UNIVERSIDADE POSITIVO
NÚCLEO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
ENGENHARIA DA COMPUTAÇÃO

JOGO DE BATALHA NAVAL *WIRELESS* DINÂMICO

Luiz Roberto Castellar Junior

Monografia apresentada à disciplina de Trabalho de Conclusão de Curso como requisito parcial à conclusão do Curso de Engenharia da Computação, orientada pelo Prof. Amarildo Geraldo Reichel.

UP/NCET
Curitiba
2008

TERMO DE APROVAÇÃO

Luiz Roberto Castellar Junior

Jogo de batalha naval *wireless* dinâmico

Monografia aprovada como requisito parcial à conclusão do curso de Engenharia da Computação da Universidade Positivo, pela seguinte banca examinadora:

Prof. Amarildo Geraldo Reichel (Orientador)

Prof. Edson Pedro Ferlin

Prof. Maristela Regina Weinfurter

Curitiba, 15 de Dezembro de 2008.

AGRADECIMENTOS

Agradeço a Deus por ter me dado o direito de viver e por me dado uma família que sempre me ajudou nas horas difíceis.

RESUMO

Este projeto descreve o desenvolvimento de um jogo eletrônico de batalha naval *wireless* dinâmico totalmente embarcado em dois consoles. Os consoles são confeccionados em acrílico transparente e possuem componentes eletrônicos (*hardware*) e códigos de *firmware* (*software* embarcado).

O sistema de *hardware* é composto basicamente por: um microcontrolador de baixo consumo de energia (responsável pelo processamento e gerenciamento de todo o sistema), um *display* gráfico *LCD* (para a interação entre o sistema e o usuário), um transceptor de radiofreqüência (para a comunicação sem fio entre os consoles) e três baterias de íon-lítio como fonte de energia.

O sistema de *firmware* foi implementado em linguagem C, sendo divido em cinco camadas. Também foi desenvolvido um *driver* específico para cada componente de *hardware* (microcontrolador, rádio transceptor de frequência, *display* gráfico *LCD*, motor e baterias). Uma dessas camadas é responsável pelo gerenciamento de todos os *drivers*.

Palavras chave: batalha naval, *LCD (Liquid Crystal Display)*, microcontrolador

DYNAMIC WIRELESS BATTLEFIELD GAME

ABSTRACT

This project describes the development of an electronic dynamic wireless battlefield game thoroughly embedded in two consoles. The consoles are built of transparent acrylic and have electronic components (hardware) and firmware code (embedded software).

The hardware system is basically compounded by: one low power microcontroller (in charge of processing and managing the system), one LCD graphic display (for the interaction between the system and the user), one radiofrequency transceiver (for the wireless communication between the consoles) and one battery pack used as power source.

The firmware system was implemented in C language, being divided in five layers. Also, specific drivers were developed for each hardware component (microcontroller, radiofrequency transceiver, LCD graphic display, engine and batteries). One of these layers is in charge of managing all the drivers.

Key words: battlefield, LCD (Liquid Crystal Display), microcontroller

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACLK - *Auxiliary clock*

ADC - *Analog Digital Converter*

A/D – Analógico/digital

CPU - *Central Processing Unit*

EEPROM - *Electrically Erasable Programmable Read-Only Memory*

EMI - *Electromagnetic interference*

EPROM - *Erasable Programmable Read-Only Memory*

GHz - *Gigahertz*

GPS - *Global Positioning System*

Hz - *Hertz*

IDE - *Integrated Development Environment*

IHM - Interface Homem Máquina

ISR - *Interrupt Service Routine*

JTAG - *Joint Test Action Group*

Kbps - *kilobit per second*

kHz - *kilohertz*

LCD - *Liquid Crystal Display*

LED - *Light-emitting diode*

LPM1 - *LOW-POWER MODE 1*

mA – *mili Ampere*

MCLK - *Main clock*

MHz - *Megahertz*

NCET - Núcleo de Ciências Exatas e Tecnológicas

PWM - *Pulse-Width Modulation*

RAM - *Random-access memory*

RFI - interferência de radio freqüência

RFID - *Radio Frequency Interference*

RISC - *Reduced instruction set computer*

ROM - *Read-Only Memory*

SMCLK - *Sub-Main clock*

SMD - *surface-mount devices*

SPI - *Serial Peripheral Interface*

UART - *universal asynchronous receiver/transmitter*

UP - Universidade Positivo

USB - *Universal Serial Bus*

V - Volt

LISTA DE SÍMBOLOS

Ω - ohm

CAPÍTULO 1 - INTRODUÇÃO

Este projeto tem como finalidade o desenvolvimento de um jogo de batalha naval *wireless* dinâmico, que possua um nível de confiabilidade e interação aceitável para os jogadores. Com isso em mente, também foram projetadas e implantadas algumas mudanças para tornar o projeto mais interessante visualmente e mais compacto, modificando algumas regras básicas do jogo de batalha naval desenvolvido por SANTOS (2006).

Devido à crescente demanda por jogos cada vez mais interativos, de menor porte e de custo acessível, foi percebida a oportunidade de suprir estas necessidades por meio do desenvolvimento deste projeto (PHAM, 2008).

Através das modificações propostas para este sistema, obteve-se uma melhoria do conforto e do bem-estar dos jogadores, além do aumento da complexidade do jogo, o que o torna mais interessante para crianças, que estão em processo de desenvolvimento do raciocínio lógico.

Este projeto dá continuidade e melhora os projetos desenvolvidos por CORSO (2005) e por SANTOS (2006). Na primeira versão, elaborada por CORSO (2005), não foi possível desenvolver os consoles e a transmissão por radiofreqüência. Na versão seguinte, melhoras foram feitas por SANTOS (2006), como a confecção dos consoles e o funcionamento da transmissão por radiofreqüência, propondo algumas melhorias que foram implementadas neste projeto, durante o ano de 2008.

Este projeto tem como objetivo principal colocar em funcionamento as principais modificações citadas por SANTOS (2006). Dessa forma, as metas do desenvolvimento deste projeto são:

- Incorporação do *display* gráfico *LCD* ao projeto;
- Melhora da confiabilidade e da distância permitida para transmissão de dados *wireless*;
- Aumento da autonomia de energia do sistema;
- Desenvolvimento do módulo dinâmico do jogo;
- Redução do tamanho dos consoles.

Este trabalho está dividido em 6 capítulos e dois apêndices, descritos a seguir.

O capítulo 1, “Introdução”, apresenta uma breve explicação do projeto e como esta monografia está organizada.

O capítulo 2, “Fundamentação teórica”, faz uma revisão da literatura.

No capítulo 3, “Especificação, desenvolvimento e implementação”, são indicados os materiais e os métodos utilizados, bem como descreve em detalhes a metodologia utilizada durante o desenvolvimento e implementação do projeto.

No capítulo 4, “Validação e resultados”, são apresentados os resultados obtidos e os testes realizados.

O capítulo 5, “Conclusão”, propõe melhoras para continuação do projeto em trabalhos futuros e discute as conclusões e descobertas decorrentes deste estudo.

O capítulo 6, “Referências bibliográficas”, apresenta as fontes consultadas durante o estudo e o desenvolvimento do projeto.

O apêndice A, “Manual do usuário”, apresenta um manual para os jogadores.

O apêndice B, “Fotos”, ilustra o projeto em seus estágios finais de construção.

O apêndice C, “Artigo”, apresenta o artigo técnico.

CAPÍTULO 2 – FUNDAMENTAÇÃO TEÓRICA

Batalha naval é um jogo de tabuleiro de dois jogadores, no qual os jogadores têm que adivinhar em que quadrados estão os navios do adversário.

Não se tem conhecimento de quem criou o jogo de batalha naval. Segundo JOGOS ANTIGOS (2006), acredita-se que tenha sido criado originalmente por soldados russos durante a 1ª Guerra Mundial, porém, há também a hipótese de que a origem esteja no jogo militar francês "L'Attaque", também desenvolvido na 1ª Guerra Mundial.

Embora o primeiro jogo de batalha naval de tabuleiro tenha sido comercializado e publicado pela Milton Bradley Company em 1931, o jogo foi originalmente jogado com lápis e papel (JOGOS ANTIGOS, 2006).

Originalmente é jogado em quatro grades quadradas, duas para cada jogador. As grades são tipicamente 10x10, e os quadrados individuais nas grades são identificados por letras e números (Figura 2.1). Em uma grade o jogador organiza seus próprios barcos e registra os tiros do adversário. Na outra grade o jogador registra os seus próprios tiros.

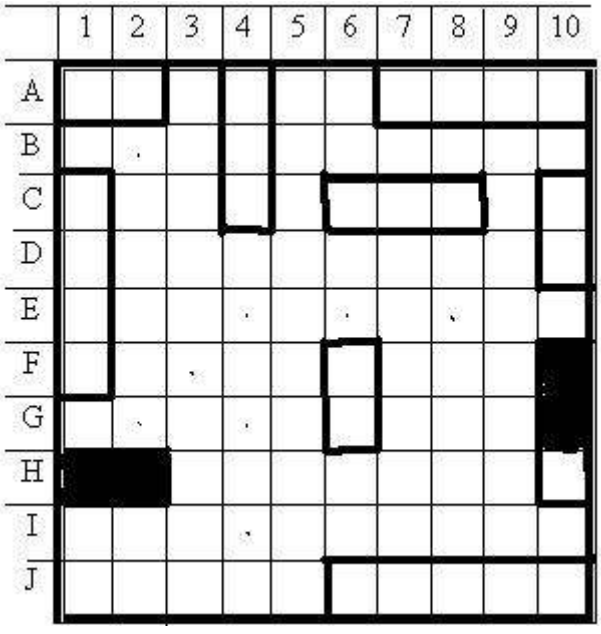

Fig. 2.1 – Jogo de Batalha Naval em papel

FONTE: WIKIPEDIA (2006)

Antes do jogo ser iniciado, cada jogador ordena um número de barcos secretamente na sua grade. Cada barco ocupa um número de quadrados consecutivos na grade, disposto horizontal ou verticalmente. O número de quadrados para cada barco é determinado pelo tipo do navio. Os barcos não podem se sobrepor. Os tipos e quantidade de barcos permitidos são os mesmos para cada jogador, podendo variar na forma e quantidade, dependendo das regras (JOGOS ANTIGOS, 2006).

A Tabela 2.1 mostra a característica típica dos navios utilizados em um jogo de batalha naval:

TABELA 2.1 – Características dos navios

Número	**Tipo de barco**	**Tamanho**
1	Porta-aviões	5
1	Navio	4
1	Cruzador	3
1	Submarino	3
1	Destróier	2

Depois que os navios são posicionados, o jogo prossegue em uma série de rodadas. Em cada rodada, cada jogador possui uma jogada. Durante a jogada, o jogador anuncia uma lista de quadrados-alvo na grade do adversário em que deseja atirar. Se um barco ocupa um dos quadrados, então ele é atingido por tiro. Quando todos os quadrados do barco foram atingidos, o barco é afundado. Depois que a lista de alvos foi divulgada, o adversário então anuncia quais dos seus barcos foram atingidos. Se ao final de uma rodada todos os barcos de um jogador foram atingidos, o jogo termina e o outro jogador vence. Se todos os barcos de ambos os jogadores foram atingidos, o jogo termina empatado.

O número de quadrados-alvo que um jogador pode atirar em uma rodada é determinado pelo número de barcos não atingidos do próprio jogador, no começo da rodada. Portanto, cada jogador tem tantos tiros quanto ele possuir barcos flutuantes, em cada rodada. Então cada vez que o barco de um jogador é inteiramente destruído, esse jogador tem um tiro a menos nas rodadas subseqüentes.

Muitas variações nas regras básicas são possíveis, incluindo o tamanho das grades, a quantidade de cada tamanho de barco, o número de tiros permitidos, e declararem ou não quando um barco

foi atingido. Algumas das variações simplificam o jogo, o que é útil para jogadores mais novos ou para pessoas inexperientes no jogo.

Na variação mais simples, jogadores alternam rodadas para atacar somente um quadrado-alvo à sua escolha por vez, com o resultado anunciado imediatamente. Esta regra é popular pela simplicidade, mas diminui o aspecto estratégico do jogo. Opcionalmente, as regras podem permitir um tiro adicional a ser disparado depois de cada tiro bem sucedido. Isso torna possível, embora pouco provável, para um jogador ganhar sem o adversário disparar um único tiro (JOGOS ANTIGOS, 2006).

2.1 - *Display* gráfico *LCD*

Um monitor de cristal líquido (ou *LCD* - *liquid crystal display*) é um dispositivo de tela fino e plano, criado a partir de uma seqüência ordenada de cores ou pontos (*pixels*) monocromáticos em frente a uma fonte luminosa ou um refletor. O *LCD* é freqüentemente utilizado em dispositivos eletrônicos a bateria por consumir pouca quantidade de energia elétrica. A Figura 2.2 ilustra o *LCD* usado no projeto.

Fig. 2.2 – *LCD* usado no projeto (frente e verso)

Cada pixel de um *LCD* normalmente consiste de uma camada de moléculas alinhadas entre dois eletrodos transparentes e dois filtros de polarização. Os eixos de transmissão são, na maioria dos casos, perpendiculares um ao outro. Sem cristal líquido entre os filtros de polarização, a luz que passa pelo primeiro filtro seria bloqueada pelo segundo filtro.

A superfície dos eletrodos que estão em contato com o material de cristal líquido é tratada de forma a alinhar as moléculas de cristal líquido em uma determinada direção. Os eletrodos são feitos de um condutor transparente chamado óxido de estanho índio (COLLINGS & HIRD, 1997).

Antes de aplicar um campo elétrico, a orientação das moléculas de cristal líquido é determinada pelo alinhamento com as superfícies. Em um dispositivo de torção nemática (ainda o dispositivo de cristal líquido mais comum), os dois eletrodos são perpendiculares um ao outro e, por isso, as moléculas organizam-se em uma estrutura helicoidal. Porque o cristal líquido é bi-refringente, a luz que passa pelo filtro polarizador é girada pela hélice de cristal líquido ao passar pela camada de cristal líquido, permitindo-lhe passar pelo segundo filtro polarizado. Metade da luz incidente é absorvida pelo primeiro filtro polarizador (GEORGE & STEPHEN, 1999).

Quando uma tensão é aplicada entre os eletrodos, as moléculas de cristal líquido se alinham paralelamente ao campo elétrico, distorcendo a estrutura helicoidal. Isso reduz a rotação da polarização da luz incidente, e o dispositivo aparece cinza. Se a tensão aplicada é suficientemente grande, as moléculas de cristal líquido no centro da camada são quase completamente separadas e a polarização da luz incidente não é girada ao passar pela camada de cristal líquido. Esta luz será polarizada perpendicularmente ao segundo filtro, portanto, será bloqueada, e o pixel será exibido preto. Ao controlar a tensão aplicada em toda a camada de cristal líquido em cada pixel, será permitida a passagem a luz em quantidades variadas, constituindo assim diferentes níveis de cinza (GEORGE & STEPHEN, 1999).

Tanto o de cristal líquido quanto a camada de alinhamento contêm compostos iônicos. Se um campo elétrico de uma polaridade particular é aplicado por um longo período de tempo, este material iônico é atraído para as superfícies e degrada o desempenho do dispositivo. Isso pode ser evitado mediante a aplicação de uma corrente alternada ou invertendo a polaridade do campo elétrico (a resposta da camada de cristal líquido é idêntica, independentemente da polaridade do campo aplicado) (COLLINGS & HIRD, 1997).

Quando um grande número de pixels é necessário, não é tecnicamente possível direcionar cada um, pois cada pixel exigiria eletrodos independentes. Em vez disso, a exibição é multiplexada. Em um display multiplexado, eletrodos em um lado da tela são agrupados e alinhados (normalmente em colunas), e cada grupo tem a sua própria fonte de tensão. Do

outro lado, os eletrodos também são agrupados (normalmente em linhas), com cada grupo recebendo uma tensão.

Em *LCDs* coloridos, cada *pixel* individual é dividido em três células ou *subpixels*, podendo ser nas cores vermelho, verde e azul, respectivamente. Cada pixel pode ser controlado independentemente para produzir milhares ou milhões de cores possíveis para cada pixel.

Existem fatores importantes que devem ser considerados ao avaliar um *LCD*, que incluem: resolução, distância entre os centros de dois pixels adjacentes (*dot pitch*), tempo de resposta, taxa de atualização, tipo de matriz (ativa ou passiva), ângulo de visibilidade, brilho, contraste e a relação entre a largura e a altura (*aspect ratio*) (LCI, 2005).

2.2 - Microcontrolador

Segundo MNEINA (2006), o microcontrolador é um dispositivo que integra um grande número de componentes de um sistema micro processado em um único micro *chip* e é otimizado para interagir com o mundo externo por meio de interfaces *on-board*.

Microcontroladores se diferem de microprocessadores de várias formas. O primeiro e mais importante é a sua funcionalidade. Para um microprocessador ser usado, outros componentes como memória, ou componentes para receber e enviar dados devem ser adicionados a ele. Por outro lado, microcontroladores são designados para ser tudo em um. Nenhum componente externo é necessário para suas aplicações porque todos os periféricos necessários já estão incluídos nele. Desta forma, poupa-se tempo e espaço na construção de dispositivos de processamento e interface com o mundo externo (MNEINA, 2006).

Portanto, um microprocessador é normalmente otimizado para coordenar o fluxo de informações entre memórias separadas e dispositivos periféricos que estão localizados fora dele. O processador de um microcontrolador e periféricos são construídos na mesma pastilha de silício, e raramente contêm estruturas de barramento estendidos para fora do seu invólucro.

Um único microcontrolador geralmente incorpora os seguintes componentes no mesmo circuito integrado (MNEINA, 2006):

- Microprocessador – de pequenos e simples processadores de 4 bits a sofisticados processadores de 32 ou 64 bits;
- Entradas/saídas – interfaces tais como portas seriais (*UARTs*);
- Periféricos – tais como *timers* e circuitos de *watchdog*;
- *RAM* – para armazenamento de dados;
- *ROM, EPROM, EEPROM* ou *FLASH* – para armazenamento de programas;
- Gerador de *clock* – freqüentemente um cristal de quartzo ou um circuito RC (Resistor-Capacitor);
- *ADC* – conversores analógico-digitais.

Esta integração reduz drasticamente a quantidade de *chips*, fios e espaço em placas de circuito impresso que seriam necessários para produzir sistemas equivalentes com estes componentes em *chips* separados.

2.3 - Comunicação *wireless*

Comunicação *wireless* é a transferência de informação à distância sem o uso de condutores elétricos ou cabos. As distâncias envolvidas podem ser curtas (poucos metros como o controle remoto de uma televisão), ou muito longas (milhares e talvez milhões de quilômetros para comunicação via rádio).

Exemplos comuns de equipamentos *wireless* em uso hoje são (*Dailywireless*, 2008):

- Telefone celular: permite conectividade para aplicações móveis e portáteis, ambos pessoais ou para negócios;
- Sistema de Posicionamento Global (*GPS* ou *Global Positioning System*): permite que motoristas de carros e caminhões, capitães de barcos e navios e pilotos de aeronaves determinem sua localização em qualquer lugar;
- Periféricos *wireless* para computador: o mouse sem fio é um exemplo comum; teclados e impressoras também podem ser ligados a um computador via wireless;
- Telefone sem fio: dispositivos com alcance limitado.
- Televisão por satélite: permite que telespectadores em qualquer localização selecionem até centenas de canais.

As seguintes situações justificam o uso de tecnologia *wireless* (*Dailywireless*, 2008):

- Alcançar uma distância além da capacidade de cabeamento normal;
- Evitar obstáculos tais como estruturas físicas, *EMI* (interferências eletromagnéticas) ou *RFI* (interferência de rádio freqüência);
- Prover um link de comunicação alternativo em caso de falha da rede.
- Ligar estações de trabalho temporárias ou portáteis;
- Superar situações aonde cabeamento normal é difícil ou financeiramente impraticável;
- Conectar remotamente redes ou usuários móveis.

A comunicação *wireless* pode ser via:

- rádio freqüência;
- microondas;
- infravermelho.

Aplicações podem envolver comunicação ponto-a-ponto, ponto-a-multiponto, *broadcasting* (distribuição de sinais de áudio/vídeo que transmitem programas para uma audiência), redes de celular e outras redes *wireless*.

A lista a seguir categoriza algumas das várias implementações, dispositivos e padrões *wireless* (*Dailywireless*, 2008):

- *Broadcasting;*
- Rádio amador;
- Comunicação via rádio;
- Telefonia sem fio;
- Celulares;
- Comunicação ponto-a-ponto de curto alcance (como o *RFID*);
- Redes de área pessoal (como o *Bluetooth*);
- Redes de computador wireless (como o *WiMAX*).

CAPÍTULO 3 – ESPECIFICAÇÃO, DESENVOLVIMENTO E IMPLEMENTAÇÃO

Este capítulo é dedicado à apresentação das tecnologias, dos materiais e dos métodos empregados no desenvolvimento e implementação do projeto.

3.1 - *Hardware*

O projeto é composto por dois consoles idênticos e cada um está inserido em uma caixa de acrílico transparente. Foi escolhido usar caixas de acrílico transparente por proporcionarem um visual diferente, com um acabamento diferenciado, e por oferecerem um nível adequado de proteção aos componentes de *hardware*. A Tabela 3.1 exibe as dimensões exatas das caixas de acrílico.

TABELA 3.1 – Dimensões das caixas de acrílico

Dimensão	**Tamanho**
Altura	3,2 cm
Comprimento	23,0 cm
Largura	12,5 cm
Espessura	4,0 mm

3.1.1 - Microcontrolador

O "cérebro" do console é o microcontrolador, pois é ele o responsável pelo controle de todo o *hardware*, e nele está armazenado o código (*firmware*) do jogo. O microcontrolador usado para o desenvolvimento deste projeto foi o MSP430F169, fabricado pela *Texas Instruments*, escolhido por ser de baixo custo, consumir pouca energia (330 uA a 1 MHz) e por possuir memória interna suficiente para armazenar o *firmware* (60 Kbytes de memória *FLASH*). Nos testes de desempenho, o MSP430F169 mostrou-se confiável para o porte deste projeto.

Estes microcontroladores estão configurados para funcionar a uma freqüência de 5 MHz (à partir do MCLK ou *Main clock*). Possuem uma arquitetura RISC de 16 bits e são alimentados por uma tensão de 3,4V. Seus periféricos (*timer*, *SPI* e *PWM*) usam o SMCLK (*Sub-Main clock*) como fonte de *clock*, que está configurado para funcionar a uma freqüência de 1 MHz. O ACLK (*Auxiliary clock*), não é usado. O conversor A/D é usado somente para calcular o

medidor da bateria (em 2,8V, a bateria está carregada e em 1,7V, a bateria está chegando ao fim). Essa verificação é feita a cada 60 segundos. O microcontrolador usa a interface *SPI* (*Serial Peripheral Interface*) para comunicação com o transceptor de radiofreqüência (dispositivo responsável pela transmissão *wireless*), que possui maior taxa de transferência de dados do que o *I²C*, o *SMBus* ou o *RS-232*. Possui seis modos de operação: um modo ativo e cinco modos *LOW-POWER* selecionáveis via software. Quando não está ativo, o microcontrolador está configurado para entrar em modo *LOW-POWER* 1 (LPM1). No modo LPM1, a *CPU* é desligada, o ACLK e o SMCLK permanecem ativos, e o MCLK é desligado. Desta forma economiza-se bateria por deixar o microcontrolador em modo de baixo consumo.

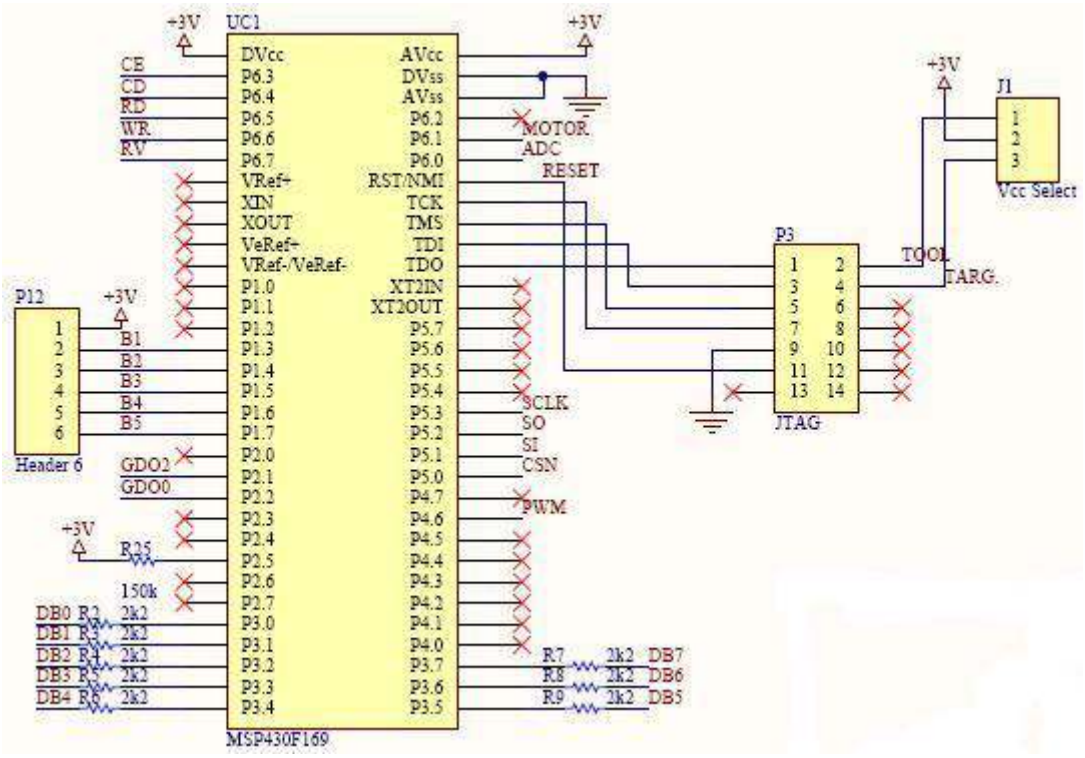

Fig. 3.1 – Diagrama esquemático do microcontrolador

Os pinos P1.0 a P1.7 são pinos de entradas/saídas digitais para propósitos gerais, e estão configurados da seguinte forma: os pinos P1.0, P1.1 e P1.2 não são utilizados, enquanto que os pinos P1.3 a P1.7 estão ligados diretamente nos botões usados como interface de entrada de dados pelos jogadores. Esses pinos estão configurados para gerar uma interrupção quando há uma borda de descida (sinal que estava em 1 descendo para 0), para que o tratamento do pressionamento dos botões seja feito.

Os pinos P2.0 a P2.7 são pinos de entradas/saídas digitais para propósitos gerais, e estão configurados da seguinte forma: os pinos P2.0, P2.3, P2.4, P2.6 e P2.7 não são utilizados, os

pinos P2.1 (está configurado para gerar uma interrupção quando há uma borda de descida) e P2.2 são ligados ao pinos GDO2 e GDO0 do RF, respectivamente, e são usados para controlar o status da transmissão de dados do RF (término e início da transmissão). O pino P2.5 é ligado a um resistor de 100 k, que é ligado ao Vcc, com a finalidade de melhorar a imunidade a ruídos, bem como ajudar a estabilizar o oscilador a cristal.

Os pinos P3.0 a P3.7 são pinos de entradas/saídas digitais para propósitos gerais, e são usados como barramento de dados do *LCD*.

Os pinos P4.0 a P4.7 são pinos de entradas/saídas digitais para propósitos gerais, e somente o pino P4.6 é utilizado, para geração do sinal *PWM* de entrada para o contraste do *display* gráfico *LCD*.

Os pinos P5.0 a P5.7 são pinos de entradas/saídas digitais para propósitos gerais, e estão configurados da seguinte forma: os pinos P5.0, P5.1, P5.2 e P5.3 são ligados aos pinos CSn (*chip select*), SI (entrada serial de dados), SO (saída serial de dados) e SCLK (entrada do sinal de *clock*) do CC1100, respectivamente. Os pinos P5.4, P5.5, P5.6 e P5.7 não são utilizados.

Os pinos P6.0 a P6.7 são pinos de entradas/saídas digitais para propósitos gerais, e estão configurados da seguinte forma: o pino P6.0 é usado como entrada/saída de dados do conversor analógico-digital (usado no medidor de bateria), o pino P6.1 é usado para acionar o circuito do motor, o pino P6.2 não é utilizado, e os pinos P6.3 a P6.7 são usados para comunicação com o *LCD*.

Os pinos AVcc e DVcc são usados como fonte de tensão para alimentar os terminais analógicos e digitais do microcontrolador, respectivamente.

Os pinos AVss e DVss são usados como fonte de aterramento para os terminais analógicos e digitais do microcontrolador, respectivamente.

O pino RST/NMI é usado para reiniciar o microcontrolador, e está conectado ao circuito de JTAG. Normalmente reinicia-se o microcontrolador após efetuar mudanças no *firmware* (pois foi necessário regravá-lo).

Os pinos TCK, TDI/TCLK, TDO/TDI e TMS são pinos de teste, e, quando usados em conjunto com o circuito de JTAG, como é o caso, cumprem a função de facilitar o acesso à programação do *firmware* na memória *FLASH*. Eles são, respectivamente: pino de entrada do *clock* , pino de entrada de dados, pino de saída de dados e seletor do modo de teste. Estes quatro pinos são usados em conjunto com o circuito de JTAG para leitura e gravação do *firmware*.

Os pinos Ve_ref+, Vref+, Vref- / Ve_ref-, XIN e XT2IN, XOUT e XT2OUT são, respectivamente: pino de entrada para tensão de referência externa, pino de saída do terminal positivo para a tensão de referência, terminal negativo para a tensão de referência, porta de entrada para osciladores de cristal, terminal de saída dos osciladores de cristal. No entanto, nenhum desses pinos é usado no projeto.

A Tabela 3.2 lista a conexão de cada pino do microcontrolador, discutida nos parágrafos acima.

TABELA 3.2 – Lista de conexões dos pinos do microcontrolador

Terminal (pino)	**Descrição / função**
P1.0	Não usado
P1.1	Não usado
P1.2	Não usado
P1.3	Conectado no botão 1
P1.4	Conectado no botão 2
P1.5	Conectado no botão 3
P1.6	Conectado no botão 4
P1.7	Conectado no botão 5
P2.0	Não usado
P2.1	Conectado no GDO2 do CC1100
P2.2	Conectado no GDO0 do CC1100
P2.3	Não usado
P2.4	Não usado
P2.5	Conectado à um resistor de 100 kohms, que é ligado ao Vcc
P2.6	Não usado
P2.7	Não usado

P3.0	Conectado ao barramento de dados do *LCD*
P3.1	Conectado ao barramento de dados do *LCD*
P3.2	Conectado ao barramento de dados do *LCD*
P3.3	Conectado ao barramento de dados do *LCD*
P3.4	Conectado ao barramento de dados do *LCD*
P3.5	Conectado ao barramento de dados do *LCD*
P3.6	Conectado ao barramento de dados do *LCD*
P3.7	Conectado ao barramento de dados do *LCD*
P4.0	Não usado
P4.1	Não usado
P4.2	Não usado
P4.3	Não usado
P4.4	Não usado
P4.5	Não usado
P4.6	Sinal *PWM* de entrada para o circuito inversor de tensão
P4.7	Não usado
P5.0	Conectado ao CSn (*chip select*) do CC1100
P5.1	Conectado ao SI (entrada serial de dados) do CC1100
P5.2	Conectado ao SO (saída serial de dados) do CC1100
P5.3	Conectado ao SCLK (entrada do sinal de *clock*) do CC1100
P5.4	Não usado
P5.5	Não usado
P5.6	Não usado
P5.7	Não usado
P6.0 / A0	Entrada/saída de dados do conversor *AD* (medidor de bateria)
P6.1 / A1	Aciona o circuito do motor
P6.2 / A2	Não usado
P6.3 / A3	Conectado no CE (*Chip Enable*) do *display LCD*
P6.4 / A4	Conectado no CD (*Code/Data*) do *display LCD*
P6.5 / A5	Conectado no RD (*Data Read*) do *display LCD*
P6.6 / A6	Conectado no WR (*Data Write*) do *display LCD*
P6.7 / A7	Conectado no RV (*Reverse Data In*) do *display LCD*
AVcc	Fonte de tensão para alimentar os terminais analógicos (Vcc)
AVss	Fonte de aterramento para os terminais analógicos (GND)

DVcc	Fonte de tensão para alimentar os terminais digitais (Vcc)
DVss	Fonte de aterramento para os terminais digitais (GND)
RST/NMI	Pino de reset. Está conectado ao JTAG
TCK	Pino de entrada do *clock*. Está conectado ao JTAG
TDI/TCLK	Pino de entrada de dados. Está conectado ao JTAG
TDO/TDI	Pino de saída de dados. Está conectado ao JTAG
TMS	Seletor do modo teste. Está conectado ao JTAG
Ve_ref+	Não usado
Vref+	Não usado
Vref- / Ve_ref-	Não usado
XIN	Não usado
XOUT	Não usado
XT2IN	Não usado
XT2OUT	Não usado

3.1.2 - *LCD*

A interface entre o *hardware* e o usuário é feita por meio do *LCD* RG241281 - *display* de cristal líquido - cujo fabricante é a *Optech Tecnologia* que possui 240 *pixels* que formam as colunas, por 128 *pixels* que formam as linhas, sendo que o tamanho de cada *pixel* é 0,47 mm por 0,47 mm. Suas medidas são: 17,0 cm de comprimento, 10,3 cm de altura e 1,4 cm de profundidade. É alimentado por +5 V. Configurar o *LCD* foi um dos maiores desafios deste projeto pelo fato de existir pouca documentação e nenhum exemplo de utilização deste modelo de *LCD* específico. Houve grande dificuldade também na configuração da biblioteca para o correto funcionamento dos pinos de controle do *LCD*. Somente após muito tempo gasto na preparação e configuração foi possível realizar os testes necessários. Os testes foram bem sucedidos e compreenderam a apresentação de textos na tela bem como o desenho de objetos.

A Figura 3.2 ilustra o diagrama esquemático do *LCD* e também o circuito inversor de tensão, usado no fornecimento dos -12V necessários para gerar o contraste do *LCD*.

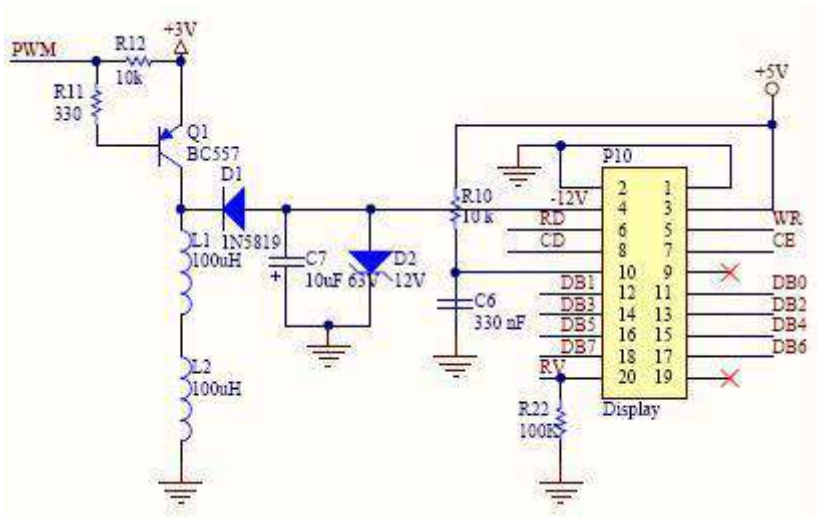

Fig. 3.2 – Diagrama esquemático do *LCD*

Os pinos FGND (*Frame Ground*) e V_{SS} são usados como fonte de aterramento, e por isso estão conectados no GND.

Os pinos V_{DD} e V_{EE} são usados como fonte de alimentação para o LCD (+5V) e para o contraste (-12V), respectivamente.

Os pinos WR (*data write*) e RD (*data read*) são usados para controle de escrita e leitura de dados, respectivamente.

O pino CE (*chip enable*) é usado para habilitar e selecionar o *LCD*, tornando-o disponível para uso.

O pino C/D (*code/data*) seleciona o tipo de informação transmitida entre o *LCD* e o microcontrolador (código/comandos ou dados).

O pino RESET é usado para reiniciar o *LCD*.

Os pinos DB0 a DB7 são usados como barramento de dados entre o *LCD* e o microcontrolador.

A função do pino FS (*Font Select*) é de selecionar a fonte de alimentação, enquanto o pino NC não possui função alguma. Ambos não são usados.

O pino RV (*Reverse Data In*) é usado para mudar a cor de fundo do *LCD* (fundo azul com *pixels* brancos ou fundo branco com *pixels* azuis).

A Tabela 3.3 lista a conexão de cada pino do *LCD*, discutida nos parágrafos acima.

TABELA 3.3 – Lista de conexões dos pinos do *LCD*

Terminal (pino)	**Descrição / função**
FGND	Aterramento
V_{SS}	Aterramento
V_{DD}	Fonte de alimentação (+5V)
V_{EE}	Fonte de alimentação do contraste (-12V)
WR	Pino de controle de escrita de dados
RD	Pino de controle de leitura de dados
CE	*Chip Enable*
C/D	Pino de escrita/leitura de dados/comandos
NC	Não usado
RESET	Pino de reset
DB0	Barramento de dados
DB1	Barramento de dados
DB2	Barramento de dados
DB3	Barramento de dados
DB4	Barramento de dados
DB5	Barramento de dados
DB6	Barramento de dados
DB7	Barramento de dados
FS	Não usado
RV	*Reverse data in* (controle da cor de fundo)

3.1.3 - Circuito do *backlight*

Backlight é a forma de iluminação usada em *displays LCDs*. *Backlights* se diferem de *frontlights* porque a fonte luminosa vem de trás ou de lado, enquanto nos *frontlights* a fonte luminosa vem de frente. O circuito do *backlight* serve para ligar o *backlight* do *LCD*, que é alimentado por +4,8 V. O *backlight* nada mais é do que uma cadeia de *LEDs*.

O pino A é o anodo e o pino K é o catodo do LED. Quando o LED é polarizado, a luz é emitida, e o *backlight* é ligado, conforme a Figura 3.3.

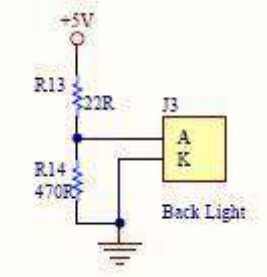

Fig. 3.3 – Diagrama esquemático do circuito do *backlight*

3.1.4 - Botões

A interação entre o usuário e o *hardware* é efetuada por meio de um botão com cinco teclas, como em um controle de videogame convencional (para cima, para baixo, esquerda, direita e tiro). A Figura 3.4 ilustra o botão direcional usado em cada console do projeto.

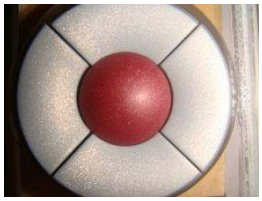

Fig. 3.4 – Botão direcional e de tiro

O botão é conectado diretamente nas entradas dos pinos do microcontrolador, conforme mostra a Figura 3.5. Desta forma, economiza-se espaço físico.

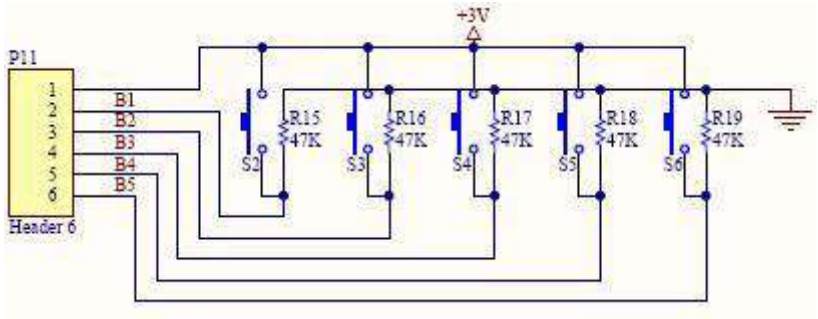

Fig. 3.5 – Diagrama esquemático do botão

Primeiramente foram testados 5 *push-buttons* ligados diretamente nos pinos do microcontrolador para atuarem como botões de tiro e de movimentação. Foi verificado que o manuseio (por parte do jogador) ficaria complicado, partindo então para a escolha de outro tipo de botão, que é o que foi usado, por ser de melhor manuseio e por ter um visual mais arrojado. Cada console possui também um botão LIGA/DESLIGA.

3.1.5 - Transceptor de radiofreqüência

Para a transmissão sem fio entre os consoles, foi escolhido o CC1100, um transceptor de radiofreqüência (ou simplesmente CC1100) que opera até 1 GHz, fabricado pela *Texas Instruments*. O transceptor de radiofreqüência é um componente desenhado para aplicações *wireless* de baixo consumo de energia, cuja transmissão e recepção de sinais de rádio estão integradas em um único *chip*. Este componente foi escolhido por ser de baixo custo e por atender às expectativas quanto à distância máxima para transmissão de dados (até 30 metros sem obstáculos) e pela sua confiabilidade. A Figura 3.6 ilustra o CC1100 usado no projeto.

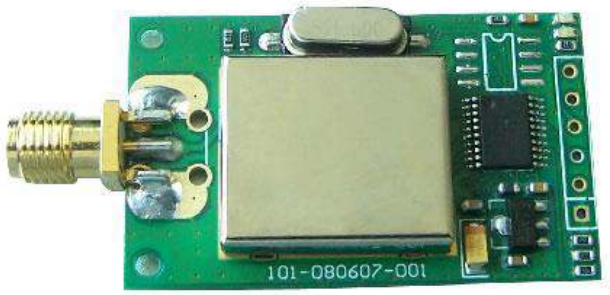

Fig. 3.6 – Transceptor de radiofreqüência CC1100

O CC1100 consome pouca energia (16,4 mA a 915 MHz), possui alta sensibilidade (-93 *dBm*), e está configurado para operar a uma freqüência de 915 MHz, o que garante uma taxa de transmissão de dados de 250 *Kbps*. Assim como o microcontrolador, o CC100 está alimentado por uma tensão de +3,4V. O oscilador usa um cristal externo com dois capacitores e gera a freqüência de referência, configurada para operar a 26 MHz. A largura de banda do filtro para canais digitais (*digital channel filter bandwidth*) está configurada para 540 KHz. A sensibilidade está otimizada para o máximo, a -93 *dBm*. A técnica de modulação digital utilizada é a MSK (*Minimum Shift Keying*), que alterna freqüências em uma fase contínua. A interface serial *SPI* é usada em modo *half-duplex* para a configuração do CC1100 e para

acessar o *buffer* de dados. A potência de saída está configurada em +10 *dBm*, o máximo disponível.

O pino SCLK é o pino de entrada do *clock*. O *clock* está configurado com a freqüência de 1 MHz, com saída do SMCLK do microcontrolador, dispensando, portanto, o uso de um cristal externo, e consequentemente os pinos XOSC_Q1 e XOSC_Q2.

O pino SO (GDO1) é usado para a saída de dados, enquanto que o pino SI é usado para a entrada de dados.

O pino GDO2 é usado para determinar o status do sinal no início da transmissão, enquanto que o pino GDO é usado para determinar o status do sinal no fim da transmissão. Ou seja, são usados como *flag* para sinalizar o início e o fim da transmissão.

O pino CSn (*chip select*) é usado para habilitar e selecionar o CC1100, tornando-o disponível para uso.

Os pinos GND (*Ground*) são usados como fonte de aterramento, e por isso estão conectados no GND.

Os pinos DVDD (*digital power supply*) e AVDD (*analog power supply*) são usados como fonte de alimentação (+3,4V).

A Tabela 3.4 lista a conexão de cada pino do CC1100.

TABELA 3.4 – Lista de conexões dos pinos do CC1100

Terminal (pino)	**Descrição / função**
SCLK	Pino de entrada do *clock*
SO (GDO1)	Pino de saída de dados
GDO2	Usado para determinar o status do sinal no início da transmissão
DVDD	Fonte de alimentação (+3,4V)
DCOUPL	Não usado
GDO0	Usado para determinar o status do sinal no fim da transmissão
CSn	*Chip Select*

XOSC_Q1	Não usado
AVDD	Fonte de alimentação (+3,4V)
XOSC_Q2	Não usado
AVDD	Fonte de alimentação (+3,4V)
RF_P	Não usado
RF_N	Não usado
AVDD	Fonte de alimentação (+3,4V)
AVDD	Fonte de alimentação (+3,4V)
GND	Aterramento
RBIAS	Não usado
DGUARD	Não usado
GND	Aterramento
SI	Pino de entrada de dados

A Figura 3.7 exibe o diagrama esquemático do circuito do CC1100.

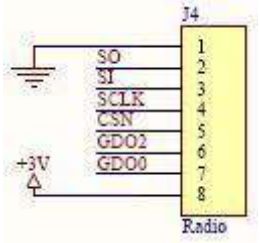

Fig. 3.7 – Diagrama esquemático do circuito do CC1100

O *software* usado para configurar o CC1100 é o *SmartRF Studio*, desenvolvido pela própria *Texas Instruments* (Fig. 3.8).

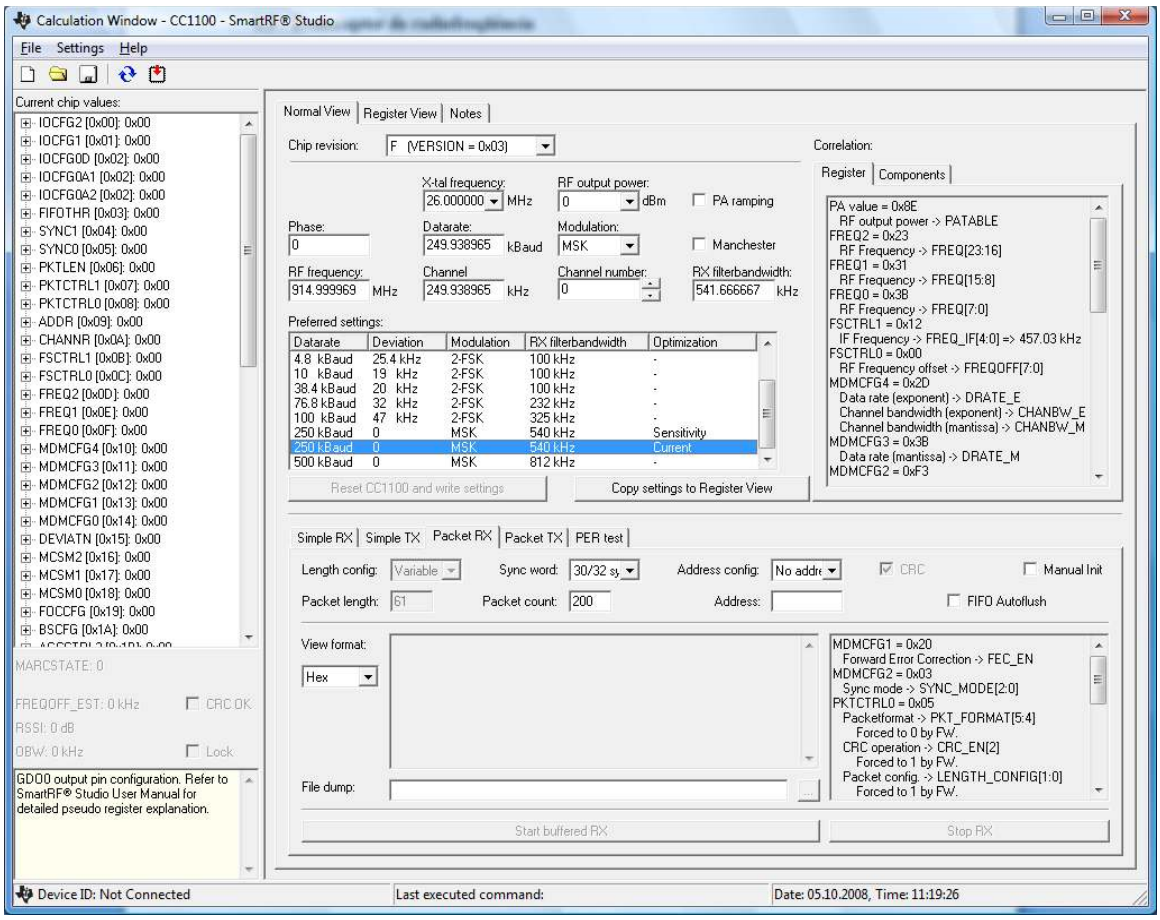

Fig. 3.8 – *SmartRF Studio*, software usado para configuração do CC1100

3.1.6 - Fonte de energia e redutores de tensão

Como fonte de energia são utilizadas três baterias de íon-lítio de 3,7V e 2200 mA cada, formando um *pack* de baterias de 11,1V a 2200 mA (modelo 18650, do fabricante *Powerizer*).

Inicialmente foram feitos testes com duas baterias alcalinas de 9V e 200 mA cada, porém após a substituição, a autonomia do sistema passou de cerca de 30 minutos para cerca de 3 horas e 30 minutos. Para funcionar adequadamente, o *display LCD* e seu *backlight* necessitam de +5,0V e +4,8V de tensão, respectivamente, enquanto o rádio transceptor, o motor e o microcontrolador, de +3,4V. Portanto, o pacote de três baterias tornou-se a opção mais interessante devido à sua alta autonomia.

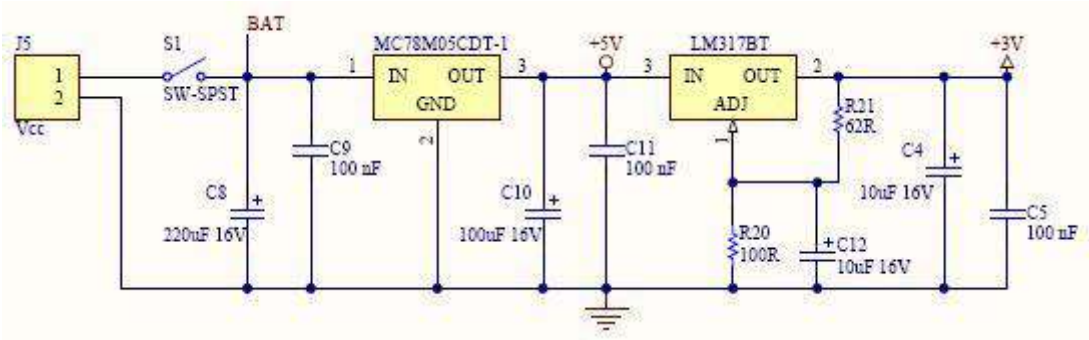

Fig. 3.9 – Diagrama esquemático do circuito de alimentação, com os redutores de tensão

Como é necessário reduzir a tensão de entrada para a tensão que cada componente consome, são usados dois redutores de tensão para este fim. Os reguladores de tensão usados são da *National Semiconductor* nos modelos LM7805 (para saída de +5,0V) e LM317 (para saída de +3,4V).

O redutor de tensão LM7805 reduz a tensão de entrada de +11,1 V para +5 V, para alimentar o *display LCD* (+ 5,0 V) e seu *backlight* (+ 4,8 V).

O redutor de tensão LM317 reduz a tensão de entrada de +5,0 V para +3,4 V, para alimentar o microcontrolador (+ 3,4 V), o transceptor de radiofreqüência (+ 3,4 V) e o motor (+ 3,4 V).

3.1.7 - Circuito de *reset*

O circuito de *reset* serve para ligar o microcontrolador somente quando a alimentação estiver estabilizada, assim ele estará sendo protegido e sua vida útil será prolongada.

A saída do circuito de *reset* é ligada no pino RST/NMI do microcontrolador.

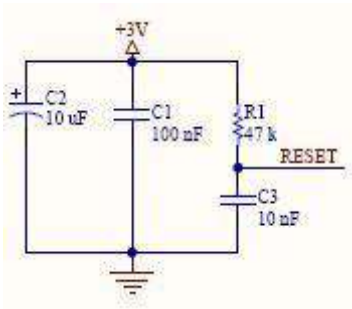

Fig. 3.10 – Diagrama esquemático do circuito de *reset*

3.1.8 - Circuito de JTAG

Um circuito de JTAG é um mecanismo para *debug* de *software* em sistemas embarcados – *firmware* - e, no caso deste projeto, cumpre a função de facilitar o acesso à programação do *firmware* na memória *FLASH*, por meio dos seus pinos de teste. O circuito de JTAG usado para gravação é conectado via *USB*.

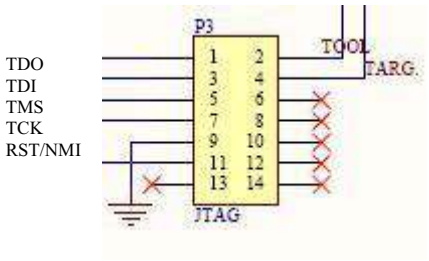

Fig. 3.11 – Diagrama esquemático do circuito de JTAG

3.1.9 - Circuito do motor

Este circuito serve para acionar um pequeno motor do tipo usado em celulares para a função vibra call, caso um barco seja atingido ou caso o jogo termine. Quando o motor precisa ser acionado, o transistor Q2 (Figura 3.12) é ativado pelo microcontrolador, acionando o motor. O diodo em paralelo com o motor serve para proteger o transistor da corrente gerada pelo chaveamento do motor.

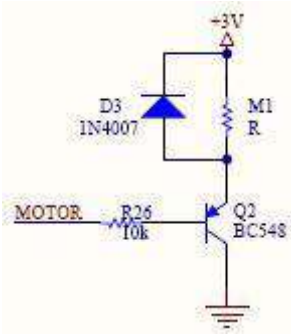

Fig. 3.12 – Diagrama esquemático do circuito do motor

A Figura 3.13 ilustra o diagrama esquemático completo do projeto de *hardware*.

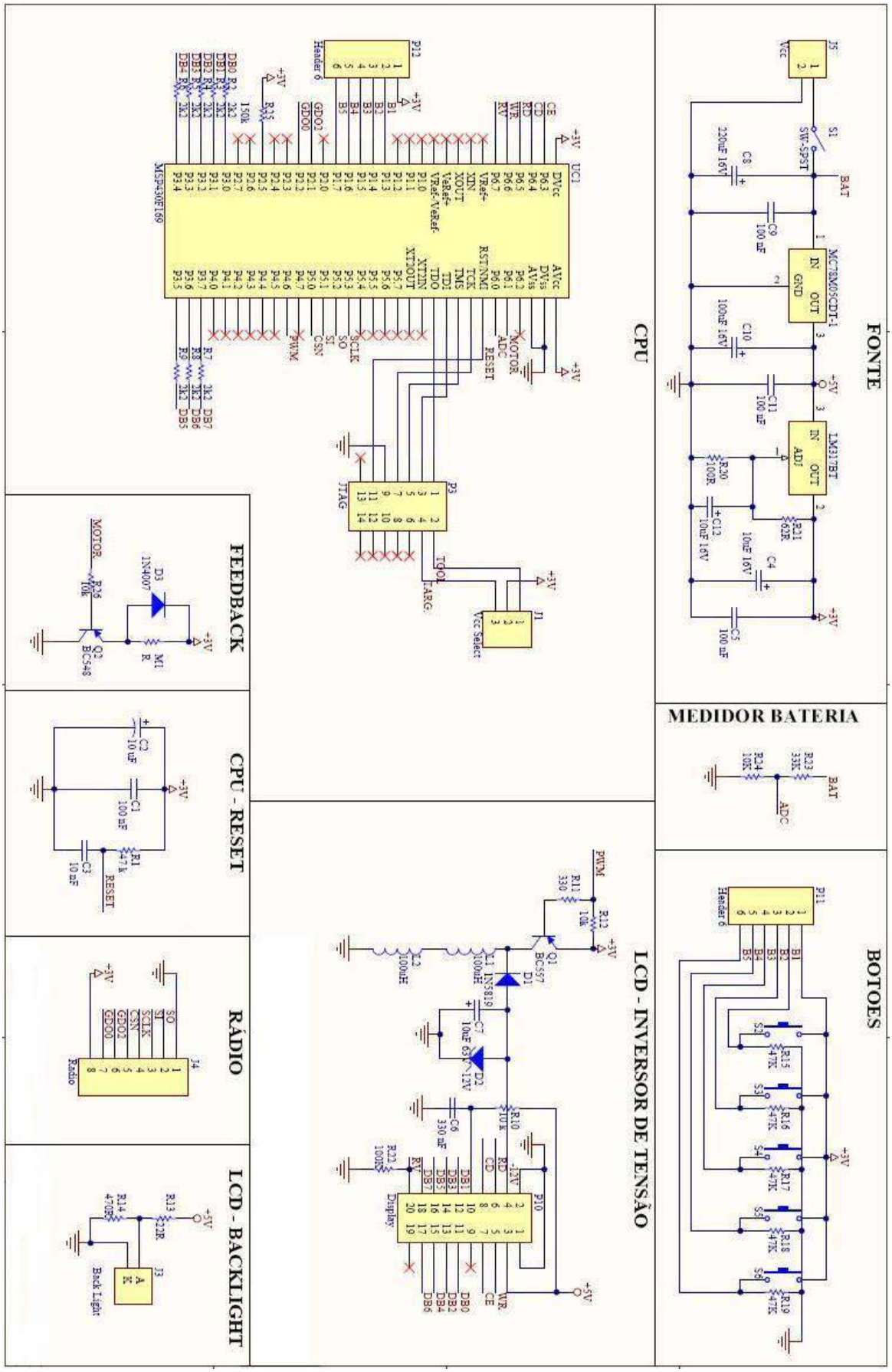

Fig. 3.13 – Diagrama esquemático do projeto de *hardware*

A Tabela 3.5 lista alguns dos principais testes realizados com os dispositivos de *hardware*, para certificar que o funcionamento dos mesmos correspondesse às expectativas e necessidades do projeto.

TABELA 3.5 – Testes realizados com os dispositivos de *hardware*

Componente	Teste	Resultado
MSP (*clock* principal)	Teste 1: *clock* interno de 1 MHz Teste 2: *clock* interno de 4 MHz Teste 3: *clock* interno de 8 MHz	Somente com 5 MHz obteve-se o desempenho esperado, sem atrasos.
MSP (*clock* do timer)	Teste 1: cristal externo de 32,768 Hz Teste 2: *clock* secundário interno de 4 MHz Teste 3: *clock* secundário interno de 1 MHz	Somente com 1 MHz obteve-se o desempenho esperado, estável.
LCD (contraste)	Teste 1: freqüência *PWM* de 1 MHz Teste 2: freqüência *PWM* de 20 kHz num ciclo ativo de 75%	Somente no teste 2 obteve-se o contraste desejado (-12V na saída do circuito inversor).
Botão	Teste 1: conectado diretamente no MSP Teste 2: conectado diretamente no MSP, mas com atraso por *software* (*debounce*)	O ruído foi eliminado com um atraso de 50ms por *software*.
Baterias (autonomia)	Teste 1: alimentação do circuito com 2 baterias alcalinas de 9V e 200 mA cada Teste 2: alimentação do circuito com 3 baterias íon-lítio de 3,7V e 2200 mA cada	A autonomia do console passou de cerca de 30 minutos para cerca de 2 horas e 30 minutos.
Baterias (medidor)	Teste 1: tempo de amostragem (4 ciclos) Teste 2: tempo de amostragem (128 ciclos)	Aumentando os ciclos de *clock* de 4 para 128 amostras, o medidor tornou-se mais preciso.

A Tabela 3.6 lista todos os componentes de *hardware* utilizados na construção do projeto.

TABELA 3.6 – Componentes de *hardware* utilizados no projeto

Componente	Quantidade
Transceptor de radiofreqüência CC1100	2
Microcontrolador MSP430F169	2

Display gráfico *LCD* RG241281	2
Regulador de tensão LM7805	2
Regulador de tensão LM317	2
Caixa acrílica transparente	2
Baterias de íon-lítio de 3,7V	6
Motor	2
Botão	2
Componentes discretos	Resistores, capacitores, diodos, etc.

3.2 - *Firmware*

Primeiramente foram estudados quais componentes de *hardware* deveriam ser usados para a obtenção de um desempenho satisfatório de velocidade de processamento, ocupação do menor espaço físico, menor consumo de energia possível e viabilidade econômica.

Foi então escolhida a linguagem de programação C para desenvolver o *firmware*, por ser uma linguagem de programação apropriada para a interação entre componentes de *hardware* e o *firmware*.

Inicialmente foi pensado em desenvolver o *firmware* em 4 camadas: camada principal (responsável por inicializar e configurar todo o sistema), camada de sistema (coordena toda a inteligência do jogo), camada de comunicação (responsável por toda comunicação entre o *firmware* e o *hardware*) e a camada de *hardware* (gerencia os componentes de *hardware* em todos os seus aspectos). Este modelo começou a ser desenvolvido e chegou a ser testado, porém verificou-se que a manutenção se tornaria demasiadamente complexa na medida em que as várias camadas fossem construídas. Pensando em diferentes maneiras de desenvolver o *firmware*, chegou-se à conclusão que a melhor abordagem para tal seria desenvolvê-lo em cinco camadas, e em *drivers*, ou seja, cada dispositivo de *hardware* teria o seu próprio arquivo, um separado do outro.

O jogo é baseado em estados, no qual cada estado tem uma ou mais formas de ser acessado e também uma ou mais maneiras de mudar de estado. A Figura 3.14 ilustra o diagrama de estados do projeto de *firmware* do jogo, que será explicado adiante com mais detalhes.

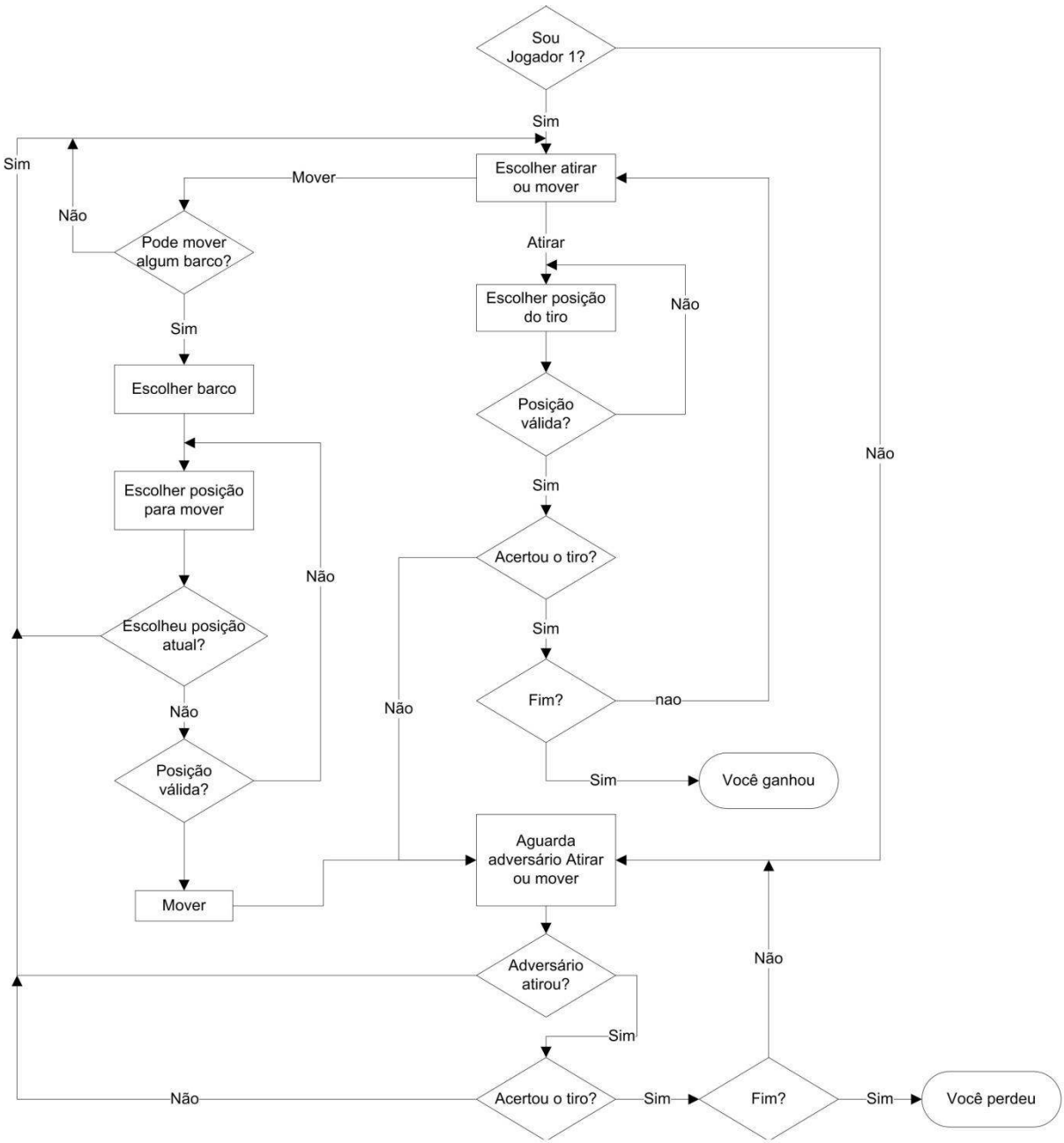

Fig. 3.14 – Diagrama de estados

Nesta terceira versão do jogo de Batalha Naval, o que o diferencia ainda mais do jogo feito por SANTOS (2006), é a opção de poder escolher mover o barco para uma posição vazia, isto é, que não tenha sido atingida por um tiro. A opção de mover o barco para uma posição vazia ocorre somente quando for a vez do jogador atirar. Ao invés de atirar, ele opta por mover um de seus barcos. Esta nova característica do jogo foi denominada como sendo “dinâmica”.

A Figura 3.15 ilustra o fluxograma do sistema de comunicação do projeto.

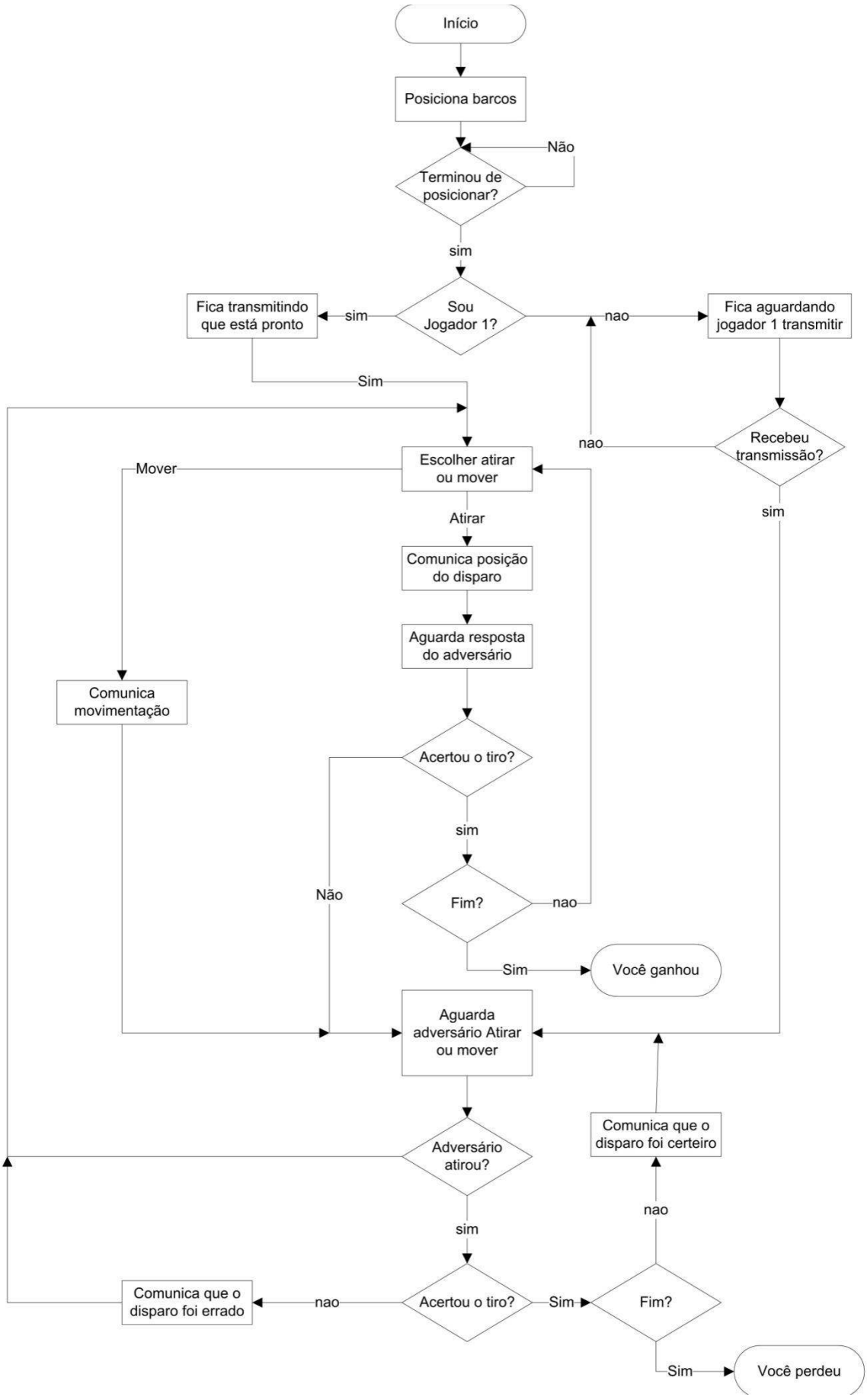

Fig. 3.15 – Fluxograma do sistema de comunicação

3.2.1 - Camadas e *drivers*

O *firmware* é dividido em cinco camadas, em que cada camada possui um ou mais *drivers*, ou seja, cada dispositivo de *hardware* tem o seu próprio arquivo, um separado do outro. Desta forma fica muito mais fácil realizar manutenções, devido à melhor organização. Portanto, foi desenvolvido um *driver* específico para cada componente de *hardware*.

A camada do microcontrolador (*HW_MSP*) é a camada responsável por configurar os *clocks* do microcontrolador (*MCLK* e *SMCLK*), os pinos, o modo de *LOW-POWER* e por configurar, desativar e ativar o *timer*.

A segunda camada é a camada dos *drivers* dos componentes periféricos de *hardware*. Cada *driver* é responsável pela inicialização, configuração e operação do seu respectivo periférico. Foram desenvolvidos 5 *drivers*: IHM (Interface Homem Máquina), *LCD* (*display* gráfico *LCD*), bateria (medidor de bateria), motor e rádio (CC1100).

A camada Sistema Operacional é a camada que configura e define as estruturas de uso do sistema. Responsável por desenhar objetos (tabuleiros, medidor de bateria, indicador do rádio, barcos, água, barco quebrado) e escrever textos na tela do *display LCD*, inicializa os periféricos, liga o motor, envia, recebe e faz o tratamento dos pacotes recebidos.

A camada Batalha Naval possui a máquina de estados, a lógica e a inteligência do jogo de batalha naval em si.

A quinta camada não possui *drivers*, mas sim arquivos que gerenciam toda a estrutura abaixo dessa camada: *main* e *ISR*. O *main* é responsável pela inicialização do sistema e pelo gerenciamento da máquina de estados. O *ISR* (*Interrupt Service Routine*) é responsável pelo tratamento das interrupções (GDO0, GDO2, conversor A/D, botão, *timer*). Interrupções, como o próprio nome diz, são "desvios" que ocorrem no decorrer do ciclo de processamento normal do programa, para tratamento de um evento específico, como o pressionamento de um botão.

3.2.2 - Máquina de estados

O *firmware* é projetado como uma máquina de estados (modelo de comportamento composto de um número finito de estados, transição entre esses estados e ações), no qual cada estado tem uma ou mais formas de ser acessado. A máquina de estados é representada pela *struct* MEBN (Máquina de Estados de Batalha Naval), que contém apenas uma informação, o seu estado (E_ESCOLHE_MASTER_SLAVE, E_MOVER, E_ATIRAR, E_FIM_PERDEU, E_FIM_GANHOU, E_AGUARDA_ATIRAR_MOVER, E_ESCOLHE_ATIRAR_MOVER, E_AGUARDA_MONTA_TAB, E_MONTA_TAB).

Cada console é um jogador, que por sua vez é representado pela *struct* Jogador. Cada jogador possui dois tabuleiros (o próprio e o do adversário), a posição atual do cursor (linha e coluna), uma variável indicando se ele é o *Master* ou o *Slave*, a quantidade de barcos intactos e a quantidade de barcos atingidos.

O jogo permanece em um loop até que a máquina de estados chegue a um dos estados finais, E_FIM_PERDEU ou E_FIM_GANHOU. Quando alcançado um desses estados, a variável de condição de fim é ativada, e encerra-se o loop.

Primeiramente deve-se escolher qual console será o Jogador 1 (*Master*) e qual será o Jogador 2 (*Slave*).

Em seguida passa-se para outro estado, e deve-se montar o tabuleiro (posicionar os seus barcos). Se o *Master* terminar de montar o tabuleiro antes que o *Slave* ele fica aguardando o *Slave* e transmitindo que está pronto até que o *Slave* receba a informação. Caso o *Slave* termine antes, ele fica aguardando receber a informação de que o *Master* já está pronto. O pacote de transmissão contém apenas o campo função, pois somente se deseja informar o outro jogador que terminou de montar o tabuleiro.

O *Master* começa escolhendo se ele atira ou move um barco e o *Slave* começa aguardando a ação do *Master*.

Caso o jogador escolha mover o barco, primeiramente é verificado se existe algum barco que pode ser movido, pois o barco não pode ser movido para uma posição que já tenha sido atingida. Então se o barco estiver cercado por posições que já tenham sido atingidas, ele não

pode ser movimentado. Se existir algum barco que possa ser movimentado, seleciona-se o barco que deseja ser movimentado e o movimenta. Se não existir, o jogador é automaticamente direcionado para o estado atirar. O pacote de transmissão contém apenas o campo função, pois somente se deseja informar o outro jogador que um barco foi movimentado.

Se a escolha for atirar, é transmitida a posição do tiro e aguarda-se a reposta. Se a resposta for que acertou um barco, é verificado se o jogo já acabou (pelo contador de barcos atingidos). Se acabou, passa para o estado de fim, senão atira-se novamente. Se não acertou, vai para o estado de aguardar o adversário jogar. O pacote de transmissão contém os campos função e dados (dados é um vetor – dados[0] = linha e dados[1] = coluna), o outro jogador recebe o pacote, extrai a posição (linha e coluna) do tiro e verifica se nessa posição possui um barco, e então responde informando se o tiro acertou algum barco ou não.

No estado de aguardar o adversário jogar, permanece-se aguardando receber um pacote de transmissão. Quando o mesmo chega, analisa-se o campo função. Se for mover, comuta-se para o estado de escolher atirar ou mover. Caso seja atirar, aguarda-se novamente por um pacote de transmissão. Ao chegar o pacote, extraem-se os dados do campo dados e verifica se algum barco foi atingido e responde para o adversário se o tiro dele foi certeiro ou não. Se o tiro não foi certeiro, muda-se para o estado escolher atirar ou mover. Se o tiro foi certeiro, verifica se o jogo acabou (por meio do contador de barcos não atingidos). Se acabou, vai para o estado de fim, senão continua-se no mesmo estado (aguardando adversário jogar). O pacote de transmissão de resposta contém os campos função e dados.

Conforme exibida anteriormente, a Figura 3.21 ilustra o diagrama da máquina de estados projetada para o jogo.

3.2.3 - Ferramentas de desenvolvimento

O *firmware* foi programado em linguagem C e desenvolvido utilizando o *IAR Embedded Workbench for MSP430*, que é o ambiente de desenvolvimento integrado (*IDE*) para construção e *debug* de aplicações embarcadas para microcontroladores da família MSP430.

Para efetuar a gravação do *firmware* no microcontrolador, foi utilizado o gravador *USB* da *Texas Instruments*, o *MSP-FET430UIF*.

CAPÍTULO 4 – VALIDAÇÃO E RESULTADOS

Para realizar a validação do projeto, foi pedido para que quinze pessoas testassem o jogo, sendo elas crianças, adultos e jovens. O questionário utilizado nas entrevistas está no Apêndice C.

Inicialmente foi mostrado o "Manual do usuário" a cada jogador, para que entendessem como o jogo funciona. Após lerem o manual e combinarem quem seria o "Jogador 1" e o "Jogador 2", cada jogador iniciava a distribuição dos seus barcos em seu respectivo tabuleiro, e aguardava o adversário posicionar seus barcos.

A partir de entrevistas realizadas com os jogadores, foram observados os seguintes comentários:

- Pontos positivos:
    - Fácil de compreender as regras do jogo;
    - Fácil de jogar;
    - Interface com o usuário atrativa;
    - Boa autonomia de energia.
- Pontos negativos:
    - Console pesado;
    - Jogo pode demorar demais para acabar.

Entre os itens que contribuíram com os pontos positivos ressaltados pelos jogadores, destaca-se a redução do tamanho dos consoles, a implementação dos *displays LCDs* que fornecem um aspecto visual atraente e a elaboração do manual do usuário explicando como o jogo funciona.

Entre os itens que contribuíram com os pontos negativos ressaltados pelos jogadores, destaca-se o peso do *pack* de baterias e a espessura do acrílico. Se houvesse uma bateria a menos e os consoles fossem de plástico ao invés de acrílico, o peso do console seria menor.

Vários testes foram realizados com os consoles para simular como o sistema se comportaria em situações adversas (considerando uma situação normal de 3 metros de distância sem nenhuma barreira entre os consoles, e em ambiente fechado), e verificar a estabilidade e o comportamento do sistema. A Tabela 4.1 mostra o resultado dos testes realizados. Foram feitos testes em distâncias diferentes - sem e com barreiras (paredes de alvenaria) – e em ambientes diferentes (ambientes fechados e à céu aberto).

TABELA 4.1 – Testes realizados

Ambiente	Distância (m)	Resultado
Fechado, sem barreira	3	Sinal de transmissão cheio
Fechado, sem barreira	9	Sinal de transmissão cheio
Fechado, sem barreira	15	Sinal de transmissão cheio
Fechado, sem barreira	21	Metade do sinal de transmissão
Fechado, sem barreira	27	Metade do sinal de transmissão
Fechado, com barreira	3	Sinal de transmissão cheio
Fechado, com barreira	9	Sinal de transmissão cheio
Fechado, com barreira	15	Metade do sinal de transmissão
Fechado, com barreira	21	Sinal de transmissão fraco
Fechado, com barreira	27	Sem sinal de transmissão
Aberto, sem barreira	5	Sinal de transmissão cheio
Aberto, sem barreira	15	Sinal de transmissão cheio
Aberto, sem barreira	25	Metade do sinal de transmissão
Aberto, sem barreira	35	Metade do sinal de transmissão
Aberto, sem barreira	45	Sinal de transmissão fraco
Aberto, sem barreira	55	Sinal de transmissão fraco
Aberto, sem barreira	65	Sem sinal de transmissão
Aberto, sem barreira	75	Sem sinal de transmissão
Aberto, sem barreira	85	Sem sinal de transmissão
Aberto, sem barreira	95	Sem sinal de transmissão

Antes da escolha do CC1100, foi testado o modelo TRF-2.4G *Transceiver* da *Laipac Tech*, também transceptor de radiofreqüência (para aplicações de até 2,4 GHz), porém o quesito confiabilidade (baixa taxa de erros durante a transmissão) não foi satisfatoriamente atendido, tornando o seu uso impraticável para este projeto.

CAPÍTULO 5 - CONCLUSÃO

Este projeto teve como objetivo principal colocar em funcionamento as principais modificações citadas por SANTOS (2006). Dessa forma, as metas alcançadas no desenvolvimento deste projeto foram:

- Incorporação do *display* gráfico *LCD* ao projeto;
- Aumento da confiabilidade (baixa taxa de erros) e da distância para transmissão de dados *wireless*;
- Aumento da autonomia de energia do sistema;
- Desenvolvimento do módulo dinâmico do jogo;
- Redução do tamanho dos consoles.

Pode-se concluir que, com a implementação das melhoras sugeridas por SANTOS (2006), o projeto tornou-se mais interessante para seus jogadores, principalmente quando se trata do aspecto visual, pois com a implementação do *LCD*, do motor e do botão, possibilitou-se melhor interação do jogador com o sistema. Também ficou clara a melhora na estabilidade do sistema, pelo fato do aumento substancial da autonomia de energia e do melhoramento na comunicação *wireless* (com o desenvolvimento de um protocolo de comunicação mais robusto).

Como sugestões de possíveis melhorias para trabalhos futuros, poderiam ser consideradas:

- A implementação de barcos maiores, com mais de uma posição (como porta-aviões, navios, cruzadores, submarinos e destróieres);
- Adição de um circuito carregador de baterias;
- Usar uma tela *touchscreen*;
- Substituir os componentes da fonte (atualmente ela é do tipo linear, com componentes que, para reduzir a tensão, geram calor). O ideal seria uma fonte chaveada. Com este tipo de alimentação, o *pack* de baterias poderia ser reduzido para apenas duas baterias;
- Projetar uma placa com componentes *SMD* (reduziria o tamanho e a espessura da placa);
- Elaborar um sistema de som (tipo *buzzer*, similar ao dos PCs);
- Utilizar protocolos de comunicação conhecidos, por já terem sido testados à exaustão e serem estáveis, como o *Bluetooth* e *Zigbee*;
- Reduzir ainda mais o tamanho e o peso dos consoles.

CAPÍTULO 6 - REFERÊNCIAS

ANTIGOS, J. **Batalha Naval**. Disponível em: <http://www.jogos.antigos.nom.br/bnaval.asp>. Acesso em: março de 2008.

COLLINGS, P. J.; HIRD, M. **Introduction to Liquid Crystals: Chemistry and Physics**. Editora *Taylor and Francis*. *London*. 1997.

CORSO, F. R. **Jogo de batalha naval embarcado**. Curitiba, 2005. Monografia (graduação). Universidade Positivo.

DAILYWIRELESS.ORG. **Glossary.** Disponível em: <http://www.dailywireless.org/sponsors/>. Acesso em: março de 2008.

GEORGE W. G.; STEPHEN M. K.: **Liquid crystals for twisted nematic display devices**. Editora J. Mater. 1999.

FULLERTON, T.; SWAIN, C.; HOFFMAN, S.: **Game Design Workshop. Designing, prototyping, and playtesting games**. Editora CMP BOOKS. 2004.

IEEE. **The history of Liquid-Crystal Displays**. Disponível em: <http://www.ieee.org/portal/cms_docs_iportals/iportals/aboutus/history_center/LCD-History.pdf>. Acesso em: março de 2008.

INSTRUMENTS, T. **CC1100.** Disponível em: <http://focus.ti.com/docs/prod/folders/print/cc1100.html>. Acesso em: março de 2008.

INSTRUMENTS, T. **MSP430F169.** Disponível em: <http://focus.ti.com/docs/prod/folders/print/msp430f169.html>. Acesso em: março de 2008.

JOGOS, U. **Alta demanda atrapalha planos.** Disponível em: <http://jogos.uol.com.br/wii/ultnot/2007/12/20/ult4097u1067.jhtm>. Acesso em: março de 2008.

MNEINA, OSAMA. **What is a Microcontroller?** Disponível em: <http://o.mneina.googlepages.com/what_is_microcontroller.htm>. Acesso em: março de 2008.

NAGY, C.: **Embedded Systems Design Using the TI MSP430 Series**. Editora Newnes. 2003.

NOBEL PRIZE. **History and Properties of Liquid Crystals**. Disponível em: <http://nobelprize.org/educational_games/physics/liquid_crystals/history/>. Acesso em: março de 2008.

PEREIRA, F.: **Microcontroladores MSP430: Teoria e Prática.** Editora Érica. 2005.

PHAM, A. **Video-game design graduates find themselves in high demand.** Disponível em: <http://seattletimes.nwsource.com/html/businesstechnology/2008340466_gamejobs02.html>. Acesso em: novembro de 2008.

SANTOS, G. A. **Jogo de batalha naval *wireless* embarcado**. Curitiba, 2006. Monografia (graduação). Universidade Positivo.

SEMICONDUCTOR, F. **LM7805.** Disponível em: <http://www.fairchildsemi.com/pf/LM/LM7805.html>. Acesso em: março de 2008.

SEMICONDUCTOR, N. **LM317.** Disponível em: <http://www.national.com/mpf/LM/LM317.html>. Acesso em: março de 2008.

VANSICLE, T. **Programming Microcontrollers in C**. Editora Newnes. 2000.

WIKIPEDIA. **Battleship (game).** Disponível em: <http://en.wikipedia.org/wiki/Battleship_game>. Acesso em: março de 2008.