Processamento de sinal embarcado utilizando diferentes implementações

Kaique Guimarães Cerqueira*
Engenharia de Controle e Automação, Instituto Federal Fluminense.
(Dated: 2 de junho de 2022)

Este documento tem como objetivo detalhar a implementação de duas lógicas distintas: a aritmética de ponto fixo (*fixed-point arithmetic*), e a aritmética de ponto flutuante (*floating-point arithmetic*) no microcontrolador PIC18F1220. A escrita obedece a seguinte ordem: 1) contextualização geral do experimento e configuração do microcontrolador, 2) breve descrição dos critérios de avaliação e seus respectivos resultados, e 3) conclusão.

I. INTRODUÇÃO

O PIC18F1220 é um microcontrolador de 8-bits de baixo custo utilizado em diferentes aplicações. Seu microprocessador, junto com seus periféricos, foram feitos para trabalhar em ciclos intermitentes (código embarcado), onde é feita a aquisição, armazenagem, computação e comunicação de diferentes estímulos ao seu redor. Apesar da sua versatilidade, o mesmo não possui uma FPU (*floating point unit*), o que torna operações utilizando lógica de ponto flutuante uma tarefa fisicamente impossível (via hardware). Ao longo deste artigo, é comparado o desempenho do dispositivo utilizando a lógica de ponto fixo e o ponto flutuante "emulado"pelo compilador do software *PIC C*.

Para objeto de análise, o controlador PID foi implementado no código embarcado. O sistema calcula o sinal de erro através da leitura de dois sinais analógicos, sendo eles a referência e a realimentação simulada, que basicamente é a resposta ao degrau unitário de uma função de transferência de segunda ordem.

Para realizar a leitura da saída, foi utilizada a EPROM como buffer. Neste modelo de PIC[1], ela possui uma resolução de 8 bits, podendo armazenar até 256 amostras.

A decisão entre a melhor alternativa de implementação será definida ao analisar os critérios:

- Tempo de computação (Ts mínimo seguro)
- Precisão numérica (ao comparar com alguma ferramenta de computação científica)
- Uso de memória RAM
- Uso de memória Flash

A. Configuração dos conversor A/D

Como já dito anteriormente, utilizamos duas entradas analógicas no loop, logo é preciso configurar as mesmas antes de utilizá-las. O clock utilizado foi o interno (8MHz), que ele compartilha com o microprocessador.

```
setup_adc(  ADC_CLOCK_INTERNAL  );
setup_adc_ports( sAN0 | sAN1 );
```

B. Configuração do Interrupt

Em seguida, configura-se o interrupt do timer, para tornar possível rodar o código em intervalos definidos. Em um único comando, declara-se o uso do oscilador interno e o prescaler do clock, que é quantos ciclos de instrução deve-se passar até que se incremente o valor de 1 bit no registrador do timer.

```
setup_timer_0( T0_INTERNAL | T0_DIV_2 );
```

Como o PIC18F precisa de 4 ciclos de clock para completar uma instrução, acaba que o incremento do timer tem frequência de $\frac{1}{2} \times 2MHz$, Que com o prescaler de 1:2 fica $1MHz$.

A fim de definir o período de amostragem da simulação, é preciso ter total controle do tempo em que os comandos serão executados, e para isso nós inicializamos o registrador do timer com um valor diferente de zero. Ao invés de esperar o mesmo contar de 0-65535 (16-bits), compensa-se adicionando um *bias* no sistema. Para a interrupção levantar a cada 1 ms, pode-se utilizar a seguinte fórmula:

$$REG = 65535 - \frac{0.001}{((\frac{4}{F_{OSC}}) * PRESCALER))}$$

```
set_timer0(64535);
```

Com os periféricos devidamente configurados, é possível rodar a simulação.

C. Parâmetros do PID

Para facilitar os cálculos e manter a relação de proporcionalidade entre as constantes, definimos os parâmetros Kp, Ki e Kd como múltiplos e submúltiplos de 2. O período de amostragem utilizado nos cálculos é um valor simbólico, e não reflete as as respostas encontradas no tempo de computação.

*Electronic address: kaique.cerqueira@gsuite.iff.edu.br

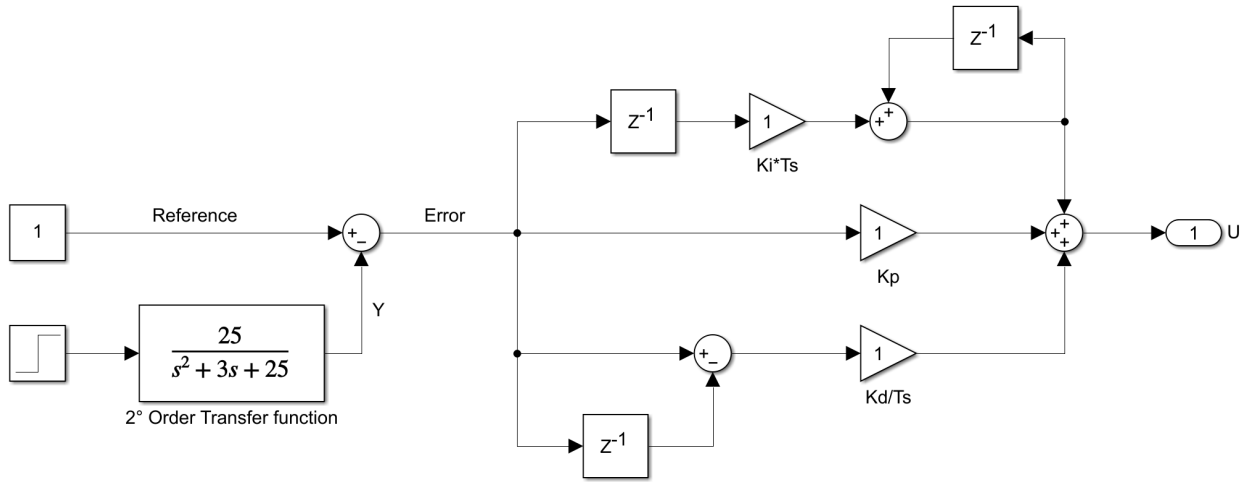

Figura 1: Diagrama de blocos dos sinais simulados no proteus

Constante	Valor escalar	Bit-shift
Kp	2	<<1
Ki	0,25	>>2
Kd	0,125	>>3
Ts	0,03125 s	>>5

II. CRITÉRIOS DE AVALIAÇÃO

Com os periféricos devidamente configurados, é possível passar para o estágio de simulação, utilizando o *Proteus 8* como ferramenta. As entradas foram simuladas ($1V$ na constante de referência e step unitário na realimentação simulada) e o bloco de osciloscópio foi utilizado para visualizar o tempo de computação.

A. Tempo de Computação

O *Profiling* foi realizado observando uma saída digital, que foi previamente configurada para o nível lógico alto enquanto a computação do PID estivesse sendo realizada, e baixo para os estados seguintes. As duas estratégias de computação foram então simuladas em paralelo e analisadas lado a lado, usando diferentes canais do osciloscópio.

As duas computações deram valores na casa dos microssegundos, sendo que a implementação em ponto fixo deu um valor com uma ordem de grandeza 10 vezes menor.

Implementação	Tempo de Computação
Ponto Fixo	62,5 us
Ponto Flutuante	588 us

Nas implementações mais usuais, o microprocessador executa outras tarefas como a aquisição de dados e atuação em motores e válvulas. Para isso, pode-se generalizar e acrescentar mais 200 us no período de amostragem.

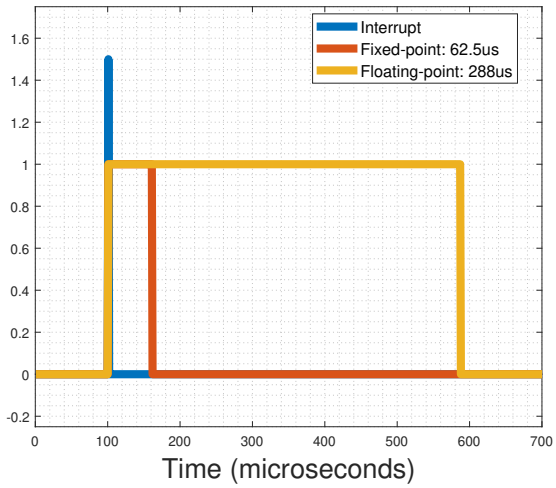

Figura 2: Tempo de computação analisado para ambas as implementações

B. Precisão numérica

A fim de realizar análises, compara-se a resposta da implementação nos dois métodos com uma ferramenta de computação científica, neste caso o *MATLAB*. A resposta foi escalada como um sinal de 0-5 V.

Para analisar grandezas vetoriais, diversos parâmetros podem ser utilizados. O sinal computado pelo MATLAB é tido como Y_i e os valores dados pelas implementações $\hat{Y}_i$. Neste artigo, utilizaremos 3 destas, sendo elas:

1. **Erro quadrático médio**
   O MSE (da sigla em inglês Mean Squared Error), é

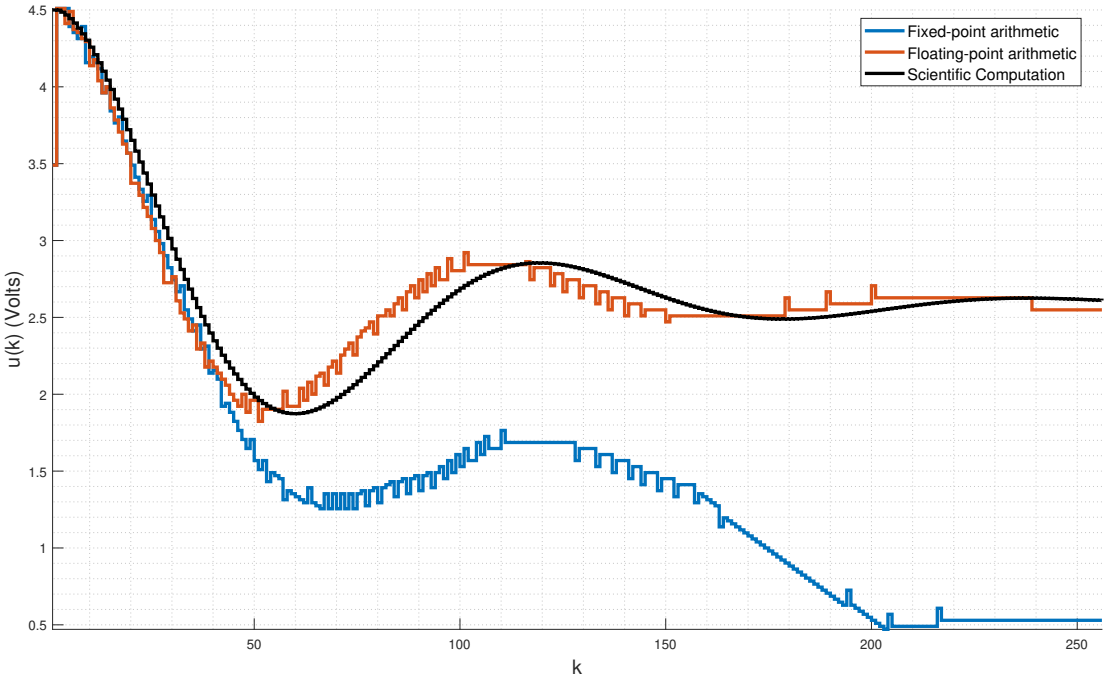

Figura 3: Resposta da ação de controle nos diferentes métodos

comumente usado para quantificar, utilizando um valor escalar, a qualidade de modelos. Este método penaliza mais os maiores erros, já que, ao ser calculado, cada erro é elevado ao quadrado individualmente e, após isso, a média desses erros quadráticos é calculada. Nesta métrica, quanto menor, melhor.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Mean Squared Error	
Ponto Fixo	1,8336
Ponto Flutuante	0,0221

2. **Erro máximo**
   O erro máximo é útil de se ver em ambiente controlado, como em simulações ou lugares isolados de interferência eletromagnética, por exemplo, e diz se os valores do cálculo de implementação estão dentro de uma faixa de tolerância aceitável. Assim como na métrica anterior, quanto menor, melhor.

$$M = max(|(Y_i - \hat{Y}_i)|)$$

Maximum Error	
Fixed-point	2,1027
Floating-point	1,0098

3. **Média Percentual Absoluta do Erro**
   Na MAPE (da sigla em inglês Mean Absolute Percentage Error)), é possível fazer comparações entre erros percentuais das implementações, calculando o quanto está longe do modelo da computação científica, em média.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{Y}_i)}{Y_i} \cdot 100\%$$

Mean Absolute Percentage Error	
Fixed-point	15,97 %
Floating-point	0,27 %

C. Uso da RAM

Neste quesito, analisamos os valores estimados pelo compilador. Nesta implementação, temos como resposta:

RAM Usage	
Fixed-point	19 %
Floating-point	37 %

D. Uso da memória Flash

O método utilizado é o mesmo da métrica anterior:

ROM Usage	
Fixed-point	15 %
Floating-point	58 %

III. CONCLUSÕES

Neste artigo investiga-se a implementação do PID em dois modos distintos, onde cada um tem suas vantagens e desvantagens. Alguns detalhes são dignos de nota, como o aparente ruído de quantização do sinal, e o fato da implementação utilizando ponto fixo destoar da resposta ideal: isso muito provavelmente se dá por conta do fator integrativo ser ínfimo, a ponto de quase não conseguir ultrapassar o limiar do sinal discretizado a fim de representar uma mudança de bit em sua resposta.

Apesar de requerer apenas um décimo do seu antagonista para sua computação, o parâmetro integrativo dos cálculos em ponto fixo se tornou praticamente inexistente, devido a quantidade de bit-shifting utilizado em seu cálculo, diminuindo significativamente a sua resolução. Se a planta representar um sistema que requer uma resposta muito rápida, talvez essa seja a implementação ideal. Os cálculos em ponto flutuante se mostraram bem mais precisos, e isso se reflete nos cálculos das métricas de sua precisão numérica. Se o sistema tiver uma resposta lenta, e o microcontrolador for utilizado apenas para o controle do mesmo (sem RTOS, por exemplo), essa estratégia de implementação se sairá mais recomendável.

Mais detalhes da implementação podem ser vistas no repositório do projeto no Github [2]

IV. REFERÊNCIAS

[1] Microchip.com. 2022. [online] Available at: https://www.microchip.com/wwwproducts/en/PIC18F1220 [Accessed 25 May 2022].

[2] PID Micro Repository. 2022. [online] Available at: https://github.com/kkkiq/pid_micro [Accessed 31 May 2022]

Apêndice A: Recursos computacionais

Todas as simulações e análises descritas neste artigo foram feitas utilizando os recursos dos softwares *PIC C Compiler*, da *CCS Inc.*; o *Proteus 8 Professional*, da *Labcenter Electronics Ltd.*; e o *MATLAB*, da *MathWorks Inc.*; O chip PIC18F1220, utilizado como referência nas simulações, é desenvolvido pela *Microchip Technology*.
Todos os direitos reservados.