Relatório Técnico

Nº Grupo: 07
Nome dos integrantes: Ana Carolina Fiorini Mioki, Camila Yukari Jodai, Caroline Gonçalves Soares, Erick Lopes da Silva, Felipe Marcos Bastos Pena e João Victor Alves Ferreira
Turma: 1ADSB

Tema do projeto: Monitoramento de gás em restaurantes
Sensor: MQ-2 (Gás)

Introdução

Nosso grupo desenvolveu um sistema voltado para o monitoramento de vazamento de gás em restaurantes. No Brasil, é comum que esses estabelecimentos utilizem o gás GLP (Gás Liquefeito de Petróleo), que é armazenado em cilindros de aço, para seu funcionamento. Sua densidade é superior à do ar, portanto, esse gás tende a se acumular nas partes mais baixas dos ambientes, deslocando o oxigênio e, em locais fechados, podendo causar asfixia. No meio do setor de restaurantes que utilizam esse gás, independende do porte, todos possuem esse problema em comum. Para detecção de vazamentos, empregamos o sensor MQ-2 integrado com o Arduino UNO R3, que foram, respectivamente, o sensor e a plataforma escolhidos para a implementação do sistema.

Arquitetura de Montagem do Sensor

Abaixo está uma imagem ilustrando a arquitetura de montagem do projeto em uma mini protoboard. A foto apresenta como o sensor MQ-2 foi conectado ao microcontrolador Arduino UNO R3.

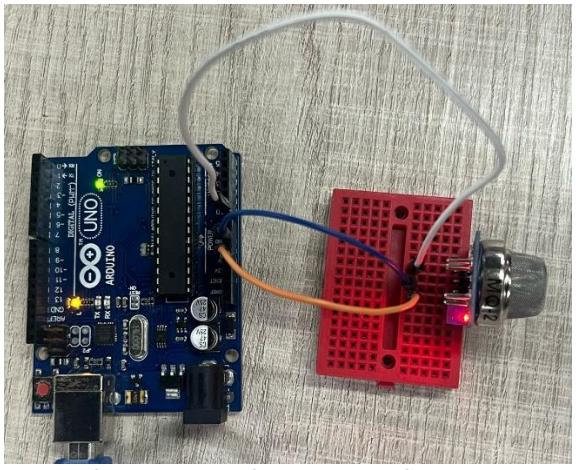


Figura 1 - Foto da montagem do sensor

Arquitetura do Sistema

O sistema abaixo demonstra como o nosso projeto funciona. Primeiramente, há uma integração entre o Arduino UNO R3 junto com o sensor MQ-2, que são responsáveis por identificar a presença do gás GLP no ambiente, logo, caso haja, o Arduino interpreta esses dados coletados e repassa para um sistema, que é a API, que recebe as informações do sensor, realiza o tratamento necessário e as armazena em um banco de dados.

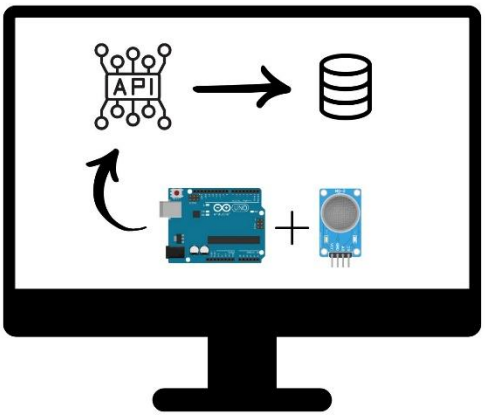


Figura 2 - Ilustração da arquitetura do sistema

Código do Projeto

O código do Arduino IDE lê o valor do sensor de gás no pino A0, calcula a porcentagem de gás presente no ar, e exibe o resultado em PPM (partículas por milhão), imprimindo o valor no monitor serial.

gascodigo | Arduino IDE 2.3.4

File Edit Sketch Tools Help

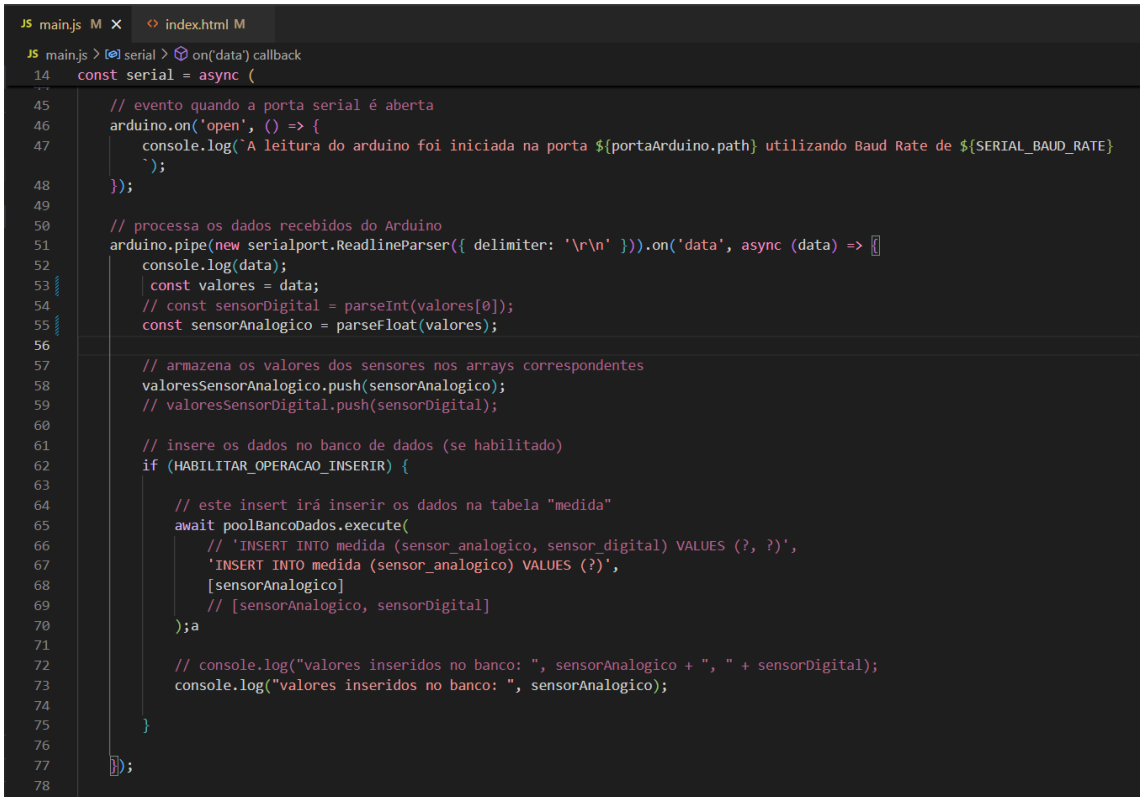
Arduino Uno

gascodigo.ino

```
const int PINO_SENSOR_MQ2 = A0;// cria a variavel e seta o pino A0 do sensor

const int VALOR_MINIMO = 300;//cria uma variavel com a valor mínimo do sensor
const int VALOR_MAXIMO = 1000;//cria uma variavel com a valor máximo do sensor

void setup() {
  Serial.begin(9600);//velocidade de bits de transferência entre arduino e o computador
}

void loop() {
  int valorSensor = analogRead(PINO_SENSOR_MQ2) + 300;//adiciona a uma variável o valor do sensor que está sendo recebida pelo pino A-
  float porcentagem = ((float)(valorSensor - VALOR_MINIMO) / (VALOR_MAXIMO - VALOR_MINIMO)) * 100;//calcula o percentual da variacão
  int maximo=100;//variável sobre o maximo
  int perigo = 2;//variável avisando perigo de passar a porcentagen
  int minimo = 0;
  if (porcentagem < 0) {
    porcentagem = 0;
  }
  else if(porcentagem > 100) {
    porcentagem = 100;
  }
  // Serial.print("Porcentagem:");
  Serial.print(porcentagem);
  // Serial.println(" ");
  // Serial.print("maximo:");
  // Serial.print(maximo);
  // Serial.print(" ");
  // Serial.print("minimo:");
  // Serial.print(minimo);
  // Serial.print(" ");
  // Serial.print("Perigo:");
  // Serial.println(perigo);
  delay(1000);
  // put your main code here, to run repeatedly:
}
```

Figura 3 - Código Arduino IDE

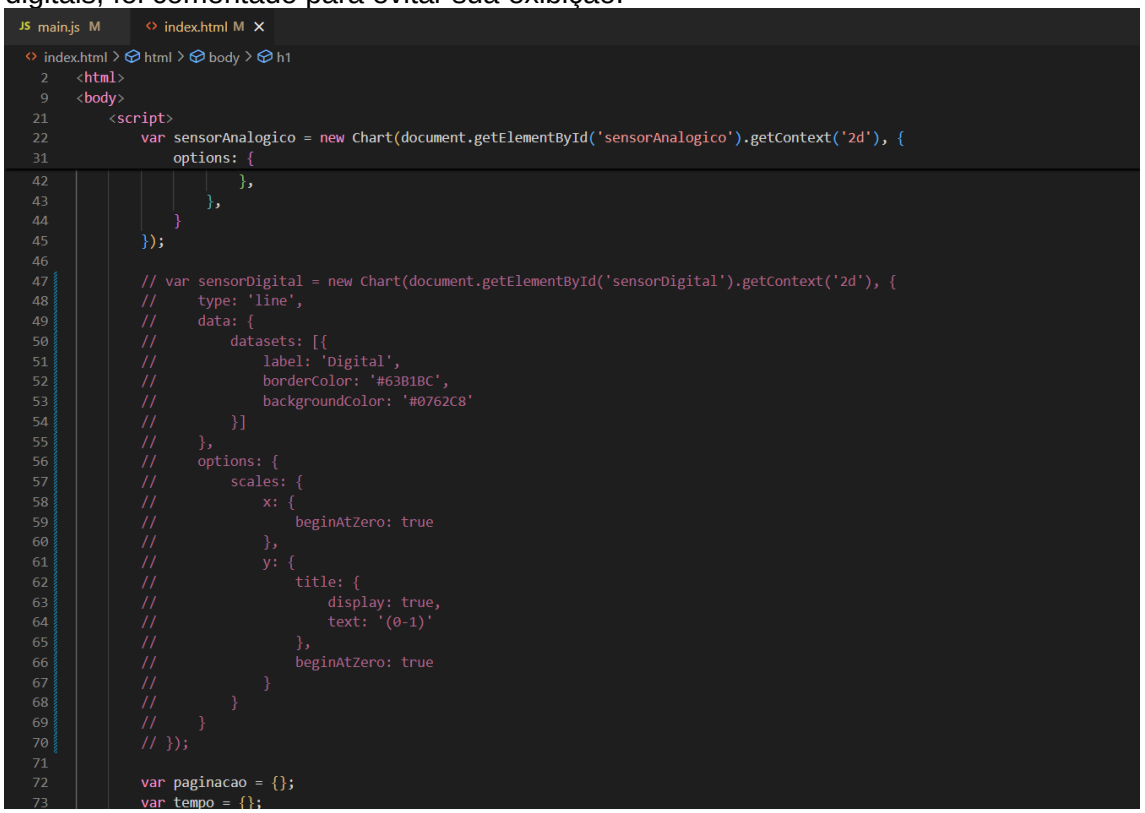
Já no código da API, especificamente no arquivo “main.js”, apenas deixamos comentado toda a parte de valores digitais, pois apenas utilizamos os valores analógicos, para sua melhor coleta constantemente.



```
main.js M  index.html M
main.js > serial > on('data') callback
const serial = async (

    // evento quando a porta serial é aberta
    arduino.on('open', () => {
        console.log(`A leitura do arduino foi iniciada na porta ${portaArduino.path} utilizando Baud Rate de ${SERIAL_BAUD_RATE}
        `);
    });

    // processa os dados recebidos do Arduino
    arduino.pipe(new serialport.ReadlineParser({ delimiter: '\r\n' })).on('data', async (data) => {
        console.log(data);
        const valores = data;
        // const sensorDigital = parseInt(valores[0]);
        const sensorAnalogico = parseFloat(valores);

        // armazena os valores dos sensores nos arrays correspondentes
        valoresSensorAnalogico.push(sensorAnalogico);
        // valoresSensorDigital.push(sensorDigital);

        // insere os dados no banco de dados (se habilitado)
        if (HABILITAR_OPERACAO_INSERIR) {

            // este insert irá inserir os dados na tabela "medida"
            await poolBancoDados.execute(
                // 'INSERT INTO medida (sensor_analogico, sensor_digital) VALUES (?, ?)',
                'INSERT INTO medida (sensor_analogico) VALUES (?)',
                [sensorAnalogico]
                // [sensorAnalogico, sensorDigital]
            );a

            // console.log("valores inseridos no banco: ", sensorAnalogico + ", " + sensorDigital);
            console.log("valores inseridos no banco: ", sensorAnalogico);

        }

    });
```

Figura 4 - Código API

Assim como no arquivo “main.js”, onde os valores digitais foram comentados para não serem exibidos, no arquivo da “index.html”, o gráfico que seria exibido com os valores digitais, foi comentado para evitar sua exibição.



```
main.js M  index.html M
index.html > html > body > h1
<html>
<body>
    <script>
        var sensorAnalogico = new Chart(document.getElementById('sensorAnalogico').getContext('2d'), {
            options: {
                        },
                    },
                }
            });

            // var sensorDigital = new Chart(document.getElementById('sensorDigital').getContext('2d'), {
            //     type: 'line',
            //     data: {
            //         datasets: [{
            //             label: 'Digital',
            //             borderColor: '#63B1BC',
            //             backgroundColor: '#0762C8'
            //         }]
            //     },
            //     options: {
            //         scales: {
            //             x: {
            //                 beginAtZero: true
            //             },
            //             y: {
            //                 title: {
            //                     display: true,
            //                     text: '(0-1)'
            //                 },
            //                 beginAtZero: true
            //             }
            //         }
            //     }
            // });

            var paginacao = {};
            var tempo = {};
```

Figura 5 - Código Index API

Resultados Iniciais

Após toda a coleta de dados e a comunicação entre o Arduino e a API funcionando corretamente, os valores coletados podem ser exibidos tanto no terminal Bash, como em uma página web.

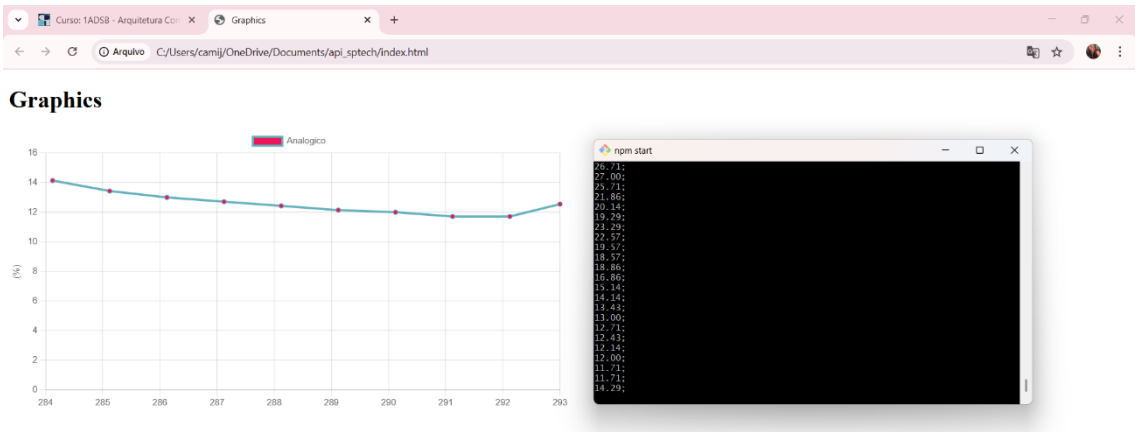


Figura 6 - Visualização dos dados