



UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA MECÂNICA

THALES FRAGOSO VIEIRA

UNIDADE DE CONTROLE PARA INJEÇÃO ELETRÔNICA COMMON-RAIL
PIEZOELÉTRICO PARA MOTOR MONOCILÍNDRICO

FORTALEZA

2017

THALES FRAGOSO VIEIRA

UNIDADE DE CONTROLE PARA INJEÇÃO ELETRÔNICA COMMON-RAIL
PIEZOELÉTRICO PARA MOTOR MONOCILÍNDRICO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Mecânica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Engenheiro Mecânico.

Orientador: Prof. Dr. André Valente Bueno

FORTALEZA

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária

Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V719u Vieira, Thales Fragoso Vieira.

Unidade de Controle para Injeção Eletrônica Common-Rail Piezoelétrico para Motor Monocilíndrico /
Thales Fragoso Vieira Vieira. – 2017.
58 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia,
Curso de Engenharia Mecânica, Fortaleza, 2017.
Orientação: Prof. Dr. André Valente Bueno.

1. Injeção eletrônica. 2. Common-Rail. 3. Piezoelétrico. 4. Motor de combustão interna monocilíndrico. I.
Título.

CDD 620.1

THALES FRAGOSO VIEIRA


UNIDADE DE CONTROLE PARA INJEÇÃO ELETRÔNICA COMMON-RAIL
PIEZOELÉTRICO PARA MOTOR MONOCILÍNDRICO


Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Mecânica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Engenheiro Mecânico.

Aprovada em: 22 / 12 / 2017 .

BANCA EXAMINADORA


Prof. Dr. André Valente Bueno (Orientador)
Universidade Federal do Ceará (UFCE)


Prof. Dr. Paulo Alexandre Costa Rocha
Universidade Federal do Ceará (UFC)


Prof.^a Dr.^a Maria Alexandra de Sousa Rios
Universidade Federal do Ceará (UFC)

À minha mãe e à minha avó.

AGRADECIMENTOS

A Deus e à minha família, especialmente à pessoa de minha mãe, que desde que eu nasci se esforçou para prover o auxílio necessário para que eu pudesse chegar até aqui.

À minha vó e minha irmã, por todo o amor e cuidado.

A meu amigo Hugo e sua família, por toda a hospitalidade que foi essencial nos meus primeiros anos na graduação.

À Rebeca, pelo seu poder de mudar o meu dia.

Aos meus amigos: Israel, Leandro, Pedro e Victor, por sempre estarem presentes durante esses muitos anos de estudos.

Ao Prof. Dr. André Valente Bueno, por todo ensino e orientação que me forneceu durante minha graduação. E ao senhor Laércio, por toda a sua disposição em me ajudar.

A todos os colaboradores do Laboratório de Motores de Combustão Interna, em particular ao meu amigo Daniel, por permanecer até tarde da noite olhando luzes em um osciloscópio para que esse trabalho pudesse ser possível.

Ao capacitor de $0.1\mu F$ que me devolveu a esperança quando eu já não sabia mais o que fazer.

E ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

“It feels good to have made something. The best thing is that each person’s would be different. In a way, you’ve already won in this world because you’re the only one who can be you. And that’s the way it’s suppose to be.”

(Fred McFeely Rogers)

RESUMO

Este trabalho visa o projeto e construção de uma unidade de controle para um injetor eletrônico capaz de alterar, em tempo real, o ângulo desejado para injeção e a duração da mesma em um motor de combustão interna monocilíndrico. Desenvolvida especificamente para sistemas que fazem uso da tecnologia *Common-Rail* e bicos injetores acionados por princípios piezoelétricos, esta unidade faz uso de microcontroladores em conjunto com transistores IGBTs e outros componentes eletrônicos para processar um sinal obtido através da rotação do motor e produzir um sinal de controle preciso para o bico injetor com início e duração escolhidos pelo usuário durante a operação do motor. Foram realizados testes com o auxílio de um gerador de sinal e um osciloscópio, mostrando-se possível controlar o ângulo de início da injeção e a duração de injeção.

Palavras-chave: Injeção eletrônica. *Common-Rail*. Piezoelétrico. Motor de combustão interna monocilíndrico.

ABSTRACT

This work proposes the design and construction of a control unity for an electronic fuel injector capable of modify, in real time, the desired angle of injection and its duration in a single-cylinder internal combustion engine. Specifically developed for systems which use the Common-Rail technology and fuel injectors powered by piezoelectric principles, this unit makes use of microcontrollers together with IGBTs transistors and others electronic components to process a signal obtained through the rotation of the engine and produce a precise control signal for the fuel injector with begging and duration requested by the user during the engine's operation. Tests with the help of a pulse generator and an oscilloscope were made and it was possible to observe the capability to control the start angle of injection and the injection's duration.

Keywords: Electronic fuel injection. Common-Rail. Piezoelectric. Single-cylinder internal combustion engine.

LISTA DE FIGURAS

Figura 1 – Bosch EFI	14
Figura 2 – Sistema <i>Common-Rail</i>	15
Figura 3 – Injetor piezoelétrico Bosch	16
Figura 4 – <i>Encoder</i> rotacional - Omega Engineering	18
Figura 5 – Microcontrolador PIC18F45K22-I/P	19
Figura 6 – Símbolo para um IGBT	22
Figura 7 – Display LCD 16x2	23
Figura 8 – Ponte H	24
Figura 9 – Sinal de entrada e saída	27
Figura 10 – Resultado obtido com duração de injeção de <i>5ms</i>	28
Figura 11 – Resultado medido na saída para o bico injetor	28
Figura 12 – Mapa ou tabela de injeção	29

LISTA DE ABREVIATURAS E SIGLAS

BJT	<i>Bipolar Junction Transistor</i>
ECU	<i>Electronic Control Unit</i>
EFI	<i>Electronic Fuel Injection</i>
GDI	<i>Gasoline Direct Injection</i>
IGBT	<i>Insulated-Gate Bipolar Transistor</i>
MOSFET	<i>Metal–Oxide–Semiconductor Field-Effect Transistor</i>
UART	<i>Universal Asynchronous Receiver-Transmitter</i>

LISTA DE SÍMBOLOS

V_{DS}	Tensão entre os terminais <i>drain</i> e <i>source</i>
V_{GS}	Tensão entre os terminais <i>gate</i> e <i>source</i>
V_{th}	Tensão limite (<i>threshold</i>) entre os terminais <i>gate</i> e <i>source</i>

SUMÁRIO

1	INTRODUÇÃO	13
2	FUNDAMENTAÇÃO TEÓRICA	14
2.1	Sistema de injeção eletrônica	14
2.2	Sistema <i>Common-Rail</i>	15
2.3	Bico injetor piezoelétrico	16
2.4	<i>Encoder</i> rotacional	17
2.5	Microcontroladores	18
2.6	Transistores	19
2.6.1	<i>Transistor de junção bipolar</i>	19
2.6.2	<i>Transistor de efeito de campo metal-óxido-semicondutor</i>	20
2.6.3	<i>Transistor bipolar de porta isolada</i>	21
3	METODOLOGIA	23
3.1	Interação com o usuário	23
3.2	Controle	24
3.3	Alimentação	25
4	RESULTADOS	26
4.1	Comparação entre sinal de entrada e saída no microcontrolador	26
4.2	Variáveis de controle	26
4.3	Resultado obtido na saída para o injetor	27
5	CONCLUSÃO	30
6	SUGESTÃO DE TRABALHOS FUTUROS	31
	REFERÊNCIAS	32
	APÊNDICES	34
	APÊNDICE A – Diagramas Eletrônicos	34
	APÊNDICE B – Códigos-fontes	37
	APÊNDICE C – Diagrama de blocos para a lógica de injeção	58

1 INTRODUÇÃO

Durante muito tempo, os sistemas de alimentação de combustível para motores de combustão interna se dividiam em dois grupos mais comuns: sistemas com injeção de combustível e sistemas carburados. Os motores Diesel sempre utilizaram um sistema com injeção, pois é intrínseco ao seu projeto; já nos motores a gasolina, o uso de carburadores era predominante devido à sua simplicidade.

Entre os anos de 1960 a 1980, começaram a surgir ao redor do mundo regulamentações que restringiam cada vez mais as emissões produzidas por motores de combustão interna, especialmente naqueles usados em automóveis, como por exemplo, a emenda *Motor Vehicle Air Pollution Control Act* da lei federal norte-americana *Clean Air Act* (ESTADOS UNIDOS DA AMÉRICA, 1965). Fez-se necessária a criação de carburadores consideravelmente mais complexos para cumprir as novas normas, diminuindo cada vez mais sua simplicidade, que era sua principal vantagem. A partir desse momento, a popularidade do uso de motores a gasolina com sistema de injeção aumentou drasticamente, ao passo que nos dias atuais, a produção de motores carburados é quase inexistente (WELSHANS, 2013).

Com o desenvolvimento crescente de produtos eletrônicos, sua produção e aquisição ficaram consideravelmente mais acessíveis. Assim, o uso de sistemas de injeção eletrônicos cresceu em popularidade. A injeção controlada eletronicamente possui vantagens consideráveis sobre aquela controlada por meios mecânicos, apresentando maior flexibilidade e precisão na medição da quantidade ideal de combustível a ser injetado e no momento em que este deve ser injetado. Além disso, o uso de sistemas eletrônicos possibilitou o surgimento de novas classes de funções de controle, como mudanças de estratégias durante situações transientes, aprendizado adaptativo com o uso de táticas para detectar e tentar compensar mudanças no sistema, etc (FAIZ *et al.*, 1996).

Dessa forma, esse trabalho tem como objetivo a construção de uma unidade de controle para um injetor eletrônico piezoelétrico capaz de alterar, em tempo real, o ângulo desejado para injeção e a duração da mesma em um motor de combustão interna monocilíndrico.

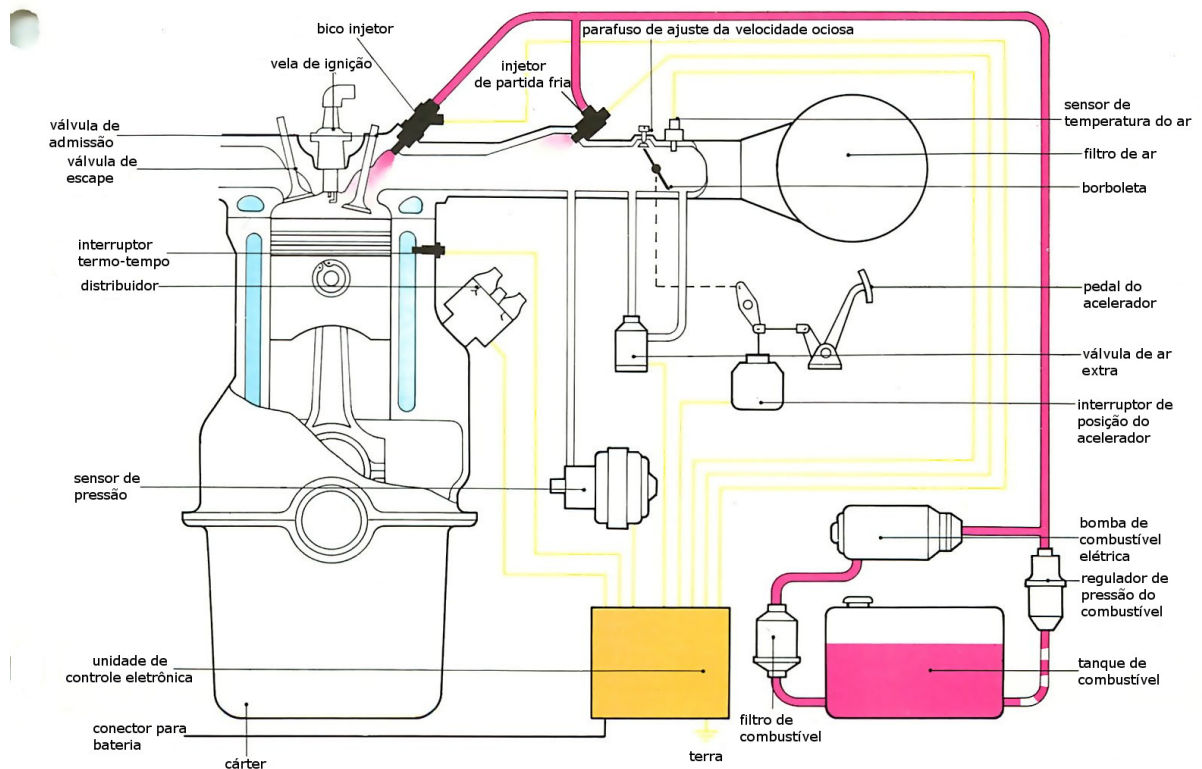
2 FUNDAMENTAÇÃO TEÓRICA

Para uma maior compreensão dos métodos utilizados e resultados obtidos nesse trabalho, faz-se necessário um certo grau de compreensão de alguns conceitos. Este capítulo tem a função de oferecer conhecimentos básicos nas áreas de mecânica e eletrônica que foram usados na confecção deste projeto.

2.1 Sistema de injeção eletrônica

Um sistema de injeção eletrônica de combustível, *Electronic Fuel Injection* (EFI), é controlado por um microcontrolador (ver Seção 2.5) que recebe informações de sensores que medem diversas variáveis importantes para uma operação eficiente do motor. A Figura 1 expõe uma configuração comum de um sistema Bosch de injeção eletrônica para um motor a gasolina.

Figura 1 – Bosch EFI



Fonte: How a Car Works (2011), adaptado.

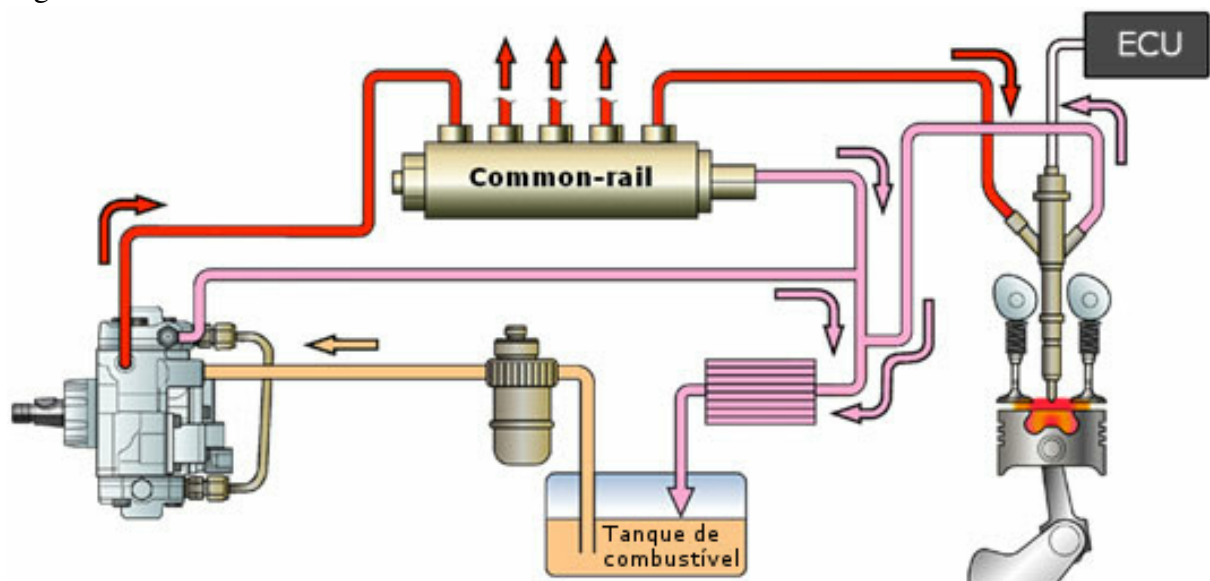
Com as informações obtidas pelos diversos sensores, a unidade de controle eletrônica, *Electronic Control Unit* (ECU), decide o momento em que a injeção deve acontecer e a sua duração.

Vale ressaltar que esse estudo é voltado para a aplicação em um motor Diesel, no qual a injeção ocorre dentro da câmara de combustão e não nos dutos de admissão, como em alguns motores a gasolina, conforme representado na Figura 1. Entretanto, alguns motores a gasolina também fazem uso de injeção diretamente na câmara de combustão, técnica comumente conhecida como injeção direta de gasolina, *Gasoline Direct Injection* (GDI).

2.2 Sistema *Common-Rail*

Trata-se de um sistema com um canal único que mantém o combustível em alta pressão e distribui para todos os injetores quando há demanda. A Figura 2 exemplifica uma configuração comum do sistema *Common-Rail*, onde a cor vermelha representa uma linha de alimentação de alta pressão, a amarela uma de baixa pressão e a rosa representa uma linha de retorno de combustível.

Figura 2 – Sistema *Common-Rail*



Fonte: Thomas Auto Injection Centre Ltd (2017), adaptado.

Nessa configuração, uma bomba de baixa pressão eleva a pressão do combustível contido no tanque para a pressão requerida na entrada da bomba de alta pressão, podendo ser usado um regulador de pressão entre essas duas bombas para manter uma pressão estável e precisa. A bomba de alta pressão possui uma saída de retorno de combustível para ajudar a manter uma pressão constante na entrada do *rail* (canal responsável por distribuir o combustível em alta pressão para os injetores), ademais, alguns *rails* também possuem uma saída para retorno

do combustível. O injetor, com o fornecimento de combustível vindo do *rail*, aguarda o comando da ECU para então injetar o combustível por meio de sistema solenoide ou piezoelétrico.

Devido a sua possibilidade de trabalhar com altas pressões, maiores que 2000 bar (ROBERT BOSCH GMBH, 2014), esse sistema é capaz de alcançar uma melhor atomização do combustível. Múltiplas injeções por ciclo e combustível disponível sob demanda são outras vantagens do *Common-Rail* (KITCHEN, 2013).

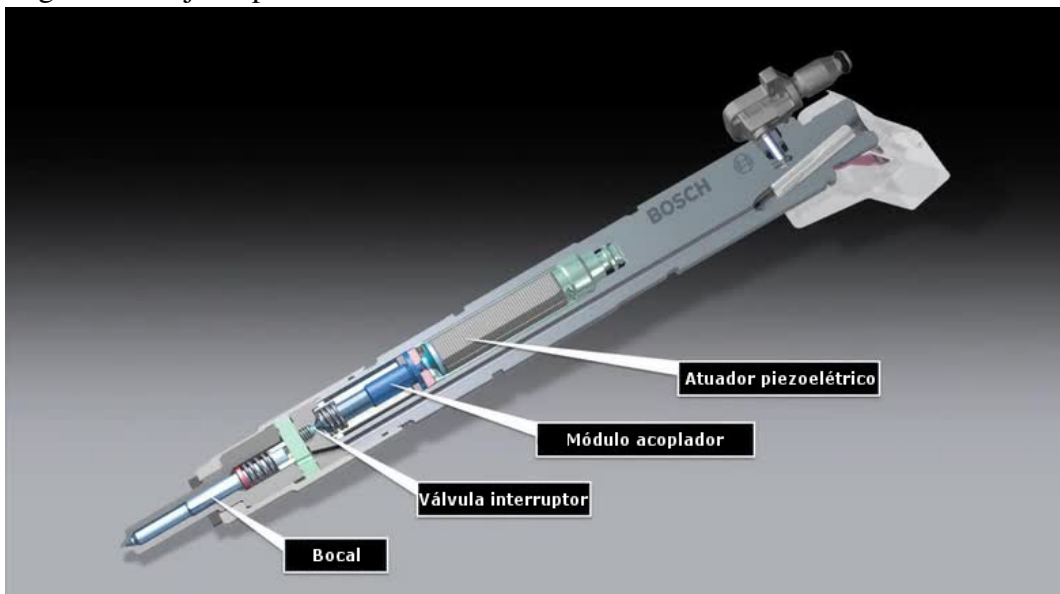
2.3 Bico injetor piezoelétrico

O efeito piezoelétrico se caracteriza pela capacidade de um material de gerar um potencial elétrico quando submetido à uma tensão mecânica. Uma característica importante para essa aplicação é que o efeito piezoelétrico é reversível, ou seja, quando se aplica um potencial elétrico a um material com essa característica o mesmo sofre tensões mecânicas de compressão ou extensão dependendo da polaridade do potencial elétrico aplicado.

Com esse efeito, é possível deformar um material através da aplicação de uma carga elétrica com uma precisão notável (na ordem de nanômetros), além da capacidade de operar em altas frequências (BALBINOT; BRUSAMARELLO, 2007).

A Figura 3 esquematiza um bico injetor piezoelétrico da fabricante Bosch.

Figura 3 – Injetor piezoelétrico Bosch



Fonte: Audi (2016), adaptado.

Injetores piezoelétricos são uma das tecnologias em uso mais avançadas para sistemas *Common-Rail*, chegando a ser quatro vezes mais rápidos que um injetor com atuador solenoide (KITCHEN, 2013) e atingir pressões até 2700 bar (ROBERT BOSCH GMBH, 2017).

2.4 *Encoder* rotacional

Encoder rotacional é um dispositivo usado para converter movimentos rotacionais em sinais elétricos. Em relação ao seu sinal de saída, os *encoders* podem ser divididos em duas categorias: absolutos e incrementais.

Na categoria de resposta absoluta, os dispositivos possuem um sinal de saída que contém a posição absoluta do eixo, ou seja, fornecem um sinal de saída diferente para cada passo de ângulo (dependente da resolução do instrumento).

Encoders incrementais transmitem respostas idênticas para cada passo de resolução, assim eles fornecem uma posição relativa. Geralmente possuem duas saídas de sinais, ao passo que a segunda configura um pulso de zero ou referência. Assim, na maioria das aplicações é necessária uma unidade de processamento para se obter uma posição absoluta a partir da referência do pulso de zero, sendo possível obter tal posição apenas após o *encoder* ter passado pelo ponto zero ao menos uma vez. São muito utilizados devido a sua maior simplicidade e menor custo se comparado a um *encoder* de resposta absoluta.

Quanto ao seu método de funcionamento, os *encoders* rotacionais são divididos em duas categorias básicas, por contato ou sem contato. Os *encoders* por contato são mais simples, mas possuem várias limitações, como velocidade máxima e alto desgaste.

Os *encoders* com método de leitura sem contato podem utilizar vários fenômenos físicos para possibilitar seu funcionamento. *Encoders* ópticos usualmente são os mais preferidos por suas altas resoluções e sua capacidade de operar eficientemente em altas velocidades (DYNAMIC RESEARCH CORPORATION, 1992). *Encoders* magnéticos e capacitivos são outros exemplos de dispositivos com método de leitura sem contato.

Um *encoder* rotacional da Omega Engineering pode ser visto na Figura 4.

Figura 4 – *Encoder* rotacional - Omega Engineering



Fonte: Omega Engineering (2015).

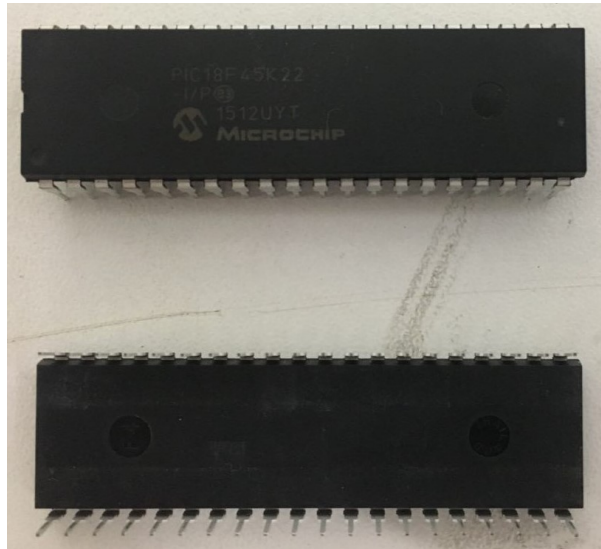
2.5 Microcontroladores

Um microcontrolador pode ser considerado como um computador completo, possuindo um processador, memória e sistemas periféricos em apenas um circuito integrado. Possuem uma memória programável onde é possível realizar a gravação de um software, que será responsável pela execução da função desejada. Constitui um sistema muito versátil, assim, em vários casos, somente a adição de um software é necessária para sua aplicação em um sistema embarcado (HEATH, 2003).

O microcontrolador consegue receber informações externas e processá-las com a ajuda de vários periféricos, como conversores, comparadores, *timers* (contadores de tempo), etc. Os microcontroladores da atualidade já possuem diversos destes periféricos inclusos no circuito integrado e prontos para uso. Depois do processamento, é possível emitir sinais externos através de portas de entrada e saída de dados, *I/O (Input/Output)*; esses sinais podem ser usados para controle, por exemplo. Também é possível utilizar essas saídas de dados para se comunicar com outros dispositivos, como uma tela LCD para exibir informações para um usuário, etc.

A Figura 5 ilustra o microcontrolador PIC18F45K22-I/P da Microchip Technology Inc.

Figura 5 – Microcontrolador PIC18F45K22-I/P



Fonte: O autor.

2.6 Transistores

Os transistores são dispositivos capazes de controlar a passagem de corrente entre dois terminais através de um sinal elétrico aplicado em um terceiro terminal, dependendo de sua construção, os transistores podem ser controlados por corrente ou tensão. O transistor pode ser considerado como uma espécie de "válvula" de sinais elétricos, com a vantagem de ser controlado de forma completamente eletrônica e com altas velocidades de operação (BACON, 1968).

O transistor é capaz de amplificar sinais, isto é, obter um sinal de saída com mais potência do que aquele usado para controlar o dispositivo, de modo que essa potência adicional vem de uma fonte externa. Em outras palavras, o transistor é capaz de controlar a passagem de corrente fornecida por uma fonte externa com um sinal de menor potência (HOROWITZ; HILL, 1989).

2.6.1 Transistor de junção bipolar

O transistor de junção bipolar (*Bipolar Junction Transistor* (BJT)) é capaz de controlar correntes através de uma corrente de controle bem menor e possuem três terminais: base, coletor e emissor. Os BJTs são fabricados em dois tipos: NPN (corrente flui do coletor para emissor) e PNP (corrente flui do emissor para coletor) (TRAYLOR, 2017a). As principais regiões de operação de um BJT do tipo NPN são:

- **Zona de corte:** Ocorre quando a diferença de potencial entre base e emissor

é menor que o valor necessário para superar a queda de tensão na junta base-emissor (geralmente $+0.7V$). Nessa região nenhuma corrente fluirá do coletor para o emissor, o transistor está essencialmente desligado, funcionando como um circuito aberto entre coletor e emissor.

- **Zona ativa direta:** Ocorre quando a diferença de potencial entre base e emissor é positiva e ultrapassa o valor de queda de tensão da junta. Nessa zona o transistor pode ser considerado como um amplificador de corrente quasi-linear, onde a corrente entre coletor e emissor é uma função de corrente menor aplicada na base.
- **Zona de saturação:** Com o incremento da corrente na base, chega-se em um ponto onde o aumento de corrente na base não é mais capaz de causar um aumento na corrente do coletor. Diz-se então que o transistor se encontra saturado. A corrente do coletor não varia mais com a corrente na base, mas depende somente da carga aplicada entre coletor e emissor. Nessa zona o transistor atua como um interruptor fechado (curto circuito), com perdas geralmente negligenciáveis.

Através do uso das zonas de corte e de saturação de um BJT, torna-se possível o uso do mesmo como um interruptor eletrônico com altas frequências de operação (TRAYLOR, 2017b).

2.6.2 Transistor de efeito de campo metal-óxido-semicondutor

O transistor de efeito de campo metal-óxido-semicondutor, *Metal-Oxide-Semiconductor Field-Effect Transistor* (MOSFET), possui seu terminal de controle isolado dos demais por uma camada muito fina de óxido (geralmente dióxido de silício), assim, ao contrário dos transistores BJT, os MOSFETs não são controlados por corrente, e sim por um campo elétrico causado por uma aplicação de tensão entre seu terminal de controle (*gate*) e *source* que controla a passagem de corrente entre os terminais *drain* e *source*. Assim como os BJTs, os MOSFETs podem ser fabricados em dois tipos: N e P. Adicionalmente, os MOSFETs do tipo N podem ser fabricados em dois modos: *enhancement mode* ou *depletion mode*, já os do tipo P geralmente só são encontrados em *enhancement mode*.

Para um MOSFET do tipo N em *enhancement mode* é necessária uma diferença de potencial positiva entre *gate* e *source* para que ocorra condução. Por sua vez, em *depletion*

mode, já existe um canal de condução mesmo sem uma diferença de potencial positiva, sendo necessário que se aplique um potencial negativo de alguns volts para parar a condução entre *drain* e *source* (HOROWITZ; HILL, 1989). Tendo em vista o objetivo deste estudo, bem como para fins de simplificação, somente o MOSFET do tipo N em *enhancement mode* será estudado a partir de agora.

A maior vantagem desse tipo de transistor está no fato de que a corrente no *gate* é quase inexistente, tendo-se um dispositivo com uma alta impedância de entrada, o que facilita o seu controle, além de ser uma característica essencial em muitas aplicações (HOROWITZ; HILL, 1989). As regiões de operação se encontram listadas abaixo:

- **Zona de Corte:** Quando a diferença de potencial entre *gate* e *source* é menor que uma tensão V_{th} limite o dispositivo funciona como um interruptor aberto, podendo ocorrer pequenas perdas. Essa tensão limite, V_{th} , varia com o MOSFET utilizado.
- **Zona Linear ou região de Triodo:** Ocorre quando a tensão V_{GS} , entre *gate* e *source*, supera V_{th} , mas a tensão V_{DS} (diferença de potencial entre *drain* e *source*) é menor que a diferença entre V_{GS} e V_{th} . Nessa região, o transistor tem operação similar a um resistor, isto é, a corrente entre *drain* e *source* é aproximadamente proporcional à V_{DS} .
- **Zona de Saturação ou Ativa:** Quando $V_{GS} > V_{th}$ e $V_{DS} \geq V_{GS} - V_{th}$, diz-se que o MOSFET se encontra saturado ou na zona ativa (vale ressaltar a diferença na nomenclatura saturação entre um MOSFET e um BJT). Nessa região, a corrente entre *drain* e *source* tem uma dependência de V_{DS} geralmente negligenciável, sendo agora proporcional à $(V_{GS} - V_{th})^2$ (HOROWITZ; HILL, 1989).

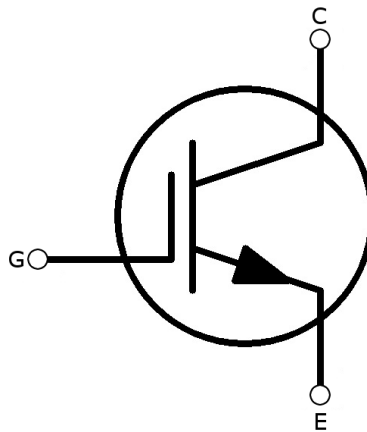
2.6.3 Transistor bipolar de porta isolada

O transistor bipolar de porta isolada (*Insulated-Gate Bipolar Transistor* (IGBT)) é uma combinação entre um BJT e um MOSFET. O IGBT combina a alta impedância de entrada de um MOSFET com a características de saída de um BJT. Dessa forma, o IGBT é um dispositivo de fácil controle, sem virtualmente nenhuma corrente de controle, além de possuir perdas quando ligado consideravelmente menores que um MOSFET equivalente para aplicações de alta tensão e corrente. Por causa disso, os IGBTs são mais recomendados no controle de altas tensões e

correntes (Eletronics Tutorials (2014)).

A Figura 6 ilustra o símbolo usado para representar um IGBT em esquemas elétricos. A partir de um potencial aplicado entre *gate* (g) e emissor (e) é possível controlar a passagem de corrente entre os terminais coletor (c) e emissor.

Figura 6 – Símbolo para um IGBT



Fonte: RS Components (2014), adaptado.

Este estudo fez uso de IGBTs em uma configuração do tipo "ponte H" para auxiliar na abertura e fechamento do bico injetor piezoelétrico.

3 METODOLOGIA

O circuito da unidade de controle para injeção desenvolvida neste estudo foi dividido em três partes: circuito de interação com o usuário, circuito de controle e circuito de alimentação.

3.1 Interação com o usuário

A interface usuário-máquina desse controlador é composta por um display LCD de 16 caracteres por linha e duas linhas, botões e um microcontrolador Microchip PIC18F45K22.

Essa interface permite ao usuário escolher duas variáveis na injeção: ângulo da manivela onde ocorrerá o início da injeção e a duração da injeção. O ângulo pode ser escolhido dentro de duas faixas (limites inclusos): de $-45,72^\circ$ a $46,08^\circ$ ou $54,28^\circ$ a $146,08^\circ$, com um passo (resolução) de $0,36^\circ$, pois o encoder utilizado com essa unidade possui uma resolução de mil pulsos por rotação, isto é, $0,36^\circ$ por pulso. Já a duração da injeção pode ser selecionada em uma faixa de 0 a 25,5ms, com uma resolução de 0,5ms.

O microcontrolador recebe as informações do usuário por meio dos botões, e transmite as variáveis de controle para a visualização na tela LCD e também para outro microcontrolador localizado no circuito de controle. Um dispositivo do tipo Transmissor/Receptor Universal Assíncrono, *Universal Asynchronous Receiver-Transmitter* (UART), foi utilizado para a comunicação entre microcontroladores.

A Figura 7 ilustra a tela LCD utilizada para a visualização na interface usuário-máquina. Foi utilizada uma biblioteca cedida pela Microchip Technology Inc. no código fonte do software do microcontrolador para a comunicação com a tela LCD.

Figura 7 – Display LCD 16x2



Fonte: O autor.

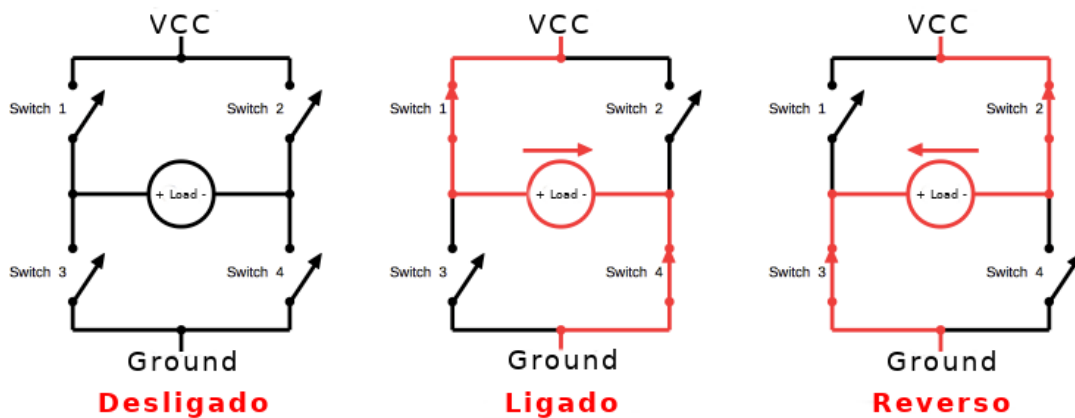
O diagrama do circuito utilizado pode ser encontrado no Apêndice A e o código fonte do software utilizado no microcontrolador desse circuito se encontra no Apêndice B.

3.2 Controle

Um segundo microcontrolador Microchip PIC18F45K22, quatro optoacopladores e quatro IGBTs são os componentes principais dessa porção da unidade de controle. O diagrama eletrônico completo pode ser visualizado no Apêndice A.

O microcontrolador recebe informações do ângulo atual da manivela do *encoder* e também recebe informações do outro microcontrolador sobre o ângulo desejado de injeção e a duração da mesma. Quando a posição atual da manivela condiz com a desejada, o microcontrolador manda sinais para os optoacopladores para operar os transistores que se encontram em uma configuração de ponte H, ilustrada na Figura 8.

Figura 8 – Ponte H



Fonte: Shirriff (2016), adaptado.

Nessa configuração, pode-se aplicar uma diferença de potencial tanto positiva quanto negativa nos polos de controle do bico injetor através do acionamento de dois transistores de diagonais opostas. Dessa forma, é possível abrir e fechar o bico com indubitabilidade.

O software programado no microcontrolador proporciona um atraso de $1,4\text{ms}$ entre a desativação de um par e ativação de outro, para garantir, com uma boa margem de segurança, tempo suficiente para a resposta dos optoacopladores e IGBTs, de acordo com seus respectivos *datasheets*. Essa medida foi feita para garantir que em nenhum momento a alimentação (VCC) seja ligada diretamente ao terra (*ground*), o que acarretaria em um curto-circuito, o qual poderia causar danos à unidade. O código fonte do software utilizado pode ser visualizado no Apêndice B e um diagrama de blocos representando sua lógica simplificada está ilustrado no Apêndice C.

3.3 Alimentação

O circuito de alimentação pode ser dividido em duas partes: alta e baixa tensão. Seu diagrama pode ser encontrado no Apêndice A.

A porção de alta tensão é usada na alimentação do bico injetor que será operado a aproximadamente 160V em corrente contínua. É composta por um transformador, um retificador de onda completa com diodos em ponte e um capacitor para amenizar variações na tensão.

O segmento de baixa tensão comporta quatro fontes separadas, uma de 5V e três de 19V, todas de corrente contínua. A fonte de mais baixa tensão será utilizada para a alimentação dos microcontroladores, e por isso, necessita de uma maior precisão. Com esse intuito, foi utilizado o regulador linear de tensão *LM7805*, alimentado por um retificador com derivação central (*center-tapped*) a 12V, e um conjunto de capacitores de diferentes faixas para amenizar ruídos.

As fontes de 19V, usadas para alimentar os optoacopladores, utilizam retificadores com derivação central a 15V e capacitores para mitigar variações. Foram necessárias três fontes independentes, pois os transistores nas posições superiores na ponte H estão com seus terminais emissores em potenciais diferentes entre si e diferentes dos outros dois transistores contidos na configuração.

4 RESULTADOS

A unidade de controle desenvolvida nesse estudo possui a finalidade de operar um motor de combustão interna, cujo sistema de injeção original dá-se por meios mecânicos, sendo ainda necessária uma adaptação em seu cabeçote para possibilitar a substituição do seu sistema de injeção para um com a metodologia *Common-Rail* e operado eletronicamente. Sendo assim, os testes realizados utilizaram um gerador de sinais para reproduzir os pulsos gerados por um *encoder* e um osciloscópio para examinar os sinais de saída.

Então, através da interface usuário-máquina, foi escolhido o ângulo zero (ponto morto superior do pistão) como a posição desejada para injeção. Essa escolha foi feita para simplificar a execução dos testes, pois, assim, somente é necessário um sinal de entrada de zero para a unidade produzir sinais para abrir o injetor. Dessa forma, elimina-se a necessidade de dois geradores de pulsos.

4.1 Comparação entre sinal de entrada e saída no microcontrolador

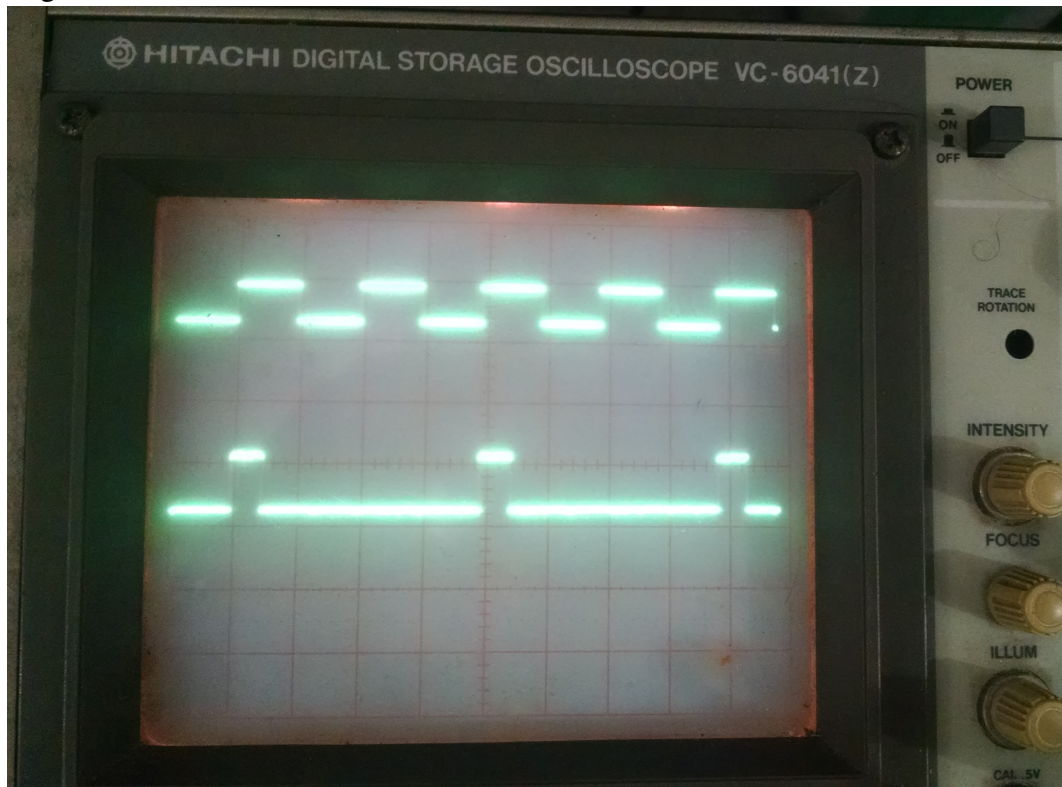
Com o uso do osciloscópio foi possível obter uma comparação lado a lado entre os sinais de entrada e saída no microcontrolador, uma vez que estes possuem a mesma referência negativa. O resultado obtido foi o desejado, com o microcontrolador emitindo pulsos para os optoacopladores a cada pulso de entrada alternado, pois essa unidade foi projetada para um motor de quatro tempos, que realizam duas rotações por ciclo, uma para compressão e outra para exaustão. Assim, o bico deve injetar combustível em rotações alternadas.

Nesse teste, foi utilizado um gerador de pulsos a uma frequência de $10Hz$ e *duty-cycle* de 50%, de modo que cada pulso alto possui uma duração de $50ms$. O tempo escolhido para a duração da injeção foi de $25ms$, assim, por inferência gráfica foi possível validar o funcionamento do controle no quesito tempo de injeção. A Figura 9 expõe o resultado obtido, com o sinal de entrada na posição superior e o sinal de saída logo abaixo.

4.2 Variáveis de controle

Durante os testes foram feitas mudanças em tempo real nas variáveis ângulo e duração de injeção. Como esperado, quando o ângulo de injeção é alterado para qualquer valor diferente de zero, os sinais de saída cessam imediatamente, pois somente o pulso de

Figura 9 – Sinal de entrada e saída



Fonte: O autor.

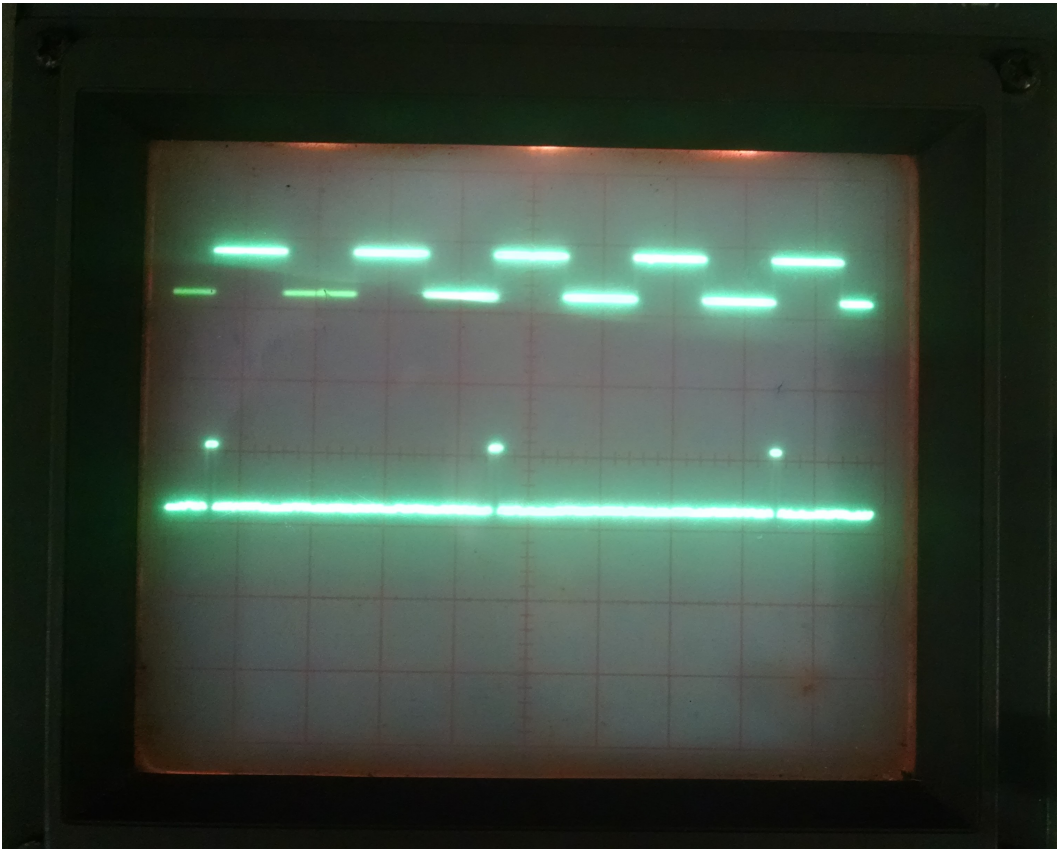
referência (zero) está sendo fornecido ao microcontrolador. A duração de injeção também foi variada durante os testes. A Figura 10 expõe o resultado obtido quando uma duração de $5ms$ foi escolhida, podendo ser comparada a Figura 9 para verificação de coerência.

4.3 Resultado obtido na saída para o injetor

A Figura 11 exhibe o resultado obtido no osciloscópio ao analisar o sinal de saída da unidade de controle para o injetor. A passagem de tempo se apresenta da esquerda para direita, podendo-se notar o pulso positivo onde ocorre a injeção e sua duração, também sendo possível ver o subsequente pulso negativo, usado para garantir o fechamento do bico, e o atraso programado entre eles. O pulso negativo possui duração fixa e programada de $1,4ms$, o mesmo tempo de atraso mínimo programado entre pulsos. Esse pulso tem como única finalidade garantir que o bico injetor volte ao seu estado fechado, em razão disso, sua duração pode ser bem breve.

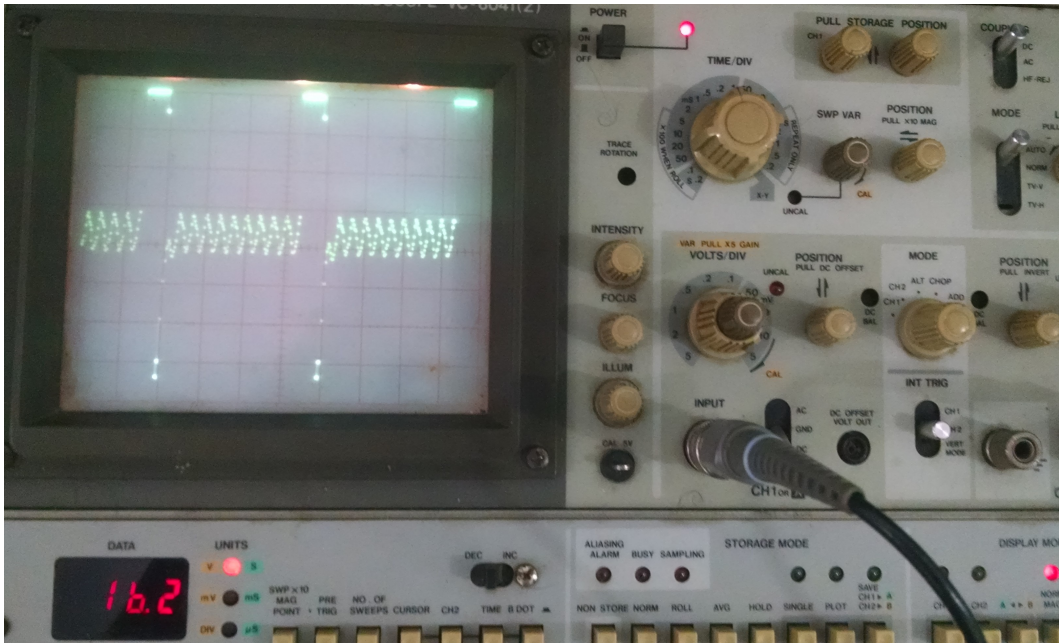
As ondulações vistas no osciloscópio entre as injeções se dão pelo fato de que, quando todos os transistores se encontram fechados, os pontos onde se faz a medição se encontram em potenciais flutuantes diferentes. Assim, essas ondulações não representam variações reais de

Figura 10 – Resultado obtido com duração de injeção de 5ms



Fonte: O autor.

Figura 11 – Resultado medido na saída para o bico injetor



Fonte: O autor.

potencial elétrico, tomando como referência o negativo da fonte alimentadora do bico injetor. Na Figura 11 também é possível verificar a diferença de potencial dos pulsos positivo e negativo, em relação ao referencial da fonte, através do visor localizado no canto inferior esquerdo do

osciloscópio. Essa medição foi feita com a ponta de prova com configuração *X10*, então o valor visto no visor equivale a 162V.

O controle de variáveis pode ser utilizado para validar resultados obtidos com softwares de simulação, criando assim a possibilidade de produção de estudos na área da otimização do uso de combustível, como por exemplo, a criação de mapas de injeção (*Fuel Maps*), que visam relacionar a quantidade de combustível injetada com as condições de operação do motor para obter um objetivo específico, como economia ou potência. A Figura 12 expõe um exemplo de um mapa de injeção, onde é possível ver a razão entre massa de ar e massa de combustível (*AF*) relacionada com a carga no eixo vertical e a velocidade de rotação do motor no eixo horizontal.

Figura 12 – Mapa ou tabela de injeção

AF desejado, "Mapa de Combustível", para loop de controle aberto (1995 3000GT Spyder VR4)															
RPM	500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500	7000	7500
C a r g a	BAIXA	14.0	14.7	19.8	19.8	19.8	19.8	18.8	18.1	18.1	18.1	18.1	18.1	18.1	18.1
		14.0	14.7	14.7	16.4	16.4	16.4	16.5	16.8	16.8	16.8	16.8	16.8	16.8	16.8
		14.0	14.7	14.7	14.7	14.7	14.7	14.7	15.7	15.7	15.3	14.9	14.9	14.9	14.9
		14.0	14.7	14.7	14.7	14.7	14.7	14.7	14.7	14.7	13.9	13.3	13.3	13.3	13.3
		14.7	14.7	14.7	14.7	14.7	14.7	14.7	14.7	14.7	14.5	12.9	12.9	12.9	12.9
		14.7	14.7	14.7	14.7	14.7	14.7	14.7	14.7	14.3	13.3	12.6	12.1	11.8	11.8
		14.7	14.7	14.7	14.7	14.7	14.7	14.7	14.7	13.6	12.9	12.2	11.8	11.3	11.3
		13.6	13.6	14.7	14.7	14.7	14.7	14.7	14.7	13.3	12.5	11.9	11.4	10.9	10.9
		13.4	13.4	13.8	14.3	14.3	14.7	14.7	13.1	13.1	12.2	11.5	11.1	10.7	10.7
		13.4	13.4	13.4	13.4	13.4	13.6	13.6	12.1	12.1	11.6	11.2	10.8	10.5	10.5
		13.4	13.4	13.4	13.4	13.1	13.1	13.1	11.8	11.8	11.2	10.7	10.5	10.3	10.3
		13.4	13.4	13.4	13.4	12.9	12.9	12.5	11.6	11.3	10.5	10.4	10.3	10.2	10.2
	ALTA	13.4	13.4	13.4	13.4	12.9	12.9	12.5	11.6	11.3	10.5	10.4	10.3	10.2	10.2

Fonte: Formula 1 Dictionary (2014), adaptado.

5 CONCLUSÃO

Tendo em vista os resultados obtidos, constata-se que os objetivos iniciais foram parcialmente alcançados com a construção de uma unidade de controle que atende os requisitos específicos deste trabalho. Os resultados obtidos por meio do uso de osciloscópio demonstram a obtenção de uma máquina capaz de controlar um injetor eletrônico piezoelétrico, entretanto, testes com o uso do injetor ainda são necessários.

Além disso, a unidade de controle construída é capaz de modificar o ângulo de início de injeção e a duração da mesma, sendo essas variáveis escolhidas pelo o usuário durante a operação do injetor.

6 SUGESTÃO DE TRABALHOS FUTUROS

A real utilidade da unidade desenvolvida nesse trabalho encontra-se na possibilidade de usá-la em trabalhos posteriores de pesquisa ou ensino. Esta unidade de controle torna possível o estudo da relação que o ângulo de início de injeção e a duração da mesma tem com a eficiência e a potência na operação de um motor de combustão interna. A aplicação da unidade de controle aqui desenvolvida é voltada especificamente para motores monocilíndricos, que possuem uma maior simplicidade e facilidade na obtenção da relação entre duas variáveis isoladas, facilitando, dessa forma, a obtenção de objetivos de pesquisa.

REFERÊNCIAS

- AUDI, A. **Piezo injector**. 2016. Disponível em: <<https://www.audi-technology-portal.de/en/drivetrain/tdi-engines/piezo-injectors>>. Acesso em: 04 out. 2017.
- BACON, W. S. The transistor's 20th anniversary: How germanium and a bit of wire changed the world. Bonnier Corp.: Popular Science, 1968.
- BALBINOT, A.; BRUSAMARELLO, V. J. **Instrumentação e Fundamentos de Medidas**. [S.l.]: LTC, 2007. v. 2.
- DYNAMIC RESEARCH CORPORATION. **Techniques for Digitizing Rotary and Linear Motion**. [S.l.]: Drc, 1992.
- ELETRONICS TUTORIALS. **Insulated Gate Bipolar Transistor**. 2014. Disponível em: <<http://www.electronics-tutorials.ws/power/insulated-gate-bipolar-transistor.html>>. Acesso em: 03 nov. 2017.
- ESTADOS UNIDOS DA AMÉRICA. **Motor Vehicle Air Pollution Control Act (Pub.L. 89-272)**. 1965. Disponível em: <<https://www.gpo.gov/fdsys/pkg/STATUTE-79/pdf/STATUTE-79-Pg992-2.pdf>>. Acesso em: 30 dez. 2017.
- FAIZ, A.; WEAVER, C. S.; WALSH, M. P.; GAUTAM, S. P. **Air pollution from motor vehicles; standards and technologies for controlling emissions**. [S.l.]: The World Bank, 1996.
- FORMULA 1 DICTIONARY. **Fuel Map or Fuel Table**. 2014. Disponível em: <http://www.formula1-dictionary.net/map_fuel.html>. Acesso em: 19 dez. 2017.
- HEATH, S. **Embedded Systems Design**. [S.l.]: Newnes, 2003. v. 2.
- HOROWITZ, P.; HILL, W. **The Art Of Electronics**. [S.l.]: Cambridge University Press, 1989. v. 2.
- HOW A CAR WORKS. **How a fuel injection system works**. 2011. Disponível em: <<https://www.howacarworks.com/basics/how-a-fuel-injection-system-works>>. Acesso em: 28 set. 2017.
- KITCHEN, T. **Technical Overview of Common Rail Diesel Fuel Systems**. 2013. AK Automotive Training. Disponível em: <<http://www.yildiz.edu.tr/~sandalci/dersnotu/AKTraining.pdf>>. Acesso em: 02 out. 2017.
- OMEGA ENGINEERING. **Flange Mount Rotary Pulse Generator**. 2015. Disponível em: <https://br.omega.com/pptst/ZDH_SERIES.html>. Acesso em: 21 nov. 2017.
- ROBERT BOSCH GMBH. **Common Rail Systems CRSN3 with 2,000 to 2,500 bar**. 2014. Diesel Systems. Disponível em: <http://products.bosch-mobility-solutions.com/media/ubk_europe/db_application/downloads/pdf/antrieb/en_3/DS-Sheet_P1AS_CRSN3-25_EN_low.pdf>. Acesso em: 03 out. 2017.
- ROBERT BOSCH GMBH. **Common-rail System with piezo injectors**. 2017. Disponível em: <[http://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/powertrain-systems/common-rail-system-\(piezo\)](http://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/powertrain-systems/common-rail-system-(piezo))>. Acesso em: 04 out. 2017.

RS COMPONENTS. **Semiconductor Buying Guide**. 2014. Disponível em: <<http://au.rs-online.com/web/generalDisplay.html?id=infozone&file=electronics/semiconductor-buying-guide#top>>. Acesso em: 23 out. 2017.

SHIRRIFF, K. **Sonicare toothbrush teardown: microcontroller, H bridge, and inductive charging**. 2016. Adaptado. Disponível em: <<http://www.righto.com/2016/09/sonicare-toothbrush-teardown.html>>. Acesso em: 27 nov. 2017.

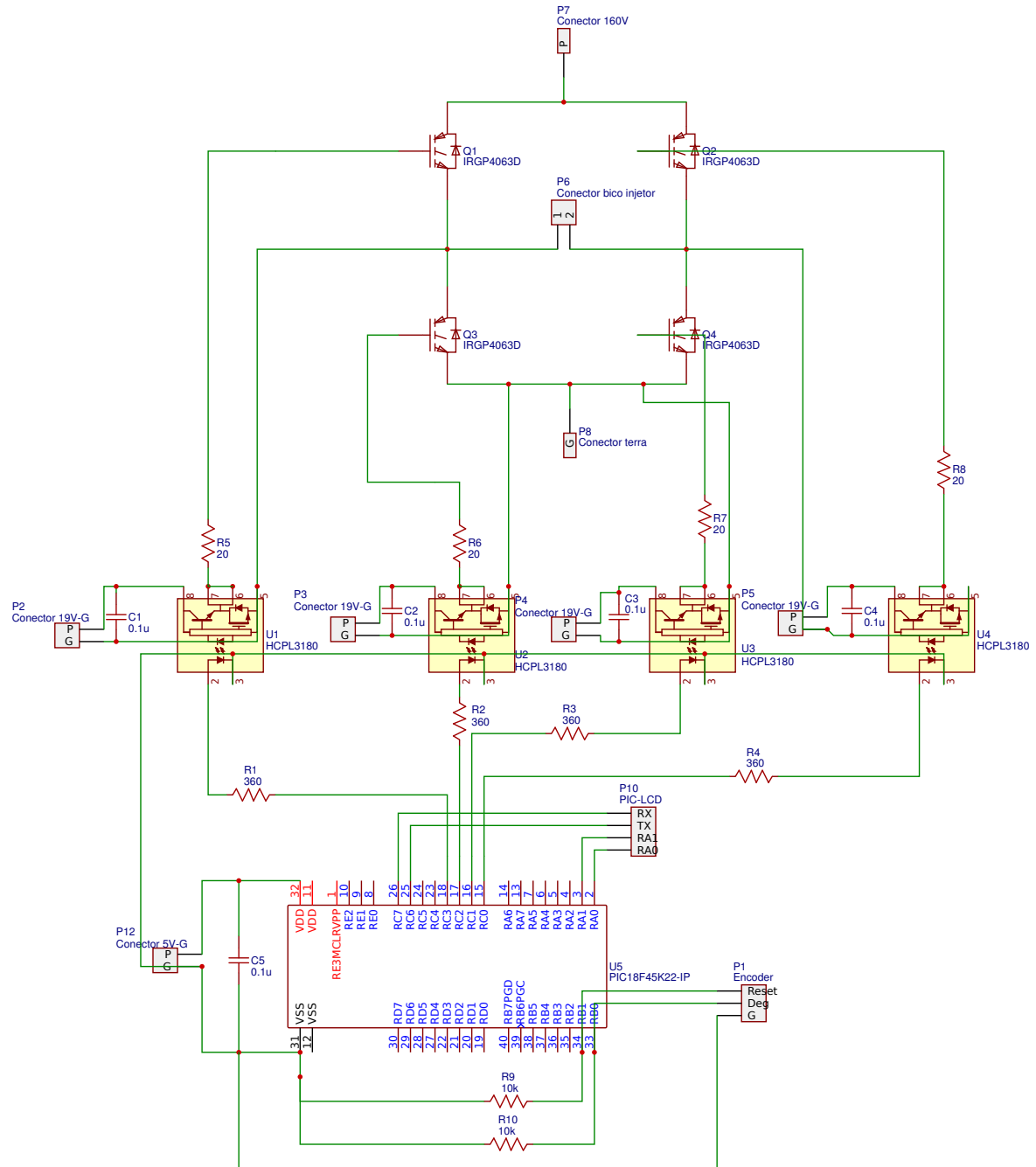
THOMAS AUTO INJECTION CENTRE LTD. **Common Rail System**. 2017. Disponível em: <<http://www.thomasautos.co.uk/Commonrailssystem.html>>. Acesso em: 03 out. 2017.

TRAYLOR, R. L. **Bipolar Junction Transistors (BJTs) - Structure**. 2017. Disponível em: <http://web.engr.oregonstate.edu/~traylor/ece112/beamer_lectures/bjt_structure.pdf>. Acesso em: 27 out. 2017.

TRAYLOR, R. L. **BJT Regions of Operation**. 2017. Disponível em: <http://web.engr.oregonstate.edu/~traylor/ece112/beamer_lectures/bjt_reg_of_op.pdf>. Acesso em: 27 out. 2017.

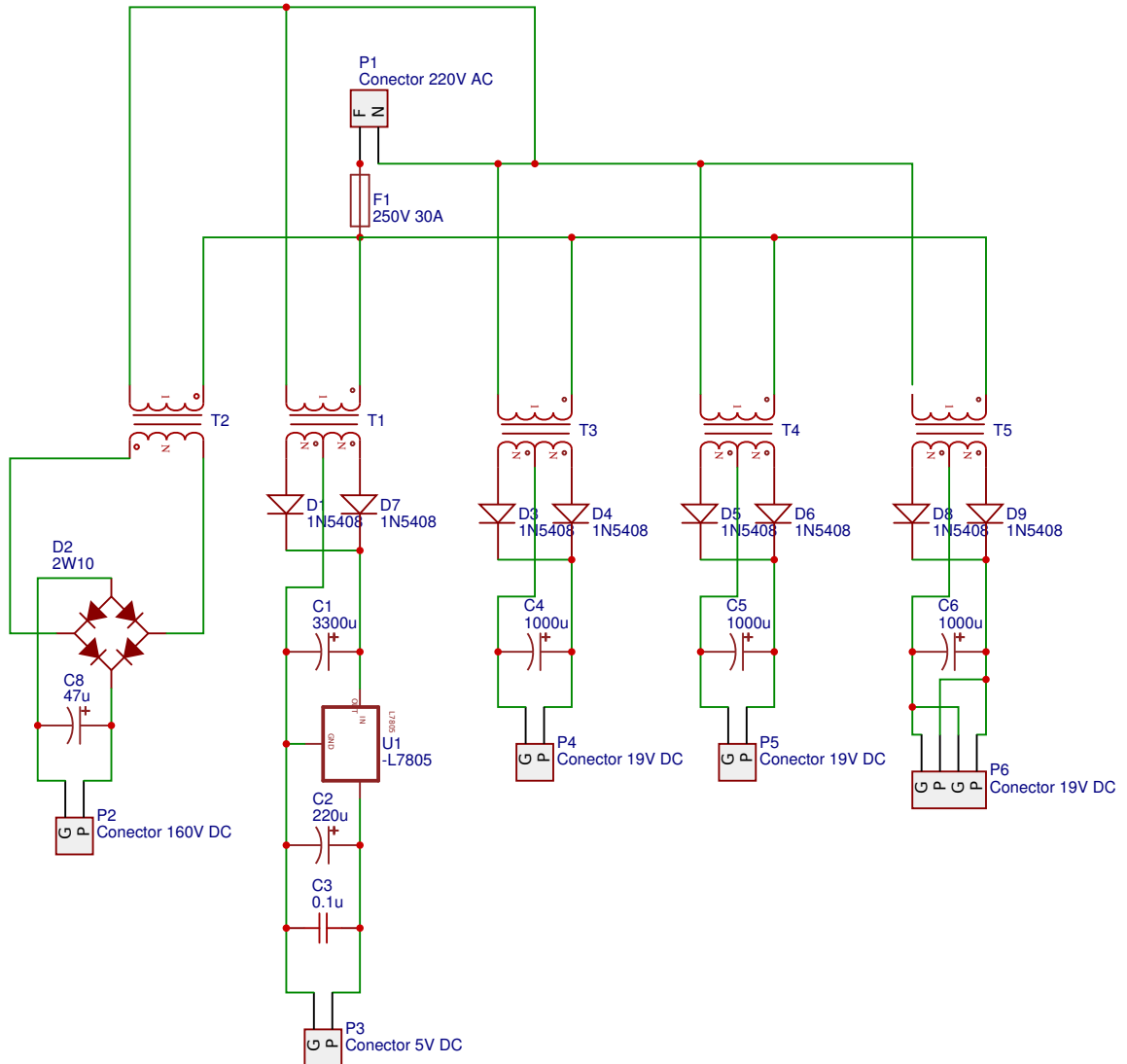
WELSHANS, T. W. **A Brief History of Aircraft Carburetors and Fuel Systems**. 2013. For the Aircraft Engine Historical Society. Disponível em: <<http://www.enginehistory.org/Accessories/HxFuelSys/FuelSysHx01.shtml>>. Acesso em: 27 set. 2017.

Diagrama eletrônico para circuito de controle



Fonte: O autor.

Diagrama eletrônico para circuito de alimentação



Fonte: O autor.

APÊNDICE B – CÓDIGOS-FONTES

Código-fonte 1 – Controle

```

1  /**
2
3   Esse software é fornecido sem nenhuma garantia , use por sua própria conta e risco .
4   Em nenhum momento o autor desse software poderá ser acusado por qualquer dano ou perda
5   que o uso desse código venha a causar a qualquer usuário ou produto .
6
7   O uso desse software constata a aceitação dos termos apresentados acima .
8
9   Este software foi desenvolvido para uso no microcontrolador Microchip PIC18F45K22 .
10  Compilador: XC8 1.35
11  IDE: MPLAB X 3.40
12
13  Esse software faz uso de códigos gerados pela Microchip Inc. ; Outros termos de uso podem
14     ser aplicados .
15  */
16
17  /*
18  Esquema de pinos :
19  * RC6= Eusart TX (Transmissor)
20  * RC7= Eusart RX (Receptor)
21  * RA0= Eusart "multiplex" angle/inj_time
22  * RA1= "Multiplex" para a faixa de ângulo
23  * RC3= IGBT inj aberta T1
24  * RC1= IGBT inj aberta T4
25  * RC2= IGBT inj fechada T3
26  * RC0= IGBT inj fechada T2
27  * RB0= Pino de contagem de ângulo
28  * RB1= Pino para reset de ângulo
29  *
30  IGBT Status :
31  * 0 = Todos os IGBTs desligados
32  * 1 = IGBT 1 e 4 ligados , 2 e 3 desligados , Injetor aberto
33  * 2 = Todos os IGBTs desligados , dead time para IGBT 1 e 4
34  * 3 = IGBT 2 e 3 ligados , 1 e 4 desligados , Injetor fechado
35  * 4 = Todos IGBTs desligados , dead time para IGBT 2 e 3
36  */
37
38  #include <xc.h>
39  #include <stdint.h>
40  #include "stdbool.h"
41
42  #define INTERRUPT_GlobalInterruptHighEnable() (INTCONbits.GIEH = 1)
43  #define INTERRUPT_GlobalInterruptHighDisable() (INTCONbits.GIEH = 0)

```

```

44 #define INTERRUPT_GlobalInterruptLowEnable() (INTCONbits.GIEL = 1)
45 #define INTERRUPT_GlobalInterruptLowDisable() (INTCONbits.GIEL = 0)
46
47 // Variables:
48 volatile uint16_t timer1ReloadVal;
49 volatile uint16_t timer3ReloadVal;
50 volatile float angle_val=0; // Ângulo de injeção desejado. Múltiplo de 0.36 (1000 pulsos
    por revolução)
51 volatile float angle_count=0; // Ângulo atual
52 volatile float inj_time=13000; // Variável para cálculo de inj_bits através dos dados
    recebidos do UART
53 volatile uint16_t inj_bits=14535; // Duração de injeção; FOSC=16MHz e timer1 pre-scale=1/2,
    1 incremento do contador = 0.5us
54 volatile uint8_t IGBT_status=0; // Variável para checar o status da ponte H
55 volatile uint8_t eusart1_val; // Variável para guardar dados recebidos pelo UART
56 volatile uint8_t power_stroke=1; // Variável usada para definir se o motor está no ciclo de
    combustão
57 volatile uint8_t angle_neg=0; // Variável para checar se o ângulo de injeção desejado é
    negativo
58 volatile uint8_t local_known=0; // Variável usada para determinar se o microcontrolador
    sabe o ângulo atual exato
59
60
61 // Configuration bits: Código gerado com o auxílio do MPLAB X IDE
62
63 // CONFIG1H
64 #pragma config FOSC = INTIO67 // Oscillator Selection bits->Internal oscillator block
65 #pragma config PLLCFG = OFF // 4X PLL Enable/Disable->Oscillator used directly
66 #pragma config PRICLK = ON // Primary clock enable bit->Primary clock is always
    enabled
67 #pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit->Fail-Safe Clock
    Monitor disabled
68 #pragma config IESO = OFF // Internal/External Oscillator Switchover bit->Oscillator
    Switchover mode disabled
69
70 // CONFIG2L
71 #pragma config PWRTE = ON // Power-up Timer Enable bit->Power up timer disabled
72 #pragma config BOREN = SBORDIS // Brown-out Reset Enable bits->Brown-out Reset enabled
    in hardware only (SBORDIS is disabled)
73 #pragma config BORV = 190 // Brown Out Reset Voltage bits->VBOR set to 1.90 V nominal
74
75 // CONFIG2H
76 #pragma config WDTEN = OFF // Watchdog Timer Enable bits->Watch dog timer is always
    disabled. SWDTEN has no effect.
77 #pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits->1:32768
78
79 // CONFIG3H
80 #pragma config CCP2MX = PORTC1 // CCP2 MUX bit->CCP2 input/output is multiplexed with
    RC1

```

```

81 #pragma config PBADEN = OFF    // PORTB A/D Enable bit->PORTB<5:0> pins are configured as
    digital I/O on Reset
82 #pragma config CCP3MX = PORTB5 // P3A/CCP3 Mux bit->P3A/CCP3 input/output is multiplexed
    with RB5
83 #pragma config HFOFST = ON     // HFINTOSC Fast Start-up->HFINTOSC output and ready status
    are not delayed by the oscillator stable status
84 #pragma config T3CMX = PORTC0  // Timer3 Clock input mux bit->T3CKI is on RC0
85 #pragma config P2BMX = PORTD2  // ECCP2 B output mux bit->P2B is on RD2
86 #pragma config MCLRE = INTMCLR // MCLR Pin Enable bit->MCLR pin disabled, RE3 input pin
    enabled
87
88 // CONFIG4L
89 #pragma config STVREN = ON     // Stack Full/Underflow Reset Enable bit->Stack full/
    underflow will cause Reset
90 #pragma config LVP = ON       // Single-Supply ICSP Enable bit->Single-Supply ICSP enabled if
    MCLRE is also 1
91 #pragma config XINST = OFF    // Extended Instruction Set Enable bit->Instruction set
    extension and Indexed Addressing mode disabled (Legacy mode)
92 #pragma config DEBUG = OFF    // Background Debug->Disabled
93
94 // CONFIG5L
95 #pragma config CP0 = OFF      // Code Protection Block 0->Block 0 (000800-001FFFh) not code-
    protected
96 #pragma config CP1 = OFF      // Code Protection Block 1->Block 1 (002000-003FFFh) not code-
    protected
97 #pragma config CP2 = OFF      // Code Protection Block 2->Block 2 (004000-005FFFh) not code-
    protected
98 #pragma config CP3 = OFF      // Code Protection Block 3->Block 3 (006000-007FFFh) not code-
    protected
99
100 // CONFIG5H
101 #pragma config CPB = OFF      // Boot Block Code Protection bit->Boot block (000000-0007FFFh)
    not code-protected
102 #pragma config CPD = OFF      // Data EEPROM Code Protection bit->Data EEPROM not code-
    protected
103
104 // CONFIG6L
105 #pragma config WRT0 = OFF     // Write Protection Block 0->Block 0 (000800-001FFFh) not
    write-protected
106 #pragma config WRT1 = OFF     // Write Protection Block 1->Block 1 (002000-003FFFh) not
    write-protected
107 #pragma config WRT2 = OFF     // Write Protection Block 2->Block 2 (004000-005FFFh) not
    write-protected
108 #pragma config WRT3 = OFF     // Write Protection Block 3->Block 3 (006000-007FFFh) not
    write-protected
109
110 // CONFIG6H
111 #pragma config WRTC = OFF     // Configuration Register Write Protection bit->Configuration
    registers (300000-3000FFFh) not write-protected

```



```

112 #pragma config WRIB = OFF // Boot Block Write Protection bit->Boot Block (000000-0007FFh
    ) not write-protected
113 #pragma config WRID = OFF // Data EEPROM Write Protection bit->Data EEPROM not write-
    protected
114
115 // CONFIG7L
116 #pragma config EBTR0 = OFF // Table Read Protection Block 0->Block 0 (000800-001FFFh)
    not protected from table reads executed in other blocks
117 #pragma config EBTR1 = OFF // Table Read Protection Block 1->Block 1 (002000-003FFFh)
    not protected from table reads executed in other blocks
118 #pragma config EBTR2 = OFF // Table Read Protection Block 2->Block 2 (004000-005FFFh)
    not protected from table reads executed in other blocks
119 #pragma config EBTR3 = OFF // Table Read Protection Block 3->Block 3 (006000-007FFFh)
    not protected from table reads executed in other blocks
120
121 // CONFIG7H
122 #pragma config EBTRB = OFF // Boot Block Table Read Protection bit->Boot Block
    (000000-0007FFFh) not protected from table reads executed in other blocks
123
124
125 // Declarações das funções:
126
127
128
129 void interrupt High_ISR(void); // Função para servir interrupções de alta prioridade
130 void interrupt low_priority Low_ISR(void); // Função para servir interrupções de baixa
    prioridade
131
132 void TMR1_StartTimer(void);
133 void TMR1_StopTimer(void);
134 uint16_t TMR1_ReadTimer(void);
135 void TMR1_WriteTimer(uint16_t timerVal);
136 void TMR1_Reload(void);
137
138 void TMR3_StartTimer(void);
139 void TMR3_StopTimer(void);
140 uint16_t TMR3_ReadTimer(void);
141 void TMR3_WriteTimer(uint16_t timerVal);
142 void TMR3_Reload(void);
143
144
145
146 void main(void)
147 {
148     #define _XTAL_FREQ 16000000
149
150
151
152     // Ativa os vetores de prioridade das interrupções

```

```
153     RCONbits.IPEN = 1;
154
155     // Configura os vetores de prioridade para cada interrupção:
156
157     // Interrupção INT0I não possui bit de prioridade. Será sempre de alta prioridade.
158
159     // INT1I – alta prioridade
160     INTCON3bits.INT1IP = 1;
161
162     // TMR3 – alta prioridade
163     IPR2bits.TMR3IP = 1;
164
165     // TMRI – alta prioridade
166     IPR1bits.TMR1IP = 1;
167
168     // RCI – baixa prioridade
169     IPR1bits.RC1IP = 0;
170
171
172     // Ativa interrupção INT1 em rising edge (aumento do potencial no pino):
173     INTCON3bits.INT1IF = 0;
174     INTCON2bits.INTEDG1 = 1;
175     INTCON3bits.INT1IE = 1;
176
177     // Ativa interrupção INT0 em rising edge (aumento do potencial no pino):
178     INTCONbits.INT0IF = 0;
179     INTCON2bits.INTEDG0 = 1;
180     INTCONbits.INT0IE = 1;
181
182
183     // Configuração do oscilador:
184
185     // SCS FOSC; IRCF 16MHz_HFINTOSC; IDLEN desativado;
186     OSCCON = 0x70;
187     // PRISD ativado; SOSCGO desativado; MFIOSSEL desativado;
188     OSCCON2 = 0x04;
189     // INTSRC desativado; PLEN desativado; TUN 0;
190     OSCTUNE = 0x00;
191
192     // Configuração dos pinos:
193
194     // Nenhum pino analógico:
195     ANSELA=0x00;
196     ANSELB=0x00;
197     ANSELC=0x00;
198     ANSELD=0x00;
199     ANSELE=0x00;
200     ANSELCbits.ANSC7=0;
201     ANSELCbits.ANSC6=0;
```

```

202
203     TRISE = 0x00;
204     TRISA = 0b00000011;
205     TRISB = 0b00000011;
206     TRISC = 0b11000000;
207     TRISD = 0x00;
208     INTCON2bits.nRBPU = 1; // Desabilita resistores "weak pull-ups" na PORTA B
209
210     LATC = 0x00; // Saída digital na porta C baixa
211
212     // Configuração do EUSART1:
213
214     // Desativa interrupções antes de mudar configuração
215     PIE1bits.RC1IE = 0;
216     PIE1bits.TX1IE = 0;
217
218     // ABDONV no_overflow; CKTXP async_noninverted_sync_fallingedge; BRG16 16bit_generator;
219     // WUE desativado; ABDEN desativado; DTRXP not_inverted;
220     BAUDCON1 = 0x08;
221
222     // SPEN ativado; RX9 8-bit; CREN ativado; ADDEN desativado; SREN desativado;
223     RCSTA1 = 0x90;
224
225     // TX9 8-bit; TX9D 0; SENDB sync_break_complete; TXEN desativado; SYNC asynchronous;
226     // BRGH hi_speed; CSRC slave_mode;
227     TXSTA1 = 0x04;
228
229     // Baud Rate = 10417;
230     SPBRG1 = 0x7F;
231
232     // Baud Rate = 10417;
233     SPBRGH1 = 0x01;
234
235     // Ativa interrupção de recepção
236     PIE1bits.RC1IE = 1;
237     PIE1bits.TX1IE = 0;
238
239     // Configuração do TIMER1
240
241     //TICKPS 1:2; TIOSCEN desativado; TISYNC synchronize; TMRICS FOSC/4; TMRION off; T1RD16
242     // desativado;
243     TICON = 0x10;
244
245     //T1GSS T1G_pin; TMR1GE desativado; T1GTM desativado; T1GPOL low; T1GGO done; T1GSPM
246     // desativado;
247     T1GCON = 0x00;
248
249     //TMR1H 56;
250     TMR1H = 0x38;

```

```
247
248 //TMR1L 200;
249 TMR1L = 0xC8;
250
251 // Carrega o valor do timer na variável para reload
252 timer1ReloadVal=TMR1;
253
254 // Limpando bandeira antes de ativar interrupção
255 PIR1bits.TMR1IF = 0;
256
257 // Ativa interrupção TMR1
258 PIE1bits.TMR1IE = 1;
259
260 // Configuração do TIMER3
261
262 //T3CKPS 1:1; T3OSCEN desativado; nT3SYNC synchronize; TMR3CS FOSC/4; TMR3ON off;
263 // T3RD16 ativado;
264 T3CON = 0x02;
265
266 //T3GSS T3G_pin; TMR3GE desativado; T3GTM desativado; T3GPOL low; T3GGO done; T3GSPM
267 // desativado;
268 T3GCON = 0x00;
269
270 TMR3H = 0xFF;
271
272 TMR3L = 0xDB;
273
274 timer3ReloadVal=60000;
275
276 // Limpando bandeira antes de ativar interrupção
277 PIR2bits.TMR3IF = 0;
278
279 // Ativa interrupção TMR3
280 PIE2bits.TMR3IE = 1;
281
282 // Ativa interrupções globais de alta prioridade
283 INTERRUPT_GlobalInterruptHighEnable();
284
285 // Ativa interrupções globais de baixa prioridade
286 INTERRUPT_GlobalInterruptLowEnable();
287
288
289 while (1)
290 {
291
292 // Todo o código é baseado em interrupções
293
```

```

294     }
295
296 }
297
298
299 void interrupt High_ISR(void) // Rotina de serviço para interrupções de alta prioridade
300 {
301
302     if((INTCONbits.INT0IE == 1 && INTCONbits.INT0IF == 1) || (INTCON3bits.INT1IE == 1 &&
303         INTCON3bits.INT1IF == 1) ) // Interrupção do pino de incremento ou reset
304     {
305         if(INTCON3bits.INT1IE == 1 && INTCON3bits.INT1IF == 1){ // Interrupção de reset
306             angle_count=0;
307             if(local_known==0){
308                 local_known=1;
309             }
310             if(power_stroke==1){
311                 power_stroke=0;
312             }
313             else{
314                 power_stroke=1;
315             }
316             INTCON3bits.INT1IF = 0;
317         }
318         else if (INTCONbits.INT0IE == 1 && INTCONbits.INT0IF == 1){ // Interrupção de
319             incremento
320             angle_count +=0.36;
321             INTCONbits.INT0IF = 0;
322         }
323
324         if (angle_count <=angle_val+0.3 && angle_count >=angle_val -0.3 && IGBT_status==0 &&
325             local_known==1){
326             if((angle_neg==0 && power_stroke==1) || (angle_neg==1 && power_stroke==0)){
327
328                 LATCbits.LATC3=1; // IGBT 1
329                 LATCbits.LATC1=1; // IGBT 4
330                 IGBT_status=1;
331                 TMR1_WriteTimer(inj_bits); // Seta timer para duração da injeção
332                 TMR1_StartTimer();
333             }
334         }
335
336         if (angle_count >=130 && IGBT_status==1 && power_stroke==1){ // Fecha injetor se
337             ainda estiver aberto depois dos 130 graus
338             TMR1_StopTimer();
339             LATCbits.LATC3=0;

```

```

339         LATCbits.LATC1=0;
340         IGBT_status=2;
341         // Timer para dead time dos IGBTs
342         TMR3_Reload();
343         TMR3_StartTimer();
344     }
345
346
347 }
348
349
350 if(PIE2bits.TMR3IE == 1 && PIR2bits.TMR3IF == 1) // Interrupção do Timer3, IGBTs dead
    time
351 {
352
353     TMR3_StopTimer();
354     if (IGBT_status==2){
355         LATCbits.LATC2=1;
356         LATCbits.LATC0=1;
357         IGBT_status=3;
358         TMR3_Reload();
359         TMR3_StartTimer();
360
361     }
362     else if (IGBT_status==3){
363         LATCbits.LATC2=0;
364         LATCbits.LATC0=0;
365         IGBT_status=4;
366         TMR3_Reload();
367         TMR3_StartTimer();
368     }
369     else if (IGBT_status==4){
370         IGBT_status=0;
371     }
372     PIR2bits.TMR3IF = 0;
373 }
374
375 if(PIE1bits.TMR1IE == 1 && PIR1bits.TMR1IF == 1) // Interrupção do Timer1, timer para a
    duração de injeção
376 {
377
378     TMR1_StopTimer();
379     if (IGBT_status==1){
380         LATCbits.LATC3=0;
381         LATCbits.LATC1=0;
382         IGBT_status=2;
383
384     }
385

```

```

386     // Começa timer para o dead time dos IGBTs
387     TMR3_Reload();
388     TMR3_StartTimer();
389     PIR1bits.TMR1IF = 0;
390
391 }
392
393 }
394
395 void interrupt low_priority Low_ISR(void) // Rotina de serviço para interrupções de baixa
    prioridade
396 {
397
398     if(PIE1bits.RC1IE == 1 && PIR1bits.RC1IF == 1) // Interrupção do Eusart1
399     {
400         if(1 == RCSTA1bits.OERR){
401
402             // EUSART1 error - restart
403
404             RCSTA1bits.CREN = 0;
405             RCSTA1bits.CREN = 1;
406         }
407
408         eusart1_val=RCREG1; // Lê o dado de 8bits do registro
409         if(PORTAbits.RA0==1){
410             if(PORTAbits.RA1==0){
411                 angle_val=-45.72+(float) eusart1_val*0.36;
412             }
413             else if(PORTAbits.RA1==1){
414                 angle_val=100-45.72+ (float) eusart1_val*0.36;
415             }
416         }
417         if(angle_val<0){
418             angle_val=360+angle_val;
419             angle_neg=1;
420         }
421
422     }
423     else if(PORTAbits.RA0==0){
424         inj_time=(float) eusart1_val/(0.005);
425         inj_bits=65535-(uint16_t) inj_time;
426
427     }
428
429
430
431 }
432
433

```

```
434 }
435
436 void TMR1_StartTimer(void)
437 {
438     T1CONbits.TMRION = 1;
439     return;
440 }
441
442 void TMR1_StopTimer(void)
443 {
444     T1CONbits.TMRION = 0;
445     return;
446 }
447
448 uint16_t TMR1_ReadTimer(void)
449 {
450     uint16_t readVal;
451     uint8_t readValHigh;
452     uint8_t readValLow;
453
454     readValLow = TMRIL;
455     readValHigh = TMRIH;
456
457     readVal = ((uint16_t)readValHigh << 8) | readValLow;
458
459     return readVal;
460 }
461
462 void TMR1_WriteTimer(uint16_t timerVal)
463 {
464     if (T1CONbits.T1SYNC == 1)
465     {
466
467         T1CONbits.TMRION = 0;
468
469
470         TMRIH = (timerVal >> 8);
471         TMRIL = (uint8_t) timerVal;
472
473
474         T1CONbits.TMRION =1;
475     }
476     else
477     {
478
479         TMRIH = (timerVal >> 8);
480         TMRIL = (uint8_t) timerVal;
481     }
482     return;
```



```
483 }
484
485 void TMR1_Reload(void)
486 {
487
488     TMR1H = (timer1ReloadVal >> 8);
489     TMR1L = (uint8_t) timer1ReloadVal;
490     return;
491 }
492
493 void TMR3_StartTimer(void)
494 {
495
496     T3CONbits.TMR3ON = 1;
497     return;
498 }
499
500 void TMR3_StopTimer(void)
501 {
502
503     T3CONbits.TMR3ON = 0;
504     return;
505 }
506
507 uint16_t TMR3_ReadTimer(void)
508 {
509     uint16_t readVal;
510     uint8_t readValHigh;
511     uint8_t readValLow;
512
513     readValLow = TMR3L;
514     readValHigh = TMR3H;
515
516     readVal = ((uint16_t)readValHigh << 8) | readValLow;
517
518     return readVal;
519 }
520
521 void TMR3_WriteTimer(uint16_t timerVal)
522 {
523     if (T3CONbits.nT3SYNC == 1)
524     {
525
526         T3CONbits.TMR3ON = 0;
527
528
529         TMR3H = (timerVal >> 8);
530         TMR3L = (uint8_t) timerVal;
531
```

```

532
533     T3CONbits.TMR3ON =1;
534 }
535 else
536 {
537
538     TMR3H = (timerVal >> 8);
539     TMR3L = (uint8_t) timerVal;
540 }
541 return;
542 }
543
544 void TMR3_Reload(void)
545 {
546
547     TMR3H = (timer3ReloadVal >> 8);
548     TMR3L = (uint8_t) timer3ReloadVal;
549     return;
550 }

```

Código-fonte 2 – Interação com Usuário

```

1  /**
2
3   Esse software é fornecido sem nenhuma garantia , use por sua própria conta e risco .
4   Em nenhum momento o autor desse software poderá ser acusado por qualquer dano ou perda
5   que o uso desse código venha a causar a qualquer usuário ou produto .
6
7   O uso desse software constata a aceitação dos termos apresentados acima .
8
9   Este software foi desenvolvido para uso no microcontrolador Microchip PIC18F45K22 .
10  Compilador: XC8 1.35
11  IDE: MPLAB X 3.40
12
13  Esse software faz uso de códigos gerados pela Microchip Inc.; Outros termos de uso podem
14  ser aplicados .
15  */
16
17
18  /*
19  Esquema dos pinos:
20  * RB0= pino de mudança entre ângulo e injeção
21  * RB1= ângulo/inj +1
22  * RB2= ângulo/inj -1
23  * RB3= ângulo/inj +30
24  * RB4= ângulo/inj -30
25  * RB5= faixa de ângulo 0

```

```

26 * RB6= faixa de ângulo 100
27 * RC0= Eusart "multiplex" para enviar angle e inj_time
28 * RC1= "multiplex" da faixa de ângulo
29 * RC6= Eusart TX
30 * RC7= Eusart RX
31 * RA0= LCD D0
32 * RA1= LCD D1
33 * RA2= LCD D2
34 * RA3= LCD D3
35 * RA4= LCD D4
36 * RA5= LCD D5
37 * RA6= LCD D6
38 * RA7= LCD D7
39 * RD5= LCD E
40 * RD6= LCD RW
41 * RD7 LCD RS
42
43 */
44
45
46 #include <xc.h>
47 #include <stdint.h>
48 #include <stdbool.h>
49 #include <stdio.h>
50 #include "xlcd.h"
51 #include "delays.h"
52
53 // Declarações de funções:
54 void EUSART1_Write(uint8_t txData);
55 void LCD_printdata(void);
56 void Eusart_senddata(void);
57
58 void DelayFor18TCY(void); // Delay por 18 ciclos de instrução
59 void DelayPORXLCD(void); // Delay 15ms
60 void DelayXLCD(void); // Delay 5ms
61
62 // Declarações de variáveis:
63 volatile float angle; // Ângulo desejado de injeção
64 volatile float inj_time; // Duração de injeção desejada
65 char line1[30]; // Variáveis usadas para expor texto no lcd
66 char line2[30];
67 uint8_t mode=0; // Botão para mudar entre ângulo ou tempo desejado
68 uint8_t angle_range=0;
69 volatile uint8_t angle_count=127; // Ângulo desejado convertido para um inteiro de 8bit
70 volatile uint8_t inj_count=155; // Duração desejada convertida para um inteiro de 8bit
71
72
73 // Bits de configuração: Código gerado com o auxílio do MPLAB X IDE
74

```

```

75 #define _XTAL_FREQ 16000000
76
77 // CONFIG1H
78 #pragma config FOSC = INTIO67 // Oscillator Selection bits->Internal oscillator block
79 #pragma config PLLCFG = OFF // 4X PLL Disable->Oscillator not multiplied by 4
80 #pragma config PRICLK = ON // Primary clock enable bit->Primary clock is always
    enabled
81 #pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit->Fail-Safe Clock
    Monitor disabled
82 #pragma config IESO = OFF // Internal/External Oscillator Switchover bit->Oscillator
    Switchover mode disabled
83
84 // CONFIG2L
85 #pragma config PWRTEN = ON // Power-up Timer Enable bit->Power up timer disabled
86 #pragma config BOREN = SBORDIS // Brown-out Reset Enable bits->Brown-out Reset enabled
    in hardware only (SBOREN is disabled)
87 #pragma config BORV = 190 // Brown Out Reset Voltage bits->VBOR set to 1.90 V nominal
88
89 // CONFIG2H
90 #pragma config WDTE = OFF // Watchdog Timer Enable bits->Watch dog timer is always
    disabled. SWDTE has no effect.
91 #pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits ->1:32768
92
93 // CONFIG3H
94 #pragma config CCP2MX = PORTC1 // CCP2 MUX bit->CCP2 input/output is multiplexed with
    RC1
95 #pragma config PBAEN = OFF // PORTB A/D Enable bit->PORTB<5:0> pins are configured as
    digital input channels on Reset
96 #pragma config CCP3MX = PORTB5 // P3A/CCP3 Mux bit->P3A/CCP3 input/output is multiplexed
    with RB5
97 #pragma config HFOFST = ON // HFINTOSC Fast Start-up->HFINTOSC output and ready status
    are not delayed by the oscillator stable status
98 #pragma config T3CMX = PORTC0 // Timer3 Clock input mux bit->T3CKI is on RC0
99 #pragma config P2BMX = PORTD2 // ECCP2 B output mux bit->P2B is on RD2
100 #pragma config MCLRE = INTMCLR // MCLR Pin Enable bit->MCLR pin disable, RE3 input pin
    enable
101
102 // CONFIG4L
103 #pragma config STVREN = ON // Stack Full/Underflow Reset Enable bit->Stack full/
    underflow will cause Reset
104 #pragma config LVP = ON // Single-Supply ICSP Enable bit->Single-Supply ICSP enabled if
    MCLRE is also 1
105 #pragma config XINST = OFF // Extended Instruction Set Enable bit->Instruction set
    extension and Indexed Addressing mode disabled (Legacy mode)
106 #pragma config DEBUG = OFF // Background Debug->Disabled
107
108 // CONFIG5L
109 #pragma config CP0 = OFF // Code Protection Block 0->Block 0 (000800-001FFFh) not code-
    protected

```

```

110 #pragma config CPI = OFF // Code Protection Block 1->Block 1 (002000-003FFFh) not code-
    protected
111 #pragma config CP2 = OFF // Code Protection Block 2->Block 2 (004000-005FFFh) not code-
    protected
112 #pragma config CP3 = OFF // Code Protection Block 3->Block 3 (006000-007FFFh) not code-
    protected
113
114 // CONFIG5H
115 #pragma config CPB = OFF // Boot Block Code Protection bit->Boot block (000000-0007FFFh)
    not code-protected
116 #pragma config CPD = OFF // Data EEPROM Code Protection bit->Data EEPROM not code-
    protected
117
118 // CONFIG6L
119 #pragma config WRT0 = OFF // Write Protection Block 0->Block 0 (000800-001FFFh) not
    write-protected
120 #pragma config WRT1 = OFF // Write Protection Block 1->Block 1 (002000-003FFFh) not
    write-protected
121 #pragma config WRT2 = OFF // Write Protection Block 2->Block 2 (004000-005FFFh) not
    write-protected
122 #pragma config WRT3 = OFF // Write Protection Block 3->Block 3 (006000-007FFFh) not
    write-protected
123
124 // CONFIG6H
125 #pragma config WRIC = OFF // Configuration Register Write Protection bit->Configuration
    registers (300000-3000FFFh) not write-protected
126 #pragma config WRIB = OFF // Boot Block Write Protection bit->Boot Block (000000-0007FFFh)
    ) not write-protected
127 #pragma config WRID = OFF // Data EEPROM Write Protection bit->Data EEPROM not write-
    protected
128
129 // CONFIG7L
130 #pragma config EBTR0 = OFF // Table Read Protection Block 0->Block 0 (000800-001FFFh)
    not protected from table reads executed in other blocks
131 #pragma config EBTR1 = OFF // Table Read Protection Block 1->Block 1 (002000-003FFFh)
    not protected from table reads executed in other blocks
132 #pragma config EBTR2 = OFF // Table Read Protection Block 2->Block 2 (004000-005FFFh)
    not protected from table reads executed in other blocks
133 #pragma config EBTR3 = OFF // Table Read Protection Block 3->Block 3 (006000-007FFFh)
    not protected from table reads executed in other blocks
134
135 // CONFIG7H
136 #pragma config EBTRB = OFF // Boot Block Table Read Protection bit->Boot Block
    (000000-0007FFFh) not protected from table reads executed in other blocks
137
138
139 void main(void)
140 {
141     // Configuração de pinos:

```

```

142
143 // Nenhum pino analógico
144 ANSELA=0x00;
145 ANSELB=0x00;
146 ANSELC=0x00;
147 ANSELD=0x00;
148 ANSELE=0x00;
149 ANSELCbits.ANSC7=0;
150 ANSELCbits.ANSC6=0;
151
152 TRISE = 0x00;
153 TRISA = 0x00;
154 TRISB = 0xFF;
155 TRISC = 0b11000000;
156 TRISD = 0x00;
157 WPUB = 0b11111111; // Escolha de resistores com weak pull-ups ativados na porta B
158 INTCON2bits.nRBPU = 0; // Ativação dos resistores de weak pull-ups na porta B
159
160
161 // Configuração do oscilador:
162 // SCS FOSC; IRCF 16MHz_HFINTOSC; IDLEN desativado;
163 OSCCON = 0x70;
164 // PRISD ativado; SOSCGO desativado; MFIOSEL desativado;
165 OSCCON2 = 0x04;
166 // INTSRC desativado; PLEN desativado; TUN 0;
167 OSCTUNE = 0x00;
168
169 // Configuração do EUSART1:
170 // ABDVDF no_overflow; CKTXP async_noninverted_sync_fallingedge; BRG16 16bit_generator;
171 // WUE desativado; ABDEN desativado; DTRXP not_inverted;
172 BAUDCON1 = 0x08;
173
174 // SPEN ativado; RX9 8-bit; CREN ativado; ADDEN desativado; SREN desativado;
175 RCSTA1 = 0x90;
176
177 // TX9 8-bit; TX9D 0; SENDB sync_break_complete; TXEN ativado; SYNC asynchronous; BRGH
178 // hi_speed; CSRC slave_mode;
179 TXSTA1 = 0x24;
180
181 // Baud Rate = 10417;
182 SPBRG1 = 0x7F;
183
184 // Baud Rate = 10417;
185 SPBRGH1 = 0x01;
186
187 OpenXLCD(EIGHT_BIT & LINES_5X7); // Função para iniciar LCD; Parte da biblioteca XLCD
188 // da Microchip Inc.
189
190 while(BusyXLCD()); // Espera a desocupação do processador do lcd

```

```
188
189 WriteCmdXLCD(0x06); // Movimenta o cursor para direita e não movimenta a linha
190
191 while (BusyXLCD());
192 WriteCmdXLCD(0x0C); // Liga a tela sem cursor
193 while (BusyXLCD());
194
195 Eusart_senddata(); // Envia dados para o EUSART1 e para o LCD.
196
197
198 while (1) // Procurando por entrada do usuário
199 {
200     if (PORTBbits.RB0==0){
201         if (mode==0){
202             mode=1;
203         }
204         else {
205             mode=0;
206         }
207         Eusart_senddata();
208     }
209
210     if (PORTBbits.RB1==0){
211         if (mode==0){
212             if (angle_count!=255){
213                 angle_count+=1;
214             }
215         }
216         else {
217             if (inj_count!=255){
218                 inj_count+=1;
219             }
220
221         }
222         Eusart_senddata();
223     }
224
225     if (PORTBbits.RB2==0){
226         if (mode==0){
227             if (angle_count!=0){
228                 angle_count -=1;
229             }
230         }
231         else {
232             if (inj_count!=0){
233                 inj_count -=1;
234             }
235         }
236         Eusart_senddata();
```

```
237     }
238
239     if (PORTBbits.RB3==0){
240         if (mode==0){
241             if (angle_count >=225){
242                 angle_count=255;
243             }
244             else{
245                 angle_count+=30;
246             }
247         }
248         else{
249             if (inj_count >=225){
250                 inj_count=255;
251             }
252             else{
253                 inj_count+=30;
254             }
255         }
256     }
257     Eusart_senddata ();
258 }
259
260
261 if (PORTBbits.RB4==0){
262     if (mode==0){
263         if (angle_count <=30){
264             angle_count=0;
265         }
266         else{
267             angle_count -=30;
268         }
269     }
270     else{
271         if (inj_count <=30){
272             inj_count=0;
273         }
274         else{
275             inj_count -=30;
276         }
277     }
278 }
279
280 Eusart_senddata ();
281 }
282
283
284 if (PORTBbits.RB5==0){
285     angle_range =0;
```



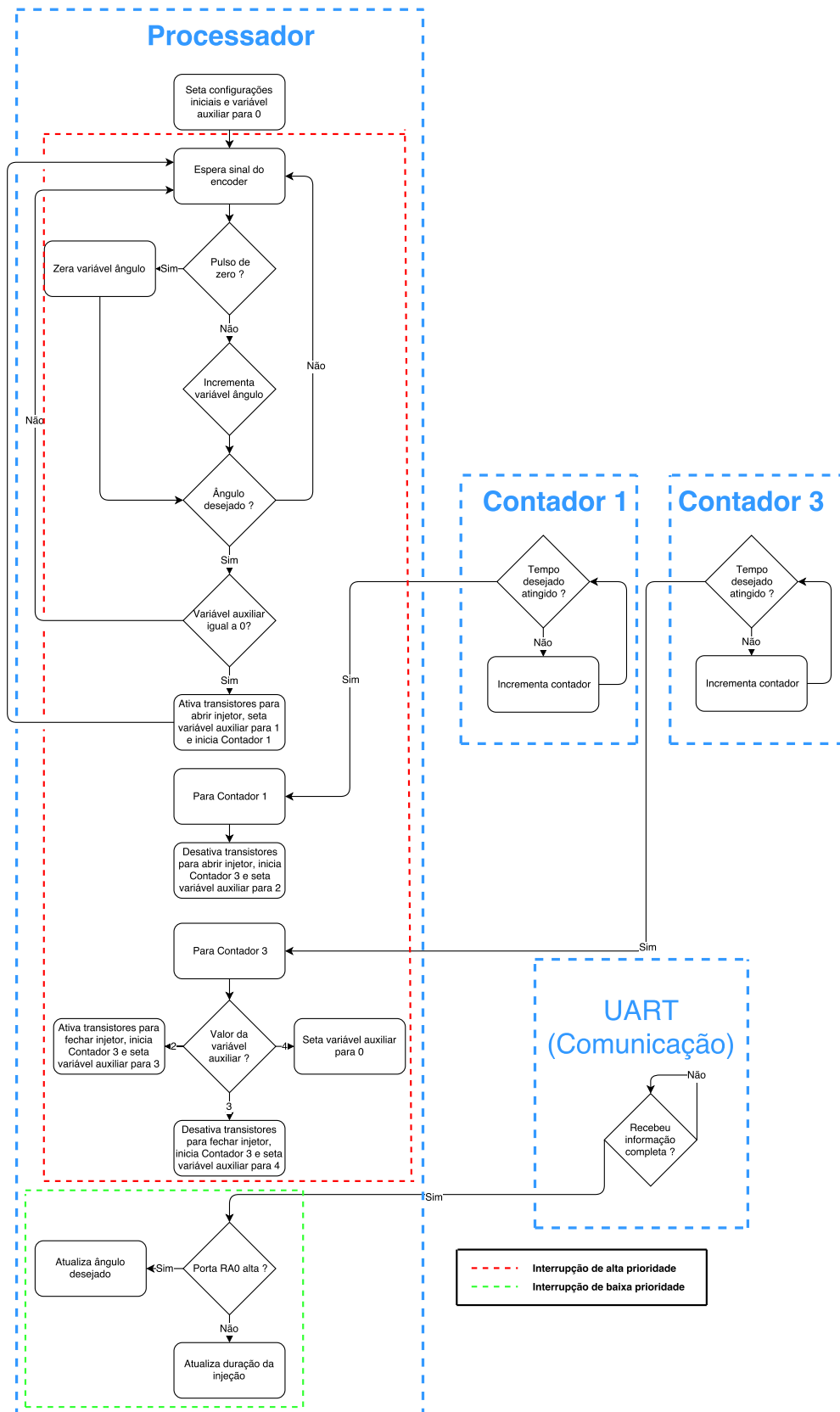
```

286         PORTCbits.RC1=0;
287         Eusart_senddata ();
288     }
289
290     if (PORTBbits.RB6==0){
291         angle_range=1;
292         PORTCbits.RC1=1;
293         Eusart_senddata ();
294     }
295
296     Delay10KTCYx(70); // Delay para registrar somente uma entrada por pressionamento do
                botão
297
298
299
300     }
301 }
302
303
304 void EUSART1_Write(uint8_t txData)
305 {
306     while(0 == PIR1bits.TX1IF)
307     {
308
309
310     }
311
312     TXREG1 = txData; // Escreve o dados no registro do EUSART1
313 }
314
315 void LCD_printdata(void){ // Função para atualizar a tela do LCD
316     if(angle_range==0){
317         angle=-45.72+ (float) angle_count*0.36;
318     }
319     else{
320         angle=100-45.72+ (float) angle_count*0.36;
321     }
322     inj_time=(float) inj_count/10;
323     sprintf(line1 , "Angulo:%.2f" ,angle);
324     sprintf(line2 , "Tempo:%.2f" ,inj_time);
325     WriteCmdXLCD(0x01); // Limpa tela
326     while(BusyXLCD());
327     WriteCmdXLCD(0x02); // Move o cursor para a posição inicial do LCD
328     while(BusyXLCD());
329     putsXLCD(line1);
330     while(BusyXLCD());
331     SetDDRamAddr(0x40); // Move o cursorsr para a segunda linha
332     while(BusyXLCD());
333     putsXLCD(line2);

```

```
334     while (BusyXLCD());
335
336 }
337
338 void Eusart_senddata(void){
339
340     PORTCbits.RC0=1;
341     EUSART1_Write(angle_count);
342     LCD_printdata();
343     PORTCbits.RC0=0;
344     EUSART1_Write(inj_count);
345
346 }
347
348 void DelayFor18TCY(void){
349     Delay1TCYx(20); //delay de 20 ciclos
350     return;
351 }
352
353 void DelayPORXLCD(void){
354     Delay1KTCYx(60); //delay de 15ms
355     return;
356 }
357
358 void DelayXLCD(void){
359     Delay1KTCYx(20); //delay de 5ms
360     return;
361 }
```

APÊNDICE C – DIAGRAMA DE BLOCOS PARA A LÓGICA DE INJEÇÃO



Fonte: O autor.