

Documentação_do_jogo_RPG

Problema:

Desenvolver um recurso eficaz para o funcionamento de um jogo, que seja capaz de lançar dados numéricos e identificar o número maior entre eles.

Visão geral:

Será um jogo onde funcionará um lançamento de dados e identificação de numero maior.

Descrição do Produto:

O jogo será feito no Arduino UNO, para garantir funcionamento de todos os equipamentos usados, com funcionalidades como: lançamento de dados, identificação de maior número dos dados para saber o ganhador, verificação de empate se houver fará uma nova partida.

Requisitos Funcionais

Requisitos Funcionais	Prioridade	Impacto nos negócios	Descrição Detalhada
Funções de botões	ALTA	Permitir interação com a máquina	Definir os botões (1 botão para o herói e 1 para o vilão e 2 para lançamento de dados) definir também os botões que faram o sorteio dos números
Sorteio e visualização do dado	ALTA	Realizar o sorteio dos números	O programa ao apertar o botão que vai lançar o dado, acionará a função de sorteio automático
Desempate	ALTA	Possibilitando que haja um vencedor	Se houver empate, ocorrerá uma nova partida até surgir um vencedor
Mostrar o vencedor	ALTA	Possibilizando que todos vejam quem foi o vencedor da partida	Mostrar na tela o vencedor (se foi os vilão ou o herói que ganhou, e mostrar a pontuação do vencedor logo após o led vai piscar 10vezes, e vai emitir um som (através

			da buzzer) diferente para cada vencedor)
Mensagem indicativa para início do jogo	BAIXA	permitir visualização do início da rodada	mostrar no terminal a mensagem: ===== Bem-vindo ao jogo, selecione um personagem. =====
Sistema de comparação	ALTA	Determinar o vencedor	Verifica os valores recebidos e os compara

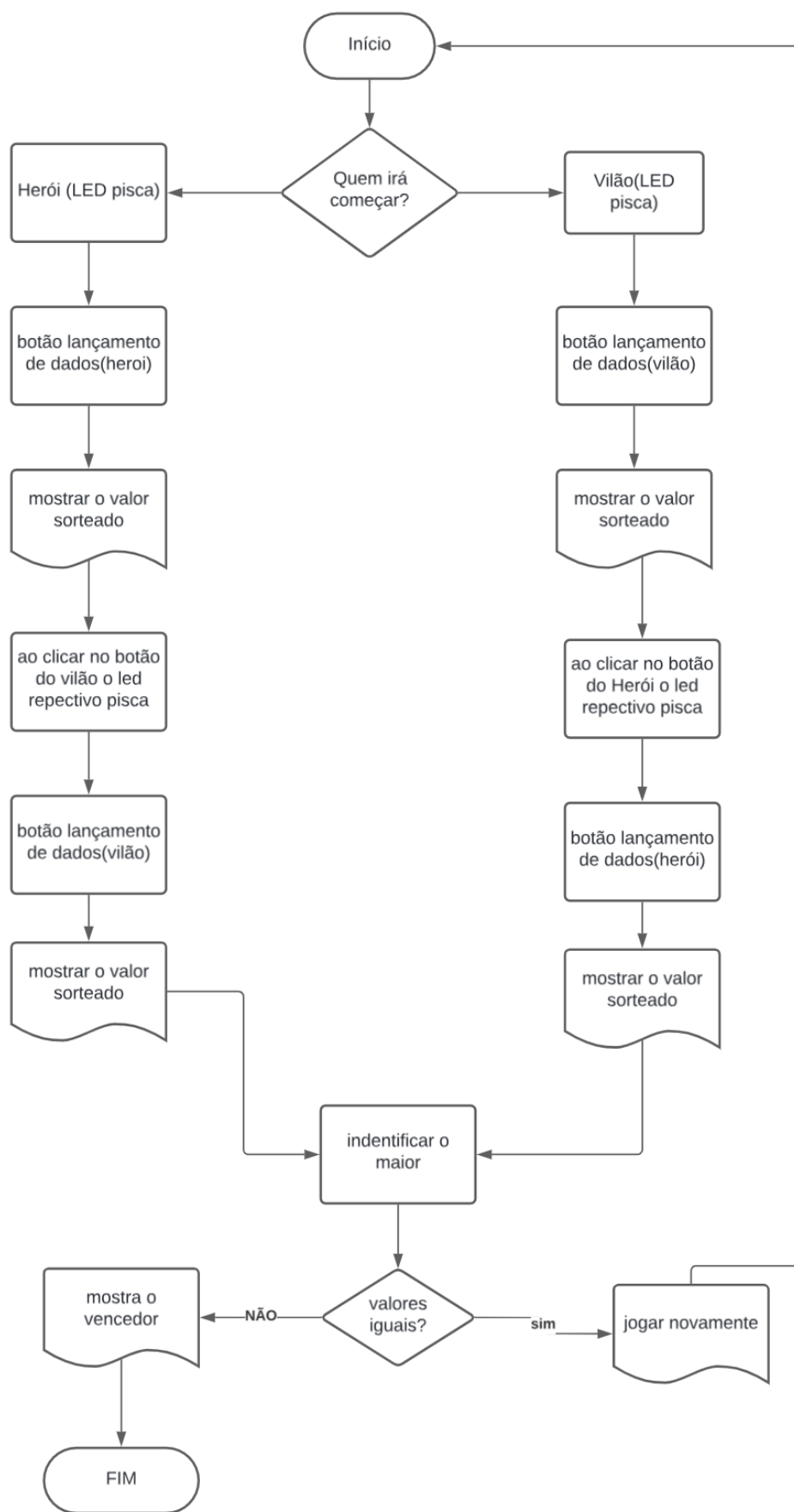
Requisitos não funcionais

Requisitos não funcionais	Prioridade	Impacto nos negócios	Descrição Detalhada
Usabilidade	ALTA	Permitir que o jogo seja funcional e usável	Facilidade e eficiência com que os usuários interagem com o jogo

Regras de negócio

Regras de negócio	Prioridade	Impacto nos negócios	Descrição Detalhada
Quantidade de jogadores	ALTA	Ocasionalmente que o jogo funcione apenas com dois jogadores	A quantidade permitida é de apenas 2 jogadores sendo um herói e um vilão, que competiram entre si.
Sistema de comparação	ALTA	Possibilitando que reconheça o maior, e apenas um ganhe	Definir o ganhador pelo número sorteado de valor maior, ao oponente.
Número sorteado	MÉDIA	Não permitir que outros números aleatórios sejam sorteados	O valor do número sorteado, é como de um dado real (de 1 à 6)
Empate	MEDIA	Para que não haja confusão	Caso aconteça um empate realizará uma nova partida.

Fluxograma



Implementação:

```
// Constantes para os pinos
#define botao_selecionar_heroi 2
#define botao_dado_heroi 3
#define led_heroi 6
#define botao_selecionar_vilao 5
#define botao_dado_vilao 4
#define led_vilao 7
#define Buzzer 8 // Pino do buzzer

bool partida_em_andamentoH = false;
bool partida_em_andamentoV = false;
bool dado_heroi_lancado = false;
bool dado_vilao_lancado = false;

int contH = 0;
int contV = 0;
int Dado_heroi = 0; // Variável para o sorteio dos números
int Dado_vilao = 0; // Variável para o sorteio dos números

void setup() {
    // Definição das entradas e saídas de dados
    pinMode(led_heroi, OUTPUT);
    pinMode(led_vilao, OUTPUT);
    pinMode(Buzzer, OUTPUT);
    pinMode(botao_selecionar_heroi, INPUT);
    pinMode(botao_selecionar_vilao, INPUT);
    pinMode(botao_dado_heroi, INPUT);
    pinMode(botao_dado_vilao, INPUT);
    randomSeed(analogRead(0));
    Serial.begin(9600);
    delay(1000);
    Serial.println("\n
===== \n\
| Bem-vindo ao jogo, |\n\
| selecione um      |\n\
```

```

|      personagem.      |\n\
=====");
  Serial.println(" ");
}

void loop() {

  int estado_botao_heroi = digitalRead(botao_selecionar_heroi);
  int estado_botao_vilao = digitalRead(botao_selecionar_vilao);

  delay(1000);

  // Selecionando o Herói
  if (estado_botao_heroi == HIGH && partida_em_andamentoH == false) {
    Serial.println("Herói selecionado");
    Serial.println("-----");
    digitalWrite(led_heroi, HIGH);
    delay(1000);
    digitalWrite(led_heroi, LOW);
    contH += 1;
    partida_em_andamentoH = true;
  }

  // Sorteando os dados do héroi
  if (contH == 1 && digitalRead(botao_dado_heroi) == HIGH && partida_em_andamentoH == true) {
    Serial.print("Lancando os dados do herói");
    delay(1000);
    Serial.print(".");
    delay(1000);
    Serial.print(".");
    delay(1000);
    Serial.print(".");
    delay(1000);
    Dado_heroi = random(1, 7);
    Serial.println("");
    delay(500);
    Serial.println("-----");
    delay(500);
  }
}

```

```

    Serial.print("Valor sorteado do Heroi: ");
    Serial.println(Dado_heroi);
    delay(500);
    Serial.println("-----");
    dado_heroi_lancado = true;
}

// Selecionando o vilão
if (estado_botao_vilao == HIGH && partida_em_andamentoV == false)
{
    Serial.println("Vilao selecionado");
    Serial.println("-----");
    digitalWrite(led_vilao, HIGH);
    delay(1000);
    digitalWrite(led_vilao, LOW);
    contV += 1;
    partida_em_andamentoV = true;
}

// Sorteando os dados do vilão
if (contV == 1 && digitalRead(botao_dado_vilao) == HIGH && partida_em_andamentoV == true)
{
    Serial.print("Lancando os dados do vilao");
    delay(1000);
    Serial.print(".");
    delay(1000);
    Serial.print(".");
    delay(1000);
    Serial.print(".");
    delay(1000);
    Serial.print(".");
    delay(1000);
    Dado_vilao = random(1, 7);
    Serial.println("");
    delay(500);
    Serial.println("-----");
    delay(500);
    Serial.print("Valor sorteado do Vilao: ");
    Serial.println(Dado_vilao);
    delay(500);
    Serial.println("-----");
    dado_vilao_lancado = true;
}

```

```

}

// Sistema de comparação

if (Dado_heroi > 0 && Dado_vilao > 0) {

    // Vilão venceu
    if (Dado_heroi < Dado_vilao) {
        Serial.print("");
        delay(1000);
        Serial.println("Vilao venceu!!!");
        fim_de_jogo();
        for (byte cont = 0; cont < 10; cont++) {
            digitalWrite(led_vilao, HIGH);
            delay(300);
            digitalWrite(led_vilao, LOW);
            delay(300);
        }
        musica_vilao();
        delay(2000);
        for (byte cont = 0; cont < 20; cont++) {
            Serial.println(" ");
        }
        delay(1000);
        Serial.println("\
===== \n\
| Bem-vindo ao jogo,  |\n\
| selecione um        |\n\
|     personagem.     |\n\
=====");
        Serial.println("");
    }

    // Herói venceu
    else if (Dado_heroi > Dado_vilao) {
        Serial.print("");
        delay(1000);
        Serial.println("Heroi venceu!!!");
    }
}

```



```

    fim_de_jogo();
    for (byte cont = 0; cont < 10; cont++) {
        digitalWrite(led_heroi, HIGH);
        delay(300);
        digitalWrite(led_heroi, LOW);
        delay(300);
    }
    musica_heroi();
    delay(2000);
    for (byte cont = 0; cont < 20; cont++) {
        Serial.println(" ");
    }
    delay(1000);
    Serial.println("\n\
===== \n\
| Bem-vindo ao jogo, |\n\
| selecione um      |\n\
|   personagem.    |\n\
=====");
    Serial.println(" ");
}

// Empate
else {
    delay(1000);
    Serial.println("Empate!");
    delay(1000);
    Serial.println("Jogue novamente");
    fim_de_jogo();
    digitalWrite(led_heroi, HIGH);
    digitalWrite(led_vilao, HIGH);
    delay(3000);
    digitalWrite(led_heroi, LOW);
    digitalWrite(led_vilao, LOW);
    delay(2500);
    for (byte cont = 0; cont < 20; cont++) {
        Serial.println(" ");
    }
}

```



```
tone(Buzzer, 392); //G
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 311); //D#
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 466); //A#
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 392); //G
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 311); //D#
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 466); //A#
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 392); //G
delay(582);
noTone(Buzzer);
delay(582);

tone(Buzzer, 587); //D
delay(291);
noTone(Buzzer);
```

```
delay(291);

tone(Buzzer, 587);  //D
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 622);  //D#
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 466);  //A#
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 370);  //F#
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 311);  //D#
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 311);  //D#
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 466);  //A#
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 392);  //G
```

```
delay(582);
noTone(Buzzer);
delay(582);

tone(Buzzer, 784); //G
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 392); //G
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 392); //G
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 784); //G
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 740); //F#
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 698); //F
delay(97);
noTone(Buzzer);
delay(97);

tone(Buzzer, 659); //E
delay(97);
noTone(Buzzer);
delay(97);
```

```
tone(Buzzer, 622); //D#
delay(97);
noTone(Buzzer);
delay(97);

tone(Buzzer, 659); //E
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 415); //G#
delay(145);
noTone(Buzzer);
delay(145);

tone(Buzzer, 554); //C#
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 523); //C
delay(291);
noTone(Buzzer);
delay(291);

tone(Buzzer, 494); //B
delay(97);
noTone(Buzzer);
delay(97);

tone(Buzzer, 466); //A#
delay(97);
noTone(Buzzer);
delay(97);

tone(Buzzer, 440); //A
delay(97);
```

```
noTone(Buzzer);  
delay(97);  
  
tone(Buzzer, 466); //A#  
delay(291);  
noTone(Buzzer);  
delay(291);  
  
tone(Buzzer, 311); //D#  
delay(145);  
noTone(Buzzer);  
delay(145);  
  
tone(Buzzer, 370); //F#  
delay(291);  
noTone(Buzzer);  
delay(291);  
  
tone(Buzzer, 311); //D#  
delay(145);  
noTone(Buzzer);  
delay(145);  
  
tone(Buzzer, 466); //A#  
delay(145);  
noTone(Buzzer);  
delay(145);  
  
tone(Buzzer, 392); //G  
delay(291);  
noTone(Buzzer);  
delay(291);  
  
tone(Buzzer, 311); //D#  
delay(145);  
noTone(Buzzer);  
delay(145);
```

```

    tone(Buzzer, 466);  //A#
    delay(145);
    noTone(Buzzer);
    delay(145);

    tone(Buzzer, 392);  //G
    delay(582);
    noTone(Buzzer);
    delay(582);
}

void musica_heroi() {

    delay(1000);

    tone(Buzzer, 659);  //E
    delay(130);
    noTone(Buzzer);
    delay(130);

    tone(Buzzer, 659);  //E
    delay(130);
    noTone(Buzzer);
    delay(130);

    tone(Buzzer, 659);  //E
    delay(160);
    noTone(Buzzer);
    delay(160);

    tone(Buzzer, 523);  //C
    delay(150);
    noTone(Buzzer);
    delay(150);

    tone(Buzzer, 659);  //E
    delay(130);
    noTone(Buzzer);

```



```
delay(130);

tone(Buzzer, 784); //G
delay(300);
noTone(Buzzer);
delay(300);

tone(Buzzer, 392); //G
delay(250);
noTone(Buzzer);
delay(250);

tone(Buzzer, 523); //C
delay(320);
noTone(Buzzer);
delay(320);

tone(Buzzer, 392); //G
delay(178);
noTone(Buzzer);
delay(178);

tone(Buzzer, 440); //A
delay(178);
noTone(Buzzer);
delay(178);

tone(Buzzer, 494); //B
delay(119);
noTone(Buzzer);
delay(119);

tone(Buzzer, 466); //A#
delay(119);
noTone(Buzzer);
delay(119);

tone(Buzzer, 440); //A
```

```
delay(119);
noTone(Buzzer);
delay(119);

tone(Buzzer, 392); //G
delay(178);
noTone(Buzzer);
delay(178);

tone(Buzzer, 523); //C
delay(150);
noTone(Buzzer);
delay(150);

tone(Buzzer, 659); //E
delay(150);
noTone(Buzzer);
delay(150);

tone(Buzzer, 784); //G
delay(150);
noTone(Buzzer);
delay(150);

tone(Buzzer, 880); //A
delay(150);
noTone(Buzzer);
delay(150);

tone(Buzzer, 784); //G
delay(150);
noTone(Buzzer);
delay(150);

tone(Buzzer, 659); //E
delay(150);
noTone(Buzzer);
delay(150);
```

```
tone(Buzzer, 784); //G
delay(150);
noTone(Buzzer);
delay(150);

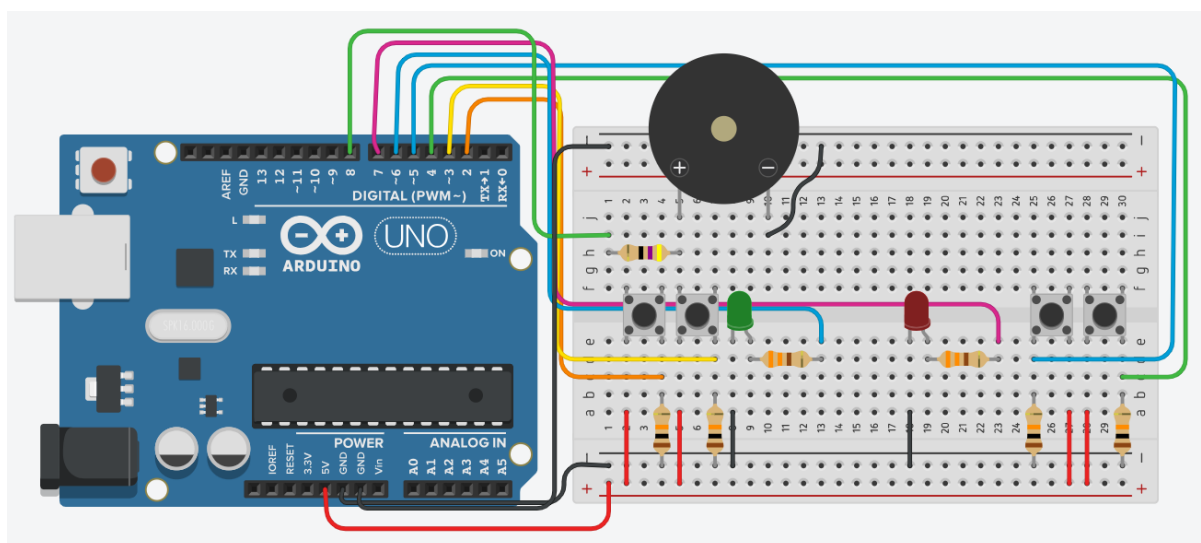
tone(Buzzer, 659); //E
delay(150);
noTone(Buzzer);
delay(150);

tone(Buzzer, 523); //C
delay(150);
noTone(Buzzer);
delay(150);

tone(Buzzer, 659); //E
delay(150);
noTone(Buzzer);
delay(150);

tone(Buzzer, 523); //C
delay(500);
noTone(Buzzer);
delay(500);
}
```

Implementação no Tinkercard:



Diferencial do Jogo:

Nosso jogo oferece uma experiência e competição entre jogadores apostando na sorte, para que haja emoção a cada lançamento de dado, assim criando uma batalha onde a vitória é conquistada pela sorte.

Realização/divisão do projeto

Projeto	Descrição	Estimativa
Definição dos Botões	Yasmin e Clara P.	3 horas
Sorteio e visualização do dado	Maria Júlia e Arthur	4 horas
Sistema de comparação	Arthur e Maria Júlia	4 horas
Mostrar o vencedor	Clara P. e Yasmin	3 horas
Montagem do Arduino	Clara P. e Yasmin	15 min.