COMPUTADOR DE BORDO DE KART PARA AQUISIÇÃO DE PARAMETROS DE MOTOR E ANALISE OFF-LINE.

ALUNO: Fábio de Castro Oliveira – RA: 2001577/0

Orientador: Thiago de Miranda Leão Toribio, M. Sc.

Brasília – DF, dezembro de 09

COMPUTADOR DE BORDO DE KART PARA AQUISIÇÃO DE PARAMETROS DE MOTOR E ANALISE OFF-LINE

por

Fábio de Castro Oliveira

Monografia apresentada à Banca examinadora do curso de engenharia de computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília como requisito parcial para obtenção do titulo de Engenheiro da Computação.

Brasília – DF, dezembro de 2009

Banca Examinadora

AGRADECIMENTOS

Aos meus pais e irmã, pelo amor, dedicação, paciência e esforço que sempre tiveram para me orientar e educar em todos os momentos.

Aos meus amigos, colegas e chefes que me deram apoio e incentivo para a realização deste projeto. Em especial, Ticiano Bragatto pelos ensinamentos e ajudas com a eletrônica, Thiago Abreu com as duvidas do .NET, Gabriel e Augusto.

A todos os professores e mestres do curso de Engenharia da Computação que compartilharam seus conhecimentos, durante toda a formação acadêmica. Ao meu orientador Thiago Toribio, que confiou no meu trabalho tantas vezes.

Por fim, a todos que me apoiaram, incentivaram, colaboraram e contribuíram de alguma forma para a realização deste trabalho.

RESUMO

Este trabalho apresenta o desenvolvimento de uma solução de computação embarcada em um kart, que permite ao piloto e equipe fazerem uma análise de performance *off-line* do equipamento logo após um treino ou corrida. A solução desenvolvida foi implementada utilizando os seguintes componentes eletrônicos: um acelerômetro, modelo DE-ACCM5G, que expressa, em múltiplos da aceleração da gravidade “g”, as acelerações exercidas na longitudinal e lateral, um *Reed Switch*, que abre e fecha a cronometragem de volta, um transdutor LM35, que mede a temperatura do motor, além de um microcontrolador atmel ATmega 328 para gerenciar todos os sensores. Os dados obtidos são transmitidos para o computador pela porta USB.

Com a transmissão de dados obtidos na pista para o computador, a equipe e piloto podem visualizar através de gráficos os principais dados do motor e pista: a temperatura em que o motor está trabalhando, o tempo de volta e as acelerações linear e lateral. Foi utilizado o Visual Studio 2008 da Microsoft para desenvolver a aplicação que importa, trata, armazena os dados e gera os gráficos. O banco de dados utilizado é o SQLite, a linguagem utilizada para a programação do microcontrolador foi C e o compilador utilizado é desenvolvido pela fabricante da placa de teste Arduíno.

Um carro de brinquedo com o protótipo embarcado foi utilizado para fazer as aferições dos parâmetros para a banca. Os dados foram obtidos de forma manual adotando a gravidade como aceleração nos eixos e o aquecimento do sensor de temperatura com um secador de cabelo simulando o aquecimento do motor.

Palavras-chave:

Acelerômetro; *Reed Switch* ; microcontrolador; Arduíno; SQLite; transmissão de dados; Microsoft Visual Studio 2008 .net; Banco de dados.

ABSTRACT

This study presents the development of an onboard computer solution for a go-kart, which allows pilot and team to evaluate the equipment's performance off-line, after a practice or racing session. This solution was implemented using the following electronic components: a DE-ACCM5G accelerometer, which acquires longitudinal and lateral acceleration as multiples of the gravitational acceleration, a Reed Switch that starts and stops a lap timing chronometer, a LM35 transducer to measure the engine's temperature and finally a Atmel ATmega microcontroller to control all the sensors. The data is stored and transfered to a computer. The firmware used in the microcontroller is programmed in the C language, using the Arduino platform.

After transmitting the acquired data to the computer, team and pilot are able to visualize engine temperature, lap time and linear acceleration through graphics. The application that imports and manages the data and generates the graphics was developed using Microsoft Visual Studio 2008 C#.NET. The SQLite database management system was used to store the program data.

A toy vehicle was used to generate acceleration data to test the prototype. A blowdryer was used to simulate a engine's heating, and a magnet was used to verify the lap timing.

. Key-words:

Accelerometer; *Reed Switch* ; microcontroller; Arduíno; SQLite; Microsoft Visual Studio; Database.

SUMÁRIO

LISTA DE FIGURAS

1 - INTRODUÇÃO

1.1. MOTIVAÇÃO

A principal motivação para esse projeto consiste na elaboração de um protótipo acadêmico, que pode servir de base para um produto comercial de baixo custo para analise off-line dos seguintes parâmetros: temperatura do motor, aceleração do kart e tempo de volta, obtidos em testes e competições de kart. Apesar de relativamente simples, por utilizar apenas um pistão, ou seja, um braço mecânico como os utilizados nas máquinas de antigamente, o motor de um kart envolve muita tecnologia. Basta citar que um motor de kart de competição, 2 tempos, chega a velocidades acima dos 100km/h em menos de 5 segundos, com mais de 15.000 rotações por minuto. Nas categorias maiores do automobilismo, formula 3, formula 1, stock car e outros, é muito comum o uso da telemetria, que auxilia na elaboração de ajustes que podem ser feitos ou até mesmo na prévia manutenção de algum desgaste imprevisto em peças.

O kart é considerado no Brasil a categoria escola do automobilismo. Grandes pilotos de renome internacional começaram suas carreiras nesta categoria. Pilotos como Felipe Massa e Rubens Barrichello, sempre que estão de férias, continuam “brincando” nesses pequenos bólidos, para manterem o reflexo em dia. Como em toda competição automobilística, no kart, a briga por décimos de segundos também pode valer posições. O ajuste de um *chassis* bem feito para evitar um desgaste prematuro de pneus, um motor que trabalhe com regime alto em desempenho sem perigo de quebra, um carburador que atenda o motor e a forma de pilotagem do piloto, todas essas variáveis são importantes em uma competição, e podem ser estudadas através de

parâmetros obtidos nas pistas, dentro do *box*. O fato de aliar a tecnologia computacional à tecnologia mecânica torna esse ajuste muito mais simples se bem estudado.

Os aparelhos com este objetivo que se encontram no mercado são importados e de custo elevado. Como exemplo, um sensor de temperatura custa em torno de 400 reais. Muitas vezes são de difícil compreensão e sem assistência técnica no Brasil. Em 1998, uma empresa de Belo Horizonte entrou no mercado, com um produto que, apesar de bem construído, apenas mostra na pista os tempos obtidos, número de voltas e a temperatura do motor, não fazendo a transferência de dados obtidos para o computador [INJETEC, 2009]

O protótipo construído capta dados, como aceleração linear, temperatura similar a do motor e tempo de volta, e exibe gráficos que podem ser utilizados para obter melhores ajustes.

1.2. OBJETIVOS

O projeto tem como finalidade adquirir dados e processá-los, que podem ser utilizados para proporcionar maior durabilidade de motor e otimização nas preparações para as competições. Através de sensores mais baratos, armazenar e estudar os melhores resultados obtidos que podem ser utilizados para equalizar no topo os melhores rendimentos.

São objetivos específicos do trabalho:

- Obter, mostrar e armazenar a temperatura similar a gerada por um motor;

- Obter, mostrar e armazenar a aceleração linear e lateral em termos da aceleração da gravidade “g”;
- Obter, mostrar e armazenar o tempo de volta e trechos;
- Construir um *software* simples que faça a interface dos dados obtidos com o sistema operacional;
- Transferir através da leitura de arquivo texto todos os dados armazenados para o banco de dados;
- Armazenar os dados e gerar os gráficos no *software*.

1.3. ESTRUTURA DA MONOGRAFIA

Este trabalho de graduação contém cinco capítulos, a introdução (Capítulo 1) e outros quatro assim dispostos:

- Capítulo 2 – Referencial Tecnológico – abrange os principais conceitos e definições envolvidas nesse projeto, como o motor 2 tempos, telemetria, sensores, cartão de memória SD, acelerômetro, microcontrolador, C#.NET, Linguagem “C”, Banco de dados, SQLite;
- Capítulo 3 – Desenvolvimento – detalha o projeto, construção do protótipo, ferramentas utilizadas, código desenvolvido na aplicação;
- Capítulo 4 – Testes e resultados obtidos;
- Capítulo 5 – Conclusões – descreve as principais conclusões obtidas no projeto.

2 - REFERENCIAL TECNOLÓGICO

Neste capítulo é abordado o referencial tecnológico das partes que compõe o protótipo e do motor de um kart.

2.1. MOTOR

O motor converte o calor produzido pela combustão do carburante em energia mecânica. O carburante, normalmente uma mistura de gasolina e ar, é queimado no interior do cilindro.

A estrutura de um motor deve ser extremamente rígida para poder suportar as elevadas pressões a que são sujeitos as peças. É constituído basicamente por duas partes ligadas por meio de parafusos: superior, o cabeçote e a inferior, o bloco do motor, normalmente feitos em ferro fundido. [MARTINS, 2006]

2.1.1. MOTOR QUATRO TEMPOS

São bastante comuns. Encontrado em quase todos os carros, o motor de quatro tempos de combustão tem como maior vantagem a durabilidade. Trabalham com baixa freqüência e por isso são econômicos quando comparado ao motor com ciclo de dois tempos de combustão.

Pode-se dividir os quatro tempos do ciclo de um motor da seguinte maneira: [MARTINS, 2006]

- Tempo de Admissão;
- Tempo de Compressão;

- Tempo de Explosão;
- Tempo de Escapamento.

No Tempo de Admissão, a válvula de admissão abre-se, o pistão aspira a mistura de ar e combustível para o interior do cilindro. Quando o pistão chega ao ponto mais baixo, chamado de ponto morto baixo, a válvula fecha-se e a mistura gasosa fica presa dentro do cilindro, conforme ilustrado na figura 2.1

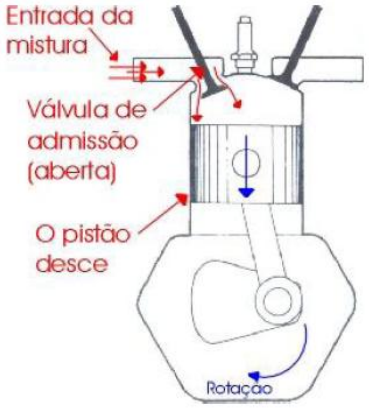

Figura 2.1 – Tempo de Admissão – Ciclo quatro tempos

[RCMASTER, 2009]

No tempo de compressão, o pistão sobe, porém as válvulas se encontram fechadas e comprime-se a mistura na câmara de explosão, conforme ilustrado na figura 2.2. [MARTINS, 2006]

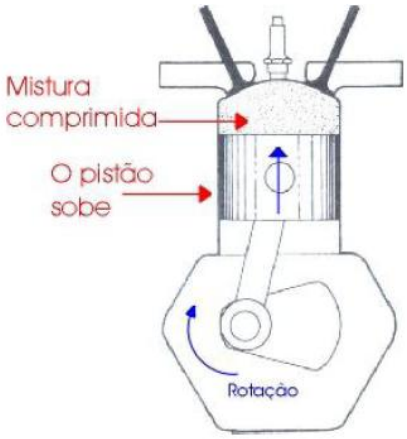

Figura 2.2 – Tempo de Compressão – Ciclo quatro tempos

[RCMASTER, 2009]

A próxima ação do ciclo, tempo de explosão, a vela de ignição produz uma faísca, que explode a mistura presa dentro da câmara causando a combustão e empurra o pistão para baixo, conforme ilustrado na figura 2.3.

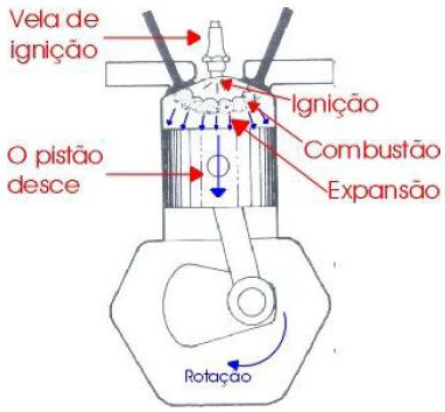

Figura 2.3 – Tempo de explosão – Ciclo quatro tempos.

[RCMASTER, 2009]

O quarto tempo chama-se tempo de escapamento. Ocorre a subida do pistão com a válvula de escape aberta. Os gases queimados são expelidos. Quando o pistão chega ao topo, conhecido como ponto morto alto, a válvula de escape fecha-se, encerrando o ciclo, e tudo se repete na mesma seqüência, conforme ilustrado na figura 2.4. [MARTINS, 2006]

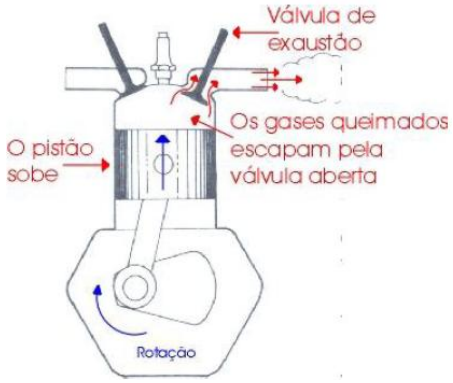

Figura 2.4 – Tempo de escapamento – Ciclo quatro tempos.

[RCMASTER, 2009

2.1.2. MOTOR DOIS TEMPOS.

O motor com dois tempos no ciclo é bastante simples. Possui poucas peças moveis. É mais leve e potente se comparado ao motor 4 tempos. Não existem as válvulas como no citado anteriormente, e a abertura e fechamento das janelas de admissão e escape são feitas pelo próprio pistão. [MARTINS, 2006]

Primeiro estagio (primeiro tempo): Ao subir comprimindo a mistura no cilindro, o pistão acaba produzindo uma rarefação no cárter. Quando o pistão chega bem próximo ao cabeçote ou ao ponto morto alto, dá-se a ignição e a combustão da mistura, é nesse momento também que ocorre a admissão da nova mistura no cárter, devido a nova subida do pistão, conforme ilustrado na Figura 2.5. [MARTINS, 2006]

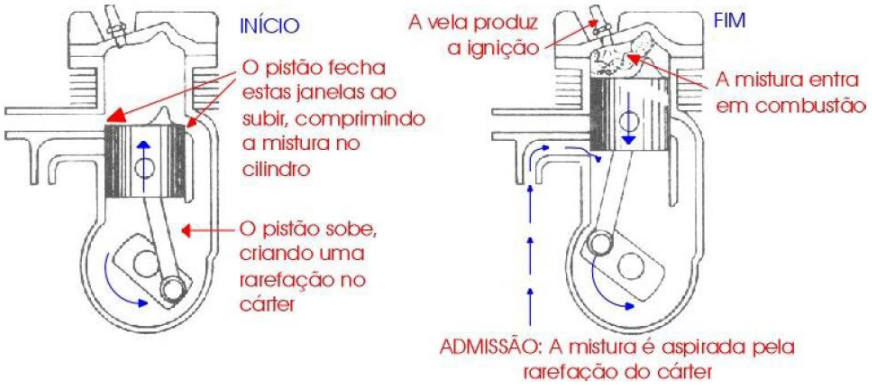

Figura 2.5 – 1° Tempo – Ciclo dois tempos.

[RCMASTER, 2009]

Segundo estágio (segundo tempo): Ao se expandirem, os gases da combustão empurram o pistão que comprime a mistura no cárter. Ao chegar ao seu ponto mais baixo, também conhecido como ponto morto baixo, o pistão abre a janela de exaustão permitindo que seja feita a saída dos gases queimados durante o processo. A janela de transferência é aberta e a mistura comprimida no cárter empurra os gases queimados ao invadir o cilindro. Finalizando assim o ciclo e tudo se repete na mesma seqüência, conforme ilustrado na figura 2.6. [MARTINS, 2006]

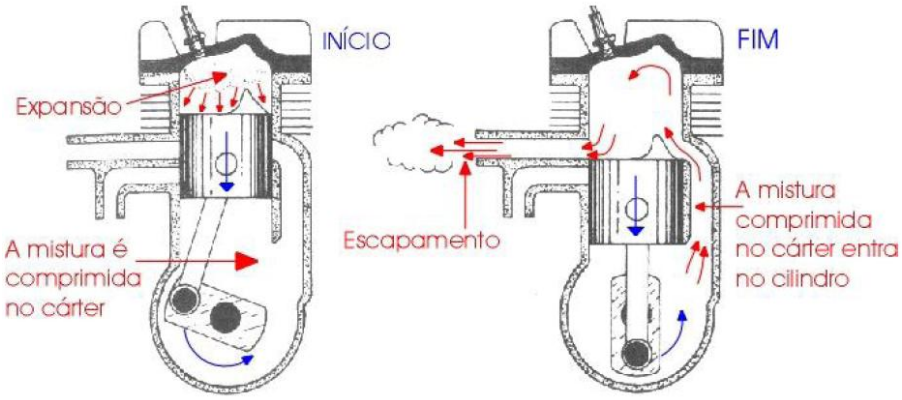

Figura 2.6 – 2° tempo – Ciclo dois tempos.

[RCMASTER, 2009]

2.2. *HARDWARE*

2.2.1. TELEMETRIA

A função da telemetria é transmitir dados para medições ou comunicação de informações. Ela auxilia o operador no funcionamento ótimo da máquina, e seu significado mais evidente é de uma medição realizada à distância. Por se tratar de uma tecnologia onde a maior vantagem é a troca de dados de forma instantânea, ou de funcionamento remoto, a tecnologia comumente utilizada é a transmissão sem fio. [MATTOS, 2004]

Sua utilização começou devida a necessidade da realização de medições em lugares inacessíveis como, por exemplo, na capacidade de teleguiar um míssel ou mesmo em medir a temperatura de um forno [MATTOS, 2004]. Na figura 2.7 é ilustrado o vôo de um míssel em uma área de teste. O míssil ou outro dispositivo em teste contém a bordo um pacote de telemetria de baixa potência, que pode ser chamado de módulo de telemetria. O módulo de telemetria contém os equipamentos que medem os parâmetros físicos no dispositivo em teste, o circuito condicionador de sinal que prepara os sinais para a transmissão para a estação de terra. O módulo de telemetria também contém um rádio transmissor e antena omnidirecional para suprir os sinais de rádio à estação receptora de telemetria. [MATTOS, 2004]

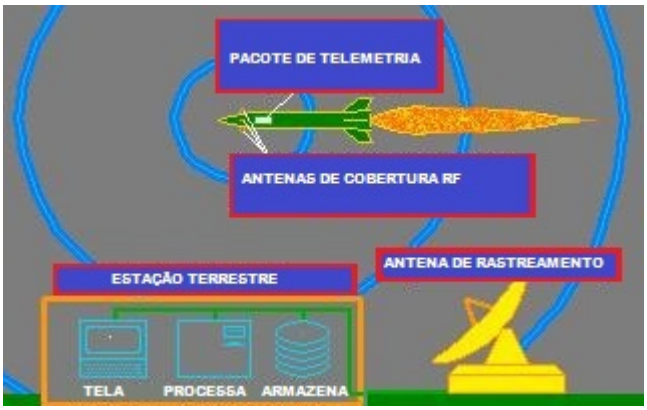

Figura 2.7 - Míssil em uma área de testes enviando sinais de telemetria .[MATTOS, 2004]

De longe, a maior quantidade de processamento realizado nos dados de telemetria não são em tempo real. Embora algum processamento pós-operacional possa ser iniciado durante o teste, não há necessidade de processá-los quase em tempo real. [MATTOS, 2004]

A telemetria geralmente refere-se à comunicações sem fio e usa um sistema de rádio para implementar um enlace de dados, porém pode também referir-se aos dados transferidos sobre outras mídias, tais como telefone, redes de computadores ou através de um enlace óptico. No setor automotivo e de logística as informações também podem ser transmitidas via GPRS juntamente com localização e outras informações de rastreamento. [MATTOS, 2004]. A metodologia de transferência utilizada no protótipo deste projeto de conclusão de curso é a transferência de dados através do cartão SD.

A telemetria é muito utilizada em competições automobilísticas, onde os engenheiros acompanham o que acontece com o carro durante uma prova. Porém, essa tecnologia também está bastante presente nas indústrias de

grande porte, em mísseis, no setor aéreo e mais recentemente vem sendo muito utilizada no campo da medicina. [MATTOS, 2004].

Considerando as corridas de Formula 1, os carros estão equipados com sensores que fazem as medições devidas, armazenam os dados e os transmitem. Essencialmente, esse estudo supervisiona o desempenho do carro de corrida e aperfeiçoa seu ajuste. Nesse esporte, pilotos e engenheiros precisam monitorar constantemente as reações que ocorrem devido a qualquer alteração executada. Parâmetros de motor, transmissão, suspensão ou até mesmo a pressão dos pneus são de extrema importância. Na figura 2.8 os engenheiros da Williams estudando o comportamento do carro durante a corrida. [COCCO; DAPONTE, 2008]

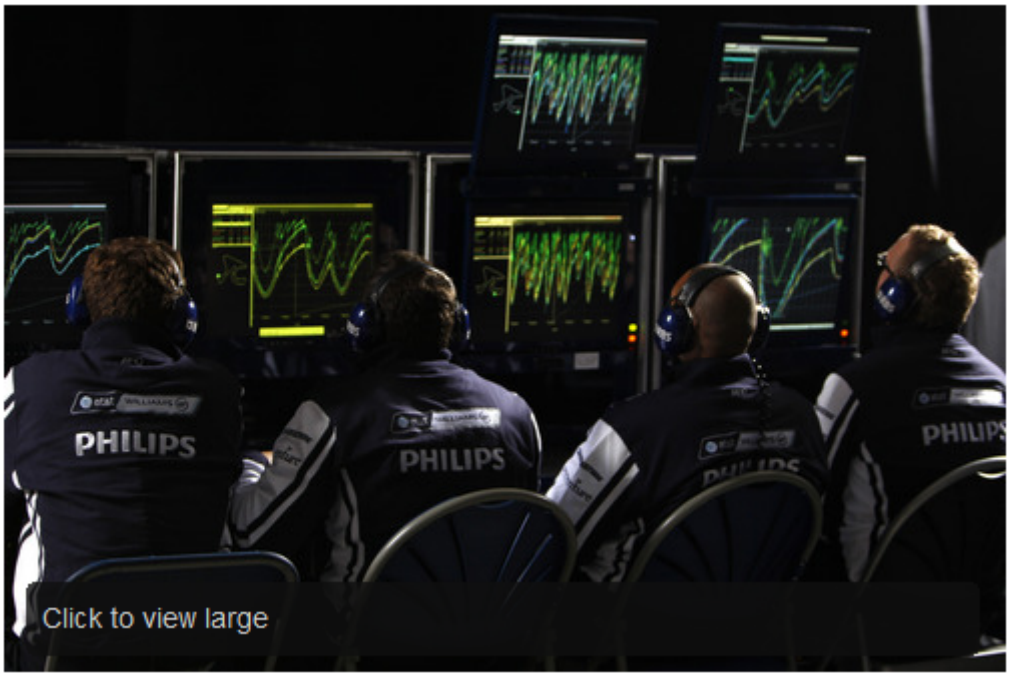

Figura 2.8 - Engenheiros e Telemetria

[BIZSOLUTION, 2009]

2.2.2. SENSORES

“Um transdutor é um dispositivo que converte um estímulo (sinal de entrada) em uma resposta proporcional (sinal de saída) adequada à transferência, energia, medição ou processamento da informação.” [NOLL; SOARES, 2009].

“Um sensor é o nome dado a um dispositivo que converte um estímulo (sinal de entrada) em uma resposta (sinal de saída) proporcional, adequando este sinal à transferência de energia, medição ou processamento da informação.” [NOLL; SOARES, 2009].

Logo, todos os sensores têm internamente transdutores, que são chamados transdutores do tipo sensor. Outro tipo de transdutor é o chamado transdutor atuador, que é capaz de interferir em um processo e controlar variáveis, como por exemplo, um resistor. [NOLL; SOARES, 2002] Sensores ou transdutor de aquisição tem como objetivo medir uma grandeza física, como temperatura. A entrada pode ser qualquer grandeza física que seja mensurável, mas a saída é quase sempre tensão ou corrente elétrica. Visto que a grandeza física pode mudar no tempo, a tensão tem que seguir a mudança. O tempo todo deve haver uma correspondência direta entre a tensão de saída do transdutor e a medida física.

Por exemplo, um sensor de temperatura, cada 1ºC a que for exposto, retorna um saída de tensão linear de 10mV como resposta ao circuito integrado, ou seja, há a correspondência entre uma medida física e a tensão elétrica. [NATIONAL SEMICONDUCTOR, 1994]

Nos carros, mesmo os de passeio, existem sensores que mostram ao motorista o que está acontecendo durante o seu uso. Como exemplo pode citar o medidor da pressão do óleo. No caso dos mais novos, como o Uno Economy, existe um indicador de consumo instantâneo no painel que indica a forma mais econômica de dirigir. Seu funcionamento é eletrônico e se baseia em informações de consumo instantâneo de combustível e velocidade do veículo.[ROSSETTI, 2009]

No protótipo é utilizado como sensor de passagem um *Reed Switch ,* um acelerômetro para a medição de aceleração lateral e um CI sensor LM35 para inferir a temperatura. Todos são especificados no capítulo de desenvolvimento.

2.2.3. CARTÃO DE MEMORIA SD

Evoluídos dos antigos disquetes, esses pequenos cartões fazem parte do cotidiano das famílias. Tem como objetivo maior o armazenamento de dados. Porém, adicionam a capacidade de criptografia e gestão de direitos digitais.[SD ASSOCIATION, 2009]

Como os *pen-drives*, este cartão utiliza memória *flash*, chips parecidos com os utilizados em memória RAM (*Randomic Access Memory*) sem a perda de dados quando não há mais o fornecimento de energia. [ALECRIM, 2005]. São pequenos, baratos e de grande capacidade de transferência de dados e armazenamento (os maiores já passam dos quatro gigabytes), e tem o consumo de energia pequeno, perfeito para os aparelhos portáteis. Leves, chegam a pesar aproximadamente dois gramas. Assim como os MMC,(*MultiMediaCard*), antigos disquetes, os cartões SD utilizam conectores

de contato metálico, evitando assim possíveis danos durante o manuseio. .[SD ASSOCIATION, 2009]

No mercado já são encontradas variações de Cartões SD, os Mini e os Micros SD. Funcionam da mesma maneira que o “irmão mais velho”, mas são menores. São muito encontrados nos novos telefones celulares. Apenas com o uso de um adaptador físico, o mini ou micro SD, se transformam em um SD de tamanho comum, ilustrado na figura 2.9.[SD ASSOCIATION, 2009].

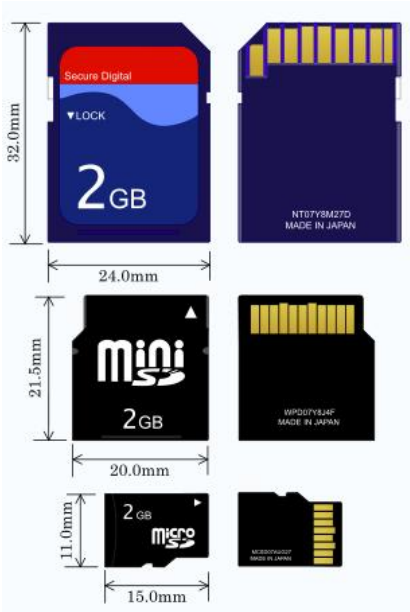

Figura 2.9 - Cartão SD
[SECURE DIGITAL CARD, 2009]

2.2.4. ACELERÔMETROS

O acelerômetro é um transdutor o qual transforma uma aceleração em sinal de tensão elétrica analógica. A empresa Analog Devices produz esses componentes há mais de 15 anos. Acelerômetros de baixa aceleração, entre 1g e 10g, são muito utilizados pela indústria automobilística na produção de

peças de segurança como *airbags*. São pequenos, baratos e precisam de pouca energia para funcionar. [ANALOG DEVICES, 2009].

O valor da aceleração inferida pelo acelerômetro é dada em g, ou aceleração provocada pela gravidade, aproximadamente 9,81m/s^2. Se o acelerômetro aferir 10g, então, diz-se que o corpo foi exposto a 10 vezes a aceleração da gravidade. [TIPLER; 1999]

Esse tipo de acelerômetro é utilizado em diversos tipos de aplicações. Estão presentes nos telefones celulares de ultima geração, Apple iphone, Nokia N95 e outros, em vídeo games, Nintendo Wii e nos GPS (*Global Position System*), fazendo medições inerciais. Sensores de abalos sísmicos também utilizam esse sistema para a medição de vibração e choque. [MASSUDA, 2007]

2.2.5. TECNOLOGIA MEMS

MEMS (*Micro-Electro-Mechanical Systems*) ou Sistema Microeletromecânico tem como proposta integrar elementos mecânicos, sensores, atuadores e dispositivos em micromáquinas. [MEMS AND NANOTECHNOLOGY CLEARINGHOUSE, 2009]

Para construir um acelerômetro deste tipo, é necessário mesclar elementos de circuitos integrados e micromáquina.

No caso do acelerômetro o polisilício e alumínio mostraram-se atrativos. No polisilício, a fadiga é insignificante e há pouca histerese, ou seja a perda de sua propriedade é pequena quando flexionado. O alumínio processa em baixa temperatura, simplificando a integração com a eletrônica. [BOSER; HOWE, 1995]

O diagrama conceitual é mostrado na figura 2.10.

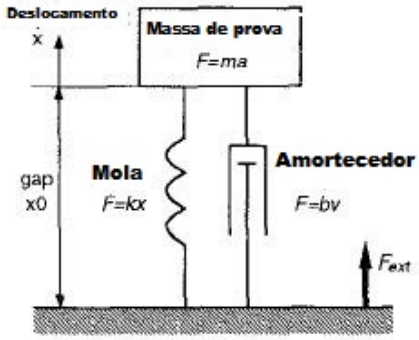

Figura 2.10 - modelo conceitual do acelerômetro

[BOSER; HOWE, 1995]

Através da equação diferencial do deslocamento em função de uma força externa, é encontrada uma força resultante igual a força elástica. Assumindo as relações de linearidade entre o coeficiente de amortecimento e as constantes envolvidas, constata-se que a aceleração é diretamente proporcional ao deslocamento. [MASSUDA, 2007]

$$a \propto x \tag{2.1}$$

Formado por diversos prolongamentos em forma de dedo, ilustrado na figura 2.11, placas paralelas de capacitores medem o deslocamento do corpo de prova. A capacitância então pode ser comparada ao sistema massa mola, onde as superfícies das placas e a permissividade elétrica do dielétrico são constantes. [MASSUDA, 2007]

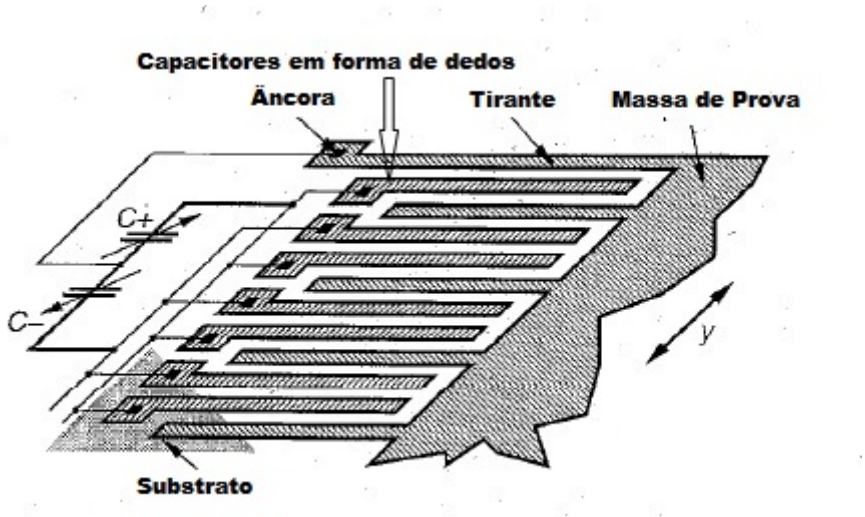

Figura 2.11 – Capacitores em forma de dedos.

[BOSER; HOWE, 1995]

A capacitância é proporcional a carga elétrica “q” e inversamente proporcional a tensão elétrica “V” [MASSUDA; 2007]

$$C = \frac{q}{V} \tag{2.2}$$

Cria-se então uma relação de proporcionalidade da tensão elétrica “V” e o deslocamento “x”. Conclui-se então que:

$$a \propto V \tag{2.3}$$

Calculando o deslocamento de “x” em função da capacitância, tem-se que a aceleração é calculada em função da tensão e suas constantes.

$$x = \frac{\varepsilon A V}{q} \tag{2.4}$$

$$a = V\left(\frac{k\varepsilon A}{qm}\right) \tag{2.5}$$

2.2.6. MICROCONTROLADOR

Um microcontrolador é um sistema de microprocessadores que contém dados em programas de memória, interface serial, interface paralela, temporizadores, controladores de interrupção, controladores de eventos, todos integrados em um único chip que pode ser comprado aproximadamente por R$ 150,00. Microcontroladores estão presentes em quase todos os equipamentos eletrônicos. Geralmente os microcontroladores estão embarcados nos equipamentos. [IMBRAHIM, 2008]

Normalmente, os microcontroladores são programados utilizando a linguagem de programação *Assembly,* por ser rápida, porém de difícil compreensão. Podem ser programados utilizando ainda as linguagens BASIC, PASCAL ou C. [IMBRAHIM, 2008]

Um microcontrolador executa um programa que está gravado em sua memória. Controla, assim, dados recebidos dos componentes externos de entrada, manipulando e enviando-os para os componentes de saída. [IMBRAHIM, 2008]

Pode-se exemplificar o uso de um microcontrolador em um simples sistema de temperatura, que inclui um teclado para especificar os controles do sistema, um LCD para exibi-la, tudo controlado por um microcontrolador, conforme ilustrado na figura 2.12. [IMBRAHIM, 2008]

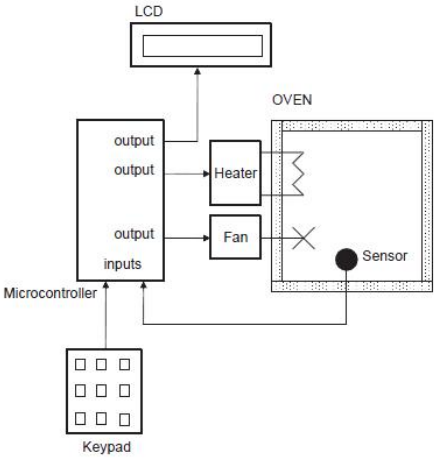

Figura 2.12 – Sistema de controle de temperatura.

[IMBRAHIM, 2008]

Suas principais vantagens são: [IMBRAHIM, 2008]

- Portabilidade;
- Baixo consumo de energia ;
- Recompilação de *software;*
- *Baixo custo.*

No protótipo desse projeto, é utilizada uma placa física com código aberto baseada em um circuito de entrada e saída, Arduíno *Duemilanove*, que utiliza o microcontrolador Atmel ATMEGA 328 e faz o controle dos sensores e grava no cartão SD. [ARDUINO, 2009]

2.3. SOFTWARE

2.3.1. LINGUAGEM C

A linguagem “C” é vista como puro sangue das linguagens de programação e foi desenvolvida em 1970, no *Bell Labs*, por Brian Kernighan e Dennis Ritchie. [KERNIGHAN; RITCHIE, 1988]

Ritchie, em conjunto com Ken Thompson, também desenvolveu o sistema operacional Linux, que veio a utilizar a linguagem “C” como código fonte. [KERNIGHAN; RITCHIE, 1988]

Antes de ser integrado à linguagem “C”, o sistema operacional Linux era todo escrito em assembly , que exigia uma trabalhosa modificação para cada processador em que ele fosse executado. Por outro lado, com o desenvolvimento de um compilador “C”, tornou-se uma linguagem portátil, porque os programas escritos nessa linguagem poderiam ser facilmente transferidos e recompilados para os computadores que tivessem o compilador C.[NORTON, 1997]

A linguagem “C” também é considerada, relativamente, como uma linguagem de plataforma baixa, ou seja, pode-se dizer que “C” lida com os mesmos tipos de objeto que os computadores, números e endereços, por exemplo. É muito poderosa para quem visa programar componentes como os microcontroladores.[KERNIGHAN; RITCHIE, 1988]

Os programas escritos em “C” produzem um código executável rápido e eficiente. Devido a liberdade de programação, a linguagem “C” tornou-se extremamente popular e ainda hoje é a linguagem mais comum entre os desenvolvedores de *software*.

O Arduíno, placa de desenvolvimento escolhida para o desenvolvimento desse trabalho, utiliza essa linguagem como ferramenta.

2.3.2. LINGUAGEM C#.NET

Moderna e orientada a objetos, a linguagem “C#”, pode ser vista como uma extensão de outras linguagens, comparadas ao Java e ao C++. Poderosa, porém simples, é voltada principalmente para desenvolvedores que criam aplicativos usando *Microsoft* .NET *framework*. Desenvolvida pela *Microsoft*, o *framework* .NET é uma plataforma de desenvolvimento e execução de sistemas, já se encontra na versão 3.5. [SHARP, 2007].

O *Microsoft* .NET é a primeira plataforma para desenvolvimento de software a ser desenhada, desde a estaca zero, tendo a Internet em mente, embora não seja exclusiva para esse tipo de desenvolvimento. Fornece um modelo de programação consistente que pode ser usada em diversos tipos de aplicações. [SHARP, 2007]

Abrangente e bastante ambiciosa, a plataforma .NET engloba várias linguagens e mais bibliotecas de classes completas, fornecendo assim funcionalidade embutida.

Os principais objetivos do .NET são: [BARWELL et al, 2004]

- Criar aplicativos altamente distribuídos, ou seja, que não precisam estar localizados exclusivamente dentro de uma única organização;
- Simplificar o desenvolvimento de software;
- Simplificar a utilização das bibliotecas, de forma que apenas copiando um modulo compilado para o sistema e executando, já possa ser utilizado;
- O suporte com uma série de linguagens, por exemplo, o C# e VB.net;

A facilidade de integração com o banco de dados, ferramentas visuais para construção das telas e vasta bibliografia encontrada, foram pontos importantes para a escolha do uso dessa linguagem no projeto.

2.3.3. BANCO DE DADOS

Um sistema de gerenciamento de banco de dados é a ferramenta que os computadores usam para obter o processamento e o armazenamento organizado dos dados. O banco de dados é um depósito de conjuntos de dados relacionados. Por exemplo, uma agenda de endereços e número de telefones de amigos e contatos comerciais. As informações podem ser acrescentadas ao banco de dados e mais tarde extraídas sem perder seu significado. [NORTON, 1996]

Os microcomputadores trouxeram o gerenciamento de banco de dados para a mesa dos indivíduos comuns, na área domestica e comercial. É comum encontrar um amigo que faz a gerência das compras ou mesmo as contas domésticas, registros telefônicos com auxilio dos SGBDs (sistema de gerenciamento de banco de dados).

SGBD é um programa ou um conjunto de programas, que armazena dados de modo a permitir que eles sejam acessados a qualquer momento. Na verdade, um dos melhores motivos para utilizar um banco de dados não é só o armazenamento de dados, mas a velocidade de acesso e a manipulação facilitada também são pontos fortes dessa tecnologia. [NORTON, 1996]

Assim como os sistemas operacionais, o SGBD é capaz de prestar serviços tanto ao usuário quanto a outros programas, como por exemplo, um

acesso aos dados do banco de dados através de um sistema desenvolvido em uma linguagem como a C#.NET, citada anteriormente.

2.3.4. SQLITE

SQLite é uma biblioteca que implementa o banco de dados SQL(*Structured Query Language*) ou simplesmente linguagem de consulta estruturada*.* Diferentemente dos outros bancos, não tem um servidor de dados separado.Ele lê e escreve no disco um banco de dados SQL completo, em um único arquivo. Seu arquivo de banco de dados tem extensão multi-plataforma e funcionam em sistemas 32 e 64 bits. [SQLITE, 2009]

Algumas vantagens do SQLite [SQLITE, 2009]:

- *Software* livre e de domínio público;
- Mecanismo de armazenamento seguro;
- Não necessita de instalação, configuração ou administração;
- O banco de dados é guardado em um único arquivo;
- Não tem dependência externa.

3 - DESENVOLVIMENTO

A implementação deste projeto faz a integração entre a eletrônica e computação. Para a construção do protótipo é necessário conhecimento aplicado nas áreas de eletrônica. Na implementação do *software*, banco de dados e programação do microcontrolador, foi utilizado todo conhecimento visto nas matérias de computação.

Neste capitulo, é explicada a construção e programação do protótipo e software de análise *off-line* proposto.

Esse projeto apresenta duas vertentes: no primeiro momento os dados físicos (aceleração linear e lateral, temperatura e tempo de volta) são aferidos em carro de passeio através dos sensores embarcados. Em tempo real, os dados são gravados em um arquivo texto através porta USB. Posteriormente, os dados são importados pelo software, que gera os gráficos de temperatura e aceleração e grava no banco de dados. Apesar de ser um computador de bordo destinado ao kart, o protótipo construído não foi embarcado nesse veículo por motivos de mau contato devido à trepidação.

3.1. DESENVOLVIMENTO DE *SOFTWARE*

Visando o desenvolvimento de um *software* simples, mas preocupado em facilitar a manutenção, a implementação do *software* de anaálise *off-line* utilizou orientação objeto e *design patterns*.

3.1.1. APLICATIVO C#.NET – *SOFTWARE* OFF-LINE

Para programar o *software* que gera os gráficos, importa e armazena os dados no banco de dados, foi utilizado o pacote de desenvolvimento Microsoft Visual Studio 2008. Este utiliza o *framework* .NET versão 3.5. [MSDN, 2009].

Dentre as linguagens utilizadas por esse *framework*, a escolhida foi o C# pois além da construção de telas e a integração com banco de dados serem implementadas de forma simples, esta linguagem permite um controle de tipos e interação de variáveis em baixo nível, permitindo maior controle do *software*.

No desenvolvimento do aplicativo, foram utilizados alguns tipos de padrões de desenvolvimento de software, *Design Patterns.* Trata-se do uso das melhores práticas de programação orientadas a objeto. [GAMMA ET AL, 2000].

Todo o desenvolvimento foi feito utilizando o conceito MVC (*Model-View-Controller*), onde sua estrutura é bem definida em apresentação, objeto e controlador, aumentando a flexibilidade e reutilização dos objetos no código. O Modelo é o objeto de aplicação, a Visão é a apresentação exibida ao usuário na tela e o Controlador é maneira como a interface do usuário reage as entradas de dados. [GAMMA ET AL , 2000].

O sistema é subdividido em pequenos subprojetos:

- *Framework*;
- KartComp2 (Apresentação);
- NegocioKartComp;
- PersistenciaKartComp;
- TransferObjectKart.

O primeiro subprojeto, *framework*, é referenciado por todos os outros subprojetos. É nele que estão as classes abstratas que são utilizadas nos outros subprojetos. No subprojeto KartComp2, estão as telas e seus controladores. Toda a parte que faz referência a apresentação, está nessa sessão. No NegocioKartComp, estão as classes de negócio do sistema. As classes que fazem acesso a banco de dados estão em PersisteciaKartComp. Por fim, as classes que são transportadas entre as camadas, chamada *Value Objects,* estão no subprojeto TransferObjectKart.

A implementação do *software off-line* utiliza a arquitetura de desenvolvimento adotada nos sistemas feitos pelo Centro de Formação de Recursos Humanos em Transportes – Ceftru/UNB.

3.1.2. ARQUITETURA

Desenvolvido pelo centro de formação de recursos humanos em transportes, CEFTRU, este tipo de arquitetura visa o baixo custo de manutenção, reusabilidade, extensibilidade e principalmente a manutenibilidade. [CEFTRU, 2007]

O sistema feito para analisar os dados foi feito em três camadas: aplicação, negócio e persistência, ilustradas na figura 3.1.

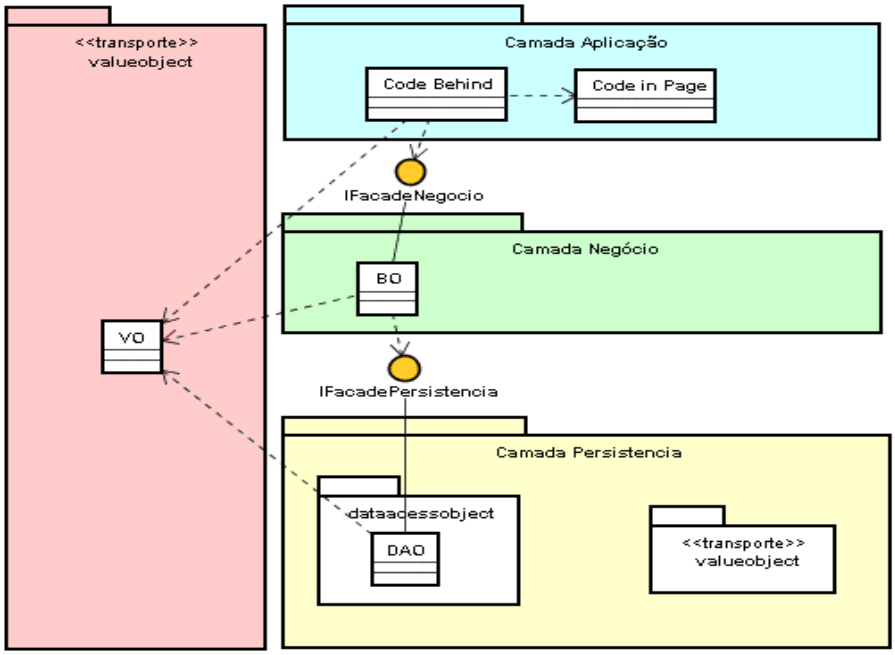

Figura 3.1 – Esquema da Arquitetura do Sistema.

[CEFTRU, 2007]

Na camada de aplicação estão as classes específicas que controlam as telas que fazem a interface com o usuário. Na camada de negócio, ficam as classes com as regras negociais do sistema. A última camada tem a função exclusiva em fazer o acesso ao banco de dados.

O *Value Object* (VO) coleta um conjunto de informações relacionadas a um único objeto e as transporta entre as camadas, ilustrado na figura 3.2. Para cada tabela existe um VO especifico, relacionando os atributos de classe com os campos da tabela do banco de dados.

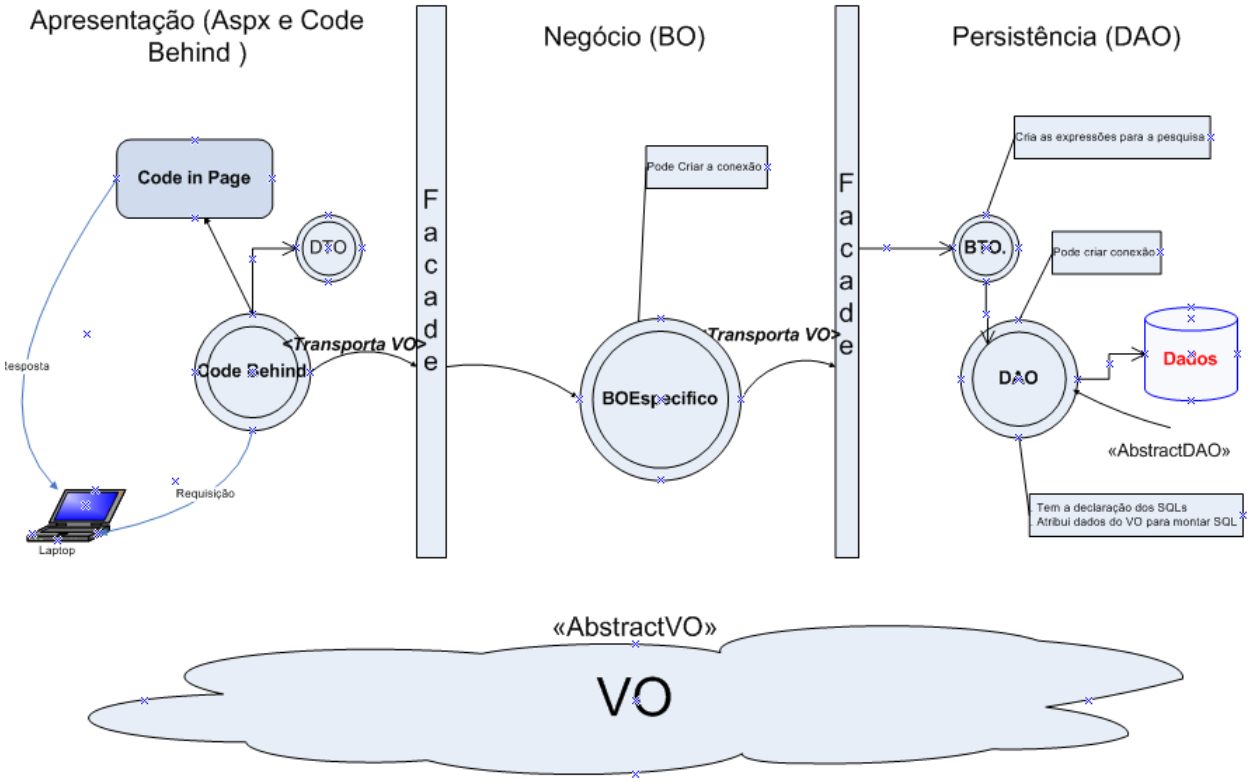

Figura 3.2 – Funcionamento da Arquitetura.

[CEFTRU, 2007]

O cadastramento de um piloto é o exemplo utilizado para explanar o funcionamento dos cadastros feitos pelo usuário do sistema. Todos os cadastros desse sistema têm a mesma estrutura.

Após construir a tela de entrada de dados, é feita uma classe que tem a função de controlar esse formulário, ações e dados. A Figura 3.3 ilustra a classe controladora da tela de cadastro de piloto.

```
namespace kartComp2.src.br.apresentacao
{
    class CtrlCadastrarPiloto
    {


        private FormCadastrarPiloto frmCadastraPiloto;
        private FacadeNegocioKartComp vFacade;

        Piloto vPiloto = null;
        AbstractBO vBO;

        public string ConteudoDescricao { get; set; }

        public CtrlCadastrarPiloto() { }

        // Inicializa a tela
        public void iniciar()...

        //Cria os eventos dos componentes da tela(Botões, Grid)
        protected void criaEventos()...
        //Executa a pesquisa de Pilotos quando o botão pequisar é acionado
        void btnPesquisar_Click(object sender, EventArgs e)...
        //Preenche um Value Object com os valores do campo
        public Piloto  formToVo() ...
        //Limpa os campos da tela quando acionado
        void btnReset_Click(object sender, EventArgs e)...

        //Grava o piloto quando botão for pressionado.
        void btnGravar_Click(object sender, EventArgs e)...
        //Fecha a Tela quando pressionar o botão fechar
        void btnFechar_Click(object sender, EventArgs e)...
        //Executa a rotina de carregar o objeto de piloto
        public void carregaVoPiloto(int pCodPiloto, string pNome, string pCategoria) ...
        //Carrega o objeto de Piloto com dados do Banco
        public Piloto carregaParametrosPiloto(Piloto pPiloto) ...
        //Método que grava o objeto de Piloto no Banco
        public void gravaPiloto(Piloto pPiloto)...
        //Método que exclui o piloto do banco
        public void excluirPiloto(Piloto pPiloto) ...
        //Método que carrega Piloto do banco sem filtro
        public void carregarTodosPilotos()...
        //Método que carrega o DataGrid
        public void carregaDataGrid(DataSet pDSPiloto)...
        //Metodo que executa uma rotina quando algum piloto é selecionado no datagrid
        void dgPiloto_SelectionChanged(object sender, EventArgs e)...
        //Metodo que executa uma rotina quando é alterado um valor no datagrid
        void dgPiloto_CellValueChanged(object sender, System.Windows.Forms.DataGridViewCellEventArgs e)...
        //Metodo que executa uma rotina quando se edita o campo do datagrid
        void dgPiloto_CellBeginEdit(object sender, System.Windows.Forms.DataGridViewCellCancelEventArgs e)...
        //Exclui o piloto selecionado
        void btnExcluir_Click(object sender, EventArgs e)...

    }
}
```

Figura 3.3 – Classe de Cadastro de Pilotos

Em todas as classes que fazem controle de tela desse sistema, quatro métodos sempre são instanciados:

- Iniciar(): Faz a inicialização da tela de apresentação e chama as rotinas que precisam executar para o preenchimento de cada objeto colocado na tela.

- CriaEventos(): Define a rotina de cada objeto que executa um evento por exemplo, a rotina que será executada ao clicar em um botão.
- FormToVo(): Faz o preenchimento de um objeto tipo VO (*Value Object)* com os dados de entrada feitos pelo usuário.
- VoToForm(): Preenche os objetos de tela com os dados que são carregados no VO (*Value Object*) do banco de dados.

Os eventos dos botões da tela fazem a chamada de classes do tipo BO (*Business Object),* onde são instanciadas as classes que fazem acesso ao banco de dados, classes do tipo DAO (*Data Access Object*), onde são feitas as transações com o banco de dados.

A codificação do cadastro de piloto exemplifica bem o processo de acesso ao banco de dados. Ao acessar a tela de cadastro de piloto, o usuário do sistema preenche os campos e grava o piloto. A classe controladora dessa tela tem um método gravaPiloto, figura 3.4. Esse método recebe um objeto do tipo piloto, ou seja, um *Value Object* de piloto que será transportado para a classe de negócio, AbstractBO.

```
public void gravaPiloto(Piloto pPiloto)
{
    AbstractBO bo = null;

    try
    {
        bo = new AbstractBO();
        bo.gravar(pPiloto);

    }
    catch (Exception)
    {
        throw;
    }
}
```

Figura 3.4 – Método de gravação do Piloto

A classe de negócio AbstractBO é responsável em fazer a interface entre o controlador da tela de piloto e a classe que faz o acesso ao banco de dados. Por se tratar de uma classe genérica ela foi utilizada para fazer a inserção, alteração e exclusão de todos os cadastros feitos pelo usuário, apenas trocando o parâmetro passado, correspondente ao *Value Object* específico que se deseja gravar, alterar ou excluir. A classe AbstractBO transporta o objeto do tipo piloto, recebida na classe controladora para a classe AbstractDAO.

A AbstractDAO é um classe que tem como função específica o acesso ao banco de dados. É nessa classe que é feito o comando SQL ao banco de dados e neste caso a inserção do *Value Object* de Piloto no banco de dados. Figura 3.5.

```
public void inserir(ref VO pVO)
{
    IDbCommand vComm = null;
    DbDataReader vDR = null;

    try
    {
        vComm = this.criaCommand();
        // Monta o SQL especifico para tipo especifico de Objeto recebido
        vComm.CommandText = getSQLIncluir(pVO);

        montaCommandParameters(true, ref pVO, ref vComm);

        vComm.ExecuteNonQuery();//executa o comando no banco de dados

    }
    catch (ExceptionAbstract ex)
    {
        throw (ex);
    }
    catch (Exception ex)
    {
        throw new ExceptionFramework(ex.Message, ex);
    }
    finally
    {
        if (vDR != null)
        {
            vDR.Close();//fecha a transação
            vDR.Dispose();
        }
        fechaConnexao();//fecha a conexão com o banco
    }
}
```

Figura 3.5 – AbstractDAO

3.1.3. IMPORTAÇÃO DOS DADOS

Depois de capturar os dados físicos durante o evento na pista, estes são transferidos a um arquivo texto, intervalados por caracteres especiais ponto e vírgula (“;”), onde cada posição será um atributo diferente. A programação de captura dos dados e construção do arquivo texto é ilustrado na figura 3.6.

TimerFunc §

```
#include <MsTimer2.h>

int AN_AC_X = 0;
int AN_AC_Y = 1;
int AN_TEMP = 2;
int fezCaptura = 0;
int fezVolta = 0;
int iniciou = 0;
int diff = 0;
int milli = 0;
int state = 0;

void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(19200);

  //Quando é interrupção ele conta um trecho
  attachInterrupt(0, FazVolta, RISING);
  //a cada 500 ms é feita a captura de dados
  MsTimer2::set(500,FazCaptura);
  MsTimer2::start();

 }

void loop() {
  digitalWrite(13,state);
  if (fezVolta == 1){
    fezVolta = 0;
    if (iniciou == 0){
      iniciou = 1;
      milli = millis();
    }else{
      diff = millis() - milli;
      milli = millis();
      Serial.print("1;");
      Serial.print(diff);
      Serial.println(";");
    }
  }
    //valida captura
    if ((fezCaptura == 1)&&(iniciou==1)){
        fezCaptura = 0;
        diff = millis() - milli;
        milli = millis();
        //Escreve o arquivo com os tipos corretos
        Serial.print("2;");
        Serial.print(diff);
        Serial.print(";");
        //Aceleração em X
        Serial.print(analogRead(AN_AC_X));
        Serial.print(";");
        //Aceleração em Y
        Serial.print(analogRead(AN_AC_Y));
        Serial.print(";");
        //Temperatura
        Serial.print(analogRead(AN_TEMP));
        Serial.println(";");
    }
 }

void FazVolta(){
  fezVolta = 1;
  state = !state;
}

void FazCaptura(){
  fezCaptura = 1;
}
```

Figura 3.6 – Construção do Arquivo Texto

O arquivo gerado tem os dados dispostos, conforme a tabela 3.1

Tabela 3.1 – Características do Arquivo gerado

Campo 1	Tipo de gravação do dado
Campo 2	Tempo em milissegundos da diferença a aferição do dado anterior
Campo 3	Aceleração linear em milivolt
Campo 4	Aceleração lateral em milivot
Campo 5	Temperatura em milivolt

O arquivo gerado, tem em seu primeiro caractere, de cada linha, um numero identificador, “1” – sempre que passar nas faixas magnéticas da pista e um número “2” – atualização a cada meio segundo. Em todas as linhas, o segundo intervalo é a diferença do tempo de aferição da linha anterior, dado em milissegundos. Caso a linha tenha como identificador o número “2”, existem mais três intervalos além dos citados anteriormente: aceleração longitudinal, aceleração lateral e temperatura,, expressos em milivolt, Ilustrado na figura 3.7.

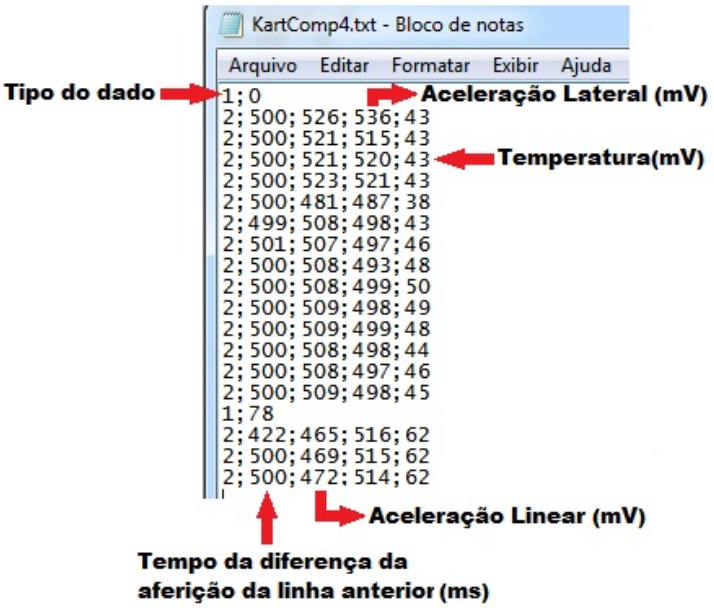

Figura 3.7 – Arquivo texto

Para que a importação seja feita de forma correta, existe no sistema *off-line* uma classe que percorre todas as linhas geradas no arquivo texto pelo sistema embarcado, e preenche as tabelas: tblParametros, tblPilotoEvento, tblVolta e tblTrechos. A rotina que faz a leitura do arquivo texto é feita na classe CtrlImportação, método lerArquivo, figura 3.8.

```
public void lerArquivo()
{
    //caminho do diretorio
    StreamReader rd = new StreamReader(@"C:\02 - Projeto Final\kartComp.txt");
    Volta voltaVO = new Volta();
    Trecho trechoVO = new Trecho();
    Parametros parametroVO = new Parametros();

    int TempoVolta = 0;
    int TempoTrecho = 0;

    //faz a validação do arquivo

    while (!rd.EndOfStream)
    {
        string linha = rd.ReadLine();//Lê a linha e atribui na variável linha

        if (linha.Length != 0)
        {
            //Grava em uma variável array os numeros separados por ";"
            string[] linhaNova = linha.Split(new char[] { ';' });

            //Para cada posição do array atribui a variavel correspondente
            int n = Int32.Parse(linhaNova[0]);
            int tempo = Int32.Parse(linhaNova[1]);
            if (n == 2)
            {
                pAceleracao = Int32.Parse(linhaNova[2]);
                pAceleracaoLateral = Int32.Parse(linhaNova[3]);
                pTemperatura = Int32.Parse(linhaNova[4]);
            }
```

Figura 3.8 – Trecho do método lerArquivo

3.1.4. SQLITE

As configurações de conexão do banco de dados SQLite foram feitas por um arquivo texto com extensão XML(*Extensible markup language*), ilustrado

na figura 3.9. Nesse arquivo são especificados o tipo, o nome e o caminho da conexão.

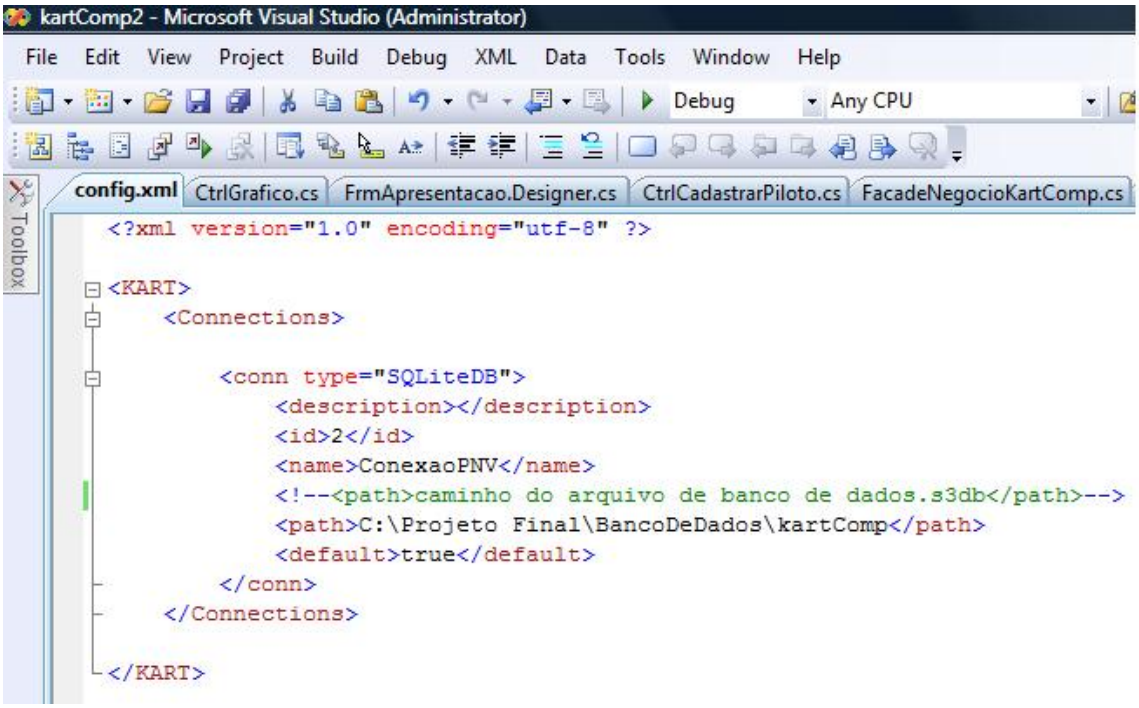

Figura 3.9 - Arquivo de configuração do banco de dados

Visando simplificar sua construção, o banco de dados foi desenvolvido na ferramenta SQLite Studio v1.1.1 através de interface gráfica.

O banco de dados foi implementado utilizando as técnicas de modelos de dados relacionais. Atualmente essa técnica é bastante utilizada em aplicações comerciais. [MACHADO, 2004]

O sistema *off-line* conta com 8 tabelas, onde as informações de piloto, pista, evento e equipamento são feita através de cadastro e as outras tabelas são preenchidas através do processamento dos dados adquiridos pelo sistema embarcado. O modelo de dados é ilustrado na figura 3.10.

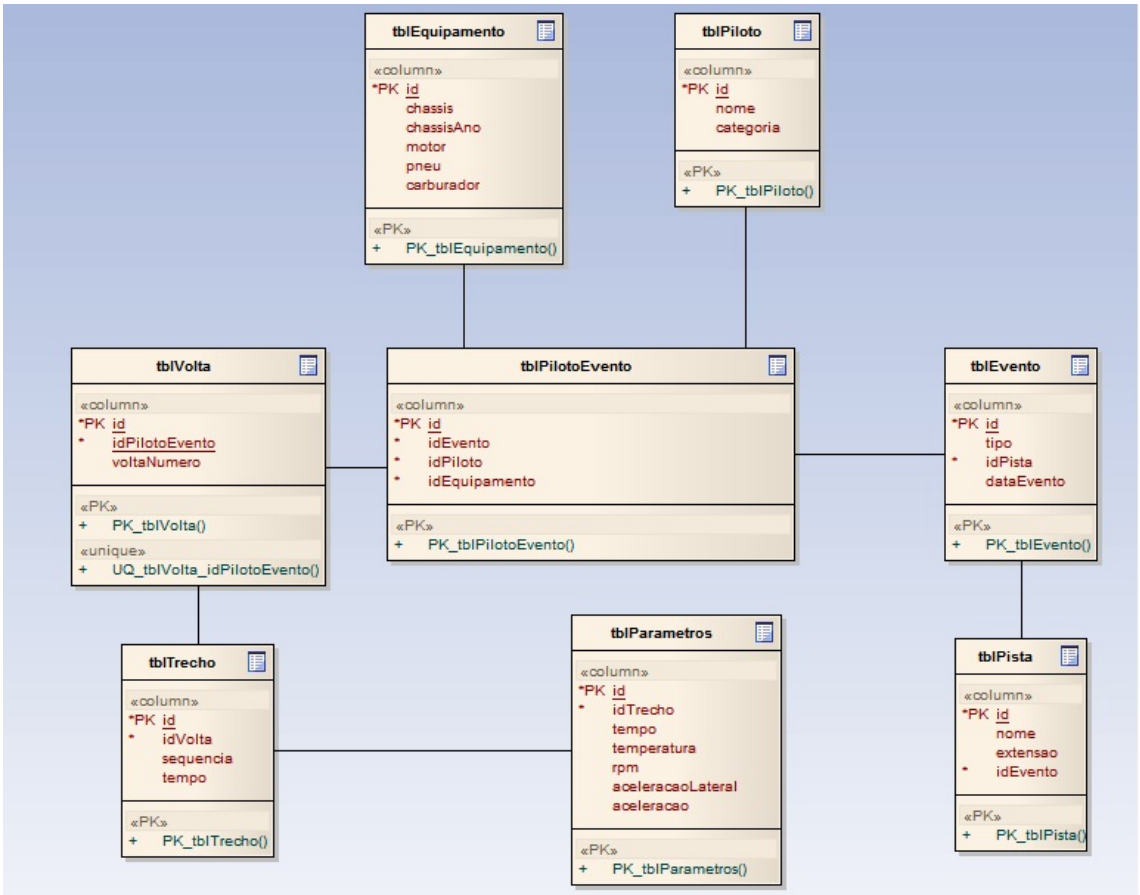

Figura 3.10 – Modelo de Dados

Com os dados persistidos no banco de dados, é possível a construção de gráficos no sistema *off-line*.

3.1.5. GRÁFICOS

A implementação do sistema *off-line* foi desenvolvida com a preocupação de auxiliar o piloto e chefe de equipe na melhoria do veículo. A forma escolhida para fazer comparativo entre os pilotos e os dados adquiridos em pista é através de gráficos.

A construção dos gráficos é feita pela leitura de dados que estão persistidos no banco, anteriormente gravados pela importação.

O *software off-line* lê e trata as informações que serão escritas nos gráficos através da classe ctrlGrafico. Para cada tipo de gráfico, existe um método que faz o seu preenchimento, figura 3.11.

```
private void populaGraficoTemperatura()
{
    int x = 0;
    double y = 0;

    Random rnd = new Random();
    vParametro = new Parametros();
    DataSet vDS = new DataSet();
    facade = new FacadeNegocioKartComp();

    try
    {   //leitura da lista de piloto
        for (int i = 0; i < frmGrafico.grfAceleracao.Series.Count; i++)
        {
            //this.carregarAceleracao();
            x = 0;
            //leitura da lista com os parametros para cada piloto
            foreach (DataRow dr in vListaParametros[i].Tables[0].Rows)
            {
                //Preenche valores de y
                //Conversão do valor gravado no banco para °C.
                y = (500 * double.Parse(dr["temperatura"].ToString())) / 1023;
                //Preenche valores de x
                x = x + int.Parse(dr["tempoParametro"].ToString());

                atualizarGrafico(frmGrafico.grfAceleracao.Series[i], y, x, 5);

            }


        }


    }
```

Figura 3.11 – Método populaGraficoTemperatura

Para montar o gráfico, foi utilizado a biblioteca *MSChart* da *Microsoft*. Na classe CtrlGrafico, foi implementado um método, initChart, figura 3.12, com a função de inicializar e preencher os valores padrões do gráfico, como por exemplo, o título dos eixos, o tipo de gráfico (linha, barra ou pizza) e os valores máximo e mínimo.

```
//Pontos atribuídos ao gráfico.
private void atualizarGrafico(Series piloto, Double pYValue, int pXValue, Double pDefaultSerieChart)
{
    frmGrafico.grfAceleracao.Series[int.Parse(piloto.Tag.ToString())].Points.AddXY(pXValue, pYValue);
}
//Montagem do gráfico
private void initCharts()
{
    try
    {
        //Y - Eixo dos Parametros
        frmGrafico.grfAceleracao.ChartAreas[0].AxisY.Minimum = vYMin ;
        frmGrafico.grfAceleracao.ChartAreas[0].AxisY.Maximum = vYMax ;
        frmGrafico.grfAceleracao.ChartAreas[0].AxisY.Interval = vIntervalY ;
        frmGrafico.grfAceleracao.ChartAreas[0].AxisY.Title = vTitley ;
        //X - Eixo da Tempo
        frmGrafico.grfAceleracao.ChartAreas[0].AxisX.Minimum = vXMin ;
        frmGrafico.grfAceleracao.ChartAreas[0].AxisX.Maximum = vXMax ;
        frmGrafico.grfAceleracao.ChartAreas[0].AxisX.Interval = vIntervalX ;
        frmGrafico.grfAceleracao.ChartAreas[0].AxisX.MajorGrid.Enabled = false;
        frmGrafico.grfAceleracao.ChartAreas[0].AxisX.Title = vTitleX ;

    }
    catch (Exception)
    {
        throw;
    }
}
```

Figura 3.12 – Método de inicialização dos gráficos

Neste trabalho, o sistema *off-line* conta com três gráficos: Aceleração Linear por tempo, figura 3.13, Aceleração Lateral por tempo, figura 3.14 e Temperatura por tempo, figura 3.15.

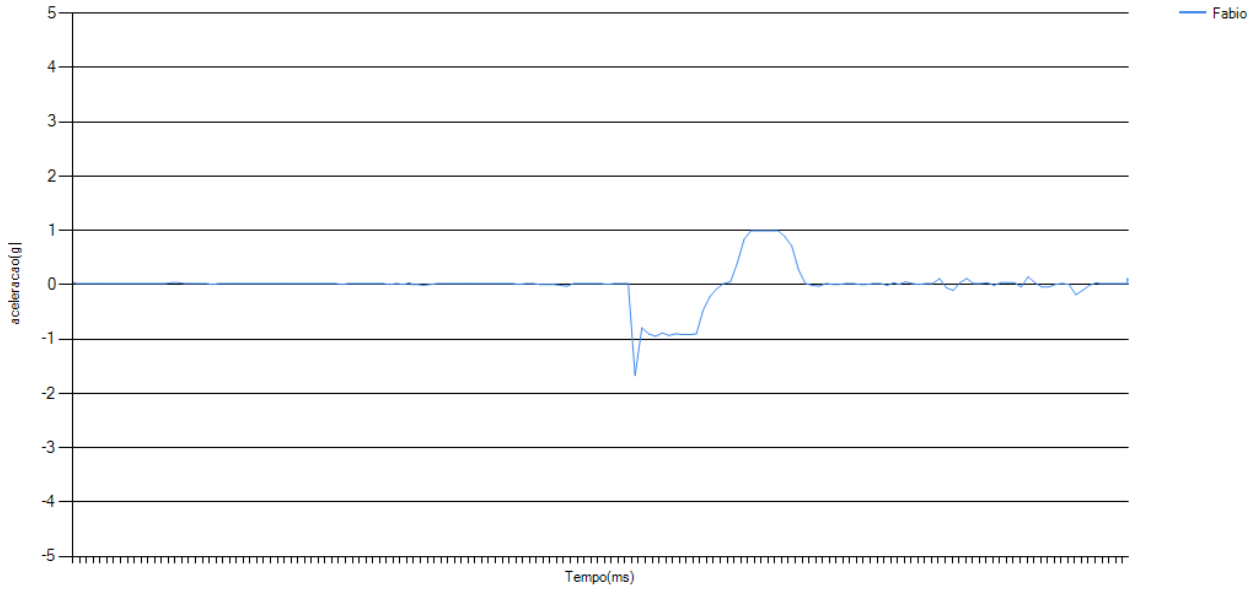

Figura 3.13 – Aceleração Linear x Tempo

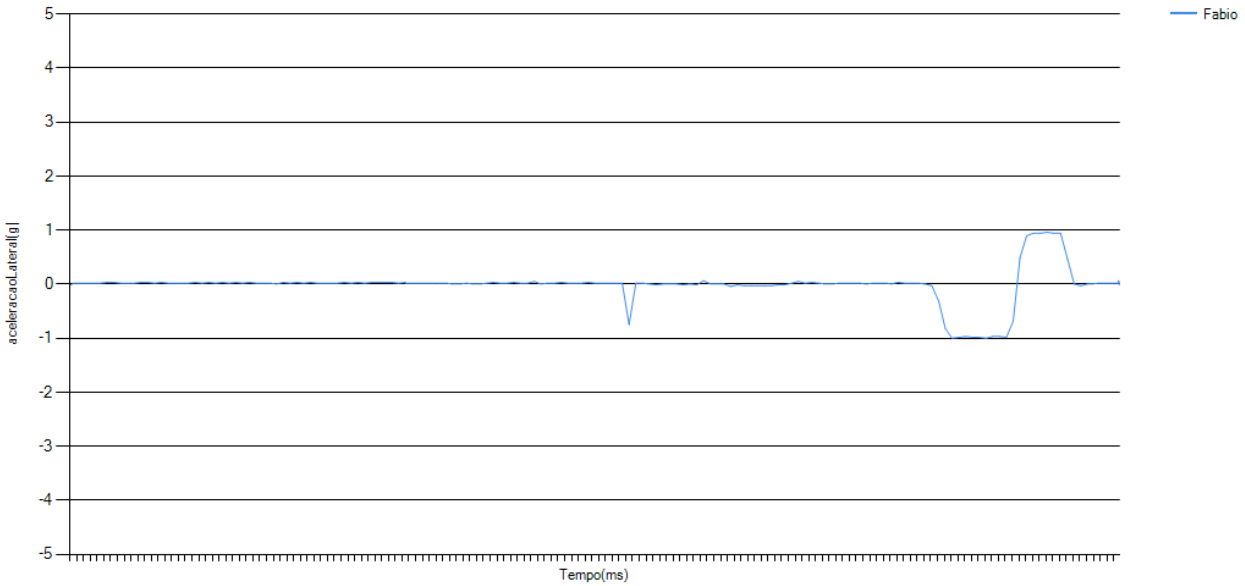

Figura 3.14 – Aceleração Lateral x Tempo

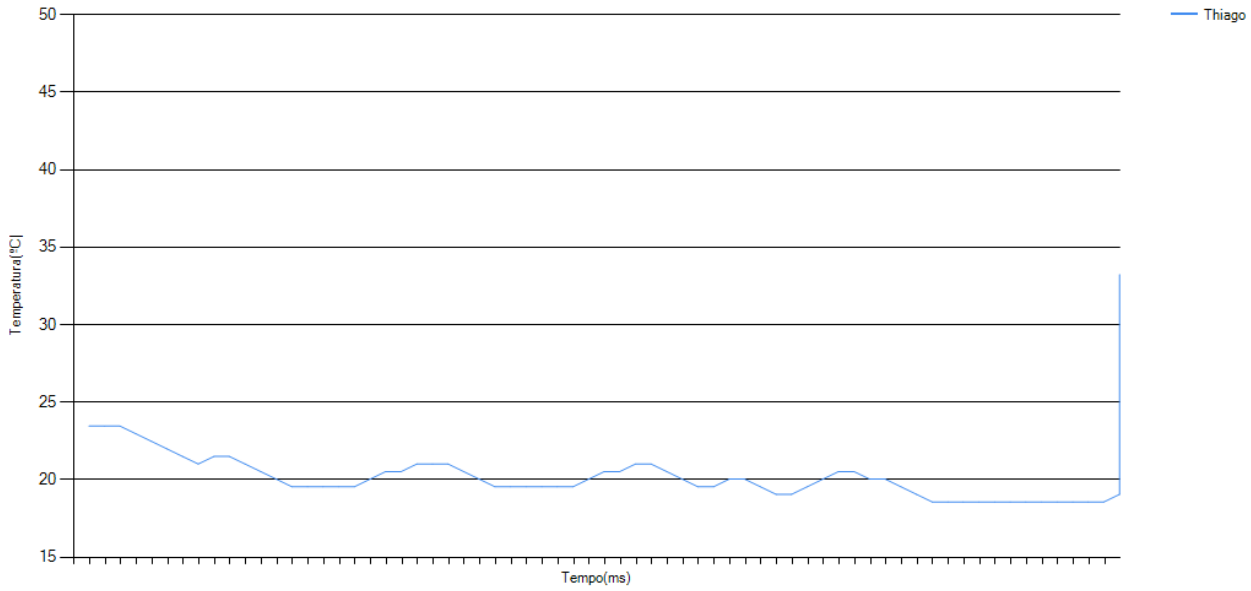

Figura 3.15 – Temperatura x Tempo

3.2. DESENVOLVIMENTO DE HARDWARE

` O protótipo proposto neste trabalho é formado de dispositivos eletrônicos capazes de capturar, processar e persistir as informações importantes para a análise de um evento, piloto ou equipamento. Os dados são capturados por dispositivos eletrônicos, como a temperatura, ou eletromecânicos, para o

tempo de volta e a aceleração. Estes dados necessitam processamento, para transformar tensão ou pulsos em informações úteis e então gravá-los em memória não latente, para posterior análise no *software* também desenvolvido.

As seções a seguir detalham as características dos componentes escolhidos e como é feita a interligação entre eles.

3.2.1. MICROCONTROLADOR

O microcontrolador escolhido para esse projeto é o modelo Atmel ATmega328. Baseado neste, o Arduíno é uma placa plataforma de desenvolvimento com *hardware* e *software* abertos. [ARDUINO, 2009]

O Arduíno contém 14 pinos digitais de entrada / saída, dos quais 6 podem ser utilizados como saída PWM - *Pulse-width modulation* e 6 entradas analógicas. Todas estas entradas podem ser facilmente configuradas e utilizadas para uso com sensores, atuadores e demais dispositivos eletrônicos, como cartões SD e *Displays* de Cristal Líquido. [ARDUINO, 2009]

A versão *Duemilanove* se conecta ao computador através de cabo padrão USB e contém o básico necessário para programar e utilizar o microcontrolador. Ela é Ilustrada na figura 3.16, com as características principais expostas na tabela 3.2.[ARDUINO, 2009]

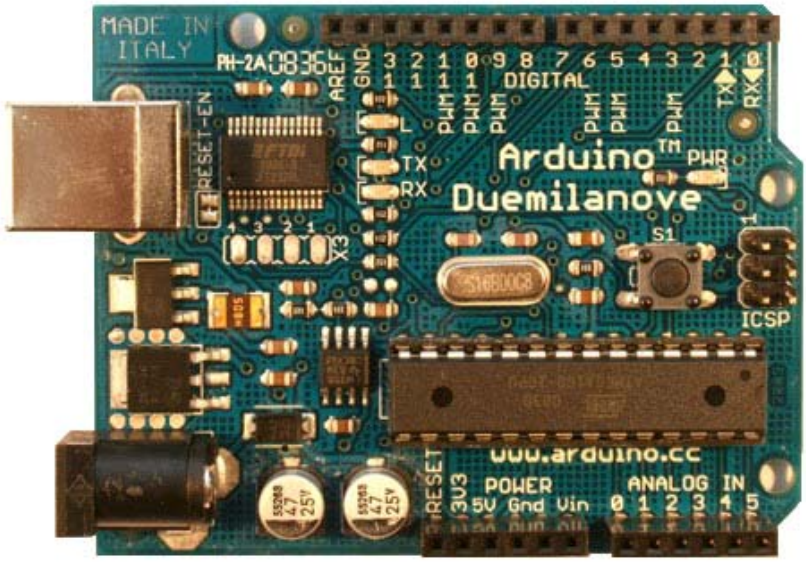

Figura 3.16 – Arduíno Duemilanove

[ARDUINO, 2009]

Tabela 3.1 – Características do Arduíno

Microcontrolador	ATmega328
Voltagem Operacional	5V
Voltagem de Alimentação	7-12V
Voltagem de Alimentação	6-20V
Pinos I/O Digitais	14
Pinos de Entrada Analógica	6
Corrente contínua por pino I/O	40mA
Corrente continua para pino 3.3V	50mA
Memória *flash*	32 KB (2KB usados para o bootloader)
SRAM	2 KB
EEPROM	1 KB
Velocidade de *clock*	16 MHz

[ARDUINO, 2009]

Por se tratar de uma plataforma completa de desenvolvimento de hardware, ele possui um ambiente de desenvolvimento integrado próprio (figura 3.17), composto por um escritor de código, compilador e um comunicador USB

serial, responsável pela gravação e comunicação entre a placa e o computador.

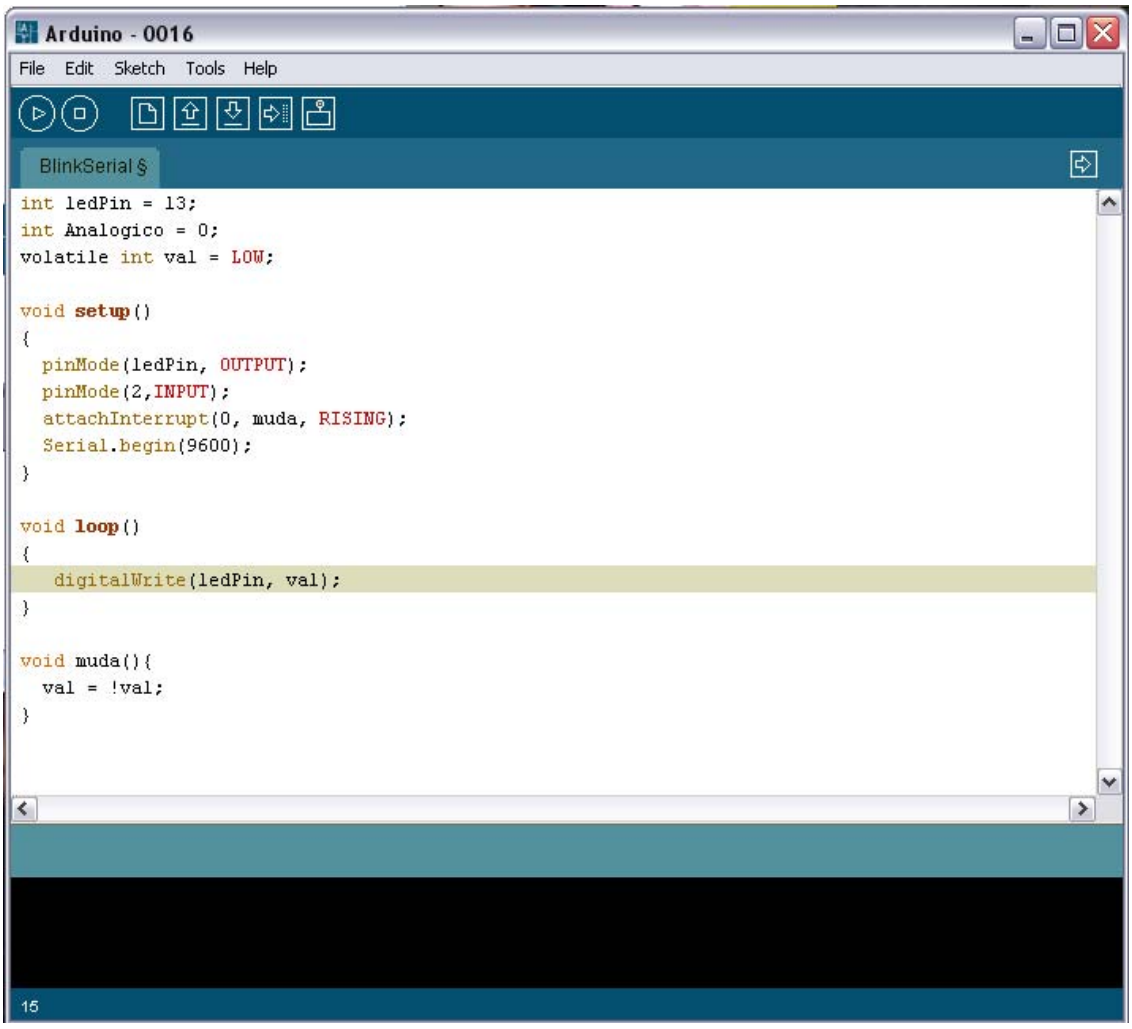

Figura 3.17 – Ambiente de Desenvolvimento Integrado

É a partir desta plataforma que o projeto é desenvolvido, permitindo atualizações e testes de código e *hardware* de forma rápida e robusta, fazendo com que o desenvolvedor tenha uma produtividade maior que em outras plataformas.

A programação é feita em uma linguagem baseada em C/C++, com uma estrutura básica, formada por dois métodos: *setup* e *loop*. A primeira é

executada apenas uma vez, logo quando o microcontrolador é ligado. Nesta função as formas de utilização de pinos são definidas, bem como protocolos e velocidades de comunicação. A segunda função, *loop*, é executada de forma cíclica após a execução da função *setup* até que o microcontrolador seja desligado ou colocado em *stand-by*. [ARDUINO, 2009]

Por se tratar de uma plataforma de código aberto, ela conta com amplo apoio da comunidade, que disponibiliza bibliotecas com funções diversas. Estas bibliotecas podem ser facilmente anexadas ao código e normalmente são projetadas em C++. [ARDUINO, 2009]

Neste projeto, são utilizadas bibliotecas responsáveis por dois itens vitais deste projeto: a temporização de voltas e o acesso ao cartão SD. As demais funcionalidades necessárias neste projeto são parte básica da plataforma Arduíno.

A biblioteca MsTimer2 permite a criação de um *timer*, através da instanciação de uma classe, que chama uma função implementada pelo usuário em intervalos constantes. A instanciação e configuração do *timer* são feitas através do comando “MsTimer2::set(Tempo,Funcao);”. O timer pode ser iniciado e parado através das funções *start() e stop()*, respectivamente. O cabeçalho desta biblioteca é apresentado no Apêndice B.

A biblioteca Fat16 permite a interface entre o Arduino e um cartão SD, já com o sistema de arquivos Fat 16 bits implementado. Com ela, é possível criar, apagar e modificar arquivos de texto, com nomes no padrão DOS, com 8 caracteres de nome e 3 caracteres de extensão, separados por um ponto. A biblioteca conta também com funções de análise de conteúdo, permitindo ao

desenvolvedor ter completo domínio dos arquivos presentes no cartão. O cabeçalho de funções desta biblioteca é mostrado no Apêndice C.

3.2.2. ACELERÔMETRO

Para medir as forças externas exercida no piloto e principalmente nos pneus do kart, durante curvas e frenagens, foi escolhido um acelerômetro MEMS, modelo DE-ACCM5G. [DIMENSION ENGINEERING, 2009]

Capaz de medir acelerações estáticas e dinâmicas, esse acelerômetro foi baseado no modelo ADXL320 da empresa *Analog Devices.* Pode medir intensidade de aceleração de até 5 vezes a da gravidade nos eixos “X” e “Y”. Tem como vantagem o baixo custo, R$40,00 e a alta sensibilidade, 312 mV/g quando aplicado 5V na tensão de entrada. Ilustrado na figura 3.18.

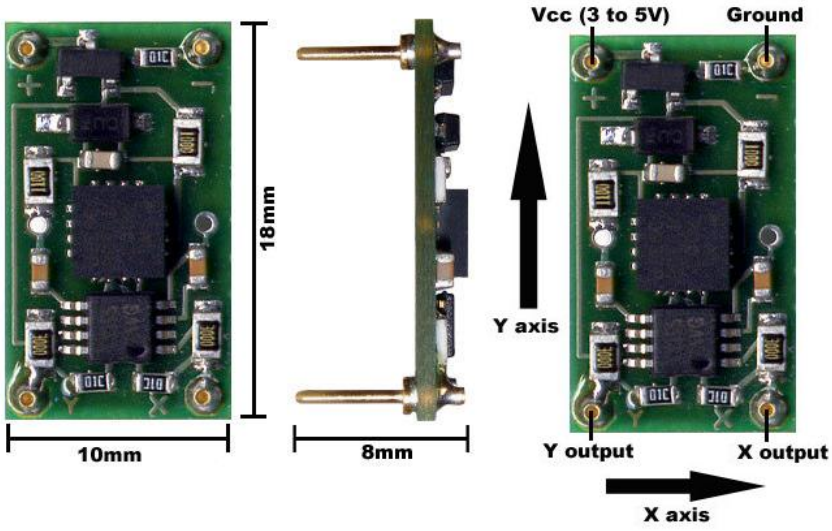

Figura 3.18 - Acelerômetro - DE-ACCM5G
[DIMENSION ENGINEERING, 2009]

A saída de dados é analógica de tensão. A tensão elétrica corresponde a ação de forças envolvidas nos 2 eixos. Esse acelerômetro trabalha com

tensão de entrada de 3 a 5 volts, variando assim sua saída. Quando 5 volts são aplicados, a sensibilidade é em torno das 312 mV/g, essa sensibilidade cai para 174 mV/g aos 3 volts de entrada. Por exemplo, para achar a tensão de saída quando uma aceleração correspondente a -0,5g em um acelerômetro alimentado por uma fonte de tensão regulada (Vcc) de 3V é dado pela equação: [DIMENSION ENGINEERING, 2009]

$$Vcc/2 + \left(F_{exc} \times Sensibilidade\left(\frac{mV}{g}\right)\right) = \quad (3.1)$$

$$= 3V/2 + \left(-0.5g \times 0.174 mV/g\right) = 1.413V \quad (3.2)$$

O tipo de aceleração medida pelo protótipo é a dinâmica, linear. É necessário então que o acelerômetro esteja paralelo ao solo, evitando assim qualquer tipo de interferência da aceleração estática da gravidade. Assim, quando o kart estiver parado, o acelerômetro deve estar aferindo 0g em ambos os eixos. Segundo o fabricante, os valores citados acima podem variar em até 4%. [DIMENSION ENGINEERING, 2009]

A ligação deste circuito é feita por apenas 4 portas, das quais 2 são utilizadas para alimentação e 2 são as saídas, analógicas, para os eixos X e Y, respectivamente.

A figura 3.19 apresenta o diagrama esquemático de ligação entre o acelerômetro e o Arduíno. Percebe-se que a alimentação do acelerômetro é feita diretamente pelo Arduíno e as saídas, dos eixos X e Y, são ligadas diretamente às entradas analógicas 0 e 1, respectivamente.

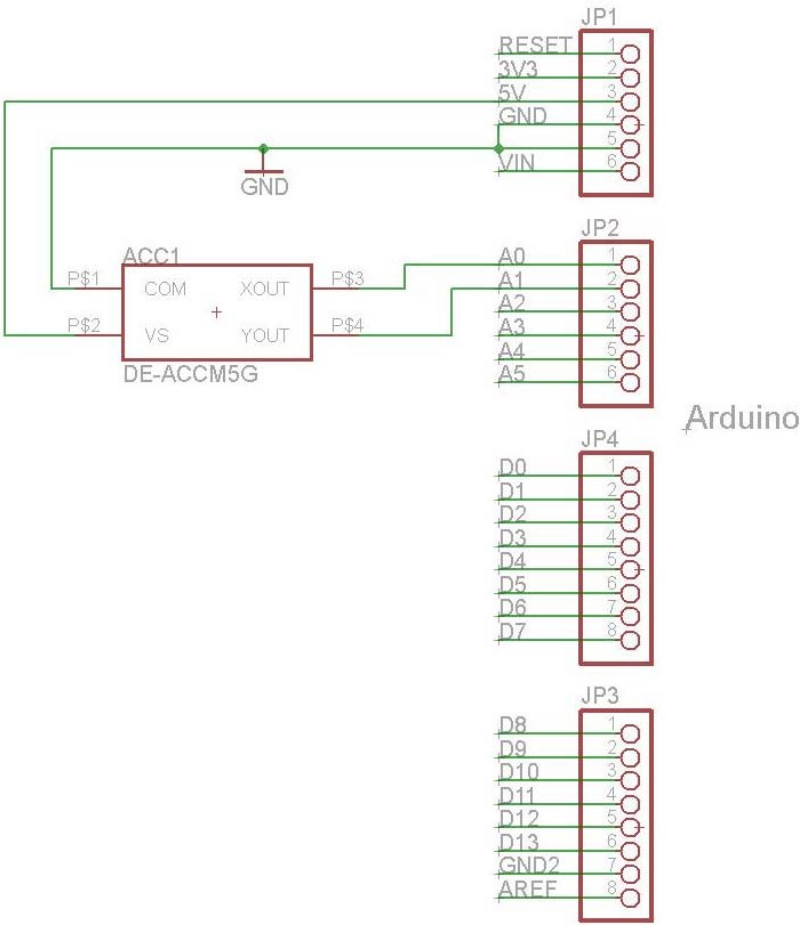

Figura 3.19 – Diagrama esquemático de ligação do Acelerômetro

3.2.3. SENSOR DE TEMPERATURA – LM35

Fabricado pela *National Semiconductor*, o sensor LM35 apresenta uma saída de tensão linear relativa à temperatura em que ele se encontra. Sua saída, quando alimentado entre 4 e 20 V é de 10 mV por cada Grau Celsius. Trabalha com valores de temperatura entre -55°C e 150°C. [NATIONAL SEMICONDUCTOR, 2009]

O LM35 é comercializado em diversos tipos de encapsulamento, possibilitando sua adoção em projetos com necessidades diferentes. O encapsulamento escolhido neste projeto foi o TO-92 (apresentado na figura

3.20), por ser amplamente disponível e barato (aproximadamente R$5,00 a unidade). O encapsulamento TO-92 é o mesmo utilizado em uma série de outros componentes eletrônicos, como transistores TJB.

Figura 3.20 - LM35 - TO-92
[NATIONAL SEMICONDUCTOR, 2009]

A escolha desse componente para o protótipo deu-se em virtude do seu preço, sua comunicação simples com o microcontrolador (uma saída analógica) e, principalmente, por sua faixa de operação de temperatura ser compatível com a temperatura a ser medida.

O LM35 é alimentado através do próprio Arduíno, através da saída de tensão de 5V e do terra. A informação de temperatura medida é fornecida ao microcontrolador de forma analógica, através da porta analógica 2, conforme visto na figura 3.21.

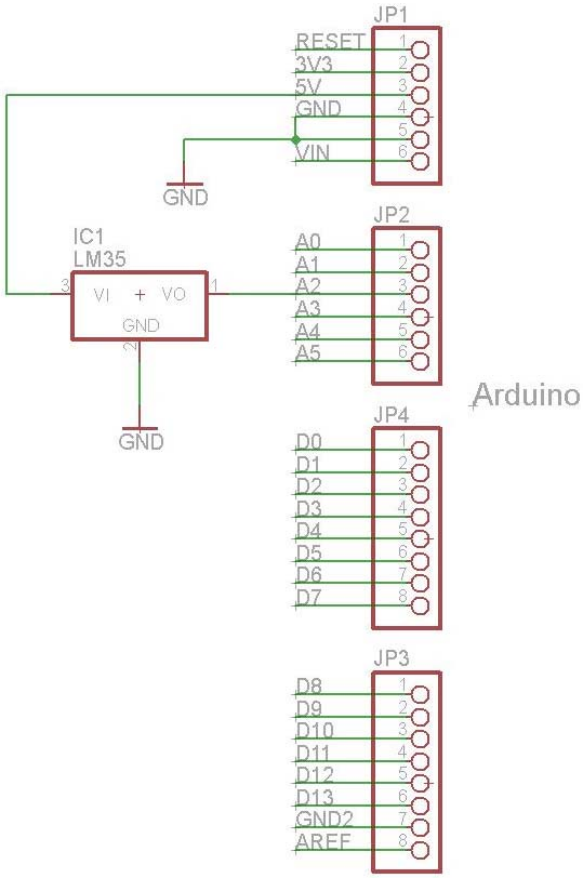

Figura 3.21 – Diagrama esquemático de conexão do LM35 ao Arduíno

3.2.4. *REED SWITCH*

É um componente de grande utilidade, pode ser usado em diversas aplicações mecânicas, robóticas e de automação. *Reed Switch* ou interruptor de lâminas consiste em um bulbo de vidro com um gás inerte que ao entrar em contado com a ação de um campo magnético, unem-se fechando o contato elétrico. A figura 3.22 ilustra a versão mais simples de um *reed switch*. [BRAGA, 2007]

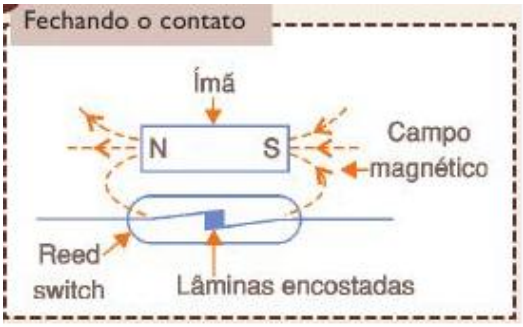

Figura 3.22 – *Reed Switch*

[BRAGA, 2007]

É um dispositivo de baixa corrente, e muitas vezes controla sistemas em que a corrente não vai muito além dos 200 mA. Tem a velocidade de resposta alta, ou seja, ao passar rapidamente pelo campo magnético fecha e abre contato quase que instantaneamente, sendo apropriado para aplicação neste protótipo.

Sua ligação é feita com um capacitor e um resistor, além de um *Schmitt Trigger*, para evitar ruídos. Por fim, é ligado a porta digital 2 do Arduíno, ilustrado na figura 3.23.

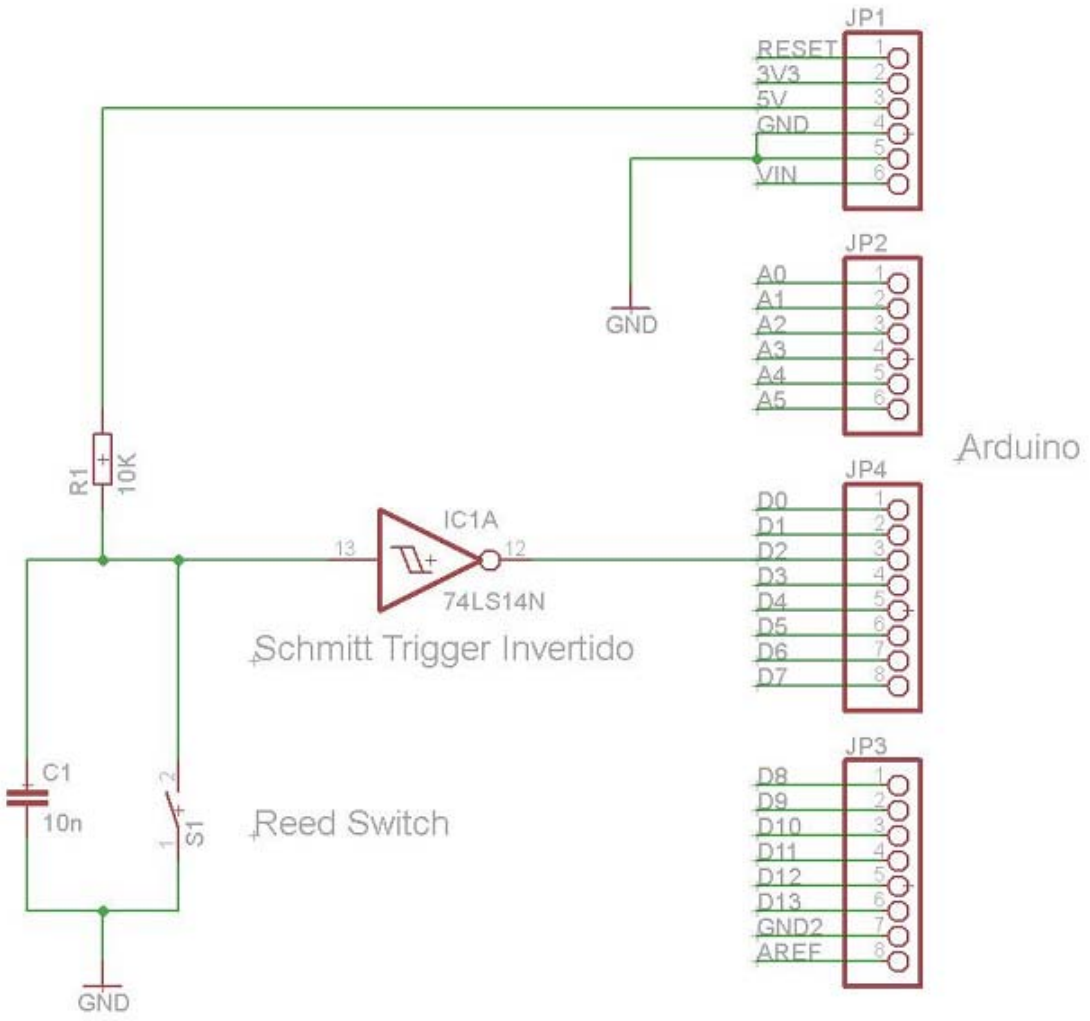

Figura 3.23 – Diagrama esquemático de conexão do *Reed Switch* ao Arduíno

3.2.5. CARTAO SD

Responsável pelo armazenamento dos dados de forma não volátil, possui duas formas básicas de comunicação: um protocolo próprio, que faz uso de todos os pinos permitindo o acesso em velocidade máxima, ou através do barramento SPI (Serial Peripheral Interface Bus), que permite o uso de apenas 4 pinos para a comunicação, que deve ser feita em nível lógico de 3,3V. [SD ASSOCIATION, 2009]

O segundo modo de acesso é utilizado, por utilizar menos pinos, e ser baseado em um protocolo já implementado no Arduino. As conexões necessárias são apresentadas na figura 3.24.

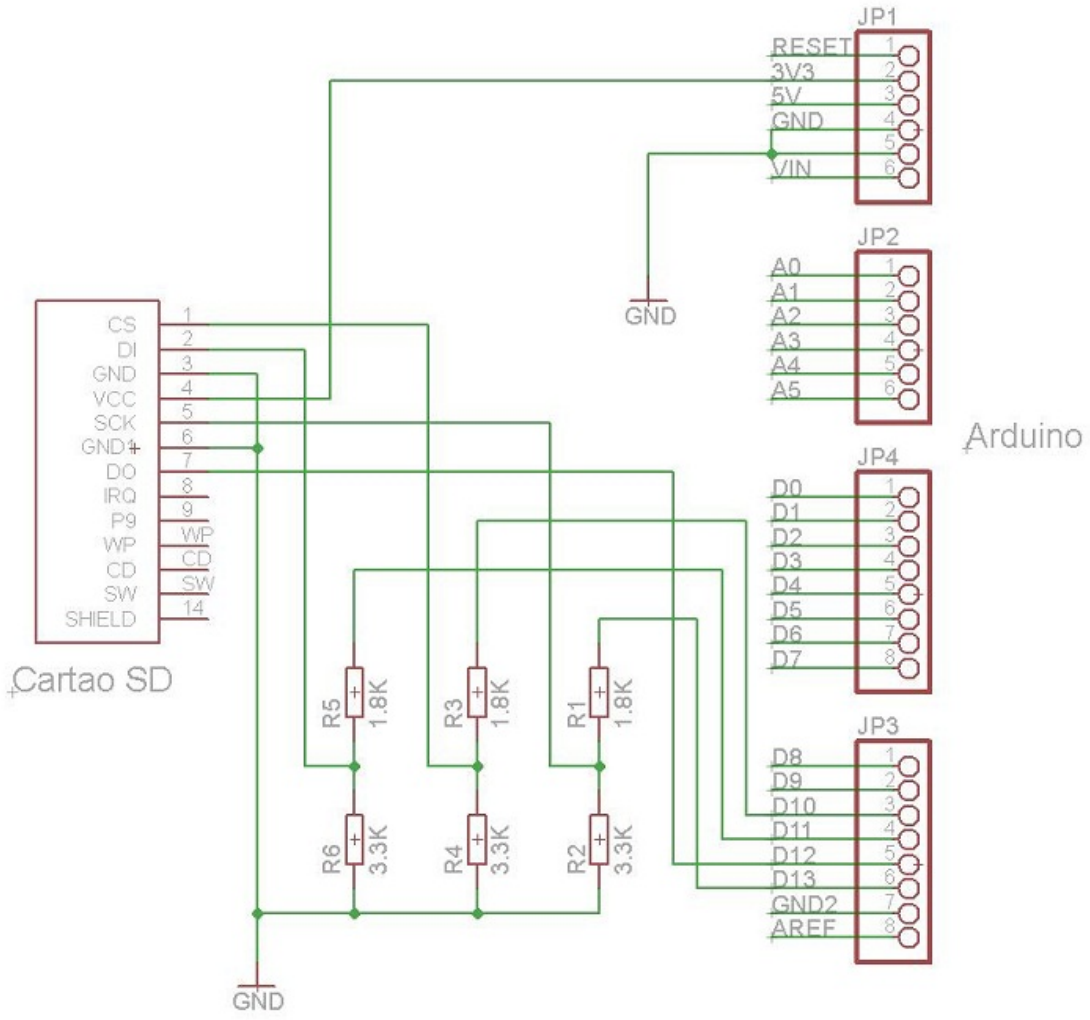

Figura 3.24 – Diagrama esquemático do Cartão SD no Arduíno.

A implementação do software é feita através de uma biblioteca, que acessa o cartão SD e gerencia o sistema de arquivos, em FAT16, permitindo que arquivos sejam criados, modificados e apagados em tempo de execução, figura 3.25. [SD ASSOCIATION, 2009]

Figura 3.25 – Diagrama de Bloco – Fat16

4 – TESTES E RESULTADOS OBTIDOS

A análise dos resultados está divida em três etapas. A primeira foi a realização dos testes com os componentes eletrônicos, funcionamento dos componentes ligados separadamente na protoboard. A segunda, o funcionamento isolado da parte de *software off-line*, inclusão e exclusão de dados e por último o funcionamento da parte de integração entre a parte eletrônica, a parte *off-line* e a importação dos dados.

Os dados do protótipo embarcado foram testados em um carro de passeio, apenas para fazer o teste do acelerômetro. O teste com o sensor de passagem foi executado no kart. O teste com sensor de temperatura foi feito com a ajuda de uma fonte geradora de calor externa.

4.1. SISTEMA *OFF-LINE*

Desenvolvido de forma que o usuário utilize intuitivamente, o sistema *off-line* além de manipular dados advindos do protótipo embarcado, inclui dados passados pelo usuário. Para testar o correto funcionamento das telas foram feitos teste de inclusão e exclusão de piloto, pista, equipamento e evento.

Ao iniciar o sistema, na tela de controle de funcionalidade, foi feita a escolha “Cadastro”, onde é escolhida a opção “Piloto”, ilustrado na figura 4.1.

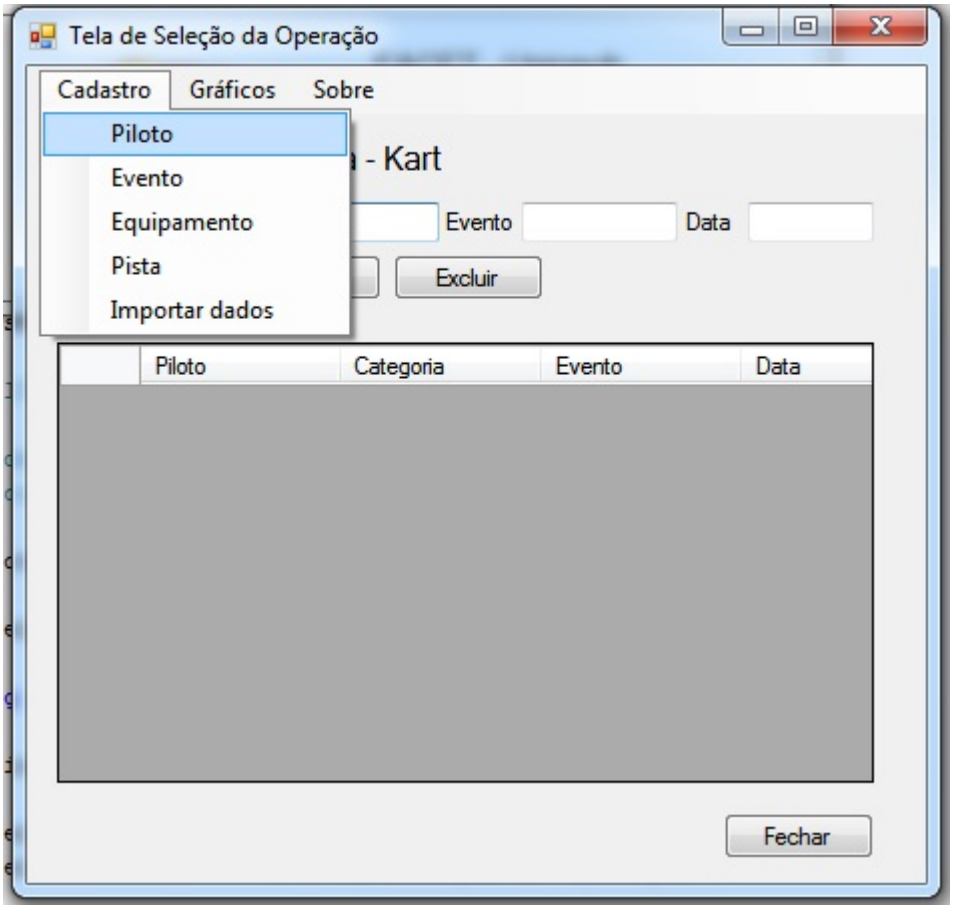

Figura 4.1 – Cadastrar Piloto

Ao selecionar a opção de cadastro de piloto, é aberta a tela ilustrada na figura 4.2, onde é feita a passagem dos parâmetros que serão cadastrados na tabela tblPiloto no banco de dados.

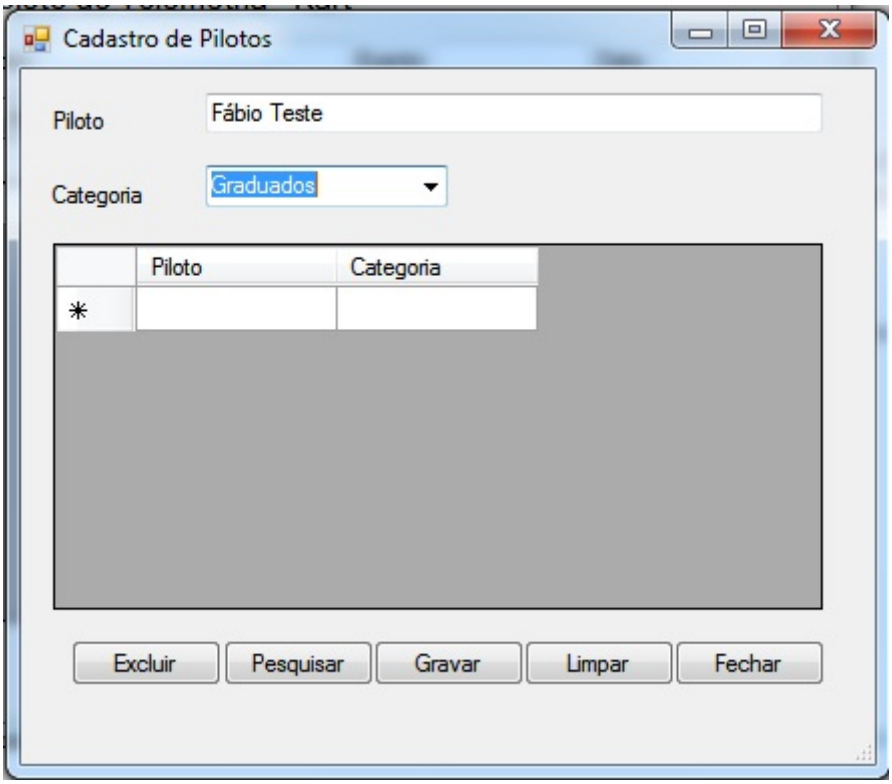

Figura 4.2 – Incluir Piloto

Após preencher os parâmetros e pressionar o botão “Gravar”, é exibido uma mensagem de conclusão do processo. Para certificar que os dados foram gravados de forma correta, abre-se o banco de dados e verifica-se que os dados foram inseridos com sucesso, ilustrado na figura 4.3.

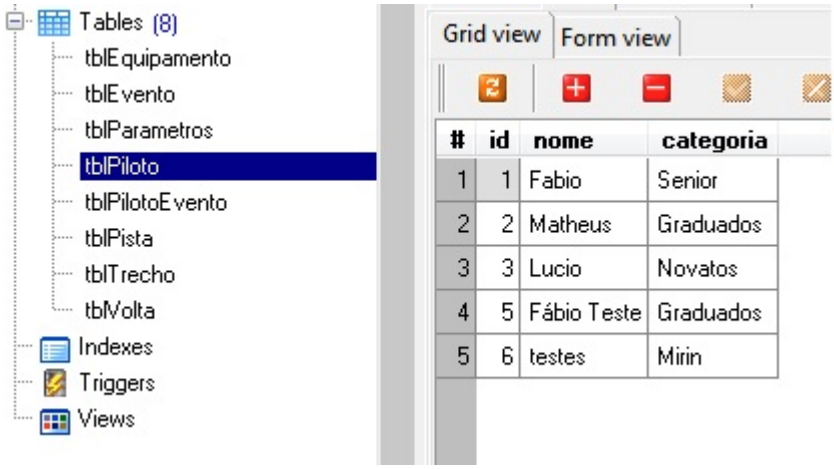

Figura 4.3 – Tabela de Piloto

A exclusão é feita na mesma tela de inclusão, porém é necessário que seja feita uma pesquisa por nome onde é preenchido a tabela, seleciona o registro que deseja excluir e pressiona o botão “Excluir”, ilustrado na figura 4.4.

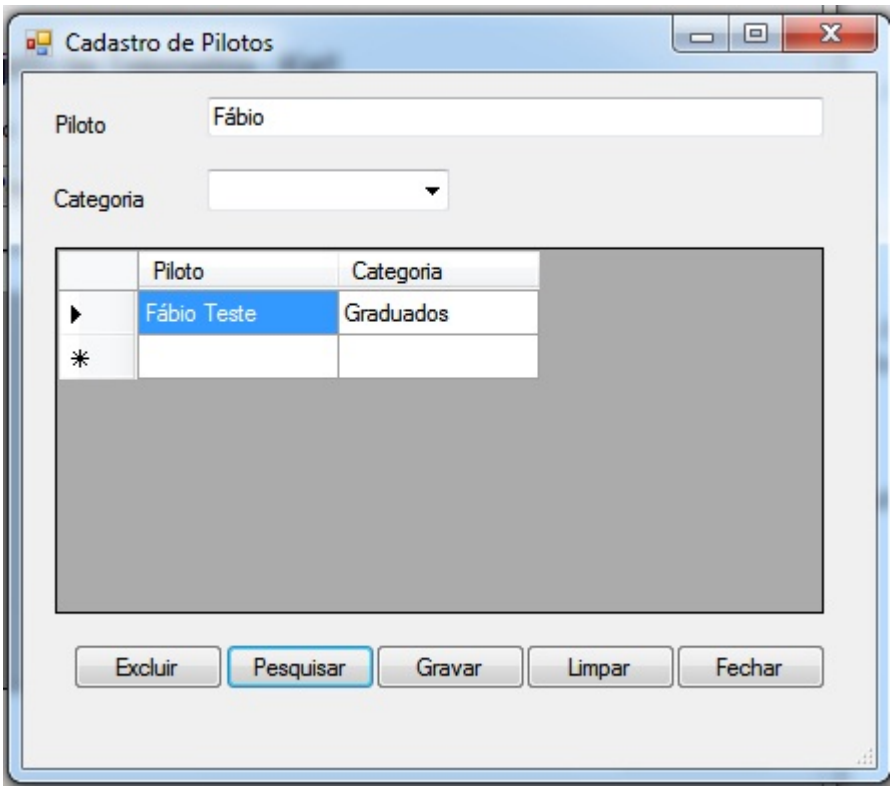

Figura 4.4 – Excluir Piloto

O processo de inclusão e exclusão de eventos, equipamento e pista, seguem a mesma lógica do processo feito com piloto, porém utilizando as tabelas tblEvento, tblEquipamento e tblPista, respectivamente.

O teste de importação dos dados foi feito após a execução dos testes do sistema embarcado e da inclusão de dados no sistema *off-line.* Este teste visou a correta execução de uma rotina que lê um arquivo texto gerado pelo protótipo embarcado, geram o arquivo texto com os parâmetros e persistem os dados

mensurados na pista no banco de dados para que possam gerar gráficos ou até mesmo manter um histórico de informações.

Acessando o mesmo *menu* de cadastro, foi selecionado a opção: “Importar dados”. Na tela de importação de dados, ilustrado na figura 4.5, foi feita a vinculação de um piloto, a um evento, previamente inseridos no banco através do sistema *off-line,* e selecionado o botão importar.

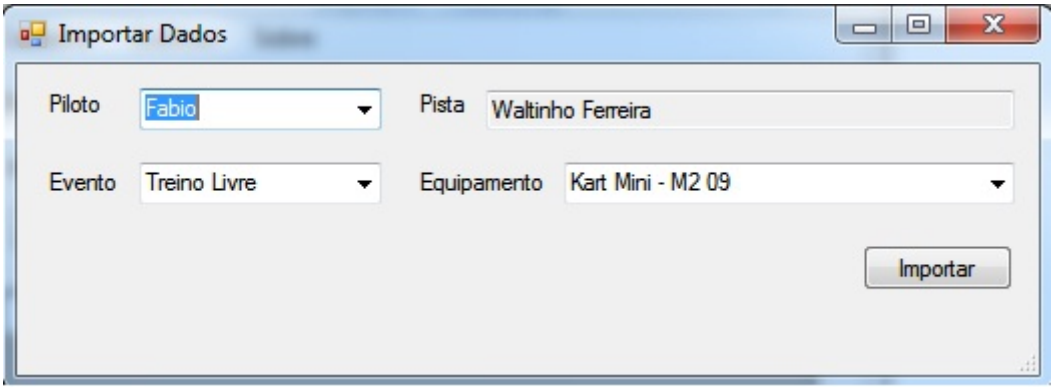

Figura 4.5 – Importar Dados

A execução da rotina foi verificada a partir da inclusão dos dados lidos no arquivo texto em três tabelas do banco de dados: tblVolta, tblTrecho, tblParametros.

Com os dados da importação persistidos no banco de dados, foram feitos os testes com os gráficos. O acesso a essa funcionalidade é feito na tela de controle de funcionalidade, seleciona a opção “Gráficos”.

A tela de seleção dos tipos dos gráficos é aberta, onde foi feita a seleção de dois pilotos e pressionado o botão aceleração, ilustrado na figura 4.6.

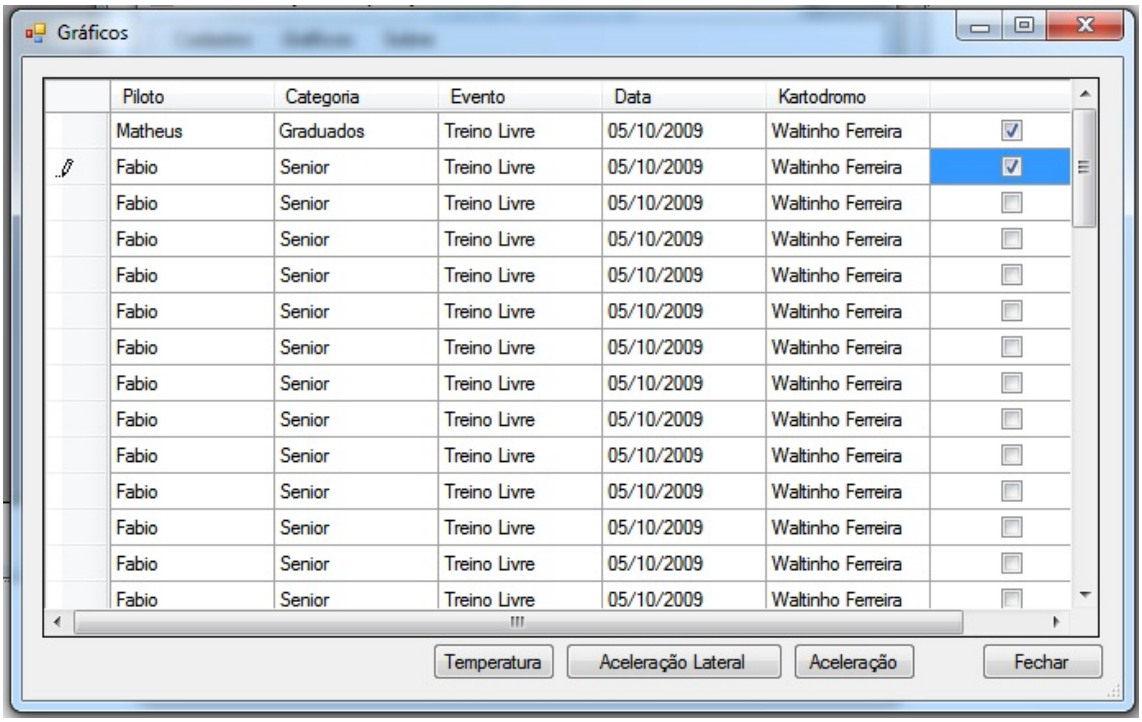

Figura 4.6 – Seleção de gráfico

Por fim é aberta a tela com os gráficos em forma de linhas onde é feito o comparativo das acelerações obtidas pelos pilotos em determinado momento de tempo, ilustrado na figura 4.7.

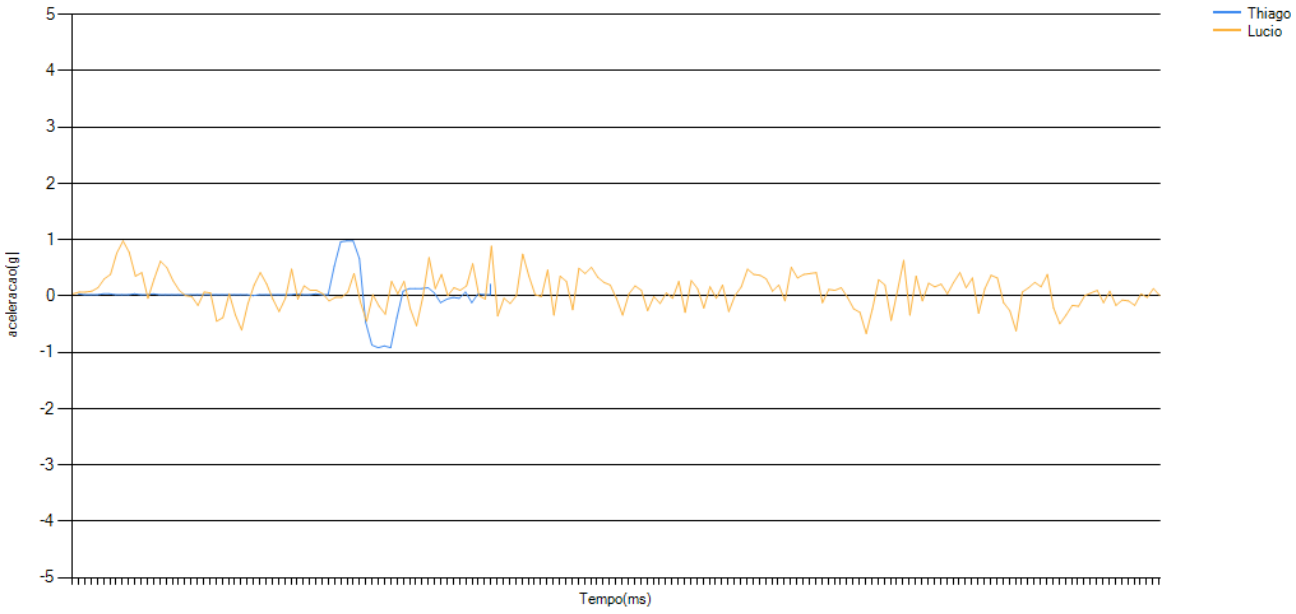

Figura 4.7 – Tela de Gráficos

O processo de confecção dos gráficos de temperatura e aceleração lateral seguem a mesma lógica do processo feito com o gráfico de aceleração, porém o eixo das ordenadas “Y” utiliza os valores em grau Celsius, no gráfico de temperatura.

4.2. SISTEMA EMBARCADO

A implementação em hardware foi feita em duas partes, a primeira, na *protoboard* (uma matriz de contatos para prototipação) e a segunda em uma placa de circuito impresso feita de forma artesanal.

A montagem na *protoboard*, que pode ser vista na figura 4.8, foi iniciada pela construção de cabos para a conexão entre o Arduino e a matriz de contatos. Após a conexão ter sido executada, os diversos módulos foram conectados através de fios.

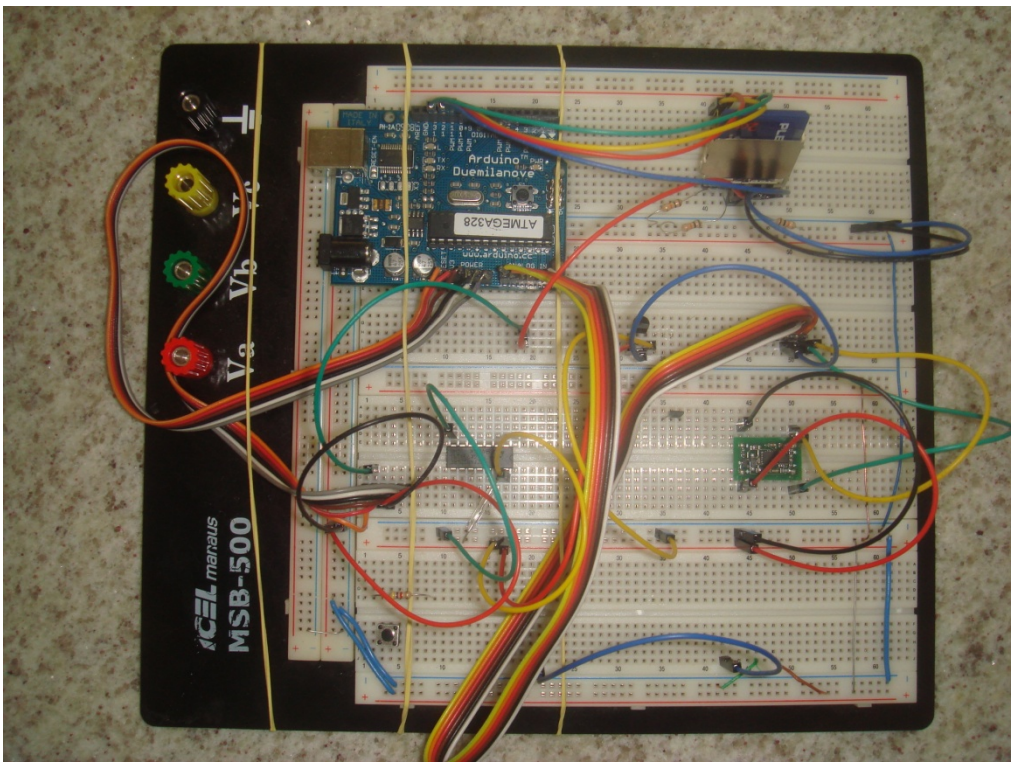

Figura 4.8 – Circuito na Protoboard

A implementação em matriz de contatos é feita para que componentes sejam ligados rapidamente, possibilitando correções nas conexões sem a necessidade de soldagem.

Após a verificação de funcionamento de todos os módulos, foi montado um circuito que permite a prototipação com soldagem, para ser utilizada nos testes embarcados, diminuindo o espaço necessário e as chances de problemas como mau contato.

A placa de circuito impresso é fabricada utilizando a técnica de transferência de tonner. Pintam-se trilhas na superfície de cobre e mergulha-se a placa em uma solução de percloreto de ferro, que corrói apenas os pedaços não pintados da superfície.

A placa foi desenhada no *software Eagle*, conforme visto na figura 4.9, que permite que o usuário desenhe esquemáticos e placas de circuito impresso em padrões internacionais.

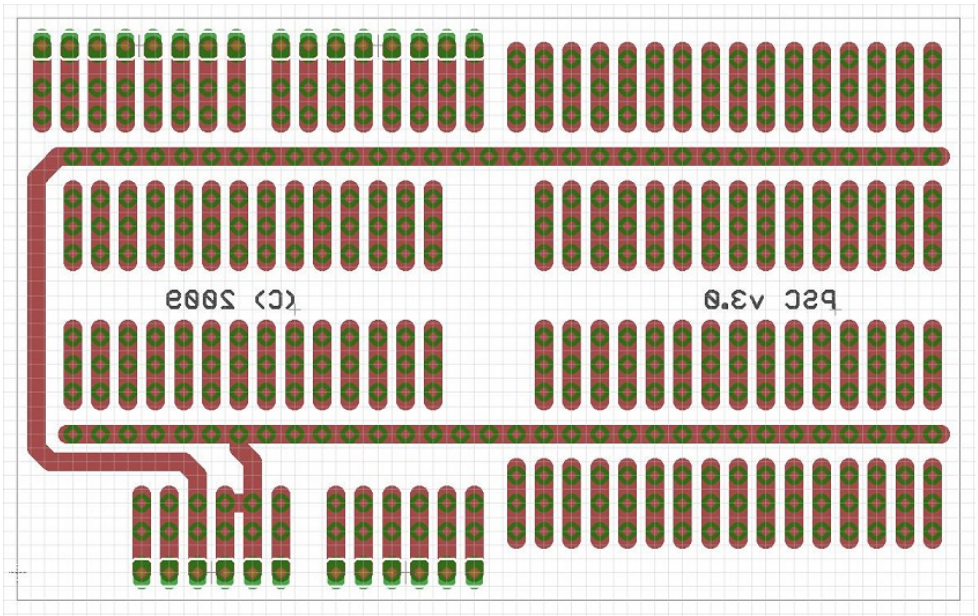

Figura 4.9 – *Protoshield* da Placa Arduíno

Para a fabricação da placa, o circuito foi transformado em preto e branco e impresso em papel tipo Glossy em uma impressora a laser.

Para a transferência do circuito para uma placa, foi utilizada uma placa de fenolite com uma camada de cobre que deve ser limpa utilizando escova de aço, para retirar segmentos de oxidação da placa, e álcool 96%, para remoção de gordura e sujeiras que dificultem à transferência do tonner a placa.

Após isto, a folha contendo o circuito é fixada na placa com o uso de fita adesiva e se utiliza um ferro de passar roupa comum para esquentar o circuito e transferir o tonner para a camada de cobre.

Com a transferência feita, o papel é removido e a placa é mergulhada em uma solução aquosa de percloreto de ferro, que ira remover o cobre onde não há tonner, resultando uma placa formada por uma série de trilhas que permitem a conexão do Arduino a diversos componentes.

4.2.1. ACELERÔMETRO

Desde os testes iniciais, na matriz de contato, o acelerômetro se comportou de maneira adequada, apresentando valores de tensão condizentes com a aceleração aplicada a cada um dos eixos. Porém, existe uma constante, de 0,2 G, no eixo Y, que é causada por um defeito de fabricação da placa, onde o circuito integrado responsável pelas medições apresenta uma leve inclinação.

Este problema é corrigindo somando-se 0,2 G das medições efetuadas o problema foi facilmente resolvido via *software,* ilustrado na figura 4.10, não houve necessidade de troca ou compra de outra unidade.

```
void loop()
{


 val = analogRead(vout);    // lê valor do sensor
 val = map(val, 0, 1023, 0, 500); //Uso da função map para correção na porta AD
 Serial.print("Temperatura : ");
 Serial.print(val);
 Serial.println(" Graus Celsius ");

  // print the sensor values:
  valX = analogRead(xPin);
  valGX = ((valX-512)*10)/64;
  Serial.print("X:");
  Serial.print(valGX);
  // print a tab between values:
  Serial.print("\t");
  valY = analogRead(yPin);
  valGY = ((valY-512)*10)/64;
  valGY = valGY +2; //Correção do Acelerometro
  Serial.print("Y:");
  Serial.print(valGY);
  Serial.println();
  // delay before next reading:
  delay(1000);
}
```

Figura 4.10 – Correção do Acelerômetro no *Software*

A execução do teste de campo com acelerômetro foi feito com o protótipo preso ao console de um carro de passeio, figura 4.11. Foram feitos teste de aceleração partindo com o veículo em repouso, frenagem e curvas a esquerda e direita. Os dados foram gravados direto no *notebook* via USB.

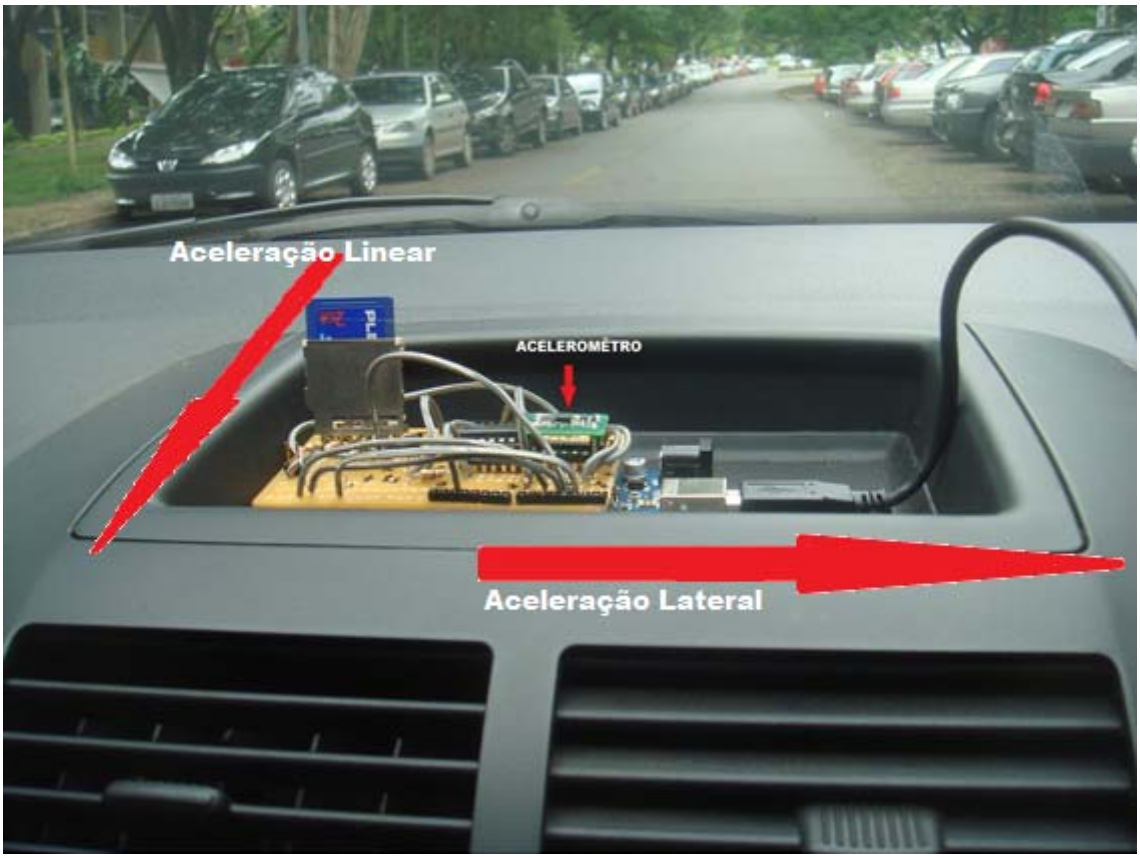

Figura 4.11 – Protótipo preso ao console do veículo de passeio

Os valores máximos verificados no teste com o acelerômetro estão na tabela 4.1, valores da aceleração linear e lateral.

Tabela 4.1 – Valores máximos – Teste Acelerômetro

Aceleração	1g
Frenagem	0,7g
Curva a direita	1g
Curva a esquerda	1,15g

4.2.2. SENSOR DE TEMPERATURA

O CI responsável pela medição de temperatura é formado por apenas 3 terminais , acarretando uma conexão extremamente simples. Alimentado por uma fonte de 5V, ele apresenta tensões de saída variando entre 0 e 1,5 V, para temperaturas entre 0 ºC e 150 ºC.

O teste com esse componente foi feito a temperatura ambiente e com seu aquecimento feito por um secador de cabelo. Inicialmente o sensor encontrava-se com a temperatura em 19ºC e foi aquecido até 45ºC, porém houve um atraso na transferência de calor do componente. Ao parar de fornecer calor, o sensor ainda continuou com o processo de aquecimento. Esse fenômeno se dá devido à constante térmica do tempo, onde é necessário um tempo para que a temperatura se equilibre em todo seu corpo. Segundo o datasheet desse componente esse tempo é de aproximadamente 16 segundos. [NATIONAL SEMICONDUCTOR, 2009]

4.2.3. REED SWITCH

Visando testar o seu funcionamento em alta velocidade, um *reed switch* foi instalado no assoalho do kart ligado apenas a uma bateria de 9 volts, figura 4.12, e um *led*, ligado no volante. A cada passagem pela faixa magnética instalada na pista, o *reed switch* fechava a ligação e o *led* piscava rapidamente mais de uma vez.

Figura 4.12 – Teste de campo do *Reed Switch*

Com base neste comportamento, percebeu-se que o *reed switch* se comporta como um botão normal, apresentando *bouncing*, um efeito comum que basicamente consiste em uma seqüência de liga-desliga antes da estabilização do dispositivo, ocasionado pelo acoplamento físico das duas placas de metal, gerando um efeito de várias ligações seguidas entre as lâminas.

O efeito de *bouncing* pode ser solucionado via *hardware* ou via *software*. Como o programa efetuado deve ser de alto desempenho, permitindo que os parâmetros e os tempos de volta sejam capturados da forma mais precisa possível, escolheu-se fazer o *debouncing* via *hardware*.

Para solucionar este problema, um capacitor de 10nF foi inserido em paralelo e um resistor de 10k Ohms em série, funcionando como carregador do capacitor. Este circuito tem comportamento de um filtro passa baixas, filtrando as oscilações do *switch*. Apesar do efeito *bouncing* ter diminuído, ainda existia,

possibilitando que uma passagem pela fita de imãs gerasse até 4 ligações do *led*.

Foi necessária a adição de um segundo estágio no circuito tratador do sinal do *reed switch*, a inserção de um gatilho de *Schimidt*, que provê um laço de histerese ou efeito memória e é amplamente utilizado neste tipo de aplicação. O gatilho escolhido foi o 7414, de lógica TTL, compatível com a utilizada no Arduino.

Com a adição de mais este elemento, o diagrama esquemático do circuito final é apresentado na figura 4.13. Com isto, os problemas de múltiplos acionamentos foram encerrados e o módulo foi incorporado ao projeto.

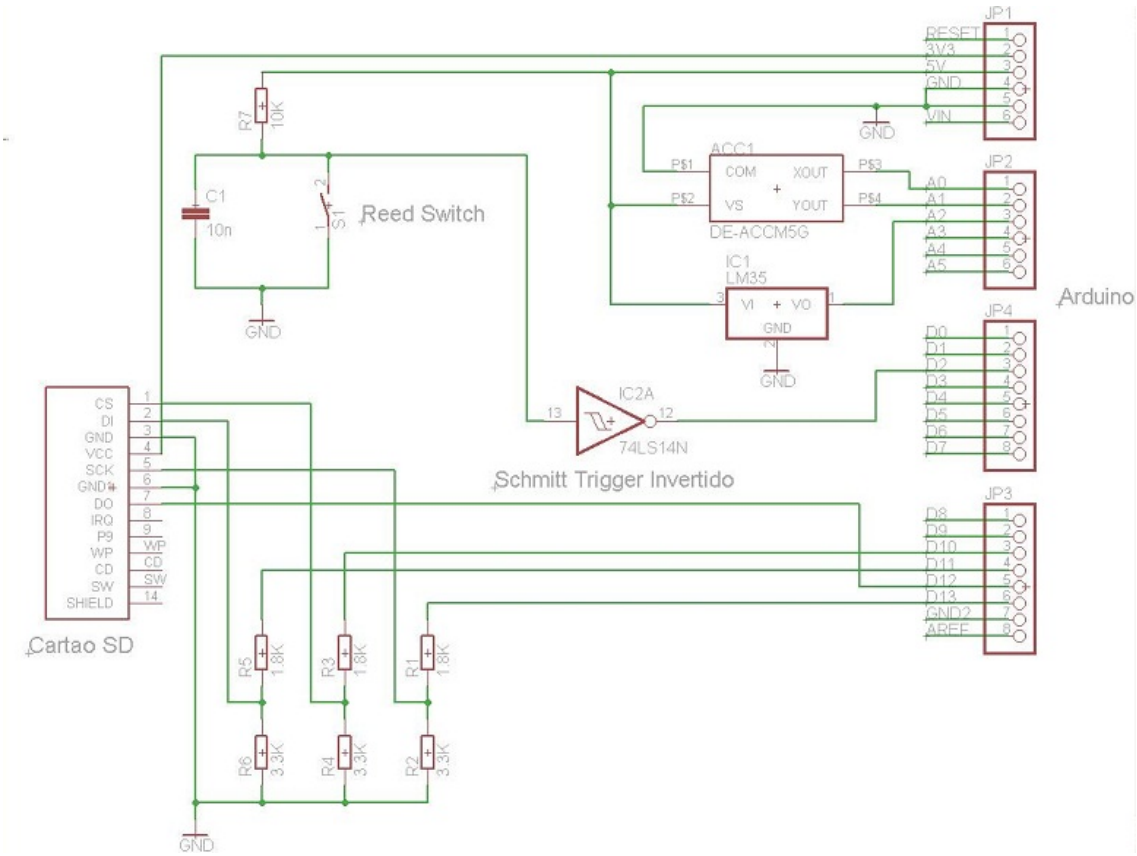

Figura 4.13 – Diagrama Circuito Final

4.2.4. GRAVAÇÃO DOS DADOS

O cartão SD funciona apenas com tensão de 3,3V, enquanto o Arduino funciona em 5V. A alimentação do cartão SD é feita através da saída de 3.3V presente no arduino, enquanto a troca de dados é feita utilizando um divisor de tensão, segundo a fórmula

$$V_{0=} \frac{R_2}{(R_1 + R_2)V}$$

Utilizando resistores de 10k e 4,7k, a tensão de 5V é transformada em 3,4V, dentro dos limites de operação do cartão SD. O pino de retorno de dados (DO) é acoplado diretamente ao Arduíno.

Porém, desde o início da implementação do cartão SD na matriz de contatos, ele tem apresentado comportamento de mau contato, funcionando apenas quando levemente pressionado no protoboard. Já na placa de circuito impresso, esse comportamento se agravou, impossibilitando o funcionamento do módulo.

Com a ajuda de um multímetro, as tensões e continuidade do circuito foram testadas. O mau contato aparenta acontecer dentro do soquete, por defeito de fabricação.

A solução adotada para a gravação dos dados adquiridos pelos sensores foi a implementação de um programa em C#, apresentado Apêndice “E”, que grava o arquivo texto com a transmissão via porta serial pela USB ligado ao *notebook*.

O programa faz comunicação com a porta serial COM4 e grava o arquivo gerado pelo microcontrolador na área de trabalho do sistema

operacional. O nome do arquivo e o caminho de gravação é pré-definido no código fonte. Após a inicialização do programa, é necessário o acionamento do *reed switch* para que o programa grave o arquivo e mostre na tela os dados que estão sendo gravados no arquivo texto.

O circuito final, completo, com todos os componentes soldados às placas de circuito impresso é apresentado na figura 4.14. No lado esquerdo encontra-se a placa com os sensores de aceleração e temperatura, além do cartão SD e do circuito necessário ao correto funcionamento do *reed swich*. Do lado direito é apresentado a placa de circuito impresso contendo o *reed switch*, que é conectada a placa principal por um cabo.

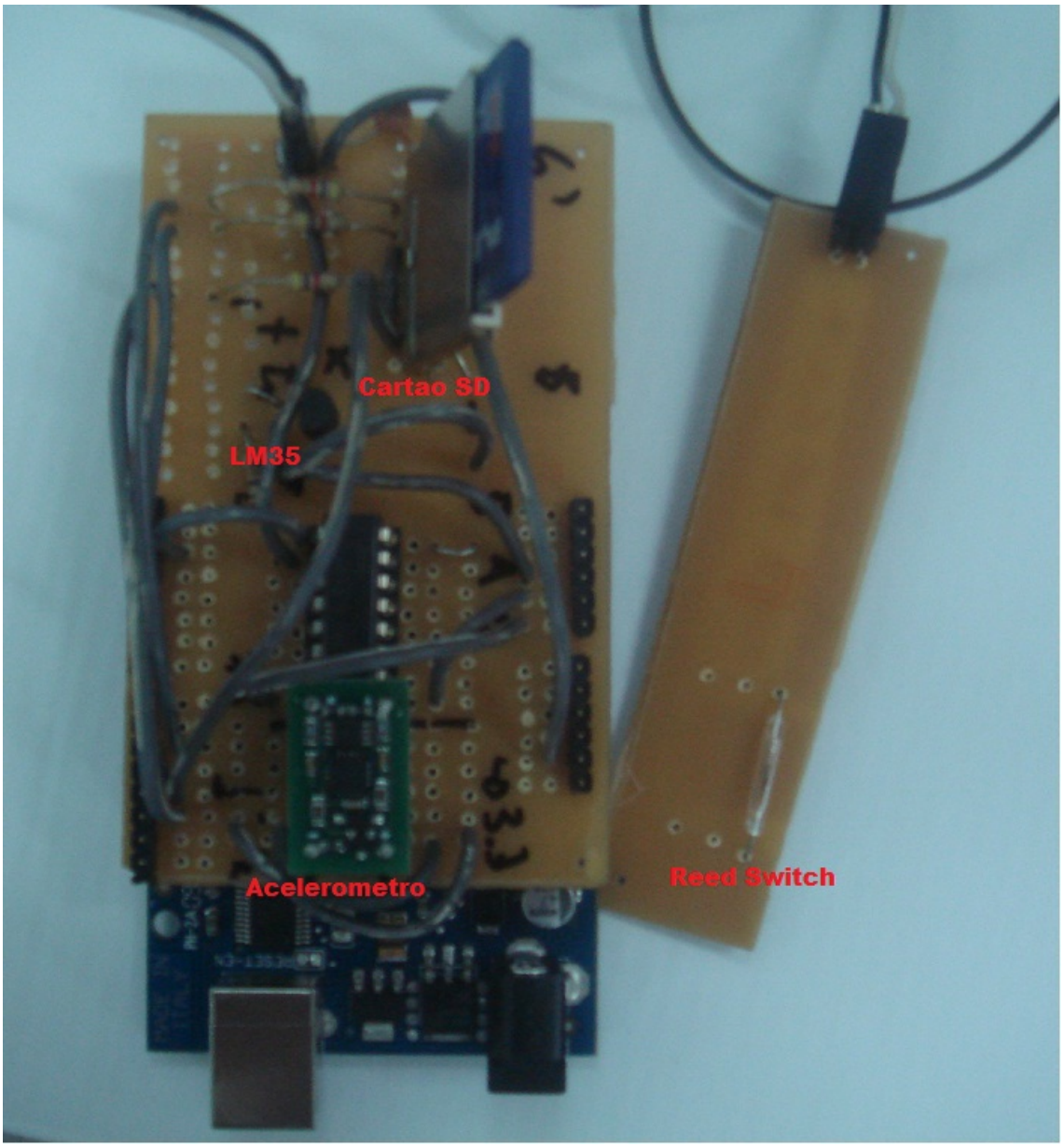

Figura 4.14 – Protótipo

5 – CONCLUSÃO

Este projeto teve como finalidade o desenvolvimento de um protótipo de computador de bordo para kart, onde dados relevantes foram gravados no banco de dados a partir do arquivo texto gerado pelo protótipo e gravado no computador via conexão USB. A aferição dos parâmetros de tempo, aceleração e temperatura obtiveram bons resultados, os dados foram importados de forma correta, e os gráficos foram gerados. Pode-se então comprovar que os sensores escolhidos atingiram os objetivos propostos. O protótipo desenvolvido ainda não é capaz de ser embarcado em um kart devido à grande trepidação que o mesmo proporciona, podendo acarretar em mau contato e pelo não funcionamento do cartão SD.

Durante o desenvolvimento do projeto foram encontradas várias dificuldades, atrasando o cronograma estabelecido. Primeiramente o microcontrolador escolhido anteriormente, PIC18F452, era complexo na gravação de programas, gastando muito tempo na curva de aprendizagem para cumprir o objetivo proposto. Como solução foi adotada uma placa de teste *open source*. Outra dificuldade foi a compra dos componentes eletrônicos, muitas vezes feita fora de Brasília devido a pouca variedade de loja no ramo. Problemas de mau contato na implementação do cartão SD tomara bastante tempo na construção do protótipo. Na parte de *software,* a maior dificuldade foi à implementação dos gráficos, pois a biblioteca escolhida não apresenta muito material de consulta.

O desenvolvimento utilizando a placa Arduino foi fácil, possibilitando diversas implementações, correções de código e testes em um curto intervalo de tempo, acarretando em alta produtividade.

O código final, compilado, tem 9 kB de tamanho, ocupando pouco menos de um terço do espaço disponível pelo Arduino (32 kB). O processador utilizado apresentou um excelente desempenho, não acarretando atrasos maiores que 2 milissegundos nas capturas dos dados dos sensores e atrasos ainda menores na captura de tempos parciais.

Como continuidade desse projeto podem ser implementadas melhorias na forma de transmissão dos dados, com a implementação de transmissão *on-line,* melhorias no software de aquisição dos dados, ampliando os estudos dos dados nos gráficos. A adoção de novos sensores, sensor de RPM fazendo a contagem do pulso elétrico que chega até a vela de ignição, sensor de velocidade utilizando a mesma técnica que é feita hoje nos computadores de bordo utilizado em bicicleta.

Uma segunda frente de trabalho pode ser desenvolvida, para aumentar a interface entre o computador de bordo e o piloto, que hoje é inexistente. Para isto, seria necessário a adição de um *display* de cristal líquido e de alguns botões, possibilitando que o piloto tenha informações importantes, como tempo de volta e temperatura de motor a sua disposição.

.

REFERÊNCIAS BIBLIOGRÁFICAS

ALECRIM, Emerson. ***Cartões de Memória flash***. InfoWester, São Paulo, 2007.

ANALOG DEVICES. ***iMEMS Accelerometers***. Disponível em: <http://www.analog.com>. Acesso em: 5 de março de 2009.

ARDUINO. ***Hardware*.** Disponível em: <http://arduino.cc/en/Main/Hardware>. Acesso em: 10 de julho de 2009.

BARWELL, R. et al. ***Professional Visual Basic.NET***, Pearson Education do Brasil,São Paulo, 2004.

BIZSOLUTION. ***AT&T's global enterprise business services***. Disponível em:<http://bizsolutions.posterous.com/?page=3>. Acesso em: 08 de dezembro de 2009.

BOSER, Bernhard E; HOWE, Roger T. ***Surface micromachined Accelerometers***, California, 1995.

BRAGA, Newton C. ***Reed-Swiches*.** Revista Mecatrônica Fácil – Ano 6, nº 36. Set/out 2007.

CEFTRU, UnB. ***Elementos de Arquitetura de Software*.** 2007.

DUARTE, Fábio. ***Arquitetura e tecnologias de informação da revolução industrial à revolução digital***, São Paulo: Annablume, 1999.

DIMENSION ENGINEERING. ***Datasheet DE-ACCM5G*.** Disponível em: <http://www.dimensionengineering.com/datasheets/DE-ACCM5G.pdf > acesso em : 10 de março de 2009

GAMMA Erich. et al. ***Padrões de Projeto***: Solução Reutilizáveis de Software Orientado a Objeto, 1 ed., São Paulo: Bookman, 2000.

IMBRAHIM, Dogan. ***Advanced PIC Microcontroller projects in C***, Elsevier, Oxford, 2008.

KERNIGHAN, Brian W.; RITCHIE, Dennis M. ***The C programming language,*** Prentice Hall, Englewood, 2nd ed,. 1988.

INJETEC, Automação Eletrônica – **Cronotec Kart III**. Disponível em : <http://www.injetec.com.br/racing.php> - Acesso em: 20 de Julho de 2009.

L. COCCO , P. DAPONTE. ***Metrology and Formula One Car***, IEEE International Instrumentation and Measurement , 2008.

MACHADO Felipe. ***Banco de Dados Projeto e Implementação***, São Paulo: Erica,- 2004.

MARTINS, Jorge. ***Motores de combustão interna***, Publindústria, Porto, 2006.

MASSUDA, Sergio. ***Uso de acelerômetro MEMS para aferir o desempenho de automóveis***, Uniceub, Brasília, 2007.

MATTOS, Alessandro Nicoli. ***Telemetria e conceitos relacionados***, São José dos Campos, 2004.

MEMS AND NANOTECHNOLOGY CLEARINGHOUSE. ***What is MEMS?***. Disponível em: <http://www.memsnet.org/>. Acesso em: 5 de março 2009.

MSDN. ***.NET Framework Developer Center***. Disponível em: <http://msdn.microsoft.com/pt-br/netframework/default.aspx>. Acesso em: 10 de março de 2009.

NATIONAL SEMICONDUCTOR. ***Datasheet LM35***. Disponível em: <http://www.datasheetcatalog.org/datasheet/nationalsemiconductor/DS005516.PDF >. Acesso em: 15 de março de 2009.

NOLL, Valdir; SOARES, Flávio Augusto. ***Transdutores***, disponível em: <http://florianopolis.ifsc.edu.br/~vnoll/transdutores2.pdf > Acesso em: 13 de maio de 2009.

NORTON, Peter. ***Introdução à Informática***, Makron Books, São Paulo, 1997.

TIPLER,Paul A. ***Física*** :Volume 1. 4ª Ed. São Paulo:LTC, 2000.

RCMASTER. Artigos – **Motores a Combustão, 4 e 2 tempos**. Disponível em: <http://www.rcmasters.com.br/files/motores2e4tempos.pdf> – Acesso em: 20 de Abril de 2009.

ROSSETTI, Arthur. **Da Oficina: Mille Economy.** Disponível em: <http://www.webmotors.com.br/wmpublicador/Manutencao_Conteudo.vxlpub?hnid=42116>. Acesso em 5 de dezembro de 2009.

SECURE DIGITAL CARD. **SD *Pin out.*** Disponível em: <http://www.interfacebus.com/Secure_Digital_Card_Pinout.html>. Acesso em: 5 de Março de 2009.

SHARP, John. **Microsoft Visual C#:** Passo a Passo. 1.ed. São Paulo:Bookman, 2007.

SQLite. **About SQLite**. Disponivel em:<http://www.sqlite.org/about.html>. Acesso em: 12 de Abril de 2009.

SD ASSOCIATION. **Sd Technology Overview**. Disponível em: <http://www.sdcard.org/developers/tech/> Acesso em: 12 de Maio de 2009.

APÊNDICE A – CADASTRAR PILOTO

Controlador Cadastrar Piloto, para todos os cadastros feitos pelo sistema *off-line*, segue a mesma estrutura de código.

```
using System;
using System.Data;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Framework.src.br.simemp.apresentacao;
using Framework.src.br.simemp.negocio;
using Framework.src.br.simemp.dominio;
using NegocioKartComp.src.br.facade;

using TransferObjectKart.src.br.valueobject;

using kartComp2.src.br.apresentacao;

namespace kartComp2.src.br.apresentacao
{
    class CtrlCadastrarPiloto
    {


        private FormCadastrarPiloto frmCadastraPiloto;
        private FacadeNegocioKartComp vFacade;

        Piloto vPiloto = null;
        AbstractBO vBO;

        public string ConteudoDescricao { get; set; }

        public CtrlCadastrarPiloto() { }

        // Inicializa a tela
        public void iniciar(){
            try
            {
                frmCadastraPiloto  = new FormCadastrarPiloto();
                frmCadastraPiloto.Text = "Cadastro de Pilotos";
                this.criaEventos();
                frmCadastraPiloto.ShowDialog();

            }
            catch (Exception)
            {

                throw;
            }

        }

        //Cria os eventos dos componentes da tela(Botões, Grid)
        protected void criaEventos()
        {
```

```
            try
            {
                frmCadastraPiloto.btnFechar.Click += new
EventHandler(btnFechar_Click);
                frmCadastraPiloto.btnGravar.Click += new
EventHandler(btnGravar_Click);
                frmCadastraPiloto.btnReset.Click += new
EventHandler(btnReset_Click);
                frmCadastraPiloto.btnPesquisar.Click += new
EventHandler(btnPesquisar_Click);
                frmCadastraPiloto.dgPiloto.SelectionChanged += new
EventHandler(dgPiloto_SelectionChanged);
                frmCadastraPiloto.dgPiloto.CellBeginEdit += new
System.Windows.Forms.DataGridViewCellCancelEventHandler(dgPiloto_CellB
eginEdit);
                frmCadastraPiloto.dgPiloto.CellValueChanged += new
System.Windows.Forms.DataGridViewCellEventHandler(dgPiloto_CellValueCh
anged);
                frmCadastraPiloto.btnExcluir.Click += new
EventHandler(btnExcluir_Click);


            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
        //Executa a pesquisa de Pilotos quando o botão pequisar é
acionado
        void btnPesquisar_Click(object sender, EventArgs e)
        {

            this.carregarTodosPilotos();

        }
        //Preenche um Value Object com os valores do campo
        public Piloto  formToVo() {
            vPiloto = null;
            try
            {
              vPiloto = new Piloto();

              vPiloto.Nome = frmCadastraPiloto.txtNome.Text;
              vPiloto.Categoria =(string )
frmCadastraPiloto.cbxCategoria.SelectedItem ;

              return vPiloto;

            }

            catch (Exception)

            {

                throw;

            }
        }
        //Limpa os campos da tela quando acionado
        void btnReset_Click(object sender, EventArgs e)
```

```
        {
            frmCadastraPiloto.txtNome.Text  = "";
            frmCadastraPiloto.cbxCategoria.SelectedText  = "";
        }

        //Grava o piloto quando botão for pressionado.
        void btnGravar_Click(object sender, EventArgs e)
        {
            try
            {
                this.gravaPiloto(formToVo());
                MessageBox.Show("Operacao concluída com sucesso");
                frmCadastraPiloto.Close();
            }
            catch (Exception)
            {
                throw;
            }
        }
        //Fecha a Tela quando pressionar o botão fechar
        void btnFechar_Click(object sender, EventArgs e)
        {
            frmCadastraPiloto.Close();
        }
        //Executa a rotina de carregar o objeto de piloto
        public void carregaVoPiloto(int pCodPiloto, string pNome,
string pCategoria) {
            vPiloto = new Piloto();

            try {
                vPiloto.Id = pCodPiloto;
                vPiloto.Nome = pNome;
                vPiloto.Categoria = pCategoria;
                gravaPiloto(vPiloto);
            }catch(Exception ){
                throw;
            }
        }
        //Carrega o objeto de Piloto com dados do Banco
        public Piloto carregaParametrosPiloto(Piloto pPiloto) {

            try
            {
                vBO = new AbstractBO();
                vPiloto = (Piloto)vBO.consultar(pPiloto );
                return vPiloto;
            }
            catch (Exception)
            {

                throw;
            }
        }
        //Método que grava o objeto de Piloto no Banco
        public void gravaPiloto(Piloto pPiloto)
        {
            AbstractBO bo = null;

            try
            {
                bo = new AbstractBO();
```

```
                bo.gravar(pPiloto);

            }
            catch (Exception)
            {
                throw;
            }
        }
        //Método que exclui o piloto do banco
        public void excluirPiloto(Piloto pPiloto) {
            AbstractBO bo = null;
            try
            {
                bo = new AbstractBO();
                bo.excluir(pPiloto);

            }
            catch (Exception)
            {

                throw;
            }
        }
        //Método que carrega Piloto do banco sem filtro
        public void carregarTodosPilotos()
        {
            try
            {
                vFacade = new FacadeNegocioKartComp();
                DataSet vDSPilotos;

                vDSPilotos = vFacade.pesquisarTodosPilotos(" Nome like
'%" + frmCadastraPiloto.txtNome.Text + "%'");

                carregaDataGrid(vDSPilotos);
            }
            catch (Exception)
            {

                throw;
            }

        }
        //Método que carrega o DataGrid
        public void carregaDataGrid(DataSet pDSPiloto)
        {
            try
            {
                frmCadastraPiloto.dgPiloto.AutoGenerateColumns =
false;
                frmCadastraPiloto.cod_piloto.DataPropertyName = "id";
                frmCadastraPiloto.dgvPiloto.DataPropertyName = "nome";
                frmCadastraPiloto.dgvCategoria.DataPropertyName =
"categoria";
                frmCadastraPiloto.dgPiloto.DataSource =
pDSPiloto.Tables[0].DefaultView;

            }
            catch (Exception)
            {
```

```
                throw;
            }

        }
        //Metodo que executa uma rotina quando algum piloto é
selecionado no datagrid
        void dgPiloto_SelectionChanged(object sender, EventArgs e)
        {
            try
            {
                vPiloto = new Piloto();
                CtrlCadastrarPiloto ctrlCadPiloto = new
CtrlCadastrarPiloto();

                vPiloto.Id =
int.Parse(frmCadastraPiloto.dgPiloto.CurrentRow.Cells["cod_piloto"].Va
lue.ToString());
                vPiloto =
ctrlCadPiloto.carregaParametrosPiloto(vPiloto);


            }
            catch (Exception)
            {
                throw;
            }
        }
        //Metodo que executa uma rotina quando é alterado um valor no
datagrid
        void dgPiloto_CellValueChanged(object sender,
System.Windows.Forms.DataGridViewCellEventArgs e)
        {
            try
            {
                CtrlCadastrarPiloto ctrlCadPiloto = new
CtrlCadastrarPiloto();
                if (frmCadastraPiloto.dgPiloto.CurrentCell != null)
                {
                    string vPiloto =
frmCadastraPiloto.dgPiloto.CurrentRow.Cells["dgvPiloto"].Value.ToStrin
g();
                    string vCategoria =
frmCadastraPiloto.dgPiloto.CurrentRow.Cells["dgvCategoria"].Value.ToSt
ring();

                    if (vPiloto != this.ConteudoDescricao)
                    {

ctrlCadPiloto.carregaVoPiloto(int.Parse(frmCadastraPiloto.dgPiloto.Cur
rentRow.Cells["cod_piloto"].Value.ToString()), vPiloto, vCategoria);
                    }


                }

            }
            catch (Exception)
            {
                throw;
            }
        }
```

```
        //Metodo que executa uma rotina quando se edita o campo do
datagrid
        void dgPiloto_CellBeginEdit(object sender,
System.Windows.Forms.DataGridViewCellCancelEventArgs e)
        {
            this.ConteudoDescricao =
frmCadastraPiloto.dgPiloto.CurrentCell.Value.ToString();
        }
        //Exclui o piloto selecionado
        void btnExcluir_Click(object sender, EventArgs e)
        {
            try
            {
                this.excluirPiloto(vPiloto);
                this.carregarTodosPilotos();

            }
            catch (Exception)
            {

                throw;
            }
        }

    }
}
```

DAO Piloto, classe de acesso ao banco de dados, a estrutura é a mesma para todos os cadastros feitos no sistema *Off-line.* Para cada cadastro, existe uma classe específica.

```
using PersistenciaKartComp.src.br.dataacessobject.interfaces;

using System.Data.SQLite;

using Framework.src.br.simemp.persistencia;
using Framework.src.br.simemp.exception;

using Framework.src.br.simemp.connection;
using Framework.src.br.simemp.dominio;

using TransferObjectKart.src.br.valueobject;

namespace PersistenciaKartComp.src.br.dataacessobject
{
    class DAOPiloto : AbstractDAO
    {
        public DAOPiloto()
            : base()
        {
        }

        public DAOPiloto(DbTransaction pTransacao)
            : base(pTransacao)
        {
```

```
        }

        public DAOPiloto(DbConnection pConn)
            : base(pConn)
        {
        }

        private const string SQL_PESQUISAR_PILOTOS = "SELECT " +
                                                     "
tblPiloto.Id, " +
                                                     "
tblPiloto.nome, " +
                                                     "
tblPiloto.categoria " +
                                                     " FROM " +
                                                     "   tblPiloto "
+
                                                     " WHERE ";

        private const string SQL_PESQUISAR_PILOTOS_EVENTO = "SELECT "
+
                                                    "   tblPiloto.Id, " +
                                                    "   tblPiloto.nome, " +
                                                    "   tblPiloto.categoria
" +
                                                    " FROM " +
                                                    "   tblPiloto " +
                                                    " WHERE ";
        private const string SQL_PESQUISAR_PILOTO_SELETOR = "select
pi.id as id, pi.nome as Nome, pi.categoria as Categoria, ev.tipo as
Evento, ev.dataEvento as dataEvento " +
                                                        " from tblPiloto
pi, tblPilotoEvento pilEv, tblEvento ev " +
                                                        " where pi.id =
pilEv.idPiloto " +
                                                        " and ev.id =
pilEv.idEvento ";



        public DataSet pesquisarPilotos(String pExpressao)
        {
            IDbCommand vComm = null;
            IDbDataAdapter vDA;
            DataSet vDS = new DataSet();

            try
            {

                vComm = this.criaCommand();
                if (pExpressao.Equals(""))
                {
                    vComm.CommandText =
String.Format(SQL_PESQUISAR_PILOTO_SELETOR);
                }
                else
                {
                    vComm.CommandText =
String.Format(SQL_PESQUISAR_PILOTO_SELETOR + pExpressao);
                }
```

```
                vDA =
((IConnection)FactoryConnection.getInstance().getLastConnectionSet()).
getNewDataAdapter();

                vDA.SelectCommand = vComm;

                vDA.Fill(vDS);

                return vDS;

            }
            catch (ExceptionAbstract)
            {
                throw;
            }
            catch (Exception ex)
            {
                throw new ExceptionDAO(ex.Message, ex);
            }

        }

        public DataSet pesquisarTodosPilotos(String pExpressao)
        {
            IDbCommand vComm = null;
            IDbDataAdapter vDA;
            DataSet vDS = new DataSet();

            try
            {

                vComm = this.criaCommand();
                if (pExpressao.Equals(""))
                {
                    vComm.CommandText =
String.Format(SQL_PESQUISAR_PILOTOS);
                }
                else
                {
                    vComm.CommandText =
String.Format(SQL_PESQUISAR_PILOTOS + pExpressao);
                }


                vDA =
((IConnection)FactoryConnection.getInstance().getLastConnectionSet()).
getNewDataAdapter();

                vDA.SelectCommand = vComm;

                vDA.Fill(vDS);

                return vDS;

            }
            catch (ExceptionAbstract)
            {
                throw;
            }
            catch (Exception ex)
```

```
            {
                throw new ExceptionDAO(ex.Message, ex);
            }

        }

        public DataSet pesquisarPilotosEvento(String pExpressao)
        {
            DataSet vDS = null;
            string expressao;

            try
            {

                vDS = this.pesquisaExpressaoDS(typeof(Piloto),
pExpressao);
                return vDS;

            }
            catch (ExceptionAbstract)
            {
                throw;
            }
            catch (Exception ex)
            {
                throw new ExceptionDAO(ex.Message, ex);
            }

        }

        public DataSet pesquisarPilotosLikeNome(string nome)
        {
            DataSet vDS = null;
            string expressao;

            try
            {
                expressao = "nome like '%" + nome + "%'";

                vDS = this.pesquisaExpressaoDS(typeof(Piloto),
expressao);
                return vDS;

            }
            catch (ExceptionAbstract)
            {
                throw;
            }
            catch (Exception ex)
            {
                throw new ExceptionDAO(ex.Message, ex);
            }
        }

    }
}
```

Classe de Importação de Dados

```
namespace kartComp2.src.br.apresentacao
{
    class CtrlImportacao
    {
        public FormImportacao formImportacao ;
        public String pathArquivo = "C:\02 - Projeto
Final\teste1.txt";
        public int tempoDeVolta = 0;
        public int numeroDeVolta = 0;
        public int sequenciaTrecho = 0;
        public int contaNumero = 0;
        public int pTempo = 0;
        public int pAceleracao = 0;
        public int pAceleracaoLateral = 0;
        public int pTemperatura = 0;
        public AbstractBO bo;
        public PilotoEvento pilotoEventoVO;
        public Pista vPista;
        public Evento vEvento;

        public void iniciar() {
            formImportacao = new FormImportacao();
            formImportacao.Text = "Importar Dados";
            this.criarEventos();
            this.carregarCombos();
            formImportacao.ShowDialog();
        }

        public void criarEventos() {
            formImportacao.btnImportar.Click += new
EventHandler(btnImportar_Click);
            formImportacao.cbxEvento.Click += new
EventHandler(cbxEvento_Click);
        }

        void cbxEvento_Click(object sender, EventArgs e)
        {
            bo = new AbstractBO();
            vEvento = new Evento();
            vPista = new Pista();

            vEvento.Id =
Convert.ToInt32(formImportacao.cbxEvento.SelectedValue);
            vEvento = (Evento)bo.consultar(vEvento);

            vPista = (Pista)bo.consultar(vEvento.pista);
            formImportacao.txtPista.Text = vPista.Nome;
            formImportacao.txtPista.Dispose();
        }


        void btnImportar_Click(object sender, EventArgs e)
        {
            try
            {
                this.gravaPilotoEvento(this.formToVo());
                this.lerArquivo();
            }
            catch (Exception)
            {
```

```
                throw;
            }

        }


        public void gravaPilotoEvento(PilotoEvento pPilotoEvento)
        {
            AbstractBO bo = null;

            try
            {
                bo = new AbstractBO();
                bo.gravar(pPilotoEvento);

            }
            catch (Exception)
            {
                throw;
            }
        }

        public void carregarCombos() {
            this.carregaPiloto();
            this.carregaEquipamento();
            this.carregaEvento();
        }
        public void carregaPiloto()
        {
            ArrayList vListaPiloto = null;
            AbstractBO bo = null;

            try
            {
                bo = new AbstractBO();
                vListaPiloto = bo.consultar(typeof(Piloto));
                if (vListaPiloto != null)
                {
                    formImportacao.cbxPiloto.DataSource =
vListaPiloto;
                    formImportacao.cbxPiloto.DisplayMember = "nome";
                    formImportacao.cbxPiloto.ValueMember = "id";
                }
            }
            catch (Exception)
            {
                throw;
            }
        }

        public void carregaEquipamento()
        {
            ArrayList vListaEquipamento = null;
            try
            {
                bo = new AbstractBO();
                vListaEquipamento = bo.consultar(typeof(Equipamento));
                if (vListaEquipamento != null)
                {
                    formImportacao.cbxEquipamento.DataSource =
vListaEquipamento;
```

```
                    formImportacao.cbxEquipamento.DisplayMember =
"Chassis";
                    formImportacao.cbxEquipamento.ValueMember = "id";
                }
            }
            catch (Exception)
            {
                throw;
            }
        }

        public void carregaEvento()
        {
            ArrayList vListaEvento = null;
            try
            {
                bo = new AbstractBO();
                vEvento = new Evento();
                vPista = new Pista();

                vListaEvento = bo.consultar(typeof(Evento));
                if (vListaEvento != null)
                {
                    formImportacao.cbxEvento.DataSource =
vListaEvento;
                    formImportacao.cbxEvento.DisplayMember = "Tipo";
                    formImportacao.cbxEvento.ValueMember = "id";

                    vEvento.Id =
Convert.ToInt32(formImportacao.cbxEvento.SelectedValue);
                    vEvento = (Evento)bo.consultar(vEvento);

                    vPista = (Pista)bo.consultar(vEvento.pista);
                    formImportacao.txtPista.Text = vPista.Nome;
                    formImportacao.txtPista.ReadOnly = true ;
                }
            }
            catch (Exception)
            {
                throw;
            }
        }

        public PilotoEvento formToVo()
        {
            pilotoEventoVO = null;
            try
            {
                pilotoEventoVO = new PilotoEvento();

                pilotoEventoVO.piloto = new Piloto();
                pilotoEventoVO.evento = new Evento();
                pilotoEventoVO.equipamento = new Equipamento();


                pilotoEventoVO.piloto.Id =
(int)formImportacao.cbxPiloto.SelectedValue;
                pilotoEventoVO.evento.Id =
(int)formImportacao.cbxEvento.SelectedValue;
                pilotoEventoVO.equipamento.Id =
(int)formImportacao.cbxEquipamento.SelectedValue;
```

```
                return pilotoEventoVO;

            }

            catch (Exception)
            {

                throw;

            }
        }



        /*public void lerArquivo()
        {
            //caminho do diretorio
            StreamReader rd = new StreamReader(@"C:\02 - Projeto
Final\teste3voltas.txt");
            Volta voltaVO = new Volta();
            Trecho trechoVO = new Trecho();
            Parametros parametroVO = new Parametros();


            int TempoVolta = 0;
            int TempoTrecho = 0;

            //faz a validação do arquivo
            while (!rd.EndOfStream)
            {
                string linha = rd.ReadLine();
                string[] linhaNova = linha.Split(new char[] { ';' });

                int n = Int32.Parse(linhaNova[0]);
                int tempo = Int32.Parse(linhaNova[1]);
                if(n == 2){
                     pAceleracao  = Int32.Parse(linhaNova [2]);
                     pAceleracaoLateral = Int32.Parse(linhaNova[3]);
                     pTemperatura  = Int32.Parse(linhaNova[4]);
                }

                FacadeNegocioKartComp vFacade = new
FacadeNegocioKartComp();

                switch (n)
                {
                    case 1:
                        if (sequenciaTrecho == 0) //Abertura do Evento
                        {
                            numeroDeVolta = 1;
                            //Abre a 1a volta e 1o trecho
                            this.abreVolta();

                        }
                        else
                        {
                            //TempoVolta = TempoVolta + tempo;
                            //TempoTrecho = TempoTrecho + tempo;
```

```
                            if (contaNumero % 4 == 0) //Verifica se
está na terceira parcial para contagem de volta.
                            {
                                //Fecho Volta
                                voltaVO =
(Volta)bo.pesquisarExpressao(typeof(Volta), "Id = " +
vFacade.pesquisarUltimaIDVolta(typeof(Volta)))[0];
                                voltaVO.Tempo = TempoVolta;
                                voltaVO.pilotoEvento = new
PilotoEvento();
                                voltaVO.pilotoEvento.Id =
vFacade.pesquisarUltimoPilotoEvento(typeof(PilotoEvento));

                                bo.gravar(voltaVO);
                                sequenciaTrecho = 1;
                                TempoVolta = 0;
                                numeroDeVolta = numeroDeVolta + 1;

                                //Abro Volta
                                this.abreVolta();

                                //Fecho Trecho
                                trechoVO = new Trecho();
                                trechoVO.volta = new Volta();
                                trechoVO.volta.Id =
vFacade.pesquisarUltimaIDVolta(typeof(Volta));
                                trechoVO.Tempo = TempoTrecho;
                                trechoVO.Sequencia = sequenciaTrecho;

                                bo.gravar(trechoVO);
                                contaNumero = 1;
                            }
                            else
                            {
                                //Grava trechos que nao fecham voltas
                                //Fecho Trecho
                                if (sequenciaTrecho == 1)
                                {
                                    abreTrecho();
                                    trechoVO.Id =
vFacade.pesquisarUltimaIDTrecho(typeof(Trecho));
                                    trechoVO.Tempo = TempoTrecho;
                                    trechoVO.Sequencia =
sequenciaTrecho;
                                    trechoVO.volta = new Volta();
                                    trechoVO.volta.Id =
vFacade.pesquisarUltimaIDVolta(typeof(Volta));

                                    bo.gravar(trechoVO);
                                    TempoTrecho = 0;
                                }
                                else
                                {
                                    trechoVO = new Trecho();
                                    trechoVO.Tempo = TempoTrecho;
                                    trechoVO.Sequencia =
sequenciaTrecho;
                                    trechoVO.volta = new Volta();
                                    trechoVO.volta.Id =
vFacade.pesquisarUltimaIDVolta(typeof(Volta));
```

```
                                    bo.gravar(trechoVO);
                                    //Abro Trecho pra mesma volta
                                    //abreTrecho();
                                    TempoTrecho = 0;
                                }
                            }
                        }
                        contaNumero = contaNumero + 1;
                        sequenciaTrecho = sequenciaTrecho + 1;
                        break;
                    case 2:
                        //if (sequenciaTrecho > 0)
                        //{
                        TempoVolta = TempoVolta + tempo;
                        TempoTrecho = TempoTrecho + tempo;

                        parametroVO.trecho = new Trecho();
                        parametroVO.trecho.Id =
vFacade.pesquisarUltimaIDTrecho(typeof(Trecho));

                        parametroVO.Aceleracao = pAceleracao;
                        parametroVO.AceleracaoLateral =
pAceleracaoLateral;
                        parametroVO.Temperatura = pTemperatura;

                        bo.gravar(parametroVO);
                        //}
                        break;
                }

            }
            rd.Close();
            MessageBox.Show("Operacao concluída com sucesso");
        }*/

        public void lerArquivo()
        {
            //caminho do diretorio
            StreamReader rd = new StreamReader(@"C:\02 - Projeto
Final\kartComp.txt");
            Volta voltaVO = new Volta();
            Trecho trechoVO = new Trecho();
            Parametros parametroVO = new Parametros();

            int TempoVolta = 0;
            int TempoTrecho = 0;

            //faz a validação do arquivo

            while (!rd.EndOfStream)
            {
                string linha = rd.ReadLine();

                if (linha.Length != 0)
                {

                    string[] linhaNova = linha.Split(new char[] { ';'
});

                    int n = Int32.Parse(linhaNova[0]);
                    int tempo = Int32.Parse(linhaNova[1]);
```

```
                    if (n == 2)
                    {
                        pAceleracao = Int32.Parse(linhaNova[2]);
                        pAceleracaoLateral =
Int32.Parse(linhaNova[3]);
                        pTemperatura = Int32.Parse(linhaNova[4]);
                    }

                    FacadeNegocioKartComp vFacade = new
FacadeNegocioKartComp();

                    switch (n)
                    {
                        case 1:
                            if (sequenciaTrecho == 0) //Abertura do
Evento
                            {

                                numeroDeVolta = 1;
                                //Abre a 1a volta e 1o trecho
                                this.abreVolta();

                            }
                            else
                            {
                                //Se a sequencia é diferente de 0
verifica a contagem de volta.
                                if (contaNumero % 4 == 0) //Verifica
se está na terceira parcial para contagem de volta.
                                {

                                    //Fecho Volta

                                    voltaVO =
(Volta)bo.pesquisarExpressao(typeof(Volta), "Id = " +
vFacade.pesquisarUltimaIDVolta(typeof(Volta)))[0];
                                    voltaVO.Tempo = TempoVolta;
                                    voltaVO.pilotoEvento = new
PilotoEvento();
                                    voltaVO.pilotoEvento.Id =
vFacade.pesquisarUltimoPilotoEvento(typeof(PilotoEvento));

                                    bo.gravar(voltaVO);
                                    sequenciaTrecho = 1;
                                    TempoVolta = 0;
                                    numeroDeVolta = numeroDeVolta + 1;

                                    //Abro Volta
                                    this.abreVolta();

                                    //Fecho Trecho

                                    trechoVO = new Trecho();
                                    trechoVO.volta = new Volta();
                                    trechoVO.volta.Id =
vFacade.pesquisarUltimaIDVolta(typeof(Volta));
                                    trechoVO.Tempo = TempoTrecho;
                                    trechoVO.Sequencia =
sequenciaTrecho;

                                    bo.gravar(trechoVO);
```

```
                                    contaNumero = 1;
                                }
                                else
                                {
                                    //Grava trechos que nao fecham
voltas
                                    //Fecho Trecho
                                    if (sequenciaTrecho == 1)
                                    {
                                        abreTrecho();
                                        trechoVO.Id =
vFacade.pesquisarUltimaIDTrecho(typeof(Trecho));
                                        trechoVO.Tempo = TempoTrecho;
                                        trechoVO.Sequencia =
sequenciaTrecho;
                                        trechoVO.volta = new Volta();
                                        trechoVO.volta.Id =
vFacade.pesquisarUltimaIDVolta(typeof(Volta));

                                        bo.gravar(trechoVO);
                                        TempoTrecho = 0;


                                    }
                                    else
                                    {

                                        trechoVO = new Trecho();
                                        trechoVO.Tempo = TempoTrecho;
                                        trechoVO.Sequencia =
sequenciaTrecho;
                                        trechoVO.volta = new Volta();
                                        trechoVO.volta.Id =
vFacade.pesquisarUltimaIDVolta(typeof(Volta));

                                        bo.gravar(trechoVO);
                                        //Abro Trecho pra mesma volta
                                        //abreTrecho();
                                        TempoTrecho = 0;
                                    }
                                }
                            }
                            contaNumero = contaNumero + 1;
                            sequenciaTrecho = sequenciaTrecho + 1;

                            break;
                        case 2:

                            TempoVolta = TempoVolta + tempo;
                            TempoTrecho = TempoTrecho + tempo;

                            parametroVO.trecho = new Trecho();
                            parametroVO.trecho.Id =
vFacade.pesquisarUltimaIDTrecho(typeof(Trecho));

                            parametroVO.Aceleracao = pAceleracao;
                            parametroVO.AceleracaoLateral =
pAceleracaoLateral;
```

```
                                parametroVO.Temperatura =
pTemperatura;
                            parametroVO.Tempo = tempo;

                            bo.gravar(parametroVO);
                            break;
                        }

                    }
                }
                rd.Close();
                MessageBox.Show("Operacao concluída com sucesso");
            }
            public void abreVolta()
            {
                Volta voltaVO = new Volta();
                FacadeNegocioKartComp vFacade = new
FacadeNegocioKartComp();

                voltaVO.pilotoEvento = new PilotoEvento();
                voltaVO.pilotoEvento.Id =
vFacade.pesquisarUltimoPilotoEvento(typeof(PilotoEvento));
                voltaVO.VoltaNumero = numeroDeVolta;
                bo.gravar(voltaVO);

            }
            public void abreTrecho()
            {
                FacadeNegocioKartComp vFacade = new
FacadeNegocioKartComp();
                Trecho trechoVO = new Trecho();
                trechoVO.volta = new Volta();
                trechoVO.volta.Id =
vFacade.pesquisarUltimaIDVolta(typeof(Volta));
                trechoVO.Sequencia = sequenciaTrecho;
                bo.gravar(trechoVO);
            }


        }


}
```

Classe de Controlador dos Gráficos

```
namespace kartComp2.src.br.apresentacao
{
    class CtrlGrafico
    {
        public FormGrafico frmGrafico = null;
        private AbstractBO bo = null;
        private FacadeNegocioKartComp facade = null;
        private Pista vPista = null;
        private Evento vEvento = null;
        private Parametros vParametro = null;
        private PilotoEvento vPilotoEvento = null;
        private List<PilotoEvento> vListaPilotoEvento;
```

```
        int vTempo = 0 ;

        //valores de preenchimento do grafico
        int vYMax = 0;
        int vXMax = 0;
        int vYMin = 0;
        int vXMin = 0;
        int vIntervalY = 0;
        int vIntervalX = 0;
        string vTitleX;
        string vTitley;

        List<DataSet> vListaParametros;

        public void iniciar(List<Piloto> pPiloto, PilotoEvento
pPilotoEvento, string pTipoGrafico, List<PilotoEvento>
pListaPilotoEvento)
        {
            frmGrafico = new FormGrafico();
            frmGrafico.Text = "Tela de Gráfico";


            try
            {
                vPilotoEvento = new PilotoEvento();
                vPilotoEvento = pPilotoEvento;
                vListaPilotoEvento = pListaPilotoEvento;
                this.criarEventos();
                this.carregarParametros();
                CriaPilotos(pPiloto);
                this.carregarTempoEvento();
                this.verificaTipoGrafico(pTipoGrafico );
                initCharts();
                carregarAceleracao();

                if (pTipoGrafico.Equals("aceleracao"))
                {
                    this.populaGraficoAceleracao();
                }
                else if (pTipoGrafico.Equals("aceleracaoLateral"))
                {
                    this.populaGraficoAceleracaoLateral();
                }
                else
                {
                    this.populaGraficoTemperatura();
                }
                frmGrafico.Show();
            }
            catch (Exception)
            {
                throw;
            }
        }

        public void criarEventos()
        {

            frmGrafico.btnFechar.Click += new
EventHandler(btnFechar_Click);
        }
```

```
        void btnFechar_Click(object sender, EventArgs e)
        {
            frmGrafico.Close();
        }
        public void verificaTipoGrafico(string pTipoGrafico)
        {

if((pTipoGrafico.Equals("aceleracao"))||(pTipoGrafico.Equals("acelerac
aoLateral")))
            {
                vYMax = 5;
                vYMin = -5;
                vXMax = vTempo;
                vXMin = 0;
                vIntervalX = 500;
                vIntervalY = 1;
                vTitleX = "Tempo(ms)";
                vTitley = pTipoGrafico+"(g)";
            }else
            {
                vYMax = 50;
                vYMin = 15;
                vXMax = vTempo;
                vXMin = 0;
                vIntervalX = 500;
                vIntervalY = 5;
                vTitleX = "Tempo(ms)";
                vTitley = "Temperatura(ºC)";
            }
        }

        public void carregarTempoEvento() {
            int tmp1 = 0;
            int tmp2 = 0;
            try
            {
                facade = new FacadeNegocioKartComp();
                tmp1 =
facade.pesquisarTempoEventoID(vListaPilotoEvento[0].Id);
                if (vListaPilotoEvento.Count > 1)
                    tmp2 =
facade.pesquisarTempoEventoID(vListaPilotoEvento[1].Id);
                vTempo = (tmp1 > tmp2) ? tmp1 : tmp2;
            }
            catch (Exception)
            {

                throw;
            }

        }

        //carrega a aceleração do pilotoEvento
        private void carregarAceleracao()
        {
            try
            {

                facade = new FacadeNegocioKartComp();
                vListaParametros = new List<DataSet>();
```

```
                foreach (PilotoEvento  vPilotinho in
vListaPilotoEvento )
                {
                    DataSet vDSLocal = new DataSet();

                    vDSLocal =
facade.pesquisarParametrosPilotoEvento(vPilotinho.Id);
                    vListaParametros.Add(vDSLocal);
                }



            }
            catch (Exception)
            {

                throw;
            }

        }

        private void CriaPilotos(List<Piloto> pPiloto)
        {
            int contador = 0;
            foreach (Piloto vPiloto in pPiloto)
            {
                Series serie = new Series();
                serie.Name = vPiloto.Nome;
                serie.Tag = contador;
                serie.ChartType = SeriesChartType.Line;

                frmGrafico.grfAceleracao.Series.Add(serie);
                contador++;
            }
        }

        private void populaGraficoAceleracao()
        {
            int x = 0;
            double y = 0;

            DataSet vDS = new DataSet();
            facade = new FacadeNegocioKartComp();

            try
            {
                //Lista com os pilotos selcionados
                for (int i = 0; i <
frmGrafico.grfAceleracao.Series.Count; i++)
                {
                    //this.carregarAceleracao();
                    x = 0;
                    //lista com os parametros
                    foreach (DataRow dr in
vListaParametros[i].Tables[0].Rows)
                    {
                        //Preenche valores de y
                        y = double.Parse(dr["aceleracao"].ToString());
                        //y = ((y - 512) * 10) / 64; //Conversão em
g/10
```

```
                        y = (y * 5 / 1023 - 2.5) / 0.312;

                        if (y > 5)
                            y = 5;
                        else if (y < -5)
                            y = -5;

                        //Preenche valores de x
                        x = x +
int.Parse(dr["tempoParametro"].ToString());

atualizarGrafico(frmGrafico.grfAceleracao.Series[i], y, x, 5);
                    }

                }

            }
            catch (Exception err)
            {
                string messa = err.Message;
            }
        }

        private void populaGraficoAceleracaoLateral()
        {
            int x = 0;
            double y = 0;

            vParametro = new Parametros();
            DataSet vDS = new DataSet();
            facade = new FacadeNegocioKartComp();

            try
            {

                for (int i = 0; i <
frmGrafico.grfAceleracao.Series.Count; i++)
                {
                    //this.carregarAceleracao();
                    x = 0;
                    foreach (DataRow dr in
vListaParametros[i].Tables[0].Rows)
                    {
                        //Preenche valores de y
                        y =
double.Parse(dr["aceleracaoLateral"].ToString());
                        //y = ((y - 512) * 10) / 64;
                        y = (y * 5 / 1023 - 2.5) / 0.312;
                        //Calibraçao de aceleracao lateral
                        y = y + 0.2;
                        if (y > 5)
                            y = 5;
                        else if (y < -5)
                            y = -5;

                        //Preenche valores de x
                        x = x +
int.Parse(dr["tempoParametro"].ToString());


atualizarGrafico(frmGrafico.grfAceleracao.Series[i], y, x, 5);
```

```
                    }

                }

            }
            catch (Exception err)
            {
                string messa = err.Message;
            }
        }


        private void populaGraficoTemperatura()
        {
            int x = 0;
            double y = 0;

            Random rnd = new Random();
            vParametro = new Parametros();
            DataSet vDS = new DataSet();
            facade = new FacadeNegocioKartComp();

            try
            {   //leitura da lista de piloto
                for (int i = 0; i <
frmGrafico.grfAceleracao.Series.Count; i++)
                {
                    //this.carregarAceleracao();
                    x = 0;
                    //leitura da lista com os parametros para cada
piloto
                    foreach (DataRow dr in
vListaParametros[i].Tables[0].Rows)
                    {
                        //Preenche valores de y
                        //Conversão do valor gravado no banco para ºC.
                        y = 500 *
double.Parse(dr["temperatura"].ToString()) / 1023;
                        //Preenche valores de x
                        x = x +
int.Parse(dr["tempoParametro"].ToString());


atualizarGrafico(frmGrafico.grfAceleracao.Series[i], y, x, 5);

                    }
                }
            }
            catch (Exception err)
            {
                string messa = err.Message;
            }
        }

        private void atualizarGrafico(Chart pChart, Double pYValue,
int pXValue, Double pDefaultSerieChart)
        {
            int i = 0;
            double vLastXValue;
```

```
            try
            {

                foreach (Series vSeries in pChart.Series)
                {
                    vLastXValue = vSeries.Points[vSeries.Points.Count
- 1].XValue;

                    if (i == 0)
                    {
                        vSeries.Points.AddXY(pXValue, pYValue);
                    }
                    else
                    {
                        vSeries.Points.AddXY(pXValue,
pDefaultSerieChart);
                    }

                    if (vLastXValue == -1)
                    {
                        vSeries.Points.RemoveAt(0);
                    }


                    i++;
                }


            }
            catch (Exception)
            {
                throw;
            }
        }
        //Pontos atribuídos ao gráfico.
        private void atualizarGrafico(Series piloto, Double pYValue,
int pXValue, Double pDefaultSerieChart)
        {

frmGrafico.grfAceleracao.Series[int.Parse(piloto.Tag.ToString())].Poin
ts.AddXY(pXValue, pYValue);
        }
        //Montagem do gráfico
        private void initCharts()
        {
            try
            {
                //Y - Eixo dos Parametros
                frmGrafico.grfAceleracao.ChartAreas[0].AxisY.Minimum =
vYMin ;
                frmGrafico.grfAceleracao.ChartAreas[0].AxisY.Maximum =
vYMax ;
                frmGrafico.grfAceleracao.ChartAreas[0].AxisY.Interval
= vIntervalY ;
                frmGrafico.grfAceleracao.ChartAreas[0].AxisY.Title =
vTitley ;
                //X - Eixo da Tempo
                frmGrafico.grfAceleracao.ChartAreas[0].AxisX.Minimum =
vXMin ;
                frmGrafico.grfAceleracao.ChartAreas[0].AxisX.Maximum =
vXMax ;
```

```
                frmGrafico.grfAceleracao.ChartAreas[0].AxisX.Interval
= vIntervalX ;


frmGrafico.grfAceleracao.ChartAreas[0].AxisX.LabelStyle.Enabled =
true;

frmGrafico.grfAceleracao.ChartAreas[0].AxisX.MajorGrid.Enabled =
false;
                frmGrafico.grfAceleracao.ChartAreas[0].AxisX.Title =
vTitleX ;

            }
            catch (Exception)
            {
                throw;
            }
        }

        //carrega os parametros do gráfico
        private void carregarParametros()
        {
            bo = new AbstractBO();
            vEvento = new Evento();
            vEvento.Id = vPilotoEvento.evento.Id;
            vEvento = (Evento)bo.consultar(vEvento);
            frmGrafico.txtData.Text =
Convert.ToString(vEvento.DataEvento);
            frmGrafico.txtEvento.Text = vEvento.Tipo;
            vPista = new Pista();
            vPista.Id = vEvento.pista.Id;
            vPista = (Pista)bo.consultar(vPista);
            frmGrafico.txtPista.Text = vPista.Nome;
        }


    }
      }
```

Classe Abstrata de Acesso a Banco

```
namespace Framework.src.br.simemp.persistencia
{
    public class AbstractDAO
    {

        public AbstractDAO()
        {
            vTipoConn = TIPO_CONNEXAO.NO_CONN;

            //pega o dbparameter value de cada tipo de connexao.
            vDBParameterValue =
((IConnection)FactoryConnection.getInstance().getConnectionTypeDefault
()).getDBParameterConstant();
        }

        public AbstractDAO(IDbTransaction pTransacao)
        {
            vTipoConn = TIPO_CONNEXAO.TRANS;
```

```
            vTransacao = pTransacao;
            vConn = pTransacao.Connection;

            //pega o dbparameter value de cada tipo de connexao.
            vDBParameterValue =
((IConnection)FactoryConnection.getInstance().getLastConnectionSet()).
getDBParameterConstant();
        }

        public AbstractDAO(IDbConnection pConn)
        {
            vTipoConn = TIPO_CONNEXAO.CONN;
            vConn = pConn;

            //pega o dbparameter value de cada tipo de connexao.
            vDBParameterValue =
((IConnection)FactoryConnection.getInstance().getLastConnectionSet()).
getDBParameterConstant();
        }

        protected enum TIPO_CONNEXAO
        {
            NO_CONN = 1,
            CONN = 2,
            TRANS = 3
        }

        protected IDbConnection vConn;
        protected IDbTransaction vTransacao;
        protected TIPO_CONNEXAO vTipoConn;
        private Hashtable vHashIndicesAtributos = new Hashtable();

        protected StringBuilder orderByClause = new StringBuilder("");

        //atributo que guarda o valor do parameter dependendo do banco
de dados, mysql e ?, Oracle e : ...
        private string vDBParameterValue;

        private void setDBParameterValue()
        {
        }

        protected virtual IDbCommand criaCommand()
        {
            IDbCommand vComm = null;

            try
            {
                vComm =
((IConnection)FactoryConnection.getInstance().getLastConnectionSet()).
getNewCommand();

                if (vTipoConn == TIPO_CONNEXAO.NO_CONN)
                {
                    vConn =
((IConnection)FactoryConnection.getInstance().getConnectionTypeDefault
()).getConnection();

                } else if(vTipoConn == TIPO_CONNEXAO.TRANS)
                {
                    vComm.Transaction = vTransacao;
```

```
            }

            vComm.Connection = vConn;

            return vComm;
        }
        catch (Exception  ex)
        {
            throw new ExceptionFramework(ex.Message, ex);
        }

    }

    protected void fechaConnexao()
    {
        try
        {
            if (vTipoConn == TIPO_CONNEXAO.NO_CONN)
            {
                //fecha connexao
                if (vConn != null)
                    vConn.Close();
            }
        }
        catch (Exception  ex)
        {
            throw (ex);
        }
    }

    protected object convertLink(VO pVO)
    {
        try
        {
            if (pVO != null)
            {
                if (pVO.Id == 0)
                {
                    return DBNull.Value;
                }
                else
                {
                    return pVO.Id;
                }
            }
            else
            {
                return DBNull.Value;
            }

        }
        catch (Exception  ex)
        {
            throw (ex);
        }
    }

    protected object convert(object vCampo)
    {
        try
        {
```

```
                if (vCampo == null)
                {
                    return DBNull.Value;
                }
                else
                {
                    if (vCampo == DBNull.Value){ return null; }
                    else if (vCampo.GetType().Equals(typeof (string)))
{ return ((string)vCampo).Trim(); }
                    else if
(vCampo.GetType().Equals(typeof(DateTime)))
                    {
                        DateTime vData = new DateTime();

                        if (vData.Equals((DateTime)vCampo)) { return
DBNull.Value; }
                        else { return vCampo; }
                    }
                    else { return vCampo; }
                }
            }
            catch (Exception ex)
            {
                throw(ex);
            }
        }


        private string getSQLIncluir(VO pVO)
        {
            StringBuilder vSQLInserir = new StringBuilder("INSERT INTO
");
            MemberInfo[] vMembers;
            CustomAttributePersistence vCustom;
            object[] vObj;
            string vValue;
            ArrayList vListaValues = new ArrayList();
            //Int32 idx = 0;

            try
            {

                vCustom =
(CustomAttributePersistence)Attribute.GetCustomAttribute(pVO.GetType()
, typeof(CustomAttributePersistence));

                vSQLInserir.Append(vCustom.NomeTabela + " (");

                vMembers = pVO.GetType().GetMembers();

                foreach (MemberInfo  vMem in vMembers)
                {
                    if (vMem.MemberType == MemberTypes.Property)
                    {
                        vObj = vMem.GetCustomAttributes(true);

                        if (vObj != null)
                        {
                            if (vObj.Length > 0)
                            {
```

```
                                vCustom =
(CustomAttributePersistence)vObj[0];

                                if (!vCustom.IsIdentity)
                                {

vSQLInserir.Append(vCustom.NomeAtributoTabela + ",");
                                    vListaValues.Add(vDBParameterValue
+ vCustom.NomeAtributoTabela);

                                }
                            }
                        }
                    }

                }

                //retira a ultima virgula e substitui por fecha
parenteses
                vSQLInserir.Remove(vSQLInserir.Length - 1, 1);
                vSQLInserir.Append(" ) VALUES(");

                for (int i = 0; i <= vListaValues.Count-1; i++)
                {
                    vValue = (string)vListaValues[i];

                    if (i != vListaValues.Count-1)
                    {
                        vSQLInserir.Append(vValue + ",");
                    }else {
                        vSQLInserir.Append(vValue);
                    }
                }

                 vSQLInserir.Append(")");

                 return vSQLInserir.ToString();

            }
            catch (Exception ex)
            {
                throw (ex);
            }
        }

        private string getSQLAlterar(VO pVo)
        {
            StringBuilder vSQLFinal = new StringBuilder();
            StringBuilder vSQLAlterar = new StringBuilder(" UPDATE ");
            MemberInfo[] vMembers;
            CustomAttributePersistence vCustom;
            object[] vObj;
            //Int32 idx = 0;

            try
            {

                 //pega o atributo com nome da tabela
                vCustom =
(CustomAttributePersistence)Attribute.GetCustomAttribute(pVo.GetType()
, typeof(CustomAttributePersistence));
```

```
                vSQLAlterar.Append(vCustom.NomeTabela + " SET ");

                vMembers = pVo.GetType().GetMembers();

                foreach (MemberInfo  vMem in vMembers)
                {
                    if (vMem.MemberType == MemberTypes.Property)
                    {
                        vObj = vMem.GetCustomAttributes(true);

                        if ((vObj != null) && (vObj.Length > 0))
                        {
                            vCustom =
(CustomAttributePersistence)vObj[0];

                            if (!vCustom.IsPrimaryKey)
                            {
                                //mudanca oracle.

//vSQLAlterar.Append(vCustom.NomeAtributoTabela + "= ?" +
vCustom.NomeAtributoTabela + ",");

vSQLAlterar.Append(vCustom.NomeAtributoTabela + "= " +
vDBParameterValue + vCustom.NomeAtributoTabela + ",");
                            }
                            else
                            {
                                //vSQLFinal.Append(" WHERE " +
vCustom.NomeAtributoTabela + "= ?" + vCustom.NomeAtributoTabela);
                                vSQLFinal.Append(" WHERE " +
vCustom.NomeAtributoTabela + "= " + vDBParameterValue +
vCustom.NomeAtributoTabela);
                            }


                        }
                    }
                }

                //retira a ultima virgula e substitui por fecha
parenteses
                vSQLAlterar.Remove(vSQLAlterar.Length - 1, 1);
                vSQLAlterar.Append(vSQLFinal.ToString());

                return vSQLAlterar.ToString();
            }
            catch (Exception ex)
            {

                throw(ex);
            }
        }

        private string getSQLExcluir(VO pVo)
        {
            StringBuilder vSQLExcluir = new StringBuilder(" DELETE
FROM ");
            CustomAttributePersistence vCustom;

            try
            {
```

```
                vCustom =
(CustomAttributePersistence)Attribute.GetCustomAttribute(pVo.GetType()
, typeof(CustomAttributePersistence));
                vSQLExcluir.Append(vCustom.NomeTabela + " WHERE Id = "
+ vDBParameterValue + "Id ");

                return vSQLExcluir.ToString();
            }
            catch (Exception ex)
            {

                throw(ex);
            }
        }

        private string getSQLDosAtributos(VO pVo)
        {
            StringBuilder vSQLSelect = new StringBuilder(" SELECT ");
            MemberInfo[] vMembers;
            CustomAttributePersistence vCustom;
            object[] vObj;
            Int32 idx = 0;
            Int32 idxOrderByAttributes = 0;

            try
            {
                vHashIndicesAtributos = new Hashtable();
                vMembers = pVo.GetType().GetMembers();

                foreach (MemberInfo  vMem in vMembers )
                {
                    if (vMem.MemberType == MemberTypes.Property)
                    {
                        vObj = vMem.GetCustomAttributes(true);

                        if ((vObj != null)&&(vObj.Length>0))
                        {
                            vCustom =
(CustomAttributePersistence)vObj[0];


vHashIndicesAtributos.Add(vCustom.NomeAtributoTabela , idx);

vSQLSelect.Append(vCustom.NomeAtributoTabela + ",");

                            if (vCustom.IsOrderByAtributte)
                            {
                              if (idxOrderByAttributes == 0)
                              {
                                orderByClause.Append(" ORDER BY " +
vCustom.NomeAtributoTabela + ",");
                              }
                              else
                              {

orderByClause.Append(vCustom.NomeAtributoTabela + ",");
                              }

                              idxOrderByAttributes += 1;

                            }
```

```
                            idx += 1;
                        }
                    }
                }

                //retira a ultima virgula e substitui por fecha
parenteses
                vSQLSelect.Remove(vSQLSelect.Length - 1, 1);

                //retira a ultima virgula da clausula order by
                if (orderByClause.Length > 0)
                  orderByClause.Remove(orderByClause.Length - 1, 1);

                //pega o atributo com nome da tabela
                vCustom =
(CustomAttributePersistence)Attribute.GetCustomAttribute(pVo.GetType()
, typeof(CustomAttributePersistence));
                vSQLSelect.Append(" FROM " + vCustom.NomeTabela);

                return vSQLSelect.ToString();
            }
            catch (Exception ex)
            {

                throw(ex);
            }
        }

        public void gravar(VO pVo)
        {
            try
            {
                if (pVo != null)
                {
                    if (pVo.Id != 0)
                    {
                        alterar(ref pVo);
                    }
                    else
                    {
                        inserir(ref pVo);
                    }
                }
            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
            }
            catch (Exception ex)
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
        }

        private void montaCommandParameters(Boolean pIsInsert, ref VO
pVO, ref IDbCommand pComm)
        {
            MemberInfo[] vMembers;
            CustomAttributePersistence vCustom;
            object[] vObj;
            IDbDataParameter param;
```

```
            try
            {
                vMembers = pVO.GetType().GetMembers();

                foreach (MemberInfo vMem in vMembers)
                {
                    if (vMem.MemberType == MemberTypes.Property)
                    {
                        vObj = vMem.GetCustomAttributes(true);

                        if ((vObj != null) && (vObj.Length > 0))
                        {
                            param =
FactoryConnection.getInstance().getLastConnectionSet().getNewParameter
();
                            vCustom =
(CustomAttributePersistence)vObj[0];
                            //mudanca pro oracle.
                            //param.ParameterName = "?" +
vCustom.NomeAtributoTabela;
                            param.ParameterName = vDBParameterValue +
vCustom.NomeAtributoTabela;
                            param.DbType = vCustom.TipoAtributoTabela;

                            if (vCustom.IsPrimaryKey)
                            {
                                //verifica se e para inserir, se for
tem que verificar e o Id e identity se nao for e alteracao
                                //entao nao tem problema
                                if (pIsInsert)
                                {
                                    if (!vCustom.IsIdentity)
                                    {
                                        param.Value =
this.convert(((PropertyInfo)vMem).GetValue(pVO, null));
                                    }
                                }
                                else
                                {
                                    param.Value =
this.convert(((PropertyInfo)vMem).GetValue(pVO, null));
                                }

                            }
                            else if (vCustom.IsLink)
                            {
                                VO vLink =
(VO)((PropertyInfo)vMem).GetValue(pVO, null);

                                param.Value = convertLink(vLink);
                                //pComm.Parameters.Add("?" +
vCustom.NomeAtributoTabela) = convertLink(vLink);

                                //pComm.Parameters.AddWithValue("?" +
vCustom.NomeAtributoTabela, convertLink(vLink));
                            }
                            else
                            {
                                param.Value =
this.convert(((PropertyInfo)vMem).GetValue(pVO, null));
```

```
                                //pComm.Parameters.Add("?" +
vCustom.NomeAtributoTabela).Value =
this.convert(((PropertyInfo)vMem).GetValue(pVO, null));
                                //pComm.Parameters.AddWithValue("?" +
vCustom.NomeAtributoTabela, convert(((PropertyInfo)vMem).GetValue(pVO,
null)));
                            }

                            pComm.Parameters.Add(param);
                        }
                    }
                }
            }
            catch (Exception)
            {

                throw;
            }

        }

        private ArrayList getLista(DbDataReader pDR, Type pTipoVo)
        {
            MemberInfo[] vMembers;
            CustomAttributePersistence vCustom;
            object[] vObj;
            VO vVO;
            ArrayList vLista = null;

            try
            {

                if (pDR.HasRows)
                {
                    vLista = new ArrayList();

                    while (pDR.Read())
                    {
                        vVO =
(VO)System.Activator.CreateInstance(pTipoVo);
                        vMembers = vVO.GetType().GetMembers();

                        foreach (MemberInfo  vMem in vMembers)
                        {
                            if (vMem.MemberType ==
MemberTypes.Property)
                            {
                                vObj = vMem.GetCustomAttributes(true);

                                if ((vObj != null) && (vObj.Length >
0))
                                {
                                    vCustom =
(CustomAttributePersistence)vObj[0];

                                    if (vCustom.IsLink)
                                    {
                                        VO vLinkVo =
(VO)System.Activator.CreateInstance(vCustom.TipoVo);
```

```
                                        Object resultadoConvert =
convert(pDR.GetValue((Int32)vHashIndicesAtributos[vCustom.NomeAtributo
Tabela]));

                                        if (resultadoConvert == null)
                                            vLinkVo = null;
                                        else
                                            //TODO : verificar se
vale.
                                            //vLinkVo.Id =
(int)resultadoConvert;
                                            vLinkVo.Id =
Int32.Parse(resultadoConvert.ToString());


((PropertyInfo)vMem).SetValue(vVO, vLinkVo, null);
                                    }
                                    else
                                    {
                                        if (vCustom.TipoAtributoTabela
== System.Data.DbType.Boolean)
                                          //teste com boolean.parse

//((PropertyInfo)vMem).SetValue(vVO,
convert(pDR.GetBoolean((Int32)vHashIndicesAtributos[vCustom.NomeAtribu
toTabela])), null);

((PropertyInfo)vMem).SetValue(vVO, convert(Boolean.Parse
(pDR.GetValue((Int32)vHashIndicesAtributos[vCustom.NomeAtributoTabela]
).ToString())), null);

                                            //TODO : teste p oracle
                                        else
if(vCustom.TipoAtributoTabela == DbType.Int32)

((PropertyInfo)vMem).SetValue(vVO,
convert(Int32.Parse(pDR.GetValue((Int32)vHashIndicesAtributos[vCustom.
NomeAtributoTabela]).ToString())), null);
                                        else

((PropertyInfo)vMem).SetValue(vVO,
convert(pDR.GetValue((Int32)vHashIndicesAtributos[vCustom.NomeAtributo
Tabela])), null);
                                    }
                                }
                            }
                        }

                        vLista.Add(vVO);
                    }
                }

                return vLista;

            }
            catch (Exception)
            {

                throw;
            }
        }
```

```
        public void inserir(ref VO pVO)
        {
            IDbCommand vComm = null;
            DbDataReader vDR = null;

            try
            {
                vComm = this.criaCommand();
                // Monta o SQL especifico para tipo especifico de
Objeto recebido
                vComm.CommandText = getSQLIncluir(pVO);

                montaCommandParameters(true, ref pVO, ref vComm);

                vComm.ExecuteNonQuery();//executa o comando no banco
de dados

            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
            }
            catch (Exception ex)
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
            finally
            {
                if (vDR != null)
                {
                    vDR.Close();//fecha a transação
                    vDR.Dispose();
                }
                fechaConnexao();//fecha a conexão com o banco
            }
        }

        public void alterar(ref VO pVO)
        {
            IDbCommand vComm = null;
            DbDataReader vDR = null;

            try
            {
                vComm = this.criaCommand();
                vComm.CommandText = getSQLAlterar(pVO);

                montaCommandParameters(false, ref pVO, ref vComm);

                vDR = (DbDataReader)vComm.ExecuteReader();
            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
            }
            catch (Exception ex)
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
            finally
```

```
            {

                if (vDR != null)
                {
                    vDR.Close();
                    vDR.Dispose();
                }
                fechaConnexao();
            }
        }

        public void excluir(VO pVO)
        {
            IDbCommand vComm = null;
            IDbDataParameter param = null;

            try
            {
                vComm = this.criaCommand();
                vComm.CommandText = getSQLExcluir(pVO);

                param =
FactoryConnection.getInstance().getLastConnectionSet().getNewParameter
();

                param.ParameterName = vDBParameterValue + "Id";
                param.DbType = System.Data.DbType.Int32;
                param.Value = pVO.Id;

                vComm.Parameters.Add(param);

                vComm.ExecuteNonQuery();
            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
            }
            catch (Exception ex)
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
            finally
            {
                fechaConnexao();
                vComm.Dispose();
            }
        }

        public ArrayList consultar(Type pTipoVo)
        {
            IDbCommand vComm = null;
            DbDataReader vDR = null;
            ArrayList vLista = null;
            VO vVO = null;

            try
            {
                vVO = (VO)System.Activator.CreateInstance(pTipoVo);

                vComm = this.criaCommand(); ;
```

```
                vComm.CommandText = getSQLDosAtributos(vVO) +
orderByClause.ToString();

                vDR = (DbDataReader)vComm.ExecuteReader();

                vLista = getLista(vDR, pTipoVo);

                return vLista;
            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
            }
            catch (Exception ex)
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
            finally
            {
                if (vDR != null)
                {
                    vDR.Close();
                    vDR.Dispose();
                }
                fechaConnexao();

            }
        }

        public ArrayList consultarSQL(Type pTipoVo, string vSQL,
ArrayList listaParametros)
        {
            IDbCommand vComm = null;
            DbDataReader vDR = null;
            ArrayList vLista = null;
            VO vVO = null;

            //prov
            string vSQLProv;

            try
            {
                vVO = (VO)System.Activator.CreateInstance(pTipoVo);

                vComm = this.criaCommand();
                vSQLProv = getSQLDosAtributos(vVO);
                vComm.CommandText = vSQL;

                carregaParametros(ref vComm, listaParametros);

                vDR = (DbDataReader)vComm.ExecuteReader();

                vLista = getLista(vDR, pTipoVo);

                return vLista;
            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
            }
            catch (Exception ex)
```

```
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
            finally
            {

                if (vDR != null)
                {
                    vDR.Close();
                    vDR.Dispose();
                }
                fechaConnexao();
            }
        }

        private void carregaParametros(ref IDbCommand vCommand,
ArrayList listaParametros)
        {
            try
            {
                foreach (object vParam in listaParametros)
                {
                    vCommand.Parameters.Add((DbParameter)vParam);
                }

            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
            }
            catch (Exception ex)
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
        }

        public VO consultar(VO pVO)
        {
            IDbCommand vComm = null;
            DbDataReader vDR = null;
            ArrayList vLista = null;
            VO vVO = null;

            try
            {
                vComm = this.criaCommand();
                vComm.CommandText = getSQLDosAtributos(pVO) + " WHERE
Id = " + pVO.Id;

                vDR = (DbDataReader)vComm.ExecuteReader();

                vLista = getLista(vDR, pVO.GetType());

                if ((vLista != null) && (vLista.Count > 0))
                    vVO = (VO)vLista[0];

                return vVO;
            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
```

```
            }
            catch (Exception ex)
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
            finally
            {
                if (vDR != null)
                {
                    vDR.Close();
                    vDR.Dispose();
                }
                fechaConnexao();


            }
        }

        public ArrayList pesquisaExpressao(Type pTipoVo, string
pExpressao)
        {
            IDbCommand vComm = null;
            DbDataReader vDR = null;
            ArrayList vLista = null;
            VO vVO = null;

            try
            {
               vVO = (VO)System.Activator.CreateInstance(pTipoVo);

                vComm = this.criaCommand();
                vComm.CommandText = getSQLDosAtributos(vVO) + " WHERE
" + pExpressao + orderByClause.ToString();

                vDR = (DbDataReader)vComm.ExecuteReader();

                vLista = getLista(vDR, pTipoVo);

                return vLista;
            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
            }
            catch (Exception ex)
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
            finally
            {

                if (vDR != null)
                {
                    vDR.Close();
                    vDR.Dispose();
                }
                fechaConnexao();
            }
        }
```

```
        public DataSet pesquisaExpressaoDS(Type pTipoVo, string
pExpressao)
        {
            IDbCommand vComm = null;
            IDbDataAdapter vDA = null;
            DataSet vDS = new DataSet();
            VO vVO = null;

            try
            {
                vVO = (VO)System.Activator.CreateInstance(pTipoVo);

                vComm = this.criaCommand();
                vComm.CommandText = getSQLDosAtributos(vVO) + " WHERE
" + pExpressao + orderByClause.ToString();

                vDA =
((IConnection)FactoryConnection.getInstance().getLastConnectionSet()).
getNewDataAdapter();

                vDA.SelectCommand = vComm;

                vDA.Fill(vDS) ;

                return vDS;
            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
            }
            catch (Exception ex)
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
            finally
            {
                fechaConnexao();
                if (vDA != null)
                {
                    vDA = null;
                }

            }
        }

        public DataSet consultarDS(Type pTipoVo)
        {
          IDbCommand vComm = null;
          IDbDataAdapter vDA = null;
          DataSet vDS = new DataSet();
          VO vVO = null;

          try
          {
            vVO = (VO)System.Activator.CreateInstance(pTipoVo);

            vComm = this.criaCommand();
            vComm.CommandText = getSQLDosAtributos(vVO) +
orderByClause.ToString();
```

```
            vDA =
((IConnection)FactoryConnection.getInstance().getLastConnectionSet()).
getNewDataAdapter();

            vDA.SelectCommand = vComm;

            vDA.Fill(vDS);

            return vDS;
          }
          catch (ExceptionAbstract ex)
          {
            throw (ex);
          }
          catch (Exception ex)
          {
            throw new ExceptionFramework(ex.Message, ex);
          }
          finally
          {
            fechaConnexao();
            if (vDA != null)
            {
              vDA = null;
            }

          }
        }


        public int pesquisaUltimoID(Type pTipoVo)
        {
            IDbCommand vComm = null;
            DbDataReader vDR = null;
            ArrayList vLista = null;
            VO vVO = null;
            CustomAttributePersistence vCustom;
            StringBuilder vSQLSelect;

            try
            {
                vVO = (VO)System.Activator.CreateInstance(pTipoVo);

                vSQLSelect = new StringBuilder(" SELECT MAX(Id) FROM
");

                //pega o atributo com nome da tabela
                vCustom =
(CustomAttributePersistence)Attribute.GetCustomAttribute(vVO.GetType()
, typeof(CustomAttributePersistence));
                vSQLSelect.Append(vCustom.NomeTabela);

                vComm = this.criaCommand();
                vComm.CommandText = vSQLSelect.ToString();

                vDR = (DbDataReader)vComm.ExecuteReader();

                vDR.Read();
                string resultadoConvert = String.Empty;

                if (vDR.IsDBNull(0))
```

```
                    resultadoConvert = "1";
                else
                    resultadoConvert =
convert(vDR.GetValue(0)).ToString();

                return Int32.Parse(resultadoConvert);
            }
            catch (ExceptionAbstract ex)
            {
                throw (ex);
            }
            catch (Exception ex)
            {
                throw new ExceptionFramework(ex.Message, ex);
            }
            finally
            {

                if (vDR != null)
                {
                    vDR.Close();
                    vDR.Dispose();
                }
                fechaConnexao();
            }
        }

    }
}
```

APÊNDICE B – BIBLIOTECA TIMER

Cabeçalho da Biblioteca de Timer

```
#ifndef MsTimer2_h
#define MsTimer2_h

#include <avr/interrupt.h>

namespace MsTimer2 {
	extern unsigned long msecs;
	extern void (*func)();
	extern volatile unsigned long count;
	extern volatile char overflowing;
	extern volatile unsigned int tcnt2;

	void set(unsigned long ms, void (*f)());
	void start();
	void stop();
	void _overflow();
}

#endif
```

APÊNDICE C – BIBLIOTECA DE ACESSO AO CARTÃO SD

Cabeçalho da Biblioteca de acesso ao Cartão SD.

```
/* Arduino FAT16 Library
 * Copyright (C) 2008 by William Greiman
 *
 * This file is part of the Arduino FAT16 Library
 *
 *  This  Library  is  free  software:  you  can  redistribute  it  and/or
modify
 * it under the terms of the GNU General Public License as published
by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This Library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.

 * You should have received a copy of the GNU General Public License
 * along with the Arduino Fat16 Library.  If not, see
 * <http://www.gnu.org/licenses/>.
 */
#ifndef Fat16_h
#define Fat16_h
#include <string.h>
#include <avr/pgmspace.h>
#include "SdCard.h"
#include "Print.h"
#include "FatStructs.h"
#include "Fat16Config.h"
/**
 * \typedef fat_t
 *
 * \brief Type for FAT16 entry
 */
typedef uint16_t fat_t;
/**
 * \union cache16_t
 *
 * \brief Cache buffer data type
 *
 */
union cache16_t {
          /** Used to access cached file data blocks. */
  uint8_t data[512];
          /** Used to access cached FAT entries. */
  fat_t   fat[256];
          /** Used to access cached directory entries. */
  dir_t   dir[16];
          /** Used to access a cached Master Boot Record. */
  mbr_t   mbr;
          /** Used to access to a cached FAT16 boot sector. */
```

```
  fbs_t   fbs;
};
// use the gnu style oflags
/** open for reading */
#define O_READ    0X01
/** same as O_READ */
#define O_RDONLY  O_READ
/** open for write */
#define O_WRITE   0X02
/** same as O_WRITE */
#define O_WRONLY  O_WRITE
/** open for reading and writing */
#define O_RDWR    (O_READ | O_WRITE)
/** mask for access modes */
#define O_ACCMODE (O_READ | O_WRITE)
/** The file offset shall be set to the end of the file prior to each
write. */
#define O_APPEND  0X04
/** synchronous writes - call sync() after each write */
#define O_SYNC    0X08
/** create the file if nonexistent */
#define O_CREAT   0X10
/**  If  O_CREAT  and  O_EXCL  are  set,  open()  shall  fail  if  the  file
exists */
#define O_EXCL    0X20
/* truncate the file to zero length */
#define O_TRUNC   0X40
//------------------------------------------------------------------------
----------
/** \class Fat16
 * \brief Fat16 implements a minimal Arduino FAT16 Library
 *
 * Fat16 does not support subdirectories or long file names.
 */
#if FAT16_PRINT_SUPPORT
class Fat16 : public Print {
#else // FAT16_PRINT_SUPPORT
class Fat16 {
#endif //FAT16_PRINT_SUPPORT
  // Volume info
  static  uint8_t    volumeInitialized_;    //true  if  volume  has  been
initialized
  static uint8_t  fatCount_;           //number of FATs
  static  uint8_t   blocksPerCluster_;     //must  be  power  of  2,  max
cluster size 32K
  static uint16_t rootDirEntryCount_;  //should be 512 for FAT16
  static fat_t    blocksPerFat_;       //number of blocks in one FAT
  static fat_t    clusterCount_;       //total clusters in volume
  static uint32_t fatStartBlock_;      //start of first FAT
  static uint32_t rootDirStartBlock_;  //start of root dir
  static uint32_t dataStartBlock_;     //start of data clusters
  //block cache
#define CACHE_FOR_READ   0           //cache a block for read
#define CACHE_FOR_WRITE  1           //cache a block and set dirty for
the block
  static SdCard *rawDev_;             // Device
  static  cache16_t  cacheBuffer_;       //512  byte  cache  for  storage
device blocks
  static uint32_t cacheBlockNumber_; //Logical number of block in the
cache
```

```
  static uint8_t cacheDirty_;          //cacheFlush() will write block
if true
  static uint32_t cacheMirrorBlock_ ;// mirror block for second FAT
  // define fields in flags_
#define F_OFLAG (O_ACCMODE | O_APPEND | O_SYNC ) // should be 0XF
#define F_FILE_SIZE_DIRTY 0X80 // sync of directory entry file size
required
  uint8_t flags_;         // see above for bit definitions
  int16_t dirEntryIndex_; // index of directory entry for open file
  fat_t firstCluster_;    // first cluster of file
  uint32_t fileSize_;     // fileSize
  fat_t curCluster_;      // current cluster
  uint32_t curPosition_;  // current byte offset
  //private functions for cache
  static uint8_t blockOfCluster(uint32_t position) {
    // depends on blocks per cluster being power of two
    return (position >> 9) & (blocksPerCluster_ - 1);}
  static uint16_t cacheDataOffset(uint32_t position) {return position
& 0X1FF;}
  static dir_t *cacheDirEntry(uint16_t index, uint8_t action = 0);
  static uint8_t cacheRawBlock(uint32_t blockNumber, uint8_t action =
0);
  static uint8_t cacheFlush(void);
  static void cacheSetDirty(void) {cacheDirty_ |= CACHE_FOR_WRITE;}
  static uint32_t dataBlockLba(fat_t cluster, uint8_t blockOfCluster)
{
       return     dataStartBlock_     +     (uint32_t)(cluster     -
2)*blocksPerCluster_
       + blockOfCluster;}
  static uint8_t fatGet(fat_t cluster, fat_t &value);
  static uint8_t fatPut(fat_t cluster, fat_t value);
  //find and cache a directory entry
  static dir_t *findDirEntry(uint16_t &entry, uint8_t *name,
                             uint8_t skip = DIR_ATT_VOLUME_ID);
  //end of chain test
  static uint8_t isEOC(fat_t cluster) {return cluster >= 0XFFF8;}
  //allocate a cluster to a file
  uint8_t addCluster(void);
  //free a cluster chain
  uint8_t freeChain(fat_t cluster);
//------------------------------------------------------------------------
public:
  /*
   * Public functions
   */
  // create with file closed
  Fat16(void) : flags_(0) {}
  /** \return The current cluster number. */
  fat_t curCluster(void) {return curCluster_;}
  uint8_t close(void);
  /** \return The count of clusters in the FAT16 volume. */
  static fat_t clusterCount(void) {return clusterCount_;}
  /** \return The number of 512 byte blocks in a cluster */
  static uint8_t clusterSize(void) {return blocksPerCluster_;}
  /** \return The current file position. */
  uint32_t curPosition(void) {return curPosition_;}
  /** \return The file's size in bytes. */
  uint32_t fileSize(void) {return fileSize_;}
  static uint8_t init(SdCard &dev, uint8_t part);
  /**
   *  Initialize a FAT16 volume.
```

```
   *  First try partition 1 then try super floppy format.
   * \param[in] dev The SdCard where the volume is located.
   * \return The value one, true, is returned for success and
   * the value zero, false, is returned for failure.  reasons for
   * failure include not finding a valid FAT16 file system, a call
   * to init() after a volume has been successful initialized or
   * an I/O error.
   */
  static uint8_t init(SdCard &dev) {
  if (init(dev, 1)) return true; return init(dev, 0);}
  /**
   * Checks the file's open/closed status for this instance of Fat16.
   * \return The value true if a file is open otherwise false;
  uint8_t isOpen(void) {return (flags_ & O_ACCMODE) != 0;}
  uint8_t open(const char *fileName, uint8_t oflag);
  uint8_t open(uint16_t entry, uint8_t oflag);
  int16_t read(void);
  int16_t read(void *buf, uint16_t nbyte);
  static uint8_t readDir(dir_t &dir, uint16_t &index,
                    uint8_t     skip     =     (DIR_ATT_VOLUME_ID     |
DIR_ATT_DIRECTORY));
  static dir_t *readDir(uint16_t &index,
                    uint8_t     skip     =     (DIR_ATT_VOLUME_ID     |
DIR_ATT_DIRECTORY));
  /** \return The number of entries in the root directory. */
  static uint16_t rootDirEntryCount(void) {return rootDirEntryCount_;}
  uint8_t remove(void);
  /**     Seek   to   current   position   plus   \a   pos   bytes.   See
Fat16::seekSet(). */
  uint8_t seekCur(uint32_t pos) {return seekSet(curPosition_ + pos);}
  /** Seek to end of file.  See Fat16::seekSet(). */
  uint8_t seekEnd(void) {return seekSet(fileSize_);}
  uint8_t seekSet(uint32_t pos);
  uint8_t sync(void);
  uint8_t truncate(uint32_t size);
  /** Fat16::writeError is set to true if an error occurs during a
write().
   *  Set  Fat16::writeError  to  false  before  calling  print()  and/or
write() and check
   * for true after calls to write() and/or print().
   */
  bool writeError;
  int16_t write(const void *buf, uint16_t nbyte);
  void write(uint8_t b);
  void write(const char *str);
  void write_P(PGM_P str);
  void writeln_P(PGM_P str);
//------------------------------------------------------------------------------
#if FAT16_DEBUG_SUPPORT
  /** For debug only.  Do not use in applications. */
  static cache16_t *dbgBufAdd(void) {return &cacheBuffer_;}
  /** For debug only.  Do not use in applications. */
  static void dbgSetDev(SdCard &dev) {rawDev_ = &dev;}
  /** For debug only.  Do not use in applications. */
  static uint8_t *dbgCacheBlock(uint32_t blockNumber) {
    return cacheRawBlock(blockNumber) ? cacheBuffer_.data : 0; }
  /** For debug only.  Do not use in applications. */
  static dir_t *dbgCacheDir(uint16_t index) {
    return cacheDirEntry(index);}
#endif //FAT16_DEBUG_SUPPORT
};
```

```
#endif//Fat16_h
```

APÊNDICE D – CÓDIGO ARDUINO

Código fonte desenvolvido no Arduino.

```
#include <Fat16.h>
#include <Fat16util.h>
#include <MsTimer2.h>

//Variaveis Globais de Arquivo
SdCard cartao;
Fat16 arquivo;
char name[] = "TESTE00.TXT";

//Pinos de aquisiçao dos sensores
int AN_AC_X = 0;
int AN_AC_Y = 1;
int AN_TEMP = 2;

//Variaveis globais de controle e dados
int fezCaptura = 0;
int fezVolta = 0;
int iniciou = 0;
int diff = 0;
int milli = 0;
int state = 0;

//Função de erro: anuncia pela seria e fica travado no while(1)
#define error(s) error_P(PSTR(s))
void error_P(const char *str)
{
  PgmPrint("ERRO: ");
  SerialPrintln_P(str);
  while(1);
}

//Funcao de configuraçao do arduino
void setup() {
  pinMode(13, OUTPUT);

  //Abre comunicaçao serial a 19200 bPS
  Serial.begin(19200);

  //Configura a interrupçao do sensor de volta
  attachInterrupt(0, FazVolta, RISING);

  //Inicia Timer de controle de aquisicao de dados dos sensores: 100
em 100 mili segundos
  MsTimer2::set(100,FazCaptura);
  MsTimer2::start();

  //Cria o arquivo com nome variavel para nao sobrescrever arquivos
existentes
  for (uint8_t i = 0; i < 100; i++) {
    name[5] = i/10 + '0';
    name[6] = i%10 + '0';
    if (arquivo.open(name, O_CREAT | O_EXCL | O_WRITE)) break;
  }
  if (!arquivo.isOpen()) error ("create");
```

```
  PgmPrint("Utilziando Arquivo: ");
  Serial.println(name);
 }

void loop() {
  //Muda o led do pino 13 de estado, para indicar passagem por fita
  digitalWrite(13,state);

  //Controla a passagem por fita
  if (fezVolta == 1){
    fezVolta = 0;

    //se passou pela 1a vez, abre o monitoramento
    if (iniciou == 0){
      iniciou = 1;
      milli = millis();
    //Senao, ja insere no arquivo o fato de ter passado.
    }else{
      diff = millis() - milli;
      milli = millis();
      Serial.print("1;");
      Serial.print(diff);
      Serial.println(";");

      //Escreve dados de volta no arquivo
      arquivo.print("1;");
      arquivo.print(diff, DEC);
      arquivo.println(";");

      //Esvazia o buffer para o arquivo em toda passagem
      arquivo.sync();
    }
  }

    //Se ja iniciou o evento, captura os dados de sensores.
    if ((fezCaptura == 1)&&(iniciou==1)){
        fezCaptura = 0;
        diff = millis() - milli;
        milli = millis();

        Serial.print("2;");
        Serial.print(diff);
        Serial.print(";");
        Serial.print(analogRead(AN_AC_X));
        Serial.print(";");
        Serial.print(analogRead(AN_AC_Y));
        Serial.print(";");
        Serial.print(analogRead(AN_TEMP));
        Serial.println(";");

        //Escreve dados de sensores
        arquivo.print("2;");
        arquivo.print(diff);
        arquivo.print(";");
        arquivo.print(analogRead(AN_AC_X),DEC);
        arquivo.print(";");
        arquivo.print(analogRead(AN_AC_Y), DEC);
        arquivo.print(";");
        arquivo.print(analogRead(AN_TEMP), DEC);
        arquivo.println(";");
```

```
    }
 }

// Funcao para indicar a passagem por fita
void FazVolta(){
  fezVolta = 1;
  state = !state;
}

//Funcao para indicar que o tempo de captura encerrou e
//os dados devem ser adicionados.
void FazCaptura(){
  fezCaptura = 1;
}
```

APÊNDICE D – PROGRAMA DE GRAVAÇÃO DO ARQUIVO TEXTO VIA USB

```
namespace LeKartComp
{
      class Program
      {
            static void Main(string[] args)
            {
                  SerialPort sp = new SerialPort("COM4", 19200);
                  StreamWriter sw = new
StreamWriter(Environment.GetFolderPath(Environment.SpecialFolder.Deskt
op) +
                "\\KartComp.txt");

                  string linha;
            string[] param;
                  sp.ReadTimeout = 20000;
            sp.DtrEnable = true;
                  sp.Open();
            Thread.Sleep(2000);
            Console.WriteLine("Pronto");
            sp.WriteLine("C");

                  while (Console.KeyAvailable != true){
                        linha = sp.ReadLine();
                param = linha.Split(';');
                if (param[0] == "2")
                {
                    Console.WriteLine("Ac X: " +
((float.Parse(param[2]) * 5 / 1023 - 2.5) / 0.312).ToString("0.00") +
                        " G\tAc Y: " + ((float.Parse(param[3]) * 5 /
1024 - 2.5) / 0.312).ToString("0.00") +
                        " G\tTemp: " + float.Parse(param[4]) * 500 /
1023);
                }
                else
                {
                    Console.WriteLine(linha);
                }
                        sw.WriteLine(linha);
                  }
```

```
                    sp.Close();
                    sw.Close();


                }
            }
        }
```