

PROJETO ARQUITETURA DE COMPUTADORES

TECLADO MATRICIAL COM SOM

Disciplina: Arquitetura de Computadores

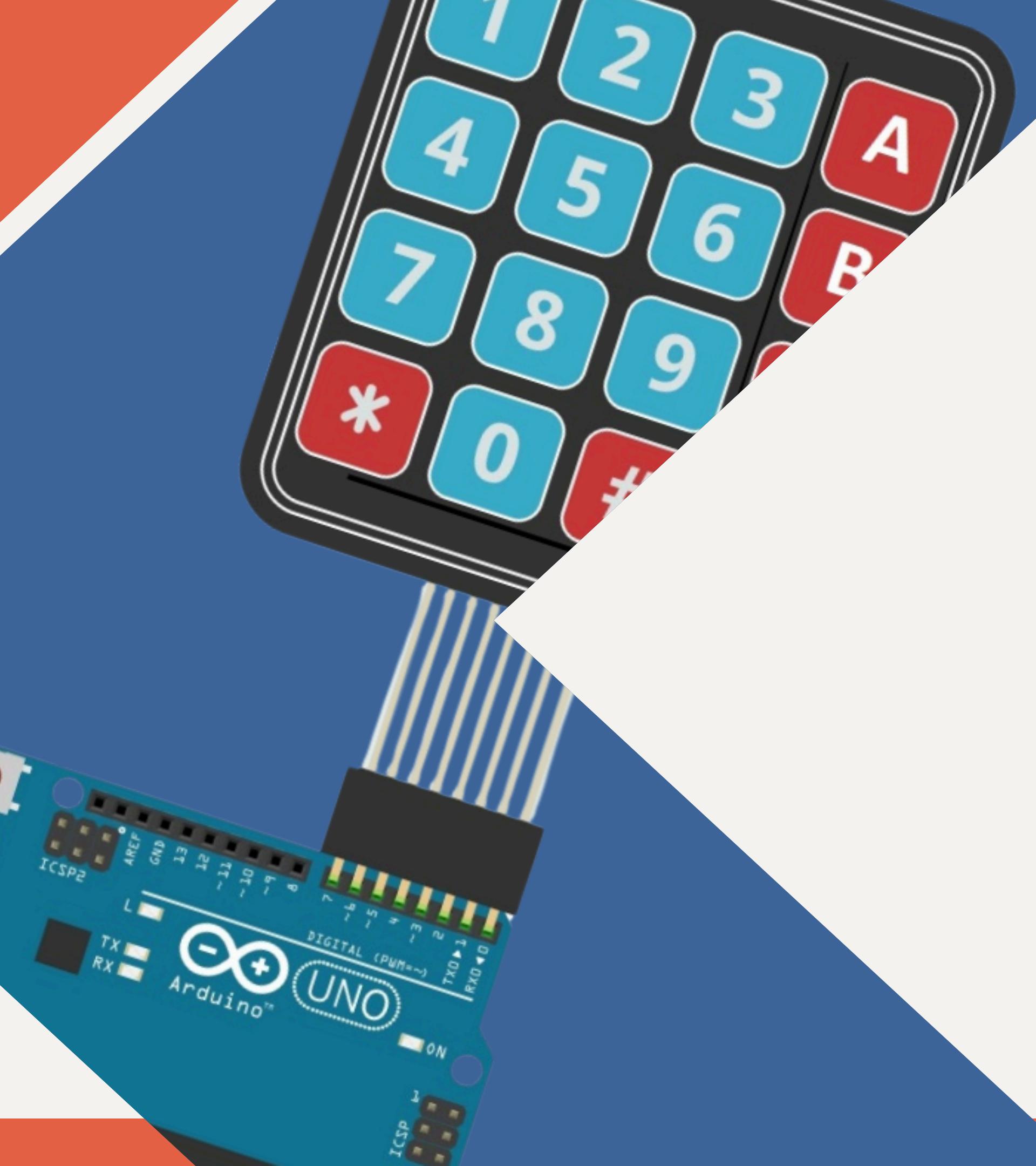
Professor: Rafael Rosa

Alunos: Luciano Smanioto Neto

Rodrigo Corrêa da Silva

Willian Belmonte

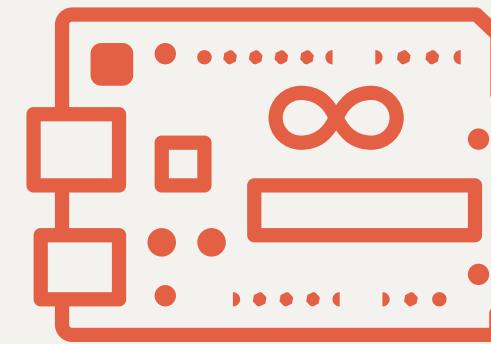
VISÃO



Como os três alunos gostam muito de música e tocam ou já tocaram algum instrumento musical, escolhemos realizar um projeto que envolvesse som, transformando o teclado matricial utilizado no exercício da calculadora em um “teclado musical”.

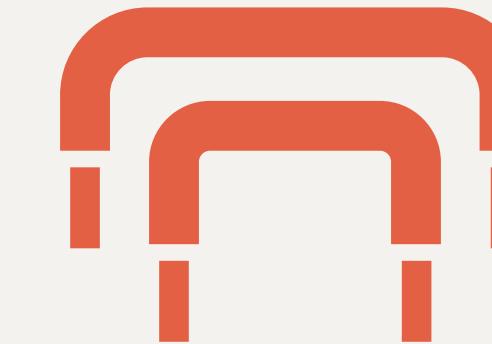
Para incrementar um pouco o projeto, decidimos colocar uma linha de LEDs acendendo cada vez que uma tecla for pressionada no teclado matricial.

HARDWARE ARDUINO



- MATERIAL

1 Arduino UNO
1 Teclado Matricial
1 Alto-Falante (6 ohms)
9 LEDs
Jumpers



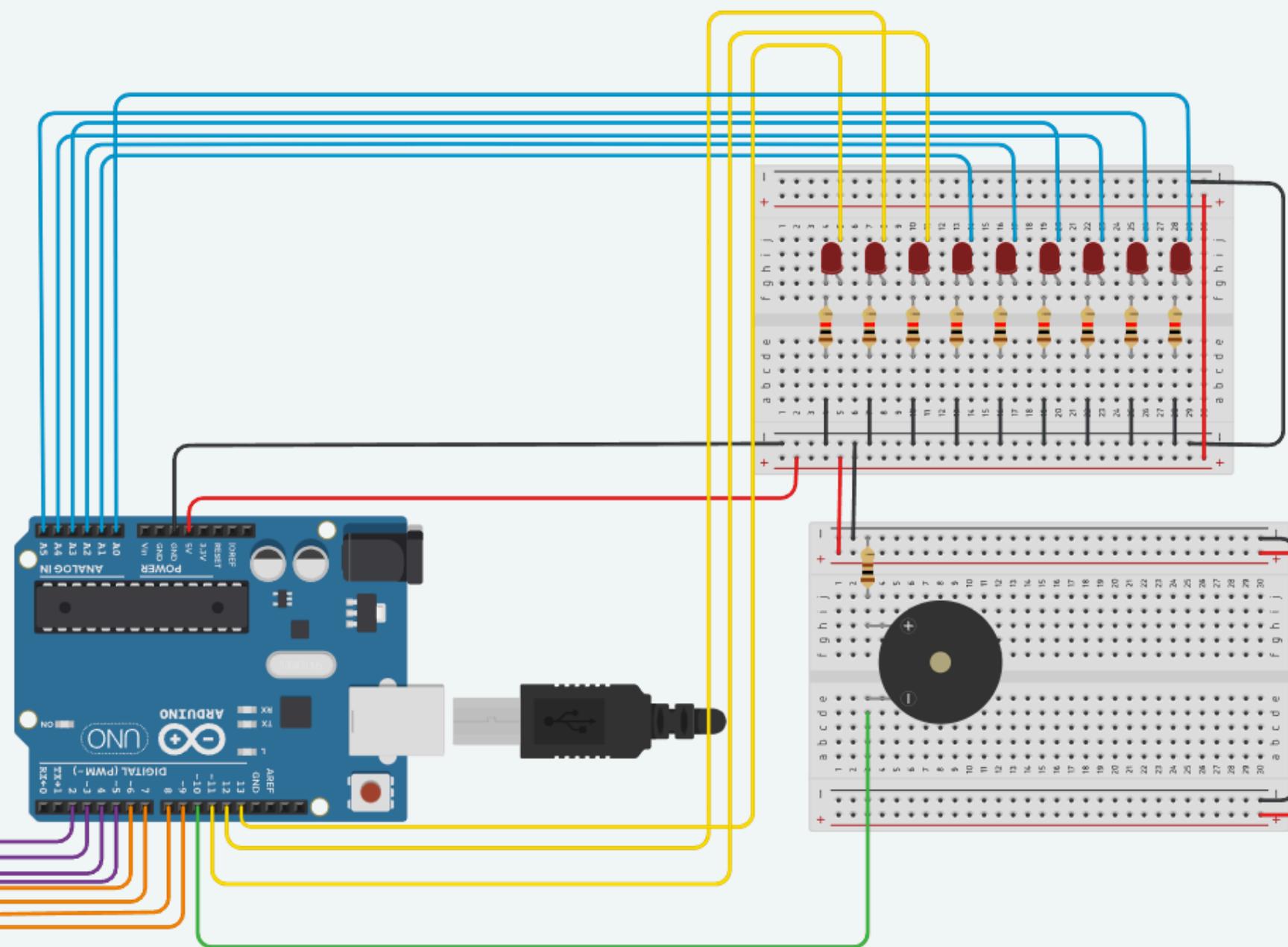
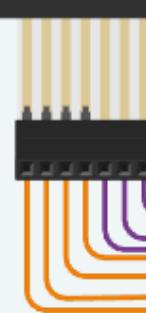
- LIGAÇÕES

> Teclado Matricial
Pinos 2-9 do Arduino

> Alto-Falante
Pino 10 (output) e GND

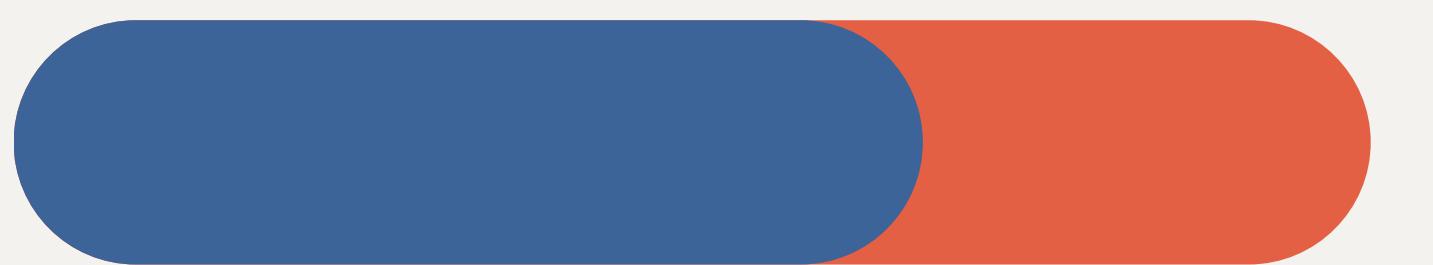
> LEDs
Pinos 13, 12, 11, A0, A1, A2, A3, A4, A5

PROTÓTIPO NO TINKERCAD

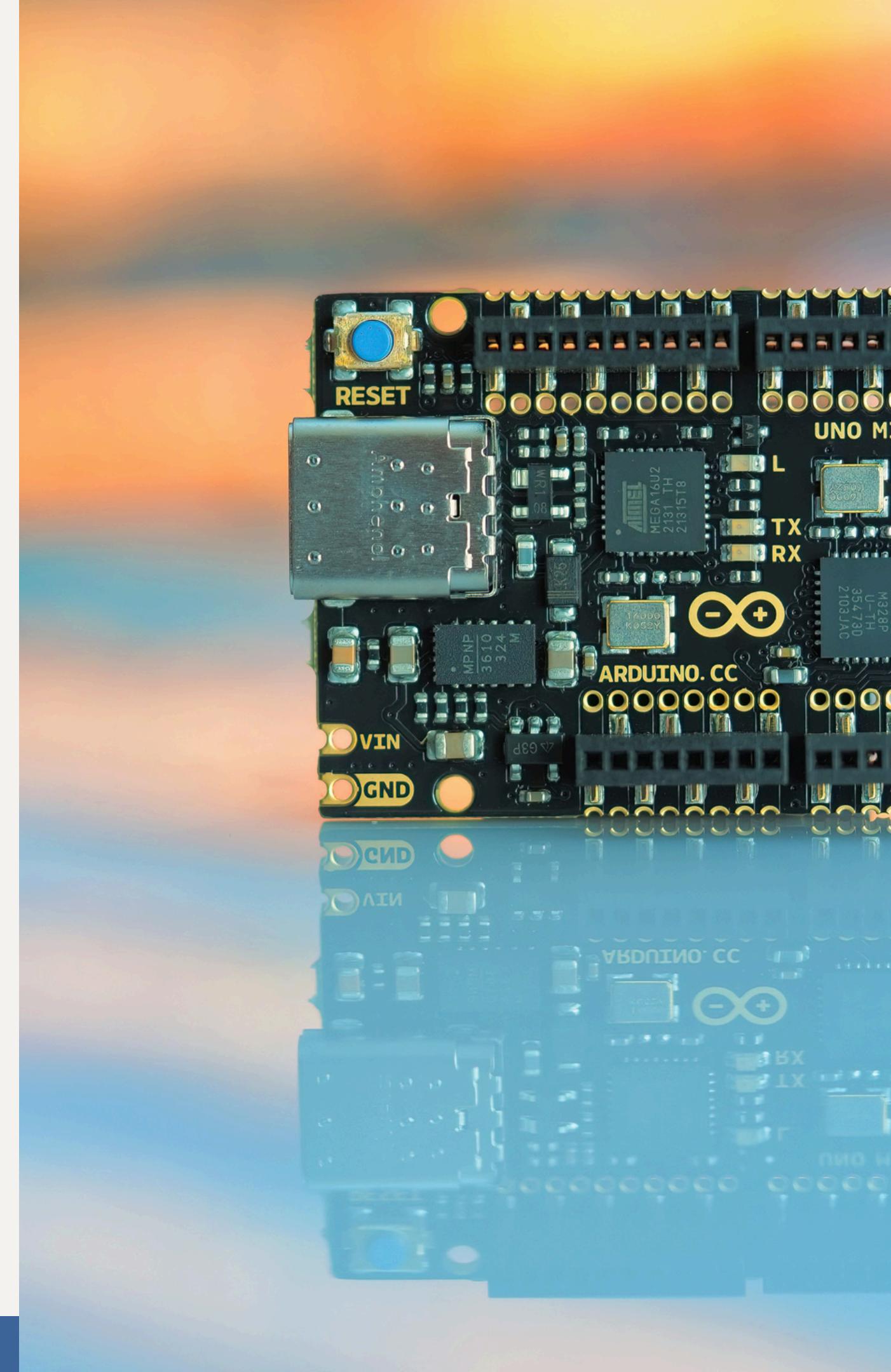


<https://www.tinkercad.com/things/3KvYlhKHgXB-leds-dancam-ao-piano/editel?returnTo=%2Fdashboard%2Fdesigns%2Fcircuits>

CÓDIGO C⁺⁺



SEÇÃO DE DECLARAÇÕES



DEFINIÇÃO DO TECLADO MATRICIAL

```
const byte ROWS = 4;
```

- Esta linha de comando define o número de linhas

```
const byte COLS = 4;
```

- Esta linha de comando define o número de colunas



MAPEAMENTO DAS TECLAS

```
charkeys[ROWS][COLS] = {  
    {'1', '2', '3', 'A'},  
    {'4', '5', '6', 'B'},  
    {'7', '8', '9', 'C'},  
    {'*', '0', '#', 'D'}  
};
```

- Nesta seção, com o comando "charkeys" são definidas as teclas que constam nas colunas (COLS) e linhas (ROWS) no teclado

DEFINIÇÃO DAS LIGAÇÕES DO TECLADO MATRICIAL DECLARADO COM OS PINOS DO ARDUINO

```
byte rowPins[ROWS] = {9, 8, 7, 6};
```

- Aqui, estamos definindo que as linhas serão conectadas aos pinos 6, 7, 8 e 9 do Arduino

```
byte colPins[COLS] = {5, 4, 3, 2};
```

- E aqui, estamos definindo que as colunas serão conectadas aos pinos 2, 3, 4 e 5 do Arduino

INICIALIZAÇÃO DO TECLADO

```
Keypadkeypad =  
Keypad(makeKeymap(keys), rowPins,  
colPins, ROWS, COLS);
```

- **Depois de declarar o teclado e informar as conexões no Arduino, é necessário inicializar as teclas, o que é feito com esta linha de comando, mapeando os pinos declarados para as linhas e para as colunas do teclado matricial (keypad)**

DECLARAÇÃO DO PINO DE SAÍDA DO ARDUINO

```
const int buzzerPin = 10;
```

- Outra linha importante do código, define o pino 10 do Arduino como o pino de saída dos dados que serão enviados para o alto-falante

DECLARANDO OS SONS

```
const int noteFrequencies[16] = {  
    261, 294, 329, 349,  
    392, 440, 493, 523,  
    587, 659, 698, 784,  
    880, 987, 1046, 1175  
};
```

- Neste trecho do código, definimos os tons sonoros correspondentes a cada tecla. Assim, quando cada tecla for acionada, um som será reproduzido no alto-falante em uma nota específica. Por exemplo: a tecla "1" vai reproduzir um C4 (dó na quarta); a tecla "A" vai reproduzir um F4 (Fá na quarta); e assim por diante.
- As notas são definidas a partir de uma frequência. A frequência de um som é a expressão da velocidade na qual as ondas de som se propagam no ar. Ela ocorre por meio de vibrações, que podem ser mais espaçadas ou mais intensas, definindo, assim, agudos e graves.

DECLARANDO OS LEDS

```
const int ledPins[] = {13, 12, 11, A0, A1, A2, A3, A4, A5};
```

- Aqui, definimos os pinos do Arduino aos quais os LEDs estão conectados. Mesmo os pinos analógicos (A0, A1, A2, A3, A4 e A5) do Arduino podem ser utilizados como saída de dados, permitindo que mais LEDs sejam conectados.

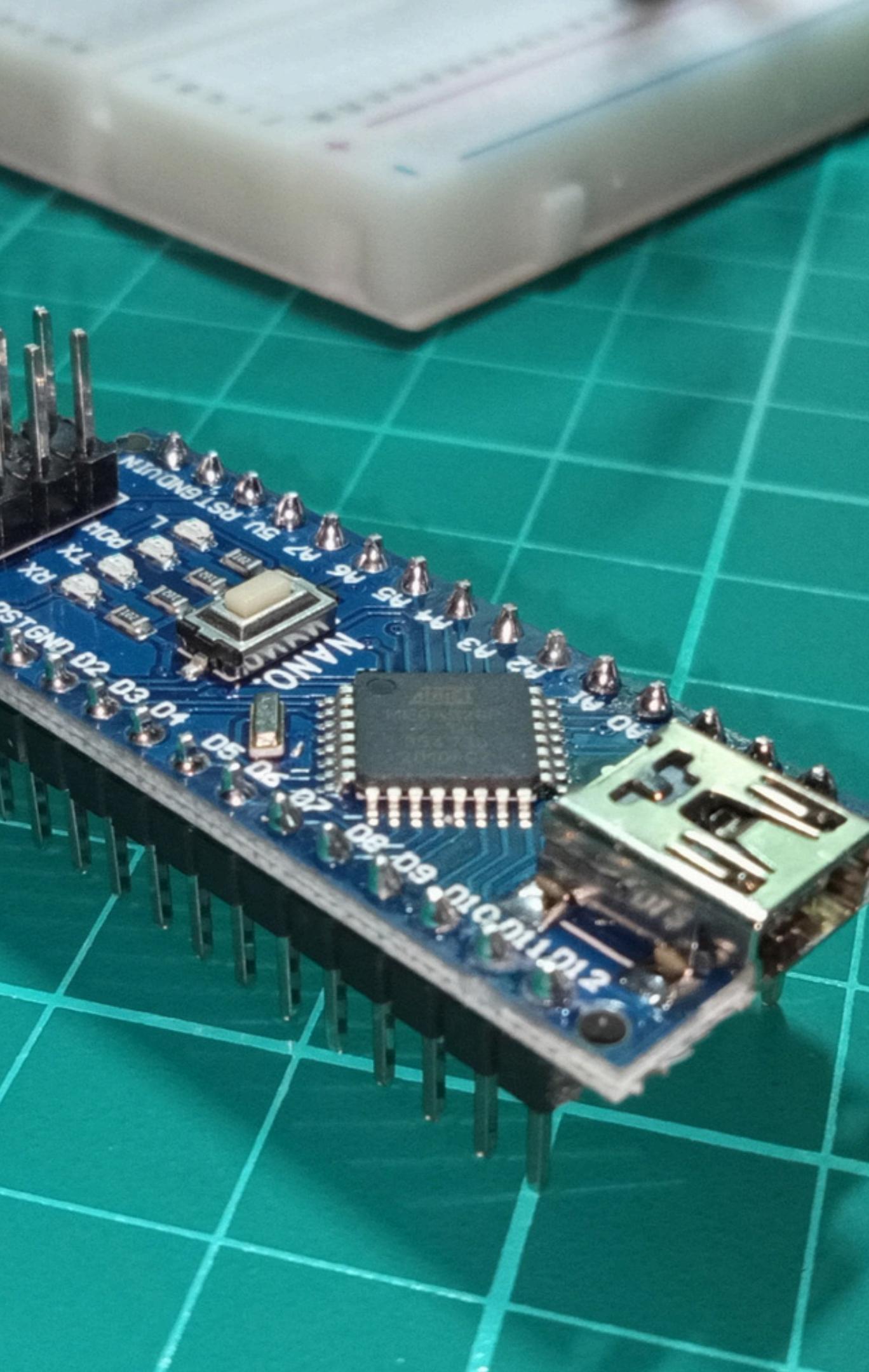
```
const int numLeds = sizeof(ledPins) / sizeof(ledPins[0]);
```

- Esta linha de comando é utilizada para calcular o número de elementos no vetor “ledPins”, armazenado o valor na constante “numLeds”. Isso é útil para executar a repetição do acionamento de todos os pinos de forma dinâmica, sem precisar contar manualmente os elementos.

```
const int ledInterval = 5;
```

- Aqui, estabelecemos o intervalo de tempo (5 milissegundos) que cada LED permanecerá ligado a cada acionamento de tecla

SEÇÃO DE COMANDOS



INICIALIZANDO AS VARIÁVEIS

```
void setup() {  
Serial.begin(9600);  
pinMode(buzzerPin, OUTPUT);  
}
```

- O void setup() é executado quando o programa começa e vai inicializar as variáveis declaradas, configurar os pinos de entrada e saída, e executar outras configurações iniciais necessárias para o funcionamento do programa.

- **Serial.begin(9600)**. - Esta linha é utilizada no Arduino para iniciar a comunicação serial. A função Serial.begin(9600) configura o hardware serial do Arduino - neste caso específico, iniciando que a comunicação serial com os dispositivos conectados (como o teclado matricial) será realizada em uma taxa de transmissão de dados de 9600 bits por segundo.

- pinMode(buzzerPin, OUTPUT). - Aqui, nós estamos inicializando a variável declarada "buzzerPin" e informando que ela é utilizada como saída de dados (OUTPUT).

```
for(int i = 0; i < numLeds; i++){
```

- Aqui, estamos criando o laço que vai contar os LEDs, iniciando em zero e incrementando em 1 até que o valor alcance a quantidade de LEDs identificada anteriormente. Aí, ele vai passar a repetir o laço. Isso serve para controlar a ação de ligar e desligar os LEDs. Ou seja, estamos inicializando a variável “i”, que funciona como um contador para percorrer todos os LEDs na hora de executar o comando.

```
pinMode(ledPins[i], OUTPUT);
```

- Esta linha define os pinos do Arduino aos quais os LEDs estão conectados como “saída” de dados (OUTPUT);

```
digitalWrite(ledPins[i], LOW);  
}
```

- Nesta linha, ao utilizarmos a expressão “LOW”, garantimos que todos os LEDs estejam apagados no início da ação.

DEFININDO OS PROCESSAMENTOS QUE SERÃO EXECUTADOS NO LOOP PRINCIPAL

```
void loop() {  
    char key = keypad.getKey();
```

- **void loop()** - Depois de inicializar as variáveis com o **setup()**, o programa está pronto para realizar o processamento do loop principal, que começa a ser executado a partir desta linha de comando.

- char key = keypad.getKey(); - Aqui, vamos "ler" o caractere digitado no tecla do matricial. A função `keypad.getKey()` verifica as teclas do teclado e retorna o caractere correspondente à tecla pressionada. E a função "char key =" inicializa uma variável chamada "key" do tipo "char", que armazenará o caractere (char) digitado no teclado. Combinadas as funções, a linha de código está obtendo o próximo caractere digitado no teclado e armazenando na variável "key".

```
if (key) {  
Serial.println(key);
```

- **if(key){** e **Serial.println(key);** - Estas funções combinadas vão verificar se alguma tecla foi pressionada e enviar (por meio da comunicação serial) o comando para o hardware executar.

```
for (int i = 0; i < ROWS; i++) {  
    for (int j = 0; j < COLS; j++) {  
        if (keys[i][j] == key) {  
            int noteIndex = i * COLS + j;
```

- Com esta sequência de comandos, localizamos a posição da tecla digitada dentro da matriz de linhas e colunas declarada para o teclado.

```
tone(buzzerPin, noteFrequencies[noteIndex], 200);
```

- Este é o comando que vai mandar o alto-falante reproduzir a frequência selecionada, com uma duração de 200 milissegundos.

```
if (key) {  
  
    for (int i = 0; i < numLeds; i++) {  
        digitalWrite(ledPins[i], HIGH);
```

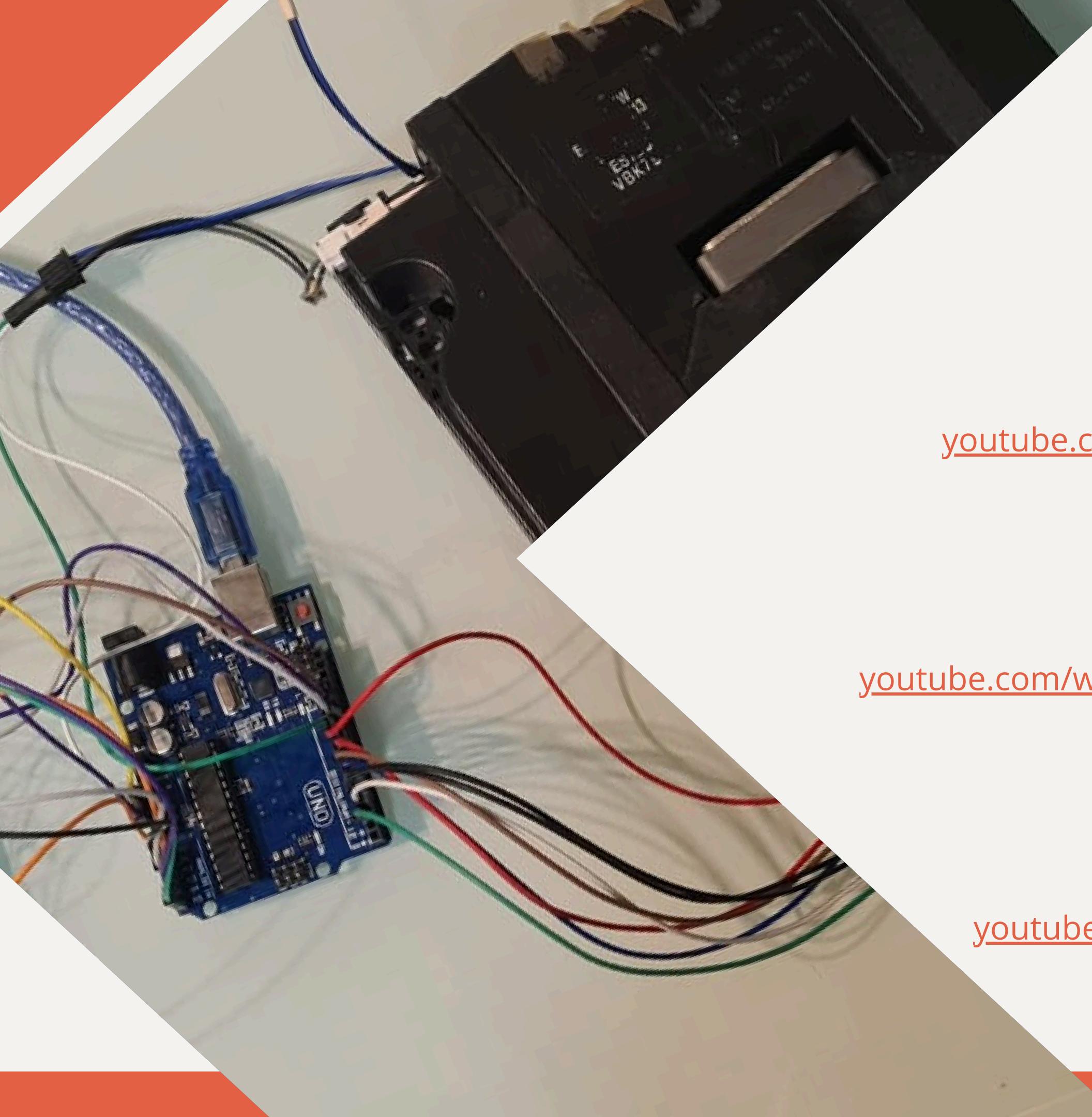
- Aqui, estas linhas de comando estão orientando o programa a percorrer o vetor e acionar os LEDs (mudando o parâmetro de “LOW” para “HIGH”) assim que uma tecla for acionada.

```
delay(ledInterval);

for (int i = 0; i < numLeds; i++) {
    digitalWrite(ledPins[i], LOW);
}
}
```

- Com estas linhas, é executado o intervalo de tempo definido para o acionamento do LED; e então é percorrido novamente o vetor para o retorno ao estado “apagado” (LOW) de cada um dos LEDs, que só voltarão a estar ligados (HIGH) quando uma nova tecla for pressionada

VÍDEOS



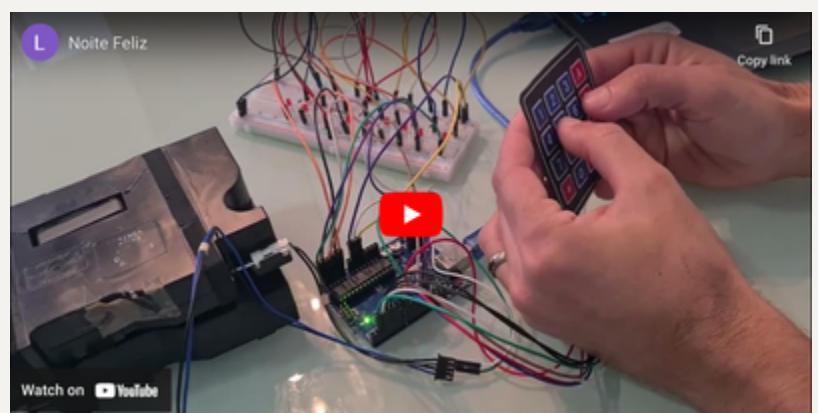
[youtube.com/watch?v=bNn373k3msY](https://www.youtube.com/watch?v=bNn373k3msY)



[youtube.com/watch?v=J09-F97N53k&t=26s](https://www.youtube.com/watch?v=J09-F97N53k&t=26s)



[youtube.com/watch?v=r31g8oAiLVc](https://www.youtube.com/watch?v=r31g8oAiLVc)

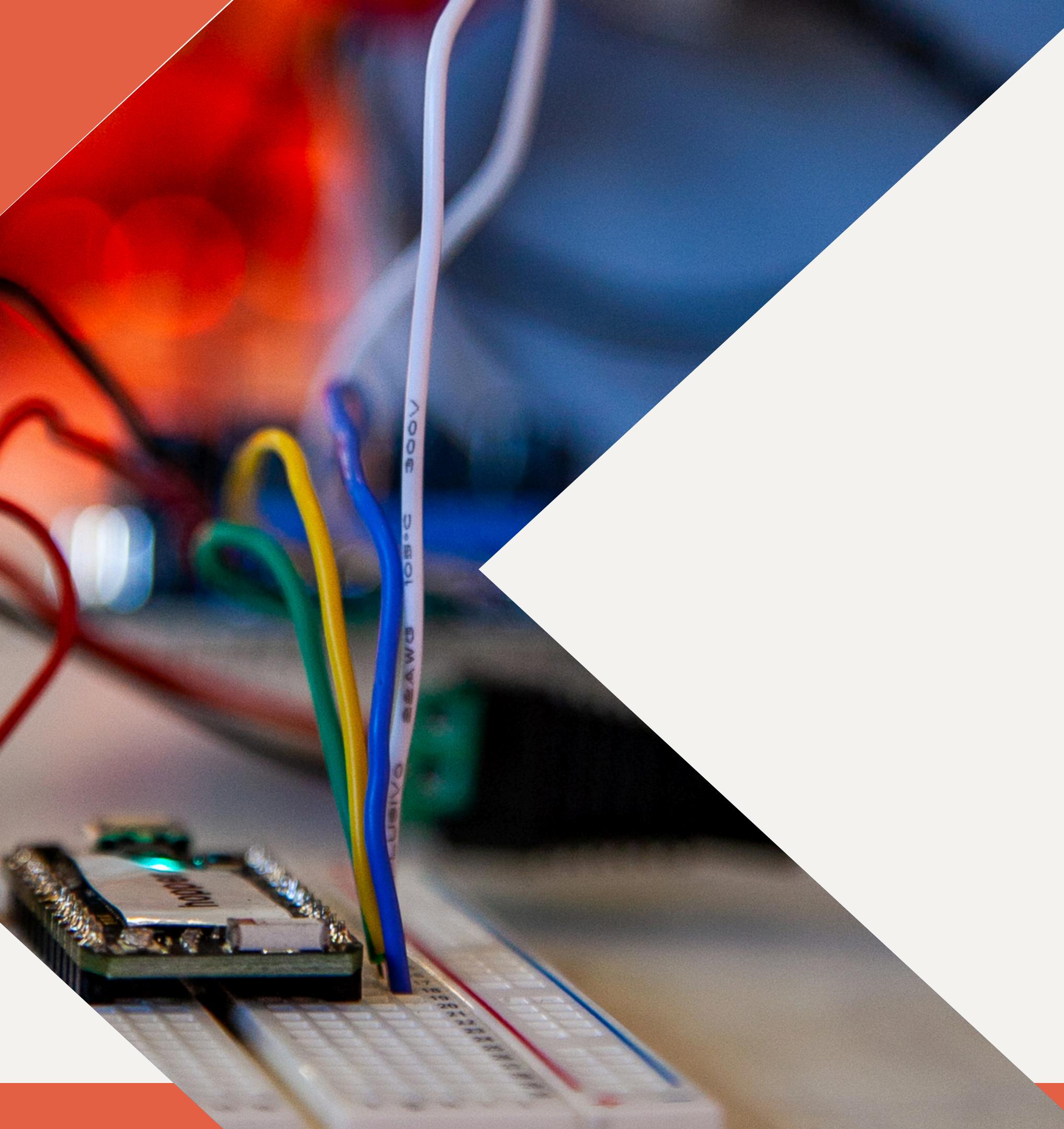


CONCLUSÃO



A disciplina de Arquitetura de Computadores utilizou a plataforma do Arduino como uma forma de trabalhar a noção de funcionamento de circuitos eletrônicos e da comunicação entre software e hardware, importantíssima para o entendimento sobre como funcionam os equipamentos eletrônicos, desde a correlação entre componentes até os detalhes da montagem e a utilização de comandos de execução de tarefas por meio de uma linguagem que a máquina compreenda. Portanto, o projeto foi importante por abranger diferentes núcleos de desenvolvimento, envolvendo hardware e software, dois aspectos fundamentais para a rotina de trabalho dentro da profissão de Análise e Desenvolvimento de Sistemas.

APÊNDICES



CÓDIGO COMPLETO

```
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins,
    ROWS, COLS);

const int buzzerPin = 10;
const int noteFrequencies[16] = {
    261, 294, 329, 349,
    392, 440, 493, 523,
    587, 659, 698, 784,
    880, 987, 1046, 1175
};

const int ledPins[] = {13, 12, 11, A0, A1, A2, A3, A4, A5};
const int numLeds = sizeof(ledPins) / sizeof(ledPins[0]);
const int ledInterval = 5;

void setup() {
    Serial.begin(9600);
    pinMode(buzzerPin, OUTPUT);
    for (int i = 0; i < numLeds; i++) {
        pinMode(ledPins[i], OUTPUT);
        digitalWrite(ledPins[i], LOW);
    }
}

void loop() {
    char key = keypad.getKey();
    if (key) {
        Serial.println(key);
        for (int i = 0; i < ROWS; i++) {
            for (int j = 0; j < COLS; j++) {
                if (keys[i][j] == key) {
                    int noteIndex = i * COLS + j;
                    tone(buzzerPin, noteFrequencies[noteIndex], 200);
                }
            }
        }
        if (key) {
            for (int i = 0; i < numLeds; i++) {
                digitalWrite(ledPins[i], HIGH);
                delay(ledInterval);
            }
            delay(ledInterval);
            for (int i = 0; i < numLeds; i++) {
                digitalWrite(ledPins[i], LOW);
            }
        }
    }
}
```

#include <Keypad.h>

const byte ROWS = 4;

const byte COLS = 4;

char keys[ROWS][COLS] = {

{'1', '2', '3', 'A'},

{'4', '5', '6', 'B'},

{'7', '8', '9', 'C'},

{'*', '0', '#', 'D'}

};

byte rowPins[ROWS] = {9, 8, 7, 6},

byte colPins[COLS] = {5, 4, 3, 2};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins,

ROWS, COLS);

const int buzzerPin = 10;

const int noteFrequencies[16] = {

261, 294, 329, 349,

392, 440, 493, 523,

587, 659, 698, 784,

880, 987, 1046, 1175

};

ARDUINO TM

MATRIPARTITURAS

- Sobre o Tempo
(Nenhum de Nós)

6
5-6-7
6-7-8
7-8-C
8-7-6
6-B-6-5-4

- Sweet Child O'Mine
(Guns'n Roses)

4-6-5-4-8-7-8-7-5
5-6-5-8-7-8-7-5
5-6-5-8-7-8-7
8-7-8-9-C-9-8-9

5-5-5-5-6-5-5-5-5-6-4-
4-4-4-5-4-4-4-4-5-3-3-
3-3-4-3-3-3-3-4-2-2-2-
2-3-2-2-2-2-3



- **Noite Feliz**

(Canção Natalina)

4-5-4-3

4-5-4-3

7-7-6-B-B-4

5-5-B-6-5-4-5-4-3

5-5-B-6-5-4-5-4-3

7-7-9-7-6-B-8

B-4-3-4-A-2-1

- **Parabéns a Você**

(Canção Popular)

4-4-5-4-B-6

4-4-5-4-7-B-B

4-4-8-B-B-6-5

8-8-7-6-7-B-B