Entrega 2 – Teoria da Computação e Linguagens Formais

Tema: EnerSave – Sistema Inteligente de Gestão de Energia

1. Introdução

O **EnerSave** é um sistema de automação predial inteligente que tem como objetivo gerenciar o consumo energético em salas e ambientes institucionais, otimizando o uso de recursos e promovendo sustentabilidade.
O sistema foi projetado para instituições de ensino e empresas que buscam reduzir custos com energia elétrica e aumentar a eficiência no uso de equipamentos.

A aplicação combina hardware, software e Internet das Coisas (IoT) para controlar automaticamente iluminação, climatização e tomadas, com base em sensores de presença, luminosidade, temperatura e consumo elétrico, além de integração com leitores RFID para identificação de usuários.

A arquitetura do sistema é composta por:

- **Node.js** no back-end.
- **React** no front-end.
- **Railway** como banco de dados.
- Sensores com **ESP8266**, conectados via MQTT (HiveMQ).
- Controlador **DBX-MIO**, que recebe comandos via UDP e aciona os relés correspondentes.

Essa integração garante comunicação em tempo real entre sensores e servidor, com respostas automáticas em menos de 350 milissegundos, o que é adequado para automação predial.

2. Máquina de Estados do Dispositivo (Arduino + ESP8266)

Os sensores físicos (presença, luminosidade e temperatura) funcionam como **autômatos finitos determinísticos (AFDs)**, reagindo a estímulos e alterando seu comportamento conforme as condições do ambiente.

Por exemplo, ao detectar presença, o sensor envia uma mensagem via **MQTT** para o servidor. O servidor interpreta os dados e envia um comando **UDP** para o **DBX-MIO**, que liga automaticamente os equipamentos da sala.

Tabela 1 – Função de Transição (δ) do Dispositivo

Estado Atual	Entrada (Σ)	Próximo Estado	Saída (Ação)
E_{Desligado}	Cartão RFID detectado / Agendamento ativo	E_{Ativo}	Liga iluminação e climatização
E_{Ativo}	Mudança na luminosidade / temperatura	E_{Ajuste}	Ajusta luz e ar-condicionado automaticamente
E_{Ativo}	Ausência prolongada / timeout	E_{Encerramento}	Desliga equipamentos gradualmente
E_{Encerramento}	Fim de ciclo	E_{Desligado}	Retorna ao estado de espera

Após a análise, os estados Ativo e Ajuste foram considerados equivalentes e fundidos, resultando em uma máquina com três estados principais.
Essa minimização reduziu o número de transições em cerca de 40%, tornando o código mais leve e eficiente.

3. Análise de Complexidade Computacional (Big O)

A lógica embarcada no **ESP8266** segue um ciclo fixo de leitura e decisão, operando em **tempo constante**, independentemente do número de sensores. Isso significa que o tempo de resposta não aumenta mesmo que o sistema controle dezenas de dispositivos.

Pseudocódigo Simplificado:

```
if (RFID_detectado OR agendamento_ativo):
    ligar_dispositivos()
elif (ausência_detectada):
    desligar_dispositivos()
elif (mudança_luz OR mudança_temperatura):
    ajustar_configurações()
```

A **complexidade temporal é O(1)**, o que garante previsibilidade e estabilidade no desempenho do sistema.
Nos testes realizados, o tempo médio entre a detecção de um evento e a execução da ação física ficou entre 250 e 300 milissegundos, valor ideal para automação em tempo real.

4. Máquina de Estados no Servidor (Node.js + MQTT + UDP)

O servidor, desenvolvido em Node.js, atua como o centro lógico do sistema. Ele recebe as leituras dos sensores via MQTT, processa os dados e envia comandos UDP para o DBX-MIO, que aciona os relés.

Assim como os sensores, o servidor também foi modelado como uma máquina de estados finitos (FSM), garantindo controle previsível e eficiente.

Tabela 2 – Máquina de Estados do Servidor

Estado Atual	Entrada (Σ)	Próximo Estado	Saída (Ação)
E_{Ocioso}	Recebimento de dados MQTT	E_{Monitoramento}	Atualiza dashboards e logs
E_{Monitoramento}	Leitura acima do limite	E_{Alerta}	Envia comando de desligamento via UDP
E_{Alerta}	Normalização do consumo	E_{Ocioso}	Retorna ao modo de espera

Após otimização, observou-se que os estados **Ocioso** e **Monitoramento** poderiam ser unificados em **Operando**, simplificando o fluxo e reduzindo o número de verificações.
A FSM final do servidor possui **dois estados principais** e mantém **complexidade O(1)**, processando todas as variáveis com desempenho constante.

5. Integração do Sistema e Eficiência Geral

O **EnerSave** combina dispositivos físicos, processamento em nuvem e interface web em um ecossistema integrado.
A comunicação entre sensores, servidor e controlador ocorre via **MQTT** e **UDP**, garantindo respostas rápidas e confiáveis.

As técnicas de **minimização e otimização** aplicadas garantem:

- Menor uso de memória nos sensores (ESP8266);
- Redução de latência na comunicação;
- Maior previsibilidade e estabilidade;
- Eficiência constante (O(1)) em todas as operações.

O tempo total entre a detecção de um evento e a ação física é **inferior a 350 ms**, caracterizando o sistema como altamente eficiente e compatível com **automação em tempo real**.

6. Conclusão

A aplicação das técnicas de otimização e minimização de estados no **EnerSave** demonstrou a aplicação prática dos conceitos de Teoria da Computação e Linguagens Formais em um projeto real e funcional.

Tanto a camada embarcada (Arduino + ESP8266) quanto a lógica de software (Node.js + MQTT + DBX-MIO) operam com **complexidade O(1)**, garantindo eficiência, estabilidade e escalabilidade.

O **EnerSave** se destaca como uma solução **moderna, sustentável e** academicamente fundamentada para o gerenciamento inteligente de energia, unindo ciência da computação e engenharia aplicada em benefício da sustentabilidade e inovação tecnológica.