UNIVERSIDADE CATÓLICA DE PERNAMBUCO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
COORDENAÇÃO DO CURSO DE FÍSICA

HEMILLY MONARA RODRIGUES DA SILVA
JOÃO VITOR LIMA SILVA
KAROLAYNE TEIXEIRA DA SILVA
LUCAS GABRIEL DIAS DA LUZ
SAMARA REGINA RODRIGUES DA SILVA

Hikas

O jogo musical programável

Recife
Jun/2020

HEMILLY MONARA RODRIGUES DA SILVA
JOÃO VITOR LIMA SILVA
KAROLAYNE TEIXEIRA DA SILVA
LUCAS GABRIEL DIAS DA LUZ
SAMARA REGINA RODRIGUES DA SILVA

HIKAS

Trabalho apresentado à disciplina introdução à robótica, como parte dos requisitos necessários para obtenção de nota da disciplina.

Prof. Antônio Cruz.

Recife
Jun/2020

SUMÁRIO

1.0 - INTRODUÇÃO

Hikas é um jogo musical que estimula de diversas formas o aprendizado musical e sonoro. Com seus modos diversificados e interativos, Hikas procura se manter divertido e não monótono juntando em um só lugar várias táticas de ensino e jogo musical.

Com seu estilo dinâmico surge a possibilidade do contínuo aprendizado em diversos âmbitos. Nele, além de se aprender a tocar uma música, há a estimulação da memória, da criatividade e da precisão. Além de outros benefícios que vem junto com a educação musical básica.

1.1 – A importância da educação musical

- A música ajuda no desenvolvimento neurológico da criança

Uma pesquisa realizada pela Universidade de Vermont nos EUA, constatou depois de analisar tomografias de 232 crianças entre 6 e 18 anos, que as crianças que estudavam música desde a infância apresentavam ganhos positivos e duradouros em relação ao desenvolvimento cerebral.

- O aprendizado musical modifica fisicamente o cérebro, principalmente na primeira infância (0 a 6 anos) e os ganhos se mantêm por toda a vida.

Com os estudos avançados da Neurociência, hoje temos informações seguras que o período entre 0 e 6 anos é de extrema importância para o desenvolvimento infantil. É a fase de estruturação do Sistema Nervoso, onde comportamentos são aprendidos e consolidados.

- Todos podem desenvolver a inteligência musical.

Howard Gardner, professor da Faculdade de Harvard, ganhador de diversos prêmios, afirma que o que leva as pessoas a desenvolverem a inteligência (inclusive a musical) é: O ambiente que vivem, experiências significativas que proporcionem a vontade de praticar, a educação e as oportunidades que surgem ao longo da vida.

- A música é interdisciplinar.

Desde uma brincadeira de roda, ou tocar um instrumento musical, nas aulas de música, o cérebro é desafiado a utilizar diversas coordenações ao mesmo tempo (cantar/dançar, cantar/tocar/dançar, dançar/tocar, etc.) Essas informações são assimiladas pelo cérebro através dos sentidos, colaborando de forma significativa nas aquisições linguísticas, sociais e cognitivas, lembrando que todos os aspectos do desenvolvimento (motor, intelectual e afetivo) estão interligados e um exerce influência sobre o outro.

- Praticar música ensina a importância da disciplina.

A música é uma linguagem matemática, mas ao mesmo tempo afetiva sempre inserida em um contexto social. Por ser tão poderosa e influente, através dela podemos aprender qualquer conteúdo, pois é uma linguagem que traz a motivação necessária para aprender.

1.2 – Público-Alvo

A partir dos dados coletados é possível concluir que atualmente o público-alvo do nosso jogo educacional é na grande maioria crianças. Porém é caracterizada como uma área versátil e abrangente, assim pode atingir os mais variados públicos. Apresentamos um jogo educacional que engloba mais de uma forma de ensinar música de uma maneira divertida e não monótona. Fazendo com que seu jogador tenha acesso ao jogo físico mas ao mesmo tempo tenha elementos de jogos digitais.

1.3 – O Projeto

O que é?

O Hikas é um jogo musical o qual busca ser versátil e iterativo de forma a se moldar a seu jogador. No Hikas é possível programar o seu jogo do seu jeito, com uma acessibilidade ampla para vários públicos.

Como funciona?

Hikas busca ser dinâmico e divertido e para isso possui quatro modos base, são eles:

1 - **Modo Livre**: Nesse modo, pode-se tocar livremente as notas alterando suas escalas entre grave e agudo o quando quiser.

2 - **Modo Genius**: Inspirado no clássico jogo Genius dos anos 80. O modo Genius tem diferenças importantes. Nele, ao invés de quatro, tem-se sete teclas que simulam um mini teclado. Assim, é possível selecionar a música e a velocidade desejada e o jogo repetirá a música estimulando a memória auditiva do jogador. Ao contrário do que ocorre no Genius o qual as notas são dadas aleatoriamente pelo controlador.

3 - **Modo Scroll**: Como o próprio nome diz, o modo Scroll se baseia no famoso estilo de jogo musical, o qual as notas chegam, uma atrás da outra, como é o caso do jogo Guitar Hero. Para vencer esse modo é necessário precisão e velocidade.

3 - **Modo Programação**: Neste modo tem-se duas opções:

3.1 - **Edição**: Nele pode-se liberar as notas do jogo Genius e Scroll para serem livres, ou seja, não mais se prenderem as notas padrões (com frequências predefinidas);

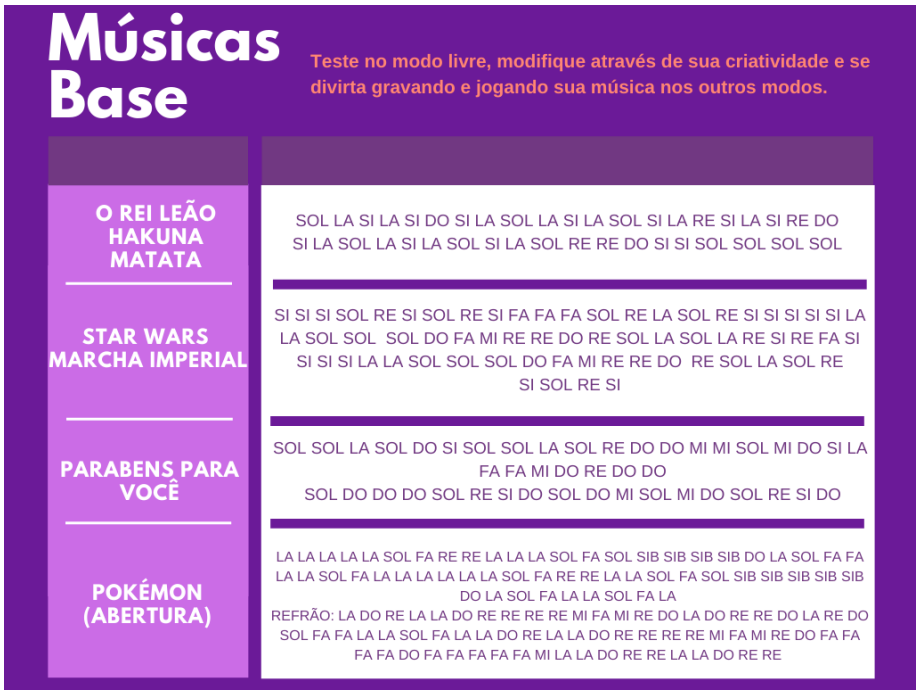
3.2 - **Gravar**: Aqui é possível gravar sua música na memória do Hikas para que posteriormente possa ser jogada no modo Genius ou Scroll. Na hora da gravação também é possível modificar as escalas musicais, dando assim mais opções e mais vida a música.

Para ver os modos em funcionamento acesse nosso vídeo no YouTube:

O manual do Jogo

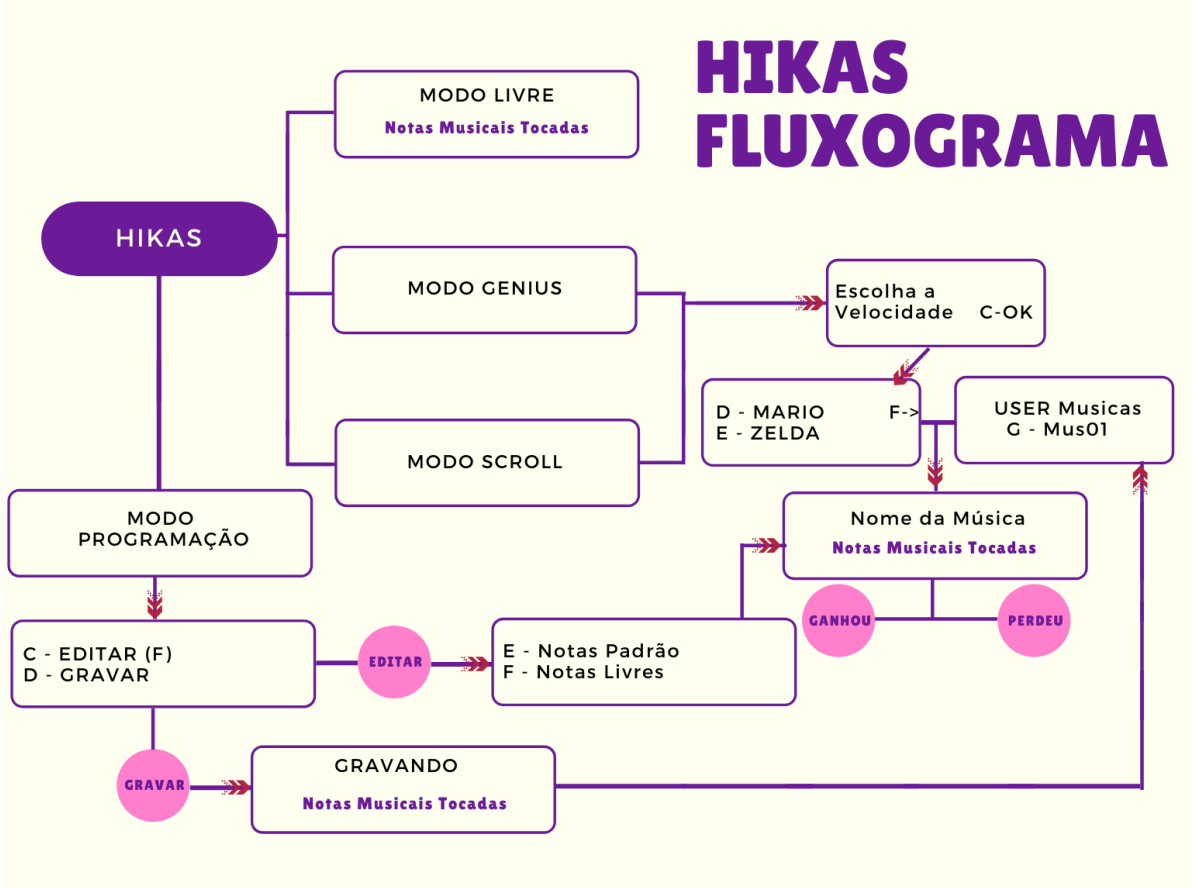
Como todo jogo que demanda mais conhecimento prévio, Hikas possui um mini manual que acompanha o jogo, explicando de forma simples e resumida seus processos:

Verso do manual



Fluxograma dos menus

A seguir um fluxograma ilustrativo que descreve o comportamento desses modos no jogo:



1.3.1 – Justificativa

Ensinar teoria musical básica através de diversos estímulos e metodologias que são aplicadas por meio dos modos do jogo. Tornando assim, o ensino e aprendizagem musical mais dinâmico e não-monótono. Um exemplo é quando se compra um instrumento para uma criança, a paciência e a perseverança são as principais virtudes necessárias para continuar o aprendizado. E essas virtudes, por vezes, não são conquistadas tão facilmente o que leva ao aluno a desistência. Com a junção da educação e do jogo, se tem um ambiente mais acessível o qual promove o contínuo aprendizado e reduz as desistências, pois através de um jogo e sua estrutura é possível ter avanços mais visíveis, rápidos e facilitados.

2.0 - OBJETIVOS

Ensinar teoria musical básica através de metodologias que buscam aprimorar vários aspectos importantes para o aprendizado mais completo em música. Tais como, a memória, a velocidade e precisão, reconhecimento sonoro de notas e criatividade.

3.0 - MATERIAIS E MÉTODOS

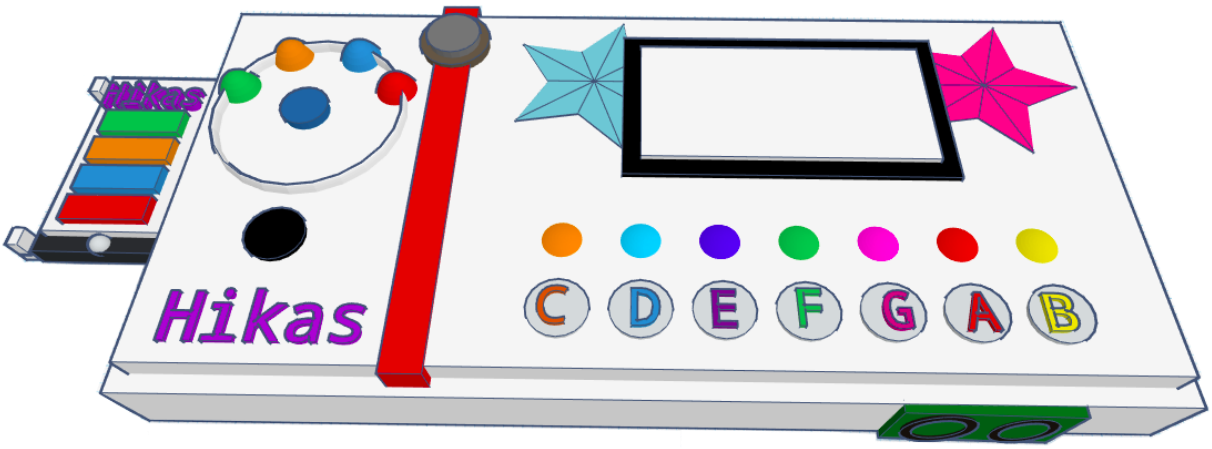
3.1- Materiais

Quantidade	Nome	Preço (Aprox.)
1	Arduino Uno R3	59,90
1	LCD 16x2	20,00
8	Botões	1,92
2	Potenciômetros	3,00
11	LEDs	2,75
12	Resistores 180 Ω	1,80
1	Resistor 200 Ω	0,08
1	Alto-falante	8,00
1	Bateria 9v	7,50
1	Conversor de 5v	4,20
2	Shift Register (74HC595)	2,60
1	Sensor Ultrassônico	11,00
1	EEPROM	10,00
1	Capacitor	0,50
Total		132,80

3.2- Metodologia

O Hikas foi desenvolvido inteiramente online, através plataforma Tinkercad. E como tal plataforma possui algumas limitações, o Hikas ganhou mais uma versão, sendo esta desenvolvida no Fritzing de forma mais avançada. Essa versão mais completa será descrita posteriormente.

A versão um, com apenas sete botões, foi estruturada baseando-se no modelo 3D proposto do jogo, que se encontra abaixo (Clique na imagem para ser redirecionado para o modelo 3D no Tinkercad):

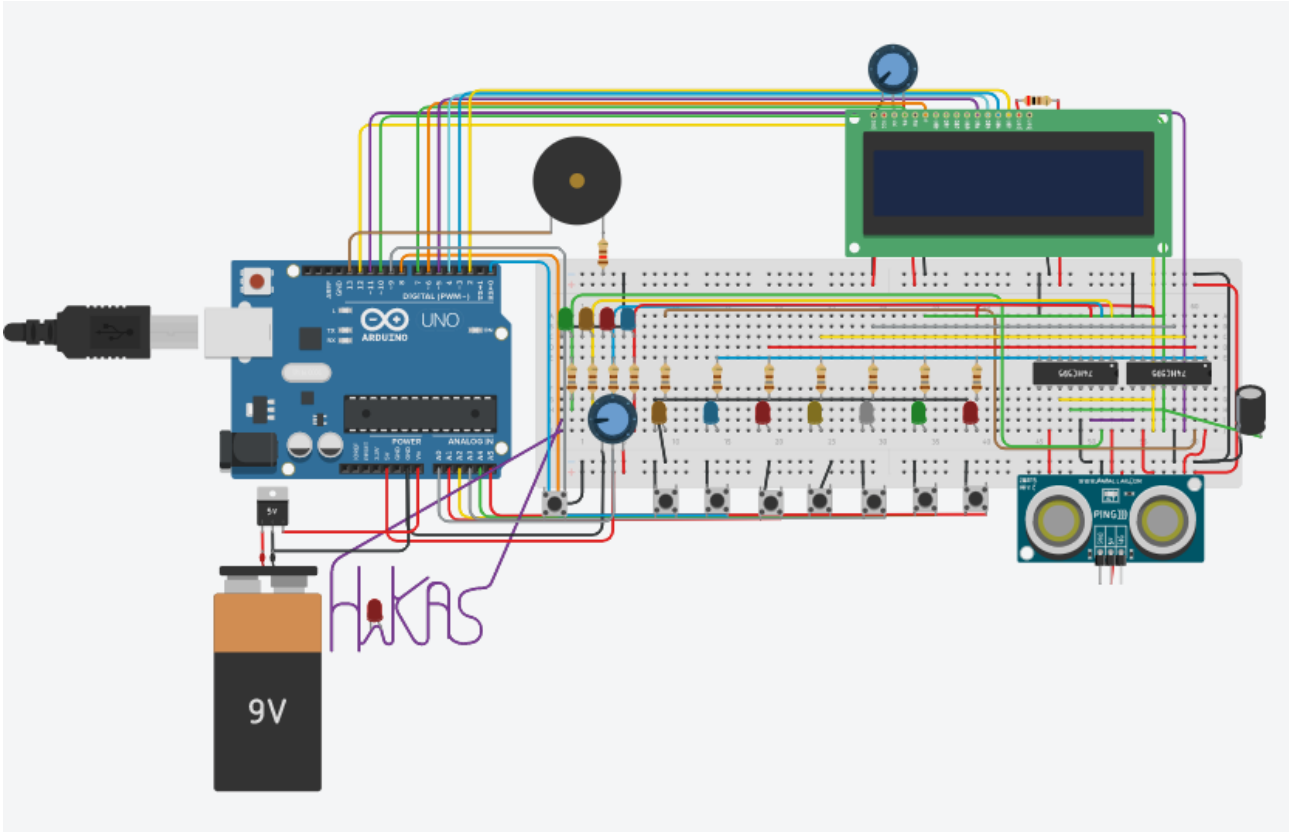


Foi visado um jogo portátil e compacto que não deixe a desejar em questão de funcionalidades.

O modelo foi feito de maneira a ficar mais compacto e de fácil utilização, com cores vibrantes e letras para identificar cada nota disposta em sua base. A partir do modelo, é possível identificar onde os materiais e componentes utilizados ficariam dispostos.

4.0 - MONTAGEM DO PROJETO

Abaixo o circuito do projeto (Clique para ser redirecionado para o projeto no Tinkercad):



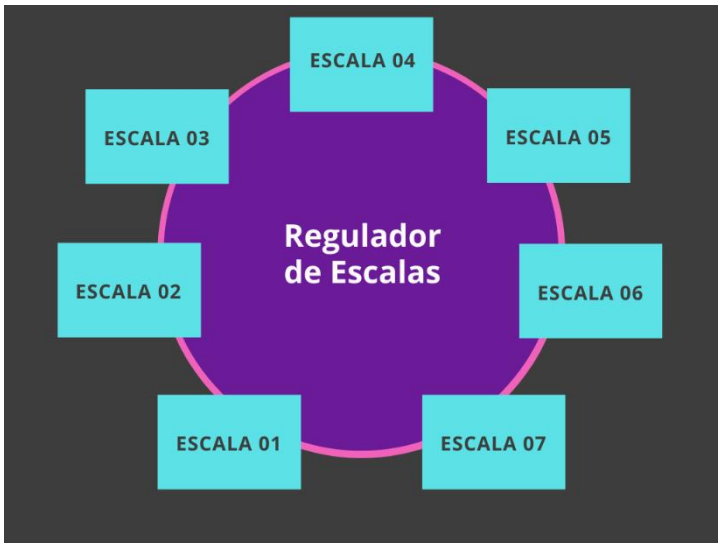
4.1 - Circuito principal

Fazem parte do circuito principal: os 7 botões que representam as notas: Dó, Ré, Mi, Fá, Sol, Lá e Si juntamente com o botão principal. As leds e o potenciômetro principal (o abaixo das quatro leds que representam os modos).

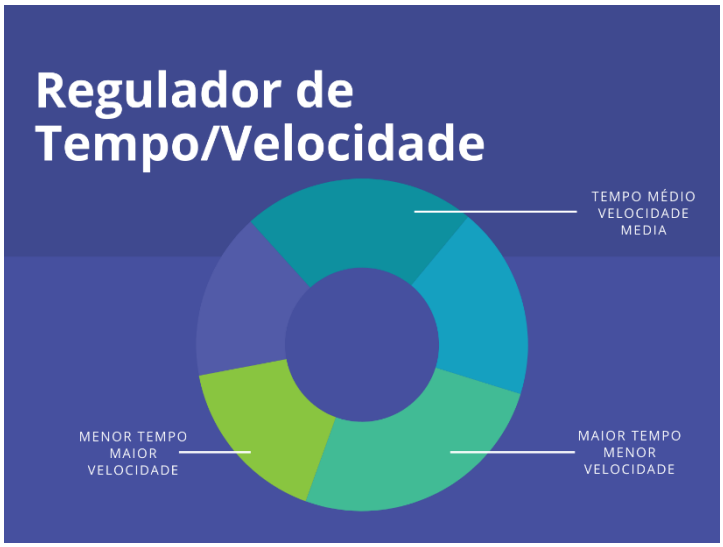
O Botão principal é de grande importância, pois com ele é possível mudar entre os modos de forma simplificada através da lógica de um botão de multifunções.

O potenciômetro tem uma função de grande importância dentro do jogo, ele possibilita a modelagem do jogo de acordo com seu jogador.

Nos Modos Livre e Programação (Gravar): ele pode mudar a escala musical com um simples girar. Nele estão dispostas 7 escalas musicais (não atendendo as notas bemol e sustenido, por questões de espaço).



Nos modos Genius e Scroll: o potenciômetro regula o tempo entre as notas, alterando assim a velocidade do jogo.



4.2 - **LCD: a Interface Jogador-Arduino**

A função do Lcd dentro de nosso projeto é, a todo momento, mostrar para o jogador as notas e suas escalas, para que assim, o mesmo, se torne familiarizado com as notas musicais. Além de ser um visor dos menus existentes no Hikas.

4.3 - **Sensor – A economia**

O sensor de distância ultrassônico tem um papel fundamental dentro do jogo. O de economizar bateria se não há ninguém jogando o jogo por um determinado tempo. Para isso, utilizamos o modo Sleep.

O Modo Sleep é basicamente uma ferramenta que não está presente de forma nativa no Arduino, todavia que pode ser implementada. Sua função é de desativar alguns componentes que estão sendo pouco usados, dessa forma economizando energia, geralmente desativa-se alguns dos clocks presentes no microcontrolador, tal recurso é de fundamental importância em componentes eletrônicos em geral para não só a redução do consumo de energia mais também para a maior duração da vida útil dos componentes caso necessário.

Essa funcionalidade pode reduzir em até 50% o consumo de energia de todo e qualquer aparelho que use um microcontrolador, como o caso do projeto apresentado do jogo HIKAS, onde o microcontrolador(Arduino) reduz a quantidade de operações internas a cada intervalo de tempo, consumindo assim menos potência, o modo sleep é comumente chamado de Stand By, vejamos a tabela abaixo para melhor exemplificar como funciona tal tarefa:
Como vemos o microcontrolador faz a desabilitação dos clocks, porém, pode surgir uma dúvida em como o Arduino faz para ativá-los, para isso existe o modo wake-up sources.

	Active Clock Domains					Oscillators		Wake-up Sources							Software BOD Disable
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT and PCINT	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	
Idle			Yes	Yes	Yes	Yes	Yes[2]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
ADC Noise Reduction				Yes	Yes	Yes	Yes[2]	Yes[3]	Yes	Yes[2]	Yes	Yes	Yes		
Power-down								Yes[3]	Yes				Yes		Yes
Power-save					Yes		Yes[2]	Yes[3]	Yes	Yes			Yes		Yes
Standby[1]						Yes		Yes[3]	Yes				Yes		Yes
Extended Standby					Yes[2]	Yes	Yes[2]	Yes[3]	Yes	Yes			Yes		Yes

Economia de energia: A economia de energia está totalmente ligada com o algoritmo e a forma como o circuito foi montado, além disso a utilização da plaquinha do Arduino também influencia, tendo em vista que os elementos dela geram um determinado gasto de

energia. Por exemplo o regulador de tensão, que pode consumir cerca de 10 mA. Como utilizado no projeto faremos a verificação e medição do gasto de corrente do Arduino UNO no modo normal e com o modo sleep ativo.

Modo	Arduino UNO	Microcontrolador à parte
Normal	85,7mA	10,4mA
Sleep	58,8mA	0,32mA

Observe que neste exemplo a efetividade do sleep mode no Arduino é muito perceptível, uma vez que o consumo de energia cai drasticamente quando utilizado em modo Stand By. Podemos também fazer um paralelo no que diz respeito a jogos, a arquitetura dos componentes de um computador pode ser perfeitamente comparado com o nosso exemplo, uma vez que, na programação de um game por exemplo, onde o foco do uso está mais centrado na GPU(placa de vídeo/memória de vídeo) por exemplo, o próprio jogo faz com que o uso do processador caia pela metade, consequentemente diminuindo a temperatura e as atribuições correlatas ao microcontrolador. Vejamos com um algoritmo simples como implementar o modo sleep e o wake up sources, lembrando que o Arduino não possui essa biblioteca, tornando-se necessária a importação com o seguinte comando:

```
#define interrupcao 2
#define led 13
void setup()
{
   // Define o pino 2 como entrada e ativa o resistor de pull-up
   pinMode(interrupcao, INPUT_PULLUP);
   // Define o LED como saida
   pinMode(led, OUTPUT);
}
```

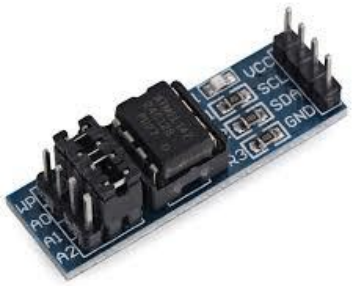
Função Sleep(dormir):

```
void dormir(){
   set_sleep_mode(SLEEP_MODE_PWR_DOWN);
   sleep_enable();
   attachInterrupt(digitalPinToInterrupt(interrupcao), acordar, LOW);
   sleep_cpu();
}
```

Função Wake Up Sources:

```
1    void acordar(){
2      // Desabilita o sleep
3      sleep_disable();
4      // Desabilita a interrupção
5      detachInterrupt(0);
6    }
```

4.4 - **EEPROM – Memória extra**



EEPROM é uma forma de ampliar a memória do Arduino, onde para projetos um pouco maiores e uma boa escolha para utilizar. A memória disponível vai de acordo com a placa utilizada, no nosso caso um Arduino Uno. O EEPROM disponibiliza 1KB de memória, possui 1024 posições de memória para os dados de um byte (8 bits), ou seja, o maior valor que tem a capacidade de gravar e de 255, para o nosso projeto seria utilizado por causa das suas diversas funções, podendo armazenar dados de determinadas funções, valores de medição e determinados resultados de cálculos, a biblioteca vem com a função, EEPROM.write, onde permite que envie determinado dado a uma determinada posição, ou endereço.

5.0 - APLICATIVO HIKAS

Os smartphones estão por toda parte hoje em dia, e pensando nisso e nas diversas possibilidades que essa tecnologia oferece, o aplicativo Hikas busca integrar o celular com o jogo físico através de uma conexão bluetooth. Assim, com o celular controlando as opções do jogo, o grau de acessibilidade aumenta consideravelmente e a expansão do jogo em si também.

O aplicativo foi desenvolvido através do site App Inventor 2, criado pelo MIT, e toda sua programação feita pelo mesmo site. Obviamente testes prévios não puderam ser feitos para comprovação total de seu funcionamento. Todo o código pode ser encontrado juntamente com todo material do projeto hospedado no GitHub. A seguir as telas:

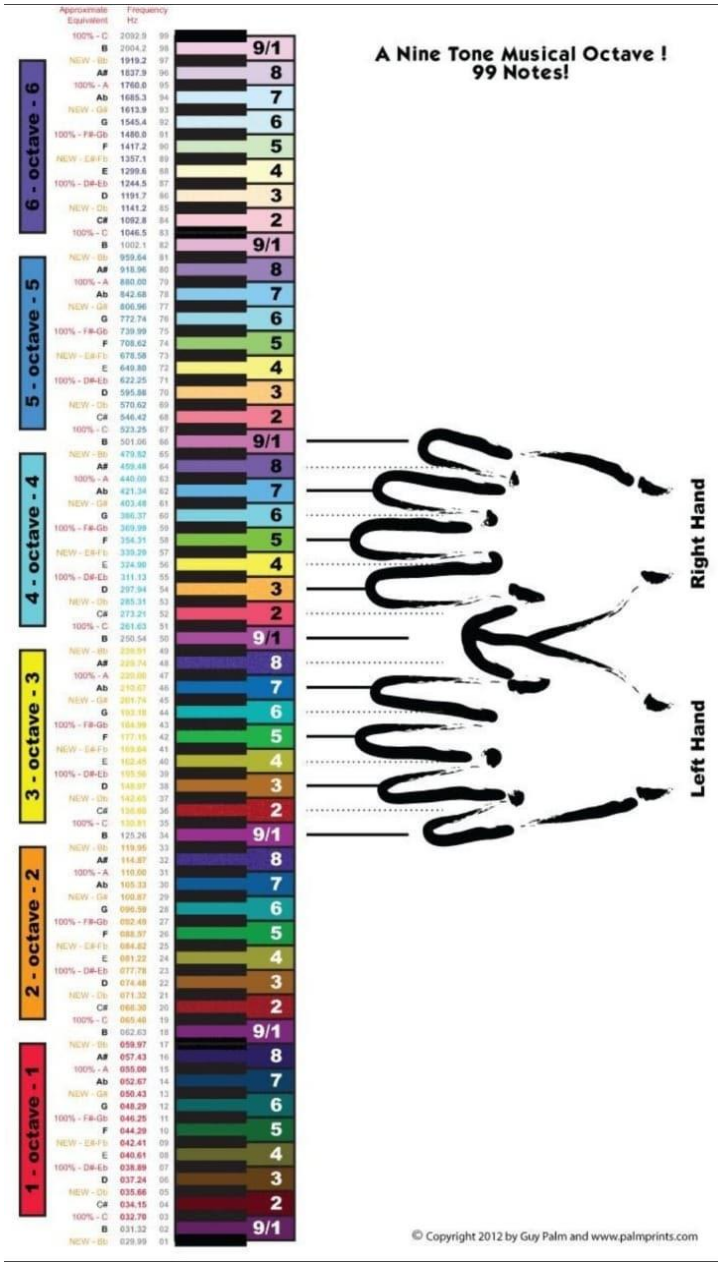
6.0 - PROTÓTIPO HIKAS – Versão Completa

Como falado anteriormente, o Tinkercad oferece muitas limitações, porém a simulação e comprovação do funcionamento pode ser feita por ele. Pensando na expansão do Hikas, um protótipo de sua versão completa foi feito pelo Fritzing.

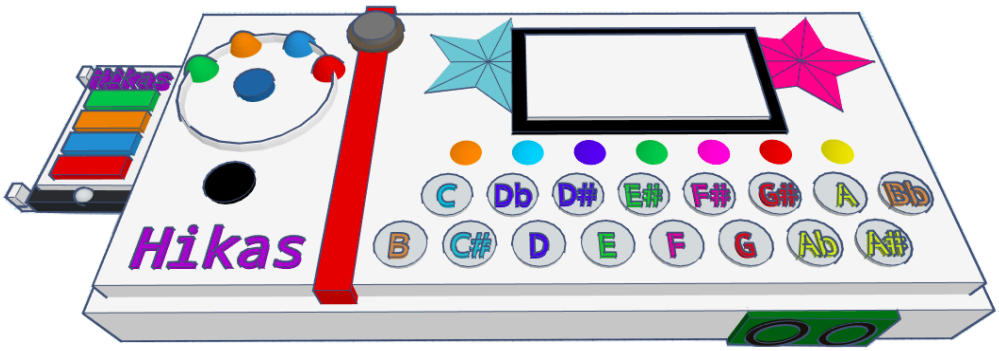
Essa versão oferece suporte ao App, com o bluetooth e uma gama maior de teclas. Podendo, agora sim, simular verdadeiramente um teclado musical.

Com essa expansão o Hikas consegue atingir 96 notas de um teclado musical com apenas 16 teclas e um regulador.

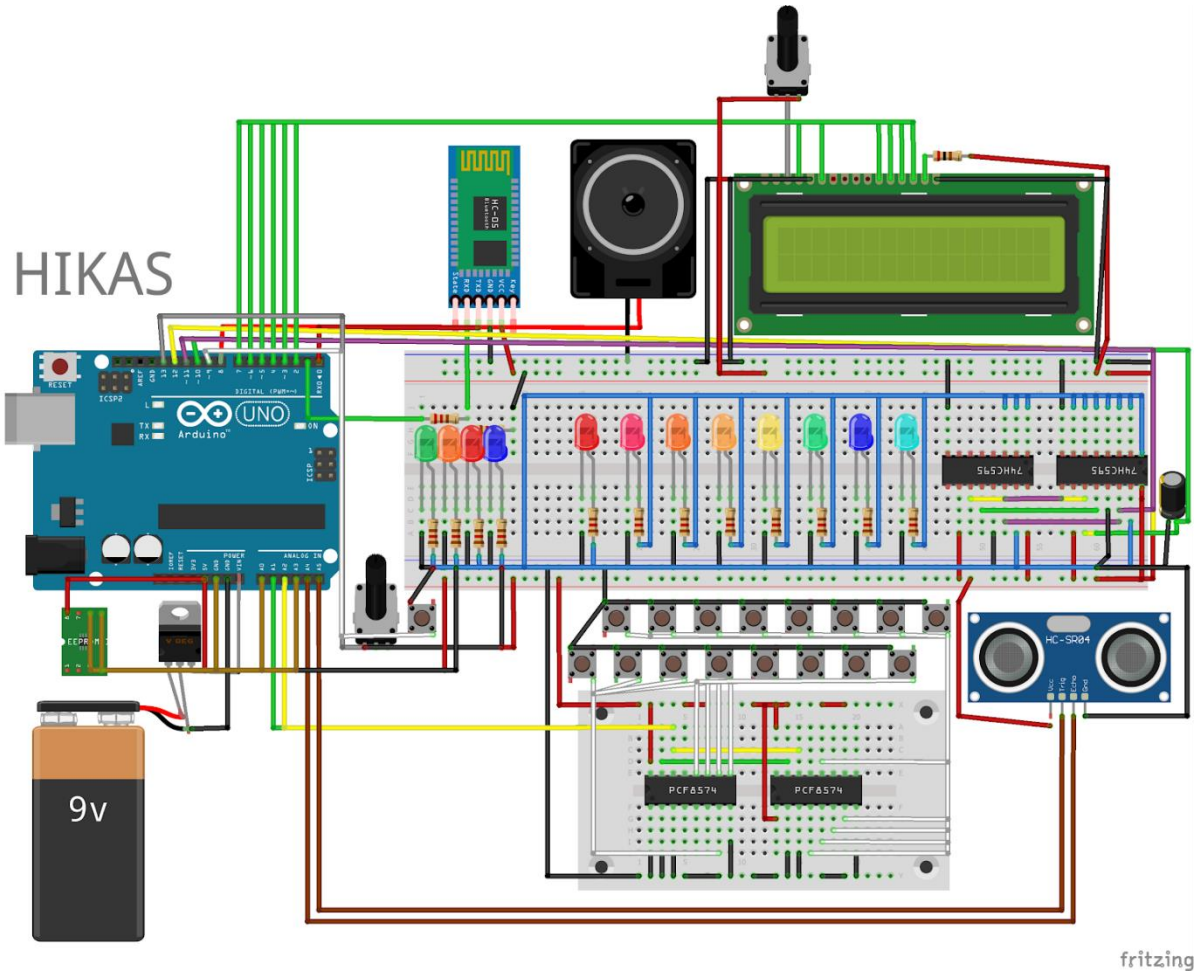
Teclado e frequências utilizados como base



Modelo 3D



Circuito (Fritzing)



Com o acréscimo de dois PCF8574, módulo bluetooth e 9 botões, o preço aproximado do Hikas versão completa é de R$150,36. Considerando o preço unitário de cada peça.

7.0 - O CÓDIGO DO PROJETO (VERSÃO UM)

O código foi dividido em partes para melhor entendimento. A seguir as declarações de variáveis e importações:

Declaração de variáveis

```
//Classe para botões – melhorando seu funcionamento dentro do jogo (Utilizado no modo Genius)
class PushButton {
  public:
    PushButton(byte pinBotao);
    void button_loop();
    bool pressed();
  private:
    unsigned long debounceBotao;
    bool estadoBotaoAnt = HIGH;
    bool apertado = false;
    byte pino;
};

PushButton::PushButton(byte pinBotao) {
  pinMode(pinBotao, INPUT_PULLUP);
  pino = pinBotao;
}

void PushButton::button_loop() {
  bool estadoBotao = digitalRead(pino);
  apertado = false;
  if (!estadoBotao && estadoBotaoAnt) {
        apertado = true;
  }
  estadoBotaoAnt = estadoBotao;
}

bool PushButton::pressed() {
  return apertado;
}
//------------------------GERAL-------------------------//
//Bibliotecas
#include <LiquidCrystal.h>
//#include <avr/sleep.h>
```

```
//ShiftRegister
#define pinSH_CP 12   //Pino Clock
#define pinST_CP 10  //Pino Latch
#define pinDS    11  //Pino Data
#define qtdeCI 2

// DEFINIÇÕES DE PINOS
#define pinBot1 8
#define pinBot2 0
#define pinBot3 15
#define pinBot4 16
#define pinBot5 17
#define pinBot6 18
#define pinBot7 19

#define pinBuzzer 13
#define pinPot 14
//Potenciometro
int pot;

//Usado no Millis
unsigned long agora = 0;

//Botão Multifunções
#define botaoPrincipal 9

//LCD
String mensagem = "HIKAS";
LiquidCrystal lcd(7,6,5,4,3,2);


//Sensor
/*#define sig 9
unsigned long distancia = 0;
unsigned long cm = 0;
*/
//------------------------LIVRE-----------------------//
//Notas Piano
int notasPadrao[] = {262,294,330,349,392,440,494};
int notas[7][7] = {
```

```
{33,37,41,44,49,55,62}, //Escala 01
{65,73,82,87,98,110,123}, //Escala 02
{131,147,165,175,196,220,247}, //Escala 03
{262,294,330,349,392,440,494}, //Escala 04
{523,587,659,698,784,880,988}, //Escala 05
{1047,1175,1319,1397,1568,1760,1976}, //Escala 06
{2093,2349,2637,2794,3136,3520,3951}}; //Escala 07

int escalaAtual = 4;
String nomeNotas[] = {"DO","RE","MI","FA","SOL","LA","SI"};
//------------------------GENIUS------------------------//
// Musica da Vitoria
int vitoria[] = {880,988,523,988,523,587,523,587,659,587,659,659};
#define duracaoVitoria 75

// Musica da Falha
int errado[] = {392,247};
int duracaoErrado[] = {500,750};

// DEFINIÇÕES
int nivelMax = 0;
// INSTANCIANDO OBJETOS
PushButton bot1(pinBot1);
PushButton bot2(pinBot2);
PushButton bot3(pinBot3);
PushButton bot4(pinBot4);
PushButton bot5(pinBot5);
PushButton bot6(pinBot6);
PushButton bot7(pinBot7);

// DECLARAÇÃO DE VARIÁVEIS
byte contador = 0;
unsigned int velocidade = 1000;
unsigned int tempoLimite;
bool timeOut = false;
String musicaEscolhida;
bool escolhaConcluida=false;
bool flagTrocaVez=true;
bool perdeu = false;
bool ganhou = false;
```

```
byte escala1,escala2;
bool escolhaConcluidaMenu;
byte escalas[20];

//MUSICA 01 -> MARIO THEME
//PADRÃO
/*int marioMelodiaLivre[] = {
  2637,2637,2637,2093,2637,3136,1568,2093,1568,
  1319,1760,1977,1760,1760,1568,2637,3136,
  3520,2794,3136,2637,2093,2349,1976,
  2093,1568,1319,1760,1976,1760,1760,
  1568,2637,3136,3520, 2794,3136,2637,2093,
  2349,1976
};

int marioMelodiaPadrao[] = {
 330,330,330,262,330,392,392,262,392,330,440,494,440,
 440,392,330,392,440,349,392,330,262,294,494,262,392,
 330,440,494,440,440,392,330,392,440,349,392,330,262,294,494
};
byte tempoMario[] = {
  9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,
  9,9,9,9,9,9,9,9,7,7,7,9,9,9,9,9,9,9,9,
  9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,
  7,7,7,9,9,9,9,9,9,9,9,9,9,9,9,
};*/

byte sequenciaMario[41];
int noteDuration;
int pauseEntreNotas;
byte sequenciaUsuario[41];
//MUSICA 02 -> Zelda(Por motivos de armazenamento não é possivel ter
//ambas as musicas simultaneamente)
int zeldaMelodiaLivre[] = {
  440,349,349,440,392,349,392,440,349,349,440,440,392,
  440,440,349,440,440,440,523,587,587,698,
  698,698,698,698,784,880, 880, 880, 880, 784, 698,784,
  698,698,698,587,587
};
//                C   D   E   F   G   A  B
```

```
//int notasPadrao[] = {262,294,330,349,392,440,494};
int zeldaMelodiaPadrao[] = {
  440,349,349,440,392,349,392,440,349,349,440,440,392,440,
  440,349,440,440,440,262,294,294,349,349,349,349,349,392,
  440,440,440,440,392,349,392,349,349,349,294,294,349,349,
  349,294,262,262,294,349,294
};
byte tempoZelda[] = {
  1,4,4,4,8,8,2,8,1,4,4,4,8,8,2,2,2,8,8,8,8,8,8,2,2,
  8,8,8,8,8,2,2,8,8,8,8,8,8,8,2,4,4,4,4,4,4,4,4,4,4,4,
  8,8,4,4,4,4,4,2,8,8,4,8,8,4,8,8,4,4,2,2,8,8,8,8,8,8,2,
  2,8,8,8,8,8,8,2,4,4,1,4,1,4,4,4,1,4,1,4,4,4,1,4,1,4,2,1,
  4,4,8,8,4,4,8,8,8,4,8,8,4,8,8,4,4,4
};
byte sequenciaZelda[40];

//-----------------------SCROLL-------------------------//
bool proxima;

//--------------------PROGRAMAÇÃO-----------------------//
int melodiaMus01[25];
int musicasGravadas = 0;
int contNotas = 0;
String opcaoEscolhida;
bool escolhaOpcaoConcluida = false;
bool melodiaPadrao = true;
bool musicaFinalizada = false;
byte sequenciaMus01[25];
bool gravado;
```

Setup

```
void setup() {
  Serial.begin(9600);
  //LCD
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(1, 0);
   lcd.print(mensagem);
  //Botoes
   pinMode(botaoPrincipal, INPUT_PULLUP);
   pinMode(pinBot1, INPUT);
```

```
  digitalWrite(pinBot1,HIGH);
  pinMode(pinBot2, INPUT);
  digitalWrite(pinBot2,HIGH);
  pinMode(pinBot3, INPUT);
  digitalWrite(pinBot3,HIGH);
  pinMode(pinBot4, INPUT);
  digitalWrite(pinBot4,HIGH);
  pinMode(pinBot5, INPUT);
  digitalWrite(pinBot5,HIGH);
  pinMode(pinBot6, INPUT);
  digitalWrite(pinBot6,HIGH);
  pinMode(pinBot7, INPUT);
  digitalWrite(pinBot7,HIGH);
 //ShiftRegister
  pinMode(pinSH_CP, OUTPUT);
  pinMode(pinST_CP, OUTPUT);
  pinMode(pinDS, OUTPUT);
  //Buzzer
  pinMode(pinBuzzer, OUTPUT);

  pot = map(analogRead(pinPot),0,1023,1,147);
  mapearEscala();
  velocidade = 100;
  tempoLimite = 1000;
  randomSeed(analogRead(0));
}
```

Modo Livre

```
void modoLivre(){
 while(digitalRead(botaoPrincipal)==HIGH){
 mapearEscala();
 int ultimaLed=0;
  while(digitalRead(pinBot1) == LOW)
  {
   tone(pinBuzzer,notas[escalaAtual][0]);
   ligarLed(0);
   ultimaLed = 0;
   print(nomeNotas[0]+"_"+(escalaAtual+1)+"        ",0,1);
  }

  while(digitalRead(pinBot2) == LOW)
```

```
{
  tone(pinBuzzer,notas[escalaAtual][1]);
   ligarLed(1);
   ultimaLed=1;
   print(nomeNotas[1]+"_"+(escalaAtual+1)+"        ",0,1);
}
while(digitalRead(pinBot3) == LOW)
{
  tone(pinBuzzer,notas[escalaAtual][2]);
   ligarLed(2);
   ultimaLed=2;
   print(nomeNotas[2]+"_"+(escalaAtual+1)+"        ",0,1);
}
while(digitalRead(pinBot4) == LOW)
{
  tone(pinBuzzer,notas[escalaAtual][3]);
   ligarLed(3);
   ultimaLed=3;
   print(nomeNotas[3]+"_"+(escalaAtual+1)+"        ",0,1);
}
while(digitalRead(pinBot5) == LOW)
{
  tone(pinBuzzer,notas[escalaAtual][4]);
   ligarLed(4);
   ultimaLed=4;
   print(nomeNotas[4]+"_"+(escalaAtual+1)+"        ",0,1);
}
while(digitalRead(pinBot6) == LOW)
{
  tone(pinBuzzer,notas[escalaAtual][5]);
   ligarLed(5);
   ultimaLed=5;
   print(nomeNotas[5]+"_"+(escalaAtual+1)+"        ",0,1);
}
while(digitalRead(pinBot7) == LOW)
{
  tone(pinBuzzer,notas[escalaAtual][6]);
   ligarLed(6);
   ultimaLed=6;
   print(nomeNotas[6]+"_"+(escalaAtual+1)+"        ",0,1);
```

```
  }

  noTone(pinBuzzer);
  desligarLed(ultimaLed);
  }
}
void mapearEscala(){
  pot = map(analogRead(pinPot),0,1023,1,147);
  if(pot <=21){ escalaAtual = 0;}
  else if(pot<=42){ escalaAtual = 1;}
  else if(pot<=63){ escalaAtual = 2;}
  else if(pot<=84){ escalaAtual = 3;}
  else if(pot<=105){ escalaAtual = 4;}
  else if(pot<=126){ escalaAtual = 5;}
  else if(pot<=147){ escalaAtual = 6;}
}
```

Modo Genius

```
void zerarGenius(){
  contador = 0;
  timeOut = false;
  escolhaConcluida=false;
  flagTrocaVez=true;
  perdeu = false;
  ganhou = false;
}
void modoGenius(){
 bool velocidadeEscolhida = false;
   while(velocidadeEscolhida==false){
    print("Escolha        ",0,0);
    print("Velocidade C-OK",0,1);
  if(digitalRead(pinBot1) == LOW){
    velocidade = map(analogRead(pinPot),0,1023,40,100);
    tempoLimite = map(velocidade,400,2000,1500,3000);
    velocidadeEscolhida=true;
  }
  }
  while(escolhaConcluida==false){
   menu();
  }
  while(!perdeu && !ganhou && digitalRead(botaoPrincipal)==HIGH){
```

```
  if(musicaEscolhida=="Mario"){
   /* if(melodiaPadrao==true){
      jogarPadrao(sequenciaMario,marioMelodiaPadrao,tempoMario);
    }
    else{
      jogarLivre(sequenciaMario,marioMelodiaLivre,tempoMario);
    }*/
    }
    else if(musicaEscolhida=="Zelda"){
     if(melodiaPadrao==true){
     jogarPadrao(sequenciaZelda,zeldaMelodiaPadrao,tempoZelda);
     }
     else{
      jogarLivre(sequenciaZelda,zeldaMelodiaLivre,tempoZelda);
     }
    }
    else if(musicaEscolhida=="Mus01"){
      jogarLivre(sequenciaMus01,melodiaMus01,tempoZelda);//vetor de tempo padrao
   }
  }
}
void menu(){
  Serial.print(musicasGravadas);
  //if(musicasGravadas>=1){
  if(gravado==true){
  print("D-Mario     F ->",0,0);
  print("E-Zelda         ",0,1);
  if(digitalRead(pinBot2) == LOW){
    musicaEscolhida = "Mario";
    nivelMax = 41;
    escolhaConcluida=true;
  }
  if(digitalRead(pinBot3) == LOW){
    musicaEscolhida = "Zelda";
    nivelMax = 40;
    escolhaConcluida=true;
  }

  if(digitalRead(pinBot4) == LOW){
    while(escolhaConcluida==false){
```

```
    telaUser();
    }
  }
  }else{
  print("D-Mario       ",0,0);
  print("E-Zelda        ",0,1);
  if(digitalRead(pinBot2) == LOW){
    musicaEscolhida = "Mario";
    nivelMax = 41;
    escolhaConcluida=true;
  }
  if(digitalRead(pinBot3) == LOW){
    musicaEscolhida = "Zelda";
    nivelMax = 40;
    escolhaConcluida=true;
  }
  }
}
void telaUser(){
   print("MusicaUser    ",0,0);
   print("G-Mus01       ",0,1);
  if(digitalRead(pinBot5) == LOW){
    musicaEscolhida = "Mus01";
    escolhaConcluida=true;
  }
}

void jogarPadrao(byte sequencia[], int melodia[], byte tempo[]){
  trocaVez();
  jogadaArduino(sequencia,melodia);
  musicaAteAgora(sequencia,melodia,tempo);
  trocaVez();
  vezUsuario(melodia);

 if (confere(sequencia) == 0) {
      print("DERROTA",0,0);
      perdeu=true;
      for (byte i = 0 ; i < 7; i++) {
        ligarLed(i);
        delayMillis(10);
```

```
      desligarLed(i);
     }
  }
  if (contador == nivelMax) {
   print("VITORIA",0,0);
   ganhou=true;
   for (byte nota = 0; nota < 12 ; nota++) {
    tone(pinBuzzer, vitoria[nota], duracaoVitoria);
    delayMillis(duracaoVitoria);
   }
    for (byte i = 0 ; i < 7; i++) {
     ligarLed(i);
     delayMillis(10);
     desligarLed(i);
    }
  }
}

//FUNÇÕES - MODO PADRÃO
void musicaAteAgora(byte sequencia[], int melodia[], byte tempo[]) {
  for (byte i = 0 ; i < contador; i++) {
   noteDuration = 1000 / tempo[i];
   print(nomeNotas[sequencia[i]]+"_"+(acharEscala(melodia[i])+1)+"        ",0,1);
   ligarLed(sequencia[i]);
   //delayMillis(20);
   tone(pinBuzzer, melodia[i], velocidade);
   delayMillis(velocidade);
   desligarLed(sequencia[i]);
   //pauseEntreNotas = noteDuration * 1.30;
   //delayMillis(pauseEntreNotas);
   delayMillis(velocidade/2);
  }
}
void jogadaArduino(byte sequencia[], int melodia[]){
  sequencia[contador] = identificaBotao(melodia[contador]);
  contador++;
  sequencia[contador] = identificaBotao(melodia[contador]);
  contador++;
}
```

```
void vezUsuario(int melodia[]) {
  unsigned long controleTempo = millis();
  byte leitura = 0;
  while (millis() < controleTempo + tempoLimite && leitura < contador) {
    bot1.button_loop(); bot2.button_loop(); bot3.button_loop(); bot4.button_loop();
    bot5.button_loop(); bot6.button_loop(); bot7.button_loop();
    if (bot1.pressed()) {
      sequenciaUsuario[leitura] = 0;
            controleTempo = millis();
      leitura++;
      ligarLed(0);
      print(nomeNotas[0]+"_"+4+"        ",0,1);
      tone(pinBuzzer, notasPadrao[0], velocidade);
      delayMillis(velocidade);
      desligarLed(0);
    } else if (bot2.pressed()) {
      sequenciaUsuario[leitura] = 1;
            controleTempo = millis();
      leitura++;
      ligarLed(1);
      print(nomeNotas[1]+"_"+4+"        ",0,1);
      tone(pinBuzzer, notasPadrao[1], velocidade);
      delayMillis(velocidade);
      desligarLed(1);
    } else if (bot3.pressed()) {
      sequenciaUsuario[leitura] = 2;
            controleTempo = millis();
      leitura++;
      ligarLed(2);
      print(nomeNotas[2]+"_"+4+"        ",0,1);
      tone(pinBuzzer, notasPadrao[2], velocidade);
      delayMillis(velocidade);
      desligarLed(2);
    } else if (bot4.pressed()) {
      sequenciaUsuario[leitura] = 3;
            controleTempo = millis();
      leitura++;
      ligarLed(3);
      print(nomeNotas[3]+"_"+4+"        ",0,1);
      tone(pinBuzzer, notasPadrao[3], velocidade);
```

```
    delayMillis(velocidade);
    desligarLed(3);
   }else if (bot5.pressed()) {
    sequenciaUsuario[leitura] = 4;
            controleTempo = millis();
    leitura++;
    ligarLed(4);
    print(nomeNotas[4]+"_"+4+"       ",0,1);
    tone(pinBuzzer, notasPadrao[4], velocidade);
    delayMillis(velocidade);
    desligarLed(4);
   }else if (bot6.pressed()) {
    sequenciaUsuario[leitura] = 5;
            controleTempo= millis();
    leitura++;
    ligarLed(5);
    print(nomeNotas[5]+"_"+4+"       ",0,1);
    tone(pinBuzzer, notasPadrao[5], velocidade);
    delayMillis(velocidade);
    desligarLed(5);
   }else if (bot7.pressed()) {
    sequenciaUsuario[leitura] = 6;
            controleTempo = millis();
    leitura++;
    ligarLed(6);
    print(nomeNotas[6]+"_"+4+"       ",0,1);
    tone(pinBuzzer, notasPadrao[6], velocidade);
    delayMillis(velocidade);
    desligarLed(6);
   }

   if (millis() > controleTempo + tempoLimite){
    timeOut = true;
    break;
   }

  }

  delayMillis(velocidade/2);
}
```

```
// FUNÇÃO PARA INDICAR A TROCA DE VEZ
void trocaVez() {
  if(flagTrocaVez==true){
    print("VEZ HIKAS   ",0,0);
    flagTrocaVez=false;
  }else{
    print("SUA VEZ     ",0,0);
    flagTrocaVez=true;
  }
}

bool confere(byte sequencia[]) {
  bool resultado = true;
  for (byte i = 0; i < contador; i++) {
    if (sequencia[i] != sequenciaUsuario[i]) {
      resultado = false;
      //som de erro
      for (byte nota = 0; nota < 2 ; nota++) {
        tone(pinBuzzer, errado[nota], duracaoErrado[nota]);
        delayMillis(duracaoErrado[nota]);
      }
      break;
    }

  }
  return resultado;
}
//FUNÇÕES - MODO LIVRE
void jogarLivre(byte sequencia[], int melodia[], byte tempo[]){
  trocaVez();
  jogadaArduinoLivre(sequencia,melodia);
  musicaAteAgoraLivre(sequencia,melodia,tempo);
  trocaVez();
  vezUsuarioLivre();

 if (confere(sequencia) == 0) {
      print("    DERROTA     ",0,0);
      print("                ",0,1);
      perdeu=true;
```

```
      for (byte i = 0 ; i < 7; i++) {
        ligarLed(i);
        delayMillis(10);
        desligarLed(i);
      }
  }
  if (contador == nivelMax) {
    print("     VITORIA    ",0,0);
    print("                ",0,1);
    ganhou=true;
    for (byte nota = 0; nota < 12 ; nota++) {
      tone(pinBuzzer, vitoria[nota], duracaoVitoria);
      delayMillis(duracaoVitoria);
    }
      for (byte i = 0 ; i < 7; i++) {
        ligarLed(i);
        delayMillis(10);
        desligarLed(i);
      }
  }
}
byte identificaBotao(int nota){
  if(nota==notas[escalaAtual][0]){ return 0;}
  else if(nota==notas[escalaAtual][1]){ return 1;}
  else if(nota==notas[escalaAtual][2]){ return 2;}
  else if(nota==notas[escalaAtual][3]){ return 3;}
  else if(nota==notas[escalaAtual][4]){ return 4;}
  else if(nota==notas[escalaAtual][5]){ return 5;}
  else if(nota==notas[escalaAtual][6]){ return 6;}
}
void jogadaArduinoLivre(byte sequencia[], int melodia[]){

  escalas[contador] = acharEscala(melodia[contador]);
  escalaAtual = escalas[contador];
  sequencia[contador] = identificaBotao(melodia[contador]);
  contador++;

  escalas[contador] = acharEscala(melodia[contador]);
  escalaAtual = escalas[contador];
  sequencia[contador] = identificaBotao(melodia[contador]);
```

```
  contador++;
}

byte acharEscala(int nota){
  if(nota<=62){
    return 0;
  }else if(nota<=123){
    return 1;
  }else if(nota<=247){
    return 2;
  }else if(nota<=494){
    return 3;
  }else if(nota<=988){
    return 4;
  }else if(nota<=1976){
    return 5;
  }else{
    return 6;
  }
}
void vezUsuarioLivre(){
  unsigned long controleTempo = millis();
  byte leitura = 0;
  int i =0;

  while (millis() < controleTempo + tempoLimite && leitura < contador) {
    bot1.button_loop(); bot2.button_loop(); bot3.button_loop(); bot4.button_loop();
    bot5.button_loop(); bot6.button_loop(); bot7.button_loop();
    if (bot1.pressed()) {
      sequenciaUsuario[leitura] = 0;
            controleTempo = millis();
      leitura++;
      ligarLed(0);
      print(nomeNotas[0]+"_"+(escalas[i]+1)+"        ",0,1);
      tone(pinBuzzer, notas[escalas[i]][0], velocidade);

      delayMillis(velocidade);
      desligarLed(0);
      i++;
    } else if (bot2.pressed()) {
```

```
  sequenciaUsuario[leitura] = 1;
        controleTempo = millis();
  leitura++;
  ligarLed(1);
  print(nomeNotas[1]+"_"+(escalas[i]+1)+"       ",0,1);
  tone(pinBuzzer, notas[escalas[i]][1], velocidade);
  i++;

  delayMillis(velocidade);
  desligarLed(1);
 } else if (bot3.pressed()) {
  sequenciaUsuario[leitura] = 2;
        controleTempo = millis();
  leitura++;
  ligarLed(2);
  print(nomeNotas[2]+"_"+(escalas[i]+1)+"       ",0,1);
  tone(pinBuzzer, notas[escalas[i]][2], velocidade);
  i++;

  delayMillis(velocidade);
  desligarLed(2);
 } else if (bot4.pressed()) {
  sequenciaUsuario[leitura] = 3;
        controleTempo = millis();
  leitura++;
  ligarLed(3);
  print(nomeNotas[3]+"_"+(escalas[i]+1)+"       ",0,1);
  tone(pinBuzzer, notas[escalas[i]][3], velocidade);
  i++;

  delayMillis(velocidade);
  desligarLed(3);
 }else if (bot5.pressed()) {
  sequenciaUsuario[leitura] = 4;
        controleTempo = millis();
  leitura++;
  ligarLed(4);
  print(nomeNotas[4]+"_"+(escalas[i]+1)+"       ",0,1);
  tone(pinBuzzer, notas[escalas[i]][4], velocidade);
  i++;
```

```
    delayMillis(velocidade);
    desligarLed(4);
   }else if (bot6.pressed()) {
    sequenciaUsuario[leitura] = 5;
          controleTempo= millis();
    leitura++;
    ligarLed(5);
    print(nomeNotas[5]+"_"+(escalas[i]+1)+"       ",0,1);
    tone(pinBuzzer, notas[escalas[i]][5], velocidade);
    i++;

    delayMillis(velocidade);
    desligarLed(5);
   }else if (bot7.pressed()) {
    sequenciaUsuario[leitura] = 6;
          controleTempo = millis();
    leitura++;
    ligarLed(6);
    print(nomeNotas[6]+"_"+(escalas[i]+1)+"       ",0,1);
    tone(pinBuzzer, notas[escalas[i]][6], velocidade);
    i++;

    delayMillis(velocidade);
    desligarLed(6);
   }

   if (millis() > controleTempo + tempoLimite){
    timeOut = true;
    break;
   }

  }

  delayMillis(velocidade/2);
}
void musicaAteAgoraLivre(byte sequencia[], int melodia[], byte tempo[]) {
  for (byte i = 0 ; i < contador; i++) {
   noteDuration = 1000 / tempo[i];
   print(nomeNotas[sequencia[i]]+"_"+(acharEscala(melodia[i])+1)+"       ",0,1);
```

```
    ligarLed(sequencia[i]);
    delayMillis(20);
    tone(pinBuzzer, melodia[i], noteDuration);
    desligarLed(sequencia[i]);
    pauseEntreNotas = noteDuration * 1.30;
    delayMillis(pauseEntreNotas);
  }
}
```

Modo Scroll

```
void modoScroll(){
  float vel= 200;
  bool velocidadeEscolhida = false;
  escolhaConcluida = false;
  perdeu = false;
  ganhou = false;
  while(velocidadeEscolhida==false){
    print("Escolha        ",0,0);
    print("Velocidade C-OK",0,1);
  if(digitalRead(pinBot1) == LOW){
    vel= map(analogRead(pinPot),0,1023,60,250);
    velocidadeEscolhida=true;
  }
  }
  while(escolhaConcluida==false){
   menu();
  }

  while(perdeu==false && ganhou==false && digitalRead(botaoPrincipal)==HIGH){
   if(musicaEscolhida=="Mario"){
    /*if(melodiaPadrao==true){
      scrollMusic(vel,marioMelodiaPadrao,nivelMax);
     }
     else{
      scrollMusic(vel,marioMelodiaLivre,nivelMax);
     }*/
     }
    else if(musicaEscolhida=="Zelda"){
      if(melodiaPadrao==true){
      scrollMusic(vel,zeldaMelodiaPadrao,nivelMax);
     }
```

```
     else{
      scrollMusic(vel,zeldaMelodiaLivre,nivelMax);
     }
    }
    else if(musicaEscolhida=="Mus01"){
     scrollMusic(vel,melodiaMus01,nivelMax);
   }
  }
}
void scroll(float ScrollSpeed,int pinBot, String nota, byte escala,byte pos){
  bool flag = false;
  int count = 1;
  lcd.begin(16, 2);
  lcd.setCursor(16,pos);

  lcd.print(nota+"_"+(escala+1));
  Serial.print(nota);

    while (digitalRead(pinBot)==HIGH && count < 18) {
    delay(ScrollSpeed);
    lcd.scrollDisplayLeft();
    count++;
     if(digitalRead(pinBot)==LOW){
       ligarLed(identificaNota(nota));
       tone(pinBuzzer,notas[escala][identificaNota(nota)],300);
       delayMillis(40);
       desligarLed(identificaNota(nota));
       proxima = true;
      }
  }

  if  (digitalRead(pinBot)==HIGH){
    lcd.begin(16, 2);
    lcd.setCursor(3,0);
    perdeu = true;

    lcd.print("GAME OVER!!");
    delay(2000);
    lcd.setCursor(0,1);
    lcd.print("Jogue Novamente");
```

```
    delay(2000);
     lcd.print("               ");
  }
  else{ }

}
int identificaNota(String nota){
  if(nota=="DO"){ return 0;}
  else if(nota=="RE"){ return 1;}
  else if(nota=="MI"){ return 2;}
  else if(nota=="FA"){ return 3;}
  else if(nota=="SOL"){ return 4;}
  else if(nota=="LA"){ return 5;}
  else if(nota=="SI"){ return 6;}
}

void scrollMusic(float vel,int melodia[], int tam){
  int nota;
  int i = 0;
  byte escala;
  byte pos;
  for(i=0;i<tam&&perdeu==false&&ganhou==false;i++){

    nota = melodia[i];
    proxima = false;
    escala = acharEscala(nota);
    pos = random(2);

    if(nota==notas[escala][0]){
     while(proxima==false&&perdeu==false){
     if(digitalRead(botaoPrincipal)==LOW){
            perdeu = true;
            break;
     }
      scroll(vel,pinBot1,"DO",escala,pos);
     }
    }
    else if(nota==notas[escala][1]){
       while(proxima==false&&perdeu==false){
        if(digitalRead(botaoPrincipal)==LOW){
```

```
        perdeu = true;
        break;
 }
  scroll(vel,pinBot2,"RE",escala,pos);
}
}
 else if(nota==notas[escala][2]){
  while(proxima==false&&perdeu==false){
   if(digitalRead(botaoPrincipal)==LOW){
        perdeu = true;
        break;
   }
   scroll(vel,pinBot3,"MI",escala,pos);
  }
}
 else if(nota==notas[escala][3]){
 while(proxima==false&&perdeu==false){
  if(digitalRead(botaoPrincipal)==LOW){
        perdeu = true;
        break;
 }
   scroll(vel,pinBot4,"FA",escala,pos);

 }
}
 else if(nota==notas[escala][4]){
  while(proxima==false&&perdeu==false){
   if(digitalRead(botaoPrincipal)==LOW){
        perdeu = true;
        break;
 }
   scroll(vel,pinBot5,"SOL",escala,pos);
  }
}
 else if(nota==notas[escala][5]){
  while(proxima==false&&perdeu==false){
   if(digitalRead(botaoPrincipal)==LOW){
        perdeu = true;
        break;
 }
```

```
      scroll(vel,pinBot6,"LA",escala,pos);
     }
    }
     else if(nota==notas[escala][6]){
      while(proxima==false&&perdeu==false){
       if(digitalRead(botaoPrincipal)==LOW){
            perdeu = true;
            break;
     }
       scroll(vel,pinBot7,"SI",escala,pos);
      }
      }
    if(i==tam-1){
     ganhou = true;
    }
  }
  if(ganhou==true){
   lcd.begin(16, 2);
   lcd.setCursor(2,0);
   lcd.print("  VITORIA!!  ");
   delay(2000);
   lcd.setCursor(0,1);
   lcd.print("Jogue Novamente");
   delay(2000);
  }
}
```

Modo Programação

```
void modoProgramacao(){
  while(escolhaOpcaoConcluida==false){
  menuProg();
  }
  if(opcaoEscolhida=="Gravar"){
   musicasGravadas=1;
   gravado = true;
   print("Gravando...     ",0,0);
   print("                ",0,1);
   while(musicaFinalizada == false){
      gravarMusica();
   }
   print("Musica Gravada  ",0,0);
```

```
    print("                ",0,1);
    tocarMusicaGravada();
   }
   if(opcaoEscolhida=="Editar"){
     print("Genius Editado",0,0);
     print("                ",0,1);
   }

}
  void menuProg(){
  print("C-Editar     ",0,0);
  print("D-Gravar     ",0,1);

  if(digitalRead(pinBot1) == LOW){
    ligarLed(0);
    delayMillis(40);
    desligarLed(0);
    while(escolhaOpcaoConcluida==false){//ou retornar
      menuEdicao();
    }
  }
  if(digitalRead(pinBot2) == LOW){
    ligarLed(1);
    delayMillis(40);
    desligarLed(1);
    opcaoEscolhida = "Gravar";
    escolhaOpcaoConcluida=true;
  }
}

void menuEdicao(){
  print("E-Tom Padrao",0,0);
  print("F-Tom Livre",0,1);
  if(digitalRead(pinBot3) == LOW){
    editar("Padrao");
    ligarLed(2);
    delayMillis(40);
    desligarLed(2);
    escolhaOpcaoConcluida=true;
    opcaoEscolhida = "Editar";
```

```
  }
  if(digitalRead(pinBot4) == LOW){
    editar("Livre");
    ligarLed(3);
    delayMillis(40);
    desligarLed(3);
    escolhaOpcaoConcluida=true;
    opcaoEscolhida = "Editar";

  }
}
void editar(String edicao){
  if(edicao=="Padrao"){
   melodiaPadrao = true;
  }else{
   melodiaPadrao = false;
  }
}
void gravarMusica(){
 mapearEscala();
 int ultimaLed=-1;

  if(digitalRead(botaoPrincipal)==LOW){
     musicaFinalizada=true;
   }else{
   while(digitalRead(pinBot1) == LOW)
   {
     if(ultimaLed!=0){
     tone(pinBuzzer,notas[escalaAtual][0]);
     ligarLed(0);
     ultimaLed=0;
     print(nomeNotas[0]+"_"+(escalaAtual+1)+"        ",0,1);
     melodiaMus01[contNotas]=notas[escalaAtual][0];
     //sequenciaLedMus01[contNotas]=0;
     contNotas++;
     }
   }

   while(digitalRead(pinBot2) == LOW)
   {
```

```
   if(ultimaLed!=1){
   tone(pinBuzzer,notas[escalaAtual][1]);
   ligarLed(1);
   ultimaLed=1;
   print(nomeNotas[1]+"_"+(escalaAtual+1)+"        ",0,1);
   melodiaMus01[contNotas]=notas[escalaAtual][1];
   //sequenciaLedMus01[contNotas]=1;
   contNotas++;
   }
}
while(digitalRead(pinBot3) == LOW)
{
  if(ultimaLed!=2){
  tone(pinBuzzer,notas[escalaAtual][2]);
   ligarLed(2);
   ultimaLed=2;
   print(nomeNotas[2]+"_"+(escalaAtual+1)+"        ",0,1);
   melodiaMus01[contNotas]=notas[escalaAtual][2];
   //sequenciaLedMus01[contNotas]=2;
   contNotas++;
  }
}
while(digitalRead(pinBot4) == LOW)
{
  if(ultimaLed!=3){
  tone(pinBuzzer,notas[escalaAtual][3]);
   ligarLed(3);
   ultimaLed=3;
   print(nomeNotas[3]+"_"+(escalaAtual+1)+"        ",0,1);
   melodiaMus01[contNotas]=notas[escalaAtual][3];
  // sequenciaLedMus01[contNotas]=3;
   contNotas++;
  }
}
while(digitalRead(pinBot5) == LOW)
{
  if(ultimaLed!=4){
  tone(pinBuzzer,notas[escalaAtual][4]);
   ligarLed(4);
   ultimaLed=4;
```

```
    print(nomeNotas[4]+"_"+(escalaAtual+1)+"        ",0,1);
    melodiaMus01[contNotas]=notas[escalaAtual][4];
   // sequenciaLedMus01[contNotas]=4;
    contNotas++;
   }
  }
  while(digitalRead(pinBot6) == LOW)
  {
   if(ultimaLed!=5){
   tone(pinBuzzer,notas[escalaAtual][5]);
    ligarLed(5);
    ultimaLed=5;
    print(nomeNotas[5]+"_"+(escalaAtual+1)+"        ",0,1);
    melodiaMus01[contNotas]=notas[escalaAtual][5];
    //sequenciaLedMus01[contNotas]=5;
    contNotas++;
   }
  }
  while(digitalRead(pinBot7) == LOW)
  {
   if(ultimaLed!=6){
    tone(pinBuzzer,notas[escalaAtual][6]);
    ligarLed(6);
    ultimaLed=6;
    print(nomeNotas[6]+"_"+(escalaAtual+1)+"        ",0,1);
    melodiaMus01[contNotas]=notas[escalaAtual][6];
    //sequenciaLedMus01[contNotas]=6;
    contNotas++;
   }
  }

  noTone(pinBuzzer);
  desligarLed(ultimaLed);
  }
}
void tocarMusicaGravada(){
  nivelMax = contNotas;

for (byte i = 0 ; i < contNotas; i++) {
   Serial.print(i);
```

```
    escalaAtual = acharEscala(melodiaMus01[i]);
    ligarLed(identificaBotao(melodiaMus01[i]));
    tone(pinBuzzer, melodiaMus01[i], 100);
    delayMillis(20);
    desligarLed(identificaBotao(melodiaMus01[i]));
    delayMillis(100);
  }
}
```

shiftRegister 74HC595

```
//Ligar e Desligar Leds
void ligarLed(int led){
  ci74HC595Write(led, HIGH);
}
void desligarLed(int led){
  ci74HC595Write(led, LOW);
}
//74HC595
void ci74HC595Write(byte pino, bool estado) {
static byte ciBuffer[qtdeCI];

bitWrite(ciBuffer[pino / 8], pino % 8, estado);

digitalWrite(pinST_CP, LOW); //Inicia a Transmissão

digitalWrite(pinDS, LOW);    //Apaga Tudo para Preparar Transmissão
digitalWrite(pinSH_CP, LOW);

for (int nC = qtdeCI-1; nC >= 0; nC--) {
   for (int nB = 7; nB >= 0; nB--) {

      digitalWrite(pinSH_CP, LOW);  //Baixa o Clock

      digitalWrite(pinDS,  bitRead(ciBuffer[nC], nB) );     //Escreve o BIT

      digitalWrite(pinSH_CP, HIGH); //Eleva o Clock
      digitalWrite(pinDS, LOW);     //Baixa o Data para Previnir Vazamento
   }
}

digitalWrite(pinST_CP, HIGH);  //Finaliza a Transmissão
```

```
}
```

Botão Principal

```
byte digitalReadOnce(byte val){
 static byte lastVal = HIGH;
  static unsigned long m = 0;
  if(lastVal != val && millis() > (m+100)){
   lastVal = val;
    m = millis();
    return lastVal;
  }
  return HIGH;
}

int getCommand(){
  static unsigned long m1 = 0;//millis no momento que inicia de pressionar
  static unsigned long m2 = 0;//millis apos soltar
  static byte count =0;

  byte r = digitalRead(botaoPrincipal);

  if(digitalReadOnce(r)==LOW){
   m1 = millis();
    count++;
  }
  if(r==LOW){
   m2 = millis();
  }else{
    if(!(m2>0 && m2-m1 < 500)){
     count = 0;
      m1 =0;
      m2 = 0;
    }
    if(m2>0 && millis()-m2>700){
      byte c = count;
      count = 0;
      m1 = 0;
      m2 = 0;
      return c;
```

```
    }

  }
  return 0;
}
```

Sensor Ultrassônico

```
/*
long leituraDoSensor() {
  //o Sensor utilizado no projeto possui o canal Sig,onde emite e recebe o sinal
  pinMode(sig, OUTPUT); //inicia a emição por um tempo
  digitalWrite(sig, HIGH);
  delayMicroseconds(5);
  digitalWrite(sig, LOW); //desliga a emição

  pinMode(sig, INPUT); //recebe o sinal
  distancia = pulseIn(sig, HIGH); //armazena o sinal
  return distancia;
}
bool sensor() {
  leituraDoSensor();
  distancia = 5; //limite maximo
  cm = 0.01723 * leituraDoSensor(); //transforma em cm

  //compara a distancia do objeto com o limite fornecido
  if (cm >= distancia) {
    //caso o sinal exceda o limite vai dar falso
    result = false;

  } else {
    //se for menor ou  igual ao limite,ou seja se estiver proximo ele da true e continua
    result = true;
  }
  return result;
}
void acordar() {
  //se o sensor detectar alguma coisa ,ele acorda

  // Desabilita o sleep
  sleep_disable();
```

```
  // Desabilita a interrupção
  detachInterrupt(0);

}
void dormir() {
  //define o modo sleep
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);

  //habilita o modo
  sleep_enable();

  //interrupções
  attachInterrupt(digitalPinToInterrupt(pinBot1), acordar, LOW);
  attachInterrupt(digitalPinToInterrupt(pinBot2), acordar, LOW);
  attachInterrupt(digitalPinToInterrupt(pinBot3), acordar, LOW);
  attachInterrupt(digitalPinToInterrupt(pinBot4), acordar, LOW);
  attachInterrupt(digitalPinToInterrupt(pinBot5), acordar, LOW);
  attachInterrupt(digitalPinToInterrupt(pinBot6), acordar, LOW);
  attachInterrupt(digitalPinToInterrupt(pinBot7), acordar, LOW);
  attachInterrupt(digitalPinToInterrupt(botaoPrincipal), acordar, LOW);

  //manda ele dormir
  sleep_cpu();
}
*/
```

Outros

```
//Delay
void delayMillis(int tempo){
 agora = millis();
 while(millis() < (agora + tempo)) {
    //Pausa
 }
}
//lCD
void print(String msg,int c,int l){
 lcd.setCursor(c, l);
 lcd.print(msg);
}
```

Loop

```
void loop() {
```

```
/*OBS: Por questão de armazenamento do Tinkercad para
utilizar o modo programação o descomente e comente dois outros
modos (Comente o if)
 */
//if (sensor() == true) { //Sensor ultrassonico
  //  acordar();
int command = getCommand();
  bool flag = false;

  if(command==1){
      zerarGenius();
      print("   Modo Livre   ",0,0);
       print("                ",0,1);
      ligarLed(8);
      desligarLed(9);
      desligarLed(10);
      desligarLed(11);
      delayMillis(100);
      modoLivre();
  }
  if(command==2){
   print("   Modo Genius  ",0,0);
      print("                ",0,1);
      zerarGenius();
      ligarLed(9);
      desligarLed(8);
      desligarLed(10);
      desligarLed(11);
      delayMillis(100);
      modoGenius();
  }
  if(command==3){
    zerarGenius();
     print("     Scroll    ",0,0);
     print("                ",0,1);
     ligarLed(10);
     desligarLed(11);
     desligarLed(9);
     desligarLed(8);
     delayMillis(100);
```

```
    modoScroll();
  }
 /*if(command==4){
    zerarGenius();
    print("  Programacao   ",0,0);
    ligarLed(11);
    desligarLed(10);
    desligarLed(9);
    desligarLed(8);
    delayMillis(100);
    modoProgramacao();
  }*/
  /*}else{
  dormir();
}*/
}
```

8.0 - O CÓDIGO DO PROJETO (Versão Completa)

A versão completa do jogo tem em sua base a versão um, com o aumento de botões e a inclusão do bluetooth. Mantendo sempre a lógica de programação empregada no modelo menor. Por essas e outras questões, como questões de tamanho excessivo de linhas desta versão, a versão completa esta disponível, juntamente com todos os documentos importantes do projeto, no GitHub que se encontra no item 9.0 – Como acessar o projeto completo.

9.0 - COMO ACESSAR O PROJETO COMPLETO?

Criamos um projeto no GitHub e lá se tem acesso a todos os documentos, desde o código até as imagens do projeto completo. Acesse clicando na imagem a seguir.

10.0 - CONCLUSÃO

O Hikas atendeu a seus objetivos propostos, ao passo que possibilita o ensino musical básico de uma forma divertida e mais acessível. Com seu design genérico, Hikas aceita qualquer vetor musical com apenas alguns ajustes, o que possibilita expansões futuras em questão de aumento de músicas. Além de que, o acréscimo de novos modos pode ser feito de maneira igualmente facilitada, o que mostra assim o quanto o jogo é algo que pode evoluir junto com seu jogador.

11.0 - REFERÊNCIAS

SUMMERFUEL ROBOTICS. Arduino Hero 1.0. Disponível em: https://sites.google.com/site/summerfuelrobots/arduino-sensor-tutorials/arduino-hero.

BRINCANDO COM IDEIAS. Explorando o Arduino - Aula 6 - Expansão de Portas via I2C. Disponível em: https://www.youtube.com/watch?v=r-3p-weAtOY.

BRINCANDO COM IDEIAS. Sleep Mode | Economia de energia no Arduino. Disponível em: https://www.youtube.com/watch?v=j9EPzit1nDM.

BRINCANDO COM IDEIAS. Genius com Arduino - Brinquedos Inteligentes. Disponível em: https://www.tinkercad.com/things/bMNY9wFDzg6-genius-com-arduino-brinquedos-inteligentes.

ARDUINO. LowPower.deepSleep(). Disponível em: https://www.arduino.cc/en/Reference/LowPowerDeepSleep.

SUPER MARIO THEME. Disponível em: https://www.tinkercad.com/things/81sIEGzvYqU.

LEGEND OF ZELDA THEME. Disponível em: https://www.tinkercad.com/things/8rqWr3faVRx.

A IMPORTÂNCIA DA EDUCAÇÃO MUSICAL. Disponível em: https://leiturinha.com.br/blog/10-razoes-para-investir-na-educacao-musical-do-seu-filho/.

MUNDO PROJETADO. Módulo Bluetooth – Criando aplicativo. Disponível em: http://mundoprojetado.com.br/modulo-bluetooth-criando-aplicativo-parte-2/.