EXPERIMENTOS DIDÁTICOS DE PROCESSAMENTO DIGITAL DE SINAIS USANDO MICROCONTROLADORES

L. S. Caires e J. R. A. Kaschny
Instituto Federal da Bahia - Campus Vitória da Conquista
leoscdm@gmail.com – kaschny@ifba.edu.br

RESUMO

Na presente contribuição apresentamos um conjunto de experimentos didáticos básicos relacionados ao processamento digital de sinais usando microcontroladores. Para isso, empregamos microcontroladores da família AVR e utilizamos a plataforma de desenvolvimento BASCOM para a criação do firmware. Tanto o hardware quanto o firmware foram concebidos para operação em tempo real. Apesar das limitações na frequência de operação e amostragem, os experimentos explorados ilustram os diversos aspectos sobre o tema, onde o aluno pode perceber as diversas limitações e problemas que surgem durante a concepção de um protótipo, alem da intima conexão entre hardware e software. Tais experimentos podem ser facilmente aplicados em atividades práticas nas disciplinas da área de processamento de sinais por um custo altamente convidativo.

Palavras-chave: microcontroladores, eletrônica digital, processamento digital de sinais.

1. INTRODUÇÃO

Nos dias atuais, a utilização de microcontroladores se faz cada vez mais presente nas mais inusitadas áreas. Tais dispositivos são utilizados em quase todos os equipamentos eletro-eletrônicos presentes em nosso cotidiano, possuindo as mais surpreendentes aplicações. Tal evolução tecno-mercadológica se deve aos grandes avanços da microeletrônica e dos diversos outros setores envolvidos na concepção e produção de dispositivos semicondutores. O baixo custo e a facilidade de aquisição de tais componentes facilitou incrivelmente a tarefa de desenvolvimento, que hoje pode ser efetuada em pequena escala por um custo razoavelmente baixo, incentivando assim diversos projetistas independentes.

Entre as diversas aplicações dos microcontroladores podemos salientar o processamento digital de sinais (DSP), que em boa parte são de origem analógica (LATHI, 2007). Atualmente existe uma grande variedade de microcontroladores e outros dispositivos dedicados a tal tipo processamento. Contudo, nem sempre temos os recursos e a infra-estrutura adequada para utilizá-los em atividades didáticas de cunho prático. De fato, existem diversos kits de desenvolvimento para DSP que são bastante sofisticados e altamente eficientes, com um custo proporcional a sua eficiência e sofisticação. Certamente este tipo de recurso seria o que poderíamos chamar de ideal. Por outro lado, é bem conhecido que há diversos problemas que podem surgir no caminho entre o ideal e o viável. Um deles diz respeito aos possíveis acidentes que tipicamente ocorrem durante aulas de laboratório. Essa não idealidade surge como parte do próprio processo de aprendizagem e que muitas vezes inutilizam completamente um equipamento.

Na presente contribuição apresentamos um conjunto de experimentos didáticos básicos relacionados ao processamento digital de sinais usando microcontroladores comuns. Apesar das limitações na frequência de operação e amostragem, os experimentos explorados ilustram os diversos aspectos sobre o tema, onde o aluno pode perceber as diversas limitações e problemas que surgem durante a concepção de um protótipo, alem da intima conexão entre hardware e software., podendo ser facilmente reproduzidos.

2. ELABORAÇÃO DOS EXPERIMENTOS

No presente trabalho nos baseamos no microcontrolador ATmega8, de 8 bits, que possui 8 kbytes de memória flash para armazenamento do programa - firmware, 512 bytes de memória EEPROM para dados estáticos, 1 kbyte de SRAM para armazenamento de variáveis e 6 conversores analógico-digital de 10 bits (ATMEL, 2011). Com tais características ficam evidentes as limitações que serão enfrentadas no que diz respeito à frequência de amostragem e, portanto, do sinal a ser processado. O diagrama esquemático básico é mostrado na figura 1. Como é possível observar, utilizamos um clock de 14.7456 MHz. A entrada de sinal analógico é conectada a um conversor analógico-digital (ADC) via um resistor. A saída de 8 bits, correspondente a porta D do ATmega8, é então conectada a uma rede resistiva R2R que servirá como conversor digital-analógico (DAC). Salienta-se aqui que a rede R2R foi elaborada usando resistores com 1% de tolerância com a intenção de reduzir o erro de conversão digital-analogica.

Em todos os experimentos o sinal a ser processado foi obtido a partir de um gerador de funções senoidal, sendo sua amplitude à cerca de 1 V pico a pico, acrescentado-se ainda um offset continuo de aproximadamente 0.5 V. Todo o processamento é efetuado em tempo real. A saída do conversor DAC foi sempre analisada usando um osciloscópio digital com comunicação USB, o que conferiu uma grande flexibilidade na aquisição de dados. Como esperado, os testes preliminares demonstraram claramente que o maior "gargalo" para o processamento está na amostragem do sinal de entrada, sendo causado principalmente pela baixa velocidade do ADC incorporado no microcontrolador. Além disso, é importante ressaltar o fato que estamos usando um ADC de 10 bits e um DAC de 8 bits. Portanto, os dois bits mais significativos oriundos da conversão analógico-digital são meramente desprezados. Desta forma, todo o processamento é efetuado usando uma palavra de 8 bits, compatível com o ATmega8.

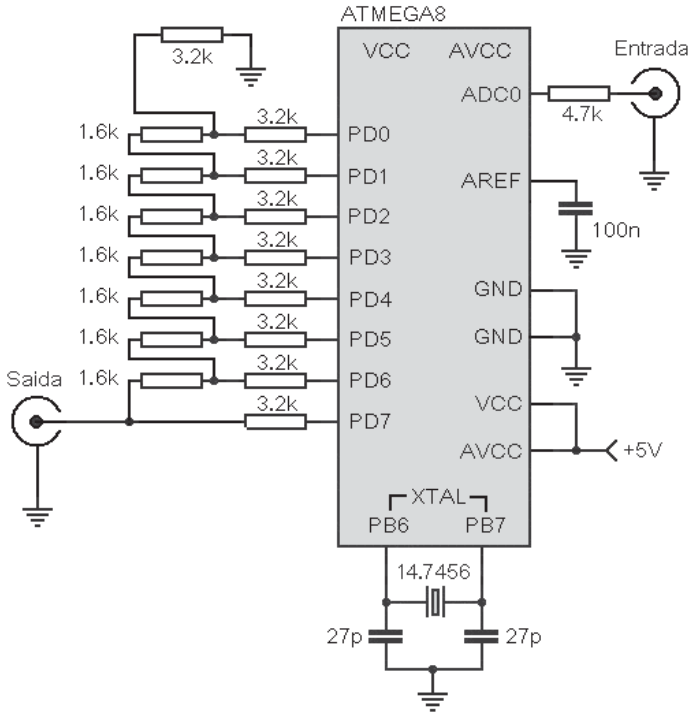

Figura 1 - Diagrama esquemático do hardware. Resistores em Ω e capacitores em F.

Baseando-se nesse circuito simples, o restante dos esforços é então concentrado na programação do microcontrolador. O firmware correspondente a cada experimento foi desenvolvido usando a plataforma BASCOM AVR (MCS ELECTRONICS, 2011). A tabela 1 mostra o bloco comum do código sendo a posição do núcleo do processo assinalado em vermelho. Portanto, no restante do presente texto iremos comentar somente os tópicos relativos a tais núcleos.

Tabela 1 - Bloco comum do firmware de cada experimento.

```
$regfile = "m8def.dat"
$crystal = 14745600
$baud = 115200
$hwstack = 32
$swstack = 16
$framesize = 32
Config Portd = Output
Config Adc = Single, Prescaler = Auto, Reference = Avcc
Start Adc
Dim Canal As Byte
Dim Valor As Byte
Canal = 0
<NÚCLEO DO PROCESSO>
End
```

Como exemplo é mostrado na figura 2 o resultado do núcleo de processo mais simples, ou seja, aquele onde é feita a leitura do ADC e seu valor movido para o DAC. O gráfico em vermelho mostra o sinal injetado na entrada e o em azul o sinal obtido na saída do DAC, para um sinal senoidal de 200 Hz.

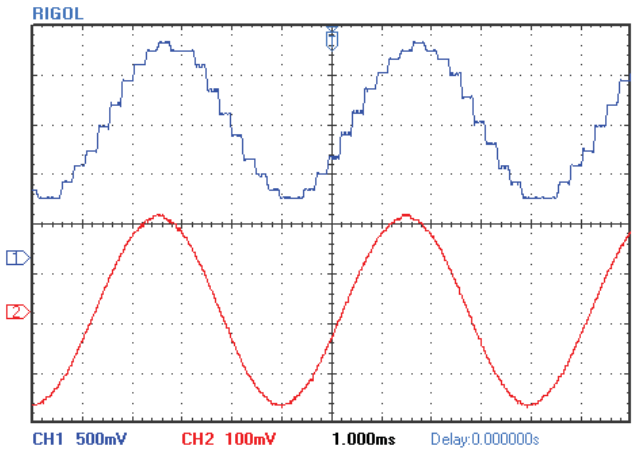

Figura 2 - Amostragem de um sinal senoidal de 200 Hz. A curva em vermelho ilustra o sinal injetado na entrada do ADC e a curva em azul o sinal obtido na saída DAC.

Observando com cuidado, podemos concluir a partir desta figura que em um ciclo do sinal senoidal é possível tomar cerca de 22 amostras. Isso fornece um tempo de aproximadamente 230 µs por amostra. Então, o teorema de Nyquist-Shannon (LATHI, 2007) será de fato obedecido até uma freqüência máxima de 46 Hz. Além disso, é possível constatar que o sinal obtido na saída do DAC esta atrasado cerca de 200 µs com relação ao sinal original.

3. FILTROS DIGITAIS

Para ilustrar a elaboração de filtros digitais, implementamos um algoritmo de filtro de média móvel representando um filtro exponencial de primeira ordem. Neste tipo de concepção, um filtro passa-baixa pode ser representado pela equação diferencial:

$$k_f \frac{dy(t)}{dt} + y(t) = u(t) \qquad (1)$$

onde y(t) é o sinal filtrado e u(t) o valor medido. Então via a aproximação:

$$\frac{dy(t)}{dt} \approx \frac{y(k) - y(k-1)}{\Delta k} \qquad (2)$$

obtemos:

$$y(k) = \beta y(k-1) + \alpha u(k) \quad \text{onde} \quad \beta = 1 - \alpha \quad \text{e} \quad \alpha = \Delta k / (k_f + \Delta k) \qquad (3)$$

Os códigos implementados com base neste resultado são listados na tabela 2. Salienta-se que no caso do filtro passa-alta a resposta do filtro foi obtida simplesmente subtraindo-se o sinal filtrado do sinal medido. Os resultados desses procedimentos são ilustrados nas curvas de resposta em frequência mostradas na figura 3, sendo estas obtidas a partir de medidas do sinal processado.

Como podemos ver na figura 3, ambos os filtros funcionam corretamente com uma taxa de atenuação relativamente baixa, tal como esperado, ficando a frequência de corte na faixa entre 40 e 60 Hz. Podemos também notar que devido às limitações de nosso circuito a faixa de frequências analisadas, ou melhor, manipulada, não passa dos 400 Hz. Acima disso, a taxa de amostragem se torna

Tabela 2 - Programas referentes aos filtros passa-baixa e passa-alta.

Filtro Passa-Baixa	**Filtro Passa-Alta**
Dim Ya As Single Dim Yb As Single Dim T1 As Single Dim T2 As Single Dim U As Single Dim Beta As Single Dim Alfa As Single Alfa = 0.1 Beta = 1 - Alfa Valor = Getadc(canal) Yb = Valor Do Valor = Getadc(canal) U = Valor T1 = Beta * Yb T2 = Alfa * U Ya = T1 + T2 Valor = Ya Portd = Valor Yb = Ya Loop	Dim Ya As Single Dim Yb As Single Dim T1 As Single Dim T2 As Single Dim T3 As Single Dim U As Single Dim Beta As Single Dim Alfa As Single Alfa = 0.1 Beta = 1 - Alfa Valor = Getadc(canal) Yb = Valor Do Valor = Getadc(canal) U = Valor T1 = Beta * Yb T2 = Alfa * U Ya = T1 + T2 T3 = U - Ya Valor = T3-128 Portd = Valor Yb = Ya Loop

demasiadamente reduzida para que esta represente, mesmo que remotamente, o sinal injetado. A análise comparativa do sinal obtido na saída do DAC com relação à entrada do ADC e a comparação com a situação ilustrada na figura 2, nos leva a concluir que a fração mais significativa do tempo de latência, ou seja, o gargalo do processamento, esta realmente na baixa velocidade do ADC e não na execução das instruções do núcleo de processamento. Isso é de fato bastante animador, pois indica que podemos implementar filtros de ordens mais altas, e mais elaborados, sem que o núcleo do processo tenha um peso demasiadamente significativo no processamento do sinal.

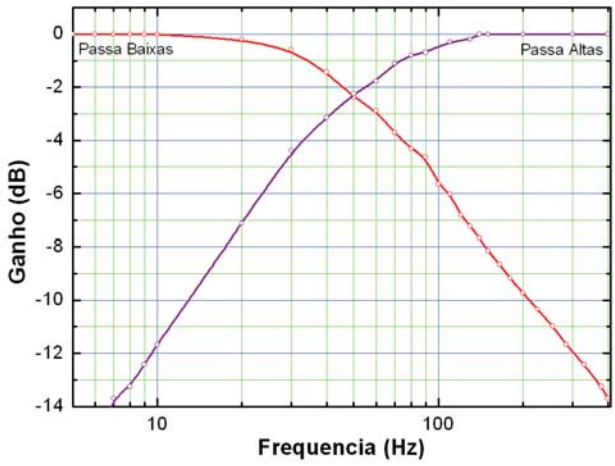

Figura 3 - Curva dos filtros implementados nos presentes experimentos.

4. APLICAÇÕES DIVERSAS

Algumas aplicações simples, porém ilustrativas, podem também ser facilmente implementadas. Como exemplo, temos na tabela 3 os códigos correspondentes a um retificador de onda completa, um ceifador (ou limitador) e um dobrador de frequência.

Tabela 3 - Programas referentes às rotinas para um retificador, um ceifador e um dobrador de frequências.

Retificador	**Ceifador**	**Dobrador**
Dim U As Single	Dim L1 as Byte	Dim U As Single
	Dim L2 as Byte	Dim U2 As Single
Do	L1=100	
Valor = Getadc(canal)	L2=20	Do
U = Valor - 128		Valor = Getadc(canal)
U = Abs(u)	Do	U = Valor - 128
Valor = U	Valor = Getadc(canal)	U2 = U * U
Portd = Valor	If Valor > L1 then Valor = L1	U = U2 / 256
Loop	If Valor < L2 then Valor = L2	Valor = U
	Portd = Valor	Portd = Valor
	Loop	Loop

Além dessas aplicações é também possível desenvolver um codificador de sinais, abrindo uma boa perspectiva para o desenvolvimento de outros experimentos. Para isso empregamos o circuito mostrado no diagrama esquemático da figura 4. Neste circuito empregamos dois microcontroladores, o primeiro (à direita) responsável pela codificação do sinal analógico original, usando um dos ADCs, e o segundo (à esquerda) pela decodificação do mesmo, utilizando um DAC R2R de somente 6 bits. A transmissão deste sinal é feita utilizando a UART incorporada no ATmega8, como ilustrado na figura 4.

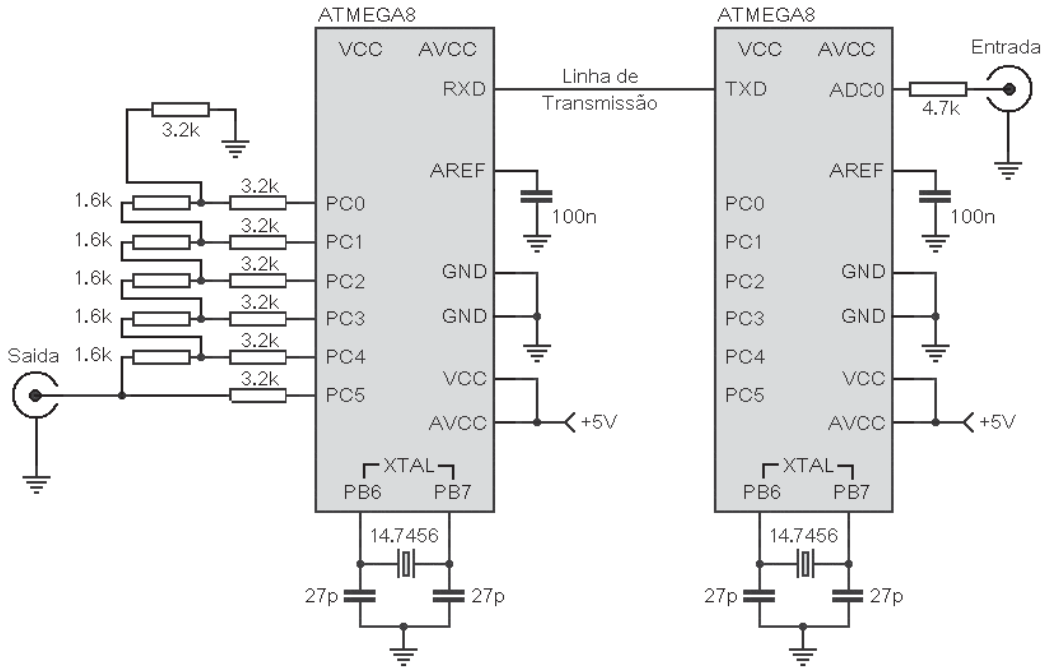

Figura 4 - Circuitos de codificação e decodificação de sinais. Resistores em Ω e capacitores em F.

Para o algoritmo de codificação usamos uma abordagem extremamente simples. Ela é baseada em duas tabelas de transcrição, uma direta empregada na codificação (array X) e outra inversa aplicada na decodificação (array Y). Os núcleos de processamento usados são listados na tabela 4, onde a atribuição do valor de cada elemento das tabelas são indicados resumidamente em vermelho. Assim o codificador faz uma amostragem do sinal analógico aplicado na entrada e envia pela UART o valor do byte contido na tabela direta. O decodificador recebe este byte via UART e executa a transcrição inversa, enviando o resultado para o DAC na porta C.

Tabela 4 - Programas referentes às rotinas de codificação e decodificação de um sinal.

Codificação	**Decodificação**

Dim X(256) As Byte
 	
'INICIO DA TABELA DIRETA
X(1) = 109
X(2) = 232
X(3) = ...
....
X(256) = 247
 	
Do
  Valor = Getadc(0)
  Valor = Valor + 1
  Printbin X(valor)
Loop
	Dim Y(256) As Byte
	
	'INICIO DA TABELA INVERSA
	Y(109) = 1
	Y(232) = 2
	Y(3) = ...
	.....
	Y(247) = 256
	
	Do
	  Inputbin Valor
	  Valor = Valor + 1
	  PortC = Y(valor)
	Loop

Os resultados desses procedimentos são ilustrados na figura 5. Na figura 5a temos uma comparação entre o sinal injetado na entrada do ADC do codificador e o sinal decodificado tomado na saída do DAC do decodificador. Como podemos ver, não há perdas significativas, sendo o sinal devidamente recuperado. Contudo, o sinal decodificado obviamente incorpora as limitações do processo de digitalização comentadas anteriormente. Na figura 5b ilustramos o caso onde o processo de decodificação é propositalmente suprimido, ou seja, o programa no decodificador simplesmente move o valor recebido pela UART diretamente para a porta C e, portanto para o DAC. Assim, é possível observar que o sinal original fica realmente transcrito, ou seja, codificado de uma maneira que ele se torna ininteligível.

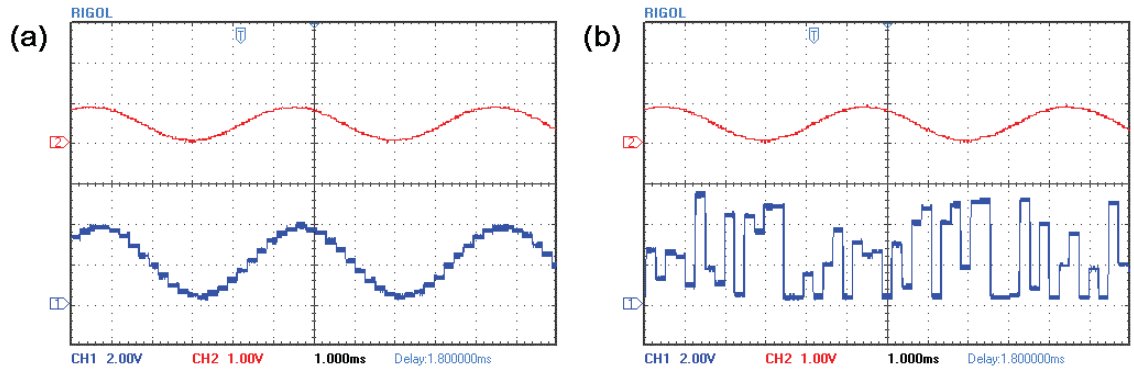

Figura 5 - Resultado obtido para o codificador: (a) Sinal senoidal com frequência de 200 Hz original (vermelho) e sinal decodificado (azul); (b) Sinal original (vermelho) e sinal transmitido (azul) pela linha.

5. CONCLUSÕES

Em resumo, apresentamos um conjunto de experimentos didáticos relacionados com o processamento digital de sinais usando microcontroladores. Nestes experimentos implementamos filtros digitais, retificadores, ceifadores, dobradores de frequência, codificadores e decodificadores. Tanto o hardware quanto o firmware foram concebidos para operação em tempo real. Apesar das limitações na frequência de amostragem, os experimentos explorados ilustram muito bem os diversos aspectos sobre o tema, podendo ser facilmente aplicados em atividades práticas nas disciplinas da área de processamento de sinais.

REFERÊNCIAS

ATMEL. **8-bit AVR with 8K Bytes In-System Programmable Flash – ATMEGA8 / ATMEGA8L** Disponível em: <http://www.atmel.com>. Acesso em 10 set 2011.

LATHI, B.P. **Sinais e Sistemas Lineares**. Editora Bookman, 2007.

MCS ELECTRONICS. **Bascom AVR Compiler**. Disponível em: <http://www.mcselec.com>. Acesso em 10 set 2011.