Processamento digital de sinais em tempo real utilizando Arduino

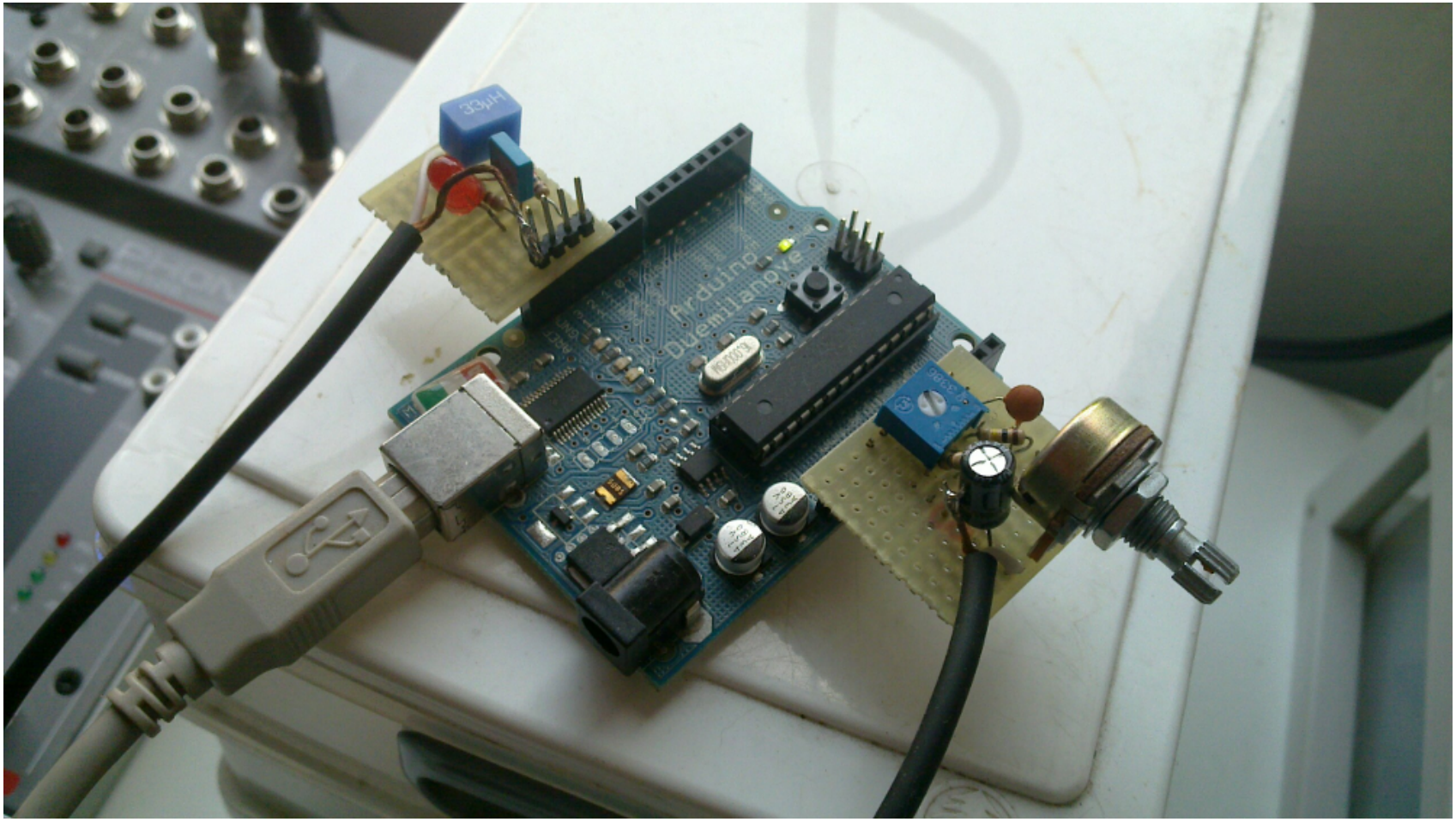
André Jucovsky Bianchi
ajb@ime.usp.br

Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

9 de outubro de 2012

Estrutura da apresentação

Arduino



Arduino

Características do projeto

- Estrutura minimal para interface com um microcontrolador.
- Processing (MIT 2001) + Wiring (Ivrea 2003) $\rightarrow$ Arduino (Ivrea 2005).
- Geralmente usado como interface para controle.
- Baixo custo: 20-50 USD.
- Licenciamento livre:
  - Projetos de hardware: CC BY-SA 2.5.
  - Software: GPL (IDE) e LGPL (bibliotecas C/C++).
  - Documentação: CC BY-SA 3.0.
- Comunidade.
- Mobilidade.
- Expansibilidade.

Microcontroladores Atmel AVR (ATmega328P)

- CPU: unidade aritmética e registradores (16 MHz - 8 bits).
- Interrupções.
- Memórias: Flash (32 KB), SRAM (2 KB) e EEPROM (1 KB).
- Relógios de sistema (diversas fontes, pré-escalonadores).
- Gerenciamento de energia.
- Portas digitais de entrada e saída.
- Contadores (com PWM).
- Interface serial.
- Conversão analógico-digital.
- *Boot-loader* e autoprogramação.

Processamento Digital de Sinais de Áudio em tempo real

Restrição de tempo máximo para o cálculo do resultado:

- Período do bloco de processamento: N amostras.
- Frequência de amostragem: R Hz.
- Período do ciclo DSP: $T_{DSP} = \frac{N}{R}$ s.

Perguntas:

- Qual é o número máximo de operações que se pode realizar em tempo real?
- Quais detalhes de implementação fazem diferença?
- Qual é a qualidade do sinal de áudio resultante?

Estrutura da apresentação

Conversor analógico-digital (ADC)

Características do ADC no ATmega328P:

- Amostragem:
  1. *Sample and hold*.
  2. Aproximação sucessiva.
- Resolução: 8 ou 10 bits.
- Tempo de conversão: 13 a 260 μs.
- Frequência própria / redução de ruído.
- Conversão manual ou automática.

Conversor analógico-digital (ADC)

Medição do tempo de conversão, usando diferentes valores de pré-escalonador (frequência principal: 16 MHz):

pré-escalonador	f_{ADC} (KHz)	T_{ADC} (μs)	$\tilde{T}_{\text{conv}}$ (μs)	$\tilde{f}_{\text{conv}}$ ($\approx$Hz)
2	8.000	0,125	12,61	79.302
4	4.000	0,25	16,06	62.266
8	2.000	0,50	19,76	50.607
16	1.000	1	20,52	48.732
32	500	2	34,80	28.735
64	250	3	67,89	14.729
128	125	8	114,85	8.707

Obs:

- Resolução da função `micros()`: 4 μs.
- Período de conversão: $\approx 14,5 \times T_{\text{ADC}}$.
- $R = 44.100$ Hz $\Rightarrow T_{\text{amostra}} \approx 22,67$ μs.
- $R = 31.250$ Hz $\Rightarrow T_{\text{amostra}} = 32,00$ μs.

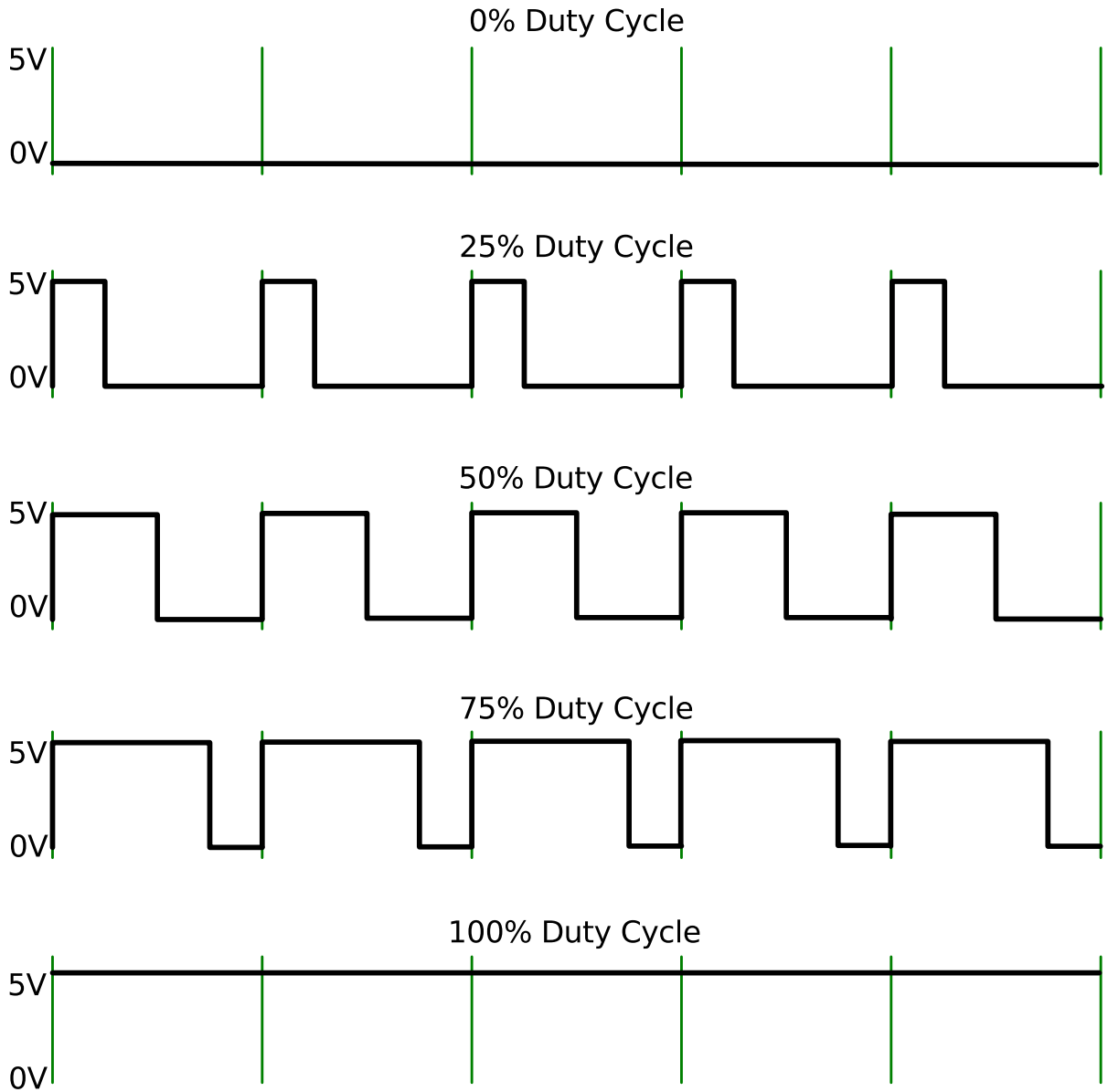
Conversor analógico-digital (ADC)

Parâmetros escolhidos:

- Conversão alinhada à esquerda (8 bits).
- Pré-escalonador igual a 8.

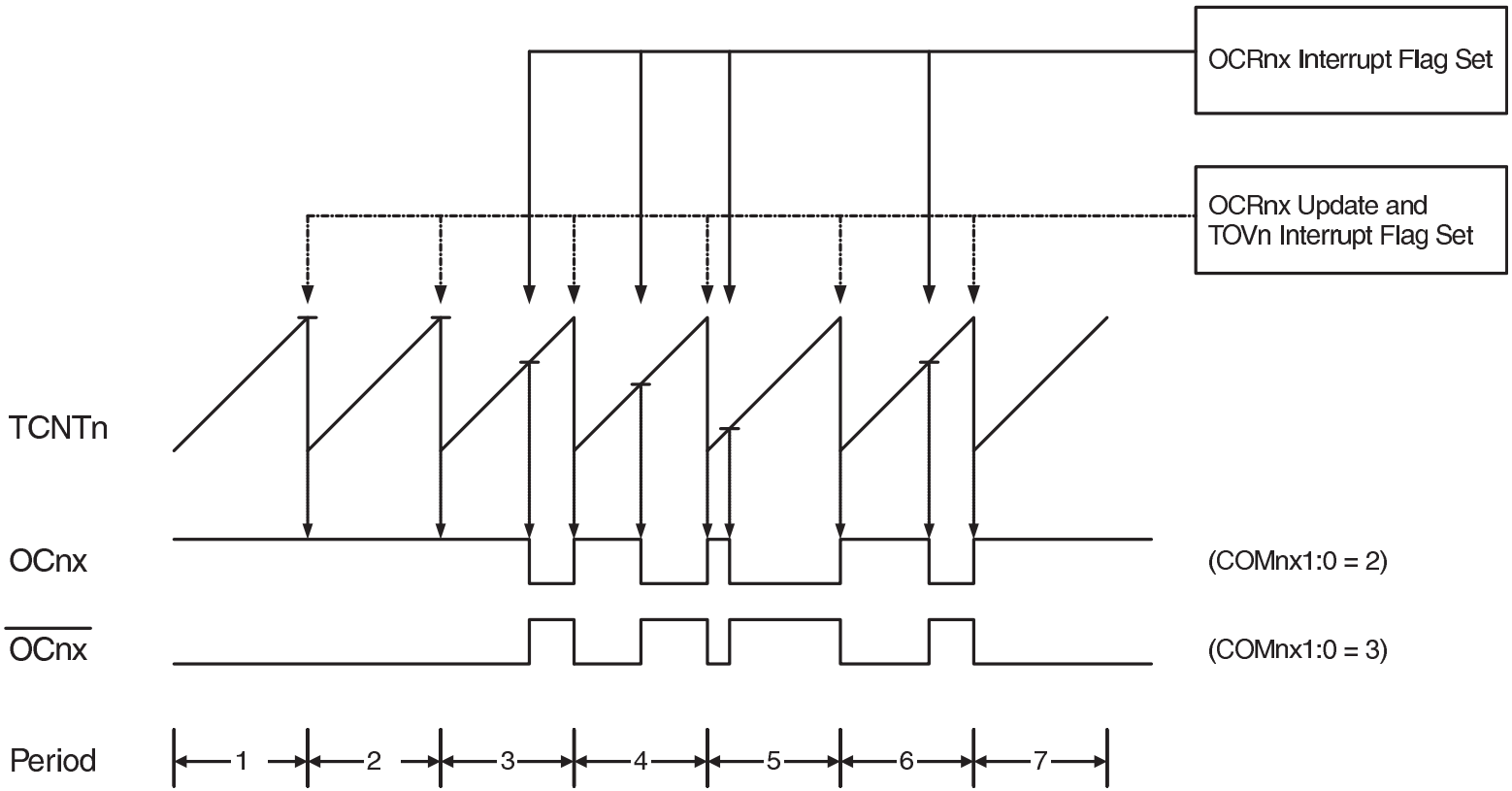
Estrutura da apresentação

Modulação por largura de pulso (PWM)

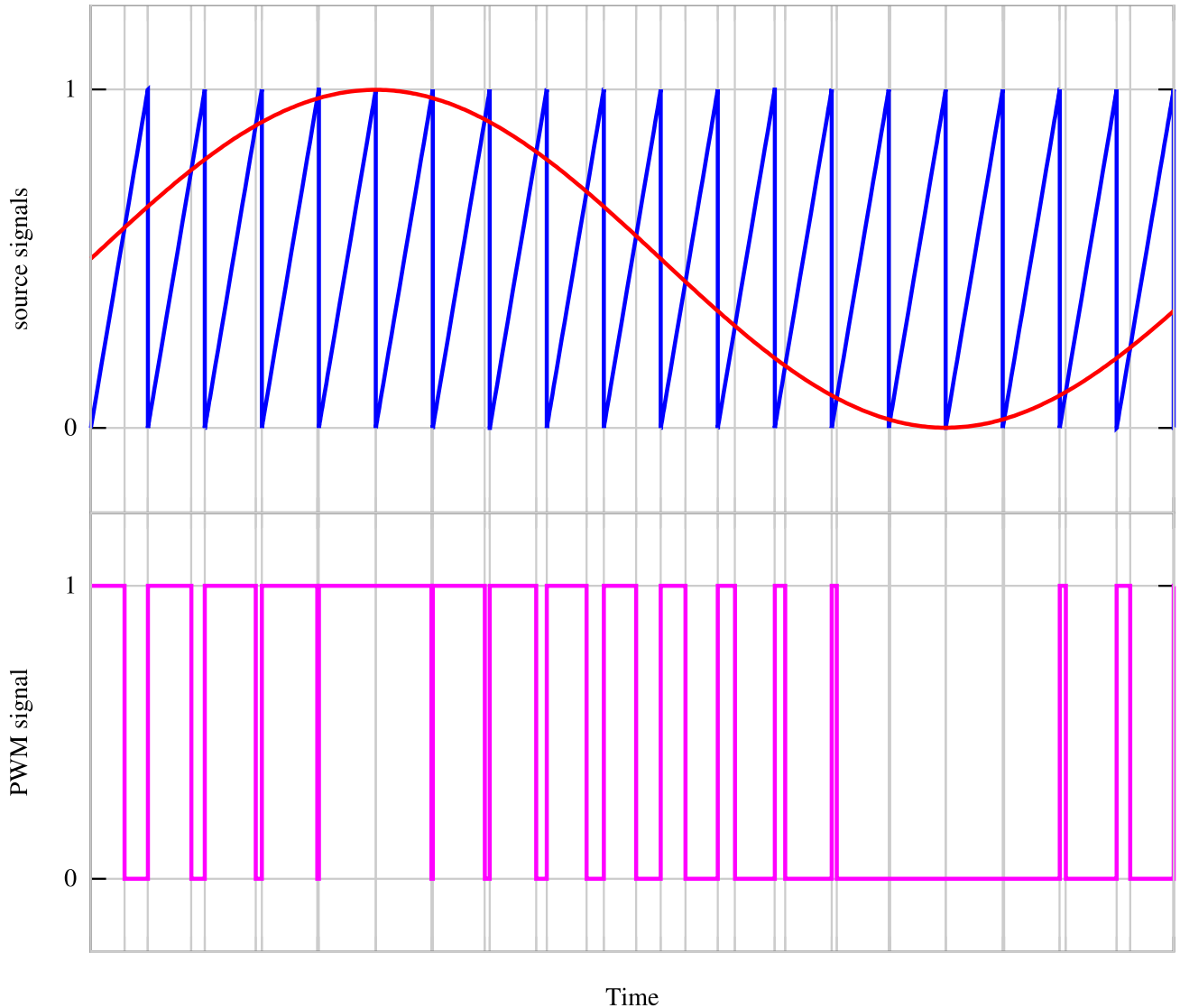


Modulação por largura de pulso (PWM)

Figure 15-6. Fast PWM Mode, Timing Diagram



Modulação por largura de pulso (PWM)



Modulação por largura de pulso (PWM)

Características de PWM no ATmega328P:

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

Modulação por largura de pulso (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

Modulação por largura de pulso (PWM)

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.

Estrutura da apresentação

Acoplamento de entrada e saída

```
1  // 1. leitura da entrada: ADC
2  x[ind] = ADCH;
3
4  // 2. escrita na saida: PWM
5  OCR2A = y[(ind-MIN_DELAY)&(BUFFER_SIZE-1)];
6
7  // 3. sinalizacao de um novo bloco de amostras
8  if ((ind & (BLOCK_SIZE - 1)) == 0) {
9    rind = (ind-BLOCK_SIZE) & (BUFFER_SIZE-1);
10   dsp_block = true;
11 }
12
13 // 4. incremento do indice de leitura/escrita
14 ind++;
15 ind &= BUFFER_SIZE - 1;
16
17 // 5. inicia uma nova conversao ADC
18 sbi(ADCSRA,ADSC);
```

Implementação

Detalhes importantes para implementar o sistema:

- ADC:
  - Valor do pré-escalonador.
  - Alinhamento do resultado (resolução).
  - Valor de referência.
  - Pino de entrada.
- PWM:
  - Modo *Fast PWM*.
  - Valor do pré-escalonador.
  - Pino de saída.
- Compilação: `avr-gcc`, `avr-g++`.
- Monitoramento serial: `minicom`.

Memória

Limites da Memória:

- 2 Kb de SRAM para dados.
- Uma tabela com 512 bytes ocupa $\frac{1}{4}$ da memória!
- Buffer máximo de 2.000 amostras.

Desempenho para processamento em tempo real

Perguntas:

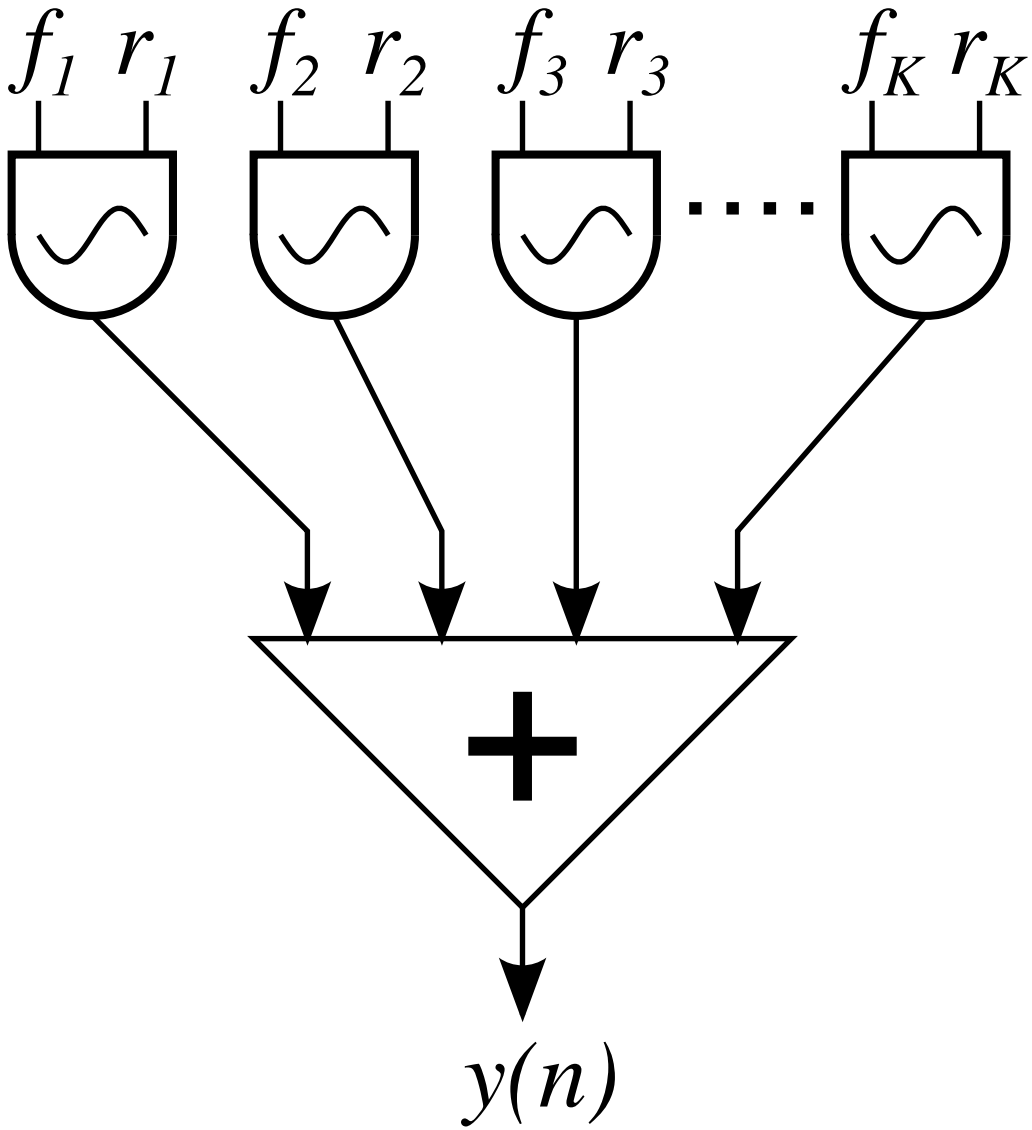
- Qual é número máximo de operações computáveis em tempo real?
- Quais detalhes de implementação fazem diferença?
- Qual é a qualidade do áudio resultante?

Implementações:

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

Estrutura da apresentação

Síntese aditiva



Síntese aditiva

Código em alto nível:

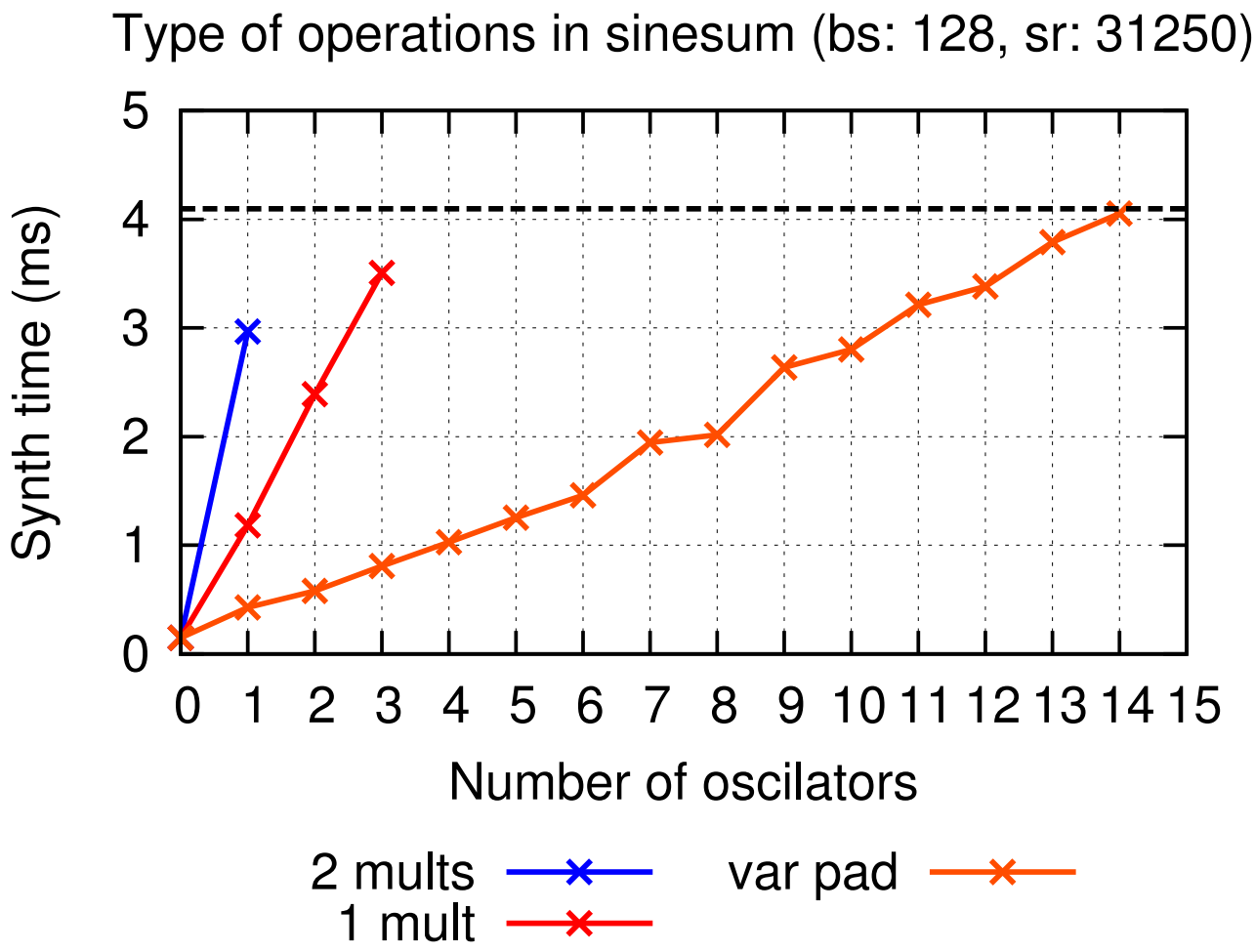
```
1 for (n = 0; n < N; n++)
2 {
3   angle = 2.0 * M_PI * t;
4   y[n] = 0.0;
5   for (k = 0; k < numFreqs; k++)
6     y[n] += r[k]*sin(f[k] * angle);
7   t += 1.0 / SR;
8 }
```

Implementação da linha 6:

```
1   ind[k] = (ind[k]+f[k]) & (SINETABLE_SIZE-1);
2   y[n&(BUFFER_SIZE-1)] += sine[ind[k]] >> pad;
```

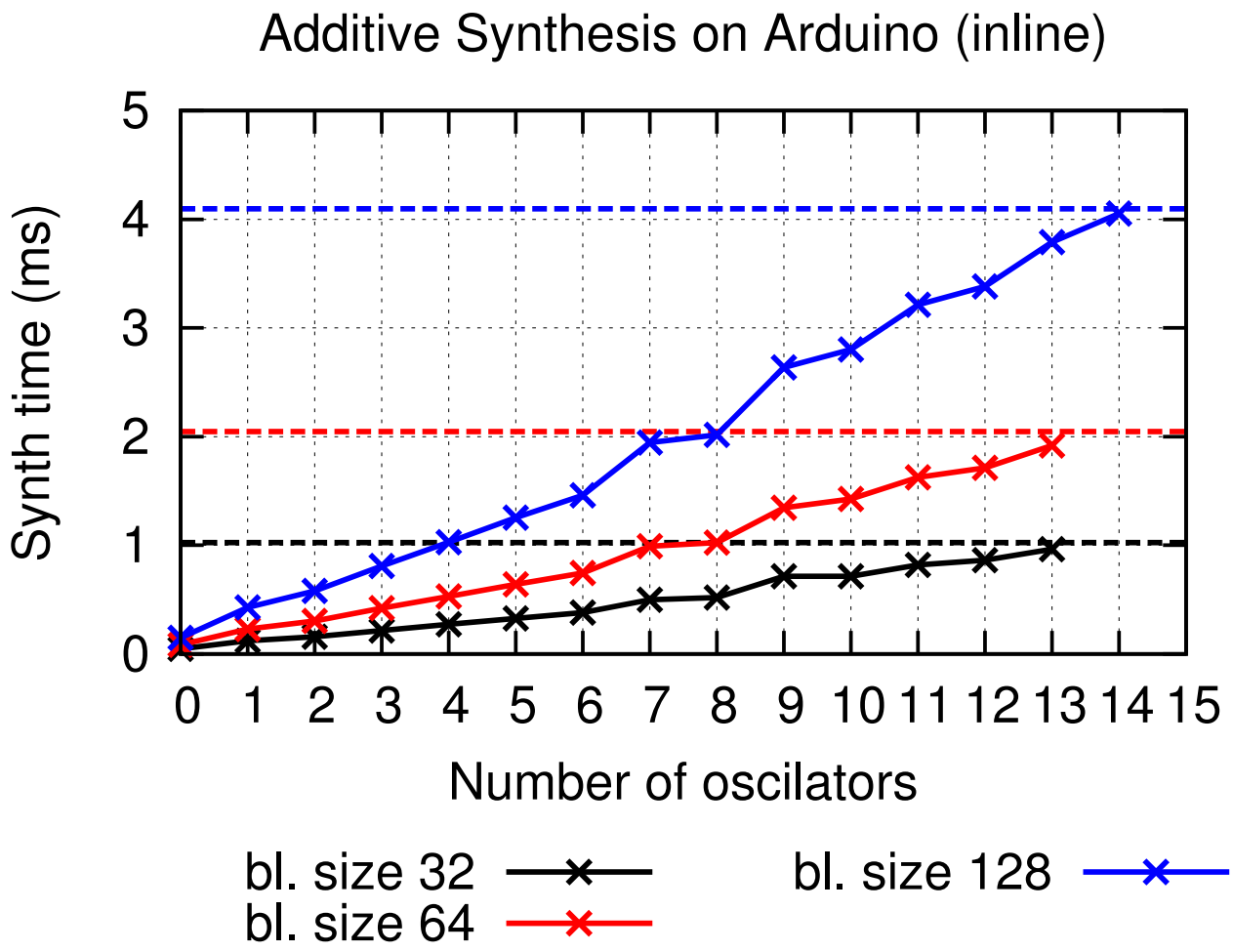
Síntese aditiva

Tipo e número de operações fazem a diferença



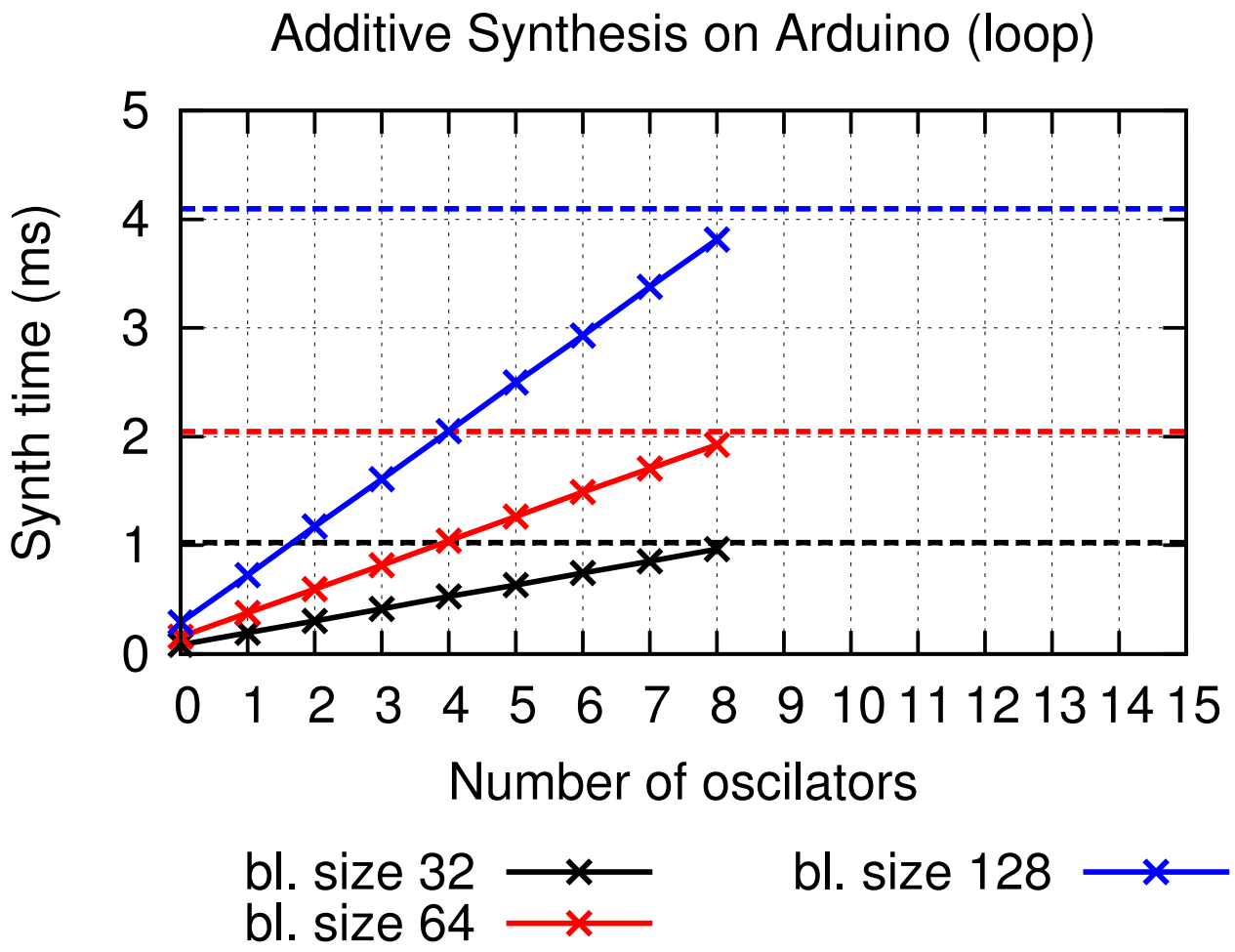
Síntese aditiva

Resultados para blocos de diferentes tamanhos



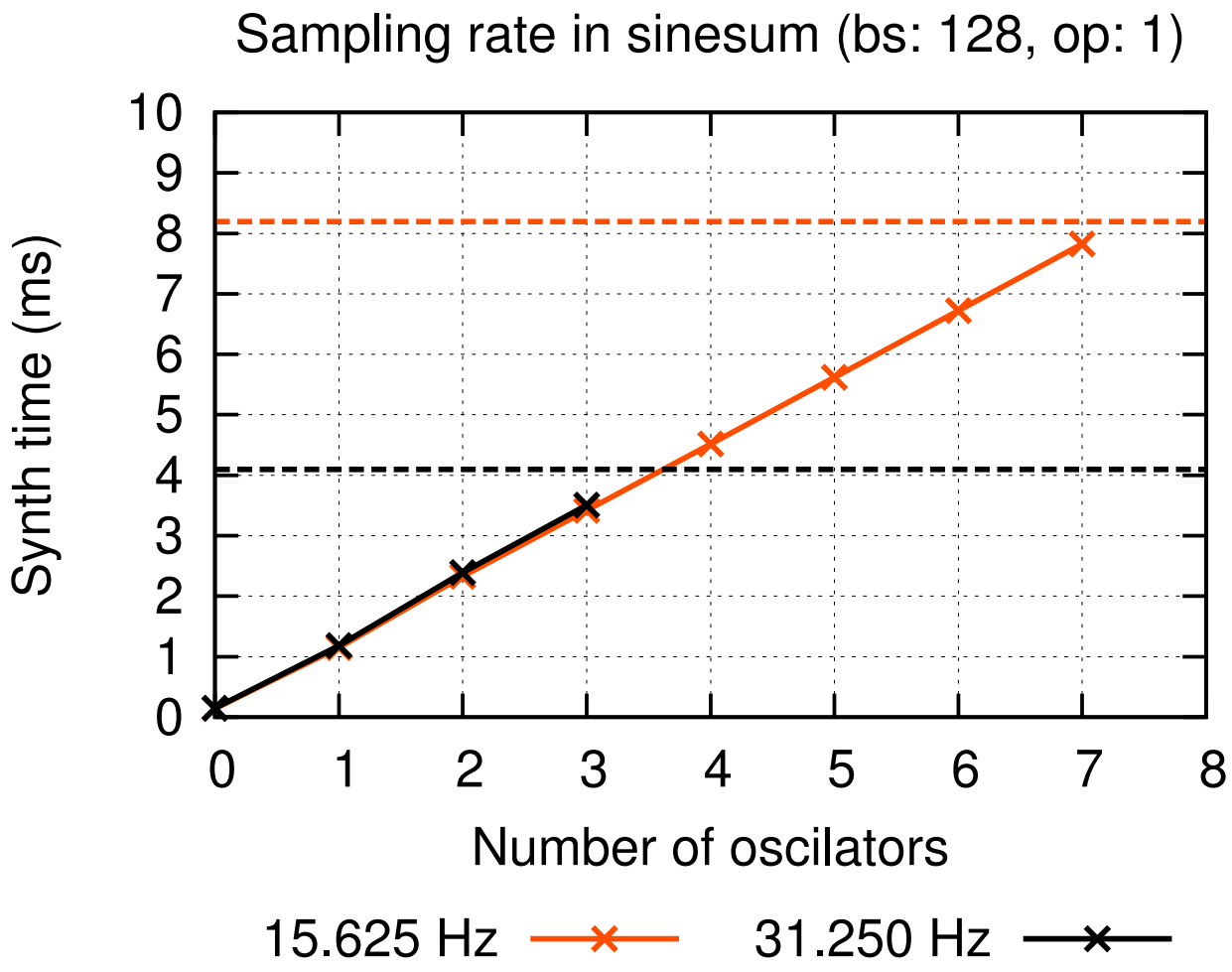
Síntese aditiva

Resultados para blocos de diferentes tamanhos



Síntese aditiva

Resultados para diferentes taxas de amostragem



Síntese aditiva

Resumo dos resultados

Número de osciladores máximo em cada cenário ($R = 31.250$ Hz):

block size	2op	1op	pad+for	pad
32	2	4	8	14
64	2	4	8	14
128	2	4	8	15

Síntese aditiva

Resumo dos resultados

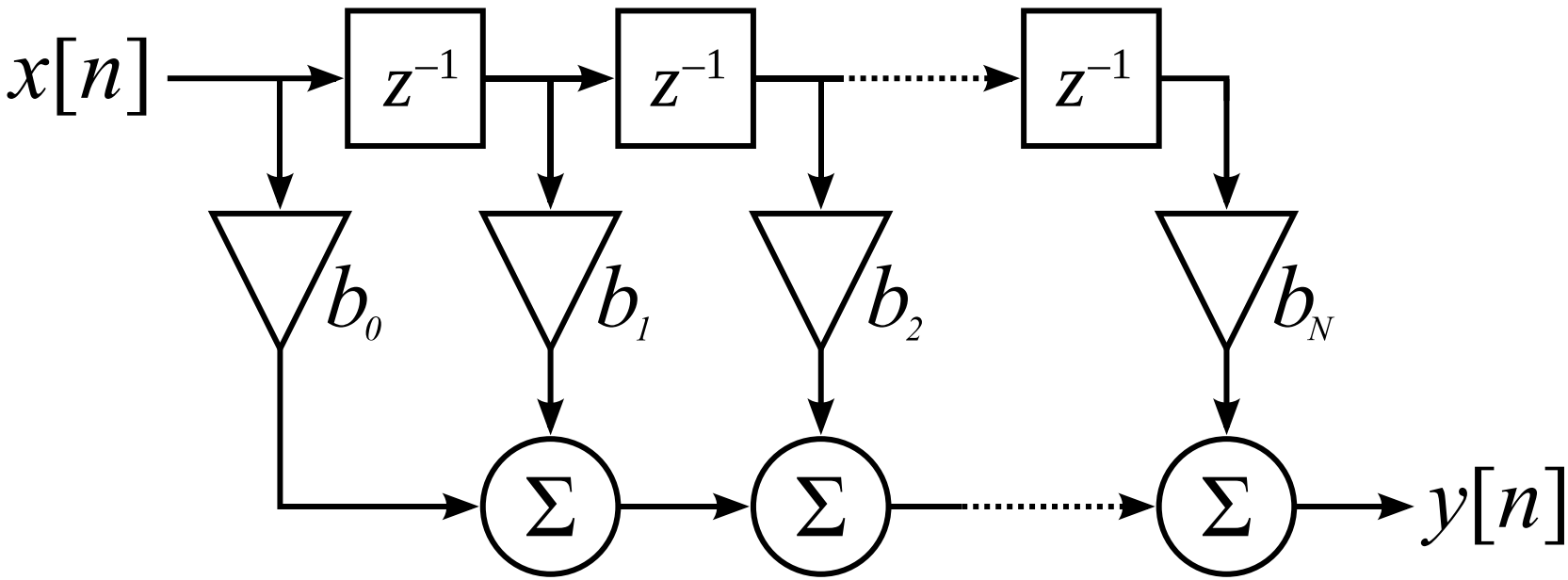
Número de osciladores máximo em cada cenário ($R = 31.250$ Hz):

block size	2op	1op	pad+for	pad
32	2	4	8	14
64	2	4	8	14
128	2	4	8	15

- Exemplo: soma de harmônicos de 200 Hz.

Estrutura da apresentação

Convolução no domínio do tempo



Convolução no domínio do tempo

Qual o tamanho máximo de um filtro computável em tempo real?

Código em alto nível:

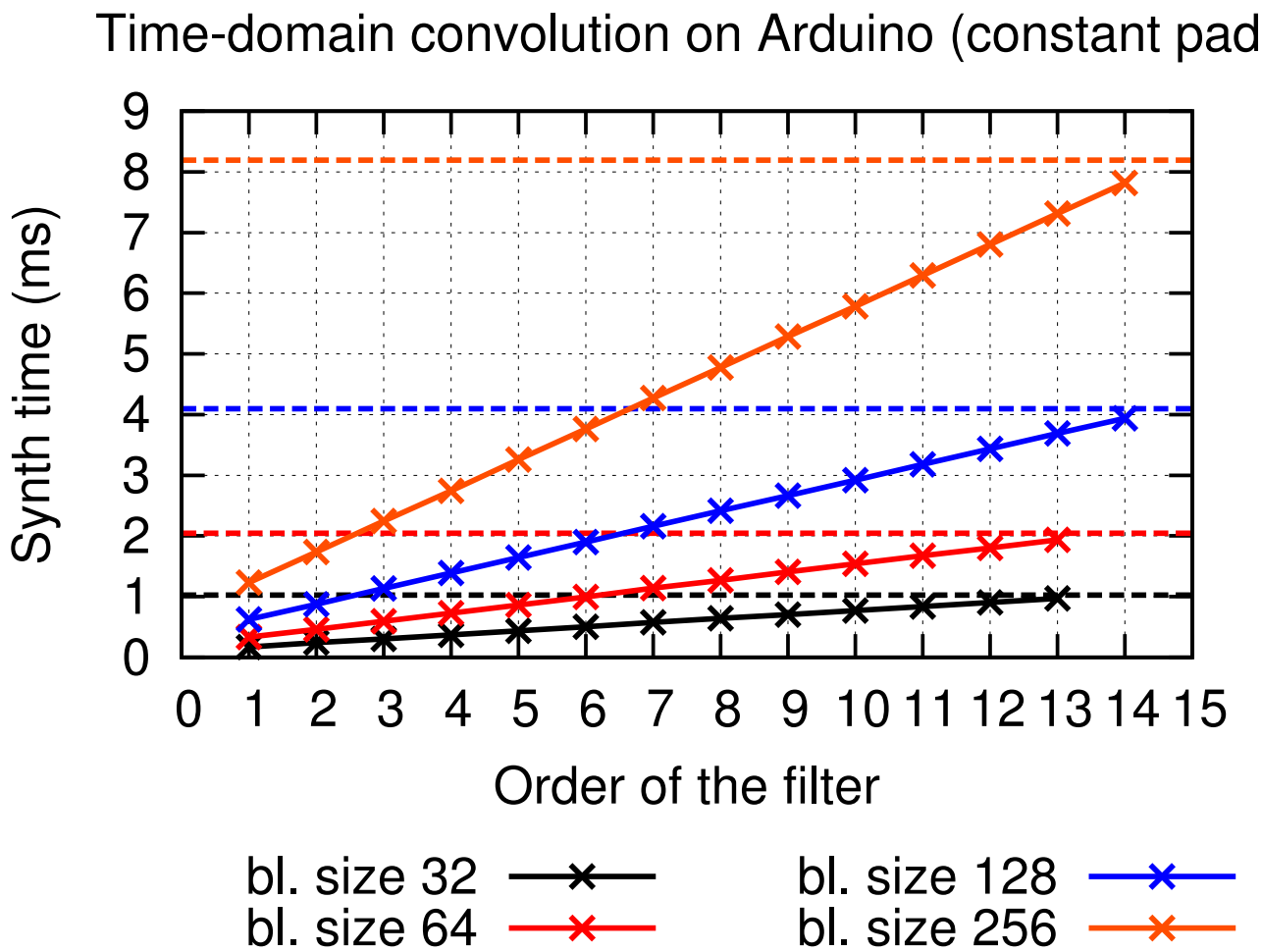
```
for (k = 0; k < N; k++)
  y[n] += b[k]*x[n-k];
```

Implementação:

```
for (int n = 0; n < N; n++) {
  int yn = 0, xtmp;
  for (int i = 0; i < order; i++) {
    xtmp = 127 - TMOD(x, n-i, BUFFER_SIZE);
    yn += xtmp * 10 / 100;
  }
  LIMIT(yn); /* limita a +- 127 */
  TMOD(y, n, BUFFER_SIZE) = 127 + yn;
}
```

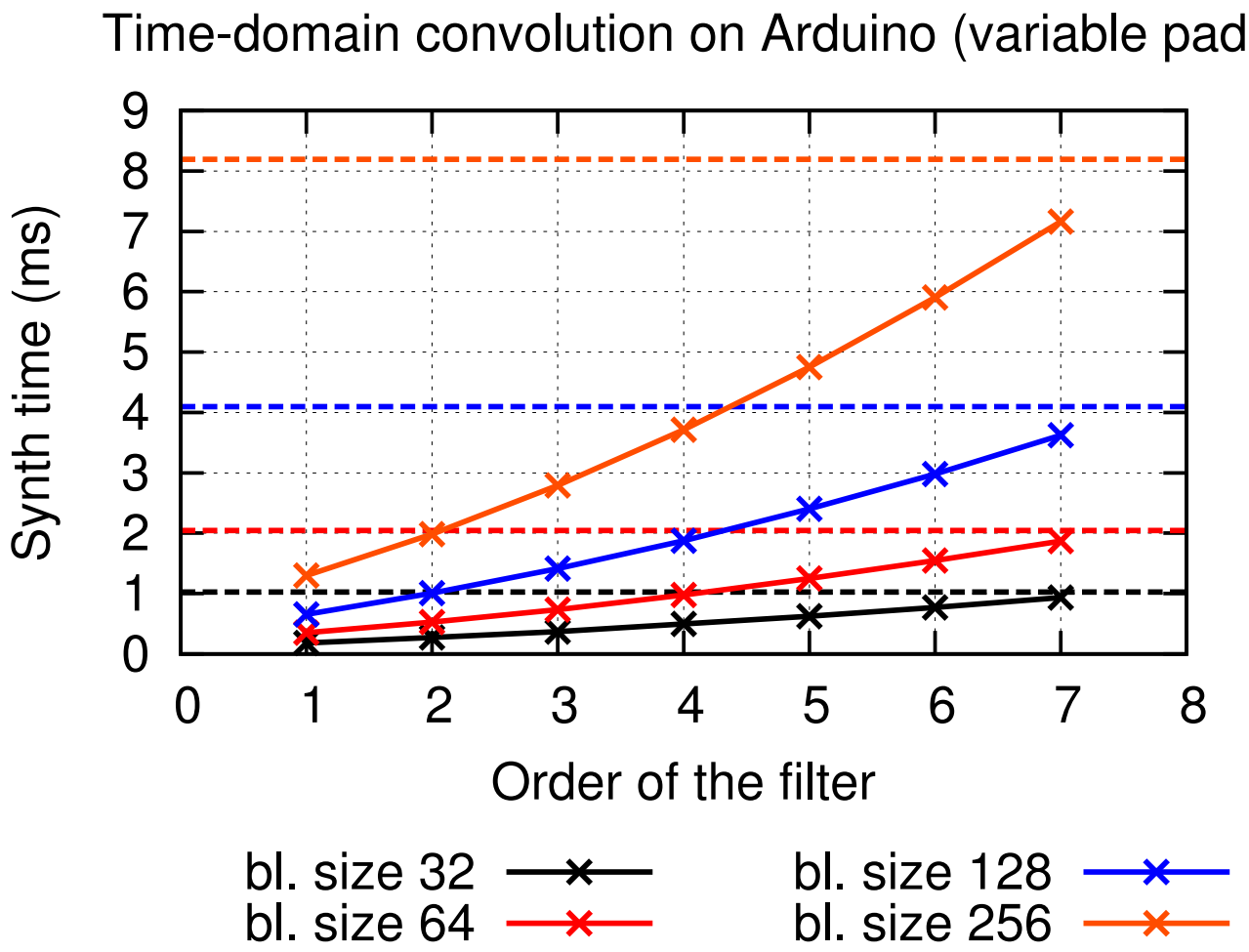
Convolução no domínio do tempo

Resultados para blocos de diferentes tamanhos



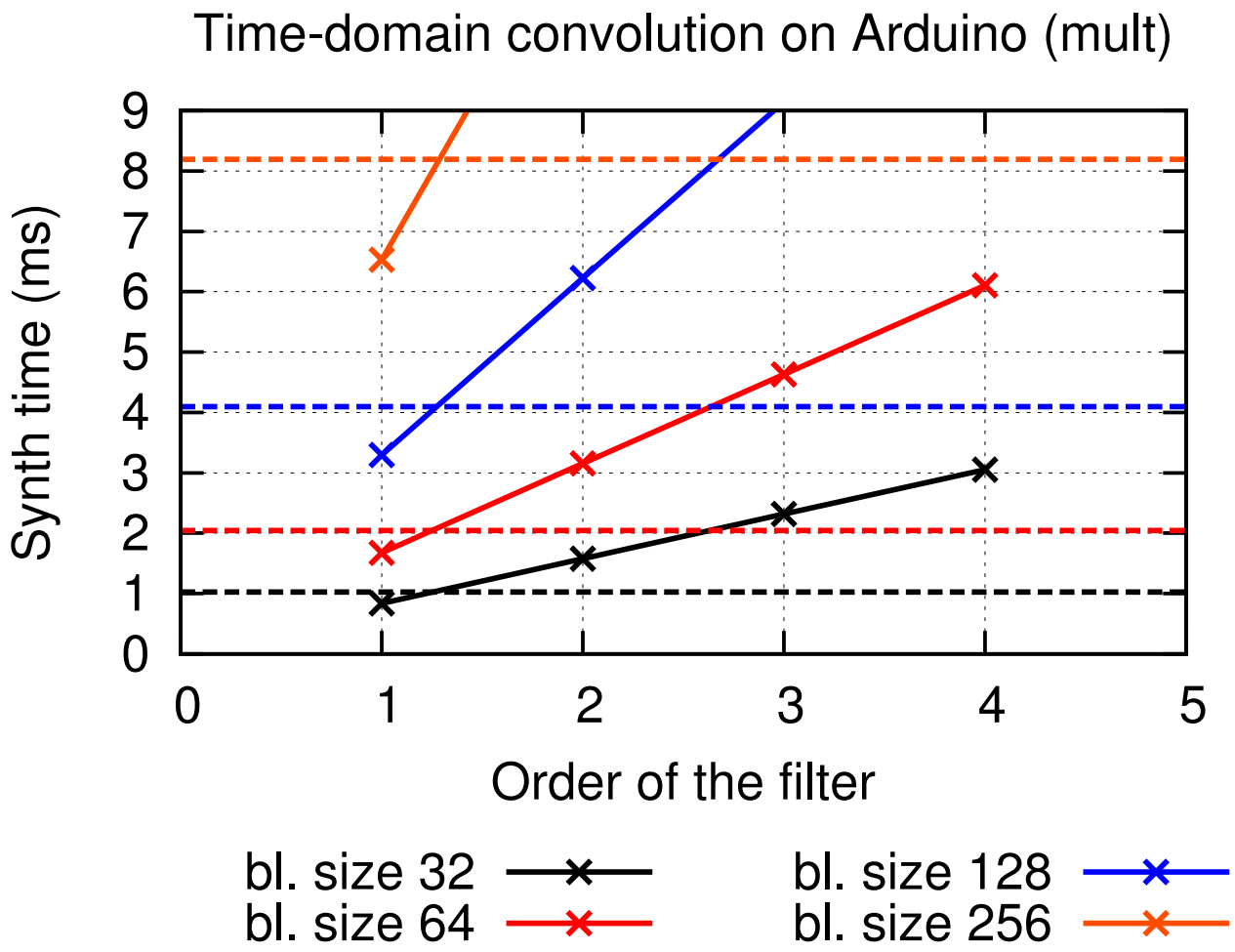
Convolução no domínio do tempo

Resultados para blocos de diferentes tamanhos



Convolução no domínio do tempo

Resultados para blocos de diferentes tamanhos



Convolução no domínio do tempo

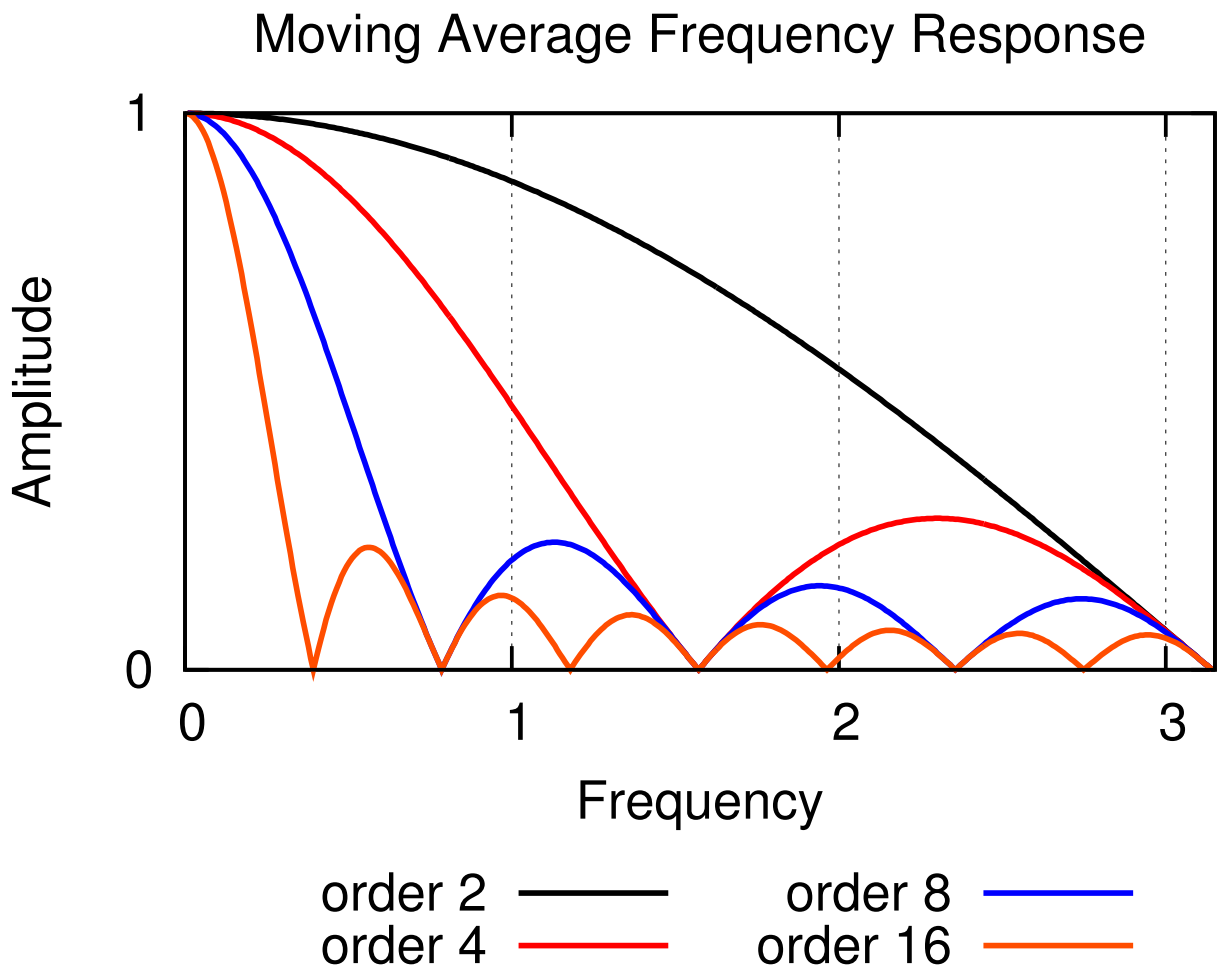
Resultados para blocos de diferentes tamanhos

Ordem máxima do filtro FIR em cada cenário ($R = 31.250$ Hz):

block size	multiplicação	pad variável	pad constante
32	1	7	13
64	1	7	13
128	1	7	14
256	1	7	14

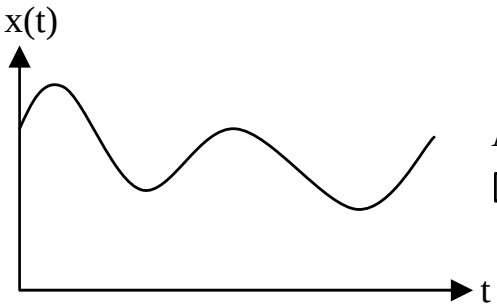
Convolução no domínio do tempo

Exemplo: moving average

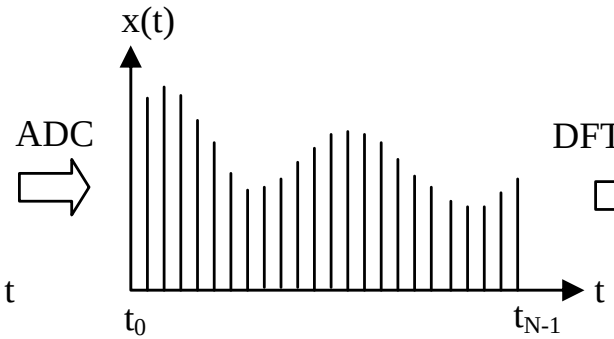


Estrutura da apresentação

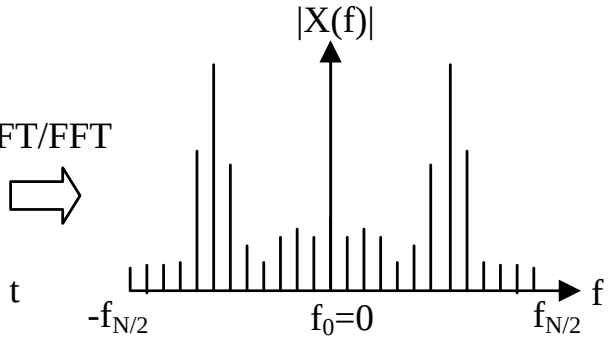
Fast Fourier Transform



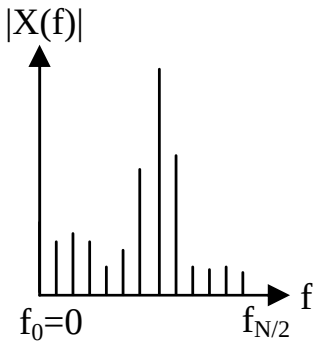
1) continuous signal in time domain



2) N points in time domain



3) N points in frequency domain containing both negative and positive frequency parts



4) N/2+1 points in amplitude/power spectrum

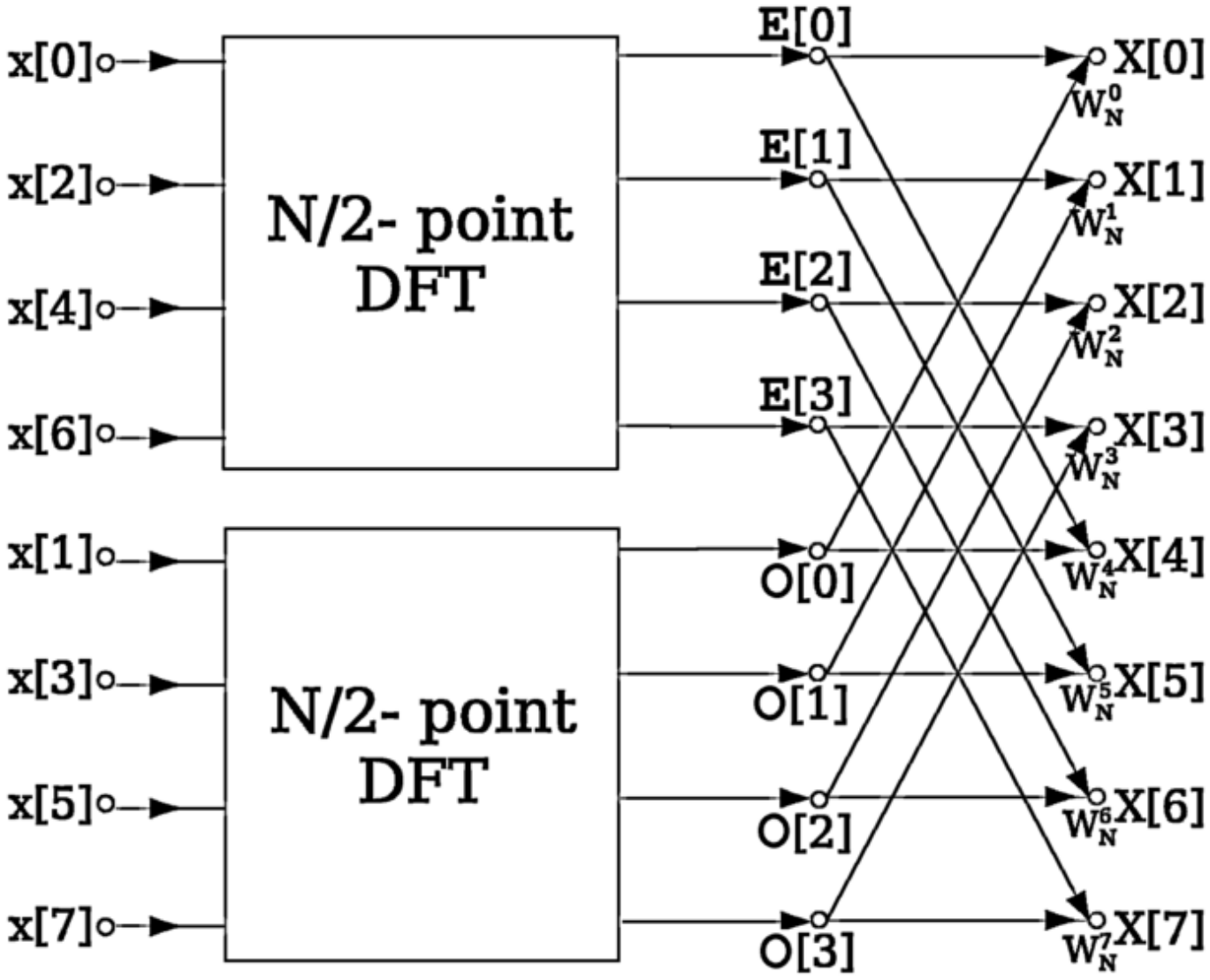
Fast Fourier Transform

A transformada discreta de Fourier (DFT) de um vetor de N pontos é:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}}, \qquad k = 1, \ldots, N-1.$$

- Implementação ingênua da DFT: $O(N^2)$.
- Implementações de FFT para diferentes valores de N: $O(N \log(N))$.

Fast Fourier Transform



Fast Fourier Transform

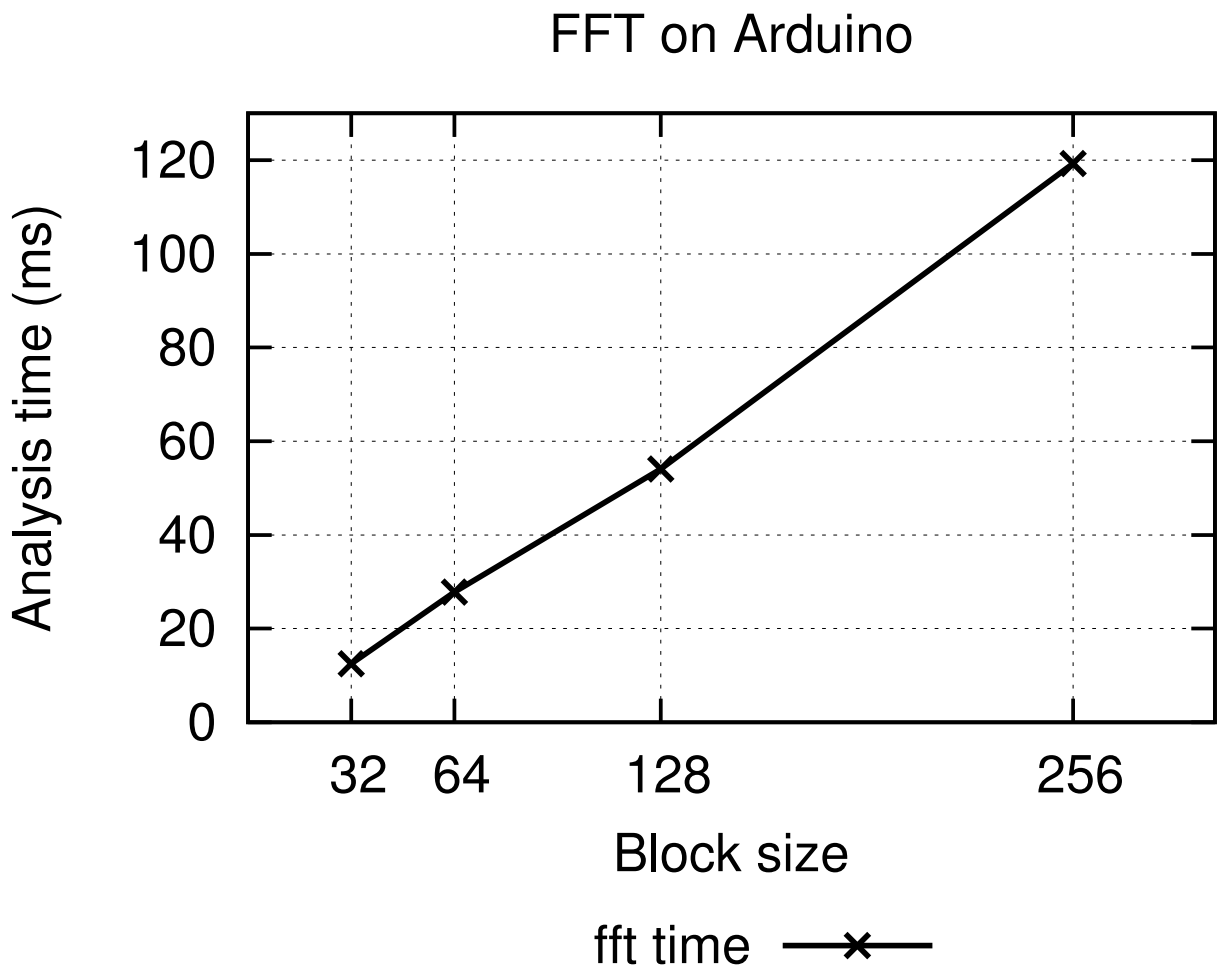
Código em altíssimo nível:

```
1 four1(x, N, 1); /* O(N*log(N)) */
```

- Qual é o tamanho máximo de uma FFT computável em tempo real?

Fast Fourier Transform

Resultados



Fast Fourier Transform

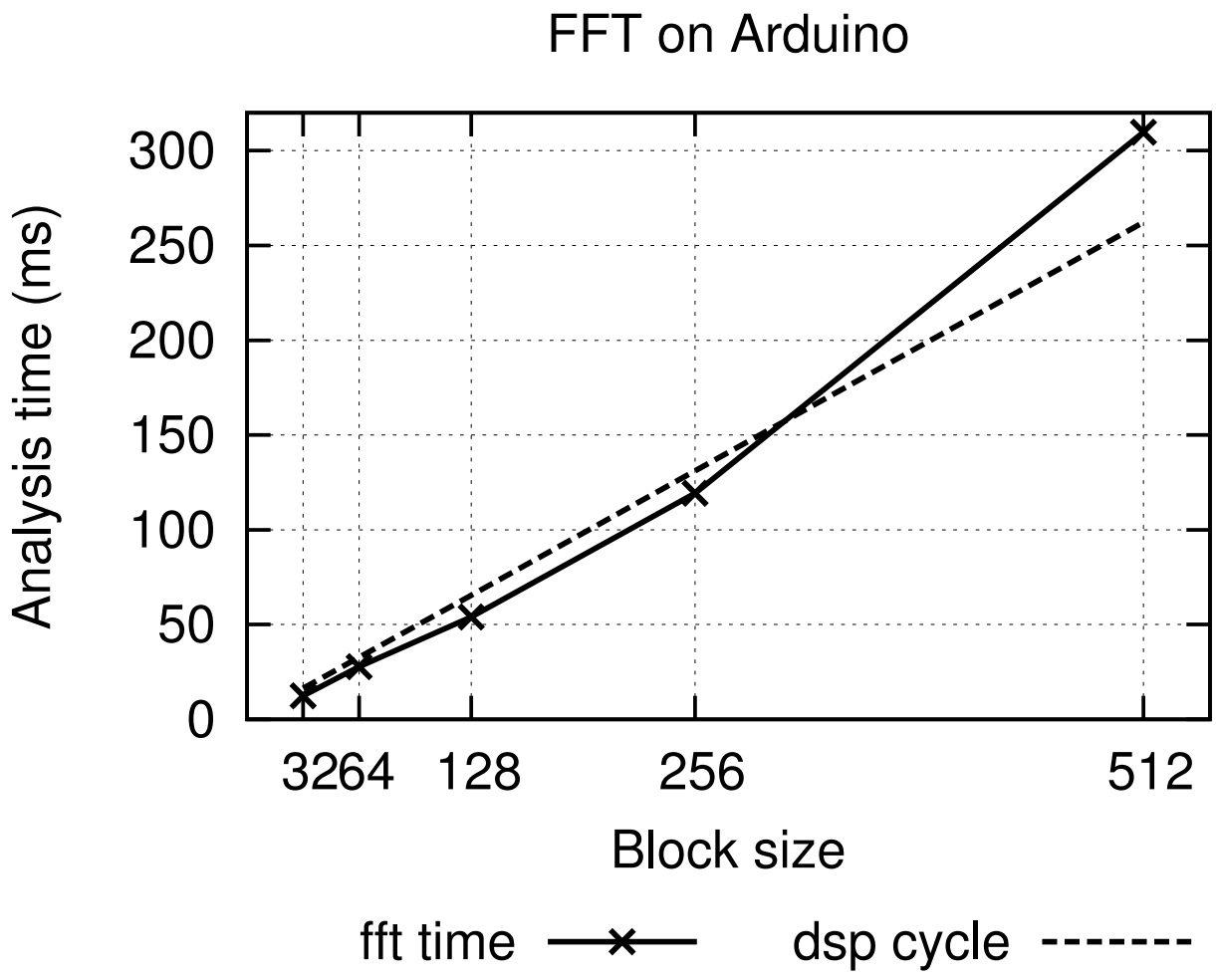
Resultados

Determinação de frequência máxima:

- Média de 428,15 μs por amostra.
- Frequência máxima $\approx$ 2.335 Hz.
- Pré-escalonador PWM de 32 $\Rightarrow$ $R = 1.953$ Hz.

Fast Fourier Transform

Resultados



Conclusões

Detalhes de implementação que fazem a diferença:

- 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.

Obrigado pela atenção!

Atribuição de autoria das figuras utilizadas:

- Figura PWM: Zurecs (zureks@gmail.com).
- Figura Síntese Aditiva: Chrisjonson.
- Figura FFT: Virens.

Dados de contato:

- Meu email: ajb@ime.usp.br
- Esta apresentação: http://www.ime.usp.br/~ajb/
- CM no IME: http://compmus.ime.usp.br/