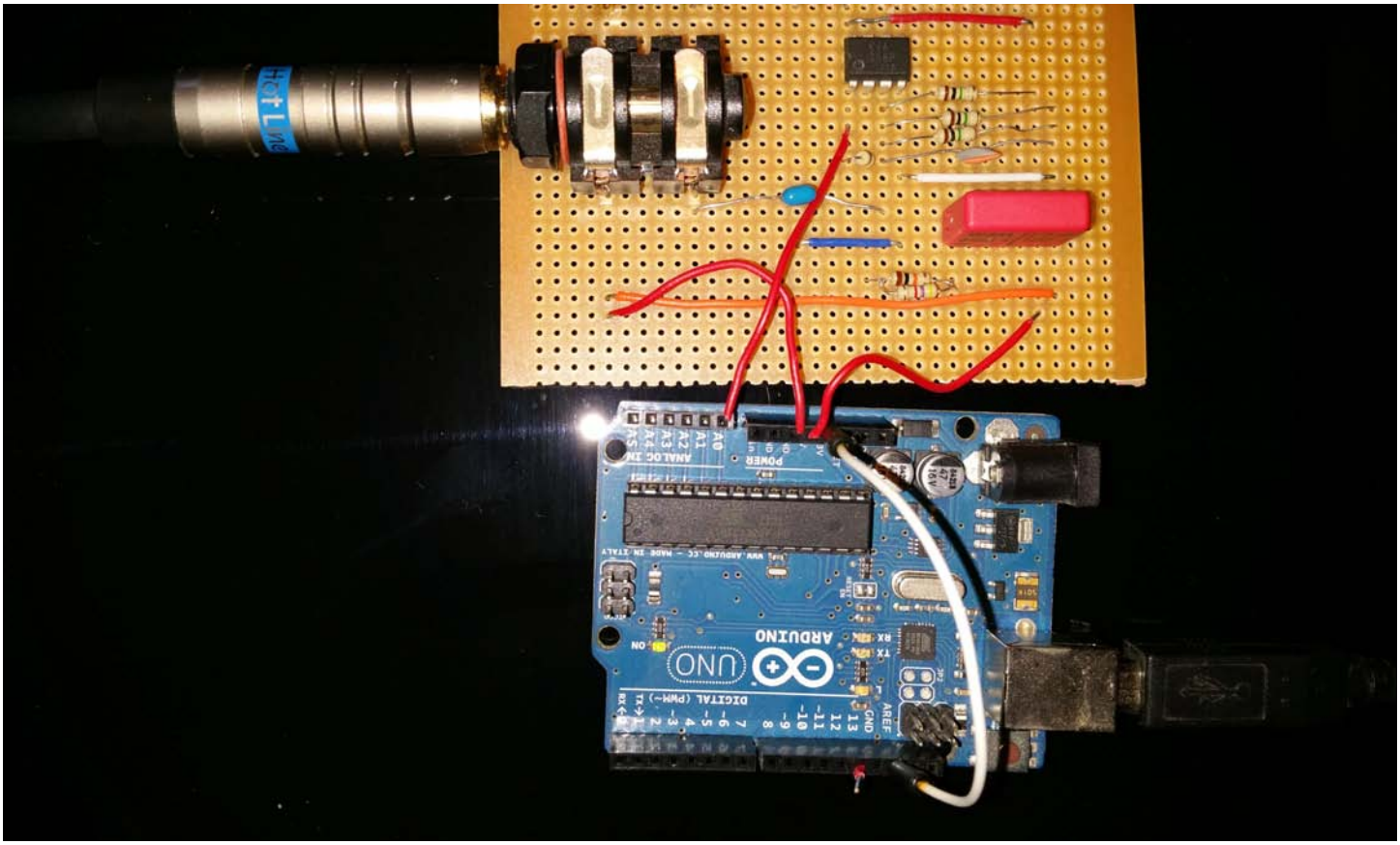
# TECH STORIES

This is a blog for people interested in technology. I'll be posting projects and tutorials on things such as Arduino, IoT devices, Electronics, Machine Learning and whatever springs to mind.




**ARDUINO GUITAR TUNER**

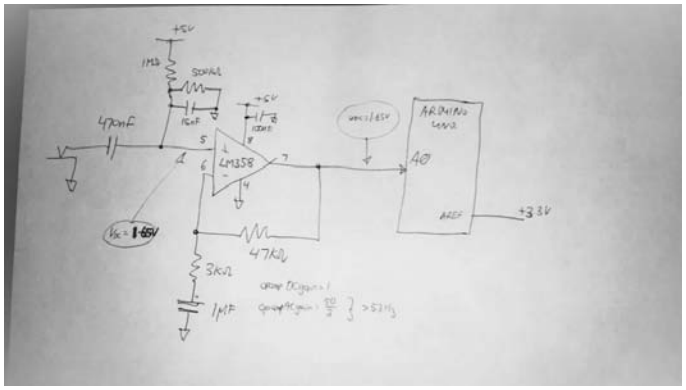
We're going to use the techniques covered in my  Reliable Frequency Detection Using DSP Techniques article to build a guitar tuner.

We'll need to build an input amplifier. The input amplifier will amplify the AC signal from the guitar and provide a DC offset to the Arduino. The reason for the DC offset is so the Arduino ADC readings i.e. analogRead(A0), sit about half way up the ADC range, leaving room for the AC from the amplifier to move the readings up and down.

## You Will Need:

- Arduino (Uno preferrably)
- Electronic components:
  - LM358 op-amp
  - resistors : 47k, 3k, 3* 1M,
  - capacitors: 1uF,  470nF,  15nF,  100nF
  - Phono-Socket (for a guitar jack)
  - Veroboard, wires and solder

## Here's the Circuit




The circuit is shown above.

The op-amp is 1/2 of an LM358 dual op-amp. This is a good choice because it operates off a single 5V supply, many op-amps require a dual-supply.

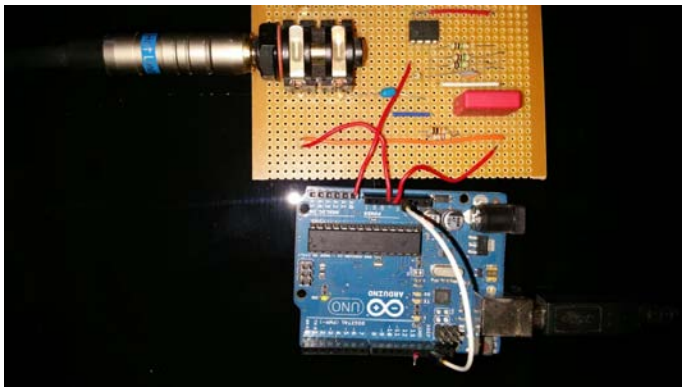
The op-amp is wired up as a non-inverting amplifier with a gain of 16 to AC (f> 53Hz) and a gain of 1 to DC.

The 1M / 500k resistor network sets the '+' input to 1.65V and the DC gain of the op-amp is 1, so the output voltage sits at 1.65V

The 470nF capacitor in series with the guitar input blocks the 1.65V from reaching the guitar by forming a high-pass filter with the 1M/500k resistors; passing signals at f> 1Hz.

Also, the Arduino AREF is connected to 3.3V as we're using the External Vref for the ADC. This puts Vref/2 at 1.65V, which is the voltage we've DC biased the A0 input to, so the ADC will be at 1/2 its range when there is no signal.

Build the circuit on a breadboard or veroboard as shown below. Please note: if you're not using the other half of the LM358 wire pins 2, 3 to ground.



## The Code

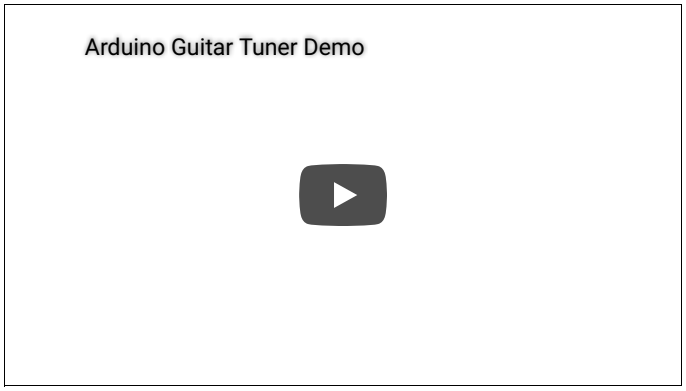
The code is based on the Reliable Frequency Detection Using DSP Techniques project, but also filling the buffer with readings from the ADC.

You can find it on Github here.

At the moment the code prints out the detected frequency to the Serial Monitor, but it wouldn't be too difficult to add some LEDs to indicate the note and whether the tuning is flat, sharp, or in tune. Or alternatively, you could add an LCD display.

## Demo

And finally, this is how it should work.




Some Thoughts

Notice how the code calculates the frequency:

```
freq_per = sample_freq/period;
```

I measured the sample frequency to be 8919Hz. So for a low E (E2) you'd get a period count of 108 or 109 which gives a frequency difference of 0.7Hz but at high E (E4) the period would be 27 or 28 which is a frequency difference of 12Hz, so it won't be as easy to get fine tunings on higher strings.