Sean Voisen

Re-creating the "breathing" LED

October 31, 2011

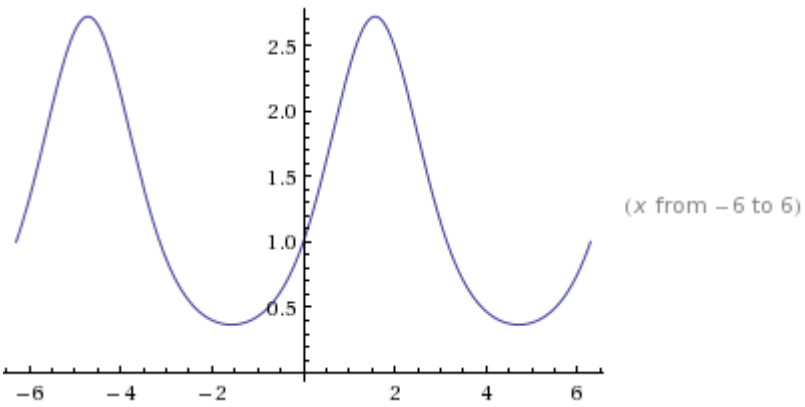
Perhaps it's not surprising, but Apple has a patent on the iconic "breathing" pattern used for the sleep indicator LED on all their computers.

Recently — for a personal project — I wanted to see if I could replicate this effect. It turns out that I wasn't alone. Ladyada tried to reverse engineer the pattern a few years ago. Unfortunately, she stopped short of providing anything — like code — that the lazy web surfing Arduino hacker might use to recreate the effect. That's the purpose of this short tutorial.

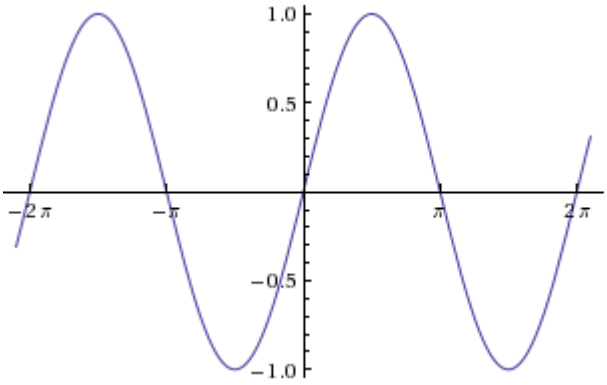
Digging Into the Math

The Apple patent claims that the breathing pattern is a simple sinusoid, but observation of one's own (heavy) breathing will show that the pattern is a little more complicated than that. Maybe a sine wave works for Apple, but it doesn't look quite right to me. In my own breathing, I tend to ease in to a fast inhale, and stop suddenly before easing out to a fast exhale. Also, the period between inhalation and exhalation happens to be shorter than the period between exhalation and the next inhalation. This is not a simple sinusoid, as the oscilloscope plot from Ladyada's own investigations will attest.

A commenter on Ladyada's blog suggested that the pattern is probably more accurately modeled by $f(x) = e^{\sin(x)}$. Plotting this equation gives the following:



Compare the curvature characteristics of this plot with the simple sinusoid below:



The plot of f(x) = $e^{\sin(x)}$ has wider "troughs" (periods between inhale and exhale) and narrower "peaks" (periods between exhale and inhale), more accurately matching natural breathing patterns. As a simple experiment, try breathing "sinusoidally" and you'll see how unnatural it feels.

Turning it Into Code

For my own experimentation, I used the ubiquitous Arduino. The Arduino supports analog output using pulse-width modulation (PWM) mapped to integer values from 0 to 255. To recreate the breathing LED, this means manipulating the original equation f(x) = $e^{\sin(x)}$ such that the amplitude fits within the PWM range.

I took enough math in school to know that the minima and maxima of any equation occur at critical points in the equation, where the derivative of that equation is either 0 or its not differentiable. Beyond that, I left it to Wolfram Alpha to do the hard work. It turns out that the minimum of the wave occurs at 1 - e, and the maximum at e - 1/e. Using this information to adjust the amplitude of the equation such that it fits within the 0 to 255 range gives the following:

$$\left(\exp(\sin(x)) - \frac{1}{e}\right) \times \frac{255}{e - \frac{1}{e}}$$

Swap *x* for the number of seconds that have elapsed, and map the above equation to PWM output on any supported Arduino pin, and you have the beginnings of a breathing pattern. The problem is that the frequency may be too high or low (depending on your preference), and so the breathing will appear fast or slow. Easy enough: Multiply *x* by any value to adjust the frequency. I like π/2.

Finally, 1 - e, and 255/(e - 1/e) are constants, and can be pre-calcuated to reduce overhead. The final Arduino sketch is as follows (with the LED connected to pin 11, a suitable resistor in series, yadda, yadda ...):

```
#include <math.h>

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  float val = (exp(sin(millis()/2000.0*PI)) - 0.36787944)*108.0;
```

```
    analogWrite(ll, val);
  }
```

Gratuitous Video

And, in case you don’ t have an Arduino handy, here’ s a short video of the final effect:

Conclusion

You might be asking: Is it really that big of a difference? Wouldn’ t a simple sinusoid suffice? To answer the latter question: yes. To answer the former: the difference is noticeable, but only slightly. Steve Jobs was a notorious perfectionist. I like to think that he would’ ve cared about such things.

More posts in the archives →