Estácio
Niterói

Painel de Chamadas de Pedido com Wi-Fi (Autoatendimento)

Vinicius do Valle Pinto
Rafael Matos
Emanuel Peçanha
Júlia Vassimon da Silva
Luan de Salles Pinheiro da Costa

Professor: Ralfh Ansusattigui

2025

Niterói - RJ

Sumário

1. DIAGNÓSTICO E TEORIZAÇÃO

1.1. Identificação das partes interessadas e parceiros

O projeto "Painel de Chamadas de Pedido com Wi-Fi (Autoatendimento)" tem como principal parte interessada o cliente final, que se beneficia da automação do processo de chamada de pedidos. Além disso, os estabelecimentos como

restaurantes, lanchonetes e cafés são parceiros cruciais, pois são o local de aplicação da solução. O projeto visa justificar sua pertinência social ao demonstrar como ele pode otimizar a operação desses negócios, melhorando a experiência do cliente

1.2. Problemática e/ou problemas identificados

A problemática principal que motiva este projeto é a ineficiência e a falta de automação na comunicação entre a cozinha e a área de atendimento em estabelecimentos de comida. A chamada manual de pedidos pode levar a erros, atrasos e desorganização, impactando a experiência do cliente e a produtividade do negócio. A solução busca resolver essa questão ao automatizar a notificação de pedidos prontos via comunicação Wi-Fi, tornando o processo mais rápido e preciso. Essa demanda foi identificada a partir da observação de problemas comuns no atendimento ao cliente, que exigem uma escuta da comunidade onde o projeto será desenvolvido.

1.3. Justificativa

Este projeto é academicamente pertinente porque a aprendizagem baseada em projetos envolve a produção e aplicação de conhecimento para resolver demandas reais, como a otimização de processos em restaurantes. A criação do "Painel de Chamadas de Pedido" permite que o grupo de trabalho aplique conhecimentos de Tecnologia da Informação, como a comunicação Wi-Fi, o uso de sensores e a programação de microcontroladores (ESP32). O projeto se alinha diretamente com objetivos de formação que envolvem a automação e o desenvolvimento de soluções tecnológicas práticas. A motivação do grupo é a de aplicar os conceitos teóricos aprendidos em sala de aula para construir uma solução funcional que atenda a uma necessidade social.

1.4. Objetivos/resultados/efeitos a serem alcançados (em relação ao problema identificado e sob a perspectiva dos públicos envolvidos)

O projeto busca alcançar os seguintes objetivos, de maneira clara e sucinta:

- **Automatizar a chamada de pedidos prontos:** Usar a comunicação Wi-Fi para notificar os clientes quando seus pedidos estiverem prontos, substituindo o método manual.
- **Melhorar a eficiência operacional:** Reduzir o tempo de espera e a desorganização na entrega dos pedidos, melhorando o fluxo de trabalho no estabelecimento.

- **Aprimorar a experiência do cliente:** Fornecer uma notificação clara e audível para o cliente, garantindo que ele seja alertado prontamente.

O sucesso do projeto será avaliado pela capacidade de o sistema automatizar a chamada de pedidos de forma eficiente, com base na perspectiva dos usuários envolvidos, ou seja, os clientes e o pessoal do estabelecimento.

1.5. Referencial teórico (subsídio teórico para propositura de ações da extensão)

fundamentação teórica do projeto se baseia na automação de processos por meio de **sistemas embarcados** e comunicação sem fio. O projeto utiliza o conceito de **Internet das Coisas (IoT)**, onde dispositivos se comunicam e trocam dados via internet para automatizar tarefas. O protocolo **MQTT** é central para a comunicação entre a "cozinha" e o "painel de atendimento", permitindo uma troca de mensagens leve e eficiente.

As obras e autores citados para justificar as ações do projeto devem abordar temas como:

2. **Comunicação de dados sem fio:** O uso de redes Wi-Fi e a aplicação do protocolo MQTT.
3. **Microcontroladores e sistemas embarcados:** O funcionamento e a programação de dispositivos como o ESP32.

Automação e eficiência de processos: A aplicação da tecnologia para otimizar fluxos de trabalho em ambientes de serviço.

2. PLANEJAMENTO E DESENVOLVIMENTO DO PROJETO

3.1. Plano de trabalho (usando ferramenta acordada com o docente)

Em sala de aula, juntamente com o professor, o tema do projeto foi escolhido e debatido para definir a melhor estratégia de desenvolvimento. Após discussões em grupo, decidiu-se que o simulador utilizado seria o **Wokwi**.

O grupo foi dividido conforme as etapas de entrega, de modo que cada integrante ficou responsável por uma parte específica. Entretanto, todos participam coletivamente da execução prática do projeto. A seguir, estão descritas as entregas e seus respectivos responsáveis:

1. **Primeira entrega:** diagnóstico e teorização — responsável: **Vinicius do Valle Pinto**.
2. **Segunda entrega:** planejamento e desenvolvimento do projeto — responsável: **Júlia Vassimon da Silva**.
3. **Terceira entrega:** apresentação de um MVP ou protótipo funcional acompanhado de relatório — responsável: **Rafael Matos**.

4. **Quarta entrega:** apresentação de um MVP ou protótipo funcional — responsáveis: **Emanuel Peçanha** e **Luan**.
5. **Quinta e sexta entregas:** elaboração dos relatos coletivos e individuais — realizadas em grupo.

Abaixo, apresenta-se um minicalendário com o resumo das datas e conteúdos previstos para cada etapa do projeto.

DATA:	CONTEÚDO
08/08/2025	Apresentação da disciplina
15/08/2025	Formação de grupos – discussões inicial dos recursos necessários para o desenvolvimento dos trabalhos
05/09/2025	Apresentação dos temas
19/09/2025	Diagnóstico e teorização
10/10/2025	Planejamento e desenvolvimento do projeto
31/10/2025	Apresentação de um MVP ou Protótipo Funcional/Relatório
07/11/2025	Apresentação de um MVP ou Protótipo Funcional
14/11/2025	Apresentação Pitch/Relato Coletivo e Individual
21/11/2025	Apresentação Pitch/Relato Coletivo e Individual

3.2. Descrição da forma de envolvimento do público participante na formulação do projeto, seu desenvolvimento e avaliação, bem como as estratégias pelo grupo para mobilizá-los.

A parceria com um restaurante local foi fundamental para o planejamento e desenvolvimento do projeto. Desde o início, o estabelecimento atuou como participante sociocomunitário ativo, contribuindo com informações sobre as demandas reais do ambiente de atendimento e oferecendo feedbacks práticos para o aprimoramento da solução tecnológica.

Durante os encontros realizados entre a equipe acadêmica e os representantes do restaurante, foram discutidos aspectos como o fluxo de atendimento atual e as principais dificuldades na comunicação entre a cozinha e o salão, a necessidade de reduzir o tempo de espera e erros de chamada de pedidos, o layout ideal do painel, a clareza das informações exibidas e a importância de um alerta sonoro acessível.

Essas trocas permitiram alinhar o desenvolvimento do sistema às necessidades reais do parceiro comunitário, tornando o processo de criação colaborativo e contextualizado. Com base nessas conversas, o grupo acadêmico ajustou o protótipo no simulador Wokwi, incorporando funções como o histórico de pedidos prontos e o botão de repetição do último chamado, sugestões diretas do restaurante parceiro.

Na etapa de avaliação, o restaurante participou de uma demonstração prática da solução simulada, contribuindo com observações sobre usabilidade, clareza das mensagens e aplicabilidade do sistema no dia a dia do estabelecimento. Essa interação reforçou a importância da escuta ativa e da construção conjunta, permitindo que o projeto unisse saberes técnicos e experiências comunitárias.

Durante todo o processo foram produzidos registros das reuniões, capturas de tela do simulador Wokwi e anotações de feedbacks, que evidenciam a troca mútua de conhecimento entre a equipe acadêmica e o parceiro local. Esses materiais servirão como base documental para futuras implementações e para a evolução do projeto em versões reais, fortalecendo a integração entre a instituição de ensino e a comunidade.

Abaixo conversa com o pai, dono do restaurante, de um dos integrantes do projeto:

[09/10/2025, 23:00:23] emanuel peçanha: Oi pai. Então, eu tô fazendo um projeto na faculdade com um grupos de amigos. É um painel de chamadas de pedido com Wi-Fi, tipo aqueles que mostram o número do pedido quando tá pronto. Eu pensei em testar isso no praça mix, na loja de pastel, que tem uma grande rotatividade de pessoas.

[09/10/2025, 23:00:26] emanuel peçanha: O que acha?

[09/10/2025, 23:00:46] Pai: Oi filho, me explica pra que isso serve exatamente

[09/10/2025, 23:01:08] emanuel peçanha: A ideia é bem simples: quando o pastel ficar pronto, ao invés de você ou alguém ter que gritar o número ou ficar chamando no balcão, a cozinha aperta um botão e o número aparece no painel. O cliente vê ou escuta o alerta e já vem buscar. Isso deixa o atendimento mais rápido, organizado e evita confusão, principalmente quando tem muita gente

[09/10/2025, 23:01:47] Pai: Poxa, que show, estava precisando disso mesmo. Lá quando enche vira uma bagunça

[09/10/2025, 23:02:19] emanuel peçanha: Sim, e com o painel ninguem precisa ficar repetindo o número do pedido ou se estressar.

[09/10/2025, 23:02:29] Pai: Gostei da ideia, se funcionar vai me ajudar muito

3.3. Grupo de trabalho (descrição da responsabilidade de cada membro)

O grupo foi organizado de forma colaborativa, com divisão de tarefas conforme as etapas do projeto, garantindo que todos os integrantes participassem tanto das atividades teóricas quanto práticas. A seguir, descrevem-se as responsabilidades e atividades atribuídas a cada membro:

• **Vinicius do Valle Pinto** – Responsável pela **primeira entrega**, envolvendo o diagnóstico inicial e a teorização do projeto. Coube a ele a pesquisa de fundamentos teóricos, contextualização do tema e elaboração da base conceitual que sustentará as demais etapas.

• **Júlia Vassimon da Silva** – Responsável pela **segunda entrega**, correspondente ao planejamento e desenvolvimento do projeto. Suas atividades incluem a organização do cronograma, definição das ações a serem executadas e acompanhamento da execução prática em conjunto com o grupo.

• **Rafael Matos** – Responsável pela **terceira entrega**, que compreende a apresentação de um **MVP ou protótipo funcional**, acompanhada do relatório técnico. Atua na documentação do processo e na validação dos resultados obtidos.

• **Emanuel Peçanha** – Responsável, junto a Luan, pela **quarta entrega**, focada na apresentação prática do **MVP ou protótipo funcional**. Sua principal função é contribuir para o desenvolvimento e testes do simulador no ambiente Wokwi, assegurando o funcionamento correto do projeto.

• **Luan de Salles Pinheiro da Costa** – Atua em parceria com Emanuel na **quarta entrega**, sendo corresponsável pela parte prática e pela demonstração funcional do protótipo. Também auxilia na integração entre as etapas de planejamento e execução.

• **Todos os integrantes** – Participam coletivamente da **quinta e sexta entregas**, que consistem na elaboração dos **relatos coletivos e individuais**, registrando o aprendizado adquirido, os desafios enfrentados e as contribuições de cada membro para o desenvolvimento do projeto. E também da parte do desenvolvimento do projeto na prática.

3.4. Metas, critérios ou indicadores de avaliação do projeto

O projeto de extensão “Painel de Chamadas de Pedido com Wi-Fi (Autoatendimento)” tem como objetivo principal desenvolver uma solução funcional que automatize a comunicação entre o setor de preparo e o painel de atendimento, proporcionando maior agilidade e organização nos pedidos.

Para alcançar esse objetivo, foram definidas **metas específicas** e **critérios de avaliação** que permitirão acompanhar o progresso e a efetividade das ações realizadas:

1. Metas:

- **Meta 1:** Realizar o diagnóstico e levantamento teórico sobre sistemas de autoatendimento e comunicação via Wi-Fi.
- **Meta 2:** Planejar e estruturar o funcionamento do painel de chamadas, definindo componentes, fluxos de informação e interface.
- **Meta 3:** Desenvolver e testar o protótipo no simulador Wokwi, assegurando a comunicação entre os módulos (cozinha e painel).

- **Meta 4:** Apresentar um MVP (Produto Mínimo Viável) funcional, demonstrando o envio e a exibição de chamadas de pedidos.
- **Meta 5:** Elaborar relatórios coletivos e individuais, registrando o processo de aprendizagem e os resultados obtidos.

2. Critérios de avaliação:

- Cumprimento dos prazos estabelecidos no cronograma.
- Participação ativa e colaboração de todos os integrantes do grupo nas etapas teóricas e práticas.
- Qualidade técnica do protótipo apresentado (funcionalidade, clareza do código e simulação correta).
- Clareza e organização dos relatórios de acompanhamento e apresentações.
- Aplicação dos conceitos estudados e coerência com os objetivos propostos.

3. Indicadores de efetividade:

- **Indicador 1:** Protótipo funcional testado com sucesso no simulador Wokwi.
- **Indicador 2:** Entregas realizadas dentro dos prazos previstos.
- **Indicador 3:** Relatórios e apresentações avaliados positivamente pelo professor orientador.
- **Indicador 4:** Demonstração da compreensão prática dos conteúdos de eletrônica e programação aplicados ao contexto do projeto.

Esses indicadores permitirão verificar não apenas o cumprimento das metas técnicas, mas também o desenvolvimento das competências individuais e coletivas, garantindo que o projeto alcance seus objetivos de forma efetiva e integrada ao processo de ensino-aprendizagem.

3.5. Recursos previstos

O projeto de extensão “Painel de Chamadas de Pedido com Wi-Fi (Autoatendimento)” será desenvolvido com o objetivo de criar uma solução acessível e funcional para automatizar a comunicação entre a cozinha e o painel de atendimento em estabelecimentos alimentícios, como restaurantes, lanchonetes e cafés.

Para o desenvolvimento do projeto, estão previstos os seguintes recursos:

1. Recursos materiais:

- Utilização do simulador Wokwi, que permite o desenvolvimento e teste de circuitos eletrônicos e códigos de forma virtual, sem necessidade de compra de componentes físicos.
- Equipamentos pessoais dos integrantes e equipamentos disponibilizados pelo campos da faculdade para o desenvolvimento e simulação do sistema.
- Materiais de apoio em sala de aula, como quadro branco e projetor para a organização do cronograma e do planejamento coletivo.

2. Recursos institucionais:

- Apoio do professor orientador, responsável por acompanhar as etapas do projeto, validar as entregas e oferecer suporte técnico e pedagógico.
- Utilização do ambiente escolar e de plataformas digitais para registro de reuniões, compartilhamento de arquivos e acompanhamento do progresso.

3. Recursos humanos:

- Equipe composta por cinco integrantes (Vinicius do Valle Pinto, Júlia Vassimon da Silva, Rafael Matos, Emanuel Peçanha e Luan), com divisão de responsabilidades conforme as etapas do projeto.
- Colaboração contínua entre os membros para execução das atividades teóricas e práticas.

O projeto foi planejado com foco em minimizar custos financeiros, utilizando recursos gratuitos e de fácil acesso, como o simulador Wokwi e ferramentas digitais disponíveis. Não há previsão de gastos diretos, e eventuais custos eventuais (como impressão de materiais ou uso de equipamentos pessoais) serão assumidos voluntariamente pelos participantes.

3.6. Detalhamento técnico do projeto

A solução desenvolvida consiste em um Painel de Chamadas de Pedido com comunicação via Wi-Fi, voltado para o uso em restaurantes, lanchonetes e café**s**. O sistema foi projetado para automatizar o processo de chamada de pedidos prontos, integrando duas placas ESP32 por meio do protocolo MQTT, simuladas no ambiente virtual Wokwi.
Durante o desenvolvimento, foram aplicados os conhecimentos adquiridos nas etapas anteriores do plano de ensino, abrangendo desde a configuração de rede Wi-Fi, até o uso de protocolos de comunicação IoT e interfaces de exibição de dados.

A arquitetura do sistema é composta por dois módulos principais:

- **Módulo Cozinha (Emissor):** responsável por enviar notificações de pedidos prontos. A interação pode ocorrer por meio de um botão físico ou via comunicação serial. Ao acionar o comando, o ESP32 publica uma mensagem no broker MQTT, contendo o número do pedido e o status “pronto”.
- **Módulo Painel (Receptor):** recebe as mensagens via MQTT e exibe o número do pedido em um display OLED, acompanhado de um alerta sonoro gerado por um buzzer. O módulo também mantém um histórico de pedidos e oferece um botão de “repetir último chamado”, permitindo reexibir e repetir o aviso anterior.

A comunicação entre os dispositivos foi simulada integralmente no Wokwi, utilizando um broker MQTT em nuvem (test.mosquitto.org), validando o funcionamento da solução sem necessidade de hardware físico.
A implementação fez uso de bibliotecas específicas, como WiFi.h, PubSubClient.h, Adafruit_SSD1306.h e ArduinoJson**.**h, para o gerenciamento de rede, publicação/assinatura MQTT, exibição gráfica e manipulação de mensagens em formato JSON.

Abaixo do arquivo contém todo o código para a aplicação do trabalho.

2.7 – Apresentação de um MVP ou Protótipo Funcional – Em Sala / No Parceiro

4. ENCERRAMENTO DO PROJETO

4.1. Relato Coletivo:

O presente relatório coletivo tem como objetivo socializar o percurso de desenvolvimento do projeto de extensão “Painel de Chamadas de Pedido com Wi-Fi (Autoatendimento)”, desenvolvido no curso de Tecnologia da Informação da Estácio – Niterói, sob orientação do professor Ralfh Ansusattigui. O projeto surgiu da necessidade de aprimorar a comunicação entre a cozinha e o atendimento em estabelecimentos alimentícios, promovendo uma experiência mais eficiente e automatizada para clientes e colaboradores.

O objetivo geral do projeto foi desenvolver uma solução tecnológica acessível que automatizasse o processo de chamada de pedidos prontos, substituindo a comunicação manual por um sistema digital de autoatendimento com Wi-Fi.

Entre os objetivos específicos, destacam-se:
• Criar um protótipo funcional de painel de chamadas utilizando microcontroladores ESP32.
• Integrar a comunicação entre os módulos de cozinha e atendimento por meio do protocolo MQTT.
• Melhorar a eficiência operacional e reduzir erros de comunicação em restaurantes e lanchonetes.
• Promover o aprendizado prático dos alunos na aplicação de conceitos de automação e Internet das Coisas (IoT).

4.1.1. Avaliação de reação da parte interessada

A parte interessada neste projeto preencheu um formulário para sabermos se o projeto foi bem aceito, útil e se atendeu as expectativas.

Abaixo está as perguntas e respostas:

O que motivou vocês a procurarem uma solução automatizada para a chamada de pedidos?

1 resposta

Nos horários de pico, a comunicação entre a cozinha e o balcão fica confusa. Muitas vezes o atendente grita o número do pedido, e o cliente não escuta. Queríamos algo mais organizado e profissional.

Como era o processo antes do sistema?

1 resposta

Antes, a gente chamava os pedidos na voz mesmo. Quando o restaurante enchia, o barulho atrapalhava e alguns clientes acabavam esperando sem saber que o pedido já estava pronto.

E como o novo sistema melhorou essa comunicação?

1 resposta

Agora a cozinha aperta um botão e o número aparece no painel, com um som de alerta. É rápido, e os clientes já olham direto pro display. Melhorou muito o fluxo.

Vocês acharam fácil usar o sistema?

1 resposta

Sim, é bem intuitivo. O botão da cozinha é simples e o painel mostra o número bem legível. Não precisa de treinamento, qualquer funcionário consegue usar.

E sobre a confiabilidade da conexão Wi-Fi, tiveram problemas?

1 resposta

Não até agora. Fizemos os testes e o envio dos pedidos é quase instantâneo. Mas seria bom ter uma forma offline se o Wi-Fi cair.

4.2. Relato de Experiência Individual (Pontuação específica para o relato individual)

Nesta seção, cada aluno deve citar seu nome, e sistematizar as aprendizagens construídas sob sua perspectiva individual. O relato deve necessariamente cobrir os seguintes itens:

4.2.1. CONTEXTUALIZAÇÃO

Explicitar a experiência/projeto vivido e contextualizar a sua participação no projeto.

4.2.2. METODOLOGIA

O desenvolvimento do projeto ocorreu entre agosto e novembro de 2025, dividido em etapas conforme o cronograma estabelecido em aula. As fases incluíram: diagnóstico e teorização, planejamento, desenvolvimento técnico, testes e relatórios finais.

O grupo utilizou o simulador Wokwi para testar e validar o funcionamento do sistema, evitando custos com componentes físicos. O sistema foi composto por dois módulos principais:
• **Módulo Cozinha (Emissor):** responsável por enviar as notificações de pedidos prontos.
• **Módulo Painel (Receptor):** responsável por exibir os pedidos prontos em um display OLED e emitir alerta sonoro.

A metodologia adotada foi colaborativa e interdisciplinar. O grupo se reuniu semanalmente para discussão dos avanços, e houve envolvimento direto com um restaurante local parceiro, que contribuiu com feedbacks e sugestões sobre o funcionamento e aplicabilidade do painel.

4.2.3. RESULTADOS E DISCUSSÃO:

O projeto alcançou resultados positivos. O protótipo simulou com sucesso a automação da comunicação entre a cozinha e o atendimento, demonstrando redução no tempo de resposta e aumento da organização.

A avaliação do parceiro comunitário (restaurante local) foi bastante favorável. O proprietário destacou que a solução proposta traria benefícios reais ao fluxo de atendimento, evitando confusões e agilizando a entrega dos pedidos.

O grupo também apresentou o MVP (Produto Mínimo Viável) em sala de aula, recebendo feedbacks positivos do professor e dos colegas quanto à aplicabilidade e ao potencial de expansão do sistema.
Durante o desenvolvimento, as principais dificuldades estiveram relacionadas à integração do código entre os módulos e à configuração da comunicação Wi-Fi simulada. Além disso, o grupo enfrentou desafios na organização do tempo e na tradução dos conceitos técnicos para linguagem acessível no relatório teórico.

Essas barreiras foram superadas através da colaboração entre os integrantes e do apoio do professor orientador.

4.2.4. REFLEXÃO APROFUNDADA

Espaço para relato sobre a experiência vivida versus teoria apresentada no relato coletivo.

4.2.5. CONSIDERAÇÕES FINAIS

O projeto de extensão “Painel de Chamadas de Pedido com Wi-Fi (Autoatendimento)” proporcionou uma experiência enriquecedora para todos os participantes, integrando teoria e prática em um contexto real e comunitário.

O trabalho reforçou a importância da automação e da Internet das Coisas como ferramentas acessíveis para resolver problemas cotidianos e melhorar serviços. Além disso, demonstrou o papel fundamental da extensão universitária na formação profissional, promovendo a união entre conhecimento acadêmico e impacto social.

Como perspectiva futura, o grupo pretende transformar o protótipo virtual em um modelo físico e funcional, realizando testes reais no restaurante parceiro e ampliando a pesquisa com novas funcionalidades.

OBSERVAÇÃO: Exige-se que todo o processo de desenvolvimento do projeto de extensão seja documentado e registrado através de evidências fotográficas ou por vídeos, tendo em vista que o conjunto de evidências não apenas irá compor a comprovação da realização das atividades, para fins regulatórios, como também poderão ser usadas para exposição do projeto em mostras acadêmico-científicas e seminários de extensão a serem realizados pelas IES.

ANEXO I - RELATO DE EXPERIÊNCIA INDIVIDUAL

Relato de Experiência Individual

Nome do Discente
Nome do Professor orientador

1. CONTEXTUALIZAÇÃO

Explicitar a experiência/projeto vivido e contextualizar a sua participação.

2. OBJETIVOS

Apresentar de forma clara os objetivos da experiência.

3. METODOLOGIA

Descrever como a experiência foi vivenciada: local; sujeitos/públicos envolvidos; período; detalhamento das etapas.

4. RESULTADOS E DISCUSSÃO

Detalhar a expectativa e o vivido; descrever o que foi observado e o que resultou a experiência; explicitar como se sentiu, as descobertas/aprendizagens, facilidades, dificuldades e recomendações, caso necessário;

5. REFLEXÃO APROFUNDADA

Discorrer sobre a relação entre a experiência vivida e a teoria estudada.

```
#include <WiFi.h>
#include <WebServer.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// ================================================
// CONFIGURAÇÕES
// ================================================

// --- Configs de Rede
const char* ssid = "Wokwi-GUEST";
const char* password = "";
WebServer server(80);

// --- Configs do Display OLED
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define BUZZER_PIN 25
#define BTN_REPETIR 13
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// ================================================
// LÓGICA DE ESTADO (EM RAM)
// ================================================
struct Pedido {
  String numero;
  String nomeCliente;
  String status;
  String descricao;
  bool urgente;
  unsigned long timestampCriacao;     // NOVO: Quando o pedido foi criado
  unsigned long timestampAtualizacao; // NOVO: Hora da última mudança de status
  unsigned long tempoPreparo;       // NOVO: Tempo (ms) entre criação e "Pronto"
};

// Pedidos Ativos
#define MAX_PEDIDOS_ATIVOS 20
Pedido listaDePedidos[MAX_PEDIDOS_ATIVOS];
static int proximoNumeroPedido = 101; // Será reiniciado a cada boot

// Histórico de Pedidos Entregues
#define MAX_HISTORICO_ENTREGUES 30
Pedido historicoDeEntregues[MAX_HISTORICO_ENTREGUES];
int indiceHistorico = 0; // Para buffer circular

// Variáveis globais para o *PAINEL OLED*
String ultimoPedidoOLED = "--";
String ultimoStatusOLED = "Aguardando";
bool btnAnteriorPainel = HIGH;
bool pedidoProntoAtivoOLED = false;
```

```
// ================================================
// PROTÓTIPOS DE FUNÇÃO
// ================================================

// Funções do Painel OLED
void tocarAlarme();
void atualizarDisplayOLED();
void repetirUltimoPedidoOLED();
void conectarWiFi();

// Funções de Gerenciamento de Pedidos (Core)
void inicializarArraysDePedidos();
int encontrarPedidoPorNumero(String numero);
int encontrarSlotVazio();
void adicionarNovoPedido(String nomeCliente, String descricao, bool urgente);
void atualizarStatusPedido(String numero, String novoStatus);
void moverParaHistorico(int indexLista);

// Funções de Servidor Web (Handlers)
void handleCozinha();
void handleEnviarNovoPedido();
void handleDashboardCozinha();
void handleAtualizarStatus();
void handlePainelCliente();
void handleHistoricoPage();
void handleRelatorioPage(); // Atualizado
void handleApiPedidosAtivos();
void handleApiPainelCliente();
void handleApiHistorico();
void handleApiRelatorio();  // Atualizado
void handleNotFound();


// ================================================
// HTML, CSS E JAVASCRIPT
// ================================================

// --- CSS Centralizado (Tema Preto e Amarelo) ---
const char htmlEstiloCSS[] PROGMEM = R"rawliteral(
<style>
    :root {
        --cor-fundo: #000000;
        --cor-fundo-card: #1a1a1a;
        --cor-texto: #FFFFFF;
        --cor-amarelo-primario: #FFD700;
        --cor-amarelo-secundario: #FFFF00;
        --cor-borda: #333333;
        --cor-urgente: #dc3545; /* Vermelho */

        --cor-sucesso: #28a745;
        --cor-pronto: #28a745;
        --cor-aviso: #ffc107; /* Em Preparo */
        --cor-aguardando: #f0a500; /* Aguardando Busca */
        --cor-entregue: #6c757d;
    }
```

```
    body {
        font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica,
Arial, sans-serif;
        background-color: var(--cor-fundo);
        color: var(--cor-texto);
        margin: 0;
        padding: 0;
    }
    .container {
        max-width: 1200px;
        margin: 20px auto;
        padding: 0 20px;
    }
    nav {
        background-color: #111;
        padding: 15px 20px;
        box-shadow: 0 2px 5px rgba(0,0,0,0.3);
        text-align: center;
    }
    nav a {
        color: var(--cor-amarelo-primario);
        text-decoration: none;
        font-weight: bold;
        font-size: 1.1em;
        margin: 0 15px;
        padding: 5px 10px;
        border-radius: 5px;
        transition: background-color 0.3s;
    }
    nav a:hover, nav a.active {
        background-color: #333;
    }
    .card {
        background: var(--cor-fundo-card);
        border-radius: 8px;
        padding: 25px;
        box-shadow: 0 4px 12px rgba(0,0,0,0.05);
        border: 1px solid var(--cor-borda);
        margin-top: 20px;
    }
    h1, h2 {
        color: var(--cor-amarelo-primario);
        margin-top: 0;
        text-align: center;
        border-bottom: 2px solid var(--cor-borda);
        padding-bottom: 15px;
    }
    label {
        display: block;
        margin-bottom: 8px;
        font-weight: 600;
    }
    input[type="text"], input[type="number"], textarea {
        width: calc(100% - 24px);
        padding: 12px;
```

```
        border: 1px solid var(--cor-borda);
        border-radius: 5px;
        font-size: 1em;
        font-family: Arial, sans-serif;
        background-color: #222;
        color: #fff;
    }
    button, .btn {
        padding: 10px 15px;
        color: white;
        border: none;
        border-radius: 5px;
        cursor: pointer;
        font-size: 0.9em;
        font-weight: bold;
        text-decoration: none;
        display: inline-block;
        margin: 2px 0;
        transition: background-color 0.3s, transform 0.1s;
    }
    button:active, .btn:active {
        transform: scale(0.98);
    }

    /* Botões do Cardápio */
    .btn-add, .btn-categoria, .btn-voltar {
        background-color: var(--cor-amarelo-primario);
        color: #000;
        padding: 10px 15px;
    }
    .btn-add:hover, .btn-categoria:hover, .btn-voltar:hover { background-color:
var(--cor-amarelo-secundario); }

    /* Botões de Ação */
    .btn-submit { width: 100%; padding: 14px; font-size: 1.1em; background-color:
var(--cor-sucesso); }
    .btn-submit:hover { background-color: #218838; }
    .btn-preparando { background-color: var(--cor-aviso); color: var(--cor-texto);}
    .btn-pronto { background-color: var(--cor-pronto); }
    .btn-aguardando { background-color: var(--cor-aguardando); }
    .btn-entregue { background-color: var(--cor-entregue); }

    /* Tabelas (Dashboard / Histórico) */
    table {
        width: 100%;
        border-collapse: collapse;
        margin-top: 20px;
    }
    th, td {
        padding: 12px 15px;
        border-bottom: 1px solid var(--cor-borda);
        text-align: left;
        vertical-align: middle;
    }
    th {
```

```
        background-color: #222;
        font-weight: 600;
    }
    tbody tr:nth-child(even) { background-color: #111; }
    tbody tr:hover { background-color: #252525; }

    tr.urgente td {
        border-top: 2px solid var(--cor-urgente);
    }
    tr.urgente .nome-cliente {
        color: var(--cor-urgente) !important;
    }

    .status-badge {
        padding: 5px 10px;
        border-radius: 15px;
        font-weight: bold;
        color: #fff;
        font-size: 0.9em;
        text-align: center;
    }
    .status-preparando { background-color: var(--cor-aviso); color: var(--cor-texto);
}
    .status-pronto { background-color: var(--cor-pronto); }
    .status-aguardando_busca { background-color: var(--cor-aguardando); }
    .status-entregue { background-color: var(--cor-entregue); }
</style>
)rawliteral";

// --- Página 1: / (Formulário com NOVO CARDÁPIO DE PASTEL) ---
const char htmlPaginaCozinha[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cozinha - Novo Pedido</title>
    {ESTILO_CSS}
    <style>
        .container { max-width: 900px; }
        .grid-container {
            display: grid;
            grid-template-columns: 1fr 1fr;
            gap: 20px;
        }
        @media (max-width: 768px) {
            .grid-container { grid-template-columns: 1fr; }
        }

        /* Cardápio */
        #menu-itens .item {
            padding: 15px;
            border: 1px solid var(--cor-borda);
            border-radius: 5px;
            margin-bottom: 15px;
```

```
        }
        #menu-itens .item-header {
            display: flex;
            justify-content: space-between;
            align-items: center;
        }
        #menu-itens .item-header h4 { margin: 0; font-size: 1.2em; color:
var(--cor-amarelo-primario); }
        #menu-itens .item-obs {
            width: calc(100% - 20px);
            padding: 8px 10px;
            font-size: 0.9em;
            margin-top: 10px;
        }
        .btn-categoria {
            width: 100%;
            margin-bottom: 10px;
            text-align: left;
            font-size: 1.1em;
        }
        .btn-voltar { width: 100%; background: #444; }

        /* Resumo */
        #resumo-pedido li {
            list-style-type: none;
            padding: 8px;
            border-bottom: 1px dashed var(--cor-borda);
        }
        #resumo-pedido li strong { color: var(--cor-amarelo-primario); }
        #resumo-pedido li small { color: #ccc; display: block; }

        /* Checkbox Urgente */
        .urgente-label {
            display: flex;
            align-items: center;
            font-size: 1.1em;
            color: var(--cor-urgente);
            font-weight: bold;
            margin-top: 15px;
        }
        .urgente-label input {
            width: auto;
            margin-right: 10px;
            transform: scale(1.3);
        }
    </style>
</head>
<body>
    <nav>
        <a href="/" class="active">Novo Pedido</a>
        <a href="/dashboard">Dashboard Cozinha</a>
        <a href="/painel">Painel Cliente</a>
        <a href="/historico">Histórico</a>
        <a href="/relatorio">Relatório</a>
    </nav>
```

```
    <div class="container">
        <h1>Criar Novo Pedido</h1>

        <form action="/enviar" method="POST">
            <div class="grid-container">
                <div class="card">
                    <h2 id="menu-titulo">Categorias</h2>
                    <div id="menu-itens">
                        </div>
                </div>

                <div class="card">
                    <h2>Resumo do Pedido</h2>

                    <div style="margin-bottom: 20px;">
                        <label for="nome">Nome do Cliente:</label>
                        <input type="text" id="nome" name="nome" placeholder="Ex:
Maria" required>
                    </div>

                    <ul id="resumo-pedido">
                        <li id="placeholder-resumo" style="color: #888; text-align:
center;">Clique nos itens ao lado...</li>
                    </ul>

                    <textarea id="descricao" name="descricao" rows="4" required
style="display:none;"></textarea>

                    <label class="urgente-label">
                        <input type="checkbox" id="urgente" name="urgente"
value="true">
                        🔥 PEDIDO URGENTE
                    </label>

                    <button id="btn-enviar-pedido" type="submit" class="btn-submit"
style="margin-top: 20px;" disabled>Adicionar Pedido</button>
                </div>
            </div>
        </form>
    </div>

    <script>
        // ATUALIZADO: Cardápio focado em Pastelaria
        const CARDAPIO = {
            "Categorias": [
                { id: 1, nome: "Pastéis Salgados", subMenu: "Pasteis Salgados" },
                { id: 2, nome: "Pastéis Doces", subMenu: "Pasteis Doces" },
                { id: 3, nome: "Caldo de Cana", subMenu: "Caldos" },
                { id: 4, nome: "Bebidas", subMenu: "Bebidas" }
            ],
            "Pasteis Salgados": [
                { id: 10, nome: "Carne", preco: 8.00 },
                { id: 11, nome: "Queijo", preco: 8.00 },
                { id: 12, nome: "Frango", preco: 8.00 },
                { id: 13, nome: "Pizza", preco: 8.50 },
```

```
                { id: 14, nome: "Palmito", preco: 8.50 },
                { id: 15, nome: "Carne Seca", preco: 9.00 },
                { id: 16, nome: "Camarão", preco: 10.00 },
                { id: 17, nome: "Frango c/ Catupiry", preco: 9.50 },
                { id: 18, nome: "Carne Seca c/ Catupiry", preco: 10.00 },
                { id: 19, nome: "Camarão c/ Catupiry", preco: 11.00 },
                { id: 20, nome: "Palmito c/ Catupiry", preco: 9.50 }
            ],
            "Pasteis Doces": [
                { id: 50, nome: "Chocolate", preco: 9.00 },
                { id: 51, nome: "Romeu e Julieta", preco: 9.00 },
                { id: 52, nome: "Banana c/ Canela", preco: 8.00 },
                { id: 53, nome: "Nutella", preco: 10.00 },
                { id: 54, nome: "Nutella c/ Morango", preco: 11.00 },
                { id: 55, nome: "Doce de Leite", preco: 9.00 },
                { id: 56, nome: "Doce de Leite c/ Nutella", preco: 11.00 }
            ],
            "Caldos": [
                { id: 30, nome: "Caldo P (300ml)", preco: 5.00 },
                { id: 31, nome: "Caldo M (500ml)", preco: 7.00 },
                { id: 32, nome: "Caldo G (700ml)", preco: 9.00 }
            ],
            "Bebidas": [
                { id: 40, nome: "Suco Natural", subMenu: "Sucos" },
                { id: 41, nome: "Refrigerante Lata", preco: 6.00 },
                { id: 42, nome: "Água sem Gás", preco: 4.00 },
                { id: 43, nome: "Água com Gás", preco: 4.00 }
            ],
            "Sucos": [
                { id: 100, nome: "Suco Laranja", preco: 7.00 },
                { id: 101, nome: "Suco Maracujá", preco: 7.00 },
                { id: 102, nome: "Suco Abacaxi", preco: 7.00 },
                { id: 103, nome: "Suco Morango", preco: 7.00 }
            ]
        };

        const pedidoAtual = [];
        const menuEl = document.getElementById('menu-itens');
        const menuTituloEl = document.getElementById('menu-titulo');
        let menuHistory = [];

        function renderMenu(menuKey) {
            const menuData = CARDAPIO[menuKey];
            menuTituloEl.innerText = menuKey;
            menuEl.innerHTML = '';

            if (menuHistory.length > 0) {
                const parentKey = menuHistory[menuHistory.length - 1];
                menuEl.innerHTML += `
                    <button type="button" class="btn btn-voltar" style="margin-bottom:
15px;" onclick="voltarMenu()">
                        &larr; Voltar para ${parentKey}
                    </button>
                `;
            }
```

```
            menuData.forEach(item => {
                if (item.subMenu) {
                    menuEl.innerHTML += `
                        <button type="button" class="btn btn-categoria"
onclick="navegarParaMenu('${item.subMenu}')">
                            ${item.nome} &rarr;
                        </button>
                    `;
                }
                else if (item.preco) {
                    menuEl.innerHTML += `
                        <div class="item">
                            <div class="item-header">
                                <div>
                                    <h4>${item.nome}</h4>
                                    <span>R$ ${item.preco.toFixed(2)}</span>
                                </div>
                                <button type="button" class="btn btn-add"
onclick="adicionarItem(${item.id}, '${item.nome}')">Adicionar</button>
                            </div>
                            <input type="text" id="obs-${item.id}" class="item-obs"
placeholder="Obs: (Opcional)">
                        </div>
                    `;
                }
            });
        }

        function navegarParaMenu(menuKey) {
            menuHistory.push(menuTituloEl.innerText);
            renderMenu(menuKey);
        }

        function voltarMenu() {
            const parentKey = menuHistory.pop();
            renderMenu(parentKey);
        }

        document.addEventListener('DOMContentLoaded', () => {
            renderMenu("Categorias");
        });

        function adicionarItem(id, nome) {
            const obsEl = document.getElementById('obs-' + id);
            const obs = obsEl.value;

            pedidoAtual.push({
                nome: nome,
                obs: obs || ""
            });

            obsEl.value = '';
            atualizarResumo();
        }
```

```
        function removerItem(index) {
            pedidoAtual.splice(index, 1);
            atualizarResumo();
        }

        function atualizarResumo() {
            const resumoEl = document.getElementById('resumo-pedido');
            const hiddenTextarea = document.getElementById('descricao');
            const placeholder = document.getElementById('placeholder-resumo');
            const btnEnviar = document.getElementById('btn-enviar-pedido');

            resumoEl.innerHTML = '';
            let descricaoFinal = '';

            if (pedidoAtual.length === 0) {
                resumoEl.appendChild(placeholder);
                btnEnviar.disabled = true;
            } else {
                btnEnviar.disabled = false;

                const agrupado = {};
                pedidoAtual.forEach((item, index) => {
                    const chave = item.nome + (item.obs ? ` (${item.obs})` : '');
                    if (!agrupado[chave]) {
                        agrupado[chave] = { qt: 0, nome: item.nome, obs: item.obs,
indices: [] };
                    }
                    agrupado[chave].qt++;
                    agrupado[chave].indices.push(index);
                });

                Object.keys(agrupado).forEach(chave => {
                    const item = agrupado[chave];

                    const li = document.createElement('li');
                    let htmlItem = `<strong>${item.qt}x ${item.nome}</strong>`;
                    if (item.obs) {
                        htmlItem += `<small>Obs: ${item.obs}</small>`;
                    }
                    htmlItem += ` <button type="button" class="btn" style="background:
#dc3545; font-size: 0.8em; padding: 2px 5px;"
onclick="removerItem(${item.indices[0]})">x</button>`;
                    li.innerHTML = htmlItem;
                    resumoEl.appendChild(li);

                    descricaoFinal += `${item.qt}x ${item.nome}`;
                    if (item.obs) {
                        descricaoFinal += ` (${item.obs})`;
                    }
                    descricaoFinal += "; ";
                });
            }

            hiddenTextarea.value = descricaoFinal.slice(0, -2);
```

```
        }
    </script>
</body>
</html>
)rawliteral";

// --- Página 2: /dashboard (Dashboard com AJAX e URGENTE) ---
const char htmlPaginaDashboard[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dashboard - Cozinha</title>
    {ESTILO_CSS}
    <style>
      .acoes-form { display: flex; flex-wrap: wrap; gap: 5px; }
      .acoes-form .btn { flex-grow: 1; font-size: 0.8em; }
      td .nome-cliente { font-size: 1.1em; font-weight: bold; color:
var(--cor-amarelo-primario); }
      td .num-pedido { font-size: 0.9em; color: #888; }
      td .urgente-flag {
          font-weight: bold;
          color: var(--cor-urgente);
          display: block;
          font-size: 0.8em;
      }
    </style>
</head>
<body>
    <nav>
        <a href="/">Novo Pedido</a>
        <a href="/dashboard" class="active">Dashboard Cozinha</a>
        <a href="/painel">Painel Cliente</a>
        <a href="/historico">Histórico</a>
        <a href="/relatorio">Relatório</a>
    </nav>
    <div class="container">
        <div class="card">
            <h1>Pedidos Ativos</h1>
            <table id="dashboard-table">
                <thead>
                    <tr>
                        <th style="width: 15%;">Cliente</th>
                        <th>Descrição</th>
                        <th style="width: 15%;">Status</th>
                        <th style="width: 15%;">Há (min)</th>
                        <th style="width: 30%;">Ações</th>
                    </tr>
                </thead>
                <tbody id="dashboard-body">
                    </tbody>
            </table>
        </div>
    </div>
```

```
    <script>
        function criarBotoesAcao(numero, status) {
            return `
                <div class="acoes-form" id="acoes-${numero}">
                    <button type="button" name="status" value="preparando" class="btn
btn-preparando" ${status === 'preparando' ? 'disabled' : ''}
onclick="atualizarStatus('${numero}', 'preparando')">Preparo</button>
                    <button type="button" name="status" value="pronto" class="btn
btn-pronto" ${status === 'pronto' ? 'disabled' : ''}
onclick="atualizarStatus('${numero}', 'pronto')">Pronto</button>
                    <button type="button" name="status" value="aguardando_busca"
class="btn btn-aguardando" ${status === 'aguardando_busca' ? 'disabled' : ''}
onclick="atualizarStatus('${numero}', 'aguardando_busca')">Aguardando</button>
                    <button type="button" name="status" value="entregue" class="btn
btn-entregue" onclick="atualizarStatus('${numero}', 'entregue')">Entregue</button>
                </div>
            `;
        }

        function atualizarStatus(numero, novoStatus) {
            const formData = new URLSearchParams();
            formData.append('numero', numero);
            formData.append('status', novoStatus);

            fetch('/atualizar', {
                method: 'POST',
                headers: {
                  'Content-Type': 'application/x-www-form-urlencoded',
                },
                body: formData
            })
            .then(response => response.json())
            .then(data => {
                if (data.success) {
                    const row = document.getElementById('pedido-row-' + numero);
                    if (!row) return;

                    if (novoStatus === 'entregue') {
                        row.style.opacity = 0;
                        setTimeout(() => row.remove(), 500);
                    }
                    else {
                        const statusEl = row.querySelector('.status-badge');
                        statusEl.className = 'status-badge status-' +
novoStatus.replace('_', '-');
                        statusEl.innerText = novoStatus.charAt(0).toUpperCase() +
novoStatus.slice(1).replace('_', ' ');

                        const botoes = row.querySelector('.acoes-form');
                        botoes.querySelectorAll('button').forEach(btn => {
                            btn.disabled = (btn.value === novoStatus);
                        });
                    }
                } else {
                    alert('Erro ao atualizar status.');
```

```
                }
            })
            .catch(error => console.error('Erro no Fetch:', error));
        }

        function buscarPedidos() {
            fetch('/api/pedidos')
                .then(response => response.json())
                .then(data => {
                    const tbody = document.getElementById('dashboard-body');
                    tbody.innerHTML = '';
                    if (data.length === 0) {
                        tbody.innerHTML = '<tr><td colspan="5"
style="text-align:center; padding: 20px; color: #888;">Nenhum pedido
ativo.</td></tr>';
                        return;
                    }

                    data.sort((a, b) => {
                        if (a.urgente !== b.urgente) {
                            return a.urgente ? -1 : 1;
                        }
                        const ordem = { 'preparando': 1, 'pronto': 2,
'aguardando_busca': 3 };
                        return (ordem[a.status] || 99) - (ordem[b.status] || 99) ||
a.numero - b.numero;
                    });

                    data.forEach(pedido => {
                        const tr = document.createElement('tr');
                        tr.id = 'pedido-row-' + pedido.numero;
                        if (pedido.urgente) {
                            tr.classList.add('urgente');
                        }

                        let statusClasse = 'status-' + pedido.status.replace('_',
'-');
                        let statusTexto = pedido.status.charAt(0).toUpperCase() +
pedido.status.slice(1).replace('_', ' ');
                        // USA O TIMESTAMP DE ATUALIZAÇÃO
                        let minutosAtras = ((Date.now() - pedido.timestamp) /
60000).toFixed(1);

                        tr.innerHTML = `
                            <td>
                                <div class="nome-cliente">${pedido.nomeCliente}</div>
                                <div class="num-pedido">#${pedido.numero}</div>
                                ${pedido.urgente ? '<span class="urgente-flag">🔥
URGENTE</span>' : ''}
                            </td>
                            <td>${pedido.descricao.replace(/; /g, '<br>')}</td>
                            <td><span class="status-badge
${statusClasse}">${statusTexto}</span></td>
                            <td>${minutosAtras} min</td>
                            <td>${criarBotoesAcao(pedido.numero, pedido.status)}</td>
```

```
                        `;
                        tbody.appendChild(tr);
                    });
                })
                .catch(error => console.error('Erro ao buscar pedidos:', error));
        }
        document.addEventListener('DOMContentLoaded', buscarPedidos);
        setInterval(buscarPedidos, 5000);
    </script>
</body>
</html>
)rawliteral";

// --- Página 3: /painel (Painel KDS Cliente) ---
const char htmlPaginaPainel[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Painel de Pedidos</title>
    <style>
        :root {
            --cor-fundo: #000000;
            --cor-texto: #FFFFFF;
            --cor-amarelo-primario: #FFD700;
            --cor-amarelo-secundario: #FFFF00;
            --cor-borda: #333333;
            --cor-card-fundo: #1a1a1a;
        }
        body {
            font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,
Helvetica, Arial, sans-serif;
            background-color: var(--cor-fundo);
            color: var(--cor-texto);
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
        nav {
            background-color: #111;
            padding: 15px 20px;
            box-shadow: 0 2px 5px rgba(0,0,0,0.3);
            text-align: center;
        }
        nav a {
            color: var(--cor-amarelo-primario);
            text-decoration: none;
            font-weight: bold;
            font-size: 1.1em;
            margin: 0 15px;
            padding: 5px 10px;
            border-radius: 5px;
            transition: background-color 0.3s;
        }
```

```
nav a:hover, nav a.active {
    background-color: #333;
}
.kds-container {
    display: flex;
    width: 100%;
    height: calc(100vh - 70px);
    margin-top: 0;
    padding: 0;
}
.kds-coluna {
    width: 50%;
    padding: 15px;
    box-sizing: border-box;
    overflow-y: auto;
}
.kds-coluna h1 {
    text-align: center;
    padding-bottom: 15px;
    margin-bottom: 20px;
    font-size: 2.5em;
}
#col-preparo {
    background-color: #0a0a0a;
    border-right: 2px solid var(--cor-borda);
}
#col-preparo h1 {
    color: var(--cor-amarelo-primario);
    border-bottom: 2px solid var(--cor-borda);
}
#col-pronto {
    background-color: #000;
}
#col-pronto h1 {
    color: var(--cor-amarelo-secundario);
    border-bottom: 2px solid var(--cor-borda);
}

.pedido-card {
    background: var(--cor-card-fundo);
    color: var(--cor-texto);
    border-radius: 8px;
    padding: 20px;
    margin-bottom: 15px;
    text-align: center;
    border: 1px solid var(--cor-borda);
    box-shadow: 0 4px 10px rgba(0,0,0,0.3);
}

.pedido-card-nome {
    font-size: 3.5em;
    font-weight: bold;
    line-height: 1.1;
    margin: 0;
    text-transform: uppercase;
```

```
        }
        .pedido-card-numero {
            font-size: 1.2em;
            color: #888;
            margin-top: 5px;
        }
        .pedido-card-status {
            font-size: 1.2em;
            font-style: italic;
            margin-top: 10px;
        }

        #col-preparo .pedido-card { border-color: var(--cor-amarelo-primario); }
        #col-preparo .pedido-card-nome { color: var(--cor-amarelo-primario); }

        #col-pronto .pedido-card { border: 2px solid var(--cor-amarelo-secundario); }
        #col-pronto .pedido-card-nome { color: var(--cor-amarelo-secundario); }
        #col-pronto .status-pronto .pedido-card-status {
            color: var(--cor-amarelo-secundario);
            font-weight: bold;
        }
        #col-pronto .status-aguardando { animation: piscar 1.2s linear infinite; }
        #col-pronto .status-aguardando .pedido-card-status {
            color: var(--cor-amarelo-secundario);
            font-weight: bold;
        }
        @keyframes piscar {
            0% { box-shadow: 0 0 15px var(--cor-amarelo-secundario); }
            50% { box-shadow: 0 0 0px transparent; }
            100% { box-shadow: 0 0 15px var(--cor-amarelo-secundario); }
        }
    </style>
</head>
<body>
    <nav>
        <a href="/">Novo Pedido</a>
        <a href="/dashboard">Dashboard Cozinha</a>
        <a href="/painel" class="active">Painel Cliente</a>
        <a href="/historico">Histórico</a>
        <a href="/relatorio">Relatório</a>
    </nav>

    <div class="kds-container">
        <div class="kds-coluna" id="col-preparo">
            <h1>EM PREPARO</h1>
            <div id="lista-preparo">
                </div>
        </div>
        <div class="kds-coluna" id="col-pronto">
            <h1>PRONTOS / AGUARDANDO</h1>
            <div id="lista-pronto">
                </div>
        </div>
    </div>
```

```
    <script>
        function criarCardPedido(pedido) {
            let statusTexto = "";
            let statusClasse = "";

            if (pedido.status === 'pronto') {
                statusTexto = "Pronto!";
                statusClasse = "status-pronto";
            } else if (pedido.status === 'aguardando_busca') {
                statusTexto = "Aguardando Retirada";
                statusClasse = "status-aguardando";
            } else {
                statusTexto = "Em Preparo...";
            }

            return `
                <div class="pedido-card ${statusClasse}" id="pedido-${pedido.numero}">
                    <h2 class="pedido-card-nome">${pedido.nomeCliente}</h2>
                    <p class="pedido-card-numero">#${pedido.numero}</p>
                    <p class="pedido-card-status">${statusTexto}</p>
                </div>
            `;
        }
        function buscarPedidosPainel() {
            fetch('/api/painel')
                .then(response => response.json())
                .then(data => {
                    const listaPreparo = document.getElementById('lista-preparo');
                    const listaPronto = document.getElementById('lista-pronto');
                    listaPreparo.innerHTML = '';
                    listaPronto.innerHTML = '';

                    if (data.preparando && data.preparando.length > 0) {
                        data.preparando.sort((a,b) => a.numero - b.numero);
                        data.preparando.forEach(pedido => {
                            listaPreparo.innerHTML += criarCardPedido(pedido);
                        });
                    } else {
                        listaPreparo.innerHTML = '<p style="text-align:center; color:
#888;">Nenhum pedido em preparo.</p>';
                    }

                    if (data.prontos && data.prontos.length > 0) {
                        data.prontos.sort((a,b) => {
                            if (a.urgente !== b.urgente) return a.urgente ? -1 : 1;
                            return b.numero - a.numero;
                        });
                        data.prontos.forEach(pedido => {
                            listaPronto.innerHTML += criarCardPedido(pedido);
                        });
                    } else {
                        listaPronto.innerHTML = '<p style="text-align:center; color:
#888;">Nenhum pedido pronto.</p>';
                    }
                })
```

```
                .catch(error => console.error('Erro ao buscar pedidos do painel:',
error));
        }
        document.addEventListener('DOMContentLoaded', buscarPedidosPainel);
        setInterval(buscarPedidosPainel, 2500);
    </script>
</body>
</html>
)rawliteral";

// --- Página 4: /historico (Página de Histórico) ---
const char htmlPaginaHistorico[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Histórico de Pedidos</title>
    {ESTILO_CSS}
    <style>
      td .nome-cliente { font-size: 1.1em; font-weight: bold; color:
var(--cor-amarelo-primario); }
      td .num-pedido { font-size: 0.9em; color: #888; }
      td .urgente-flag {
          font-weight: bold;
          color: var(--cor-urgente);
          display: block;
          font-size: 0.8em;
      }
    </style>
</head>
<body>
    <nav>
        <a href="/">Novo Pedido</a>
        <a href="/dashboard">Dashboard Cozinha</a>
        <a href="/painel">Painel Cliente</a>
        <a href="/historico" class="active">Histórico</a>
        <a href="/relatorio">Relatório</a>
    </nav>
    <div class="container">
        <div class="card">
            <h1>Histórico de Pedidos Entregues</h1>
            <p style="text-align:center; color: #888;">Este histórico está na memória
RAM e será perdido se o ESP32 for reiniciado.</p>
            <table id="historico-table">
                <thead>
                    <tr>
                        <th style="width: 15%;">Cliente</th>
                        <th>Descrição</th>
                        <th style="width: 15%;">Status</th>
                        <th style="width: 20%;">Entregue Há (min)</th>
                    </tr>
                </thead>
                <tbody id="historico-body">
                    </tbody>
```

```
            </table>
        </div>
    </div>
    <script>
        function buscarHistorico() {
            fetch('/api/historico')
                .then(response => response.json())
                .then(data => {
                    const tbody = document.getElementById('historico-body');
                    tbody.innerHTML = '';
                    if (data.length === 0) {
                        tbody.innerHTML = '<tr><td colspan="4"
style="text-align:center; padding: 20px; color: #888;">Nenhum pedido no
histórico.</td></tr>';
                        return;
                    }
                    data.sort((a, b) => b.timestamp - a.timestamp); // Usa o timestamp
de atualização
                    data.forEach(pedido => {
                        const tr = document.createElement('tr');
                        let minutosAtras = ((Date.now() - pedido.timestamp) /
60000).toFixed(1);

                        tr.innerHTML = `
                            <td>
                                <div class="nome-cliente">${pedido.nomeCliente}</div>
                                <div class="num-pedido">#${pedido.numero}</div>
                                ${pedido.urgente ? '<span class="urgente-flag">🔥
URGENTE</span>' : ''}
                            </td>
                            <td>${pedido.descricao.replace(/; /g, '<br>')}</td>
                            <td><span class="status-badge
status-entregue">Entregue</span></td>
                            <td>${minutosAtras} min</td>
                        `;
                        tbody.appendChild(tr);
                    });
                })
                .catch(error => console.error('Erro ao buscar histórico:', error));
        }
        document.addEventListener('DOMContentLoaded', buscarHistorico);
        setInterval(buscarHistorico, 10000);
    </script>
</body>
</html>
)rawliteral";

// --- PÁGINA ATUALIZADA: /relatorio (Relatório de TEMPO DE PREPARO) ---
const char htmlPaginaRelatorio[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Relatório de Desempenho</title>
```

```
    {ESTILO_CSS}
    <style>
        .stats-container {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
            gap: 20px;
            margin-top: 20px;
        }
        .stat-card {
            background: #222;
            padding: 20px;
            border-radius: 5px;
            text-align: center;
        }
        .stat-card h3 {
            margin-top: 0;
            color: var(--cor-amarelo-primario);
            font-size: 1.2em;
        }
        .stat-card .valor {
            font-size: 2.5em;
            font-weight: bold;
            color: var(--cor-amarelo-secundario);
        }
    </style>
</head>
<body>
    <nav>
        <a href="/">Novo Pedido</a>
        <a href="/dashboard">Dashboard Cozinha</a>
        <a href="/painel">Painel Cliente</a>
        <a href="/historico">Histórico</a>
        <a href="/relatorio" class="active">Relatório</a>
    </nav>
    <div class="container">
        <div class="card">
            <h1>Relatório de Desempenho da Cozinha</h1>
            <p style="text-align:center; color: #888;">Análise de pedidos marcados
como "Pronto" nesta sessão.</p>

            <div id="report-content">
                </div>
        </div>
    </div>
    <script>
        function formatarSegundos(segundos) {
            const min = Math.floor(segundos / 60);
            const seg = segundos % 60;
            return `${min}m ${seg}s`;
        }

        function buscarRelatorio() {
            fetch('/api/relatorio')
                .then(response => response.json())
                .then(data => {
```

```
                    const container = document.getElementById('report-content');

                    if (!data || data.totalPedidos === 0) {
                         container.innerHTML = '<p style="text-align:center; color:
#888;">Nenhum dado de relatório ainda. Prepare e entregue alguns pedidos.</p>';
                         return;
                    }

                    container.innerHTML = `
                        <div class="stats-container">
                            <div class="stat-card">
                                <h3>Total de Pedidos Prontos</h3>
                                <div class="valor">${data.totalPedidos}</div>
                            </div>
                            <div class="stat-card">
                                <h3>Tempo Médio de Preparo</h3>
                                <div
class="valor">${formatarSegundos(data.mediaSegundos)}</div>
                            </div>
                            <div class="stat-card">
                                <h3>Preparo Mais Rápido</h3>
                                <div
class="valor">${formatarSegundos(data.minSegundos)}</div>
                            </div>
                            <div class="stat-card">
                                <h3>Preparo Mais Lento</h3>
                                <div
class="valor">${formatarSegundos(data.maxSegundos)}</div>
                            </div>
                        </div>
                    `;
                })
                .catch(error => console.error('Erro ao buscar relatório:', error));
        }
        document.addEventListener('DOMContentLoaded', buscarRelatorio);
    </script>
</body>
</html>
)rawliteral";


// ================================================
// FUNÇÕES C++ (SETUP, LOOP, HANDLERS, ETC)
// (COM LÓGICA DE TEMPO DE PREPARO)
// ================================================

// ... (OLED)
void tocarAlarme() {
  for (int i = 0; i < 3; i++) {
    tone(BUZZER_PIN, 2000, 200);
    delay(250);
    tone(BUZZER_PIN, 2500, 200);
    delay(250);
  }
  noTone(BUZZER_PIN);
```

```
}

void atualizarDisplayOLED() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);

  if (pedidoProntoAtivoOLED && (millis() / 500) % 2 == 0) {
    display.fillRect(0, 0, 128, 12, SSD1306_WHITE);
    display.setTextColor(SSD1306_BLACK);
  }

  display.setCursor(8, 2);
  display.println("PAINEL DE PEDIDOS");
  display.setTextColor(SSD1306_WHITE);
  display.drawLine(0, 13, 128, 13, SSD1306_WHITE);

  display.setTextSize(3);
  display.setCursor(10, 20);
  display.print("#");
  display.println(ultimoPedidoOLED);

  display.setTextSize(1);
  display.setCursor(5, 50);

  if (ultimoStatusOLED == "preparando") {
    display.print("Em preparo");
    int dots = (millis() / 500) % 4;
    for (int i = 0; i < dots; i++) display.print(".");
  } else if (ultimoStatusOLED == "pronto") {
    display.setTextSize(2);
    display.setCursor(10, 48);
    display.println("PRONTO!");
  } else if (ultimoStatusOLED == "aguardando_busca") {
    display.setTextSize(1);
    display.setCursor(5, 50);
    display.println("AGUARDANDO RETIRADA");
  } else {
    display.println(ultimoStatusOLED);
  }

  display.display();
}

void repetirUltimoPedidoOLED() {
  if (ultimoPedidoOLED == "--") {
    Serial.println("Nenhum pedido no painel OLED para repetir!");
    return;
  }
  Serial.println("\nREPETINDO ALARME (OLED):");
  Serial.print("  Pedido #");
  Serial.println(ultimoPedidoOLED);
  pedidoProntoAtivoOLED = true;
  tocarAlarme();
}
```

```
void conectarWiFi() {
  Serial.println("=== SISTEMA KDS INICIANDO ===");
  WiFi.begin(ssid, password);
  Serial.print("Conectando WiFi");
  int tentativas = 0;
  while (WiFi.status() != WL_CONNECTED && tentativas < 20) {
    delay(500);
    Serial.print(".");
    tentativas++;
  }
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nWiFi conectado!");
    Serial.print("IP: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\nFalha WiFi!");
  }
}

// ... (Gerenciamento de Pedidos - ATUALIZADO)
void inicializarArraysDePedidos() {
  for (int i = 0; i < MAX_PEDIDOS_ATIVOS; i++) {
    listaDePedidos[i].status = "vazio";
    listaDePedidos[i].numero = "";
    listaDePedidos[i].nomeCliente = "";
    listaDePedidos[i].tempoPreparo = 0;
  }
  for (int i = 0; i < MAX_HISTORICO_ENTREGUES; i++) {
    historicoDeEntregues[i].status = "vazio";
    historicoDeEntregues[i].numero = "";
    historicoDeEntregues[i].nomeCliente = "";
    historicoDeEntregues[i].tempoPreparo = 0;
  }

  proximoNumeroPedido = 101;
  indiceHistorico = 0;

  Serial.println("Arrays de pedidos e histórico inicializados (RAM).");
}

int encontrarPedidoPorNumero(String numero) {
  for (int i = 0; i < MAX_PEDIDOS_ATIVOS; i++) {
    if (listaDePedidos[i].numero == numero && listaDePedidos[i].status != "vazio") {
      return i;
    }
  }
  return -1;
}

int encontrarSlotVazio() {
  for (int i = 0; i < MAX_PEDIDOS_ATIVOS; i++) {
    if (listaDePedidos[i].status == "vazio") {
      return i;
    }
```

```
  }
  return -1;
}

void adicionarNovoPedido(String nomeCliente, String descricao, bool urgente) {
  int i = encontrarSlotVazio();
  if (i == -1) {
    Serial.println("ERRO: Lista de pedidos cheia!");
    return;
  }

  unsigned long agora = millis();
  listaDePedidos[i].numero = String(proximoNumeroPedido++);
  listaDePedidos[i].nomeCliente = nomeCliente;
  listaDePedidos[i].descricao = descricao;
  listaDePedidos[i].status = "preparando";
  listaDePedidos[i].timestampCriacao = agora;     // Salva tempo de criação
  listaDePedidos[i].timestampAtualizacao = agora; // Salva tempo de atualização
  listaDePedidos[i].tempoPreparo = 0;           // Zera tempo de preparo
  listaDePedidos[i].urgente = urgente;

  Serial.println("\n=== NOVO PEDIDO ADICIONADO ===");
  Serial.print("Numero: #");
  Serial.println(listaDePedidos[i].numero);
  Serial.print("Cliente: ");
  Serial.println(listaDePedidos[i].nomeCliente);
  if(urgente) Serial.println("Status: 🔥 URGENTE");
}

void atualizarStatusPedido(String numero, String novoStatus) {
  int i = encontrarPedidoPorNumero(numero);
  if (i == -1) {
    Serial.print("ERRO: Tentativa de atualizar pedido nao encontrado: #");
    Serial.println(numero);
    return;
  }

  Serial.println("\n=== STATUS DO PEDIDO ATUALIZADO ===");
  Serial.print("Numero: #");
  Serial.println(numero);
  Serial.print("Novo Status: ");
  Serial.println(novoStatus);

  // ATUALIZADO: Lógica de Tempo de Preparo
  if ((novoStatus == "pronto" || novoStatus == "aguardando_busca") &&
      (listaDePedidos[i].status == "preparando") &&
      (listaDePedidos[i].tempoPreparo == 0))
  {
    listaDePedidos[i].tempoPreparo = millis() - listaDePedidos[i].timestampCriacao;
    Serial.print("Tempo de Preparo registrado: ");
    Serial.print(listaDePedidos[i].tempoPreparo / 1000);
    Serial.println("s");
  }

  if (novoStatus == "entregue") {
```

```
    moverParaHistorico(i);
  } else {
    listaDePedidos[i].status = novoStatus;
    listaDePedidos[i].timestampAtualizacao = millis(); // Atualiza o tempo
  }

  if (novoStatus == "pronto" || novoStatus == "aguardando_busca") {
    ultimoPedidoOLED = numero;
    ultimoStatusOLED = novoStatus;
    pedidoProntoAtivoOLED = true;
    if (novoStatus == "pronto") {
      tocarAlarme();
      Serial.println("ALARME TOCADO!");
    }
  }
  else if (novoStatus == "entregue" && ultimoPedidoOLED == numero) {
    ultimoPedidoOLED = "--";
    ultimoStatusOLED = "Aguardando";
    pedidoProntoAtivoOLED = false;
  }

  atualizarDisplayOLED();
  Serial.println("================================\n");
}

void moverParaHistorico(int indexLista) {
  // Copia o pedido inteiro (incluindo tempo de preparo)
  historicoDeEntregues[indiceHistorico] = listaDePedidos[indexLista];
  historicoDeEntregues[indiceHistorico].status = "entregue";
  historicoDeEntregues[indiceHistorico].timestampAtualizacao = millis(); // Marca a
hora da entrega

  Serial.print("Pedido #");
  Serial.print(historicoDeEntregues[indiceHistorico].numero);
  Serial.println(" movido para o histórico.");

  indiceHistorico = (indiceHistorico + 1) % MAX_HISTORICO_ENTREGUES;

  // Limpa o slot no array de pedidos ativos
  listaDePedidos[indexLista].status = "vazio";
  listaDePedidos[indexLista].numero = "";
  listaDePedidos[indexLista].descricao = "";
  listaDePedidos[indexLista].nomeCliente = "";
  listaDePedidos[indexLista].urgente = false;
  listaDePedidos[indexLista].tempoPreparo = 0;
  listaDePedidos[indexLista].timestampCriacao = 0;
  listaDePedidos[indexLista].timestampAtualizacao = 0;
}


// ... (Web Handlers)
String processarTemplate(const char* html) {
  String htmlProcessado = String(html);
  htmlProcessado.replace("{ESTILO_CSS}", htmlEstiloCSS);
  return htmlProcessado;
```

```
}

void handleCozinha() {
  server.send(200, "text/html", processarTemplate(htmlPaginaCozinha));
}

void handleEnviarNovoPedido() {
  if (server.hasArg("nome") && server.hasArg("descricao")) {
    bool urgente = server.hasArg("urgente");
    adicionarNovoPedido(server.arg("nome"), server.arg("descricao"), urgente);
    server.sendHeader("Location", "/dashboard");
    server.send(302, "text/plain", "Redirecionando...");
  } else {
    server.send(400, "text/plain", "Erro: Faltando nome ou descricao.");
  }
}

void handleDashboardCozinha() {
  server.send(200, "text/html", processarTemplate(htmlPaginaDashboard));
}

void handleHistoricoPage() {
  server.send(200, "text/html", processarTemplate(htmlPaginaHistorico));
}

// ATUALIZADO: Handler para a nova página de Relatório
void handleRelatorioPage() {
  server.send(200, "text/html", processarTemplate(htmlPaginaRelatorio));
}

void handleAtualizarStatus() {
  if (server.hasArg("numero") && server.hasArg("status")) {
    String numero = server.arg("numero");
    String status = server.arg("status");
    atualizarStatusPedido(numero, status);

    String jsonResponse = "{\"success\":true, \"numero\":\"" + numero + "\",
\"novoStatus\":\"" + status + "\"}";
    server.send(200, "application/json", jsonResponse);
  } else {
    server.send(400, "application/json", "{\"success\":false, \"error\":\"Faltando
argumentos.\"}");
  }
}

void handlePainelCliente() {
  server.send(200, "text/html", htmlPaginaPainel);
}

// ATUALIZADO: APIs enviam 'timestampAtualizacao' como 'timestamp'
void handleApiPedidosAtivos() {
  StaticJsonDocument<4096> doc;
  JsonArray array = doc.to<JsonArray>();
  for (int i = 0; i < MAX_PEDIDOS_ATIVOS; i++) {
    if (listaDePedidos[i].status != "vazio") {
```

```
      JsonObject p = array.createNestedObject();
      p["numero"] = listaDePedidos[i].numero;
      p["nomeCliente"] = listaDePedidos[i].nomeCliente;
      p["status"] = listaDePedidos[i].status;
      p["descricao"] = listaDePedidos[i].descricao;
      p["timestamp"] = listaDePedidos[i].timestampAtualizacao; // Envia o tempo de
ATUALIZAÇÃO
      p["urgente"] = listaDePedidos[i].urgente;
    }
  }
  String output;
  serializeJson(doc, output);
  server.send(200, "application/json", output);
}

void handleApiPainelCliente() {
  StaticJsonDocument<4096> doc;
  JsonObject root = doc.to<JsonObject>();
  JsonArray preparo = root.createNestedArray("preparando");
  JsonArray prontos = root.createNestedArray("prontos");

  for (int i = 0; i < MAX_PEDIDOS_ATIVOS; i++) {
    String status = listaDePedidos[i].status;
    if (status == "preparando") {
      JsonObject p = preparo.createNestedObject();
      p["numero"] = listaDePedidos[i].numero;
      p["nomeCliente"] = listaDePedidos[i].nomeCliente;
      p["status"] = status;
      p["urgente"] = listaDePedidos[i].urgente;
    }
    else if (status == "pronto" || status == "aguardando_busca") {
      JsonObject p = prontos.createNestedObject();
      p["numero"] = listaDePedidos[i].numero;
      p["nomeCliente"] = listaDePedidos[i].nomeCliente;
      p["status"] = status;
      p["urgente"] = listaDePedidos[i].urgente;
    }
  }
  String output;
  serializeJson(doc, output);
  server.send(200, "application/json", output);
}

void handleApiHistorico() {
  StaticJsonDocument<4096> doc;
  JsonArray array = doc.to<JsonArray>();
  for (int i = 0; i < MAX_HISTORICO_ENTREGUES; i++) {
    if (historicoDeEntregues[i].status != "vazio") {
      JsonObject p = array.createNestedObject();
      p["numero"] = historicoDeEntregues[i].numero;
      p["nomeCliente"] = historicoDeEntregues[i].nomeCliente;
      p["descricao"] = historicoDeEntregues[i].descricao;
      p["timestamp"] = historicoDeEntregues[i].timestampAtualizacao; // Envia o tempo
de ATUALIZAÇÃO
      p["urgente"] = historicoDeEntregues[i].urgente;
```

```
      p["tempoPreparo"] = historicoDeEntregues[i].tempoPreparo; // Envia o tempo de
preparo
    }
  }
  String output;
  serializeJson(doc, output);
  server.send(200, "application/json", output);
}

// ATUALIZADO: API para o Relatório de Tempo de Preparo
void handleApiRelatorio() {
  StaticJsonDocument<1024> doc;

  unsigned long totalTempo = 0;
  int totalPedidos = 0;
  unsigned long minTempo = 0;
  unsigned long maxTempo = 0;

  for (int i = 0; i < MAX_HISTORICO_ENTREGUES; i++) {
    if (historicoDeEntregues[i].status == "entregue" &&
historicoDeEntregues[i].tempoPreparo > 0) {
      unsigned long tempo = historicoDeEntregues[i].tempoPreparo;

      totalTempo += tempo;
      totalPedidos++;

      if (minTempo == 0 || tempo < minTempo) {
        minTempo = tempo;
      }
      if (tempo > maxTempo) {
        maxTempo = tempo;
      }
    }
  }

  unsigned long mediaTempo = 0;
  if (totalPedidos > 0) {
    mediaTempo = totalTempo / totalPedidos;
  }

  doc["totalPedidos"] = totalPedidos;
  doc["mediaSegundos"] = mediaTempo / 1000;
  doc["minSegundos"] = minTempo / 1000;
  doc["maxSegundos"] = maxTempo / 1000;

  String output;
  serializeJson(doc, output);
  server.send(200, "application/json", output);
}


void handleNotFound() {
  server.send(404, "text/plain", "404: Pagina nao encontrada.");
}
```

```
// ================================================
// SETUP e LOOP PRINCIPAL
// ================================================
void setup() {
  Serial.begin(115200);
  delay(1000);

  // --- Setup do Painel Físico (OLED) ---
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(BTN_REPETIR, INPUT_PULLUP);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("Display OLED falhou!");
    while (1);
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(15, 20);
  display.println("SISTEMA KDS");
  display.setCursor(10, 35);
  display.println("Iniciando...");
  display.display();

  inicializarArraysDePedidos();

  conectarWiFi(); // Conecta ao WiFi

  // --- Setup do Servidor Web ---
  server.on("/", HTTP_GET, handleCozinha);
  server.on("/enviar", HTTP_POST, handleEnviarNovoPedido);
  server.on("/dashboard", HTTP_GET, handleDashboardCozinha);
  server.on("/atualizar", HTTP_POST, handleAtualizarStatus);
  server.on("/painel", HTTP_GET, handlePainelCliente);
  server.on("/historico", HTTP_GET, handleHistoricoPage);
  server.on("/relatorio", HTTP_GET, handleRelatorioPage);

  server.on("/api/pedidos", HTTP_GET, handleApiPedidosAtivos);
  server.on("/api/painel", HTTP_GET, handleApiPainelCliente);
  server.on("/api/historico", HTTP_GET, handleApiHistorico);
  server.on("/api/relatorio", HTTP_GET, handleApiRelatorio);

  server.onNotFound(handleNotFound);

  server.begin();

  Serial.println("\n=== SERVIDOR WEB INICIADO ===");
  Serial.println("Acesse o IP acima no navegador.");
  Serial.println("==============================\n");

  Serial.println("\n=== PAINEL OLED PRONTO ===");
  Serial.println("Aguardando pedidos...");
  Serial.println("Botao Amarelo (PIN 13) = Repetir Alarme\n");

  atualizarDisplayOLED(); // Mostra o display inicial
```

```
}

void loop() {
  server.handleClient();

  static unsigned long ultimaAtualizacaoOLED = 0;
  if (millis() - ultimaAtualizacaoOLED > 500) {
    atualizarDisplayOLED();
    ultimaAtualizacaoOLED = millis();
  }

  bool btnAtual = digitalRead(BTN_REPETIR);
  if (btnAnteriorPainel == HIGH && btnAtual == LOW) {
    delay(50);
    repetirUltimoPedidoOLED();
  }
  btnAnteriorPainel = btnAtual;

  delay(10);
}
```