Jogo do Choque

Danilo Souza, Hugo Santos, Welton Araújo
Emails: {dhcsouza, hugoleonardoeng07, weltonmaxx007}@gmail.com
Matrículas: 10080000801, 10080000701, 10080000501

***Abstract*—There is a lot of possibilities to do with microcontrollers because they are very flexible. This project implements a multiplayer shock game for 2 until 4 players. The code was made on machine language, in other words Assembly with specifics instructions for the PIC16F628A. This PIC has several limitations, but could perfectly attend the game requirements in this case.**

***Index Terms*—microcontroller, PIC16F628, game, shock, multiplayer**

I. Introduçao

O PIC16F628A faz parte da família PIC de microcontroladores da Microchip Technology, composta também por diversos outros modelos, como o PIC16F627, PIC16F648, que diferem somente na capacidade de memória flash disponível para armazenamento de programas, que para o PIC16F628A é de 2048 palavras de 32 bits (8KB)

A família PIC é amplamente utilizada hoje em dia devido à sua facilidade de programação e uma grande documentação disponível, suas aplicações podem variar desde simples protótipos até automação industrial, sua flexibilidade garante sua utilização em praticamente qualquer indústria.

A partir disto, o projeto abordado neste artigo aproveitou a possibilidade de implementar um jogo eletrizante para vários jogadores no PIC16F628A por ser um um dispositivo simples, de baixíssimo custo, porém com uma linguagem padrão Assembly.

II. Descrição do Projeto

O projeto baseia-se em um jogo, onde o objetivo é não levar choque, o brinquedo possui 4 controles com um botão cada e uma luz no meio, usada para indicar a hora em que os jogadores devem apertar os botões, o último leva um choque. As Figuras 1 e 2 mostram como é o dispositivo do jogo.

Ao começo do jogo, um LED fica aceso, a rodada termina quando o LED apaga e o perdedor leva um choque que é descarregado no próprio controle, onde há componentes metálicos. Caso algum jogador aperte o botão antes do tempo, este também levará choque como punição.

Para o projeto foram usados: 4 botões, um para cada jogador; 2 LED's para cada botão de jogador, sendo um para indicar que aquele jogador está participando do jogo e outro para indicar que aquele jogador levou um choque; um outro botão para representar o *start*; mais um botão para escolher o número de jogadores, sendo no mínimo 2 e no máximo 4; e um outro LED para indicar o começo de uma rodada.

Fig. 1. Ilustração do jogo vista de cima

Fig. 2. Ilustração do jogo vista de lado

III. Funcionamento do Programa

O algoritmo de funcionamento do programa funciona com o auxílio das variáveis da Tabela I onde também são descritas suas funções de controle. A Tabela II contém os pinos do PIC utilizados como saídas e entradas juntamente com suas funções. A Figura 3 mostra o fluxograma das rotinas mais determinantes do código.

Inicialmente, no *loop* da rotina *START*, o programa tem somente dois botões como entrada, *BOTAO_START* para iniciar e *BOTAO_N_PS* para definir o número de jogadores da partida. Na seleção do número de jogadores, os bits *BOT,2* e *BOT,3* são setados ou limpados enquanto que a variável *COUNTER* inicia

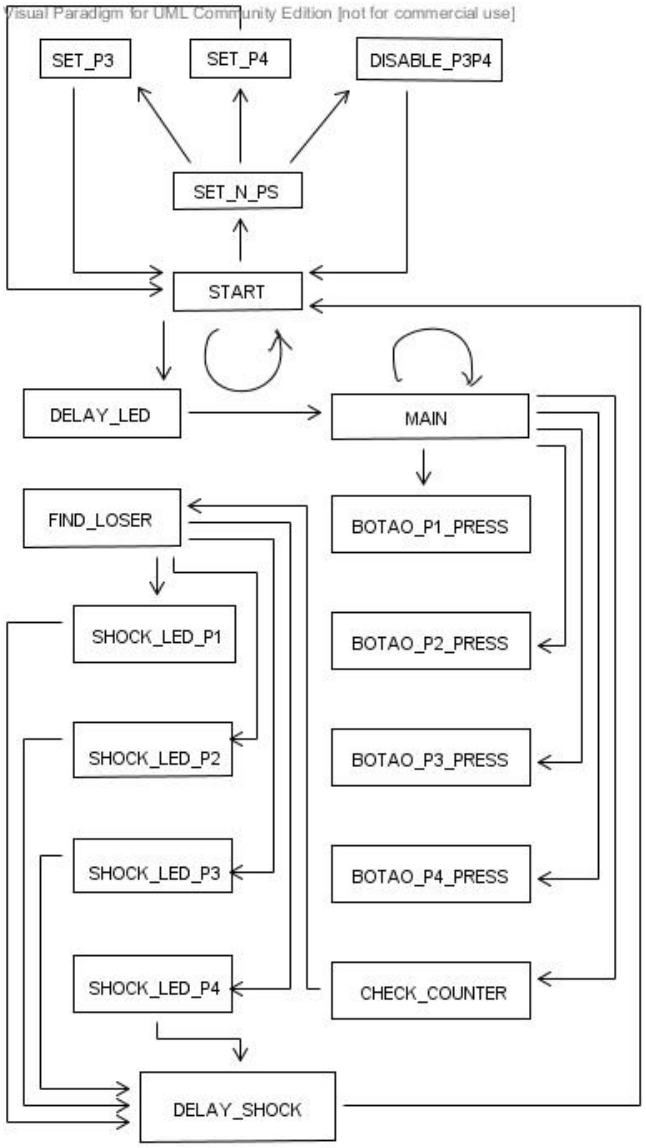

Fig. 3. Fluxograma de rotinas

como 0, 1 ou 2 para 4, 3 ou 2 jogadores, respectivamente. Isso se deve porque, logo que o contador atinge um valor igual a 3, o perdedor deve levar um choque.

Ao pressionar o botão de selecionar o número de jogadores, a rotina *SET_N_PS* é chamada onde os bits *ENABLED_LED_P3* e *ENABLED_LED_P4* são lidos, caso tenham valor lógico alto, significa que os jogadores 3 e 4, respectivamente, estão habilitados e a rotina *DISABLE_P3P4* é chamada para desabilitá-los. Caso somente 2 estiverem habilitados, o jogador 3 é habilitado pela rotina *SET_P3* onde bit *ENABLED_LED_P3* é setado para alto. Em último caso, se houverem 3 habilitados, a rotina *SET_P4* é chamada para setar como 1 o bit *ENABLED_LED_P4*.

No começo de uma partida, o programa passa a operar no delay da rotina *DELAY_LED*. Sua função é manter *START_LED* setado até o momento de limpá-lo e os botões do jogo serem apertados, porém também checa constantemente, na sua "subrotina" *DELAY2_LED*, as entradas dos botões do jogo para punir com um choque, através das rotinas *SHOCK_P1*, *SHOCK_P2*, *SHOCK_P3* e *SHOCK_P4*, o jogador que apertar o botão antes da hora certa e, em seguida, esperar o início de uma nova partida.

Quando o *START_LED* apagar, o programa estará rodando no loop da rotina *MAIN*, definido como momento de decisão, onde os botões do jogo são novamente checados. Caso algum seja apertado, alguma das rotinas *BOTAO_P1_PRESS*, *BOTAO_P2_PRESS*, *BOTAO_P3_PRESS* ou *BOTAO_P4_PRESS* são chamadas para, além de incrementar 1 no valor de *COUNTER*, setar o bit *BOT,0* , *BOT,1* , *BOT,2* ou *BOT,3* , respectivamente para cada jogador.

Durante o momento de decisão, os botões *BOTAO_P1*, *BOTAO_P2*, *BOTAO_P3* e *BOTAO_P4* passam a ser funcionais. No entanto, realmente funcionarão somente os botões daqueles jogadores que foram habilitados. Por exemplo, caso sejam somente dois oponentes, os botões *BOTAO_P3* e *BOTAO_P4* ainda funcionam como entrada, porém os seus estados de pressionado ou não já foram setados para o estado pressionado durante a selecão do número de jogadores, isto é, neste caso, *BOT,2* e *BOT,3* estão setados como 1 fazendo-os indeferentes neste momento.

Ademais, a variável *COUNTER* já foi incrementada em 1 para cada jogador desabilitado ou vice-versa para cada jogador é reabilitado. Os valores de *BOT* e *COUNTER* são armazenados em *LAST_COUNTER* e *LAST_BOT*.

Ainda dentro do loop da *MAIN*, existe uma rotina chamada *CHECK_COUNTER*. Nesta rotina, a variável *COUNTER* é checada à procura de um valor igual a 3 para fazer a chamada da rotina *FIND_LOSER*.

A rotina *FIND_LOSER* procura pelo primeiro 0 entre os bits de *BOT,0* , *BOT,1* , *BOT,2* e *BOT,3* , nesta ordem, para setar algum dos bits de *SHOCK_LED_P1*, SHOCK_LED_P2, SHOCK_LED_P3 e SHOCK_LED_P4 como 1 simbolizando o choque, respectivamente para o jogadores 1, 2, 3 e 4.

Em sequência, a rotina *DELAY_SHOCK* é chamada e delimita a duração do choque, limpa os bits *SHOCK_LED_P1*, *SHOCK_LED_P2*, *SHOCK_LED_P3* e *SHOCK_LED_P4*, carrega os valores de *LAST_COUNTER* e *LAST_BOT* em *COUNTER* e *BOT*, respectivamente, para recomeçar o jogo com a última configuração de jogadores.

IV. SIMULAÇAO

A simulaçao foi feita no *software* Proteus, o circuito lá montado buscava somente simular o funcionamento do projeto da forma mais simples e clara possíveis. Portanto, não foram adotados circuitos de proteção para o funcionamento idêntico ao do projeto de *hardware*.

A Figura 4 mostra o momento em que o número de jogadores podem ser selecionados. Os LED's verdes acesos indicam que aquele jogador está participando da partida. Por conseguinte, ao mudar o número de jogadores, os dois LED's da direita poderiam ser acesos ou apagados.

A Figura 5 mostra o LED vermelho aceso. Este LED fica aceso por alguns poucos segundos e apaga para os jogadores apertarem o botão. Se apertar antes da hora, acontece o processo da Figura 6.

Tabela I
VARIÁVEIS DE CONTROLE

Variáveis	Endereco	Descrição
COUNTER	0x20	Contador do número de botões apertados antes de detectar quem perdeu a partida
LAST_COUNTER	0x21	*Backup* da configuração de CONT
BOT	0x22	Campo de 8 bits responsável por memorizar o estado de apertado ou não durante o momento de decisão da partida
LAST_BOT	0x23	*Backup* da configuração de BOT
COUNT1	0x24	Contador 1 para o delay de duração do choque
COUNT2	0x25	Contador 2 para o delay de duração do choque
COUNT3	0x26	Contador 3 para o delay de duração do choque

Tabela II
BITS DE CONTROLE

Bits	Pino	Descrição
BOTAO_P1	PORTA,0	Botão para mudar o estado do jogador 1 durante o momento de decisão da partida
BOTAO_P2	PORTA,1	Botão para mudar o estado do jogador 2 durante o momento de decisão da partida
BOTAO_P3	PORTA,2	Botão para mudar o estado do jogador 3 durante o momento de decisão da partida
BOTAO_P4	PORTA,3	Botão para mudar o estado do jogador 4 durante o momento de decisão da partida
BOTAO_START	PORTA,4	Botão para iniciar a partida
BOTAO_N_PS	PORTA,5	Botão para escolher o número de jogadores da partida
SHOCK_LED_P1	PORTB,0	LED que simboliza o sinal de choque no jogador 1
SHOCK_LED_P2	PORTB,1	LED que simboliza o sinal de choque no jogador 2
SHOCK_LED_P3	PORTB,2	LED que simboliza o sinal de choque no jogador 3
SHOCK_LED_P4	PORTB,3	LED que simboliza o sinal de choque no jogador 4
ENABLED_LED_P3	PORTB,4	LED indicador de participação do jogador 3 na partida
ENABLED_LED_P4	PORTB,5	LED indicador de participação do jogador 4 na partida
START_LED	PORTB,6	LED que sinaliza o início do momento de decisão

A Figura 6 mostra o LED amarelo aceso. Este LED simboliza a saída para o circuito de dar o choque. No caso desta figura, o perdedor foi o jogador 2. Por conta disso, o LED acendeu.

V. DIFERENÇAS DO JOGO ORIGINAL

No jogo original, há algumas implementaçoes mais interativas com os usuários enquanto a partida ocorre, são descritas a seguir.

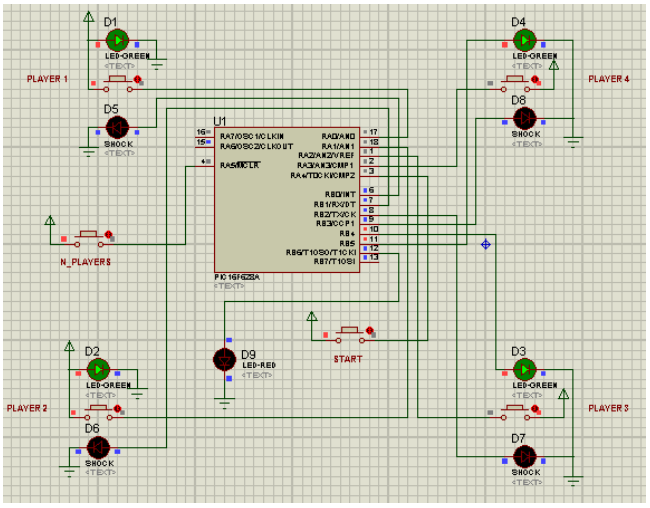

Fig. 4. Ilustração da simulaçao configurando para 4 jogadores e aguardando o inicio da partida

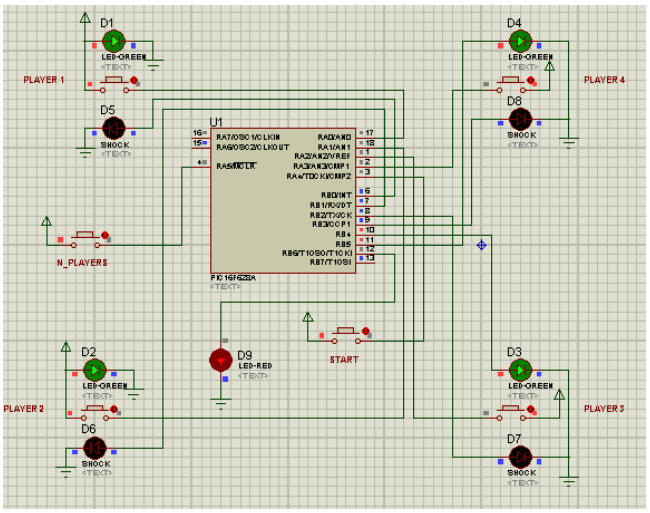

Fig. 5. Ilustração da simulaçao que os jogadores esperam o LED vermelho apagar para apertar o botão

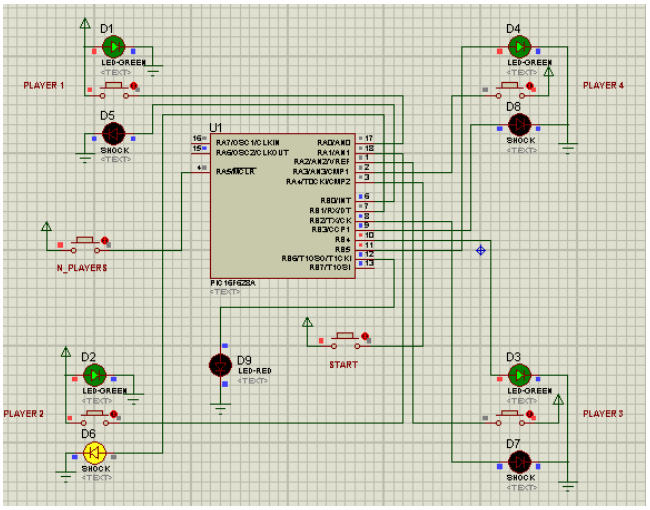

Fig. 6. Ilustração da simulaçao que mostra o jogador 2 levando um choque representado pelo LED amarelo aceso

Existe um *buzzer* que toca uma melodia característica para o jogo. Outro caso é que , diferentemente do LED que acende e apaga para sinalizar o momento de decisão, há um LED vermelho que fica piscando para indicar que foi dado início à partida e mais um outro com a mesma função do *START_LED*, porém ele é da cor verde e acende no momento de decisão.

VI. Conclusão

A idéia do projeto reflete uma simplicidade muito grande para a implementaçao, isto não deixa de ser verdade pensando em implementar em linguagens de níveis mais alto, como C e Java. Em Assembly, os conhecimentos do grupo em linguagem de máquina foram essenciais para a conclusão do projeto e o tempo de conclusão foi significativamente aumentado comparado a uma linguagem de alto nível.

O projeto funciona de uma forma mais básica em relação ao jogo original, porém realiza perfeitamente a mesma proposta de jogo para múltiplos jogadores, pois existe até mesmo uma memória para iniciar uma nova partida com as mesmas configuraçoes da partida anterior, tornando mais confortável a jogabilidade.

Referências

[1] http://pt.scribd.com/doc/20282565/Apostila-Microcontrolador-PIC-16F628