Troubleshooting is a big part of my job as a Radar Technician in the US Marine Corps. Unfortunately, as you probably know if you've worked with electronics for long, good test equipment can often be prohibitively expensive, even for the government. One of the most common aspects of our troubleshooting process is checking Radio and Intermediate frequency signals with a spectrum analyzer. The particular model of analyzer we use has an enormous frequency range up into the tens of gigahertz, but its input power range is limited to about 30dBm. One signal we often look at is supposed to be at 28dBm, very close to the input power limits of the analyzer. If the signal hasn’t been properly attenuated within the system, the output could potentially be above 30dBm. There are several other areas where similar potential damage could occur.

Seeking to prevent this potential damage, I looked for an adjustable attenuator, in order to lower the RF output to a more useable level. I did end up finding a few adjustable attenuators, but I found that most of the attenuators that met my needs cost upwards of $2000.

So I set out to make my own adjustable attenuator with digital control, meeting the following specifications:

Required:

- Operation from 1 to 2GHz (the frequency range most often used in my line of work)

- Protect Spectrum Analyzer in case of very high power (input power should be well over 30dBm, which is equivalent to 1 Watt. Ideally we’ll have an input capability of at least 40dBm, which is equivalent to 10 Watts)

- Wide range of attenuation levels (The commercial attenuator I was looking at operates from 0-70dB, so I want to approach that same range)

- Good accuracy

- Easy to operate

- Easy to calibrate at the user-level using a Laboratory calibrated frequency generator and spectrum analyzer (all of our spectrum analyzers and frequency generators are calibrated regularly)

Optional, but desired:

- Operational at a much wider frequency range. Ideally DC-3GHz

- Should Display at least an approximate input power level