Projeto Integrador - Desenvolvimento Estruturado de Sistemas

Monitoramento de Ruído com ESP8266

Guilherme Henrique Veloso Santos
Levi Freitas da Silva
Pedro Henrique Melo da Silva
Rodrigo Eduardo Nogueira

São Paulo
2023

Sumário

1. Objetivo.

O objetivo deste projeto é criar um sistema de monitoramento de níveis de ruído em um ambiente específico usando um microcontrolador ESP8266 e um sensor de som. O sistema irá analisar continuamente os níveis de ruído e, com base nessa análise, exibirá informações visuais por meio de um semáforo LED de três cores. O objetivo principal é fornecer aos usuários uma representação visual intuitiva e em tempo real do ambiente sonoro, permitindo-lhes tomar decisões informadas sobre as atividades apropriadas para o local.

2. Finalidade.

A finalidade deste projeto é fornecer um meio eficiente e preciso de monitorar os níveis de ruído em um ambiente específico, permitindo a detecção de variações significativas que possam afetar a qualidade do ambiente e a saúde das pessoas, como salas de aula, escritórios, espaços de trabalho compartilhados ou qualquer local onde o ruído seja uma consideração importante. Além disso através de uma representação visual clara dos níveis de ruído em tempo real por meio de um semáforo LED de três cores (verde, amarelo e vermelho). Isso permite que os ocupantes do ambiente possam reconhecer instantaneamente os níveis de ruído e ajustem seu comportamento de acordo, promovendo um ambiente mais tranquilo e produtivo quando necessário. Um sistema simples e de fácil compreensão que permite a identificação do que é pode ser prejudicial à saude, promovendo a conscientização sobre o controle de ruidos em determinados ambientes.

3. Componentes

Para desenvolvimento do projeto precisamos de alguns componentes essenciais:

O **ESP8266** é o microcontrolador central do projeto e será usado para coletar dados do sensor de som, processá-los e controlar o semáforo LED.

Sensor de Som: (ou Módulo de Microfone): Para detectar os níveis de ruído no ambiente. Usaremos um sensor de som com um potenciômetro que permite regular a sensibilidade do sinal analógico proporcional ao som ambiente.

LEDs: Para criar um semáforo com cores diferentes para indicar níveis de ruído.

Resistores: Serão necessários resistores para limitar a corrente que passa pelos LEDs e para realizar as conexões adequadas com o ESP8266.

Placa de Circuito Perfurado ou Protoboard: Por causa da quantidade de LEDs será necessário um circuito perfurado ou protoboard para montar os componentes e fazer as conexões, permitindo que usemos um terra para otimizar a utilização dos pinos do ESP8266

Fonte de Alimentação: Como fonte de energia utilizaremos um cabo USB que conectará à um computador para alimentar o circuito.

Fios e Cabos: Fios e cabos elétricos serão necessários para fazer as conexões entre os componentes.

Computador com IDE Arduino: Um ambiente de desenvolvimento para que seja possível o funcionamento do sistema de acordo com o que foi proposto.

Custo dos componentes

Item	Preço (unitário)
ESP8266	R$26,00
Sensor de som analógico KY-037	R$6,56
LEDs	R$3,50
Resistores	R$1,00
Protoboard (400 pontos)	R$15,00
Fonte de Alimentação	R$15,00
Fios e Cabos	R$5,00
Total Aproximado	**R$72,06**

Tabela 1 - Valor unitário dos componentes

4. Conexão dos componentes.

A conexão dos componentes deve ser feita começando pela placa ESP8266, que é a base do circuito e é responsável por controlar o sensor e os LEDs. Para que a placa funcione, ela precisa de energia. A tensão VCC (circuito de alimentação) da placa será conectada à fonte de alimentação de 5 V. O pino GND (terra) da placa será conectado ao GND da fonte de alimentação para que a placa seja aterrada. O sensor de som deve ser conectado com o GND do microcontrolador como mostra a *Figura 1,* ou na parte da protoboard destinada ao terra e seu pino analógico de saída deve ser conectado à um pino digital do ESP8266. Já o resistores serão utilizados para limitar a corrente dos LEDs, para que os mesmos funcionem corretamente, além dos pinos GNDs dos LEDs que devem ser aterrados. Na ilustração abaixo é possível ver a conexão dos componentes que deve ser semelhante mudando apenas o microcontrolador que estará conectado por um cabo usb à um computador que o fornecerá energia, adicionando uma protoboard e o display é apenas um demonstrativo, mas utilizaremos de um Streaming de dados para guardar essas informações.

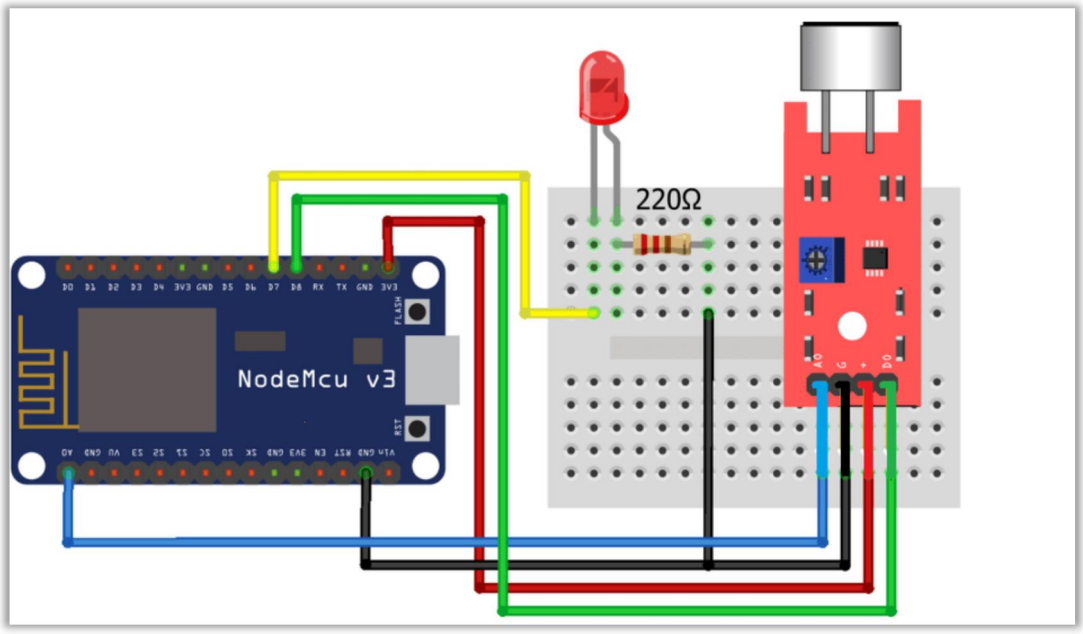

Figura 1 - Circuito Arduino conectado. Fonte: iotprojectsideas.com

5. Código do Projeto

```
const int janelaAmostragem = 150;
// Largura da janela de amostragem em mS (50 mS = 20Hz
unsigned int amostra;

#define PINO_SENSOR 17 //A0
#define PINO_CALMO 14 //D5
#define PINO_MODERADO 2 //D4
#define PINO_ALTO 13 //D7

void setup()
{
    pinMode(PINO_SENSOR, INPUT); // Configura o pino do sensor como entrada
    pinMode(PINO_CALMO, OUTPUT);
    pinMode(PINO_MODERADO, OUTPUT);
    pinMode(PINO_ALTO, OUTPUT);

    digitalWrite(PINO_CALMO, LOW);
    digitalWrite(PINO_MODERADO, LOW);
    digitalWrite(PINO_ALTO, LOW);

    Serial.begin(115200);

}
void loop()
{

    unsigned long inicioMillis = millis(); // Início da janela de amostragem
    float picoAPico = 0; // Nível pico a pico

    unsigned int sinalMaximo = 0; // Valor mínimo
    unsigned int sinalMinimo = 1024; // Valor máximo

    // Coleta dados por 50 mS
    while (millis() - inicioMillis < janelaAmostragem)
    {
        amostra = analogRead(PINO_SENSOR); // Obtém leitura do microfone
        if (amostra < 1024) // Descarta leituras espúrias
        {
            if (amostra > sinalMaximo)
            {
                sinalMaximo = amostra; // Salva apenas os níveis máximos
            }
            else if (amostra < sinalMinimo)
            {
                sinalMinimo = amostra; // Salva apenas os níveis mínimos
            }
        }
    }

    picoAPico = sinalMaximo - sinalMinimo; // Máximo - mínimo = amplitude pico a pico
    int decibeis = map(picoAPico, 20, 900, 62, 145) * 1.25;
    // Calibrar para decibéis conforme app celular
if (decibeis <= 65)
     {

         digitalWrite(PINO_CALMO, HIGH);
         digitalWrite(PINO_MODERADO, LOW);
         digitalWrite(PINO_ALTO, LOW);
         Serial.println(decibeis);
     }
     else if (decibeis > 65 && decibeis < 105)
     {

          lightRGB(PINO_ALTO, PINO_CALMO, PINO_MODERADO);
         Serial.println(decibeis);
     }
     else if (decibeis >= 105)
     {

         digitalWrite(PINO_CALMO, LOW);
         digitalWrite(PINO_MODERADO, LOW);
         digitalWrite(PINO_ALTO, HIGH);
         Serial.println(decibeis);
     }

     delay(150);

 }
```

Figura 2 - Código de execução do projeto

6. Streaming de Dados.

Para o streaming de dados foi feito através do Excel. Selecionando o recurso disponível do serviço da Microsoft Data Streamer. Com o dispositivo corretamente conectado foi possível coletar os dados e com isso gerar uma planilha onde de acordo com os niveís de decibéis e criando uma formula para definir as cores de cada célula para facilitar a leitura dos dados como indicadores, de acordo com o que foi proposto e com as cores do led.

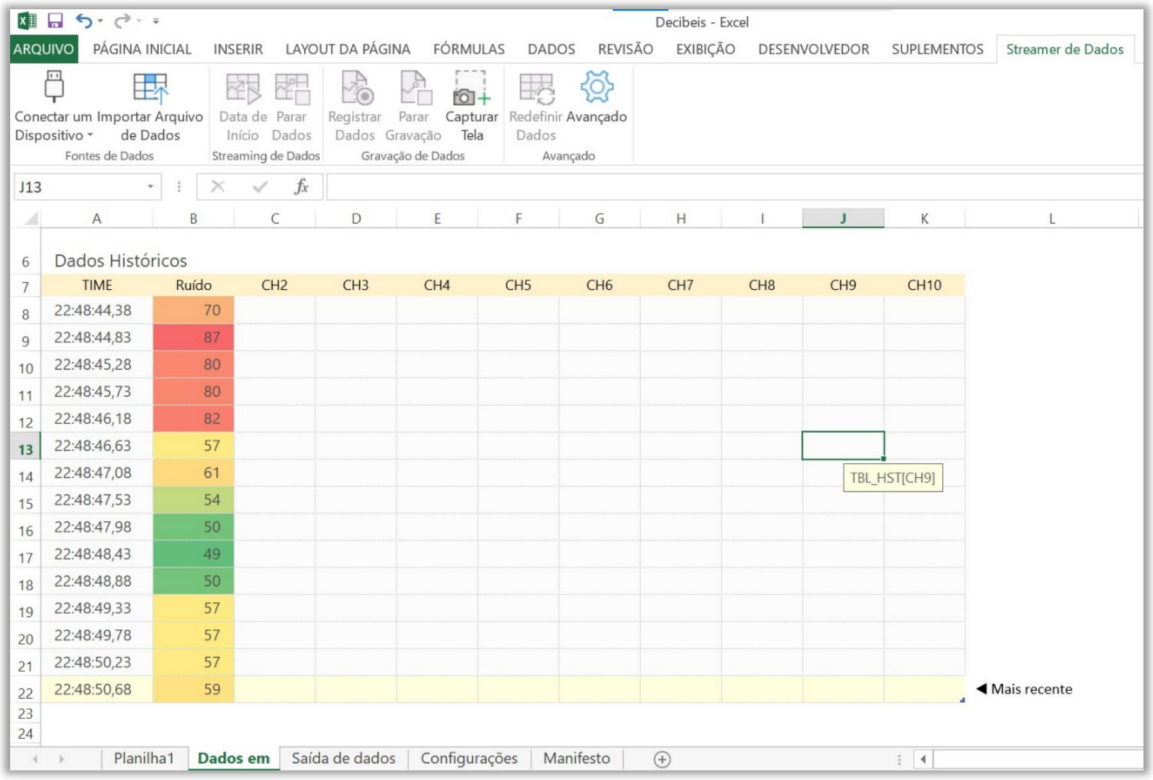

Figura 3 - Dados coletados através do serviço de Data Streamer Excel.

7. Banco de Dados

A partir do arquivo CSV gerado, salvo pelo Excel e criando um banco de dados SQL com os mesmos campos, fizemos um algoritmo usando JavaServer Pages que fosse possível ler este arquivo e extrair os dados enviar para o SQL e exibi-los em uma página HTML através do JSP. O JavaServer Pages como mostra a *Figura 4*, realiza a leitura do arquivo contendo os registros de tempo e níveis de decibéis. Utilizando JDBC, se estabelece uma conexão com um banco de dados MySQL, percorre as linhas do arquivo e lê apenas o que tiver números(expressão regex utilizada para ignorar cabeçalho), formata as horas, e insere os dados no banco. Durante esse processo, é feito o tratamento de

exceções para garantir a integridade dos dados e caso ocorra algum erro, seja exibido o mesmo em tela.

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.sql.Connection" %>
<%@ page import="java.sql.PreparedStatement" %>
<%@ page import="java.sql.DriverManager" %>
<%@ page import="java.io.BufferedReader" %>
<%@ page import="java.io.FileReader" %>
<%@ page import="java.io.IOException" %>
<%@ page import="java.sql.SQLException" %>
<%@ page import="java.text.ParseException" %>
<%@ page import="java.text.SimpleDateFormat" %>
<%@ page import="java.sql.Time" %>

<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Cadastro via Excel</title>
</head>
<body>
<%
try (BufferedReader br = new BufferedReader(new FileReader("D:\\Dev\\Projeto PI\\web\\decibeis.csv"))) {
    Connection conecta;
    Class.forName("com.mysql.cj.jdbc.Driver");
    String url = "jdbc:mysql://localhost:3306/bancopi";
    conecta = DriverManager.getConnection(url, "root", "P@$$w0rd");

    String linha = "";
    while ((linha = br.readLine()) != null) {
        if (linha.trim().startsWith("TIME") || linha.trim().isEmpty()) {
            continue;
        }

        String[] dados = linha.split(";");
        if (dados.length > 1 && dados[1] != null && !dados[1].isEmpty()) {
            String horaString = dados[0].replaceAll("[^0-9]", "");

            // Manipulação da String para formatar em horas
            if (horaString.length() >= 6) {
                int decibeis = Integer.parseInt(dados[1]);

                String horaFormatada = horaString.substring(0, 2) + ":" + horaString.substring(2, 4) + ":" + horaString.substring(4, 6);

                try {
                    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
                    java.util.Date date = sdf.parse(horaFormatada);
                    Time hora = new Time(date.getTime());

                    String sql = "INSERT INTO ruido(hora, nivelDecibel) VALUES (?, ?)";
                    try (PreparedStatement stat = conecta.prepareStatement(sql)) {
                        stat.setTime(1, hora);
                        stat.setInt(2, decibeis);
                        stat.executeUpdate();

                        out.print("Hora: " + horaFormatada + ", Decibéis: " + decibeis);
                        out.print("O arquivo foi carregado com sucesso");
                    } catch (SQLException e) {
                        e.printStackTrace();
                        // Lógica para lidar com a exceção do SQL
                    }
                } catch (ParseException e) {
                    e.printStackTrace();
                    // Lógica para lidar com a exceção de análise de data
                }
            } else {
                // Lógica para lidar com o caso em que horaString não tem comprimento suficiente
            }
        } else {
            // Lógica para lidar com o caso em que dados[1] não é válido
        }
    }
    br.close();
} catch (IOException | ClassNotFoundException | SQLException e) {
    e.printStackTrace();
    // Lógica para lidar com exceções de leitura de arquivo e de banco de dados
}
%>
</body>
</html>
```

Figura 4 - Código Java para tratamento e inserção dos dados. Disponivel em https://github.com/rodrigoeducativa/esp8266

Em seguida estão as telas de cadastro dos dados no HTML e o SQL usado para armazenar os dados registrados. Além disso através do CSS modificamos as células das tabelas para que fossem intuitivas e correspondessem as cores de acordo com os valores e niveís padronizados pela OMS.

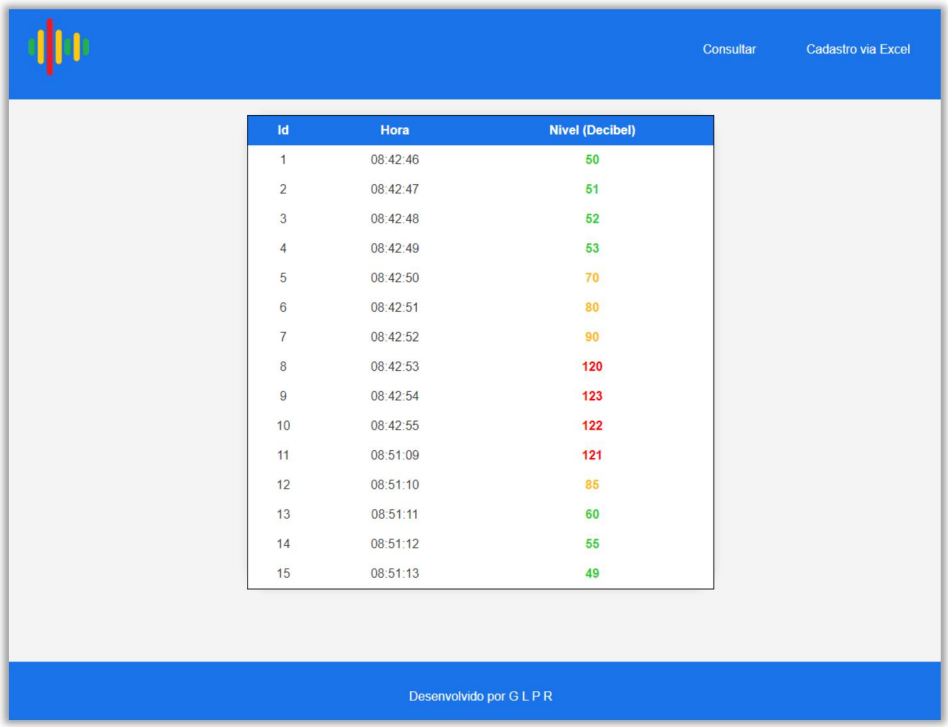

Figura 5 - Página JSP do Projeto

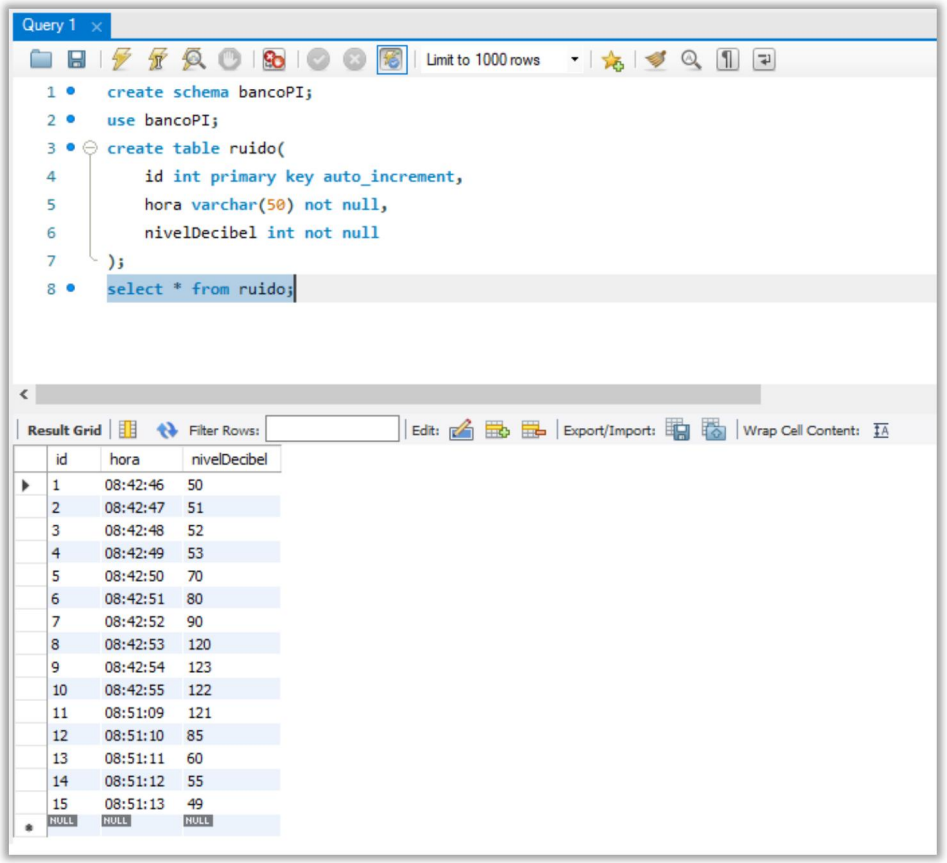

Figura 6 - Banco de Dados SQL gerado para inclusão dos dados

8. Conclusão

Apesar de um projeto simples encontramos algumas dificuldades como por exemplo a escolha dos sensores, devido à sua precisão, alguns podem ser digitais e outros analógicos, porém para o nosso caso era necessário um com sinal analógico, pois dentro de um intervalo, esse valor pode variar de acordo com os ruídos. Além disso no desenvolvimento do código tivemos que implementar uma função map, para poder calibrar uma amplitude pico a pico para uma faixa de valores captados pelo sensor, convertê-los para decibéis em níveis reais e controlar os LEDs de acordo com sua intensidade.

9. Referências

IoT Project Ideas. Arduino-Based Decibel Meter with Sound Sensor. Disponível em: https://iotprojectsideas.com/arduino-based-decibel-meter-with-sound-sensor. Acesso em: 5 out. 2023.

MICROSOFT. (2023). Habilitar o fluxo de dados no Excel. Microsoft Learn, [S.l.], 10 ago. 2023. Disponível em: <https://learn.microsoft.com/pt-br/microsoft-365/education/data-streamer/enable-in-excel>. Acesso em: 6 out. 2023.