

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE GRADUAÇÃO E EDUCAÇÃO PROFISSIONAL
CENTRO DE COORDENAÇÃO
GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

MARCUS VINICIUS MEDEIROS

**UTILIZAÇÃO DE UM MICROCONTROLADOR DA FAMÍLIA KINETIS PARA
CONTROLE DE MOTORES CC POR MEIO DO MATLAB/SIMULINK**

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO

2017

MARCUS VINICIUS MEDEIROS

**UTILIZAÇÃO DE UM MICROCONTROLADOR DA FAMÍLIA KINETIS PARA
CONTROLE DE MOTORES CC POR MEIO DO MATLAB/SIMULINK**

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina TCC 2, do curso de Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof. Dr. Marcio Aurelio Furtado Montezuma
Coorientador: Prof. André Sanches Fonseca Sobrinho

CORNÉLIO PROCÓPIO
2017



Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia de Controle e Automação



FOLHA DE APROVAÇÃO

Marcus Vinícius Medeiros

Utilização de um microcontrolador da família kinetis para controle de motores CC por meio do Matlab/Simulink

Trabalho de conclusão de curso apresentado às 13:00hs do dia 13/11/2017 como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação no programa de Graduação em Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Avaliadora composta pelos professores abaixo assinados. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

Prof(a). Dr(a). Marcio Aurelio Furtado Montezuma - Presidente (Orientador)

Prof(a). Dr(a). André Sanches Fonseca Sobrinho - (Coorientador)

Prof(a). Dr(a). Cristiano Marcos Agulhari - (Membro)

Prof(a). Dr(a). Luis Fernando Caparroz Duarte - (Membro)

Dedico esse trabalho à Deus e à toda minha família, que estão sempre ao meu lado.

AGRADECIMENTOS

Gostaria de agradecer especialmente:

Ao Prof. Dr. Marcio Aurelio Furtado Montezuma, orientador deste trabalho, pela disponibilidade oferecida, paciência, incentivo, amizade, enfim por todo conhecimento que me transmitiu ao longo dos anos.

À minha família, em especial ao meu pai Sidney, minha mãe Maria Antonia, minha irmã Maria Clara, meu irmão Mateus Ulisses e minha avó Maria Huggler, grandes batalhadores, fontes da minha determinação que, mesmo estando longe, sempre me apoiaram e ensinaram o significado de amor e respeito.

Aos amigos de república, a Jokei's House, Vinícius Valério e Gustavo Floriano, por todos estes anos compartilhando momentos de descontração, companheirismo e lealdade.

Aos amigos do laboratório LaSisC, Lucas Niro, Shimada, Wagner, Takahashi, Mollon, Du, Gabriel, David e tanto outros que passaram pelo laboratório, pelas trocas de conhecimento que foram fundamentais para minha formação. Agradeço em especial ao Sequela, pela colaboração durante a realização de qualquer atividade prática.

Aos amigos Fellps e Caricas que são pessoas extraordinárias das quais aprendi muitos valores, lembrei outros esquecidos e tive o prazer de ensinar alguns.

Aos amigos da empresa Chiptronic, que não mediram esforços para transmitir conhecimento e apoio no período de estágio. Em especial ao Ricardo Kurumoto e ao Tiago Ribeiro que sempre me incluem em confraternizações do grupo.

A todos os professores da UTFPR, campus Cornélio Procópio, pelo ensino de qualidade e excelência.

E concluo agradecendo à todos que de uma maneira ou de outra contribuíram para que este trabalho pudesse ser concluído.

RESUMO

MEDEIROS, Marcus Vinicius. **Utilização de um microcontrolador da família kinetis para controle de motores cc por meio do matlab/simulink.** . 54 f. Trabalho de Conclusão de Curso – Graduação em Engenharia de Controle e Automação, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2017.

Atualmente plantas didáticas existem como ferramentas de auxílio ao ensino de sistemas de controle em razão de se mostrar um meio simples de conexão entre a prática e a teoria. Entretanto o acesso a estas plataformas de aprendizado tem restrição em relação ao tempo de uso e aos horários permitidos, uma vez que, na maioria das vezes os componentes das plantas estão contidos em laboratórios . Este trabalho de conclusão de curso tem como objetivo o desenvolvimento de um sistema de comunicação sem fio que tem por finalidade controlar a velocidade de um servomotor de forma remota. Para isso será utilizado um sistema de comunicação sem fio *bluetooth*, um microcontrolador NXP da família Kinetis e uma bancada de servomotor, de forma que a técnica de controle desejada possa ser implementada por meio do MATLAB/Simulink. A planta didática em questão mostrou funcionalidade e estabilidade nos testes realizados, satisfazendo os objetivos do trabalho. Ainda, toda documentação do protocolo utilizado foi detalhada para que qualquer pessoa possa replicar o conteúdo e resultados deste trabalho, auxiliando desta maneira em problemas de estudantes com restrições de acesso a laboratórios.

Palavras-chave: Controle de velocidade, Comunicação sem fio, Família Kinetis, Planta didática.

ABSTRACT

MEDEIROS, Marcus Vinicius. **Using Kineti's family microcontroller to control DC motors by means of Matlab / Simulink.** . 54 f. Course Completion Work - Control and Automation Engineering, Federal University of Technology - Paraná. Cornélio Procópio, .

Currently, didactic plants exist as tools to support the teaching of control because it is a simple way of connecting practice and theory. However the access to these learning platforms has restriction of time of use and the hours allowed to work with them. Therefore, in most cases, the components of the plants are contained in disputed laboratories. This work proposes the development of a wireless communication system designed to control the speed of a servomotor remotely. For that, it be used a bluetooth wireless communication, an NXP microcontroller from the Kinetis family and a servomotor, with control technique implemented through MATLAB / Simulink. Finally, one of the results was that the didactic plant in question showed functionality and stability in the tests performed. In addition, all documentation of the protocol used was detailed, so that anyone can replicate the content and results of this work, helping students solving their problems with laboratory restrictions.

Keywords: Speed control, Wireless communication, Kineti's family, Didactic plant

LISTA DE ILUSTRAÇÕES

FIGURA 1 – estrutura simplificada de um motor CC	12
FIGURA 2 – Exemplo dos discos de um encoder absoluto e incremental	13
FIGURA 3 – Exemplo de um microcontrolador de encapsulamento LQFP de 48 terminais	14
FIGURA 4 – Exemplo do sinal PWM com valores diferentes de ciclo ativo	15
FIGURA 5 – Disposição e descrição dos pinos da porta serial presente em computadores comerciais	16
FOTOGRAFIA 1 – Placa de desenvolvimento TWR - K60D100M	19
FOTOGRAFIA 2 – Bateria de <i>Lithium-polymer</i> utilizada no trabalho.	20
FOTOGRAFIA 3 – Módulo <i>bluetooth</i> HC-05 utilizado no trabalho.	21
FOTOGRAFIA 4 – Bancada do motor CC onde também se encontram o drive de potencia e o encoder.	22
FOTOGRAFIA 5 – Placa conversora de nível lógico.	23
FOTOGRAFIA 6 – Placa de Resistores <i>Pull-Up</i>	24
FIGURA 6 – conexões do sistema de transmissão e recepção de dados da interface RS232 do computador.	25
FIGURA 7 – Detalhes de uma visão geral do sistema.	26
FIGURA 8 – Configuração do pacote de entrada de dados no MATLAB/Simulink.	27
FIGURA 9 – Configuração do pacote de saída de dados no MATLAB/Simulink.	28
FIGURA 10 – Funcionamento do protocolo de comunicação do sistema.	28
FIGURA 11 – Implementação do controle PID no MATLAB/Simulink	30
FIGURA 12 – Janela aberta dentro do KDS onde encontram-se os componentes do PEx	31
FIGURA 13 – Configurações utilizadas no componente PWM1	32
FIGURA 14 – Configurações utilizadas no componente AS1	33
FIGURA 15 – Configurações utilizadas no componente EC161	33
FIGURA 16 – Configurações utilizadas no componente TI1	34
GRÁFICO 1 – Resposta do sistema à uma sinal de referência do tipo degrau com uma amplitude de 5000 RPM	38
GRÁFICO 2 – Resposta do sistema à uma sinal de referência do tipo seno com uma velocidade de pico de 6000 RPM e frequência de 0,2 Hz	39
GRÁFICO 3 – Resposta do sistema à uma sinal de referência do tipo seno com uma velocidade de pico de 6000 RPM e frequência de 0,5 Hz	40
GRÁFICO 4 – Resposta do sistema á um sinal de referência do tipo seno com uma velocidade de pico de 5000 RPM e frequência de aproximadamente 0,08 Hz obtida por Mollon (2016)	41

LISTA DE TABELAS

TABELA 1 – Especificações da resposta do sistema ao degrau de amplitude de 5000 RPM 39

SUMÁRIO

1	INTRODUÇÃO	9
1.1	OBJETIVO GERAL	9
1.2	OBJETIVOS ESPECÍFICOS	10
1.3	PROBLEMA	10
1.4	JUSTIFICATIVA	10
2	FUNDAMENTAÇÃO TEÓRICA	12
2.1	MOTOR DE CORRENTE CONTÍNUA	12
2.2	ENCODER	13
2.3	MICROCONTROLADOR	14
2.4	MODULAÇÃO POR LARGURA DE PULSO	14
2.5	COMUNICAÇÃO SERIAL	15
2.6	TECNOLOGIA <i>Bluetooth</i>	16
2.7	CONTROLE PROPORCIONAL, INTEGRAL E DERIVATIVO	17
3	MATERIAIS E MÉTODOS	18
3.1	PLACA DE DESENVOLVIMENTO TWR - K60D100M	18
3.2	TIPOS DE FONTES DE TENSÃO UTILIZADAS	20
3.3	MÓDULO <i>Bluetooth</i> HC-05	20
3.4	BANCADA COM MOTOR CC	21
3.5	CONVERSOR SN74LVC4245A	22
3.6	PLACA DE RESISTORES <i>Pull-Up</i> .	23
3.7	METODOLOGIA UTILIZADA PARA TRANSMISSÃO DE DADOS PARA O COMPUTADOR	24
3.8	VISÃO GERAL DO SISTEMA	25
3.9	IMPLEMENTAÇÃO DO CONTROLE PID	26
3.10	IMPLEMENTAÇÃO DO CÓDIGO NO MICROCONTROLADOR	31
3.10.1	Janela de componentes do KDS	31
3.10.2	Componente PWM1	32
3.10.3	Componente AS1	32
3.10.4	Componente EC161	33
3.10.5	Componente TI1	34
3.10.6	Código de aquisição e comunicação	34
4	RESULTADOS	38
4.1	RESPOSTA DO SISTEMA À FUNÇÃO DEGRAU	38
4.2	RESPOSTA DO SISTEMA À FUNÇÃO SENO	39
4.3	VALIDAÇÃO DE MEDIDAS DO SISTEMA	41
5	CONCLUSÃO	42
5.1	LIMITAÇÕES DO TRABALHO	43
5.2	TRABALHOS FUTUROS	43
	REFERÊNCIAS	45
	ANEXO A – CODIGO FONTE UTILIZADO NO MICROCONTROLADOR	
	PK60DN512VMD10	49

1 INTRODUÇÃO

O controle de velocidade em malha fechada de motores CC é de grande relevância prática tanto no meio acadêmico, auxiliando no processo de ensino da teoria de controle clássico e moderno quanto em nível industrial, sendo indicado em processos que necessitam de velocidades precisas ou com baixo grau de variação.

O controle automático desempenha um papel fundamental no avanço da engenharia e da ciência. Nos últimos anos disseminou-se a utilização de sistemas de controle automáticos (OGATA, 2007), que tem como intuito aperfeiçoar sistemas, ou seja, minimizar custos e tempo e maximizar a qualidade (BAYER; ARAÚJO, 2002). Muitos sistemas de automação só se tornaram possíveis devido aos recentes e grandes avanços na eletrônica. Os sensores que medem o valor ou estado de variáveis importantes em um sistema de controle são as entradas do sistema, mas o coração do sistema é o controlador eletrônico microprocessado (RIBEIRO, 2003).

A grande vantagem de se utilizar microcontroladores é que eles chegam a custar muitas vezes menos que um simples semicondutor, por se tratar de um circuito integrado (CI). Além disso, possui uma unidade central de processamento (CPU), capaz de processar várias informações, de uma memória ou de um periférico, ou mesmo diversos cálculos em curto espaço de tempo (CABRERA et al., 2010). A ARM traz a série *Advanced TIDC machine* (ARM) de microcontroladores de 32 bits de baixo custo e uso eficiente de energia. Mais de 10 bilhões de processadores contendo um núcleo ARM foram entregues, primeiramente para uso em sistemas embarcados. Atualmente 98 por cento dos telefones móveis contem pelo menos um processador (INSTRUMENTS, 2013), pela qual o microcontrolador escolhido tem seu núcleo ARM.

A placa de desenvolvimento utilizada no trabalho era é fabricada pela *Freescale Semiconductor*, que foi comprada em março de 2015 pela NXP, empresa líder atual no setor de semicondutores, por esta razão os símbolos e emblemas de ambas estarão presentes em algumas imagens deste trabalho.

Com bases nas informações supracitadas, o presente trabalho tem por finalidade utilizar um microcontrolador de 32 bits para aquisição de dados e controle por meio do MATLAB/Simulink.

1.1 OBJETIVO GERAL

Elaborar um código fonte para o microcontrolador PK60DN512VMD10 presente na plataforma de desenvolvimento TWR-K60D100M da NXP, para controle de pulsos de PWM que serão enviados ao servomotor, e ainda comunicar-se com o MATLAB/Simulink por meio da interface RS232 que por sua vez esta conectada a um módulo *bluetooth* que fará comunicação

ponto-a-ponto com outro módulo *bluetooth* ligado á placa de desenvolvimento. Para leitura de velocidade do servomotor foi utilizado um decodificador de quadratura, uma fonte de alimentação foi utilizada para energizar todo o sistema e fios flexíveis para as conexões elétricas necessárias.

1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um protocolo de comunicação de dados em linguagem C, para troca de dados entre MATLAB/Simulink e o microcontrolador;
- Montar um sistema de comunicação e controle com todos os periféricos;
- Aprimorar uma aplicação de controle no *simulink*;
- Ajustar os ganhos K_p , K_i e K_d do sistema em malha fechada;

1.3 PROBLEMA

A finalidade do trabalho consiste em criar um sistema de comunicação sem fio que controle a velocidade do servomotor de forma remota. Para isso, foi usado um módulo *bluetooth*, um microcontrolador NXP da família Kinetis e um kit de servomotor, com controle por MATLAB/Simulink.

A placa utilizada, TWR-K60D100M, juntamente com módulos *bluetooth* do modelo HC-05 fazem a comunicação do sistema de aquisição com o sistema de controle utilizando um protocolo desenvolvido. O microcontrolador capta os sinais do *encoder* em quadratura, decodifica-os e faz o controle da corrente do servomotor por meio do ciclo ativo de um sinal PWM proveniente de um cálculo de ação de controle realizado pelo software MATLAB/Simulink em tempo real, que torna acessível a troca do sistema de controle sem a necessidade de reprogramar o microcontrolador.

O resultado é um sistema de aquisição e controle completo utilizando o MATLAB/Simulink que controla a velocidade do servomotor sem fio.

O trabalho consiste em implementar um sistema de controle de velocidade em um servomotor utilizando em conjunto ao MATLAB/Simulink um microcontrolador de 32 bits com comunicação sem fio, com isso obter resultados analíticos e quantitativos, relacionados ao controle da planta.

1.4 JUSTIFICATIVA

Para elaboração deste trabalho aplicaram-se conhecimentos adquiridos ao longo do curso de engenharia de controle e automação, tais como sistemas microcontrolador e sistemas

de controle. Ambos mesclam-se na aplicação abordada uma vez que o controle acontece por intermédio de um microcontrolador.

A realização deste trabalho complementa o curso de sistemas microcontrolados, uma vez que foi utilizado um outro fabricante de microcontroladores, não estipulado na ementa, porém de grande utilização em diversos ramos da automação.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados conceitos pertinentes ao trabalho e estão divididos da seguinte maneira: Motor de corrente contínua; Encoder; Microcontrolador; Modulação por largura de pulso; Comunicação serial; Comunicação *bluetooth* e, por fim, Controle proporcional, integral e derivativo.

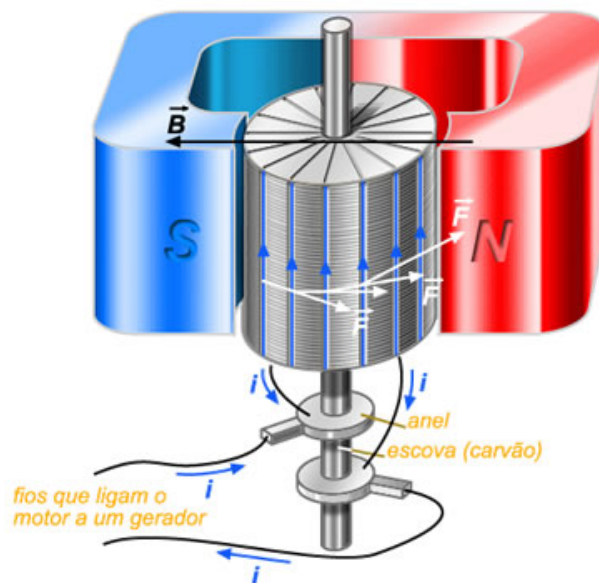
2.1 MOTOR DE CORRENTE CONTÍNUA

Nesta seção serão apresentados conceitos básicos relacionados ao motor de corrente contínua, seu princípio de funcionamento e a utilização do mesmo no trabalho.

O motor de corrente contínua é uma máquina capaz de converter a energia elétrica em mecânica por meio da interação de um campo magnético com condutores de corrente elétrica, e a consequência é o movimento do eixo (MOURA, 2014).

Sua construção é dada por elementos fixos e móveis, sendo que a parte fixa do motor é chamada de estator e a parte móvel é chamada de rotor (CHAPMAN, 2013). O motor CC apresenta um circuito de armadura encontrado no rotor e um circuito de campo encontrado no estator.

Figura 1 – estrutura simplificada de um motor CC



Fonte: (SANTOS, 2017)

Existem três métodos mais usuais de controle de velocidade de máquinas CC: ajuste do fluxo magnético; ajuste da resistência associada ao circuito de armadura; e ajuste da tensão

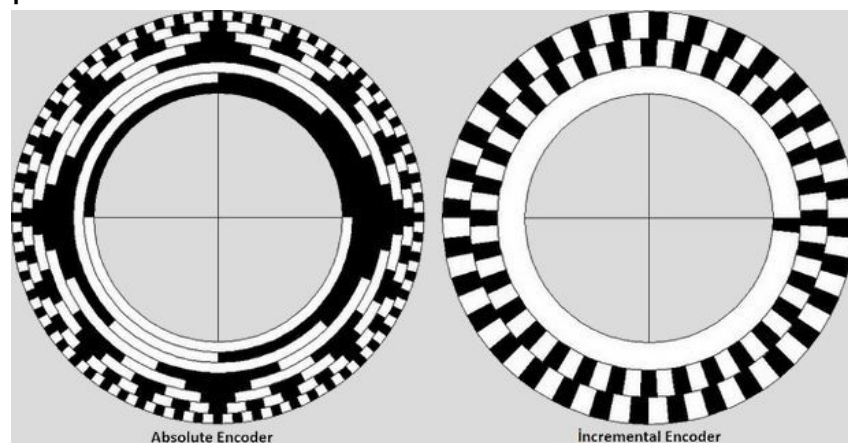
terminal de armadura (UMANS; JUNIOR; FITZGERALD, 2006). No trabalho que presente o último método citado foi usado.

O controle de velocidade por variação de tensão terminal de armadura é baseado no fenômeno em que uma mudança na tensão terminal de armadura de um motor é acompanhada, em regime permanente, por uma mudança substancialmente igual na força contra eletromotriz e, com fluxo constante, uma mudança conseqüentemente proporcional na velocidade do motor (UMANS; JUNIOR; FITZGERALD, 2006). Desta maneira é possível controlar grandezas como velocidade e corrente.

2.2 ENCODER

Caracterizado por ser um sensor que converte um movimento angular ou linear em uma série de pulsos digitais elétricos, o encoder fornece para o microcontrolador dados suficientes para transformá-los em informações, como posição ou velocidade angular. A conversão desses movimentos em pulsos elétricos é feita por meio da detecção fotoelétrica, na qual uma série de pulsos são gerados pela passagem da luz em um disco opaco, com várias aberturas transparentes. O receptor tanto detecta a luz enviada pelo emissor quanto a falta de luz, gerando assim os pulsos digitais com cada nível lógico diferenciando a existência ou não da abertura no disco (HEISS, 2012).

Figura 2 – Exemplo dos discos de um encoder absoluto e incremental



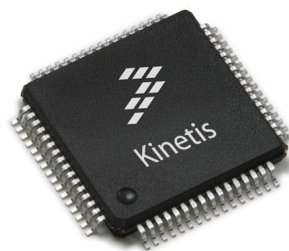
Fonte: (VOLT, 2017)

Existem diversos tipos de *encoder* que geralmente são caracterizados pelo seu disco, podem ser citados o do tipo absoluto, visto á esquerda da Figura 2, que se trata de um encoder que tem uma precisão ainda mais elevada do que o incremental. O encoder utilizado no trabalho é do tipo incremental. No entanto para fins didáticos, será utilizada apenas uma linha do mesmo, o que torna a implementação mais simples.

2.3 MICROCONTROLADOR

Um microcontrolador é um computador em um único chip. Ele possui um processador, memória e periféricos de entrada e saída. Sua principal diferença em relação a um microprocessador está na sua funcionalidade, uma vez que o último necessita de outros componentes como memória, chipsets, e periféricos para receber e enviar dados. Confrontando essa ideia, o microcontrolador foi projetado para ter todas estas funcionalidades dentro de uma única pastilha (SOUZA, 2016).

Figura 3 – Exemplo de um microcontrolador de encapsulamento LQFP de 48 terminais



Fonte: (ROSSI, 2017)

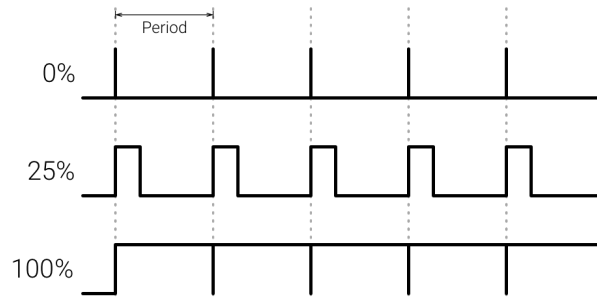
Com avanço da tecnologia tornou-se possível fabricar circuitos integrados cada vez mais compactos. Na figura 3 pode-se observar o encapsulamento LQFP (*Low Profile Quad Flat Pack*), que é um dos tipos de encapsulamento utilizados por microcontroladores da família *Kinetis*, e tem apenas 10 milímetros de lado e 16 terminais em cada lado (STATSCHIPPAC, 2017). Por esta razão torna-se mais versátil embarcar o sistema de aquisição em um microcontrolador e o fato deste ser de 32 bits está relacionado tanto ao poder de processamento quanto à possibilidade do uso de variáveis de ponto flutuante. Por fim, o microcontrolador da placa de desenvolvimento *Tower* conta com diversos recursos que foram necessários no trabalho, como por exemplo, decodificadores de quadratura.

2.4 MODULAÇÃO POR LARGURA DE PULSO

A modulação por largura de pulso (PWM) é usada para controlar circuitos analógicos a partir das saídas digitais de um microcontrolador, consistindo em representar um valor pelo ciclo ativo, que é o tempo em nível lógico alto, de um trem de pulsos de frequência fixa e determinada (SOBRINHO, 2012).

Em grande parte das aplicações de PWM para microcontroladores aproveita-se a energia de um sinal retangular proporcional ao seu ciclo ativo, uma vez que a energia de um sinal está relacionada com a área entre o sinal e o eixo do tempo (FABRETTI, 2016b).

Figura 4 – Exemplo do sinal PWM com valores diferentes de ciclo ativo



Fonte: (ANDROID, 2017)

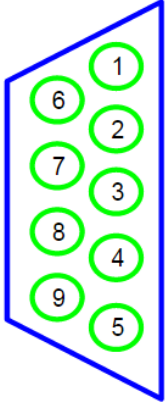
Para expor a situação de uma forma exemplificada, na Figura 4, quando o ciclo ativo é configurado com o valor total, o sinal resultante será uma onda que sempre estará em nível lógico alto. O ciclo ativo configurado com 1/4 de seu valor total na segunda onda de cima para baixo, a onda resultante é uma em que em 25% do tempo ela está em nível lógico alto, e 25% em nível lógico baixo (SOBRINHO, 2012).

A modulação por largura de pulso foi utilizada neste trabalho de maneira que a saída do sistema de malha fechada fosse um sinal PWM ajustável, de modo a ser aplicado aos terminais do motor CC e então chegar até as rotações desejadas.

2.5 COMUNICAÇÃO SERIAL

Comunicação em série consiste no processo de enviar dados de bit em bit, de maneira sequencial. É uma forma de comunicação muito comum utilizada por diversos dispositivos para instrumentação e também para aquisição de dados em conjunto com um dispositivo de amostragem. A comunicação é completada usando três linhas de transmissão : terra, transmissão e recepção, e pode ser síncrona ou assíncrona (NI, 2013).

Figura 5 – Disposição e descrição dos pinos da porta serial presente em computadores comerciais



Sigla	Descrição	Pino
DCD	Carrier Detect	1
RX	Receive Data	2
TX	Transmit Data	3
DTR	Data Terminal Ready	4
GND	Ground	5
DSR	Data Set Ready	6
RTS	Request to Send	7
CTS	Clear to Send	8
RI	Ring Indicator	9

Fonte: MecaWeb

A Figura 5 apresenta a disposição dos três pinos supracitados utilizados para comunicação serial entrar em funcionamento do modo conectado no trabalho. Entretanto, comunicar com dispositivos que não estão dentro do padrão TIA/EIA-232-F, é necessário a instalação de um circuito integrado do tipo *dual driver/receiver* em suas linhas de TX e RX. Por esta razão foi utilizado o circuito MAX232 da *Texas Instruments* com finalidade de ligar o módulo *bluetooth* HC-05 em uma linha de comunicação cujo padrão elétrico nativo não era o TTL (INSTRUMENTS, 2017).

2.6 TECNOLOGIA BLUETOOTH

Em meados de 1994, surgia o desenvolvimento de uma nova tecnologia de transmissão sem fio, iniciada pela Ericsson. A partir de 1998, fez-se um consórcio com finalidade de estudar e padronizar esse tipo de tecnologia que era representado pelas empresas: Sony, Ericsson, IBM, Intel, Toshiba e Nokia. O consórcio atualmente tem o nome de: *Bluetooth Special Interest Group* (SIG), e inclui mais de 2000 empresas (BONATTO; CANTO, S.A.).

Com o passar do tempo a tecnologia disseminou e hoje é conhecida por ser um dispositivo de curto alcance, com objetivo de eliminar os cabos nas conexões entre dispositivos eletrônicos. As principais características desta tecnologia são suas confiabilidades, baixo consumo e mínimo custo. Visando uma melhor praticidade em pesquisas, no desenvolvimento de protótipos e até mesmo produtos, módulos que embarcam o *Bluetooth* tem sido criados de maneira a deixar desenvolvedores e pesquisadores um passo a diante do projeto (BONATTO;

CANTO, S.A.).

Um módulo *Bluetooth* constrói um caminho físico sem fio para a informação serial trafegar. Os protocolos de comunicação da tecnologia em questão são encapsulados, ou seja, não há necessidade de se compreender detalhes do seu funcionamento. O usuário deve apenas considerar o módulo como se fosse um par de fios que possibilita a ligação dos terminais TX e RX do seu microcontrolador.

Para realização deste trabalho foram utilizados dois módulos do modelo HC-05, uma vez que o trabalho baseia-se na conexão sem fio de ponto-a-ponto entre o sistema supervisor desenvolvido no MATLAB/Simulink e o sistema composto pelo microcontrolador e a bancada do servomotor (INTELIGENTES, 2017).

2.7 CONTROLE PROPORCIONAL, INTEGRAL E DERIVATIVO

Genericamente, um controlador PID lê uma entrada, e calcula a saída por meio da soma do cálculo proporcional, do cálculo integral e do cálculo derivativo. É um dos tipos de controle clássico que consiste em ajustar a saída a partir de ganhos relacionados ao erro, a integral e derivada do erro em função do tempo. A integral do erro em função do tempo permite que a saída possa acompanhar a entrada com um erro muito menor, em alguns casos chegando próximo ao valor nulo. Entretanto, este pode também instabilizar o sistema se a ação integral for muito acentuada, sendo necessário uma sintonia com o sistema (OGATA, 2007).

A parcela proporcional do controlador é a que produz um sinal de saída proporcional à amplitude do erro, com constante de proporcionalidade K_p . Já a parcela de ação integral produz um sinal de saída que trabalha com o erro em função do tempo, ou seja, o erro acumulado, com constante de ganho da parcela integral K_i (NI, 2016). Os valores das constantes do controlador podem ser calculados utilizando vários métodos (OGATA, 2007), e podem ser escritos na forma da Equação (1):

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad (1)$$

3 MATERIAIS E MÉTODOS

Neste capítulo apresentam-se todos os materiais utilizados ao longo do trabalho, bem como as metodologias utilizadas.

3.1 PLACA DE DESENVOLVIMENTO TWR - K60D100M

Os experimentos foram realizados utilizando microcontrolador PK60DN512VMD10 de 32 bits da NXP. Esse encontra-se acoplado em uma placa de desenvolvimento TWR-K60D100M Fotografia 1. Algumas características do PK60DN512VMD10 são apresentadas abaixo (NXP SEMICONDUCTORS, 2016):

- Faixa de alimentação: 1,71V a 3,6V;
- Faixa de temperatura: -40°C , a 105°C ,.

Características de processamento e memória:

- Núcleo ARM Cortex-M4 com DSP;
- 512KB de memória flash programável;
- 128KB de memória RAM.

Periféricos essenciais para o projeto:

- Dois canais decodificadores de quadratura;
- Temporizadores de interrupção periódica.

Fotografia 1 – Placa de desenvolvimento TWR - K60D100M



Fonte: Autoria própria

A placa principal da plataforma de desenvolvimento TWR-K60D100M faz parte de uma estrutura que abriga vários outros tipos de plataformas do modelo TWR, que se caracterizam por ter uma aparência análoga a uma torre, em que os encaixes laterais fazem com que uma placa de desenvolvimento fique sobre a outra, já devidamente conectadas entre si, facilitando o desenvolvimento de conceitos e oferecendo acesso aos recursos das placas.

Dentre a diversidade modular de recursos, os essenciais que acompanham qualquer modelo TWR-K60D100M são:

- Circuito de *debug* JTAG, o que elimina o uso de um programador/debugador externo;
- *Slot* para SD card;
- Acelerômetro de três eixos (MMA78451Q);
- Quatro LEDs;
- Quatro *touch pads* capacitivos;
- Duas chaves do tipo *pushbutton*;
- Um potenciômetro.

Para o desenvolvimento do *firmware* programado no microcontrolador será utilizada a IDE Kinetis Design Studio na versão 3.0. Trata-se de uma interface de desenvolvimento aberta que permite a programação de toda família Kinetis e conta também com plugins que aceleram o desenvolvimento do código fonte criando componentes, subindo consideravelmente o nível da

linguagem de programação de forma a não necessitar de configuração manual dos registradores, diminuindo o período de adaptação na mudança de microncontroladores.

3.2 TIPOS DE FONTES DE TENSÃO UTILIZADAS

Esta seção relata os meios de alimentação da placa de desenvolvimento *Tower*, do motor CC e da interface RS-232, que junto com o módulo *bluetooth* HC-05 formam as conexões de transmissão e recepção de dados para o computador.

A alimentação da placa de desenvolvimento foi feita via USB (*Universal Serial Bus*). Isso foi possível pelo fato da mesma possuir uma entrada USB mini para alimentação, gravação e *debug*.

A Fotografia 2 mostra a bateria utilizada na alimentação do motor de corrente contínua. Feita de Lítio Polímero, a bateria possui capacidade de 5000 mAh e tensão de 18,9 V, quando carregada completamente, com 4 células.

Fotografia 2 – Bateria de *Lithium-polymer* utilizada no trabalho.



Fonte: Autoria própria.

A necessidade do uso da bateria veio em razão do motor CC ser uma carga indutiva. Ao desligá-lo, é gerada uma diferença de potencial reversa à fonte que o alimenta. Deste modo, se o motor fosse ligado a uma fonte de tensão provavelmente ocasionaria danos à mesma ou frearia o motor.

Já a alimentação do sistema de transmissão e recepção de dados da interface RS-232 do computador foi feita por uma fonte linear variável do modelo PS-6100 da marca Icel. Ajustada em 3,3 V, a mesma alimenta tanto um dos módulos HC-05 utilizados no trabalho, quanto o circuito conversor de sinal MAX232, como pode ser observado na figura 6.

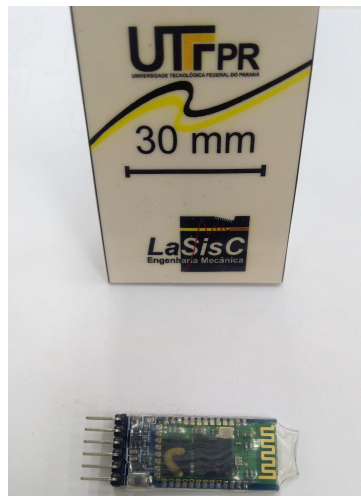
3.3 MÓDULO *BLUETOOTH* HC-05

O módulo utilizado na conversão de um par TX RX serial de nível lógico padrão TTL para *Bluetooth* [®] mostrado na Fotografia 3. Este tem dois modos de funcionamento: ordem-resposta

e de conexão automática, sendo esta última a utilizada neste trabalho.

No primeiro modo, o usuário envia comandos "AT" para o módulo para configurar os parâmetros de controle e enviar a ordem de controle. No segundo, o módulo permanece com a última configuração efetuada para transmissão de informações. Ao ligar o módulo, este automaticamente estará em modo de funcionamento de conexão automática, ou seja, em busca de outro módulo para pareamento.

Fotografia 3 – Módulo *bluetooth* HC-05 utilizado no trabalho.



Fonte: Autoria própria.

O módulo HC-05 tem a flexibilidade de ser configurado como um dispositivo mestre ou como um dispositivo escravo. Neste trabalho, um dispositivo HC-05 foi adotado como um dispositivo mestre, podendo ser pareado com qualquer dispositivo que queira se comunicar com ele e um dispositivo HC-05 escravo pareado com o anterior, estabelecendo dessa maneira a conexão de ponto-a-ponto.

3.4 BANCADA COM MOTOR CC

Para realização dos experimentos do trabalho foi utilizada uma bancada de motor de corrente contínua que se encontra no Laboratório de Sistemas Automatizados e Controle (LaSisC) da Universidade Tecnológica Federal do Paraná, Campus Cornélio Procópio. A bancada pode ser vista na Fotografia 4.

Fotografia 4 – Bancada do motor CC onde também se encontram o drive de potencia e o encoder.



Fonte: Autoria própria.

A bancada contém um motor CC de ímã permanente da marca Pittman, com tensão de trabalho máxima de 19,1 V e rotação de 10000 RPM. Acoplado ao mesmo eixo do motor CC, existe um encoder incremental de quadratura, com 3 canais, modelo HDS-5310 da marca Avago Technologies, que possui 512 PPR.

A título de conhecimento, um outro motor CC genérico é acoplado por meio de uma corrente ao motor CC estudado na bancada, possibilitando testes com carga. Entretanto os ensaios deste trabalho foram realizados com o motor rodando livremente. Outro componente presente na bancada é o *drive* de potência que é responsável pelo acionamento do motor CC a partir do sinal PWM, sua tensão de operação de 5 V a 28 V, corrente de até 5 A e frequência máxima de PWM de até 10 kHz.

3.5 CONVERSOR SN74LVC4245A

Fabricado pela empresa Texas Instruments, o conversor SN74LVC4245A possui dois portais, com 8 pinos cada, que podem ser usados para conversão bidirecional de níveis de tensão. O portal A é dedicado para dispositivos que operam com faixa de tensão de 4,5 V até 5,5 V. Por outro lado o portal B é dedicado para ligação de dispositivos que operam entre 2,7V e 3,6V. Os demais componentes desta placa são do os circuito regulador também disponível na mesma, para que quando a placa for alimentada com 5V, a mesma possa oferecer 3,3V ao circuito integrado.

Fotografia 5 – Placa conversora de nível lógico.



Fonte: Autoria própria.

O uso desta placa no trabalho foi necessário pois o microcontrolador utilizado opera em 3,3V e o *drive* de potência da bancada de controle do motor CC opera com entradas de 5V.

É importante destacar que existem na placa pinos de configuração que direcionam o fluxo dos sinais. Para a aplicação no trabalho os pinos OE e DIR presentes na placa foram ligados ao nível lógico baixo (GND no caso) para que o fluxo de sinal fosse do portal B para o portal A fazendo com que o sinal de PWM trabalhasse para uma faixa operacional entre 0V e 5V ao invés de 0 e 3.3V.

3.6 PLACA DE RESISTORES *PULL-UP*.

Projetada e confeccionada por estudantes do Laboratório de Sistemas de Controle (LaSisC) a placa de resistores *pull-up* (em detalhe na Fotografia 6) é utilizada para melhorar a forma de onda fornecida pelo encoder, evitando problemas com a leitura da posição angular com o aumento da velocidade.

Fotografia 6 – Placa de Resistores *Pull-Up*.



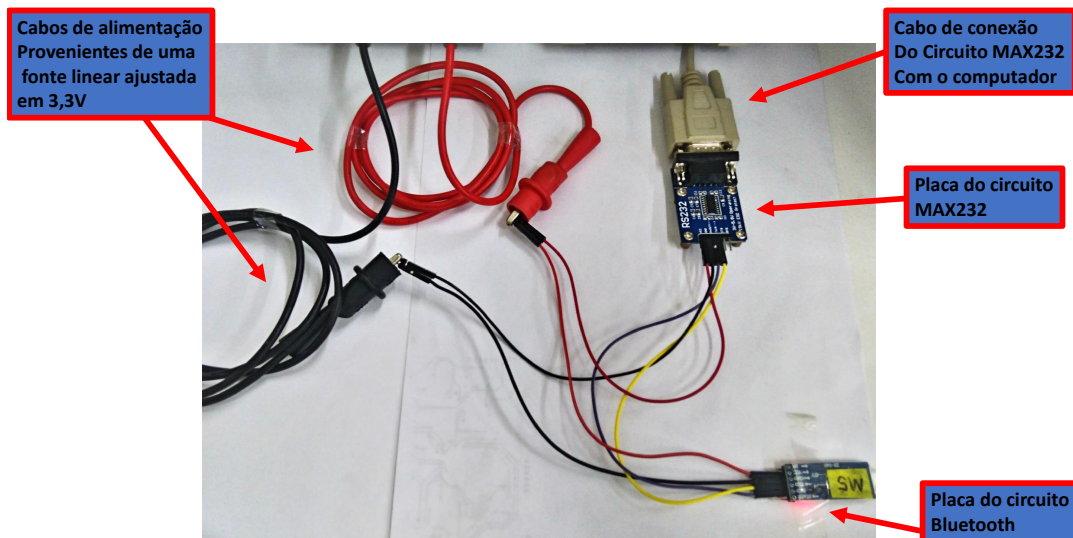
Fonte: Autoria própria.

É possível visualizar na Fotografia 6 que a placa possui quatro terminais de cada lado. Ambos os lados são espelhados, ou seja, tanto na direita quanto na esquerda, estão as ligações para os pinos de: alimentação, canal A, canal B, e GND respectivamente de cima para baixo. Um lado da placa é conectado nas saídas do encoder, enquanto o outro é conectado na placa de desenvolvimento *Tower*.

3.7 METODOLOGIA UTILIZADA PARA TRANSMISSÃO DE DADOS PARA O COMPUTADOR

Como dito anteriormente, para estabelecer uma conexão de ponto-a-ponto sem fio o ensaio contou com dois módulos *Bluetooth* do modelo HC-05. Nesta subseção será apresentado como foram feitas as conexões entre a porta serial do computador, a placa de desenvolvimento tem um circuito conversor de RS-232 e um dos módulos de comunicação sem fio de curta distância. A Figura 6 mostra as ligações feitas entre os quatro componentes: O computador, a fonte de alimentação PS-6100 ajustada em 3,3 V, a placa conversora de sinal do tipo RS-232 para o padrão TTL e o módulo HC-05 configurado como *slave* e com uma taxa de transmissão de 115200 bits por segundo.

Figura 6 – conexões do sistema de transmissão e recepção de dados da interface RS232 do computador.



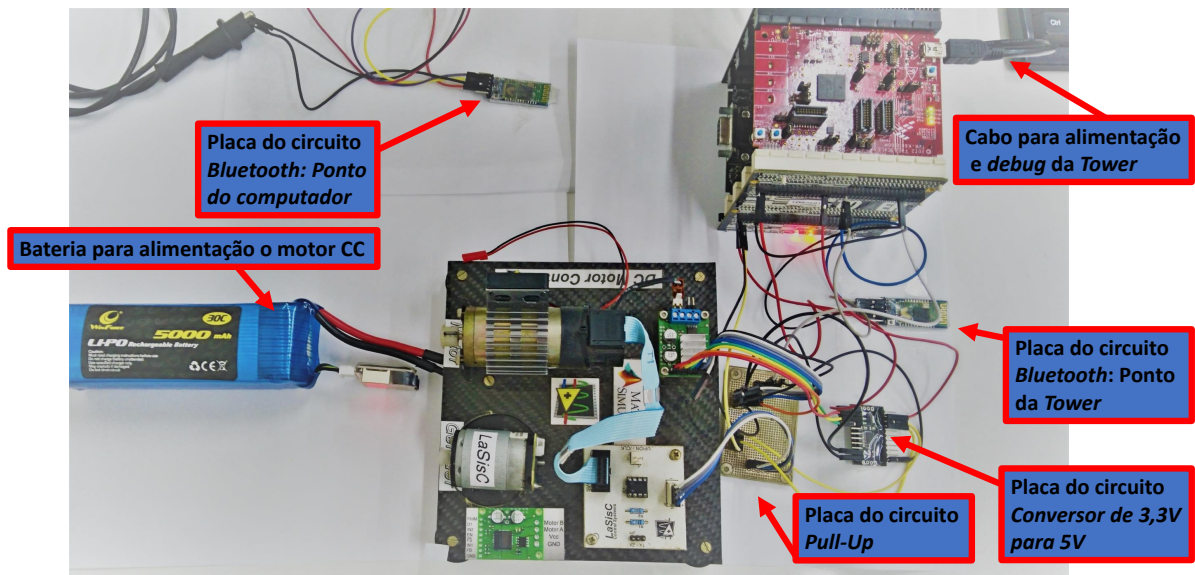
Fonte: Autoria própria.

Os cabos e fios vermelhos e pretos vistos na Figura 6 são respectivamente 3,3 V e GND, já os fios amarelos e roxo são o par TX RX, também respectivamente identificados. Por fim um cabo que tem um terminal do tipo DB9 conecta a placa do circuito integrado MAX232 ao computador. A fonte alimenta todos os componentes e logo após estes estarem energizados é possível estabelecer uma conexão com outro módulo *bluetooth*.

3.8 VISÃO GERAL DO SISTEMA

Tendo todas as conexões da parte do computador realizadas, os demais componentes do experimento foram conectados na *Tower* e suas disposições podem ser vistas na Figura 7.

Figura 7 – Detalhes de uma visão geral do sistema.



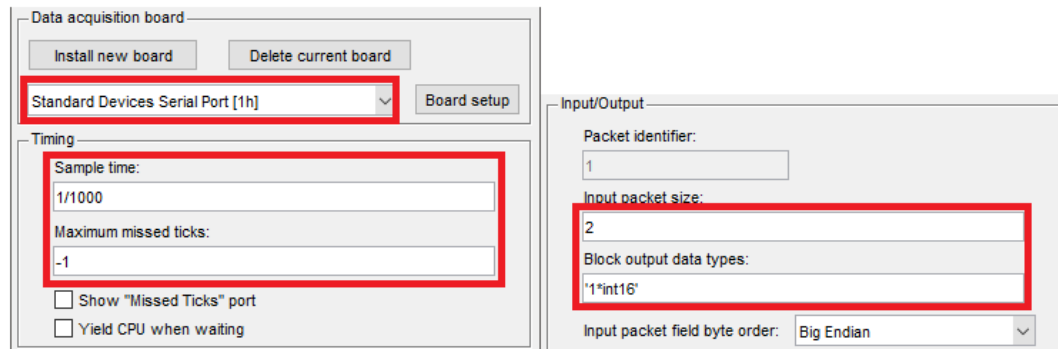
Fonte: Autoria própria.

Devidamente conectados e alimentados os componentes do sistema, foi possível a implementação do algoritmo de controle e a programação do microcontrolador.

3.9 IMPLEMENTAÇÃO DO CONTROLE PID

O ambiente de desenvolvimento utilizado para implementação do algoritmo de controle foi o MATLAB/Simulink, que trabalha com diagrama de blocos e por meio de pacotes específicos, e permite que o usuário tenha acesso a algumas portas de entrada e saída de dados do computador, como a porta serial que foi utilizada nesse trabalho. Para ter acesso á porta serial, é necessário que a biblioteca *Simulink Desktop Real-Time* esteja previamente instalada. Após esse processo, o próximo passo é o de configuração dos blocos *Packet Input* e *Packet Output* que estão presentes nesta biblioteca.

Figura 8 – Configuração do pacote de entrada de dados no MATLAB/Simulink.

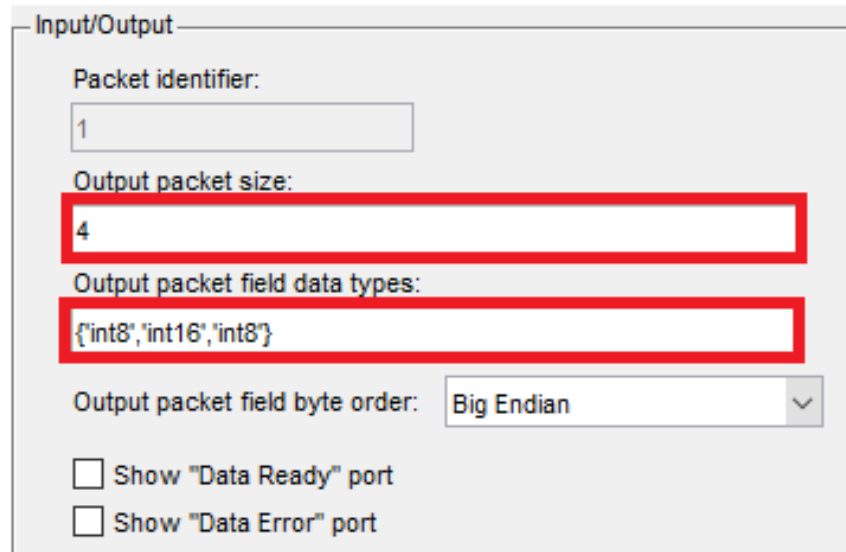


Fonte: Autoria própria.

Como visto na Figura 8, a primeira configuração é a de seleção da placa que irá ser utilizada. Para este trabalho foi utilizada a porta serial do computador, que geralmente é intitulada no MATLAB/Simulink como *Standard Devices Serial Port*. A seguir encontra-se uma caixa de texto para determinar o *Sample Time*, que vem a ser o tempo de amostragem, ou seja, o período em que serão capturados os dados do sistema. Em seguida existe a opção para determinar o número máximo de pacotes perdidos, neste campo o valor de -1 indica que está sendo carregada uma configuração geral contida em *Model Configuration Parameters*. Nesta aba encontra-se o mesmo campo de pacotes perdidos da porta serial, entretanto quando determinado um número máximo de pacotes perdidos nos parâmetros de configuração do modelo, este dado é enviado a todos os campos correspondentes que estão com a identificação -1.

Ainda na Figura 8 é configurado o tamanho do pacote de entrada, onde existe um campo que deve ser preenchido com o número de bytes que chegará em cada pacote. Há também um campo onde se definem quantos bytes compõem as estruturas de dados a serem lidas. Em especial o pacote de entrada existe apenas uma estrutura que é formada por dois bytes, diferentemente do pacote de saída (Figura 9) que tem seu tamanho diferente e sua estrutura de dados é dividida em três partes: um byte para sinalização de início de transmissão, dois bytes de dados e um byte de sinalização de fim de transmissão.

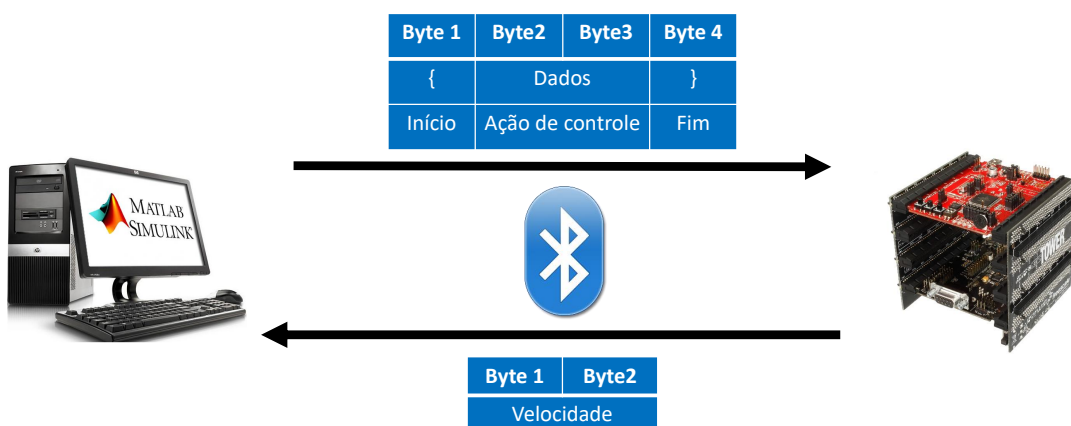
Figura 9 – Configuração do pacote de saída de dados no MATLAB/Simulink.



Fonte: Autoria própria.

Juntando as informações contidas nas Figuras 8 e 9 nota-se que o sistema estabeleceu, por meio das configurações dos pacotes de entradas e saída da porta serial, um protocolo de comunicação próprio entre a *Tower*, que terá a descrição da sua parte do protocolo na subseção a seguir, e o MATLAB/Simulink. A Figura 10 abaixo exemplifica o protocolo em detalhes.

Figura 10 – Funcionamento do protocolo de comunicação do sistema.



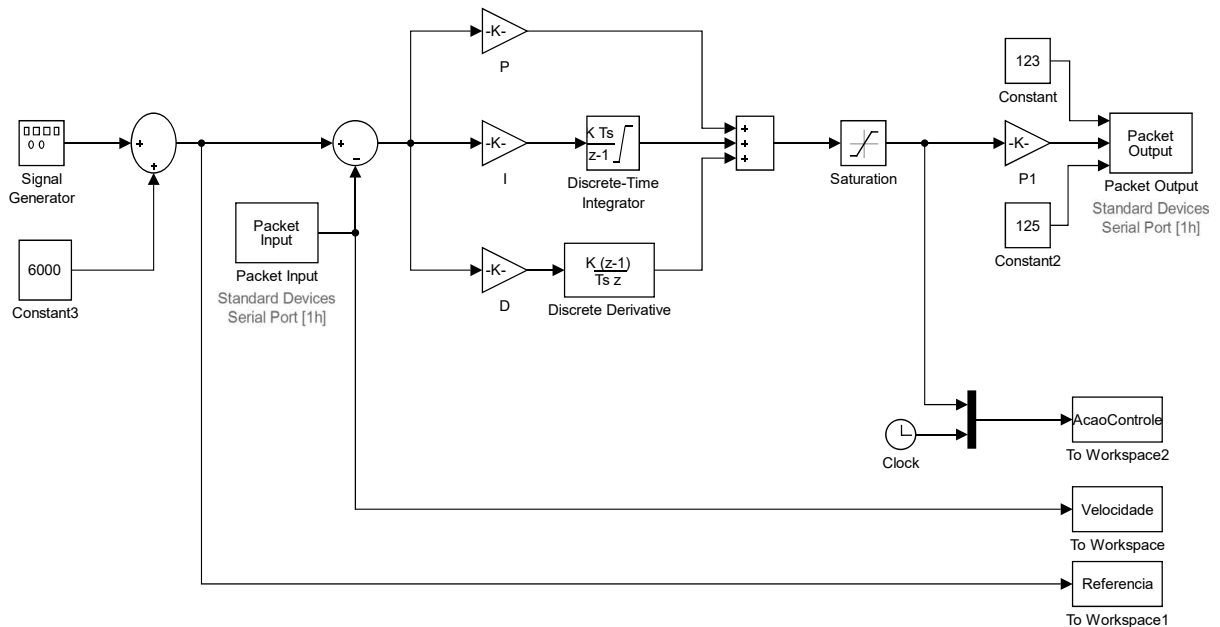
Fonte: Autoria própria.

O pacote de saída do MATLAB/Simulink inicia o protocolo e envia o valor decimal de

123 que equivale ao caracter "{" na tabela ASCII, indicando no protocolo o início da comunicação. Em seguida são enviados dois bytes de dados e, por fim o último byte que representa o caracter "}" na tabela ASCII, que significa o fim dessa transmissão. Da outra ponta do sistema a *Tower* verifica se o primeiro e o último byte tem os valores esperados pelo protocolo. Em caso afirmativo o *firmware* está programado para guardar os dois bytes de dados e usar como ciclo ativo do PWM, pois o dado que vem do computador é justamente a ação de controle, que nesta planta vem a ser o ciclo ativo. E ainda segue coletando as informações necessárias para adquirir o valor de velocidade e enviá-lo ao computador por meio dos dois bytes que outrora foram configurados para entrada de dados no computador. Dessa maneira fecha-se o *loop* de comunicação entre uma ponta e outra do sistema.

Os demais blocos são mais usuais para modelos comuns, que não utilizam periféricos do computador. Esses podem ser vistos na Figura 11, como os blocos de ganho que servem tanto para implementar o controlador, quanto para ajustar na faixa de operação do ciclo ativo do PWM. Ainda pode-se notar o bloco integrador e o derivativo, que também fazem parte da implementação do controlador.

Figura 11 – Implementação do controle PID no MATLAB/Simulink



Fonte: Autoria própria.

Os blocos não relatados nesse trabalho que encontram-se na Figura 11 não fazem parte do sistema, estão acoplados com finalidade de coleta de dados e visualização das curvas formadas. Sendo esses de fácil configuração e manuseio.

O ganho K_p teve valor de $20 * 10^{-6}$ enquanto o ganho K_i foi de $2 * 10^{-6}$ tendo uma limitação de 17% do valor final do integrador, uma vez que esse bloco pode levar a saturação do sistema em momentos desnecessários e indesejáveis. O valor de K_d de $3,5 * 10^{-9}$, assumindo uma parcela mínima no controle, uma vez que com o controlador PI o sistema já convergia para referência com o valor da somatória, com o valor da somatória dos três blocos, seu resultado era ajustado e limitado em uma faixa de 0 a 1, e logo em seguida, reajustado para outra faixa de 0 a 65535 que são os possíveis valores que o ciclo ativo pode assumir.

A sintonia do controlador foi feita por meio de um método empírico já utilizado tanto por Repinaldo (2015) quanto por Mollon (2016). em uma comparação com método tradicional de Ziegler Nichols, o qual geralmente tem-se um sobressinal de no máximo 25%, os resultado obtidos foram de sobressinais menores do que essa porcentagem.

Uma das vantagens de implementar o controle fora do microcontrolador é que as alterações de ganho taxa de amostragem do sistema não implicam em ter que reprogramar o microcontrolador. E ainda, é possível alterar a técnica de controle utilizada sem que o *firmware* do microcontrolador seja alterado.

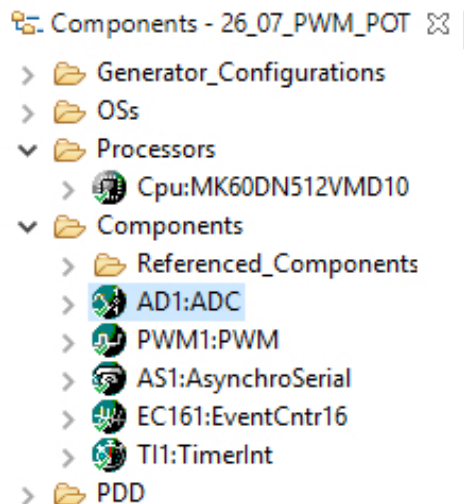
3.10 IMPLEMENTAÇÃO DO CÓDIGO NO MICROCONTROLADOR

Como descrito na Seção 3.1 o ambiente de desenvolvimento dos códigos deste trabalho foi o KDS *Kinetis Design Studio*. Neste ambiente existe o PEx (*Processor Expert*) que permite a criação de componentes que auxiliam, por exemplo, a configuração de periféricos. Esta seção apresenta os métodos cujos componentes utilizados no trabalho foram configurados, bem como o código de aquisição e comunicação.

3.10.1 Janela de componentes do KDS

Ao criar um programa nesta IDE algumas opções devem ser selecionadas para que o *plugin* de geração de componentes seja ativado. A principal delas é a confirmação da utilização do pacote na janela de configuração de um programa novo. Logo após, é perceptível uma nova janela acoplada ao ambiente de desenvolvimento, como visto na Figura 12.

Figura 12 – Janela aberta dentro do KDS onde encontram-se os componentes do PEx



Fonte: Autoria própria.

O PEx gera automaticamente as configurações iniciais que inicializam o microcontrolador e o deixam em estado de funcionamento, tais como *clock* e modo de operação. A Figura 12 trás como exemplo cinco componentes. Entretanto os utilizados neste trabalho foram os quatro últimos: PWM1; AS1; EC161 e o TI1.

O componente AD1 é responsável por configurar um canal do conversor analógico-digital e foi utilizado com finalidade de testar o periférico PWM, da seguinte maneira: no canal configurado do ADC foi conectado um potenciômetro, assim o programa obtinha o valor do ADC que era ajustado de acordo com o potenciômetro e atribuía esse valor ao ciclo ativo do PWM. Foi possível então notar a variação do ciclo ativo do PWM por meio de um osciloscópio conectado ao pino do periférico, conforme a resistência do potenciômetro era modificada. Como ambas as variáveis dos componentes eram de 16 bits, se o potenciômetro tinha valor máximo de resistência, o ciclo ativo era nulo e vice-versa, sem que fosse necessário nenhum tipo de transformação linear.

3.10.2 Componente PWM1

A configuração deste componente resume-se em 3 fatores: canal; pino utilizado; e período do sinal. A Figura 13 mostra vários outros parâmetros opcionais que podem ou não ser configurados.

Figura 13 – Configurações utilizadas no componente PWM1

Component name	PWM1
PWM or PPG device	FTM1_C1V
Duty compare	
Output pin	PTA9/FTM1_CH1/MII0_RXD3/FB_AD16/FTM1_C
Output pin signal	
Counter	FTM1_CNT
Period	5 ms
Starting pulse width	0 ms
Initial polarity	high
Same period in modes	<input type="checkbox"/>
Component uses entire timer	<input type="checkbox"/>

Fonte: Autoria própria.

Ainda na Figura 13 vê-se que o canal utilizado tem o nome de FTM1_C1V, e o pino utilizado foi o mesmo da porta 9 do portal A. Como existem vários tipos de encapsulamento do microcontrolador, o número do terminal correspondente do encapsulamento utilizado pode ser visto ao passar o *mouse* sobre este campo. Por fim, foi definido um período de 5 milissegundos, o que corresponde a uma frequência de 200 Hz no sinal.

3.10.3 Componente AS1

O componente AS1 é um dos responsáveis pela comunicação entre sistema de aquisição e o de controle, uma vez que o mesmo configura um módulo de comunicação serial assíncrona. São configurados seis campos, sendo cinco presentes na Figura 14.

Figura 14 – Configurações utilizadas no componente AS1

Channel: UART3

Interrupt service/event: Settings Initialization CPU clock/speed selecti Referenced components

Parity: none none

Width: 8 bits 8 bits

Stop bit: 1 1

Receiver

RxD: PTC16/CAN1_RX/UART3_RX/ENET0_1588_TMRI

RxD pin signal: [Empty]

Transmitter

TxD: PTC17/CAN1_TX/UART3_TX/ENET0_1588_TMR

Fonte: Autoria própria.

Os primeiros três campos da figura Figura 14 dizem respeito à paridade, ao tamanho do pacote e à quantidade de bits de parada, respectivamente. Os dois últimos correspondem à seleção dos pinos de transmissão e recepção e foram alocados nas portas 17 e 18 do portal C, respectivamente. O último campo configurável do componente vem a ser o que define o *baud rate* da comunicação, sendo no trabalho utilizado o valor de 115200 bits por segundo.

3.10.4 Componente EC161

Com finalidade de contar os pulsos do encoder, o componente EC161 foi configurado de maneira que o mesmo conta as bordas de subida de uma das quadraturas do disco, os campos de configuração podem ser vistos na Figura 15.

Figura 15 – Configurações utilizadas no componente EC161

Component name: EC161

Counter: LPTMR0_CNR

Interrupt service/event: Event Mode Initialization Referenced components

Counter input pin: PTC5/LLWU_P9/SPI0_SCK/LPTMR0_ALT2/I2S0_F

Input pin signal: [Empty]

Edge: rising or falling edge rising edge

Fonte: Autoria própria.

O primeiro campo da Figura 15 seleciona qual o contador utilizado no componente, neste caso o LPTMR0_CNR de 16 bits, o segundo campo informa qual porta vai ser utilizada para captura do evento, sendo utilizada no trabalho a porta 5 do portal C. Ainda existe o campo de seleção de borda, no entanto o microcontrolador utilizado no trabalho apenas faz contagens de borda de subida com esse tipo de componente.

3.10.5 Componente TI1

Para coletar amostras do sinal mesmo que não haja uma requisição foi adotada uma interrupção periódica, e para configurar a mesma, como mostrado na Figura 16, foi criado o componente TI1.

Figura 16 – Configurações utilizadas no componente TI1

The image shows a configuration window for a component named 'TI1'. The settings are as follows:

- Component name: TI1
- Periodic interrupt source: FTM0_MOD
- Counter: FTM0_CNT
- Interrupt period: 1 ms (Clock cfg. 0: 1.000 ms)
- Same period in modes:
- Component uses entire timer:
- Interrupt service/event:
- Interrupt: INT_FTM0
- Interrupt priority: medium priority (8)

Fonte: Autoria própria.

Um dos campos configurados foi a fonte de interrupção periódica, sendo selecionado a fonte FTM0_MOD, que disponibilizou o contador FTM0_CNT para este fim. Todas as interrupções do trabalho foram habilitadas com o mesmo nível de prioridade e isso não afetou o funcionamento do sistema.

3.10.6 Código de aquisição e comunicação

Esta subseção detalha o desenvolvimento do código utilizado no microcontrolador em estudo. Para tanto, é importante saber que foram utilizadas funções criadas pelo PEx por meio dos componentes outrora discriminados. O código encontra-se em sua íntegra no anexo 1 para que toda comunidade acadêmica possa ter um proveito melhor deste.

O Algoritmo 1 mostra a função EC161_GetNumEvents que retorna o valor da contagem de eventos, a mesma encontra-se dentro de uma interrupção periódica que acontece num período de um milissegundo para logo em seguida, ser calculada a velocidade. Antes de ser feito o cálculo é feita uma verificação. Em caso da variável atual de contagem de eventos, intitulada *pos*, ser menor que a variável que guarda o valor da iteração anterior, o cálculo da velocidade muda, uma vez que é necessário fazer um ajuste no cálculo de deslocamento. A multiplicação por 1000 corresponde á transformação da unidade de tempo de milissegundos para segundos e a multiplicação de 60 transforma a mesma unidade em minutos. Por fim a divisão por 512 tem o proposito de converter a unidade de deslocamento de pulsos para rotação. Assim a unidade de velocidade tratada no algoritmo é RPM (Rotações Por Minuto). Depois de calculada a velocidade

é calculada a média entre a velocidade atual e a velocidade anterior com finalidade de diminuir o erro que acontece na coleta. Essa média é guardada na variável global `vel_med` e transmitida em outra parte do código.

Algoritmo 1 – Código para coleta de pulsos do *encoder* e cálculo da velocidade

```

void T11_OnInterrupt(void) {
    EC161_GetNumEvents(&pos);

    if (pos >= pos_ant) {
        vel = ((pos - pos_ant)*1000*60)/512;
        vel_med = (vel + vel_ant)/2;
        pos_ant = pos;
        vel_ant = vel;
    }
    if (pos < pos_ant) {
        vel = (((65535 - pos_ant)+pos)*1000*60)/512;
        vel_med = (vel + vel_ant)/2;
        pos_ant = pos;
        vel_ant = vel;
    }
}

```

A interrupção de recepção do componente AS1 acontece quando o MATLAB/Simulink envia a informação para a placa de desenvolvimento, neste instante é verificado se o primeiro byte da informação tem o valor decimal de 123. Em caso afirmativo uma *flag* é ativada para habilitar a memorização dos dois bytes da ação de controle em um vetor de duas posições. O valor desse vetor será o valor do ciclo ativo do sinal de PWM para que a velocidade atual da planta atinja a velocidade de referência do sistema, essa variável tem um tamanho de 16 bits exatamente o tamanho da variável que a função do componente PWM1 recebe como parâmetro para ajuste do ciclo ativo.

Depois de guardar os dados de ação de controle, o último byte da recepção é checado, e se o mesmo tiver o valor decimal de 125 que corresponde ao símbolo "]" na tabela ASCII, o protocolo que recebe os dados é terminado, a flag de recepção é desativada e a de transmissão, ativada.

Com a *flag* de transmissão ativada, a velocidade outrora calculada está habilitada para ser transmitida. As linhas de código que seguem fazem a divisão da variável "vel_med" em dois bytes para que a mesma possa ser transmitida pela função de AS1, que envia um byte por vez. Existe um atraso feito com o laço *for* entre a transmissão de um byte e outra, para que o módulo utilizado pelo componente possa estar completamente vazio.

Algoritmo 2 – Código para recepção da ação de controle advinda do módulo *bluetooth*

```

void AS1_OnRxChar(void) {
    AS1_RecvChar(&recebido);
    if (recebido == 123){flag = 1;}
    if ( flag ==1 && x < 2 && recebido != 123 ) {
        data[x] = recebido;
        x++;
    }
    if ( x == 2 && recebido == 125) {
        flag = 0;
        flag_main = 1;
    }
}

```

Após o envio dos bytes de velocidade, o vetor *data* tem o valor de suas duas posições somado e alocado na variável *acao_controle* que vem a ser um parâmetro da função que ajusta o ciclo ativo do componente PWM1. Antes deste ajuste é verificado se existe um estouro no valor da variável, em caso afirmativo o valor é convertido em um dos seus limites, inferior para um estouro negativo e superior para estouros positivos.

Algoritmo 3 – Código para transmissão da velocidade

```

if (flag_main == 1)
{
    AS1_SendChar((0x0000FF00 & vel_med)>>8);
    for(int j = 0; j<10000;j++);
    AS1_SendChar(0x000000FF & vel_med);
    acao_controle = ((data[1]<<8) & 0b1111111100000000)
        ↔ +((data[0]) & 0b0000000011111111);
    if(acao_controle > 65535) {acao_controle = 65535;};
    if(acao_controle < 0) {acao_controle = 0;};
    PWM1_SetRatio16(acao_controle);
    x=0;
    flag_main = 0;
}

```

O passo seguinte ao ajuste do valor da ação de controle é o de limpar a variável que posiciona o vetor que aloca o valor de velocidade medido e enfim desabilitar a flag de transmissão.

Desta maneira encerra-se um ciclo de controle da planta, o que significa dizer que, aplicada uma referência, os dados de saída foram lidos por meio de sensores e transmitidos ao sistema de controle que por sua vez fez cálculos para que o erro do sistema fosse nulo. A ação de controle gerada foi então transmitida e recebida pelo sistema de aquisição que ajustou o ciclo

ativo. Esse ciclo ocorre inúmeras vezes, conforme estimado o tempo do experimento no modelo do MATLAB/Simulink. Depois de sintonizado o controlador empiricamente, a repetição de vários ciclos de controle mostrou um sistema estável, com resultados satisfatórios que serão descritos e analisados no capítulo a seguir.

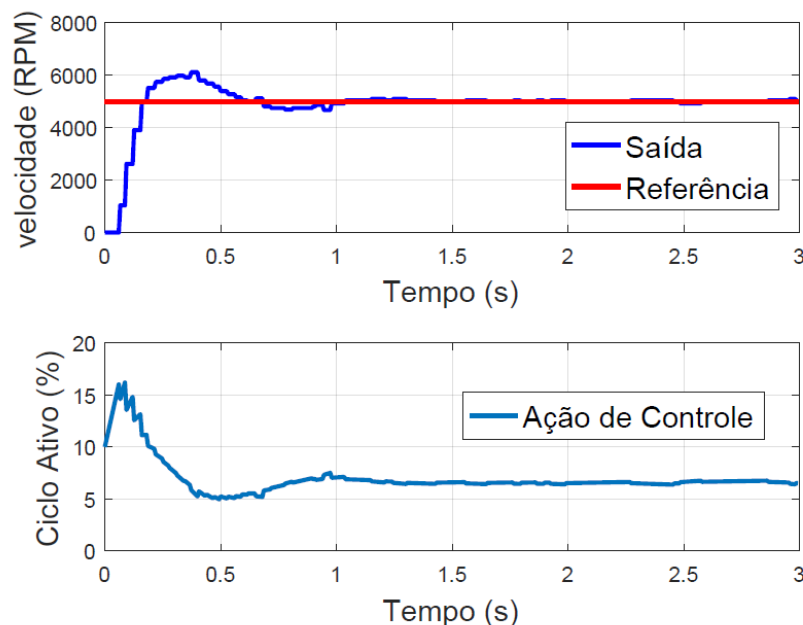
4 RESULTADOS

Neste capítulo estão relatados os resultados obtidos no decorrer do trabalho, bem como comparações pertinentes a trabalhos realizados no mesmo laboratório.

4.1 RESPOSTA DO SISTEMA À FUNÇÃO DEGRAU

Geralmente as características de desempenho de um sistema de controle são especificadas em termos da resposta transitória para uma entrada em degrau unitário, pois esta entrada é de fácil geração e suficientemente severa (CAMPOS, 2010). Por isso, em uma primeira abordagem, com finalidade de analisar os parâmetros de regime transitório do motor, foi aplicado como referência um sinal do tipo degrau como visto no Gráfico 1 com amplitude de 5000 rotações por minuto. A linha vermelha do gráfico superior exibe os dados da referência aplicada ao sistema, enquanto a linha azul, ainda do gráfico superior, mostra os dados coletados na saída no mesmo. O Gráfico 1 também ilustra, em porcentagem, a ação de controle do controlador que vem a ser o ciclo ativo do sinal PWM enviado ao motor.

Gráfico 1 – Resposta do sistema à uma sinal de referência do tipo degrau com uma amplitude de 5000 RPM



Fonte: Autoria própria

Por meio de indicadores visuais do software MATLAB observa-se um sobressinal de 19,52% e um tempo de acomodação de aproximadamente 0,98 segundos, fazendo desta, uma resposta satisfatória do sistema em relação á referência aplicada. Outros dados podem ser visualizados na Tabela 1.

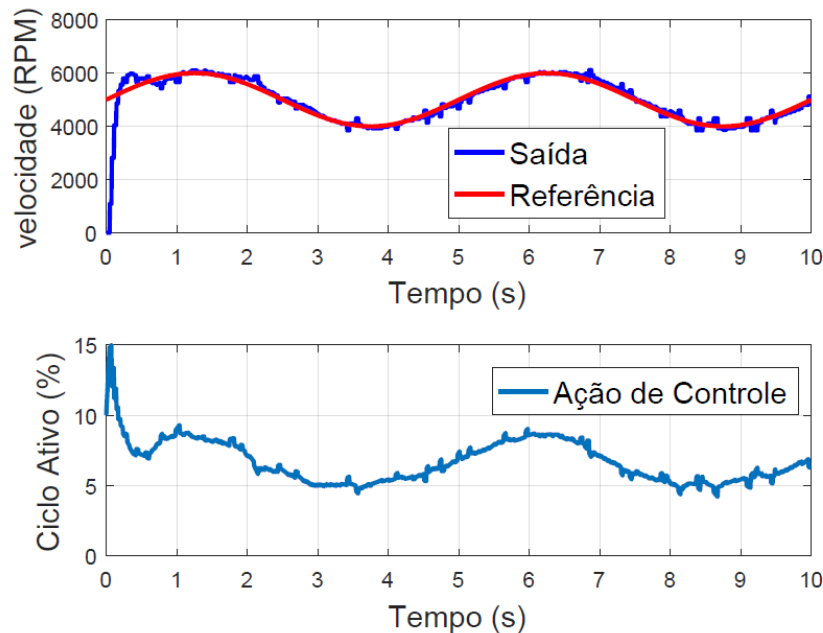
Tabela 1 – Especificações da resposta do sistema ao degrau de amplitude de 5000 RPM

	Velocidade (RPM)	Tempo (s)
Instante de pico	5976	0,3133
Tempo de atraso	de 500 até 4500	0,77
Tempo de subida	de 0 até 2500	0,093
Tempo de assentamento	4921	0,98

O instante de pico do sistema, que é o momento em que se atinge a velocidade máxima ocorreu em 0,3133 segundo, cujo valor de velocidade era 5976 RPM. O valor de 6015 RPM pode ser considerado um ruído externo do sensor, uma vez que a velocidade começa cair depois do que foi considerado instante de pico, e ainda que o número de ocorrências desse valor (uma ocorrência) não é significativo.

4.2 RESPOSTA DO SISTEMA À FUNÇÃO SENO

A planta também mostrou características de estabilidade com outras funções, como por exemplo a função do tipo seno, o primeiro experimento atual consiste da aplicação de uma forma de onda de referência com período de 5 segundos, velocidade de pico de 6000 RPM e amplitude de 1000 RPM.

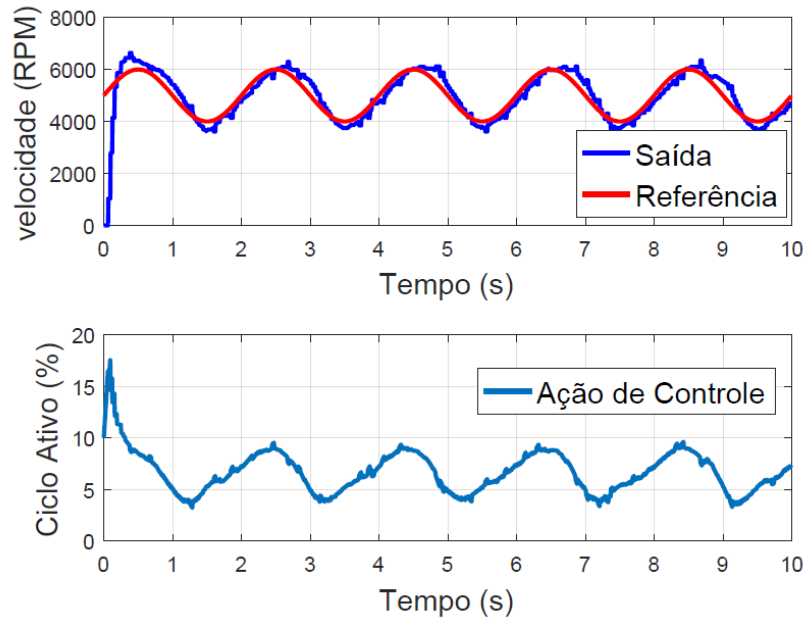
Gráfico 2 – Resposta do sistema à uma sinal de referência do tipo seno com uma velocidade de pico de 6000 RPM e frequência de 0,2 Hz

Fonte: Autoria própria

Afim de confirmar a estabilidade neste tipo de forma de onda, o período da mesma foi alterado para 0,5 Hz como visto no Gráfico 3 e ainda pode ser comparado com o resultado obtido

no trabalho de [Mollon \(2016\)](#) no Gráfico 4.

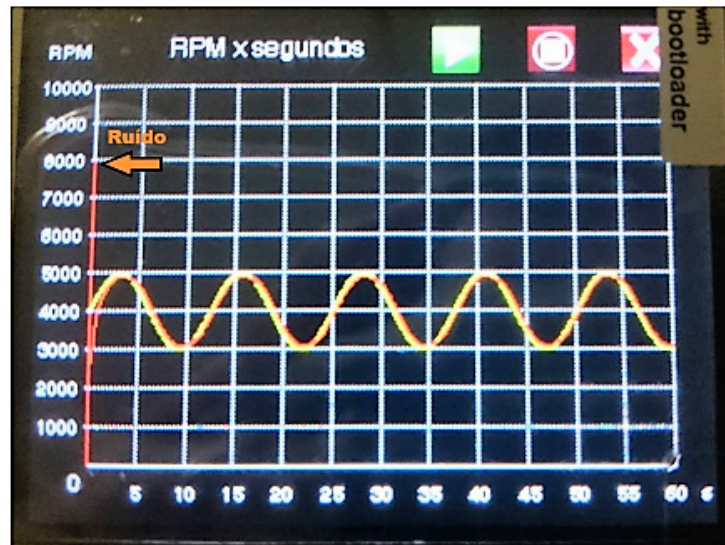
Gráfico 3 – Resposta do sistema à uma sinal de referência do tipo seno com uma velocidade de pico de 6000 RPM e frequência de 0,5 Hz



Fonte: Autoria própria

É importante ressaltar que o trabalho de [Mollon \(2016\)](#) constituiu de uma planta didática de motor CC para controle PID de um servomotor com *display* TFT e suporte á conexão remota, por esse motivo o Gráfico 4 é mostrado em um *display* TFT e o tempo de amostras do experimento é fixado em 60 segundos. Entretanto nota-se que a estabilidade é um fator em comum nos dois sistemas que utilizaram o mesmo método de sintonia.

Gráfico 4 – Resposta do sistema á um sinal de referência do tipo seno com uma velocidade de pico de 5000 RPM e frequência de aproximadamente 0,08 Hz obtida por Mollon (2016)



Fonte: Adaptado de Mollon (2016)

Os dois sistemas, embora desenvolvidos com tecnologias diferentes, apresentam resultados semelhantes e satisfatórios para o controle da planta. Ambos os trabalhos seguem a mesma linha de pesquisa cujas tecnologias estudadas fazem parte do aprendizado de estudantes de disciplinas de sistemas microcontrolados e sistemas de controle.

4.3 VALIDAÇÃO DE MEDIDAS DO SISTEMA

Para validação do *baud rate* da transmissão foi utilizado o analisador lógico da marca *Logic* modelo de 16 entradas, onde 4 foram utilizadas, sendo elas o par TX/RX da ponta do computador e o par TX/RX da ponta do microcontrolador. No analisador foi identificada toda informação transmitida e recebida.

Para verificar o cálculo da velocidade foi utilizado o tacômetro digital MDT-2238A da Minipa que tem a função "*photo RPM*", que dispara um *laser* refletor em uma fita refletora acoplada ao eixo do motor. Foram feitas medidas em várias faixas de operação, sendo o erro mais alto de 2,4% do valor real, o que tornou-se um resultado satisfatório.

Por fim é válido lembrar que foram feitos testes com o osciloscópio modelo MO-2061 da Minipa para validação da frequência do sinal PWM, e todos os valores ajustados no *firmware* foram confirmados no equipamento de medição.

5 CONCLUSÃO

Em busca de um melhor aperfeiçoamento de controle de processos industriais faz-se necessário o conhecimento de sistemas de controle tanto na teoria quanto na prática. Entretanto, na maioria dos casos, as plantas didáticas são comercializadas por um elevado valor aquisitivo, e ainda os estudantes têm acesso restrito as mesmas em laboratórios de ensino uma vez que seu uso é disputado.

Por essas razões este trabalho teve como objetivo o controle de um servomotor, em que o sistema de aquisição foi inserido em uma placa de desenvolvimento que possui um microcontrolador de 32 bits e controle implementado por meio do *software MATLAB/Simulink*. Fora desenvolvida uma implementação de um controle clássico, o PID, justamente para mostrar-se de maneira didática um controle de uso comum ensinado em cursos de engenharia elétrica e afins.

A interface de controle foi separada da interface de aquisição de maneira que o usuário do sistema consiga ajustar os ganhos da planta, ou ainda mudar a técnica de controle sem preocupação com o código fonte do microcontrolador. Isso possibilita um rápido ajuste de uma planta para outra, e também uma rápida mudança de técnica de controle, podendo ser aplicadas inclusive técnicas de controle moderno de realimentação de estados ou de lógica nebulosa por exemplo. O único requisito de projeto, seguir o protocolo desenvolvido para comunicação entre o sistema de controle e o sistema de aquisição.

Outra parte tão interessante quanto a separar o controle da aquisição foi a utilização de um microcontrolador que nem sempre é encontrado como foco de estudo em cursos de sistemas microcontrolados, sendo esta uma alternativa de tecnologia para estudantes que desejam aprofundar-se no assunto. Além disso o PK60DN512VMD10 tem processamento de 32 bits, o que possibilita desenvolver *firmwares* que trabalhem com ponto flutuante (variáveis tipo *float*). E uma particularidade útil neste trabalho, a presença de decodificador de quadratura, o que permitiu o uso do componente de controle de eventos. Sem este, seria necessário o uso de um circuito de decodificação externo, o que fragmentaria e deixaria mais complexo o *hardware* do sistema.

O uso da comunicação sem fio de ponta-a-ponta implementada neste trabalho por meio de módulos providos da tecnologia *Bluetooth* é um dos meios mais simples e menos custosos de estabelecer uma conexão *wireless*. Desta maneira torna-se acessível tanto ao orçamento quanto á capacidade didática dos alunos.

Em termos de validação, foram considerados aspectos básicos e fundamentais para o funcionamento de uma planta de controle. Logo foram avaliados o comportamento do controle e a velocidade de referência formada por uma curva, bem como tempo de amostragem, dos

protocolos de comunicação, do cálculo da velocidade e da frequência do PWM.

Os critérios de avaliação apresentaram resultados satisfatórios, uma vez que o controlador PID sintonizado realizou o controle da maneira esperada e se manteve próximo à velocidade de referência, a qual foi gerada exatamente de acordo com os parâmetros desejados. Uma forma pertinente de avaliar medidas de velocidade foi adotada no trabalho, trata-se de comparar a velocidade obtida pelo algoritmo do sistema de controle com a de um tacômetro digital, esta comparação apresentou um erro considerado aceitável com valor médio de 2,4% .

5.1 LIMITAÇÕES DO TRABALHO

Devido ao objetivo de desenvolver um trabalho que possa servir de referência para estudantes iniciantes tanto no estudo de sistemas microcontrolados quanto no estudo de sistemas de controle, foi utilizado um controlador clássico e um método simples de aquisição.

Alguns terminais do *drive* de potência foram fixados com 5V ou com GND com finalidade de forçar o motor a girar em apenas um sentido, uma vez que essa realidade já traria funcionalidade ao experimento.

A precisão *encoder* incremental de 3 faixas não foi usufruída em sua plenitude pois apenas um componente de controle de eventos foi configurado. Isso aconteceu pela razão de que, nesta placa de desenvolvimento, outros terminais deste periférico não encontram-se disponíveis.

5.2 TRABALHOS FUTUROS

Assim como o trabalho de [Fabretti \(2016a\)](#), [Mollon \(2016\)](#), e [Shimada \(2011\)](#) esse trabalho segue uma linha de desenvolvimento estipulada no LaSisC para que o leque de tecnologias utilizadas em controle de motores CC se abra e então a escolha desta seja mais apurada conforme a necessidades e requisições de projeto.

Os métodos usuais de sintonia de controladores PID podem ser utilizados no sistema desenvolvido, com poucas modificações no modelo MATLAB/Simulink, já que o modelo utilizado facilita a sintonia de maneira experimental.

Com uma outra placa de desenvolvimento da mesma família ou ainda com o mesmo microcontrolador em outra placa, é possível configurar mais componentes do tipo EC16 ou ainda do tipo EC32 que fariam com que o *encoder* desse uuma maior precisão. Embora que para fins didáticos o presente desenvolvimento se mostrou satisfatório.

Existem várias outras funcionalidades e recursos presentes na TWR-K60D100M que podem ser utilizados em outros trabalhos de conclusão bem como: comunicações utilizadas no

segmento automotivo, sensores capacitivos do tipo *touch*, 256 níveis de prioridade de interrupção, entre outros.

Os módulos *bluetooth* HC-05 são amplamente utilizados em uma infinidade de aplicações e podem ser usados em sistemas de controle não apenas em uma comunicação do tipo ponta-a-ponta mas também em casos com diversas interfaces.

REFERÊNCIAS

- ANDROID, Developer. **PWM**. 2017. Disponível em: <<https://developer.android.com/things/sdk/pio/pwm.html>>. Citado na página 15.
- BAYER, F. M.; ARAÚJO, O. C. B. DE. **Curso técnico em automação industrial: Controle automático de processos**. 3. ed. [s.n.], 2002. Disponível em: <http://estudio01.proj.ufsm.br/cadernos_automacao/quinta_etapa/controle_automatgico_processos_2012.pdf>. Citado na página 9.
- BONATTO, Aurélio; CANTO, Diego Oliveira do. **BLUETOOTH TECHNOLOGY (IEEE 802.15)**. S.A. Artigo Científico, S.A. Disponível em: <<http://www.inf.pucrs.br/~cnunes/redes/Trabalho%20Bluetooth.pdf>>. Citado 2 vezes nas páginas 16 e 17.
- CABRERA, A.A. Alvarez et al. Towards automation of control software: A review of challenges in mechatronic design. 2010. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0957415810000838?via%3Dihub>>. Citado na página 9.
- CAMPOS, Paulo Roberto Brero de. **Análise de resposta transiória para sistemas de primeira e segunda ordem**. 2010. Notas de aula, 2010. Disponível em: <<http://pessoal.utfpr.edu.br/brero/primeirasegundaordem.pdf>>. Citado na página 38.
- CHAPMAN, Stephen J. **Fundamentos de máquinas elétricas**. [S.l.]: AMGH Editora, 2013. Citado na página 12.
- FABRETTI, Henrique Dalavale. **DESENVOLVIMENTO DE UM SISTEMA DE COMUNICAÇÃO WIRELESS COM MATLAB/SIMULINK USANDO MICROCONTROLADOR DA FAMÍLIA HCS08**. 2016. Trabalho de Conclusão de Curso. (Graduação em Engenharia da Computação), 2016. Citado na página 43.
- FABRETTI, Henrique Dalavale. **Desenvolvimento de um sistema de comunicação wireless com MATLAB/Simulink usando microcontrolador de 8 bits da família HCS08**. 2016. 58 f. Trabalho de Conclusão de Curso (Graduação) - Engenharia de Computação. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016. Citado na página 14.
- HEISS, Augusto. Encoders saiba como funcionam os sensores mais usados na automação industrial. **Mecatrônica Atual**, Editora Saber, n. 56, p. 14–17, 2012. Citado na página 13.
- INSTRUMENTS, National. **Introdução à tecnologia de microcontroladores ARM para sistemas embarcados**. 2013. Disponível em: <<http://www.ni.com/newsletter/50786/pt/>>. Citado na página 9.
- INSTRUMENTS, Texas. **MAX232x Dual EIA-232 Drivers/Receivers**. [S.l.], 2017. Disponível em: <<http://www.ti.com/lit/ds/symlink/max232.pdf>>. Citado na página 16.
- INTELIGENTES, Pisciotta Soluções. **Módulo Bluetooth com Arduino**. 2017. Disponível em: <<http://pisciotta.com.br/professor.asp?prof=5>>. Citado na página 17.
- MOLLON, Matheus Fernando. **PLANTA DIDÁTICA PARA CONTROLE PID DE UM SERVOMOTOR COM DISPLAY TFT E SUPORTE A CONEXÃO REMOTA**. 2016. TRABALHO DE CONCLUSÃO DE CURSO, 2016. Citado 5 vezes nas páginas 6, 30, 40, 41 e 43.

MOURA, Marcos André da Silva. **Modelação e identificação de motor DC**. 2014. Tese (Doutorado), 2014. Citado na página 12.

NI. **Introdução à tecnologia de microcontroladores ARM para sistemas embarcados**. 2013. Disponível em: <<http://www.electrical4u.com/working-or-operating-principle-of-dc-motor/>>. Acesso em: 2 out. 2016. Citado na página 15.

_____. **What is a Pulse Width Modulation (PWM) Signal and What is it Used For?** 2016. Disponível em: <<http://digital.ni.com/public.nsf/allkb/2AC662081E01E1AE86257F5D00011B6D/>>. Acesso em: 2 out. 2016. Citado na página 17.

NXP SEMICONDUCTORS. **TWR-K60D100M: Kinetis K60 100 MHz MCU Tower System Module**. 2016. Disponível em: <<http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/kinetis-cortex-m-mcus/k-series-performance-m4/k6x-ethernet/kinetis-k60-100-mhz-mcu-tower-system-module:TWR-K60D100M>>. Acesso em: 14 out. 2016. Citado na página 18.

OGATA, KAFSUTUKO. **Engenharia de Controle Moderno**. 4. ed. São Paulo: [s.n.], 2007. Citado 2 vezes nas páginas 9 e 17.

REPINALDO, Joana Pereira. **Controle de Velocidade de um Servomotor Utilizando Software Labview Real-Time**. 2015. Trabalho de Conclusão de Curso, 2015. Citado na página 30.

RIBEIRO, MARCO A. **Fundamentos da Automação**. 1. ed. Salvador: [s.n.], 2003. Citado na página 9.

ROSSI, Henrique. **Kinetis K8x - Microcontroladores ARM Cortex-M4 seguros da Freescale**. 2017. Disponível em: <<https://www.embarcados.com.br/k8x-kinetis-freescale/>>. Citado na página 14.

SANTOS, Marco Aurélio da Silva. **Eletricidade: Acionamento de Motores Elétricos**. 2017. Disponível em: <<http://brasilescola.uol.com.br/fisica/eletricidade-acionamento-motores-eletricos.htm>>. Citado na página 12.

SHIMADA, Bruno Masaharu. **Desenvolvimento de uma Bancada Didática para Controle de Motores CC Utilizando Matlab/Simulink em Tempo Real**. 2011. Trabalho de Conclusão de Curso. (Graduação em Engenharia Mecânica), 2011. Citado na página 43.

SOBRINHO, André Sanches Fonseca. **Sistemas Microcontrolados**. 2012. 58 f. Apostila da disciplina de sistemas microcontrolados do curso de graduação - Engenharia de Computação. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2012. Citado 2 vezes nas páginas 14 e 15.

SOUZA, Vitor Amadeu. **A história e a diferença entre um microcontrolador e um microprocessador**. 2016. Disponível em: <www.cerne.com.br/Artigo3_Historia.pdf>. Acesso em: 14 out. 2016. Citado na página 14.

STATSCHIPPAC. **Low Profile Quad Flat Pack: LQFP, LQFP-ep**. [S.l.], 2017. Disponível em: <<http://www.statschippac.com/~media/Files/Package%20Datasheets/LQFP.ashx>>. Citado na página 14.

UMANS, Stephen D.; JUNIOR, Charles Kingsley; FITZGERALD, A. E. **Máquinas Elétricas com Introdução à Eletrônica de Potência**. 6. ed. [S.l.: s.n.], 2006. Citado na página 13.

VOLT, 320. **ENCODER, ANGLE MEASUREMENT CCS C PIC16F628
SAMPLE APPLICATION.** 2017. Disponível em: [http://320volt.com/en/
encoder-kullanimi-aci-olcumu-ve-ccs-c-pic16f628-ornek-uygulama/](http://320volt.com/en/encoder-kullanimi-aci-olcumu-ve-ccs-c-pic16f628-ornek-uygulama/)>. Citado na página
13.

Anexos

ANEXO A – CODIGO FONTE UTILIZADO NO MICROCONTROLADOR PK60DN512VMD10

Algoritmo 1 – Arquivo main.c - parte 1 de 3.

```

/* #####
**      Filename      : main.c
**      Project       : 26_07_PWM_POT
**      Processor     : MK60DN512VMD10
**      Version       : Driver 01.01
**      Compiler      : GNU C Compiler

/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "AD1.h"
#include "AdcLdd1.h"
#include "PWM1.h"
#include "PwmLdd1.h"
#include "TU1.h"
#include "AS1.h"
#include "ASerialLdd1.h"
#include "EC161.h"
#include "EventCntrLdd1.h"
#include "TU2.h"
#include "TI1.h"
#include "TimerIntLdd1.h"
#include "TU3.h"
/* Including shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

```

Algoritmo 2 – Arquivo main.c - parte 2 de 3.

```

/* User includes (#include below this line is not maintained by
↳ Processor Expert) */
long int acao_controle =0;
unsigned int pos = 0x0, pos_ant=0x0;
unsigned int vel = 0, vel_med = 0, vel_ant = 0 ;
extern char data[2];
extern char x;
extern char flag_main;

/* lint -save -e970 Disable MISRA rule (6.3) checking. */
int main(void)
/* lint -restore Enable MISRA rule (6.3) checking. */
{
    /* Write your local variable definition here */
    unsigned int potenciometro = 0;
    /** Processor Expert internal initialization. DON'T REMOVE THIS
↳ CODE!!! ***/
    PE_low_level_init();
    /** End of Processor Expert internal initialization.
↳ ***/

    /* Write your code here */
    /* For example: for(;;) { } */
    for(;;) {
        //AD1_Measure(TRUE);
        //AD1_GetValue16(&potenciometro);
        //PWM1_SetRatio16(potenciometro);
        if(flag_main == 1)
        {
            for(int j = 0; j<10000;j++);
            AS1_SendChar((0x0000FF00 & vel_med)>>8);
            for(int j = 0; j<10000;j++);
            AS1_SendChar(0x000000FF & vel_med);
            for(int j = 0; j<10000;j++);
            acao_controle = ((data[1]<<8) & 0b1111111100000000)
↳ +((data[0]) & 0b0000000011111111);
            if(acao_controle > 65535) {acao_controle = 65535;};
            if(acao_controle < 0) {acao_controle = 0;};
            PWM1_SetRatio16(acao_controle);
            x=0;
            flag_main = 0;
        }
    }
}

```

Algoritmo 3 – Arquivo main.c - parte 3 de 3.

```

/** Don't write any code pass this line , or it will be deleted
    ↪ during code generation. ***/
/** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS
    ↪ component. DON'T MODIFY THIS CODE!!! ***/
#ifdef PEX_RTOS_START
    PEX_RTOS_START(); /* Startup of the selected
    ↪ RTOS. Macro is defined by the RTOS component. */
#endif
/** End of RTOS startup code. ***/
/** Processor Expert end of main routine. DON'T MODIFY THIS
    ↪ CODE!!! ***/
for (;;) {}
/** Processor Expert end of main routine. DON'T WRITE CODE
    ↪ BELOW!!! ***/
} /** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/

/* END main */

```

Algoritmo 4 – Arquivo events.c - parte 1 de 3.

```

/*
** Filename      : Events.c
** Project      : 26_07_PWM_POT
** Processor    : MK60DN512VMD10
** Component    : Events
** Version      : Driver 01.00
** Compiler     : GNU C Compiler
** Date/Time    : 2017-07-26, 10:39, # CodeGen: 0
** Abstract     :
** This is user's event module.
** Put your event handler code here.
** Contents     :
** Cpu_OnNMIINT – void Cpu_OnNMIINT(void);
*/
/* !
** @file Events.c
** @version 01.00
** @brief
** This is user's event module.
** Put your event handler code here.
*/
/* !
** @addtogroup Events_module Events module documentation
** @{
*/
/* MODULE Events */

#include "Cpu.h"
#include "Events.h"

#ifdef __cplusplus
extern "C" {
#endif

```

Algoritmo 5 – Arquivo events.c - parte 2 de 3.

```

/*
** Event      : AS1_OnRxChar (module Events)
**
** Component  : AS1 [AsynchroSerial]
** Description :
** This event is called after a correct character is
  ↪ received.
** The event is available only when the <Interrupt
** service/event> property is enabled and either the
  ↪ <Receiver>
** property is enabled or the <SCI output mode> property (if
** supported) is set to Single-wire mode.
** Parameters : None
** Returns    : Nothing
*/

char flag = 0;
char flag_main = 0;
char data[2] = {0,0};
char recebido =0;
char x = 0;
void AS1_OnRxChar(void) {
    AS1_RecvChar(&recebido);
    if(recebido == 123){flag =1;}
    if( flag ==1 && x < 2 && recebido != 123 ) {
        data[x] = recebido;
        x++;
    }
    if( x == 2 && recebido == 125) {
        flag = 0;
        flag_main = 1;
    }
}

```

Algoritmo 6 – Arquivo events.c - parte 3 de 3.

```

/*
** Event      :  TI1_OnInterrupt (module Events)
**
** Component  :  TI1 [TimerInt]
** Description :
** When a timer interrupt occurs this event is called (only
** when the component is enabled – <Enable> and the events
  ↪ are
** enabled – <EnableEvent>). This event is enabled only if a
** <interrupt service/event> is enabled.
** Parameters  :  None
** Returns     :  Nothing
*/

extern unsigned int pos, pos_ant;
extern unsigned int vel, vel_ant, vel_med;

void TI1_OnInterrupt(void) {
    EC161_GetNumEvents(&pos);

    if(pos >= pos_ant) {
        vel = ((pos – pos_ant)*1000*60)/512;
        vel_med = (vel + vel_ant)/2;
        pos_ant = pos;
        vel_ant = vel;
    }
    if(pos < pos_ant) {
        vel = (((65535 – pos_ant)+pos)*1000*60)/512;
        vel_med = (vel + vel_ant)/2;
        pos_ant = pos;
        vel_ant = vel;
    }
}

#ifdef __cplusplus
} /* extern "C" */
#endif

/* !
** @}
**/
/*
**
** This file was created by Processor Expert 10.5 [05.21]
** for the Freescale Kinetis series of microcontrollers.
**
**/

```
