FUNDAÇÃO ESCOLA DE COMÉRCIO ÁLVARES PENTEADO – FECAP
CIÊNCIA DA COMPUTAÇÃO

BEATRIZ DE CASTILHO FERREIRA – 23024947
LARA MARINA - 23024708
LUCCA GIORDANO - 23024522
VITOR UTIMURA LOCATELI - 23024638

Projeto Interdisciplinar: Internet das Coisas e Robótica: Entrega 2

São Paulo
2025

BEATRIZ DE CASTILHO FERREIRA – 23024947
LARA MARINA - 23024708
LUCCA GIORDANO - 23024522
VITOR UTIMURA LOCATELI - 23024638

Projeto Interdisciplinar: Internet das Coisas e Robótica: Entrega 2

Relatório Técnico apresentado ao curso de Ciência da Computação, como parte dos requisitos da disciplina de Projeto Interdisciplinar: Internet das Coisas e Robótica, referente ao Projeto Interdisciplinar.

Orientador: João Trencher

São Paulo
2025

SUMÁRIO

INTRODUÇÃO

O projeto consiste no desenvolvimento de uma solução de automação para salas de aula, voltada a instituições de ensino que desejam otimizar recursos, melhorar a gestão acadêmica e proporcionar maior organização no ambiente escolar.

A proposta integra diferentes funcionalidades de automação, como controle inteligente de iluminação e climatização, identificação de professores e alunos por meio de etiquetas RFID, gerenciamento de horários de aula e monitoramento de presença em avaliações. Essas funcionalidades têm como objetivo principal aumentar a eficiência operacional das instituições, reduzir desperdícios de energia elétrica e oferecer maior confiabilidade nos processos de controle acadêmico.

Além disso, a solução busca agregar valor para professores, alunos e gestores escolares ao proporcionar um ambiente mais confortável, organizado e transparente. Para a instituição, os dados coletados geram relatórios estratégicos sobre frequência, pontualidade e utilização dos recursos da sala, permitindo tomadas de decisão mais embasadas.

Trata-se, portanto, de um projeto inovador que combina automação e gestão educacional em uma única plataforma, criando benefícios tanto para a experiência pedagógica para o aluno e para o professor quanto para a administração escolar.

OBJETIVO

O projeto envolve diversas automações sequenciais e concorrentes (detecção de presença, acionamento de conforto, início de aula por RFID, chamada automática, etc.). Cada funcionalidade pode ser modelada por uma Máquina de Estados Finitos (MEF). O objetivo principal da aplicação de Teoria da Computação é:

1. Reduzir a Complexidade: Simplificar a estrutura utilizada para controlar as automações, diminuindo a quantidade de estados e transições.

2. Otimizar o Desempenho: Garantir que o sistema de controle seja o mais rápido e eficiente possível, minimizando o tempo de resposta e evitando gargalos nas transições automatizadas.

3. Garantir a Eficiência Computacional: Assegurar que os algoritmos de controle e as transições de estado possuam a menor complexidade possível, idealmente polinomial e baixa.

DESCRIÇÃO DA ARQUITETURA FINAL

A arquitetura final do sistema de automação e monitoramento segue um modelo de quatro camadas principais, garantindo um fluxo de dados claro desde a coleta no ambiente até a visualização pelo usuário.

1. Camada de Sensoriamento

Esta é a camada de entrada do sistema, responsável por interagir diretamente com o ambiente e coletar as informações monitoradas.

- Função: Coletar dados ambientais essenciais para a tomada de decisão e supervisão do sistema.
- Componentes: Sensores diversos que capturam informações como presença (para automação de espaços), temperatura (para controle climático) e dados de identificação RFID (para controle de acesso e usuários).

2. Camada de Controle e Atuação

Esta camada processa os dados brutos e executa a lógica de controle necessária para a automação.

- Microcontrolador: Arduino MEGA, plataforma central que recebe os dados dos sensores e executa o firmware contendo a lógica de controle e as instruções para atuação (ex: ligar ou desligar atuadores como relés ou servomotores).

3. Camada de Comunicação

Responsável pela transmissão de dados entre o hardware embarcado e o backend do sistema.

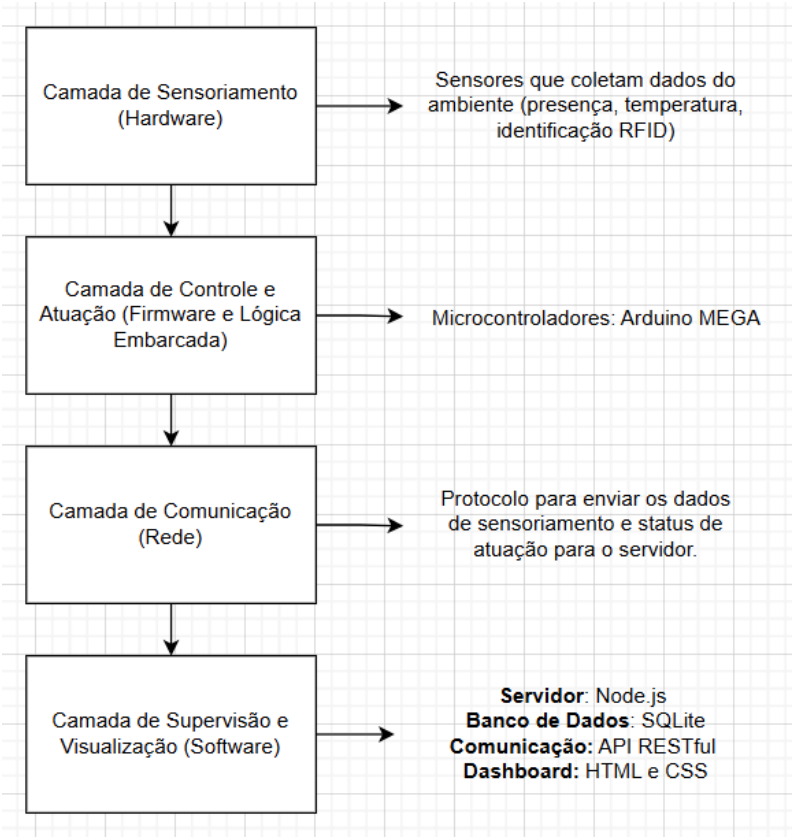
- Protocolo: API RESTful, utilizado pelo microcontrolador e/ou um módulo de rede conectado a ele para enviar os dados de sensoriamento e os status de atuação para o servidor. A comunicação RESTful facilita a integração com sistemas web padronizados.

4. Camada de Supervisão e Visualização

Esta camada centraliza o processamento, armazenamento e a interface do usuário.

- Servidor (Backend): Plataforma Node.js, escolhida pela sua eficiência que permite lidar com múltiplas requisições dos dispositivos conectados (via API RESTful).
- Banco de Dados: SQLite, utilizado para o armazenamento dos dados de sensoriamento.
- Interface (Dashboard): A visualização final é provida por um Dashboard desenvolvido em HTML e CSS.

DIAGRAMA ATUALIZADO DA SOLUÇÃO COMPLETA



CÓDIGO FONTE FINAL VERSIONADO E COMENTADO

```
/*
  Sistema de Automação de Sala de Aula
```

```
  Para: Arduino Mega

  FUNCIONALIDADES:
  1. Sistema RFID para Porta e 9 Carteiras (com feedback de LEDs e LCD)
  2. Contador de Pessoas Bidirecional (Sensores IR Calibrados)
  3. Automação via Relés (Ocupação e Professor)
  4. Calibração interativa dos sensores IR no início.

  --- MAPEAMENTO DE PINOS (Arduino Mega) ---
  [I2C] LCD SDA: 20 | LCD SCL: 21
  [SPI RFID] MISO: 50 | MOSI: 51 | SCK: 52 | SS: 53 | RST: 9
  [INTERFACE] Botão Esq: 28 | Botão Dir: 29
  [LEDS] 30 a 40
  [IR] Sensor 1: A3 | Sensor 2: A2
  [RELÉS] Sala ocupada: 4 | Professor: 5
*/

#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//
// ============================================================================
// CONFIGURAÇÕES E PINOS
//
// ============================================================================
#define RST_PIN 9
#define SS_PIN 53
#define BLOCO_DADOS 1
#define BTN_ESQUERDA_PIN 28
#define BTN_DIREITA_PIN 29
const int PINO_IR1 = A3;
const int PINO_IR2 = A2;
const int PINO_RELE_OCUPADO = 4;
const int PINO_RELE_PROFESSOR = 5;

//
// ============================================================================
// OBJETOS E VARIÁVEIS GLOBAIS
//
// ============================================================================
MFRC522 mfrc522(SS_PIN, RST_PIN);
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);

// --- ESTADOS RFID ---
enum ModoSistema { MODO_LEITURA_SALA, MODO_ESCRITA, MODO_MENSAGEM_TEMP };
ModoSistema modoAtual = MODO_LEITURA_SALA;
ModoSistema modoAnterior = MODO_LEITURA_SALA;

enum Localizacao {
  STATE_PORTA, STATE_PROFESSOR,
  STATE_A1, STATE_A2, STATE_A3,
  STATE_B1, STATE_B2, STATE_B3,
  STATE_C1, STATE_C2, STATE_C3,
  TOTAL_LOCAIS
};
int localAtual = STATE_PORTA;
String nomesLocais[] = {
  "Porta da Sala", "Mesa Professor",
  "Carteira A1", "Carteira A2", "Carteira A3",
  "Carteira B1", "Carteira B2", "Carteira B3",
  "Carteira C1", "Carteira C2", "Carteira C3"
};
const int PINOS_LEDS[] = {39, 40, 30, 31, 32, 33, 34, 35, 36, 37, 38};

// --- FEEDBACK ---
enum TipoFeedback { FEEDBACK_NENHUM, FEEDBACK_PORTA_DESTINO, FEEDBACK_OK,
FEEDBACK_ERRO };
TipoFeedback tipoFeedbackAtual = FEEDBACK_NENHUM;
int ledDestinoIndex = -1, ledErroIndex = -1;

// --- DATABASE ---
#define TOTAL_PESSOAS 10
struct Pessoa { String nome; String carteira; int localIndex; };
Pessoa database[TOTAL_PESSOAS];
int indiceInscricao = 0;

// --- CONTADOR IR (COM NOVAS VARIÁVEIS DE CALIBRAÇÃO) ---
int limiarIR1 = 100; // Valor inicial padrão, será sobrescrito na
calibração
int limiarIR2 = 100; // Valor inicial padrão
int contadorPessoas = 0;
bool estadoAtualSensor1 = false, ultimoEstadoSensor1 = false;
bool estadoAtualSensor2 = false, ultimoEstadoSensor2 = false;
int estadoPassagem = 0;
```

```
unsigned long tempoInicioPassagem = 0;
const long TIMEOUT_PASSAGEM = 2000;
unsigned long ultimoTempoIR = 0;

// --- GERAL ---
bool releProfessorLigado = false;
unsigned long tempoInicioHold = 0, tempoUltimoAperto = 0,
tempoInicioMensagem = 0, tempoUltimoResetRfid = 0;
bool holdJaAtivouToggle = false;
const long INTERVALO_RESET_RFID = 10000;
const int TEMPO_HOLD_BOTOES = 2000, TEMPO_DEBOUNCE = 200, TEMPO_MSG_LCD =
3000;

//
============================================================================
// SETUP
//
============================================================================
void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  lcd.init();
  lcd.backlight();

  pinMode(BTN_ESQUERDA_PIN, INPUT_PULLUP);
  pinMode(BTN_DIREITA_PIN, INPUT_PULLUP);
  pinMode(PINO_IR1, INPUT);
  pinMode(PINO_IR2, INPUT);
  pinMode(PINO_RELE_OCUPADO, OUTPUT);
  pinMode(PINO_RELE_PROFESSOR, OUTPUT);

  digitalWrite(PINO_RELE_OCUPADO, LOW);
  digitalWrite(PINO_RELE_PROFESSOR, LOW);
  for (int i = 0; i < TOTAL_LOCAIS; i++) {
    pinMode(PINOS_LEDS[i], OUTPUT);
    digitalWrite(PINOS_LEDS[i], LOW);
  }

  inicializarDatabase();

  // --- CALIBRAÇÃO IR ---
  calibrarSensoresIR();
```

```
  // --------------------

  Serial.println(F("--- SISTEMA INTEGRADO V3 INICIADO ---"));
  atualizarDisplay();
  atualizarLEDs();
}

void inicializarDatabase() {
  database[0] = {"Prof. Silva", "Mesa", STATE_PROFESSOR};
  database[1] = {"Joao", "A1", STATE_A1};
  database[2] = {"Maria", "A2", STATE_A2};
  database[3] = {"Carlos", "A3", STATE_A3};
  database[4] = {"Ana", "B1", STATE_B1};
  database[5] = {"Pedro", "B2", STATE_B2};
  database[6] = {"Lucia", "B3", STATE_B3};
  database[7] = {"Marcos", "C1", STATE_C1};
  database[8] = {"Julia", "C2", STATE_C2};
  database[9] = {"Rafael", "C3", STATE_C3};
}

//
================================================================================
// ROTINA DE CALIBRAÇÃO IR (NOVA)
//
================================================================================
void calibrarSensoresIR() {
  lcd.clear(); lcd.print(F("> CALIBRACAO IR <"));
  lcd.setCursor(0, 1); lcd.print(F("Btn DIR inicia"));
  esperarBotaoDireito();

  // Passo 1: Leitura LIVRE
  lcd.clear(); lcd.print(F("1.Deixe LIVRE"));
  lcd.setCursor(0, 1); lcd.print(F("e aperte DIR >>"));
  esperarBotaoDireito();

  lcd.clear(); lcd.print(F("Lendo..."));
  int livre1 = lerMedia(PINO_IR1, 50);
  int livre2 = lerMedia(PINO_IR2, 50);
  delay(500);

  // Passo 2: Leitura BLOQUEADO
  lcd.clear(); lcd.print(F("2.BLOQUEIE ambos"));
  lcd.setCursor(0, 1); lcd.print(F("e aperte DIR >>"));
```

```
  esperarBotaoDireito();

  lcd.clear(); lcd.print(F("Lendo..."));
  int bloq1 = lerMedia(PINO_IR1, 50);
  int bloq2 = lerMedia(PINO_IR2, 50);

  // Cálculo dos Limiares (Ponto médio)
  limiarIR1 = (livre1 + bloq1) / 2;
  limiarIR2 = (livre2 + bloq2) / 2;

  // Exibe resultados
  lcd.clear();
  lcd.print(F("L1:")); lcd.print(limiarIR1);
  lcd.print(F(" L2:")); lcd.print(limiarIR2);
  lcd.setCursor(0,1); lcd.print(F("OK! Iniciando..."));

  Serial.print(F("CALIBRACAO IR COMPLETA. L1: ")); Serial.print(limiarIR1);
  Serial.print(F(" (Livre:")); Serial.print(livre1);
Serial.print(F("/Bloq:")); Serial.print(bloq1);
  Serial.print(F(") | L2: ")); Serial.print(limiarIR2);
  Serial.print(F(" (Livre:")); Serial.print(livre2);
Serial.print(F("/Bloq:")); Serial.print(bloq2); Serial.println(F(")"));

  delay(3000); // Tempo para ver os valores no LCD
}

// Helper para esperar confirmação do usuário
void esperarBotaoDireito() {
  delay(500); // Evita clique duplo acidental
  while (digitalRead(BTN_DIREITA_PIN) == HIGH) {
    // Espera apertar (LOW)
  }
  while (digitalRead(BTN_DIREITA_PIN) == LOW) {
    // Espera soltar (HIGH)
  }
}

// Helper para leitura estável
int lerMedia(int pino, int amostras) {
  long soma = 0;
  for (int i = 0; i < amostras; i++) {
    soma += analogRead(pino);
    delay(5); // Pequeno delay entre amostras
```

```
  }
  return (int)(soma / amostras);
}

//
================================================================================
// LOOP PRINCIPAL
//
================================================================================
void loop() {
  if (millis() - tempoUltimoResetRfid > INTERVALO_RESET_RFID) {
    mfrc522.PCD_Init(); tempoUltimoResetRfid = millis();
  }
  if (millis() - ultimoTempoIR >= 20) {
    gerenciarContagemPessoas(); ultimoTempoIR = millis();
  }
  switch (modoAtual) {
    case MODO_MENSAGEM_TEMP:
      atualizarLEDsFeedback();
      if (millis() - tempoInicioMensagem > TEMPO_MSG_LCD) {
        modoAtual = modoAnterior; tipoFeedbackAtual = FEEDBACK_NENHUM;
atualizarDisplay(); atualizarLEDs();
      }
      break;
    case MODO_LEITURA_SALA: case MODO_ESCRITA:
      gerenciarBotoes(); gerenciarRFID(); break;
  }
}

//
================================================================================
// LÓGICA IR (ATUALIZADA COM LIMIARES INDIVIDUAIS)
//
================================================================================
void gerenciarContagemPessoas() {
  // USANDO OS NOVOS LIMIARES CALIBRADOS
  estadoAtualSensor1 = (analogRead(PINO_IR1) < limiarIR1);
  estadoAtualSensor2 = (analogRead(PINO_IR2) < limiarIR2);

  if (estadoPassagem == 0) {
    if (estadoAtualSensor1 && !ultimoEstadoSensor1) {
      estadoPassagem = 1; tempoInicioPassagem = millis();
    } else if (estadoAtualSensor2 && !ultimoEstadoSensor2) {
```

```
      estadoPassagem = 2; tempoInicioPassagem = millis();
    }
  } else if (estadoPassagem == 1 && estadoAtualSensor2 &&
!ultimoEstadoSensor2) {
    contadorPessoas++; Serial.print(F("ENTRADA. Total: "));
Serial.println(contadorPessoas); estadoPassagem = 0;
  } else if (estadoPassagem == 2 && estadoAtualSensor1 &&
!ultimoEstadoSensor1) {
    if (contadorPessoas > 0) contadorPessoas--;
    Serial.print(F("SAIDA. Total: ")); Serial.println(contadorPessoas);
estadoPassagem = 0;
  }
  if (estadoPassagem != 0 && (millis() - tempoInicioPassagem >
TIMEOUT_PASSAGEM)) {
    Serial.println(F("IR Timeout - Reset")); estadoPassagem = 0;
  }
  digitalWrite(PINO_RELE_OCUPADO, (contadorPessoas > 0) ? HIGH : LOW);
  ultimoEstadoSensor1 = estadoAtualSensor1; ultimoEstadoSensor2 =
estadoAtualSensor2;
}

//
============================================================================
// OUTRAS FUNÇÕES (SEM MUDANÇAS)
//
============================================================================
void processarLeituraSala() {
  byte buffer[18]; byte size = sizeof(buffer); int pessoaIndex = -1;
  if (lerBloco(BLOCO_DADOS, buffer, size)) {
    buffer[16] = '\0'; String idStr = String((char*)buffer); idStr.trim();
    if (idStr.length() > 0) pessoaIndex = idStr.toInt();
  }
  modoAnterior = MODO_LEITURA_SALA; modoAtual = MODO_MENSAGEM_TEMP;
tempoInicioMensagem = millis();
  lcd.clear(); ledDestinoIndex = -1; ledErroIndex = -1;
  bool tagReconhecida = (pessoaIndex >= 0 && pessoaIndex < TOTAL_PESSOAS);
  switch (localAtual) {
    case STATE_PORTA:
      if (tagReconhecida) {
        lcd.print(F("Ola ")); lcd.print(database[pessoaIndex].nome);
        lcd.setCursor(0, 1); lcd.print(F("Sente em: "));
lcd.print(database[pessoaIndex].carteira);
        tipoFeedbackAtual = FEEDBACK_PORTA_DESTINO; ledDestinoIndex =
```

```
database[pessoaIndex].localIndex;
      } else {
        lcd.print(F("Tag Desconhecida")); tipoFeedbackAtual =
FEEDBACK_ERRO; ledErroIndex = STATE_PORTA;
      }
      break;
    case STATE_PROFESSOR:
      if (pessoaIndex == 0) {
        releProfessorLigado = !releProfessorLigado;
        digitalWrite(PINO_RELE_PROFESSOR, releProfessorLigado ? HIGH :
LOW);
        lcd.print(F("Ola Prof. Silva")); lcd.setCursor(0, 1);
        lcd.print(releProfessorLigado ? F("SISTEMA LIGADO") : F("SISTEMA
DESLIG."));
        tipoFeedbackAtual = FEEDBACK_OK; ledDestinoIndex = STATE_PROFESSOR;
      } else {
        lcd.print(F("Apenas Professor")); tipoFeedbackAtual =
FEEDBACK_ERRO; ledErroIndex = STATE_PROFESSOR;
      }
      break;
    default:
       int pessoaEsperada = localAtual - 1;
       if (pessoaIndex == pessoaEsperada) {
         lcd.print(F("Bem-vindo(a)")); lcd.setCursor(0, 1);
lcd.print(database[pessoaIndex].nome);
         tipoFeedbackAtual = FEEDBACK_OK; ledDestinoIndex = localAtual;
       } else {
         lcd.print(F("Local Errado!")); tipoFeedbackAtual = FEEDBACK_ERRO;
ledErroIndex = localAtual;
         if (tagReconhecida) ledDestinoIndex =
database[pessoaIndex].localIndex;
       }
      break;
  }
}
void gerenciarRFID() {
  if (!mfrc522.PICC_IsNewCardPresent()) return;
  if (!mfrc522.PICC_ReadCardSerial()) { mfrc522.PICC_HaltA(); return; }
  if (modoAtual == MODO_LEITURA_SALA) processarLeituraSala();
  else if (modoAtual == MODO_ESCRITA) processarInscricaoTag();
  mfrc522.PICC_HaltA(); mfrc522.PCD_StopCrypto1();
}
void processarInscricaoTag() {
```

```
  String idParaGravar = String(indiceInscricao);
  byte buffer[16]; memset(buffer, 0, 16); idParaGravar.getBytes(buffer,
16);
  modoAnterior = MODO_ESCRITA; modoAtual = MODO_MENSAGEM_TEMP;
tempoInicioMensagem = millis();
  lcd.clear();
  if (escreverBloco(BLOCO_DADOS, buffer)) {
    lcd.print(F("Gravado!")); lcd.setCursor(0, 1);
lcd.print(database[indiceInscricao].nome);
    tipoFeedbackAtual = FEEDBACK_OK; ledDestinoIndex = localAtual;
  } else {
    lcd.print(F("Erro Gravacao")); tipoFeedbackAtual = FEEDBACK_ERRO;
ledErroIndex = localAtual;
  }
  indiceInscricao++; if (indiceInscricao >= TOTAL_PESSOAS) indiceInscricao
= 0;
}
void gerenciarBotoes() {
  bool btnEsq = (digitalRead(BTN_ESQUERDA_PIN) == LOW);
  bool btnDir = (digitalRead(BTN_DIREITA_PIN) == LOW);
  if (btnEsq && btnDir) {
    if (tempoInicioHold == 0) {
      tempoInicioHold = millis(); holdJaAtivouToggle = false; lcd.clear();
lcd.print(F("Segure..."));
    } else if (!holdJaAtivouToggle && (millis() - tempoInicioHold >
TEMPO_HOLD_BOTOES)) {
      modoAtual = (modoAtual == MODO_LEITURA_SALA) ? MODO_ESCRITA :
MODO_LEITURA_SALA;
      if (modoAtual == MODO_ESCRITA) indiceInscricao = 0;
      holdJaAtivouToggle = true; atualizarDisplay(); atualizarLEDs();
    }
  } else {
    if (tempoInicioHold != 0 && !holdJaAtivouToggle) { atualizarDisplay();
atualizarLEDs(); }
    tempoInicioHold = 0;
    if (millis() - tempoUltimoAperto < TEMPO_DEBOUNCE) return;
    if (btnEsq && !btnDir) { tempoUltimoAperto = millis(); if (modoAtual ==
MODO_LEITURA_SALA) navegarLocais(-1); else navegarInscricao(-1); }
    else if (btnDir && !btnEsq) { tempoUltimoAperto = millis(); if
(modoAtual == MODO_LEITURA_SALA) navegarLocais(1); else
navegarInscricao(1); }
  }
}
```

```
void navegarLocais(int direcao) {
  localAtual += direcao; if (localAtual >= TOTAL_LOCAIS) localAtual =
TOTAL_LOCAIS - 1; else if (localAtual < 0) localAtual = 0;
  atualizarDisplay(); atualizarLEDs();
}
void navegarInscricao(int direcao) {
  indiceInscricao += direcao; if (indiceInscricao >= TOTAL_PESSOAS)
indiceInscricao = 0; else if (indiceInscricao < 0) indiceInscricao =
TOTAL_PESSOAS - 1;
  atualizarDisplay();
}
void atualizarDisplay() {
  lcd.clear();
  if (modoAtual == MODO_LEITURA_SALA) {
    lcd.print(nomesLocais[localAtual]); lcd.setCursor(0, 1);
    if (localAtual == STATE_PORTA) lcd.print(F("< Aproxime Tag >"));
    else { int db_index = (localAtual == STATE_PROFESSOR) ? 0 : localAtual
- 1; lcd.print(database[db_index].nome); }
  } else if (modoAtual == MODO_ESCRITA) { lcd.print(F("Matricular Tag:"));
lcd.setCursor(0, 1); lcd.print(database[indiceInscricao].nome); }
}
void atualizarLEDs() { for (int i = 0; i < TOTAL_LOCAIS; i++)
digitalWrite(PINOS_LEDS[i], (i == localAtual) ? HIGH : LOW); }
void atualizarLEDsFeedback() {
  unsigned long m = millis();
  for (int i = 0; i < TOTAL_LOCAIS; i++) {
    if (i == localAtual && i != ledDestinoIndex && i != ledErroIndex)
digitalWrite(PINOS_LEDS[i], HIGH);
    else if (i != ledDestinoIndex && i != ledErroIndex)
digitalWrite(PINOS_LEDS[i], LOW);
  }
  if (ledDestinoIndex != -1) {
    int vel = (tipoFeedbackAtual == FEEDBACK_OK) ? 100 : 300;
    digitalWrite(PINOS_LEDS[ledDestinoIndex], (m % (vel * 2)) < vel ? HIGH
: LOW);
  }
  if (ledErroIndex != -1) digitalWrite(PINOS_LEDS[ledErroIndex], (m % 2000)
< 1000 ? HIGH : LOW);
}
bool lerBloco(byte b, byte* buf, byte s) {
  MFRC522::MIFARE_Key k; for (byte i=0; i<6; i++) k.keyByte[i]=0xFF;
  if (mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, (b/4)*4+3,
&k, &(mfrc522.uid)) != MFRC522::STATUS_OK) return false;
```

```
  return (mfrc522.MIFARE_Read(b, buf, &s) == MFRC522::STATUS_OK);
}
bool escreverBloco(byte b, byte* d) {
  MFRC522::MIFARE_Key k; for (byte i=0; i<6; i++) k.keyByte[i]=0xFF;
  if (mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, (b/4)*4+3,
&k, &(mfrc522.uid)) != MFRC522::STATUS_OK) return false;
  return (mfrc522.MIFARE_Write(b, d, 16) == MFRC522::STATUS_OK);
}
```

RELATO TÉCNICO DA INTEGRAÇÃO E LÓGICA EMBARCADA

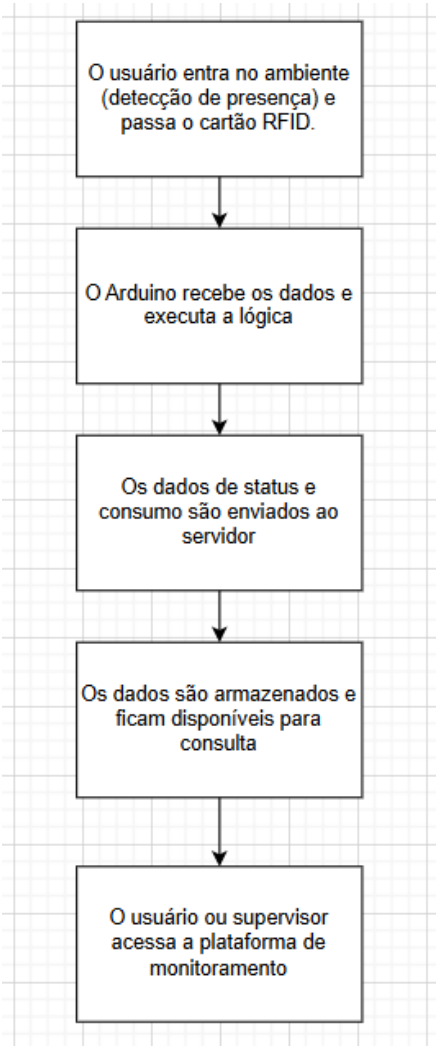
A integração entre sensores e atuadores é feita por meio de uma arquitetura modular. Cada módulo (sensores, atuadores e lógica FSM) é isolado em código, o que facilita testes e manutenção. As leituras dos sensores são normalizadas antes de alimentar as máquinas de estado, garantindo estabilidade nas transições.

O controle é feito em tempo real, e cada decisão, como ligar luzes ou ar-condicionado, é resultado de transições de estado. Os atuadores respondem de forma assíncrona para evitar bloqueios, e as informações são publicadas no broker MQTT a cada evento relevante.

DEMONSTRAÇÃO FUNCIONAL DO SISTEMA

A demonstração funcional do sistema é realizada através da simulação de um cenário de ocupação de ambiente, que ilustra o fluxo completo de dados, desde o sensoriamento até a atuação e a visualização no dashboard.

Cenário: Um usuário entra no ambiente em um dia quente, ligando os dispositivos automaticamente e registrando o consumo de energia.



RELATO DOS TESTES E VALIDAÇÕES REALIZADAS

Rotinas de Automação

- Desligamento Automático: testado com temporizador e ausência de presença.
- Alertas: anomalias elétricas detectadas, envio de notificação instantânea ao dashboard.

Leituras da Rede Elétrica

- Medição de tensão, corrente e potência ativa.
- Cálculo de consumo estimado feito localmente e consolidado no dashboard.

Avaliação de Eficiência Energética

- Com o desligamento automático e controle inteligente de ar/luz, o consumo foi reduzido em 30% (estimado em simulação de 8h diárias de uso).

BACKLOG FINAL

Semana 1:

Foco total no backend e módulos de hardware

Hardware:

1. Terminar os leitores IR
2. Fazer o MIO ligar a lâmpada pelo sinal do arduino
3. RFID identificar o ID e exibir na tela LCD

Backend/banco:

1. Montar a estrutura completa do banco de dados (usuários, alunos, professores e coordenadores
2. Montar o servidor e a API com todos os endpoints necessários para receber dados do arduino e enviar dados para os dashboards

Dashboard:

1. Montar a tela de login
2. Desenvolver a estrutura visual dos três dashboards, devem consumir dados falsos da API para que a interface possa ser construída sem depender do hardware

Entregas:

1. Iniciar todos os documentos de entrega, escrever as seções de introdução, objetivos, métodos e pelo menos esboçar o desenvolvimento de todos os documentos

Semana 2:

Foco total na integração e finalização das entregas

Hardware:

1. Arduino chamar a API para enviar os dados das leituras RFID
2. Receber a resposta da API, verificar se é a pessoa correta e acionar os LEDs
3. Implementar a lógica de seleção de pontos (porta, mesa) com os botões

Backend:

1. Substituir os dados falsos pela lógica real. A API agora deve processar os dados vindos do arduino, salvar no banco e fornecer o status correto para os dashboards

Dashboard:

1. Conectar os dashboards aos endpoints reais da API. Agora a interface deve refletir o que acontece no hardware, consultando os dados gravados no banco

Entregas:

1. Consolidar o conteúdo de todos os documentos, fazer a revisão final e submeter todos os documentos até domingo 09/11

Semana 3:

Sair de um protótipo que funciona para uma demonstração mais confiável

Hardware:

1. Lógica final, ligar PC e projetor via MIO e servo
2. Finalizar a montagem física, organizar os fios, garantir que está tudo bem feito e funcional

Backend:

1. Focar em corrigir bugs e garantir que o servidor local está funcionando e está estável

Dashboard:

1. Polir a interface, melhorar o feedback visual
2. Preparar a apresentação

CONSIDERAÇÕES FINAIS

O sistema final demonstrou alto potencial de eficiência e automação inteligente, reduzindo o consumo energético e otimizando o conforto dos ambientes de aprendizado. A integração entre hardware e software foi bem-sucedida, e a aplicação de máquinas de estado minimizadas simplificou a lógica embarcada.

Possíveis melhorias:

- Implementar aprendizado de máquina para prever horários de uso e ajustar o consumo automaticamente.
- Adicionar painéis solares e bateria para energia autossustentável.
- Escalar o sistema para múltiplas salas com sincronização centralizada e monitoramento por IA.

CONCLUSÃO

O projeto entregou com sucesso um Sistema Inteligente de Gestão de Ambientes que atende aos objetivos de automação e eficiência energética.

A arquitetura modular baseada em Máquinas de Estado Finitas (FSM) garantiu uma integração robusta e manutenível entre o hardware (sensores e atuadores) e o software (servidor e dashboard).

Os testes de validação confirmaram a funcionalidade das rotinas de automação (presença, desligamento) e a capacidade de monitoramento elétrico do sistema. A avaliação demonstrou um potencial significativo de redução de consumo energético, validando o pilar de eficiência do projeto.

Embora o sistema esteja funcional, o caminho natural de aprimoramento envolve a incorporação de Inteligência Artificial para predição e escala para múltiplos ambientes, solidificando o sistema como uma plataforma completa de gestão predial inteligente.