ANDRÉ MINORU BEPPU

CHRISTIAN STROTTMANN KERN

DALMO GUEDES DE OLIVEIRA

SISTEMA DE GERENCIAMENTO DESCENTRALIZADO PARA A INJEÇÃO ELETRÔNICA DE UM MOTOR DE CICLO OTTO

São Paulo
2009

ANDRÉ MINORU BEPPU

CHRISTIAN STROTTMANN KERN

DALMO GUEDES DE OLIVEIRA

SISTEMA DE GERENCIAMENTO DESCENTRALIZADO PARA A INJEÇÃO ELETRÔNICA DE UM MOTOR DE CICLO OTTO

São Paulo
2009

ANDRÉ MINORU BEPPU

CHRISTIAN STROTTMANN KERN

DALMO GUEDES DE OLIVEIRA

SISTEMA DE GERENCIAMENTO DESCENTRALIZADO PARA A INJEÇÃO ELETRÔNICA DE UM MOTOR DE CICLO OTTO

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do diploma de Engenheiro Eletricista – Ênfase em Sistemas Eletrônicos

Orientador: Prof. Dr. Armando Antônio Maria Laganá

São Paulo
2009

RESUMO

Nos motores a combustão interna de ciclo Otto, amplamente utilizados na atualidade devido à sua aplicação em veículos de passeio e motocicletas, é imprescindível que se tenha um controle preciso da mistura ar/combustível presente nos cilindros para o início de cada ciclo de operação do motor. A importância desse controle está relacionada essencialmente a duas necessidades: a econômica, de obter o mais alto rendimento em função da quantidade de combustível utilizada, e a ambiental, relacionada também com a redução do consumo e com a diminuição das emissões poluentes. Nesses motores, o sistema de controle de dosagem de combustível de uso predominante atualmente é o sistema de injeção eletrônica, em que o gerenciamento das tarefas realizadas ao longo do ciclo de operação do motor é tipicamente implementado centralmente, por meio de uma ECU (*Electronic Control Unit*, Unidade de Controle Eletrônico). Este trabalho propôs a implementação de um sistema de gerenciamento descentralizado para um sistema de injeção eletrônica de combustível de um motor de ciclo Otto. A realização do projeto partiu de protótipos iniciais dos módulos de ignição, injeção e válvula borboleta, de um módulo com os sensores utilizados pelo sistema de injeção eletrônica e de um motor elétrico adaptado para simular um motor de combustão interna, pretendendo aperfeiçoar os protótipos dos subsistemas já existentes (com ênfase no software dos módulos que compõem estes subsistemas), e ainda o desenvolvimento do subsistema de macro-gerenciamento, objetivando-se a consolidação de um sistema de injeção eletrônica com gerenciamento descentralizado.

ABSTRACT

In the Otto cycle internal combustion engines, widely used today thanks to their application in automobiles and motorcycles, it is essential to obtain a precise control over the air/fuel mixture contained in the cylinders in the beginning of each cycle of the engine's operation. The importance of this control is essentially related to two necessities: the economic, in obtaining the highest efficiency for a given amount of fuel, and the environmental, also related with the reduction of fuel consumption and with the reduction of polluting emissions. In these engines, the fuel dosage control system in predominant use today is the electronic injection system, in which the management of tasks performed throughout the engine's operating cycle is typically implemented centrally by an ECU (Electronic Control Unit). This work proposed the implementation of a decentralized management system for the electronic fuel injection system of an Otto cycle engine. The implementation of the project started from initial prototypes of the ignition, injection and throttle modules, of a module containing the sensors used by the electronic injection system and of an electric motor adapted in order to simulate an internal combustion engine, with the intent to improve the existing subsystem prototypes (with emphasis in the software for the modules that compose these subsystems), plus the development of the macro-management subsystem, aiming to consolidate an electronic injection system with decentralized management.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CTS	Coolant Temperature Sensor
ECU	Electronic Control Unit
EGO	Exhaust Gas Oxygen
IAC	Idle Air Control
IAT	Intake Air Temperature
MAF	Mass Air Flow
MAP	Manifold Air Pressure
PMI	Ponto Morto Inferior
PMS	Ponto Morto Superior
PWM	Pulse Width Modulation
TPS	Throttle Position Sensor

SUMÁRIO

1 Introdução

Dentre as diversas máquinas térmicas desenvolvidas ao longo da história, uma das mais utilizadas atualmente é o motor a combustão interna. Seu funcionamento, como o de qualquer outra máquina térmica, se baseia em ciclos termodinâmicos, envolvendo etapas de compressão, expansão e troca de calor entre fluidos. No caso particular dos motores a combustão interna, os próprios gases da combustão são utilizados como fluido de trabalho, sendo processados em ciclo a fim de realizar a conversão de energia na qual o funcionamento do motor pode ser resumido.

Existem diversos ciclos termodinâmicos em que o funcionamento de um motor a combustão interna pode se basear. Como exemplos, pode-se citar o ciclo Otto (caso dos motores mais frequentemente empregados em automóveis de passeio e motocicletas), o ciclo Diesel (caso dos motores utilizados em veículos de carga, como ônibus e caminhões) e o ciclo Brayton (caso das turbinas a gás, utilizadas na propulsão de aeronaves e em geradores elétricos industriais). Neste trabalho, será dado enfoque sobre os motores de ciclo Otto, cujas características serão detalhadas mais à frente.

Os motores a combustão interna tem seu funcionamento fortemente afetado pela relação ar/combustível no interior da câmara de combustão. Para que se consiga obter a melhor operação, sob um critério qualquer (por exemplo, maior eficiência ou maior potência específica), é imprescindível que se estabeleça um mecanismo de controle preciso para a dosagem de combustível. Atualmente, a importância desse controle está relacionada principalmente a duas necessidades: a necessidade econômica, de se obter a melhor eficiência de modo a reduzir o consumo de combustível, e a ambiental, no sentido também de reduzir o consumo e ainda de reduzir as emissões de poluentes que decorrem naturalmente da operação desses motores.

Os sistemas automotivos de injeção eletrônica produzidos em escala industrial são todos gerenciados centralmente por uma ECU (*Electronic Control Unit*, Unidade de Controle Eletrônico). Este trabalho propõe a implementação de um sistema de gerenciamento descentralizado para um sistema de injeção eletrônica de

combustível de um motor de ciclo Otto. Sob essa abordagem, cada um dos módulos que compõem o sistema de injeção é implementado separadamente, e dotado da capacidade requerida para o controle de uma tarefa específica do ciclo de operação do motor. A integração desses módulos é realizada através de um módulo adicional, de macro-gerenciamento, responsável essencialmente pela geração de parâmetros e pela sincronização das tarefas executadas pelos demais módulos, de maneira a se obter um sistema que opere de forma harmoniosa ao longo de todo o ciclo do motor.

1.1 Motivações para explorar um sistema descentralizado

O sistema proposto neste trabalho, no qual o gerenciamento da injeção eletrônica é descentralizado, estabelece um novo paradigma na concepção e operação dos sistemas de injeção eletrônica. Com esse novo paradigma, abre-se portas para outros avanços científicos na área de controle de motor a combustão, com potencial para impactar de forma positiva a indústria automotiva.

A título de exemplo das possibilidades proporcionadas pelo sistema proposto, temos o fato de que um sistema modular em geral pode ser expandido de forma muito mais natural e conveniente do que um sistema integrado. Assim, o sistema proposto facilita o acréscimo de novos componentes (novos sensores, por exemplo) que permitam aprimorar o seu funcionamento. Mais do que isso, num contexto de evolução e disseminação das tecnologias de comunicação sem fio, não é difícil imaginar que a integração entre os módulos possa no futuro fazer uso de uma interface de comunicação sem fio, de forma que a modularização do sistema viabilize inclusive a sua extensão para partes de um automóvel que tipicamente não se encontram próximas do sistema de injeção eletrônica. Em um sistema monolítico, tal extensão exigiria a ampliação da fiação (ou *chicote*, como é chamada no contexto automotivo) do veículo, o que é extremamente indesejável (particularmente na indústria automotiva, em que tradicionalmente se observa níveis de competitividade muito agressivos) por aumentar a complexidade do sistema e introduzir novos pontos de falha.

Além disso, a modularização pode ser vista como vantajosa por simplificar o projeto do sistema e a realização de aperfeiçoamentos futuros, já que cada módulo

passa a ter uma função bem determinada e sua complexidade pode ser consideravelmente reduzida. Da mesma forma, o sistema torna-se uma boa plataforma para a implementação e testes de novas tecnologias que atuem sobre o motor de ciclo Otto. No momento atual, em que se percebe uma forte procura por combustíveis alternativos que permitam reduzir as emissões, um sistema que facilite o trabalho de implementar os ajustes necessários para que se explore diferentes possibilidades de combustíveis novamente se destaca, podendo servir de base para sistemas de injeção específicos de combustíveis que venham a ser utilizados futuramente.

Outro aspecto dos sistemas de injeção eletrônica centralizados, como será explicado mais adiante, é que estes sistemas fazem uso de parâmetros que tradicionalmente são obtidos por mapeamento, isto é, são armazenados em tabelas na memória da ECU e de lá são obtidos em tempo real para a operação do sistema. Com a descentralização, uma vez que cada módulo fica responsável por algumas poucas e bem determinadas tarefas, torna-se possível dedicar parte do processamento de cada módulo à implementação de tecnologias "inteligentes", em que o aprendizado de máquina exerça um papel definitivo no ajuste de parâmetros armazenados na memória para cada módulo. Por exemplo, no módulo de ignição poderia ser implementada uma rede neural que determinasse constantemente o avanço ótimo do ponto de ignição para os diferentes regimes do motor, modificando a tabela de parâmetros desse módulo ao longo do tempo de vida do motor de modo a refletir, por exemplo, o desgaste natural das peças ou as variações na composição do combustível utilizado. Essa capacidade de auto-ajuste do sistema se refletiria em uma eficiência ainda maior do motor, reduzindo o consumo e as emissões de poluentes.

Assim, o paradigma de descentralização inerente ao sistema que desenvolvemos apresenta inúmeras vantagens sobre o modelo de controle centralizado utilizado atualmente pela indústria, e viabiliza avanços científicos que dificilmente ocorreriam nos sistemas já existentes. Por esses motivos, acreditamos que o modelo proposto neste trabalho apresenta potencial para se tornar uma tendência no setor automotivo, o que justifica o interesse em desenvolver um sistema que implemente esse novo modelo para os sistemas de injeção eletrônica.

1.2 Objetivos

O objetivo geral deste projeto foi a implementação de um sistema de gerenciamento descentralizado para a injeção eletrônica de um motor de ciclo Otto. Como explicaremos adiante, o projeto partiu de protótipos dos principais módulos funcionais de um sistema de injeção eletrônica, visando inicialmente consolidá-los para então desenvolver o módulo de macro-gerenciamento do sistema. Por fim, integrou-se os diversos módulos em um sistema de injeção eletrônica funcional e completo.

Assim, como objetivos específicos do projeto podemos mencionar:

- O aprimoramento dos circuitos de interface dos módulos de ignição, injeção e da válvula borboleta;
- O aperfeiçoamento dos softwares empregados em cada um desses módulos, de forma a consolidá-los como subsistemas independentes;
- O desenvolvimento do módulo de macro-gerenciamento, responsável pela geração de parâmetros, pela integração e pela sincronização dos demais módulos do sistema durante sua operação;
- A consolidação do sistema de injeção eletrônica com gerenciamento descentralizado, conectando e integrando todos os módulos e realizando os ajustes necessários em cada um deles para que funcionem em conjunto da melhor maneira possível;
- A elaboração de uma documentação compreensiva sobre todas as etapas do projeto.

2 Contextualização e conceitos

2.1 O motor de Ciclo Otto

O ciclo Otto é um ciclo termodinâmico, ilustrado no diagrama Pressão x Volume da Figura 1, que consiste nos seguintes processos:

1. Admissão isobárica (0 – 1);
2. Compressão adiabática (1 – 2);
3. Combustão isocórica (2 – 3);
4. Expansão adiabática (3 – 4);
5. Redução de pressão isocórica (4 – 5);
6. Exaustão isobárica (5 – 0).

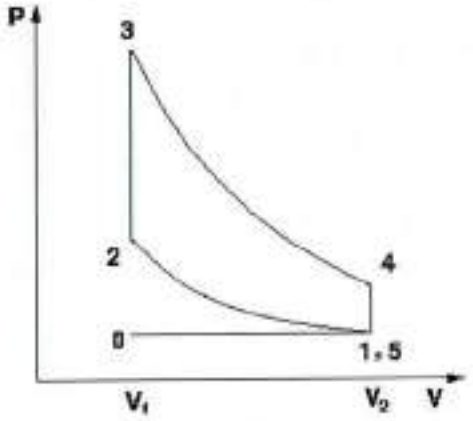


Figura 1. Diagrama PxV do Ciclo Otto. [1]

A implementação de uma máquina térmica baseada no ciclo Otto normalmente se dá na forma de motores alternativos, ou seja, motores em que as diferentes etapas do ciclo ocorrem pelo deslocamento repetitivo de pistões ao longo de cilindros. Tais máquinas são ainda classificadas como motores de quatro tempos, sendo que os quatro tempos se referem aos quatro deslocamentos do pistão que realizam as diferentes etapas do ciclo: admissão, compressão, expansão e exaustão.

De forma resumida, e focando-se no comportamento observado para um único cilindro do sistema, as etapas do funcionamento de um motor de ciclo Otto podem ser descritas como segue:

1. Admissão da mistura ar/combustível, através do deslocamento do pistão desde o ponto morto superior (PMS) até o ponto morto inferior (PMI) do cilindro, realizado com a válvula de admissão aberta;
2. Compressão da mistura, através do deslocamento do pistão desde o PMI até o

PMS com ambas as válvulas, de admissão e de exaustão, fechadas;

3. Produção de uma faísca de modo a iniciar a combustão da mistura, da qual decorre um aumento na pressão interna do cilindro que por sua vez impulsiona o pistão desde o PMS até o PMI;
4. Exaustão dos gases resultantes da combustão, através do deslocamento do pistão do PMI até o PMS com a válvula de exaustão aberta.

Deve-se observar que os motores de ciclo Otto reais podem possuir mais de um cilindro, e que estes podem ainda estar defasados entre si em relação ao ciclo de operação (por exemplo, em um motor de quatro cilindros, se em um dado instante um dos pistões está iniciando a etapa de admissão temos tipicamente os outros três iniciando, respectivamente, as etapas de compressão, expansão e exaustão).

Deve-se notar, além disso, que na etapa de expansão é que ocorre efetivamente o processo de conversão de energia pelo motor, visto que nesta etapa a combustão da mistura ar/combustível, que impulsiona o pistão, corresponde de fato à transformação da energia química armazenada no combustível em energia mecânica cinética correspondente ao deslocamento do pistão.

Outro ponto característico dos motores de ciclo Otto é a forma de ignição da mistura, a qual se dá através da produção de uma centelha na câmara de combustão. Esta é uma das principais diferenças desses motores em relação aos motores de ciclo Diesel, nos quais ocorre a auto-ignição da mistura, sem centelha, ocasionada tão somente pelo aumento da pressão no cilindro decorrente da etapa de compressão do ciclo de operação do motor.

2.2 Breve histórico da injeção eletrônica

Historicamente, dois tipos de sistemas de dosagem de combustível foram empregados com sucesso em motores de ciclo Otto: os carburadores, baseados na sucção de combustível, e os sistemas de injeção eletrônica, baseados na injeção de combustível.

Os carburadores são sistemas puramente mecânicos, cujos elementos principais são: uma *cuba*, na qual o combustível é armazenado; um *venturi*, que

consiste de um estreitamento na tubulação de admissão de ar; e um furo calibrado, denominado *giclê*, que conecta a cuba ao venturi. O funcionamento de um carburador é extremamente sofisticado, e seu detalhamento foge ao escopo deste trabalho. De forma simplificada, o venturi produz uma depressão na tubulação do coletor de ar, criando um diferencial de pressão entre a tubulação e a cuba (que se encontra à pressão atmosférica). Em decorrência desse diferencial de pressão, o combustível contido na cuba é aspirado, através do giclê, para o interior do coletor de ar. A calibração do giclê permite então regular a relação ar/combustível da mistura admitida pelo motor.

Uma das principais limitações do carburador se deve ao fato de a dosagem de combustível ser determinada basicamente pela abertura do giclê. Uma vez que o giclê esteja calibrado, não se consegue realizar um ajuste fino sobre a relação ar/combustível em tempo real, isto é, durante a operação do motor. Com isso, e considerando-se os diferentes regimes de operação por que um motor tipicamente passa durante seu funcionamento, o carburador não oferece a flexibilidade necessária para que se consiga operar sempre no ponto ótimo de trabalho para cada um desses diferentes regimes.

Apesar disso, os carburadores foram utilizados de maneira predominante na indústria até a década de 1980, quando a necessidade de atender à rígida legislação ambiental nos Estados Unidos e Japão, inicialmente, e posteriormente na Europa, tornou os sistemas eletrônicos baseados na injeção de combustível a única alternativa viável para atender às novas exigências referentes ao controle de emissões, graças à maior precisão do controle sobre a combustão proporcionada por esses sistemas. Antes disso, entre as décadas de 50 e 70, haviam sido feitas aplicações pontuais de sistemas de injeção – inicialmente mecânicos, mas estes rapidamente evoluíram para os sistemas eletrônicos –, com o objetivo principal de aumentar a potência específica dos motores.

No Brasil, a mudança efetiva dos sistemas carburados para os injetados se deu na década de 90, e atualmente todos os automóveis de passeio lançados no mercado brasileiro contam com um sistema de injeção eletrônica. As motocicletas, por sua vez, ainda possuem em grande parte motores carburados, por serem tipicamente veículos de menor complexidade e menor custo, e também por terem sofrido no passado menor pressão por parte dos órgãos reguladores. Entretanto, nos últimos anos tem se observado consistentemente a adoção de sistemas de injeção

eletrônica também nesses veículos, de modo que hoje muitas das motocicletas de alto desempenho lançadas no mercado já apresentam motores injetados, e também nas de baixo custo os carburadores estão sendo substituídos pelos sistemas de injeção eletrônica, principalmente porque as legislações ambientais vêm se mostrando cada vez mais exigentes também em relação a esses veículos.

2.3 A importância dos sistemas de injeção eletrônica

Hoje, a importância dos sistemas de injeção eletrônica diz respeito principalmente a duas questões: a questão econômica e a questão ambiental.

A questão econômica está relacionada ao fato de os sistemas de injeção eletrônica permitirem que motores de ciclo Otto operem sempre com maior eficiência do que os carburados, reduzindo o desperdício de combustível. Por esse motivo, os motores com injeção eletrônica se mostram mais econômicos, representando ao usuário um menor gasto com combustível.

Quanto à questão ambiental, aqui também cabe destacar a importância da operação mais eficiente dos motores injetados, uma vez que a redução no consumo de combustíveis, fósseis ou não, é hoje vista como uma importante diretiva ambiental a ser seguida. Os combustíveis mais freqüentemente utilizados nos motores de ciclo Otto são a gasolina e o etanol. No caso do primeiro, a importância de se reduzir o consumo diz respeito ao fato de ser um combustível derivado de petróleo, este último um recurso esgotável da natureza. No caso do segundo, embora se trate de um recurso renovável, de origem vegetal, há a consciência de que os recursos vegetais utilizados para a produção do etanol poderiam ser utilizados, quando não para consumo humano, para outros fins igualmente importantes, como a compostagem para a produção de adubo com alto teor energético. Além da redução do consumo, outro aspecto dos motores com injeção eletrônica, também ligado à questão ambiental e igualmente de extrema importância, é a redução das emissões poluentes, contribuindo para a redução do aquecimento global e da poluição atmosférica de maneira geral.

Esses aspectos da importância dos sistemas de injeção eletrônica são ainda potencializados pelo fato de serem sistemas amplamente difundidos hoje em

dia. A frota mundial de automóveis vem crescendo de forma ininterrupta e significativa nos últimos anos, e portanto a importância de uma eventual melhoria nesses sistemas é drasticamente amplificada pela sua popularidade e ampla aplicação a nível global.

Diante dos aspectos – econômico e ambiental – destacados, cabe ressaltar que, embora a motivação inicial para o desenvolvimento dos sistemas de injeção eletrônica tenha sido a possibilidade de melhorar o desempenho dos motores, foi o potencial desses sistemas, em termos do possível impacto ambiental proporcionado por eles, o real responsável por consagrá-los como a tecnologia mais difundida hoje para o controle da dosagem de combustível nos motores automotivos.

3 Os sistemas de injeção eletrônica tradicionais

Os sistemas de injeção eletrônica surgiram inicialmente como alternativa aos carburadores, tendo o propósito de controlar a injeção de combustível nos motores. Entretanto, graças à facilidade de integração proporcionada pelos sistemas eletrônicos, acabou-se concentrando diversas funcionalidades de controle e monitoramento do motor no sistema, ampliando suas funções no automóvel. Hoje, os sistemas de injeção eletrônica podem ser considerados sistemas eletrônicos complexos, formados por múltiplos sensores e atuadores, e responsáveis pelo gerenciamento do motor.

Do ponto de vista do número de bicos injetores, os sistemas de injeção eletrônica podem ser classificados em sistemas *monoponto* (com apenas um bico injetor para todos os cilindros) e *multiponto* (com um injetor para cada cilindro). Do ponto de vista do posicionamento dos injetores, estes últimos podem ser classificados em sistemas de *injeção direta* (quando injetam combustível diretamente dentro da câmara de combustão) e de *injeção indireta* (quando injetam combustível no tubo coletor de ar, próximo às válvulas de admissão). Os sistemas monoponto são sempre de injeção indireta.

Na Figura 2, pode-se observar alguns dos principais componentes de um sistema de injeção eletrônica multiponto, assim como sua estrutura geral. O funcionamento de um sistema como esse será brevemente explicado a seguir.

Nos sistemas de injeção eletrônica tradicionais, uma única central de controle eletrônico (*ECU, Electronic Control Unit*, também chamada de *centralina*, indicada na Figura 2 pelo número 11) realiza todo o controle eletrônico do sistema, recebendo os sinais dos diversos sensores e enviando sinais para os diversos atuadores do sistema. Em uma memória da ECU ficam armazenadas diversas tabelas de referência, utilizadas pela ECU para a definição dos parâmetros do sistema.

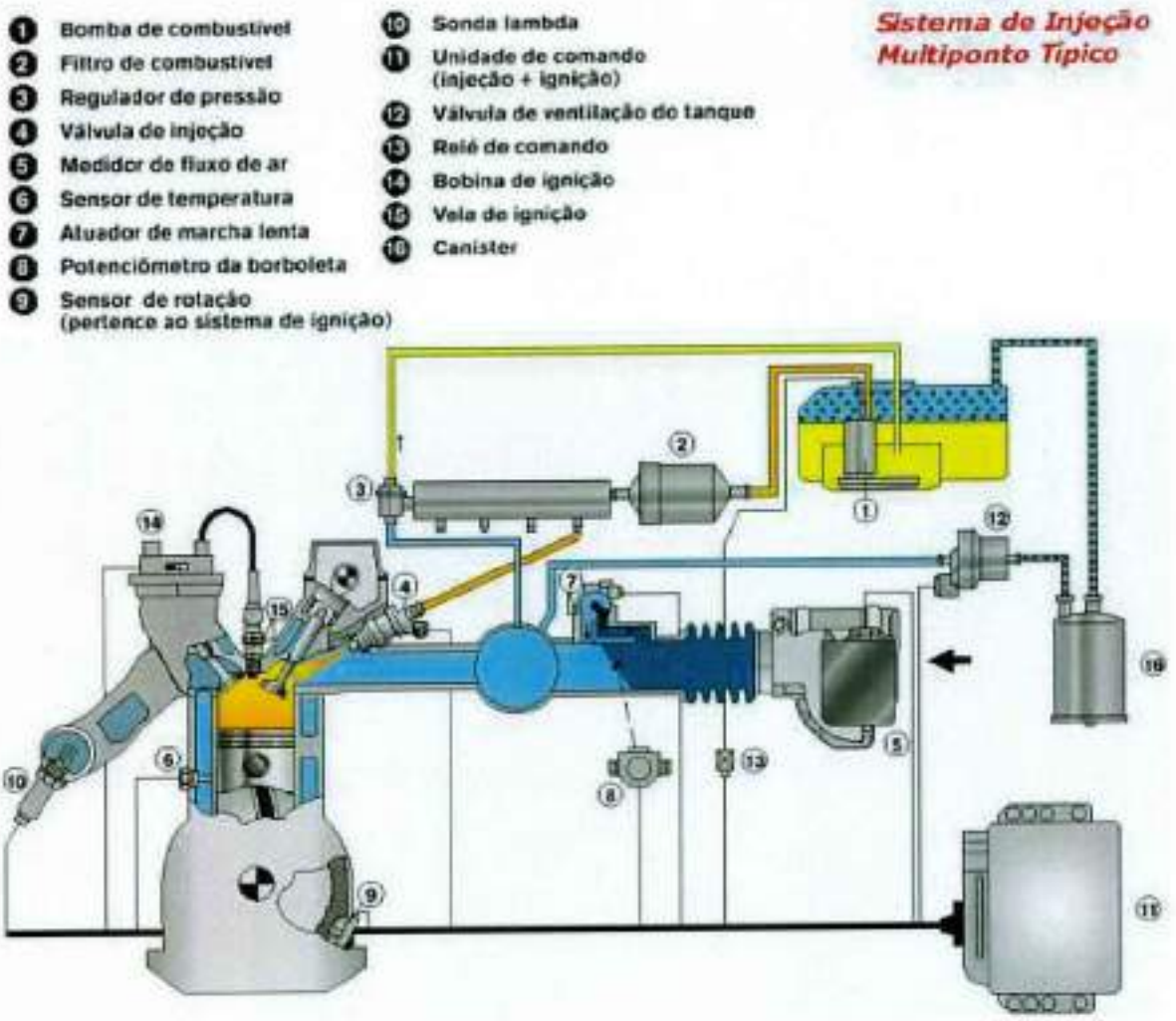


Figura 2. Estrutura e principais componentes de um sistema de injeção multiponto tradicional. [2]

Na tubulação de admissão de ar, cuja entrada aparece apontada na figura por uma seta preta, fica instalada a *válvula borboleta*, ou *válvula de aceleração* (que, na figura, separa a tubulação de admissão de ar em duas regiões, uma em azul claro e a outra em azul escuro), responsável por regular a quantidade de ar admitida no motor de acordo com as suas condições de aceleração. À válvula borboleta ficam conectados um sensor de posição (denominado *TPS, Throttle Position Sensor* – essencialmente um potenciômetro, indicado na figura pelo número 8) e um motor de passo, através dos quais a ECU controla a entrada de ar do sistema. Em alguns motores, uma válvula (chamada *válvula IAC, Idle Air Control*) é instalada em paralelo à borboleta especificamente para o controle da entrada de ar em marcha lenta (isto é, quando o motor não está sendo acelerado). É o caso do sistema ilustrado na Figura 2, em que a válvula IAC aparece indicada pelo número 7. Neste trabalho, entretanto, assumiremos um motor que não possui uma válvula específica para esse

fim, de modo que a própria válvula borboleta será comandada pelo motor de passo para a regulagem de marcha lenta.

Um procedimento importante para que se possa determinar a correta dosagem de combustível é a medição da massa de ar admitida pelo motor. Tipicamente, são instalados na tubulação de admissão de ar sensores de pressão absoluta (*MAP, Manifold Absolute Pressure*) e de temperatura do ar no coletor (*IAT, Intake Air Temperature*), cujas informações são utilizadas pela ECU para estimar a massa de ar admitida. Em alguns sistemas mais modernos, há um único sensor que realiza diretamente a medição de massa de ar (*MAF, Mass Air Flow*), evitando a necessidade de a ECU realizar estes cálculos. No sistema da Figura 2, é o bloco indicado pelo número 5 o responsável por realizar essa medição.

O combustível, que fica armazenado no tanque de combustível, é bombeado pela *bomba de combustível* (indicada pelo número 1 na figura), passando pelo *filtro de combustível* (indicado pelo número 2) e chegando ao *tubo distribuidor*. Um *regulador de pressão* (indicado na figura pelo número 3) se encarrega de fornecer um caminho para que parte do combustível possa retornar ao tanque, de modo a manter a pressão na *linha de combustível* constante. Com essa premissa, de que a pressão do combustível na linha será mantida constante, torna-se possível regular a quantidade de combustível injetada controlando-se unicamente o tempo de abertura dos *bicos injetores* (um dos quais aparece na figura indicado pelo número 4; incidentalmente, observa-se pelo posicionamento do injetor na Figura 2 que o sistema ilustrado é um sistema de injeção indireta).

Pode-se observar ainda na Figura 2, no corte abaixo do pistão, parte da *roda-fônica* – uma roda dentada que, conectada ao eixo do motor, possui uma falha de referência, a qual em conjunto com um *sensor de efeito Hall* (indicado pelo número 9) permite a medição da velocidade de rotação do motor e a sincronização das diversas ações de controle ao longo do ciclo de operação.

Por fim, temos indicada pelo número 15 na figura a *vela de ignição*, responsável pela produção da centelha que inicia a combustão da mistura, e os sensores de temperatura do líquido de arrefecimento do motor (*CTS, Coolant Temperature Sensor*, indicado pelo número 6) e de oxigênio na tubulação de exaustão (a chamada *sonda lambda*, ou ainda *sensor EGO, Exhaust Gas Oxygen*, indicada pelo número 10).

Todos esses sensores e atuadores estão conectados à ECU através do

barramento indicado por um traço mais largo na parte inferior da figura, e viabilizam à ECU estabelecer o gerenciamento completo sobre a operação do motor. Em particular, o sensor da roda-fônica, a sonda lambda e o sensor da válvula borboleta fornecem informações acerca do regime de operação do motor (por exemplo: marcha lenta, aceleração, desaceleração, meia carga, plena carga, etc.). Com essas informações, e ainda através do cálculo da massa de ar viabilizado pelos sensores MAF, MAP e IAT, o tempo de abertura dos injetores pode ser obtido pela ECU através de consultas a tabelas e de cálculos de correção sobre os valores consultados. Além disso, com base no regime de operação também é obtido, por consulta a tabelas e eventuais cálculos de correção, o ângulo de avanço da ignição (já que a ignição deve ser acionada com um certo adiantamento em relação ao PMS, para dar conta do tempo de propagação da centelha quando do início da combustão).

Por fim, cabe ressaltar que o sistema apresentado é bastante simples, apresentando somente os principais recursos de um sistema de injeção eletrônica. Em muitos dos sistemas comercialmente disponíveis são acrescentados outros sensores e atuadores, de modo a aprimorar ainda mais o funcionamento do motor. A título de exemplo, pode-se mencionar os sensores de detonação presentes em muitos sistemas comerciais, que visam detectar o fenômeno da *detonação* (que ocorre quando o ângulo de avanço da ignição é exagerado, fazendo com que a pressão máxima no interior do cilindro seja atingida antes do PMS – o que produz forças contrárias ao movimento do motor, podendo provocar danos permanentes ao sistema), e também as válvulas de recirculação de gás da exaustão (*EGR, Exhaust Gas Recirculation*), que permitem redirecionar parte dos gases resultantes da combustão de volta para a entrada de ar do motor, proporcionando a redução da emissão de alguns poluentes, notavelmente os óxidos de nitrogênio (*NOx*).

4 Métodos e materiais

4.1 Metodologia

A metodologia empregada neste trabalho consistiu no desenvolvimento de cada módulo funcional do motor separadamente, e na posterior conexão dos módulos através de um módulo adicional de macro-gerenciamento.

Como já mencionamos, o trabalho em que este projeto se insere já havia sido iniciado no Laboratório de Sistemas Integráveis, sob a coordenação do Prof. Dr. Armando Antônio Maria Laganá, de modo que já existiam inicialmente no Laboratório protótipos para os subsistemas de injeção, de ignição e da válvula borboleta. Cada módulo compreende hardware e software específicos para as funções que se deseja implementar, e podemos considerar que quando completamente desenvolvidos os módulos correspondem aos diferentes subsistemas que devem ser integrados para que se obtenha um sistema de injeção eletrônica completo e funcional.

O trabalho foi dividido em etapas da seguinte forma:

1. Inicialmente, trabalhamos sobre cada um dos protótipos já existentes, a nível de hardware e de software, de modo a obter sucessivamente os subsistemas consolidados da injeção, da ignição e da válvula borboleta (que constituem os *subsistemas funcionais* do projeto).
2. Em seguida, foi desenvolvido o software para implementar a interface de comunicação entre os subsistemas, o qual foi então integrado a cada um dos subsistemas funcionais.
3. Foi então desenvolvido o módulo de macro-gerenciamento, com o auxílio dos subsistemas funcionais já obtidos.
4. Por fim, realizou-se a integração entre todos os subsistemas, de forma a operarem paralelamente e de forma harmoniosa.

A metodologia proposta permitiu que os subsistemas funcionais fossem testados individualmente, garantindo-se que cada um deles estivesse implementando corretamente as atividades que lhe cabem. Além disso, a interface de comunicação pôde ser desenvolvida separadamente, simplificando seus testes, e

uma vez funcionando corretamente esta pôde ser integrada em todos os subsistemas funcionais, já com suas funcionalidades específicas desenvolvidas.

Para o desenvolvimento do módulo de macro-gerenciamento, já possuíamos então os subsistemas funcionais consolidados, implementando corretamente as suas funcionalidades individuais e já dotados da interface de comunicação que seria utilizada na integração do sistema final. Dessa forma, os subsistemas desenvolvidos auxiliaram no processo de desenvolvimento do módulo de macro-gerenciamento, podendo ser utilizados individualmente ou em conjunto conforme as necessidades de implementação e desenvolvimento de cada uma das funcionalidades do módulo de macro-gerenciamento.

Para a comunicação entre os subsistemas, implementou-se uma interface I^2C. O protocolo I^2C, desenvolvido pela Philips em meados da década de 1980, é um protocolo de comunicação entre circuitos integrados extremamente simples, constituído fisicamente de apenas um par de fios. Consideramos a simplicidade do I^2C sua maior virtude, e ainda que não se trate de um protocolo com a robustez necessária para aplicações automotivas de alta confiabilidade é a interface que utilizamos para o desenvolvimento por introduzir no sistema um *overhead* praticamente nulo. Após o desenvolvimento inicial, se houvesse tempo, estudaríamos a possibilidade de substituir a interface I^2C por uma de maior robustez (por exemplo, uma interface CAN). Para o desenvolvimento inicial, consideramos o I^2C apropriado, por habilitar a comunicação de forma simples, permitindo ao mesmo tempo que o foco de desenvolvimento e de processamento esteja na implementação das funcionalidades específicas de cada um dos subsistemas.

Finalmente, a documentação foi elaborada ao longo de todo o trabalho, seguindo a atividade que estava sendo realizada em cada momento. Ao final, foi feita a consolidação da documentação na forma desta Monografia Final de Projeto.

4.2 Recursos utilizados

Cada um dos protótipos já existentes possui uma placa de desenvolvimento McLab2, produzida pela empresa LabTools, com um microcontrolador PIC16F877A, responsável por gerenciar a operação de uma tarefa específica no ciclo do motor. Além disso, cada módulo compreende um componente mecânico (as velas de ignição, os bicos injetores ou a válvula borboleta), e uma placa de circuito impresso desenvolvida especificamente para o interfaceamento da placa de desenvolvimento com o componente mecânico em questão. Assim são constituídos os módulos de ignição, de injeção e da válvula borboleta.

O módulo de macro-gerenciamento consistirá em uma placa de desenvolvimento McLab2 adicional (já disponível no laboratório), também com um microcontrolador PIC16F877A, a qual será conectada diretamente às outras placas de desenvolvimento através do barramento I^2C. Este módulo tem como função interagir com os demais microcontroladores a fim de transferir parâmetros e garantir uma operação síncrona e harmoniosa do sistema como um todo.

A programação dos microcontroladores está sendo feita através do programador ICD2, também produzido pela empresa LabTools, o qual pode ser conectado diretamente às placas de desenvolvimento, facilitando as etapas de desenvolvimento e teste de software. O desenvolvimento do software dos módulos, por sua vez, está sendo feito sobre o ambiente de desenvolvimento MPLAB v8.30, fornecido pela própria Microchip (fabricante do microcontrolador), integrado ao compilador para linguagem C da empresa CCS.

Assim, as ferramentas utilizadas para o desenvolvimento do software permitem a programação em linguagem C, de alto nível, possibilitando maior agilidade no desenvolvimento, assim como a produção de código de mais simples manutenção. Ao mesmo tempo, o ambiente utilizado permite a inspeção do código *assembly* produzido pela compilação, e o compilador suporta a inserção de instruções *assembly* "em linha", de modo que, se houver necessidade de otimizar algum trecho de código, é possível reescrever somente o trecho de interesse diretamente em linguagem *assembly*, o que possibilita um maior controle sobre as instruções que serão efetivamente realizadas pelo microcontrolador. O ambiente possui ainda um *debugger* funcional e compatível com o circuito programador

utilizado, permitindo a inspeção da memória do microcontrolador em tempo de execução para auxiliar na solução de eventuais problemas que venham a surgir no desenvolvimento do software.

A infra-estrutura disponível no laboratório consiste em um módulo com os sensores utilizados pelo sistema de injeção eletrônica e um motor elétrico adaptado para simular um motor de ciclo Otto, além dos protótipos iniciais dos módulos de ignição, injeção e válvula borboleta, conforme citamos anteriormente.

Assim, em cada etapa do desenvolvimento, o motor elétrico e o módulo com os sensores serão utilizados para simular a operação de um motor a combustão em diferentes regimes, estabelecendo o contexto a partir do qual cada um dos módulos poderá ser desenvolvido, adaptado e aperfeiçoado de modo a otimizar o funcionamento global do sistema.

Não se pretende realizar grandes modificações nos protótipos já existentes dos módulos de ignição, injeção e válvula borboleta. Nestes, serão realizados predominantemente aperfeiçoamentos, de modo a permitir a melhor integração possível com o módulo de macro-gerenciamento. Além disso, as adaptações nesses módulos devem ser quase exclusivamente sobre o software.

Tendo em vista que não se pretende realizar grandes modificações no hardware dos protótipos, e que o hardware do módulo de macro-gerenciamento consiste quase exclusivamente na placa de desenvolvimento, não são esperados gastos consideráveis com componentes para o projeto. Se houver necessidade de realizar alterações no hardware já disponível, pode-se antecipar (com base no hardware que já está sendo utilizado e na própria natureza do projeto) que as modificações serão simples e não envolverão componentes de alto custo.

Quanto aos equipamentos que serão utilizados, trata-se de equipamentos de bancada comuns que também já se encontram no laboratório, como fontes de tensão, osciloscópios e multímetros. Assim, não se espera que haja necessidade de grandes investimentos para o desenvolvimento deste projeto.

5 Análise de viabilidade

Para realizar o estudo de viabilidade deste trabalho, abordaremos o projeto sob os seguintes aspectos: econômico, técnico, estrutural/financeiro e temporal. Ao nos referirmos aos aspectos econômico e técnico, daremos enfoque à implementação da estratégia de modularização aqui proposta em escala industrial – ou seja, as considerações apresentadas a seguir sobre esses aspectos dizem respeito à possível adoção de um sistema como o proposto pela indústria, e não ao desenvolvimento do sistema em laboratório como se pretende realizar neste trabalho.

Do ponto de vista econômico, em termos da produção em larga escala, observamos que a modularização pode reduzir os custos de produção e desenvolvimento para a indústria, uma vez que os mesmos módulos podem a princípio ser reutilizados em projetos diferentes. Além disso, o processo de desenvolvimento e testes de cada subsistema é simplificado, devido à redução da complexidade de cada um dos subsistemas. Por outro lado, a transição dos sistemas já existentes para sistemas descentralizados pode acarretar custos de engenharia relativamente altos para a indústria, uma vez que boa parte do desenvolvimento e dos testes precisaria ser refeita "do zero".

Do ponto de vista técnico, notamos novamente que o desenvolvimento de uma visão modular poderá beneficiar o ciclo de desenvolvimento de novos produtos na indústria, ao diminuir a complexidade de cada módulo e ainda permitir a realização de testes mais exaustivos sobre cada módulo separadamente, o que aumentaria a confiabilidade do sistema. Por outro lado, é sabido que a distribuição do sistema, conseqüência natural da modularização, aumenta o número de possíveis pontos de falha, o que por sua vez reduz a confiabilidade. Não é possível prever qual será o balanço final entre estas duas tendências opostas que surgem como conseqüência da descentralização do sistema.

Em todo caso, tendo em vista as possibilidades de desenvolvimento futuro viabilizadas pelo caráter descentralizado do sistema proposto, algumas das quais já apontamos nas seções anteriores, consideramos que o desenvolvimento do projeto fica justificado, ao menos no contexto acadêmico em que nos situamos (isto é,

considerando pesquisas posteriores que poderão ser desenvolvidas tomando por base o sistema que pretendemos construir). Assim, embora não seja possível prever com precisão o rumo que a indústria tomará no futuro, acreditamos que os frutos deste projeto poderão servir de base para ainda maiores e mais importantes desenvolvimentos científicos e tecnológicos, de modo que consideramos o desenvolvimento do projeto, sob os aspectos econômico e técnico, viável.

Do ponto de vista estrutural, é importante enfatizar que, pelo menos a princípio, todo o hardware que será exigido já está disponível. Se necessário, pequenos ajustes serão feitos, o que pode acarretar a compra de alguns (poucos) componentes. Nesse caso, tendo em vista o estado atual dos protótipos e a natureza do sistema que estamos desenvolvendo, os componentes necessários dificilmente apresentarão alto custo e com boa probabilidade serão de fácil obtenção no mercado. Por fim, ressaltamos que os equipamentos de apoio ao desenvolvimento deste projeto também já estão disponíveis no laboratório. Dessas constatações, segue que, do ponto de vista financeiro, não serão necessários investimentos altos para a realização do projeto. Assim, sob o aspecto estrutural/financeiro, não encontramos impedimentos à viabilidade do projeto.

Por fim, do ponto de vista temporal, lembramos novamente que o trabalho parte de uma base já consideravelmente sólida: os protótipos dos módulos de ignição, injeção e válvula borboleta já estão com o desenvolvimento bastante avançado, necessitando nesse momento de apenas alguns ajustes para a sua consolidação. Lembramos ainda que esses ajustes são quase exclusivamente de software, possuindo implementação mais rápida do que ajustes de hardware. Consideramos que o tempo disponível para o desenvolvimento do projeto é suficiente para a execução de todas as atividades que podemos prever como necessárias, e que já foram detalhadas nas seções anteriores. Assim, acreditamos ser possível o estabelecimento e o cumprimento de prazos para todas as atividades a serem desenvolvidas, de modo que, também sob o aspecto temporal, não encontramos impedimentos à viabilidade do projeto.

Com base no exposto nos parágrafos acima, concluímos pela viabilidade do desenvolvimento do projeto proposto.

6 Atividades desenvolvidas

Em um momento inicial deste trabalho, realizou-se o estudo teórico do funcionamento geral dos motores de ciclo Otto, e em particular da atuação dos sistemas de injeção eletrônica sobre esses motores. Além disso, através do estudo da documentação e de exercícios, promoveu-se a familiarização com a placa de desenvolvimento e com o ambiente de desenvolvimento de software, assim como com as especificidades (especialmente no tocante aos periféricos) do microcontrolador PIC16F877A.

Iniciou-se ainda um estudo detalhado do hardware que compõe os circuitos de interface dos protótipos, além de pesquisas por circuitos integrados específicos para o interfaceamento da placa de desenvolvimento com os diferentes componentes mecânicos dos subsistemas funcionais.

Cabe destacar que, embora tenha havido uma preocupação em estabelecer já de início uma sólida base teórica para o desenvolvimento do projeto, inevitavelmente se faz necessária a continuidade desse estudo ao longo de todas as etapas do trabalho, aprofundando-se em detalhes do sistema conforme o aparecimento das necessidades.

Após o estudo inicial, estando já familiarizados com as ferramentas adotadas, pudemos iniciar as atividades propostas para o desenvolvimento de fato do sistema, assim como de eventuais ferramentas auxiliares que poderão facilitar as etapas seguintes deste trabalho.

Detalharemos a seguir as atividades do projeto que já foram realizadas até o presente momento.

6.1 Sinal da roda-fônica

A roda-fônica é uma roda dentada, com uma falha de referência, utilizada para a sincronização e para a temporização das atividades de gerenciamento do motor. Neste trabalho, estamos utilizando uma roda-fônica 58 + 2, isto é, que possui 58 dentes e uma falha correspondente ao espaçamento de dois dentes.

Figura 3. Visão frontal da roda-fônica.

Tipicamente, a roda-fônica fica acoplada ao próprio eixo do motor; no caso do sistema disponível no laboratório, a roda-fônica está acoplada ao eixo do motor elétrico. Para trabalhar com uma velocidade de rotação máxima de 6000 RPM, típica dos motores de ciclo Otto automotivos, e levando em consideração que o motor elétrico disponível no laboratório opera no máximo a 3000 RPM, estamos admitindo que a roda-fônica estaria, no caso deste trabalho, acoplada ao eixo do comando de válvulas (o qual apresenta uma velocidade de rotação igual à metade da velocidade do motor, uma vez que o uma volta do eixo do comando de válvulas corresponde a uma execução completa do ciclo Otto, e nesse intervalo o motor realiza duas revoluções completas).

Assim, enquanto um motor de ciclo Otto tipicamente opera na faixa de 400 a 6000 RPM, utilizaremos o motor elétrico na faixa de 200 a 3000 RPM, simulando então a rotação do eixo do comando de válvulas de um motor, com a roda-fônica acoplada justamente a este eixo.

6.1.1 Software para simulação do sinal da roda-fônica

Uma das primeiras atividades práticas realizadas neste projeto foi o

desenvolvimento de um software, para o PIC16F877A, capaz de simular o sinal gerado pela roda-fônica. O objetivo era obter uma ferramenta adicional para o desenvolvimento do restante do sistema, que produzisse um sinal semelhante ao que seria produzido pelo circuito do sensor de efeito Hall para diferentes velocidades do motor, porém sem a necessidade de operar fisicamente o motor elétrico.

Optou-se por utilizar o potenciômetro disponível nas placas de desenvolvimento como a entrada de seleção de velocidade, convertendo-se a tensão selecionada através dele (a qual se situa na faixa de 0 a 5V) em uma velocidade de rotação na faixa de 600 a 3000 RPM. Tal operação foi realizada com auxílio do conversor analógico-digital presente no microcontrolador, configurado para fornecer um valor digital de 8 bits na faixa de tensão fornecida pelo potenciômetro. De posse do valor selecionado, basta calcular a rotação correspondente, para então calcular a duração dos dentes da roda-fônica e utilizar um dos timers embutidos para produzir o sinal desejado. Esse exercício, embora simples, permitiu uma compreensão clara do caráter crítico dos atrasos de execução das instruções ao lidar com o microcontrolador, particularmente para rotações mais altas do motor.

Em uma primeira tentativa, os cálculos foram realizados em duas etapas com operações de ponto flutuante: primeiro, calculava-se a velocidade de rotação em RPM e em seguida a duração de um dente em µs. Verificou-se que o sistema não produzia o sinal desejado. Após uma breve investigação constatou-se que, embora o tempo utilizado na leitura do conversor A/D fosse de apenas 31µs, o sistema levava 1,877ms para o cálculo da rotação e mais 1,500ms para o cálculo da duração de um dente. Entretanto, rotações entre 600 e 3000 RPM correspondem a intervalos de duração de um dente de aproximadamente 833µs e 167µs, respectivamente, de modo que o tempo consumido pelas operações era muito maior do que o disponível (isto é, enquanto o microcontrolador efetuava os cálculos, deveriam ocorrer diversas transições do sinal – o que obviamente não acontecia, já que o PIC16F877A não é capaz de processar mais de uma instrução simultaneamente).

Verificou-se também que realizando os cálculos ainda em duas etapas, porém com operações de inteiros, os tempos caíam drasticamente (260µs para o cálculo da velocidade de rotação, e 263µs para o cálculo do tempo de duração de um dente), mas ainda totalizavam um intervalo grande demais para as rotações mais altas.

O cálculo foi então simplificado, ficando reduzido a apenas duas operações – uma soma e uma divisão –, fornecendo diretamente o intervalo de duração de um dente em µs a partir do valor lido do conversor A/D. O timer foi configurado para operar com resolução de 1µs, de modo que bastasse realizar as duas operações mencionadas e utilizar o resultado obtido como limiar na contagem do timer. Como o arredondamento do cálculo é inevitável, o erro obtido em relação à velocidade nominal com essa estratégia é comparável ao do primeiro caso, no qual as operações eram realizadas em ponto flutuante. Em ambos os casos, o erro fica entre 10 e 20 RPM, dependendo da velocidade de rotação selecionada. No entanto, o cálculo completo passou a levar apenas 353µs para ser realizado. Esse, mesmo sendo grande demais para que o valor seja atualizado a cada dente, já é suficiente se for aceitável atualizar a velocidade apenas uma vez a cada volta, sempre na falha da roda-fônica (na qual temos nível lógico alto por cerca de 4 vezes a duração de um dente e em seguida nível lógico baixo por cerca de 2 vezes a duração de um dente – totalizando, no pior caso, 4x167µs=668µs de nível lógico alto, suficiente para realizar com segurança as operações).

Finalmente, decidiu-se ampliar a faixa de rotações da simulação para ficar entre 300 e 3000 RPM, e os parâmetros das operações foram ajustados manualmente para oferecer a melhor aproximação para as rotações pretendidas, conforme medidas tomadas com auxílio de um osciloscópio.

Abaixo indicamos as instruções utilizadas para cada uma das possibilidades de cálculo descritas acima.

```
// Primeira tentativa: operações com ponto flutuante
rpm = 2400.0 * ((float) valor / 255) + 600.0;    // atraso: 1877us
t = (long) ((500000.0 / rpm) + 0.5);             // atraso: 1500us


// Segunda tentativa: operações com inteiros
rpm = 2400 * (valor / 255) + 600;    // atraso: 260us
t = 500000 / rpm;                    // atraso: 263us

// Terceira tentativa: operações otimizadas
taux = valor + 64;      // atraso total: 353us
taux = 53125 / taux;

// Parâmetros obtidos para faixa de rotações ampliada, após
// ajuste experimental com auxílio do osciloscópio:
taux = valor + 32;
taux = 26563 / taux;
```

6.1.2 Circuito de interface para o sinal da roda-fônica

O sinal fornecido pelo sensor de efeito Hall da roda-fônica é aproximadamente senoidal e sua amplitude varia com a velocidade de rotação do motor. Para utilizar esse sinal nos diversos módulos do sistema, convém quadrá-lo e adequá-lo aos níveis lógicos do PIC16F877A, ou seja, entre 0 e 5V.

O circuito previamente utilizado para esse fim, entretanto, fornecia um sinal de saída distorcido, o qual não era perfeitamente quadrado e estava situado entre aproximadamente -1V e 3V. Esse sinal se mostrou inadequado para as entradas do microcontrolador, visto que 3V é muito próximo do limiar de transição do nível lógico (de baixo para alto) ocasionando transições que não eram detectadas pelo sistema.

Para resolver esse problema, foi produzido um novo circuito baseado no circuito integrado (CI) LM1815. Esse componente funciona como um detector de cruzamentos com zero, emitindo em sua saída um pulso a cada vez que o sinal de entrada passa de um valor positivo de tensão para um valor negativo. A largura do pulso emitido independe da freqüência do sinal de entrada e é configurável através de um circuito RC conectado a uma das portas do CI.

No circuito construído, o sinal do sensor Hall é passado através de um amplificador operacional montado em configuração inversora e com ganho unitário, que produz uma cópia idêntica do sinal, porém invertida. Em seguida, tanto o sinal do sensor quanto essa cópia invertida são aplicados às entradas de dois LM1815, de modo que um deles produza pulsos quando o sinal do sensor passa de um valor positivo de tensão a um valor negativo, e o outro produza pulsos na situação contrária, quando o sinal passa de um valor negativo para um positivo. Por fim, esses dois sinais pulsantes são aplicados às entradas R e S de um *latch R-S*, respectivamente, de modo a reproduzir na saída do latch uma forma de onda que se assemelhe ao perfil da própria roda-fônica.

Na Figura 4 pode-se observar capturas dos principais sinais, obtidas através de um osciloscópio para a velocidade de rotação de 1200 RPM, enquanto a Figura 5 apresenta o diagrama elétrico do circuito completo.

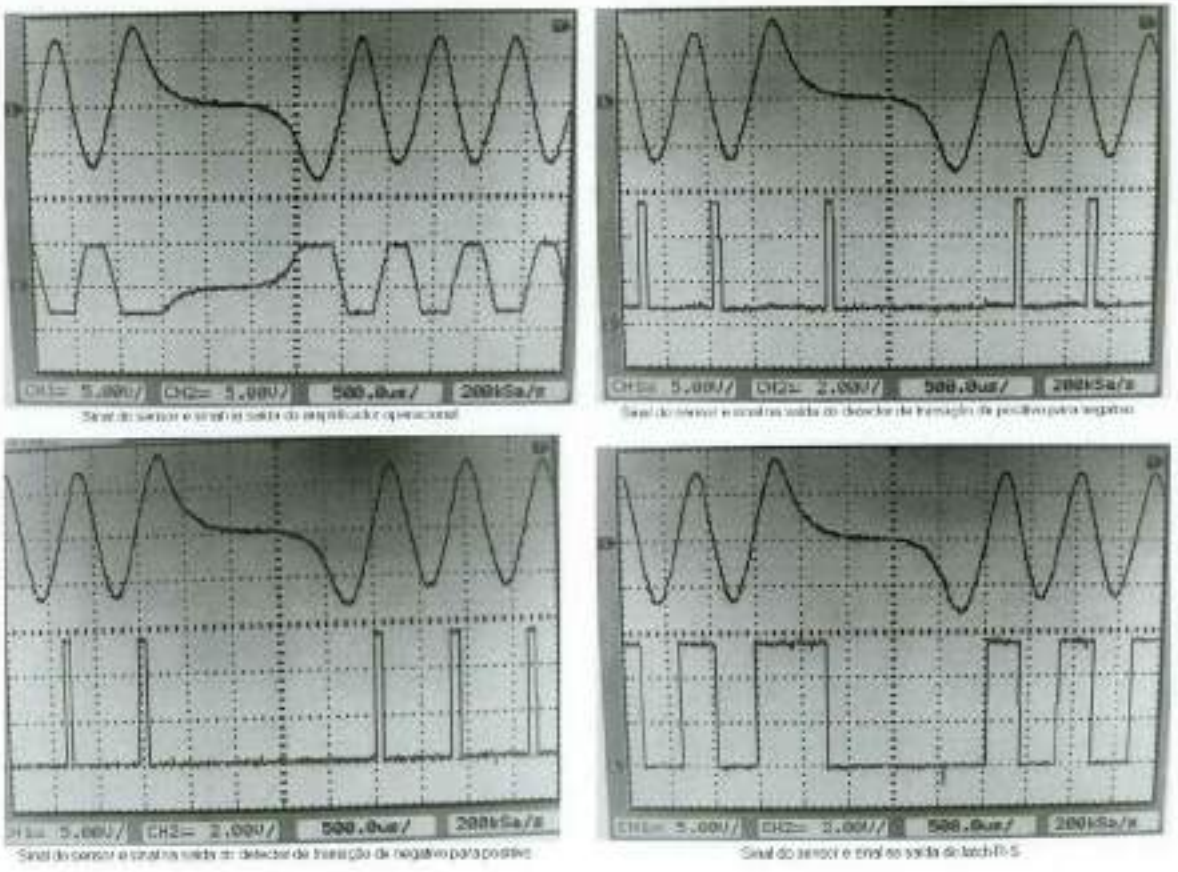


Figura 4. Formas de onda dos principais sinais do circuito da roda-fônica (1200 RPM).

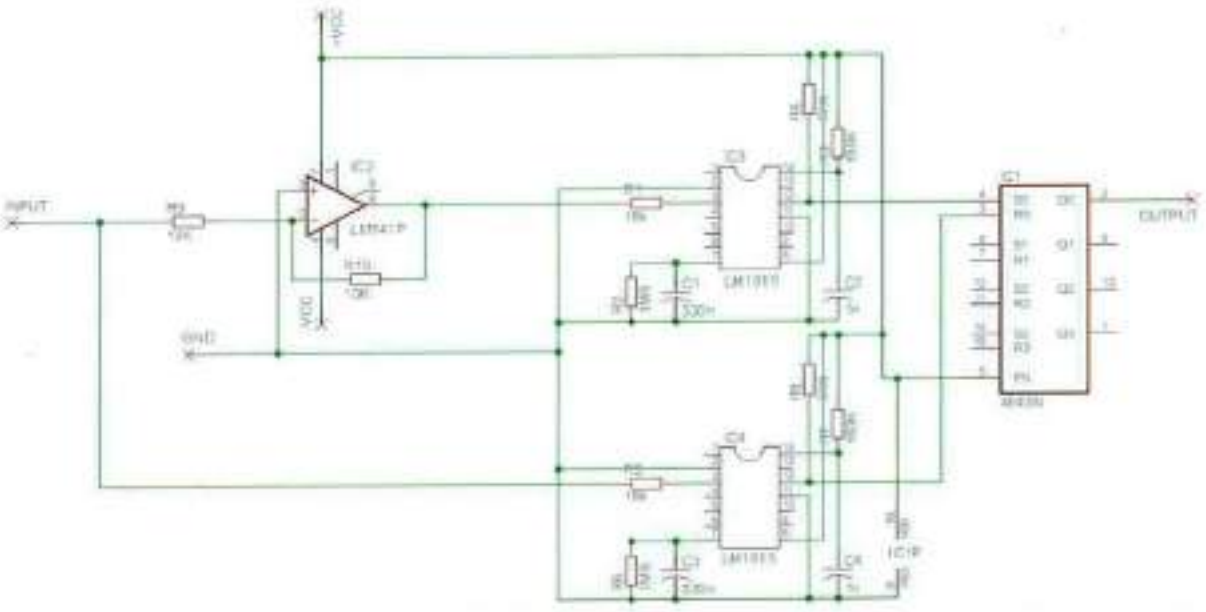


Figura 5. Diagrama do circuito da roda-fônica.

6.2 Subsistema da Ignição

A funcionalidade central do módulo responsável pela ignição se resume essencialmente no controle do ângulo de avanço do acionamento das velas.

Cabe lembrar que a função do subsistema de ignição é iniciar a combustão da mistura ar/combustível no interior dos cilindros, de modo a aumentar a pressão interna e efetivamente impulsionar o pistão durante a etapa de expansão do ciclo Otto. Para que se obtenha o melhor aproveitamento da conversão de energia – isto é, aquele que proporciona o maior impulso possível sobre o pistão –, deve-se buscar a situação em que a pressão máxima no interior dos cilindros é atingida precisamente no PMS (ponto morto superior). Para que isso ocorra, a ignição deve ser acionada com um certo adiantamento (ou *avanço*), de modo a dar conta do atraso de propagação da combustão através da massa de ar/combustível.

É de suma importância para o funcionamento adequado do motor que se tenha um controle preciso sobre o adiantamento mencionado. Se por um lado um avanço demasiadamente pequeno resultaria em perda de eficiência, por outro um avanço exageradamente grande ocasionaria um pico de pressão antes de o pistão alcançar o PMS, produzindo por alguns instantes uma força contrária ao sentido de rotação do motor. Tal fenômeno, denominado *detonação*, é bastante nocivo ao funcionamento do motor, podendo inclusive danificar sua estrutura se ocorrer repetidamente. A curva da Figura 6 ilustra a influência do ângulo de avanço da ignição sobre o perfil temporal da pressão interna de um cilindro.

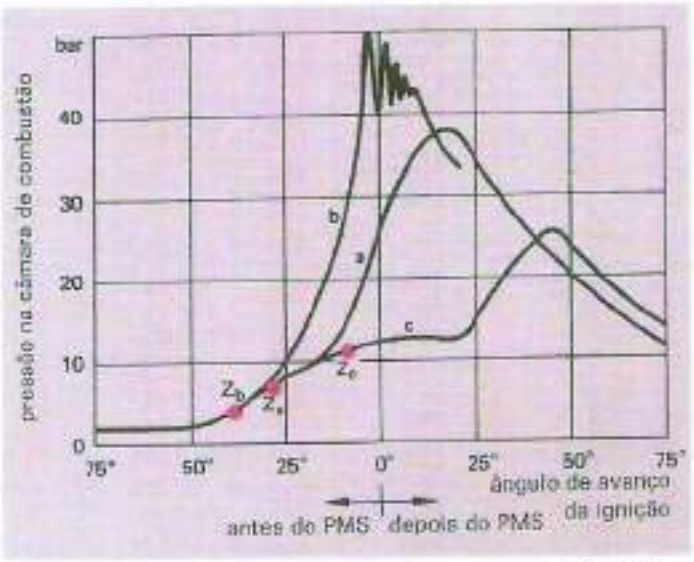


Figura 6. Influência do ângulo de avanço da ignição sobre a pressão interna de um cilindro.

Atendendo às expectativas iniciais, o circuito de interface existente no protótipo do subsistema da ignição pôde ser mantido, por já apresentar um comportamento adequado às necessidades do projeto. Assim, para esse subsistema, não houve necessidade de alterar o hardware já existente, e as atenções puderam se concentrar quase integralmente no software. A Figura 7 apresenta o circuito utilizado, baseado no componente LM1949.

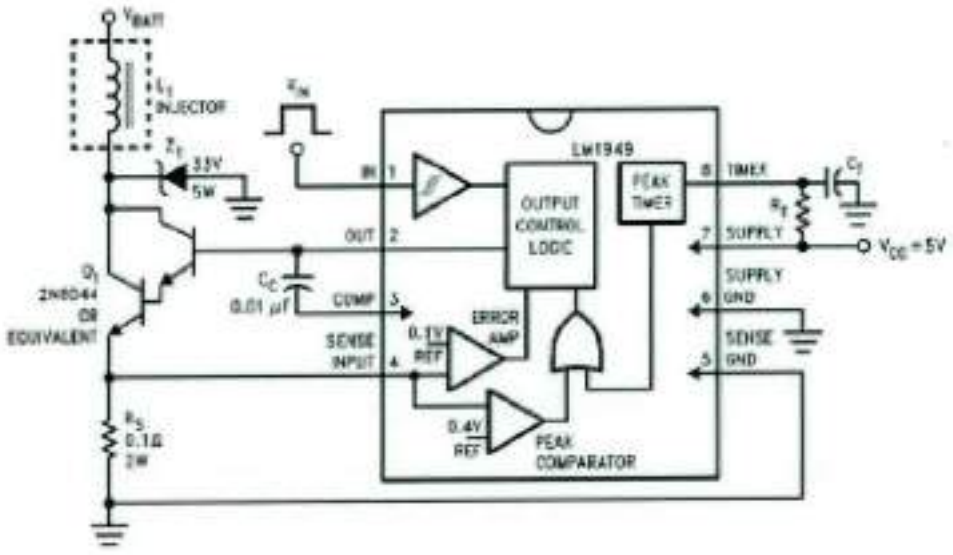


Figura 7. Esquema elétrico do circuito de interface da ignição.

O software desse módulo, por sua vez, foi completamente reescrito, a fim de permitir o ajuste dinâmico de parâmetros por parte do módulo de gerenciamento. A versão anterior do software possuia os diferentes ângulos de avanço da ignição inseridos diretamente no código, e rotinas completamente independentes para cada ângulo. Dessa forma, não era possível variar os mapas de parâmetros, tornando o sistema pouco versátil.

A nova versão do software possui os parâmetros armazenados em vetores, simplificando muito a variação do ângulo de avanço para diferentes situações. Esses vetores podem também ser facilmente armazenados no módulo de macro-gerenciamento, e de lá transferidos no início da operação do sistema para o módulo da ignição, tornando possível concentrar a responsabilidade pela gestão de parâmetros do motor inteiramente no módulo de macro-gerenciamento. Entretanto, devido às restrições de tempo e visando focar os esforços de desenvolvimento nas funcionalidades mais importantes, neste trabalho mantivemos os parâmetros no código do próprio módulo da ignição. Com isso, evitou-se a necessidade de implementar a comunicação I^2C neste módulo, já que esta serviria unicamente para permitir a transferência dos mapas de parâmetros.

Devido à intensa necessidade de sincronização e temporização das atividades desse módulo (para que se inicie a centelha de cada cilindro com o ângulo de avanço adequado e com boa precisão, fator essencial para o bom desempenho do motor), há uma necessidade de calcular a velocidade de rotação em tempo real diretamente no módulo, ao mesmo tempo em que é realizado o controle sobre as velas de ignição. Esse paralelismo de funcionalidades introduz no software da ignição uma complexidade bastante elevada, levando a extremos a dificuldade em se lidar com os tempos de processamento que já encontramos no simulador da roda-fônica.

Por conta dessa complexidade, surgiram no software desse módulo pequenos problemas que custamos a resolver. Em particular, para velocidades altas de rotação, ocorria por vezes perda de sincronismo na geração dos pulsos de acionamento das velas de ignição. Visando reduzir esses problemas, três medidas foram tomadas em relação ao subsistema de ignição.

A primeira delas foi a otimização do código, de modo a reduzir os tempos de processamento nos pontos mais críticos. Experimentou-se diferentes configurações de timers, combinando os timers de 8 e de 16 bits disponíveis no

PIC16F877A. Experimentou-se ainda a utilização de uma interrupção para delimitar o intervalo de tempo utilizado para a medição da velocidade de rotação, ao invés do *polling* que estávamos utilizando até então. Após diversos experimentos, decidiu-se por utilizar o Timer0, de 8 bits, para medir através de *polling* o tempo para determinação da velocidade de rotação, e o Timer1, de 16 bits, para medir a duração dos dentes e intervalos da roda-fônica e para determinar com precisão o instante de início dos pulsos da ignição.

A segunda medida que tomamos foi a implementação de código para identificar a falha da roda-fônica com base na duração dos seus dentes e intervalos, em adição ao método já utilizado anteriormente de identificação através da contagem dos dentes. Como resultado, observou-se uma melhoria significativa na confiabilidade do sistema, uma vez que antes a perda de sincronismo ocasionava uma perda de funcionamento definitiva, até que o *reset* do módulo fosse acionado, e após a implementação dessa melhoria passou-se a observar a retomada do funcionamento na volta seguinte da roda-fônica mesmo quando ocorria perda de sincronismo.

Por fim, a terceira medida que tomamos foi a troca do cristal da placa de desenvolvimento utilizada na ignição, de forma a aumentar o clock do microcontrolador de 4 MHz para 20 MHz (sendo esta última a frequência máxima de operação suportada pelo PIC16F877A). Com o novo cristal, pôde-se observar uma melhora expressiva no comportamento do subsistema da ignição, tanto em termos de precisão na geração dos pulsos de acionamento das velas quanto em termos da faixa de rotações em que o sistema funciona de maneira adequada. Ainda assim, é a ignição que define a velocidade máxima de rotação permitida no sistema, a qual fixamos em 2400 RPM após observarmos que para velocidades acima desta eventualmente as perdas de sincronismo tornam a ocorrer.

Apresentamos no fluxograma da Figura 8 a estratégia geral adotada no software do módulo da ignição, e em seguida o trecho de código correspondente à lógica principal implementada nesse módulo.

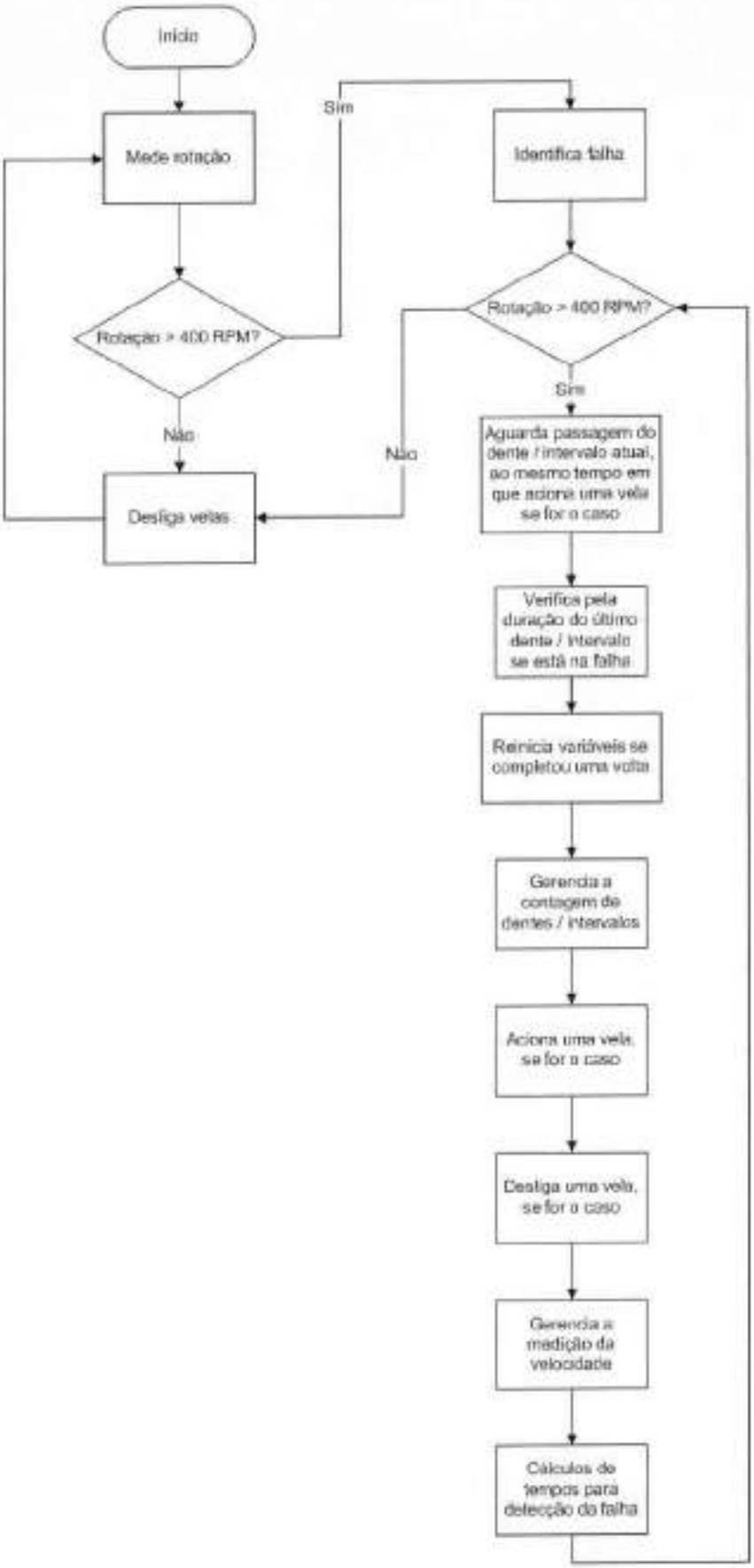


Figura 8. Fluxograma geral da estratégia adotada para o controle da ignição.

```
/* Controla a ignicao e calcula um novo valor para a rotacao */
while (1)
{
    /* Se encontrar um valor para a rotacao menor que a rotacao
minima, retorna */
    if (rot100rpm < rotmin)
        return;

    /* Percorre um dente ou um intervalo entre dentes (dependendo
do valor */
    /* da variavel "dente") */
    while (RODAFONICA == dente)
    {
        /* Se estava esperando um atraso para comandar uma vela,
verifica */
        /* se ja passou o atraso */
        if (atrasando && get_timer1() >= atraso)
        {
            ComandaVela(cil_lig, LIGAR);
            AtualizaFlags();
            atrasando = 0;
        }
    }

    /* Le a duracao do ultimo intervalo e alterna entre dente e
intervalo */
    t_int = get_timer1();
    if (intervalo < 114)
        set_timer1(0);
    dente = !dente;
    if (t_int > t_ref)
        falha = 1;

    /* Se completou uma volta, atualiza os flags e reinicia a
contagem de intervalos */
    if (falha && dente)
    {
        set_timer0(0);
        set_timer1(0);
        intervalo = 0;
        contando = 1;
        falha = 0;
        if (reset)
        {
            liga = intervalos[rot100rpm];
            atraso = atrasos[rot100rpm];
            cil_lig = 0;
            reset = 0;
        }
    }
    else if (++intervalo >= 114)
    {
        intervalo = 114;
        falha = 1;
    }

    /* Comanda as velas e atualiza os flags de ignicao */
    if (intervalo == liga)
    {
        if (atraso == 0)
        {
            ComandaVela(cil_lig, LIGAR);
            AtualizaFlags();
        }
        else
            atrasando = 1;
    }
    if (intervalo == desliga)
    {
```

```
            ComandaVela(cil_desl, DESLIGAR);
            desliga = 255;
        }

        /* Atualiza a velocidade de rotacao */
        if (contando && get_timer0() > 195)
        {
            rot100rpm = intervalo;
            contando = 0;
        }

        /* Atualiza o tempo de referencia para identificacao da falha
*/
        t_ref = t_int;
        t_ref += (t_int >> 1);
}
```

A Tabela 1 abaixo apresenta os valores adotados para o ângulo de avanço da ignição, para diferentes faixas de velocidade de rotação. Cabe ressaltar que os ângulos estão dados em relação ao ciclo Otto, e as velocidades em relação ao eixo do comando de válvulas, de modo que ambos se aplicam diretamente à roda-fônica do sistema construído (a qual, destacamos novamente, se encontra montada no eixo do comando de válvulas). Conforme explicaremos mais adiante, foi adotado o valor de 400 RPM como rotação de início para a operação da ignição, e a rotação máxima do sistema foi limitada em 2400 RPM (embora a mesma tabela possa ser utilizada para rotações de até 6000 RPM, como se pode observar abaixo).

Tabela 1. Ângulos de avanço da ignição.

Faixa de Rotações (RPM)	Ângulo de avanço (°)
400 – 999	9
1000 – 1499	12
1500 – 1999	15
2000 – 2499	19
2500 – 2999	23
3000 – 3499	25
3500 – 3999	28
4000 – 4499	30
4500 – 4999	32
5000 – 6000	34

Na Figura 9, apresentamos as formas de onda obtidas através de um osciloscópio para o pulso de acionamento da vela referente ao quarto cilindro, para as rotações de 1200 RPM e 1800 RPM. A 1200 RPM, o período de uma rotação é

50ms, e portanto o avanço desejado (12°, conforme a Tabela 1) corresponde a 1,667ms. Pela Figura 9, observa-se a medida de 1,720ms, ou seja, um erro inferior a 4%. Da mesma forma, para 1800 RPM temos que o período de uma rotação vale 33,333ms, e portanto o adiantamento desejado (de 15°) corresponde a 1,389ms. Pela Figura 9, observa-se a medida de 1,440ms, fornecendo um erro também inferior a 4%.

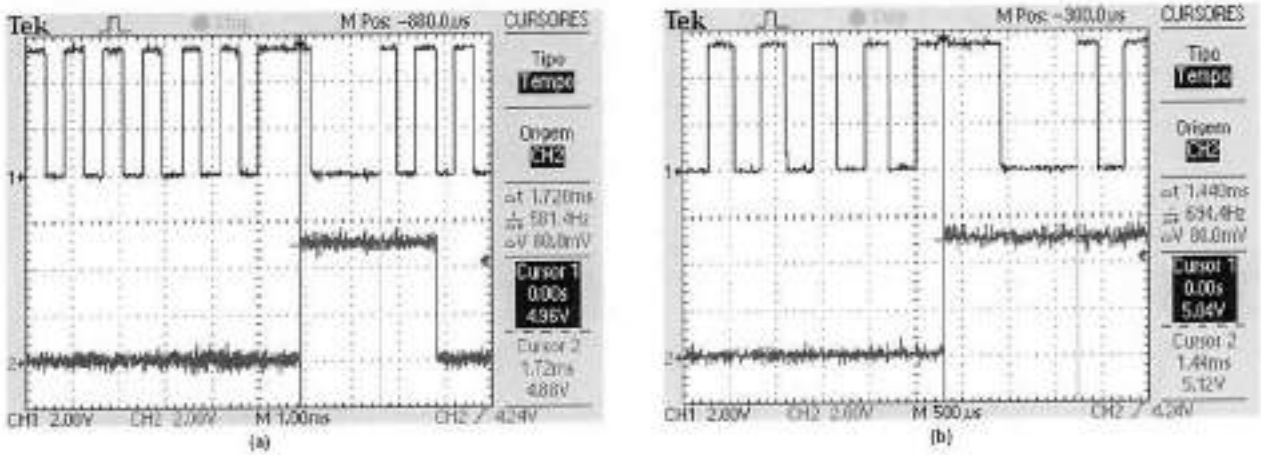


Figura 9. Verificação do funcionamento do subsistema da ignição para rotações de: a) 1200 RPM e b) 1800 RPM.

As Figuras 10 e 11 apresentam, respectivamente, uma visão geral do subsistema da ignição finalizado, e uma visão em detalhe das velas durante o acionamento.

Figura 10. Visão geral do subsistema da ignição finalizado.

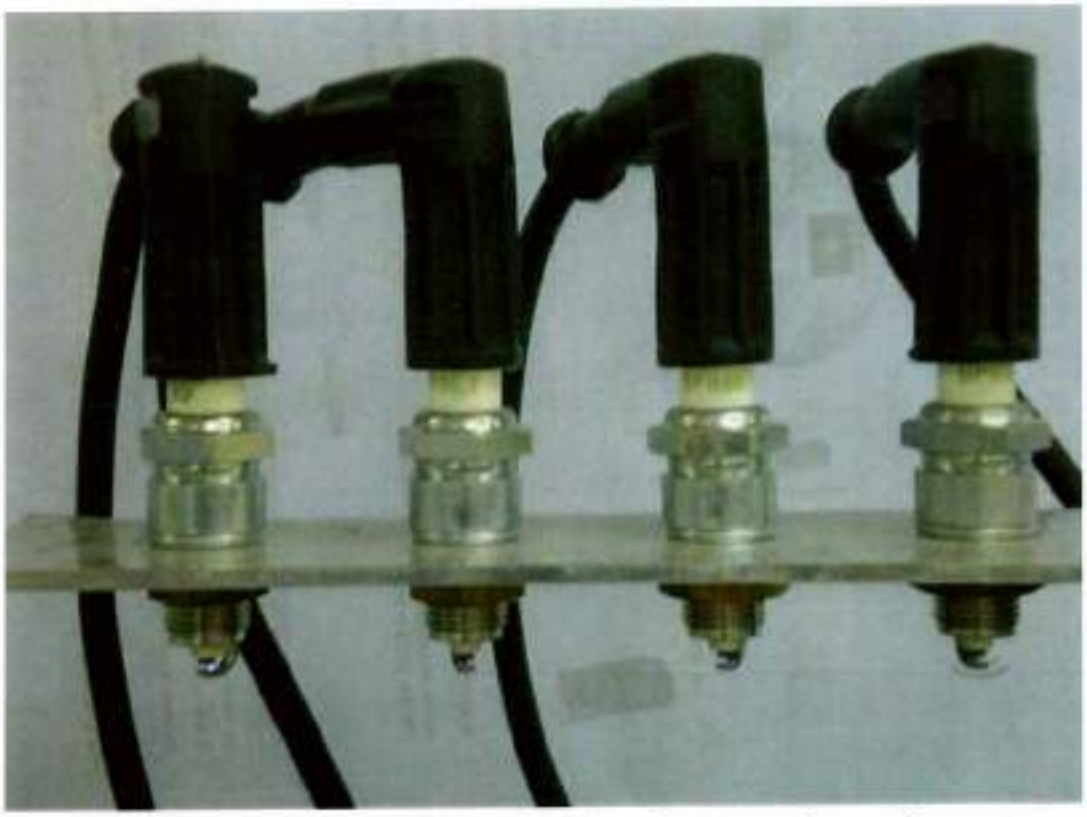


Figura 11. Visão em detalhe das velas durante o acionamento.

6.3 Subsistema de injeção

O subsistema da injeção tem a função de gerenciar o acionamento das válvulas injetoras, acionando-as nos instantes adequados e durante um intervalo de tempo adequado. Sua funcionalidade principal é o controle do tempo de abertura dos injetores, uma vez que é este tempo que determina a massa de combustível injetada (já que a pressão na linha de combustível é mantida constante pela bomba de combustível e pelo regulador de pressão presente no distribuidor da injeção).

Assim como para a ignição, o circuito de interface da injeção pôde ser aproveitado exatamente da forma como estava no protótipo inicial, sem a necessidade de se realizar alterações, satisfazendo às expectativas iniciais do projeto. O circuito de interface da injeção é idêntico ao da ignição, fazendo uso do componente LM1949, e pode ser visto na Figura 7. A possibilidade de se utilizar o mesmo circuito se deve à natureza semelhante das velas e dos injetores, que consistem essencialmente em bobinas (transformadores, no caso das velas, e solenóides no caso dos injetores) acionadas por pulsos, os quais são produzidos pelos microcontroladores e tratados por circuitos de potência.

Também neste caso, o software do microcontrolador foi completamente reescrito, visando torná-lo mais robusto em termos da sua operação em geral, e visando aproveitar o grande esforço despendido no desenvolvimento do software da ignição, uma vez que ambos são muito semelhantes.

Tanto o software da ignição quanto o da injeção devem gerar pulsos com larguras determinadas e em instantes determinados de cada revolução do motor. A diferença entre eles reside essencialmente no fato de que, para a ignição, precisa-se de uma precisão maior no instante de início dos pulsos, enquanto a duração dos pulsos é menos importante, ao passo que para a injeção temos a situação inversa, em que o instante de início dos pulsos não necessita de muita precisão mas sua duração exige um controle bastante preciso. Além disso, o comportamento da ignição depende essencialmente da velocidade de rotação do motor, enquanto o da injeção depende de fatores mais complexos – sendo os principais o estado do motor (recém ligado, marcha lenta, aceleração, desaceleração, etc.) e a massa de ar admitida. Por esse motivo, enquanto o subsistema da ignição pôde ser desenvolvido para operação isolada, sem a necessidade de comunicação através da rede I^2C,

para o da injeção a comunicação é essencial, pois os tempos de injeção só podem ser definidos de maneira apropriada pelo subsistema de macro-gerenciamento.

A transferência do controle sobre os parâmetros para o módulo de macro-gerenciamento permitiu claramente a redução da complexidade das atividades do módulo da injeção. O paralelismo de funcionalidades exigido no caso do módulo da ignição, em que era necessário medir a velocidade de rotação ao mesmo tempo em que se controlava os pulsos produzidos, deixa de ser necessário no caso da injeção, uma vez que não se precisa realizar a medição da velocidade de rotação. Além disso, o tempo de processamento gasto com a comunicação I^2C é desprezível, e o controle da duração dos pulsos pode ser realizado de maneira simples através do uso de um timer.

Para o módulo da injeção utilizou-se uma placa de desenvolvimento com cristal de 4 MHz, e o desenvolvimento do software partiu de uma cópia do software de ignição, realizando-se em seguida as alterações necessárias. O subsistema da injeção deve receber periodicamente do macro-gerenciamento um parâmetro simples, de apenas um byte, informando o tempo de injeção. Foram definidos dois valores especiais para tal parâmetro, um deles (0x00) permitindo o desligamento total da injeção, e outro (0xFF) permitindo a abertura permanente das válvulas injetoras (o que possibilita a obtenção de uma mistura extremamente rica, o que pode ser desejado em algumas situações, como a aceleração inicial logo após o arranque, com motor frio). Por simplicidade, o avanço foi fixado arbitrariamente em 30°, garantindo o início dos pulsos sempre em sincronismo com as transições do sinal da roda-fônica, e nunca no meio de um dente/intervalo.

Na Figura 12 apresentamos a lógica implementada para controlar a injeção. Pode-se notar a semelhança com a da ignição. Pode-se observar também que, no caso da injeção, foram implementadas funcionalidades de diagnóstico, acessíveis através dos botões da placa de desenvolvimento.

Figura 12. Fluxograma geral da estratégia adotada para o controle da injeção.

```
/* controla a injecao */
while (1)
{
    //caso tenha acontecido alguma colisão, este bit será setado.
Entao só por software que devemos limpá-lo.
    if(SSPCON_WCOL) SSPCON_WCOL=0;
    //Se um novo byte foi recebido antes do registrador SSPBUFF
ser lido, este bit será setado e só por software devemos limpá-
lo.
    if(SSPCON_SSPOV) SSPCON_SSPOV=0;

    /* Trata o acionamento completo ou desligado */
    if (tinj_off)
    {
        for (i = 0; i < 4; i++)
            ComandaValvula(i, DESLIGAR);
        atrasando = 0;
    }
    else if (tinj_full)
    {
        for (i = 0; i < 4; i++)
            ComandaValvula(i, LIGAR);
        atrasando = 0;
    }

    /* Percorre um dente ou um intervalo entre dentes (dependendo
do valor */
    /* da variavel "dente") */
    while (RODAFONICA == dente)
    {
        /* Se estava esperando o tempo de injecao para desligar
uma valvula, */
        /* verifica se ja passou o tempo */
        if (atrasando && get_timer0() >= tinj)
        {
            ComandaValvula(cil_desl, DESLIGAR);
            atrasando = 0;
        }
    }

    /* Le a duracao do ultimo intervalo e alterna entre dente e
intervalo */
    t_int = get_timer1();
    if (intervalo < 114)
        set_timer1(0);
    dente = !dente;
    if (t_int > t_ref)
        falha = 1;

    /* Se completou uma volta, atualiza os flags e reinicia a
contagem de intervalos */
    if (falha && dente)
    {
        set_timer1(0);
        intervalo = 0;
        falha = 0;
        liga = int_ini;
        cil_lig = 0;
    }
    else if (++intervalo >= 114)
    {
        intervalo = 114;
        falha = 1;
    }

    /* Comanda as valvulas e atualiza os flags de injecao */
    if (intervalo == liga && !tinj_off && !tinj_full)
    {
        ComandaValvula(cil_lig, LIGAR);
```

```
        set_timer0(0);
        atrasando = 1;
        cil_desl = cil_lig;
        liga += 30;
        cil_lig++;
    }

   /*Atualiza o tempo de referencia para identificacao da falha*/
    t_ref = t_int;
    t_ref += (t_int >> 1);

    /* Trata os botoes para diagnostico */
    if (!input(pin_b1))
    {
        tinj_full = 0;
        tinj_off = 1;
        enable_interrupts(INT_SSP);     // gerenciamento
        LimpaLinha(1);
    }
    while (!input(pin_b1));
    if (!input(pin_b2))
    {
        disable_interrupts(INT_SSP);    // aberto full
        tinj_full = 1;
        tinj_off = 0;
        LimpaLinha(1);
        sprintf(texto, "diagnose: full");
        EnviaTexto(texto, 0xC0);
    }
    while (!input(pin_b2));
    if (!input(pin_b3))
    {
        disable_interrupts(INT_SSP);    // abertura fixada
        tinj_full = 0;
        tinj_off = 0;
        tinj = 100;
        LimpaLinha(1);
        sprintf(texto, "diagnose: 100");
        EnviaTexto(texto, 0xC0);
    }
    while (!input(pin_b3));
}
```

Foi adotada a seguinte equação para o cálculo dos tempos de injeção em função do valor fornecido pelo módulo de sensores e entradas para o sensor da válvula borboleta, cálculo esse realizado pelo macro-gerenciamento:

$$t_{inj} = 1{,}343 * s_{borboleta} - 112{,}69$$

Considerando que para o controle do tempo de injeção foi utilizado o Timer0 com *prescaler* igual a 32, e que o módulo dos sensores fornece para o sensor da válvula borboleta os valores de 107 para abertura de marcha lenta e 177 para abertura máxima, temos o tempo de injeção variando linearmente entre

$$t_{inj} = 1{,}343 * 107 - 112{,}69 = 31 = 31 * 32\mu s \approx 1ms$$

para a válvula com abertura de marcha lenta e

$$t_{inj} = 1{,}343 * 177 - 112{,}69 = 125 = 125 * 32\mu s = 4ms$$

para a válvula com abertura máxima.

O funcionamento do subsistema da injeção pode ser verificado na Figura 13, em que aparece a captura das formas de onda (realizada com auxílio de um osciloscópio) para as rotações de 1200 RPM e 1800 RPM, e ainda para o funcionamento com os injetores constantemente abertos. Para a rotação de 1200 RPM, temos o módulo de sensores fornecendo o valor de 124 referente ao sensor da válvula borboleta, resultando em um tempo de injeção de

$$t_{inj} = 1{,}343 * 124 - 112{,}69 = 54 = 54 * 32\mu s = 1{,}728ms,$$

enquanto para a rotação de 1800 RPM o módulo de sensores fornece o valor de 151 para o sensor da válvula borboleta, resultando em um tempo de injeção de

$$t_{inj} = 1{,}343 * 151 - 112{,}69 = 90 = 90 * 32\mu s = 2{,}880ms.$$

Observando a Figura 13, temos que para 1200 RPM o tempo de injeção medido foi de 1,840ms, resultando em um erro de aproximadamente 6,5%, e para 1800 RPM mediu-se o tempo de injeção em 2,920ms, resultando um erro de cerca de 1,4%. Deve-se ressaltar que, como a realização dessas medidas exige o funcionamento do sistema comandado pelo gerenciamento, as rotações foram estabelecidas experimentalmente pressionando-se suavemente o pedal acelerador, à medida em que se acompanhava o valor da velocidade medida. Esse método implica um erro considerável no estabelecimento da rotação, uma vez que a medida de rotação é tomada em passos de 100 RPM, de modo que os erros calculados acima indicam que o sistema consegue estabelecer um controle muito preciso sobre o tempo de injeção.

Ainda na Figura 13, podemos observar o comportamento do sistema com o bico injetor mantido constantemente aberto. Como se poderia esperar, a forma de onda produzida pelo microcontrolador nesse caso é um nível de tensão contínua em cerca de 5V.

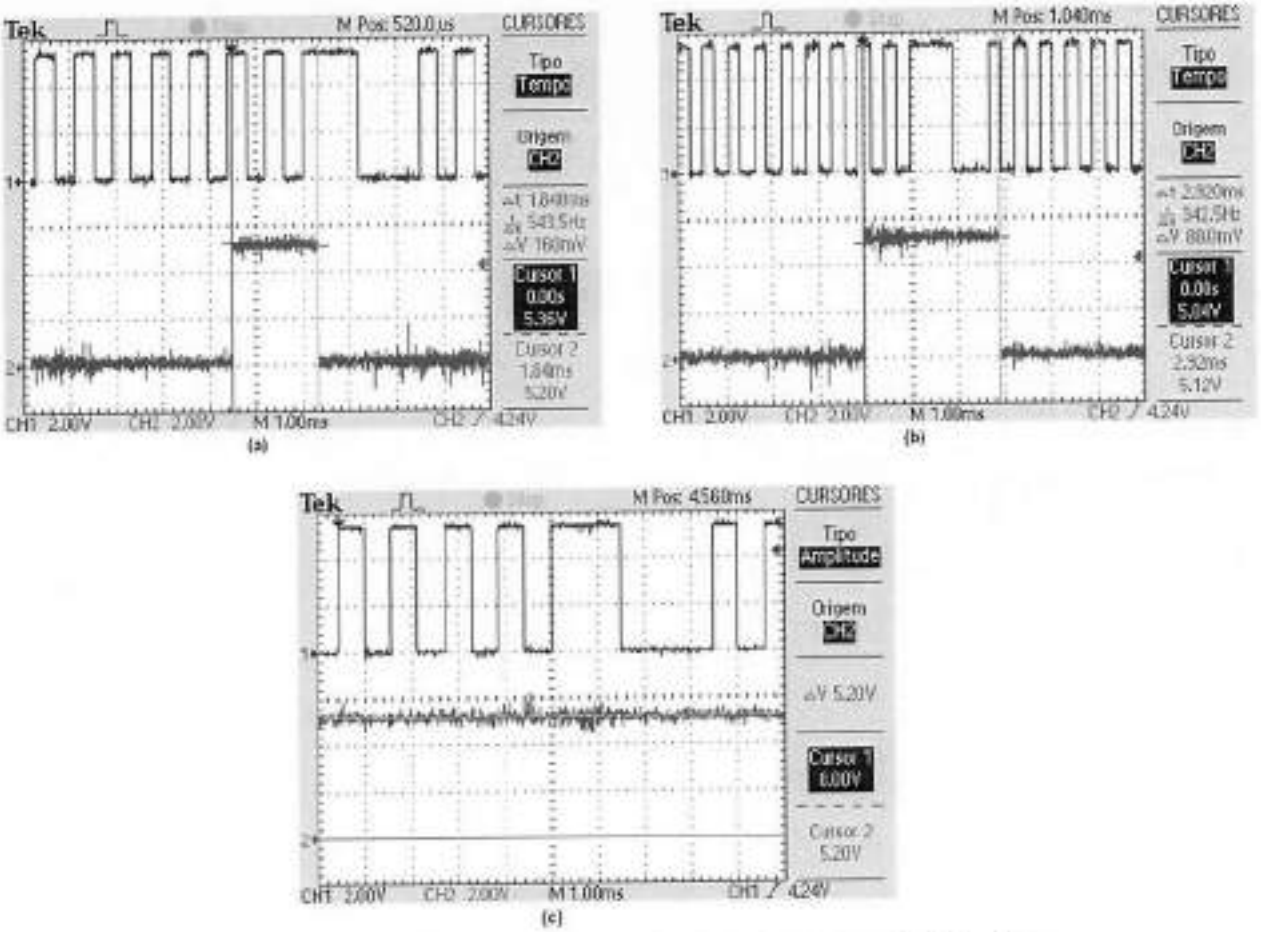


Figura 13. Verificação do funcionamento do subsistema da injeção para rotações de: a) 1200 RPM; b) 1800 RPM; e c) abertura completa.

Por fim, a Figura 14 apresenta uma visão geral do subsistema de injeção finalizado, e a Figura 15 apresenta em detalhe os bicos injetores durante seu acionamento.

Figura 14. Visão geral do subsistema da injeção finalizado.

Figura 15. Visão em detalhe dos bicos injetores durante o acionamento.

6.4 Subsistema da válvula borboleta

A válvula borboleta é um atuador que controla o volume de ar a ser admitido no motor a partir de um sinal analógico enviado pelo pedal acelerador. Possui um disco cuja posição pode variar entre 0° (válvula fechada - disco perpendicular ao fluxo) e 90° (válvula aberta - disco paralelo ao fluxo). Para se medir o ângulo de abertura, junto a ela existe um sensor denominado TPS (*Throttle Position Sensor*).

A interface da válvula é constituída de seis pinos. Dois desses pinos são utilizados para alimentar o motor que movimenta o disco, e os outros quatro estão relacionados ao sensor. Estes seis fios, mostrados na Figura 16, possuem as seguintes funções:

- O primeiro fio corresponde a uma saída do sensor TPS, e pelo gráfico nota-se que quanto maior for a abertura da válvula maior é a resistência do potenciômetro, de modo que a tensão de saída nesse fio diminui conforme aumenta o ângulo de abertura.
- O segundo fio corresponde à tensão de alimentação do sensor, e deve ser mantido em 5V.
- O terceiro fio corresponde a outra saída do sensor TPS, porém, ao contrário do primeiro, fornece uma tensão crescente proporcional a abertura da válvula.
- O quarto fio corresponde ao terra do sensor TPS.
- Já o quinto e o sexto fios correspondem às tensões de alimentação e terra do servo-motor da válvula borboleta (responsável por movimentar o disco).

Nota-se que a existência de dois sensores internos ao TPS oferece redundância para a medida do ângulo de abertura da válvula borboleta.

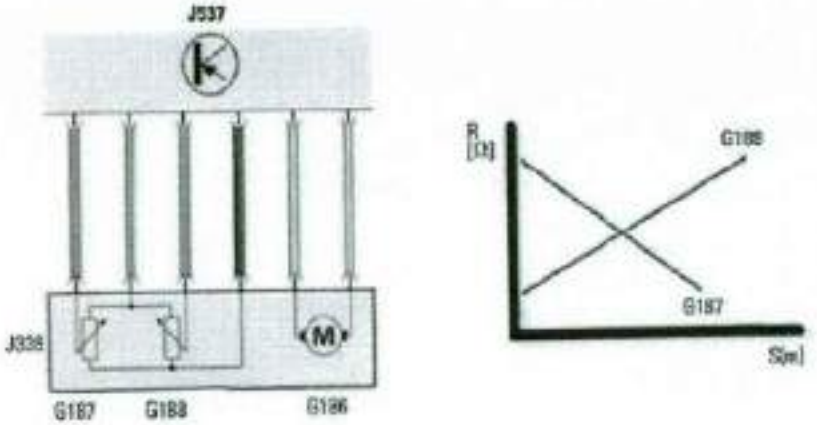


Figura 16. Diagrama e curvas de resposta do sensor TPS.

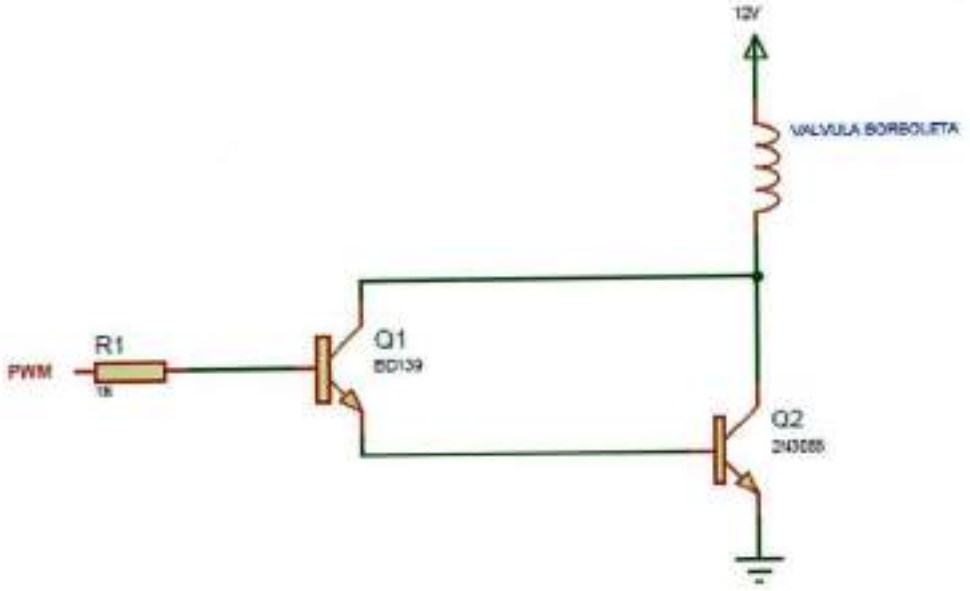


Figura 17. Circuito Darlington utilizado para controlar a abertura da válvula borboleta.

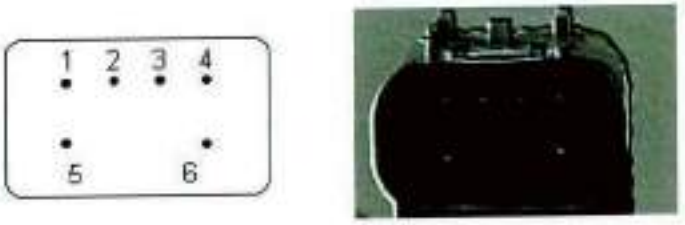


Figura 18. Adoção no número dos pinos.

Inicialmente foi medido as resistências nos seus terminais com a válvula fechada e a válvula totalmente aberta (90°), seguin do a pinagem abaixo:

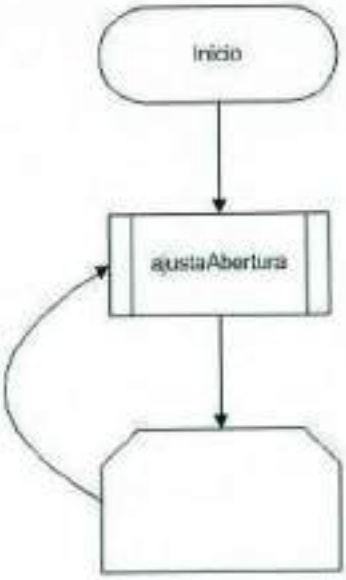
Pinos	Fechado(kΩ)	Aberta (kΩ)
1-2	1,31	2,77
1-3	2,68	1,38
1-4	1,96	1,96
2-3	3,55	3,57
2-4	2,77	1,44
3-4	1,23	2,62
5-6 (servo-motor)	2Ω	2Ω

Foi a partir destes testes que observamos que o pino 5 é o positivo e o 6 é o negativo da alimentação do servo-motor. A alimentação indicada para o servo-motor é de 12V. Já os sensores de posição de abertura é de 5V.

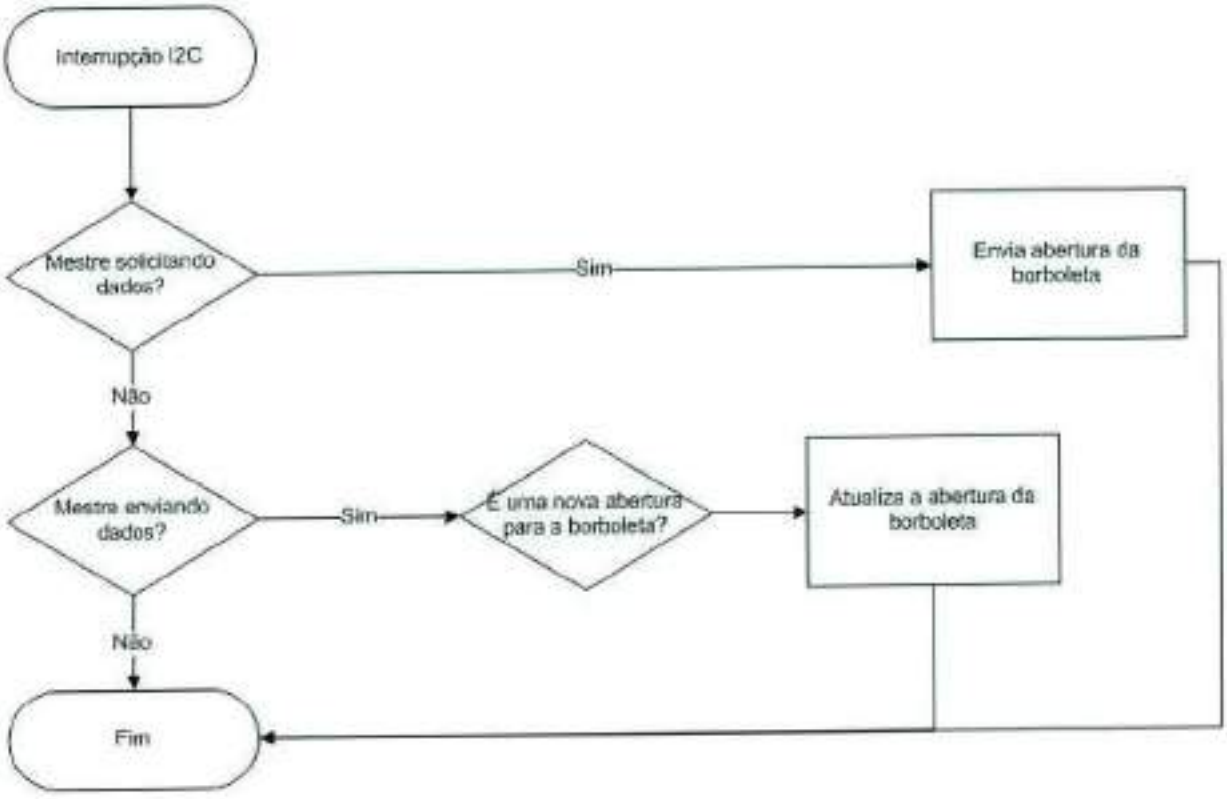
Para controlar a abertura da válvula foi utilizado um circuito Darlington como o da Figura 17. Sua entrada é composta por um sinal PWM (*Pulse Width Modulation*, Modulação por Largura de Pulso) que será fornecido pelo microcontrolador. Esse PWM fornecerá uma tensão média de 0 a 5V dependendo do *duty cicle*. O circuito será ligado em série com o negativo da alimentação da válvula. Assim é possível controlar o ângulo de abertura da borboleta. Foi verificado, também, que a corrente que circula por ele é por volta de 3A.

A seguir o fluxograma de como foi implementada a lógica de controle da válvula borboleta:

Fluxograma da rotina principal



Fluxograma da Interrupção I2C da Borboleta



Como pode ser observado, na rotina principal a abertura da borboleta é constantemente imposta, sendo assim o servomotor sempre irá impor a última

posição recebida do gerenciamento na borboleta. A rotina de interrupção é mostrada também. É a partir dela que o subsistema tanto recebe quanto envia a abertura atual da borboleta.

Abaixo apresentamos o código da rotina *ajustaAbertura*:

```
/*Esta função irá ajustar o valor da abertura da borboleta de acordo com o valor
fornecido.
Entrada:
      vref: valor desejado (de 0 a 255).
Saida:
      retorna o valor para o qual foi ajustado.
*/
int ajustaAbertura(int vref)
{
      //esta variável guardará o valor obtido do sensor TPS da válvula.
      int valorSensorValvula;
      //contém o valor do erro atual
      signed int erro=0;

      vref=0.42*vref +84.0;

      /*
      No microcontrolador a porta RA2 estará ligada ao sensor TPS da borboleta,
sendo assim
      selecionamos essa porta como aquela que fornecerá a conversão A/D.
      */
      set_adc_channel(2);
      delay_us(30);

      /*
        Este loop acabará quando atingirmos uma abertura aceitável perante aquela
que desejamos.
      */
      while(true)
      {
         delay_us(10);

         //Obtendo o valor da porta A/D selecionada. Neste caso de RA2
         valorSensorValvula=read_adc();

         //Calculando o erro entre o valor retornado e o desejado
         erro = valorSensorValvula - vref;

         //Se o erro for aceitável
         if(abs(erro)<2){

            /*
              Ao aplicarmos um PWM de 2.5V a válvula vai se manter aberta na
posição atual. Entao criamos um duty
              de 50% a seguir.
            */
             set_pwm1_duty(128);
             return valorSensorValvula;
         }

         /*
                  Se o erro for positivo, ou seja, a abertura está maior do que
gostariamos, forçamos a válvula a
         fechar tudo, até que durante sua abertura passe pela abertura que
desejamos.
         */
         if(erro>0) set_pwm1_duty(0);
         /*
            Se o erro for negativo, ou seja, a abertura está menor do que
gostariamos, forçamos a válvula
         a abrir tudo seguindo o princípio anterior.
         */
         else set_pwm1_duty(255);
      }
```

```
}
```

E por fim podemos verificar na Figura 19 como ficou fisicamente implementado o subsistema da válvula borboleta.

Figura 19. Subsistema da válvula borboleta.

Nas Figuras 20 e 21 é mostrado o controle da válvula quando ela está aberta no ângulo de correspondente ao arranque do motor e no ângulo correspondente a máxima aceleração respectivamente.

Figura 20. Válvula aberta no ângulo equivalente ao arranque.

Figura 21. Válvula totalmente aberta.

6.5 Comunicação I^2C

O I^2C é um protocolo serial de comunicação entre circuitos integrados. Por ser serial, a informação é transmitida seqüencialmente, e não em paralelo. O PIC16F877A já possui recursos que possibilitam o uso desse protocolo mais facilmente.

Para sua utilização é necessário um barramento com duas vias:

- Clock (SCL: Serial Clock) : utilizando o pino RC3 do PIC;
- Dados (SDA: Serial Data): utilizando o pino RC4 do PIC.

Evidentemente, o terra deve ser comum a todos os microcontroladores do barramento.

O estado padrão do barramento é em nível alto, sendo necessário instalar resistores de pull-up (com valores entre 1kΩ e 10kΩ) tanto na via do relógio quanto

na de dados, já que tanto os pinos do SCL quanto SDA podem funcionar com coletor aberto.

O I^2C permite que o endereçamento seja feito utilizando 7 ou 10 bits; como não será necessário endereçar mais do que 128 dispositivos, optou-se por utilizar um endereçamento com 7 bits. Além disso, é necessário garantir que a capacitância máxima na via seja de 400µF. Haverá um único mestre, que é o microcontrolador do módulo de macro-gerenciamento, e os escravos serão os microcontroladores dos outros módulos do sistema.

6.5.1 Regras do barramento

Em relação ao barramento, as seguintes regras devem ser obedecidas:

1. O sinal na via de dados só pode ser alterado no nível baixo da linha de clock;
2. O dado só pode ser lido durante o nível alto da linha de clock;
3. Quando tanto SCL quanto SDA permanecem em nível alto, isso significa que o barramento não está em uso.

Porém existem dois sinais especiais que violam a primeira regra:

- START: força SDA de "1" para "0" enquanto SCL está em "1";
- STOP: força SDA de "0" para "1" enquanto SCL está em "1".

Além desses, outro sinal especial importante é o ACK (*Acknowledge* – utilizado para confirmar o recebimento de mensagens, e para sinalização em alguns casos específicos). O ACK é enviado mantendo-se a linha em "0" na próxima borda de subida do sinal de clock.

6.5.2 Procedimento para a comunicação entre dois dispositivos

O I^2C define um protocolo para que se estabeleça a comunicação entre dois dispositivos. A seguir indicamos os passos necessários para que um dispositivo mestre se comunique:

1. Esperar que o barramento esteja livre, com SDA = "1" e SCL = "1".
2. Sinalizar, com uma mensagem de START, que o barramento agora está ocupado.
3. Prover o sinal de clock (SCL) que será utilizado por todos os outros dispositivos como referência de tempo. O dado (SDA) será válido somente na borda de subida do clock.
4. Serializar o endereço do dispositivo com o qual se deseja comunicar, mais um bit indicando se será realizada uma escrita ou uma leitura. No nosso caso, serão 7 bits de endereçamento, mais um bit indicando se é escrita ou leitura.
5. Esperar por um ACK, enviado pelo dispositivo escravo para indicar que existe um dispositivo com o endereço na via e que este está pronto para se comunicar.
6. O mestre então transfere/recebe dados para/do outro dispositivo. A cada byte transferido, o dispositivo que enviou o dado espera um ACK indicando que o dado foi recebido com sucesso.
7. Para encerrar a comunicação, é necessário sinalizar que não serão enviados novos dados através de uma mensagem de STOP.

6.5.3 Detalhes da implementação

Para aproveitar os recursos específicos para I^2C presentes no PIC16F877A utilizamos as seguintes funções, que se encontram na própria biblioteca do compilador CCS PCWH v4.078:

- I2c_start: Inicia a condição de START no barramento I2C. Somente pode ser chamada por um dispositivo mestre.
- I2c_stop: Inicia uma condição de STOP no barramento I2C. Chamada tanto pelo mestre, quando este não quer mais receber dados, quanto pelo escravo, quando não tem mais dados para enviar.
- I2c_read: efetua a leitura de um byte do barramento.
- I2c_write: escreve um byte no barramento.

No dispositivo escravo, quando um dado é recebido, é acionada a interrupção INT_SSP. Assim, deve-se associar em cada dispositivo escravo uma função que atenda a essa interrupção.

A seguir, indicamos um exemplo de comunicação I^2C, implementado no módulo da válvula borboleta (estão indicados somente os trechos de código referentes à comunicação).

Inicialmente, as definições importantes para configurar o dispositivo como escravo:

```
/*
Esta é uma diretiva que habilitará o uso da biblioteca
interna de comunicação I2C
-slave : indica que o dispositivo funcionará como escravo
-address: o endereço do dispositivo no barramento
-sda: pino do microcontrolador que corresponderá ao SDA
-scl: pino do microcontrolador que corresponderá ao SCL
-slow: significa que o barramento irá operar a 100kHz
-force_hw: para forçar a utilização do hardware interno do PIC,
 aproveitando assim o uso das interrupções
*/
#use I2C (slave, address = 0xC0 ,sda = PIN_C4, scl = PIN_C3,
force_hw, slow)

// 1:Mestre solicitando dados 0:Mestre enviando dados.
#BIT SSPSTAT_RW = 0x94.2
//1:O byte que chegou corresponde a um dado  0:O byte que chegou
//corresponde a um endereço.
#BIT SSPSTAT_DA = 0x94.5
//Indica colisão na escrita do registrador SSPBUF.
#BIT SSPCON_WCOL = 0x14.7
//Indica erro de overflow na recepção.
#BIT SSPCON_SSPOV = 0x14.6
```

A seguir, a função que está associada à interrupção INT_SSP:

```
#INT_SSP
void sspinterupt()
{
    boolean ack;
    //mestre solicitando dados
    if(SSPSTAT_RW == 1){
 //ao escrever um dado na via, é retornado um ACK=0 ou NACK=1.
 ack=i2c_write(g_abertura);
 //a seguir é dado um stop significando que dados não serão mais
enviados.
 i2c_stop();
    }
   //mestre está enviando dados.
   else{
        /*
         como o mestre inicialmente escreve o endereço do
dispositivo na via,é necessário saber se o dado enviado pelo
mestre corresponde a um endereço ou a uma informação.
        */

        ///se for uma informação
        if(SSPSTAT_DA == 1)
        {
            //lendo o dado da via.
            ultimo_dado=i2c_read();
         }
                //é um endereço e precisamos ler o dado para
esvaziar o buffer de recepção do microcontrolador
                else i2c_read();
     }
    delay_us(5);
}
```

Dentro da rotina principal, é necessário configurar inicialmente as portas associadas aos sinais SDA e SCL como entradas.

Para configurar um dispositivo como mestre, as configurações são as seguintes:

```
/*
 Esta é uma diretiva que habilitará o uso da biblioteca
 interna de comunicação I2C
 -master : indica que o dispositivo funcionará como mestre
 -sda: pino do microcontrolador que corresponderá ao SDA
 -scl: pino do microcontrolador que corresponderá ao SCL
 -slow: significa que o barramento irá operar a 100kHz
 -force_hw: para forçar a utilização do hardware interno do PIC,
  aproveitando assim o uso das interrupções
*/
#use I2C (master, sda = PIN_C4, scl = PIN_C3, force_hw, slow)
#BIT SSPCON_WCOL = 0x14.7 //Indica colisão na escrita do
                          //registrador SSPBUF.
#BIT SSPCON_SSPOV = 0x14.6 //Indica erro de overflow na recepção.
```

Obs.: Nota-se que não foi associado um endereço ao dispositivo mestre, e que não se fez uso do registrador SSPSTAT como na configuração do escravo (verificar no comentário do código do escravo as explicações).

A seguir, será mostrado o trecho de código correspondente ao loop principal de um dispositivo mestre que poderia se comunicar com o escravo configurado acima (fazendo-se uso da placa de desenvolvimento McLab2):

```
while(true)
{
 //caso tenha acontecido alguma colisão, este bit será setado.
Entao só por software que devemos limpá-lo.
 if(SSPCON_WCOL) SSPCON_WCOL=0;

 //Se um novo byte foi recebido antes do registrador SSPBUFF ser
lido, este bit será setado e só por
 //software devemos limpá-lo.
 if(SSPCON_SSPOV) SSPCON_SSPOV=0;

 delay_us(30);
 //obtendo os dados do potenciometro.
 valorPot = read_adc();

 //Botão 0 irá enviar o dado do potenciometro para o escravo.
 if(!input(pin_b0)){
       i2c_start();

       /*
        Para se enviar um dado, primeiro é necessário escrever no
barramento o endereço do escravo.
        Caso exista algum escravo com esse endereço ack vai ser
igual a 0, caso contrário será
        igual a 1.
                  */
       ack=i2c_write(SLAVE_ADDR);

       /*
         Agora finalmente escrevo o dado que quero enviar para o
escravo, aqui se o dado foi
       recebido.
         Se o dado foi recebido ack=0, caso contrário 1.
       */
       ack =i2c_write(valorPot);
       i2c_stop();

       /*OBS: Qualquer função de escrita (i2c_write) travará o
programa esperando ou um ack ou
       um nack*/

       //Agora mostro no LCD qual o dado que foi transmitido.
       lcd_pos_xy(5,1);
       lcd_Atualiza(valorPot);
 }while(!input(pin_b0))


              //Botão 1 irá solicitar um dado para o escravo.
 if(!input(pin_b1)){
       i2c_start();

       /*
         Perceba que como estamos solicitando dados, escrevemos o
endereço do escravo forçando
       que o bit menos significativo seja 1.
         Quando simplesmente enviamos os dados, o bit menos
significativo era 0.
                                  */
       ack=i2c_write(SLAVE_ADDR | 1);

       /*
         Agora estamos lendo o dado que o escravo enviou.
```

```
          Aqui algo muito importante, caso a gente nao espere mais
dados do escravo, é necessário
        enviar um NACK para o escravo quando lermos o dado que ele
enviou. Isto é feito passando 0
        como argumento da função i2c_read.
          Caso a gente quisesse que o escravo enviasse dois dados,
ou seja, chamariamos duas vezes
        a função i2c_read, na primeira vez chamariamos ela
passando 1, na segunda vez teriamos que
        passar 0.
          Como neste programa o escravo irá mandar somente 1 dado,
passamos 0 para a função.
        */
        dado = i2c_read(0);
        i2c_stop();

        //atualizando o display.
        lcd_pos_xy(5,2);
        lcd_Atualiza(dado);

  } while(!input(pin_b1))
}
```

6.6 Inversor de frequência

Para simular o motor elétrico foi utilizado o inversor de frequência CFW-08, que para finalidade de testes pode-se utilizar o software Super Drive (versão utilizada é a 5.80). Com ele é possível impor rotações que vão desde 200rpm a 3000rpm diretamente para a roda fônica.

Para isso ele dispõe de algumas fontes de comandos, sendo as principais:

- Teclas das HMIs (Human Machine Interfaces);
- Bornes de controle (XC1) - via entradas digitais;
- Via interface serial.

Neste projeto, utilizaremos o controle através da interface serial para controlá-lo, conforme detalhado a seguir.

6.6.1 Comunicação serial

O controle do inversor se dará por comunicação serial.

Para utilizarmos essa comunicação do lado do mestre, dispomos de uma interface serial já inclusa na placa didática McLab2 e recursos do próprio PIC16F887A para facilitar a transmissão e recepção de dados. Já do lado do inversor, é necessário utilizar o módulo KCS-CFW08, conforme mostrado na Figura 22. Essa interface permite a conexão ponto a ponto (inversor-mestre), sendo possível comandar, parametrizar e supervisionar o CFW-08 através desta. O protocolo utilizado nessa comunicação é do tipo pergunta/resposta (mestre/escravo) de acordo com as normas ISO 1745, ISO 646, com troca de caracteres do tipo ASCII entre o inversor e o mestre (no nosso caso o microcontrolador) com taxa de transmissão de 9600bps.

A seguir será mostrado como acontece a troca de mensagens entre mestre e escravo.

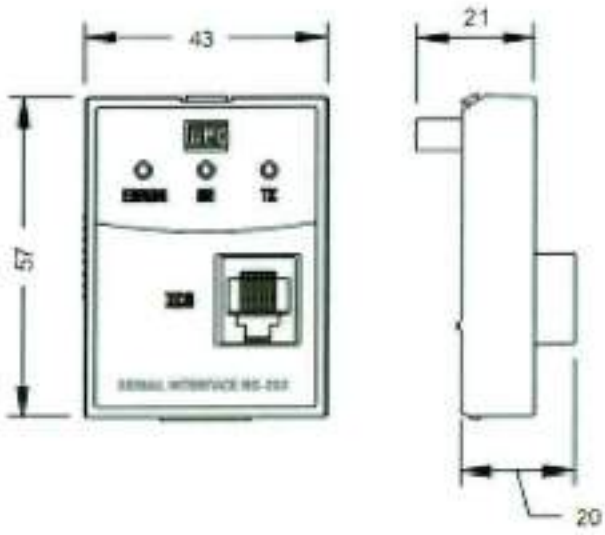


Figura 22. Dimensões do módulo de comunicação serial RS-232 KCS-CFW08.

6.6.2 Telegramas

A troca de mensagens entre mestre e escravo acontece através de dois tipos de pacotes:

1. Telegrama de leitura: no qual é solicitado o conteúdo de alguma variável do inversor.
2. Telegrama de escrita: no qual é solicitada uma alteração em alguma variável do inversor.

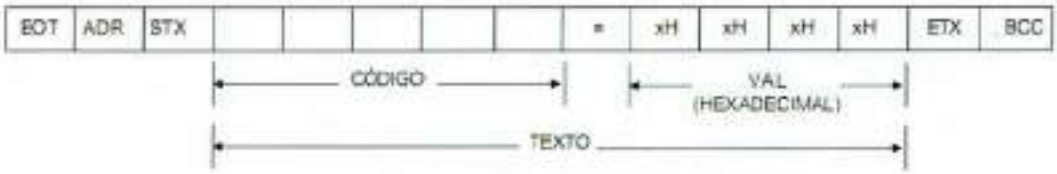


Figura 23. Telegrama de escrita.

Como só estamos interessados em controlar o inversor, utilizamos somente os telegramas de escrita que possuem o seguinte formato:

- EOT: caractere de controle (End of transmission) cujo valor é 0x04 (hexadecimal)
- ADR: endereço do inversor, que no nosso caso é "A"
- STX: caractere de controle (Start of Text) cujo valor é 0x02
- TEXTO: consiste em:
  - CÓDIGO: endereço da variável
  - = : caractere de separação
  - VAL: valor composto de 4 dígitos hexadecimais
- ETX: caractere de controle (End of Text) cujo valor é 0x03
- BCC: byte de checksum que é um ou exclusivo de todos os bytes entre STX(inclusive) e ETX(inclusive)

A seguir o código responsável pelo envio de um telegrama de escrita para o inversor:

```
int telegrama_escrita (char* codigo, char* valor) {
   char texto[16];
   char resp[3];
   int i;
   if (strlen(codigo) < 5 || strlen(valor) < 4) return (0);
```

```
    texto[0] = 0x04;//adicionado bit de paridade par
    texto[1] = 'A'; //endereco do inversor
    texto[2] = 0x02;
    texto[3] = *codigo;
    codigo++;
    texto[4] = *codigo;
    codigo++;
    texto[5] = *codigo;
    codigo++;
    texto[6] = *codigo;
    codigo++;
    texto[7] = *codigo;
    texto[8] = '-';
    texto[9] = *valor;
    valor++;
    texto[10] = *valor;
    valor++;
    texto[11] = *valor;
    valor++;
    texto[12] = *valor;
    texto[13] = 0x03;
    texto[14] = 0x00;
    for (i = 0; i < 14; i++) {
        texto[i] = calc_paridade(texto[i]);
    }
    for (i = 3; i <= 13; i++) {
        texto[14] = texto[14]^texto[i];
    }
    texto[14] = calc_paridade(texto[14]);
    texto[15] = 0x00;
    fputs(texto, serial1);
}
```

6.6.3 Variáveis básicas

Como visto, o telegrama de escrita possui um campo chamado CÓDIGO. É através dele que informamos o código da variável que desejamos alterar. E existe um conjunto de variáveis que só podem ser modificadas através da serial, denominadas variáveis básicas.

São através delas que podemos iniciar a rotação do motor, definir o sentido de giro, alterar a rotação e desligá-lo. Dentre essas variáveis duas foram utilizadas, a de código 00703 e 00704.

Através da variável 00703 iniciamos a rotação do inversor ou paramos, como pode ser visto a seguir:

```
//código da variável básica
cod[0] = '0';
cod[1] = '0';
cod[2] = '7';
cod[3] = '0';
```

```
cod[4] = '3';
cod[5] = 0x00;
//A seguir o valor dessa variável, que indica que é para
desabilitar o JOG
//sentido de rotação anti-horário
//habilitar a rampa e girar
val[0] = 0x70;
val[1] = 0x7F;
val[2] = 0x70;
val[3] = 0x73;
val[4] = 0x00;
//enviando o comando para o inversor
telegrama_escrita (cod, val);
```

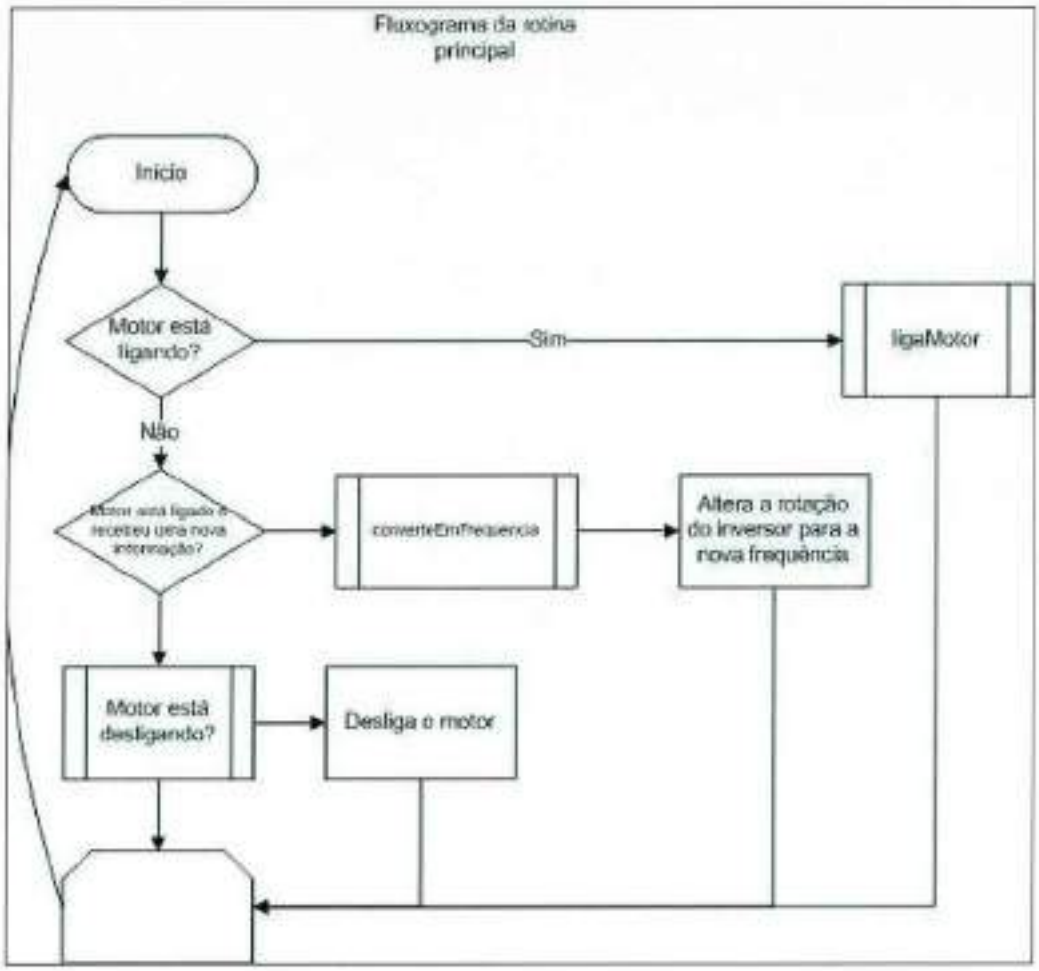
Já através da variável 00704 é possível alterar a rotação do inversor, como no código a seguir:

```
//cod para mudanca de velocidade
frequencia*=100l;
cod[0] = '0';
cod[1] = '0';
cod[2] = '7';
cod[3] = '0';
cod[4] = '4';
cod[5] = 0x00;

//é necessário um deslocamento de bits para preencher
corretamente o valor
i = (frequencia >> 12) & 0x0f;
val[0] = i + 0x70;
i = (frequencia >> 8) & 0x0f;
val[1] = i + 0x70;
i = (frequencia >> 4) & 0x0f;
val[2] = i + 0x70;
i = frequencia & 0x0f;
val[3] = i + 0x70;
val[4] = 0x00;
i = 1;

//limitando a frequencia para uma faixa que o inversor suporte.
if (frequencia > 300 && frequencia < 10000) {
 i = telegrama_escrita (cod, val);
}
```

6.6.4 Contexto do inversor

O inversor fará o papel do motor, controlando a rotação da roda fônica.

O fluxograma a seguir representa a rotina principal que controla a lógica do inversor:



Pelo fluxograma, percebe-se que existem quatro estados. Um é antes da partida (Ligando), outro após a partida (Ligado), um Desligando e finalmente o Desligado.

Inicialmente o gerenciamento irá esperar que o motor atinja a rotação equivalente a partida, como pode ser visto no trecho de código abaixo:

```
void ligaMotor()
{
        long int d=0;
            //colocando frequencia minima
            muda_rotacao_inteligente(5);
            //código da variável básica
            cod[0] = '0';
            cod[1] = '0';
            cod[2] = '7';
            cod[3] = '0';
            cod[4] = '3';
            cod[5] = 0x00;
            //A seguir o valor dessa variável, que indica que é para
desabilitar o JOG
            //sentido de rotação anti-horário
            //habilitar a rampa e girar
            val[0] = 0x70;
            val[1] = 0x7F;
            val[2] = 0x70;
            val[3] = 0x73;
```

```
            val[4] = 0x00;
            //enviando o comando para o inversor
            telegrama_escrita (cod, val);

            //quem irá alterar o estado do motor é o gerenciamento
            while(g_estado_motor == ligando)
            {
                  muda_rotacao_inteligente(g_frequencia_atual + 1);
                  d=2000/g_frequencia_atual;
                  d=(g_delay>=d)?g_delay:d;
                  delay_ms(d);
            }
}
```

Após atingir essa rotação o estado do motor passará a ser considerado como "ligado" e então o gerenciamento irá controlar a velocidade de rotação a partir da abertura da válvula borboleta, convertendo a abertura da válvula na frequência requerida através da seguinte rotina:

```
int converteEmFrequencia(int vb_abertura)
{
      int freq;
      float temp;
      //OBS: vou impor que a abertura inicial da valvula borboleta
corresponda a 1,83V(o que corresponde a 400 RPM)
      //valor do sensor da valvula borboleta vai de 80 a 177, entao estou
normalizando a avertura para ir de 3 a 80(2400RPM)
    temp = 0.761*vb_abertura-54.761;
    if (temp < 3)
        freq = 3;
    else if (temp > 100)
        freq = 100;
    else
        freq = temp + 0.5;  // somando 0.5 antes de truncar estamos
arredondando corretamente
      return freq;
}
```

Através do código anterior, percebe-se a característica linear entre a abertura da válvula e a velocidade de rotação.

6.7 Sensores e Entradas

Para simular algumas ações externas sobre o sistema foi utilizado um painel com quatro dispositivos: Marcha, Pedal de aceleração, botão de liga e desliga do motor, e o breque.

Além destes sensores e entradas existe o sensor de pressão, o MAP, que mede a pressão do ar dentro do coletor de admissão entre a válvula borboleta e o motor. Esse sensor não foi utilizado pela dificuldade de controlar a variação de pressão desejada para o projeto. A massa de ar simplesmente será feita pela

abertura da válvula borboleta.

Através do I^2C o gerenciamento pode obter o estado de cada um deles e atuar nas outras partes do sistema.

O pedal de aceleração é constituído por dois potenciômetros independentes que são variáveis de acordo com sua posição. Sendo que um sinal é de redundância para segurança do motorista.

A Figura 25 mostra um exemplo do circuito interno do pedal de aceleração. Observe que há 6 pinos sendo que 3 pinos são de um potenciômetro e outros 3 de outro.

Foi realizado a medida da resistência entre todos os pinos e observamos que os potenciômetros estão nos pinos 1-2-6 e 3-4-5. Os testes da resistência com o pedal em repouso e acelerando até o final estão indicados nas tabelas abaixo.

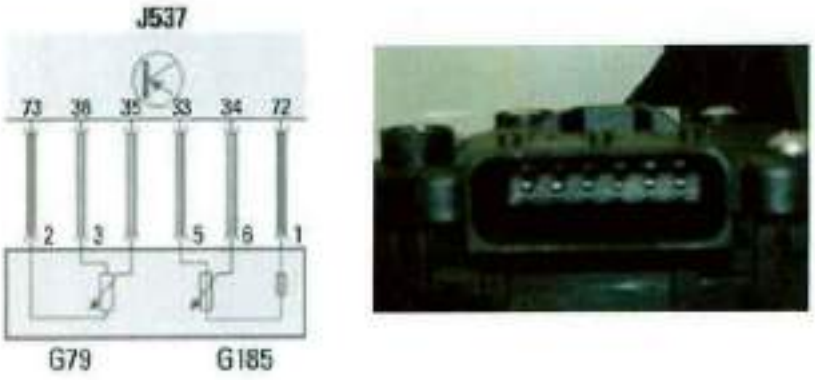


Figura 24. Figura demonstrativa do circuito interno do pedal.

Figura 25. Pedal de aceleração desmontado.

Circuitos dos pinos 1-2-6:

Pinos	Sem acelerar(kΩ)	Acelerando(kΩ)
1-2	1,26	1,69
1-6	2,59	2,19
2-6	1,66	1,66

Circuitos dos pinos 3-4-5:

Pinos	Sem acelerar(kΩ)	Acelerando(kΩ)
3-4	1,31	1,7
3-5	0,92	0,92
4-5	1,79	1,38

A alimentação indicada para o pedal será de 5V, então a corrente máxima para a menor resistência será aproximadamente de 5,5mA. A entrada do microcontrolador suporta essa corrente (máxima de 25mA).

Para o desenvolvimento do software foram utilizadas as entradas analógicas RE2 e RA2 do conector de expansão do kit, mostrado na Figura 26.

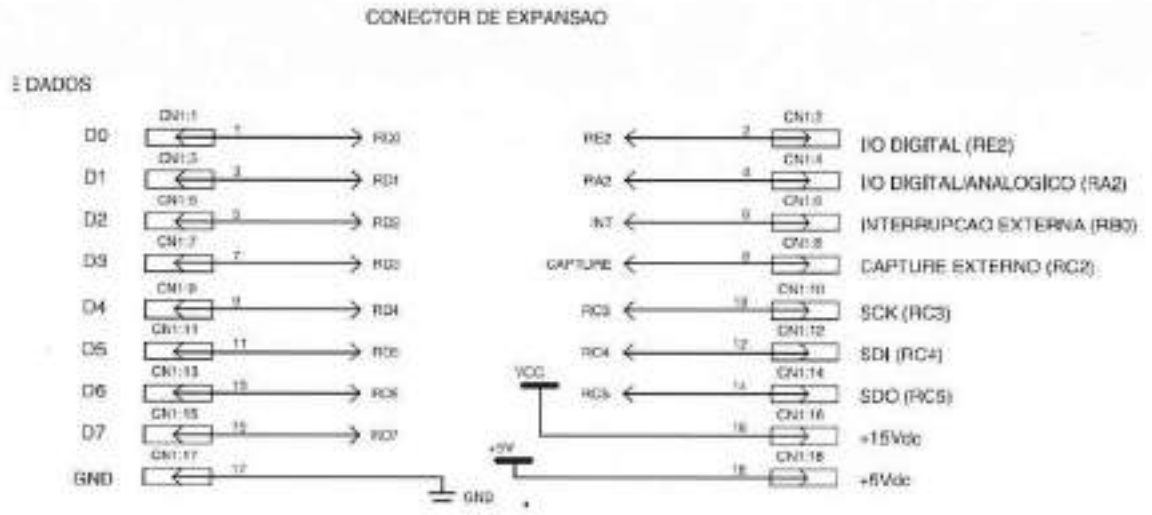


Figura 26. Circuito de expansão do kit McLab2.

Após o teste do pedal foi desenvolvido uma placa similar com o kit mclab2, disponibilizando, além das portas necessárias para o projeto, outras portas e 3 potenciômetros para futuras implementações ou simulações de outros sensores.

O breque é constituído por um interruptor com duas saídas em níveis lógicos diferentes. Quando o breque é acionado uma saída está em aberto e outro em curto e caso não esteja acionado ocorre o contrário.

As portas utilizadas do microcontrolador foram: RA0 e RA1 para o sinal do pedal de aceleração; RB1 e RB2 para o freio; RD0 a RD4 para a chave de posição para simular as marchas; as portas RC3 e RC4 para a comunicação por I2C e as portas RB3 e RB4 para os botões de liga/desliga do motor.

O programa para fazer o layout da placa foi o Eagle 5.0.0 da Cadsoft e o seu esquemático e o board são mostrados nas figuras a seguir:

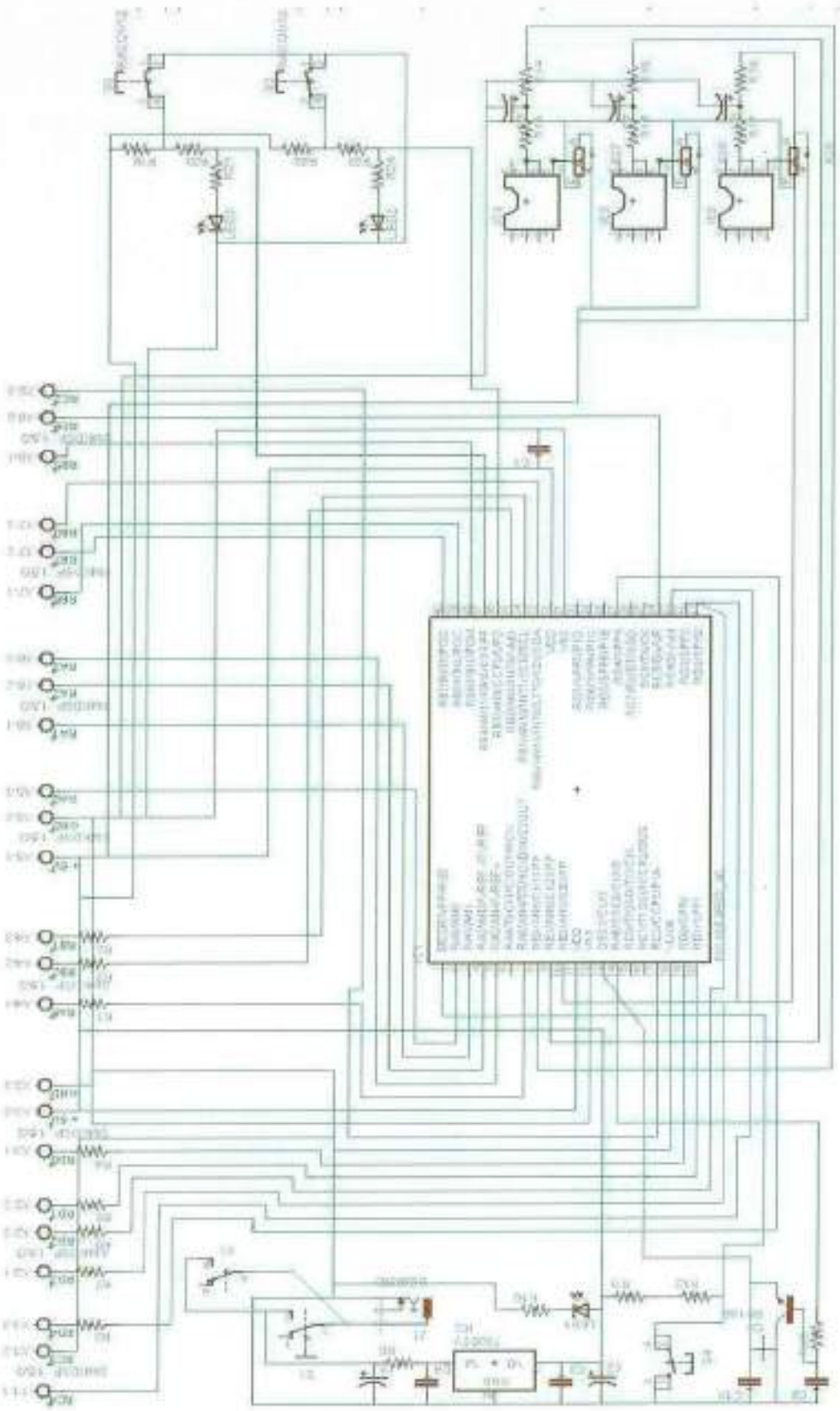


Figura 27. Esquemático do circuito dos sensores.

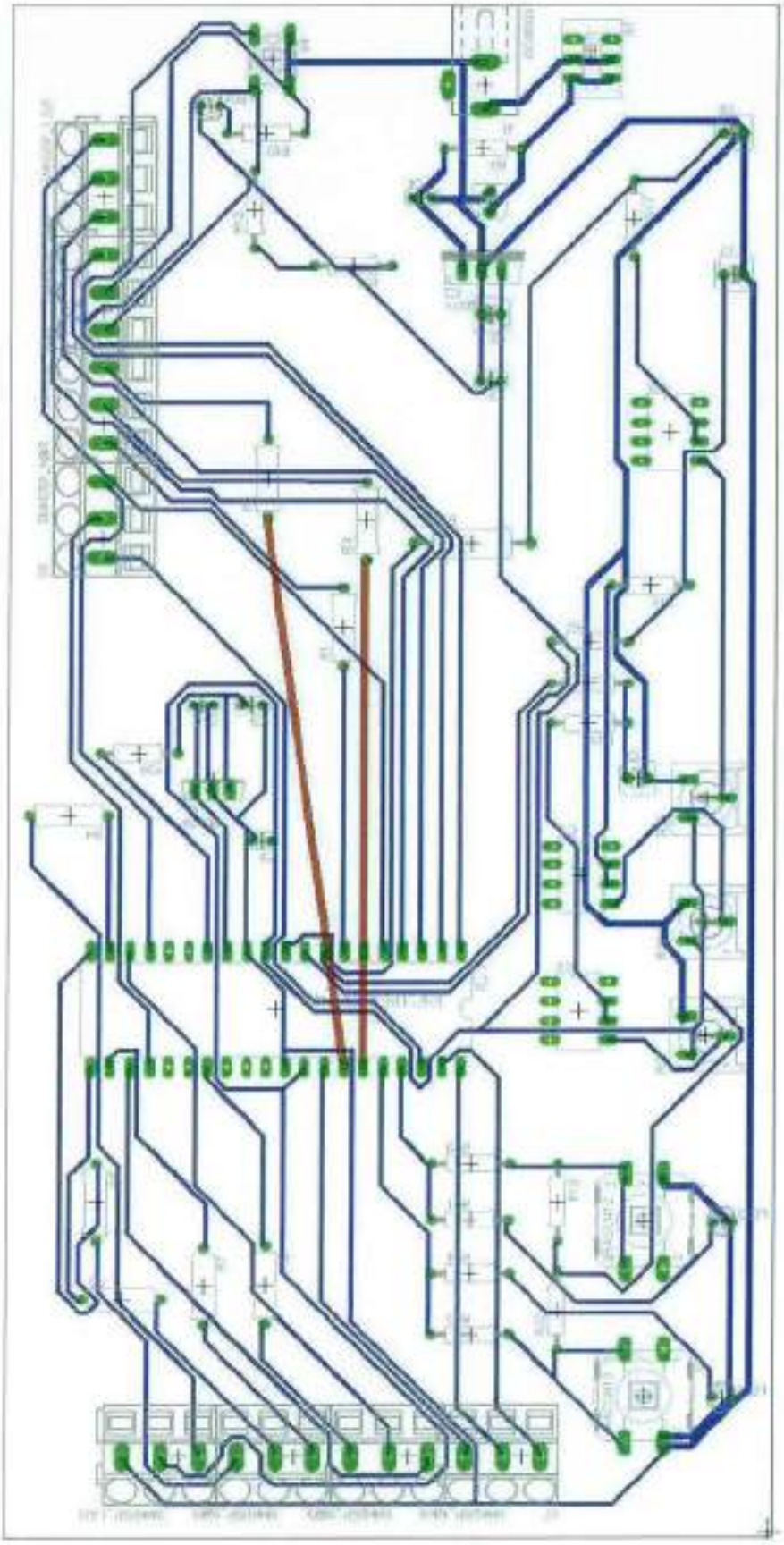


Figura 28. Layout do circuito dos sensores.

Para o desenvolvimento do software os dados dos sensores serão enviados através de 2 bytes de dados sendo que:

- o 1° byte é o valor do potenciômetro do pedal e o
- 2° byte é dividido com os 4 primeiros bits para o valor da marcha, o 5° e 6° bit o valor do freio (01 para pressionado e 00 caso contrário) e o 7° e 8° para o liga/desliga do motor (01 para o motor ligado e 00 caso desligado)

Abaixo é mostrado como ficou implementado fisicamente os sensores e entradas.

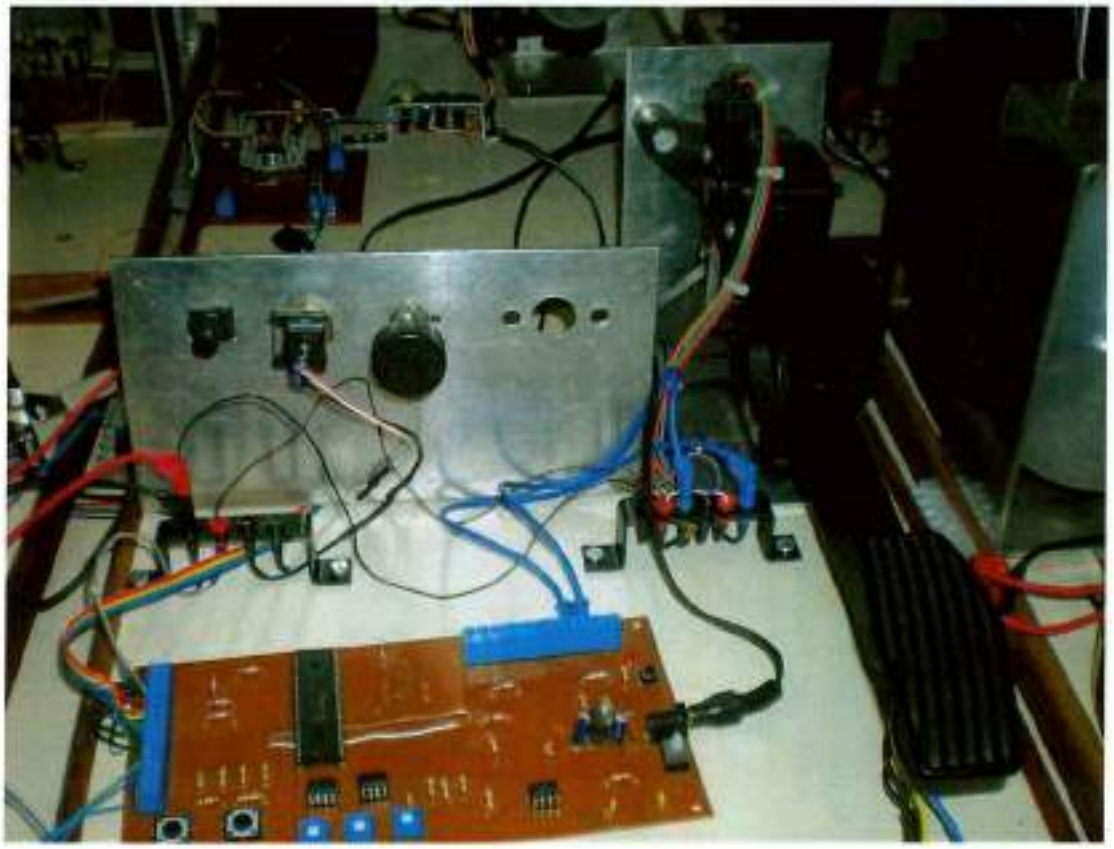


Figura 29. Sensores e entradas

6.8 Macro-gerenciamento

O gerenciamento é o responsável por coordenar o funcionamento do sistema, como podemos ver no fluxograma da Figura 29.

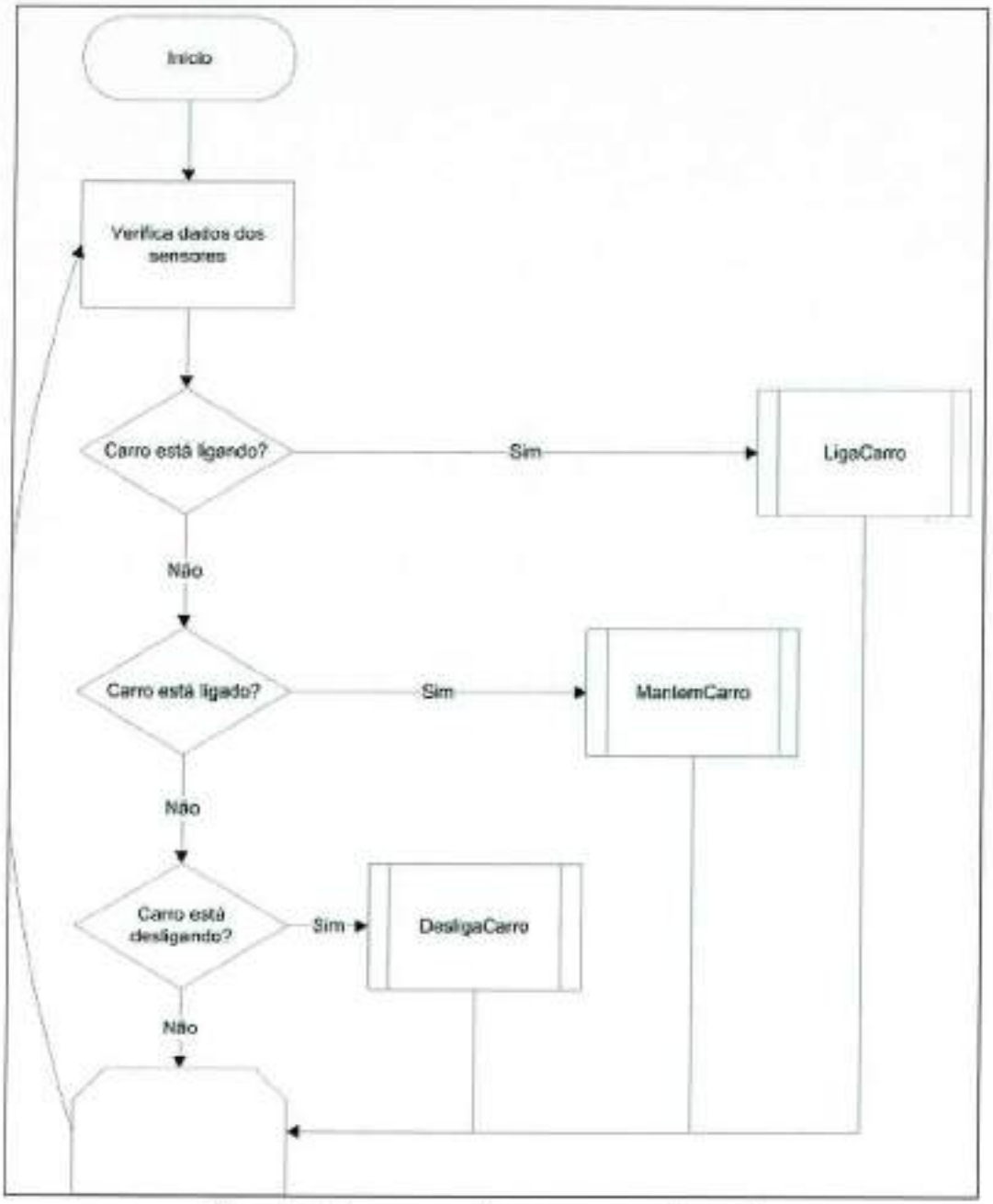


Figura 30. Fluxograma do macro-gerenciamento.

O laço principal correspondente ao fluxograma é o seguinte:

```
while(true)
{
      //caso tenha acontecido alguma colisão, este bit será setado. Entao
só por software que devemos limpá-lo.
      if(SSPCON_WCOL) SSPCON_WCOL=0;
      //Se um novo byte foi recebido antes do registrador SSPBUFF ser lido,
este bit será setado e só por software devemos limpá-lo.
      if(SSPCON_SSPOV) SSPCON_SSPOV=0;
```

```
        //primeiro eu verifico os sensores
        verificaParametros(estadoMotor);
    lcd_Atualiza();
        //aqui sinalizamos que o carro foi ligado, ou seja, a chave foi
ativada.
        if( (estadoMotor == desligado) && (g_motorLigado) )
        {
                iniciaVisor();
                estadoMotor=ligando;
        }
        else if(estadoMotor==ligado && !(g_motorLigado) )
        {
                estadoMotor=desligando;
        }
        switch(estadoMotor)
        {
                case ligado:
                        mantemMotor();
                break;
                case ligando:
                        ligaMotor();
                        estadoMotor=ligado;
                break;
                case desligando:
                        desligaMotor();
                        estadoMotor=desligado;
                break;
                case desligado:
                    delay_ms(1);
        }
}
```

A todo instante o gerenciamento irá consultar os seguintes parâmetros:

- Válvula borboleta
- Pedal
- Breque
- Sensor Hall
- Chave liga/desliga

A seguinte rotina é a responsável por obter esses parâmetros. Oberve a necessidade de checagens dos dados obtidos através dos sensores, para que eles estejam no limite imposto pela Tabela 4:

```
void verificaParametros(ESTADO_MOTOR estadoMotor)
{
      int dados[3];
```

```
        int estadoAtual=0;
        int temp;

        //verificando os dados dos sensores
        i2c_recebe(ENDERECO_SENSORES,2,dados);

        //checagem
        if( (dados[1] == 0x00) || (dados[1]==0xFF) ) return;
        if( (dados[0] < 15) || (dados[0] > 100)) return;

        g_breque=((dados[1] & 0b00010000) == 0b00010000);
        g_pedal=dados[0];
        g_marcha=dados[1] & 0b00001111;
        estadoAtual=((dados[1] & 0b01000000) == 0b01000000);
        if(estadoAtual==ultimoEstado)
        {
        redundancia++;
        if(redundancia>=3)
        {
              redundancia=0;
              g_motorLigado = estadoAtual;
        }
        }
        else
        {
          redundancia=0;
        }
        ultimoEstado = estadoAtual;

        //obtendo o valor da rotacao
        if(estadoMotor != desligado)
        {
              redundancia = 0;
              temp = 0;
              while (redundancia < 3)
              {
                    temp=mede100RPM();
                    if (temp < g_rotacao - 1 || temp > g_rotacao + 1)
                          redundancia = 0;
                    else
                    redundancia++;
                    g_rotacao = temp;
            }

            i2c_recebe(ENDERECO_BORBOLETA,1,dados);
            if(dados[0]>=80 && dados[0]<=177)
            {
               g_sensorBorboleta=dados[0];
            }
        }
}
```

E existem as rotinas principais que correspondem aos estados do carro:

- ligaCarro: Na qual o Gerenciamento irá inicialmente impor a rotação correspondente ao arranque (400 RPM). Enquanto essa rotação não é atingida, nem a ignição nem a injeção funcionam. Ao atingir essa rotação, o próximo passo e atingir a rotação de marcha lenta (rotação 800 RPM). Nesse momento é solicitado que os bicos injetores fiquem abertos constantemente, a ignição já começa a funcionar e a velocidade de rotação agora é dependente somente da

abertura da válvula borboleta.

- mantemCarro: Nesse estado o Gerenciamento irá, a partir do sinal do pedal, alterar a abertura da válvula borboleta através da equação "*abertura = 2.877*pedal-9.05*" que por sua vez irá alterar o tempo de injeção através da equação "*tempo = 1.343* abertura -112.69*" velocidade de rotação. Nesse momento, quando o pedal estiver completamente pressionado, a rotação máxima será de 2400 RPM.
- desligaMotor: Neste momento o gerenciamento irá fechar a válvula borboleta, fechar os bicos injetores e solicitar para o motor que pare.

Pela Tabela 4 é possível verificar os valores adotados para a calibração do sistema.

Tabela 2. Tabela com faixas de valores dos sensores e atuadores.

	ROTAÇÃO	INVERSOR	PEDAL ESQ	PEDAL DIR	VB	SENSOR VB	INJEÇÃO
mínimo	90	3	24	49	10	85	21
máximo	3000	100	81	163	224	177	125
range	2910	97	57	114	214	92	104
MOTOR_LIGAR_RPM	400	13	---	---	10	89	---
MOTOR_MARCHALENTA_RPM	800	27	24	49	60	107	31
rotação máxima adotada	2400	80	81	163	224	177	125
referência	---	SENSOR VB	---	---	PEDAL ESQ	---	SENSOR VB
"a" (y=a*x+b)		0,76136364			2,87719298		1,342857143
"b" (y=a*x+b)		-54,761364			-9,0526316		-112,6857143
verificação		80			224		125

Na Figura 30 é possível verificar como foi implementado o subsistema de macro-gerenciamento. Foi utilizada a placa didática McLab2 e a partir dela foram utilizados resistores de Pull Up no barramento do I^2C.

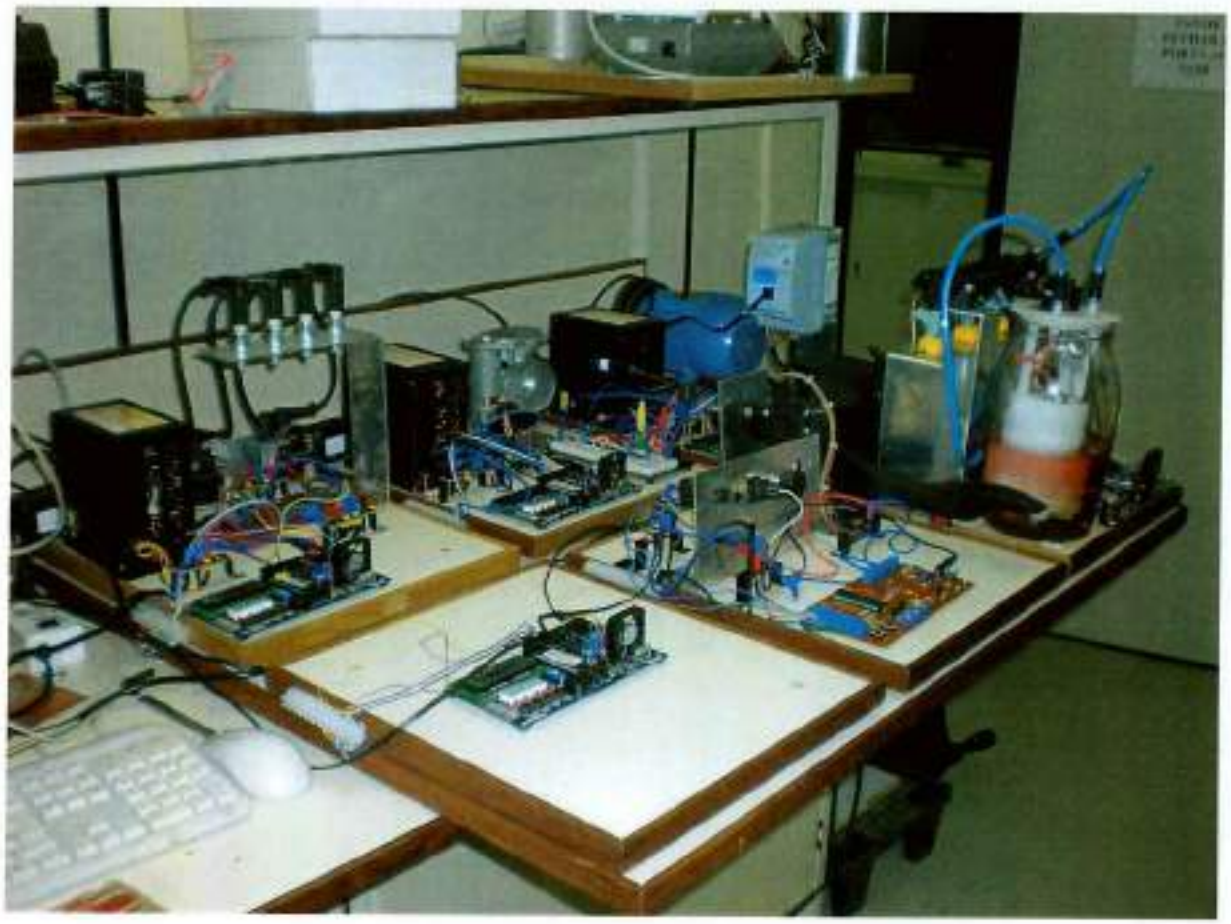


Figura 31. Subsistema de macro-gerenciamento.

6.9 Outras atividades

Durante o desenvolvimento das atividades mencionadas, houve a necessidade de reestruturar e aperfeiçoar a montagem dos módulos já existentes, bem como criar um barramento que os interligasse.

Foram substituídas todas as fontes de alimentação dos protótipos e instalados suportes com conectores, de modo a organizar os cabos de alimentação, aquisição e envio de sinais, conforme as necessidades específicas de cada um dos subsistemas.

7 Funcionamento integrado

O diagrama de blocos do projeto demonstrado na prática será:

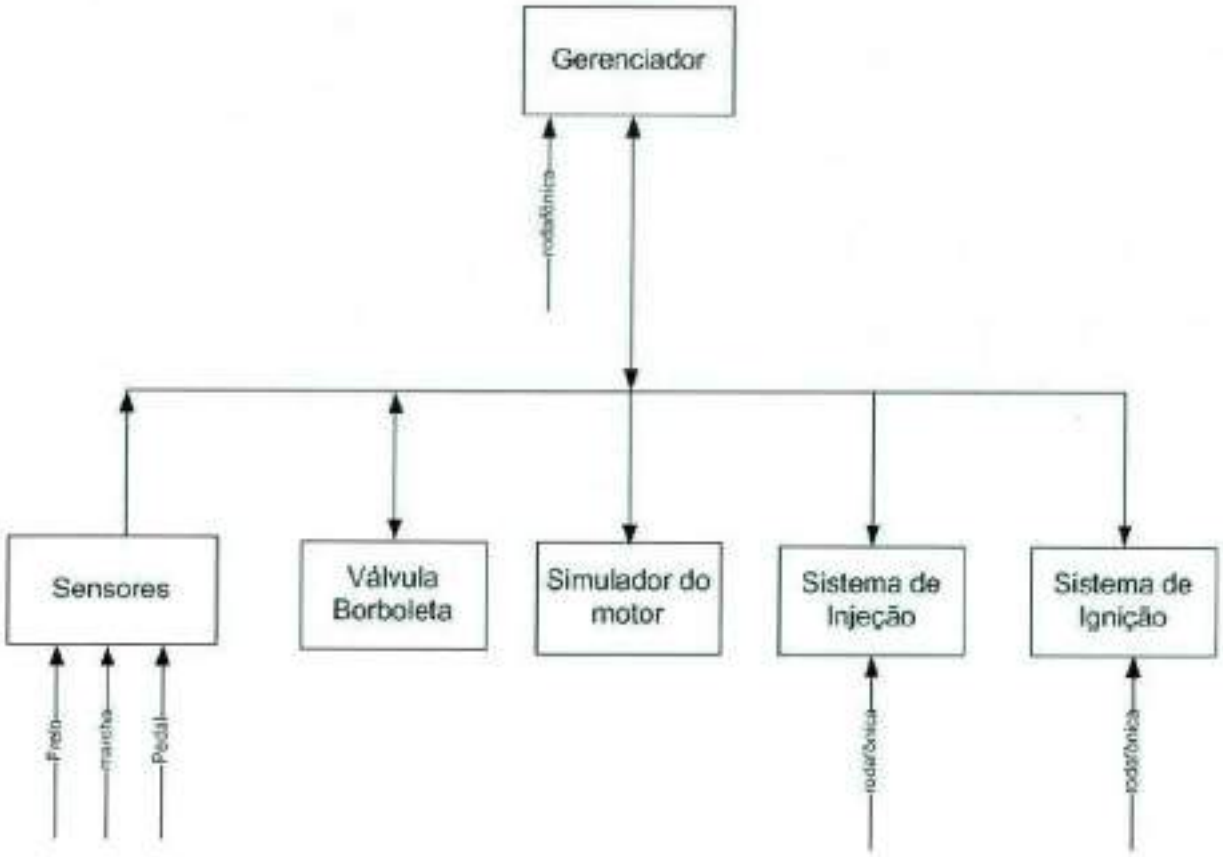


Figura 32. Diagrama de blocos do sistema

Em Sensores pode-se ler Sensores e Entradas pois é nela que estará situada o botão de liga e desliga do motor.

O começo da parte prática foi feito através do botão de ligar na parte de Sensores e Entradas. O motor é ligado até chegar a rotação do motor de arranque ou de partida adotado, 400rpm. A partir deste momento os bicos injetores injetarão continuamente o combustível para obter uma mistura mais rica e começará a faísca no subsistema de ignição até chegar a rotação de marcha lenta, 800rpm. Na marcha lenta o gerenciamento obterá o sinal do pedal de aceleração para controle dos atuadores do sistema. O subsistema de injeção atuará de acordo com rotação e o subsistema de ignição ajustará os ângulos de avanços.

A próxima figura mostra o sistema funcionando no dia da demonstração

pratica.

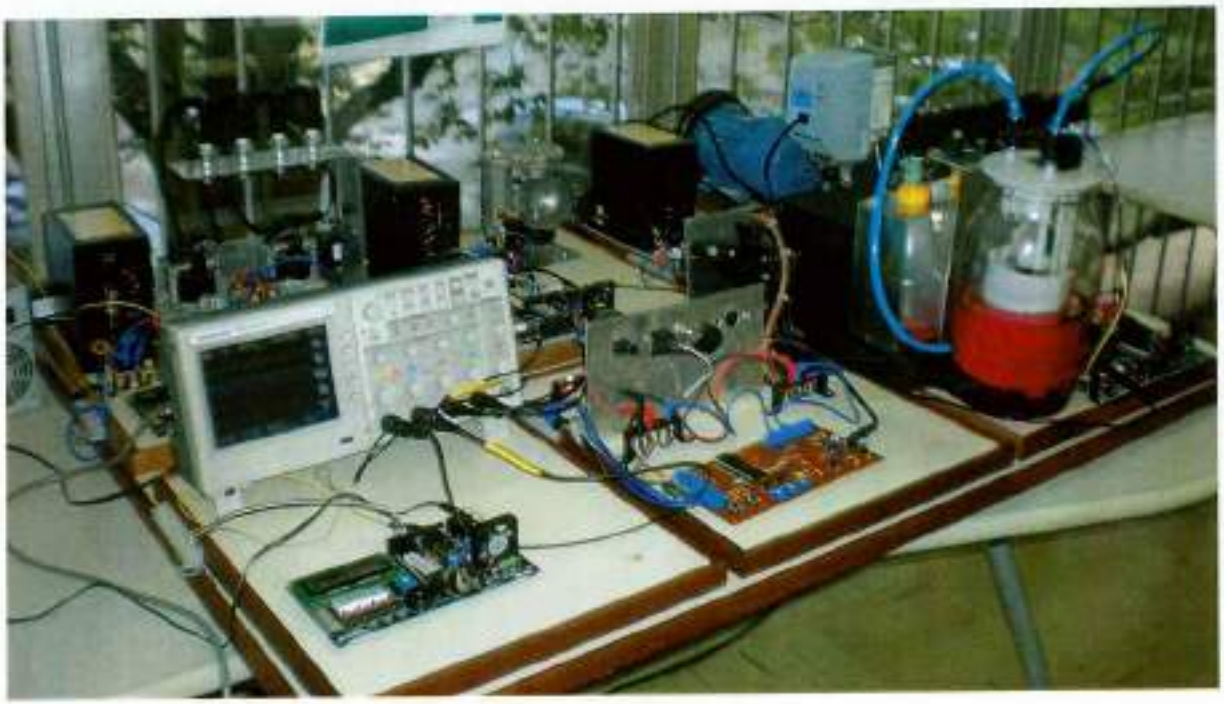


Figura 33. Foto geral na demonstração prática

8 Conclusão

Atualmente discute-se muito sobre as emissões de CO_2 na atmosfera, e o Brasil por ser um dos maiores produtores de automóveis do mundo tem a sua grande importância com essas questões. Uma das evoluções tecnológicas foi a utilização de um gerenciamento da injeção eletrônica num carro que fez diminuir a quantidade de gazes nocivos para o meio ambiente.

Então o projeto proposto está bem inserido nas questões ambientais e pelo seu sistema descentralizado oferece uma maior didática para auxiliar disciplinas relacionadas com sensores, atuadores e microcontroladores. Assim uma maior capacitação de pessoas para seguirem no ramo automotivo.

Verificou ao longo do projeto algumas melhorias que serão citadas posteriormente para a continuação do trabalho como a troca por microcontroladores de maior desempenho e a inclusão de mais sensores.

9 Sugestões para projetos futuros

Ao longo de todo o trabalho, pudemos observar diversos pontos de possíveis melhorias no sistema desenvolvido. Devido ao tempo escasso para a realização do projeto, várias dessas melhorias tiveram que ser deixadas de lado, para que pudéssemos focar nossos esforços em alcançar os resultados previstos e estabelecidos como escopo do projeto.

Sendo assim, gostaríamos de deixar estas idéias como sugestões para trabalhos futuros, que possam tomar por base este trabalho e estendê-lo ou melhorá-lo:

- A substituição dos microcontroladores utilizados por outros de maior desempenho, permitindo que se trabalhe em faixas maiores de rotação;
- A inclusão de um número maior de sensores, de modo a se obter um controle mais refinado e mais próximo do realizado por sistemas de injeção eletrônica comerciais;
- Testes em um motor real (*mockup*), validando o funcionamento do sistema desenvolvido sobre um motor a combustão interna real;
- Testes de outros protocolos de comunicação, de modo a tornar a comunicação entre os módulos mais robusta.

10 Referências bibliográficas

[1] WIKIMEDIA COMMONS. **Diagrama Pressão x Volume de um ciclo ideal de Otto**. Disponível em: <http://commons.wikimedia.org/wiki/File:Otto_PxV.png>. Acesso em: 28 jun. 2009.

[2] ENVENENADO. **Como funciona – Injeção eletrônica**. Disponível em: <http://www.envenenado.com.br/howwork/injecao/injecao.html>. Acesso em: 14 mai. 2009.

[3] MANAVELLA, Humberto J. **Controle integrado do motor**: sistemas de injeção-ignição eletrônica. [São Paulo: s.n.]

[4] PENIDO FILHO, Paulo. **Os motores a combustão interna**: para curso de máquinas térmicas, engenheiros, técnicos e mecânicos em geral que se interessam por motores. Belo Horizonte: Lemi, 1991. 2v.

[5] PEREIRA, Fábio. **Microcontroladores PIC – Programação em C**. 7ª ed. São Paulo: Editora Érica, [200?].

[6] MICROCHIP. **PIC16F87XA Data Sheet**. U.S.A.: [s.n.], 2003.

[7] NATIONAL SEMICONDUCTOR. **LM 1949 Injector Drive Controller**. [s.l.: s.n.], 2001.

[8] IRAZABAL, Steve Blozis Jean-Marc. **I^2C bus**. San Jose, Mar 2003. 24p. AN10216-01 – I^2C Manual.

[9] WEG. **Manual do inversor de frequência CFW-08**. [s.l.: s.n.], Ago 2003. 167p. Revisão 6.

ANEXO I

Em anexo a esta monografia, incluímos um CD com os códigos-fonte desenvolvidos para todos os módulos, além de material multimídia (vídeos e fotos) demonstrando o funcionamento do sistema.