802.3at) Network Switch (1)
  - with enough ports to connect all Embers and other devices
  - with enough total power to power all Embers and LEDs (30W per Ember)
- 12V RGB / White LEDs (1 or more)
  - not to exceed 25W per Ember
- CAT5/6/7 Cable (enough to connect all network devices)
- UNIX based computer
  - need Netcat and Ping utilities or
  - if no UNIX machine is available, ping and netcat alternatives can be used for Windows and other operating systems.

5.2.1 Router Selection

Every Ethernet network starts with a router. Ember needs a router with DHCP to resolve an IP address when it is connected to the network. All consumer routers sold today have this feature, so just about any router will do. Your router will also need to have at least one free RJ45 Ethernet port (to connect the powered switch).

5.2.2 PoE+ Network Switch Selection

Embers are powered by Power Over Ethernet Plus (PoE+). This is specified in the IEEE 802.3at standard. Technically speaking, any switch meeting this standard will work to power Ember, but available switches vary widely in feature sets within this standard. Ember uses efficient control data and a low data rate of 10/100Mbps, so any switch available will be fast enough. This also means Ember won't add considerable load to your network.

When choosing a network switch to use with your Ember(s) make sure you take into account the following requirements:

- Ember needs a PoE switch capable of supplying 30W per port to receive the power it needs to drive the LEDs. Make sure your switch meets this requirement.
- Each Ember can draw a maximum of 30W, so make sure the switch that has a total power high enough to support the number of Ember(s) you want to use.
  - Example: 5 Embers x 30W per Ember = 150W total power

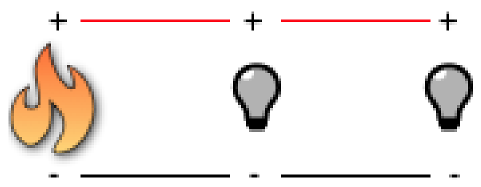
The Ember Team recommends the following switches. These switches are generally low speed, high power, and fairly cost effective. They are also capable of providing power to a fully loaded Ember on every ports.

- 16 Port:
  - TrendNET TPE-T160
- 8 Port
  - TrendNET TPE-T80

5.2.3 Light Fixture Selection

Ember was designed to be used with 12V LEDs. Any LED that powers on at 12V will work with Ember. You can choose between white LEDs and RGB LEDs. Each Ember has one white port and one set of RGB ports, which can support lights chained in parallel as described below. Remember that all the LEDs connected to the Ember may not exceed 25W.

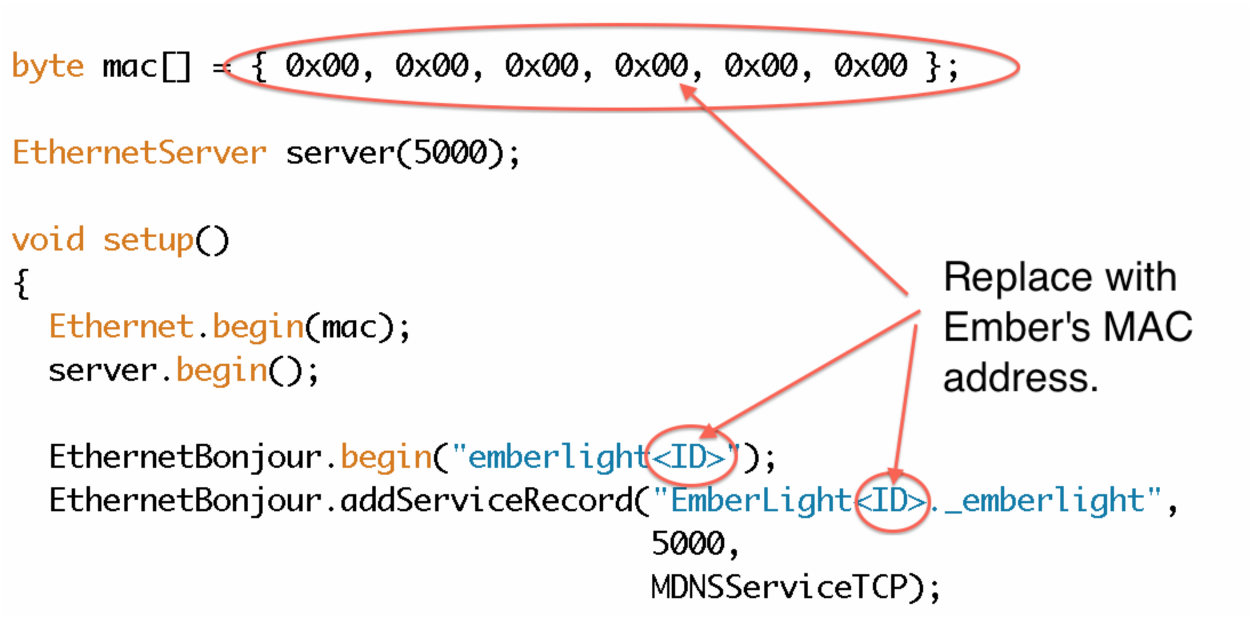
Lights can be wired in parallel off of each Ember output (up to a total power draw of 25W) as shown below.



5.3 Firmware Update

Your Ember comes preconfigured with all the necessary firmware. If you don't want to modify or update this firmware, please proceed to the next section.

If you would like to update or modify the firmware, make sure the MAC address field is changed to match the one printed on the bottom of your board. The places in the code that need to be changed are highlighted in the screenshot below.



```
byte mac[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };

EthernetServer server(5000);

void setup()
{
  Ethernet.begin(mac);
  server.begin();

  EthernetBonjour.begin("emberlight<ID>");
  EthernetBonjour.addServiceRecord("EmberLight<ID>._emberlight",
                                   5000,
                                   MDNSServiceTCP);
```

The firmware can be modified and reloaded like any Arduino project using the Arduino IDE.

5.4 Physical Setup

1. Power down the switch and the router.
2. Connect one of the available Ethernet ports on the router to any of the available Ethernet ports on the powered switch.
3. Connect Ember(s) to available Ethernet port(s) on the powered switch.
4. Connect LEDs to the corresponding Ember output ports as needed.
5. Power up the Router, then the switch, the Ember(s) will power on automatically.
6. Wait 30 seconds for the network to assign addresses and boot up before proceeding to the next section.

5.5 Network Setup

Ember is designed to be controlled by any network device capable of TCP communication. This allows Ember to integrate into home automation systems with a wide variety of interfaces. This section only covers establishing a direct TCP connection to Ember, which can be useful for testing and debugging.

1. Take note of the MAC address on the Ember board (shown below)

2. Ping the Ember using the command (no colons in the MAC address). (Figure 7)

3. Use Netcat to open a TCP connection to the Ember on port 5000. For example, if the Ember is at 192.168.1.14, connect with the command:

   ```
   nc 192.168.1.14 5000
   ```

Once netcat connects, you can send commands by simply typing the command and pressing return. Several example commands are shown in the screenshot below. For a full specification of the network API, consult section 3.2.

5. You're all Done! Enjoy using your new Ember!

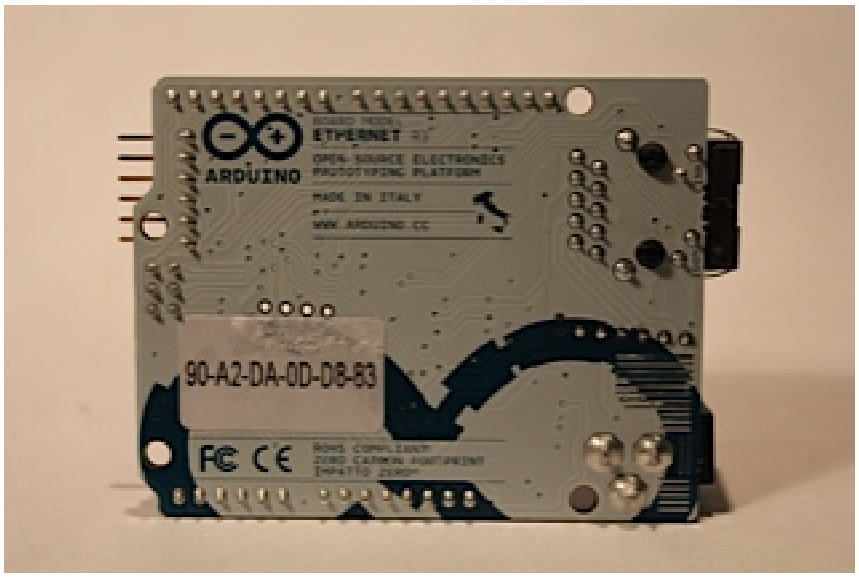
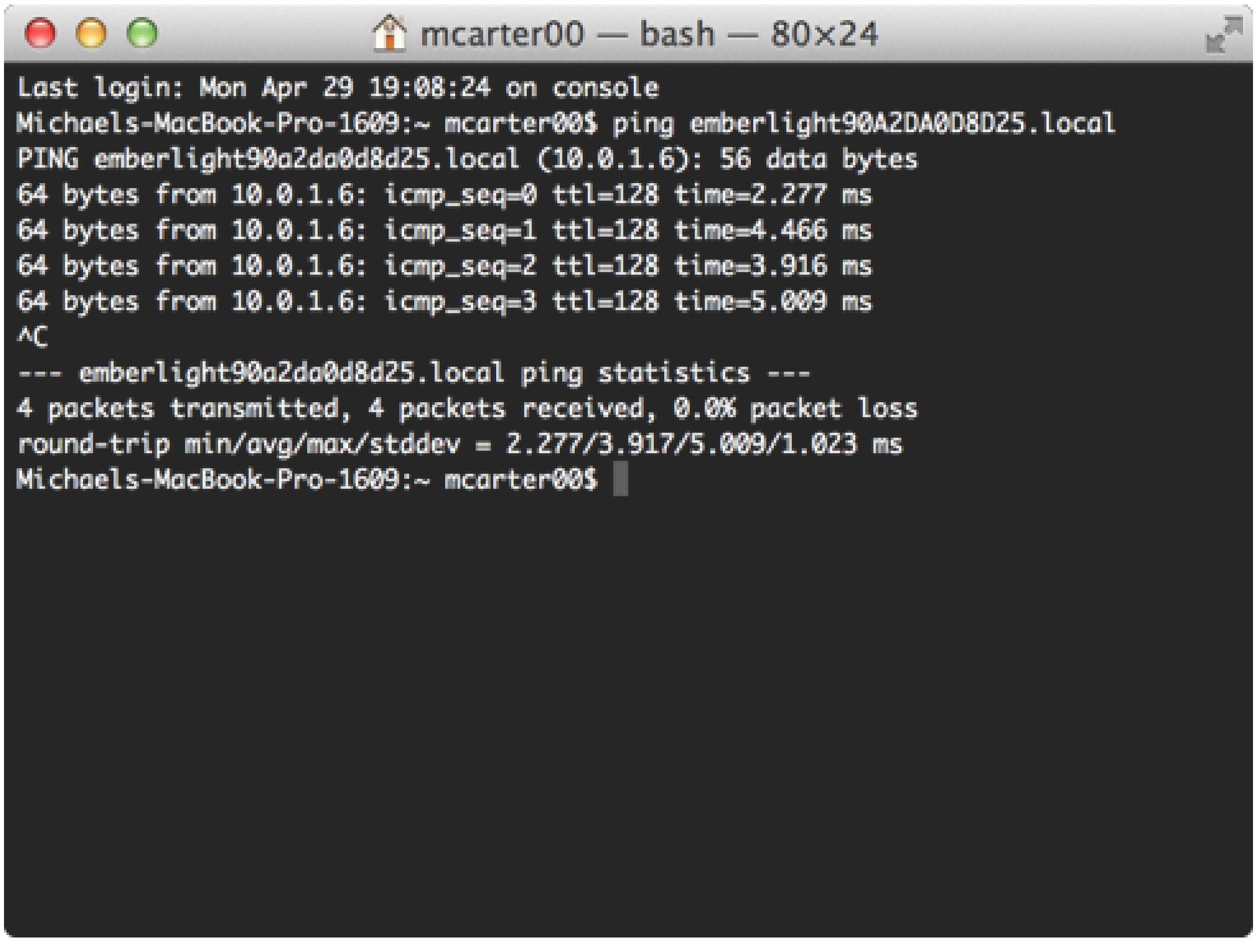
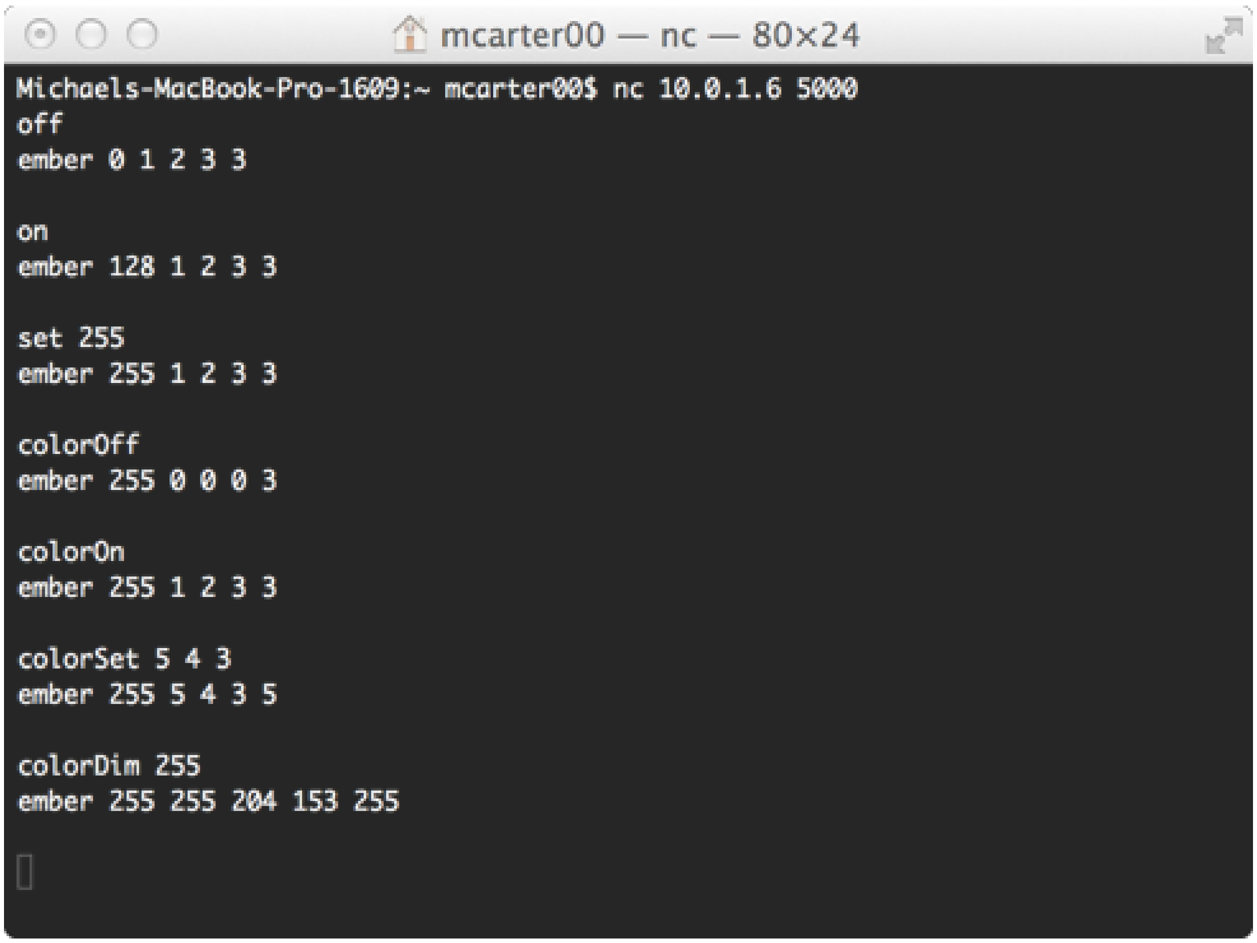


Figure 6: Finding the MAC addres of your Ember board



```
Last login: Mon Apr 29 19:08:24 on console
Michaels-MacBook-Pro-1609:~ mcarter00$ ping emberlight90A2DA0D8D25.local
PING emberlight90a2da0d8d25.local (10.0.1.6): 56 data bytes
64 bytes from 10.0.1.6: icmp_seq=0 ttl=128 time=2.277 ms
64 bytes from 10.0.1.6: icmp_seq=1 ttl=128 time=4.466 ms
64 bytes from 10.0.1.6: icmp_seq=2 ttl=128 time=3.916 ms
64 bytes from 10.0.1.6: icmp_seq=3 ttl=128 time=5.009 ms
^C
--- emberlight90a2da0d8d25.local ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 2.277/3.917/5.009/1.023 ms
Michaels-MacBook-Pro-1609:~ mcarter00$
```

Figure 7: The Ember will respond with its IP address if it has been successfully connected to the network.



```
Michaels-MacBook-Pro-1609:~ mcarter00$ nc 10.0.1.6 5000
off
ember 0 1 2 3 3

on
ember 128 1 2 3 3

set 255
ember 255 1 2 3 3

colorOff
ember 255 0 0 0 3

colorOn
ember 255 1 2 3 3

colorSet 5 4 3
ember 255 5 4 3 5

colorDim 255
ember 255 255 204 153 255
```

Figure 8: Using netcat

6 Appendix A: Standards

- IEEE 802.3at
- TCP - RFC 2581
- mDNS - RFC 6762

7 Appendix B: Glossary

PD - Powered Device. The device powered by a PSE. In this document, the Light Controller.

PoE/PoE+ - Power over Ethernet (plus). A standard for transmitting power and data over the same Ethernet cable. Specified by IEEE 802.3af (PoE) and IEEE 802.3at (PoE+).

PSE - Power Sourcing Equipment. In this document, usually refers to a powered network switch conforming to IEEE 802.3at.

PWM - Pulse Width Modulation. A technique for controlling power sent to a load by sending rapid pulses with a particular duty cycle

TCP - Transmission Control Protocol. A Transport Layer protocol allowing for error free data transmission over an internet network.

8 Bibliography

Aimtec. 2011. “Pulse Width Modulation (PWM) vs Analog Dimming of LEDs.” `http://www.em.avnet.com/en-us/design/designzones/leds/Documents/Aimtec-PWM-vs-Analog-Dimming-of-LE`
`pdf`.

Kaindl, George. 2006. “Bonjour/Zeroconf with Arduino.” `http://gkaindl.com/software/arduino-ethernet/bonjour`.