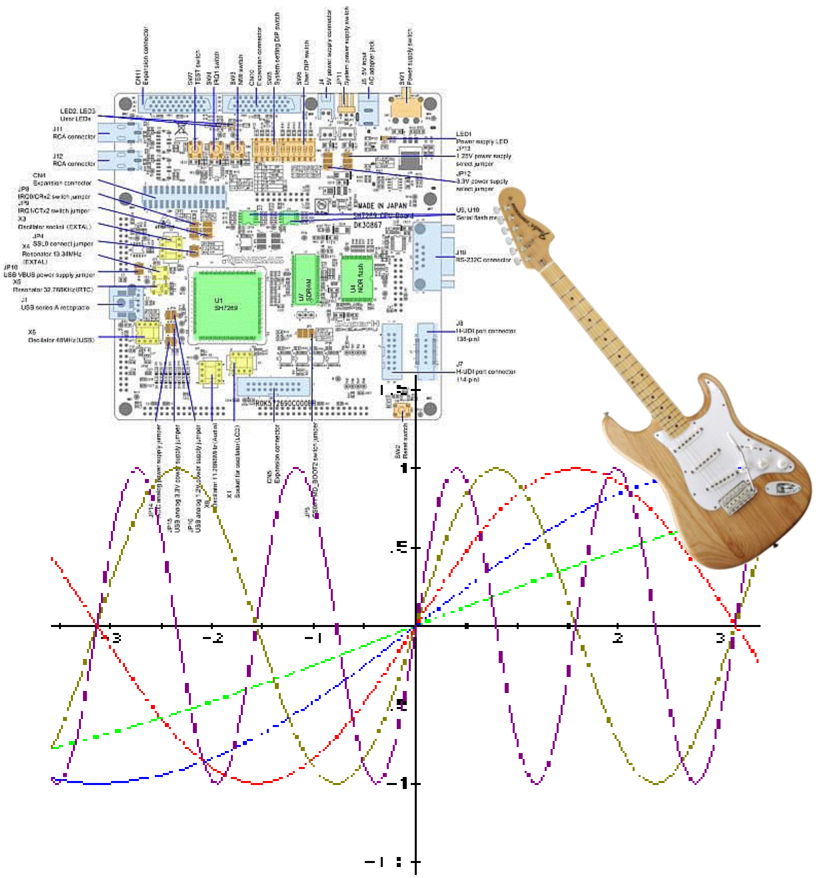
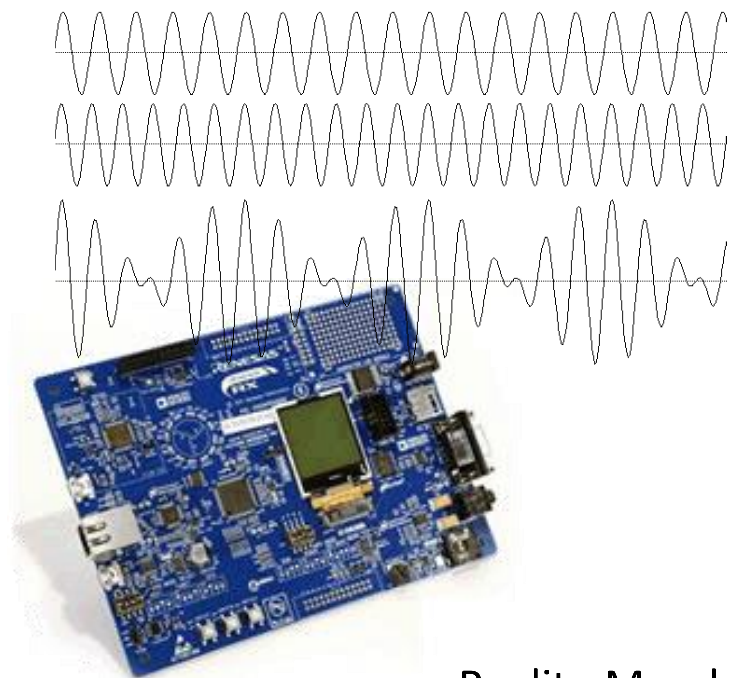
Real-Time Frequency Estimation and Tracking

aka “Let’s Build a Guitar Tuner”





Paulito Mendoza
github.com/Paulito-M

Agenda

- Overview
- The Approach
- Algorithms
- MATLAB Plots
- Real-Time Implementation
- Demo (or video, in case of technical difficulty)
- Conclusions
- References

Overview

- Build a guitar tuner
  - Demonstrate adaptive filtering to the family
  - You can never have enough guitar tuners
  - Evaluation boards collecting dust

The Approach

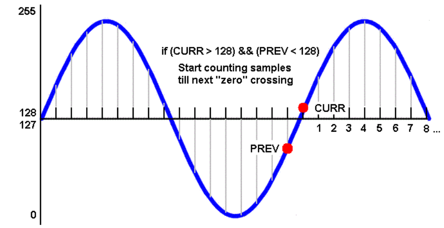
- Find a couple of pitch tracking algorithms
- Prototype in MATLAB
- Port to C on eval board
- Microphone input; sample real fast
- Test!

Risk Summary

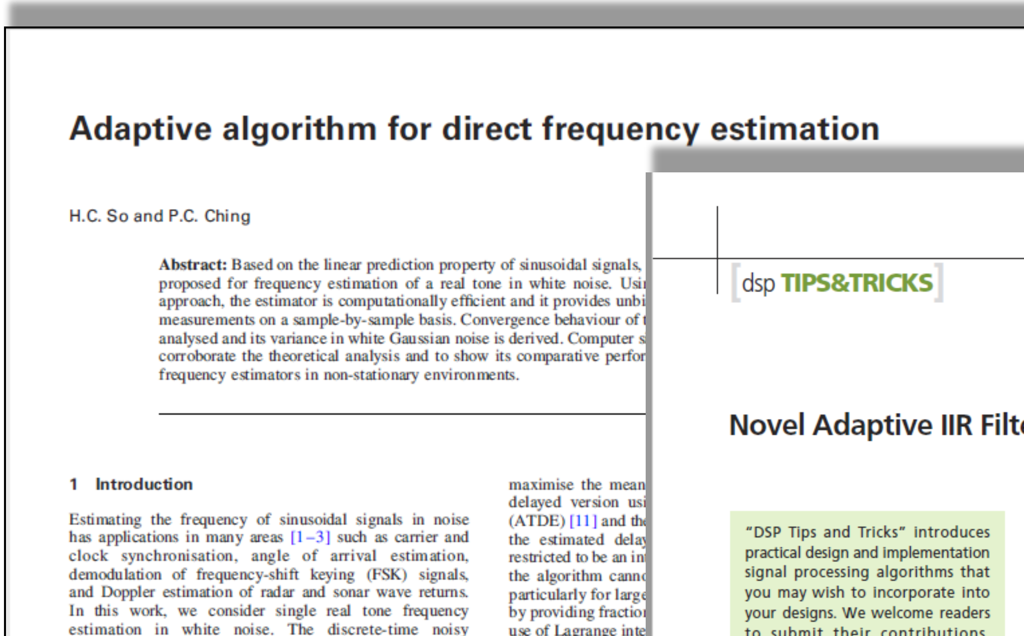
- ~~Algorithms; MATLAB~~
- Real-time demonstration
  - Does the eval board still work?
  - Are the toolchains compatible with Windows 10?
  - How do I connect the microphone to the ADC?
  - Is this CPU fast enough to sample @10kHz?
  - How do I illustrate results?
  - Watch out for ESD…ON THE WAY TO CLASS!!!!

Algorithms

- Zero-crossing
  - Count when sample goes from + to –



- Linear Prediction
  - $s=\cos(\omega)$ $\quad\quad s_n=2\cos(\omega)s_{n-1} - s_{n-2}$
- Adaptive Notch Filter
  - Adjust notch to minimize signal power



Adaptive algorithm for direct frequency estimation

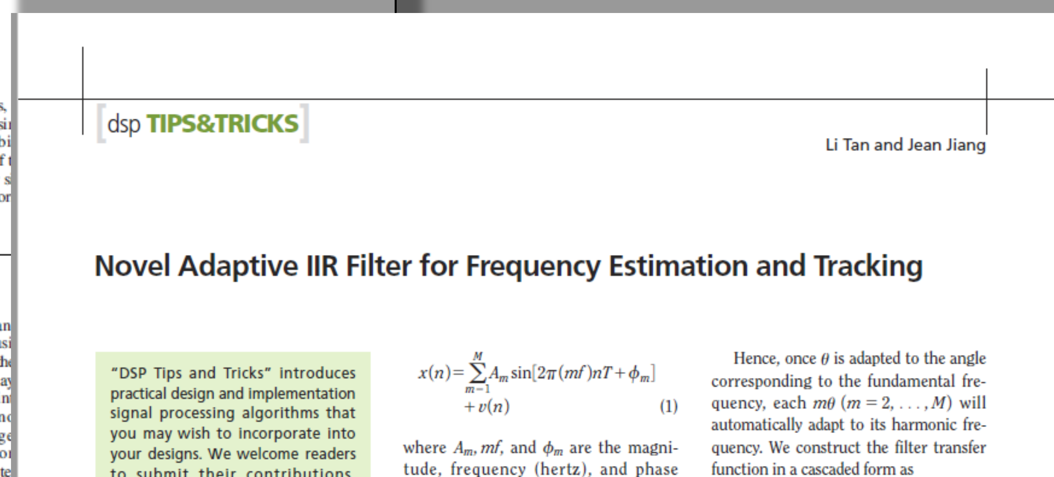
H.C. So and P.C. Ching

Abstract: Based on the linear prediction property of sinusoidal signals,
proposed for frequency estimation of a real tone in white noise. Usi
approach, the estimator is computationally efficient and it provides unbi
measurements on a sample-by-sample basis. Convergence behaviour of t
analysed and its variance in white Gaussian noise is derived. Computer s
corroborate the theoretical analysis and to show its comparative perfor
frequency estimators in non-stationary environments.

1 Introduction

Estimating the frequency of sinusoidal signals in noise has applications in many areas [1–3] such as carrier and clock synchronisation, angle of arrival estimation, demodulation of frequency-shift keying (FSK) signals, and Doppler estimation of radar and sonar wave returns. In this work, we consider single real tone frequency estimation in white noise. The discrete-time noisy

maximise the mean
delayed version usi
(ATDE) [11] and the
the estimated delay
restricted to be an in
the algorithm canno
particularly for large
by providing fractio
use of Lagrange inte



dsp TIPS&TRICKS

Li Tan and Jean Jiang

Novel Adaptive IIR Filter for Frequency Estimation and Tracking

"DSP Tips and Tricks" introduces practical design and implementation signal processing algorithms that you may wish to incorporate into your designs. We welcome readers to submit their contributions

$$x(n)=\sum_{m=1}^{M}A_m\sin[2\pi(mf)nT+\phi_m] + v(n) \quad (1)$$

where A_m, mf, and ϕ_m are the magnitude, frequency (hertz), and phase

Hence, once θ is adapted to the angle corresponding to the fundamental frequency, each $m\theta$ $(m = 2, \ldots, M)$ will automatically adapt to its harmonic frequency. We construct the filter transfer function in a cascaded form as

Algorithms

Linear Prediction aka Direct Frequency Estimation

$x_n = \alpha\cos(\omega n + \phi) + q_n \equiv s_n + q_n$	Input: sinusoid + noise
$s_n = 2\cos(\omega)s_{n-1} - s_{n-2}$	“It can be shown that…”
$\hat{s}_n = 2\cos(\hat{\omega})x_{n-1} - x_{n-2}$	Sinusoid approximated by two previous samples
$e_n \equiv x_n - \hat{s}_n$	Error signal
$E\left[e_n^2\right] \equiv 2\alpha^2\left(\cos(\hat{\omega}) - \cos(\omega)\right)^2 + 2\sigma_q^2\left(2 + \cos(2\hat{\omega})\right)$	Mean Squared Error
$E\left[\zeta_n^2\right] \equiv \dfrac{E\left[e_n^2\right]}{2(2+\cos(2\hat{\omega}))}$	Mean Squared Error, scaled
$\zeta_n^2 = \dfrac{e_n^2}{2(2+\cos(2\hat{\omega}_n))}$	Squared Error, scaled
$\dfrac{\partial\zeta_n^2}{\partial\hat{\omega}_n} = \dfrac{2\sin(\hat{\omega}_n)}{\left[2(2+\cos(2\hat{\omega}_n))\right]^2} e_n\left[(x_n + x_{n-2})\cos(\hat{\omega}_n) + x_{n-1}\right]$	Stochastic gradient estimate: differentiate wrt frequency
$\dfrac{\partial\zeta_n^2}{\partial\hat{\omega}_n} \approx e_n\left[(x_n + x_{n-2})\cos(\hat{\omega}_n) + x_{n-1}\right]$	Stochastic gradient estimate: simplified
$\hat{\omega}_{n+1} = \hat{\omega}_n - \mu e_n\left[(x_n + x_{n-2})\cos(\hat{\omega}_n) + x_{n-1}\right]$	LMS update equation

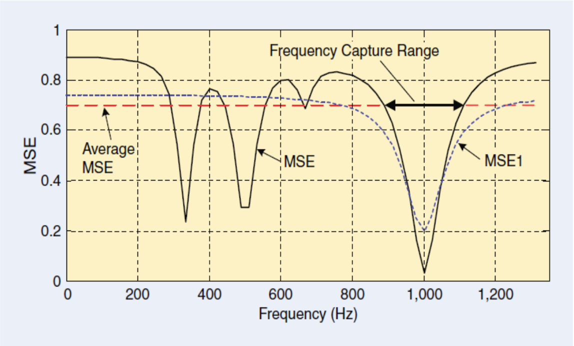
$$\hat{\omega}_{n+1} = \hat{\omega}_n - \mu e_n\left[(x_n + x_{n-2})\cos(\hat{\omega}_n) + x_{n-1}\right]$$

Algorithms

Adaptive Notch Filter

$$H(z)=\frac{Y(z)}{X(z)}=\frac{1-2\cos(\theta)z^{-1}+z^{-2}}{1-2r\cos(\theta)z^{-1}+r^2z^{-2}}$$ Transfer function

‘r’ controls notch width



Difference equation is

$$y(n)=x(n)-2\cos(\theta(n))x(n-1)+x(n-2)+2r\cos(\theta(n))y(n-1)-r^2y(n-2)$$

Since this is a notch filter, intended to minimize output y(n): the error e(n) = y(n)

Similar to previous: error squared; differentiate wrt frequency; obtain stochastic gradient expression

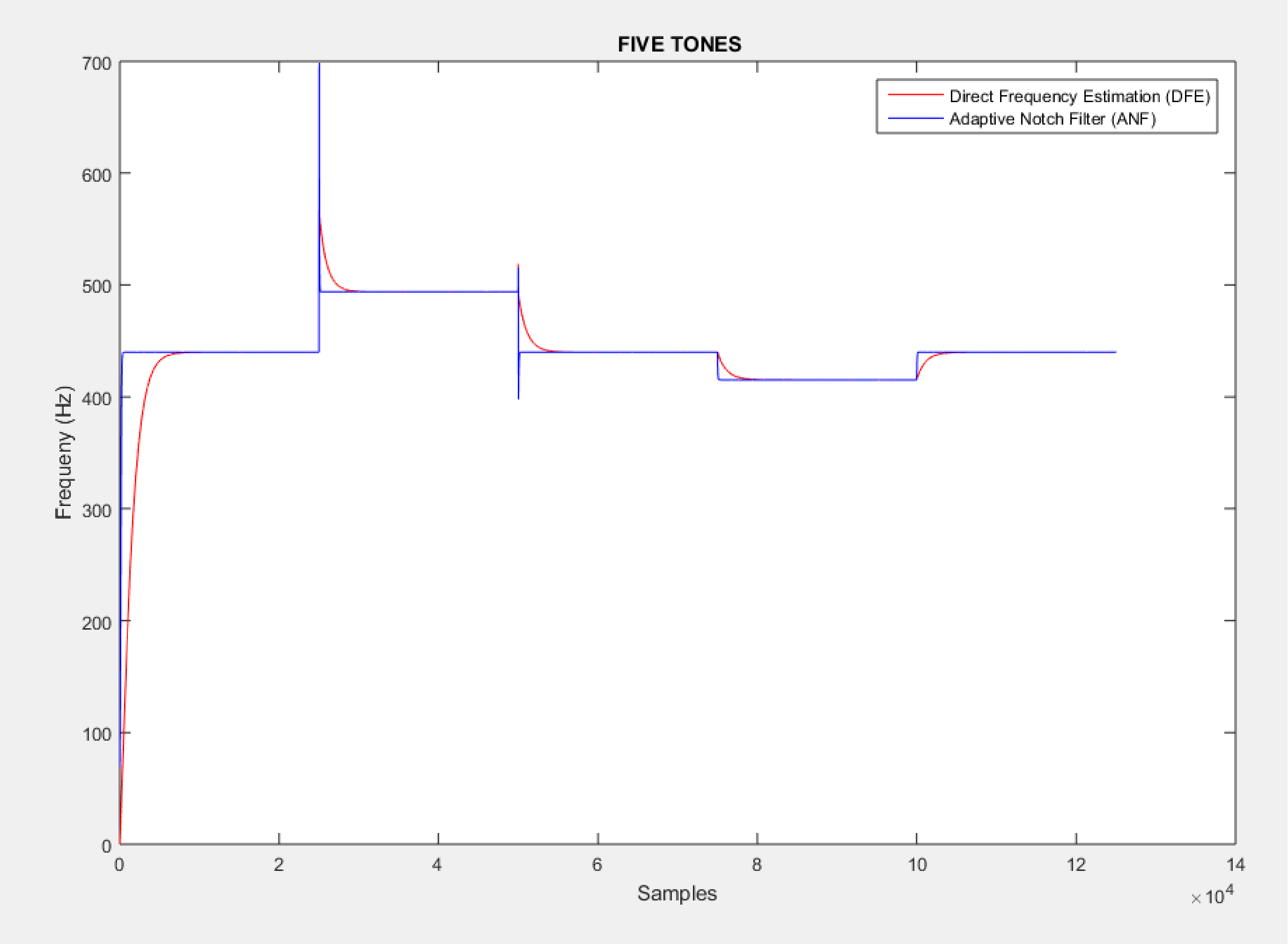
$$\frac{d(e_n^2)}{d\theta}=\beta(n)=2\sin(\theta(n))x(n-1)-2r\sin(\theta(n))y(n-1)+2r\cos(\theta(n))\beta(n-1)-r^2\beta(n-2)$$

$$\theta(n+1)=\theta(n)-2\mu y(n)\beta(n)$$ LMS update equation

$$\theta(n+1)=\theta(n)-2\mu y(n)\beta(n)$$

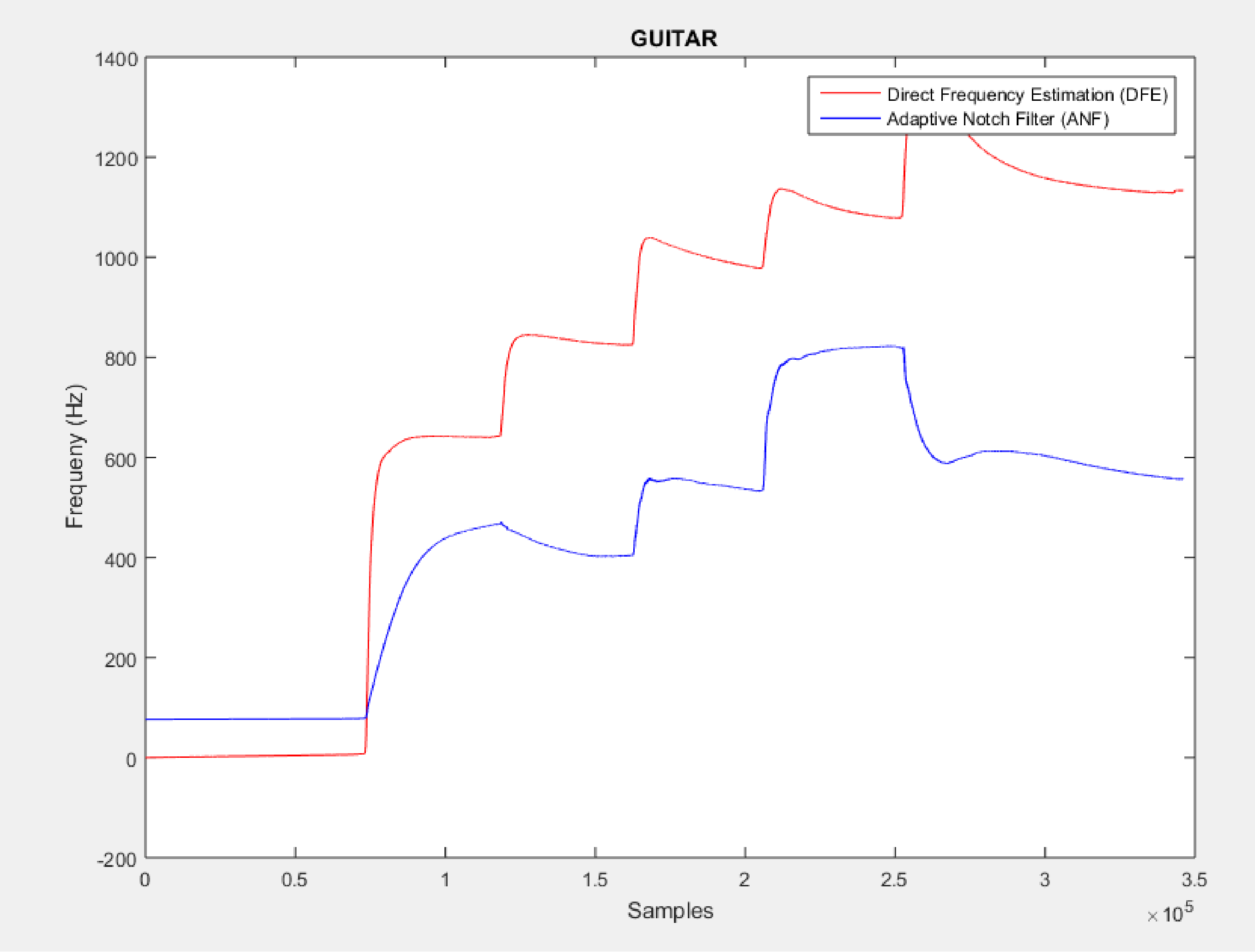
$$\beta(n)=2\sin(\theta(n))x(n-1)-2r\sin(\theta(n))y(n-1)+2r\cos(\theta(n))\beta(n-1)-r^2\beta(n-2)$$

MATLAB Plots



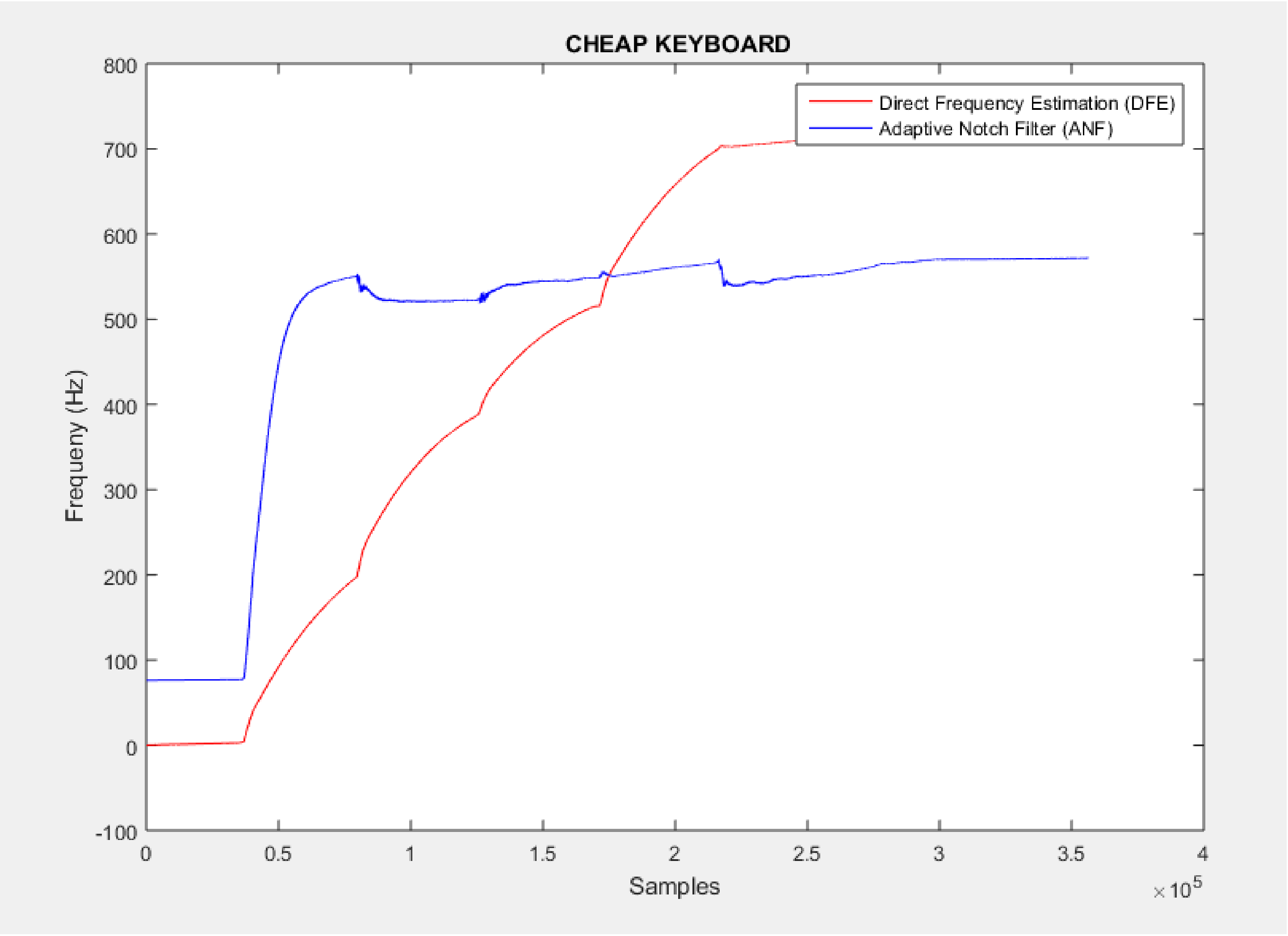
AUDIO

MATLAB Plots



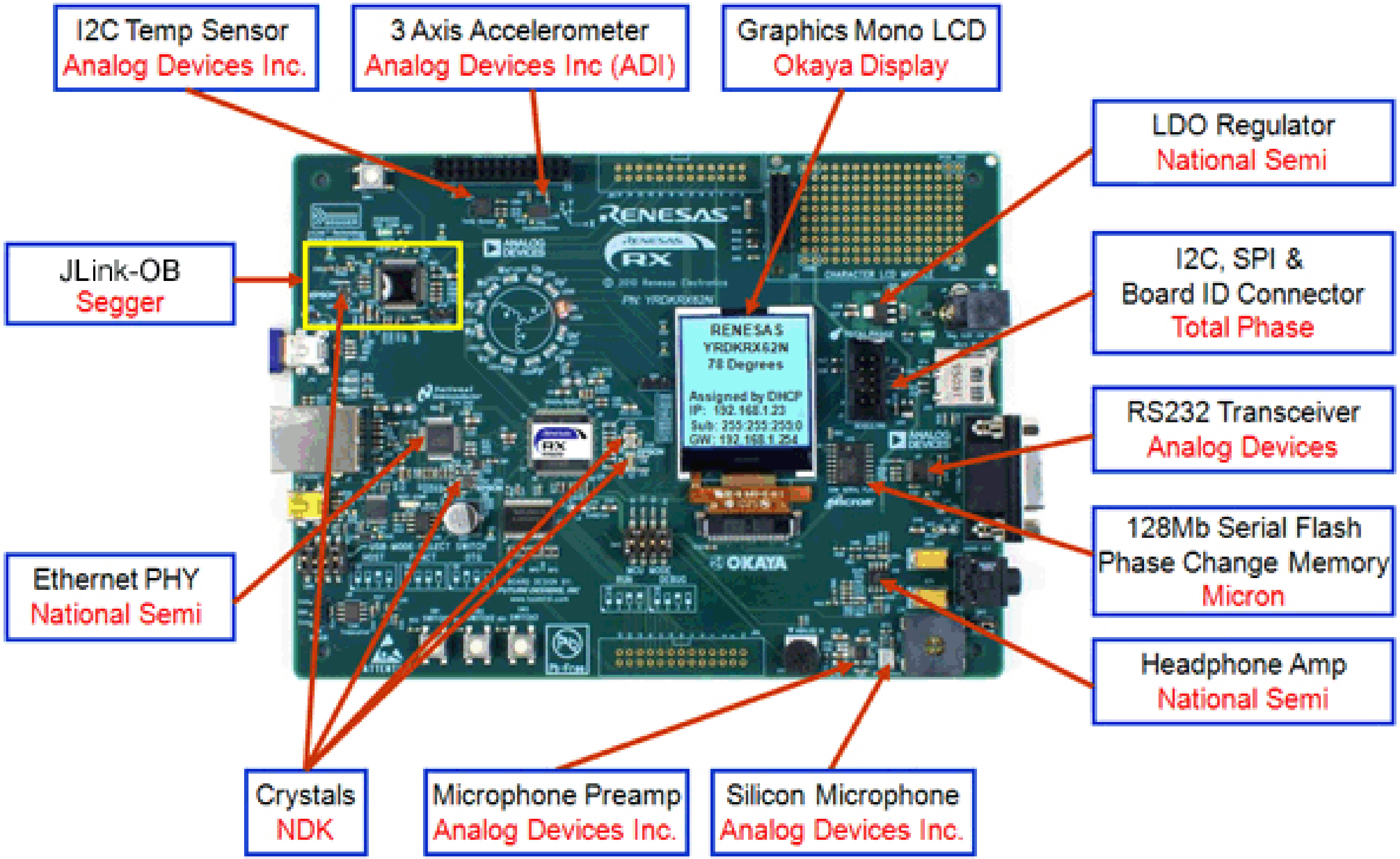
AUDIO

MATLAB Plots

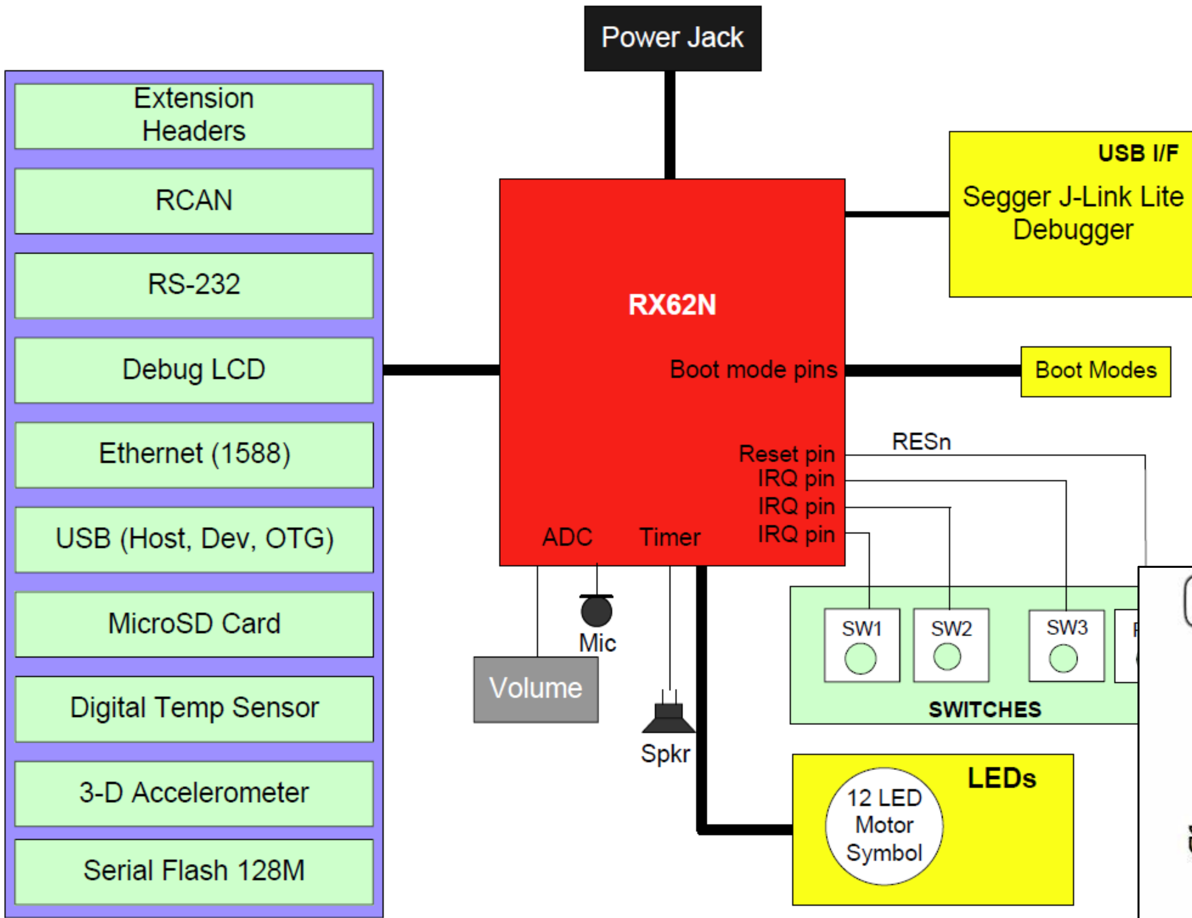


AUDIO

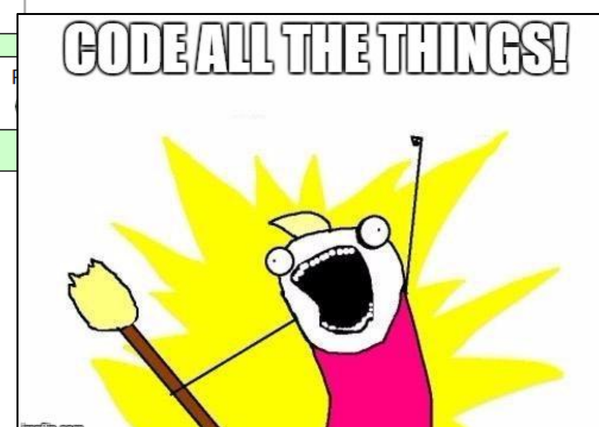
Real-Time Implementation



Real-Time Implementation



- ✓32-bit CPU
- ✓Floating point
- ✓ADC
- ✓Microphone
- ✓Works on Win10
- ✓Demo code works
- ✓Up to 20kHz



Demo

Or Video in case of technical difficulty

- Pure tones (Windows WAV)
- Piano: A4-440, C5-523.25
- Guitar: E4=329.63, A4=440, C5=523.25

Conclusions

- Embedded implementation appeared to work better than MATLAB & recordings
  - Recordings were of poor quality
  - Emphasized noise, overtones
- ANF superior to DFE
- Test hardware earlier…in case it doesn't work

References

1. So, H.C. and Ching, P.C.: “Adaptive algorithm for direct frequency estimation”
2. Tan, Li and Jiang, Jean: “Novel Adaptive IIR Filter for Frequency Estimation and Tracking”
3. renesas.com

Appendix: proof of cos()

$$s_n = \alpha\cos(\omega n + \phi)$$

$$= \alpha\cos(\omega n + \phi) + \alpha\cos(\omega n - 2\omega + \phi) - \alpha\cos(\omega n - 2\omega + \phi)$$

cosine is an even function: $\cos(a) = \cos(-a)$ so we can obtain

$$= \alpha\cos(\omega n + \phi) + \alpha\cos(-(\omega n - 2\omega + \phi)) - \alpha\cos(\omega n - 2\omega + \phi)$$

$$= \alpha\cos(\omega + \omega n - \omega + \phi) + \alpha\cos(\omega - \omega n + \omega - \phi) - \alpha\cos(\omega n - 2\omega + \phi)$$

$$= \alpha\cos(\omega + (\omega(n-1) + \phi)) + \alpha\cos(\omega - (\omega(n-1) + \phi)) - \alpha\cos(\omega n - 2\omega + \phi)$$

$$= 2\alpha\frac{1}{2}\left[\cos(\omega + (\omega(n-1) + \phi)) + \cos(\omega - (\omega(n-1) + \phi))\right] - \alpha\cos(\omega n - 2\omega + \phi)$$

recall the trigonometric identity:

$$\cos(a)\cos(b) = \frac{1}{2}(\cos(a-b) + \cos(a+b))$$

so we can write

$$= 2\alpha\left[\cos(\omega)\cos(\omega(n-1) + \phi))\right] - \alpha\cos(\omega n - 2\omega + \phi)$$

$$= 2\cos(\omega)\alpha\cos(\omega(n-1) + \phi)) - \alpha\cos(\omega(n-2) + \phi)$$

$$= 2\cos(\omega)s_{n-1} - s_{n-2}$$