A Cortex M3 Guitar Tuner

Chris J Arges
<christopherarges@gmail.com>

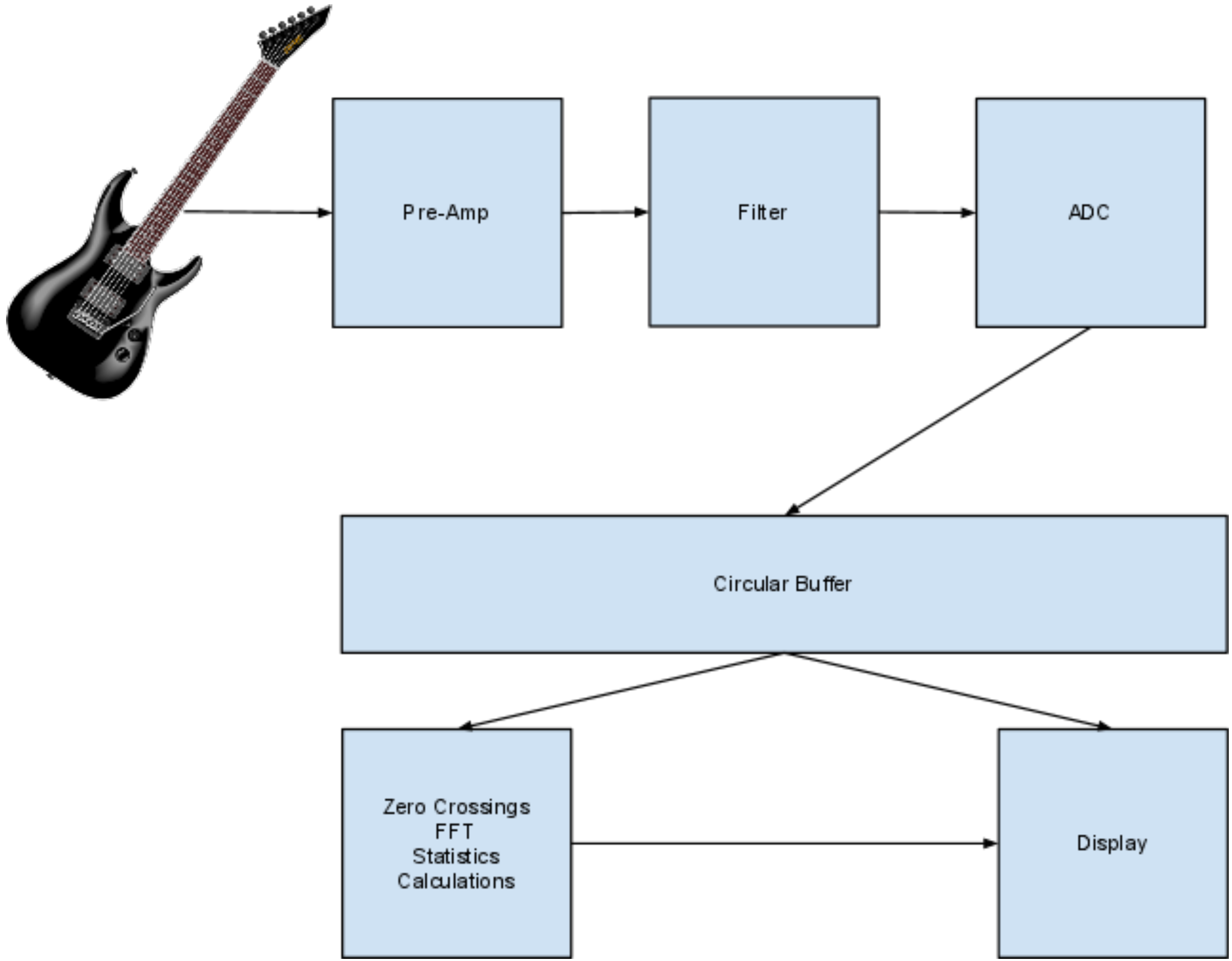
Guitar Tuners

- Guitars needs to be tuned often
- Involves increasing and decreasing tension in strings
- Can plug into a tuner which tells if notes are flat or sharp
- Need to respond quickly to a plucked note

Requirements

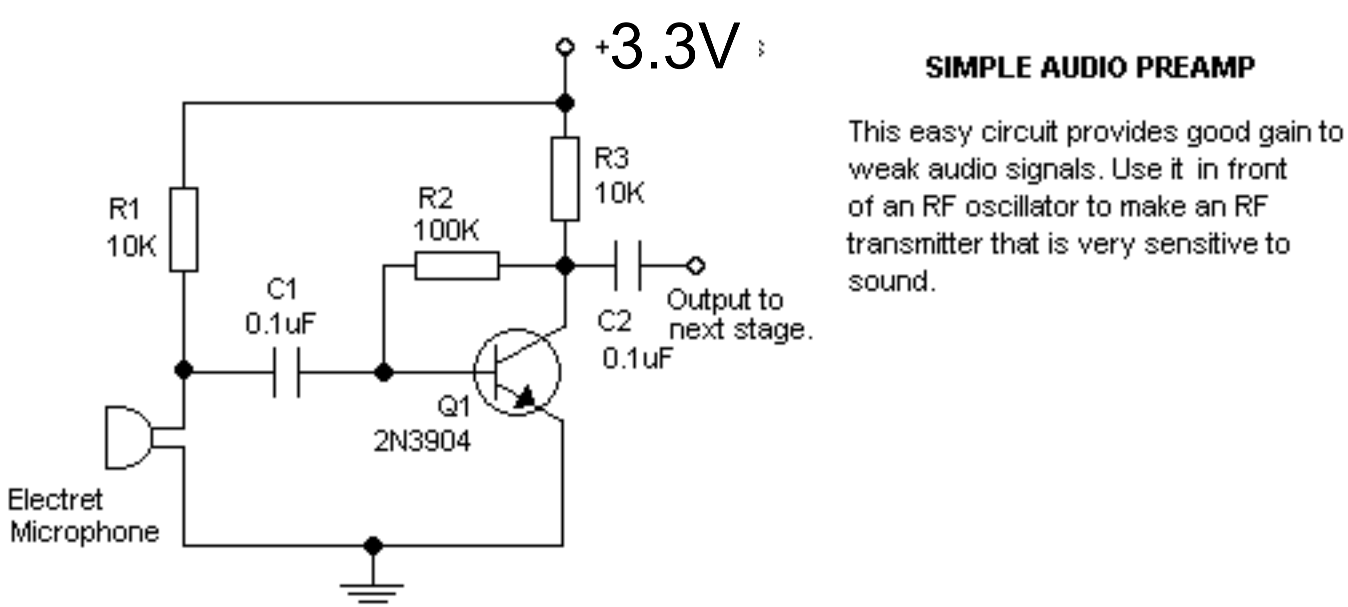
- Inputs: Guitar signal, Buttons
- Output: OLED Display
- Functions: Tuner, Waveform display, FFT display
- Performance: Fast enough to tune guitar, sampling freq. high enough for all guitar strings.
- Multi-rate System
    - ADC Sampling - Hard Deadline
    - Display results on OLED - Soft Deadline
- Respond to user input of buttons
    - Asynchronous
- Amplify guitar signal and digitize
    - Must be done in hardware before sampling
- Detect if a guitar string is in tune
    - Calculations must be done between displaying results

Guitar Tuner Architecture



Guitar->Preamp

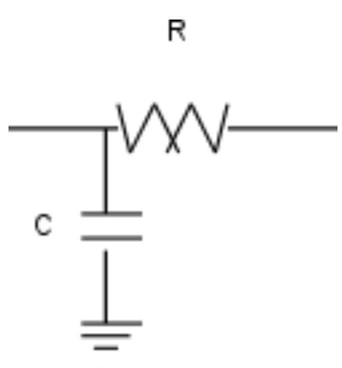
- Vibrating strings induce voltage on magnetic coils (pickups)
- Voltage is plugged into an amplifier
- Voltage [1]
  - 100mV - 1V RMS
- Fundamental String Frequencies [2]
  - 82 Hz, 110 Hz, 147 Hz, 196 Hz, 247 Hz, 330 Hz
- Need to amplify signal before going into ADC!
- Preamp boosts signal such that it can be digitized by ADC
- Used this circuit [3]:



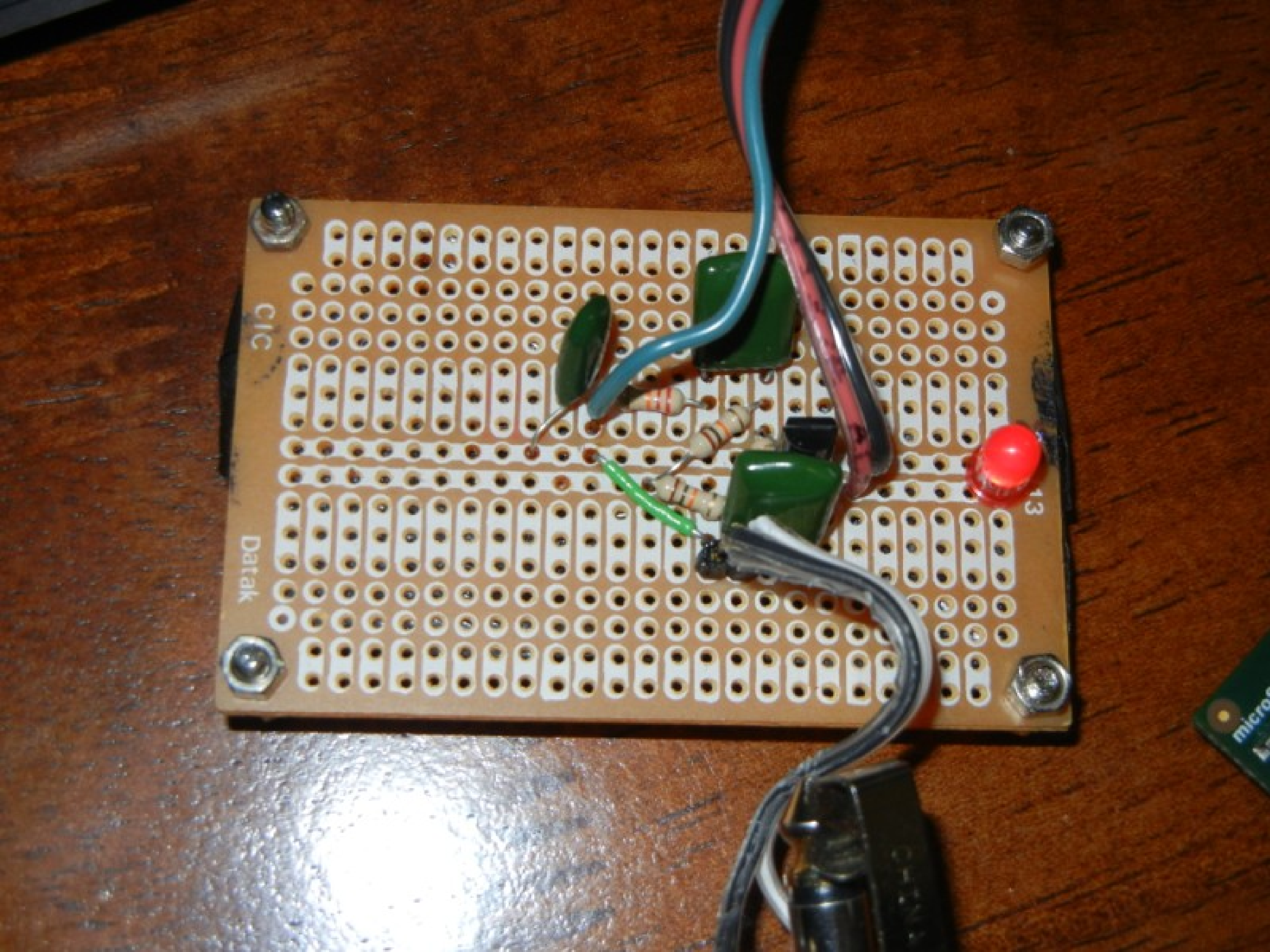
Filter

Needed to filter for interference on audio line
Since we only need 330 Hz, can design filter for this.
I had R = 22kOhm and C = 0.001 uF
Thus, f_c = 8376Hz, should be good enough.

$$fc = \frac{1}{2\pi RC} \, Hz$$ <--Low-Pass Butterworth Filter Equation [7]



Add filter to output of pre-amp stage.

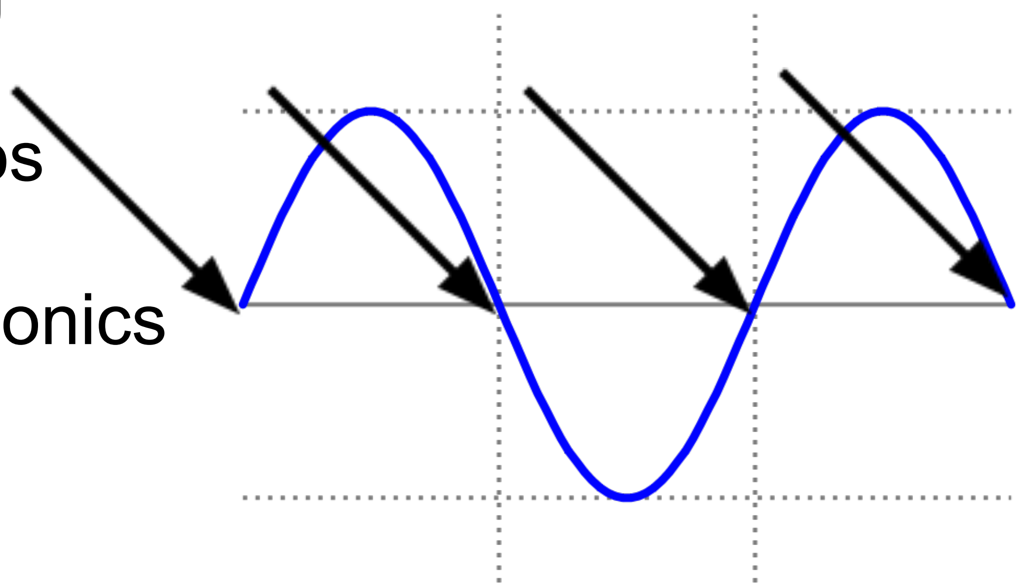

CIC
Datak

ADC

- Can input voltages from 0 to 3V
- Outputs a 10-bit value from 0x000 to 0x3FF [4]
- Nyquist Theorem [5]
  - Need to sample at twice the highest frequency
  - Highest frequency = 330Hz
- Oversampling
  - Sampling isn't that accurate, thus sample many times and get an average
  - I've set this for 8x oversampling
- Timer
  - We use a timer to drive the ADC
  - ADC Interrupts when 8 samples have been averaged and ready to read
- Thus we set our timer for at least 330 * 2 * 8 = 5280 Hz

Tuning Calculation

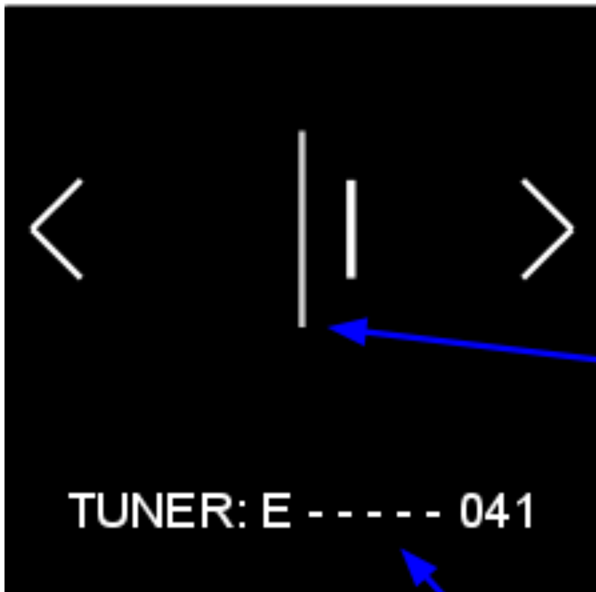
- Multiple ways to handle this
- I chose to count 'zero-crossings', whenever a transition from previous to next signal crosses 'zero'
- Zero is defined as the rolling average of the signal since we range from 0x000 to 0x3fff, and because there is some bias in the signal
- Use a reference signal to determine the number of zero crossings needed for a string
- Roughly
  - higher pitch == more zeros
  - lower pitch == less zeros
  - Need to account for harmonics
  - Filtering would help



FFT

- Transforms the time domain signal into a frequency domain signal.
- Needed fixed point arithmetic and no trig functions, therefore a lookup table is needed.
- Tried to write this in C myself and failed miserably.
- Found an effecient 128 point function written in cortex m3 assembly.
- 128 point not good enough for tuning, can't see enough variation in the maximum bin.

Display - Tuner Mode

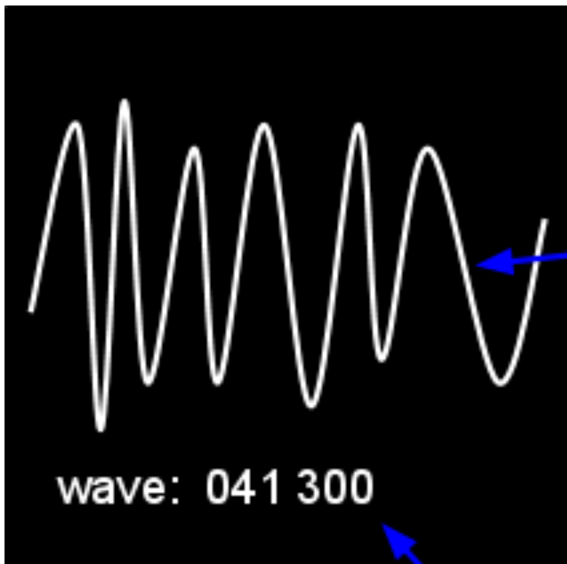


Arrows indicate if pitch is higher or lower

Center line is 'in-tune', thicker line is the current reading. Want to get these matching.

Need to select the current string (E,a,d,g,b,e) with Left and Right buttons. The number shows the zero crossings.

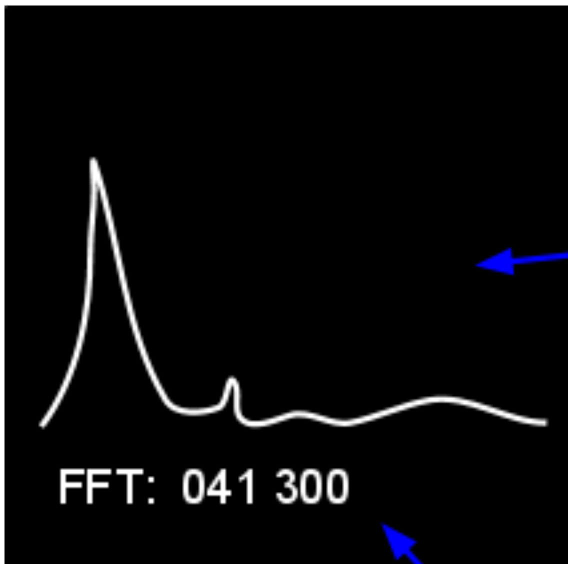
Display - Waveform Mode



This display shows the contents of the circular buffer at a particular interval (FPS rate)

shows zero crossings and difference between max and min values for a period of time (used for threshold calculations)

Display - FFT Mode



This display shows the contents of the FFT transformation.

shows a count and bin with maximum value (not the 0th bin)


microSD CARD
LUMINARY MICRO
LM3S8962 EVALUATION BOARD
HanRun
TUNER E
000
DEBUG
RESET
STATUS
SELECT

Demo

If I had more time...

- Better pre-amp
  - Need more gain and make it less noisy
- Filtering per string (Digital)
- Better resolution of tuner, faster response
- Better UI / User Feedback

References

[1]http://buildyourguitar.com/resources/lemme/
[2] http://ffden-2.phys.uaf.edu/211.web.stuff/billington/main.htm
[3] http://www.reconnsworld.com/forum/read.php?9,10
[4] TI Stellaris Documentation (spms001f.pdf)
[5] http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem
[6] http://www.precisionstrobe.com/apps/guitarharm/guitarharm.html
[7]http://www.electronics-tutorials.ws/filter/filter_8.html

Credits:
Cover slide photo (Creative Commons) by a_roadbiker http://www.fotopedia.com/items/flickr-4620748466
Sine wave: http://upload.wikimedia.org/wikipedia/commons/thumb/0/02/Simple_sine_wave.svg/800px-Simple_sine_wave.svg.png