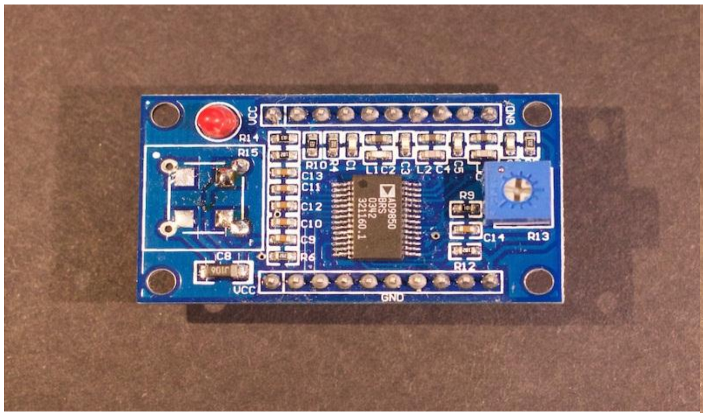
HARDWARE OF LISSAJOUS CURVES

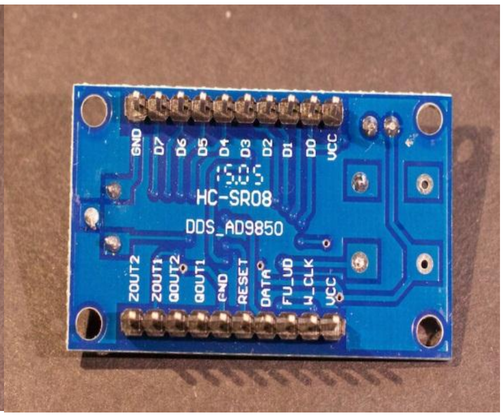
The key component is the AD9850 DDS synthesizer. Conceptually this generates a sine wave whose frequency is given by

$$f=f_m\times n2_{32}.$$

where n is a 32-bit number we choose, and f_m is the frequency of the master clock supplied to the chip.

Rather than buying the chip directly it’s easier to buy a module containing the AD9850 from eBay. Most of the these modules consist of the AD9850plus a 125MHz oscillator. I think most of the modules are clones of the same design, but *caveat emptor*. Mine looked like this:





Note: The AD9850 can also produce a square-wave output, and the trimmer adjusts its duty-cycle: we can ignore this.

In essence, we just apply power, send the relevant configuration data from the Arduino, and we’re done.

In practice, there is one issue which has us reaching for the soldering iron: we need to synchronize the master oscillators. That’s just a case of removing the oscillator from one module and bodging a connection to the other. It’s easy to do if you remember which pin to connect! I suspect doing this isn’t ideal: the wire’s carrying (and thus presumably radiating) 125MHz.

Before I did this, I found the master frequencies differed by about 6ppm. That’s a bit better than I expected.

I wasn’t able to find a data-sheet for the precise oscillators on my boards, but they’re a bit like the TXC 7C series. Certainly pin 3 (diagonally opposite the pin 1 spot) is the clock output.

Although sharing the master oscillator ensures that the synthesizers won’t drift with respect to each other, they can still be a constant phase apart. Assuming that the synthesizers lock to the phase of the master clock, they will potentially be an integral number of cycles apart. At 125MHz, one cycle takes 8ns.

If we’re synthesizing a 100kHz sine wave, that’s a phase error of about 0.3°. Effectively then, the sinusoids have a random arbitrary offset from each other.

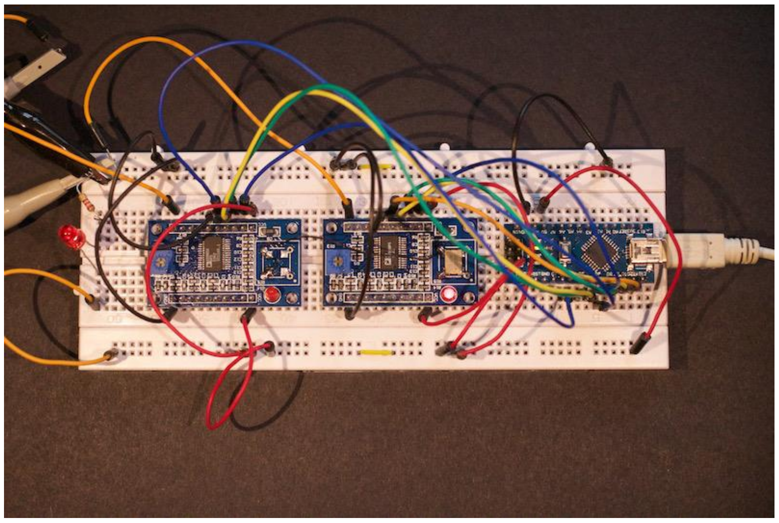
To some extent we could address the problem by better synchronizing the synthesizers’ reset signals and frequency adjustments, but I’ve not explored that.

Instead it’s easy to adjust the phase to the desired value. There are a couple of different approaches:

- The AD9850 has a 5-bit phase register which allows adjustments in increments of 11.25°. This is rather coarse!
- We can simply increase the frequency of one of the oscillators by a small amount, wait until the desired phase difference is reached and then restore locked operation.

The latter can be improved by making the Arduino perturb the frequency for a fixed time. The software described below does this and gives a tuning resolution of roughly 1.5°. One could easily do better, though I’ve no idea about the accuracy of the method.

The original prototype was build on a breadboard,



Source: http://www.mjoldfield.com/atelier/2015/06/ad9850-lissajous.html