Processamento de áudio em tempo real em dispositivos computacionais de alta disponibilidade e baixo custo

André J. Bianchi

15/10/2013

Introdução

Contexto e motivação

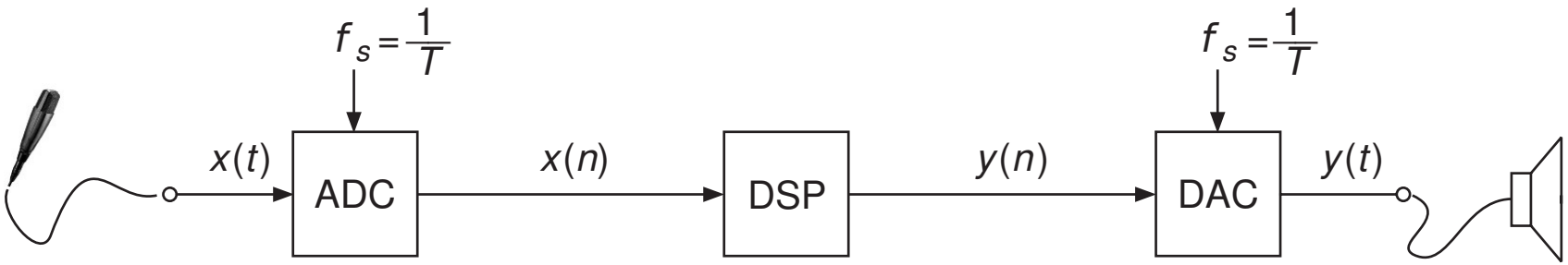
Pesquisa em computação musical:

- Recuperação de informações.
- Composição / improvisação / acompanhamento.
- Síntese sonora.
- Auralização / espacialização.
- Processamento de áudio.

Contexto tecnológico:

- Lei de Moore.
- Obsolescência programada.
- Propaganda / Fetichização / consumismo.
- Lixo tecnológico.

Processamento de áudio digital



Ciclo de processamento

- Entrada:
  - Conversão Analógico-Digital.
  - Frequência de amostragem e amplitude de espectro.
  - Quantização e profundidade de bits.
  - Sinal digital: $x : \mathbb{N} \to \{0,1\}^b$.
- Processamento.
  - Manipulação do sinal: análise, síntese, recuperação de informação, efeitos.
  - Processamento em blocos, sobreposição.
  - Representações: tempo e frequência.
- Saída.
  - Sinal digital: $y : \mathbb{N} \to \{0,1\}^{b'}$.
  - Taxa de geração de amostras e profundidade de bits.
  - Conversão Digital-Analógico.

Processamento em tempo real

Processamento em blocos:

- Período do bloco: N amostras.
- Frequência de amostragem: R Hz.
- Atraso mínimo: $\frac{N}{R}$ s.
- Fator de sobreposição: M.
- Cada ciclo avança $\frac{N}{M}$ amostras ou $\frac{N}{MR}$ s.

Restrições de tempo real:

- O ciclo tem que ser executado periodicamente.
- O período do ciclo tem que ser menor que tempo de computação tem que ser menor que o período do ciclo de processamento.

Proposta de pesquisa

Perguntas sobre processamento em tempo real:

- Quais modelos computacionais e dispositivos podem ser utilizadas? Quais as restrições e possibilidades que cada um possui?
- Como medir o desempenho de uma modelo ou dispositivo para o processamento de áudio em tempo real?

Proposta:

- Escolher dispositivos de baixo custo e alta disponibilidade.
- Escolher uma série de algoritmos de processamento de áudio frequentemente utilizados.
- Investigar a implementação e o desempenho dos algoritmos nos dispositivos.

Objetivo da pesquisa

Desenvolvimento de uma metodologia . . .

. . . para análise de desempenho. . .

. . . do processamento de áudio . . .

. . . em tempo real. . .

Objetivo da pesquisa

Desenvolvimento de uma metodologia . . .

. . . para análise de desempenho. . .

. . . do processamento de áudio . . .

. . . em tempo real. . .

. . . em dispositivos computacionais. . .

. . . de baixo custo. . .

. . . e alta disponibilidade.

Metodologia

Metodologia geral

Formatação do problema:

- Identificação de modelos computacionais de interesse.
- Escolha de dispositivos.
- Escolha de algoritmos de processamento em tempo real.
- Estabelecimento de métricas e métodos para avaliação de desempenho.

Experimentação:

- Desenvolvimento de testes.
- Implementação.
- Execução dos testes e obtenção dos resultados.
- Análise dos resultados.

Identificação de modelos computacionais de interesse

Critérios de interesse geral:

- Baixo custo.
- Alta disponibilidade.

Critérios de interesse para a área de computação musical:

- Sensores.
- Mobilidade.
- Poder de processamento.
- Licenças de uso.
- Expressão computacional.

Modelos computacionais escolhidos

Microcontroladores:

- Baixo custo.
- Baixo consumo de energia.

Processadores paralelos:

- Alta capacidade computacional.
- Possibilidade de exploração do paralelismo.

Dispositivos móveis:

- Ubiquidade.
- Versatilidade.

Dispositivos escolhidos

- Microcontrolador: Arduino com ATmega328P.
- Processador paralelo: placas GPU.
- Dispositivos móveis: Sistema operacional Android.

Algoritmos de processamento em tempo real

Alguns algoritmos frequentemente utilizados em rotinas de manipulação de áudio:

- FFT.
- Convolução no domínio do tempo.
- Síntese aditiva.

Transformada Discreta de Fourier (DFT)

Seja $\mathbf{x} \in \mathbb{C}^N$ um vetor $(x_0, x_1, \ldots, x_{N-1})$. A *Transformada Discreta de Fourier* de $\mathbf{x}$ é o vetor $\mathbf{X} \in \mathbb{C}^N$ com componentes

$$X_k = \sum_{m=0}^{N-1} x_m e^{-2\pi i k m/N},$$

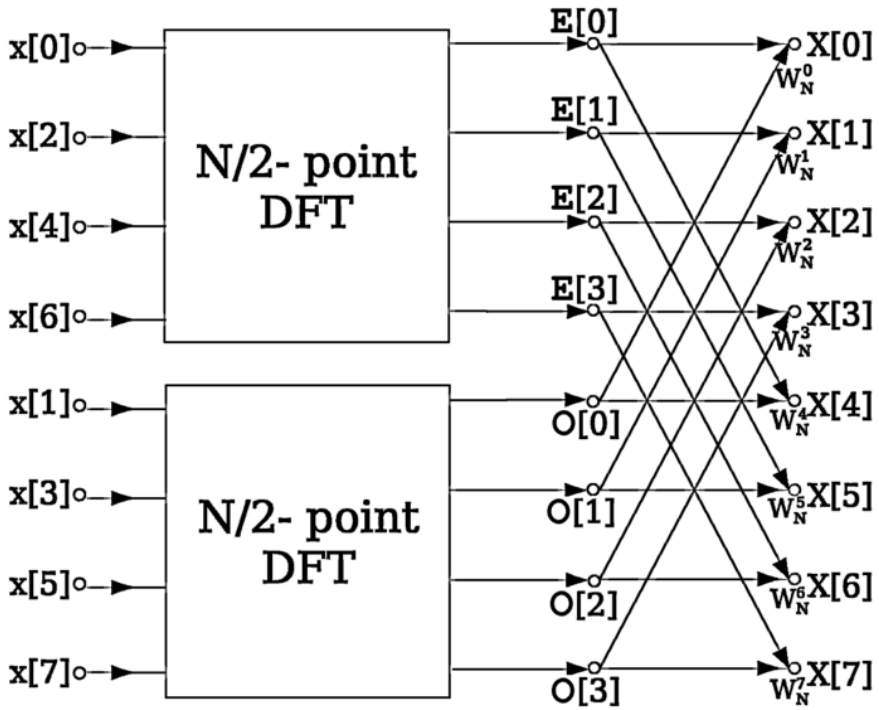
para $0 \leq k \leq N-1$.

Propriedades:

- Linearidade.
- Inversível (IDFT).
- Simetria para sinais reais.

Transformada Rápida de Fourier (FFT)

- A implementação ingênua da DFT é $O(N \log(N))$.
- É possível explorar a estrutura da DFT para acelerar a computação.



A FFT é $O(N \log(N))$

Transformada Discreta de Fourier:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) e^{\frac{-2\pi i}{N} nk} \\ &= \underbrace{\sum_{m=0}^{\frac{N}{2}-1} x(2m) e^{\frac{-2\pi i}{N}(2m)k}}_{E_k} + e^{\frac{-2\pi i}{N}k} \underbrace{\sum_{m=0}^{\frac{N}{2}-1} x(2m+1) e^{\frac{-2\pi i}{N}(2m)k}}_{O_k}. \end{aligned}$$

Transformada Rápida de Fourier:

$$X(k) = \begin{cases} E_k + e^{\frac{-2\pi i}{N}k} O_k & \text{se } k < \frac{N}{2} \\ E_{k-\frac{N}{2}} + e^{\frac{-2\pi i}{N}(k-\frac{N}{2})} O_{k-\frac{N}{2}} & \text{se } k \geq \frac{N}{2} \end{cases}$$

Convolução circular

Sejam $\mathbf{x}$, $\mathbf{y} \in \mathbb{C}^N$. A *convolução circular* de $\mathbf{x}$ e $\mathbf{y}$ é o vetor $\mathbf{w} \in \mathbb{C}^N$ com componentes

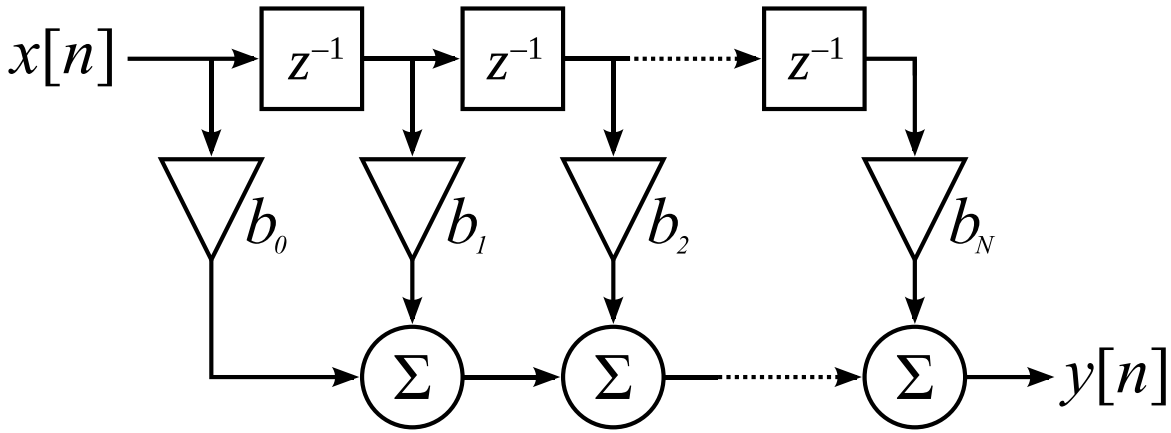
$$w_r = \sum_{n=0}^{N-1} x_k y_{(r-k \bmod N)},$$

para $0 \leq r \leq N-1$. A convolução circular é denotada por $\mathbf{w} = \mathbf{x} * \mathbf{y}$.

Teorema da convolução: Se $\mathbf{X}$, $\mathbf{Y}$ e $\mathbf{W}$ são as DFTs de $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{w}$, então

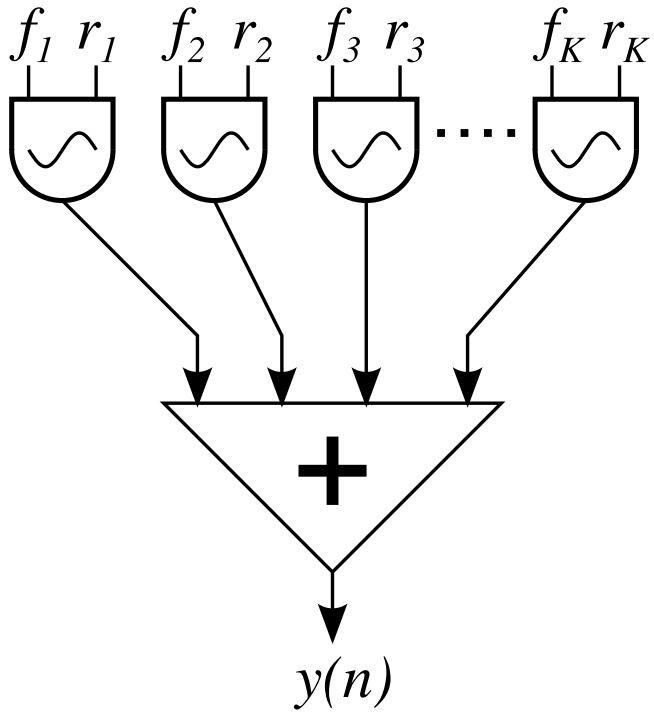
$$W_k = X_k Y_k.$$

Convolução no domínio do tempo




```
y[n] = 0
para k=0 até S, faça:
    y[n] += b[k] * x[n-k]
```

Síntese aditiva



$$y(n) = \sum_{k=1}^{K} r_k(n) \sin\left(\frac{2\pi f_k n}{R}\right), \quad n \geq 0.$$

Síntese aditiva: implementação com consulta a tabela

Para frequências fixas:

$$y(n) = \sum_{k=1}^{K} r_k(n)\mathbf{s}[f_k nS/R].$$

Para frequências que variam com o tempo:

$$y(n) = \sum_{k=1}^{K} r_k(n)q(\mathbf{s}, l_k(n)),$$

onde:

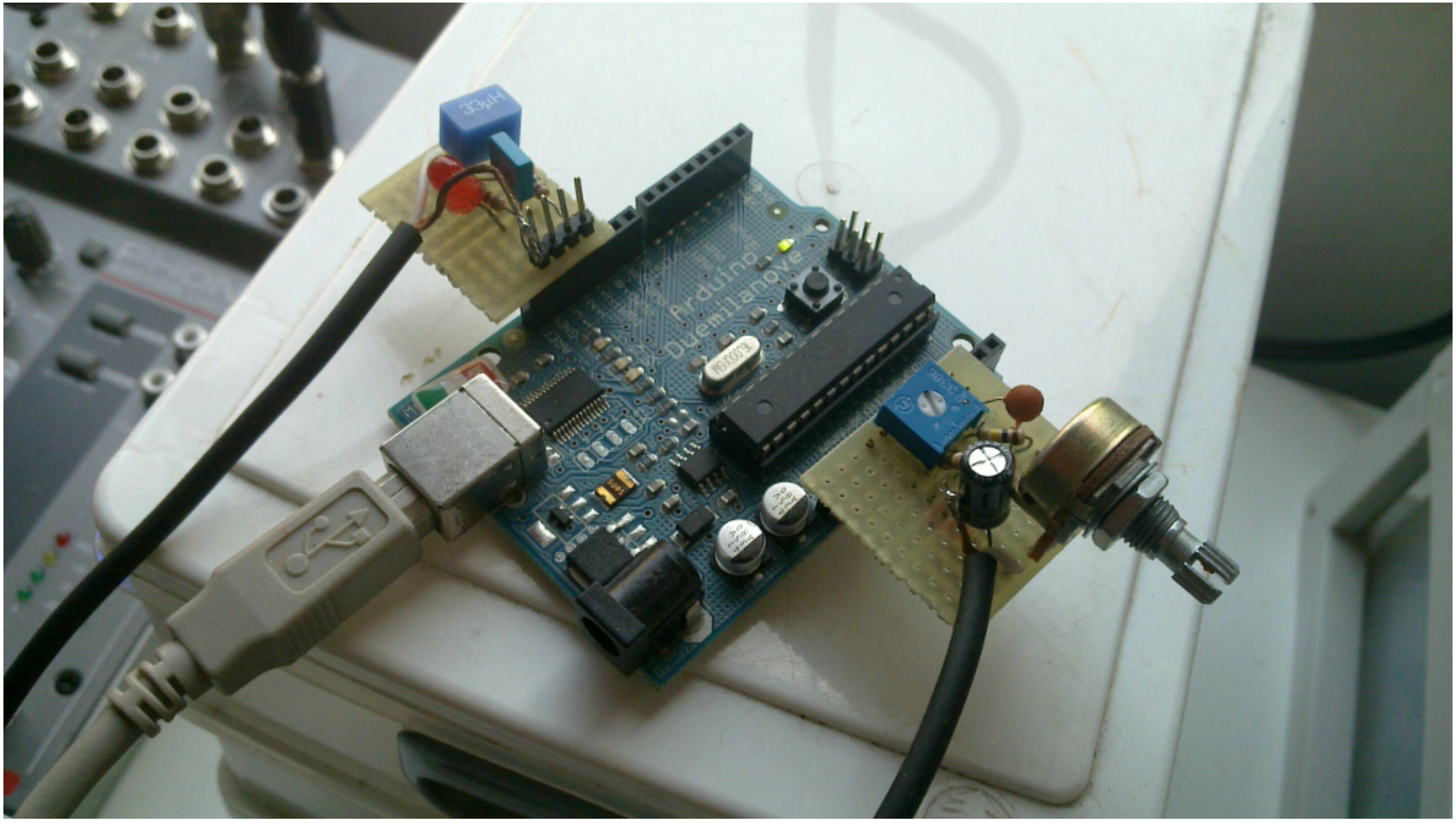
- $l_k(n)$ é o índice de consulta.
- q é uma função de consulta que incrementa $l_k(n)$ de acordo com o valor atual de f_k.

Perguntas sobre os algoritmos

- Quais parâmetros de cada algoritmo influenciam na complexidade?
- Qual é o tempo gasto para computar uma certa instância de um algoritmo?
- Qual é a maior instância computável em tempo real?
- Quais as possibilidades e restrições de implementação em cada dispositivo?

Arduino

Arduino



O projeto Arduino

Características principais:

- Estrutura minimal para interface com um microcontrolador.
- Baixo custo: 20-50 USD.
- Baixo consumo de energia.
- Licenciamento livre.

Microcontrolador Atmel AVR ATmega328P:

- CPU: unidade aritmética e registradores (16 MHz - 8 bits).
- Memórias: Flash (32 KB), SRAM (2 KB) e EEPROM (1 KB).
- Portas digitais de entrada (com ADC) e saída (com PWM).

Idiossincrasias do Arduino

- Entrada/saída de dados depende de ADC e DCA.
- Limitação de memória.
- Arquitetura RISC de 8 bits.

Entrada de áudio: período de amostragem

pré-escalonador	T_{conv} (μs)	$\tilde{T}_{\text{conv}}$ (μs)	$\tilde{f}_{\text{conv}}$ ($\approx$KHz)
2	1,81	12,61	79,30
4	3,62	16,06	62,26
8	7,25	19,76	50,60
16	14,50	20,52	48,73
32	29,00	34,80	28,73
64	58,00	67,89	14,72
128	116,00	114,85	8,70

- Resolução da função `micros()`: 4˜μs.
- $R = 44.100$˜Hz $\Rightarrow 1/R \approx 22,67$ μs.
- $R = 31.250$˜Hz $\Rightarrow 1/R = 32,00$ μs.

Entrada de áudio: ADC

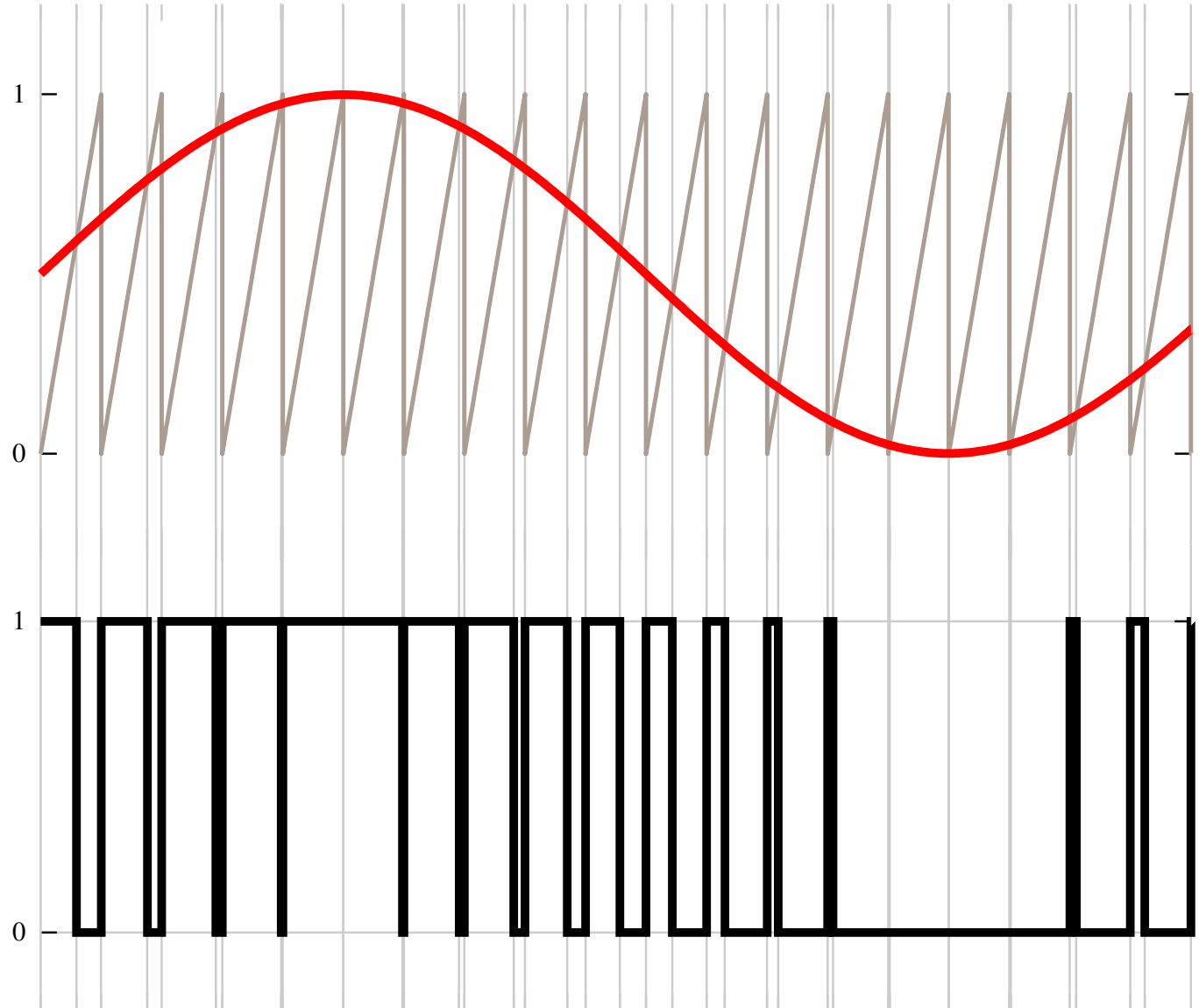
Conversor analógico-digital:

- Resolução: 8 ou 10 bits.
- Tempo de conversão: 13 a 260 μs.
- Conversão manual ou automática.

Parâmetros escolhidos:

- Resolução de 8 bits.
- Pré-escalonador igual a 8 (até 50 KHz).

Saída de áudio: Pulse Width Modulation (PWM)



Frequências de operação de um contador de 8 bits

pré-escalonador	f_{incr} (KHz)	f_{overflow} (Hz)
1	16.000	62.500
8	2.000	7.812
32	500	1.953
64	250	976
128	125	488
256	62,5	244
1024	15,625	61

Saída de áudio

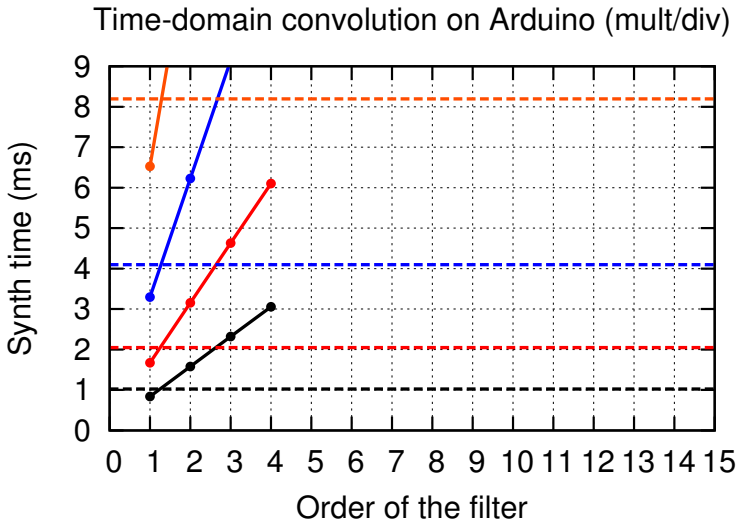
PWM no ATmega328P:

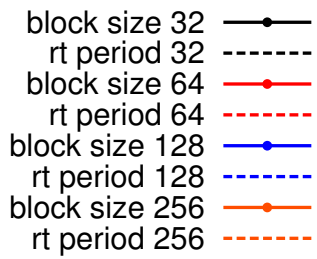
- Modos de operação: *Fast* e *Phase Correct*.
- Pré-escalonador.
- 2 contadores de 8 bits e 1 de 16 bits.
- Interrupção por *overflow*.

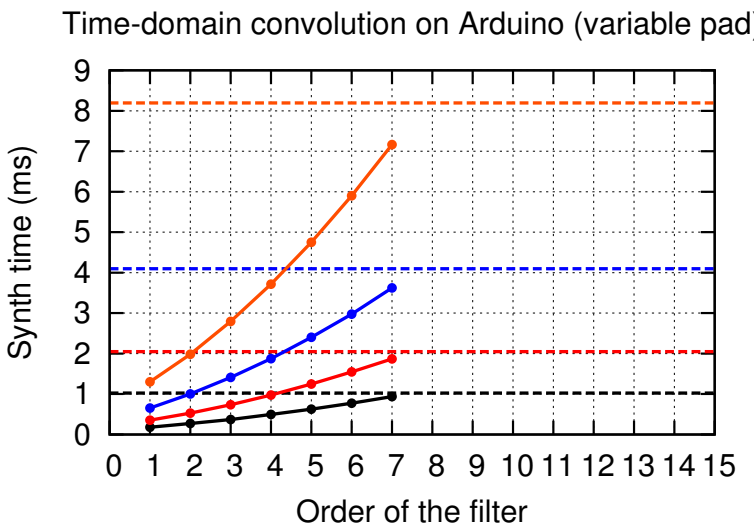
Parâmetros escolhidos:

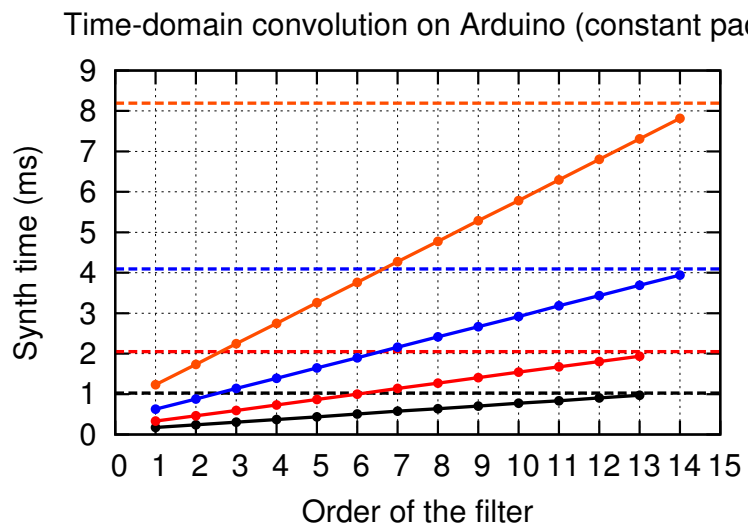
- *Fast PWM*.
- Contador de 8 bits.
- Pré-escalonador igual a 1.
- Frequência de *overflow*: 16˜MHz / 1 / 2^8 = 62.500˜Hz.
- Taxa de geração de amostras: 31.250˜Hz.

Convolução no Arduino (usando operações sobre bits)

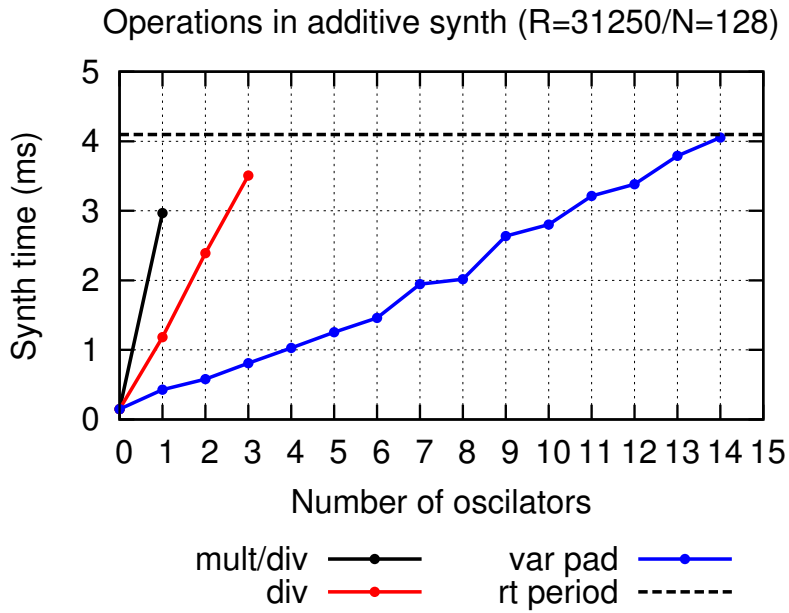




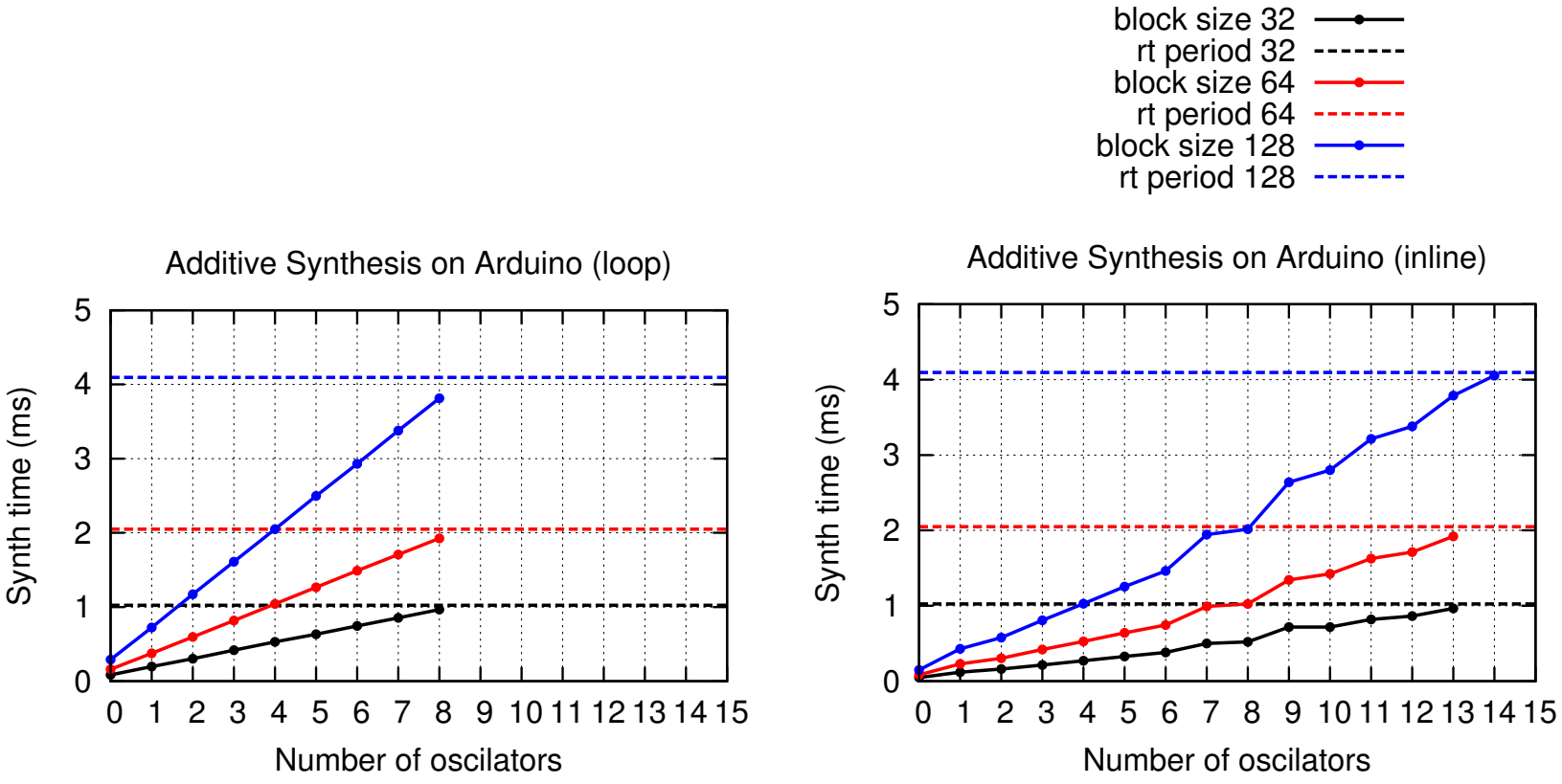




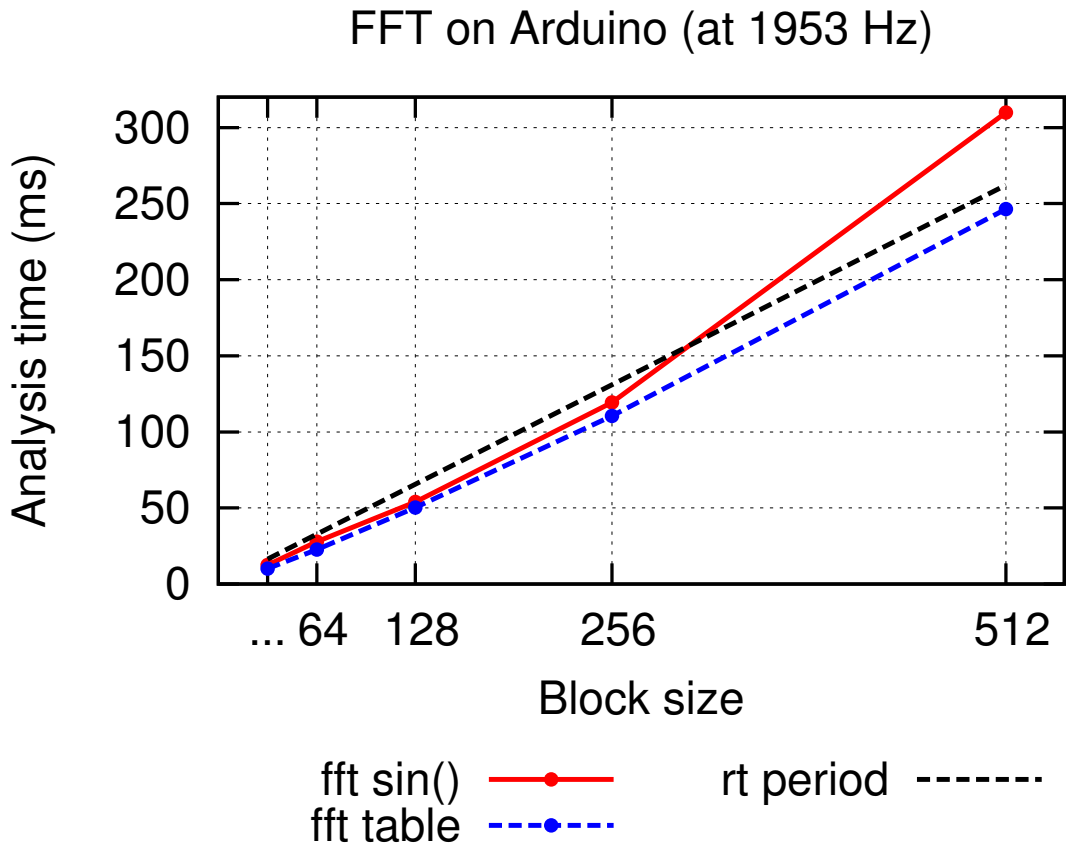
Síntese aditiva no Arduino: quantidade e tipos de operação utilizadas



Síntese aditiva no Arduino: uso de laços



FFT no Arduino (a 1953 Hz)



Algumas conclusões...

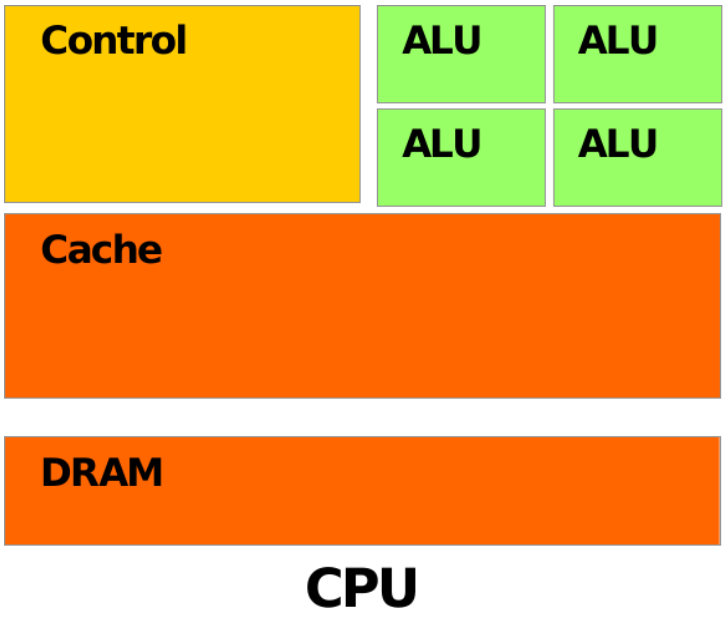
... sobre processamento de áudio em tempo real em Arduino:

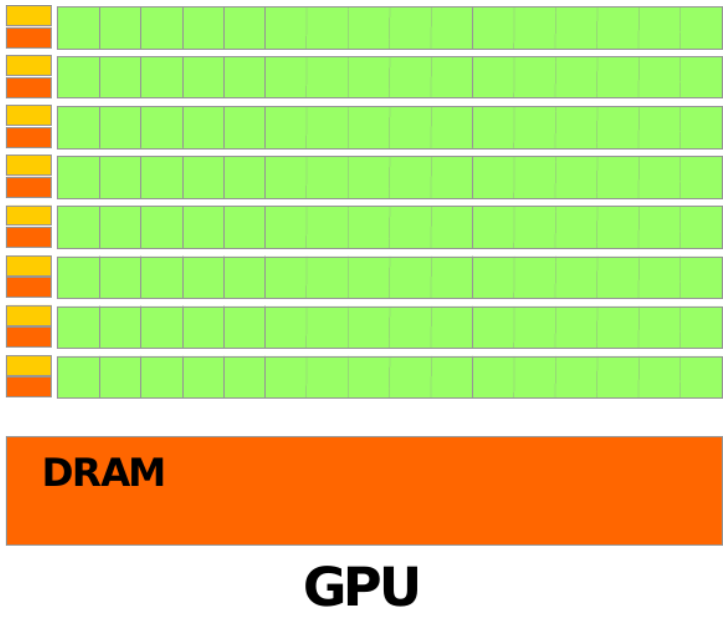
- Tipos utilizados (byte, unsigned long, int, float, etc) são fundamentais.
- Multiplicação/divisão (de inteiros) demoram pelo menos o dobro que operações sobre inteiros.
- A quantidade de laços e condicionais faz diferença.
- Consulta a variáveis e vetores também faz diferença.

GPU

GPU: Graphics Processing Unit

Diferenças gerais em relação à CPU





Idiossincrasias da GPU

- Paralelismo.
- A memória da GPU não é compartilhada com a da CPU.

Análise de desempenho na GPU

Valores importantes:

- Tempo de transferência de memória (ida e volta).
- Tempo de execução do programa.
- Tempo total (soma dos tempos acima).

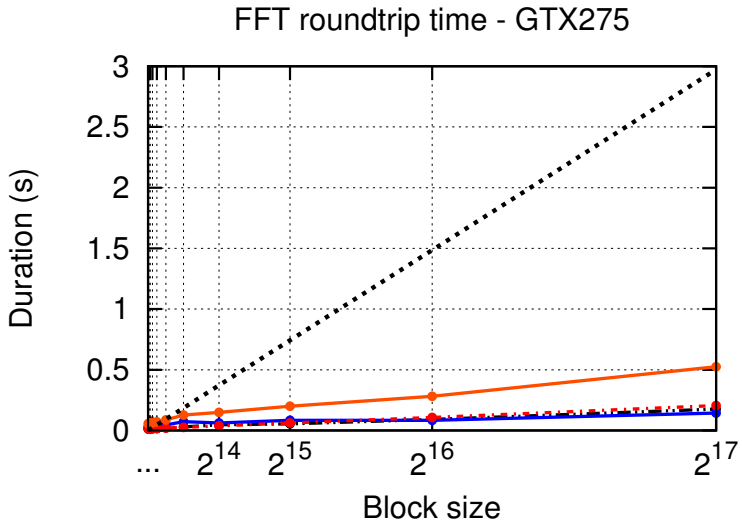
Implementações paralelas:

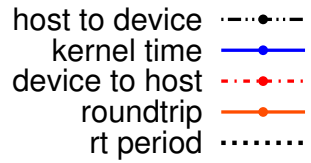
- FFT (da API).
- Phase Vocoder (FFT + Síntese Aditiva).

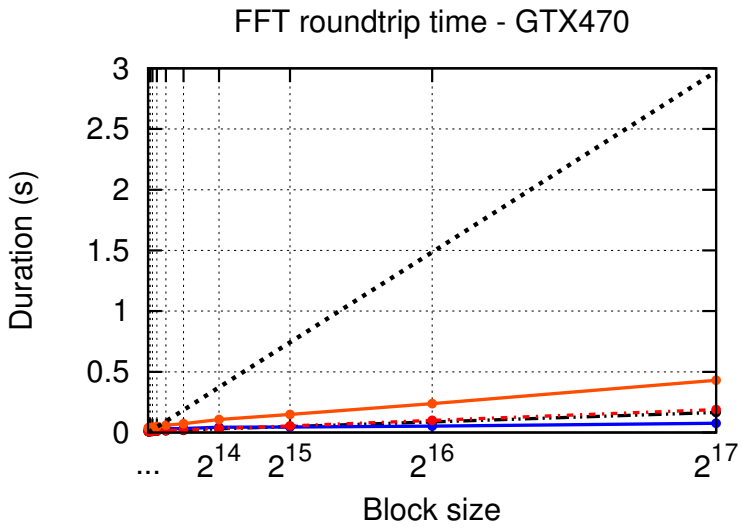
Placas utilizadas para testes

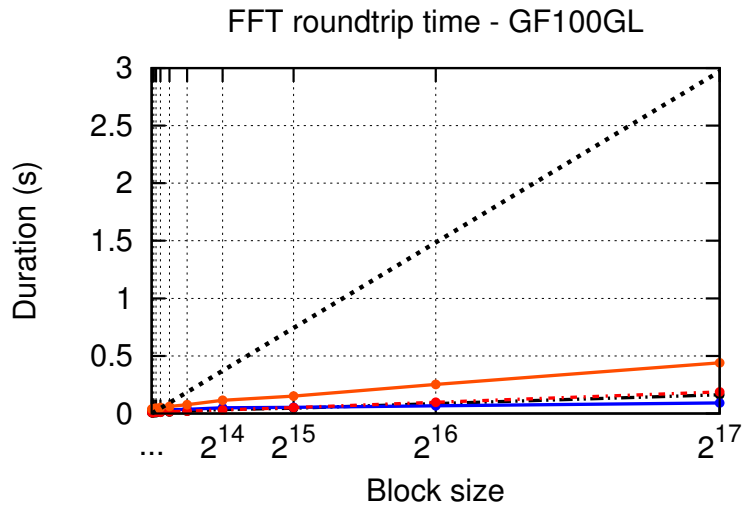
	GF100GL	GTX275	GTX400
CUDA cores	256	240	448
Memória RAM (MB)	2000	896	1280
Banda de memória (GB/s)	89.6	127.0	133.9

FFT na GPU

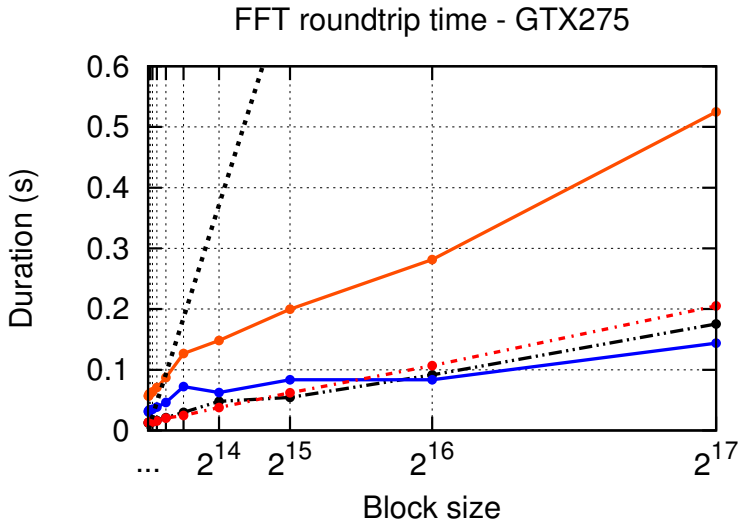

FFT roundtrip time - GTX275
Duration (s)
3
2.5
2
1.5
1
0.5
0
...
2^{14}
2^{15}
2^{16}
2^{17}
Block size

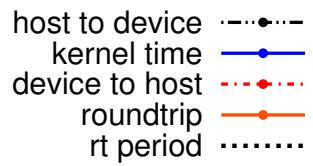

host to device
kernel time
device to host
roundtrip
rt period

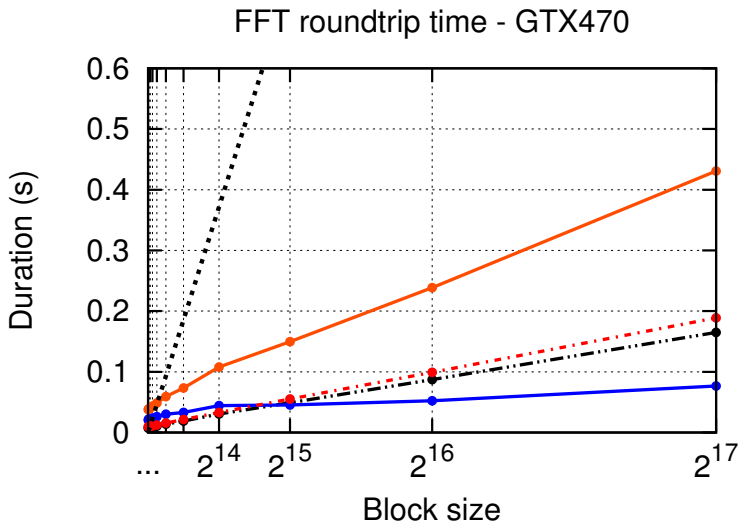

FFT roundtrip time - GTX470
Duration (s)
3
2.5
2
1.5
1
0.5
0
...
2^{14}
2^{15}
2^{16}
2^{17}
Block size

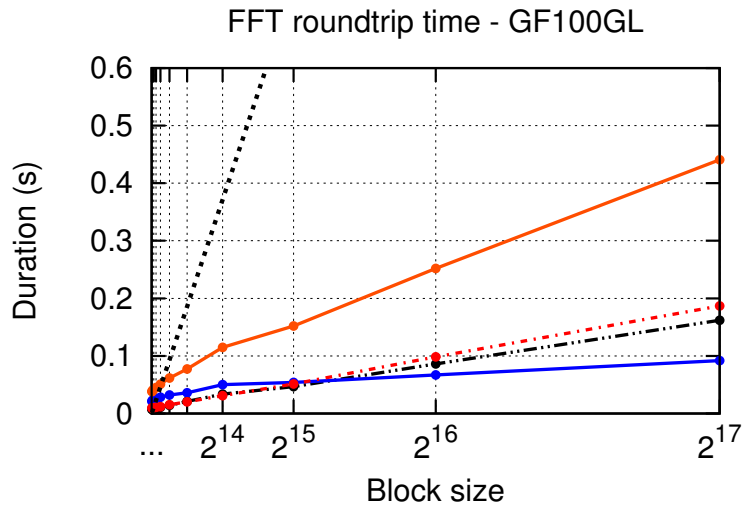

FFT roundtrip time - GF100GL
Duration (s)
3
2.5
2
1.5
1
0.5
0
...
2^{14}
2^{15}
2^{16}
2^{17}
Block size

FFT na GPU (zoom)

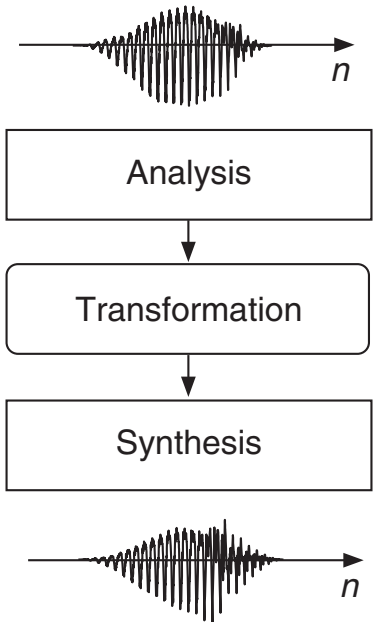


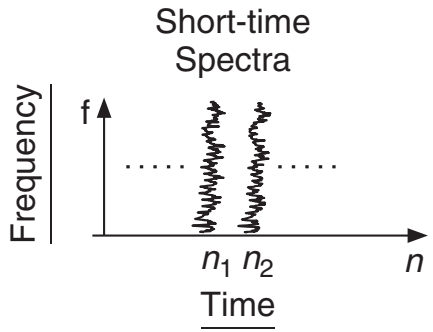






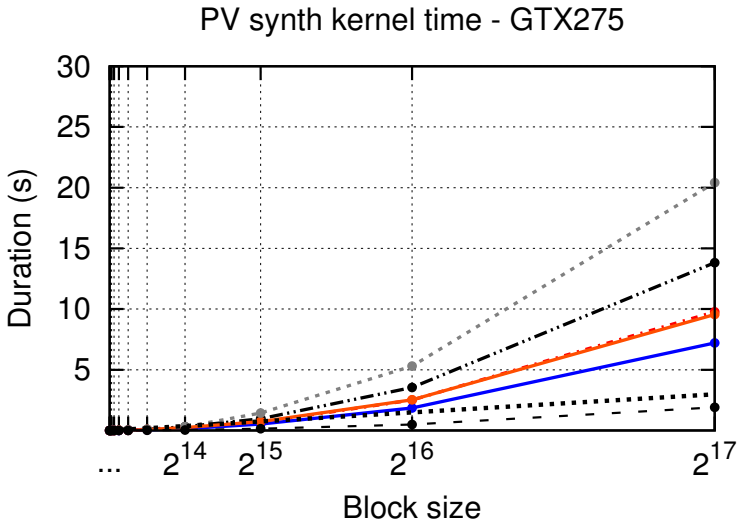
Mais um algoritmo: Phase Vocoder

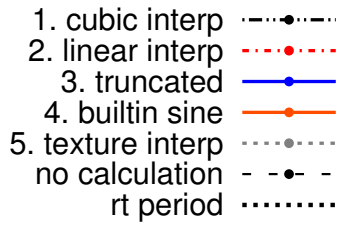


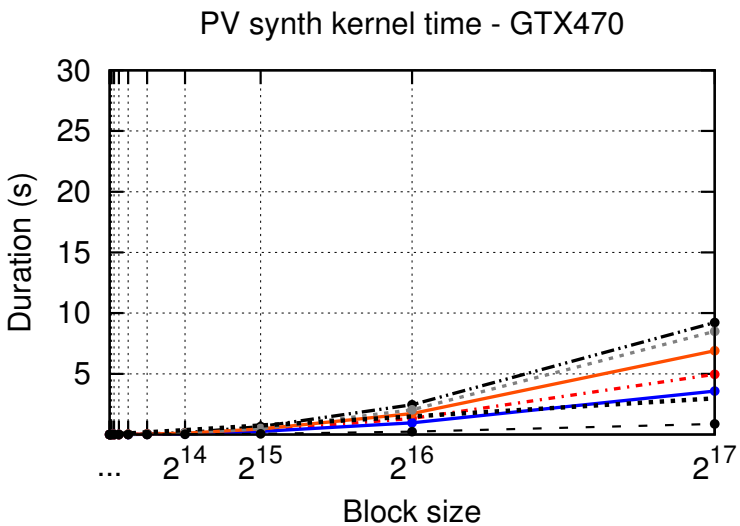


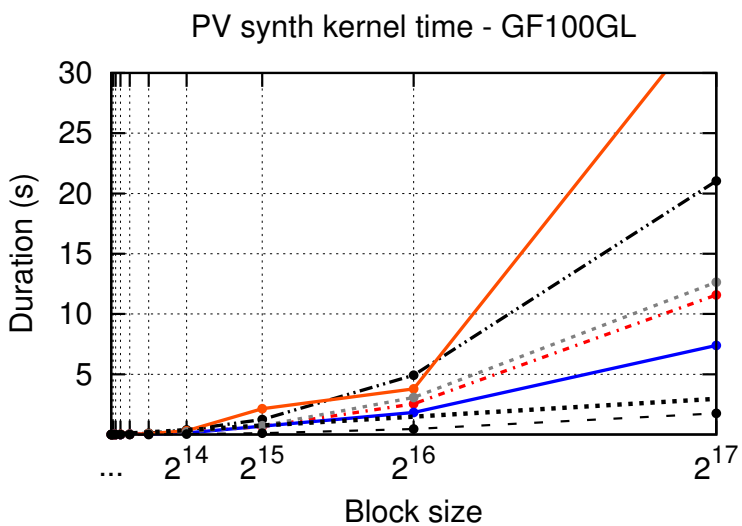
1. **Análise:**
   - Janela deslizante de tamanho finito.
   - FFTs consecutivas do sinal $x(n)$.
   - Espectro variante no tempo: $X(n, k)$.
2. **Transformação:** modificações em $|X(n, k)|$ e $\varphi(X(n, k))$.
3. **Síntese:** Síntese aditiva.

Síntese aditiva do Phase Vocoder na GPU

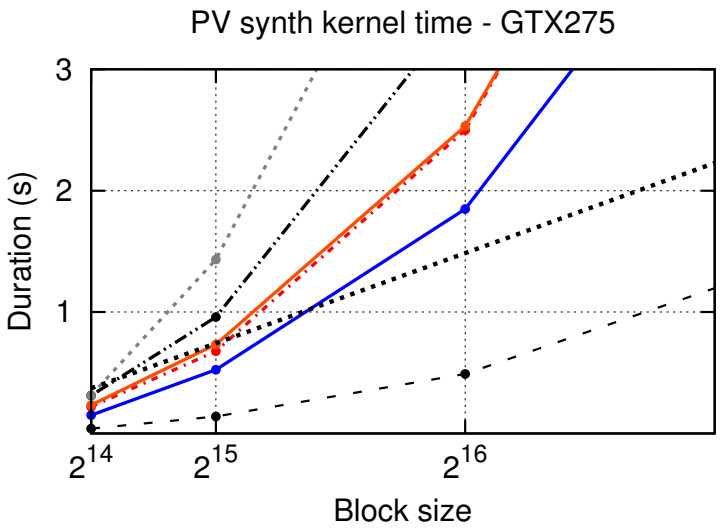


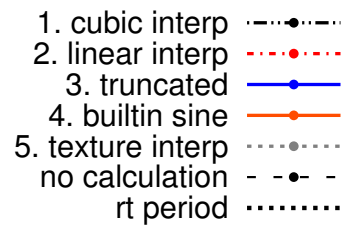


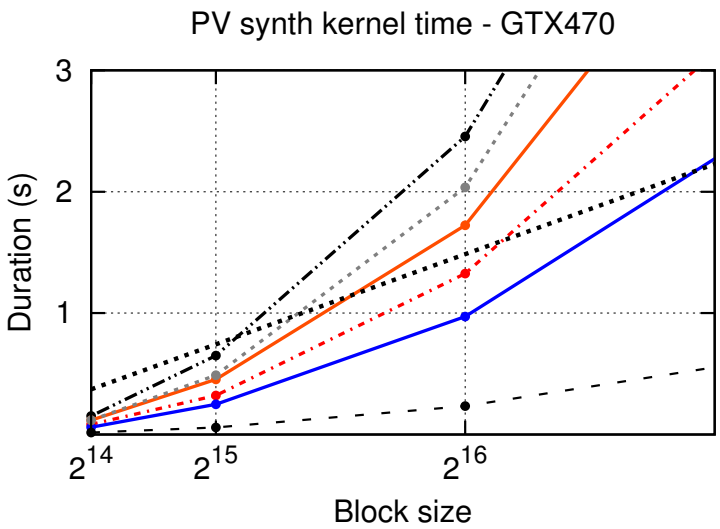


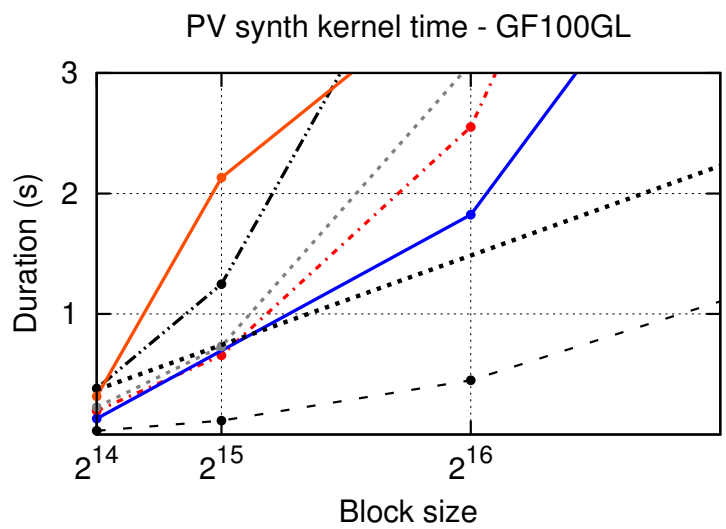


Síntese aditiva do Phase Vocoder na GPU (zoom)









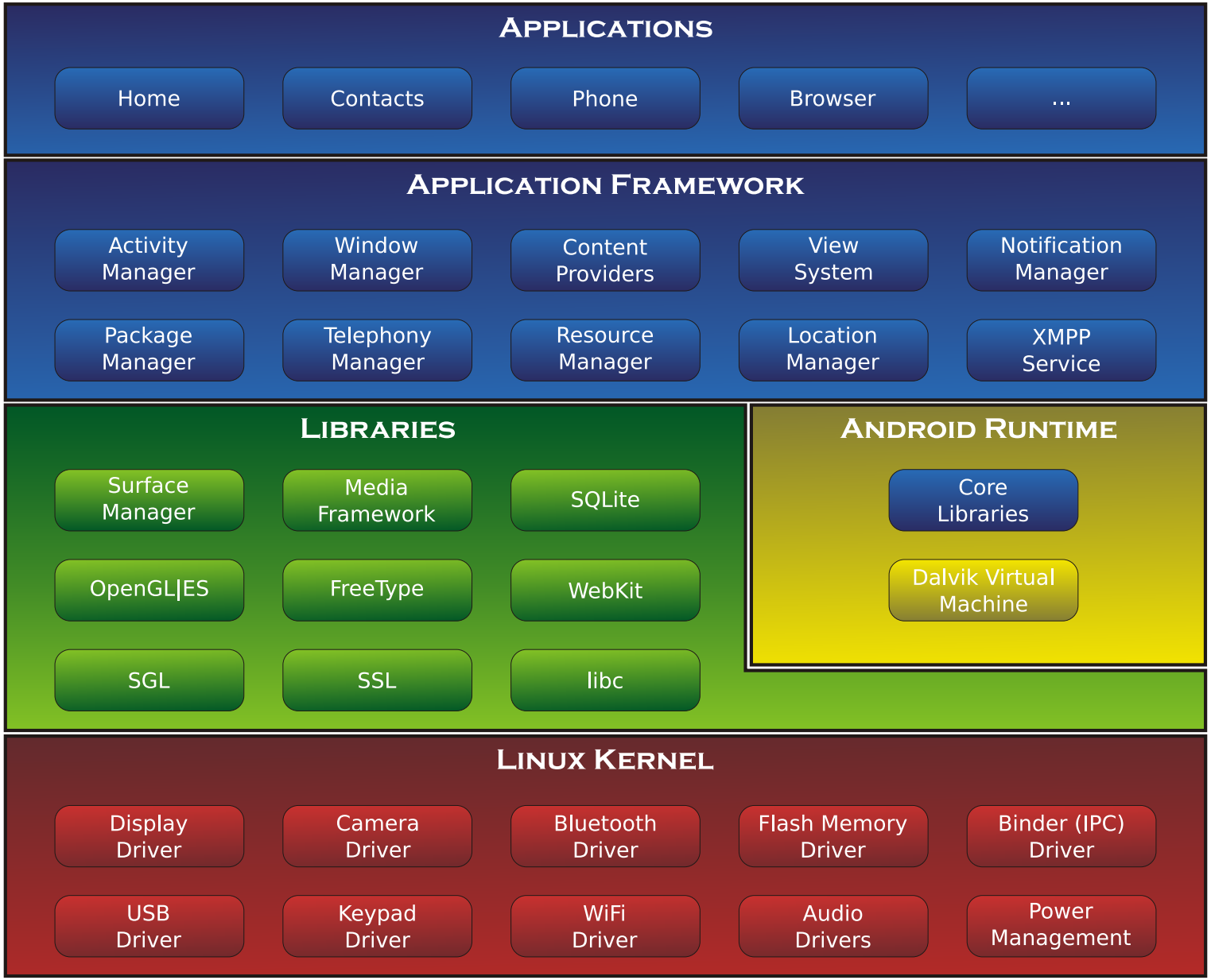
Algumas conclusões...

... sobre processamento de áudio em tempo real em GPU

- Tempo de transferência de memória é comparável ao da FFT.
- Diferenças na implementação fazem diferença:
  - Consulta a tabela (truncada, ou com interpolação linear ou cúbica).
  - Consulta a tabela em memória de textura.
  - Função seno da API.

Android

Sistema operacional Android



O projeto Android

- Kernel do Linux.
- Drivers para muitos dispositivos.
- Aplicativos e API em Java (máquina virtual própria).
- Conectividade e sensores.
- Licenciamento livre (com exceção de drivers).

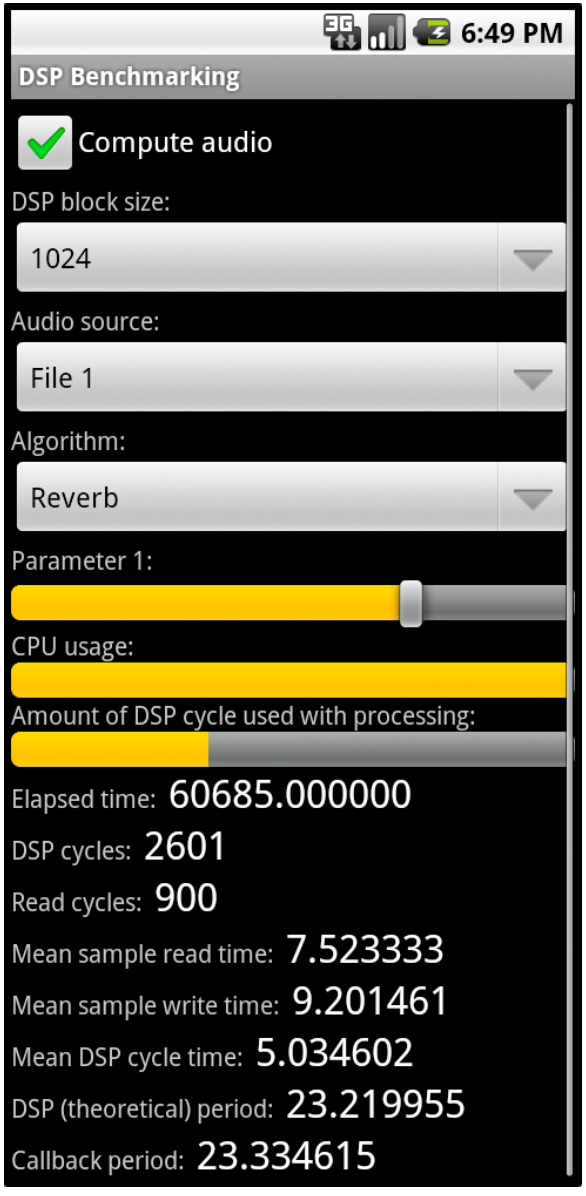
Idiossincrasias do sistema Android

- Java (JNI incluso).
- API de alto nível para:
- acesso ao hardware de áudio.
- agendamento de funções de manipulação.
- Programação no espaço de usuário.

Processamento em tempo real em Android

- Entrada de áudio:
  - Classe: `AudioRecord`.
  - Microfones, áudio da chamada (garante 1 canal, 16 bits).
- Processamento:
  - Método: `scheduleAtFixedRate`.
  - Agendamento.
- Saída de áudio:
  - Classe: `AudioTrack`.
  - 8 e 16 bits.

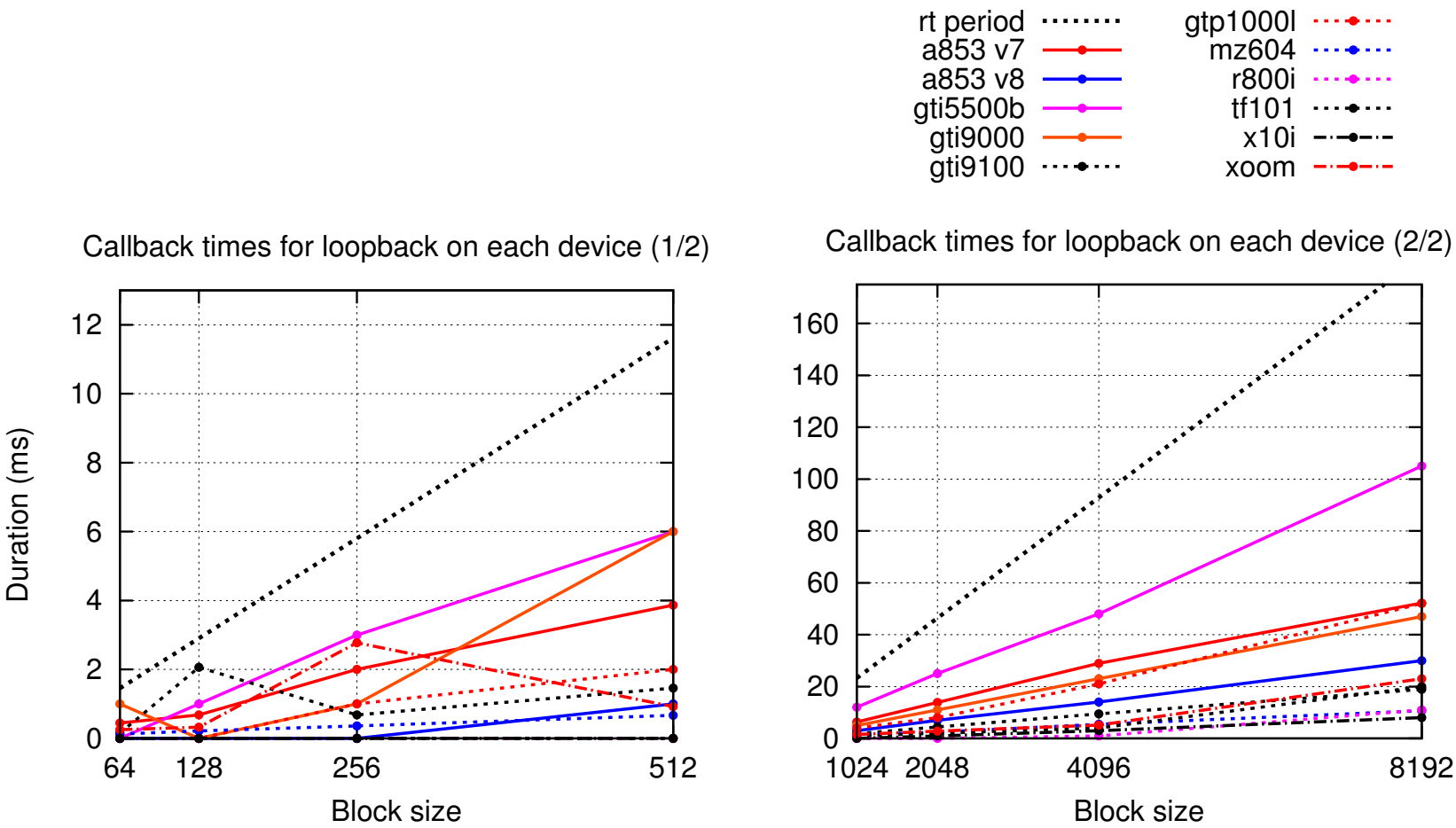
Aplicativo para Android e testes



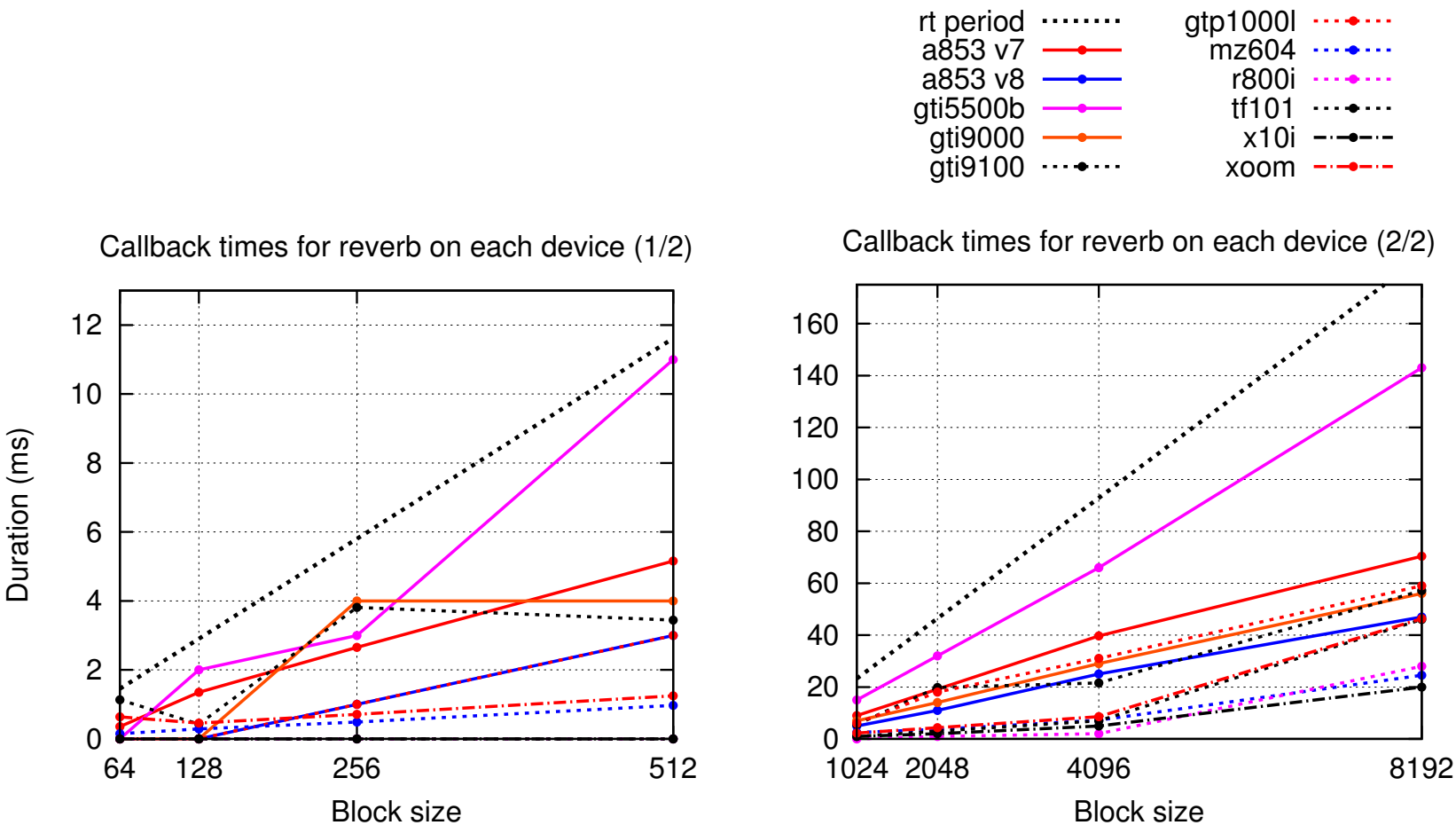
Cenário dos testes:

- Chamado por email.
- Instruções para execução do teste.
- Envio dos resultados por email.
- 11 aparelhos testados.
- Menor número de ciclos de teste para blocos grandes.

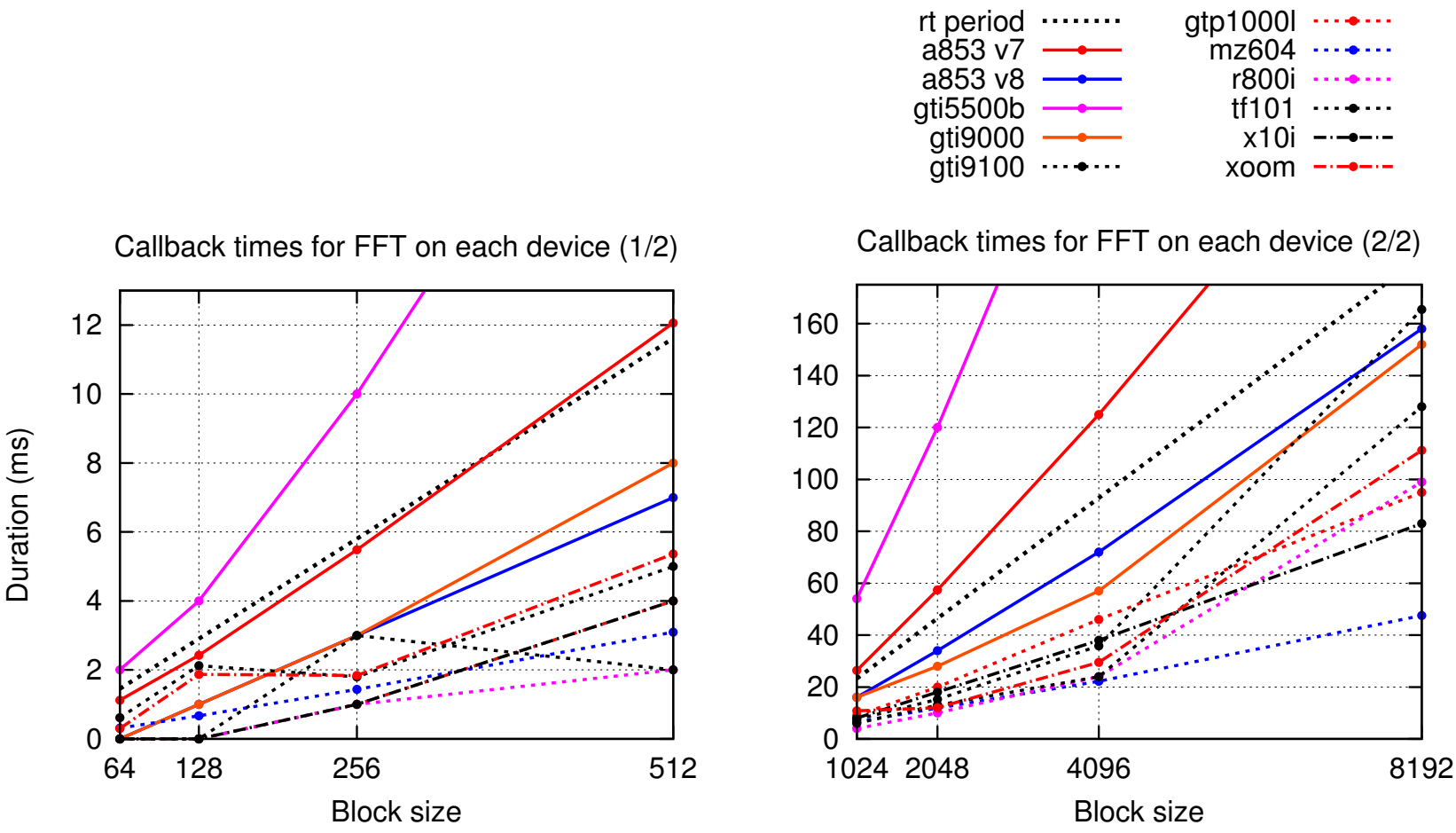
Android: Loopback em diferentes aparelhos



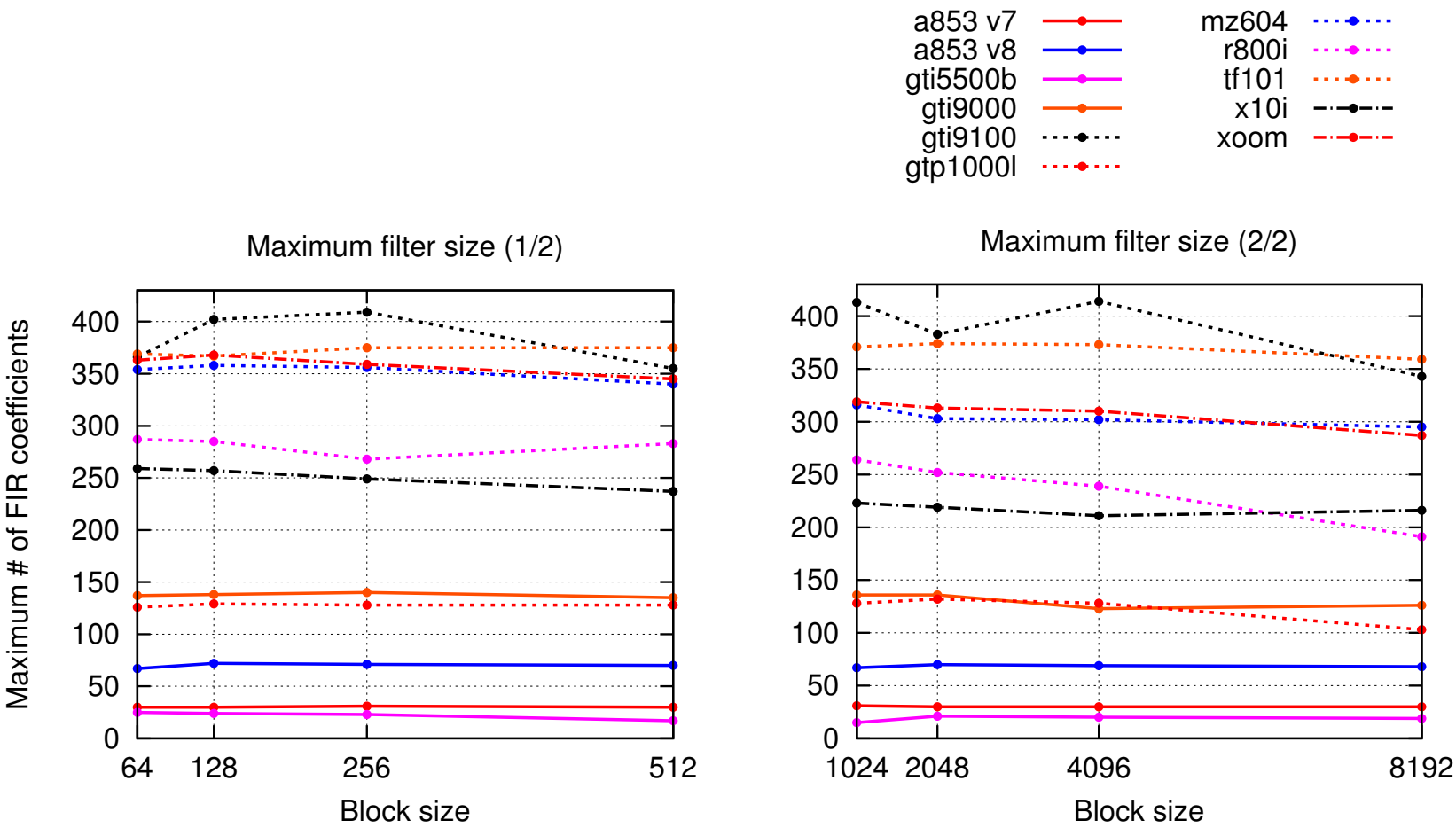
Convolução no Android: Filtro IIR em diferentes aparelhos



Convolução no Android: FFT em diferentes aparelhos



Android: ordem máxima de um filtro



Algumas conclusões. . .

. . . sobre processamento de áudio em tempo real em Android:

- O modelo de aparelho faz diferença significativa no desempenho.
- O sistema desenvolvido pode ser usado para avaliação do desempenho (em tempo real ou não).

Conclusões gerais

- Pode-se usar a metodologia apresentada para avaliação de qualquer dispositivo computacional.
- Não acreditem em preconcepções sobre os modelos computacionais e dispositivos.

Fim!

Obrigado pela atenção!

- Contato: `ajb@ime.usp.br`
- Grupo de Computação Musical do IME/USP: `http://compmus.ime.usp.br/`
- Esta apresentação: `http://www.ime.usp.br/~ajb/`

Crédito aos autores das imagens

- Convolução no domínio do tempo: Blanchardj - http://en.wikipedia.org/wiki/File:FIR_Filter.svg
- Síntese aditiva: Chrisjohnson - http://en.wikipedia.org/wiki/File:Additive_synthesis.svg
- FFT: Virens - http://en.wikipedia.org/wiki/File:DIT-FFT-butterfly.png
- Arduino: domínio público.
- PWM: CyrilB - http://en.wikipedia.org/wiki/File:Pwm.svg
- NVIDIA GPU: http://www.nvidia.com/
- Camadas do sistema operacional Android: Smieth - http://en.wikipedia.org/wiki/File:Android-System-Architecture.svg
- Software de análise de desempenho no Android: domínio público.