DANIEL ISSAO ORUI EIRO

GUSTAVO COLLARES FACCINI TREVISAN

SIE: SISTEMA DE INJEÇÃO ELETRÔNICA

São Paulo
2010

DANIEL ISSAO ORUI EIRO

GUSTAVO COLLARES FACCINI TREVISAN

SIE: SISTEMA DE INJEÇÃO ELETRÔNICA

Monografia de Projeto de Conclusão de Curso

apresentada à Escola Politécnica da USP.

Disciplina: Projeto de Formatura II – PSI-2594

São Paulo

2010

DANIEL ISSAO ORUI EIRO

GUSTAVO COLLARES FACCINI TREVISAN

SIE: SISTEMA DE INJEÇÃO ELETRÔNICA

Monografia de Projeto de Conclusão de Curso

apresentada à Escola Politécnica da USP.

Disciplina: Projeto de Formatura II – PSI-2594

Área de Concentração: Sistemas Eletrônicos

Orientador: Prof. Dr. Armando Antônio Maria Laganá

São Paulo
2010

RESUMO

O presente projeto tem por objetivo o desenvolvimento de um sistema de injeção eletrônica de combustível, o sistema deve se comportar de maneira similar aos sistemas de injeção eletrônica já existentes, que são desenvolvidos de modo sigiloso e, portanto, não possibilitam alterações em seus códigos. Com base na obscuridade sob projetos consolidados, esse trabalho apresenta uma gama de conceitos teóricos que tendem a possibilitar uma reprodução confiável do sistema. O projeto apresenta certos conceitos importantes para um sistema de injeção eletrônica, como é o caso do sincronismo entre os sinais a serem controlados.

O projeto foi implementado utilizando-se a tecnologia dos microcontroladores PIC, de forma que este poderá ser considerado como um bom exemplo de aplicação prática de tal tecnologia, onde muitos conceitos foram utilizados, como os Timers, Interrupções e Comunicação Serial entre microcontroladores.

Palavras-chave: injeção eletrônica de combustível, sistemas microprocessados.

ABSTRACT

The objective of this Project is to develop an Electronic Fuel Injection System, this system should have a POC (*proof of concept*) similar to the real Electronic Fuel Injection Systems, which are developed in confidential way, and, consequently, people cannot access the Code (*Software*) of an original system. Due to the secret about the development of Injection Systems, this project shows lots of theoretical concepts that tend to become possible a reliable reproduction system. The project will present some very important concepts for an electronic injection system, such as the timing between the signals to be controlled.

The project will be developed using the technology of PIC microcontrollers, so it can be considered as a good example of practical application of such technology, where many concepts have been used, as timers, interrupts and serial communication between microcontrollers.

Keywords: electronic fuel injection, microprocessor system.

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

SIE	Sistema de Injeção Eletrônica
NTC	Negative Temperature Coefficient
MAP	Manifold Absolute Pressure
SPI	Serial Peripheral Interface
CAN	Controller Area Network
CCS	Custom Computer Services
ECU	Eletronic Control Unit

LISTA DE TABELAS

Tabela 1 - Características do microcontrolador PIC 16F87XA 25

Tabela 2 - Lista de Componentes 29

LISTA DE FIGURAS

Figura 1 - Ciclo de trabalho de um motor ciclo Otto – Ref [3] 17
Figura 2 - Temperatura x Resistência para um sensor NTC – Ref [9] 19
Figura 3 - Três tipos de sensores de temperatura NTC – Ref. [9] 20
Figura 4 - Pressão x Tensão para o sensor MAP – Ref. [9] 20
Figura 5 - Sensor MAP – Ref. [9] 21
Figura 6 - Sensor Hall e a roda-fônica – Ref. [11] 21
Figura 7 - Sinal medido pelo sensor Hall 22
Figura 8 - Sinal medido pelo senso Hall "quadrado" 22
Figura 9 - Sensores Efeito Hall – Ref. [9] 23
Figura 10 - Atuação da Sonda Lambda – Ref. [8] 23
Figura 11 - Sonda Lambda – Ref. [9] 24
Figura 12 - Método Pulsado ("Chopped") – Ref.[1] 26
Figura 13 - Bico Injetor – Ref. [12] 26
Figura 14 - Fluxograma 28
Figura 15 - Diagrama Lógico 30
Figura 16 - Kit da roda-fônica 33
Figura 17 - Sinal onda quadrada 34
Figura 18 - Kit LabTools: Microcontrolador 34
Figura 19 - Diagrama lógico bloco de sincronismo 35
Figura 20 - Diagrama lógico bloco de gerenciamento 36
Figura 21 - Tempos dos pulsos 37
Figura 22 – Cálculo da rotação do motor 38
Figura 23 - Análise da posição dos cilindros com o motor funcionando de acordo com o Ciclo Otto – Ref. [15] 39
Figura 24 - Período de injeção, segundo a estrutura dentada da rodafônica 40
Figura 25 – Método Prático versus Método Teórico 41
Figura 26 – Diagrama de ligações Mestre e Escravo – Ref. [14] 44
Figura 27 – Topologia do sinal da roda-fônica 47
Figura 28 – Circuito de Interface do LM1949 – Ref. [13] 48
Figura 29 – Formas de onda do LM1949 – Ref. [13] 48
Figura 30 – Identificação da falha para 1340 RPM 50
Figura 31 – Identificação da falha para 2876 RPM 50
Figura 32 – Identificação da falha para 4630 RPM 50
Figura 33 – Identificação da falha para 6230 RPM 51

Figura 34 – Interrupção de 5ms em escala ampliada (à esquerda) Duração de um volta (à direita) **52**
Figura 35 – Interrupção de 5ms em diferentes escalas **52**
Figura 36 – Pulso de Injeção de 2,25 ms para rotação de 3000 RPM **53**
Figura 37 – Foto do SIE implementado com a roda-fônica e os bicos injetores **54**

SUMÁRIO

1 – Introdução 11
2 – Estado da Arte do Projeto SIE 12
2.1 – Breve Contexto Histórico 12
2.2 - O Sistema de Injeção Eletrônica 13
2.3 – O Escopo do Projeto 14
2.4 – O Processo de Injeção 15
2.4.1 – Tipos de Mistura 15
2.4.2 – Ciclo de trabalho de um motor 17
2.4.3 – Cálculo do Tempo Básico de Injeção 18
2.5 – Sensores 19
2.5.1 – Sensor de Temperatura 19
2.5.2 – Sensor de Pressão 20
2.5.3 – Sensor de Rotação 21
2.5.4 – Sensor de Medição de Oxigênio no Escape 23
2.6 – Microcontrolador 24
2.7 – Atuador: Bico Injetor 25
2.8 – Metodologia 26
2.8.1 – Descrição do Funcionamento 26
2.8.2 – Diagrama Lógico 28
2.8.3 – Lista de Materiais e Orçamento 29
2.8.4 – Diagrama de Blocos do Projeto 30
2.8.5 – Resultados Esperados 31
2.9 – Requisitos do Projeto 31
2.10 – Viabilidade e Riscos 31
3 – Desenvolvimento Prático 33
3.1 - O Ambiente de Desenvolvimento 33
3.2 - Sincronismo e Gerenciamento 35
3.3 - Identificação da Falha 37
3.4 - Cálculo do valor de rotação 38
3.5 - A injeção segundo o ciclo Otto 39
3.6 - Tempo de injeção 41
3.7 - Protocolo de Comunicação SPI 42
3.7.1 - Funcionamento do Protocolo 43
3.8 - Timers e Interrupções 45

3.8.1 - Timer 0 45
3.8.2 - Timer 1 46
3.8.2 - Timer 2 46
3.8.3 - Interrupção Externa 46
3.9 - Módulo de Injeção 48
3.10 – Validação e Depurações 49
3.10.1 - Identificação da falha 49
3.10.2 - Tempo de referência para achar a rotação 52
3.10.3 - Dente de referência e tempo de injeção 53
3.11 - Dificuldades Encontradas 54
4 - Conclusão 56
5 – Sugestões para Trabalhos Futuros 57
6 – Referências Bibliográficas 58
Apêndice A – Código do Microcontrolador de Gerenciamento 60
Apêndice B – Código do Microcontrolador do Sincronismo 63

1 – Introdução

O projeto *SIE* - (Sistema de Injeção Eletrônica), como o próprio nome sugere, visa criar um sistema de injeção eletrônica de combustível em um motor automotivo. O projeto tem como um dos objetivos, implementar um sistema de injeção eletrônica utilizando uma tecnologia simples e acessível, e possivelmente aplicando-o em um "mock-up"[1] de um motor automotivo. A importância disto vem tanto do ponto de vista didático quanto do ponto de vista implementacional, visto que se todas as funcionalidades do carro forem dominadas, futuros alunos poderão analisar integralmente o funcionamento de um veículo podendo até, se necessário, desenvolver algum tipo de máquina que funcione baseada em motor de combustão. Paralelamente a isso, será possível comparar o sistema implementado com os sistemas reais em utilização, e se possível, buscar alguma otimização destes sistemas.

Levando em consideração que os sistemas de injeção eletrônica utilizados em veículos automotivos são implementados utilizando-se de uma tecnologia protegida, ou seja, inacessível aos usuários, um dos principais objetivos deste projeto visa implementar um sistema utilizando uma tecnologia totalmente aberta, logo, qualquer usuário poderá ter total acesso ao conteúdo do mesmo, e consequentemente terá todo o conhecimento sobre o funcionamento de tal sistema.

[1] *mock up*: é um modelo, nesse caso em tamanho real, que é usado para o estudo das funcionalidades do carro. É constituído por um motor e seus periféricos, tais como injeção eletrônica, válvula borboleta, válvula de escape, etc.

2 – Estado da Arte do Projeto SIE

2.1 – Breve Contexto Histórico

O primeiro sistema de injeção eletrônica de combustível foi implementado em 1967 pela indústria alemã Bosch. Trata-se do modelo D-Jetronic que perdurou até o ano de 1976. Esse foi o primeiro sistema que conseguiu aproveitar a mistura ar-combustível de maneira satisfatória, ou seja, houve um aumento de performance, redução do consumo de combustível e menor emissão de gases em relação aos antigos sistemas com carburadores. A unidade de controle desse sistema era formada por cerca de 300 elementos, sendo cerca de 30 transistores e 40 diodos.

Esse sistema foi evoluindo e nasceram muitos outros sistemas com o passar dos anos. Um bem interessante é o K-Jetronic, onde pela primeira vez houve um sistema de controle fechado por um sensor "lambda". Esse sistema ajudou muito para a diminuição da emissão de gases, já que a sonda lambda atua no processo de balanceamento da mistura, ou seja, avisa se há excesso de oxigênio no produto resultante da combustão. Outra evolução foi a criação do sistema L-Jetronic, que tinha um sensor para medir a massa de ar, o que tornou o sistema mais confiável em relação ao K-Jetronic. Porém o sistema com o sensor "lambda" substituiu esse tipo de sistema, como no sistema Mono-Jetronic, onde o sensor substitui sensores que medem fluxos e massas de ar. Para se ter um exemplo, esse último sistema mencionado foi criado em 1988 e usa um microcontrolador Intel 8051 que é infinitamente mais complexo que o controlador usado nos primórdios do sistema de injeção eletrônica. É importante salientar que essa evolução dos sistemas não é um movimento estritamente exato, isso quer dizer que sistemas mais antigos podem ser usados por mais tempo do que novos e vice-versa. Não houve uma padronização do sistema de injeção eletrônica, e sim diferentes implementações sob o prisma da engenharia.

2.2 - O Sistema de Injeção Eletrônica

Um sistema de injeção de combustível tem como função o controle da relação ar/combustível que é admitida nos cilindros a fim de que a mistura seja a mais conveniente às condições de funcionamento do motor. Para realizar este controle, o sistema precisará ter o conhecimento da massa de ar admitida nos cilindros, assim como o estado de funcionamento do motor. Tal estado pode ser conhecido através de alguns parâmetros do motor, tais como: rotação, temperatura, pressão. Com a posse destes dados, uma unidade de controle será capaz de calcular a quantidade de combustível a ser injetada nos cilindros.

Um sistema de injeção eletrônica completo é composto por:

1) Sensores: São os dispositivos que farão as medições físicas tais como temperatura, pressão, velocidade de rotação. A partir destas medições, sinais elétricos serão gerados e enviados à unidade de controle.

2) Atuadores: São os dispositivos de acionamento, uma vez que a unidade de controle precisa de algum meio de alterar o funcionamento do sistema controlado, os atuadores são os responsáveis por tal ação.

3) Unidade de controle: Responsável pelo processamento das informações recebidas pelos sensores, e a partir disso, controlar os elementos atuadores. O processamento dessas informações é implementado através de programas de controle, que realizam os cálculos necessários para o bom funcionamento do sistema a ser controlado.

Há três principais situações que definem um estado do motor, e três tipos de mistura ar/combustível:

1) Aceleração/Regime de plena carga: Nesta situação é preciso que o motor utilize uma mistura rica, que é um mistura cuja quantidade de ar é menor do que a mistura ideal (que será explicada a seguir), ou seja, nesta situação consome-se mais combustível.

2) Cargas Parciais/Marcha Lenta: Nesta situação o motor utiliza uma mistura ideal (ou estequiométrica), que é a mistura que possui a relação mais adequada ao bom funcionamento do motor, esta situação visa otimizar a combustão e o nível de emissões.

3) Desacelerações: Nesta situação o motor utiliza uma mistura pobre, que é uma mistura com menos combustível do que a mistura ideal, porém nesta situação a combustão não é tão eficiente e há um aumento no nível de emissões.

A unidade de controle visa controlar a quantidade de combustível a ser injetada de acordo com as situações citadas acima, e para controlar a quantidade de combustível a ser injetada a unidade de comando controla o tempo que o injetor de

combustível fica aberto, que por sua vez é controlado pela duração de um pulso elétrico.

2.3 – O Escopo do Projeto

Nos sistemas de injeção eletrônica presentes nos veículos automotivos, o controle da injeção é feita pela ECU (*Eletronic Control Unit*), e tal dispositivo é protegido, tendo em vista que os usuários não têm acesso à tecnologia presente no interior do mesmo, logo, ele é visto somente como sendo uma "caixa-preta", o que é totalmente compreensível, uma vez que uma montadora de automóveis deve proteger sua própria tecnologia.

Em vista disto, há um problema. Essa falta de acesso ao conteúdo da ECU leva a uma grande dificuldade no estudo das funções que esta exerce, por parte dos estudantes de engenharia, por exemplo. Dento desta situação que se encaixa o *SIE* (Sistema de Injeção Eletrônico), o projeto visa implementar um modelo bastante fidedigno, e razoavelmente simples, que visa simular o funcionamento de um sistema de injeção eletrônica em um veículo automotivo.

Levando em conta que os principais *stakeholders* serão alunos de engenharia, ou até mesmo alunos de ensino médio/técnico, o projeto possuirá uma restrição de ser razoavelmente simples, de forma que o entendimento de seu funcionamento não exija conhecimentos técnicos muito avançados. Outra grande restrição está no fato de que um sistema de injeção real possui várias variáveis a serem controladas, e utilizá-las em totalidade seria algo inviável em um projeto de formatura, em face disto, optamos em utilizar, a princípio, sensores vistos durante a disciplina *PSI2618 – Circuitos Eletrônicos Automotivos*, considerando a maior familiaridade com os mesmos, tais sensores serão descritos posteriormente nesta monografia. Esta redução de variáveis também contribuirá na simplificação do modelo. Outra restrição, desta vez relacionada a uma decisão de projeto, está na tecnologia a ser utilizada, que será um microcontrolador PIC16F877A, foi escolhido este microcontrolador devido ao fato dos alunos do presente projeto já terem um certo grau de familiaridade com tal tecnologia, e também pelo fato de que a programação em PIC é altamente acessível.

2.4 – O Processo de Injeção

O combustível utilizado pelo sistema estará armazenado em um reservatório. Uma bomba de combustível (elétrica) será encarregada de levar o combustível até a válvula injetora onde juntamente com o fluxo de ar, será formada a mistura que será injetada nos cilindros. É importante salientar que esta injeção é feita a uma pressão maior do que a pressão atmosférica, por isso, o sensor MAP (*Manifold Absolute Pressure*) terá um papel fundamental neste processo, pois como será explicado posteriormente, esse sensor que fará a medição da pressão do ar admitido.

Em um sistema de injeção eletrônica, a injeção de combustível é feita de forma intermitente. Geralmente, o bico injetor é acionado por curtos intervalos de tempo, portanto, é de extrema importância que a ECU esteja sincronizada com os ciclos de ignição dos cilindros. Basicamente, o que a ECU deverá fazer neste caso é:

1) Medir a massa de ar admitida nos cilindros.
2) Calcular a quantidade de combustível necessária para se obter a relação ar/combustível desejada para a mistura, além do tempo de injeção. A quantidade de combustível calculada leva em conta a massa de ar admitida e a relação ar/combustível, e o tempo leva em conta a quantidade de combustível calculada e um fator de calibração do injetor (dado pelo fabricante do mesmo).
3) Enviar o sinal para o injetor durante um tempo igual ao calculado, dosando assim, a quantidade de combustível desejada.

2.4.1 – Tipos de Mistura

Em um processo de Injeção de combustível temos três principais tipos de mistura que serão utilizadas. Esses três tipos serão um pouco mais detalhados a seguir:

1) Mistura Estequiométrica (ou Ideal): É o tipo de mistura que possui a relação ar/combustível mais adequada ao funcionamento do motor. É a mistura que possibilita a combustão total do combustível. Através de um balanceamento da reação de combustão é possível obter um valor aproximado da massa de ar admitida para se obter essa mistura, este valor é conhecido por uma relação estequiométrica.

2) Mistura Rica: É o tipo de mistura que possui menor ar do que o correspondente à mistura ideal, logo, há um excesso de combustível, que não será consumido pela reação, havendo um aumento na emissão de poluentes, e também um maior desperdício do combustível.

3) Mistura Pobre: É o tipo de mistura que possui menos combustível do que o correspondente à mistura ideal. Nessa mistura, a combustão também não será totalmente eficiente, além de ocorrer um aumento da temperatura, devido ao excesso de oxigênio, o que pode causar um mal funcionamento do sistema.

Utilizando como combustível a gasolina comum, que nada mais é do que uma mistura de hidrocarbonetos, o componente iso-octano (C_8H_{18}) será utilizado na reação química a seguir, devidamente balanceada:

$$C_8H_{18} + 12{,}5 * O_2 + 45{,}31 * N_2 \rightarrow 8 * CO_2 + 9 * H_2O + 45{,}31 * N_2$$

Obs.: A seguinte proporção de massa foi considerada para o ar:
78% N_2 (45,31 kg) e 21% O_2 (12,5 kg)

Logo, a massa de ar admitida de ar ($O_2 + N_2$) no caso será: 12,5 * 32 + 45,31*28 = 1.668,86 kg, e a massa de combustível: 8*12 + 18 = 114 kg

Assim sendo, a relação ar / combustível ideal obtida será de: 1.668,86 : 114 $\Longrightarrow$ 14,64 : 1

Há um índice muito difundido na literatura sobre motores de combustão interna, conhecido por *Fator Lambda*, que é definido por:

$$Fator\ Lambda = \frac{relação\ ar/combustível\ real}{relação\ ar/combustível\ ideal}$$

Esse fator poderá ser muito útil, porventura, no desenvolvimento do código a ser programado no microcontrolador, uma vez que a partir dele será possível determinar:

- Se Fator Lambda = 1 $\Longrightarrow$ Mistura Estequiométrica ou Ideal
- Se Fator Lambda > 1 $\Longrightarrow$ Mistura Pobre
- Se Fator Lambda < 1 $\Longrightarrow$ Mistura Rica

O nome *Fator Lambda* vem do fato que a relação ar/combustível real será obtida através dos valores medidos pelo sensor *Sonda Lambda*, que será explicado posteriormente nesse relatório na parte específica de sensores (item 2.5).

2.4.2 – Ciclo de trabalho de um motor

Para entendermos corretamente o que foi chamado de *Ciclo Correto* de injeção de combustível, vamos explicar, sucintamente, o Ciclo Otto. Esse ciclo se divide em quatro partes, como mostrado na figura a seguir:

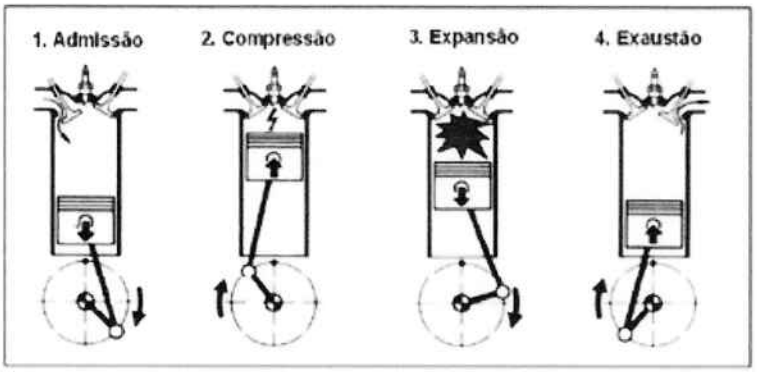


Figura 1 - Ciclo de trabalho de um motor ciclo Otto – Ref [3]

Na primeira fase há a admissão da mistura ar–combustível, realizada com a abertura da válvula de admissão seguida da injeção de combustível e entrada de ar. Em seguida essa mistura é comprimida dentro do cilindro através de um pistão. Com o combustível comprimido, é realizada a "explosão" do mesmo através de uma faísca. Então, há a expansão do pistão com as válvulas ainda fechadas. É nesse momento que a energia do combustível é liberada, sendo transformada em movimento. Por fim, a válvula de exaustão é aberta a fim de promover a exaustão dos gases gerados pela explosão.

Esse ciclo perdura por duas voltas completas do rotor e mostra por que devemos saber bem quando é o momento certo para a injeção de combustível. Esse momento é controlado pela informação proveniente do sensor de efeito Hall, que nos dá, exatamente, a rotação (sinal da roda fônica).

2.4.3 – Cálculo do Tempo Básico de Injeção

Para calcularmos o tempo básico de injeção de combustível, devemos primeiramente calcular a massa de combustível necessária. Esse cálculo é baseado na relação estequiométrica entre combustível e ar, que já foi mostrada anteriormente. Ou seja, precisamos agora calcular a massa de ar admitida, e então usarmos a relação estequiométrica para ver a massa de combustível. Podemos falar que a massa de ar é o produto da densidade pelo volume de ar, ajustado à rotação do motor. Da equação fundamental dos gases perfeitos tiramos a relação da densidade com a pressão e temperatura, que é dado por:

$$Densidade = \frac{Pressão}{R * Temperatura}$$

Onde: R = constante universal dos gases perfeitos (R=8,314472 J · K^{-1} · mol^{-1})

Agora o volume de ar é o volume do cilindro (cilindrada) multiplicado por um fator tabelado, pois nem todo o cilindro é preenchido (η).

Por fim, fazemos o ajuste de acordo com a rotação do rotor, o que nada mais é que fazê-lo de acordo com o Ciclo Otto, que diz que é necessário fazer a injeção a cada duas voltas da roda-fônica. Então, a fórmula final fica:

Speed-Density:

$$Massa\ de\ ar = \eta * \frac{Pressão}{R * Temperatura} * Cilindrada * \frac{RPM}{2 * 60}$$

Portanto vemos que conseguimos obter a massa de ar a partir apenas do sensor MAP de pressão, do sensor de temperatura e da rotação, a partir do sensor de efeito Hall.

Com a massa de ar calculada, devemos agora usar a relação estequiométrica para obtermos a massa de combustível.

O tempo de injeção é calculado a partir da massa de combustível, da densidade do combustível e da característica do injetor. O injetor nos fornece um fator de calibração usualmente em cm^3/min. Logo basta apenas dividirmos o volume de combustível, isto é, a sua massa pela sua densidade, por esse fator de calibração para chegarmos no tempo que o sistema deve permanecer injetando.

2.5 – Sensores

Como já dito anteriormente, foram feitas algumas restrições com relação ao número de sensores que serão utilizados no projeto SIE. Porém, essa redução foi feita de forma a não afetar consideravelmente a funcionalidade do sistema. Os sensores serão: sensor de temperatura, sensor de rotação do motor, sensor de pressão e sensor de medição de oxigênio no escape.

2.5.1 – Sensor de Temperatura

Será utilizado um sensor do tipo NTC (Negative Temperature Coefficient), este sensor trabalha com uma relação praticamente linear (inversamente proporcional) resistência *versus* temperatura, e que possui uma faixa de medição de -40 °C a +130 °C. Na figura a seguir está ilustrada esta relação linear, figura retirada de um catálogo de Sensores da *Bosch*.

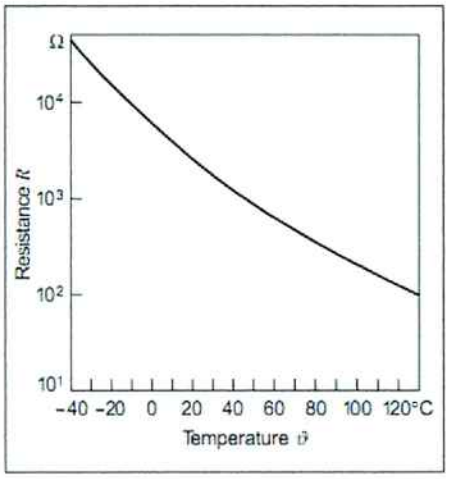


Figura 2 - Temperatura x Resistência para um sensor NTC – Ref [9]

Porém, para ser possível tornar mais claras as verificações, pelo microcontrolador, dos valores a serem medidos neste sensor faz-se necessário transformar essa relação resistência *versus* temperatura para uma relação tensão *versus* temperatura. Para contornar esse problema será utilizado um artifício tecnicamente simples, aprendido durante os cursos de eletrônica da Escola. O artifício a ser utilizado será: alimentar um circuito contendo o sensor e um resistor em série, este resistor será de valor conhecido (Rref), assim como a tensão de alimentação (Vref) e o valor devolvido pelo sensor (Rsen). Assim, aplicando um divisor de tensão temos o seguinte valor de tensão sobre o sensor:

$$Tensão\ no\ sensor\ (Vsensor) = \frac{Rsensor}{Rsensor\ + Rref} * Vref$$

De posse deste valor de tensão, este será ligado a uma das portas de entrada do microcontrolador, e a partir dos valores medidos o programa ficará encarregado

de determinar o tempo de injeção de acordo com o descrito na Metodologia (seção 2.8). A seguir se encontra um foto contendo três tipos deste sensor, retirada do catálogo da *Bosch*:

Figura 3 - Três tipos de sensores de temperatura NTC – Ref. [9]

2.5.2 – Sensor de Pressão

O sensor a ser utilizado será um do tipo *MAP* (*Manifold Absolute Pressure*), este sensor medirá a pressão no coletor de admissão, ou seja, será uma medida da pressão do ar admitido no motor. Essa medida de pressão servirá de auxílio no cálculo da massa de ar admitida (seção 2.4.3). Este sensor possui uma relação linear de tensão x temperatura, podendo medir pressões de até 120 kPa. Na figura a seguir está ilustrada a curva obtida na saída deste sensor, retirada do catálogo da *Bosch*.

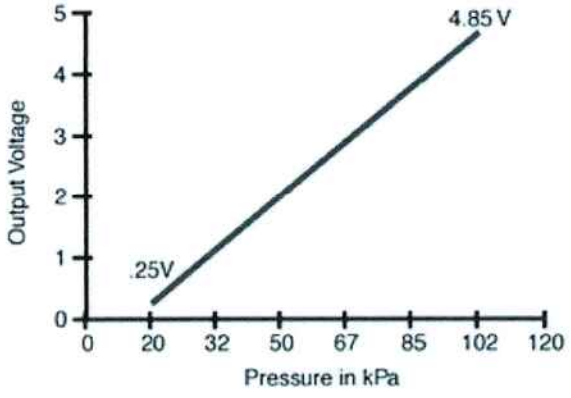


Figura 4 - Pressão x Tensão para o sensor MAP – Ref. [9]

Como a saída deste sensor já é uma tensão que está de acordo com as especificações de tensão de entrada do microcontrolador, não há a necessidade de ser inserido nenhum circuito de conversão de valores, e a princípio a saída deste sensor poderá ser ligada diretamente em uma das portas do microcontrolador.

O sensor *MAP* atuará no motor "*mock-up*" da seguinte maneira: Quando o motor estiver operando em marcha lenta, a pressão no coletor de admissão será baixa, e quando o motor estiver operando em regime de aceleração (plena carga) a

pressão no coletor de admissão será alta. O regime de desaceleração será detectado quando houver uma mudança brusca de pressão de alta para baixa. A seguir se encontra uma foto deste sensor, retirada do catálogo da *Bosch*.

Figura 5 - Sensor MAP – Ref. [9]

2.5.3 – Sensor de Rotação

Este sensor que será utilizado para medir a velocidade de rotação do motor, tem como princípio o Efeito Hall, que verifica a variação do campo magnético causado pela rotação da roda-fônica e na saída mostra um valor de tensão condizente com este campo. O sensor *Hall* atuará na roda-fônica da seguinte forma:

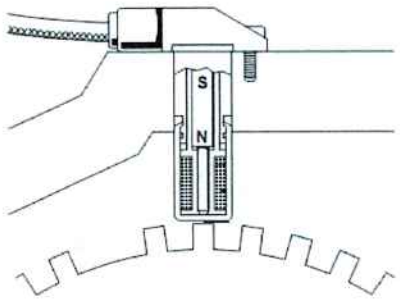


Figura 6 - Sensor Hall e a roda-fônica – Ref. [11]

Este sensor detectará a presença (ou ausência) dos *dentes* da roda fônica, uma vez que a relutância magnética destes são diferentes da relutância do ar. Diferentemente dos outros sensores descritos neste relatório, este não possui uma relação linear de tensão x rotação, logo, para que seja possível ter conhecimento do valor de rotação da roda-fônica será preciso ser feito um processamento do sinal medido, tal processamento será realizado pelo programa contido no microcontrolador. Com a roda-fônica em funcionamento, o sinal medido pelo sensor terá uma forma semelhante com a forma de onda a seguir:

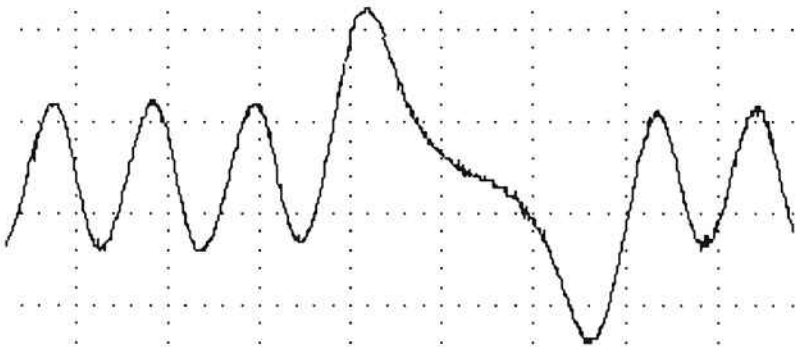


Figura 7 - Sinal medido pelo sensor Hall

Entretanto, utilizar um sinal senoidal como entrada do microcontrolador não é aconselhável, logo é preciso ser feito um pequeno ajuste deste sinal, "quadrando-o" através de um circuito. Portanto, o sinal de entrada ficaria:

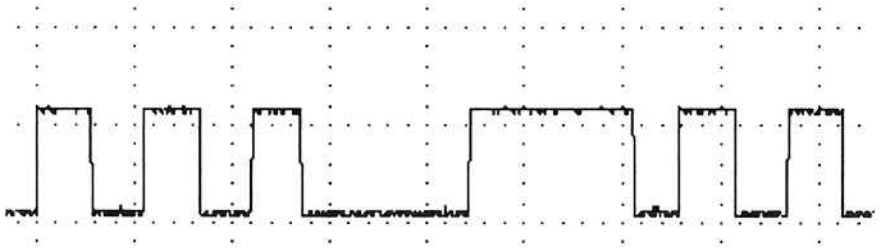


Figura 8 - Sinal medido pelo senso Hall "quadrado"

Na figura 6, onde é mostrada a atuação do sensor *Hall* na roda-fônica, percebe-se que a mesma possui uma "falha", ou seja, há um dente faltando, nos sinais medidos isso é evidenciado através de um ciclo com maior duração (largura), essa informação é de extrema importância, pois é através desta falha que é possível que seja calculado o valor de rotação da roda-fônica, uma vez que essa falha determinará uma rotação completa, e também será de extrema importância para a realização do sincronismo do processo de injeção. Basicamente, o que o programa do microcontrolador faz: Após ser detectada a falha, terá início uma contagem do número de pulsos em um determinado intervalo de tempo, a partir disso será possível ter conhecimento do valor de rotação, uma vez que quanto maior for o número de pulsos contados, maior será a rotação e vice-versa.

Assim como foi indicado nos outros sensores, a seguir se encontra uma foto ilustrando esse sensor, retirada do catálogo da *Bosch*.

Figura 9 - Sensores Efeito Hall – Ref. [9]

2.5.4 – Sensor de Medição de Oxigênio no Escape

A medição da quantidade de oxigênio exaurida pelo motor no escape será feita pela *Sonda Lambda*. Esse sensor mantém em contato com o ar ambiente e compara o nível de oxigênio entre essa referência e o encontrado no escapamento do veículo. Sua funcionalidade é avisar à Unidade de Controle se há combustível sobrando na reação química, ou seja, se é necessário prolongar mais o tempo da injeção. O principal cuidado que devemos ter com esse sensor é que ele só opera em temperaturas superiores a 300°C, sendo necessário, portanto, se verificar essa temperatura antes de usarmos a informação contida no sensor. O esquema do sensor acoplado ao motor é apresentado abaixo. No esquema fica claro que o sensor atua fechando a malha do circuito de injeção, ou seja, faz o ajuste da injeção de combustível a partir da quantidade de oxigênio presente no gás escape.

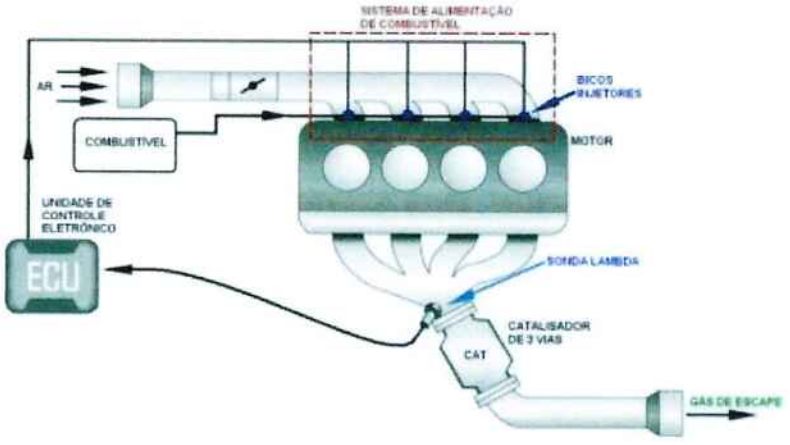


Figura 10 - Atuação da Sonda Lambda – Ref. [8]

E a seguir, uma ilustração da Sonda Lambda, retirada do catálogo da Bosch:

Figura 11 - Sonda Lambda – Ref. [9]

2.6 – Microcontrolador

No Sistema de Injeção Eletrônica, o microcontrolador desempenhará o papel da Unidade de Controle, ou seja, ele será o grande responsável por receber todos os sinais medidos pelos sensores, e a partir disso determinar o quanto de combustível deverá ser injetado pelo injetor, a forma como isso será feito será explicada no item Metodologia (seção 2.8), esta presente seção visa dar uma visão geral do componente microcontrolador em si.

O *SIE* possuirá quatro variáveis a serem medidas: temperatura, pressão, rotação da roda-fônica e quantidade de oxigênio no escape, que serão medidas pelos respectivos sensores: *NTC*, *MAP*, *Efeito Hall* e *Sonda Lambda*, e possuirá uma variável a ser controlada: quantidade de combustível injetado, que será feito através do bico injetor. Em outras palavras, o microcontrolador terá: quatro parâmetros de entrada e um de saída. Logo, é necessário que o microcontrolador possua, no mínimo, cinco portas do tipo I/O (Entrada/Saída). Da família do microcontrolador PIC 16F87XA, há um que atende a essas especificações, o PIC 16F877A. Abaixo se encontra uma tabela retirada do *datasheet* deste microcontrolador.

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

Tabela 1 - Características do microcontrolador PIC 16F87XA – Ref. [7]

Uma característica mostrada na tabela acima que é relevante para o sistema é a presença dos *Timers*. Uma vez que estes *Timers* possuem contadores, que terão um importante papel no cálculo rotação da roda-fônica, sendo que o programa realiza uma contagem (no tempo) de pulsos do sinal da roda-fônica, e a partir disso determina a rotação. Nota-se que o PIC16F877A possui três *Timers*, sendo que um deles, o *TIMER0 MODULE* já atende as especificações do sistema, pois este módulo possui um *Counter Mode* que contará cada borda de subida (ou descida) de um sinal ligado à porta uma porta específica (RA4/T0CKI), assim, o sinal da roda-fônica pode ser ligado nessa porta também, e assim, ser feita a contagem.

2.7 – Atuador: Bico Injetor

Todo processamento do *SIE* e os sensores se resumem na alimentação do injetor de combustível, para assim ser concretizada a injeção de combustível. Para o acionamento desse atuador devemos nos ater a dois pontos: princípio básico de funcionamento e tempo de resposta do injetor. Por esse componente se tratar basicamente de um solenóide, não devemos manter uma tensão constante sob ele durante todo o tempo de funcionamento, pois poderíamos esquentá-lo demais e, portanto, danificá-lo. Por isso, fazemos o seu acionamento através de pulsos, a fim de minimizarmos o aquecimento excessivo.

A principio nós utilizaremos pulsos com *Duty-Cycle* igual a 50%, podendo, ou não, alterá-lo de acordo com resultados experimentais. Outro ponto é a respeito do tempo de resposta do atuador, ou seja, quanto tempo após acionado que o dispositivo atinge um fluxo constante de saída líquido. Baseado na teoria a respeito desse atuador foi visto que quanto maior, em módulo, é a tensão aplicada (em relação a referência) à sua entrada, mais rápida será a resposta do dispositivo. Portanto, no início do acionamento, devemos aplicar um pulso com uma tensão igual a zero (o dispositivo está normalmente em '1', quando não conduz). Após isso,

podemos, então, jogar os pulsos com uma amplitude menor para apenas mantermos a abertura do atuador. O esboço da entrada do dispositivo é apresentado abaixo:

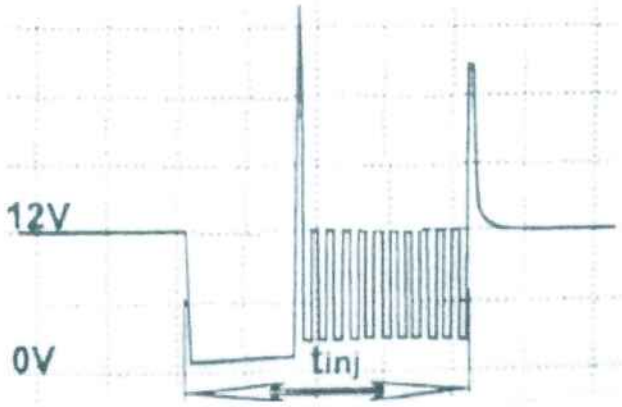


Figura 12 - Método Pulsado ("Chopped") – Ref.[1]

A seguir, uma ilustração de um Injetor, retirada do catálogo da *Delphi*:

Figura 13 - Bico Injetor – Ref. [12]

2.8 – Metodologia

2.8.1 – Descrição do Funcionamento

Através do diagrama lógico (Figura 15, a seguir), apresentaremos de uma forma didática o funcionamento do sistema. Se olharmos a realidade, temos que nos atentar, sobretudo, ao fato não ser possível a realização do sistema através de estruturas "SIM" E "NÃO". São necessários, então, ajustes em cima das características vistas no diagrama lógico. Por exemplo, não podemos considerar apenas se o motor está frio ou se não está: devemos saber quão frio ele está para assim tomarmos a devida decisão quanto à alteração do tempo de injeção. Além disso, outros fatos importantes que foram ocultados no diagrama lógico serão apresentados a seguir.

O primeiro passo ao se iniciar o processo de injeção é a medição da massa de ar que for admitida no motor. Não será dado muito enfoque na origem dessa massa de ar, e como ela foi admitida, já que o foco é mais na injeção de combustível em si. A medição da massa de ar será feita por um método indireto: a partir da pressão, temperatura, volume e densidade. Esse cálculo é relativamente simples, uma vez que o volume de ar é dado por uma fração do volume do cilindro, pois nem todo o cilindro é preenchido com ar. Já a densidade é uma grandeza calculada em função da pressão do coletor e da temperatura do ar, o que pode ser muito bem tabelado no processador para apenas ser consultado durante o cálculo da massa de ar admitida. Antes de seguir com o processo deve-se verificar se há a necessidade de injetar o combustível. Caso o veículo esteja desacelerando (freio motor), não há, a principio, a necessidade de se injetar combustível. O que o diagrama lógico não mostra é que não se pode cessar demasiadamente o fornecimento de combustível, devido ao esfriamento da câmera de combustão. O que se faz na verdade é cessar o fornecimento de combustível até que haja aceleração ou até que o sensor de temperatura aponte ser necessária a injeção de combustível por alguns ciclos.

Considerando agora, a necessidade de se injetar combustível, a unidade de controle realizará, então, o cálculo dessa quantidade, que é traduzida para o sistema como o tempo que o bico injetor deve permanecer injetando combustível. Primeiramente a massa de combustível é calculada a partir da massa de ar, através da relação estequiométrica combustível - ar. Ou seja, há uma quantidade certa de combustível, dada uma quantidade de ar, para que haja o equilíbrio químico da reação. Então, a partir dessa massa de combustível calculada, teremos o tempo básico para se manter o injetor aberto, através de um parâmetro chamado *calibração*, que nos dá a relação de quanto volume é injetado em certo tempo - a unidade usual é cm^3/min. Esse tempo foi chamado de tempo básico, pois ele apenas nos dá uma referência a respeito do tempo de injeção, podendo ser mudado ao longo do processo. Dentro os fatores que necessitam alterar o tempo de injeção estão: Temperatura do Motor, Aceleração, Velocidade e se o evento se trata de uma partida do motor. Para a mudança dos tempos de acordo com esses parâmetros, serão utilizados pesos referentes a essas modalidades, ou seja, será feita uma modelagem a fim de ser obtido um melhor fator de mudança do tempo requerido, lembrando que o sinal de atuação do sistema original de injeção eletrônica vinculado ao motor do "*mock-up*" poderá ser tido como uma referência. Será realizada a injeção assim que alcançado o ciclo certo, o que é referido a partir do sinal da roda-fônica. Por fim, será visto se todo o combustível injetado fora aproveitado, através do sensor lambda, podendo, se necessário, haver um ajuste para aumentar o tempo de injeção.

Alguns conceitos apresentados aqui estão detalhados ao longo da monografia, uma vez que aqui foi apresentada apenas uma visão geral da metodologia, baseada na unidade de controle, sensores e atuadores

2.8.2 – Diagrama Lógico

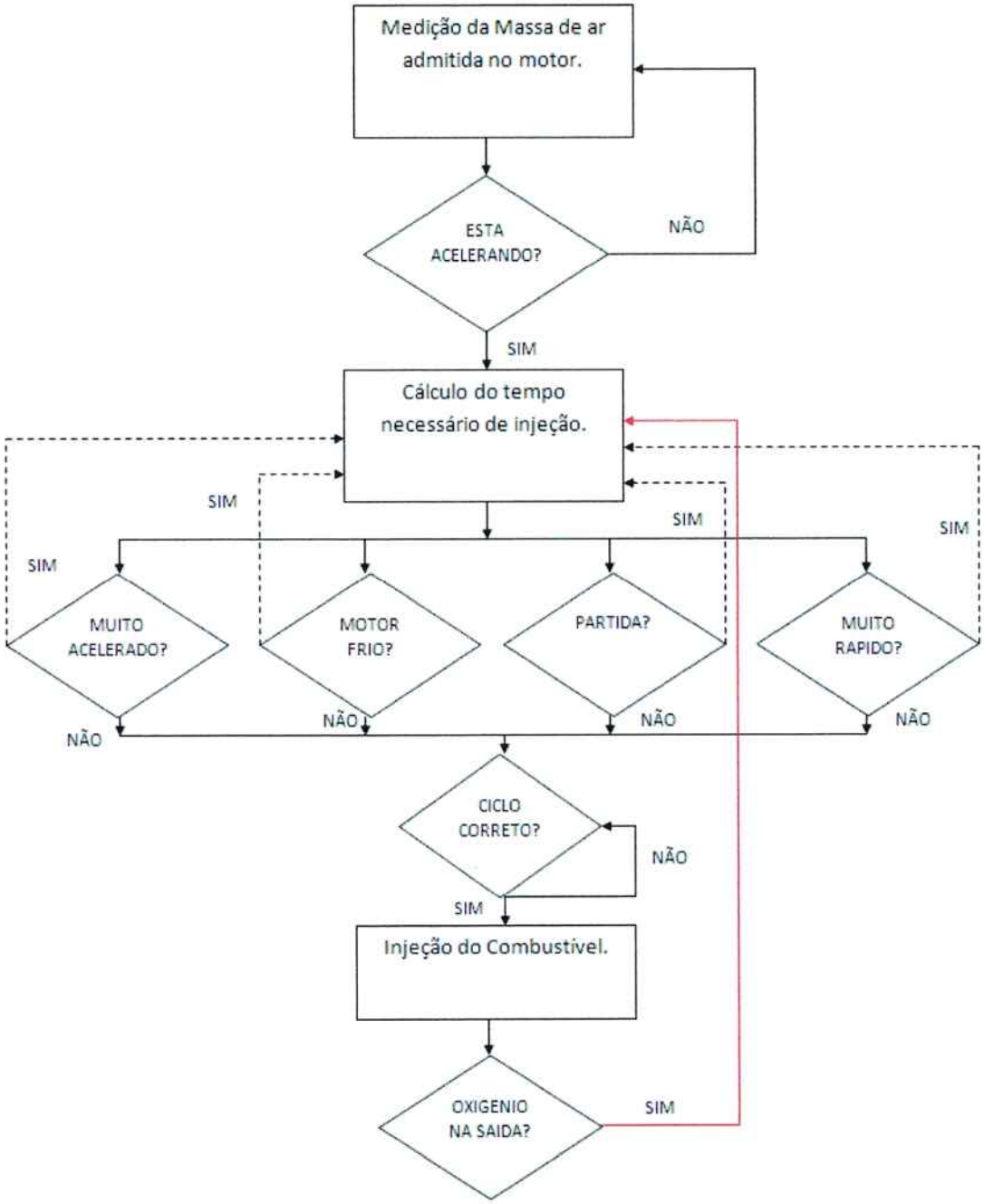


Figura 14 - Fluxograma

2.8.3 – Lista de Materiais e Orçamento

A seguir encontra-se uma tabela contendo a lista de materiais a serem usados no projeto, juntamente com os custos dos mesmos.

Componente	Quantidade	Valor Unitário	Valor Total
Kit CAN - LabTools	1	R$ 1.185,00	R$ 1.185,00
Sensor de Temperatura NTC	1	R$ 50,00	R$ 50,00
Sensor de Pressão MAP	1	R$ 90,00	R$ 90,00
Sonda Lambda	1	R$ 100,00	R$ 100,00
Sensor de Rotação (Efeito Hall)	1	R$ 220,00	R$ 220,00
Bico Injetor	1	R$ 160,00	R$ 160,00
Motor "*mock-up*"	1	Não se aplica	Não se aplica
TOTAL	R$ 1.805, 00		

Tabela 2 - Lista de Componentes

O motor "*mock-up*" é um componente que não fará parte do sistema propriamente dito. Ele será apenas um adicional ao projeto, pois será nele que poderão feitas medições dos parâmetros, e também, onde poderá ser feita a aplicação da injeção em si. O valor do mesmo não se aplica neste caso, pois este a encargo do professor orientador, sendo que "mock-up" pertence à *FATEC* – Santo André.

Os preços dos sensores e do bico injetores são valores aproximados, há a possibilidade de eles serem um pouco mais caros ou mais baratos, este presente orçamento tem a utilidade de mostrar uma visão geral da ordem de grandeza de quanto foi gasto neste projeto, considerando que os componentes foram disponibilizados pelo professor orientador, em caráter de empréstimo, minimizando consideravelmente os custos do desenvolvimento do projeto.

2.8.4 – Diagrama de Blocos do Projeto

A seguir, encontra-se um Diagrama de Blocos contendo as atividades realizadas no Projeto.

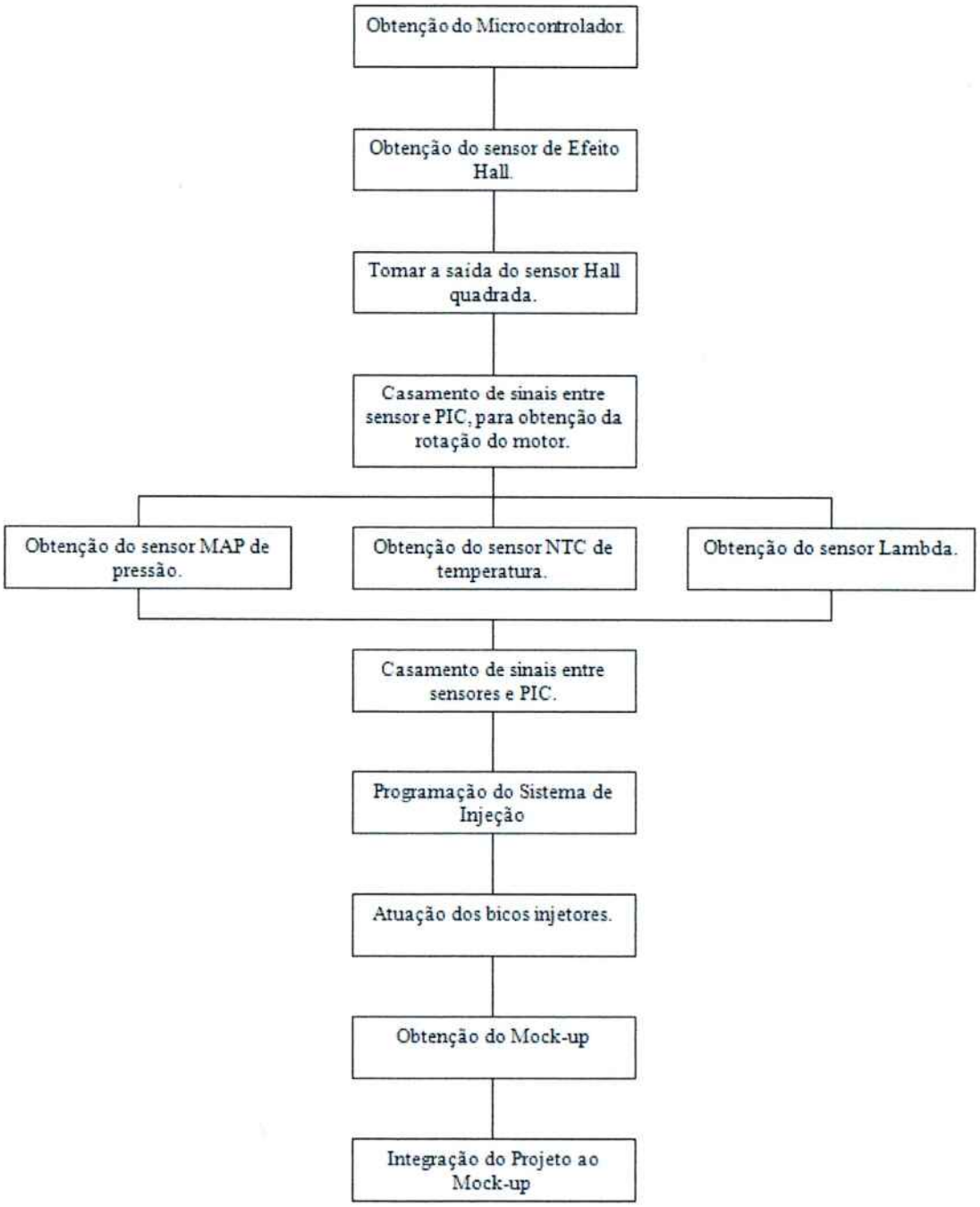


Figura 15 - Diagrama Lógico

2.8.5 – Resultados Esperados

Após a implementação do projeto espera-se ter uma solução simples, mas ao mesmo tempo robusta, de um sistema de injeção eletrônica. Os resultados apresentáveis serão os pulsos gerados pelo microcontrolador, que por sua vez acionam os bicos injetores, e podem ser vistos em um osciloscópio digital. Em um segundo momento, o sistema poderá ser integrado ao "*mock-up*" para ser feita uma comparação. Essa comparação poderá ser vista tanto sob o espectro da análise de desempenho quanto por meio de uma análise de sinais.

2.9 – Requisitos do Projeto

- A mínima freqüência de operação do microcontrolador deve ser de 6kHz.

Justificativa: Considerando-se que 6000RPM seja o máximo valor de rotação do motor a ser utilizado, vemos que esse valor equivale a um período de 100 Hz, considerando os 60 dentes, e que cada instrução deverá identificar um dente, temos 60*100 = 6 kHz, em teste. Logo, no mínimo, o microcontrolador deverá satisfazer essa freqüência, porém, como há mais instruções presentes no código, este valor, na prática, é um pouco maior, porém difícil de mensurar em valores precisos, em ordem de grandeza podemos dizer algo em torno de 10 a 15 kHz. No caso do SIE, o sistema opera com um oscilador de 20 MHz.

- A temperatura interna do nosso sistema de controle não pode passar os 125°C.

Justificativa: A temperatura de um microcontrolador da família PIC16F877A não pode ultrapassar os 125°C, segundo seu datasheet.

- O sistema deve ser alimentado com, no máximo, 12 Volts.

Justificativa: Por se tratar de um projeto voltado à eletrônica automotiva, o mesmo deverá ser alimentado com uma bateria de automóvel de 12 Volts.

2.10 – Viabilidade e Riscos

Para ser realizado um estudo de viabilidade do projeto *SIE*, será preciso listar alguns parâmetros que possam definir se o projeto é viável ou não. Tais parâmetros são:

Viabilidade Técnica / Recursos:

O projeto é extremamente viável, já que necessita, basicamente, apenas de um motor, alguns sensores, microcontroladores e os bicos injetores. Além disso, temos o amparo do mercado para confirmar ainda mais a viabilidade do projeto. Obviamente que o fato de haver sistemas de injeção eletrônica comercializáveis não

garante a finalização do projeto, garantindo apenas uma referência para o desenvolvimento do mesmo.

O maior risco do projeto é a não obtenção de um motor a combustão "mock-up". Porém, um motor elétrico está à disposição, que apresenta sinais semelhantes ao de um motor a combustão. Portanto, no pior dos casos teremos um sistema de injeção eletrônica moldado para o motor elétrico, e utilizando bicos injetores, que ao invés de injetar o combustível nos cilindros do motor, fará a injeção em compartimentos, como garrafas plásticas adaptadas.

Viabilidade de Tempo:

O tempo disposto para realizar o projeto pode ser considerado pequeno, levando em consideração que os alunos devem conciliar o tempo gasto no projeto com o tempo gasto em outras atividades, tais como aulas, provas e estágio, por exemplo. Logo, devido ao fato de não ser possível dedicar 100% do tempo ao projeto, a falta de tempo pode ser considerada um agravante para a execução do mesmo.

3 – Desenvolvimento Prático

3.1 - O Ambiente de Desenvolvimento

Praticamente todo o desenvolvimento do projeto foi feito dentro da própria Escola Politécnica, em uma sala no laboratório LSI (Laboratório de Sistemas Integráveis). Nesta sala, um kit da roda-fônica esteve à disposição, que encontra-se na figura abaixo:

Figura 16 - Kit da roda-fônica

Neste kit temos:

- Uma fonte de 5V que alimenta o circuito quadrador do sinal da roda-fônica.
- Um circuito quadrador, este circuito transforma o sinal proveniente da roda-fônica, que é senoidal, em um sinal em forma de onda quadrada, já pronto para ser utilizado no microcontrolador PIC. Este circuito foi montado por alunos da *FATEC,* e nos foi apresentado por Felipe Albaladejo, aluno de mestrado do nosso orientador.
- Um motor elétrico, onde será simulada e medida a rotação.
- Amplificador da rotação do motor elétrico, que amplifica a rotação do motor até 6.000 RPM
- Sensor de rotação, ou Sensor de Efeito Hall
- Osciloscópio digital para serem feitas as medições

Com este kit em mãos pode-se simular fielmente o mesmo sinal da roda-fônica que seria obtido no "*mock-up*". Como já explicado anteriormente, este sensor não medirá a rotação da roda-fônica propriamente dita, ele irá "transformar" a rotação em um sinal senoidal, que por sua vez, será quadrado e conectado em uma porta de entrada do microcontrolador, este sinal encontra-se na figura abaixo:

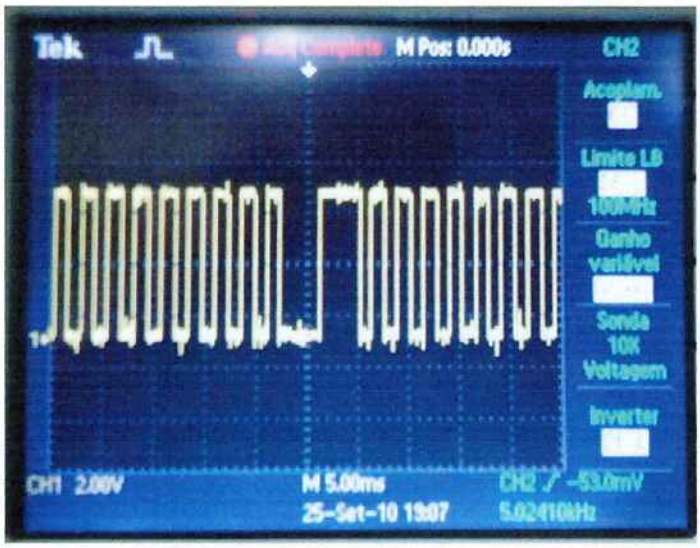


Figura 17 - Sinal onda quadrada

Utilizando o conceito da falha nos dentes da roda-fônica, é possível termos uma noção do valor da rotação da roda-fônica, uma vez que quanto maior for a rotação, maior será a freqüência do sinal adquirido. Basicamente, o que o microcontrolador deverá fazer é: detectar esta falha, e a partir disto, fazer o cálculo do valor aproximado da rotação, nos tópicos subseqüentes será detalhado o modo como foi implementado a detecção da falha, bem como o cálculo da rotação.

Em termos de trabalho com o microcontrolador foi utilizado um KIT CAN da *LabTools*, disponibilizado pelo orientador, este kit possui: três microcontroladores (PIC16F877A), três controladores CAN (MCP2515) e três transceptores CAN (PCA82C251), abaixo encontra-se uma foto deste kit:

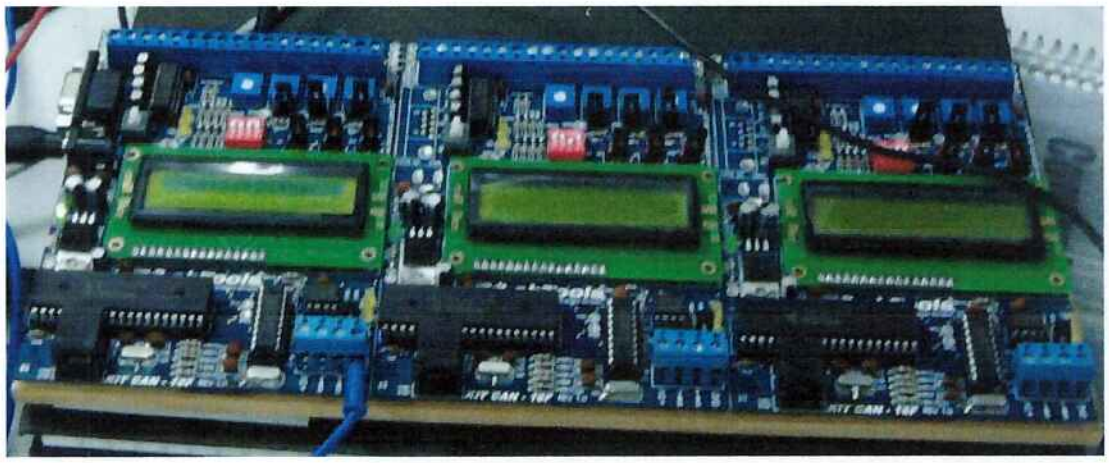


Figura 18 - Kit LabTools: Microcontrolador

Para a programação dos código, foi utilizado o *MPLAB*, software desenvolvido pela própria *Microchip*, fabricante dos microcontroladores PIC, além de serem desenvolvidos os códigos, o *MPLAB foi usado, também,* para fazer a gravação dos mesmos no microcontrolador, utilizando o kit CAN da *LabTools*, que já possui a ligação USB pronta para este fim. O compilador utilizado é o *CCS* (*Custom Computer* Services), presente no pacote *MPLAB*.

3.2 - Sincronismo e Gerenciamento

Para o processamento e implementação do sistema, foram utilizados dois microcontroladores. O primeiro, que é denominado de 'Gerenciamento' tem a função de capturar as informações necessárias para a injeção, enquanto que o segundo, denominado de 'Sincronismo' se utiliza dessas informações para executar a injeção. Ambos os processadores têm consigo o sinal da roda-fônica, que determina o sincronismo entre a troca de informações.

O diagrama lógico para o Bloco de Sincronismo encontra-se a seguir:

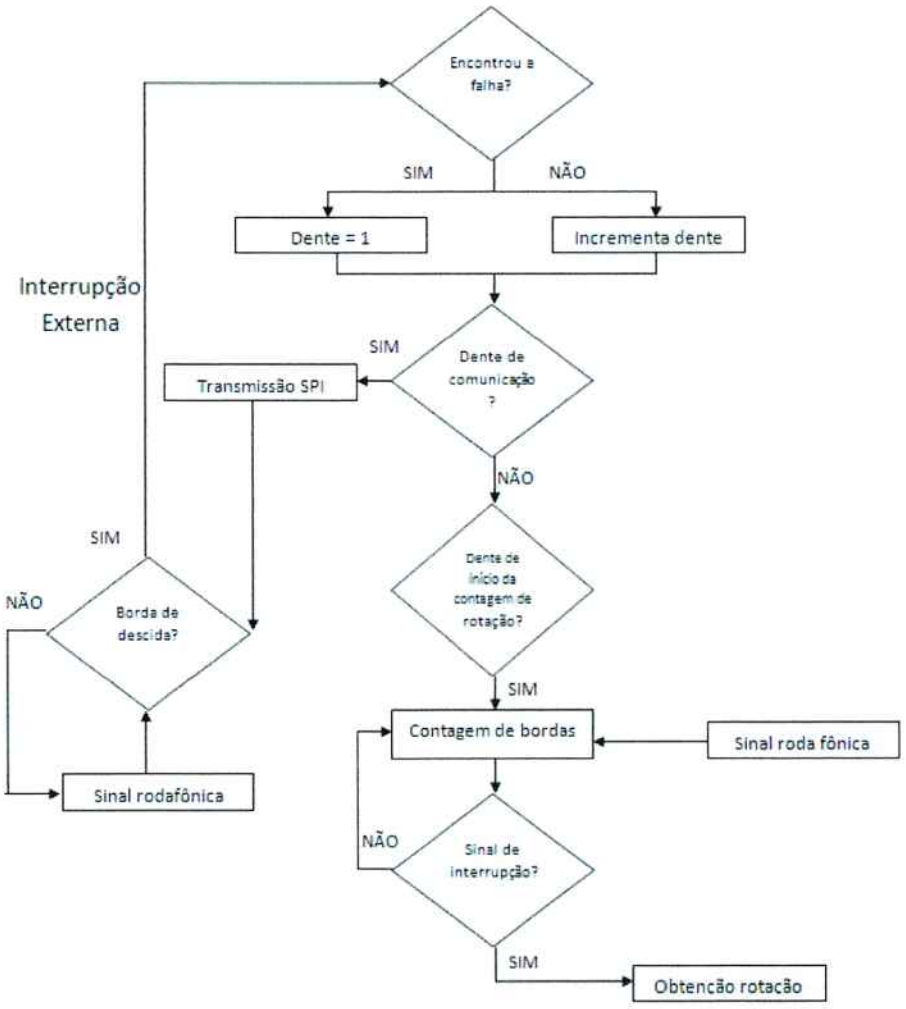


Figura 19 - Diagrama lógico bloco de sincronismo

No diagrama da Figura 19, podemos ver que não há uma estrutura em seqüência no processo. Porém para termos uma estrutura coerente é necessário que as ações sejam tomadas em seqüência. É esse o momento em que percebemos a grande importância do sincronismo gerado devido à falha que há na roda dentada. Mesmo com as ordens não sendo ditadas em seqüência, o programa percorrerá todos os processos de uma maneira coerente e sem se utilizar de informações atrasadas.

Para o bloco de Gerenciamento, segue o diagrama lógico a seguir:

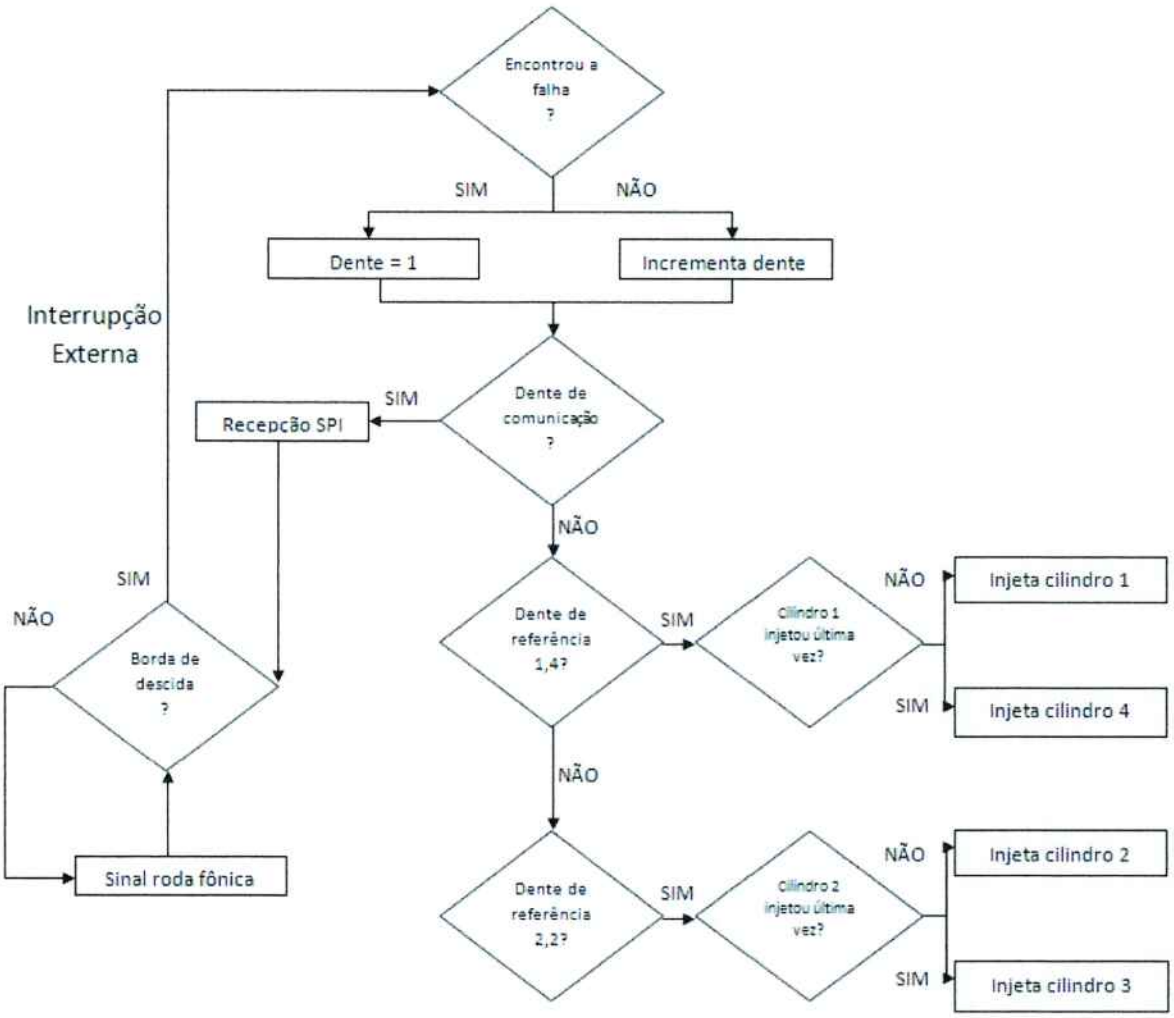


Figura 20 - Diagrama lógico bloco de gerenciamento

Para o diagrama da Figura 20 vemos uma estrutura parecida com a anterior, porém ele se difere na parte onde é obtido o valor de rotação para servir de referência para a injeção de combustível.

O que não está explicito nos diagramas anteriores é a obtenção da falha, que serve exatamente para numerar os dentes percorridos e assim compará-los aos dentes de referência para injeção e comunicação SPI, que obviamente é o mesmo para os dois controladores. Essa identificação é feita assim que a borda de descida é detectada, em caso de se tratar da falha.

Além disso, para efeito de processamento do bloco de Sincronismo, não é interessante que haja uma perda de tempo para o cálculo do dente de referência e do tempo de injeção. Como a rotação que seria de 0 a 6000 RPM é transformada, devido à aproximação, em uma rotação que cresce de 200 em 200 rotações por minuto, serão preenchidos vetores com os valores de dente de referência e tempo de injeção para cada uma das 30 rotações possíveis, para assim termos o valor pronto ao invés de ser calculado durante o processo.

3.3 - Identificação da Falha

A identificação da falha que há na roda-fônica é a responsável por todo o sincronismo que há entre os processadores, além de possibilitar fielmente a implementação de um sistema de injeção que respeite o ciclo Otto de funcionamento de um motor.

A idéia básica para se encontrar a falha é a baseada no tempo entre duas bordas de descida. Durante a falha esse tempo é muito maior que o tempo existente entre os outros dentes, como pode ser observado na figura a seguir.

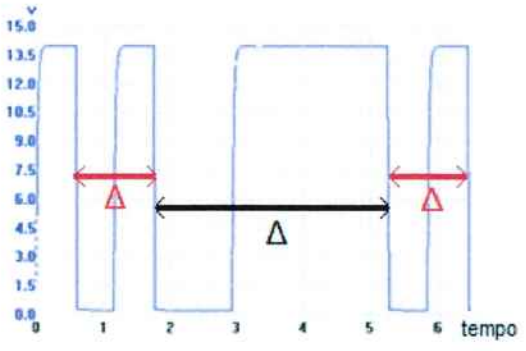


Figura 21 - Tempos dos pulsos

Na prática foi utilizado um timer que é lido e inicializado a cada borda de descida. Se o valor gerado pelo timer for muito maior que o valor anterior, significa que se trata da falha. Logo o próximo dente é classificado como sendo o dente número um. A consistência desse método foi analisada através de sucessivos testes, além de serem testadas outras formas para se achar a falha. Para o motor elétrico testado, foi encontrado o fator ideal como sendo k = 1.35, ou seja, se o timer atual acusar um valor 35% maior que o anterior, estamos diante de uma falha – Isso para todas as rotações.

3.4 - Cálculo do valor de rotação

Como o projeto foi baseado na rotação, um cálculo aproximado da rotação teve que ser feito segundo o sinal proveniente da roda-fônica. Um método bastante interessante que foi utilizado foi o de contar o número de bordas do sinal existentes em um dado tempo. Por uma razão estratégica que será explicada adiante, adotamos esse período de tempo como sendo 5ms. Para exemplificar o método, segue o esquema abaixo:

Seja uma rotação Ω:

O período de 1 dente, pelo sinal da rodafônica é:

$$T = \underbrace{\frac{60}{\Omega}}_{\text{Período de 1 volta}} * \underbrace{\frac{1}{60}}_{\text{Total de dentes}} = \frac{1}{\Omega} \quad [s]$$

Portanto em 5ms o o sinal apresentará N dentes, onde:

$$N = 0{,}005 * \Omega$$

Se definirmos como M o número de bordas que há nesse sinal, teríamos

$$M = 2N = \frac{\Omega}{100}$$

Figura 22 – Cálculo da rotação do motor

Portanto, basta contarmos o número de bordas que há no sinal durante 5ms que teremos a informação, em outra escala, da rotação do motor. A precisão desse método é de 100 RPM para menos, uma vez que é possível se interromper essa contagem quando uma borda estiver na iminência de ser contabilizada.

Poderíamos ter considerado para esse cálculo um período maior de tempo de amostragem, porém, apesar de aumentar a precisão, poderíamos estar usando um valor atrasado de rotação. Na rotação máxima que foi utilizada, 6000RPM, em 5 ms percorremos 60 dentes, o que corresponde à metade de uma rotação. Isso garante que esse cálculo sempre dura menos que uma rotação.

3.5 - A injeção segundo o ciclo Otto

Segundo a teoria envolvida no funcionamento do motor ditado pelo Ciclo Otto, cada cilindro deve completar o seu ciclo em duas voltas do eixo do motor, o que quer dizer que cada um contribuirá com um giro de 180° no eixo. Vemos pela imagem abaixo que os cilindros 1,4 e 2,3 são opostos. Isto quer dizer que o que um deles faz em uma volta o outro fará na volta seguinte. Por essa razão, para efeito de processamento esses pares de cilindros podem ser tratados igualmente, devendo ser diferenciado apenas o canal de saída do controlador, que tem que ser direcionado a um ou ao outro cilindro, dependendo da volta em que está.

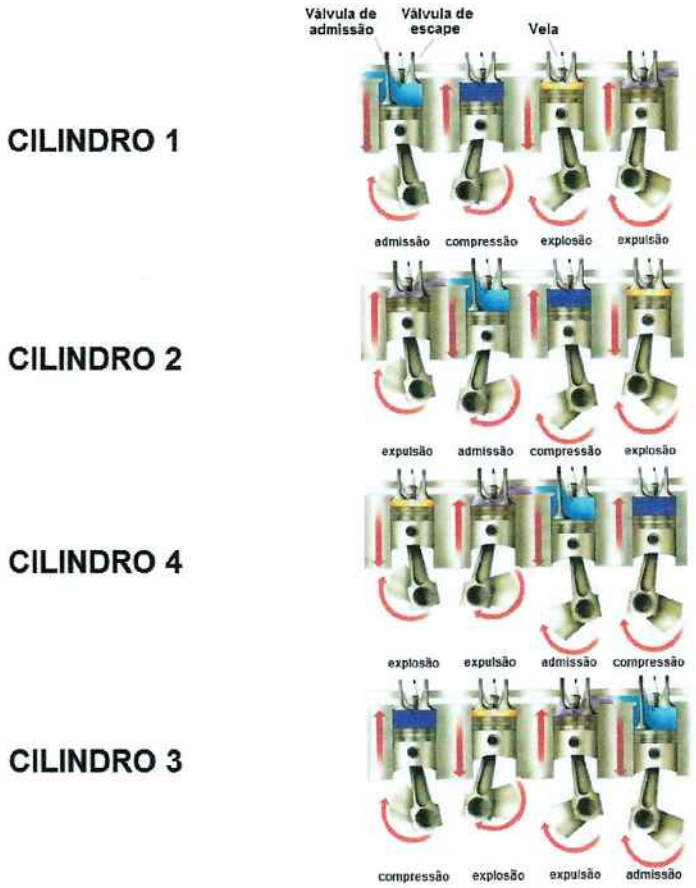


Figura 23 - Análise da posição dos cilindros com o motor funcionando de acordo com o Ciclo Otto – Ref. [15]

Pela figura acima também podemos ver que o fenômeno da admissão deve ocorrer a cada meia volta, que é a diferença que há entre duas fotos de um mesmo cilindro na figura. Para garantir que isso acontecesse, foi definido que todas as injeções de combustível devem acabar em um dente específico da roda-fônica, para assim se garantir o funcionamento do sistema de injeção de combustível segundo o ciclo Otto. Esse dente foi definido no projeto como sendo o dente 13 para os cilindros 1 e 4 e o dente 43 para os cilindros 2 e 3. Se fosse definido um dente para se iniciar a injeção, em uma eventual mudança brusca de rotação (como o que

ocorre em mudanças de marchas) o sincronismo seria perdido, o que comprometeria o ciclo correto de funcionamento do motor.

Para a realização desse método que garante que sempre há rotações a cada meia volta, é necessário, então, se calcular o dente de referência, que é o dente onde a rotação deve se iniciar para que ela se encerre no dente predeterminado. Esse dente de referência é dependente da rotação e do tempo de injeção necessário para aquela rotação. Além disso, como a contagem de dentes é um número inteiro, deve-se, também, calcular um tempo de atraso na injeção para que seja injetada a quantidade certa de combustível. Por exemplo, se o calculo mostrar que a injeção deve perdurar por 13,5 dentes, o que é feito é se colocar o dente de referência a 14 dentes do dente de encerramento e se atrasar em um tempo proporcional a 0,5 dente o início do processo de injeção.

Figura 24 - Período de injeção, segundo a estrutura dentada da rodafônica

Como o máximo tempo de injeção é de 3,75ms, podemos garantir que 24 dentes são suficientes para qualquer rotação até 6000RPM, uma vez que essa rotação necessita de 22,5 dentes para a completa injeção. No esquema acima vemos que em nenhum momento haverá um sobreposição de injeções, o que certamente contradizeria o ciclo correto de funcionamento do motor.

3.6 - Tempo de injeção

Pela teoria apresentada no escopo do projeto, foi visto que o tempo de injeção, em um ciclo, depende da massa de ar que é admitida pelo motor, além de outros fatores que são fixos para um dado motor. Portanto, podemos considerar que o tempo de injeção é uma função da massa de ar admitida, ajustada a um fator k, que é fixo.

Para o projeto, como não pôde ser considerada a massa de ar admitida, foi utilizada outra variável que está diretamente ligada a essa massa de ar: a rotação. Para efeito teórico, a rotação do motor seria uma conseqüência da massa de ar admitida, e depois reagida com a massa de combustível para gerar energia. Porém para efeito prático podemos simular essa massa admitida através da rotação em que se encontra o motor, uma vez que há uma ligação íntima entre essas duas variáveis.

Como o projeto carrega certo teor de Engenharia Reversa unido à teoria de injeção eletrônica, foram consultados alguns métodos práticos para ver como a massa de ar admitida e a rotação se relacionam. Em outras palavras, nos utilizamos de um conhecimento prático para complementar a teoria a respeito da injeção de combustível, devido ao foco do projeto estar na estruturação do processo de injeção de combustível propriamente dita.

O método escolhido foi apresentado ao orientador, Prof. Dr. Armando Laganá, por um companheiro seu de Fatec. Trata-se de Marco Aurélio Fróes, Consultor de Pós Vendas da Volkswagen. O método prático desenvolvido por ele, assim como o método teórico são apresentados abaixo:

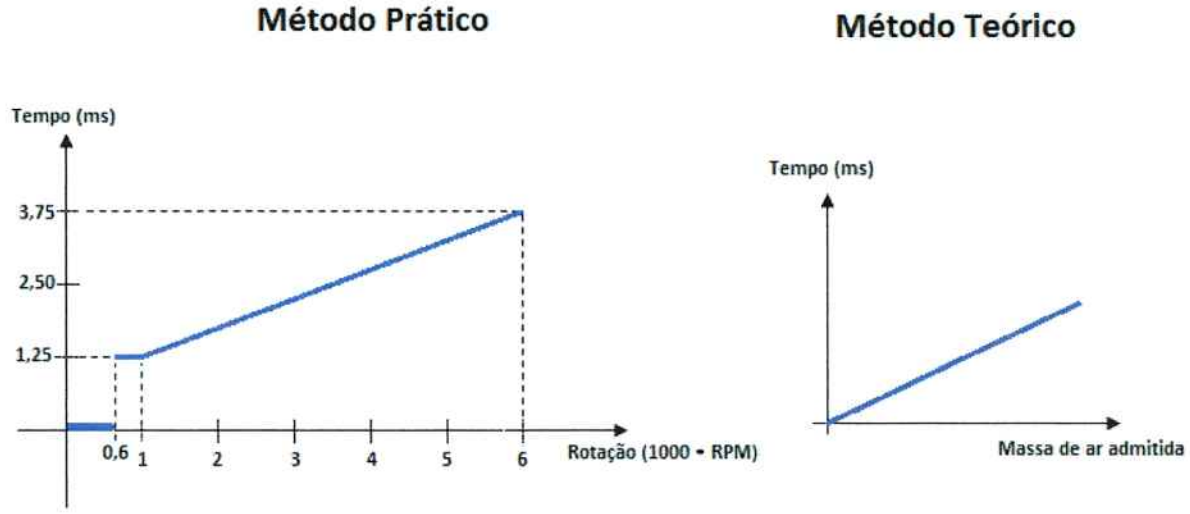


Figura 25 – Método Prático versus Método Teórico

Se olharmos na raiz da solução, ambos os métodos são indiferentes, uma vez que o importante para o projeto é haver um sincronismo e eficiência na hora da injeção e comunicação dos blocos controladores. Na prática esses dois métodos só diferem nos valores de tempo de injeção e dentes de referência que são

armazenados dentro dos microcontroladores. Porém a estrutura a que são expostos é exatamente a mesma.

3.7 - Protocolo de Comunicação SPI

Considerando o fato de que o projeto trabalha com um conjunto de dois microcontroladores PIC, um para o gerenciamento e outro para o sincronismo, tornou-se necessária a existência de uma comunicação entre ambos, como já dito anteriormente no tópico *3.2 - Sincronismo e Gerenciamento*, contudo, esta comunicação é de somente uma via, os dados vão do microcontrolador do gerenciamento para o microcontrolador do sincronismo.

Para ser realizada a comunicação entre microcontroladores, existem diversos tipos de técnicas de comunicação, que podem ser divididas em duas grandes categorias: Serial e Paralela. Na comunicação serial, os dados a serem transmitidos são fracionados em pequenas partes (bits), e então transmitidos, daí a denominação serial, dentre os tipos de técnicas podemos citar: I²C, SPI, 1-WIRE, LIN, CAN, etc. Já na comunicação paralela, os dados (bits) são transmitidos simultaneamente, em paralelo, como exemplos desta técnica, podemos citar os barramentos internos dos microprocessadores e microprocessadores, barramento ISA, PCI, VESA, AGP, SCSI, IDE, etc.

Considerando este razoável número de técnicas disponíveis, era necessária a escolha de uma para ser utilizada no projeto, obviamente, não houve tempo suficiente para serem analisadas, a fundo, todas as vantagens e desvantagens de das técnicas citadas, o leque de opções foi limitado para três delas: I²C, SPI e CAN.

Em um primeiro momento, deu-se início ao trabalho com o protocolo CAN, uma vez que o projeto foi desenvolvido em um kit CAN, já mostrado anteriormente, esse kit já possui toda a infra-estrutura necessária para ser feita essa comunicação, além disso, já havia uma certa experiência neste protocolo, por já ter sido estudado na disciplina PS2618, porém, ao longo do desenvolvimento, notamos que esse protocolo exigia uma maior velocidade de processamento do microcontrolador, e para o projeto isso é altamente indesejável, outro motivo para este protocolo ser descartado foi o fato de que este protocolo utiliza um controlador externo, o chip MCP2515, e o mesmo utiliza o pino de interrupção externa (RB0) do PIC 16F877A, que é o mesmo pino que foi utilizado para realizar a leitura do sinal da roda-fônica, logo, teve-se que fazer uma escolha, ou trocávamos de protocolo de comunicação, ou deixávamos de usar o pino de interrupção externa para ler o sinal. Vendo o custo x benefício destas duas opções, a opção de trocar o protocolo de comunicação, optando pelo protocolo SPI foi escolhida.

O protocolo SPI, é de certa forma, mais simples do que o protocolo CAN. O próprio protocolo CAN pode ser considerado uma "evolução" do protocolo SPI, uma

vez que o CAN se utiliza do SPI para fazer a comunicação, com a grande diferença de que no CAN é possível fazermos o endereçamento dos dispositivos. No caso do protocolo SPI, a comunicação só pode ser de um para um, ou seja, de um mestre para um escravo, como já dito anteriormente, esse é exatamente o caso do projeto, o microcontrolador de gerenciamento fará o papel de escravo, e o microcontrolador de sincronismo fará o papel de mestre, logo, para este caso, essa limitação do SPI não foi um problema, e fazendo isso, economizou-se processamento do microcontrolador, pois não será mais preciso fazer o endereçamento dos dispositivos.

3.7.1 - Funcionamento do Protocolo

O protocolo de comunicação SPI é um protocolo de transmissão de dados de forma serial, o seu nome é uma abreviação de Serial Peripheral Interface, ou Interface Serial de Periféricos. O PIC 16F877A já possui todos os recursos que possibilitam o uso deste protocolo. Este protocolo trabalha com 4 sinais básicos:

- CS ou SS: É o Chip Select (Seleção de Dispositivos), utilizado para habilitar o dispositivo com o qual se deseja comunicar. Pino RA5 do PIC.
- Clock ou SCLK: como o próprio nome diz, é o pino de clock, utilizado para a sincronização entre mestre e escravo. Pino RC3 do PIC
- SI (Serial In) ou SDI: É o pino de entrada de dados. Pino RC4 do PIC
- SO (Serial Out) ou SDO: É o pino de saída de dados. Pino RC5 do PIC

Para realizarmos as configurações do PIC16F877A, utilizamos as seguintes funções, já presentes na biblioteca do compilador CCS:

- SETUP_SPI( ): Essa função que configura se o dispositivo será um mestre ou escravo, se os dados serão transmitidos na borda de subida ou descida do Clock, e o próprio Clock em si.
- SPI_READ ( ): Como o nome sugere, é a função que fará a leitura do dado recebido pela comunicação.
- SPI_WRITE ( ): Como o nome sugere, é a função que fará a escrita do dado a ser enviado pela comunicação.

Basicamente, a comunicação do Sistema de Injeção Eletrônica será configurada de acordo com a figura a seguir:

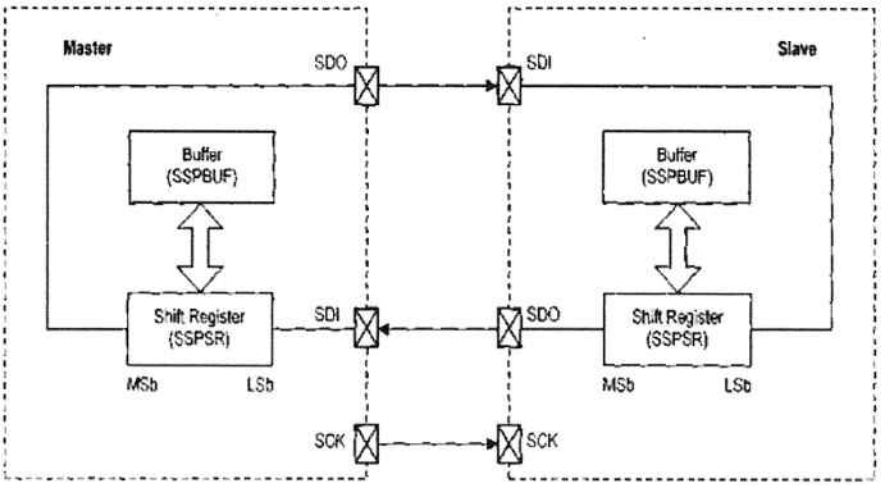


Figura 26 – Diagrama de ligações Mestre e Escravo – Ref. [14]

A figura mostra que há dois registradores, SSPBBUF e SSPSR, que ficam encarregados pela manipulação dos dados que serão transmitidos e recebidos. O protocolo SPI permite que, enquanto o mestre envia algum dado para o escravo, o escravo também envia outro dado para o mestre, simultaneamente, porém, para o projeto, esse segundo caso será desconsiderado. Basicamente, a transmissão é feita da seguinte forma:

- Mestre possui um dado a ser enviado, que fica no Buffer SSPBUF.
- Esse dado precisa ser serializado (quebrado), quem faz isso é o registrador SSPSR (este registrador é interno, não acessível).
- Mestre gera oito pulsos de Clock (cada pulso transmite um bit)
- É feita a troca de dados, o dado (8 bits) recebido fica no SSPSR do escravo.
- No escravo, o dado (8 bits) recebido vai para o SSPBUF
- O procedimento se repete até que todos os bits sejam enviados e recebidos

3.8 - Timers e Interrupções

O microcontrolador PIC16F877A possui três *Timers*, e um pino de interrupção externa, que serão de extrema importância para o nosso projeto. Cada um destes *Timers* possui características próprias, como o limite de contagem, o tipo de incremento, os pré e post-scales, geração de interrupções, etc.

Há, basicamente, quatro funções que são utilizadas na manipulação dos *Timers*:

- GET_TIMERX( ): Essa função lê o conteúdo de um timer, sendo que o valor retornado é uma variável inteira de 8 ou 16 bits, dependendo do timer que está sendo lido, e do modelo do PIC em questão.
- SET_TIMERX( ): Essa função modifica o conteúdo de um timer, sendo que o valor a ser escrito irá depender do tipo de timer e do modelo do PIC em questão.
- Função trata_timer: Esta função, que sempre vem precedida por um comando #int_timerX, é responsável pelo tratamento da rotina de interrupção, o programa entra nesta rotina de interrupção quando o contador (Timer) estiver "estourado" a sua contagem.
- SETUP_TIMER_X( ): Esta função que determina o modo de operação do Timer, ou seja, determina se o Timer utilizará um clock interno ou externo, se esse clock será dividido por 2, 4, 8, 16, etc (dependendo do Timer, esses valores mudam).

A seguir, iremos descrever brevemente cada um destes três Timers, bem como o modo que os utilizamos no projeto.

3.8.1 - Timer 0

O Timer 0, ou TMR0, é um contador de 8 bits, cujo valor é armazenado no registrador TMR0 do PIC, que pode ser tanto lido quanto escrito, possibilitando a inicialização do contador. No caso do projeto, este Timer 0 foi utilizado para identificar a falha da roda-fônica, em ambos os PICs (sincronismo e gerenciamento.

O método para encontrar a falha da roda-fônica baseou-se nas funções get_timer0( ) e set_timer0( ), a cada ciclo de iteração do programa, guarda-se o valor do contador TMR0 em uma variável int, este valor simboliza a duração de um pulso (ou, de um dente da roda-fônica), caso este valor seja maior que um valor de referência significa que o pulso (ou dente) em questão é uma falha, e então teremos uma sub-rotina própria que fará o tratamento adequado dos sinais a serem gerados neste caso.

3.8.2 - Timer 1

O Timer 1, ou TMR1, é um contador de 16 bit, cujo valor é armazenado em dois registradores, TMR1H e TMR1L, que podem ser escritos e lidos pelo programador. No caso do projeto, utilizou-se este Timer 1 para: cálculo da rotação, para o PIC do sincronismo; e gerar o sinal para o bico injetor para os cilindros 1 e 4, para o PIC do gerenciamento.

O método para calcular o valor de rotação baseou-se na utilização de uma rotina de interrupção do TMR1, no PIC do gerenciamento. Para isso, dois comandos foram feitos na rotina principal do comando: enable_interrupts(global | int_timer1), que é o comando que habilitará a interrupção vinda do TMR1, e um set_timer1( ), onde coloca-se um valor no TMR1 para que esse gere a interrupção de 5ms. Conforme explicado no tópico *3.4 - Cálculo do valor de rotação*, assim que a contagem do TMR1 "estourar", o programa irá entrar na rotina de interrupção, onde será feita a contagem dos pulsos durante 5 ms, obtendo, assim, o valor de rotação em RPM.

O método para realizar a injeção nos cilindros 1 e 4 baseou-se, também, na utilização de uma rotina de interrupção do TMR1, no PIC do sincronismo. Este é um caso análogo ao anterior, onde foram inseridas na rotina principal do programa as mesmas funções: enable_interrupts( ) e set_timer1( ), o intuito é basicamente o mesmo, habilitar a interrupção do TMR, e "setar" o TMR1 com o valor adequado para o tempo de Injeção, de acordo com a regra prática que foi adotada, explicada no tópico *3.6 – Tempo de Injeção*, assim, quando o TMR1 "estourar", o programa irá entrar na rotina de interrupção, e lá gerar o pulso que será utilizado para acionar os bicos injetores dos cilindros 1 e 4.

3.8.2 - Timer 2

O Timer 2, ou TMR2, assim como o Timer 0, é um contador de 8 bits, cujo valor é armazenado no registrador TMR2 do PIC, que pode ser tanto lido quanto escrito, possibilitando a inicialização do contador. No projeto, esse Timer 2 é utilizado somente no PIC do sincronismo, para gerar o sinal para o bico injetor para os cilindros 2 e 3.

O método para realizar a injeção nos cilindros 2 e 3, é exatamente o mesmo que o utilizado para os cilindros 1 e 4, a única diferença é a que o Timer 2 possui 8 bits, e por esse motivo, o valor utilizado na função set_timer2 ( ) é diferente.

3.8.3 - Interrupção Externa

Como dito, o PIC16F877A possui apenas um pino para a interrupção externa, que a princípio seria utilizado pelo controlador CAN, porém, ao longo do desenvolvimento do projeto notou-se a necessidade de utilizá-lo como se fosse um sinal de entrada. A contagem de dentes da roda-fônica é algo que precisaria ser feita

em ambos os PICs (sincronismo e gerenciamento), e tal contagem estava consumindo processamento da rotina principal do programa. Uma solução para este problema foi colocar essa contagem de dentes em uma subrotina, no caso, uma subrotina de interrupção.

O método para a implementação desta rotina de interrupção externa é razoavelmente simples, uma vez que este só irá fazer a contagem de dentes. Na rotina principal do programa, habilitou-se a interrupção externa através de duas funções simples: enable_interrupts (global | int_ext) e EXT_INT_EDGE (H_TO_L), a primeira função é a que habilita a interrupção propriamente dita, e a segunda é que configura se a mesma será numa borda de descida (H_TO_L) ou numa de subida (L_TO_H). No caso, optou-se por ser de borda de descida, devido ao formato do sinal da roda-fônica, como mostra a figura abaixo:

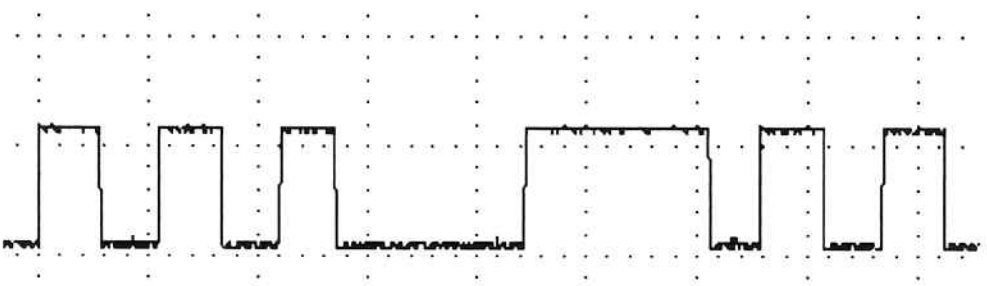


Figura 27 – Topologia do sinal da roda-fônica

Pelo pulso da falha, notou-se que o início de cada pulso (dente) é sempre em baixo, por esse motivo, optou-se por configurar a interrupção externa como sendo de borda de descida. Logo, a cada borda de descida do sinal da roda-fônica, a sub-rotina de interrupção externa irá fazer a contagem dos dentes, e verificará se o dente em questão é uma falha ou não. Ao fazer isso, há uma economia no processamento do programa principal (*main*).

3.9 - Módulo de Injeção

O módulo de injeção será composto pelos bicos injetores. Como mostrado no tópico *2.7 – Atuador: Bico Injetor,* o sinal de entrada do bico injetor possui uma topologia bem particular, o Método Pulsado ("Chopped"). Gerar um sinal como o da Figura 12 com o microcontrolador seria algo bastante trabalhoso. Entretanto, há um dispositivo, o componente LM1949, que fará esta conversão do sinal. A figura abaixo mostra o circuito de interface deste componente.

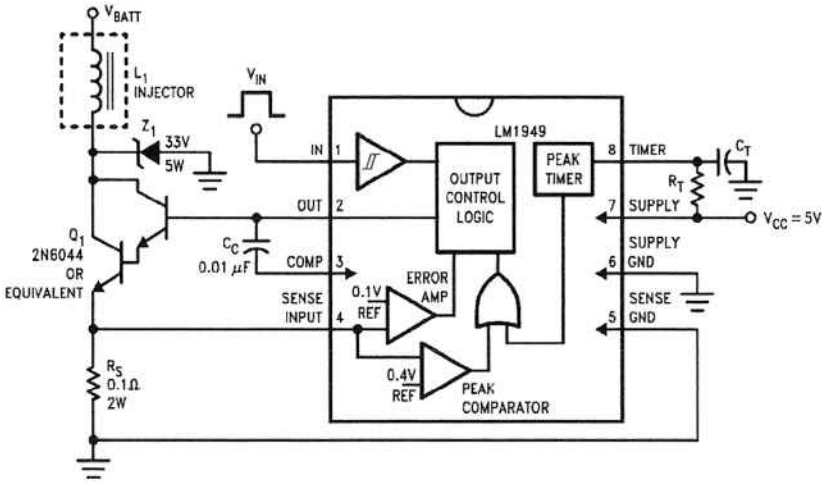


Figura 28 – Circuito de Interface do LM1949 – Ref. [13]

Como mostra a figura a seguir, na entrada deste componente só será preciso um pulso quadrado e na saída teremos o sinal pulsado desejado.

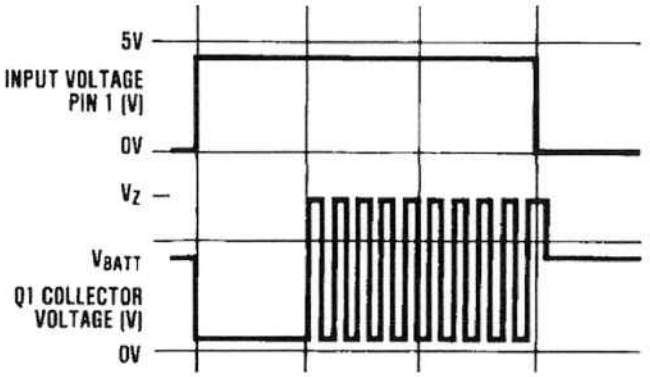


Figura 29 – Formas de onda do LM1949 – Ref. [13]

A figura mostra também que o tempo de injeção do método pulsado é exatamente o tamanho do sinal de entrada, logo, para controlar adequadamente os tempos de injeção para cada caso, só é necessário gerar, pelo microcontrolador, os pulsos quadrados com os respectivos tamanhos.

3.10 – Validação e Depurações

Para verificar a consistência dos resultados originados ao longo da implementação prática, utilizou-se diversos métodos, uma vez que a depuração de um sistema micro-controlado é muito mais complicada que um sistema em Desktop, onde podemos rastrear o valor de todas as variáveis e assim analisá-los.

Além disso, teve-se que fazer o projeto por etapas, podendo apenas seguir para próxima etapa após a conclusão e precisão dos resultados obtidos na etapa anterior.

A seguir serão apresentadas as etapas, com a devida explicação a respeito da depuração, assim como o comportamento do processo de acordo com o aumento da rotação do motor.

3.10.1 - Identificação da falha

Como já explicado anteriormente, a falha é encontrada a partir da comparação do comprimento que há entre cada borda de descida. Do ponto de vista teórico, todos os comprimentos são iguais, sendo a única exceção o da falha, que é aproximadamente três vezes maior que os demais. Porém na prática vemos não conseguiu-se comparar tão facilmente o tamanho desses pulsos.

Após uma análise muito grande do comprimento dos pulsos, que é dado pelo valor lido no Timer 0, conclui-se que se um pulso for 35% maior que o anterior, ele corresponderá à falha, para qualquer rotação dentro das especificadas no projeto. Para chegar nesse valor, analisou-se através de amostras em displays o valor lido no timer para cada dente. Para isso, foram armazenados 60 valores consecutivos de timer, aproximadamente, em vetores e após isso imprimiu-se e anotou-se, lentamente, esses valores no display. Processo esse que se repetiu por diversas rotações.

Para a depuração, simplesmente mudou-se o valor de um bit toda vez que a falha era encontrada, e então se analisou essa variação num osciloscópio, juntamente com o sinal original da roda-fônica. A seguir seguem exemplos da depuração e confirmação da consistência dos resultados, tanto de forma específica, como nas figuras à esquerda, quanto de uma forma mais geral, para se verificar a capacidade de captura de falhas simultâneas, nas figuras à direita:

Ω = 1340 RPM

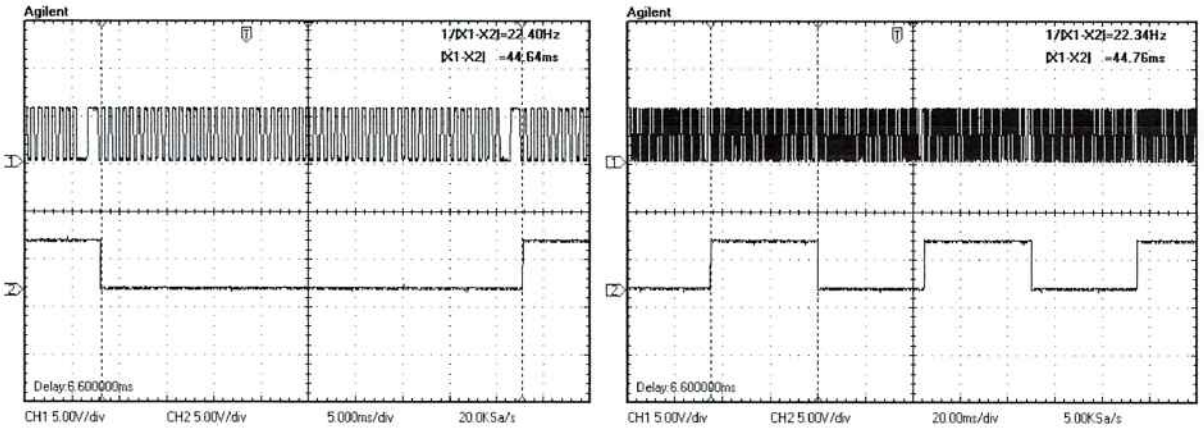


Figura 30 – Identificação da falha para 1340 RPM

Ω = 2876 RPM

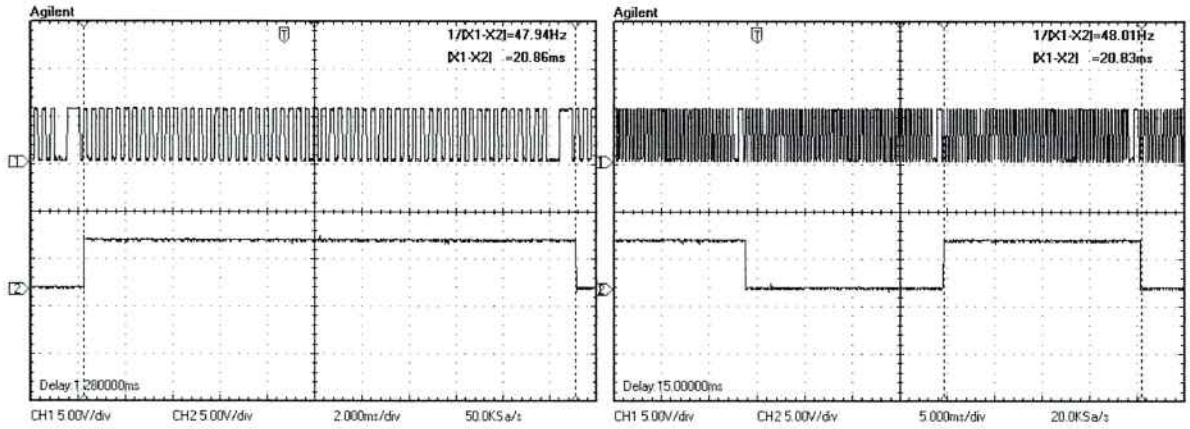


Figura 31 – Identificação da falha para 2876 RPM

Ω = 4630 RPM

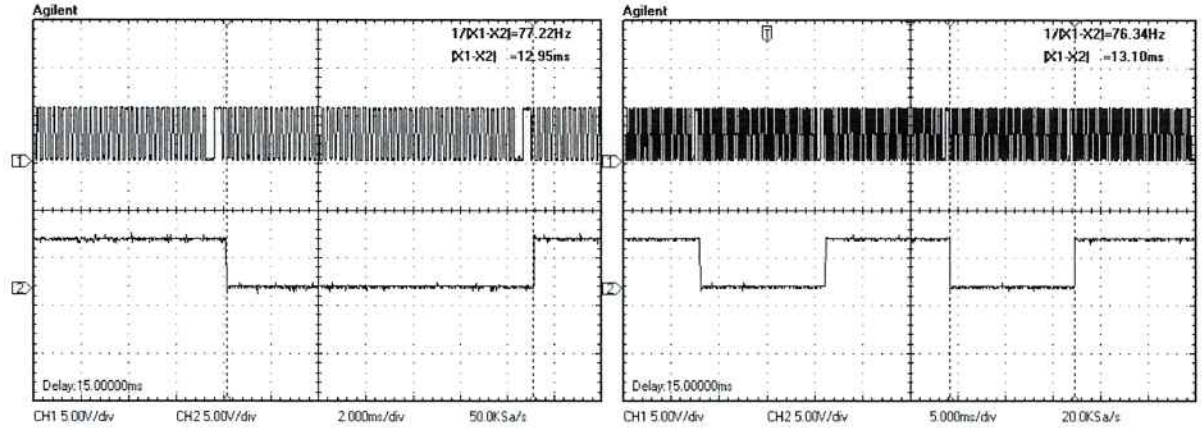


Figura 32 – Identificação da falha para 4630 RPM

Ω = 6230 RPM

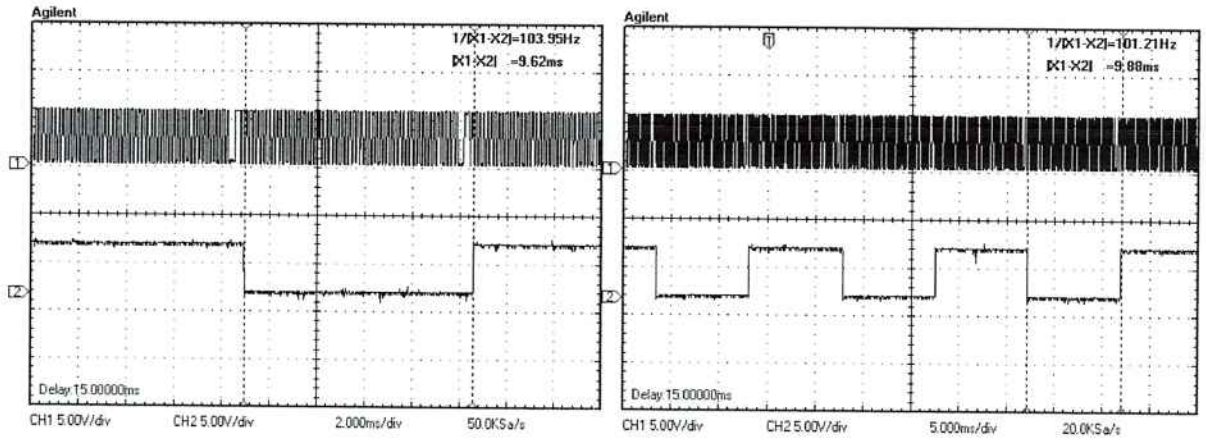


Figura 33 – Identificação da falha para 6230 RPM

Uma observação importante a respeito das figuras acima está no valor de duração da falha, valor este que foi medido com os cursores do osciloscópio. Ao transformarmos este valor de Hz para RPM em uma simples multiplicação por 60, teremos o valor aproximado de rotação.

3.10.2 - Tempo de referência para achar a rotação

Para a determinação dos 5ms necessários como referência para o nosso cálculo de rotação foi apenas necessário se inicializar o timer com um valor correto para que após 5ms ele interrompesse a contagem. Essa parte foi relativamente simples, uma vez que se trata de um período pequeno de tempo – para períodos maiores precisaríamos de técnicas que aumentassem a precisão do timer, pois ao contrario do que tivemos que fazer, o timer precisaria ser "estourado" diversas vezes, o que gera um erro crônico. Por fim, seguem abaixo figuras que ilustram esse tempo referência assim como a contagem de bordas e a verificação de um valor coerente de rotação. O momento para se iniciar a contagem é assim que a falha é encontrada, ou seja, ocorre no dente número um, como mostrado abaixo. Além disso, é mostrado também que em todas as voltas é calculado o valor da rotação.

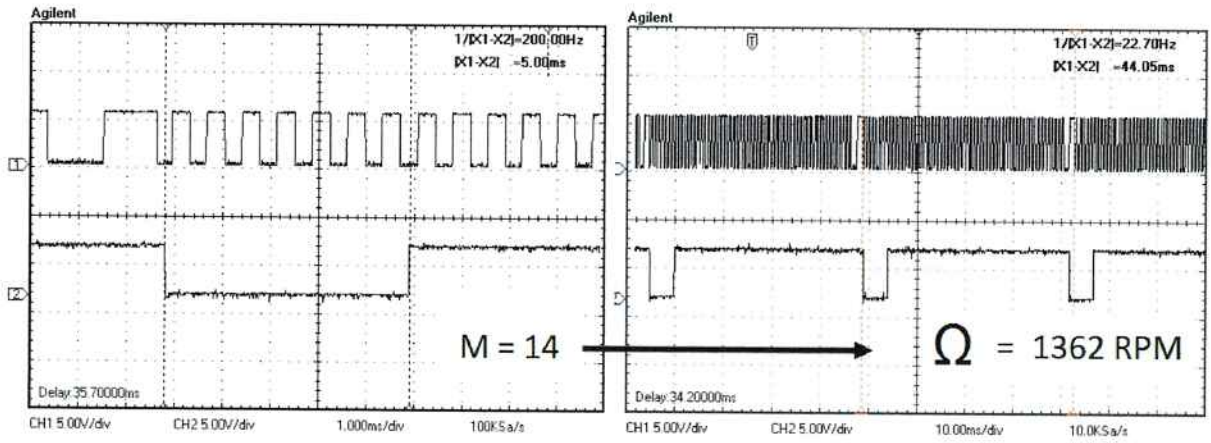


Figura 34 – Interrupção de 5ms em escala ampliada (à esquerda) Duração de um volta (à direita)

Nota-se pela figura acima, a coerência dos valores, uma vez que o valor de M = 14 (o que equivaleria a 1400 RPM), e a rotação de 1362 RPM (22,70Hz * 60 = 1362). Abaixo, temos mais figuras ilustrando a interrupção gerada.

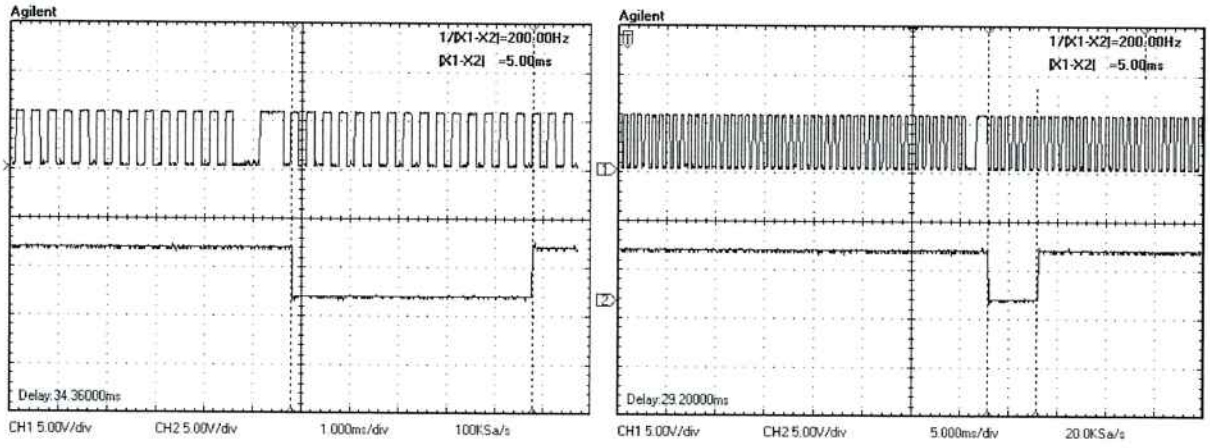
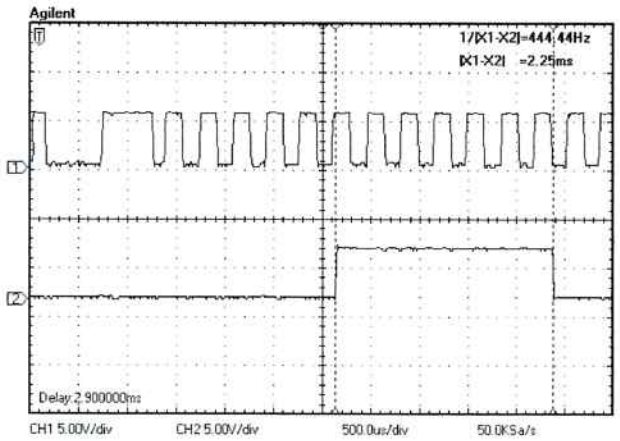


Figura 35 – Interrupção de 5ms em diferentes escalas

3.10.3 - Dente de referência e tempo de injeção

Para a utilização do dente de referência e do tempo de injeção armazenados em vetores optou-se pela rotação de 3000RPM, uma vez que esses sinais são os sinais finais gerados pelo sistema de injeção. A depuração desse sinal é bem simples, já que precisamos apenas analisar o nosso resultado final. Ademais, o grande desafio é o de encontrar um sincronismo para que se possa realizar esses sinais. Segue em seguida o sinal para o exemplo citado acima:



Pelo gráfico prático apresentado no relatório

$t = 0.0005 * \Omega + 0.75$

para $\Omega = 3000$ RPM:

$t = 0{,}0005 * 3000 + 0{,}75 = 2{,}25$ ms

Figura 36 – Pulso de Injeção de 2,25 ms para rotação de 3000 RPM

Verificamos que o tempo de pulso, que é o tempo de injeção corresponde com o teórico, é bem condizente com o proposto, além do que o processo se encerra no dente 12, que é o dente especificado para o encerramento da injeção.

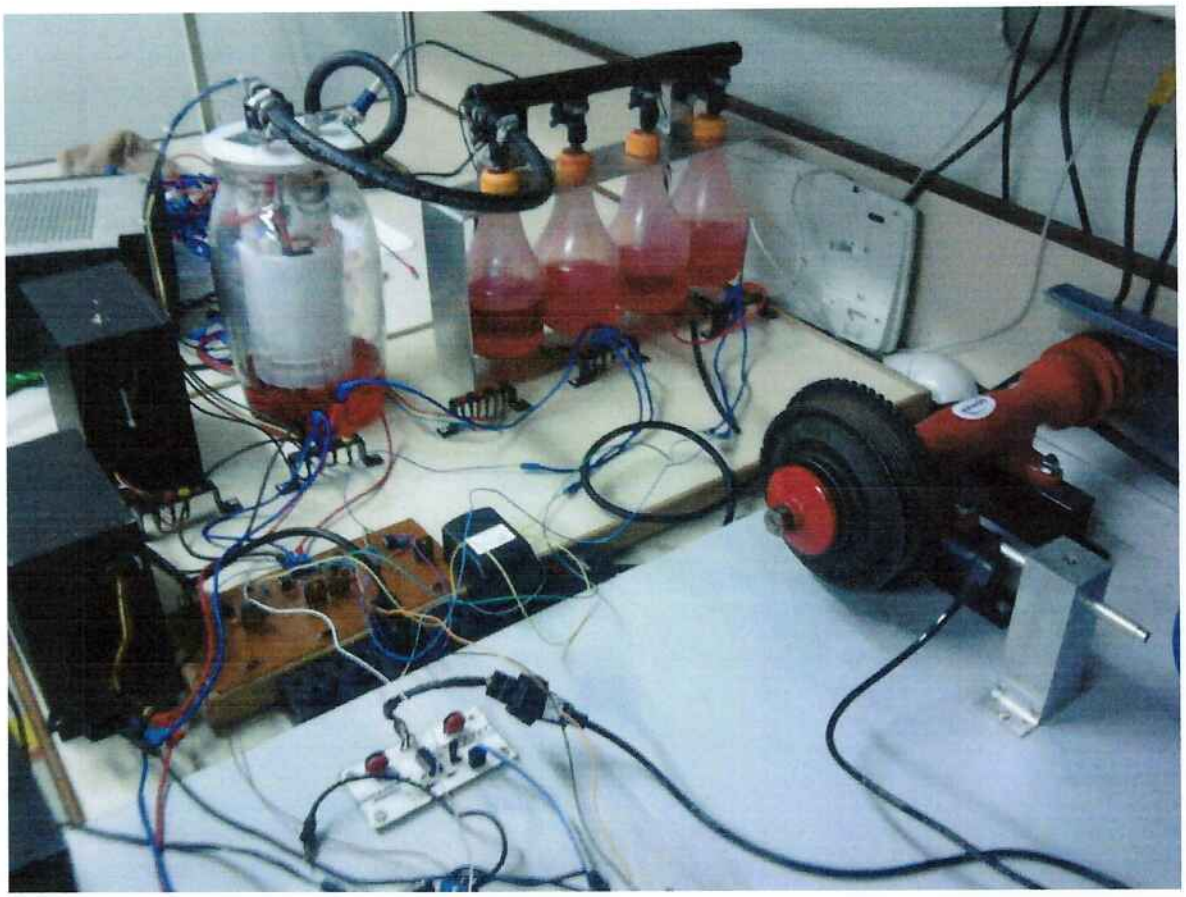


Figura 37 – Foto do SIE implementado com a roda-fônica e os bicos injetores

3.11 - Dificuldades Encontradas

Em todo projeto de Engenharia, é esperado que hajam algumas dificuldades encontradas durante o desenvolvimento do mesmo, esta seção visa listar as dificuldades encontradas durante o desenvolvimento do projeto e mostrar as alternativas de solução que foram tomadas frentes às mesmas.

A primeira grande dificuldade está relacionada com a velocidade de processamento do microcontrolador PIC 16F877A, apesar do mesmo operar com um cristal externo, de freqüência 20 MHz, vimos que de fato, a velocidade era menor, devido a uma série de fatores, como as configurações de bibliotecas ("includes"), de algumas funções, etc. Para rotações altas, isso acabou sendo um grande problema, pois perdíamos a contagem dos dentes da roda-fônica. Em face deste problema, tínhamos algumas alternativas a serem seguidas: Uma delas foi realizarmos a migração do microcontrolador PIC para algum outro, como o Atmel, por exemplo; esta hipótese foi descartada devido ao fato de que esta migração não seria nem um pouco simples, ou seja, exigiria um re-trabalho considerável, e fugiria um pouco do escopo do nosso projeto, que é a simplicidade da implementação, considerando o fato de que a tecnologia PIC é muito mais acessível, com relação à disponibilidade

de material para estudo. A segunda alternativa foi a migração para um outro microcontrolador PIC, no caso, migramos para o PIC18F452, que é semelhante ao 16F877A, fizemos algumas tentativas nesse microcontrolador, e não notamos muitas vantagens em realizar tal migração. Por último, temos a alternativa realmente adotada por nós, que foi a descentralização do sistema, ou seja, trabalhar com dois microcontroladores em paralelo, fazendo isso, praticamente dobramos o nosso processamento, porém, como desvantagem, teríamos que ter o cuidado para que a sincronização entre ambos seja confiável o suficiente.

Outra dificuldade foi a inacessibilidade de algumas informações, como dito anteriormente, as tecnologias implementadas nos sistemas de injeção atuais estão protegidas. A maior dificuldade foi precisar o tempo de injeção, sabe-se que o mesmo depende de parâmetros como a rotação, pressão, temperatura, etc, mas não há uma regra clara de como cada um destes fatores vão interferir. Como alternativa a este problema, adotamos duas frentes, uma foi utilizar como referência o método *Speed-Density*, contido nesta monografia, e a outra foi utilizar uma regra prática, mostrada ao nosso orientador por Marco Aurélio Fróes, da FATEC, e Consultor de Pós Vendas da *Volkswagen*.

Por último, tivemos como dificuldade a falta do "mock-up", que era um risco que nós já estávamos assumindo desde o início. Porém, nosso projeto não se limitou a apenas aplicarmos o nosso sistema em "mock-up", o fato de o aplicarmos em um motor elétrico não impactará grandes mudanças em termos de implementação, visto que o sinal mais importante para o nosso projeto é o da roda-fônica, e o mesmo pode ser medido perfeitamente no moto elétrico. Para medirmos valores como temperatura e pressão, podemos utilizar os potenciômetros contidos no próprio kit CAN. E por último, para efeito de realizarmos a injeção propriamente dita, os bicos injetores foram adaptados para realizar a injeção em compartimentos, como garrafas plásticas.

4 - Conclusão

Inserindo o projeto do Sistema de Injeção Eletrônica (*SIE*) no contexto da sustentabilidade e preservação do meio-ambiente, questões que estão em destaque recentemente, podemos concluir que o mesmo se encaixa no quesito redução da emissão de CO_2 e outros gases nocivos da atmosfera, além da economia de combustível.

Considerando ainda, o objetivo de desenvolver um sistema utilizando uma tecnologia totalmente aberta, o projeto SIE atente os requisitos. O sistema SIE apresenta conceitos desconhecidos para muitos usuários, como é o caso do sincronismo, e o processo de injeção segundo o Ciclo Otto de funcionamento do motor.

Obviamente, por se tratar de um sistema que possui certas limitações, não podemos afirmar que o sistema desenvolvido pelo projeto em questão seja mais eficiente dos que os sistemas existentes no mercado, sendo que este não era nem um dos objetivos do mesmo. Porém, ao criarmos um sistema totalmente transparente e acessível aos usuários, o projeto *SIE* abre uma margem para o desenvolvimento de projetos futuros que poderão contribuir para a criação de novos sistemas de injeção eletrônica mais eficientes. Logo, conclui-se que o maior valor agregado deste projeto não é o projeto fisicamente dito, e sim o conhecimento nele envolvido.

Por fim, lamentamos o fato de não termos tido a possibilidade de implementarmos o SIE em um "*mock-up*", porém, como já dito anteriormente o projeto não limitou-se a somente ser aplicável em um "*mock-up*", considerando o fato de que não houveram perdas na funcionalidade do sistema ao ser adaptado à um motor elétrico, e considerando também, o fato de que o maior valor agregado do projeto é o conhecimento nele envolvido, e não o projeto físico em si.

5 – Sugestões para Trabalhos Futuros

Como sugestões para futuros projetos baseados no SIE, temos:

- Implementação do SIE com tecnologias diferentes, como a utilização de microncontroladores *Atmel*, por exemplo.

  Justificativa: Fugindo um pouco da discussão de qual tecnologia é melhor (PIC *versus Atmel*), o simples fato de utilizar uma tecnologia diferente poderá agregar muito valor ao SIE.

- Adição de mais alguns sensores, deixando o sistema cada vez mais próximo aos sistemas disponíveis no mercado.

  Justificativa: Considerando que o tempo disposto para um projeto de formatura é razoavelmente baixo, limitações no número de sensores ocorrerão. Algo interessante a ser feito é: desenvolver vários projetos como sub-módulos que utilizam sensores diferentes, e por um fim, um único sistema macro de gerenciamento para tais sub-módulos.

6 – Referências Bibliográficas

[1] MANAVELLA, HUMBERTO J., **Controle Integrado do Motor – Sistemas de Injeção – Ignição Eletrônica.** São Paulo: Ed. HM Autotrônica, 2003

[2] SILVA, EDSON DA., **Injeção Eletrônica de Motores Diesel.** São Paulo: Ed. Ensino Profissional, 2006

[3] MILHOR, CARLOS E., **Sistema de desenvolvimento para controle eletrônico dos motores de combustão interna ciclo Otto.** São Paulo: USP, 2004. 101p. Dissertação de Mestrado

[4] PENIDO FILHO, P., **Os motores à combustão interna.** Belo Horizonte: Ed. Lemi, 1983

[5] SANTOS, PROF. ANTONIO M. D., **Preparação da Mistura Ar-Combustível.** Disponível em: <http://www.scribd.com/doc/13183026/Estequiometria-Preparacao-Da-Mistura-ArCombustivel>.

[6] Volksvagen, **A Tecnologia Total-Flex.** Disponível em: <http://www.volkswagen.com.br/pecas/noticias/pdf/203_treina.pdf>.

[7] Microchip Technology Inc., **PIC16F87XA Data Sheet.** Disponível em: <http://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf>

[8] NGK – NTK. **Sonda Lambda.** Disponível em: <http://www.ngkntk.com.br/sens_ox/default.html>

[9] Bosch, **Sensor Catalog.** Disponível em: <http://www.bosch.se/content/language1/downloads/Sensorkatalog_som_pdf.pdf>

[10] Bosch, **Automotive Industry Portal.** Disponível em: <http://www.bosch-motorsport.com/content/language2/html/index.htm>

[11] Relógios com Estilo, **Sensor de rotação do Virabrequim.** Disponível em: <http://relogioscomestilo.blogspot.com/2009/02/sensor-de-rotacao.html>

[12] Delphi, **Multec 3.5 Top Feed Fuel Injector - Application Manual**

[13] National Semiconductor, **LM1949 Data Sheet.** Disponível em: <http://www.national.com/ds/LM/LM1949.pdf>

[14] DE SOUZA, DAVID J., **Conectando o PIC – Recursos Avançados.** São Paulo: Ed. Érica, 2008

[15] COSTA, ANTÔNIO CARLOS N., **Emprego do Álcool Etílico nos Motores Aeronáuticos Alternativos.** Belo Horizonte, CIAAR (Centro de Instrução e Adptação da Aeronáutica), 2006. Disponível em: <http://www.ciaar.com.br/EM%20FOCO/2006/av-2/av2-alcool.html>

[16] PEREIRA, FÁBIO. **Microcontroladores PIC – Programação em C**. São Paulo: Ed. Érica, 2007

[17] BEPPU, ANDRÉ M.; KERN, CHRISTIAN S.; OLVEIRA, DALMO M.; **Sistema de gerenciamento descentralizado para a injeção eletrônica de um motor de ciclo Otto.** São Paulo: EPUSP, 2009. 90 p.

Apêndice A – Código do Microcontrolador de Gerenciamento

```
#include <16F877A.h>
#include "regs_16f87x.h"
#include <stdio.h>

#use delay (clock=20000000)
#fuses HS,NOWDT,PUT,BROWNOUT,NOLVP

// ENTRADAS
// As entradas devem ser associadas a nomes para facilitar a programação e
// futuras alterações do hardware.
#bit    ENT = portb.0    //sinal rodafonica - pino de interrupcao

// SAÍDAS
// As saídas devem ser associadas a nomes para facilitar a programação e
// futuras alterações do hardware.
#bit  RS           = portb.5
#bit  TX           = portc.6

int cont1; /*contador de dentes*/
int cont3; /*contador de bordas*/
int num_ciclos_med;
int num_ciclos;
int num_ciclos_ant0;
int rotacao;


#int_timer1
void trata_t1 ()
{
   TX = 1;
}

#int_ext
void external_interrupt()
{

if(ENT == 0) /*para rotacoes baixas ele ve ambas as bordas*/
{
      cont3++;
      cont1++; /*conta numero de dentes*/
      num_ciclos = get_timer0();
      set_timer0(10);

      num_ciclos_med = num_ciclos_ant0;
      num_ciclos_ant0 = num_ciclos;

        if (num_ciclos > 1.35*(num_ciclos_med)) /*a falha foi encontrada*/
      {
        cont1 = 1;
            RS=!RS;
      }
```

```
      if(cont1 == 45) /*comunicacao SPI*/
      {
         spi_write(rotacao);
      }
}

}

#int_timer0


int flag;
int data;

void main ()
{
   setup_adc_ports(NO_ANALOGS);
   setup_adc(adc_off);
   delay_ms(20);
   set_TRIS_A (0b00011000);
   set_TRIS_B (0b11011011);
   EXT_INT_EDGE(H_TO_L);
   enable_interrupts(global|int_ext);

   set_TRIS_C (0b10011111);
   setup_psp(PSP_DISABLED);
/*configuracao escravo*/
   setup_spi(SPI_SLAVE | SPI_L_TO_H | SPI_CLK_DIV_4 | SPI_SS_DISABLED |
SPI_XMIT_L_TO_H);

   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);    //define timer 0
   setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);

   cont1=2;
   cont3=0;

   flag = 0;
   TX = 1;


while (true)
 {


    if(flag == 0 && cont1==1) /*Começa o calculo da rotacao sempre no dente
um*/
    {
      set_timer1(40714); /*A interrupção ocorrerá após 5ms*/
      TX = 0;
      enable_interrupts(global);
      enable_interrupts(int_timer1);
      flag = 1;
```

```
      cont3 = 0;
    }
    else if( TX == 1 && flag == 1) /*Quando acabar essa interrupcao -
"seta" TX na trata_t1*/
    {
     disable_interrupts(global|int_timer1);
     rotacao = cont3;
     flag = 0;  /*Obter o valor só uma vez por volta*/
    }
 }
}
```

Apêndice B – Código do Microcontrolador do Sincronismo

```
#include <16F877A.h>
#include "regs_16f87x.h"
#include <stdio.h>

#use delay (clock=20000000)
#fuses HS,NOWDT,PUT,BROWNOUT,NOLVP

// ENTRADAS
// As entradas devem ser associadas a nomes para facilitar a programação e
// futuras alterações do hardware.
#bit    ENT = portb.0      //sinal interrupção

// SAÍDAS
// As saídas devem ser associadas a nomes para facilitar a programação e
// futuras alterações do hardware.
#bit  RS            = portb.5
#bit  TX            = portc.6
#bit  BICO1         = portc.0 //IO5
#bit  BICO2         = porta.3 //AN3
#bit  BICO3         = porta.5 //AN4
#bit  BICO4         = portc.1 //IO6
#bit  RX            = portc.7


int rotacao;
int i;
int num_ciclos_ant0;
int num_ciclos_med;
int num_ciclos;
int inj2;
int inj;
int cont1;
int flag14;
int flag23;



#int_timer1
void trata_timer1()
{
if (inj ==1)
{
      if (flag14 == 0)
      {
      BICO1 = 1;
      flag14 = 1;
      }
      else
      {
      BICO4 = 1;
      flag14 = 0;
      }
inj=0;
}
}
```

```
#int_timer2
void trata_timer2()
{
if (inj2 ==1)
{
      if (flag23 == 0)
      {
            BICO2 = 1;
            flag23 = 1;
      }
      else
      {
            BICO3 = 1;
            flag23 = 0;
      }
inj2=0;
}
}


#int_timer0


#int_ext
void external_interrupt()
{
if(ENT == 0) /*para rotações baixas, vê ambas as bordas*/
{
      i=0;
      cont1++; /*conta numero de dentes*/
      num_ciclos = get_timer0();
      set_timer0(10);

      num_ciclos_med = num_ciclos_ant0;
      num_ciclos_ant0 = num_ciclos;

      if (num_ciclos > 1.35*(num_ciclos_med)) /*encontrou a falha*/
      {
      //RS=!RS;
      cont1 = 1;
      }
      else if (cont1 == 45)    /*comunicação SPI*/
      {
      delay_cycles(4); /*A função read precisa esperar um tempo para os
dois pics se "alinharem" */
      rotacao = spi_read(0);
      }

}
}



int dente_referencia14;
int dente_referencia23;

int dente14[31];
int dente23[31];
long int t_inj[31];
```

```
void main ()
{

   setup_adc_ports(NO_ANALOGS);
   setup_adc(adc_off);
   delay_ms(20);
   set_TRIS_A (0b00000000);
   set_TRIS_B (0b11011011);

   EXT_INT_EDGE(H_TO_L);
   enable_interrupts(global|int_ext);

   set_TRIS_C (0b00010100);
   setup_spi(SPI_MASTER | SPI_L_TO_H | SPI_CLK_DIV_4 | SPI_XMIT_L_TO_H); //
Configura a comunicação SPI como Master, com uma atuação na borda de subida
e com uma divisão de 4 no clock
   setup_psp(PSP_DISABLED);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);
   setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);
   setup_timer_2(T2_DIV_BY_1, 255, 1);

   cont1=2;
   rotacao =10;
   flag14 = 1;
   flag23 = 1;
   BICO1 = 0;
   BICO2 = 0;
   BICO3 = 0;
   BICO4 = 0;
   TX = 0;
   inj=0;
   inj2=0;
   RS = 0;
   RX = 0;
   dente_referencia14 = 2;
   dente_referencia23 = 2;

 //Preenchimento dos vetores de dente de referência

   dente14[0]=12; dente14[1]=12; dente14[2]=12; dente14[3]=11;
   dente14[4]=11; dente14[5]=10; dente14[6]=10;
   dente14[7]=9; dente14[8]=9; dente14[9]=9; dente14[10]=8;
   dente14[11]=7; dente14[12]=7; dente14[13]=6;
   dente14[14]=5; dente14[15]=5; dente14[16]=4; dente14[17]=3;
   dente14[18]=2; dente14[19]=1; dente14[20]=1;
   dente14[21]=57; dente14[22]=57; dente14[23]=56; dente14[24]=55;
   dente14[25]=54; dente14[26]=53; dente14[27]=52;
   dente14[28]=51; dente14[29]=49; dente14[30]=48;


   dente23[0]=42; dente23[1]=42; dente23[2]=42; dente23[3]=41;
   dente23[4]=41; dente23[5]=40; dente23[6]=40;
   dente23[7]=39; dente23[8]=39; dente23[9]=39; dente23[10]=38;
   dente23[11]=37; dente23[12]=37; dente23[13]=36;
   dente23[14]=35; dente23[15]=35; dente23[16]=34; dente23[17]=33;
   dente23[18]=32; dente23[19]=31; dente23[20]=31;
   dente23[21]=30; dente23[22]=29; dente23[23]=27; dente23[24]=26;
   dente23[25]=25; dente23[26]=24; dente23[27]=23;
   dente23[28]=22; dente23[29]=20; dente23[30]=19;

 //Preenchimento do vetor de tempo de injeção
```

```
t_inj[0]=65000; t_inj[1]=65000; t_inj[2]=65000; t_inj[3]=64000;
t_inj[4]=65500; t_inj[5]=62100; t_inj[6]=65000;
t_inj[7]=62100; t_inj[8]=65000; t_inj[9]=65530; t_inj[10]=65008;
t_inj[11]=63700; t_inj[12]=65100; t_inj[13]=65008;
t_inj[14]=65000; t_inj[15]=65530; t_inj[16]=65100; t_inj[17]=65008;
t_inj[18]=65008; t_inj[19]=65008; t_inj[20]=65500;
t_inj[21]=63300; t_inj[22]=65008; t_inj[23]=65008; t_inj[24]=65008;
t_inj[25]=65100; t_inj[26]=65100; t_inj[27]=65300;
t_inj[28]=65300; t_inj[29]=65100; t_inj[30]=65300;

   while (true)
   {
         if(i== 0)
         {

             if (cont1 == 12)
               {
                BICO1 = 0;    /*Funciona mesmo que já esteja em zero*/
                BICO4 = 0;
               }

               else if (cont1 == 42)
               {
                BICO2 = 0;
                BICO3 = 0;
               }

               else if (cont1 == 47)
               {
               dente_referencia14 = dente14[rotacao];
               dente_referencia23 = dente23[rotacao];
               }

               else if (cont1 == dente_referencia14 && cont1 != 12)
               {
                 inj = 1;

                  set_timer1(t_inj[rotacao]);
                 enable_interrupts(global);
                  enable_interrupts(int_timer1);
               }

               else if (cont1 == dente_referencia23 && cont1 != 42)
               {
                 inj2 = 1;
                set_timer2(0);
                enable_interrupts(global);
                enable_interrupts(int_timer2);
               }
         i++;
         }
   }
}
```