SENSORES, ATUADORES E CONTROLO

Projeto Final

Prof. Fernão de Magalhães

Prof. Joaquim Gabriel

Mestrado Integrado em Bioengenharia

Engenharia Biomédica, Turma A

João Carlos Ramos Gonçalves de Matos

201704111

Maria Jorge Miranda Loureiro

201704188

Maria Manuel Domingos Carvalho

201706990

Dezembro de 2019

Introdução

Muitas aplicações industriais requerem equipamentos com controlo automático, que atua sobre um processo, que se procura otimizar. Em termos de controlo, uma abordagem muito utilizada é o anel de controlo por retroação, composto pelo controlador, o elemento final de controlo, o processo, e o medidor. O controlador calcula, através de valores medidos, a ação de saída (OP, *Output to Process*) que irá entrar no elemento final de controlo. Daqui, irá resultar uma ação que entra no processo e este retorna a variável manipulada (PV, *Process Value*). De seguida, por retroação, PV irá ser novamente medida pelo medidor e, calculando a diferença entre o valor estabelecido e PV, o valor do erro é obtido, que entrará novamente no cálculo da ação de controlo e o ciclo repete-se.

A programação de um controlador pode ser feita por controlo de ação descontínua ou contínua. No que toca à ação contínua, é possível programar o controlador de diferentes formas, sendo a mais completa utilizando controlo proporcional integral derivativo (PID). Este controlo é definido pela equação 1.

$$C = C^* + K_c \left(\varepsilon + \frac{1}{\tau_i} \int_0^t \varepsilon \, dt + \tau_d \times \frac{d\varepsilon}{dt} \right) \quad \text{Eq. 1}$$

Um sistema pode também ser de ação direta ou inversa: se for de ação direta, uma alteração na ação de controlo irá modificar no mesmo sentido a variável controlada, enquanto se for de ação inversa esta alteração na ação de controlo irá causar uma variação no sentido contrário da variável controlada. Em situações de ação direta, K_c toma valores negativos. Adicionalmente, o controlo pode ser servo ou regulatório. No controlo servo, o controlador acompanha as alterações feitas no valor estabelecido enquanto, no controlo regulatório, o controlador, após uma alteração feita ao sistema, reage de forma a recuperar o PV para o valor estabelecido, que não se altera.

Utilizando estes conceitos, no âmbito da unidade curricular de Sensores, Atuadores e Controlo, realizou-se este projeto, que tem como objetivo implementar num Arduino UNO um algoritmo de controlo PI, otimizando um processo simulado, disponibilizado pelos docentes e implementado em LabVIEW. No presente trabalho, pretende-se que o sistema estabilize no valor estabelecido 20, com erro máximo de $\pm$ 1. Para além disso, pretende-se minimizar o desvio máximo e o tempo de estabilização perante uma perturbação de magnitude 10. O controlador irá operar em controlo regulatório, deve operar em ciclos de 0,4s e com uma amplitude de saída entre 0 e 100.

De modo a implementar o controlo deste sistema, utilizou-se o programa LabVIEW e um Arduino Uno. O controlador PI é implementado, em C, no microcontrolador Arduino e enviado por porta série para o LabVIEW, onde existe um bloco relativo ao simulador do processo, que, neste caso, já inclui todo o processo, elemento final de controlo e medidor. Assim, na interface gráfica do LabVIEW existem os blocos relativos à transmissão de informação do Arduino para este, seguidos de blocos que convertem esta informação de modo a que possa ser lida pelo processo, cuja resposta é mostrada num gráfico PV vs t, armazenada num ficheiro .txt quando desejado, e convertida novamente para poder ser lida pelo Arduino. O esquema da implementação pretendida encontra-se na figura 1.

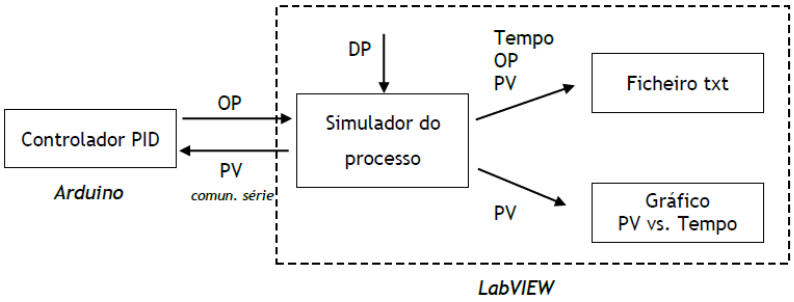

Figura 1 – Fluxo de informação no presente projeto

Após finalizada a construção do algoritmo do código e da interface gráfica, os passos seguintes são a determinação do parâmetro C*, que corresponde à ação necessária para que variável de saída seja constante e igual ao valor estabelecido. Finalmente, é feita a sintonização do controlador, ou seja, a definição de Kc (ganho proporcional) e de τ_i (constante temporal integral), através do método de oscilação contínua, com posterior afinação de parâmetros para otimização dos critérios de desempenho do sistema: desvio máximo e tempo de estabilização.

Métodos e Resultados

Explicação do Código Desenvolvido em LabVIEW

Em LabVIEW, desenvolveu-se o código, recorrendo a diagramas de blocos, capaz de comunicar, por porta série, com o Arduino UNO, no sentido da receção da variável de controlo (OP) e de envio do valor produzido pelo processo (PV). O Arduino, com o valor de PV recebido será capaz de retornar, para o LabVIEW, um valor de OP, como resposta, que irá controlar o comportamento do processo.

O diagrama de bloco desenvolvido divide-se em três partes, através de uma flat sequence com 3 frames: uma 1ª frame, de inicialização; uma 2ª frame, em que é executado um ciclo *while,* para receção, processamento e envio de dados, terminada quando o utilizador desejar, através de um controlo stop; uma 3ª frame, de encerramento do programa.

Na 1ª frame da flat sequence global, é inicializada e definida a porta série para comunicação com o Arduino, através do bloco VISA Configure Serial Port VI; com o bloco VISA Flush I/O Buffer Function são limpos todos os dados remanescentes no buffer. A estes blocos associa-se uma variável local, SERIAL Port, que será a referência da porta série ao longo do programa. Através do bloco Open/Create/Replace File Function é ainda inicializado o ficheiro onde serão guardados os dados produzidos pelo programa, com uma path definida pelo utilizador na interface e no modo open or create, o que significa que o ficheiro tanto pode ser aberto como criado. A variável local File REF, associada a estes blocos, servirá de referência, ao longo do programa, para o ficheiro inicializado. Finalmente, ainda na 1ª frame, é eliminado o histórico dos gráficos OP vs t e PV vs t, presentes na interface gráfica, para que estes estejam estejam vazios de cada vez que o programa seja iniciado.

Na 2ª frame, dentro de um ciclo while, criou-se mais uma flat sequence, com 3 frames.
Na 1ª frame, com o bloco Wait Until Next ms Multiple Function, definiu-se um intervalo de 400 ms por cada iteração do ciclo. Com o bloco Increment Function e um Feedback Node, é criada uma

variável local t que funciona como um contador simples. A começar no -1, a cada iteração, é incrementada em +1, assumindo valores entre 0 e n-1 iterações.

Na 2ª frame, a variável local SERIAL Port é chamada para alimentar o bloco VISA Read Function e indicar a porta série, já inicializada, a utilizar; este bloco será responsável por ler os dados que estão no buffer de entrada da porta série e que, portanto, estão a ser enviados pelo Arduino para o LabVIEW. O bloco VISA Bytes at Serial Port é responsável por indicar, na forma de buffer de leitura, o número de bytes presentes no buffer de entrada, alimentando o bloco responsável por fazer a leitura dos dados. O bloco VISA Read Function tem como parâmetro de saída uma string que contém o buffer de leitura. Esta string será convertida em número com o bloco Fract/Exp String To Number Function uma vez que a os números que estão a ser recebidos no computador são do tipo float. Com uma constante do tipo booleano no estado False, define-se o ponto final como separador decimal; o offset é mantido a 0, por defeito. Deste bloco sai uma variável do tipo número e que corresponde ao OP. Esta variável, já convertida, irá alimentar um bloco Waveform Chart, que gerará, na interface, um gráfico com os valores de OP ao longo do tempo; a variável OP irá também entrar como parâmetro do bloco processo.vi, fornecido pelos docentes. Este bloco tem como parâmetros de entrada OP, DP e PV in; como parâmetro de saída, PV out. A variável DP é introduzida pelo utilizador na interface gráfica através de um bloco de controlo numérico que varia entre -10 e 10; o parâmetro PV out alimenta, por feedback, a variável PV in, através de um Feedback Node, a cada iteração do ciclo. O valor de PV out alimenta, posteriormente, um gráfico do tipo Waveform Chart, que conterá na interface, os valores PV vs t. O valor de PV out é ainda processado num bloco Number To Fractional String Function, com precisão de 4 casas decimais, no qual é produzido uma string. A essa string é acrescentada, com o bloco Concatenate Strings Function, uma string do tipo Carriage Return, um caráter de controlo que irá fazer reset à posição de leitura para o início da linha. Este caráter é muito importante uma vez que esta string irá ser enviada para o Arduino e viabiliza que o número, do tipo float, seja lido corretamente. A string resultante é, finalmente, ligada a um bloco VISA Write Function, alimentado também pela variável local SERIAL Port, que funciona como referência para a porta série a usar. Este bloco será responsável por enviar, para o Arduino, o valor de PV produzido pelo Processo, escrevendo-o no buffer de saída do computador para o Arduino.

Na 3ª frame, o utilizador, na interface gráfica, tem a possibilidade de escolher, através de um botão booleano, acoplado a uma case structure, se deseja gravar (true) ou não (false) os dados recolhidos pelo LabVIEW no ficheiro .txt. Em caso negativo, a case structure não é ativada e o processamento de strings para gravação no ficheiro .txt não acontece; o ciclo while regressa ao início para uma nova iteração. Em caso afirmativo, os dados t, PV e OP são tratados, enquanto strings, para que seja possível a escrita de um ficheiro .txt com os mesmos. Com a variável local t, criada na 1ª frame do ciclo, multiplicando 0,4 s, obtém-se, a cada iteração, o tempo decorrido desde o início do programa. Esta string é concatenada com uma string do tipo tab, para que o ficheiro .txt seja legível e facilmente importado para Excel. À string resultante acrescenta-se uma string que contém o valor de PV out produzido, a que se volta a adicionar uma string do tipo tab, uma string que contém o valor de OP e uma string do tipo end of line, para que cada iteração corresponda a uma linha diferente. Finalmente, a string produzida, que corresponde aos valores de t, PV e OP de um ciclo devidamente formatados, são escritos no ficheiro já aberto através do bloco Write to Text File Function. Este bloco tem como referência a variável local File REF, criada na inicialização do ficheiro.

Na 3ª frame da flat sequence global do programa, usa-se a variável local SERIAL Port e o bloco VISA Close Function para fechar a porta série de comunicação; com a variável local File REF a alimentar o bloco Close File Function, é fechado o ficheiro com os dados produzidos.

Explicação do Código Desenvolvido para o Arduino UNO

Primeiramente, todas as constantes globais são inicializadas, conforme os valores que serão explicados e as variáveis necessárias para o funcionamento do código são também declaradas globais.

De seguida, no programa setup() do Arduino, inicializa-se a comunicação série com a função Serial.begin(), com o parâmetro de 9600, que corresponde ao *Baud Rate*, isto é, a taxa de bits por segundo da comunicação. Este programa corre apenas uma vez, quando o Arduino é ligado.

O programa que se segue é o loop(), ou seja, após o setup() correr uma vez, este irá correr continuamente. Dentro deste, usa-se a função Serial.parseFloat() para recolher os valores de PV, recebidos do LabVIEW. Com estes valores, calcula-se o erro, que corresponde à diferença entre o PV estabelecido, 20, e o recebido naquele ciclo do LabVIEW, que é necessário para o cálculo do controlo. A seguir, calcula-se a acumulação de erro naquele ciclo e, posteriormente, o valor de controlo, usando a equação 1, à exceção da porção derivativa. De seguida, cria-se uma condição if para garantir que os valores de controlo apenas variam entre 0 e 100 e para retirar a acumulação de erro quando o controlo está saturado, ou seja, é igual a 0 ou 100. Para que se verifique o intervalo de 400ms por cada ciclo, utiliza-se a função delay() para esperar 400ms até continuar a correr o código. Finalmente, usa-se a função Serial.print() para enviar o valor calculado do controlo para o LabVIEW.

Procedimento Usado para Determinar o Valor de Controlo de Referência, C*

Para calcular o controlo proporcional integral, segundo a equação 1, é necessário averiguar qual o valor do controlo de referência do sistema, C*. Este valor corresponde à ação necessária para que a variável PV esteja estável e no seu valor de referência, sem perturbações. Assim, para o determinar, eliminámos a porção proporcional e integral do controlo, para que este fosse constante e igual a C*. De seguida, testaram-se vários valores, entre 0 e 100, até se encontrar um C* em que o sistema estabilizasse o PV a 20. O valor encontrado foi de 42,8. O gráfico da variação do PV ao longo do tempo, com o controlo constante e igual a 42,8, encontra-se na figura 2.

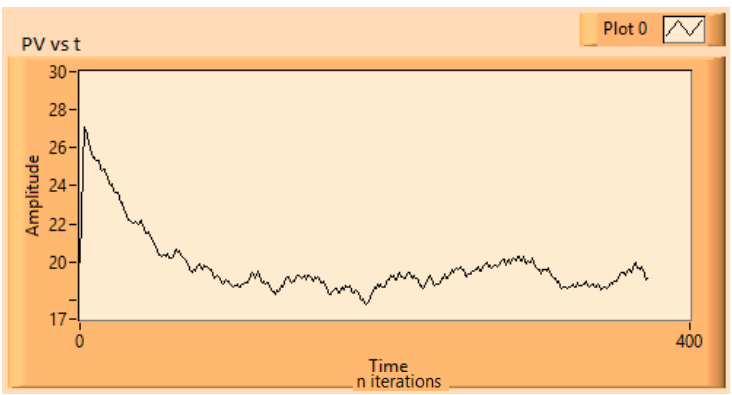

Figura 2 - Gráfico da variação de PV ao longo do tempo com o controlo (C*) igual a 42,8

Sintonização por Oscilação Contínua

Para realizar a sintonização do sistema, utilizou-se o método de oscilação contínua. Com o sistema estabilizado e em modo proporcional, sem componente integral, fez-se uma primeira estimativa para Kc(lim), começando por um valor absoluto baixo, de 1. Os valores de Kc testados foram negativos visto que se observou que o sistema era de ação direta: variações positivas de PV induzem variações positivas de controlo. De seguida, adicionou-se uma pequena alteração, DP, e observou-se a resposta do sistema. Se a oscilação não fosse contínua, isto é, de amplitude constante, aumentava-se, em módulo, Kc(lim). O valor mínimo encontrado, para uma amplitude constante da resposta do sistema, foi -12,1. A variação de PV, ao longo das iterações percorridas, com Kc = -12,1, está na figura 3.

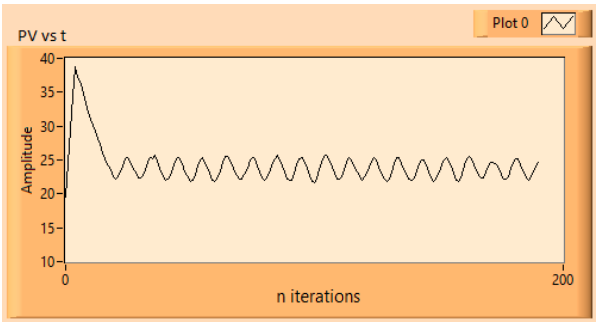

Figura 3 - Gráfico da variação de PV ao longo do tempo com Kc = -12,1

Encontrado o valor de Kc(lim), calculou-se o período limite de oscilação, T(lim). Para isso, recolheram-se os valores de PV ao longo do tempo para um ficheiro .txt, com o sistema ainda em modo proporcional e com um Kc = Kc(lim). De seguida, recolheram-se os instantes temporais de 23 picos consecutivos, calcularam-se 23 períodos e fez-se a média destes, tendo sido obtido um valor de 4,0 s para o T(lim).

Determinado o período, utilizaram-se as relações empíricas de Tyreus-Luyben (Eq. 2 e 3) para calcular o Kc e o τ_i do sistema. Com estas, foram então obtidos os seguintes valores: Kc = -3,751 e τ_i = 8,8s. A variação de PV ao longo do tempo, após a sintonização do sistema, pode ser observada na figura 4.

$$Kc = 0{,}31 \times Kc_{lim} \qquad \text{Eq. 2}$$

$$\tau_i = 2{,}2 \times T_{lim} \qquad \text{Eq.3}$$

Figura 4 - Gráfico da variação de PV ao longo do tempo programa com Kc = -3,751 e τ_i = 8,8s

Critérios de Desempenho do Sistema

Com a sintonização do sistema, obteve-se uma estimativa inicial para os valores de Kc e τ_i. De seguida, averiguaram-se os critérios de desempenho do sistema para afinar estes valores, com o intuito de os minimizar. Os critérios considerados foram o desvio máximo e o tempo de estabilização do sistema. O tempo de estabilização corresponde ao intervalo de tempo necessário para os valores de PV se situarem entre uma banda de ±5 % do valor estabelecido, neste caso, 20, após uma perturbação no sistema; assim, os valores de PV desde sistema, após estabilização, deverão apenas variar entre 19 e 21. O desvio máximo corresponde à maior diferença entre os valores de PV e o valor estabelecido. Ambos os critérios foram analisados perante uma perturbação de magnitude 10.

Após a determinação de Kc e de τ_i através das relações empíricas de Tyreus-Luyben, usam-se estes valores como pontos de partida para afinar o sistema, de modo a otimizar os critérios de desempenho. Fez-se variar Kc mantendo τ_i constante e vice-versa; os resultados encontram-se nas tabelas 1 e 2.

Tabela 1 - Valores de tempo de estabilização e desvio máximo para valores variáveis de Kc

Kc	τ_i (s)	Tempo de estabilização	Desvio Máximo
-4,7	8,8	19,2	10,4
-4,5	8,8	22	10,82
-4,4	8,8	24,4	10,47
-4,2	8,8	22,8	10,42
-4	8,8	20,4	10,97
-3,751	8,8	24,4	10,77
-3,5	8,8	25,2	11,22
-3	8,8	23,2	11,37

Tabela 2 - Valores de tempo de estabilização e desvio máximo para valores variáveis de τ_i

Kc	τ_i (s)	Tempo de estabilização	Desvio máximo
-3,751	9	20,8	10,85
-3,751	8,8	24,4	10,77
-3,751	8,7	22,4	11,27
-3,751	8,5	26,8	11,08
-3,751	8	25,2	10,37
-3,751	7,5	19,6	10,55
-3,751	7	18,4	10,86
-3,751	6,5	15,6	10,68

Como é possível observar, um aumento do valor absoluto do ganho diminui o tempo de resposta e o desvio máximo; no entanto, observam-se oscilações na resposta que se mantêm todas dentro do

intervalo permitido. Assim, pode-se considerar favorável para a otimização do sistema o aumento de Kc. Quando se faz variar τ_i, conclui-se que quando este diminui em módulo, o tempo de resposta melhora, enquanto para o desvio máximo, não foi possível observar uma tendência. Após a análise destes resultados, testaram-se algumas hipóteses de modo a tentar conciliar o efeito destes dois parâmetros. Assim, testaram-se os valores presentes na tabela 3.

Tabela 3 - Valores de tempo de estabilização e desvio máximo para valores variáveis de τ_i e kc

Kc	τ_i (s)	Tempo de estabilização	Desvio Máximo
-5	**5,5**	**13,2**	**9,78**
-5	**6**	**14,0**	**9,79**
-5	6,5	16,0	9,93
-5,5	**5,5**	**14,0**	**9,26**
-5,5	**6,5**	**14,4**	**9,43**

Após a observação destes valores, observamos mais atentamente a representação gráfica das respostas que apresentaram melhores resultados (valores a negrito na tabela 3) :

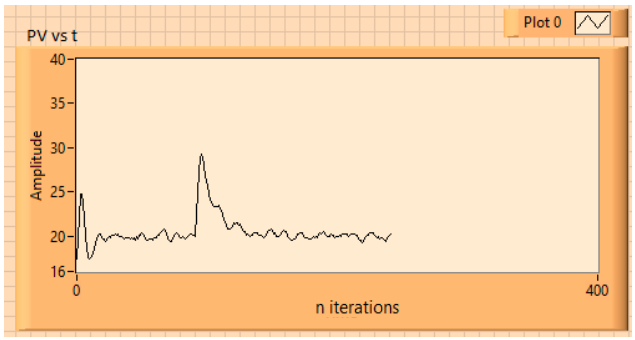

Figura 5 - PV vs t, Kc= -5,5 τ_i = 5,5 s

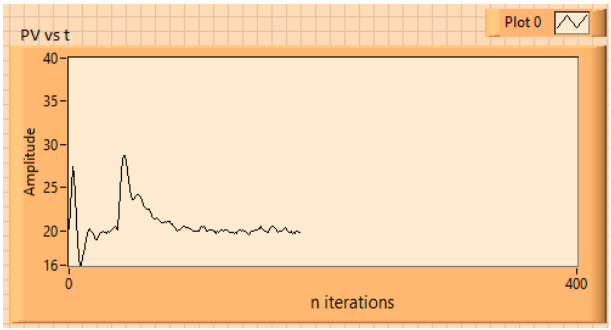

Figura 6 - PV vs t, Kc= -5,5 e τ_i = 6,5s

Figura 7 - PV vs t, Kc= -5 e τ_i = 5,5s

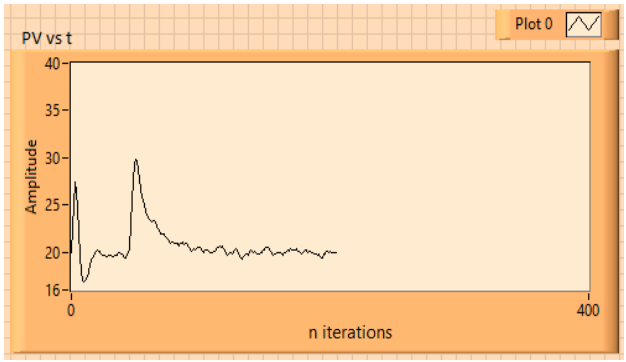

Figura 8 - PV vs t, Kc = -5 e τ_i = 6s

Considerando todos os parâmetros, escolheu-se o modelo com Kc = -5,5 e τ_i = 6,5 s como uma boa solução, uma vez que concilia um bom tempo de estabilização com a própria estabilidade do sistema: apesar de não ter o melhor tempo de estabilização, este continua a ser muito bom, 14,4 segundos, tem um desvio máximo de 9,43 e a sua resposta apresenta muito pouca oscilação.

Discussão de resultados

Após a determinação do controlo de referência, da sintonização e da afinação dos parâmetros PID, o resultado é um sistema estável, que apresenta um desvio máximo de 9,43; e que, após uma perturbação, consegue regressar a um valor de referência após, aproximadamente, 14,4 segundos, intervalo de tempo correspondente ao tempo de estabilização. Estas características podem ser observadas na figura 9, onde é possível ver o sistema a atuar e a recuperar de uma perturbação de magnitude 10.

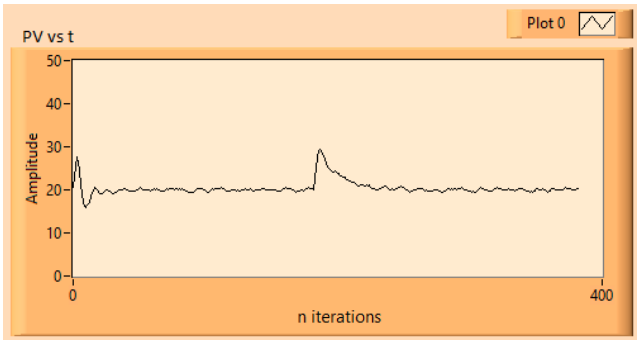

Figura 9 - Gráfico da variação de PV ao longo do tempo com Kc = -5,5 e τ_i = 6,5 s

Os parâmetros deste sistema foram escolhidos numa tentativa de conciliar um tempo de estabilização rápido com um desvio máximo pequeno, e ainda minimizar a oscilação. No entanto, se, por exemplo, o tempo de resposta fosse a primeira prioridade, esta solução não seria a ideal, uma vez que ainda seria possível aumentar a velocidade da resposta, aumentando o ganho e/ou diminuindo o τ_i. Contudo, para existir elevada rapidez na resposta, é necessário sacrificar um pouco a estabilidade e vice-versa. Assim sendo, estes parâmetros podem ser afinados consoante o resultado pretendido, sendo que, neste caso, se optou por equilibrar os critérios de desempenho com uma resposta pouco oscilatória.

Caso fosse pretendida uma otimização que aliasse, ao máximo, uma boa rapidez de resposta a um sistema o mais estável possível, com pouca oscilação, poderia ser usado um controlo PID completo. A componente derivativa em funcionamento seria responsável por minimizar a oscilação do sistema, mantendo um otimizado tempo de resposta. Há, no entanto, que ter em atenção que o ruído poderia ser amplificado. Além disso, seria necessário afinar o sistema com presença de uma 3ª variável, τ_D.

Conclusão

Após a implementação do algoritmo de um controlador PI, no Arduino, e da elaboração do código, em LabVIEW, determinou-se C*, valor do controlo de referência do sistema que torna o anel de controlo

funcional. Enviaram-se para o processo valores constantes que fizessem com que o sistema estabilizasse com PV = 20 e determinou-se valor 42,8 para o controlo de referência.

Após uma análise do comportamento do sistema, conclui-se que se trata de um sistema com ação direta, pelo que Kc deverá ser negativo.

Procedeu-se a uma sintonização por oscilação contínua: com o controlo em modo proporcional, determinou-se o Kc limite e o Período limite. Os valores obtidos foram Kc limite= -12,1 e Plimite = 4 s. Estes valores foram usados na correlação empírica Tyreus-Luyben de modo a calcular um Kc e um τ_i iniciais. Usando estes valores iniciais como um ponto de partida, estes forma feitos variar até se obterem os resultados otimizados do tempo de estabilização e desvio máximo.

Finalmente, escolharam-se como valores finais o controlo com Kc = -5,5 e τ_i = 6,5s. Apesar de não serem os parâmetros que conferem ao sistema os melhores valores de tempo de estabilização (14,4s) e desvio máximo (9,43) per si, trata-se do conjunto de parâmetros que apresenta a melhor relação de compromisso entre rapidez de resposta e estabilidade do sistema.

Referências

[1] Magalhães, F. D. (2019) SAC: Sintonização de Controladores
[2] Gabriel, J. (2019) SAC: Programação Arduino