Electronic Load Controller Prototype with a Microcontroller

ENGN0930A Spring 2022 - Professor Haas, Brown University
Joshua Phelps

Outline

1 Why build an Electronic Load Controller (ELC)?

To produce constant power of the right frequency and voltage with a synchronous generator, the electrical load must be kept fairly constant. An Electronic Load Controller is a device that adjusts how much power it draws in order to keep the generator’s frequency steady. An ELC can be built more cheaply and respond more quickly than a system that furls a wind vane or turns a valve on a hydro system, and the energy “wasted” by the ELC can be used to heat water or run a pump. There are already many designs of ELCs, but the cost can be reduced by using a microcontroller instead of complex circuitry, and it is easier to improve code than rebuild a complex circuit.

2 Existing ELC Designs

- Humming Bird by Jan Portegijs: http://microhydropower.net/mhp_group/portegijs/humbird/humb_main.html
  - http://www.aidg.org/documents/ELC_Modifications_Experiences.pdf
    - 3 individual hummingbird ELCs on a 3 phase generator were causing issues by conflicting with each other and causing “feedback and noise issues”
      - maybe software controlled ELCs could be networked together to coordinate themselves
    - circuit boards can be assembled with clothes irons!
    - units should be assigned serial numbers
    - “An ELC that used pulse width modulation (PWM) with IGBTs or MOSFETs could provide a power factor corrected load which would dramatically reduce the noise and loading on the generators.”
    - used 240VAC generators
  - one phase 10kW (3 dump load version).
  - $125 of parts
  - “It uses phase angle regulation, but with 2 (or 3) small dump loads instead of one large one. Phase angles for these dump loads are regulated such that if one dump load has a phase angle of 90° (the worst case), the other one is either completely on or completely off. This makes that distortion of generator current is much less than with one large dump load and there is no need to choose a 25 % oversized generator (see type 1 above).”
  - PI controller is made from electronic components instead of software!
  - Protection Features
    - Generator overload (ELC turned off all dump loads and generator frequency still dropping
      - hummingbird generates pulses in the grid to notify users
        - aidg review of hummingbird found that people weren’t responding to it (partially since the grid was unreliable already), and removed the feature
        - while it doesn’t add much additional load, it is adding additional load when the grid is already struggling.
    - Voltage spikes, lightning, and high voltage protection
      - see hummingbird document section 3.8 for some discussion, a comment in 4.1 (top right of that page) seems to suggest there isn’t much protection for that in the humming bird
    - humming bird 4.1
      - more discussion in section D
      - overspeed

        - unsure how it responds (if it can't increase its load, because it is either out of loads to turn on or maybe because it is broken)
        - Some kind of alert would likely be a good idea, or turning off the water if that's possible.
    - overvoltage
        - what's it do?
        - I see mentions of a relay. Does the ELC cut off all power from the generator to the grid?
        - A schematic in the aidg hummingbird review also shows generator brakes
        - At some point the ELC might need to protect itself, but then it's really important that the generator can prevent overspeed by itself
    - undervoltage
    - fast undervoltage (self protection)
        - protects "the relay" from rattling if a voltage falls so low that the relay can hardly be powered
    - overheat (self protection)

- https://ludens.cl/Electron/picelc/picelc.html
    - PIC microcontroller
    - triacs, multiple dump loads
    - code is quite short, but confusing and somewhat specialized to a specific person's house.
    - Spreads load somewhat evenly across loads
    - detects "zero crossing" of power, and switches after full cycles.
- https://www.sciencedirect.com/science/article/pii/S2215098619318804
    - this paper says that loads should be turned on when the voltage is at zero for less noisy power, and they made a prototype with an Arduino Mega
    - not instructions for people wanting to build an ELC. I don't see any Arduino code
- https://www.homemade-circuits.com/simple-electronic-load-controller-elc/
    - describes multiple loads that switch on and off, with ICs, (though it seems not in binary)
    - describes PWM'd single load (with 555 timer)
    - describes triac dimmer style
- https://usermanual.wiki/m/01f04d3d2ae761a71fc13a04010d2257aa25ea429f97890da3733eb0f82ca68d.pdf
    - seems to be a class project
    - arduino (atmega) control system, rectifier and mosfets
    - Discussion of binary resistors vs PWM, and the possibility of combining both methods
    - discussion of relays vs various solid state switches, and they chose MOSFETS

- PI controller
- The Arduino code in their appendix doesn't look like it uses binary for controlling the resistors
- total cost $46
- did some testing, say they plan on installing it somewhere for real

- http://jasper.sikken.nl/electronicload/index.html
  - arduino nano based "programmable load"
  - doesn't use relays, uses one mosfet
- https://electronoobs.com/eng_arduino_tut123.php
  - mosfet, fairly low power
- https://eprints.untirta.ac.id/10075/1/Paper%20JITEKI.pdf
  - triac
  - arduino
  - frequency detection
  - a prototype, not an instruction manual
  - discusses zero-crossing detector
- https://ujcontent.uj.ac.za/vital/access/services/Download/uj:29561/SOURCE1
  - discusses theories for a few designs
  - prototyped and tested an ELC that uses a rectifier, filtering capacitor, transistor, and Arduino
- https://www.homepower.com/_files/ugd/d49ff9_7f80f1a47e7d47f3826616bd1f447bec.pdf
  - HomePower magazine#33
  - page 14 mentions Coarse and Fine control
  - also discusses prioritizing loads
  - this article is a description of a relatively large system using a commercially manufactured "Electronic Load Control Governor (ELCG)" from Thomson and Howe Energy Systems.

3 Alternative ideas (other than 8 relay module)

- Triacs are cheaper than relays,
  - $0.32 https://www.alibaba.com/product-detail/transistor-bta30A-triac-800V_60796544375.html
  - They are used in light dimmers, so they could let partial power through (though according to humming bird that causes problems by distorting the current wave)

- Unlike relays they don't isolate the microcontroller electrically, so it would probably be a good idea to use optoisolators.

3.1 Load control comparison:

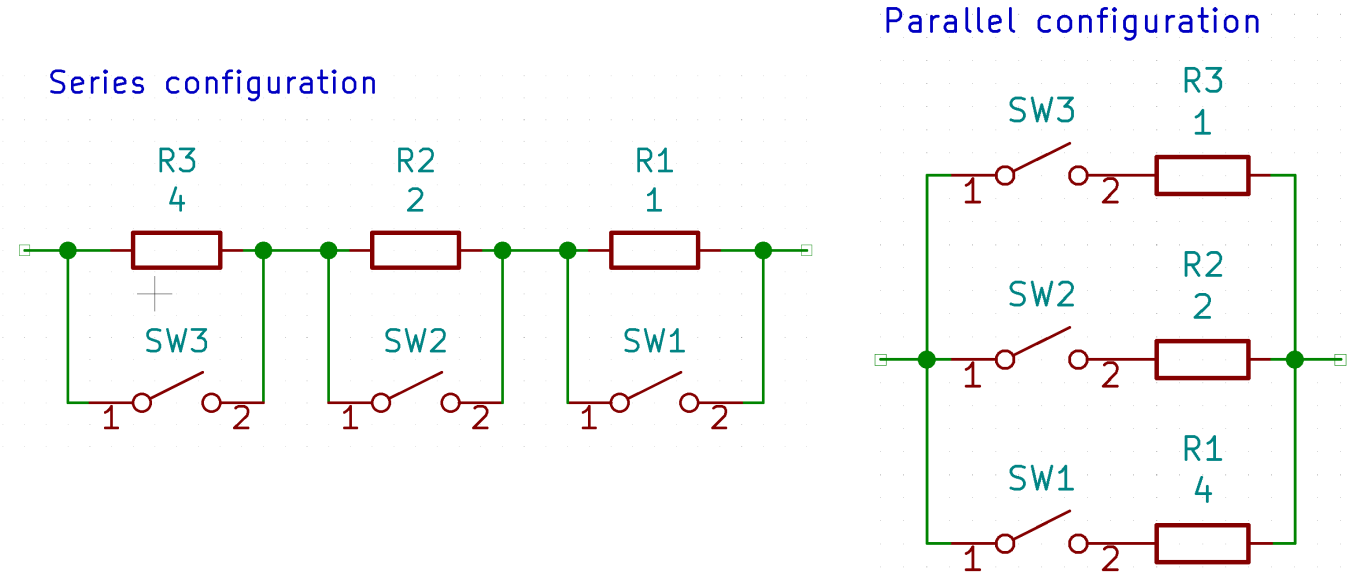
- Plain binary switching
  - types of switches
    - relays
    - rectifier and mosfet
    - solid state relay
    - triac
  - pros:
    - when not changing value, has just a perfect resistive load
  - cons:
    - could switch at any point in the AC wave, and causes a big voltage spike each time it switches
- zero-crossing binary switching (turns on and off switches when the AC voltage is near zero)
  - types of switches
    - triac
    - solid state relay
    - rectifier and mosfet
  - pros:
    - less electrical "noise"
  - cons:
    - can only switch once or maybe twice per period of AC wave
    - more complex
- pwm
  - switches
    - rectifier and mosfet
    - solid state relays that can be PWMed
  - pros:
    - lots of resolution from just one component
  - cons:
    - might need to filter out "noise" from PWM
    - might be expensive to get a component that can PWM AC power
- zero-crossing dimmer (classic triac control)
  - switches
    - triac
  - pros:
    - lots of resolution from just one component
  - cons:
    - adds "noise" to power, and might mess up power factor?

4 Series or Parallel Resistor Configuration

There are two main ways to wire load resistors and relays to create a variable resistor.

Illustrations here are done with three relays for simplicity. A real system might have 7 or 8.

(schematics made with KiCad)



The following table shows what resistances each configuration would have if the switches counted in binary:

Binary relays (3,2,1)	Series Ω	Parallel Ω
000	7	∞
001	6	4
010	5	2
011	4	1.3333
100	3	1
101	2	0.8
110	1	0.6667
111	0	0.5714

To avoid the danger of the series circuit shorting itself, an additional resistor could be added to the end of the series chain (shifting the range upwards, for example to 1-8 ohms).

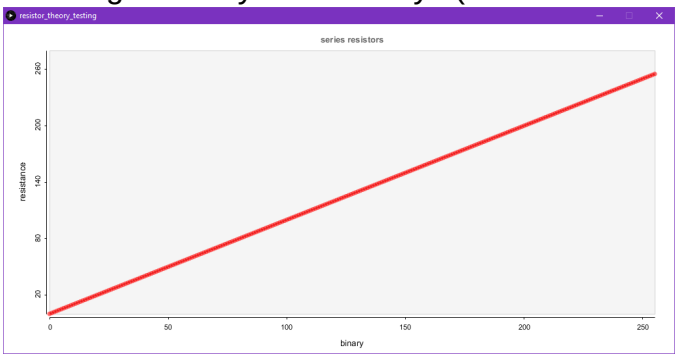
Since the goal is to create an adjustable load and vary how much power is drawn, the resistance doesn't really matter, what matters is the current.

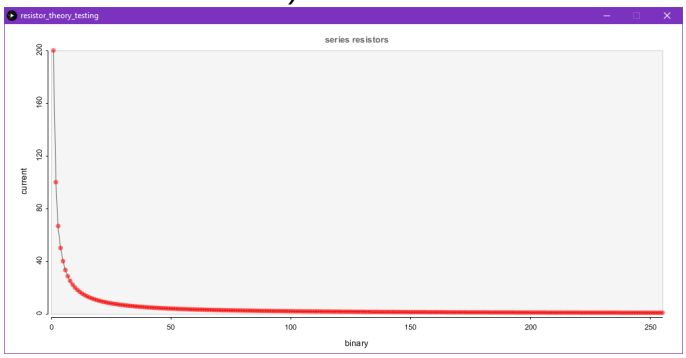
$$V = IR \rightarrow I = \frac{V}{R} \qquad P = IV \rightarrow P = \frac{V^2}{R}$$

Current and power are inversely proportional to the resistance.

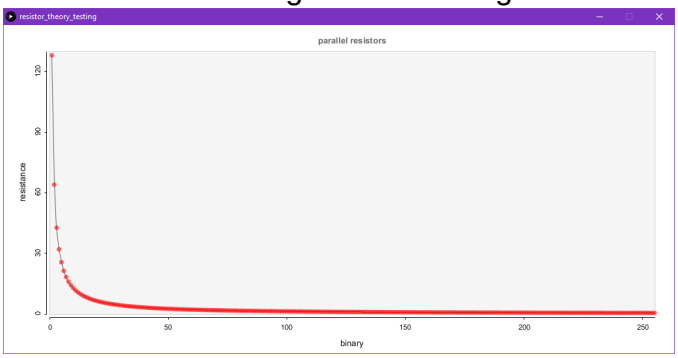
To visualize the relationships, I wrote a small program with Processing and the Graphica plotting library to simulate the resistance and current drawn by series and parallel configurations. I wanted to see whether a series or parallel configuration would make it easier to vary how much

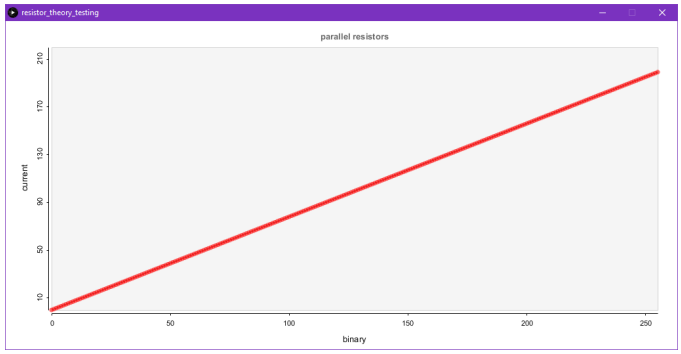
current is drawn. It uses the equations for adding resistors in series and parallel and simulates counting in binary on the relays (like shown in the schematics above).





The series configuration has a linear resistance vs binary graph, but an inverse current relationship. This wouldn't be so good at drawing various amounts of current.





The parallel configuration has an inverse linear resistance graph but a linear current relationship. This is perfect for drawing various amounts of current.

4.1 Series vs Parallel

For the purpose of drawing a variable load (or power), a parallel configuration of resistors is better since the inputs create a linearly distributed amount of power drawn.

Another advantage of the parallel configuration is that in order to reduce the resistance and increase the power draw, more resistors tend to be activated, as opposed to in the series configuration where at the highest power draw only a single resistor is left taking all of the load.

The maximum current limit of the parallel configuration, ignoring the resistors and assuming that the relays are the limiting factor, is almost twice the current limit of a single relay. 1111=15 is about twice 1000=8, or this could be thought of as a geometric series with r=0.5, since only the lowest resistance relay can draw it's maximum current, with each higher resistance relay only drawing half the current of the one before it since the resistances double.

One solution to enable even higher currents would be to use two relays for the highest current/lowest resistance resistor, in effect leaving 7 separate relays. The system could now draw almost 4 times the current limit of a single relay, and if the system has 8 relays, it would still have 128 different values.

All of this assumes that the voltage is known, and relatively constant. When calculating what values the resistors should be, a safety factor should be added. It's better to have higher resistance values and have a slightly lower max load than burn out the relays.

5 Load resistors

The resistors for the system will need to be

- high current
  - over 10 amps, since that's the limit of the relays
- cheap
- precise
  - If the resistors have incorrect values, instead of a straight linear relationship there could be jagged zigzags. This would make it much more difficult to program a good control loop, though in theory there could be a calibration and then the software could compensate for it by "sorting" the relay settings by their actual resistances.
- varied resistances
  - The lowest value resistor would be 128 times smaller than the highest value resistor for a system with 8 resistors.

I think lengths of wire are probably the best option. The wire gauge should be enough to handle the current, and the length can be changed to adjust the resistance. The lowest resistance value should be chosen based on how much current the relay can handle (and how much capacity the ELC needs to have) and then each resistor should be twice the value of the one before it.

For example, if the generator makes 120V, use R=V/I, and the 10A limit of the relays to find that 12 ohms is the minimum resistance. (but a safety factor should be added, and I don't know if the relays are rated with peak or rms values). Usually wire gauges are chosen to keep the wires from getting hot and wasting power, but in this case the wires should get hot, just not so hot that they burn their insulation off. If the resistors are in a tank of water to be a water heater that would probably help keep the wires down to a safe temperature.

The lowest value resistors that take the highest currents need to be made of thick wire to handle the current, but that means they would have a low resistance per length, so the wires might need to be quite long. Wire for the load resistors might be a majority of the cost of the system. Maybe actual heating elements could be better, but they would have to be the perfect resistance or have a length of wire added to trim out the value. If wires get too hot, their resistances would go up. This would make the ELC not able to pull as much load and could make the control loop not work as well (PID tuning would be off), but since higher temperature would decrease the load it could be considered somewhat self-stabilizing.

In theory, the ELC could open the relays if the voltage gets to a damaging level, but it would be even safer to put fuses in series with the resistors and relays. Or it might be better to damage the relatively inexpensive relays and load resistors than stop protecting the generator.

This video shows a technique for measuring lengths of wires for use as current measuring resistors, and a similar technique could work for this project. One important note was that the wire should be folded in half before it is coiled to avoid inductance effects. This method uses a lab style current limiting power supply and is what I used to make the resistors for my first prototype (my minimum resistor is 2 ohms). For a real system, the voltage from the generator would likely be higher, and the higher resistances can likely be measured with just a good multimeter.

Since 8 powers of two cover a large range, different gauges of wire should probably be used in order to keep all the wires to a manageable length.

6 Relays

Boards with 8 relays and driver circuitry can be found for under $10 on Amazon and for under $5 on Alibaba.
I doubt much if any money could be saved by buying individual relays and driver circuitry, and it would make assembly a lot more difficult.

6.1 Relays aren't instant

Since mechanical relays have to physically move an internal switch, changing between binary numbers won't be instant. While a relay is switching, it's not connected, so whenever the relays move there is a short spike (~15ms) of higher resistance. When a more significant bit changes, there could be a significant spike in the resistance.

6.1.1 Solutions

If the time that it takes for the relays to switch is consistent, it should be possible to start moving the relays that are closing a little bit before moving the relays that are opening. This could reduce the length of the spike.

6.2 Lifetime of mechanical relays

This datasheet says the relays should last 10^5 cycles (a commonly cited minimum life expectancy for relays at full load).

If a relay switches once per minute, it would reach 10^5 cycles in a little over two months.
(10^5 /365/24/60 = .19)

Unless the system is very stable by itself and only needs occasional load adjustments, cheap mechanical relays might not be good enough.

The datasheet also says it can last 10^7 cycles mechanically if there's very little electrical load. Relays with higher resistance resistors are loaded less and switch more frequently, and maybe the load can be spread across a couple relays, since if we could get closer to 10^7 cycles that would be more acceptable.

Solid state chips like triacs would be better and cheaper, and be more flexible (PWM is possible for fine adjustments).

7 Testing the relay board

Boards with 8 relays can be bought on amazon for under $10,
https://www.amazon.com/dp/B00DR9SE4A
Similar boards can be bought on AliExpress for under $4,
https://www.aliexpress.com/item/32649659086.html

7.1 Control pins

The 8 input pins each activate a relay when pulled low.

When Vcc (photocoupler power) is 5 volts, they source 0.88 mA.
When Vcc is 3.3 volts, the relays can still be activated.
When the input pins are disconnected, the relays are not activated.
If Vcc is 5v, the input pins float at 2.5V, so there's no danger to a 3.3v microcontroller

7.2 Relay power

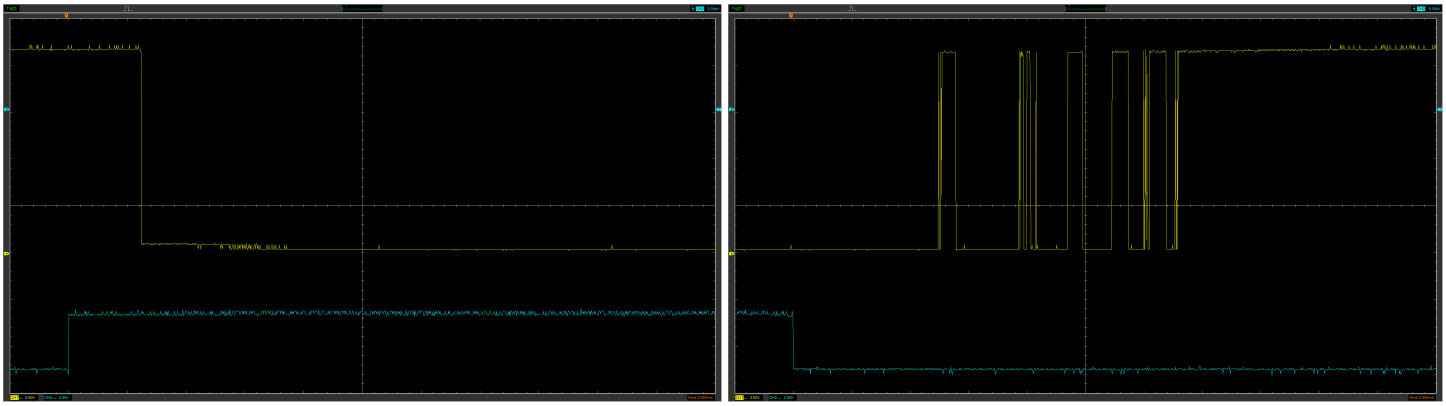
JDVcc (relay power) needs to be 5 volts. 3.3v doesn't make the relay move.

I measured the current draw of the board while powering JDVcc and Vcc with 5 volts.
Powering no relays draws zero current (as well as I can measure it).
Powering one relay draws 62 mA
Powering two relays draws 120 mA
Activating all 8 relays draws 421mA
I should account for half an Amp of power for the relay board.

7.3 Relay speed

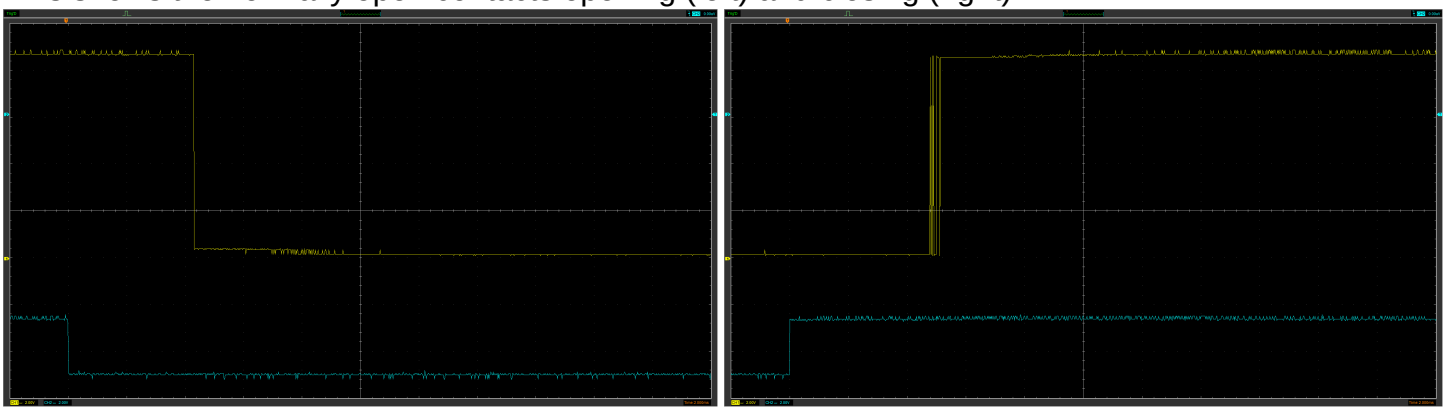
I connected a 9V battery, the relay's switch contacts, and a 10K resistor in series, and measured the voltage across the resistor with an oscilloscope.
The lower blue line shows whether the relay is activated.
The upper yellow line shows whether the contacts are connected.

This shows the normally closed contacts opening (left) and closing (right):



About 2.2ms after the relay is triggered, the contacts open. It takes 5ms for the contacts to close, and 14ms to stabilize.

This shows the normally open contacts opening (left) and closing (right):



About 4.2ms after the relay is released, the contacts open. It takes 4.4ms for the contacts to close and 4.6ms to stabilize.

All of these tests were done with just one relay. These numbers are probably similar for all of the relays, but more trials would be needed to get proper data. It took awhile to get pictures with an oscilloscope, so if I wanted to do a full experiment I would want to control and measure the relay with a microcontroller to help automate the process.

This all makes sense because of how a relay works. The switch opens before it closes, since the switch takes some time to move between the two contacts. The relay also moves faster when it is powered than when it is relaxed and a spring resets the switch. The switch also bounces against the contacts longer when the relay isn't powered and forcing the contacts together.

Mainly because of the long switch bounce time with the normally closed contacts, I plan on using the normally open contacts (pulling a control pin low and energizing the relay will cause the contacts to close). (Note: *I ended up wiring the resistors through the normally closed terminals by accident on my prototype. I don't think it made a significant difference to my tests. My program has a setting so that either way of wiring it can be used.)*

8 Choosing a Microcontroller

8.1 Requirements

- Cheap
- Not too difficult to program
- Capable of running the control loop fast enough
- Capable of controlling enough outputs

8.2 Options

- Arduino Nano
  - $2.00 https://www.alibaba.com/product-detail/nano-V3-0-ATMEGA328P-Development-board_60112187852.html
  - pros:
    - I've used Nanos before
    - simple
  - cons:
    - less powerful processor
- ESP32
  - $2.00 https://www.alibaba.com/product-detail/NodeMcu-CH340-CP2102-V3-V2-Lua_62104656489.html
  - pros:
    - I've used ESP32s before and like them
    - very powerful processor
    - wifi opens interesting possibilities (and potential security issues)
  - cons:
    - they're more complex and I don't understand them as well as Nanos. I've had issues with my code crashing
    - some of its pins pulse as it boots so be careful (though there are probably plenty that are fine)
- ESP8266

    - $1.00 https://www.alibaba.com/product-detail/The-CH340C-NodeMCU-Lua-WiFi-development_1600407612630.html
    - pros:
        - powerful processor
        - wifi opens interesting possibilities (and potential security issues)
    - cons:
        - I have very little experience with these
        - most of its pins pulse as it boots https://rabbithole.wwwdotorg.org/2017/03/28/esp8266-gpio.html and there aren't as many pins
- ATMega328 by itself (the processor in an Uno or Nano)
    - $1.00 https://www.alibaba.com/product-detail/Microcontroller-ic-ATMEGA328-atmega328p-pu-ATMEGA328P_60664652302.html
    - pros:
        - can be programmed like an Arduino (but so can everything else here) https://docs.arduino.cc/built-in-examples/arduino-isp/ArduinoToBreadboard
        - cheap
    - cons:
        - hardest to use

8.3 Decision

I chose an ESP32 microcontroller since it has more processing power and is cheaper than a classic Arduino, and because it has more pins and I have more experience with it compared to an esp8266. I will still use the Arduino.h library, so the software could probably be transferred to a different microcontroller quite easily.

9 First Prototype

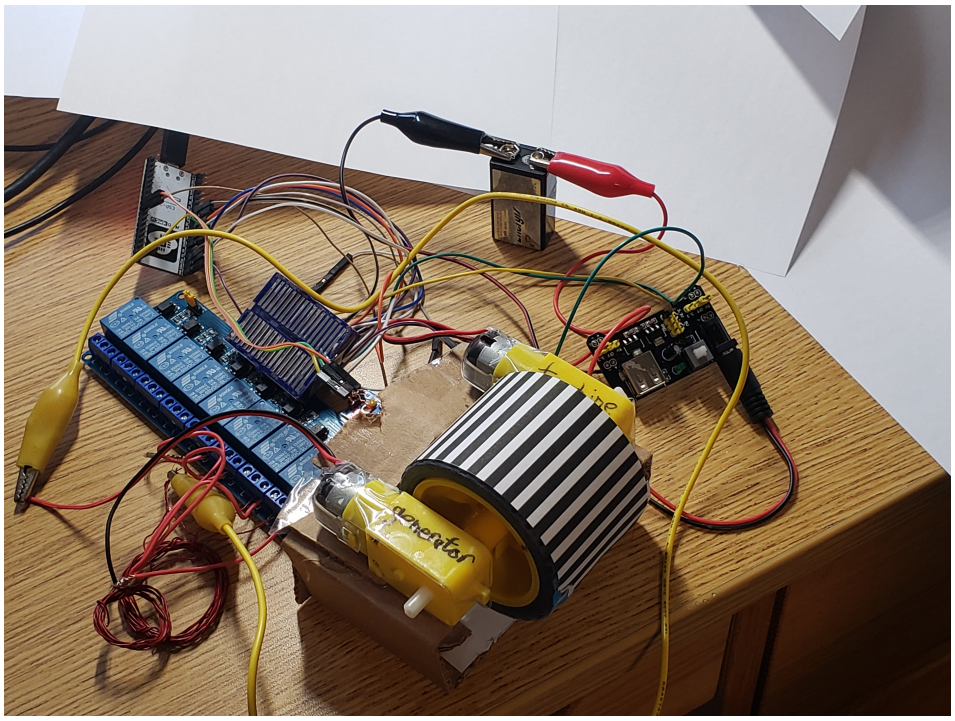
9.1 Design

The first design I wanted to try was an 8 relay board that could turn resistors on and off in a "binary" pattern.

For cost and safety reasons I started with a small scale prototype. I connected the shafts of two small geared dc motors together, powered one motor, and connected the other through my prototype ELC. Drawing current from a dc motor that is being spun increases the force needed to spin the motor.

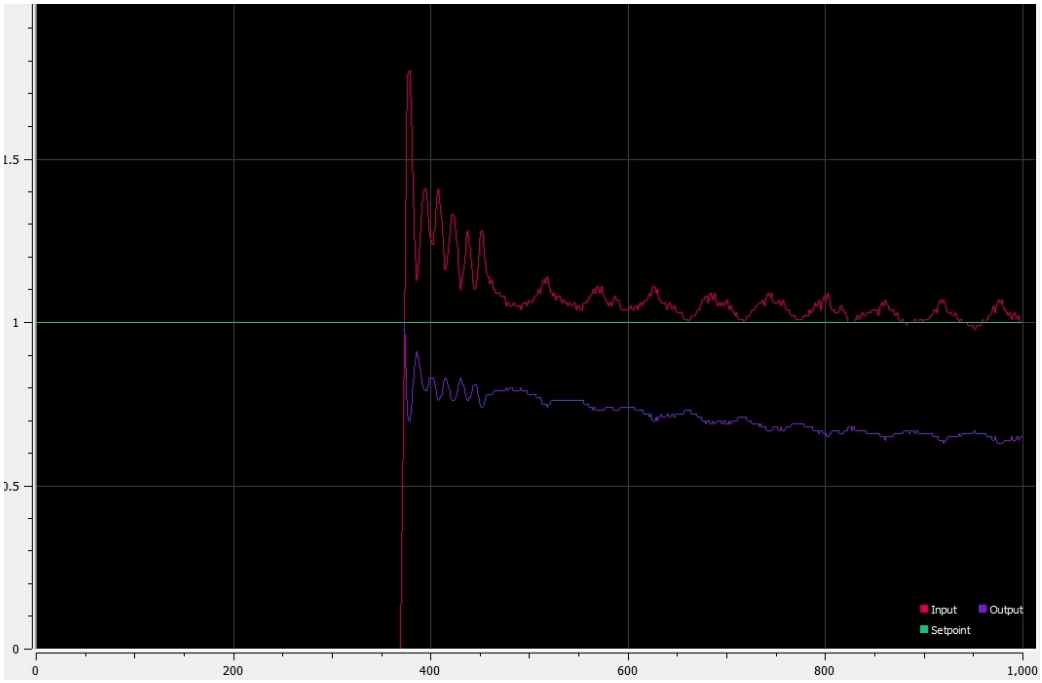
I used an IR proximity sensor and a striped pattern printed on paper wrapped around wheels on the shafts to make a tachometer to measure the rotation rate of the shafts. The goal was to keep that rate steady with a PID control loop.

The primary question I wanted to answer was whether a microcontroller can flip relays fast enough to keep the speed steady. I was also able to check the code I had written so far.



For my prototype I only connected 5 of the 8 relays. I used values between 2 and 32 ohms, and measured the resistance of the motor at 5 ohms.

9.2 Results



After a few seconds of settling time (each gridline is about 5 seconds), the speed was kept to within 10% of the setpoint. The remaining oscillations might be solvable with more resistors and/or better PID tuning, or maybe a generator with a bit more inertia than a tiny geared motor is needed.
I also noticed that when the motor spins freely without any control loop, the measured speed oscillates by itself; I think there is uneven friction in the motor and I could hear that the sound of the

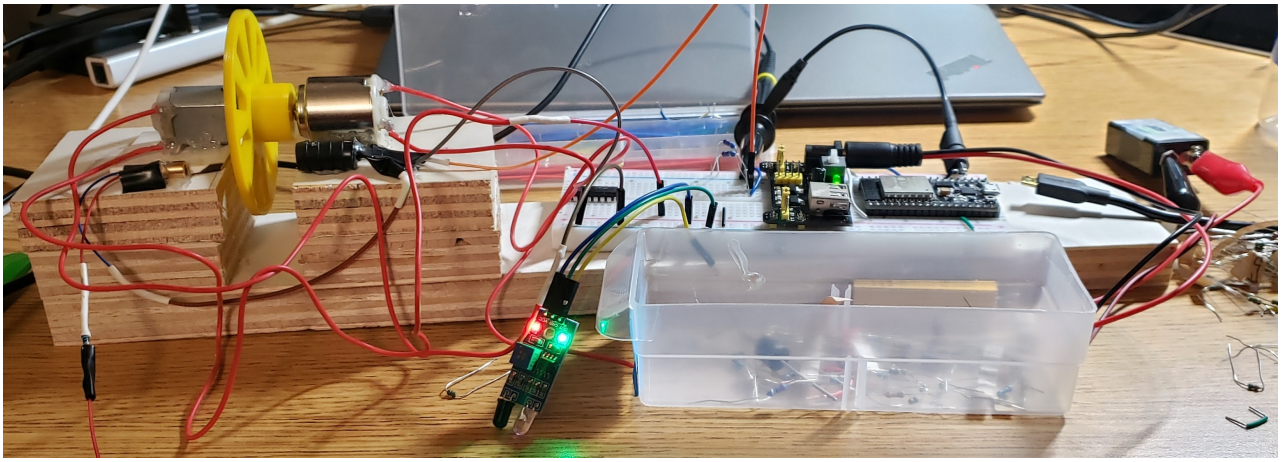
motor wasn't quite steady. While the control loop couldn't control the oscillation, it wasn't the main cause of it, and this test was able to get the average frequency fairly close to the setpoint.

9.3 Misc

The motors have an internal resistance of 5 Ohms.

Relays were flipped a few times a second. While the gearmotor is likely less stable than a real generator, if relays had to switch as quickly in a real installation, the relays would break quite quickly. The relays could be broken in just a couple days if flipping once a second! (10^5 /24/60/60 = 1.16)

10 The issues with my second prototype



10.1 Changes for the second prototype

- Direct drive dc motors
- Beam break sensor and 3d printed wheel instead of reflectance based tachometer

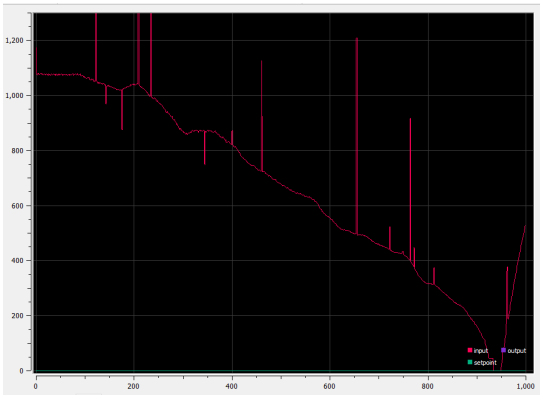
10.2 Problems

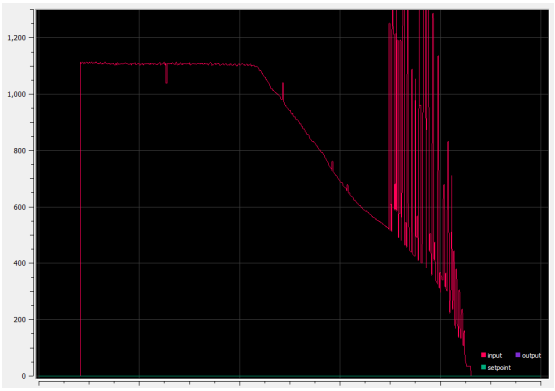
10.2.1 Motor issues

- One of the motors seemed broken – it was just really weak and shorting it had no effect on the speed of the other motor.

10.2.2 Tachometer issues

- First, I tried using a laser (as an excuse to play with a laser diode) and light sensitive resistor as a beam break, but after testing it I learned that LSRs are a bit slow to respond to changes in light. The wheel had 10 slits in it and when the motor spun at 3000 RPM, the amplitude of the output signal got too small for the ESP32 to read the change with an interrupt. I tried using an opamp to fix this, but the opamp I was using couldn't output a voltage within a volt or two of its supply voltages and I didn't have a proper power supply for it.
- Then, I modified an IR proximity sensor (just like the sensor on the first prototype) by lengthening the wires to the LED so it could reach around and shine through the wheel, but the output wasn't a clean signal and gave really bad velocity measurements. I tried filtering the signal with capacitors and while I was able to improve the signal some, there were still enough false velocity spikes that an aggressively tuned PID loop would have trouble. I considered trying to filter the velocity in software (by tossing the highest and lowest points in an interval and averaging the rest), but that wouldn't have been so relevant to the goal of building an ELC.





These are some graphs of measured velocity. The errors in velocity measurement would have caused issues with the PID loop.

I went back to using my first prototype before even trying the second prototype with the relays.

11 Improved original prototype

11.1 Changes

11.1.1 Better Encoder

I still wasn't happy with the quality of the velocity measurements from the original striped wheel and IR sensor so I switched to using an AS5048b absolute angle magnetic encoder.

11.1.2 Varied "turbine" power

To better simulate changes in wind or water pressure, or changes in electrical load from the rest of the grid, I attached the "turbine" motor (the one turning the "generator" motor) to an adjustable power supply. This let me test how well the prototype could control a dynamic system.

11.1.3 Coarse and Fine Control

See the following section.

I was still only using 5 relays since making the load resistors took awhile. Adding the fine adjust really helped since only 32 steps isn't that many.

11.2 Results

A video of a test of the prototype can be watched here: https://vimeo.com/708770644

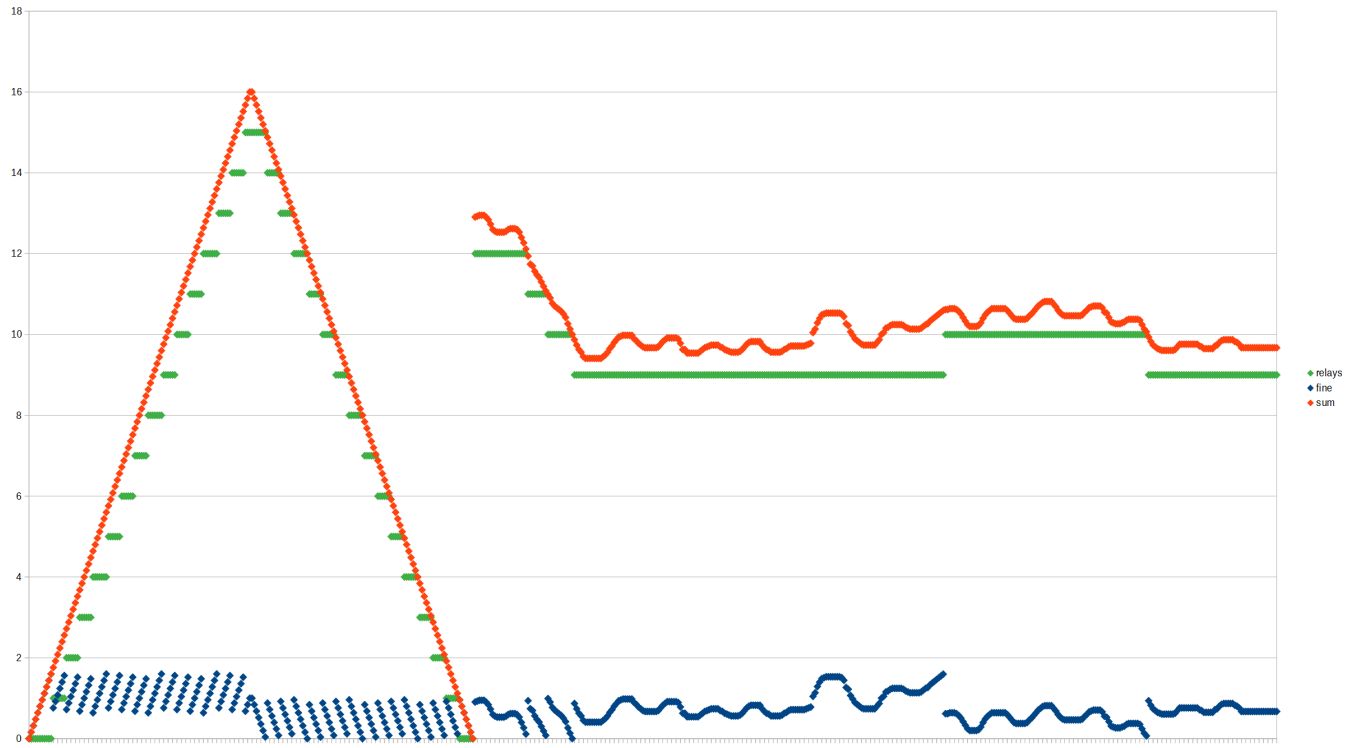
I didn't collect actual data on the performance of the prototype since it's so different from a real generator system. The error fluctuated by about ±5% when I wasn't changing the power to the motor, but over twice that when I was adjusting the power supply. I think that a small geared motor is more challenging to control than a proper generator that would have more inertia. I also think that better PID control loop tuning could increase that farther.

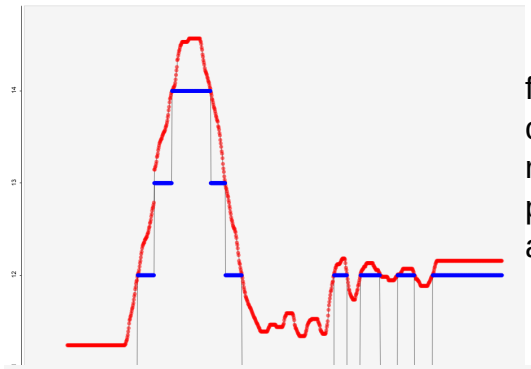
12 Coarse and Fine Control

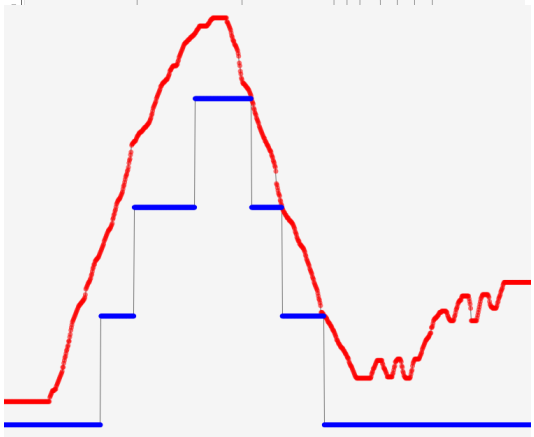
I was thinking about the fact that relays break if they have to switch too many times and that they can't switch instantly. I decided to try combining the relay array, which has a limited resolution, with a transistor (more specifically the IRLB8721PbF MOSFET) that can be controlled with PWM through the analogWrite function of Arduinos.

I started thinking of the relays as a coarse adjust and the transistor as a fine adjust. I could have used the transistor to just make up the difference between the relay steps, but instead I gave the fine adjust a bit of extra capacity.

I modified the Processing program I wrote earlier to simulate adding resistors in binary patterns, and added a fine adjust. I used this program based on my earlier Processing sketch to write and test an algorithm for coarse and fine control. My program can be used with any arbitrary size resistor/load capacity for the fine adjust. I realized that giving the fine adjust some extra capacity has some benefits. It works by stepping the relay values only when needed, as shown in the following image where the orange line is the overall required load, the green is the load taken by the relays and the blue is the load taken by the fine adjust.







This first graph shows the overall load in red and the load from the relays in blue, and shows that if the fine adjust has only just enough load capacity to cover the gaps between the relay steps, while the load can now be any value, there are points where very small oscillations can make the relays switch a lot.

A bit more fine adjust capacity gives some “wiggle room” so that small oscillations won’t trip the relays unless the needed load changes a lot and the fine adjust is maxed out.

Ideally the fine adjust gives just enough capacity to handle any adjustments needed when the grid’s load is fairly constant, and then the relays only get used when someone turns on or off a device. The fine adjust load “dimmer” might cause some distortion to the power wave, but the effect should be minimal if the fine adjust is a small fraction of the full ELC’s capacity.

I want to mention that combining a coarse and a fine control is a commonly used design for controlling systems with more resolution. I also assume that some designs have used the method of giving “wiggle room” to the fine adjust so that a coarse adjust that only has discrete steps doesn’t have to step as often, but I didn’t take the time to find any examples of that.

I did find some ELC designs with some kind of coarse and fine control. One of the ELCs in the Existing ELC Designs section (this one) uses analogWrite() on one of their outputs. Also, by searching specifically for ELCs and coarse and fine control, I found an article describing a system that combines relays that turn circuits on and off with a finely adjustable triac controlled resistor (homepower magazine issue 33 see page 14), but the ELC is a commercial one and I couldn’t find much else about it.

There’s also still room for improvement to my coarse and fine algorithm. Right now if the load changes a large amount, the relays step through every setting in between as they only step when they have to and only the minimum amount vs moving two steps at once to get ahead of the load’s trend. Creating some space ahead of the direction the load is changing would need to be balanced with not changing too much when the load is fluctuating around a fairly stable point.

13 Software

The program I wrote is C++, uses the Arduino.h library (so it’s “written in Arduino” except I used the PlatformIO extension for VSCode instead of the Arduino IDE), and was tested on an ESP32. I tried to use object oriented programming and define interfaces for different functions. For example, the class for controlling an array of relays and the class for PWMing a transistor both implement the LoadAdjust interface. This should make it easier to add new components and features to my software in an organized way. I recognize that my software is not optimized for fast performance, but when some of the cheapest microcontrollers (ESP32s) are also extremely fast, I think it’s fine to prioritize ease of programming.

I also used an Arduino library I wrote last summer that includes classes for reading some kinds of encoders to measure the speed of the motors. It defines an interface for encoders, so when I was having trouble with the speed measurement encoder and switched from one that outputs pulses to one that communicates over I2C I only had to change a couple lines of code.

An autogenerated Doxygen reference for all the classes in my program can be found here: https://joshua-8.github.io/micro-ELC/hierarchy.html

13.1 Initial prototype: v0.1.0

- Tachometer readings
- Binary relay control
- PID

13.2 Improvements to code (for the second round of tests with the modified original prototype): v0.2.0

- LoadAdjustCoarseFine (see notes in the section above)
- PWM load control with LoadAdjustAnalogWrite class
- PID tuning and some bug fixes.

14 Parts

part	cost (prototype)	cost (Alibaba)
8 channel relay board	$10.00	$5.00
ESP32	$10.00	$2.00
2x DC geared motors	$10.00	n/a
Mosfet	$2.25	$0.60
Triac	n/a	$0.32
AS5048b encoder	$15.00	n/a
Adjustable regulator	$3.00	n/a
Wire- various gauges	BDW (free)	?

I had most of the parts I used in the prototype from previous projects, so I don't have a full list of every component with exact prices (like wires and a small breadboard). An ELC could probably be made for under $10 if a few are built at once and cheap suppliers are use. That is not including the cost of the wire for the load resistors, and that could be a significant cost unless spare scraps of wire can be found or already existing loads like heaters and pumps get used instead.

Also, when working with mains power the esp32 should probably be protected with optoisolators.

15 Misc

15.1 How to power the electronics of the ELC

Maybe a "wall wart" power adapter that regulates voltage to 3 or 5 volts? There will be power near by since there's a generator. Or maybe a usb phone charger?

The ELC will probably run on well under 1 amp of current, since powering all 8 relays is under half an amp.

A limiting factor is the ESP32's usb port: https://www.reddit.com/r/esp32/comments/b00juz/esp32_vin5v_current_limits/).

In my testing, after a minute of powering all 8 relays, a component near the USB port gets very hot! A separate power supply from the esp32's USB port is required if relays are used.

For my testing, I ended up using a 5 volt power supply brick to power the relays and motors, and my computer's USB port to power the ESP32, though the whole thing could have been run from only the power supply.

16 Potential next steps:

16.1 Frequency Measurement From AC Power and Zero-Crossing detection

A circuit that sends a 3 volt pulse to the microcontroller when the AC power is very close to zero volts is an important part of many ELC designs.

This "zero-crossing" detection could be used to measure the frequency (and speed) of the generator.

I also think that potentially the ESP32 could poll an analog pin and calculate frequency, zero crossing, and voltage without a full zero crossing circuit, and just a circuit to bring the voltage to a safe level. This could be an example of software complexity saving hardware complexity.

16.2 Zero-Crossing Switching

Zero-crossing detection could also be used to trigger switches when the voltage is smallest to reduce electrical noise. Since my prototype used relays which switched comparatively slowly, and a DC motor, I didn't try to add this feature.

16.3 Triac Control

Driving triacs involves sending pulses at just the right time in the AC cycle to turn the triac on for different amounts of the AC wave (the triac turns off as it crosses zero). This would be another use for zero-crossing detection.

16.4 Load Prioritization

One feature that many ELCs have is a method for prioritizing which loads lose power first if the load needs to be decreased. This means that some of the ELC loads can be useful loads other than just water heaters. While turning loads on and off in a linear sequence has fewer different values than a binary sequence, a linear sequence allows for prioritizing loads. I think it would be simple to add a linear sequence LoadAdjust to my software if needed.

16.5 Conclusion

I realize that my prototype is far from a design that someone could actually use for a micro hydro system. I still think it shows that microcontroller-based ELCs have the potential to be cheap, effective, and easy to upgrade and modify. I hope that maybe something in this document could be useful to someone designing an ELC in the future like how the designs I researched at the start of this project were useful to me.