UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTOS ACADÊMICOS DE ELETRÔNICA E MECÂNICA
CURSO SUPERIOR DE TECNOLOGIA EM MECATRÔNICA INDUSTRIAL

ANDRÉ LUIS RADIGONDA

DESENVOLVIMENTO DE UM PAINEL MODULAR DE LED

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2014

ANDRÉ LUIS RADIGONDA

DESENVOLVIMENTO DE UM PAINEL MODULAR DE LED

Trabalho de Conclusão de Curso apresentado à disciplina de Trabalho de Diplomação, como requisito parcial para obtenção de grau de Tecnólogo em Mecatrônica Industrial dos Departamentos Acadêmicos de Eletrônica (DAELN) e Mecânica (DAMEC) da Universidade Tecnológica Federal do Paraná – UTFPR.

Orientador: Prof. M. Sc. Juliano Mourão Vieira.

CURITIBA
2014

ANDRÉ LUIS RADIGONDA

DESENVOLVIMENTO DE UM PAINEL MODULAR DE LED

Este trabalho de conclusão de curso foi apresentado no dia 19 de dezembro de 2013, como requisito parcial para obtenção do título de Tecnólogo em Mecatrônica Industrial, outorgado pela Universidade Tecnológica Federal do Paraná. O aluno foi arguído pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____

Prof. Milton Luiz Polli
Coordenador de Curso
Departamento Acadêmico de Mecânica

_____

Prof. Sergio Moribe
Responsável pela Atividade de Trabalho de Conclusão de Curso
Departamento Acadêmico de Eletrônica

BANCA EXAMINADORA

_____

Prof. Me. Gabriel Kovalhuk

_____

Prof. Me. Juliano Mourão Vieira
(Orientador)

_____

Prof. Dr. Luís Paulo Laus

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

AGRADECIMENTOS

Agradeço ao Professor Juliano Mourão Vieira pela orientação e aos colegas Marcos Likio Nogawa e Fernanda Dorneles Malaquias pela grande ajuda em momentos importantes do desenvolvimento deste trabalho.

RESUMO

RADIGONDA, André L. **Desenvolvimento de um painel modular de led**. 2014. 62 f. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Mecatrônica Industrial) – Departamentos Acadêmicos de Eletrônica (DAELN) e Mecânica (DAMEC), Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

Esse trabalho consiste no desenvolvimento de um painel de LED (diodo emissor de luz, do inglês *light emitting diode*) modular e interativo. São apresentados circuitos eletrônicos de fornecimento de energia, comunicação por infravermelho e controle de matriz de LED. Além disso, também fez parte deste trabalho o desenvolvimento de um software de controle e simulação de um jogo, o qual foi aplicado aos módulos do painel desenvolvido.

Palavras-chave: Matriz de LED. Comunicação por Infravermelho. Painel de LED.

ABSTRACT

RADIGONDA, André L. **Development of modular led panel**. 2014. 62 f. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Mecatrônica Industrial) – Departamentos Acadêmicos de Eletrônica (DAELN) e Mecânica (DAMEC), Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

This work consists of a modular and interactive LED (light emitting diode) panel development. Electronics circuits for power supply, infrared communication and LED matrix control were developed. Furthermore, it was also developed software for a game control and simulation, which was used with the developed panel modules.

Keywords: LED Matrix. Infrared Communication. LED Panel.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

“Em diferentes graus, todos os animais brincam, exploram, movimentam-se sem motivo aparente. Mas somente alguns conservam na idade adulta a capacidade juvenil de brincar, como certos pássaros (o corvo, por exemplo), os roedores, os carnívoros superiores, os primatas, e, evidentemente, o homem. Note-se que as espécies verdadeiramente capazes de brincar são também as mais ‘cosmopolitas’, que souberam se adaptar aos climas mais diversos e aumentaram, com isso, suas possibilidades de sobreviver.” (ANTUNES, 1998)

Há muito tempo somos estimulados a levar um estilo de vida sem muito esforço mental. É comum vermos, por exemplo, propagandas de métodos de ensino rápidos, onde é vendida a idéia de aprender sem esforço. Porém, deve-se notar que, por um lado, é bastante duvidoso que metas concretas de aprendizado possam ser alcançadas gratuitamente, e, por outro, a conquista de algo através de esforço traz satisfação ao indivíduo. (BATLLORI, 2004).

Treinar, resolver enigmas, pensar, solucionar problemas lógicos e estimular o cérebro pode ser uma atividade muito prazerosa. Para isso, existem diferentes tipos de jogos que estimulam de diversas maneiras a nossa capacidade intelectual e motora.

Dentro desse contexto, atualmente vemos com frequência o lançamento de novas tecnologias de jogos eletrônicos que visam estimular cada vez mais os diversos sentidos de quem joga. Tecnologias de comunicação sem fio, materiais sensíveis ao toque e detecção de movimento são exemplos de como o desenvolvimento da eletrônica e da informática têm sido utilizado em favor da criação de jogos mais atrativos, divertidos e que estimulam o cérebro dos jogadores.

Com isso, esse projeto visa desenvolver uma nova tecnologia de jogos: um painel eletrônico modular interativo. Composto por vários módulos que se comunicam entre si, será oferecida uma nova forma de interagir com os jogos, a qual será através da movimentação dos próprios módulos que compõem o painel.

1.1 PROBLEMA

O painel proposto é dividido em módulos independentes, os quais se comunicaram entre si por radiação infravermelha. Cada módulo é capaz de detectar se há ou não outros módulos ao seu redor, num total de quatro, um em cada lado. Dessa forma, cria-se uma rede de módulos independentes que forma o painel eletrônico final, como ilustrado na figura 1.

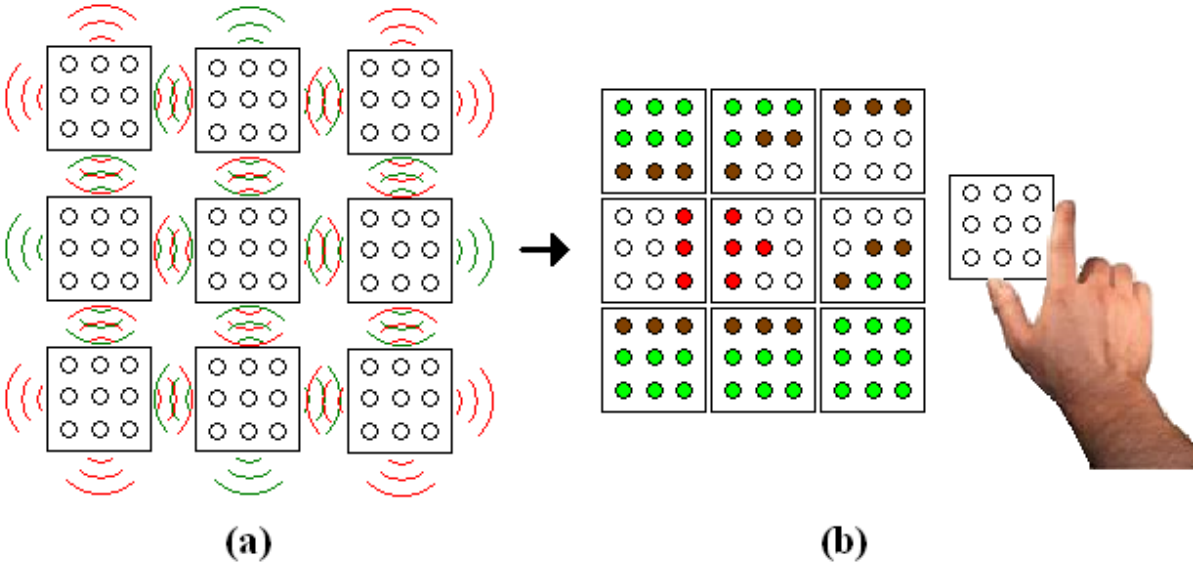


Figura 1 – (a) Ilustração da comunicação entre os módulos. (b) Ilustração da formação de um painel e da manipulação de um módulo para o controle do jogo.
Fonte: Autoria própria.

A interatividade com o painel é exercida através de um módulo extra construído especificamente para atuar como um módulo de controle. Uma vez que, através do software desenvolvido, é possível saber a disposição dos módulos em cada momento, é também possível detectar o deslocamento dos módulos. Isso possibilita a criação de jogos interativos.

Primeiramente, tendo em vista que essa comunicação é feita através de radiação infravermelha e que em um mesmo módulo há mais de um transceptor de infravermelho, o primeiro desafio é implementá-los de uma forma que um transceptor

não interfira na comunicação de outro. Essa questão é solucionada com a emissão de sinais de infravermelho fracos, os quais alcançam somente alguns centímetros.

Tendo solucionado o impasse da comunicação por infravermelho, tem-se como próximo desafio a implementação de um software de controle dos módulos capazes de identificar os sinais de módulos distintos.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um painel eletrônico modular interativo.

1.2.2 Objetivos Específicos

a. Estudar as características do microprocessador Microchip PIC®;
b. Estudar as características do formato de comunicação serial USART;
c. Projetar o algoritmo de comunicação entre os módulos do painel;
d. Elaborar o programa de processamento e controle;
e. Estudar os componentes eletrônicos que serão utilizados nos módulos do painel;
f. Testar os componentes eletrônicos que serão utilizados nos módulos do painel;
g. Projetar os circuitos eletrônicos dos módulos do painel;
h. Montar o módulo de processamento central;
i. Montar os módulos do painel;
j. Fazer a comunicação entre os módulos do painel;
k. Documentar o projeto de acordo com as normas da UTFPR.

1.3 JUSTIFICATIVA

O LED (do inglês, *light emitting diode*) está sendo cada vez mais utilizado pela indústria por oferecer algumas vantagens em relação a outras tecnologias de geração de luz. Hoje o vemos sendo utilizado em produtos como automóveis, celulares, semáforos, placas de trânsito e aparelhos de televisão. Neste último, atualmente é visto o lançamento de novas tecnologias de LED cada vez mais atraentes aos usuários, tanto de televisores para uso caseiro como em grandes painéis comerciais.

Esses televisores e painéis têm uma grande área de aplicação no mercado atual. Eles podem ser utilizados em áreas como o marketing, comércio, eventos públicos e entretenimento. Em algumas áreas em específico, a união de novas tecnologias com as telas e painéis eletrônicos vem se destacando por proporcionar aos usuários novas interfaces e formas de interação com o mundo digital, como é o caso das telas sensíveis ao toque e o uso de acelerômetros em aparelhos portáteis. Essas novas interfaces de controle e interação vêm sendo bem aceitas pelos usuários, uma vez que um grande número de pessoas está adquirindo esses produtos.

De acordo com o site de uma associação de software de entretenimento dos Estados Unidos (ESA), três quartos das famílias estadunidenses jogam jogos eletrônicos. Isso reflete o crescimento do setor que, também de acordo com o site, está entre os que mais crescem no país, onde, em 2010, os consumidores gastaram em torno de 25,1 bilhões de dólares em jogos, consoles e acessórios. (THE ENTERTAINMENT SOFTWARE ASSOCIATION, 2011)

Além disso, o desenvolvimento da eletrônica para fins de entretenimento pode impulsionar o surgimento de novas tecnologias de comunicação, materiais eletrônicos, software, e etc. Mais especificamente, essas novas tecnologias podem ajudar ainda no desenvolvimento de crianças e adultos através de jogos eletrônicos interativos que incentivam o exercício mental e motor.

Para tanto, esse projeto visa desenvolver uma nova opção de dispositivo de jogos eletrônicos, a qual oferecerá uma nova interface de controle e interatividade. O desenvolvimento dessa tecnologia terá como objetivo inicial construir um console de jogos caseiros. Tal produto será voltado para um público bastante diverso, de faixa

etária de 13 a 35 anos, dependendo somente do nível de dificuldade dos jogos a serem implementados. Porém, essa tecnologia também poderá ser utilizada em eventos publicitários, escolas e até mesmo como base para o desenvolvimento de novas tecnologias de painéis industriais.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 DIODO EMISSOR DE LUZ (LED)

O LED (do inglês *Light Emitting Diode*), como o próprio nome diz, é um diodo capaz de emitir luz quando acionado. Ele é um dispositivo semicondutor de estado sólido de junções p-n, como mostrado na figura 2. Devido aos materiais utilizados em sua construção, o LED é um dispositivo que oferece muitas vantagens em relação a outros dispositivos de geração de luz, como: elevado tempo de vida útil, larga faixa de temperatura de operação, baixa tensão e corrente para a produção de luz, alta velocidade de ativação e desativação, entre outras vantagens. (GAGE, 1977).

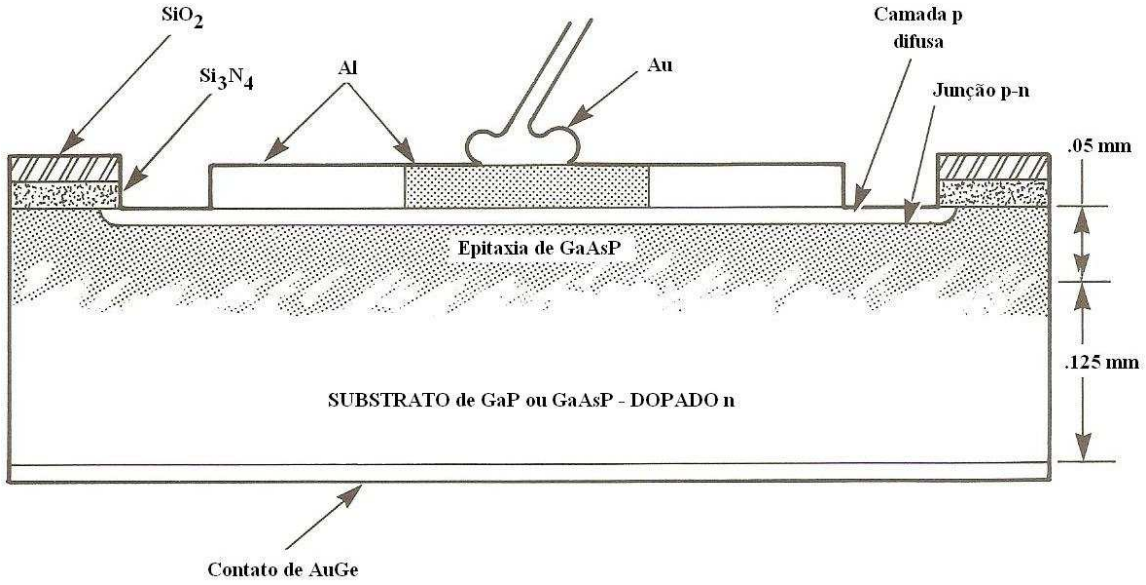


Figura 2 - Seção do corte de um LED.

Fonte: Adaptado de GAGE et al.

2.2 MICROCONTROLADOR

O microcontrolador pode ser entendido como um componente eletrônico programável capaz de executar operações lógicas. Sua estrutura contém uma Unidade Lógica Aritmética (ULA), memórias, pinos de entrada e saída de sinais entre outros componentes. Com isso, é possível programá-lo para controlar diversos dispositivos como LEDs, *displays*, relés e sensores. (SOUZA, 2005).

2.3 MICROCONTROLADOR PIC 16F873A

O microcontrolador PIC16F873A é produzido pela Microchip Technology, possui 28 pinos, como mostrado na figura 3, e oferece as seguintes características:

- Dois *timers*/contadores de 8 bits e um timer/contador de 16 *bits*;
- Dois módulos para geração de sinal PWM (*Pulse Width Modulation*);
- Módulo de comunicação serial;
- USART (*Universal Synchronous Asynchronous ReceiverTransmitter*);
- Módulo de conversão de sinal analógico/digital;
- Módulo de comparação de sinal analógico;
- Memória *Flash*/EEPROM de alta velocidade de gravação e baixo consumo de energia;
- Opera com tensões de alimentação variando entre 2.0V e 5.5V;
- Oferece baixo consumo de energia.

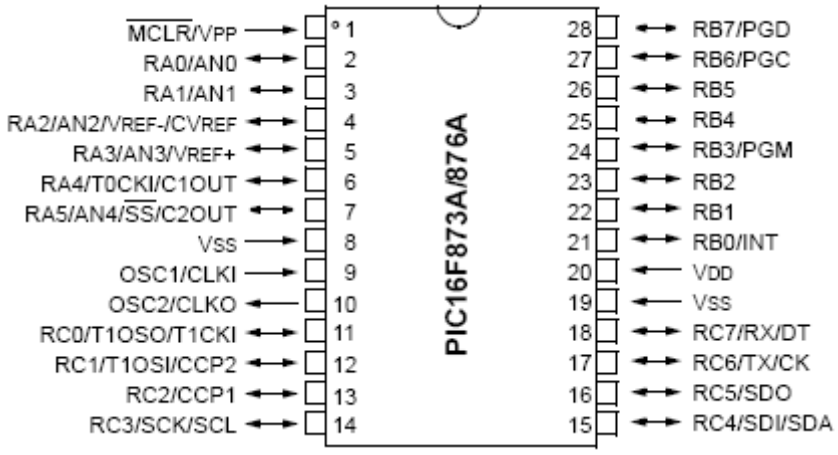


Figura 3 - Pinagem do microcontrolador PIC 16F873A.
Fonte: MICROCHIP TECHNOLOGY, 2003.

2.4 INFRAVERMELHO - IrDA

Nomeado de acordo com a *Infrared Data Association*, o IrDA é um sistema de comunicação sem fio disponível em dispositivos eletrônicos, sendo um dos padrões mais populares do mundo. (DORNAN, 2001).

Nesse sistema, a comunicação é feita por transmissões de infravermelho e, por isso, é limitada a uma área pequena. Na maioria dos casos, essa tecnologia também exige que o dispositivo transmissor do sinal esteja direcionado para o receptor para que a comunicação ocorra com sucesso. (COMER, 2007).

Há vários padrões de transmissão serial de dados por infravermelho definidos pela *Infrared Data Association* (IrDA), os quais incluem taxas de transmissão de 115,2 kbps, 0,576 Mbps, 1,152 Mbps e 4 Mbps.

2.5 CIRCUITO INTEGRADO TIR1000

O circuito integrado TIR1000 é produzido pela Texas Instruments. Suas principais características são (TEXAS INSTRUMENTS, 1999):

- Interconecta o transmissor de infravermelho ao dispositivo UART (*Universal Asynchronous Receiver Transmitter*);
- Compatível com IrDA (*Infrared Data Association*) e HPSIR (*Hewlett Packard Serial Infrared*);
- Oferece taxas de transmissão de dados de 1200bps a 115kbps;
- Opera com tensões de 2,7V a 5,5V;
- Oferece interface simples com UART;
- Decodifica pulsos positivos ou negativos;

Sua função é codificar e decodificar sinais para a transmissão serial por sinal de infravermelho. Para isso, ele atua na interface entre um emissor/receptor de sinal serial (UART) e um dispositivo de emissão/recepção de infravermelho, como mostra a figura 4.

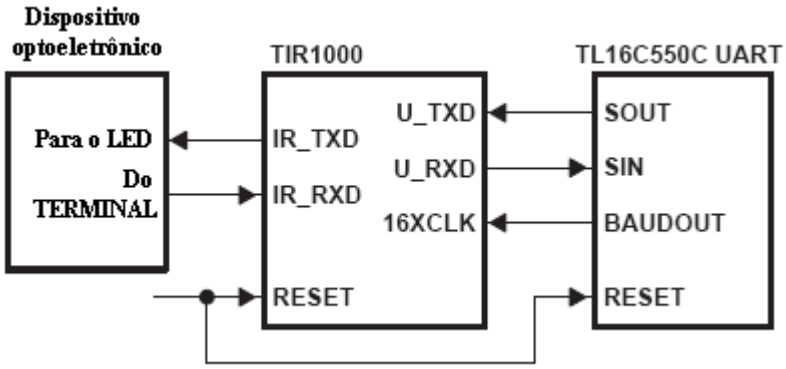


Figura 4 - Aplicação típica do circuito integrado TIR1000.
Fonte: Adaptado de TEXAS INSTRUMENTS, 1999.

O TIR1000 segue somente o protocolo definido para taxas de transmissão de 115kbps. Nele a transmissão de dados é feita através de um *bit* inicial igual a 0 seguido do pacote de dados formado por dezesseis *bits* (*word*), o qual contém a informação a ser transmitida, seguido de um *bit* final de parada igual a 1. O *clock* (ciclo de máquina) utilizado pelo circuito é dezesseis vezes maior que o *clock* utilizado pelo dispositivo UART. Assim, o período de codificação de um *bit* é igual a 16 ciclos de *clock*. A codificação de um *bit* igual a 0 consiste no envio de um pulso de 3 ciclos de máquina durante esse período. Já na codificação de um *bit* igual a 1

não é enviado nenhum pulso durante todo o ciclo de 16 ciclos de máquina, como mostra a figura 5. (TEXAS INSTRUMENTS, 1999)

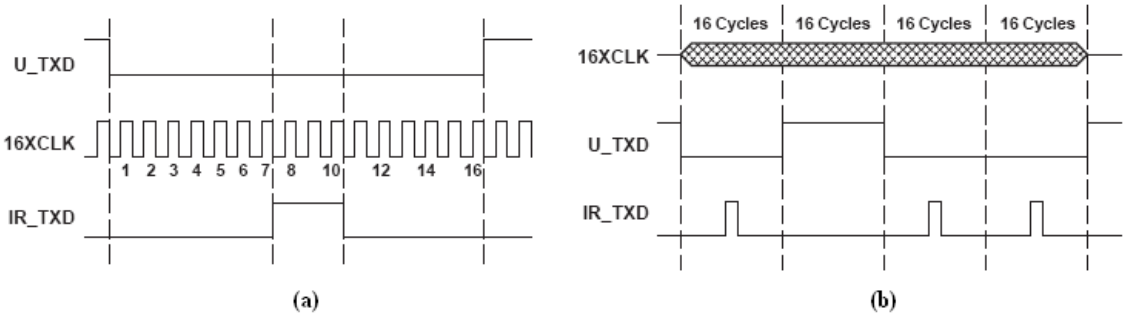


Figura 5 - Esquema de codificação IrDA-SIR (*Serial Infrared*). (a) Codificação detalhada de um bit. (b) Visão macro da codificação de 4 bits.
Fonte: TEXAS INSTRUMENTS, 1999.

De forma oposta, o esquema de decodificação analisa o sinal de infravermelho para gerar o sinal serial para o dispositivo UART. Após o reinício do circuito, a saída U_RXD (figura 6) está em nível alto. Logo após detectar uma borda de descida em IR_RXD, gerada pelo dispositivo optoeletrônico, U_RXD é levado a nível baixo e assim permanece por 16 ciclos de máquina (16XCLK). U_RXD retorna então ao nível lógico alto, em cumprimento ao padrão estabelecido pela IrDA. U_RXD permanecerá em nível lógico alto enquanto não houver outra borda de descida detectada em IR_RXD, como mostrado na figura 6. (TEXAS INSTRUMENTS, 1999)

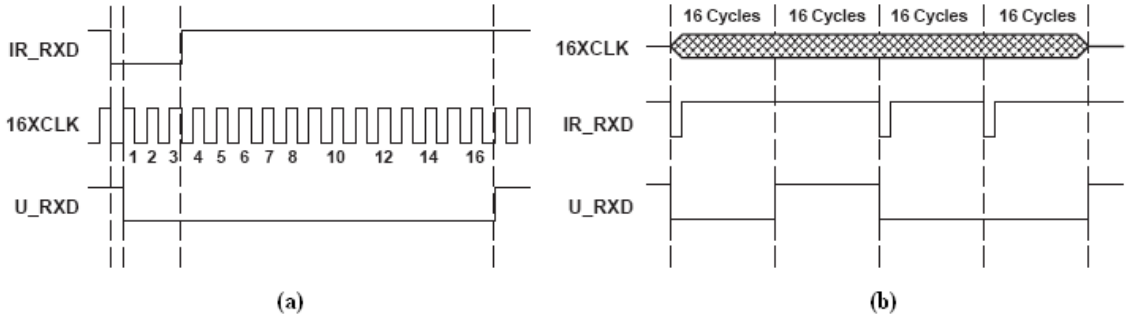


Figura 6 - Esquema de decodificação SIR-Irda. (a) Decodificação detalhada de um *bit*. (b) Visão macro da decodificação de 4 *bits*.
Fonte: TEXAS INSTRUMENTS, 1999.

2.6 REGISTRADORES DE DESLOCAMENTO

O registrador de deslocamento é um circuito eletrônico formado por *flip-flops*, onde as saídas de cada um são ligadas nas entradas do bloco seguinte, sendo que o primeiro tem suas entradas ligadas na forma de um *flip-flop* do tipo D, como ilustrado na figura 7. (IDOETA, 2008)

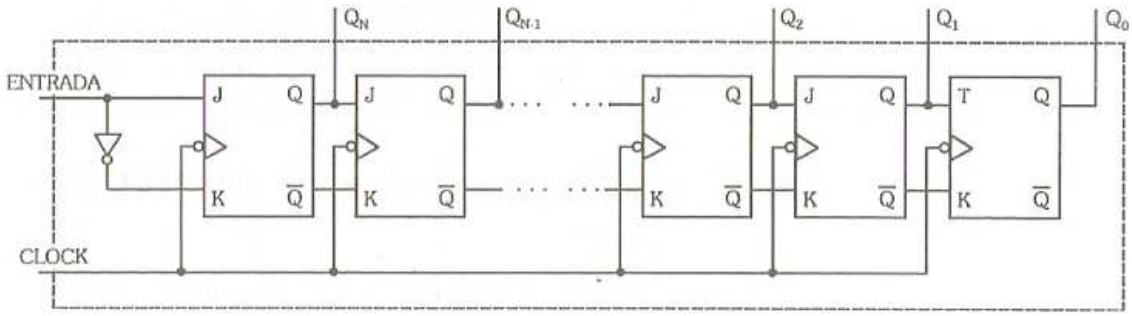


Figura 7 – Registrador de deslocamento generalizado.
Fonte: IDOETA, 2008. p. 251.

Dessa forma, acionando o circuito com o sinal de *clock* é possível deslocar de forma sequencial um valor de entrada pelas saídas Q dos respectivos registradores.

2.7 CIRCUITO INTEGRADO SN74HC164

Fabricado pela Texas Instruments, o circuito integrado SN74HC164 é um registrador de deslocamento de 8 *bits* formado por *flip-flops* do tipo D. O circuito contém duas entradas de sinal serial ligadas em uma porta lógica “E”, da qual se tem o sinal a ser transmitido para o primeiro *flip-flop*, que é então transferido para o próximo registrador ao próximo pulso de *clock*. Segue na figura 8 o diagrama do circuito.

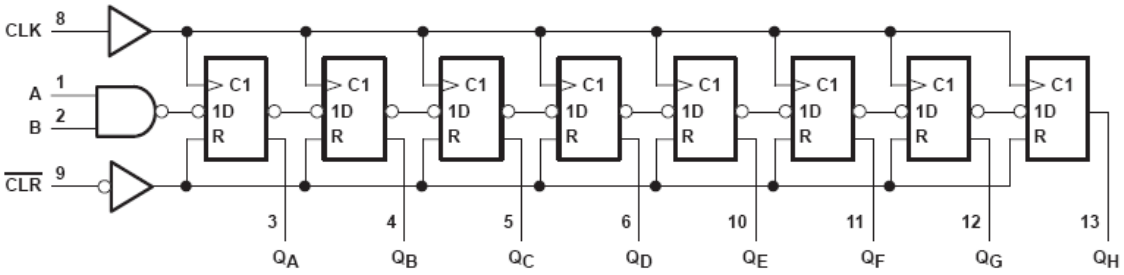


Figura 8 - Esquemático do circuito integrado SN74HC164.

Fonte: TEXAS INSTRUMENTS, 1997.

2.8 CIRCUITO INTEGRADO ZHX1810

O circuito integrado ZHX1810, fabricado pela empresa Zilog, tem a função de atuar como um transceptor de infravermelho. Suas principais características são:

- Cumpre com as especificações da IrDA;
- Opera com tensões de alimentação de 2.4V a 5.5V;
- Opera com baixa corrente de leitura: 90µA em 3V;
- Possui um encapsulamento compacto: 9.1mm x 3.8mm x 2.73mm;
- Necessita de somente dois componentes externos;
- Trabalha em uma margem de temperatura de -30ºC a +85ºC.
- Cumpre com as especificações de segurança para os olhos IEC 825-1 Classe 1.

O circuito possui 6 pinos de controle, como mostrado na figura 9, os quais têm suas funções explicadas na tabela 1.

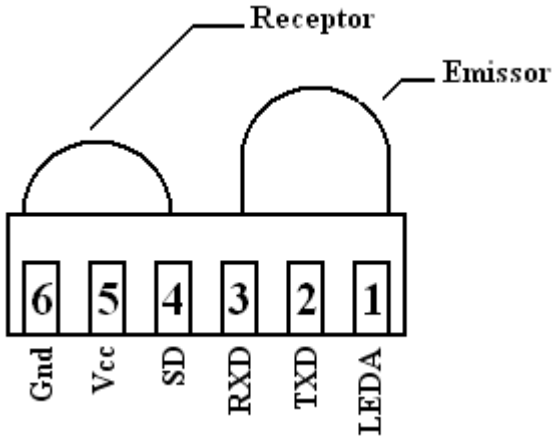


Figura 9 - Ilustração dos pinos do circuito integrado ZHX1810.
Fonte: Adaptado de ZILOG, 2010.

Tabela 1 – Detalhamento da função dos pinos do circuito integrado ZHX1810

Pino	Nome	Função
1	LEDA	Anodo do LED transmissor de infravermelho
2	TXD	Entrada para transmissão de sinal serial
3	RXD	Saída do receptor de sinal serial
4	SD	Habilita modo *shudown*
5	Vcc	Alimentação
6	Gnd	Terra

Fonte: Adaptado de ZILOG, 2010.

2.9 CONVERSOR DE TENSÃO DC/DC

Podemos definir os conversores DC/DC como sendo circuitos que transformam um nível de tensão em outro. Esses circuitos podem ser formados de várias formas diferentes, onde a maioria deriva de circuitos básicos, os quais possuem duas chave: uma ativa (transistor) e outra passiva (diodo), como ilustrado na figura 10. Sobre o transístor atua um sinal de controle que determina a característica de funcionamento do circuito. (MELLO, 1996)

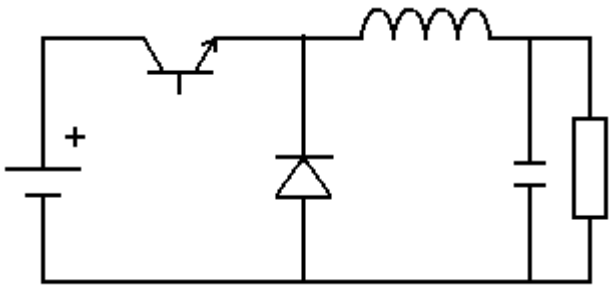


Figura 10 - Exemplo de conversor com duas chaves.
Fonte: Adaptado de MELLO, 1996.

2.10 CIRCUITO INTEGRADO MAX1674

Fabricado pela Maxim Integrated, o circuito integrado MAX1674 é um conversor de tensão DC-DC. Algumas de suas principais características são (MAXIM, Rev 3):

- Alta eficiência (94% de eficiência a uma corrente de saída de 200mA);
- Retificador síncrono interno;
- Detector de bateria com baixa carga;
- Baixo ruído.

A figura 11 ilustra a vista superior do circuito integrado e nomeia os pinos do mesmo, os quais têm sua funcionalidade descrita na tabela 2.

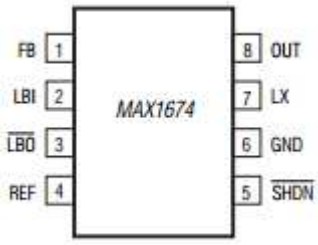


Figura 11 - Ilustração da vista superior do circuito integrado MAX1674.
Fonte: Adaptado de MAXIM, Rev 3.

Tabela 2 – Funcionalidade dos pinos do circuito MAX1674.

Nome do Pino	Função
FB	Entrada Dual-Mode™ de realimentação. Conectar ao GND para +5.0V de tensão de saída. Conectar ao OUT para +3.3V de tensão de saída. Usar a rede de resistores para setar a tensão de saída de +2.0V a +5.5V.
LBI	Entrada do comparador para bateria com baixa carga.
LBO	Saída Open-Drain do comparador de bateria com baixa carga. Conectar LBO ao OUT através de um resistor de 100kΩ. O sinal de saída fica em nível baixo quando VLBI é <1.3V. LBO tem alta impedância quando o circuito está em modo de shutdown.
REF	1.3V de voltagem de referência.
SHDN	Entrada para o modo de shutdown (circuito inativo). Manter em alto nível lógico (>80% do VOUT) para o modo de operação. Manter em nível baixo (<20% do VOUT) para ativar o circuito em modo de shutdown. Conecte ao OUT para operação normal.
GND	Terra
LX	Dreno de Canal-N e Canal-P para MOSFET de potência
OUT	Saída de energia.

Fonte: Adaptado de MAXIM, Rev 3.

Detalhes do circuito eletrônico que compõe o MAX1674 podem ser vistos na figura 12.

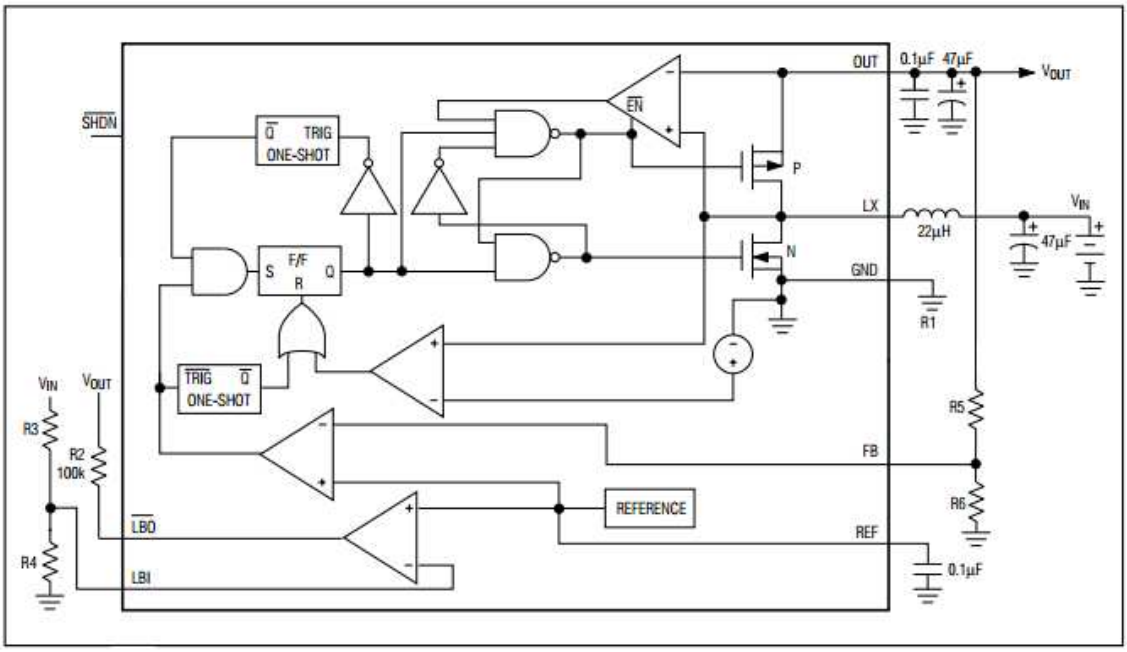


Figura 12 - Circuito integrado MAX1674 detalhado.

Fonte: Adaptado de MAXIM, Rev 3.

2.11 O JOGO SNAKE

O jogo Snake consiste basicamente em guiar uma cobra crescente pela tela o máximo que o jogador conseguir até que ela se choque com o seu próprio corpo ou um obstáculo. A versão original do jogo, chamada “Blockade”, foi desenvolvida para fliperama pela desenvolvedora de jogos Gremlin em 1976. Em 1997 o jogo foi incluído no aparelho celular Nokia 6110. (NOVAK, 2012)

Hoje, Snake é considerado um dos grandes precursores dos jogos de plataforma móvel. A figura 13 mostra a tela de um celular Nokia com o jogo Snake em andamento.

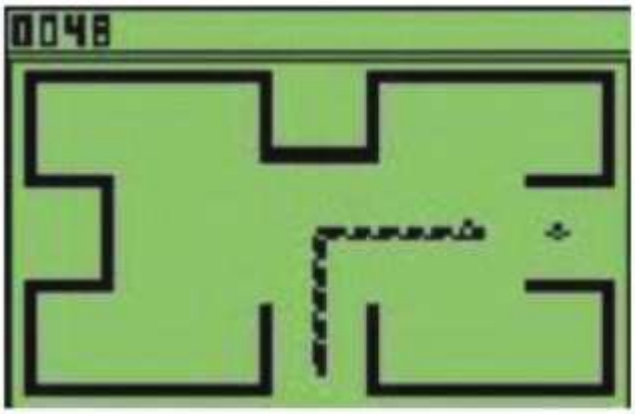


Figura 13 – Tela de um celular Nokia com o jogo Snake em andamento.
Fonte: NOVAK, 2012.

3 DESENVOLVIMENTO

O desenvolvimento desse projeto pode ser dividido em três partes: eletrônica, estrutura mecânica e programação. Cada uma delas é explicada a seguir.

3.1 ELETRÔNICA

Para iniciar o desenvolvimento da parte eletrônica foi necessário testar cada componente a partir de circuitos simples.

3.1.1 Testes Iniciais

O primeiro teste foi feito com o microcontrolador PIC 16F873A, fabricado pela Microchip®. Com ele foi utilizado o compilador CCS, mostrado na figura 14, e o software ICPROG, mostrado na figura 15, em conjunto com o programador de PIC EDUTECHKITS (figura 16) para a transferência do código ao microcontrolador.

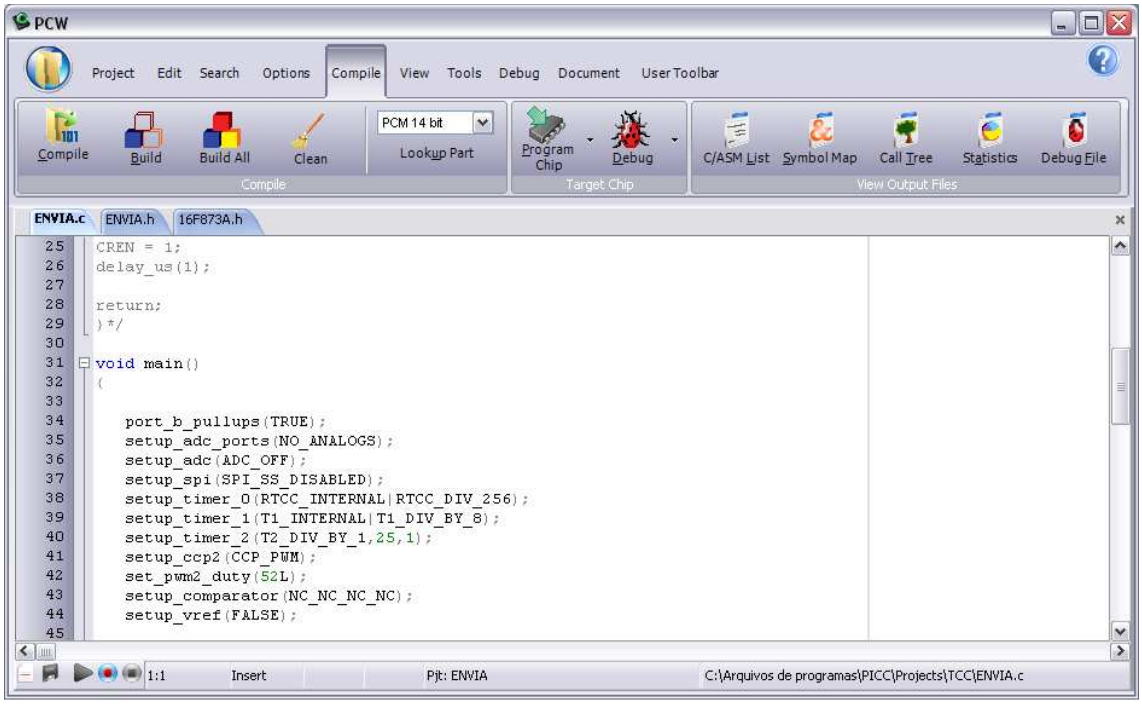


Figura 14 - Exemplo de tela do compilador CCS.

Fonte: Autoria própria.

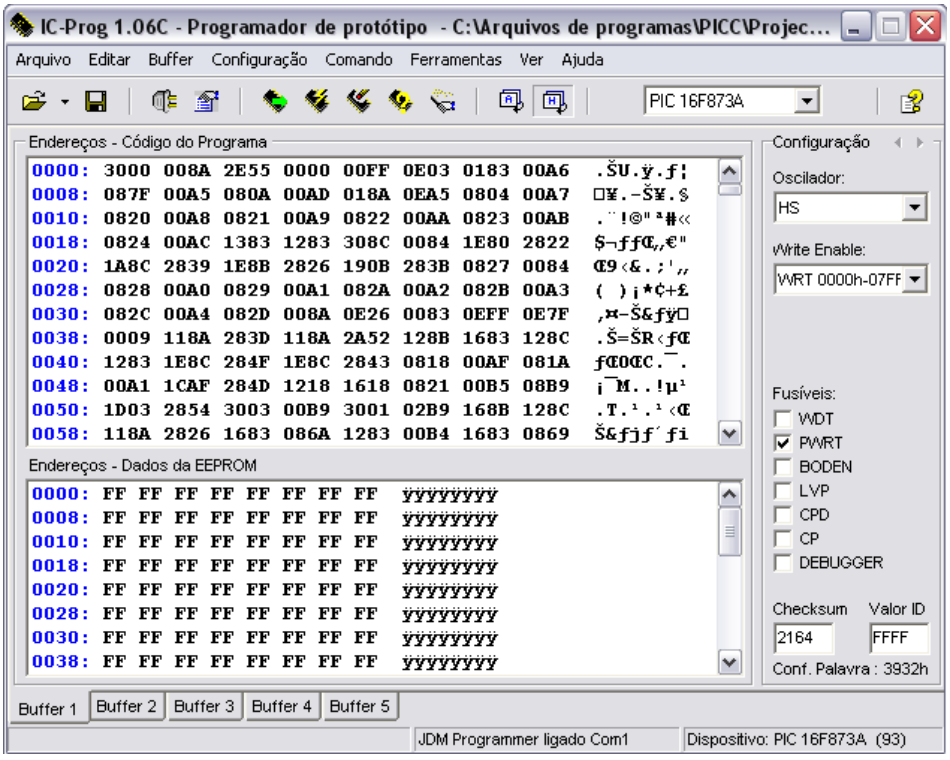


Figura 15 - Exemplo de tela inicial do software ICPROG.

Fonte: Autoria própria.

Figura 16 - Gravador de PIC PIC EDUTECHKITS.
Fonte: Autoria própria.

Os primeiros testes se deram com sucesso. Eles se basearam na ativação e controle da porta B do microcontrolador. O circuito utilizado está ilustrado na figura 17.

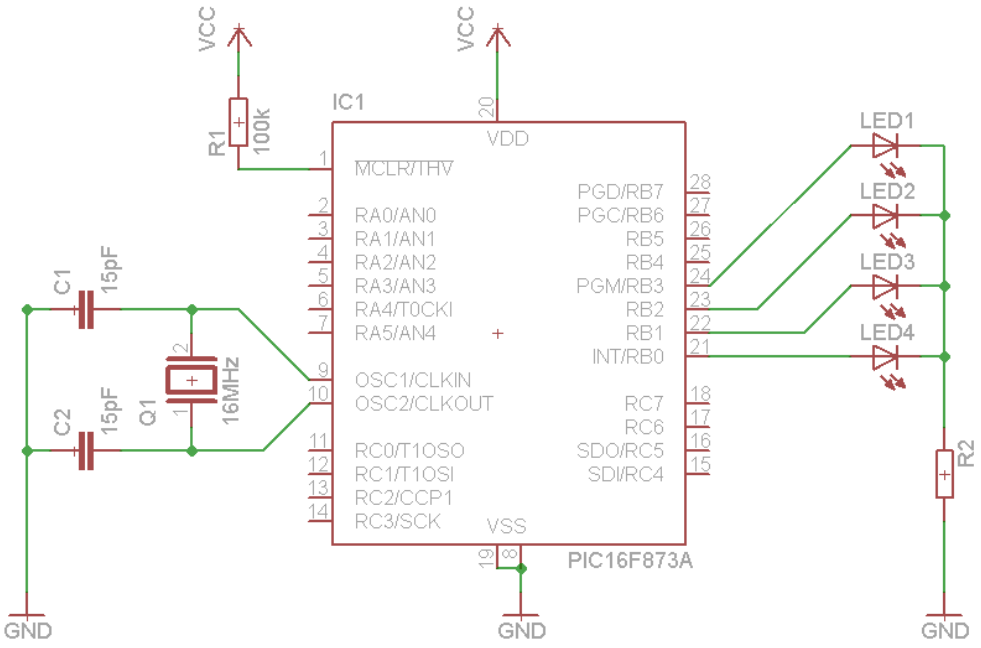


Figura 17 - Circuito utilizado para os testes iniciais do microcontrolador.
Fonte: Autoria própria.

A partir de então foram iniciados os testes de comunicação serial assíncrona entre dois microcontroladores. Para isso, além de utilizar o circuito da figura 17 como base para cada PIC, foram interligadas as portas RC6 e RC7 referentes à saída TX e entrada RX do módulo interno de comunicação serial dos microcontroladores. Os testes se deram sem grandes problemas. Foi possível transferir um caractere para outro PIC, que o recebeu adequadamente.

Em seguida foram testados os componentes necessários para a comunicação por infravermelho, que são: encoder/decoder TIR1000, da Texas Instruments, e transceiver ZXH1810, da Zilog. O circuito para cada microcontrolador utilizado nessa etapa pode ser visto na figura 18.

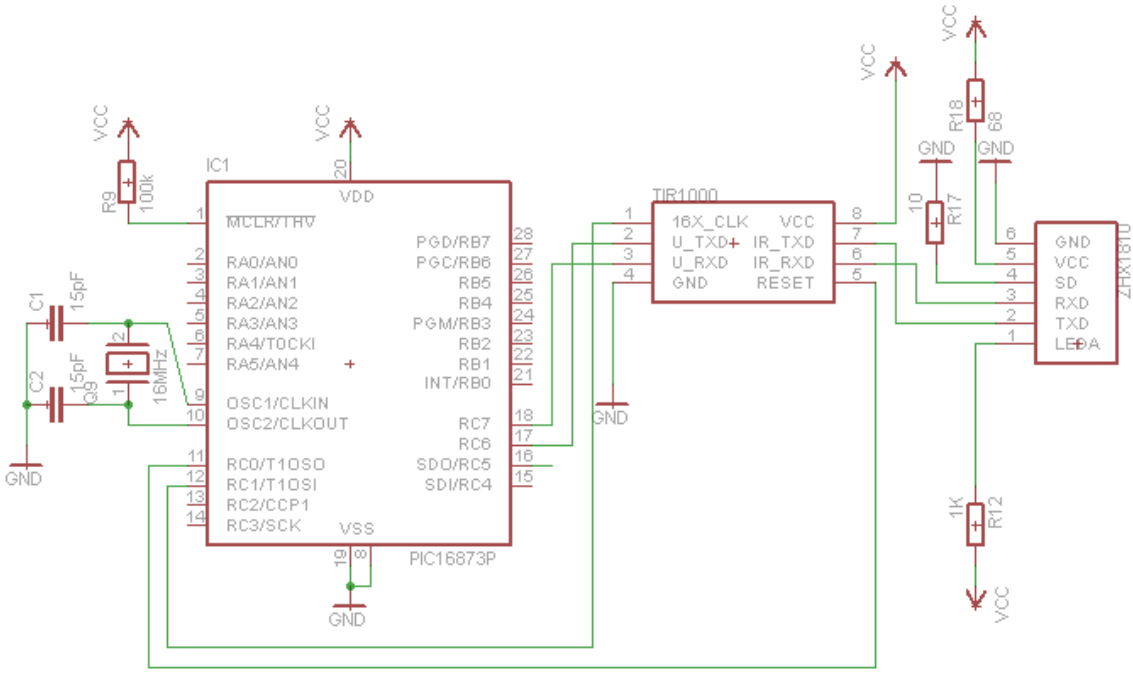


Figura 18 - Circuito para teste de comunicação por infravermelho.
Fonte: Autoria própria.

Os próximos componentes a serem testados foram os que pertencem ao circuito de alimentação, mostrado na figura 19. O propósito desse circuito é obter 5V a partir de duas pilhas AA recarregáveis de 1.2V cada com a aplicação do circuito integrado elevador de tensão MAX1674.

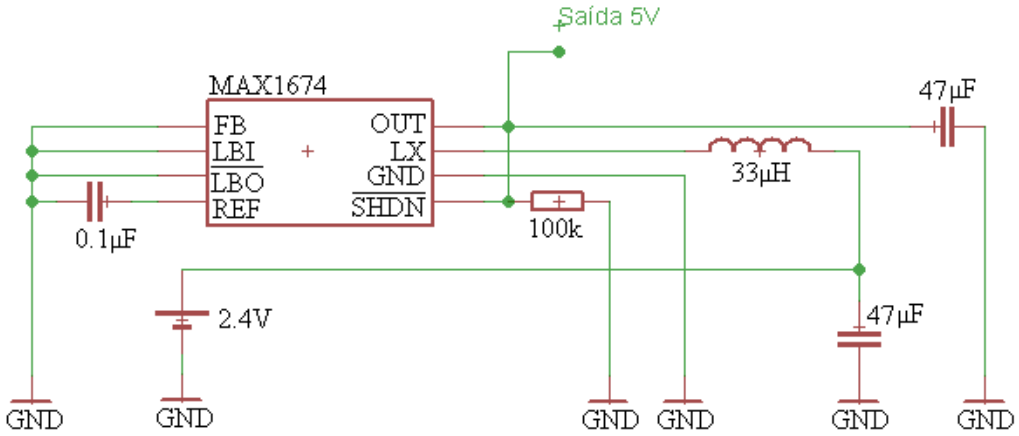


Figura 19 - Circuito de alimentação.

Fonte: Autoria própria.

O último teste realizado foi com relação à ativação das matrizes de LED. Cada matriz é formada por 64 LEDs, os quais estão distribuídos em 8 linhas e 8 colunas. Seu esquema elétrico se dá de acordo com a figura 20.

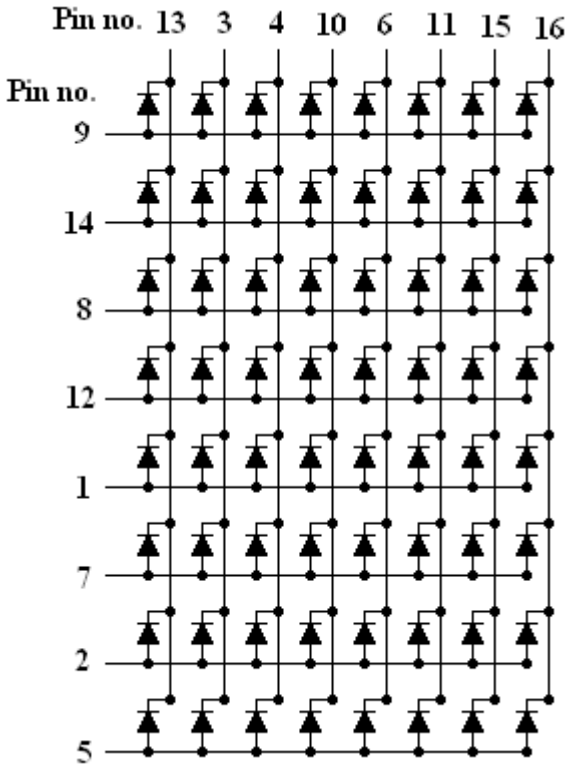


Figura 20 - Esquema elétrico da matriz de LED.

Fonte: Autoria própria.

O teste se deu pela ligação alternada dos pinos da matriz à saída positiva e negativa de uma fonte de 5V DC. No teste foi utilizado um resistor de 1000 Ohms para limitar a corrente de ativação dos LEDs em 5mA. Com isso, foi possível avaliar cada LED das matrizes.

Ao concluir os testes acima mencionados foi então dado início ao desenvolvimento dos circuitos eletrônicos. Esse processo foi iniciado pela definição da finalidade de cada porta do microcontrolador. Para isso, foi feita uma análise geral dos componentes e a quantidade de pinos requerida para o controle de cada um.

3.1.2 Matriz de LED

No caso da matriz de LED têm-se dezesseis pinos de controle, como mostrado anteriormente na figura 20: oito conectando os anodos de uma linha de LEDs e oito conectando os catodos de uma coluna de LEDs. Assim, a princípio é necessário utilizar dezesseis pinos do microtrolador para o seu controle. Porém, a fim de otimizar o número de portas do PIC a serem utilizadas, foi adotado nesse trabalho o método de controle por varredura de colunas.

Nesse método, o controle da matriz é feito pela varredura de suas colunas, ativando-as em sequência. Sendo feita a uma alta velocidade, essa varredura nos dá a impressão de que mais de uma coluna está ativada ao mesmo tempo, efeito conhecido como “persistência da visão”, como ilustrado na figura 21.

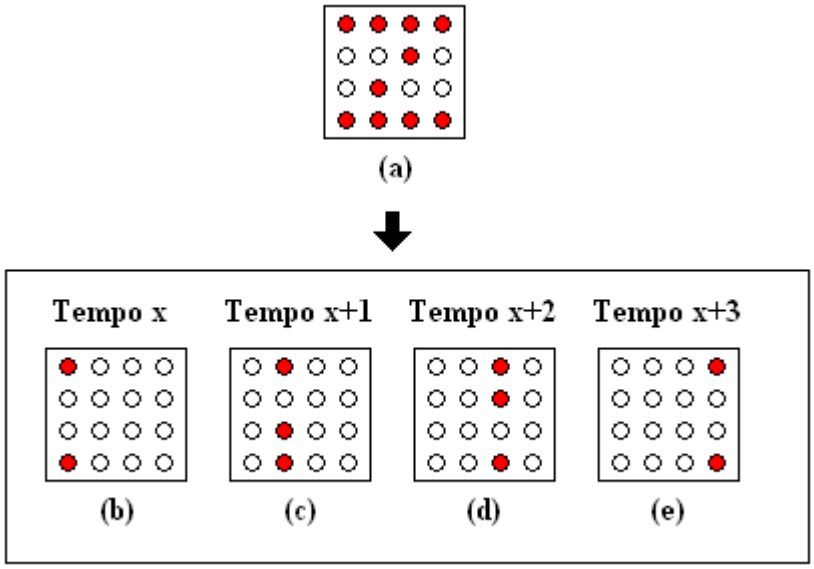


Figura 21 - Ilustração do controle de matriz 4x4 de LED pela varredura de suas colunas. Em (a) têm-se a matriz formando a letra Z, vista a olho nu. Já em (b), (c), (d) e (e) têm-se uma amostra da matriz em cada tempo do processo de varredura.
Fonte: Autoria própria.

Como visto, em um determinado instante de tempo estão ativos somente os LEDs de uma mesma coluna da matriz. No próximo instante somente os LEDs da coluna seguinte serão ativados e assim por diante. Com isso, para controlar uma matriz 8x8 é preciso utilizar um circuito que faça a varredura das colunas e indicar quais LEDs devem estar ativos a cada passo do processo de varredura.

Para isso, foi utilizado o registrador de deslocamento 74HC164 para varrer as colunas da matriz enquanto a porta B de oito *bits* do microcontrolador indica quais LEDs da coluna em questão devem ser ativados. Uma vez que o registrador de deslocamento requer somente 3 sinais de controle, têm-se uma otimização do número de portas utilizadas para o controle da matriz de 16 para 11 portas. Segue a figura 22 com circuito final de controle da matriz de LED.

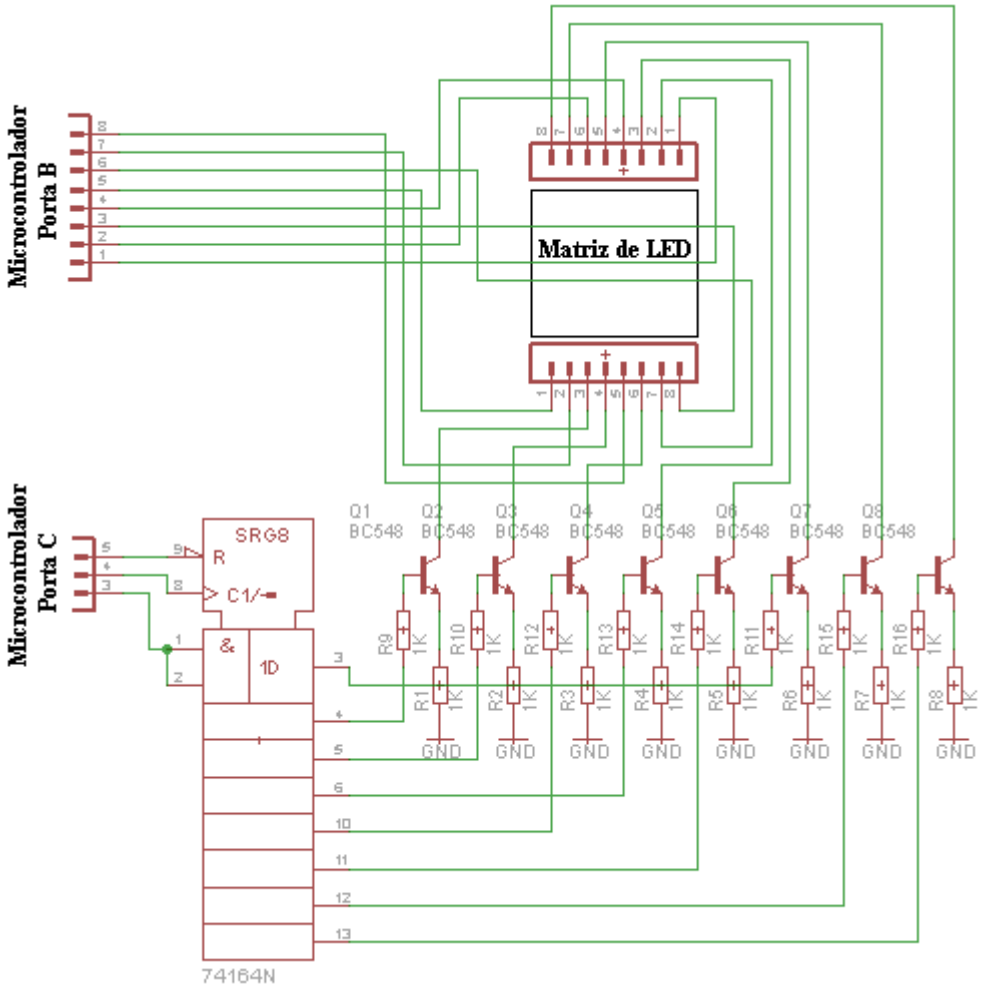


Figura 22 - Circuito final de controle da matriz de LED.

Fonte: Autoria própria.

3.1.3 Comunicação por Infravermelho

Como mostrado na figura 1, cada módulo do painel será capaz de se comunicar com até 4 outros módulos, localizados um em cada um de seus lados. Para isso, é necessário que ele tenha quatro transceptores de infravermelho, como mostra a figura 23.

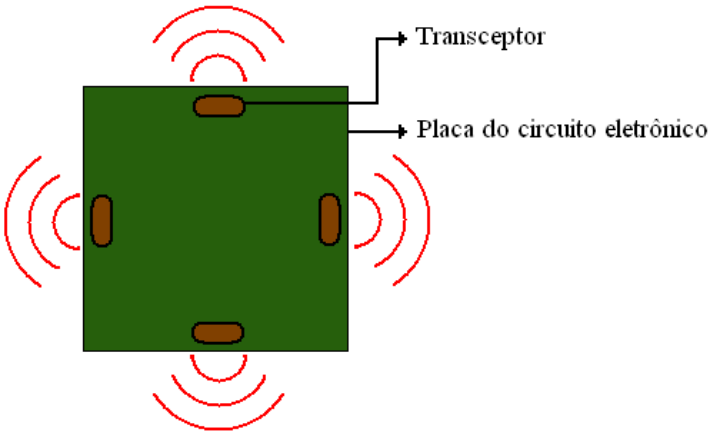


Figura 23 - Vista superior da placa de circuito eletrônico com destaque para a localização dos quatro transceptores de infravermelho.
Fonte: Autoria própria.

Tendo em vista que o microcontrolador utilizado nesse projeto, PIC 16F873A, contém somente um módulo de comunicação serial, foi necessário desenvolver um circuito para direcionar os sinais desse módulo para cada transceptor. Do contrário, seria necessário desenvolver outros módulos de comunicação serial manualmente, o que foi evitado nesse projeto.

Assim, foram utilizados circuitos MUX e DEMUX interligando os sinais de transmissão (IR_TX) e recepção (IR_RX) do codificador/decodificador TIR1000 e dos transceptores. A seleção de MUX e DEMUX é feito por pinos do microcontrolador, na qual uma das entradas de cada porta lógica “E” está conectada a um pino. Além disso, as portas lógicas conectadas a um mesmo transceptor são ativadas pelo mesmo pino de controle do microcontrolador. Dessa forma, é possível controlar para onde ou de onde o sinal serial será transmitido ou recebido ativando o respectivo pino de controle do microcontrolador. O circuito final desenvolvido para a comunicação serial por infravermelho pode ser visto na figura 24.

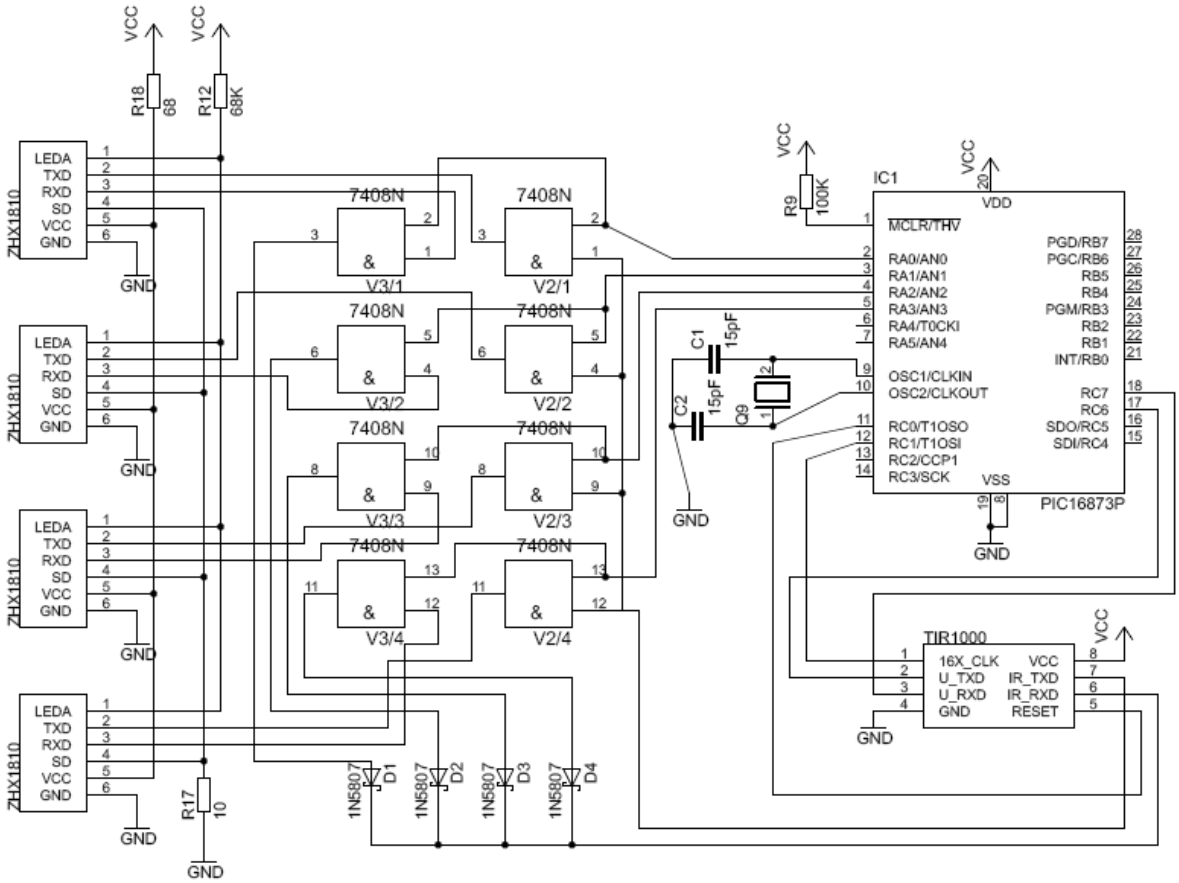


Figura 24 - Circuito final de comunicação por infravermelho.

Fonte: Autoria própria.

3.1.4 Processador

Com o desenvolvimento dos circuitos de controle da matriz de LED e da comunicação serial por infravermelho ficaram definidas as finalidades de cada porta do microcontrolador PIC, como mostra a figura 25.

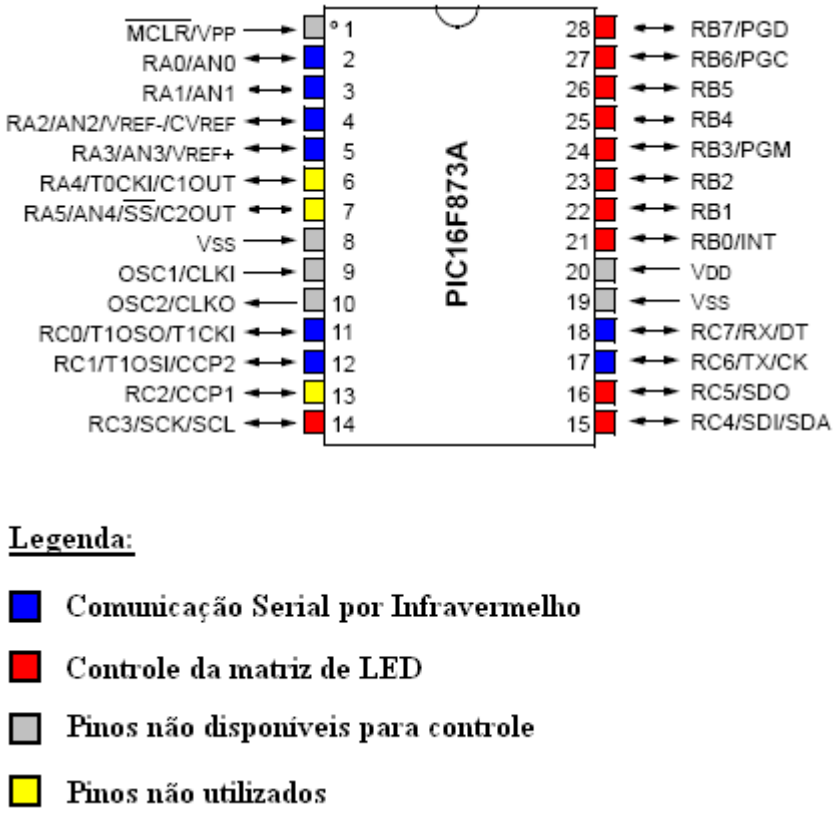


Figura 25 - Finalidade atribuída aos pinos do microcontrolador.

Fonte: Autoria própria.

Com relação ao circuito de alimentação dos módulos, não houve a necessidade de alterá-lo. O mesmo circuito utilizado para os testes iniciais, mostrado na figura 19, foi aplicado como circuito final de alimentação.

Dessa forma, foram desenvolvidos os circuitos para o funcionamento completo de um módulo do painel. O próximo passo foi projetar como os seus circuitos e componentes estão dispostos mecanicamente.

3.2 MECÂNICA

Os circuitos desenvolvidos compõem-se principalmente pelos seguintes componentes eletrônicos, mostrados na figura 26.

1. Suporte de pilha;
2. Indutor 33µH;
3. Cristal Oscilador de 16 Mhz;
4. Transceiver Infravermelho, modelo ZHX1810, da Zilog®;
5. Transistor BC548;
6. Capacitores: 47µF, 0,1µF e 15pF;
7. Microcntrolador PIC16F873A;
8. Circuito integrado DM7408 com 4 portas E;
9. Registrador de deslocamento 74164;
10. Encoder/Decoder de sinal infravermelho TIR1000, da Texas Instruments®;
11. Coversor DC-DC modelo MAX1674, da MAXIM®;
12. Resistores: 1000, 68 , 10, 100000 e 5200 Ohms;
13. Matriz de LED.

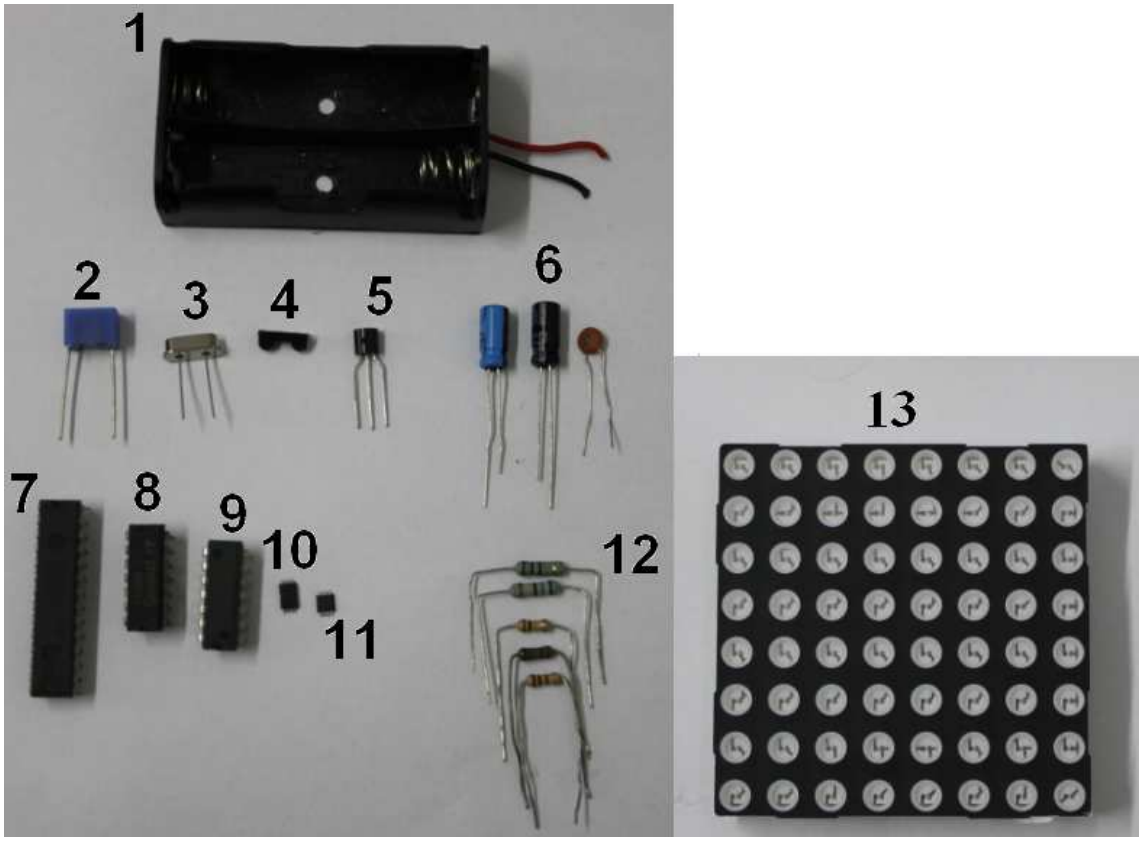


Figura 26 - Visualização dos principais componentes eletrônicos que compõem um módulo do painel.

Fonte: Autoria própria.

Tendo em vista que as bordas das matrizes de LED distintas devem se encostar para formar um painel sem falhas, é necessário que o comprimento da placa de circuito impresso não seja maior que o comprimento da matriz de LED, a qual possui 6x6cm, como ilustrado na figura 27.

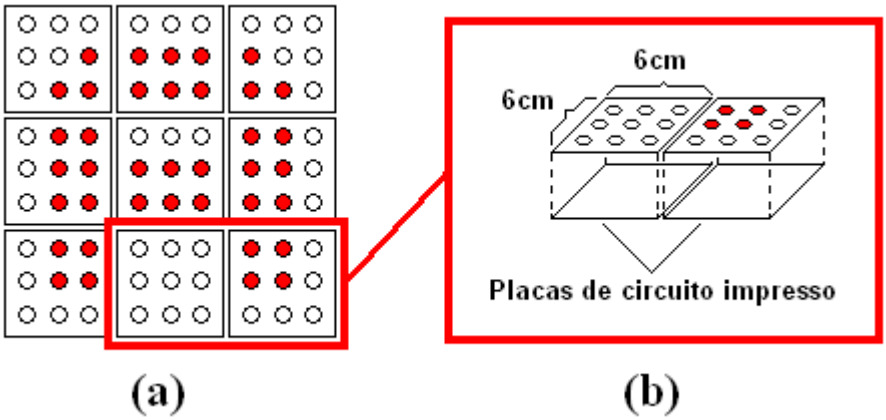


Figura 27 - (a) Vista superior de 9 módulos lado a lado. (b) Ilustração da limitação do tamanho das placas de circuito impresso imposta pelas medidas da matriz de LED.
Fonte: Autoria própria.

Por causa da limitação de tamanho, foram produzidas quatro placas de circuito impresso por módulo, onde cada uma delas contém uma parte do circuito eletrônico. Elas foram projetadas para ficarem sobrepostas e se unirem por barras de pinos torneados, os quais interligam os circuitos eletrônicos de placas distintas e sustentam toda a estrutura, como mostra a figura 28.

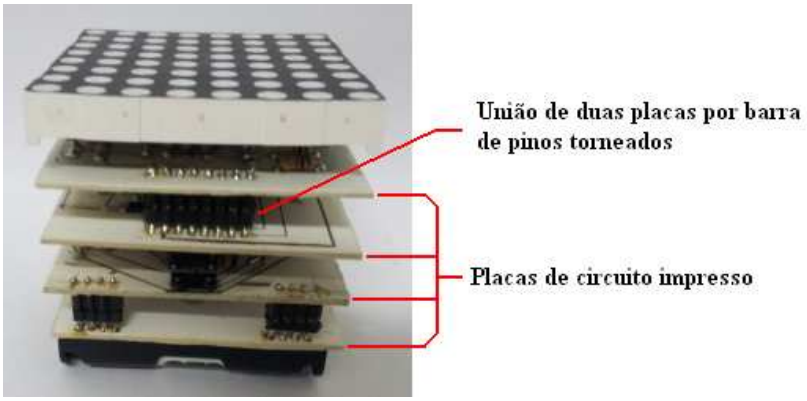


Figura 28 - Imagem real mostrando a disposição mecânica das placas de um módulo do painel de LED.
Fonte: Autoria própria.

3.3 PROGRAMAÇÃO

Os programas foram desenvolvidos em linguagem C e os códigos compilados com o auxílio do software MPLAB Integrated Development Environment (IDE), da Microchip. O objetivo foi desenvolver um jogo onde uma cobra se move pelo painel e tem como meta comer as frutas que surgirão de forma aleatória. A cada fruta que a cobra come ela deve crescer uma unidade. O jogo é reiniciado se a cobra colidir com uma parte de seu próprio corpo.

Para tanto, foram desenvolvidos dois códigos distintos: um para a simulação da cobra e outro para o controle da mesma, sendo que cada código foi aplicado a um módulo do painel. Dessa forma, obteve-se um módulo de controle e outro de simulação do jogo, como indica a figura 29.

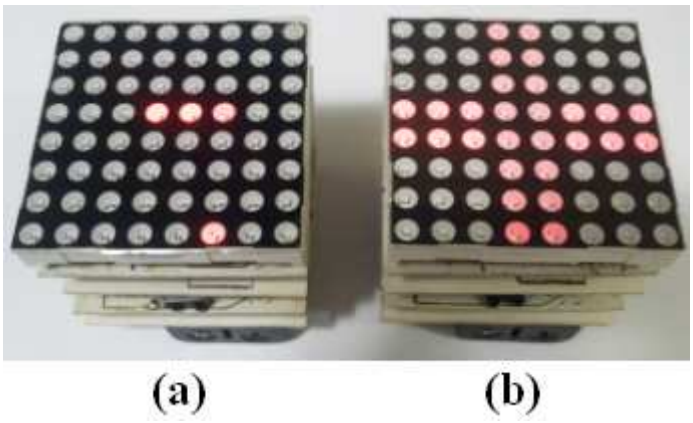


Figura 29 - Foto real de módulos do painel em funcionamento. Em (a) encontra-se um módulo simulando o jogo. Já em (b) se vê um módulo executando o software de controle do jogo.
Fonte: Autoria própria.

3.4 SOFTWARE DE CONTROLE

Tendo em vista o *hardware* desenvolvido, a única maneira de interagir com o jogo é pela movimentação dos módulos. Por isso, foi definido que o controle do jogo se dará pela movimentação do módulo de controle, o qual deverá estar próximo o

suficiente ao módulo de simulação do jogo para que esse receba o sinal de infravermelho enviado. Ao detectar uma rotação do módulo de controle o programa irá alterar a direção da cobra conforme o sentido de rotação detectado. Portanto, o objetivo principal do software de controle é transmitir um sinal de infravermelho único por cada transmissor do módulo de controle.

Como informado em “3.1.3 Comunicação por Infravermelho”, a transmissão de sinal serial é feita pelo único módulo serial do microcontrolador em questão. Por isso, os transceptores do módulo de controle deverão enviar os respectivos sinais de infravermelho um de cada vez. Dessa forma, o software de controle se baseia em distribuir a transmissão dos sinais entre os transceptores do módulo em questão.. Além disso, com o propósito de identificação do módulo de controle o software também aciona a matriz de LED de forma a formar uma cruz, como indicado na figura 28 acima. Nos apêndices A e B se encontram o fluxograma e o código do software de controle do jogo, respectivamente.

3.5 SOFTWARE DE SIMULAÇÃO DO JOGO

O código para a simulação do jogo tem como principal função a movimentação da cobra de acordo com os comandos de controle. Para isso, a recepção do sinal do módulo de controle deve se dar através de qualquer um dos transceptores do módulo de simulação. Por isso, o software de simulação mantém ativos os quatro transceptores do módulo em questão.

O software de simulação basicamente toma as seguintes ações:

- Atualiza o display de LED de forma a formar a cobra e a fruta;
- Movimenta a cobra pelo display do módulo;
- Altera a direção do movimento da cobra de acordo com os sinais de controle recebidos;
- Detecta a colisão da cobra com uma parte do seu próprio corpo e, caso necessário, reinicia o jogo.

Nos apêndices C e D se encontram o fluxograma e o código do software de simulação do jogo, respectivamente.

4 CONCLUSÃO

Jogos eletrônicos podem estimular de diversas maneiras nossa capacidade intelectual e motora. Por isso, o desenvolvimento de novas tecnologias voltadas a esse setor pode ter grande importância social.

Com frequência vemos o lançamento de jogos e consoles com novas formas de interatividade como, por exemplo, o controle sem fio e o reconhecimento de movimentos corporais. O desenvolvimento desses produtos só é possível por causa da pesquisa e aplicação de novas tecnologias como a comunicação sem fio e componentes eletrônicos mais poderosos.

Dentro desse contexto, esse projeto propôs a construção de um painel modular de LED voltado para o desenvolvimento de jogos. Utilizando sinais de infravermelho, os módulos do painel são capazes de se comunicar e reconhecer a movimentação de um módulo distinto, o que possibilita a interação do homem com o painel.

O desenvolvimento do painel se deu sem grandes problemas. Os resultados dos testes iniciais foram satisfatórios, o que resultou em pouca modificação dos circuitos eletrônicos no desenvolvimento do projeto.

A construção dos softwares de simulação e controle do jogo proposto foi uma das etapas que mais consumiu tempo. Isso se deu principalmente pela utilização inicial do compilador CCS, com o qual foram enfrentados problemas para executar o programa desenvolvido. Foi então que se optou pela utilização do compilador MPLAB Integrated Development Environment (IDE), da Microchip, com o qual foi construído o programa final.

Foi pretendida a construção de dez módulos para o painel, nove para a simulação do jogo e um para o controle do mesmo. Porém, o esforço e tempo requeridos para o desenvolvimento dos mesmos resultou na construção de apenas três módulos, o que foi considerado suficiente para a demonstração da viabilidade do projeto proposto. Com eles é possível demonstrar a comunicação entre módulos e interagir com o jogo criado.

Com isso, fica aberta a possibilidade de aprimorar esse projeto em trabalhos futuros. A construção de novos módulos trará maiores desafios, principalmente com relação à programação. Por exemplo, utilizar vários módulos para a

formação de uma imagem em movimento é um grande desafio que certamente proporcionará um rico aprendizado para os alunos envolvidos.

REFERÊNCIAS

ANTUNES, Celso. **Jogos para a estimulação das múltiplas inteligências**. 11ª Ed. Petrópolis, Rio de Janeiro: Vozes, 1998. 295p.

BATLLORI, Jorge. **Jogos para treinar o cérebro**.São Paulo-SP:Mandras, 2004. 147p.

COMER, Douglas E. **Redes de computadores e internet**. 4ª Ed. Porto Alegre: Bookman, 2007. 632p.

DORNAN, Andy. **Wireless communication: o guia essencial de comunicação Sem Fio**. Rio de Janeiro-RJ: Campus, 2001. 304p.

GAGE, Stan. **Optoelectronics applications manual**. New York: McGraw-Hill, 1977. 133p.

GILLESSEN, Klaus; SCHAIRER, Werner. **Light emitting diodes: an introduction**. Cambridge: Prentice-Hall International, 1987.

IDOETA, Ivan V; CAPUANO, Gabriel F. **Elementos de eletrônica digital**. 40. ed. São Paulo: Érica, 2008.

MAXIM INTEGRATED PRODUCTS. **High-efficiency, low-supply-current, compact, step-up DC-DC converters.** Rev 3. San Jose: Maxim Integrated Products. 12p.

MELLO, Luiz Fernando P. de. **Análise e projeto de fontes chaveadas**. São Paulo: Érica, 1996.

MICROCHIP TECHNOLOGY. **PIC16F87XA data sheet**. Chandler: Microchip Technology, 2003. 234p.

NOVAK, Jeannie; UNGER, Kimberly. **Game development essentials: mobile game development.** 1ª Ed. Stamford: Cengage Learning, 2012. 228p.

SOUZA, David José de. **Desbravando o PIC: baseado no microcontrolador PIC16F84**.5ª Ed. São Paulo – SP: Ed. Érica, 2000. 200p.

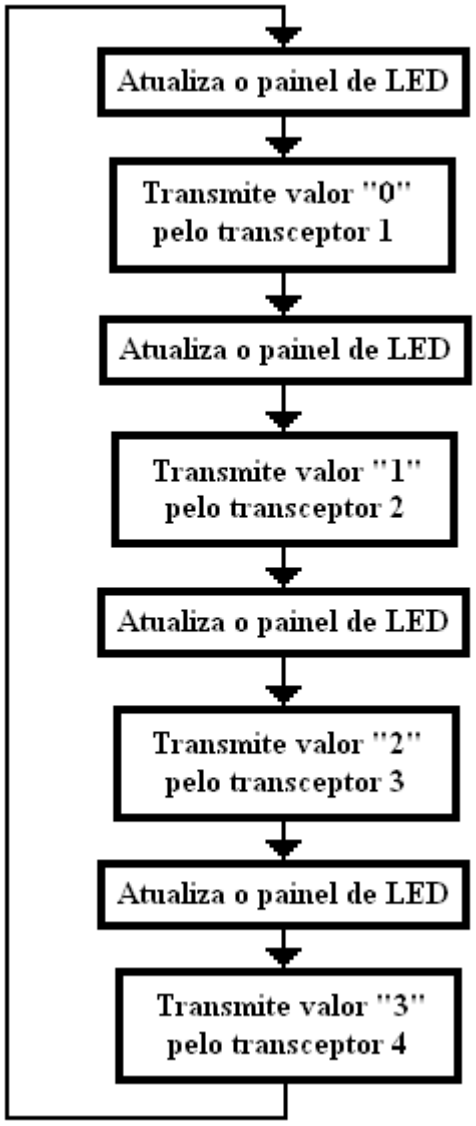
TEXAS INSTRUMENTS. **SN54HC164, SN74HC164 8-Bit parallel-out serial shift registers.** Chandler: Texas Instruments, 1997. 7p.

TEXAS INSTRUMENTS. **TIR1000, TIR1000I standalone IrDA encoder and decoder.** Chandler: Texas Instruments, 1999. 9p.

THE ENTERTAINMENT SOFTWARE ASSOCIATION. **Industry facts**. Disponível em <http://www.theesa.com/facts/index.asp>. Acesso em: 31 OUT. 2011.

ZILOG. **ZHX1810: Slim series SIR transceiver product specification**. Zilog, Inc. 2010.

APÊNDICE A – Fluxograma do Software de Controle do Jogo



APÊNDICE B – Código do software de Controle do Jogo

```
#include <htc.h>
#include <stdio.h>
#include <stdlib.h>

__CONFIG(HS & WDTDIS & BORDIS & LVPDIS & DEBUGDIS);

#define _XTAL_FREQ 16000000

#define bitset(var, bitno) ((var) |= 1UL << (bitno))
#define bitclr(var, bitno) ((var) &= ~(1UL << (bitno)))


ANSEL = 0;
ANSELH = 0;


int t0_int_flag=0, contador=0;

unsigned char dado=0;
unsigned char info=0,linha=0, coluna=0, letra=0, contador_timer0=60, funcao=0, linha_cabeca_cobra=4,
coluna_cabeca_cobra=0, linha_rabo_cobra=4,coluna_rabo_cobra=0, inicializador_cobra=4, direcao_cobra=0, indice_trajetoria_cobra=0,
indice_direcao_rabo=2, direcao_rabo=0, linha_comida=3, coluna_comida=6, reset=0, reset_delay=0, tcver=1;

//DIRECAO_COBRA
//        1
//        |
//        |
//   2 ---- ----- 0
//        |
//        |
//        3

unsigned char
A[8]={0x00,0xfe,0xff,0x33,0x33,0xff,0xfe,0x00},
B[8]={0x00,0xff,0xff,0xdb,0xdb,0xdb,0x66,0x00},
C[8]={0x00,0x7e,0xff,0xc3,0xc3,0xc3,0x42,0x00},
D[8]={0x00,0xff,0xff,0xc3,0xc3,0xff,0x7e,0x00},
T[8]={0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff},
X[8]={0x81,0x42,0x24,0x18,0x18,0x24,0x42,0x81},
CRUZ[8]={0x18,0x18,0x18,0xff,0xff,0x18,0x18,0x18};

//trajetoria_cobra[65];

unsigned char  flag_comida=0;
unsigned char display_matrix[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```
void putch(unsigned char byte)
{

RC0 = 1; // Reset Irda Encoder/Decoder (TIR1000)
__delay_ms(1);
RC0 = 0;
__delay_ms(1);

  /* output one byte */
  while(!TXIF);   /* set when register is empty */
  TXREG = byte;
}

unsigned char getch(void) {

        if(OERR) //if over run error, then reset the receiver
        {
                CREN = 0;
                CREN = 1;
        }

  /* retrieve one byte */
  while(!RCIF);   /* set when register is not empty */
  return RCREG;
}

void interrupt ISR(void)

{
        // **** UART Interruption ****
        if(RCIF)  //If UART Rx Interrupt
        {
                if(OERR) //if over run error, then reset the receiver
                {
                        CREN = 0;
                        CREN = 1;
                }
                dado = RCREG;
        }

return;
}

void InitUART(void)
{
        //Comm Setup
        #define BAUDRATE 9600  //bps
        #ifndef _XTAL_FREQ
```

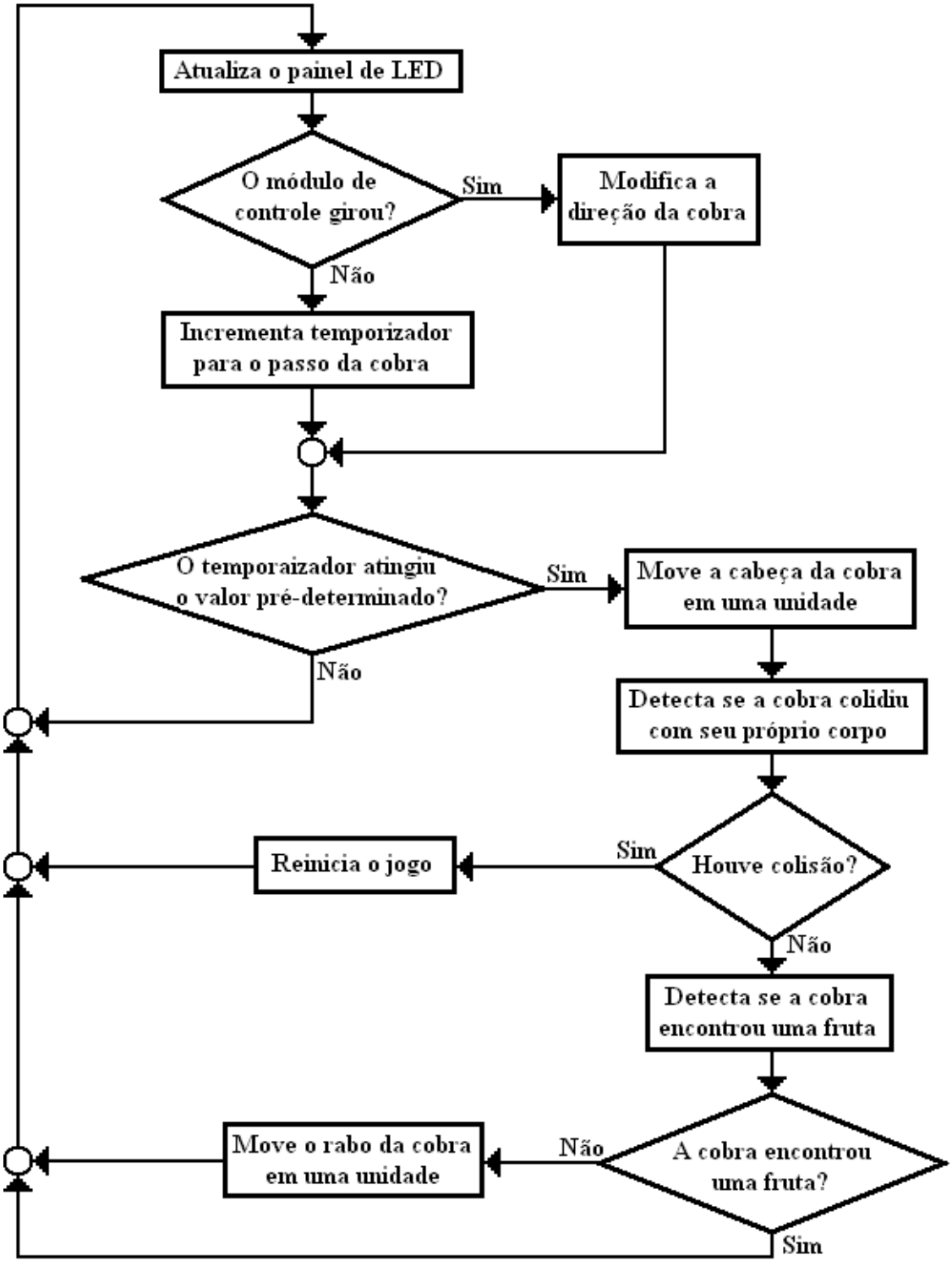
```
        #define _XTAL_FREQ 16000000   //MHz
        #endif

        TRISC6 = 0;   //TX Pin
        TRISC7 = 1;   //RX Pin

        SPBRG = ((_XTAL_FREQ/16)/BAUDRATE) - 1;
        BRGH  = 1;                  //fast baudrate
        SYNC  = 0;                                      //asynchronous
        SPEN  = 1;                                      //enable serial port pins
        CREN  = 1;                                      //enable reception
        SREN  = 0;                                      //no effect
        TXIE  = 0;                                      //disable tx interrupts
        RCIE  = 1;                                      //enable rx interrupts
        TX9   = 0;                                      //8-bit transmission
        RX9   = 0;                                      //8-bit reception
        TXEN  = 0;                                      //reset transmitter
        TXEN  = 1;                                      //enable the transmitter
}

void InitTIMER0(void)
{
        T0CS = 0;       // Internal instruction cycle clock (CLKO)
        PSA  = 0;   // Prescaler is assigned to the Timer0 module
        PS0  = 1;       // To have prescaler as 1:256
        PS1  = 1;       // To have prescaler as 1:256
        PS2  = 1;       // To have prescaler as 1:256
    TMR0IE = 0; // Enables the TMR0 interrupt
        TMR0IF = 0; // Reset Timer0 Overflow Interrupt Flag bit
}

void display_refresh(void)
{
//Inicializa a varredura das colunas

  RC3 = 1;
  //Inicializa array
  RC4 = 1;
  //Array_clock
  RC5 = 0;
  RC5 = 1;
  RC4 = 0;

  for(coluna=0;coluna<=7;coluna++)
  {
    if(letra == 0)
    {
      PORTB = A[coluna];
    }
    else if(letra == 1)
```

```
        {
          PORTB = B[coluna];
        }
        else if(letra == 2)
        {
          PORTB = C[coluna];
        }
                else if(letra == 3)
          {
            PORTB = D[coluna];
          }
                        else if(letra == 4)
                {
                        PORTB = T[coluna];
                }
                                else if(letra == 5)
                        {
                                PORTB = CRUZ[coluna];
                        }
                                        else
                                                {
                                                        PORTB = X[coluna];
                                                }

        __delay_ms(1);
        PORTB = 0x00;

        //Array_clock
        RC5 = 1;
        RC5 = 0;
        RC5 = 1;
    }
}

void main(void)
{

ADCON1= 0x06 ; // Changes PORTA to digital
CMCON = 0x07 ; // Disable analog comparators


TRISA = 0x00 ; // Configure PORTA as output
TRISB = 0x00 ; // Configure PORTB as output
TRISC3 = 0 ;
TRISC4 = 0 ;
TRISC5 = 0 ;
RBPU = 0;

PORTA = 0x00 ; // Reset PORTA
PORTB = 0x00 ; // Reset PORTB
```

```
        // PWM Setup
        PR2 = 0x19   ;        // In order to generate a 153.6 Khz PWM frequency
        CCP2CON = 0X0C; // Set CCP2 PWM mode and PWM duty cycle register less significant bits (3:4) as 00
        CCPR2L = 0X0D;  // In order to have the PWM duty cycle at 50% of the pwm frequency
        T2CON = 0x04;      // Enable Timer2 operation in order to have PWM enabled as well
        TRISC1  = 0;          // Make CCP1 pin as output


        //Enable Interruptions
        GIE = 1;              // In order to work with interruption
        PEIE = 1;             // In order to work with peripheral interruption

        InitUART();
        InitTIMER0();

        unsigned char data_transmit;

        while(1){

        display_refresh();

        // TRANSMISSOR ***************
        letra = 5;

        contador++;
        if(contador == 20)
                {
                        contador = 0;

                RA3 = 0;
                RA0 = 1;
                putch(0);

                display_refresh();

                RA0 = 0;
                RA1 = 1;
                putch(1);

                display_refresh();

                RA1 = 0;
                RA2 = 1;
                putch(2);

                display_refresh();
```

```
        RA2 = 0;
        RA3 = 1;
        putch(3);

        display_refresh();

        }
}
}
```

APÊNDICE C – Fluxograma do Software de Simulação do Jogo



APÊNDICE D – Código do Software de Simulação do Jogo

```
#include <htc.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

__CONFIG(HS & WDTDIS & BORDIS & LVPDIS & DEBUGDIS);

#define _XTAL_FREQ 16000000

#define bitset(var, bitno) ((var) |= 1UL << (bitno))
#define bitclr(var, bitno) ((var) &= ~(1UL << (bitno)))


ANSEL = 0;
ANSELH = 0;


int contador=0;

unsigned char dado=0, buffer_dado=0, buffer_lado=0, coluna, indice_direcao_snake, indice_direcao_rabo, interruption_flag=0, linha_comida, coluna_comida,
                                comida_detected=0, contador_retorna_dado=0;
signed char calculo_direcao_snake, direcao_snake, linha_cabeca_snake, linha_rabo_snake, coluna_rabo_snake, coluna_cabeca_snake;

//  DIRECAO_SNAKE
//        1
//        |
//        |
//  2 ----- ----- 0
//        |
//        |
//        3

unsigned char
X[8]={0x81,0x42,0x24,0x18,0x18,0x24,0x42,0x81},
snake_directions_array[64],
display[8]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};


void putch(unsigned char byte)
{

RC0 = 1; // Reset Irda Encoder/Decoder (TIR1000)
__delay_ms(1);
RC0 = 0;
__delay_ms(1);
```

```
  /* output one byte */
  while(!TXIF);   /* set when register is empty */
  TXREG = byte;
}

unsigned char getch(void) {

        if(OERR) //if over run error, then reset the receiver
        {
                CREN = 0;
                CREN = 1;
        }

  /* retrieve one byte */
  while(!RCIF);   /* set when register is not empty */
  return RCREG;
}

void interrupt ISR(void)

{
        // **** UART Interruption ****
        if(RCIF)  //If UART Rx Interrupt
        {
                if(OERR) //if over run error, then reset the receiver
                {
                        CREN = 0;
                        CREN = 1;
                }
                dado = RCREG;
                interruption_flag = 1;
        }

return;
}

void InitUART(void)
{
        //Comm Setup
        #define BAUDRATE 9600  //bps
        #ifndef _XTAL_FREQ
        #define _XTAL_FREQ 16000000   //MHz
        #endif

        TRISC6 = 0;   //TX Pin
        TRISC7 = 1;   //RX Pin

        SPBRG = ((_XTAL_FREQ/16)/BAUDRATE) - 1;
        BRGH  = 1;                      //fast baudrate
```

```
        SYNC  = 0;                                  //asynchronous
        SPEN  = 1;                                  //enable serial port pins
        CREN  = 1;                                  //enable reception
        SREN  = 0;                                  //no effect
        TXIE  = 0;                                  //disable tx interrupts
        RCIE  = 1;                                  //enable rx interrupts
        TX9   = 0;                                  //8-bit transmission
        RX9   = 0;                                  //8-bit reception
        TXEN  = 0;                                  //reset transmitter
        TXEN  = 1;                                  //enable the transmitter
}

void InitTIMER0(void)
{
        T0CS = 0;       // Internal instruction cycle clock (CLKO)
        PSA  = 0;   // Prescaler is assigned to the Timer0 module
        PS0  = 1;        // To have prescaler as 1:256
        PS1  = 1;        // To have prescaler as 1:256
        PS2  = 1;        // To have prescaler as 1:256
    TMR0IE = 0; // Disable the TMR0 interrupt
        TMR0IF = 0; // Reset Timer0 Overflow Interrupt Flag bit
}

void display_refresh(void)
{
//Inicializa a varredura das colunas

  RC3 = 1;
  //Inicializa array
  RC4 = 1;
  //Array_clock
  RC5 = 0;
  RC5 = 1;
  RC4 = 0;

  for(coluna=0;coluna<=7;coluna++)
  {

                PORTB = display[coluna];

      __delay_ms(1);
      PORTB = 0x00;

      //Array_clock
      RC5 = 1;
      RC5 = 0;
      RC5 = 1;
  }
}
```

```
char get_display_bit(char linha_display_set,char coluna_display_set)
{

	char mascara_display_set, result;

		switch(linha_display_set)
		{
		case 0:
		mascara_display_set = 0b00000001;
		break;
		case 1:
		mascara_display_set = 0b00000010;
		break;
		case 2:
		mascara_display_set = 0b00000100;
		break;
		case 3:
		mascara_display_set = 0b00001000;
		break;
		case 4:
		mascara_display_set = 0b00010000;
		break;
		case 5:
		mascara_display_set = 0b00100000;
		break;
		case 6:
		mascara_display_set = 0b01000000;
		break;
		case 7:
		mascara_display_set = 0b10000000;
		break;
		}

	result = mascara_display_set & display[coluna_display_set];

return result;
}

void change_display_bit(char linha_display,char coluna_display,char valor)
{
	char mascara_display;

	switch(linha_display)
		{
		case 0:
		mascara_display = 0b00000001;
		break;
		case 1:
		mascara_display = 0b00000010;
		break;
```

```
                case 2:
                mascara_display = 0b00000100;
                break;
                case 3:
                mascara_display = 0b00001000;
                break;
                case 4:
                mascara_display = 0b00010000;
                break;
                case 5:
                mascara_display = 0b00100000;
                break;
                case 6:
                mascara_display = 0b01000000;
                break;
                case 7:
                mascara_display = 0b10000000;
                break;
                }

        switch(valor)
                {
                case 0:
                                if(get_display_bit(linha_display,coluna_display))
display[coluna_display] = display[coluna_display] - mascara_display;
                break;
                case 1:
                if(!get_display_bit(linha_display,coluna_display))
display[coluna_display] = display[coluna_display] + mascara_display;
                break;
                }

        }


        void solta_comida()
        {

                srand(TMR0);
                linha_comida = rand() % 10;
                srand(TMR0);
                coluna_comida = rand() % 10;

                while((linha_comida > 7) || (coluna_comida > 7) || (linha_comida < 0) ||
(coluna_comida < 0) || (get_display_bit(linha_comida,coluna_comida) != 0) )
                {
                        srand(TMR0);
                        linha_comida = rand() % 10;
                        srand(TMR0);
                        coluna_comida = rand() % 10;
```

```
		}

		change_display_bit(linha_comida,coluna_comida,1);
	}

	void reset_snake()
	{
		linha_cabeca_snake = 3;
		coluna_cabeca_snake = 2;

		linha_rabo_snake = 3;
		coluna_rabo_snake = 0;

		direcao_snake = 0;

		indice_direcao_snake = 2;
		indice_direcao_rabo = 0;

		for(char reset_counter = 0; reset_counter<8 ; reset_counter++)
			{
				display[reset_counter] = 0x00;
			}

		solta_comida();

		change_display_bit(3,0,1);
		change_display_bit(3,1,1);
		change_display_bit(3,2,1);

		for(char reset_counter2 = 0; reset_counter2<64 ; reset_counter2++)
			{
				snake_directions_array[reset_counter2] = 0x00;
			}

	}

	bit detecta_colisao()
	{
		if(!get_display_bit(linha_cabeca_snake,coluna_cabeca_snake)) return 0;
		if(get_display_bit(linha_cabeca_snake,coluna_cabeca_snake)		&&
((linha_cabeca_snake != linha_comida) || (coluna_cabeca_snake != coluna_comida)))
			{
				for(char counter = 0;counter<8;counter++)
					{
						display[counter]=X[counter];
					}
				for(char counter2 = 0;counter2<100;counter2++)
					{
						display_refresh();
					}
```

```
                        reset_snake();
                        return 1;
                }
                else
                        {
                                return 0;
                        }
        }

        void snake_passo()
        {
            //*************************| MOVE CABECA |*************************
                switch(direcao_snake)
                        {
                        case 0:
                        coluna_cabeca_snake++;
                        break;
                        case 1:
                        linha_cabeca_snake--;
                        break;
                        case 2:
                        coluna_cabeca_snake--;
                        break;
                        case 3:
                        linha_cabeca_snake++;
                        break;
                        }

                if(coluna_cabeca_snake == 8) coluna_cabeca_snake = 0;
                if(coluna_cabeca_snake < 0) coluna_cabeca_snake = 7;
                if(linha_cabeca_snake == 8) linha_cabeca_snake = 0;
                if(linha_cabeca_snake < 0) linha_cabeca_snake = 7;

                if(!detecta_colisao())
                {
                        change_display_bit(linha_cabeca_snake,coluna_cabeca_snake,1);

                        snake_directions_array[indice_direcao_snake] = direcao_snake;

                        indice_direcao_snake++;
                        if(indice_direcao_snake == 64) indice_direcao_snake = 0;

                        //*************************| MOVE RABO |*************************

                        if(!comida_detected)
                        {
                                change_display_bit(linha_rabo_snake,coluna_rabo_snake,0);
```

```
                        switch(snake_directions_array[indice_direcao_rabo])
                                {
                                        case 0:
                                coluna_rabo_snake++;
                                break;
                                case 1:
                                linha_rabo_snake--;
                                        break;
                                case 2:
                                coluna_rabo_snake--;
                                break;
                                case 3:
                                linha_rabo_snake++;
                                break;
                                }

                        if(coluna_rabo_snake == 8) coluna_rabo_snake = 0;
                        if(coluna_rabo_snake < 0) coluna_rabo_snake = 7;
                        if(linha_rabo_snake == 8) linha_rabo_snake = 0;
                        if(linha_rabo_snake < 0) linha_rabo_snake = 7;

                        indice_direcao_rabo++;
                        if(indice_direcao_rabo == 64) indice_direcao_rabo = 0;
                }
                else
                        {
                                comida_detected = 0;
                        }
        }
}

void detecta_comida()
{
        if((linha_comida == linha_cabeca_snake) && (coluna_comida ==
coluna_cabeca_snake))
                {
                        comida_detected = 1;
                        solta_comida();
                }
}

void snake()
{
        snake_passo();
        detecta_comida();
}

void muda_direcao_snake()
{
        calculo_direcao_snake = dado - buffer_dado;
```

```
	if((calculo_direcao_snake == 1) || (calculo_direcao_snake == -3))
		{
			direcao_snake--;
		}
		else if((calculo_direcao_snake == -1) || (calculo_direcao_snake == 3))
			{
				direcao_snake++;
			}

	if(direcao_snake == 4) direcao_snake = 0;
	if(direcao_snake == -1) direcao_snake = 3;

	buffer_dado = dado;

}

void informa_lado()
{
	RA3 = 0;
	RA0 = 1;
	putch(4);

	display_refresh();

	RA0 = 0;
	RA1 = 1;
	putch(5);

	display_refresh();

	RA1 = 0;
	RA2 = 1;
	putch(6);

	display_refresh();

	RA2 = 0;
	RA3 = 1;
	putch(7);

	display_refresh();
}

void retorna_lado()
{

	for(char contador_retorno = 0;contador_retorno<3;contador_retorno++)
	{
		RA3 = 0;
```

```
                RA0 = 1;
                putch(dado);

                display_refresh();

                RA0 = 0;
                RA1 = 1;
                putch(dado);

                display_refresh();

                RA1 = 0;
                RA2 = 1;
                putch(dado);

                display_refresh();

                RA2 = 0;
                RA3 = 1;
                putch(dado);

                display_refresh();
        }

        buffer_lado = dado;

}

void main(void)
{

ADCON1= 0x06 ; // Changes PORTA to digital
CMCON = 0x07 ; // Disable analog comparators


TRISA = 0x00 ; // Configure PORTA as output
TRISB = 0x00 ; // Configure PORTB as output
TRISC3 = 0 ;
TRISC4 = 0 ;
TRISC5 = 0 ;
RBPU =      0       ;

PORTA = 0x00 ; // Reset PORTA
PORTB = 0x00 ; // Reset PORTB

// PWM Setup
PR2 = 0x19   ;       // In order to generate a 153.6 Khz PWM frequency
CCP2CON = 0X0C; // Set CCP2 PWM mode and PWM duty cycle register less significant bits (3:4) as 00
```

```
	CCPR2L = 0X0D;  // In order to have the PWM duty cycle at 50% of the pwm frequency
	T2CON = 0x04;	// Enable Timer2 operation in order to have PWM enabled as well
	TRISC1  = 0;	// Make CCP1 pin as output


	//Enable Interruptions
	GIE = 1;		// In order to work with interruption
	PEIE = 1;		// In order to work with peripheral interruption

	InitUART();
	InitTIMER0();

	reset_snake();

	RA0 = 1;
	RA1 = 1;
	RA2 = 1;
	RA3 = 1;

	while(1){

	display_refresh();

	if(interruption_flag)
		{
			if(((dado == 0) || (dado == 1) || (dado == 2) || (dado == 3)) & (dado != buffer_dado)) muda_direcao_snake();
		}

	contador++;
	if(contador == 50)
		{
			snake();
			contador = 0;
		}

	}
	}
```