Relatório Técnico

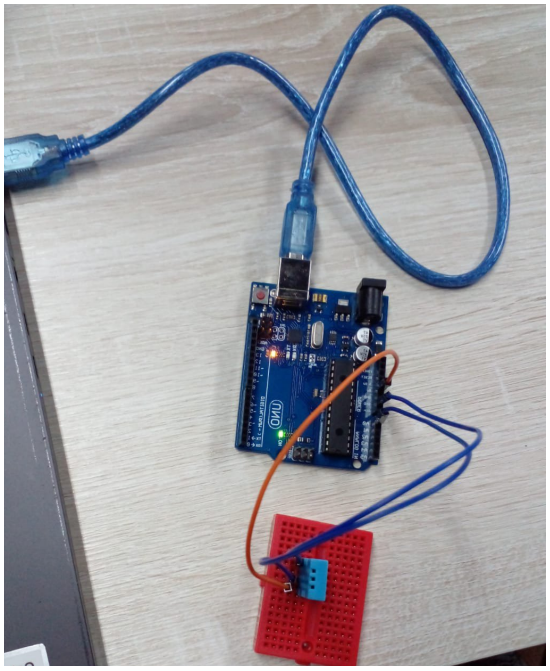
Nº Grupo: 02
Nome dos integrantes: Guilherme Marques, Lucas Sousa, Andre Ferreira, Diogo Gabriell, Juan Vieira.
Turma: 1ADSB

Tema do projeto: Medição de temperatura e umidade na APA Bororé-Colônia na prevenção de incêndios.
Sensor: DHT11 (Temperatura e Umidade)

Introdução

O projeto consiste no medição de temperatura e umidade na APA (Área de proteção ambiental) Bororé-Colônia, essa área protege a remanescentes da Mata Atlântica. Nossa solução visa monitorar de forma constante o perímetro dessa região, pois com base em evidencias, os principais pontos de inicio de incêndio são ao redor dessas áreas, por intermédio de ações humanas. Com os sensores espalhados por todo o perímetro, situações de pré-incêndio e de incêndio podem ser contidas de forma mais rápidas ou em alguns casos até prevenidas através de alertas de calor e umidade críticos.

Arquitetura de Montagem do Sensor

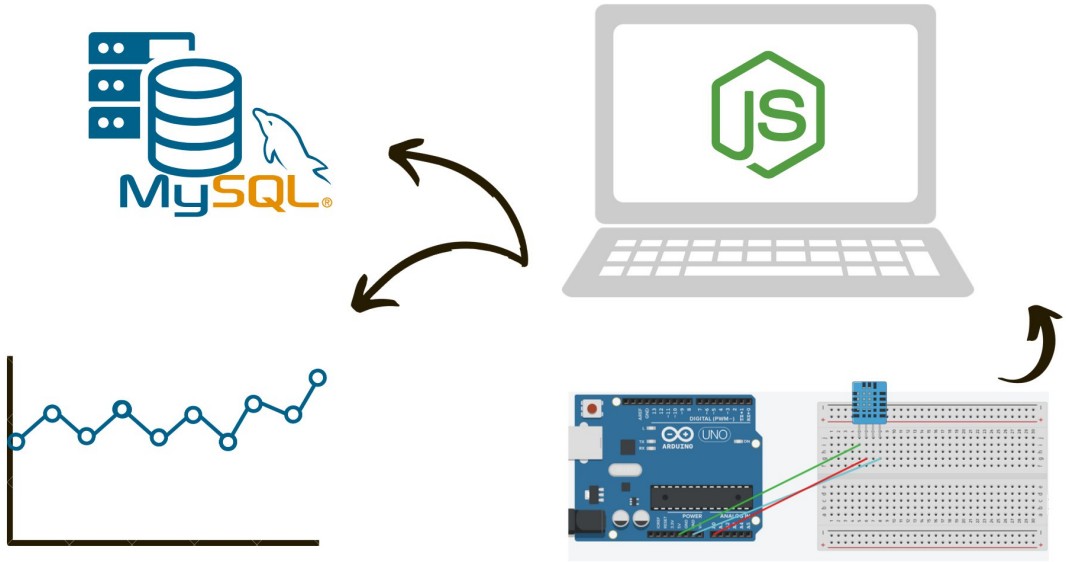


Nossa montagem do arduino foi constituída por 3 etapas, a primeira fase de separação dos dispositivos necessários, sendo eles:
- O Arduino Uno;
- Cabo USB padrão AB;
- 3 jumpers;
- Protoboard;
- Sensor de umidade e temperatura DHT11.

Na segunda fase, é onde foi feita a montagem, que também foram seguidos os seguintes passos:
- Encaixar o sensor na protoboard;
- Encaixar os jumpers em seus devidos lugares no arduino, para isso utilizamos a seguinte sigla “VA-G” onde o V significa Voltagem que no arduino conectamos o 5v, A que significa Analógico, que é a porta analógica que utilizamos, o “-“ que seria o espaço sem um jumper e por ultimo o G que seria o ground, nosso aterramento.

O terceiro e ultimo passo está em ligar o arduino no computador, via o cabo USB padrão AB, implantar o código nele e verificar o funcionamento via node.

Arquitetura do Sistema



Modelagem da arquitetura do sistema.

Modelagem lógica do banco de dados.

```sql
CREATE TABLE Dados(
    idDados INT PRIMARY KEY AUTO_INCREMENT,
    temperatura DECIMAL(4,2),
    umidade DECIMAL(4,2),
    dataAquisicao DATETIME
);
```

Criação da tabela que receberá os dados no MySQL Server.

```
// este insert irá inserir os dados na tabela "medida"
await poolBancoDados.execute(
    'INSERT INTO Dados(temperatura,umidade,dataAquisicao) VALUES (?, ?, current_timestamp())',
    [sensorUmid, sensorTemp]
);
console.log("valores inseridos no banco: ", sensorUmid + ", " + sensorTemp);
```

Configurando o valor que será inserido na tabela.

```
// conexão com o banco de dados MySQL
let poolBancoDados = mysql.createPool(
    {
        host: 'localhost',
        user: 'root',
        password: '0910',
        database: 'Sprint2',
        port: 3306
    }
).promise();
```

Criando a conexão com o banco de dados.

Código do Projeto

```
#include "DHT.h" //Inclui uma biblioteca externa própria do sensor

#define TIPO_SENSOR DHT11 //Define o sensor que está sendo utilizado
const int PORTA_DHT11= A0; //Define em qual porta analógica está conectado

DHT sensorDHT(PORTA_DHT11, TIPO_SENSOR); //Define que as informações coletadas se referem ao sensor informado

void setup() {
  Serial.begin(9600); //Define a taxa de transferência em bits p/segundo para transmissão serial(baud rate)
sensorDHT.begin(); //Iniciar a função atrelada ao sensorDHT
}

void loop() { //Define um processo que será feito em repetição

  float umidade = sensorDHT.readHumidity(); // Cria a variável e define o valor captado pela leitura
  float temperatura = sensorDHT.readTemperature(); // Cria a variável e define o valor captadop ela leitura

      Serial.print(umidade); //Preenche a string com o valor da variavel "umidade"
      Serial.print(";");
      Serial.println(temperatura);//Preenche a string com o valor da variavel "temperatura"

  delay (1000); //Define o tempo para refazer a ação do loop em ms
}
```

Código que será enviado para o arduino permitindo a execução correta do programa.

```
function obterDados(grafico, endpoint) {
    fetch('http://localhost:3300/sensores/' + endpoint)
        .then(response => response.json())
        .then(valores => {
            if (paginacao[endpoint] == null) {
                paginacao[endpoint] = 0;
            }
            if (tempo[endpoint] == null) {
                tempo[endpoint] = 0;
            }

            var ultimaPaginacao = paginacao[endpoint];
            paginacao[endpoint] = valores.length;
            valores = valores.slice(ultimaPaginacao);

            valores.forEach((valor) => {
                if (grafico.data.labels.length == 10 && grafico.data.datasets[0].data.length == 10) {
                    grafico.data.labels.shift();
                    grafico.data.datasets[0].data.shift();

                }

                grafico.data.labels.push(tempo[endpoint]++);
                grafico.data.datasets[0].data.push(parseFloat(valor));
                grafico.update();
            });
        })
        .catch(error => console.error('Erro ao obter dados:', error));
}

setInterval(() => {
    obterDados(sensorTemp, 'temperatura');
    obterDados(sensorUmid, 'umidade');
```

Função que obtêm os dados.

```
// 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
    }
);

// 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 sensorTemp = parseFloat(valores[0]);
    const sensorUmid = parseFloat(valores[1]);

    // armazena os valores dos sensores nos arrays correspondentes
```

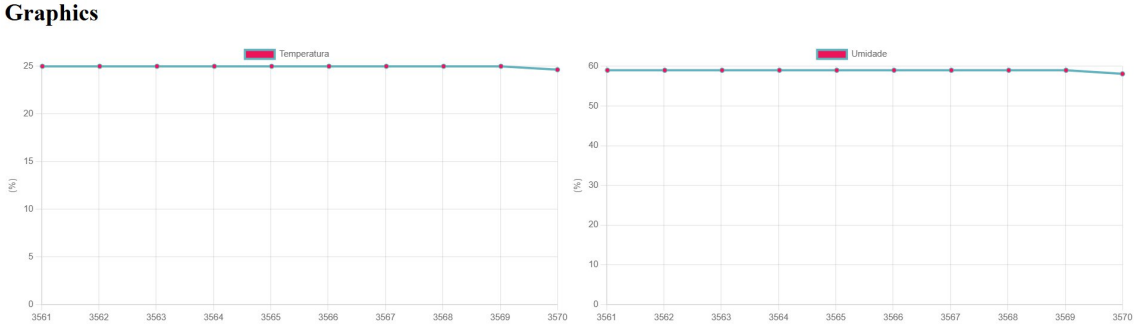
Configuração das portas e processamento dos dados.

```
const servidor = (
    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/temperatura', (_, response) => {
        return response.json(valoresSensorTemp);
    });
    app.get('/sensores/umidade', (_, response) => {
        return response.json(valoresSensorUmid);
    });
}

// função principal assíncrona para iniciar a comunicação serial e o servidor web
(async () => {
    // arrays para armazenar os valores dos sensores
    const valoresSensorTemp = [];
    const valoresSensorUmid = [];

    // inicia a comunicação serial
    await serial(
        valoresSensorTemp,
        valoresSensorUmid
    );

    // inicia o servidor web
    servidor(
        valoresSensorTemp,
        valoresSensorUmid
    );
})();
```

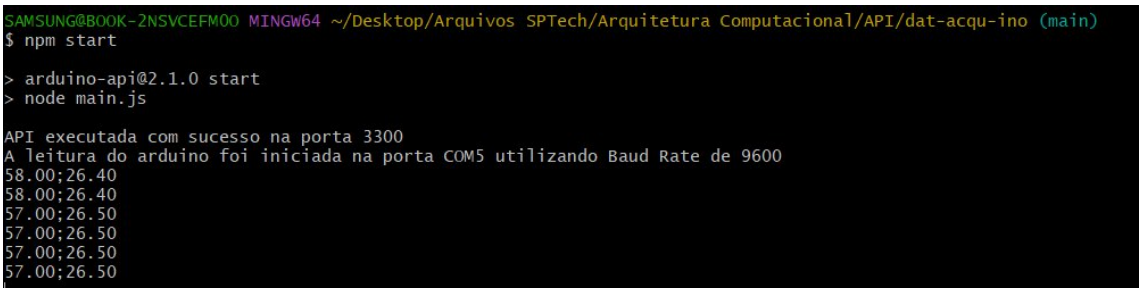
Inicialização e definição dos endpoints.

(obs: Só colocamos os códigos que modificamos).

Resultados Iniciais



Foram gerados dois gráficos, ambos utilizados para a visualização dos dados capturados pelo sensor, um sendo de temperatura e o outro de umidade.



Visualização via o shell bash, permitindo uma visualização mais precisa desses dados.