CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA

FATEC SANTO ANDRÉ

Tecnologia em Eletrônica Automotiva

ELIZEU MANOEL DA SILVA

RICARDO MENDES SOARES

RODRIGO TAKESHI FUJIMOTO

SOFTWARE DE DESENVOLVIMENTO PARA CLUSTER AUTOMOTIVO

Santo André – São Paulo

2014

CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA

FATEC SANTO ANDRÉ

Tecnologia em Eletrônica Automotiva

ELIZEU MANOEL DA SILVA

RICARDO MENDES SOARES

RODRIGO TAKESHI FUJIMOTO

SOFTWARE DE DESENVOLVIMENTO PARA CLUSTER AUTOMOTIVO

Monografia apresentada ao Curso de Tecnologia em Eletrônica Automotiva da FATEC Santo André, como requisito parcial para conclusão do curso em Tecnologia em Eletrônica Automotiva.

Orientador: Professor Weslley Medeiros Torres.

Santo André – São Paulo

2014

Fatec CENTRO PAULA SOUZA GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Santo André

LISTA DE PRESENÇA

SANTO ANDRÉ, 24 DE NOVEMBRO DE 2014.

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA **"SOFTWARE DE DESENVOLVIMENTO PARA CLUSTER AUTOMOTIVO"** DOS ALUNOS DO 6° SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. WESLLEY MEDEIROS TORRES _____

MEMBROS:
PROF. KLEBER NOGUEIRA HODEL _____

TECNÓLOGO SRº. MARCO ANTONIO DE CARVALHO GUEDES _____

ALUNO:

ELIZEU MANOEL DA SILVA _____

RICARDO MENDES SOARES _____

RODRIGO TAKESHI FUJIMOTO _____

Fujimoto, Rodrigo Takeshi
Ferramenta para o desenvolvimento de cluster automotivo / Rodrigo Takeshi Fujimoto, Elizeu Manoel Silva, Ricardo Mendes Soares. - Santo André,2014. – f: 120 il.

Trabalho de conclusão de curso - FATEC- Santo André.
Curso de Eletrônica Automotiva, 2014.
Orientador: Weslley Medeiros Torres

1.Cluster 2. LCD 3. Customização 4. Microcontrolador 5. Software
I. Silva, Elizeu Manoel II. Soares, Ricardo Mendes

Dedicamos este projeto a todos os auto entusiastas e todas as pessoas que buscam conhecimento.

AGRADECIMENTOS

A todos os docentes que nos trouxeram conhecimento no decorrer do curso e aqueles que apoiaram e ajudaram no desenvolvimento deste projeto.

RESUMO

O uso de *displays* de tela colorida pode ser encontrado em diversos dispositivos, e aos poucos está sendo aderida ao mercado automotivo, como é o caso dos painéis de instrumentos customizáveis (C*lusters*), que já são realidade em veículos de luxo. Com a finalidade de facilitar o desenvolvimento gráfico destes painéis e incentivar o seu crescimento no mercado automotivo, este projeto consiste na criação de um *software* para criação e simulação de composições gráficas para o *cluster*, com configurações de cor, posição, tamanho entre outras características. O software é desenvolvido em linguagem C# (ambiente Microsoft Visual Studio e Atmel Studio) e simulado em uma plataforma de teste (*Hardware de Simulação*). Com kernel pré carregado, arquitetura baseada no controlador Atmel SAM3X8E e *display* colorido de 5 polegadas com controlador da *Itead Studio*, possibilita o desenvolvimento total em laboratório, sem necessidade de veículo como protótipo e reduzir tempo e custos com programação.

Palavras-chaves: cluster, LCD, customização, microcontrolador, software.

ABSTRACT

The use of color screen displays can be found in many devices, and gradually is being adhered to the automotive market, as is the case with customizable dashboards (clusters), which are a reality in luxury vehicles. In order to facilitate the development of these graphic panels and encourage its growth in the automotive market, this project is based in a creation of a software for creating and simulate graphic compositions for the cluster, with color configurations, position, size among other features. The software is developed in C # (Microsoft Visual Studio and Atmel Studio environment) and simulated on a test platform (hardware simulation). With preloaded kernel, architecture based on Atmel controller SAM3X8E and 5 inch color display with Itead Studio controller, enables the full development in laboratory without need a vehicle as a prototype and reduce programming time and costs.

Keywords: cluster, LCD, customization, microcontroller, software.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIAÇÕES

ANSI - American National Standarts Institute
API - Application Programming Interface
CAN - Controller Area Network
DMA - Direct Memory Access
GPS - Global Positioning System
GUI - Graphical User Interface
HIL – Hardware In the Loop
HUD - Head Up Display
I/O - Input/Output
I2C - Inter Integrated Circuit
IEC - International Electrotechnical Commission
ISO - International Organization for Standardization
JTAG - Joint Test Action Group
Km/h - quilômetro/hora
LCD - Liquid Crystal Display
Mb - Mega Bit
MISRA - Motor Industry Software Reliability Association
PC – Personal Computer
PWM - Pulse width modulation
RAM - Random Access Memory
RTOS - Real Time Operating System
SD - Secure Digital
SPI - Serial Peripheral Interface
SRAM - Static random-access memory
TFT - Thin Film Transistor
V - Volts
VFD – Vacuum Fluorescent Display

SUMÁRIO

1. OBJETIVO

Este projeto consiste no desenvolvimento de um *software* para criação e simulação de composições gráficas para o *cluster* automotivo, com configurações de cores, posições, tamanhos entre outras características.

Uma vez que esta ferramenta se encontra funcionando e com configuração do hardware de forma automatizada, possibilita o desenvolvimento deste tipo de painéis a um projetista sem o conhecimento em linguagem de programação.

O hardware de testes por outro lado, permite o desenvolvimento total em laboratório, sem necessidade de veículo como protótipo, reduzindo assim tempo de desenvolvimento, custos com programação e riscos ligados ao veículo.

2. MOTIVAÇÃO

A baixa nos preços dos TFTs disponibilizados no mercado, devido à produção em larga escala e a universalização do uso nos mais variados dispositivos, proporcionam maior durabilidade, fácil montagem e manutenção.

Além disso, com um único *hardware* desenvolvido, pode-se atender toda linha de veículos, com a aplicação de máscaras simples para delimitar o formado e tamanho desejado da tela, uma vez que os desenhos são virtuais.

Por outro lado o custo de programação nestes *displays* não é favorável, devido o tempo necessário para programá-lo.

Como solução, a ferramenta gráfica proposta neste projeto visa minimizar as desvantagens com programação, fazendo com que o desenvolvimento destes *clusters* sejam mais rápidos e com menor custo.

3. METODOLOGIA

O desenvolvimento deste projeto, segue uma lógica de construção econômica, onde todos os recursos foram adaptados ou planejados de forma a obter o máximo desempenho com o mínimo investimento.

Os *softwares* utilizados são de procedência gratuita ou *open source* (ambiente Microsoft Visual Studio Express, Atmel Studio e Arduino), contando com ampla assistência a pesquisas online e operando em ambiente Windows, que facilitar a adaptação e o tempo de aprendizado.

O *hardware* de simulação oferece suporte *touch*, eliminando a necessidade de botões e fios, fato que elimina mal contato entre os dispositivos e a comunicação com o computador é realizada através de comunicação serial USB.

O software de desenvolvimento, foi criado em linguagem de alto nível, que possibilita versatilidade e rapidez no desenvolvimento se comparada à programação de baixo nível.

4. INTRODUÇÃO

O *cluster* automotivo é uma das ferramentas de importância nos veículos, sendo o responsável pela *interface* das informações entre o homem e a máquina, permitindo ao condutor obter informações significativas, como velocidade, rotação, quantidade de combustível e temperatura do motor através de indicadores em formato de relógios.

Com o passar do tempo, os veículos atuais passaram a disponibilizar estas informações não mais em instrumentos mecânicos, mas sim em equipamentos totalmente digitais em telas de *display* colorido, porém devido ao seu maior custo de desenvolvimento, estes *displays* somente estão disponíveis em veículos de luxo, segundo o site da fabricante Continental

Porém, estes custos tendem a cair, pois os displays de tela colorida para uso automotivo vêm crescendo (Figura 1), e de acordo com o centro de pesquisa da *IHS*

iSupply, até 2016, o número de *displays* pode chegar a 116,8 milhões, incluindo *cluster* customizáveis, multimídias, GPS, câmeras de ré entre outros.

Previsão de envio de TFT-LCD coloridos para o mercado automotivo mundial (milhões de unidade)

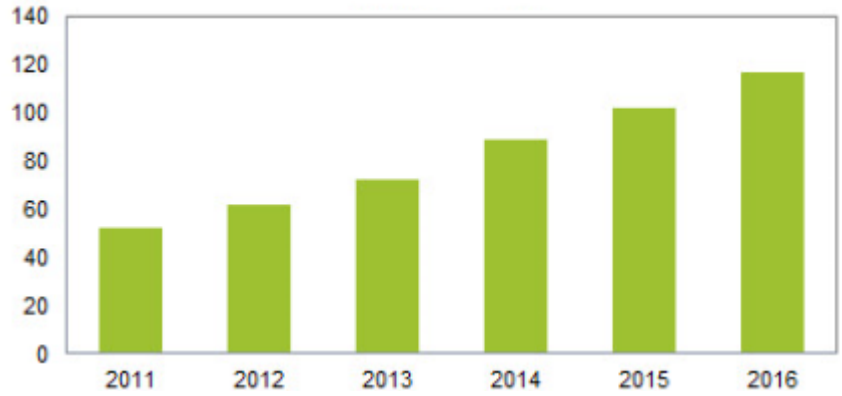

Figura 1 - Previsão de crescimento dos TFT-LCD. (Adaptado de IHS iSuppli Research, 2013)

A partir deste crescimento, o desenvolvimento de painéis de instrumentos customizáveis torna-se um potencial, que de acordo com a *IHS iSupply*, empresas como *Fujitsu* e *Freescale* se tornaram os principais fornecedores de *clusters* digitais com capacidades gráficas avançadas.

Para se obter um melhor resultado no desenvolvimento destes clusters, há a necessidade de um *software* de desenvolvimento gráfico, como é o caso do *software* Altia, que é utilizada em companhias como a *Ford*, *GM* e *FIAT*, por exemplo.

Estes *softwares* auxiliam o desenvolvimento da GUI (*Graphical User Interface* ou Interface Gráfica do Usuário) dos *displays*, diminuindo o tempo de programação e consequentemente o custo, e deixando o instrumento com uma aparência mais agradável ao consumidor.

O objetivo deste projeto é criar uma ferramenta de desenvolvimento gráfico similar ao software da Altia, porém focando em uma ferramenta de baixo custo.

O diagrama de blocos do projeto pode ser visualizada na Figura 2.

A ferramenta desenvolvida neste projeto está focada em gerar composições a partir do tamanho, cor e posição dos instrumentos, utilizando o mouse e o teclao para realizar estas funções.

Visando atingir estes objetivo, se faz necessário uma análise de custos para o desenvolvimento destes painéis de instrumentos e a sua confiabilidade em relação aos instrumentos analógicos.

Para a linguagem de programação, uma linguagem com fácil suporte para solução de problemas, como a C, C++ ou Python e equipamentos de teste em bancada para verificar a funcionalidade e a integridade do programa, que neste projeto é composto por uma tela de TFT-LCD de 5 polegadas, com funções *touch* e *SD card* e uma placa de desenvolvimento composto por um controlador ARM SAM3X8E.

Este trabalho está dividido nos seguintes capítulos:

- **Histórico**: será abordada a evolução dos painéis de instrumentos;
- **Análise de design**: demonstrará a importância de se criar as grafias dos instrumentos de forma correta, a fim de facilitar a leitura por parte do motorista;
- **Arquitetura do cluster digital**: aborda a arquitetura eletrônica, no que diz respeito a software e hardware do cluster digital;
- **Projeto**: demonstrará como foi realizado a ferramenta de desenvolvimento, assim como o hardware utilizado para simulá-lo;
- **Confiabilidade e custo**: irá expor a confiabilidade de se utilizar os sistemas de TDT-LCD e a viabilidade do uso da ferramenta criada neste projeto;
- **Conclusão**: abordará os resultados e considerações finais e trabalhos futuros.

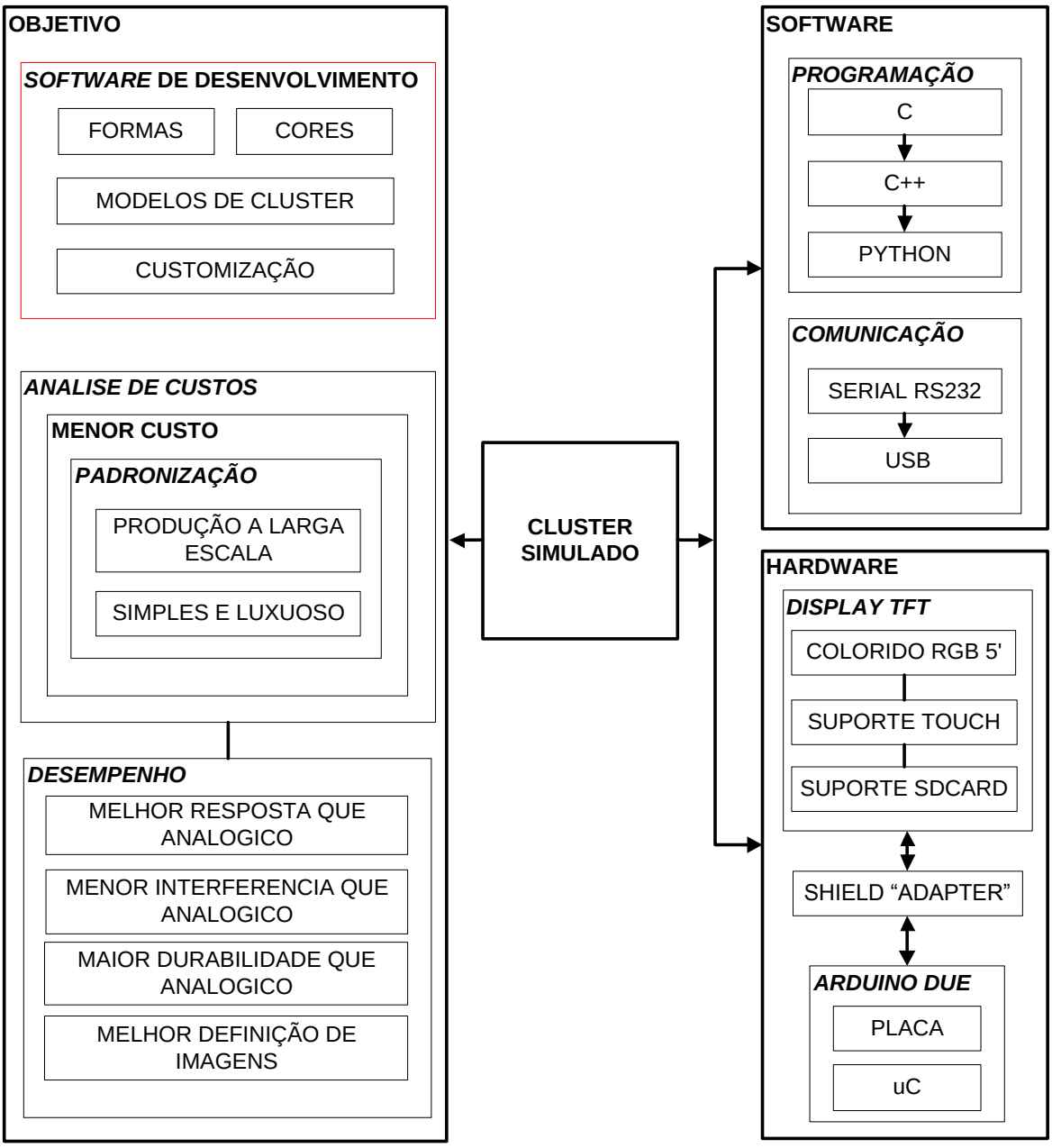

Figura 2 - Diagrama de blocos

5. HISTÓRICO

O primeiro instrumento a surgir nos automóveis foi o velocímetro, criado pelo croata *Josip Belušić* (*Giuseppe Bellussich*) em 1888.

Inicialmente, os veículos não possuíam indicadores (Figura 3), e isso fez com que o número de acidentes envolvendo veículos automotores aumentassem a medida que os veículos ganhavam velocidade conforme a tecnologia avançava.

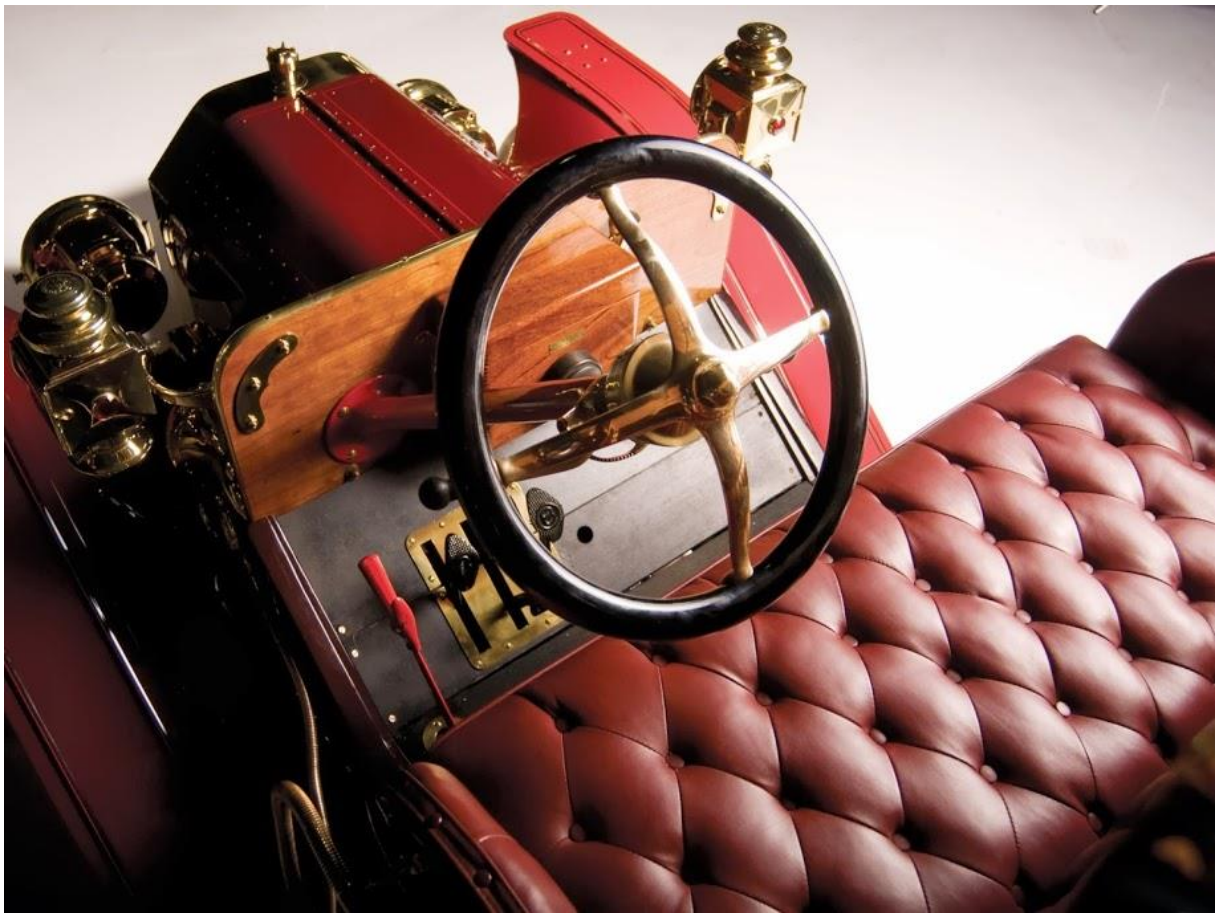

Figura 3 - Ford Model T. (Fonte: plus.google/photos, mai 2014)

Como referência, o primeiro veículo de Karl Benz em 1886 trafegava a uma velocidade média de 16 km/h e em meados de 1895 os veículos já chegavam a 30 km/h (WESNER, 2005).

Para resolver o problema, as autoridades decidiram implementar dois velocímetros, um menor para o motorista, e outro maior para o policial monitorar (WESNER, 2005).

O primeiro velocímetro patenteado ocorreu em 07 de outubro de 1902, pelo engenheiro *Otto Schulze*, em Berlin, na Alemanha.

Em sua publicação *Otto Shulze* demonstrou o método de medição de velocidade por corrente de *Foucault*. O seu funcionamento pode ser visto na Figura 4.

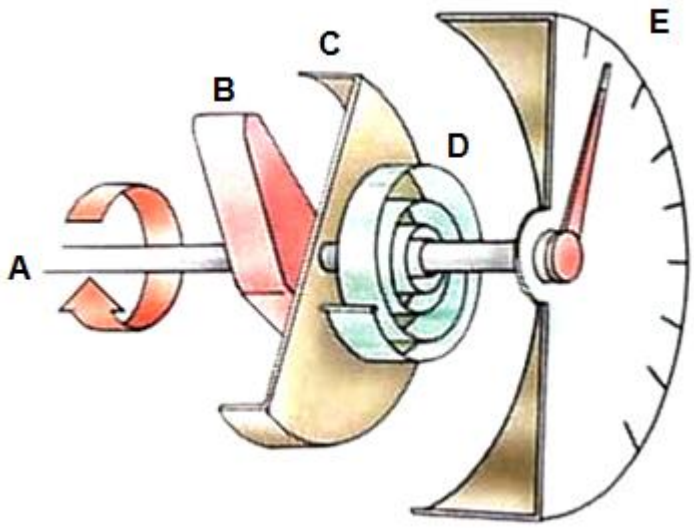

Figura 4 - Funcionamento do velocímetro. (Fonte: flatout.com.br, mai, 2014)

O seu funcionamento se baseia num cabo flexível que gira de acordo com a velocidade do eixo motriz. Este cabo é ligado na parte de trás do velocímetro (A) que possui um imã (B) ligado à ela. De acordo com que a velocidade aumenta, este imã gira, e há o surgimento da corrente de *Foucault* no copo metálico (C), que gera um campo magnético e gira o ponteiro (E). Ligada ao eixo da agulha, há uma mola (D) calibrada de acordo com a velocidade do carro, que é responsável por limitar o seu movimento e realizar o seu retorno. Para que o ponteiro fique parado no zero, há a necessidade de um pino para limitar o seu curso, como pode ser visto na Figura 5.

Figura 5 - Pino no indicador mecânico. (Fonte: terra.com.br, mai 2014)

Quase dois meses depois, em 16 de dezembro de 1902, *Laurence Mott*, com a patente US716263 A (Figura 6), demonstrou um velocímetro baseado na força centrífuga.

Analisando seu funcionamento pela Figura1 da patente, quando a roda (10) está parada ou começa a girar lentamente, o contra-peso (26) preso pela mola (28) encontra-se retraída. De acordo com que a velocidade avança, o contra-peso vai avançando de acordo com a força centrífuga, e um contato (27) vai se fechando com os terminais 19, 20 e 21, que vai ligando as respectivas lâmpadas (32, 33, 34). Cada lâmpada refere-se a uma escala de velocidade diferente, sendo cada uma diferenciada por uma cor.

A figura 2 e figura 3 da patente são modificações propostas pelo autor, porém sempre seguindo o conceito de força centrífuga.

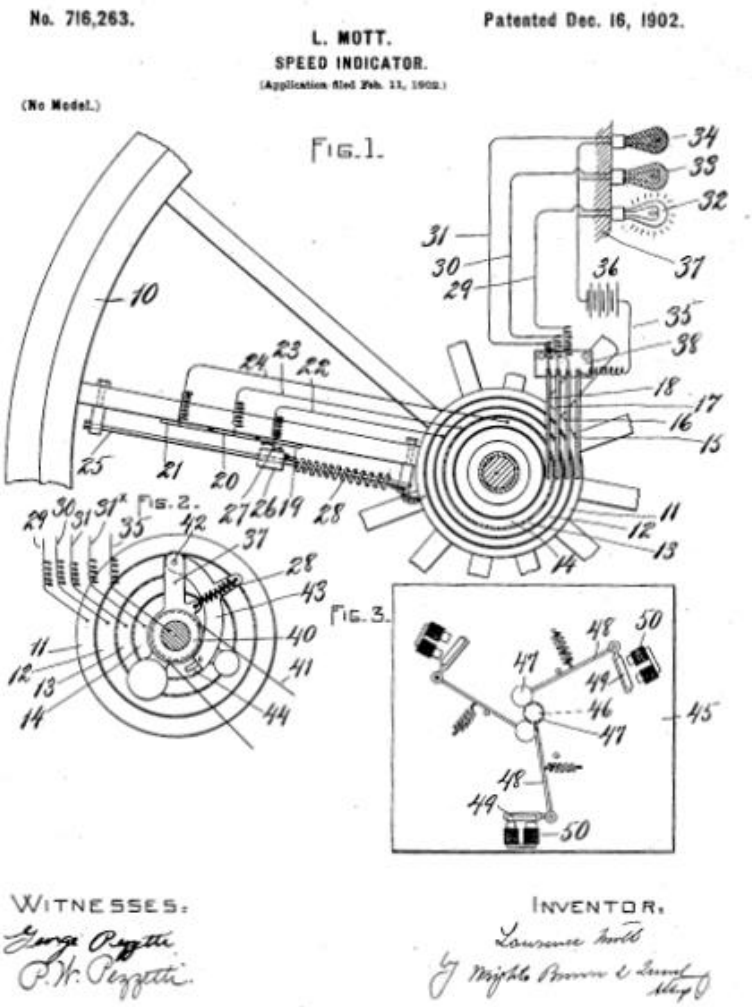

Figura 6 - Patente US716263 A (Fonte: google.com/patents, mai 2014)

A justificativa para o uso deste velocímetro era a sua simplicidade de construção, porém não há registros do seu uso em algum veículo, pois o projeto de *Schulze* mostrou-se mais preciso e confiável.

Seguindo a ordem cronológica, os principais eventos ocorreram:

- 1903: surgem os primeiros odômetros;
- 1905: surgem os primeiros veículos de produção com o velocímetro integrado;
- 1910: os velocímetros passaram a ser equipamento de série;
- 1914: surgem os medidores de quantidade de combustível.

A partir deste período, o *cluster* passou por diversas evoluções, principalmente com a chegada da eletrônica embarcada, possibilitando reduzir o número de elementos mecânicos e ampliar os recursos na instrumentação veicular.

A eletrônica surgiu primeiramente através das luzes piloto (Figura 7), popularmente conhecidas como luzes espias, que tem por finalidade gerar advertência ao motorista através de sinais luminosos, como os piscas, bateria e temperatura do motor.

Figura 7 - Luzes piloto. (Fonte: c2.staticflickr.com, mai 2014)

Em meados da década de 80, surgem os primeiros veículos a oferecerem os painéis de instrumento totalmente digitais com números representados em *displays* de sete segmentos, através de VFD (*Vacuum Fluorescent Display* ou Display de Gráfico Fluorescente), como pode ser visto na Figura 8 e mais tarde em LCD (Liquid Crystal Display, ou Display de Cristal Líquido), representado na Figura 9.

Figura 8 - Display VFD - Chrysler Lebaron 1981 (Fonte: plus.google/photos, mai 2014)

Figura 9 - Display LCD. Golf II e Jetta II 1988. (Adaptado de continental-corporation.com, out 2014)

Em meados da década de 90, começaram a surgir os veículos equipados com *air-bag* e servo-direção, o que diminuiu o campo de visão do *cluster* devido a maior altura do compartimento do volante (BOSCH, 2005).

A diminuição da área de visão e da crescente necessidade de informações adicionais no *cluster*, fez com que os instrumentos totalmente digitais fossem descontinuados, voltando-se os instrumentos analógicos, porém combinados com pequenos *displays* monocromáticos como auxiliar de informações (BOSCH, 2005), como o odômetro ou mesmo o computador de bordo (Figura 10).

Figura 10 - Display digital auxiliar. Toyota Yaris 2008. (Fonte: driversdc.blog, out 2014)

Nesta mesma época, começam a surgir os indicadores controlados por motores de passo em substituição aos antigos movidos pelo princípio da corrente de *Foucault* (o mesmo criado por *Otto Schulze*), além do emprego de LED para a iluminação, que aumenta a nitidez de leitura do cluster (Figura 11).

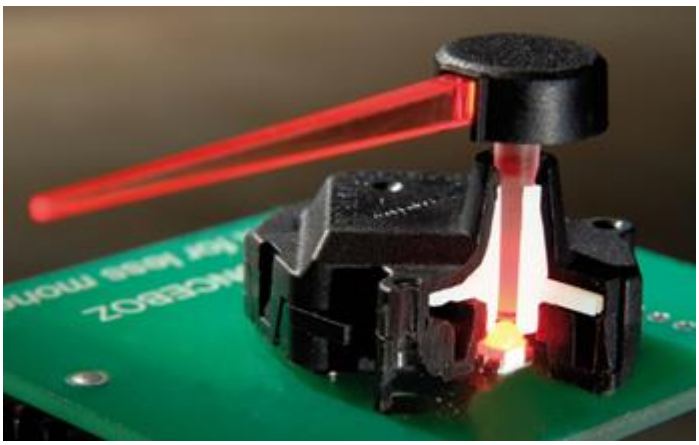

Figura 11 - Motor de passo e LED. (Fonte: http://articles.sae.org, out 2014)

A vantagem dos motores de passo é o seu tamanho mais compacto e movimentação dos ponteiros mais precisos em relação aos de acionamento mecânico.

Os motores de passo funcionam através de um microcontrolador que recebe os sinais de entrada por um barramento (geralmente pela *CAN*), processa estes sinais, e controlam o movimento do motor.

Aos poucos, as telas de LCD monocromáticas foram substituídas pelas primeiras telas de TFT-LCD (*Thin Film Transistor Liquid Crystal Display*, ou Display de Cristal Líquido de Transistor de Película Fina) em veículos de luxo (Figura 12), que devido a maior gama de cores, proporciona maior segurança e velocidade de leitura (BOSCH, 2005).

Figura 12 - Primeiras telas TFT-LCD. Jaguar XK 2007. (Adaptado de media.caranddriver.com, out 2014)

Por volta de 2006, os *displays* TFT-LCD passaram a ocupar toda a área de visualização do painel de instrumentos com representações de instrumentos analógicos (BOSCH, 2005), e com o avanço da nanotecnologia e o aumento da velocidade dos processadores em meados de 2010, os painéis de instrumentos começaram a integrar novas funções aos motoristas, como o *GPS* integrado e até mesmo a personalização do *cluster* de acordo com a preferência do motorista (Figura 13), que é a tendência atual dos painéis de instrumento.

Figura 13 - Cluster customizável. Cadillac XTS 2014 (Fonte: carcoops.com, mai 2014)

6. ANÁLISE DE DESIGN

Embora o *cluster* com painéis em TFT-LCD possibilitem diversas configurações de *design*, é importante salientar que a estrutura gráfica deve ser construída de forma a facilitar a leitura e evitar o desvio de atenção ao conduzir o veículo.

A primeira característica é em relação à área de visualização do *cluster*, que na maioria dos veículos se encontra entre o volante. Quando o volante é esterçado, esta área tende a diminuir drasticamente, como pode ser visualizada a seguir:

Figura 14 - Área de visualização. (Fonte: paulvanslembrouck.com, out 2014)

Não há uma regra para o posicionamento dos instrumentos, mas levando em consideração a área de visualização, alguns fabricantes posicionam o velocímetro no centro, pois ao girar o volante em pequenos ângulos, ainda é possível visualizá-lo (SLEMBROUCK, 2012), como visto na Figura 15.

Figura 15 - Velocímetro central. Fiat Uno Sporting 2014. (Fonte: www.dezeroacem.com.br, out 2014)

Além do correto posicionamento dos instrumentos, existe a opção de instrumentos analógicos ou digitais.

Figura 16 - Cluster analógico e digital. (Fonte: www.paulvanslembrouck.com, out 2014)

A escolha destes formatos influem diretamente na leitura do instrumento, principalmente no velocímetro.

No caso de instrumentos analógico, este possui a capacidade de indicar a aceleração através da velocidade e da direção de movimentação do ponteiro, obtendo-se uma referência da aceleração ou desaceleração do veículo (SLEMBROUCK, 2012), como exemplificado na Figura 17.

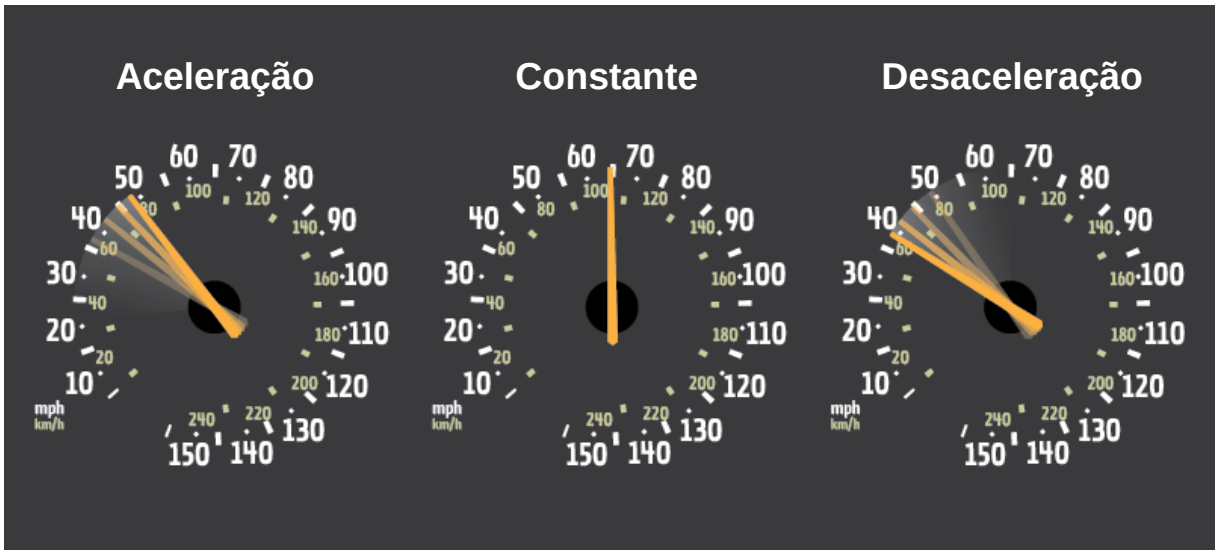

Figura 17 - Movimentação do ponteiro. (Adaptado de www.paulvanslembrouck.com, out 2014)

Geralmente os instrumentos analógicos possuem a escala entre 100 e 120 km/h na posição de 90º, isto ajuda a indicar se o veículo está em uma velocidade baixa ou

alta (SLEMBROUCK, 2012), em que o instrumento realiza a função de um "gráfico de pizza" (Figura 18).

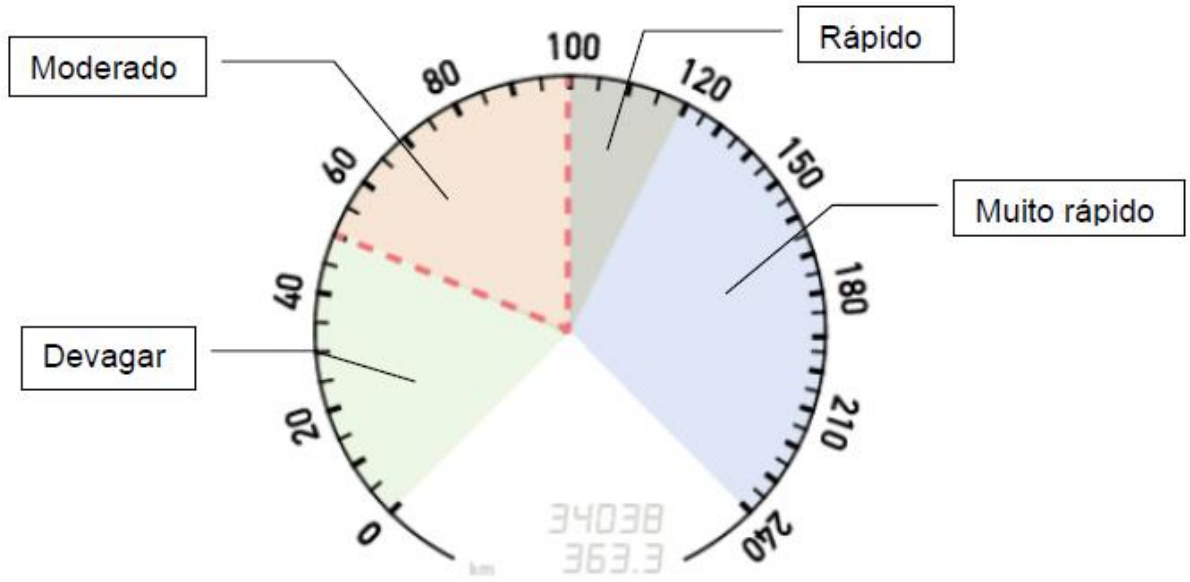

Figura 18 - Gráfico de pizza do velocímetro. (Adaptado de vis4.net, out 2014)

Todas estas características que facilitam a leitura possuem maior dificuldade de implementação em velocímetros digitais e por este motivo, a maioria dos painéis de instrumentos são analógicos.

No cluster customizável, embora seja possível infinitas composições de instrumentos, a maioria dos fabricantes ainda disponibilizam as informações em simulações de instrumentos analógicos.

Uma das únicas exigências na construção do *cluster* é em relação a simbologia das luzes pilotos, que devem seguir a norma ISO 2575 (*Road vehicles - Symbols for controls, indicators and tell-tales*, ou Veículos rodoviários - Símbolos para controles, indicadores e avisos).

Figura 19 - Simbologia ISO 2575. (Adaptado de i.dailymail.co.uk, out 2014)

Esta norma contém cerca de 350 simbologias e cores padrões, que visa facilitar o reconhecimento e evitar confusões.

Algumas destas simbologias podem ser visualizadas na Figura 19.

7. ARQUITETURA DO *CLUSTER* DIGITAL

É inevitável o surgimento cada vez maior da eletrônica embarcada nos veículos, e isto sempre gerou discussões sobre a sua confiabilidade, pois toda programação é passível de possuir falhas em seu algoritmo.

Em solução a este problema, surge a necessidade de adoção de normas para evitar o máximo de erros possível, e para isso, existem a ANSI (*American National Standarts Institute*) e a MISRA (*Motor Industry Software Reliability Association*).

A ANSI surgiu em 1988 como um meio de padronizar a linguagem C, como as declarações de variáveis, chamadas de funções e bibliotecas, a fim de que o programa possa ser utilizado em qualquer sistema e compilado em qualquer compilador. Porém nem sempre isto é possível nos sistemas embarcados, pois há os drivers específicos de cada microcontrolador, que na maioria das vezes não são feitos no padrão ANSI C.

Para solucionar este problema, a associação de algumas indústrias automotiva do Reino Unido publicou a MISRA C, que possui a mesma finalidade da ANSI C, porém focado nos sistemas automotivos. A MISRA é baseada na norma ISO/IEC 1989:1990, que é idêntica ao padrão ANSI.

O uso da MISRA C fez tanto sucesso, que posteriormente também passou a ser utilizada no setor aeronáutico e de equipamentos hospitalares, devido a sua confiabilidade em proporcionar uma linguagem de melhor entendimento e evitando erros de interpretação e programação, e em 2008, surge a MISRA C++, que possui os mesmos fundamentos da MISRA C, porém para a linguagem orientada a objeto.

Mas quando se fala em painéis de instrumentos automotivos em TFT-LCD, além da padronização da linguagem, o *cluster* do veículo deve garantir que as informações sejam mostradas com certa precisão e com o mínimo de atraso para evitar frustrações ao condutor, assim um tempo recomendado para a inicialização é em torno de 2 segundos (BOSCH?).

Para que isso seja possível, deve-se utilizar um sistema baseado em tempo real, também conhecido como RTOS (*Real Time Operating System*).

O RTOS é um sistema operacional em que as tarefas ou funções são comandadas em tempos pré determinados, garantindo maior fluidez em operações multi-tarefas e permitindo que a unidade de controle não trabalhe no seu limite.

Com o uso da norma MISRA e de sistemas de tempo real, o *display* em painéis de instrumentos tornam-se tão confiáveis quando os atuais sistemas eletromecânicos (via motor de passo), além de maior versatilidade, tais como:

- Possibilidade de implementar instrumentos adicionais sem modificar o *hardware* e a estrutura do painel do veículo;
- Utilizar o mesmo *hardware* para diversos veículos e modificar somente a programação, fato que diminui os custos;
- Possibilidade de integrar a tela de diagnose no próprio painel de instrumentos;
- Não possui partes móveis, o que diminui o peso e elimina o desgaste de peças;
- Informação otimizada de acordo com a situação;
- Personalização de acordo com a preferência do usuário.

A principal dificuldade dos *clusters* baseados em TFT-LCD é o custo, pois como a manipulação de imagens requer maior poder de processamento, o uso de microprocessadores ao invés de microcontroladores são inevitáveis, assim como a memória RAM e memória *flash* externas com maiores capacidades. Uma comparação entre as arquiteturas pode ser visualizadas na Figura 20 e Figura 21.

Para fins comparativos, na linha de controladores da *Freescale*, enquanto o microcontrolador para um cluster básico é de 16 bits e frequência de 40 Mhz, o microprocessador para aplicação em cluster reconfigurável com capacidade de processamento 3D pode ter até quatro núcleos de 1,2 Ghz, demostrando a complexidade no processamento de imagens.

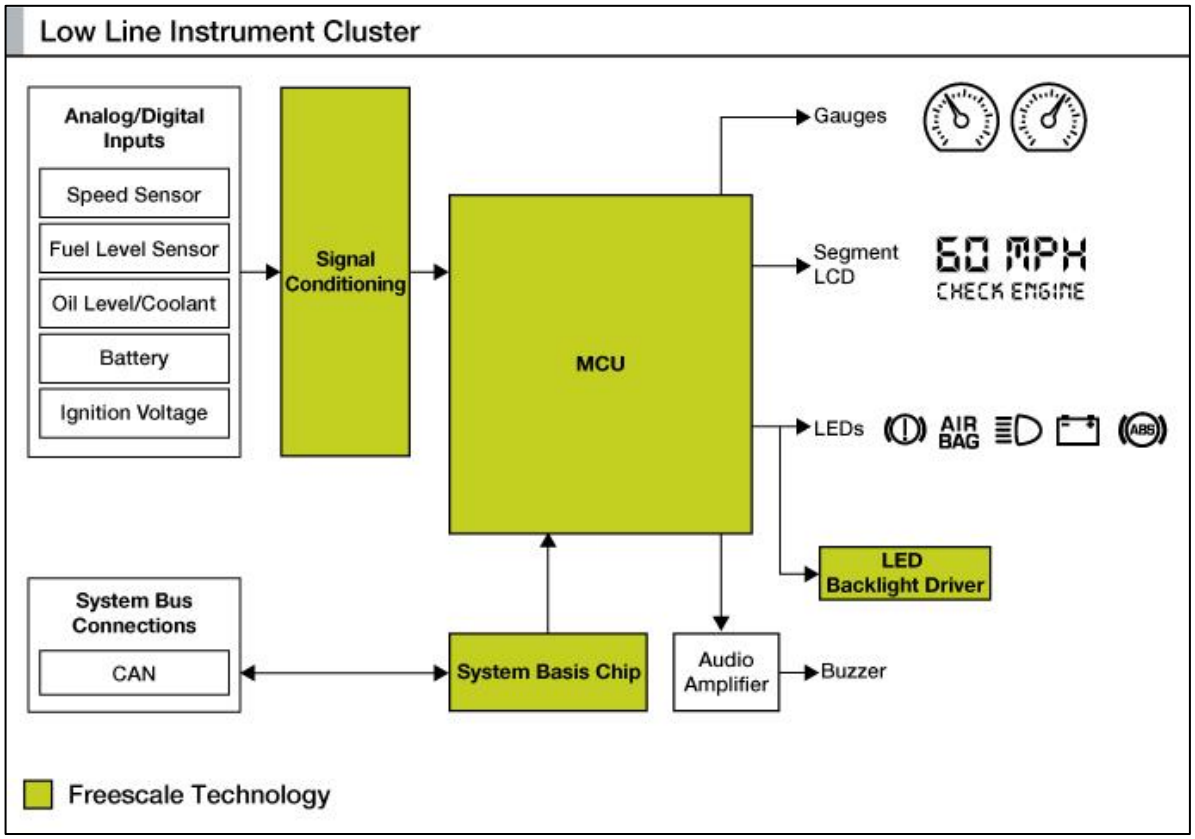

Figura 20 - Arquitetura com motor de passo. (Fonte: freescale.com, out 2014)

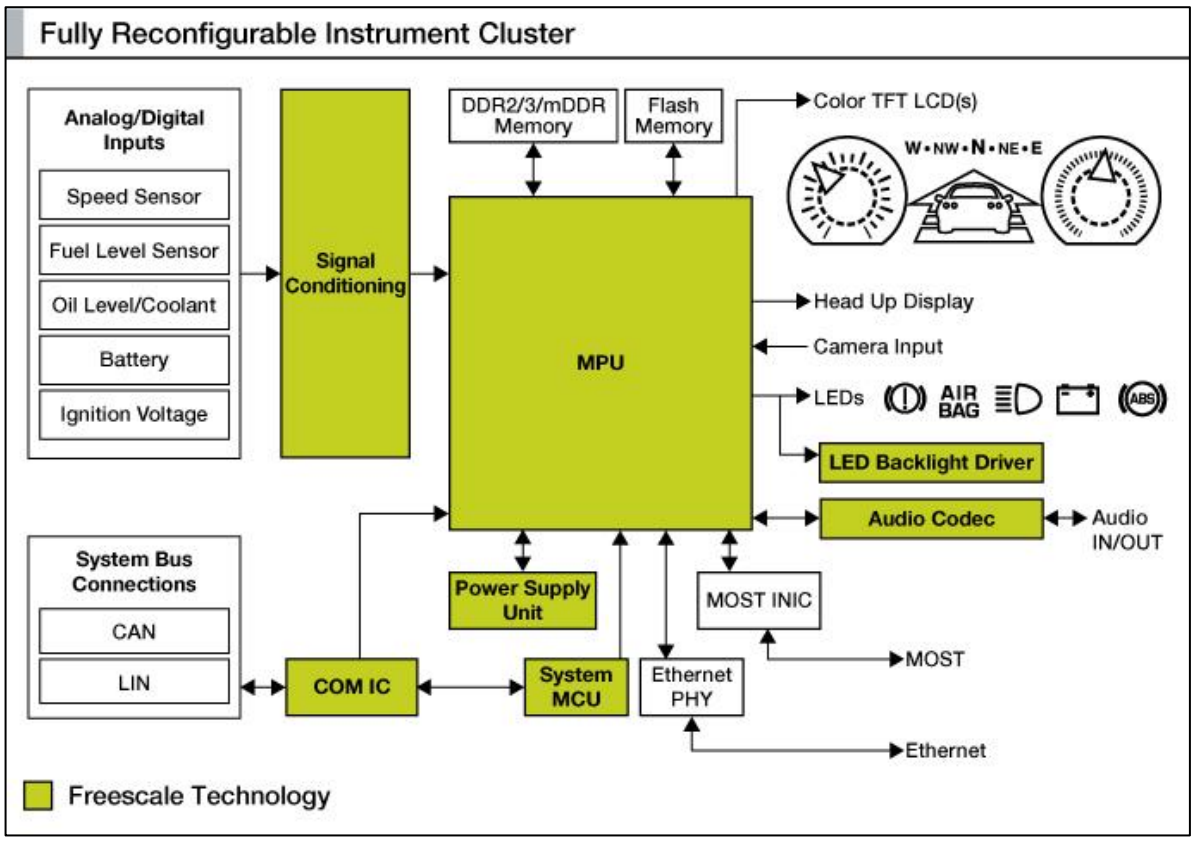

Figura 21 - Arquitetura com display TFT-LCD. (Fonte: freescale.com, out 2014)

8. PROJETO

O projeto consiste na criação de um *software* de desenvolvimento de instrumentação digital para telas de TFT-LCD, sendo similar a uma API (*Application Programming Interface*, ou Interface de Programação de Aplicativo), em que através de formatos pré-definidos do *cluster*, o desenvolvedor utiliza-o de forma a compor o painel de instrumentos sem se preocupar com a programação.

A fase de desenvolvimento do cluster pode ser dividida conforme a Figura 22.

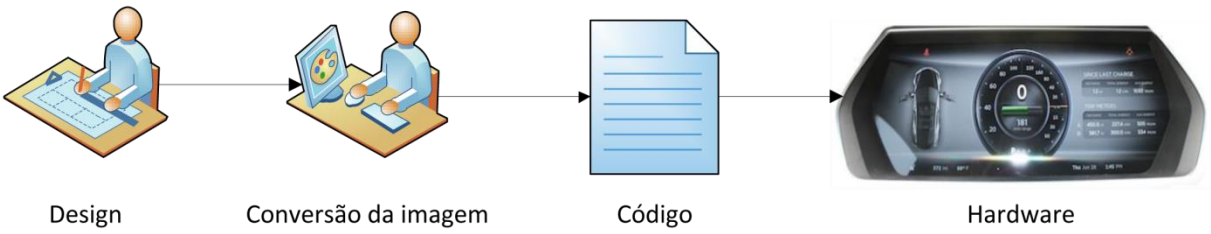

Figura 22 - Implementação do projeto.

As quatro etapas se baseiam em:

1. ***Design***: consiste na criação do design do cluster;
2. **Conversão da imagem**: feito o design, este é passado para o programa, que realiza as etapas de conversão da imagem e adequação do display;
3. **Código**: a aplicação gera um código de programação pronta para ser implementada no *cluster*;
4. ***Hardware***: é utilizado um display TFT-LCD para a simulação do código

8.1 *Software* de *design* e simulação

A ferramenta de desenvolvimento do projeto foi desenvolvido em linguagem C# na IDE *Microsoft C# Express 2010*, que é disponibilizada pela Microsoft de forma gratuita, fato que facilita futuras implementações do projeto sem gastos adicionais.

A ferramenta do projeto é dividida em dois sub-programas, que é o *design* (Figura 23) e a simulação (Figura 25).

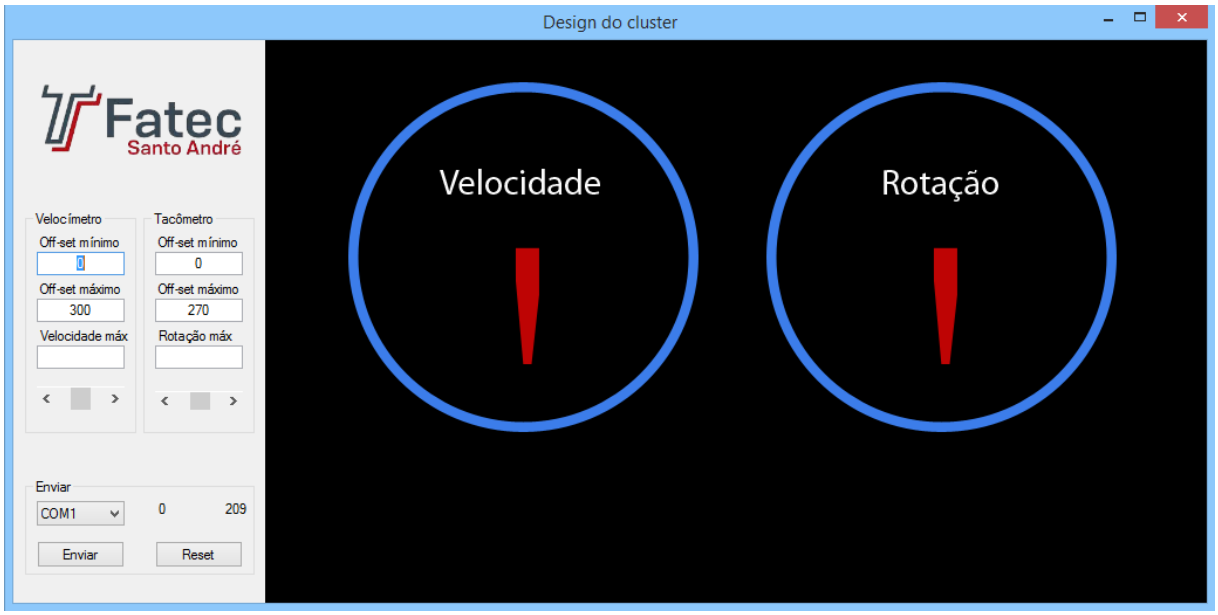

Figura 23 - Software de desenvolvimento do cluster. (Criação do autor, out 2014)

O *software* de *design* possibilita ao desenvolvedor criar a composição do *cluster* conforme a necessidade do mercado.

Como fase inicial de desenvolvimento, o *software* possibilitará somente a manipulação de instrumentos analógicos circulares pré-definidos, em que o posicionamento na tela do *cluster* (coordenadas x e y) é realizada com o *mouse* e as configurações através do teclado.

O funcionamento do *software* pode ser visto na Figura 24.

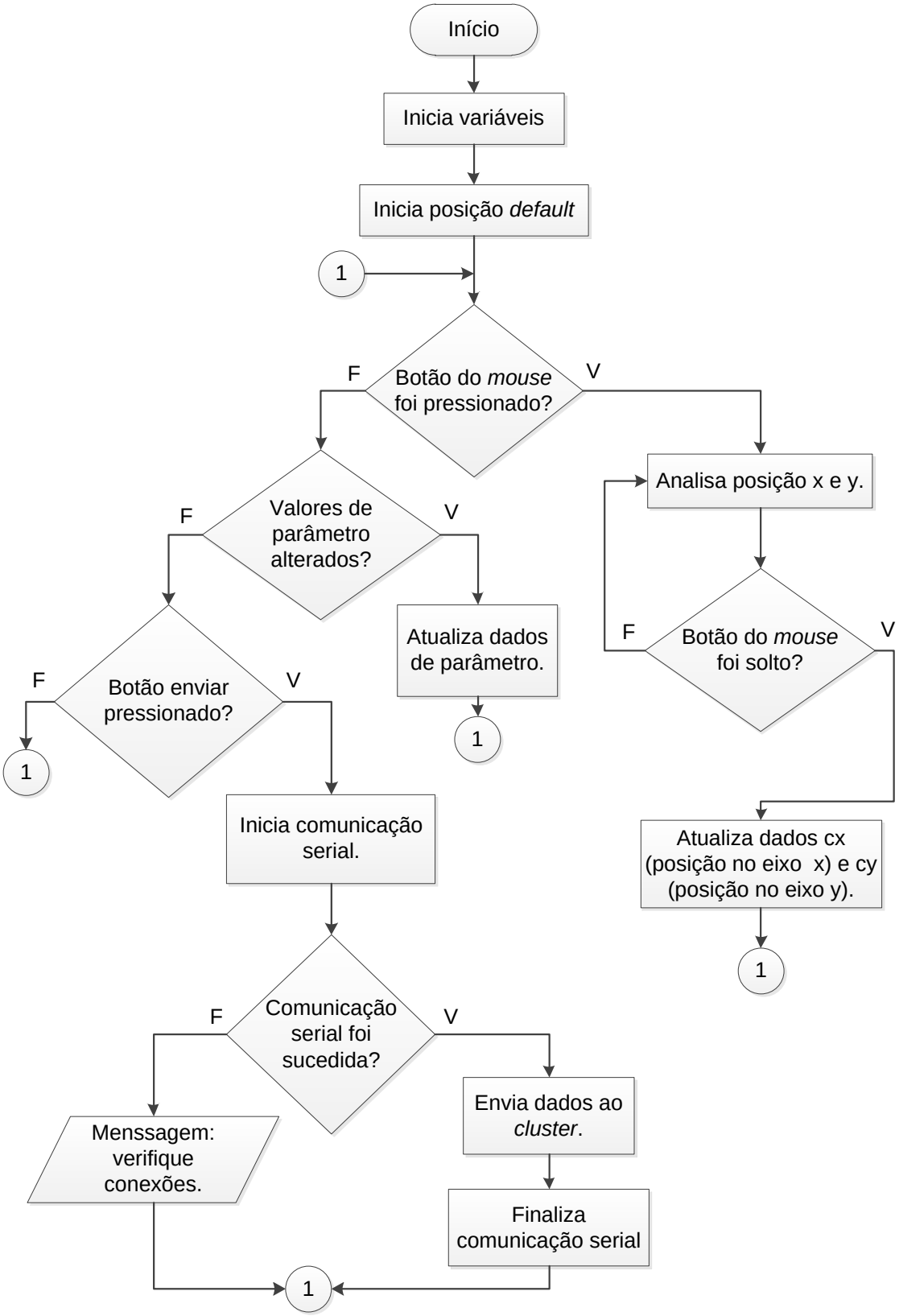

Figura 24 - Fluxograma do *software* de *design*.

Analisando o fluxograma da Figura 24, o *software* inicia com valores *default* (padrão) de posicionamento dos instrumentos (Figura 23) para iniciar a rotina principal.

Este posicionamento ocorre através de coordenadas no eixo cartesiano, onde x possuirá valores entre 0 a 800 e y de 0 a 480, que é exatamente o tamanho em pixel do display utilizado no projeto.

A rotina principal do programa consiste em verificar três eventos principais.

A primeira consiste em verificar se o botão do *mouse* foi pressionado, caso verdadeiro, analisa o posicionamento da imagem no eixo x e y e armazena os valores na variável quando o botão do *mouse* é solto;

Depois é verificado se os valores de parâmetro foram modificados pelo teclado, como a posição mínima e máxima do ponteiro e do valor máximo do instrumento (por exemplo, 250 km/h).

Após o *design* do *cluster* estar finalizado, o projetista deve clicar no botão enviar. Caso exista falha na comunicação, uma mensagem de erro é mostrada e deve-se verificar as conexões do cabo e se a porta COM selecionada está correta. Se ao clique do botão enviar não houver erros, os dados de configuração são enviados através de um *frame* por comunicação serial para uma plataforma de desenvolvimento composto por um processador ARM SAM3X8E e um *display* TFT-LCD de 5 polegadas.

O formato de construção deste frame é dada conforme Tabela 1:

Tabela 1 - Matriz de design.

Matriz de dados de design					
Função	**Variáveis**				
Uso geral	2	F_draw			
Velocidade	cx	cy	off_min	off_max	val_max
Rotação	cx	cy	off_min	off_max	val_max
Temperatura	cx	cy	off_min	off_max	val_max
Combustível	cx	cy	off_min	off_max	val_max

A primeira linha contém as variáveis de uso geral, composta por:

- Constante 2: indica que a matriz possui valores referentes ao *design* do *cluster*;
- Variável f_draw: é uma *flag* que indica se o cluster deve ser redesenhado ou não, em que será 0 se não houver necessidade e 1 para necessidade de redesenho.

As demais variáveis são referentes à instrumentação de velocidade, rotação, temperatura e combustível, podendo ser expandida conforme a necessidade de novos instrumentos. As variáveis destas funções são:

- Cx: indica a posição da figura no eixo x;
- Cy: indica a posição da figura no eixo y;
- Off_min: indica em que ângulo da circunferência o ponteiro e a numeração se inicia;
- Off_max: indica em que ângulo da circunferência o ponteiro e a numeração se finaliza;
- Val_max: indica o valor máximo do instrumento.

A outra ferramenta do projeto é o *software* de simulação (Figura 25), que possui a função de verificar se a parte dinâmica do *cluster* opera corretamente, como por exemplo a movimentação dos ponteiros e o acionamento das luzes piloto.

Figura 25 - Simulador de sinais.

O funcionamento do simulador consiste em digitar os valores dos instrumentos na caixa de texto para movimentar os ponteiros e clicar nos icones das luzes piloto para acioná-la ou desacioná-la.

Ao clique do botão “enviar”, um frame é gerado e enviado ao display pela comunicação serial.

O frame de simulação é gerado conforme Tabela 2:

Tabela 2- Matriz de simulação.

Matriz de dados de simulação						
Função	**Variáveis**					
Uso geral	1	F_vel	F_rot	F_temp	F_comb	
Luzes piloto	Bat	F_mão	Lim	S_esq	S_dir	Farol
Valores	vel	rot	temp	comb		

Na linha de “uso geral” é composta por:

- Constante 1: indica que se trata de uma matriz de simulação;
- Variáveis F_vel, F_rot, F_temp e F_comb: são *flags* para o uso de movimentação dos ponteiros, onde:
    - F_vel = flag de velocidade;
    - F_rot = flag de rotação;
    - F_temp = flag de temperatura;
    - F_comb = flag de combustível.

A linha de "luzes piloto" contém os estados das luzes piloto, onde:

- Bat = flag da bateria;
- F_mão = flag do freio de mão;
- Lim = flag da luz indicadora de mal funcionamento;
- S_esq = flag da seta esquerda;
- S_dir = flag da seta direita;
- Farol = flag do farol.

A linha de “valores” contém todas as variáveis de controle dos instrumentos, onde:

- Vel = valor da velocidade;
- Rot = calor da rotação;
- Temp = valor da temperatura;
- Comb = valor do combustível.

Feito este processo, o resultado pode ser visualizado na tela do display, como pode ser visto na Figura 26.

Figura 26 - Display do cluster.

Para que estas imagens sejam mostradas no display, é necessário que as matrizes de simulação e design sejam processadas no hardware, sendo essencial portanto um programa, que neste caso, foi criada na IDE *Atmel Studio 6*, utilizando a linguagem C++.

O uso desta IDEs diminui o tempo de aprendizado no uso das ferramentas, pois uma vez que o C# Express e o Atmel Studio utilizam como base o Visual Studio, a interface de ambas são similares,

A análise do software de simulação do cluster, ocorre conforme a Figura 27.

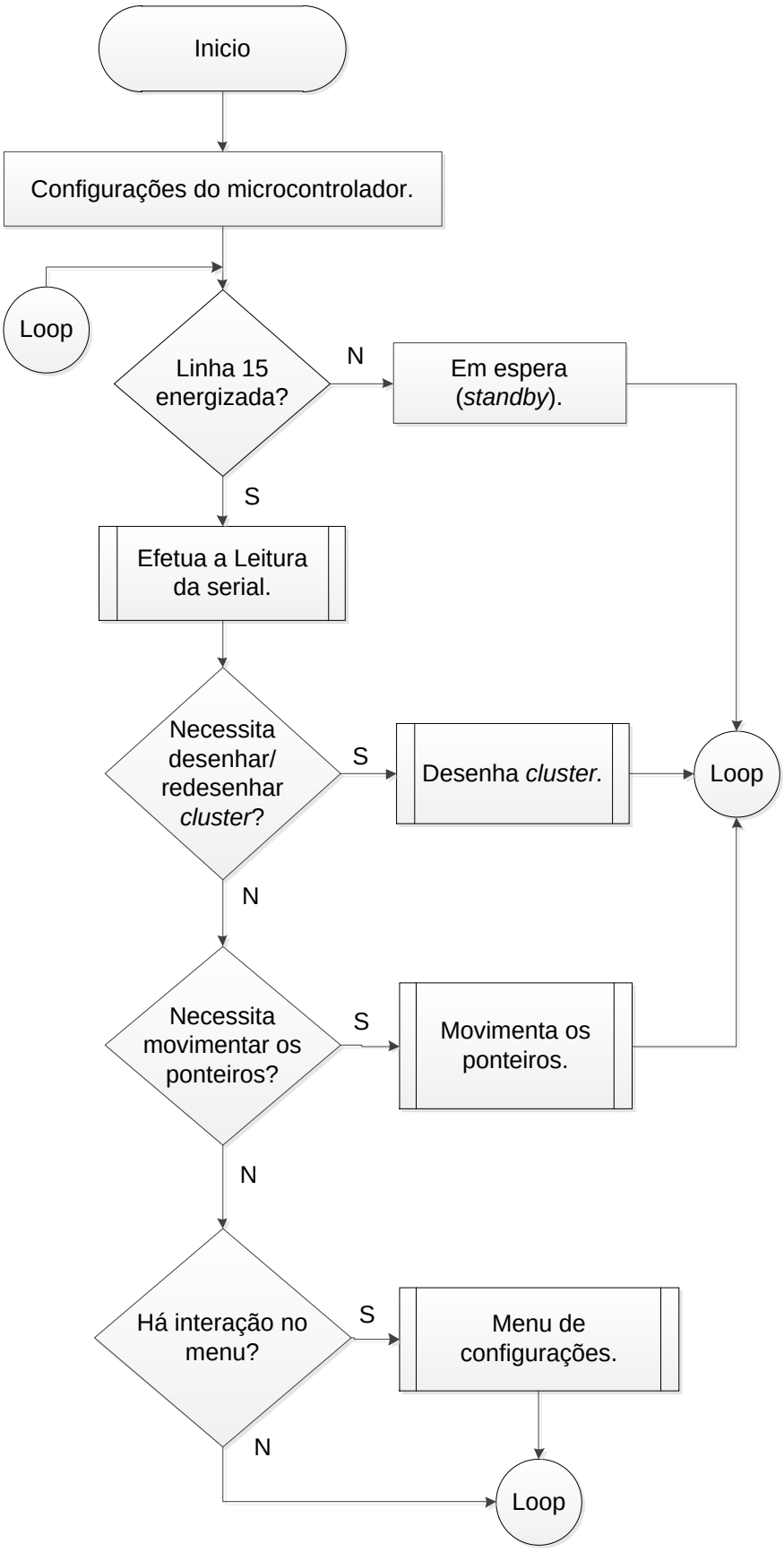

Figura 27 - Rotina principal do Arduino.

Inicialmente é realizada as configurações básicas do *hardware*, como por exemplo o *display*, *touch*, entradas e saídas e a comunicação serial.

Após o processo de configuração, inicia-se a rotina principal do programa, que consiste em analisar se a linha 15 (pós-chave) do veículo se encontra energizada. Caso não esteja, o *cluster* permanece no estado de *standby* (espera), caso contrário, a rotina prossegue com a leitura dos dados através da comunicação serial (Figura 28).

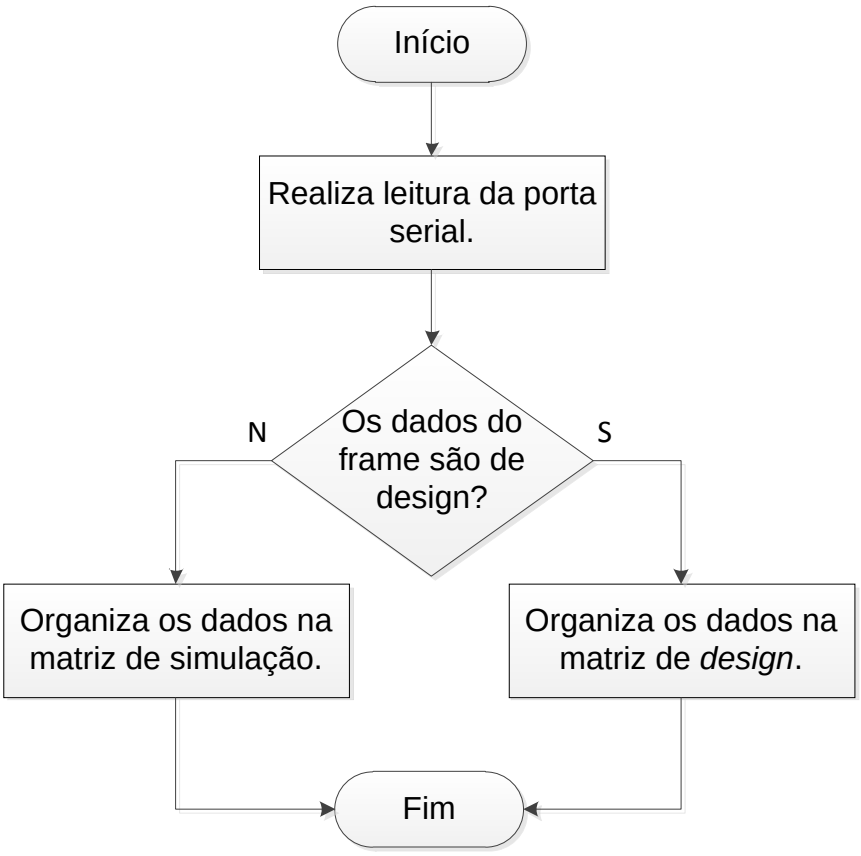

Figura 28 - Subrotina da leitura serial.

Neste processo, é verificado se o *frame* de dados enviado pelo *software* de desenvolvimento do *cluster* são dados de simulação ou *design* através do primeiro bit do *frame,* sendo 1 ou 2, respectivamente.

Após esta identificação, a subrotina finaliza e volta à rotina principal.

Continuando a leitura do fluxograma principal (Figura 27), o próximo passo é verificar se há a necessidade de desenhar ou redesenhar o *cluster* de acordo com o estado de F_draw, da matriz de *design*. Se ela for verdadeira, é chamada a subrotina de desenhar o *cluster*, representada pela Figura 29.

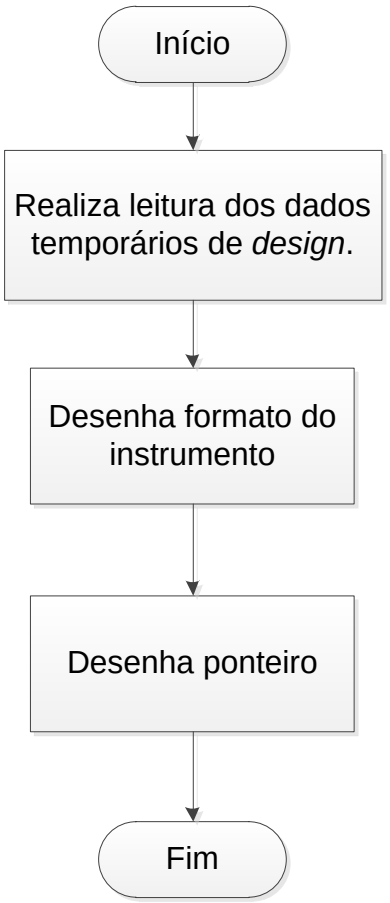

Figura 29 - Subrotina de desenhar *cluster*.

Nesta subrotina, todos os dados dos instrumentos obtida pela matriz de *design* são lidas, e a partir destes valores, o *hardware* envia comandos para realizar o processo de desenho das formas do *cluster*, como as circunferências, os números, as cores, os ponteiros, etc.

O próximo passo da rotina principal é a movimentação dos ponteiros, conforme Figura 30.

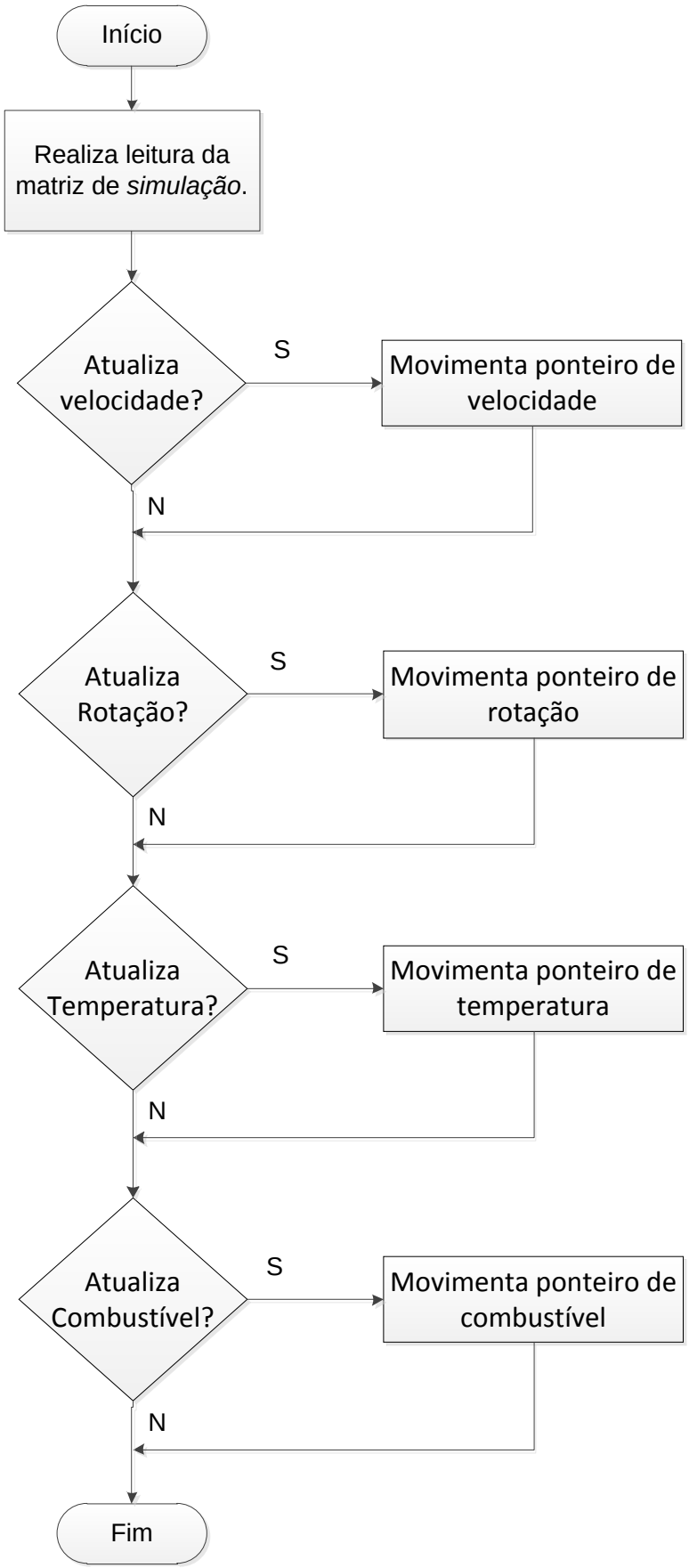

Figura 30 - Subrotina de movimentação dos ponteiros.

Nesta subrotina, são lidos os valores da matriz de simulação e verificado se há ou não a necessidade de atualização dos ponteiros de acordo com as *flags* (F_vel, F_rot, F_temp e F_comb). Se alguma destas *flags* possuir valor 1, haverá atualização dos ponteiros.

A etapa final da rotina principal consiste em verificar se houve interação com o menu através do *touch*. Caso possua, ocorre a subrotina do menu (Figura 31).

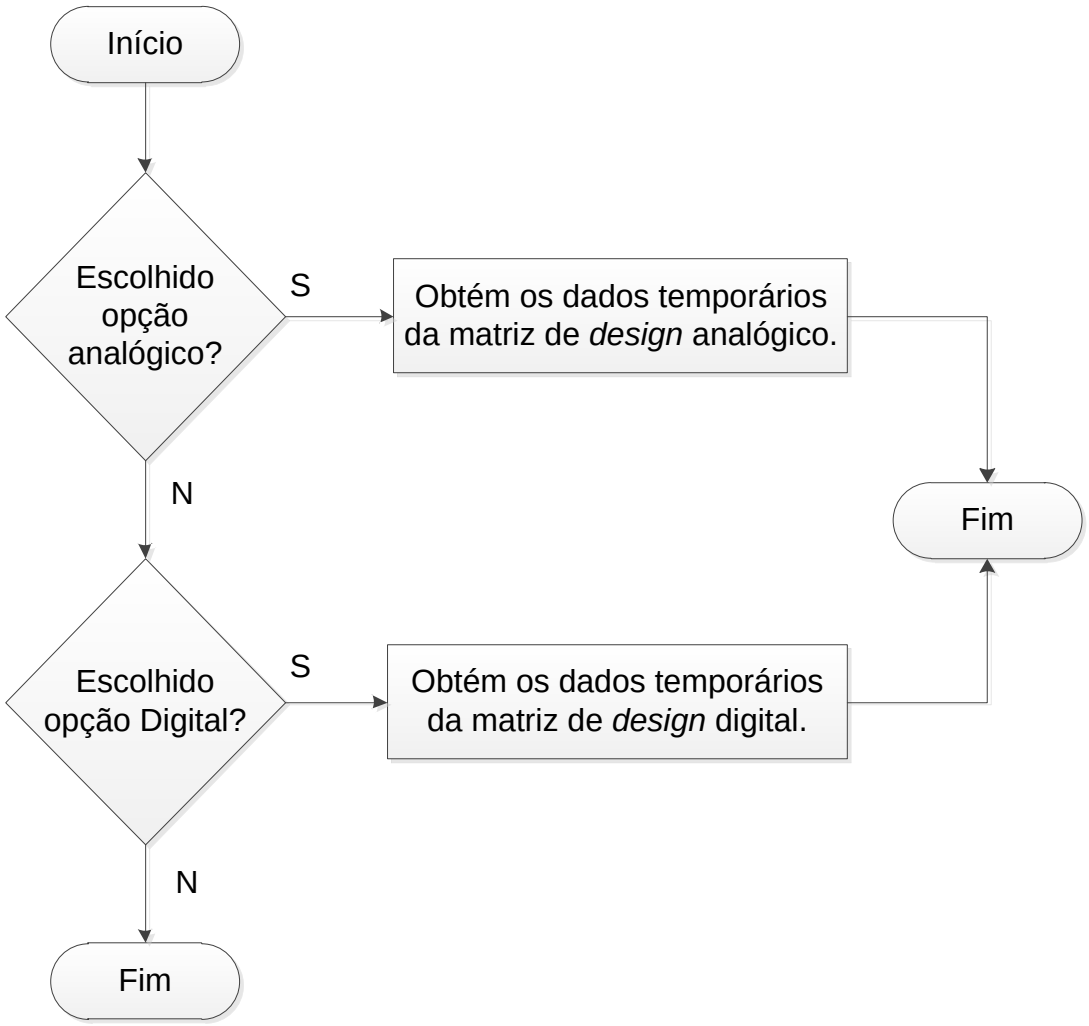

Figura 31 - Subrotina de menu.

Nesta subrotina, é verificado qual opção o usuário selecionou, que no caso do projeto, consiste apenas o modo digital ou analógico.

Após selecionado, os dados da matriz de *design* são modificados para posterior redesenho quando a rotina se iniciar novamente.

8.2 *Hardware* do *Cluster* de simulação

Para atender grande parte das aplicações de um *cluster* de forma dinâmica e confiável em um tempo de resposta adequado, devido o processamento de imagem, buscou-se um microcontrolador *mid-range*.

Estes microcontroladores são facilmente encontrados em conjunto à placas de desenvolvimento para hobistas, como o *Arduino*, *Beagle Bones Black* ou *Raspberry*.

Dentre os modelos disponíveis no mercado, o *hardware* que atendia os requisitos necessários foi o Arduino Due (Figura 32), que possui como características:

- Microcontrolador Atmel SAM3X8E ARM Cortex-M3, que possui largura de banda de processamento de 32 bits e velocidade de processamento de até 84 Mhz;
- Interface padrão JTAG para depuração;
- Suporte gráfico para *displays* coloridos;
- 54 portas de I/O, sendo 12 destas configuráveis como PWM, 12 como entradas analógicas e 4 como seriais RS 232;
- Comunicações I2C, SPI e CAN;
- 96 KBytes de SRAM e 512 KBytes de memória *Flash* para armazenar o programa;
- DMA (*Direct Memory Access*), ou acesso direto à memória.

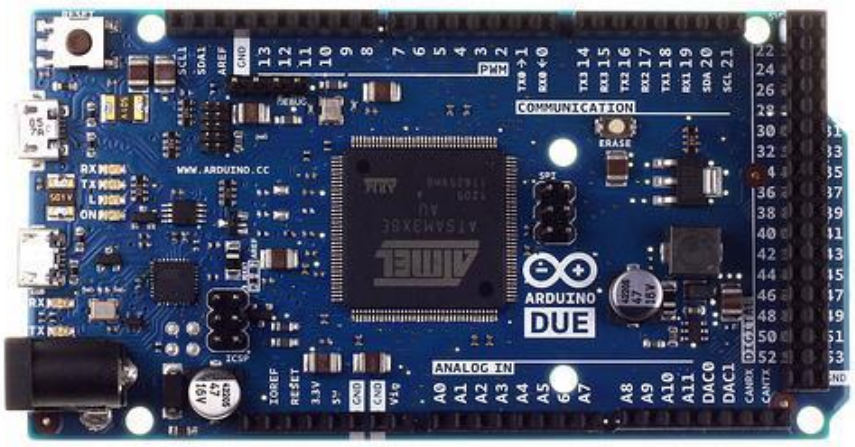

Figura 32 - Arduino Due. (Fonte: arduino.cc, mai 2014)

Os periféricos e as portas do Arduino Due pode ser visto na Figura 33.

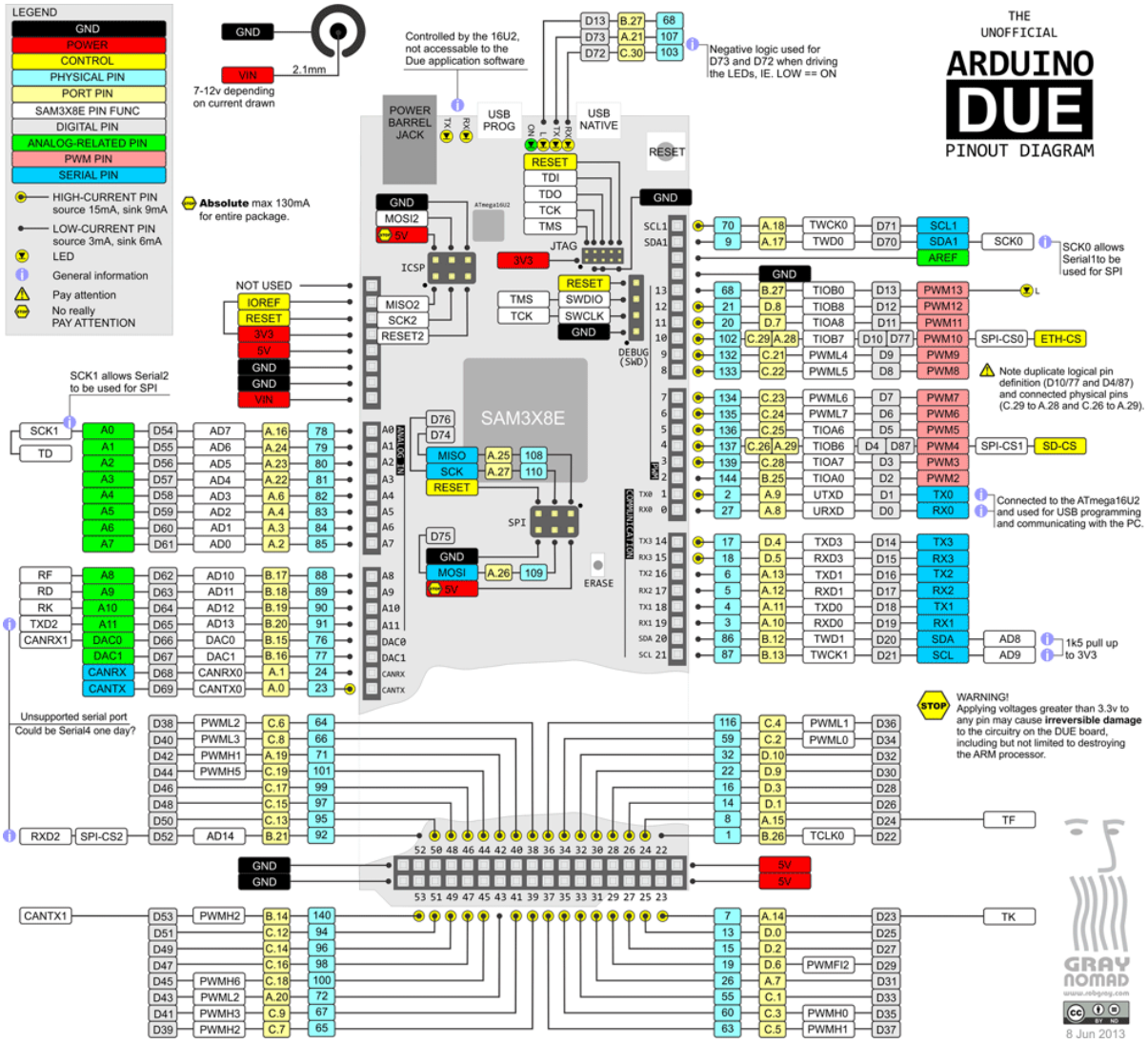

Figura 33 - Diagrama Arduino Due. (Fonte: arduino.cc, mai 2014)

O display (Figura 34) do projeto é da fabricante *Itead Studio*, que possui tamanho de 5 polegadas e resolução de 800x480 pixels, de até 65.000 cores e 24 bits de profundidade, entrada *touchscreen* e interface para cartão SD de até 4Gb.

Figura 34 - Display de 5 polegadas. (Fonte: imall.inteadstudio.com, mai 2014)

O controlador de vídeo, do *touch* e do cartão SD é o SSD1963, com um *buffer* de 1 Mb, opção de controle de brilho e contraste e tensão de controle de 3,3 V.

A comunicação do vídeo e do *touch*, é feita de forma paralela de 16 bits por instrução, já as operações com o cartão SD utilizam comunicação SPI.

Para conectar o Arduino ao *display*, é necessário o uso do "*Shield* ITDB02" (Figura 35) cujo objetivo é a interface dos dois dispositivos, uma vez que não são padronizados.

O *shield* não comporta nenhum processador, apenas alguns amplificadores operacionais, de forma a saturar em 3.3 V e garantir que a tensão de trabalho do *display* permaneça nesta tensão.

Além destas funções, o *shield* evita mal contato e ruídos por não possuir ligações com fios ou cabos, e sim um conector.

Este conector faz as seguintes ligações:

- 16 pinos para a comunicação de dados paralela;
- 4 de comandos;
- Alimentação do *display* através do Arduino Due;
- Conexão dos pinos do *touch* resistivo;
- Comunicação SPI do leitor de cartão SD da placa do LCD.

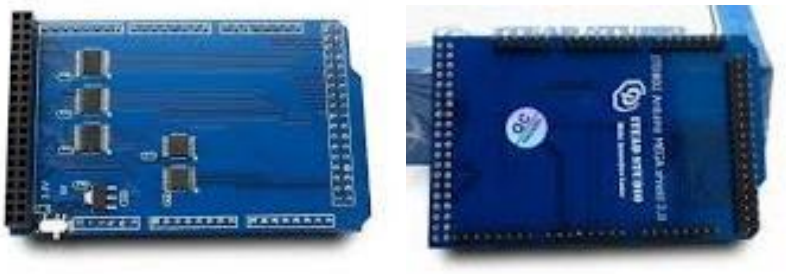

Figura 35 - Shield. (Fonte: imall.inteadstudio.com, mai 2014)

A figura a seguir mostra o diagrama de blocos com a ligação do display ao shield e este enviando um barramento de imagem e touch para o microcontrolador, que também recebe alimentação e comunicação serial pela porta USB:

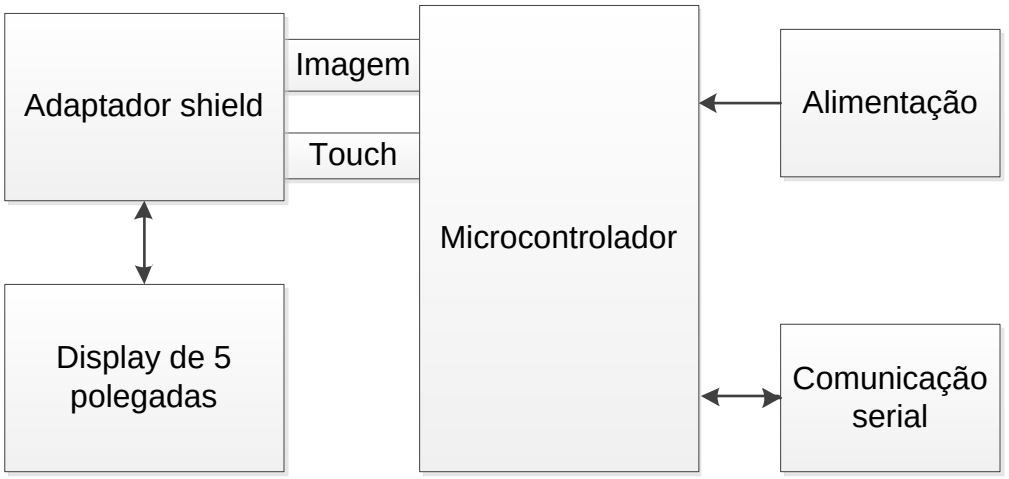

Figura 36 - Diagrama de blocos do harware.

8.3 Confiabilidade e custo

Os displays de tela colorida possibilitam ampla gama de customizações e variadas funções que auxiliam o motorista.

Porém, como a maioria dos equipamentos eletrônicos, muito se questiona sobre a substituição da mecânica pela eletrônica, no que se refere à confiabilidade e custos.

Grande parte das pesquisas e desenvolvimentos se iniciam na área espacial, fato que comprova a sua confiabilidade em sistemas embarcados, além do desenvolvimento difundido em diversas áreas, como nos celulares, sistemas multimídias e computadores, em que o uso de displays se mostram confiáveis.

Mas a grande vantagem do uso destes *displays*, é que através de um único *hardware* desenvolvido, o *cluster* poderá atender desde o nível básico até o nível *premium*, que diminui custos tanto na produção quanto na manufatura, pois não é necessário possuir partes mecânicas como os motores de passo, sendo necessário apenas o uso de máscaras para o formato do *cluster*, como pode ser visualizada na Figura 37.

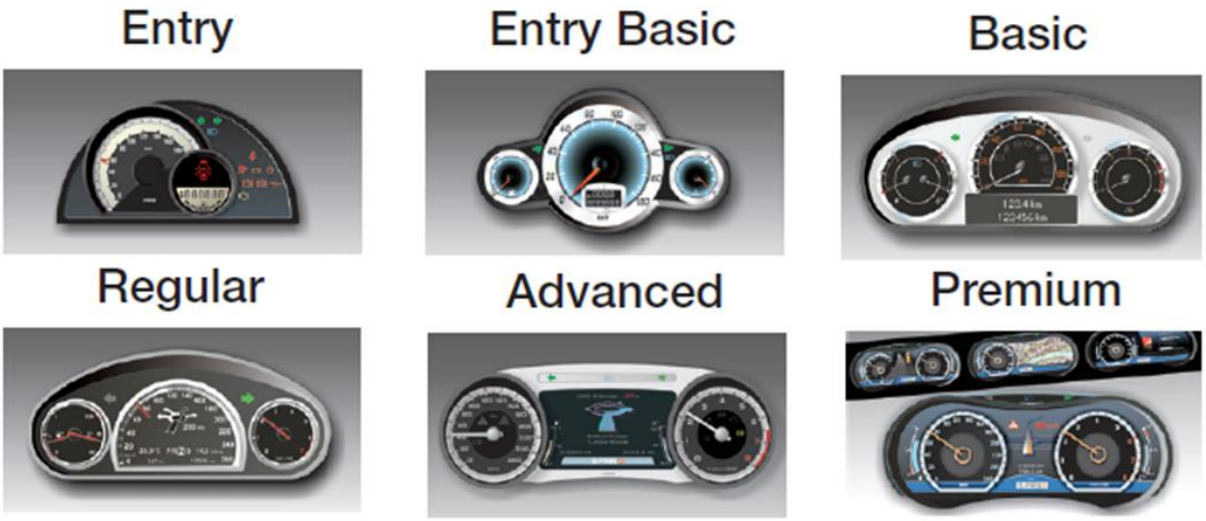

Figura 37 - Máscara no cluster.

Para que o desenvolvimento do *cluster* seja viável, é necessário uma ferramenta de desenvolvimento (criada neste projeto) para diminuir o tempo de desenvolvimento, pois uma vez que a ferramenta já se encontra programada, o tempo gasto para realizar a composição do *cluster* é mínima.

Por exemplo, supondo que o tempo gasto para criar a estrutura de desenvolvimento (baixo nível de programação) seja de 6 meses e para criar a composição do *cluster* (alto nível de programação) seja de 2 horas.

Caso não exista a ferramenta de desenvolvimento, o tempo para produzir 3 modelos de *cluster* seria:

Tabela 3 - Sem software

Sem software			
Projeto	**Estrutura**	**Composição**	**Tempo**
1º modelo	6 meses	2 horas	6 meses e 2 h
2º modelo	6 meses	2 horas	6 meses e 2 h
3º modelo	6 meses	2horas	6 meses e 2 h
Total de tempo para projetar 3 *cluster 5'*			*18 meses e 6h*

Se a ferramenta de desenvolvimento (que pode ser usada em qualquer projeto) for utilizada e esta ferramenta durou 6 meses para ser realizada, logo:

Tabela 4 - Com software.

Com software			
Projeto	**Estrutura**	**Composição**	**Tempo**
1º modelo	6 meses	2 horas	6 meses e 2 h
2º modelo	0	2 horas	2 h
3º modelo	0	2horas	2 h
Total de tempo para projetar 3 *cluster 5'*			*6 meses e 6h*

Verifica-se pelas tabelas que o tempo de desenvolvimento com a ferramenta é de um terço do tempo gasto com o desenvolvimento de três modelos distintos de *cluster*, sem o uso da mesma.

Na produção de três modelos, pode-se supor a seguinte equação sem a ferramenta de desenvolvimento:

$$3\ x\ custo = \frac{tempo\ de\ programação}{número\ de\ unidades\ produzidas} + hardware$$

Com os mesmos três modelos, porém com a ferramenta de desenvolvimento:

$$custo = \frac{tempo\ de\ programação/\ 3}{número\ de\ unidades\ produzidas} + hardware$$

Através das equações verifica-se que o custo diminui aproximadamente em 3 vezes, o uso prolongado desse tipo de ferramenta, diminui o tempo de criação em função do número de unidade produzidas, o que demonstra a viabilidade de uso da ferramenta de desenvolvimento.

Ao dar valores para estimar estes gasto, sem a ferramenta de desenvolvimento, teríamos um gasto fixo, para criações de modelos diferenciados, independente da quantidade.

$$3\ \text{x}\ (\text{Custo unitário} = \frac{4\ 000\ 000{,}00}{\text{Nº de unidades}} + \ 475{,}00 < 945{,}30\ (\ 8511\ unidades\ @\ 945{,}30)$$

Este modelo se repetindo ao longo do tempo, com a ferramenta de desenvolvimento, teríamos um gasto variável, para criações de modelos iguais ou diferenciados, uma vez que somente se gastaria com dessenvolvimento da ferramenta uma vez, ou seja com o aumento da quantidade, independente de ser igual ou não, dividiria o valor de projeto por uma grande quantidade a ponto de torna-lo desprezível.

$$\text{Custo unitário} = \frac{4\ 000\ 000{,}00\ +\ 200\ 000{,}00}{\text{Nº de unidades}} + \ 475{,}00 < 945{,}30\ \ =\ \ 8937\ @\ 945{,}30$$

Com o aumento do número de unidades, Restaria praticamente os gastos com hardware.

$$\text{Custo unitário} = \frac{4\,000\,000{,}00 \;+\; 200\,000{,}00}{4\,2000\,000} + 475{,}00 < 945{,}30$$
$$= 4\,200\,000\;\mathit{unidades}\;@\;476{,}20$$

A idéia da ferramenta de desenvolvimento associada ao *hadware* de teste (Figura 38), possibilita a minimização do desenvolvimento, uma vez que o hardware se encontra pronto com kernel e drivers.

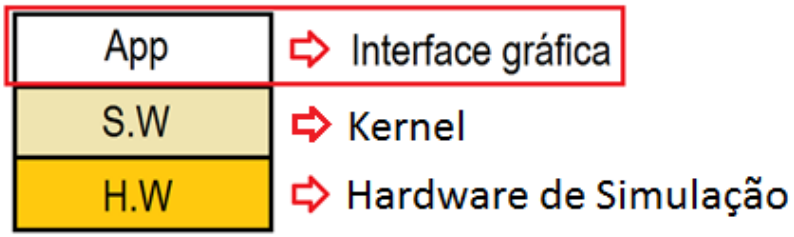

Figura 38 - Estrutura do software.

A criação fica limitada apenas a interface gráfica, considerando uma linguagem de programacao de alto nível, possibilita:

- Melhores recursos gráficos que em C.
- Mais formas e efeitos nas apresentações.
- Menor prazo, Menor preço e a possibilidade de adiantamento de protótipos.

9. CONCLUSÂO

Os painéis de instrumentos com base em *displays* TFT-LCD provê a disponibilidade de um produto capaz de atender uma grande gama de veículos, entre modelos básicos e *premium*, com padronização de *hardware* e distinção de modelos através de pacotes de *software e mascara de acabamento*.

O crescimento desta tecnologia é possível devido a globalização dos veículos cada vez maior, principalmente em países emergentes, em que o custo de desenvolvimento dos *displays* seria condizente com o maior volume de produção.

Com o objetivo de desenvolver estes painéis de instrumentos, se faz necessário uma aplicação de desenvolvimento e simulação, que é o foco principal deste projeto, baseado na idéia produção em laboratorio (*hardware de Simulação*), que possibilita o desenvolvimento sem a presença física de um veículo, sendo necessário apenas a aquisição de dados fornecida pelo mesmo para o funcionamento do instrumento.

Este método possibilita um desenvolvimento transparente e rápido se comparado aos métodos de injeção ou modelagem de estrutura, pois uma vez que a estrutura do *software* e *hardware* já se encontra finalizada, reduz-se o número de protótipos na fase de desenvolvimento, e se faz necessário somente de profissionais de *design* para desenvolver a composição do *cluster*, fato que diminui o tempo de entrega do produto ao cliente e principalmente queda no custo.

A plataforma de desenvolvimento usada no projeto foi capaz de atender as expectativas, como também os sistemas de comunicação de dados, que possibilitou a comunicação entre o *display* e a ferramenta de desenvolvimento do projeto.

Como o projeto não se encontra finalizado, existem funções adicionais a serem implementadas futuramente no programa de desenvolvimento, como:

- Realizar sobreposição de ponteiro (sistema *multilayer*);
- Sugestão de desenvolvimento de um sistema de diagnóstico;
- Implementar comunicação com um veículo real;
- Implementar o HIL para testes do equpamento.

10. BIBLIOGRAFIA

ARDUINO, ArduinoDue. Disponível em: http://arduino.cc/en/Main/ArduinoBoardDue Acesso em: 04 de mai. 2014.

ATMEL. Disponível em: http://www.atmel.com/Images/doc11057.pdf Acesso em: 04 de mai. 2014.

BAPTISTA, J.P. Microcontroladores, 2001

BOSCH, Manual de Tecnologia Automotiva, 25ª Ed, São Paulo, Ed. Edgard Blucher, 2005, 1076 p.

CONTINENTAL. Around it goes: 110 years of the speedometer. Disponível em: http://www.continental-corpora-tion.com/www/pressportal_com_en/themes/press_releases/3_automotive_group/interior/press_releases/pr_2012_10_12_110_years_speedometer_en.htmlAcesso em: 10 de mar. 2014.

DAIMLER. The birth of the automobile.Disponível em: http://www.daimler.com/dccom/0-5-1322446-1-1323352-1-0-0-1322455-0-0-135-0-0-0-0-0-0-0.html Acesso em: 04 de mai de 2014.

Digifiz – Disponível em: http://de.wikipedia.org/wiki/Digifiz. Acesso em 14 de outubro de 2014.

Duble you digital, digital dashboard. Disponível em: http://www.doubleyoudigital.nl/~cars_digital%20dashboards.php Acesso em: 04/05/14.

GANSSLE, J. The art of designing embedded systems, 2º Ed, Newnes, 2008. 75 p.

GRYC, A. Design challenges for digital instrument clusters. Disponível em: http://electronicdesign.com/automotive/design-challenges-digital-instrument-clusters. Acessoem: 04 de mai.de 2014.

HAMMERSCHMIDT, C - Automotive LCD shipments market to run hot, IHS predicts. Disponível em: http://www.electronics-eetimes.com/en/automotive-lcd-shipments-

market-to-run-hot-ihs-predicts.html?cmp_id=7&news_id=222915427&page=0. Acesso em 08 de outubro de 2014

IHS – Instrument Clusters: Time for a Change Disponível em: https://technology.ihs.com/394320/instrument-clusters-time-for-a-change. Acesso em 08 de outubro de 2014.

ISO - No road blocks with ISO standard for dashboard symbols. Disponível em http://www.iso.org/iso/home/news_index/news_archive/news.htm?refid=Ref1343. Acesso em 23 de outubro de 2014

ITDB02 Disponível em: http://imall.iteadstudio.com/display/tft-lcm/im120419008.html Acesso em: 04 de mai. 2014

MISRA, Guidelines for the use of the C language in critical Systems. Nuneaton, England, 2004, 13 P.

RUSSEL, D. Introduction to embedded systems: using ANSI C and the Arduino Development Environment. Mitchell Thornton Series editor, 2010, 23 p.

Sem título. Disponível em: https://plus.google.com/photos/101103003601736065377/albums Acesso em: 04/05/14.

SLEMBROUCK, P. V. - Speedometer Design: Why It Works. Disponível em: http://www.paulvanslembrouck.com/2012/speedometer-design-that-thing-in-your-car/. Acesso em 12 de outubro de 2014.

SOBEY E. A field guide do automotive Technology, Chicago, Chicago Review Press incorporated, 2009, 78 p.

SSD1963. Disponível em: http://www.microtipsusa.com/pdf/SSD1963.pdf Acesso em: 04 de mai. 2014.

Visual Studio. Disponível em: http://www.visualstudio.com. Acesso em: 04 de mai. 2014.

WESNER, G. From speedometer to modern instrument cluster, 2005

Which car has the world´s Best instrument cluster? Disponível em: http://www.carthrottle.com/which-car-has-the-worlds-best-instrument-cluster/Acesso em: 04/05/14.