

Desenvolvimento de um Filtro Passa-Baixa Digital

Cassius Rossi de Aguiar*, Gabriel Alexandre de Souza Braga†, Gianluca Hiss Garbim‡,
Guilherme Gabriel de Oliveira§ e Marcus Vinícius Silvério¶

*Coordenação de Engenharia de Computação

Universidade Tecnológica Federal do Paraná, Toledo, Paraná 85902-490

Email: *cassiusaguilar@utfpr.edu.br, †gabrielbraga@alunos.utfpr.edu.br, ‡gianluca@alunos.utfpr.edu.br,
§guioli@alunos.utfpr.edu.br, ¶marcussilverio@alunos.utfpr.edu.br

Resumo—Utilizando os conceitos aprendidos na matéria de Sensores e Atuadores, foi proposto aos alunos o desenvolvimento de um filtro passa-baixa digital através da plataforma de prototipagem Arduino Due. Neste contexto, o código do filtro foi desenvolvido em C/C++ e os testes realizados no Arduino. Desta maneira é possível ponderar a utilização de filtros digitais e também entender melhor projetá-los.

I. INTRODUÇÃO

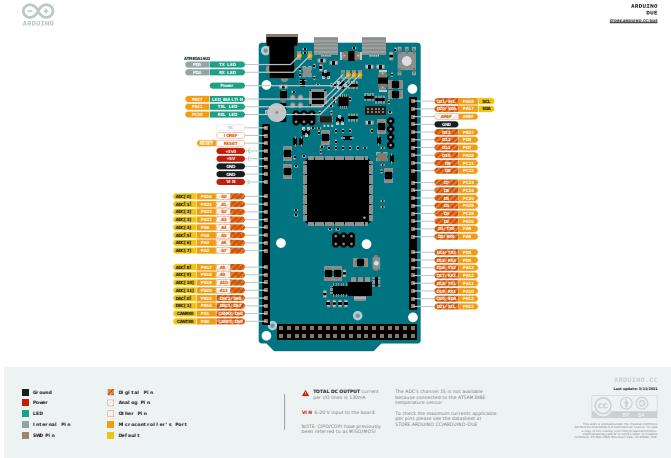
Filtros analógicos são amplamente utilizados em sistemas de alta precisão para diminuir o ruído presente nos sinais da estrutura. Com o aumento da qualidade e complexidade dos dispositivos existentes, a utilização de filtros se tornou cada vez mais frequente. Pensando nisso, a implementação de filtros digitais começou a ganhar destaque pois através destes há uma redução de custo, dado que caso haja uma mudança no circuito e alterações no filtro sejam necessárias, basta apenas trocar uma linha de comando, sem a necessidade de modificar componentes inteiros. Além da facilidade envolvida na manutenção, a mudança de um código por muitas vezes se apresenta mais simples do que a troca de um dispositivo eletrônico. Por fim, também há uma redução no tamanho e peso dos sistemas, uma vez que não é mais necessário ter o filtro fisicamente presente no sistema.

Desta maneira, foi desenvolvido neste trabalho acadêmico um filtro passa-baixa digital. A plataforma de prototipagem utilizada foi o Arduino Due, uma ótima opção para o desenvolvimento de baixo custo, pois apesar de ser um Arduino ainda apresenta boas especificações para o processamento digital de sinais, tendo 84 MHz de clock, 512 KB de memória flash, 2 DACs, 12 ADCs, pinos com resolução de 12 bits e E/S com suporte a tensão de 0 a 3,3 volts. A figura 1 demonstra os pinos presentes na plataforma de prototipagem utilizada para este projeto.

II. MÉTODO

A primeira etapa realizada foi o projeto para o filtro na frequência de corte $f_c = 300 \text{ Hz}$, no qual foi descrita a função de transferência do mesmo, posteriormente discretizada para possibilitar o trabalho com um sinal digital. Estes passos foram descritos no relatório "Projeto e Simulação de Filtro Passa-Baixa" previamente apresentado por esta equipe. Assim, foi possível projetar a magnitude da resistência $R = 1k\Omega$ e capacitância $C \approx 531nF$.

Figura 1. Mapa de pinos do Arduino DUE.



Fonte: store.arduino.cc (2022).

A equação 1 representa a equação em diferenças calculada no primeiro relatório da equipe.

$$v_o[k] = 0.086058519(v_s[k] + v_s[k-1]) \\ + 0.8278829604v_o[k-1] \quad (1)$$

Em posse da equação em diferenças, é possível programar o filtro em Arduino — linguagem muito semelhante às linguagens de programação C/C++. No código II é descrita somente a função PWM_Handler, por ser a mais relevante para a explicação da proposta.

Como é notável no código II, entre as linhas 8 e 14, está sendo efetuada a emulação do filtro-RC, fazendo assim o processamento digital dos sinais amostrados em bancada. É importante ressaltar que para a efetuação desse trecho de código, primeiro faz-se a normalização da tensão em relação a resolução dos pinos, tendo em vista que a resolução é de 12 bits, isso nos dá uma resolução com alcance de 0 a 4095 níveis, além de que o Arduino não consegue entregar de fato os 0 a 3,3 volts dito no *data sheet*, na prática suas tensões tem um alcance de 0,6 a 2,8 volts. Dessa maneira foi feito uma normalização, onde concluiu-se que 1,65 volts são equivalentes a um nível de 3071. Com isso foi efetuado o *offset* do sinal, descendo-o para que sua média seja 0 volts,

por fim, o sinal é convertido para volt, de tal forma a termos um sinal de $\frac{1,65}{2}$ a $-\frac{1,65}{2}$ volt dentro do Arduino. Em seguida é feito o processamento do sinal, como visto no código II.

```

1 void PWM_Handler()
2 {
3     volatile long dummy = PWM_INTERFACE->...
4         PWM_ISR1; // clear interrupt flag
5     dummy = PWM_INTERFACE->PWM_ISR2; // clear ...
6         interrupt flag
7
8     digitalWrite(13,HIGH);
9
10    Vs[1] = Vs[0];
11    Vs[0] = ((analogRead(A0)*3.3)/4095.0) - ...
12        (1.65/2.0);
13
14    Vo[1] = Vo[0];
15    Vo[0] = 0.086058519*(Vs[0]+Vs[1]) + ...
16        0.8278829604*(Vo[1]);
17
18    analogWrite( DAC1, ((Vo[0]+(1.65/2.0))...
19        *3071.0)/1.65 );
20
21    // Atualizacao dos PWMs, 1050 representa um...
22    // Duty de 50% para freq de 10kHz
23    PWMC_SetDutyCycle(PWM_INTERFACE, channel_1 ,...
24        42000*saida);
25    PWMC_SetDutyCycle(PWM_INTERFACE, channel_2 ,...
26        42000*saida1);
27    PWMC_SetDutyCycle(PWM_INTERFACE, channel_3 ,...
28        42000*saida2);
29    PWMC_SetDutyCycle(PWM_INTERFACE, channel_4 ,...
30        20);
31    digitalWrite(13,LOW);
32
33    PWM_INTERFACE->PWM_IDR1 = 0x10; //enable ...
34        interrupt on channel 4 - 00010000
35 }
```

Listing 1. Código introduzido no Arduino.

É importante lembrar, que essa conversão para volts, é feito por meio de uma simples regra de três, e o que teremos de fato no programa é uma variável *double* que representa a tensão que se teria num filtro RC de fato. Além disso, vale ressaltar que na função *void setup()* é configurado a resolução dos pinos de ADC e DAC, por padrão o Arduino DUE tem uma resolução de 10 bits, então é usado as funções *analogWriteResolution()* e *analogReadResolution()* para configurar a resolução para 12 bits.

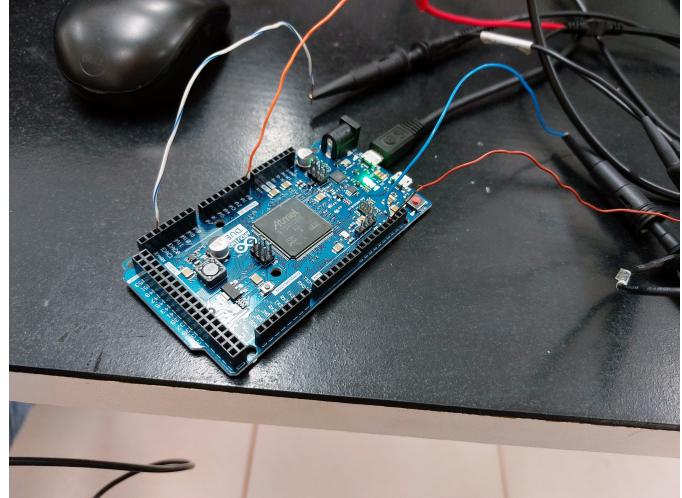
Na figura 2 é possível ver a bancada de teste, com foco na plataforma de prototipagem utilizada.

Após a montagem do circuito na plataforma, foi configurado o gerador de sinais: este ficou alternando entre 60 Hz e 600 Hz, sob uma tensão de 0 a 1,65 volts, para possibilitar o teste do filtro. Na figura 3 é demonstrado o gerador de sinais utilizado.

III. RESULTADOS

Com base nisso foram então realizados os testes. Primeiro seria necessário testar se as interrupções do Arduino, as quais apresentam uma frequência de $f_i = 10 \text{ kHz}$, estavam funcionando através da utilização de um osciloscópio para medir a saída 13 do Arduino. É importante ressaltar que a amostragem do sinal é feita por meio dessas interrupções,

Figura 2. Foto do Arduino Due na bancada de testes.



Fonte: Autoria própria (2022).

Figura 3. Foto do gerador de sinais.



Fonte: Autoria própria (2022).

sendo $T_s = \frac{1}{f_i}$. Assim, na figura 4, é elucidada a medição do sinal de interrupção, em azul, e o sinal de saída do DAC, em amarelo — sendo este o sinal de saída do filtro digital.

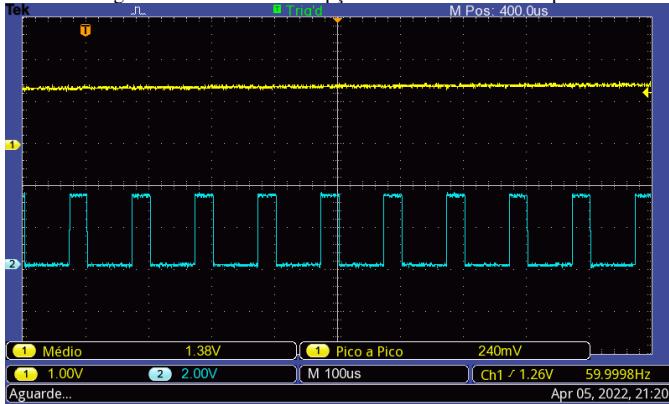
Em seguida é testado o sinal de saída do filtro digital. A figura 5 demonstra o sinal de saída do DAC para 60 Hz. Como o filtro é do tipo passa-baixa e sua frequência de corte é 300 Hz, o sinal de 60 Hz deve passar sem nenhuma atenuação.

A figura 6 demonstra a saída do filtro digital para 600 Hz — neste caso é esperado que o filtro atenuie o sinal de saída, pois 600 Hz é maior que a frequência de corte do filtro.

IV. CONCLUSÃO

Como é possível analisar na figura 5, existem pequenas perdas provindas da utilização do Arduino e do próprio filtro, apresentando uma tensão de saída de aproximadamente 1,14V. Para resolver isto seria necessária a utilização de equipamentos de instrumentação de melhor qualidade e o

Figura 4. Sinal de interrupção medido no osciloscópio.

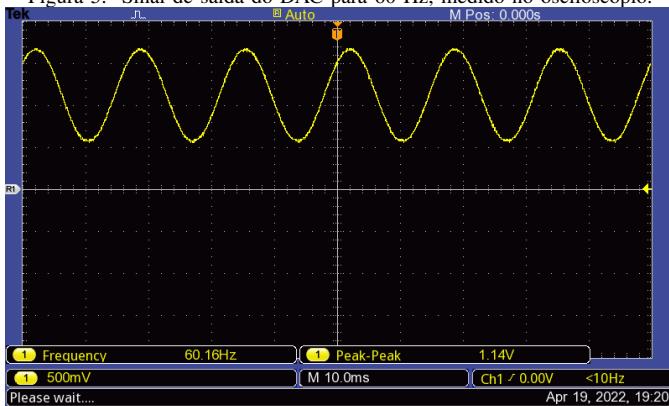


Fonte: Autoria própria (2022).

0 a 3.3 volts, assim também tendo um período de $100\mu s$, que é equivalente aos 10 kHz citado anteriormente, demonstrando assim o funcionamento das interrupções.

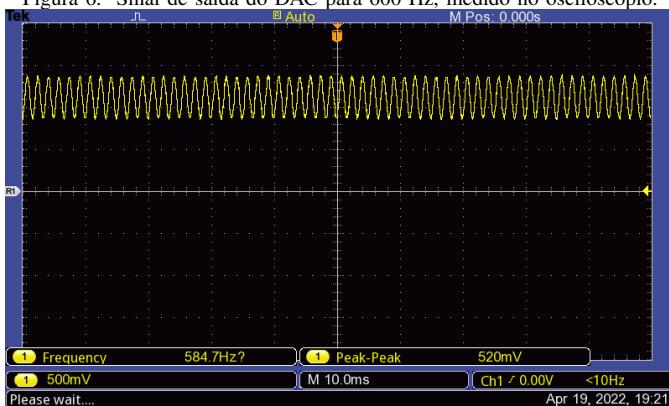
Dessa maneira foi possível ampliar os conhecimentos obtidos na matéria, colocando-os em prática através do projeto de um filtro passa-baixa digital funcional, efetuado de maneira a ter um baixo custo com utilização do Arduino Due, que no momento da publicação deste relatório é encontrado por menos de R\$300,00 no mercado.

Figura 5. Sinal de saída do DAC para 60 Hz, medido no osciloscópio.



Fonte: Autoria própria (2022).

Figura 6. Sinal de saída do DAC para 600 Hz, medido no osciloscópio.



Fonte: Autoria própria (2022).

projeto de um filtro mais robusto. Porém, isso tornaria a discretização mais complexa e também aumentaria o custo do projeto como um todo, tornando assim ele inviável para proposta de baixo custo.

Para a figura 6 temos um correto funcionamento, demonstrando uma atenuação da saída. Na figura 4 é notável o sinal em azul, que demonstra a interrupção, mostrando sua saída de