Faculdade de Tecnologia de Americana
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

DESENVOLVIMENTO DE UM VOLANTE UTILIZANDO ARDUINO

VICTOR AMARAL DO NASCIMENTO

Americana, SP
2013

Faculdade de Tecnologia de Americana
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

DESENVOLVIMENTO DE UM VOLANTE UTILIZANDO ARDUINO

VICTOR AMARAL DO NASCIMENTO
Victor.a.nascimento@hotmail.com

Trabalho Monográfico, desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Fatec-Americana, sob orientação do Prof. Me. Kléber de Oliveira Andrade

Área: Análise e Desenvolvimento de Sistemas

Americana, SP
2013

BANCA EXAMINADORA

Professor: Prof. Me. Kleber de Oliveira Andrade (Orientador)

Professor: Prof. Drª Mariana Godoy Vazquez Miano

Professor: Prof. Adnan Bakri

DEDICATÓRIA

Dedico este trabalho a Deus, aos meus pais, pelo apoio e paciência, aos meus amigos que aqui fiz e que eternamente estarão em minhas lembranças.

RESUMO

Visando uma melhor interatividade entre usuários e os games, este trabalho mostra o desenvolvimento de um joystick criado especificamente para jogos de corrida. Os capítulos foram separados por etapas de criação, para que fiquem bem claros cada fase de desenvolvimento do projeto, como: parte elétrica, mecânica e lógica. Temas como testes e projetos futuros também serão abordados ao final da introdução.

Palavras Chave: gameplay, USB communication, Arduino, wheel games.

ABSTRACT

Seeking a better interactivity between users and games, this work shows the development of a joystick designed specifically for racing games. Steps of creation separated the chapters, to be clear each stage of project development, such as electrical, mechanical and logical. Topics such as testing and future projects will also be discussed at the end of the introduction

Keywords: Joystick, racing, games.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Graças à evolução tecnológica, está surgindo rapidamente novos instrumentos para auxiliar as pessoas em suas tarefas diárias e até mesmo na diversão. Os seres humanos de uma forma geral procuram se divertir nos momentos de lazer, seja conversando, contando anedotas, brincando, jogando em seus aparelhos móveis (celulares, *tablets*, etc.) ou jogando v*ideogames*[1] com seus respectivos dispositivos de entrada, conhecidos por *joysticks*.

O termo *joystick* ao longo dos tempos virou um sinônimo para os controladores de jogos que poderiam ser conectados ao computador. Os *joysticks* podem ser classificados em três categorias: *joystick* básico, *gamepads* e volantes. Eles têm geralmente um ou mais botões de pressão cujo estado pode ser lido pelo computador.

Os volantes são utilizados nos jogos de corrida para aumentar o realismo e a interatividade, proporcionando mais diversão para quem gosta de pilotar carros. Visando aumentar a imersão dos usuários que gostam de corridas, este trabalho tem por objetivo estudar e desenvolver um *joystick* do tipo volante para o mesmo, utilizando o Arduino que enviará informações através da porta USB. O trabalho mostra a construção do mesmo, apresentando os principais desafios e os detalhes mais relevantes.

Para viabilizar o desenvolvimento deste projeto, foi necessário fazer: um levantamento de requisitos para especificar quais funções seriam implementados no volante; desenvolver o volante físico e a sua eletrônica, baseado nos levantamentos realizados para o projeto; discutir se as funcionalidades dos usuários foram supridas, objetivando a melhor interatividade e usabilidade do mesmo.

O restante do trabalho foi estruturado em três capítulos, sendo que o capítulo dois conceitua a base necessária para desenvolver o volante. O capítulo três descreve

[1] Um jogo eletrônico ou também vídeo game no Brasil e jogo de vídeo ou videojogo, em Portugal.

os passos realizados para o desenvolvimento do *joystick*. Por fim, o capítulo quatro apresenta as considerações finais e os trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo são abordados os temas básicos, para um bom entendimento de como funcionam os denominados sistemas embarcados.

2.1 SISTEMAS EMBARCADOS

Com o avanço da tecnologia, cada vez mais produtos integrados com sistemas embarcados veem ganhando espaço e inovando o mercado. Segundo Marimoto (2007), mesmo tendo muitos componentes similares ao de um computador, tais como: processador, memória e algum tipo de armazenamento, os sistemas embarcados (SE) priorizam o trabalho de apenas uma tarefa, podendo assim, ser muito mais eficiente em determinada tarefa do que um PC, que trabalha com diversas tarefas em paralelo.

Andando pela rua, é possível encontrar diversos sistemas embarcados, porém, eles estão tão presentes na vida das pessoas, que muitas vezes passam despercebidos por seus olhos. Caminhando pela rua, provavelmente pessoas se deparam com semáforos, postes de iluminação, cancelas automáticas, portões elétricos entre outras coisas. Tudo isso pode ser definido como um sistemas embarcado, que desempenham tarefas pré-estabelecidas para efetuar algum tipo de trabalho. Como por exemplo: o abrir e fechar de um portão elétrico, a troca entre luzes nos painéis dos semáforos e o acender da luz de um poste logo que a noite cai e a luminosidade fica menor. De acordo com Marimoto (2007), um Sistema embarcado não tem tamanho nem forma especifica, portanto pode estar presente desde um brinquedo para crianças como um Furby (Figura 1), até mesmo em painéis altamente complexos para controle de aviões (Figura 2).

Figura 1. Exemplo de sistemas embarcados, brinquedo Furby.
Fonte: http://jenstayrook.com/2012/afurbychristmas

Figura 2. Controlador de trafego aéreo.
Fonte: http://www.eldigitaldecanarias.net/noticia26555.php

O custo é um fator que influencia muito no produto, e com o avanço da tecnologia foram desenvolvidos chips de alto desempenho que podem minimizar o preço final. Marimoto (2007), explica que isso ocorre porque diferente dos chips antigos, os chips mais atuais têm o poder de controlar mais de uma tarefa. Por exemplo: um mp4 player funciona perfeitamente com o uso de apenas três chips (Figura 3), sendo eles: um microcontrolador, uma memória flash (utilizada para armazenamento) e o terceiro um sincronizador de rádio AM/FM, que poderia ser retirado do sistema sem implicar em nenhum funcionamento básico a não ser a funcionalidade do rádio.

Esta tecnologia ajuda muito na redução do preço, pois ao invés de utilizar diversos chips para cada funcionalidade do sistema, é utilizado apenas um controlador que possui o poder de controlar diversas tarefas embutidas. Delai, (2013) diz que ao se destrinchar um projeto com um sistema embarcado, pode-se ver com clareza a unidade de processamento, memória e periféricos (Figura 4).

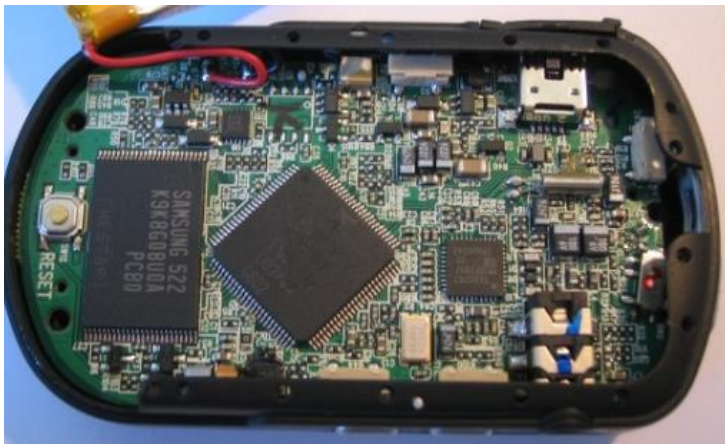

Figura 3. MP4 player usando apenas três chips.
Fonte: http://www.hardware.com.br/artigos/entendendo-sistemas-embarcados/

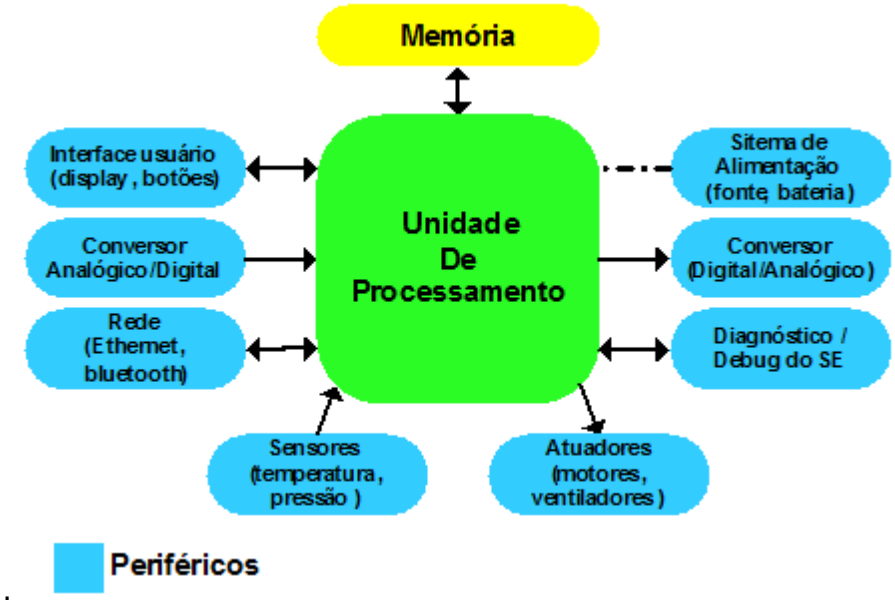

Figura 4. Exemplo de subdivisão de um sistema embarcado.
Fonte: http://www.hardware.com.br/artigos/sistemas-embarcados-computacao-invisivel/conceito.html

Delai (2013) explica que a unidade de processamento executa instruções responsáveis por realizar cálculos, tomar decisões e tratar eventos. Possui normalmente a arquitetura elementar clássica de um processador de computador convencional, como a unidade lógica/aritmética (ULA, que tem a função de realizar cálculos de soma, divisão, subtração, verifica se um numero é positivo, negativo ou

igual a zero, e comparar números e letras), unidade de controle (UC, responsável por gerar todos os sinais que controlam as operações no exterior do CPU), registradores (unidade de memória capaz de armazenar bits), etc. De acordo com Jordão (2012), a função da memória é guardar dados temporariamente para que o processador possa acessar essas informações de forma muito mais rápida. Delai, (2013) afirma que os periféricos são interfaces da unidade de processamento com o mundo externo. Um exemplo é o sensor de temperatura, que capta a temperatura do ambiente e envia as informações para o processador.

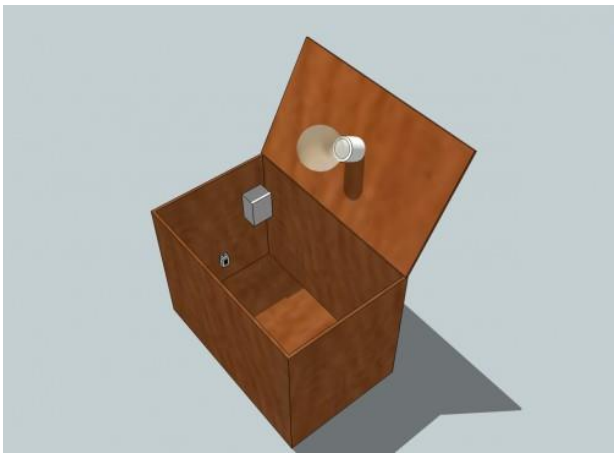

Figura 5. Estufa para manter papéis aquecidos.
Fonte: http://www.hardware.com.br/artigos/sistemas-embarcados-computacao-invisivel/conceito.html

A finalidade de um sistema embarcado é controlar processos. Ou seja, trabalhar em cima de algum problema (DELAI, 2013). Duas categorias de periféricos se destacam em um SE, os sensores e os atuadores. Segundo Borges e Dores (2010) *apud* por Beock (2011) os sensores quando operam de forma direta, transformando uma forma de energia em outra são chamados de transdutores. Os sensores onde as operações ocorrem de forma indireta alteram suas propriedades, como a resistência, capacitância ou indutância, sob a ação da grandeza de forma que essa alteração ocorre mais ou menos proporcional. Os atuadores fazem com que o SE possa intervir o meio onde atua. Como o próprio nome diz, são dispositivos que possuem o poder de interferir no processo em controle. Alguns exemplos são: Motores, luzes, aquecedores e chaveadores. São periféricos que enviam informações do SE para o processo. Um exemplo de sistema embarcado relatado por Delai (2013) é a criação uma caixa estufa para manter papéis sem umidade (Figura 5) para obter uma maior qualidade no uso do papel ao realizar cópias. A parte mecânica do projeto consiste

em uma caixa de madeira com uma medida relativa ao tamanho do papel utilizado, e com altura planejada para que caibam todos os componentes elétricos e também uma dobradiça para uma porta, que servirá para manter a estufa em ativação.

Foi utilizado um chip microcontrolador com controle de memória já embutida. Como atuador, foi utilizado uma lâmpada incandescente de 40 W, acionada por intermédio de um relê interligado ao chip microcontrolador, que irá fornecer o calor necessário para que o papel se mantenha na temperatura desejada. O sistema consiste em malha aberta, que segundo Wikipédia (2013) consiste em um sistema que não possui realimentação. Um atuador aplica um sinal de entrada esperando que um valor desejado seja atingido na saída deste sinal em uma variável ou tome determinada ação referente ao processo, por isso não existe um sensor de umidade embutida no projeto. Uma programação foi feita no microcontrolador onde ele determina um tempo para que a lâmpada fique acesa e um tempo para que ela fique apagada. Suponha-se que, com base em testes realizados, foi determinado um tempo para que a lâmpada se mantenha acesa por três minutos, e em seguida ela se mantem apagada por quinze minutos, tudo isso para que o papel se mantenha na temperatura desejada. O fluxograma de execução do programa (Figura 6) ilustra esse processo.

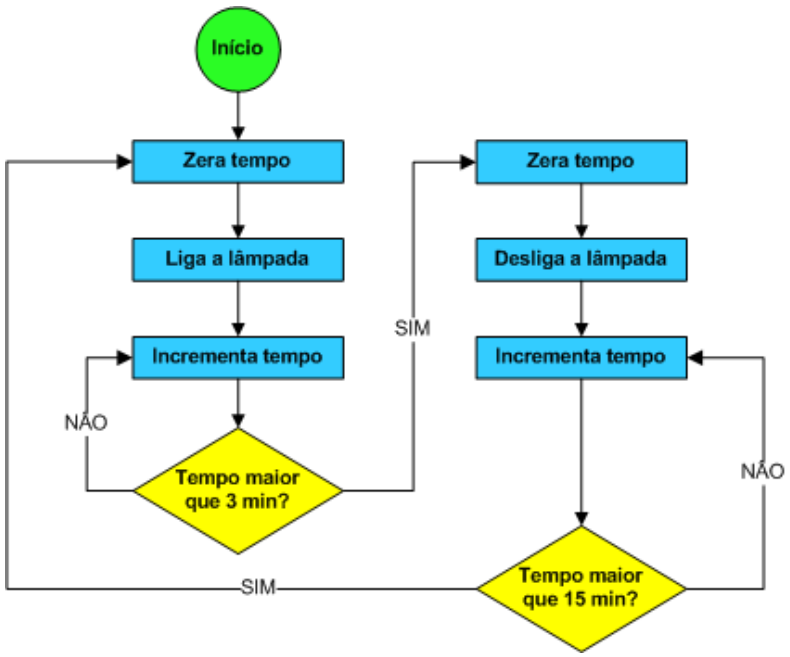

Figura 6. Fluxograma de operação com funcionamento em malha aberta.
Fonte: http://www.hardware.com.br/artigos/sistemas-embarcados-computacao-invisivel/pequeno-exemplo-de-se.html

Um sistema em malha fechada (Figura 7) terá quase os mesmo componentes do projeto citado acima, porém, também contaria com um sensor de umidade, que retornaria o nível de umidade dentro da estufa. A diferença no funcionamento seria que, ao invés do sistema ter um tempo pré-estabelecido para ligamento e desligamento da lâmpada, isso seria uma função para o sensor de umidade. Portanto, o sistema teria um controle muito mais eficaz no processo de manter o papel sempre na temperatura esperada.

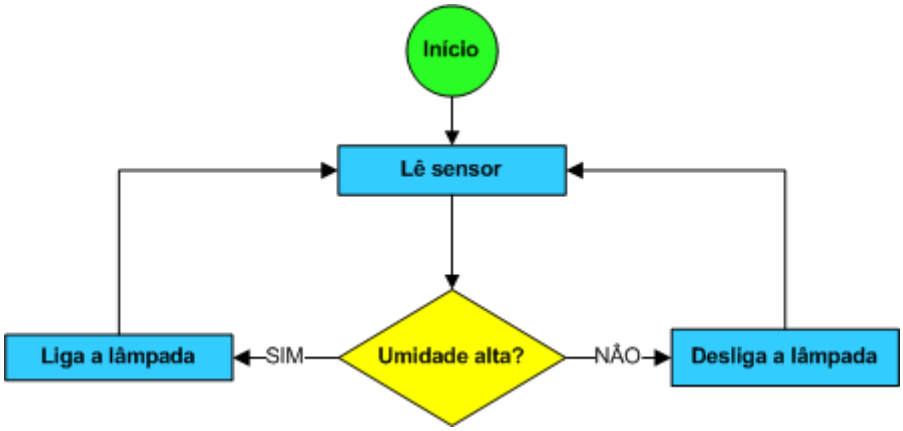

Figura 7. Funcionamento de operação com malha fechada.
Fonte: http://www.hardware.com.br/artigos/sistemas-embarcados-computacao-invisivel/pequeno-exemplo-de-se.html

O sistema com malha aberta tem uma desvantagem evidente em relação ao de malha fechada, pois em caso de variações de temperatura no ambiente, um dia chuvoso, por exemplo, o sistema continuará com o tempo pré-estabelecido funcionando da mesma maneira a um dia ensolarado. Já no sistema integrado com malha fechada, apesar de um custo um pouco mais elevado, terá o controle exato da temperatura da caixa, independente de variações de ambiente.

2.1.1.1 Sistema em Tempo Real

Como seria viver em um mundo sem relógios, sensores de temperatura, câmeras fotográficas, ou até mesmo sem meios de transportes eletrônicos como carros, barcos ou aviões? Shaw, (2001) afirma que os sistemas em tempo real, são sistemas que tem a necessidade de uma resposta imediata. Isto é, ao requisitar uma ação, o sistema deve executa-la com o menor tempo possível.

É possível imaginar um sistema em um hospital em que o paciente depende dele para que seu coração continue pulsando. De repente, por um atraso na execução de alguma tarefa o sistema tem um atraso de cinco minutos. Isso implicará na finalidade do sistema, acarretando talvez na morte do paciente. Outro exemplo são as pernas de um robô. Suponha-se que ao dar um passo o sistema tenha um atraso de apenas alguns segundos, se as duas pernas não são executadas de forma sincronizadas, com certeza o robô cairá.

2.2 JOYSTICKS

De acordo com Harris (2013) o joystick (Figura 8) é uma ferramenta que permite a conversão de movimentos naturais para sequencias matemáticas variando entre zero e um. Isso facilita muito em certas tarefas em que é necessário controlar algo utilizando apenas movimentos. A maior vantagem de um joystick é realizar tudo isso sem que o usuário ao menos perceba que está utilizando-o. Neste capitulo será mostrado como ocorre esta conversão, como é feita a comunicação e também como é o funcionamento variando dos diversos designes.

Figura 8. Imagem de um joystick padrão.
Fonte: http://eletronicos.hsw.uol.com.br/joystick.htm

2.2.1 Comunicação

O dispositivo em que o joystick está conectado tem como função interpretar essas variações entre zero e um em coordenadas para a realização do movimento. Suponhamos que o bastão do joystick seja inclinado para direita, Harris (2013) explica que o acontecerá, é que a chave de ativação (Figura 9 e 10) da direita será pressionada pelo seu próprio mecanismo, fazendo assim, com que esta chave envie o sinal um (pressionado), enquanto os demais continuam com o valor zero (não pressionado). Existe também a possibilidade de movimentos diagonais, que seriam a combinação de duas chaves pressionadas. Por exemplo. Caso o bastão esteja inclinado para um movimento diagonal sentido para cima/para esquerda, o que acontecerá é que as chaves ativadas pelo mecanismo sejam a chave da esquerda e a chave para cima. Os botões para disparo funcionam da mesma maneira, porém seu mecanismo de ativação é diferente da do bastão.

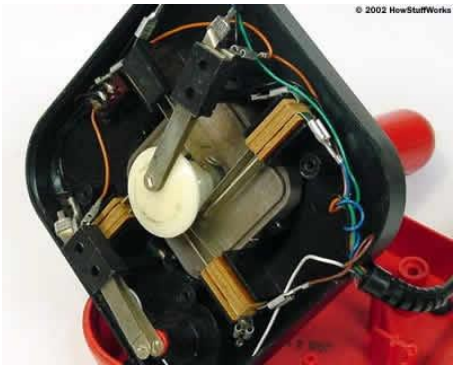

Figura 9. Joystick chaves de ativação de um joystick.
Fonte: http://eletronicos.hsw.uol.com.br/joystick2.htm

Este modelo de joystick pode apenas interpretar sinais absolutos, ou seja, ele não é capaz de medir a intensidade em que o bastão ou os botões foram acionados. Portando é um modelo simples onde é de fácil manuseio para jogos não tão complexos como Tetris ou Pacman.

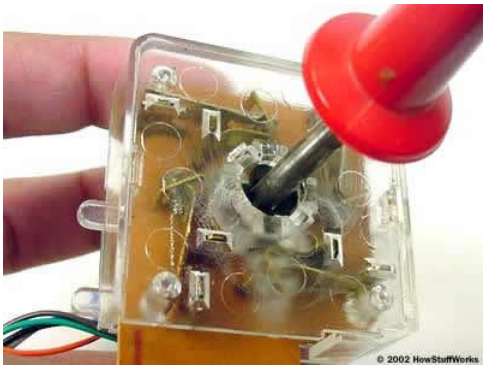

Figura 10. Bastão utilizado para direcionar a posição do joystick.
Fonte: http://eletronicos.hsw.uol.com.br/joystick2.htm.

2.2.2 Funcionamento de um Joystick

A interpretação acontece de modo em que o bastão seja interpretado pelo seu dispositivo de conexão como um eixo X e outro Y, de forma que o eixo X concede a percepção de que o bastão foi inclinado na horizontal, e o eixo Y na vertical. Harris, (2013) afirma que como na geometria, as coordenadas X e Y determinam a posição do movimento feito com o bastão. Quando o bastão é inclinado para algum lado, o que acontece é que a chave de ativação do lado correspondente será ativada e interpretada. Por exemplo: caso o bastão seja inclinado para a direita, a chave de ativação do lado direito (eixo X) terá o valor igual a um. Porém se o bastão for inclinado para cima, a chave de ativação de cima (eixo Y) retornará o valor um, indicando que a direção a ser interpretada é para cima. Quando o bastão é inclinado na diagonal, ambas as chaves são ativadas. Por exemplo: caso seja inclinado para cima/para esquerda, as chaves que serão ativadas serão as correspondentes, ou seja, eixo X chave1 e eixo Y chave 1. O sistema em que o joystick está conectado simplesmente monitora o valor de seus eixos e chaves correspondentes.

O Joystick analógico tem sua funcionalidade um pouco diferente, pois ao invés de botões são utilizados potenciômetros para que o movimento possa ter intensidades diferentes. O potenciômetro é um componente elétrico e analógico com duas extremidades ligadas através de uma trilha resistente de carvão. Possui também um cursor interligado ao eixo, que informa a resistência da posição do eixo perante a voltagem em que o carvão foi ligado.

O computador irá interpretar esse valor determinado pela posição do cursor e transformará em informação da intensidade em que o bastão foi inclinado. Segundo Harris, (2013) conforme giramos o eixo do potenciômetro, o cursor também gira relativamente no carvão. Em uma das pontas não há resistência, e conforme o alcance da outra extremidade seu valor de resistência vai aumentando. Geralmente o terminal fixo do meio, é onde a resistência relativa ao carvão e o cursor se encontra. Por isso, ao ligar um positivo no primeiro terminal fixo, e um negativo em outra extremidade, o terminal do meio retornará o valor de resistência referente a posição do eixo.

2.3 ARDUINO

Ao pensar em Arduino (Figura 11), vem logo em mente a possibilidade de infinitas criações e inovações de projetos. Por ser uma plataforma Open Source, vem ganhando cada vez mais espaço no mercado, podendo fazer desde um simples acender e apagar de um led, ou até mesmo um robô com diversas funções complexas. De acordo com Fonseca (2010) e Vega (2010) Arduino é uma plataforma de computação física, onde um sistema digital com ajuda de sensores e atuadores, é capaz de perceber realidades físicas, processar informações e assim responder conforme o resultado de determinado processo.

Fonseca (2010) e Vega (2010) afirmam que um microcontrolador, é basicamente um computador que também possui processador, memória e periféricos de entrada e saída. Eles são pré-programados para realizar tarefas específicas e podem interagir com outros microcontroladores embarcados em diferentes sistemas. Segundo Fonseca (2010) e Vega (2010) o Arduino é uma ferramenta de entrada e saída de dados, onde é possível ser acionadas através de sensores e ações pré-programadas. Por exemplo: depois da detecção de baixa temperatura em um sensor, uma lâmpada será acesa.

Figura 11. Arduino Uno.
Fonte: http://meetarduino.wordpress.com/2012/05/04/arduino-aumentando-numero-de-portas-output/

Com o seu funcionamento similar ao de um computador que comunica com periféricos de entrada e saída como: Mouse, teclado e monitor, o Arduino também se comunica com seus mais variados periféricos. Para entender mais sobre o processo e atuação, é possível identificar os elementos principais do circuito através do diagrama de bloco (figura 12).

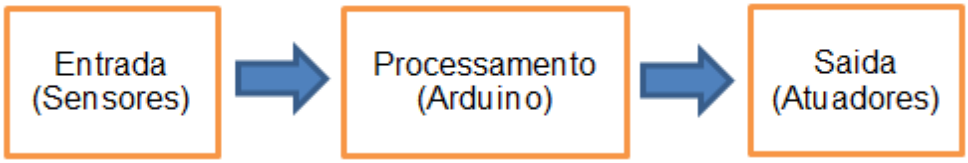

Figura 12.Diagrama de bloco.
Fonte: Do autor

O Arduino é baseado em um microcontrolador (Atmega), dessa forma pode ser logicamente programado, ou seja, é possível a criação de programas, utilizando uma linguagem própria baseada em C/C++ (Fonseca (2010) e Vega (2010)).

A vantagem de se usar o Arduino é que ele possui diversas comunidades para troca de informações e projetos, portanto, fica muito mais fácil o domínio e aprendizagem. Fonseca (2010) e Vega (2010) dizem que ele foi projetado com uma finalidade de ser de fácil entendimento e manuseio, além de ser um circuito multiplataforma, que significa que deve ser compatível com Windos, Linux, Mac OS entre outros sistemas. Portanto, pode ser utilizado de forma didática sem o receio de limitações dos diferentes sistemas dos alunos.

A alimentação do Arduino pode ser feita de duas formas: através de porta USB, ou externamente (com o uso de fontes ou baterias). Possui uma saída de cinco Volts e conta com o auxílio de diversas portas digitais e analógicas dependendo da versão. No caso de uso de alimentação externa, é recomendado uma voltagem de sete até 12 Volts. Entretanto, no caso de ser uma alimentação abaixo de 7 V, o Arduino pode fornecer menos Voltagem para sua saída de cinco V e a placa pode ficar instável. Se a alimentação for superior a maior do que 12 V, o regulador de tensão pode superaquecer e variar a placa.

Os pinos de alimentação são:

- V in: É a entrada de alimentação da placa quando uma fonte externa for utilizada. O Arduino pode também fornecer alimentação por este pino.
- 5 V: É a fonte de alimentação utilizada pelo microcontrolador e outros componentes da placa.
- V3: uma alimentação de 3,3 V fornecida pelo circuito integrada FTDI (controlador USB) que possui corrente máxima de 50 mA.
- GND (ground): é o pino terra.

De acordo com Fonseca (2010) e Vega (2010) O ATmega328 tem 32 KB de memória flash, 2 KB de SRAM e 1 KB de EEPROM (esta última pode ser lida e escrita através da biblioteca EEPROM, que guarda os dados permanentemente, mesmo que desliguemos a placa). A memória SRAM é apagada toda vez que desligamos o circuito.

Os pinos digitais podem ser usados de maneira de entrada a dados ou por saída de tensão. Tudo isso depende do que o desenvolvedor precisa. Essa escolha é feita por meio de programação, e pode ser facilmente configurada devido as funções pinMode(), digitalRead() e digitalWite(). Os pinos digitais trabalham com uma voltagem de 5 V, e podem fornecer até 40 mA. Os pinos analógicos são semelhantes aos digitais, porém recebem valores analógicos, cada porta é ligada em um conversor analógico-digital de 10 bits que transformam a leitura em um valor inteiro de 0 até 1023.

Segundo Fonseca (2010) e Vega (2010), no Arduino, a comunicação entre um computador, outro Arduino ou até mesmo outro microcontrolador, acontece de forma muito simples. O ATmega328 possibilita comunicação serial no padrão UART TTL (5 V), que está disponível nos pinos zero (RX) e um (TX). Através de um chip FTDI FT232RL, é possível estabelecer uma comunicação com a porta USB, e os drivers FTDI (incluído no software do Arduino) fornecem uma porta virtual para o software no computador. O software do Arduino possui um monitor virtual onde é possível monitorar as informações de entrada e saída da serial. Isso facilita muito no desenvolvimento e nos testes de qualquer projeto. O ATmega328 também oferece suporte aos padrões de comunicação I2C (TWI) e SPI. O software do Arduino inclui uma biblioteca Wire para simplificar o uso do barramento I2C (Fonseca, 2010) e (Vega, 2010).

3 PROJETO DO VOLANTE

O projeto consiste em criar um joystick para jogos de corrida, com uma ergonomia desenvolvida para proporcionar ao jogador uma sensação melhor de controle e liberdade. Funções básicas como: acelerar, frear e trocar de marchas foram embutidas no próprio volante, para que o usuário possa ter em mãos todos os recursos do jogo em relação a corrida.

3.1 MATERIAIS

Neste capítulo serão mostrados os principais componentes que compõem o projeto, visando a qualidade e viabilidade em custos.

3.1.1 Materiais Elétricos

Para as funções: acelerar, frear, aumentar/marcha e diminuir/marcha, foram usados *Push Buttons*, (Figura 13.a) pois os mesmos são de fácil manuseio e suprem perfeitamente as necessidades específicas destas tarefas. Dois motores elétricos (Figura 13.b) posicionados em cada extremidade em que o usuário está em contato, foram usados com um leve peso na ponta, para proporcionar ao jogador uma sensação de realismo em casos de colisões ou más condições de terreno. Quatro displays de led com 7 segmentos, (Figura 13.c) com intermédio de 4 expansores de portas, (Figura 13.d) foram utilizados para obter retornos visuais ao jogador. Tais como: velocidade e RPM (Rotação por minuto) auxiliando o volante na prestação de mais uma serviço, beneficiando quem o manuseia. O que possibilita a execução de curvas feitas pelo volante é o potenciômetro, (Figura 13.e) já que o mesmo tem poder de fornecer diversos valores de corrente devido a sua fita resistora.

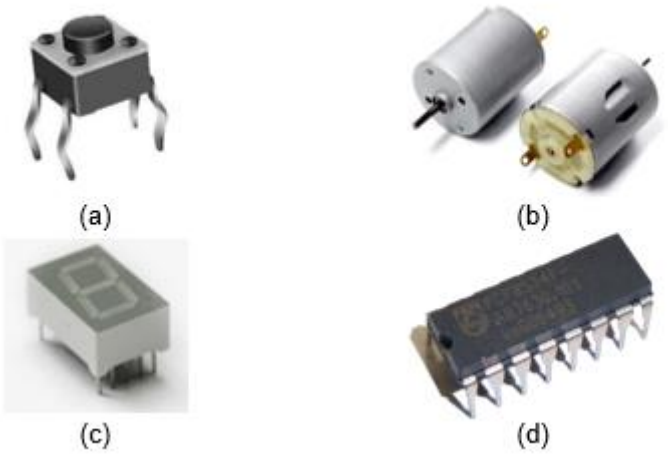

Figura 13 Componentes utilizados: (a) Botão; (b) Motor elétrico; (c) Display de led de 7 Segmentos; (d) Expansor de Portas(PCF8574A).
Fonte: Do autor

3.2 DESENVOLVIMENTO

O desenvolvimento do volante será mostrada neste capítlo, contando com ajuda de imágens e floxogramas para um melhor entendimento.

3.2.1 Parte Elétrica

Com o intuito de causar uma sensação de mais realismo ao usuário, foram ligados 2 motores em duas portas digitais do Arduino, para que possam ser ativadas ou desativadas sempre que houver necessidade (Figura14). Os botões usados como acelerador, freio, subir/marcha e descer/marcha foram ligados em 5 volts com um fio secundário usado para leitura na extremidade positiva (Figura 15). Este fio (Fio amarelo) conectado a uma porta digital configurada para entrada de dados, é usado pelo Arduino para leitura do status do botão (LIGADO/DESLIGADO). Um resistor de pull up foi integrado no circuito, pois sem ele a corrente ficaria instável e a leitura exata do status do botão seria impossível, pois o status variaria seu valor constantemente entre 0 e 1.

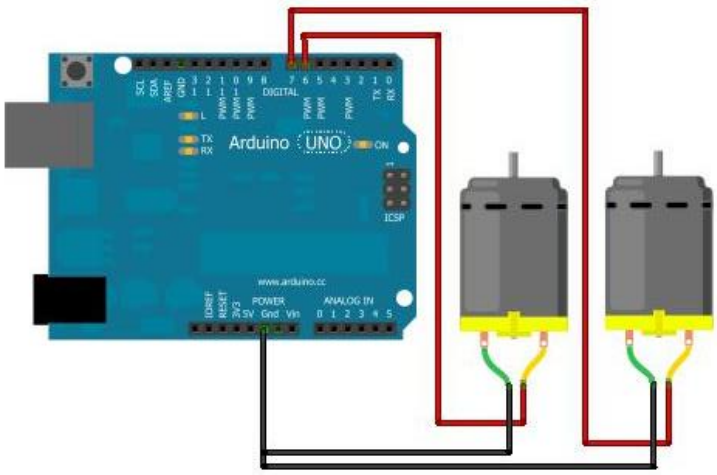

Figura 14 Ligamento dos motores
Fonte: Do autor.

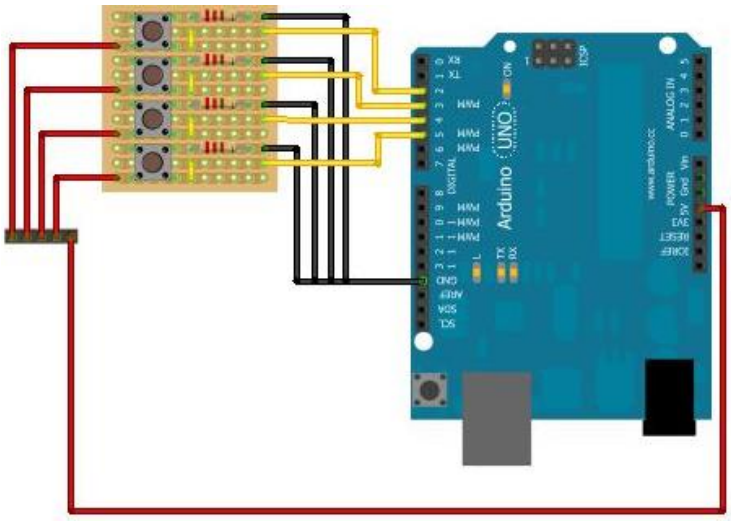

Figura 15 Ligamento dos botões
Fonte: do autor

Foram usados 3 leds de 7 segmentos para indicar a velocidade atual em que o carro se encontra. Porém, cada um dos componentes necessitam de 7 portas digitas, e é ai que um expansor de portas é necessário.

O I2C é uma sigla para Inter-Integrated Circuit que é um protocolo de comunicação parecido com Serial, onde também possui dois fios, sendo um para

comunicação e outro para sincronia. O I2C cria pontos de BUS, necessários para a comunicação com diversos dispositivos. Esses pontos tem a mesma analogia de uma linha de ônibus, onde cada ponto de parada é referente ao endereço de um dispositivo.

Os dispositivos no BUS compartilham os mesmos fios SDA (comunicação), SCL (sincronismo), onde o máster pode se comunicar com os demais slaves. Diferente de uma comunicação serial, o fio SDA do I2C é tanto para leitura quanto para escrita. Como mostrado na (Figura16).

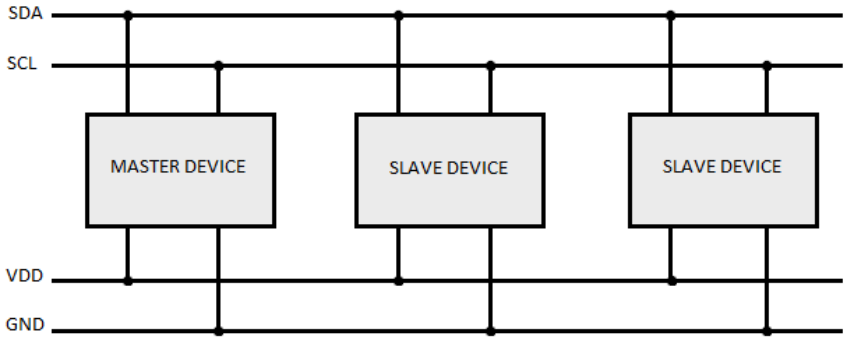

Figura 16 Compartilhamento dos fios no BUS.
Fonte: Do autor

O modo de endereçamento do I2C utilizando o componente PCF8574A possui quatro pinos de endereço fixo (endereço interno), três pinos para endereço externo e um pino para identificação de leitura ou escrita, onde o endereço fixo combinado com o endereço externo forma um endereço slave como mostrado na (figura 17).

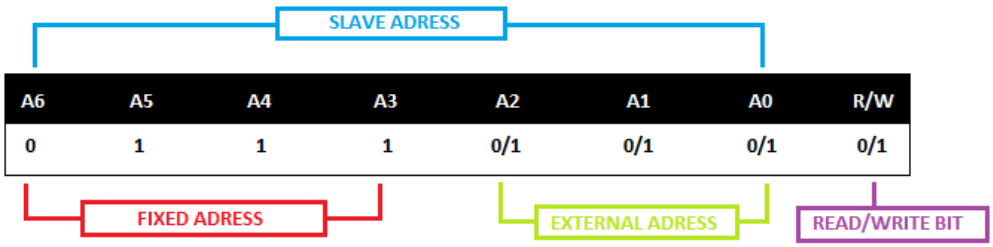

Figura 17 Endereço do componente
Fonte: Do autor

O endereço fixo é essencial para a identificação do dispositivo que será acessado no começo da comunicação. O I2C é uma sigla para Inter-Integrated Circuit que é um protocolo de comunicação parecido com Serial, onde também possui dois

fios, sendo um para comunicação e outro para sincronia. O I2C cria pontos de BUS, necessários para a comunicação com diversos dispositivos. Esses pontos tem a mesma analogia de uma linha de ônibus, onde cada ponto de parada é referente ao endereço de um dispositivo.

Os dispositivos no BUS compartilham os mesmos fios SDA (comunicação), SCL (sincronismo), onde o máster pode se comunicar com os demais slaves. Diferente de uma comunicação serial, o fio SDA do I2C é tanto para leitura quanto para escrita. Como mostrado na (Figura 18).

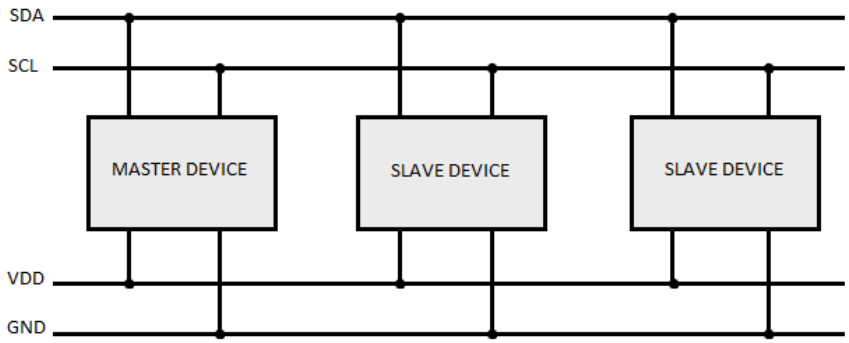

Figura 18 Compartilhamento dos fios no BUS
Fonte: Do autor

A conexão I2C acontece nos pinos 14 e 15, ambas interligadas por dois resistores de pull up de (VALOR DO RESISTOR). A ligação do Shift Register contem com um capacitor de (100 nf) interligado entre o positivo e o negativo. As configurações de external adress dos componentes foram: 000; 111; 110. Portas de endereçamento são ligadas a cada segmento do display para que possa ocorrer o gerenciamento dos mesmos (Figura 19). O pino 14 é ligado na porta analógica 5 do arduino, já a porta 15 é ligada na porta analógica 4. Após todas as conexões feitas, o circuito estará pronto. (Figura 20).

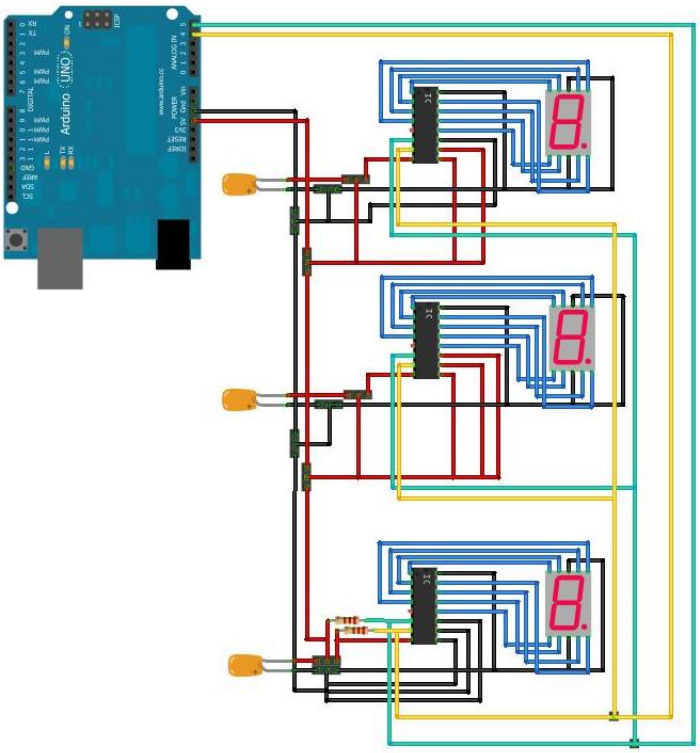

Figura 19 Ligamento dos leds
Fonte: Do autor

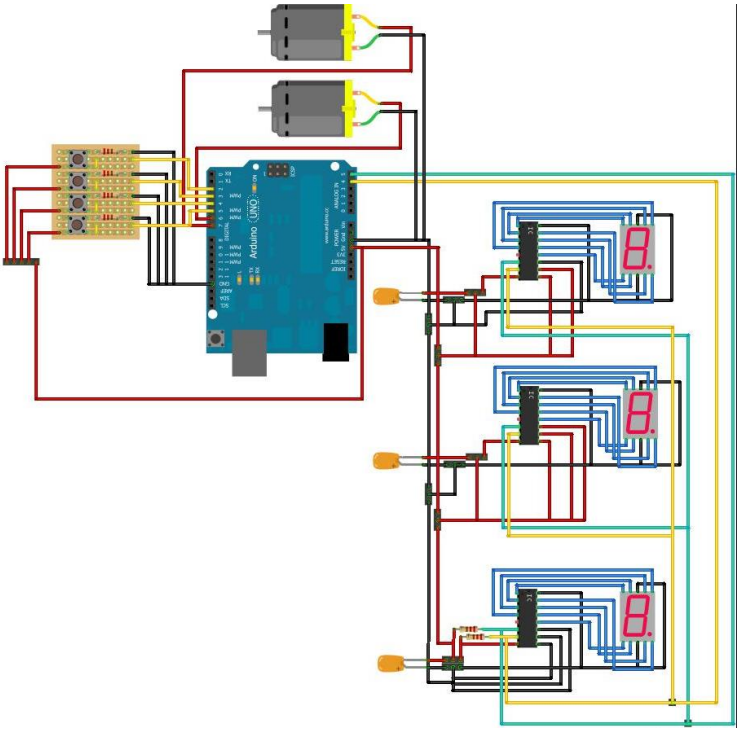

Figura 20 Circuito completo
Fonte: Do autor

3.2.2 Parte Lógica

O funcionamento lógico do projeto consiste em entrada e saída de dados. Portanto, o circuito ficará sempre na escuta caso haja alguma informação nova a ser atualizada. Um fluxograma (Figura 21) mostra todo o ciclo lógico do projeto.

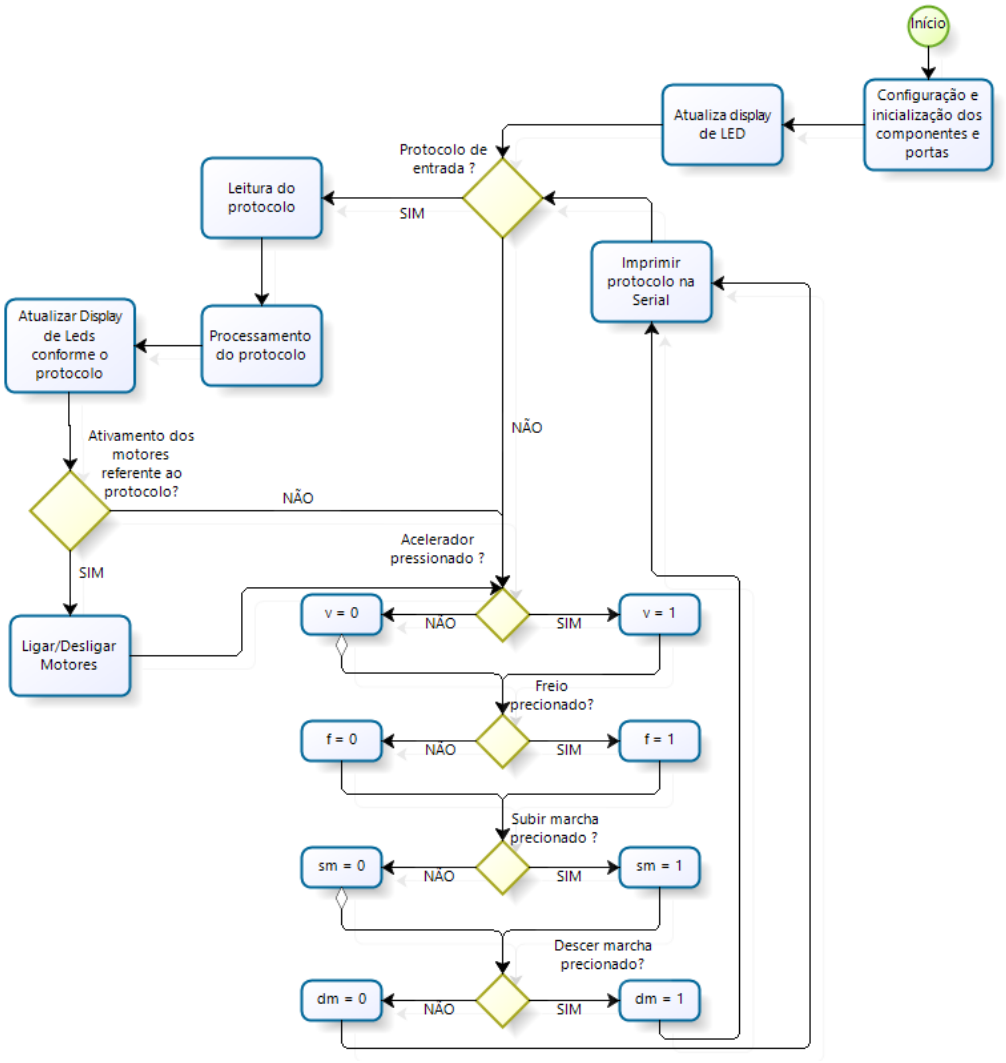

Figura 21 Fluxograma do sistema
Fonte: Do autor

Os protocolos de entrada e saída foram criados com o intuito de evitar possíveis falhas de comunicação ou perda de dados. Os mesmos são formados por uma string contendo as informações necessárias para cada protocolo. O protocolo de saída terá como padrão o envio do volante(v), acelerador(a), freio(f), subir marcha(x) e descer marcha(y) exatamente nessa sequência. Portanto a string ficará semelhante a esta: V---A-F-X-Y-. O tratamento de erros funciona por garantir que a string esteja exatamente como a padronizada, portanto pode-se ter certeza de cada dado sendo processado, substituindo apenas os valores (indicados por "-"). O protocolo de entrada contara com apenas duas informações, o status do motor(m) e os valores do display de led(d), fazendo com que a string seja recebida neste parâmetro: M-D---, onde o M fará parte da informação do motor e o D dos displays.

Ao ser ligado, a primeira coisa a ser feita pelo sistema é a configuração das portas de entrada e saída e a inicialização dos componentes. Logo em seguida será atualizado os displays de leds com o valores que informam a velocidade atual do carro. Depois de todas as configurações feitas o sistema entrará em um loop infinito até que o sistema seja desligado. A primeira condição do loop é a verificação do protocolo de entrada entrando pela porta serial, esse dado é enviado do jogo para o Arduino, e sua finalidade é atualizar os displays de led e ligar ou desligar os motores. Se houver algum dado de entrada, o sistema fará a leitura desse protocolo os processará e finalmente irá atualizar os displays e o status dos motores.

O próximo passo é a verificação e execução da ativação/desativação dos motores. Feito isso, o próximo passo do ciclo é verificar o status da posição do volante e logo em seguida verificar se o botão referente ao acelerador foi pressionado. Se estiver pressionado seu status mudara imediatamente para 1. Caso contrário, imediatamente para 0. Com a leitura do acelerador já feita o sistema passara para a leitura do próximo botão até que chegue ao último, como mostrado na (Figura 21). Então com todos os dados coletados e o protocolo de envio completo, o sistema ira imprimir o protocolo através da porta serial, que será interpretada pelo jogo e começará o ciclo novamente.

4 EXPERIMENTOS

Foram desenvolvidos diversos softwares com intuito de analisar a qualidade do projeto, testar e identificar possíveis falhas. Os principais testes foram baseados em tratamento de protocolos enviados e recebidos, e tempo de respostas. As (Figuras 22 e 23) mostra um teste de confiabilidade das informações enviadas do Arduino para o jogo. Onde a (Figura 22) mostra o retorno do potenciômetro utilizado no volante com o valor de 540, e o botão do acelerador e freio tendo o seu retorno 0, ou seja, não pressionado. Já na (Figura 23), a imagem mostra que o valor do potenciômetro do volante é de 537, e os botões de acelerador e freio estão pressionados, tendo seu valor igual a 1 e seu fundo verde.

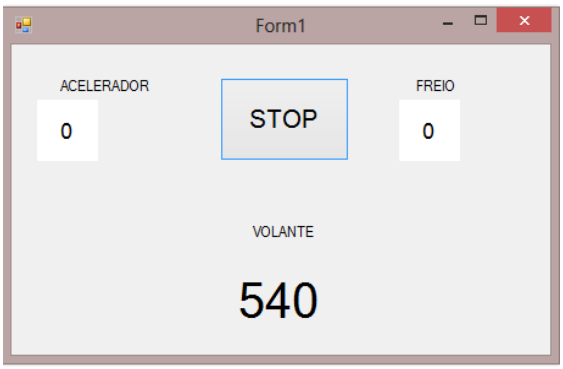

Figura 22 Software de teste de comunicação.
Fonte: Do autor

Figura 23 Software de teste de comunicação.
Fonte: Do autor

Outro software foi Desenvolvido para testar o envio de protocolo através de um PC para o Arduino. As (Figuras 24 e 25) mostram um software que faz um requisito para ligamento de três leds. Os leds possuem cada um uma cor, sendo elas: amarelo, verde e vermelho. Cada botão controla um led com a cor correspondente. Por exemplo, ao clicar no botão amarelo (Figura 24) o led que será aceso ou apagado será o amarelo (Figura 25).

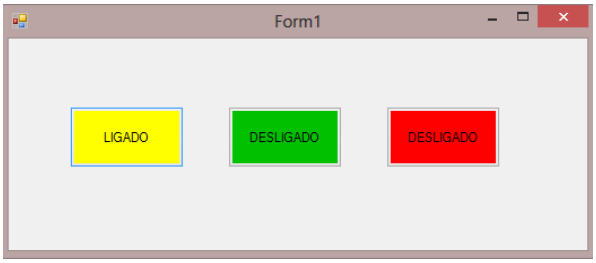

Figura 24 Software para teste de comunicação
Fonte: Do autor

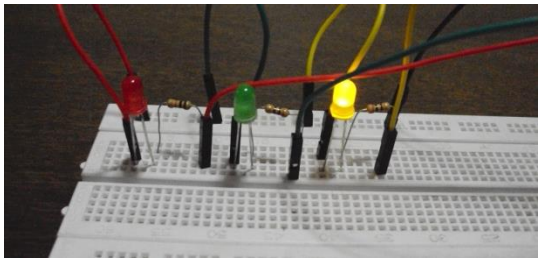

Figura 25 Led aceso conforme o requisito do software de teste.
Fonte: Do autor

4.1 RESULTADOS

Os resultados foram alcançados conforme o esperado para uma primeira versão do projeto. O software está funcionado perfeitamente em um protótipo contendo: dois motores, três display de leds com 7 segmentos e quatro botões. (Figura 25). Pode-se perceber o sincronismo na troca de informações entre o Arduino e o PC, onde foram passados como dados do protocolo de recebimento os dados: 0;8 que solicita o desligamento do motor e a interpretação do numero 8 no display de 7 segmentos, e como protocolo de envio o v843;a0;f0; (Figura 26).

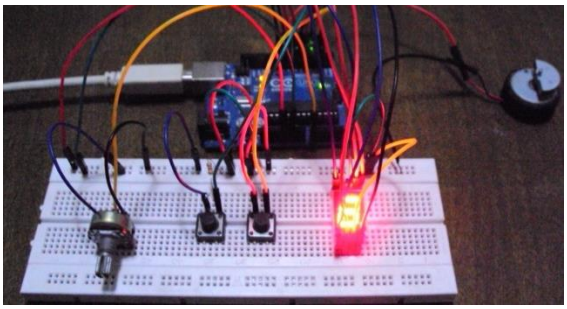

Figura 26 Prototipo recebendo protocolo de recebmento com os dados 0;8.
Fonte: Do autor.

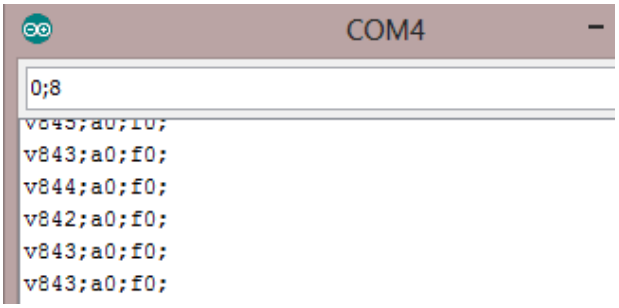

Figura 27 Envio de protocolo para o Arduino
Fonte: Do autor.

5 CONSIDERAÇÕES FINAIS

Pode-se concluir que o protótipo do volante foi desenvolvido com sucesso. Porém ainda há melhorias que podem ser feitas em questão de funcionalidades e comodidade para proporcionar cada vez mais um bom divertimento e conforto para o usuário. Apesar do pouco acesso a materiais de qualidade como moldes para criação da parte física e componentes de alto rendimento e conforto, o protótipo já pode servir de base para que projetos futuros possam vir desempenhar resultados ainda melhores.

A maior dificuldade no decorrer, foi a falta de conhecimento em eletrônica e robótica, que pode ser compensada ao longo do desenvolvimento com vídeos aulas sobre o assunto e artigos publicado por autores da área. Um dos principais desafios foi manter uma boa qualidade em relação ao tempo de resposta na comunicação entre hardware e software, que com ajuda dos softwares desenvolvidos para testes puderam ser melhorados. Foi de grande ajuda fóruns de discussão a respeito do Arduino, onde foi possível adquirir muito conhecimento, acarretando a diversas ideias a respeito da criação do volante.

Como projetos futuros podem ser desenvolvidos diversos tipos de designer de volantes para carros específicos, fazendo com que o usuário sinta um maior realismo perante os diferentes modelos de carros (esportivos, clássicos, casuais e etc). Nesta forma pode-se explorar desde modelos com foco em desempenho, até os destinados a conforto e estabilidade. Testes com diferentes protocolos também podem ser feitos para que o volante funcione em diversas plataformas como diferentes tipos de videogames e plataformas.

6 REFERENCIAS BIBLIOGRAFICAS

MORIMOTO, Carlos. **Entendendo os sistemas embarcados.** Disponível em: http://www.hardware.com.br/artigos/entendendo-sistemas-embarcados/. Acesso em Acesso em 10 Jan. 2013.

DELAI, Andre. **Sistemas Embarcados: a computação invisível.** Disponível em: http://www.hardware.com.br/artigos/sistemas-embarcados-computacao-invisivel/conceito.html. Acesso em Acesso em 10 Jan. 2013.

_____. **Sistemas Embarcados: a computação invisível.** Disponível em: http://www.hardware.com.br/artigos/sistemas-embarcados-computacao-invisivel/pequeno-exemplo-de-se.html. Acesso em Acesso em 10 Jan. 2013.

_____. **Sistemas Embarcados: a computação invisível.** Disponível em: http://www.hardware.com.br/artigos/sistemas-embarcados-computacao-invisivel/chips-perifericos-memoria-num-so-lugar.html. Acesso em Acesso em 11 Jan. 2013.

PRADO, Sergio. **Sistemas de Tempo Real — Parte 1.** Disponível em: http://sergioprado.org/sistemas-de-tempo-real-part-1/. Acesso em Acesso em 11 Jan. 2013.

JORDÃO, Fabio. **O que é memória RAM? [vídeo].** Disponível em: http://www.tecmundo.com.br/memoria/918-o-que-e-memoria-ram-video-.htm. Acesso em Acesso em 11 Jan. 2013.

BEOCK, Leandro. **Conceito de sensores e atuadores.** Disponível em: http://leandro-robotica.blogspot.com.br/2011/05/conceito-e-sensores-e-atuadores.html. Acesso em 10 Jan. 2013.

HARRIS, Tom. **Como funcionam os joysticks.** Disponível em: http://eletronicos.hsw.uol.com.br/joystick.htm. Acesso em 22 Abr. 2013.

_____. **Como funcionam os joysticks.** Disponivel em: http://eletronicos.hsw.uol.com.br/joystick1.htm. Acesso em 22 Abr. 2013.

_____. **Como funcionam os joysticks.** Disponivel em: http://eletronicos.hsw.uol.com.br/joystick2.htm. Acesso em 23 Abr. 2013.

_____. **Como funcionam os joysticks.** Disponivel em: http://eletronicos.hsw.uol.com.br/joystick3.htm. Acesso em 23 Abr. 2013.

_____. **Como funcionam os joysticks.** Disponivel em: http://eletronicos.hsw.uol.com.br/joystick4.htm. Acesso em 24 Abr. 2013.

_____. **Como funcionam os joysticks.** Disponivel em: http://eletronicos.hsw.uol.com.br/joystick5.htm. Acesso em 24 Abr. 2013.

WIKIPÉDIA. **Arduino.** Disponivel em: http://pt.wikipedia.org/wiki/Arduino. Acesso em 08 Mai. 2013.

_____. **Sobre Arduino.** Disponivel em: http://www.labdegaragem.com.br/wiki/index.php?title=Sobre_Arduino. Acesso em 08 Mai. 2013.

ANEXO

Anexo 1: Código lógico do Arduino.

```
#include <Wire.h>
#define expan0 0x38
#define expan1 0x39
#define expan2 0x3A

#define ZERO   B10001000
#define UM     B11101011
#define DOIS   B01001100
#define TRES   B01001001
#define QUATRO B00101011
#define CINCO  B00011001
#define SEIS   B00011000
#define SETE   B11001011
#define OITO   B00001000
#define NOVE   B00001011


void setup ()
{
  Wire.begin();
  Serial.begin(9600);
  pinMode(6,OUTPUT);
}

void loop ()
{
  if(Serial.available()>0)
  {
    String txt = "";

   while(Serial.available())
   {
     char c = Serial.read();
     delay(5);
     txt  += c;
   }

   if(txt.length() >0)
   {
     if(txt[0] == 'm' && txt[1] == '1')
     {
       digitalWrite(6,1);
     }
     if(txt[0] == 'm' && txt[1] == '0')
     {
       digitalWrite(6,0);
     }
```

```
if(txt[2] == 'd')
{
  switch (txt[3])
     {
        case '0': numerar(expan0, ZERO);
        break;
        case '1': numerar(expan0, UM);
        break;
        case '2': numerar(expan0, DOIS);
        break;
        case '3': numerar(expan0, TRES);
        break;
        case '4': numerar(expan0, QUATRO);
        break;
        case '5': numerar(expan0, CINCO);
        break;
        case '6': numerar(expan0, SEIS);
        break;
        case '7': numerar(expan0, SETE);
        break;
        case '8': numerar(expan0, OITO);
        break;
        case '9': numerar(expan0, NOVE);
        break;
     }

    switch (txt[4])
     {
        case '0': numerar(expan1, ZERO);
        break;
        case '1': numerar(expan1, UM);
        break;
        case '2': numerar(expan1, DOIS);
        break;
        case '3': numerar(expan1, TRES);
        break;
        case '4': numerar(expan1, QUATRO);
        break;
        case '5': numerar(expan1, CINCO);
        break;
        case '6': numerar(expan1, SEIS);
        break;
        case '7': numerar(expan1, SETE);
        break;
        case '8': numerar(expan1, OITO);
        break;
        case '9': numerar(expan1, NOVE);
        break;
     }
```

```
        switch (txt[5])
        {
           case '0': numerar(expan2, ZERO);
           break;
           case '1': numerar(expan2, UM);
           break;
           case '2': numerar(expan2, DOIS);
           break;
           case '3': numerar(expan2, TRES);
           break;
           case '4': numerar(expan2, QUATRO);
           break;
           case '5': numerar(expan2, CINCO);
           break;
           case '6': numerar(expan2, SEIS);
           break;
           case '7': numerar(expan2, SETE);
           break;
           case '8': numerar(expan2, OITO);
           break;
           case '9': numerar(expan2, NOVE);
           break;
        }
    }
   }


  }

  impressao();
}


void numerar (byte expansor, byte numero)
{
  Wire.beginTransmission(expansor);
  Wire.write(numero);
  Wire.endTransmission();
}

void impressao()
{
 Serial.print('v');
 Serial.print(analogRead(0));
 Serial.print(';');
 Serial.print('a');
 Serial.print(digitalRead(2));
 Serial.print(';');
 Serial.print('f');
 Serial.print(digitalRead(3));
```

```
 Serial.println(';');
}
```