

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
COORDENAÇÃO ENSINO A DISTÂNCIA – CEAD
ESCOLA POLITÉCNICA E DE ARTES
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS



Projeto Integrador

Ana Gabriela Silva Durães
Lara Eduarda Ribeiro de Assis
Suzanny Síntique Nascimento Martins

GOIÂNIA,
2025

Relatório do Projeto - Sistema de Monitoramento com Arduino e Integração Java

1. Introdução

O projeto teve como objetivo o desenvolvimento de um sistema de monitoramento de ambiente utilizando sensores conectados ao Arduino, com exibição em um display LCD e comunicação com uma aplicação em Java. A proposta simula uma casa inteligente, monitorando variáveis como temperatura, umidade e luminosidade, além de emitir alertas conforme os dados captados pelos sensores.

2. Divisão de Tarefas

Ana: Responsável pela montagem do circuito físico no Arduino, pela programação em C++ e pela integração com o display LCD e sensores.

Lara: Responsável pela estrutura de software em Java que realiza a leitura dos dados enviados via porta serial, a modelagem do sistema com UML e organização da documentação geral do projeto.

Suzanny: Responsável pela criação do protótipo da interface visual do sistema, apresentado no formato de um aplicativo, conforme o arquivo em PDF anexo.

3. Parte Física - Circuito Arduino (por Ana)

A montagem física contou com:

- Umidade: Sensor capacitivo de umidade (A1)
- Temperatura: Sensor analógico (A0)
- Luminosidade: Fotorresistor (LDR) (A2)
- Display LCD 16x2 para exibir os dados
- LED de alerta conectado à porta digital 8

Código-fonte do Arduino:

```
#include <LiquidCrystal.h>
```

```
int rs = 6, en = 7, d4 = 2, d5 = 3, d6 = 4, d7 = 5;  
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

```
int ledAlerta = 8;
```

```
int sensorUmidade = A1;  
int sensorTemperatura = A0;  
int sensorLuz = A2;
```

```
void setup() {  
  lcd.begin(16, 2);  
  Serial.begin(9600);  
  pinMode(sensorTemperatura, INPUT);  
  pinMode(sensorUmidade, INPUT);  
  pinMode(sensorLuz, INPUT);  
  pinMode(ledAlerta, OUTPUT);  
  lcd.clear();  
  lcd.setCursor(0, 0);  
  lcd.print("Sistema Iniciado");  
  delay(1500);  
}
```

```
void loop() {  
  int umidadeValor = analogRead(sensorUmidade);  
  lcd.clear();  
  lcd.setCursor(0, 0);  
  lcd.print("Umidade: ");  
  if (umidadeValor <= 20) {  
    lcd.print("BAIXA");  
    Serial.print("BAIXA\n");  
  } else if (umidadeValor <= 70) {  
    lcd.print("MEDIA");  
    Serial.print("MEDIA\n");  
  } else {  
    lcd.print("TOTAL");  
    Serial.print("TOTAL\n");  
  }  
  delay(1500);
```

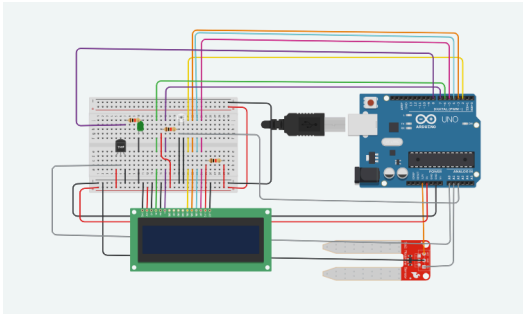
```
  int temperaturaC = 30;  
  lcd.clear();  
  lcd.setCursor(0, 0);  
  lcd.print("Temp: ");  
  lcd.print(temperaturaC);  
  lcd.print(" C");
```

```
  if (temperaturaC >= 30 && temperaturaC <= 35) {  
    digitalWrite(ledAlerta, HIGH);  
    lcd.setCursor(0, 1);
```

```

    lcd.print("Alerta: Quente!");
} else if (temperaturaC > 35) {
    digitalWrite(ledAlerta, HIGH);
    lcd.setCursor(0, 1);
    lcd.print("Perigo: Muito Quente!");
} else {
    digitalWrite(ledAlerta, LOW);
}
delay(1500);
}

```



4. Parte Lógica - Software Java (por Lara)

Foi desenvolvida uma estrutura em Java responsável por receber, processar e armazenar os dados dos sensores. O sistema utiliza uma arquitetura orientada a objetos, com as seguintes classes principais:

- ArduinoMonitor: Controla a leitura dos dados e gerenciamento de alertas.

```

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

public class ArduinoMonitor {
    private SerialConnection serialConnection;
    private List<SensorListener> ouvintes;
    private ScheduledExecutorService executor;

    public ArduinoMonitor() {
        this.serialConnection = new SerialConnection();
        this.ouvintes = new ArrayList<>();
    }

    public void iniciarMonitoramento() {
        serialConnection.conectar();

        executor = Executors.newSingleThreadScheduledExecutor();
        executor.scheduleAtFixedRate(() -> {
            String dadosBrutos = serialConnection.lerDados();
            DadosSensor dados = processarDados(dadosBrutos);
            notificarDados(dados);
            verificarAlertas(dados);
        }, initialDelay:0, period:3, TimeUnit.SECONDS);
    }

    private DadosSensor processarDados(String dadosBrutos) {
        String[] partes = dadosBrutos.split(regex:","");
        double temperatura = Double.parseDouble(partes[0]);
        double umidade = Double.parseDouble(partes[1]);
    }
}

```

```

1 ArduinoMonitor.java > ...
2 public class ArduinoMonitor {
3     private DadosSensor processarDados(String dados brutos) {
4         double umidade = Double.parseDouble(partes[1]);
5         double luminosidade = Double.parseDouble(partes[2]);
6         return new DadosSensor(temperatura, umidade, luminosidade);
7     }
8
9     private void verificarAlertas(DadosSensor dados) {
10         if (dados.getTemperatura() > 35.0) {
11             Alerta alerta = new Alerta("Temperatura alta: " + dados.getTemperatura() + "°C", TipoAlerta.PERIGO);
12             notificarAlerta(alerta);
13         } else if (dados.getTemperatura() > 30.0) {
14             Alerta alerta = new Alerta("Temperatura acima do ideal: " + dados.getTemperatura() + "°C", TipoAlerta.ALERTA);
15             notificarAlerta(alerta);
16         }
17
18         if (dados.getLuminosidade() < 20.0) {
19             Alerta alerta = new Alerta("Luminosidade baixa: " + dados.getLuminosidade() + "%", TipoAlerta.ALERTA);
20             notificarAlerta(alerta);
21         }
22     }
23
24     public void adicionarOuvinete(SensorListener ouvinete) {
25         ouvintes.add(ouvinete);
26     }
27
28     private void notificarDados(DadosSensor dados) {
29         for (SensorListener ouvinete : ouvintes) {
30             ouvinete.onDadosRecebidos(dados);
31         }
32     }
33 }

```

```

1 ArduinoMonitor.java > ...
2 public class ArduinoMonitor {
3
4     public void adicionarOuvinete(SensorListener ouvinete) {
5         ouvintes.add(ouvinete);
6     }
7
8     private void notificarDados(DadosSensor dados) {
9         for (SensorListener ouvinete : ouvintes) {
10             ouvinete.onDadosRecebidos(dados);
11         }
12     }
13
14     private void notificarAlerta(Alerta alerta) {
15         for (SensorListener ouvinete : ouvintes) {
16             ouvinete.onAlertaRecebida(alerta);
17         }
18     }
19
20     Run/Debug
21     public static void main(String[] args) {
22         ArduinoMonitor monitor = new ArduinoMonitor();
23
24         monitor.adicionarOuvinete(new SensorListener() {
25             @Override
26             public void onDadosRecebidos(DadosSensor dados) {
27                 System.out.printf("Dados recebidos: Temperatura = %.2f°C | Umidade = %.2f% | Luminosidade = %.2f%%\n",
28                     dados.getTemperatura(), dados.getUmidade(), dados.getLuminosidade());
29             }
30
31             @Override
32             public void onAlertaRecebida(Alerta alerta) {
33                 System.out.println("Á ALERTA: " + alerta);
34             }
35         });
36
37         monitor.iniciarMonitoramento();
38     }
39 }

```

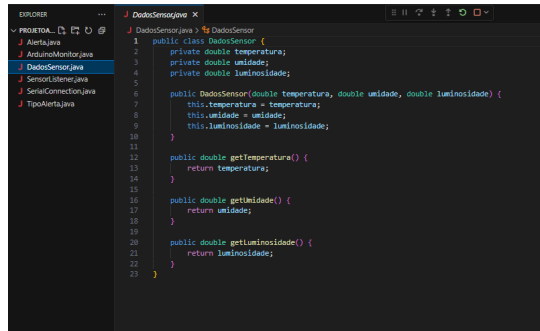
- **SerialConnection:** Gerencia a comunicação com a porta serial.

```

1 Explorer
2 ...
3 SerialConnection.java X
4 SerialConnection.java > SerialConnection > conectar()
5 1 import java.util.Random;
6 2
7 3 public class SerialConnection {
8 4     private boolean conectado = false;
9 5     private Random random = new Random();
10 6
11 7     public void conectar() {
12 8         conectado = true;
13 9         System.out.println("Conectado à porta serial (simulada)");
14 10     }
15 11
16 12     public String lerDados() {
17 13         double temperatura = 25 + random.nextDouble() * 15;
18 14         double umidade = 40 + random.nextDouble() * 30;
19 15         double luminosidade = random.nextDouble() * 100;
20 16         return String.format("T: %.2f, U: %.2f, L: %.2f", temperatura, umidade, luminosidade);
21 17     }
22 18
23 19     public boolean estaConectado() {
24 20         return conectado;
25 21     }
26 22 }

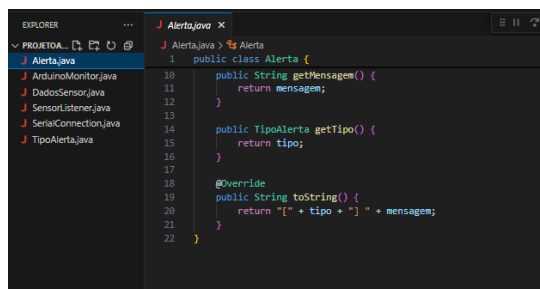
```

- **DadosSensor:** Representa os dados de sensores com atributos como temperatura, umidade e luminosidade.



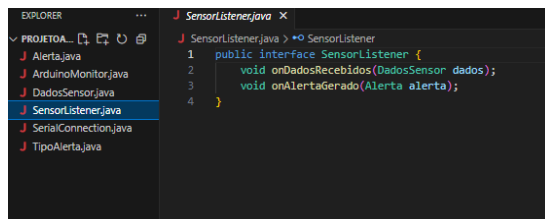
```
1 public class DadosSensor {
2     private double temperatura;
3     private double umidade;
4     private double luminosidade;
5
6     public DadosSensor(double temperatura, double umidade, double luminosidade) {
7         this.temperatura = temperatura;
8         this.umidade = umidade;
9         this.luminosidade = luminosidade;
10    }
11
12    public double getTemperatura() {
13        return temperatura;
14    }
15
16    public double getUmidade() {
17        return umidade;
18    }
19
20    public double getLuminosidade() {
21        return luminosidade;
22    }
23 }
```

- Alerta: Gera alertas com base em condições específicas.



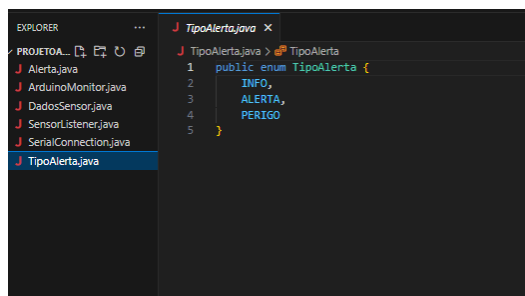
```
1 public class Alerta {
2
3     private String mensagem;
4
5     public String getMensagem() {
6         return mensagem;
7     }
8
9     public TipoAlerta getTipo() {
10        return tipo;
11    }
12
13    @Override
14    public String toString() {
15        return "[" + tipo + "] " + mensagem;
16    }
17 }
```

- SensorListener: Interface para notificações em tempo real.



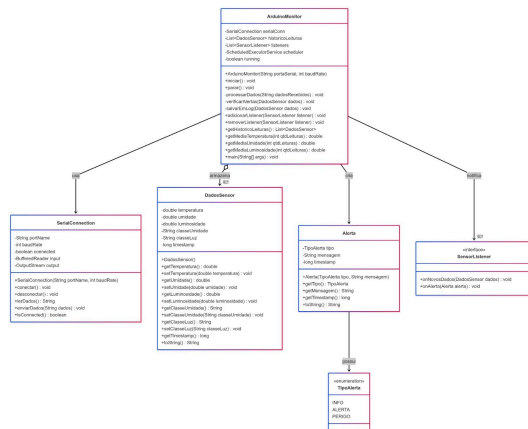
```
1 public interface SensorListener {
2     void onDadosRecebidos(DadosSensor dados);
3     void onAlertaGerado(Alerta alerta);
4 }
```

- TipoAlerta: Enum para classificação dos alertas (INFO, ALERTA, PERIGO).



```
1 public enum TipoAlerta {
2     INFO,
3     ALERTA,
4     PERIGO
5 }
```

Diagrama UML:



Código Java completo está no arquivo ZIP + UML do projeto: ProjetoArduinoJava.zip

5. Protótipo Visual - App (por Suzanny)

A interface visual do projeto foi idealizada como um aplicativo de casa inteligente. O protótipo representa um painel de monitoramento em tempo real, mostrando os seguintes dados:

- Temperatura (em °C)
- Umidade (%)
- Luminosidade (lux)
- Lista de alertas recentes



6. Conclusão

O projeto permitiu a aplicação prática dos conhecimentos em eletrônica, programação embarcada e desenvolvimento de software. A integração entre hardware e software trouxe uma experiência completa de construção de um sistema funcional e escalável para monitoramento.

7. Links Arduino e Figma (Protótipo)

Arduino:

<https://www.tinkercad.com/things/30Y6FYgteru-projeto-integrador/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard&sharecode=PKsSWT1jmwKMNOpGZ1y7XB3uVo3k5WHIFrQumZRI-RQ>

Figma (Protótipo) :

<https://www.figma.com/proto/MDNkqQgzMCm3SDjpem0MvE/Untitled?node-id=1-2&p=f&t=pyjZxMX22qilQrSZ-1&scaling=scale-down&content-scaling=fixed&page-id=0%3A1>