UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA

ESCOLA SUPERIOR DE TECNOLOGIA

ENGENHARIA DE COMPUTAÇÃO

VICTOR KALEB LEITE GOMES

DESENVOLVIMENTO DE UM JOGO DE MEMORIZAÇÃO LUMINOSA NA PLATAFORMA ARDUINO

Manaus

2011

VICTOR KALEB LEITE GOMES

DESENVOLVIMENTO DE UM JOGO DE MEMORIZAÇÃO LUMINOSA NA PLATAFORMA ARDUINO

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Computação.

Orientador: Prof. M.Sc. Jucimar Maia da Silva Júnior

Manaus

2011

Universidade do Estado do Amazonas - UEA
Escola Superior de Tecnologia - EST

Reitor:
José Aldemir de Oliveira
Vice-Reitor:
Marly Guimarães Fernandes Costa
Diretor da Escola Superior de Tecnologia:
Mário Augusto Bessa de Figueirêdo
Coordenador do Curso de Engenharia de Computação:
Danielle Gordiano Valente
Coordenador da Disciplina Projeto Final:
Raimundo Correa de Oliveira

Banca Avaliadora composta por: *Data da Defesa: 15/12/2011.*
Prof. M.Sc. Jucimar Maia da Silva Júnior *(Orientador)*
Prof. M.Sc. Raimundo Correa de Oliveira
Prof. M.Sc. Ricardo da Silva Barbosa

CIP - Catalogação na Publicação

G633d Gomes, Victor Kaleb Leite

DESENVOLVIMENTO DE UM JOGO DE MEMORIZAÇÃO LUMINOSA NA PLATAFORMA ARDUINO / Victor Kaleb Leite Gomes; [orientado por] Prof. MSc. Jucimar Maia da Silva Júnior - Manaus: UEA, 2010.

43 p.: il.; 30cm

Inclui Bibliografia

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação). Universidade do Estado do Amazonas, 2011.

CDU: 004

VICTOR KALEB LEITE GOMES

DESENVOLVIMENTO DE UM JOGO DE MEMORIZAÇÃO LUMINOSA NA PLATAFORMA ARDUINO

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Computação.

Aprovado em: 15/12/2011

BANCA EXAMINADORA

Prof. Jucimar Maia da Silva Júnior, Mestre
UNIVERSIDADE DO ESTADO DO AMAZONAS

Prof. Raimundo Correa de Oliveira, Mestre
UNIVERSIDADE DO ESTADO DO AMAZONAS

Prof. Ricardo da Silva Barbosa, Mestre
UNIVERSIDADE DO ESTADO DO AMAZONAS

Agradecimentos

Aos meu pai Ednilson Gomes,minha mãe Francisca Leite, a minha irmã Bruna Leite, toda minha família, também aos amigos João Guilherme, Clarice Santos, Yasminie, George Nunes,Emiliano e Bruno Mendes que, com muito apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

Ao professor orientador Jucimar Maia pelo grande apoio, incentivo e muita paciência, que me levaram à execução e conclusão desta Monografia.

Resumo

Este trabalho descreve o desenvolvimento de um jogo de memorização baseada no Genius Simon,jogo de memorização famoso na decada de 80. O jogo utiliza-se dos recursos da plataforma arduino para se tornar funcional, tem como objetivo a memorização da sequência luminosa gerada aleatoriamente toda vez que o jogo se inicia.O produto consiste na geração de uma sequência luminosa, que sera repitida pelo seu usuario ate o numero de vezes correta, para seu fim.

Abstract

This study describes the development of a memory game based on the Genius Simon, a famous game in the 80's. The game makes use of the resources Arduino prototyping platformaims to a bright memory of the sequence generated randomly each time the game starts. The game generates a sequence of light, which would be repeated by your player to the correct number of times until the end.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Códigos

Capítulo 1

Introdução

1.1 Introdução

Sistemas embarcados são sistemas microprocessados, no qual o computador é completamente encapsulado ou dedicado ao dispositivo controlado por ele. Diferentemente dos computadores de propósito geral, este realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos. Devido a sua especificidade, através de engenharia pode-se otimizar o projeto reduzindo tamanho, recursos computacionais e custo do produto.Com a grande crescente de automatização dos produtos, a necessidade de se entender e pesquisar este ramo da engenharia ficou mais eminente, surge então a necessidade de maiores atuações nesta área.Com uma utilidade específica escolheu-se a aplicação do conhecimento e o estudo embarcados a um jogo de memorização luminosa.

Segundo [McRoberts2011],em seu livro Arduino Basico, o Arduino Project teve início em 2005, mais de 150.000 placas Arduino foram vendidas em todo o mundo. O número de placas-clone não oficiais sem dúvida supera o de placas oficiais, assim, é provável que mais de 500 mil placas Arduino e suas variantes tenham sido vendidas. Sua popularidade não para de crescer, e cada vez mais pessoas percebem o incrível potencial desse maravilhoso projeto de fonte aberta para criar projetos interessantes rápida e facilmente, com uma curva de aprendizagem relativamente pequena.

O que propicia um boa estrutura para desenvolvimento, ferramenta com comunidade forte e com muitas estruturas para construções em desenvolvimento.

1.2 Justificativa

Os sistemas embarcados uma das formas mais presentes de controle eletrônico em aparelhos eletroeletrônicos e brinquedo inteligentes, provavelmente devido ao baixo custo do material e também ao reduzido custo energético, tornou-se objeto de estudo devido a sua procura de mercado. Este trabalho utiliza-se de uma plataforma de desenvolvimento arduino, que é estruturada de forma a facilitar o desenvolvimentode de novas tecnologias embarcadas. Visando este desenvolvimento de sistemas embarcados cria-se um jogo de memorização, aos moldes embarcados de forma a aplicar este conhecimento.

1.3 Objetivo

Desenvolver uma jogo de memorização na plataforma arduino.

1.3.1 Objetivos Específicos

- Gerar sequência e métodos de comparação;
- Estruturar emissão de comandos;
- Estruturar para recepção de comandos;
- Montar do circuito das luzes(interface de saída);
- Montar circuito dos botões(interface de entrada);

1.4 Metodologia

Inicialmente para o desenvolvimento utiliza-se da plataforma arduino para programação das sequências luminosas em seu microcontrolador. No mesmo microcontrolador se faz a

programação para entrada e saída de dados para interagir com o hardware da plataforma de forma a reconhecer os comando lógicos de entrada e ser capaz de emitir saída elétrica responsiva nos leds. Depois de realizada a programação, será feita a construção dos circuitos que servirão de interface para usuário do jogo. No caso da saída, o circuito com os quatros leds coloridos, funcionam como a interface de saída.Ao que diz repeito a entrada de dados, esta serão feitas a partir de interceptação de voltagem com botões. E após desenvolvida a parte funcional será necessária uma caixa protetora de forma a facilitar interação do jogador.

Capítulo 2

Arduino

2.1 Arduino

Arduino é uma plataforma de hardware programável, ao modelo de uma plataforma Wiring, open source composto de uma linguagem de programação, uma IDE (integrated development environment),e um único controlador.Criado com o objetivo de ser uma plataforma que facilite o desenvolvimento, componentes ligados ao um microcontrolador Atmel AVR. Trazendo a possibilidade de construção de controladores dos simples aos mais complexos, ressaltando também a capacidade de fácil integração com outros componentes, feitos para encaixar a arquitetura do Arduino conhecidos como shields.

O projeto do desta plataforma inciou-se em 2005, na cidade italiana de Ivrea,idealizado pelo quinteto Massimo Banzi, David Cuartales, Tom Igoe, Gianluca Martino com intuito de construir uma plataforma de prototipagem com custo menor que as existentes de forma a facilitar para trabalhos escolares. De 2005 a 2008 mais de 50 mil Arduinos foram vendidos, mostrando o reconhecimento da plataforma, que também foi lembrada pela Prix Ars Electronica com o recebimento de uma menção honrosa na categoria Comunidades Digitais em 2006.

Segundo [McRoberts2011] a maior vantagem do Arduino sobre outras plataformas de

desenvolvimento de microcontroladores é a facilidade de sua utilização; pessoas que não são da área técnica podem, rapidamente, aprender o básico e criar seus próprios projetos em um intervalo de tempo relativamente curto.

2.2 Hardware

Os Arduinos originais utilizam a série de chips megaAVR, especialmente os ATmega8, ATmega168, ATmega328 e a ATmega1280; porém muitos outros processadores foram utilizados por clones deles.

A grande maioria de placas inclui um regulador linear de 5 volts e um oscilador de cristal de 16 MHz Existem muitos modelos de Arduino que se aproximam quanto arquitetura mas possuem pequenas diferenças, mostrar-se-a o hardware do Arduino uno para exemplificar esta arquitetura.

2.3 Arduino Uno

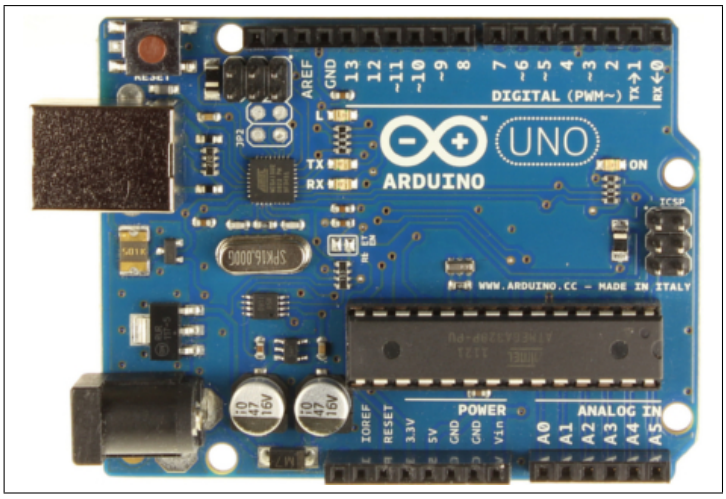

Figura 2.1: Arduino Uno Visão Frontal

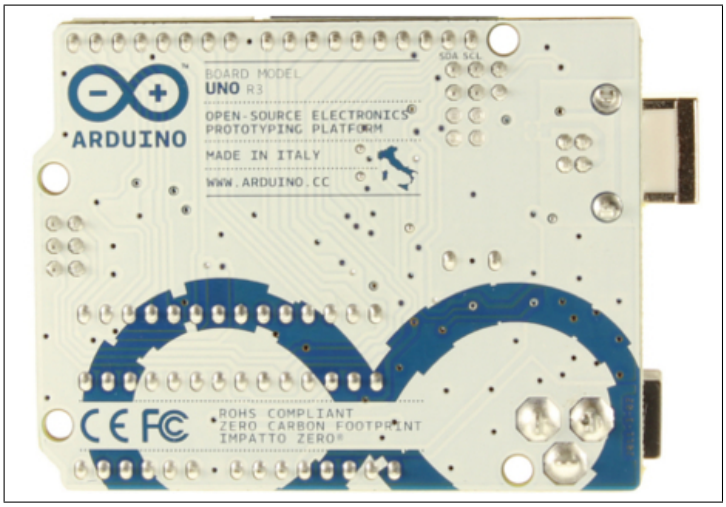

Figura 2.2: Arduino Uno Visão Traseira

O Arduino Uno trata-se de um dos dispositivos simples da família Arduino, sendo que ele recebe este nome por tratar-se de um hardware para iniciantes na plataforma, com baixo custo, mas que atende as muitas necessidades de desenvolvimento.

Este Arduino baseado no ATmega328 (datasheet), composto de 14 canais digitais de entrada/saída (sendo que destes 6 podem ser usadas como saídas PWN), 6 entradas

analógicas , a 16 MHz oscilador de cristal, um conexão USB , tomada de energia, um ICSP header, e um botão de reset. Contendo o necessário para o suporte ao microcontrolador.

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recomendado)	7-12V
Input Voltage (limites)	6-20V
Digital I/O Pins	14 (destes 6 geram saída PWM)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) com 0.5 KB para bootloader
SRAM	2 KB (ATmega328) .
EEPROM	1 KB (ATmega328) .
Clock Speed	16 MHz

Tabela 2.1: Especificação

O Arduino Uno pode ser alimentado pela conexão USB ou outro suprimento de energia externa sendo a fonte escolhida automaticamente.

Alimentação externa pode ser feita com um adaptador AC-to-DC (wall-wart) ou bateria. O adaptador pode ser conectado por conector a 2.1mm center-positive. Conduz a partir de uma bateria pode ser inserido nos pinos Gnd e Vin.

A placa pode operar com um fornecimento de energia externa de 6 a 20 volts. Se fornecido menos de 7V, entretanto, o pino de 5V pode fornecer menos de cinco volts deixando o sistema instável. Caso mais do que 12V, o regulador de voltagem pode ter um 'overheat' e danificar o sistema. Por isso é recomendado uma voltagem entre 7 e 12 volts.

Os Pinos para controle de alimentação são:

VIN. A voltagem de entrada para a placa do Arduino quando esta sendo usado para

fonte externa. Você pode alimentar por este pino ou se o fornecimento de tensão for feito pela tomada, acessá-lo deste pino.

5V. Fonte regulada usada para alimentar o microcontrolador e outros componentes . Podendo vir tanto do VIN como de outro regulador interno, ou fornecido pelo USB ou outro fornecimento regulado a 5V.

3V3. Um fornecimento 3.3 volt gerados pelo regulador on-board. Com corrente máxima de 50 mA.

GND. Aterramento.

2.4 Memória

O ATmega328 tem 32 KB (com 0.5 KB usado para 'bootloader'). Também tem 2 KB de SRAM e 1 KB de EEPROM (o qual pode ser ler e escrever com a biblioteca EEPROM).

2.5 Interface(Entrada e Saída)

Cada um dos 14 pinos digitais que podem ser usados tanto para saída quanto para entrada, usando as funções pinMode(), digitalWrite(), digitalRead(). Eles operam a 5 volts. Cada pino pode prover ou receber no máximo 40 mA e possui um resistor interno de pull-up de 20-50 kOhms. E como aditivo, alguns pinos tem funções especializadas:

Serial: 0 (RX) e 1 (TX). Usado para (RX) e transmitir (TX) TTL serial data. Esses pinos são conectados aos pinos correspondentes do ATmega8U2 USB-to-TTL Serial chip.

Interrupções externas: 2 e 3. Estes pinos podem ser configurados para disparar uma interrupção em baixo valor, para subida ou queda,ou uma mudança no valor. Por meio da

função attachInterrupt() .

PWM: 3, 5, 6, 9, 10 e 11. Fornece um saída de 8-bit PWM com a função analogWrite() .

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estes pinos dão suporte para comunicação SPI usando a biblioteca SPI.

LED: 13. Trata-se de um LED embutido ligado ao pino 13. Quando o pino está em Alto Valor, o LED fica ligado, quando baixo, este permanece desligado.

O Uno tem 6 entradas analógicas, listadas A0 à A5, cada um dos quais com 10 bits de resolução (i.e. 1024 valores diferentes). No 'default' eles medem do terra para 5 volts, embora seja possível mudar o limite superior de sua faixa usando o pino AREF e a função analogReference () . Em aditivo, alguns pinos tem funcionalidades especializadas:

TWI: pino A4 ou SDA, e o pino A5 ou SCL. Fornecem comunicação TWI usando a biblioteca Wire. AREF. Voltagem de referência. usado junto a função analogReference().

Reset. Utilizado para reiniciar o controlador. Tipicamente usado quando Shiels atrapalham a interação com o botão do arduino.

2.6 Comunicação

O Arduino tem inúmeras facilidades de comunicação com computadores, outro Arduino, ou outros microcontroladores. The ATmega328 fornece uma comunicação serial UART TTL (5V) , que se disponibiliza nos pinos 0 (RX) e 1 (TX). Um ATmega16U2 nos canais de comunicação com o USB cria uma porta virtual com software do computador(IDE). O '16U2 firmware se utiliza de drivers padrão USB COM, e sem a necessidade de um driver externo. Entretanto no Windows um arquivo inf. é necessário. O software inclui um monitoramento via serial que permite textos simplex sejam mandados da placa do Arduino.

Os LEDs RX e TX piscam quando os dados são transmitidos USB-to-serial pela conexão USB com o computador (mas não para a comunicação serial nos pinos 0 e 1).

2.7 Atmel AVR

AVR é um microcontrolador RISC de chip único com uma arquitetura Harvard modificada de 8-bit,desenvolvido pela Atmel em 1996. Foi um dos primeiros da família de microcontroladores a utilizar uma memória flash com o intuito de armazenar a programação, diferentemente de seus concorrentes da época, que utilizavam memórias do tipo PROM, EPROM ou EEPROM.

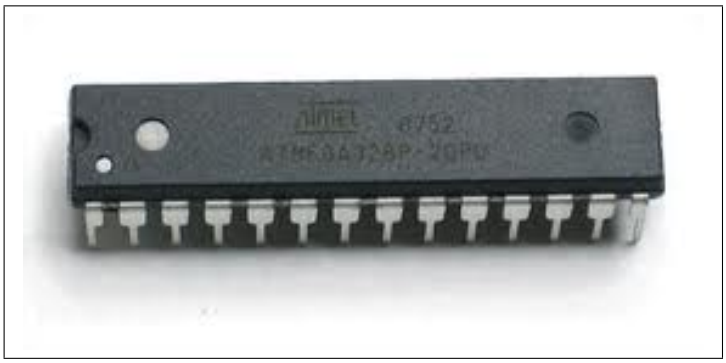

Figura 2.3: Atmel AVR328

Acreditava-se que sua arquitetura base foi conceituada por dois estudantes do Instituto Norueguês de Tecnologia (INT) Alf-Egil Bogen e Vegard Wollan.

Entretanto, ela foi desenvolvida por eles na Atmel Noruega, em Trondheim, onde, sendo eles dois fundadores da subsidiária naquele país, venderam a tecnologia para a Atmel como $\mu RISC (MicroRISC)$.

A empresa diz que o nome AVR não é um acrônimo e não tem nenhum significado em especial. Os criadores nunca deram uma resposta definitiva sobre o assunto.

A linha AT90S8515 está entre as primeiras fabricadas, na qual seu formato de

pacote DIP de 40 pinos teve a mesma pinagem que o 8051, incluindo endereços externos multiplexados e dados de controlador. A polaridade do RESET era invertida, mas fora isso, o restante da pinagem era idêntica.

A arquitetura Havard modificada do AVR tem o programa e os dados armazenados em sistemas de memória física separados, que aparecem em diferentes espaços de endereços, mas possui a habilidade de ler os itens da memória do programa usando instruções especiais.

2.7.1 Arquitetura

Flash, EEPROM e SRAM são todos integrados em um único chip, fora o necessário pelas memórias externas nas maiorias das aplicações. Algumas aplicações têm a opção de inserir um barramento paralelo externo para permitir memória para dados adicionais, código ou para mapeamento de dispositivos. Todos possuem interface seriais, que podem ser usadas para conectar EEPROMs seriais ou chips flash.

2.7.2 Memória(Flash)

As instruções de programas são armazenados em memórias flash não voláteis. Apesar de serem de 8 bits, cada instrução consiste de palavras de 16 bits, além disso, não há suporte de utilizar os códigos de programas externamente, todos as instruções devem residir no núcleo do dispositivo. O tamanho da memória do programa é normalmente indicado no nome do próprio dispositivo. Por exemplo, a linha ATmega64x tem 68 kbytes de flash, assim como a ATmega32x tem apenas 32kB. Os dados do espaço de endereçamento consistem de arquivos registradores, registradores de E/S e SRAM.

2.7.3 Registradores

Os AVRs têm 32 registradores de byte-único e são classificados como dispositivos de 8-bits RISC.

Em algumas variantes, os registradores em funcionamento são mapeados nos primeiros 32 endereços de memória (000016-001F16) seguidos por 64 registradores de E/S (002016-005F16). Atualmente, a SRAM inicia depois destas seções de registradores (endereço 006016). O espaço registrador de E/S pode ser ampliado em alguns dispositivos extensivos, no caso dos registradores de memória mapeada em E/S, uma parte do espaço de endereçamento SRAM será ocupada.

Ainda que haja separados esquemas e códigos otimizados para o arquivo registrador e para os acessos de E/S, tudo poderá continuar endereçado e manipulado como se estivesse na SRAM. Com exceção da XMEGA, na qual o funcionamento do arquivo registrador não é mapeado para o espaço de endereçamento de dados. Em vez disso, eles são mapeados para o endereçamento de começando logo no início dele. Consequentemente, a parte dedicada ao endereçamento para os registradores de E/S foram acrescidos para 4096 bytes (000016-0FFF16).

Como na geração anterior, no entanto, as instruções rápidas para manipular E/S apenas podem acessar as primeiras 64 posições dos registradores de E/S (sendo que as primeiras 32 posições são para instruções bitwise). E logo em seguida aos registradores de E/S, a séria XMEGA reserva uma faixa de 4096 bytes do espaço de endereçamento de dados que pode ser usado opcionalmente para mapear a EEPROM interna (100016-1FFF16). A SRAM real é localizada após essas faixas, começando em 200016.

2.7.4 EEPROM

Quase todos os microcontroladores AVR tem EEPROM interna para armazenamento de dados semi-permanente. Como memória flash, EEPROM pode mandar seu conteúdo quando a energia é desligada.

Na maioria das variantes da arquitetura AVR, esta memória EEPROM interna não é mapeada para o espaço de memória endereçável da MCU. Ela só pode ser acessada da mesma forma que um dispositivo periférico externo, usando registradores de ponteiros

especiais e instruções de leitura/escrita, o que faz o acesso à EEPROM muito mais lento que outra RAM interna.

Entretanto, alguns dispositivos na família SecureAVR (AT90SC) usam um mapeamento de EEPROM especial para a memória de dados ou texto (programa) dependendo da configuração. A família XMEGA também permite que a EEPROM seja mapeada para o espaço de endereçamento de dados.

Como o número de escritas na EEPROM não é ilimitado a Atmel especifica 100.000 ciclos de escrita uma boa rotina de escrita na EEPROM deve comparar o conteúdo do endereço da EEPROM a ser escrito com o valor desejado a ser gravado e fazer a escrita apenas se o conteúdo precisar ser modificado.

2.7.5 Open-source

Com a crescente popularidade dos FPGAs na comunidade open-source, começaram a desenvolver processadores open-source compatíveis com o conjunto de instruções AVR, devido a característica open-source e a boa documentação. O sítio OpenCores lista os maiores projetos clones do AVR a seguir:

pAVR, escrito em VHDL, é focada a criar o mais rápido e mais completo processador AVR, implementando técnicas não encontradas no processador AVR original, como uma pipeline mais profunda. core é um clone em VHDL que objetiva ser o mais próximo possível ao ATmega103.

Navré é escrito em Verilog, implementa todo o núcleo clássico do conjunto de instruções AVR e é voltado para alta performance e baixo consumo. Não suporta interrupções.

2.7.6 Plataforma Ideal

Ao longo deste capítulo, foi possível, entender algumas das capacidades desta platoforma, uma plataforma atual com muitas possibilidades para desenvolvimento, hardware atual e bem especificado e aberto o que permite o estudo mais aprofundado.Verificou-se um hardware confiável e ideal para o desenvolvimento de estruturas automatizadas.

Capítulo 3

JOGOS DE MEMÓRIA

3.1 Memória

A memória é a capacidade de adquirir (aquisição), armazenar (consolidação) e recuperar (evocar) informações disponíveis, seja internamente, no cérebro (memória biológica), seja externamente, em dispositivos artificiais (memória artificial).

A memória focaliza coisas específicas, requer grande quantidade de energia mental e deteriora-se com a idade. É um processo que conecta pedaços de memória e conhecimentos a fim de gerar novas idéias, ajudando a tomar decisões diárias.

Memória, segundo diversos estudiosos, é a base do conhecimento. Como tal, deve ser trabalhada e estimulada. É através dela que da-se significado ao cotidiano e acumulama-se experiências para utilizar durante a vida.

Segundo [Cardoso1996] a memória é um processo de retenção de informações no qual nossas experiências são arquivadas e recuperadas quando chamada. É uma função cerebral superior relacionada ao processo de retenção de informações obtidas em experiências vividas.

Memória declarativa. É a capacidade de verbalizar um fato. Classifica-se por sua vez em:

Memória imediata. É a memória que dura de frações a poucos segundos. Um exemplo é a capacidade de repetir imediatamente um número de telefone que é dito. Estes fatos são após um tempo completamente esquecidos, não deixando ”traços”.

Memória de curto prazo. É a memória com duração de alguns segundos ou minutos. Neste caso existe a formação de traços de memória. O período para a formação destes traços se chama de Período de consolidação. Um exemplo desta memória é a capacidade de lembrar eventos recentes que aconteceram nos últimos minutos.

Memória de longo prazo. É a memória com duração de dias, meses e anos. Um exemplo são as memórias do nome e idade de alguém quando se reencontra essa pessoa alguns dias depois. Como engloba um tempo muito grande pode ser diferenciada em alguns textos como memória de longuíssimo prazo quando envolve memória de muitos anos atrás.

Memória de procedimentos. É a capacidade de reter e processar informações que não podem ser verbalizadas, como tocar um instrumento ou andar de bicicleta. Ela é mais estável, mais difícil de ser perdida.

A memória também é fator importantíssimo para o cognição. Cognição é mais do que simplesmente a aquisição de conhecimento e consequentemente, a nossa melhor adaptação ao meio - é também um mecanismo de conversão do que é captado para o nosso modo de ser interno. Ela é um processo pelo qual o ser humano interage com os seus semelhantes e com o meio em que vive, sem perder a sua identidade existencial. Ela começa com a captação dos sentidos e logo em seguida ocorre a percepção. É portanto, um processo de conhecimento, que tem como material a informação do meio em que se vive e o que já está registrado na nossa memória.

3.1.1 Mnemotécnica

A mnemotécnica ou mnemônica (vocábulo derivado dos termos gregos que designam ”memória”e ”técnica”) consiste em uma série de procedimentos destinados a facilitar a lembrança de algo, recorrendo para isso à associação de outras ideias mais simples e frequentes que o tragam facilmente à memória. Para Sanz, um dos recursos mais frequentes para favorecer a retenção consiste em fazer listas de palavras encadeadas através de associações ilógicas, exageradas, de movimento, muito visuais e frequentemente grotescas.

As técnicas para a memória consistem basicamente em efetuar elos ou ganchos entre diferentes conceitos para que as lembranças surjam mais facilmente, e existem muitas escolas, sobretudo baseadas nas associações lógicas.

”De todo modo, não há receitas fixas. O importante é que cada pessoa efetue sua associação com a primeira coisa que lhe ocorra ou lhe venha à mente, que é o que sempre lembrará, e que além disso pratique os exercícios mnemotécnicos até incorporá-los a sua vida diária como algo automático”, assinala Sanz.

Os jogos de memórização,de um modo geral, como jogo de memória, genius , sudoko , de maneira geral,fazem uso mnemotécnica, o que os torna de grande valia para exercícios de memória.

3.2 Genius

Existem inúmeros jogos que trabalham a memória, como sudoku, jogo de memória, e direta ou indiretamente todos eles ajudam a exercitar e melhorar a nossa capacidade de memorização.

Genius um brinquedo muito popular na década de 1980 e que buscava estimular a memorização de cores e sons. Com um formato semelhante a um OVNI, possuía botões coloridos que emitiam sons harmônicos e se iluminavam em seqüência. Cabia aos jogadores repetir o processo sem errar.

O Genius original possuía três jogos diferentes:

Figura 3.1: Genius-simon

- Repita a seqüência
- Crie sua própria seqüência
- Repita a seqüência, agora apertando somente uma cor

Quatro níveis de dificuldade.

- 8 repetições
- 14 repetições
- 20 repetições
- 31 repetições

Atualmente é fabricado pela Estrela, que rebatizou o jogo com o nome de Genius Simon.

3.2.1 Histórico

O Genius lançado em 1980 pela Estrela foi o primeiro jogo eletrônico vendido no Brasil, era a versão do Simon, do fabricante americano Hasbro. Muitos brinquedos eletrônicos

da Estrela dos anos 80, como o Pégasus, Colossus, Gênius e outros, saíram de linha, frustrando milhares de crianças, hoje já crescidas, que não puderam tê-los na época.

Em 1987, a Prosoft desenvolveu um Genius para MSX 1, O programa foi desenvolvido em Basic.

Capítulo 4

Desenvolvimento

Neste capítulo será apresentada como foi feita a concepção do projeto, fluxograma que define o processo, figuras que exemplificam o projeto e código que desempenham procedimentos, chaves para o desenvolvimento do mesmo, e também a disposição do hardware para chegar a este resultado. A jogo foi desenvolvidada na plataforma arduino(hardware e software).

4.1 Material Utilizado

- 1 protoboard
- 1 led amarelo
- 1 led vermelho
- 1 led azul
- 1 led verde
- 5 botões
- 1 placa arduino UNO
- 1 cabo USB

- 1 computador com a interface do arduino para programação

4.2 Regras do Jogo

A regras adotadas para esse jogo são baseadas no funcionamento do genius. Este conjunto de regras descritas a seguir servirão de guia para construção do algorismo que será executado no microprocessador (ATmega328).

- Ganha o jogo ao se repetir corretamente a sequencia completa de luzes(8).
- Por meio dos botões pode-se acionar as luzes.
- Cada botão fica abaixo de sua luz correspondente.
- O sistema do jogo gera uma sequencia aletória.
- O sistema emite o sinal luminoso de acordo com a sequência aletória gerada, que irá ser emitida em subsequências.
- O jogador a aumenta passa a subsequência seguinte ao acertar a subsequência presente, ou seja acendendo as luzes na mesma ordem que o sistema do jogo emitiu por meio dos botões.
- Caso o jogador não repita a sequência correta o jogo volta a menor subsequência e uma nova sequência é gerada.
- Caso pressionado o botão de **reset** o jogo volta a menor subsequência e uma nova sequência é gerada.
- Ao repetir a sequência completa corretamente as quatro luzes acendem indicando o fim do jogo, e o jogo volta a menor subsequência e uma nova sequência é gerada.

  O fluxograma 4.1 representa graficamente parte das regras.

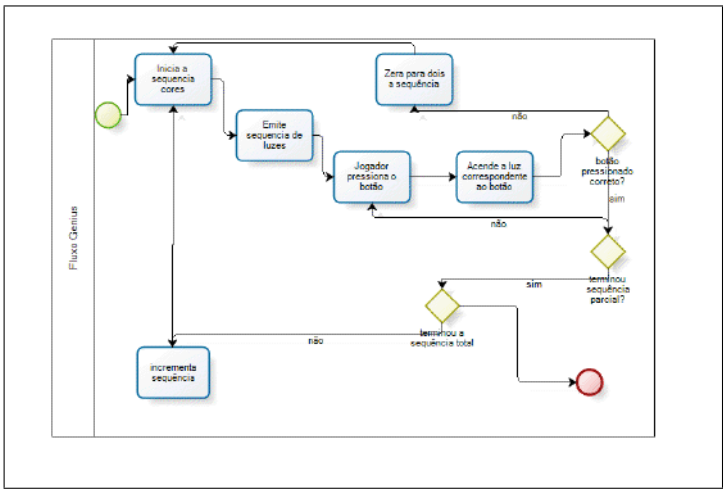

Figura 4.1: Fluxo do Jogo

4.3 Geração de sequência e controle da verificação

No Arduino utliza-se uma escrita na linguagem C para execução dos algoritmos. Aqui então cria-se a lógica do programa que irá controlar as regras do jogo.

O número aleatório é gerado a partir de uma leitura de uma porta analógica que gera um coportamento aleatório numa porta vazia, que irá ser base para a geração da sequência. utilizando o método *randomSeed()* Utilizando o *Random()* para gerar o método aleatório.

Esta parte do código representa como a lógica foi gerada.

```
1  Serial.begin(9600);
2  randomSeed(analogRead(6));
3  for(j=0;j<MaxSeq;j++)
4  {
5  seq[j]=((random(0,2000))%4)+1;
6  }
```

Código 4.3.1: *Geração Aleatória*

Nota-se que após gerada a semente e o número aleatório de 0 a 2000, é feita a operação de resto por quatro, para que se encontrem valores de 0 a 3 de forma a preencher o vetor com um equivalente a cor, no caso quatro cores, quatro numeros distintos (0,1,2,3).
Depois ao receber os dados de entrada ,este serão comparados de imediato a sequência que já está armazenada em um vetor. Em caso de acerto amplia-se a sequência luminosa, e em caso de erro inicia-se novamente o processo.

4.4 Emissão de comandos

São utilizadas portas digitais para se fazer a comunicação do Arduino, com o meio exterior, e por meio dessas com o controle de voltagem LOW e HIGH.

No caso deste projeto a resposta elétrica servirá para acender as quatro luzes coloridas:verde,vermelho,azul,amarela. Para esta saída utiliza-se. O código a seguir mostra um teste com a saída no método digitalWrite().

Este tem como efeito o teste a resposta das luzes a um sequência imposta de modo que este módulo tem a idéia de testar, a comunicabilidade do Arduino as portas de saida digital com interação com led.

```
1  int ledPinGreen =  13,ledPinRed =  12, int ledPinYellow =  11;
2  int red=LOW, yellow=LOW, green=LOW;
3  int seq[20]={1,2,1,3,1,2,3,2,3,3,1,2,1,2,3,2,1,1,1};
4  int i;
5  void setup()   {
6
7    pinMode(ledPinGreen, OUTPUT); // (pinMode  uma funcao que serve para configurar os pinos
8    pinMode(ledPinRed, OUTPUT);
9    pinMode(ledPinYellow, OUTPUT);//pode ser Saida(OUTPUT) ou uma Entrada(INPUT)
10
11 }
12
13 void loop()
14 {
15   red=yellow=green=LOW;
16
17   switch(seq[i]){
18    case 1:
19      digitalWrite(ledPinRed, HIGH);
20      break;
21    case 2:
22      digitalWrite(ledPinGreen, HIGH);
23      break;
24    case 3:
25      digitalWrite(ledPinYellow, HIGH);
26      break;
27   }
28   delay(1000);      // (tempo para o proximo comando)
29   digitalWrite(ledPinRed, LOW);
30   digitalWrite(ledPinGreen, LOW);
31   digitalWrite(ledPinYellow, LOW);
32   delay(1000);
33    if(i<20)
34     i++;
35 }
```

Código 4.4.1: *Comunicação LEDs*

4.5 Entrada de Dados

Neste projeto, a interação com processador ocorre da interrupção feita pela recepção HIGH e LOW feita nas portas de leitura.

Primeiro define-se na pinagem que determinado pino será para leitura. Após feito isso o mesmo se torna apto para leitura, sendo este agora sensível a entender uma ação externa, neste caso, oriunda do próprio Arduino. O controle é sinalizado por meio de uma alimentação de 5V originária da própria placa, que pode ser indentificada logicamente pela interrupção feita pelo botão ligado ao resistor.

```
1  void setup()   {
2    pinMode(buttonPinGreen, INPUT);
3    pinMode(buttonPinRed, INPUT);
4    pinMode(buttonPinYellow, INPUT);
5    pinMode(buttonPinBlue, INPUT);
6
7  }
8
9  void loop()
10 {
11   buttonStateGreen = digitalRead(buttonPinGreen);
12   buttonStateRed = digitalRead(buttonPinRed);
13   buttonStateYellow = digitalRead(buttonPinYellow);
14   buttonStateBlue = digitalRead(buttonPinBlue);
15 }
```

Código 4.5.1: *Entrada de Dados*

No código 4.5.1 é feita uma referência da parte do código que prepara o recebimento do dados. Configura-se as portas, para durante o loop poderem ser lidas, e com o valor de seus estados nas variáveis manipuladas.

4.6 Interface de Entrada e Saída

Os LEDs e os botões funcionam com a interface do jogo. Os LEDs que respondem diretamente a saída digital de escrita,ligado diretamente a saída(portas configuradas INPUT) de forma direta.

Um valor HIGH na porta imediatamente reflete no acender do LED a 5V. O que vale neste ponto a observação para construção da geração das cores, dado que o programa não permite que duas çuzes fiquem acesas ao mesmo tempo não ocorrerá uma queda de tensão

que ocasionaria no enfraquecimento da exceto, nos acertos na passagem de subsequências, e ao ganhar o jogo, que sinalizado com todas as luzes acesas.

Já ao que concerne a entrada de dados, é feita por meio dos botões, neste caso 5, 1 para cada cor de luz(amarelo,vermelho,azul,verde), e 1 para possibilitar o reinício do jogo. O programa está prepara-do para identificar a interrupção da voltagem que se recebe direto de sua própria fonte. O botão fecha o circuito, ao pressionar o botão (interrupção) se identifica na porta que não se recebe mais a voltagem e se reconhece logicamente o botão pressionado.

Como resultado das interações da interface do programa surge um produto em desenvolvimento de acordo com o esquema da figura 4.2.

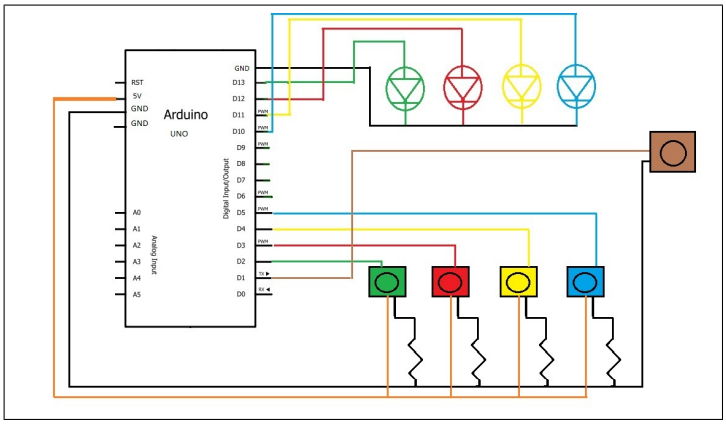

Figura 4.2: Circuito do Jogo

Capítulo 5
Conclusão

5.1 Conclusão

O desenvolvimento no Arduino se provou eficiente, trazendo facilidades para o desenvolvimento do jogo, o este pode ser concluído com êxito em seus módulos, quanto a geração do código para a sequência e lógica do jogo, quanto a entrada e saída de dados, foi facilitado pela estrutura do arduino que permitiu um fácil controle dos leds bem com acesso aos botões, que apesar de não responderem rapidamente conseguem manter o funcionamento do jogo.

Durante o desenvolvimento pensava-se em construir um produto, pronto para a venda, durante o percurso do projeto a entender os custos e possibilidades ficou bem claro que tratava-se de uma plataforma apenas para desenvolvimento.Verifica-se que hardware é necessário, e depois de testes com arduino se reproduziria um produto para venda aos moldes da estrutura.

Foi visível também durante a pesquisa e desenvolvimento a capacidade de expansão da a facilidade de interação pronta com os shields, bem como os elementos já incorparodos ao Arduino que trazem uma gama de possibilidades.

5.2 Trabalhos Futuros

- Construir caixa de proteção:

A construção de uma caixa de proteção, de forma a estabilizar o sistema, e criar uma

estrutura de interface amigável e de fácil utilização.

- Utilização de som:

  Implementação de emissão sonora com correspondência luz-som, que tornaria mais eficaz em relação a acessibilidade. Seria possível com a utilização de um speaker com controle de frequência.

- Reconstruir apenas com hardware restritamente necessário:

  Para que o produto adquira possibilidades de venda precisa ser reconstruido com apenas componentes necessários, como o Arduino trata-se de uma plataforma de desenvolvimento, tem muitos componentes que não foram utilizados para a construção do projeto, mesmo sendo barato para um sistema de prototipagem torna-se cara para mercado.

- Melhorar a resposta dos botões:

  A resposta do jogo é um pouco lenta em relação aos botões, a codificação é feita em loop e as verificações na repetição, deve-se aplicar-se de forma a parar o loop para captar as entradas.

Referências Bibliográficas

[Atm] Sítio oficial da atmel. [Online;http://www.atmel.com accessed 10-Outubro-2011].

[Ard2010] (2010). Arduino. [Online;http://www.arduino.cc accessed 12-Outubro-2011].

[Uno2011] (2011). *Arduino Uno Rev 3 Reference Manual*. Arduino.

[ATm2011] (2011). *ATmega48A/PA/88A/PA/168A/PA/328/P Reference*. Atmel.

[Cardoso1996] Cardoso, S. H. (1996). Memória: O que é e como melhorá-la.

[da Silveira2011] da Silveira, J. A. (2011). *Arduino Básico*. Ensino Profissional, first edition.

[de Faria2006] de Faria, T. H. D. (2006). Introdução a microcontroladores.

[McRoberts2011] McRoberts, M. (2011). *Arduino Básico*. Novatec, first edition.

[Wazlawick2009] Wazlawick, R. S. (2009). *Metodologia de Pesquisa para Ciência da Computação*. Elsevier, first edition.

[Wikipédia2011a] Wikipédia (2011a). Genius— wikipédia, a enciclopédia livre. [Online; accessed 28-Outubro-2011].

[Wikipédia2011b] Wikipédia (2011b). Mnemotécnica — wikipédia, a enciclopédia livre. [Online; accessed 28-Outubro-2011].

Capítulo 6

Apêndice

6.1 Código da Aplicação

```
1  int ledPinGreen =  13, ledPinYellow =  11, ledPinBlue =  10, fimseq=1,seq[20]; rep[20];i,j,a=0,fim=5;
2  const int buttonPinGreen = 2,buttonPinRed = 3, buttonPinYellow = 4;buttonPinBlue = 5;
3  int buttonStateRed,buttonStateGreen,buttonStateYellow,buttonStateBlue = 0;
4
5  void setup()
6  {
7    pinMode(ledPinGreen, OUTPUT);
8    pinMode(ledPinRed, OUTPUT);
9    pinMode(ledPinYellow, OUTPUT);
10   pinMode(ledPinBlue, OUTPUT);
11   pinMode(buttonPinGreen, INPUT);
12   pinMode(buttonPinRed, INPUT);
13   pinMode(buttonPinYellow, INPUT);
14   pinMode(buttonPinBlue, INPUT);
15   Serial.begin(9600);
16   randomSeed(analogRead(6));
17   for(j=0;j<6;j++)
18   {
19    seq[j]=((random(0,2000))%4)+1;
20    }
21 }
22
```

Código 6.1.1: *Código da aplicação - Declarações e Setup*

```
void mostrar_sequencia(){
  if(i<=fimseq)
     {
        switch(seq[i])
        {
         case 1:
           digitalWrite(ledPinGreen, HIGH);
           break;
         case 2:
           digitalWrite(ledPinRed, HIGH);
           break;
         case 3:
           digitalWrite(ledPinYellow, HIGH);
           break;
         case 4:
           digitalWrite(ledPinBlue, HIGH);
           break ;
         }
        i++;
     }
}
void acender(){
             if(a<=fimseq)
    {

      if (buttonStateGreen == HIGH)
      {
        digitalWrite(ledPinGreen, HIGH);
        rep[a]=1;
        if(seq[a]==rep[a])
          a++;
        else
        {
          a=i=0;
          fimseq=1;
        }
    }
    if (buttonStateRed == HIGH)
    {
    digitalWrite(ledPinRed, HIGH);
    rep[a]=2;
      if(seq[a]==rep[a])
        a++;
      else
      {
        a=i=0;
        fimseq=1;
      }
    }
    if (buttonStateYellow == HIGH)
    {
    digitalWrite(ledPinYellow, HIGH);
    rep[a]=3;
      if(seq[a]==rep[a])
        a++;
      else
      {
        a=i=0;
        fimseq=1;
      }
    }
    if (buttonStateBlue == HIGH)
     {
      digitalWrite(ledPinBlue, HIGH);
      rep[a]=4;
      if(seq[a]==rep[a])
        a++;
      else
      {
        a=i=0;
        fimseq=1;
      }
     }
  }

}
```

Código 6.1.2: *Código da aplicação - Métodos Auxiliares*

```
1  void loop()
2  {
3    buttonStateGreen = digitalRead(buttonPinGreen);
4    buttonStateRed = digitalRead(buttonPinRed);
5    buttonStateYellow = digitalRead(buttonPinYellow);
6    buttonStateBlue = digitalRead(buttonPinBlue);
7
8    mostrar_sequencia()
9    acender()
10
11   delay(1000);
12   digitalWrite(ledPinRed, LOW);
13   digitalWrite(ledPinGreen, LOW);
14   digitalWrite(ledPinYellow, LOW);
15   digitalWrite(ledPinBlue, LOW);
16   delay(300);
17   if(a>fimseq){
18     i=0;
19     a=0;
20     digitalWrite(ledPinRed, HIGH);
21     digitalWrite(ledPinGreen, HIGH);
22     digitalWrite(ledPinYellow, HIGH);
23     digitalWrite(ledPinBlue, HIGH);
24     delay(200);
25     digitalWrite(ledPinRed, LOW);
26     digitalWrite(ledPinGreen, LOW);
27     digitalWrite(ledPinYellow, LOW);
28     digitalWrite(ledPinBlue, LOW);
29     fimseq++;
30    }
31    if(fimseq==fim)
32   {
33     digitalWrite(ledPinRed, HIGH);
34     digitalWrite(ledPinGreen, HIGH);
35     digitalWrite(ledPinYellow, HIGH);
36     digitalWrite(ledPinBlue, HIGH);
37     delay(5000);
38     a=i=0;
39     fimseq=1;
40     setup();
41   }
42 }
```

Código 6.1.3: *Código da aplicação - Loop Principal*