Projeto - Jogo Gênius

Gabriel David
Lucas Deolindo
Lucas Souza

BCC34C - SISTEMAS MICROCONTROLADOS
Profesor: Frank Helbert Borsato
20 de dezembro de 2018

Resumo

O presente relatório tem o objetivo de expor os dados da implementação do jogo Gênius em um microcontrolador. O relatório está dividido em quatro tópicos onde será apresentado a lista dos materiais utilizados para o desenvolvimento do projeto, uma explicação dos métodos utilizados e um apêndice no qual foi descrito os desafios enfrentados na implementação.

O primeiro tópico Introdução, tem como objetivo introduzir o tema ao leitor, apresentando uma breve explicação do funcionamento do projeto com as informações consideradas essenciais para o seu entendimento. Já o segundo tópico trata de expor os materiais necessários para a implementação do jogo, assim como os materiais e recursos incrementados para garantir uma melhor jogabilidade, e os requisitos mínimos apresentados pelo professor seguido do que foi feito para suprir esses requisitos. Seguindo para o terceiro tópico, temos uma explicação das funções e dos métodos utilizados para a implementação do jogo, e por último foram discorridos os desafios enfrentados pelo grupo na implementação, tanto na parte do software quanto do hardware.

1 Introdução

O jogo Gênius ficou conhecido no Brasil em 1980 quando uma empresa chamada Estrela o comercializou, todavia o mesmo foi inspirado em um jogo americano de nome Hasbro. Na implementação deste projeto, nos inspiramos no jogo da empresa Estrela, que contém as seguintes regras básicas:

- ✓ O jogo contém quatro LEDs de cores distintas, e a mesma quantidade de botões, um para cada LED;
- ✓ O jogo inicia com o piscar de um LED*, e o jogador deverá apertar o botão correspondente ao LED aceso;

- ✓ Se o jogador acertar, a sequência que antecede o acerto será repetida e um novo LED aleatório irá piscar;*
- ✓ Caso o jogador erre, os quatro LEDs irão aceder juntos e a sequência será zerada, sinalizando o fim do jogo.*

Nosso jogo irá implementar as mesmas regras, mas com alguns incrementos. Ele terá um método de pontuação em tempo real do seu jogo que será implementado com display de 7 segmentos, para que o jogador possa ver sua pontuação enquanto joga.

Além disso, o jogo possuirá um método de pontuação geral e níveis de dificuldade, quanto o nível de dificuldade, mais longe você for e em menor tempo fizer, seu ranking será considerado maior na tabela de jogadores. Seu nível poderá ser escolhido pelo jogador através do computador, e quanto maior o nível, maior será o número de LEDs acendendo em cada sequência. Para que seja possível visualizar este ranking de jogadores, iremos guardar todos estes dados na memória EEPROM e o jogador poderá vê-los na tela do computador. E por último, para facilitar a jogabilidade, iremos inserir sons para início e término do jogo.

2 Componentes necessários

Para a implementação e total funcionamento do jogo, foi necessário a utilização dos seguintes componentes:

- ✓ 4 LEDs, azul, vermelho, verde e amarelo;
- ✓ 4 botões, que correspondem a cada LED;
- ✓ 1 display 7 segmentos com multiplexador;
- ✓ 1 Arduino;
- ✓ 1 alto-falante;
- ✓ Resistores;
- ✓ Fios;
- ✓ Protoboard;
- ✓ 1 Relógio de tempo real.

2.1 LED's e Botões

Como citado a cima, foram utilizados 4 LED's nas cores azul, verde, amarelo e vermelho, cada um ligado a um resistor de 1000 Ohms de resistência, e 4 botões com as cores na mesma sequência, cada um representando um LED. Por falta de portas no Arduino, foi necessário utilizar as mesmas portas para controlar os botões e os LED's, o botão azul utiliza a mesma porta do LED azul, o botão vermelho a mesma porta do LED vermelho, e assim sucessivamente. Com a entrada do LED ligado na mesma trilha de saída do botão correspondente, e assim ligado em sua porta no Arduino.

2.2 Display de 7 segmentos

O display de 7 segmentos utilizado na implementação contém 4 conjuntos de 7 segmentos, com 8 portas para os segmentos e 4 portas para chaveamento de contexto, pois o componente possui um multiplexador pra facilitar a troca de contexto. Os resistores de 1000 Ohms também foram necessários nesse componente, um para cada segmento.

2.3 Relógio de tempo real

Para implementar o placar de líderes no jogo, foi necessário o uso do relógio de tempo real, onde conseguimos obter a data atual da jogada a partir de uma configuração inicial no código, que será explicado mais a frente. Vale frisar que as portas do microcontrolador utilizadas para o relógio em questão são as portas A4 e A5, não sendo possível utilizar outras portas, pois o componente usado só aceita as duas portas ditas.

2.4 Alto-Falante

Um importante componente que garante uma boa jogabilidade do Gênius, é o buzzer (alto-falante) que emite sons em determinados momentos do jogo. O mesmo foi implementado no código com sua porta de saída A3 no Arduino, mas na prática o componente fornecido não teve um bom funcionamento, pois necessita de muita energia, tornando seu som muito baixo durante a jogabilidade. Portanto o mesmo não foi implementado no hardware.

3 Métodos de Implementação

As bibliotecas necessárias para o uso dos métodos e total funcionamento do código estão listadas a baixo:

- ✓ avr/io.h: Esta é uma das mais utilizadas bibliotecas do Arduino, ela controla tudo referente a entrada e saída como LEDs e botões;
- ✓ util/delay.h: Para que fosse possível utilizar delays, fez-se necessário a utilização desta biblioteca, ela fornece funções como _delay_ms(), que geram um delay em mili segundos passados para ela;

- ✓ DS1307.h: Esta foi a biblioteca utilizada para trabalhar com o relógio de tempo real;
- ✓ EEPROM.h: Esta foi utilizada para a criação de funções que trabalhassem em leitura e escrita de inteiros e strings na EEPROM;
- ✓ string.h: e por ultimo a biblioteca para facilitar o manuseio de strings foi utilizada.

Para melhor manipulação dos LED's, foram definidas algumas funções pra chaveamento de contexto e algumas variáveis globais representando as portas do microcontrolador. Além delas temos algumas variáveis globais que serão explicadas conforme a aprofundação na explicação do código. Para iniciar a analise do código, este artigo focará em explicar todas as funções criadas, após isto serão listadas todas as interrupções necessárias e por ultimo serão expostos o setup e o loop deste código, onde o algoritmo fica na maior parte do tempo.

A Primeira das funções analisadas será a `rando()`, ela possui duas utilidades básicas, gerar a sequência aleatória de LED's e acender eles para que o jogador possa ver a mesma. Para gerar a sequencia aleatória uma das variáveis globais é utilizada a variável `sequencia`, ela guarda os LED's das sequencias passadas e adiciona os da nova sequencia, fazendo assim que a cada novo acerto do usuário a dificuldade aumente. Além disso é aqui também auxilia no nível do jogo, pois, dependendo do nível selecionado pelo jogador o número de sequencia passada por rodada aumenta.

Outro método deste algoritmo é a função `tocar()`, cada nota possui uma frequência, nosso código possui duas musicas que são mapas de frequências harmônicas e para que este método funcione você deve passar o mapa que deseja passar como parâmetro e a quantidade de tempo que você deseja que a musica toque.

A Próxima das funções é a `acende()`, para facilitar o manuseio do display de 7 segmentos esta função recebe um numero inteiro de 0 a 9 e como retorno acende o equivalente ao numero passado no display. É importante ressaltar que ao passar o número para esta função ela apenas aciona as portas correspondentes aos segmentos do display, e não faz o chaveamento dos mesmo, este chaveamento é feito na interrupção por tempo que ainda será abordada.

A seguir temos a `readName()`, ela realiza 3 grandes tarefas, a primeira dela é captar o nome do jogador através de uma função denominada `Serial.readBytes`, que capta oque foi informado pelo jogador até encontrar um carácter especifico, outra função é captar o nível que o jogador deseja jogar, se ele escolher um nível não permitido pela maquina ela irá setar o nível 1 como padrão e por ultimo esta função contem a `seed` que a função `rando` já explicada utiliza para iniciar suas sequencias, foi decidido colocar ela neste ponto do código pois aqui tempos diferentes com certeza serão passados nas escolhas de nome e nível e por isso ficará quase impossível acertar a mesma `seed`.

E por ultimo em relação as funções temos 4 delas responsáveis exclusivamente para o controle de leitura e escrita da EEPROM, seus nomes são `escreveInt()`, `lerInt()`, `escreveString()`, `leString()`. Elas são bem auto explicativas, todavia utilizando

o auxilio da biblioteca `EEPROM.h` elas conseguem respectivamente escrever inteiros na EEPROM, ler inteiros, escrever strings e ler strings na mesma.

Agora falando de interrupções dois tipos foram utilizados neste projeto, uma interrupção por tempo e outra pelas portas do arduino. A interrupção por tempo tem como propósito realizar o chaveamento de contexto do display de 7 segmentos, a cada 16 ms ela é chamada, imprime um dos blocos do display e passa para o próximo, realizando esse chaveamento a cada 16 ms os olhos humanos não conseguem identificar a diferença de algo ligado para algo desligado e tudo parece fluido. Além disso temos interrupções por portas do arduino, elas são realizadas toda vez que há mudanças nas portas em que os botões estão conectados, um grande problema encontrado neste tipo de abordagem foi que como os LEDs estão ligados lógicamente nas mesmas portas que os botões, acender ou apagar um LED é sempre necessario desligar está interrupção para que um looping infinito não seja criado.

E por ultimo temos as duas funções onde o códio realmente executa que são o `setup()` e o `loop()`. N função de `setup()` todas as dependências deste algoritmo são inicializados, incluindo nelas as portas dos Arduinos as setando como entrada e saída quando necessário, iniciando o auxiliador de comunicação via porta serial, iniciando o relógio de tempo real utilizando o rtc e qualquer outra coisa que precise ser setada antes da inicialização real do código. já na `loop()` é onde o código fica preso e em um laço infinito, a primeira coisa vista neste trecho de código é uma chamada para a função `readName()` já explicada neste artigo, que tem como propósito fazer com que o jogador se identifique e escolha seu nível, após o usuário se identificar a função `toca()` é chamada passando como parâmetro uma das musicas, e depois do término da mesma as interrupções são ligadas. Depois dos preparos iniciais todos realizados o jogo verdadeiramente inicia, a lógica dele consiste em um laço de repetição maior que prende o jogador e um menor, ao entrar na primeira vez no laço maior uma sequencia é passada entrando na função `rando()` e até que o usuário realize sua jogada ele fica preso dentro do laço interno, após realiza-la ele é solto e se ele tiver acertado uma nova sequencia é dada, se não ele entra na areá de fim de jogo, todos os dados são salvos na EEPROM, o placar dos jogadores antigos são mostrados na tela e o jogo finalmente acaba.

4 Apêndice

O projeto do jogo Gênius, um jogo da memoria com Arduíno, foi um trabalho bastante interessante e desafiador, que envolveu o uso de diversos componentes.

O maior problema encontrado, foi a fato do Arduíno utilizado ter poucas portas, o que acabou gerando dificuldade na implementação dos componentes, pelo fato de ter usado 4 LEDs, 4 BOTÕES e 1 display de 7 segmentos oque consome muitas portas. Uma das solução encontradas para sanar essa falta de portas, foi a implementação da entrada do LED na saída dos BOTÕES, o que acabou liberando algumas delas. Outro agravante para a falta das portas, foi o display de 7 segmentos que utilizou nove portas.

E ainda as portas 0 e 1 do Arduíno não poderem serem utilizadas, por serem portas de entrada e saída de dados seriais.

Algumas outras dificuldades encontradas, foram na falta de equipamentos necessário para a implementação,e para acabar com estes problemas algumas soluções necessárias foram improvisações: na falta de um buzzer foi utilizado um auto-falante, mas implicou em experiencias ruins pelo fato do mesmo ser baixo e ainda desabilitava algumas portas, e ainda consumia mais energia que o Arduíno podia suportar. Uma das soluções encontrada foi utilizar um fone de ouvido para testar o som; e ainda foi necessário pedir emprestado para os colegas alguns componentes para que fosse possível a implementação de LEDs e botões por exemplo. E ainda por não poder levar alguns equipamentos para casa ou poder ficar, para poder testar em momentos fora da aula tivemos que nos focar apenas em trabalhar em sala, onde todos estavam reunidos e com todos os materiais em mãos.

5 Referências

http://mundoprojetado.com.br/tocando-musica-com-o-arduino-buzzer/

http://labdegaragem.com/forum/topics/modulo-rtc-ds1307

https://www.arduino.cc/reference/pt/language/functions/random-numbers/randomseed/

http://labdegaragem.com/m/blogpost?id=6223006%3ABlogPost%3A142550

https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/