Benedito Augusto de Almeida Soares

APLICATIVO DE CONTROLE DE UM TABULEIRO ELETRÔNICO DO JOGO WAR BASEADO EM SISTEMAS MULTIESTÁGIO

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia Elétrica ênfase em
Eletrônica

Orientador: Prof. Dr. Evandro Luís Linhari Rodrigues

São Carlos
2009

Dedicatória

Aos meus pais que trabalharam muito para que eu pudesse estudar e para que eu tivesse a chance de perseguir meus sonhos.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho é fruto da vontade de seus autores e orientador de se realizar um trabalho que trouxesse aplicação prática aos conceitos passados ao longo da graduação e que fosse também de fácil interação com um usuário leigo. A paixão por jogos e o gosto por sistemas multiestágio fizeram nascer idéia a que possibilitou casar essas vontades. Aqui é descrito todo o projeto de criação de uma versão eletrônica do jogo de estratégia *War*, que se utiliza de vários sistemas (tais como microcontroladores, memórias, comunicação infravermelha, displays, rotinas de programação, etc.) para a implementação eletrônica de um jogo, onde se possam aplicar os conhecimentos de engenharia elétrica adquiridos ao longo da graduação, em especial, conhecimentos de sistemas multiestágio, isto é, sistemas constituídos de diversos módulos com funções especificas como por exemplo interface com o usuário, comunicação visual, comunicação entre os módulos utilizando protocolos, armazenamento de dados, etc. O projeto é baseado em um tabuleiro eletrônico semelhante ao jogo original e um controle sem fio que permite aos jogadores efetuarem as diferentes opções oferecidas durante uma partida do jogo. Esse trabalho é dividido em duas partes: uma para hardware e outra para software. Essa monografia tange tudo aquilo concernente ao software. O trabalho de conclusão de curso de Alício Aparecido Moraes de Souza versa sobre os aspectos de hardware. Esses dois trabalhos são, portanto, complementares e devem ser encarados como um trabalho único.

Palavras-chave: multiestágio, EEPROM, LCD, infravermelho, War, PIC.

Abstract

This work is consequence of its authors and advisor's will to implement a project that brings practical application to the concepts passed throughout the graduation and that is also of easy interaction with a lay user. The passion for games and the liking for multistage systems produced the idea which has made possible these wills to be joined. Here it is described the whole project of an electronic version to the strategy game called Risk (War in Brazil) which uses many systems, such as microcontrollers, memories, infrared communication, displays, programming routines, etc., to implement a game where it is possible to apply the electrical engineering knowledge acquired throughout the graduation, specially, multistage systems knowledge, that is, systems consisting of several modules with specific functions such as the user interface, visual communication, communication between the modules using protocols, data storage, etc. The project relays on an electronic board game similar to the original and a wireless remote control which allow players to execute different options available in a game. This work is divided into two parts: one for hardware and another for software. This monograph concerns to everything related with software. The course conclusion work of Alício Aparecido Moraes de Souza turns on the hardware aspects. These two works are, therefore, complementary and must be faced as an only work.

Keywords: multistage, EEPROM, LCD, infrared, Risk, PIC.

1. Introdução

O jogo de estratégia *War* é umas das versões internacionais do jogo de tabuleiro norte americano *Risk* que permite de dois a seis participantes. A versão utilizada durante o desenvolvimento do projeto é apenas uma das cinco versões brasileiras: War, War Edição Especial, War II, War Júnior e War Império Romano.

O tabuleiro do jogo tem formato de um mapa-múndi com 42 territórios agrupados em seis continentes e todos os jogadores controlam exércitos de cores distintas, com o intuito de conquistar os territórios de seus adversários, e visando a execução de um objetivo que lhes é apresentado no início do jogo e que é secreto para os demais jogadores (Grow, 2004). A disposição dos países no tabuleiro é definida visando facilitar a dinâmica do jogo e não representa completamente a realidade conforme a Figura 1:

Figura 1 - Jogo de War

Esse projeto tem o objetivo de aplicar os conhecimentos de engenharia elétrica na confecção de um tabuleiro eletrônico para se jogar War utilizando de sistemas multiestágio. As peças, cartas-objetivo, dados de ataque e defesa e toda a dinâmica do jogo passam a ser implementados num misto de hardware e software.

A cada país ou território é atribuído um circuito chamado de célula-país que se conecta a outros circuitos semelhantes formando o mapa-múndi do projeto. O diagrama abaixo foi montado para ilustrar essa união:

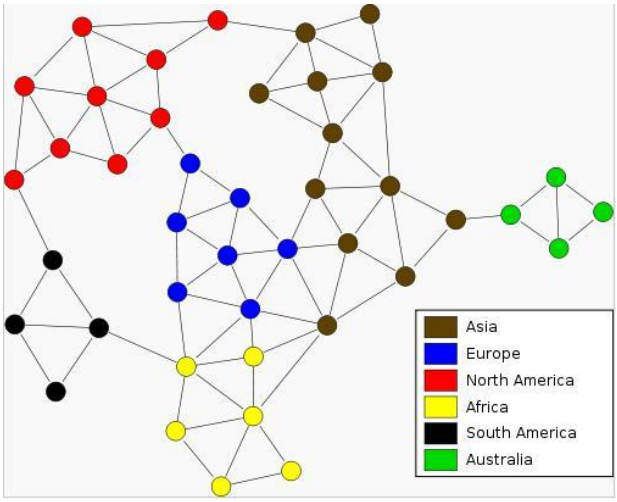


Figura 2 - Diagrama Mapa-múndi

Um microcontrolador rege toda a interação entre as células-país, os controles e os dados, além de exibir as jogadas num display informando aos jogadores os passos a serem tomados. Para representar as cores e número de exércitos presentes em cada um dos territórios foi proposto um sistema constituído de um display de sete segmentos de dois dígitos e um conjunto de LED de cores diferentes. Um tabuleiro eletrônico é capaz de adicionar ao jogo original as seguintes vantagens:

- Fornecimento de estatísticas;
- Capacidade de armazenamento do jogo em progresso;
- Controle de violação das regras por parte dos jogadores;
- Interface de controle mais eficiente para algumas funções (Ex: posicionamento de exércitos e lançamento de dados).

Na confecção desse projeto, foi proposta a divisão do mesmo em duas partes, para caráter de apresentação: uma parte que aborda os aspectos de software e outra que aborda os aspectos de hardware. Essa monografia versa sobre o segundo aspecto: software.

2. Regras do jogo

Nas seções abaixo, a transcrição das regras do jogo para a familiarização do leitor com o mesmo e para posterior entendimento do fluxograma que adapta o jogo original ao implementado nesse projeto.

2.1 Componentes do jogo

- 1 tabuleiro.
- 6 conjuntos de fichas de valores diferentes.
- 14 cartas-obejtivo.
- 44 cartas de território (incluindo dois coringas).
- 3 dados vermelhos.
- 3 dados amarelos.

2.2 Preparação

Cada jogador escolhe o exército da cor que preferir, dentre as seis possíveis (branca, vermelha, preta, azul, amarela e verde). Esta escolha pode ser feita por sorteio ou de comum acordo entre os participantes.

2.3 Valor das fichas

As fichas representam os exércitos de cada jogador. Uma ficha pequena é igual a um exército e uma ficha grande, a dez.

2.4 Cartas-objetivo

Cada jogador recebe uma carta por sorteio. Ao tomar conhecimento de seu objetivo o jogador não o revela a seus adversários (os objetivos restantes não serão utilizados no jogo).

É importante que, antes do sorteio, os jogadores que estão se iniciando no jogo façam uma leitura de todos os objetivos possíveis.

Obs.: no caso do número de jogadores ser inferior a seis, os objetivos relacionados com os exércitos não participantes devem ser excluídos do sorteio. Exemplo: se ninguém jogar com

os exércitos amarelos, a carta-objetivo que manda destruir os exércitos amarelos deverá ser retirada.

2.5 Distribuição de exércitos e territórios

Cada jogador pega um dado e o lança. Aquele que obtiver mais pontos será o distribuidor. O distribuidor pega o conjunto de cartas de território, retira o coringa e as distribui no sentido horário, começando pelo jogador a sua esquerda. Cada jogador deve colocar um exército da sua cor em cada um dos territórios recebidos durante o sorteio.

Ao final dessa operação, todos os territórios estarão ocupados por um exército. Em seguida, o distribuidor recolhe as cartas de território, recoloca os coringas e embaralha as cartas. Finalmente, coloca esse monte de cartas, com a face para baixo, próximo ao tabuleiro, onde deverá ficar durante o jogo.

2.6 O Jogo

O jogo tem início com o jogador seguinte ao último que recebeu a carta de território. Na **primeira rodada**, cada um dos jogadores, na sua vez, deve receber exércitos e colocá-los no mapa segundo sua estratégia.

A partir da **segunda rodada**, cada jogador, na sua vez, cumpre as seguintes etapas (sempre nessa ordem):

1) **recebe novos exércitos**, em função dos territórios conquistados e, se puder, em função da troca de cartas, e os coloca no tabuleiro segundo sua estratégia;
2) **ataca os adversários**, se desejar;
3) **desloca seus exércitos**, se houver conveniência;
4) **recebe uma carta território**, se conquistar, no mínimo, um território.

2.7 Recebimento de exércitos

No início da jogada, existem três maneiras de receber exércitos:

1) **a partir do número de territórios possuídos**;
2) **se possuir um continente inteiro**;
3) **se puder trocar as cartas (explicações na seção 2.12 – Troca de Cartas)**.

2.7.1 Territórios possuídos

No inicio da sua vez, o jogador soma o número de territórios que possui e divide por 2.

O resultado dessa conta será o número de exércitos que ele deverá receber (só considerando a parte inteira do resultado).

Exemplo:

- 8 territórios, recebe 4 exércitos.
- 11 territórios, recebe 5 exércitos.

O jogador deverá colocar todos os exércitos recebidos em um ou mais de seus territórios, conforme sua estratégia.

2.7.2 Continentes possuídos

Se, no início de sua vez de jogar, o jogador possuir um continente inteiro, além dos exércitos recebidos pela contagem dos territórios, receberá outros exércitos, de acordo com os valores da tabela abaixo.

Continente	Exércitos extras
Ásia	7
Europa	5
América do Norte	5
África	3
América do Sul	2
Oceania	3

Tabela 1 - Exércitos extras por continente

Nota: os exércitos recebidos pela posse de um continente deverão ser distribuídos obrigatoriamente nos territórios do próprio continente.

Exemplo: O jogador possui 19 territórios, sendo 15 espalhados por vários continentes e 4 da América do Sul.

- Número de exércitos a receber pelo número de territórios: 9 (19÷2=9,5).
- Número de exércitos a receber por possuir a América do Sul: 2.

- Total de exércitos a receber: 11 (9+2), sendo que dois da América do Sul devem ser postos obrigatoriamente nos territórios desse continente.

Obs.: O número mínimo de exércitos a receber é sempre 3, mesmo que o jogador possua menos de 6 territórios.

2.8 Ataque

A partir da segunda rodada, os jogadores podem ou não atacar algum adversário, tentando conquistar mais territórios. Cada território é ocupado por um exército que é chamado de exército de ocupação. Esse exército de ocupação não pode ser utilizado para atacar. Por isso, são necessários, no mínimo, dois exércitos no território de onde vai partir o ataque.

Como atacar:

1) O ataque usa os dados vermelhos e o de defesa usa os amarelos.
2) O ataque, a partir de um território qualquer, só pode ser dirigido a um território adversário contiguo (que tenha fronteiras em comum ou que esteja ligado através de uma linha pontilhada).
3) O atacante deve anunciar de que território vai partir o ataque e qual será atacado, bem como quantos exércitos estará usando para atacar.
4) O número máximo de exércitos participantes em cada ataque é três, mesmo que haja mais exércitos no território atacante.
5) Um território pode ser atacado independentemente do número de exércitos.
6) O território atacado pode usar, inclusive, o exército de ocupação para se defender.
7) Na sua vez de jogar, cada jogador pode atacar quantas vezes quiser para conquistar um território adversário.
8) Os ataques podem partir de um ou mais territórios, mas sempre um de cada vez, de acordo com a estratégia do atacante. O jogador também pode atacar vários territórios na sua vez de jogar. A cada ataque deve haver uma confrontação de dados (explicação na seção Batalhas)
9) O jogador atacante jogará com o número de dados correspondente ao número de seus exércitos participantes da batalha, ocorrendo o mesmo com o jogador da

defesa. Assim, se o atacante usar 3 exércitos contra 1 da defesa, ele jogará três dados contra um do defensor.

2.9 Batalha

Nas seções abaixo, exemplos da forma que as batalhas se dão.

2.9.1 Atacante com quatro exércitos e defensor com três exércitos.

Tanto o atacante (Congo) quanto o defensor (África do Sul) podem lutar com três exércitos. Assim, ambos podem jogar com três dados. Após a batalha (realizada através do lançamento dos dados), é feita a comparação dos pontos do atacante com os pontos do defensor, para verificar quem perde exércitos. Compara-se o dado com mais pontos do atacante com o de mais pontos do defensor. A vitória será de quem tiver com mais pontos. No caso de empate, a vitória será da defesa.

Em seguida, compara-se o segundo dado com mais pontos do atacante com o segundo dado com mais pontos do defensor: a vitória será decidida como no caso anterior.

Por fim, são comparados os dados com menos pontos, com base no mesmo procedimento anterior.

Supondo-se que o atacante tenha tirado 5, 4 e 1, e o defensor 6, 3 e 1, a comparação será feita da seguinte forma:

		Ataque	Defesa	Vencedor		
Comparação	Maior			Defesa	Ataque perde 1 exército	Resultados
	Segundo Maior			Ataque	Defesa perde 1 exército	
	Menor			Defesa	Ataque perde 1 exército	

Tabela 2 - Representação de batalha

Neste caso, o atacante perde dois exércitos e o defensor perde um exército. Assim, tanto os territórios do atacante, que tinha quatro exércitos, como o do defensor, que tinha três exércitos, ficam com dois exércitos cada um.

Observe que, apesar do atacante ter ficado com dois exércitos em seu território, só poderia atacar novamente com um, já que o outro é o exercito de ocupação, que não pode ser usado para ataques.

Se houvesse interesse o atacante poderia fazer um novo ataque usando um exército contra dois da defesa.

Nota: os exércitos perdidos na batalha voltam para as caixinhas podendo ser reutilizados posteriormente.

2.9.2 Atacante com três exércitos e defensor com um.

Neste caso, o ataque pode jogar com dois dados contra um da defesa. Supondo-se que os pontos sejam: ataque (3 e 2) e defesa (6), serão comparados os dados com mais pontos do ataque (3) com o de mais ponto da defesa (6), rejeitando-se o dado com menos ponto do atacante. A vitória neste caso cabe à defesa e o atacante retira um de seus exércitos.

Obs.: Cada vez que um atacante perde, ele retira de seu território apenas o número de exércitos com que a defesa se defendeu. No exemplo acima, a defesa usou apenas um dado. Portanto, o atacante perde apenas um exército.

2.10 Conquista de território

Se, após uma batalha, o atacante destruir todos os exércitos do território do defensor, terá então conquistado o território e deverá deslocar seus exércitos atacantes para o território conquistado. Esse deslocamento obedece à seguinte regra:

O número de exércitos a ser deslocado é igual, no máximo, ao número de exércitos que participaram do último ataque.

2.11 Deslocamentos

Há dois momentos em que o jogador pode deslocar seus exércitos:

O primeiro, refere-se ao **deslocamento dos exércitos atacantes para o território conquistado**; o segundo, refere-se aos **deslocamentos permitidos quando o jogador já finalizou seus ataques** na sua vez de jogar.

Ao finalizar seus ataques, o jogador poderá, de acordo com sua estratégia, efetuar deslocamentos de exércitos entre seus territórios contíguos.

Estes deslocamentos deverão obedecer às seguintes regras:

1) Em cada território deve permanecer sempre pelo menos um exército (o de ocupação), que nunca pode ser deslocado.
2) Um exército pode ser deslocado uma única vez, ou seja, não é permitido deslocar um exército para um segundo território contíguo, numa mesma jogada.

Exemplo: um jogador que possua o Brasil, a Venezuela e o México pode deslocar seus exércitos do Brasil para a Venezuela, mas não pode deslocar, na mesma jogada, esses mesmos exércitos da Venezuela para o México.

2.12 Conquista de cartas

O jogador que conquistar um ou mais territórios durante a sua vez tem o direito a uma carta território no final da jogada (após ter realizado os deslocamentos). É interessante notar que o jogador recebe uma única carta território por jogada, independentemente do número de territórios conquistados. O conteúdo dessa carta deve ser mantido em segredo até o momento de sua troca.

2.13 Troca de cartas

O jogador que possuir três cartas com figuras diferentes, ou então três cartas com figuras iguais, poderá, na sua vez de jogar, trocá-las por exércitos extras de acordo com a Tabela 3:

Número de Exércitos	
Primeira Troca	4
Segunda Troca	6
Terceira Troca	8
Quarta Troca	10
Quinta Troca	12
Sexta Troca	15
Demais Trocas	20, 25, 30,...

Figura 3 - Valores de exércitos por troca

Exemplo: o primeiro jogador que efetuar uma troca durante a partida recebe quatro exércitos; o segundo jogador que efetuar uma troca recebe seis; o terceiro oito e assim sucessivamente. Essa seqüência não obedece às trocas dos jogadores individualmente, mas a todas as trocas ocorridas no jogo.

O jogador não é obrigado a efetuar a troca enquanto tiver até quatro cartas. Entretanto, quando o jogador acumula cinco cartas, é obrigado, na sua vez de jogar, a trocar três cartas por exércitos. Os exércitos recebidos podem ser distribuídos pelos territórios do jogador da maneira que melhor lhe convier.

Se, dentre as cartas usadas para efetuar a troca, existir alguma que represente um território de sua propriedade, ele ganha mais dois exércitos extras para cada carta que represente um território deste jogador, e que devem ser colocados, obrigatoriamente, naquele território.

2.14 Eliminação de um jogador

Se, durante o jogo, um jogador destrói por completo um exército, não sendo este o seu objetivo (caso em que ganharia o jogo), ele recebe as cartas do jogador eliminado. Se, ao somar as cartas recebidas do jogador eliminado às suas, o jogador ficar com mais de cinco cartas, realiza-se um sorteio: ele retira, sem olhar, o número de cartas necessárias para completar cinco.

2.15 Final do jogo

O jogo termina quando um jogador consegue terminar seu objetivo. Neste momento, ele deve mostrar sua carta objetivo, comprovando sua vitória.

3. Adaptando o jogo

As regras do jogo de War apresentadas anteriormente tiveram de ser adaptadas para a realização desse projeto. As razões disso foram: 1º) a necessidade de se ater ao objetivo básico desse trabalho, que foi a aplicação prática dos conhecimentos de engenharia elétrica adquiridos na graduação e não a representação fiel do jogo original; e 2º) a necessidade de se adaptar o projeto aos recursos financeiros disponíveis. Posto isso, é apresentada nas seções abaixo a regra adaptada para esse projeto.

3.1 Componentes do jogo

Compõe esse jogo um tabuleiro eletrônico formado por células de circuito que representam os países e os dados, por uma célula central que gerencia as demais e por duas outras que formam os controles remotos, sendo um para ataque e outro para defesa.

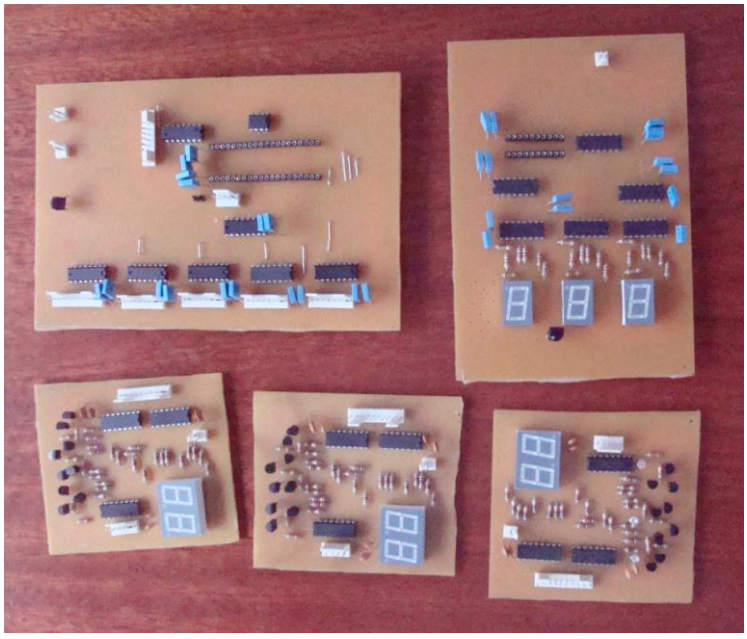


Figura 4 - Placas do protótipo de testes do tabuleiro eletrônico.

3.2 Cartas-objetivo

As cartas-objetivos foram eliminadas. Os jogadores passam a ter um objetivo único e comum a todos: conquistar o mundo, ou seja, todos os territórios.

3.3 Distribuição de territórios e exércitos

Assim que se inicia o dispositivo, uma mensagem aparece na tela de LCD perguntando aos jogadores se eles desejam iniciar um novo jogo ou continuar um antigo. Sendo a escolha a primeira opção, cada jogador receberá aleatoriamente um território ao qual será atribuído um exército de ocupação e, também, a cor do jogador em questão. Se a escolha for pela segunda opção, o jogo é iniciado da forma como foi salvo anteriormente, ou seja, cada país receberá a cor e a quantidades de exércitos que possuíam da última vez.

3.4 O jogo

O jogo se inicia com a escolha do jogador por uma das opções: iniciar um novo jogo ou recuperar um jogo inacabado.

Optando pela primeira opção, a primeira rodada se realiza e cada jogador recebe territórios e dispõe seus exércitos segundo sua estratégia. A partir disso, são realizadas n rodadas até que o objetivo seja alcançado por um dos jogadores.

A dinâmica dessas n rodadas é a seguinte:

Na sua vez, cada jogador cumpre as seguintes etapas:

1) **Recebe exércitos**.
2) **Ataca os adversários**, se desejar.
3) **Desloca seu exércitos**, se desejar.

A opção de receber uma carta de território, quando da conquista de um país, foi eliminada. O critério de recebimento de exércitos foi alterado para melhor se adaptar à ausência dessas cartas. Na próxima seção é explicada a nova forma de recebimento de exércitos.

Se a escolha for pela segunda opção, o jogo continua da rodada em que terminou e corre como explicado acima.

3.5 Recebimento de exércitos

No início de cada rodada, o jogador recebe exércitos de acordo com o número de territórios que possui. A parte inteira da divisão da quantidade de países que possui por dois é o número total de exércitos que ele receberá. Exemplo: se possui 10 territórios no começo da rodada, receberá cinco exércitos; se possui 11, receberá cinco também.

Não há recebimento de exércitos extras por razão de um jogador possuir um continente na totalidade. E, como já foi dito, também não há recebimento de exércitos extras decorrente de trocas de cartas, já que elas foram eliminadas.

O número mínimo de exércitos que um jogador pode receber é três. Assim, mesmo que possua menos que seis territórios, o jogador irá receber um mínimo de três exércitos no início da jogada.

3.6 Ataque, batalha, conquista de território e deslocamento de exércitos

Essas etapas do jogo são mantidas como no original. Seguem, portanto, as mesmas regras. Para mais informações, ver seções 2.7, 2.8, 2.9 e 2.10.

3.7 Eliminação de um jogador

Quando um jogador perde a posse de seu último território, ele é eliminado do jogo e, logo, passa a não mais participar das rodadas.

3.8 Final do jogo

Quando um jogador atinge seu objetivo, em outras palavras, quando um jogador adquire todos os 42 territórios, o jogo é finalizado e uma mensagem parabenizando o ganhador aparece na tela. A seguir, uma mensagem questionando se um novo jogo deve ser iniciado surge e, caso os jogadores decidam afirmativamente, uma nova partida é realizada; do contrário, o dispositivo é desligado.

4. Aplicativo do jogo

As disciplinas Introdução à Ciência da Computação I e II e Algoritmos e Estruturas de Dados I formaram a base do conhecimento de programação do autor. Para esse projeto, foi necessário conhecimento adicional em Engenharia de Software para garantir maior organização, produtividade e qualidade. A Engenharia de Software é uma área do conhecimento da computação voltada para a especificação, desenvolvimento e manutenção de sistemas de software aplicando tecnologias e práticas de gerência de projetos com o intuito de trazer organização, produtividade e qualidade (Pressman, 2005). Os fundamentos científicos para a engenharia de software envolvem o uso de modelos abstratos e precisos que permitem ao engenheiro especificar, projetar, implementar e manter sistemas de software, avaliando e garantindo suas qualidades (Magela, 2006).

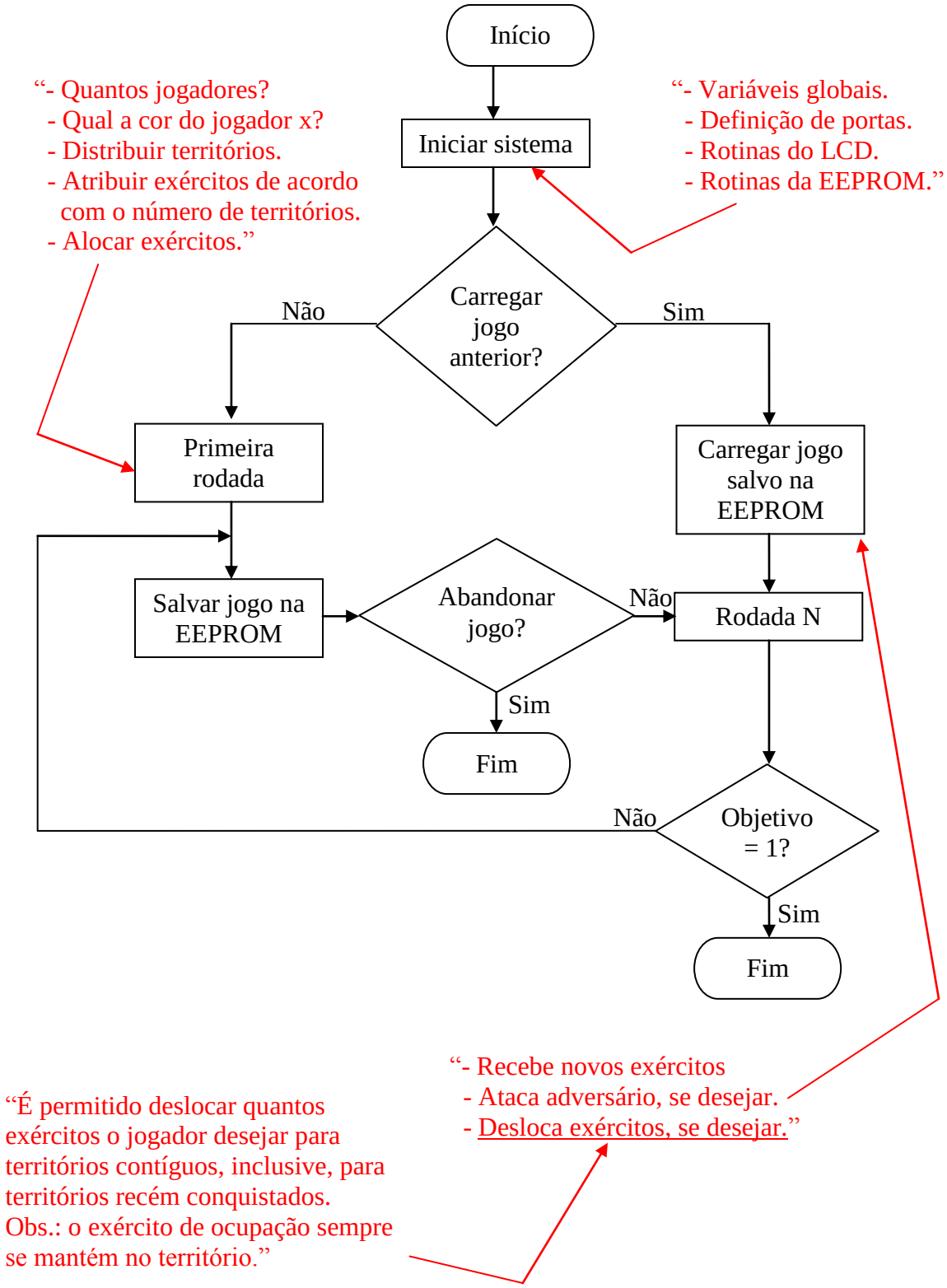
A ferramenta de engenharia de software escolhida para projetar o aplicativo foi o fluxograma. Ela é uma ferramenta para modelagem de sistemas que fornece uma visão estruturada das funções, ou seja, do fluxo dos dados (Molinari, 2009).

A partir do estabelecimento das regras, como visto nos capítulos anteriores, foi possível construir os fluxogramas que exemplificam toda a dinâmica do jogo.

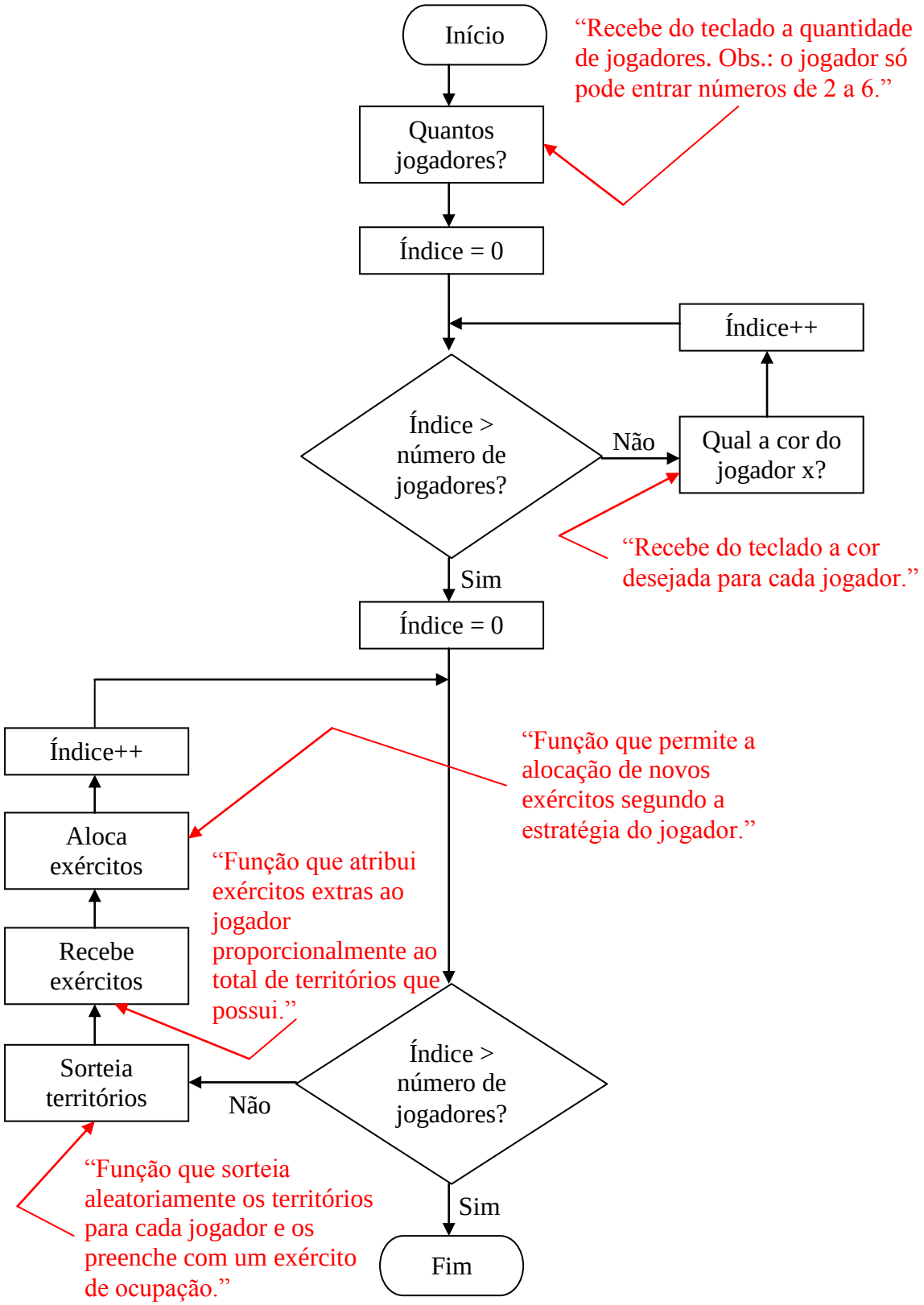
Por ter estrutura de linguagem computacional, o fluxograma se torna uma boa forma de se representar algoritmos: é uma grande ferramenta de programação.

Nas seções subseqüentes, apresentam-se os fluxogramas do jogo e de suas etapas.

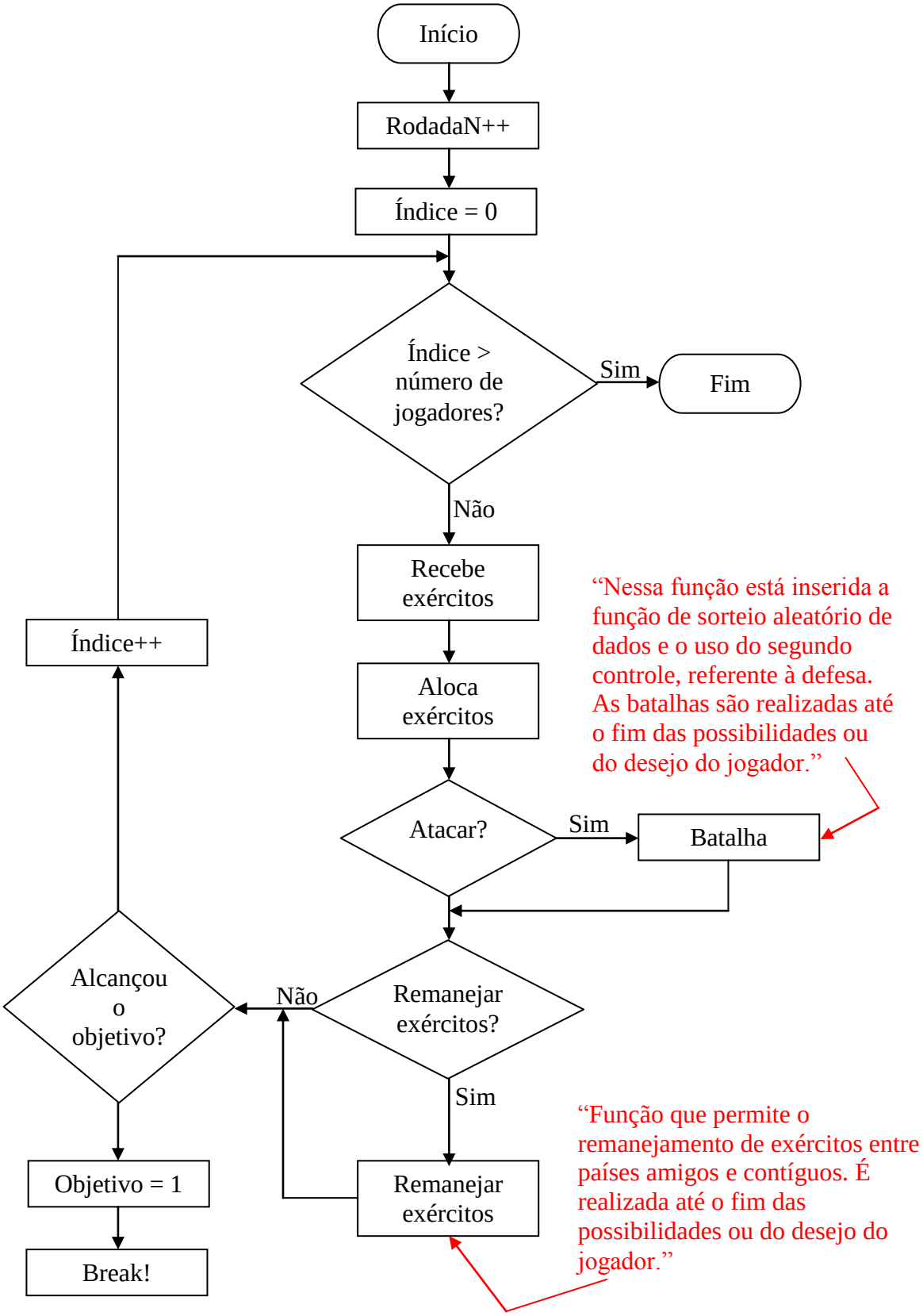
4.1 Fluxograma do jogo



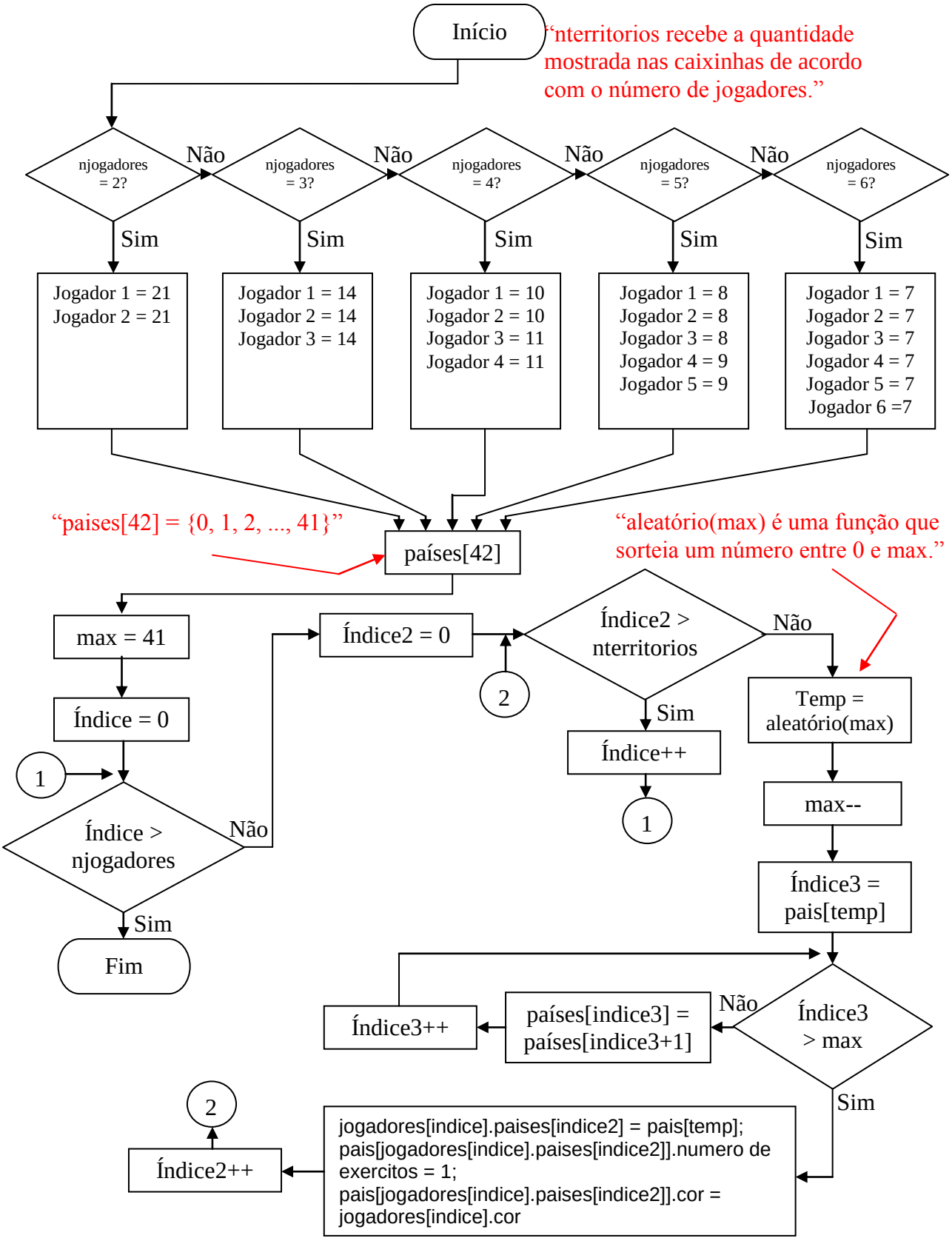
4.2 Primeira rodada



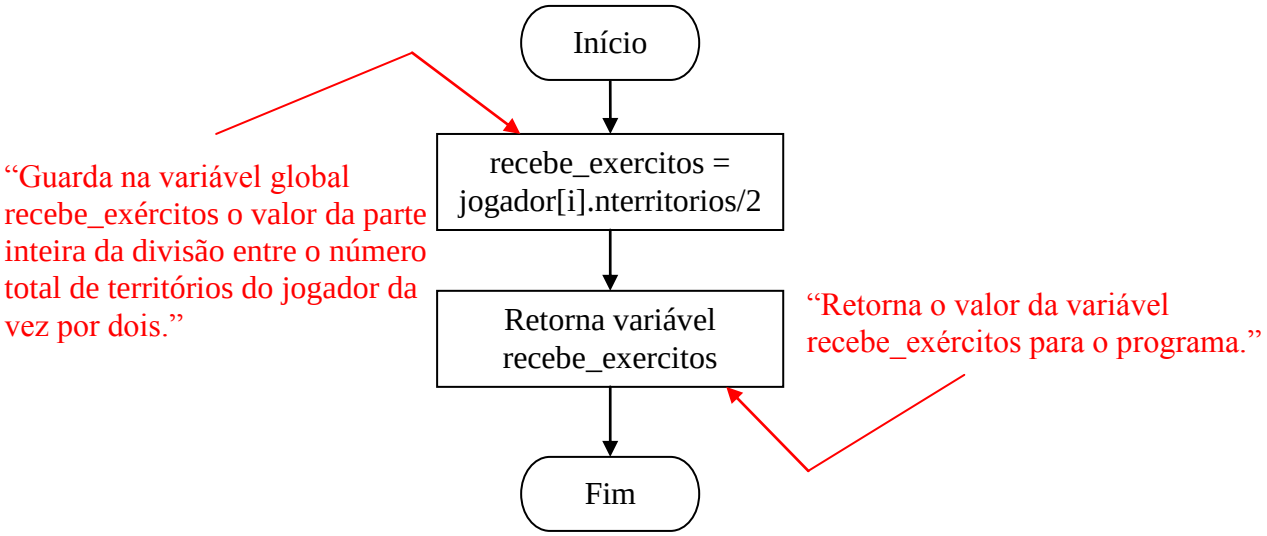
4.3 RodadaN


Início
RodadaN++
Índice = 0
Índice > número de jogadores?
Sim
Fim
Não
Recebe exércitos
Aloca exércitos
Atacar?
Sim
Batalha
Remanejar exércitos?
Não
Sim
Alcançou o objetivo?
Índice++
Objetivo = 1
Break!
Remanejar exércitos
"Nessa função está inserida a função de sorteio aleatório de dados e o uso do segundo controle, referente à defesa. As batalhas são realizadas até o fim das possibilidades ou do desejo do jogador."
"Função que permite o remanejamento de exércitos entre países amigos e contíguos. É realizada até o fim das possibilidades ou do desejo do jogador."

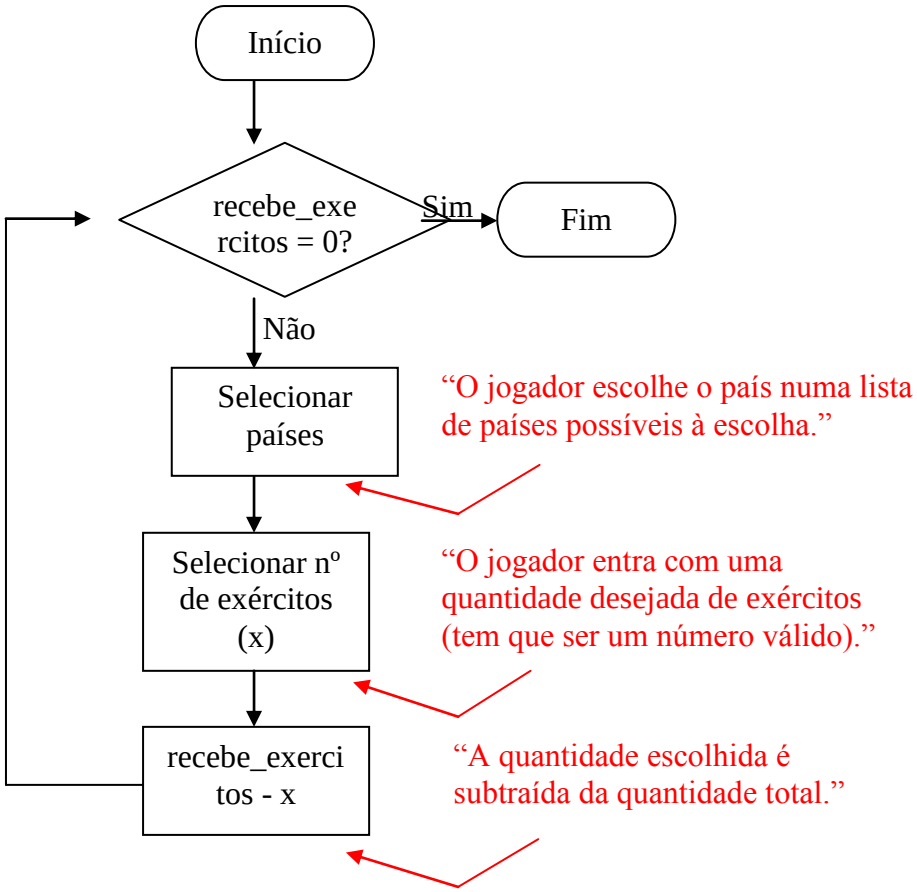
4.4 Sorteio de territórios



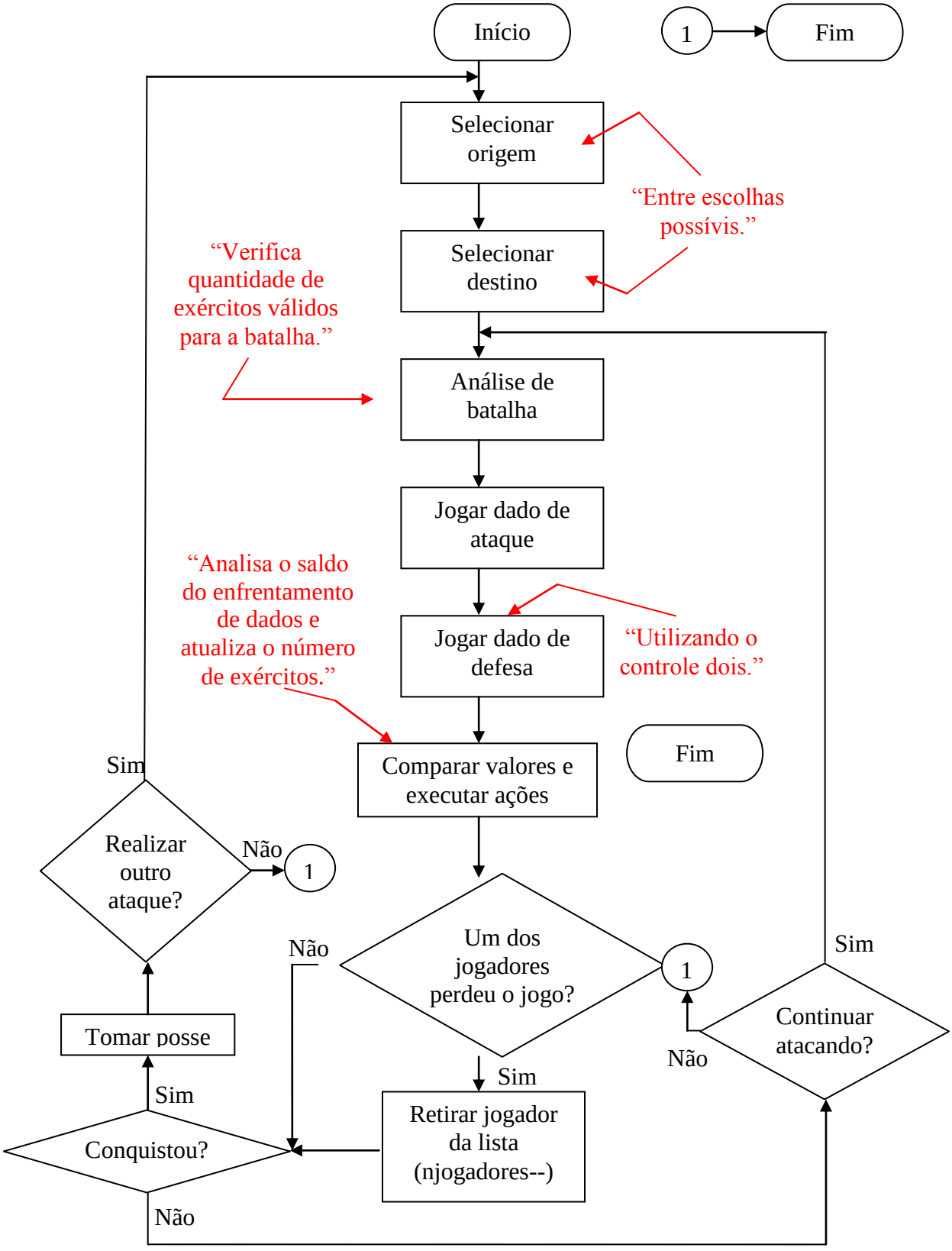
4.5 Recebimento de exércitos



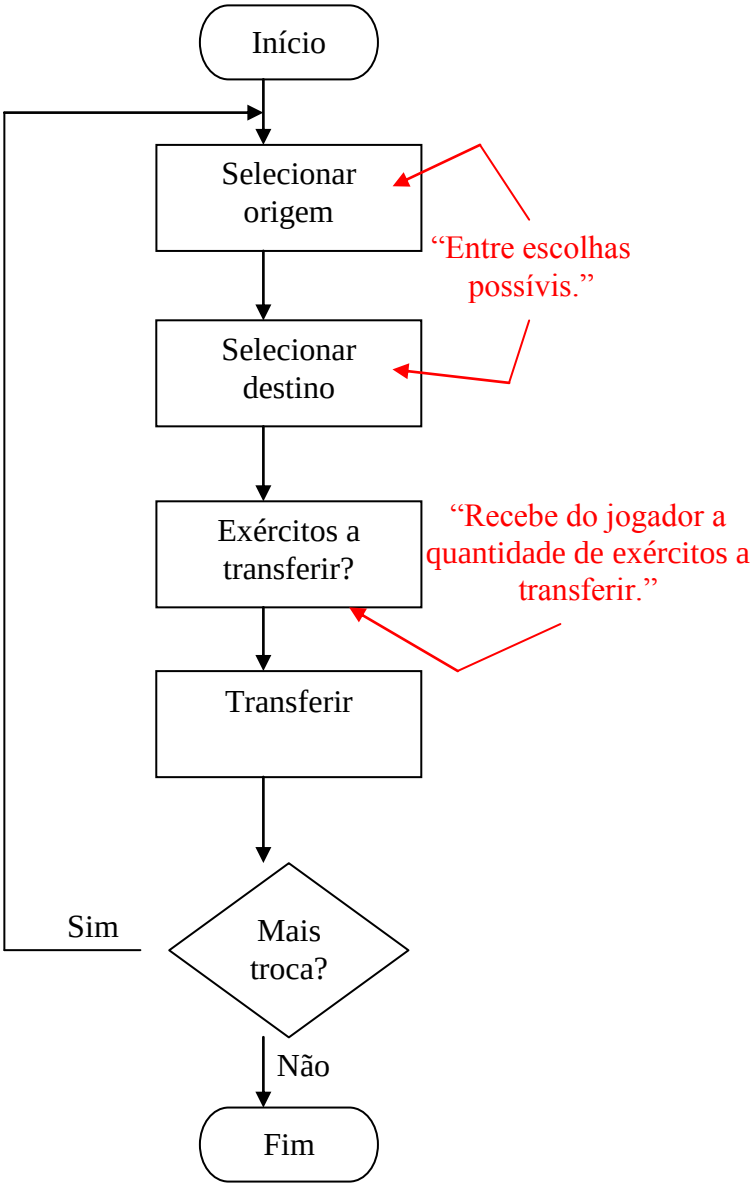
4.6 Alocação de exércitos



4.7 Batalha


Início
1
Fim
Selecionar origem
"Entre escolhas possívis."
Selecionar destino
"Verifica quantidade de exércitos válidos para a batalha."
Análise de batalha
Jogar dado de ataque
"Analisa o saldo do enfrentamento de dados e atualiza o número de exércitos."
Jogar dado de defesa
"Utilizando o controle dois."
Fim
Comparar valores e executar ações
Sim
Realizar outro ataque?
Não
1
Não
Um dos jogadores perdeu o jogo?
1
Sim
Continuar atacando?
Sim
Não
Tomar posse
Sim
Retirar jogador da lista (njogadores--)
Conquistou?
Não

4.8 Remanejamento de exércitos



5. Variáveis do jogo

Posto o que foi mostrado nos capítulos anteriores sobre o programa que rege o tabuleiro eletrônico de War, vale a pena ressaltar agora alguns aspectos mais técnicos da programação para trazer melhor entendimento das rotinas esquematizadas nos fluxogramas do Capítulo 4.

5.1 Definições

A definição de jogador e a definição de país foram tomadas imaginando-se cada um como caixinhas que contém outros elementos. Assim, um jogador é uma caixinha contendo uma variável “cor”, uma variável “quantidade de territórios possuídos” (nterritórios) e um vetor de variáveis denominado “territórios”, que lista os territórios possuídos por esse jogador. Do mesmo modo, um país é uma caixinha contendo uma variável “cor” e uma variável “quantidade de exércitos possuídos”.

Em linguagem C, essas caixinhas são, na verdade, registros e podem ser esquematizadas como abaixo:

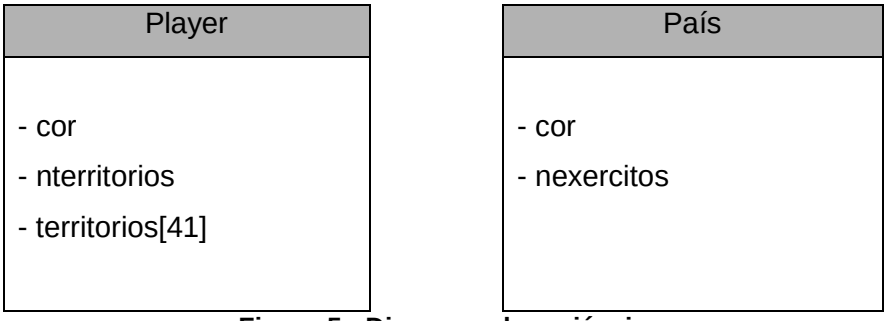


Figura 5 - Diagrama de variáveis

Outras variáveis que merecem mais atenção são as vaiáveis globais “rodadan”, que funciona como um contador de rodadas, “recebe_exercitos”, variável usada para guardar a quantidade de exércitos extras de um jogador quando na sua vez, e “objetivo_ok”, que é carregado com valor “1” toda vez que um jogador obtiver nterritórios = 42, ou seja, toda vez que um jogador obtiver a totalidade dos territórios.

Para a determinação de vizinhança entre países, modificou-se a Figura 1 da seguinte forma:

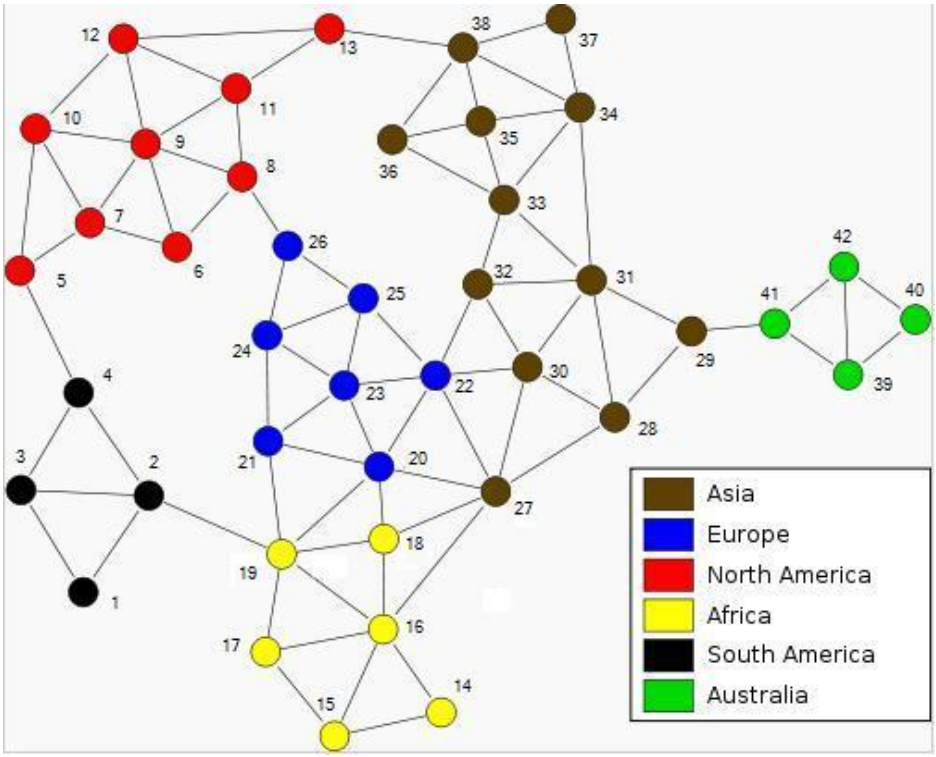


Figura 6 - Fronteiras

A partir dessa figura foi possível criar a seguinte tabela de correspondência entre países:

		n	Fronteiras					
Países	**1**	2	2	3	-	-	-	-
	2	2	4	19	-	-	-	-
	3	2	2	4	-	-	-	-
	4	2	2	5	-	-	-	-
	5	3	4	7	10	-	-	-
	6	3	7	9	8	-	-	-
	7	4	5	6	9	10	-	-
	8	4	6	9	11	26	-	-
	9	6	6	7	8	10	11	12
	10	3	7	9	12	-	-	-
	11	4	8	9	12	13	-	-
	12	4	9	10	11	13	-	-
	13	3	11	12	38	-	-	-
Países	**14**	2	16	15	-	-	-	-
	15	2	16	17	-	-	-	-
	16	3	17	18	19	-	-	-

	17	2	16	19	-	-	-	-
	18	5	16	19	20	21	27	-
	19	5	16	17	18	20	21	-
	20	6	18	19	21	22	23	27
	21	4	19	20	23	24	-	-
	22	6	20	23	25	27	30	32
	23	5	20	21	22	24	25	-
	24	4	21	23	25	26	-	-
	25	4	22	23	24	26	-	-
	26	3	8	24	25	-	-	-
	27	6	16	18	20	22	28	30
	28	4	27	29	30	31	-	-
	29	3	28	31	41	-	-	-
	30	5	22	27	28	31	32	-
	31	5	29	28	31	32	33	-
	32	4	22	30	31	33	-	-
	33	5	31	32	34	35	36	-
	34	5	31	33	35	37	38	-
	35	4	33	34	36	38	-	-
	36	3	33	35	38	-	-	-
	37	2	38	34	-	-	-	-
	38	5	13	34	35	36	37	-
	39	3	40	41	42	-	-	-
	40	2	39	42	-	-	-	-
	41	3	29	39	42	-	-	-
	42	3	39	40	41	-	-	-

Figura 7 - Matriz Mapa

Dessa forma, definiu-se uma variável chamada “fronteiras” que é uma matriz estática e usada nas funções que precisam deslocar exércitos entre países adjacentes.

5.2 Estatíscas

As variáveis descritas nesse capítulo são utilizadas também para gerar estatísticas de jogo. No display de LCD, informações do tipo: vez do jogador, valor da rodada, porcentagem de conclusão do objetivo, ranking de pontuação, entre outras, são disponibilizadas aos jogadores.

5.3 Variável aleatória

Em programação é muito difícil se gerar números verdadeiramente aleatórios. As funções ditas aleatórias são, na verdade, pseudoaleatórias. Uma forma de se criar uma função verdadeiramente imprevisível é fazer uso de alguma variável casual como semente do algoritmo que gera os números aleatórios da função.

Sabendo disso, é fácil perceber que a variável “cor” é excelente para esse papel. O jogador, seguindo apenas a sua vontade, escolhe de maneira aleatória uma das seis opções de cor que posteriormente será usada como semente do algoritmo da função mencionada.

5.4 Mapa de memória

O número de bytes que cada jogador ocupa é 44. Cada país ocupa 2 Bytes. As variáveis *rodadan*, *recebe_exércitos* e *objetivo_ok* ocupam, respectivamente, 4 Bytes, 1 Byte e 1 Byte. Assim, os endereços para o armazenamento dessas variáveis foram definidos da seguinte forma:

Variável	End. Memória	Página da Memória
Jogador 1	00h – 2Bh	Página 1
Jogador 2	2Ch – 57h	Página 1
Jogador 3	58h – 83h	Página 1
Jogador 4	84h – AFh	Página 1
Jogador 5	B0h – DBh	Página 1
Jogador 6	DCh – FFh	Página 1
	00h – 09h	Página 2
Todos os países	10h – 92h	Página 2
Rodadan	93h – 96h	Página 2
Recebe_exércitos	97h	Página 2

Tabela 3 - Mapa de memória

6. Estudo teórico

Para o desenvolvimento desse projeto, foi necessário estudar e entender os conceitos referentes ao uso de displays, memórias e comunicação infravermelha. A seguir são apresentadas algumas noções importantes para o entendimento dessas tecnologias.

6.1 LCD

Não há forma melhor de interação usuário-dispositivo senão através de meios que permitam comunicação visual. O LCD é uma excelente interface em sistemas microprocessados e foi escolhido para esse projeto com o intuito de trazer informações importantes aos jogadores sobre as partidas e estatísticas de jogo.

Para colocá-lo em funcionamento, primeiro é preciso configurá-lo, ou seja, dizer ao display como se transferirá os dados para ele, se 8 ou 4 bits, quantas linhas se utilizará, se a mensagem será fixa ou se vai rolar, se a escrita será da esquerda para a direita ou da direita para a esquerda, enfim, todas essas configurações são necessárias antes de escrever qualquer mensagem (Hitachi, 1998).

A temporização é um elemento importante também. Uma temporização equivocada inviabilizará o funcionamento do LCD. A análise para se chegar ao melhor valor de temporização se dá através do datasheet do dispositivo.

Nesse projeto, utilizou-se o display 16x2. Seus pinos são designados da seguinte forma:

- Pinos de dados: D7...D0 (8 bits). São usados tanto para enviar as palavras de configuração quanto os dados (caracteres).
- Pinos de controle: EN, RS e R/W. O pino EN é usado para habilitar ou desabilitar o display. O pino RS é usado para informar ao LCD se o dado enviado é de configuração ou caractere. R/W é usado para determinar se o processo será de escrita ou leitura.

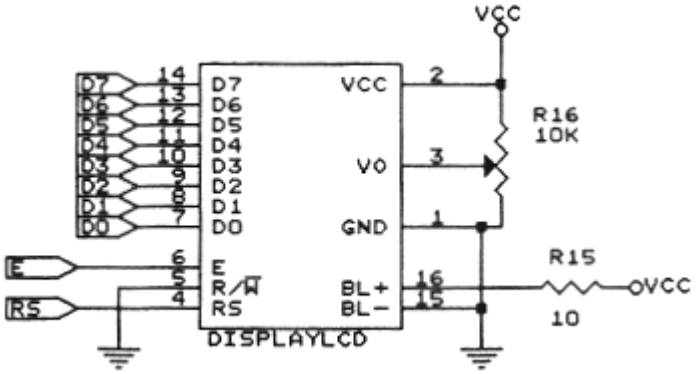


Figura 8 - Pinos do LCD

As tabelas abaixo ilustram como configurar o display de LCD.

Instrução									
Configurar Função									
RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	0	0	0	1	DL	N	F	0	0
Descrição									
Esta instrução é a mais importante, pois configura o display com relação ao número de bits, número de linhas e tamanho do caractere. DL: 0 => 4 bits ou DL: 1 => 8 bits. N: 0 => 1 linha ou N: 1 => 2 linhas, para display de 4 linhas, o N: 1. F: 0 => 5x7 dots ou F: 1 => 5x10 dots. Normalmente se usa: DL= 1, N=1 e F=0, ou seja, (00111000)b => 38h. OBS.: em alguns displays é necessário enviar essa informação três vezes para o display.									

Tabela 4 - Configurar Função

Instrução									
Controle do Display									
RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	**0**	**0**	**0**	**0**	**0**	**1**	**D**	**C**	**B**
Descrição									
Controle do display - **D:** 0 => off ou**D:** 1 => on; Controle do cursor - **C:** 0 => cursor desligado ou **C:** 1 => cursor ligado;									

Cursor piscante, mesmo que o cursor esteja desligado, toda a matriz ficará piscando - **B:** 0 => não piscante ou **B:** 1 => piscante. **Normalmente se usa: D= 1, C=1 e B=0, ou seja, (00001110)b => 0Eh.**

Tabela 5 - Controle do Display

Instrução									
Deslocamento do Cursor									
RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**I/D**	**S**
Descrição									
Este comando controla o deslocamento do cursor no display de LCD. O cursor pode deslocar da esquerda para direita - incremento(**I/D**:1) ou da direita para esquerda - decremento (**I/D**:0). Além disso, podemos configurar o display a função scroll, ou seja, a medida que se escreve a mensagem vai se deslocando, na realidade o cursor fica sempre parado, para isso, **S**:0 scroll desligado e **S**:1 scroll ligado. **Normalmente se usa: I/D= 1 e S=0, ou seja, (00000110)b => 06h.**									

Tabela 6 - Deslocamento do Cursor

Instrução									
Retorno do Cursor									
RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
Descrição									
Este comando faz o cursor retornar para a primeira posição do display de LCD.									

Tabela 7 - Retorno do Cursor

Instrução									
Limpar o Display									
RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
Descrição									
Este comando limpa tudo que foi escrito no display de LCD. **Código de configuração = 01h**									

Tabela 8 - Limpar o Display

As instruções acima são algumas instruções utilizadas no projeto. Outras instruções relevantes ao projeto podem ser obtidas no datasheet do dispositivo.

6.2 EEPROM

A memória EEPROM é uma sigla para Electrically Erasable Programmable Read Only Memory - Memória Somente de Leitura Programável Apagável Eletricamente (Atmel, 2001).

A escolha por se utilizar a EEPROM nesse projeto se justifica por ela atender à necessidade de se gravar constantemente numa memória o status atual do jogo. A cada fim de rodada, é salvo na EEPROM as variáveis que carregam informações sobre os jogadores e sobre as jogadas, o que permite pausar a partida e continuá-la posteriormente.

O funcionamento da EEPROM segue o protocolo I2C da Philips exemplificado abaixo pela Figura 6.2.

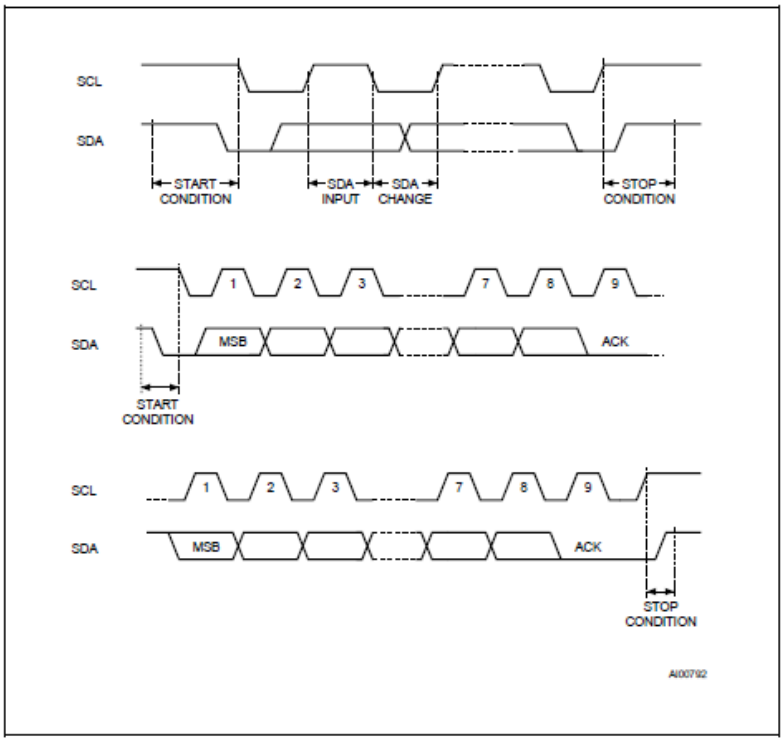


Figura 9 - I2C (Atmel, 2001)

Os processos de escrita e leitura na EEPROM podem ser resumidos pelas figuras abaixo:

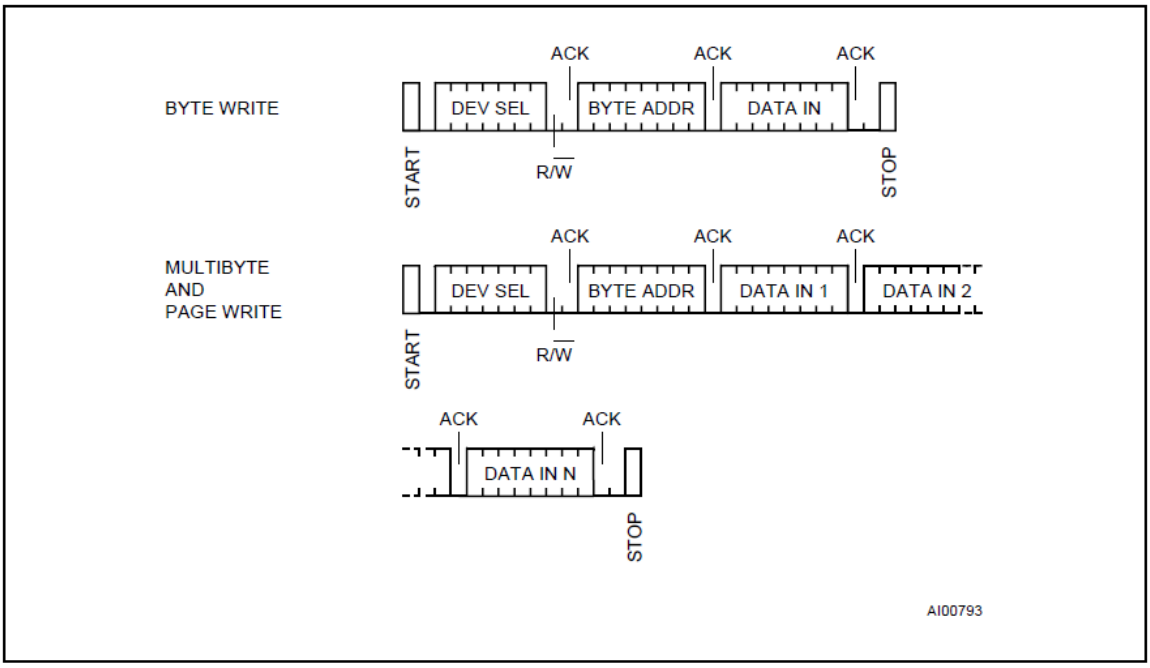


Figura 10 - Escrita na EEPROM (Atmel, 2001)

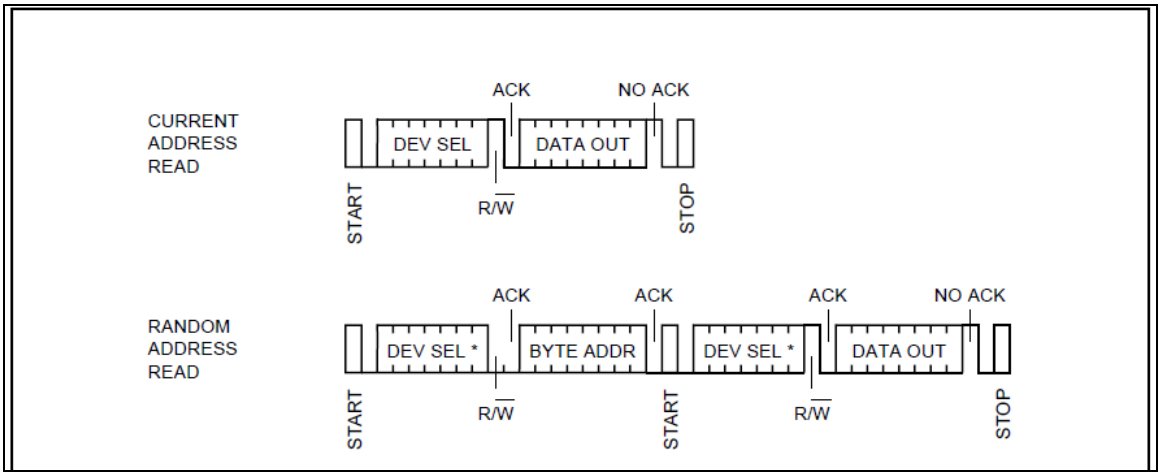


Figura 11 - Leitura na EEPROM (Atmel, 2001)

As rotinas para implementação do protocolo I2C e das funções de escrita e leitura na EEPROM foram criadas a partir do estudo das figuras acima. Para maior entendimento, consultar datasheet do componente 24C04.

6.3 Infravermelho

Para criar maior jogabilidade, foi decidido por se implementar um controle remoto. O controle remoto traz maior liberdade de movimento ao jogador e, portanto, permite maior conforto na operação do dispositivo.

A decisão por se fazer esse controle utilizando comunicação infravermelha se deve ao fato de essa tecnologia ser altamente difundida e muito barata.

6.3.1 RC-5

Modulação é a chave para fazer um sinal sobressair em relação ao ruído. Há muitas fontes de radiação infravermelha: o sol (maior de todas), o corpo humano, as lâmpadas incandescentes, etc, todas “disputam” a atenção de um receptor de IR (infrared). Desse modo, faz-se necessário utilizar-se da modulação para se distinguir sinal de ruído. É necessário criar-se um código de transmissão para se fazer o sinal “piscar” de um jeito especial chamando a atenção do receptor de IR e permitindo a ele atentar-se somente ao sinal e rejeitar todo o ruído.

O protocolo RC-5 desenvolvido pela Philips foi criado com esse intuito e é hoje, certamente, o código mais usado para aplicações de comunicação por infravermelho. É muito bem definido para diferentes tipos de dispositivos e oferece compatibilidade entre vários sistemas.

6.3.2 Características

- 5 bits de endereçamento e 6 bits de comando.
- Codificação Manchester.
- Freqüência da portadora de 36 kHz.
- Tempo de bit constante de 1,778 ms (64 ciclos de 36kHz).

Obs.: o código Manchester se baseia em fazer com que cada bit tenha uma transição e que ocupe um mesmo tempo. Abaixo, uma figura explicativa do código Manchester:

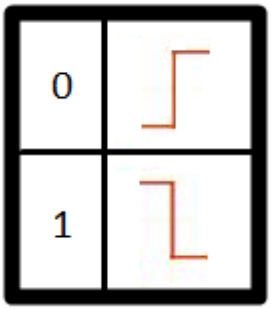


Figura 12 - Código Manchester

6.3.3 Modulação

O protocolo usa modulação bifásica, também chamada de codificação Manchester. Todos os bits são de igual tamanho (1,778 ms) com metade de seu tempo completado por uma rajada de pulsos da portadora e a outra metade deixando-se ociosa. A lógica “0” é representada por uma rajada de pulsos na primeira metade do tempo de bit seguida de atividade ociosa e a lógica “1” é representada por atividade ociosa na primeira metade do tempo de bit seguida de uma rajada da portadora.

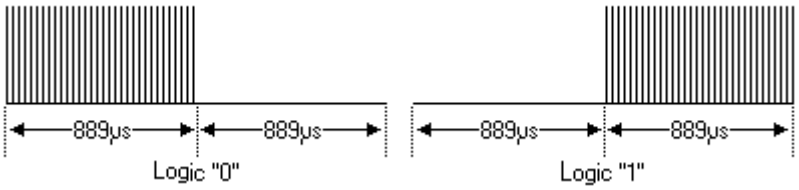


Figura 13 - Rajada de pulsos

6.3.4 Protocolo

A imagem abaixo mostra um típico trem de pulsos de uma mensagem RC-5.

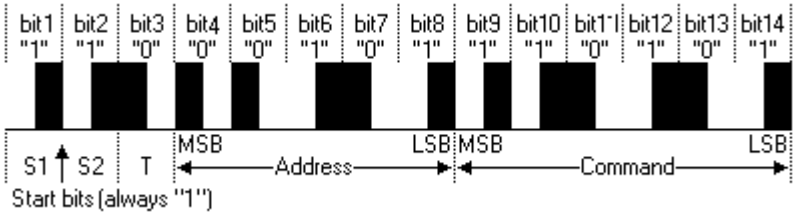


Figura 14 - Trem de pulsos RC-5

Os primeiros dois pulsos são pulsos de start, ambos de lógica 1. O terceiro bit é um bit de chaveamento. Este bit é invertido cada vez que o botão é liberado e pressionado novamente. Deste modo, o dispositivo consegue distinguir entre um botão que permanece pressionado e um botão que é pressionando repetidamente.

Os próximos cinco bits representam o endereço do dispositivo, que é enviado com o bit mais significativo à frente. Os bits subseqüentes são bits de comando, e também são enviados com o bit mais significativo à frente.

Uma mensagem consiste num total de 14 bits, cuja duração total de é de 25 ms.

O tempo em que um botão permanecer apertado, a mensagem será repetida a cada 114ms. O bit de chaveamento manterá seu estado lógico durante toda a repetição dessa mensagem. Cabe ao software do receptor interpretar esse caráter de auto repetição.

7. Resultados

Para se realizar os testes das rotinas desenvolvidas, desde aquelas com funções de interfaceamento e comunicação até rotinas de jogo propriamente ditas, utilizamos um kit didático como giga de teste. Esse kit é fornecido pela Cerne Tecnologia. Sua composição é a seguinte:

- LCD alfanumérico;
- LCD Gráfico;
- Displays de leds de 7 segmentos;
- Varredura de leds;
- Comunicação serial RS232;
- Comunicação serial RS485;
- Comunicação USB;
- Comunicação PS2;
- Comunicação com Shift Register;
- Conversão A/D;
- Botão de reset manual;
- Gravação on-board;
- Acionamento de cargas externas;
- Varredura de Teclas.

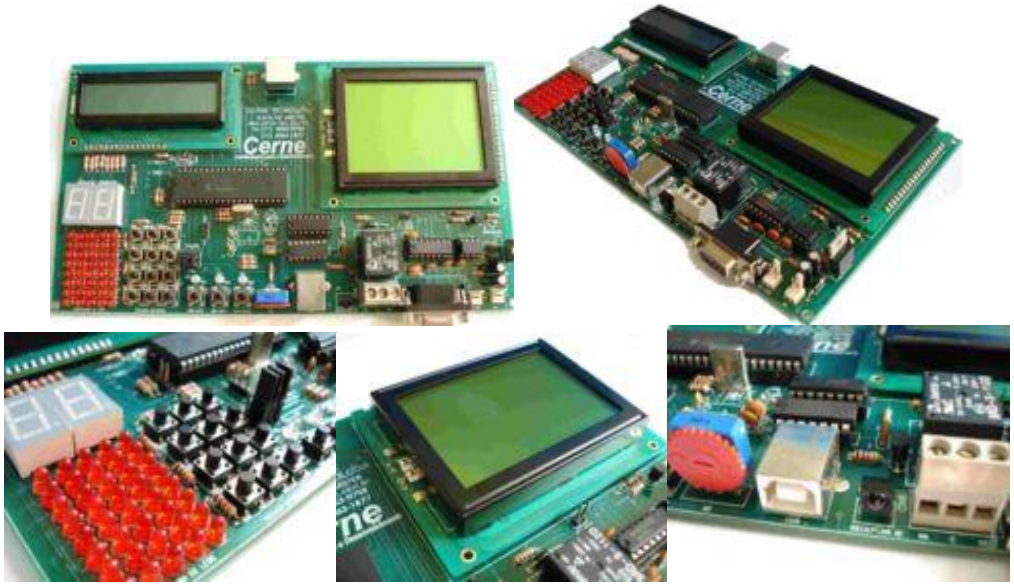


Figura 15 - Giga de teste

O uso desse kit se justifica pelo motivo de o desenvolvimento do hardware nem sempre acompanhar a velocidade do desenvolvimento do software. O uso de uma giga de teste permite ao programador mais liberdade e autonomia.

7.1 Compilador

Para a criação do código das rotinas do jogo, utilizou-se o software MikroC. A opção pela utilização desse programa se deve ao fato de ele vir com bibliotecas que facilitam

imensamente a implementação de LCD's, memórias e comunicação infravermelha. Sua estrutura simples elimina muitos problemas de programação e garante um bom andamento para o projeto.

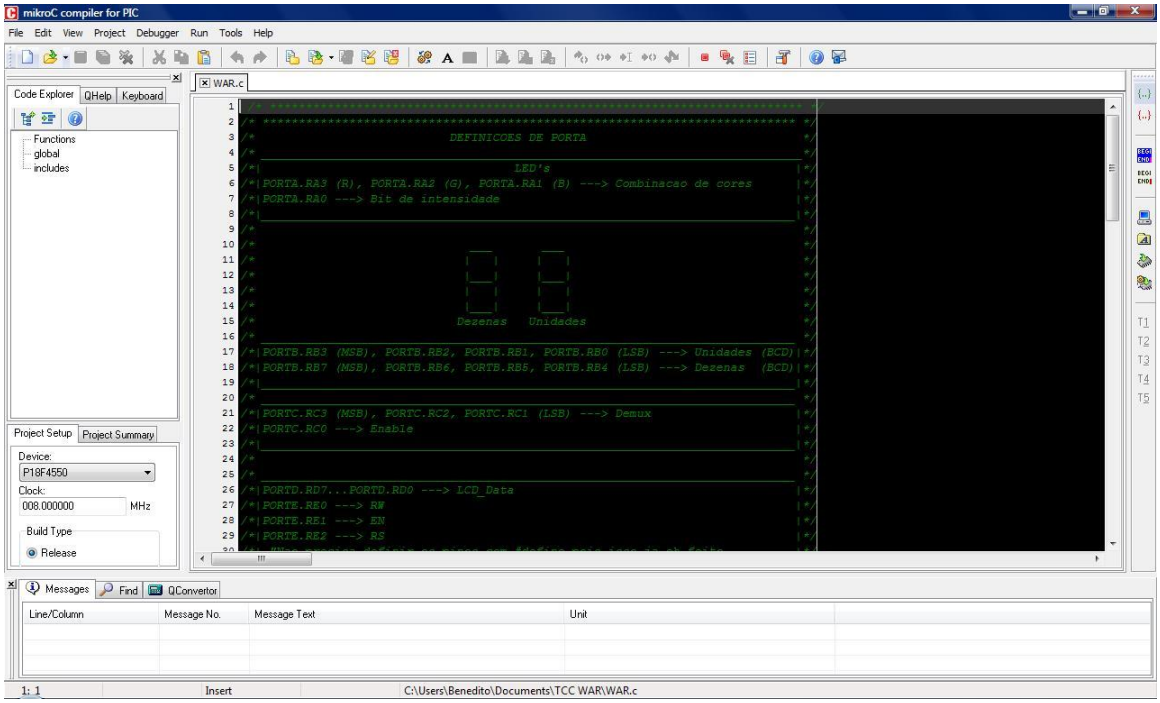


Figura 16 - MikroC

8. Conclusão

A proposta desse projeto foi o casamento entre duas vontades: aplicação de conhecimentos adquiridos ao longo da graduação e criação de um projeto de fácil interação com um usuário leigo. Assim sendo, conclui-se que os objetivos iniciais foram alcançados já que nesse projeto tivemos a chance de aplicar os conhecimentos de sistemas multiestágio aprendidos no curso de engenharia elétrica e de utilizar a interface de um jogo para interagir com um usuário comum.

Ao que compete ao software, pode-se dizer que foi um desafio iniciar as bases desse projeto e que há muito para melhorar. O desenvolvimento de um programa é algo continuo e sem fim. Deve sempre atender às exigências dos usuários e de performance.

Como melhorias futuras, pode-se sugerir a busca por um algoritmo que melhor solucione a questão da variável aleatória e o aumento da complexidade das rotinas de gerenciamento.

8. Bibliografia

ATMEL, Folha de dados do componente 24C04, 12 de Setembro de 2001.

GROW, Manual do jogador, 2004.

HITACHI, Folha de dados do componente HD44780U, 1998.

MAGELA, Rogerio. Engenharia de Software Aplicada: Princípios (volume 1). Alta Books. 2006.

MAGELA, Rogerio. Engenharia de Software Aplicada: Fundamentos (volume 2). Alta Books. 2006.

MOLINARI, Leonardo. *Gerência de Configuração - Técnicas e Práticas no Desenvolvimento do Software*. Florianópolis: Visual Books, 2007. 85-7502-210-5

MORIMOTO, Carlos. Hardware: o guia definitivo, 2ª edição, Editota Sulina, 2009.

PRESSMAN, Roger. Software Engineering: A Practitioner's Approach, 6ªedição, Mc Graw Hill, 2005.

9. Apêndice

```
#define PORTA.RA4 RC5     /* Comunicacao infravermelha*/
#define PORTA.RA3 Red
#define PORTA.RA2 Green
#define PORTA.RA1 Blue
#define PORTA.RA0 Brilho
#define OK 0x10
#define ESC 0x11
#define INC 0x20
#define DEC 0x21
#define Num_Jogadores 0x06
#define Num_Paises    0x2A

struct player
{
    char cor;
    int nterritorios;
    int territorios[Num_Paises];
} jogador[Num_Jogadores];

struct pais
{
    char cor;
    int nexercitos;
}

//char Mapa_Mundi[42][7];
char Mapa_Mundi[4][4];

/* Variaveis Globais*/
int rodadan;
int recebe_exercitos;
int objetivo_ok;
char last_rx, semente;

void decodifica_bit(void)
{
    last_rx=0;
    if (rc5) last_rx=1;

    delay_us(890); }

/* Declaracao de funcoes*/
char get_rc5(void)
```

```
{
  char comando = 0x00;
  while(!comando)
  {
    if (!rc5)     //Botão está pressionado?
    {
      delay_us(445);
      comando=0;

      delay_us(890);
      decodifica_bit();     //Bit F
                  //Descartado neste exemplo

      delay_us(890);
      decodifica_bit();     //Bit C
                  //Descartado neste exemplo

      delay_us(890);        //Bit 4 de sistema
      decodifica_bit();     //Descartado neste exemplo

      delay_us(890);        //Bit 3 de sistema
      decodifica_bit();     //Descartado neste exemplo

      delay_us(890);        //Bit 2 de sistema
      decodifica_bit();     //Descartado neste exemplo

      delay_us(890);        //Bit 1 de sistema
      decodifica_bit();     //Descartado neste exemplo

      delay_us(890);        //Bit 0 de sistema
      decodifica_bit();

      delay_us(890);
      decodifica_bit();
      if (last_rx) comando|=32;          //Bit 5 de comando

      delay_us(890);
      decodifica_bit();
      if (last_rx) comando|=16;          //Bit 4 de comando

      delay_us(890);
      decodifica_bit();
      if (last_rx) comando|=8;          //Bit 3 de comando

      delay_us(890);
```

```
            decodifica_bit();
            if (last_rx) comando|=4;          //Bit 2 de comando

            delay_us(890);
            decodifica_bit();
            if (last_rx) comando|=2;          //Bit 1 de comando

            delay_us(890);
            decodifica_bit();
            if (last_rx) comando|=1;          //Bit 0 de comando
        }
    }
    return comando;
}

void Escreve_RGB(char Parametro)
{
    char Caracter1, Caracter2, Caracter3;
    Caracter1 = (Parametro&0x01) + 0x30;
    Caracter2 = (Parametro&0x02) + 0x30;
    Caracter3 = (Parametro&0x04) + 0x30;
    Lcd8_Char(2,1, Caracter3);
    Lcd8_Char(2,2, Caracter2);
    Lcd8_Char(2,3, Caracter1);
    Lcd8_Out(2,5,"(R G B)");
}

void Sorteia_Territorios(void)
{
    int paises[Num_Paises];
    char max, indice1, indice2, indice3;
    int temporario;
    for(indice1 = 0;indice1 < Num_paises;indice++)
    {
       paises[indice] = indice;
    }
    switch(Num_Jogadores)
    {
       case 2 : jogador[0].nterritorios = 21;
                jogador[1].nterritorios = 21;
                break;

       case 3 : jogador[0].nterritorios = 14;
                jogador[1].nterritorios = 14;
                jogador[2].nterritorios = 14;
```

```
            break;

        case 4 : jogador[0].nterritorios = 10;
            jogador[1].nterritorios = 10;
            jogador[2].nterritorios = 11;
            jogador[3].nterritorios = 11;
            break;

        case 5 : jogador[0].nterritorios = 8;
            jogador[1].nterritorios = 8;
            jogador[2].nterritorios = 8;
            jogador[3].nterritorios = 9;
            jogador[4].nterritorios = 9;
            break;

        case 6 : jogador[0].nterritorios = 7;
            jogador[1].nterritorios = 7;
            jogador[2].nterritorios = 7;
            jogador[3].nterritorios = 7;
            jogador[4].nterritorios = 7;
            jogador[5].nterritorios = 7;
            break;
    }
    max = 41;
    semente = jogador[0].cor;
    for(indice1 = 0;indice1 < Num_Jogadores; indice++)
    {
      for(indice2 = 0;indice2 < jogador[indice1].nterritorios; indice2++)
      {
        srand(semente);
        temporario = (int)(rand()/32767)*max; // valor aleatorio entre 0 e max
        max--;
        indice3 = paises[temporario];
        while(indice3 < max)
        {
          paises[indice3] = paises[indice3+1];
          indice3++;
        }
        jogador[indice1].territorios[indice2] = paises[temporario];
        paises[jogador[indice1].territorios[indice2]].cor = jogador[indice].cor;

        semente += 3;
      }
    }
}
```

```
void Aloca_EXercitos(char indice)
{
  char recebe_exercitos;
  char aux = 0x00;
  char contador = 0x00;
  char exercitos = 0x00;

  recebe_exercitos = (int)jogador[indice].nterritorios/2;

  while(recebe_exercitos>0)
  {
    Lcd8_Cmd(LCD_CLEAR);
    Lcd8_Out(1,1,"Selecione o país");
    Lcd8_Out(2,1,"OK p selecionar");

    while(aux!= OK)
    {
       aux = get_rc5();
       if (aux == INC)  contador++;
       if (aux == DEC)  contator--;
       if (contador => jogador[indice].nterritorios) contador = 0x00;
       if (contador < 0x00) contador = jogador[indice].nterritorios;
       Atualiza_Territorio(jogador[indice].territorios[contador],jogador[indice].cor, 0x01, 0x00)
    }
    Lcd8_Cmd(LCD_CLEAR);
    Lcd8_Out(1,1,"N  exercitos: 01");
    Lcd8_Out(2,1,"    OK    ESC   ");

    aux =0x00;
    exercitos = 0x01
    while (!aux)
    {
        aux = get_rc5();
        if(aux == INC)
        {
           exercitos++;
           if(exercitos > recebe_exercitos) exercitos = 1;
        }
        if(aux == DEC)
        {
           exercitos--;
           if(exercitos == 0) exercitos = recebe_exercitos;
        }
        if(aux == OK)
```

```
        {
          recebe_exercitos-= exercitos;
          Atualiza_Territorio(jogador[indice].territorios[contador],jogador[indice].cor, 0x01, exercitos)

        }
    }

  }
}

void Primeira_Rodada(void)
{
  char aux = 0x00;
  char contador = 0x32;
  Lcd8_Cmd(LCD_CLEAR);
  Lcd8_Out(1,1,"Inicio de jogo!");
  delay_ms(1500);                 /* espera 1,5 s*/
  Lcd8_Cmd(LCD_CLEAR);

  // Seleciona a quantidade de jogadores
  Lcd8_Out(1,1, "Qnts jogadores?");
  delay_ms(1500);
  while(aux!= OK)
  {
    aux = get_rc5();
    if (aux = INC)  contador++;
    if (aux = DEC)  contator--;
    if (contador > (Num_Jogadores + 0x30)) contador = 0x32;
    if (contador < 0x32) contador = 0x36;
    Lcd8_Char(2,1, contador);
  }
  njogadores = contador - 0x30;

  // Escolhe a cor de cada jogador
  Lcd8_Cmd(LCD_CLEAR);
  for(contador=0x31; contador < (njogadores + 0x31); contador++)
  {
      Lcd8_Out(1,1, "Jogador: ");
      Lcd8_Char(1,10,contador);
      jogador[contador-0x30].cor = 0x01;
      while(aux!= OK)
      {
        aux = get_rc5();
        if (aux = INC)  jogador[contador-0x31].cor++;
        if (aux = DEC)  jogador[contador-0x31].cor--;
```

```
            if (jogador[contador-0x31].cor > 0x06) jogador[contador-0x31].cor = 0x01;
            if (jogador[contador-0x31].cor < 0x01) jogador[contador-0x31].cor = 0x06;
            Escreve_RGB(jogador[contador-0x31].cor);
        }
    }

    // Sorteia os territórios de cada jogador
    Sorteia_Territorios();

    // Recebe exercitos extras e os aloca segundo estrategia do jogador
    for(contador=0x31; contador < (njogadores + 0x31); contador++)
    {
        //Recebe_Exercitos();
        Aloca_Exercitos(contador - 0x31);
    }
}

void Batalha(char indice)
{
    char aux = 0x00;
    char atacar = 0x01;
    char contador = 0x00;
    char contador2 = 0x00;
    char indice2 = 0x00;
    while (atacar)
    {
        //  Seleciona a origem
        Lcd8_Cmd(LCD_CLEAR);
        Lcd8_Out(1,1,"Sel. Origem");
        Lcd8_Out(2,1,"OK Sel. ESC Fin.");

        while(aux!= OK)
        {
            aux = get_rc5();
            if (aux == INC)  contador++;
            if (aux == DEC)  contator--;
            if (aux == ESC)  atacar = 0x00;
            if (contador => jogador[indice].nterritorios) contador = 0x00;
            if (contador < 0x00) contador = jogador[indice].nterritorios;
            Atualiza_Territorio(jogador[indice].territorios[contador],jogador[indice].cor, 0x01, 0x00)
        }

        // Seleciona o destino
        if (atacar)
        {
```

```
          Lcd8_Cmd(LCD_CLEAR);
          Lcd8_Out(1,1,"Sel. Destino");
          Lcd8_Out(2,1,"OK Sel. ESC Ret.");

          while((aux != OK)||(aux != ESC))
          {
             aux = get_rc5();
             if (aux == INC)  contador2++;
             if (aux == DEC)  contator2--;
             for(indice2 = 0x00; indice2 < jogador[indice].nterritorios; indice2++)
             {
                if(contador == jogador[indice].territorios[indice2])
                {
                   if (aux == INC)  contador2++;
                   if (aux == DEC)  contator2--;
                }
             }
             if (contador2 => Num_Paises) contador2 = 0x00;
             if (contador2 < 0x00) contador2 = Num_Paises - 1;
             if (aux != ESC) Atualiza_Territorio(contador2,0x00, 0x01, 0x00);
          }
        }
      }
}

void Remanejar_Exercitos(char indice)
{

}

void RodadaN(void)
{
  char indice;
  int temporario;

  for(indice = 1; indice <= numjogadores; indice++)
  {
    //temporario = (int)jogadores[indice].nterritorios/2;
    Aloca_Exercitos(indice);

    // Batalha
    Lcd8_Cmd(LCD_CLEAR);
    Lcd8_Out(1,1, "Deseja atacar?");
    Lcd8_Out(2,1, "   OK   ESC  ");
    if (get_rc5()== OK)  Batalha(indice);
```

```
    // Remanejar
    Lcd8_Out(1,1, "Remanejar Exec?");
    Lcd8_Out(2,1, "   OK   ESC  ");
    if (get_rc5() == OK)  Remanejar_Exercitos(indice);

    // Verifica a conclusão do objetivo
    if(jogador[indice].nterritorios == Num_Paises)
    {
       objetivo_ok = 1;
       break;
    }
  }
}

void main(void)
{
    char jogar = 0x01;
    /* ******************* Inicializacao do Sistema ******************** */
         /* 76543210 */
    TRISA = 0b00010000; /* Seta RA4 como entrada e o restante como saida */
    TRISB = 0b00000000; /* Seta Porta B inteira como saida */
    TRISC = 0b00000000; /* Seta Porta C inteira como saida */
    TRISD = 0b00000000; /* Seta Porta D inteira como saida */
    TRISE = 0b00000000; /* Seta Porta E inteira como saida */
    ADCON1 = 15;        /* Deixa os pinos da forma digital */
    intcon2.intedg0=1;
    delay_ms(2000);     /* Tempo de startup do LCD*/
    Lcd8_Config(&PORTE,&PORTD,2,1,0,7,6,5,4,3,2,1,0);
    Lcd8_Init(&PORTE, &PORTD);
    Mapa_Mundi = {2,2,4,0,3,1,3,4,2,2,4,0,3,1,2,3};
    //Mapa_Mundi = {}
    /* ********************** Algoritmo Principal ********************** */
    while(jogar)
    {
       objetivo_ok = 0;
       rodadan = 1;
       Primeira_Rodada();
       while(!objetivo_ok)
       {
          rodadan++;
          RodadaN();
       }

       // Encerramento
```

```
        Lcd8_Cmd(LCD_CLEAR);
        Lcd8_Out(1,1, "Você venceu!!!");
        delay_ms(2000);
        Lcd8_Cmd(LCD_CLEAR);
        Lcd8_Out(1,1, "OK p/ reiniciar");
        Lcd8_Out(2,1, "ESC p/ desligar");
        if (get_rc5() = ESC) jogar = 0x00;
    }
}
```