Relatório Final – F 590A – Iniciação Científica I

Estabilização em frequência de laser de corante com LabVIEW

Coordenador: José Joaquim Lunazzi

Orientador: Luís Eduardo Evangelista de Araujo

Aluno: Felipe Gustavo da Silva Santos – RA 076237

Junho de 2010

1. Resumo

Os procedimentos, principais problemas e resultados envolvidos no desenvolvimento do projeto de estabilização de um laser de corante com LabVIEW são discutidos. Nesta primeira etapa, o principal objetivo era estabelecer a comunicação entre o LabVIEW e um conversor analógico digital, e implementar um sistema de PID também através do LabVIEW.

2. Pesquisa realizada e resultados atingidos

A porta paralela (pinagem na figura 1) nada mais é que um canal de comunicação entre um computador e um dispositivo externo. Possui essencialmente três registradores, sendo um deles usualmente dedicada a receber dados (registrador *status*), um capaz de enviar e receber dados (*data*) e outro que pode apenas enviar ou enviar e receber dados (*control*) dependendo do estado do *bit* 5 deste *byte*. Para os objetivos desse projeto, a principal característica explorada da porta é sua capacidade de enviar e receber múltiplos *bits* simultaneamente.

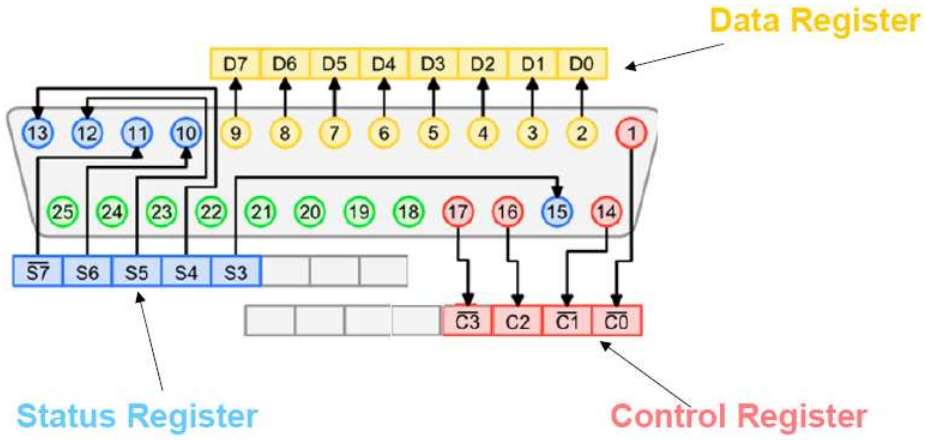


Figura 1: Pinagem da porta paralela.

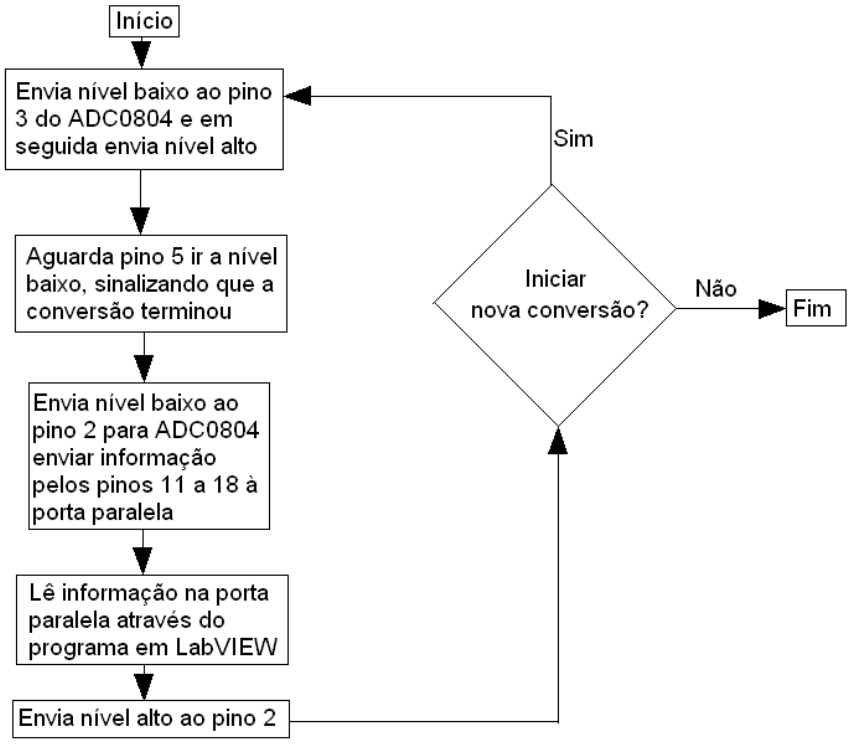


Figura 2: Algoritmo do programa para comunicação do ADC0804 com computador via porta paralela.

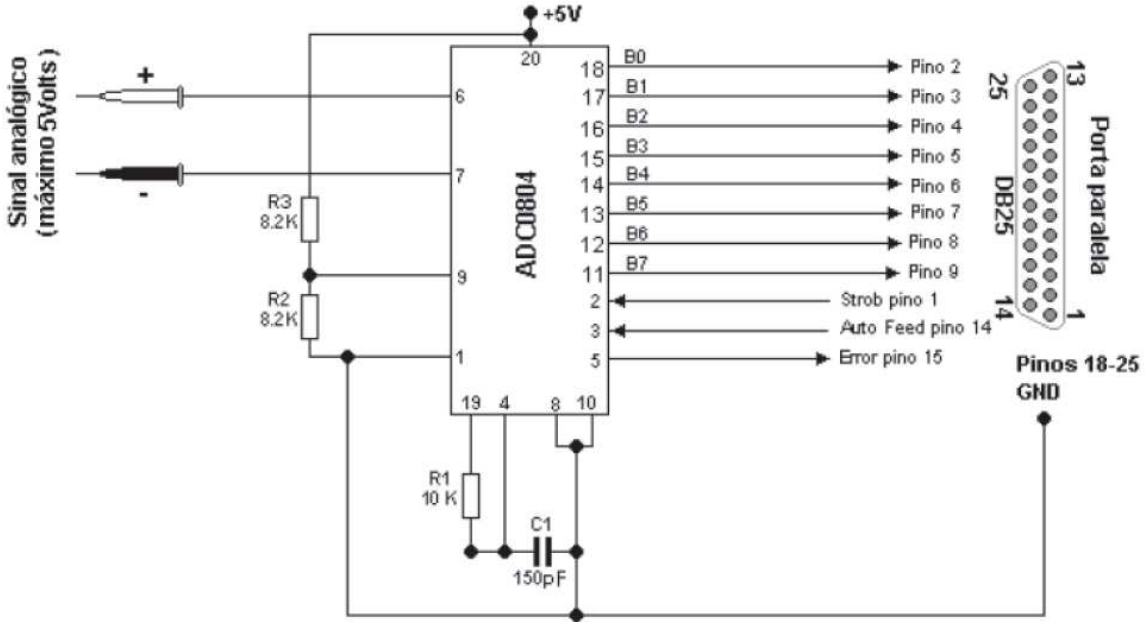


Figura 3: Diagrama do circuito implementado para comunicação da porta paralela com o ADC0804 via LabVIEW.

2.1 Sistema de comunicação entre ADC0804 e computador

Para realizar a comunicação da porta com o A/D, a principal referência foi [2], um trabalho no qual se utiliza um ADC0804 para enviar informação ao computador através da porta paralela. Em função de diferenças entre o algoritmo dessa referência e as informações do manual do ADC0804 [3], foi dada preferência à orientação do fabricante, isto é, utilizamos o algoritmo (figura 2) indicado pelo manual. O hardware utilizado inicialmente foi o mesmo de [2] (figura 3), pois não havia contradições com o manual.

Ao fim da implementação, introduzimos diferentes formas de onda na entrada analógica do A/D e observamos o resultado da conversão do sinal analógico para sinal digital pelo sistema A/D+LabVIEW.

Observamos, para todas as frequências utilizadas, ruídos intensos no sinal convertido (figuras 4 a 7). Esse ruído não pode ser gerado pelo programa, ou seja, vem do circuito. De fato, descobrimos que uma provável fonte de ruído é a forma como o aterramento foi feito. No circuito proposto em [2], os terras da parte analógica (pino 8) e da digital (pino 10) estão conectados um a outro logo na saída do A/D. Segundo a referência [7], o certo seria separar duas malhas, uma dedicada aos pinos da parte digital do A/D e outra dedicada aos pinos da parte analógica, sendo que essas duas malhas devem ser conectadas uma à outra através de seus aterramentos apenas próximo da fonte de tensão. Inicialmente, não se buscou corrigir esse problema, pois a intenção inicial era apenas testar o ADC0804 para reproduzir resultados de [2] e, então, trabalhar com outro A/D, o MCP3204. Em função de dificuldades com este CI não previstas inicialmente, decidimos corrigir o circuito do ADC0804 e prosseguir trabalhando com ele.

Observando a pinagem do ADC0804 [3], rearranjamos o circuito como na figura 4. Como resultado, o ruído após a conversão foi completamente anulado (figura 4). Ao introduzir sinais analógicos na entrada, o ruído observado (figuras 5 a 7) era completamente compatível com o ruído medido diretamente na fonte do sinal por outros meios.

Quanto à resposta em frequência do conversor, vimos que o sistema operava bem até aproximadamente 600 Hz no sinal de entrada. O ADC0804, bem como o circuito em que estava montado, deveria ser capaz de converter sinais de até 10 kHz. Portanto, conclui-se que o próprio software é o responsável pela limitação na frequência de operação do sistema. Esta conclusão é confirmada pelo fato do LabVIEW realizar suas operações em tempos da ordem de milissegundos quando executado através do sistema operacional utilizado, o Windows [4].

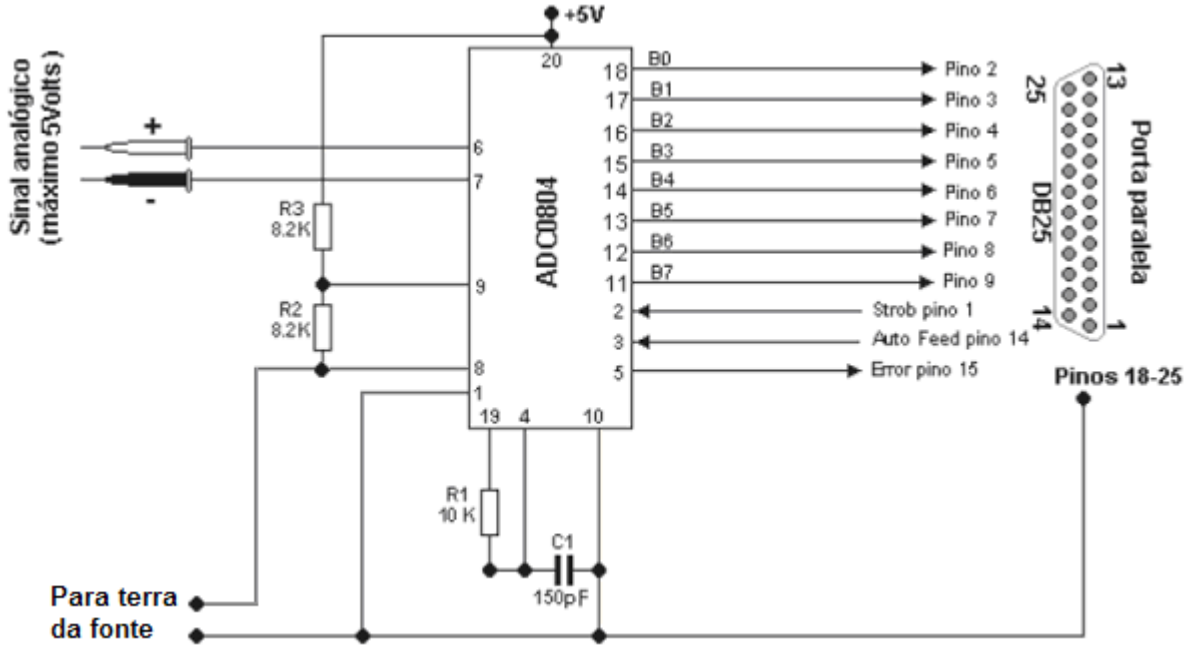


Figura 4: Diagrama do circuito com aterramento corrigido para redução de ruídos implementado para comunicação da porta paralela com o ADC0804 via LabVIEW.

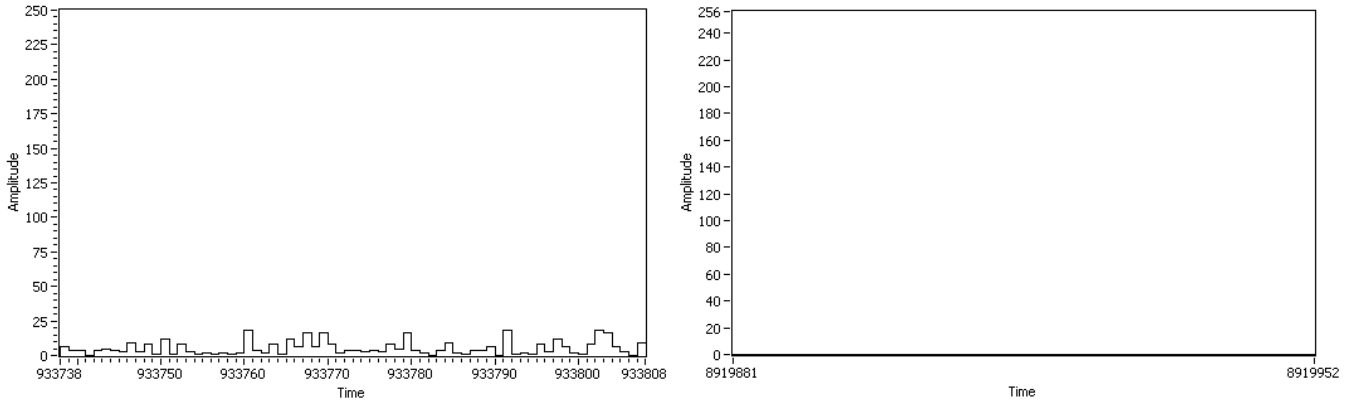


Figura 5: Amplitude (em unidades arbitrárias) do sinal convertido em função do tempo (em unidades arbitrárias) para sistema com ADC0804 sem sinal na entrada do conversor antes (esquerda) e depois (direita) da correção no aterramento.

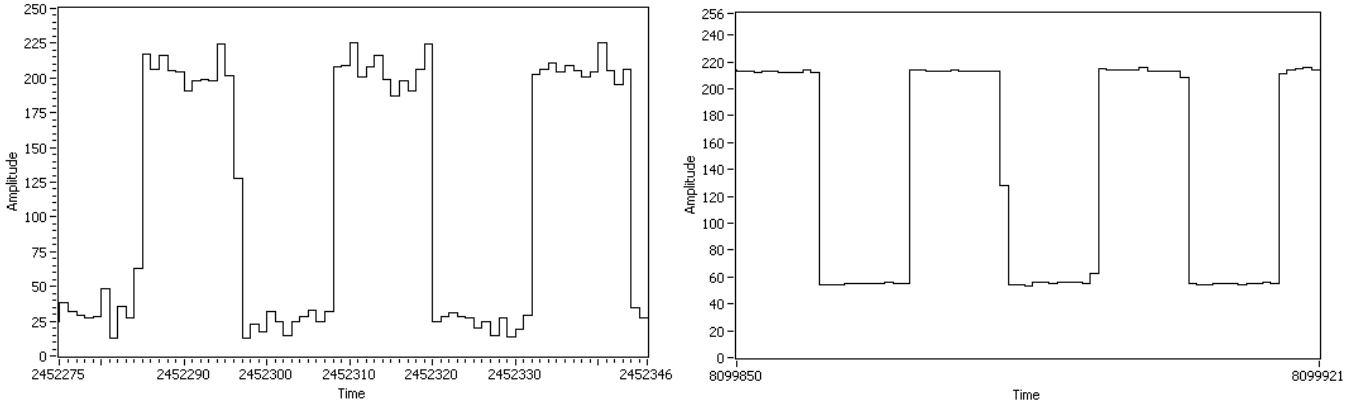


Figura 6: Amplitude (em unidades arbitrárias) do sinal convertido em função do tempo (em unidades arbitrárias) para sistema com ADC0804 para onda quadrada na entrada do conversor de 210 Hz antes (esquerda) e depois (direita) da correção do aterramento.

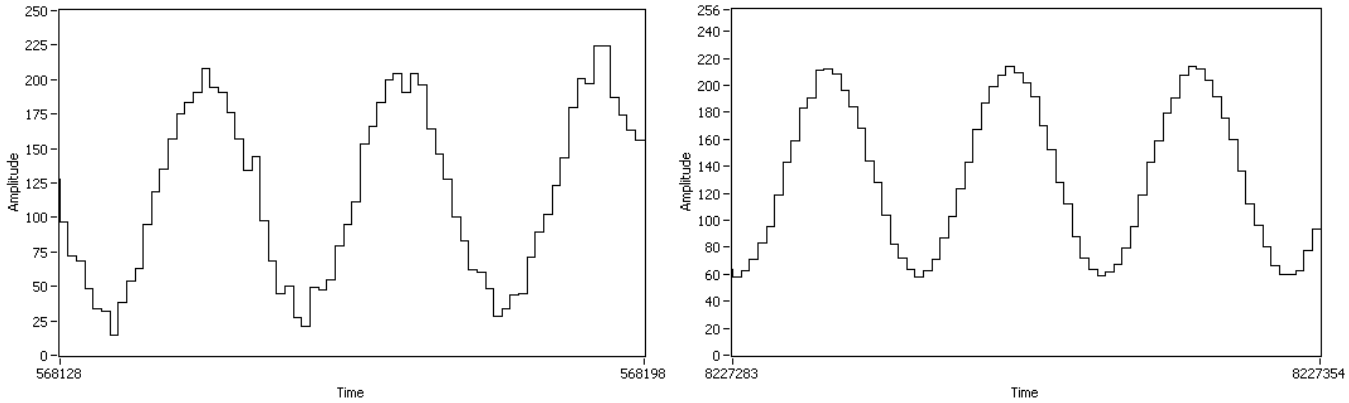


Figura 7: Amplitude (em unidades arbitrárias) do sinal convertido em função do tempo (em unidades arbitrárias) para sistema com ADC0804 para onda senoidal na entrada do conversor de 210 Hz antes (esquerda) e depois (direita) da correção do aterramento.

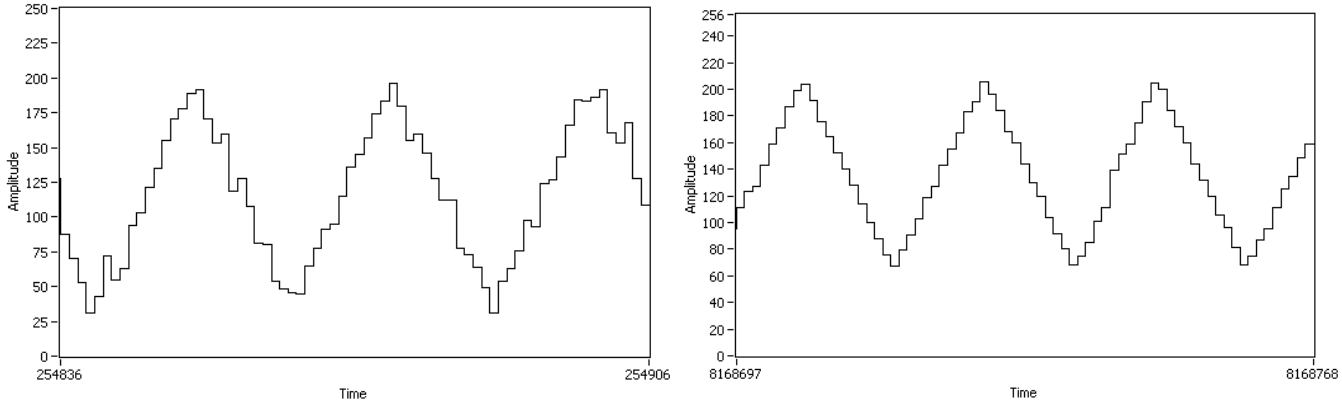


Figura 8: Amplitude (em unidades arbitrárias) do sinal convertido em função do tempo (em unidades arbitrárias) para sistema com ADC0804 para onda triangular na entrada do conversor de 210 Hz antes (esquerda) e depois (direita) da correção do aterramento.

2.2 Sistema de comunicação entre computador e MCP3204

Em substituição ao ADC0804, seria utilizado o MCP3204. A principal motivação dessa troca é a resolução do A/D: o ADC0804 converte sinais para apenas 8 *bits*, enquanto o MCP3204 converte para 12. Contudo, esse aumento de resolução tem um preço: enquanto o ADC0804 comunica os 8 *bits* de saída paralelamente, isto é, através de pinos diferentes, o MCP3204 utiliza comunicação serial, o mesmo pino é encarregado de enviar os 12 *bits* do sinal convertido. Os *bits* são enviados a intervalos regulares de tempo, determinados por um sinal de *clock* (marcador de tempo, geralmente uma onda quadrada com mínimo em 0 V e máximo em 5 V). Para que esses 12 *bits* sejam identificados corretamente, deve haver sincronia entre o sistema que os decodifica e os intervalos de tempo entre o envio de cada *bit*, ou seja, o funcionamento do decodificador de *bits* deve ser subordinado ao *clock*.

Além de identificar corretamente os *bits* convertidos, dados de controle devem ser enviados ao MCP3204 de forma a iniciar conversões, procedimento esse também subordinado ao sinal de *clock*. Um diagrama de como os *bits* de entrada devem ser manipulados, como os *bits* de saída são gerados e como isso se relaciona ao *clock* está na figura 9, retirada do manual do MCP3204 [5]. Nesse A/D, os estados dos *bits* mudam sempre na descida do *clock*, isto é, quando este passa do nível lógico alto para o baixo.

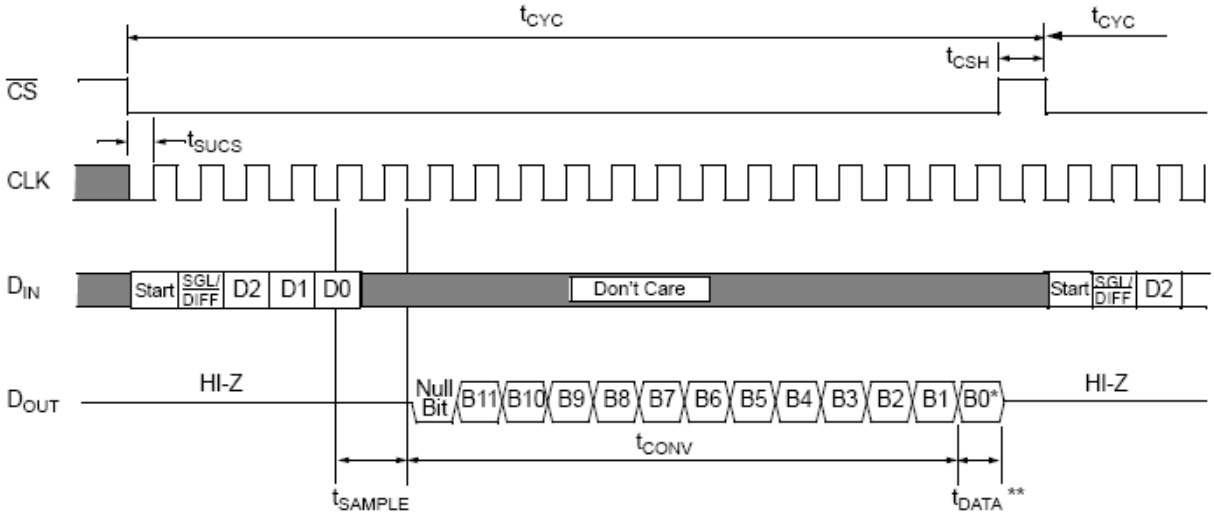


Figura 9: Comportamento dos *bits* de entrada (D_{in}, CS) e de saída (D_{out}) em função do *clock* (CLK). Os *bits* "start", "sgl/diff", "D2", "D1" e "D0" são escolhidos dependendo da funcionalidade que se deseja dar ao MCP3204 (veja [5]). Aqui, escolhemos 1, 1, 1, 0 e 0, respectivamente.

Para gerar o sinal de *clock*, utilizou-se um CI bastante conhecido, o *timer* 555 [6], que tem duas configurações básicas. Numa delas, o multivibrador astável (figura 10), o sinal de saída oscila entre zero (nível lógico zero ou baixo) e um valor de referência que depende da tensão de alimentação V_{CC} (nível um ou alto). As durações dos níveis (alto e baixo) são determinadas pelos valores dos capacitores e resistores mostrados na figura 10. Esta configuração foi utilizada para gerar o sinal de *clock*.

Para controlar os *bits* de entrada do MCP3204 bem como para ler sua saída, seria utilizado o LabVIEW. Para haver a necessária sincronia entre o A/D utilizado e o programa, o funcionamento deste deveria ser subordinado ao *clock*, ou mais especificamente, informação deveria ser trocada com o A/D sempre na descida do sinal de *clock*. Para tanto, seria necessário detectar a descida do *clock*, o que mostrou ser a maior dificuldade na elaboração de todo o sistema com o MCP3204 até agora.

Para detectar a descida do *clock*, a primeira tentativa foi observar diretamente o nível lógico atual deste sinal através do próprio programa, isto é, uma instrução era enviada ao A/D, esperava-se que o *clock* estivesse em nível baixo, e então se dava sequência às instruções. Tal alternativa mostrou-se completamente inadequada, pois observar o nível atual do *clock* pode originar duas situações básicas de erro no sistema.

Se a frequência do *clock* for maior que a taxa temporal com que o programa executa suas instruções, enquanto determinada instrução é executada, pode ser que o *clock* desça mais de uma vez, ou seja, o A/D muda de estado mais de uma vez enquanto o programa muda apenas uma. Por outro lado, se a frequência do *clock* for menor que a taxa temporal de execução das instruções do programa, o contrário pode ocorrer, isto é, o programa executa uma instrução em um intervalo de tempo tal que o *clock* não tem tempo para voltar ao nível alto; logo, o programa interpreta que o *clock* desceu novamente, quando na verdade este sequer subiu. Se isso acontece, o programa muda de estado mais de uma vez enquanto o A/D muda apenas uma. Nos dois casos, a sincronia não existe.

Para solucionar esse problema, outro *timer* 555 foi utilizado, mas dessa vez na segunda das suas configurações básicas, o multivibrador monostável (figura 11). Nessa configuração, o CI emite um pulso (com largura determinada pelos componentes externos "*timing components*" na figura 11) sempre que o sinal em sua entrada passa por uma descida. Então, para evitar os problemas discutidos, a largura do pulso de saída foi feita inferior ao tempo gasto pelo *clock* em cada nível lógico.

Contudo, para o correto funcionamento do multivibrador monostável, o sinal em sua entrada não pode permanecer no nível lógico baixo por tempo superior à largura do pulso de saída,

exatamente o que se desejava fazer. Por isso, fez-se necessário introduzir um bloco entre a entrada do multivibrador monostável e a saída do *clock* (“*trigger network*”, na figura 11) responsável por transformar o sinal de *clock* como mostra a figura 12.

Dessa forma, o sinal de *clock* gerado pelo multivibrador astável seria acoplado tanto à entrada de *clock* do MCP3204 quanto à entrada do multivibrador monostável, e o sinal da saída do monostável seria encaminhado ao LabVIEW através da porta paralela para estabelecer a sincronia.

Apesar de estudar todos esses detalhes, não foi possível implementar esse sistema utilizando LabVIEW, novamente pela sua limitação de tempo de execução. O MCP3204 requer um *clock* mínimo de 10 kHz, equivalendo a um período máximo de 0,1 ms; o LabVIEW novamente não seria capaz de ser sincronizado a este *clock*. Entretanto, este problema pode ser solucionado utilizando uma linguagem de programação de nível mais baixo (o C, por exemplo) e, portanto, de execução mais rápida. Em compensação, o código do programa ficaria mais complexo e demandaria maior tempo para elaboração, sendo esse o motivo pelo qual se decidiu retornar a utilizar o ADC0804.

Também podemos resolver este problema utilizando soluções proporcionadas pelo fabricante do LabVIEW, nas quais uma máquina funcionaria dedicado apenas à execução do LabVIEW, tornando-o até 1000 vezes mais rápido [4].

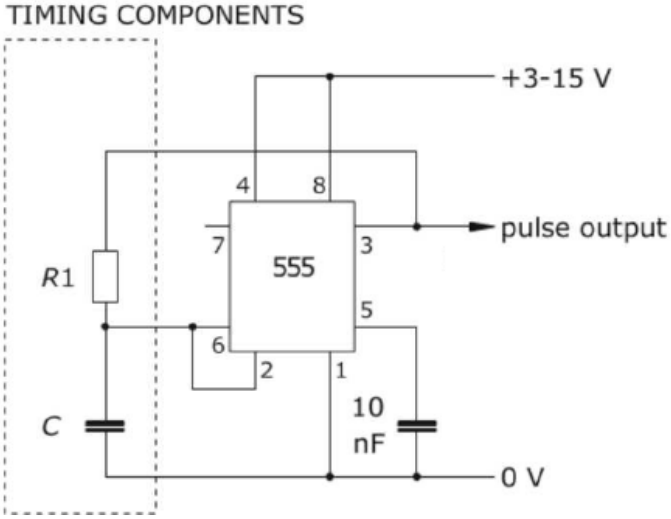


Figura 10: Diagrama do multivibrador astável utilizando *timer* 555

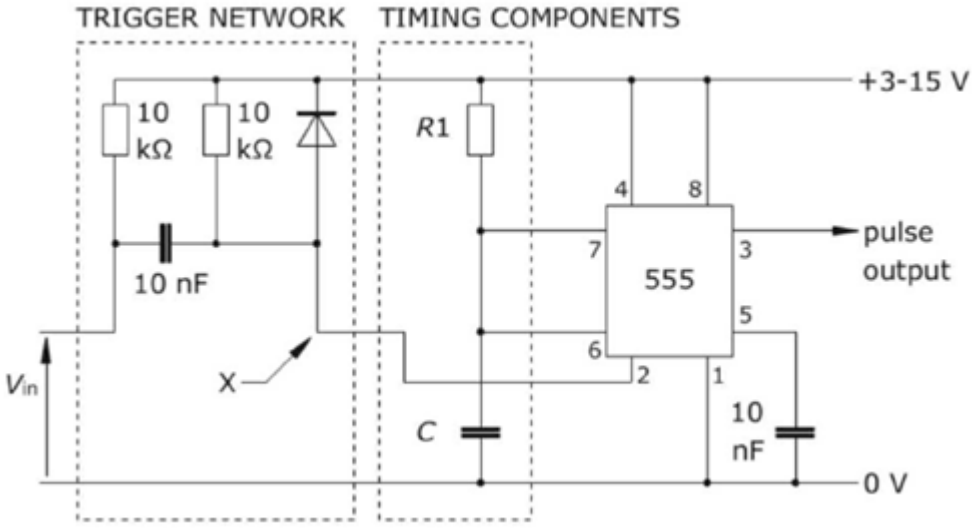


Figura 11: Diagrama do multivibrador monostável utilizando *timer* 555.

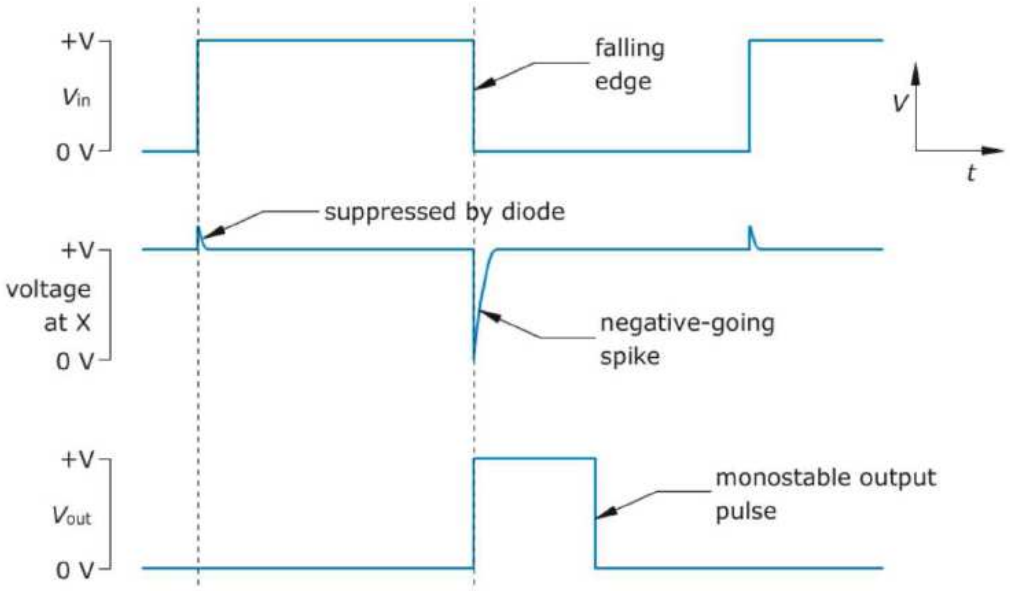


Figura 12: Efeito do "*trigger network*" no sinal de *clock* do multivibrador astável.

2.3 *Loop filter* PID

Um dos algoritmos mais usados para controle (com realimentação) de processos é o PID (Proporcional-Integral-Derivada) [8]. A função de qualquer sistema de controle com realimentação é medir um sinal de erro ε (adquirido a partir da diferença entre a saída atual e o valor desejado na saída) e, por meio de um algoritmo, gerar um sinal (sinal de atuação) capaz de levar o estado das saídas próximo de um valor de referência. Como todo sistema físico tem flutuações, o sistema de controle é importante para minimizar o efeito dessas flutuações sobre as grandezas físicas importantes para um experimento ou uma aplicação.

No PID, como o nome sugere, o sinal de atuação (A) é composto por três componentes, proporcional (P), integral (I) e derivada (D). Matematicamente, temos:

$$A = \underbrace{K_P \varepsilon}_{P} + \underbrace{K_I \int_{t_0}^{t} \varepsilon dt}_{I} + \underbrace{K_D \frac{d\varepsilon}{dt}}_{D},$$

onde K_P, K_I e K_D são os ganhos correspondentes a cada correção.

Num sistema real, enquanto P pode ser adquirida instantaneamente, as componentes I e D requerem um tempo de aquisição de dados, pois se tratam de operações que dependem do comportamento do sinal de erro num instante de tempo e na vizinhança desse instante.

Provavelmente o procedimento mais óbvio para o controle do sistema é a utilização da componente P com $K_P = 1$. Contudo, em função da aleatoriedade com que as flutuações podem ocorrer, outras medidas são necessárias para manter a estabilidade desse controle.

Se, por exemplo, a saída a ser controlada começa a crescer muito rapidamente, P pode não ser capaz de manter a saída no valor de referência. É então que emerge a importância de D: flutuações muito intensas possuem derivadas elevadas, e então são corrigidas a partir dessa derivada.

Tipicamente, as componentes P e D não são capazes de levar a saída exatamente ao valor de referência, originando um pequeno e constante sinal de erro. Daí a importância da integração: a componente I atua acumulando o erro ao longo de certo período de aquisição, para então corrigir essa diferença. Assim, o pequeno e constante sinal de erro fica ainda menor.

Para a implementação do PID no nosso sistema, seria necessário diferenciar e integrar o sinal discreto gerado pelo A/D, pois desejamos enviar um sinal (elétrico) de erro relacionado à frequência central de um laser de corante para ser convertido e então gerar o sinal de atuação pelo computador; o sinal de atuação estabilizará a frequência central do laser. Para tanto, o LabVIEW

possui blocos que fazem essas duas operações através de aproximações. A aproximação mais grosseira que podemos utilizar para derivação é:

$$\left(\frac{d\varepsilon}{dt}\right)_i \approx \frac{\varepsilon_i - \varepsilon_{i-1}}{t_i - t_{i-1}},$$

onde o índice i corresponde ao i-ésimo ponto (ε,t) adquirido (tais pontos são adquiridos a intervalos regulares de tempo; no nosso sistema, utilizaremos um múltiplo do tempo de conversão do A/D para tais intervalos). Para a integração:

$$\int_{t_{i-1}}^{t_i} \varepsilon dt \approx \varepsilon_{i-1}(t_i - t_{i-1})$$

Algumas aproximações mais sofisticadas podem ser usadas (e possuem blocos correspondentes no LabVIEW), mas inicialmente daremos preferência a essas.

3. Conclusão

Modificamos a ideia inicial de se utilizar um A/D de 12 *bits* em função da dificuldade em sincronizar este com o computador, dando preferência a um A/D de 8 *bits*, inicialmente utilizado para testes de comunicação entre o LabVIEW e um circuito através da porta paralela. O próximo passo relacionado a essa parte é melhorar a resposta em frequência do sistema de conversão analógico-digital para garantir que o sinal analógico (futuramente um sinal de erro relacionado à frequência central de um laser de corante) seja convertido com a maior fidelidade possível para melhorar a eficiência do PID.

Por ser muito mais usado que a porta paralela, referências sobre o PID podem ser facilmente encontradas entre fabricantes de componenetes eletrônicos. Já temos um algoritmo para implementar esse sistema, e a finalização do controle deve ocorrer em breve.

Finalizado o PID e melhorada a resposta em frequência do A/D, passaremos à segunda etapa do projeto, a qual prevê a utilização de espectroscopia de absorção saturada para localizar a transição da molécula de iodo na qual a frequência do laser será travada.

4. Referências

[1] Manual de introdução do LabVIEW da National Instruments, “LabVIEW – Getting Started with LabVIEW”, edição de abril de 2003.

[2] Gaião, E. N., “Uma interface lab-made para aquisição de sinais analógicos instrumentais via porta paralela do microcomputador”, Química Nova, Vol. 27, No. 5, 2004

[3] National Semiconductor Corporation, “ADC0801 / ADC0802 / ADC0803 / ADC0804 / ADC0805 – 8-bit µP Compatible A/D Converters” (manual), dezembro de 1994.

[4] Johnson, G. W., Jennings, R., “LabVIEW Graphical Programming”, 4ª edição, 2006, capítulo 5, item “Where Do Little Timers Come From?”.

[5] Microchip Technology Inc., “MCP3204/3208 – 2.7 V 4-Channel/8-Channel 12-Bit A/D Converters with SPI Serial Interface” (manual), 2008.

[6] ST Microelectronics, “NE555 – SA555 – SE555 – General purpose single bipolar timers”.

[7] AN018 (Nota de Aplicação), Intersil Corporation, “Do’s and Don’ts of Applying AD Converters”, 1999.

[8] National Instruments Corporation, “PID Theory Explained”, versão 3, 2006 (Anexo 1).

5. Parecer do orientador

"O aluno trabalhou na montagem de um circuito de conversão analógico/digital. Esse circuito será utilizado na aquisição de dados de um experimento a ser realizado posteriormente por ele para estabilização em freqüência de um laser de corante. Além da montagem do circuito, ele também trabalhou na programação em LabView para estabelecer uma interface do micro com o circuito. O Felipe está trabalhando com bastante dedicação a este projeto. No decorrer deste, encontrou vários problemas associados à programação do circuito que não foram previstos inicialmente. O aluno tem demonstrado bastante iniciativa, propondo soluções para esses problemas. Considero que ele vem tendo um ótimo desempenho no desenvolvimento do projeto."

6. Anexo 1

PID Theory Explained

Overview

Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industry and has been universally accepted in industrial control. The popularity of PID controllers can be attributed partly to their robust performance in a wide range of operating conditions and partly to their functional simplicity, which allows engineers to operate them in a simple, straightforward manner.

As the name suggests, PID algorithm consists of three basic coefficients; proportional, integral and derivative which are varied to get optimal response. Closed loop systems, the theory of classical PID and the effects of tuning a closed loop control system are discussed in this paper. The PID toolset in LabVIEW and the ease of use of these VIs is also discussed.

Table of Contents

Control System

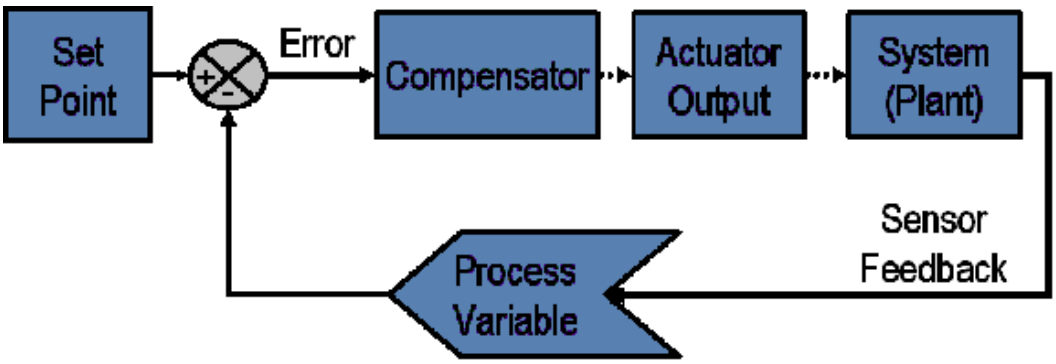
The basic idea behind a PID controller is to read a sensor, then compute the desired actuator output by calculating proportional, integral, and derivative responses and summing those three components to compute the output. Before we start to define the parameters of a PID controller, we shall see what a closed loop system is and some of the terminologies associated with it.

Closed Loop System

In a typical control system, the *process variable* is the system parameter that needs to be controlled, such as temperature (ºC), pressure (psi), or flow rate (liters/minute). A sensor is used to measure the process variable and provide feedback to the control system. The *set point* is the desired or command value for the process variable, such as 100 degrees Celsius in the case of a temperature control system. At any given moment, the difference between the process variable and the set point is used by the control system algorithm *(compensator)*, to determine the desired actuator output to drive the system (plant). For instance, if the measured temperature process variable is 100 ºC and the desired temperature set point is 120 ºC, then the *actuator output* specified by the control algorithm might be to drive a heater. Driving an actuator to turn on a heater causes the system to become warmer, and results in an increase in the temperature process variable. This is called a closed loop control system, because the process of reading sensors to provide constant feedback and calculating the desired actuator output is repeated continuously and at a fixed loop rate as illustrated in figure 1.

In many cases, the actuator output is not the only signal that has an effect on the system. For instance, in a temperature chamber there might be a source of cool air that sometimes blows into the

chamber and disturbs the temperature.Such a term is referred to as *disturbance*. We usually try to design the control system to minimize the effect of disturbances on the process variable.

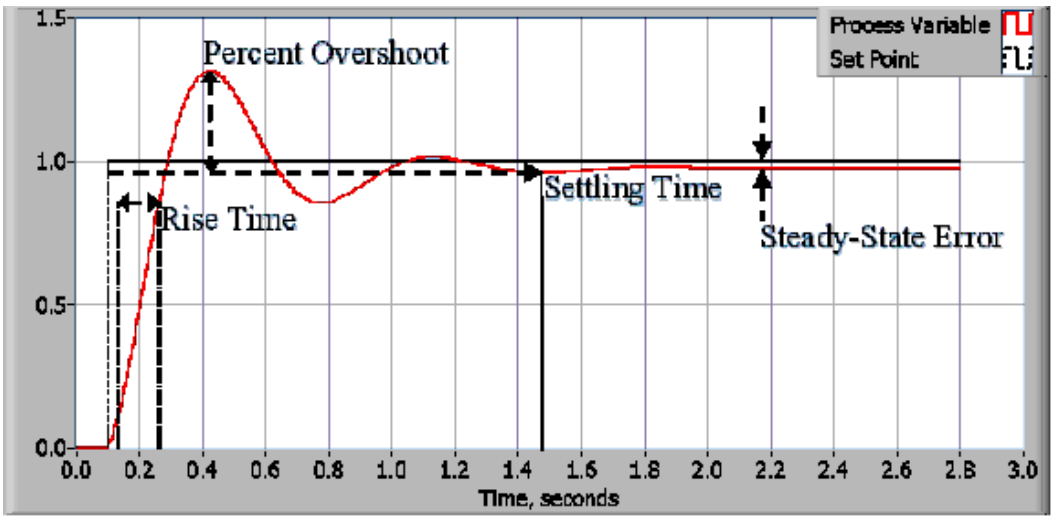


[+] Enlarge Image

Figure 1: Block diagram of a typical closed loop system.

Defintion of Terminlogies

The control design process begins by defining the performance requirements. Control system performance is often measured by applying a step function as the set point command variable, and then measuring the response of the process variable. Commonly, the response is quantified by measuring defined waveform characteristics. Rise Time is the amount of time the system takes to go from 10% to 90% of the steady-state, or final, value. Percent Overshoot is the amount that the process variable overshoots the final value, expressed as a percentage of the final value. Settling time is the time required for the process variable to settle to within a certain percentage (commonly 5%) of the final value. Steady-State Error is the final difference between the process variable and set point. Note that the exact definition of these quantities will vary in industry and academia.



[+] Enlarge Image

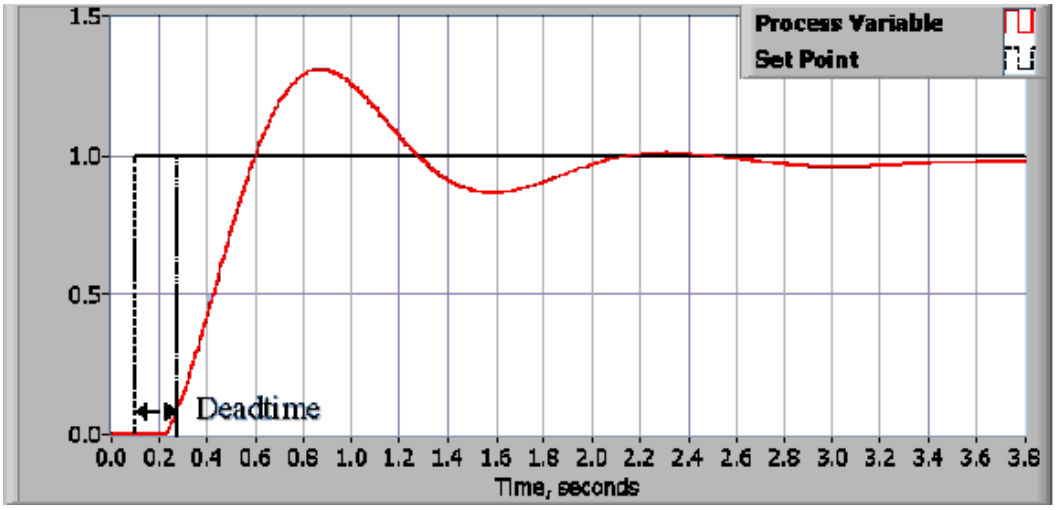
Figure 2: Response of a typical PID closed loop system.

After using one or all of these quantities to define the performance requirements for a control system, it is useful to define the worst case conditions in which the control system will be expected to meet these design requirements. Often times, there is a disturbance in the system that affects the process variable or the measurement of the process variable. It is important to design a control system that performs satisfactorily during worst case conditions. The measure of how well the control system is able to overcome the effects of disturbances is referred to as the *disturbance rejection* of the control system.

In some cases, the response of the system to a given control output may change over time or in relation to some variable. A *nonlinear system* is a system in which the control parameters that produce a desired response at one operating point might not produce a satisfactory response at another operating point. For instance, a chamber partially filled with fluid will exhibit a much faster response to heater output when nearly empty than it will when nearly full of fluid. The measure of how well the control system will tolerate disturbances and nonlinearities is referred to as the *robustness* of the control system.

Some systems exhibit an undesirable behavior called *deadtime*. Deadtime is a delay between when a process variable changes, and when that change can be observed. For instance, if a temperature sensor is placed far away from a cold water fluid inlet valve, it will not measure a change in temperature immediately if the valve is opened or closed. Deadtime can also be caused by a system or output actuator that is slow to respond to the control command, for instance, a valve that is slow to open or close. A common source of deadtime in chemical plants is the delay caused by the flow of fluid through pipes.

Loop cycle is also an important parameter of a closed loop system. The interval of time between calls to a control algorithm is the loop cycle time. Systems that change quickly or have complex behavior require faster control loop rates.



[+] Enlarge Image

Figure 3: Response of a closed loop system with deadtime.

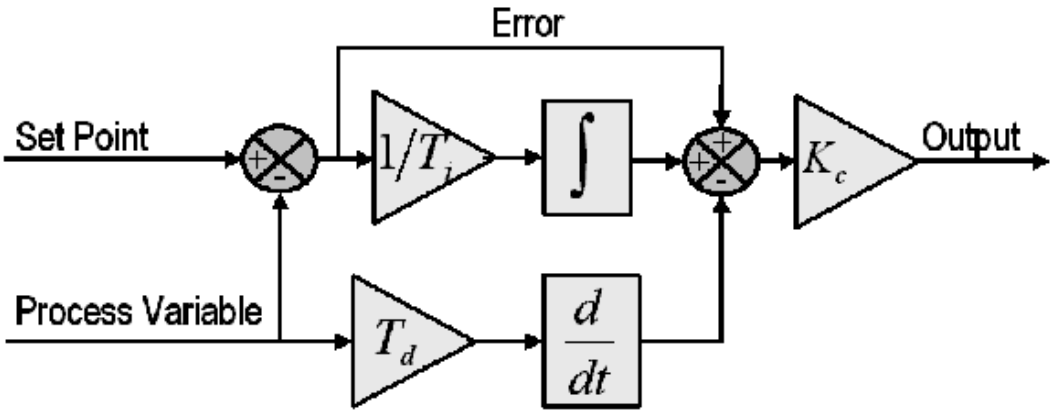
Once the performance requirements have been specified, it is time to examine the system and select an appropriate control scheme. In the vast majority of applications, a PID control will provide the required results

PID Theory

Proportional Response

The proportional component depends only on the difference between the set point and the process variable. This difference is referred to as the Error term. The *proportional gain (K_c)* determines the ratio of output response to the error signal. For instance, if the error term has a magnitude of 10, a proportional gain of 5 would produce a proportional response of 50. In general, increasing the proportional gain will increase the speed of the control system response. However, if the proportional gain is too large, the process variable will begin to oscillate. If K_c is increased further,

the oscillations will become larger and the system will become unstable and may even oscillate out of control.



[+] Enlarge Image

Figure 4: Block diagram of a basic PID control algorithm.

Integral Response
The integral component sums the error term over time. The result is that even a small error term will cause the integral component to increase slowly. The integral response will continually increase over time unless the error is zero, so the effect is to drive the Steady-State error to zero. Steady-State error is the final difference between the process variable and set point. A phenomenon called integral windup results when integral action saturates a controller without the controller driving the error signal toward zero.

Derivative Response
The derivative component causes the output to decrease if the process variable is increasing rapidly. The derivative response is proportional to the rate of change of the process variable. Increasing the *derivative time (T_d)* parameter will cause the control system to react more strongly to changes in the error term and will increase the speed of the overall control system response. Most practical control systems use very small derivative time (T_d), because the Derivative Response is highly sensitive to noise in the process variable signal. If the sensor feedback signal is noisy or if the control loop rate is too slow, the derivative response can make the control system unstable

Tuning

The process of setting the optimal gains for P, I and D to get an ideal response from a control system is called *tuning*. There are different methods of tuning of which the “guess and check” method and the Ziegler Nichols method will be discussed.

The gains of a PID controller can be obtained by trial and error method. Once an engineer understands the significance of each gain parameter, this method becomes relatively easy. In this method, the I and D terms are set to zero first and the proportional gain is increased until the output of the loop oscillates. As one increases the proportional gain, the system becomes faster, but care must be taken not make the system unstable. Once P has been set to obtain a desired fast response, the integral term is increased to stop the oscillations. The integral term reduces the steady state error, but increases overshoot. Some amount of overshoot is always necessary for a fast system so that it could respond to changes immediately. The integral term is tweaked to achieve a minimal steady state error. Once the P and I have been set to get the desired fast control system with minimal steady state error, the derivative term is increased until the loop is acceptably quick to its set point. Increasing derivative term decreases overshoot and yields higher gain with stability but would cause

the system to be highly sensitive to noise. Often times, engineers need to tradeoff one characteristic of a control system for another to better meet their requirements.

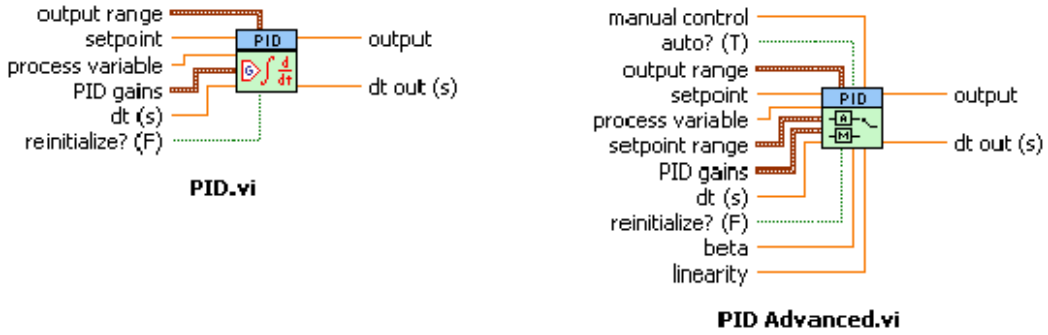
The Ziegler-Nichols method is another popular method of tuning a PID controller. It is very similar to the trial and error method wherein I and D are set to zero and P is increased until the loop starts to oscillate. Once oscillation starts, the critical gain K_c and the period of oscillations P_c are noted. The P, I and D are then adjusted as per the tabular column shown below.

Control	P	Ti	Td
P	0.5Kc	-	-
PI	0.45Kc	Pc/1.2	-
PID	0.60Kc	0.5Pc	Pc/8

Table 1. Ziegler-Nichols tuning, using the oscillation method.

NI LabVIEW and PID

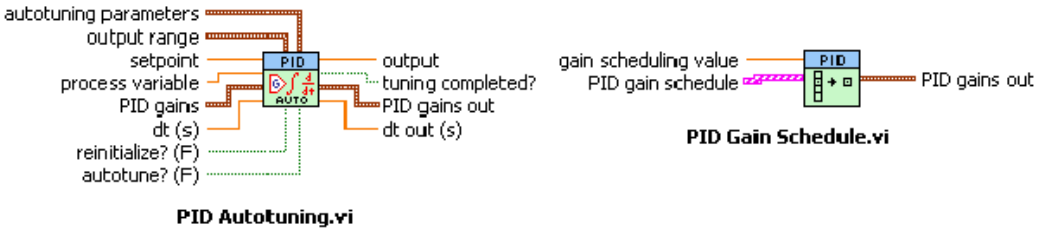
LabVIEW PID toolset features a wide array of VIs that greatly help in the design of a PID based control system. Control output range limiting, integrator anti-windup and bumpless controller output for PID gain changes are some of the salient features of the PID VI. The PID Advanced VI includes all the features of the PID VI along with non-linear integral action, two degree of freedom control and error-squared control.



[+] Enlarge Image

Fig 5: VIs from the PID controls palette of LabVIEW

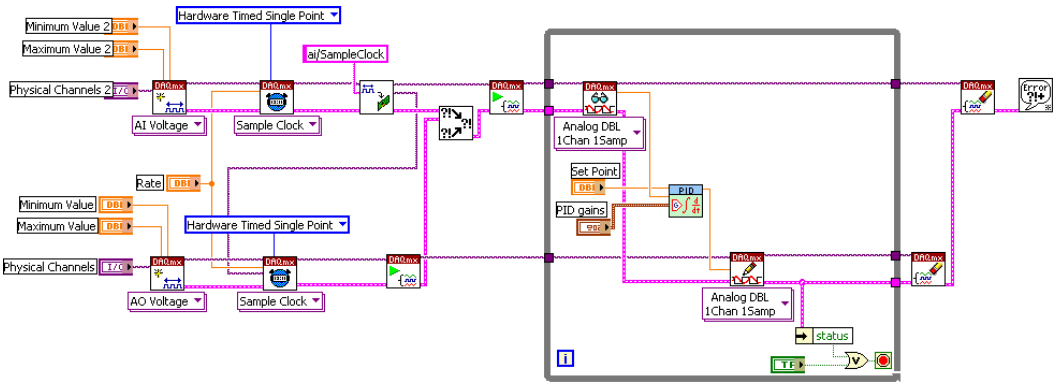
PID palette also features some advanced VIs like the PID Autotuning VI and the PID Gain Schedule VI. The PID Autotuning VI helps in refining the PID parameters of a control system. Once an educated guess about the values of P, I and D have been made, the PID Autotuning VI helps in refining the PID parameters to obtain better response from the control system.



[+] Enlarge Image

Fig 6: Advanced VIs from the PID controls palette of LabVIEW

The reliability of the controls system is greatly improved by using the LabVIEW Real Time module running on a real time target. National Instruments provides the new M Series Data Acquisition boards which provide higher accuracy and better performance than an average control system.



[+] Enlarge Image

Fig 7: A typical LabVIEW VI showing PID control with a plug-in NI data acquisition device

The tight integration of these M Series boards with LabVIEW minimizes the development time involved and greatly increases the productivity of any engineer. Figure 7 shows a typical VI in LabVIEW showing PID control using NI-DAQmx API of M series devices.

Summary

The PID control algorithm is a robust and simple algorithm that is widely used in the industry. The algorithm has sufficient flexibility to yield excellent results in a wide variety of applications and has been one of the main reasons for the continued use over the years. NI LabVIEW and NI plug-in data acquisition devices offer higher accuracy and better performance to make an excellent PID control system.

References

1. **Classical PID Control**
by Graham C. Goodwin, Stefan F. Graebe, Mario E. Salgado
Control System Design, Prentice Hall PTR

2. **PID Control of Continuous Processes**
by John W. Webb Ronald A. Reis
Programmable Logic Controllers, Fourth Edition, Prentice Hall PTR