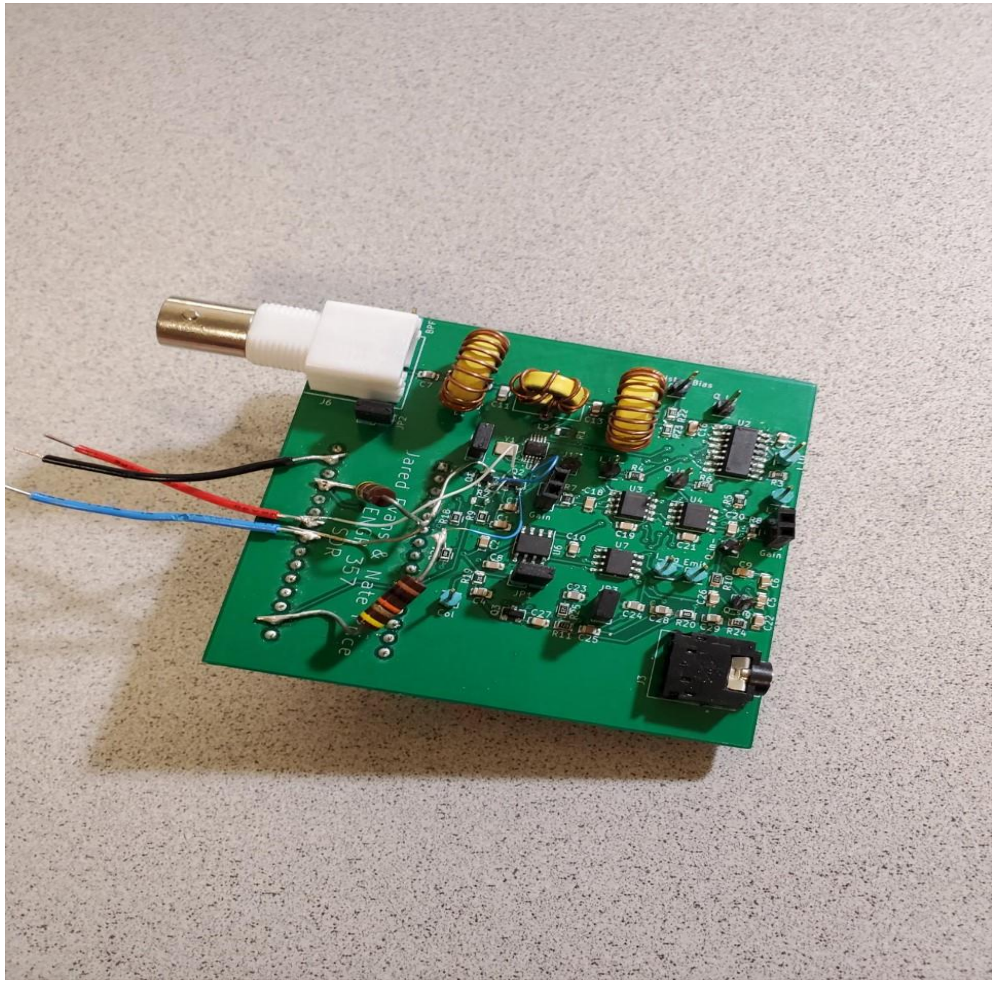
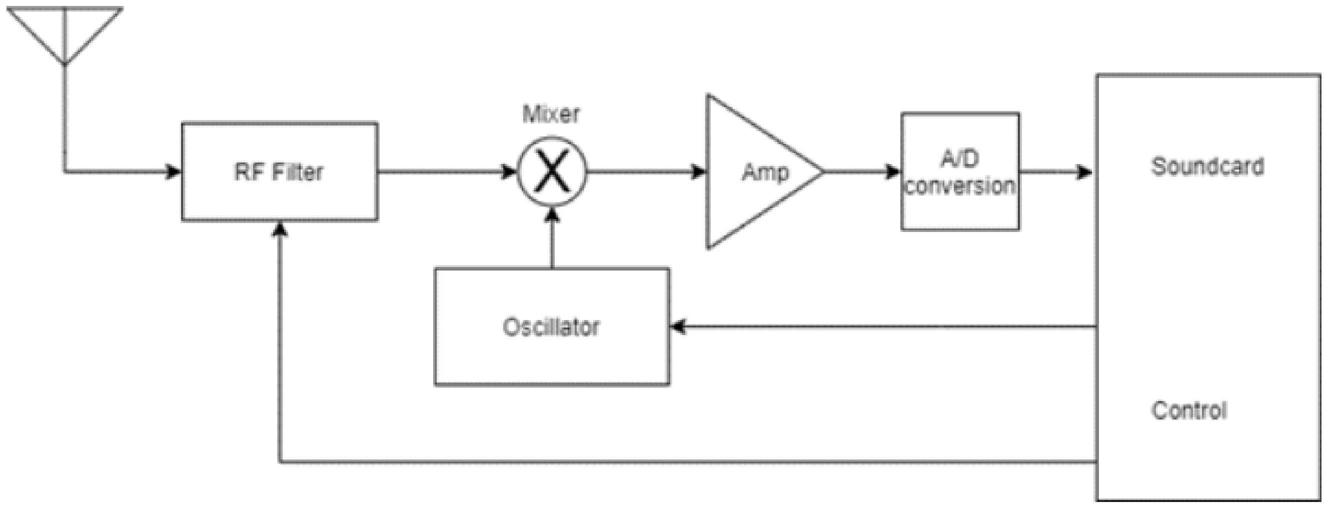
Software Defined Radio

Jared Evans and Nate Price

What is a Software Defined Radio (SDR)?

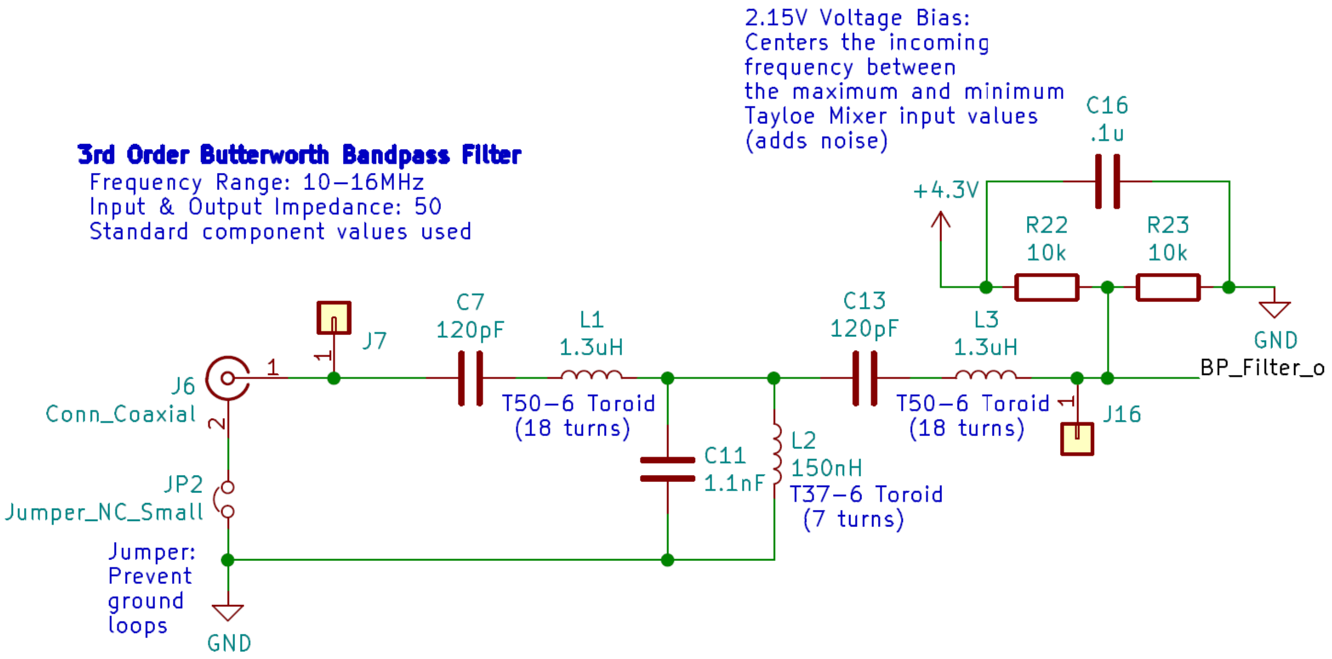
- It is a radio receiver controlled by software implemented using an Arduino in the case of this project.





Block Diagram

Bandpass Filter



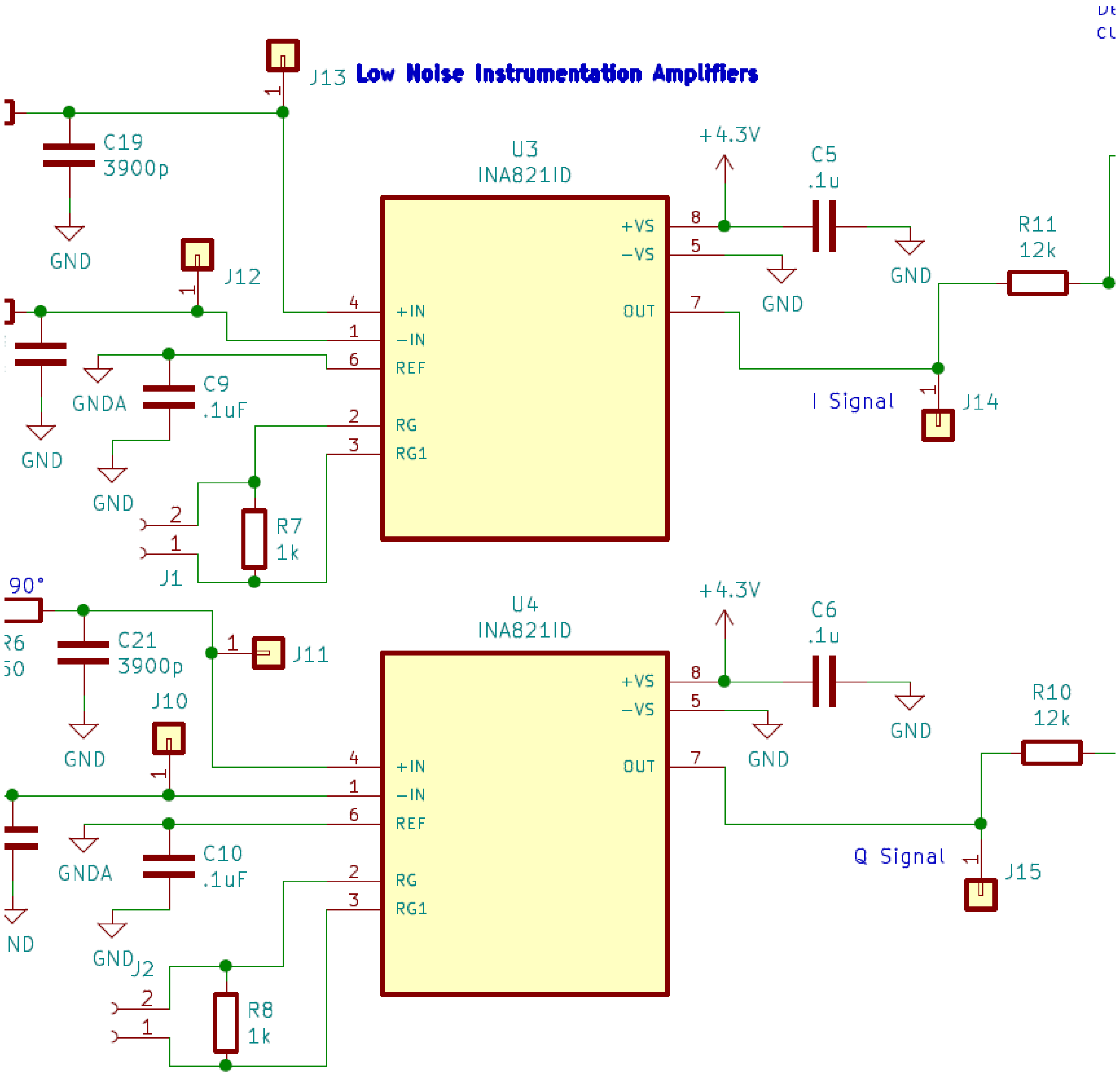
3rd Order Butterworth Bandpass Filter
Frequency Range: 10–16MHz
Input & Output Impedance: 50
Standard component values used

2.15V Voltage Bias:
Centers the incoming
frequency between
the maximum and minimum
Tayloe Mixer input values
(adds noise)

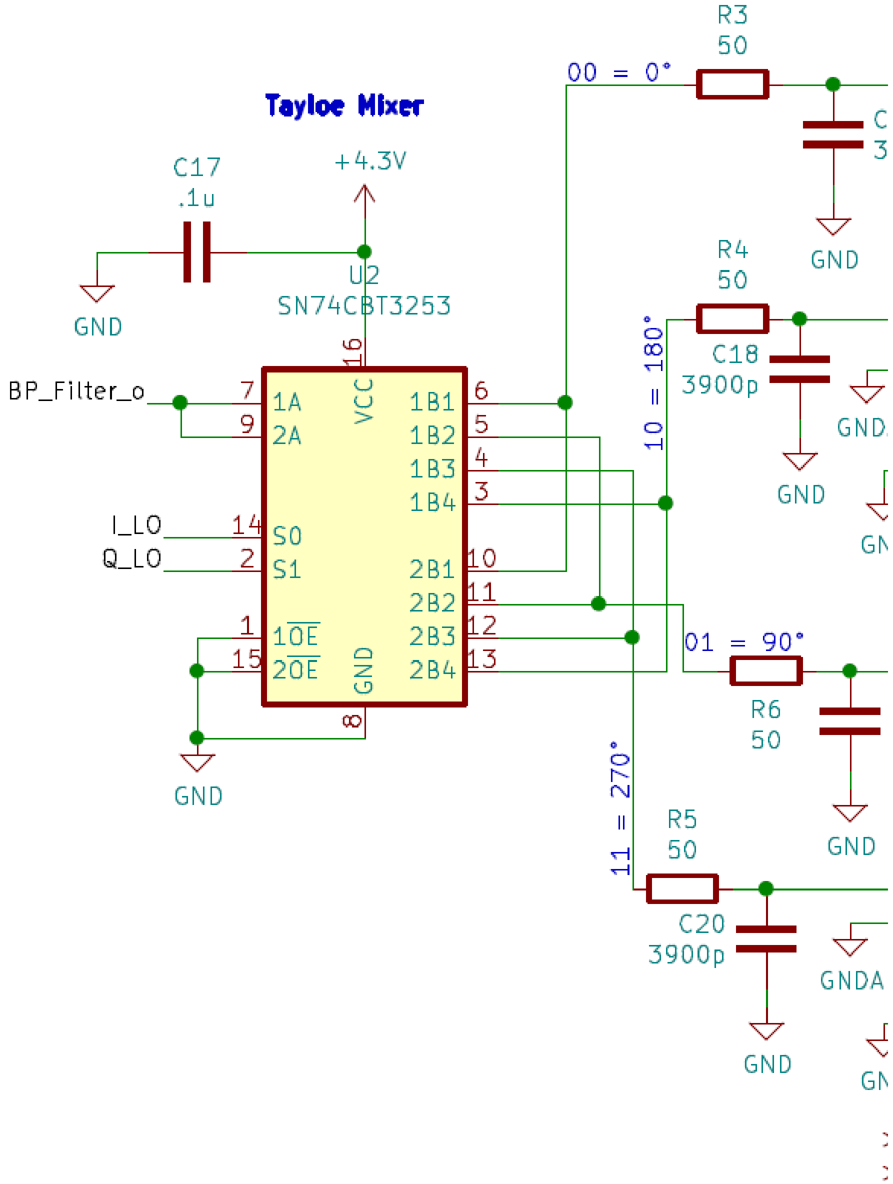
Jumper:
Prevent
ground
loops

- Frequency Range: 10-16 MHz
- Based on Caleb Froelich and Konrad McClure’s bandpass filter
- Values calculated using rf-tools.com

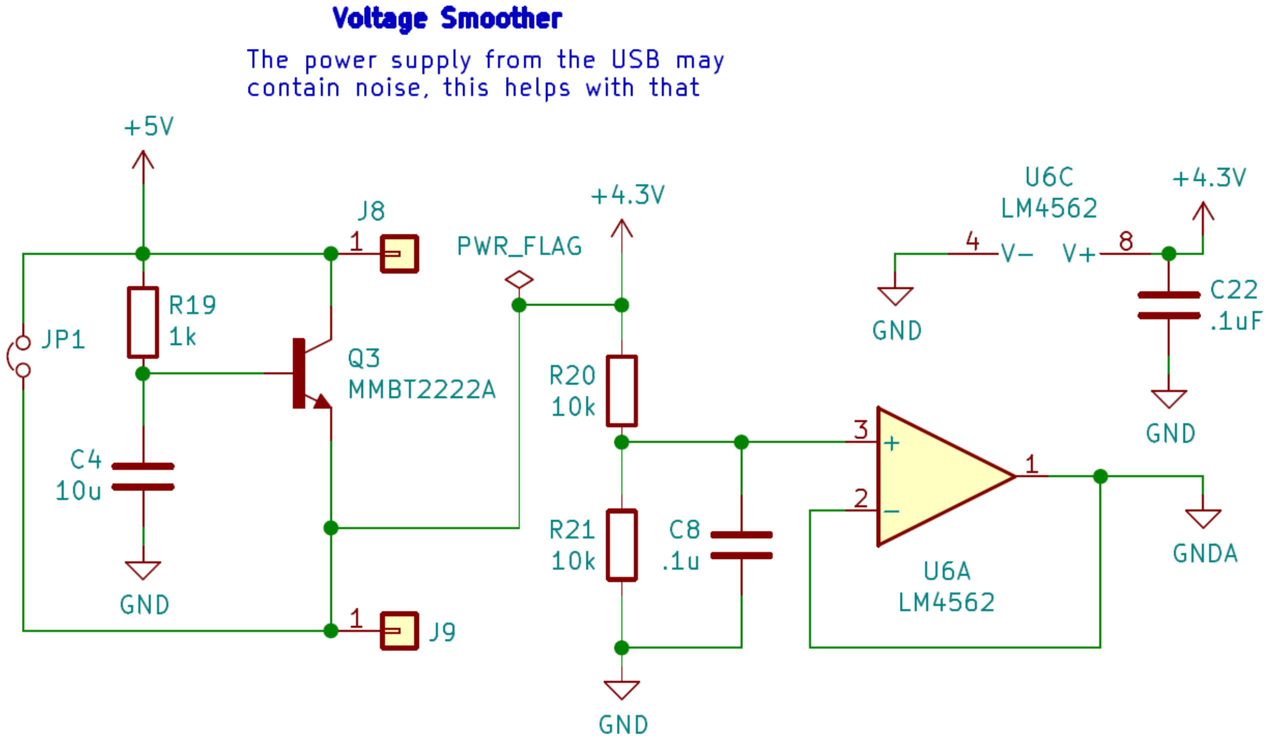
Instrumentation Amplifier



Tayloe Mixer

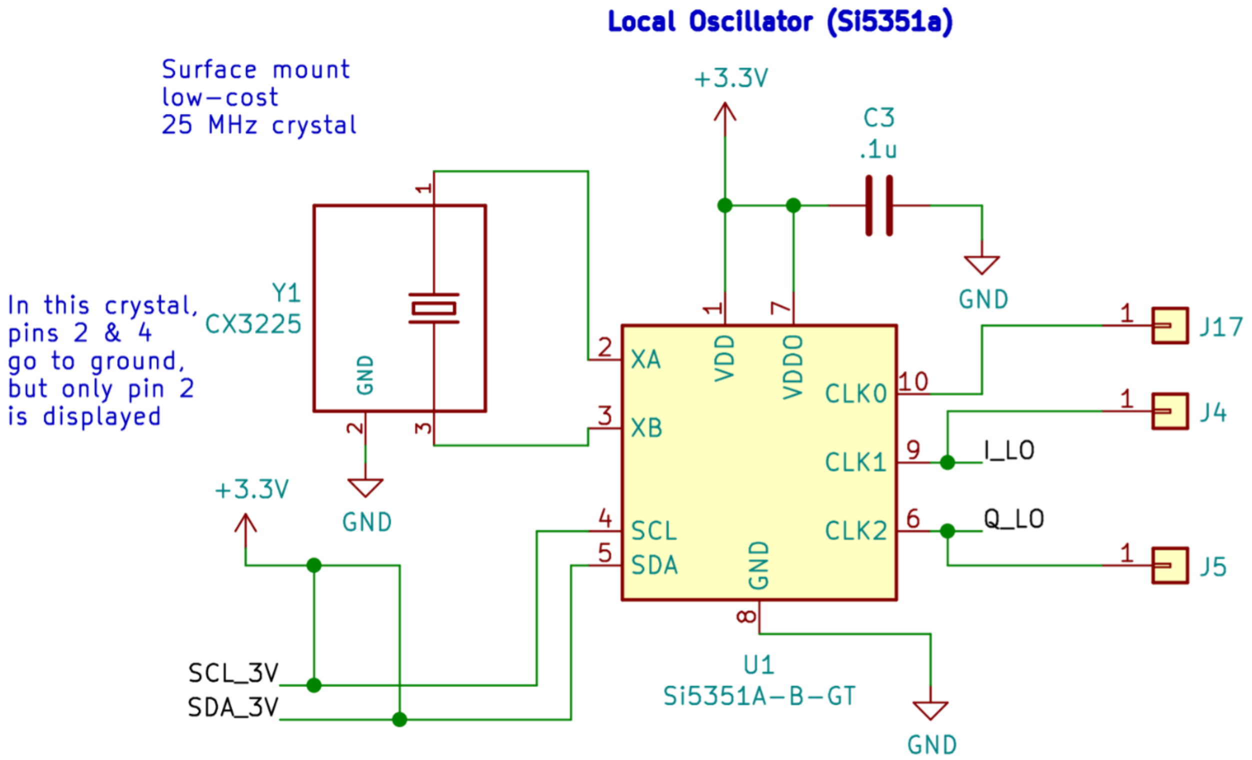


Voltage Smoother



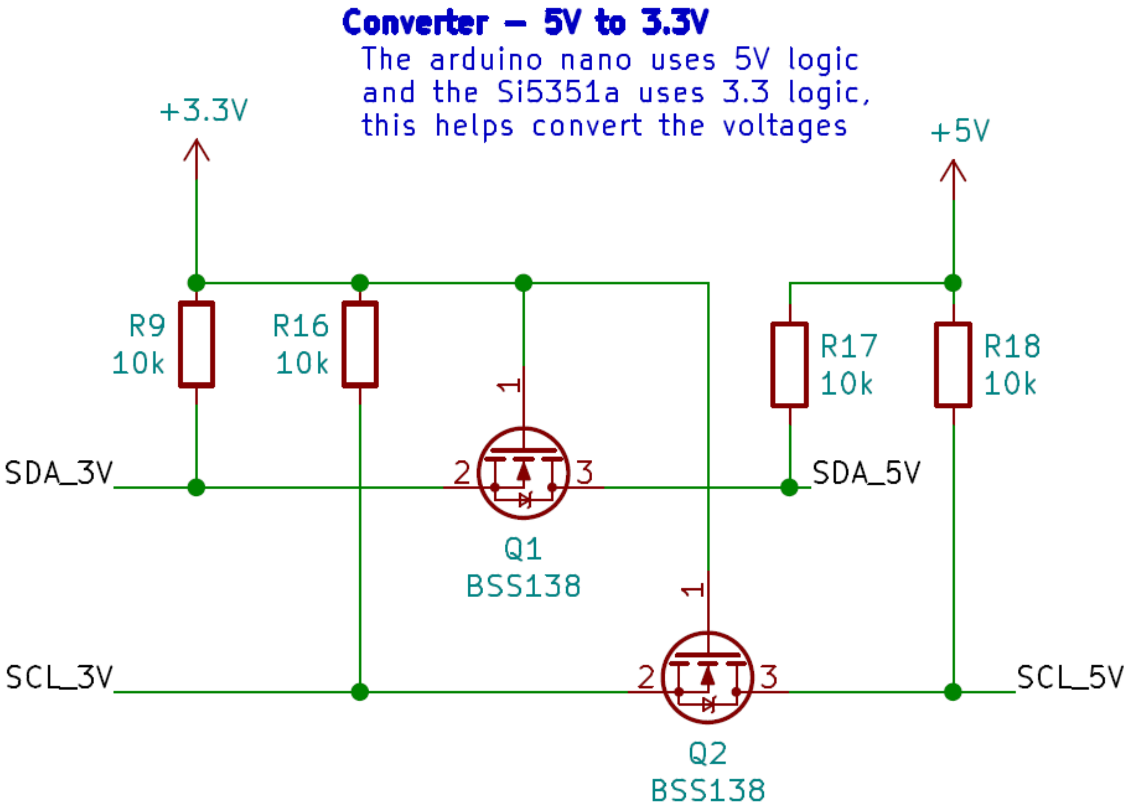
- Based on Caleb Froelich and Konrad McClure’s voltage smoother
- Modified to use an LM4562 op-amp

Local Oscillator



- Originally incorrectly used resistors between +3.3V and ground

Voltage Converter

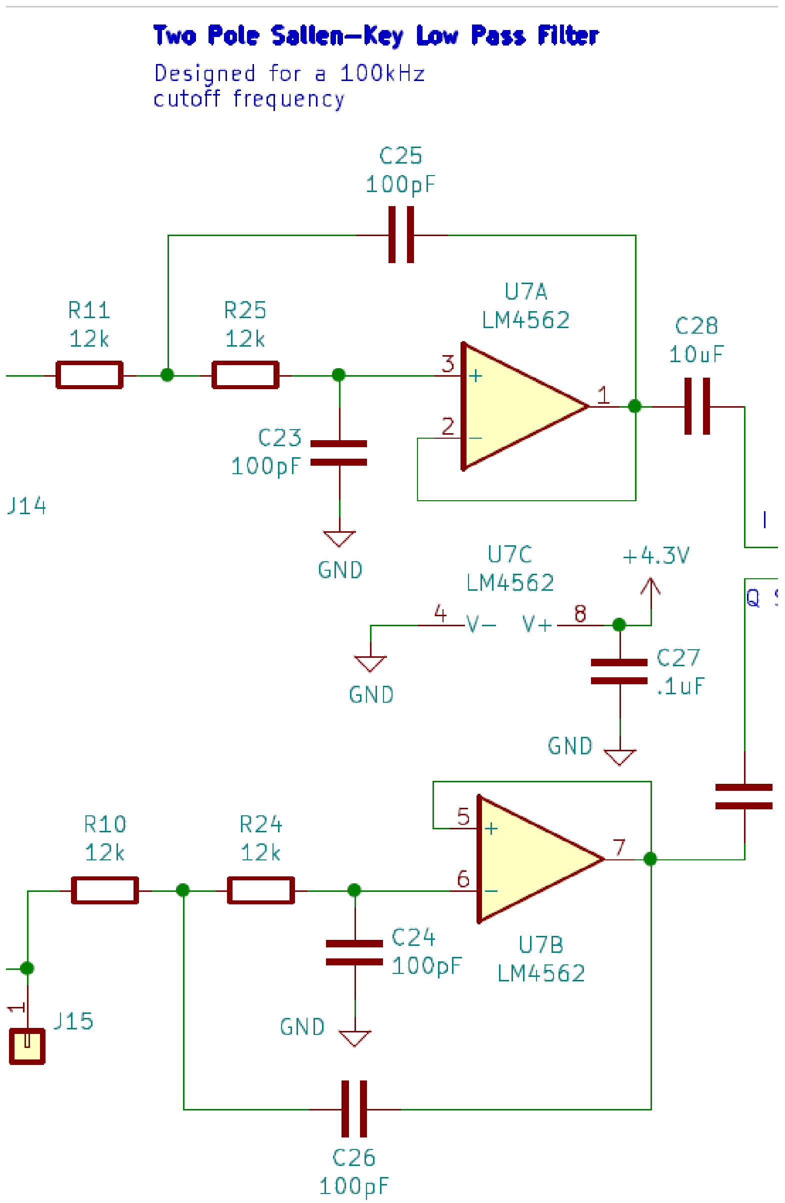


Converter – 5V to 3.3V

The arduino nano uses 5V logic
and the Si5351a uses 3.3 logic,
this helps convert the voltages

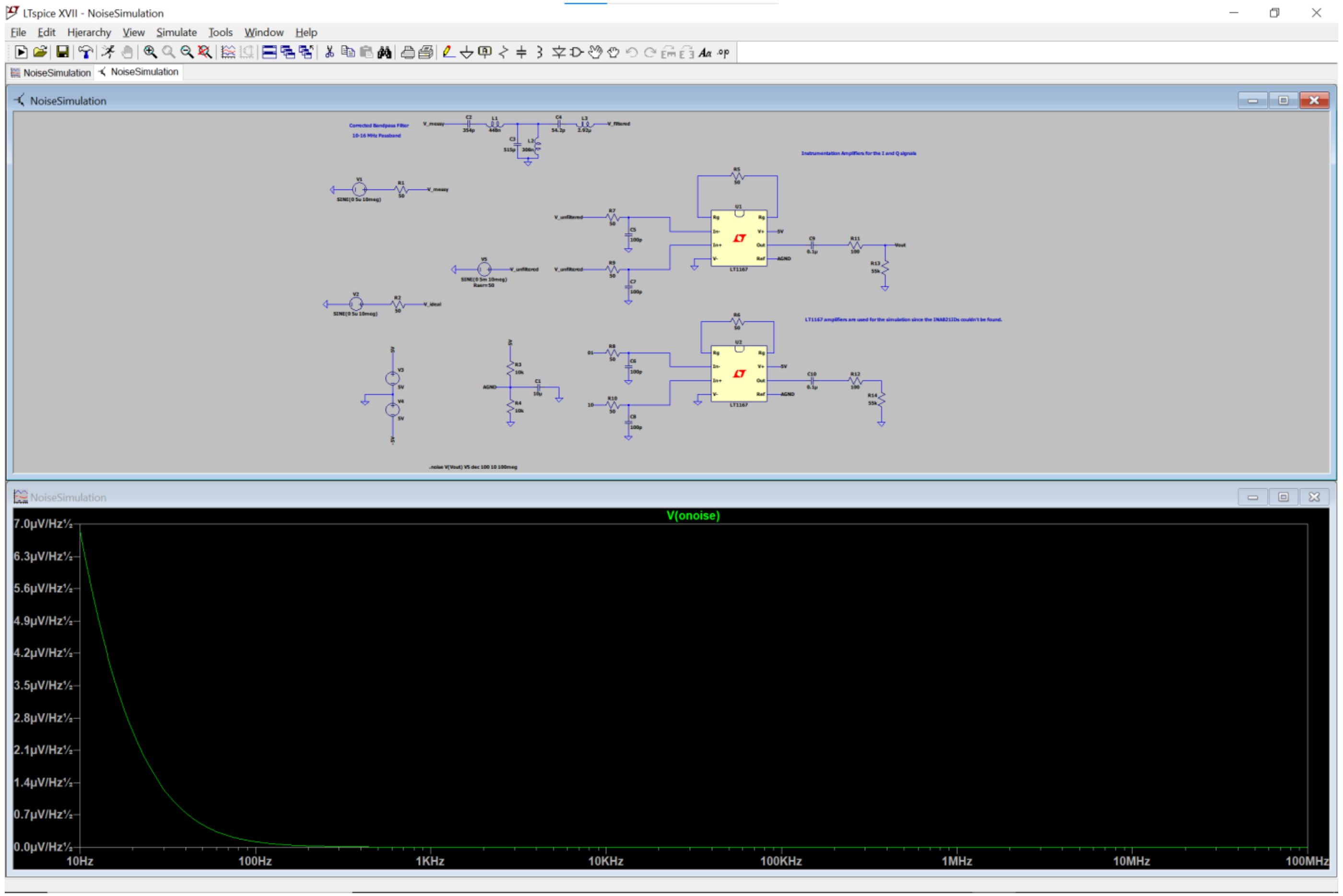
- Design comes from Adafruit
- Originally incorrectly tried to use fewer resistors

Lowpass Filter



- Design based on and values calculated using http://sim.okawa-denshi.jp/en/OPseikiLowkeisan.htm

Simulations



Corrected Bandpass Filter
10-16 MHz Passband

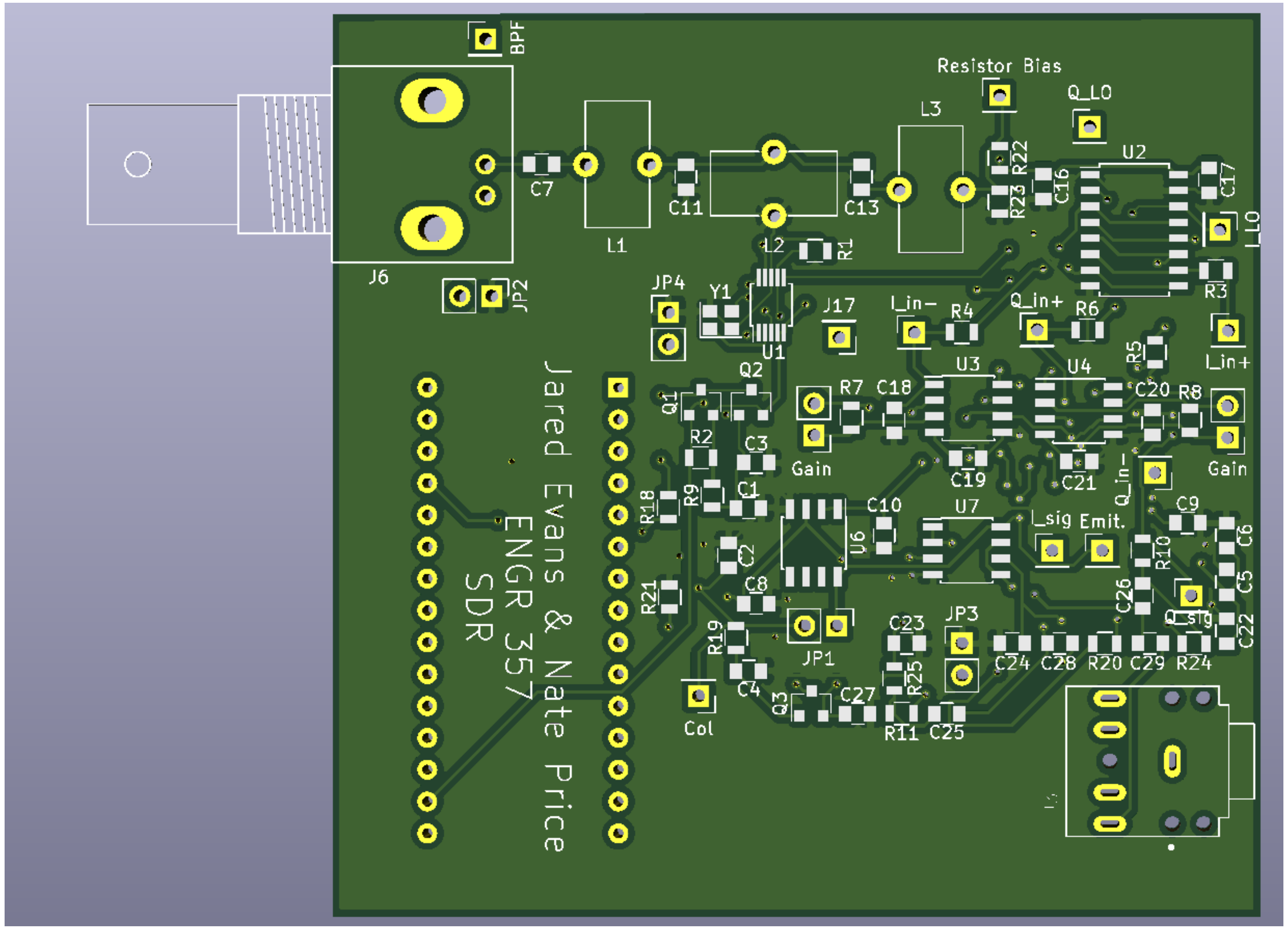
Instrumentation Amplifiers for the I and Q signals

LT1167 amplifiers are used for the simulation since the INA821IDs couldn't be found.

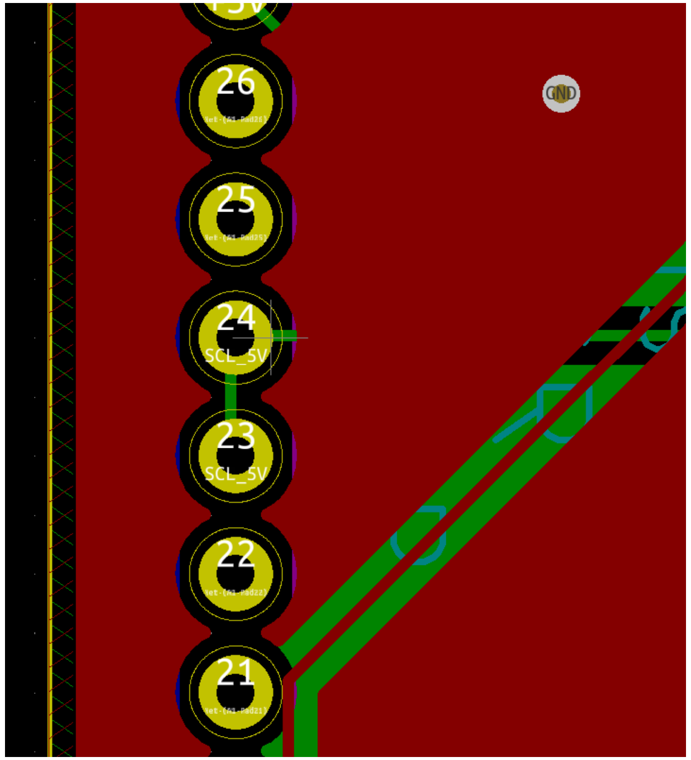
```
.noise V(Vout) V5 dec 100 10 100meg
```

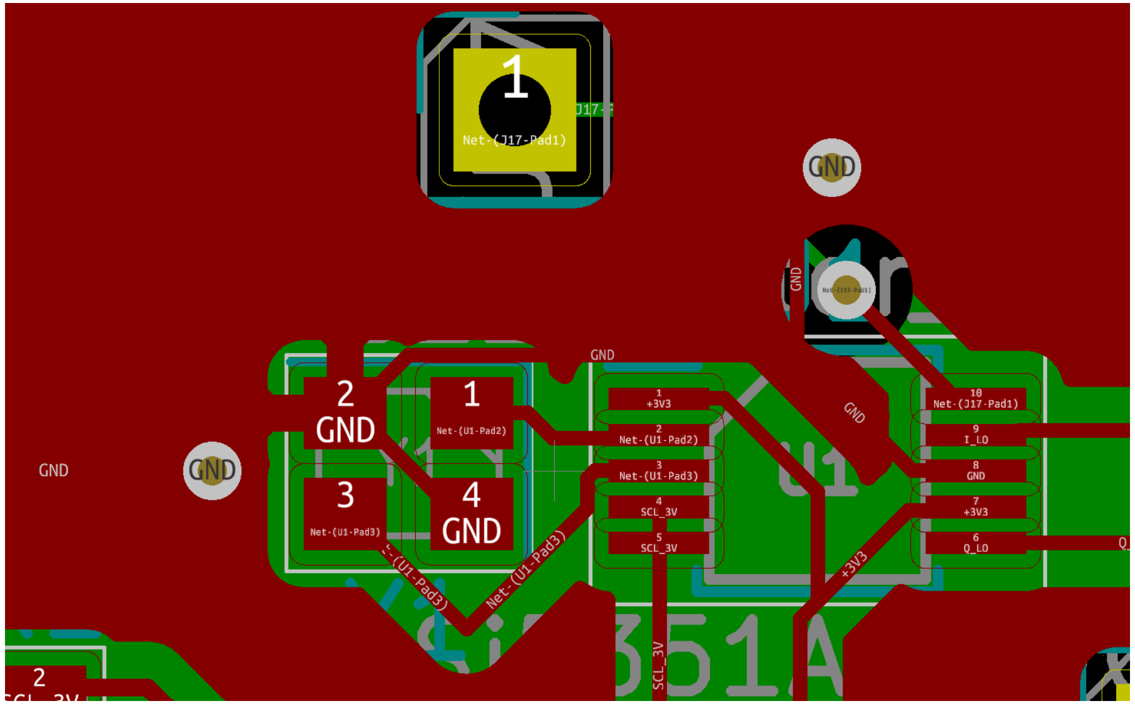
- This was one of our noise simulations

PCB



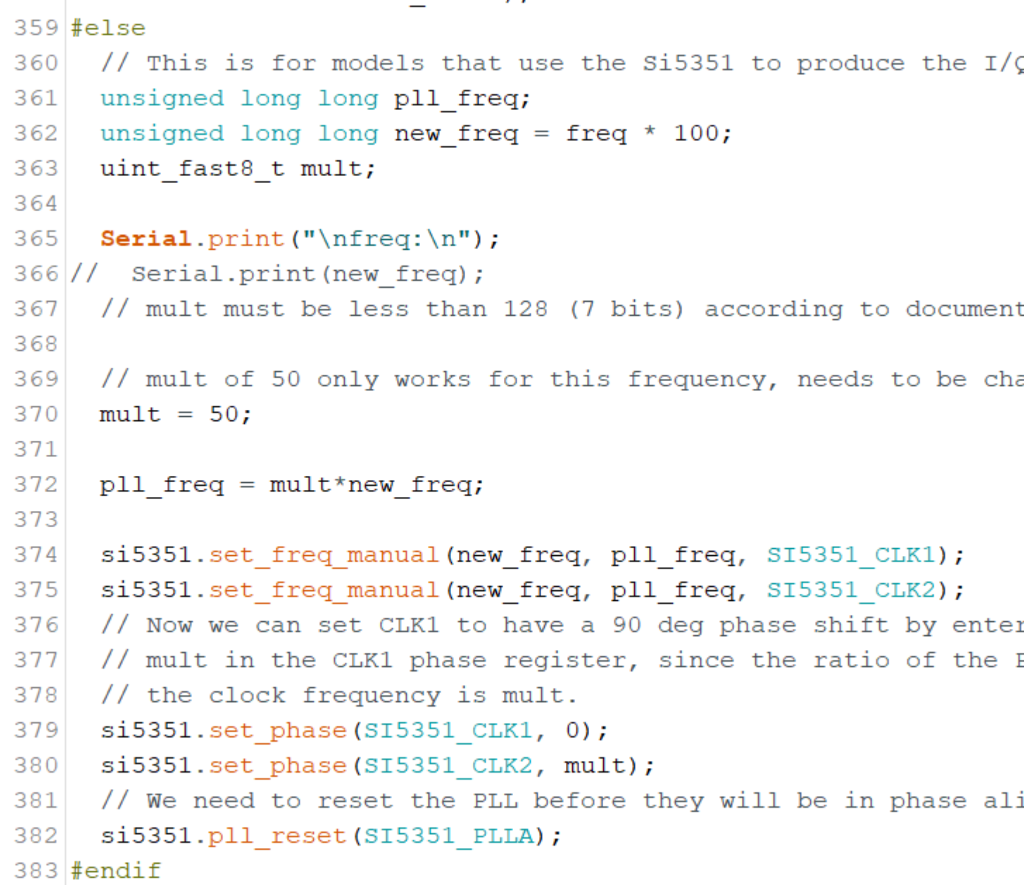
Tip: Check your PCB connections carefully!



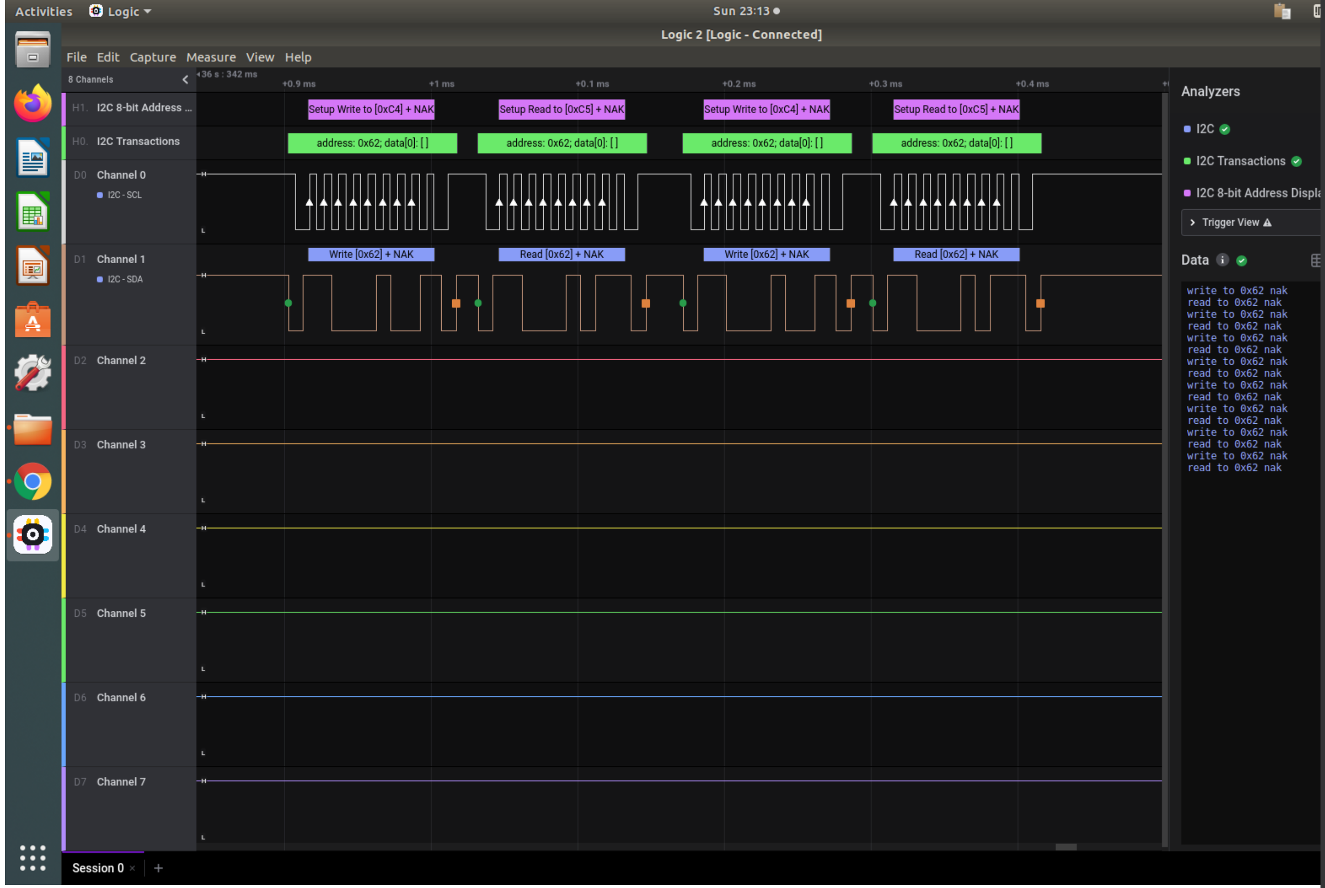


Errors Encountered

- The SCL and SDA pins on the Si5351 were shorted which led to several traces needing to be cut.
- Another trace was cut but shouldn't have been.
- Several THT components are externally connected to correct these mistakes.



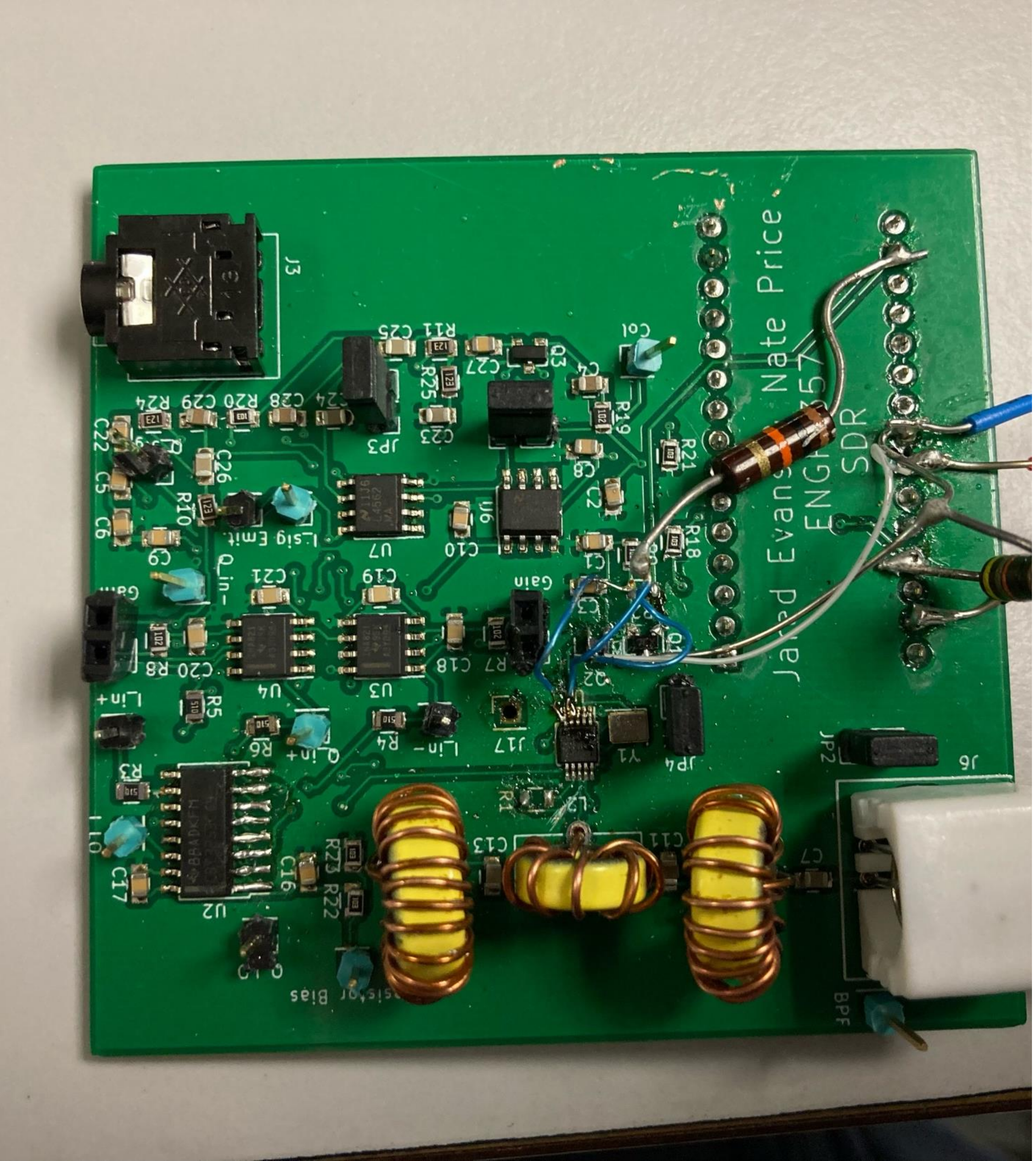
```
#else
  // This is for models that use the Si5351 to produce the I/Q
  unsigned long long pll_freq;
  unsigned long long new_freq = freq * 100;
  uint_fast8_t mult;

  Serial.print("\nfreq:\n");
//  Serial.print(new_freq);
  // mult must be less than 128 (7 bits) according to document

  // mult of 50 only works for this frequency, needs to be cha
  mult = 50;

  pll_freq = mult*new_freq;

  si5351.set_freq_manual(new_freq, pll_freq, SI5351_CLK1);
  si5351.set_freq_manual(new_freq, pll_freq, SI5351_CLK2);
  // Now we can set CLK1 to have a 90 deg phase shift by enter
  // mult in the CLK1 phase register, since the ratio of the P
  // the clock frequency is mult.
  si5351.set_phase(SI5351_CLK1, 0);
  si5351.set_phase(SI5351_CLK2, mult);
  // We need to reset the PLL before they will be in phase ali
  si5351.pll_reset(SI5351_PLLA);
#endif
```



```
write to 0x62 nak
read to 0x62 nak
write to 0x62 nak
read to 0x62 nak
write to 0x62 nak
read to 0x62 nak
write to 0x62 nak
read to 0x62 nak
write to 0x62 nak
read to 0x62 nak
write to 0x62 nak
read to 0x62 nak
write to 0x62 nak
read to 0x62 nak
write to 0x62 nak
read to 0x62 nak
```

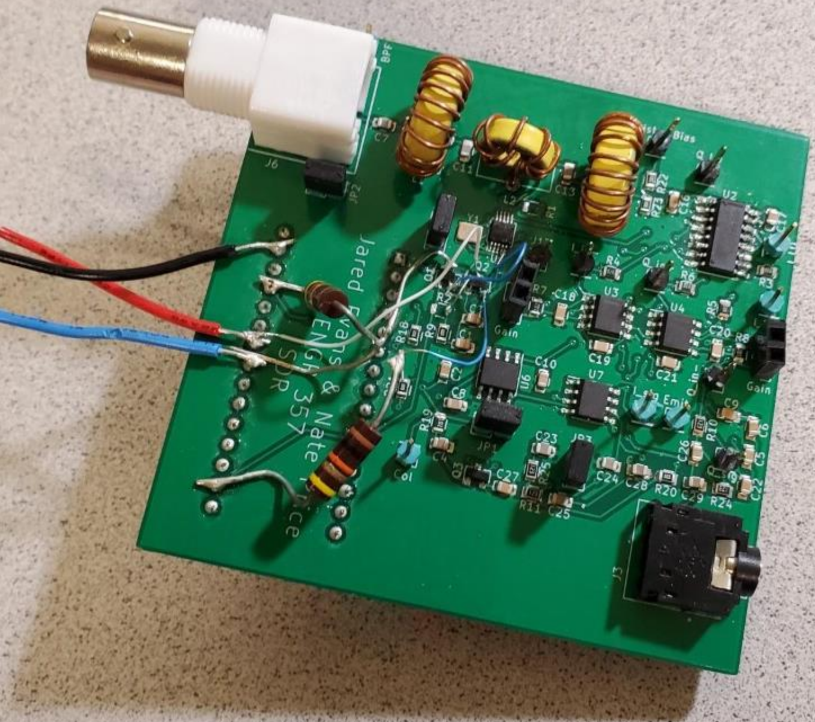
Software Issues

- Understanding how the use the logic analyzer
- Making sure the correct I2C address was used/acknowledged
- Quisk configuration settings
- Frequency used to set the phase with the Si5351Arduino library needed to be multiplied by 100
- Some code needed to be reconfigured to work with clocks 1&2 instead of clocks 0&1



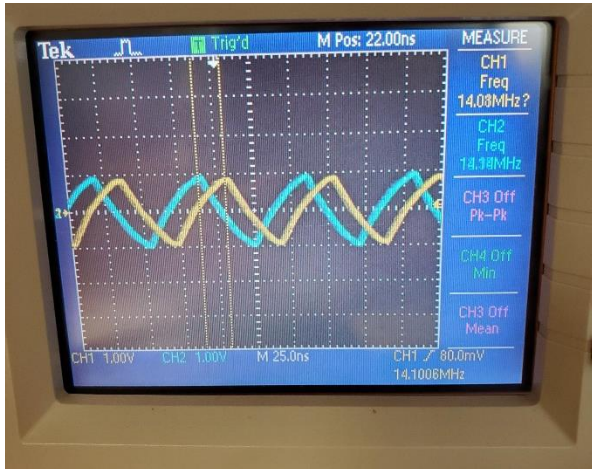
Results – Nate's Board

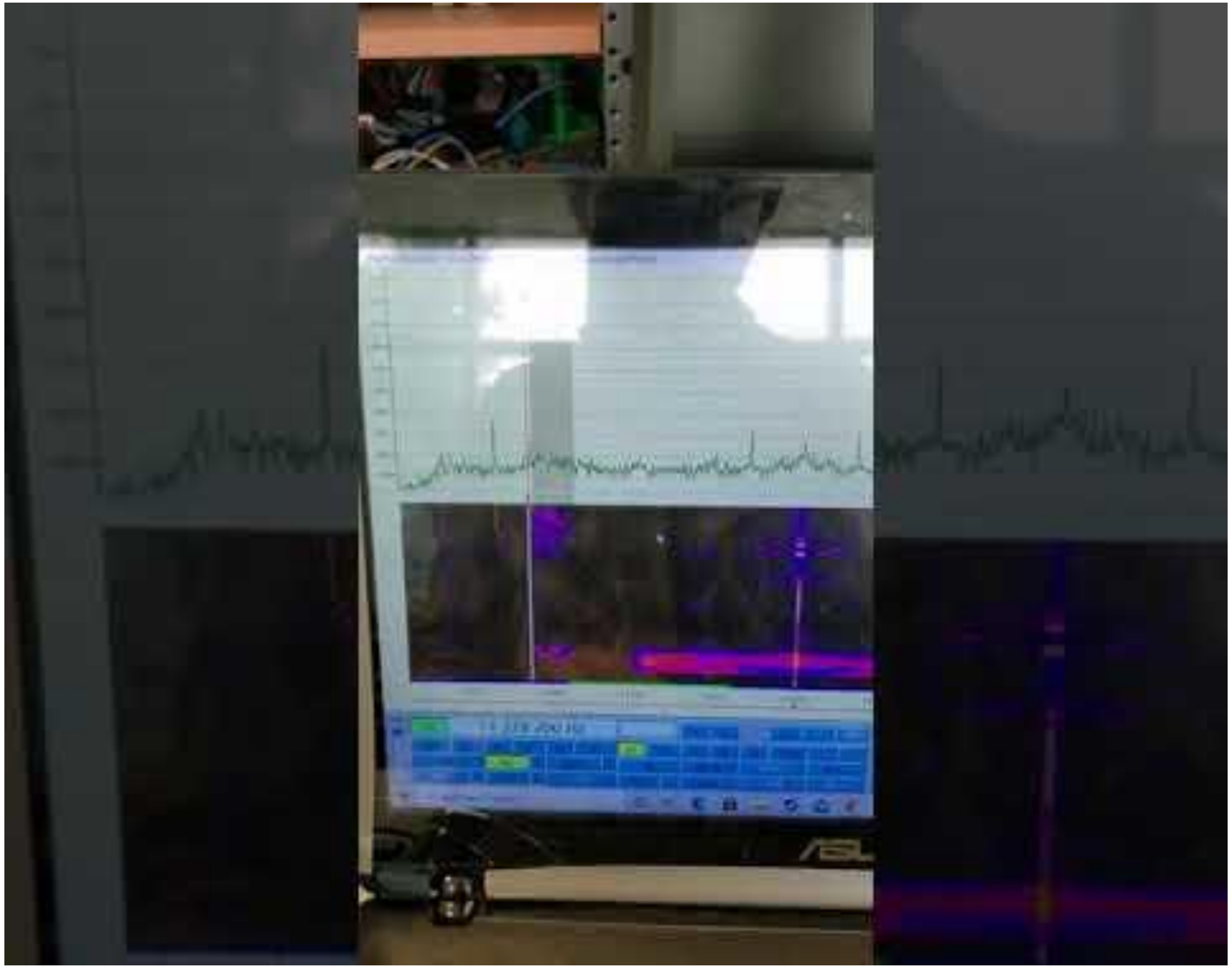
- I2C address 0x62 "nak"
- Would need to replace the FET where many soldering fixes were made
- Best ways to fix it:
  - Design a new PCB, have it made, solder everything again
  - Use one of the extra broken PCB's to start again, knowing which fixes are needed
- Either way requires a lot of time desoldering and then soldering again
- Ultimately decided to make sure Jared's board worked



Results – Jared's Board

- Si5351 communicates with Arduino
- Able to set phases
- Really good sensitivity (< 0.1 uV)
- Quisk able to pick up signal from the antenna





Video

It's a bit difficult to see on the video, but the frequency running when this video was being recorded was 14.2787 MHz. Also, while it's a little hard to hear, we could hear someone who sounded like he was asking about practicing law in a "public sense".

Lessons Learned

Put in the effort into understanding circuits designed by others

Check the PCB (and everything else) carefully!

Collaborate as much as possible

Get as much software as possible configured early

Questions