UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ENGENHARIA MECÂNICA

LILLYANE RODRIGUES CINTRA

DESENVOLVIMENTO DE UMA INTERFACE BLUETOOTH PARA TRANSMISSÃO SERIAL EM TEMPO REAL APLICADA A ENCODERS

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO

2017

LILLYANE RODRIGUES CINTRA

DESENVOLVIMENTO DE UMA INTERFACE BLUETOOTH PARA TRANSMISSÃO SERIAL EM TEMPO REAL APLICADA A ENCODERS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso de Engenharia Mecânica da Universidade Tecnológica Federal do Paraná - UTFPR, como requisito parcial para obtenção do título de Bacharel.

Orientador: Prof. Dr. Marcio Aurélio Furtado Montezuma

CORNÉLIO PROCÓPIO

2017

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Mecânica
Curso de Engenharia Mecânica

TERMO DE APROVAÇÃO

DESENVOLVIMENTO DE UMA INTERFACE BLUETOOTH PARA TRANSMISSÃO SERIAL EM TEMPO REAL APLICADA A ENCODERS

POR

LILLYANE RODRIGUES CINTRA

Este trabalho de conclusão de curso foi apresentado às 13:50hs do dia 14 de novembro de 2017, como requisito parcial para a obtenção do título de ENGENHEIRA MECÂNICA, linha de pesquisa – Mecânica dos Sólidos, no programa de Graduação em Engenharia Mecânica 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. Dr. Marcio Aurélio Furtado Montezuma – Orientador (UTFPR)

_____
Prof. Me. Conrado Di Raimo – (UTFPR)

_____
Prof. Me. Lucas Niro – (UTFPR)

“A Folha de aprovação assinada encontra-se na Coordenação de Curso.”

Dedico este trabalho aos meus pais, por todo apoio e incentivo durante o curso.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por sempre me dar forças para vencer os obstáculos.

Aos meus pais Mauro Sérgio Cintra e Lucineia Rodrigues Cintra por todo carinho, incentivo e por sempre acreditarem em mim.

Ao meu orientador prof. Dr. Marcio Aurélio Furtado Montezuma pela confiança e por todo suporte na realização deste trabalho.

A todos do Laboratório de Sistemas Automatizados e Controle da Universidade Tecnológica Federal do Paraná - Campus Cornélio Procópio, pelas inúmeras contribuições na realização deste trabalho.

Aos meus amigos, pela amizade e motivação durante todo o curso.

Ao meu namorado Jairo Moura, por toda paciência e colaboração durante a realização deste trabalho.

E a todos que contribuíram de alguma forma para que este trabalho pudesse ser realizado.

“It is the time you spent on your rose that makes your rose so important” (Antoine de Saint-Exupéry, 1943).

“Foi o tempo que dedicaste à tua rosa que a fez tão importante” (Antoine de Saint-Exupéry, 1943).

RESUMO

CINTRA, Lillyane Rodrigues. **Desenvolvimento de uma interface Bluetooth para transmissão serial em tempo real aplicada a encoders**. 2017. 61 f. Trabalho de Conclusão de Curso – Engenharia Mecânica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2017.

Existem sistemas com eixos rotativos onde a necessidade de garantia tanto da velocidade de rotação correta, quanto da angulação exata, ou ainda do posicionamento necessário, pode ser de extrema importância. Para este controle o uso de encoder acoplado ao eixo é muito comum. Porém, há um possível problema quando se faz referência à transmissão de dados do encoder, pois essa transmissão é realizada por meio de fios e isso pode não ser desejado em alguns ambientes. Este projeto vem então atender à necessidade do uso de encoders em ambientes onde o uso de fios nas proximidades do eixo é inviável ou indesejável, através do uso da interface Bluetooth. No projeto desenvolvido, os dados do encoder passam por um microcontrolador que os transmite via Bluetooth para outro microcontrolador, que é responsável por interpretar os dados e fazer a transmissão por meio do protocolo RS232 para a plataforma MATLAB/Simulink, para exibição dos dados coletados. É possível também exibir simultâneamente os dados no MATLAB e em um display LCD, para auxiliar em casos onde a transmissão por RS232 tenha distância considerável e seja necessária a verificação das condições do eixo em distância menor.

Palavras-chave: Interface Bluetooth. Transmissão serial. Microcontrolador.

ABSTRACT

CINTRA, Lillyane Rodrigues. **Development of an interface Bluetooth to serial transmission in real time applied to encoders**. 2017. 61 f. Trabalho de Conclusão de Curso – Engenharia Mecânica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2017.

There are systems with rotating shafts where the need for assurance of both the correct rotation speed, of the exact angulation, or even the necessary positioning, can be extreme important. To this control the use of encoder coupled to the shaft is very common. However, there is a possible problem when referring to encoder data transmission, because this transmission is realized by means of wires and this may not be desired in some environments. This project comes then to answer to the need of the use of encoder in environments where the use of wires in the vicinity of the shaft is infeasible or undesirable, through the use of Bluetooth interface. In the developed project, the encoder data passes by a microcontroller that transmits them via Bluetooth to another microcontroller, which is responsible to for interpreting the data and to do the transmission through the RS232 protocol to the platform MATLAB/Simulink, for exhibition of the data collected. It is also possible to simultaneously exhibit the data in the MATLAB/Simulink and in an LCD display, to assist in cases where the RS232 transmission has a considerable distance and it is necessary the verification of the conditions of the shaft in shorter distance.

Keywords: Bluetooth Interface. Serial transmission. Microcontroller.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

I2C	Circuito Inter-Integrado
ICSP	Programação em Série em Circuito
LCD	Display de Cristal Líquido
PCI	Placa de Circuito Impresso
PIC	Controlador Integrado de Periféricos
PLL	Elo Travado por Fase
PWM	Modulação por Largura de Pulso
RAM	Memória de Acesso Aleatório
ROM	Memória de Apenas uma Leitura
UART	Transmissor/Receptor Assíncrono Universal
ULA	Unidade Lógica Aritmética
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

O ser humano sempre esteve em busca de vencer obstáculos gerados pela natureza, e assim, passou a desenvolver equipamentos que facilitassem o seu dia-a-dia. Foi inventando e reinventando que hoje a humanidade conta com milhares de equipamentos e tecnologias disponíveis, para as mais diversas funções. Um exemplo disso é o microcontrolador PIC (*Peripheral Interface Controller*), que para Souza (2013) é um conjunto de elementos integrados que em um único equipamento armazena todos os circuitos indispensáveis para a montagem de um sistema digital perfeito e consegue fazer, de forma rápida e com poucos itens externos, o que seria árduo realizar com circuitos tradicionais.

Há também a tecnologia Bluetooth que foi criada para satisfazer necessidades de comunicação a curtas distâncias, com baixo custo e consumo de energia, alto nível de segurança e sem a utilização de fios, tornando os sistemas mais simples e eficientes (SILVA, 2009).

Este projeto irá utilizar os dois exemplos acima citados, microcontrolador e Bluetooth, como elementos principais para realizar a aquisição de dados de encoders.

A aquisição de dados é algo muito comum no nosso dia-a-dia, porém nem todos têm essa consciência. Somos portadores de um complexo sistema de aquisição de dados que nos permite obter amostragens de luminosidade, cheiro, temperatura, entre várias outras. Através da amostragem recebida, conseguimos tomar decisões que favoreçam o bem-estar de nosso corpo e nossa mente. De forma semelhante, no âmbito da engenharia é preciso coletar dados para ter a garantia do bom funcionamento dos sistemas e, muitas vezes, realizar o controle destes, de modo a favorecer o bem-estar geral entre os componentes do sistema, a função que o sistema deve realizar e as pessoas ao seu redor (BAPTISTA, 2017).

Como a aquisição de dados é fundamental para sistemas de controle, este trabalho se aplica no meio industrial bem como no meio acadêmico, pois facilita a aquisição de dados tanto para certos ambientes da indústria quanto para diversos sistemas utilizados para ensino e pesquisa no meio acadêmico.

O sistema desenvolvido possibilita captar os dados do encoder e enviá-los por meio

de comunicação sem fio para uma Placa de Circuito Impresso (PCI) que esteja próxima de um computador, onde esses dados são interpretados na plataforma Matlab/Simulink em tempo real, por meio da extensão *Real-time Windows Target*. A transmissão entre a PCI e o computador se dá através do protocolo RS232, permitindo uma distância maior entre o computador e o encoder.

Dessa forma, o trabalho desenvolvido minimiza a quantidade de fios necessários para a coleta dos dados de encoders, diminui a poluição visual do sistema e o torna mais versátil.

1.1 Problema

O uso de encoders pode ser observado em inúmeras opções de máquinas e aparelhos como, por exemplo, esteiras, máquina de solda automática, entre outras. Em situações onde a posição ou a velocidade podem ser críticas para um processo, encoders são adotados para auxiliar no controle ou na supervisão do processo em questão (TIMÓTEO, 2013).

Porém pode haver sistemas onde o número de fios necessários para ligar o encoder possa gerar problemas devido à complexidade do processo ou do sistema, ou talvez até mesmo seja indesejável a presença de fios nas proximidades do sistema em decorrência do ambiente em que se encontra o processo que deve ser monitorado ou controlado. Nesses sistemas, como realizar a aquisição de dados de encoders?

1.2 Justificativa

Considerando o problema apresentado, este trabalho de conclusão de curso vem suprir a necessidade de um sistema capaz de captar dados de encoders e realizar a transmissão desses dados por intermédio de comunicação sem fio.

Além disso, levou-se em consideração o custo dos materiais necessários e também o custo de operação, optando sempre por qualidade e menor custo.

1.3 Objetivos

Esta seção aborda o objetivo geral e os objetivos específicos deste trabalho.

1.3.1 Objetivo geral

Este trabalho de conclusão de curso tem como objetivo principal estudar e desenvolver um método de aquisição de dados de encoders por meio da tecnologia de comunicação sem fio,

Bluetooth.

1.3.2 Objetivos específicos

- Estudar, definir e caracterizar os elementos necessários para compor o sistema;
- Desenvolver a programação de microcontroladores no ambiente de desenvolvimento e compilador MikroC PRO for PIC;
- Utilizar o software EAGLE para o desenvolvimento de placas de circuito impresso (PCIs);
- Visualizar os dados em um display LCD alfanumérico e também na plataforma MATLAB/Simulink;
- Analisar os dados obtidos no sistema através do software MATLAB/Simulink.

1.4 Organização do texto

Este trabalho está organizado da seguinte forma, no Capítulo 2 está a fundamentação teórica, que aborda toda a teoria contida neste projeto. No Capítulo 3 é apresentada a metodologia deste trabalho, mencionando as tecnologias e métodos utilizados. O Capítulo 4 mostra as análises e os resultados obtidos. No Capítulo 5 é apresentada a conclusão sobre este trabalho e a indicação de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os principais assuntos teóricos que devem estar claramente definidos para o bom entendimento do desenvolvimento do projeto no próximo capítulo. A seção 2.1 é dedicada aos microcontroladores, enquanto que as seções 2.2 e 2.3 abordam comunicação serial e encoder óptico, respectivamente.

2.1 Microcontroladores

Segundo Maia (2016), o microcontrolador é um computador de pequena dimensão que pode ser encontrado em diversos aparelhos, como eletrodomésticos e eletroeletrônicos. Sua função é gerenciar dispositivos.

Três importantes características dos microcontroladores são: controle de processo lógico, capacidade programável e dispositivo de pequena proporção. Controle de processo lógico significa que o sistema realiza ações baseando-se nos estados de entrada e/ou saída dos periféricos, ou seja, para qualquer ação é necessário o controle de periféricos, como sensores, displays, botões, entre outros (MOLLON, 2016).

A capacidade programável está associada à programação do dispositivo, que se dá a partir da gravação de toda a lógica de operação no microcontrolador. Após a programação, que pode ser realizada várias vezes, sempre que o dispositivo receber alimentação o último programa gravado será executado. A capacidade citada também está diretamente ligada à Unidade Lógica Aritmética (ULA), que é responsável por realizar todas as funções de cálculo e manipulação de dados. Desta forma, o nível de informações que o sistema é capaz de processar depende diretamente da ULA e também da quantidade de ULAs disponíveis (MOLLON, 2016; SILVA, 2009).

O dimensionamento do microcontrolador o torna aplicável a inúmeros sistemas, pois possui um sistema computacional completo em um dispositivo de pequeno porte. Ou seja, ele possui de internamente todos os componentes necessários para realizar o controle de suas ações, como por exemplo, memórias RAM e ROM, *timers*, contadores, interrupções, portas I/O, con-

versores Analógico/Digital, e consequentemente ocupam um pequeno espaço e realizam muitas funções nos dispositivos onde são instalados. Possibilitam projetos compactos e requintados com baixo custo, se comparados aos microprocessadores A principal diferença entre um microcontrolador e um microprocessador são os vários periféricos integrados que o microcontrolador possui em um único chip (MAIA, 2016; SILVA, 2011).

Para este trabalho, o uso do microcontrolador é de suma importância, pois ele gerencia a comunicação entre as interfaces utilizadas, através da comunicação *Universal Asynchronous Receiver/Transmiter* (UART). A seguir são apresentadas com mais detalhes as funcionalidades do microcontrolador utilizadas neste trabalho.

2.1.1 Interrupções

É comum fazer leitura frequente do nível de um pino de entrada para saber o estado em que ele se encontra para, a partir daí realizar as ações correspondentes. Porém, há casos onde este método mais simples não satisfaz as condições do sistema devido à necessidade de resposta instantânea do processador para mudanças no nível lógico do pino. Para estes casos, o uso das denominadas interrupções, é o método mais indicado (PENIDO; TRINDADE, 2013).

As interrupções são eventos externos ao programa de execução que interrompem o microcontrolador de continuar no programa principal, e o forçam a executar uma determinada sub-rotina. Após o término da execução dessa sub-rotina em questão, o programa principal volta a ser executado a partir do ponto onde se encontrava antes da interrupção (SILVA, 2009).

Segundo Silva (2017), há diferentes níveis de prioridade para interrupções do tipo mascarável. Esses níveis são responsáveis por revelar à unidade de processamento qual a ordem de execução das interrupções no caso de duas interrupções ou mais serem chamadas em um único momento.

As interrupções do tipo mascarável permite que o sistema execute a interrupção de maior prioridade enquanto a de menor prioridade aguarda para ser executada na sequência, ou seja, a interrupção de baixa prioridade é “mascarada” (SILVA, 2017).

Quando a interrupção é do tipo não mascarável ela possui alta prioridade, e deve ser executada de imediato. É comumente utilizada para casos de queda de energia (SILVA, 2017).

Para o projeto desenvolvido, as interrupções são necessárias para fazer a contagem de temporização para envio da amostragem e para realizar a recepção dos dados através da comunicação UART.

2.2 Comunicação serial

Segundo Onofre (2014), é possível fazer a transmissão de dados entre duas localidades de um sistema de várias maneiras, porém as duas mais comuns são as comunicações paralela e serial.

A comunicação paralela envia *bits* simultaneamente por diferentes canais. Ela é normalmente mais rápida, pois mais dados são transmitidos ao mesmo tempo (ONOFRE, 2014).

Na comunicação serial apenas um canal é utilizado para fazer a comunicação. Os *bytes* são separados em bits e transmitidos uma a um por meio do canal de transmissão. Esse tipo de comunicação se aplica onde não há exigência de elevadas taxas de transmissão de dados e onde a distância requerida para a comunicação é extensa. Quando comparada à comunicação paralela, a comunicação serial possibilita mais simplicidade utilizando-se de um menor custo (COSTA, 2013; ONOFRE, 2014).

A comunicação serial é dividida em síncrona e assíncrona. A transmissão é síncrona quando um sinal de clock controla a transmissão de dados, e assíncrona quando não há o controle pelo sinal de *clock* (COSTA, 2013).

A comunicação serial UART que foi a utilizada neste trabalho, bem como seus protocolos, Bluetooh e RS232, são detalhados nos tópicos seguintes.

2.2.1 Universal asynchronous receiver/transmiter (UART)

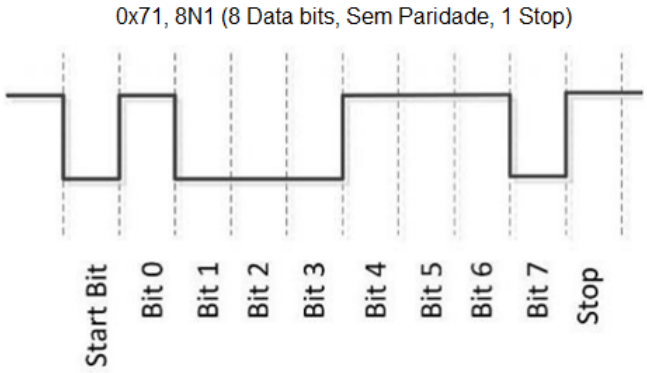
A comunicação serial UART foi criada entre 1960 e 1965 por Gordon Bell, e desde sua criação é amplamente utilizada. O módulo de transmissão e recebimento de dados desenvolvido por Gordon basicamente transformava os dados de transmissão de barramento paralelo para serial (CARVALHO, 2016).

A comunicação UART é definida como uma comunicação serial assíncrona, utilizada para transmissão de dados com periféricos externos em distâncias superiores àquelas onde se utiliza a comunicação síncrona. Por possuir um hardware relativamente simples, a UART deve ser usada quando se pretende uma comunicação de baixo custo e velocidade. Para a comunicação faz-se necessário o uso de apenas dois fios, sendo um para cada direção (TALIANI JUNIOR, 2012).

Segundo Carvalho (2016), como a UART é uma comunicação assíncrona, é necessário o envio de sinais de sincronismo pela mesma linha de transmissão de dados. Geralmente a comunicação é iniciada com um *start bit*, depois são enviados os *bits* de dados até que, se

houver, sejam enviados *bits* de paridade, se servem para verificar a sequência de dados recebida, e a finalização da comunicação é dada com *stop bits*. A configuração mais comum encontrada é com o *start bit*, 8 *bits* de dados, sem paridade e com 1 *stop bit*, conforme a figura 1. Esse tipo de comunicação é capaz de realizar transmissão *full-duplex*, ou seja, receptor e transmissor podem enviar dados simultaneamente em ambos os sentidos.

Figura 1: **Diagrama de Comunicação UART.**



Fonte: (CARVALHO, 2016) adaptado.

Antes de utilizar a UART é preciso configurar o módulo em todos os dispositivos. A velocidade para a transferência de dados, por exemplo, também conhecida como *baud rate*, é uma das configurações a se fazer. Essa velocidade de transmissão tem como limitante a capacidade de recepção do periférico com o qual se deseja fazer a comunicação e seu valor máximo é de 115,2 kbps (JUNIOR, 2012; COSTA, 2013).

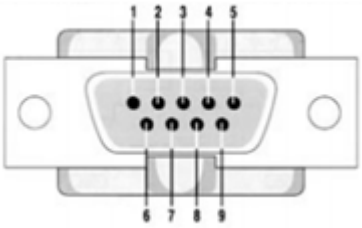
Devido à simplicidade de seu *hardware*, à transferência *full-duplex*, a sua distância de alcance e a sua capacidade de transmissão através do protocolo Bluetooth, a UART foi escolhida para compor este projeto,

2.2.1.1 RS232

Com o objetivo de fazer diretamente uma comunicação bidirecional de dados entre um microcomputador e um dispositivo - a uma distância máxima variando de 150m a 300m dependendo da taxa de transmissão e do cabo empregado - foi criado em 1962 o padrão RS232, que durante muitos anos ficou sendo o único padrão serial de computadores pessoais, cuja entrada era apenas do tipo RS232C (Figura 2) - onde o pino 1 detecta a presença de sinal em uma comunicação entre modems, o pino 2 é o canal por onde os dados são recebidos, o pino 3 é o canal por onde os dados são enviados, o pino 4 é responsável por avisar ao transmissor que o dispositivo está proto para receber dados, o pino 5 é o sinal de terra usado como referência pelos

outros pinos, o pino 6 é utilizado para informar o estado da linha telefônica em comunicação entre modems, o pino 7 habilita recepção ou seleção de direção em comunicação *half-duplex*, o pino 8 faz o controle de fluxo e solicita permissão para envio de dados, por fim, o pino 9 informa quando um sinal está sendo recebido para comunicações entre modems. Esse padrão utiliza um conversor de sinal para comunicar o microcontrolador com o microcomputador amplificando ou reduzindo sinais dos pinos RX e TX. A velocidade máxima atingida pelo padrão RS232 é de 0,014 Megabits por segundo. (SILVA, 2011; GUIMARÃES, 2003).

Figura 2: **Pinos do conector DB9 do microcomputador.**



Fonte: (SILVA, 2011) adaptado.

Neste trabalho o protocolo RS232 é responsável pelo envio dos dados do microntrolador para o Matlab/Simulink.

2.2.1.2 Bluetooth

Bluetooth é o nome dado ao protocolo, criado em 1994, capaz de efetuar comunicação sem fio entre dispositivos eletrônicos em curtas distâncias e com baixo custo. A única exigência para que os dispositivos consigam se comunicar é que ambos devem possuir o protocolo. Sua utilização tem aumentado consideravelmente em áreas como da saúde e desporto, pois é usado para monitoramento pessoal devido a sua portabilidade (CAMPOS, 2015; PEREIRA, 2016).

Através da interface Bluetooth os dispositivos se comunicam formando uma rede conhecida como piconet, onde podem estar conectados até oito dispositivos. Entre os oito dispositivos apenas um pode ser mestre. Para aumentar o número de dispositivos interligados torna-se necessário o aumento do número de mestres, que pode chegar, no máximo, a um total de 10. Possui capacidade de transmissão de dados bidirecional e as principais características são a confiabilidade, o baixo consumo e o baixo custo. (BONATTO; CANTO, 2017).

Segundo Campos (2015), a interface Bluetooth trabalha na banda dos 2,4 GHz e pode ser dividida em três classes, de acordo com a distância máxima de alcance:

- Classe 1: 100 metros;

- Classe 2: 10 metros;
- Classe 3: até 1 metro.

O uso da tecnologia Bluetooth age na transferência dos dados do encoder para o microcontrolador, onde se encontra o módulo RS232, e é o ponto forte na execução deste trabalho, pois depende dela toda a aplicação deste projeto.

2.2.1.3 Encoder óptico

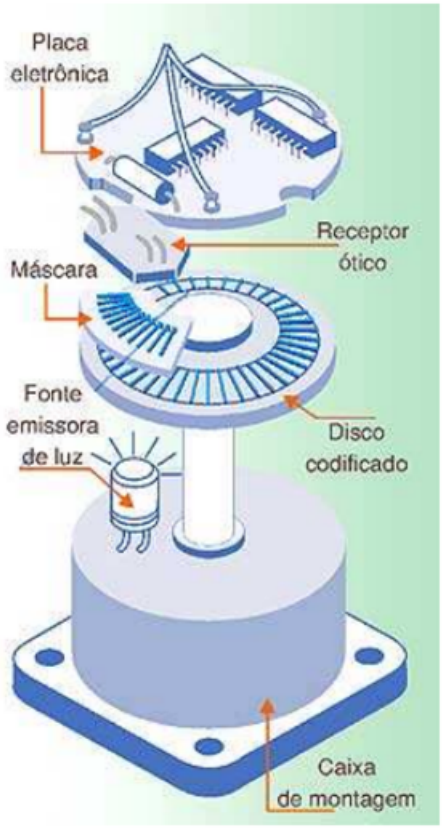
Encoders são sensores que transformam movimentos lineares ou angulares em pulsos elétricos gerando informações binárias, que são interpretadas como velocidade do eixo, posição e, em alguns casos, sentido de rotação. O encoder age como uma realimentação do sistema, informando a posição atual para que possa ser comparada com a posição que se deseja e então sejam possíveis movimentos planejados (FUSCO, 2009).

Existem dois tipos principais de encoders, o incremental e o absoluto. O incremental guarda a contagem dos pulsos na memória do controlador e se ocorrer a fata de energia, todos os dados podem ser perdidos. Normalmente equipamentos que trabalham com este tipo de encoder têm em sua rotina um comando para zerar a posição quando há o reestabelecimento de energia após uma queda. Por outro lado, o encoder do tipo absoluto guarda sempre a sua posição, até mesmo quando se encontra desligado, ou seja, uma queda de energia nunca afeta o valor da posição, e caso o eixo gire estando o encoder desligado quando a energia se reestabelece a posição é atualizada (SILVA, 2015).

Quando se deseja saber também o sentido de rotação é necessário que o disco do encoder tenha canal duplo, e que cada fissura seja defasada em 90^{o} elétricos de um canal para o outro (SILVA, 2015).

Geralmente são compostos por um disco de plástico ou vidro, com perfurações espaçadas igualmente em torno da circunferência, que gira, conforme a rotação do eixo ao qual está acoplado, entre uma fonte emissora de luz e um receptor óptico (Figura 3). Sempre que a luz atravessa o disco por um de seus feixes e chega até o receptor óptico é gerado um pulso, e o número de pulsos gerados é o que determina o posicionamento ou a velocidade de rotação (FUSCO, 2009; MOLLON, 2016).

Figura 3: **Constituição física do encoder.**



Fonte: (FUSCO, 2009) adaptado.

Para este trabalho o encoder a ser utilizado para os testes é do tipo incremental com canal duplo.

3 DESENVOLVIMENTO

Neste capítulo constam todas as etapas e métodos para a realização deste trabalho.

Na primeira etapa encontra-se a escolha dos componentes, pois é preciso saber quais elementos são necessários para conseguir iniciar quaisquer das outras etapas.

A segunda etapa é a fase de projeto e fabricação das PCIs (placas de circuito impresso), que consiste em, através do software Eagle e utilizando os componentes escolhidos na primeira etapa, montar os circuitos e o layout de cada PCI realizando na sequência a fabricação de cada uma delas.

A terceira etapa é responsável pelo desenvolvimento de toda a programação de micro-controladores utilizada neste trabalho através da plataforma MikroC.

A quarta e última etapa consiste no desenvolvimento da interface Matlab/Simulink com a utilização do *Windows Real-Time Target* para a exibição dos dados obtidos com o encoder em tempo real.

Na Figura 4 é possível observar as etapas nas quais o trabalho foi dividido. Cada seção deste capítulo, a partir da seção 3.2, se remete à explicação de uma dessas etapas.

Figura 4: **Etapas do trabalho.**

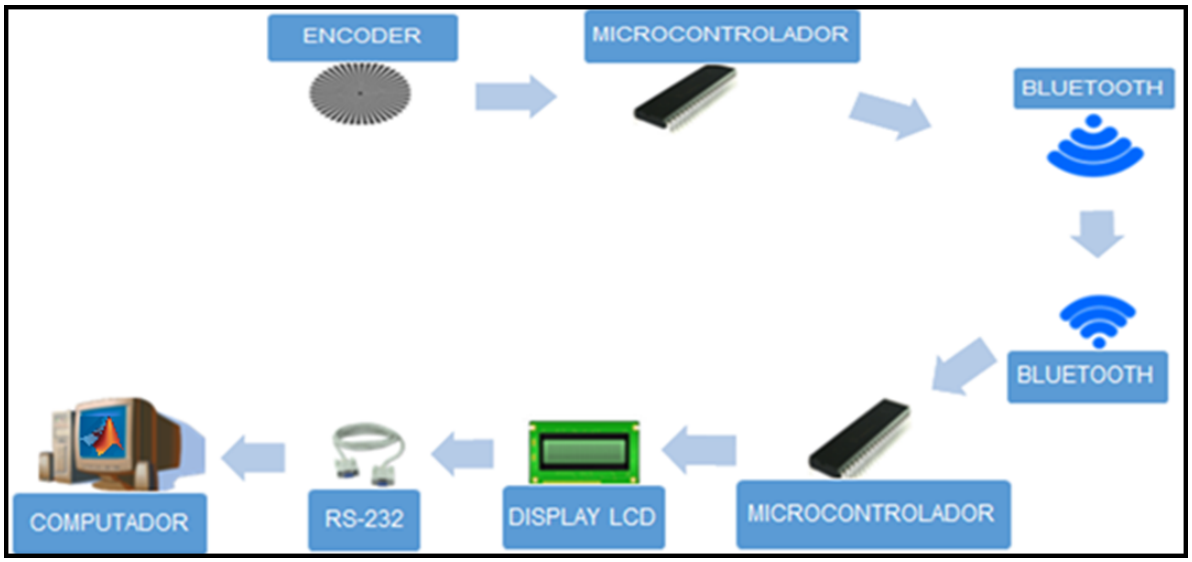
Fonte: Autoria Própria.

3.1 Visão geral do sistema

Nesta seção é apresentado de maneira simplificada todo o funcionamento do sistema com o objetivo de facilitar a compreensão do trabalho desenvolvido.

Na Figura 5 é possível observar os principais componentes e tecnologias para a realização do trabalho proposto.

Figura 5: **Diagrama simplificado do funcionamento do sistema.**



Fonte: Autoria Própria.

É possível notar o uso de dois microcontroladores que fazem o gerenciamento do sistema. O primeiro, com decodificador de quadratura interno, é responsável por coletar os dados do encoder e o segundo por interpretar os dados e enviá-los para fazer a exibição tanto no Matlab/Simulink, utilizando transferência por RS232, quanto em um display LCD, situado junto ao microcontrolador. A exibição também pode ser feita sem a utilização do segundo microcontrolador, através de uma segunda PCI que recebe os dados via Bluetooth e os transfere via RS232 (possibilitando maior distância entre o encoder e o computador) para serem exibidos somente no Matlab/Simulink.

Pode-se observar também a comunicação que ocorre entre alguns componentes. Entre os dois microcontroladores, ou entre o primeiro e a PCI que não possui microcontrolador, a comunicação é feita utilizando a interface Bluetooth. Por outro lado, o envio de dados para o Matlab/Simulink é feito através do protocolo RS232.

3.2 Escolha dos componentes

Nesta seção constam todos os componentes utilizados em cada PCI, e demais componentes necessários para o bom funcionamento do sistema.

A escolha de cada item levou em consideração fatores como: tamanho, praticidade e custo.

Para a PCI que deve ser ligada ao encoder, os seguintes itens são necessários:

- Bateria de lítio polímero de 7,4 V;
- PIC18F2331;
- Reguladores de tensão, para 5 V (LM7805) e para 3,3 V (AMS1117);
- Conversor de nível lógico, de 3,3 V para 5 V;
- Cristal de 7,3728 MHz;
- Chave para *switch power*;
- Diodo 1N4007;
- Resistores de 1kΩ, 1,8kΩ, 3,3kΩ e 10kΩ;
- Capacitores de 0,000022uF, 0,1uF e 0,33uF;
- Capacitor eletrolítico de 100uF;
- Conectores *pin head*.

Para a PCI que recebe os dados via Bluetooth e transmite por RS232, para exibição somente através do Matlab/Simulink, os itens fundamentais são:

- Bateria de lítio polímero de 7,4 V;
- Conversor RS232;
- Reguladores de tensão, para 3,3 V (AMS1117);
- Chave para *switch power*;
- Diodo 1N4007;

- Capacitores de 0,1uF e 0,33uF;
- Capacitor eletrolítico de 100uF;
- Conectores *pin head*.

O circuito que tanto envia por RS232 quanto exibe em display LCD foi montado apenas em protoboard, pois é simplesmente uma variação do sistema geral. Para seu funcionamento devem conter os seguintes componentes:

- Protoboard;
- Placa de desenvolvimento com o PIC18F4331 e cristal de 7,3728 MHz;
- Conversor RS232;
- Display LCD 16x2;
- Jumpers.

Para que fosse possível realizar testes com o sistema desenvolvido, um servomotor (que é um motor acoplado a um encoder) foi necessário. O modelo para utilização nos testes é composto por um motor da fabricante Pittman, de corrente contínua que aceita alimentação de até 19 V e rotação máxima de 10.000 RPM, e por um encoder do modelo HEDS-5310 da fabricante HP, que possui 512 linhas e aceita tensão de operação de 3,3 V e 5 V.

3.2.1 Microcontrolador PIC18F2331

O microcontrolador PIC18F2331 possui 28 pinos, incluindo pinos para PWM (*Pulse Width Modulation*), comunicação I2C (*Inter-Integrated Circuit*), comunicação UART (*Universal Synchronous Receiver/Transmitter*) e, dentre outros, pinos para decodificação de quadratura, que foi o principal motivo para o uso deste microcontrolador no projeto. O custo benefício também foi levado em consideração.

Para a decodificação de quadratura existem dois modos de contagem de quadratura aceitos por este PIC, X2 e X4. Para este projeto, o modo selecionado foi o X4. Dessa forma, as 512 linhas do encoder são multiplicadas por quatro resultando que para uma volta completa do encoder sejam necessários 2048 pulsos do encoder.

A inclusão de um decodificador de quadratura neste microcontrolador permitiu a redução do circuito que é ligado ao encoder, de maneira a deixar o sistema mais prático e aplicável independentemente do ambiente.

3.2.2 Cristal oscilador

A escolha do valor do cristal levou em consideração a frequência de clock do PIC18F2331, que é o PIC responsável por coletar os dados do encoder, e seu valor de PLL (*Phase Loked Loop*). Como se deseja trabalhar com alta velocidade de transmissão, considera a máxima frequência de clock que o PIC consegue alcançar, que é de 40 MHz, e o máximo PLL que é 4, segundo Microchip (2017). Para o cálculo de máxima frequência possível do cristal a se utilizar tem-se a Equação (1):

$$A = \frac{B}{PLL} \tag{1}$$

Onde:

A é a frequência máxima que o cristal pode atingir;

B é a máxima frequência do PIC18F2331, ou seja, 40 MHz;

PLL é *Phase Loked Loop*, com valor igual a 4.

Sendo assim, obtêm-se o valor de A como sendo 10 MHz. Porém, é preciso verificar a porcentagem de erro que um cristal desse valor poderia gerar para uma transmissão UART de 115200 bps, que é o *baud rate* desejado. Para a determinação desse erro em porcentagem, são necessárias as equações (2), (3) e (4), que foram extraídas do roteiro de configuração de *baud rate* em Microchip (2017). Os resultados das equações devem ser arredondados.

$$X = \frac{Fosc/baud\ rate\ desejado}{64} - 1 \tag{2}$$

$$baud\ rate\ calculado = \frac{Fosc}{64 * (X+1)} \tag{3}$$

Em que Fosc é a frequência do oscilador para as Equações (2) e (3).

O erro, em porcentagem, é finalmente encontrado na equação (4):

$$Erro = \frac{baud\ rate\ calculado - baud\ rate\ desejado}{baud\ rate\ desejado} \tag{4}$$

Pela porcentagem de erro é possível perceber que o cristal oscilador mais apropriado para ser utilizado neste caso é que possui 7,3728 MHz de frequência, pois o erro gerado na transmissão com 115200 bps é de 0,00%. O cristal escolhido é o que mais se aproxima do valor

de 10 MHz com porcentagem de erro igual a zero para o sistema. Neste caso, a velocidade da transmissão é de 7,3728 mips (milhões de instruções por segundo).

3.2.3 Conversor de nível lógico (3,3V - 5V)

Foi necessário o uso de um conversor de nível lógico para 5 V para que o sistema desenvolvido fosse capaz de realizar a leitura de encoders com alimentação tanto de 5 V quanto de 3,3 V. Como o microcontrolador PIC18F2331 trabalha em 5 V, seria inviável fazer a comunicação dele com um encoder alimentado com tensão menor, sem o uso de um conversor.

O conversor escolhido para este trabalho foi o LVC4245A, da Texas Instruments. Ele possui 16 pinos, os quais são divididos em dois *ports* com 8 pinos cada. O *port* A pode ser alimentado com tensão entre 4,7 V e 5,5 V e o *port* B pode receber tensão entre 2,7 V e 3,6 V. Outra característica interessante deste componente é que ele consegue realizar conversão bidirecional (TEXAS INSTRUMENTS, 2015).

Para utilizar esse conversor é importante observar o nível lógico que deve ser aplicado em alguns pinos, para isso a Texas Instruments (2015) fornece alguns dados que podem ser visualizados na Tabela 1.

Tabela 1: **Configuração de pinos do conversor**

Entradas		Operação
OE	DIR	
L	L	Dados de B para A
L	H	Dados de A para B
H	X	*Ports* isolados

Fonte: (TEXAS INSTRUMENTS, 2015).

Como para o sistema desenvolvido a conversão é necessária para aumentar a tensão vinda dos pinos do encoder, a operação requerida é de que os dados entrem no *port* B e saiam no *port* A, ou seja, de 3,3 V para 5 V. Sendo assim, pela tabela 1 pode-se observar que os pinos OE e DIR devem receber nível lógico baixo, pois L e H significam *low* e *high*, respectivamente, e X determina o pino que não deve ser conectado. Para nível lógico baixo, a ligação deve ser feita ao GND (ground) e para nível lógico alto deve receber a tensão de 3,3 se o pino estiver no *port* B ou 5 V se o pino estiver no *port* A.

3.2.4 Conversor RS232

Para a transmissão UART por protocolo RS-232 fez-se uso do conversor RS232 *Board*, da fabricante Waveshare. Esse conversor recebe alimentação de 3,3 V a 5 V e pode ser considerado de pequeno porte, facilitando assim seu uso no projeto. A função deste conversor é receber os dados que chegam através do Bluetooth e, através do componente SP3232, realizar a transmissão dos dados por comunicação RS-232 para o computador. Na Figura 6 pode-se observar o conversor utilizado.

Figura 6: **RS232 Board.**

Fonte: (WAVESHARE, 2017).

3.2.5 Bluetooth

No sistema desenvolvido a tensão de alimentação dos dois módulos Bluetooth é de 3,3 V. Sendo assim, adotou-se o modelo DF-BluetoothV3 para o desenvolvimento do projeto e para os testes finais. Esse modelo contém dois pinos destinados à recepção de tensão, sendo um deles para tensão de 3,3 V e outro para tensão de 5 V. O uso desse modelo durante o desenvolvimento do sistema não implica na obrigatoriedade de seu uso, o modelo HC-05, por exemplo, poderia substituí-lo.

Para a configuração de parâmetros do modelo escolhido o software a ser utilizado é o CoolTerm, da SparkFun.

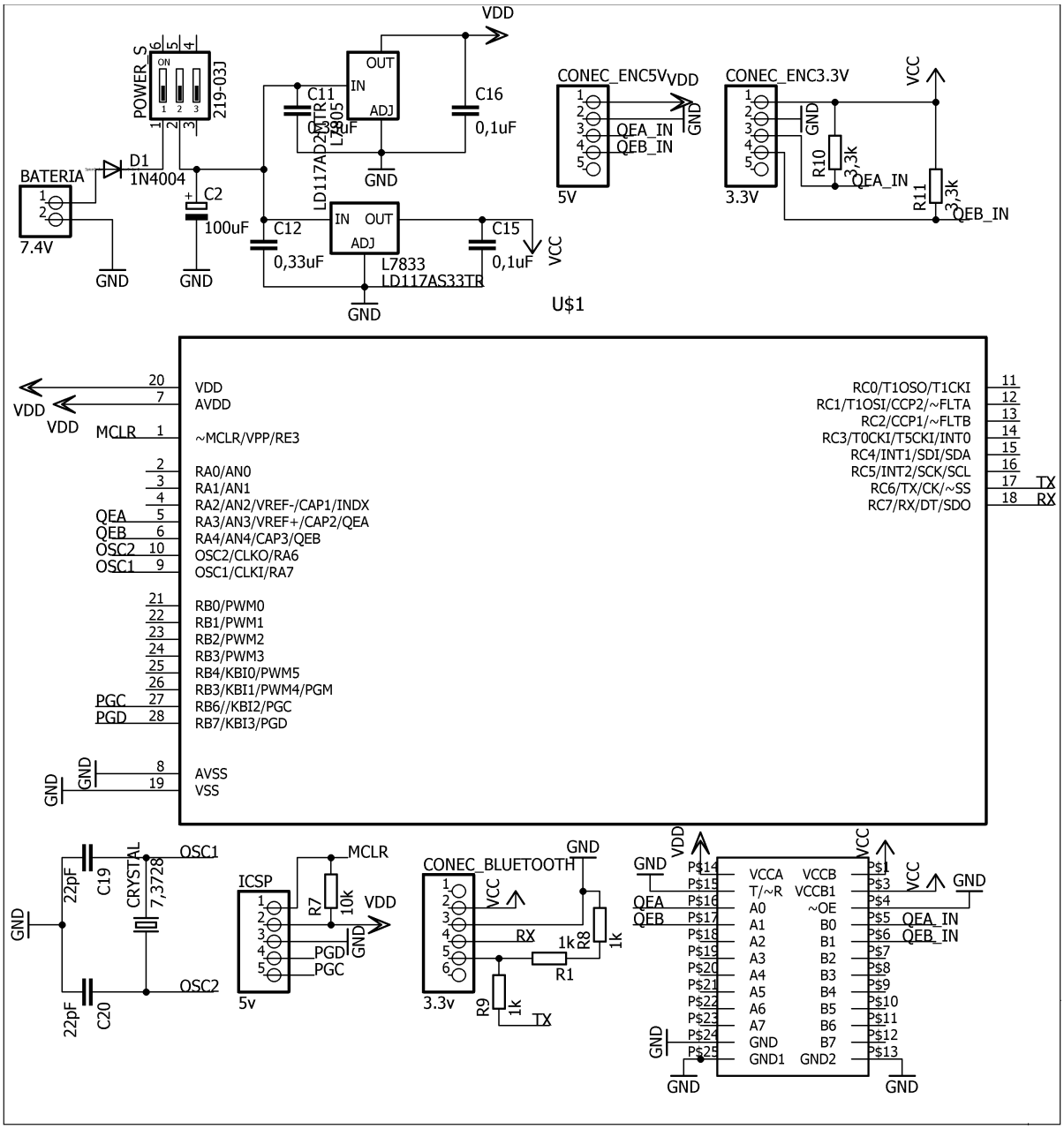
3.3 Projetos das PCIs

Para o projeto e desenvolvimento das PCIs (placas de circuito impresso) foi utilizado o *software* Eagle, versão 7.7.0, que permite desenhar todo o esquemático das placas e também montar os layouts para impressão.

Os esquemáticos dos circuitos necessários ao projeto podem ser visualizados a seguir,

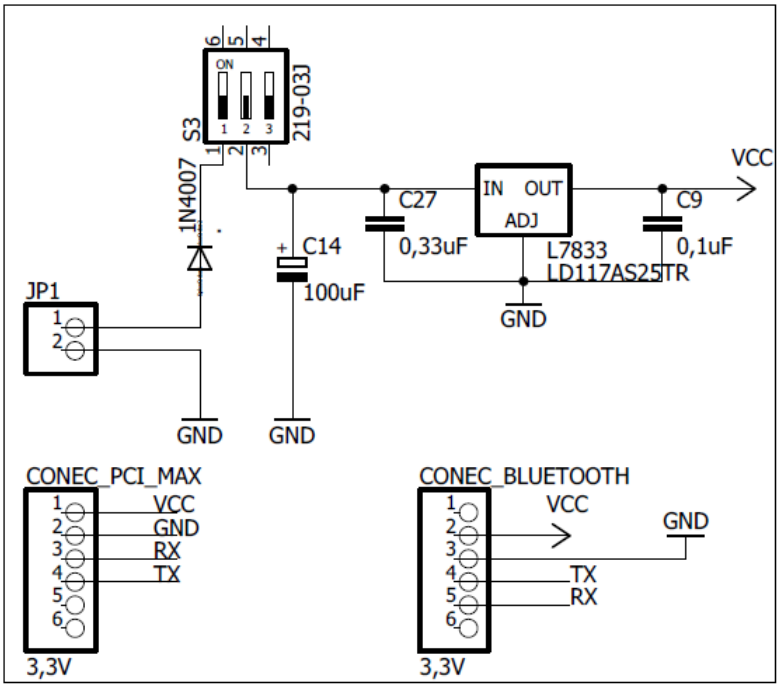
sendo que a primeira PCI é a placa que deve ser ligada ao encoder e a segunda deve ser escolhida mediante necessidade ou não do display.

Figura 7: **Esquemático da primeira PCI (para ligar ao encoder).**



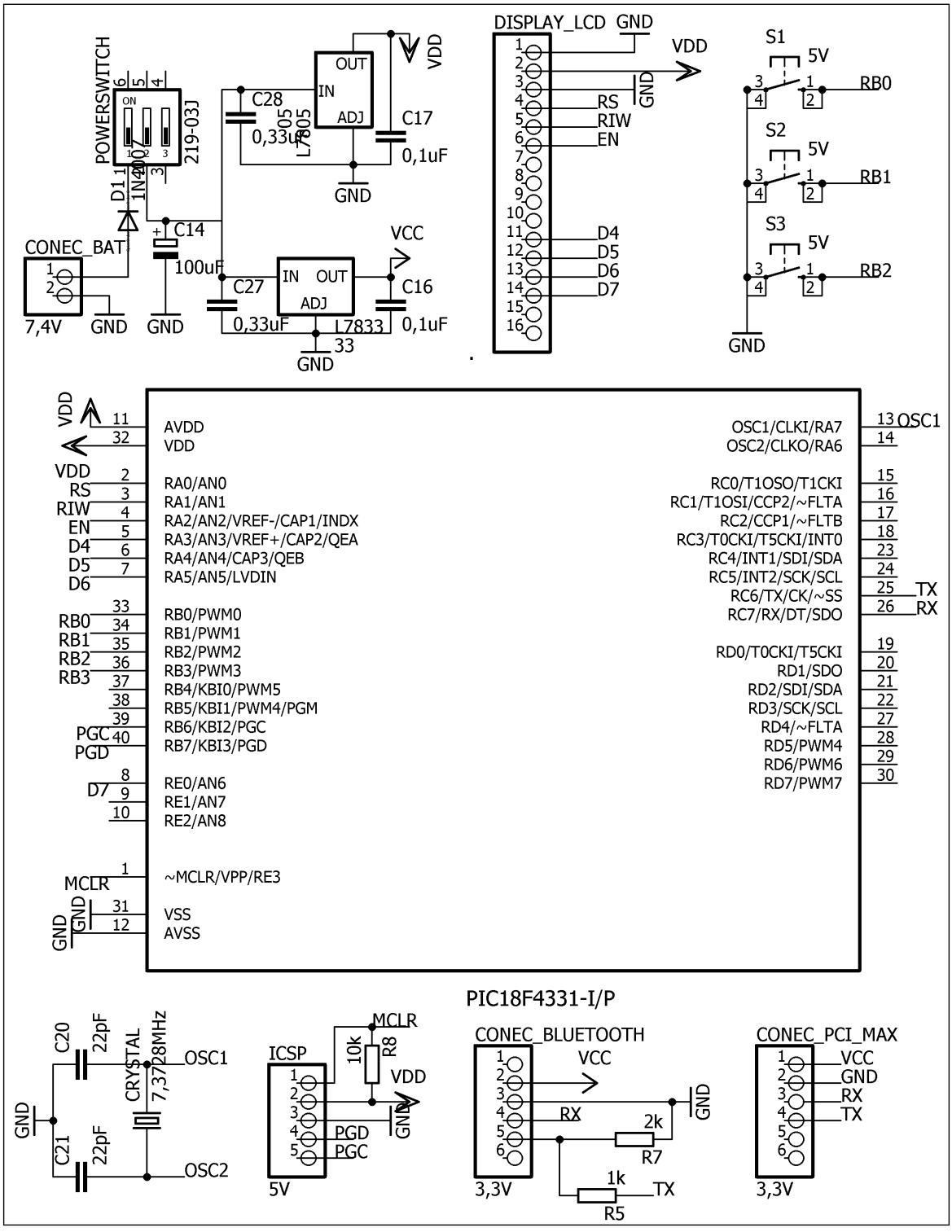
Fonte: Autoria Própria.

Figura 8: **Esquemático da segunda PCI (opção sem display).**



Fonte: Autoria Própria.

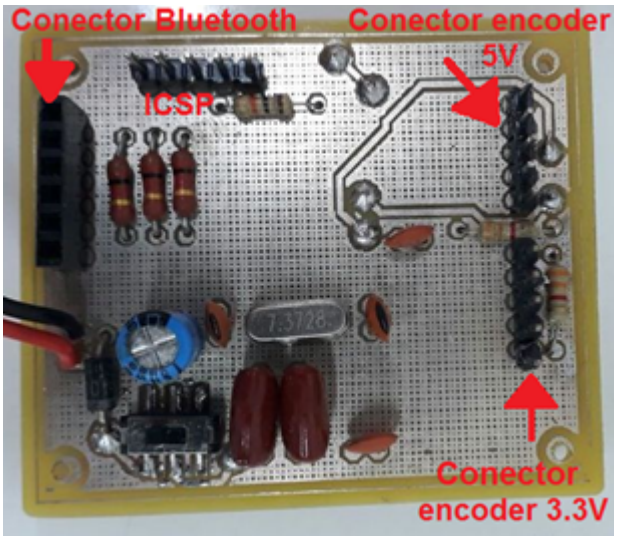
Figura 9: **Esquemático da segunda PCI (opção com display).**



Fonte: Autoria Própria.

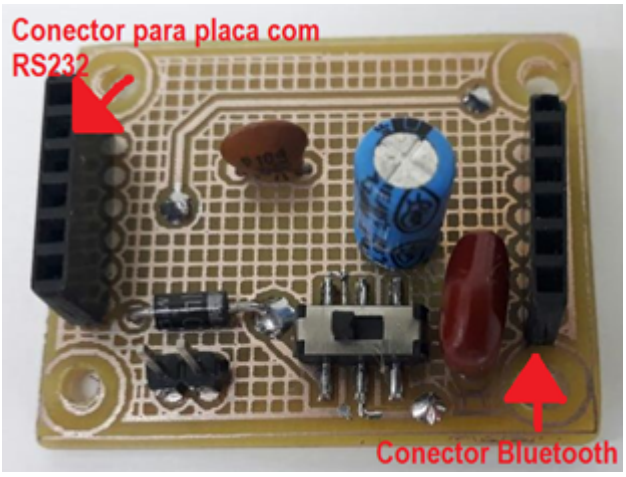
As PCIs desenvolvidas neste projeto, após todos os processos de fabricação, podem ser vistas nas Figuras 10 e 11.

Figura 10: **Primeira PCI (para ligar ao encoder).**



Fonte: Autoria Própria.

Figura 11: **Segunda PCI (opção sem display).**



Fonte: Autoria Própria.

3.4 Desenvolvimento das rotinas de programação

As rotinas de programação dos microcontroladores foram desenvolvidas no ambiente de desenvolvimento e compilador MikroC PRO for PIC versão 6.6.1, fazendo uso da linguagem C. O MikroC PRO for PIC possui algumas vantagens que auxiliaram no desenvolvimento das programações, como bibliotecas, conversor *Quick Converter* e recurso *Edit Project*.

Muitas bibliotecas são inclusas no ambiente de desenvolvimento e dispensam o uso da diretiva #include. Elas podem ser selecionadas de acordo com a necessidade de cada programa e são automaticamente incluídas no projeto.

O conversor *Quick Converter* é um recurso que realiza conversão de números, de até 32 bits, para binário, decimal e hexadecimal. Desta forma, é possível trabalhar com diferentes tipos de números sem fazer uso de ferramentas externas para a conversão. A partir da declaração de um número, que pode estar tanto em binário, quanto em decimal ou hexadecimal, a conversão se dá automaticamente nas outras duas formas.

A opção *Edit Project* é uma maneira simples de configurar vários parâmetros de um projeto. Para cada um dos parâmetros presentes neste recurso todas as configurações possíveis são mostradas, a fim de que o usuário selecione a que melhor possa se adaptar no projeto a ser desenvolvido. É também no *Edit Project* que o microcontrolador a ser programado deve ser selecionado, e a frequência de *clock* declarada.

Foram necessárias duas rotinas, a primeira para o microcontrolador ligado ao encoder, a segunda para o microcontrolador que tanto envia os dados para o Matlab/Simulink quanto exibe por meio do display LCD. As duas rotinas de programação se encontram nos Anexos A e B deste trabalho.

Para o uso da UART, é importante definir o *baud rate* nos códigos de programação. Todos os dispositivos ou interfaces a se comunicar pela UART devem receber o mesmo valor de *baud rate* nos códigos, para que seja possível a comunicação. O valor de *baud rate* escolhido para este projeto é de 115200 bits/s, um valor considerado alto, que permite a transmissão em alta velocidade sem perda de dados.

3.4.1 Programador PICkit 2

Para escrever nos microcontroladores as rotinas desenvolvidas tornou-se necessário o uso de um programador, para isso foi usado o PICkit 2 da fabricante Microchip. Ele utiliza cinco pinos que são conectados aos pinos do microcontrolador mostrados na tabela 2. Os cinco pinos destinados à programação são facilmente identificáveis nas PCIs e fazem parte do circuito conhecido como ICSP (*In-Circuit Serial Programming*), que é o circuito destinado à programação serial interna em cada placa de circuito impresso (MICROCHIP, 2017b).

Tabela 2: **Configuração dos pinos do ICSP.**

Pinos ICSP	Pinos do microcontrolador
1	MCLR/VPP
2	VDD
3	VSS (GND)
4	PGD
5	PGC

Fonte: (MICROCHIP, 2017b).

O PICkit 2 faz uso de um dispositivo, que pode ser visualizado na Figura 12, para transmitir a programação que sai do computador via USB e se destina aos pinos já citados do microcontrolador.

Figura 12: **Dispositivo PICkit 2 para programação.**

Fonte: (MICROCHIP, 2017b).

Na Figura 12 são mostrados alguns pontos importantes do dispositivo usado na programação. A seguir são identificados cada um dos pontos sinalizados.

- 1: LEDs de estado;
- 2: *Push Button*;
- 3: Gancho para cordão;
- 4: Conector USB;
- 5: Identificação do pino 1 (MCLR);
- 6: Conector da programação.

Para a programação, o usuário deve seguir os seguintes passos:

1 - Conectar o dispositivo PICkit 2 no ICSP e no computador;

2 - Abrir, no computador, o *software* de programação PICkit 2;

3 - Observar no campo "*device*" se o microcontrolador a ser programador foi encontrado;

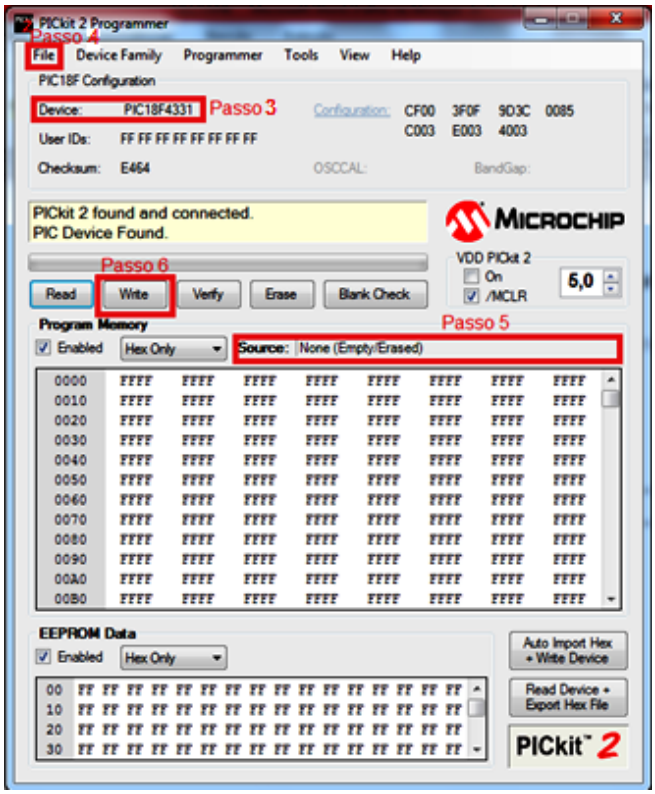
4 - Ir em "*File*", selecionar a opção "*Import Hex*", encontrar e selecionar o arquivo .hex a ser escrito no microcontrolador;

5 - Verificar no campo "*Source*" se o arquivo selecionado é o correto;

6 - Clicar em "*Write*" e esperar a programação ser concluída.

A Figura 13 mostra a interface para a programação.

Figura 13: **Interface de programação do PICkit 2.**



Fonte: Autoria Própria.

3.5 Desenvolvimento da interface Matlab/Simulink

O uso da interface é necessário para fazer o controle do tempo da amostragem e garantir a exibição em tempo real dos dados obtidos no encoder. É ela também que exibe o gráfico da

posição mostrando que os dados estão sendo coletados corretamente.

Antes de abordar o desenvolvimento da interface, é importante salientar alguns pontos sobre o Matlab e o Simulink.

O Matlab é um software de alto nível que permite a manipulação e compilação de funções matemática, construção de gráficos, resolução de problemas complexos em tempo muito inferior ao que se gastaria para encontrar a solução do problema de outro modo ou em outro software. Possui um ambiente de fácil utilização e com diversas bibliotecas para auxiliar o usuário (BECKER, 2010).

O Simulink é uma das ferramentas do Matlab e possui um ambiente que trabalha somente com diagrama de blocos. Gerar códigos automaticamente, realizar simulações, verificar sistemas embarcados e exportar resultados de simulações para o Matlab são algumas das suas várias funções (MATHWORKS, 2017). Uma ferramenta do Simulink importante para este trabalho é a *Real-time Windows Target*, que realiza a execução dos modelos do Simulink em tempo real em desktop ou laptop do Windows ou Mac (MATHWORKS, 2017c).

Para que o sistema fosse capaz de fazer a leitura de encoder em tempo real, a interface montada no Simulink foi desenvolvida na extensão *Real-time Windows Target*. Para o projeto desenvolvido, essa extensão envia um byte para o microcontrolador ligado ao encoder e espera a resposta do sistema, que é o valor obtido do encoder naquele exato momento. Para conseguir utilizar essa ferramenta, o programa foi desenvolvido no sistema operacional Windows XP e com o MATLAB/Simulink 2012a.

Os blocos utilizados do Simulink e suas respectivas funções, extraídas de MathWorks (2017b), são descritos a seguir:

- *Constant*: Gera um valor constante, real ou complexo. É responsável por escrever um valor fixo para ser enviado à serial e também para o denominador da operação de divisão que calcula a posição do encoder.
- *Packet Output*: Pertencente à extensão *Real-time Windows Target*. Escreve um dado em binário no canal de comunicação. É esse bloco que envia o valor vindo do bloco constante para a serial.
- *Packet Input*: Pertencente à extensão *Real-time Windows Target*. Lê os dados, em binário, recebidos do canal de comunicação. È esse bloco que recebe os dados vindos da comunicação serial.
- *Data Type Conversion*: Converte sinais de entrada para um tipo de dado específico. É

responsável por converter os dados de maneira que se tornem compreensíveis ao usuário.

- *Display*: Mostra os valores durante a simulação.
- *Scope*: Display para exibição de gráfico durante a simulação.
- *Gain*: Multiplica o valor que o bloco recebe por uma constante. Faz multiplicação do valor do encoder por uma constante resultante de cálculos para transformar o valor do encoder no valor da posição.
- *Math Function (Mod)*: Executa uma função matemática, neste caso, a função MOD, que é responsável por fazer a divisão do valor obtido no encoder por um valor obtido no bloco constant e apresentar como resultado o resto dessa divisão.

3.5.1 Configuração da interface

Para o bom funcionamento do sistema é essencial à configuração de alguns parâmetros na interface. A seguir são mencionados todos os parâmetros e valores que devem ser selecionados.

O valor de *baud rate* deve ser definido também nessa interface e, sendo assim, o usuário deve selecionar na aba "*Simulation*" o recurso "*Configuration Parameters*" e clicar no item "*Board Setup*", que abrirá uma nova janela onde há o parâmetro "*Baud rate*" e seus possíveis valores. Esse parâmetro deve receber o mesmo valor que as rotinas de programação no MikroC, ou seja, 115200 bits/s.

Ainda no recurso "*Configuration Parameters*", o "*Type*" deve ser definido como "*Fixed-step*", o "*Solver*" como "*Discrete (no continuous states)*" e o *stop time* como "inf".

O tempo de amostragem deve ser determinado no recurso "*Configuration Parameters*" e também nos dois blocos da extensão *Real-time Windows Target*, no campo "*Sample Time*", e para este projeto foi definido como sendo 1/500 s.

Ainda na aba "*Simulation*", porém no recurso "*Code Generation*", a opção "*System Target File*" deve ser configurada como "rtwin.tlc". No recurso "*Hardware Implementation*" a opção selecionada para "*Device Type*" deve ser "*32-bit x86 compatible*".

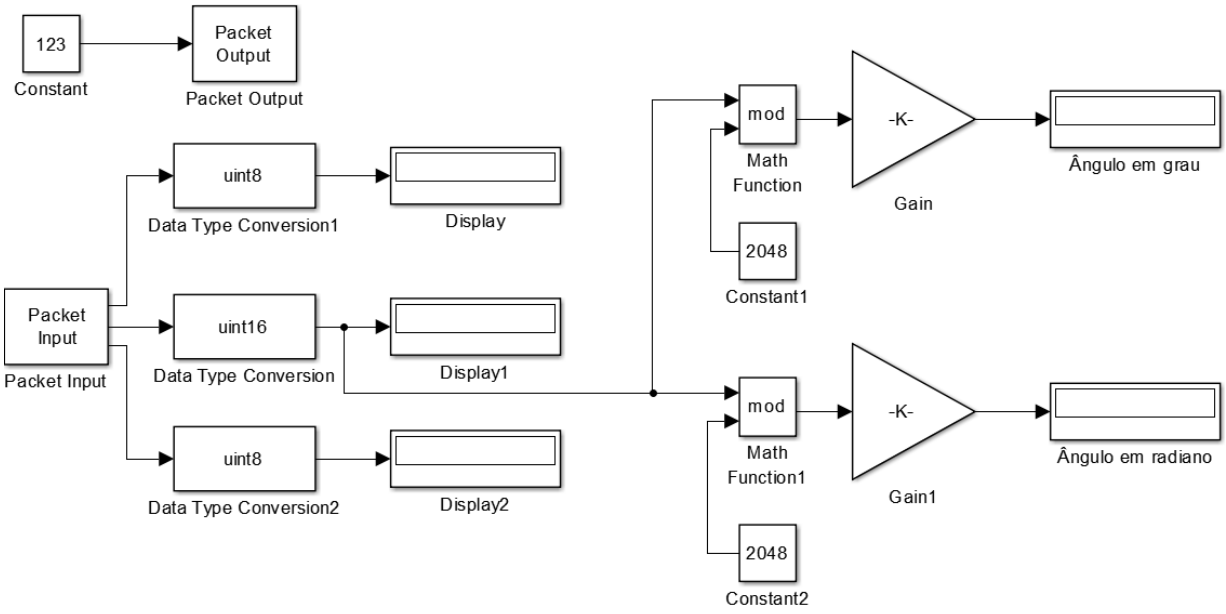
O usuário deve também selecionar, em "Simulation mode", a opção "*External*".

O bloco "*Packet Output*" deve receber o valor "1" em "*Output packet size*", que é a quantidade de *bytes* enviados, e "uint8" em "*Output packet field data types*", pois são 8 *bits* que serão enviados a cada intervalo de tempo.

O bloco “*Packet Input*” deve receber o valor “4” em “*Input packet size*”, pois recebe 4 *bytes*, e “uint8”,“uint16”,“uint8” em “*Block output data types*”, pois são os *bits* que serão direcionado a cada saída deste bloco, respectivamente. Em uint16 o segundo e terceiro *bytes* são unidos e o resultado dessa junção é enviado para a saída 2 do bloco.

A interface desenvolvida no Matlab/Simulink pode ser visualizada na figura 14.

Figura 14: **Interface desenvolvida no Matlab/Simulink.**



Fonte: Autoria Própria.

3.5.2 Funcionamento da interface

A interface possui funcionamento simples e sua descrição é feita a seguir.

O bloco, *Packet Output*, realiza o envio de um *byte* para o sistema a cada intervalo de tempo definido pelo usuário (0,002s neste trabalho). O *byte* enviado é a constante obtida no bloco Constant, que para este trabalho foi definido como sendo “123” (representação em decimal) ou “{” (representação em ASCII).

Quando o PIC18F2331 recebe o *byte* enviado pelo Matlab/Simulink, ele então retorna o valor do encoder como resposta. O bloco *Packet Input* recebe como resposta um *bit* de *start*, com o valor “40” (representação decimal), 16 *bits* com o valor do encoder e um *stop bit*, com o valor “41” (representação decimal).

Para o cálculo da posição, os valores do encoder além de serem enviados para o display e para o *Scope*, são também enviados para o bloco *Mod*. Esse bloco recebe também o valor 2048 vindo de um bloco *Constant*, e faz internamente a divisão do valor do encoder por 2048,

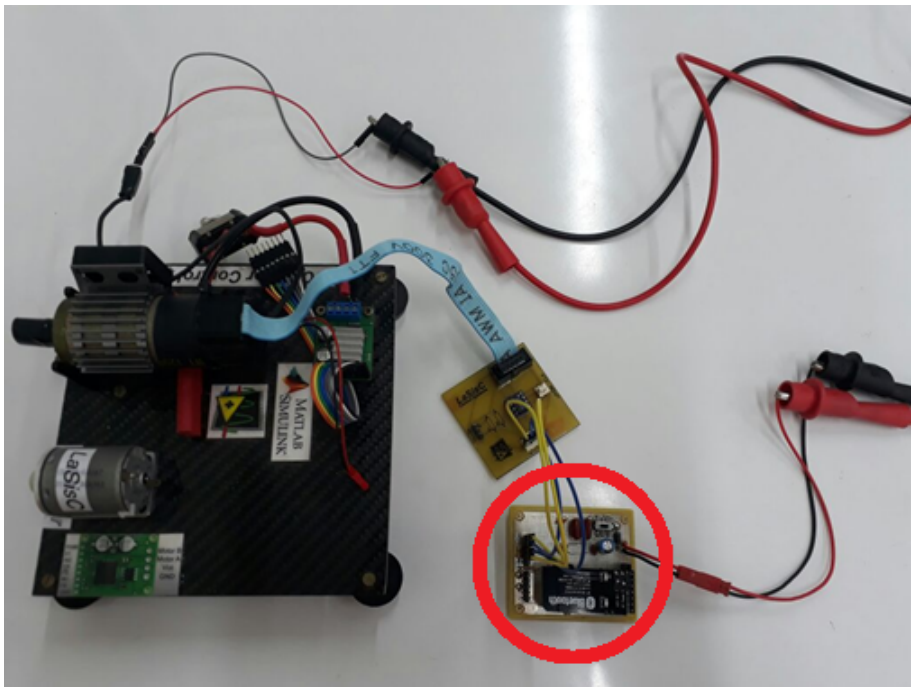
enviando na sua saída o resto dessa divisão. A saída do bloco *Mod* é então enviada como entrada para um bloco *Gain* onde é multiplicada por alguns valores, sendo eles 0,17578125 para grau e (6,28318537/2048) para radiano, que são os valores obtidos para a conversão do valor do encoder para o valor da posição em cada unidade. Na saída de cada bloco *Gain* tem-se um display e um *Scope* para exibir a posição na unidade calculada.

4 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados obtidos com o sistema desenvolvido.

A Figura 15 mostra a parte do sistema desenvolvido que é ligada ao encoder. Na imagem, apenas a PCI circulada foi elaborada neste projeto e deve ser ligada ao encoder, os outros componentes pertencem ao conjunto do servomotor e à fonte de tensão.

Figura 15: **Parte do sistema ligado ao encoder (com uso da fonte de tensão).**



Fonte: Autoria Própria.

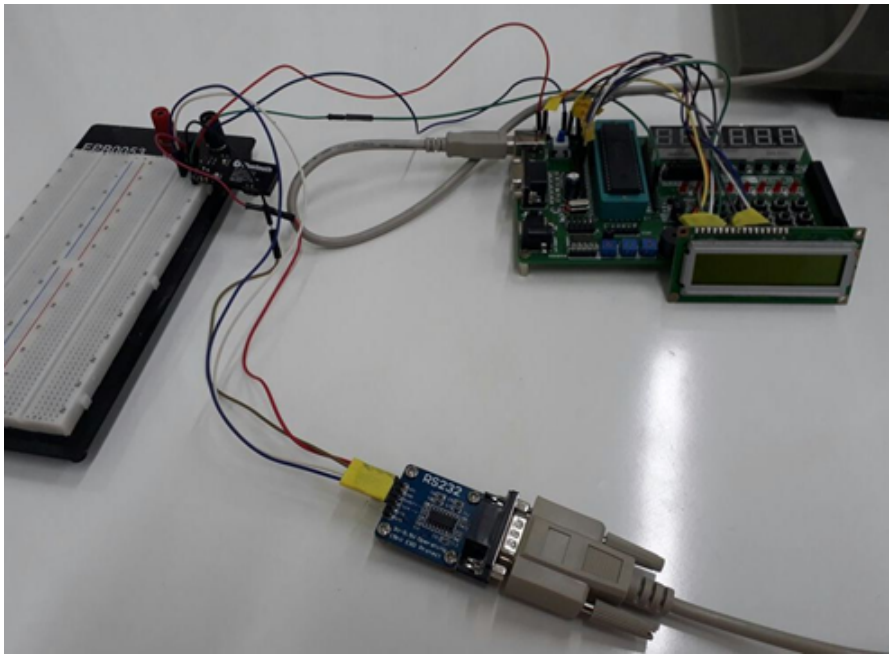
Na Figura 16 é apresentada a parte do sistema que possui tanto comunicação Bluetooth quanto RS232, porém sem a exibição dos dados obtidos no display LCD.

Figura 16: **Parte do sistema ligado ao conversor RS232 board sem o uso do display LCD (com uso da fonte de tensão).**

Fonte: Autoria Própria

Como para o circuito com display LCD não foi feita nenhuma PCI, a Figura 17 mostra esse circuito com protoboard, placa de desenvolvimento e *jumpers*.

Figura 17: **Parte do sistema ligado ao conversor RS232 board sem o uso do display LCD (com uso da fonte de tensão).**



Fonte: Autoria Própria.

4.1 Validação do tempo de amostragem

Para garantir que o tempo escolhido pelo usuário para cada amostra está de acordo com o tempo gasto pelo sistema, foi realizado um teste com o analisador lógico Logic 8 em conjunto com o *software* Logic 1.2.14, ambos da fabricante Saleae.

O pino utilizado pelo analisador lógico para aferição do tempo de amostragem foi o

pino Rx do conversor RS232 *board*. Através desse pino pode-se observar a emissão de sinal do Matlab/Simulink para o PIC, quando o pino Rx recebe o *byte* com "{" (representação em ASCII).

Para esse teste, o analisador lógico foi configurado para capturar a transmissão de dados durante 10 segundos, mas somente algumas partes da amostra foram utilizadas para fazer a validação.

A Tabela 3 exibe os valores obtidos com o analisador lógico, os valores desejados, a porcentagem de erro e o resultado do teste para cada valor.

Tabela 3: **Resposta do teste com o analisador lógico.**

Tempo desejado de amostragem (s)	Tempo real de amostragem (s)	Taxa de erro (%)	Resultado do teste
0,1	0,0996	0,39	Aprovado
0,15	0,1503	0,26	Aprovado
0,156	0,1562	0,16	Aprovado
0,16	0,1601	0,10	Aprovado
0,17	0,1699	0,05	Aprovado
0,25	0,2500	0,00	Aprovado
0,5	0,5000	0,00	Aprovado
0,75	0,7500	0,00	Aprovado
1	1,0000	0,00	Aprovado
1,25	1,2500	0,00	Aprovado
1,5	1,5000	0,00	Aprovado
1,75	1,7500	0,00	Aprovado
2	2,0000	0,00	Aprovado
2,25	2,2500	0,00	Aprovado
2,5	2,5000	0,00	Aprovado
2,75	2,7500	0,00	Aprovado
3	3,0000	0,00	Aprovado
3,5	3,5000	0,00	Aprovado
4	4,0000	0,00	Aprovado
5	5,0000	0,00	Aprovado

Fonte: (MICROCHIP, 2017b).

Através dos resultados do teste é possível então afirmar que o sistema executa a rotina

de amostragem no tempo planejado, validando assim o tempo de amostragem do sistema.

4.2 Posição do encoder

Na interface desenvolvida no Matlab/Simulink foram inclusos alguns blocos para cálculo da posição, tanto em grau quanto em radiano. Uma forma de verificar a posição calculada é manualmente, girando o motor uma volta e conferindo se o valor obtido no Matlab/Simulink é condizente. Essa forma de averiguação também confere o valor do encoder, pois é preciso 2048 pulsos para completar uma volta e 32 voltas para que o encoder zere sua contagem e a recomece. Nesses testes o sistema foi considerado aprovado.

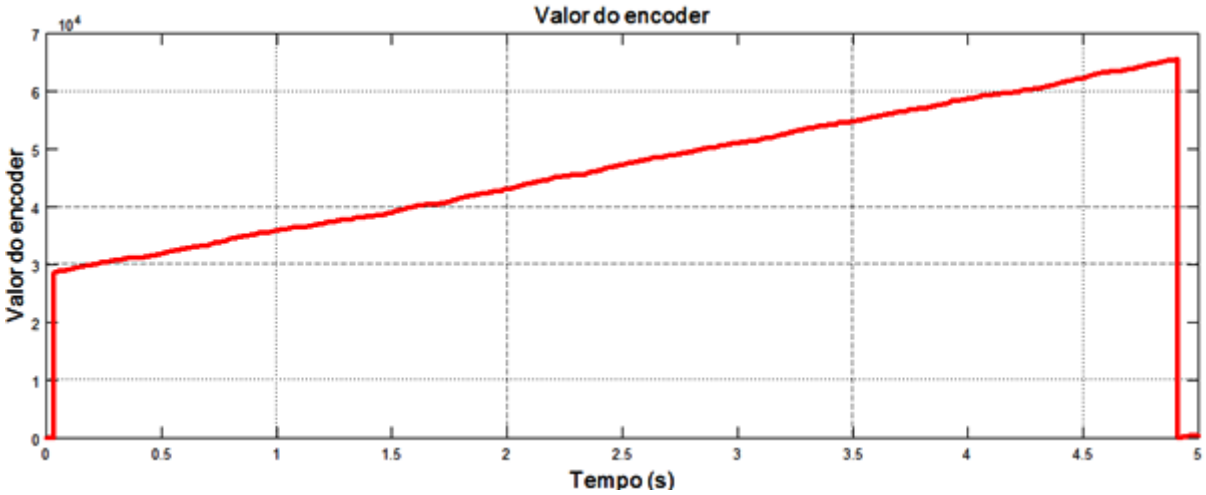
Conforme o aumento de tensão na alimentação do servomotor há um aumento equivalente na sua velocidade de rotação, isso possibilita testar a coleta de dados do encoder em diferentes velocidades. Para realizar esses testes foi utilizada a fonte de tensão de corrente contínua, modelo PS6000 da fabricante Icel.

Para conseguir obter os gráficos tornou-se necessário complementar a interface do Matlab/Simulink com o uso dos blocos *Mux*, que combina os valores obtidos no sistema com o tempo e apresenta apenas um vetor de saída, e *To Workspace*, que armazena os dados obtidos e permite que estes sejam enviados para a área de trabalho do Matlab.

Abaixo são apresentados os gráficos com os valores de pulsos do encoder, obtidos pela interface do Matlab/Simulink em um período de 5 segundos, para diferentes tensões de alimentação no servomotor. Apenas um gráfico com os valores da posição em grau é apresentado e corresponde à tensão de 1,5 V no servomotor, pois acima de 3 V na alimentação os gráficos de posição apresentam um número alto de ciclos e o número de amostras por segundo pode não ser suficiente para permitir que forme o desenho correto dos ciclos. Os valores das tensões utilizadas na alimentação do servomotor para a obtenção dos gráficos e as figuras referentes a cada uma delas são mencionadas a seguir:

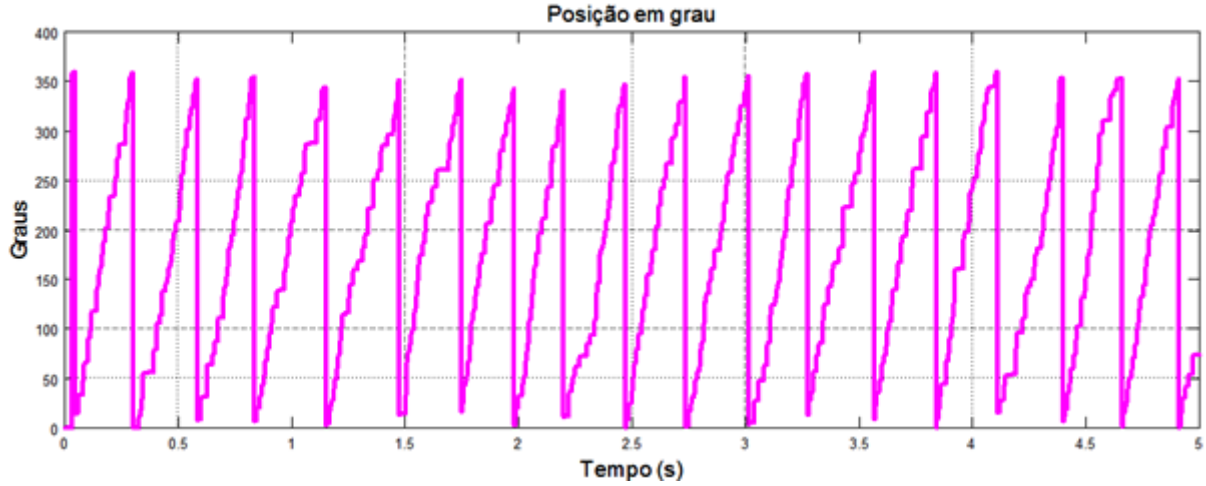
- 1,5 Volt (Figura 18 e Figura 19);
- 4 Volts (Figura 20);
- 7 Volts (Figura 21);
- 10 Volts (Figura 22).

Figura 18: **Valor do encoder para 1,5 V.**



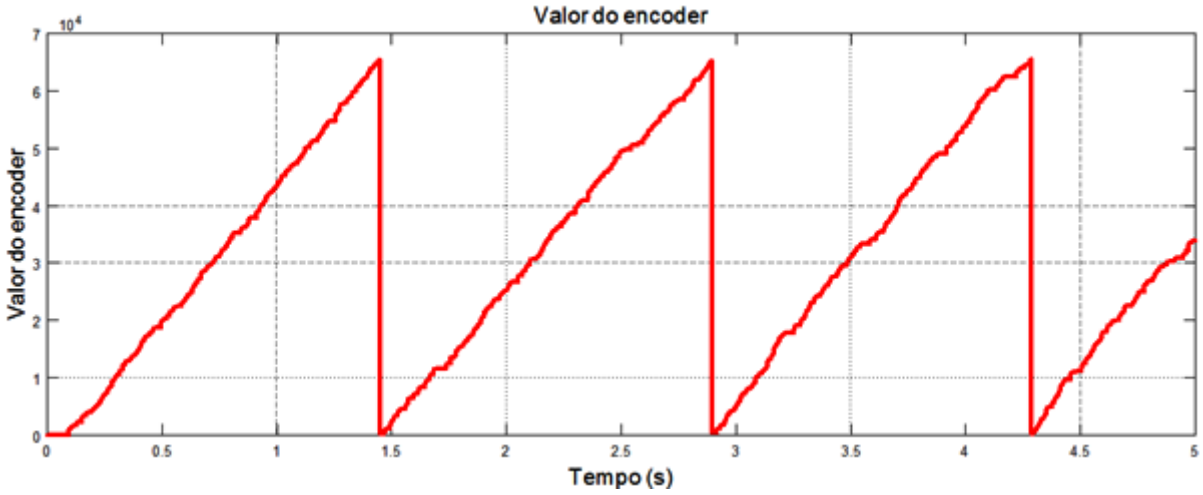
Fonte: Autoria Própria.

Figura 19: **Posição em grau para 1,5 V.**



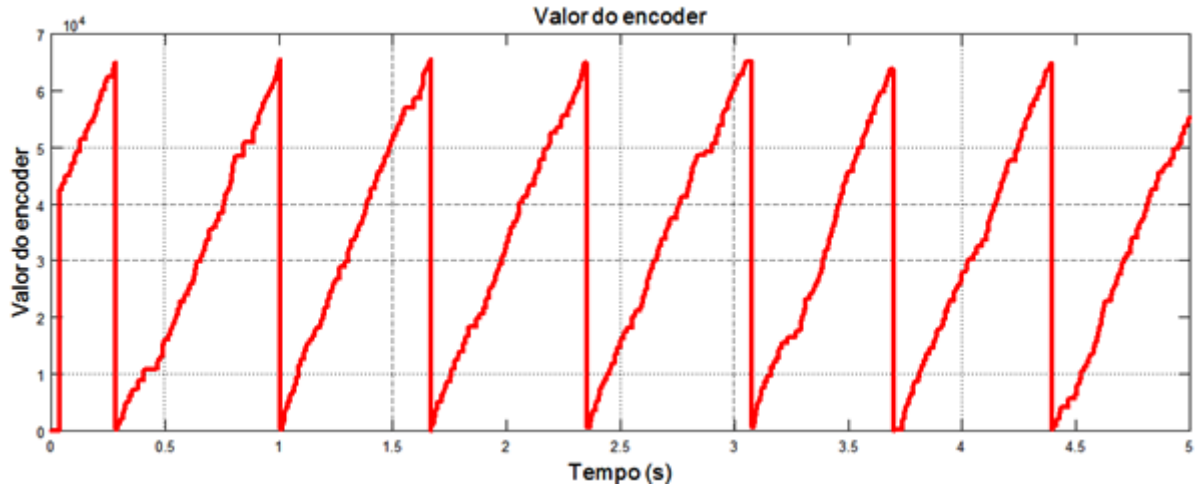
Fonte: Autoria Própria.

Figura 20: **Valor do encoder para 4 V.**



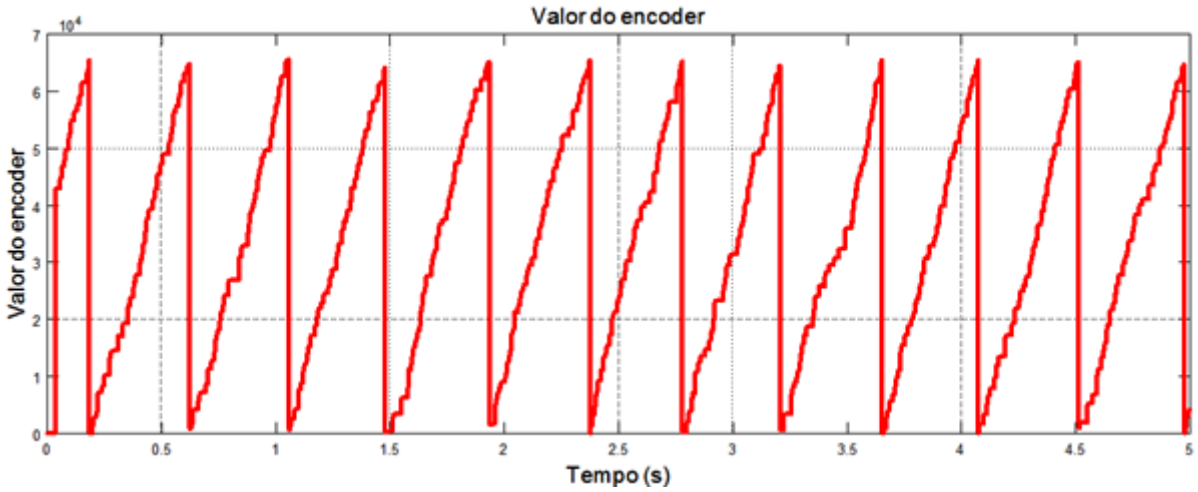
Fonte: Autoria Própria.

Figura 21: **Valor do encoder para 7 V.**



Fonte: Autoria Própria.

Figura 22: **Valor do encoder para 10 V.**



Fonte: Autoria Própria.

Pode-se observar nos gráficos que existem pequenos desvios em alguns valores –visíveis principalmente nos gráficos dos pulsos do encoder –que não permitem o aparecimento de retas perfeitas, isso ocorre devido a uma mínima variação presente na rotação do servomotor. Mesmo ele sendo alimentado com uma fonte de corrente contínua, foi verificado através de medição com o uso de um tacômetro que havia pequenas variações acontecendo na velocidade desse motor.

Fazendo mais algumas análises dos gráficos é possível notar que o valor mínimo e máximo do encoder, 0 e 65535 respectivamente, estão de acordo com o que se é esperado, assim como o gráfico da posição, que varia entre 0 e 360 graus. Observando ainda a posição, e fazendo uma simples regra de três, nota-se que realmente são necessárias 32 voltas para zerar o valor do encoder, validando assim o valor do encoder e a posição encontrada.

5 CONCLUSÃO

Neste trabalho foi desenvolvido um sistema de aquisição de dados de encoder utilizando interface Bluetooth e protocolo RS232. Para isso, foram projetadas e fabricadas duas placas de circuito impresso e elaborada uma interface na plataforma Matlab/Simulink para a aquisição de dados em tempo real.

A validação do sistema foi feita utilizando analisador lógico e análise dos gráfico obtidos.

A aquisição de dados foi feita no tempo planejado e os resultados obtidos são considerados adequados, como pôde ser observado com o analisador lógico e com os gráficos. Além de obter o valor do encoder, foram feitos cálculos e implementações no Simulink para a exibição da posição em graus e em radianos, simultaneamente. Para calcular a velocidade instantânea também foram feitas algumas implementações, como bloco de derivada discreta e de ganho, mas não foi possível alcançar um resultado satisfatório para os cálculos dentro do prazo disponível, pois os valores alcançados continham ruídos exorbitantes que inviabilizaram a medição. Assim sendo, a obtenção da velocidade instantânea não foi finalizada, mas como a velocidade não era, de fato, objetivo deste projeto, e sim realizar a aquisição dos dados e estes foram corretamente coletados, o trabalho não foi prejudicado.

Pode-se dizer então que o trabalho desenvolvido atingiu seu objetivo, pois tinha por objetivo realizar a correta aquisição de dados de encoder através de comunicação serial por Bluetooth e protocolo RS232 para que posteriormente pudesse ser utilizado para verificação de parâmetros ou controle de um sistema, e através dos testes foi verificada a eficiência do trabalho para o objetivo proposto.

5.1 Limitações do trabalho

O trabalho desenvolvido apresenta a necessidade do uso do conversor RS232 board. No entanto, uma PCI com esse conversor já incluso poderia ser desenvolvida.

Além do valor do encoder e da posição poderia também ser encontrada a velocidade.

O circuito responsável pela exibição no display LCD apenas foi executado em uma placa de desenvolvimento unida a uma protoboard. Uma PCI exclusiva para esse circuito poderia ser implementada.

5.2 Trabalhos futuros

É sugerido como trabalhos futuros o desenvolvimento de um sistema de controle atrelado ao sistema de aquisição de dados aqui desenvolvido. Recomenda-se também colocar em prática os itens citados na seção 5.1.

REFERÊNCIAS

BAPTISTA, M. **Sistemas de Instrumentação**. 2017. Disponível em: <http://www.estgv.ipv.pt/paginaspessoais/maeb/im/Teorica_Bibliografia/Cap_E_Sistemas%20de%20Aquisi%C3%A7%C3%A3o%20de%20Dados/1-Introdu%C3%A7%C3%A3o/Texto%20de%20Estudo%20%20Sistemas%20de%20Instrumenta%C3%A7%C3%A3o%20-%20Capitulo%204.pdf>. Acesso em: 19 set. 2017.

BECKER, A. J. et all. Noções Básicas de Programação em Matlab. 2010. Disponível em: <http://www.inf.ufsc.br/~bosco.sobral/ensino/ine5201/Apostila_MATLAB.pdf>. Acesso em: 02 out. 2017.

BONATTO, A.; CANTO, D. O. **Bluetooth Technology (IEEE 802.15)**. 2017. 11 f. Pontifícia Universidade Católica do Rio Grande do Sul. Disponível em: <http://www.inf.pucrs.br/~cnunes/redes/Trabalho%20Bluetooth.pdf>. Acesso em: 28 jun. 2017.

CAMPOS, D.M. G. **Sistema de Comunicação e Controlo de uma espoleta eletrônica**. 2015. 74 f. Dissertação (Mestrado) – Engenharia Eletrotécnica e de Computadores. Instituto Superior Técnico de Lisboa. Lisboa, 2015. Disponível em: <https://fenix.tecnico.ulisboa.pt/downloadFile/1126295043834859/dissertacao.pdf>. Acesso em: 28 jun. 2017.

CARVALHO, V. B. **Desenvolvimento e Teste de um Monitor de Barramento I2C para Proteção Contra Falhas Transientes**. 2016. 87 f. Dissertação (Mestrado) – Ciência da Computação. Universidade Federal do Rio Grande do Sul. Porto Alegre, 2016. Disponível em: <https://www.lume.ufrgs.br/bitstream/handle/10183/150164/001008274.pdf?sequence=1>. Acesso em 28 jun. 2017.

COSTA, M. D. **Central Meteorológica Microcontrolada de Baixo Custo**. 2013. 62 f. Trabalho de Conclusão de Curso (Graduação) – Tecnologia em Automação de Processos Industriais. Universidade Tecnológica Federal do Paraná. Pato Branco, 2013. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/4130/1/PB_COAUT_2012_2_02.pdf>. Acesso em 29 jun. 2017

FUSCO, M. A. V. **Substituição do Sincronismo Mecânico por Sincronismo Eletrônico das Correias e Esteiras de Envernizadeiras e Impressoras Litográficas de Folhas Metálicas**. 2009. 100 f. Dissertação (Mestrado) – Engenharia Mecânica. Universidade de Taubaté. Taubaté, 2009. Disponível em: <http://www.bdtd.unitau.br/tedesimplificado/tde_arquivos/5/TDE-2012-10-08T172649Z-279/Publico/Miguel%20Alexandre%20Vieira%20Fusco.pdf>. Acesso em 31 jul. 2017.

GUIMARÃES, A. A. **Análise da Norma ISO11783 e sua Utilização na Implementação do Barramento do Implemento de um Monitor de Semeadora**. 2003. 114 f. Dissertação (Mestrado) – Engenharia. Escola Politécnica da Universidade de São Paulo. São Paulo, 2003. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3141/tde-13082008-161944/ptbr.php>. Acesso em: 29 jun. 2017.

TALIANI JUNIOR, H. **Estudo dos Protocolos de Comunicação das Arquiteturas Eletroeletrônicas Automotivas, com Foco nas suas Características e Respectivas Aplicações, Visando o Direcionamento para o Uso Adequado e Customizado em Cada Categoria de Veículo**. 2012. 97 f. Monografia (Pós Graduação) – Engenharia de Processos Industriais. Instituto Mauá de Tecnologia. São Caetano do Sul, 2012. Disponível em: <http://maua.br/files/monografias/estudo-dos-protocolos-de-comunicacao-das-arquiteturas-eletroeletronicas.pdf >. Acesso em 27 jun. 2017.

MAIA, R. F. **Interface Tangível do Usuário como Dispositivo de Entrada e Saída**. 2016. 108 f. Dissertação (Mestrado) – Ciência da Computação. Universidade do Estado do Rio Grande do Norte. Mossoró, 2016. Disponível em: <https://ppgcc.ufersa.edu.br/wp-content/uploads/sites/42/2014/09/Interface-Tang%C3%ADvel-do-Usu%C3%A1rio-como-Dispositivo-de-Entrada-e-Sa%C3%ADda.pdf>. Acesso em: 20 jul. 2017.

MATHWORKS. **Simulink**. 2017. Disponível em: <https://www.mathworks.com/products/simulink.html>. Acesso em: 02 set. 2017.

MATHWORKS. **Blocks**. 2017b. Disponível em: <https://www.mathworks.com/help/simulink/blocklist.html?s_cid=doc_ftr>. Acesso em: 02 set. 2017.

MATHWORKS. **Simulink Desktop Real-Time**. 2017c. Disponível em: <https://www.mathworks.com/help/sldrt/>. Acesso em: 02 set. 2017.

MICROCHIP. **PIC18F2331/2431/4331/4431 Datasheet**. 2017. Disponível em: <ww1.microchip.com/downloads/en/devicedoc/39616b.pdf>. Acesso em: 15 ago. 2017.

MICROCHIP. **PICkit 2 Programmer/Debugger User's Guide**. 2017b. Disponível em: < http://ww1.microchip.com/downloads/en/devicedoc/51553e.pdf>. Acesso em: 17 ago. 2017.

MOLLON, M. F. **Planta Didática para Controle PID de um Servomotor com Display TFT e Suporte a Conexão Remota**. 2016.147 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Computação. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

ONOFRE, T. B. **Desenvolvimento de um Protocolo de Comunicação Bidirecional em VHDL para Eletrodos Digitais de EEG**. 2014. 51 f. Dissertação (Mestrado) – Engenharia Elétrica. Universidade Federal de Itajubá. Itajubá, 2014. Disponível em: <https://repositorio.unifei.edu.br/xmlui/bitstream/handle/123456789/404/dissertacao_onofre_2014.pdf?sequence=1&isAllowed=y>. Acesso em: 18 jul. 2017.

PENIDO; É. de C. C.; TRINDADE, R. S. **Microcontroladores**. Ouro Preto: Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais; Santa Maria: Universidade Federal de Santa Maria, Colégio Técnico Industrial de Santa Maria, 2013. 80 f. Disponível em: <http://estudio01.proj.ufsm.br/cadernos/ifmg/tecnico_automacao_industrial/microcontroladores.pdf>. Acesso em: 20 jul. 2017.

PEREIRA, T. M. **SportsMeter:** Sistema de Medição de Desempenho do Movimento Humano em Atividades Desportivas. 2016. 85 f. Dissertação (Mestrado) – Engenharia Biomédica. Faculdade de Engenharia Da Universidade Do Porto. Porto, 2016. Disponível em: <https://repositorioaberto.up.pt/bitstream/10216/87319/2/159269.pdf>. Acesso em 28 jun. 2017.

SILVA, A. J. **Desenvolvimento de um Leitor Digital de Absorbância Microprocessado**. 2011. 55 f. Dissertação (Mestrado) – Ciências. Universidade de São Paulo. Ribeirão Preto, 2011. Disponível em: <http://www.teses.usp.br/teses/disponíveis/59/59135/tde-04052011-003401/pt-br.php>. Acesso em: 28 jun. 2017.

SILVA, D. F. **Sistema de Comunicação Bluetooth Utilizando Microcontrolador**. 2009. 74 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia da Computação. Universidade de Pernambuco. Recife, 2009. Disponível em: <http://tcc.ecomp.poli.br/20092/TCC_final_Davidson.pdf>. Acesso em: 20 de jul. 2017.

SILVA, J. L. e. **Sistemas Embarcados**. 2017. Disponível em: <http://conteudo.icmc.usp.br/pessoas/jsilva/Disciplinas/SE/AulaSE6.PDF>. Acesso em: 21 jul. 2017.

SILVA, M. S. **Automação de Recuperadora de Bauxita para Melhoria da Eficiência Energética**. 2015. 96 f. Dissertação (Mestrado) – Engenharia de Processos. Universidade Federal do Pará. Belém, 2015. Disponível em: <http://ppgep.propesp.ufpa.br/ARQUIVOS/dissertacoes/Dissertacao2015-PPGEP-MP-MarceloSiqueiradaSilva>. Acesso em 31 jul. 2017.

SOUZA, G. S. S. **Arranjo Linear de Dez Eletrodos Ativos Sem Fio Para Eletromiografia de Superfície.** 2013. 229 f. Dissertação (Mestrado) – Engenharia Elétrica. Universidade Federal de Goiás, Goiânia, 2013. Disponível em: <https://repositorio.bc.ufg.br/tede/bitstream/tede/3895/2/Disserta%C3%A7%C3%A3o%20-%20Gustavo%20Souto%20de%20S%C3%A1%20e%20Souza%20-%202013.pdf>. Acesso em: 29 mar. 2017.

TEXAS INSTRUMENTS. **SN74LVC4245A Octal bus transceiver and 3,3-V to 5-V shifter with 3-state outputs**. 2015. Disponível em: <http://www.ti.com/lit/ds/symlink/sn74lvc4245a.pdf>. Acesso em: 07 abr. 2017.

TIMÓTEO, L. **Automação Industrial:** Codificadores ópticos rotativos. 2013. Disponível em: <https://pt.slideshare.net/MarioTimotius/automao-industrial-encoders-pticos-rotaivos>. Acesso em: 29 set. 2017.

WAVESHARE. **RS232 Board**. Disponível em: < https://www.waveshare.com/rs232-board.htm>. Acesso em: 03 out. 2017.

APÊNDICE A - Rotina de Programação para o PIC18F2331

ROTINA DE PROGRAMAÇÃO PARA O PIC18F2331

```
char lido=0;
void interrupt ();

void main() {
ADCON0.ADON = 0;
ANSEL0=0;
trisa.ra2=1;
trisa.ra3=1;
trisa.ra4=1;

QEICON.QEIM2=1;
QEICON.QEIM1=1;
QEICON.QEIM0=0;

rcon.ipen=1;
intcon.gieh= 1;
intcon.giel= 0;

pir1.rcif=0;
pie1.rcie=1;
ipr1.rcip=1;

UART1_init(115200);
delay_ms(100);

CAP2BUFL = 0;
CAP2BUFH = 0;
```

```
while(1){
        }
}
void interrupt(){
pir1.rcif=0;
    if(UART1_Data_Ready()){
       lido=UART1_Read();
       if(lido=='{'){
       UART1_WRITE('(');
       UART1_WRITE(CAP2BUFL);
       UART1_WRITE(CAP2BUFH);
       UART1_WRITE(')');
       }
   }
}
```

APÊNDICE B - Rotina de Programação para o PIC18F4331

ANEXO B – ROTINA DE PROGRAMAÇÃO PARA O PIC18F4331

```
sbit LCD_RS at RA0_bit;
sbit LCD_RW at RA1_bit;
sbit LCD_EN at RA2_bit;
sbit LCD_D4 at RA3_bit;
sbit LCD_D5 at RA4_bit;
sbit LCD_D6 at RA5_bit;
sbit LCD_D7 at RE0_bit;

sbit LCD_RS_Direction at TRISA0_bit;
sbit LCD_RW_Direction at TRISA1_bit;
sbit LCD_EN_Direction at TRISA2_bit;
sbit LCD_D4_Direction at TRISA3_bit;
sbit LCD_D5_Direction at TRISA4_bit;
sbit LCD_D6_Direction at TRISA5_bit;
sbit LCD_D7_Direction at TRISE0_bit;

int lido;
int buffer[2];
int cont =0;
int flag = 0;
char txt[12];
unsigned int encoder =0;
int x =0;
unsigned int angulo = 0;

void interrupt(){
    if(UART1_Data_Ready()){
    rd2_bit = ~rd2_bit;
    lido = UART1_Read();
    if(lido == 40 && flag==0){
```

```
    flag = 1;
         rd1_bit = ~rd1_bit;
       }else if(flag == 1 && cont < 2){
        buffer[cont] = lido;
        cont++;
        }
       if(lido ==41 && cont ==2){
       flag=0;
       cont=0;
       encoder =((buffer[1]<< 8)  | buffer[0]);
       }
    }
PIR1.RCIF = 0;
}
void main() {
  intcon2.rbpu=0;
  trisb=1;
  trisd =0;
  rd0_bit = 0;
  ADCON0.ADON =0;
  ANSEL0 = 0;
  LCD_RW_Direction = 0;
  LCD_RW = 0; //GND

  RCON.IPEN = 1;
  INTCON.GIEH = 1;
  INTCON.GIEL = 0;
  PIR1.RCIF = 0;
  PIE1.RCIE = 1;
  IPR1.RCIP = 1;
  Lcd_Init();
  UART1_init(115200);
```

```
delay_ms(100);
Lcd_Cmd(_LCD_CLEAR);
Lcd_Cmd(_LCD_CURSOR_OFF);

while(1){
        if(x==0){
        LongToStr(encoder, txt);
        if (encoder>=0 && encoder<=9){
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Encoder Value:");
        Lcd_Out(2,10,Ltrim(txt));
        }
        if (encoder>=9 && encoder<=99 ){
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Encoder Value:");
        Lcd_Out(2,9,Ltrim(txt));
        }
        if (encoder>99 && encoder<=999){
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Encoder Value:");
        Lcd_Out(2,8,Ltrim(txt));
        }
        if (encoder>=999 && encoder<=9999 ){
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Encoder Value:");
        Lcd_Out(2,7,Ltrim(txt));
        }
        if(encoder>=9999 ){
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Encoder Value:");
        Lcd_Out(2,6,Ltrim(txt));
        }
        if(portb.rb1==0){
```

```
        x=1;
         }
         delay_ms(10);
}else{
         if(x==1){
        angulo=((encoder%2048)*0.17578125);
        Lcd_Cmd(_LCD_CLEAR);
        LongToStr(angulo, txt);
                          if (angulo>=0 && angulo<=9){
                         Lcd_Cmd(_LCD_CLEAR);
                         Lcd_Out(1,1,"Posicao em grau");
                         Lcd_Out(2,10,Ltrim(txt));
                         }
                         if (angulo>=9 && angulo<=99 ){
                         Lcd_Cmd(_LCD_CLEAR);
                         Lcd_Out(1,1,"Posicao em grau");
                         Lcd_Out(2,9,Ltrim(txt));
                         }
                        if (angulo>99 && angulo<=360){
                        Lcd_Cmd(_LCD_CLEAR);
                        Lcd_Out(1,1,"Posicao em grau");
                        Lcd_Out(2,8,Ltrim(txt));
                        }

            if(portb.rb0==0){
            x=0;
            }
            delay_ms(10);
          }
       }
   }
}
```