

# Relatório Técnico

**Nº Grupo:** 02

**Nome dos integrantes:** Luís Fernando, Alexsander Torres, Fernanda de Oliveira, Giovanna de Oliveira, Kauany Marques e Isabelle de Carvalho

**Turma:** 1ADSA

---

**Tema do projeto:** Monitoramento de temperatura em silos de soja

**Sensor:** LM35 (Temperatura)

---

## Introdução

O projeto realizado visa armazenar os dados capturados pelo sensor a um banco de dados com intenção de demonstrá-los em um dashboard.

O sistema utilizado monitora os dados capturados por um sensor, que conectado a um Arduino, envia os dados recebidos para o framework utilizado, o Node.JS.

Os dados serão armazenados no banco de dados MySQL Server e exibidos em um dashboard web, por meio do Chart.js.

## Arquitetura de Montagem do Sensor

### 1. Itens utilizados

<b>Arduino</b>	<b>Arduino Uno</b>
<b>Protoboard</b>	<b>Mini Protoboard 170</b>
<b>Jumpers</b>	<b>3 Fios Macho-Macho</b>
<b>Sensor</b>	<b>LM35 (Temperatura)</b>

## 2. Configuração dos itens

Os itens foram configurados em uma ordem específica, primeiro aplicamos o sensor LM35 na mini protoboard e logo depois montamos os jumpers macho-macho em fileiras, aplicando o primeiro jumper na porta 5V e o segundo na porta A0, e por fim o último jumper estará configurado na porta GND para a aterragem do sistema.

## 3. Foto da arquitetura do sensor

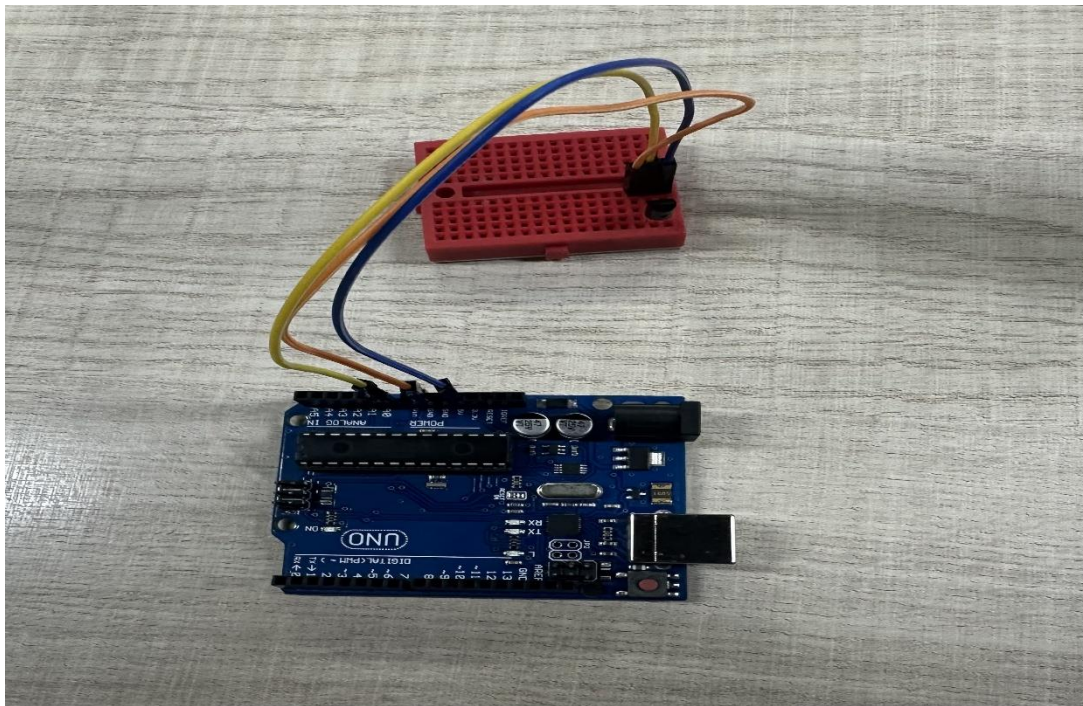


Foto 1: Arquitetura do Sensor

## Arquitetura do Sistema

### 1. Tabela especificando arquitetura do sistema

<b>Hardware</b>	<b>Arduino, sensor e jumpers</b>
<b>Back-end</b>	<b>Código que permitirá funcionamento e conexão com o site para melhor utilização</b>
<b>Software</b>	<b>Permite melhor interação com o cliente, com páginas interativas e exibição em dashboard</b>

Utilizamos a plataforma de execução de código Node.js, que permite a execução da linguagem fora do navegador (Google). Além disto, utilizamos também o Banco de dados MySQL Server, que é o meio de armazenagem dos dados captados pelo sensor.

## 2. Diagrama especificando arquitetura do sistema

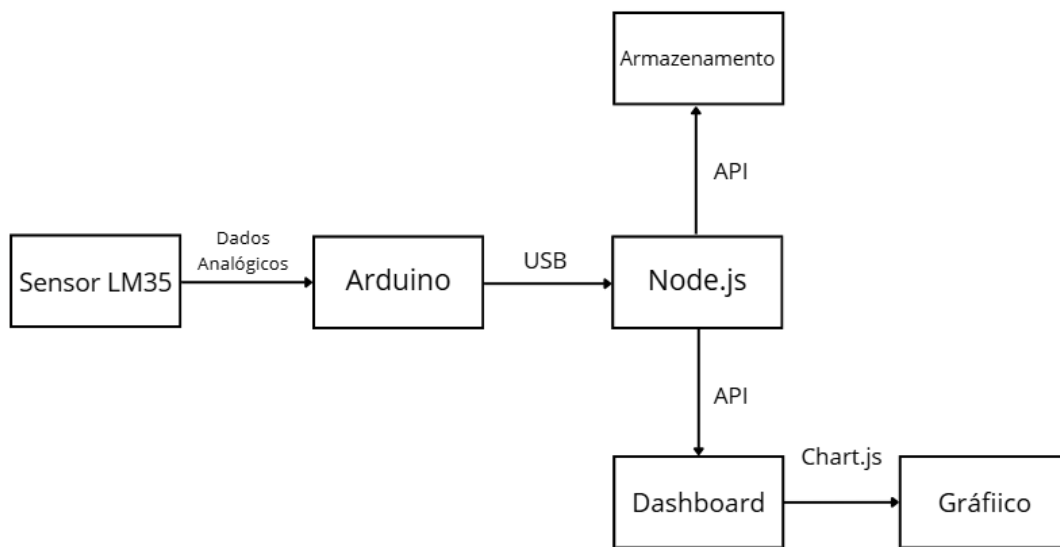


Foto 2: Diagrama do Sistema

### Especificações:

1. Conexão entre o sensor de temperatura LM35 e a placa Arduino Uno, por meio de jumpers.
2. Dados analógicos (representados de forma contínua) enviados ao Arduino.
3. Conexão via USB entre Arduino e máquina, seguido de conexão com Node.js.
4. Por meio de uma API, via MySQL Server, os dados obtidos são armazenados.
5. Com outra API, uma dashboard será exibida ao cliente, permitindo monitoramento em tempo real.

## 6. Exibição em gráficos.

### Código do Projeto

#### 1. Código Arduino IDE

O código utilizado foi configurado com base na porta utilizada na montagem do Arduino, assim capturando a temperatura ambiente e configurando os dados para delimitar a temperatura máxima e mínima com base em manter a saúde dos grãos de soja no final do processo.

Por fim, o código printa as informações obtidas e as delimitadas para melhor absorção dos dados.

```
// Definição da variável para recebimento dos dados da porta analógica A0
const int PINO_SENSOR_TEMPERATURA = A0;

// Definição da variável do tipo float (aceitar valores decimais) para alocar a temperatura futuramente.
float temperaturaCelsius;

// Função para inicializar junto do programa. Irá rodar apenas uma vez.
void setup(){

    Serial.begin(9600); // Inicialização do Arduino

}

// Função que vai se repetir enquanto o código se manter aberto
void loop() {

    // Variável para leitura dos dados da porta analógica A0 definida no começo do código
    int valorLeitura = analogRead(PINO_SENSOR_TEMPERATURA);

    // Conversão dos dados recebidos pela leitura para °C
    temperaturaCelsius = (valorLeitura * 5.0 / 1023.0) / 0.01;

    // Variável para valor fixo de temperatura máxima que o grão pode atingir enquanto se mantém saudável.
    int tempMaxima = 30;

    // Variável para valor fixo de temperatura mínima que o grão pode atingir enquanto se mantém saudável.
    int tempMinima = 20;
```

Foto 3: Código do Arduino IDE

```
// Variável para valor ideal da temperatura do grão para maior eficiência do grão.
int tempIdeal = 25;

// Bloco de estrutura de decisão. Temperaturas entre 25°C e 23°C serão acrescidas em 1°C e entre 27°C e 26°C diminuídas em 1°C apenas para demonstração.
if (temperaturaCelsius >= 23){

    if (temperaturaCelsius <= 25){

        temperaturaCelsius += 1;

    }

    else if (temperaturaCelsius <= 27){

        temperaturaCelsius -= 1;

    }

};
```

Foto 4: Código do Arduino IDE

```

Serial.print("Temp_Atual:"); // Nome da label (rótulo) para temperatura atual. Sintaxe: ("nome_da_label:"). ATENÇÃO: o caractere ":" é indispensável.
Serial.println(temperaturaCelsius); // Registro do valor para a label Temp_Atual
Serial.print(","); // Necessário para divisão das labels e valores
Serial.print("Temp_Ideal:"); // Label para temperatura ideal
Serial.print(tempIdeal); // Registro do valor para a label Temp_Ideal
Serial.print(","); // Necessário para divisão das labels e valores
Serial.print("Temp_Maxima:"); // Label para temperatura máxima
Serial.print(tempMaxima); // Registro do valor para a label Temp_Maxima
Serial.print(","); // Necessário para divisão das labels e valores
Serial.print("Temp_Minima:"); // Label para temperatura mínima
Serial.println(tempMinima); // Use Serial.println(). Registro do valor para a label Temp_Minima

delay(500); // Tempo de "descanso" entre uma execução e outra do loop.

```

Foto 5: Código do Arduino IDE

## 2. Código Node.js

Configuramos o código para aceitar somente valores analógicos, assim estando de acordo com o código inserido no sistema do Arduino IDE.

Além disso colocamos a virgula como delimitador para também ficar de acordo com o código feito e não dar erro de leitura.

```

// 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[1]);
  const sensorAnalogico = parseFloat(valores[0]);
});

```

Foto 6: Código do Main.js no Node.js

## Resultados Iniciais

### 1. Problemas iniciais enfrentados

Enfrentamos problemas relacionados a configuração da porta utilizada, já que o sistema não estava reconhecendo.

Visto a existência desse problema, os membros do grupo optaram por entender melhor o que estava causando essa interferência.

```

Microsoft Windows [versão 10.0.22631.5126]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Windows\System32>cd C:\Users\isabe\Documents\Projeto API\dat-acqu-ino
C:\Users\isabe\Documents\Projeto API\dat-acqu-ino>npm start

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

node:internal/modules/cjs/loader:1228
  throw err;
  ^

Error: Cannot find module 'serialport'
Require stack:
- C:\Users\isabe\Documents\Projeto API\dat-acqu-ino\main.js
  at Function._resolveFilename (node:internal/modules/cjs/loader:1225:15)
  at Function._load (node:internal/modules/cjs/loader:1055:27)
  at TracingChannel.traceSync (node:diagnostics_channel:322:14)
  at wrapModuleLoad (node:internal/modules/cjs/loader:220:24)
  at Module.require (node:internal/modules/cjs/loader:1311:12)
  at require (node:internal/modules/helpers:136:16)
  at Object.<anonymous> (C:\Users\isabe\Documents\Projeto API\dat-acqu-ino\main.js:2:20)
  at Module._compile (node:internal/modules/cjs/loader:1554:14)
  at Object.<js> (node:internal/modules/cjs/loader:1706:10)
  at Module.load (node:internal/modules/cjs/loader:1289:32) {
  code: 'MODULE_NOT_FOUND',
  requireStack: [ 'C:\\Users\\isabe\\Documents\\Projeto API\\dat-acqu-ino\\main.js' ]
}

Node.js v22.14.0

```

Foto 7: Código no Prompt de Comando

Outro problema enfrentado estava relacionado à forma como os dados estavam configurados e printados no sistema, visto que o código no main.js não aceitava dados printados em modo 'String'.

Portanto com esse erro em execução o servidor somente lia as informações inseridas no formato de texto, esquecendo as informações em número, sendo assim inexecutável no chart.js.

```

C:\Users\isabe\Documents\Projeto API\dat-acqu-ino>npm start

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

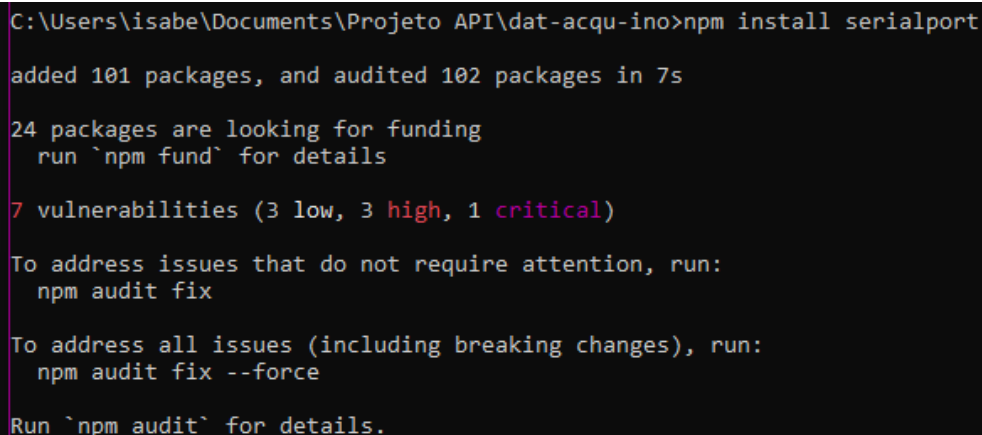
API executada com sucesso na porta 3300
A leitura do arduino foi iniciada na porta COM3 utilizando Baud Rate de 9600
inima:20
Temp_Amp_Atual:24.90,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.90,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_ATemp_Atual:24.42,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.90,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.42,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.90,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.90,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.42,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.90,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.42,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.90,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.42,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20
Temp_Atual:24.42,Temp_Ideal:25,Temp_Maxima:30,Temp_Minima:20

```

Foto 8: Código no Prompt de Comando

## 2. Resolução dos problemas

Para a resolução do problema de porta enfrentado, nosso grupo utilizou o seguinte código abaixo, que configurou automaticamente a porta utilizada pelo servidor no sistema e resolveu os problemas iniciais.



```
C:\Users\isabe\Documents\Projeto API\dat-acqu-ino>npm install serialport
added 101 packages, and audited 102 packages in 7s

24 packages are looking for funding
  run `npm fund` for details

7 vulnerabilities (3 low, 3 high, 1 critical)

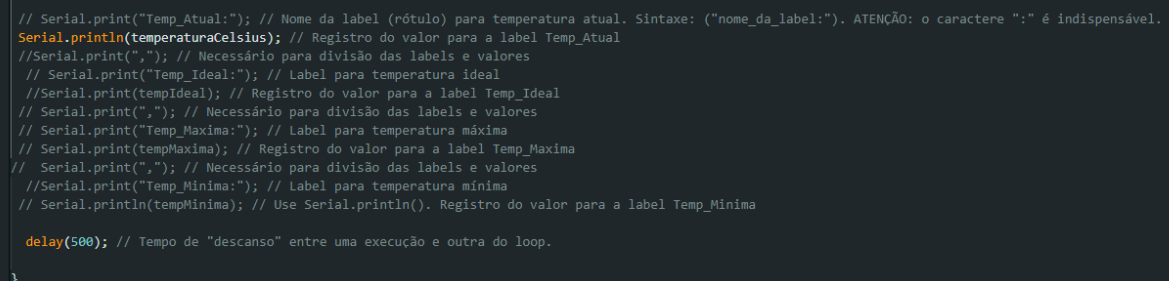
To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Foto 9: Código no Prompt de Comando

E para a resolução de problema enfrentados no código do Arduino IDE, nossa equipe comentou os códigos que printavam dados em 'String' e retirou os dados printados adicionais, deixando somente o dado capturado da temperatura ambiente para o melhor funcionamento da API no sistema.



```
// Serial.print("Temp_Atual:"); // Nome da label (rótulo) para temperatura atual. Sintaxe: ("nome_da_label:"). ATENÇÃO: o caractere ":" é indispensável.
Serial.println(temperaturaCelsius); // Registro do valor para a label Temp_Atual
//Serial.print(","); // Necessário para divisão das labels e valores
// Serial.print("Temp_Ideal:"); // Label para temperatura ideal
//Serial.print(tempIdeal); // Registro do valor para a label Temp_Ideal
// Serial.print(","); // Necessário para divisão das labels e valores
// Serial.print("Temp_Maxima:"); // Label para temperatura máxima
// Serial.print(tempMaxima); // Registro do valor para a label Temp_Maxima
// Serial.print(","); // Necessário para divisão das labels e valores
//Serial.print("Temp_Minima:"); // Label para temperatura mínima
// Serial.println(tempMinima); // Use Serial.println(). Registro do valor para a label Temp_Minima

delay(500); // Tempo de "descanso" entre uma execução e outra do loop.
}
```

Foto 10: Código no Arduino IDE

### 3. Resultados Finais

Como resultados, obtivemos sucesso na conexão entre a API e a leitura do Arduino, o que permite a execução plena do projeto, a partir das leituras que serão exibidas durante o processo.

```
API executada com sucesso na porta 3300
A leitura do arduino foi iniciada na porta COM3 utilizando Baud Rate de 9600
Deseja finalizar o arquivo em lotes (S/N)? s

C:\Users\isabe\Documents\Projeto API\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 COM3 utilizando Baud Rate de 9600
.93
25.93
25.93
25.93
25.93
25.44
25.44
25.93
25.44
25.93
25.93
25.93
25.93
25.93
```

Foto 11: Código no Prompt de Comando

Na foto 11, temos a leitura do Arduino. Estes valores serão armazenados no Banco de Dados e exibidos no Dashboard, como nos exemplos a seguir.



Graphics

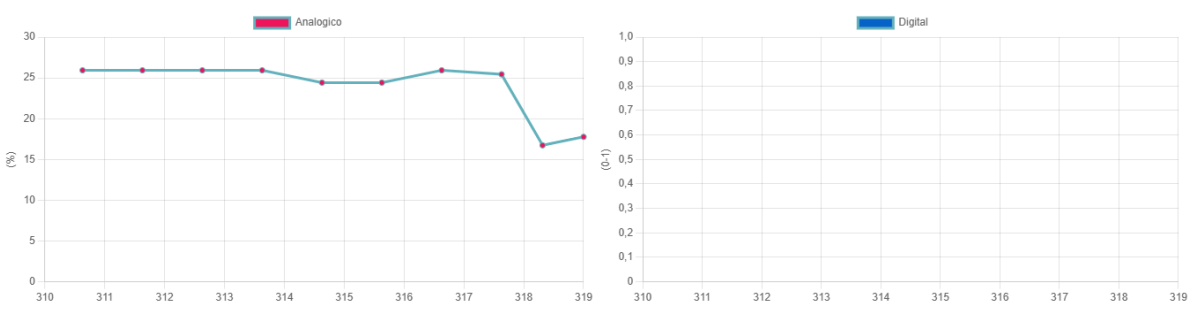


Foto 12: Resultados de gráfico no index.html

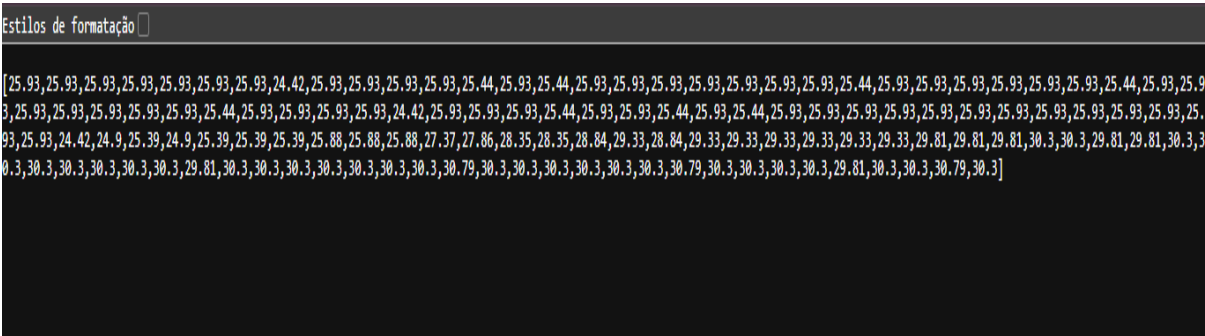


Foto 13: Resultados na porta serial utilizada