CAL POLY
SAN LUIS OBISPO

Printed Circuit Board for Introductory Animatronics Course

Final Design Report

Preston Brown
candidate for
B.S., Computer Engineering

Advised by Hugh Smith, Ph.D.
Department of Computer Engineering

California Polytechnic State University
San Luis Obispo, CA

Spring 2015

Table of Contents

Appendices

Introduction

Project Overview / Abstract

For many years, freshmen Computer Engineering students at California Polytechnic State University have taken a course that introduces them to the "processes of electronics manufacturing. They are lectured on concepts such as CAD/CAM design, Design for Manufacture (DFM), documentation requirements, prototyping and production planning" [1]. The laboratory portion of the course allows students to "use hands-on techniques to solidify knowledge of project planning, soldering, automation, hand tool usage and production methods" by manufacturing their own power supply, starting with aluminum sheets, a bag of components, and and an unassembled printed circuit board (PCB) [1].

While the project is popular among students, department faculty see the need for modernization to keep up with today's emerging technical education tools. With the proliferation of easy-to-use development boards and prototyping software, such as the Arduino and Fritzing, people who aren't pursuing degrees can easily master basic microcontroller programming and PCB design. With this in mind, Dr. Smith is testing a course that has the potential to fill this new role. In previous quarters, students have taken this optional elective to build and program an animated stuffed animal. They have been using the Arduino environment and a custom shield that interfaces with the servo motors. In collaboration with Dr. Smith, we have expanded the course by adding custom printed circuit board (PCB) design for each student, as well as allowing for the replacement of the Arduino+shield with a standalone breadboard prototype.

My responsibilities included researching methods of transitioning the project to a breadboard prototype followed by a student-designed printed circuit board (PCB). This involved compiling hardware requirements to operate an ATmega328p microcontroller separate from the Arduino board. I also created breadboard and PCB reference designs using the Fritzing prototyping software environment. Following my reference design, I created a set of instructions that leads students through the process, starting with a bag of parts and a breadboard, through the creation of a breadboard prototype, and culminating in a custom-designed PCB.

Stakeholders

This project is supervised by Hugh Smith, Ph.D. of the California Polytechnic State University Computer Engineering Department. He is the instructor for the experimental animated stuffed animal course (CPE 200) and wants to expand the course to include PCB design, as mentioned in the project overview. The project materials are funded by HuSmith, Inc., a private software consulting company located in San Luis Obispo, CA.

Other stakeholders include the Cal Poly Computer Engineering Department, the Cal Poly Industrial/Manufacturing Engineering Department (henceforth referred to as CPE and IME, respectively), and students studying Computer Engineering at the university. The CPE and IME departments want to use this course to explore alternatives to the introductory electronics manufacturing course (IME 156). The students need a fun and engaging project to introduce them to the necessary skills for PCB design and electronics manufacturing that are included in a well-rounded computer engineering education.

Client Needs and Discussion

Throughout my communication with Dr. Smith, I gained an understanding of the underlying goal of the animatronics course and the PCB project. I was tasked with developing the PCB project for CPE students, aligning with requirements summarized below.

- ➔ Create step-by-step instructions to create a breadboard prototype of the PCB design.
- ➔ Complete a design of the microcontroller PCB system.
- ➔ List known issues and common design mistakes for future iterations of the course.
- ➔ Develop instructions for interacting with the Fritzing prototyping environment to model a breadboard prototype and design a printed circuit board.
- ➔ Research and implement methods to mitigate servo motor noise in the prototype and PCB.

The CPE Department and Dr. Smith are looking for a fun and cost-effective way for students to learn about electronics manufacturing and provide them with a wholesome experience that will build skills necessary to become a professional computer engineer. Focusing more on robotics and firmware programming will provide a much more relevant experience for CPE students as compared to assembling a power supply.

Goals and Objectives

As mentioned above, the overall goal of this project is to expand an existing animatronics project course to include breadboard prototyping and PCB design and assembly. Below, I have enumerated specific project goals and objectives.

Goals

1. Create a series of laboratory assignments for CPE 200 students.
2. Develop lecture materials and curriculum to guide students through the completion of an animated stuffed animal project.
3. Design and implement a breadboard solution with auxiliary parts (e.g. servos, power board) for the stuffed animal.
4. Research the feasibility of shrinking the project into a printed circuit board.
5. Implement a reference PCB solution.
6. Guide students through the project and provide them with a valuable learning experience.

Objectives

1. Research external power methods for servo motors interacting with a PCB.
2. Develop initial Fritzing wiring diagrams for the standalone breadboard controller.
3. Compile a bill of materials for required parts.
4. Assemble the breadboard prototype and verify correct operation.
5. Develop instructions for students to implement the breadboard controller.
6. Research feasibility and methods of transitioning the project to a PCB.
7. Design a PCB schematic in Fritzing for the animatronics controller.
8. Order a copy of the PCB design and verify it works with an animated skeleton.
9. Test the PCB controller.

Outcomes and Deliverables

Upon project completion, deliverables will include a set of instructions, breadboard wiring diagrams, a PCB design, and a completed animatronics controller. The project will provide a foundation for leading students through future iterations of this course. The written materials, found in the Appendix, include the following described items:

Breadboard Fritzing Steps and Files:

A collection of fifteen Fritzing breadboard diagrams, designed as iterative steps from an empty breadboard to a complete, standalone breadboard animatronics controller prototype. Each step has a nearly photorealistic representation of the breadboard system. These involve adding specific subsystems one step at a time (e.g. barrel jack power subsystem, oscillator subsystem, etc.). The Fritzing (.fzz) files are not included in the Appendix, but have been provided to Dr. Smith. Included with the design files is a Fritzing *parts bin file* (.fzbz). This is a collection of the necessary parts for the students' breadboard prototypes and ensures that the students will use parts that have accurate PCB layouts and pin configurations.

Breadboard Instructions:

A document outlining the steps of collecting parts and performing the parallel tasks of assembling the breadboard prototype and wiring it in a Fritzing diagram. This includes instructions on importing the course's parts bin, the wiring configurations of the controller and its subsystems, and basic information about how many of the components work. This document includes some screenshots of the Fritzing environment and pictures of the components so students know what to look for.

Printed Circuit Board (PCB) Final Design File:

The "Step 15" Fritzing design file (.fzz) contains a final (second revision) PCB design, which takes into account all design considerations discovered while researching methods of PCB implementation. An image of both PCB revisions are located in the Appendix.

PCB-Related Instructions:

Documents outlining steps related to the requirements to implement the animatronics controller on a PCB. The design considerations document includes information necessary to ensure ATmega328p operation on a standalone PCB. There are also instructions on testing the board for correct electrical connections prior to assembly, as well as a document containing notes and reminders for students to reference while soldering parts to their PCB. I have also included instructions on ordering a PCB from Advanced Circuits (www.4pcb.com), which is specific to the Spring 2015 iteration of the course.

Fritzing 'Getting Started' PowerPoint Presentation:

A short presentation meant to familiarize students with the Fritzing environment's breadboard and schematic tabs. This leads them through a basic tutorial, including drawing up a two-LED circuit on a breadboard connected to an Arduino in the Fritzing environment. This exposes students to the process of setting up a breadboard diagram as well as making a clean schematic.

Bill of Materials:

A document containing a list of PCB components, their quantities, and expected prices. This also contains the pricing information for the PCBs ordered for the Spring 2015 iteration of the course.

Background

This project required background research on the Fritzing prototyping environment, the feasibility of reproducing the Arduino/shield project on a standalone PCB, and PCB servo noise mitigation. Details describing my findings for each topic are below.

The Fritzing Prototyping and Development Environment

Fritzing is an open-source software tool that is designed to support designers and those learning about electronics to create product prototypes, schematics, and printed circuit board (PCB) designs. It is developed by the *Friends-of-Fritzing* foundation in conjunction with *IXDS*, who aim to "[foster] a creative ecosystem that allows users to document their prototypes, share them with others, teach electronics in a classroom, and layout and manufacture professional PCBs" [2]. The software has three major components we touched on in the Spring 2015 CPE 200 course: breadboard prototype design, circuit diagram layout, and printed circuit board schematic design.

Core Interface Panes:

Parts Pane

The *Parts* pane, shown in Figure 1, contains all part bins that come with the Fritzing environment, which include vendors such as Sparkfun, Arduino, and Analog Devices. A Fritzing "part" constitute a photorealistic drawing of the part (used in the breadboard prototype view), a circuit diagram symbol (used in the schematic view), and one or more PCB part layouts. The *Parts* pane allows the user to create his/her own parts bin, create custom parts, and import parts bins from third parties.

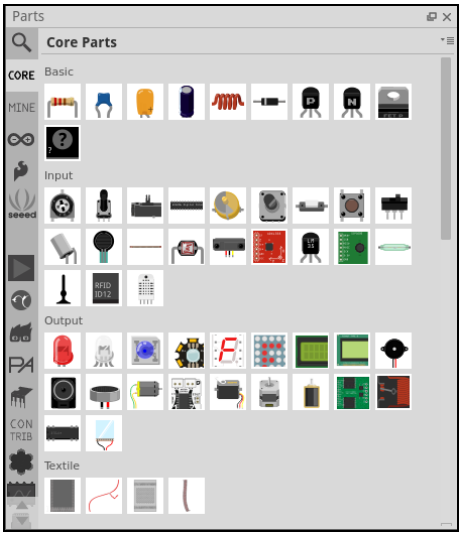


Figure 1 *Fritzing Parts Pane*

There are many different versions of common parts, like LEDs and capacitors, each with different options for their PCB layout counterparts. To make sure that each student would choose a part with the correct PCB layout, I created a CPE 200 Fritzing parts bin to provide to the students and documented the correct PCB part packages to use. Part characteristics like this can be edited in the *Parts Inspector*.

Parts Inspector Pane

The *Parts Inspector*, shown in Figure 2, lets the user change the properties of a given Fritzing part. This includes nominal values for parts like resistors and capacitors, voltage ratings, packages, colors, etc. Some of the part options, such as resistance values, are purely for documentation purposes since Fritzing does not provide any form of circuit simulation.

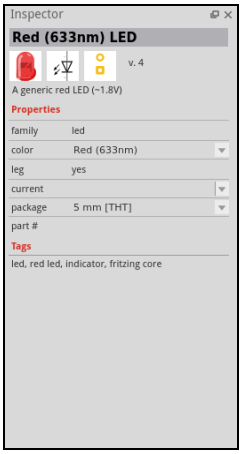


Figure 2 *Parts Inspector Pane*

Breadboard Prototype Design:

The *Breadboard View*, shown in Figure 3, allows users to add any amount of parts to wire up a breadboard prototype configuration and model a project design. In this view, users can place and drag components and wires, and the software saves the underlying circuit connections, using them to keep track of the model in *Schematic* and *PCB* views. Users can add bendpoints to the wires and change their color, allowing them to easily differentiate and keep track of important connections.

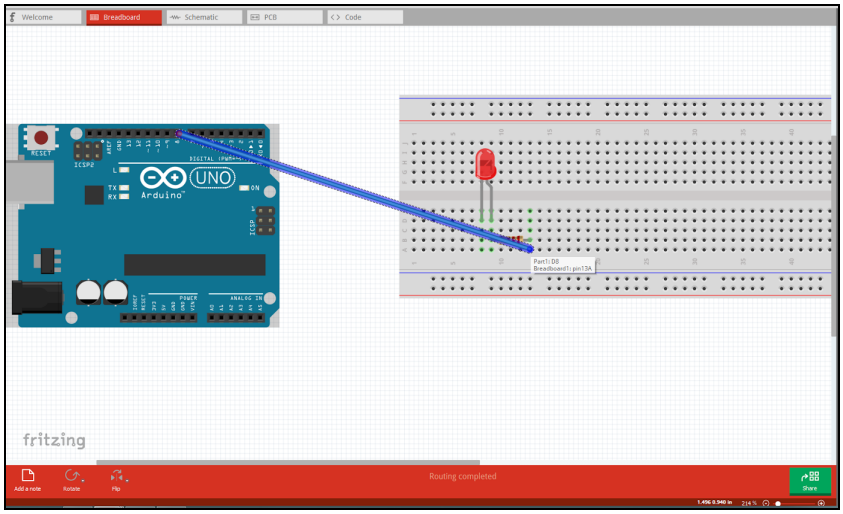


Figure 3 *Fritzing Breadboard View*

We took advantage of Fritzing's ability to model a photorealistic breadboard prototype by having students first wire their physical breadboard and then use it as a reference to duplicate the design in Breadboard View.

Schematic View:

Fritzing also comes with a *Schematic View*, shown in Figure 4, which includes circuit component drawings and pinouts for every Fritzing-compatible part. Adding and connecting components in the breadboard view automatically generates a schematic counterpart. Like most PCB design software, Fritzing keeps track of "ratsnests" for each electrical node in the circuit and connects them with dotted lines. While the schematic view isn't functionally useful, it is a great documentation tool and can be used to make neat, professional system diagrams.

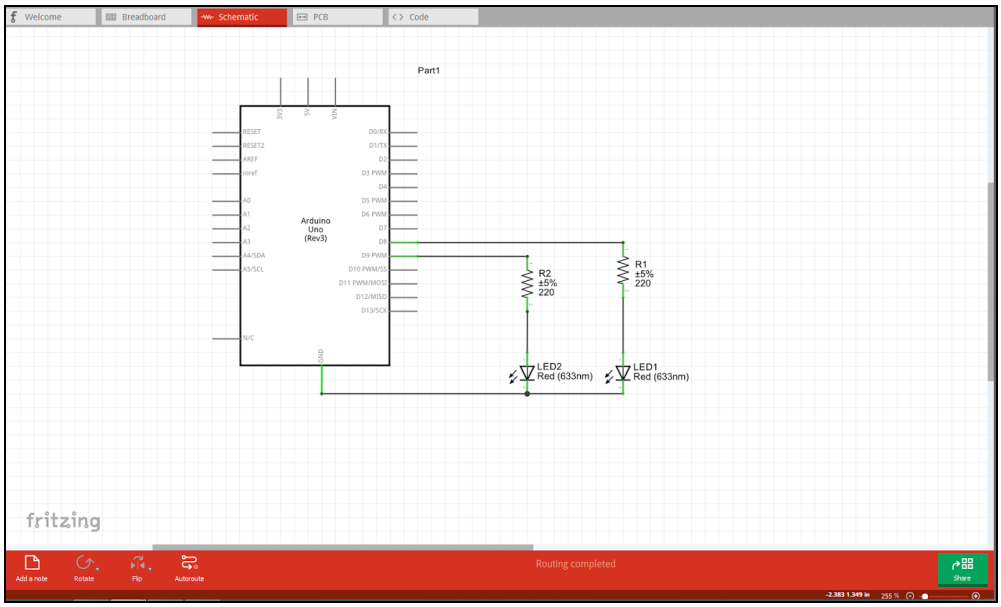


Figure 4 *Fritzing Schematic View*

Printed Circuit Board (PCB) View:

The focal point of this project is centered upon Fritzing's PCB design capabilities. While Fritzing is not a fully-featured PCB design tool, it is a fantastic tool for beginners looking to understand the basics of PCB design. Its user-friendly interface and simple operations make it a perfect tool for an introductory course like the CPE 200 animatronics course. In the view, users can make a PCB with any outline (provided they import a .svg file), with up to two layers of traces. The software includes common PCB design features like design rule tweaks and verification, autoroute, copper fills, ground fills (copper fills automatically connected to ground), and extended Gerber file generation. For this iteration of the course, we instructed students to do very minimal manual routing, and instead stick to the autoroute function. A student's completed design in the PCB view is shown in Figure 5 below. My full reference schematic designs (both revisions) are in the Appendix.

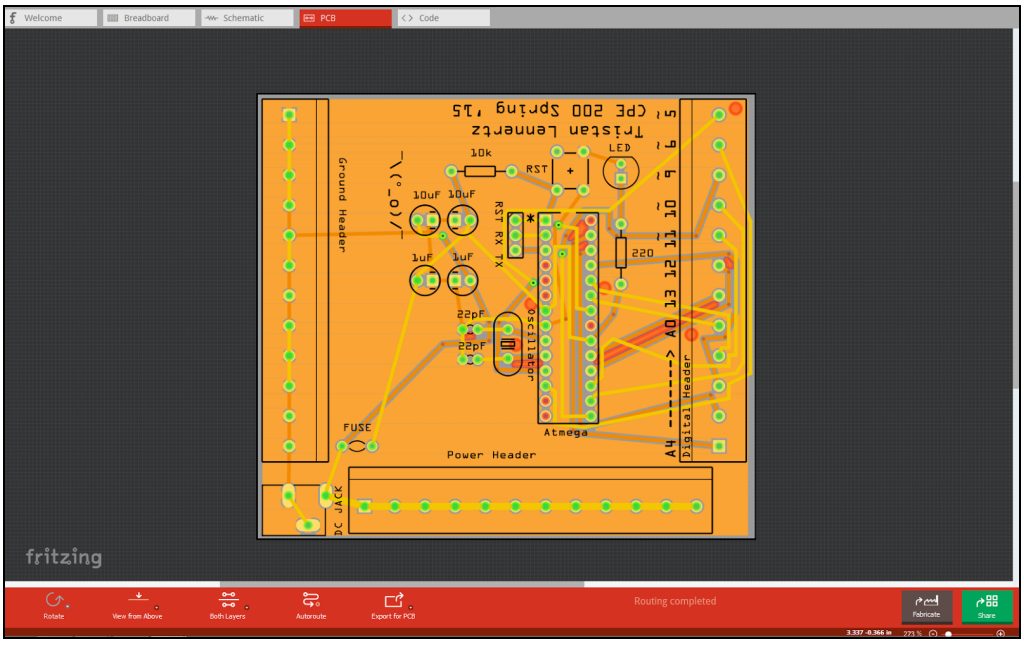


Figure 5 *Fritzing PCB Design View*

The option to export the PCB to extended Gerber specifications allows for easy adaptation of the course to use any PCB manufacturer. In our case, we chose to order PCBs from Advanced Circuits in Colorado. Fritzing also has an option to order through their fabrication contractors/facilities, although at a slight premium.

The Fritzing prototyping environment is a great tool for designers of all skill levels who are fabricating relatively simple projects and don't need to learn the nuances of complicated software like OrCAD or CadSoft EAGLE

Standalone PCB Controller Requirements

Clock Configuration:

To operate the Arduino's ATmega328p microcontroller independent of the Arduino board, a specific configuration with extra components is required. The ATmega 328p microcontroller has a few digital clock source options. The first option is activating an internal 8 MHz internal RC (resistor-capacitor) oscillator. An internal clock divider is set to a division by 8, resulting in a 1 MHz system clock. The other option is connecting a full-swing crystal oscillator, accompanies by a resonant circuit, to the XTAL1 and XTAL2 pins on the microcontroller. This allows for a smaller frequency tolerance, decreases susceptibility to noisy environments, and greatly increases the clock speed (the clock input accepts signals up to 20nMHz when Vcc is above 4.5 V) [3]. We chose a 16 MHz external oscillator for these advantages and to ensure compatibility with existing PWM libraries that rely on a 16 MHz system clock for accurate signal generation.

Using a 16 MHz oscillator increases power consumption, but is negligible in our circumstances since the PCBs are connected directly to a wall power supply. To ensure correct operation, the crystal must be connected in the configuration shown in Figure 6. The XTAL1 and

XTAL2 pins of the ATmega328p are are the input and output of an inverting amplifier, which is configured for use as an oscillator. The capacitors ensure a rail-to-rail swing on the XTAL2 output at the crystal's resonant frequency. Atmel recommends ceramic capacitors of 12-22 picofarads (pF) for 16 MHz crystal oscillators [3].

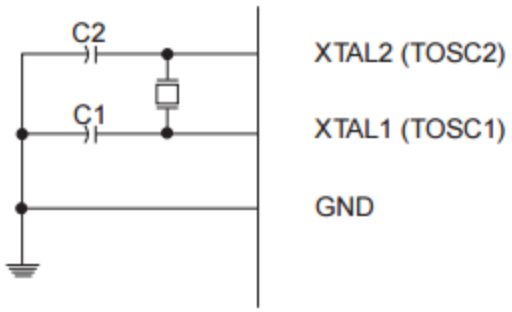


Figure 6 *Crystal Oscillator Connections [3]*

With regards to PCB placement, Atmel recommends that all PCB designs with an AVR chip should "place the resonator as close to the AVR as possible and shield the resonator by surrounding it with a ground plane" [4]. The PCB designs for the animatronics course keep the crystal close and use a ground plane that fills the entire area of the board, thus the recommended requirements are met.

Reset Functionality:

All AVR chips, like most microcontrollers, come with a reset pin that can be triggered to perform a chip reset, which re-starts the bootloader and loads a program from memory. On AVR chips, the reset pins are *active low*, meaning they need to be kept at 5V during normal operation, and pulled down to logic low (0 V) to initiate a reset. We chose to use a momentary button that shorts the pin to ground while the button is pressed. Atmel recommends that the switching

mechanism is used with a series resistance to limit the current spike that happens when the button is pressed, since the PCB trace and switch metal have a small inductance that can cause voltage spikes that will likely exceed input ratings for the reset pin [4]. We mitigated this by using a series resistance value larger than recommended (Atmel recommended at least 330 Ω, we used 10 kΩ), which will severely limit the current spike.

Programming interface:

The Arduino Uno uses a separate on-board microcontroller (an ATmega16U2), programmed as a USB-to-serial converter to program the ATmega328p chip [5]. To cut down on costs and remove complexity, the Arduino board can be used as a programmer to target the ATmega328p while it is soldered into the PCB.

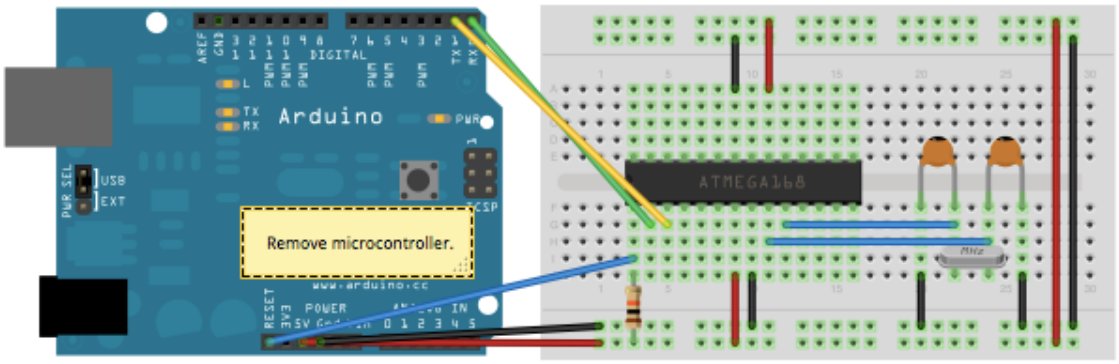


Figure 7 *Arduino as a programmer [6]*

To use the Arduino as a programmer, the user must remove the microcontroller from the board (which is presumably completed if it is already soldered to the PCB), and reroute the serial connections and reset signal to the PCB according to Table 1 below. Both the Arduino board and the PCB need to share the same ground reference so the data is transmitted properly, which can be

done with a jumper wire. If the PCB is *not* being externally powered, the user will need to connect the 5V pin on the Arduino to the main power rail on the PCB, as shown in Figure 7 above.

Table 1

Arduino	ATmega328p
RESET	*RST (pin 1)*
RX	*RX (pin 2)*
TX	*TX (pin 3)*

If the PCB is being externally powered, the two power rails should not be connected together, since the Arduino board is already being powered with a clean 5V from the USB port. It is okay that the Arduino and PCB will have two different power levels (as long as there is a common ground), since the PCB will easily interpret any 5V signals from the Arduino as logic high, allowing for accurate data transmission.

Noise Mitigation

The servo motors that control the limbs of the animated bear will be powered from the same power rail as the microcontroller. This is an issue since the servos may introduce noise in the controller system. The microcontroller needs a relatively clean and stable power source to operate correctly and efficiently. To mitigate possible servo noise, we added filtering capacitors between power and ground rails. We added 10 microfarad and 1 microfarad capacitors close to the microcontrollers to avoid sharp voltage changes in the power rail and filter out high frequency noise.

Normally, it would make sense to have a power and ground plane for these rails, but doing so (along with decoupling with capacitors) can create a high current loop, resulting is noise

spreading across all components of the PCB. This is shown in Figure 8 below, which was taken from a design considerations document by Atmel. According to Atmel, this would cause the entire ground plane to act as an antenna for the noise, instead of only the high current loop. In our design, we only placed a ground plane and placed thick traces straight from the DC input power to the headers that supply power to the servos.

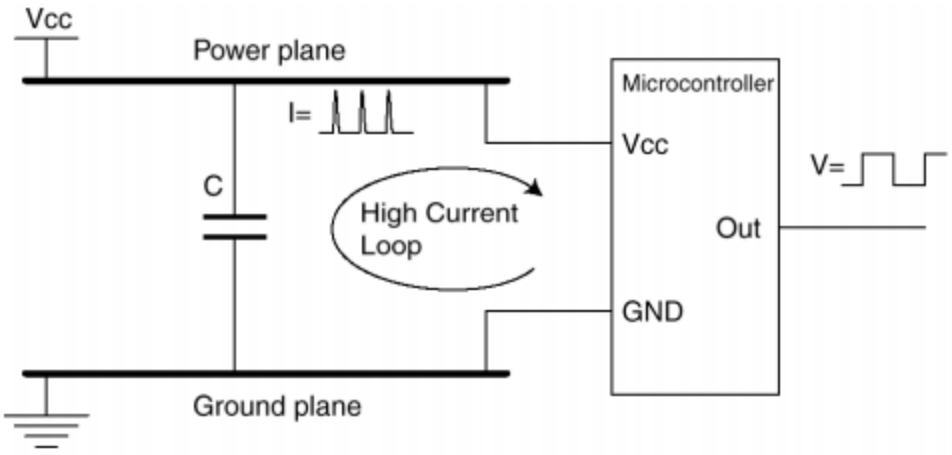


Figure 8 *Design issues for far-away capacitors and power/ground rail plane decoupling.*

Atmel recommends further decoupling with a ferrite bead, but the absence of a power plane in our design, as well as the fact that our board is small and the decoupling capacitors are close to the microcontroller, we will not run into these issues. According to Analog Devices, the large amount of metal in a ground plane will have as low a resistance as possibly, as well as a low inductance, offering the best possible conduction and minimizing differences in voltage across the plane [7].

Final Design

Hardware

The final outcome of the project consists of a fabricated controller board, two revisions of the design, edits to student PCB designs, as well as a collection of instructions, notes, and presentations for future iterations of the course. The latter can be found in the Appendices.

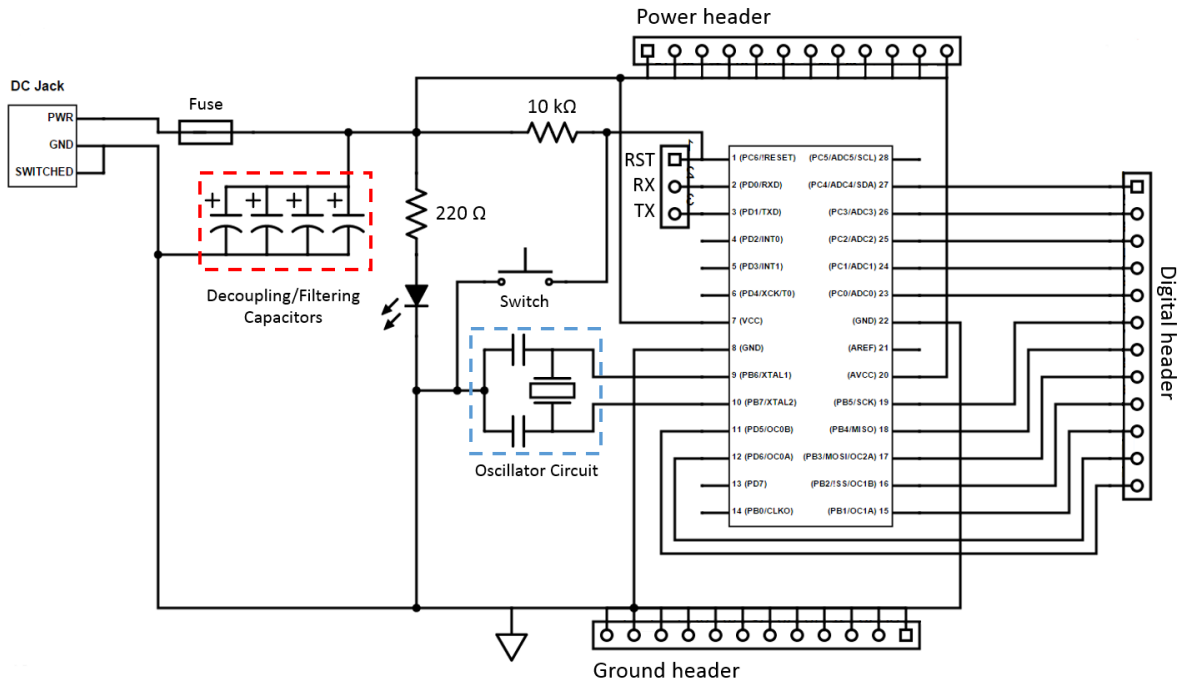


Figure 9 *Final design schematic of the PCB controller system*

Figure 9 shows the final design schematic of the PCB controller system. The final design incorporates a fuse to ensure that the servos pull no more than five amps from the wall DC supply. The three sets of twelve headers are screw terminals that ensure secure connections to servos and sensors for the students' animated stuffed animals. The wires need to reach inside the bear and have a secure connection that cannot be pulled out. The three-wide pin header in the middle of the

figure enables system programming from the Arduino, as described in the previous section. The ATmega328p will receive the RST, RX, and TX signals from the empty Arduino during sketch programming. The design also includes an LED to show when the system is being powered. The final breadboard schematic design as viewed in the Fritzing environment is shown below in Figure 10. I had to model the fuse as a capacitor with 200-mil spacing in Fritzing, as the software didn't include an appropriate part for a fuse. Since Fritzing is not a circuit simulation tool, this did not make any difference. Unfortunately, due to this substitution, the through-hole diameters were just too small on the final production PCBs. The next iteration of this project will require modeling the fuse with a Fritzing part that has the correct hole diameter, but still has 200-mil spacing.

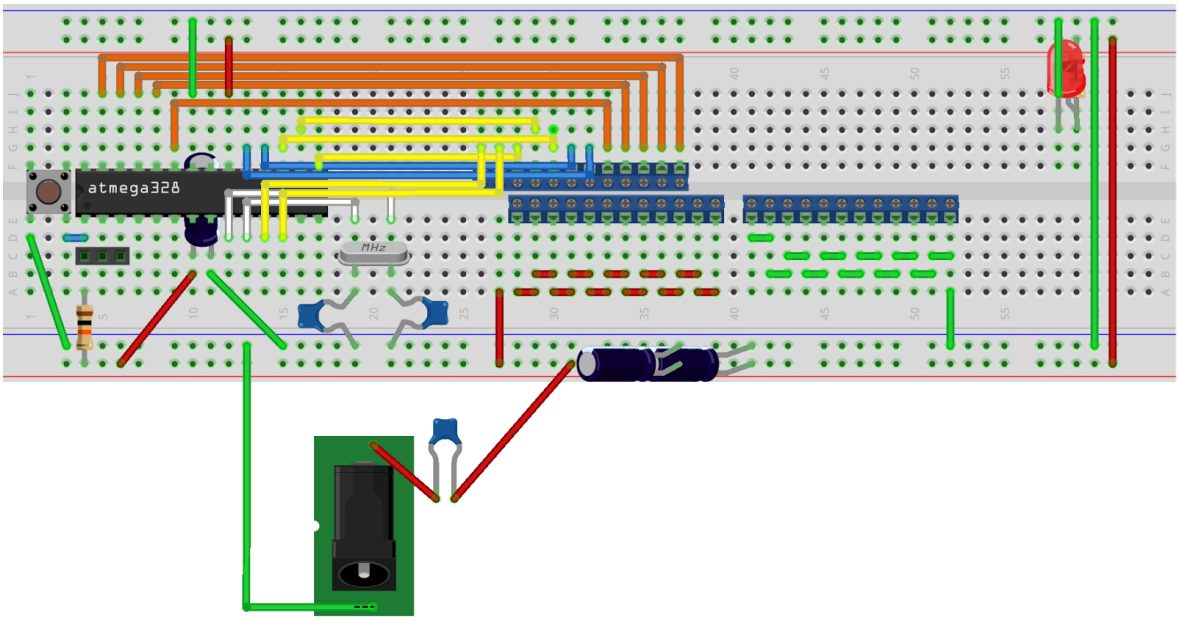


Figure 10 *Breadboard drawing of the final design*

The final PCB comes in two revisions. The first revision was the only one sent to the fabrication plant to cut down on costs, since all of its mistakes were easily fixed without needing a new board. The first revision's layout can be found in the Appendices. The second revision, which

fixed issues that are described in *System Integration and Testing*, is shown below in Figure 11. One difference between the initial and final designs is the addition of a five amp fuse to ensure the servos don't pull too much current. Yellow traces are located on the top layer, and orange traces (along with the orange copper fill) are on the bottom layer.

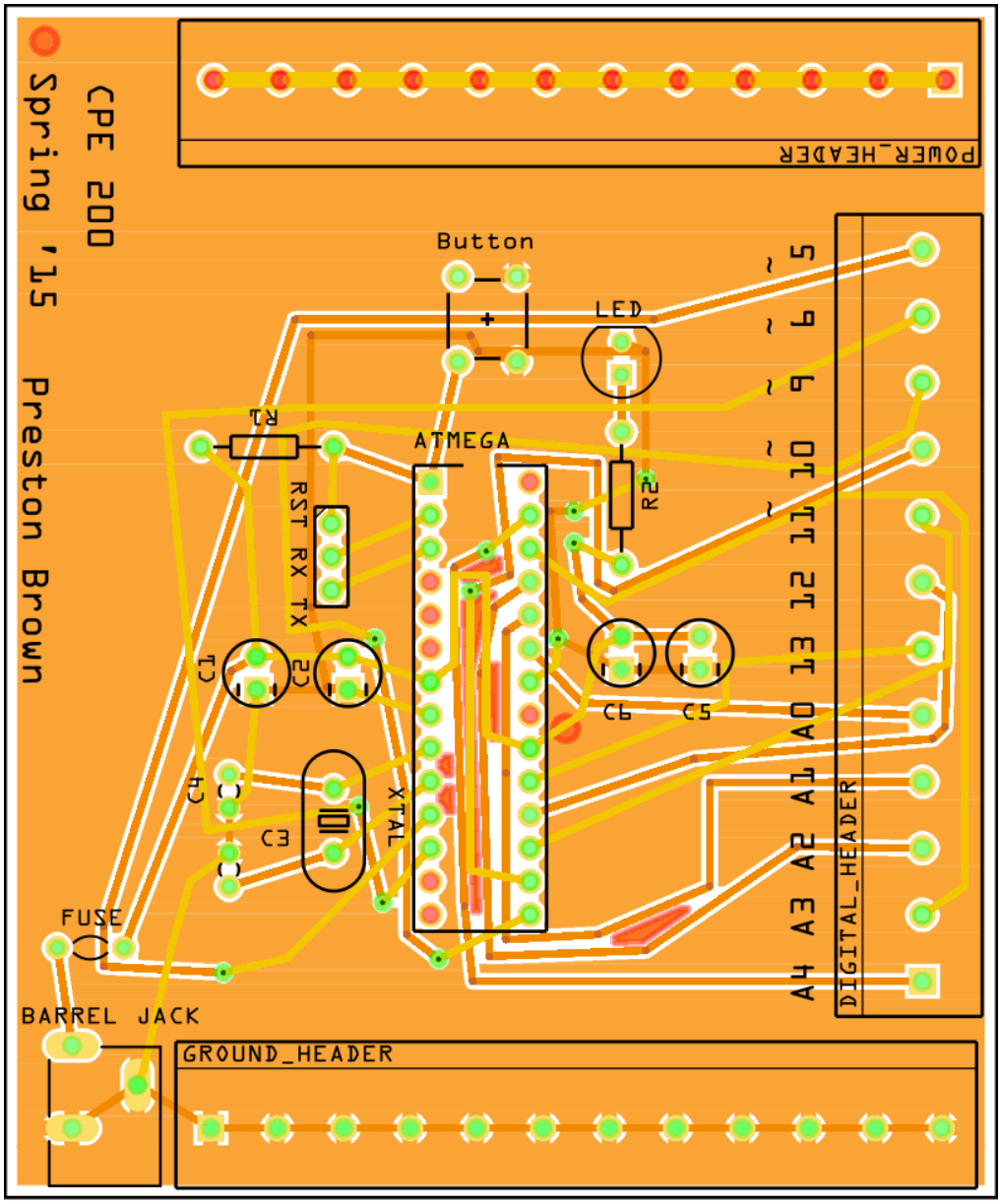


Figure 11 *Final PCB layout*

Software

Although the software libraries used by the students were written by Dr. Smith, it is important to touch on their interaction with the system. The students in the course use an online development environment, *Codebender*, to develop Arduino sketch code. Using servo object libraries, PWM libraries, software state machines, and ADC libraries, the students were able to develop fully-functional animated animals. Much like the Arduino, students can attach their servo and analog sensor wires to the digital header and direct the software to interact with these specific pins. By labeling the corresponding pins on the PCB silkscreen, we expect a similar development experience given by the Arduino. In more advanced applications of microcontrollers on PCBs, the normal ATmega pinout is used, requiring extensive reading of the datasheet.

System Integration and Testing

Continuity Testing

Prior to soldering the parts to the printed circuit board, I used a multimeter to continuity test corresponding pads. Tested points included testing chip continuities between power and ground interfaces. Pins 7 and 20 on the ATmega328p are the Vcc pins need to be continuous with the power pin on the DC barrel jack, while pins 8 and 22 need to be continuous with the ground pin. I continued to test the power and ground headers to make sure they were all connected to the DC barrel jack's relevant pins.

I then tested for continuity between all of the digital pins and their respective screw terminals on the PCB. I finished up continuity testing by making sure the button pins were not

shorted in a way that would break reset functionality and by verifying that the oscillator pins were connected to their respective ATmega pins and were not shorted together.

Servo Testing

I powered up the PCB with the wall DC barrel jack and loaded servo test sketches onto the PCB with the Codebender development environment. I ensured that the servos could move for a sustained period of time and that multiple servos could act simultaneously. The motors had no problems with drawing heavy current from the wall supply and I noted no noise issues on the board. Programming the board through the Arduino proved to be effortless, as the computer interfaced with the system as if it were an Arduino. By rerouting programming signals via jumper wires, students do not need to jump through any hoops to load sketches on their PCB, but they do need to keep an Arduino around.

Initial PCB Design

After soldering the parts to the initial PCB design, I noticed a few issues. First, I had rotated the DC barrel jack PCB pad incorrectly (as seen in Figure 12), which did not affect system operation as a whole. I had also forgotten to route a direct path from the DC barrel jack to the power headers for heavy current flow. My PCB routing configuration would have pulled all of the current *through* the microcontroller, exceeding rated current limits. I fixed this by soldering a direct path onto the back of the PCB, which is shown in Figure 13 as the white wire.

Initial testing showed that power was not being delivered to many parts of the board correctly. After probing the entire board with a multimeter, I realized I mistakenly connected most of the components to a common ground that was routed to the switched jack pin on the DC barrel.

This pin disconnects from the ground pin whenever the barrel supply is plugged in. This explains why continuity testing showed that the connections were correct while the DC barrel was unplugged. I fixed this by soldering the switched jack pin to the ground pin with the black wire shown in Figure 13.

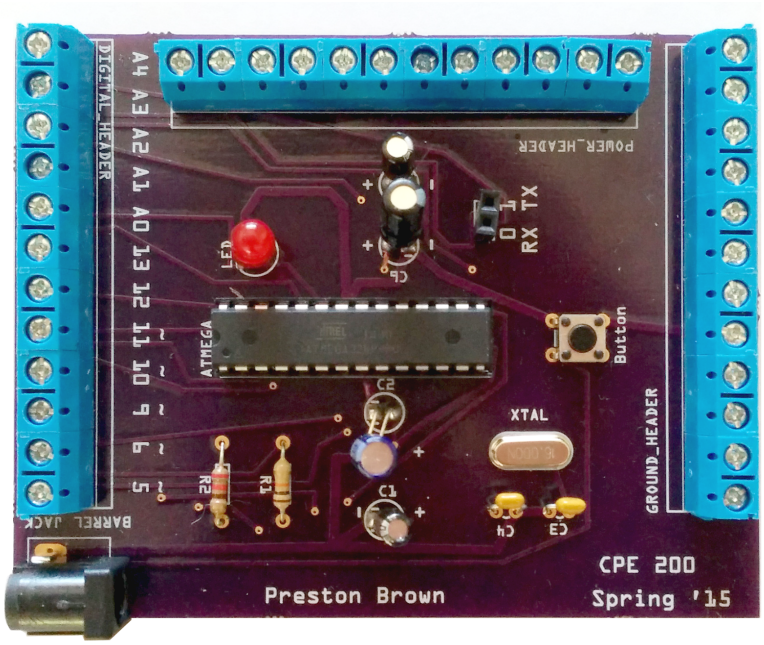


Figure 12 *Manufactured PCB, design revision 1*

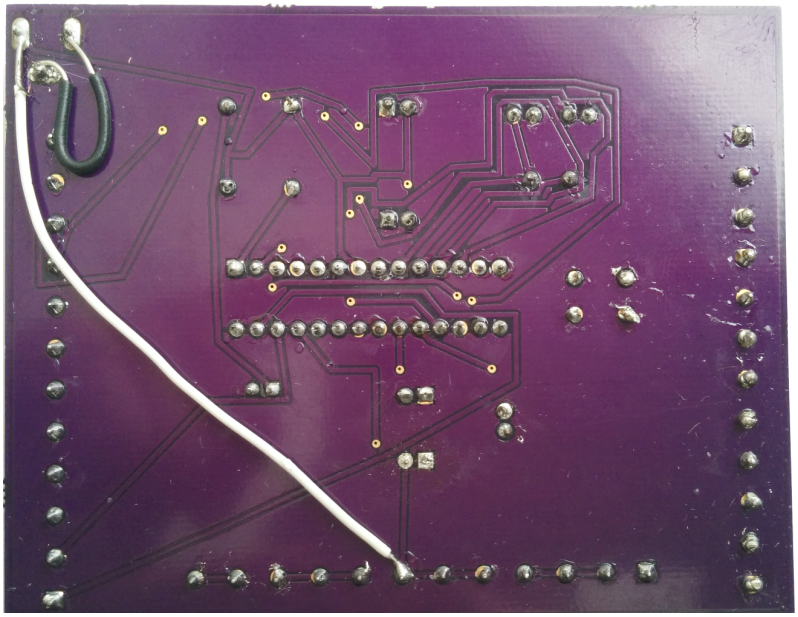


Figure 13 *Design fixes for first PCB*

References

[1] “Course Catalog 2015-2017.” California Polytechnic State University. 8 May 2015. Web. http://www.catalog.calpoly.edu

[2] “Fritzing.” Friends of Fritzing. Web. 12 May 2015. http://www.fritzing.org

[3] “Atmel 8-bit Microcontroller Datasheet.” Atme, Inc. October 2014. Web.

[4] “AVR042: AVR Hardware Design Considerations.” Atmel, Inc. September 2014. Web.

[5] “Arduino Uno.” Arduino. Web. 12 May 2015. http://www.arduino.cc/en/Main/ArduinoBoardUno

[6] “From Arduino to a Microcontroller on a Breadboard.” Arduino. Web. 12 May 2015. http://www.arduino.cc/en/Tutorial/ArduinoToBreadboard

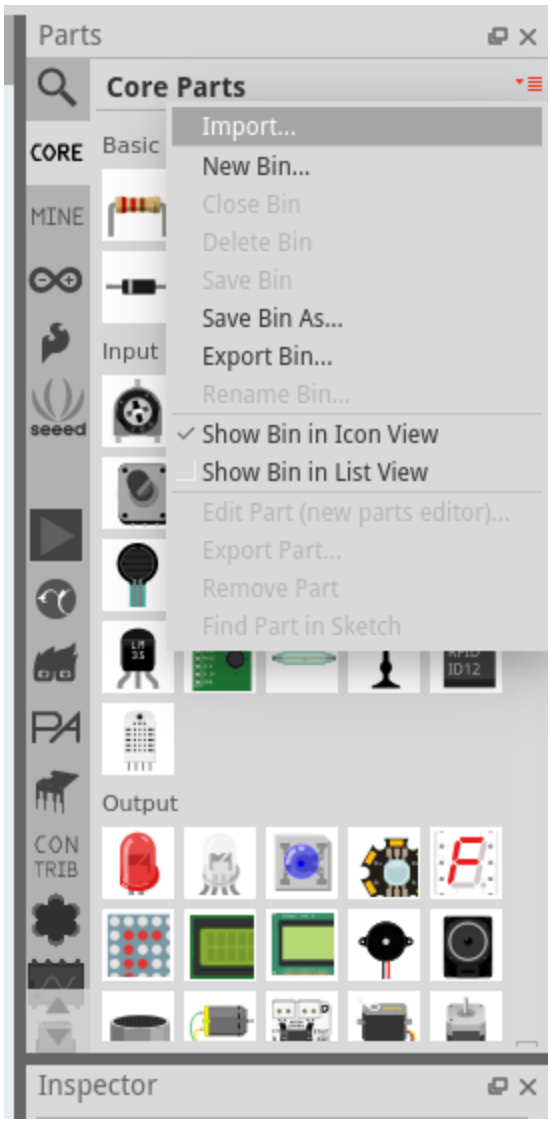
[7] “Printed Circuit Board Issues.” *Basic Linear Design*. Analog Devices, Inc. Web. May 2015.

Appendices

Designing and Wiring a Breadboard for the Bear

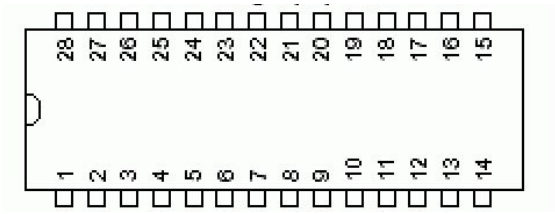
For each of the following steps, lay out the components on a breadboard and update an accompanying Fritzing diagram. The Fritzing diagram will later be used to easily lay out a schematic and design a printed circuit board (PCB).

1. Download the CPE 200 parts bin from PolyLearn and import it into your Fritzing environment.



Click the menu in the Parts Pane and click ***Import…*** *and navigate to the CPE 200 parts bin. You will be adding these parts to your Fritzing diagram as you follow these instructions.*

2. Remove the ATMega328p from the Arduino board (or use a separate one given to you) and place it across the gap at the center of the breadboard. It may be easier to have the microcontroller situated similar the image below, so you so left-to-right matches the numerical pin order. The notch represents the side on which pin 1 is located.

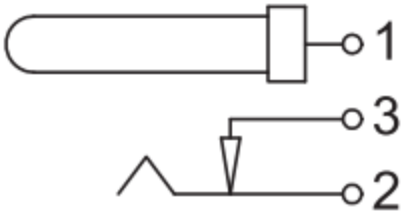


Fritzing part:

3. Connect the + and - rails of the opposite ends of the breadboard (red to red, blue to blue).
4. Configure the reset function for the ATMega chip
    a. **Place a 10kΩ resistor between pin 1 and +5V supply**. This is necessary because the reset pin (pin 1/RST) is *active low*, meaning it resets the chip if the voltage at the pin is ever set to 0V (logic low). The resistor keeps the pin at 5V (logic high) when it is not being deliberately driven by the microcontroller, and also allows for an external device to ground pin 1 (i.e. drive it low) when it needs to reset or program the chip.
    Fritzing part:

    b. **Connect a push button between pin 1 (RST) and ground (0 V)**. Make sure the two pins you choose are not electrically connected while the button is not being pressed. You can test this with the continuity test on a multimeter or by double checking the button's data sheet.
    Fritzing part:

5. Connect the DC barrel jack to the breadboard.
    a. Most DC barrel jacks have 3 pins:
        i. Power (+5V)
        ii. Ground (0V)
        iii. Switched jack



In this sample diagram, **Pin 1 is the +5V power connection**, **Pin 2 is the ground**

connection, and **Pin 3 is the switched jack**. When the wall supply is plugged into the barrel jack, the connection from pin 3 is broken (decoupled) from pin 2. This is useful when there is a battery circuit on a PCB that needs to be "unhooked" when a wall supply is plugged in. **For our purposes, connect pin 3 to the ground pin (pin 2).**

Fritzing part:

b. Connect the barrel jack's power pin to the positive rail for now (red line) and it's ground pin to the ground/blue rail.

6. Place two electrolytic capacitors (one 10 uF, one 1uF; in parallel) between pins 7 (Vcc/+5v) and 8 (ground) on the chip. This ensures that a relatively steady voltage is being supplied to the chip in the presence of noise from the servos. **Electrolytic capacitors are polarized, so make sure that the negative lead (signified by the stripe on the capacitor, or sometimes the shorter lead) is connected to GND (Pin 8).**

   Fritzing part:

7. Connect the supply and ground lines to the chip's relevant pins.
   a. Power (+5V) connections are pins 7 and 20.
   b. Ground connections are pins 8 and 22.
   c. Note: When connecting to pins 7 and 8, make sure the capacitors are closer to the chip than the wires you use (i.e. the wires come before or are below the capacitors).
8. Set up the external oscillator for the ATMega chip. This provides the clock signal for the microcontroller.

   Fritzing part:

   a. **Place a 16 MHz crystal** somewhere on the breadboard where columns are not already connected to something.
   b. **Place a 22pF ceramic capacitor between each crystal lead and ground (0V)**. These allow the crystal to resonate at 16 MHz when given a voltage by the chip.
   c. **Connect the two crystal leads to pins 9 and 10 (X1 and X2)**. It doesn't matter which side of the crystal is connected to which pin, since they are interchangeable.
9. Set up the power status LED. This tells us whether or not power is being supplied to the board (and therefore, the microcontroller).

a. **Connect a 220 Ω resistor between the 5V supply and an unused node on the breadboard.**
b. **Connect an LED between the resistor and ground (0 V)**. Be sure to connect the anode (longer lead) to the resistor and cathode (shorter lead) to ground.

10. **Place 10 screw headers in the breadboard**. These will be secure spots wires that connect to servos, sensors, and for general purpose I/O.
    Fritzing part:

11. **Connect the analog inputs to the first 5 screw headers.**
    a. Analog input pins are pins 23-27. These correspond with pins A0-A4 on the Arduino. Note: If you connect them left-to-right with the open part of the screw terminals facing you, it will be in a readable order on the final PCB.
12. **Connect PWM outputs to remaining 5 screw headers**.
    a. Our PWM outputs are pins 11 (D5), 12 (D6), 15 (D9), 16 (D10), and 17 (D11).
13. Place another 10 screw headers in the breadboard and **connect all 10 pins to the (-)/GND** rail. These will be the ground references for the servos. Note: The screw terminals require some *oomph* to place in the breadboard. There is one empty breadboard space between each terminal pin, so take that into account while wiring. You may place these pin headers directly on the negative/blue power rail to avoid extra wires and save breadboard real estate.
14. Place another 10 screw headers in the breadboard and **connect all 10 pins to the +5V output** from the DC barrel jack. Don't directly attach these to the 5V power rail in Fritzing, since we will be adding in a fuse before designing the PCB.
15. Place the remaining two electrolytic capacitors (one 10 uF, one 1uF; in parallel) between the +5V header (any of the pins work, whichever is closest to the ground header) and GND header. Make sure the (-) side of the capacitor is connected to ground.
16. On the +5V node of the capacitors, opposite of the headers, run a wire to the (+) line on the breadboard. This will provide a clean 5V.
17. Place a 3x1 female pin header so that they are connected to pins 1 (RST), 2 (RX), and 3 (TX). Although this is a superfluous step for the breadboard, it is necessary to do it in Fritzing so the PCB comes out correctly. This will allow us to program the ATmega chip using the Arduino.
    Fritzing part:

Parts legend, if you're unsure:

Crystal Oscillator

Electrolytic Capacitor

Ceramic Capacitor

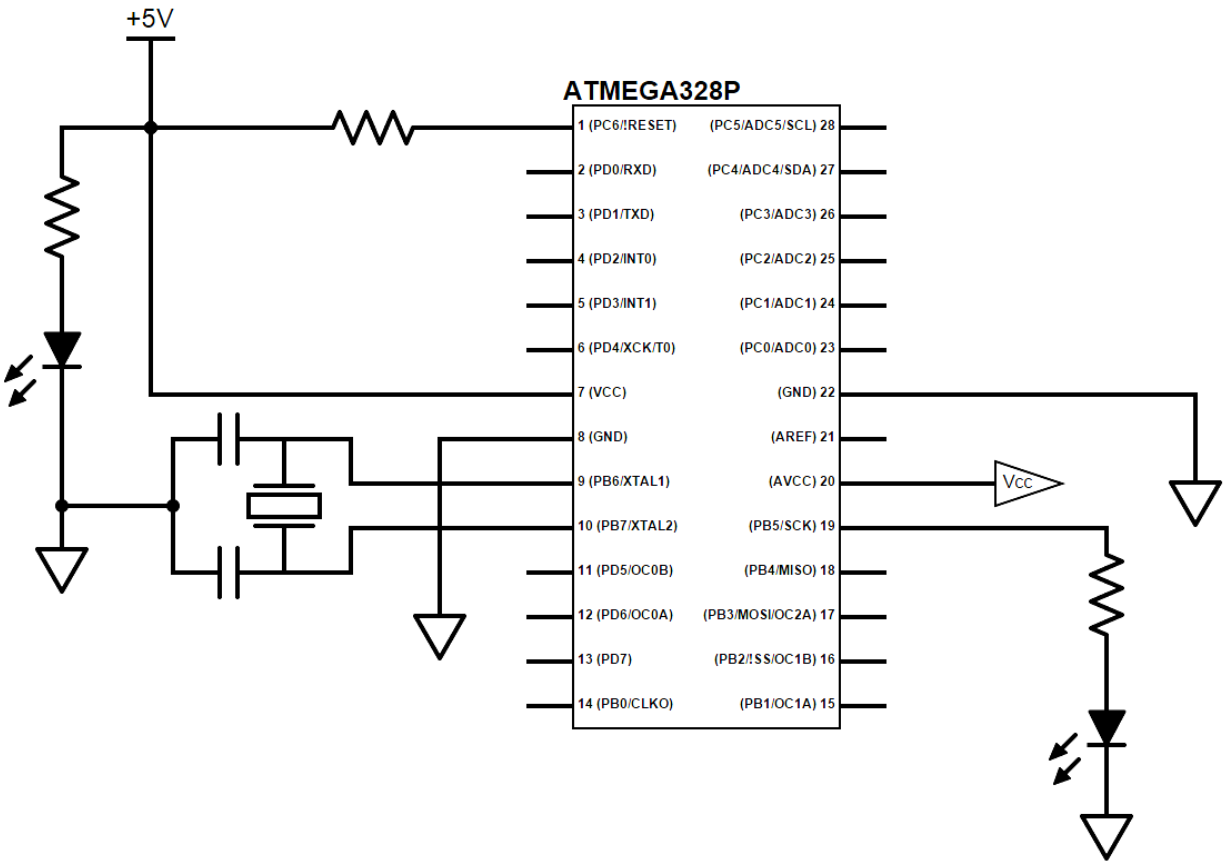
DC Barrel Jack

PCB Requirements

Hardware Requirements

To operate the ATMega328p on a PCB apart from the Arduino Uno board, a few simple needs should be met:

1. Connect an external 16 MHz crystal oscillator between external oscillator pins.
    a. **XTAL1/PB6 (9)** and **XTAL2/PB7 (10)** pins.
2. Connect a 20 pF capacitor between each crystal node and ground (2 caps total).
3. Connect ground pins to a common ground.
    a. **GND (8)** and **GND (22)** pins.
4. Connect the power pin of a 5V barrel jack to the relevant V_{CC} pins on the chip, and its ground pin(s) to the common ground.
    a. **VCC (7)** and **AVCC (20)** pins.
5. Connect the active-low reset pin to VCC, through a 10 kΩ resistor.
    a. **!RESET/PC6 (1)** pin.
6. Connect an LED (optional) to serial clock pin, cathode connected to ground (a 220 Ω current-limiting resistor may be necessary).
    a. **PB5/SCK (19)** pin
7. Connect an LED between V_{CC} and ground (with a 220 Ω resistor) (VCC -> R -> LED -> GND), so we can know when the chip is powered on.
8. Connect a push button between the RESET pin and ground for chip reset purposes.
    a. **!RESET/PC6 (1)** pin



On-Board Programming

The student has likely removed the ATMega from the Arduino Uno and soldered it to the PCB. If not, remove the microcontroller from the Arduino Uno board.

PCB requirements

The PCB needs a 3-pin header to "breakout" the TX, RX, and RST pins for uploading sketches

1. On the PCB, electrically connect one of the header slots to TX
    a. **PD1/TXD (3)**
2. Connect other header slot to RX
    a. **PD0/RXD (2)**
3. Connect third header slot to the reset pin
    a. RST (1)

Instructions

Connect TX on Arduino UNO to "broken-out" TX pin on PCB. Do the same for RX and RST.

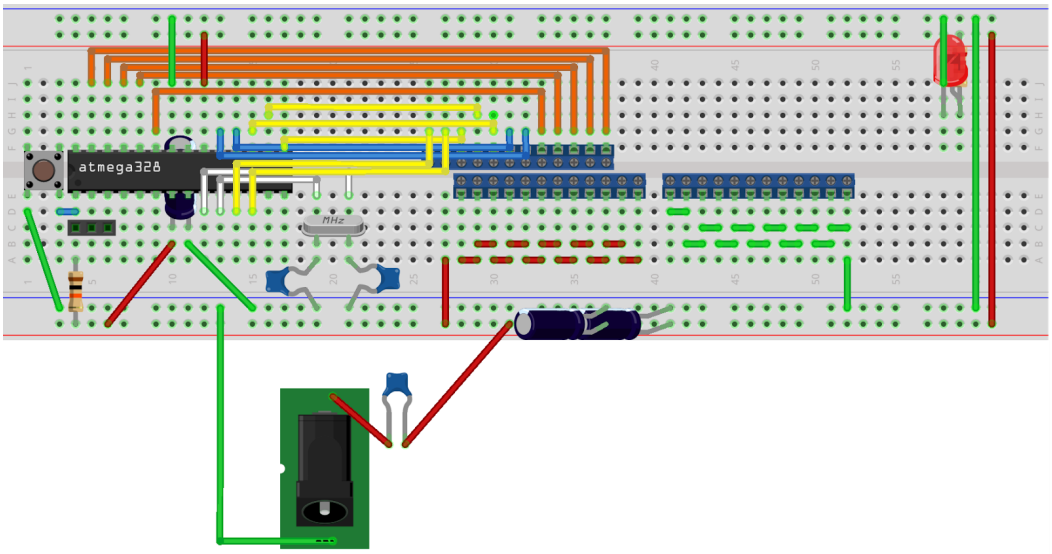
Connect GND pins from Arduino UNO to a ground header on the PCB. If you're not externally powering the PCB (which you likely are), then you may connect the 5V pin from the Arduino to a PCB power pin.

Source: http://arduino.cc/en/Tutorial/ArduinoISP
http://arduino.cc/en/Tutorial/ArduinoToBreadboard

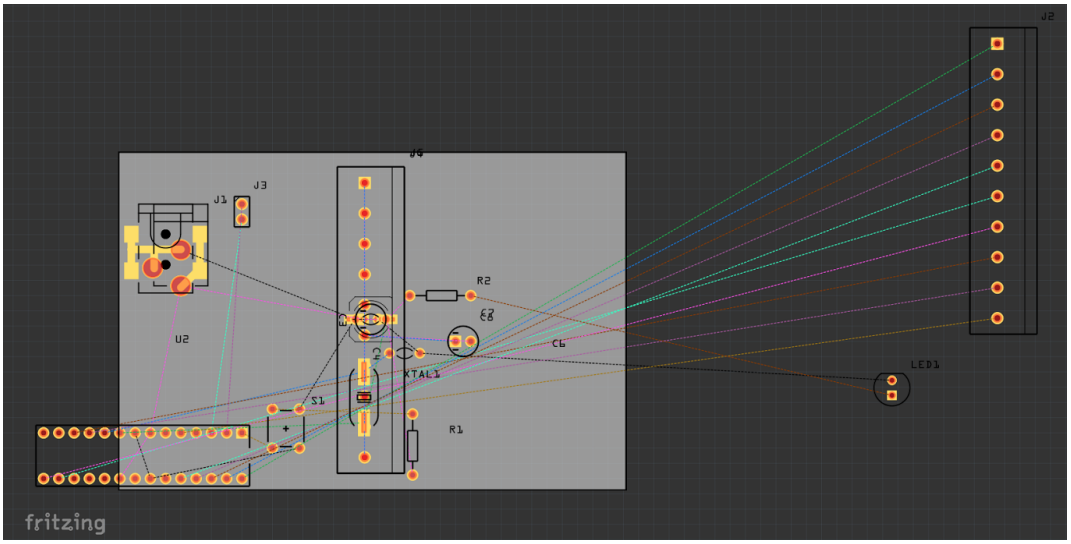
PCB Design Instructions

You've finished your Fritzing breadboard and now you're ready to design the PCB!

1. Before you begin, make sure you add a fuse to your breadboard design and wire it according to the following instructions.
    a. Grab one of the ceramic capacitors from the Parts bin, and change it's package to *"200 mill [THT, multilayer]"*. This will act as the fuse on your PCB.
    b. Connect the PWR pin on the barrel jack to one end of the fuse, and the opposite end to the entire +5V power rail on the breadboard design.
    c. Make sure the barrel jack GND and GNDbreak pins are shorted together and connected to the entire circuit's ground.
    Example:

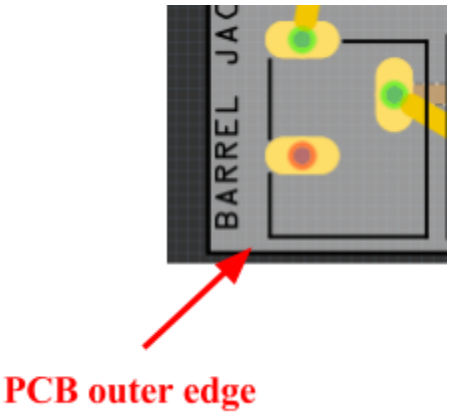


2. Verify component packages (if the components do not already have this package, change it in the "Inspector" window.)
    a. LED package: *"5 mm [THT]"*
    b. Electrolytic Capacitors: *"100 mil [THT, electrolytic]"*
    c. Ceramic Caps: *"100 mil [THT, multilayer]"*
    d. Fuse (will be a ceramic cap object in Fritzing): *"200 mil [THT, multilayer]"*
    e. Resistors: *"THT"*
    f. Oscillator: *"THT"*
    g. Barrel jack: *"Power jack slot"*
    h. Screw header pin spacing: *0.2 in*
3. **Go to the PCB View**
    a. You will see a large, gray rectangle and your component footprints scattered across the window, similar to the image below. The gray rectangle represents your PCB boundaries.



You can enlarge the PCB by dragging the corner of the gray rectangle.

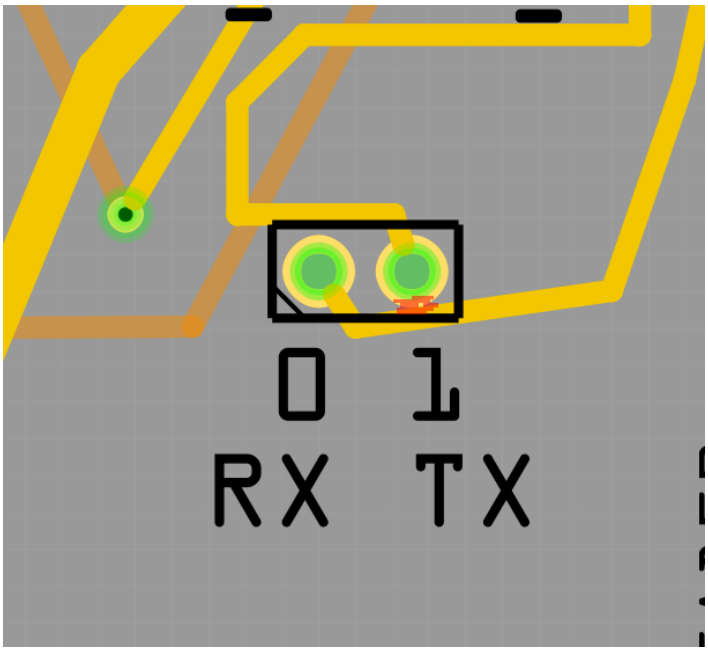
4. Place all of your components in any orientation you are comfortable with, you don't need to match the sample given to you. There are a few things you need to keep in mind:
    a. Make sure the barrel jack is facing an outer edge, in a configuration similar to this:



    The empty edge that has no exposed pads should be facing an outer edge.
    b. Place all other components wherever you please, minimizing the board size as much as possible.
5. **Edit Autorouter Settings (Routing > Autorouter/DRC settings)**
    a. Choose "Custom" (the below settings are usually default)
        i. Trace width: *"Standard (24 mil)"*
        ii. Keepout: *0.0100 in*
        iii. Via size: *"Standard (.4 mm)"*
        iv. Hole diameter: *.015748 in / .4 mm*
        v. Ring thickness: *.011811 in / .3 mm*
6. **Click "Autoroute"**
7. **Labeling:** Add silkscreen labels as you please to the board (the Fritzing part is called "logo"). Refer to the sample design passed out in class as an example.

a. Make sure to label all of your I/O terminals with their corresponding Arduino pin # (5,6,9,10,11,A0,A1,A2,A3,A4), as shown in the example given to you. You may choose to label PWM outputs with a tilde ~.
b. You'll also want to label the 3-pin female header with their functional names RST, TX, and RX. This will help when you are programming the chip after you have soldered it to the PCB.
c. Label your power jack and screw headers.
d. To make future assembly of your PCB easier, you should label each resistor and capacitor with their nominal values. Instead of "R1", for example, you can label it "10k". Label the crystal's resonant capacitors "22 pF" and the electrolytic capacitors "10 uF" and "1 uF." All other parts will be self-explanatory during assembly.
e. Add your name, class #, and quarter somewhere on the board.
e.g. "John Smith", "CPE 200 Spring '15"

8. Check for Design Rules violations. **Routing > Design Rules Check (DRC)**
   a. It is likely that autoroute was not perfect and there are a number of violations.
   b. Most of these violations should be overlap errors, meaning the autorouter put a trace too close to another component or another trace. Clicking on one of the errors highlights the part/wire on the PCB.



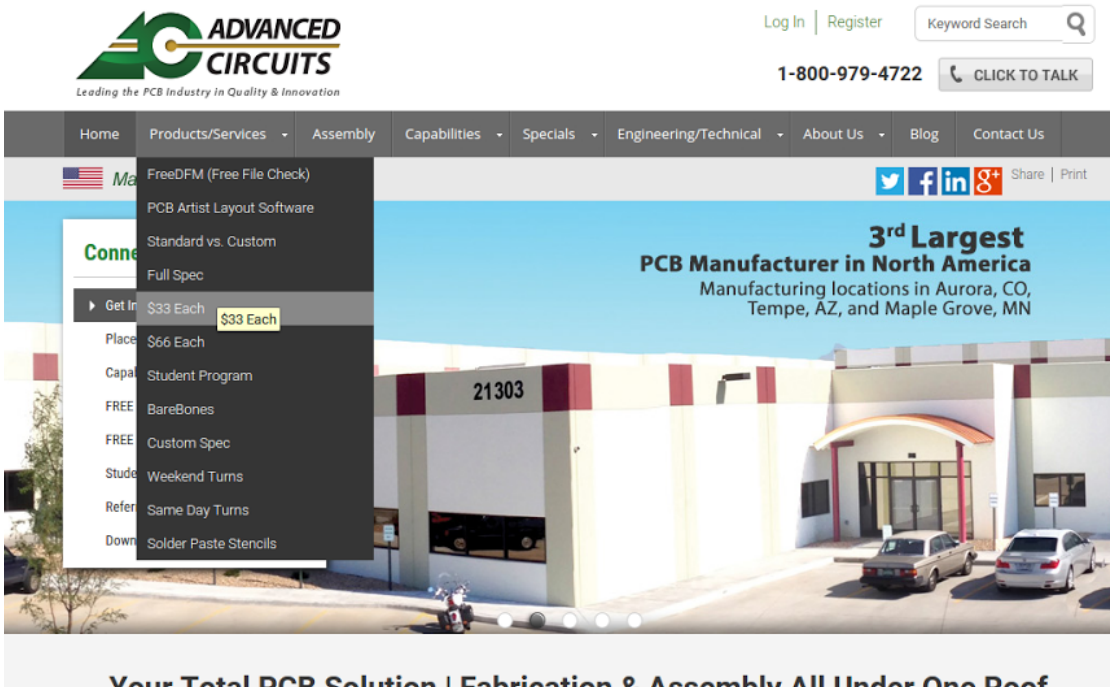
Find the red markings on the part/wire that caused the error and fix it. To fix something like this, just move the wire farther away from its neighboring part.

   c. Re-run DRC checks and fix errors until it reports that there are no violations.
9. Switch to the *bottom layer only*, using the Layers button on the bottom panel.
10. Go to **Routing > Ground Fill > Ground Fill (bottom).**
    a. This adds a ground plane that sinks current for the microcontroller and servos.
11. You're done! Get it checked by a TA and Export it as a Gerber file:
    a. File > Export > For Production > Extended Gerber

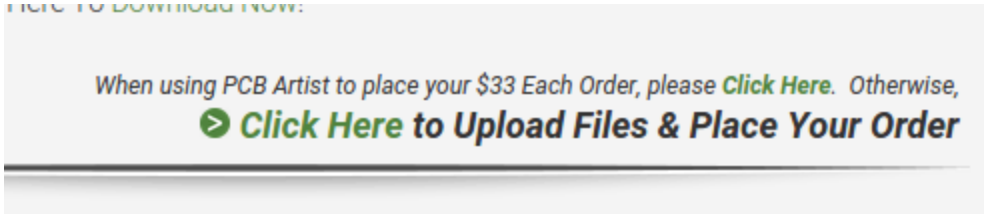
PCB Ordering Instructions

You're ready to order your PCB! Follow the instructions below:

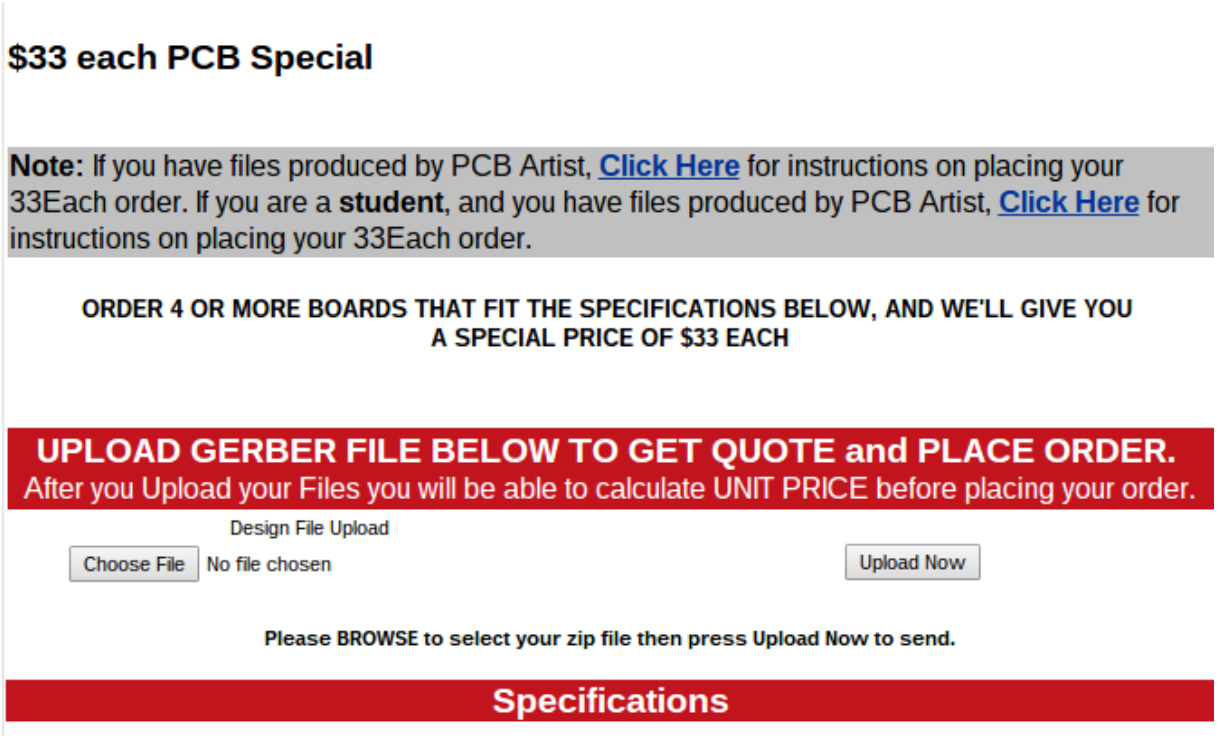
1. Go to www.4pcb.com (Advanced Circuits)
2. Click on the **$33 Each** link under the **Products/Services** drop down menu.



3. Blindly click where it tells you to (the bottom link, we are not using PCB Artist):

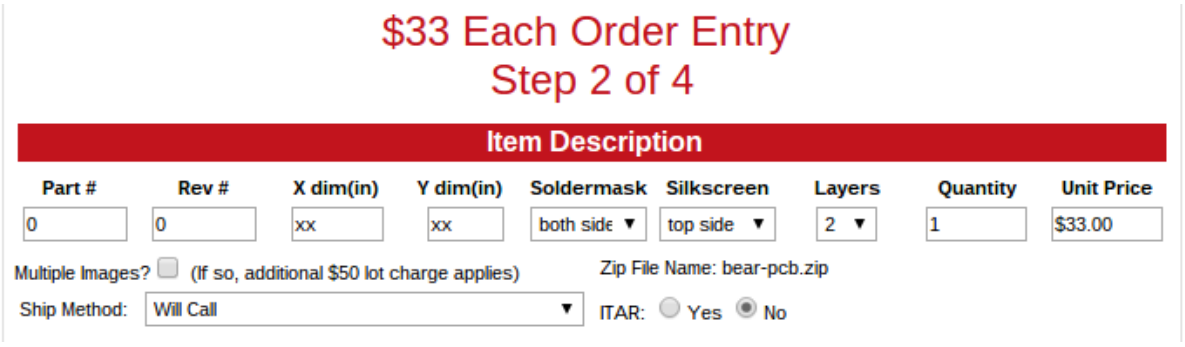


4. Create an account with 4pcb. If you already have one, log in with it.
5. You should end up at this screen after a successful log in:



Click **Choose File** and find your .zip file (You exported it from Fritzing, it's a zip of Gerber files).
Click **Upload Now**.

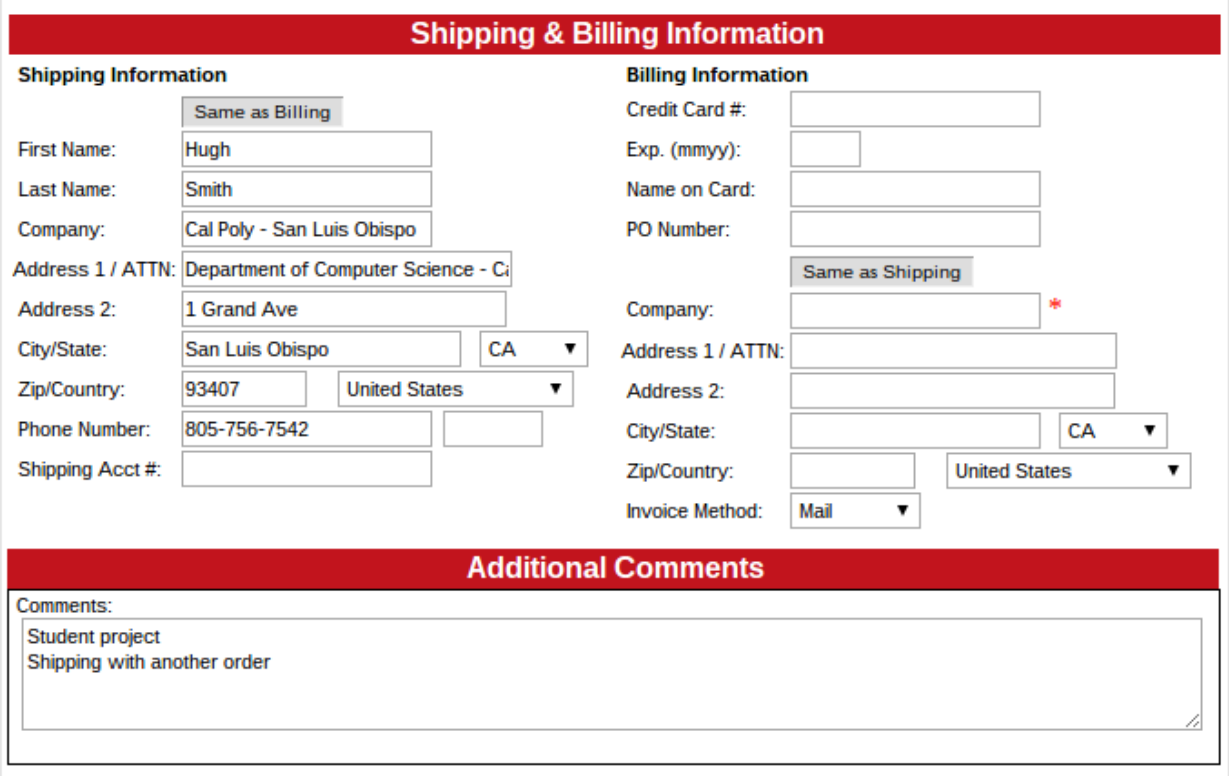
6. You will be shown this screen:



Enter the following information:
Part #: *0* (doesn't matter)
Rev #: *0* (doesn't matter)
X dim: the X dimension of your board, in inches
7. **Y dim:** the Y dimension of your board, in inches
Soldermask: *both sides*
Silkscreen: *both sides*
Quantity: *1*
Multiple images: *leave unchecked*

Ship Method: *Will Call*
ITAR: *No*

8. Below, you will need to enter billing and "shipping" information. The shipping information doesn't matter since you chose Will Call. For shipping information, enter the following:



*__For SHIPPING information:__
Name: *Hugh Smith*
Company: *Cal Poly - San Luis Obispo*
Address 1: Department of Computer Science - Cal Poly
Address 2: 1 Grand Ave
City/State: *San Luis Obispo, CA*
ZIP: 93407
Phone number: *805-756-7542*

*__Enter your billing information__

*** Additional Comments (IMPORTANT):**
"Student project, shipping with another order"

9. Continue with and complete the order.
10. Send your order/invoice # to prbrown@calpoly.edu by Wednesday night!

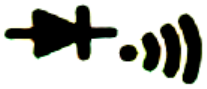
PCB Testing

Before you solder your parts to your PCB, you should test it for critical issues. Soldering parts on a broken or misdesigned board would be a waste of time and parts!

Continuity Testing

Most multimeters have a feature called "continuity testing," which allows you to see if two points in a circuit are electrically connected (i.e. shorted). Most have piezoelectric buzzers that beep when there is a continuity.

Find the continuity testing mode on your multimeter. The logo usually looks something like this:



A diode and/or a sound wave

If your multimeter doesn't have a continuity tester, simply switch it to its lowest nominal resistance measuring mode (having it act as an ohmmeter). Instead of beeping when there is a continuity, the meter will measure *very* small resistances, usually less than 10 ohms. When there is not a continuity, it will likely have a much larger resistance or not be able to display a resistance value at all.

NOTE: Continuity testing should only be performed when the board is *unpowered*.

1. Testing power connection continuities

We need to make sure power is getting delivered to the correct spots on the PCB. Open your PCB design and check to see if you connected the power line from your DC barrel jack to the power screw headers *before* or *through* the fuse. (i.e. did you feed barrel jack power directly to the screw terminals or not?).

A. If you connected them *before* the fuse, place your first multimeter lead on the DC power pin's PCB pad.
B. If you connected them *through* the fuse, place your first multimeter lead on the fuse PCB pad that is not connected to the DC jack (you can look at the traces or PCB design to determine which pad it is).

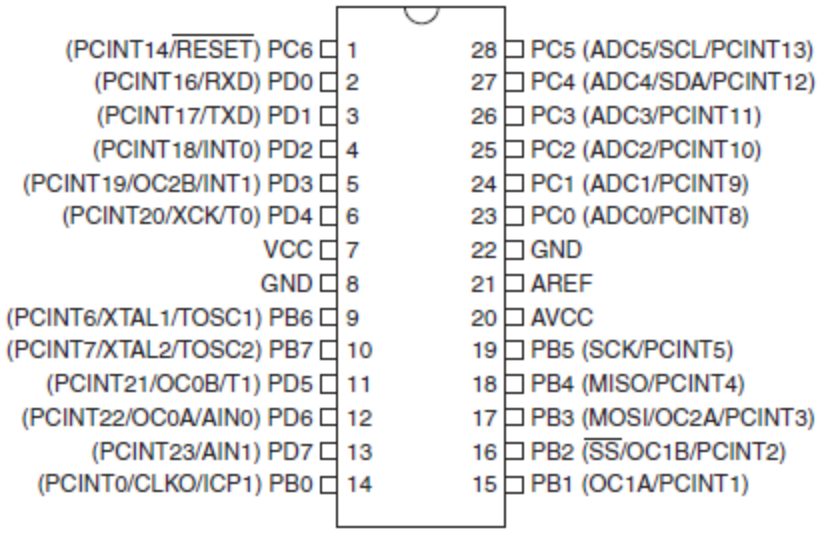
Continuity-test this pad with all of your power screw terminal pins. There should be a continuity. If there isn't, let the TA know and we can figure out what is wrong.

Follow up this test by moving your second lead to ground to ensure there are no shorts from power to ground. These types of shorts are very bad.

2. Testing chip continuities

If you followed step B above, skip this instruction (you're already here): place your first multimeter lead on the fuse PCB pad that is not connected to the DC jack (you can look at the traces or PCB design to determine which pad it is).

Continuity-test with ATmega pins 7 and 20. These are the VCC pins and are the locations at which the chip is supplied power. There should be a continuity. If there isn't, let the TA know and we can figure out what is wrong.



ATMega 328p pin layout

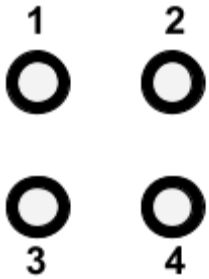
3. Testing ground continuities

Place your first multimeter lead on the DC barrel jack PCB pad. Continuity-test with all 12 ground headers.

Continuity-test with ATMega pins 8 and 22, keeping your first lead on the DC jack pad.

4. Testing for button shorts

Do a continuity-test on both sets of diagonal pads. Using the drawing below as a reference, you would test between 1 and 4 and between 2 and 3. **These should not be continuous (shouldn't beep).**



5. Testing pin continuities

Using the ATMega pin diagram, continuity test all your digital/analog pins from the screw terminal to their respective pin locations on the microcontroller PCB pads.

6. Testing oscillator continuities

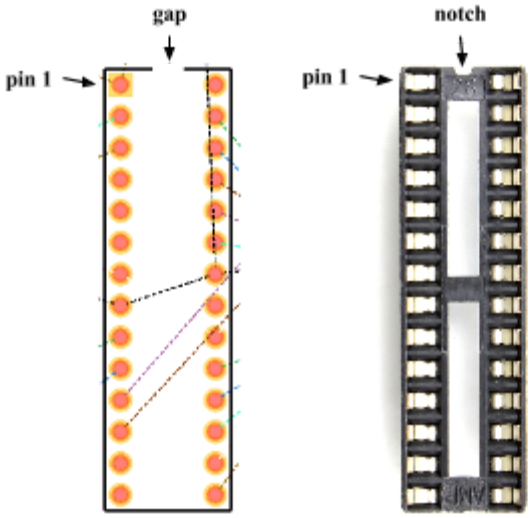
Place your first multimeter lead on one of the oscillator PCB pads. Make sure it is continuous with one of the XTAL pins on the Atmega328 (pins 9 or 10).

Then, make sure the other oscillator PCB pin is *only* continuous with the opposite pin (9 or 10, whichever you did not just find to be continuous with the other). In other words, make sure that the oscillator pins are both connected to the ATmega on different pins and not connected to each other.

PCB Assembly/Soldering Notes

Below is a legend and collection of notes so you can solder the correct parts to your PCB in the correct manner!

ATMega



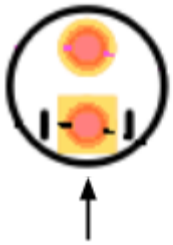
The left image above shows what your PCB silk screen will look like for the 28-pin DIP socket. There is a gap on one side of the rectangle, which signifies the side that has pins 1 and 28. Pin 1 is to the left of the gap.

The picture on the right is the DIP socket you will be soldering into the board. Make sure to align the notch in the socket with the gap in the silkscreen.

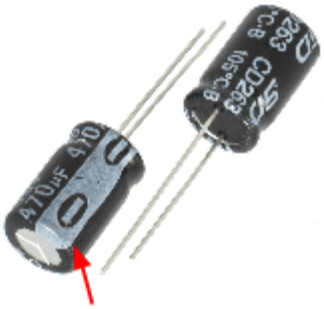
Ceramic Capacitors

The ceramic capacitor (shown on the right) fits into the PCB slot that looks like this. These are non-polarized, so you may solder them in either orientation.

Electrolytic Capacitors



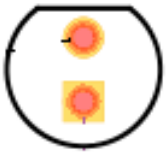
negative pad



negative stripe

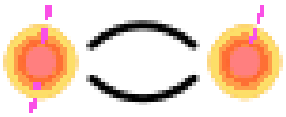
Electrolytic capacitors are polarized, so make sure you solder the negative stripe to the negative pad (marked with the two parallel lines, shown in the left picture).

LED



The LED is also a polarized device, so make sure you solder the negative lead (the shortest lead) to the pad marked with the flat side (the top pad in the picture above).

Fuse

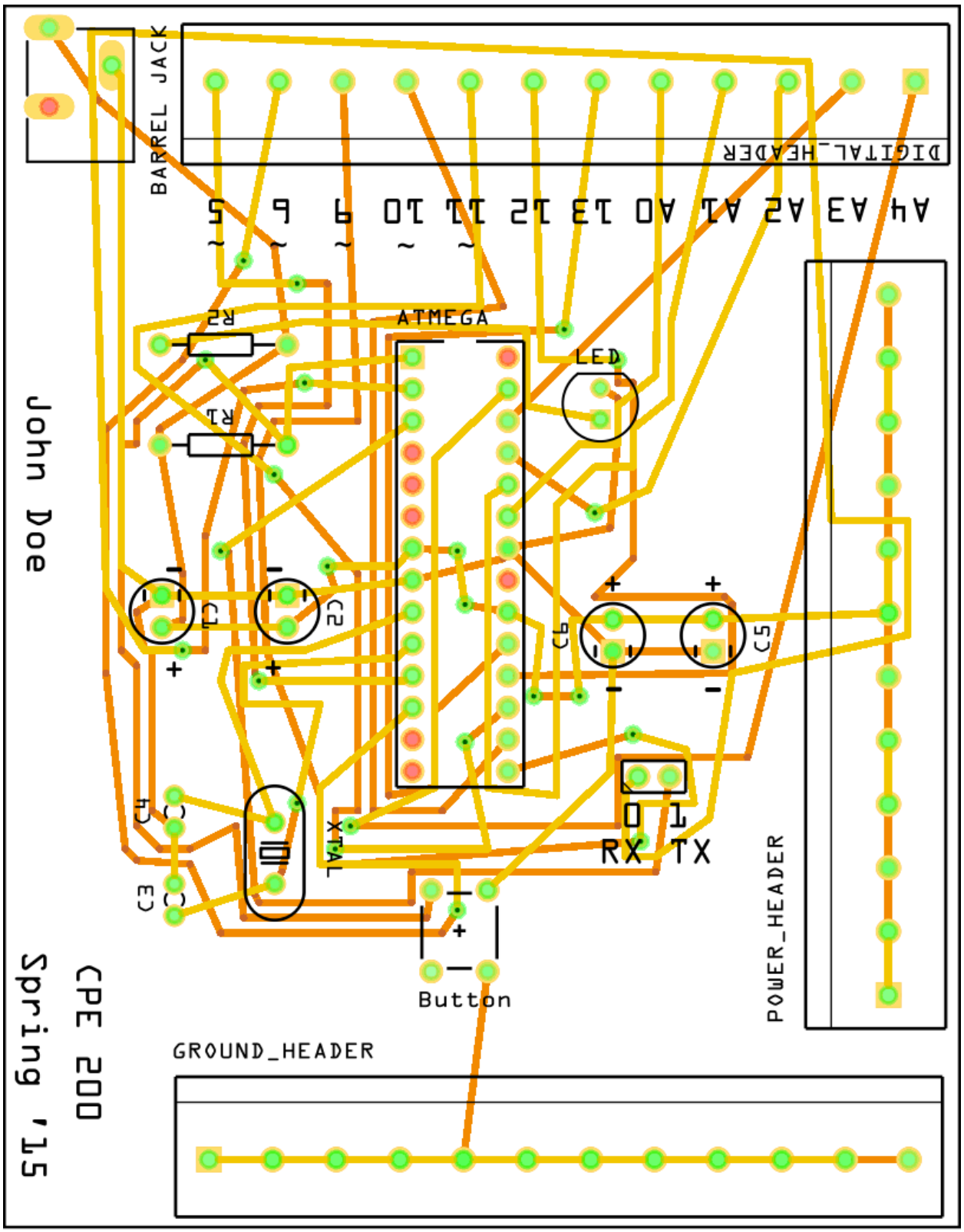


The fuse silk screen looks similar to the ceramic capacitor silk screen (because we used a placeholder capacitor in Fritzing). However, the width between the traces are 200 mill instead of 100 mill. The fuse is not polarized, so solder it in any orientation.

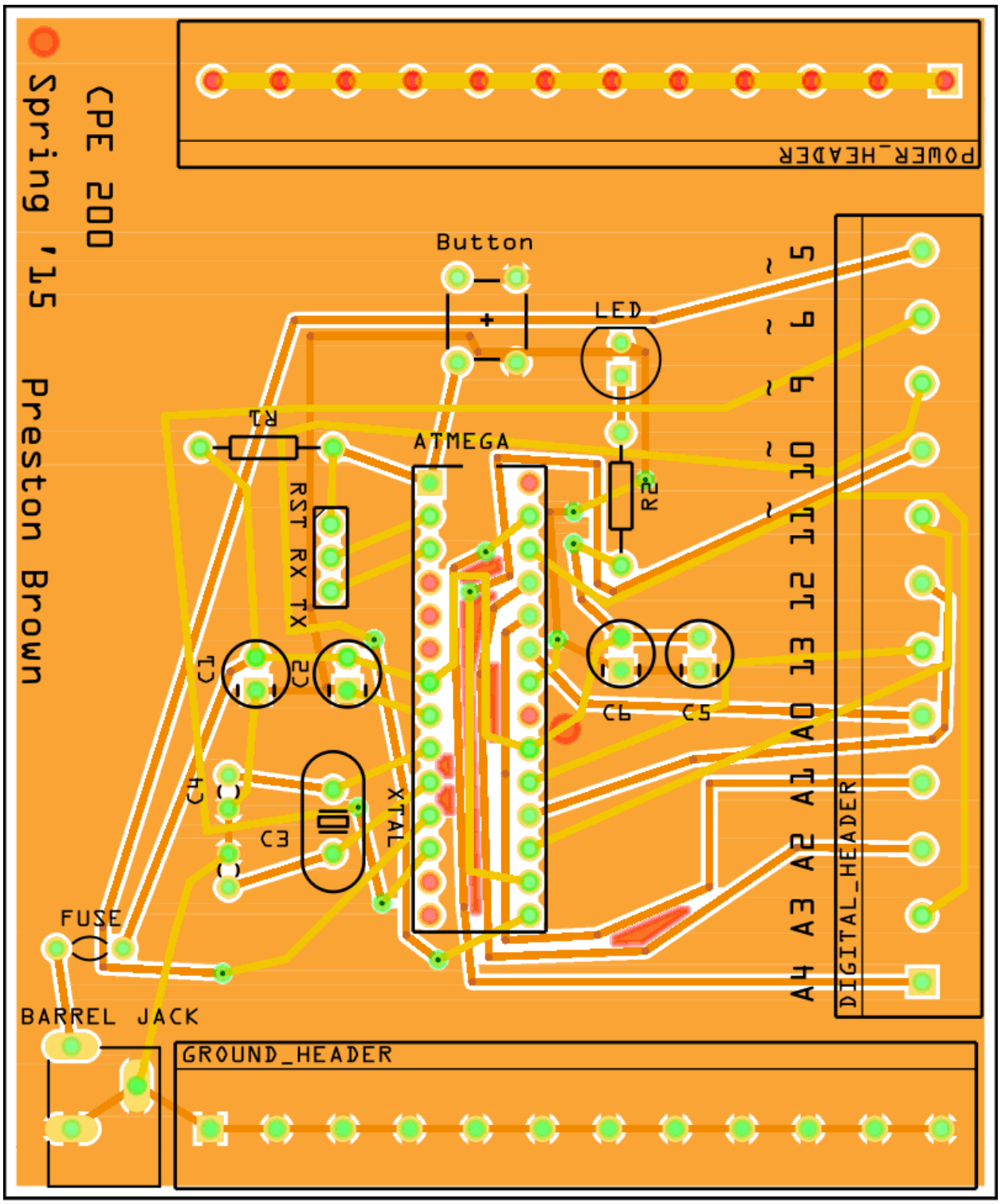
Everything else...

- Solder screw terminals to ensure that the actual terminal is facing the outside of the board.
- The button and DC jack can only fit in the holes in valid orientations, so no notes are needed.
- Resistors are also non-polarized and are not tricky.
- The crystal is also non-polarized.

PCB Design Revision 1



PCB Design Revision 2



Analysis of Senior Project Design

Summary

My project is an experimental design for the transition of the current CPE 200 Animated Bear class curriculum into one that includes basic printed circuit board (PCB) design and implementation for freshman students. The course currently uses the Arduino development board and a custom screw-terminal shield so students can connect servo motors and a power supply and create a fully-functioning animatronic stuffed animal. I was tasked with familiarizing myself with the Fritzing prototyping and PCB environment and creating a reference design and set of instructions and tips for future iterations of the course. My project fulfills these requirements and will help Dr. Smith transition the course, if he chooses, to a unique design course option for freshmen students interested in learning about electronics.

Constraints

There were a few limitations to keep in mind throughout the project which influenced my project choices knowing that the project will be used in a concurrent course. The two most influential constraints were time and cost. Developing and making changes to the breadboard prototype, PCB design, and accompanying instructions for students in the course required us to stay ahead of the students' progression. This required adapting to changes in the course schedule and developing instructions and designs prior to the students' design activities.

Since most costs of the class were funded through donations from HuSmith, Inc., keeping the cost burden as low as possible was another operating constraint. Electronic parts are generally inexpensive, but the costs add up when scaled for an entire class of students. The overall cost of implementing a breadboard PCB was relatively inexpensive, but PCB fabrication was the most expensive part. Despite the relatively small size of the students' PCBs, ordering eight would have added hundreds of dollars to the budget. Thus, the fabrication of student PCB design was an optional component of the course, and students who elected to order their design fronted the cost. The cost constraint added another constraint to time, ensuring that all students finished their PCB designs near the same time to get combined shipping and lower the per-board cost.

The project was further constrained in complexity; student activities and instructions were required to be easily understood by students with little previous electronics experience. The course is a 1-unit optional elective and is not intended to burden students with work outside of their major courses. Thus, they were provided with a code base and used auto-routing tools to cut down on time spent dedicated to learning the details of PCB design.

Economic

The main economic impacts of the project were student payments for their PCB fabrication and the donation budget from HuSmith, Inc. for the required project components. As seen on the attached bill of materials, initial cost estimates for development were $77.13. Including costs of seven additional designs, the expected the expected total was $617.04. Due to expedited shipping and processing fees, as well as slight disparities between expected and actual PCB fabrication costs, my individual project costs amounted to $127.13, while the seven additional designs amounted to $68.63 per project, totaling $607.54. The difference between expected and actual PCB costs stem from the choice of PCB manufacturer for my design; the chosen company charges according to board area, provides three copies of the design, and charges extra for 5-day fabrication. We ordered the student designs through a different manufacturer with a more cost-friendly deal.

I expected the project to take approximately one and a half quarters, since it was centered on a 10-week Spring quarter course. Expected development time for the breadboard prototype, which began in Winter quarter, was originally around one to two week. Although the prototype did not take long to put together, I spent much of the time developing instructions for students,, amounting to around six hours of work. Development and design of the PCBs and related instructions were expected to take the most time, likely around 6-7 weeks of Spring quarter. This estimate was on track; I completed my PCB development in the earlier half of the quarter while students built their Arduino-based bear. By week 6, I had a completed design which I began testing with servos during the class period. Students did not order their PCBs until Week 8 and received them during Week 10. The students designs took around a week for their initial edits, and I dedicated many hours to editing their designs before fabrication.

Environmental

PCB waste is a subset of the greater e-Waste problem that has arisen since the introduction of mass-production electronics. PCBs usually contain epoxy resin, fiberglass, copper, nickel, iron, aluminum, and sometimes precious metals such as gold and silver. Some estimates show that PCBs are 40% metals, 30% ceramics,and 30% plastics. When disposing of this project or any electronics, we must keep in mind that these compounds and chemicals that reside in a manufactured PCB may harm the environment if not disposed of properly. Luckily, there have been many developments and services in the world of PCB recycling, where companies can melt down forgotten boards and re-use the materials in manufacturing new electronics.

Manufacturability

The project will not likely be manufactured on a large scale or sold. The manufacturing costs for low-volume production, similar to our batch of student orders, would come out to around $35 per student design. Completion of the project requires hand assembly and soldering, with less than $20 of components. Manufacturing commitment would total around fifty dollars per project, plus assembly labor. The cost to use the device is minimal, as it draws very little power. There is extra cost associated with assembling the animated bear's PVC body and servos.

Sustainability

Expanding the PCB project may run into sustainability issues at scale. For the Spring quarter iteration of the course, we only had eight students. This allowed for a more leisurely-paced development schedule and allowed students to work at different speeds. This also kept the price of parts relatively low. Producing this project with more students in the future will require more instructional assistants and more time dedicated to individual PCB design, which would push back or eliminate other aspects of the course. Ultimately, it is up to Dr. Smith and other faculty on what direction to take class in. A possible upgrade to the design would involve switching to surface-mount components and reflow soldering for a more advanced look at electronics manufacturing; however, it is likely too time-consuming and difficult for a 1-unit introductory course.

Ethical

There were no pertinent ethical issues related to the development of this project. I was fortunate to have the opportunity to experiment with the future direction of an electronics course and work with younger students learn about basic PCB design; for most of the students, this project was their first ever fabricated PCB design.

Health and Safety

As touched upon in the environmental analysis section above, the relevant health and safety issues concerning this project are pollution created during PCB manufacturing and the eventual waste of the project, which may cause harmful chemicals and plastics to be introduced into the environment. It is strongly recommended that the PCB designs eventually be recycled with an electronics recycler so its materials may eventually be used again.

Social and Political

The PCBs were manufactured in Colorado by Advanced Circuits, Inc. Due to its American origin, we can likely be assured that the employees who run the manufacturing plant are paid fair wages and are working under humane working conditions. Most of the electronic components, however come from manufacturers based in Asia, which have a history of low wages and poor working conditions. Supporting companies like this may contribute to the social issue of a struggling working-class in these foreign areas.

Development

The project helped me gain experience in PCB design principles, instructional development, and familiarized me with the Fritzing development environment. Researching and designing a PCB for this project further exposed me to principles of PCB design, and helped me understand and account for many design considerations that present themselves during the creation of circuit boards. This included learning how to prevent electrical noise, avoid current loops, and ensure proper component function. It also gave me an opportunity to learn to express concepts I have learned in my coursework to students who are only just beginning their engineering education. Explaining concepts helps me internalize and explore ideas more compared to taking a course and passing a midterm.

Bill of Materials

Initial Estimations - Per Board

#	*Component*	*Supplier*	*Unit Cost*
1	16 MHz Crystal Oscillator ECS-160-20-4X	*Digi-Key*	**$0.81**
2	Vishay 20 pF Ceramic Capacitors	*Digi-Key*	**$0.31**
1	10 kΩ Resistor	*IEEE*	**$0.10**
1	220 Ω Resistor	*IEEE*	**$0.10**
1	LED	*IEEE*	**$0.50**
1	28 DIP Socket	*Sparkfun*	**$0.95**
1	DC Barrel Power Jack	*Sparkfun*	**$1.25**
11	3-pin Screw Terminals	*Sparkfun*	**$0.95**
1	Momentary push button	*Sparkfun*	**$0.35**
2	10 uF Electrolytic Capacitors	*IEEE*	**$0.50**
2	1 uF Electrolytic Capacitors	*IEEE*	**$0.50**
1	Breadboard	*IEEE*	**$6.00**
1	Arduino	*Sparkfun*	**$25.00**
1	PCB Design Fabrication	----	**~$30.00**
		TOTAL	**$77.13**

Final - Per Student Board

#	*Component*	*Supplier*	*Unit Cost*
1	16 MHz Crystal Oscillator ECS-160-20-4X	*Digi-Key*	**$0.81**
2	Vishay 20 pF Ceramic Capacitors	*Digi-Key*	**$0.31**
1	10 kΩ Resistor	*IEEE*	**$0.10**
1	220 Ω Resistor	*IEEE*	**$0.10**
1	LED	*IEEE*	**$0.50**
1	28 DIP Socket	*Sparkfun*	**$0.95**
1	DC Barrel Power Jack	*Sparkfun*	**$1.25**
11	3-pin Screw Terminals	*Sparkfun*	**$0.95**
1	Momentary push button	*Sparkfun*	**$0.35**
2	10 uF Electrolytic Capacitors	*IEEE*	**$0.50**
2	1 uF Electrolytic Capacitors	*IEEE*	**$0.50**
1	Breadboard	*IEEE*	**$6.00**
1	Arduino	-----	**$10.00**
1	PCB Design Fabrication	*Advanced Circuits*	**$33.00**
1	Split Shipping Cost	*UPS*	**$3.50**
		TOTAL	**$68.63**

Final - My Board

#	*Component*	*Supplier*	*Unit Cost*
1	16 MHz Crystal Oscillator ECS-160-20-4X	*Digi-Key*	**$0.81**
2	Vishay 20 pF Ceramic Capacitors	*Digi-Key*	**$0.31**
1	10 kΩ Resistor	*IEEE*	**$0.10**
1	220 Ω Resistor	*IEEE*	**$0.10**
1	LED	*IEEE*	**$0.50**
1	28 DIP Socket	*Sparkfun*	**$0.95**
1	DC Barrel Power Jack	*Sparkfun*	**$1.25**
11	3-pin Screw Terminals	*Sparkfun*	**$0.95**
1	Momentary push button	*Sparkfun*	**$0.35**
2	10 uF Electrolytic Capacitors	*IEEE*	**$0.50**
2	1 uF Electrolytic Capacitors	*IEEE*	**$0.50**
1	Breadboard	*IEEE*	**------**[1]
1	Arduino	*Sparkfun*	**------**[1]
3	PCB Design Fabrication	*OSHpark*	**$54.00**
1	Expedited Processing	*OSHpark*	**$89.00**
		TOTAL	**$127.13**

[1] *Indicates item previously owned.*