UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CRISTIANE SILVA GARCIA

AJUSTE BASEADO EM DADOS DE CONTROLADORES: O MÉTODO VRFT EMBARCADO EM UM APLICATIVO MÓVEL

Porto Alegre
2017

CRISTIANE SILVA GARCIA

AJUSTE BASEADO EM DADOS DE CONTROLADORES: O MÉTODO VRFT EMBARCADO EM UM APLICATIVO MÓVEL

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Controle e Automação

ORIENTADOR: Prof. Dr. Alexandre Sanfelice Bazanella

Porto Alegre
2017

CRISTIANE SILVA GARCIA

AJUSTE BASEADO EM DADOS DE CONTROLADORES: O MÉTODO VRFT EMBARCADO EM UM APLICATIVO MÓVEL

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Alexandre Sanfelice Bazanella, UFRGS
Doutor pela UFSC – Florianópolis, Brasil

Banca Examinadora:

Prof. Dr. Mario Cesar Mello Massa de Campos, PETROBRAS
Doutor pela Ecole Centrale Paris – Paris, França

Prof. Dr. Ivan Müller, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dra. Lucíola Campestrini, UFRGS
Doutora pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Coordenador do PPGEE: _____
Prof. Dr. Valner João Brusamarello

Porto Alegre, março de 2017.

DEDICATÓRIA

Dedico este trabalho ao meu companheiro Róger, por todas as vezes em que a vida me mostrou um obstáculo, e você com todo carinho e compreensão, me ajudou a superar.

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus, pois sem Ele jamais teria chegado até aqui.

Agradeço ao Róger, pelo apoio e companheirismo, principalmente nos momentos mais difíceis desta jornada. Obrigada pelo carinho, apoio e compreensão que tens dedicado a mim todos estes anos.

Agradeço também ao meu orientador, professor Alexandre Bazanella, pela orientação e auxílio neste trabalho.

Finalmente, agradeço à minha família por compreender a minha ausência em vários momentos.

RESUMO

O método VRFT (*Virtual Reference Feedback Tuning*) é um método baseado em dados utilizado para ajustar os parâmetros de um controlador. Uma das vantagens deste método é que não é necessário um conhecimento detalhado do modelo do processo a ser controlado. Neste trabalho é apresentado o desenvolvimento de um aplicativo Android que utiliza o método VRFT, e uma versão recursiva do mesmo, para ajustar os parâmetros de um controlador PID com base nos dados de entrada e saída coletados de um experimento realizado no processo. O experimento pode ser configurado, inicializado, monitorado, e interrompido através da aplicação móvel. Para viabilizar a comunicação com o controlador também foram desenvolvidos um gateway Bluetooth-Modbus e um protocolo de comunicação simples, que opera sobre o serviço de comunicação serial do protocolo Bluetooth. A solução proposta foi validada através de experimentos realizados em processos simulados e em um processo real, mostrando bons resultados em ambos os casos.

Palavras-chave: VRFT, controle baseado em dados, protocolo de comunicação, Modbus, Android, Bluetooth.

ABSTRACT

The *Virtual Reference Feedback Tuning* (VRFT) method is a data-driven method employed to tune the parameters of a controller. An advantage of this method lies in the fact that it does not need a detailed knowledge of the process model. This work presents the development of an Android application which employs the VRFT method and a newly developed recursive version of it to tune a PID controller based upon the data collected from an experiment conducted on the process. The experiment can be set up, started, monitored, and stopped from inside the mobile application. To communicate with the controller a Bluetooth-Modbus gateway and an application layer protocol, designed over the Bluetooth serial communication service, are employed, both newly developed. Finally, the entire solution is verified in two steps: first some experiments were conducted on a simulated process, then another experiment was conducted to tune a real commercial controller, both showing good results.

Keywords: VRFT, data-based control, communication protocol, Modbus, Android, Bluetooth.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACL	*Asynchronous ConnectionLess*
ADB	*Android debbuger bridge*
ADT	*Android development tools*
API	*Application programming interface*
ARQ	*Automatic Repeat reQuest*
ASCII	*American standard code for information interchange*
CRC	*Cyclic Redundacy Check*
DLL	*Data Link Layer*
FCS	*Frame check sequence*
FEC	*Forward Error Correction*
FHSS	*Frequency-Hopping Spread Spectrum*
FT	Função de transferência
HCI	*Host Controller Interface*
IDE	*Integrated development environment*
IP	*Internet protocol*
ISM	*Industrial, scientific, and medical*
L2CAP	*Logical Link Control and Adaptation Protocol*
MIMO	*Multi-Input Multi-Output*
OSI	*Open systems interconnection*
P	Proporcional
PI	Proporcional e integral
PID	Proporcional, integral e derivativo
PRBS	*Pseudo-random binary sequence*
RF	Rádio frequência
RFCOMM	*Radio Frequence Communications*
RTU	*Remote terminal unit*

SCO	*Synchronous Connection-Oriented*
SDK	*Software development kit*
SISO	*Single-Input Single-Output*
SRp	Suficientemente rico de ordem p
TCP	*Transmission control protocol*
TRAS	*Twin rotor aerodynamic system*
TTL	*Transistor-to-transistor level*
VI	Variável instrumental
VRFT	*Virtual Reference Feedback Tuning*
2DoF	Dois graus de liberdade

LISTA DE SÍMBOLOS

$C(z,\rho)$	Função de transferência do controlador
$C(z,\rho_d)$	Controlador ideal
$C_d(z)$	Controlador ideal
$\bar{C}(z)$	Vetor que representa a classe de controladores
$\bar{e}(t)$	Erro de seguimento de referência virtual
$\varphi(t)$	Vetor regressor
$\Phi_x(e^{j\omega})$	Espectro do sinal $x(t)$
$G(z)$	Função de transferência do processo
$J(\rho)$	Função custo do erro de seguimento de referência
$J^{\mathrm{VR}}(\rho)$	Função custo do método VRFT
$J_y(\rho)$	Função custo de seguimento de referência
K_i	Ganho integral
K_p	Ganho proporcional
K_v	Ganho derivativo
$M(z)$	Modelo de referência
$r(t)$	Sinal de referência
$\bar{r}(t)$	Referência virtual
ρ	Vetor de parâmetros
ρ_d	Vetor de parâmetros desejado
$T(z,\rho)$	Função de transferência do sistema em laço fechado com o controlador $C(z,\rho)$
$u(t)$	Sinal de entrada
$y(t)$	Sinal de saída
$y(t,\rho)$	Resposta em laço fechado do sistema
$y_d(t)$	Saída desejada

1 INTRODUÇÃO

O ajuste dos parâmetros de um controlador pode ser realizado através de métodos baseados no modelo ou através de métodos baseados em dados. Em métodos baseados no modelo, geralmente, é necessário um conhecimento aprofundado do modelo do processo a ser controlado. Já em métodos baseados em dados, como é o caso do método VRFT (*Virtual Reference Feedback Tuning*), a estimativa dos parâmetros do controlador é realizada a partir dos dados de entrada e saída do processo. Tais dados podem ser adquiridos durante a operação normal do sistema, ou durante a execução de um experimento específico.

O VRFT é um método baseado em dados utilizado para ajustar os parâmetros de um controlador, no qual não é necessário conhecer completamente a função de transferência do processo a ser controlado. Além disso, este método possui a vantagem de ser um método não iterativo, ou seja, os parâmetros do controlador são estimados a partir dos dados de um único experimento, segundo Bazanella, Campestrini e Eckhard (2012).

Em termos gerais, para o cálculo dos parâmetros através do método VRFT é preciso determinar o tipo de controlador que será empregado no processo e, além disso, é necessário também especificar o comportamento desejado do sistema em laço fechado. Uma vez especificados estes dois itens, o objetivo do método é encontrar um conjunto de ganhos para o controlador, cuja estrutura foi previamente definida, que tornará o comportamento do sistema em laço fechado, o mais próximo possível do comportamento desejado.

Independente do método utilizado para o ajuste dos parâmetros de um controlador, é bastante conveniente ter uma ferramenta à mão capaz de calcular estes parâmetros, e configurá-los no processo automaticamente, tornando mais prática e simples a tarefa de ajustar os parâmetros de um controlador. Atualmente, dispositivos móveis como *tablets* e *smartphones* são ferramentas largamente disseminadas que possuem poder computacional suficiente para uma tarefa como esta, apesar de não serem muito utilizadas para este propósito.

Com tudo isso em mente, neste trabalho, propõem-se o desenvolvimento de uma aplicação Android para calcular os parâmetros de um controlador através do método VRFT e configurá-los em um controlador. Para viabilizar a comunicação entre o aplicativo Android e o controlador foi desenvolvido um gateway Bluetooth-Modbus. A comunicação entre o aplicativo Android e o gateway é realizada através da tecnologia Bluetooth, através de um protocolo simples de comunicação que também foi desenvolvido e que opera sobre esta mesma tecnologia. Já a comunicação entre o gateway e o controlador é realizada via Modbus, uma vez que este é o protocolo disponível no controlador escolhido. O gateway foi construído utilizando um Arduino Mega, uma plataforma de prototipação também bastante disseminada, em conjunto com dois módulos que permitem a comunicação Bluetooth e Modbus. A rede formada pelos dispositivos é ilustrada a através do diagrama de blocos apresentado na Figura 1.

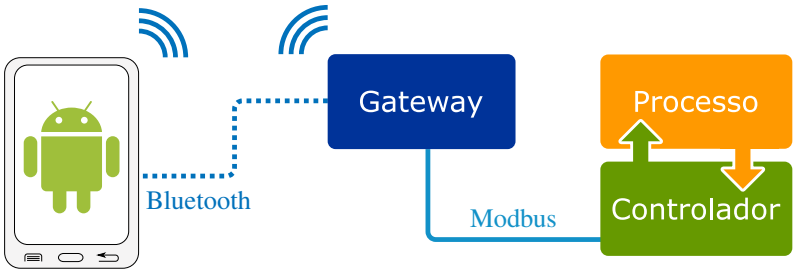


Figura 1: Diagrama geral da proposta.

No aplicativo Android tanto a abordagem original do método VRFT como uma abordagem recursiva do mesmo foram implementadas. Esta versão recursiva se adapta melhor às restrições dos dispositivos móveis, por ser uma abordagem com menor custo computacional em termos de processamento e memória. Além disso, esta abordagem possibilita que os parâmetros sejam estimados a cada novo dado coletado, eliminando a necessidade de aguardar o término do experimento para visualizar os parâmetros calculados. Por conta disso, esta abordagem traz duas principais vantagens: ela elimina a necessidade de armazenar em memória todos os dados coletados, ao mesmo tempo em que também libera o projetista da necessidade de determinar previamente a quantidade de dados a serem coletados no experimento. Além disso, algumas outras funcionalidades foram implementadas no aplicativo, como armazenar os dados adquiridos, e visualizá-los de forma *on-line* por meio de um recurso gráfico, aumentando ainda mais a praticidade do processo de ajuste do controlador.

Já a aplicação que é executada no gateway permite que os experimentos sejam realizados com baixa latência de comunicação com o controlador, ao mesmo tempo que alivia um pouco a carga de processamento imposta sobre o dispositivo Android. Isto é alcançado pois o gateway Bluetooth-Modbus é responsável por realizar a comunicação entre o aplicativo e o processo, convertendo as mensagens de alto nível enviadas através do aplicativo Android em sequências de comandos Modbus de mais baixo nível, e vice-versa. Para tanto um novo protocolo teve de ser desenvolvido, implementado os comandos necessários para a comunicação entre o aplicativo e o gateway.

Uma contextualização mais aprofundada e uma explicação mais detalhada de cada um dos componentes desenvolvidos neste trabalho será apresentada nos capítulos seguintes. No Capítulo 2 são apresentados os trabalhos encontrados na literatura que servem de contexto para o presente trabalho. No Capítulo 3 é apresentado o método VRFT juntamente com as abordagens pertinentes a este trabalho. O protocolo de comunicação desenvolvido é apresentado no Capítulo 4, e o gateway é apresentado no Capítulo 5. Já no Capítulo 6 é descrito em detalhes o aplicativo Android desenvolvido. Finalmente, no Capítulo 7 são apresentados os experimentos realizados para validação da solução desenvolvida.

2 TRABALHOS RELACIONADOS

Este capítulo é dedicado a apresentar a literatura que considera-se relacionada ao presente trabalho. Desta forma, inicialmente, serão apresentados alguns trabalhos que descrevem e/ou utilizam o método VRFT, ou um caso específico do mesmo. Note-se aqui que esta não é uma listagem exaustiva de todas as variações do método, mas apenas um conjunto das que mais compartilham do contexto deste trabalho. Em seguida, também serão descritos alguns trabalhos relacionados à aplicação de dispositivos móveis no controle de processos.

2.1 O VRFT e suas variações

O método VRFT foi originalmente apresentado no trabalho de Campi, Lecchini e Savaresi (2002), no qual foi apresentada uma versão do método já própria para uso (*ready-to-use*). As principais características do método foram apresentadas neste trabalho, dentre elas pode-se citar que o VRFT é um método direto, ou seja, os parâmetros são estimados diretamente a partir dos dados de entrada e saída do sistema, desta maneira, evitando a etapa de modelagem do processo. Outra característica do VRFT, é que este método não é iterativo, isto é, no caso ideal são utilizados apenas os dados de entrada e saída de um único experimento para estimar os parâmetros do controlador. Além disso, o algoritmo não requer inicialização, e para a aquisição dos dados não é necessário realizar um experimento específico, ou seja, podem ser utilizados os dados tanto de um experimento em laço aberto quanto de um experimento em laço fechado, desde que o sinal de entrada seja suficientemente rico. A riqueza do sinal de entrada será melhor detalhada no Capítulo 3.

No trabalho citado foram considerados sistemas SISO (*Single-Input Single-Output*) lineares e invariantes no tempo. Além disso, foram considerados casos em que o comportamento desejado do sistema em malha fechada pode ou não ser alcançado com o controlador disponível. Por fim, também foi ilustrado o emprego do método para o caso em que o sistema é afetado por ruído de forma significativa.

Em outro trabalho dos mesmos autores (CAMPI; LECCHINI; SAVARESI, 2003) o método foi aplicado em um sistema de suspensão ativa. Nesta caso, o problema de controle foi solucionado através da utilização de um controlador de ordem elevada.

Algumas variações do método foram desenvolvidas ao longo do tempo. Ainda em outro trabalho dos mesmos autores (LECCHINI; CAMPI; SAVARESI, 2002), o método foi estendido para lidar com controladores com dois graus de liberdade (2DoF), onde um controlador aparece na malha interna e o outro na malha externa. A estimativa dos parâmetros dos dois controladores é realizada através dos dados de um único experimento e, além disso, o trabalho também apresenta como ajustar os parâmetros de controladores com 2DoF quando o processo é afetado por ruído.

Nakamoto (2005) estende a aplicação do método para sistemas MIMO (*Multi-Input Multi-Output*). O seu trabalho apresenta um algoritmo simples para ajustar os parâmetros dos controladores deste tipo de sistema e, para finalizar, o autor apresenta a aplicação deste algoritmo em um processo real com duas entradas e duas saídas. Além disso, é realizada uma comparação entre os resultados obtidos através do algoritmo proposto e dos resultados obtidos através da aplicação do VRFT para o caso SISO nos dois controladores em separado.

No trabalho de Campi e Savaresi (2006) o VRFT foi empregado no ajuste dos parâmetros de sistemas SISO não-lineares. Novamente foram apresentadas as abordagens do método para os casos em que o comportamento desejado do sistema em malha fechada pode ou não ser alcançado com o controlador disponível. Neste último caso uma técnica de filtragem dos sinais coletados é utilizada para que o comportamento final seja mais próximo do esperado. Além disso, também foi apresentado como lidar com o caso ruidoso.

Por fim, uma adaptação do método foi desenvolvida por Campestrini, Gevers e Bazanella (2009) para lidar com plantas de fase não mínima. Segundo os autores, se o processo possui zeros de fase não mínima e estes não são incluídos no modelo desejado para sistema em laço fechado, isto pode levar a sérios problemas de instabilidade. A solução proposta é para processos SISO lineares e, no algoritmo proposto, os zeros de fase não mínima são identificados juntamente com os parâmetros do controlador.

A forma como o método é apresentado nestes trabalhos não facilita a sua aplicação em sistemas com restrições de processamento e quantidade de memória disponível. Isto resulta na necessidade de utilização de um computador para realizar o cálculo e o ajuste dos parâmetros, reduzindo a praticidade da solução. Por fim, nenhum destes trabalhos considera o caso em que algum parâmetro do processo pode ser alterado durante o seu regime de trabalho (sistemas variantes no tempo). Deste modo, caso ocorra alguma variação, é necessário um novo experimento para possibilitar o cálculo dos novos parâmetros do controlador. O presente trabalho visa preencher estas lacunas deixadas na literatura, através do desenvolvimento de outra variação do método. Esta variação emprega um algoritmo mais leve, que pode ser utilizado em um dispositivo móvel, e uma técnica para utilização em sistemas variantes no tempo.

2.2 Emprego do VRFT na literatura

Por conta das suas características, o VRFT foi considerado no ajuste dos parâmetros de diversos sistemas de controle. No trabalho de Filho et al. (2016) o método VRFT foi utilizado para controlar o nível de dois tanques através de duas válvulas, uma em cada tanque. Foi utilizada uma estrutura de controle descentralizada de modo que, para controlar o sistema, foram projetados dois controladores SISO, cada um responsável pela abertura de uma das válvulas. O primeiro controlador é do tipo proporcional (P) e foi projetado com restrições para que os zeros de fase não mínima aparecessem no segundo controlador, este com ação proporcional e integral (PI). Além disso, a abordagem do VRFT para o caso ruidoso foi utilizada para ajustar os parâmetros do segundo controlador, uma vez que o processo apresenta ruído. Por fim, o trabalho apresenta os testes realizados com os dois controladores no processo e, segundo os autores, os zeros de fase não mínima foram estimados corretamente e o resultado obtido com o controlador estimado foi bem próximo do resultado esperado.

No trabalho proposto por Previdi et al. (2016), foi desenvolvido um algoritmo que emprega o VRFT em sistemas que necessitam de controladores em cascata. Neste caso,

o sistema a ser controlado era composto por duas malhas de controle, e a proposta foi encontrar os dois controladores de uma única vez utilizando apenas as informações de uma batelada de dados. Um estudo de caso foi realizado, onde o algoritmo proposto foi aplicado no controle de um atuador eletro-hidráulico.

Já Previdi et al. (2010), utilizaram o método VRFT para controlar a velocidade de um manipulador manual auto-balanceado utilizado no transporte de pequenas cargas. De acordo com os autores, o auto-ajuste dos parâmetros do controlador no final da linha de produção era desejável, uma vez que a dinâmica do processo poderia variar ao longo do tempo, por isso a utilização do VRFT, por este ser um método direto e não iterativo. Para cumprir a tarefa foi utilizado um controlador PI e os parâmetros foram estimados através de uma única batelada de dados.

Corleta et al. (2016) emprega o método VRFT em fontes de energia ininterruptas (UPS), com o objetivo de melhorar o desempenho transitório. Neste trabalho foram projetados dois controladores: um controlador ressonante, e um controlador de ganho de corrente na realimentação, este último ajustado através de uma variação do método VRFT. O desempenho do controlador ajustado pelo método VRFT foi semelhante ao de um controlador ajustado através de LMI (*linear matrix inequalities*), um método baseado em modelo. Neste caso o método adaptado foi validado apenas através de simulações, embora tenha potencial para ser implementado embarcado na própria UPS.

No trabalho de Hedrea, Radac e Precup (2016), o VRFT foi empregado no controle de um processo MIMO não-linear para o controle de posição de um sistema aerodinâmico (*twin rotor aerodynamic system* – TRAS). O objetivo do trabalho foi controlar os ângulos de atitude e de azimute do processo através de controladores lineares. Nos testes realizados foram utilizados tanto o VRFT para o caso SISO como para o caso MIMO, e para ambos os casos, foram testados vários controladores com ordens diferentes.

Rojas, Tadeo e Vilanova (2010) apresentam em seu trabalho o método VRFT aplicado no controle de um sistema SISO não-linear. O sistema em questão tem por objetivo o controle de neutralização do pH de uma planta, e um controlador linear de ordem restrita com 2DoF foi utilizado para alcançar este objetivo. Os testes foram realizados primeiramente em um ambiente simulado, onde foi utilizado um modelo não-linear do processo, e na sequência, também foram conduzidos testes em uma planta real.

Já em Rojas, Morilla e Vilanova (2016) o método VRFT foi utilizado no controle de um sistema MIMO não-linear. Neste caso, o método foi empregado no ajuste do controlador de uma caldeira e o sistema controlado era composto por três entradas e três saídas. Para atingir os objetivos de controle foram empregados controladores do tipo PI. Já para estimar os parâmetros dos controladores, duas abordagens foram utilizadas: na primeira, os controladores foram estimados de forma descentralizada, cada um sendo influenciado pela sua respectiva entrada, e na segunda, a estimativa de cada controlador era influenciada por todas as entradas.

Rojas et al. (2012) aplicaram o VRFT no controle do tratamento de águas residuais — outro processo não-linear. Neste trabalho, foi utilizada uma variação do método para o caso multivariável, uma vez que o sistema considerado possui duas ou três entradas e o mesmo número de saídas. Desta vez, para atingir os objetivos de controle, quatro ou seis controladores PI foram empregados. Um simulador foi utilizado para realizar os testes e várias simulações foram feitas, em cada uma foram utilizadas diferentes combinações de entradas e saídas possíveis. Os autores concluem que a aplicação do método apresentou bons resultados.

Os trabalhos citados acima se diferenciam do presente trabalho pois não deixam claro

que tipo de dispositivo foi empregado para o cálculo dos parâmetros, ficando implícito que foi um computador. Já o presente trabalho implementa o cálculo dos parâmetros em um dispositivo móvel e, para tanto, emprega um algoritmo recursivo mais leve e compatível com este dispositivo. Além disso, o método também é estendido para permitir a sua utilização em processos variantes no tempo.

2.3 Emprego de dispositivos móveis no controle de processos

A ideia de utilizar um dispositivo móvel como um *tablet* ou *smarphone* em conjunto com técnicas de controle não é nova. Na literatura já são encontrados alguns trabalhos com uma proposta neste tema. Em geral, estes trabalhos apresentam propostas como o monitoramento do processo, o cálculo de parâmetros, ou o envio dos mesmos para ajustar o funcionamento do processo. Entretanto, não foi encontrado nenhum trabalho que realizasse estas três tarefas através de um único dispositivo móvel. A seguir são apresentados alguns destes trabalhos encontrados na literatura.

Alexander (2015) apresenta o desenvolvimento de um aplicativo para o sistema operacional Android para monitoramento dos parâmetros de uma planta industrial. Os parâmetros monitorados são a temperatura e o nível de um tanque. Para comunicação entre a planta e o aplicativo foi utilizado um microcontrolador em uma placa Arduino, que é ligado diretamente à planta e coleta os dados da mesma. Os dados coletados são enviados através da tecnologia Bluetooth para a aplicação Android, onde são armazenados no dispositivo. Por meio do aplicativo é possível monitorar os dados coletados, contudo, nenhuma técnica de controle foi implementada.

Pratumsuwan e Chanaisawan (2016) apresentam um aplicativo Android desenvolvido para auxiliar no monitoramento e controle de um sistema servo pneumático. Os dados dos sinais do sistema são coletados por um microcontrolador e ficam disponíveis ao aplicativo via Wi-Fi. Através do aplicativo é possível coletar, monitorar e configurar os parâmetros de um controlador PID. Porém, o cálculo dos parâmetros não é realizado no aplicativo, os parâmetros são calculados através do Matlab, informados manualmente no aplicativo, e então enviados ao microcontrolador, que realiza o ajuste do mesmos no sistema. Nos testes realizados, o Matlab foi utilizado para estimar o modelo da planta a partir dos dados coletados através do aplicativo. Em seguida, com a estimativa da planta, foi utilizada a ferramenta Simulink do Matlab para encontrar os parâmetros do controlador. Ou seja, nenhuma técnica de controle baseado em dados foi empregada, pois os parâmetros foram calculados a partir do modelo estimado da planta.

Arrieta et al. (2015) desenvolveram um aplicativo Android para auxiliar na tarefa de projetar controladores com ação proporcional, integral e derivativa (PID). Para tanto, é necessário informar o modelo do processo a ser controlado e escolher a classe de controle entre PI ou PID com 1DoF ou 2DoF. Contudo, esta ferramenta foi desenvolvida apenas para uso educacional. Além disso, nenhuma técnica de controle baseado em dados foi empregada na estimativa dos parâmetros do controlador, e nenhum tipo de comunicação com alguma real planta foi implementado.

Faragher et al. (2015) propõe a utilização de um *smartphone* no controle de um *drone*. Neste caso, o *smartphone* é utilizado como hardware principal, ou seja, todo o controle é realizado pelo *smartphone* que é agregado fisicamente ao *drone*. Para o ajuste dos parâmetros, é utilizado um controlador do tipo PID. Através de uma aplicação que é executada em um computador é possível calcular e enviar os parâmetros do controlador para o *drone*, durante o voo, através de Wi-Fi. Segundo os autores, a abordagem sugerida

apresentou bons resultados. Apesar da utilização de um *smartphone*, o seu poder de processamento não foi aproveitado para o cálculo dos parâmetros e nem a sua tecnologia Bluetooth foi aproveitada para a comunicação.

Claudi et al. (2013) apresentam em seu trabalho o desenvolvimento de um robô autônomo para reconhecimento e seguimento de faces em tempo-real para monitoramento de sistemas. Neste caso, um aplicativo Android foi desenvolvido para adquirir e monitorar os dados coletados. No aplicativo um algoritmo foi implementado para ajustar a posição do robô com base nas imagens coletadas. O ajuste desta posição é realizado através de um controlador do tipo PID implementado no microcontrolador agregado ao hardware do robô e alimentado por um sinal de erro gerado pelo aplicativo. Neste trabalho, não é apresentado como os parâmetros do controlador foram calculados e não fica claro se foi empregado algum tipo de comunicação sem fio. De qualquer modo, segundo os autores a abordagem mostrou bons resultados.

O presente trabalho se diferencia destes apresentados acima pois utiliza o poder de processamento do dispositivo móvel para estimar o melhor conjunto de parâmetros do controlador. Esta estimativa utiliza um método baseado em dados (VRFT) adaptado para as restrições que um sistema deste tipo apresenta. Além disso, os dados e os parâmetros trafegam via Bluetooth, permitindo que todo o sistema seja independente de qualquer infraestrutura de rede adicional. Com tudo isto em vista, espera-se aumentar a praticidade da tarefa de ajuste dos parâmetros do controlador de um sistema real.

3 VRFT

O método VRFT é um método baseado em dados, utilizado para ajustar os parâmetros do controlador a partir dos dados de entrada e saída do processo. Além disso, este é um método não iterativo, ou seja, os parâmetros do controlador são estimados utilizando-se os dados de um único experimento.

Nas seções seguintes são apresentadas, de forma detalhada, as abordagens do método VRFT que são pertinentes a este trabalho, sendo estas o caso ideal, o caso descasado, e o caso em que o processo é de fase não mínima. Além disso, serão apresentadas uma abordagem recursiva do método VRFT, e o fator de esquecimento recursivo.

3.1 Caso ideal

O método VRFT pode ser empregado no ajuste dos parâmetros tanto de um controlador parametrizado linearmente, como de um controlador não parametrizado linearmente. Neste trabalho, considera-se o caso em que a função de transferência do controlador é linearmente parametrizada, por este ser o tipo de controlador encontrado na maioria dos controladores utilizados na indústria. Além disso, assumindo-se que o controlador é deste tipo torna mais simples a análise do problema de projeto do mesmo.

A função de transferência do controlador parametrizado linearmente pode ser escrita da seguinte forma

$$C(z,\rho) = \rho^{\mathrm{T}}\bar{C}(z), \tag{1}$$

onde, ρ é o vetor de parâmetros, $\bar{C}(z)$ é um vetor que representa a classe de controladores, e $C(z,\rho)$ é a função de transferência do controlador.

O VRFT tem como objetivo minimizar a função custo do critério de seguimento de referência $J_y(\rho)$, que leva em consideração apenas a resposta do sistema em laço fechado, $y(t,\rho)$, dada pela seguinte equação:

$$y(t,\rho) \triangleq T(z,\rho)r(t),$$

onde $T(z,\rho)$ é a função de transferência do sistema em laço fechado com o controlador $C(z,\rho)$, e $r(t)$ é o sinal de referência aplicado à entrada do sistema.

O objetivo de controle é tornar a saída do processo tão próxima quanto possível da entrada a ser seguida. Por conta disso, deseja-se minimizar o critério de desempenho do erro de seguimento de referência, dado por

$$J(\rho) = \bar{E}\left[r(t) - y(t,\rho)\right]^2.$$

Define-se $\bar{E}[.]$ como

$$\bar{E}(x^2(t)) \triangleq \lim_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} E[x^2(t)]$$

onde $E[x^2(t)]$ é a esperança do quadrado da variável aleatória $x(t)$.

Como, na prática, não é possível ter seguimento de referência com erro nulo para todo t, este critério é relaxado, tornando-se uma especificação de quão próximas a saída e a referência devem ser, de modo a ainda ser satisfatório. Para tanto, é definido um modelo de referência para o sistema $M(z)$, que representa o comportamento desejado do sistema em laço fechado, obtido a partir de critérios como sobrepasso e tempo de acomodação. Assim, a saída desejada do sistema $y_d(t)$, dada por

$$y_d(t) = M(z)r(t),$$

deve ser tão próxima quanto possível da saída real do sistema $y(t,\rho)$.

Desta forma, o projeto de controle consiste em encontrar um conjunto de parâmetros ρ para o controlador que, quando adicionado ao sistema em laço fechado, fará com que a saída obtida $y(t,\rho)$ seja tão próxima quanto possível da saída desejada $y_d(t)$. Portanto, o critério de desempenho de seguimento de referência é definido como

$$\begin{aligned} J_y(\rho) &\triangleq \bar{E}\left[y(t,\rho) - y_d(t)\right]^2 \\ &= \bar{E}\left[\big(T(z,\rho) - M(z)\big)r(t)\right]^2. \end{aligned} \tag{2}$$

Por conta do formato desta função custo, o VRFT busca transformar o problema de minimizar a função $J_y(\rho)$, relacionada ao desempenho de seguimento de referência, em um problema de minimizar uma outra função custo $J^{\mathrm{VR}}(\rho)$, relacionada a um problema de identificação do controlador ideal. Isso porque, sob algumas condições, o mínimo de $J_y(\rho)$ é igual ao mínimo de $J^{\mathrm{VR}}(\rho)$ e, uma vez que $J^{\mathrm{VR}}(\rho)$ é uma função quadrática, encontrar o mínimo desta função é mais simples do que encontrar o mínimo de $J_y(\rho)$, como ilustrado através da Figura 2. As condições para que os mínimos globais das duas funções custo, $J_y(\rho)$ e $J^{\mathrm{VR}}(\rho)$, coincidam, são traduzidas no que é chamado de caso ideal.

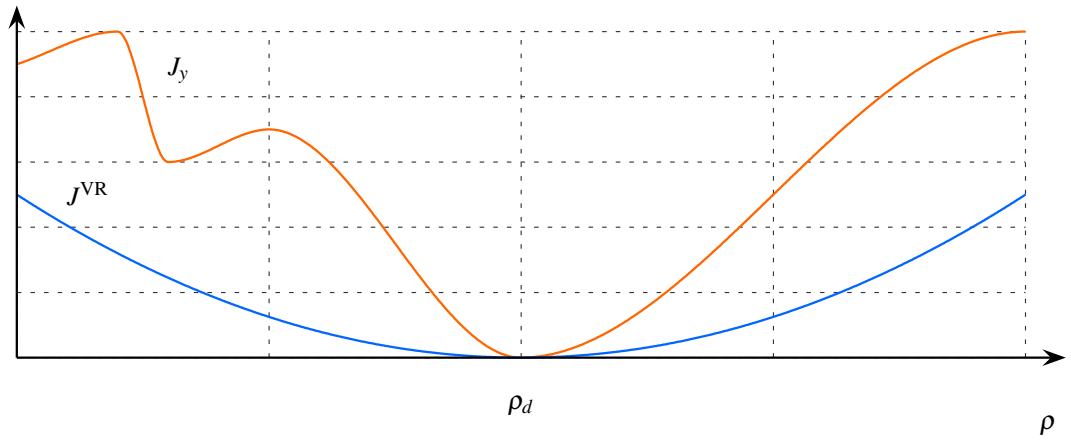


Figura 2: Exemplo de resposta das funções custo $J_y(\rho)$ e $J^{\mathrm{VR}}(\rho)$ para o caso de um parâmetro escalar Fonte: Adaptado de Bazanella, Campestrini e Eckhard (2012).

No caso ideal considera-se que o sistema não é afetado por ruído. Na prática nenhum sistema físico é completamente livre de ruído, entretanto, quando o nível do ruído é considerado pequeno quando comparado aos sinais gerados pelo sistema, isto não torna-se um grande problema, podendo ser negligenciado. Além da ausência de ruído, no caso ideal são consideradas duas outras suposições que devem ser satisfeitas.

Suposição 1 *O controlador ideal pertence à classe de controle, $C_d(z) \in \mathscr{C}$, ou seja,*

$$\exists \rho_d : C(z, \rho_d) = C_d(z).$$

A primeira suposição diz que existe um vetor de parâmetros desejados, ρ_d, que fará com que o sistema em malha fechada tenha o comportamento desejado, ou seja, existe um controlador ideal $C_d(z)$ que fará com que o sistema em malha fechada tenha a função de transferência definida por $M(z)$.

Na segunda suposição considera-se o que foi especificado no início, ou seja, o que foi definido em 1.

Suposição 2 *O controlador é parametrizado linearmente, ou seja,*

$$C(z, \rho) = \rho^T \bar{C}(z).$$

O que esta suposição diz é que é possível escrever a equação do controlador na forma de um vetor de parâmetros multiplicado pela classe de controladores. O vetor de parâmetros ρ é um vetor de números reais e a classe de controladores $\bar{C}(z)$ é um vetor formado por funções racionais em z. Adiciona-se ainda que as funções racionais de $\bar{C}(z)$ devem ser linearmente independentes sobre o campo dos números reais, ou seja, assume-se que a parametrização é mínima. Existem infinitas representações para a classe de controladores $\bar{C}(z)$ em que as funções racionais de $\bar{C}(z)$ são linearmente independentes, o que significa que, para cada representação de $\bar{C}(z)$ o controlador é univocamente determinado pelo respectivo vetor de parâmetros ρ.

Assim sendo, assumindo-se que o sistema não é afetado por ruído e que as duas suposições acima são verdadeiras pode-se estimar os parâmetros do controlador utilizando-se o método VRFT para o caso ideal, conforme descrito na sequência.

3.1.1 Descrição do método

O método VRFT consiste na execução de um experimento em laço aberto ou fechado, onde os dados de entrada $u(t)$ e saída $y(t)$ do processo são coletados. Em um experimento em laço aberto, conforme apresentado na Figura 3, um sinal de entrada $u(t)$ conhecido é inserido no processo e a sua respectiva saída é medida. Já em um experimento em laço fechado um sinal de referência $r(t)$ é aplicado ao sistema e são medidos os dados de entrada $u(t)$ e saída $y(t)$ da planta.

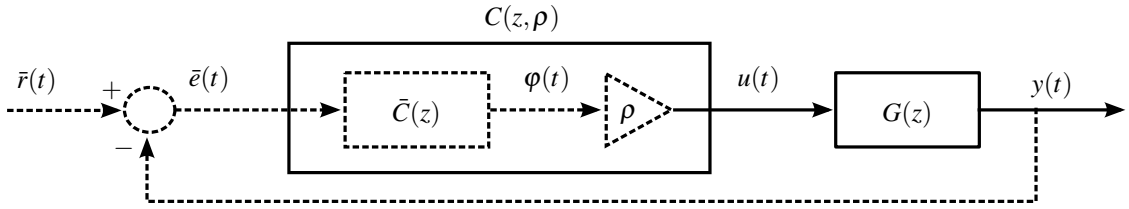


Figura 3: Experimento para o VRFT.

De posse dos sinais reais $u(t)$ e $y(t)$, realiza-se a parte virtual do método (que corresponde às linhas pontilhadas na Figura 3), onde, de posse da saída medida $y(t)$, e imaginando-se que o controlador ideal $C(z, \rho)$ está realmente inserido no processo, define-se a referência virtual $\bar{r}(t)$ como

$$\bar{r}(t) = M^{-1}(z) y(t),$$

onde $M(z)$ é o modelo de referência. Ou seja, encontra-se qual seria o sinal de referência necessário para gerar os sinais medidos caso o sistema já estivesse se comportando conforme desejado.

Em seguida, encontra-se o sinal de entrada do controlador $\bar{e}(t)$, também chamado de erro de seguimento de referência virtual, calculado como

$$\bar{e}(t) = \bar{r}(t) - y(t).$$

Assim, pode-se identificar o controlador ideal a partir dos dados de entrada e saída do mesmo, sendo estes $\bar{e}(t)$ e $u(t)$, respectivamente. O controlador ideal é aquele que quando inserido em sua entrada o sinal $\bar{e}(t)$ irá gerar como saída o sinal $u(t)$ (CAMPI; LECCHINI; SAVARESI, 2002), em outras palavras, quanto melhor o controlador mais próxima a saída do mesmo será do sinal $u(t)$. Uma vez que $\bar{e}(t)$ é a entrada do controlador ideal, $C(z,\rho)$ é um modelo para o controlador, e que a saída do controlador é predita por $C(z,\rho)\bar{e}(t)$, pode-se definir o $J^{\mathrm{VR}}(\rho)$, que diz o quão distante a entrada da planta está da entrada obtida, da seguinte forma

$$J^{\mathrm{VR}}(\rho) \triangleq \bar{E}\left[u(t) - C(z,\rho)\bar{e}(t)\right]^2 \tag{3}$$

Como, no caso ideal, o controlador é parametrizado linearmente (Suposição 2 é satisfeita) o critério $J^{\mathrm{VR}}(\rho)$ pode ser reescrito como:

$$\begin{aligned} J^{\mathrm{VR}}(\rho) &= \bar{E}\left[u(t) - \rho^{\mathrm{T}}\bar{C}(z)\bar{e}(t)\right]^2 \\ &= \bar{E}\left[u(t) - \rho^{\mathrm{T}}\varphi(t)\right]^2, \end{aligned} \tag{4}$$

onde, $\varphi(t)$ é o vetor regressor, ou seja,

$$\begin{aligned} \varphi(t) &= \bar{C}(z)\bar{e}(t) \\ &= \bar{C}(z)\frac{1-M(z)}{M(z)}y(t). \end{aligned} \tag{5}$$

O problema de encontrar o mínimo global de (4) pode ser encarado como um problema de mínimos quadrados (BAZANELLA; CAMPESTRINI; ECKHARD, 2012), para resolver este problema resolve-se a equação normal, ou seja, os parâmetros do controlador podem ser estimados da seguinte forma:

$$\hat{\rho} = \left[\sum_{t=1}^{n}\varphi(t)\varphi^{\mathrm{T}}(t)\right]^{-1}\sum_{t=1}^{n}\varphi(t)u(t), \tag{6}$$

onde n é o número de amostras coletadas no experimento.

No caso ideal, onde o sistema não é afetado por ruído, o valor estimado para $\hat{\rho}$ é o valor exato que fará com que o sistema em laço fechado tenha o comportamento desejado, definido pela função de transferência $M(z)$.

Porém, se os dados de entrada $u(t)$ e saída $y(t)$ do processo apresentarem ruído não negligenciável, o controlador obtido será polarizado, ou seja, o controlador encontrado não será o controlador ótimo. Nestes casos em que o ruído presente no sistema é mais pronunciado pode ser utilizado o método VRFT com variável instrumental (VI), porém esta abordagem não será detalhada neste trabalho. Uma explicação detalhada do método VRFT com VI pode ser encontrada em Bazanella, Campestrini e Eckhard (2012).

3.1.2 Equivalência das funções custo

Conforme mencionado anteriormente, sob certas condições o mínimo da função custo do critério de seguimento de referência, $J_y(\rho)$, é igual ao mínimo da função custo do método VRFT, $J^{\mathrm{VR}}(\rho)$.

A partir de (4), e uma vez que $u(t) = C_d(z)\bar{e}(t)$ é possível reescrever $J^{\mathrm{VR}}(\rho)$ da seguinte forma

$$\begin{aligned} J^{\mathrm{VR}}(\rho) &= \bar{E}\left[C_d(z)\bar{e}(t) - \rho^{\mathrm{T}}\bar{C}(z)\bar{e}(t)\right]^2 \\ &= \bar{E}\left[\rho_d^{\mathrm{T}}\bar{C}(z)\bar{e}(t) - \rho^{\mathrm{T}}\bar{C}(z)\bar{e}(t)\right]^2 \\ &= \bar{E}\left[(\rho_d - \rho)^{\mathrm{T}}\varphi(t)\right]^2. \end{aligned} \tag{7}$$

Agora substituindo-se (5) em (7), e lembrando-se que $y(t) = G(z)u(t)$, obtém-se:

$$J^{\mathrm{VR}}(\rho) = \bar{E}\left[(\rho_d - \rho)^{\mathrm{T}}\bar{C}(z)\frac{1 - M(z)}{M(z)}G(z)u(t)\right]^2. \tag{8}$$

Lembrando das relações de laço fechado do sistema

$$C_d(z) = \frac{M(z)}{G(z)\left(1 - M(z)\right)},$$

pode-se ainda reescrever (8) da seguinte forma

$$J^{\mathrm{VR}}(\rho) = \bar{E}\left[(\rho_d - \rho)^{\mathrm{T}}\bar{C}(z)\frac{1}{C_d(z)}u(t)\right]^2. \tag{9}$$

Aplicando-se o teorema de Parseval em (9) obtém-se:

$$J^{\mathrm{VR}}(\rho) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{1}{\left|C_d\left(e^{j\omega}\right)\right|^2}\left|(\rho - \rho_d)^{\mathrm{T}}\bar{C}(e^{j\omega})\right|^2\Phi_u(e^{j\omega})d\omega \tag{10}$$

onde, $\Phi_u(e^{j\omega})$ é o espectro do sinal $u(t)$.

A integral acima existe nas seguintes condições, ou $C_d(z)$ não possui zeros no círculo unitário ou estes zeros não correspondem a frequências no suporte de $u(t)$, além disso, tem-se que $J^{\mathrm{VR}}(\rho_d) = 0$, assim sendo, ρ_d é um mínimo global de $J^{\mathrm{VR}}(\rho)$.

É possível também reescrever (10) em sua forma quadrática,

$$J^{\mathrm{VR}}(\rho) = (\rho - \rho_d)^{\mathrm{T}}A^{\mathrm{VR}}(\rho - \rho_d), \tag{11}$$

onde,

$$A^{\mathrm{VR}} = \frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{1}{\left|C_d(e^{j\omega})\right|^2}\bar{C}(e^{j\omega})\bar{C}^*(e^{j\omega})\Phi_u(e^{j\omega})d\omega.$$

A matriz A^{VR} é uma matriz positiva semi-definida por construção. A partir de (11) pode-se observar que ρ_d é um mínimo global desta função e que este é o único mínimo global possível se a matriz A^{VR} for uma matriz positiva definida.

3.1.3 Riqueza do sinal de entrada

Para garantir que a matriz A^{VR} seja positiva definida também é preciso garantir que o sinal de entrada $u(t)$ seja suficientemente rico. Considerando-se, por exemplo, que a entrada do sistema possui um espectro finito e discreto formado por q componentes, ou seja,

$$\Phi_u(e^{j\omega}) = \sum_{k=1}^{q} \lambda_k \delta(\omega - \omega_k),$$

onde $\delta(.)$ é a função delta de Dirac e λ_k são números reais positivos, de forma que

$$A^{\mathrm{VR}} = \frac{1}{2\pi} \sum_{k=1}^{q} \frac{\lambda_k}{|C_d(e^{j\omega_k})|^2} \bar{C}(e^{j\omega_k}) \bar{C}^*(e^{j\omega_k}).$$

Cada uma das matrizes $\bar{C}(e^{j\omega_k})\bar{C}^*(e^{j\omega_k})$ é positiva semi-definida e a matriz A^{VR} é uma soma ponderada de q dessas matrizes. Como uma propriedade genérica, tem-se que o posto de uma soma como esta é o $\min(q, p)$.

Deste modo, o sinal de entrada $u(t)$ deve possuir pelo menos p componentes em frequência (ser suficientemente rico de ordem p — SRp), garantindo assim que a matriz A^{VR} seja positiva definida, e por conseguinte, garantindo que ρ_d seja o único mínimo global de $J^{\mathrm{VR}}(\rho)$.

3.2 Planta com zeros de fase não mínima

Nesta seção será apresentado como proceder caso a planta possua zeros de fase não mínima, considerando-se que as mesmas suposições para o caso ideal são satisfeitas, ou seja, o sistema não é afetado por ruído, o controlador ideal pertence à classe de controladores, e o controlador é parametrizado linearmente.

Se o processo possui zeros de fase não mínima, estes devem aparecer no modelo de referência $M(z)$, uma vez que os zeros do processo em laço aberto continuarão sendo zeros do sistema em laço fechado, a menos que sejam cancelados pelos polos do controlador. E para cancelar os zeros de fase não mínima os polos do controlador seriam instáveis, o que não é desejável. Neste caso duas abordagens podem ser utilizadas: quando a localização dos zeros de fase não mínima é conhecida, estes são incluídos no modelo de referência, caso contrário utiliza-se uma outra abordagem do VRFT desenvolvida para lidar com este caso, conhecida como método VRFT flexível, no qual os zeros de fase não mínima são identificados juntamente com os parâmetros do controlador — uma descrição detalhada desta abordagem é apresentada em (CAMPESTRINI; GEVERS; BAZANELLA, 2009), e (BAZANELLA; CAMPESTRINI; ECKHARD, 2012). Contudo, esta abordagem não será apresentada no presente trabalho, aqui será descrito como proceder no caso em que os zeros de fase não mínima são conhecidos.

Como o VRFT usa a inversa de $M(z)$ para estimar os parâmetros do controlador, estes zeros acabam tornando-se polos instáveis de $M^{-1}(z)$, resultando em um sinal de referência virtual instável.

Assim sendo, para lidar com processos de fase não mínima algumas alterações são realizadas de modo que os sinais utilizados pelo VRFT sejam filtrados através de filtros estáveis. Para tanto, inicialmente fatora-se $M(z)$ de modo a isolar os zeros de fase não mínima da seguinte maneira

$$M(z) = \frac{n_u(z) n_s(z)}{d(z)}$$

onde o termo,

$$n_u(z) = \prod_{k=1}^{m}(z - x_k)$$

possui somente os zeros de fase não mínima.

Assim, define-se o filtro estável que será utilizado para filtrar o sinal $u(t)$, sendo este um filtro passa-tudo ponderado com magnitude igual a 1 para todas as frequências, da seguinte forma

$$L_{ap}(z) \triangleq \frac{n_u(z)}{n_u^*(z)},$$

onde,

$$n_u^*(z) = \prod_{k=1}^{m}(x_k z - 1)$$

é o polinômio cujas raízes são o inverso dos zeros de fase não mínima, que são polos estáveis.

Assim, uma pequena modificação é realizada no critério $J^{\mathrm{VR}}(\rho)$ de modo que o sinal $\bar{r}(t)$ é obtido filtrando-se $y(t)$ por um filtro $M_a^{-1}(z)$, onde

$$M_a(z) = \frac{n_u^*(z) n_s(z)}{d(z)},$$

$n_s(z)$ é um polinômio contendo os zeros de fase mínima, e $d(z)$ é um polinômio contendo os polos de $M(z)$. Cabe salientar que a função de transferência de $M_a(z)$ possui a mesma magnitude que a função de transferência de $M(z)$, porém apenas com zeros de fase mínima.

Então, utilizando o filtro $L_{ap}(z)$ tem-se que:

$$J_a^{\mathrm{VR}}(\rho) = \bar{E}\left\{L_{ap}(z)\left[u(t) - \left(C(z,\rho)\frac{1 - M_a(z)}{M_a(z)}\right)y(t)\right]\right\}^2$$

cujo mínimo é o mesmo de $J^{\mathrm{VR}}(\rho)$, sendo esta a função custo a ser minimizada quando o processo possui zeros de fase não mínima.

Assim, de maneira resumida, de posse dos sinais $u(t)$ e $y(t)$, filtrados através dos filtros estáveis apresentados acima, o cálculo dos parâmetros do controlador é o mesmo apresentado anteriormente (equação 6), e reapresentado aqui da seguinte forma

$$\hat{\rho} = \left[\sum_{t=1}^{n} \varphi_{Lap}(t)\varphi_{Lap}^{\mathrm{T}}(t)\right]^{-1} \sum_{t=1}^{n} \varphi_{Lap}(t) u_{Lap}(t), \tag{12}$$

onde $u_{Lap}(t)$ é o sinal $u(t)$ filtrado através de $L_{ap}(z)$, e $\varphi_{Lap}(t)$ é o vetor regressor obtido utilizando-se ou $M_a^{-1}(z)$ ou o sinal de saída $y(t)$ filtrado através de $L_{ap}(z)$.

Outra possibilidade é multiplicar o critério $J^{\mathrm{VR}}(\rho)$ por $M(z)$, assim (3) se torna

$$J_{a'}^{\mathrm{VR}}(\rho) \triangleq \bar{E}\left[M(z)u(t) - \left(C(z,\rho)\left(1 - M(z)\right)\right)y(t)\right]^2,$$

cujo mínimo é igual ao mínimo do critério do VRFT, porém, não existe o inconveniente de filtrar os sinais através de filtros instáveis. Ou seja, o processo consiste em utilizar os sinais $u(t)$ e $y(t)$ filtrados através de $M(z)$, gerando $u_M(t)$ e $y_M(t)$. Deste modo, a referência virtual torna-se simplesmente $\bar{r}_M(t) = y(t)$. Em seguida, o erro virtual é obtido através de $\bar{e}_M(t) = r_M(t) - y(t)$. O vetor regressor é dado por $\varphi_M(t) = \bar{C}(z)\bar{e}_M(t)$ e os parâmetros podem ser finalmente obtidos por

$$\hat{\rho} = \left[\sum_{t=1}^{n} \varphi_M(t)\varphi_M^{\mathrm{T}}(t)\right]^{-1} \sum_{t=1}^{n} \varphi_M(t) u_M(t).$$

3.3 Caso descasado

Esta seção aborda o caso em que a suposição 1 não é satisfeita, ou seja, o controlador ideal não pertence à classe de controladores. Porém, as outras duas suposições se mantém, o sistema não é afetado por ruído, e o controlador é parametrizado linearmente.

Quando o controlador possui ordem cheia o caso ideal funciona corretamente. No caso descasado o controlador não possui ordem cheia, ou seja, o controlador ideal não pertence à classe de controladores. Acontece então que o mínimo de $J^{\mathrm{VR}}(\rho)$ pode ser bem diferente do mínimo de $J_y(\rho)$, porém este problema pode ser mitigado filtrando-se de modo apropriado os dados a serem utilizados para o cálculo dos parâmetros do controlador (BAZANELLA; CAMPESTRINI; ECKHARD, 2012).

De modo a encontrar o filtro $L(e^{j\omega})$ que irá solucionar o problema do caso descasado, primeiramente serão comparados os dois critérios $J_y(\rho)$ e $J^{\mathrm{VR}}(\rho)$ através de suas expressões no domínio da frequência.

De (2) e com algum esforço pode-se reescrever $J_y(\rho)$ como

$$\begin{aligned} J_y(\rho) = \frac{1}{2\pi}\int_{-\pi}^{\pi} & |G(e^{j\omega})|^2 |S(e^{j\omega},\rho)|^2 |S_d(e^{j\omega})|^2 \\ & \times |C(e^{j\omega},\rho) - C_d(e^{j\omega})|^2 \Phi_r(e^{j\omega}) d\omega \end{aligned} \tag{13}$$

onde, $S(e^{j\omega},\rho)$ é a função de sensibilidade do sistema, e $S_d(e^{j\omega})$ é a função de sensibilidade desejada, isto é, a função de sensibilidade que resulta de adicionar o controlador ideal $C_d(z,\rho)$ ao processo.

De modo a obter uma expressão para $J^{\mathrm{VR}}(\rho)$, primeiramente, filtra-se os sinais $u(t)$ e $\bar{e}(t)$ através de um filtro definido como $L(z)$, obtendo-se a seguinte expressão para a função custo em questão

$$\begin{aligned} J^{\mathrm{VR}}(\rho) &= \bar{E}\left[L(z)\left(u(t) - C(z,\rho)\bar{e}(t)\right)\right]^2 \\ &= \bar{E}\left[L(z)\left(u(t) - \left(C(z,\rho)\frac{1-M(z)}{M(z)}\right)y(t)\right)\right]^2. \end{aligned} \tag{14}$$

Novamente, com algum esforço a partir de (14) pode-se reescrever $J^{\mathrm{VR}}(\rho)$ como

$$\begin{aligned} J^{\mathrm{VR}}(\rho) = \frac{1}{2\pi}\int_{-\pi}^{\pi} & |L(e^{j\omega})|^2 \frac{|G(e^{j\omega})|^2 |S_d(e^{j\omega})|^2}{|M(e^{j\omega})|^2} \\ & \times |C_d(e^{j\omega}) - C(e^{j\omega},\rho)|^2 \Phi_u(e^{j\omega}) d\omega \end{aligned} \tag{15}$$

Uma explicação detalhada de como encontrar estas expressões pode ser encontrada em Bazanella, Campestrini e Eckhard (2012).

É de interesse tornar as duas funções custo $J_y(\rho)$ e $J^{\mathrm{VR}}(\rho)$ iguais, fazendo com que o mínimo global destas duas funções coincida. Para tanto, o filtro $L(e^{j\omega})$ deve respeitar a seguinte equação

$$|L(e^{j\omega})|^2 = |M(e^{j\omega})|^2 |S(e^{j\omega},\rho)|^2 \frac{\Phi_r(e^{j\omega})}{\Phi_u(e^{j\omega})}, \quad \forall \omega \in [-\pi;\pi]. \tag{16}$$

Contudo, $S(e^{j\omega},\rho)$ não é conhecida, uma vez que não se conhece a função de transferência do sistema $G(z)$, assim uma estimativa para $S(e^{j\omega},\rho)$ deve ser encontrada. A estimativa utilizada pelo método VRFT é dada por

$$|S(e^{j\omega},\rho)|^2 \approx |S_d(e^{j\omega})|^2 = |1-M(e^{j\omega})|^2,$$

que provavelmente não é a única aproximação possível. Porém, esta aproximação é bastante razoável pois, espera-se, que as duas funções de sensibilidade $S(e^{j\omega},\rho)$ e $S_d(e^{j\omega})$ sejam bem próximas uma da outra ao redor do mínimo global.

Com base nisso, a equação para o filtro $L(e^{j\omega})$ apresentada em (16) pode ser reescrita como

$$|L(e^{j\omega})|^2 = |M(e^{j\omega})|^2|1-M(e^{j\omega})|^2\frac{\Phi_r(e^{j\omega})}{\Phi_u(e^{j\omega})}, \quad \forall\omega\in[-\pi;\pi]. \tag{17}$$

Quando o sinal de entrada $u(t)$ é selecionado pelo projetista $\Phi_u(e^{j\omega})$ é conhecido, caso contrário o mesmo deve ser estimado (CAMPI; LECCHINI; SAVARESI, 2002). Quando é possível escolher que o sinal de entrada de um experimento em laço aberto seja o mesmo sinal que é geralmente aplicado ao processo como sinal referência tem-se que os espectros $\Phi_r(e^{j\omega})$ e $\Phi_u(e^{j\omega})$ são iguais, o que torna

$$\frac{\Phi_r(e^{j\omega})}{\Phi_u(e^{j\omega})} = 1,$$

simplificando o filtro $L(e^{j\omega})$ para:

$$L(e^{j\omega}) = M(e^{j\omega})\big(1-M(e^{j\omega})\big). \tag{18}$$

Assim, de posse do filtro $L(e^{j\omega})$ calculado em (17) o valor do vetor de parâmetros ρ é dado por

$$\rho_* = \bar{E}\left[\varphi_L(t)\varphi_L^{\mathrm{T}}(t)\right]^{-1}\bar{E}\left[\varphi_L(t)u_L(t)\right],$$

onde, $\varphi_L(t)$ e $u_L(t)$ são, respectivamente, o vetor $\varphi(t)$ e o sinal $u(t)$ filtrados através do filtro $L(e^{j\omega})$.

Neste caso, $\hat{\rho}$ é igual ao valor de ρ_* e os parâmetros do controlador podem ser estimados através da solução da seguinte equação:

$$\hat{\rho} = \left[\sum_{t=1}^{n}\varphi_L(t)\varphi_L^{\mathrm{T}}(t)\right]^{-1}\sum_{t=1}^{n}\varphi_L(t)u_L(t). \tag{19}$$

3.4 VRFT recursivo

A abordagem aqui apresentada é baseada no fato de que o problema de mínimos quadrados apresentado anteriormente pode ser resolvido de forma recursiva. Para tanto, o método de mínimos quadrados apresentado em (SÖDERSTRÖM; STOICA, 1989) e (AGUIRRE, 2004) foi utilizado com algumas alterações.

Assim, a partir de (6) e considerando-se somente as n primeiras amostras, pode-se

definir a seguinte matriz e a sua inversa:

$$P_n \triangleq \left[\sum_{t=1}^{n} \varphi(t)\varphi^{\mathrm{T}}(t)\right]^{-1}$$

$$\begin{aligned} P_n^{-1} &= \varphi(n)\varphi^{\mathrm{T}}(n) + \sum_{t=1}^{n-1} \varphi(t)\varphi^{\mathrm{T}}(t) \\ &= \varphi(n)\varphi^{\mathrm{T}}(n) + P_{n-1}^{-1}, \end{aligned}$$

e reescrever (6) até a n-ésima amostra como

$$\begin{aligned} \hat{\rho}_n &= P_n \sum_{t=1}^{n} \varphi(t)u(t) \\ &= P_n \left(\varphi(n)u(n) + \sum_{t=1}^{n-1} \varphi(t)u(t)\right), \end{aligned} \tag{20}$$

sendo que $\hat{\rho}_n$ é o vetor de parâmetros até a amostra n.

Agora, reescrevendo-se (6) para o instante $n-1$ obtém-se:

$$\hat{\rho}_{n-1} = \left[\sum_{t=1}^{n-1} \varphi(t)\varphi^{\mathrm{T}}(t)\right]^{-1} \sum_{t=1}^{n-1} \varphi(t)u(t),$$

que pode ser reescrita como

$$\begin{aligned} \sum_{t=1}^{n-1} \varphi(t)\varphi^{\mathrm{T}}(t)\hat{\rho}_{n-1} &= \sum_{t=1}^{n-1} \varphi(t)u(t) \\ P_{n-1}^{-1}\hat{\rho}_{n-1} &= \sum_{t=1}^{n-1} \varphi(t)u(t). \end{aligned} \tag{21}$$

Substituindo-se (21) em (20) obtém-se

$$\hat{\rho}_n = P_n \left[P_{n-1}^{-1}\hat{\rho}_{n-1} + \varphi(n)u(n)\right].$$

Ainda é possível reescrever a equação anterior, lembrando que

$$P_{n-1}^{-1} = P_n^{-1} - \varphi(n)\varphi^{\mathrm{T}}(n),$$

chegando-se ao seguinte resultado:

$$\begin{aligned} \hat{\rho}_n &= P_n\left[\left(P_n^{-1} - \varphi(n)\varphi^{\mathrm{T}}(n)\right)\hat{\rho}_{n-1} + \varphi(n)u(n)\right] \\ &= P_n\left[P_n^{-1}\hat{\rho}_{n-1} - \varphi(n)\varphi^{\mathrm{T}}(n)\hat{\rho}_{n-1} + \varphi(n)u(n)\right] \\ &= \hat{\rho}_{n-1} - P_n\varphi(n)\varphi^{\mathrm{T}}(n)\hat{\rho}_{n-1} + P_n\varphi(n)u(n) \\ &= \hat{\rho}_{n-1} + P_n\varphi(n)\left(u(n) - \varphi^{\mathrm{T}}(n)\hat{\rho}_{n-1}\right) \\ &= \hat{\rho}_{n-1} + K_n\eta(n), \end{aligned}$$

onde, $K_n = P_n\varphi(n)$ é uma matriz de ganhos e $\eta(n)$ é a inovação no instante n. Aqui uma questão interessante e importante é evitar a inversão de matriz em cada iteração envolvida no cálculo de P_n, uma vez que esta é uma operação mais custosa. Para tanto, utiliza-se o lema de inversão de matriz:

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B\left[C^{-1} + DA^{-1}B\right]^{-1}DA^{-1}. \tag{22}$$

Aplicando-se então este lema ao problema em questão e considerando-se que $A = P_{n-1}^{-1}$, $B = \varphi(n)$, $C = 1$ e $D = \varphi^{\mathrm{T}}(n)$ obtém-se:

$$P_n = P_{n-1} - P_{n-1}\varphi(n)\left[1 + \varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)\right]^{-1}\varphi^{\mathrm{T}}(n)P_{n-1}. \tag{23}$$

Finalmente, lembrando que $K_n = P_n\varphi(n)$, substituindo-se (23) nesta equação, e notando-se que

$$\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n) + 1$$

é um escalar, é possível reescrever K_n da seguinte forma

$$\begin{aligned}
K_n &= P_n\varphi(n) \\
&= \left(P_{n-1} - P_{n-1}\varphi(n)\left[1 + \varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)\right]^{-1}\varphi^{\mathrm{T}}(n)P_{n-1}\right)\varphi(n) \\
&= P_{n-1}\varphi(n) - P_{n-1}\varphi(n)\left[1 + \varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)\right]^{-1}\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n) \\
&= P_{n-1}\varphi(n) - \frac{P_{n-1}\varphi(n)\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)}{\left[\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n) + 1\right]} \\
&= \frac{P_{n-1}\varphi(n)}{\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n) + 1}
\end{aligned}$$

Assim, os parâmetros do controlador são estimados através do método de mínimos quadrados recursivo por estas três equações:

$$K_n = \frac{P_{n-1}\varphi(n)}{\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n) + 1} \tag{24}$$

$$P_n = P_{n-1} - K_n\varphi^{\mathrm{T}}(n)P_{n-1} \tag{25}$$

$$\hat{\rho}_n = \hat{\rho}_{n-1} + K_n\left(u(n) - \varphi^{\mathrm{T}}(n)\hat{\rho}_{n-1}\right). \tag{26}$$

As equações recursivas acima foram derivadas a partir da equação (6) do caso ideal, contudo é possível, através de algumas modificações nas equações (24), (25) e (26), adaptar esta abordagem para atender o caso em que a planta possui zeros de fase não mínima e o caso em que o controlador ideal não pertence à classe de controladores. Para tanto, é preciso filtrar os sinais $\varphi(t)$ e $u(t)$ através dos filtros mostrados nas seções 3.2 e 3.3.

Entretanto, para estimar os parâmetros do controlador é preciso inicializar o vetor $\hat{\rho}_n$ e a matriz P_n com algum valor. É importante notar que o valor escolhido irá influenciar a capacidade de convergência do algoritmo. Assim sendo, um valor muito distante do valor ideal pode fazer com que a convergência dos parâmetros seja muito lenta. Uma boa opção, e talvez a mais óbvia, é utilizar a abordagem não recursiva do método VRFT para inicializar o vetor $\hat{\rho}_n$ e a matriz P_n^{-1} nas primeiras amostras, até que P_n^{-1} seja inversível. Utilizando esta abordagem inicial e obtendo P_n a partir de P_n^{-1}, garante-se que a abordagem

recursiva irá convergir exatamente para o mesmo vetor de parâmetros que seria obtido se fosse empregada a abordagem não recursiva em todas as amostras.

A abordagem recursiva é particularmente conveniente para ser utilizada em uma aplicação embarcada, uma vez que dispensa o armazenamento de uma grande quantidade de dados. Outro quesito interessante, é que nesta abordagem o projetista não precisa definir *a priori* a quantidade de dados a ser coletado no experimento. Além disso, a quantidade de dados influencia na resposta obtida para os parâmetros do controlador e varia de processo para processo.

Para permitir que o algoritmo decida quando parar o experimento é preciso definir, antes de iniciar o experimento, um critério de parada. Assim, quando o critério é atingido o algoritmo é capaz de parar o experimento automaticamente.

Vários critérios podem ser considerados, dentre eles são listados os critérios abaixo:

a) Variação mínima absoluta ou relativa de cada parâmetro, ou seja, o algoritmo deve parar caso

$$|\hat{\rho}_i(n) - \hat{\rho}_i(n-1)| < \alpha_{\text{abs}}, \qquad \forall i \in \{1,2,\ldots,p\}$$

ou

$$\left|\frac{\hat{\rho}_i(n) - \hat{\rho}_i(n-1)}{\hat{\rho}_i(n-1)}\right| < \alpha_{\text{rel}}, \qquad \forall i \in \{1,2,\ldots,p\}$$

b) Uma variação mínima absoluta ou relativa na norma do vetor de parâmetros

$$|\hat{\rho}(n) - \hat{\rho}(n-1)| < \alpha_{\text{abs}}$$

ou

$$\frac{|\hat{\rho}(n) - \hat{\rho}(n-1)|}{|\hat{\rho}(n-1)|} < \alpha_{\text{rel}}$$

c) O valor mínimo para a média dos quadrados da diferença entre $\rho^{\mathrm{T}}\varphi(t)$ e $u(t)$

$$\frac{1}{n}\sum_{t=1}^{n}[\hat{\rho}(t)\varphi(t) - u(t)]^2 < \alpha_{mqd}$$

d) Duração máxima do experimento, $n > \alpha_{nmax}$

e) Uma combinação destes.

3.5 Fator de esquecimento

Nesta seção a abordagem do método de mínimos quadrados recursivo com fator de esquecimento é apresentada, para tanto, o método apresentado em (SÖDERSTRÖM; STOICA, 1989) e (AGUIRRE, 2004) foi utilizado com algumas adaptações.

Na abordagem recursiva mostrada na seção anterior, todas as amostras contribuem igualmente no cálculo dos parâmetros do controlador. Segundo Aguirre (2004), esta abordagem pode geralmente ser empregada em sistemas que não variam no tempo. Já em sistemas que variam no tempo é interessante que as amostras sejam ponderadas de forma diferente, de modo que as amostras mais recentes influenciem mais na estimativa dos parâmetros do controlador, uma vez que, para sistemas variantes no tempo as amostras mais recentes são as que possuem informação mais atualizada do sistema. Além disso,

a variação dos parâmetros ao longo do tempo pode se dar por várias razões, como por exemplo: operação do sistema em regiões diferentes em que um único modelo linear não seria suficiente, envelhecimento dos componentes do sistema, ocorrência de falhas, entre outros.

Assim sendo, a partir de (6), define-se o vetor de parâmetros estimados $\hat{\rho}_n$ como segue

$$\begin{aligned}\hat{\rho}_n &= \left[\sum_{t=1}^{n} w_t(n)\varphi(t)\varphi^{\mathrm{T}}(t)\right]^{-1} \sum_{t=1}^{n} w_t(n)\varphi(t)u(t) \\ &= P_n F_n, \end{aligned} \tag{27}$$

onde w é uma sequência de pesos. No caso recursivo com fator de esquecimento, a última amostra deve possuir peso máximo e as amostras mais antigas um peso inferior, ou seja, quanto mais antiga a amostra menor será sua contribuição na estimativa dos parâmetros do controlador, possibilitando assim o que foi dito anteriormente, que as amostras mais recentes tenham uma contribuição maior na estimativa dos parâmetros do controlador.

De modo a ponderar as amostras como foi mencionado anteriormente, assume-se as seguintes restrições:

$$\begin{cases} w_n(n) = 1 \\ w_t(n) = \lambda w_t(n-1), \quad t < n, \end{cases} \tag{28}$$

ou seja, o peso da amostra t na iteração atual, $w_t(n)$, é igual ao peso da mesma amostra na iteração anterior multiplicado por um fator de esquecimento λ menor que 1, exceto pela amostra mais recente que possui peso unitário. Usualmente, são utilizados valores de λ entre 0,95 e 0,99, e quanto menor o valor de λ mais rápido as amostras mais antigas serão "esquecidas" (SÖDERSTRÖM; STOICA, 1989). Deste modo, pode-se escolher o valor de λ levando-se em consideração o quão rápido deseja-se que as contribuições mais antigas sejam suprimidas do cálculo dos parâmetros do controlador.

Assim, a partir de (27) pode-se escrever P_n^{-1} como:

$$P_n^{-1} = \sum_{t=1}^{n-1} w_t(n)\varphi(t)\varphi^{\mathrm{T}}(t) + w_n(n)\varphi(n)\varphi^{\mathrm{T}}(n). \tag{29}$$

Agora, lembrando das restrições de pesos definidas em (28) pode-se ainda reescrever P_n^{-1} da seguinte forma:

$$\begin{aligned} P_n^{-1} &= \sum_{t=1}^{n-1} \lambda w_t(n-1)\varphi(t)\varphi^{\mathrm{T}}(t) + \varphi(n)\varphi^{\mathrm{T}}(n) \\ &= \lambda P_{n-1}^{-1} + \varphi(n)\varphi^{\mathrm{T}}(n). \end{aligned} \tag{30}$$

A partir de (27) e utilizando-se novamente as restrições definidas em (28), pode-se escrever F_n como

$$\begin{aligned} F_n &= \sum_{t=1}^{n} w_t(n)\varphi(t)u(t) \\ &= \sum_{t=1}^{n-1} w_t(n)\varphi(t)u(t) + w_n(n)\varphi(n)u(n) \\ &= \sum_{t=1}^{n-1} \lambda w_t(n-1)\varphi(t)u(t) + \varphi(n)u(n). \end{aligned} \tag{31}$$

Sabendo-se que

$$F_{n-1} = \sum_{t=1}^{n-1} w_t(n-1)\varphi(t)u(t),$$

pode-se ainda reescrever (31) como

$$F_n = \lambda F_{n-1} + \varphi(n)u(n). \tag{32}$$

Substituindo-se (32) em (27) obtém-se:

$$\hat{\rho}_n = P_n \left[\lambda F_{n-1} + \varphi(n)u(n)\right].$$

Agora, sabendo-se que

$$F_{n-1} = P_{n-1}^{-1}\hat{\rho}_{n-1}$$

é possível reescrever $\hat{\rho}_n$ como segue

$$\hat{\rho}_n = P_n \left[\lambda P_{n-1}^{-1}\hat{\rho}_{n-1} + \varphi(n)u(n)\right]. \tag{33}$$

Agora, lembrando-se que (30) pode ser reescrita como

$$P_{n-1}^{-1} = \frac{P_n^{-1} - \varphi(t)\varphi^{\mathrm{T}}(t)}{\lambda},$$

substituindo-se esta equação em (33) obtém-se:

$$\begin{aligned}
\hat{\rho}_n &= P_n \left\{ \lambda \left[\frac{P_n^{-1} - \varphi(t)\varphi^{\mathrm{T}}(t)}{\lambda} \right] \hat{\rho}_{n-1} + \varphi(n)u(n) \right\} \\
&= \hat{\rho}_{n-1} + P_n\varphi(n)\left[u(n) - \varphi^{\mathrm{T}}(n)\hat{\rho}_{n-1}\right] \\
&= \hat{\rho}_{n-1} + K_n\left[u(n) - \varphi^{\mathrm{T}}(n)\hat{\rho}_{n-1}\right],
\end{aligned} \tag{34}$$

onde, $K_n = P_n\varphi(n)$ é o ganho de adaptação do vetor estimado. Aqui novamente deseja-se evitar a inversão da matriz P_n^{-1} a cada iteração do algoritmo, para tanto, também será utilizado o lema da inversão apresentado em (22) na equação em questão (30), onde $A = \lambda P_{n-1}^{-1}$, $B = \varphi(n)$, $C = 1$ e $D = \varphi^{\mathrm{T}}(n)$, obtendo-se:

$$\begin{aligned}
P_n &= \frac{P_{n-1}}{\lambda} - \frac{P_{n-1}}{\lambda}\varphi(n)\left[1 + \frac{\varphi^{\mathrm{T}}(t)P_{n-1}\varphi(n)}{\lambda}\right]^{-1}\varphi^{\mathrm{T}}(n)\frac{P_{n-1}}{\lambda} \\
&= \frac{P_{n-1}}{\lambda} - \frac{P_{n-1}}{\lambda}\varphi(n)\left[\frac{\lambda + \varphi^{\mathrm{T}}(t)P_{n-1}\varphi(n)}{\lambda}\right]^{-1}\varphi^{\mathrm{T}}(n)\frac{P_{n-1}}{\lambda} \\
&= \frac{P_{n-1}}{\lambda} - \frac{P_{n-1}}{\lambda}\varphi(n)\left[\lambda + \varphi^{\mathrm{T}}(t)P_{n-1}\varphi(n)\right]^{-1}\lambda\varphi^{\mathrm{T}}(n)\frac{P_{n-1}}{\lambda} \\
&= \frac{P_{n-1}}{\lambda} - \frac{P_{n-1}\varphi(n)\varphi^{\mathrm{T}}(n)P_{n-1}}{\lambda\left[\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n) + \lambda\right]}
\end{aligned} \tag{35}$$

Substituindo-se (35) em $K_n = P_n\varphi(n)$ obtém-se:

$$\begin{aligned} K_n &= P_n\varphi(n) \\ &= \left(\frac{P_{n-1}}{\lambda} - \frac{P_{n-1}\varphi(n)\varphi^{\mathrm{T}}(n)P_{n-1}}{\lambda\left[\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)+\lambda\right]}\right)\varphi(n) \\ &= \frac{P_{n-1}\varphi(n)\left[\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)+\lambda\right] - P_{n-1}\varphi(n)\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)}{\lambda\left[\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)+\lambda\right]} \\ &= \frac{P_{n-1}\varphi(n)}{\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)+\lambda}. \end{aligned} \tag{36}$$

Finalmente, o estimador recursivo de mínimos quadrados com fator de esquecimento λ pode ser escrito da seguinte forma:

$$\begin{aligned} K_n &= \frac{P_{n-1}\varphi(n)}{\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)+\lambda} \\ P_n &= \frac{1}{\lambda}\left(P_{n-1} - \frac{P_{n-1}\varphi(n)\varphi^{\mathrm{T}}(n)P_{n-1}}{\varphi^{\mathrm{T}}(n)P_{n-1}\varphi(n)+\lambda}\right) \\ \hat{\rho}_n &= \hat{\rho}_{n-1} + K_n\left[u(n) - \varphi^{\mathrm{T}}(n)\hat{\rho}_{n-1}\right]. \end{aligned}$$

Através destas três equações é possível estimar os parâmetros do controlador com o fator de esquecimento, que como visto anteriormente, irá considerar que as amostras mais recentes contribuirão mais para a estimativa dos parâmetros do controlador do que as amostras mais antigas.

Cabe salientar que o algoritmo recursivo mostrado na seção 3.4 é um caso particular do algoritmo mostrado acima, e que para um fator de esquecimento unitário, isto é, $\lambda = 1$, os dois algoritmos são iguais. Além disso, as mesmas alterações podem ser realizadas para lidar com o caso em que a planta possui zeros de fase não mínima e o caso descasado, ou seja, os mesmos filtros podem ser aplicados.

Da mesma forma que a abordagem apresentada na seção 3.4, no VRFT recursivo com fator de esquecimento também é preciso inicializar o vetor $\hat{\rho}_n$ e a matriz P_n. E como mencionado anteriormente, o valor escolhido para inicializar o vetor irá influenciar a capacidade de convergência do algoritmo. Aqui, novamente, utiliza-se o método VRFT original para inicializar $\hat{\rho}_n$ e P_n. Além disso, os mesmos critérios de parada mencionados anteriormente podem ser utilizados neste caso.

4 PROTOCOLO DE COMUNICAÇÃO

Para atingir os objetivos do presente trabalho foi desenvolvido um protocolo simples de comunicação, porém, com os requisitos necessários para permitir a comunicação entre dois dispositivos através da tecnologia Bluetooth. O protocolo desenvolvido utiliza o modelo cliente-servidor e foi implementado em duas camadas, a camada de aplicação e a camada de enlace de dados.

4.1 Bluetooth

O protocolo Bluetooth é um protocolo de comunicação sem fio, de curto alcance, projetado para ter como características: robustez, baixo consumo e baixo custo (BLUETOOTH, 2014). As especificações do protocolo definem uma interface de comunicação sem fio de rádio frequência (RF), bem como os protocolos de comunicação e os perfis que compõem a tecnologia Bluetooth (BHAGWAT, 2001).

4.1.1 Características

O Bluetooth é um protocolo padronizado para troca de dados entre dispositivos a curtas distâncias e opera na banda industrial, científica e médica (ISM – *industrial, scientific, and medical*) de 2,4 GHz. A banda é dividida em 79 canais de 1 MHz cada, e na maioria dos dispositivos, o alcance da comunicação é de aproximadamente 10 m.

O Bluetooth utiliza o método de transmissão de sinais de rádio FHSS (*Frequency-Hopping Spread Spectrum*), conhecido como espalhamento espectral por saltos ou sequência de saltos em frequência. No FHSS é utilizado um gerador de números pseudo-aleatórios para produzir as frequências de saltos. Uma vez que ambos os dispositivos (transmissor e receptor) possuam a mesma semente de geração de sequência, e permaneçam sincronizados, ambos conseguirão saltar para a mesma frequência simultaneamente (TANENBAUM, 2003).

Os dados a serem transmitidos são divididos em pacotes que são enviados através dos 79 canais nos quais a banda é dividida. Geralmente, ocorre um salto a cada 625 µs (uma fatia de tempo) resultando em 1600 saltos por segundo. Os períodos de tempo em que os dispositivos permanecem em cada frequência são chamados de *frames*.

4.1.2 Arquitetura da pilha de protocolos do Bluetooth

A pilha de protocolos do Bluetooth é constituída por vários protocolos agrupados em camadas. A arquitetura básica da tecnologia Bluetooth é apresentada através da Figura 4, onde podem ser visualizadas suas principais camadas. As camadas pertinentes a este trabalho são: rádio Bluetooth, banda base, L2CAP e RFCOMM que serão descritas na

sequência.

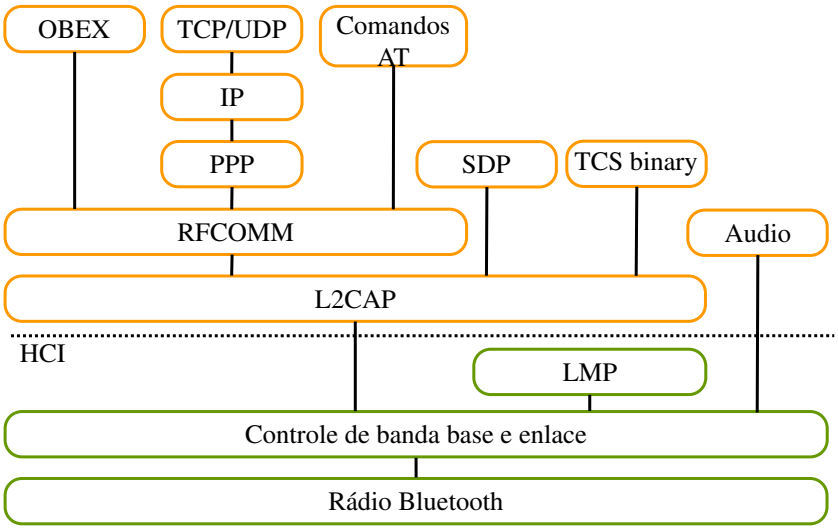


Figura 4: Arquitetura da pilha Bluetooth.

A camada inferior do protocolo é a camada de rádio do Bluetooth. Esta camada é implementada em hardware e é responsável pela modulação, e pelo envio e recebimento dos dados, que são sequências de bits. A segunda camada da pilha é a camada de controle de banda base e enlace, cuja função é processar os dados que são recebidos e enviados pelo rádio Bluetooth. Além disso, é responsável por controlar a correção de erros, enlaces (*links*), pacotes, controle de fluxo e canais.

Um enlace é um canal lógico utilizado para transmissão de *frames*. Dois tipos de enlaces podem ser estabelecidos: ACL (*Asynchronous ConnectionLess*) e SCO (*Synchronous Connection-Oriented*).

Enlaces ACL são utilizados para transmissão de dados sem determinismo. Neste tipo de enlace não há garantias na entrega dos pacotes, contudo, no caso de perda de pacotes os mesmos podem ser retransmitidos. Para determinar se há erro nas mensagens é utilizada a técnica de detecção de erros CRC (*Cyclic Redundacy Check*). Nesta técnica o transmissor adiciona ao *frame* a ser enviado uma sequência extra de *n* bits (FCS, *Frame Check Sequence*) calculada para conter informações redundantes sobre o mesmo. Esta sequência extra é utilizada no receptor para detectar erros no *frame* recebido, através de um novo cálculo. No ACL é utilizado o gerador de CRC CCITT que gera um FCS com comprimento de 16 bits (AGRAWAL; ZENG, 2011).

Um enlace do tipo SCO é geralmente utilizado para transmissão de áudio em tempo-real. Neste caso os dispositivos combinam quais fatias de tempo serão reservadas para a comunicação e não há retransmissão dos pacotes não recebidos. Para lidar com os erros que podem ocorrer nos canais é utilizada a técnica de processamento de sinal FEC (*Forward Error Correction*). Nesta técnica são adicionados bits redundantes de dados de modo que o sinal possa ser recuperado aumentando a confiabilidade dos dados transmitidos.

A comunicação entre as três camadas inferiores e as camadas mais acima da pilha é realizada pelo HCI (*Host Controller Interface*). Este protocolo pode ser visto como um protocolo de transporte entre o rádio Bluetooth e as camadas mais altas da pilha (L2CAP, RFCOMM, etc.). Em outras palavras o HCI é o responsável por definir como o dispositivo *host* se comunica com o hardware do adaptador Bluetooth.

A camada L2CAP (*Logical Link Control and Adaptation Protocol*) foi desenvolvida

para atender à necessidade de transmitir dados maiores entre as camadas superiores e a camada de banda base. Na especificação da L2CAP considera-se que os pacotes que esta camada recebe estão corretos. Por conta disso, nesta camada não são realizados testes de integridade, isto é, nenhum tipo de verificação é realizada nos pacotes. Além disso, também considera-se que os pacotes são entregues na ordem em que devem ser enviados. Deste modo, cada pacote é reenviado até que uma mensagem de confirmação seja recebida ou que ocorra perda de conexão (BHAGWAT, 2001).

O protocolo RFCOMM (*Radio Frequence Communications*) é um protocolo de transporte simples e confiável. Este protocolo foi desenvolvido para emular portas seriais sobre o protocolo L2CAP, possibilitando que aplicações escritas para operar utilizando este tipo de porta possam utilizar o Bluetooth sem grandes modificações (BENSKY, 2004). O RFCOMM emula portas seriais do padrão RS-232, que suporta tanto comunicação síncrona quanto assíncrona. Este padrão também emula comunicação *full-duplex*, ou seja, o envio e recebimento de dados pode ocorrer simultaneamente.

4.1.3 Topologia

Em uma rede Bluetooth a topologia utilizada é a de uma rede do tipo estrela. O dispositivo central opera como dispositivo mestre e os demais dispositivos operam como escravos. Desta forma, apenas um dispositivo pode operar como mestre e toda a comunicação ocorre entre o mestre e um ou mais escravos, portanto, não há comunicação direta entre os escravos. A sincronização dos dispositivos escravos é realizada através da sequência de salto e do *clock* do mestre (VERMA; SINGH; KAUR, 2015).

Uma *piconet* é uma rede Bluetooth básica que suporta até 8 dispositivos conectados ao mesmo tempo. Nesta rede, um dos dispositivos é o mestre e os demais são dispositivos escravos. Na Figura 5a e Figura 5b podem ser visualizadas duas estruturas diferentes de uma *piconet*.

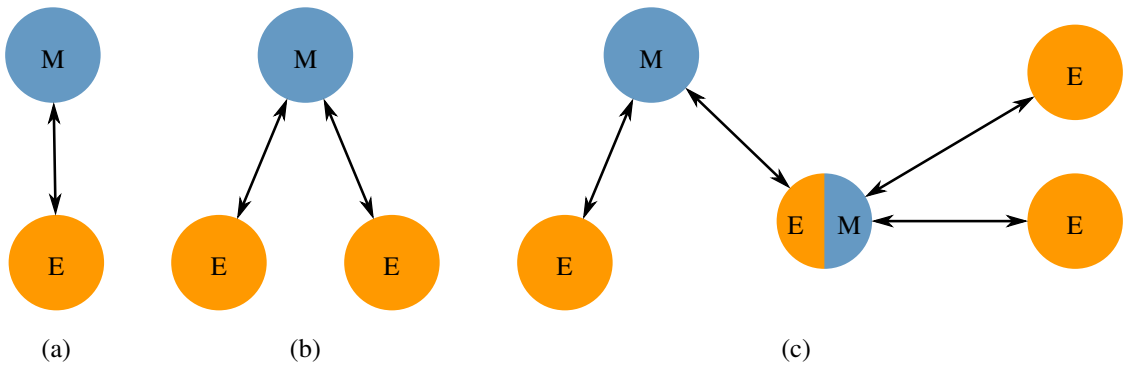


Figura 5: Topologias de uma rede Bluetooth: a) *piconet* com um único escravo, b) *piconet* com vários escravos, e c) *scatternet*.

Uma *scatternet* é formada pela conexão entre duas ao mais *piconets*, conforme pode ser visto na Figura 5c. Um dispositivo pode ser escravo em ambas *piconets* mas só pode ser mestre em uma única *piconet* (VERMA; SINGH; KAUR, 2015).

4.1.4 Processo de busca e conexão com dispositivos

O processo de estabelecer comunicação entre os dispositivos ocorre em duas etapas. Primeiramente é realizada a busca de dispositivos (*inquiry*) e na sequência é estabelecida a conexão com o dispositivo selecionado (*paging*).

Para o processo de busca são utilizados 32 canais da banda Bluetooth. O dispositivo que deseja conectar-se a dispositivos próximos então inicia a varredura destes 32 canais e aguarda uma resposta. Para varrer os canais, primeiramente o dispositivo divide os canais e os agrupa em dois grupos (A e B) de 16 canais cada, então em cada canal do grupo A é enviada uma mensagem de descoberta e é aguardada uma resposta. Assim que ocorre a varredura de todos os canais do grupo A este processo é então repetido no grupo B. O envio e o recebimento levam 625 µs e cada canal é percorrido ao menos 256 vezes. Além disso, cada grupo é varrido duas vezes, resultando em um total de 4 varreduras. Desta forma é possível calcular o tempo total necessário para o processo de busca, pois conhece-se o tempo de varredura de cada canal. O tempo total gasto é de $625\,\mu s \times 16 \times 256 \times 4 = 10{,}24\,s$.

No processo de conexão são trocadas mensagens com informações sobre a sincronização entre os dispositivos. O dispositivo que enviou a mensagem de descoberta torna-se o mestre, enquanto o outro é o escravo da *piconet* (HUANG; RUDOLPH, 2007; BHAGWAT, 2001).

4.2 Protocolo implementado

Nesta seção será descrito o protocolo de comunicação desenvolvido. Para atender as especificações do presente trabalho o protocolo foi implementado tanto na aplicação Android, utilizando-se a linguagem Java, quanto no gateway Modbus, utilizando-se a linguagem C++.

O protocolo de comunicação desenvolvido é um protocolo simples que utiliza o modelo de comunicação cliente-servidor. Uma porta serial RS-232 emulada sobre o protocolo Bluetooth é utilizada como camada física virtual, e o protocolo foi dividido em duas camadas: a camada de enlace de dados e a camada de aplicação.

4.2.1 Modelo de comunicação

O modelo utilizado no protocolo de comunicação desenvolvido é o modelo cliente-servidor. Neste tipo de modelo, há dois processos: o processo cliente e o processo servidor. O processo cliente envia uma solicitação ao servidor e aguarda uma resposta do mesmo. Já o servidor ao receber a solicitação deve processar a mesma: esta solicitação pode ser executar uma tarefa, apenas responder com alguma informação, entre outras (TANENBAUM, 2003).

O protocolo desenvolvido possui duas implementações: uma cliente que irá fazer as requisições e uma servidora que irá tratar e responder as requisições feitas pelo cliente. No caso deste trabalho, o dispositivo servidor é o gateway Modbus e o cliente é o aplicativo Android, que serão detalhados nos capítulos 5 e 6, respectivamente. Os comandos e requisições partem do usuário que utiliza o aplicativo, logo as requisições serão feitas pelo aplicativo Android e o gateway Modbus é responsável por processar as mesmas.

4.2.2 Camadas

O modelo OSI (*Open Systems Interconnection*) é um modelo de referência que pode ser seguido para padronizar os protocolos de comunicação. Este modelo é composto por sete camadas apresentadas na Figura 6 e os princípios que regem estas camadas são definidos como (TANENBAUM, 2003):

a) uma camada deve ser criada quando uma abstração diferente é necessária;

b) cada camada deve possuir uma função bem definida;

c) cada função deve ser escolhida levando-se em consideração os padrões de protocolos;

d) para a especificação de cada camada deve-se levar em consideração o fluxo de informação entre as interfaces;

e) a quantidade de camadas deve ser grande o suficiente para que cada função seja bem definida dentro da camada e de modo que a arquitetura se não torne pesada.

Figura 6: Camadas do modelo OSI.

Com base nestes princípios, neste trabalho foram desenvolvidas duas camadas do modelo OSI: a camada de enlace, e a camada de aplicação. Uma porta serial RS-232, emulada através do protocolo RFCOMM da tecnologia Bluetooth, foi utilizada como camada física virtual. Deste modo, apenas as camadas de interesse para este trabalho (física, de enlace, e de aplicação) serão descritas na sequência, as demais camadas do modelo OSI serão omitidas. Uma explicação detalhada dessas demais camadas pode ser encontrada em Tanenbaum (2003). A arquitetura das camadas que compõem o protocolo implementado pode ser visualizada através da Figura 7.

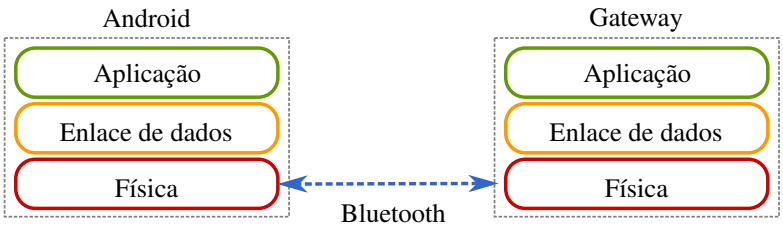


Figura 7: Camadas do protocolo desenvolvido.

De acordo com o modelo OSI a camada física é responsável por transmitir o fluxo de bits através do canal de comunicação. Já a camada de enlace de dados é responsável por transformar a interface com a camada física em uma linha aparentemente livre de erros para a camada acima, que no caso deste trabalho é a camada de aplicação. Esta camada também é responsável por enviar um pacote ou mensagem de confirmação (*acknowledgement*) toda vez que uma mensagem válida é recebida. Finalmente, a camada de aplicação tem como objetivo interagir diretamente com a aplicação de software (TANENBAUM, 2003).

4.2.3 Camada de enlace

Nesta seção será apresentada a camada de enlace de dados (DLL — *Data Link Layer*) desenvolvida. Esta camada é responsável por validar as mensagens recebidas, através dos critérios de validação especificados, e implementar a estratégia ARQ (*Automatic Repeat reQuest*). Na estratégia ARQ o remetente envia uma mensagem e aguarda uma confirmação de recebimento para enviar a mensagem seguinte. Em caso de não recebimento da mensagem de confirmação o remetente continua reenviando a última mensagem até que uma confirmação seja recebida (TANENBAUM, 2003).

A troca de informações entre os dispositivos é realizada através de pacotes cujo formato é apresentado pela Figura 8.

Figura 8: Formato dos pacotes.

Na Figura 8, *m* pode variar entre 0 e 250 bytes, assim o tamanho mínimo que um pacote pode possuir é de 6 bytes e o tamanho máximo é de 256 bytes. Uma descrição de cada campo dos pacotes é apresentada na Tabela 1.

Tabela 1: Campos dos pacotes

Campo	Descrição
SRC	endereço do remetente, que pode ser um valor entre 0 e 255
DST	endereço do destinatário, também pode ser um valor entre 0 e 255
TYPE	indica o tipo de pacote
ID	número de identificação da mensagem, incrementado a cada nova mensagem recebida
LEN	tamanho da carga útil
DATA	são os dados em si que podem estar presentes ou não, dependendo do tipo de pacote
CHK	byte utilizado para verificação da integridade da mensagem (*checksum*)

Exceto pelo campo DATA todos os demais campos possuem o tamanho fixo de 1 byte cada. Todos os oito tipos de pacote implementados e suas respectivas descrições são apresentados na Tab. 2.

Tanto no cliente quanto no servidor foram implementados *buffers* de envio e de recebimento dos pacotes. Na aplicação cliente os *buffers* comportam até 10 pacotes cada. Já no servidor, isto é, no gateway, os *buffers* comportam um pacote cada, pois levou-se em consideração o espaço reduzido de armazenamento de memória do mesmo, por ser uma aplicação embarcada.

Neste protocolo também foi implementado o reenvio de mensagens, como mencionado anteriormente. Quando uma mensagem é enviada o remetente fica aguardando uma mensagem de confirmação e, caso a mensagem de confirmação não seja recebida em um determinado período de tempo ou se for recebida uma mensagem de confirmação com um *id* diferente do esperado, a última mensagem enviada é retransmitida.

Tabela 2: Tipos de pacote

Tipo	Descrição
Ack	Mensagem de confirmação (apenas na DLL)
Restart ids	Requisição para reiniciar a conexão (apenas na DLL)
Config	Requisição para configurar os parâmetros do experimento
Send data	Requisição para iniciar um experimento e coletar os dados
Stop	Requisição para finalizar o experimento
Set gains	Requisição para ajustar os parâmetros do controlador
Response	Resposta contendo o dado coletado
End	Resposta indicando o fim do experimento

4.2.3.1 Validação das mensagens

A integridade das mensagens recebidas é realizada através da verificação de alguns dos campos dos pacotes, sendo estes campos: o endereço de destino, o *id*, o tamanho do pacote, e o *checksum*, conforme mencionado abaixo.

Verificação através do endereço de destino: Uma vez que cada dispositivo possui um endereço único na rede, caso o dispositivo receba uma mensagem que não foi endereçada a ele a mensagem é simplesmente descartada.

Verificação do *id*: Caso o endereço de destino esteja correto é realizada a verificação do *id* da mensagem, que espera-se ser recebido em sequência. Caso uma mensagem seja recebida porém o *id* da mesma seja diferente do *id* esperado a mensagem é considerada inválida e então descartada.

Verificação do tamanho máximo do pacote: Para tanto valida-se o campo LEN, se o mesmo é maior do que 250 bytes o pacote é descartado.

Verificação do *checksum*: Este campo é formado através de uma operação de ou exclusivo (XOR) entre os demais bytes da mensagem. Assim para validar a mensagem faz-se uma operação XOR entre todos os bytes da mensagem (incluíndo o *checksum*), se o resultado for diferente de 0 o pacote é descartado.

Apenas as mensagens validadas através destes critérios são disponibilizadas à camada de aplicação. A camada de aplicação é notificada através de uma *flag* que sinaliza que uma mensagem válida está disponível no *buffer* de recebimento.

4.2.3.2 Algoritmo de recuperação

Do modo como o protocolo foi implementado, a cada nova mensagem válida recebida o *id* é incrementado. Assim, se a conexão for reiniciada em um dos dispositivos, o *id* do próximo pacote a ser enviado pelo mesmo dificilmente será o *id* esperado pelo outro dispositivo. Esta situação ocorre toda vez que a conexão em um dos dispositivos é reiniciada, pois ao reiniciar a conexão todo o estado da aplicação também é reiniciado. Isto é necessário, uma vez que não é possível recuperar os dados perdidos, e qualquer experimento deve ser cancelado.

Uma forma de resolver este problema de *id's* desencontrados é fazer com que os *id's* sejam reiniciados em ambos os dispositivos automaticamente. Contudo, como utiliza-se uma porta serial física entre o hardware do gateway e o adaptador Bluetooth, não é possível determinar diretamente se a conexão está ativa. A única pista sobre a perda de conexão é dada pelos seus efeitos colaterais ou falhas (*timeout* e perda de pacotes). A

solução escolhida é reiniciar os *id's* toda vez que um determinado número de falhas ocorre. Ressalta-se que esta não é a única solução possível, mas foi escolhida por questões de simplicidade. O diagrama de blocos do algoritmo pode ser visualizado através da Figura 9.

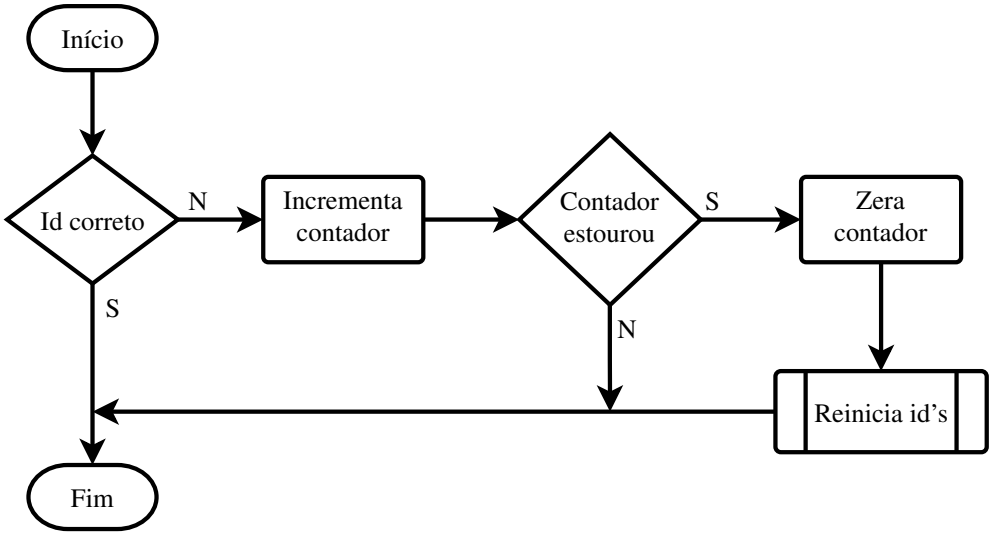


Figura 9: Diagrama de blocos do algoritmo de recuperação

O algoritmo de recuperação foi implementado na aplicação cliente utilizando-se um contador de falhas. Este é incrementado cada vez que uma falha ocorre e reiniciado para zero caso um pacote válido seja recebido. Toda vez que o contador atinge um limite predeterminado (neste caso, 4) o cliente reinicia seu *id* e envia uma mensagem solicitando que o *id* do servidor também seja reiniciado. Os tipos de falha considerados são: qualquer mensagem com *id* diferente do esperado ou o não recebimento de uma mensagem de confirmação dentro do período de tempo estipulado (*timeout*).

4.2.4 Camada de aplicação

A camada de aplicação trata as mensagens que chegam através da camada de enlace ou que são geradas pela aplicação (cliente ou servidor). As mensagens que chegam através da camada de enlace já foram validadas e não necessitam de correção de erros. As mensagens podem ser de dois tipos: mensagens de comando (comandos ou requisições) ou mensagens de resposta, conforme apresentado na Tab. 2.

Cada aplicação tem uma forma específica de lidar com as mensagens recebidas que será descrita nos capítulos seguintes.

5 GATEWAY BLUETOOTH-MODBUS

5.1 Modbus

O protocolo Modbus é um protocolo de transmissão desenvolvido para sistemas de controle de processos. É um protocolo simples, aberto e flexível que permite a troca de dados analógicos e digitais entre dispositivos. Por isso, este é um protocolo consolidado, utilizado amplamente na indústria.

O modelo de comunicação implementado pelo Modbus é o modelo mestre-escravo, onde um dispositivo mestre pode se comunicar com até 247 dispositivos escravos. Neste tipo de comunicação somente o mestre faz requisições e os escravos apenas respondem quando uma requisição é recebida.

No protocolo Modbus é implementada apenas a sétima camada do modelo OSI, que é a camada de aplicação, assim sendo, nenhuma camada física é definida. Por esta razão, o Modbus pode operar sobre diferentes tipos de camadas físicas, como pode ser observado na Figura 10. Em outras palavras, pode-se dizer que o protocolo Modbus disponibiliza uma forma simples de comunicação entre várias camadas físicas (MACKAY et al., 2004), (MODBUS, 2012).

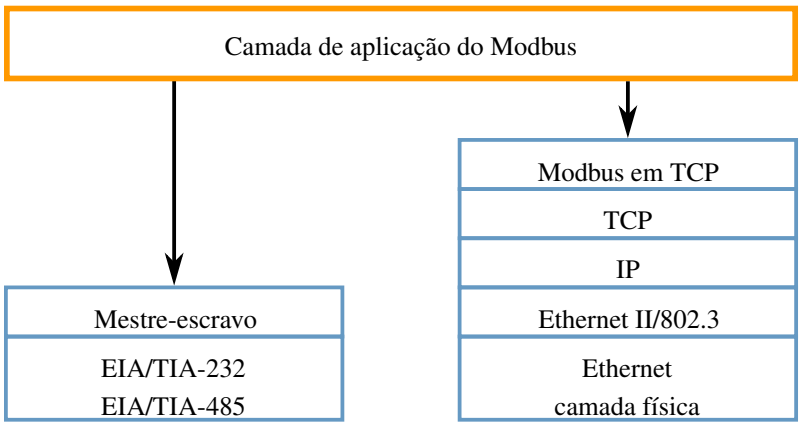


Figura 10: Pilha do protocolo Modbus. Fonte: Adaptado de MODBUS (2012).

A Figura 10 mostra algumas das camadas físicas sobre as quais é possível utilizar o Modbus como camada de aplicação. É possível utilizar o Modbus sobre o protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*) possibilitando a comunicação sobre o protocolo Ethernet. Outra possibilidade é utilizar o Modbus sobre uma camada física serial, como o EIA/TIA-232, e o EIA/TIA-485, por exemplo (MODBUS, 2012). Quando o Modbus é utilizado sobre uma camada física serial dois modos de transmissão podem ser utilizados: o modo RTU (*Remote Terminal Unit*), ou o modo ASCII (*American*

Standard Code for Information Interchange). O modo de comunicação define como os bits são arranjados nos campos das mensagens. Além disso, determina como a informação é empacotada nos campos das mensagens e decodificada (MODBUS, 2006).

5.1.1 Modo de transmissão RTU

Neste trabalho, o modo de transmissão utilizado no gateway desenvolvido foi o modo de transmissão RTU, pois de acordo com a especificação, a implementação deste modo é obrigatória enquanto que a implementação do modo ASCII é opcional. A Figura 11 apresenta o formato padrão dos pacotes para o modo RTU.

Figura 11: Formato dos pacotes do protocolo Modbus.

Os campos “Endereço” e “Função” possuem o tamanho fixo de 1 byte cada, e o campo “Dados” pode assumir o tamanho de 0 a 252 bytes. Já o campo “CRC” possui um tamanho fixo de 2 bytes. Assim os pacotes podem possuir o tamanho mínimo de 4 bytes e o tamanho máximo de 256 bytes.

O campo “Endereço” corresponde ao endereço do destinatário, isto é, o endereço do dispositivo escravo que deve processar a requisição. Os endereços possíveis para os dispositivos escravos podem ser entre 1 e 247, e cada escravo deve possuir um endereço único na rede Modbus. O endereço zero é utilizado para definir mensagens de *broadcast*, que são processadas nos dispositivos escravos mas nenhuma mensagem de resposta é enviada ao mestre. Já o segundo campo do pacote é o campo “Função”, sendo utilizado para indicar qual tipo de ação deve ser executada pelo dispositivo escravo.

O terceiro campo da mensagem é o campo “Dados” que pode estar presente ou não na mensagem. Este campo possui tamanho variável que é definido de acordo com o tipo de função especificado no pacote, podendo conter parâmetros de requisição e resposta.

Para verificar a integridade das mensagens é utilizado o campo “CRC”. A validação do pacote é realizada através do cálculo do CRC nos bits da mensagem, o tipo de CRC utilizado no protocolo Modbus é o CRC-16. A detecção de erros no pacote não é utilizada apenas para garantir a integridade dos mesmos, mas também para prevenir que o dispositivo escravo atue com base em uma mensagem que tenha sido corrompida durante a transmissão.

Em linhas gerais, o processo de comunicação entre dois dispositivos em uma rede Modbus acontece da seguinte maneira: quando o mestre deseja fazer uma requisição a um dos escravos, ele envia uma mensagem específica (de acordo com a requisição a ser executada) com o endereço do respectivo escravo. O escravo ao receber a mensagem verifica se a mesma foi recebida corretamente. Em caso afirmativo, a mensagem é processada, e um pacote de resposta (contendo informações de sucesso ou falha) com o seu próprio endereço é enviado ao mestre, permitindo que o mestre consiga identificar de qual escravo a mensagem foi recebida. Caso o escravo tenha recebido a mensagem com erro a mesma é descartada e nenhuma resposta é enviada ao mestre.

5.1.1.1 Transmissão das mensagens

O conjunto de bytes que compõem o pacote são transmitidos em uma sequência contínua de caracteres. Para cada caractere transmitido são utilizados 11 bits: um bit de *start*, 8 bits de dados, 1 bit de paridade, e 1 bit de *stop*. Para verificar a integridade dos bits

transmitidos é utilizado o método de verificação de dados conhecido como paridade, que pode ser par ou ímpar.

O bit de paridade é um bit ao qual é atribuído o valor 0 ou 1 para indicar a quantidade total de bits iguais a 1 estão na sequência de interesse. Na paridade par garante-se que as mensagens transmitidas sempre possuirão um número par de bits iguais a 1. Para tanto, são contados todos os bits 1 da mensagem, se o resultado é ímpar o bit de paridade recebe 1, fazendo assim com que a mensagem possua paridade par, caso o resultado seja par o bit de paridade recebe zero. Já no caso da paridade ímpar é realizado o oposto, de modo a garantir que as mensagens serão sempre verificadas com um número ímpar de bits iguais a 1 (MACKAY et al., 2004).

Além disso, pode-se não utilizar paridade, neste caso, nenhuma verificação é realizada e assume-se que as mensagens são recebidas sem erros. Contudo, é preciso manter a sequência de 11 bits do caractere, para tanto o bit de paridade é preenchido com o bit de *stop*, ou seja, este bit é repetido duas vezes.

5.1.1.2 Funções

As funções do protocolo Modbus são identificadas através do campo "Função" do pacote, conforme mencionado anteriormente. Cada função possui um código único que corresponde a um tipo de ação a ser executado. Uma série de funções distintas são implementadas, mas apenas as pertinentes a este trabalho serão abordadas e são apresentadas na Tabela 3.

Tabela 3: Tipos de funções do Modbus

Código da função	Descrição
03 (0x03)	Leitura de registradores
05 (0x05)	Escrita em uma bobina
06 (0x06)	Escrita em um registrador
16 (0x10)	Escrita em múltiplos registradores

Leitura de registradores (0x03): Utilizada para realizar a leitura de um ou mais registradores em sequência. O campo "Dados" deste tipo de mensagem é formado por quatro bytes, dois bytes para representar o endereço do primeiro registrador a ser lido, e dois bytes para informar a quantidade de registradores que devem ser lidos. O formato do pacote deste tipo de função é mostrado através da Figura 12. A mensagem de resposta é composta pela quantidade de registradores que foram lidos e pelos valores desses registradores.

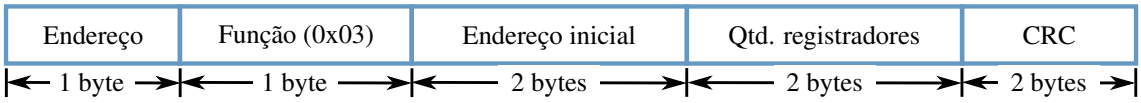


Figura 12: Formato do pacote da função para leitura de registradores. Fonte: Adaptado de (MODBUS, 2012)

Escrita em uma bobina (0x05): Esta função é utilizada para escrever em uma única bobina. Os valores de escrita possíveis são ligado e desligado. Os dados do pacote de requisição são compostos por dois bytes contendo o endereço da bobina, e dois bytes contendo o valor a ser escrito. Depois de alterar o valor da bobina uma mensagem idêntica

a recebida é enviada ao mestre. Na Figura 13 é apresentado o formato dos pacotes desta função.

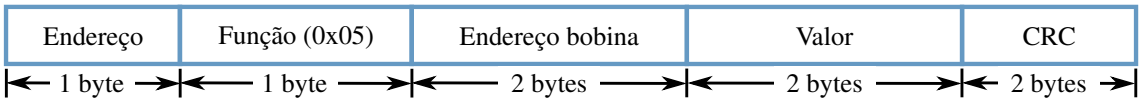


Figura 13: Formato do pacote da função para escrita em uma bobina. Fonte: Adaptado de (MODBUS, 2012)

Escrita em um registrador (0x06): Este código de função é utilizado para escrever em um único registrador. Na requisição são enviados o endereço do registrador e o valor a ser escrito no mesmo. Como resposta, após escrever no registrador, o escravo envia ao mestre uma mensagem igual a recebida, cujo formato pode ser visualizado na Figura 14.

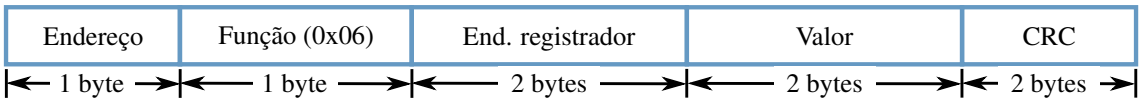


Figura 14: Formato do pacote da função para escrita em uma registrador. Fonte: Adaptado de (MODBUS, 2012)

Escrita em múltiplos registradores (0x10): Esta função é utilizada para escrita em um bloco contínuo de registradores. Os dados do pacote contém: o endereço do primeiro registrador, a quantidade de registradores a serem escritos, quantos bytes são enviados com os valores a serem escritos, e os novos valores dos registradores. O formato dos pacotes é apresentado na Figura 15, onde N representa a quantidade de registradores a serem escritos. A mensagem de resposta contém o endereço do primeiro registrador e a quantidade de registradores que foram escritos.

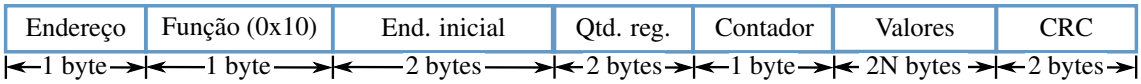


Figura 15: Formato do pacote da função para escrita múltiplos registradores. Fonte: Adaptado de (MODBUS, 2012)

Independente do código de função utilizado, em caso de erro, uma mensagem de resposta é enviada ao mestre. Esta mensagem contém: o endereço do escravo, o código de função recebido com seu bit mais significativo com valor igual a 1, o código da exceção, e o CRC. Os quatro tipos de exceção que podem ser gerados pelas funções acima citadas podem ser visualizados através da Tabela 4.

5.1.2 EIA/TIA-485

O padrão de comunicação serial EIA/TIA-485, também conhecido como RS-485, é uma extensão do padrão EIA-422. Este padrão foi desenvolvido para permitir a transferência de informação a altas taxas de transmissão. Além disso, o mesmo foi projetado com a intenção de ser utilizado como um padrão de camada física para outros padrões de nível mais alto.

Neste padrão o alcance da comunicação é de até 1200 m, e a taxa de dados pode chegar a 10 Mbps. O meio físico do padrão RS-485 é um par trançado com condutores de cobre blindados, a estratégia de acesso ao meio empregada é a mestre-escravo, e o tipo de

Tabela 4: Códigos de exceção do Modbus

(Código) Nome	Descrição
(01) Função ilegal	A função a ser executada não é suportada pelo dispositivo
(02) Endereço ilegal	O endereço de dados não é suportado pelo dispositivo
(03) Valor ilegal	O valor especificado não é válido
(04) Falha no dispositivo	Não foi possível responder a mensagem

transmissão do barramento é *half-duplex*. Em uma rede RS-485 até 32 dispositivos podem ser adicionados ao mesmo segmento. Um segmento é definido como uma seção de rede, onde o mesmo sinal elétrico é compartilhado pelos dispositivos.

5.2 Gateway

Nesta seção será apresentado o gateway desenvolvido. O gateway foi projetado para permitir a comunicação entre um dispositivo que utilize o protocolo de comunicação apresentado anteriormente (Capítulo 4), e um controlador que utilize o protocolo Modbus RTU sobre a camada física RS-485.

5.2.1 Hardware

No gateway Bluetooth-Modbus desenvolvido, na parte de hardware, utilizou-se uma placa Arduino Mega com o microcontrolador Atmega 2560. Já a parte de software foi implementada através da linguagem de programação C++, por esta ser uma das linguagens compatíveis com o hardware utilizado.

O Arduino é uma plataforma aberta que disponibiliza uma forma simples de utilização de seus componentes de hardware e software. Além disso, é uma ferramenta de baixo custo, a plataforma de desenvolvimento é gratuita, e é possível agregar diferentes componentes de hardware e software de forma simples e fácil.

Para atender os objetivos deste trabalho, foram adicionados dois módulos de comunicação ao hardware do gateway: um módulo conversor Bluetooth para TTL, e um módulo conversor TTL para RS-485. A Figura 16 apresenta o hardware do gateway com os dois

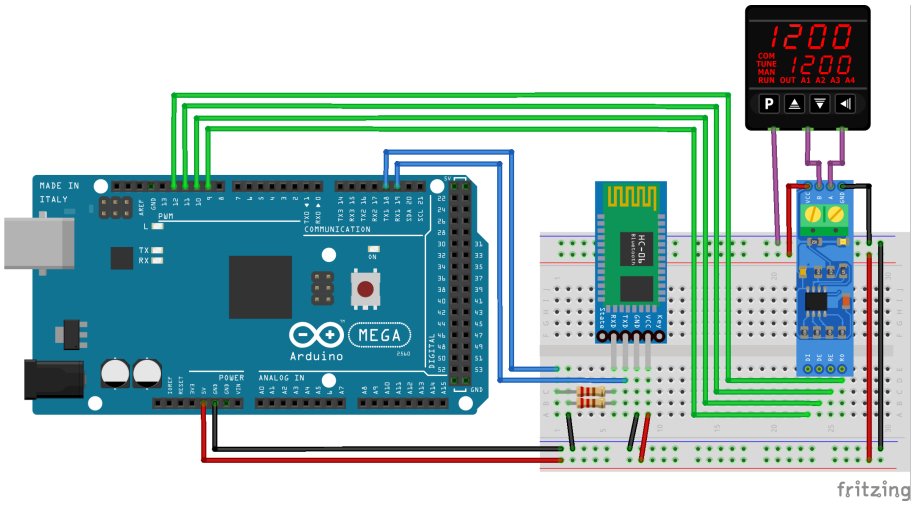


Figura 16: Hardware do gateway Bluetooth-Modbus.

módulos adicionados.

Embora o Arduino possua portas seriais, não é possível apenas conectar a porta serial RS-485 do dispositivo Modbus à porta serial do Arduino. Isto ocorre pois o nível de tensão do Arduino é TTL (5V) e o nível de tensão do RS-485 é de -7V a 12V (MACKAY et al., 2004), por isso foi necessário adicionar um conversor TTL para RS-485 ao hardware do gateway.

5.2.2 Comunicação

Através da Figura 17 é possível ter uma visão geral da aplicação desenvolvida. Os pacotes enviados pelo aplicativo Android são recebidos pelo gateway e tratados através da máquina de estados. Esses pacotes são convertidos em pacotes de comando Modbus, que são enviados ao controlador. A resposta do controlador é convertida pelo gateway em uma mensagem específica e enviada ao aplicativo.

Para tanto, foram implementadas duas classes no gateway: uma para lidar com a comunicação via Modbus, e outra para lidar com a comunicação através do protocolo Bluetooth. Na classe responsável pela comunicação Modbus foram implementados os comandos apresentados na Tabela 3. Já na classe que utiliza o Bluetooth para comunicação foi implementado o protocolo de comunicação desenvolvido, conforme descrito no Capítulo 4.

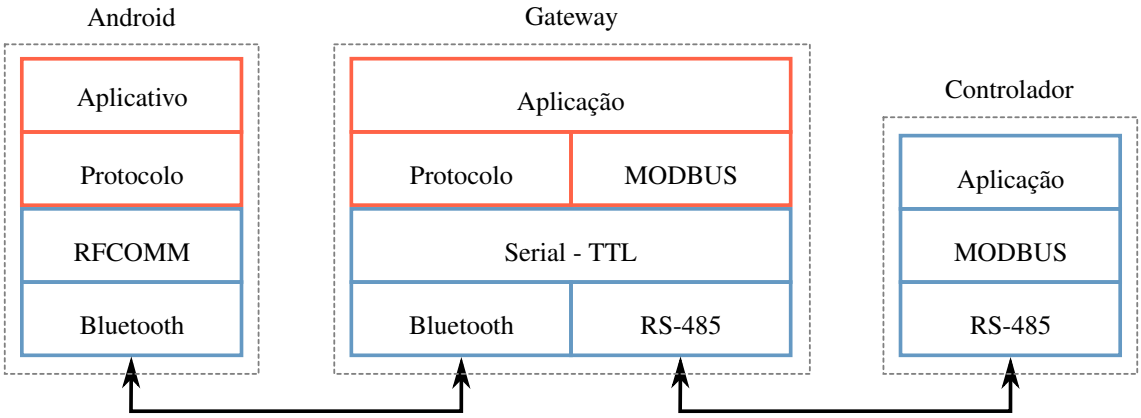


Figura 17: Diagrama da aplicação.

O modelo de comunicação utilizado é o modelo cliente-servidor, onde o gateway é o servidor e o aplicativo Android é o cliente. Desta forma, o gateway nunca inicia a comunicação, e apenas responde quando uma mensagem de requisição é enviada pelo aplicativo. Cada pacote recebido pelo gateway (apresentados na Tabela 2) é tratado através de uma máquina de estados implementada no mesmo. A Figura 18 apresenta a máquina de estados simplificada implementada no gateway.

A máquina de estados inicia com o estado “Idle”, enquanto aguarda o início da comunicação. Quando um pacote do tipo “Config” é recebido os dados de configuração são armazenados e o estado da máquina não é alterado. O mesmo ocorre quando uma mensagem do tipo “Set gains” é recebida, os parâmetros do controlador são ajustados e o estado da máquina permanece o mesmo.

Quando um pacote do tipo “Send data” é recebido o gateway muda para o estado “Iniciar”, onde o controlador é configurado para o experimento. Na sequência o gateway altera seu estado para “Estabilizar”, onde a entrada é estabilizada em um valor definido. Na prática, esta etapa não é necessária, porém, como a aplicação será validada em um

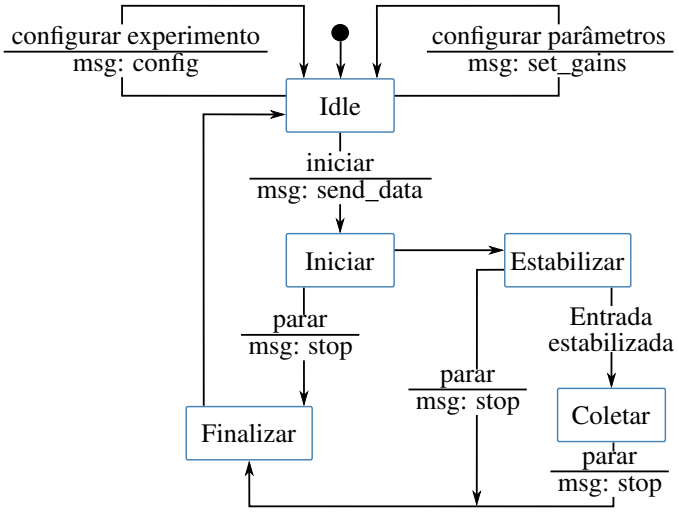


Figura 18: Máquina de estados do gateway.

processo térmico deseja-se que ao menos as condições iniciais sejam semelhantes para permitir a comparação dos resultados. Embora não tenha sido apresentado na máquina de estados, por questões de simplicidade, para estabilizar a entrada do processo antes de executar o experimento verifica-se se a diferença entre o último valor medido e a média dos últimos cinco valores medidos é menor do que um valor previamente estipulado. Em caso afirmativo, inicia-se a execução do experimento, senão, continua-se realizando a verificação dos valores até que a diferença seja menor que o valor estipulado. Assim que a entrada é estabilizada o estado do gateway é alterado para “Coletar”. Neste estado o experimento é realizado, onde a entrada do processo é alterada de acordo com o tipo de entrada especificado. Além disso, os dados de entrada e saída do processo são medidos, e enviados através do protocolo Bluetooth para o aplicativo Android.

Em qualquer um desses três estados (“Inicializar”, “Estabilizar” ou “Coletar”) caso uma mensagem de “Stop” seja recebida pelo gateway o estado do mesmo é alterado para “Finalizar”, onde o experimento é interrompido e uma mensagem de “End” é enviada como resposta ao aplicativo. Por fim, o gateway volta para o estado “Idle” e aguarda uma nova comunicação.

Como pode ser visto, através da máquina de estados implementada, não é possível configurar o experimento, ou alterar os ganhos do controlador durante o processo de coleta de dados. Desta maneira, caso seja necessário realizar algum ajuste enquanto a coleta de dados é realizada é preciso parar o experimento em execução.

6 APLICATIVO ANDROID

Este capítulo é dedicado à apresentação da aplicação móvel desenvolvida para atender os requisitos do trabalho proposto. Esta aplicação foi desenvolvida para a plataforma Android utilizando-se a linguagem de programação Java. O Android foi escolhido por ser o sistema operacional da maioria dos dispositivos móveis, além disso, é um sistema de código aberto, livre, e existem diversas ferramentas gratuitas para o desenvolvimento de aplicativos para esta plataforma. A linguagem Java foi empregada por ser a linguagem oficial para desenvolvimento de aplicativos Android. Desta forma, nas seções seguintes, serão apresentadas uma breve introdução à plataforma Android, bem como uma descrição detalhada do aplicativo desenvolvido.

6.1 Android

O Android é um sistema operacional desenvolvido para dispositivos móveis baseado em uma versão modificada do Linux. Portanto, é um sistema operacional de código aberto, e grande parte do seu código foi desenvolvida sob a licença *open source* do Apache (LEE, 2012). A arquitetura da plataforma Android, de modo geral, pode ser dividida em quatro camadas principais como ilustrado na Figura 19.

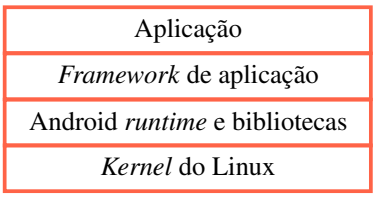


Figura 19: Arquitetura da plataforma Android. Fonte: Adaptado de Lee (2012).

A camada inferior da plataforma é o *kernel* do Linux, ou seja, a base do sistema operacional. Esta camada contém todos os *drivers* de baixo nível para os componentes de hardware dos dispositivos Android. Além disso, é responsável pelo gerenciamento de processos e gerenciamento de recursos, tais como memória, rede, entre outros. A camada seguinte é composta por um conjunto de bibliotecas e pelo Android *runtime*. Através deste conjunto de bibliotecas são disponibilizadas as funcionalidades principais do sistema operacional. Já o Android *runtime* disponibiliza um conjunto específico de bibliotecas que permitem o desenvolvimento de aplicativos Android na linguagem de programação Java.

Através da camada de *framework* de aplicação pode-se acessar os recursos disponibilizados pelo sistema operacional. Em outras palavras, esta camada disponibiliza as APIs (*Application Programming Interface*) suportadas pela plataforma, que são utilizadas no

desenvolvimento de aplicativos. Já a última camada da plataforma é a camada de aplicação que compreende todos os aplicativos do dispositivo Android, ou seja, todas as aplicações que rodam no dispositivo operam nesta camada (LEE, 2012; ZECHNER; GREEN, 2011).

O desenvolvimento de aplicativos Android requer a utilização de algumas ferramentas como o SDK (*Software Development Kit*), o ADB (*Android Debbuger Bridge*), o ADT (*Android Development Tools*), uma IDE (*Integrated Development Environment*), e uma linguagem de programação. O SDK possui um conjunto de ferramentas para o desenvolvimento dos aplicativos, dentre as principais estão: um depurador, um emulador, e bibliotecas. O ADB é uma das ferramentas disponibilizadas pelo SDK, que oferece uma interface de depuração para o Android. Assim sendo, esta é uma ferramenta de linha de comando que realiza a comunicação entre o computador, onde o software está sendo desenvolvido, e o dispositivo Android. Já o ADT é uma extensão para o ambiente de desenvolvimento que permite a criação e depuração de aplicações para a plataforma Android.

O ambiente de desenvolvimento (IDE) é a ferramenta na qual o aplicativo é desenvolvido. No presente trabalho utilizou-se o JetBrains' IntelliJ IDEA. Como linguagem de programação utilizou-se a linguagem Java, que como mencionado anteriormente, é a linguagem padrão para desenvolvimento de aplicativos Android (YENER; DUNDAR, 2016; LEE, 2012).

6.2 Aplicativo

Nesta seção será apresentada a aplicação Android desenvolvida. As funcionalidades da aplicação foram distribuídas em três telas que são acessadas através de abas no topo do aplicativo. A tela principal é dedicada ao método VRFT, ao cálculo dos parâmetros, e aos quesitos relacionados ao experimento, como configuração, início e interrupção do mesmo. A tela seguinte é reservada para as configurações do experimento e do cálculo dos parâmetros. Estas últimas incluem a seleção do caso descasado e do fator de esquecimento. Na última tela do aplicativo é disponibilizado um recurso gráfico para visualização dos dados coletados. Além disso, foi implementado o protocolo de comunicação apresentado no Capítulo 4.

6.2.1 Tela principal

Na tela principal do aplicativo foram implementadas as funcionalidades relacionadas ao método VRFT, à comunicação Bluetooth, e a execução e configuração do experimento. Através da Figura 20 é possível visualizar a tela inicial da aplicação desenvolvida.

A funcionalidade principal do aplicativo é realizar o ajuste dos parâmetros do controlador através do método VRFT, para tanto, foram implementadas a abordagem original, e a abordagem recursiva do método, conforme apresentado no Capítulo 3. Referente à abordagem original, foi implementado o caso ideal e o caso descasado, com ou sem zeros de fase não mínima. Já para a abordagem recursiva foi implementado o VRFT recursivo com fator de esquecimento. As principais interações disponíveis na interface do aplicativo são apresentadas nos parágrafos seguintes.

Classe do controlador e modelo de referência: independente da abordagem utilizada, para o cálculo dos parâmetros, é necessário escolher a classe de controladores $\bar{C}(z)$, e o modelo de referência $M(z)$. A classe de controladores pode ser escolhida através da combinação das três ações básicas de controle: proporcional, integral, e derivativa. No

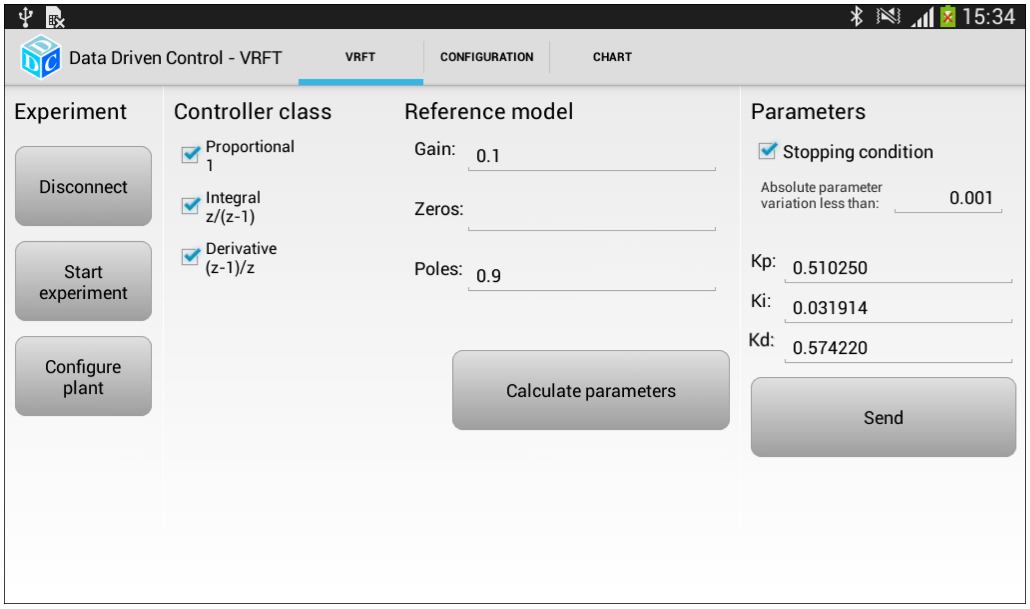


Figura 20: Tela inicial da aplicação.

caso de um controlador PID, a classe de controle utilizada é dada por

$$\bar{C}(z) = \begin{bmatrix} 1 & \dfrac{z}{z-1} & \dfrac{z-1}{z} \end{bmatrix}^{\mathrm{T}}, \tag{37}$$

que devido à sua estrutura, faz com que os ganhos do vetor de parâmetros estimado $\hat{\rho}$ correspondam exatamente aos ganhos K_p, K_i e K_v do controlador. Já a função de transferência do modelo de referência $M(z)$ é escolhida através da informação do ganho, dos zeros e dos polos do modelo. As abordagens implementadas são resumidas nos parágrafos seguintes.

Planta com zeros de fase não mínima: a abordagem do método VRFT para o caso de o modelo de referência apresentar zeros de fase não mínima também foi implementada, conforme apresentado na Seção 3.2. Contudo, esta abordagem foi implementada apenas para o VRFT original e os zeros de fase não mínima são identificados automaticamente do modelo de referência.

VRFT original: a abordagem original do método VRFT foi implementada conforme apresentada na Seção 3.1, sendo possível utilizar esta opção quando um experimento não está sendo executado e há um arquivo de dados armazenado no aplicativo, ao requisitar o cálculo dos parâmetros. Esta opção permite inclusive que um novo modelo de referência seja utilizado sem a necessidade de um novo experimento.

VRFT recursivo: implementado conforme a Seção 3.4, é utilizado automaticamente durante a execução do experimento, empregando as configurações informadas para recalcular os parâmetros até que o critério de parada seja atingido ou que o usuário finalize o experimento. Assim sendo, a cada nova amostra recebida os parâmetros do controlador são calculados e atualizados nos campos K_p, K_i e K_d do aplicativo, o que permite uma prática visualização dos ganhos do controlador a cada iteração. Além disso, uma vez que o cálculo dos parâmetros foi implementado em um dispositivo móvel, para garantir o desempenho da aplicação as operações matemáticas sobre as matrizes foram implementadas em uma biblioteca própria. Esta biblioteca tira proveito das propriedades bem conhecidas das matrizes envolvidas, como a sua dimensão e simetria, aumentando o desempenho do algoritmo, ou seja, reduzindo o consumo de memória e diminuindo o tempo de processamento.

Critério de parada: esta configuração determina uma variação mínima absoluta de cada um dos parâmetros do controlador. Assim, esta funcionalidade permite que o experimento seja interrompido automaticamente quando a variação dos parâmetros já não é mais "significativa". Se habilitado, este critério pode ser ajustado de acordo com a necessidade, ou seja, não é um parâmetro fixo. Esta característica é útil para experimentos longos, pois desobriga o usuário de permanecer próximo do dispositivo móvel.

Ajuste do controlador: no caso do VRFT original, assim que os parâmetros são calculados os mesmos podem ser enviados ao gateway. Já no caso do VRFT recursivo, os parâmetros ficam disponíveis para serem enviados assim que o experimento é interrompido. Os parâmetros são enviados através de um pacote específico ao gateway, que configura o controlador.

Além destas funcionalidades, o usuário tem a opção de salvar os dados em um arquivo no dispositivo cada vez que é iniciado um novo experimento. É conveniente salvar os dados no arquivo pois é possível visualizá-los posteriormente através do gráfico e, além disso, é com base nos dados salvos que o VRFT original calcula os parâmetros do controlador quando requisitado.

6.2.2 Tela de configuração

Na tela de configuração, apresentada na Figura 21, é possível configurar as demais funcionalidades do aplicativo, que são relacionadas à abordagem do método VRFT para o caso descasado, ao fator de esquecimento, e a configuração do experimento. Desta forma, nesta tela é possível escolher entre o VRFT para o caso ideal ou descasado, bem como informar o valor a ser utilizado para o fator de esquecimento λ. Além disso, pode-se configurar o tipo de experimento, o período de amostragem, e o tipo de entrada do processo. Entretanto, estes parâmetros de configuração ainda precisam ser enviados ao gateway através do botão na tela principal do aplicativo.

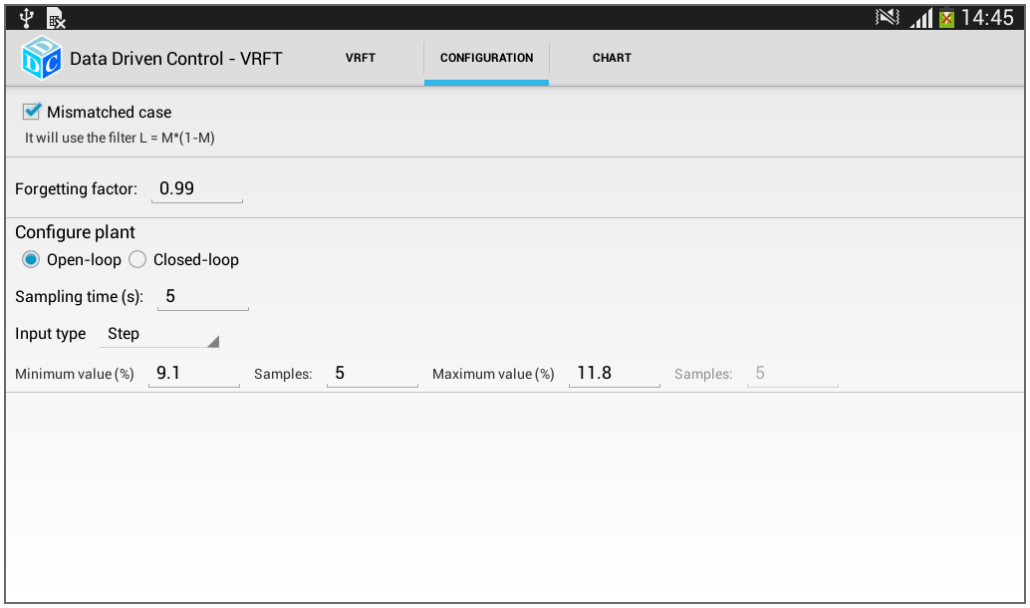


Figura 21: Tela de configuração da aplicação.

Caso descasado: este caso foi implementado conforme apresentado na Seção 3.3, contudo, a sua utilização é opcional. Através da tela de configuração é possível ativá-lo para utilizar o filtro (18), que é aplicado independente do tipo de entrada utilizado. Quando

ativado, este filtro é utilizado tanto para a abordagem original do método VRFT quanto para a abordagem recursiva do mesmo.

Fator de esquecimento: controla se o VRFT recursivo (Seção 3.4) original ou o VRFT recursivo com fator de esquecimento (Seção 3.5) será empregado para o cálculo dos parâmetros. A utilização de um ou outro depende do valor informado para o fator de esquecimento λ, que está disponível para ajuste nesta tela. Caso seja escolhido um fator de esquecimento unitário os parâmetros são calculados através do VRFT recursivo. Caso contrário, se o valor escolhido para λ for menor que um a abordagem com fator de esquecimento é utilizada. Em qualquer um dos casos utilizados, o cálculo dos parâmetros do controlador é realizado automaticamente, enquanto o experimento é executado.

Tipo de experimento: pode-se configurar o tipo de experimento a ser executado podendo ser um experimento em laço aberto ou fechado. Em um experimento em laço aberto, a entrada é gerada e ambos os sinais de entrada $u(t)$ e de saída $y(t)$ são amostrados. Já em um experimento em laço fechado é gerada como entrada o sinal $r(t)$, e os sinais $u(t)$ e $y(t)$ são amostrados. Embora o ajuste dos parâmetros seja realizado utilizando-se os sinais $u(t)$ e $y(t)$, os três sinais $r(t)$, $u(t)$ e $y(t)$ são sempre enviados para o aplicativo, desta forma, possibilitando que estes sinais sejam salvos para posterior análise. Além disso, pode-se escolher o período de amostragem T_s do experimento, que é um valor inteiro em segundos.

Configurações do sinal de entrada: o sinal de entrada do experimento pode ser escolhido dentre quatro tipos: degrau, PRBS (*Pseudo-Random Binary Sequence*), onda quadrada, ou sinusoidal. Cada tipo de entrada é identificado através de um código único, e todas as informações pertinentes à entrada são enviadas em uma mensagem de configuração. A implementação de cada entrada é realizada no gateway, ou seja, é no gateway que os valores do sinal de entrada do processo são efetivamente calculados. As outras configurações do sinal variam dependendo do tipo de entrada escolhido. Para uma entrada do tipo degrau pode-se escolher um valor para o nível baixo, quantas amostras deseja-se que a mesma permaneça em nível baixo, e o valor de nível alto da entrada. Os valores para os níveis são valores decimais, e o número de amostras é um valor inteiro. Já em uma entrada PRBS é necessário informar o valor para nível baixo, alto, e o período do sinal, também em número de amostras. Contudo, a quantidade de amostras, neste caso, é utilizada para os dois níveis. Em uma entrada do tipo onda quadrada pode-se escolher o nível baixo, o nível alto, e a quantidade de amostras para cada um dos níveis. A última opção de entrada, do tipo sinusoidal, é a composta por três senoides com características diferentes, caracterizando um sinal suficientemente rico de ordem 3. A este sinal é ainda somado um valor constante, visando evitar a saturação do sinal de controle. Este tipo de entrada não permite nenhuma configuração adicional e foi implementada de acordo com a seguinte equação:

$$u(t) = 6 + 2\sin\left(\frac{2\pi t}{60}\right) - 2\sin\left(\frac{2\pi t}{30}\right) + 2\sin\left(\frac{2\pi t}{10}\right).$$

6.2.3 Tela com o recurso gráfico

Um recurso gráfico para exibição dos dados coletados foi adicionado à última tela da aplicação Android, como mostrado na Figura 22. Assim, é possível monitorar o experimento em execução, através da visualização dos dados coletados, ou seja, toda vez que um dado novo é recebido o mesmo é atualizado no gráfico. Neste gráfico são mostrados os três sinais coletados: o sinal de entrada $u(t)$, o sinal de saída $y(t)$ e o sinal de referência $r(t)$.

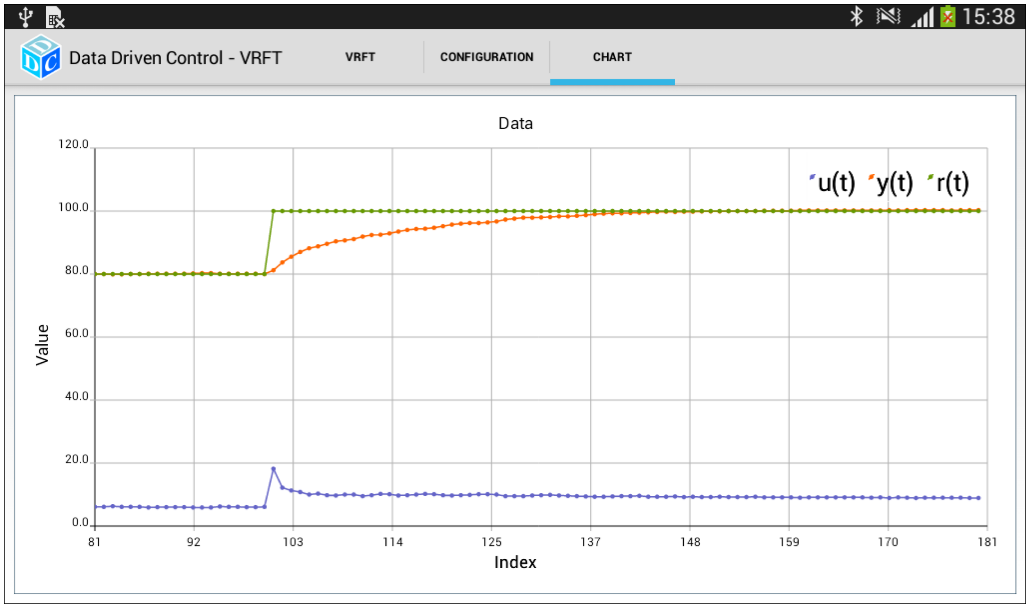


Figura 22: Tela do gráfico da aplicação.

Através deste recurso também é possível visualizar os dados armazenados no arquivo no dispositivo Android, desde que o experimento não esteja sendo executado. Para tanto, as últimas 100 amostras do arquivo são carregadas e disponibilizadas para visualização no gráfico.

6.2.4 Processo de conexão

A comunicação Bluetooth também pode ser gerenciada através do aplicativo, ou seja, a conexão e a desconexão com o gateway são realizadas manualmente pelo usuário. Através da Figura 23 é possível visualizar a máquina de estados simplificada do processo de conexão. A desconexão com o gateway pode ocorrer por diferentes causas, por conta disso e visando simplificar a máquina de estados, o tratamento da desconexão, bem como as

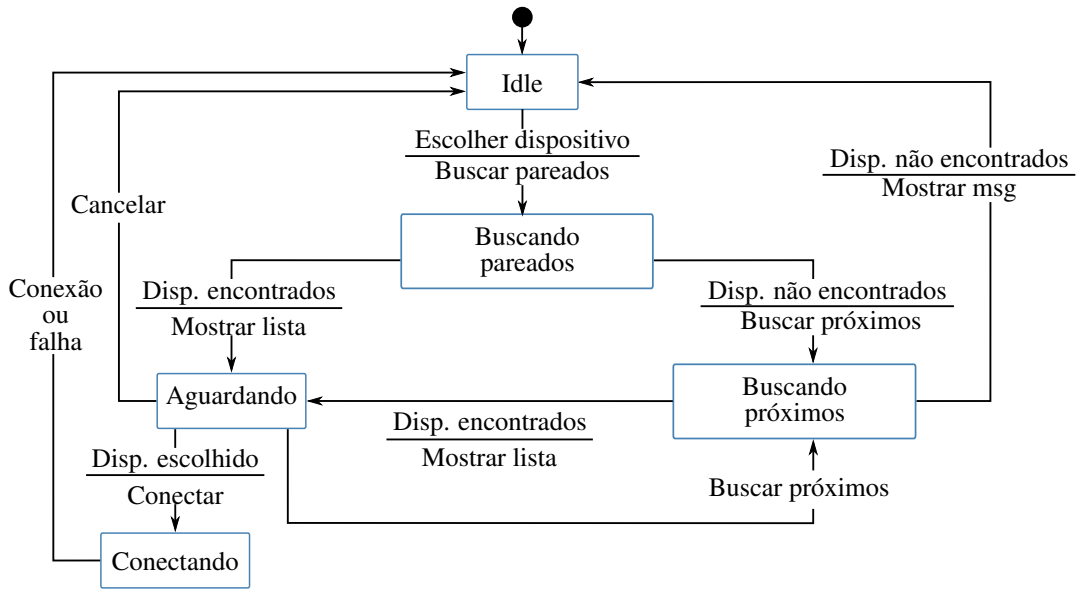


Figura 23: Máquina de estados do processo de conexão via Bluetooth.

possíveis exceções geradas, não são apresentadas na mesma.

A máquina de estados inicia com o estado “Idle” e assim que um evento de escolha de dispositivo é gerado é iniciada a busca por dispositivos pareados e o estado da máquina é alterado para “Buscando pareados”. Neste estado, caso sejam encontrados dispositivos é mostrado ao usuário uma lista com os dispositivos encontrados e o estado da máquina é alterado para “Aguardando”. Caso não sejam encontrados dispositivos é iniciado o processo de localizar dispositivos próximos e a máquina tem seu estado alterado para “Buscando próximos”. No estado “Aguardando” caso a operação seja cancelada a máquina tem seu estado alterado para “Idle”, caso contrário, ou seja, se um dispositivo for escolhido tenta-se conectar no mesmo e o estado é alterado para “Conectando” onde é realizada a conexão com o dispositivo. No estado “Buscando próximos” é realizada uma busca pelos dispositivos próximos, caso nenhum seja encontrado o estado da máquina retorna para “Idle”. Contudo, caso sejam localizados o estado é alterado para “Aguardando” e o comportamento segue como mencionado anteriormente para este estado.

6.2.5 Máquina de estados da aplicação

O protocolo de comunicação desenvolvido para este trabalho foi implementado e utilizado na forma de uma biblioteca dentro do aplicativo Android. Mais especificamente, a camada de enlace foi implementada nesta biblioteca e a camada de aplicação foi implementada diretamente no aplicativo.

O aplicativo tem papel de cliente e, por conta disso, é a partir da aplicação que as requisições são enviadas ao gateway. A máquina de estados simplificada pode ser visualizada através da Figura 24. Por questões de simplicidade foram omitidas as partes onde são tratadas a perda da conexão Bluetooth com o gateway e o tratamento das possíveis exceções geradas durante a comunicação.

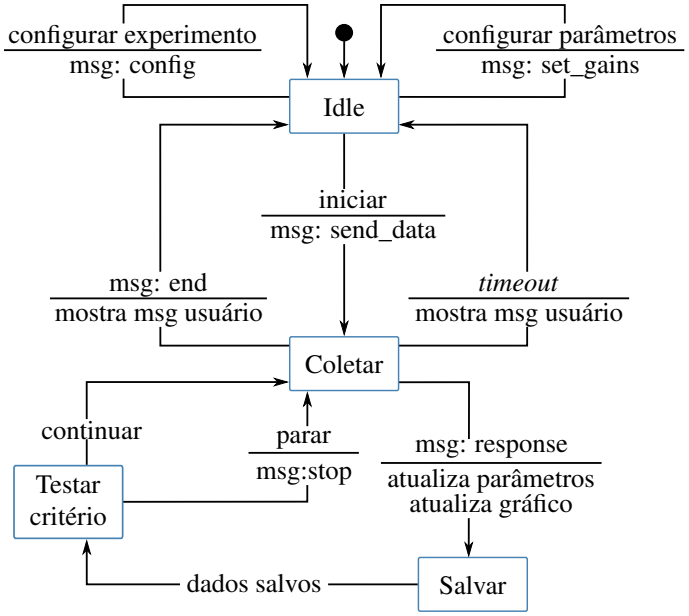


Figura 24: Máquina de estados simplificada da camada de aplicação.

O estado inicial da aplicação é o estado “Idle”, no qual a aplicação aguarda o recebimento de uma ação a ser executada. Caso a ação seja configurar o experimento uma mensagem do tipo “config” é enviada ao gateway e a aplicação permanece no estado “Idle”. O mesmo acontece caso a ação a ser executada seja configurar os parâmetros do

controlador, uma mensagem do tipo “set_gains” é enviada e o estado da aplicação se mantém inalterado. Por outro lado, caso a ação recebida seja para iniciar o experimento, uma mensagem de “send_data” é enviada ao gateway e o estado da máquina é alterado para “Coletar”.

No estado “Coletar” é realizada a coleta dos dados do experimento que está sendo executado. Caso nenhuma mensagem seja recebida em um determinado período de tempo, ou seja, ocorra *timeout*, ou caso tenha sido recebida uma mensagem do tipo “end”, uma mensagem específica para cada caso é apresentada para o usuário e a máquina retorna para o estado “Idle”. Contudo, enquanto as mensagens do tipo “response” são recebidas com novos dados, estes são utilizados para calcular os parâmetros do controlador através do método VRFT recursivo. Além disso, o gráfico é atualizado com os novos dados e o estado da máquina é alterado para “Salvar”, onde os dados recebidos são armazenados em um arquivo no dispositivo. Quando o processo de escrita é finalizado o estado da máquina é alterado para “Testar critério”. No estado “Testar critério” é verificado se o critério de parada está habilitado e foi atingido. Se o critério está habilitado e foi atingido, uma mensagem de “stop” é enviada ao gateway, assim espera-se que a próxima mensagem a ser recebida seja do tipo “end”, fazendo com que o experimento seja encerrado. Caso contrário, a máquina retorna para o estado “Coletar” e a coleta de dados continua acontecendo. Embora não tenha sido mostrado na máquina de estados é possível parar o experimento mesmo que o critério de parada não tenha sido atingido, ou se o mesmo não esteja habilitado. Conforme mencionado, isto é realizado manualmente pelo usuário através de um dos botões da aplicação, que envia um pacote de “stop” ao gateway.

7 ESTUDO DE CASO

Nesta seção são apresentados alguns testes realizados com o objetivo de validar a solução desenvolvida. Estes testes foram conduzidos em duas etapas: na primeira etapa foram realizados experimentos utilizando-se dados obtidos através de simulações, e na segunda etapa conduziram-se experimentos em uma planta real.

7.1 Testes com plantas simuladas

Para realizar as simulações, uma aplicação embarcada foi implementada utilizando-se o mesmo hardware empregado no gateway. Esta aplicação é capaz de simular o comportamento de uma planta que pode ser configurada conforme a necessidade, além disso, é possível simular tanto o comportamento do processo em laço aberto quanto em laço fechado (com um controlador PI ou PID adicionado ao mesmo). Por conta disto o gateway precisa ser programado com as informações dos coeficientes da função de transferência da planta, e os parâmetros do controlador.

Para simplificar o processo de simulação, o mesmo protocolo de comunicação foi utilizado e não realizou-se nenhuma alteração no aplicativo Android. Assim sendo, a comunicação entre o aplicativo Android e a aplicação embarcada também é realizada via Bluetooth. Desta maneira, não há comunicação através do Modbus, ou seja, não há comunicação com o controlador pois o processo é inteiramente simulado no Arduino.

Os experimentos simulados foram feitos de modo a explorar todas as abordagens implementadas do método VRFT. Desta forma, foram realizados quatro tipos diferentes de simulações. Na primeira, uma planta simples é simulada, e na segunda é simulado um processo de fase não mínima. Já na terceira simulação considera-se o caso em que o controlador ideal não pertence à classe de controle. Finalmente, no último processo simulado é considerado um processo variante no tempo.

7.1.1 Processo simples

Neste caso, primeiramente conduziram-se testes em uma planta simples, de primeira ordem, que possui a seguinte função de transferência

$$G(z) = \frac{0.5}{z - 0.9},$$

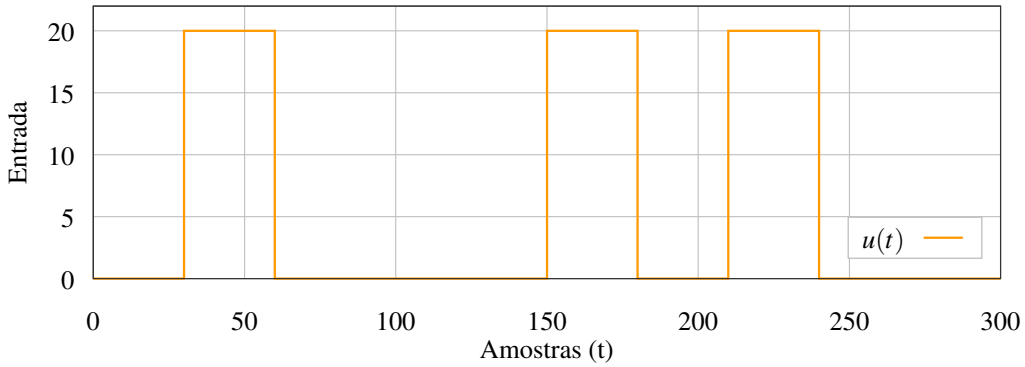
o tempo de acomodação para este processo é de aproximadamente 30 amostras, considerando-se tempo de acomodação para 95% do valor final.

Para melhorar a resposta do sistema escolheu-se um modelo de referência $M(z)$ que apresenta um tempo de acomodação mais rápido do que o tempo de acomodação do

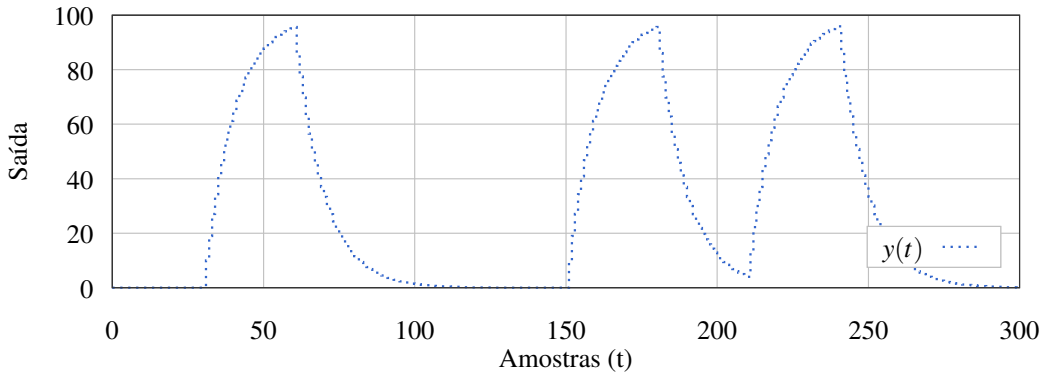
processo $G(z)$. Assim, optou-se pela seguinte função de transferência para o sistema em laço fechado

$$M(z) = \frac{0.4}{z - 0.6}.$$

No experimento em laço aberto, um sinal PRBS foi inserido na entrada do processo. Este sinal varia entre 0 e 20, e possui período de 40 amostras. Além disso, utilizou-se o período de amostragem de 1 s para a coleta dos dados. A Figura 25 apresenta os sinais coletados. Na Figura 25a é apresentada a entrada do experimento, $u(t)$, e a sua saída correspondente, $y(t)$, é apresentada na Figura 25b.



(a) Entrada do experimento em laço aberto.



(b) Saída do experimento em laço aberto.

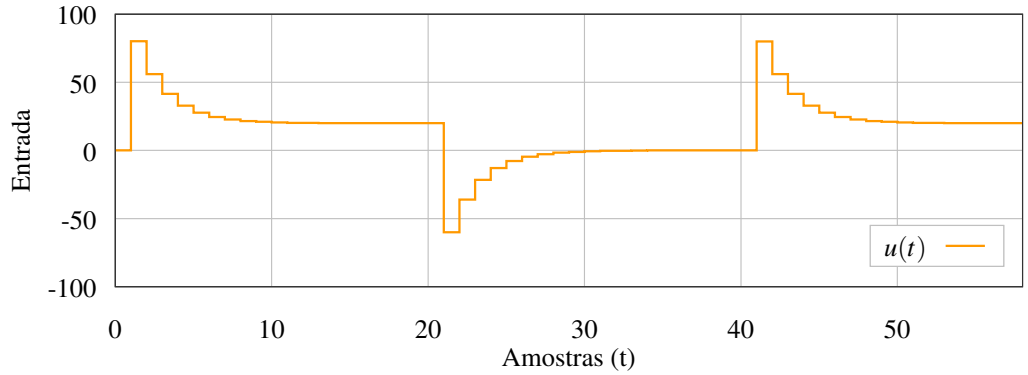
Figura 25: Resposta em laço aberto de um processo simples.

Devido à natureza da função de transferência da planta e do modelo de referência, optou-se por um controlador PI. Os parâmetros do controlador foram calculados durante a execução do experimento, ou seja, através do método VRFT recursivo com fator de esquecimento unitário. Desta forma, o controlador encontrado é dado por

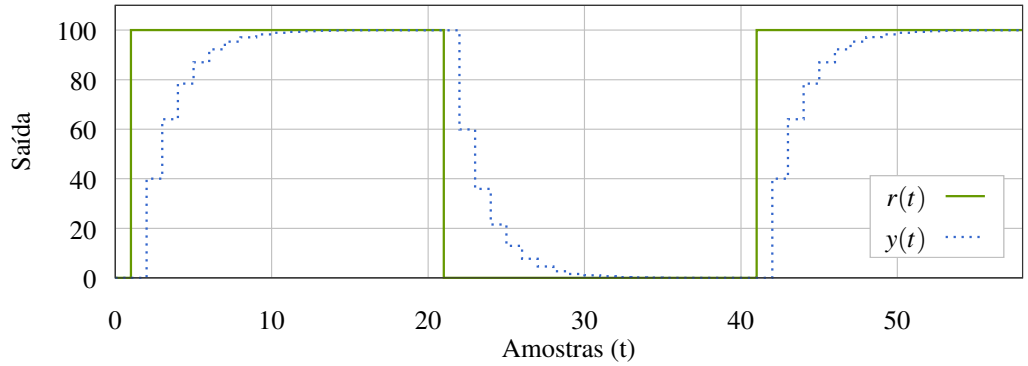
$$C(z,\rho) = \begin{bmatrix} 0.721 & 0.08 \end{bmatrix} \begin{bmatrix} 1 & \frac{z}{z-1} \end{bmatrix}^{\mathrm{T}}.$$

Para validar a resposta do processo com o controlador estimado, executou-se um experimento em laço fechado, onde os parâmetros calculados para o controlador foram configurados no processo. Neste experimento, utilizou-se uma entrada do tipo onda quadrada com período de 40 amostras, variando de 0 a 100. Os dados obtidos neste experimento são apresentados na Figura 26, o sinal de controle $u(t)$ pode ser visualizado

na Figura 26a, já os sinais de referência, $r(t)$, e de saída do processo, $y(t)$, podem ser vistos na Figura 26b.



(a) Entrada do processo em laço fechado.



(b) Saída do experimento em laço fechado.

Figura 26: Resposta em laço fechado de um processo simples.

Uma comparação entre a resposta obtida com o controlador, $y(t)$, e a resposta original do processo, $y_g(t)$, para uma entrada do tipo degrau unitário é apresentada na Figura 27. Cabe salientar que as respostas foram normalizadas para permitir uma melhor visualização das mesmas.

Pode-se observar, da Figura 27, que o propósito de controle foi alcançado, uma vez

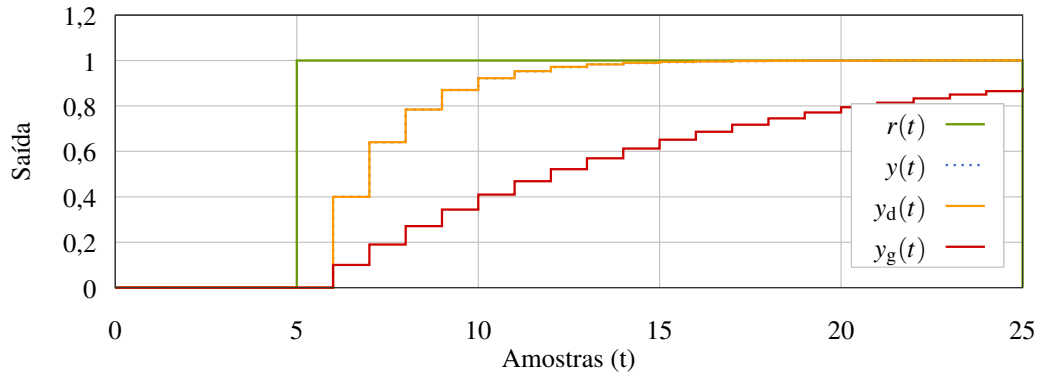


Figura 27: Comparação entre as saídas $y(t)$ e $y_g(t)$.

que a resposta obtida com o controlador, $y(t)$, é idêntica à resposta desejada, $y_d(t)$, o que é já era esperado, uma vez que o processo não é influenciado por ruído e o controlador ideal é um PI.

7.1.2 Processo com zeros de fase não mínima

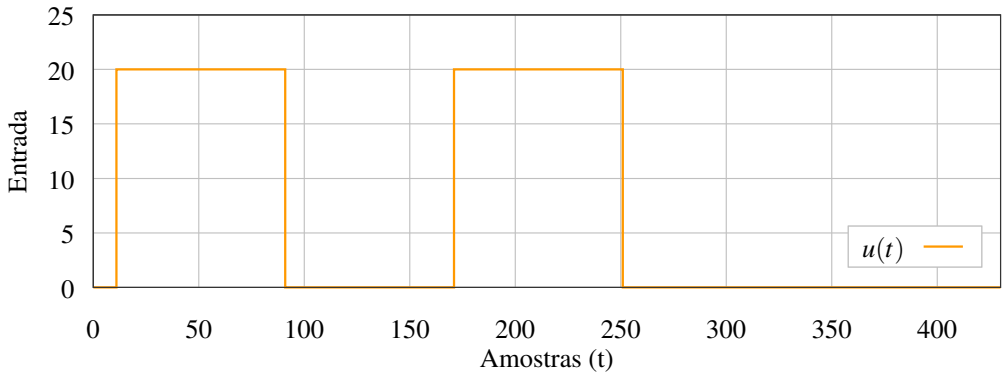
Para realização dos experimentos neste caso, uma planta de segunda ordem com um zero de fase não mínima foi simulada. A função de transferência desta planta é dada por

$$G(z) = \frac{z+1.2}{(z-0.95)(z-0.6)}.$$

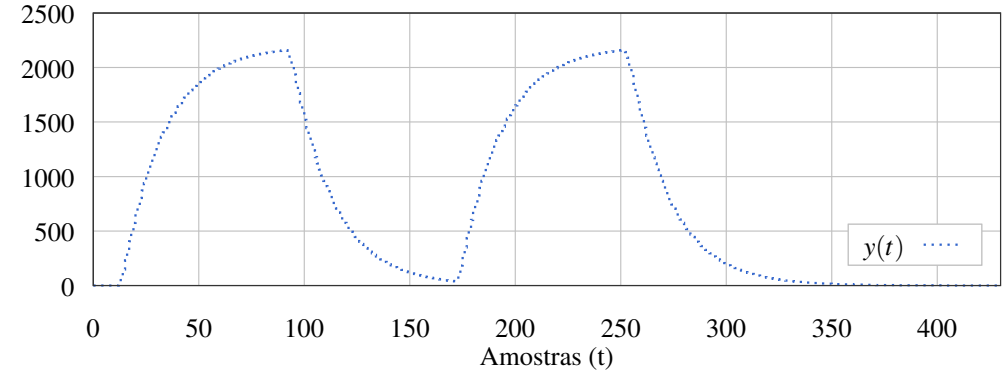
Uma vez que o zero de fase não mínima da planta é conhecido, utilizou-se a abordagem do método VRFT descrita na Seção 3.2. Desta forma, o modelo de referência $M(z)$ deve conter este zero, conforme mostrado abaixo

$$M(z) = \frac{0.1779\,(z+1.2)\,(z-0.7289)}{(z-0.8177)\,(z^2-0.6018z+0.1837)}.$$

Cabe salientar que escolheu-se $M(z)$ de modo a possibilitar a validação desta abordagem do método, assim sendo, o sentido físico desta função de transferência não foi levado em consideração. Desta forma, para a execução do experimento em laço aberto, no aplicativo foi informada a função de transferência $M(z)$ apresentada acima. A Figura 28 apresenta a entrada inserida no processo (Figura 28a) e a saída coletada (Figura 28b).



(a) Entrada do experimento em laço aberto.



(b) Saída do experimento em laço aberto.

Figura 28: Resposta em laço aberto do processo de fase não mínima.

Contudo, para a estimativa dos parâmetros do controlador o aplicativo utiliza uma função de transferência $M_a(z)$, que contém o inverso do zero de fase não mínima, de modo que o inverso de $M_a(z)$ possui somente polos estáveis, como pode ser visto abaixo

$$M_a(z) = \frac{-0.21348\,(z+0.8333)\,(z-0.7289)}{(z-0.8177)\,(z^2-0.6018z+0.1837)}.$$

Além disso, os sinais $u(t)$ e $y(t)$ foram filtrado através de um filtro dado por

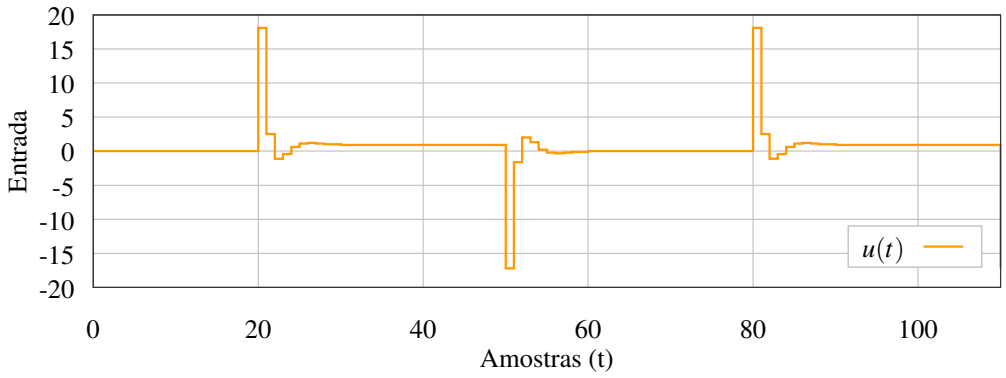
$$L_{ap}(z) = \frac{z+1.2}{-1.2z-1},$$

sendo este um filtro passa-tudo ponderado com magnitude igual a um. Como entrada para o experimento em laço aberto utilizou-se um sinal do tipo PRBS entre 0 e 20 com período de amostragem de 1 s.

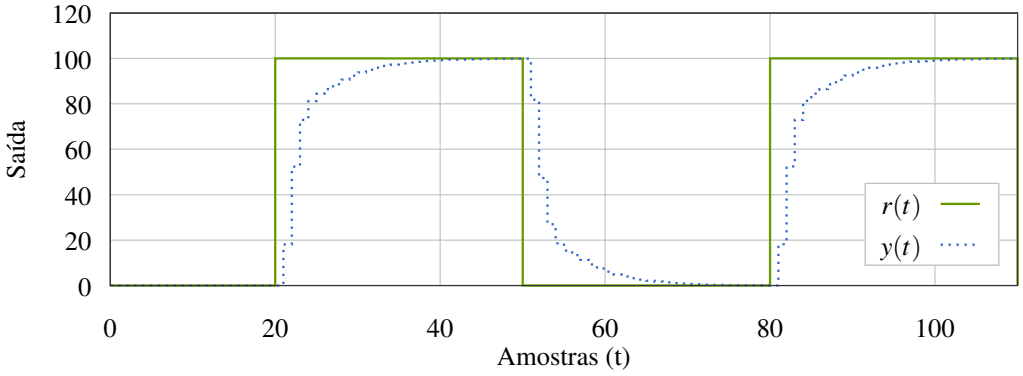
De posse dos dados coletados, estimou-se os parâmetros de um controlador PID através do método VRFT original. O controlador estimado $C(z,\rho)$ encontrado é dado pela seguinte equação

$$C(z,\rho) = \begin{bmatrix} 0.0538 & 0.0025 & 0.1252 \end{bmatrix} \begin{bmatrix} 1 & \dfrac{z}{z-1} & \dfrac{z-1}{z} \end{bmatrix}^{\mathrm{T}}.$$

Em seguida os parâmetros estimados foram empregados para configurar o controlador simulado e um experimento em laço fechado foi executado. Neste experimento, uma onda



(a) Entrada do experimento em laço fechado.



(b) Saída do experimento em laço fechado.

Figura 29: Resposta em laço fechado do processo de fase não mínima.

quadrada com período de 40 amostras foi utilizada como sinal de entrada e o período de amostragem foi mantido o mesmo. A resposta do sistema em laço fechado é mostrada na Figura 29, onde é possível observar o sinal de controle, $u(t)$, a entrada, $r(t)$, e a saída do experimento, $y(t)$.

Através da Figura 30 são apresentados os sinais $r(t)$ e $y(t)$ obtidos em laço fechado juntamente com os sinais $y_d(t)$ e $y_g(t)$, estes dois últimos são o sinal de saída desejado e o sinal de saída original da planta, respectivamente. A partir desta figura pode-se observar que os sinais $y(t)$ e $y_d(t)$ são praticamente iguais. Por conta disso, pode-se afirmar que o objetivo de controle foi alcançado com o controlador projetado.

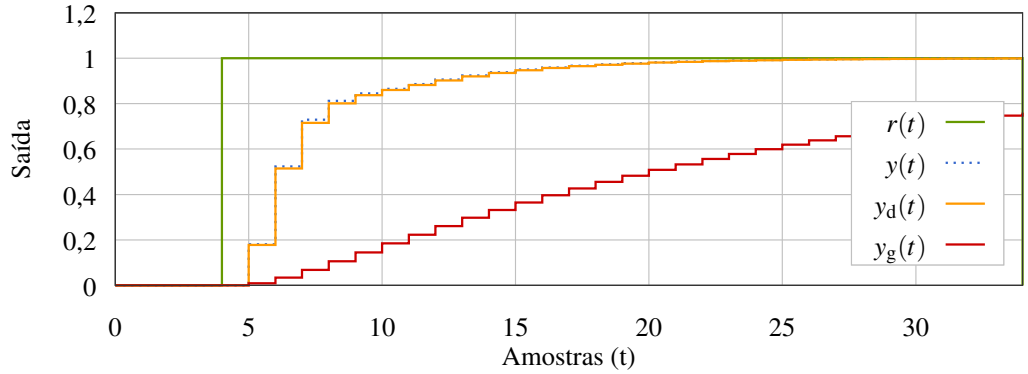


Figura 30: Comparação entre as saídas $y(t)$ e $y_g(t)$.

7.1.3 Processo com controlador fora da classe

Para simular um processo cujo controlador ideal está fora da classe de controladores considerada, primeiramente, escolheu-se a seguinte função de transferência para o processo

$$G(z) = \frac{z}{(z-0.95)\,(z^2+0.35z+0.9)},$$

sendo esta uma função de terceira ordem com tempo de acomodação de aproximadamente 60 amostras.

Em seguida, escolheu-se o modelo de referência $M(z)$ dado por

$$M(z) = \frac{0.1}{z-0.9},$$

que possui um tempo de acomodação de aproximadamente 30 amostras, considerando-se um tempo de acomodação de 95% do valor final.

Deseja-se utilizar um controlador PID com a seguinte estrutura

$$\bar{C}(z) = \begin{bmatrix} 1 & \dfrac{z}{z-1} & \dfrac{z-1}{z} \end{bmatrix}^{\mathrm{T}},$$

contudo, esta estrutura não é suficiente para que o objetivo seja alcançado exatamente, ou seja, para que o comportamento em laço fechado seja dado por $M(z)$.

Na sequência, realizou-se um experimento em laço aberto para coletar os dados necessários para estimar os parâmetros do controlador. Como entrada, foi inserido no processo um sinal do tipo degrau entre 0 e 20, a saída do processo quando aplicada esta entrada pode ser visualizada na Figura 31.

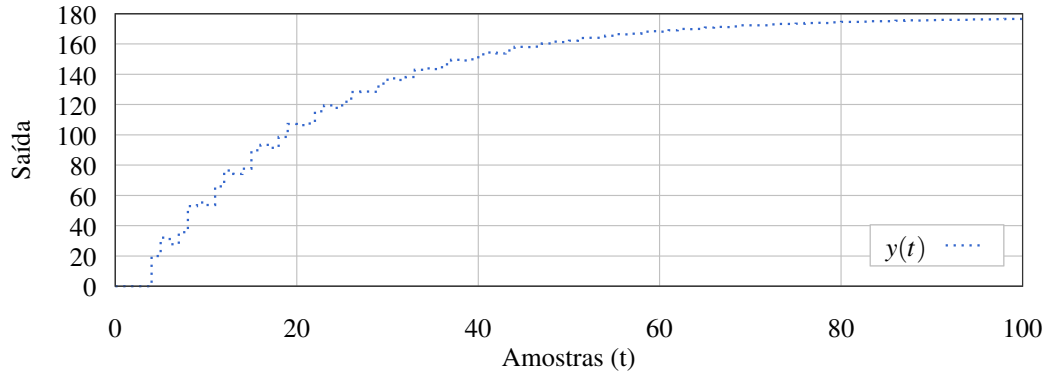
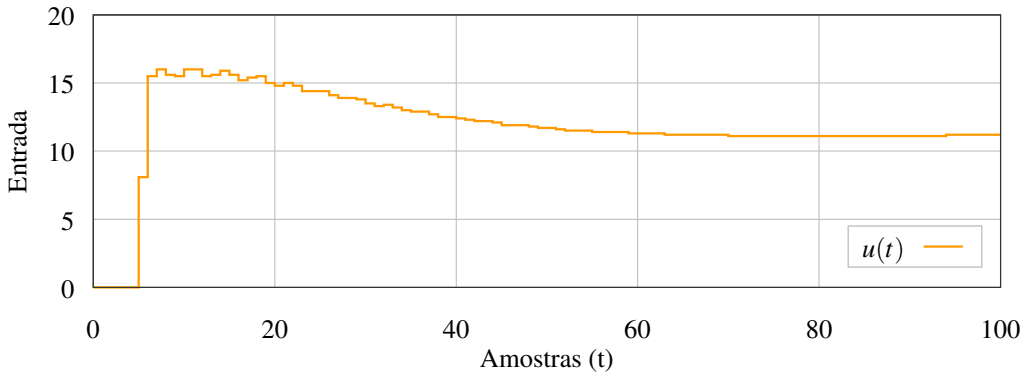


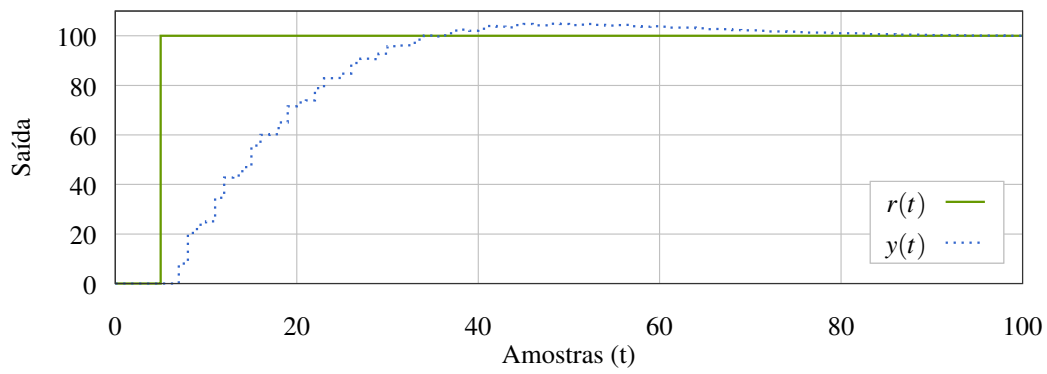
Figura 31: Saída do experimento em laço aberto.

Os parâmetros foram calculados através do método VRFT recursivo com fator de esquecimento unitário, inicialmente considerando-se que o controlador pertence à classe de controladores, obteve-se o seguinte vetor de ganhos $\hat{\rho}_1 = \begin{bmatrix} 0.132361 & 0.011552 & -0.063018 \end{bmatrix}$.

De posse dos ganhos calculados, os mesmos foram configurados no controlador simulado e um experimento em laço fechado foi executado, onde um sinal do tipo degrau entre 0 e 100 foi inserido na entrada do processo. A resposta obtida é apresentada na Figura 32, o sinal de controle, $u(t)$, é apresentado na Figura 32a, e os sinais de entrada, $r(t)$,



(a) Entrada do experimento em laço fechado.



(b) Saída do experimento em laço fechado.

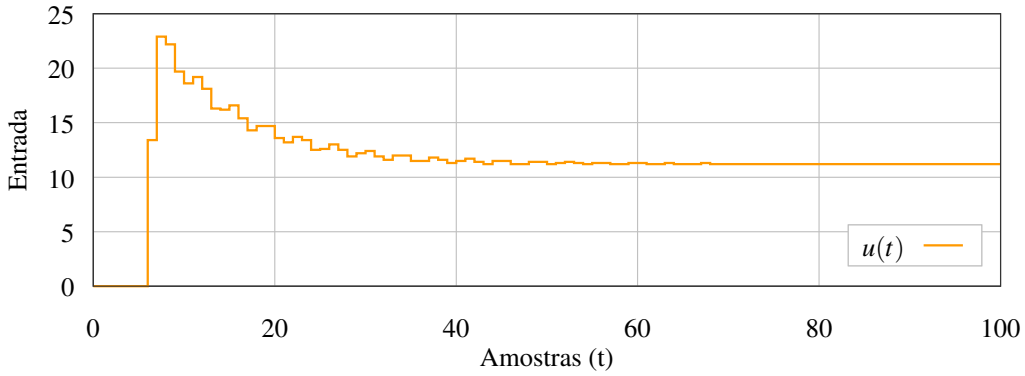
Figura 32: Resposta em laço fechado do processo com o controlador.

e saída, $y(t)$, são apresentados na Figura 32b. Pode-se observar que a resposta apresentou *overshoot*, o que não condiz com o modelo de referência, que é um modelo de primeira ordem.

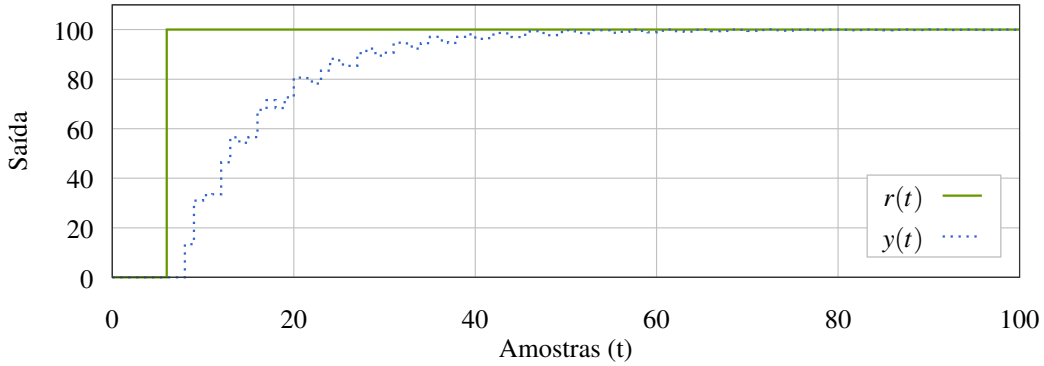
Portanto, outro experimento em laço aberto foi executado, com a mesma entrada utilizada anteriormente, para estimar os parâmetros do controlador, porém agora considerando-se o caso descasado, ou seja, passando-se os sinais pelo filtro $L(e^{j\omega})$, conforme apresentado na Seção 3.3. Os parâmetros também foram calculados através da abordagem recursiva do VRFT com fator de esquecimento unitário, e encontrou-se o seguinte conjunto de valores para os mesmos

$$\hat{\rho}_2 = \begin{bmatrix} 0.207849 & 0.011030 & -0.084404 \end{bmatrix}.$$

Um novo experimento em laço fechado foi conduzido, com o novo controlador estimado inserido no processo. Através da Figura 33 é apresentado o resultado deste experimento, através dos sinais $u(t)$, $r(t)$ e $y(t)$. Pode-se observar que a saída $y(t)$ não apresenta *overshoot*, conforme o esperado.



(a) Entrada do experimento em laço fechado.



(b) Saída do experimento em laço fechado.

Figura 33: Resposta em laço fechado do processo com o controlador estimado utilizando o filtro.

Uma comparação entre os sinais de saída obtidos com os controladores estimados e a resposta do processo sem os mesmos é apresentada na sequência. Na Figura 34 é apresentada a resposta do sistema, $y(t)$, obtida com o primeiro controlador $\hat{\rho}_1$, e na Figura 35 a resposta do sistema com o segundo controlador $\hat{\rho}_2$.

Além disso, para verificar numericamente o quão próximas as saídas obtidas, $y(t)$,

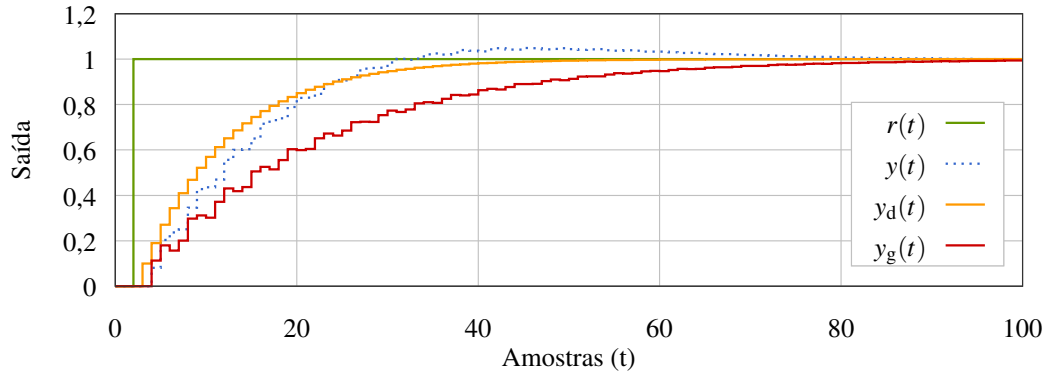


Figura 34: Comparação entre as saídas $y(t)$ e $y_g(t)$.

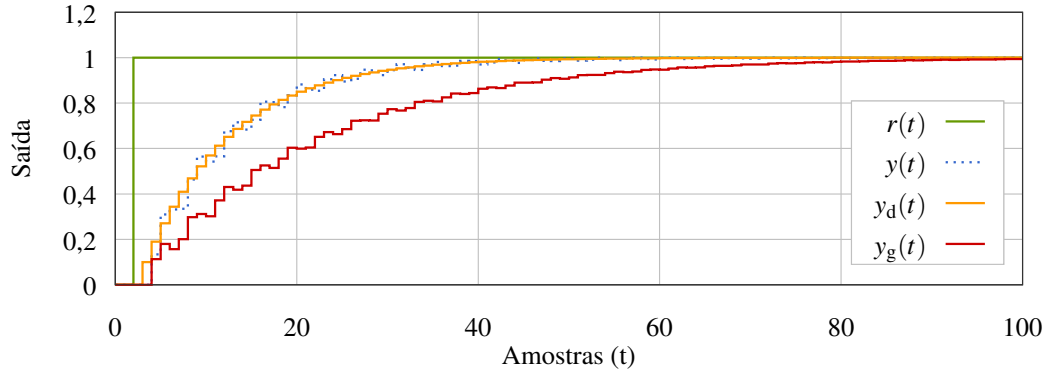


Figura 35: Comparação entre as saídas $y(t)$ e $y_g(t)$.

estavam da saída desejada, $y_d(t)$, empregou-se a estimativa de $\hat{J}_y(\hat{\rho})$ dada por

$$\hat{J}_y(\hat{\rho}) = \frac{1}{N}\sum_{t=1}^{N}\left(y(t,\hat{\rho}) - y_d(t)\right)^2.$$

Calculando-se $\hat{J}_y(\hat{\rho})$ das 100 amostras do sinal de saída em laço fechado para cada um dos experimentos apresentados nas figuras 34 e 35 e em seguida extraindo-se a raiz quadrada para que o valor tenha algum significado físico, obteve-se os seguintes resultados

$$\hat{J}_y(\hat{\rho}_1) = 4{,}3785\,^{\circ}\mathrm{C/amostra}$$

$$\hat{J}_y(\hat{\rho}_2) = 1{,}6029\,^{\circ}\mathrm{C/amostra}$$

Com base nos valores acima obtidos, verifica-se novamente que o controlador estimado considerando-se o caso descasado apresenta uma resposta melhor. Uma vez que $\hat{J}_y(\hat{\rho}_2)$ apresenta um valor menor comparado ao valor encontrado para $\hat{J}_y(\hat{\rho}_1)$.

7.1.4 Processo variante no tempo

Para testar a abordagem do método VRFT com fator de esquecimento foram conduzidas simulações de um sistema cuja função de transferência é variante no tempo. Para tanto, tal planta foi simulada utilizando-se a seguinte função

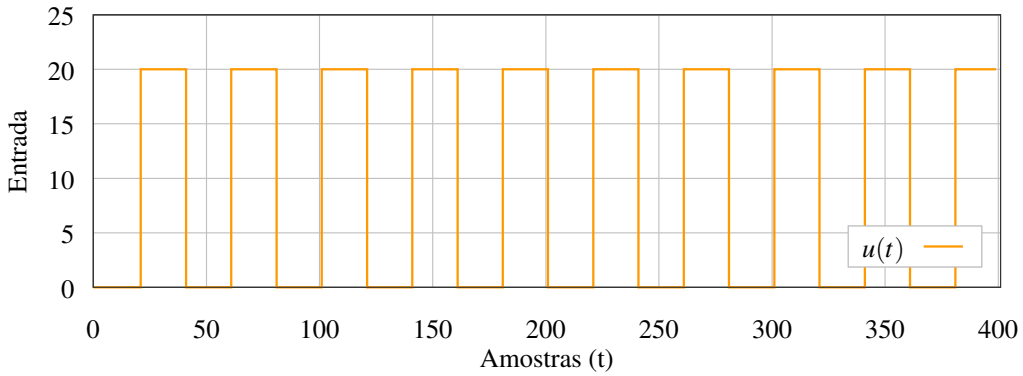
$$G(z) = \frac{0.1 + 0.1e^{-0.01t}}{z - 0.9},$$

onde t é o número da amostra. Esta função terá seu ganho alterado com o passar do tempo. O que acontece é que, em malha fechada, alterando-se o ganho da planta é alterada a localização dos polos da função de transferência resultante, fazendo com que a resposta do sistema em malha fechada mude com o decorrer do tempo.

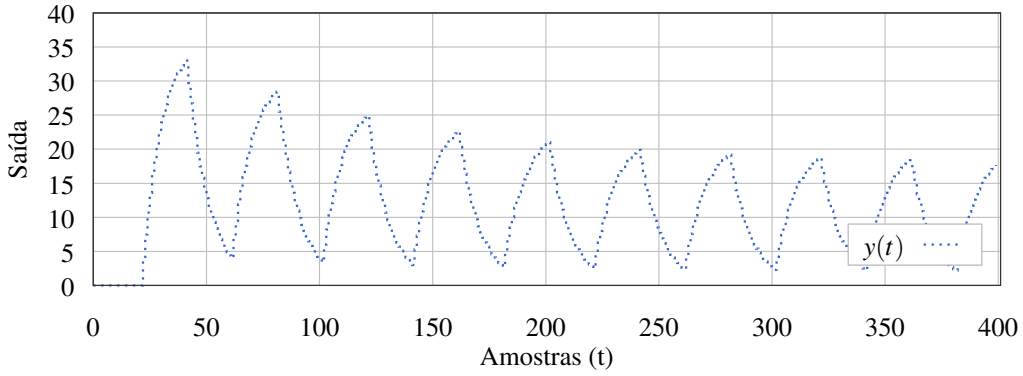
O comportamento desejado para o sistema em malha fechada é dado por

$$M(z) = \frac{0.2}{z - 0.8}.$$

Com o objetivo de coletar os dados necessários para o cálculo dos parâmetros foi realizado um experimento em laço aberto. Para execução deste experimento foi definido um período de amostragem de 1 s, e um sinal do tipo onda quadrada com período de 40 amostras foi aplicado na entrada do processo. Foram coletadas 400 amostras dos sinais de entrada, $u(t)$, e saída, $y(t)$, que são apresentados na Figura 36.



(a) Entrada do experimento em laço aberto.



(b) Saída do experimento em laço aberto.

Figura 36: Resposta em laço aberto do processo variante no tempo.

Primeiramente, o cálculo dos parâmetros do controlador foi realizado através do método VRFT recursivo com fator de esquecimento unitário. Neste caso, optou-se por um controlador PI, que teve seus ganhos identificados conforme a seguinte equação:

$$C(z, \rho_1) = \begin{bmatrix} 1.340304 & 0.147331 \end{bmatrix} \begin{bmatrix} 1 & \frac{z}{z-1} \end{bmatrix}^{\mathrm{T}}.$$

Como os parâmetros foram estimados utilizando-se $\lambda = 1$, todas as amostras contribuíram igualmente para o cálculo do seu valor final. Assim sendo, para visualizar o

deslocamento dos polos do processo em malha fechada, utilizou-se o controlador acima mencionado no sistema em laço fechado. Através da Figura 37 pode-se observar que existe uma variação dos polos com o decorrer do tempo. O polo do sistema em malha fechada inicia em aproximadamente 0,7 e após 400 amostras ele se encontra próximo de 0,85. Isto indica que em algum momento durante o experimento o sistema se comportou exatamente como desejado (com o polo localizado em 0,8), porém com o passar do tempo este comportamento não é mais alcançado.

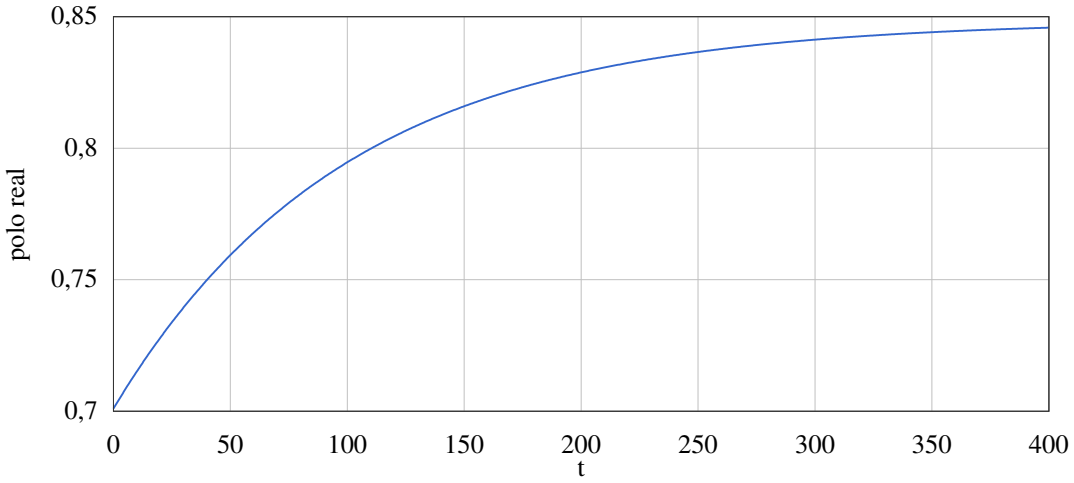


Figura 37: Localização do polo.

Os parâmetros estimados foram inseridos no controlador e um experimento em laço fechado foi conduzido. Neste experimento, foi aplicada uma entrada do tipo onda quadrada com período de 40 amostras variando de 0 a 100, e foram coletadas 400 amostras. Uma comparação entre a resposta do sistema com o controlador e a resposta desejada pode ser visualizada na Figura 38, onde são apresentados os sinais de referência $r(t)$, de saída $y(t)$, e a saída desejada $y_d(t)$ das últimas 25 amostras do experimento. Já na Figura 39 são apresentados os dados do experimento completo.

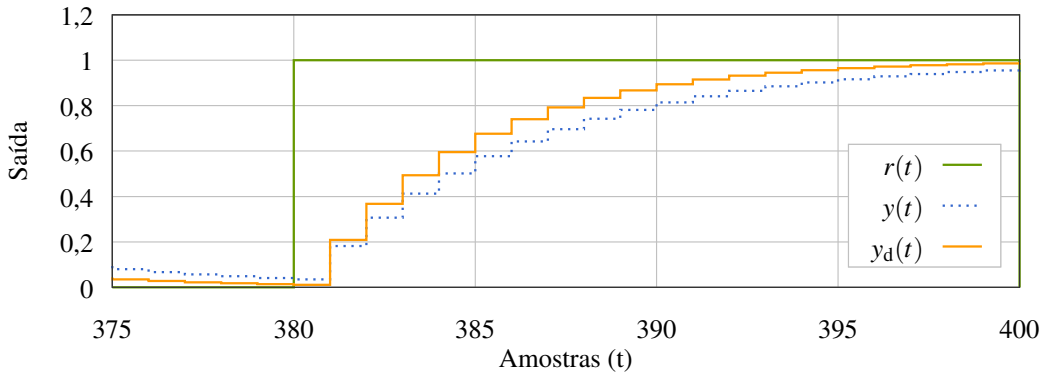
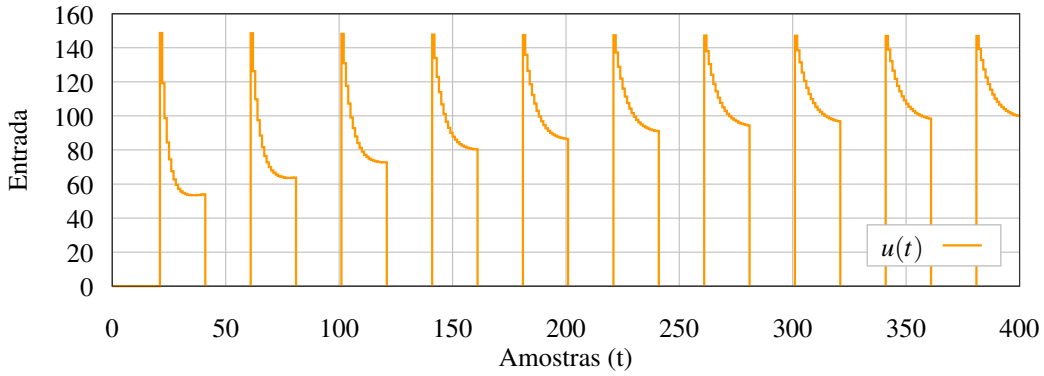
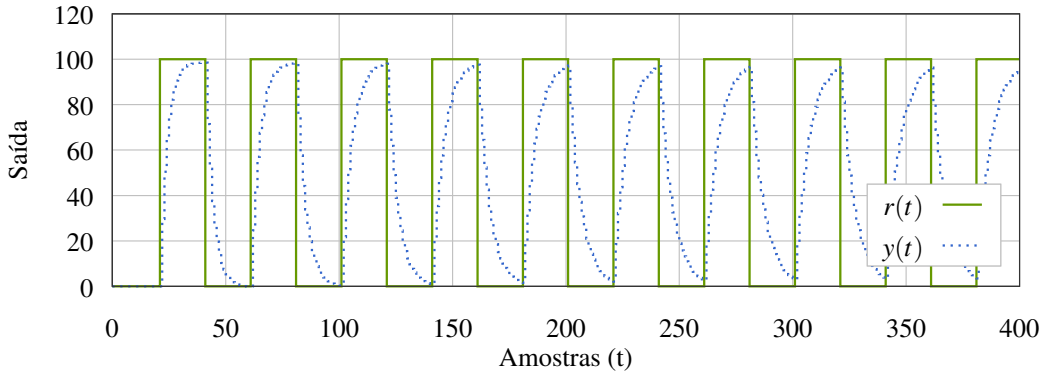


Figura 38: Comparação entre $y(t)$ e $y_d(t)$.

Na sequência, um novo experimento em laço aberto foi executado, desta vez com fator de esquecimento $\lambda = 0.99$. Para a realização do experimento, a mesma entrada do tipo onda quadrada foi utilizada, e foram coletadas 400 amostras. O ajuste dos parâmetros do



(a) Entrada do experimento em laço fechado.



(b) Saída do experimento em laço fechado.

Figura 39: Resposta em laço fechado do processo variante no tempo.

controlador foi realizado através da abordagem recursiva do método VRFT com o fator de esquecimento mencionado acima. Como resultado foi estimado o seguinte controlador:

$$C(z,\rho_2) = \begin{bmatrix} 1.615305 & 0.180410 \end{bmatrix} \begin{bmatrix} 1 & \dfrac{z}{z-1} \end{bmatrix}^{\mathrm{T}}.$$

Um novo experimento em laço fechado foi conduzido, onde o controlador estimado foi ajustado no processo. Para realização do experimento, o mesmo sinal de entrada foi inserido no processo, e novamente 400 amostras foram coletadas.

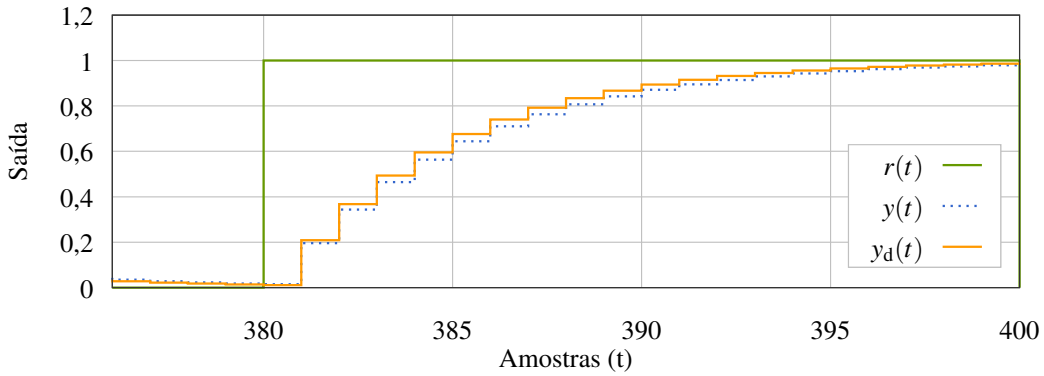
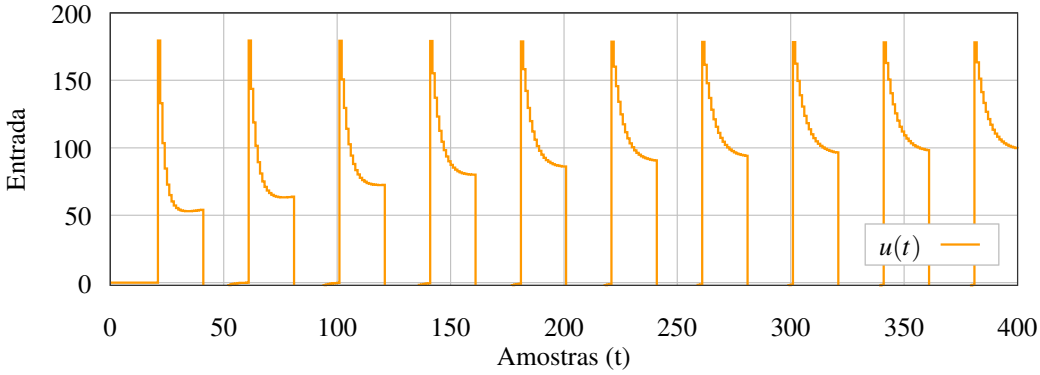
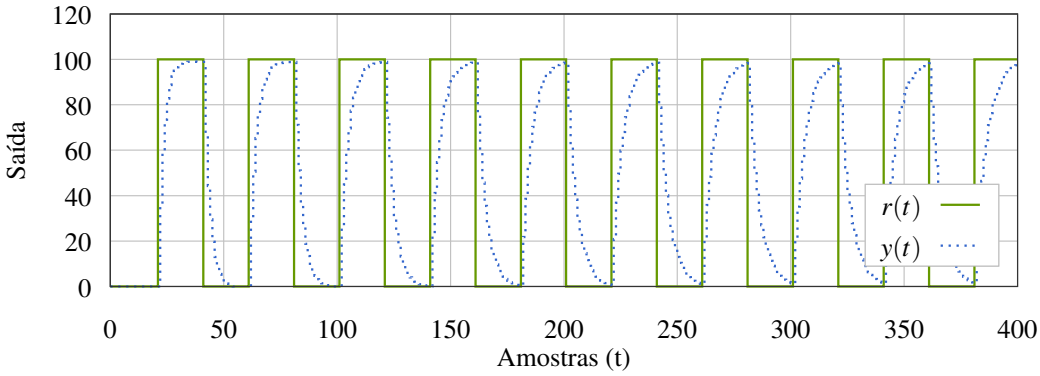


Figura 40: Comparação entre $y(t)$ e $y_d(t)$.

Para verificar o quão próxima a saída obtida com o controlador estava da saída desejada elaborou-se o gráfico da Figura 40, onde são apresentados os sinais coletados das últimas 25 amostras. O conjunto completo de dados coletados é disponibilizado na Figura 41.



(a) Entrada do experimento em laço fechado.



(b) Saída do experimento em laço fechado.

Figura 41: Resposta em laço fechado do processo variante no tempo.

Comparando-se as respostas obtidas com o controlador estimado com fator de esquecimento unitário (Figura 38) e com fator de esquecimento $\lambda = 0.99$ (Figura 40), observa-se que esta última foi a que apresentou uma resposta mais próxima da saída desejada. Além disso, calculou-se a estimativa $\hat{J}_y(\hat{\rho})$ para as últimas 25 amostras dos dois sinais, pois as últimas amostras do experimento representam um comportamento mais próximo do comportamento que o sistema teria após um longo período de operação. Desta forma, chegou-se a $\hat{J}_y(\hat{\rho}_1) = 6{,}9435\,^{\circ}\mathrm{C}/\text{amostra}$, e $\hat{J}_y(\hat{\rho}_2) = 2{,}1091\,^{\circ}\mathrm{C}/\text{amostra}$, o que comprova o que foi mencionado anteriormente.

7.2 Processo térmico

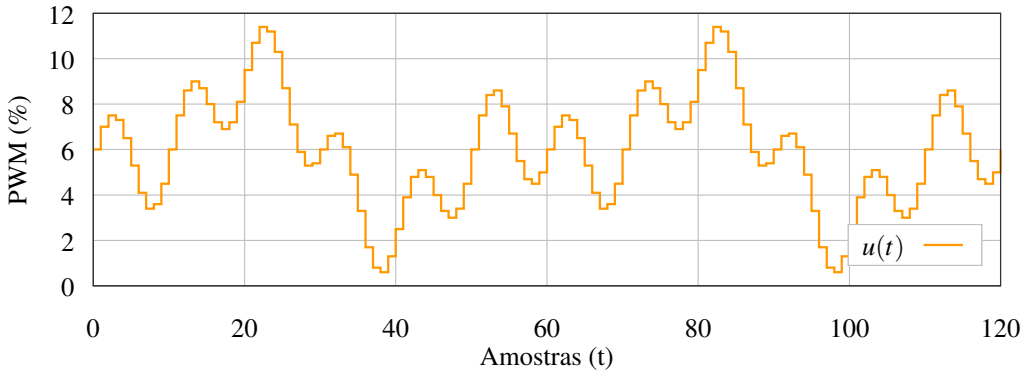
Na segunda etapa de testes foram realizados experimentos em um processo real. O processo em questão é um processo térmico composto por um elemento aquecedor, um sensor de temperatura, e um controlador PID comercial. Este processo é conectado ao gateway através de uma porta serial RS-485.

O elemento aquecedor é um resistor que dissipa calor através do efeito Joule. O sensor de temperatura é um termopar tipo K que converte a temperatura medida em uma diferença de tensão. A tensão no aquecedor é controlada por um sinal de pulso PWM (*Pulse Width*

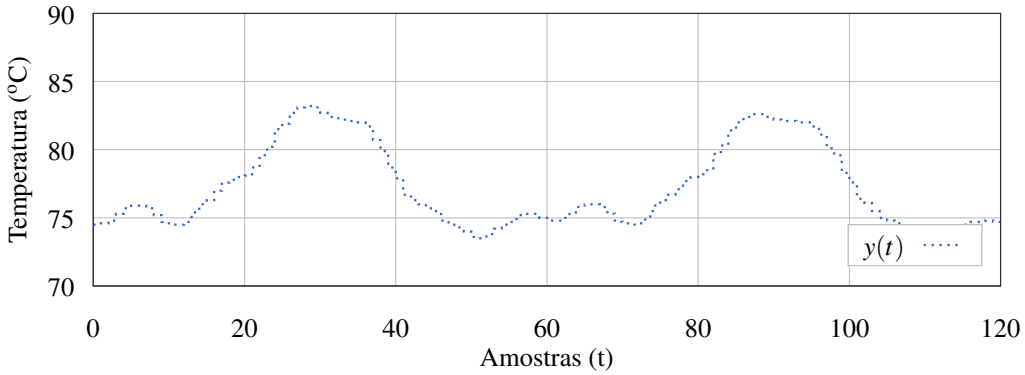
Modulated) com período de 3 s, e a entrada do processo é uma porcentagem do tamanho do pulso do sinal. A saída do processo é a tensão medida pelo termopar convertida para temperatura em graus Celsius.

Para o ajuste dos parâmetros foram realizados dois experimentos, um primeiro experimento em laço aberto para coletar os dados, e um segundo em laço fechado com os ganhos do controlador já adicionados ao processo. No primeiro experimento, executado em laço aberto, uma entrada sinusoidal arbitrária foi inserida no processo com período de amostragem de 3 s. A forma de onda da entrada pode ser visualizada através da Figura 42a, sendo descrita pela seguinte equação

$$u(t) = 6 + 2\sin\left(\frac{2\pi t}{60}\right) - 2\sin\left(\frac{2\pi t}{30}\right) + 2\sin\left(\frac{2\pi t}{10}\right).$$



(a) Entrada do experimento em laço aberto.



(b) Saída do experimento em laço aberto.

Figura 42: Sinais do experimento em laço aberto realizado no processo térmico.

A saída amostrada do processo, com a entrada sinusoidal, é apresentada na Figura 42b, e na Figura 43 é apresentada a tela do aplicativo com os mesmos dados de entrada e saída do experimento em laço aberto.

O modelo de referência escolhido para este processo é igual ao utilizado em Bazanella, Campestrini e Eckhard (2012) para o mesmo processo, sendo dado por

$$M(z) = \frac{0.1}{z - 0.9}.$$

Para o ajuste dos parâmetros do controlador utilizou-se a abordagem recursiva do método VRFT com fator de esquecimento unitário. Os parâmetros foram calculados

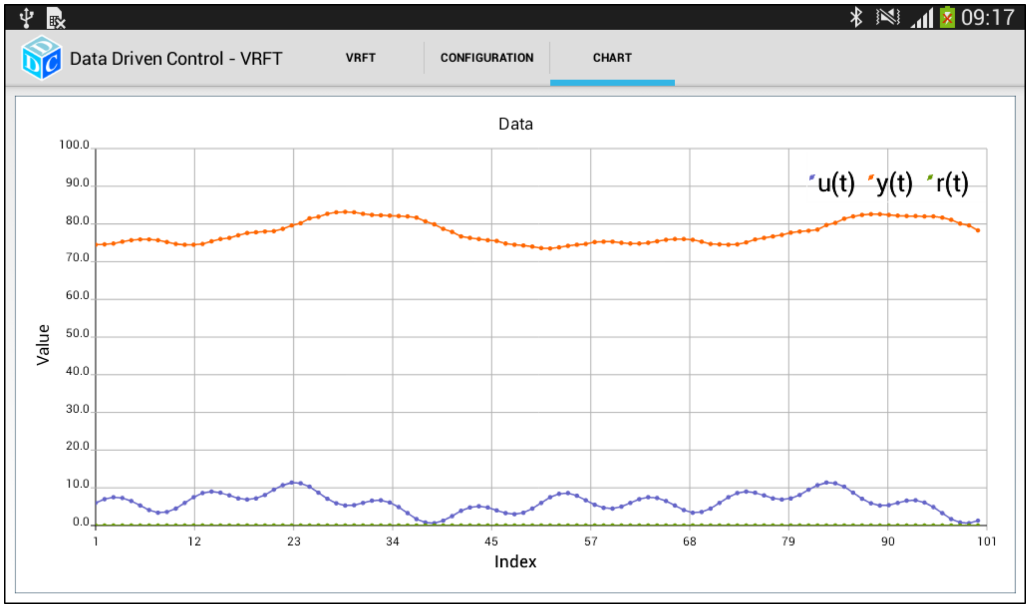


Figura 43: Tela do aplicativo com os dados do experimento em laço aberto.

utilizando-se as 120 amostras apresentadas anteriormente (Figura 42a e Figura 42b), obtendo-se o seguinte controlador

$$C(z,\rho) = \begin{bmatrix} 0.51025 & 0.03191 & 0.57422 \end{bmatrix} \begin{bmatrix} 1 & \dfrac{z}{z-1} & \dfrac{z-1}{z} \end{bmatrix}^{\mathrm{T}}.$$

Os parâmetros calculados foram ajustados no controlador conectado ao processo, e um experimento em laço fechado foi realizado. Neste experimento foi utilizada como entrada um degrau com valores de 80 °C a 100 °C. Através da Figura 44 pode-se observar o sinal de entrada $r(t)$, o sinal de saída $y(t)$ coletado e armazenado no arquivo de dados, e o sinal de saída desejado $y_d(t)$ que é a resposta simulada da saída do modelo de referência.

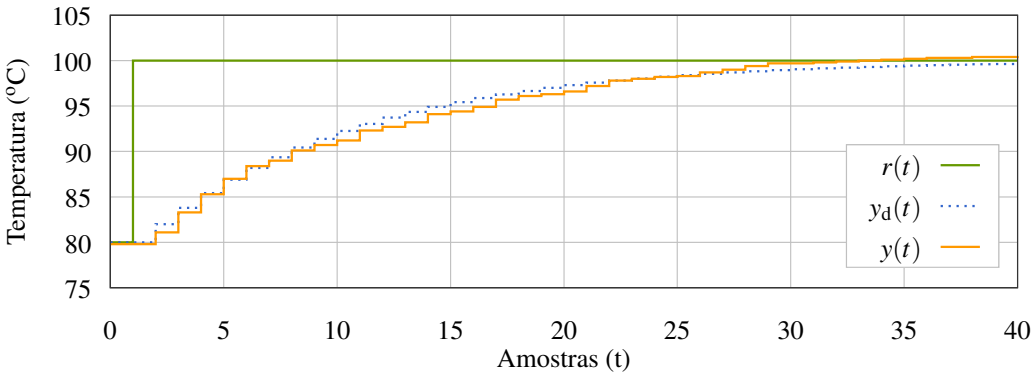


Figura 44: Sinais do experimento em laço fechado.

Para verificar o quão próximo a saída obtida $y(t)$ estava da saída desejada $y_d(t)$ empregou-se a estimativa de $\hat{J}_y(\hat{\rho})$. Calculando-se $\hat{J}_y(\hat{\rho})$ das 100 amostras do sinal de saída em laço fechado e extraindo-se a raiz quadrada do resultado, para que o resultado tenha a mesma unidade do sinal de saída, obtêm-se o seguinte valor

$$\hat{J}_y(\hat{\rho}) = 0{,}53\,^{\circ}\mathrm{C/amostra}$$

sendo este um valor bem baixo para a função custo, que condiz com a distância visual entre as curvas na Figura 44.

CONCLUSÃO

Neste trabalho foi apresentada uma proposta para facilitar a utilização do método VRFT no ajuste dos parâmetros de um controlador. Esta solução foi composta por um aplicativo Android, um gateway, e um protocolo de comunicação simples. No aplicativo Android implementou-se o método VRFT e duas novas variações deste método. A primeira é uma abordagem recursiva do mesmo para estimar os parâmetros de um controlador a cada nova amostra, sem a necessidade do armazenamento de todos os dados ou da espera do término do experimento. Já a segunda apresenta uma técnica que emprega um fator de esquecimento, possibilitando a sua utilização em sistemas variantes no tempo. Além disso, outras funcionalidades foram implementadas, como o armazenamento dos dados coletados em um arquivo e a visualização *on-line* dos mesmos. Um gateway Bluetooth-Modbus também foi desenvolvido para possibilitar a comunicação entre a aplicação e o processo a ser controlado. Desta forma, o gateway é responsável por converter as mensagens enviadas pelo aplicativo em sequências de comandos para o controlador, e vice-versa.

Para validar a solução proposta, um estudo de caso foi realizado no qual vários experimentos foram conduzidos, em processos simulados no sistema do gateway, e em um processo térmico real. Os experimentos foram conduzidos em plantas simuladas com o objetivo de validar todas as funcionalidades implementadas no aplicativo. Já o experimento com o processo térmico foi realizado com vistas a verificar o correto funcionamento da solução apresentada em conjunto com um sistema real onde os sinais não são completamente livres de ruído e a temporização é relevante. Ao final dos experimentos, pôde-se observar através dos resultados obtidos que a solução desenvolvida atendeu aos requisitos, apresentando resultados conforme o esperado tanto nas simulações quanto no processo real.

Apesar disto ainda existem algumas questões em aberto. Uma delas é a implementação do método VRFT com variável instrumental (VI), para lidar com sistemas que apresentam ruído significativo. Neste caso, algumas alterações precisam ser realizadas no aplicativo para possibilitar a realização de dois experimentos. A abordagem original do método VRFT com VI já foi apresentada na literatura, assim, acredita-se que sua implementação no aplicativo não apresentaria grandes dificuldades. Porém, a implementação do VRFT recursivo com VI, acredita-se, demandaria um maior esforço, pois seria necessário determinar uma VI compatível com este tipo de abordagem. O VRFT flexível é ainda uma outra variação que poderia ser implementada, utilizado para sistemas que apresentam zeros de fase não mínima e estes não são conhecidos. Um estudo mais aprofundado deste algoritmo seria necessário para viabilizar a sua implementação.

Além disso, outras melhorias pequenas poderiam ser realizadas tanto no gateway quanto no protocolo desenvolvido. A aplicação do gateway poderia ser ajustada para realizar a comunicação com outros protocolos industriais, além do Modbus. Ademais, o

protocolo poderia ser aprimorado, adicionando-se ao mesmo um algoritmo mais eficaz na correção de falhas, e para validação do estado da conexão poderia-se utilizar mensagens de *keep alive*, por exemplo.

REFERÊNCIAS

AGRAWAL, D.; ZENG, Q.-A. **Introduction to wireless and mobile systems**. 3. ed. Stamford: Cengage Learning, 2011.

AGUIRRE, L. A. **Introdução à identificação de sistemas**: Técnicas lineares e não-lineares aplicadas a sistemas reais. 3. ed. Belo Horizonte: Editora UFMG, 2004.

ALEXANDER, T. J. An implementation of mobile control room environment in android platform for industrial applications. In: INTERNATIONAL CONFERENCE ON CIRCUIT, POWER AND COMPUTING TECHNOLOGIES (ICCPCT), 2015, Thuckalay. **Proceedings...** Piscataway: IEEE, 2015. p. 1–4. Disponível em: <https://doi.org/10.1109/iccpct.2015.7159269>. Acesso em: 23 jan. 2017.

ARRIETA, O. et al. Development of a mobile application for robust tuning of one-and two-degree-of-freedom PI and PID controllers. **IFAC-PapersOnLine**, Amsterdam, v. 48, n. 29, p. 76–81, 2015. Trabalho apresentado no IFAC Workshop on Internet Based Control Education IBCE15. Disponível em: <http://dx.doi.org/10.1016/j.ifacol.2015.11.216>. Acesso em: 23 jan. 2017.

BAZANELLA, A. S.; CAMPESTRINI, L.; ECKHARD, D. **Data-Driven Controller Design**: The H2 approach. Dordrecht: Springer, 2012. (Communications and Control Engineering). Disponível em: <https://doi.org/10.1007/978-94-007-2300-9>. Acesso em: 23 jan. 2017.

BENSKY, A. **Short-range wireless communication**: Fundamentals of RF system design and application. 2. ed. Burlington: Newnes, 2004. (Communications Engineering Series). Disponível em: <http://dx.doi.org/10.1016/B978-075067782-0/50018-2>. Acesso em: 23 jan. 2017.

BHAGWAT, P. Bluetooth: technology for short-range wireless apps. **IEEE Internet Computing**, Piscataway, v. 5, n. 3, p. 96–103, 2001. Disponível em: <https://doi.org/10.1109/4236.935183>. Acesso em: 23 jan. 2017.

BLUETOOTH SPECIAL INTEREST GROUP. **Specification of the Bluetooth® system**. Kirkland, 2014. 2772 p. Versão 4.2. Disponível em: <https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439>. Acesso em: 24 jan. 2017.

CAMPESTRINI, L.; GEVERS, M.; BAZANELLA, A. Virtual reference feedback tuning for non minimum phase plants. In: EUROPEAN CONTROL CONFERENCE (ECC), 2009, Budapest. **Proceedings...** Piscataway: IEEE, 2009. p. 1955–1960. Disponível em: <http://ieeexplore.ieee.org/document/7074690/>. Acesso em: 23 jan. 2017.

CAMPI, M. C.; LECCHINI, A.; SAVARESI, S. M. Virtual reference feedback tuning: a direct method for the design of feedback controllers. **Automatica**, Amsterdam, v. 38, n. 8, p. 1337–1346, 2002. Disponível em: <http://dx.doi.org/10.1016/S0005-1098(02)00032-8>. Acesso em: 23 jan. 2017.

CAMPI, M. C.; LECCHINI, A.; SAVARESI, S. M. An application of the virtual reference feedback tuning method to a benchmark problem. **European Journal of Control**, Amsterdam, v. 9, n. 1, p. 66–76, 2003. Disponível em: <http://dx.doi.org/10.3166/ejc.9.66-76>. Acesso em: 23 jan. 2017.

CAMPI, M. C.; SAVARESI, S. M. Direct nonlinear control design: the virtual reference feedback tuning (VRFT) approach. **IEEE Transactions on Automatic Control**, Piscataway, v. 51, n. 1, p. 14–27, 2006. Disponível em: <https://doi.org/10.1109/tac.2005.861689>. Acesso em: 23 jan. 2017.

CLAUDI, A. et al. Marvin: mobile autonomous robot for video surveillance networks. In: UKSIM/AMSS EUROPEAN SYMPOSIUM ON COMPUTER MODELING AND SIMULATION (EMS), 2012, Malta. **Proceedings...** Piscataway: IEEE, 2013. p. 21–26. Disponível em: <https://doi.org/10.1109/ems.2012.37>. Acesso em: 23 jan. 2017.

CORLETA, A. et al. Data-driven control design applied to uninterruptible power supplies. In: IEEE CONFERENCE ON CONTROL APPLICATIONS (CCA), 2016, Buenos Aires. **Proceedings...** Piscataway: IEEE, 2016. p. 1312–1317. Disponível em: <https://doi.org/10.1109/cca.2016.7587988>. Acesso em: 23 jan. 2017.

FARAGHER, R. M. et al. Captain buzz: An all-smartphone autonomous delta-wing drone. In: WORKSHOP ON MICRO AERIAL VEHICLE NETWORKS, SYSTEMS, AND APPLICATIONS FOR CIVILIAN USE, 1., 2015, Florence. **Proceedings...** New York: ACM, 2015. p. 27–32. Disponível em: <https://doi.org/10.1145/2750675.2750682>. Acesso em: 23 jan. 2017.

FILHO, R. S. et al. Application of virtual reference feedback tuning to a non-minimum phase pilot plant. In: IEEE CONFERENCE ON CONTROL APPLICATIONS (CCA), 2016, Buenos Aires. **Proceedings...** Piscataway: IEEE, 2016. p. 1318–1323. Disponível em: <https://doi.org/10.1109/cca.2016.7587989>. Acesso em: 23 jan. 2017.

HEDREA, E.-L.; RADAC, M.-B.; PRECUP, R.-E. Virtual reference feedback tuning for position control of a twin rotor aerodynamic system. In: IEEE INTERNATIONAL SYMPOSIUM ON APPLIED COMPUTATIONAL INTELLIGENCE AND INFORMATICS (SACI), 11., 2016, Timisoara. **Proceedings...** Piscataway: IEEE, 2016. p. 57–62. Disponível em: <https://doi.org/10.1109/saci.2016.7507431>. Acesso em: 23 jan. 2017.

HUANG, A.; RUDOLPH, L. **Bluetooth essentials for programmers**. New York: Cambridge University Press, 2007. Disponível em: <https://doi.org/10.1017/cbo9780511546976.001>. Acesso em: 23 jan. 2017.

LECCHINI, A.; CAMPI, M. C.; SAVARESI, S. M. Virtual reference feedback tuning for two degree of freedom controllers. **International Journal of Adaptive Control Signal Process**, Budapest, v. 16, p. 355–371, 2002. Disponível em: <https://doi.org/10.1002/acs.711>. Acesso em: 23 jan. 2017.

LEE, W. M. **Beginning Android 4 Application Development**. Indianapolis: Wrox, 2012. Disponível em: <http://www.wiley.com/WileyCDA/WileyTitle/productCd-1118199545.html>. Acesso em: 23 jan. 2017.

MACKAY, S. et al. **Practical industrial data networks**: design, installation and troubleshooting. Burlington: Newnes, 2004. Disponível em: <https://doi.org/10.1016/b978-0-7506-5807-2.x5024-9>. Acesso em: 23 jan. 2017.

MODBUS ORGANIZATION. **MODBUS over Serial Line Specification and Implementation Guide**. Hopkinton, 2006. 44 p. Versão 1.02. Disponível em: <http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf>. Acesso em: 24 jan. 2017.

MODBUS ORGANIZATION. **MODBUS Application Protocol Specification**. Hopkinton, 2012. 50 p. Versão 1.1b3. Disponível em: <http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf>. Acesso em: 24 jan. 2017.

NAKAMOTO, M. An application of the virtual reference feedback tuning for an mimo process. In: SICE ANNUAL CONFERENCE, 2004, Sapporo. **Proceedings...** Piscataway: IEEE, 2005. p. 2208–2213. Disponível em: <http://ieeexplore.ieee.org/document/1491812>. Acesso em: 23 jan. 2017.

PRATUMSUWAN, P.; CHANAISAWAN, K. Applying the smartphone on position tracking control of servo pneumatic systems. In: INTERNATIONAL CONFERENCE ON ELECTRICAL ENGINEERING/ELECTRONICS, COMPUTER, TELECOMMUNICATIONS AND INFORMATION TECHNOLOGY (ECTI-CON), 13., 2016, Chiang Mai. **Proceedings...** Piscataway: IEEE, 2016. p. 1–5. Disponível em: <https://doi.org/10.1109/ECTICon.2016.7561241>. Acesso em: 23 jan. 2017.

PREVIDI, F. et al. Virtual reference feedback tuning (VRFT) design of cascade control systems with application to an electro-hydrostatic actuator. **IFAC Proceedings Volumes**, Amsterdam, v. 43, n. 18, p. 626–632, 2016. Trabalho apresentado no 5th IFAC Symposium on Mechatronic Systems. Disponível em: <http://dx.doi.org/10.3182/20100913-3-US-2015.00033>. Acesso em: 23 jan. 2017.

PREVIDI, F. et al. Virtual reference feedback tuning (VRFT) of velocity controller in self-balancing industrial manual manipulators. In: AMERICAN CONTROL CONFERENCE, 2010, Baltimore. **Proceedings...** Piscataway: IEEE, 2010. p. 1956–1961. Disponível em: <https://doi.org/10.1109/ACC.2010.5531358>. Acesso em: 23 jan. 2017.

ROJAS, J. D. et al. Application of multivariate virtual reference feedback tuning for wastewater treatment plant control. **Control Engineering Practice**, Amsterdam, v. 20, n. 5, p. 499–510, 2012. Disponível em: <http://dx.doi.org/10.1016/j.conengprac.2012.01.004>. Acesso em: 23 jan. 2017.

ROJAS, J. D.; MORILLA, F.; VILANOVA, R. Multivariable PI control for a boiler plant benchmark using the virtual reference feedback tuning. **IFAC Proceedings Volumes**, Amsterdam, v. 45, n. 3, p. 376–381, 2016. Trabalho apresentado na 2nd IFAC Conference on Advances in PID Control. Disponível em: <http://dx.doi.org/10.3182/20120328-3-IT-3014.00064>. Acesso em: 23 jan. 2017.

ROJAS, J. D.; TADEO, F.; VILANOVA, R. Control of a pH neutralization plant using the VRFT framework. In: IEEE INTERNATIONAL CONFERENCE ON CONTROL APPLICATIONS (CCA), 2010, Yokohama. **Proceedings...** Piscataway: IEEE, 2010. p. 926–931. Disponível em: <https://doi.org/10.1109/CCA.2010.5611255>. Acesso em: 23 jan. 2017.

SÖDERSTRÖM, T. S.; STOICA, P. G. **System Identification**. Cambridge: Prentice Hall, 1989. (Prentice Hall International Series In Systems And Control Engineering).

TANENBAUM, A. S. **Computer Networks**. 4. ed. Upper Saddle River: Prentice Hall, 2003.

VERMA, M.; SINGH, S.; KAUR, B. An overview of bluetooth technology and its communication applications. **International Journal of Current Engineering and Technology**, Valley Village, v. 5, n. 3, p. 1588–1592, 2015. Disponível em: <http://inpressco.com/an-overview-of-bluetooth-technology-and-its-communication-applications>.

YENER, M.; DUNDAR, O. **Expert Android® Studio**. Indianapolis: Wrox, 2016.

ZECHNER, M.; GREEN, R. **Beginning Android 4 Games Development**. New York: Apress, 2011. Disponível em: <https://doi.org/10.1007/978-1-4302-3988-8>. Acesso em: 23 jan. 2017.