ADALBERTO MENDES

IMPLEMENTAÇÃO DE UM JOGO DE TABULEIRO EM UM SISTEMA EMBARCADO PARA APOIO AO ENSINO DE MATEMÁTICA

LAVRAS - MG

2015

ADALBERTO MENDES

IMPLEMENTAÇÃO DE UM JOGO DE TABULEIRO EM UM SISTEMA EMBARCADO PARA APOIO AO ENSINO DE MATEMÁTICA

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, na área de concentração em Ciência da Computação, para a obtenção do título de Mestre.

Orientador

Prof. Dr. Wilian Soares Lacerda

LAVRAS - MG

2015

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Mendes, Adalberto .
Implementação de um jogo de tabuleiro em um sistema embarcado para apoio ao ensino de Matemática / Adalberto Mendes. – Lavras : UFLA, 2015.
135 p. : il.

Dissertação (mestrado acadêmico)–Universidade Federal de Lavras, 2015.
Orientador(a): Wilian Soares Lacerda.
Bibliografia.

1. Microcontrolador. 2. Projetos em Sistemas Embarcados. 3. Jogos Computacionais. 4. Educação Matemática. I. Universidade Federal de Lavras. II. Título.

ADALBERTO MENDES

IMPLEMENTAÇÃO DE UM JOGO DE TABULEIRO EM UM SISTEMA EMBARCADO PARA APOIO AO ENSINO DE MATEMÁTICA

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, na área de concentração em Ciência da Computação, para a obtenção do título de Mestre.

APROVADA em 23 de fevereiro de 2015.

Prof. Dr. Daniel Furtado Leite	UFLA
Profa. Dra. Regina Célia Grando	USF
Prof. Dr. Júlio Cezar David de Melo	UFMG

Prof. Dr. Wilian Soares Lacerda

Orientador

LAVRAS - MG

2015

Dedico este trabalho a Deus, minha esposa, meus filhos, meus pais e minha irmã. São eles a motivação de minha vida.

AGRADECIMENTOS

Agradeço a todos os membros docentes e discentes do Departamento de Ciência de Computação (DCC) da UFLA, onde fui muito bem recebido nestes dois anos em que estudei e trabalhei.

Ao meu orientador, Prof. Dr. Wilian Soares Lacerda que, com muita paciência, suportou minhas limitações e sempre esclareceu minhas dúvidas com muita parcimônia.

À minha eterna companheira e amada esposa, Rosana Maria Mendes que, com muito amor, me empurrou ao meio acadêmico e foi uma verdadeira incentivadora nos momentos mais difíceis.

A meus filhos, Asaph, Keyla e Sophia, que considero as alavancas para a minha vida ser o que é hoje.

A meu pai, Benedito, que me ensinou a só iniciar algo que possa terminar.

À minha mãe, Cecilia, que me ensinou a fazer tudo com muito carinho e amor.

À minha irmã, Marisa, que sempre me apoiou.

Quero agradecer especialmente a Deus, na pessoa de Jesus Cristo, por sempre apresentar uma boa oportunidade nos momentos em que nenhuma solução foi encontrada.

RESUMO

As dificuldades na utilização de sistemas computacionais em sala de aula, tanto por professores quanto por alunos, é tema bastante discutido por autores na área da Educação. Considerados sistema computacional, os chamados sistemas embarcados têm como uma de suas principais características a possibilidade de seu uso com resultados eficientes, mesmo que o utilizador não tenha grandes conhecimentos computacionais. Neste contexto, e aproveitando discussões sobre as potencialidades dos jogos eletrônicos na Educação Matemática, é apresentado neste trabalho o desenvolvimento de um sistema embarcado, instanciado num microcontrolador da família *Peripheral Interface Controller* (PIC) para um tabuleiro do jogo Contig 60®. A confecção dos requisitos funcionais e não funcionais, a partir das regras do jogo e dos estudos de Grando (2004) sobre o uso deste jogo como ferramenta em sala de aula, proporcionou a criação deste sistema, cujo objetivo geral é aprimorar essa ferramenta na utilização por professores no ensino de Matemática. Para alcançar este objetivo, foram pesquisados métodos em projetos de sistemas embarcados, técnicas de otimização de hardware, implementação de conceitos de Inteligência Artificial para a criação de um jogador autônomo e arquivamento em memória *flash* dos registros do jogo. Como resultado, foram feitos testes para validar este sistema baseando-se no documento de requisitos, em que cada item foi concluído com sucesso e devidamente justificado. Além deste objetivo imediato e já concluído, é esperado deste trabalho de pesquisa e do protótipo já pronto um exemplo que facilite a introdução de sistemas computacionais embarcados em sala de aula e proporcione pesquisas sobre sua utilização na Educação Básica.

Palavras-chave: Microcontrolador, Projetos em Sistemas Embarcados, Jogos Computacionais e Educação Matemática.

ABSTRACT

The difficulties on the use of computer systems in a classroom, for both teachers and students, have been widely discussed between authors in the education field. Being considered a computer system, the so-called embedded systems have as one of their main features the possibility of being effective even if the user is not proficient with computers. In this context, based on the discussions on the potential of the recreational aspect of electronic games for Education, this work presents the development of an embedded system, instantiated on a microcontroller of the PIC family (Peripheral Interface Controller) for the Contig 60® game board. A list of functional and non-functional requirements, based on the rules of the game and studies of Grando (2004) on the use of this game as a tool in the classroom, made the creation of this system possible. The main goal of the underlying system is to improve this tool so it can be used by Math teachers in their classes. To attain this goal, embedded systems design methods were researched. These include techniques of hardware optimization, implementation of Artificial Intelligence concepts for the creation of an autonomous player, and methods for saving games in flash memory. Tests were made to validate this system based on a list of requirements. The result show that the system well succeeded in attending such requirements. In the future, it is expected that the system developed helps in the introduction of embedded computational systems in class and helps new researches on the use of embedded systems in Basic Education.

Keywords: Microcontroller, Co-design of Embedded Systems, Computer Games and Mathematics Education.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

Lista de Algoritmos

LISTA DE SIGLAS

ADC	*Analog Digital Converter*
CAD	*Computer Aided Design*
CI	Circuito Integrado
CPU	*Central Processing Unit*
DCC	Departamento de Ciência da Computação
DEX	Departamento de Ciências Exatas
ESA	*Entertainment Software Association*
ESL	*Eletronic System-Level*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
HW	Hardware
FN	*Firmware*
IA	Inteligência Artificial
LAPd	Laboratório de Psicopedagogia
LED	*Light Emitting Diode*
LCD	*Liquid Crystal Display*
PIC	*Peripheral Interface Controller*
RAM	*Random Access Memory*
RISC	*Reduced Instruction Set Computer*
ROM	*Read Only Memory*
SdCard	*Secure Digital Card*
SE	Sistemas Embarcados
SPI	*Serial Peripheral Interface*
SoC	*System on a Chip*
SSL	*Solid Stat Lighting*
SW	software
PWM	*Pulse-width modulation*
TIC	Tecnologia da Informação e Comunicação
UML	*Unified Modeling Language*
UFLA	Universidade Federal de Lavras
USB	*Universal Serial Bus*
USP	Universidade de São Paulo
USART	*Universal Synchronous Asynchronous Receiver Transmitter*

SUMÁRIO

1 INTRODUÇÃO

Pesquisadores na área da educação têm discutido as dificuldades, as metodologias e as potencialidades das Tecnologias da Informação e Comunicação (TICs) para o processo de ensino e aprendizagem. Pesquisas realizadas por autores como Oliveira (2009), Mendes e Grando (2006), Miskulin et al. (2011), Marco (2013), entre outros, mostram que uma das possibilidades para a utilização das TICs, no contexto educacional, é o de jogos eletrônicos. Assim, vários teóricos ponderaram sobre a relação entre o jogo, a criança e a educação, tais como Grando (2000) e Mendes e Grando (2008), que apontaram para as potencialidades dos jogos para o processo de ensino e aprendizagem da Matemática.

Mendes e Grando (2008) afirmam que os jogos eletrônicos fazem parte da cultura lúdica de jovens e adolescentes. Como aponta Brougére (2002, p. 23), "a cultura lúdica é um conjunto de regras e significações próprias do jogo que o jogador adquire e domina no contexto de seu jogo". Dessa forma, espera-se o surgimento cada vez maior de pesquisas sobre os ambientes onde estes jogos estão inseridos, como computadores, *tablets* e sistemas embarcados.

A inserção de computadores nas empresas como ferramenta de trabalho impulsionou a indústria de hardware e software. Porém, o impulso transformador, tanto para a indústria como para a sociedade, aconteceu quando os jogos encontraram no computador um ambiente propício ao alcance de jovens e adolescentes. O pesquisador Aarseth (2001) escreve que o potencial da cultura de jogos de computador no futuro é imensurável. Ele tinha razão, pois, de acordo com a *Entertainment Software Association* (ESA), as vendas de jogos eletrônicos alcançaram US$ 22 bilhões, em 2008.

Naquele ano, nem o cinema faturou tanto no mundo[1].

Por outro lado, Almeida e Valente (2012) citam que o impacto das TICs, se comparado com o de outras ramificações da sociedade, não tem pleno aproveitamento de suas potencialidades na área da educação. Autores como Staa (2009), Mendes e Grando (2008) e Salgado et al. (2008) demonstraram que os recursos tecnológicos para as escolas, além de financeiramente custosos, costumam ficar confinados em laboratórios, sendo utilizados apenas em horários e dias específicos. Esses autores ainda discutem questões sobre a necessidade da criação de novas disciplinas para os alunos, o que necessariamente impõe, para a utilização destes recursos, horas em capacitação dos professores. A utilização de sistemas embarcados pode ser encarada como uma das possibilidades para a solução dos problemas relacionados com estas dificuldades, pois, além de custo reduzido, se comparados a computadores pessoais, têm como uma de suas principais características (BERGER, 2002) que o utilizador não necessita ter conhecimento computacional ou das tecnologias envolvidas na sua construção, mas apenas do problema no qual o sistema embarcado está inserido.

Nesta perspectiva e embasado pelos estudos de Grando (2004) e Silva (2009), neste trabalho abordam-se possíveis relações entre a Computação, mais especificamente a subárea de Sistemas Embarcados (SE) e sua possível aplicabilidade para a área de Educação Matemática, com a utilização de jogos eletrônicos no processo de ensino e aprendizagem. Para tanto, foi desenvolvido um protótipo de um sistema embarcado, instanciado num microcontrolador da família *Peripheral Interface Controller* (PIC) para um

[1]Informação obtida do site oficial da Entertainment Software Association (EAS). Disponível em:<http://www.theesa.com/newsroom/release_detail.asp?releaseID=44>. Acesso em: 20 fev. 2014

tabuleiro do jogo Contig 60®. Foram utilizadas técnicas de Inteligência Artificial (IA) para a criação de um jogador autômato, sendo os lances do jogo gravados em arquivo num cartão de memória *Secure Digital Card* (SdCard) e, para garantir custo reduzido, todo o projeto foi desenvolvido considerando técnicas de otimização de hardware.

1.1 Motivação

De acordo com Morales et al. (2013), anualmente são fabricados milhões de processadores destinados a sistemas embarcados. Além disso, o fato deste tipo de sistema estar presente em todas as áreas da sociedade mostra a necessidade crescente de pesquisas que tornem o processo de construção destes equipamentos mais eficiente. Na Figura 1 pode ser observado que o atraso no lançamento de um produto pode causar sérias reduções nos lucros de uma empresa. Este atraso pode ter diferentes motivos, entre eles as falhas na aplicação da metodologia escolhida, a falta de experiência dos projetistas e as escolhas erradas de ferramentas de simulação.

Uma pesquisa[2] realizada por um portal de conteúdos sobre desenvolvimento de sistemas embarcados, com o objetivo de traçar as principais tendências neste setor do mercado brasileiro, demonstrou que 61% dos projetos foram concluídos com algum atraso e que o maior desafio é o cumprimento dos prazos, além de indicar que 35% das empresas não usam nenhuma ferramenta de versionamento para apoiar o projeto. Estes dados justificam o estudo acadêmico mais aprofundado de técnicas e metodologias em projetos

[2]Pesquisa realizada em agosto de 2014, pelo portal http://www.embarcados.com.br, com 901 empresas participantes de todo o território nacional, por maio de questionários online. O relatório geral desta pesquisa está disponível em: http://www.embarcados.com.br/2014-brazilian-embedded-systems-development-market-study. Acesso em 29 out. 2014.

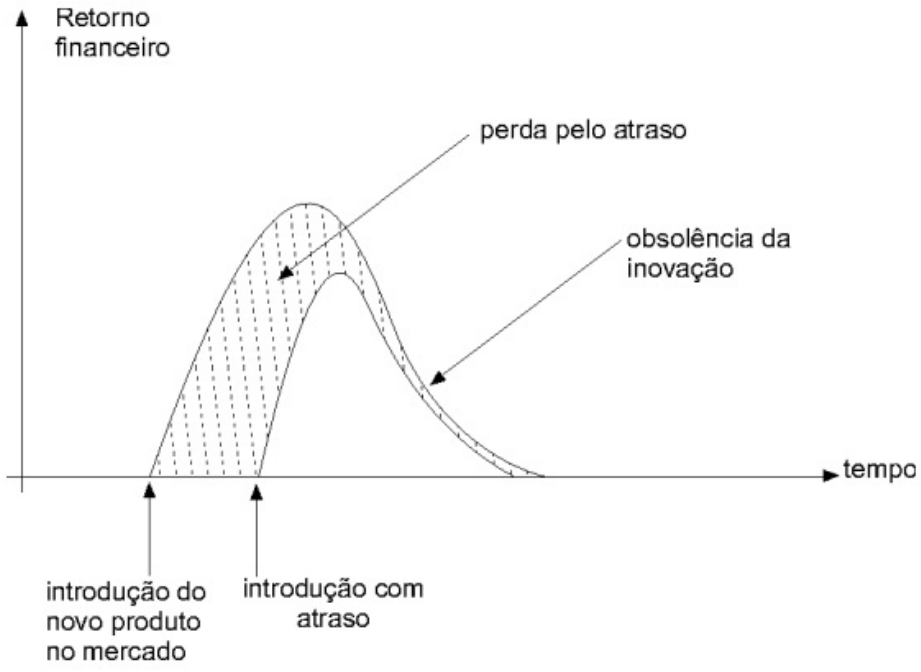


Figura 1 Gráfico de *time of market* de Carro e Wagner (2003)

de sistemas embarcados, principalmente em projetos de pequeno porte, nos quais custos reduzidos e agilidade são prioridades. Outro fator importante para a área da Computação é que a pesquisa realizada por Ossada e Martins (2010) demonstrou que, apesar de a maioria dos profissionais que trabalham na área ter formação em engenharia, nos últimos anos, os profissionais de TI têm migrado para a área de sistemas embarcados, o que também torna relevante esta pesquisa, que pode ser considerada um estudo de caso, como ferramenta para o planejamento da disciplina de Sistemas Embarcados. Em estudos como os de Wolf e Madsen (2000), Jackson (2005) e Koopman et al. (2005) cita-se a importância desta disciplina ter foco na capacitação em projetos multidisciplinares e não no ensino específico de um determinado microcontrolador.

1.2 Objetivo geral

Este trabalho foi realizado com o objetivo geral de desenvolver, em um sistema embarcado, um jogo de tabuleiro com custo reduzido e que seja robusto, versátil e eficiente. A utilização desse sistema deve auxiliar o professor em sala de aula para o ensino de Matemática.

1.3 Objetivos específicos

Os objetivos específicos foram:

1. priorizar soluções em software que otimizem o hardware e assim conseguir um sistema de custo reduzido;
2. implementar um agente autônomo que consiga jogar com competência;
3. armazenar cada jogada em arquivo de texto simples, num SdCard, para posterior análise pelos jogadores ou professor.

1.4 Estrutura deste trabalho

Este trabalho está estruturado em cinco capítulos. Encontra-se no **capítulo 2** o referencial teórico, onde são descritos os principais conceitos de software, hardware, as regras do jogo que foram implementadas e as técnicas escolhidas para a criação do agente autônomo. A metodologia utilizada é apresentada no **capítulo 3**, onde é descrito minunciosamente, todo o processo de construção do sistema. No **capítulo 4** estão relatados os resultados desta pesquisa e algumas discussões sobre o processo de construção e sua utilidade. As conclusões estão no **capítulo 5**, juntamente com algumas sugestões de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo tem início com uma explanação dos conceitos de sistemas embarcados e o seu desenvolvimento em projetos sob perspectivas que visam à otimização de hardware. Em seguida, conceituam-se algumas técnicas para o desenvolvimento de agentes autônomos em jogos de tabuleiro. Também são apresentadas as regras do jogo Contig 60® e sua utilização como ferramenta pedagógica para o ensino de Matemática em sala de aula.

2.1 Sistemas embarcados

Como explicado por Ottley (2007), sistemas embarcados são sistemas computacionais de pequeno porte, projetados para resolver um problema ou um pequeno conjunto de problemas dentro de um contexto, normalmente executando tarefas específicas. A Figura 2 apresenta uma típica generalização de um sistema embarcado em que é possível ver as variáveis de entrada que capturam informações do ambiente no qual o sistema está inserido. Estas informações, quando processadas no hardware pelo software, geram variáveis de saída que alteram o ambiente e podem gerar visualização de dados por meio da interface do usuário. Por sua vez, a interface pode ou não exigir confirmação ou alguma informação do utilizador para agir. Além disso, é comum sistemas embarcados exigirem ligações com outros tipos de sistemas computacionais.

Tarefas do cotidiano são auxiliadas por diversos equipamentos que utilizam sistemas embarcados. Na cozinha, os alimentos são aquecidos pelo forno de micro-ondas ou refrigerados pela geladeira, todos com muita precisão, graças ao controle que os sistemas embarcados proporcionam. Nos

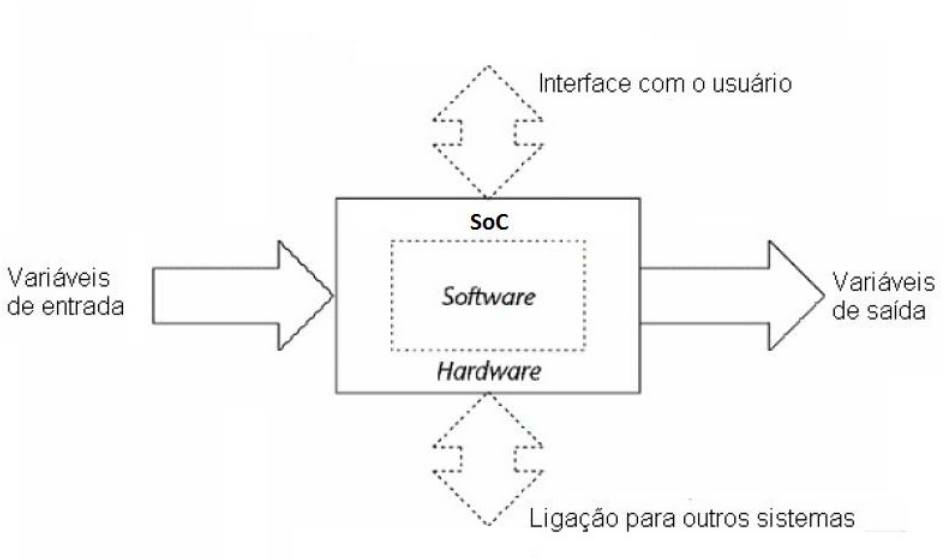


Figura 2 Generalização de sistema embarcado, adaptado de Wilmshurst (2009)

automóveis, os freios, a velocidade de cruzeiro, o consumo de combustível e muitas atividades que proporcionam segurança e conforto ao motorista e aos passageiros, são possíveis somente por utilizarem sistemas embarcados. Calculadoras, agendas eletrônicas, máquinas copiadoras, impressoras, ar condicionado e, de modo geral, a maioria dos equipamentos domésticos e de escritórios usam sistemas embarcados. Como exemplo, na Figura 3 mostra-se um refrigerador que tem, em seu interior, um microcontrolador, item indispensável para caracterizar um sistema embarcado.

A sociedade não seria o que é hoje sem estes equipamentos. Essa afirmação é confirmada por Morales et al. (2013) quando afirmam que sistemas embarcados estão presentes em 19% de todos os sistemas eletrônicos vendidos no mundo e este número poderá crescer para 33% até 2015. A popularização destes aparelhos advém, principalmente, do seu baixo custo e da

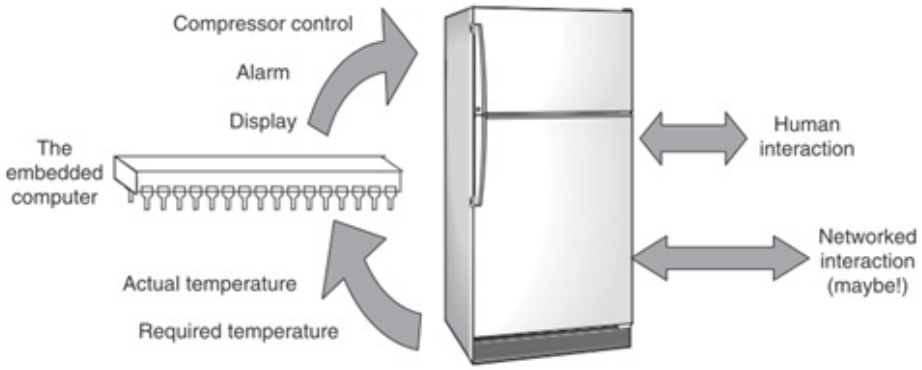


Figura 3 Exemplo de um sistema embarcado (WILMSHURST, 2009, p. 4)

simplicidade de utilização, não sendo necessário ter nenhum conhecimento na área de computação para utilizá-los. Além disso, outras características importantes são o baixo consumo de energia, a confiabilidade e o espaço.

Considerado um tipo de sistema computacional, os sistemas embarcados têm forte acoplamento entre hardware e software, o que torna difícil sua diferenciação de sistemas computacionais convencionais, como os *desktops*. Dessa forma, Berger (2002) prefere descrevê-los pelas características que os distinguem dos computadores pessoais, como as descritas a seguir:

1. São dedicados a tarefas específicas.

   No *desktop* é esperado que se execute simultaneamente, de forma rápida e eficiente, uma grande variedade de aplicativos; por outro lado, num sistema embarcado, é esperado que seja executada apenas uma ou poucas funções específicas.

2. Grande variedade de microcontroladores disponíveis.

   Mais de quarenta empresas no mundo disputam o mercado de micro-

controladores, entre elas a Microchip Technology Inc[3], uma das líderes deste segmento, que fabrica mais de quinhentos diferentes tipos de microcontroladores para diversos fins, alguns ao custo de menos de um dólar (ZURITA, 2011).

3. Consumo reduzido de energia.

   Considerando seu limitado poder computacional e o número mínimo de componentes, grande parte dos sistemas embarcados funciona com pouca energia, sendo suficiente para alimentá-los apenas algumas baterias.

4. Custos reduzidos.

   A grande diferença entre microprocessadores e microcontroladores é o fato de, no microcontrolador, os periféricos serem embutidos em um único *chip* (tecnologia *System on a Chip* ou SoC). Esta característica permite aos sistemas embarcados o uso muito reduzido de componentes discretos, se comparado aos *desktops*. Dessa forma, o impacto nos custos da adição de componentes é bem maior que em sistemas computacionais que têm centenas de componentes. É importante observar que a preocupação referente aos custos deve fazer parte, desde o início do projeto, de qualquer sistema embarcado, podendo ser fator decisivo no seu sucesso ou fracasso.

5. Recursos computacionais limitados.

   Processadores com múltiplos núcleos, frequência de clock acima dos 2 GHz, discos rígidos com capacidade acima de 1 *Terabyte*, barramento

[3]Estas e outras informações estão disponíveis no site <http://www.microchip.com>. Acessado em 12 jan. 2014.

de comunicação de alto desempenho e memórias RAM de alta capacidade não são realidades em sistemas embarcados. Na maioria dos microcontroladores a RAM não chega a 500 bytes e sem acesso a discos rígidos. A quantidade de teclas é limitada, chegando a acumular funções. A interface com o usuário, por vezes, é feita com pequenos *Liquid Crystal Displays* (LCD), display de 7 segmentos ou poucos *Light Emitting Diode* (LEDs), sendo preciso o uso de recursos como multiplexação, que aproveita a histerese (ou persistência) dos olhos do usuário para diminuir a quantidade de hardware requerido. Quanto ao código do software escrito, ele deve ser otimizado para ocupar poucos *bytes* e com o mínimo de recursos. Uma simples operação matemática pode consumir 1% ou 70% da memória em um dado microcontrolador, dependendo apenas da forma como foi escrito (ZURITA, 2011).

6. Requerem ferramentas e métodos especializados para serem projetados.

   Sendo sistemas embarcados compostos por hardware e software com limitações extremas e com forte acoplamento, exigem-se peculiaridades complexas em sua concepção, teste e depuração. Muitas vezes, é necessário projetar uma plataforma que emule o sistema ao qual o sistema embarcado será integrado. Por exemplo, não é aconselhável fazer o teste e a depuração de um sistema de controle de nível diretamente em uma plataforma de petróleo que esteja em alto mar.

7. Facilidade no uso.

   A interface de utilização de sistemas embarcados deve ser simples o bastante para que o utilizador não necessite de nenhum conhecimento

na área computacional, mas apenas do problema o qual o sistema se propõe a resolver.

2.1.1 Projetos em sistemas embarcados

Considerando o crescimento de mercado desse tipo de sistema computacional, suas características próprias e o fato de envolver diferentes especialidades, projetos em sistemas embarcados devem seguir moldes bem definidos, visando, principalmente, reduzir custos de projeto, produção e tempo de construção. De modo geral, aplicar uma metodologia de projeto com sucesso não significa que esta metodologia seja bem sucedida em qualquer projeto. Sendo assim, a definição de qual metodologia será adotada é a primeira e mais importante decisão a ser tomada no desenvolvimento de um sistema embarcado.

Metodologias como *bottom-up, top-down* e *meet-in-the-middle*, explicadas por Zurita (2011), seguem um fluxo bem definido de projeto. Nesta linha e de forma geral, na Figura 4 mostram-se as divisões básicas de um projeto em sistemas embarcados. As fases compreendem na especificação do produto ou a elicitação dos requisitos do sistema, o particionamento do projeto em componentes de hardware e software, a iteração e o refinamento do particionamento, as tarefas independentes de hardware e software, a interação das partes, os testes e a manutenção continua.

A seguir, cada fase da Figura 4 é descrita com mais detalhes:

1. Especificação do produto.

   Nesta etapa, o problema a ser resolvido deve ser bem definido, inicialmente descrito de maneira informal e gradativamente formalizado, com muito cuidado, em um documento de requisitos, muitas vezes

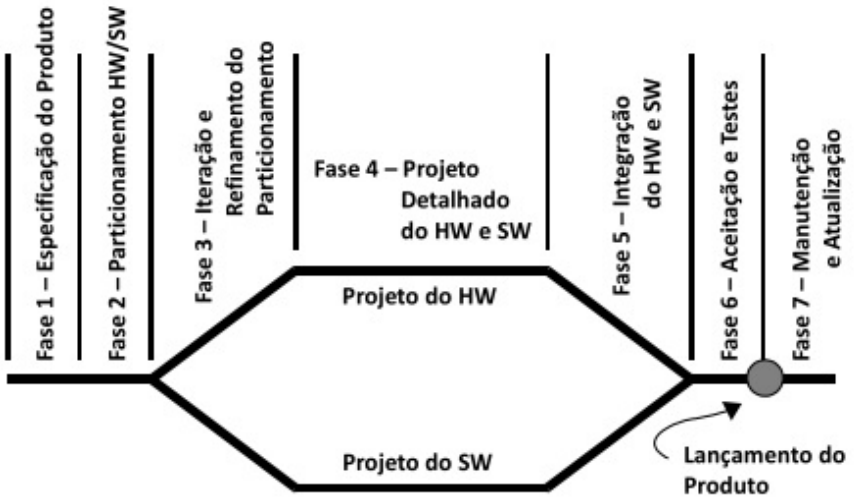


Figura 4 Fluxo de projeto em sistemas embarcados (BERGER, 2002, p. 17)

com a utilização de pessoal com competência na área de Engenharia de Requisitos. Além do documento de requisitos, quando necessário, esta fase pode gerar outros artefatos para auxiliar na continuidade do projeto, como, por exemplo, um diagrama de blocos especificando as entradas e as saídas do sistema ou, mesmo, em alguma linguagem de modelagem em alto nível, como a *Unified Modeling Language* (UML) (CARRO; WAGNER, 2003).

2. Particionamento do projeto em componentes de hardware e software.

   O momento agora é decidir, baseado nos documentos de requisitos, qual módulo será implementado em hardware e qual será em software. É importante observar que a decisão de implementar algum bloco em hardware, normalmente requer um custo maior, mas um ganho em

desempenho.

3. Iteração e refinamento do particionamento.

   Nesta fase são testados os blocos de hardware e software que foram definidos no item anterior, normalmente em ambientes de simulação, não havendo impedimento se, durante estes testes, forem propostas mudanças nestes blocos, sempre observando os requisitos não funcionais, como custos, desempenho e escalabilidade.

4. Tarefas independentes de hardware e software.

   Aqui são adquiridos os materiais e o hardware é desenvolvido, incluindo a confecção das Placas de Circuito Impresso (PCI) e a soldagem de componentes. O software também é desenvolvido em ambientes de programação e compilação.

5. Interação dos componentes de hardware e software.

   Nesta etapa costumam aparecer muitos problemas, principalmente os que podem não ser detectáveis em ambientes de simulação, como, por exemplo, qual o comportamento dos componentes de hardware em temperaturas de trabalho ou como os atrasos de propagação afetam as rotinas de software do sistema.

6. Testes e manutenção continua.

   Finalmente, testa-se de forma exaustiva o equipamento, passo a passo, seguindo o documento de requisitos. Se o equipamento passar nos testes, é inserido no ambiente do usuário final. É comum o utilizador reportar problemas no sistema. Na verdade, mais de 50% dos problemas ocorrem após a entrega do equipamento ao usuário e Ossada et

al. (2012) observaram que a maior parte desses problemas é devido aos equívocos na elaboração dos requisitos.

2.1.2 Tecnologia *System on a Chip* (SoC)

É fácil imaginar o elevado custo de um eletrodoméstico (por exemplo, uma geladeira), se este fosse controlado internamente por um computador que fosse fabricado com o mesmo conceito de um *desktop* ou mesmo um *tablet*. Além do custo alto, o desperdício de recursos computacionais e energia também deve ser considerado. Felizmente, o sistema computacional que controla esses equipamentos é fabricado com uma tecnologia que proporciona a produção com custo muito reduzido. A tecnologia SoC (*System on a Chip*) é baseada na ideia de embutir, num mesmo chip, todos os componentes de um computador e, dessa forma, conseguir um sistema computacional de menor custo, com apenas os recursos necessários para uma determinada função.

A estrutura básica de um SoC para um sistema embarcado é inserir, num único Circuito Integrado (CI), um sistema computacional completo (VARGAS et al., 2007). Sua estrutura típica contém um *Central Processing Unit* (CPU), memória *Random Access Memory* (RAM), *Read Only Memory* (ROM), *Electrically-Erasable Programmable Read-Only Memory* (EEPROM) e memória *flash*. Também são integrados aos SoCs módulos de temporização, sistemas de *clock*, portas de I/O, conversor analógico digital, entre outras possibilidades de periféricos. Também são acoplados padrões de comunicações como *Universal Serial Bus* (USB), *Universal Synchronous Asynchronous Receiver Transmitter* (USART), FireWire, Ethernet, *Serial Peripheral Interface* (SPI) e *Inter-Integrated Circuit* (I^2C).

Os componentes de hardware utilizados na construção de um microcontrolador, geralmente, são pré-validados, agrupados com o auxílio de ferramentas *Computer Aided Design* (CAD) e de bibliotecas que controlam sua operação. Dessa forma, existe uma variedade enorme de microcontroladores disponíveis para diferentes aplicações e com diversas características. Por exemplo, a empresa *Microchip Technology Inc* fabrica e comercializa microcontroladores da família *Programmable Interface Controller* (PIC). Na Figura 5 pode-se observar que essa arquitetura utiliza barramentos de dados e de programa de forma independente, e que o barramento de dados é menor que o barramento de programa, o que possibilita que a instrução seja empacotada em uma única palavra (*word*) do programa, melhorando a performance. Mais especificamente, o microcontrolador PIC18F4550, que, segundo sua ficha técnica (*datasheet*) da Microchip (2006), é baseado na arquitetura *Reduced Instruction Set Computer* (RISC), com recursos configuráveis para garantir seu uso em diferentes aplicações sem desperdício computacional. Sua velocidade de *clock* pode chegar a 48 MHz, com oite entradas analógicas, conversor analógico digital multiplexado de 10 bits e com pinos de transmissão e recepção compatíveis com o padrão de portas USB 2.0.

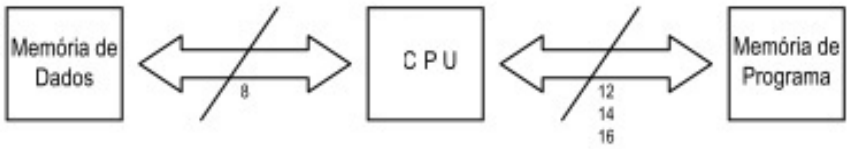


Figura 5 Arquitetura dos microcontroladores da família PIC, mostrada por Martins (2005, p. 18)

2.2 Interface com o usuário

Por vezes, a interface de um sistema embarcado é composta por conjuntos de *Light Emitting Diode* (LED), que devem ser controlados pelo microcontrolador. O LED é um diodo semicondutor (junção P-N) que, quando energizado de forma correta, emite luz visível. Atualmente, um LED emite, em média, 100 *lumens* de fluxo luminoso. Um dos problemas desse tipo de interface consiste na quantidade de saídas de que um microcontrolador deve dispor para controlar a quantidade necessária de LEDs para transmitir informações ao usuário. A ilustração da Figura 6 mostra que seriam necessários oito pinos para controlar oito LEDs, o que torna essa técnica inviável para quantidades maiores de LEDs.

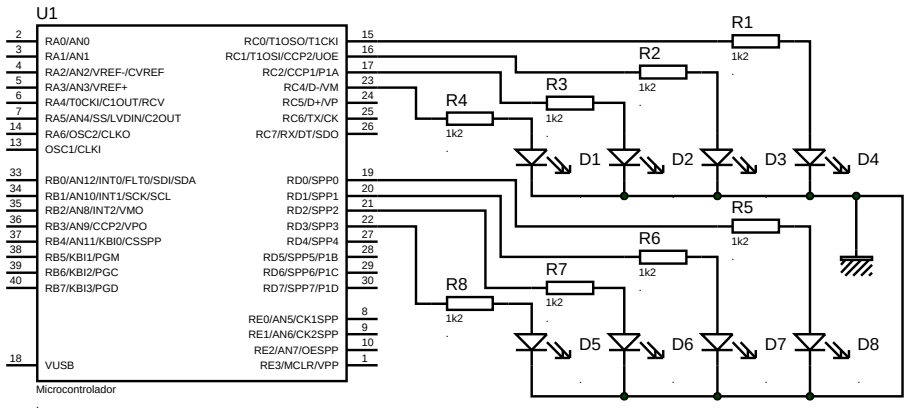


Figura 6 Controle de 8 LEDs com 8 pinos do Microcontrolador

Uma proposta desenvolvida por Charlie Allen, em 1955, chamada *charlieplexing* (MARGOLIS, 2012), consiste no posicionamento paralelo e invertido de LEDs. O terminal catodo do LED D1 é ligado ao terminal anodo do LED D2 e o anodo de D1 é ligado ao catodo de D2. Ligando o

anodo e o catodo de D1 aos pinos de saída do microcontrolador pode-se controlar quando cada LED irá acender, bastando, para isso, inverter os sinais dos pinos do microcontrolador. Este cenário, com apenas dois LEDs, parece não resolver o problema, mas, verificando a Figura 7, é possível perceber uma relação diferente entre a quantidade de LEDs e de pinos do microcontrolador. Pode-se controlar n * (n-1) LEDs, em que n é a quantidade de pinos.

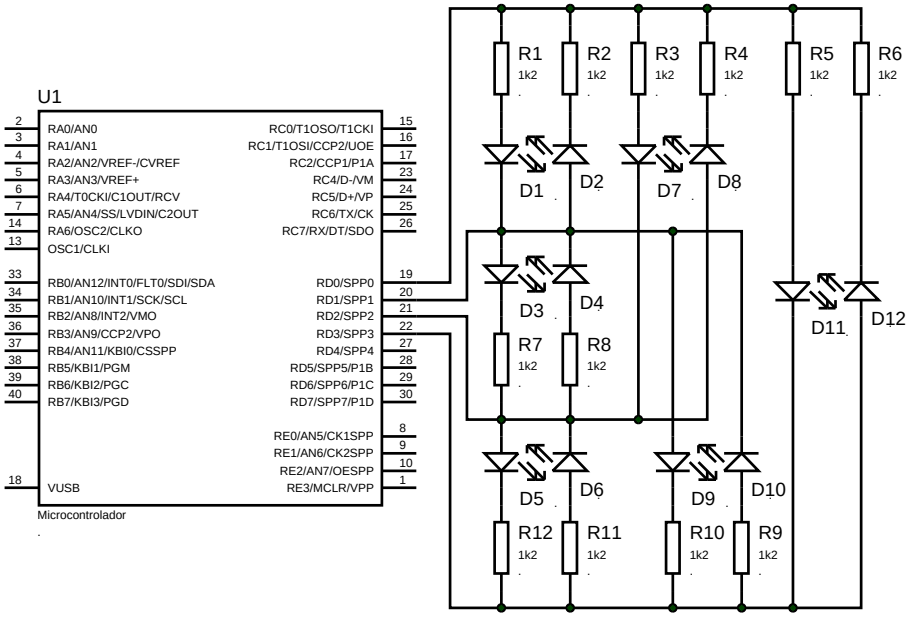


Figura 7 Controle de LEDs usando o método Charliplex

Na tentativa de eliminar a relação entre LEDs e pinos de controle do microcontrolador, a técnica de multiplexação é apresentada como uma ótima solução. O conceito aproveita um fenômeno chamado persistência retiniana, isto é, um objeto (por exemplo, um LED aceso) visto pelo olho humano persiste na retina por uma fração de segundos, mesmo quando

a luz do LED já não existe, dando a impressão de que ainda está aceso. Dessa forma, ao ser aplicado um pulso modulado por largura (do inglês *Pulse-width modulation* (PWM)), de frequência acima dos 100 Hz, é possível criar a ilusão de este LED estar permanentemente acesso. Nesse sentido, a intensidade de luz emitida pelo LED é controlada pelo tempo de ciclo ativo do sinal, ou *duty cycle* do sinal PWM. Na Figura 8 é mostrado um sinal PWM, em que ***a*** representa a amplitude do pulso que, normalmente, é fixa, sendo ***f*** a frequência do sinal, que deve ser acima dos 100 Hz e ***dc*** é o *duty cycle*, que controla a luminosidade emitida pelo LED.

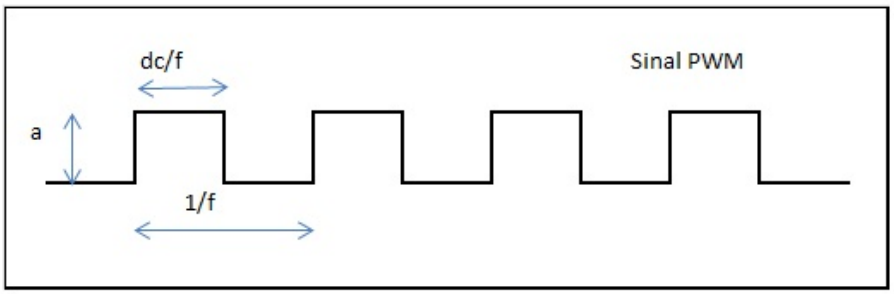


Figura 8 Sinal PWM

A utilização desta técnica, dependendo da quantidade de LEDs e da ampla gama de luminosidade que um painel de interface exige, torna a calibração de tal sistema bastante complexa (YANG; BERGMANS; SCHENK, 2009). Na Figura 9 mostra-se um diagrama genérico de um microcontrolador ligado a dois CIs, que fazem o papel de registrador de deslocamento, com entrada serial e saída paralela. Nesta configuração, são utilizados um sinal serial que entra no pino DS do registrador, um sinal de *clock* (SH_CP) e o *latch* (ST_CP). O pino Q7 é usado para enviar os dados seriais para outro registrador, tornando possível sua ligação em cascata. Apesar de essa

técnica acrescentar alguns componentes de hardware e tornar o projeto de software bastante sofisticado, resolve o problema da relação entre LEDs e pinos do microcontrolador.

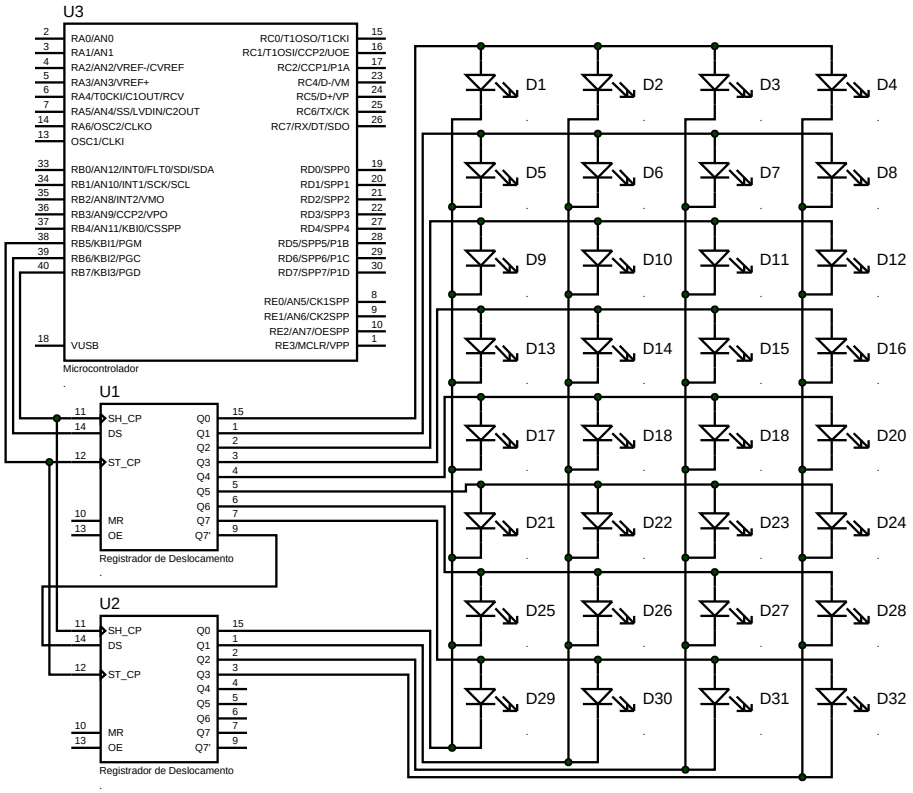


Figura 9 Esquema de multiplexação para controle de LEDs (os resistores para limitar a corrente dos LEDs foram omitidos)

2.3 Agentes autônomos

Definir um agente autônomo não é tarefa trivial, principalmente porque existe uma série de áreas do conhecimento, como Sociologia, Engenharia de Software, Teoria dos Jogos e outras, que o emprega para diferentes situa-

ções. Na área de Inteligência Artificial, Castro Neto e Julia (2007) definem que um agente autônomo pode ser uma entidade que age em um mundo de acordo com seus objetivos, percepções e o estado atual do seu conhecimento. Matematicamente pode-se afirmar que o comportamento do agente é descrito pela função de avaliação do agente que mapeia qualquer sequência de percepções específicas do ambiente.

Para restringir ainda mais essa definição, é possível, de acordo com Wooldridge e Jennings (1995), definir agentes autônomo como sendo entidades de software que devem possuir um mínimo de propriedades que são:

1. autonomia: suas ações independem de intervenções de outros agentes, sejam eles humanos ou computacionais. Isto é, têm total controle sobre suas decisões e possíveis ações;

2. sociabilidade: devem interagir suas decisões com outros agentes, por meio de algum tipo de interface;

3. capacidade de reação: devem reagir em tempo hábil às mudanças do ambiente no qual estão inseridos;

4. proatividade: os agentes devem mostrar reação não apenas movidos pelas alterações do ambiente inserido, mas também para alcançar objetivos predefinidos.

A Figura 10 representa o esquema básico de um agente autônomo. O agente tem percepções vindas do ambiente no qual está inserido. Estas percepções ativam uma função de avaliação, a qual toma decisões para possíveis ações que afetam ou modificam o ambiente e, portanto, auxilia o agente a alcançar suas metas.

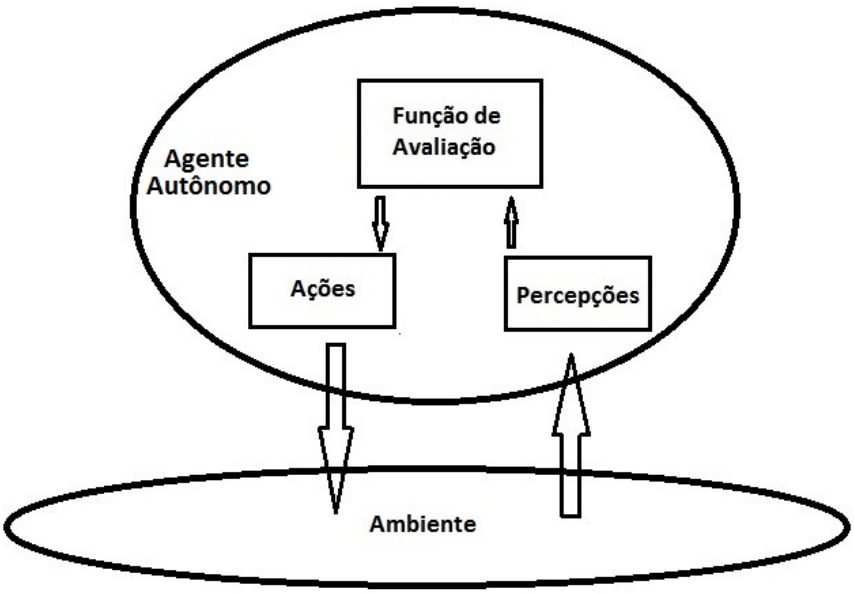


Figura 10 Diagrama básico de um agente autônomo

2.3.1 Estratégia de busca *Minimax*

Busca *Minimax* é uma técnica de Inteligência Artificial cujo procedimento é baseado numa árvore de busca em profundidade, em que a ideia é, com base na posição corrente (nó raiz) do jogo, usar um gerador de movimento plausível para gerar o conjunto de possíveis posições sucessoras. Para atribuir um custo ou peso a cada posição sucessora gerada, é aplicada uma função de avaliação a cada posição. Como resultado, tem-se a posição sucessora mais vantajosa para aquele momento do jogo.

Os jogos, principalmente os de tabuleiro, como damas, xadrez e gamão, entre outros, além de apresentarem um ambiente lúdico, têm todas as características de um domínio para experimentação de técnicas de Inteligên-

cia Artificial. A contribuição de estudos de Inteligência Artificial aplicada em jogos tem sua importância nos diversos aspectos envolvidos nas atividades inteligentes em ambientes computacionais, como memória, estratégia, conhecimento *ad hoc*, etc. (BITTENCOURT, 2001). O ambiente de jogos de tabuleiro traz um tipo especial de problema em que se pode aplicar o algoritmo *Minimax*. Como explicado por Russell e Norvig (2003), é um tipo de técnica que busca determinar uma estratégia ótima em um cenário de jogo com dois jogadores. Como objetivo, busca-se decidir qual a melhor jogada para um dado estado do jogo. Assim, há dois jogadores no *Minimax*, o MAX e o MIN. Cada lance é composto por duas jogadas alternadas, primeiro MAX depois MIN. A busca é feita para determinar a estratégia ótima para MAX, ou seja, MAX é o jogador que tenta maximizar o ganho em suas jogadas, enquanto MIN tenta minimizar os ganhos de MAX. Este conceito aparece em muitas situações de diversos jogos de tabuleiro, em que a jogada de um adversário tenta ser minimizada por meio de jogadas subsequentes que neutralizem seus ganhos.

Tomando por base a Figura 11, que mostra um exemplo de árvore binária, é possível definir alguns conceitos. O espaço de busca da árvore é constituído de um conjunto de nós em que têm-se as informações **(A, B, C, D, E, F)**. Estes nós são conectados por arcos que apontam para outros nós que ficam à sua direita e à sua esquerda, chamados de nós filhos. Estes arcos podem ou não estar associados a pesos que representam o custo de transição entre nós. Por exemplo, a informação que está no nó Raiz é **A**; o nó direito a este, chamado nó filho, tem a informação **C** e o nó esquerdo, também chamado nó filho, tem a informação **B**. Se um nó não tem nenhum arco de transição abaixo dele, isto é, não tem nós filhos, este é chamado de nó folha,

como os nós **D**, **E** e **F**, que têm os pesos para chegar neles, respectivamente, 8, 3 e -1. A aridade, ou grau, do nó é definida pela quantidade de nós filhos que este tem. A aridade, ou grau, de uma árvore é a maior aridade ou grau entre os nós da árvore. Por exemplo, a aridade ou grau da árvore da Figura 11 é dois. A cada nó tem-se associada uma profundidade e está tem valor 0 no nó raiz e aumenta uma unidade para um nó filho. O objetivo desta busca em árvore é encontrar um caminho ótimo (menor custo) entre o estado inicial (nó Raiz) e o estado final (nó folha), que representa a solução para o problema (TOSCANO, 2009).

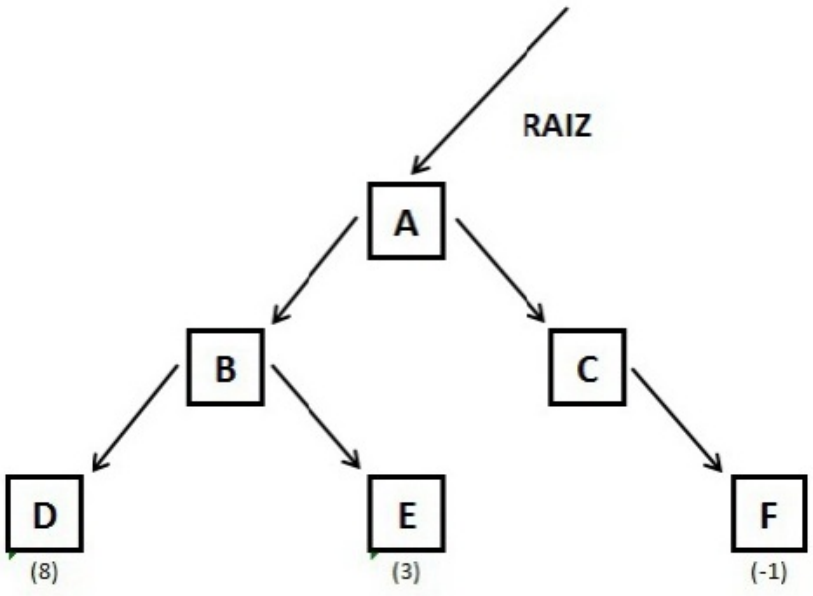


Figura 11 Exemplo de árvore binária

No Algoritmo 1 o nó raiz é a posição corrente do jogo. Se a profundidade da árvore for diferente de zero, as folhas desta árvore são avaliadas por uma função heurística predefinida pela ótica do jogador

MAX. Os valores dos nós são atribuídos de baixo para cima com estas avaliações. As folhas no nível do jogador MIN são preenchidas com o menor valor de todos seus nós filhos e o nível do jogador MAX é preenchido com o maior valor de todos os nós filhos.

Algoritmo 1: *Minimax* de Russell e Norvig (2003, p. 77)

```
Data: MiniMax
1  begin
2      Minimax (Nó, Profundidade);
3      if Nó é um nó terminal OR Profundidade == 0 then
4          return valor calculado do nó;
5      else if Profundidade limite then
6          Calcular valor aplicando a função heurística;
7          return valor calculado do nó;
8      else if Nível == MAX then
9          Aplicar Minimax aos sucessores;
10         return valor Max;
11     else if Nível == MIN then
12         Aplicar Minimax aos sucessores;
13         return valor Min;
```

2.4 O jogo Contig60®

O estudo de Grando (2000) apresenta o jogo[4] Contig 60®, criado pelo Dr. John C. Del Regato, Copyright 1980, 1986 e pertencente ao *Pentathlon Institute, inc.*. Na Figura 12 mostra-se o tabuleiro do jogo, com

[4]Encontram-se no site do *Pentathlon Institute* informações sobre o programa *Mathematics Pentathlon* e as regras oficiais do jogo, estudos sobre sua potencialidade na utilização para o ensino da Matemática, torneios realizados, eventos e muito mais. Disponível em: http://www.mathpentath.org/default.htm (último acesso em 21 de agosto de 2014) e também no site http://mathinspirations.com/2014/05/22/contig-60/. Acesso em: 05 jul. 2014.

casas numeradas. Cada jogador tem 25 fichas que têm cores diferentes das fichas de seu adversário. As quatro operações básicas (adição, subtração, divisão e multiplicação) estão presentes em toda a estrutura do jogo, fazendo com que seja uma ótima ferramenta para o ensino de conceitos fundamentais de Matemática. No jogo também está presente o fator aleatório (sorte), pois, com três dados, se obtêm os números para aplicá-los numa expressão numérica, sendo que o resultado deve coincidir com o valor de uma das casas do tabuleiro. Com a possibilidade de conseguir diferentes valores como resposta, um dos objetivos é preencher as casas do tabuleiro até conquistar uma sequência de cinco casas na vertical, na horizontal ou na diagonal.

Contig 60 Game Board

1	2	3	4	5	6	7	8
28	29	30	31	32	33	34	9
27	55	60	64	66	72	35	10
26	54	125	144	150	75	36	11
25	50	120	216	180	80	37	12
24	48	108	100	96	90	38	13
23	45	44	42	41	40	39	14
22	21	20	19	18	17	16	15

Figura 12 Exemplo de tabuleiro do do jogo Contig 60® (STOFFEL; VITÓRIA; SILVA, 2014, p. 5)

De acordo com Grando (2004), é muito comum, em sala de aulas tradicionais, apresentar aos alunos valores numéricos aos quais deve ser aplicada alguma operação matemática, em que este simplesmente calcula o resultado. No jogo Contig60® a ordem é inversa. Os resultados possíveis estão evidenciados no tabuleiro e os números a serem operados são sorteados. O que o jogador precisa é pensar qual sentença matemática deve fazer para chegar ao resultado desejado. O que torna o jogo Contig60® um jogo de estratégia é o fato de o jogador ter que coordenar as duas formas distintas de vencer o jogo (alcançar os pontos definidos no início do jogo ou conseguir fazer uma sequência de cinco casas preenchidas com suas fichas) e o inverso disto, que é impedir o oponente de alcançar estes objetivos.

2.4.1 As regras do jogo Contig60®

Material: tabuleiro, 50 fichas, sendo 25 de uma cor e 25 de outra cor e 3 dados.

Objetivo: o jogador deve conquistar uma sequência de cinco fichas nas casas do tabuleiro, na horizontal, na vertical ou na diagonal. Outra forma de vencer é o jogador ser o primeiro a alcançar o número de pontos que foi definido no início do jogo (entre 30 e 60 pontos).

Regras:

1. Os jogadores ou times de jogadores jogam de forma alternada. Cada um, em sua vez, lança os dados e, com os números, deve fazer uma sentença matemática, usando as quatro operações. Por exemplo, se os dados lançados derem os números 3, 1 e 5, o jogador poderá construir a sentença (3 + 1) x 5 = 20. Neste caso, o jogador deverá colocar sua ficha na casa marcada com o número 20 do tabuleiro.

2. No início do jogo, os jogadores devem definir um número de pontos que devem ser alcançados, normalmente entre 30 e 60 pontos. Os pontos são conquistados de acordo com a casa que o jogador escolhe no tabuleiro. Cada casa adjacente à que foi escolhida equivale a um ponto, que deve ser somado com os pontos adquiridos a cada jogada.

3. Se um jogador efetuar uma operação errada, o adversário pode acusar o erro. Acontecendo isto, o jogador que cometeu o erro perde os pontos que adquiriu de seu lance, mais dois pontos de penalidade. A partida continua sem que o jogador penalizado repita o lance.

4. Se um jogador acreditar que com os números sorteados pelos dados é impossível fazer uma sentença com resultado que coincida com uma das casas vagas do tabuleiro, este abre mão de sua vez. Nesse caso, se o adversário conseguir fazer uma operação com resultado válido, ele ganha o dobro dos pontos que normalmente ganharia e continua a partida, sendo sua vez de jogar os dados.

5. O jogo é encerrado quando um dos jogadores alcança o valor de pontos definidos no item 1. Outra forma de ganhar o jogo é conquistando cinco casas em linha reta no tabuleiro, podendo ser na horizontal, na vertical ou na diagonal.

2.4.2 Momentos do jogo

O termo **momentos de jogo**, utilizados nos estudos de Grando (2000), é considerado uma metodologia eficiente para potencializar o uso de jogos em sala de aula, nas atividades de intervenção pedagógicas.

Os momentos de jogos são:

1. Familiarização com o material do jogo: é o primeiro contato com os materiais do jogo. Neste momento, é comum o aluno criar relações com jogos conhecidos.

2. Reconhecimento das regras do jogo: o professor deve explicar as regras do jogo ao aluno. Pode apresentar as regras de forma escrita e explicitar os pontos importantes. Pode também fazer alguns jogos testes, mostrando algumas estratégias.

3. O jogo pelo jogo: neste momento, o importante é garantir que o aluno assimile com consistência as regras do jogo. O professor pode também explorar os conceitos matemáticos do jogo.

4. Intervenção pedagógica verbal: depois que o aluno conhece bem as regras do jogo, é o momento que deve-se pensar nas estratégias que o levarão a ganhar o jogo (previsão de jogadas, melhores e piores jogadas, etc.). O professor pode se ater aos procedimentos criados pelo aluno na solução do problema e buscar relacioná-los a conceitos matemáticos.

5. Registro do jogo: a importância de registrar cada lance do jogo está no fato de conseguir melhorar sua competência como jogador. Para o professor, que usa jogos para ensinar conceitos matemáticos, a importância dos registros está na possibilidade de identificar se os objetivos didáticos foram alcançados.

6. Intervenção pedagógica escrita: aqui são tratados problemas específicos do jogo. O professor pode criar situações de jogo onde os alunos irão treinar suas estratégias e, paralelamente a isso, o professor pode

tratar as falhas de conceitos que foram identificadas no momento anterior.

7. Jogar com competência: neste momento, o aluno pode colocar em prática todos os conhecimentos adquiridos durante os outros momentos, para tentar vencer o adversário.

Assim, os momentos de jogo são ferramentas de grande importância para a discussão do potencial pedagógico do jogo no contexto educacional e sua utilização em sala de aula (GRANDO, 2004).

3 METODOLOGIA

A intenção deste capitulo é descrever, com considerável nível de detalhes, a metodologia aplicada neste trabalho de pesquisa. O objetivo geral é a construção de um tabuleiro do jogo Contig 60® num sistema embarcado, sendo as seguintes subseções divididas em fases, necessárias para a construção desse sistema. A primeira subseção consiste na escolha da metodologia adotada. Em seguida são definidos os requisitos mínimos que o sistema deve ter para alcançar o objetivo geral. As fases seguintes, de acordo com a metodologia adotada, descrevem as técnicas empregadas na confecção do hardware e do software, e, por fim, como foi feita a integração entre eles.

3.1 Metodologia *meet-in-the-middle*

O desenvolvimento de projetos em sistemas embarcados tem se tornado cada vez mais complexo, principalmente pelos avanços conquistados pela indústria de semicondutores que, atualmente, consegue embutir, em apenas uma pastilha de silício de pequena dimensão, enormes quantidades de periféricos. Também devem ser consideradas as exigências referentes à janela de mercado (*time-to-market*), que é o tempo entre a identificação de uma oportunidade e o sistema pronto. Somente estes fatores justificariam a alta demanda de estudos sobre metodologias e técnicas para o desenvolvimento de sistemas embarcados. Além disso, segundo Wolf (2012), é importante seguir uma metodologia precisa, de forma a garantir o cumprimento dos requisitos do sistema de maneira formal, garantindo o uso eficiente de ferramentas de automatização que facilitem a comunicação entre equipes.

De acordo com Edwards et al. (2003), toda metodologia deve cum-

prir, durante o projeto, critérios de modelagem, validação e síntese. Assim, para este trabalho, foi escolhido o uso de uma metodologia baseada em plataforma, adaptada da proposta de Keutzer et al. (2000), que, de maneira geral, tem como fluxo de trabalho uma abordagem *meet-in-the-middle*, também citada por Mück (2013) e auxiliada pelo diagrama em Y da Figura 13, proposta por Gajski (1994) para projetos de circuitos digitais.

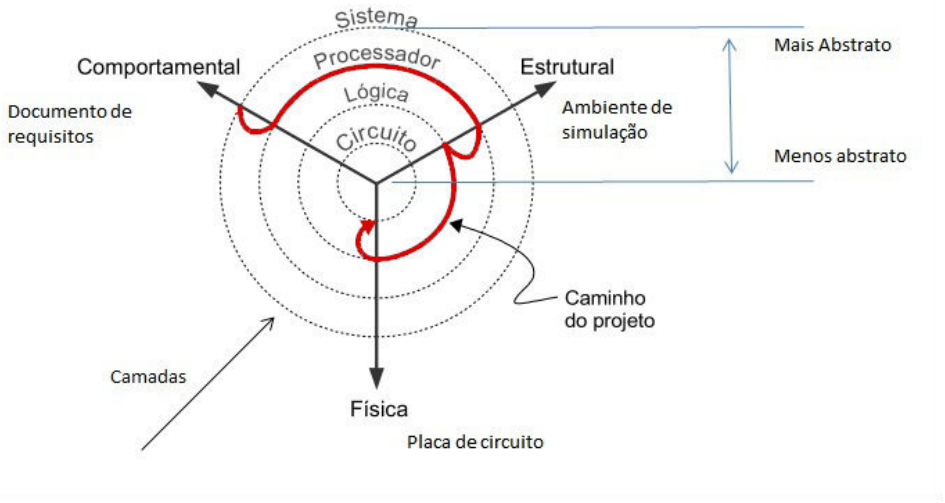


Figura 13 Diagrama em Y de Gajski, adaptado de Calazans (1995, p. 15)

As próximas subseções seguem o caminho do projeto mostrado no diagrama da Figura 13, que tem início na subseção 3.2, com um nível de abstração alto.

3.2 Dimensão comportamental

Esta subseção é o início do caminho do projeto do diagrama da Figura 13 e indicado como a primeira fase na Figura 14. O projeto é iniciado

pelo eixo Comportamental, na camada de Sistema, em que é feito um estudo do estado da arte sobre a utilização de jogos como ferramenta para auxiliar o professor no ensino de conceitos matemáticos. Estudos como os de Grando (2000), Grando (2004), Silva (2009), Silva e Silva (2009), Matos, Vailant e Lima (2012) e Morbach (2012) apresentam o jogo Contig 60® e seu uso como instrumento pedagógico, o que inclui pesquisas sobre os métodos para potencializar sua utilização como ferramenta no ensino da Matemática, como, por exemplo, o conceito dos momentos de jogo, apresentado na subseção 2.4.2.

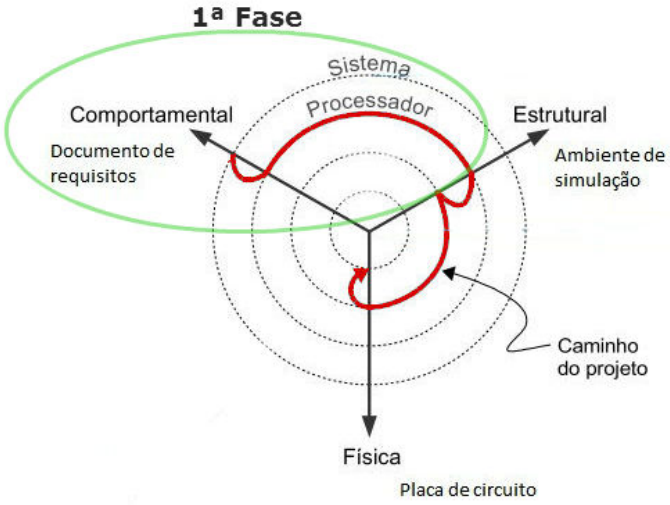


Figura 14 Diagrama em Y de Gajski, com enfase na primeira fase

Com a escolha do jogo Contig 60®, é criado o documento de requisitos do sistema (Apêndice E) sobre o qual, e de forma abstrata, é estudada a viabilidade da construção de um sistema embarcado que seja baseado nos momentos de jogo, propostos por Grando (2004), em que o foco principal

seja auxiliar o professor em seus objetivos no uso deste jogo. Dessa forma, são descritas as características do sistema, além dos requisitos funcionais e os não funcionais.

Nesta fase, cogitou-se a utilização de um *tablet* para a implementação do jogo Contig 60®. O primeiro obstaculo está no preço do *tablet* que, atualmente, no Brasil, é da ordem de R$300,00. O custo do protótipo foi estimado, no documento de requisitos, em um valor bem menor, o que pode ser reduzido duas ou três vezes quando as peças são adquiridas em quantidades maiores, numa linha de produção. Outra justificativa para a construção de um equipamento cuja interface com o usuário seja maior que a tela do *tablet* é que, de acordo com Grando (2004), o aproveitamento do lúdico no contexto educacional é potencializado pela interação entre os jogadores e, ainda maior, quando a partida é formada por vários jogadores. Por exemplo, o jogador **A** pode representar um time de dois ou três jogadores. A mesa digital ou a lousa digital, pelo seu tamanho, têm grande potencial para aplicação de jogos em sala de aula. Estudos como os de Alofs, Theune e Swartjes (2012), Nakashima e Amaral (2012) e Wu e Balakrihnan (2003) remetem a este tema, explicitando as dificuldades referentes ao alto custo, o tamanho necessário para a aplicabilidade do sistema e os problemas com interatividade. Dessa forma, para este trabalho de pesquisa, o *tablet* foi considerado inadequado, sendo um equipamento de custo elevado, pequeno e de uso individual.

Ainda no eixo Comportamental, mas agora na camada de Processador, são elaborados, com base no documento de requisito e de forma abstrata, os componentes em blocos do sistema mostrado na Figura 15, que são:

1. Entradas e saídas necessárias para a realização do projeto, mostrado nos blocos do microcontrolador, que tenham características suficientes para comportar a aplicação.

2. O teclado para que o jogador insira as operações matemáticas ou outra informação pertinente.

3. O display de LCD para que o sistema interaja com o usuário.

4. Um conversor série-paralelo para enviar os sinais elétricos para os LEDs, que estão em forma matricial, e, assim, representar as casas dos tabuleiro;

5. Uma porta USB ou dispositivo SdCard, para gravar em memória ou em qualquer dispositivo computacional os lances do jogo.

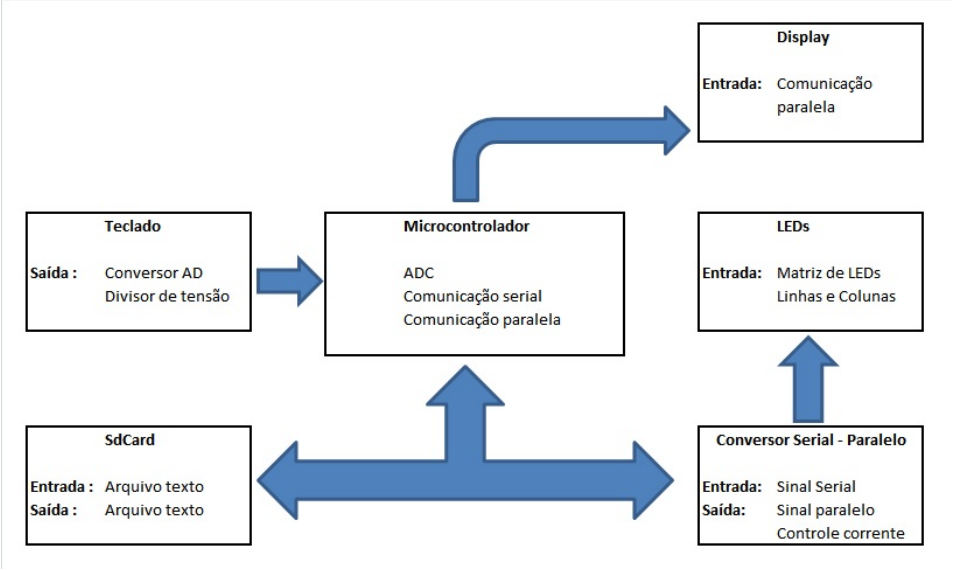


Figura 15 Diagrama de blocos do sistema

Com o Diagrama de Blocos (Figura 15) e o documento de requisitos, é possível seguir o caminho do projeto do Diagrama em Y de Gajski na Figura 13, que vai para um nível de abstração um pouco menor. O caminho do projeto sai da fase Comportamental (Subseção 3.2) para a fase Estrutural (Subseção3.3), detalhando as entradas e as saídas dos blocos do sistema e vinculando-os com as funcionalidades do documento de requisitos. Nesta fase também são definidos as ferramentas de simulação, a validação e o ambiente de programação.

3.3 Dimensão estrutural

Nas próximas subseções, o caminho do projeto vai para a segunda fase, mostrada na Figura 16, em que o eixo Estrutural da camada de Processador leva o projeto para um nível menos abstrato, em que são descritos com mais detalhes os principais componentes de hardware dos blocos do sistema e as técnicas aplicadas para otimização deste (subseções 3.3.1 a 3.3.5). Em seguida, ainda no eixo Estrutural, mas agora na camada Lógica, são definidos o ambiente de simulação, a linguagem de programação e o compilador (subseções 3.3.6 a 3.3.8). Nesta fase, a separação entre hardware e software fica mais evidente.

3.3.1 Microcontrolador

Descartando o *tablet* como base para este trabalho, foram pesquisados os possíveis microcontroladores, de diferentes marcas, famílias e características, tornando a escolha uma tarefa difícil. É importante observar que, nesta fase, uma decisão errada pode inviabilizar todo o projeto. Sendo que a primeira diretiva para auxiliar na decisão da escolha do microcontrolador

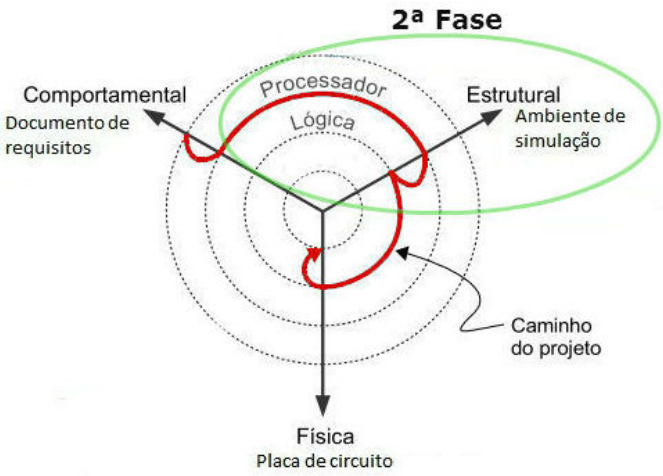


Figura 16 Diagrama em Y de Gajski, com ênfase na segunda fase

é a experiência da equipe de trabalho nas ferramentas fornecidas pelo fabricante, fator crítico se considerar a relevância que o tempo de construção do sistema tem no sucesso do equipamento. Neste caso, tanto a experiência como as ferramentas no laboratório que estão disponíveis para conclusão em tempo hábil desse projeto, são da empresa *Microchip Technology Inc.*.

Outros pontos devem ser considerados, como a quantidade de pinos disponíveis para comunicação externa. Neste projeto, é possível estimar, graças à descrição das entradas e saídas do diagrama de blocos (Figura 15), a quantidade de pinos necessários. Sendo a necessidade de um display de LCD (Subseção 3.3.4) no projeto que impõe ao menos dez pinos de comunicação. As técnicas usadas para otimizar o hardware permitem o uso de três pinos para controlar a matriz (Subseção 3.3.3) de LEDs e mais dois pinos de entradas analógicas para o teclado (Subseção 3.3.2). Para a porta USB e

o dispositivo SdCard (Subseção 3.3.5), são usados seis pinos, portanto, o microcontrolador escolhido deve ter, no mínimo, 21 pinos de comunicação de entrada e saída.

O desempenho também é importante, pois o controle da matriz de LEDs realizado com técnicas de multiplexação (Subseção 3.3.3), o monitoramento constante das entradas analógicas (Subseção 3.3.2), o envio de sinais para o dispositivo SdCard (Subseção 3.3.5) e mais o limite de tempo (RNF01) de resposta imposto pelo projeto, sugere especial atenção nos recursos de memória e frequência máxima de trabalho como fatores que devem ser levados em consideração.

Os microcontroladores PIC, fabricados pela empresa *Microchip Technology Inc.*, apresentam arquitetura *Harvard* e conjunto de instrução RISC. Sendo o barramento de dados (8 bits) separado do barramento de instrução (12, 14 ou 16 bits), isso proporciona que uma instrução seja buscada na memória, enquanto outra instrução está sendo executada, o que torna o processamento mais rápido. Além disso, é possível inserir no OPCODE o dado e a posição em que este irá operar, proporcionando, assim, que apenas uma posição de memória seja utilizada a cada instrução, economizando memória de programa (ELEOTÉRIO, 2012).

A principal característica, do PIC 18F4550 (MICROCHIP, 2006) é o encapsulamento com 40 pinos, dos quais 35 podem ser configurados como portas de I/O, com 13 entradas analógicas configuráveis de até 10 bits de resolução e pinos que aceitam o protocolo USB 2.0 e SPI. Outro fator importante é que, com frequência de operação de até 48 MHz e com arquitetura que utiliza técnicas de *pipeline*, o PIC 18F4550 garante um ótimo desempenho entre as famílias de microcontroladores. Estes fatores somados

favorecem a escolha deste microcontrolador para este projeto. Além disso, o PIC 18F4550 é facilmente encontrado nas principais lojas do ramo e em sites especializados, com preço atual de R$ 15,00 ou U$ 6,00, no varejo, o que torna este componente adequado às especificação deste projeto. Além disso, outro fator que favorece a escolha do PIC18F4550 é o fato de este projeto não estar usando a totalidade de seus recursos, o que proporciona um nível de escalabilidade razoável, sendo possível, futuramente, incluir no projeto algumas funcionalidades que não estão no escopo atual.

Em vista de todos estes pontos, define-se o microcontrolador PIC18F4550, representado na Figura 17, como principal componente de hardware do bloco Microcontrolador da Figura 15.

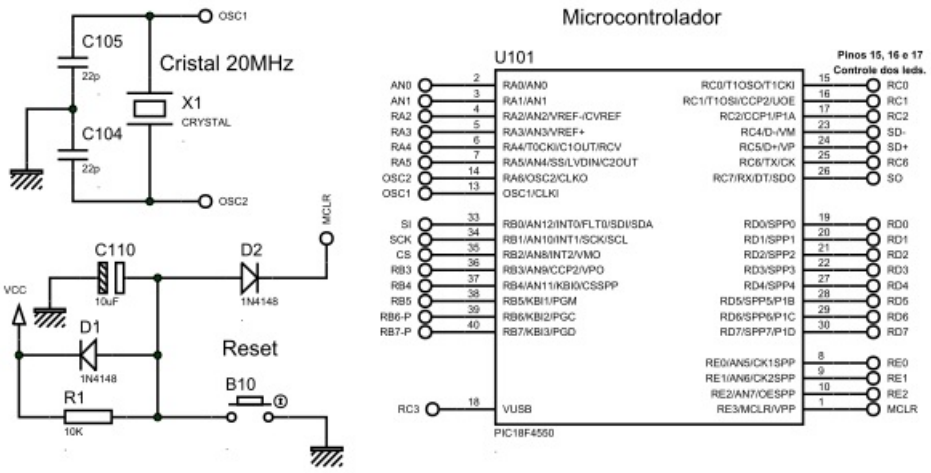


Figura 17 Bloco do Microcontrolador

Definindo-se o modelo do microcontrolador foi confeccionada a Tabela 1, que define os nomes dos pinos de entrada e saída que serão usados no software do sistema.

Tabela 1 Definição dos pinos do microcontrolador em software.

Pino	Definição no software		Função
1	#define ADC_1	portA.b0	AN0 / RA0
2	#define ADC_2	portA.b1	AN1 / RA1
15	#define Clock	portC.b0	RC0 / CCP2
16	#define Ser_Out	portC.b1	RC1 / T1OSO
17	#define Latch	portC.b2	RC2 / CCP1
23	#define USB_Neg	portC.c4	RC4 / D-
24	#define USB_Pos	portC.b5	RB5 / D+
33	#define SdCard_SI	portB.b0	RB0 / SDI
34	#define SdCard_SK	portB.b1	RB1 / SCK
35	#define SdCard_CS	portB.b2	RB2 / AN8
40	#define SdCard_SO	portC.b7	RB7 / PGD
10	#define LCD_RS	portE.b2	RE2 / AN7
9	#define LCD_EN	portE.b1	RE1 / AN6
19	#define LCD_D0	portD.b0	RD0 / SPP0
20	#define LCD_D1	portD.b1	RD1 / SPP1
21	#define LCD_D2	portD.b2	RD2 / SPP2
22	#define LCD_D3	portD.b3	RD3 / SPP4
27	#define LCD_D4	portD.b4	RD4 / SPP4
28	#define LCD_D5	portD.b5	RD5 / SPP5
29	#define LCD_D6	portD.b6	RD6 / SPP6
30	#define LCD_D7	portD.b7	RD7 / SPP7

3.3.2 Teclado

De acordo com o documento de requisitos, no item 3.3.2 do Apêndice E, é possível identificar a necessidade de dois conjuntos de teclados para este trabalho. Foi escolhida um método muito usado na industria de eletrônicos, em equipamentos de som e vídeo, para monitoras as diversas teclas de funções nesse tipo de equipamento. Essa técnica é baseada na utilização da entrada analógica como parte da construção deste módulo.

O *datasheet* do microcontrolador PIC18F4550 informa que o periférico *Analog Digital Converter* (ADC) tem resolução de 10 bits, que consegue

representar, no máximo, 1.024 valores em decimal. Considerando uma tensão de referência de 5 Volts, tem-se um degrau de, aproximadamente, 4,8 mV (5 / 1023 = 0,0048). Para cumprir os requisitos deste projeto, são necessários 24 teclas ou posições, 12 para cada jogador. Neste cenário, é possível ajustar a resolução de 10 para 5 bits, com 31 posições diferentes e um degrau mínimo de 161 mV. Este valor viabiliza o controle mais adequado do efeito *bounce* e de eventuais atrasos no acionamento dos botões causados pela propagação dos sinais.

A ideia está representada na Figura 18, junto com a Tabela 2, onde é possível ver um divisor de tensão com 25 resistores de 100 **Ω** cada. Dessa forma, têm-se 24 pontos que se diferenciam a, aproximadamente, 200 mV, valor mais que suficiente, já que é maior que o mínimo de 161 mV. Na mesma figura também é possível ver os conjuntos de teclas do jogador A e do jogador B, sendo que a cada tecla acionada será gerado um valor de tensão diferente na saída do divisor. Para garantir que a impedância de entrada do ADC do microcontrolador não influencie o divisor resistivo, é adicionado um amplificador operacional configurado como *buffer*, cujo valor de saída é representado por **Saída_Buffer**.

Com a definição do conceito de hardware estabelecido e a possibilidade de que cada tecla envie para a porta analógica do microcontrolador uma tensão diferente, desenvolve-se o Algoritmo 2 como base do software usado neste módulo. O sistema deve ficar monitorando a entrada analógica, através do laço principal da função *main* e, ao pressionar qualquer tecla, a função Teclado é chamada. Na linha 2 pode ser visto que o parâmetro Saída_Buffer, qué é a tensão de saída do bloco do teclado, é passado para a função. Na linha 3 este valor é comparado com os os valores da Tabela 2

que, se encontrado, retorna a operação equivalente.

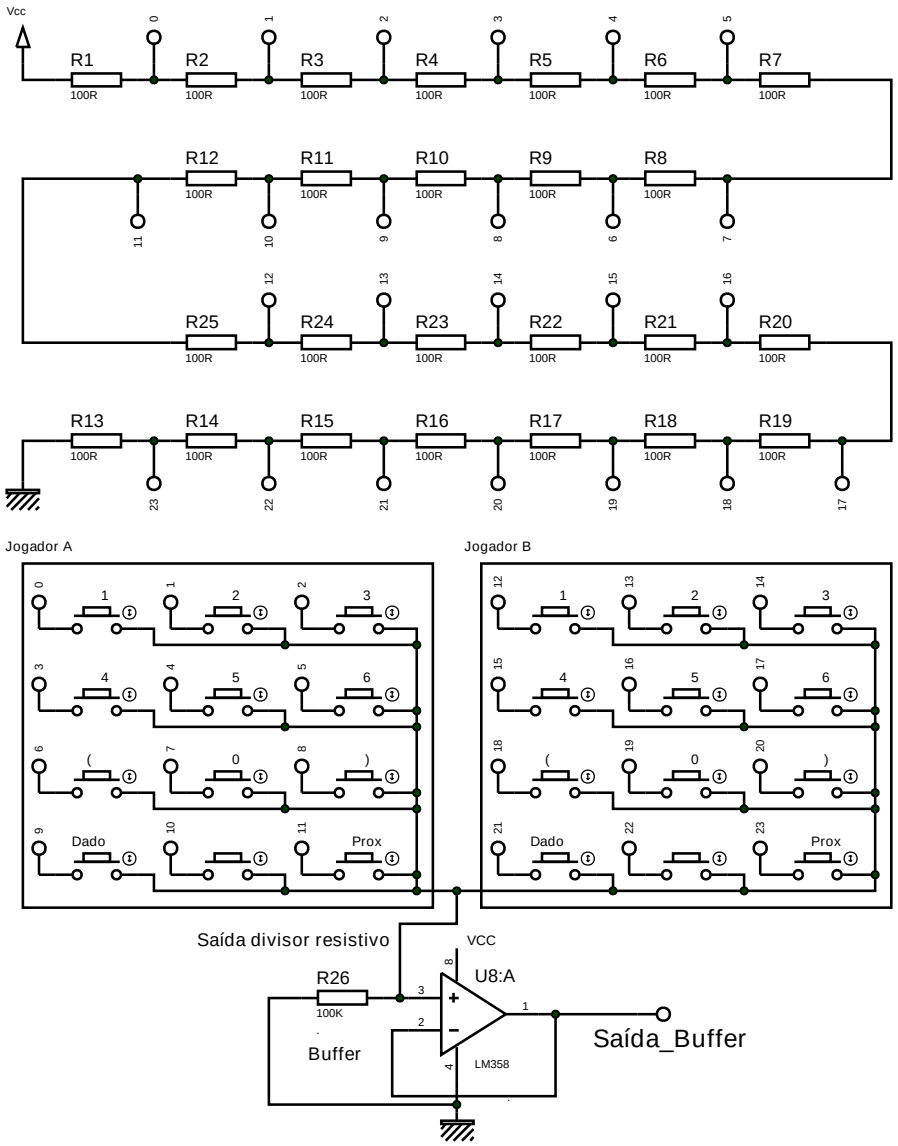


Figura 18 Bloco do Teclado

Tabela 2 Tabela de tensões na saída da Figura 18.

Posição	Operação	Tensão
1	Tecla 1 Jogador A	4.80V
2	Tecla 2 Jogador A	4.60V
3	Tecla 3 Jogador A	4.40V
4	Tecla 4 Jogador A	4.20V
5	Tecla 5 Jogador A	4.00V
6	Tecla 6 Jogador A	3.80V
7	Tecla (Jogador A	3.60V
8	Tecla 0 Jogador A	3.40V
9	Tecla) Jogador A	3.20V
10	Tecla Dado Jogador A	3.00V
11	Tecla Especial Jogador A	2.80V
12	Tecla Prox Jogador A	2.60V
13	Tecla 1 Jogador B	2.40V
14	Tecla 2 Jogador B	2.20V
15	Tecla 3 Jogador B	2.00V
16	Tecla 4 Jogador B	1.80V
17	Tecla 5 Jogador B	1.60V
18	Tecla 6 Jogador B	1.40V
19	Tecla (Jogador B	1.20V
20	Tecla 0 Jogador B	1.00V
21	Tecla) Jogador B	0.80V
22	Tecla Dado Jogador B	0.60V
23	Tecla Especial Jogador B	0.40V
24	Tecla Prox Jogador B	0.20V

Algoritmo 2: Algoritmo do modulo do Teclado.

```
Data: Teclado
1 begin
2     Teclado (Saída_Buffer);
3     Tensao ← Valores equivalente da Tabela 2;
4     Operacao ← Operação equivalente da Tabela 2;
5     if Saída_Buffer < lim_max_Tensão AND Saída_Buffer <
      lim_min_Tensão then
6         return Operacao;
```

3.3.3 Conversor Série-Paralelo e Matriz de LEDs

Os módulos representados na Figura 15, necessários para o cumprimento do item RF03 (Indicação no Tabuleiro), descrito no documento de requisitos do Apêndice E, juntamente com a Figura 1 do mesmo documento, mostram a necessidade de que cada casa do tabuleiro tenha dois LEDs de cores diferentes para representar as jogadas de cada um dos jogadores. Portanto, são 128 LEDs que o microcontrolador deve controlar. Foi definido que a disposição dos LEDs seja no formato de duas matrizes, como mostrado na Figura 19. A vantagem desta configuração está no fato de que, em cada matriz, os 64 LEDs podem ser controlados por 16 pinos do microcontrolador (entradas). Além disso, o uso de duas matrizes ao invés de apenas uma contempla o requisito RNF04 (escalabilidade), pois facilita a possibilidade da troca do painel de LEDs.

Apesar de a disposição dos LEDs em matriz auxiliar na diminuição de portas de controle, ainda não resolve por completo este problema. Afinal, serão precisos 32 pinos para controlar os LEDs e o microcontrolador escolhido não dispõe deste número de terminais livres, considerando as outras atividades que ele terá de implementar, como visto na subseção 3.3.1. Para resolver esta questão, lança-se mão de conceitos utilizando registradores de deslocamento, multiplexação e persistência retiniana, que são discutidos na seção 2.2.

O componente principal deste módulo é o CI 74HC595[5] que, de acordo com seu *datasheet*, é um registrador de deslocamento de 8 bits com entrada serial e saída paralela/serial. Neste componente, a saída serial pos-

[5]Este componente é um CI de *8 bits serial-input/serial or parallel-output shift*. Seu *datasheet* esta disponível em: http://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf. Acesso em: 29 nov. 2013.

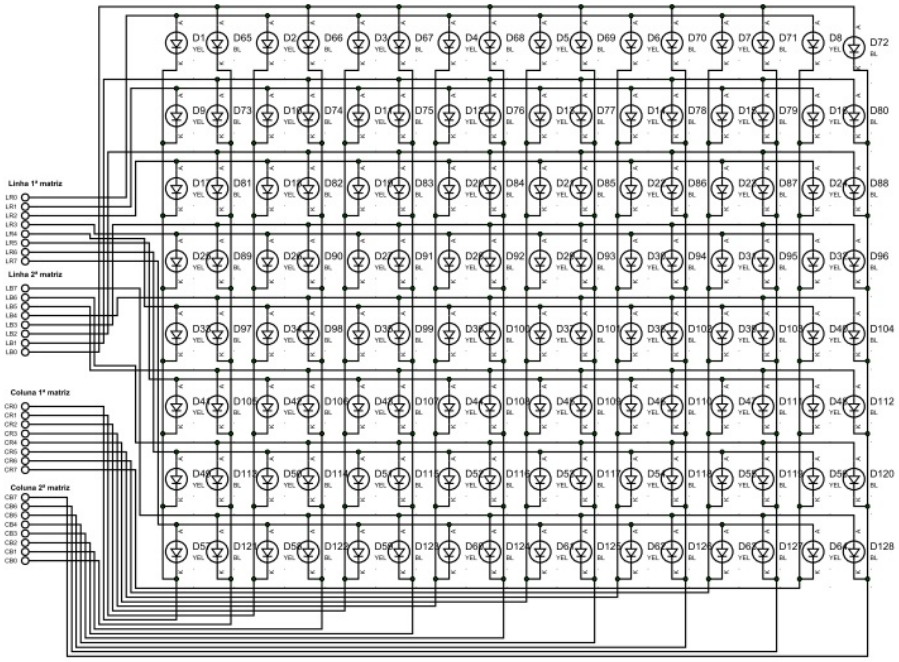
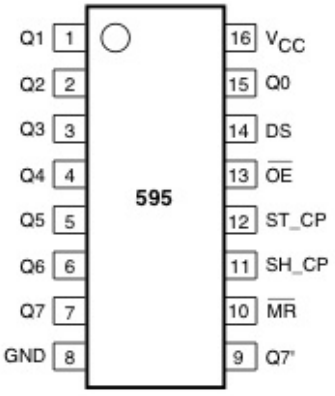


Figura 19 Matriz de LEDs

sibilita ligá-lo em cascata com outros CIs iguais, estendendo ainda mais a quantidade de portas. Na Figura 20 pode-se ver a descrição dos pinos deste componente.

Outro problema a ser resolvido é o consumo de corrente do painel de LEDs. Na pior situação, em que todos os LEDs de uma linha (oito) estiverem acesos, a corrente exigida será de aproximadamente 120 mA, mas o CI 74HC595 tem como corrente máxima nos pinos Q0 a Q7 de aproximadamente 20 mA. Para solucionar esta questão, foi introduzido no projeto o CI ULN2803 [6], que recebe o sinal de varredura das colunas. Este é um

[6] O ULN2803 é um *Octal higt voltage*. Seu *datasheet* esta disponível em: http://www.alldatasheet.com/datasheet-pdf/pdf/12687/ONSEMI/ULN2803.html. Acesso em: 06 dez. 2013.



Pinos 1-7 e 15	Q0 - Q7	Pinos de saída
Pino 8	GND	Terra
Pino 9	Q7'	Saída serial
Pino 10	MR'	Master Reclear
Pino 11	SH_CP	Shift Clock
Pino 12	ST_CP	Latch Clock
Pino 13	OE'	Output Enable
Pino 14	DS	Entrada serial
Pino 16	VCC	Alimentação

Figura 20 Descrição dos pinos do CI 74HC595

CI que possui internamente um conjunto de transistores que, devidamente polarizados, conseguem controlar uma corrente de até 500 mA, mais que suficiente para acender os LEDs. Na Figura 21 mostra-se como é feita a configuração dos componentes ULN2803 e 75HC595.

Definindo-se esses dois componentes, foi desenvolvido o circuito elétrico deste módulo, representado na Figura 22. Os pinos de *Clock, Latch e Data* são conectados a três pinos do microcontrolador. Por meio deles é possível definir os níveis de tensão de cada saída de forma individual que, por sua vez, será conectada às linhas e às colunas das matrizes de LEDs. Na Figura 23 mostra-se um exemplo de formas de onda para que alguns pinos sejam ativados. Enquanto a entrada ST1 (*Lacth*) está em *LOW* (nível baixo) e ligada ao pino 12 (*Latch*) de todos os CIs 74HC595, é possível enviar ao DS1 *Data*, pino 14 do CI U1, um sinal serial (*bits*) que será gravado na memória dos CIs. O tempo de cada *bit* é definido pela frequência do *Clock* da entrada SH1, pino 11 de cada CI. É importante perceber que o 74HC595 tem apenas oito saídas, quando se tem mais de oito bits na entrada

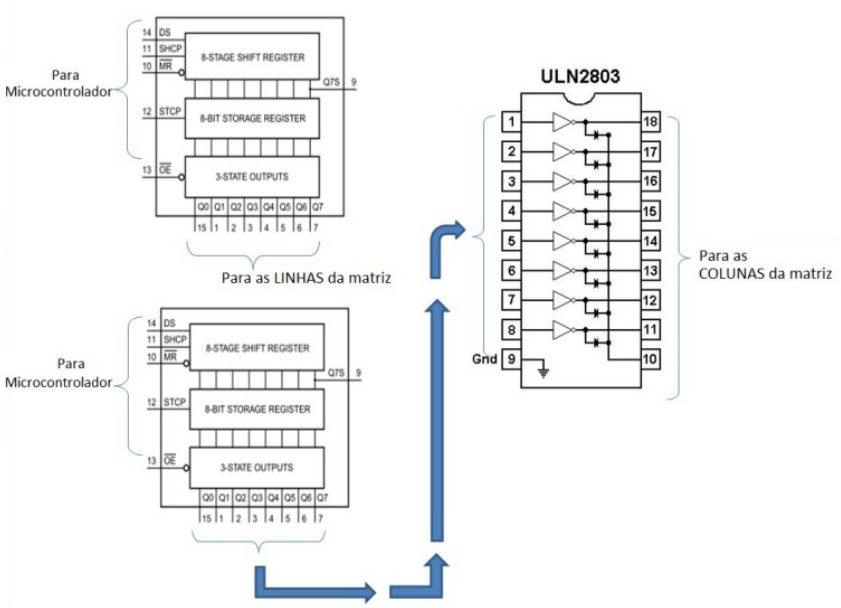


Figura 21 Drivers dos LEDs

de *data* (DS1), sem que o *Latch* seja colocado em *HIGH* (nível alto), o sinal é deslocado para o pino 9 (saída serial). Como este está ligado ao pino 14 (DS) do próximo CI, tem-se uma ligação em cascata dos componentes. Este processo permite, com três pinos de entrada, controlar de forma individual o nível de tensão dos 32 pinos de saída.

De posse do circuito elétrico e dos conceitos apresentados nesta subseção, foi possível criar o Algoritmo 3 para a criação do programa que controlasse os LEDs. É importante observar que só é possível controlar individualmente os LEDs da matriz usando o conceito de multiplexação. Isto é, o ciclo do tempo ao qual os sinais são enviados para a matriz cria uma diferença de potencial entre uma linha e uma coluna para polarizar e

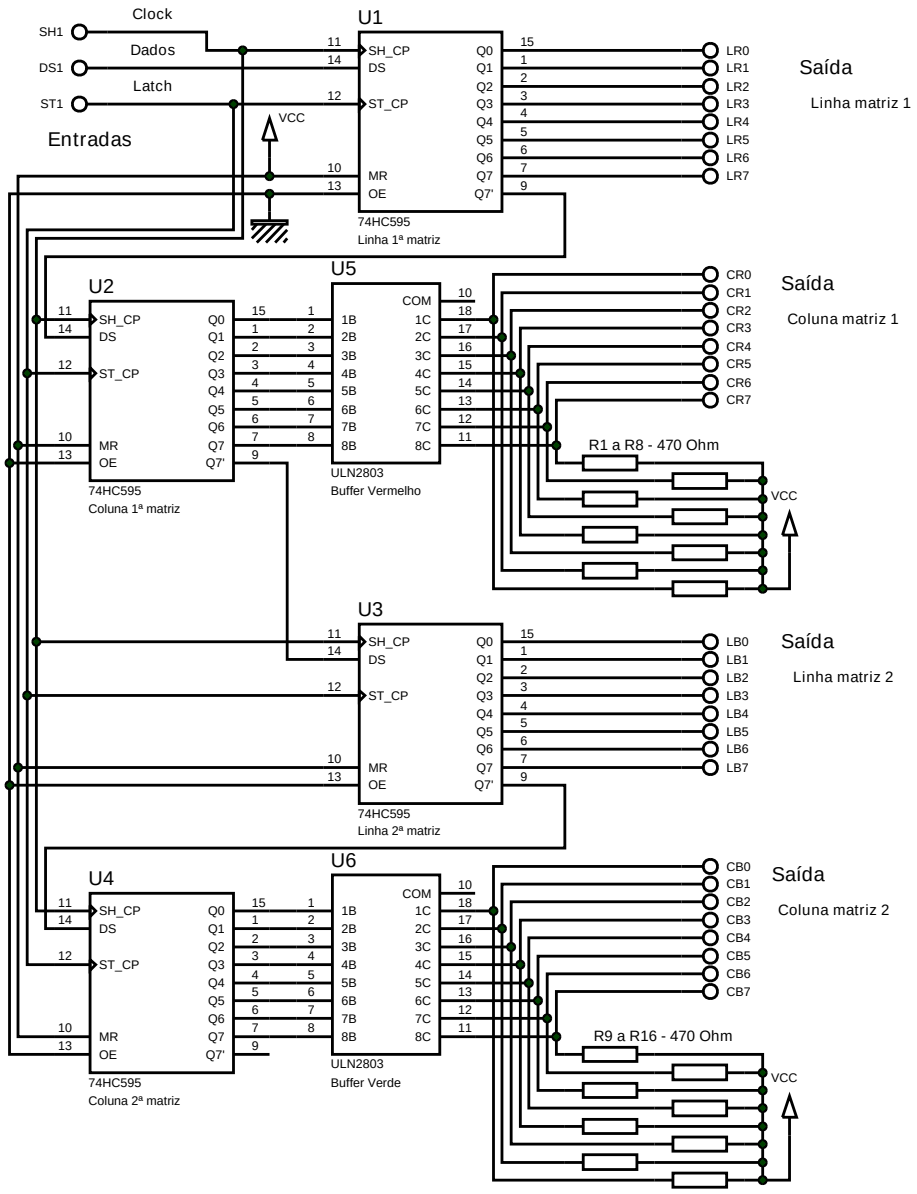


Figura 22 Circuito completo dos drivers dos LEDs

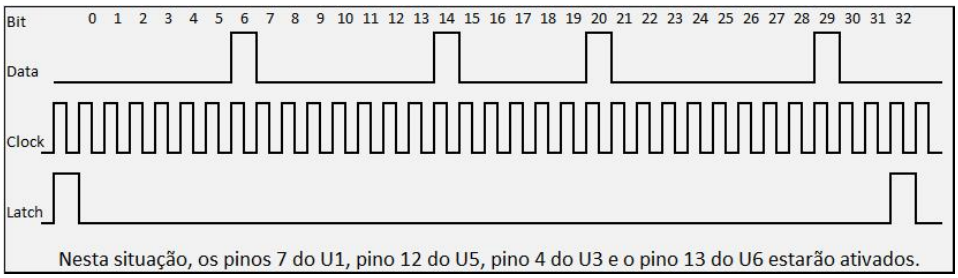


Figura 23 Forma de onda das entradas de *Clock, Lacth e Data*

acender um determinado LED. Esta frequência não pode ser menor que 100 Hz, o que garante que o efeito de persistência retiniana crie a ilusão, para o observador, de que o LED está aceso. Por outro lado, se esta frequência for muito alta, a intensidade de brilho do LED tende a diminuir. A variável *tempo*, definida na linha 3 do Algoritmo 3, determina esta frequência. As variáveis *Ser_Out, Clock e Latch* são globais e podem ser vistas na Tabela 1.

O Algoritmo 3 é iniciado recebendo, na linha 2, uma sequência de quatro *bytes* que contém as informações para acender determinado LED. O laço da linha 5 identifica o *bit* menos significativo e envia o valor correspondente ao pino *Ser_Out*. Na linha 16 é feito o deslocamento dos *bits* até o final. Terminando essa verificação, é enviado um pulso ao pino *Latch* que libera os sinais para a matriz de LEDs.

Algoritmo 3: Algoritmo para multiplexação dos LEDs

```
Data: Matriz
1  begin
2      Matriz (32Bits);
3      tempo = 10;
4      contadorBits = 0;
5      for (ContadorBit < 32; ContadorBits++) do
6          if 32Bits impar then
7              Ser_Out = 1;
8              Clock = 1;
9              Delay(tempo);
10             Clock = 0;
11         else
12             Ser_Out = 0;
13             Clock = 1;
14             Delay(tempo);
15             Clock = 0;
16         Desloca 32Bits a direita;
17     Latch = 1;
18     tempo = 1;
19     Latch = 0;
```

3.3.4 Display LCD

Para mostrar informações aos jogadores e cumprir os requisitos RF01 (Escolha do Modo do Jogo), RF02 (Geração de números aleatórios) e o

RF06 (Melhor jogada), foi escolhido um display de *Liquid Crystal Display* (LCD) de 2 linhas e 16 colunas. Além de ter um tamanho adequado para comunicação com o usuário, tem custo relativamente baixo e é facilmente encontrado nas principais lojas do ramo. Basicamente, sua construção é feita com uma lâmina de LCD e um microcontrolador (CI HD48780)[7] que se encarrega de controlar esta tela e também faz o interfaceamento para o microcontrolador externo, neste caso, o PIC 18F4550. Esta comunicação é feita de forma paralela com 8 bits, conforme Figura 24.

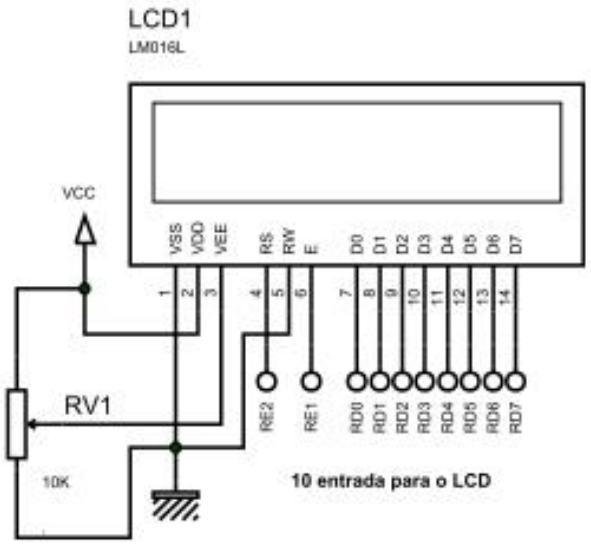


Figura 24 Circuito eletrônico do LCD

Neste circuito é possível identificar o resistor variável RV1 que ajusta o contraste do LCD. Os pinos de 7 a 14 são os bits de comunicação entre

[7] O *datasheet* deste componente esta disponível em: http://pdf1.alldatasheet.com/datasheet-pdf/view/63663/HITACHI/HD44780U.html. Acesso em: 12 fev. 2014. Os pinos de configuração do LCD são descritos em: https://www.sparkfun.com/datasheets/LCD/GDM1602K.pdf. Acesso em 12 fev. 2014.

os microcontroladores. Como será visto na seção 3.3.8, o programa usado como ambiente de programação tem diversas bibliotecas prontas para manipulação do LCD, como a função Lcd_Out(*char row, char column, char *text*), que envia aos pinos definidos na Tabela 1 as informações do ponteiro *text.

3.3.5 Registro do jogo

Para o cumprimento do requisito RF05 (Registro do jogo), como consta no Apêndice E, é importante que o sistema consiga armazenar em algum dispositivo computacional cada lance do jogo. No início do projeto foi estabelecida e incorporada ao circuito uma saída USB, para ligar um *pendrive*. No entanto, a manipulação deste dispositivo exige uma entrada *Host USB*, o que não está disponível no microcontrolador PIC 18F4550. Dessa forma, foi acrescentado ao circuito um conector SdCard (*Secure Digital Card*). Seu baixo preço e facilidade na implementação da comunicação com o microcontrolador foram os principais motivos da escolha deste dispositivo.

A comunicação com o microcontrolador é feita por meio de um protocolo de dados seriais síncronos, também conhecido como *Serial Peripheral Interface* (SPI). Segundo Ibrahim (2010), o protocolo SPI utiliza duas linhas de dados seriais, uma para entrada (SI) e outra para saída (SO), mais um pino para selecionar o dispositivo (CS) e um pino de *clock* (SCK).

O circuito eletrônico para incorporar um SdCard ao PIC 18F4550 é muito simples, como mostrado na Figura 25. Porém, um problema encontrado são as diferenças entre os níveis de tensão do microcontrolador, que é aproximadamente 4,2 volts, e do SdCard, que é 3,6 volts. Para contornar

esse problema, Ibrahim (2010) propõe que sejam incorporados divisores resistivos a cada pino do microcontrolador, para não danificar o SdCard. Com este circuito e com o conhecimento do protocolo SPI foi possível desenvolver o Algoritmo 4.

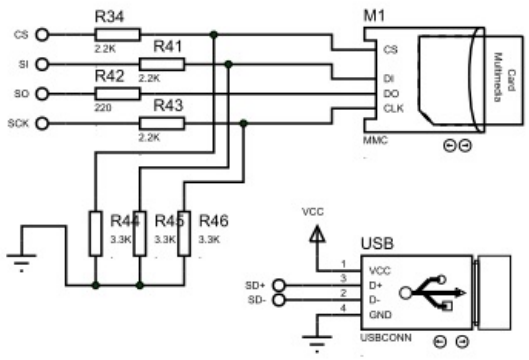


Figura 25 Circuito elétrico da porta USB e do SD Card

Algoritmo 4: Algoritmo de manipulação do SdCard.

```
Data: SDCard
1 begin
2     SDCard(true, filename, operação);
3     if Existe Cartão then
4         criarArquivo(filename);
5         enviarOperacao(operação);
6         return (true);
7     else
8         Não existe Cartão
9     return (false);
```

3.3.6 Ambiente de Simulação

Existem diferentes ambientes de simulação para sistemas embarcados e, como cada projeto tem suas peculiaridades, a escolha dessa plataforma deve ser considerada um ponto crítico do projeto.

1. *Proteus Professional Suite Desing*®:

   Para este sistema, foi escolhido como plataforma de simulação o Software Proteus Professional Suite Desing®, desenvolvido e mantido pela empresa inglesa Labcenter Electronics EDA[8]. A escolha deste programa, além do autor deste trabalho conhecê-lo bem, foi baseada no fato de que todos os componentes usados no projeto fazem parte de sua biblioteca, facilitando o desenvolvimento. No conjunto de programas que o Proteus® disponibiliza, está o ISIS® (*Inteligent Schematic Input System*), para o desenvolvimento de circuitos eletrônicos; o *Virtual System Modeling* (VSM®), que é o módulo de simulação do circuito eletrônico; e o *Advanced Routing and Editing Software* (ARES®), que é o programa que faz o desenho da placa de circuito impresso.

2. *MikroC PRO for PIC*®:

   É um dos muitos compiladores existentes que suportam as bibliotecas necessárias para um completo desenvolvimento de qualquer projeto utilizando microcontroladores da família PIC de 12, 16 e 18 da Microchip. Desenvolvido e mantido pela empresa MikroEletronika[9], disponibiliza diversas funções prontas que facilitam muito o desenvolvimento do software do sistema. Outra vantagem é que a versão

[8] Informações disponíveis em: http://www.labcenter.com/index.cfm. Acesso em: 03 jul. 2013.

[9] Diponivel em: http://www.mikroe.com. Acesso em: 01 jun. 2013.

gratuita para estudantes suporta programas de até 2K *words*.

3. *LPKF CircuitPro 1.5*:

   A máquina de fresa da marca *Laser & Eletronics*[10], modelo S63, é um *plotter* para confecção de PCB (*Printed Circuit Board*). O software LPKF *CircuitPro* 1.5 controla o *plotter*, a partir de arquivos com padrão *Gerber*, gerados pelo software ARES®, que contém as tabelas de furação, espaçamento das trilhas e as medidas das ferramentas necessárias.

De posse destes três programas, dos circuitos elétricos dos blocos e dos algoritmos das subseções anteriores, é possível fazer o circuito elétrico completo e as placas do circuito (PCB). Além disso, é feita a integração deste circuito com o *firmware* (programa que é executado diretamente no hardware) desenvolvido e, assim, testar todo o sistema no ambiente de simulação. Neste momento, é possível particionar, com muita clareza, o desenvolvimento dos componente de hardware e de software do sistema.

3.3.7 Hardware

De forma quase automática, nesta fase do projeto acontece o particionamento entre hardware e software, sendo o hardware desenvolvido na ferramenta EDA Proteus® e o software, no compilador MikroC®.

No Apêndice A é detalhado o circuito elétrico completo do sistema, em que cada componente eletrônico é definido por meio de cálculos de tensões e correntes. A partir deste circuito é gerada a lista de componentes (Apêndice D) e no ARES®, os desenhos das placas de circuito impresso

[10]Disponível em: http://www.lpkf.com. Acesso em: 2 dez. 2013

(Apêndice B), em que é considerado o posicionamento dos componentes e o tamanho das placas. Com o desenho dessas placas são gerados os arquivos no formato *Gerber* (formato especial usado em equipamentos de *fotoploter*) para a confecção da PCB na fresadora S63.

3.3.8 Software

O ambiente de programação escolhido para gerar o *firmware* foi o MikroC PRO for PIC®. Na Figura 26 é possível ver este ambiente e alguns comentários do programa desenvolvido. Depois de compilado, uma tela de mensagem informa a quantidade de memória RAM, ROM e outras informações que são utilizadas pelo programador para monitorar erros e limites do microcontrolador.

Nesta subseção, além de apresentar o ambiente de programação, também são produzidos o fluxograma e o diagrama de máquina de estado do jogo. Nestes artefatos, além de incorporar todos os algoritmos fornecidos pelas subseções anteriores, é possível ver algumas rotinas que não têm dependência direta com o hardware, isto é, não necessitam de entradas externas ou de algum pino do microcontrolador.

- Gerador de números aleatórios: para o cumprimento do requisito RF02, do documento de requisitos do Apêndice E, é preciso gerar três números aleatórios, de 1 a 6, que representem os operandos para que o jogador monte a sentença numérica, cujo resultado determine sua jogada no tabuleiro. A maneira adotada para gerar números pseudo aleatórios que sejam razoavelmente adequados ao jogo e que não usem funções recursivas que consomem memória em excesso foi gerar três contadores independentes e que fossem incrementados em tempos e por valores diferentes. Esses contadores

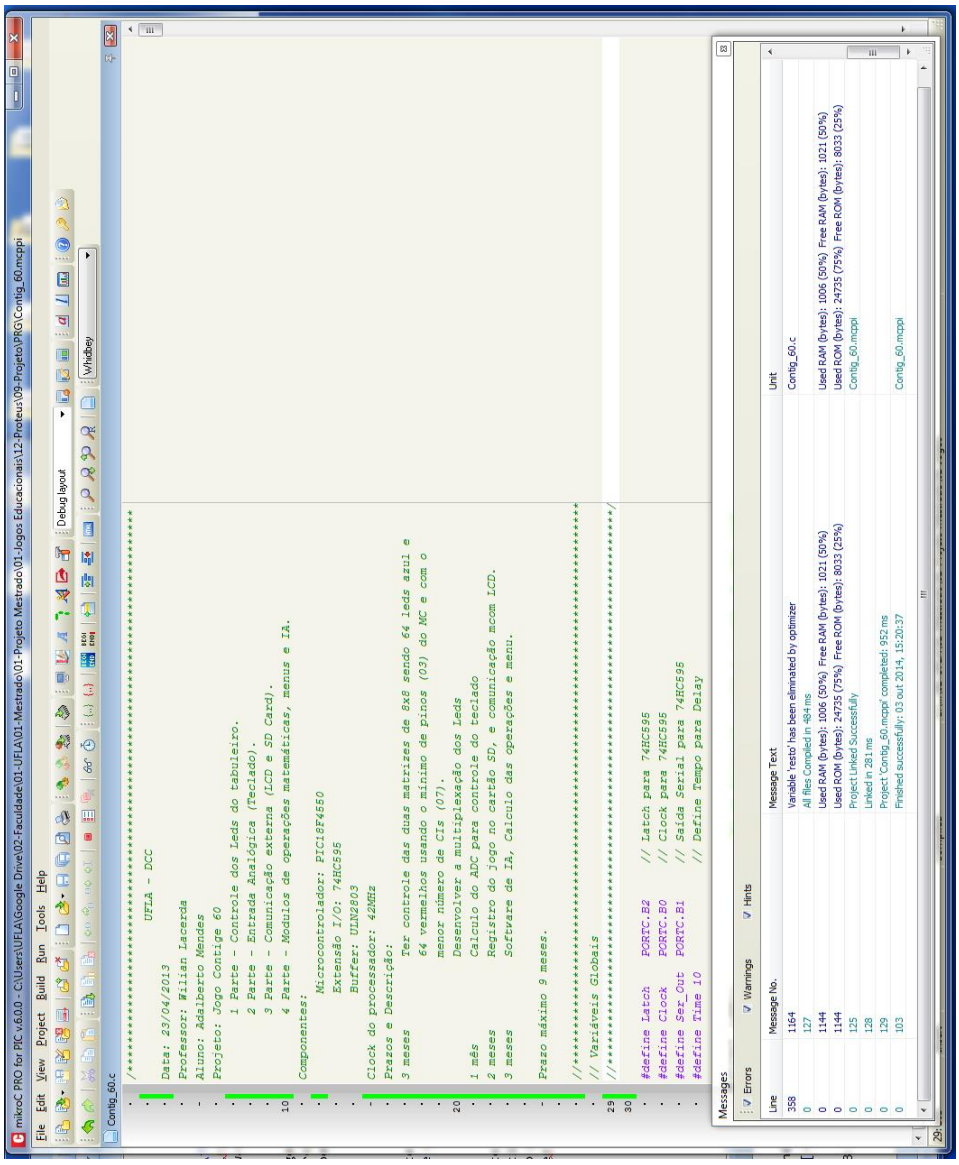


Figura 26 Tela do MikroC

usam variáveis globais e foram implementados dentro da interrupção configurada pelo periférico *Timer0* do microcontrolador. Dessa forma, quando a tecla *Dado* é acionada, o sistema simplesmente apresenta os valores destas

três variáveis no LCD.

- Jogador autônomo: a necessidade de um jogador autônomo é descrita nos requisitos RF04 - Jogar com o Sistema e RF06 - Melhor jogada. O primeiro requisito é importante para o momento em que o jogador humano deve testar seu aprendizado e jogar com competência. O segundo acontece em momentos de intervenção pedagógica, quando o orientador precisa identificar uma melhor jogada em um determinado momento do jogo. Ainda é importante citar que o objetivo deste jogador autônomo não é ser invencível, mas sim estabelecer certo nível de capacidade para jogar com um jogador humano e, dessa forma, treinar sua competência.

A base para a construção deste agente autônomo é o algoritmo *Minimax*, considerado um método adequado em situações nas quais existam problemas de busca competitiva, isto é, quando um agente tenta planejar suas ações considerando as ações de outro agente, que seja seu oponente. Jogos de tabuleiro, como o Contig 60®, apresentam esse cenário e possibilitam o uso deste algoritmo. Outra vantagem deste algoritmo é a possibilidade de limitar o espaço de busca, o que viabiliza a implementação, mesmo com poucos recursos de processamento e memória, justamente a situação de projetos em sistemas embarcados.

Neste projeto, o módulo do jogador autônomo tem duas funções principais, que são as seguintes:

1. Identificar quais são as possíveis jogadas.

No jogo Contig 60®, o tabuleiro tem 64 casas numeradas. Esses números são os possíveis resultados das operações que o jogador deve conseguir, de acordo com sua estratégia. A Expressão Numérica 1 representa um arranjo simples, onde pode-se ver que a operação a ser criada deve

ter dois (p) operadores, sendo permitido o uso das quatro (m) expressões matemáticas. Logo, têm-se 16 possíveis operações.

$$A_r(m,p) = m^p = 4^2 = 16 \tag{1}$$

Nas operações em que o uso de parênteses é permitido, eles podem alterar o resultado da equação. Por exemplo, a operação [1] A+(B*C) pode ter resultado diferente de [2] (A+B)*C, da mesma forma que [3] A*(B+C) pode ser diferente de [4] (A*B)+C. Dessa forma, a Expressão Numérica2 mostra que o uso de parenteses possibilita a criação de mais 64 operações que podem gerar resultados diferentes, totalizando 80 operações.

$$(4^2) + (16 * 4) = 80 \tag{2}$$

Ainda, a posição dos três operandos na equação pode influenciar o resultado das operações, como é mostrado na Expressão Numérica 3, que aumenta para 480 possíveis operações diferentes.

$$((4^2) + (16 * 4)) * (3!) = 480 \tag{3}$$

Na prática, nem todas as combinações de operandos, operadores e posições dos parenteses fornece um resultado diferente. Por exemplo, qualquer combinação da operação A + (B + C) vai fornecer o mesmo resultado.

Outro fator importante é que o tabuleiro tem apenas 64 casas, com valores diferentes, o que diminui ainda mais as possibilidades de operações válidas para o jogo. Considerando estas informações, esta função deve criar um conjunto de todas as possíveis operações válidas. Na Figura 27 mostra-se, em vermelho, o tabuleiro com todos os possíveis resultados se os números sorteados forem 5, 4 e 6.

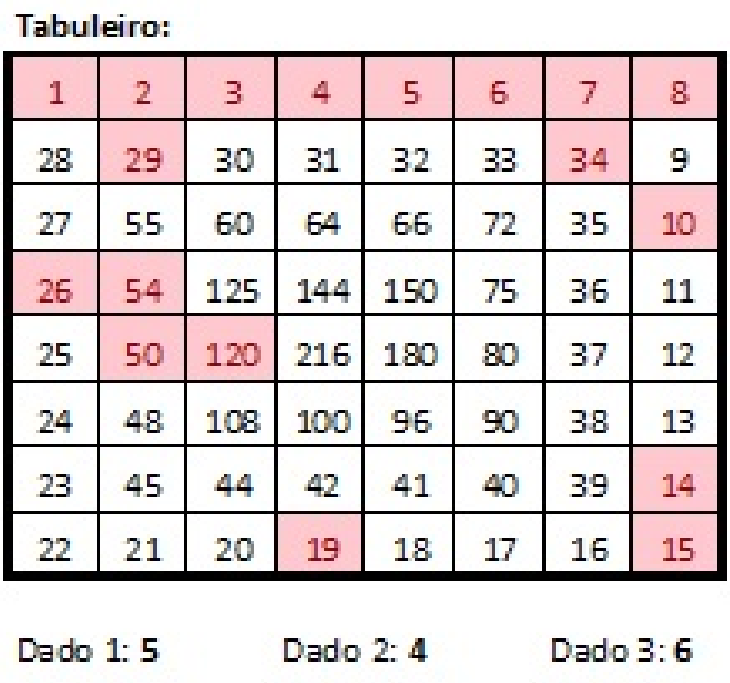


Figura 27 Possíveis resultados com os valores 4, 6 e 5

2. Escolher a melhor jogada de acordo com estado do tabuleiro.

Aqui o algoritmo *Minimax* é adaptado ao sistema, em que a primeira providencia é criar, considerando o estado atual do jogo, uma função cujo objetivo é estabelecer um valor de peso ao tabuleiro. Como o objetivo do jogo é ocupar uma sequência de cinco casas, na vertical, na horizontal

ou na diagonal, esta função deve estabelecer uma pontuação positiva ou negativa às casas adjacentes as que estão ocupadas. Pontos positivos são as casas adjacentes que o jogador MAX escolheu e pontos negativos, as do jogador MIN. Somando esses pontos, tem-se o peso do tabuleiro no estado (nó) atual. Na Figura 28 mostra-se o nó raiz, o estado inicial do tabuleiro e, a partir deste estado, com os números sorteados pelos dados, é possível identificar as próximas jogadas (nós) e construir mais um nível da árvore. Ao aplicar a fórmula de pontos em cada nó, consegue-se descobrir quais jogadas favorecem o jogador MAX (maior peso). Para a criação do próximo nível, ainda não se têm os números sorteados. Logo, é preciso aplicar a fórmula de pontos em todas as casas vazias. O melhor caminho, e portanto a melhor jogada, é o que leva à folha cujo peso é maior que o peso do nó raiz. Nas simulações, houve casos em que não foi identificado nenhum nó folha com peso superior ao nó raiz. Nessas situações, o peso da folha que mais se aproxima ao peso do nó raiz é escolhido.

É possível estender a árvore a níveis mais profundos, mas o tempo estabelecido de três segundos no requisito RNF01 impede o sistema de prosseguir na criação dos níveis da árvore. Em testes, outro problema que surgiu é que, a partir da criação do quarto nível, existem perdas de valores das variáveis dos pesos dos nós. Isso acontece pois o microcontrolador tem pouca memória para armazenar endereços de rotinas que executam recursivamente.

Para que o jogador autônomo retorne uma jogada válida, mesmo que ultrapasse o tempo limite ou que o algoritmo não encontre a melhor jogada, foi estabelecido que, nestes casos, a melhor jogada seja equivalente à casa mais ao centro (para o jogo, as casas do centro do tabuleiro são de valores maiores, portanto, mais difíceis de fazer operações para estes resultados) do

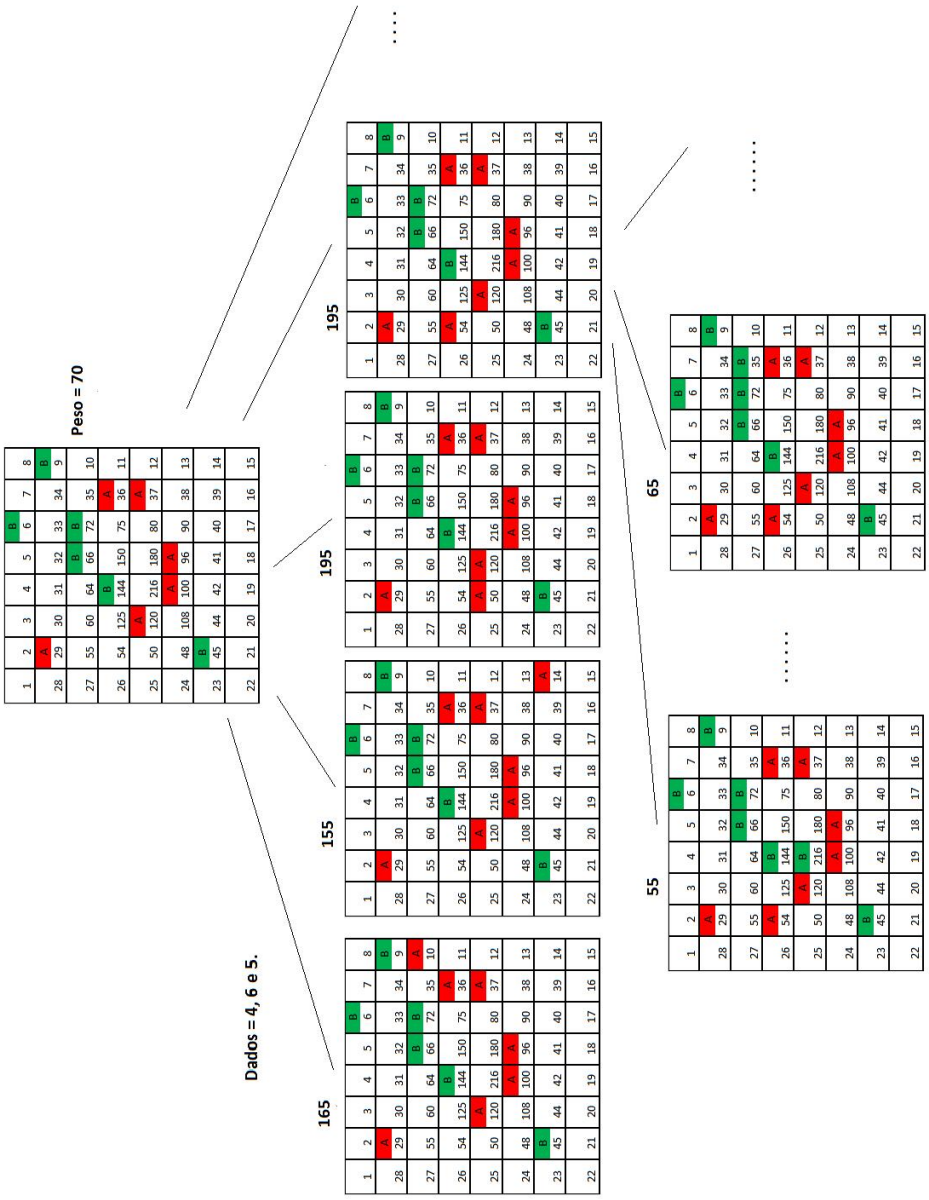


Figura 28 Parte da árvore gerada para uma situação de jogo

tabuleiro.

De posse destes conceitos, foi criado o Algoritmo 5, que auxilia a construção das funções que formam o jogador autônomo, como em todas as subseções anteriores.

Algoritmo 5: Algoritmo para a criação do jogador autônomo

Data: $MinMax(Estado, Valordosdados, Profundidade, tempo)$

```
1  begin
2      if No Raiz? then
3          Calcular peso do nó;
4      else if Jogador MAX then
5          Calcula possíveis jogadas (Valor dos Dados);
6          Calcular peso de cada nó e gravar os maiores;
7      else if Jogador MIN then
8          Calcular peso de cada nó e gravar os menores;
9      if Peso MIN < Peso RAIZ then
10         retorna melhor jogada;
11     else
12         Aplica função Minimax;
```

.

- Diagramas do jogo: para facilitar o desenvolvimento do *firmware* de forma concisa, foi necessário elaborar um diagrama de estados que, representado pela Figura 29, auxilia o entendimento da lógica do jogo. Os círculos maiores representam os estados do jogo, isto é, qual jogador deve proceder à jogada naquele momento; os círculos menores representam os estados iniciais e finais do jogo; os arcos representam os estímulos para as

mudanças de estado.

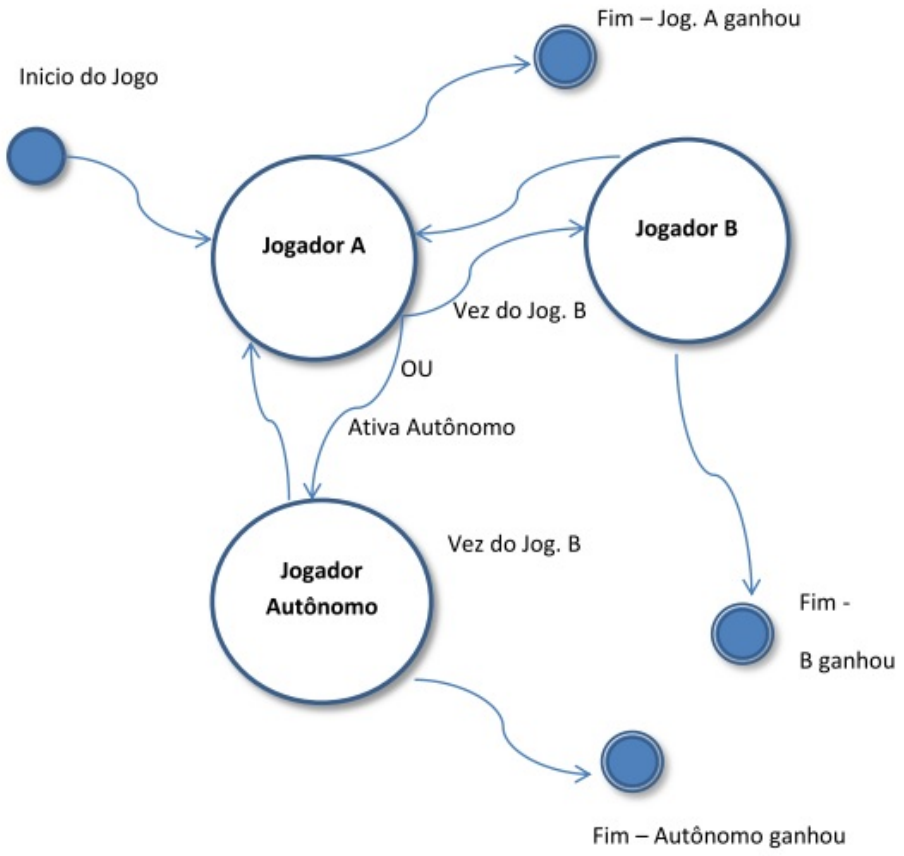


Figura 29 Diagrama de estados

Outro diagrama muito importante para o entendimento da lógica que o *firmware* deve seguir é o fluxograma. Na Figura 30 é possível ver o início do jogo, em que o sistema verifica qual é o jogador corrente. Depois, o gerador de números aleatórios mostra três valores (de 1 a 6) na tela de LCD, números com os quais o jogador deve criar a sentença numérica. Se o

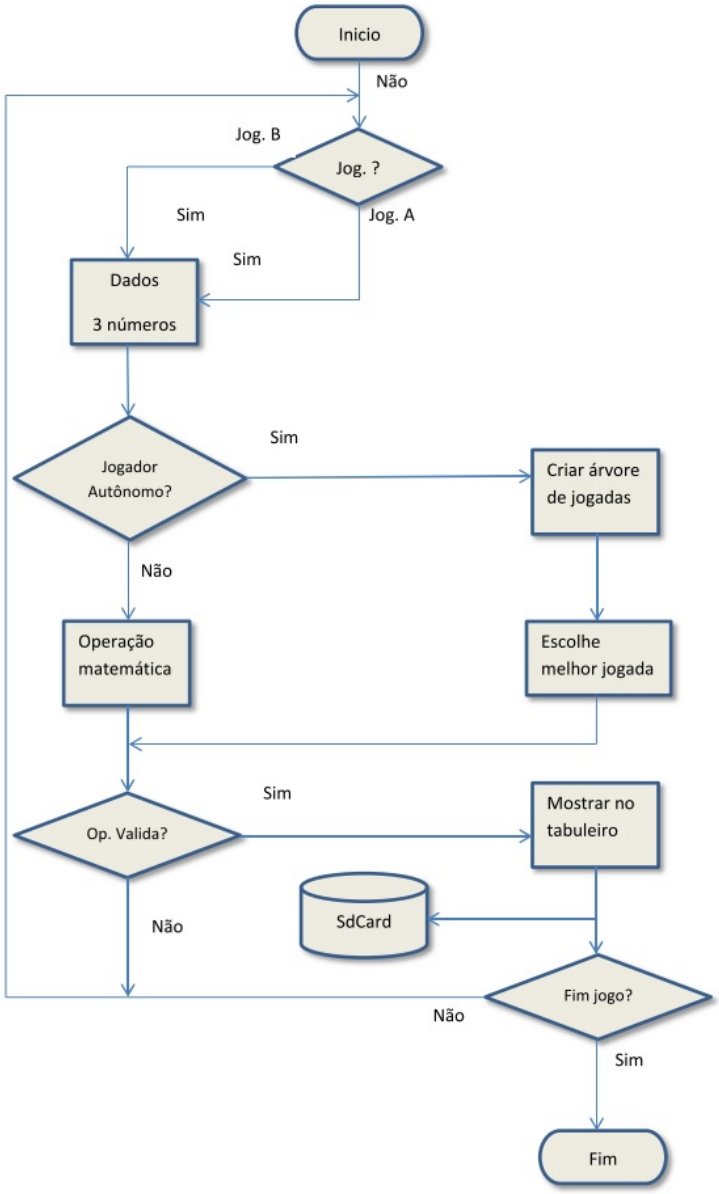


Figura 30 Fluxograma do jogo

jogador autônomo é ativado, o sistema cria a árvore de possíveis jogadas e, com o algoritmo adaptado do *Minimax*, é feita a escolha da melhor jogada,

mostrando no tabuleiro e gravando no SdCard. Se o jogador autônomo não for ativado, o jogador humano deve digitar no teclado a sentença elaborada, o sistema faz uma verificação se a sentença é valida e, novamente, mostra no tabuleiro e grava no SdCard. Se a configuração do tabuleiro for uma configuração de final de jogo, é mostrado no LCD o vencedor e o jogo termina.

Com a etapa da dimensão estrutural terminada, uma série de artefatos de hardware e software foi criada, capacitando o desenvolvedor a seguir para uma dimensão de baixa abstração, isto é, a Dimensão Física (subseção 3.4).

3.4 Dimensão física

Com os artefatos confeccionados até este momento do projeto é possível seguir o caminho do projeto da Figura 31 e abordar as camadas Lógica e Circuito, no eixo da dimensão Física. Fazem-se o levantamento de preço e a compra do material para a confecção do protótipo e, em seguida, as placas de circuito na fresadora S63, a montagem dos componentes na PCB, a integração do software desenvolvido e os testes de validação do sistema.

3.4.1 Custo do projeto

Na Tabela 3 são apresentados os preços dos componentes que foram utilizados neste projeto. No início do trabalho foi estimado um valor limite de R$ 100,00 para os gastos, indicado no documento de requisitos (Apendice E, RNF03 - Custos do sistema). Mas, no decorrer deste trabalho e das iterações que existiram, algumas mudanças se fizeram necessárias, como a inclusão do conector SdCard e de mais alguns componentes discretos, o que

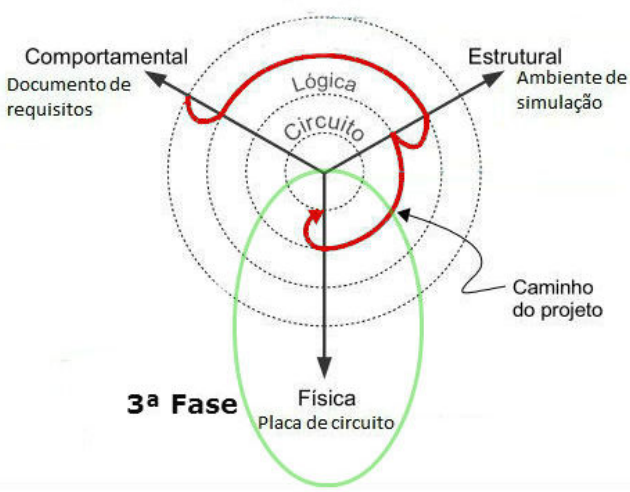


Figura 31 Diagrama em Y de Gajski, com ênfase na terceira fase

aumentou o valor final do protótipo. Como pode ser visto na Tabela 3, o custo total foi de R$ 113,95.

3.4.2 Prototipadora S63

As placas de circuito foram feitas na fresadora, ou *prototipadora*, da empresa LPKF, modelo S63 (Figura 32). Esta fresadora trabalha dentro de um gabinete acústico e isolado. As ferramentas (brocas e fresas) trabalham a uma rotação próxima a 60.000 RPM, garantindo ao equipamento uma precisão ótima para confecção das três PCBs deste trabalho: a placa de LED e do teclado que compõe o tabuleiro, a placa de controle dos LEDs e a placa do microcontrolador.

Tabela 3 Tabela de preços

Item	Componente	Qte.	Custo Unitário	Total
1	Capacitores Cerâmicos	17	R$ 0,22	R$ 3,74
2	Capacitores Eletrolíticos	5	R$ 0,37	R$ 1,85
3	CI 74HC595	4	R$ 1,75	R$ 7,00
4	CI LM358	1	R$ 1,72	R$ 1,72
5	CI LM7805	2	R$ 0,90	R$ 1,80
6	CI PIC18F4550	1	R$ 19,50	R$ 19,50
7	CI ULN2803	2	R$ 1,20	R$ 2,40
8	Conector SdCard - 9 pinos	1	R$ 4,95	R$ 4,95
9	Conectores 10 pinos	12	R$ 0,57	R$ 6,84
10	Conector energia	1	R$ 1,50	R$ 1,50
11	Cristal 48Mhz	1	R$ 0,50	R$ 0,50
12	Diodo 4007	2	R$ 0,25	R$ 0,50
13	Diodo 4148	2	R$ 0,17	R$ 0,33
14	Fonte 9 Volts	1	R$ 12,00	R$ 12,00
15	LCD 16x2 - LM016L	1	R$ 17,30	R$ 17,30
16	Led comum Verde	64	R$ 0,10	R$ 6,34
17	Led comum Vermelho	64	R$ 0,10	R$ 6,34
18	Resistores 1/4 Wats	47	R$ 0,05	R$ 2,35
19	Miscelânea	1	R$ 20,00	R$ 27,00
	TOTAL		=	R$ 113,95

3.4.3 Montagem das PCBs

De posse das placas de circuitos impressos, confeccionadas pela fresadora, foram feitos o posicionamento e a soldagem dos componentes. Usou-se verniz próprio para conservação das placas, evitando corrosão pelo tempo. Na Figura 33 mostram-se, do lado esquerdo, os componentes do circuito e do lado direito, a solda dos componentes.

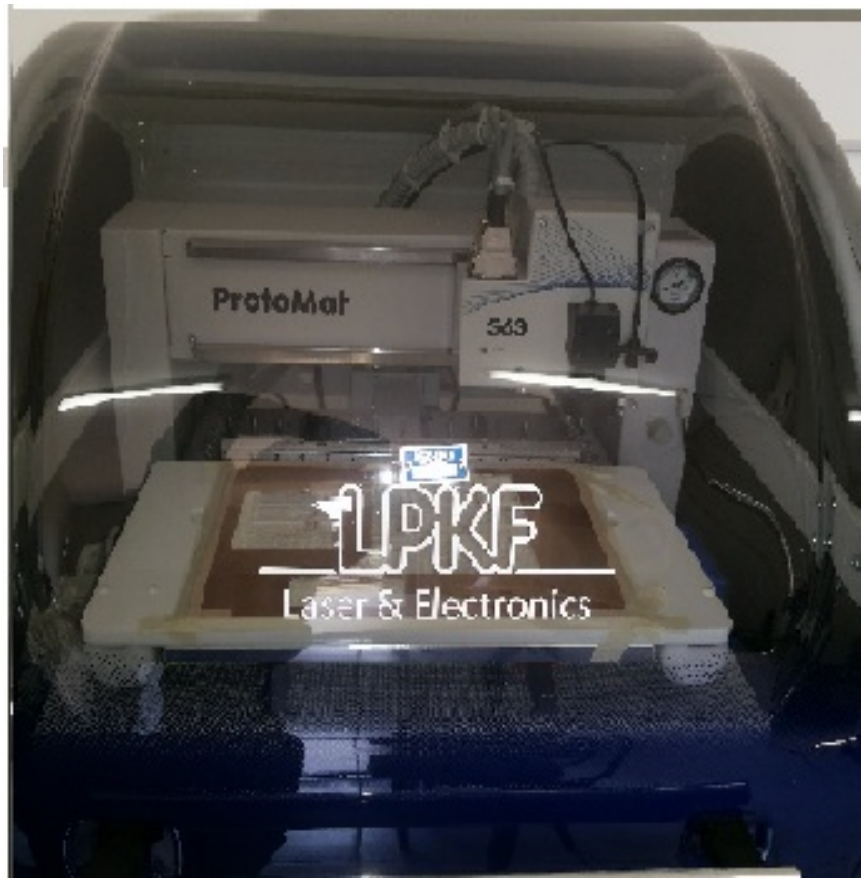


Figura 32 Prototipadora S63.

3.4.4 Integração entre software e hardware

Para a confecção do *firmware* foi utilizado o ambiente de programação MikroC PRO for PIC® e, para fazer o circuito elétrico do protótipo, foi usado o programa ISIS®, que possibilita fazer a integração do *firmware* ao circuito elétrico. Na Figura 34 observa-se o ambiente de simulação do jogo. Com estes programas, foi possível desenvolver quase todas as funções do sistema e a maioria dos testes, de forma paralela à construção da PCBs e da soldagem dos componentes. A integração entre software e hardware é

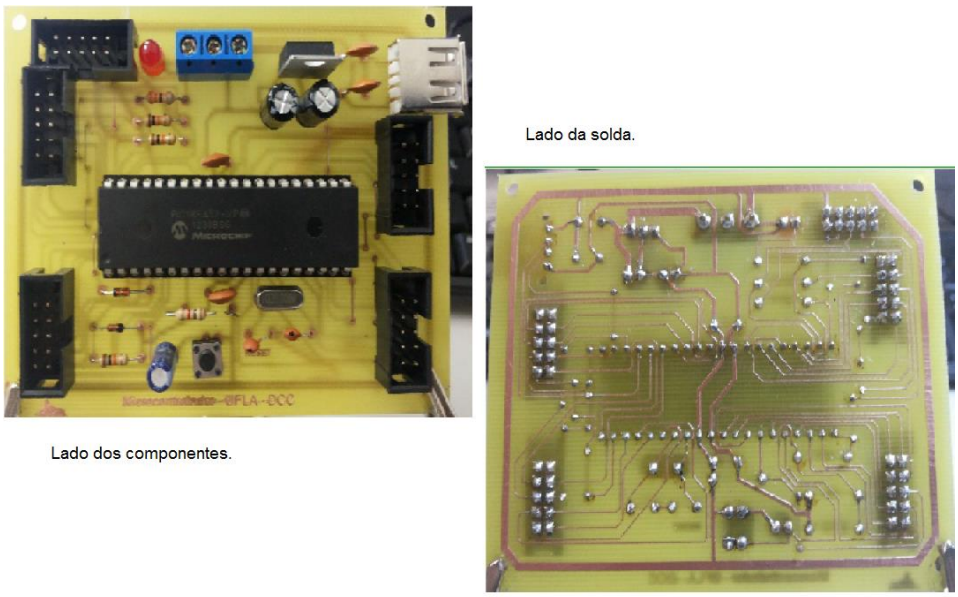


Figura 33 Placa microcontrolador, lado dos componentes e lado da solda

feita pela gravação do *firmware* na memória de programa do microcontrolador. Para isso, é utilizado o programa PICkit 2, fornecido gratuitamente pela *Microchip Technology Inc.*[11]

3.4.5 Testes de validação

Na Figura 35 mostra-se o sistema funcionando em suportes que possibilitaram a realização de vários testes antes de as placas serem acomodadas no gabinete.

Como última etapa de montagem deste trabalho, as placas foram fixadas numa caixa de acrílico, como pode ser visto na Figura 36, os números das casas do tabuleiro foram colocados sobre o painel de LEDs e todos os

[11] O programa PICkit está disponível gratuitamente no site da Microchip: http://www.microchip.com/Developmenttools/ProductDetails.aspx?PartNO=DV164121. Acesso em 11 nov. 2013.

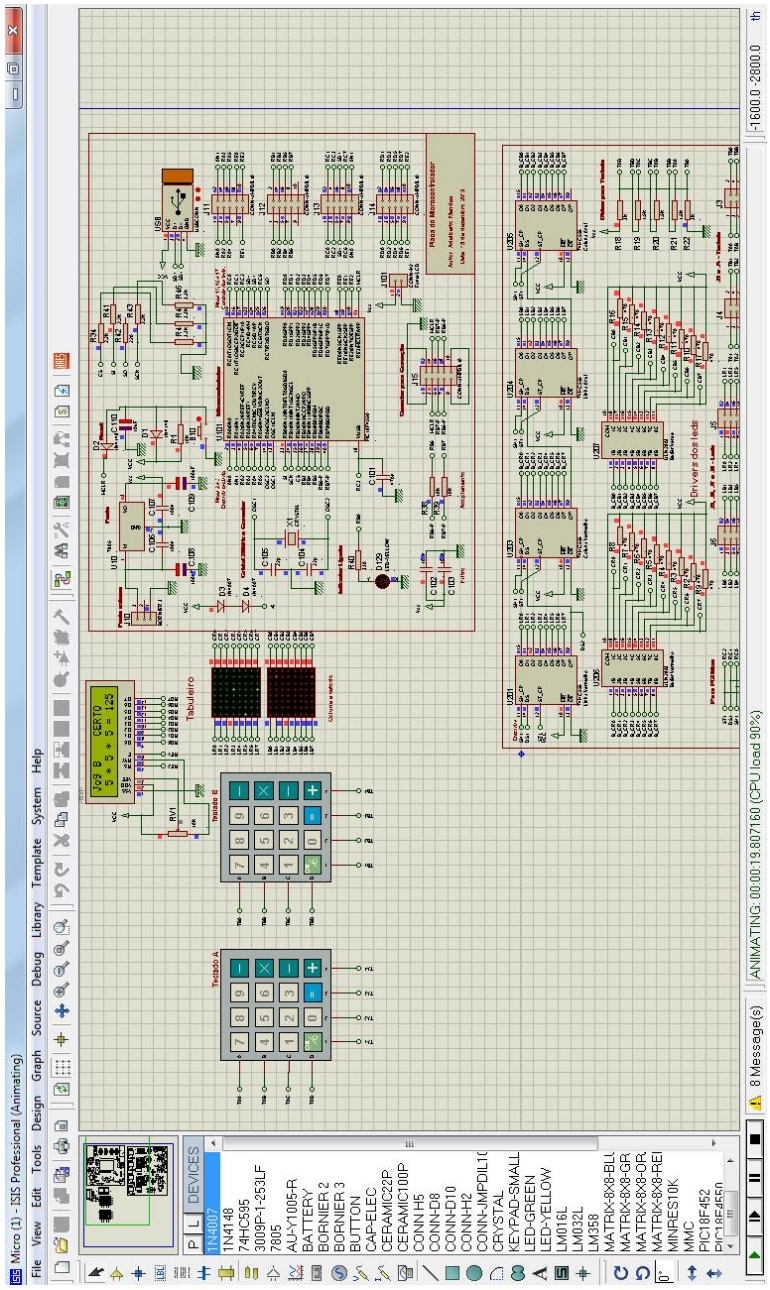


Figura 34 Interface do Proteus® executando a simulação do jogo

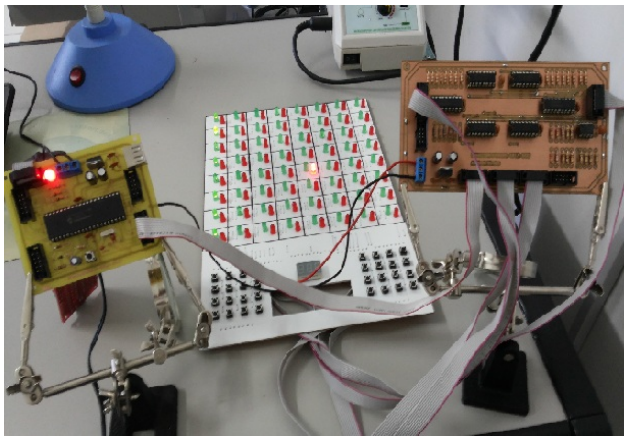


Figura 35 Testes em bancadas com suportes

testes, pontuados pelo documento de requisito do Apêndice E, foram feitos e são descritos no capítulo 4.

Figura 36 Protótipo do jogo Contig 60® finalizado

4 RESULTADOS E DISCUSSÕES

O principal resultado deste trabalho foi o desenvolvimento do protótipo mostrado na Figura 37, um sistema embarcado para o jogo Contig 60®, cujo objetivo é ser utilizado como ferramenta no apoio ao ensino de Matemática.

Figura 37 Protótipo do jogo Contig 60® pronto e funcionando.

A seguir, com base nos requisitos funcionais e não funcionais do

documento de requisito do Apêndice E, são descritos os resultados, de forma pontual, dos testes para validação do equipamento.

RF01 - Escolha do Modo do Jogo.

Assim que o tabuleiro é ligado, o sistema garante que a matriz de LED esteja desligada, limpando todos os *latch* dos registradores de deslocamento. Logo após esse procedimento, aparece no display de LCD a pergunta ao usuário sobre qual modo de jogo ele deseja iniciar. São duas as possibilidades. O modo 1 é iniciado com o tabuleiro todo apagado e pode-se jogar com dois jogadores humanos ou um jogador humano e o agente autônomo, sendo que, a cada jogada, o jogador humano deve acionar a tecla para gerar os três números aleatoriamente, e em seguida, acionar outra tecla que dispare o lance do jogador autônomo. O teste do modo 2 do jogo é descrito logo abaixo, no RF07. Na Figura 38 é possível ver o display com a pergunta inicial do jogo.

Figura 38 Foto do sistema perguntando ao jogador qual Modo do Jogo deve executar.

RF02 - Geração de números aleatórios.

Para os jogadores, sendo eles humanos ou o agente autônomo, devem aparecer no display os números que representem o sorteio dos dados. Assim, a cada lance, o jogador humano deve apertar a tecla *Dado* para que esses números sejam gerados. Na Figura 39 observa-se o teclado e os números gerados no LCD.

Figura 39 LCD mostrando os três números pseudoaleatórios que representam os dados.

RF03 - Indicação no tabuleiro.

O sistema deve monitorar se as operações digitadas estão coerentes com as regras do jogo, isto é, se o valor resultante da operação existe ou não no tabuleiro, indicando se está **Possível** ou **Não Pode!**. Na Figura 40 é mostrado o display com a expressão numérica escolhida pelo jogador. A direita informa que a operação está errada e a esquerda que a operação está

certa, acendendo o LED que corresponda ao resultado da operação.

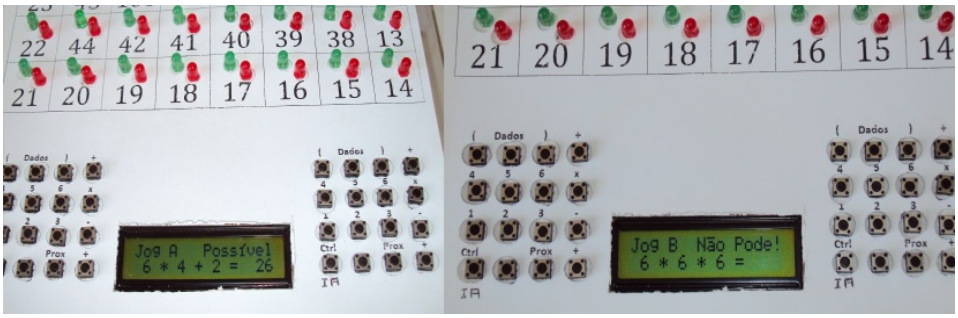


Figura 40 À esquerda, o resultado da operação está certo e à direita, a operação está errada.

RF04 - Jogar com IA.

Jogar com o agente autônomo segue a mesma sequência de acionamento nas teclas. A diferença está em que a operação escolhida é feita pelo jogador autônomo e não por um jogador humano. Assim, no início de cada lance, deve-se acionar a tecla *Dados* para gerar os números aleatórios; se for a vez do jogador humano, este deve digitar a operação matemática escolhida; se for a vez do jogador autônomo, deve-se acionar a tecla *IA* para o sistema gerar a operação matemática. Em ambas as situações, se o resultado estiver certo, acenderá o LED da casa correspondente. Na Figura 41 são mostradas as teclas *IA* e a operação gerada pelo jogador autônomo.

Nos vários testes feitos, algumas vezes, o jogador autônomo não encontrou a melhor jogada, seja por ter excedido o tempo para completar o algoritmo Minimax ou pelo limite do sistema em tratar de recursividade. Nesta situação, o sistema calcula o maior valor possível de ser jogado para o status atual do tabuleiro e faz a jogada. Se, após esse procedimento, não for encontrado nenhum valor possível de ser jogado, é indicada no LCD a impossibilidade de jogar e o sistema cede a vez para o próximo jogador.

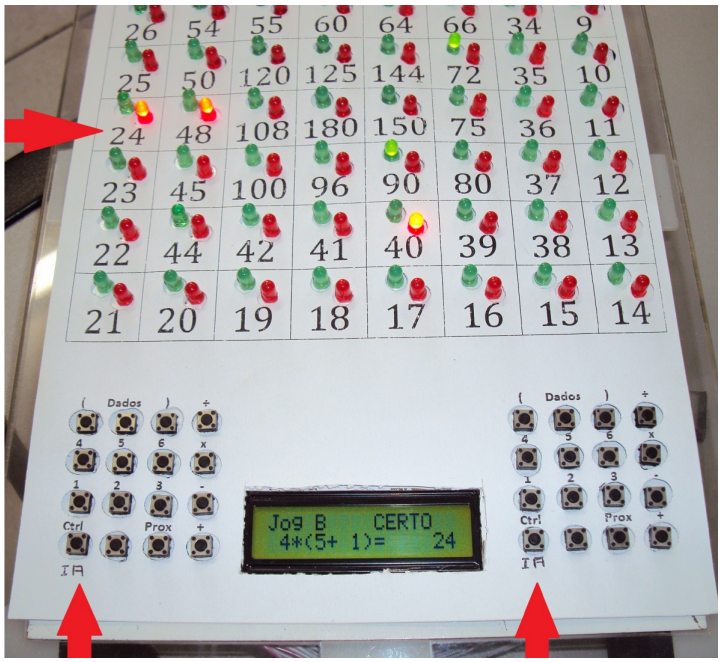


Figura 41 Acionando a tecla IA, o jogador autônomo faz a operação.

RF05 - Registro do jogo.

Se um cartão SdCard estiver no compartimento, o sistema grava todos os lances num arquivo de texto simples, que é mostrado na Figura 42.

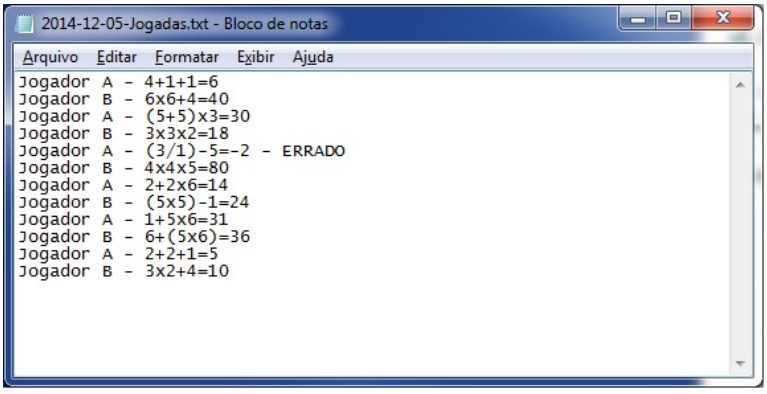


Figura 42 Arquivo com o registro das jogadas.

RF06 - Melhor jogada.

Esta situação é a mesma descrita na RF04, pois, mesmo com dois jogadores humanos, se acionada a tecla *IA*, o sistema irá gerar uma operação matemática adequada que, graças ao algoritmo Minimax e à heurística que gera o peso do tabuleiro, pode ser considerada a melhor jogada.

RF07 - Jogar a partir de um momento específico:

Ao iniciar o jogo, o jogador pode escolher o modo 2. Neste caso, o utilizador pode definir as casas usadas digitando seus números no teclado e iniciar o jogo a partir desta configuração. Na Figura 43 é possível ver a indicação do LCD para incluir as casas com as quais o jogo deve inciar. Também se podem ver alguns LEDs acesos, que representam as casas do tabuleiro que estão ocupadas. Ao acionar a tecla **Ctrl** o jogo é iniciado a partir daquele estado do tabuleiro.

Figura 43 Digitar os números para inciar o jogo.

RNF01 - Desempenho.

Este requisito limita o tempo para que o jogador autônomo gere a melhor expressão numérica em três segundos. Se o algoritmo não conseguir definir a melhor jogada neste tempo, simplesmente escolhe a operação cujo resultado tem o maior valor, ou, se não tiver encontrado nenhuma operação válida, indicar no LCD essa informação e passa a vez para o jogador humano. Outro limite deste requisito é o tempo que o sistema demora em acender o LED da casa correspondente ao resultado da operação logo após ela ser digitada, o que ficou bem menor que o limite estabelecido.

RNF02 - Atributos físicos:

O peso do protótipo é de 1,097 Kg e as suas dimensões são de 350 mm de comprimento, 230 mm de largura e 60 mm de altura (350x230x60). Essas dimensões, que são mostradas na Figura 44, são do gabinete de acrílico no qual as placas foram acondicionadas. É importante observar que a placa de interface do usuário, onde ficam os LEDs, o LCD e o conjunto de teclas, tem exatamente as dimensões de uma folha modelo A4 (297mmx210mm), sendo possível diminuir o tamanho do produto às dimensões desta placa.

RNF03 - Custo do sistema.

Este requisito estabelece o limite financeiro de R$ 100,00, que não foi atendido, pois todo o projeto teve o custo final de R$ 113,95. A justificativa para a quebra deste requisito está no fato de que, durante as iterações da metodologia adotada, houve a necessidade de realizar algumas modificações do projeto inicial, entre elas a troca do microcontrolador PIC 18F452 pelo PIC 18F4550 e o acréscimo de um conector SdCard. É importante observar que o aumento dos gastos não descaracteriza o projeto como sendo de baixo custo, principalmente porque os gastos adicionais, que não foram muitos,

Figura 44 Dimensões da placa de interface com o jogador.

proveram um alto nível de escalabilidade. Por exemplo, a adição do conector SdCard sem a retirada da porta USB possibilita a conexão de um dispositivo *Wifi USB*, tornando viável a criação de um banco de dados centralizado.

RNF04 - Escalabilidade.

Durante toda a metodologia é mostrada a preocupação do autor deste trabalho com a possibilidade de o sistema ser adequado a situações diferentes, implementar melhorias ou, com poucas modificações de hardware, implementar novos jogos no tabuleiro.

Interface com o usuário.

A interface escolhida para a implementação do protótipo foi a da Figura 3 do documento de requisitos, em que é possível ver dois conjuntos de teclas, um display de LCD e um tabuleiro de LEDs com os números nas casas. Entretanto, todo hardware foi desenvolvido pensando na possibilidade da troca de interface com o usuário, com poucas modificações.

5 CONCLUSÕES E PROPOSTA DE CONTINUIDADE

Como resultado deste trabalho, foi desenvolvido um protótipo de um tabuleiro num sistema embarcado do jogo Contig 60®. Os testes realizados cumpriram de maneira satisfatória todos os pontos dos requisitos funcionais e não funcionais descritos no documentos de requisitos (Apêndice E). O sistema foi construído utilizando técnicas bem definidas no desenvolvimento de projeto em sistemas embarcados, proporcionando que esta dissertação seja uma possível fonte de consulta para futuros desenvolvedores.

O objetivo desta pesquisa foi o aprimoramento de uma ferramenta do professor de Matemática para ser utilizada em sala de aula. Não estão no escopo deste trabalho estudos qualitativos do uso deste equipamento com testes entre alunos e professor. Para uma pesquisa qualitativa deste porte, são exigidas competências pedagógicas por parte do pesquisador. No entanto, o protótipo completamente funcional, de acordo com os testes baseados no documento de requisito, será doado ao Departamento de Matemática (DEX) da Universidade Federal de Lavras (UFLA), para posteriores estudos (monografias, dissertações, teses e artigos) sobre seu uso eficiente e possível aprimoramento do sistema.

De modo geral, projetos em sistemas embarcados são multidisciplinares, seja no momento de identificar um problema que pode ser automatizado ou durante a construção do sistema que, normalmente, utiliza conhecimentos específicos de engenharia. Dessa forma, o que se discute neste trabalho é o método utilizado para o planejamento, o projeto e a execução de um equipamento de sistema embarcado. Considerando sua multidisciplinaridade, o trabalho explora possibilidades de integração da área de sistemas embarcados com áreas de licenciatura. Além disso, por meio da definição de

uma metodologia de trabalho, buscou-se sintetizar, nesta dissertação, todo o desenvolvimento do trabalho e, assim, servir de guia para outros projetos semelhantes.

Como proposta de continuidade, o requisito cumprido NRF04, que estimula o projeto a ter alta escalabilidade, possibilita uma série de alterações, como por exemplo, a inclusão de alguma tecnologia assistiva, como alterar o tabuleiro de LED por um painel em Braille; acrescentar um módulo de rede, que interaja via rádio com um servidor de banco de dados; incorporar mais memória ao sistema e, dessa forma, possibilitar a implementação de novos jogos, como damas ou gamão.

Também é possível implementar muitas melhorias, como aprimorar os registros do jogo para criar um banco de dados de muitas partidas, possibilitando implementar ao sistema um modo de jogo que inicie uma partida a partir de outras anteriormente gravadas no SdCArd, ou mesmo dar continuidade a uma partida iniciada no dia anterior. Além disso, a gravação continua das partidas, gerando um grande banco de dados, possibilita o estudo desse jogo para criação de jogadores autônomos com metodologias de redes neurais artificiais.

Na área da Educação Matemática, é possível usar o equipamento em pesquisas sobre as potencialidades desta ferramenta em sala de aula e dessa forma fazer um levantamento das vantagens em relação ao jogo convencional e propor melhorias, como por exemplo, como aprimorar o jogador autônomo de forma a ser usado como instrumento de avaliação.

REFERÊNCIAS

AARSETH, E. Computer gane studies, year one. **Games Studies**, v. 1, n. 1, p. 1–15, 2001. Disponível em: <http://www.gamestudies.org/0101/editorial.html>. Acesso em: 13 mar. 2014.

ALMEIDA, M. E. B. de; VALENTE, J. A. Integração currículo e tecnologias e a produção de narrativas digitais. **Currículo sem Fronteiras**, v. 12, n. 3, p. 57–82, set./dez. 2012.

ALOFS, T.; THEUNE, M.; SWARTJES, I. A tabletop board game interface for multi-user interaction with a storytelling system. In: **Intelligent technologies for interactive entertainment**. [S.l.]: Springer, 2012. p. 123–128.

BERGER, A. S. **Embedded systems design**: an introduction to processes, tools, and techniques. Lawrence, Kansas: CMP Books, 2002. 209 p.

BITTENCOURT, G. **Inteligência artificial**: ferramentas e teorias. 3. ed. Florianópolis: Editora UFSC, 2006. 362 p.

BROUGÉRE, G. A criança e a cultura lúdica. In: _____. **O brincar e suas teorias**. São Paulo: Pioneira Thomson Learning, 2002. p. 19–32.

CALAZANS, N. **Métodos e ferramentas para o projeto de sistemas digitais**. Porto Alegre: Pontifícia Universidade Católica do Rio Grande do Sul, 1995. Disponível em: <https://www.inf.pucrs.br/calazans/undergrad/orgcomp/metodos.pdf>. Acesso em: 10 abr. 2014.

CARRO, L.; WAGNER, F. R. Sistemas computacionais embarcados. In: JORNADA DE ATUALIZÃO EM INFORMÁTICA, 22., 2003, Campinas. **Anais**... Campinas: Unicamp, 2003. Cap. 2, p. 45–94.

CASTRO NETO, H.; JULIA, R. M. S. Ls-draughts-a draughts learning system based on genetic algorithms, neural network and temporal diferences. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION, 2007, Singapure. **Proceedings...** Singapore: Swissôtel/The Stambord, 2007. p. 2523–2529. Disponível em: <http://ieeexplore-.ieee.org/stamp/stamp.jsp?tp=arnumber=4424788>. Acesso em: 10 dez. 2013.

EDWARDS, S. et al. Design of embedded systems: formal models, validation and synthesis. In: GIOVANNI, M. de. (Ed.). **Readings in hardware/software co-design**. Braunschweig, Georgia: Elsevier, 2001. p. 86–107.

ELEUTÉRIO, W. A.; HOVADICH, W. A. A.; BRAGA, E. Q. Controlador lógico programável utilizando PIC18F4550. **e-Xacta**, Belo Horizonte, v. 4, n. 3, p. 159–179, 2012.

GAJSKI, D. D. et al. **Specification and design of embedded systems**. Englewood Cliffs, Upper Saddle River, NJ: PRT Prentice Hall, 1994.

GRANDO, R. C. **O conhecimento matemático e o uso de jogos na sala de aula**. 2000. 224 p. Tese (Doutorado em Educação) – Universidade Estadual de Campinas. Faculdade de Educação, Campinas, 2000. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=vtls000223718>. Acesso em: 15 mar. 2014.

GRANDO, R. C. **O jogo e a matemática no contexto da sala de aula**. 2. ed. São Paulo: Paulus, 2004. 120 p. Disponível em: <http://books.google.com.br/books?id=rfOscQAACAAJ>. Acesso em: 07 ago. 2013.

IBRAHIM, D. Memory cards. In: ———. **SD Card Projects Using the PIC Microcontroller**. [S.l.]: Elsevier Science, 2010. (IT Pro). Cap. 3, p. 107–136.

JACKSON, D. J.; CASPI, P. Embedded systems education: future directions, initiatives, and cooperation. **ACM SIGBED Review**, v. 2, n. 4, p. 1–4, 2005.

KEUTZER, K. et al. System level design: orthogonalization of concerns and platform-based design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits & Systems**, New York, v. 19, n. 12, p. 1523–1543, Dec. 2000.

KOOPMAN, P. et al. Undergraduate embedded system education at carnegie mellon. **ACM Transactions on Embedded Computing Systems, TECS**, New York, v. 4, n. 3, p. 500–528. Aug. 2005.

MARCO, F. F. d.; FREITAS, M. T. M.; TAVARES, M. O professor de matemâtica e a sua atuação frente à utilização de tecnologia de informação e comunicação na cidade de Uberlândia. **Matemática e Estatística em Foco**, v. 1, n. 1, p. 1–11, 2013.

MARGOLIS, M. **Arduino Kochbuch**. 1. ed. Ausgabe: O 'Reilly Verlag, 2012. 624 p. Disponível em: <http://www.onleihe.de/static/content/oreilly/20120806/978-3-86899-354-7/v978-3-86899-354-7.pdf>. Acesso em: 02 nov. 2014.

MARTINS, N. A. **Sistemas microcontrolados** - uma abordagem com o microcontrolador PIC16F84. São Paulo: Novatec, 2005. Disponível em: <http://www.martinsfontespaulista.com.br/anexos/produtos/capitulos-/203289.pdf>. Acesso em: 13 set. 2013.

MATOS, T. F. C.; VAILANT, E. P.; LIMA, C. M. de. Vencendo desafios na escola básica... o projeto de oficinas de matemática. In: CONGRESSO DE PESQUISA, ENSINO E EXTENSÃO - COMPEEX, 8., 2011, Jataí. **Anais...** Jataí: UFG, 2012. Disponível em: <http://sbpcnet.org.br-/livro/63ra/conpeex/pibid/trabalhos-pibid/pibid-thayza-ferreira.pdf>. Acesso em: 21 jul. 2013.

MENDES, R. M.; GRANDO, R. C. As potencialidades pedagógicas do jogo computacional Simcity 4 para a apropriaço/mobilização de conceitos matemáticos. In: REUNIÃO ANUAL DA ASSOCIAÇÃO NACIONAL DE PÓS-GRADUAÇÃO E PESQUISA EM EDUCAÇÃO, 2006, Caxambu. **Anais...**: Rio de Janeiro: Anped, 2006

MENDES, R. M.; GRANDO, R. C. O jogo computacional simcity 4 e suas potencialidades pedagógicas para as aulas de matemática. **Zetetiké**, Campinas, v. 16, n. 29, p. 118–154, jan./jun. 2008.

MICROCHIP TECHNOLOGY **PIC18F4550 Datasheet Microcontrollers**. Chandler: [s.n.], 2006. 428 p. Disponível em: <http://ww1.microchip.com/downloads/en-/devicedoc/39632c.pdf>. Acesso em: 01 nov. 2013.

MISKULIN, R. G. S. et al. A prática do professor que ensina matemática e a colaboração: uma reflexão a partir de processos formativos virtuais. **Boletim de Educação Matemática**, São Paulo, v. 25, n. 41, p. 173–186, 2011.

MORALES, M. et al. A. **Worldwide Intelligent Systems 2011–2015 Forecast**: the tnext big opportunity. Massachusetts: International Data Corporation-IDC, 2011. Disponível em: <http://www-.idc.com/research/viewtoc.jsp?containerId=230242>. Acesso em: 10 abr. 2013.

MORBACH, R. P. C. **Ensinar e jogar**: possibilidades e dificuldades dos professores de matemática dos anos finais do ensino fundamental, 2012. 175 p. Dissertação (Mestrado em Educação Matemática) – Universidade de Brasília - Faculdade de Educação, 2012.

MÜCK, T. R. **Projeto unificado de componentes em hardware e software para sistemas embarcados.** 2013. 135 p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianopólis, 2013.

NAKASHIMA, R. H. R.; AMARAL, S. F. d. Indicadores didático - pedagógicos da linguagem interativa da lousa digital. **Cadernos de Educação**, n. 37, p. 381–415, set./dez. 2012

OLIVEIRA, A. I.; OLIVEIRA, P. M. **Dificuldades na implantação de tecnologias na sala de aula de uma escola pública.** Uberaba. PIBID - UFTM, 2009.

OSSADA, J. C.; MARTINS, L. E. G. Um estudo de campo sobre o estado da prática da elicitação de requisitos em sistemas embarcados. In: WORKSHOP EM ENGENHARIA DE REQUISITOS – WER, 10., 2010, Cuenca. **Anais...** Cuenca: [s.n.], 2010. p. 30–41.

OSSADA, J. C. et al. Gerse: guia de elicitação de requisitos para sistemas embarcados. In: WORKSHOP EM ENGENHARIA DE REQUISITOS – WER, 12., 2012, Cuenca. **Anais...** Buenos Aires: [s.n.], 2012. p. 1–14.

OTTLEY, A. **ECG compression for holter monitoring**. 2007. 148 p. Tese (Doutorado em Engenharia Elétrica) – University of Saskatchewan, Saskatchewan, 2007.

RUSSELL, S.; NORVIG, P. **Inteligência artificial**: uma abordagem moderna. 2. ed. São Paulo: Editora Campus, 2003. 1040 p.

SALGADO, M. U. C. et al. **Tecnologias na educação**: ensinando e aprendendo com as TIC – guia do formador. 2. ed. Brasília: Ministério da Educação, Secretaria de Educação a Distância, 2010. 120 p.

SILVA, G. C. M. da. O ensino e aprendizagem de expressões numéricas para 5ª série do Ensino Fundamental com a utilização do jogo CONTIG 60. **Educação Matemática Pesquisa**: revista do programa de estudos e pós-graduados em educação matemática, São Paulo, v. 11, n. 1, 2009.

SILVA, G. C. M. da; SILVA, M. J. F. da. O jogo Contig 60, as expressões numéricas e os registros de representação semiótica. **Dossiê**: relações entre pesquisa e prática docentes, v. 27, n. 1, p. 61, 2009.

STAA, B. v. Computadores móveis na escola: reação de pais, alunos e professores. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2009. **Anais...** v. 1, n. 1. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/download/1143/1046>. Acesso em: 10 abr. 2014.

STOFFEL, J. H.; VITÓRIA, W. A. da; SILVA, S. A. F. da. Aplicação do jogo contig 60 em turmas de 8ª série. In: ENCONTRO NACIONAL PIBID MATEMÁTICA, 2.; EIEMAT, 4., 2014. **Anais...** Disponível em: <http://w3.ufsm.br/ceem/eiemat/Anais/arquivos/ed_4/RE/RE_Stoffel_Jenifer.pdf> Acesso em: 01 nov. 2014.

TOSCANO, P. W. Inteligência artificial busca no espaço de estados. In: _____. **Engenharia mecatrônica e de sistemas mecânicos**. São Paulo: POLI-USP, 2009. p. 12.

VARGAS, R. et al. **Sistemas embarcados**: acoplamento do soft-core plasma ao barramento OPB de um PowerPC 405. Florianópolis: UFSC, 2007.

WILMSHURST, T. **Designing embedded systems with PIC microcontrollers**: principles and applications. [S.l.]: Elsevier Science, 2009. 661 p.

WOLF, M. **Computers as components**: principles of embedded computing system design. [S.l.]: Morgan Kaufmann. Elsevier, 2012. (The Morgan Kaufmann Series in Computer Architecture and Design Series).

WOLF, W.; MADSEN, J. Embedded systems education for the future. **Proceedings of the IEEE**, v. 88, n. 1, p. 23–30, 2000. Paper.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: theory and practice. **The Knowledge Engineering Review**, Cambridge, v. 10, n. 02, p. 115–152, 1995.

WU, M.; BALAKRISHNAN, R. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In: ANNUAL SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 16., 2003, Vancouver. **Proceedings...** New York: ACM, 2003. p. 193–202.

YANG, H.; BERGMANS, J. W.; SCHENK, T. C. Illumination sensing in LED lighting systems based on frequency-division multiplexing. **IEEE Transactions on Signal Processing**, v. 57, n. 11, p. 4269–4281, 2009.

ZURITA, M. E. **Projeto de sistemas embarcados**. Teresina: Universidade Federal do Piauí, 2011. (Curso de Engenharia Elétrica – Campus Universitário Ministro Petrônio Portela).

APÊNDICES

APÊNDICE A - Esquema elétrico completo

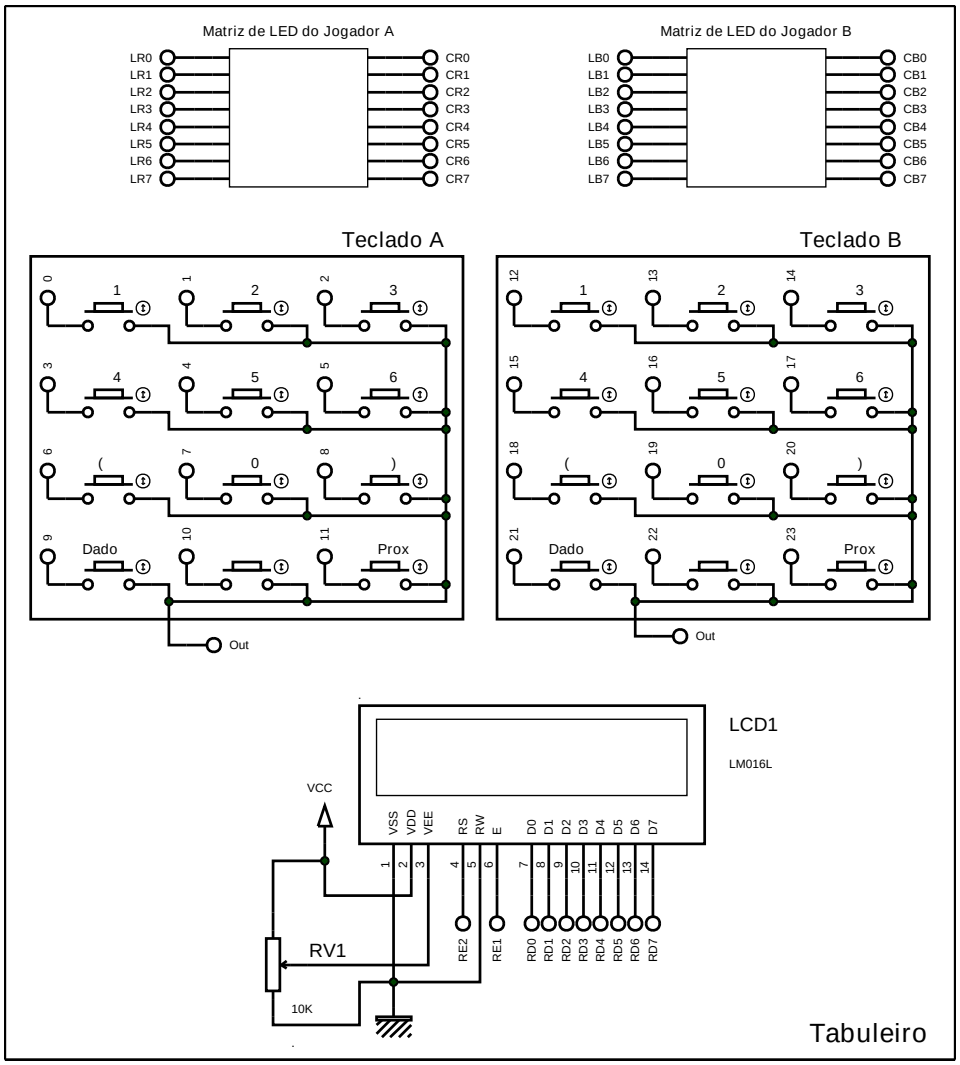


Figura 45 Esquema elétrico completo do Tabuleiro.

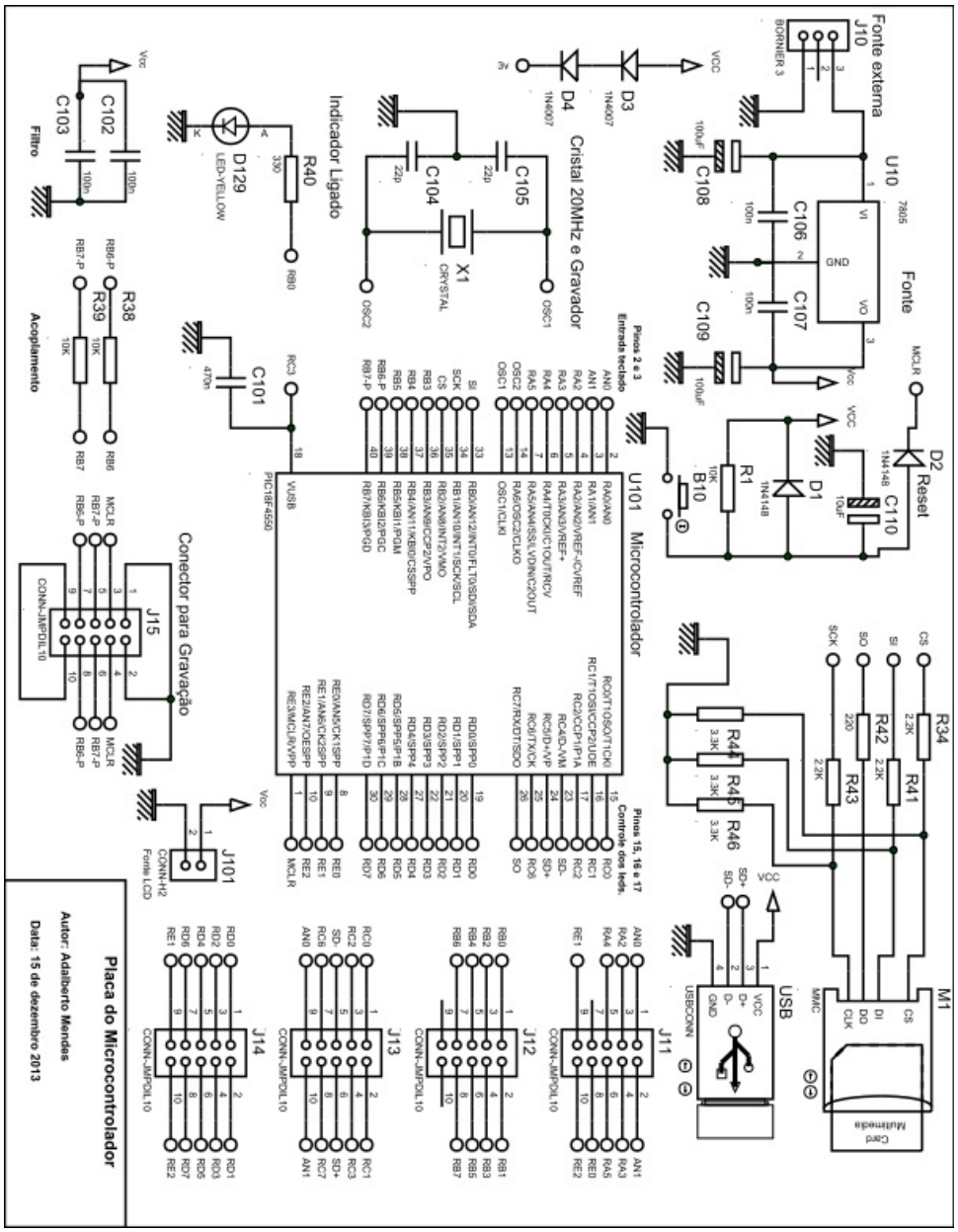


Figura 46 Esquema elétrico completo do Microcontrolador.

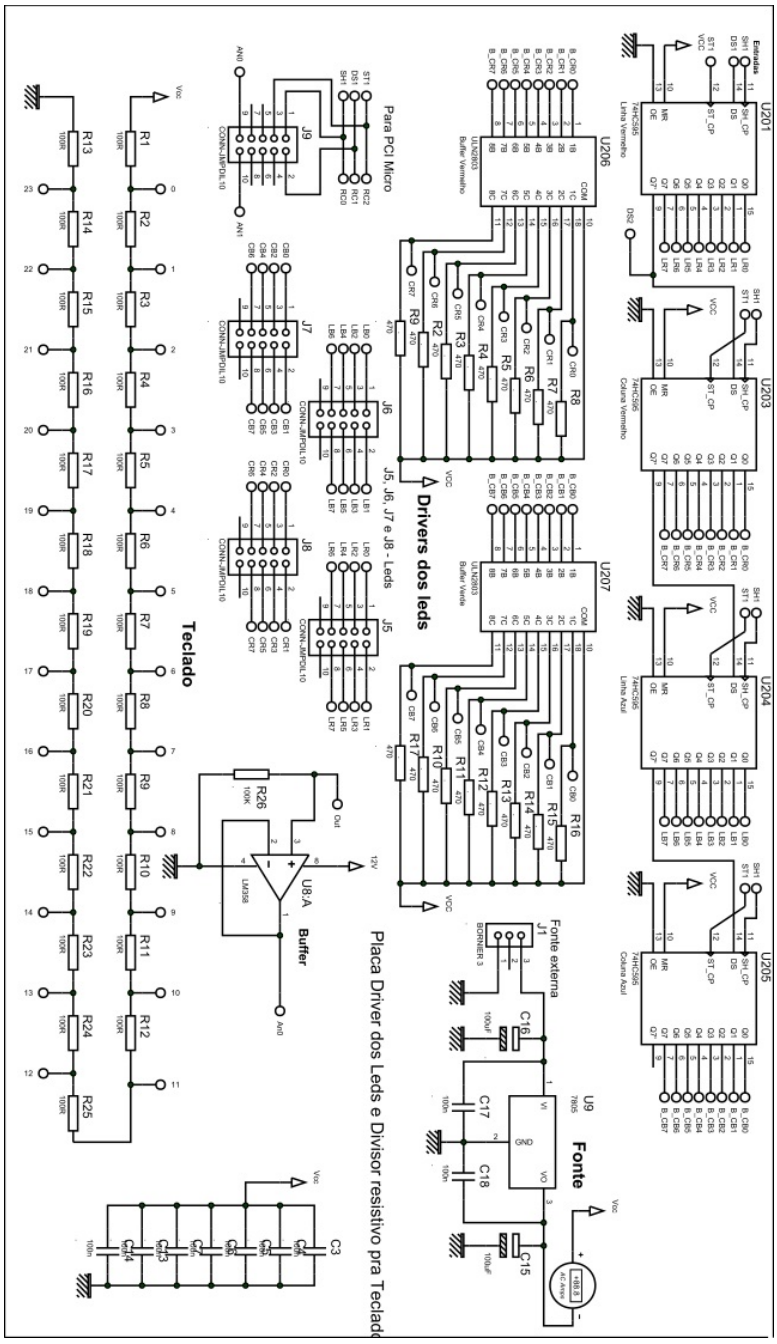


Figura 47 Esquema elétrico completo dos drivers.

APÊNDICE B - Placas de circuito (PCB)

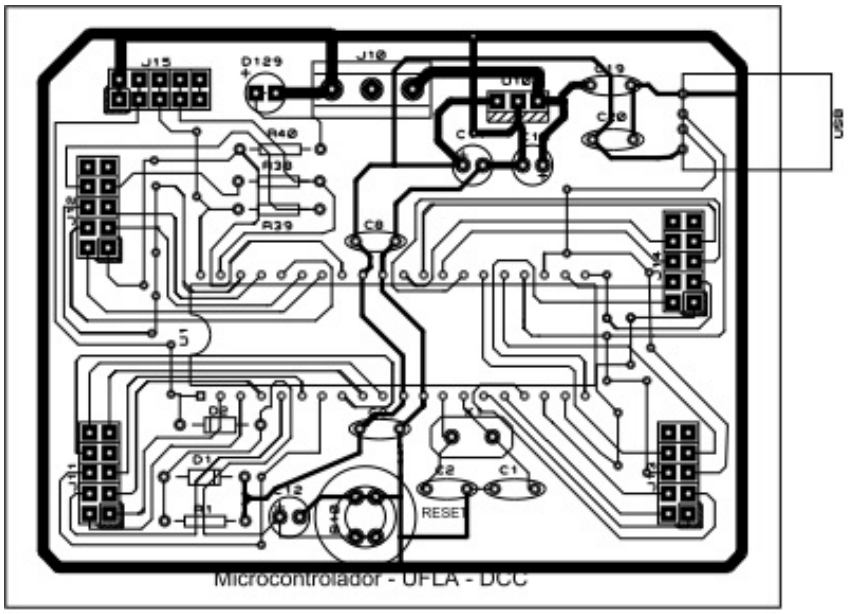


Figura 48 PCB do Microcontrolador.

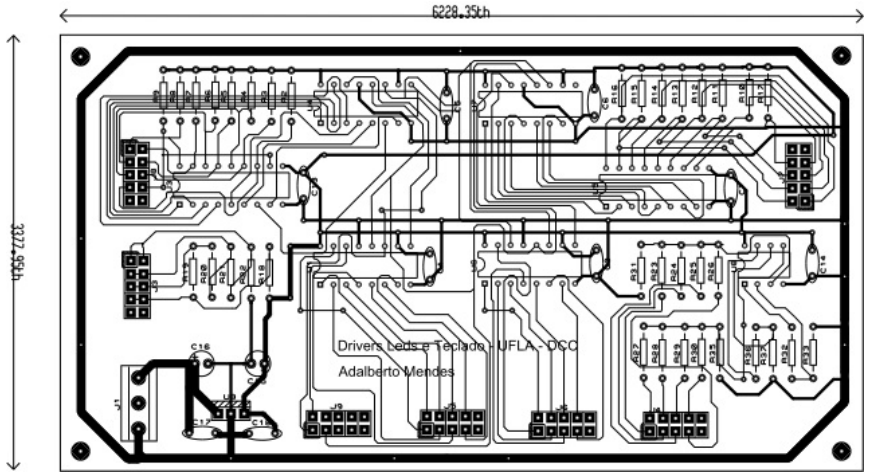


Figura 49 PCB doe controles dos LEDs e teclado.

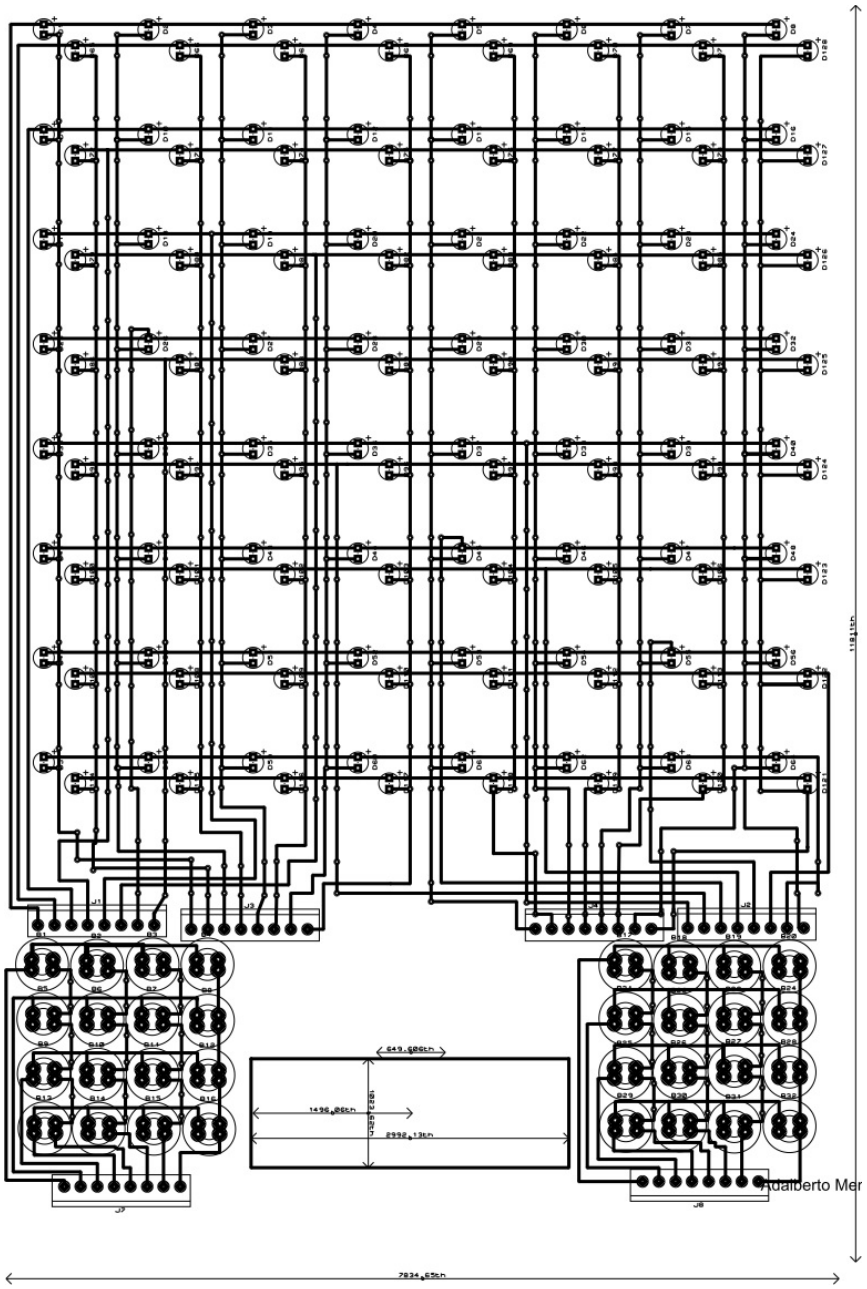


Figura 50 PCB da interface com o usuário.

APÊNDICE C - Lista de materiais

Bill Of Materials For PCB Microcontrolador

Design Title : PCB Microcontrolador
Author : Adalberto Mendes
Revision : 1.5
Design Created : sexta-feira, 3 de outubro de 2014
Design Last Modified : terça-feira, 21 de outubro de 2014
Total Parts In Design : 32

1 Modules

Quantity:	References	Value	Order Code
1	M1	MMC	

10 Resistors

Quantity:	References	Value	Order Code
3	R1, R38, R39	10K	
3	R34, R41, R43	2.2K	
1	R42	220	
3	R44-R46	3.3K	

8 Capacitors

Quantity:	References	Value	Order Code
1	C101	470n	Maplin DS80B
2	C104, C105	22p	Maplin WX48C
2	C106, C107	100n	Maplin WX48C
2	C108, C109	100uF	
1	C110	10uF	

2 Integrated Circuits

Quantity:	References	Value	Order Code
1	U10	7805	
1	U101	PIC18F4550	

2 Diodes

Quantity:	References	Value	Order Code
2	D1, D2	1N4148	

9 Miscellaneous

Quantity:	References	Value	Order Code
1	B10		
1	J10	BORNIER 3	
5	J11-J15	CONN-JMPDIL10	
1	USB	USBCONN	
1	X1	CRYSTAL	

terça-feira, 21 de outubro de 2014 17:14:21

Figura 51 Lista de materiais da PCB do Microcontrolador.

Bill Of Materials For Driver Dos LEDs E Controle Do Teclado

Design Title : Driver dos LEDs e controle do teclado
Author : Adalberto Mendes
Revision : 1.5
Design Created : sexta-feira, 3 de outubro de 2014
Design Last Modified : terça-feira, 21 de outubro de 2014
Total Parts In Design : 67

42 Resistors

Quantity:	References	Value	Order Code
16	R2-R17	470	
1	R26	100K	M100R
25	R101-R125	100R	M100R

11 Capacitors

Quantity:	References	Value	Order Code
7	C3-C7, C13, C14	100n	Maplin WX56L
2	C15, C16	100uF	
2	C17, C18	100n	Maplin WX48C

8 Integrated Circuits

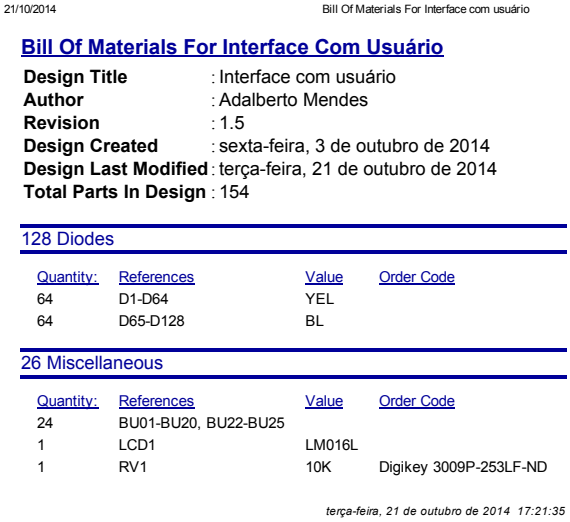
Quantity:	References	Value	Order Code
1	U8	LM358	
1	U9	7805	
4	U201, U203-U205	74HC595	
2	U206, U207	ULN2803	

6 Miscellaneous

Quantity:	References	Value	Order Code
1	J1	BORNIER 3	
5	J5-J9	CONN-JMPDIL10	

terça-feira, 21 de outubro de 2014 17:17:09

Figura 52 Lista de materiais da PCB de controle dos LEDs.



21/10/2014 Bill Of Materials For Interface com usuário

Bill Of Materials For Interface Com Usuário

Design Title : Interface com usuário
Author : Adalberto Mendes
Revision : 1.5
Design Created : sexta-feira, 3 de outubro de 2014
Design Last Modified : terça-feira, 21 de outubro de 2014
Total Parts In Design : 154

128 Diodes

Quantity:	References	Value	Order Code
64	D1-D64	YEL	
64	D65-D128	BL	

26 Miscellaneous

Quantity:	References	Value	Order Code
24	BU01-BU20, BU22-BU25		
1	LCD1	LM016L	
1	RV1	10K	Digikey 3009P-253LF-ND

terça-feira, 21 de outubro de 2014 17:21:35

file:///C:/Users/UFLA/Google%20Drive/02-Faculdade/01-UFLA/01-Mestrado/01-Projeto%20Mestrado/01-Jogos%20Educacionais/07-Defesa/img/Proteus/... 1/1

Figura 53 Lista de materiais da PCB da interface com usuário.

APÊNDICE D - Partes do firmware

C:\Users\UFLA\Desktop\MatrixLed.txt sexta-feira, 27 de fevereiro de 2015 10:47

```
1  void interrupt(){                                // Rotina de interrupção, manda os
   bits para a matriz de LEDs.
2  ...                                              // Variáveis auxiliares
3      Ser_Out = 0;
4          for(Aux = 0; Aux < 8; Aux ++){
5              if ((Lin_azul % 2) != 0){            // Se for "ímpar", manda um em...
6                  Ser_Out = 1;                     // Coloca um na saída serial
7                  Clock = 1;                       // Clock vai para nível 1
8                  Delay_us(Time);                  // Tempo do pulso
9                  Clock = 0;                       // Clock vai para nível 0
10             }else{                               // Se for "par", manda zero em...
11                 Ser_Out = 0;                     // Coloca zero na saída serial
12                 Clock = 1;                       // Coloca prepara ...
13                 Delay_us(Time);                  // Tempo do pulso
14                 Clock = 0;                       // Para o próximo pulso
15             }
16         Lin_azul = (Lin_azul >> 1);
17         }
18     for(Aux = 0; Aux < 8; Aux ++){               // Mesmo comentário anterior
19         if ((coluna_a[interr] % 2) != 0){
20             ...                                  // Mesmo laço anterior
21         }
22     coluna_a[interr] = (coluna_a[interr] >> 1);
23     }
24     for(Aux = 0; Aux < 8; Aux ++){               // Mesmo comentário anterior
25         if ((Lin_verde % 2) != 0){
26             ...                                  // Mesmo laço anterior
27         }
28     Lin_verde = (Lin_verde >> 1);
29     }
30     for(Aux = 0; Aux < 8; Aux ++){               // Mesmo comentário anterior
31         if ((coluna_v[interr] % 2) != 0){
32             ...                                  // Mesmo laço anterior
33         }
34     coluna_v[interr] = (coluna_v[interr] >> 1);
35     }
36     Ser_Out = 0;                              // Sinal de dados serial
37     Latch = 1;                                // Libera saída do 74HC595 (Latch)
38     Delay_us(Time);                           // Largura do pulso de Latch
39     Latch = 0;                                // Trava saída
40     interr ++ ;                               // Contador de interrupção
41 }
```


Figura 54 Parte do código de interrupcão - Controle matriz de LEDs.

C:\Users\UFLA\Desktop\RotinaInterrupcao(1).txt sexta-feira, 27 de fevereiro de 2015 10:56

```
1  // Rotina de interrupção, auxilia geração dos sorteio dos Dados.
2  void interrupt(){
3      if (interr == 8){                        // Contador de interrupção
4          interr = 0;                          // Usado para contar linhas/colunas
5          if (da2 == 6){                       // Altera contagem dado 2.
6             da2 = 1;
7          }
8      }
9      da1 = da1 - 1;                           // Contagem dado 1 - regressiva
10     if (da1 == 1){
11         da1 = 6;
12     }
13     da2 = da2 + 1;                           // Contagem dado 2 - progressiva
14     if (da2 == 7){                           // Com mudança na contagem de
15         da2 = 1;                             // interrupções.
16     }
17     da3 = da3 + 1;                           // Contagem dado 3 - progressiva
18     if (da3 == 7){
19         da3 = 1;
20     }
21     TMR0L = 0xEC;                            // Valor inicial de contagem
22     TMR0H = 0xFF;                            // Valor final de contagem
23     INTCON.TMR0IF = 0;                       // Apaga flag de estouro do timer0
24 /*
25     Calculo para interrupção:
26     TempoEstouro = Periodo_CicloMaquina * Prescales * (16_bits - CargaInicial)
27     Para 1 segundo = 1.000.000us
28     Para Clock de 20Mhz = 0,2us
29     1.000.000us = 0,2 * 128 * (65536 - X)
30     1.000.000us / (0,2 * 128) = 65536 - X
31     X = 49911 = CargaInciial -> são os dois registradores TMR0L e TMR0H
32     HEX = C2F7 => L=F7 e H=C2
33     Para 0,5    = 46005 = B3B4 = L=B4 e H=B3
34     Para 0,05   = 63583 = F85E = L=5E e H=F8
35     Para 0,01   = 65145 = FE79 = L=79 e H=FE
36     Para 0,005  = 65341 = FF3C = L=3C e H=FF
37     Para 0,001  = 65497 = FFD8 = L=D8 e H=FF
38     Para 0,0005 = 65516 = FFEC = L=EC e H=FF
39     Para 0,0001 = 65532 = FFFC = L=FC e H=FF
40 */
41 }
```

-1-

Figura 55 Parte do código de interrupcão - Auxilia gerar número dos Dados.

C:\Users\UFLA\Desktop\StatusTabuleiro.txt sexta-feira, 27 de fevereiro de 2015 10:36

```
1  void StatusTab(int *jogador){       // Função que estabelece Peso do tabuleiro no
   momento.
2  ...                                                 // VariÃ¡veis locais
3  if jogador == 0{                                    // Jogador MAX
4      for(l=0; l<8; l++){                             // Varre matriz
5          for(c=0; c<8; c++){
6              if tab[l][c].ocupado == 0{              // Se for ocupado peso 0
7                  tab[l][c].peso = 0;
8              }else{
9                  tab[l][c].peso = 10;
10                 tab[l][c-1].peso = 5;
11                 tab[l][c+1].peso = 5;
12                 tab[l-1][c-1].peso = 5;
13                 tab[l-1][c].peso = 5;
14                 tab[l-1][c+1].peso = 5;
15                 tab[l+1][c-1].peso = 5;
16                 tab[l+1][c].peso = 5;
17                 tab[l+1][c+1].peso = 5;
18             }
19         }
20     }
21 if jogador == 1{                                          // Jogador MIN
22     ...                                                  // Mesmo que anterior,
       mas negativos
23 }
24
25     for(l=0; l<8; l++){                                  // Varre matriz
26         for(c=0; c<8; c++){
27             pesoTab[raiz]= pesoTab[raiz] + tab[l][c].pes;   // Soma os pesos das
               casas = peso Tabuleiro
28         }
29     }
30 }
```


Figura 56 Parte do código para gerar o peso dos nós - Status do Tabuleiro.

C:\Users\UFLA\Desktop\Minmax.txt sexta-feira, 27 de fevereiro de 2015 10:37

```
1  //*****************************************************************************
2  // Algoritmo MINIMAX
3  int Minimax(int *valor, int Profundidade, int tempo){   // Cria a árvore e acha
   melhor jogada
4  ...                                                     // Variáveis auxiliares
5      if(tempo > 3){                                      // Se estourar tempo
6          StatusTabuleiro(0);                             // Calcular peso folha
7          Calculo(1);                                     // Calcular maior valor
           possível com operandos
8          return(0);
9      }
10     if(Profundidade == 0){                              // Se nível 0 árvore
11         StatusTabuleiro(0);                             // Calcular peso raiz
12         return(1);                                      // Retorna 1 para continuar
           na árvore
13     }
14     if(Profundidade == 1){                              // Se nível 1 da árvore
15         Calculo(2);                                     // Calcular possíveis
           resultados e voltar
16         StatusTabuleiro(1);                             // Acertar árvore
17             if(pesoTab[raiz] < pesoTab[raiz-1]){
18                 return(pesoTab[raiz])
19             } else {
20                 return(pesoTab[raiz-1])
21             }
22     }
23     if(Profundidade == 2){                              // Se nível 2 da árvore -
       Não tem operandos
24             if(pesoTab[raiz] > pesoTab[raiz-1]){
25                 calculo(3);                             // Calcular para todas as
                   casas do tabuleiro
26                 ...
27                 return(pesoTab[raiz])                   // Retorna peso nó esquerdo
28             } else {
29                 calculo(3);                             // Calcular para todas as
                   casas do tabuleiro
30                 ...
31                 return(pesoTab[raiz+1])                 // Retorna peso nó direito
32             }
33     }
34     if(Profundidade == 3){                              // Se nível 3 da árvore -
       Não tem operandos
35             if(pesoTab[raiz] > pesoTab[raiz-1]){
36                 calculo(3);                             // Calcular para as melhores
                   possíbilidades
37                 ...
38                 return(pesoTab[raiz])
39             } else {
40                 ...
41                 return(pesoTab[raiz+1])
42             }
43     }
44 }
45
46
47
```

-1-

Figura 57 Parte do código do jogador autônomo - Minimax.

APÊNDICE E - Documento de Requisitos

ADALBERTO MENDES

DOCUMENTO DE REQUISITOS

LAVRAS - MG

2014

Versão 1.3

Lista de Figuras

SUMÁRIO

1 Introdução

Este documento tem como finalidade especificar, de maneira clara e concisa, os requisitos e as funcionalidades de um sistema embarcado do Jogo Contig 60®. Em cada uma das etapas do desenvolvimento do sistema, este documento deve orientar o desenvolvedor e, se necessário, proceder a quantas atualizações forem necessárias para alcançar o objetivo geral. Também tem como finalidade nortear testes de validação e possíveis correções.

1.1 Escopo

O Jogo Contig 60® é muito usado por professores como ferramenta para o ensino da matemática. Implementar este jogo num sistema embarcado tem como principal função auxiliar o professor no uso desta ferramenta.

1.2 Visão geral do documento

No capítulo 2 encontra-se uma descrição geral do jogo no sistema embarcado, abordando as partes referentes à visão geral do sistema, as regras do Jogo Contig 60®, os momentos do jogo, as perspectiva do produto e as suas restrições. No capítulo 3 são descritos os requisitos funcionais e não funcionais, além da interface do sistema com o usuário.

2 Descrição Geral do Sistema

O sistema embarcado do Contig 60® deve implementar as regras deste jogo, e assim possibilitar que seja jogado substituindo os materiais descrito na subseção 2.1. Além disso, deve auxiliar o professor, que ensina Matemática com jogos, na utilização do conceito de *momentos do jogo* (2.2), que pelas pesquisas potencializam o aprendizado de conceitos matemáticos.

2.1 Regras do jogo

Material:

Tabuleiro; 50 fichas, sendo 25 de uma cor e 25 de outra cor; 3 dados.

Objetivo:

O jogador deve conquistar uma sequência de cinco fichas nas casas do tabuleiro, na horizontal, vertical ou na diagonal.

Regras:

1. Os jogadores ou duplas de jogadores jogam de forma alternada, cada um em sua vez lança os dados e com os números devem fazer uma sentença matemática, usando as quatro operações matemáticas básicas (soma, subtração, multiplicação e divisão). Por exemplo, se os dados lançados derem os números 3, 1 e 5, o jogador poderá construir a sentença (3 + 1) x 5 = 20. Neste caso, o jogador deverá colocar sua ficha na casa marcada com o número 20 do tabuleiro.

2. Se um jogador efetuar uma operação errada, o adversário pode acusar o erro. Acontecendo isto, a partida continua com o próximo jogador, sendo o jogador que errou penalizado em perder sua vez de jogar.

3. Se um jogador acreditar que, os números sorteados pelos dados é impossível fazer uma sentença que, como resultado, coincida com uma

das casas vagas do tabuleiro, este abre mão de sua vez. Nesse caso, se o adversário conseguir fazer uma operação com resultado válido, ainda pode jogar novamente, sendo que é sua vez normal do jogo.

4. Uma forma de ganhar o jogo é conquistando 5 casas em linha reta no tabuleiro, podendo ser na horizontal, vertical ou diagonal.

2.2 Momentos do jogo

O objetivo da utilização do conceito de *momentos de jogo* é potencializar a assimilação, por parte do aluno, de conceitos matemáticos envolvido no jogo.

1. Familiarização com o material do jogo;
2. Reconhecimento das regras do jogo;
3. O Jogo pelo jogo;
4. Intervenção pedagógica verbal;
5. Registro do jogo;
6. Intervenção pedagógica escrita;
7. Jogar com competência.

2.3 Perspectivas do produto

O sistema embarcado deste jogo deve apresentar um tabuleiro com as mesmas características do tabuleiro original (64 casas numeradas). Cada casa, além dos números, deve conter 2 LEDs de cores diferentes, cada cor representando um dos jogadores. Junto ao tabuleiro deve existir um teclado numérico, mais as quatro operações matemáticas (+, -, x, /), os parênteses e algumas funções especiais. Também deve existir um *display* de LCD, cuja função é dar informações do status do jogo, por exemplo, qual jogador

deve jogar, quais os três números sorteados para formar a operação, qual a operação o jogador fez e se o resultado esta correto. O sistema deve ter um dispositivo *SdCard* ou saída USB, e assim gravar em algum dispositivo um arquivo de texto, com o registro do jogo.

2.4 Funções do Produto

Cada jogador, na sua vez, deve receber pelo *display* de LCD três números, de 1 a 6, que corresponde ao lançamento de três dados. De posse destes números, o jogador deve desenvolver uma operação matemática com as quatro operações básica, inclusive com o uso de parênteses, cujo resultado deve ser equivalente ao número impresso numa das casas do tabuleiro. Ao digitar essa operação no teclado, o sistema vai apresentar o resultado no LCD e acender o LED da casa do tabuleiro cujo número corresponda ao resultado da operação, o LED acesso será da cor do jogador.

As teclas, os LEDs e o LCD devem ser dispostos no tabuleiro de forma a possibilitar que cada jogador seja representado por um time de dois ou três jogadores.

Cada lance deve ser gravada num *SdCard*, pendrive ou num computador que está ligado a saída USB do sistema, mas conectar ou não qualquer dispositivo nessa saída não deve impedir o sistema de funcionar.

O sistema deve prover a possibilidade de treinar o jogador, para isso, será criado um jogador autômato, que deve jogar com razoável competência.

Para auxiliar o professor nos momentos de intervenção, o sistema deve mostrar, ao digitar algumas teclas, qual a melhor jogada possível para aquele momento do jogo.

2.5 Restrições gerais do sistema

O sistema será montado com apenas um jogo, o Contig 60®. Nada impede de ser implementado outros jogos neste tabuleiro, mas isto não esta no escopo deste trabalho.

Apesar do Contig 60® ser considerado um jogo de estratégia e que as tecnologias de criação de jogador autônomo serem muito avançadas para este tipo de jogo, não é premissa do sistema que o jogador autônomo seja invencível.

2.6 Descrição dos *Stakeholders*

Professor: O professor ou orientador que utiliza jogos para o ensino de conceitos matemáticos, deve conhecer bem as regras do jogo e como utilizar os *momentos do jogo* para potencializar o ensino desses conceitos.

Aluno: O aluno do Ensino Fundamental, ou qualquer pessoa que esteja aprendo conceitos matemáticos que envolvam as quatro operações e a utilização de parenteses.

3 Requisitos do sistema

Os requisitos definem os serviços que o sistema deve oferecer. Eles são divididos em duas partes: os requisitos funcionais descrevem as funcionalidades que o sistema deve oferecer aos usuários e os requisitos não funcionais descrevem as características básicas do produto. Para o desenvolvedor, os requisitos norteiam como o projeto deve ser administrado até a sua conclusão.

3.1 Requisitos funcionais (casos de uso)

Os requisitos funcionais referem-se a questões relacionadas com a maneira como o sistema deve operar, em que se especificam as entradas e as saídas do sistema e o relacionamento comportamental entre elas, assim como a interação com o usuário.

3.1.1 RF01 - Escolha do modo do jogo

- Descrição: ao ligar o jogo, deve aparecer, na tela de LCD, a pergunta de qual modo do jogo deve ser iniciado. São três os modos de jogo.

- Entrada e saída:

(1) - Modo de jogo normal: neste modo cada jogador deve, na sua vez, digitar no teclado a operação escolhida.

(2) - Modo de jogo com o sistema: neste modo, o jogador humano joga com o jogador autônomo.

(3) - Modo de jogo de um estado específico: neste modo o jogo é iniciado de um estado pré definido.

O jogo deve iniciar a partir da escolha do jogador.

3.1.2 RF02 - Geração de números aleatórios

- Descrição: o sistema deve gerar três números aleatórios entre 1 a 6, possível com repetição.

- Entrada e saída: o jogador deve, na sua vez, apertar o botão *Dado*, que ira gerar três números aleatórios e mostrá-los no LCD.

3.1.3 RF03 - Indicação no tabuleiro

- Descrição: com os três números aleatórios em mãos, o jogador deve formar uma operação matemática, de acordo com sua estratégia, para tentar vencer o jogo.

- Entrada e Saída:

Cada jogador, na sua vez, deve digitar no teclado a operação escolhida. O resultado deve ser igual a um dos números inscritos no tabuleiro. O LED da cor correspondente ao jogador deve acender.

Caso o resultado da operação digitada não corresponda a nenhum número do tabuleiro, o LCD deve informar que o resultado está errado e passar a vez para o próximo jogador.

Se o resultado da operação for um número já escolhido (casa com LED acesso), o LCD deve informar o erro e passar a vez para o próximo jogador. Operações ilegais devem ser informadas, dando a vez para o próximo jogador.

3.1.4 RF04 - Jogar com IA

- Descrição: se o jogador escolhe o modo 2, descrito em RF01, o jogador humano deve jogar com um jogador autônomo.

-Entrada e saída: o jogador humano, na sua vez, e com os números gerados aleatoriamente, deve escolher a operação e digitar no teclado. Na vez do jogador autônomo, o jogador humano deve teclar o botão *Dado*. Irão aparecer os três números aleatórios no LCD e a operação escolhida pelo jogador autônomo. O resultado deve acender o LED no tabuleiro, na casa correspondente. Caso o jogador autônomo não encontre um resultado adequado, ele passa a vez para o jogador humano.

3.1.5 RF05 - Registro do jogo

- Descrição: no sistema, um dispositivo *SdCard* ou uma porta USB deve ser disponibilizada para ligar a um pendrive ou em algum microcomputador.

- Entrada e saída: em qualquer das saídas, o dispositivo ligado ao equipamento, o sistema deve criar um arquivo de texto e gravar neste os registros de cada lance do jogo.

3.1.6 RF06 - Melhor jogada

- Descrição: durante o jogo, o orientador ou o professor podem precisar fazer algum tipo de intervenção. Para o auxiliar, o sistema deve ter um meio de mostrar qual a melhor jogada, naquele momento do jogo.

- Entrada e saída: em qualquer momento do jogo, ao acionar um botão específico, o sistema deve mostrar no LCD a melhor operação matemática para aquele instante do jogo.

3.1.7 RF07 - Jogar a partir de um momento específico

- Descrição: o professor, ou orientador, pode escolher uma configuração específica para iniciar uma partida.

- Entrada e saída: quando o modo 3, descrito em RF01, é escolhido, antes de iniciar o jogo devem-se digitar no teclado os valores das casas do tabuleiro, que devem acender. Depois disso, o botão *start* é teclado e o jogo é iniciado a partir disso.

3.2 Requisitos não funcionais

Os requisitos não funcionais são aqueles que não estão especificamente relacionados com a funcionalidade do sistema. Eles impõem restrições no produto a ser desenvolvido e/ou no processo de desenvolvimento do sistema, como também especificam restrições externas, às quais o produto precisa atender.

3.2.1 RNF01 - Desempenho

- Descrição: as jogadas feitas pelo agente autônomo não devem ultrapassar a três segundos. Todas as equações digitadas pelos jogadores devem apresentar a resposta no LCD e acender o LED da casa correspondente do tabuleiro. Esse processo deve acontecer em menos de 3 segundos.

Obs. A ideia do jogador autônomo não é ser um jogador invencível.

3.2.2 RNF02 - Atributos físicos

- Descrição: o peso do produto final não deve ultrapassar a 2 Kg.

As dimensões do equipamento devem ser adequadas para prover interação entre os jogadores. O tamanho ideal é pouco maior que uma folha

de papel do tamanho A4 (210 x 297 mm).

3.2.3 RNF03 - Custos do sistema

O protótipo não deve ultrapassar o custo de R$ 100,00. Essa limitação financeira se dá pelo fato de o produto, se eventualmente for comercializado, chegar ao consumidor a custo acessível.

3.2.4 RNF04 - Escalabilidade

Em todo o projeto deve ser considerado o acréscimo de possíveis melhorias em seus recursos, como a possibilidade da troca do painel de LEDs, viabilizando 3 ou 4 jogadores, ou, mesmo, para um painel que reproduza em braille, a implementação de novos jogos, o acréscimo de dispositivos de rede, e outros.

3.3 Interface com o usuário

Um requisito que deve ser tratado de forma especial é a interface com o usuário. Consulta a profissionais que trabalham com usabilidade em jogos pode ser necessária.

3.3.1 Tabuleiro de LEDs

O tabuleiro é uma matriz 8x8, formando 64 casas numeradas. As casas devem ser grandes o suficiente para acomodar 2 LEDs, um de cada cor. Na Figura 1 mostra-se um exemplo de como deve ser o formato do teclado.

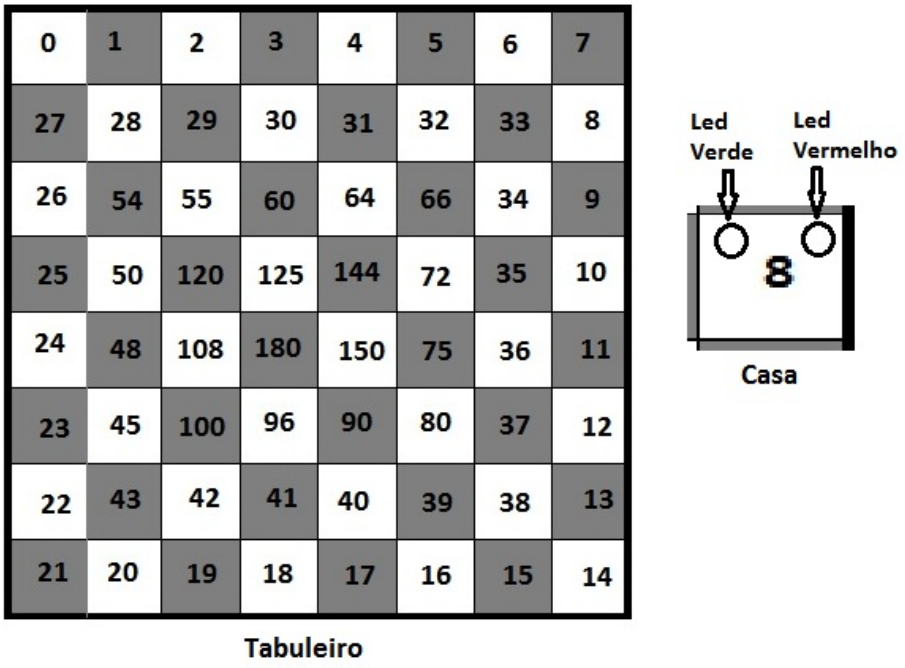


Figura 1 Exemplo do tabuleiro em formato de xadrez.

3.3.2 Teclado

Junto ao tabuleiro numerado devem existir dois conjuntos de teclas, uma para cada jogador, provendo interação do jogador com o jogo, informando, assim, as operações matemáticas. Um *display* de LCD é necessário para que, quando o jogador usar o teclado, ele possa confirmar o número digitado, além de informar ao jogador o resultado da operação, qual jogador é a vez de jogar, qual o modo do jogo e outras informações. O LCD com o teclado é mostrado na Figura 2.

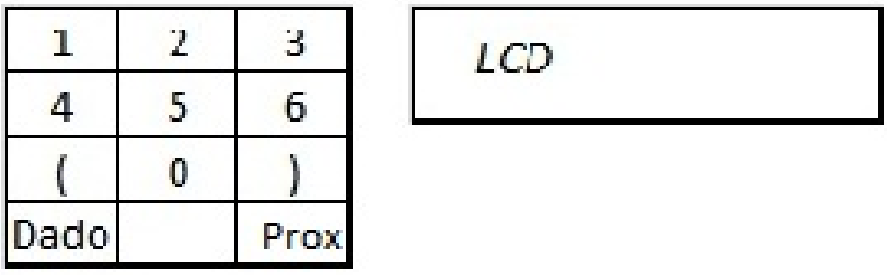


Figura 2 Teclado e LCD.

3.3.3 Disposição dos componentes do jogo

A posição em que os componentes do tabuleiro de LEDs, o teclado e o LCD devem ser dispostos, de forma a suportar 2 jogadores ou 2 times de 2 ou 3 jogadores, está sugerida nas Figuras 3 e 4.

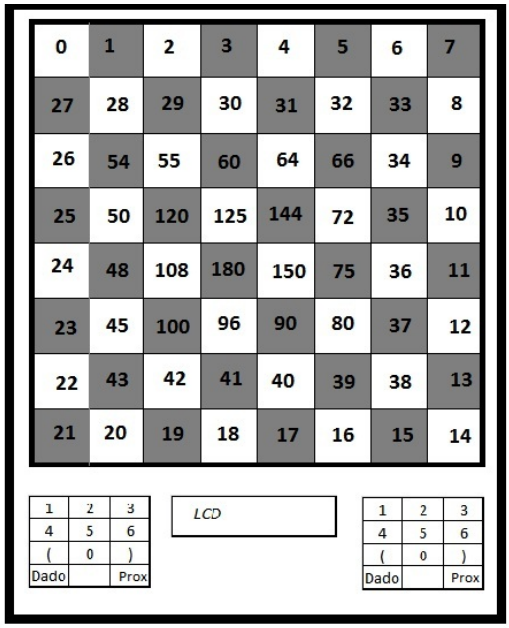


Figura 3 Exemplo de disposição do tabuleiro.

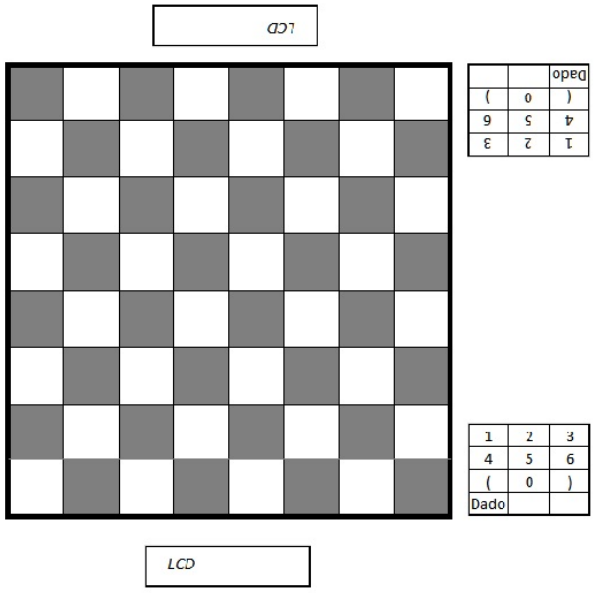


Figura 4 Exemplo de disposição do tabuleiro.