Relatório Técnico

Nº Grupo: 02
Nome dos integrantes: Diego Iacabo, Flávia Vaz, Heloisy Oliveira, Philipi Jordan, Samuel Sousa, Vitória Lima
Turma: 1CCOK

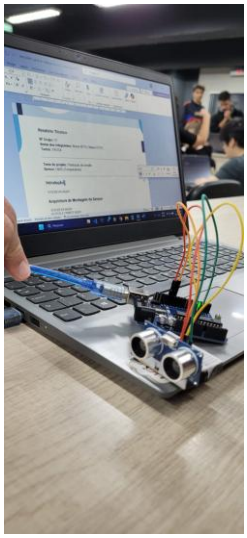
Tema do projeto: Monitoramento de vagas em vias públicas
Sensor: HC-SR04

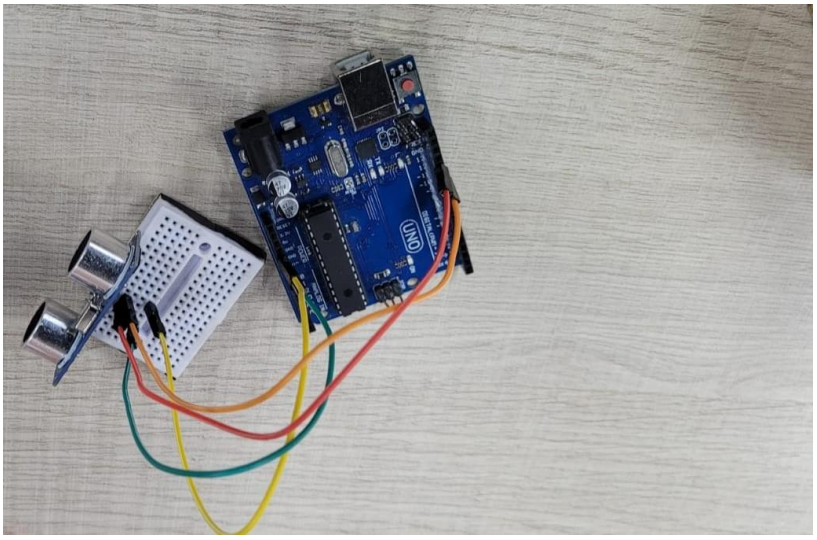
Introdução

O objetivo da solução é monitorar, por meio de sensores ultrassônicos, a ocupação de vagas em vias públicas como fonte de informação estratégica para seguradoras.

Arquitetura de Montagem do Sensor

O sensor HC-SR04 detecta a presença de um veículo em determinada vaga, para então retornar apenas duas opções: 0 se não houver a detecção, 1 caso existir um veículo na vaga.





A montagem segue da seguinte forma:

O fio amarelo é conectado na porta 5V do Arduino, que é a fonte de energia do mesmo, alimentando-o e sendo referenciado no pino do sensor HC-SR04.

O fio verde está conectado à porta GND, que tem o intuito de dar estabilidade para o circuito, sendo um ponto "morto" e referenciado no pino de mesmo nome no

sensor.

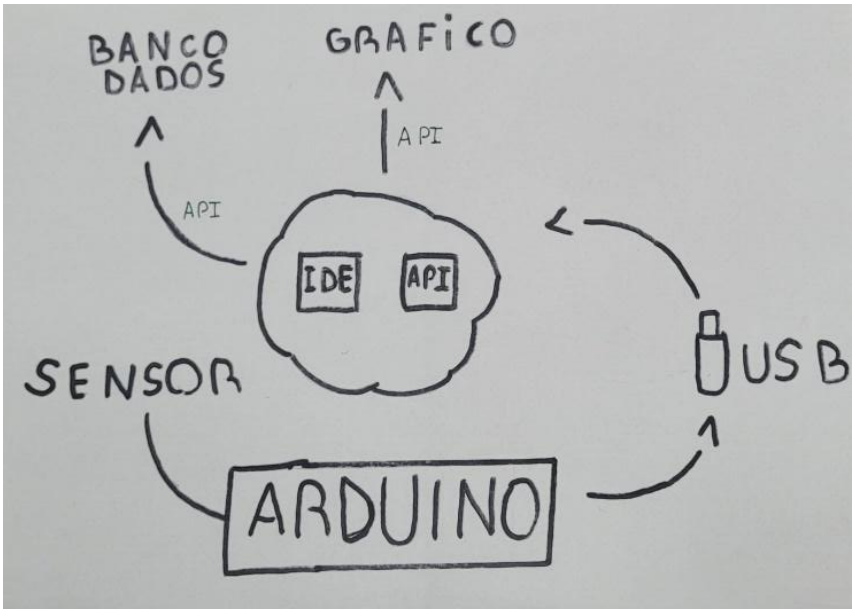
O fio de cor vermelha é conectado na porta 12 no lado digital da placa de prototipagem, sendo referenciado no pino Trigger do sensor, que tem o intuito de disparar um pulso ultrassônico para iniciar a contagem da distância entre o objeto e o sensor, enviando essa informação para o pino ECHO.

E, por fim, o fio laranja, que está conectado na porta 13 do lado Digital, referenciando o pino ECHO do sensor, tem como objetivo receber o sinal que o TRIGGER enviou e informar se houve ou não a presença de algum objeto no alcance do sensor.

Arquitetura do Sistema

A arquitetura se dá pela integração do sensor HC-SR04 e da placa Uno R3 com a API dat-acqu-ino, que, através da porta USB, recebe os dados captados e os insere no banco de dados. Além disso, ainda utilizando a API, é possível observar essas informações de forma mais intuitiva e visual em gráficos gerados por ChartJS.

Dessa forma, a arquitetura abaixo demonstra, de forma simples, a funcionalidade da aplicação, integrando o sensor com a API, possibilitando a captação, inserção e visualização dos dados.



Código do Projeto

O código do projeto tem várias etapas, pode-se dividi-las em blocos para facilitar o entendimento. No primeiro bloco, tem-se a ligação com o banco de dados, em que são colocadas as informações para o acesso como: host, usuário, senha, banco e porta (a porta, por padrão é 3306).

Em seguida é configurada a porta serial do computador, para o processamento dos dados do Arduino, para então armazená-los no banco de dados. Além disso temos as linhas dedicadas para correção de possíveis erros, para uma melhor visualização e entendimento do que está sendo trabalhado. Por último, a parte que configura o arquivo de extensão HTML, para a visualização dos dados captados em um gráfico de linha.

```
// conexão com o banco de dados MySQL
let poolBancoDados = mysql.createPool(
    {
        host: 'localhost',
        user: 'projeto',
        password: 'urubu100',
        database: 'projeto',
        port: 3306
    }
).promise();
```

```
// 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[1]);

    // 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 sensor (situacao) VALUES (?);',
            [sensorDigital]
        );
        console.log("valores inseridos no banco: " + sensorDigital);

    }
```

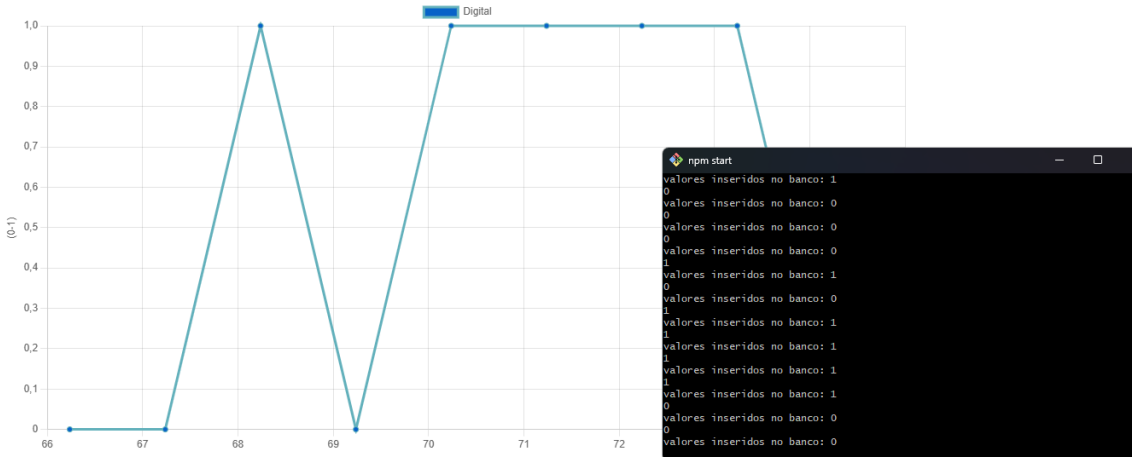
```
// evento para lidar com erros na comunicação serial
arduino.on('error', (mensagem) => {
    console.error(`Erro no arduino (Mensagem: ${mensagem}`)
});
```

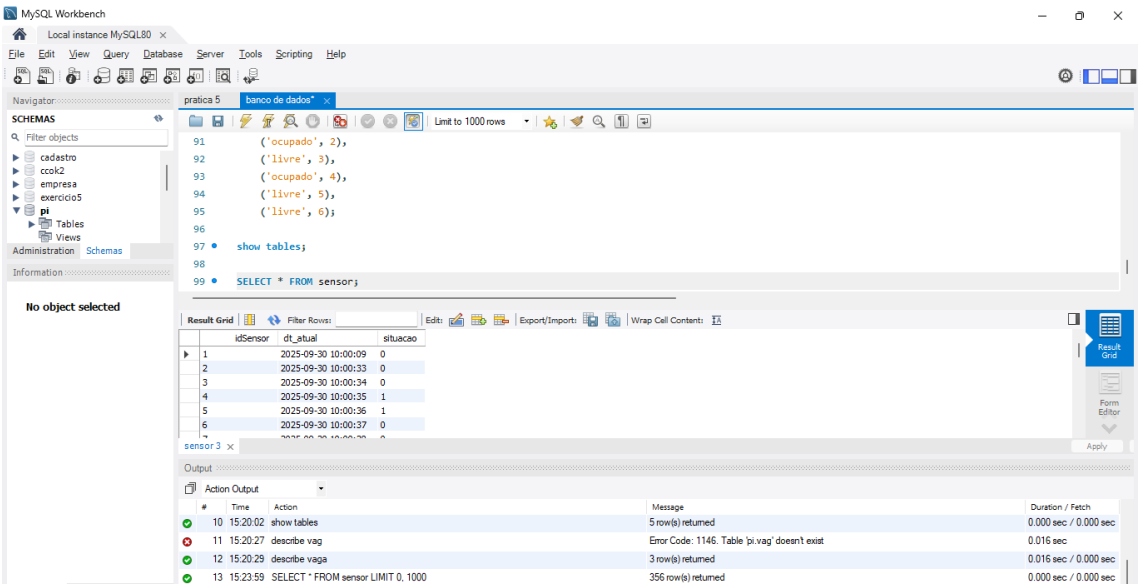
```
// função para criar e configurar o servidor web
const servidor = (
    //valoresSensorAnalogico,
    valoresSensorDigital
) => {
    const app = express();

    // configurações de requisição e resposta
    app.use((request, response, next) => {
        response.header('Access-Control-Allow-Origin', '*');
        response.header('Access-Control-Allow-Headers', 'Origin, Content-Type, Accept');
        next();
    });

    // inicia o servidor na porta especificada
    app.listen(SERVIDOR_PORTA, () => {
        console.log(`API executada com sucesso na porta ${SERVIDOR_PORTA}`);
    });

    // define os endpoints da API para cada tipo de sensor
    //app.get('/sensores/analogico', (_, response) => {
    //    return response.json(valoresSensorAnalogico);
    //});
    app.get('/sensores/digital', (_, response) => {
        return response.json(valoresSensorDigital);
    });
}
```

Resultados Iniciais

Conforme os dados supracitados, podemos visualizar os dados tanto por meio do banco de dados, utilizando as inserções realizadas pela API ou por meio do servidor web em um gráfico, que demonstra se há ou não a presença de carros na vaga.

Gráficos




MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
cadastro
ccok2
empresa
exercicio5
pi
Tables
Views
Administration Schemas
Information
No object selected
pratica 5
banco de dados*
Limit to 1000 rows
('ocupado', 2),
('livre', 3),
('ocupado', 4),
('livre', 5),
('livre', 6);
show tables;
SELECT * FROM sensor;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
idSensor dt_atual situacao
1 2025-09-30 10:00:09 0
2 2025-09-30 10:00:33 0
3 2025-09-30 10:00:34 0
4 2025-09-30 10:00:35 1
5 2025-09-30 10:00:36 1
6 2025-09-30 10:00:37 0
sensor 3
Result Grid
Form Editor
Apply
Output
Action Output
Time Action Message Duration / Fetch
10 15:20:02 show tables 5 row(s) returned 0.000 sec / 0.000 sec
11 15:20:27 describe vag Error Code: 1146. Table 'pi.vag' doesn't exist 0.016 sec
12 15:20:29 describe vaga 3 row(s) returned 0.016 sec / 0.000 sec
13 15:23:59 SELECT * FROM sensor LIMIT 0, 1000 356 row(s) returned 0.000 sec / 0.000 sec