UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

ENGENHARIA DE COMPUTAÇÃO

KARINA OGUIDO

MAURO IKEDA VIALICH

RÔMULO IANUCH SOUZA

JOGO SNAKE CONTROLADO POR ACELERÔMETRO

RELATÓRIO DE PROJETO

CURITIBA

2014

KARINA OGUIDO

MAURO IKEDA VIALICH

RÔMULO IANUCH SOUZA

JOGO SNAKE CONTROLADO POR ACELERÔMETRO

Relatório de projeto apresentado à disciplina de Oficina de Integração II, do Curso de Engenharia de Computação, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Juliano Mourão Vieira

CURITIBA

2014

Dedicamos este trabalho a todos os que nos apoiaram e que nos ajudaram em nosso projeto.

Aos professores que nos guiaram e aos familiares e amigos que nos apoiaram.

AGRADECIMENTOS

Agradecemos primordialmente aos professores Hugo Vieira Neto e Mário Sérgio Teixeira de Freitas por administrarem a disciplina de Oficina de Integração II com maestria e sabedoria. Um reconhecimento particular e indispensável ao professor Hugo Vieira Neto que nos auxiliou a concretizar nosso projeto e a desenvolver os conhecimentos absolutamente necessários para a realização do mesmo.

Agradecemos também ao nosso professor orientador Juliano Mourão Vieira por ter nos auxiliado na parte de programação do jogo, ajudando a resolver os problemas mais pontuais e específicos do projeto.

Para nós os grandes homens não são
aqueles que resolveram os problemas,
mas aqueles que os descobriram.

(SCHWEITZER, Albert)

RESUMO

OGUIDO, Karina; I. VIALICH, Mauro; I. SOUZA, Rômulo. **Jogo Snake controlado por acelerômetro**. 2014. 38 f.Relatório de projeto – Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

Este relatório descreve o projeto realizado, o qual se baseia na criação de um software interativo, o jogo clássico Snake, controlado por um sensor e representado graficamente por uma matriz de LEDs. A direção da cobra é controlada através dos movimentos feitos pelo usuário, os quais são detectados pelo acelerômetro, um sensor de aceleração. As informações coletadas pelo acelerômetro são recebidas pelo microcontrolador do Arduino, o qual representa graficamente a dinâmica do jogo em uma matriz de LEDs bicolor 8x8.

Palavras-chave: Snake. Acelerômetro. Arduino. Matriz de LEDs.

ABSTRACT

OGUIDO, Karina; I. VIALICH, Mauro; I. SOUZA, Rômulo. **Snake Game controlled by accelerometer**. 2014. 38 f. Project report - Federal Technology University - Parana. Curitiba, 2014.

This paper describes the accomplished project, which is based on an interactive software creation, the classic game Snake, controlled by a sensor and graphically represented in a LED matrix. The snake's direction is controlled by movements made by the user, that are detected by the accelerometer, an acceleration sensor. The information collected by the accelerometer is received by the Arduino's microcontroller, which represents graphically the game's dynamic in a LED matrix bicolor 8x8.

Keywords: Snake. Accelerometer. Arduino. LED matrix.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este projeto, de uma forma sucinta, baseia-se em um software interativo controlado através do movimento feito por um usuário e detectado por um sensor. Deste modo, o microcontrolador recebe e analisa as informações geradas por este sensor e as apresenta graficamente através de um dispositivo.

O sensor escolhido foi o acelerômetro, o qual fornecerá leituras em dois eixos: *x* e *y*. Tais informações serão interpretadas pelo software presente no microcontrolador de um Arduino Mega, o qual analisará os dados gerados para representá-los na interface gráfica, no caso, uma matriz de LEDs de tamanho 8x8 bicolor. Todo esse processo está representado na Figura 1, a qual ilustra como as partes dos dispositivos interagem entre si através de um diagrama de blocos.

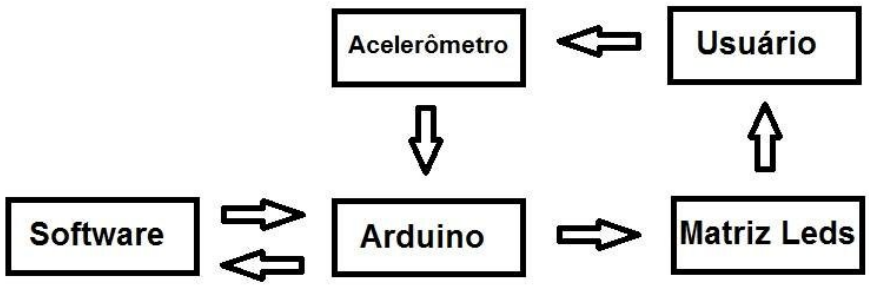

Figura 1 - Diagrama de blocos que ilustra a interação entre as partes do dispositivo
Fonte: Autoria própria

Para a exemplificação da interação entre os componentes citados, foi escolhido do clássico jogo “Snake”. O objetivo deste jogo é movimentar um personagem (uma cobra) pelo plano a fim de ingerir o maior número de maçãs, as quais surgirão no mapa de forma individual e aleatória. A Figura 2 ilustra a interface do jogo, sendo os quadrados pretos conectados o corpo da cobra e o outro quadrado isolado a maçã. Neste projeto o corpo da cobra será representado pelos LEDs verdes acesos da matriz de LEDs e a maçã será um único LED vermelho aceso.

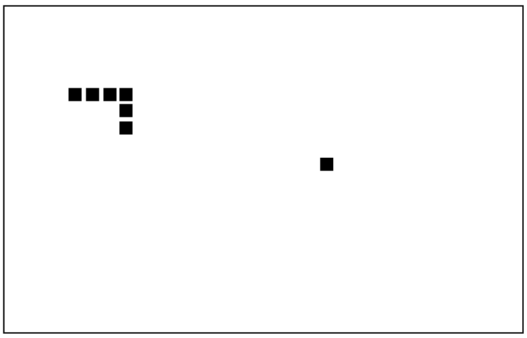

Figura 2 - Clássico jogo Snake
Fonte: Autoria própria

Como regra, a cobra não deve se chocar contra seu próprio corpo durante o decorrer do jogo, caso contrário, o jogo acaba. Quando a cobra encosta-se a um dos lados da tela, ela aparece no lado oposto. Ao ingerir uma maçã, o tamanho da cobra aumenta, isto é, um LED verde a mais é aceso ao final de seu corpo, o que dificulta cada vez mais o jogo. O movimento da cobra é contínuo, sendo que cabe ao usuário apenas orientar sua direção, através da movimentação do acelerômetro.

1.1 MOTIVAÇÃO

Apresentar uma aplicação interativa da união entre hardware e software. Essa aplicação é um projeto aparentemente complexo, cuja realização requer conhecimentos específicos de seus componentes, no entanto sua interação com um usuário sem esses conhecimentos se torna possível pelo fato do projeto ser um jogo apreciado por muitas pessoas e os comandos serem simples e facilmente compreendidos.

1.2 OBJETIVO GERAL

Criação de um dispositivo que possibilite o usuário a jogar o jogo Snake de forma mais interativa do que a forma convencional, isto é, em um display convencional e botões. O projeto faz o jogo Snake ter uma jogabilidade diferente e que o usuário se interesse mais em saber como o dispositivo funciona.

1.3 OBJETIVOS ESPECÍFICOS

- Entender o funcionamento de um Arduino e como programá-lo na sua linguagem Processing.
- Implementar o jogo Snake na linguagem Processing.
- Adquirir conhecimento de como a matriz de LEDs funciona.
- Desenvolvimento do software de acionamento dos LEDs.
- Compreensão do funcionamento do acelerômetro e como utilizá-lo.
- Integração dos três componentes citados, com cada um desempenhando seu papel corretamente.

2 CONTEÚDOS ENVOLVIDOS

Para a realização deste projeto é necessário, inicialmente, que haja um estudo sobre o funcionamento do microcontrolador e da placa à qual ele está acoplado, o Arduino. Além disso, também são exigidos os conhecimentos de como funciona o acelerômetro e a matriz de LEDs bicolor.

2.1 ARDUINO

O Arduino é uma placa de controle de entrada de dados, como sensores, e de saída de dados, como LEDs. Possui um cristal oscilador de 16 MHz, um regulador de tensão de 15 V, botão de reset, uma porta USB, que fornece alimentação enquanto estiver conectada ao computador, pinos conectores e alguns LEDs para auxiliar na visualização do funcionamento da placa [4].

Neste projeto o Arduino utilizado é o Mega R3 2560, que é baseado no ATmega2560, o qual possui 54 entradas e saídas digitais, 16 entradas analógicas, 4 UARTs (portas seriais de hardware), além das funções já citadas [4]. A escolha do Arduino facilitou o projeto devido aos requisitos disponíveis e a sua fácil manipulação, visto que a linguagem utilizada é a Processing, muito similar à linguagem C. Outro aspecto levado em conta na escolha do Arduino é a quantidade de portas digitais, visto que seriam necessárias 24 destas portas para o acionamento dos LEDs. Assim, o Arduino Mega foi escolhido para o desenvolvimento do projeto, pois possui 54 portas digitais (número superior aos outros modelos). Ele serve como uma ponte, sendo a entrada o acelerômetro e a saída a matriz de LEDs

2.2 ACELERÔMETRO

Para a obtenção dos movimentos do usuário foi utilizado um sensor de aceleração, o acelerômetro. Sua função é medir a taxa a que a velocidade de um objeto varia, isto é, sua aceleração. A saída de um acelerômetro é geralmente na forma de variação de tensão ou o deslocamento de um ponto móvel em uma escala fixa [2].

Para a execução do projeto, o acelerômetro escolhido foi o MMA7361, cuja saída é na forma de variação de tensão [3]. O acelerômetro MMA7361 é do tipo capacitivo e triaxial, ou seja, trabalha em 3 eixos. No entanto, para a realização do projeto, apenas os eixos X e Y são utilizados.

2.3 MATRIZ DE LEDS

Para a realização da representação gráfica deste projeto é necessário o conhecimento sobre diodos, como seu funcionamento e comportamento em um circuito, visto que serão utilizados LEDs (Light-emitting diode - diodo emissor de luz) [1].

O resultado do processamento das informações, feitas pelo microcontrolador, é apresentado em uma matriz de LEDs bicolor (vermelho e verde) modelo SZ012388K/9. Esta matriz é do tipo anodo comum, em que os anodos de cada LED são diretamente interligados, fazendo com que o controle do acendimento deles seja feito pelo acionamento do respectivo catodo, como é possível observar na Figura 3, a qual ilustra o diagrama esquemático desta matriz.

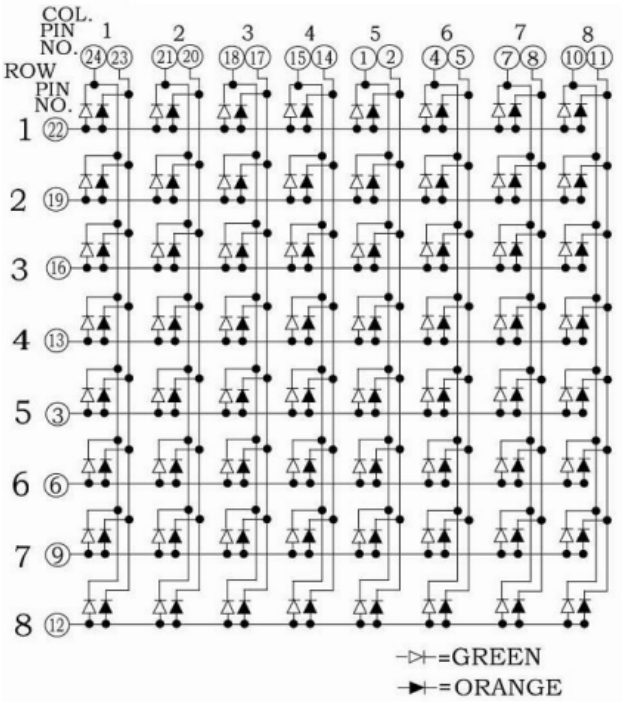

Figura 3 - Diagrama esquemático da matriz de LEDs
Fonte: Datasheet da matriz de LEDs SZ012388K/9 [6]

Por fim, para se realizar toda a montagem do que faz referência ao hardware do sistema, faz-se necessária uma base sobre funcionamento e montagem de um circuito eletrônico, o qual contém resistores e transístores conectados a protoboards. Além disso, é aconselhável um prévio conhecimento em linguagem C para facilitar a manipulação da linguagem Processing para o desenvolvimento do jogo e do tratamento de dados.

3 METODOLOGIA

Para a realização do projeto foi necessária a integração de hardware e software. Os circuitos utilizados, juntamente com seus diagramas esquemáticos, as funções empregadas nos algoritmos e a integração entre eles serão descritos detalhadamente nas próximas seções.

3.1 HARDWARE

O Arduino é a peça primordial do projeto, sendo responsável por controlar todo o funcionamento do dispositivo. Devido à sua praticidade, o Arduino permite uma fácil manipulação de dados e controle de acionamento de portas I/O. Outro aspecto positivo na utilização do Arduino é a sua linguagem padrão Processing, a qual se assemelha à linguagem C.

O Arduino recebe como entrada os valores de tensões obtidas do acelerômetro, cujas informações de orientação são obtidas pelos movimentos feitos pelo usuário, e a saída é a matriz de LEDs. O processo é mostrado na Figura 1, representado através de um diagrama de blocos.

O circuito interno do acelerômetro, quando ligado, faz com que as saídas X, Y e Z possuam tensões de acordo com o ângulo em que o dispositivo se encontra em relação à direção da aceleração da gravidade da Terra. Com o acelerômetro conectado ao Arduino, os valores de tensões referentes aos pinos de saída dos eixos (X_{OUT}, Y_{OUT} e Z_{OUT}, conforme a Figura 4) são convertidos, via conversor A/D presente no Arduino, em valores utilizáveis e precisos de acordo com o movimento do acelerômetro.

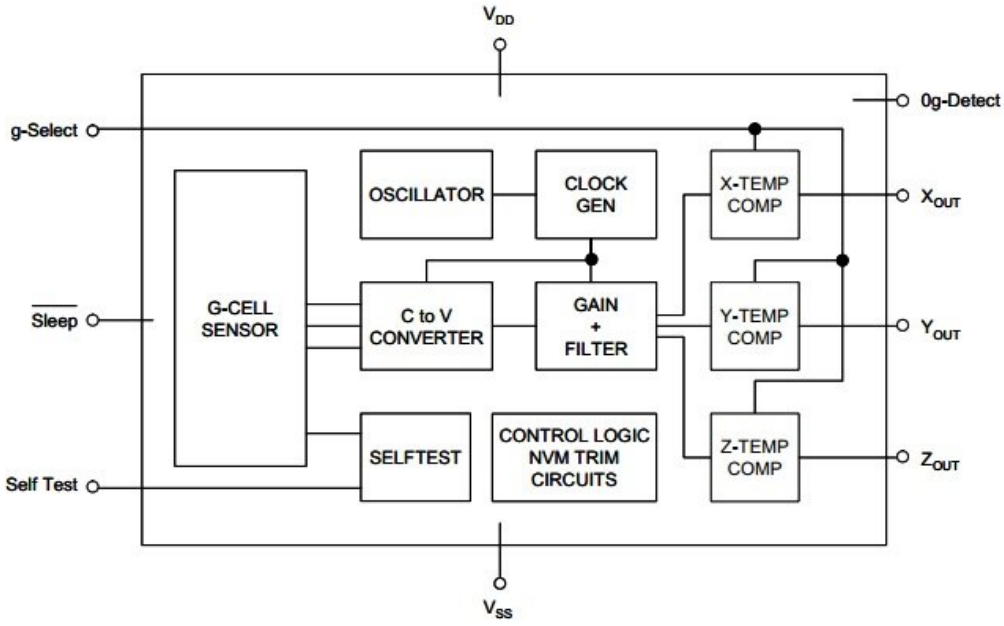

Figura 4 - Diagrama esquemático do acelerômetro MMA7361
Fonte: Datasheet do acelerômetro MMA7361 [3]

Após a análise das possibilidades e das circunstâncias do projeto, o método escolhido para fazer o acionamento dos LEDs é o de varredura por linha. Este método consiste em acionar uma linha de LEDs da matriz por vez, mas isto deve ser feito em uma frequência acima da percepção humana, como ocorre em lâmpadas e em televisões.

Outro ponto a ser observado é que os LEDs funcionam sob a tensão recomendada de 1,8 V para o LED vermelho e 2,1 V para o LED verde, sob uma corrente de 20 mA para o vermelho e 25 mA para o verde. O máximo brilho será atingido quando a corrente for máxima. [6]

O circuito para a solução do acionamento da matriz de LEDs consiste na utilização de 8 transístores BC328, 8 resistores de 330 Ω e 16 resistores de 220 Ω. Esses componentes são montados como mostra a Figura 5.

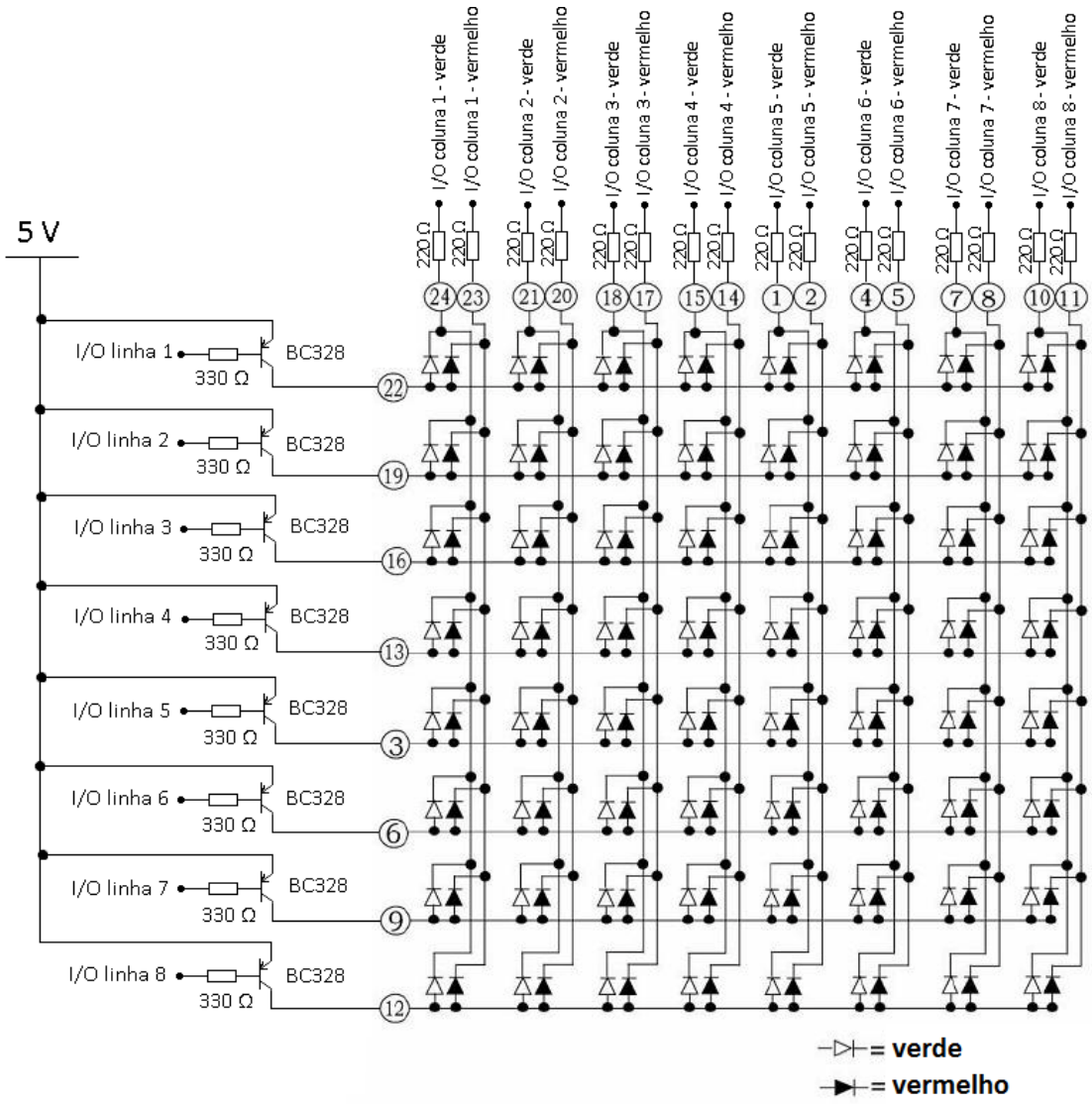

Figura 5 – Circuito do acionamento da matriz de LEDs
Fonte: Adaptada do datasheet da matriz de LEDs [6]

O transistor é um dispositivo semicondutor que amplifica e controla sinais elétricos (corrente). Existem basicamente dois tipos principais de transistores: o transistor de junção bipolar (BJT – Bipolar Junction Transistor) e o transistor de efeito de campo (FET – Field Effect Transistor). O BJT possui três terminais (emissor, base e coletor) e apresenta dois tipos: NPN e PNP. [1]

Neste projeto os transistores usados são do tipo PNP BC328. A base de cada transistor é conectada a um pino correspondente do Arduino com um resistor de 330 Ω conectado a cada base, enquanto que o coletor será conectado a uma linha da matriz e o emissor conectado à fonte de 5 V do Arduino. O acionamento de

um determinado LED ocorre quando tanto a linha quanto a coluna estão em nível baixo.

A Figura 6 mostra a que porta do Arduino cada pino da matriz de LEDs foi conectado. Por exemplo, a primeira linha da matriz está conectada conforme o que foi descrito acima, com o resistor de base conectando-se ao pino 22 do Arduino. A segunda linha idem, porém conectada ao pino 23 do Arduino, a terceira ao pino 24, e assim por diante até o pino 29, ao qual a linha 8 está conectada.

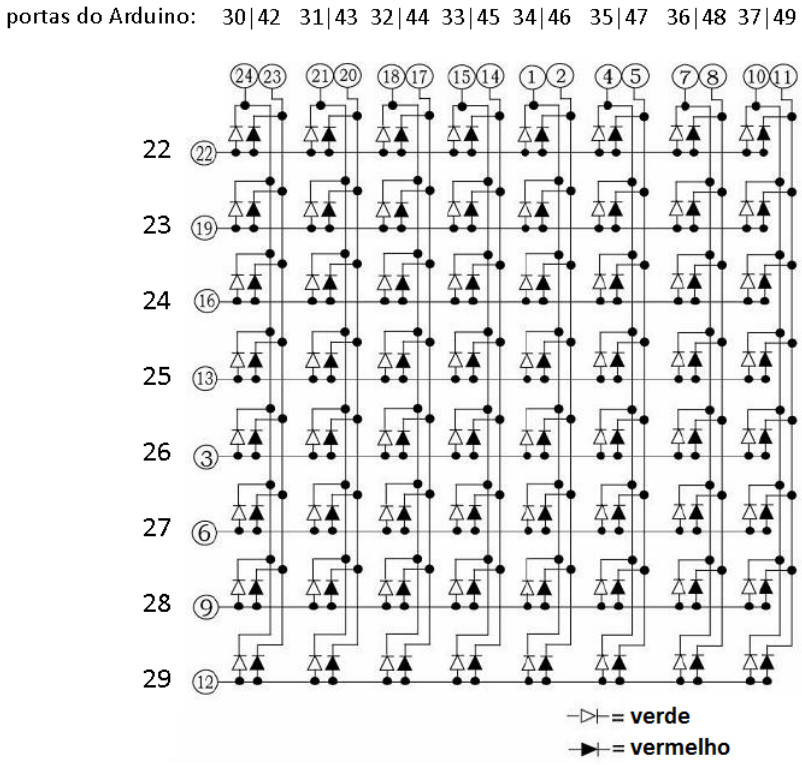

Figura 6 - Pinos correspondentes do Arduino, aos quais as linhas da matriz estão conectadas
Fonte: Adaptada do datasheet da matriz de LEDs [6]

Já a conexão das colunas consiste apenas em um resistor de 220 Ω em cada saída de coluna da matriz, como se pode ver na Figura 5. Serão usados 16 resistores, pois há 8 colunas de LEDs verdes e 8 de LEDs vermelhos. Sua conexão com as portas do Arduino está mostrada na Figura 6. A primeira coluna de LEDs verdes está conectada ao pino 30 do Arduino, a segunda ao pino 31, e assim por diante até a coluna 8, que está conectada ao pino 37. Já a sequência de pinos aos quais as colunas dos LEDs vermelhos estão conectadas vai de 42 a 49.

Essas sequências de pinos do Arduino foram escolhidas devido à maior praticidade e otimização ao programar na linguagem Processing utilizado registradores de porta, detalhados posteriormente. Os registradores de porta utilizados são o PORT L, PORT A e PORT C. A Figura 7 mostra uma parte do diagrama esquemático do Arduino Mega 2560, a qual exibe os pinos utilizados para a conexão da matriz de LEDs. Os pinos 42 a 49 correspondem ao PORT L (PL0 a PL7), os pinos 22 a 29 correspondem ao PORT A (PA0 a PA7) e os pinos 30 a 37 correspondem ao PORT C (pinos PC0 a PC7).

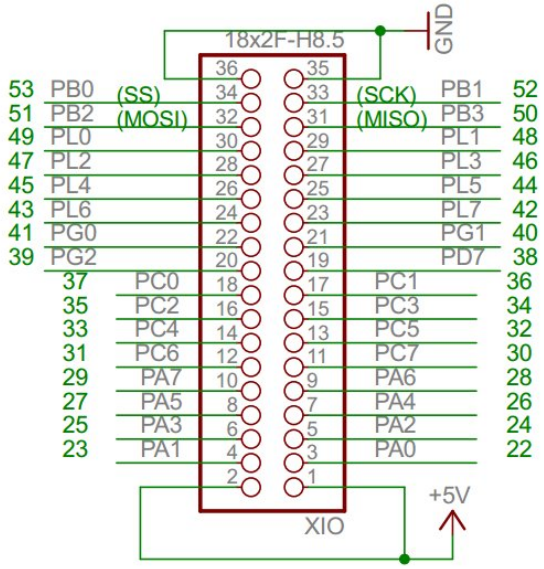

Figura 7 - Parte do diagrama esquemático do Arduino Mega 2560
Fonte: Arduino Mega 2560 Datasheet [5]

Outro circuito montado foi o da pontuação do jogo, representado em dois displays de sete segmentos. A pontuação nada mais é do que o número de maçãs que a cobra ingeriu. Para isso, foi feito um contador de módulo 100, ou seja, a contagem vai de 0 a 99, utilizando dois circuitos integrados 74LS90, contadores de década, cascateados. O diagrama esquemático do 74LS90 está representado na Figura 8.

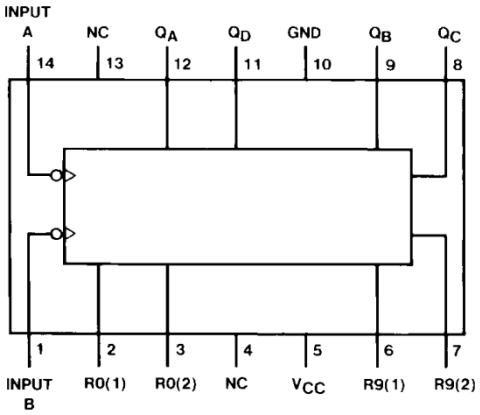

Figura 8 - Diagrama esquemático do 74LS90
Fonte: 74LS90 Datasheet [11]

O *INPUT A*, isto é, a entrada do clock do primeiro 7490 é conectado ao pino 9 do Arduino para que seja possível controlar os pulsos para a contagem e o *INPUT B* é contado à saída Q_A. O *INPUT A* do segundo é conectado à saída Q_D do primeiro 7490 e o *INPUT B* na saída Q_A, fazendo assim o cascateamento. Isso quer dizer que quando o contador de bits menos significativo (LSB), ou seja, o display posicionado à direita que conta as unidades, chega ao número 9, passando para o estado 0 posteriormente, dá um pulso ao próximo contador, de bit mais significativo (MSB), podendo assim realizar sua contagem a cada 10 contagens do LSB. Com este cascateamento, a contagem da pontuação vai de 0 a 99.

Como a saída deste circuito integrado é uma contagem binária de 4 bits, foi utilizado dois circuitos integrados 4511, um decodificador BCD para display de sete segmentos. Seu diagrama esquemático está representado na Figura 9.

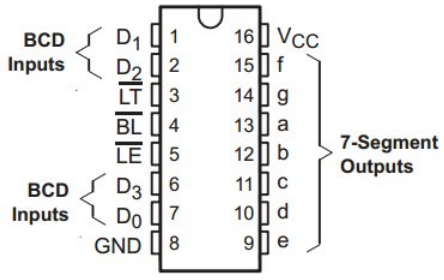

Figura 9 - Diagrama esquemático do 4511
Fonte: 4511 Datasheet [12]

Sua função é converter o estado lógico das saídas de um contador BCD (Binary Coded Decimal – Codificação Binária Decimal) em sinais que levarão ao

display de sete segmentos. O display mostra os números decimais de 0 a 9, com visualização muito mais fácil do que a contagem binária [13].

As entradas D_0, D_1, D_2 e D_3 correspondem às saídas Q_A, Q_B, Q_C e Q_D do 7490. As saídas A, B, C, D, E, F e G são conectadas aos correspondentes do display de sete segmentos. O display utilizado é o catodo comum. Seu diagrama está representado na Figura 10.

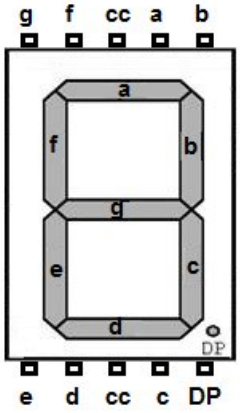

Figura 10 - Diagrama esquemático do display de sete segmentos catodo comum
Fonte: Pinagem do display de sete segmentos [14]

3.2 SOFTWARE

O software é dividido entre a dinâmica do jogo, a integração do acelerômetro com o jogo e a integração do jogo com o acionamento da matriz de LEDs. Os processos de funcionamento e as funções utilizadas serão descritas nas próximas subseções.

3.2.1 O jogo

Inicialmente o jogo foi escrito na linguagem C e depois adaptado para a linguagem Processing. Foram criadas as funções referentes à dinâmica do jogo apresentadas no Anexo B.

As posições da cobra são guardadas em um vetor de structs, as quais contêm as coordenadas x e y de cada segmento da cobra ou da maçã. A cobra é inicializada com apenas um segmento, isto é, existe apenas uma posição no vetor. No decorrer do jogo, são geradas as coordenadas da posição da maçã na matriz, de

forma aleatória, através da função `geraMaca()`. Essa função contém um método para evitar que as coordenadas da maçã possuam os mesmos valores das coordenadas de todas as posições do vetor da cobra. Consequentemente, a maçã sempre estará visível e disposta em alguma posição livre da matriz.

A movimentação da cobra se dá pela função `move()`. Nela, ocorre a verificação da direção que a cobra deve seguir através do método `direcao()`. Neste método são avaliadas as coordenadas fornecidas pela integração com o acelerômetro para que a função principal `move()` possa reposicionar o primeiro segmento da cobra em uma das posições adjacentes.

A função `colisaoMaca()`é utilizada pela função `geraMaca()` para verificar quando a cabeça da cobra, ou seja, a primeira posição do vetor cobra, tem as mesmas coordenadas das da maçã. Essa função retorna zero caso os dois objetos colidam. Em seguida, o vetor que guarda as coordenadas de cada segmento é realocado para que mais uma posição seja acrescentada a este vetor, aumentando assim o tamanho da cobra.

Quando a cabeça da cobra colide com seu próprio corpo, ou seja, as coordenadas da primeira posição se igualam a quaisquer outras referentes aos outros segmentos, o jogo acaba. Isto é verificado na função `colisaoCorpo()`, a qual realiza esta verificação em cada iteração do *loop*.

A matriz que contém o posicionamento da cobra e da maçã é preenchida com zeros através da função `preenche()` a cada *loop*. Esta função tem por objetivo limpar a matriz, para que ela possa ser atualizada posteriormente com as novas posições da cobra e da maçã. Com a matriz atualizada, é executada a função `imprime()`. Esta função é a responsável por colocar as posições da cobra em uma matriz apenas com valores 0 e 1 (1 para espaços vazios e 0 para posições da cobra) e criar uma nova matriz como a das posições da cobra, porém apenas para a maçã. É dentro desta função que ocorre o efeito dos LEDs acenderem rapidamente, porém sem comprometer a velocidade do jogo. Isto foi feito criando duas funções: `acendeLedVermelho()` e `acendeLedVerde()`. A primeira acende os LEDs vermelhos e a segundas os LEDs verdes. Elas são chamadas 10 vezes dentro da função `imprime()`, o que determina a velocidade do jogo. Quanto maior o número de vezes em que as duas funções são chamadas, menor é a velocidade do jogo, já que se essas funções são chamadas apenas uma vez, os LEDs verdes que

representam a cobra acendem uma única vez para cada posição em que a cobra anda.

3.2.2 Integração entre o jogo e a matriz de LEDs

Para o acionamento dos LEDs, é utilizado o método da varredura. Para isto, as linhas são ativadas individualmente quando estão em nível 0. Na função `imprime()`. Esta função realiza uma interpretação da matriz utilizada para marcação no decorrer do jogo e, deste modo, separa as informações obtidas em coordenadas para serem acionadas nas respectivas cores da matriz de LEDs (vermelha para a maçã e verde para a cobra).

A otimização ao programar o acionamento dos LEDs se dá devido ao uso de registradores de porta, que permitem acesso de baixo nível e manipulação mais rápida ao controle de portas I/O [8]. O registrador DDR determina quando um pino é entrada ou saída. O registrador PORT controla o estado de cada pino da sequência de pinos em nível alto ou baixo [8].

Uma das vantagens do uso dos registradores de porta é que eles permitem que o programa fique muito menor do que utilizando a função `digitalWrite()`, a qual deixa uma porta digital em nível alto ou baixo [7]. Como neste projeto serão usadas 24 portas digitais, seriam necessárias 24 funções `digitalWrite()`em um único *loop*. Já no caso das portas registradoras, é possível acionar múltiplas portas digitais ao mesmo tempo [7].

No projeto foram utilizados o *PORT A* para o acionamento das linhas de LEDs verdes. Para o acionamento do LED vermelho que representa a maçã, foi utilizado o `digitalWrite()`, já que a maçã é sempre unitária. Para que fosse possível o acionamento das linhas de LEDs verdes e vermelhos, foi utilizado o registrador de porta *PORT C*, que representa as colunas. Isto permite que sejam utilizados 8 bits sequenciais para facilmente manipular o acionamento das entradas do Arduino que formam a linha de interesse na matriz de LEDs.

Primeiramente, as linhas do *PORT C* são inicializadas com o valor em hexadecimal FE, ou seja, 11111110. Isso significa que apenas a última linha está ativada. Em seguida, são ativadas as colunas com os valores correspondentes ao

movimento do jogo, acionando os LEDs que correspondem a coordenadas que estejam com valores 0, tanto para linha, quanto para coluna.

Esta tarefa se repete por meio do deslocamento de bits. Isso ocorre devido ao fato de cada linha ser ativada individualmente. Portanto, se a última linha estava ativada, a próxima linha a ser ativada é a penúltima. Para isso, é utilizado o operador *bit shift left* (<<) e então é somado 1 bit, para que sempre haja apenas um zero na linha.

Com isso, o valor em hexadecimal que era FE (11111110), passa a ser FD (11111101), ativando assim a penúltima linha. Este processo ocorre até que o bit 0 alcance a primeira posição, ou seja, até que a linha possua o valor 7F (01111111). Este processo é executado toda vez em que se deseja imprimir a matriz na matriz de LEDs.

3.2.3 Integração entre o jogo e o acelerômetro

Para a integração entre o acelerômetro e o jogo, é utilizada a biblioteca externa AcceleroMMA7361. Esta biblioteca do Arduino facilita a coleta de dados através do acelerômetro MMA7361. Para a implementação do software, primeiramente é necessário configurar alguns parâmetros de acordo com o objetivo, conforme visto a seguir:

- Função de inicialização:
  ```
  accelero.begin(13, 12, 11, 10, A0, A1, A2);
  ```
  Seus parâmetros são os pinos do Arduino conectados aos do acelerômetro, seguindo a seguinte ordem de pinos do acelerômetro: SL (Sleep Mode), ST (Self Test), 0G (Zero G), GS (G Select), X (direção X), Y (direção Y) e Z (direção Z) [3].
- Função para definir a tensão de referência. No caso do projeto, foi utilizado 5V:
  ```
  accelero.setARefVoltage(5);
  ```
- Função que define a sensibilidade do acelerômetro como HIGH ou LOW. Para o projeto, a sensibilidade foi definida como HIGH, assim as

variações dos valores retornados das direções do acelerômetro ficam maiores, obtendo assim uma maior precisão.

```
accelero.setSensitivity(HIGH);
```

Após a inicialização, o acelerômetro já pode ser utilizado para a coleta de dados. Durante a execução do código do jogo, a cada loop são coletados os valores do ângulo em que o acelerômetro se encontra através das funções:

```
xAcelerometro = accelero.getXRaw();
yAcelerometro = accelero.getYRaw();
```

Com os valores coletados, é chamada a seguinte função, que tem por objetivo determinar se a direção da cobra deve ou não mudar de acordo com os valores coletados a partir do acelerômetro:

```
Direcao(int *xAcelerometro, int *yAcelerometro);
```

3.2.4 Pontuação no display de sete segmentos

A pontuação do jogador, ou seja, o número de maçãs que a cobra ingeriu, é mostrada em dois displays de sete segmentos, conectados em circuitos integrados. A única conexão de todo este circuito com o Arduino é através do pino 9, que está conectado ao *INPUT A* do contador de década 7490. O pino 9 é acionado com a função `digitalWrite()` assim que a cobra colide com a maçã e logo já é desligado, já que para que seja feita a contagem no circuito integrado, basta que haja um pulso. Este pulso é dado cada vez que a cobra ingere a maçã.

Assim que o jogo acaba, ou seja, a cabeça da cobra colide com qualquer outra parte do corpo, o display começa a piscar. Isto é feito desconectando o pino BI (*blank input*) do circuito integrado 4511 por 200 ms, pois ele trava a saída do C.I., e ligando novamente por 300 ms.

4 RESULTADOS

Foram encontradas várias dificuldades ao integrar o jogo à matriz de LEDs e ao tentar passar o código em C para a linguagem Processing. A maior dificuldade foi para o acionamento correto dos LEDs. A solução para o problema foi o uso de 8 transistores no circuito e acendendo-os por linhas. Seu funcionamento consiste em acionar uma linha por vez utilizando registradores de porta, que facilita a manipulação das portas do Arduino.

Outro problema encontrado foi na primeira tentativa de acionar os LEDs verdes. Como o jogo foi primeiramente escrito em C, foi preciso alterar a maneira de representar a parte gráfica do jogo. No código inteiro, a matriz que contém a cobra e a maçã era uma matriz de inteiros, com valores 0 se não possui nenhum dos dois e 1 se possui a cobra. A maçã foi colocada posteriormente. Para poder representar essa matriz na matriz de LEDs, primeiramente trocou-se os valores zeros por uns e os valores uns por zeros, já que os LEDs são acesos em nível baixo. Para poder utilizar os registradores de porta para acionar os LEDs com eficiência, é necessário que cada linha da matriz seja representada por um único número, seja ele inteiro, binário ou hexadecimal.

Para converter a linha inteira da matriz foi utilizada a função `pow()` para encontrar o valor decimal de cada linha com os valores das posições da cobra nas posições correspondentes da matriz. No entanto, os LEDs verdes da última coluna da matriz acendiam ao invés de apagar e apagavam ao invés de acender. Isso ocorreu devido a um problema de precisão do Arduino. Para compilar o código em C foi utilizado um computador Sony VPCSA22GX, com sistema operacional Windows 7 64 bits [9]. A precisão do *float* do resultado da função `pow()`com o processador do computador é muito mais precisa do que a precisão do que o processador do Arduino, o qual tem apenas de 6 a 7 dígitos decimais de precisão [10]. Para resolver este problema, ao invés dessa função, foi utilizado o *bit shift left*, deslocando 1 para a esquerda *n* vezes, que equivale a *2^n*.

Após isso, foi feita a tentativa de acionar a maçã com os LEDs vermelhos. Para isso, foi utilizado primeiramente o registrador de porta PORT L. Entretanto, este registrador de porta não funcionou e foram feitas várias tentativas com esse mesmo

registrador de porta e com vários outros como o B, o K e o D. Não foi obtido sucesso em nenhuma tentativa. Após muitos dias foi feita a tentativa de utilizar `digitalWrite()` ao invés do registrador de porta. No entanto, o LED não acendia no exato local onde a maçã sem encontrava. O erro foi corrigido após alguns ajustes na impressão da matriz dentro do software.

O último problema foi que havia LEDs que acendiam aleatoriamente na mesma coluna em que a maçã se encontrava. Isto acontecia devido às interrupções que ocorrem entre as funções de acender os LEDs verdes e vermelhos. Quando a função `acendeLedVermelho()` é executada, a linha dos LEDs vermelhos é acionada. No entanto, antes de executar a função até o final, isto é, ligar e desligar o LED, a função `acendeLedVerde()` interrompe a outra função e tenta acionar os LEDs verdes ao mesmo tempo em que a linha do LED vermelho está acionada, acendendo assim os LEDs verdes acidentalmente. A solução para este problema foi o uso das funções do Arduino `noInterrupts()` e `interrupts()`. A função `noInterrupts()` é responsável por cessar a ocorrência de quaisquer interrupções e a `interrupts()` por reativá-las [15]. Analisando-se o caso, percebeu-se que a aplicação dessas duas funções dentro da função de acionamento dos LEDs vermelhos já resolveria o erro.

.

5 CONCLUSÃO

Apesar das várias dificuldades e problemas encontrados durante a execução do projeto, sua realização foi um sucesso, obtendo resultados satisfatórios para as propostas sugeridas no início da disciplina, dando margem para o desenvolvimento do projeto além do que foi definido inicialmente: criação de uma interface da contagem da pontuação do jogo.

Para a realização deste projeto foi necessário o conhecimento de vários conteúdos que envolvem circuitos eletrônicos, como por exemplo, como acionar os LEDs da matriz de LEDs e qual transistor utilizar para seu acionamento, como funciona o contador de década 7490 e o decodificador BCD 4511 para a contagem da pontuação, entre outros. Outro conhecimento de grande importância obtido pela equipe foi o funcionamento do Arduino, desde seus pinos e registradores de portas até a sua programação com as várias funções diferentes existentes para o melhor funcionamento do dispositivo.

Este projeto foi um grande acréscimo na vida acadêmica da equipe e certamente auxiliará para a criação de projetos seguintes. O desenvolvimento deste trouxe um bom respaldo para firmar os conhecimentos da equipe adquiridos nas disciplinas do curso de Engenharia de Computação. Além disso, a realização deste projeto, principalmente pelo fato de ocorrerem tantos problemas, motiva e desperta o interesse em outras disciplinas mais à frente do curso, justamente para a obtenção da solução mais eficiente destes problemas e mais rapidamente.

Como proposta para os projetos futuros, seguem as seguintes ideias:

- Implementação de diferentes jogos. Possivelmente, até mesmo a opção de escolha entre vários jogos.
- Jogo em uma matriz de LEDs maior.

REFERÊNCIAS

[1] BOYLESTAD, R. Dispositivos Eletrônicos e Teoria de Circuitos. 5 ed. Rio de Janeiro: Prentice-Hall, 1994.

[2] Accelerometer. Encyclopedia Britannica. Disponível em: <http://global.britannica.com/EBchecked/topic/2859/accelerometer>. Acesso em Fevereiro de 2014.

[3] Datasheet Acelerômetro MMA7361. Disponível em: < https://www.sparkfun.com/datasheets/Components/General/MMA7361L.pdf>. Acesso em Dezembro de 2013.

[4] Arduino Mega 2560. Disponível em: < http://arduino.cc/en/Main/arduinoBoardMega2560>. Acesso em Dezembro de 2013.

[5] Diagrama esquemático do Arduino MEGA 2560. Disponível em: < http://arduino.cc/en/uploads/Main/arduino-mega2560_R3-sch.pdf>. Acesso em Fevereiro de 2014.

[6] Datasheet Matriz de LEDs 8x8 bicolor SZ012388K/9. Disponível em: < http://www.seeedstudio.com/depot/datasheet/SZ012388K9.pdf>. Acesso em Dezembro de 2013.

[7] Atmega Port Manipulation. Disponível em: <http://playground.arduino.cc/Learning/PortManipulation#.UwtPrPldUkg>. Acesso em Fevereiro de 2014.

[8] Ports Registers. Disponível em: <http://arduino.cc/en/Reference/PortManipulation>. Acesso em Fevereiro de 2014.

[9] Sony VPCSA22GXBI Marketing Specifications (Black). Disponível em: <https://docs.sony.com/release/specs/VPCSA22GXBI_mksp.pdf>. Acesso em Março de 2014.

[10] Float. Disponível em: <http://arduino.cc/en/Reference/Float#.UxZgE_ldUkg>. Acesso em Março de 2014.

[11] 74LS90 Datasheet. Disponível em: <http://pdf1.alldatasheet.com/datasheet-pdf/view/8342/NSC/74LS90.html>. Acesso em Março de 2014.

[12] 4511 Datasheet. Disponível em: <https://www.sparkfun.com/datasheets/IC/74HC4511.pdf>. Acesso em Março de 2014.

[13] 4511 BCD to 7-segments decoder driver. Disponível em: <http://www.doctronics.co.uk/4511.htm>. Acesso em Março de 2014.

[14] How to Drive a 7 Segments LED Display with an Arduino. Disponível em: <http://www.learningaboutelectronics.com/Articles/Arduino-7-segment-LED-display.php>. Acesso em Março de 2014.

[15] NoInterrupts(). Disponível em: <http://arduino.cc/en/Reference/noInterrupts>. Acesso em Março de 2014.

ANEXO A – ORÇAMENTO

Valores baseados em cotações da internet no ano de 2014. Fonte: <www.mercadolivre.com.br>

Componente	Quantidade	Preço total (R$)
Arduino Mega 2560 R3 Compatível	1	72,00
Acelerômetro MMA7361	1	17,00
Matriz de LEDs 8x8 bicolor	1	10,00
Jumpers	65	20,00
Protoboard	4	71,60
Mini protoboard	1	15,90
Transístor BC328 PNP	8	1,20
Resistor 330 Ω	8	2,00
Resistor 220 Ω	16	4,00
Resisto 470 Ω	14	3,50
Circuito integrado 74LS90	2	10,00
Circuito integrado 4511	2	7,00
Display de sete segmentos catodo comum	2	9,80
Total		244,00

Tabela 1 – Valores dos componentes utilizados no projeto
Fonte: Autoria própria

ANEXO B – TRECHOS DE CÓDIGO

```
void colisaoCorpo(struct posicao cobra[],int tamCobra, int *EofG)
{
        int i=1;
        for(i=1; i<tamCobra; i++){
              if (cobra[i].posX == cobra[0].posX && cobra[i].posY == cobra[0].posY){
                    *EofG=1;
               }
        }
}

void moveCorpo(struct posicao cobra[], int tamCobra){
       int cont=1;
       int auxX = cobra[0].posX,auxY = cobra[0].posY,auxX2,auxY2;
       for(cont=1; cont<tamCobra; cont++)
       {
              if(cont<(tamCobra-1)) {
                    auxX2 = cobra[cont].posX;
                    auxY2 = cobra[cont].posY;
               }
               cobra[cont].posX = auxX;
               cobra[cont].posY = auxY;
               auxX = auxX2;
               auxY = auxY2;
       }
}

void  geraMaca(struct  posicao  cobra[],  int  tamCobra,  struct  posicao  maca[],  int
*tagMaca)
{
       randomSeed(analogRead(0));
       int Mx;
       int My;
       int ok = 0;
       do
       {
              ok=0;
               Mx = random(8);
               My = random(8);
               if(ColisaoMaca(Mx,My,&cobra[0],tamCobra,&ok)==1){
                    ok=1;
               }
```

```
	}
	while(ok==1);
	maca[0].posX = Mx;
	maca[0].posY = My;
	*tagMaca=1;
}

int colisaoMaca(int x, int y, struct posicao cobra[], int tamCobra, int *ok)
{
	int cont=0;
	for(cont=0; cont<tamCobra; cont++){
		if(cobra[cont].posX==x&&cobra[cont].posY==y)
		{
			return 1;
		}
	}
	return 0;
}

void imprime(int mat[8][8])
{
	int a=0,b=0, i;
	int aux = 0, aux2 = 0;
	int vet[8];
	unsigned char linha = 0XFE;
	int mat2[8][8] = {1, 1, 1, 1, 1, 1, 1, 1,   //matriz de maçã
			  1, 1, 1, 1, 1, 1, 1, 1,   //matriz de maçã
			  1, 1, 1, 1, 1, 1, 1, 1,   //matriz de maçã
			  1, 1, 1, 1, 1, 1, 1, 1,   //matriz de maçã
			  1, 1, 1, 1, 1, 1, 1, 1,   //matriz de maçã
			  1, 1, 1, 1, 1, 1, 1, 1,   //matriz de maçã
			  1, 1, 1, 1, 1, 1, 1, 1,   //matriz de maçã
			  1, 1, 1, 1, 1, 1, 1, 1};  //todas as posições da matriz
maçã desligadas (nivel alto)

	for(a=0;a<8;a++){
		for(b=0;b<8;b++){
		if(mat[a][b] == 1) {mat[a][b] = 0;} //troca os uns por zeros por causa
do acionamento que é em nível baixo
		else if (mat[a][b] == 0) {mat[a][b] = 1;} //troca zeros por um
		if(mat[a][b] == 5){
			mat[a][b] = 1; //não acende o verde
			mat2[7-a][7-b] = 0; // acende a posição em que está a maçã
(nivel baixo)
		   }
		 }
```

```
        }

        for(a=0; a<8; a++){
                aux = 0;
                aux2 = 0;
                for(b=0; b<8 ; b++){
                    i = 7-b;
                    aux = (mat[a][b])*(1<<i) + aux; //pega uma linha inteira da
matriz e transforma seu valor "pseudo binário" em um decimal. Equivale a pow(2,i)
porém sem o uso de float, que evita o erro de precisão do arduino

                }
                vet[a]=aux;
         }
         for(a=0;a<10;a++){
                acendeLedVerde(vet);
                acendeLedVermelho(mat2);
         }
}

void acendeLedVermelho(int mat2[8][8])
{
       noInterrupts();
       int a,b;
       for(a=0;a<8;a++){
              for(b=0;b<8;b++){
                     if(mat2[a][b] == 0) {
                         digitalWrite(rows[b], LOW);  // liga o led
                         digitalWrite(colsRed[a], LOW);
                         delay(2);
                         digitalWrite(rows[b], HIGH);  // liga o led
                         digitalWrite(colsRed[a], HIGH);
                         delay(2);

                     }
                     else{
                         digitalWrite(rows[b], HIGH); // desliga o led
                         digitalWrite(colsRed[a], HIGH);

                     }
              }
        }
        interrupts();
}
```

```
void acendeLedVerde(int vet[8])
{
	int i;
	unsigned char linha = 0XFE;
	for(i=0;i<8;i++){
		PORTC = linha;
		PORTA = vet[i];
		delay(2);
		PORTC = 0XFF;
		linha = (linha << 1) +1;
	}
}

void direcao(int *Dx,int *Dy)
{
	if(xAcelerometro < 280){
		coordenada = 4; //vai para a esquerda
	}
	if(xAcelerometro > 400){
		coordenada = 2; //vai para a direita
	}
	if(yAcelerometro < 280){
		 coordenada = 3; //vai para baixo
	}
	if(yAcelerometro > 400){
		 coordenada = 1; //vai pra cima
	}
	if(coordenada == 1){
		  if(*Dx!=-1)
		  {
		      *Dx = 1;
		      *Dy = 0;
		  }
	}
	else if(coordenada == 2){
		  if(*Dy != -1){
		      *Dx = 0;
		      *Dy = 1;
		  }
	}
	else if(coordenada == 3){
		  if(*Dx!=1){
		      *Dx = -1;
		      *Dy = 0;
		  }
	}
```

```
	else if(coordenada == 4){
		if(*Dy!=1)
		{
		    *Dx = 0;
		    *Dy = -1;
		}
	}
}

void move(int mat[8][8], struct posicao cobra[0], int Dx, int Dy){
	cobra[0].posX = cobra[0].posX + Dx;
	if(cobra[0].posX==8){
		cobra[0].posX = 0;
	}
	if(cobra[0].posX==-1){
		cobra[0].posX = 7;
	}

	cobra[0].posY = cobra[0].posY + Dy;
	if(cobra[0].posY==8){
		cobra[0].posY = 0;
	}
	if(cobra[0].posY==-1){
		cobra[0].posY = 7;
	}
}
```

ANEXO C – CRONOGRAMA DE TAREFAS

	SEMANA															
ATIVIDADES	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Apresentação da disciplina																
Pesquisa sobre o projeto escolhido																
Formalização e entrega da proposta																
Pesquisa de preços e compra																
Estudo do funcionamento e testes básicos (Arduino)																
Estudo do funcionamento do acelerômetro																
Integração entre acelerômetro, Arduino e LEDs																
Estudo do funcionamento de LEDs e matrizes de LEDs																
Elaboração do relatório de progresso																
Testes básicos com a matriz de LEDs																
Integração entre a matriz de LEDs e o Arduino																
Integração entre a matriz de LEDs e o acelerômetro através do Arduino																
Implementação do software (Snake)																
Integração entre o software e o acelerômetro																
Integração entre a matriz de LEDs e o software implementado																
Elaboração da monografia final																
Apresentação do projeto final																