UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ENGENHARIA ELETRÔNICA

THIAGO DOS SANTOS CAVALI

CONTROLE APRIMORADO PARA CONSOLE DE JOGOS ELETRÔNICOS

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2015

THIAGO DOS SANTOS CAVALI

CONTROLE APRIMORADO PARA CONSOLE DE JOGOS ELETRÔNICOS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do curso Superior de Engenharia Eletrônica do Departamento Acadêmico de Eletrônica - DAELN - da Universidade Tecnológica Federal do Paraná - UTFPR, como requisito parcial para obtenção do título de Engenheiro Eletrônico.

Orientador: Prof. Msc. Paulo Denis Garcez da Luz

CAMPO MOURÃO

2015

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Campo Mourão
Coordenação de Engenharia Eletrônica

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

DO TRABALHO DE CONCLUSÃO DE CURSO INTITULADO

Controle Aprimorado para Console de Jogos Eletrônicos

por

Thiago dos Santos Cavali

Trabalho de Conclusão de Curso apresentado no dia 13 de Fevereiro ao Curso Superior de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Câmpus Campo Mourão. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho APROVADO (aprovado, aprovado com restrições ou reprovado).

Prof. Msc. Ricardo Bernardi
(UTFPR)

Prof. Dr. Roberto Ribeiro Neli
(UTFPR)

Prof. Msc. Paulo Garcez da Luz
(UTFPR)
Orientador

Prof. Msc. Leandro Castilho Brolin
Responsável pelo TCC do Curso de Eng. Eletrônica

Campo Mourão, 13 de Fevereiro de 2015.

Pela infinita importância que possuem em minha vida, dedico esse trabalho a Deus e a meus familiares.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus pela força e coragem durante o cumprimento dessa etapa da minha vida. Agradeço também a todos que me apoiaram de alguma maneira no desenvolvimento desse trabalho. Aos meus pais Rogério e Solange por toda a paciência, amor e apoio incondicional oferecido. Ao Professor Msc. Paulo Denis Garcez da Luz pela sábia maneira de orientar na realização deste projeto. Aos colegas de curso por todas as experiências e momentos compartilhados ao longo destes 5 anos.

Gostaria de agradecer de maneira especial ao amigo Renan Felipe T. B. Ikeda pelo apoio oferecido nos testes do protótipo, ao amigo Fábio Lima pelo auxílio dado na confecção do produto e a todos que, de alguma maneira, contribuíram para o resultado final desse projeto.

“A vida é uma série de experiências e, mesmo que às vezes seja difícil percebê-lo, cada uma delas nos torna um ser humano melhor. O mundo foi construído para desenvolver o caráter do indivíduo. Todos nós precisamos compreender que os obstáculos e sofrimentos que enfrentamos nos ajudam em nossa jornada.”

(Henry Ford)

RESUMO

CAVALI, Thiago dos S.. Controle aprimorado para console de jogos eletrônicos. 2015. 128 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Engenharia Eletrônica, Universidade Tecnológica Federal do Paraná. Campo Mourão, 2015.

Com a evolução dos jogos eletrônicos, principalmente do gênero luta, nos diferentes consoles, observa-se um aumento na dificuldade da jogabilidade, o que pode tornar um jogo menos prazeroso para o jogador. Nesse contexto, é válido o desenvolvimento de ferramentas que venham a melhorar a jogabilidade dos jogos desse gênero. Este projeto tem como objetivo desenvolver um controle aprimorado no formato *arcade*, que seja capaz de com um único botão executar sequências de golpes (combos) que muitas vezes são difíceis de serem realizadas normalmente. É baseado na integração de um controle de *PlayStation® PS3™* com uma placa contendo um microcontrolador AT89S52 e uma memória EEPROM AT24C512, além de circuitos integrados de chave analógica com comando digital CD4066. Os combos previamente gravados (via *software* ou *hardware*) na memória EEPROM são interpretados pelo microcontrolador que aciona as devidas chaves, executando dessa maneira o combo. O resultado deste projeto é um controle atrativo para os jogadores, que melhora a jogabilidade dos jogos de luta.

Palavras-chave: Jogos eletrônicos. Aprimoramento de controle. Microcontrolador 8051.

ABSTRACT

CAVALI, Thiago dos S.. Enhanced joystick for video games. 2015. 128 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Engenharia Eletrônica, Universidade Tecnológica Federal do Paraná. Campo Mourão, 2015.

With the electronic games evolution, mainly the fighting genre, in many different consoles, it's visible an increase on the difficulty of the gameplay, which can provide a less pleasurable game for the player. In this context, the development of tools which could enhance the fighting games gameplay is valid. This project aims to develop an enhanced control with arcade format that can with a single button perform a sequence of blows (combos) which, many times, are hard to make normally. It's based on the integration of a *PlayStation® PS3™* joystick with a board which contains a microcontroller AT89S52 and a EEPROM memory AT24C512, in addition to analogic keys with digital command integrated circuits CD4066. The combos previously recorded (by software or by hardware) on EEPROM memory are interpreted by microcontroller that triggers the proper keys, performing the combo. This project results in an attractive joystick for the players, which improves the fighting games gameplay.

Keywords: Video games. Joystick improvement. 8051 microcontroller.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

A	Ampere
ADC	Conversor Analógico Digital
bps	*bits* por segundo
CI	Circuito Integrado
cm	Centímetros
CPU	*Central Processing Unit*
EIA	*Electronic Industries Alliance*
I/O	Entrada/Saída
I^2C	*Inter-Integrated Circuit*
ICSP	*In-Circuit Serial Programming*
IDE	*Integrated Development Environment*
IHM	Interface Homem-Máquina
ISP	*In-System Programmable*
JVM	*Java Virtual Machine*
K*bit*/s	Kilo *bits* por segundo
K	Kilo
mA	mili Amperes
mAh	mili Ampere-hora
M*bit*/s	Mega *bits* por segundo
MHz	Mega Hertz
ms	mili segundo
NES	*Nintendo Entertainment System*
ns	nano segundos
pF	pico Faraday
PWM	*Pulse Width Modulation*
R/W	*Read/Write*
RPG	*Role Playing Game*
RS	*Recommended Standard*
SCK	*Serial Clock*
SCLK	*Serial Clock*
SNES	*Super Nintendo Entertainment System*

SPI	*Serial Peripheral Interface*
SS	*Slave Select*
TWI	*Two Wire Interface*
USB	*Universal Serial Bus*
V	Volts
WP	*Write Protect*

LISTA DE ACRÔNIMOS

BASIC	*Beginners All-purpose Symbolic Instruction Code*
CPHA	*Clock Phase*
CPOL	*Clock Polarity*
DAC	Conversor Digital-Analógico
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
EPROM	*Erasable Programmable Read Only Memory*
LEDs	*Light Emitting Diodes*
MISO	*Master Input Slave Output*
MOS	*Metal Oxide Semiconductor*
MOSI	*Master Output Slave Input*
PIC	Controlador Integrado de Periféricos
PROM	*Programmable Read Only Memory*
RAD	*Rapid Application Development*
RAM	*Random Access Memory*
ROM	*Read Only Memory*
ULA	Unidade Lógica Aritmética

SUMÁRIO

1 INTRODUÇÃO

Os jogos eletrônicos são um mercado em constante evolução no que diz respeito à tecnologia e à economia. Somente no Brasil o mercado de jogos eletrônicos cresceu em média 26,3% ao ano no período de 2008 até 2014, com um valor total de arrecadação de R$2,1 bilhões em 2013 e tendo previsão de uma arrecadação de R$4,6 bilhões até 2018 (UOL JOGOS, 2014).

Tendo em vista esse potencial mercadológico, pode-se concluir que o desenvolvimento de ferramentas de *hardware* e *software* que possam tornar mais atrativa e interessante a experiência de jogar *vídeo games* tem grande chance de se converter em um produto lucrativo.

1.1 HISTÓRICO DOS CONTROLES PARA *VIDEO GAMES*

Os consoles são aparelhos, desenvolvidos para uso doméstico, que exibem em uma tela um jogo armazenado em um dispositivo (seja ele cartucho, CD, DVD ou *Blu-Ray*). O jogador interage com esse dispositivo, tentando vencer os desafios impostos pelo jogo (BRESCIANI, 2001).

Os primeiros consoles desenvolvidos foram da linha *Odyssey* da *Magnavox* (BATISTA et al. 2007), nesse console o controle de jogo possuía botões giratórios para realizar ações dos elementos da tela de jogo, como mostra a figura 1.

Figura 1 - Console *Odyssey* da *Magnavox* e seu controle com botões giratórios

Fonte: Adaptado de UOL Jogos (2013).

Depois apareceram os primeiros consoles com controle do tipo *joystick*, sendo o mais importante o *Atari 2600*, que dominou o mercado entre o final dos anos 70 e início dos anos 80 (BATISTA et al. 2007). Como mostra a figura 2, os *joysticks* possuíam um botão e um manche para controle dos elementos da tela de jogo.

Figura 2 - Exemplos de *joysticks* e seus respectivos consoles
Fonte: Adaptado de UOL jogos (2013).

Com o conceito de *joystick* em alta, surgiram os primeiros *arcades*, máquinas de jogos de uso público operadas por moedas, que possuem botões e um *joystick* fixos em um gabinete que também continha a tela e o sistema de jogo (BATISTA et al. 2007). Essas máquinas permitiam aos jogadores jogar os *games* em estabelecimentos destinados ao uso de *videogames*, os chamados “Fliperamas”. A popularização desse tipo de máquina veio graças a jogos como *Donkey Kong™* (Mostrado na Figura 3), *Pac-Man™* e *Space Invaders™* (FARAH, 2012).

Figura 3 - Arcade do jogo Donkey Kong™.
Fonte: Bienaimé (2012).

Em 1985, a *Nintendo*® lançou o NES*TM* (*Nintendo Entertainment System*), que dominou o mercado durante muito tempo e trouxe uma evolução na maneira de se jogar videogame, principalmente no que diz respeito a qualidade gráfica e de som (BRESCIANI, 2001). Houve também uma revolução na forma de jogar devido ao joystick que possuía direcionais em forma de cruz (direcional digital) e dois botões para realização das ações do personagem do jogo, como mostra a figura 4.

Figura 4 - Controle do NES*TM* e seu direcional em forma de cruz
Fonte: Adaptado de UOL Jogos (2013).

Na Figura 4 pode-se notar que além da presença dos dois botões para realização das ações do personagem (A e B), há também os botões de *start* (inicia o jogo) e *select* (executa ações específicas para cada jogo).

Em 1988, a *Sega*® para competir com a *Nintendo*® lança o *Mega Drive™* (ou *Genesis™*, como ficou conhecido nos Estados Unidos). Primeiro console de 16 bits da história, foi o mais poderoso console existente na época, com qualidade de som e imagem superiores aos rivais. (BATISTA et al. 2007). Trouxe também uma melhora na forma de controlar, produzindo um controle não retangular que possuía além do botão de *start*, mais três botões para realizar ações do personagem, como mostra a figura 5.

Para entrar no mercado dos consoles de 16 bits, a *Nintendo*® em 1990 lançou o SNES*TM* (*Super Nintendo Entertainment System*) que foi um sucesso de vendas (BRESCIANI, 2001) e lançou o padrão de controle com 4 botões, além de dois

“botões de ombro” localizados um de cada lado do cabo do controle, na parte de cima do mesmo e os botões de *select* e *start*, como mostra a figura 6.

Figura 5 - Controle com 3 botões de ações do *Mega Drive*™
Fonte: Adaptado de UOL Jogos (2013).

Figura 6 - Controle do SNES™
Fonte: *Creative Commons* (2007).

Em 1991 houve um grande advento no mercado dos jogos de luta, principalmente devido ao jogo *Street Fighter 2*™ da *Capcom*® (BATISTA et al. 2007) que levou ao desenvolvimento de controles que possuíam, além de um botão de *start*, 6 botões de ação do personagem ao invés de apenas 4, o que possibilitava uma melhor jogabilidade desse tipo de jogo. Um exemplo disso é o *Sega*® *Saturn*™, cujo controle é mostrado na figura 7.

Figura 7 - Controle do *Sega® Saturn™*.
Fonte: Adaptado de UOL Jogos (2013).

O advento dos games de luta também trouxe de volta a popularidade dos *arcades* que haviam perdido espaço no mercado devido ao desenvolvimento dos *videogames* caseiros. O *Street Fighter™ 2* revolucionou o mercado dos jogos de luta e se tornou popular pelo formato de jogo, no qual haviam 2 jogadores, um contra o outro, com a escolha do personagem baseado nas ações especiais que cada um poderia executar quando pressionada determinada sequência de botões, os chamados “combos”, formato o qual é utilizado até hoje nos jogos de luta. Os *arcades* saíram de cena atualmente no ocidente, mas no Japão sua popularidade é grande até os dias atuais (FARAH, 2012). O *arcade* do *Street Fighter™ 2* é mostrado na figura 8.

Figura 8 - Arcade do jogo Street Fighter™ 2.
Fonte: *Event Hubs* (2010).

Em 1994 a *Sony*® lançou o *Playstation*™ que trouxe uma mudança na forma de jogar videogame, com os controles *DualShock*® que possuem 2 motores elétricos na parte onde o jogador segura o controle (como pode ser visto na Figura 9). Esses motores podem fazer o controle vibrar de acordo com a ação que está sendo executada no jogo.

Figura 9 - Componentes internos do *DualShock*®.
Fonte: Tyson ([20--]).

Como pode se observar na Figura 9, os eixos dos motores apoiam 2 pesos desbalanceados. Quando o motor é acionado eletricamente o peso é girado e, como os pesos são desbalanceados, os motores oscilam. Essa oscilação causa uma vibração no controle durante o jogo, o que proporciona uma experiência diferente para o jogador (TYSON, [20--]).

Os controles *DualShock*® também incorporaram os direcionais analógicos além de manterem os direcionais digitais. Os direcionais analógicos possuem dois potenciômetros (resistores variáveis) pelos quais constantemente passa corrente elétrica. A quantidade de corrente que passa depende da resistência de cada um dos potenciômetros que varia com a posição do *joystick*. Monitorando a saída de

cada potenciômetro, o console é capaz de identificar a posição exata no qual o *joystick* se encontra, executando a ação apropriada (TYSON, [20--]). As estruturas do direcional analógico podem ser vistas na Figura 10.

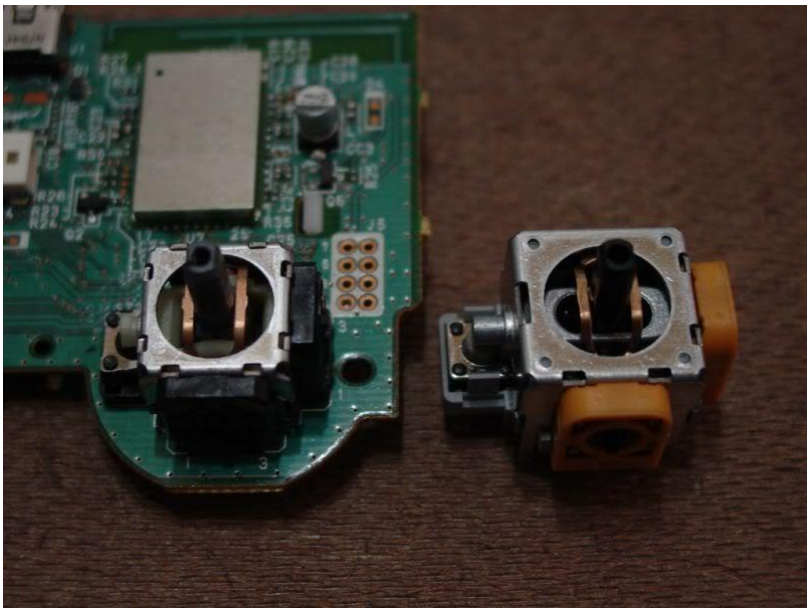


Figura 10 - Estrutura dos direcionais analógicos do *DualShock*®.
Fonte: Adaptado de HT Forum (2010).

Com a utilização do direcional analógico é possível que o jogador tenha um controle maior da movimentação do personagem, o que melhora a jogabilidade de jogos como os de esporte, já que o controle da movimentação acarreta em melhores resultados para o jogador.

Os controles *DualShock*® possuem, além das funcionalidades já citadas, 4 botões de ombro, botões *select* e *start* e quatro botões de ações do personagem. Os *DualShock*® existem até hoje, na versão sem fio e com algumas adaptações no *PlayStation*® *3* e no *PlayStation*® *4,* como mostra a figura 11. O controle *DualShock*® *3* será detalhado na sessão 2.5.7 desse trabalho.

Ao longo da última década, surgiram outros consoles com diferentes formatos de controle, como o *XBOX* da *Microsoft*© e o *Wii*^{TM} da *Nintendo*® que, juntamente com o *Dualshock*® *4*, são o que se tem de mais moderno em termos de controle de consoles na atualidade. Existindo ainda consoles que não possuem controle integrado, sendo que o movimento do personagem é dado pelo próprio movimento do jogador (*Kinect* da *Microsoft*©), ou no qual o controle serve para enviar ações

através do movimento deste por parte de quem está jogando (*PS® Move* da *Sony®* e *Wii™* da *Nintendo®*).

Figura 11 - Controles *Sony® PlayStation® Dualshock®*. Da esquerda para a direita: *DualShock® 1* (*PlayStation® 1*), *DualShock® 2* (*PlayStation® 2*), *DualShock® 3* (*PlayStation® 3*) e *Dualshock® 4* (*PlayStation® 4*)
Fonte: Adaptado de UOL Jogos (2013).

1.2 PROBLEMAS

Pode-se encontrar na *internet* comunidades dedicadas a *games*, nas quais diversos jogadores compartilham experiências sobre dificuldades encontradas em diferentes jogos. Na análise de um tópico na comunidade UOL Jogos (2013), vê-se que uma das maiores dificuldades encontradas no que diz respeito a jogos, principalmente do gênero luta, é a jogabilidade do game dependendo do controle que se usa.

Vale ressaltar que com a utilização de alguns tipos de controle, se torna complexa a realização de algumas sequências necessárias para que o personagem realize ações especiais durante o jogo (os chamados “combos”), o que no caso dos jogos de luta são sequências de golpes, magias, entre outros. Além de o próprio combo ser complexo, uma grande dificuldade é a presença de um tempo certo entre o acionamento de cada botão para que a ação especial do personagem seja realizada com sucesso.

A maior dificuldade a ser solucionada nesse projeto é o desenvolvimento de um controle que facilite a realização dos combos melhorando a jogabilidade de *games* de luta. Esse controle deve ter uma IHM (Interface Homem-Máquina) amigável para facilitar o uso, de forma que diferentes perfis de usuário possam utilizá-lo.

1.3 JUSTIFICATIVA

Em jogos eletrônicos – principalmente do gênero luta – é importante a realização dos chamados combos, sequências de botões que devem ser pressionados para que determinada ação do jogo aconteça. Esses combos muitas vezes são muito complexos de serem realizados, o que torna a jogabilidade de alguns *games* complicada e menos interessante para alguns usuários. Dentro desse contexto é válido o desenvolvimento de um produto que torne mais fácil a realização de combos nos jogos de luta, já que este é um recurso que os jogadores gostariam de possuir e consequentemente pagariam por ele.

1.4 OBJETIVOS

O objetivo desse projeto é o desenvolvimento de um controle voltado para jogos de luta que seja sem fio (*wireless*) e tenha estilo *arcade*, ou seja, que possui em uma estrutura montada um *joystick* e os botões de ação do personagem, como pode ser visto na Figura 12.

Figura 12 - Exemplo de controle estilo Arcade.
Fonte: Kevie (2012)

Esse controle deve permitir ao jogador pré-programar grandes sequências de botões a serem acionados, bem como o tempo entre cada um desses acionamentos, de forma que ao se pressionar um único botão uma sequência inteira de comandos (combos) pré-programados seja executada. O usuário poderá fazer a programação desses combos no próprio controle (gravação via *hardware*) ou pelo computador em um *software* desenvolvido para esse fim (gravação via *software*), proporcionando uma IHM amigável para o usuário do produto.

1.5 METODOLOGIA

Em um primeiro momento será feita uma pesquisa sobre produtos similares ao que se pretende desenvolver e que já são comercializados. O intuito dessa pesquisa é fazer com que o dispositivo a ser construído possua alguns diferenciais em relação ao que existe atualmente no mercado, a fim de que se possa explorar ao máximo o potencial mercadológico que o produto possui.

Depois será feita a seleção dos componentes de *hardware* e *software* que possuem a melhor relação custo-benefício para a utilização no desenvolvimento do projeto. A escolha e obtenção desses componentes levará em conta aspectos como o conhecimento técnico prévio e disponibilidade no mercado.

A seguir será realizada a aquisição dos componentes. Com esses componentes em mãos será construído um protótipo que terá suas placas roteadas utilizando um *software* de design de placas de circuito impresso, além da montagem das placas. Essas placas serão ligadas através de fios a um controle *DualShock® 3* do console *PlayStation® 3*. Esse protótipo será utilizado para a realização de testes de jogabilidade e funcionalidade.

Posteriormente será desenvolvido o *software* que permitirá a programação dos combos no computador, além do controle estilo *arcade* que também será construído.

Por fim, serão feitos testes com diferentes perfis de jogadores que darão um *feedback* sobre a funcionalidade do produto desenvolvido.

1.5.1 Estrutura do Trabalho

Esse trabalho será dividido em quatro capítulos. No primeiro capítulo será feita uma fundamentação teórica na qual será discorrido sobre a importância dos jogos eletrônicos para a sociedade. Nesse capítulo também será feito um estudo de produtos similares ao proposto nesse trabalho e que já existem no mercado. Por fim será dada uma visão geral do projeto além da descrição de algumas tecnologias que podem ser utilizadas para a execução deste.

No segundo capítulo será mostrado o desenvolvimento do trabalho, abordando aspectos desde a seleção dos componentes de *hardware* e *software* que serão utilizados até o processo de montagem do protótipo e do produto final, além de testes de funcionalidade.

No terceiro capítulo será feita uma discussão dos resultados obtidos com o projeto e uma análise de funcionalidade do produto baseada na opinião de diferentes perfis de jogadores de jogos de luta.

No quarto capítulo serão feitas as considerações finais, abordando aspectos como as conclusões obtidas com o projeto, dificuldades encontradas ao longo da realização e implementações a serem realizadas no futuro para melhorar ainda mais a funcionalidade do produto.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 DEFINIÇÃO E IMPORTÂNCIA DOS JOGOS ELETRÔNICOS

Ao longo do tempo os autores vêm dando diversas respostas sobre o que é um jogo eletrônico (GONÇALVES; FERREIRA, 2012). Segundo Huizinga (2003), o jogo é como um “círculo mágico” separado no tempo e no espaço do mundo o qual estamos acostumados. Esse novo mundo possui suas próprias regras e representa a liberdade, sendo separado do mundo real e visando a satisfação do jogador.

Crawford (2003), por sua vez, define um jogo como uma expressão criativa feita visando lucro, sendo uma atividade interativa de entretenimento com objetivos e desafios contra outros jogadores (reais ou virtuais), existindo a possibilidade de luta entre eles.

Os jogos possuem 3 elementos básicos: Regras, sistema e diversão, que segundo Hunicke et al. (2004), podem ser vistos como mecânica, dinâmica e estética. A mecânica é o modo como o jogador controla o jogo, a dinâmica é a definição do comportamento do jogo após os comandos de entrada oriundos do jogador e a estética é responsável pelo o que o jogador sente durante a interação com o jogo.

Os jogos possuem um potencial de, simultaneamente, fazer as pessoas seguirem regras e proporcionar diversão através da imersão em um mundo diferente do que vivemos. Somando-se a isso existe o fato dos jogos estimularem a imaginação dos jogadores. Através dos jogos podemos “fugir” de nossa realidade e de nosso cotidiano. Petry (2011) sugere que esta é uma das experiências mais humanas que podemos ter.

2.2 CLASSIFICAÇÃO DOS JOGOS

Segundo Tarouco et al. (2004), os jogos podem ser divididos em cinco categorias:

- Jogos de Aventura: Nesse tipo de jogo o principal componente é a exploração de ambientes. Um exemplo é a série clássica de sucesso *Super Mario World™* da *Nintendo*®.
- Jogos Lógicos: Jogos que exigem mais raciocínio que reflexos, sendo importantes para o desenvolvimento da concentração, do raciocínio lógico e estratégico, velocidade de raciocínio, percepção visual e espacial, tomada de decisões e planejamento.
- Jogos Estratégicos: Tem como enfoque o conhecimento e as habilidades para resolução de problemas, principalmente no que diz respeito à construção e administração. Como, por exemplo, administração de uma cidade (caso do jogo *SimCity™* da *Maxis*®) ou a construção de um império (como em *Age of Empires™* da *Ensemble Studios*®).
- *Role Playing Game* (RPG): Nos RPGs, o jogador assume um personagem em determinado ambiente, permitindo uma interação direta entre os personagens que constroem uma história de maneira dinâmica oferecendo um ambiente cativante e motivador.

- Jogos de ação: São os que desenvolvem os reflexos e coordenação do jogador através de uma tomada de decisão que deve ser rápida frente a uma situação inesperada. Os jogos de luta se enquadram nessa categoria.

2.3 ESTADO DA ARTE

Existem alguns produtos no mercado que tem como intuito facilitar a jogabilidade do *game* através de um controle estilo *arcade*. Um exemplo desse tipo de produto é o *Dominator* da empresa *XCM* que, além do controle estilo *arcade* que facilita a jogabilidade, possui uma memória interna que permite a programação no próprio dispositivo de uma sequência a ser realizada. Após a programação, com um único botão, toda sequência que está na memória é executada. Esse controle possui a desvantagem de não ser programável via *software*, somente por acionamentos no próprio controle, o que gera uma restrição em termos da realização do combo já que se faz necessário que o usuário consiga realizar o combo desejado ao menos uma vez para que o mesmo possa ser gravado, o que pode não ser interessante para alguns usuários.

Existem outros produtos que são comercializados que prometem melhorar a jogabilidade. Há controles personalizados que por si só não são programáveis e somente servem para aprimorar a maneira de jogar através de um controle do tipo *arcade*, como o *Mad Catz Fightstick PRO Regular Edition*© e o *Mad Catz Fightstick PRO Tournament Edition*©. Esses controles se combinados com o *software Programmable Joystick Adjustment*© da *InsaneCombo.com*®, podem ser programados. A restrição desse *software* é a IHM pouco amigável, que torna a ligação no computador e a programação algo complexo para alguns usuários.

Estes possuem uma desvantagem em comum: o fato de só poderem ser ligados no console via cabo USB (*Universal Serial Bus*) não sendo possível a conexão sem fio com o console, o que pode trazer uma restrição em termos de conforto para o jogador.

É válido ressaltar que mesmo com as restrições que os produtos possuem, há um grande potencial de mercado, haja visto o elevado número de *sites* da *internet* que os comercializam.

2.4 VISÃO GERAL DO PROJETO

O diagrama da Figura 13 apresenta a visão geral do projeto:

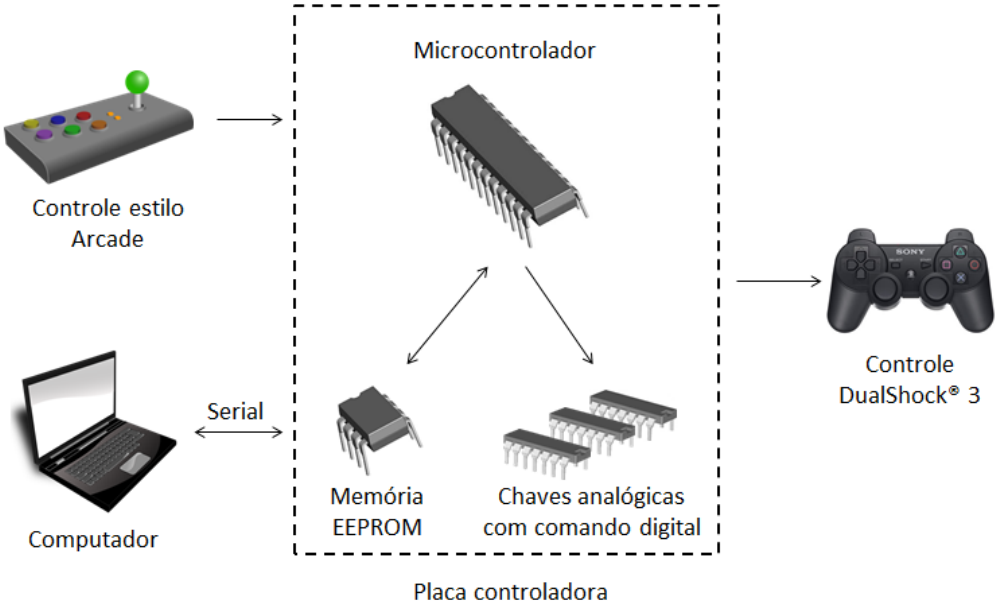


Figura 13 – Visão geral do projeto

Fonte: Autoria própria.

Na Figura 13, pode-se observar a interação entre as partes que compõe o projeto. O microcontrolador é o responsável pelo controle de todas as operações do circuito, sendo assim cabe a ele:

- Interpretar os comandos vindos do acionamento dos botões do arcade;
- Enviar os comandos digitais para acionamento das chaves;
- Escrita na memória EEPROM (*Electrically Erasable Programmable Read-Only Memory*) dos combos vindos do computador via comunicação serial (gravação via *software*) ou dos combos feitos no arcade (gravação via *hardware*);
- Leitura e interpretação dos combos gravados na memória EEPROM;
- Envio dos combos gravados na memória EEPROM para o software do computador via serial;

As outras partes têm as seguintes funções:

- Memória EEPROM: Armazenar, mesmo após o desligamento da placa, as sequências de bytes que compõe um combo;
- Chaves analógicas com comando digital: Permitem a comunicação com o controle *DualShock*® 3, realizando o acionamento dos botões do controle quando recebem um comando do microcontrolador;
- Controle *DualShock*® 3: Realiza as ações no jogo.
- Controle estilo Arcade: Interface com o jogador. É o controle construído para que o usuário possa jogar e usufruir das funcionalidades do projeto;
- Computador: Possui o *software* desenvolvido para gravação de combos. Comunica-se via serial com a placa controladora.

2.5 DESCRIÇÃO DE FERRAMENTAS

Nessa seção serão abordadas brevemente algumas tecnologias de *hardware* e *software* que podem ser utilizadas para a implementação do projeto, descrevendo aspectos como recursos, histórico, definições e funcionamento.

2.5.1 Microcontroladores

2.5.1.1 Definição de Microprocessadores

De acordo com Campos (2008) um microprocessador é um circuito integrado que contém um grande número de transistores que trabalham juntos de forma a armazenar e manipular dados que tornam possível sua utilização para uma grande variedade de funções úteis. A principal função de um microprocessador é executar uma tarefa específica gravada em sua memória de forma a atuar de diferentes maneiras em dispositivos periféricos, realizando ações como reconhecimento de alterações, acionamentos, comunicação, entre outros (NICOLOSI, 2007).

A figura 14 mostra a arquitetura básica de um microprocessador (MP), na qual se pode observar a presença da CPU (*Central Processing Unit*), e de seus dispositivos associados, tais como RAM, ROM, alimentação e periféricos. Pode-se observar também os barramentos (bus) de endereços, de dados e de entrada e saída (I/O), além de vias de controle auxiliar.

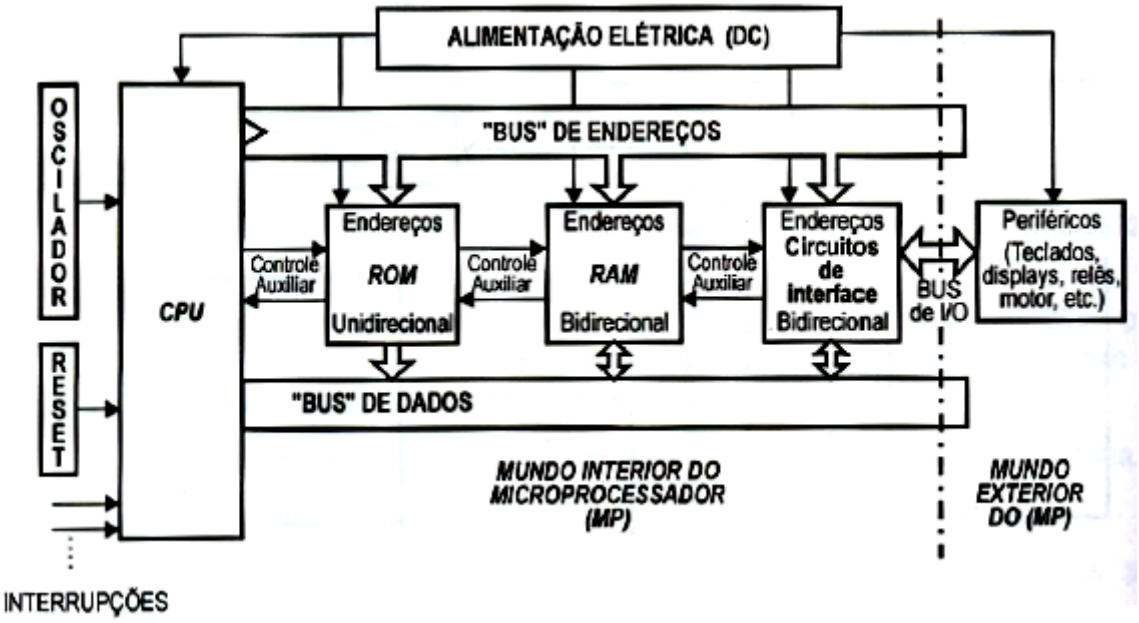


Figura 14 - Arquitetura básica de um microprocessador (MP).
Fonte: NICOLOSI (2007).

A função de cada elemento na arquitetura básica de um MP, segundo Abreu (2005) e Nicolosi (2007), é dada a seguir:

- ROM (*Read Only Memory)*: Memória de programa do microprocessador. Não-volátil (mantém os dados após o desligamento do sistema), tem por função gravar as instruções que o microprocessador deve executar.
- RAM (*Random Access Memory)*: Memória de dados do microprocessador. Volátil (perde os dados com o desligamento do sistema), tem como função armazenar temporariamente as informações que o programa utiliza.
- Periféricos: Qualquer dispositivo que seja controlado pelo microprocessador. Os periféricos podem mandar informações para o microprocessador (periféricos de entrada) ou receber informações (periféricos de saída). Exemplos de periféricos são teclados, *displays*, relês, motores, entre outros.
- Barramento de dados: Barramento no qual os dados trafegam entre a CPU e os outros elementos do microprocessador.
- Barramento de endereços: Serve para a seleção por parte do MP, de qual endereço de memória ou periférico será acessado.

- Barramento de I/O: Através desse barramento é feita a comunicação do “mundo interior” do MP com o “mundo exterior”. Ou seja, é através desse barramento e dos circuitos de interface que o microprocessador se comunica com os periféricos.
- Vias de controle auxiliar: Permitem que o MP acione apenas o dispositivo que deseja utilizar. Como os barramentos de dados e de endereços atuam de forma paralela, é necessário que haja esse acionamento exclusivo para que não ocorra acesso indevido de nenhum dispositivo.
- CPU: A unidade Central de Processamento é o principal elemento de um microprocessador sendo responsável por se comunicar com todos os outros elementos, sempre seguindo as diretivas salvas na ROM. Dentro dela encontram-se registradores, uma unidade de controle e uma ULA (Unidade Lógica Aritmética), elementos responsáveis por realizar as operações de um programa.

Além desses elementos, pode-se observar na figura 14 a presença de outros elementos, que são os seguintes:

- Oscilador: Determina a velocidade com que o MP executará suas tarefas e gera um sincronismo para a execução destas.
- *Reset*: Faz com que o MP inicie suas rotinas internas de execução.
- Interrupções: Pinos de acesso externo que fazem com que o MP pare a execução de suas tarefas normais para realizar uma tarefa específica que deve ser executada quando a interrupção ocorre.

2.5.1.2 Microprocessadores e Microcontroladores

Existem algumas diferenças entre microprocessadores e microcontroladores, principalmente no que diz respeito à utilização e a composição. Enquanto um microprocessador necessita de recursos típicos externos a ele para que possa funcionar corretamente, um microcontrolador possui todos esses recursos integrados em um único *chip*, o que traz ao mesmo tempo uma economia de espaço físico e uma perda na capacidade de realização de cálculos. Vale ressaltar que “a

ULA de um processador convencional de fato é muito mais poderosa se comparada com a ULA de um microcontrolador” (CAMPOS, 2008).

De acordo com Nicolosi (2007), um microcontrolador corresponde a um microprocessador e seus periféricos típicos, todos juntos em um só chip. A Figura 15 mostra a diferença entre um microcontrolador e um microprocessador.

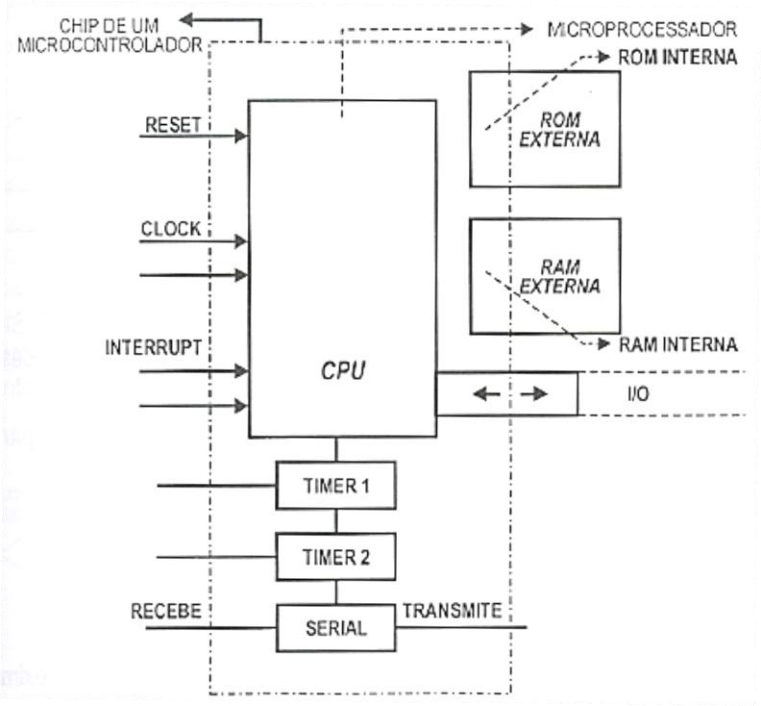


Figura 15 - Diferenças entre um Microprocessador e um Microcontrolador.
Fonte: Nicolosi (2007)

Na figura 15 pode-se observar que enquanto o microprocessador necessita de recursos externos para o seu funcionamento (*Timers*, serial, RAM, ROM, etc.) o chip de um microcontrolador possui todos esses recursos integrados (NICOLOSI, 2007).

De acordo com Campos (2008), um microcontrolador “é um microprocessador com propósito especial (...), como o nome indica, são microprocessadores destinados a controlar”, enquanto os microprocessadores são utilizados em aplicações que necessitam de cálculos matemáticos complexos e com muita velocidade.

2.5.1.3 8051

Os microcontroladores 8051 foram desenvolvida na década de 80 pela Intel® como uma evolução da família 8048 (NICOLOSI, 2007). É um microcontrolador de 8 *bits*, relativamente simples, mas com ampla aplicação baseado na família de microcontroladores MCS®-51. Suas principais características, de acordo com Nicolosi (2007) são:

- RAM interna de uso geral de 128 *bytes* e mais 128 *bytes* que correspondem aos registradores especiais;
- ROM interna de 4K *bytes*;
- 4 *Ports* de entrada e saída (I/O) com 8 pinos cada, totalizando 32 pinos de I/O;
- 2 *Timers* de 16 bits;
- Interface serial;
- Ciclos de instrução de 1 e 2μs operando com um *clock* de 12 MHz.

Existem evoluções dessa família, como o 8052 que possui um *timer* a mais, além de alguns modelos que possuem recursos como conversor analógico-digital (ADC) e saída PWM (*Pulse Width Modulation*) tudo integrado em um só chip.

2.5.2 Evolução das memórias ROM

2.5.2.1 ROM

As memórias ROM são memórias de somente leitura, ou seja, ela só pode ser lida e não pode ser escrita (ao contrário de memórias como a RAM, por exemplo). É feita para realizar uma função determinada e não pode ser modificada ou programada de nenhuma forma. É um tipo de memória não volátil, sendo assim, não perde os dados quando o circuito no qual ela se encontra é desligado. Vale ressaltar que os dados não se perdem nem mesmo quando o chip de ROM é retirado do circuito (PAULA, 2014).

Tendo em vista o fato de que as memórias ROM não podem ser programadas após sua fabricação, sua gravação é feita de acordo com as especificações do usuário no ato da fabricação do *chip* de memória. Os dados são gravados através da correta configuração dos transistores internos presentes no CI que indicam nível lógico alto (“1”) ou nível lógico baixo (“0”) de acordo com a informação que deve ser gravada (DUTTA, 2010).

Sua aplicação originalmente foi para armazenar jogos em consoles de *vídeo games* que possuíam os jogos gravados nos próprios consoles, além de dados fixos necessários para o funcionamento de aparelhos eletrônicos e outras aplicações como dados de áudio em instrumentos musicais eletrônicos (DUTTA, 2010).

2.5.2.2 PROM

As memórias PROM (*Programmable Read Only Memory*) são memórias ROM nas quais a programação pode ser feita pelo usuário com a utilização de um equipamento especial de laboratório, mas uma vez gravada a memória não pode ser modificada outra vez (PAULA, 2010). Essa impossibilidade de regravação pelo usuário se deve ao fato de que para a memória ser programada ocorre a queima de fusíveis policristalinos que estão presentes nas ligações internas entre os transistores do CI de memória PROM (FRUETT, 2007).

2.5.2.3 EPROM

Como evolução das memórias PROM, na década de 70 surgiram as memórias EPROM (*Erasable Programmable Read Only Memory*). Esse tipo de memória pode ser programada eletricamente e apagada com exposição da memória a luz UV (Ultra Violeta). Sendo assim, vale dizer que com a utilização da EPROM pode-se reprogramar os dados presentes na memória (PAULA, 2010).

O processo de gravação de uma EPROM se dá pelas características internas do Circuito Integrado (CI) de memória, no qual há células de armazenamento compostas por transistores MOS (*Metal Oxide Semiconductor*). Nas memórias EPROM esses transistores possuem uma estrutura de porta flutuante (mostrada na Figura 16) e, em estado normal, encontram-se desligados, o que indica nível lógico alto (“1”). Caso seja aplicada uma alta tensão na região da porta, ocorre um

“aprisionamento” de elétrons na porta flutuante, o que faz com que o transistor seja ligado indicando assim nível lógico baixo (“0”) (VIEIRA, 2014).

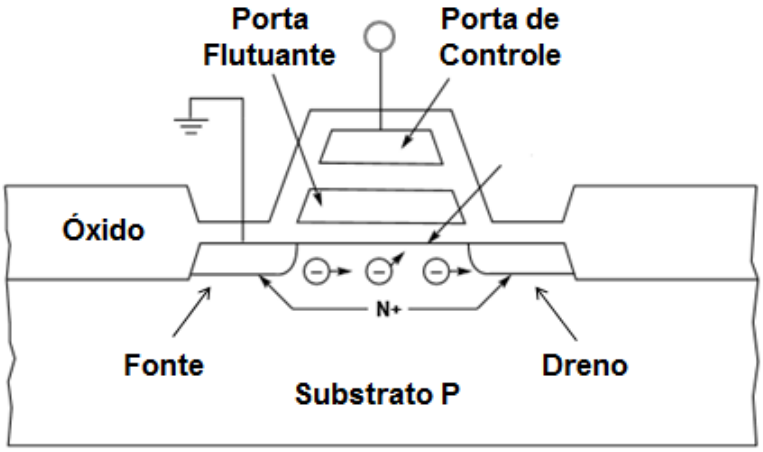


Figura 16 - Transistor MOS com estrutura de porta flutuante.
Fonte: Adaptado de Dutta (2010).

Um *chip* de memória EPROM é mostrado na Figura 17. Nessa imagem, pode-se perceber a presença de um espaço na região central do *chip* no qual está localizado um cristal de quartzo. O processo de apagamento da memória EPROM ocorre quando uma luz ultravioleta incide sobre esse cristal de quartzo durante aproximadamente 20 minutos. Essa incidência de luz produz uma corrente da porta flutuante para o substrato do transistor removendo as cargas aprisionadas e o desligando (VIEIRA, 2014). Após a escrita de dados na memória o espaço deve ser coberto com um material opaco para evitar apagamento acidental (DUTTA, 2010).

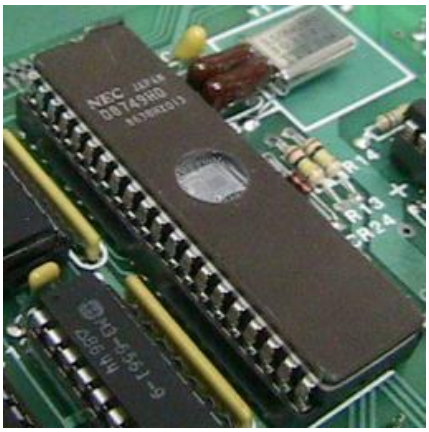


Figura 17 - Chip de memória EPROM.
Fonte: Paula (2010).

2.5.2.4 EEPROM

As memórias EEPROM são uma evolução das memórias EPROM e possuem a vantagem de poderem ser apagadas eletronicamente. Tem os transistores internos com a mesma estrutura de porta flutuante das memórias EPROM, porém com a presença de uma fina camada de metal sobre o dreno do transistor. Essa camada de metal faz com que a aplicação de uma tensão (de cerca de 21 Volts) entre a porta e o dreno do transistor produza uma carga induzida na porta flutuante e uma tensão reversa de mesmo valor remova essas cargas, de forma a gravar ou apagar a célula de memória (VIEIRA, 2014).

Como é necessária apenas a aplicação de uma tensão para programar ou apagar a memória, esse processo pode ser realizado sem que haja a remoção da memória do circuito. Outra característica importante das memórias EEPROM, de acordo com Dutta (2010), é a possibilidade de se apagar eletricamente bytes individuais, o que torna muito mais fácil e rápida a modificação de dados gravados na EEPROM.

Os *chips* modernos de memória EEPROM possuem internamente os circuitos necessários para geração das tensões necessárias para programação, controle de temporização e sequência das operações de programação, o que torna ainda mais simples sua utilização (VIEIRA, 2014). *Chips* de memória EEPROM são mostrados na Figura 18.

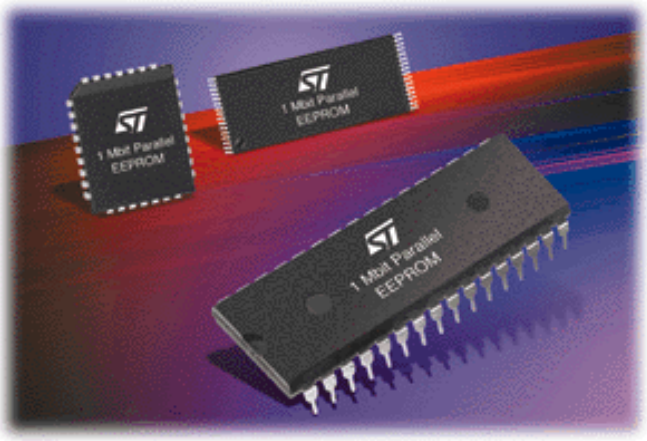


Figura 18 - *Chips* de memória EEPROM.
Fonte: Paula (2010).

De acordo com Dutta (2010), as memórias EEPROM podem ser divididas em dois grupos:

- Paralelas: Representam 10% do mercado. São rápidas, oferecem alta resistência e confiabilidade, sendo mais utilizadas em aplicações militares.
- Seriais: Representam os outros 90% do mercado. São mais lentas se comparadas às paralelas, porém possuem um preço muito menor, o que faz com que seja utilizada na maior parte das aplicações. Outra vantagem é a quantidade de pinos no CI, tendo em vista que as EEPROM seriais são geralmente comercializadas em CIs de 8 pinos enquanto as paralelas possuem um número maior (como pode ser visto na Figura 18). Essa redução no número de pinos leva a uma melhoria de aproveitamento de espaço no circuito.

As memórias EEPROM seriais se comunicam com os microcontroladores através de protocolos de comunicação serial síncrona. Alguns protocolos de comunicação desse tipo serão descritos na sessão a seguir.

2.5.3 Protocolos de comunicação serial síncrona entre periféricos

Nessa sessão serão descritos alguns protocolos utilizados na comunicação serial entre microcontroladores e dispositivos periféricos como, por exemplo, memórias EEPROM, ADC e DAC (Conversor Digital-Analógico).

2.5.3.1 SPI

De acordo com Maciel et al. (2013), o barramento SPI (*Serial Peripheral Interface*) foi desenvolvido originalmente pela Motorola para sua série de microcontroladores 68000. Com o passar dos anos esse barramento foi se popularizando e hoje está presente em produtos de diversas empresas.

É um barramento composto por 3 linhas de transmissão de informação de 8 *bits* que interligam dispositivos *masters* (mestres) e *slaves* (escravos). O dispositivo mestre é o que gerencia a transferência de informação gerando sinais de *clock* (para

sincronismo das operações) e de controle, enquanto os dispositivos escravos são controlados pelo mestre.

Os sinais básicos de comunicação entre os dispositivos SPI são (MACIEL et al., 2013):

- MOSI (*Master Output Slave Input*);
- MISO (*Master Input Slave Output*);
- SCK ou SCLK (*Serial Clock*);
- SS (*Slave Select*);

A forma como os sinais interligam os dispositivos mestre e escravos é mostrada na Figura 19.

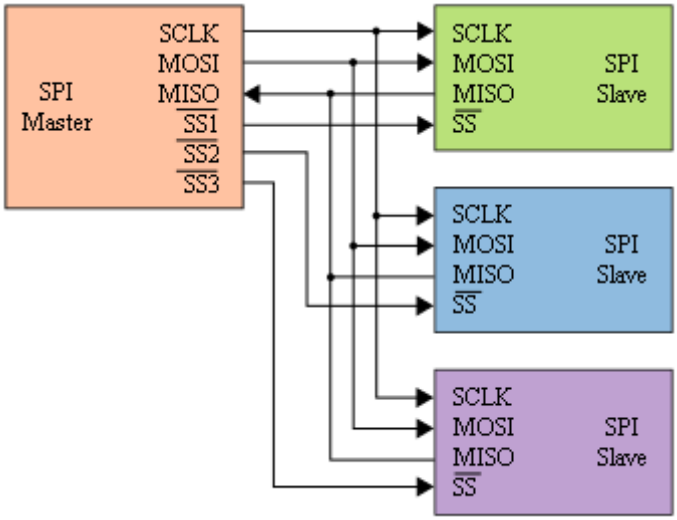


Figura 19 - Dispositivos e sinais do barramento SPI.
Fonte: Adaptado de Maciel et al. (2013)

A comunicação se dá pelo princípio do deslocamento de *bits*, o chamado "*shifter*", no qual há no transmissor a conversão de um registrador paralelo para sinais seriais de acordo com o *clock.* Da mesma maneira há no receptor a conversão do sinal recebido de maneira serial em um registrador paralelo. Essa estrutura é mostrada na figura 20.

É possível observar na Figura 20 como ocorre a comunicação. A cada pulso do *clock* gerado pelo mestre há a transmissão simultânea de uma informação em MOSI do mestre para o escravo e de uma informação em MISO do escravo para o mestre. Essas informações passam pelo *shifter* de cada dispositivo que faz a conversão de serial para um registrador paralelo ou vice-versa (ROCHA, 2010).

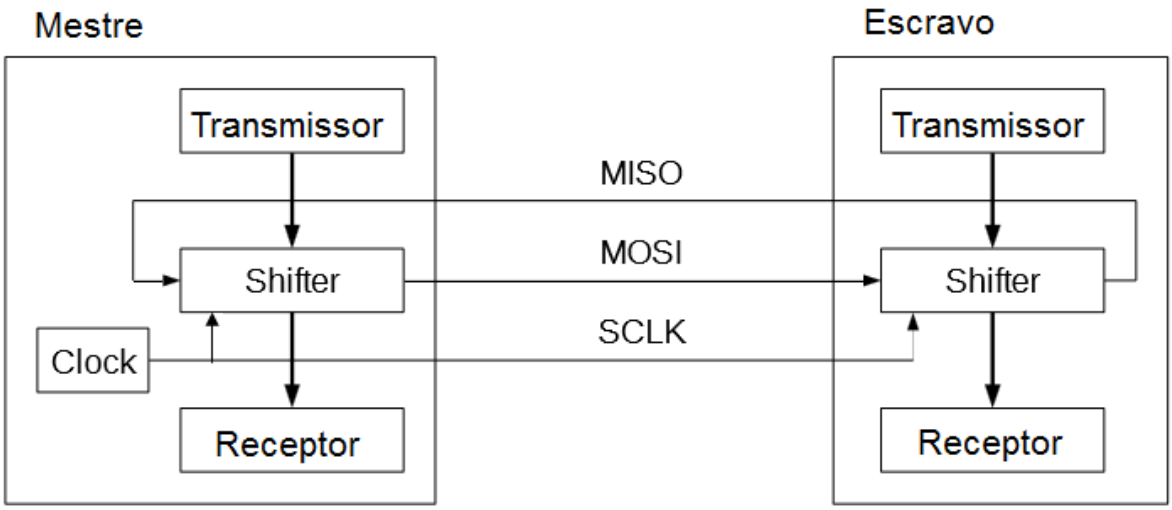


Figura 20 - Princípio da comunicação SPI entre dispositivo mestre e um dispositivo escravo.
Fonte: Rocha (2010).

No cenário no qual há a presença de mais de um escravo, durante a comunicação o mestre atribui nível lógico baixo ao fio SS onde está ligado o dispositivo escravo com o qual deseja trocar dados. Em ambos os cenários, a transmissão se encerra quando o mestre suspende a geração de *clock*.

Durante a comunicação a cada pulso de *clock* há, por parte do mestre, a troca dos *bits* de dados em MOSI e a amostragem dos *bits* de dados em MISO. A borda de *clock* na qual ocorre a troca e a amostragem é definida pelo modo do SPI, sendo que existem quatro modos que são configurados através do ajuste da polaridade e da fase do *clock*, sendo a polaridade configurada por CPOL (*Clock Polarity*) e a fase por CPHA (*Clock Phase*) (ROCHA, 2010). Os modos são mostrados no Quadro 1.

Modo	**CPOL**	**CPHA**	**Borda de troca**	**Transição**
0	0	0	Subida	Meio do *Bit*
1	0	1	Descida	Começo do *Bit*
2	1	0	Descida	Meio do *Bit*
3	1	1	Subida	Começo do *Bit*

Quadro 1 – Modos de configuração no protocolo SPI.
Fonte: Adaptado de Sacco (2014).

CPHA define se os dados serão capturados na primeira ou na segunda borda de *clock*, enquanto CPOL define se quando em repouso o sinal estará em nível alto ou baixo, o que indica o sentido das transições de *clock* durante a comunicação. O princípio básico de todos os quatro modos é o de que os dados são obtidos pelo receptor na borda de *clock* oposta à que são colocados pelo transmissor (ROCHA, 2010). As formas de onda que mostram o comportamento da comunicação SPI de acordo com a configuração do modo podem ser observadas na Figura 21.

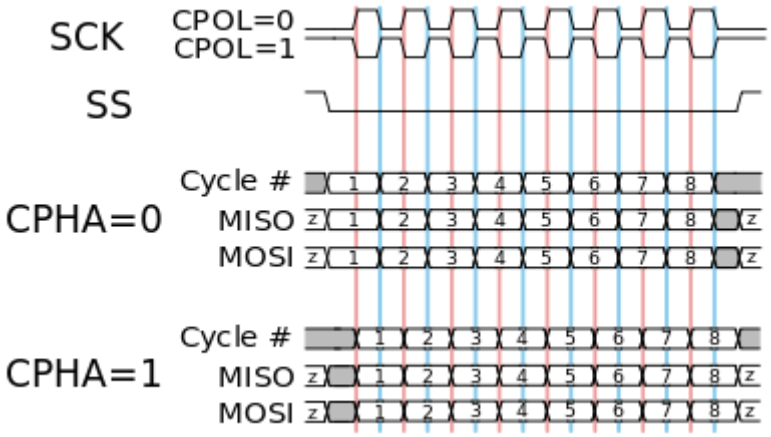


Figura 21 - Comportamento do SPI de acordo com o modo.
Fonte: Burnett (2010).

Vale ressaltar que para que a comunicação seja realizada de maneira correta o mestre e o escravo devem possuir a mesma configuração de modo, caso contrário não haverá sincronismo entre transmissão e recepção dos bits de dados.

O protocolo SPI não tem definida uma taxa máxima de transmissão, não possui modo próprio de endereçamento e não oferece controle de fluxo. Porém é válido dizer que é um protocolo rápido por se tratar de uma comunicação *full-duplex*, na qual se pode enviar e transmitir dados simultaneamente (BYTE PARADIGM, [20--]).

2.5.3.2 I^2C

Segundo (CAMARA, 2013) o protocolo I^2C (*Inter-Integrated Circuit*) ou TWI (*Two Wire Interface*) foi desenvolvido pela Philips em meados dos anos 90. Utiliza apenas dois fios (SDA e SCL) para realizar uma comunicação serial, fazendo com

que esse protocolo seja amplamente utilizado para conexão de periféricos de baixa velocidade a microcontroladores.

Nesse protocolo, tanto a unidade de controle (microcontrolador, por exemplo) quanto os periféricos devem possuir suporte a esse tipo de comunicação. Há no mercado CIs que realizam a interface entre unidade de controle e periféricos com este protocolo, porém é comumente usada uma técnica chamada *bit-bang* com a qual o funcionamento do protocolo é implementado *bit*-a-*bit* no *software* de controle.

O barramento desse protocolo possui 2 fios, chamados de SDA e SCL nos quais trafegam dados e se dá o *clock* do barramento, respectivamente. Possui também a alimentação, geralmente de 3.3V (Volts) até 5V. Tem o número de dispositivos limitados pela palavra de endereçamento, que pode ser de 7, 10 ou 12 *bits* e pela restrição de espaço, haja visto que com distâncias superiores a poucos metros de fios há uma capacitância de torno de 400 pF (pico Faraday) que impede o correto funcionamento da comunicação.

O protocolo possui dois tipos de dispositivos: *master* (mestre) e *slave* (escravo). Sendo que o *master* é a unidade de controle responsável por gerenciar a comunicação entre ele mesmo e os *slaves* (CAMARA, 2013). Um exemplo de barramento I^2C pode ser visto na figura 22.

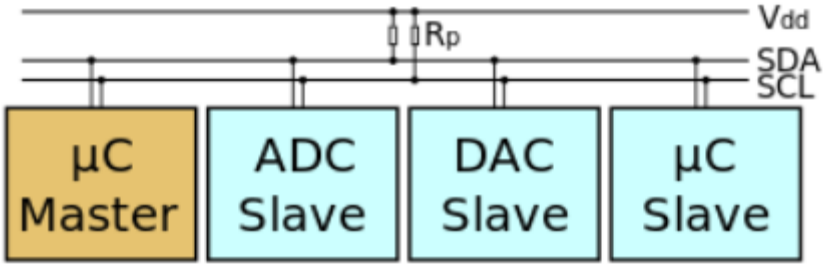


Figura 22 – Exemplo de barramento I2C.
Fonte: Camara (2013).

Na Figura 22 pode-se observar o *Master* (dispositivo mestre, microcontrolador ou µC) e os *Slaves* (dispositivos escravos, ADC, DAC e outro µC), além dos fios de alimentação (Vdd), SDA e SCL e resistores de *pull-up* (Rp) entre Vdd e os outros fios. Pode-se perceber também que quando se encontra no estado neutro o valor mantido em SDA e SCL e nível lógico alto. Para que se inicie a comunicação o

dispositivo mestre deve atribuir nível lógico baixo a SDA, o chamado *start bit*. Vale ressaltar que o estado de SDA é lido a cada pulso de SCL. O procedimento de escrita em um periférico é mostrado na Figura 23.

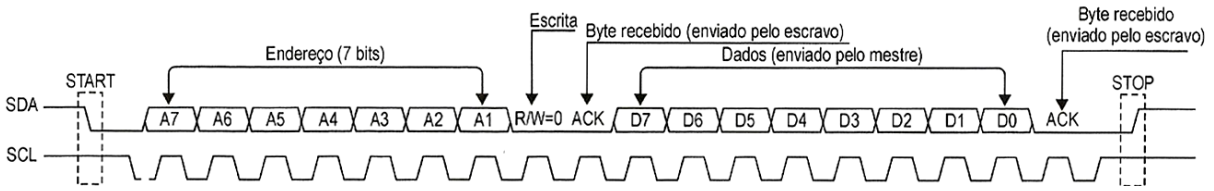


Figura 23 - Processo de escrita utilizando protocolo I²C.
Fonte: Adaptado de Maciel et al. (2013)

Na figura 23 pode-se perceber que após o *start bit,* é colocado em SDA um *byte* composto pelo endereço do dispositivo escravo que o mestre deseja acessar mais o bit que indica leitura ou escrita (R/W – *Read/Write*). No caso da Figura 23, há uma operação de escrita (R/W=0) no dispositivo escravo localizado no endereço dado nos bits de endereço. Caso esse dispositivo exista ele envia um bit de confirmação para o *master* (ACK - *Acknowledge*) (CAMARA, 2013).

Após o recebimento da confirmação enviada pelo escravo, o mestre envia *bit* a *bit*, começando pelo *bit* mais significativo, o *byte* de dados a ser escrito naquele dispositivo. Após o recebimento do *byte*, o escravo retorna um novo ACK confirmando recebimento da informação, ao passo que o mestre envia um *stop bit* que indica o fim da comunicação com aquele dispositivo.

A operação de leitura é similar a de escrita, com a diferença de que o *bit* R/W é posto em nível lógico alto e o mestre recebe do escravo um *byte* com os dados que deseja obter deste. Quando receber todos os dados que necessita, o mestre envia um bit NACK que indica fim da transmissão (interrupção da comunicação com o escravo) e em seguida envia um *stop bit* (FILHO, 2014). Esse processo é mostrado na Figura 24.

Vale ressaltar que tanto na operação mostrada na Figura 23 quanto na mostrada na figura 24, o estado do barramento SDA é lido somente no instante em que há um pulso no barramento SCL. Esse fato oferece a possibilidade de se interromper a comunicação a qualquer instante sem que os dados sejam corrompidos (CAMARA, 2013).

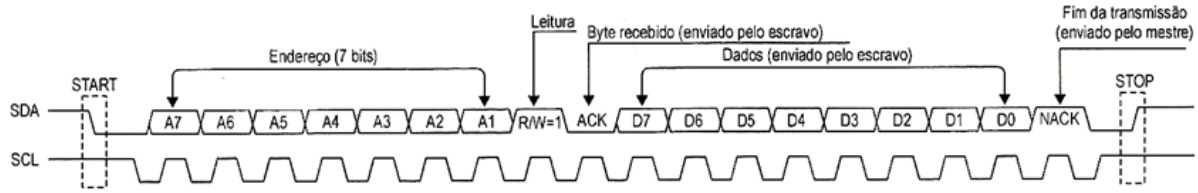


Figura 24 - Processo de Leitura utilizando protocolo I²C.
Fonte: Adaptado de Maciel et al. (2013).

O desempenho do I²C foi definido com velocidade máxima de transmissão de 100 *Kbit/s* (Kilo *bits* por segundo), o que é apropriado para a maior parte das aplicações. Para as aplicações que necessitam de velocidades maiores há o chamado I²C *fast mode* que atinge velocidades de 400 *Kbit/s* e o *High Speed* I²C que permite a comunicação em 3.4 *Mbit/s* (Mega *bits* por segundo) (I2C-BUS, [20--]).

2.5.3.3 Comparação entre os protocolos

De acordo com Maciel et al. (2013), os principais pontos de divergência que se pode citar entre I²C e SPI são:

- Número de pinos que necessita ser maior no SPI já que são necessários, além dos três pinos para a comunicação, um pino para cada dispositivo escravo. No I²C por sua vez, se faz necessário apenas o uso de 2 pinos.
- A velocidade do SPI, que é superior devido ao fato de essa comunicação ser *full-duplex* enquanto a comunicação I²C é *half-duplex*, ou seja, na I²C o fluxo de dados é unidirecional, o que torna a comunicação mais lenta.
- I²C oferece recursos mais avançados que SPI. Em contrapartida, SPI é mais facilmente compreendida e implementada, além de oferecer uma grande flexibilidade para variações e extensões.
- O SPI não possui confirmação de recebimento (*bit* ACK), o que pode ocasionar situações em que o mestre está se comunicando com um dispositivo escravo inexistente. O mesmo não ocorre na I²C.

Diante de todas as diferenças, cabe ao desenvolvedor selecionar qual é o dispositivo mais adequado para a aplicação que está sendo implementada. Vale

ressaltar que ambos os protocolos de comunicação são muito eficientes na interligação entre um microcontrolador e dispositivos periféricos ditos de baixa velocidade, como memórias EEPROM e conversores.

2.5.4 Chaves analógicas

As chaves analógicas consistem em interruptores que ao receberem um sinal de comando são acionadas de maneira a fechar contato entre duas partes de um circuito. Segundo Braga ([20--]), existem circuitos integrados que contém 4 desses interruptores além dos pinos que fazem o acionamento destes. Essa estrutura pode ser vista na Figura 25.

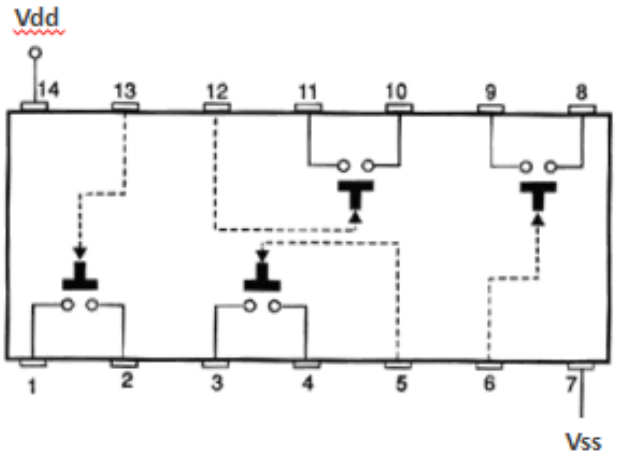


Figura 25 - Exemplo de CI de chaves analógicas.
Fonte: Adaptado de Braga ([20--]).

Na Figura 25 pode-se notar a presença de 4 chaves que podem ser acionadas separadamente de maneira eletrônica e que quando acionadas pelos seus pinos de controle interligam os pinos nos quais estão conectadas, conforme descrito no Quadro 2.

As chaves quando abertas possuem uma resistência de valor muito elevado, da ordem de dezenas de Mega ohms e que quando são fechadas apresentam uma resistência baixa, com valores entre 150 Ω (ohms) e 300 Ω dependendo do modelo do CI utilizado (BRAGA, [20--]).

Pino de acionamento	Pinos ligados
5	3 e 4
6	8 e 9
12	10 e 11
13	1 e 2

Quadro 2 – Acionamentos das chaves no CI de exemplo da Figura 25.
Fonte: Autoria Própria.

Os CIs de chaves analógicas podem operar de maneira digital ou analógica de acordo com o tipo de sinal que passa pelas chaves. Na operação analógica o CI é alimentando de forma simétrica, ou seja, há uma tensão positiva em Vdd e em Vss uma tensão negativa do mesmo módulo de Vdd. O funcionamento desse circuito ocorre quando há a aplicação de um sinal de controle para cada chave, que permite a passagem do sinal analógico que está em um de seus lados. Vale ressaltar que a amplitude do sinal de entrada não deve ultrapassar a tensão de alimentação do circuito (BRAGA, [20--]). Um exemplo de funcionamento do circuito nesse modo pode ser observado na Figura 26, na qual o CI é alimentado com uma fonte simétrica de 5 V (Volts).

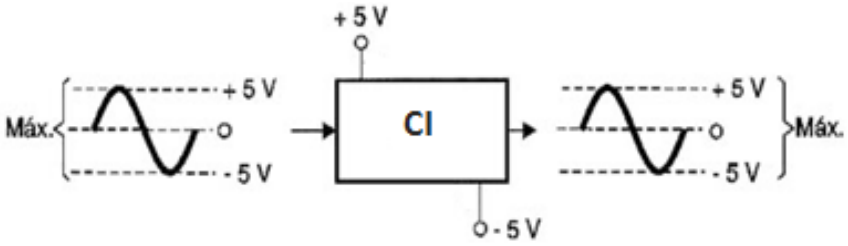


Figura 26 - Operação Analógica de um CI de chaves analógicas.
Fonte: Adaptado de Braga ([20--]).

Na operação digital, por sua vez, Vdd recebe uma tensão e Vss é ligado ao terra do circuito (BRAGA, [20--]). O funcionamento se dá através do nível lógico que chega ao pino de controle que, quando se encontra em nível lógico baixo deixa a chave aberta e em nível lógico alto a fecha (HESSEL et al., 2008). Um exemplo dessa forma de operação pode ser visto na Figura 27.

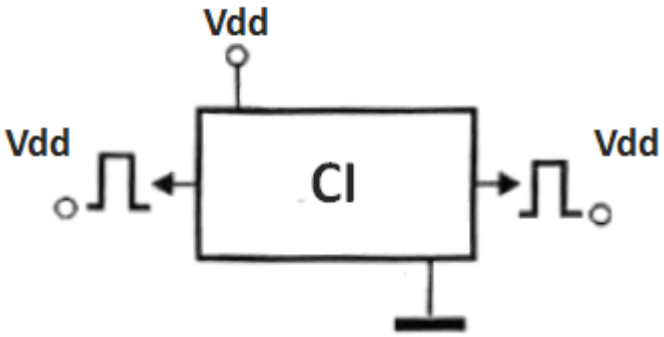


Figura 27 - Operação Digital de um CI de chaves analógicas.
Fonte: Adaptado de Braga ([20--]).

Vale Ressaltar que na operação digital o nível de tensão do sinal é o mesmo da alimentação do CI, no caso, Vdd (BRAGA, [20--]).

2.5.5 Comunicação serial RS-232 com o computador

Criado em 1962, o padrão RS-232 passou por três revisões até chegar no RS-232C que se tornou popular e foi o primeiro a ser utilizado na comunicação com computadores pessoais (PC) , sendo o mais utilizado para esse fim até o final dos anos 90 quando começou a ser substituído pelo padrão USB, porém mesmo com essa substituição gradual ainda é amplamente utilizado até hoje em setores como indústria, medicina e algumas aplicações específicas. Isso se deve a maior robustez e facilidade de implementação que o RS-232 apresenta quando comparado ao USB.

Esse protocolo utiliza sinais de tensão negativa para indicar nível lógico alto e sinais de tensão positiva para indicar nível lógico baixo, tendo valores de tensão variando de ±3V até ±25V dependendo da aplicação (SILVA et al., 2013). A figura 28 indica o funcionamento do protocolo.

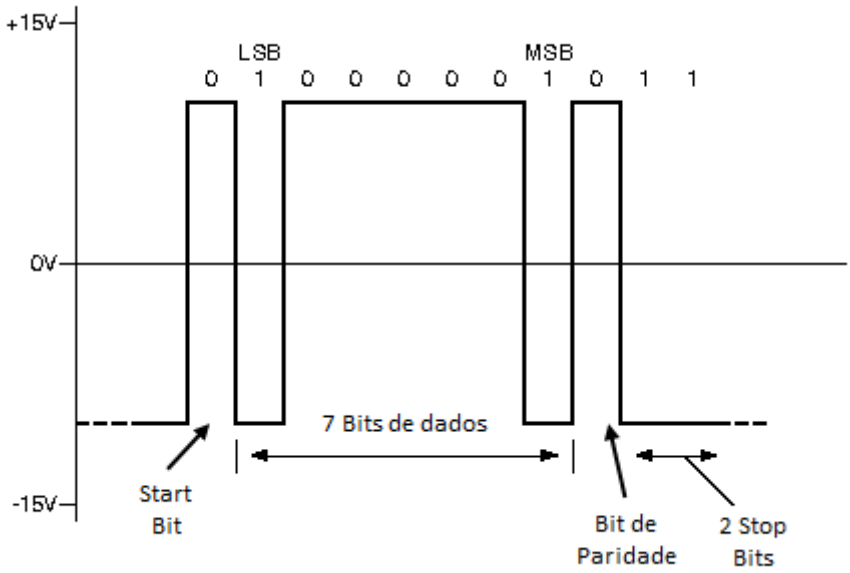


Figura 28 - Exemplo de comunicação RS-232.
Fonte: Adaptado de Silva et al. (2013).

Na Figura 28 pode-se observar a ordem de transmissão ou recepção dos *bits* (SILVA et al., 2013):

- *Start Bit:* Bit que indica o início da comunicação;
- *Bits* de dados: *Bits* que carregam a informação que se deseja transmitir ou receber.
- *Bit* de paridade: *Bit* para reconhecimento de erros. Seu valor indica a quantidade de *bits* 1 presentes na informação enviada. Caso o *bit* de paridade não seja o mesmo no transmissor e no receptor acusa-se um erro.
- *Stop bits*: Bits que indicam o final da comunicação.

O número de *bits* é configurável no dispositivo que realiza a comunicação, havendo a possibilidade de a comunicação ser, de acordo com as necessidades do usuário, configurada com os diferentes parâmetros mostrados a seguir (MESSIAS, 2007):

- *Bits* de dados: podem ser 5, 6, 7 ou 8.
- Tipo de paridade: par, marca, ímpar ou nenhuma.
- *Stop bits*: 1, 1.5 ou 2.

No que diz respeito aos conectores, o padrão especifica os tipos DB-9 e DB-25, que possuem 9 e 25 pinos respectivamente (SILVA et al., 2013). A Figura 29 mostra os pinos do conector DB-9 e no Quadro 3 observa-se a descrição de cada um.

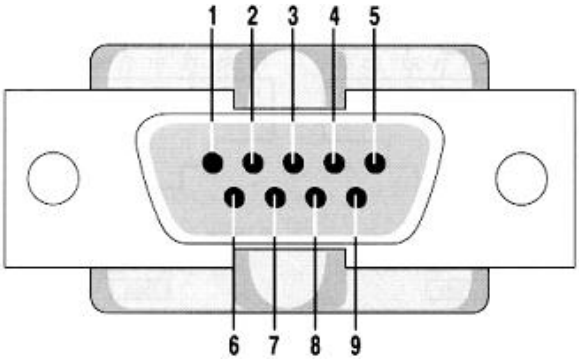


Figura 29 - Pinos do conector DB-9.
Fonte: Adaptado de Silva et al. (2013).

Pino	Sinal	Pino	Sinal
1	*Data Carrier Detect*	6	*Data Set Ready (DSR)*
2	*Received Data*	7	*Request To Send (RTS)*
3	*Transmitted Data*	8	*Clear to Send*
4	*Data Terminal Ready*	9	*Ring Indicator*
5	*Signal Ground*		

Quadro 3 - Descrição dos pinos do conector DB9.
Fonte: Autoria Própria.

A Figura 30 mostra o conector DB-25 e no Quadro 4 observa-se a descrição de cada um.

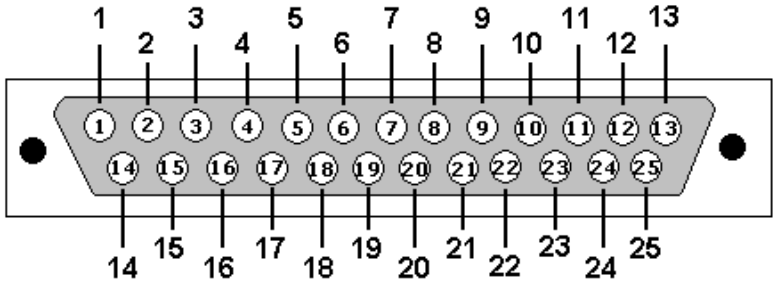


Figura 30 - Pinos do conector DB-25.
Fonte: Adaptado de Silva et al. (2013).

Pino	Sinal	Pino	Sinal
1	*Frame Ground*	14	*Sec. Transmitted Data*
2	*Transmitted Data*	15	*Transmitter Sig. Element Timing*
3	*Received Data*	16	*Sec. Received Data*
4	*Request To Send*	17	*Receiver Sig. Element Timing*
5	*Clear To Send*	18	*Undefined*
6	*Data Set Ready*	19	*Sec. Request To Send*
7	*Signal GND/Common Return*	20	*Data Terminal Ready*
8	*Received Line Signal Detector*	21	*Signal Quality Detector*
9	*V+*	22	*Ring Indicator*
10	*V-*	23	*Data Sig. Rate Selector (PC)*
11	*Undefined*	24	*Data Sig. Rate Selector (Perif.)*
12	*Sec. Rcvd. Line Sig. Detector*	25	*Undefined*
13	*Sec. Clear To Send*		

Quadro 4 – Descrição dos pinos do conector DB-25.
Fonte: Autoria Própria.

Nota-se no Quadro 4 a presença de pinos secundários (*sec.*), sendo possível através da utilização destes a comunicação com dois dispositivos simultaneamente, porém com grandes limitações de velocidade. Nota-se também a presença de pinos indefinidos (*undefined*), havendo ainda alguns outros que não são obrigatórios para o funcionamento da interface (SILVA et al., 2013). No Quadro 5 pode-se verificar alguns dos sinais utilizados na comunicação.

Atualmente o conector mais usado é o DB-9 e, muitas vezes, em uma configuração simples chamada *3-wire* na qual apenas os pinos 2 (Recepção de dados), 3 (transmissão de dados) e 5 (terra) são conectados, sendo o controle do fluxo de dados, sinalização e sincronização realizadas via *software*. Outra configuração popular é a chamada *5-wire* onde além dos 3 pinos da configuração *3-wire*, há a ligação dos pinos 6 (DSR) e 7 (RTS) para um melhor controle do fluxo.

A comunicação RS-232 é do tipo *full-duplex*, o que proporciona um ganho de velocidade com sua utilização, porém suas maiores aplicações são para ligação de

dispositivos em distâncias curtas (15 metros) e com velocidades baixas de comunicação (próximas a 115200 bps – *bits* por segundo) (SILVA et al. 2013).

Sinal	**Descrição**
Frame Ground	Terra da carcaça do conector
Transmitted Data	Dados transmitidos
Received Data	Dados Recebidos
Request To Send (RTS)	Requisição de início de transmissão
Clear To Send	Resposta ao sinal RTS
Data Set Ready (DSR)	Indicador de Funcionamento
Signal Ground / Common Return	Terra de referência
Received Line Signal Detector	Indica o estado da linha de recepção
Transmitter Signal Element Timing	Clock do Transmissor
Receiver Signal Element Timing	Clock do Receptor
Data Terminal Ready	Periférico pronto para recepção
Signal Quality Detector	Qualidade do Sinal
Ring Indicator	Indicador de Chamada (usado em modems)
Data Signal Rate Selector	Seleção de *Baud Rate*

Quadro 5 – Descrição de sinais utilizados na comunicação RS-232.
Fonte: Autoria própria.

2.5.6 Linguagens de Programação

Segundo Canalle e Araújo ([20--]), linguagens de programação são responsáveis por facilitar a transmissão de instruções para o computador. Existem várias linguagens de programação disponíveis para programação de microcontroladores e para implementação de *software* em computadores. Essa seção irá abordar brevemente algumas dessas linguagens tratando aspectos como histórico e aplicações.

2.5.6.1 C

A linguagem C surgiu na década de 70 como uma evolução de outra linguagem chamada B. Com a evolução dos computadores se tornou popular entre os desenvolvedores de *software* por ser uma linguagem estruturada (baseada em mecanismos de sequência, seleção e repetição), com fácil portabilidade entre diferentes *hardwares* e sistemas operacionais, alta interatividade com o sistema operacional, código compacto e rápido e que ao mesmo tempo dá liberdade para o programador (CAMPOS, 2010).

Com relação a eficiência, Campos (2010) diz que “Usando C, um programador pode conseguir aproximadamente a eficiência de código *Assembly* combinada com a programação estruturada”, ou seja, a linguagem C é bastante eficiente em termos de velocidade de execução. Muito desse aspecto se deve ao fato de o C ser uma linguagem compilada, ou seja, ao contrário de linguagens como o BASIC, C possui um compilador que lê todo o programa e o converte em linguagem de máquina, gerando um arquivo executável que fica salvo na memória e pode ser acessado a qualquer momento sem a necessidade de interpretação do código-fonte em tempo real.

A união de eficiência e programação estruturada faz da linguagem C uma das mais utilizadas entre os programadores para programação de microcontroladores e desenvolvimento de *softwares* computacionais (LENZ, [20--]).

2.5.6.2 C++

A linguagem C++ foi desenvolvida inicialmente como uma extensão da linguagem C para uso em programação orientada a objeto. É uma linguagem de propósito-geral que possui comandos simples das linguagens de alto-nível ao mesmo tempo em que possui comandos complexos das linguagens de baixo-nível.

Possui recursos como possibilidade de manipulação direta da memória através da utilização de ponteiros, mecanismo de tratamento de exceções e utilização de bibliotecas pré-implementadas de comandos. Outra característica é o fato de, tal qual C, ser uma linguagem compilada, o que a faz ser muito eficiente.

Como é derivado de C, muitas das funções de C funcionam em C++, o que permite a utilização dessa linguagem para a programação de microcontroladores.

Também é amplamente utilizada na implementação de *softwares* computacionais com interfaces gráficas através de sua utilização em ambientes de desenvolvimento de *software* (CANALLE, ARAÚJO, [20--]).

2.5.7 Ambiente de desenvolvimento de *software* para computador

É de grande importância a utilização de um ambiente de desenvolvimento (também chamado de IDE – *Integrated Development Environment*) para desenvolvimento de *software* para computadores, já que a utilização de um recurso desse tipo além de facilitar o desenvolvimento, oferece uma grande variedade de recursos para o programador. Essa seção abordará algumas características da IDE utilizada para o desenvolvimento do *software* computacional.

2.5.7.1 C++ Builder

O C++ Builder da Borland é um ambiente visual, orientado a objetos que visa o desenvolvimento rápido de aplicações de propósito geral, ou cliente-servidor, utilizando a linguagem C++. Utiliza um ambiente de ferramentas RAD (*Rapid Application Development*) que possui uma grande biblioteca de componentes visuais para agilizar o desenvolvimento de aplicações gráficas (CHACON, [20--]).

A partir da versão XE passou a ser distribuído pela Embarcadero® que hoje fornece no mercado a versão XE7 que possui suporte para diferentes sistemas operacionais. (EMBARCADERO, 2014).

Oferece recursos como (CHACON, [20--]):

- *Drag and drop* que permite criar interfaces gráficas de maneira rápida e fácil;
- Editor de texto com realce de códigos e recurso de auto-completar;
- Suporte a bibliotecas com funções de sistemas operacionais como o *Windows*;
- Fácil visualização e edição de propriedades de objetos utilizados no projeto;
- Depuração eficiente com visualização de recursos do sistema.

Uma visão geral da interface da IDE C++ Builder, na versão 6, é apresentada na Figura 31.

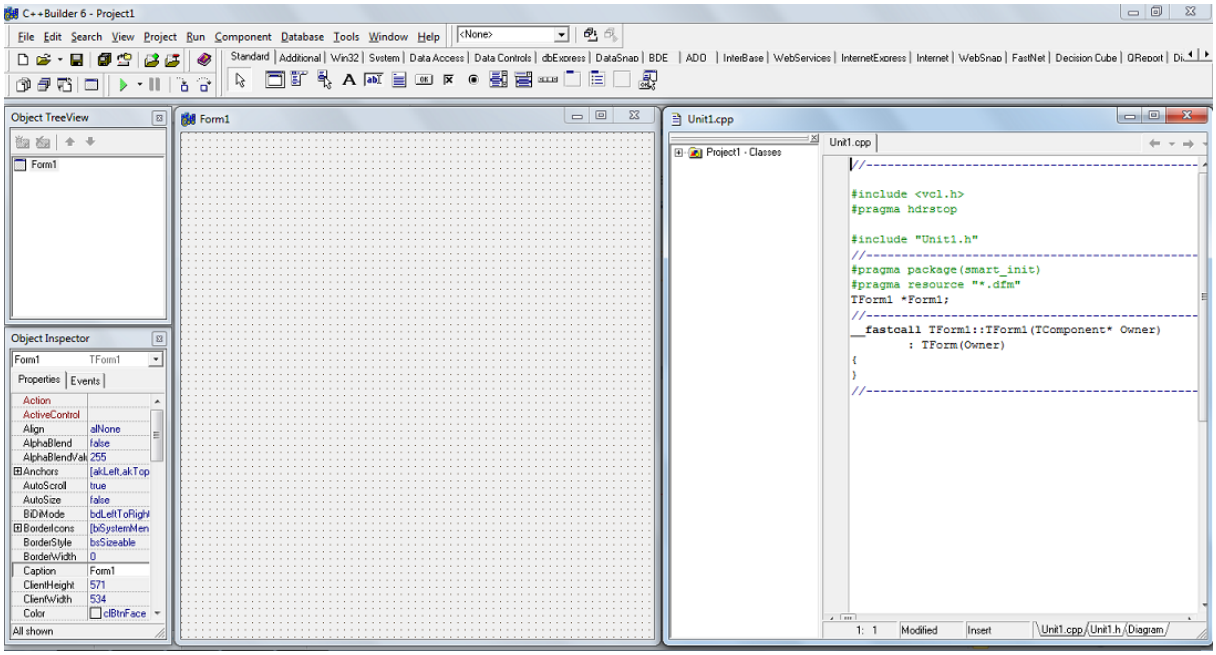


Figura 31 - Tela inicial da IDE C++ Builder versão 6.

Fonte: Autoria Própria.

2.5.8 Ambiente de desenvolvimento de *firmware*

De acordo com IEEE (2002) *firmware* pode ser definido como uma combinação de um dispositivo de *hardware*, instruções computacionais e dados que podem somente serem lidos em um dispositivo. Sendo assim, um ambiente de desenvolvimento de *firmware* deve ser capaz de proporcionar uma integração entre o *hardware* de um dispositivo (um microcontrolador, por exemplo) e o conjunto de instruções específicas para este dispositivo, de maneira a permitir que esse *hardware* opere de maneira correta quando em funcionamento.

Existe uma grande variedade de IDEs disponíveis no mercado atualmente para desenvolvimento de *firmware*, cada uma com características próprias que podem ser mais ou menos interessantes de acordo com as necessidades do programador. Os itens a seguir irão descrever brevemente a IDE utilizada no projeto, dando uma visão geral sobre ela e mostrando alguns recursos que possui.

2.5.8.1 µVision

A IDE µVision é uma ferramenta da linha Keil™ desenvolvida pela ARM® que combina gerenciamento de projeto, edição de código fonte e *debug* de código, além de oferecer um amplo suporte para simulação em um único ambiente. Oferece suporte para desenvolvimento em linguagem C/C++ para microcontroladores de 8, 16 e 32 *bits* das famílias 8051 e ARM®.

Possui recursos que facilitam o desenvolvimento de código fonte para microcontroladores, tais como (KEIL, 2009):

- *Highlight* em diferentes cores das diferentes partes do código;
- Identação otimizada para linguagem C/C++;
- Barra de ferramentas completa;
- Possibilidade de edição do código durante o processo de depuração, o que facilita a correção de possíveis erros;
- Recurso de auto-completar que torna a programação mais ágil;
- Verificação de erros em tempo real, que auxilia na redução dos erros de programação.

A Figura 32 mostra a interface da IDE Keil™ µVision na versão 4.

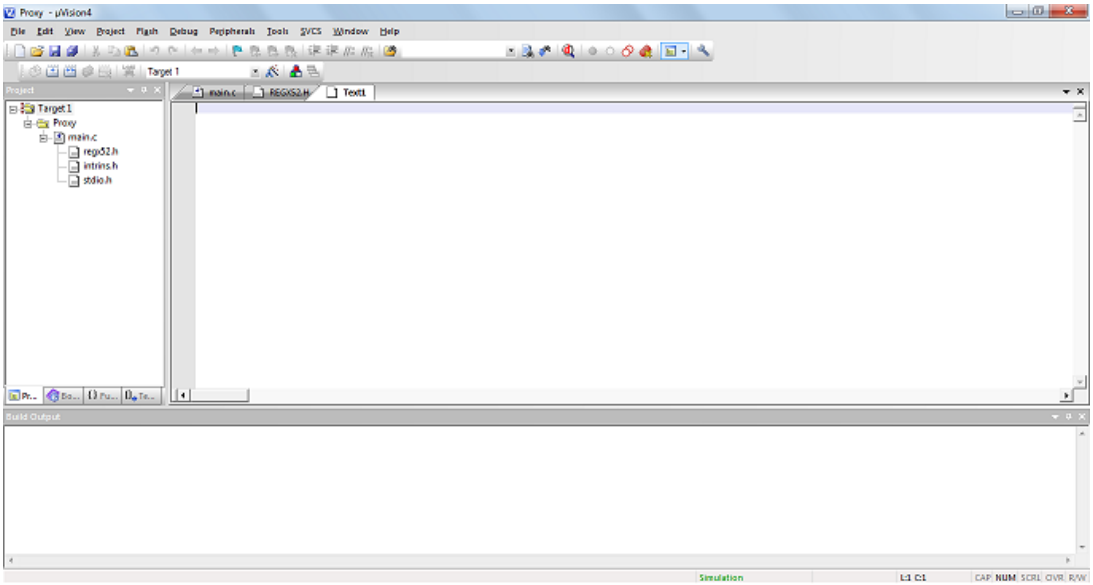


Figura 32 - Interface da IDE Keil µVision® 4.

Fonte: Autoria Própria.

2.5.9 Controle *DualShock® 3*.

O controle *DualShock® 3* foi desenvolvido pela *Sony©* para ser o controle do console *PlayStation® 3*. É um dos controles mais modernos atualmente no mercado, possuindo recursos que incrementam a jogabilidade dos *games* como, por exemplo, comunicação sem fio (tecnologia *bluetooth*) com o console, direcionais analógicos e mecanismo de vibração (PLAYSTATION, 2014). Para alimentação, o controle utiliza uma tensão de 5V e uma corrente de 500mA, sendo alimentado por uma bateria recarregável de íon de lítio que oferece uma tensão de 3.7V e tem uma capacidade de 610 mAh (mili Ampere-hora) (SONY COMPUTER ENTERTAINMENT INC., 2011).

As partes integrantes do controle *DualShock® 3* são apresentadas na Figura 33, onde pode-se observar as seguintes partes (PLAYSTATION KNOWLEDGE CENTER, 2014):

- *Port indicators*: 4 LEDs que identificam quando o controle está sendo usado. Também sinalizam quando a bateria se encontra fraca ou carregando;
- *Buttons* L1, L2, R1 e R2: Botões de ombro. Executam ações do jogo.
- *USB conector*: Conexão USB. Conexão do controle com o console ou computador para que a bateria seja carregada;
- *Directional buttons*: Direcionais digitais;
- *Buttons* O, X, Δ e ☐: Realizam ações do jogo;
- *Left Stick* / *L3 Button* e *Right Stick* / *R3 button*: Direcionais analógicos. Os botões L3 e R3 são acionados quando os direcionais são pressionados;
- *SELECT Button*: Botão de *select*;
- *START Button*: Botão de *start*;
- *PS button*: Botão PS. Liga o controle e acessa configurações e ferramentas do console.

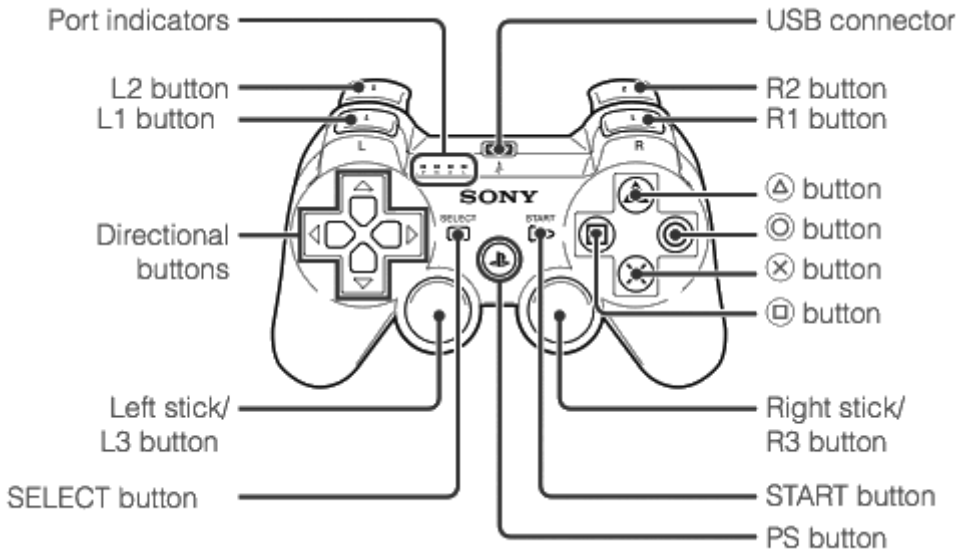


Figura 33 - Partes do controle *DualShock® 3*.
Fonte: Adaptado de *PlayStation Knowledge Center* (2013).

3 DESENVOLVIMENTO

3.1 COMPONENTES DE *HARDWARE*

3.1.1 Critérios adotados na escolha dos componentes de *hardware*

Para o desenvolvimento do *hardware* do projeto foram levados em conta os seguintes fatores:

- Análise da relação custo-benefício, já que há uma limitação na aquisição de certos componentes por indisponibilidade no mercado local. Essa indisponibilidade acarreta um custo elevado que pode ser evitado através da utilização de componentes mais facilmente adquiridos, com menor custo e que possuem uma aplicação similar, não interferindo de maneira direta no funcionamento do equipamento.
- Familiaridade com a utilização de determinados componentes, já que ao longo do curso foram abordadas algumas tecnologias que podem ser utilizadas para os fins desejados nesse projeto e com as quais há uma

facilidade no que diz respeito à utilização. Sendo assim, foi determinante na escolha dos componentes o fato de já se ter o conhecimento necessário para manipulação destes.

3.1.2 Componentes necessários para a confecção do projeto

Após a análise dos requisitos do projeto, chegou-se a conclusão de que havia a necessidade de aquisição dos seguintes componentes:

- Microcontrolador;
- Memória EEPROM;
- Chaves analógicas com comando digital;
- Componentes para circuito de comunicação serial;
- Regulador de Tensão;
- Cristal de Quartzo;
- Resistores, diodos e capacitores;
- Conectores;

Os componentes escolhidos, suas funções no circuito e a descrição dos mesmos serão apresentados nas próximas sessões.

3.1.2.1 Escolha do Microcontrolador

Adotando os critérios citados na seção 3.1.1, o microcontrolador escolhido para utilização no projeto foi o AT89S52 da Atmel©. É um microcontrolador CMOS de 8 *bits* com 8K (Kilo) *bytes* de memória flash interna e que é compatível com a família 8051 no que diz respeito ao conjunto de instruções e à pinagem. Oferece memória *flash In-System Programmable* (ISP), ou seja, permite que a memória *flash* seja gravada diretamente no sistema. Com a combinação da gravação ISP com uma CPU de 8 *bits* em um chip monolítico, o AT89S52 é um microcontrolador que oferece uma solução flexível e de baixo custo para variadas aplicações (ATMEL, 2008). A figura 34 mostra o diagrama de blocos do AT89S52.

O AT89S52 possui, além dos já citados 8K *bytes* de memória *flash*, as seguintes especificações (ATMEL, 2008):

- 256 *bytes* de memória RAM (*Random Access Memory*);
- 32 pinos de entrada e saída divididos em 4 *ports* de 8 pinos cada;
- 3 temporizadores/contadores (*timer/counters*);
- Suporte para comunicação serial *full-duplex* RS-232;
- 6 fontes de interrupção;
- Faixa de tensão de operação entre 4V e 5.5V;
- Corrente de saída de até 15 mA (mili Amperes);
- Frequência de operação com cristal de quartzo de até 33 MHz (Mega Hertz);

Essas especificações oferecidas são compatíveis com o que é necessário para o projeto proposto.

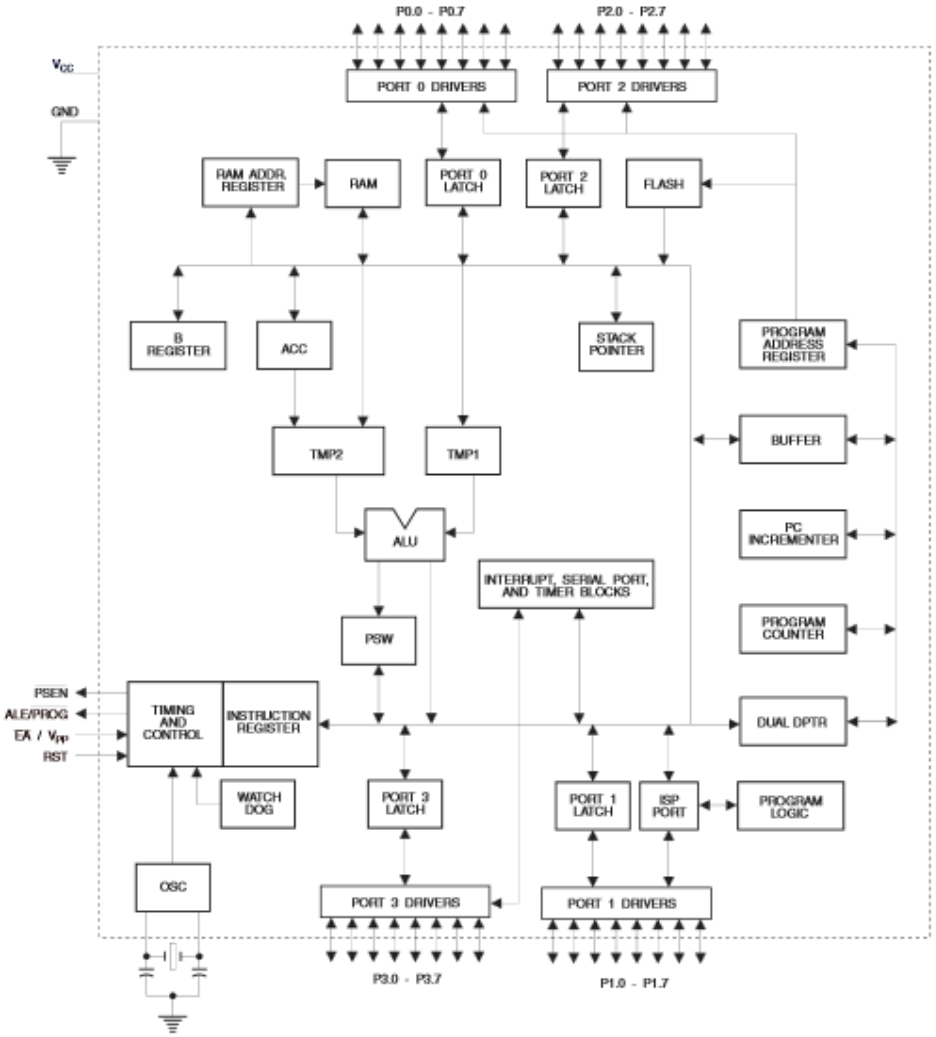


Figura 34 - Diagrama de Blocos do AT89S52.

Fonte: ATMEL, 2008

3.1.2.2 Escolha da Memória EEPROM

Tendo em vista que o AT89S52 não possui memória EEPROM interna para armazenamento de dados, se faz necessária à utilização de uma memória EEPROM externa para se armazenar os combos. A memória EEPROM escolhida foi a AT24C512 da Atmel© que é organizada em 512 páginas de 128 *bytes* cada, resultando em um total de 65K *bytes* de capacidade de armazenamento. A figura 35 mostra o diagrama de blocos do AT24C512 e nela pode-se notar que além dos pinos de alimentação (VCC e GND), esse CI de memória possui (ATMEL, 2008):

- 3 pinos de endereçamento (A0, A1 e A2): São utilizados na necessidade de ligação de mais de uma memória no mesmo barramento. Sendo 3 pinos de endereçamento, pode-se conectar até um total de 8 (2^3) memórias no mesmo barramento.
- SDA e SCL: Pinos que realizam a comunicação serial do tipo I^2C com o microcontrolador.
- WP (*Write Protect*): Pino que faz com que a memória fique bloqueada (caso esse pino esteja conectado a VCC) ou liberada (caso esse pino esteja conectado a GND) para gravação.

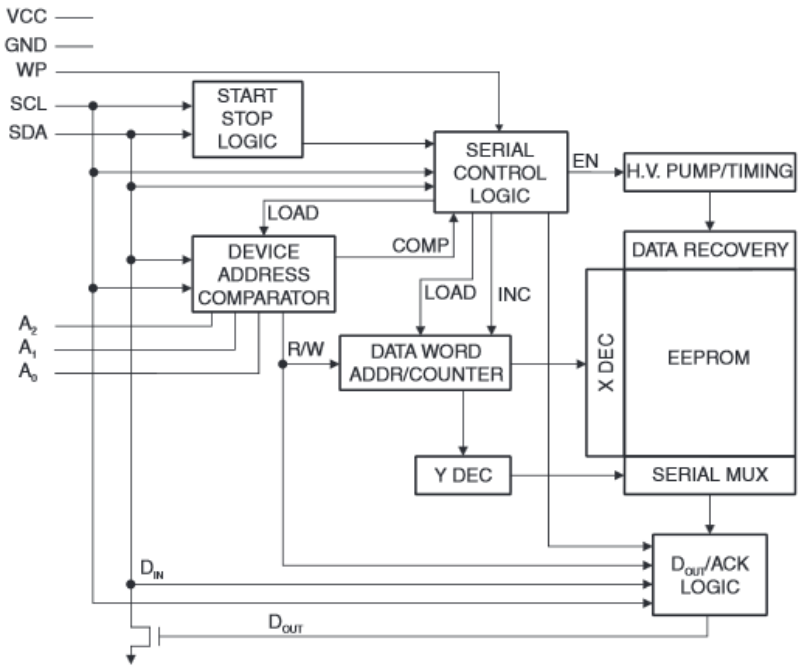


Figura 35 - Diagrama de Blocos do AT24C512.

Fonte: ATMEL, 2008.

Com a utilização dessa memória é possível o armazenamento de combos extensos por parte do usuário, atendendo assim uma das especificações do projeto. O protocolo de leitura e escrita na EEPROM está detalhado no Apêndice C e o código para implementação desse protocolo no Apêndice D.

3.1.2.3 Chaves analógicas com comando digital

De acordo com os critérios expostos na seção 3.1.1, o CI de chave analógica com acionamento por comando digital utilizado foi o CD4066. Esse CI opera com as seguintes especificações (TEXAS INSTRUMENTS, 2003):

- Tensão de entrada de 3 a 18 Volts para operação ideal;
- Resistência de chave fechada de aproximadamente 125Ω quando alimentado com 5V;
- Corrente de entrada de ±10 mA;
- Tensão de saída de no mínimo 4.6V quando alimentado com uma tensão de 5V;
- Tempo de acionamento e desligamento das chaves de 20ns (nano segundos).

As condições de operação do CD4066 estão de acordo com o que é necessário para confecção do projeto, já que os pinos de acionamento desse CI vão conectados diretamente aos pinos de saída do microcontrolador que oferecem tensão e corrente suficientes para o comando de acionamento das chaves. As saídas do CD4066 fecham contato entre os pinos do controle *DualShock® 3*, que opera com a tensão de saída fornecida pelas chaves. Vale ressaltar também que o tempo de acionamento/desligamento não influencia na jogabilidade e nem na gravação e execução dos combos.

3.1.2.4 Regulador de tensão

Os reguladores de tensão têm como função ajustar a tensão que entra em um circuito para níveis com os quais os componentes desse circuito operam. No caso, a tensão de entrada que vem da fonte de alimentação é de 9V com os componentes

operando em 5V, portanto é necessário que se utilize um regulador que ajuste a tensão de operação do circuito para 5V. O regulador escolhido para esse fim foi o LM7805.

O LM7805 é um regulador de tensão para 5V. Possui encapsulamento TO-220 com 3 terminais, sendo eles entrada (*input*), terra (GND) e saída (*output*). A tensão de entrada (no caso 9V) é colocada no pino 1 (*input*) e a tensão de saída é coletada a partir do pino 3 (*output*).

O diagrama de blocos do LM7805 pode ser visto na Figura 36.

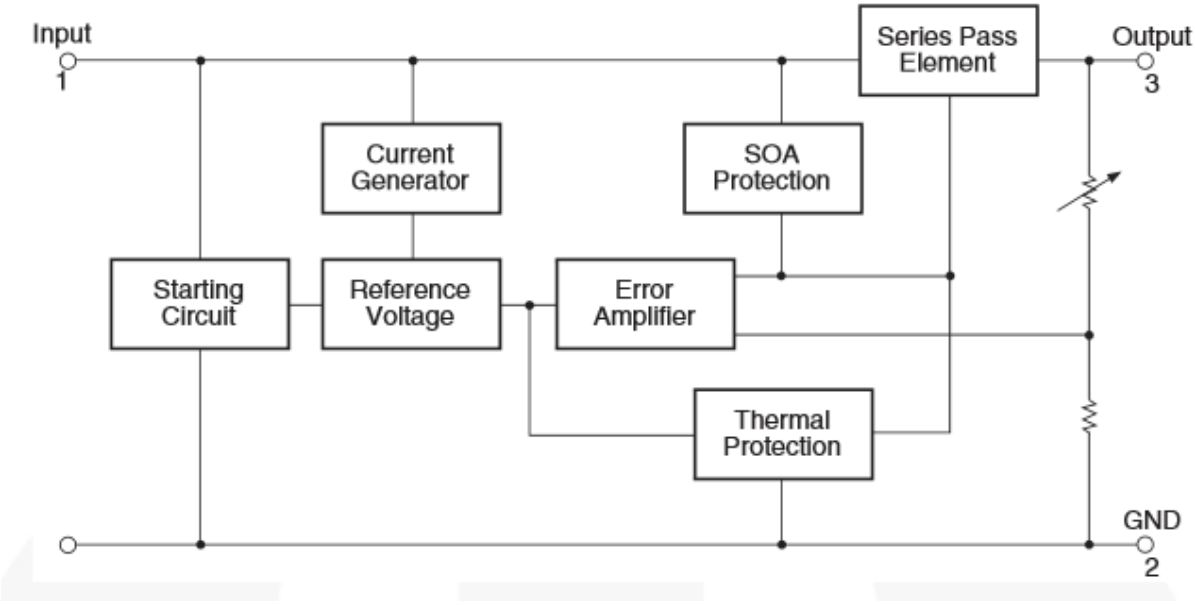


Figura 36 - Diagrama de blocos do LM7805.
Fonte: Adaptado de Fairchild Corp. (2014).

Sua operação se dá com os seguintes valores de tensão e corrente (FAIRCHILD CORP., 2014):

- Tensão de entrada: Até 35V;
- Tensão de saída: De 4.75V até 5.25V;
- Corrente de saída: De 5mA até 1A (Ampere).

O circuito de ligação do LM7805 é mostrado na Figura 37, na qual se pode observar a presença de dois capacitores (CI e CO) que servem para desacoplamento e filtro da entrada e da saída.

A tensão regulada pelo 7805 alimentará os CIs do circuito (microcontrolador AT89S52, memória EEPROM AT24C512 e chaves analógicas CD4066).

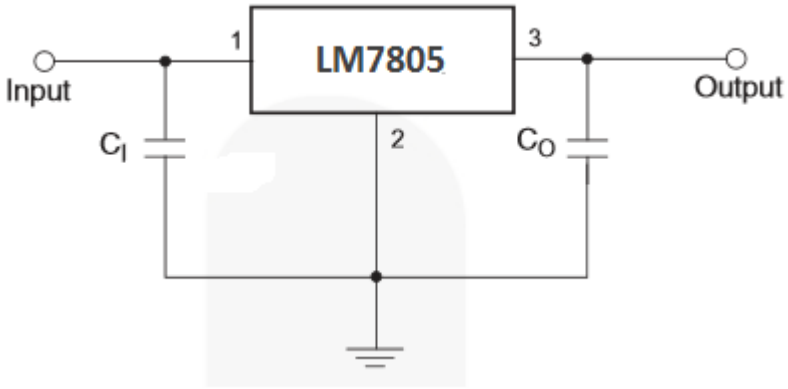


Figura 37 - Circuito básico de ligação do LM7805.
Fonte: Adaptado de Fairchild (2014).

3.1.2.5 Circuito de comunicação serial com o computador

Para a comunicação serial RS-232 com o computador foi utilizado o circuito mostrado na Figura 38.

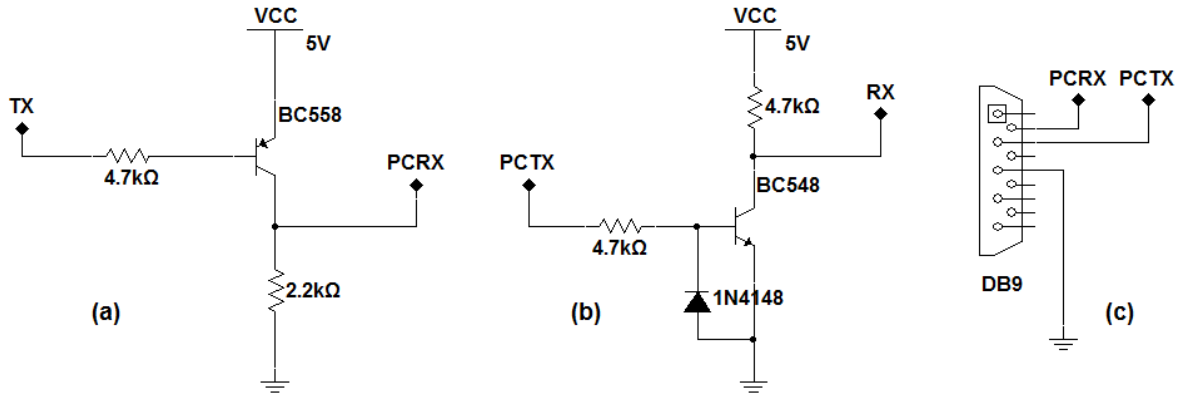


Figura 38 - Circuito para comunicação serial com o computador. (a) Transmissão do microcontrolador para o computador; (b) Transmissão do computador para o microcontrolador; (c) Conector DB-9 utilizado na comunicação.
Fonte: Autoria Própria.

Como se pode observar na Figura 46, esse circuito é composto por:

- 1 transistor NPN (BC548);
- 1 transistor PNP (BC558);

- 1 diodo 1N4148;
- 1 Conector DB-9 fêmea;
- Resistores diversos;

Nesse circuito, RX e TX são os sinais de referência para o microcontrolador, sendo ligados nos pinos 10 (P3.0/RX) e 11 (P3.1/TX) do AT89S52. Esses sinais indicam recepção (RX) ou transmissão (TX) de dados do microcontrolador para o computador via comunicação serial. Vale lembrar que o AT89S52 oferece suporte para comunicação serial em 4 modos diferentes com as seguintes configurações (NICOLOSI, 2007):

- Modo 0: 8 *bits* de dados recebidos e enviados via pino de RX, sendo o pino de TX utilizado para geração do *clock* (comunicação síncrona, *half-duplex*);
- Modo 1: 10 bits de dados enviados ou recebidos, sendo 1 *start bit*, 8 *bits* de dados e 1 *stop bit*. A recepção de dados é feita no pino de RX e a transmissão no pino de TX (Comunicação assíncrona, *full-duplex*, sem *bit* de paridade);
- Modos 2 e 3: 11 bits de dados enviados, sendo 1 *start bit*, 8 *bits* de dados, 1 bit programável e 1 *stop bit.* A recepção de dados é feita no pino de RX e a transmissão no pino de TX (Comunicação assíncrona, *full-duplex*, com *bit* programável podendo ser utilizado como *bit* de paridade);

Os sinais de referência PCRX e PCTX, por sua vez, são ligados, respectivamente, nos pinos 2 e 3 do conector DB-9 e dele no computador. Indicam recepção (PCRX) ou transmissão (PCTX) de dados do computador para o microcontrolador.

O funcionamento do circuito se dá pela realização de uma conversão de nível de tensão. Como o padrão RS-232 opera com tensões negativas para indicar nível lógico alto (*bit* 1) e tensões positivas para indicar nível lógico baixo (*bit* 0), o chaveamento dos transistores faz com que um nível de tensão de 5V que sai do microcontrolador chegue ao computador com um nível de tensão que ele compreenda como nível lógico 1 ou 0. Vale ressaltar que os níveis de tensão de

operação da porta serial são maiores que 5V, o que faria com que o microcontrolador queimasse no caso de essa tensão chegar até ele.

No circuito (a), quando se deseja enviar um *bit* 0 pela serial, o pino TX do microcontrolador recebe um nível de tensão baixo, o que chaveia o transistor PNP fazendo com que um nível de tensão alto chegue a PCRX identificado pelo computador como um *bit* 0. Quando o bit que se deseja enviar é um *bit* 1, um nível de tensão alto em TX faz com que transistor não seja chaveado e o nível de tensão permaneça baixo, o que faz com que o computador identifique esse nível baixo com uma tensão negativa, o que representa um *bit* 1.

Para o circuito (b), um nível de tensão negativo que vem do computador é desviado para o terra pelo diodo presente no circuito, não fazendo o chaveamento do transistor NPN e mantendo 5V em RX, o que equivale a recepção de um bit 0. Quando um nível de tensão positivo chega em PCTX o transistor é chaveado ligando RX ao terra, o que equivale a recepção de um bit “1”.

3.1.2.6 Outros componentes

Os outros componentes utilizados são:

- Cristal de quartzo: O cristal tem a função de gerar *clock* para o funcionamento do microcontrolador. O cristal escolhido para utilização no circuito tem valor de 22.1184 MHz. Esse valor foi escolhido por gerar uma contagem de tempo com o mínimo erro possível (o processo de geração de tempo é mostrado no código do Apêndice D).
- Conectores: Ligam o circuito aos componentes externos. Serão ligados à placa pelos conectores o controle *DualShock® 3* e o controle arcade. Também há conectores do tipo KRE que servem como entrada para alimentação do circuito e saída para alimentação do *DualShock® 3*. O outro conector do circuito é uma barra de pinos dupla que serve para gravação ISP no AT89S52.
- Capacitores, resistores e diodos: Possuem funções diversas. Servem entre outras coisas como circuito base para os reguladores e CIs, além de fazerem parte do circuito de comunicação serial.

3.2 COMPONENTES DE *SOFTWARE*

3.2.1 Critérios adotados para a escolha dos componentes de *software*

O principal critério adotado para escolha de componentes de *software* foi a familiaridade tanto com as linguagens de programação, quanto com as IDEs. Essa familiaridade é importante, pois com ela o processo de desenvolvimento é feito de maneira mais fácil e rápida. A seguir serão descritos os componentes de *software* necessários para execução do projeto.

3.2.2 Componentes de *software* necessários

Através da análise dos requisitos do projeto, concluiu-se que os componentes de *software* necessários para o desenvolvimento do projeto são:

- Linguagens de programação para *software* computacional e *firmware*;
- IDEs para *software* computacional e *firmware*.

A seguir serão descritas as ferramentas escolhidas com base nos critérios expostos na seção 3.2.1.

3.2.2.1 Linguagem de programação de *firmware*

A linguagem de programação escolhida para programação do *firmware* foi a linguagem C. O principal fator, além da familiaridade descrita na seção 3.2.1, é o fato de C ser uma linguagem que opera de maneira otimizada em microcontroladores mesmo oferecendo instruções de alto nível, o que faz com que a execução do programa seja rápida e a programação seja mais simples.

3.2.2.2 IDE para *firmware*

A IDE escolhida para *firmware* foi o Keil µVision versão 4. Um fator determinante para escolha é o fato de essa IDE oferecer suporte a programação em

linguagem C além de possuir as bibliotecas necessárias para desenvolvimento de código-fonte no AT89S52.

3.2.2.3 Linguagem de programação do *software* computacional

Para o *software* computacional a linguagem de programação escolhida foi C++. Essa linguagem possui recursos úteis para desenvolvimento de *software* e oferece suporte para o desenvolvimento facilitado dos requisitos do projeto.

3.2.2.4 IDE para *software* computacional

A IDE escolhida para implementação do *software* computacional foi o C++ Builder versão 6, da Borland. Essa IDE foi escolhida, dentre outros fatores, por oferecer suporte a programação em linguagem C++, além de facilidade para desenvolvimento de interfaces gráficas e um editor de texto que auxilia e agiliza a programação.

3.3 CONFECÇÃO DA PLACA DE PROTÓTIPO

A placa de protótipo para testes foi feita utilizando o *software* Proteus versão 7 da *Labcenter Electronics*. Foram interligados todos os componentes citados na seção 3.1.2 resultando no diagrama esquemático do apêndice A.

Serão descritos nessa seção as partes que compõe o circuito com a descrição de cada uma.

3.3.1 Alimentação

O circuito de alimentação é composto pelo regulador de tensão LM7805 descrito na seção 3.1.2.4. A tensão de entrada vem de um conector KRE (J2) no qual é conectada uma fonte que fornece uma tensão de 9V. Essa tensão passa por um diodo 1N4007 para proteção do circuito (evita a passagem de corrente no caso da inversão na ligação de VCC e terra) e é regulada para 5V que é a tensão que alimenta todo o circuito.

A tensão de 5V, além de alimentar todo o circuito, passa por um diodo 1N4148 no qual ocorre uma queda de tensão no valor de 0.7V resultando em uma tensão de 4,3V. Essa tensão é ligada a um outro conector KRE (J10) pelo qual é realizada a alimentação do controle *DualShock® 3*.

O circuito de alimentação pode ser visto na Figura 39.

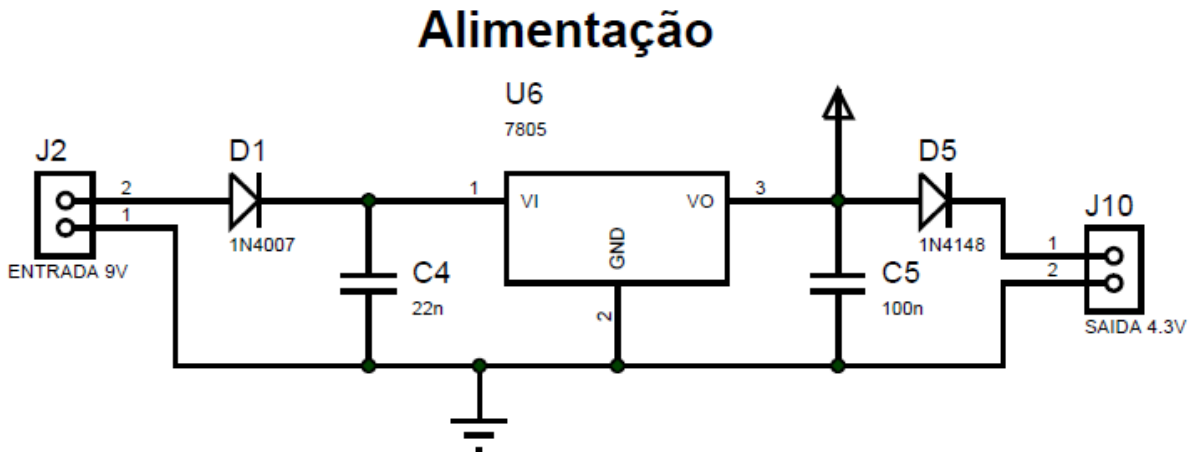


Figura 39 - Circuito de alimentação da placa.
Fonte: Autoria própria.

3.3.2 Circuito do Microcontrolador

O microcontrolador é o “cérebro” da placa. Tendo em vista que ele tem a função de gerenciar o funcionamento de todo o circuito, todas as outras partes da placa são diretamente conectadas a ele. O microcontrolador, seu circuito base e circuito de reset são mostrados na Figura 40.

Nesse circuito pode ser observada a presença do AT89S52 (U5) e do seu circuito básico de funcionamento (cristal X1 e capacitores C1 e C2). Também pode ser visto o circuito de *reset* que tem a função de reiniciar o circuito na hora da gravação via ISP ou a qualquer momento através do acionamento do botão S4.

Pode-se notar uma rede resistiva de 4k7Ω conectada diretamente ao *port* P0. Essa rede resistiva tem função de realizar *pull-up* desse *port*, já que dos 4 *ports* (P0, P1, P2 e P3) P0 é o único que não possui resistores de *pull-up* internos. Portanto, para que se garanta a presença de nível lógico alto nos pinos do *port* P0 essa rede resistiva se faz necessária.

Cada pino dos *ports* possui um sinal ligado a ele. Esses sinais são ligados nas outras partes dos circuitos, conforme poderá ser observado nas próximas seções.

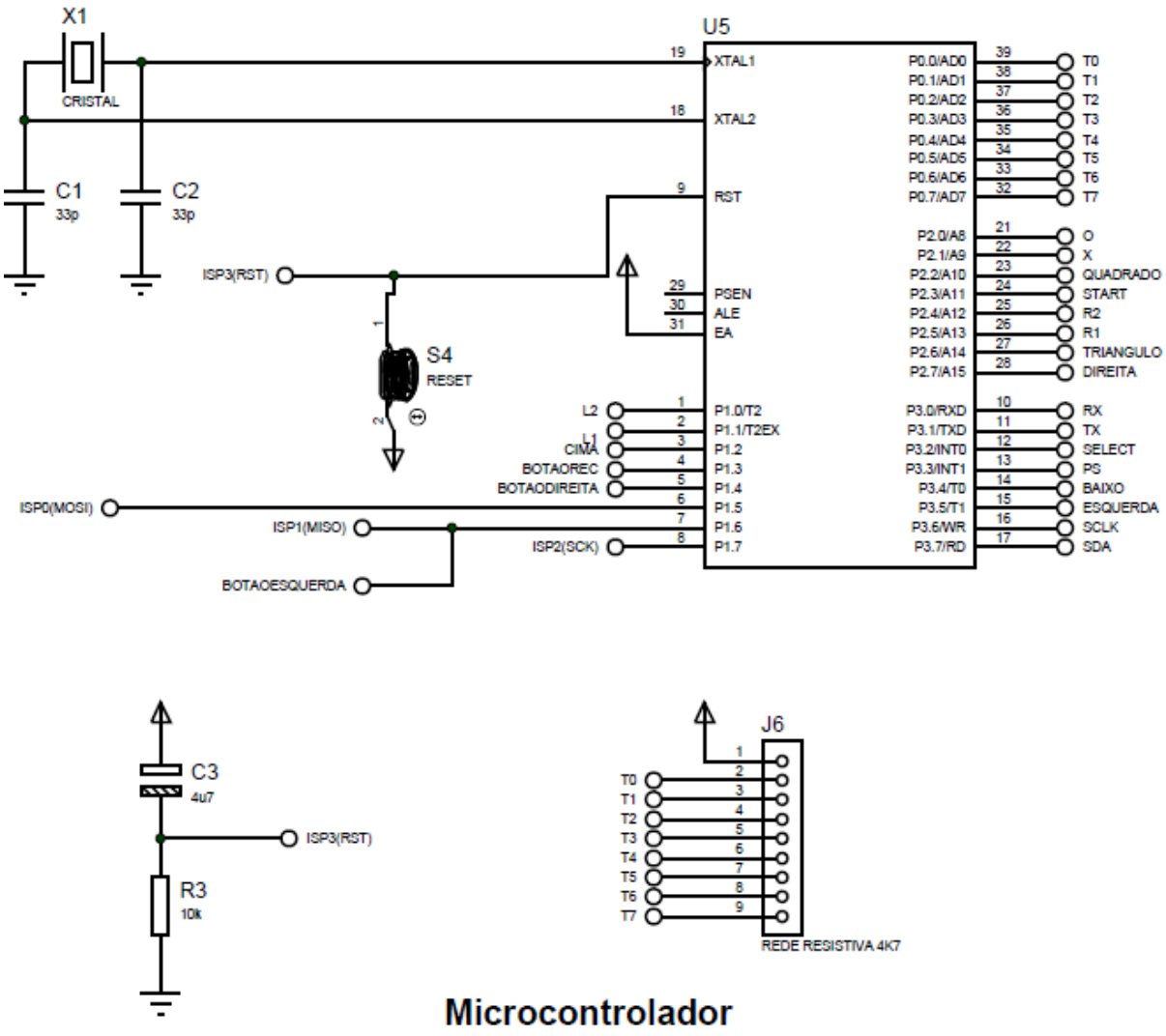


Figura 40 - Circuito do microcontrolador.
Fonte: Autoria Própria.

3.3.3 Botões da placa

Os botões da placa são acionados de maneira a fazer com que um nível lógico baixo chegue ao pino do microcontrolador que controla o botão. A placa possui três botões:

- BOTAODIREITA: Quando pressionado junto com um botão que contém um combo gravado na memória, realiza esse combo para o lado direito.

Deve ser pressionado quando o personagem do jogo está olhando para o lado direito, garantindo a execução do combo de maneira correta.

- BOTAOESQUERDA: Idem ao BOTAODIREITA, mas realiza os combos para o lado esquerdo.
- BOTAOREC: Inicia e termina a gravação de combos via *hardware*.

O circuito com os três botões é mostrado na Figura 41.

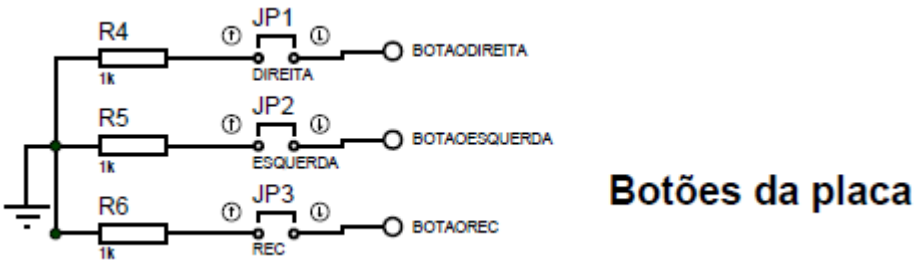


Figura 41 - Circuito com os botões da placa.

Fonte: Autoria própria.

Esses 3 botões são ligados diretamente por um fio aos correspondentes botões no controle arcade.

3.3.4 Gravação ISP

Como descrito na seção 3.1.2.1, o AT89S52 possui suporte para a gravação ISP, na qual a *flash* do microcontrolador pode ser gravada com o *chip* conectado à placa. Como essa gravação segue o protocolo SPI, para que ela ocorra é necessária a conexão dos pinos MOSI (P1.5, pino 6), MISO (P1.6, pino 7), SCK (P1.7, pino 8) e RST (*reset*, pino 9) a um conector que na hora da gravação vai ligado a um dispositivo gravador de *flash* de microcontroladores. Esse conector, no caso da placa, é uma barra de pinos macho dupla. O circuito que contém esse conector é mostrado na Figura 42.

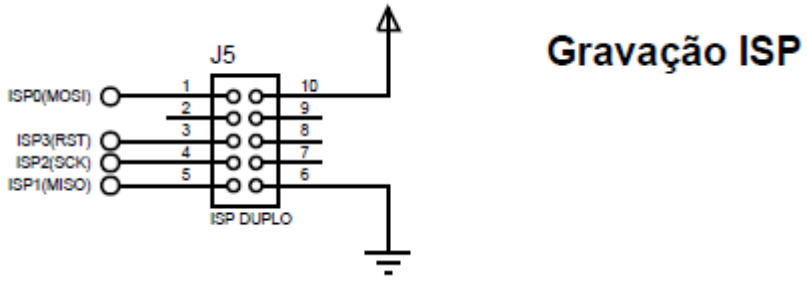


Figura 42 - Circuito de gravação ISP do microcontrolador.
Fonte: Autoria própria.

3.3.5 Conexão com o arcade

A conexão com o controle arcade é feita através de uma barra de pinos macho dupla que recebe o conector no qual estão ligados os botões do controle arcade. Essa barra de pinos é ligada ao *port* P0 do microcontrolador de maneira que cada fileira vertical vá ligada a 4 pinos consecutivos no microcontrolador (T0, T1, T2, T3 e T4, T5, T6, T7). Essa configuração facilita a implementação da leitura dos botões por varredura no *firmware*, processo o qual é mostrado no Apêndice D.

O circuito de conexão com o arcade pode ser observado na Figura 43.

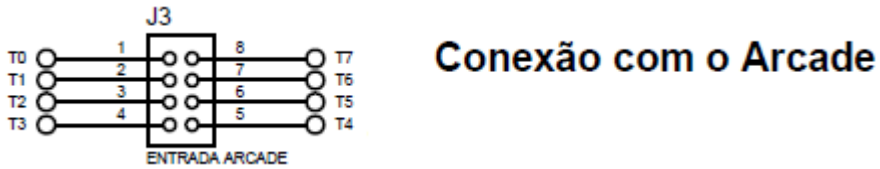


Figura 43 - Circuito de conexão com o arcade.
Fonte: Autoria própria.

3.3.6 Circuito da Memória EEPROM

Como a memória EEPROM AT24C512 utiliza o protocolo I^2C para comunicação, é necessária a ligação de apenas dois fios entre o *chip* de memória e o microcontrolador (SDA e SCL). Os outros pinos (WP, A0, A1 e A2) vão conectados diretamente ao terra da placa pois não é necessário endereçamento (visto a utilização de apenas um *chip* de memória) e não é necessária a proteção da escrita na memória.

O circuito que liga o AT24C512 ao microcontrolador (através dos pinos SCLK e SDA) é mostrado na Figura 44.

Figura 44 - Circuito de ligação da memória EEPROM.
Fonte: Autoria própria.

3.3.7 Comunicação Serial

O circuito que permite a comunicação serial RS-232 com o computador já foi descrito na seção 3.1.2.5. Àquele circuito acrescentaram-se dois LEDs de sinalização para quando ocorre recepção ou transmissão de dados e que também servem para sinalização na gravação de combos via *hardware*.

O circuito de comunicação serial pode ser visto na Figura 45.

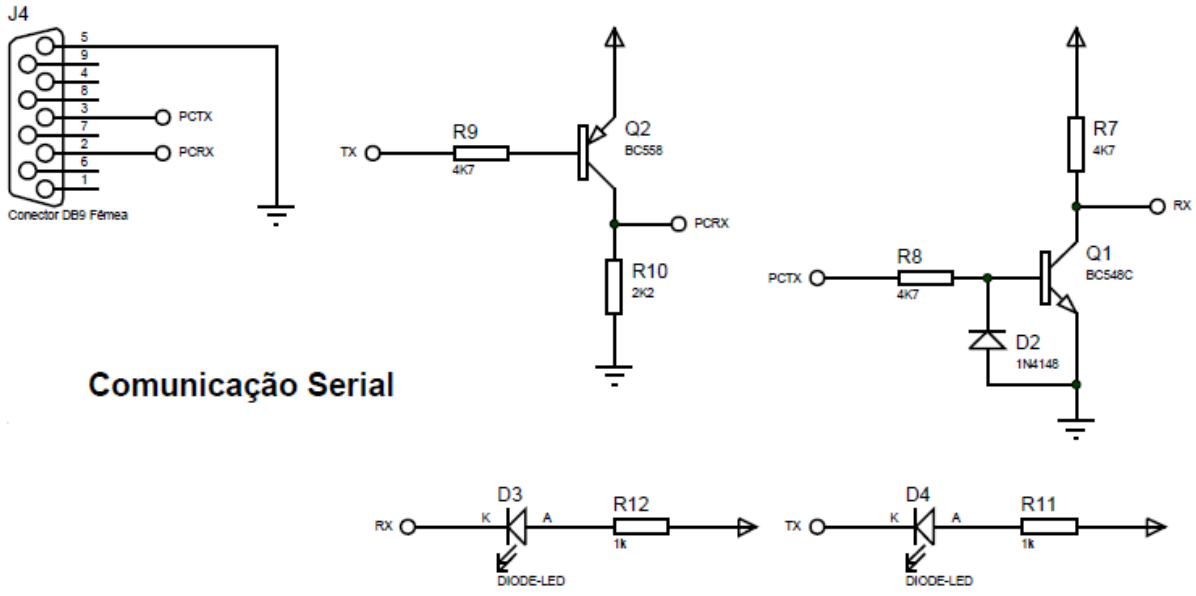


Figura 45 - Circuito de comunicação serial RS-232.
Fonte: Autoria Própria.

3.3.8 Conexão com o *DualShock® 3*

A conexão com o controle *DualShock® 3* - além da alimentação descrita na seção 3.3.1 - se dá através de uma barra de pinos fêmea que recebe um conector com fios finos soldados na placa do controle. Essa barra de pinos vai ligada às chaves CD4066 que recebem o comando do microcontrolador para serem acionadas, fechando o contato entre dois pinos. A ligação entre a barra de pinos e os CD4066 pode ser vista na Figura 46.

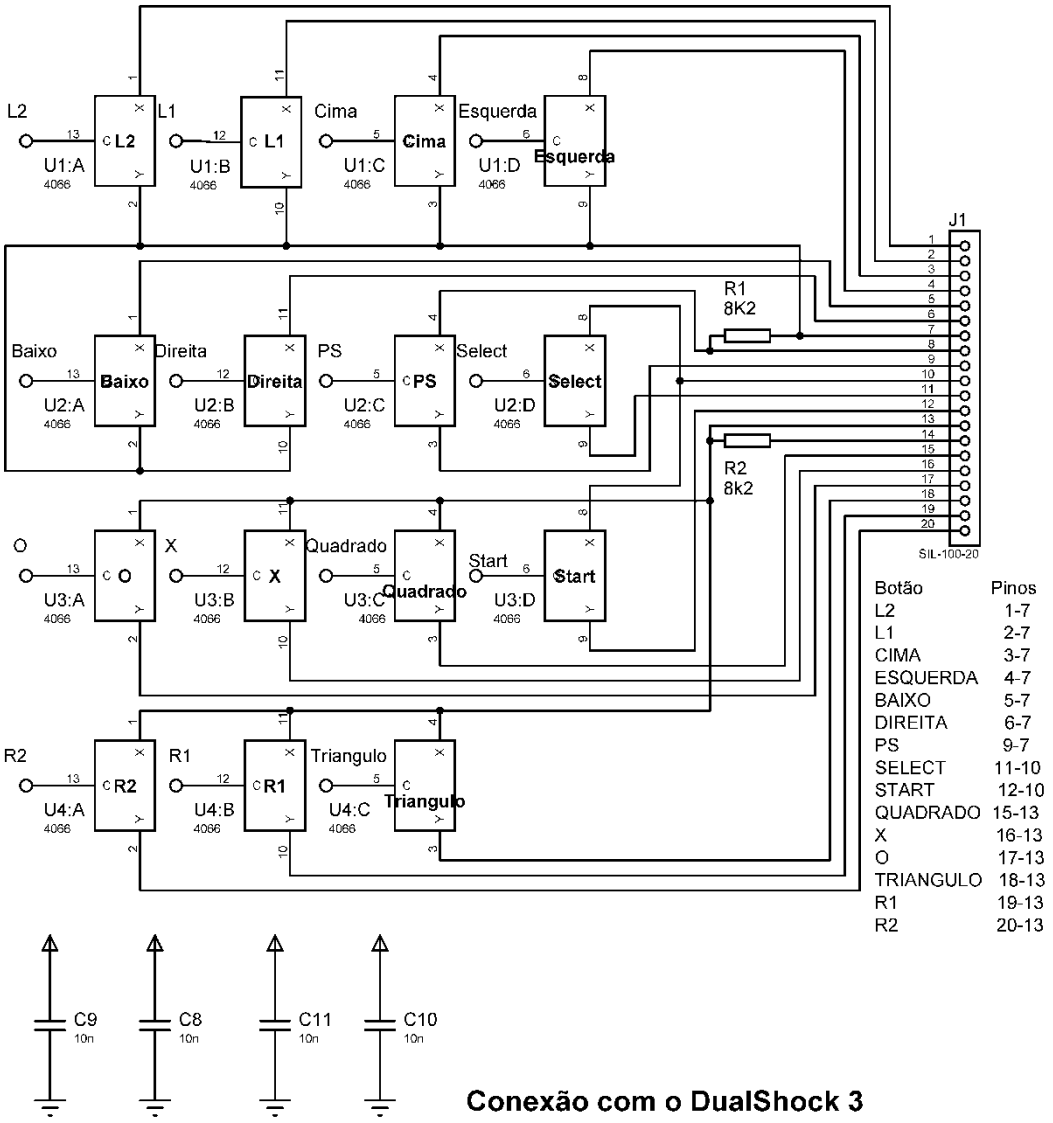


Figura 46 - Circuito de conexão com o DualShock® 3.
Fonte: Autoria Própria.

Na figura 46, se pode notar ainda a presença de dois resistores de valor 8K2Ω (R1 e R2). Esses resistores estão presentes na placa do *DualShock*® *3*, porém quando é feita a retirada dessa placa da estrutura do controle não é possível a retirada dos resistores, por isso eles foram acrescentados ao circuito. Também se pode observar a presença dos capacitores C8, C9, C10 e C11 que atuam como capacitores de desacoplamento dos CD4066. J1 é a barra de pinos fêmea.

O acionamento de um botão se dá quando os dois contatos nos quais os botões estão conectados na placa do controle do PS® 3 se fecham. Sendo assim, quando um CD4066 fecha contato entre dois pinos da barra de pinos na qual o controle está conectado, ocorre o acionamento de um botão. O controle-base do PS® 3 com o conector soldado é mostrado na Figura 47.

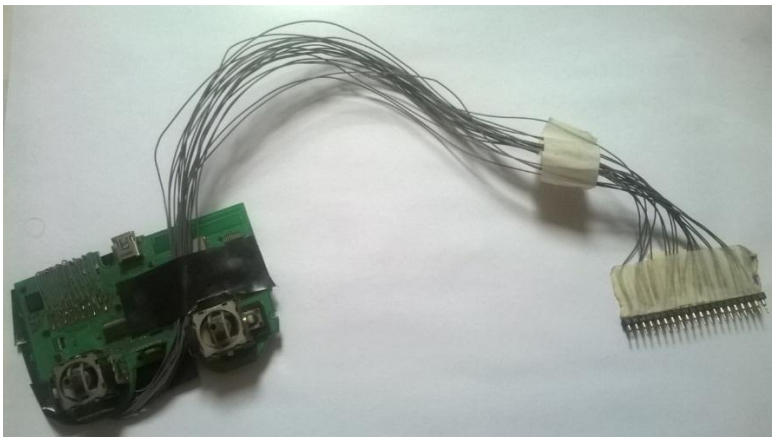


Figura 47 - Controle-Base do DualShock 3 com o conector que vai ligado à placa.
Fonte: Autoria própria.

A Figura 48 e mostra os contatos da placa do *DualShock*® 3.

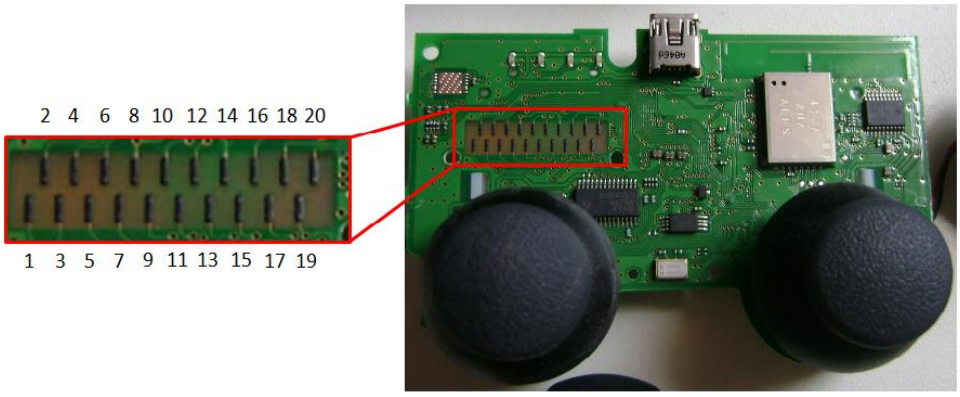


Figura 48 - Contatos da placa do controle *DualShock*® 3.
Fonte: Autoria Própria.

Após o mapeamento dos contatos chegou-se ao Quadro 6 no qual pode-se observar a quais contatos estão ligados cada botão:

Botão	Contatos	Botão	Contatos
L2	1 – 7	START	12 – 10
L1	2 – 7	□	15 – 13
UP	3 – 7	X	16 – 13
LEFT	4 – 7	O	17 – 13
DOWN	5 – 7	Δ	18 – 13
RIGHT	6 – 7	R1	19 – 13
PS	9 – 7	R2	20 – 13
SELECT	11 – 10		

Quadro 6 – Descrição dos contatos da placa base do controle *DualShock*® 3.
Fonte: Autoria Própria

No Quadro 6 pode-se perceber a ausência dos contatos 8 e 14 além da repetição dos contatos 7, 10 e 13 para mais de um botão. Nos pinos 8 e 14 são ligados os dois resistores de 8K2Ω citados anteriormente, sendo R1 ligado entre os contatos 7 e 8 e R2 ligado entre 13 e 14. Os contatos que se repetem são chamados “comuns” e o acionamento de um botão ocorre quando há o fechamento de um contato com seu respectivo comum.

3.3.9 Placa de circuito impresso

A partir do diagrama esquemático do apêndice A foi feito o roteamento da placa, resultando no *layout* mostrado no apêndice B. Com o layout foi feita a confecção da placa mostrada na Figura 49.

O tamanho total da placa foi 7,8 cm (centímetros) de largura por 10,8 cm de comprimento. Esse tamanho está de acordo com as especificações do projeto, já que a placa vai colocada dentro da caixa do controle arcade e esta comporta a presença da placa.

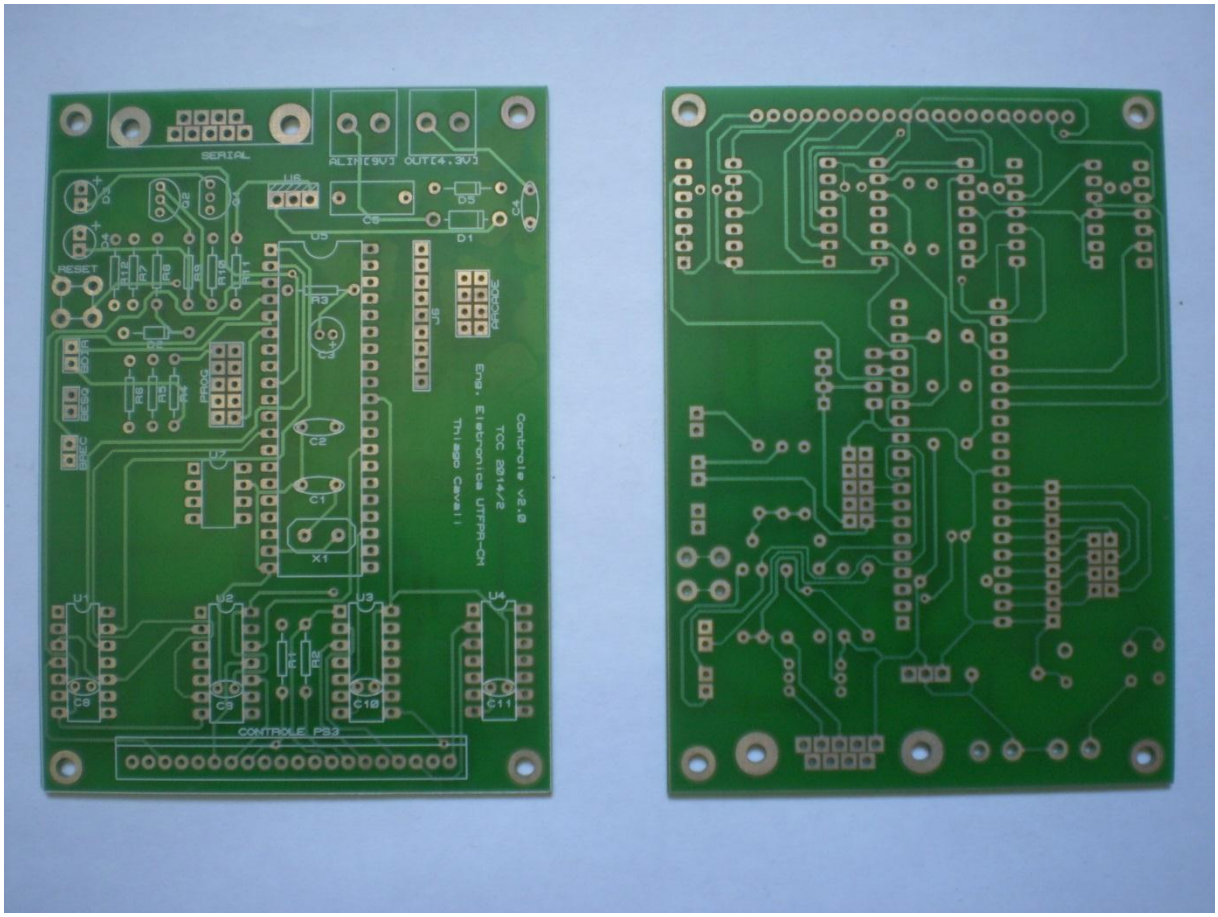


Figura 49 - Placa de circuito impresso. Da esquerda para a direita: Face superior; Face inferior. Fonte: Autoria Própria.

3.4 CONSTRUÇÃO DO CONTROLE ARCADE

A construção do controle arcade se deu em duas etapas. Primeiramente foi feita a montagem da estrutura e depois a montagem das peças. Cada uma dessas etapas será detalhada nas seções a seguir.

3.4.1 Montagem da estrutura

O material necessário para a montagem da estrutura foi o seguinte:

- Placas de madeira;
- Dobradiças;
- Parafusos;
- Trava.

Com o material em mãos foi realizado o corte das placas de madeira para a montagem da caixa que contém as placas. Com as placas de madeira do tamanho certo e devidamente furadas, foi feita a montagem da caixa e a fixação da tranca. A estrutura da caixa com a tranca e as dobradiças pode ser vista na Figura 50.

Figura 50 - Estrutura da caixa com a tranca e as dobradiças.
Fonte: Autoria própria.

Posteriormente foi feita a furação da tampa nos lugares onde foram colocados os botões e o *joystick*. A tampa com os furos é mostrada na Figura 51.

Com a caixa e a tampa pronta, as duas partes foram unidas pelas dobradiças finalizando a montagem da estrutura. Os diagramas com as dimensões da caixa e da tampa podem ser observados no Apêndice F.

Figura 51 - Tampa com os furos onde foram colocados os botões e o *joystick*.
Fonte: Autoria própria.

3.4.2 Montagem das peças

As peças necessárias para a montagem do controle foram as seguintes:

- Botões de Nylon de diferentes cores com *micro switch*;
- Um *joystick* com quatro *micro switches*;
- Fios;
- Conector fêmea duplo;

Micro switch é um tipo de chave que pode ser acionada quando uma força mecânica pequena atua sobre ela. A função dessas chaves no controle arcade é realizar o acionamento dos botões. Cada botão do arcade possui uma *micro switch* com dois terminais, sendo que o acionamento de um botão faz com que haja o fechamento do contato entre esses dois terminais dos quais saem fios que vão ligados ao conector. O *joystick* possui quatro *micro switches*, sendo uma para cada direção de movimento (baixo, cima, esquerda e direita). Um exemplo de *micro switch* conectada a um botão de *nylon* é mostrado na Figura 52.

Figura 52 - Botão de *nylon* do arcade com *micro switch*.
Fonte: Autoria própria.

Os botões do arcade devem ser colocados em conformidade com os botões que existem no controle *DualShock® 3* de maneira a permitir que o usuário jogue normalmente, além de utilizar os recursos de gravação e execução de combos. Para

tal fim, a distribuição dos botões na tampa ficou de acordo com o mostrado na Figura 53. Nessa figura, além de haver a presença dos botões do PS® 3, pode-se notar os botões de execução de combos para os dois lados (“Combo Esquerda” e “Combo Direita”) e o botão de gravação de combos via *hardware* (REC).

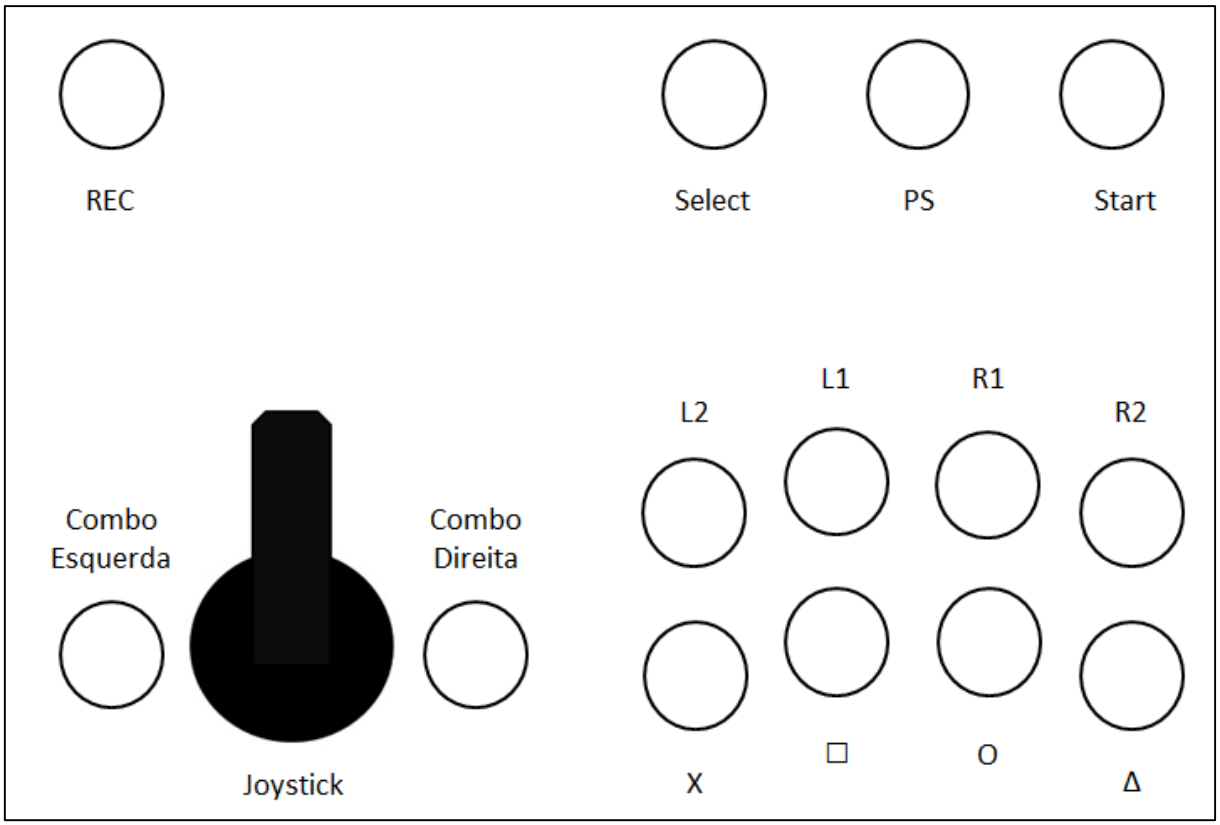


Figura 53 - Distribuição dos botões e do joystick na tampa do controle arcade.
Fonte: Autoria própria.

Com as *micro switches* ligadas aos botões de *nylon*, os botões foram fixados na tampa e realizou-se a soldagem dos fios nos terminais das *switches*. Os terminais onde cada fio foi soldado foram definidos pela forma como será feita a varredura do teclado pelo *firmware* do microcontrolador. Os fios foram ligados ao conector fêmea duplo que vai conectado na entrada do arcade na placa. A estrutura interna da tampa do controle arcade - já com os botões, o *joystick*, as *micro switches*, os fios e o conector - é mostrada na Figura 54.

A Figura 55 mostra a estrutura externa do controle arcade com os botões e o *joystick*.

Figura 54 - Estrutura interna da tampa do controle arcade.
Fonte: Autoria própria.

Figura 55 - Controle arcade.
Fonte: Autoria própria.

3.5 *SOFTWARE* DO MICROCONTROLADOR

O código-fonte do *software* implementado no microcontrolador é mostrado no Apêndice D deste trabalho. Ele executa as ações de acordo com o diagrama de blocos mostrado na Figura 56.

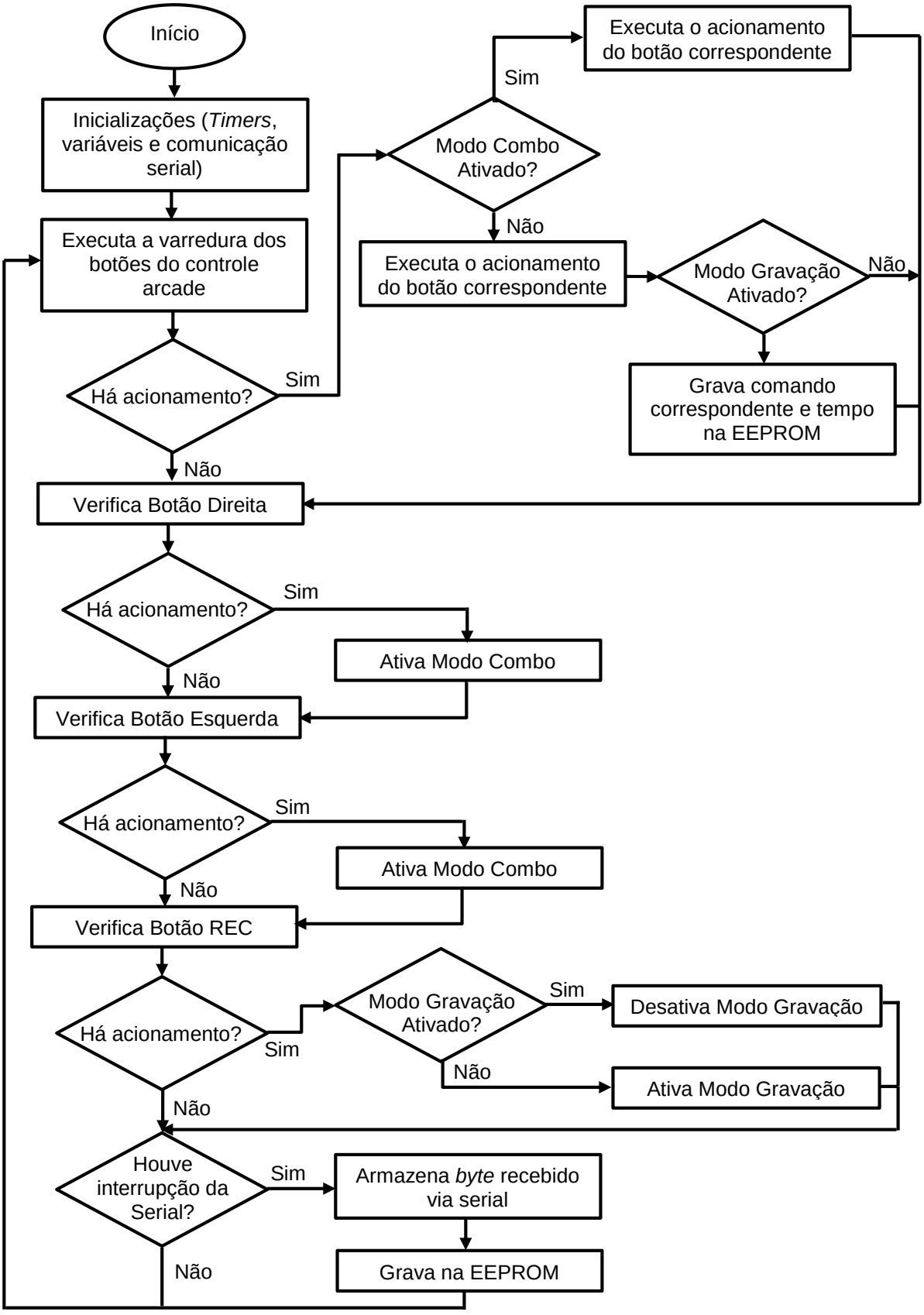


Figura 56 - Diagrama de Blocos do software do microcontrolador.

Fonte: Autoria Própria.

Nesse diagrama de blocos é possível observar que cada parte do *hardware* citado nas seções anteriores é gerenciada pelo *software* do microcontrolador, o que reforça a ideia de que o microcontrolador é o responsável pelo controle de todo o circuito.

Uma descrição de cada parte integrante do *software* do microcontrolador será apresentada nas seções a seguir. Também será descrito o protocolo utilizado para leitura, transmissão, decodificação e armazenamento dos combos na memória EEPROM.

3.5.1 *Timers*

O AT89S52, como visto na seção 3.1.2.1, possui três *timers* chamados de *Timer 0*, *Timer 1* e *Timer 2*. Foram utilizados os três *timers* na execução do projeto, cada um realizando uma tarefa diferente.

3.5.1.1 *Timer* 0

O *Timer* 0 é utilizado para geração do tempo entre acionamentos de cada botão durante o processo de gravação via *hardware*. Para que possa gerar tempo o *Timer* deve ter seus registradores (TL0 e TH0) carregados com valores específicos que variam de acordo com o tempo desejado, o modo do *timer* (determina o número de *bits* de TL0 e TH0) e com a frequência de oscilação do cristal de quartzo utilizado para geração de *clock* do sistema. O cálculo da carga pode ser feito através da seguinte relação:

$$carga\ total = 2^{número\ de\ bits} - \left(\frac{tempo\ desejado * freq.\ clock}{12}\right) \quad (1)$$

A equação (1) oferece a carga total. Esse valor deve ser convertido para hexadecimal e posteriormente atribuído aos registradores.

O número de *bits* é dado pelo modo do *timer*, que pode ser programado através da configuração do registrador TMOD. Os quatro modos e o número de *bits* de cada um são os seguintes (NICOLOSI, 2007):

- Modo 0: 13 *bits*, sendo 5 em TL e 8 em TH;
- Modo 1: 16 bits, sendo 8 em TL e 8 em TH;
- Modo 2: 8 *bits* com recarga automática, sendo que somente TH recebe a carga;
- Modo 3: 8 *bits* misto, sendo 8 em TL e 8 em TH com os contadores isolados.

No projeto foi utilizado o modo 1 para o *timer* 0 poder gerar um tempo de 1ms (mili segundo) necessário para marcar a temporização entre os acionamentos na gravação via *hardware*. A ativação do *timer* 0 se deu por meio da ativação da *flag* TR0, pertencente ao registrador TCON.

Também foi utilizada a interrupção referente ao *timer* 0 (interrupção 1) para identificar estouro na contagem (um estouro ocorre quando o valor dos registradores passa de 0xFFFF para 0x0000). Quando um estouro ocorre os registradores são recarregados para iniciar uma nova contagem e a variável que conta o tempo em ms entre os acionamentos (“tempoCom”) é incrementada, já que o estouro ocorre a cada 1 ms. A interrupção do *timer* 0 é ativada através da configuração do registrador IE.

Toda a configuração do *timer 0*, de sua interrupção e o tratamento dessa interrupção é mostrado nas funções “iniciaTimer0” e “timer0” do código mostrado no Apêndice D.

3.5.1.2 *Timer* 1

O t*imer* 1 é utilizado para a geração de *baud rate* para a comunicação serial. Segundo Nicolosi (2007), *baud rate* é definido como o número de transições de um sinal em uma linha de comunicação em um período de um segundo, em outras palavras, trata-se da frequência de comunicação que deve ser a mesma para o transmissor e para o receptor.

Para que o *timer* 1 possa gerar *baud rate* para a serial deve estar configurado em modo 2 (os modos, bem como a forma de configura-los, são comuns ao *timer* 0 e ao *timer* 1). Nesse modo a carga é feita somente no registrador TH1 que a cada estouro de contagem recarrega TL1 (registrador que realiza a contagem) com o

valor da carga. Para o cálculo da carga, a seguinte relação é utilizada (NICOLOSI, 2007):

$$carga\ total = 256 - \left(\frac{freq.clock}{384 * baud\ rate}\right) \quad (2)$$

O valor obtido da equação (2) deve ser convertido para hexadecimal e atribuído ao registrador TH1.

Para o projeto o *baud rate* escolhido foi de 9600, taxa que corresponde à frequência necessária para comunicação entre o microcontrolador e o computador. A configuração do *timer* 1 como gerador de *baud* rate, os valores da carga do *timer* para um *baud rate* de 9600 e a ativação do funcionamento do *timer* através da *flag* TR1 (pertencente ao registrador TCON) são feitos na função “serialcfg” mostrada no código apresentado no Apêndice D.

É válido ressaltar aqui, que a frequência de *clock* (dada pelo cristal de quartzo) influencia no erro obtido no cálculo da carga total mostrado na equação (2). De acordo com Nicolosi (2007) um *clock* de 12MHz pode gerar um erro de cálculo de aproximadamente 7% para a geração de um *baud rate* de 9600. Em valores absolutos, esse erro representa um *baud rate* obtido de 8923 quando o desejado é 9600. Esse erro pode ser intolerável quando se trata de comunicação serial. Esse fato explica a utilização do cristal de quartzo com frequência de 22.1184 MHz no trabalho, já que com esse *clock* o erro obtido para um *baud rate* de 9600 é nulo.

3.5.1.3 *Timer 2*

O *timer* 2 é utilizado na geração de tempo para execução dos combos e para outras aplicações diversas nas quais é necessário um tempo ocioso (*delay*). Diferentemente dos *timers* 0 e 1, o *timer* 2 possui 3 modos de operação, que podem ser configurados no registrador T2CON (ATMEL, 2008):

- 16-*bits* com recarga automática;
- 16-*bits* modo de captura;
- Gerador de *baud rate*.

O *timer* 2 permite ainda contagem de maneira crescente ou decrescente, modo escolhido pela configuração do registrador T2MOD.

No projeto o *timer* 2 é utilizado no modo de 16-*bits* com recarga automática realizando contagem crescente para poder gerar um tempo de 1 ms. Nesse modo é necessário que a carga seja atribuída aos registradores RCAP2L e RCAP2H para que estes recarreguem os registradores que realizam a contagem (TL2 e TH2, respectivamente) a cada estouro, permitindo o início de uma nova contagem.

Devido a um erro de 0,0109% na geração de ms, a cada 20 ms os registradores RCAP2L e RCAP2H tem seu valor alterado de modo a compensar esse erro na contagem. Esse ajuste da contagem do *timer* 2 pode ser observado no código do Apêndice D.

Para detecção de estouro na contagem do *timer* 2 utilizou-se a interrupção deste (interrupção 5), que tal qual as interrupções dos *timers* 0 e 1 precisa ser ativada no registrador IE. Quando essa interrupção ocorre a *flag* TF2 é setada, tendo de ser limpa na rotina de tratamento da interrupção que também incrementa o valor da variável que gera tempo em “ms” para execução dos combos (“miliseg”), pois o estouro ocorre a cada 1 ms.

A configuração do *timer* 2 e de sua interrupção, bem como o tratamento dessa interrupção é mostrada nas funções “iniciaTimer2” e “timer2” presentes no código do Apêndice D.

3.5.2 Comunicação serial e gerenciamento de interrupções

Como citado na seção 3.1.2.5 o AT89S52 possui suporte para a comunicação serial em 4 modos, sendo o modo 1 utilizado no projeto (comunicação assíncrona com 1 *start bit*, 8 *bits* de dados e 1 *stop bit*) com um *baud rate* de 9600 gerado pelo *timer* 1.

Para a utilização da serial deve-se configurar o registrador SCON que determina o modo de operação e habilita o funcionamento da comunicação. Também se deve habilitar a interrupção da serial (interrupção 4) que faz com que o microcontrolador seja interrompido quando há recepção de dados. A interrupção da serial, tal qual dos *timers* pode ser configurada no registrador IE.

Com a utilização de várias interrupções diferentes no projeto, é necessária a definição de diferentes níveis de prioridade para elas, apesar de o microcontrolador

possuir uma prioridade natural relativa para tratamento das interrupções, como mostrado a seguir (NICOLOSI; BRONZERI, 2008):

- 1ª prioridade Natural Relativa: Interrupção externa 0;
- 2ª prioridade Natural Relativa: Interrupção do *timer* 0;
- 3ª prioridade Natural Relativa: Interrupção externa 1;
- 4ª prioridade Natural Relativa: Interrupção do *timer* 1;
- 5ª prioridade Natural Relativa: Interrupção da comunicação serial;
- 6ª prioridade Natural Relativa: Interrupção do *timer* 2;

Várias interrupções foram utilizadas no projeto, sendo a comunicação serial a mais importante delas para que não haja o risco de falha na recepção e transmissão dos dados relativos aos combos. Para que fosse definida prioridade alta para a interrupção da serial foi configurado o registrador IP, conforme mostrado no Apêndice D.

Na comunicação serial há um *buffer* que recebe os dados transmitidos pelo computador e um que armazena os dados a serem transmitidos. Esses *buffers* podem ser acessados utilizando o registrador SBUF.

O processo de recepção de dados ocorre quando acontece uma interrupção da serial, então o microcontrolador seta a *flag* RI que indica que o *buffer* de recepção está cheio, pode-se então coletar os dados armazenados em SBUF e limpar a *flag* RI para que comece uma nova recepção de dados.

O processo de transmissão, por sua vez, ocorre da seguinte maneira: atribui-se o dado que se deseja transmitir para o registrador SBUF, então se espera até que a transmissão do dado seja concluída através da observação do estado da *flag* TI que quando é ligada indica que a transmissão foi concluída e o *buffer* de transmissão está vazio e pronto para transmitir um próximo dado.

A configuração da serial, de sua interrupção e os processos de transmissão e recepção de dados podem ser vistos nas funções “serialcfg”, “serialRX” e “serialTX”, presentes no código do apêndice D.

3.5.3 Protocolo de leitura e escrita de combos

Antes que se inicie a descrição das outras partes do código-fonte do microcontrolador é necessário que se descreva o protocolo utilizado para leitura, transmissão, decodificação e gravação na memória EEPROM dos combos.

Cada ação do jogo possui um botão específico que a executa, sendo que essa ação se inicia quando um botão é pressionado e é interrompida quando o jogador solta esse botão. Sendo assim é válido dizer que a realização de uma ação pode ser descrita por “Botão acionado – Tempo de acionamento – Botão solto”.

O protocolo implementado se baseia nessa sequência de ações que definem a realização de um comando. A estrutura do protocolo pode ser vista na Figura 57.

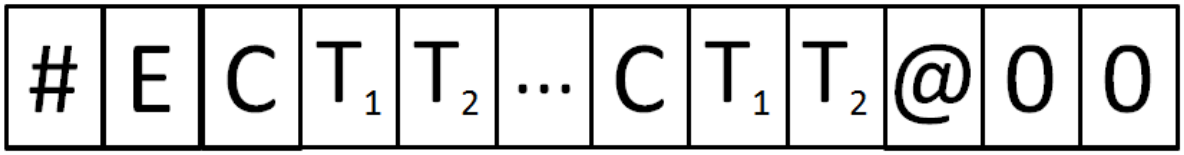


Figura 57 - Protocolo de gravação de combos.
Fonte: Autoria Própria.

Na Figura 57, cada quadrado corresponde a um caracter do protocolo que tem tamanho de um *byte*. O primeiro *byte* observado é o caracter “#” que é o cabeçalho do protocolo, o segundo *byte* (representado por E) indica o endereço do combo na memória EEPROM. Pode ser usado para:

- No caso de uma operação de gravação de combo via *software*, indicar o endereço da EEPROM no qual o combo será gravado. Cada botão acessa em um endereço específico um combo de até 1000 *bytes* na EEPROM, então esse *byte* de endereço mostra qual o botão que irá acessar o combo que está sendo gravado na memória.
- No caso de uma operação de leitura do combo pelo *software* computacional, indicar qual a posição da memória que contém o combo que será enviado para o computador.

O terceiro caracter (representado por C) é referente a uma ação do jogo que o usuário deseja realizar, vale lembrar que cada botão do controle executa uma ação diferente do jogo. O Quadro 7 mostra os botões, a ação realizada por cada um deles

(tendo como base o jogo *Super Street Fighter® 4* da *Capcom®*), o endereço da EEPROM acessado por cada um e o caracter do protocolo correspondente ao botão.

Botão	Caracter	Endereço da EEPROM	Ação do jogo
UP	U	0 – 999	Pulo
LEFT	L	1000 – 1999	Movimento para esquerda
DOWN	D	2000 – 2999	Agachar
RIGHT	R	3000 – 3999	Movimento para direita
X	X	4000 – 4999	Chute fraco (LK)
O	O	5000 – 5999	Chute médio (MK)
QUADRADO	Q	6000 – 6999	Soco fraco (LP)
TRIÂNGULO	T	7000 – 7999	Soco médio (MP)
R1	F	8000 – 8999	Soco forte (HP)
L1	E	9000 – 9999	Soco triplo (TP)
R2	H	10000 – 10999	Chute forte (HK)
L2	G	11000 – 11999	Chute Triplo (TP)
PS	P	12000 – 12999	Botão PS
SELECT	S	13000 – 13999	Varia com o modo de jogo*
START	I	14000 – 14999	Pausa o jogo

*De acordo com o modo de jogo o botão select varia de função, no modo de *trial* por exemplo, mostra os comandos a serem realizados.

Quadro 7 – Botões do controle, caracteres correspondentes a eles no protocolo, endereços da EEPROM que acessam e ação do jogo que realizam.
Fonte: Autoria Própria.

Quando há dois caracteres de comando iguais, o primeiro representa o acionamento do botão referente aquele comando e o segundo representa que a execução do comando deve ser interrompida, o que equivale ao botão ser solto.

O quarto e quinto caracteres do protocolo representam o tempo em mili segundos durante o qual o comando dado no caracter anterior será executado. Se divide em dois (T_1 e T_2) para que se consiga acessar tempos maiores, pois com dois caracteres há dois *bytes* (que representam 16 bits) que permitem a representação de 2^{16} ms, ou seja, 65536 ms que correspondem a um tempo superior a 1 minuto,

tempo suficiente para o acionamento de um comando. Caso fosse utilizado apenas 1 caracter para representação do tempo em “ms”, haveria apenas 1 *byte* (que representa 8 bits) que corresponderia a 2^8 ms, ou seja, 256 ms que é um tempo muito pequeno para o acionamento.

Para realizar essa divisão de um tempo em dois caracteres foi utilizado um recurso da linguagem C chamado *union*. Com esse recurso é possível acessar um mesmo espaço na memória com diferentes tipos de variáveis. No caso, o *union* foi utilizado para acessar o tempo entre os acionamentos tanto como um valor inteiro (tipo *unsigned short int* que possui 16 bits) quanto como dois caracteres (*array* de *unsigned char* de 2 posições, cada uma com 8 *bits*). O uso dessa estrutura pode ser observado no Apêndice D.

A sequência “C T_1 T_2”, se repete até que o usuário encerre o combo que deseja realizar. Quando termina, é acrescentado um caracter “@” que indica o fim do protocolo e dois tempos de valor zero para manter o formato “C T_1 T_2” evitando erros na identificação do fim do protocolo.

Um exemplo simples é o seguinte: no caso da recepção de um conjunto de dados pela serial representando um combo feito via *software* e esse conjunto de dados seja “# L D T_{11} T_{21} X T_{12} T_{22} D T_{13} T_{23} H T_{14} T_{24} @ 0 0”, o programa:

- Identifica o início da sequência pela leitura do caracter “#”;
- Devido a presença do caracter “L”, armazena o combo a partir da posição 1000 da EEPROM, que é acessada pelo botão LEFT.

As ações do jogo realizadas quando esse combo for decodificado (processo descrito na seção 3.5.5) serão as seguintes:

- O personagem se agacha (comando “D”);
- Após o tempo indicado por T_{11} e T_{21}, ainda agachado o personagem dá um chute fraco (comando “X”);
- Após o tempo indicado por T_{12} e T_{22}, o personagem se levanta (segundo caracter “D” que indica fim da realização do comando);
- Após o tempo indicado por T_{13} e T_{23}, o personagem dá um chute forte (comando “H”);
- Após o tempo indicado por T_{14} e T_{24} se encerra a execução do combo (caracter “@”).

3.5.4 Leitura dos botões do controle arcade por varredura

A leitura dos botões do controle arcade é uma das principais partes do projeto já que através dela são realizadas as ações do jogo, a execução dos combos gravados na memória e a gravação de combos via *hardware*. Para a leitura dos botões utilizou-se a técnica de leitura por varredura, na qual os botões são distribuídos de forma matricial (linhas e colunas) e lidos de acordo com o nível lógico de cada linha e coluna.

Uma das maiores vantagens dessa técnica é a economia de pinos do microcontrolador tendo em vista que na leitura por varredura não é necessário um pino para cada botão. Nesse projeto, por exemplo, é realizada a leitura de 15 botões através da utilização de 8 pinos (*port* P0) do AT89S52. A distribuição matricial dos botões do controle arcade pode ser vista na Figura 58.

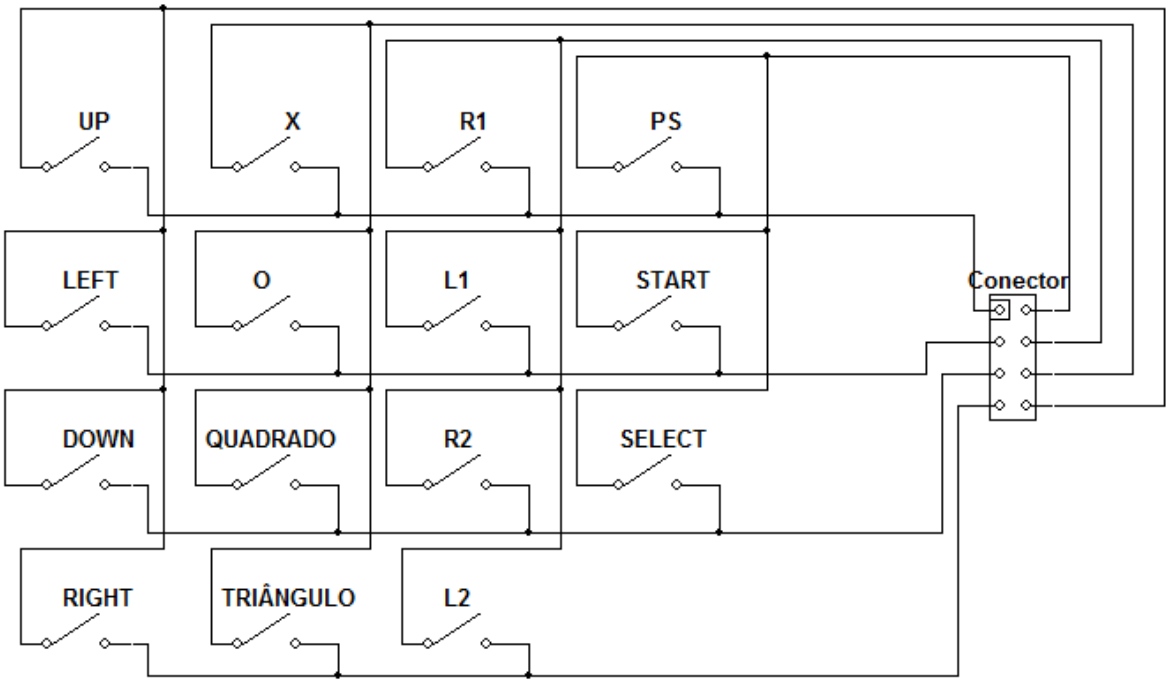


Figura 58 - Distribuição matricial dos botões no controle arcade.
Fonte: Autoria própria.

A varredura se dá através do “desligamento” de uma coluna pela colocação de um *bit* 0 na posição correspondente a ela no *port* P0. Quando essa coluna é desligada e um botão presente nela é pressionado, o *bit* 0 da coluna é transmitido para a linha na qual esse botão se encontra sendo possível a identificação do botão pressionado. Na varredura dos botões do controle arcade foram utilizadas máscaras

para leitura dos botões, conforme é possível observar no Apêndice D. Foram no total 15 máscaras utilizadas para leitura de forma que cada botão possui uma máscara própria, permitindo a leitura de mais de um botão simultaneamente, o que é necessário para o jogo. Vale ressaltar que a varredura não abrange os botões Combo Direita, Combo Esquerda e REC, que possuem um pino específico para serem lidos pelo microcontrolador.

Quando um acionamento de um botão do controle é identificado existem 3 diferentes ações a serem tomadas, execução de ação do jogo, gravação do botão acionado e do tempo na EEPROM ou realização de um combo. Essas 3 ações serão detalhadas nas subseções a seguir e seus códigos bem como os códigos da leitura do controle por varredura são vistos na função “main” do programa mostrado no Apêndice D.

3.5.4.1 Execução de ação do jogo

Como descrito na seção 3.5.3, cada botão do controle lido por varredura corresponde a uma ação no jogo, portanto quando é identificado o acionamento de algum botão alguma ação específica do jogo deve acontecer. Para que essa ação aconteça, quando um botão é pressionado é atribuído nível lógico 1 para o pino do microcontrolador que contém o comando de acionamento da chave CD4066 correspondente ao botão pressionado. Isso faz com que ocorra o fechamento da chave e consequente fechamento dos contatos correspondentes ao botão pressionado no controle *DualShock® 3* executando assim a ação desejada.

3.5.4.2 Gravação do botão acionado

Quando o modo de gravação está ativado (esse modo será detalhado na seção 3.5.6) a cada vez que um botão é acionado ou solto, além de a ação correspondente ao botão ter de ser realizada na tela de jogo, o comando referente ao botão deve ser gravado na memória EEPROM, de acordo com o protocolo descrito na seção 3.5.3.

Para o armazenamento desses comandos o caracter correspondente ao botão pressionado é colocado em uma variável (“comandoAnt”). Para o tempo transcorrido entre cada comando ser armazenado, cada vez que um novo comando

ocorre, uma variável (“tempoComFinal”) armazena o valor de tempoCom – variável que é incrementada pelo *timer* 0 a cada 1 ms – e subtrai esse tempo do tempo armazenado em outra variável (“tempoComInicial”) que grava o instante no qual se iniciou o comando anterior. O resultado dessa subtração é o tempo em “ms” transcorrido desde o acionamento anterior até o acionamento atual, sendo então convertido para dois caracteres com o auxílio do *union* e gravados na memória EEPROM juntamente com o caracter armazenado em comandoAnt. Após a gravação, comandoAnt recebe o caracter referente ao último acionamento e tempoComInicial recebe o valor armazenado em tempoComFinal para obter o tempo no qual o último acionamento ocorreu. Para resumir, pode-se dizer que a cada mudança de estado em um botão (pressionado ou solto) o tempo e comando anteriores são gravados na memória, processo que se repete até que o modo gravação seja desativado.

Para se identificar se o botão foi solto, são utilizadas variáveis que armazenam o estado anterior dos botões. Com a comparação do estado atual (dado pela leitura por varredura) e do estado anterior, decide-se a ação a ser tomada. Os estados possíveis e as ações a serem tomadas são descritas no Quadro 8.

Estado Anterior	**Estado Atual**	**Ação**
Pressionado	Pressionado	Executa ação referente ao botão
Pressionado	Solto	Grava comando e tempo anteriores na EEPROM, seta comando novo para a variável “comandoAnt” e zera variável “tempoCom”.
Solto	Pressionado	Grava comando e tempo anteriores na EEPROM, seta comando novo para a variável “comando Ant” e zera variável “tempoCom”. Realiza também outras ações referentes a gravação de combos via *hardware* mostradas na seção 3.5.6
Solto	Solto	Nenhuma ação é realizada

Quadro 8 – Estados possíveis para um botão e ações tomadas de acordo com cada estado. Fonte: Autoria própria.

3.5.4.3 Realização de um combo

Quando o Modo Combo está ativado (a ativação desse modo será descrita na seção 3.5.5) o botão do controle que foi acionado deve decodificar o combo armazenado no endereço da memória EEPROM que esse botão acessa. Vale lembrar que cada botão acessa uma posição de memória diferente, conforme descrito no Quadro 7 da seção 3.5.3.

A decodificação de um combo é feita na função “decodComboGravado” presente no código-fonte do apêndice D. Essa função é chamada quando um botão do controle é pressionado e possui como parâmetros o endereço da EEPROM que aquele botão acessa e o lado para o qual o combo será executado.

A função realiza a leitura de 3 caracteres consecutivos da memória EEPROM (“C T_1 T_2”, conforme protocolo), sendo o primeiro de comando e os outros dois de tempo. Após a leitura desses 3 caracteres, o caracter de comando é traduzido no comando que representa (vide Quadro 7, seção 3.5.3) e os caracteres de tempo são lidos como um único número (utilizando o *union*) e passados como parâmetro para a função “delayms” que gera esse tempo em ms.

A tradução do caracter de comando se dá de maneira a identificar o botão relacionado a esse comando e acionar ou desligar o pino do microcontrolador no qual está ligado o pino de controle da chave CD4066 referente ao botão, executando assim a ação na tela de jogo. O pino é ligado ou desligado de acordo com a identificação se o comando é de acionamento ou desligamento, o que se dá com a utilização de *flags* de sinalização (uma para cada comando) que mostram se anteriormente ocorreu um acionamento referente ao comando dado ou não.

A decodificação dos combos se encerra quando os 3 caracterer lidos são “@ 0 0” que indicam fim do protocolo.

3.5.5 Leitura dos botões Esquerda e Direita (Modo Combo)

Os botões Esquerda e Direita indicam o Modo Combo. Como já dito em seções anteriores, cada botão do controle acessa uma posição diferente na memória EEPROM e quando o Modo Combo está ativado os combos armazenados nessas posições são executados.

A ligação do Modo Combo ocorre quando há o acionamento do botão “Esquerda” ou do botão “Direita”, e o desligamento desse modo quando o botão é solto e estava apertado. Tal qual na leitura por varredura, a verificação de botão solto é realizada através da comparação entre o estado atual e o estado anterior.

Para que um combo seja realizado é necessário que se pressione o botão que acessa a posição da EEPROM referente ao combo desejado, concomitantemente com o Botão Esquerda ou com o Botão Direita. Caso o Botão Esquerda seja pressionado é ativada a *flag* “comboEsq” e caso o Botão Direita seja pressionado é ativada a *flag* “comboDir”. Essas *flags* indicam para qual lado o combo será executado.

O que define o lado para qual o combo deve ser executado é a posição que o personagem do jogador se encontra em relação ao personagem adversário. Caso o adversário esteja a esquerda do jogador, o combo deve ser realizado para a esquerda, caso esteja a direita do jogador, o combo deve ser realizado para a direita.

3.5.6 Leitura do botão REC (Modo de Gravação)

O Modo de Gravação é ativado e desativado pelo botão REC e permite ao usuário a gravação de combos via *hardware*, ou seja, o usuário realiza o combo no controle e esse combo é armazenado em alguma posição de memória escolhida pelo jogador, havendo a possibilidade desse combo ser visto posteriormente no *software* computacional.

O início da gravação de um combo via *hardware* ocorre quando o botão REC é pressionado. Esse acionamento faz com que um dos LEDs do controle arcade se acenda e um *flag* de sinalização (“flagIniciaGrav”) seja ligada indicando que o endereço de gravação pode ser selecionado. A seleção de endereço para gravação é feita quando o usuário aciona o botão do controle referente à posição de memória que vai armazenar o combo a ser gravado, sendo que após esse acionamento a *flag* de sinalização de gravação (“flagGravHw”) é ligada e o outro LED da placa é aceso de maneira a indicar que o combo pode começar a ser gravado, além disso é ligada uma *flag* (“flagPrimComando”) que indica que o próximo acionamento é o primeiro comando do combo.

No ato do acionamento do primeiro botão do combo que o usuário deseja gravar, o caracter referente ao comando desse botão é armazenado em uma variável (“comandoAnt”, como visto na seção 3.5.4.2) e uma variável que realiza a contagem do tempo entre cada comando (“tempoCom”, como visto na seção 3.5.4.2) armazena o valor atual da variável tempoCom que é incrementada a cada 1ms pelo *timer* 0, iniciando a contagem de tempo. Além disso, a *flag* “flagPrimComando” é desativada, fazendo com que a partir do acionamento do segundo botão o processo de gravação mostrado na seção 3.5.4.2 seja realizado.

O fim da gravação de um combo via *hardware* e o desligamento do Modo de Gravação se dá quando o Botão REC é pressionado. Quando esse acionamento ocorre o comando armazenado em “comandoAnt” é gravado na memória juntamente com um tempo de 25ms, além de um caracter “@” com dois tempos 0 serem gravados para que se mantenha o protocolo. Após essas gravações a *flag* “flagGravHw” é desligada e os LEDs são apagados indicando fim da gravação.

3.5.7 Recepção de combos pela serial

Quando ocorre a interrupção da serial é ligado um *flag* (“flagrx”) que identifica que um dado foi recebido pela serial. A cada iteração do código essa *flag* é verificada e caso esteja ligada são realizadas as ações devidas de acordo com o caracter recebido (sempre seguindo o protocolo descrito na seção 3.5.3). Esse processo consiste na gravação de combos via *software*.

O primeiro caracter recebido na gravação de um combo via *software* é o caracter “#” que indica início de protocolo. Quando esse caracter é identificado como sendo o primeiro comando vindo da serial, é ligada uma *flag* que indica que os próximos caracteres não são mais de controle de protocolo, já sendo pertencentes ao combo (“flagLeitura”) e outra *flag* indicando que o próximo caracter vai dar o endereço no qual o combo será gravado (“flagEndereco”). Também são atribuídos valores para as variáveis de controle de gravação e é enviado para o *software* via serial um caracter “1” de confirmação de recebimento do dado.

O segundo caracter indica o endereço da EEPROM que irá armazenar o combo. Quando essa seleção de endereço ocorre, uma variável (“endereco”) armazena o endereço no qual os combos serão gravados, a “flagEndereco” é limpa e é enviado um caracter “2” de confirmação para o *software*. Os próximos caracteres

são comandos e tempos do combo, sendo que a cada 3 caracteres lidos (“C T_1 T_2”) uma informação “OK!” é enviada para o *software*. Quando o comando lido é um caracter “@”, a “flagLeitura” é limpa indicando fim da recepção e a informação “@OK!@” é enviada para o *software*.

A única exceção ocorre quando o primeiro comando recebido após o “#” e o indicador de endereço é um caracter “Z”. Isso indica que o combo gravado na posição dada pela variável “endereco” deve ser transmitido para o *software* computacional pela serial. Isso é feito através da função “enviaCombo” que tem como parâmetro o endereço no qual o combo que se deseja enviar está armazenado. Essa função envia os caracteres do combo 3 a 3 até que identifique a presença do caracter “@”.

O protocolo de leitura e escrita na EEPROM é mostrado no apêndice C.

3.6 *SOFTWARE* COMPUTACIONAL

O código-fonte do *software* computacional desenvolvido é mostrado no apêndice E. Tem como propósito ser uma interface entre a placa e o usuário no computador. Através dele é possível programar combos e descarregá-los no controle arcade via comunicação serial, observar na tela do computador um combo que esteja gravado no controle e descarregar para o controle combos previamente implementados. A tela principal do *software* é mostrada na Figura 59.

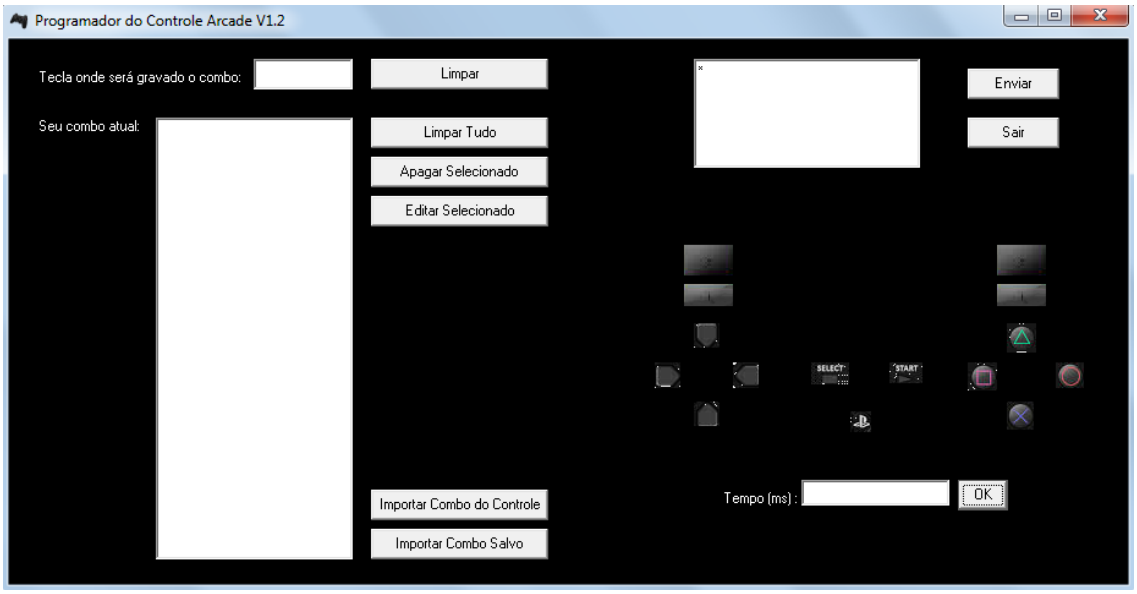


Figura 59 - Tela Inicial do software de programação do controle arcade.

Fonte: Autoria Própria.

Uma característica dessa interface é que ela é programada através da configuração dos eventos de cada elemento presente no formulário (botões, painéis, *listbox*, entre outros). Quando um evento ocorre, por exemplo, um botão é clicado, alguma ação relativa ao clique desse botão deve ser feita. Essa ação é configurada dentro de um evento (no caso “onClick”) e durante a execução do programa é realizada quando o botão é clicado.

O evento de criação (“onCreate”) do formulário que consiste na janela principal do programa ativa o serviço de multitarefas do sistema operacional *Windows* para a recepção de dados pela porta serial. Esse serviço roda em uma unidade do programa (“Unit2”) enquanto os recursos de interação com o usuário, tais como telas, botões, quadros, entre outros se encontram em outra unidade (“Unit1”).

A comunicação serial se dá através do uso da porta COM1 do computador. para que se inicie o programa é necessário que um cabo adaptador Serial-USB esteja conectado ao computador e configurado para utilizar a porta COM1. Caso essa configuração não seja identificada, a inicialização do programa é imediatamente abortada.

As características do *software* citadas nessa seção e as funcionalidades do programa são mostradas e descritas nas subseções a seguir.

3.6.1 Comunicação serial da placa com o software computacional

3.6.1.1 Configuração dos parâmetros de comunicação

No início da execução do programa é iniciado o serviço que permite a comunicação serial. Esse serviço utiliza uma variável do tipo *Handle* (“hComm”) que armazena as informações de configuração da porta serial. Essa variável é configurada através da função “CreateFile” que define os parâmetros de abertura de uma porta para comunicação serial tais como: a porta que será utilizada na comunicação, o tipo de acesso permitido a essa porta, se a porta é compartilhada, entre outros. No programa em questão é utilizada a porta COM1 que possui permissão de leitura e escrita genéricas (parâmetro “GENERIC_READ |

GENERIC_WRITE"). Se a porta não pôde ser aberta e configurada com sucesso, a aplicação é automaticamente fechada.

A segunda parte da configuração da porta serial é a configuração de *timeouts* que são os intervalos de tempo presentes no envio e recepção de dados. Essa configuração é feita através do ajuste dos seguintes parâmetros contidos em variáveis do tipo "COMMTIMEOUTS":

- "ReadTotalTimeoutMultiplier": É o multiplicador usado para definir o *timeout* das operações de leitura em ms. A cada operação de leitura o valor desse parâmetro é multiplicado pelo número de bytes a serem lidos.
- "ReadTotalTimeoutConstant": O valor constante desse parâmetro é somado ao resultado da multiplicação do "ReadTotalTimeoutMultiplier" pelo número de bytes a serem lidos, de forma a dar o *timeout* total de leitura;
- "WriteTotalTimeoutMultiplier";
- "WriteTotalTimeoutConstant";

Os dois últimos parâmetros tem a mesma função dos dois primeiros, mas para operações de escrita na serial.

A última parte da configuração da porta serial é feita com atribuição dos valores presentes em uma variável do tipo DCB. Essa variável define o valor do *baud rate*, presença de bit de paridade, número de *bits* enviados e número de *stop bits.* No projeto esses valores foram atribuídos através do uso da função "BuildCommDCB" e tem as mesmas configurações de comunicação implementadas no microcontrolador, ou seja, *baud rate* de 9600, 8 *bits* enviados, não há a presença de *bit* de paridade e há a geração de 1 *stop bit*.

A configuração desses parâmetros pode ser observada na definição do evento de criação do Form1 ("TForm1::FormCreate") na Unit1.cpp cujo código é mostrado no Apêndice E.

A porta serial é fechada quando o programa é encerrado. A função que realiza esse fechamento é a função "CloseHandle" que espera o programa armazenar a configuração dos *timeouts* e encerra a comunicação. O processo de fechamento da porta serial pode ser observado no evento de fechamento do Form1 ("TForm1::FormClose") na Unit1.cpp cujo código é mostrado no Apêndice E.

3.6.1.2 Serviço de multitarefas do *Windows* (*multithreading*)

Para que a recepção de dados pela serial ocorra enquanto o programa principal funciona, é utilizado o serviço de multitarefas do *Windows* (*Multithreading*). Com a utilização desse recurso duas ou mais partes (*threads*) de um programa são executadas um pouco por vez pelo sistema operacional, o que causa a impressão de os códigos estarem sendo executados simultaneamente. O início desse serviço acontece logo após a configuração dos parâmetros da comunicação serial e é feito através da instância da classe “TRead” que implementa os métodos da classe “TThread”. A utilização dessa classe oferece todos os recursos para a execução de multitarefas.

A classe TRead é criada no arquivo Unit2.h e possui a declaração das funções que implementa, no caso as funções “Execute” e “DisplayIt”. Essas funções são construídas no arquivo Unit2.cpp e servem respectivamente para verificar de tempos em tempos se há dados a serem recebidos na porta serial e para exibir na tela do programa o dado lido pela serial. Na Unit2.cpp também é implementado o método construtor da função TRead, necessário para que se possa criar uma instância dessa classe que permite que as funções que ela implementa sejam utilizadas.

O serviço de multitarefas é encerrado quando se elimina a instância da classe TRead, fazendo com que ela passe a não existir mais. O processo de encerramento pode ser visto no evento de fechamento do Form1 (“TForm1::FormClose”) na Unit1.cpp cujo código é mostrado no Apêndice E.

3.6.1.3 Leitura e escrita de dados via comunicação serial

Para se realizar o envio de dados (escrita) por comunicação serial é utilizada a função “WriteFile”. Essa função tem como parâmetros a variável hComm que armazena os parâmetros de configuração da comunicação, a variável na qual se encontra o dado que se deseja enviar, o número de *bytes* a serem enviados, uma variável do tipo ponteiro inteiro longo que armazena o número total de *bytes* enviados e uma variável utilizada na comunicação com *overlap* que no caso do projeto é atribuída para NULL.

O recebimento de dados (leitura), por sua vez, é realizado com a função “ReadFile” que possui os mesmo parâmetros da WriteFile. A diferença é que as variáveis indicam o local onde os dados recebidos serão salvos e o número de *bytes* recebidos.

A função WriteFile pode ser vista na Unit1.cpp, no evento de clique de cada um dos botões do controle ou do botão enviar (“TForm1::Button14Click”, no caso do botão Enviar). A função ReadFile pode ser vista na Unit2.cpp rodando dentro da função Execute.

3.6.2 Desenvolvimento e gravação de combos

A função de cada elemento na tela principal do *software* é mostrada na Figura 60. Nela é possível observar a presença dos componentes envolvidos na gravação de combos via *software*. A seguir será descrito o processo de gravação desses combos

Figura 60 - Função de cada elemento na tela principal do programa.

Fonte: Autoria Própria.

O primeiro passo para a gravação é a seleção do endereço no qual o combo será salvo quando for descarregado na placa. Quando o campo de seleção de endereço está vazio, o acionamento de qualquer botão do controle seta o botão, e consequentemente o endereço da EEPROM, no qual o combo será salvo. Através do clique no botão Limpar o campo de seleção de endereço é esvaziado e pode-se definir outra posição para o combo ser salvo.

Após a seleção de endereço pode-se começar a definir os comandos e tempos do combo. Quando se seleciona algum botão do Controle, o comando correspondente seguido da informação se é um comando de acionamento (“[ON]”) ou desligamento (“[OFF]”) é mostrado na área de visualização do combo atual. Esse comando pode ser sucedido por outro comando ou por um tempo em ms configurado no campo de seleção de tempos. Quando há um tempo nesse campo e o usuário clica no botão OK, esse tempo aparece na área de visualização do combo atual. Vale ressaltar que não é possível a inserção de dois tempos consecutivos.

Pode-se inserir novos comandos ou tempos em qualquer parte da área de visualização do combo atual (“ListBox1”), sempre se respeitando a não possibilidade de inserção de dois tempos consecutivos. Para se inserir um comando ou tempo em qualquer parte da lista de comandos, basta que se selecione com o mouse o item na posição imediatamente anterior a posição que se deseja inserir o novo comando ou tempo. Caso nenhum item esteja selecionado, o novo comando ou tempo será inserido na última posição.

Através do acionamento dos botões de edição pode-se apagar todos os comandos e tempos presentes na área de visualização de combos (Limpar Tudo), apagar apenas um item selecionado (Apagar Selecionado) e também realizar a edição de um item que esteja selecionado (Editar Selecionado). A tela de edição é mostrada na Figura 61.

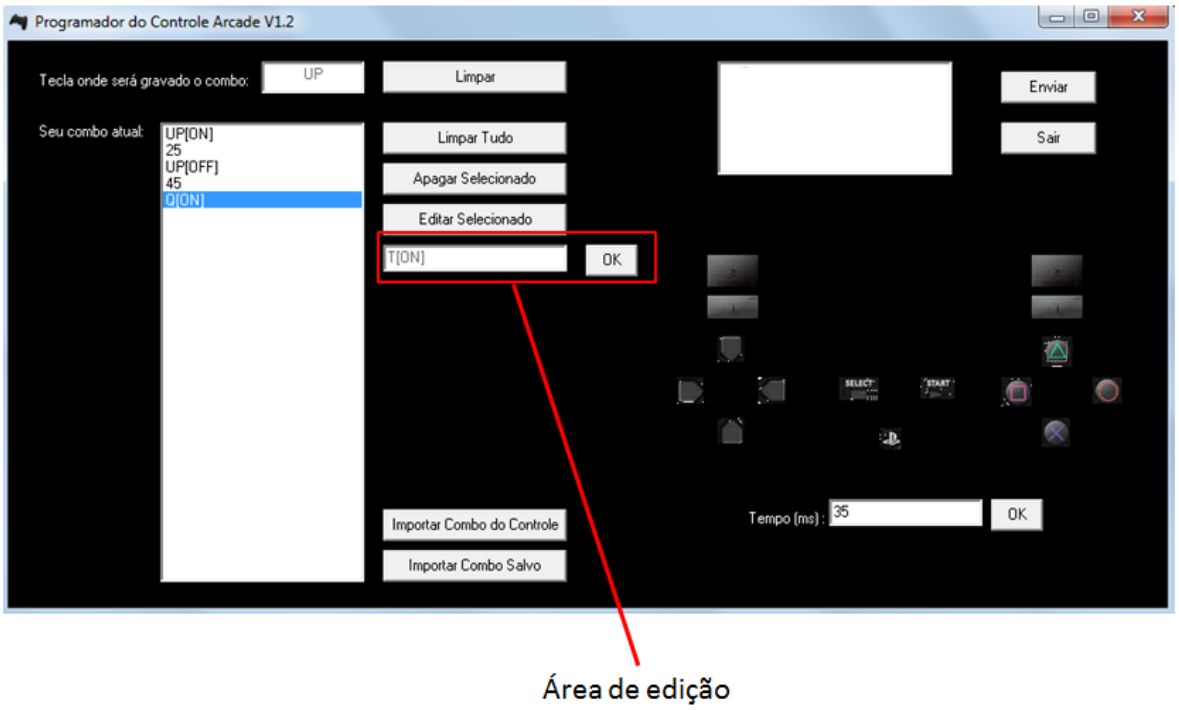


Figura 61 - Área de edição de comandos/tempos.
Fonte: Autoria Própria.

No caso de uma edição, qualquer botão selecionado do controle colocará o respectivo comando no campo da área de edição e quando ocorrer o clique no botão OK dessa área o item selecionado na área de visualização de comandos e tempos será substituído pelo comando presente no campo da área de edição. Vale ressaltar que um comando apenas pode ser substituído por outro comando e um tempo somente por outro tempo.

O envio de um combo para a placa acontece quando se clica no botão Enviar. Quando esse botão é pressionado, todos os comandos e tempos da área de visualização de combos são convertidos para seus respectivos caracteres de acordo com o Quadro 7 da seção 3.5.3 e todos os tempos são convertidos para variáveis do tipo *unsigned char* utilizando o recurso *union*, tal qual descrito na seção 3.5.3. Esses caracteres convertidos são salvos em um *array* do tipo *unsigned char* juntamente com os caracteres de controle do protocolo (“#” e “@”) e o caracter que indica o endereço no qual o combo será salvo (convertido do campo de seleção de endereço) e enviados pela serial usando a função WriteFile.

O envio dos bytes é feito da seguinte maneira: primeiro é enviado o caracter “#” que quando é recebido pela placa faz com que o microcontrolador envie um caracter “1” de confirmação. Depois é enviado o caracter que mostra o endereço

onde o combo deverá ser salvo, quando esse caracter é recebido o microcontrolador envia um caracter “2”. Por fim, os comandos e tempos do combo são enviados três a três ao passo que o microcontrolador ao recebê-los e gravá-los na EEPROM responde com uma mensagem de confirmação “OK!”. A última sequência “Comando-Tempo” é “@ 0 0”, ao receber essa sequência o microcontrolador identifica fim da transmissão e responde com a mensagem de confirmação “@OK!@”. Essas confirmações de recebimento do microcontrolador podem ser visualizadas na área de visualização de respostas.

Ao fim do envio a variável que armazena os *bytes* do combo enviado é limpa para aguardar a transmissão de uma próxima sequência.

Esse processo de conversão e envio pode ser visualizado no evento de clique do botão Enviar (“TForm1::Button14Click”) no código mostrado no Apêndice E.

3.6.3 Visualização dos combos gravados no controle

Uma das funcionalidades do programa é a visualização dos combos gravados via *hardware*. Para que essa visualização ocorra é necessário que o usuário clique no botão Importar Combo do Controle que mostra a janela da Figura 62 quando clicado:

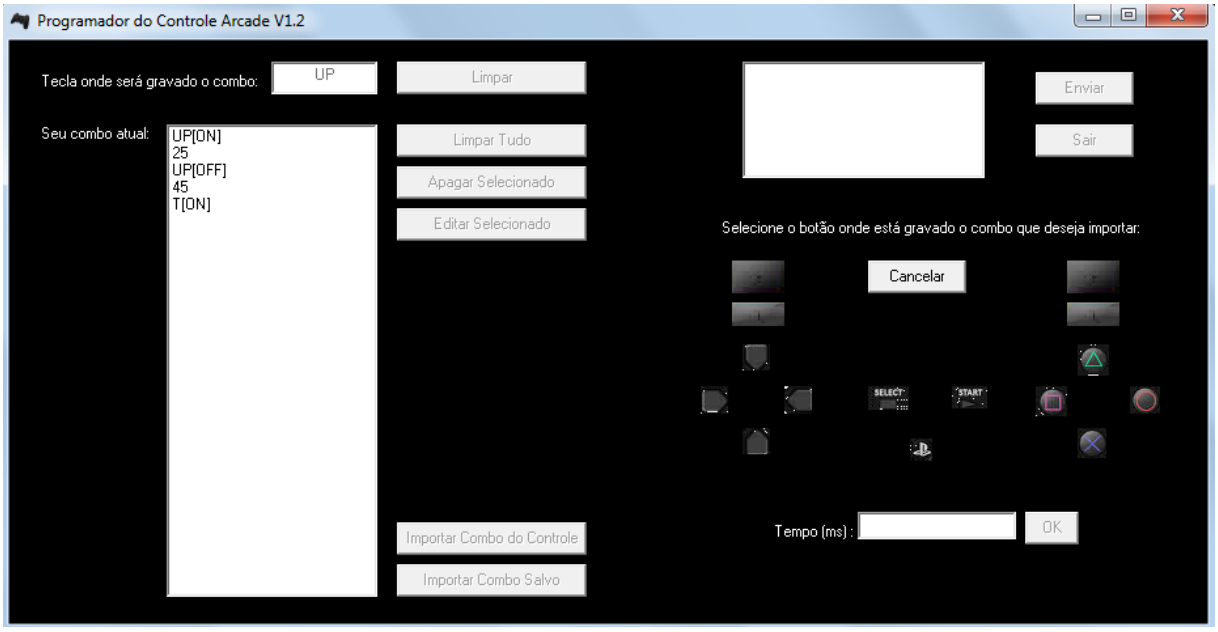


Figura 62 - Janela de importação de combos salvos no controle.

Fonte: Autoria Própria.

Pode-se observar que quando esse botão é clicado todos os outros são desabilitados. A funcionalidade básica do programa só volta ao normal quando ocorre um clique no botão Cancelar ou quando algum dos botões do controle é clicado. Quando ocorre esse clique em um dos botões do controle há a identificação do botão, conversão deste para o seu respectivo caracter e o envio de uma sequência “# End Z” para a placa via serial, onde “End” representa o caracter que marca o endereço no qual está salvo na EEPROM o combo que se deseja importar para o *software* do computador.

Quando o microcontrolador recebe essa mensagem, envia para o computador via serial o combo salvo na posição de memória solicitada. Os *bytes* referentes a esse combo são colocados em um campo do tipo *ListBox* (“ListBox2”) que não é visível ao usuário e a partir desse ListBox convertidos para o padrão do jogador (“Comando[Estado]” ou “tempo”) e atribuídos à área de visualização de combos. Essa conversão ocorre após um tempo de 1s (segundo) que começa a contar após a ativação de um componente *timer* (“Timer1”) que executa uma ação após um tempo determinado.

O processo de envio dessa informação de solicitação para o microcontrolador pode ser visto nos eventos de clique de cada um dos botões do controle, cujos códigos se encontram no Apêndice E desse trabalho.

3.6.4 Visualização de combos pré-programados

O *software* do computador vem com alguns combos pré-programados na memória, permitindo que o usuário carregue combos complexos com poucos cliques no *software*.

Para que possa visualizar os combos pré-programados o usuário deve clicar no botão Importar Combo Salvo que abre a janela mostrada na Figura 63.

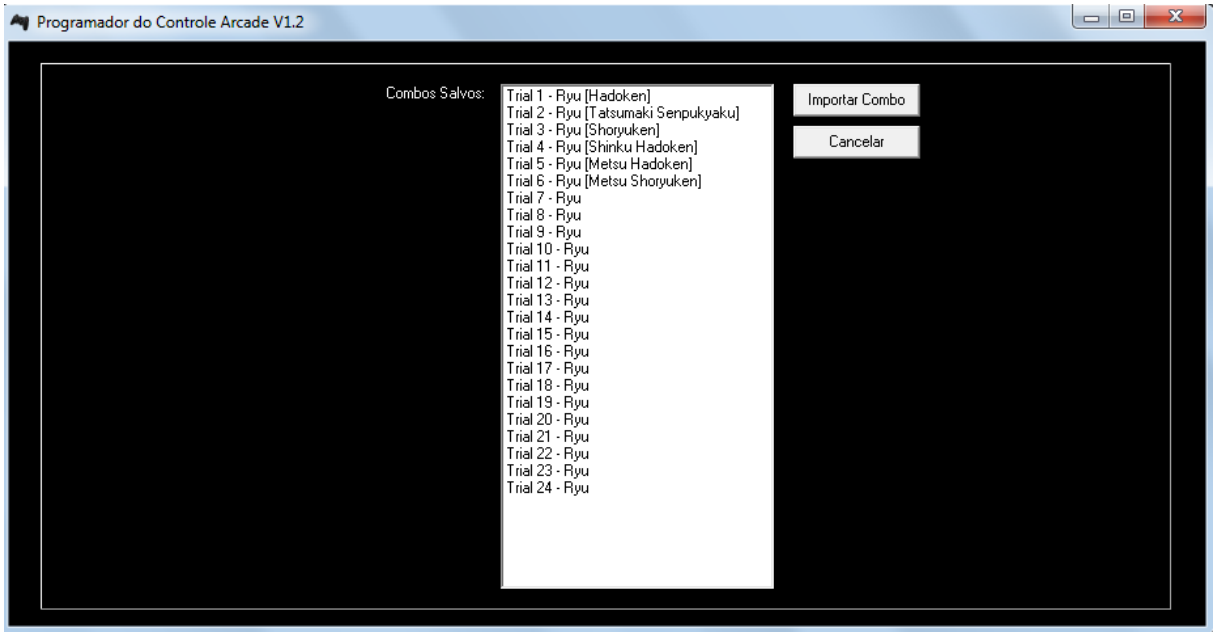


Figura 63 - Janela de importação de combos pré-programados.
Fonte: Autoria Própria.

Nessa janela o usuário pode selecionar algum dos combos e clicar em Importar Combo, o que faz com que os comandos e tempos relativos ao combo selecionado apareçam na área de visualização de combos da janela principal.

Um clique no botão Cancelar, por sua vez, faz com que o programa retorne a janela principal sem que se importe nenhum combo.

3.7 TESTES DO PROTÓTIPO

Os testes do protótipo foram realizados com a utilização do controle arcade para o jogo *Super Street Fighter*® *4* da *Capcom*®. Foram observadas as funcionalidades através de um modo de jogo chamado *trials* no qual o jogador deve realizar um combo (*trial*) para concluir um desafio e também o modo de treino onde o jogador joga contra um adversário parado somente para treinar a realização de golpes.

Para os testes da gravação via *software* foi utilizado o modo *trial*. Cada personagem do jogo possui um número total de 24 *trials* e para os testes do protótipo foi utilizado o personagem Ryu, já que este é um dos personagens mais utilizados por grande parte dos jogadores. Os *trials* do personagem Ryu foram desenvolvidos no *software* computacional e descarregados no controle arcade,

através do qual foi testado se o *trial* desenvolvido era executado com sucesso para ambos os lados.

O teste de gravação via *hardware* e de jogabilidade do controle foram realizados no modo de treino do jogo, no qual é possível a visualização dos botões que estão sendo pressionados. Essa visualização dos botões torna mais simples a verificação da gravação dos combos, já que se pode comparar os botões pressionados na gravação com os botões ativados no ato da execução desse combo gravado.

Os resultados obtidos nos testes poderão ser vistos na seção a seguir.

4 RESULTADOS OBTIDOS

4.1 JOGABILIDADE

Após alguns testes com diferentes perfis de jogadores concluiu-se que de maneira geral a jogabilidade do *game* foi facilitada com a utilização do controle. Essa facilidade observou-se na execução de combos e na movimentação do personagem.

Algumas sugestões de aprimoramento do controle dadas por alguns jogadores para melhorar ainda mais a jogabilidade com o controle foram:

- Realizar a troca do *joystick* para um com movimentação mais leve, ou seja, um *joystick* com o qual o usuário realize o movimento com menos força;
- Reorganizar os botões do controle.

Essas sugestões foram anotadas e podem proporcionar aprimoramentos em produtos futuros.

4.2 GRAVAÇÃO VIA *SOFTWARE*

O principal resultado obtido com a gravação via *software* foi a possibilidade do desenvolvimento de todos os 24 *trials* do personagem Ryu. Esses *trials* foram testados e verificou-se a correta execução do combo para ambos os lados através da execução com o Botão Direita e com o Botão Esquerda do controle arcade.

Os 24 *trials* desenvolvidos foram colocados no *software* computacional e podem ser visualizados na tela e baixados para o controle arcade a qualquer momento, bastando que o usuário clique no botão “Importar Combo Salvo” e selecione o combo que deseja visualizar.

A gravação via *software* foi considerada excelente por permitir que mesmo que o usuário não tenha um conhecimento grande no que diz respeito à realização de combos, ele possa desenvolver e gravar seus combos no controle de maneira fácil.

Uma das maiores dificuldades na execução de um combo é acertar o tempo de execução entre cada comando, já que para cada personagem e para cada golpe há um tempo de execução diferente. Tendo em vista esse fato, o *software* computacional também foi considerado ótimo, principalmente o recurso de edição, que facilita consideravelmente o desenvolvimento de combos por permitir que o usuário edite um tempo de maneira fácil e rápida, e também de maneira fácil e rápida veja o efeito da edição desse tempo na execução do combo.

No ato dos testes foi possível perceber quão complexa é a implementação de combos no que diz respeito ao tempo presente entre cada comando. Para alguns *trials* mais complexos foram necessárias algumas horas de tentativas de ajuste dos tempos para que a execução do *trial* ocorresse da maneira desejada. Percebeu-se também que após o ajuste dos tempos o combo é realizado com sucesso desde que seja respeitada a distância entre os personagens no ato do início da execução do combo. Tendo em vista esse fato pode-se dizer que a distância entre os personagens no início do combo também é um fator determinante para a correta execução do comando desejado.

O recurso do *software* de importação de combos gravados no controle arcade foi testado com sucesso e teve seu comportamento de acordo com o esperado.

4.3 GRAVAÇÃO VIA *HARDWARE*

Os resultados obtidos com os testes da gravação via *hardware* foram satisfatórios. Os testes apontaram que na maior parte das vezes a mesma sequência gravada é executada pelo personagem.

A maior dificuldade encontrada no desenvolvimento desse recurso foi a obtenção do tempo entre o acionamento de cada botão do teclado. Pelo fato da

leitura por varredura passar por todos os botões do teclado e a cada acionamento ou desligamento de uma tecla a gravação na memória EEPROM ocorrer, há uma perda de alguns ms que pode fazer com que o golpe gravado não seja executado de maneira correta. Nos testes pôde-se observar que os botões pressionados na gravação são os mesmos acionados na execução do combo, porém os tempos obtidos podem não ser os mesmos, o que algumas vezes faz com que o combo não seja realizado da mesma maneira que o usuário o gravou.

5 CONSIDERAÇÕES FINAIS

De maneira geral os testes com o produto mostraram o funcionamento de acordo com o que era esperado. Foi desenvolvido um controle estilo arcade que não necessita de um cabo que o ligue ao console, apenas de um cabo de alimentação e de um cabo adaptador serial-USB que liga o controle ao computador caso o usuário deseje utilizar o *software* computacional.

Esse controle possui uma IHM amigável que permite a gravação de combos via *hardware* ou via *software* de maneira fácil. Também permite que um combo gravado seja executado de maneira fácil, através do acionamento de um botão que define o lado para o qual a execução será realizada, juntamente com o botão que acessa esse combo salvo na memória.

As maiores dificuldades encontradas na realização do projeto foram no que diz respeito ao tempo de execução de cada comando presente em um combo. Esses problemas foram resolvidos com a utilização dos *timers* do microcontrolador que fizeram a geração de tempo para gravação (no caso da gravação via *hardware*) e para a execução dos comandos gravados.

Para projetos futuros podem ser realizadas as melhorias sugeridas por alguns usuários que fizeram testes do produto, citadas na seção 4. Além dessas melhorias outras funcionalidades podem ser implementadas, como:

- Criação de “bibliotecas de combos” nas quais o usuário pode salvar no seu computador os combos que desejar para utilizá-los a qualquer momento;
- Gravação de mais de um combo a ser acessado pelo mesmo botão do controle;

- Melhorias na interface gráfica do *software* computacional;
- Desenvolvimento de mais combos pré-programados no *software*.
- Utilização de um microcontrolador que permita a alimentação com a bateria do próprio controle, o que permitiria que o controle arcade necessitasse de fio somente para a ligação com o computador.

Com as funcionalidades básicas já desenvolvidas e com os aprimoramentos citados, o controle possui um grande potencial de mercado a ser explorado. Esse fato é ressaltado pela opinião dos diferentes jogadores que testaram o produto e disseram estar satisfeitos com o que ele oferece.

No que diz respeito ao aprendizado obtido com a execução do projeto, com certeza pode-se afirmar que foi um trabalho que proporcionou uma experiência muito grande no desenvolvimento de produtos com diferentes ferramentas de *hardware* e *software*. Experiência essa que será com toda certeza não somente útil, mas fundamental no exercício da atividade profissional de Engenheiro Eletrônico.

REFERÊNCIAS

ABREU, Ivair R. N. **Apostila Microprocessadores**. São Paulo, 2005.

ATMEL Corp. **8-bit Microcontroller with 8K Bytes In-System Programmable Flash – AT89S52**. San Jose, CA, USA: Atmel. 2008. 38 p. Disponível em: <http://www.atmel.com/images/doc1919.pdf>. Acesso em: 19 set. 2014.

ATMEL Corp. **Two-wire Serial EEPROM – AT24C512B**. San Jose, CA, USA: Atmel. 2008. 26 p. Disponível em: < http://www.atmel.com/Images/doc5297.pdf>. Acesso em: 22 set. 2014.

BATISTA, Maria de Lourdes S. et al. Um estudo sobre a história dos jogos eletrônicos. **Revista Eletrônica da Faculdade Metodista Granbery**, Juiz de Fora, n. 3, jul/dez. 2007. Disponível em: < http://re.granbery.edu.br/artigos/MjQ4>. Acesso em: 8 fev. 2014.

BIENAIMÉ, Pierre. **Square Roots:** *Donkey Kong (NES)*. Disponível em: <http://www.nintendojo.com/features/columns/square-roots/square-roots-donkey-kong-nes>. Acesso em: 23 mai. 2014.

BRAGA, Newton C. **Controle Biestável de Toque (MEC056)**. Disponível em: < http://www.newtoncbraga.com.br/index.php/176-automacao/automacao-industrial /2626-mec056>. Acesso em: 12 jan. 2015.

BRESCIANI, Alex Antonio. **A guerra dos botões:** a estética da violência nos jogos eletrônicos. In: VIII Jornada de Iniciação Cientifica do Campus de Marilia, 2001, Marília. **Resumos...** Marília: Unesp Marília Publicações, 2001. p. 98-99.

BUENO, Silvia H. Espaços entre Disputas: Do Lar para o Fliperama. Uma História das Representações do Entretenimento Informático no Brasil (1970-90). In: Simpósio Brasileiro de Jogos e Entretenimento Digital, 11, 2012, Brasília. **Anais eletrônicos...** Porto Alegre: SBC, 2012. Disponível em: <http://sbgames.org/sbgames2012/proceedings/papers/cultura/C_S19.pdf>. Acesso em: 10 fev. 2014.

BURNETT, Colin M. L. ***SPI timing diagram***. Disponível em: < http://commons .wikimedia.org/wiki/File:SPI_timing_diagram.svg > Acesso em: 11 jan. 2015.

BYTE PARADIGM. ***Introduction to I2C and SPI protocols***. Disponível em: < http:// www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/ >. Acesso em: 11 jan. 2015.

CAMARA, Romulo C. P. **Protocolo I2C**. Disponível em: <http://www.univasf.edu.br/ ~romulo.camara/novo/wp-content/uploads/2013/11/Barramento-e-Protocolo-I2C.pdf> . Acesso em: 5 out. 2014.

CAMPOS, Leonardo B. **Microprocessadores e Microcontroladores**. Petrolina, 2008.

CAMPOS, Leonardo B. **Conceitos Básicos da Linguagem C**. Disponível em: <http://www.univasf.edu.br/~leonardo.campos/Arquivos/Disciplinas/Alg_Prog_I_2007_1/Aula_01.pdf>. Acesso em: 13 jan. 2015.

CANALLE, Gabrielle K.; ARAÚJO, Everton C. C/C++ e Java: Principais Características. **Revista Easy Java Magazine**, São Paulo, v. 25, [20--]. Disponível em: < http://www.devmedia.com.br/c-c-e-java-principais-caracteristicas-revista-easy-java-magazine-25/26773>. Acesso em: 13 jan. 2015.

CHACON, José Marcelo T. **Borland C++ Builder**. Campinas, [20--].

CONTROLE PS3 – Reparo/Peças/Revisão. Disponível em: < http://www.htforum.com/forum/threads/controle-ps3-reparo-pecas-revisao.128700/>. Acesso em: 23 mai. 2014.

CRAWFORD, C. ***Chris Crowford on Game Design***. Indianapolis: New Riders, 2003.

DUTTA, Prabal. **ROM, EPROM, *and* EEPROM *Technology***. Disponível em: < https://web.eecs.umich.edu/~prabal/teaching/eecs373-f10/readings/rom-eprom-eeprom-technology.pdf>. Acesso em: 09 jan. 2015.

EMBARCADERO. **C++ Builder XE7**. Disponível em: <http://www.embarcadero.com/br/products/cbuilder>. Acesso em: 14 jan. 2015.

EVENT HUBS. Disponível em: < http://www.eventhubs.com/images/2010/oct/01/old-street-fighter-2-arcade-cabinet/>. Acesso em: 11 fev. 2014.

FAIRCHILD CORP. ***LM78XX / LM78XXA 3-Terminal 1 A Positive Voltage Regulator***. Disponível em: <https://www.fairchildsemi.com/datasheets/LM/LM7805.pdf>. Acesso em: 16 jan. 2015.

FARAH, Ricardo. **Aposentadoria Gamer:** *Arcade*. Disponível em: <http://www.techtudo.com.br/platb/jogos/2012/01/29/aposentadoria-gamer-arcade/>. Acesso em: 10 fev. 2014.

FÓRUM UOL JOGOS. Disponível em: < http://forum.jogos.uol.com.br/_t_2514765>. Acesso em: 10 fev. 2014.

FRUETT, Fabiano. **Memórias Semicondutoras**. Disponível em: < http://www.dsif.fee.unicamp.br/~fabiano/EE610/PDF/3_b%20Memorias.pdf>. Acesso em: 09 jan. 2015.

GOLÇALVES, Deise A.; FERREIRA, Marcelo G. G. *Fighting Dragons or Saving a Princess: What is a game?* In: Simpósio Brasileiro de Jogos e Entretenimento Digital, 11, 2012, Brasília. **Anais eletrônicos...** Porto Alegre: SBC, 2012. Disponível em:

<http://sbgames.org/sbgames2012/proceedings/papers/artedesign/AD_Full23.pdf >. Acesso em: 10 fev. 2014.

HESSEL, Roberto. et al. Contadores eletrônicos no laboratório didático. **Revista Brasileira de Ensino de Física**, Rio Claro, v. 30, n. 1, jan. 2008. Disponível em: < http://www.sbfisica.org.br/rbef/pdf/301501.pdf>. Acesso em: 12 jan. 2015.

HUIZINGA, J. ***Homo Ludens: A study of the Play Element in Culture***. Londres: Routledge, 2003.

HUNICKE, Robin; LEBLANC, Marc; ZUBEK, Robert. ***MDA:*** *A formal approach to game design and game research*. Disponível em: <http://www.cs.northwestern.edu/~hunicke/MDA.pdf>. Acesso em: 9 fev. 2014.

I2C-BUS. ***I2C-Bus:*** *What's that?*. Disponível em: <http://www.i2c-bus.org/>. Acesso em: 11 jan. 2015.

IEEE. ***IEEE Standard Glossary of Software Engineering Terminology***. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=159342&tag=1>. Acesso em: 13 jan. 2015.

KEIL. ***Getting Started:*** *Creating Applications with µVision® 4.* Disponível em: <http://www.keil.com/product/brochures/uv4.pdf >. Acesso em: 13 jan. 2015.

KEVIE. ***Retro Arcade Joystick Clipart By Kevie***. Disponível em: < http://clipartse .com/clipart/retro-arcade-joystick-computer-clipart-5932>. Acesso em: 23 mai. 2014.

LENZ, André L. **Linguagem para Programar Microcontroladores:** Assembly, C ou BASIC? São Paulo, [20--].

MACIEL, Braian K. et al. **Barramento Serial I^2C e SPI**. Rio Grande, 2013.
Mad Catz Fightstick PRO. Disponível em: <http://www.madcatz.com/SFXTekken/fightstickpro.html>. Acesso em: 10 fev. 2014.

MENDES, Bernardo L. O fenômeno retrô nos jogos eletrônicos: fatores que mudaram a percepção dos jogadores. In: Simpósio Brasileiro de Jogos e Entretenimento Digital, 11, 2012, Brasília. **Anais eletrônicos...** Porto Alegre: SBC, 2012. Disponível em: <http://sbgames.org/sbgames2012/proceedings/papers/artedesign/AD_Full22.pdf>. Acesso em: 10 fev. 2014.

MESSIAS, Antonio R. **Funções da API do Windows para acessar a porta serial**. Disponível em: <http://www.rogercom.com/CursoOnlineUSB/ModuloUnicoAula014 .htm>. Acesso em: 12 jan. 2015.

NICOLOSI, Denis E. C. **Microcontrolador 8051 detalhado.** São Paulo: Editora Érica, 2007.

NICOLOSI, Denys E. C; BRONZERI, Rodrigo B. **Microcontrolador 8051 com linguagem C:** prático e didático: família AT89S8252 atmel. São Paulo: Editora Érica, 2008.

PAULA, André Luiz de S. **Memórias**. Disponível em: < http://gerds.utp.br/andre/hardware/Memorias.pdf>. Acesso em: 09 jan. 2015.

PERES, Angela L. et al. Estudo dos elementos de interatividade recomendados em jogos e a satisfação na experiência dos jogadores de *The Elder Scrolls V: Skyrim*® e *Left 4 Dead 2*®. In: Simpósio Brasileiro de Jogos e Entretenimento Digital, 11, 2012, Brasília. **Anais eletrônicos...** Porto Alegre: SBC, 2012. Disponível em: <http://sbgames.org/sbgames2012/proceedings/papers/artedesign/AD_Full22.pdf>. Acesso em: 10 fev. 2014.

PETRY, Arlete dos S. *Heavy Rain* ou o que podemos vivenciar com as narrativas dos games. In: Simpósio Brasileiro de Jogos e Entretenimento Digital, 10, 2011, Salvador. **Anais eletrônicos...** Porto Alegre: SBC, 2011. Disponível em: <http://www.sbgames.org/sbgames2011/proceedings/sbgames/papers/cult/full/92019_1.pdf>. Acesso em: 10 fev. 2014.

PLAYSTATION. ***DUALSHOCK® 3 Wireless Controller***. Disponível em: <http://us.playstation.com/ps3/accessories/dualshock-3-wireless-controller-ps3.html>. Acesso em: 14 jan. 2015.

PLAYSTATION KNOWLEDGE CENTER. **PS3 Controller – Part Names**. Disponível em: <https://support.us.playstation.com/app/answers/detail/a_id/960/~/ps3-controller---part-names>. Acesso em: 14 jan. 2015.

Programmable Combo Fight stick for PS3, Xbox 360. Disponível em: <http://insanecombo.com>. Acesso em: 20 jan. 2014.

ROCHA, Rui. **Protocolo SPI**. Disponível em: < https://cld.pt/dl/download/03fa4282-3f3f-4231-8907-2e21bfed8207/N3E%20-%20SmartBoost/4%C2%BA%20Ano/1%C2%BA%20semestre/Sistemas%20Embebidos/Prof.%20Rui%20Rocha/Projectos/protocolo_spi_v2.pdf >. Acesso em: 11 jan. 2015.

SACCO, Francesco. **Comunicação SPI – Parte 1**. Disponível em: < http://www.embarcados.com.br/spi-parte-1/ >. Acesso em: 11 jan. 2014.

SENA, Alexandre; COELHO, Denis K. Motivação dos Jogadores de Videogame – Uma breve visão sobre as Técnicas de Engajamento. In: Simpósio Brasileiro de Jogos e Entretenimento Digital, 11, 2012, Brasília. **Anais eletrônicos...** Porto Alegre: SBC, 2012. Disponível em: <http://sbgames.org/sbgames2012/proceedings/papers/cultura/C_S4.pdf>. Acesso em: 10 fev. 2014.

SHAW, Greg. ***A Brief History of Programming Languages***. Disponível em: <users.cis.fiu.edu/~shawg/2210/languages.doc>. Acesso em: 13 jan. 2015.

SILVA, Bruno S. et al. **Padrões de Comunicação Serial Clássicos:** RS-232, RS-422 e RS-485. Disponível em: < http://www.peb.ufrj.br/cursos/eel710/ProtocoloRS .pdf>. Acesso em: 12 jan. 2015.

SONY COMPUTER ENTERTAINMENT INC. ***PS3 Safety and Support***. Disponível em: <http://au.playstation.com/media/k7d4A6Q3/CECH-3002A_3002B_3003A_ 3003B-3.70_SS_vf1.pdf>. Acesso em: 14 jan. 2015.

ST MICROELECTRONICS. ***L78L00 SERIES***. Disponível em: <http://pdf.datasheetcatalog.com/datasheet2/c/0h3c9x5wgla21jy0d1a9sp4y53fy.pdf> . Acesso em: 15 jan. 2015.
TAROUCO, Liane Margarida R. et al. **Jogos educacionais**. Disponível em: <http://www.cinted.ufrgs.br/ciclo3/af/30-jogoseducacionais.pdf>. Acesso em: 10 fev. 2014.

TEXAS INSTRUMENTS. ***CD4066B CMOS QUAD Bilateral Switch***. Disponível em: <http://www.ti.com/lit/ds/symlink/cd4066b.pdf>. Acesso em: 15 jan. 2015.

TYSON, Jeff. *How PlayStation Works*. Disponível em: <http://electronics.howstuff works.com/playstation3.htm>. Acesso em: 23 mai. 2014.

UOL JOGOS. **BGS: Mercado de games no Brasil cresce em média 26% ao ano, diz estudo**. Disponível em: <http://jogos.uol.com.br/ultimas-noticias/2014/10/08/bgs-mercado-de-games-no-brasil-cresce-em-media-26-ao-ano-diz-estudo.htm>. Acesso em: 28 dez. 2014.

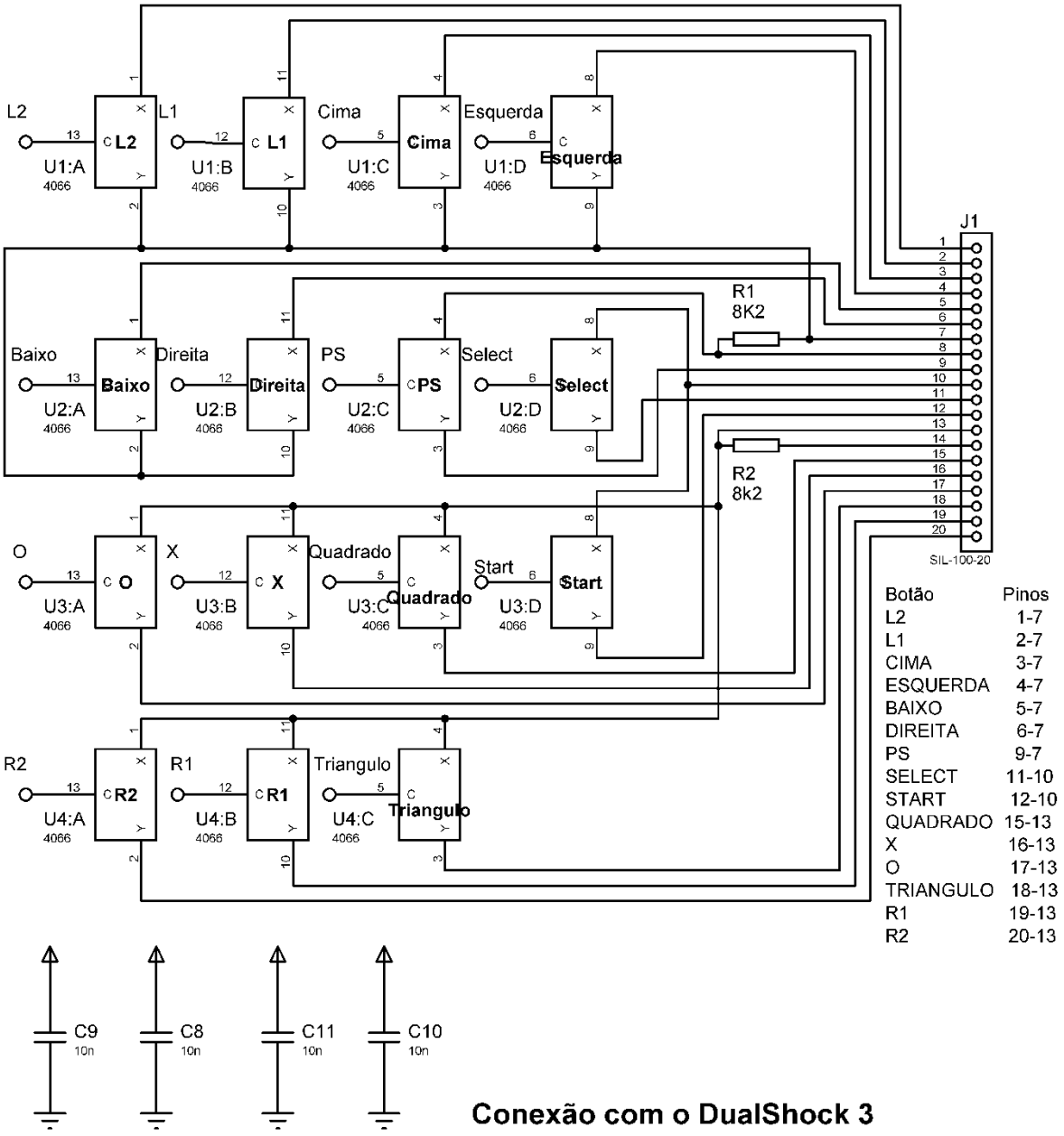
UOL JOGOS. Disponível em: < http://jogos.uol.com.br/album/2013/01/25/controles-de-videogame.htm>. Acesso em: 9 fev. 2014

VIEIRA, Marcelo A. da C. **Dispositivos de memórias semicondutoras**. Disponível em: < http://iris.sel.eesc.usp.br/sel415m/SEL0415%20-%20Cap5%20-%20Memorias _P2.pdf>. Acesso em: 09 jan. 2015.

XCM Dominator Joystick. Disponível em: <http://xcm.cc/products/xcm-dominator-joystick>. Acesso em: 09 mar. 2014.

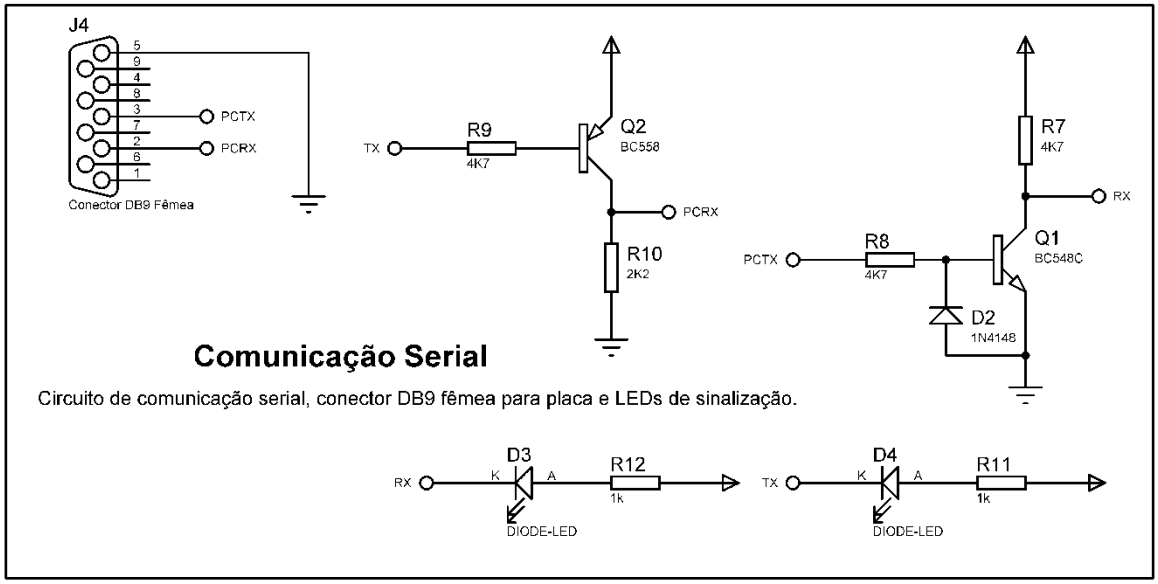
APÊNDICES

APÊNDICE A – Diagrama esquemático da placa de protótipo.



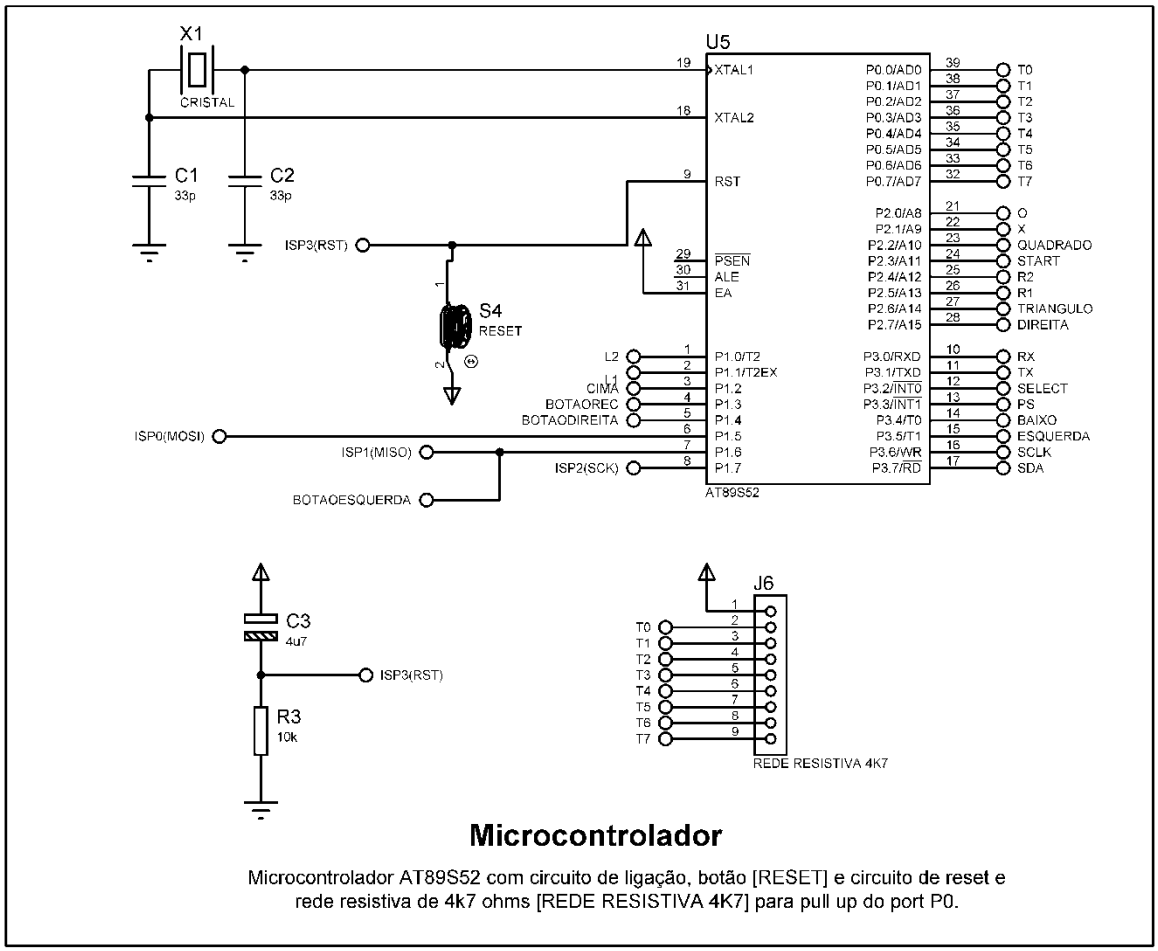
Circuitos de chave analógica com controle digital [CD4066] e capacitores de desacoplamento destes.

APÊNDICE A (CONTINUAÇÃO)



Comunicação Serial

Circuito de comunicação serial, conector DB9 fêmea para placa e LEDs de sinalização.

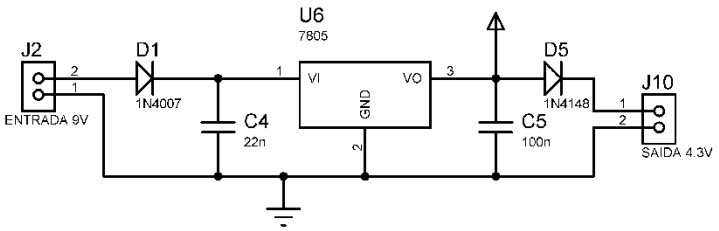


Microcontrolador

Microcontrolador AT89S52 com circuito de ligação, botão [RESET] e circuito de reset e rede resistiva de 4k7 ohms [REDE RESISTIVA 4K7] para pull up do port P0.

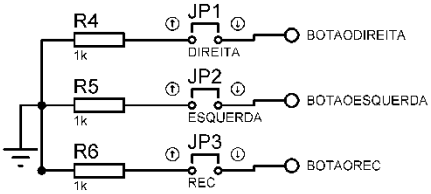
APÊNDICE A (CONTINUAÇÃO)

Alimentação



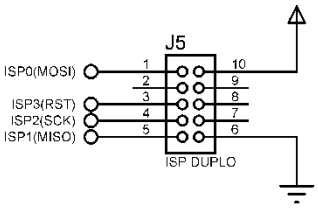
Conectores de alimentação e reguladores de tensão.
7805 = Regulador para 5V
1N4148 = Regula a tensão para 4.3V [Alimentação do controle do PS 3]
ENTRADA 9V = Conector KRE de 2 vias para alimentação da placa
SAIDA 4.3V = Conector KRE de 2 vias para alimentação do controle do PS 3.

Botões da placa



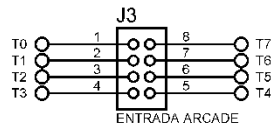
Botões da placa:
BOTAODIREITA = Realiza Combos para a direita;
BOTAOESQUERDA = Realiza Combos para a esquerda;
BOTAOREC = Gravação de Combos via Hardware.

Gravação ISP



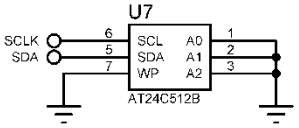
Barra de pinos dupla para gravação do microcontrolador via ISP. [ISP DUPLO]

Conexão com o Arcade



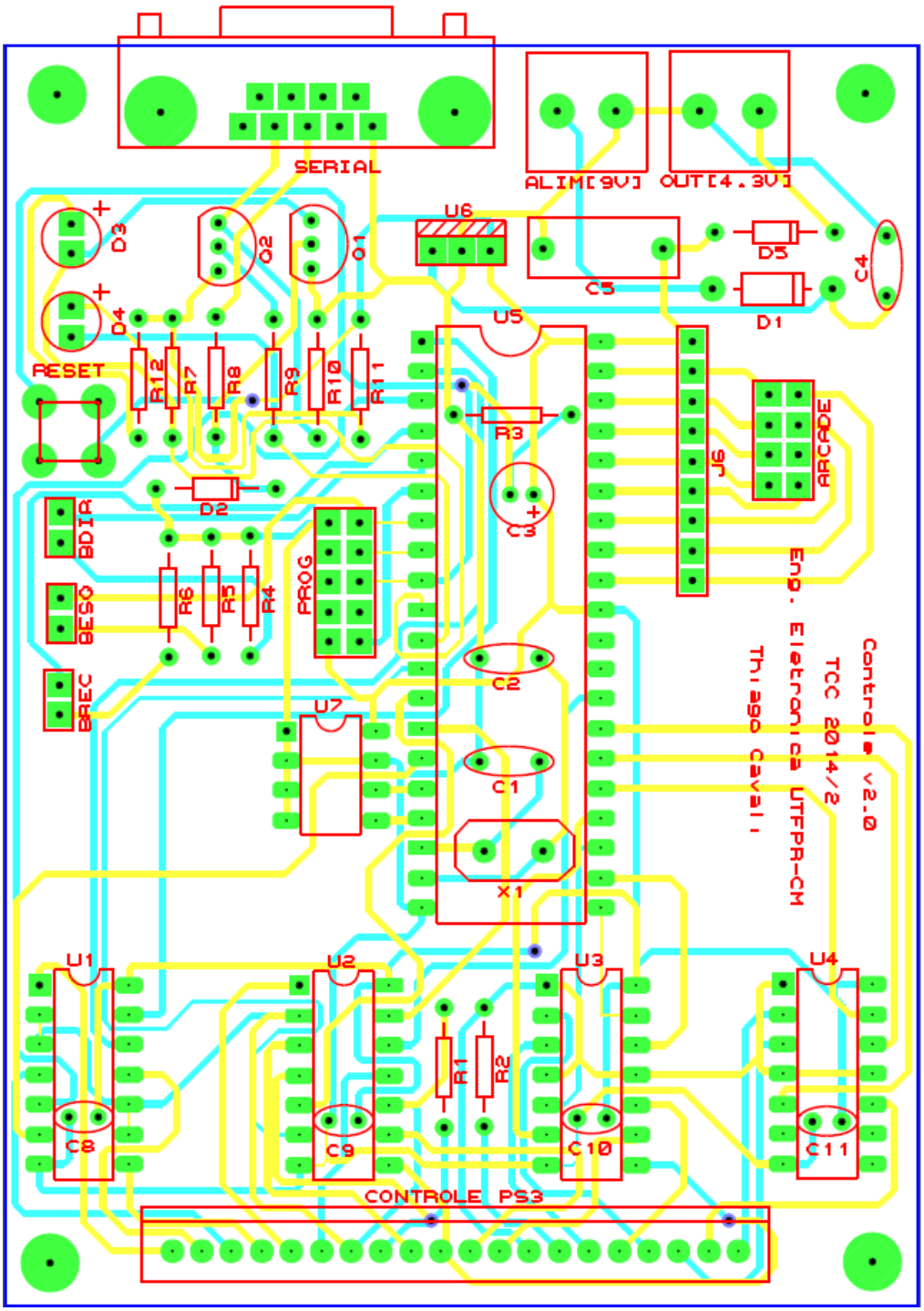
Conector duplo de 8 pinos para ligação dos botões [ENTRADA ARCADE]

Memória EEPROM



Circuito Integrado de memória EEPROM [AT24C512B]

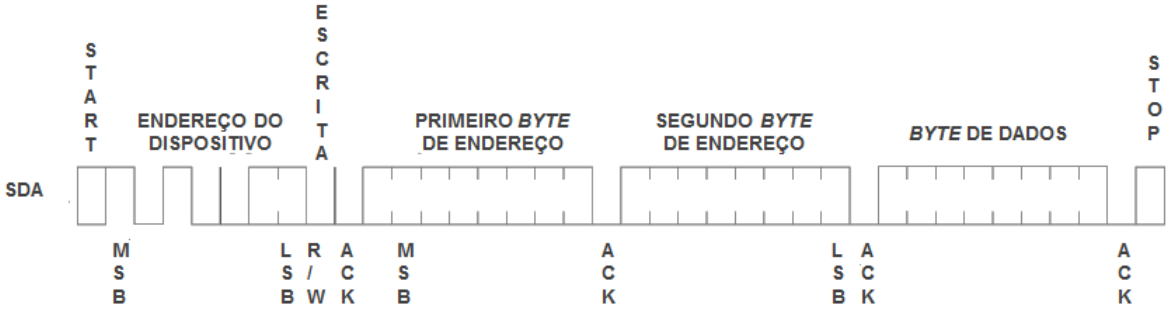
APÊNDICE B – *Layout* da Placa de Circuito Impresso.



APÊNDICE C – Protocolo de Gravação e Leitura da memória EEPPROM AT24C512

O processo de comunicação do microcontrolador com a memória EEPROM utiliza o protocolo TWI, mostrado na seção 2.5.3.2, sendo assim os processos de leitura e escrita são similares ao descrito naquela seção, com algumas diferenças como o fato de haver dois *bytes* de endereço, como será visto abaixo.

De acordo com Atmel Corp. (2008), o processo de escrita de um *byte* na memória EEPROM AT24C512 é feito de acordo com a figura abaixo:



Nessa figura pode ser observado o *start bit* que é uma transição de nível 1 para nível 0. Esse *bit* é seguido de um *byte* composto por quatro bits fixos (1010) e três *bits* (A_2, A_1 e A_0) que dão o endereço físico do dispositivo, o que é necessário em aplicações que possuem mais de uma memória ligadas no mesmo barramento. O último *bit* (R/W) representa a operação que se deseja realizar (0 para escrita e 1 para leitura). A estrutura desse *byte* pode ser vista na figura a seguir, na qual MSB (*Most Significant Bit*) indica o *bit* mais significativo do *byte* e LSB (*Least Significant Bit*) o menos significativo.

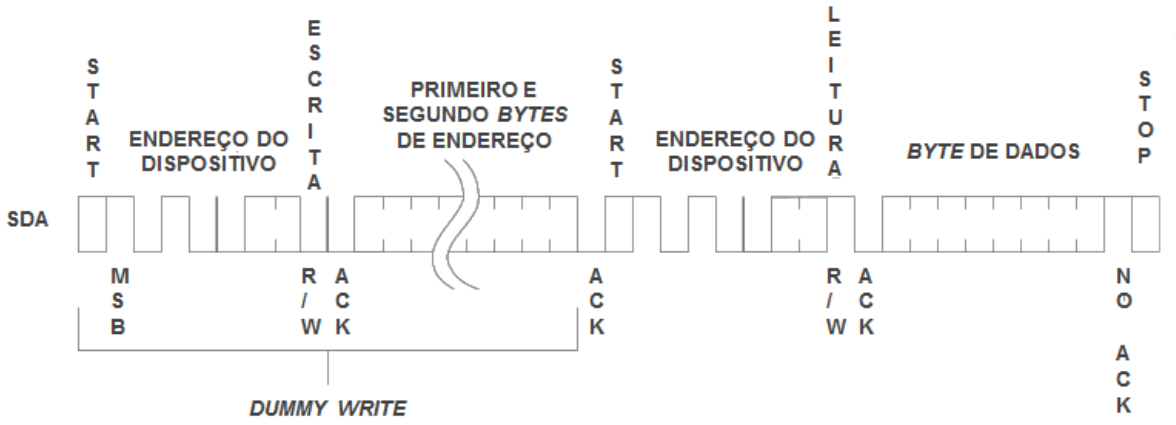
Como no projeto é utilizado somente um *chip* de memória no barramento e a operação desejada é de escrita, o valor atribuído ao *byte* é o hexadecimal 0xA0, equivalente ao binário 10100000.

Após o envio desse *byte* a memória retorna para o microcontrolador um *bit* de reconhecimento ACK que quando verificado pelo microcontrolador faz com que este

envie o primeiro *byte* que define o endereço da EEPROM no qual será guardado o *byte* de dados. Após o envio desses dados um novo ACK é verificado possibilitando o envio do segundo *byte* indicador de endereço.

Após a verificação de um novo ACK, o microcontrolador envia o *byte* de dados para ser armazenado e aguarda a confirmação da EEPROM. Com essa confirmação o microcontrolador envia um *stop bit* (transição de 0 para 1) encerrando o envio de dados.

A leitura de dados em posições randômicas da memória é feita de acordo com a figura abaixo:



Para esse modo de leitura é necessário que se realize um processo de "*Dummy Write*". Nesse processo é realizada a escrita do *byte* de endereço do dispositivo e dos dois *bytes* de endereço dentro da EEPROM, tal qual no modo de leitura. Quando o processo de "*Dummy Write*" termina o microcontrolador envia outro *start bit*, além do *byte* de endereço do dispositivo dessa vez com o *bit* R/W em valor alto. Ao receber o ACK vindo da memória passa a receber também os dados que solicitou. Ao fim da recepção transmite para a memória um *bit* 1 (NO ACK) e em seguida um *stop bit* indicando fim da comunicação.

Vale lembrar que cada *bit* 0 ou 1 tem o valor lido quando há um pulso na linha SCL que dá o sincronismo para as operações

As funções do código apresentado no Apêndice D que implementam essa comunicação com a memória são:

- "Start": Implementa o *start bit*;
- "Stop": Implementa o *stop bit*;
- "noAck": Implementa o *bit* de NO ACK;
- "testeAck": Verifica o recebimento de um ACK vindo da memória;

- “escreve8bit”: Realiza a operação de escrita de 1 *byte* na memória;
- “escreve24cXX”: Implementa o protocolo de escrita na memória;
- “le8bit”: Realiza a operação de leitura de 1 *byte* na memória;
- “le24cXX”; Implementa o protocolo de leitura de dados da memória;

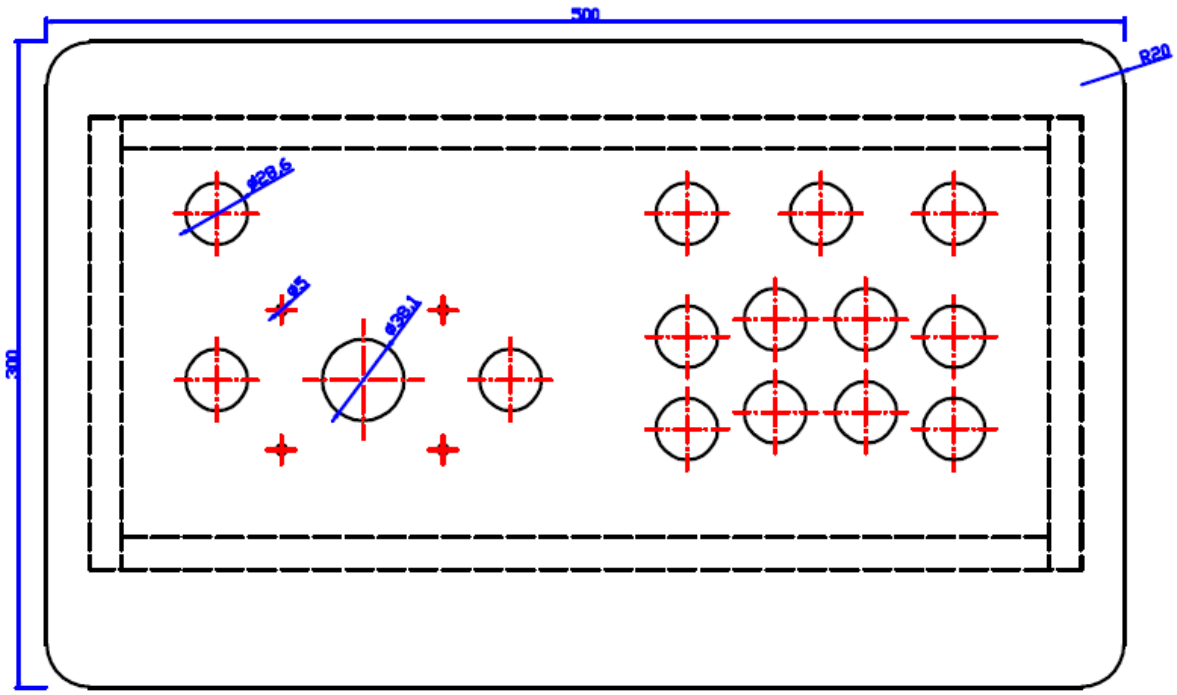
APÊNDICE D – Código-fonte do *software* do microcontrolador

O código-fonte do *software* do microcontrolador está disponível à banca de avaliação em um arquivo a parte deste. O código não constará nesse trabalho pelo fato de o produto desenvolvido ter potencialidades comerciais.

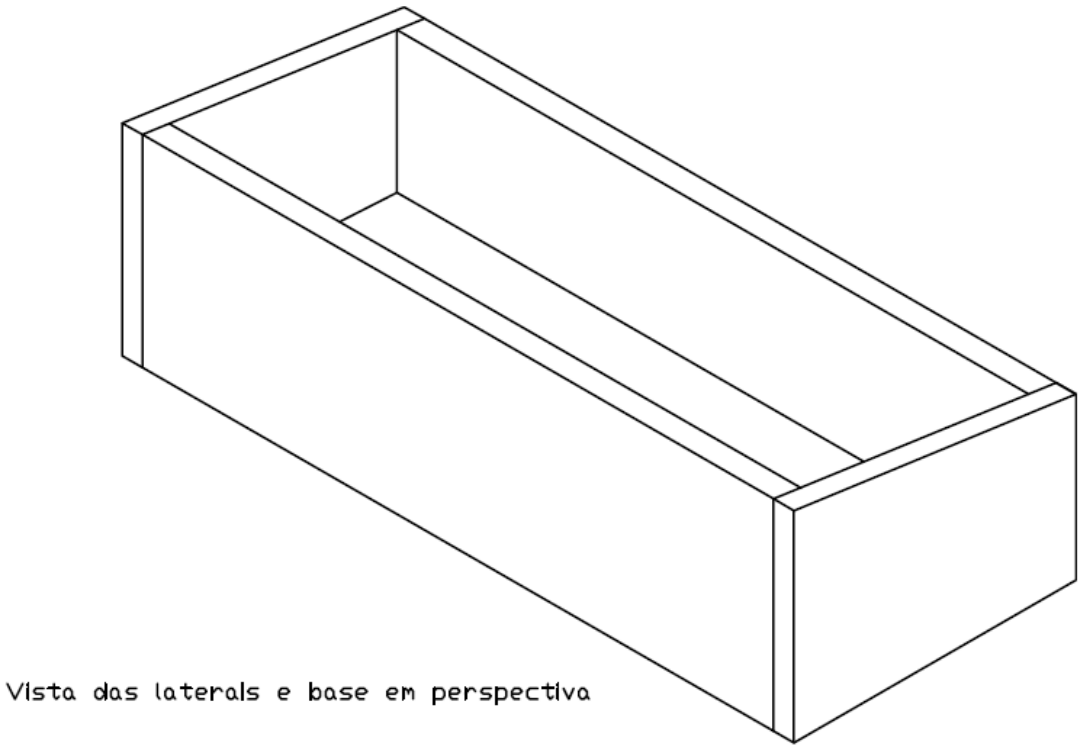
APÊNDICE E - Código-fonte do *software* do computador

O código-fonte do *software* do computador está disponível à banca de avaliação em um arquivo a parte deste. O código não constará nesse trabalho pelo fato de o produto desenvolvido ter potencialidades comerciais.

APÊNDICE F – Diagrama com as dimensões da caixa e tampa do controle arcade

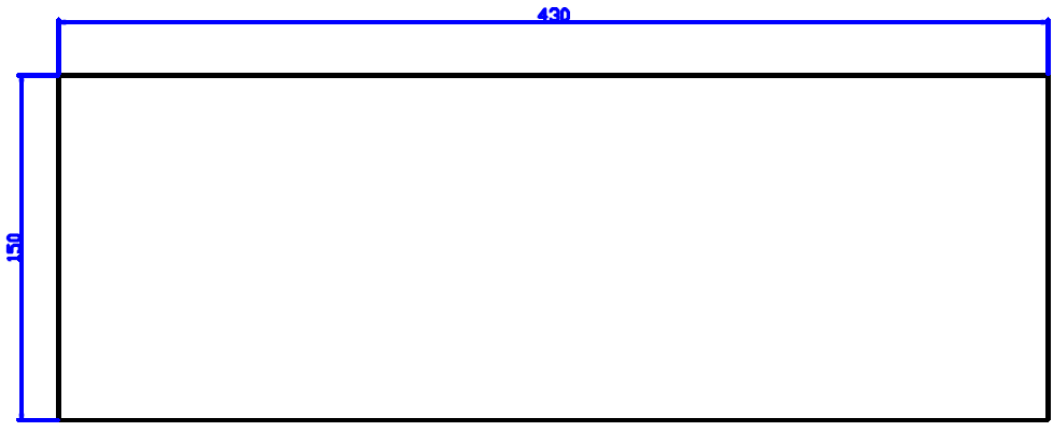


Visão do tampo furado para as chaves

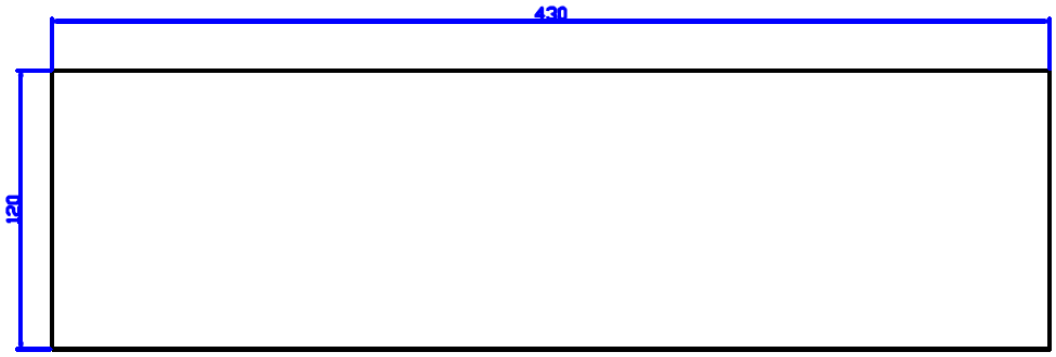


Vista das laterais e base em perspectiva

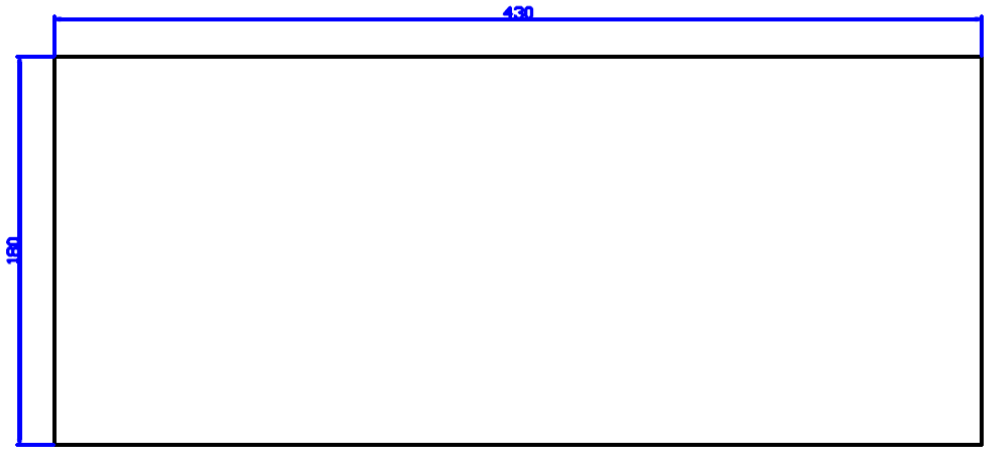
APÊNDICE F (CONTINUAÇÃO)



Frente

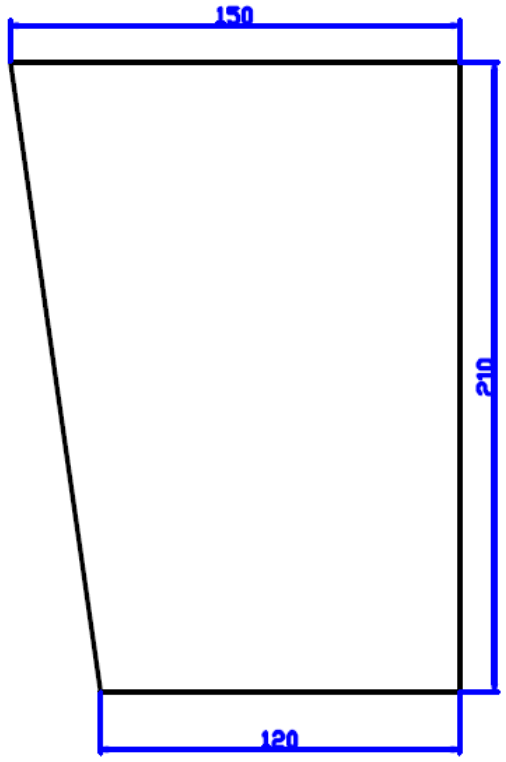


Costas



Fundo

APÊNNDICE F (CONTINUAÇÃO)



Laterais (2 peças)