Telemetria Automotiva via Internet Móvel

[1]F. Teixeira; [1]M. C. de Oliveira; [1]A. L. Helleno

[1]Universidade Metodista de Piracicaba, Faculdade de Engenharia Arquitetura Urbanismo.
fernando.crt@gmail.com; mceliamat@yahoo.com.br ;alhelleno@gmail.com

***Resumo* – O presente trabalho tem como objetivo desenvolver um sistema de telemetria automotiva via tecnologia de internet móvel. Este trabalho caracterizou-se inicialmente pelo estudo das tecnologias disponíveis para monitoramento e diagnóstico em um automóvel, como o padrão OBD (*On-board Diagnostics*), e os protocolos usados para comunicação das diversas ECUs (*Engine Control Units*) presentes no automóvel. Também é estudada a tecnologia de conexão à internet móvel GPRS, na qual o sistema foi implementado, e os softwares utilizados para a execução da telemetria. O sistema de Telemetria foi validado por meio de testes de comportamento do sistema.**

***Abstract* – This work aims to develop an automotive technology telemetry system via mobile internet. This work was characterized initially by the study of the technologies available for monitoring and diagnosis a car, as the standard OBD (On-Board Diagnostics), and protocols used to communicate with the various ECUs (*Engine Control Units*) present in the car. Also studied is the technology of GPRS mobile internet connection, in which the system was implemented, and the software used for the implementation of telemetry. The telemetry system was validated by testing of system behavior.**

I. INTRODUÇÃO

O controle das funções de veículos para diagnosticar, em tempo real, problemas no motor pode representar um atividade importante, especialmente em veículos usados para transporte de cargas. Este controle permite diagnosticar ou prevenir problemas graves de forma antecipada. Atualmente, este controle é feito por meio dos sistemas OBD. Tais sistemas são utilizados na maioria dos automóveis leves e pesados que circulam atualmente nas estradas.

Este tipo de controle data dos anos 70 e início dos anos 1980, quando os fabricantes começaram a utilizar meios eletrônicos para controle de funções do automóvel e diagnosticar problemas no motor em tempo real, motivados também, para atender aos padrões de emissão de poluentes da agência U.S. *Environmental Protection Agency* (EPA). O OBD-II (ou OBD2) é o padrão mais recente da indústria automotiva que foi introduzido nos modelos de veículos a partir de 1996, fornecendo o controle do motor quase por completo, também monitorando as partes do chassi e dispositivos acessórios, e a rede de controle de diagnóstico do veiculo.

Dada a importância do monitoramento, do diagnóstico e da evolução dos sistemas ao longo do tempo, o objetivo deste estudo é desenvolver um sistema (sendo este hardware e software) capaz de acessar a rede CAN do veiculo, e enviar alguns dados da rede para um servidor utilizando a infraestrutura da rede de telefonia celular GSM/GPRS.

O objetivo principal desta interface é permitir ao usuário ou ao operador de manutenção acesso aos dados do veículo e o respectivo status de funcionamento, de maneira remota.

Para atender ao objetivo este artigo está dividido em cinco tópicos: introdução; revisão bibliográfica com as questões teóricas que sustentam o desenvolvimento deste estudo; proposta do sistema de transmissão; análise e discussão dos resultados; e, finalmente, as conclusões.

II. REVISÃO BIBLIOGRÁFICA

A telemetria surgiu no ano de 1845 com o desenvolvimento do primeiro sistema de transmissão de dados entre o *Winter Palace* e o quartel da armada russa [1]. Em 1874 foi desenvolvida uma rede de sensores atmosféricos em *Mont Blanc*, França, com transmissão de dados em tempo real para Paris. Mais tarde, em 1906 foram montadas umas séries de estações sísmicas na Rússia, com transmissão de dados para o *Pulkovo Observatory* [1].

Um sistema de Telemetria consiste na medição de dados de um analisador e envio desses dados para um acumulador para posterior análise. Esta é uma tecnologia muito empregada em diversas áreas da ciência, com a medicina [2] e [3], e até mesmos parques temáticos [4].

Sistemas de telemetria via rede de telefonia móvel, já existem há tempos, conforme apresenta [5]. Ao movimentar-se, um dispositivo de telefonia móvel troca a central de comunicação com a qual mantém conexão constantemente, mudando assim seu endereçamento, tornando este sistema de comunicação bastante confiável. A rede de telefonia móvel, também conhecida como tecnologia *Global System for Mobile communications* (GSM), compreende toda a infraestrutura para ofertar os serviços de transmissão de dados, em quase todo o globo, inclusive nos locais de difícil acesso. Para isso, utiliza-se de operadoras de telefonia celular que mantêm a concessão dos serviços [6].

O protocolo *General Packet Radio Service* (GPRS) é uma tecnologia de transmissão de dados disponível em redes GSM, cuja taxa de transmissão de dados típica é de 26 a 40kbits. A maior parte do território nacional possui infraestrutura de rede de telefonia celular GSM/GPRS [6]. De modo geral,

podemos dizer que a rede GPRS cobre a mesma região coberta pelo GSM [7].

A telemetria automotiva é amplamente utilizada em veículos de uso comercial e máquinas agrícolas [6], onde é instalado um computador de bordo e vários sensores no veiculo, com a função de extrair informações tais como identificação do motorista, quantidade de combustível consumida, distância dirigida, locais visitados, rotas utilizadas, duração das viagens, velocidade, bem como uma série de eventos customizados pelo cliente tais como freada brusca, aceleração brusca, porta aberta, direção na chuva, etc. A vantagem deste sistema é que as informações críticas podem ser transmitidas em tempo real para o gerente da frota, contribuindo para reduções de custo de consumo de combustível, pneus, manutenção bem como redução no número de acidentes. A desvantagem deste sistema é o custo, por causa da grande quantidade de sensores que necessitam ser instaladas no veiculo, dependendo da qualidade da instalação, poderão influenciar no funcionamento de componentes eletrônicos do veiculo, com, por exemplo, falhas no alarme, ruído no sistema de som e fuga de corrente, ocasionando descargas eventuais da bateria reduzindo sua vida útil.

A rede de comunicação automotiva possui todos os parâmetros necessários para a execução de uma telemetria, com o beneficio de utilizar os sensores do veiculo, sem a necessidade de instalação de sensores e grandes intervenções no sistema elétrico do veiculo. No próximo item serão apresentadas as características e aplicações da rede de comunicação automotiva.

A. Rede de Comunicação Automotiva

Atualmente os veículos são equipados com um grande número de unidades de controle eletrônico que, para realizar as suas funções necessitam de um intenso intercâmbio de dados e de informações.

Os métodos convencionais de comunicações de dados ponto a ponto conectados através de um fio, já atingiu as suas limitações. Por um lado, já é quase impossível lidar com a complexidade dos chicotes, o que inviabiliza e limita o desenvolvimento de unidades de controle eletrônico. A solução para tais limitações está na empregabilidade de sistemas de barramentos seriais de dados, próprios para o uso veicular, sendo ele a rede CAN, que atualmente se tornou um padrão para a indústria automobilística.

Existem três áreas de aplicação de redes no veiculo, cada uma com seus próprios requisitos, sendo eles: rede multimídia, aplicação multiplex e aplicação em tempo real.

Rede multimídia: aplicações de rede para comunicações móveis conectam componentes, tais como sistema de navegação, telefone ou equipamentos de áudio e vídeo com visualizador central e unidades operacionais como, por exemplo, um computador de bordo. Nesta aplicação, para proteger outros sistemas de rede do veículo contra acesso não autorizado e interferências externas como, sistema de som, canal móvel, internet, os domínios de redes são separados por restrições de acesso (*Firewall*).

Aplicação multiplex: as aplicações do tipo multiplex são adequadas para controle e regulagem de componentes eletrônicos na área de carroceria e conforto, como por exemplo, controle da climatização, controle central de travas e posição dos bancos. As taxas típicas de transmissão de dados estão entre as velocidades de 1kbit/s e 125kbit/s [8]. Para reduzir os custos nesta área de aplicação foram desenvolvidos vários caminhos alternativos, como por exemplo, conexões econômicas ponto a ponto que podem ser empregadas entre o alternador e controle eletrônico do motor e redes locais, com taxa de transferências de 20kbit/s, podendo ser instaladas até mesmo nas portas dos veículos.

Aplicação em tempo real: as aplicações em tempo real interligam redes de sistemas eletrônicos, como gerenciamento de motor, troca de marchas e programa eletrônico de estabilidade que serve para controlar a tração e o movimento do veiculo [9]. As taxas de transmissão desta rede ficam entre 125kbit/s e 1Mbit/s para garantir a operação em tempo real [10].

Para integrar e otimizar todos estes sistemas, faz-se necessário à utilização de um sistema de barramento de dados para garantia da confiabilidade e velocidade da informação trafegando na rede veicular. Para este fim, o sistema de barramento de dados CAN, é utilizado como padrão para a aplicação veicular.

No próximo item serão apresentadas as características e aplicações do barramento de dados CAN.

B. Controller Area Network (CAN)

O CAN é um dos mais bem sucedidos protocolos de comunicação de dados em série com fio, tanto em termos de volumes de aplicação, como em termos de aceitação entre ramos da indústria, considerando os últimos vinte anos [11].

O CAN em aplicação veicular possui uma arquitetura de sistema focado em consumo reduzido de energia e recursos, resultando em redução de emissões de gases poluentes ([12] e [13]).

O protocolo CAN trabalha de acordo com o princípio de conexão *multi-master* [10], no qual várias unidades de controle eletrônico de mesmo nível de prioridade são interconectadas através de uma estrutura linear. A vantagem da conexão *multi-master* está no fato de que uma falha em algum dos integrantes da rede não impede o acesso aos demais. Em comparação com outras redes, a probabilidade de falha total é reduzida. Na rede circular ou estrela, a falha de um dos integrantes ou na unidade central é suficiente para causar falha geral do sistema.

O CAN utiliza o endereçamento baseado na mensagem. Para isso, um identificador (cabeçalho ou endereço) fixo é designado para cada mensagem. O identificador caracteriza o conteúdo da mensagem, como por exemplo, rotação do motor. Cada estação ou ECU (*Engine Control Unit*) processa exclusivamente as mensagens, os identificadores constam na sua lista de aceitação (Filtro de mensagens) [14]. Assim, o CAN não requer o endereçamento de estações para transmissão de dados e as estações não são envolvidas na configuração do sistema, facilitando o controle das diversas versões do equipamento.

O protocolo CAN é baseado em dois estados lógicos: Os bits são ou recessivos (nível lógico 1), ou dominantes (Nível lógico 0). Se um bit dominante é enviado por pelo menos uma estação, os demais bits recessivos enviados simultaneamente pelas outras estações são sobrescritos [10]. O identificador define ao mesmo tempo o conteúdo dos dados e a prioridade da mensagem. Um identificador correspondente a um número binário de baixo nível possui alta prioridade e vice-versa [10]. Quando o barramento está desocupado, qualquer estação pode

começar a transmitir suas mensagens. Se várias estações começam a transmitir simultaneamente começará a haver conflitos de acesso ao barramento. Para solucionar este problema o sistema usa um sistema de arbitragem. A mensagem com maior prioridade recebe acesso primeiro sem perda de bit ou atraso. Todo transmissor que perde a arbitragem é comutado automaticamente para o receptor e repete a sua tentativa de transmissão até que o barramento esteja livre. A Figura 1 apresenta o formato da mensagem CAN:

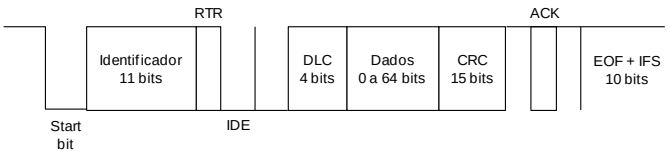

Figura 1. Formato da mensagem CAN.

O registro de dados é composto de sete campos consecutivos:

O campo *Start Bit* indica o começa da mensagem e sincroniza todas as estações.
O campo de arbitragem é composto pelo identificador da mensagem e bits adicionais de controle. Durante a transmissão deste campo, o emissor verifica a cada bit, se ele ainda detém prioridade ou se uma estação com maior prioridade esta transmitindo.
O bit de controle determina se a mensagem é classificada como registro de dados ou registro remoto.
O campo de dados dispõe de um conjunto de informação entre 0 e 8 bytes. Uma mensagem de comprimento 0 pode ser usada para sincronização dos processos distribuídos.
O campo CRC (*Cyclic Redundancy Check*) contém uma palavra para verificar possíveis interferências na transmissão.
O campo ACK (*Acknology*) contém o sinal de confirmação de todos os receptores, indicando que a mensagem foi recebida sem erro.
O fim do registro marca o final da mensagem.

Geralmente o emissor inicia a transmissão enviando um registro de dados. Entretanto, o receptor também pode solicitar dados do emissor. Para isso o receptor envia um registro remoto. O registro de dados e o registro remoto correspondente possuem o mesmo identificador (Campo de arbitragem). A diferença entre os dois registro de dados (Mensagens) é feita através do campo de controle, o bit à frente do identificador ou campo de arbitragem.

O CAN foi padronizado tanto pela ISO e SAE para o intercâmbio de informações nas aplicações automotivas. Para aplicações de baixa velocidade, até 125kbit/s, padronizou-se ISO 11 519-2 [8], e para aplicações de alta velocidade, acima de 125kbit/s padronizaram-se ISO 11 898 [15], e SAE J 22 584 [16] (Automóveis de passeio) e SAE J 1939 [17] (Caminhões e Ônibus).

A AUTOSAR (*Automotive Open Architecture System*) é outra iniciativa de padronização. Ela visa padronizar um software *Open Source*, que deverá ser desenvolvido em conjunto pela fabricante de automóveis, fornecedores e desenvolvedores de ferramentas de diagnostico [18] e [19].

No próximo item serão apresentadas as etapas para o desenvolvimento do estudo proposto neste trabalho.

C. Desenvolvimento

O desenvolvimento deste estudo foi realizado nas seguintes etapas:

Desenvolvimento do Hardware

Para interface da rede CAN do veículo com o servidor remoto, desenvolveu-se um circuito eletrônico. Este circuito é formado por dois blocos individuais de circuitos, sendo que cada um desenvolve funções especificas de cada processo, sendo eles: coleta de dados da rede CAN; processamento dos dados por meio do microcontrolador e a transmissão dos dados viam GPRS, pelo do módulo celular.

O *Transceiver* PCA82C250 [20] é responsável pela compatibilização dos níveis elétricos requeridos pela rede CAN com os níveis elétricos necessários ao trabalho do Microcontrolador e vice-versa. Este *transceiver* opera com velocidade de 1Mb/s. Na Tabela 1 segue a descrição dos pinos do PCA82C250.

Tabela 1. Descrição dos pinos do PCA82C250 [20].

Simbolo	**Pino**	**Descrição**
TXD	1	Transmissão da entrada de dados
GND	2	Terra
V_{cc}	3	Tensão de alimentação
RXD	4	Recepção de dados
V_{ref}	5	Tensão de referência
CANL	6	Nível baixo de tensão de entrada e saída da rede CAN.
CANH	7	Nível alto de tensão de entrada e saída da rede CAN.
Rs	8	Resistor de entrada.

O módulo celular GPRS SIM340E dá suporte a tecnologia GSM Quad-band / GPRS que funciona em frequências GSM 850M HZ, EGSM 900 MHz, DCS 1800 MHz e 1900 MHz PCS [21].

Com uma configuração de 40mm x 33mm x 2,85 milímetros, o módulo celular atende aos requisitos de espaço em sistemas embarcados, onde o tamanho e a espessura é um fator crítico do projeto, tais como telefone inteligente, PDA (*Personal digital assistant*) e outros dispositivos móveis [21].

A interface de conexão com a aplicação é por meio da conexão entre o módulo e o hardware (placa de circuito impresso), exceto a interface da antena de RF (*Radio Frequency*). A seguir são apresentadas algumas características do módulo celular:

1. O teclado e o display fornecem flexibilidade para desenvolver aplicações personalizadas;
2. Porta serial e uma porta de depuração que ajuda a desenvolver facilmente seus aplicativos;

3. Dois canais de áudio que incluem dois microfones como entradas e duas saídas do alto-falante.

A Figura 2 apresenta o módulo celular GPRS SIM340E.

Figura 2. Módulo celular GPRS.

Para o processamento dos dados que estão trafegando no barramento da rede CAN e controle do módulo celular é utilizado o microcontrolador (MC9S08DZ60) [22].

Apesar de não ser exclusivamente destinado para aplicações automotivas, o microcontrolador (MC9S08DZ60) [22] foi desenvolvido para atender os requisitos específicos de um veículo com barramento de dados serial com processamento em tempo real, operação de confiança no ambiente de um veículo, baixa relação custo-eficácia e largura de faixa exigida. As características básicas do MC9S08DZ60 são as seguintes [22]:

1. Implementação do protocolo CAN;
2. Comprimento de dados de 8 bytes;
3. Taxa de bits programável até 1 Mbps;
4. Cinco *buffers FIFO* (consiste de um conjunto de ler e escrever ponteiros, armazenamento e lógica de controle com regime de armazenagem);
5. Modo de *loopback* (consiste em um canal de comunicação com apenas um ponto final; qualquer mensagem transmitida por meio de tal canal é imediatamente recebida pelo mesmo canal), programável suporta a operação de auto-teste;
6. Programável funcionalidade de recuperação de barramento desligado;
7. Programável MC9S08DZ60 fonte de *clock* (é um circuito oscilador que tem a função de sincronizar e ditar a medida de velocidade de transferência de dados entre duas partes essenciais de um processamento*)*;
8. Inicialização dos registros de configuração global.

Para interface a rede CAN é utilizado um cabo OBDII, que é conectado ao barramento da rede fazendo com que os pinos CAN_H (CAN *High*) e CAN_L (CAN *Low*) do *transceiver* PCA82C250 sejam conectados ao barramento.

Nos pinos TXCAN_21 (saída do transmissor CAN) e RXCAN_22 (entrada do receptor CAN) do microcontrolador MC9S08DZ60 são conectados os pinos TXD_1 (saída do transmissor) e RXD_4 (entrada do receptor) do *transceiver* PCA82C250 [20] no qual são recebidos e enviados os dados da rede CAN.

Os pinos TXD1_11 (saída do transmissor) e RXD1_12 (entrada do receptor) do microcontrolador MC9S08DZ60 [22], são conectados ao módulo GPRS, que tem a função de enviar e receber comandos AT (conjunto de comandos de linguagem desenvolvida para modem, para produzir comandos completos para operações, como discar, desligar e mudar os parâmetros da conexão). Para inicialização do módulo celular foi utilizado o pino PTF5_15 (inicializador da transmissão).

Para sincronizar os periféricos internos do microcontrolador e ter precisão nas frequências do barramento na rede CAN é utilizado um cristal oscilador de 8MHZ conectados aos pinos EXTAL_6 (entrada do sinal do oscilador externo 1) e XTAL_7 (entrada do sinal do oscilador externo 2) do microcontrolador MC9S08DZ60.

Para gravar o microcontrolador, diagnósticos e análise em tempo real do funcionamento do software, o fabricante do microcontrolador disponibiliza um pino de comunicação serial chamado BKGD (*Back Ground Debuger*), disposto no pino 33 do microcontrolador e um pino de RESET (reinicialização), conectado ao pino 8.

Para alimentar os circuitos integrados, como o microcontrolador e o transceiver PCA82C250 foi necessário um regulador de tensão 78L05 [23], que converte a tensão da bateria 12V nominal para 5V regulado.

Para alimentação do módulo celular foi necessário um regulador de tensão MC29302BT [24], que converte a tensão da bateria 12V nominal para 4,1V regulado.

A Figura 3 ilustra o hardware completamente montado, os blocos A, B e C separam as principais partes do circuito:
O bloco A ilustra o módulo celular GPRS, o conector do SIM Card, e a fonte de alimentação. O módulo GPRS recebe os dados processados por meio do microcontrolador e os envia para um servidor via internet.

O bloco B ilustra o microcontrolador e os pinos de gravação. Esse bloco de circuito recebe os dados do transceiver, processa e envia ao módulo GPRS.

No bloco C está à conexão e o transceiver da rede CAN, que permite o acesso à rede do veículo.

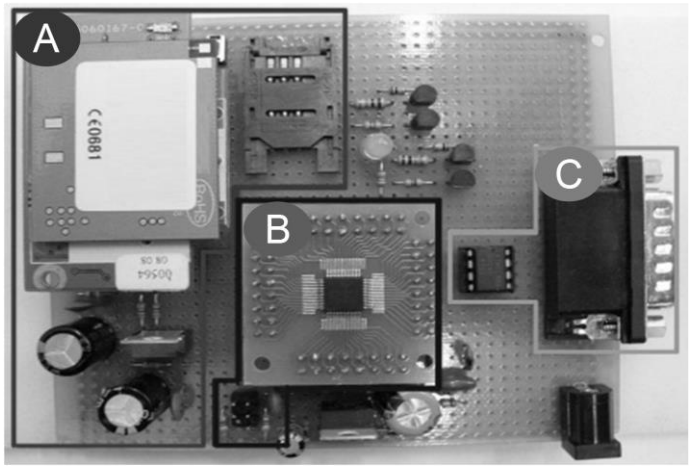

Figura 3. Hardware montado.

Desenvolvimento do Software

Esta etapa descreve todo o funcionamento do software que gerencia todo o processo lógico do projeto. A linguagem de programação utilizada foi à linguagem C para

microcontrolador, que foi desenvolvido e compilado por meio do software *CodeWarrior* [25] do próprio fabricante do microcontrolador MC9S08DZ60.

A lóogica de funcionamento do software é apresentada no diagrama de funcionamento Figura 4:

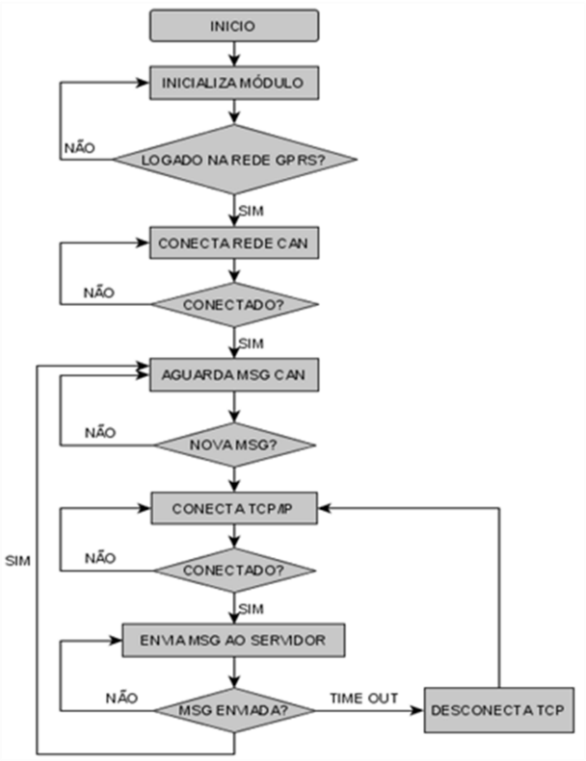

Figura 4. Diagrama de fluxo do funcionamento do software.

Após o circuito ser alimentado ou conectado ao conector OBDII, o software aciona uma sequencia que inicializa todos os periféricos internos do microcontrolador: Pinos de *input/output* (entra/saída de dados); interface serial (onde são definida velocidade de *baud rate* (taxa de transmissão de dados); interface CAN (denominado pelo fabricante do microcontrolador MC9S08DZ60 (msCAN - Mensagem CAN).

Ao término da inicialização do microcontrolador MC9S08DZ60, o software passa para um próximo estado no qual começará a inicialização do Módulo GPRS.

Nesta etapa, o software envia comandos de Modem via interface serial para definir alguns parâmetros de funcionamento do módulo. Ainda nesta etapa, o software entra em um estado de "aguarde", para esperar que o módulo se registre na rede da operadora escolhida.

Após a conexão com a rede celular, o software se conecta com a rede CAN do veículo. Neste estado, o software define alguns parâmetros para a conexão e entra em um estado de "aguarde" para sincronizar e se conectar com a rede.

Assim que a conexão com a rede CAN do veículo é estabelecida o software passa para um novo estado, entrando em um estado de "aguarde", aguardando uma nova mensagem da rede.

Caso haja uma nova mensagem, ela é armazenada em um *buffer* (retentor) na memória RAM (*Random Access Memory*) do microcontrolador e um *flag* (registro de memória) indicará que há um novo dado no buffer e já pode ser enviado.

Ao identificar que o *flag* de um novo dado no *buffer*, o software passa para um novo estado, iniciando o processo de conexão com o servidor. Neste estado, o software se conecta a um endereço IP (*Internet Protocol*) e uma porta de comunicação específica do servidor.

Após inserir os comandos de conexão com o servidor, o módulo entra em um estado de "aguarde", para esperar a resposta de sucesso de conexão com o servidor. Caso ultrapasse o tempo pré-definido de *timeout* (tempo de resposta do servidor), o software entende que não houve sucesso na conexão e retorna o processo de conexão com o servidor.

Ao obter sucesso na conexão com o servidor, o software entra em um novo estado, enviando todo o conteúdo do *buffer* ao servidor. Por segurança e garantia de envio dos dados ao servidor, escolheu-se o protocolo de comunicação TCP/IP (*Transmission Control Protocol* / *Internet Protocol*) que por padrão, possui características de garantia de entrega dos pacotes de dados transmitidos.

Por esse motivo, o software deve prover uma rotina de "aguarde" de confirmação de mensagem entregue, na qual o software fica aguardando a confirmação do servidor.

Se por algum motivo a mensagem não for entregue, o software possui uma saída alternativa por *timeout*, que faz o software se desviar para um estado alternativo, desconectado o módulo do servidor, voltando ao estado de reconexão com a rede, inicializando todo o processo de envio dos dados.

Caso a mensagem seja enviada com sucesso e/ou o servidor responder a confirmação do envio, o software se desvia para o estado "aguardando MSG CAN (mensagem CAN)". Caso neste estado haja um novo dado no *buffer* de memória, o software recomeça todo o processo de conexão e envio dos dados ao servidor.

Para leitura e escrita das mensagens na rede, duas rotinas foram desenvolvidas: Estas rotinas foram denominadas msCAN_TransmitMsg (transmitir mensagem CAN) e msCAN_Rx_ISR (receber mensagem CAN), conforme apresenta a Figura 5.

De acordo com a rotina da Figura 5, ao receber uma interrupção (sinal para atender o dispositivo que pediu a interrupção), a CPU (*central processing uni*) direciona o ponteiro (se refere a um valor alocado na memória) para esta rotina. Ao entrar nesta rotina, o software verifica se o dado que está no *buffer* é *extended ID* (Mensagem com identificador de 29 bit) ou *standard ID* (Mensagem com identificador de 11 bit). Ao identificar o tamanho do ID (identificador), o software armazena no *buffer* os conteúdos do dado e aciona um *flag*, indicando que há um dado novo no buffer, conforme Figura 5.

```
/*****************************************************************************
Function Name   :   msCAN_Rx_ISR
Parameters      :   None
Returns         :   None
Notes           :   msCAN Rx interrupt service routine.
*****************************************************************************/
interrupt void msCAN_Rx_ISR(void)
{

if(CANRIDR1 & IDE)
{
/* extended ID */
CAN_RxBuffer.ID = EXTENDED_ID;
CAN_RxBuffer.ID_FIELD.BYTE.ID0 = (CANRIDR0 >> 3);
CAN_RxBuffer.ID_FIELD.BYTE.ID1 = ((((CANRIDR1 & 0xE0) >> 3)|(CANRIDR1 & 0x06) >> 1)| (CANRIDR0 << 5));
CAN_RxBuffer.ID_FIELD.BYTE.ID2 = ((CANRIDR2 >> 1)| (CANRIDR1 << 7));
CAN_RxBuffer.ID_FIELD.BYTE.ID3 = ((CANRIDR3 >> 1)| (CANRIDR2 << 7));
}
else
{
/* standard ID */
CAN_RxBuffer.ID = STANDARD_ID;
CAN_RxBuffer.ID_FIELD.StandardID = ((unsigned int)CANRIDR0<<3)|((CANRIDR1>>5)& 0x07);
}

CAN_RxBuffer.LENGTH = (CANRDLR & 0x0F);
CopyData(&CAN_RxBuffer.BYTE0,&CANRDSR0,CAN_RxBuffer.LENGTH);

/* set RX software flag*/
msCAN_StateFlag |= msCAN_NewMsg;
/* clear Interrupt flag */
/* Note: clearing the Rx interrupt flag must be done after reading the message data */
CANRFLG = RXF;

}
```

Figura 5. Rotina de recepção da rede CAN.

Para conexão com a rede GPRS e transmissão dos dados, uma rotina é acionada pelo processo principal, e executa algumas rotinas.

O primeiro estado a ser executado é "SET_GPRS_ CONNECTION_MODE" (iniciar modo de conexão GPRS). Este estado seleciona o módulo que vai trabalhar e em qual APN (*Access Point Name*) de operadora celular o módulo vai utilizar para a conexão à rede GPRS.

Ao se conectar com a rede GPRS, o software vai para um novo estado "START_UP_TCP" (iniciar comunicação TCP/IP), no qual se conectará ao servidor. Neste estado, o software se conecta em um IP e uma porta de comunicação do servidor e escolhe com qual protocolo o módulo vai se comunicar.

Ao obter sucesso na conexão com o servidor, o software vai para um novo estado "SEND_DATA_THROUGH_TCP" (enviar dado via TCP/IP), no qual ele prepara o módulo para enviar a mensagem. Neste estado o software envia o comando "AT+CIPSEND"(mensagem de teste de conexão) e fica aguardando como resposta o caractere ">"(conexão estabelecida).

Após o recebimento desse caractere, o software vai para o próximo estado "WRITE_MSG_SERVER"(ler mensagem do servidor), para enfim enviar todo pacote de dados recebidos da rede CAN.

Após o envio da mensagem o software aguarda a confirmação de envio assim como define o protocolo de comunicação.

Após obter confirmação de envio de toda a mensagem, o software vai para o próximo estado "CLOSE_CONNECTION" (desconectar servidor), desconectando-se do servidor. Ao se desconectar do servidor, o software vai para o "DEACTIVATE_GPRS_PDP_ CONTEXT" (desconectar comunicação GPRS) para se desconectar da rede GPRS, conforme a Figura 6.

```
unsigned char SendMsgTcp (unsigned char *Msg){
  static unsigned char Estado=SET_GPRS_CONNECTION_MODE
  switch (Estado)
    case SET_GPRS_CONNECTION_MODE:

if(SCI1_SendATCommand("AT+CIPCSGP=1,\"TIM.BR\",\"TIM\",\"TIM\"\n\r",
     OK,T_OUT5,TRIES2)==TRUE)
      Estado=START_UP_TCP;
      return FALSE;
    break;
    case START_UP_TCP:

if(SCI1_SendATCommand("AT+CIPSTART=\"TCP\",\"189.92.10.71\",\"9004\"\n\
r","CONNECTOK",
    T_OUT20,TRIES2)==TRUE)      Estado=SEND_DATA_THROUGH_TCP;
     return FALSE;
    break;
    case SEND_DATA_THROUGH_TCP:

if(SCI1_SendATCommand("AT+CIPSEND\n\r",">",T_OUT5,TRIES2)==TRUE)
      Estado=WRITE_MSG_SERVER;
      return FALSE;
    break;
    case WRITE_MSG_SERVER:
      if(SCI1_SendATCommand(Msg,"+IPD",T_OUT45,TRIES5)==TRUE)
      Estado=CLOSE_CONNECTION;
      return FALSE;
    break;
    case CLOSE_CONNECTION:

if(SCI1_SendATCommand("AT+CIPCLOSE\n\r",OK,T_OUT5,TRIES2)==TRUE)
      Estado=DEACTIVATE_GPRS_PDP_CONTEXT;
      return FALSE;
    break;
    case DEACTIVATE_GPRS_PDP_CONTEXT:
if(SCI1_SendATCommand("AT+CIPSHUT\n\r",OK,T_OUT5,TRIES2)==TRUE){
       Estado=SET_GPRS_CONNECTION_MODE;
       return TRUE;
      }
      return FALSE;
    break;
   }
}
```

Figura 6. Envia mensagem via TCP/IP.

A Figura 7 apresenta a janela do software chamado *EchoServer.jar*, desenvolvido em Java para auxiliar nos testes de confirmação e recebimento das mensagens trafegadas entre a interface de monitoramento e o servidor.

Ao executar esse software em DOS (*Disk Operating System*) no Windows, ele monitora e apresenta todas as mensagens que chegam ao IP da máquina que o software está sendo executado e recebendo todo pacote na porta de comunicação 9004.

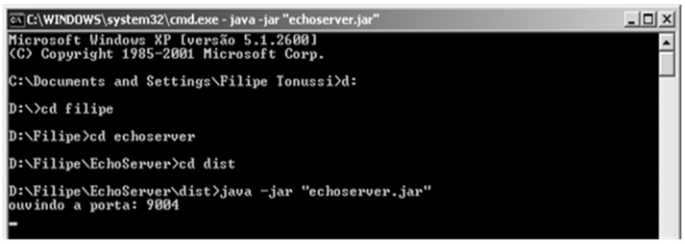

Figura 7. Ambiente de monitoramento.

III. ANÁLISE DE RESULTADOS EXPERIMENTAIS

Os testes de validação do software e hardware deste estudo foram realizados nas seguintes etapas:

A. Conexão com interface OBD

A Figura 8 apresenta a interface montada e sendo testada em laboratório.

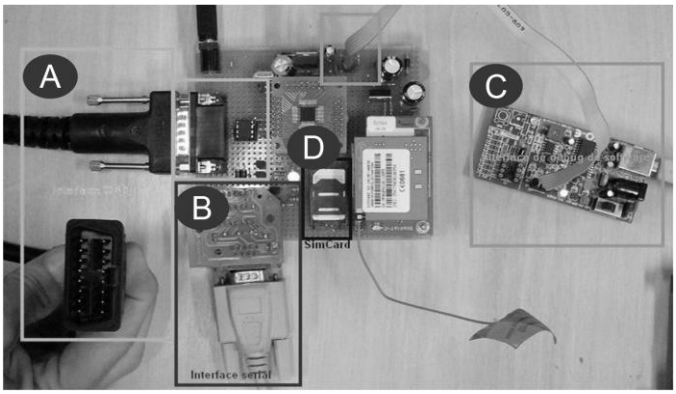

Figura 8. Interface equipada para teste e debug.

O campo A ilustra o cabo de conexão com o conector de diagnóstico do veículo (conector OBDII) para coletar os dados da rede CAN.

O campo B ilustra o cabo serial e um conversor RS232 (padrão para troca serial de dados binários entre um DTE - *Data Terminal equipment*) e um DCE – *Data Communication equipment*), para monitorar todos os comandos trafegados entre o módulo celular e o microcontrolador.

No campo D foi conectado o SimCard (*Subscriber Identity Module*) da operadora TIM para possibilitar o acesso à rede GPRS e transmitir os dados da rede CAN ao servidor.

O campo C ilustra a placa de gravação e *debug* do software.

A Figura 9 mostra a conexão com o conector de diagnóstico do veículo (conector OBDII), onde foram realizados os testes de conceito.

Para isso foi utilizado o carro da marca Honda, modelo Fit. Neste modelo e marca do veículo o conector de diagnóstico está situado abaixo do volante, próximo ao pedal do acelerador. A posição do conector de diagnóstico varia dependendo do modelo e da marca do veículo e pode ser identificado no manual de usuário.

Figura 9. Conexão com o conector de diagnóstico (OBDII).

Para validar se os dados estão trafegando na rede CAN do veículo, foi utilizado a um *sniffer* [26] (ferramenta, constituída de um software e hardware, capaz de interceptar e registrar o tráfego de dados em uma rede de CAN).

Na Figura 10 é possível observar os dados que estão sendo trafegado na rede do veículo, filtrados pelo software da interface desenvolvido e transferido para o servidor.

```
4731    > 68 6a f1 01 0c d0
4732    < 48 6b 10 41 0c 0b a3 be
4733    > 68 6a f1 01 0c d0
4734    < 48 6b 10 41 0c 0b 95 b0
4735    > 68 6a f1 01 0c d0
4736    < 48 6b 10 41 0c 0b 9d b8
4737    > 68 6a f1 01 0c d0
4738    < 48 6b 10 41 0c 0b 95 b0
4739    > 68 6a f1 01 0c d0
```

Figura 10. Dados trafegando na rede CAN.

B. Transmissão dos dados via GPRS ao servidor

Quando o software detecta um dado novo no *buffer*, o software dispara uma conexão com a rede GPRS, conforme ilustra a Figura 11.

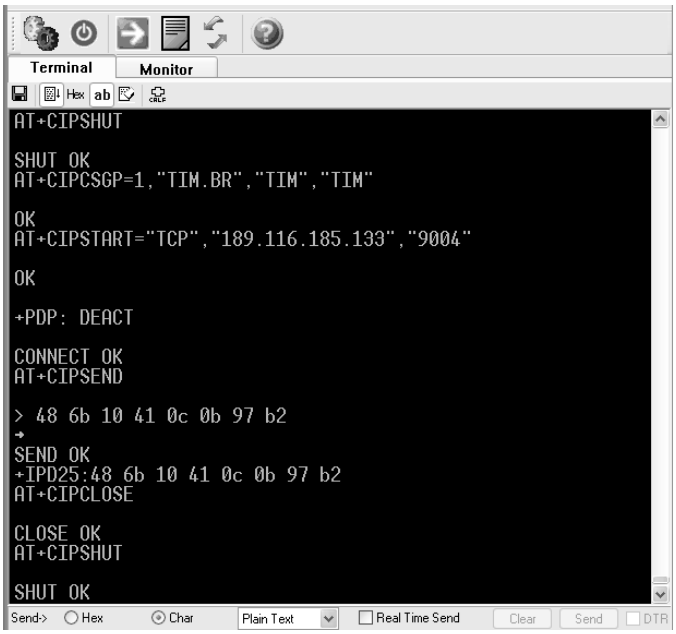

Figura 11. Terminal para análise dos comandos AT.

Neste terminal serial conectado via RS232 entre as conexões TX (transmitir dado) e RX (receber dado) do módulo e o microcontrolador para análise dos comandos AT que trafegam entre os dois. Analisando os comandos é possível observar o processo de conexão com a rede GPRS.

Para realizar essa conexão, o comando a ser enviado ao módulo é AT+CIPCSGP=1,"TIM.BR","TIM","TIM" onde o atributo 1 significa que se escolhe o tipo de conexão GPRS e os seguintes atributos informam a conexão com a operadora TIM, utilizando o APN TIM.BR, senha TIM e usuário TIM.

Fazendo isso, o módulo se conectará com o serviço de dados da operadora celular escolhida.

Conectando com o serviço de dados da operadora, o software vai para a próxima etapa, que envia o comando AT+CIPSTART="TCP","189.116.185.133","9004".

Neste comando, o módulo estabelece conexão com o servidor via "TCP", conectando com o IP do servidor "189.116.185.133" e PORTA "9004".

Ao obter sucesso na conexão com o servidor, o módulo envia uma resposta com "CONNECT OK"(conexão estabelecida).

Ao obter sucesso na conexão com o servidor, o software descarrega o buffer de dados recebido da rede CAN.

Para isso o software envia o comando AT+CIPSEND e aguarda o comando ">", que indica que já pode ser descarregado o *buffer*. Ao receber o comando ">" o software envia todo o *buffer*, que neste caso os dados são "48 6b 10 41 0c 0b 97 b2".

Ao esvaziar o *buffer*, o módulo envia uma resposta como "SEND OK"(mensagem enviada) que informa ao software que os dados saíram do módulo.

Como o tipo de conexão usado é TCP, o software fica aguardando a resposta de confirmação de pacote entregue ao servidor, caso obtenha sucesso, a resposta é "+IPD25:48 6b 10 41 0c 0b 97 b2".

Depois de finalizado todo o processo de envio dos dados ao servidor, o software envia comandos para o módulo se desconectar do servidor e da rede.

C. Análise dos dados do servidor

Para verificar os pacotes recebidos no servidor, deve-se descobrir qual o IP usado pelo servidor (em caso de IP dinâmico).

Para isso, inseriu-se o comando de DOS "ipconfig" (configuração de IP), que indica o IP usado pelo módulo para envio dos dados da rede CAN.

Após conhecer este IP e inseri-lo no software, pode-se finalmente executar o EchoServer.

A partir deste momento, o servidor já pode receber conexão e exibir o conteúdo do pacote recebido no *prompt* (interpretador de linha de comando).

Ao obter sucesso na execução do EchoServer, ele filtrará todas as informações que chegarem à porta de comunicação utilizada.

Como Figura 12, a primeira conexão foi estabelecida pelo "cliente" IP 187.82.24.25 (*Accepted Client* (cliente aceito) : ID – 0 : Address (endereço) – 187.82.24.25) e recebido o pacote "48 6b 10 41 0c 0b 97 b2" (*Client Says* (cliente envia): 48 6b 10 41 0c 0b 97 b2).

Após receber os dados, o servidor envia um pacote de confirmação ao "cliente", caso a resposta seja recebida com sucesso, o cliente encerra a conexão com o servidor (Conexão encerrada por: 0 : *Address* – 187.82.24.25) e o processo continuará caso haja novos dados sendo recebidos na porta de comunicação utilizada como ilustra os 10 pacotes seguintes, conforme Figura 12.

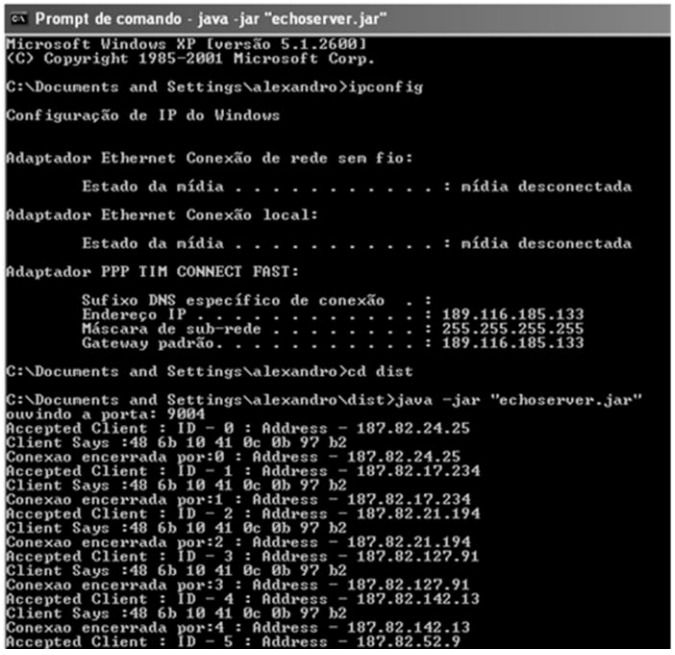

Figura 12. Análise dos dados recebidos no servidor

IV. CONCLUSÃO

A principal contribuição deste estudo está direcionada para o mercado de reparação automotiva. O sistema aqui proposto é de grande importância para auxiliar no diagnóstico e reparo de falhas referentes à ECU e aos seus periféricos como por exemplo, freio ABS, sistema de gerenciamento do motor, sensores e sistema de conforto. Quando todos estes periféricos são monitorados à distância é possível realizar a manutenção do

veículo em tempo real, que resulta em comodidade e segurança ao proprietário do veículo.

Com o sistema OBDII, também é possível acessar os dados de emissões de gases poluentes do veiculo. O acesso a estes dados é muito importante para a completa execução do Programa de Inspeção e Manutenção de Veículos em uso. Este programa de inspeção é requisito para avaliar o estado de manutenção dos veículos automotores.

Com o *hardware* proposto neste estudo é possível agregar um módulo GPS, que possibilita além da coleta *online* dos dados de funcionamento do veículo, seu rastreamento e localização. Este rastreamento permite que a concessionária mais próxima da localização do veiculo seja informada da ocorrência de uma falha. Esta informação é importante para facilitar ao dono do veículo entrar em contato com a concessionária para agendar uma revisão, ou então, enviar socorro caso o veículo não possa se locomover. Além disso, o sistema aqui proposto apresenta muitas possibilidades de aproveitamento das informações transmitidas.

REFERÊNCIAS

1. QUEIRÓS, J. M. R. **Sistema de Sensorização e Telemetria de um VEC (Veículo Eléctrico de Competição)**. 2011. 117 f. Dissertação (Mestrado Integrado em Engenharia Electrotécnica e de Computadores Major Automação) - Faculdade de Engenharia da Universidade do Porto. Cidade do Porto. 2011.
2. GUTIERREZ, E. M. **Telemetria: Aplicação de rede de sensores biomédicos sem fio**. 2006. 126 f. Dissertação (Mestrado em engenharia elétrica) - Universidade de brasília. Brasília. 2006.
3. SCHWARZ, L. **Proposta de um sistema telemétrico para aquisição de sinais fisiológicos**. 2007. 117 f. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Santa Catarina. Santa Catarina. 2007.
4. WALKER, B. et al. Augmenting amusement rides with telemetry. **Advancements in Computer Entertainment Technology**, Salzburg, 2007. 115–122.
5. KHAN, N. . S. Z. . &. H. C. C. An experiment on internet based telemetry. **Proceedings of the 6th WSEAS Int. Conf. on Electronics, Hardware,Wireless and Optical Communications**, Corfu Island, 2007. 68–78.
6. SICHONANY, O. R. D. A. O. et al. Telemetria na transmissão de dados de desempenho de máquinas agrícolas utilizando tecnologias GSM/GPRS e ZigBee. **Ciência Rural**, Santa Maria, 42, n. 8, Agosto 2012. 1430-1433.
7. ROZAS, N. O que é Telemetria. **Revista Gás Brasil**, p. 13-15, 2004.
8. ISO 15 765-2. **Road Vehicles — Diagnostics on CAN — Part 2:** Network Layer Services. EUA: International Organization for Standardization, 1999.
9. SOLTAN, N. I.; LATIF-SHABGAHI, G. Designing an Online Monitoring System for Vehicle Electric Network and Its Fault Prediction aided by Fuzzy Logic. **Proceedings of the 13th International CAN Conference iCC 2012**, Hambach Castle, Germany, March 2012. 01-11.
10. BOSCH, R. **Manual de Tecnologia Automotiva**. 25. ed. São Paulo: Edgard Blücher, 2005.
11. EISELE, H. The Benefits of CAN for In-Vehicle Networking. **Proceedings of the 13th International CAN Conference iCC 2012**, Hambach Castle, Germany, March 2012. 09-12.
12. POLFLIET, J.; MAELE, W. V. D.; BUZAS, R. CAN for energy efficiency in cars. **Proceedings of the 13th International CAN Conference iCC 2012**, Hambach Castle, Germany, March 2012. 02-18.
13. SHAH, M. B. N.; HUSAIN, A. R.; DAHALAN, A. S. A. An Analysis of CAN Performance in Active Suspension Control System for Vehicle. **Proceedings of the 13th International CAN Conference iCC 2012**, Hambach Castle, Germany, March 2012. 12-19.
14. NAVET, N.; PERRAULT, H. CAN in Automotive Applications: A Look Forward. **Proceedings of the 13th International CAN Conference iCC 2012**, Hambach Castle, Germany, March 2012. 01-06.
15. ISO 11 898-2. **Road Vehicles — Interchange of Digital Information —Controller Area Network for High-Speed Communication**. EUA: International Organization for Standardization, 2003.
16. SAE J 22 584. **CAN - Standard Format - Passenger Cars**. EUA: Society of Automotive Engineers, 1999.
17. SAE J 1939. **CAN - Extended Format - Trucks and Bus**. EUA: Society of Automotive Engineers, 1999.
18. SCHUMANN, T. Standardization in automotive industry. **Proceedings of the 13th International CAN Conference iCC 2012**, Hambach Castle, Germany, March 2012. 07-10.
19. PFEIFFER, O. Features of CiA 447 Application profile for special-purpose car add-on devices. **Proceedings of the 13th International CAN Conference iCC 2012**, Hambach Castle, Germany, 11-15 March 2012.
20. SEMICONDUCTORS NXP. **Transceiver PCA82C250**, 2010. Disponivel em: <http://www.nxp.com/documents/data_sheet/PCA82C250.pdf>. Acesso em: 22 mar. 2013.
21. SIM TECHNOLOGY. **Módulo Celular GPRS SIM340E**, 2010. Disponivel em: <http://www.simcom.us/act_admin/supportfile/SIM340_HD_V2.01.pdf>. Acesso em: 22 mar. 2013.
22. FREESCALE SEMICONDUCTOR INC. **Microcontrolador MC9S08DZ60**, 2010. Disponivel em: <http://www.freescale.com/files/ microcontrollers/doc/app_note/AN3331.pdf.>. Acesso em: 22 mar. 2013.
23. ST MICROELECTRONICS. **Regulador de tensão 78L05**, 2003. Disponivel em: <http://html.alldatasheet.com/html-pdf/22687/STMICROELECTRONICS/78L05/6491/4/78L05.html>. Acesso em: 2013 mar. 2013.
24. MOTOROLA CO. **Regulador de tensão MC29302BT**, 2010. Disponivel em: <http://www.alldatasheet.com/view.jsp?Searchword=MC29302BT>. Acesso em: 22 mar. 2013.

25. FREESCALE SEMICONDUCTOR INC. **CodeWarrior Development Tools**, 2013. Disponivel em: <http://www.freescale.com/webapp/sps/site/homepage.jsp?code=CW_HOME>. Acesso em: 22 mar. 2013.
26. BOSCH, R. **ESI[tronic]**, 2013. Disponivel em: <http://www.bosch.com.br/br/equiteste/produtos/infotec/esi_conteudo_c.htm>. Acesso em: 22 mar. 2013.