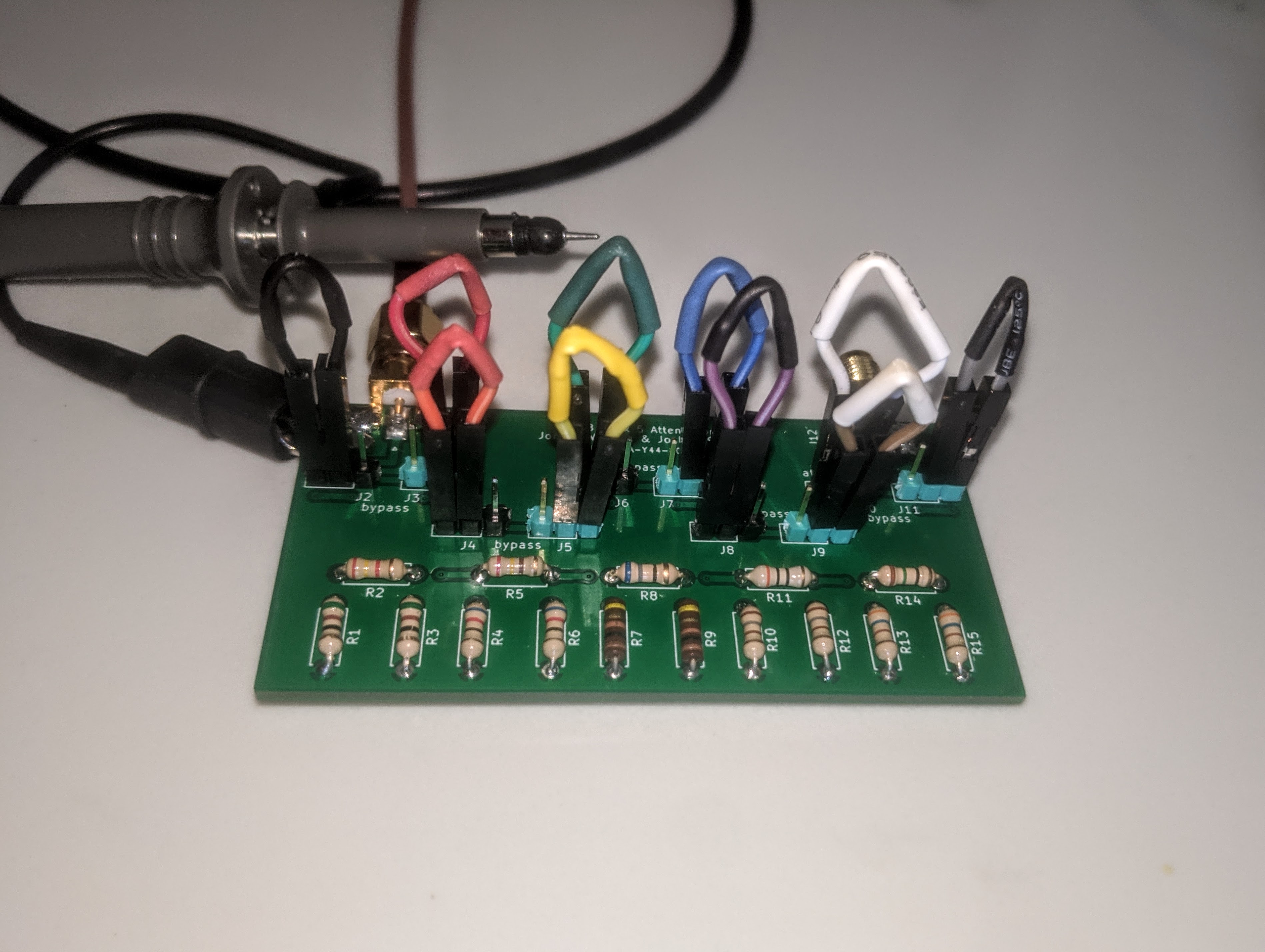
**Build and Test Results**

Build Process - Attenuator

I started the build process by first taking careful inventory of the parts I had and needed. I discovered that I was short 1 surface mount male SMA connector and jumpers. These missing parts were forwarded to Dr. Frohne, where he was able to mail them to me. Following my build plan, I started by first building the attenuator. Since the components were all through hole, the build process for this straight forward. I started with the resistors, followed by the SMA connectors, and lastly, I soldered on the male pins for the jumpers. Thus, I added the parts going from the “flattest” to the “tallest”. A note about the SMA connectors — the foot prints were a bit larger than the pins of the connector, but it was close enough that we could fudge it with some extra solder. The last part of the board that needed to be constructed were the jumpers. Since I didn’t receive any initially, I decided to make my own out of the female-female wires I had on hand. This entailed stripping the wires, cutting them to the desired length, and connecting the exposed stripped ends followed by a covering with shrink tubing. Since the jumper cables were made out of so many colors, I decided to name by board “digital skittles”.

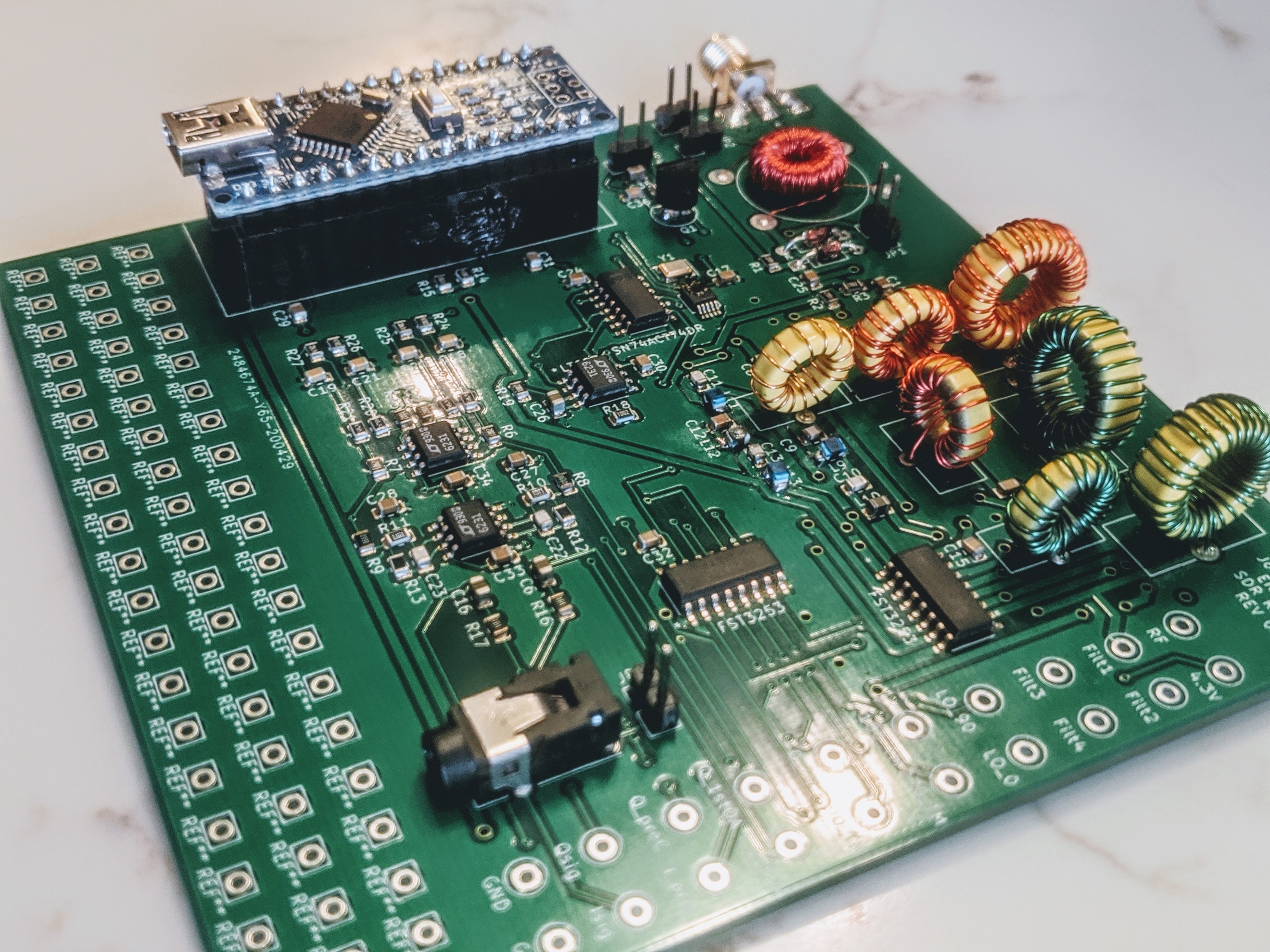


Test Process – Attenuator

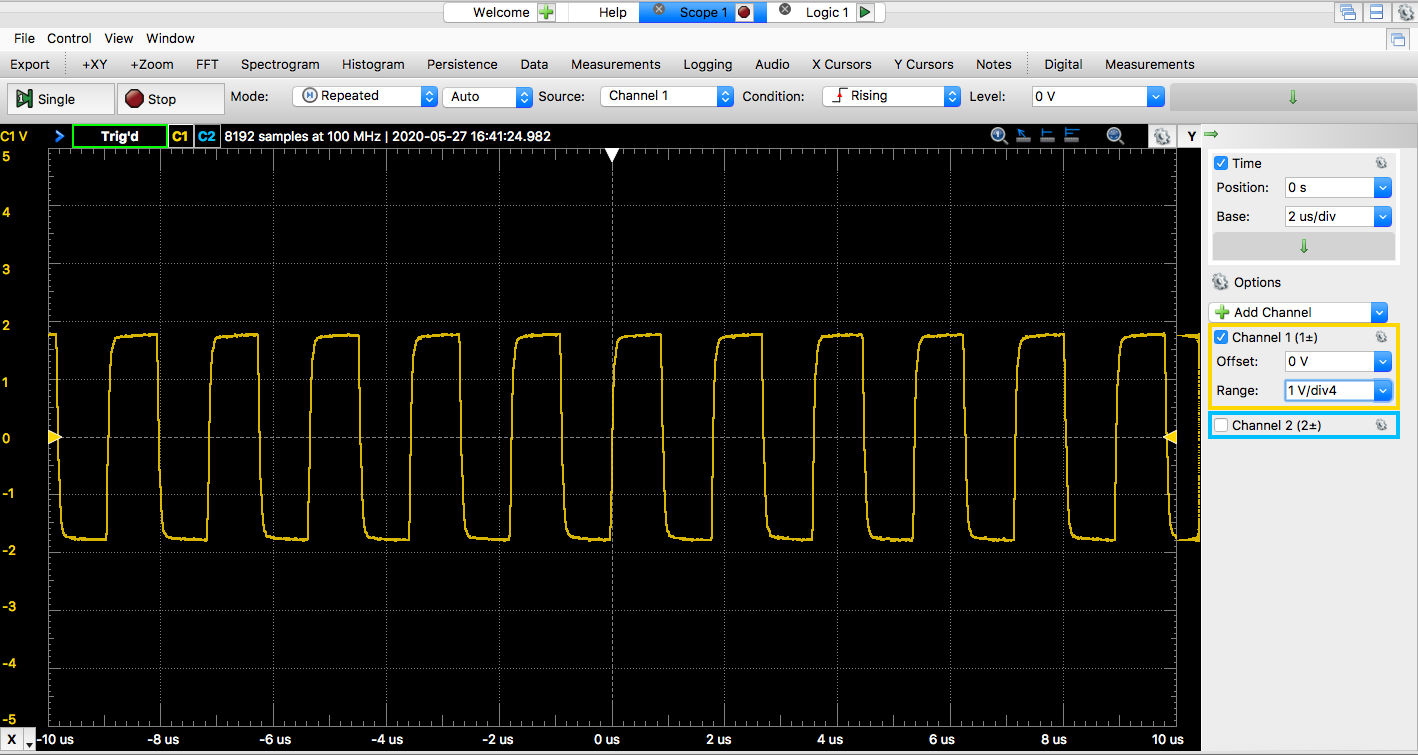
To test the attenuator, I first checked to ensure that all my connections were solid. I examined all the traces, and used a multimeter to check for continuity. I also verified my homemade jumpers worked adequately. Once I checked everything, I used the diligent analog discovery 2 (DAD2) and its related waveforms software to test the attenuator. The attenuator has roughly 80dB of attenuation. Thus, even with an input signal of 5V, the output signal is very small. What I had not looked into was the sensitivity of the DAD2 to small signals. It turns out that the oscilloscope has quite a difficult time with very high frequencies and very small signals. So, in order to account for the limitations of DAD2, I decided to test the attenuator in stages instead of all at once. I also had difficulty attaching a 50-ohm resistor as the load for testing purposes, so I decided to utilize the structure of the board to make it appear as 50 ohms. Since my board was set up with the stages 40, 20, 10, 5, 2 dB attenuation, I could test the 40dB stage by using the 20-10-5-2 stages as the apparent “50-ohm load”. Then, I could switch the input signal to the other side and use the 40dB side as the theoretical “50-ohm load” to test the 20-10-5-2dB stages. I simulated this way of the “load” on LTSpice and compared my results with that of the simulation. From the results of the comparison, I concluded that my attenuator was working as I expected it to.

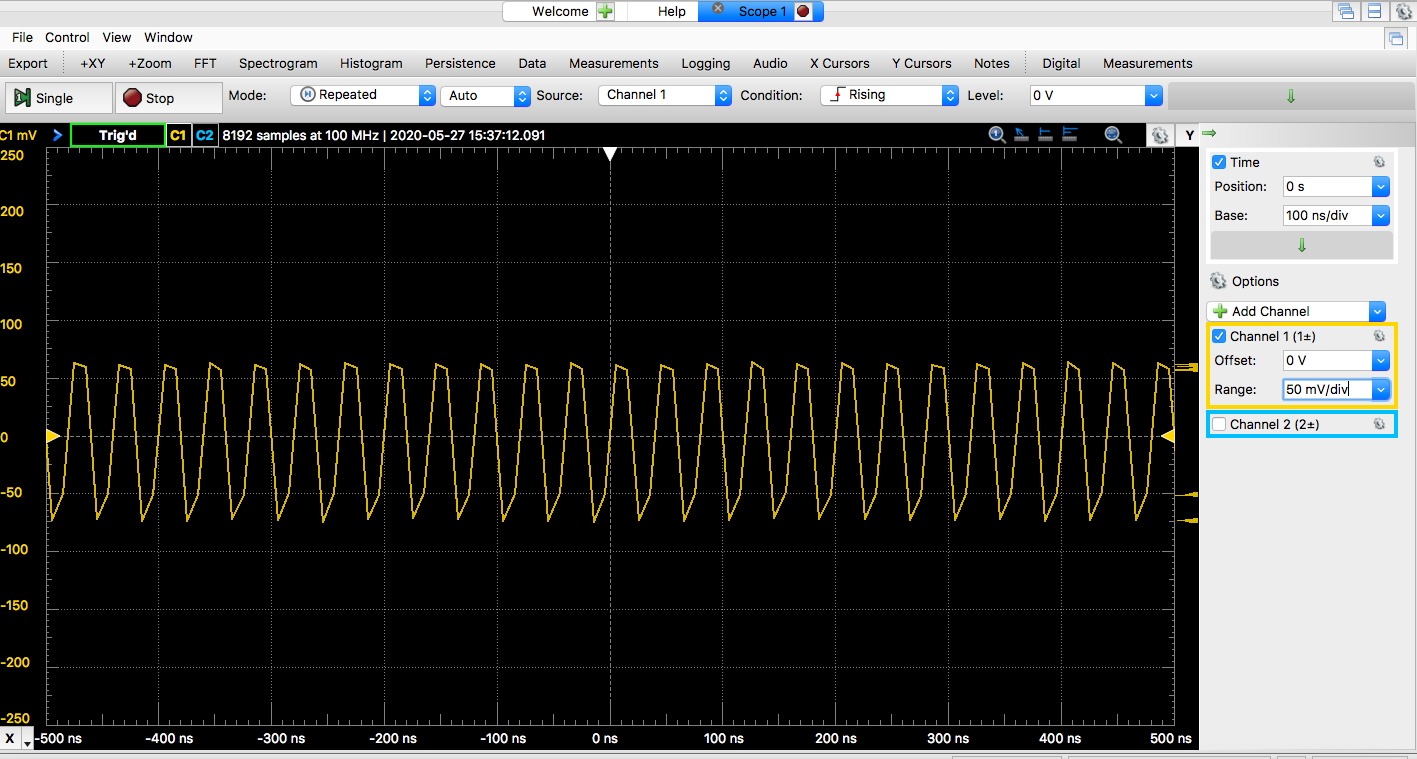
Build Process – Receiver

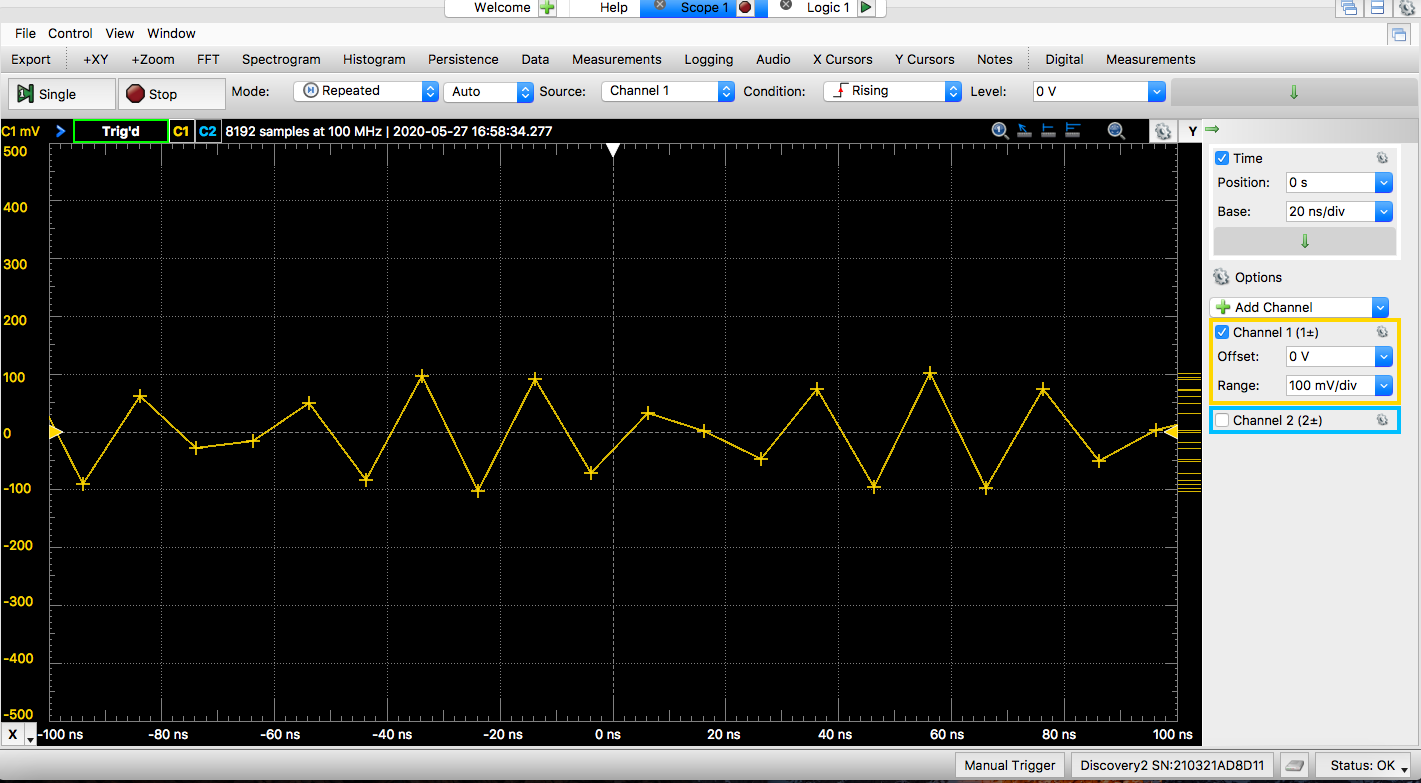
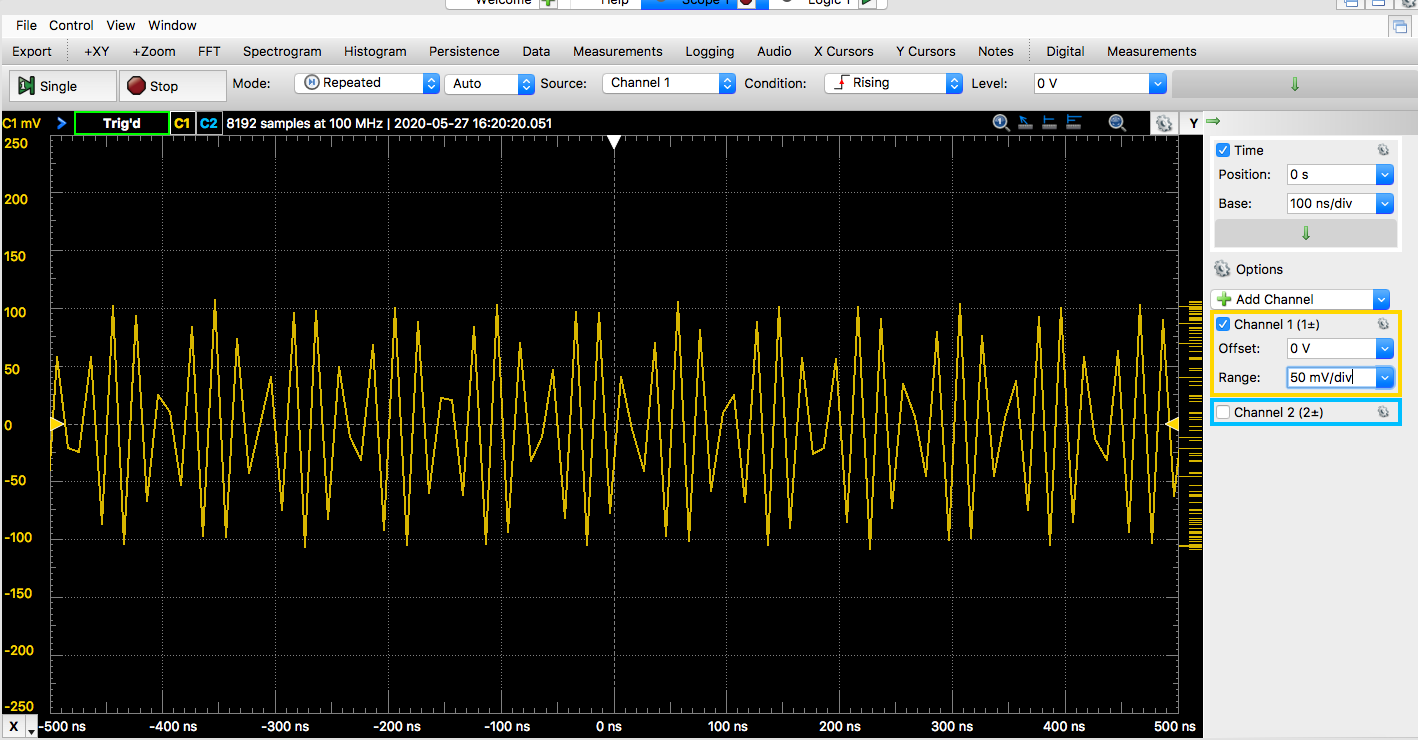
I started building this part of the project by first documenting the parts I received, and recording those that I was missing. I didn’t have enough male pin connectors for the jumper, so I used the pins for the Arduino nano. I had 2 other Arduino nanos that I could use for the project, so I wasn’t concerned with using the connector pins for the jumpers instead. My project partner, Joshua Silver, also pointed out that I forgot to order 100-ohm surface mount resistors. I submitted the missing parts list to Dr. Frohne, and he was able to send me the 100-ohm resistors. The first step in building the board after making sure I had all the parts I needed was applying the solder paste. I took a good amount of time ensuring that I applied just the right amount of solder paste through the stencil to minimize the amount of bridges. Once I was satisfied with the application of the solder paste, I placed all the surface mount parts, starting with the ICs, then working towards the resistors and capacitors. Once I had all the parts placed, I carefully placed it in the toaster oven I purchased for $20 from Walmart. I baked it for about 3 minutes at 410F, which just allowed the solder to flow and turn shiny. Then, I turned the oven off and opened the door to let it cool. Some of the solder joints were brownish in color because of the uneven heating in the oven based on the placement of the heat source and the location of the board relative to it, but it didn’t affect any of the functionality of the components. Once the board was cool, I hand soldered all the through hole parts on. Starting with the SMA connector and diodes, followed by the transistor, jumpers, audio jack, and toroid inductors and transformer. At this point, the entire board has been put together.



Test Process – Receiver

The first thing to test was for shorts between power and ground on each of the ICs. Thus, I took a multimeter and checked for continuity where it shouldn’t be. I found no shorts – but to be certain, I examined each connection again with a magnifying glass and used the multimeter once more to double check. I also checked most of the other traces to make sure that the connections were solid and matched that in the schematic. After this pre-check was done, the next step was to test the local oscillator. I went about this by first downloading Dr. Frohne’s .ino test file from his Github. I had to install the Etherkit si5251 library from Arduino, and remove the old one I downloaded from Github. With this program flashed to the Arduino nano, I plugged in the nano into my board. Using DAD2, I used the oscilloscope to see if I could see an output from the crystal of clock generator… I couldn’t. Looking for possible errors, I eventually stumbled upon the datasheet for the crystal, where I discovered we had the wrong pin numbers in our schematic. We had pins 1 and 2 supposed to be outputting a frequency to the clock generator when it should have been pins 1 and 3. I corrected this error by cutting the traes from the crystal to the si5351 chip, and adding in wires to connect pins 1 and 3 of the crystal to the appropriate pins of the si5351. Using DAD2, I could now see an output from the crystal of a magnitude between 20 and 50mV. But I still couldn’t observe any output from the si5351. Looking back at my KiCad schematic, I found that the Josh and I had accidentally connected our “pull-up” resistors to GND instead of to the 3.3V power. Again, I took my exacto knife and cut the trace connections to ground and attached a wire from the resistors to 3.3V instead. Taking a look at the output of the clock generator with the oscilloscope this time yielded an output of that used in the Arduino code! Then, I further looked at the output from the Johnson counter, and the outputs were 90 degrees out of phase as I hoped they would be! I concluded that my local oscillator was now in good operating condition. The graphs included below are the outputs from the si5351 chip given different code in the Arduino program. The one with the highest amplitude is that with a frequency of 0.56MHz, followed by that with a frequency of 5.6MHz and lastly 50.6MHz. You will notice that the amplitude of each larger frequency gets smaller and smaller. This is due to the limitations of the DAD2 scope program.





While the Arduino as plugged in (before I started to mess with the oscilloscope), checked the voltage on each of the IC to make sure they were within the limitations. I found that my Arduino only output 4.7V. Even when it wasn’t attached to the board, it only output 4.75V. Thus, we can conclude that the board does not draw much current (a very good thing). The 3.3V output was actually 3.5V, and the theoretical 4.3V, was actually 3.9V since it was based on the 4.75V instead of the theoretical 5V. Furthermore, the analog GND was at 1.99V instead of 2.15V. But all these voltages are consistent with the design parameters and the given output voltages of the Arduino.

Once I had the local oscillator working, I proceeded to test the bandpass filters. With an input signal of 10mV, I first discovered that I wasn’t getting any signal through the filters. Inquiring this with Dr. Frohne, I realized that I had not programmed the select lines into the Arduino. Once realizing this, I coded in the desired select lines for a 5Mhz input frequency to select the 2nd Bandpass filter. But to my dismay, I still didn’t get any readings. Realizing this, I checked the output of my local oscillator, and was horrified to discover that it was no longer outputting anything. Concluding that it was probably an issue with the Arduino, I removed it and tried to re-upload the program with some alterations. But I discovered that I could no longer upload to the board. After further investigation, I reasoned that I had fried the 5V USB passthrough on the USB chip on the nano ( reference: <https://forum.arduino.cc/index.php?topic=152716.0> ). Examining my schematic to see why I had fried my Arduino, we discovered that we had hooked up our band pass filters incorrectly to the multiplexer. The reason for this frying the Arduino is still to be investigated.

Further progress on testing the trouble shooting the board will be rewiring the bandpass filters by cutting traces and such, and looking further into the theory of why this misconnection may have burnt out the Arduino. The last thing after this is solved is to check the tayloe mixer is working correctly and that the summing op-amps and low pass filter work as expected by comparing to the LTSpice simulation.