Build a LED Analyzer

by **bitrex** on June 17, 2011

Table of Contents

Intro: Build a LED Analyzer

When building circuits with LEDs, it's nice to have good data on the characteristics of the LEDs that one is using! Knowing the forward voltage for an LED or string of LEDs at a particular current can help a designer define the power supply requirements of the circuit, and even the lowliest "LED with a current limiting resistor" setup benefits from knowing the LED forward voltage to size the current limiting resistor appropriately. LEDs can also come in handy in other situations where their light output is incidental, such as constructing current sources, or using them as sources of approximately constant voltage in tube amplifier bias circuitry. In applications such as these knowing the I-V characteristic of the LED to be used in a circuit can be particularly important.

As anyone who has bought batches of LEDs on Ebay from Chinese suppliers knows, a detailed datasheet on the characteristics of the LEDs can be pretty hard to come by! That's where this circuit comes in: with the help of an Arduino and a few external components, one can build a LED Analyzer that will analyze a LED's I-V characteristics and spit out an equation that one can use to calculate its voltage drop from its forward current and vice versa.

Here's a quick video of the analyzer in action:

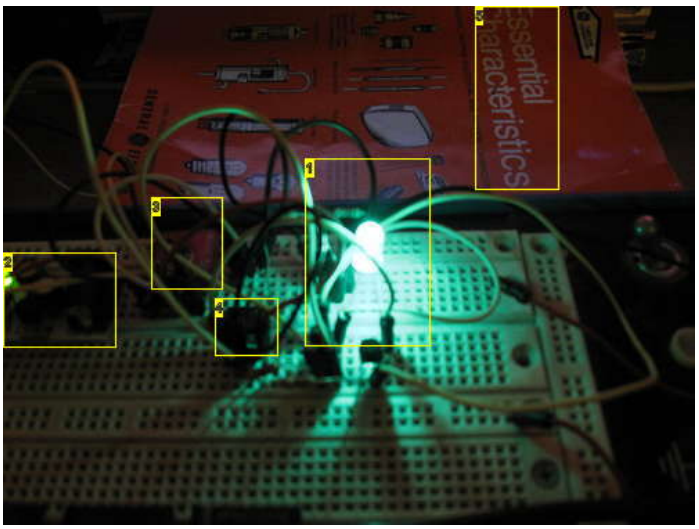


Image Notes
1. Don't look into LED with remaining eye!
2. Really Bare Bones Arduino Board from Modern Device Co.
3. MCP4725 DAC breakout board
4. LM358
5. They're essential!

Step 1: The Setup

Here's the setup. The Arduino is on the far left, and connected to its SCL and SDA lines is the very handy MCP4725 I2C DAC. This DAC converts an increasing sequence of output voltages to a sequence of increasing currents with the help of our friend the LM358 op amp and transistor Q3 IC1A, T1, and R4 form a current sink; feedback around the op amp keeps the current through R4 equal to a value defined by Vin/R4, where Vin is equal to the output voltage from the MCP4725. We have to be careful, however, to not let the input voltage to the non-inverting input go too high - the LM358 has a common-mode input voltage range above which it ceases to behave properly. Therefore the output from the DAC is restricted to a maximum level of 3 volts, which implies a maximum current through R4 of 30 mA. This is approximately the maximum current rating for most 5mm LEDs. C1 and R3 help to prevent the LM358 from oscillating.

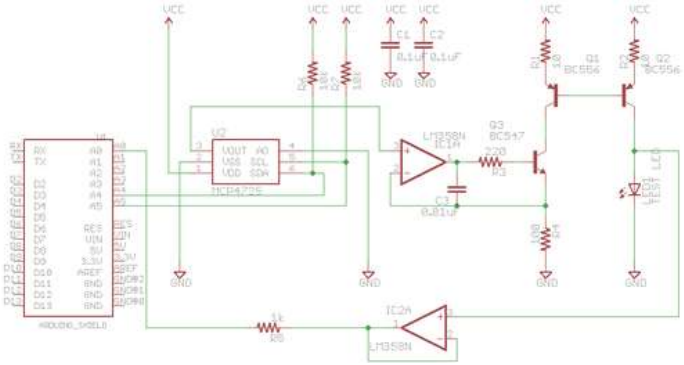
Of course, Q3 is a current *sink* and what we want is a current *source* , to pump current into the LED so its forward voltage can be measured. Q1 and Q2 perform this turnaround function. IC2A and R5 buffer the sensed voltage and protect the input to the Arduino.

The Arduino runs a sketch that ramps up the DAC voltage, and hence the current through the LED, and then reads the resulting LED forward voltage through one of its analog input pins. It then communicates with a Python script running on a PC via its serial to USB link, and this script organizes the data and displays a graph of the LED under test's I-V characteristic. It also runs a curve-fitting routine that generates coefficients for an exponential function similar to the Shockley diode model. To get a smaller error in the fit function, the Python script has the option of running the analysis multiple times and the averaging the data; the data generated for the "Results" section that follows was averaged over ten runs of the analyzer.

Here's a quick parts list for the hardware:

Arduino (1x)
MCP4725 DAC breakout board with pull up resistors and capacitor (available at http://www.sparkfun.com)
LM358 dual op amp or similar (1x)
BC547 NPN transistor or similar (1x)

BC556 PNP transistor or similar (2x)
10 ohm resistor (2x)
220 ohm resistor (1x)
1k ohm resistor (1x)
0.1uF bypass capacitor (1x)
0.01uF capacitor (1x)
LEDs to test!



Step 2: The Results - Red LED

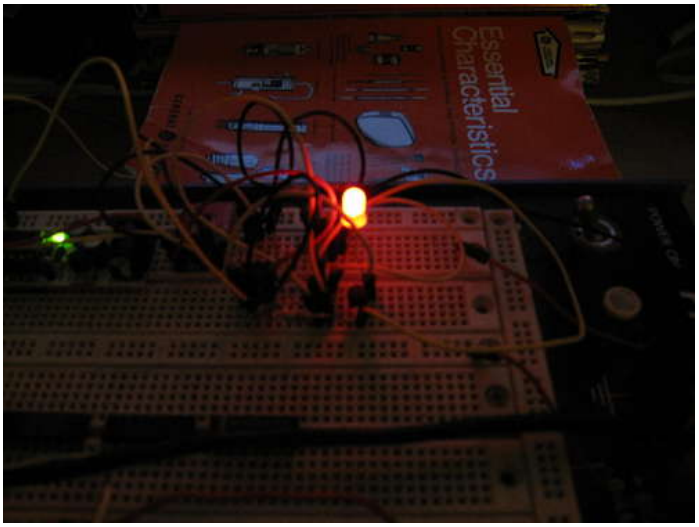
Here's a bog-standard diffuse 5mm red LED on a breadboard with the test setup, along with an image of the data the Python script outputs after analyzing the LED. IMPORTANT NOTE: the second "e" in the "Exponential Fit" line is Euler's number, 2.71828....The first "e" represents the exponent in scientific notation. So, represented another way, the first coefficient of the exponential function would be "1.1*10^-8." The LED current on the bottom axis is in amperes, and the plot is semilogarithmic on the X axis so each full division is ten times the previous value.

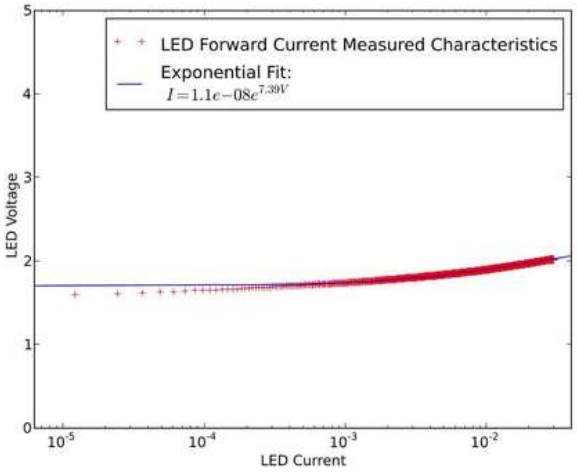
No big surprises here. The forward voltage of this LED is just under 2 volts for most of the current range, and it crosses 2 volts right at the upper limit of the scale, at around 20 - 30 mA.

What's going on at the bottom end, though? The generated exponential fit doesn't seem to fit the data very well down there. Here's what's happening: The actual diode current doesn't exactly adhere to an exponential curve. In the real world, all real components have parasitic elements that cause their behavior to deviate from an "ideal" model. In the case of an LED, the major player is the LED's internal resistance. As current through the LED increases, this internal resistance causes a voltage drop, which in turn causes the measured voltage across the LED to be _greater_ than what one would expect from a strictly exponential model. The curve fitting routine does its best to fit a curve to the data, but because of this nonideality it can't really do it, because the LED is not behaving in a strictly exponential fashion!

Because of this issue, the exponential fit generated is NOT really suitable for use as a SPICE model. To generate a proper SPICE model, one would calculate the LED exponential parameters at a low level of current where the ohmic term doesn't have much effect, and then calculate the value of resistance separately. Properly calculating the value of resistance from the measure data is somewhat tricky - hopefully it will be in the next version! The fit curve does, however, allow one to calculate a forward voltage for a particular value of current in the region where the curves overlap.

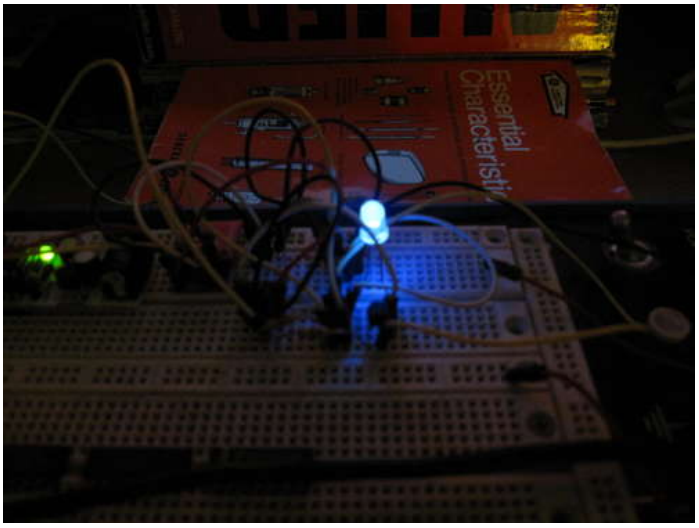
There is also an error due to the measuring setup. The current mirroring arrangement has an error because of the base currents of the transistors! The true current through the LED is approximately given by: (alphaQ3*Isense) - 2*(alphaQ3*Isense)/beta, where alphaQ3 is the alpha of transistor Q3 and beta is the beta (current gain) of transistors Q1 and Q2. Fortunately, so long as beta stays high over the region of interest, the error is not too bad.

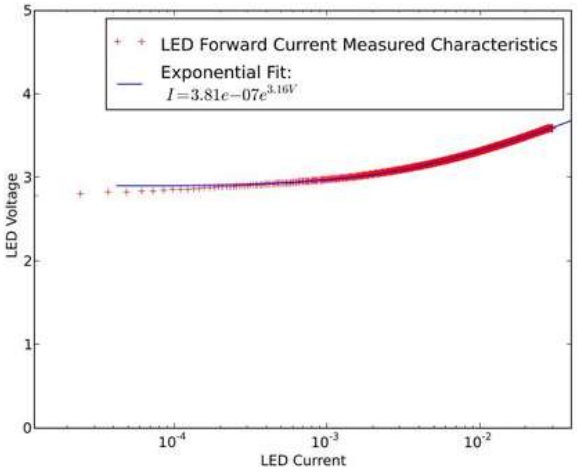




Step 3: The Results - UV Led

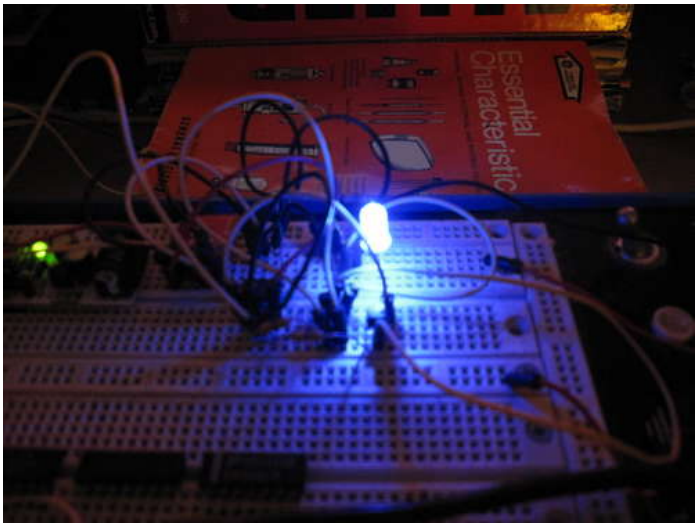
Here are the results for a no-name 5mm UV Led - unfortunately its deep violet color doesn't show up on camera and looks more blueish in the photo. As expected, it has a high forward voltage across its operating region, with a dramatic upswing in forward voltage as the current passes around 10 mA due to the previously mentioned ohmic resistance. One thing to remember about driving LEDs with DC currents is that almost all the "action" with respect to brightness happens at the lower end, and there's a region of diminishing returns where increasing the current does not increase the apparent brightness very much. For example, with this LED there is not a dime's worth of difference between its brightness at 10 mA and 30 mA, and yet the voltage across the LED increased substantially. There's no point in driving it with a DC current higher than is needed for maximum brightness - you're just wasting power! (Pulsed drive is another story, see: http://www.piclist.com/techref/io/led/pulse.htm).

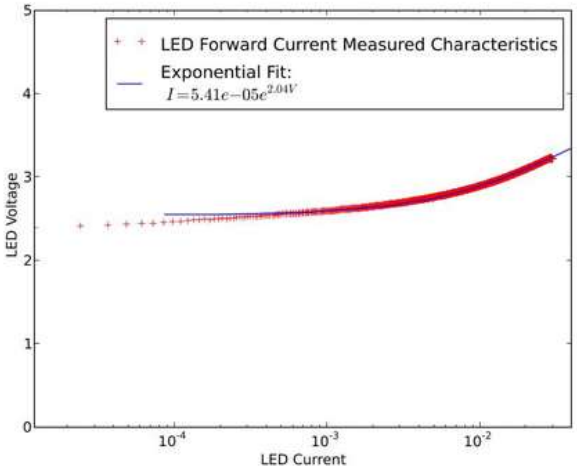




Step 4: The Results - Blue LED

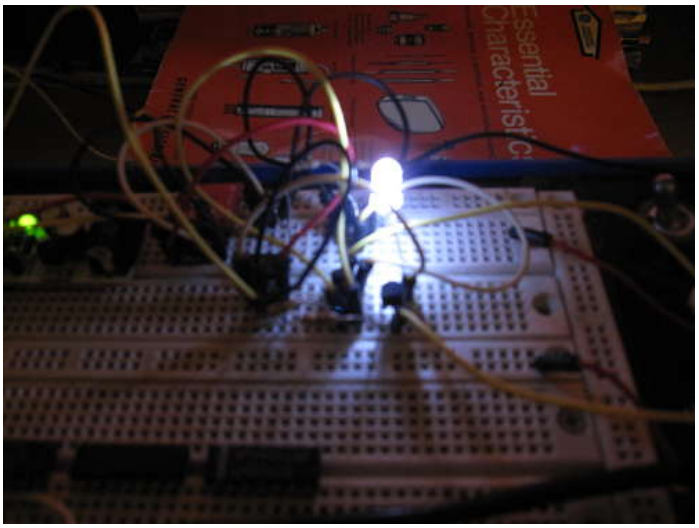
Here's a 5mm "Super Bright" Blue LED - everyone's favorite! It's characteristics are somewhat similar to the previous UV LED, but its maximum forward voltage is somewhat less. It really is bright!

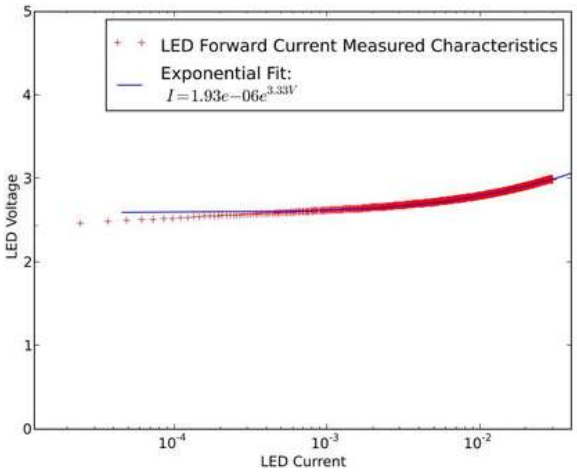




Step 5: The Results - White LED

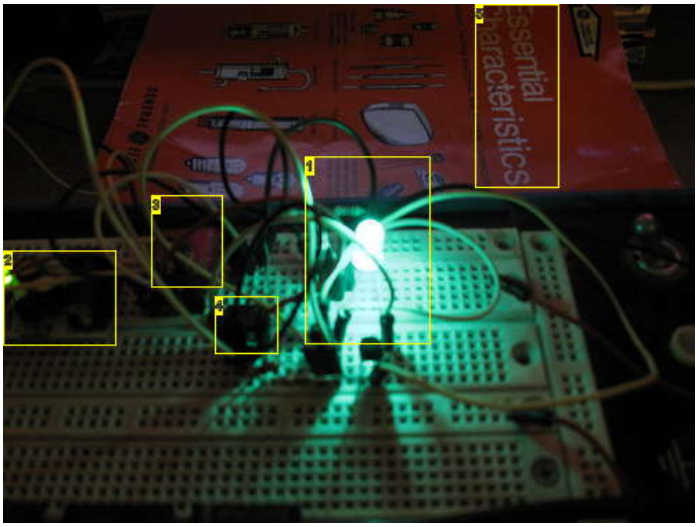
Here's the measurement of an unknown 5mm white LED. Its characteristics are quite similar to a blue LED - which is to be expected, since white LEDs are in essence blue LEDs with an added phosphor to emit yellow light in addition. The sum gives a white appearance.





Step 6: The Results - Green LED

This is a 5mm "Super Bright" green LED. It appears dazzlingly bright, to me almost brighter than the "Super Bright" blue LED analyzed previously. It almost reminds me of the coherent light emitted by a green laser pointer, and can project a bright green spot across a large distance. Interestingly, its I-V characteristics are similar to the blue and white LEDs, leading me to believe that all three are made with the same indium-gallium-nitride process.



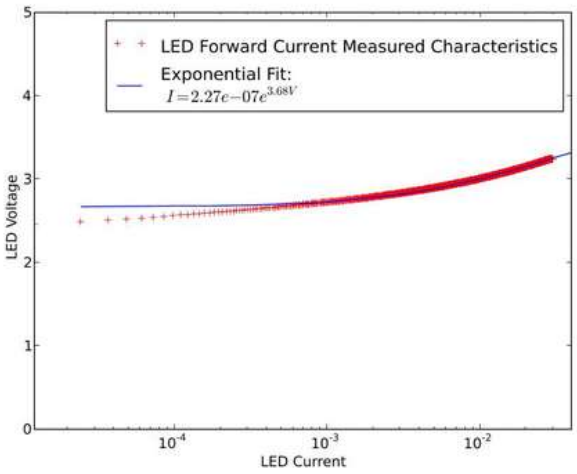
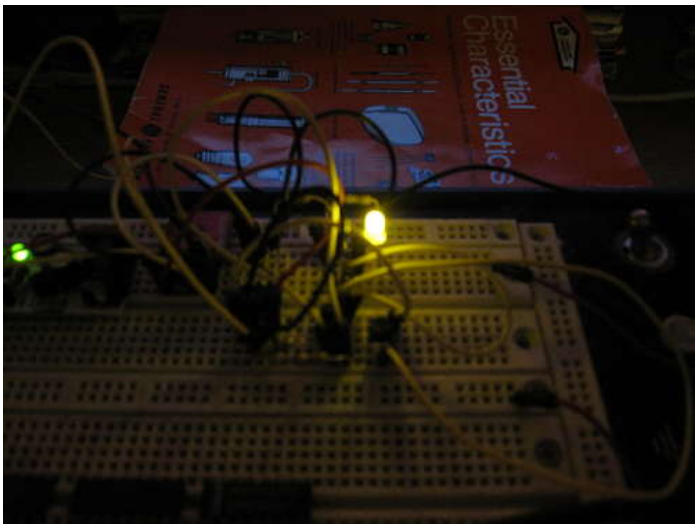


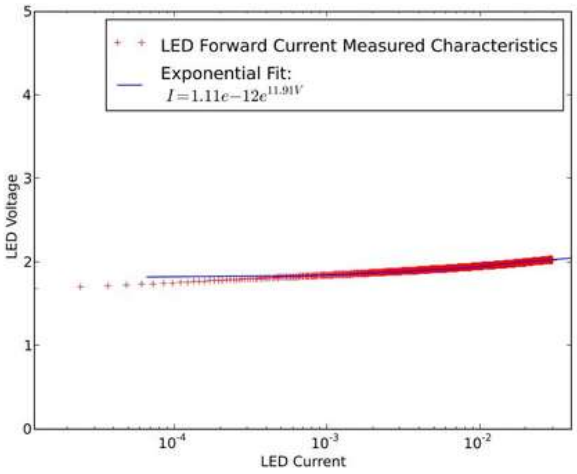
Image Notes

1. Don't look into LED with remaining eye!
2. Really Bare Bones Arduino Board from Modern Device Co.
3. MCP4725 DAC breakout board
4. LM358
5. They're essential!

Step 7: The Results - Yellow LED

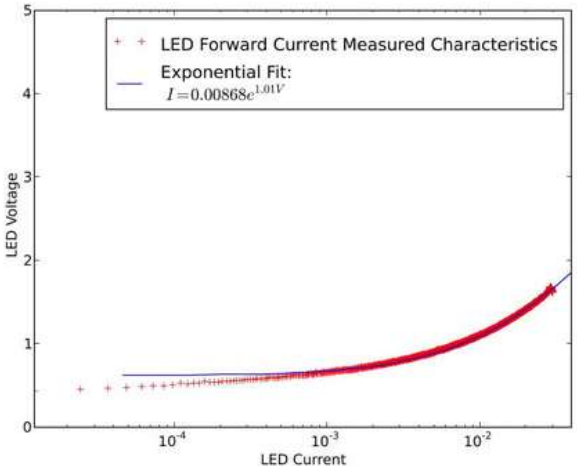
Something a little different: this is a 3mm diffuse yellow LED. This one surprised me as compared to the others it has far less "upswing" at higher currents compared to the others - its forward voltage rises very little over a wide range of currents. Perhaps the 3mm LED has significantly less internal resistance than the 5mm type? This might be a good LED to use, for example, in bias applications when one wants a constant voltage source over a wide range of currents. Of course, this tester doesn't have anything to say about what an LED's behavior over temperature changes is - all diodes are sensitive to temperature - but that's another project! It is possible to see small changes in the measured data if one, for instances, clamps the LED tightly between thumb and forefinger to warm it up while the test is running.





Step 8: The Results - Mystery Diode

Of course, this tester will work on most any kind of PN junction diode, not just LEDs! Here's the result from measuring a "mystery" small signal diode; maybe something like a 1N914/1N914A? So much for the "0.6 volt" rule! This diode has nearly 2 volts across it at 30 mA of current.



Step 9: The Code

That's all the LEDs I had time to analyze as the LED contest submission deadline approaches! Getting this project up and running was a lot of fun, and lurking in what on the surface seem to be pretty simple devices is actually quite a bit of hidden complexity. I have included the code for the LED Analyzer, both the Arduino sketch, and the Python script which runs with it. Once the circuit is set up properly running the program is a breeze: just connect the Arduino via its USB to serial cable and run the script from the command line; options for selecting the COM port, baud rate, and number of iterations to run are available and are explained by running the script with the -h flag. The script does require a few add-ons for Python to be installed such as matplotlib, Numpy, and PySerial, for Windows users I have created an executable using PyInstaller that will run as-is from the command line. It's available here: http://code.google.com/p/led-analyzer/downloads/list

Next steps? I hope to be able to create a routine that will analyze a LED and generate a proper SPICE model, including the correct values for the diode saturation current, emission coefficient, and series resistance that can be used for simulation. Hopefully when I have some time...

If anyone builds this tester, please let me know how it works out!

File Downloads

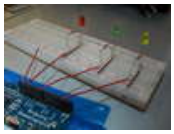
LEDAnalyzer.pde (2 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'LEDAnalyzer.pde']

LEDAnalyzer.py (8 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'LEDAnalyzer.py']

Related Instructables

Make a digital logic analyzer for less than $1 by Junkyard John

Logic Probe Kit by Mrdon219



How to Make LEDs Flash to Music with an Arduino by Hyrulian

.... by vithrar

USB controlled home automation hack by chr

Open Source Microchip LED / PWM Driver Project by jimk3038