Relatório Técnico

Nº Grupo: 06
Nome dos integrantes:
01251089 - Anne Yukari Yamasaki
01251075 - Felipe da Silva Santana
01232147 - Guilherme Oliveira Mendes
01251057 - Hygor Silva Wanderlei
01251096 - João Victor Torelli de Matos
01251080 - Victor Hugo Liz Orenga
Turma: 1ADS-B

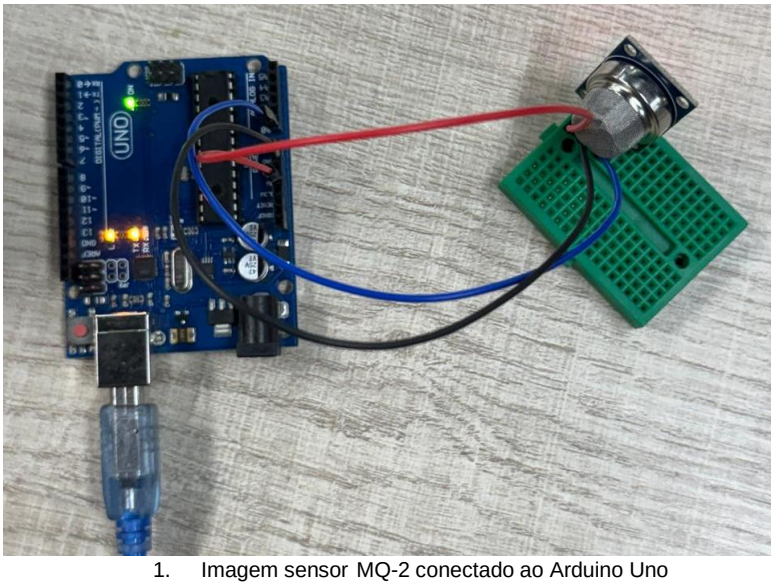
Tema do projeto: Detecção de Vazamento de Gás Natural em Condomínios Residenciais: Prevenção e Segurança
Sensor: MQ-2 (Detecção de gases inflamáveis)

Introdução

Neste trabalho, nosso grupo concentrou esforços no desenvolvimento de um sistema para detecção de vazamento de gás natural em condomínios residenciais, com o objetivo de garantir a prevenção de possíveis acidentes graves e promover a segurança, com foco inicial na cidade de São Paulo.
Para alcançar esse objetivo, empregamos o sensor MQ-2, utilizado na detecção de gases inflamáveis para simulações do projeto. O sensor foi integrado à plataforma Arduino Uno R3, que se destacou por sua eficiência e facilidade de uso.

Arquitetura de Montagem do Sensor

Abaixo está uma foto da arquitetura de montagem do projeto na mini protoboard, a imagem mostra como o sensor MQ-2 foi conectado ao Arduino Uno R3:

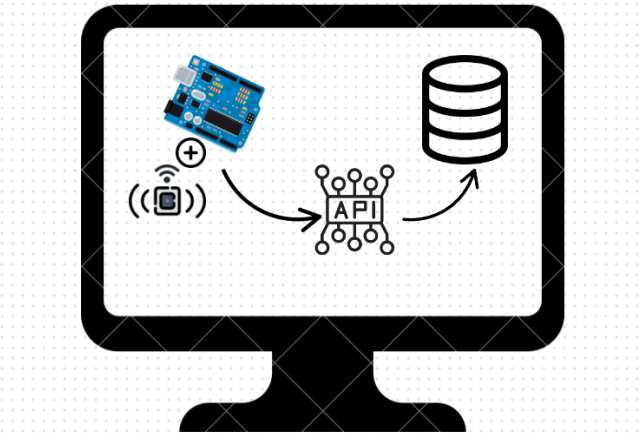


1. Imagem sensor MQ-2 conectado ao Arduino Uno

Arquitetura do Sistema

Para a realização do trabalho, utilizamos o Node.js que é um ambiente de execução de JavaScripit que pode ser executado fora do navegador, com essa ferramenta é possível desenvolver aplicações back-end, ou seja, conseguimos criar uma API que coleta os dados capturados pelo sensor e adiciona ao banco de dados selecionado, no nosso caso, mais especificamente o banco de dados MySQL Server, onde a partir dele fazemos todas as "querys" desejadas

.



1. Imagem que representa a Arquitetura do Sistema

FOTOS E PRINTS:

```
main.js > serial > on('data') callback
// importa os bibliotecas necessários
const serialport = require('serialport');
const express = require('express');
const mysql = require('mysql2');

// constantes para configurações
const SERIAL_BAUD_RATE = 9600;
const SERVIDOR_PORTA = 3300;

// habilita ou desabilita a inserção de dados no banco de dados
const HABILITAR_OPERACAO_INSERIR = true;

// função para comunicação serial
const serial = async (
    valoresSensorAnalogico
) => {

    // conexão com o banco de dados MySQL
    let poolBancoDados = mysql.createPool(
        {
            host: '127.0.0.1',
            user: 'aluno',
            password: 'Sptech#2024',
            database: 'Safe_gas',
            port: 3307
        }
    ).promise();

    // lista as portas seriais disponíveis e procura pelo Arduino
    const portas = await serialport.SerialPort.list();
    const portaArduino = portas.find((porta) => porta.vendorId == 2341 && porta.productId == 43);
    if (!portaArduino) {
        throw new Error('O arduino não foi encontrado em nenhuma porta serial');
    }

    // configura a porta serial com o baud rate especificado
    const arduino = new serialport.SerialPort(
        {
            path: portaArduino.path,
            baudRate: SERIAL_BAUD_RATE
```

2. Código usado para sincronizar o banco de dados com a API

Código do Projeto

Para a criação do código do utilizado pelo Arduíno tivemos que fazer algumas alterações, isso porque o Node.js “conversa” com os dados obtidos pelo sensor, no sentido em que os dados que chegam para o Node não podem ser do tipo string, ou seja, letras, e sim números, para que ele possa pegar esses dados e manipulá-los como desejado. Para além disso, tivemos também que fazer alguns ajustes no código do Node, no sentido em que os dados capturados pelo sensor de gás MQ-2 são do tipo float, e tivemos que ajustar essa parte, para que a visualização dos dados fossem da maneira correta.

FOTOS E PRINTS:

```ino
const int PINO_SENSOR_MQ2 = A0;  // DECLARAÇÃO DA VARIÁVEL QUE ACESSA O PINO
const int VALOR_MINIMO = 100;    // DECLARAÇÃO DA VARIÁVEL QUE GUARDA O VALOR MÍNIMO
int linhaFixa = 0;
const int VALOR_MAXIMO = 1000;  // DECLARAÇÃO DA VARIÁVEL QUE GUARDA O VALOR MÁXIMO
const float PORCENTAGEM_MIN = 5.0;
const float PORCENTAGEM_MAX = 18.0;
float porcentagem = 0;

void setup() {  // AQUI NESSE BLOCO ELE CONFIGURA O ARDUINO PARA 9600 BITS POR SEGUNDO
  Serial.begin(9600);
}
void loop() {
  int valorSensor = analogRead(PINO_SENSOR_MQ2);  // NESSE BLOCO INICIA OS CÁLCULOS PARA CAPTURA DA DENSIDADE DE GÁS E FAZ EM UMA REPETIÇÃO




  Serial.print(linhaFixa);
  Serial.print(porcentagem);
  Serial.println(15);


porcentagem = ((float)(valorSensor - VALOR_MINIMO) / (VALOR_MAXIMO - VALOR_MINIMO)) * (PORCENTAGEM_MAX - PORCENTAGEM_MIN) + PORCENTAGEM_MIN;

  if (porcentagem < 0) {
    porcentagem = 0;

  } else if (porcentagem > 100) {
    porcentagem = 100;
  }

  delay(2000);
}
```

2. Código usado no Arduíno

```node
main.js > serial > on('data') callback
const serial = async (
    const arduino = new serialport.SerialPort(
    );

    // 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.split(';');
       // const sensorDigital = parseInt(valores[0]);
        const sensorAnalogico = parseFloat(valores[0]);  // Utilizaremos apenas valores flutuantes

        // 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 (medidas) VALUES (?)',
                [sensorAnalogico]
            );
            console.log("valores inseridos no banco: ", sensorAnalogico );

        }

    });

    // evento para lidar com erros na comunicação serial
    arduino.on('error', (mensagem) => {
        console.error(`Erro no arduino (Mensagem: ${mensagem})`)
    });
}
```

3. Código usado no Node.js para visualização dos dados do Arduíno

Resultados Iniciais

Os resultados iniciais foram positivos, conseguimos manipular os códigos para a realização de uma integração entre as ferramentas, onde foi possível visualizar os dados obtidos pelo sensor ligado ao arduíno em uma página web, isso porque esses dados foram manipulados pela API e disponibilizados em uma página HTML. Além disso, foi possível sincronizar a API junto do banco de dados da Virtual Box, ou seja, os dados coletados já estão armazenados no nosso BD em ambos Ambientes de Sistema Operacional.

FOTOS E PRINTS:

```
Node.js v22.14.0
victor_hugo@victor:~/Documentos/Safe_Gas_Repositorio/safe_Gas/Desenvolvimento/ArqComp/dat-acqu-ino$ npm start

> arduino-api@2.1.0 start
> node main.js

API executada com sucesso na porta 3300
A leitura do arduino foi iniciada na porta /dev/ttyACM0 utilizando Baud Rate de 9600
00.0015
valores inseridos no banco:  0.0015
07.3115
valores inseridos no banco:  7.3115
07.2815
valores inseridos no banco:  7.2815
07.2815
valores inseridos no banco:  7.2815
07.2415
valores inseridos no banco:  7.2415
06.8115
valores inseridos no banco:  6.8115
06.8215
valores inseridos no banco:  6.8215
06.8115
valores inseridos no banco:  6.8115
06.9415
valores inseridos no banco:  6.9415
07.0815
valores inseridos no banco:  7.0815
07.1415
valores inseridos no banco:  7.1415
```

1. Dados do sensor apresentado no terminal.

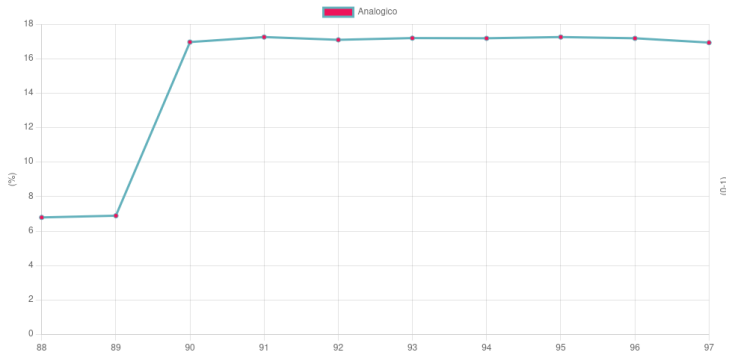
```
use Safe_gas;
create table medida (
id int primary key auto_increment,
medidas float
);

select * from medida;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

#	id	medidas
1	1	0.0015
2	2	7.3115
3	3	7.2815
4	4	7.2815
5	5	7.2415
6	6	6.8115
7	7	6.8215
8	8	6.8115
9	9	6.9415
10	10	7.0815

2. Dados do sensor apresentado no workbench após executar um select.

Graphics



3. Dados apresentados no gráfico HTML