COMANDO VIA DTMF

Anderson Henrique, Cayo Coelho, Levi Nascimento, Isaac Santos, Matheus Meneses, Prof. Me. Pedro Adolfo Gallani
Faculdade de Tecnologia de São Bernardo do Campo
Anderson.silva389@fatec.sp.gov.br, cayo.coelho@fatec.sp.gov.br, levi.nascimento@fatec.sp.gov.br, isaac.santos6@fatec.sp.gov.br matheus.meneses4@fatec.sp.gov.br, pedro.galani@fatec.sp.gov.br

RESUMO: Este trabalho aborda o desenvolvimento de um sistema para controle remoto de dispositivos elétricos utilizando a tecnologia Dual Tone Multi-Frequency (DTMF). O estudo foca na criação de uma solução prática, segura e de baixo custo para automação. A proposta consiste em um sistema capaz de receber comandos por chamadas telefônicas, decodificar os tons DTMF e acionar cargas elétricas à distância. Para gerenciar todas as funções e garantir a confiabilidade operacional, o microcontrolador ATmega328P da Atmel atua como unidade central de processamento. Este componente, utilizado na plataforma Arduino Uno, é responsável pela lógica de segurança, acionamento dos relés, leitura do sensor de corrente ACS712 e geração do *feedback* sonoro. Após enviar o comando ao relé, o ATmega328P monitora o ACS712 e, ao confirmar a passagem de corrente, comanda o módulo DFPlayer Mini para gerar a mensagem de voz de confirmação ao usuário. A metodologia envolveu pesquisa teórica, desenvolvimento de *firmware* único e testes experimentais, validando a eficácia do *loop* de confirmação de carga ativa.

Palavras-chave: DTMF; Automação; Microcontrolador; Relés; Controle remoto.

ABSTRACT: This work addresses the development of a system for remote control of electrical devices using Dual Tone Multi-Frequency (DTMF) technology. The study focuses on creating a practical, secure, and low-cost solution for automation. The proposal consists of a system capable of receiving commands via phone calls, decoding DTMF tones, and actuating electrical loads remotely. To manage all functions and ensure operational reliability, the ATmega328P microcontroller from Atmel acts as the central processing unit. This component, utilized in the Arduino Uno platform, is responsible for the security logic, relay actuation, reading the ACS712 current sensor, and generating audible *feedback*. After sending the command to the relay, the ATmega328P monitors the ACS712 and, upon confirming current flow, commands the DFPlayer Mini module to generate a voice message for user confirmation. The methodology involved theoretical research, development of a single *firmware*, and experimental tests, validating the effectiveness of the active load confirmation loop.

Keywords: DTMF; Automation; Microcontroller; Relays; Remote control.

1. Introdução

A automação de equipamentos elétricos tem se consolidado como uma área em expansão, impulsionada pela busca por praticidade, eficiência e segurança no controle de dispositivos residenciais e industriais. Nesse cenário, tecnologias de baixo custo e fácil implementação ganham destaque, especialmente quando oferecem confiabilidade e compatibilidade com recursos amplamente disponíveis.

A tecnologia *Dual Tone Multi-Frequency* (DTMF), conhecida popularmente pela sua aplicação em sistemas telefônicos, é um exemplo de solução que, mesmo após décadas de uso, mantém-se relevante nos dias atuais. Por meio da geração de dois tons em diferentes frequências sonoros, o DTMF possibilita a transmissão de comandos de forma simples, confiável e imune a ruídos, o que torna aplicável em diferentes contextos de automação.

Embora haja soluções modernas, como a comunicação via internet e aplicativos móveis, nem sempre elas são viáveis em área com restrições de infraestrutura ou custo. Além disso, conforme dados da Agência Brasil (2024), apenas 22% da população brasileira com mais de 10 anos possui condições satisfatórias de conectividade, evidenciando a limitação de acesso à internet de qualidade em diversas regiões do país. Nesse sentindo, o DTMF se destaca como uma alternativa prática para acionamento remoto de cargas, utilizando recursos acessíveis como telefones celulares e rádios comunicadores.

O presente trabalho tem como objetivo desenvolver um sistema de acionamento de cargas elétricas baseado em DTMF, integrado a um microcontrolador para processamento dos sinais. A proposta busca demonstrar a viabilidade de soluções simples e seguras de automação, explorando o potencial do DTMF como ferramenta de baixo custo e ampla aplicabilidade em diferentes cenários.

2. Fundamentação teórica

Historicamente, a comunicação telefônica era realizada por meio dos chamados telefones de disco, cujo funcionamento baseava-se em relés eletromecânicos. Neste sistema, os números eram discados e sempre que se escolhia algum número, ocorria a interrupção na conexão de corrente contínua que era utilizada para fazer a comunicação, fazendo com que os relés de passo atuassem e identificasse o número desejado.

Com o passar dos anos e com o avanço da tecnologia, a discagem por pulso foi substituída pela discagem por tom, ou DTMF (Dual Tone Multi-Frequency). Este sistema representa um avanço significativo, pois, em vez de interrupções de corrente, a discagem é realizada através de um conjunto de frequências sonoras distintas. Cada número ou símbolo

pressionado no teclado gera a combinação de duas frequências — uma da linha (baixa) e uma da coluna (alta) —, o que permite um método de transmissão de dados mais rápido, eficiente e com maior imunidade a ruídos, consolidando-se como a tecnologia padrão para a comunicação e sistemas de controle telefônico.

2.1 Tecnologia DTMF (*Dual Tone Multi-Frequency*)

O DTMF (*Dual Tone Multi Frequency*), que pode ser traduzido como frequências múltiplas, consiste em um dispositivo que processa chamadas telefônicas e é utilizado no controle e automação de diversos equipamentos existentes, sendo sua função utilizar os sons emitidos ao pressionar uma tecla, para detectar quais números foram discados.

De acordo com Medeiros (2006), os tons do *"Dual Tone"* consistem em pares de frequências combinadas especificamente para identificar as teclas pressionadas no teclado telefônico. Cada tecla gera um par único de frequências conforme apresentado na Tabela 1.

Tabela 1 – Frequências dos tons DTMF associados às teclas do teclado telefônico

		Frequências superiores (Hz)			
		1209	1336	1477	1633
Frequências inferiores (Hz)	697	1	2	3	A
	770	4	5	6	B
	852	7	8	9	C
	941	*	0	#	D

Fonte: Virtual-Call (2025).

Essas frequências são divididas em dois grupos distintos, definidos como grupo de frequências baixas (697, 770, 852 e 941 Hz) e grupo de frequências altas (1209, 1336, 1477 e 1633 Hz), sendo a combinação única destas duas frequências que identifica cada tecla (Medeiros, 2006).

Sendo assim, cada número que é selecionado na tecla do telefone emite duas frequências diferentes, o que possibilita sua identificação e, subsequentemente, a tomada de decisão. Em resumo quando é feita uma chamada para um call center e pede-se para digitar a opção desejada no atendimento, esta é escolhida pelo teclado do aparelho telefônico; o sistema, devido a isso, entende o que deve ser feito posteriormente.

Tais sinais são amplamente utilizados em sistemas de comunicação por telefone, como resposta interativa por voz (IVR), automação residencial e industrial, controle remoto e segurança. Uma característica importante do sistema DTMF é a sua imunidade a ruídos

externos, proporcionando maior confiabilidade na comunicação (Virtual-call, 2023).

Para que o sistema automatizado possa interpretar os sinais sonoros DTMF, é utilizado o circuito integrado HT9170. Trata-se de um decodificador específico para sinais DTMF, tendo como função principal interpretar os sinais sonoros recebidos, convertendo-os em sinais digitais que podem ser facilmente processados por microcontroladores ou outros dispositivos digitais. O HT9170 realiza essa tarefa por meio de filtros passa-banda integrados, que separam as frequências altas e baixas dos tons recebidos, gerando um código digital único para cada tecla pressionada (Holtek, 2016).

Adicionalmente, o HT9170 dispõe de uma saída paralela com quatro bits, permitindo conexão direta com entradas digitais de microcontroladores. Esse recurso simplifica significativamente o projeto e implementação de sistemas de automação e controle remoto.

Para o processamento dos sinais decodificados e execução das ações de controle, o projeto utiliza o microcontrolador ATmega328P, fabricado pela Atmel (Microchip Technology). Este componente é o microcontrolador central de todo o sistema. Ele utiliza uma arquitetura RISC aprimorada de 8 bits e possui um conjunto de 131 instruções poderosas, com a maioria das operações executadas em um único ciclo de *clock*, o que garante uma excelente performance em tempo real. O ATmega328P é o chip fundamental da plataforma de prototipagem Arduino Uno, utilizada neste trabalho devido à sua vasta comunidade de desenvolvimento e facilidade de integração.

O ATmega328P possui 32 KB de memória Flash para programa, 2 KB de SRAM e 1 KB de EEPROM. Para a interface com o sistema externo, conta com 23 pinos de entrada e saída digitais programáveis, além de integrar um Conversor Analógico-Digital (ADC) de 10 bits com 6 canais, permitindo a leitura precisa de sensores analógicos. Possui ainda periféricos essenciais como *timers*, interface USART serial (fundamental para o DFPlayer Mini) e interface SPI.

Dessa forma, no projeto proposto, o ATmega328P atua como a unidade principal de controle (núcleo DTMF) e de *feedback*, centralizando todas as operações. Sua utilização se justifica pela facilidade de programação, pela capacidade de consolidar a lógica de decodificação, acionamento, leitura do sensor ACS712 via seu ADC e gerenciamento do *feedback* sonoro.

O acionamento físico das cargas elétricas controladas é feito por meio de relés eletromecânicos. Relés são componentes eletromecânicos fundamentais em sistemas de automação, especialmente quando é necessário o controle de cargas elétricas de potência superior à capacidade das saídas digitais dos microcontroladores. Funcionam por meio da energização de uma bobina que produz um campo magnético capaz de atrair um contato móvel, fechando ou abrindo circuitos elétricos externos (Petruzella, 2015).

Neste projeto, a saída digital do microcontrolador energiza a bobina do relé, permitindo a ativação ou desativação segura e eficaz de dispositivos elétricos variados, oferecendo praticidade e segurança ao usuário.

2.2. Monitoramento e Feedback

Para garantir a eficácia operacional do sistema e fornecer *feedback* ao usuário, a arquitetura do projeto utiliza o microcontrolador central para processar o controle e o monitoramento. O sistema utiliza o sensor de corrente ACS712, que é baseado no Efeito Hall e capaz de medir correntes AC ou DC de forma não invasiva, convertendo a corrente em um sinal de tensão analógica proporcional. O sensor é posicionado na saída de potência do relé, permitindo a confirmação de que a corrente elétrica foi estabelecida após o comando de acionamento.

Essa leitura de confirmação é processada diretamente pelo microcontrolador ATmega328P (chip principal da plataforma Arduino Uno). O ATmega328P utiliza seu Conversor Analógico-Digital (ADC) integrado para receber e interpretar o sinal de tensão do ACS712. Uma vez que o microcontrolador central confirme a presença de corrente (carga ativa), ele é o responsável por comandar, via comunicação serial, o módulo DFPlayer Mini para a geração do áudio. O DFPlayer Mini é utilizado para armazenar e reproduzir mensagens de voz pré-gravadas (ex: "Carga 1 Ativada"), garantindo que o *feedback* seja claro e humanizado. Dessa forma, o Arduino Uno, através do chip ATmega328P, centraliza a decodificação DTMF, o controle dos relés e o *feedback*, garantindo que a mensagem sonora só seja enviada ao usuário após a confirmação física de acionamento da carga.

Quanto à segurança da comunicação, o sistema projetado utiliza um aparelho celular conectado via conector P2 ao decodificador HT9170. Ao receber uma chamada e estabelecer conexão, exige autenticação por senha para garantir segurança e evitar acessos não autorizados. Após três tentativas incorretas, o sistema envia uma mensagem de erro, sendo bloqueado temporariamente, oferecendo uma camada adicional de proteção contra uso indevido (Stallings, 2018). Essa abordagem é essencial em aplicações práticas, garantindo ao usuário controle seguro e confiável sobre seus equipamentos residenciais ou industriais.

3. Metodologia

3.1. Caracterização da Pesquisa

O presente trabalho caracteriza-se como uma pesquisa aplicada, pois o objetivo central é o desenvolvimento e implementação de uma solução tecnológica para o controle remoto de dispositivos elétricos. O foco deste tipo de pesquisa é a geração de conhecimento no que tange

a aplicação prática, sendo fundamental para o avanço em sistemas de automação, especialmente diante do conceito de acionamento de cargas a distância.

3.2. Procedimentos Metodológicos

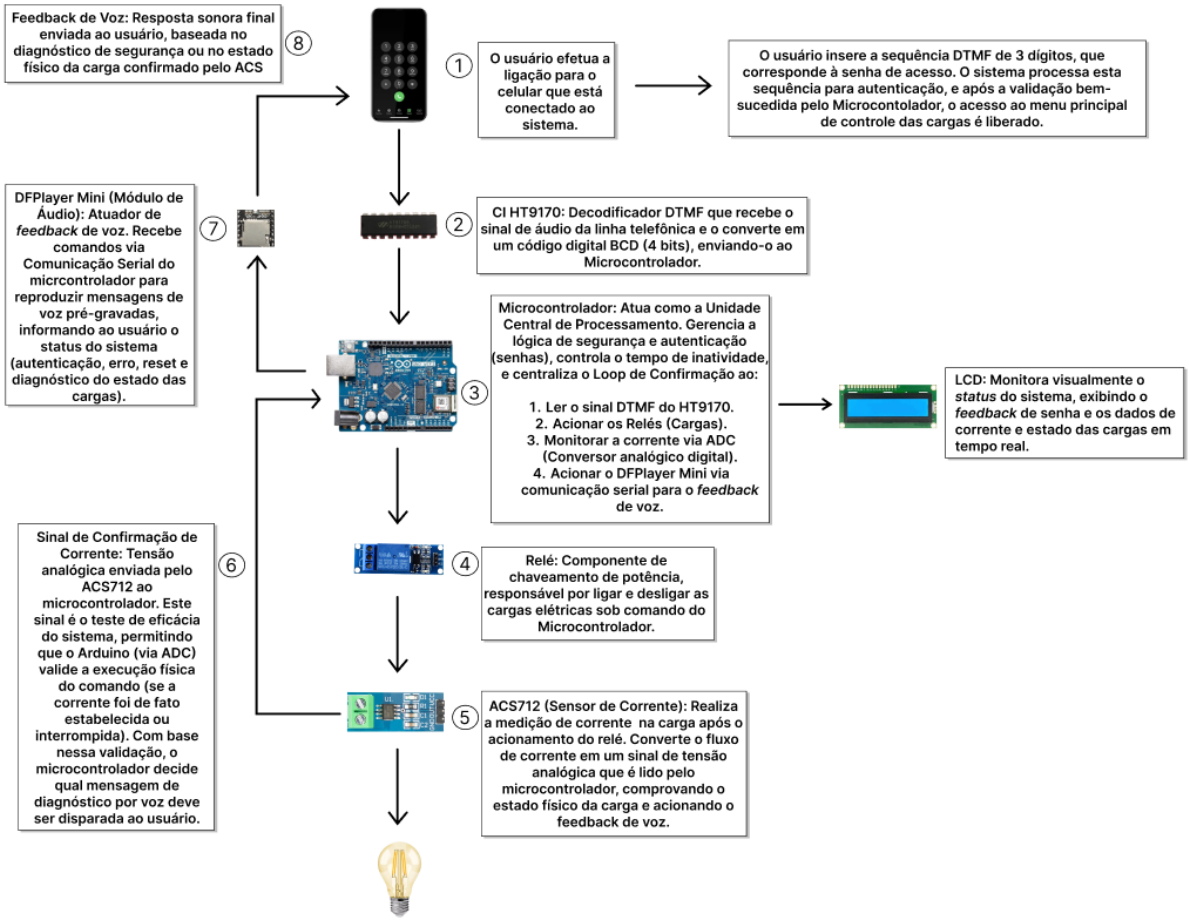
O presente trabalho caracteriza-se como uma pesquisa aplicada, voltada para o desenvolvimento e implementação de uma solução tecnológica para o controle remoto de dispositivos elétricos, sendo fundamental para o avanço em sistemas de automação que exigem acionamento de cargas a distância e com a garantia de acionamento. A execução do sistema de Comando via DTMF foi estruturada em uma abordagem de desenvolvimento de protótipo, compreendendo etapas sequenciais adaptadas para a implementação de uma arquitetura de controle centralizada e robusta. A fase inicial consistiu na Fundamentação Teórica, com uma abrangente pesquisa bibliográfica focada na tecnologia DTMF, suas aplicações em sistemas de automação e na análise detalhada dos *datasheets* de todos os componentes eletrônicos essenciais, incluindo o Circuito Integrado (CI) decodificador HT9170, o microcontrolador Arduino Uno, o sensor de corrente ACS712 e o módulo de áudio DFPlayer Mini.

Simultaneamente, ocorreu a Definição de Especificações, na qual foram estabelecidos os requisitos operacionais do sistema, como o número de cargas elétricas a serem controladas, os critérios de segurança, através de um protocolo de autenticação por senha via DTMF, e a especificação de *feedback*, definindo a necessidade de monitoramento do estado de carga e geração de uma resposta sonora humanizada. Com as especificações definidas, iniciou-se a fase de projeto. O Projeto Lógico e Simulação envolveu a elaboração do esquema elétrico detalhado do circuito, no qual todos os periféricos (HT9170, ACS712, DFPlayer Mini) foram conectados diretamente ao Arduino Uno. Para a validação prévia da lógica de *hardware* e da configuração dos pinos do microcontrolador, utilizou-se um *software* de simulação, permitindo o estudo do comportamento do sistema antes da montagem física.

Em paralelo, deu-se o Desenvolvimento do *Firmware* (Figura 2). A codificação do programa principal para o Arduino Uno foi realizada em Linguagem C, utilizando a IDE do Arduino para facilitar o desenvolvimento. O *firmware* foi centralizado, sendo responsável por quatro funções principais: decodificação DTMF, rotina de autenticação, controle dos relés e gerenciamento do *feedback*. O *firmware* foi projetado para utilizar o Conversor Analógico-Digital (ADC) do Arduino Uno para ler o sinal do ACS712 e, em seguida, comandar o módulo DFPlayer Mini via comunicação serial para reproduzir a mensagem de *feedback*. A etapa de Montagem e Protótipo abrangeu a seleção e aquisição dos componentes finais. Após testes iniciais de funcionalidade em *protoboard*, o circuito foi montado de forma definitiva em uma placa de circuito impresso (PCI), integrando todos os componentes ao Arduino Uno.

A Coleta de Dados foi efetuada de forma prática e experimental, através do envio de comandos DTMF sequenciais realizados via chamada telefônica. Os dados primários coletados e avaliados incluíram a resposta do sistema verificada pela leitura digital do HT9170, a leitura analógica de corrente do ACS712 pelo Arduino, a ação física dos relés, e a emissão da mensagem de voz pelo DFPlayer Mini como confirmação de sucesso. A etapa final consistiu na validação do protótipo e na análise dos resultados. A Análise de Dados concentrou-se na avaliação da confiabilidade, do tempo de resposta e, principalmente, na eficácia do *loop* de confirmação, verificando se a mensagem sonora só era reproduzida após a confirmação física de corrente pelo sensor ACS712 lida pelo Arduino Uno, comprovando o atendimento integral aos objetivos propostos pelo projeto.

Figura 2 – Diagrama de blocos



Fonte: Autores (2025).

4. Desenvolvimento do projeto

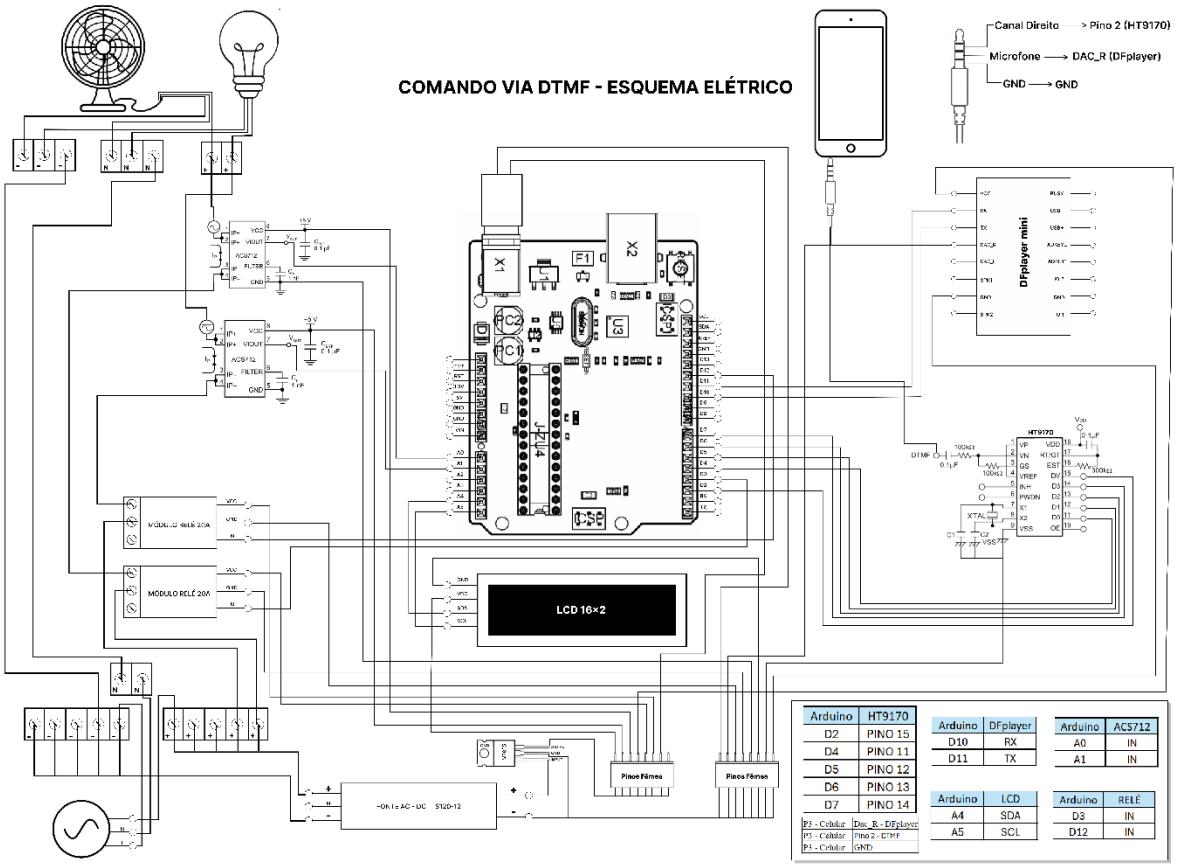
O desenvolvimento do projeto detalha a materialização física e lógica da solução, integrando os componentes teóricos para a construção do sistema de controle via DTMF. Para atender aos requisitos de segurança e garantir o *feedback* instrumentalizado, foi adotada uma

arquitetura de controle centralizada no microcontrolador Arduino Uno. Esta unidade é responsável pela decodificação DTMF, rotinas de segurança, acionamento dos relés, leitura do sensor de corrente (ACS712) e geração da resposta de áudio (DFPlayer Mini), resultando em um sistema robusto e de *hardware* mais enxuto.

O ciclo de *feedback* é totalmente gerenciado pelo Arduino Uno: ao confirmar a presença de corrente elétrica via leitura do ACS712, o microcontrolador envia um comando serial direto ao DFPlayer Mini. O DFPlayer reproduz a mensagem de voz correspondente à carga, e o áudio é injetado na linha telefônica via conector P3. Dessa forma, a unidade central é a responsável final pela geração da confirmação sonora ao usuário, certificando que a carga foi efetivamente ligada ou desligada.

A Figura 3 apresenta o esquema elétrico do projeto.

Figura 3 – Esquema elétrico



Fonte: Autores (2025).

5. Testes e resultados

Para a validação do sistema, foram realizados testes de funcionamento, tanto do sistema para o acionamento de cargas distintas, como com a utilização do *feedback*. Contando com a utilização de vários aparelhos telefônicos, a fim de verificar a confiabilidade do reconhecimento

dos tons DTMF.

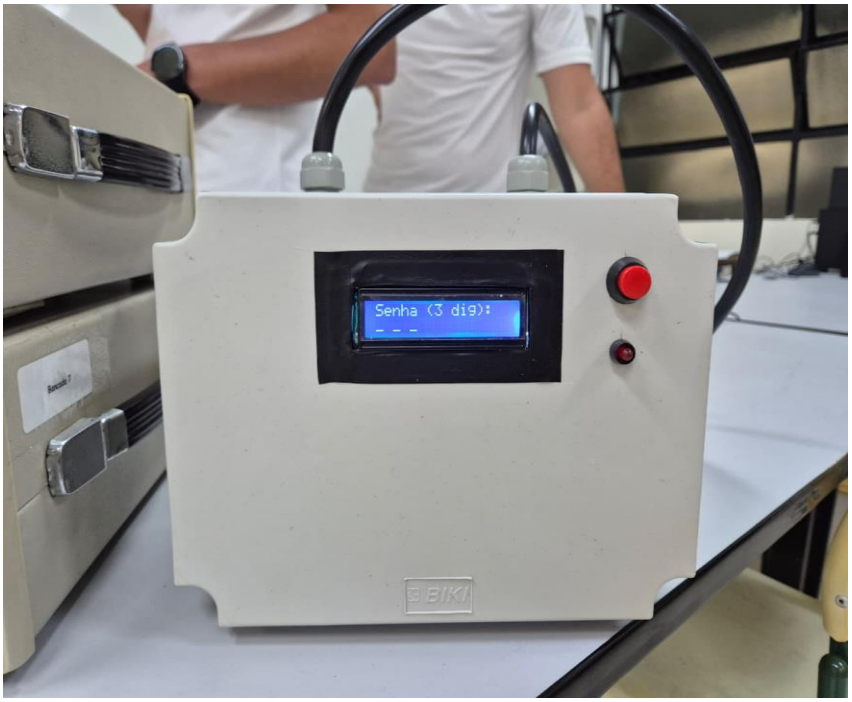
Os testes demonstraram que o circuito responde corretamente aos comandos enviados via ligação telefônica, acionando as cargas de forma precisa e gerando o retorno, informando se a carga solicitada foi ligada ou desligada corretamente. O tempo médio de resposta foi de aproximadamente 500 milissegundos a 1 segundo.

As Figuras 4 e 5 ilustram o protótipo desenvolvido.

Figura 4 – Protótipo em teste

Fonte: Autores (2025).

Figura 5 - Protótipo



Fonte: Autores (2025).

As cargas foram constantemente acompanhadas, assim como a medição feita pelo sensor de corrente, que é responsável pela precisão dos retornos ao cliente, informando o estado de cada acionamento ou desligamento. Além disso foram testados diferentes aparelhos, como: Lâmpadas de LED, esmerilhadeira angular, diodo emissor de luz, furadeira de impacto, carregadores de celular. Com isso, foi possível monitorar a corrente de cada um dos itens listados e a resposta do circuito a eles.

Observou-se, entretanto, que sinais de baixa qualidade (como em ligações com ruído) podem gerar falhas ocasionais de decodificação, sendo necessário um ambiente com certo controle sobre estes eventos.

6. Considerações finais

O projeto de acionamento remoto via DTMF atingiu o objetivo proposto de permitir o controle e a consulta do estado de cargas elétricas a partir de uma ligação telefônica convencional. O sistema apresentou funcionamento estável e de fácil operação, demonstrando a viabilidade da técnica estudada.

Como pontos fortes, destacam-se o baixo custo e a simplicidade de implementação, além da alta confiabilidade do funcionamento e de não haver fronteiras para o uso da tecnologia, tendo em vista que para operá-la basta um aparelho e sinal telefônico, trazendo a possibilidade de se encontrar em outros estados ou até mesmo fora do País. Outro ponto positivo é o produto ser totalmente isento do uso de internet para seu funcionamento, tendo em vista que em grande parte das cidades brasileiras hoje, se enfrenta como grande dificuldade, a qualidade da mesma.

Com isso, o projeto foi implementado de forma positiva e cumpriu tanto com as expectativas técnicas, quanto com o objetivo de desenvolver alternativas distintas, onde não seja necessário depender somente de um tipo de meio de comunicação, de forma eficaz e confiável.

Referências

CRISTINA, A. **DTMF VoIP**: O que é e como configurar para empresas 2025, 2024. Disponível em: https://www.google.com/amp/s/www.virtual-call.net/pt-br/blog/qual-e-dtmf-e-para-que-serve%3fhs_amp=true. Acesso em: 31 mar. 2025.

HOLTEK. **HT9170 Datasheet**. *Holtek Semiconductor Inc.*, 2016. Disponível em: https://www.holtek.com.tw/documents/10179/116711/HT9170_30v130.pdf. Acesso em: 07 abr. 2025.

HT9170 9170 Decodificador Tono Dtmf Dip Mt8870 8870 Cm8870. Disponível em: https://www.mercadolibre.com.ar/ht9170-9170-decodificador-tono-dtmf-dip-mt8870-8870-cm8870/up/MLAU296682555. Acesso em: 12 nov. 2025.

MEDEIROS, L. F. **Redes Neurais em Delphi**. 2ª ed. Florianópolis: Visual Books, 2006.

PETRUZELLA, F. D. **Automação Industrial**. 2ª ed. São Paulo: Pearson Education, 2015.

SANTOS, F. C. A. Integration of human resource management and competitive priorities of manufacturing strategy. **International Journal of Operations & Production Management**, n. 5, p. 612-628, 2000.

SMOLKA, J. R. **Smolka et catervarii**. Disponível em: https://smolkaetcaterva.blogspot.com/. Acesso em: 31 mar. 2025.

STALLINGS, W. **Segurança e Criptografia em Redes**. 7ª ed. São Paulo: Pearson, 2018.

VIRTUAL-CALL. **Tecnologia DTMF**. Disponível em: https://www.virtual-call.net/. Acesso em: 07 abr. 2025.

Apêndice A - Tutorial de operação do sistema de comando via DTMF

O presente apêndice detalha os procedimentos operacionais para a utilização do sistema de controle remoto de cargas elétricas, com o objetivo de guiar o usuário na execução de comandos via tecnologia *Dual Tone Multi-Frequency* (DTMF). O protocolo de interação é estruturado em três fases principais: Acesso e Autenticação, Controle das Cargas e Funções Especiais, descritos a seguir:

1. Acesso e Autenticação

A operação do sistema é iniciada quando o usuário efetua uma chamada para o aparelho celular conectado ao circuito. O sistema é configurado para realizar o atendimento automático da ligação, estabelecendo o canal de comunicação via áudio. Após a conexão, o usuário deve inserir a senha de acesso de três dígitos por meio do teclado do telefone. O *firmware* do microcontrolador processa o código DTMF recebido pelo decodificador HT9170 e verifica a autenticidade da sequência. Apenas após a confirmação da senha o usuário recebe a mensagem sonora de "Senha correta. Bem-vindo ao menu principal" e tem o acesso liberado para as funções de controle e monitoramento.

2. Menu Principal e Controle de Cargas

Com o acesso liberado, o sistema disponibiliza teclas para o controle direto e o monitoramento do estado físico de duas cargas (Carga 1 e Carga 2), utilizando os dígitos de 5 e 6. As funções de controle são do tipo *toggle*, alternando o estado atual da carga a cada toque:

- Tecla 1: Ativa e desativa o Relé da Carga 1. Após a ação, o sistema consulta o sensor ACS712 para diagnosticar o estado. O *feedback* de voz pode informar o sucesso da ativação ("Carga um ativada"), o sucesso do desativamento ("Carga um desativada"), ou diagnosticar uma falha ("Carga um ligada, mas sem corrente"), caso o Relé esteja acionado, mas o sensor não detecte passagem de corrente.
- Tecla 2: Ativa e desativa o Relé da Carga 2, seguindo a mesma lógica de diagnóstico e *feedback* por voz baseada na leitura do ACS712.

As funções de verificação permitem que o usuário solicite o *status* em tempo real, sem alterar o estado:

- Tecla 5: Confere o estado da Carga 1. O sistema lê o ACS712 e informa se a carga está "ativa e com corrente" ou "desativada e sem corrente".
- Tecla 6: Confere o estado da Carga 2. O sistema lê o ACS712 e informa o estado atual da carga.

3. Função Especial: Modo Reset

A função de *reset* permite desligar todas as cargas simultaneamente e retornar o sistema à tela de autenticação para um novo ciclo operacional, reforçando a segurança e estabilidade.

- Tecla 9: Ao digitar esta tecla, o sistema entra no Modo Reset, emitindo o *feedback* "Modo reset ativado. Digite 1 para confirmar o reset do sistema e desligar todas as cargas, ou 2 para cancelar a ação."
    - Confirmação (9 + 1): Desliga ambos os relés e retorna o sistema ao *prompt* de senha inicial, solicitando uma nova autenticação.
    - Cancelamento (9 + 2): Cancela a operação e retorna o usuário diretamente ao Menu Principal sem alterar o estado das cargas.

Apêndice B - Tutorial: configuração do atendimento automático em celulares (via P2)

Este guia visa configurar o celular que serve como interface DTMF para atender chamadas automaticamente, condição essencial para o funcionamento do sistema.

Pré-requisito Fundamental: Para que o recurso de atendimento automático funcione na maioria dos dispositivos Android, o sistema operacional exige a detecção de um fone de ouvido. Desse modo, o cabo P3 (que conecta o celular ao seu circuito HT9170 e injeta o *feedback* de voz) deve estar plugado e corretamente reconhecido pelo aparelho antes de iniciar a configuração. A presença do conector P3 simula um *headset*, o que é crucial para que o sistema libere a funcionalidade de atendimento automático.

Abordagem 1: Usando Configurações Nativas do Fabricante (Mais Recomendada)

Muitas interfaces de usuário (UIs) personalizadas do Android incluem esta opção, geralmente oculta nas configurações de acessibilidade ou do aplicativo de Telefone. Siga os passos a seguir para configurar a função de atendimento automático, utilizando as configurações nativas do aparelho:

Acesse as Configurações: Abra o aplicativo "Configurações" (Settings) do seu celular.

1. Localize as Configurações de Chamada:
    a. Procure por "Acessibilidade" (Accessibility) e, dentro, procure por *"Interação e Destreza"* ou *"Atender e Encerrar Chamadas"*.
    b. OU Procure diretamente nas configurações do "Aplicativo de Telefone" (Call Settings).
2. Habilite o Atendimento Automático:
    a. Encontre a opção "Atendimento Automático" (Auto Answer) ou similar.
    b. Condição: Geralmente, você terá que selecionar a opção: "Atender automaticamente quando um fone de ouvido ou dispositivo Bluetooth estiver conectado."
3. Defina o Atraso (Opcional): Muitas vezes, você pode definir um pequeno atraso (ex: 2 ou 3 segundos) antes do atendimento. Defina para um valor baixo (2s) para agilizar o processo.

4. Teste: Peça a alguém para ligar para o celular do sistema enquanto o cabo P2 está conectado. O celular deve atender sem que você precise tocar na tela.

Apêndice C – Código fonte

O código fonte representa a implementação lógica (*firmware*) desenvolvida para o microcontrolador Arduino Uno, que atua como unidade central de processamento do sistema de controle via DTMF. O *firmware* é responsável pela leitura e decodificação do sinal DTMF, pela execução da lógica de segurança, pelo gerenciamento do tempo de inatividade, pelo processamento dos dados do sensor ACS712 e pelo controle serial do módulo DFPlayer Mini para o *feedback* de voz.

Para garantir a transparência do projeto e permitir sua reprodutibilidade, o código integral é apresentado a seguir. Seu desenvolvimento exigiu a inclusão de bibliotecas específicas que facilitam a comunicação com os periféricos (LCD I2C e DFPlayer Mini) e a execução de funções avançadas. As bibliotecas de terceiros necessárias estão disponíveis para instalação e gerenciamento através do Gerenciador de Bibliotecas (Library Manager) da IDE do Arduino (versão 1.8.2, utilizada no projeto), enquanto outras são nativas da plataforma.

```
/*
  Projeto: Controle por DTMF com leitura de corrente (ACS712), DFPlayer e LCD 16x2
  Placa: Arduino UNO

  Funcionalidades principais:
  - Senha DTMF: 1-2-3 para liberar o sistema.
  - Dois canais de carga (C1 e C2) com relé + medição de corrente (ACS712).
  - Histerese de corrente:
     * Estado ON acima de 80 mA
     * Estado OFF abaixo de 50 mA
  - Ao ligar relé:
     1) Liga o relé.
     2) Espera ~500 ms.
     3) Mede corrente corrigida.
        - Se I >= 0,08 A → áudio "Carga X ligada".
        - Se I < 0,08 A → áudio "Carga X sem corrente".
  - Ao desligar → áudio "Carga X desligada".

  Mapeamento das teclas no modo UNLOCKED:
  - Tecla 1 → liga/desliga CARGA 1 (C1).
  - Tecla 2 → liga/desliga CARGA 2 (C2).
  - Tecla 3 → consulta estado de C1 (ligada/desligada) via histerese (on1).
  - Tecla 4 → consulta estado de C2 (ligada/desligada) via histerese (on2).
  - Tecla 5 → consulta CORRENTE da C1.
  - Tecla 6 → consulta CORRENTE da C2.
  - Tecla 9 → inicia fluxo de RESET (confirma com 1, cancela com 2).
*/
```

```
#include <Wire.h> // disponível na biblioteca da IDE do aruino versão 1.8.2
#include <LiquidCrystal_I2C.h> // disponível na biblioteca da IDE do aruino versão 1.8.2
#include <SoftwareSerial.h> // disponível na biblioteca da IDE do aruino versão 1.8.2
#include <DFRobotDFPlayerMini.h> // disponível na biblioteca da IDE do aruino versão 1.8.2
#include <math.h> // disponível na biblioteca da IDE do aruino versão 1.8.2
#include <string.h> // disponível na biblioteca da IDE do aruino versão 1.8.2

#define N_CH 2    // número de canais usados neste UNO

/*====================== LCD ======================*/
LiquidCrystal_I2C lcd(0x27, 16, 2);

/*==================== DFPlayer ====================*/
/*
  Ligações:
  - DFPlayer TX → D10 (RX do Arduino / SoftwareSerial)
  - DFPlayer RX → D11 (TX do Arduino / SoftwareSerial) [ideal: resistor ~1k em série]
  - GND DFPlayer → GND Arduino
  - VCC DFPlayer → 5V Arduino
*/

const uint8_t PIN_DF_RX  = 10;  // RX Arduino  <- TX DFPlayer
const uint8_t PIN_DF_TX  = 11;  // TX Arduino  -> RX DFPlayer

SoftwareSerial dfSerial(PIN_DF_RX, PIN_DF_TX);
DFRobotDFPlayerMini dfp;
bool df_ok = false;

/*
  MAPEAMENTO REAL NO DFPLAYER (ajuste se necessário):

  0001.mp3 -> 1  = (livre ou outro uso)
  0002.mp3 -> 2  = Carga 1 ligada   (exemplo)
  0003.mp3 -> 3  = Carga 1 desligada (exemplo)
  0004.mp3 -> 4  = Sistema inativo
  0005.mp3 -> 5  = Sistema resetado
  0006.mp3 -> 6  = Atenção, o sistema será reiniciado
  0007.mp3 -> 7  = Reset não autorizado
*/

// PROMPTS gerais (ajustados aos números que você está usando)
#define TRK_AUTH_OK        8   // "Senha correta"       (conforme seu SD)
#define TRK_AUTH_FAIL      9   // "Senha incorreta"
#define TRK_SYSTEM_ACTIVE  10  // "Sistema ativo"
#define TRK_SYSTEM_INACTIVE 1   // 0004.mp3 = "Sistema inativo"

// Trilhas de reset (novas)
#define TRK_RESET_DONE  11   // 0005.mp3 = "Sistema resetado"
#define TRK_RESET_WARN  13   // 0006.mp3 = "Atencao, o sistema sera reiniciado"
#define TRK_RESET_DENY  12   // 0007.mp3 = "Reset nao autorizado"

// não usados por enquanto
#define TRK_BOOT      0
#define TRK_WAIT_PIN  0
#define TRK_MENU      0
#define TRK_RESET     0
```

```
#define TRK_INVALID    0
#define TRK_IDLE_TO    0
#define TRK_CURR_OK    0
#define TRK_CURR_OFF   0

// Trilhas por canal  (mantenha como você ajustou no SD)
const uint16_t TRK_ON[N_CH]  = { 2, 4 };   // C1 ligada, C2 ligada
const uint16_t TRK_OFF[N_CH] = { 3, 5 };   // C1 desligada, C2 desligada

// "Sem corrente"
const uint16_t TRK_NC[N_CH]  = { 6, 7 };   // C1 sem corrente, C2 sem corrente

/*====================== DTMF (HT9170B) ============*/
/*
  Ligações típicas:
  - D0 → D4
  - D1 → D5
  - D2 → D6
  - D3 → D7
  - DV → D2 (INT0)
*/

#define DTMF_D0 4
#define DTMF_D1 5
#define DTMF_D2 6
#define DTMF_D3 7
#define DTMF_DV 2

volatile bool     dvRiseFlag = false;
volatile uint8_t  lastNibble = 0;

/*====================== Pinos diversos ===========*/
const uint8_t PINO_BOTAO = 8;   // botão recalib. (puxado para GND)
const uint8_t LED_HB     = 13;  // LED onboard

/*================== Relés ==================*/
const uint8_t RELAY1_PIN = 3;   // C1
const uint8_t RELAY2_PIN = 12;  // C2
const bool REL_ACTIVE_LOW = true; // LOW = liga

bool relay1On = false;
bool relay2On = false;

/*==================== Sensores ====================*/
const uint8_t PINO_ACS1 = A0;
const uint8_t PINO_ACS2 = A1;
const float   Vref_ADC  = 5.000;

// Ajustar sensibilidade conforme modelo do ACS712
//  - 5A  → 0.185 V/A
//  - 20A → 0.100 V/A
//  - 30A → 0.066 V/A
const float   sensib1   = 0.100;   // exemplo: 20 A
const float   sensib2   = 0.100;

uint16_t      N_RMS     = 400;
const uint16_t DELAY_MS = 1;
```

```
// Histerese/filtros
float TH_ON_A       = 0.080f;      // 80 mA
float TH_OFF_A      = 0.050f;      // 50 mA
float I_MIN_DISPLAY = 0.020f;      // 20 mA
const float ALPHA_EMA      = 0.20f;
const float TH_AUDIO_ON_A  = 0.080f;

// Offsets
uint16_t offset1 = 512;
uint16_t offset2 = 512;
float calibOff1_A = 0.0f;
float calibOff2_A = 0.0f;

// Estado lógico de corrente (pelo ACS + histerese)
bool  on1 = false, on2 = false;
float ema1 = 0.0f, ema2 = 0.0f;
bool  emaInit1 = false, emaInit2 = false;

// Botão
bool lastBtn = HIGH;
unsigned long btnDownAt = 0;

/*==================== Senha ====================*/
enum Mode { LOCKED, UNLOCKED };
Mode mode = LOCKED;

const uint8_t PASS_LEN = 3;
// Senha DTMF: 1-2-3
const uint8_t PASS_SEQ[PASS_LEN] = {1,2,3};
uint8_t pass_buf[PASS_LEN];
uint8_t pass_pos = 0;

/*===== Inatividade & cooldown =====*/
const unsigned long INACTIVITY_MS   = 60000;
unsigned long lastActivityAt        = 0;
const unsigned long KEY_COOLDOWN_MS = 300;
unsigned long lastKeyAt             = 0;

/*===== Delay para verificação de corrente =====*/
const unsigned long VERIFY_DELAY_MS = 500;

/*===== Estado de reset pendente =====*/
bool resetPending = false;   // true = aguardando tecla de confirmação (1 = sim, 2 = não)

/*================= PROTÓTIPOS =================*/
void initDF();
void fala(uint16_t trk);
void falaC1On();
void falaC1Off();
void falaC2On();
void falaC2Off();

uint16_t medirOffset(uint8_t pino, uint16_t N, uint16_t atraso_ms);
float    medirIrms_A(uint8_t pino, float sensib, uint16_t N, uint16_t atraso_ms, uint16_t zero);
float    aplicaEMA(float x, float &y, bool &inited);
void     recalibrar(bool showLCD);
```

```
void    lcdShowLocked();
void    lcdMsg(const char* msg, uint16_t ms);
void    printLinha2Canais(float I1, bool s1, float I2, bool s2);

int     mapNibbleToKey(uint8_t nib);
int     dtmfReadKey();
void    handleLockedKey(int k);
void    handleUnlockedKey(int k);

void    applyRelay(uint8_t pin, bool on);
void    toggleRelay1();
void    toggleRelay2();

float   medirCorrenteCorrigida_C1();
float   medirCorrenteCorrigida_C2();

// Reset geral (usado por tecla 9 + 1)
void    resetSistema();

/*====================== SETUP ====================*/
void setup() {
  // Serial.begin(9600); // se quiser debug

  pinMode(LED_HB, OUTPUT);

  pinMode(PINO_BOTAO, INPUT_PULLUP);

  pinMode(RELAY1_PIN, OUTPUT);
  pinMode(RELAY2_PIN, OUTPUT);
  applyRelay(RELAY1_PIN, false);
  applyRelay(RELAY2_PIN, false);

  pinMode(DTMF_D0, INPUT);
  pinMode(DTMF_D1, INPUT);
  pinMode(DTMF_D2, INPUT);
  pinMode(DTMF_D3, INPUT);
  pinMode(DTMF_DV, INPUT);

  attachInterrupt(digitalPinToInterrupt(DTMF_DV), [](){
    uint8_t d0 = digitalRead(DTMF_D0);
    uint8_t d1 = digitalRead(DTMF_D1);
    uint8_t d2 = digitalRead(DTMF_D2);
    uint8_t d3 = digitalRead(DTMF_D3);
    lastNibble = (d3<<3)|(d2<<2)|(d1<<1)|(d0<<0);
    dvRiseFlag = true;
  }, RISING);

  lcd.begin(16, 2);
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0); lcd.print("Senha: 1-2-3");
  lcd.setCursor(0,1); lcd.print("Aguardando...");

  dfSerial.begin(9600);
  initDF();      // inicializa DFPlayer (com retry + modo cego)
```

```
  recalibrar(false);
  lcdShowLocked();
  lastActivityAt = millis();

  // Áudio de sistema ativo (se quiser manter)
  fala(TRK_SYSTEM_ACTIVE);
}

/*======================== LOOP ======================*/
void loop() {
  // Heartbeat LED
  static unsigned long tHB = 0;
  if (millis() - tHB >= 300) {
    tHB = millis();
    digitalWrite(LED_HB, !digitalRead(LED_HB));
  }

  /* ---------- Botão de recalibração (AJUSTADO) ---------- */
  bool btn = digitalRead(PINO_BOTAO);

  // Borda de descida: botão pressionado
  if (btn == LOW && lastBtn == HIGH) {
    btnDownAt = millis();
    lastActivityAt = millis();
  }

  // Borda de subida: botão solto → decide curto x longo
  if (btn == HIGH && lastBtn == LOW) {
    unsigned long t = millis() - btnDownAt;

    if (t < 2000) {
      // Recalibração normal
      recalibrar(true);
      if (mode == LOCKED) lcdShowLocked();
    } else {
      // Hard Recalib (mesma função, só muda texto)
      lcdMsg("Hard Recalib...", 600);
      recalibrar(true);
      if (mode == LOCKED) lcdShowLocked();
    }
    lastActivityAt = millis();
  }

  lastBtn = btn;

  /* ---------- Leitura corrente ACS ---------- */
  float Iraw1 = medirIrms_A(PINO_ACS1, sensib1, N_RMS, DELAY_MS, offset1);
  float Iraw2 = medirIrms_A(PINO_ACS2, sensib2, N_RMS, DELAY_MS, offset2);

  float Icor1 = Iraw1 - calibOff1_A;
  float Icor2 = Iraw2 - calibOff2_A;
  if (Icor1 < 0.0f) Icor1 = 0.0f;
  if (Icor2 < 0.0f) Icor2 = 0.0f;

  if (Icor1 < I_MIN_DISPLAY) Icor1 = 0.0f;
  if (Icor2 < I_MIN_DISPLAY) Icor2 = 0.0f;
```

```
  float I1 = aplicaEMA(Icor1, ema1, emaInit1);
  float I2 = aplicaEMA(Icor2, ema2, emaInit2);

  // Histerese de ligado/desligado:
  // - Sempre mostramos a leitura REAL do ACS (I1/I2).
  // - MAS só consideramos a carga "LIGADA" (onX = true) se o relé daquela carga estiver ON.
  if (relay1On) {
    if (!on1 && I1 >= TH_ON_A)  on1 = true;
    if ( on1 && I1 <= TH_OFF_A) on1 = false;
  } else {
    on1 = false;     // relé 1 desligado → estado lógico OFF, mesmo que ACS veja ruído
  }

  if (relay2On) {
    if (!on2 && I2 >= TH_ON_A)  on2 = true;
    if ( on2 && I2 <= TH_OFF_A) on2 = false;
  } else {
    on2 = false;     // relé 2 desligado → estado lógico OFF
  }

  if (mode == UNLOCKED) {
    printLinha2Canais(I1, on1, I2, on2);
  }

  /* ---------- Tecla DTMF ---------- */
  int key = dtmfReadKey();
  if (key != -1) {
    lastActivityAt = millis();
    if (mode == LOCKED) handleLockedKey(key);
    else                handleUnlockedKey(key);
  }

  /* ---------- Auto-lock ---------- */
  if (mode == UNLOCKED && (millis() - lastActivityAt >= INACTIVITY_MS)) {
    mode = LOCKED;
    pass_pos = 0;
    lcdShowLocked();
    // feedback de sistema inativo pelo DFPlayer
    fala(TRK_SYSTEM_INACTIVE);
    resetPending = false; // garante que não fique pendente após timeout
  }

  delay(40);
}

/*================= DFPLAYER ================*/
// inicialização com retry + "modo cego"
void initDF() {
  df_ok = false;

  // dá tempo para o DFPlayer montar SD
  delay(2000);

  for (uint8_t tent = 0; tent < 3 && !df_ok; tent++) {
    if (dfp.begin(dfSerial)) {
      df_ok = true;
      break;
```

```
    }
    delay(500);
  }

  if (!df_ok) {
    lcd.clear();
    lcd.setCursor(0,0); lcd.print("DF sem resp.");
    lcd.setCursor(0,1); lcd.print("Modo Cego ON");
    delay(1500);

    // modo cego: vamos tentar mandar comando mesmo assim
    df_ok = true;
  }

  dfp.outputDevice(DFPLAYER_DEVICE_SD);
  dfp.volume(30);          // volume alto pra teste
  dfp.EQ(DFPLAYER_EQ_NORMAL);
}

void fala(uint16_t trk){
  if (!df_ok || trk == 0) return; // ignora trilhas 0/inexistentes
  dfp.play(trk);
  delay(20);
}

void falaC1On()  { fala(TRK_ON[0]);  }
void falaC1Off() { fala(TRK_OFF[0]); }
void falaC2On()  { fala(TRK_ON[1]);  }
void falaC2Off() { fala(TRK_OFF[1]); }

/*================== MEDIÇÃO ==================*/
uint16_t medirOffset(uint8_t pino, uint16_t N, uint16_t atraso_ms){
  unsigned long soma = 0;
  for (uint16_t i=0; i<N; i++) {
    soma += analogRead(pino);
    delay(atraso_ms);
  }
  return (uint16_t)(soma / N);
}

float medirIrms_A(uint8_t pino, float sensib, uint16_t N, uint16_t atraso_ms, uint16_t zero){
  double soma2 = 0.0;
  for (uint16_t i=0; i<N; i++) {
    int adc   = analogRead(pino);
    int delta = adc - (int)zero;
    float v   = delta * (Vref_ADC / 1023.0f);
    soma2 += (double)v * (double)v;
    delay(atraso_ms);
  }
  float Vrms = sqrt(soma2 / (double)N);
  return Vrms / sensib;
}

float aplicaEMA(float x, float &y, bool &inited){
  if (!inited) {
    y = x; inited = true;
  } else {
```

```
    y += ALPHA_EMA * (x - y);
  }
  return y;
}

void recalibrar(bool showLCD){
  if (showLCD){
    lcd.clear();
    lcd.setCursor(0,0); lcd.print("Recalibrando...");
    lcd.setCursor(0,1); lcd.print("Aguarde");
  }

  offset1 = medirOffset(PINO_ACS1, 800, 1);
  offset2 = medirOffset(PINO_ACS2, 800, 1);

  float Irep1 = medirIrms_A(PINO_ACS1, sensib1, 1000, 1, offset1);
  float Irep2 = medirIrms_A(PINO_ACS2, sensib2, 1000, 1, offset2);

  calibOff1_A = Irep1;
  calibOff2_A = Irep2;

  emaInit1 = emaInit2 = false;
  on1 = on2 = false;

  if (showLCD){
    lcd.clear();
    lcd.setCursor(0,0); lcd.print("Calib. OK");
    lcd.setCursor(0,1);
    lcd.print(Irep1,3); lcd.print(" / "); lcd.print(Irep2,3);
    delay(1500);
  }
}

/*================= LCD ==================*/
void printLinha2Canais(float I1, bool s1, float I2, bool s2){
  char b1[17], b2[17], v1[8], v2[8];
  dtostrf(I1, 6, 3, v1);
  dtostrf(I2, 6, 3, v2);
  snprintf(b1,sizeof(b1),"C1 %sA %s",v1,s1?"ON ":"OFF");
  snprintf(b2,sizeof(b2),"C2 %sA %s",v2,s2?"ON ":"OFF");

  lcd.setCursor(0,0); lcd.print(b1);
  for (int i=strlen(b1); i<16; i++) lcd.print(' ');

  lcd.setCursor(0,1); lcd.print(b2);
  for (int i=strlen(b2); i<16; i++) lcd.print(' ');
}

void lcdMsg(const char* msg, uint16_t ms){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(msg);
  delay(ms);
}

void lcdShowLocked(){
  lcd.clear();
```

```
  lcd.setCursor(0,0); lcd.print("Senha (3 dig):");
  lcd.setCursor(0,1); lcd.print("_ _ _");
}

/*================= DTMF ==================*/
int mapNibbleToKey(uint8_t nib){
  if (nib >= 1 && nib <= 9) return nib;
  if (nib == 0x0A)          return 0;
  return -1;
}

int dtmfReadKey(){
  if (!dvRiseFlag) return -1;
  noInterrupts();
  uint8_t nib = lastNibble;
  dvRiseFlag = false;
  interrupts();

  int k = mapNibbleToKey(nib);
  if (k == -1) {
    uint8_t d0 = (nib & 0x01) ? 1 : 0;
    uint8_t d1 = (nib & 0x02) ? 1 : 0;
    uint8_t d2 = (nib & 0x04) ? 1 : 0;
    uint8_t d3 = (nib & 0x08) ? 1 : 0;
    uint8_t nib_inv = (d0<<3)|(d1<<2)|(d2<<1)|(d3<<0);
    k = mapNibbleToKey(nib_inv);
  }
  return k;
}

/*============== Senha e teclas ==============*/
void handleLockedKey(int k){
  if (k < 0 || k > 9) return;

  if (millis() - lastKeyAt < KEY_COOLDOWN_MS) return;
  lastKeyAt = millis();

  lcd.setCursor(pass_pos*2,1);
  lcd.print((char)('0'+k));
  lcd.setCursor(pass_pos*2+1,1);
  lcd.print(' ');

  pass_buf[pass_pos++] = (uint8_t)k;

  if (pass_pos >= PASS_LEN){
    bool ok = true;
    for (uint8_t i=0; i<PASS_LEN; i++) {
      if (pass_buf[i] != PASS_SEQ[i]) { ok=false; break; }
    }
    if (ok){
      fala(TRK_AUTH_OK);             // "senha correta"
      lcdMsg("Senha correta",1500);
      mode = UNLOCKED;
      lcd.clear();
    } else {
      fala(TRK_AUTH_FAIL);           // "senha incorreta"
      lcdMsg("Senha incorreta",1200);
```

```
      pass_pos = 0;
      lcdShowLocked();
    }
  }
  lastActivityAt = millis();
}

void handleUnlockedKey(int k){
  auto canDoKey = [&](){ return (millis() - lastKeyAt) >= KEY_COOLDOWN_MS; };

  // Se estamos aguardando confirmação de reset (9 foi pressionado antes)
  if (resetPending) {
    if (!canDoKey()) return;
    lastKeyAt = millis();

    if (k == 1) {
      // 9 + 1 => resetar sistema
      resetSistema();
      fala(TRK_RESET_DONE);      // "sistema resetado"
    } else if (k == 2) {
      // 9 + 2 => não autoriza reset
      lcdMsg("Reset cancelado", 800);
      fala(TRK_RESET_DENY);      // "reset nao autorizado"
    } else {
      // Qualquer outra tecla também cancela o reset por segurança
      lcdMsg("Reset cancelado", 800);
      fala(TRK_RESET_DENY);
    }

    resetPending = false;
    lastActivityAt = millis();
    return; // não processa mais nada desta tecla
  }

  // Se NÃO estamos em modo de confirmação de reset, segue fluxo normal
  if (k == 1){
    // Tecla 1: LIGA/DESLIGA CARGA 1 (C1)
    if (canDoKey()){
      bool wasOn = relay1On;
      toggleRelay1();  // muda o estado do relé C1

      if (!wasOn && relay1On){
        // Acabou de ligar C1 → verifica corrente
        lcdMsg("Verif. C1...", 300);
        delay(VERIFY_DELAY_MS);
        float Icheck = medirCorrenteCorrigida_C1();

        if (Icheck >= TH_AUDIO_ON_A){
          falaC1On();                  // "Carga 1 ligada"
          lcdMsg("C1 LIGADA (OK)", 700);
        } else {
          fala(TRK_NC[0]);             // "Carga 1 sem corrente"
          lcdMsg("C1 SEM CORRENTE", 900);
        }
      } else if (wasOn && !relay1On){
        // Acabou de desligar C1
        falaC1Off();                   // "Carga 1 desligada"
```

```
      lcdMsg("Relé C1: DESLIG", 700);
    }
    lastKeyAt = millis();
  }

} else if (k == 2){
  // Tecla 2: LIGA/DESLIGA CARGA 2 (C2)
  if (canDoKey()){
    bool wasOn = relay2On;
    toggleRelay2();  // muda o estado do relé C2

    if (!wasOn && relay2On){
      // Acabou de ligar C2 → verifica corrente
      lcdMsg("Verif. C2...", 300);
      delay(VERIFY_DELAY_MS);
      float Icheck = medirCorrenteCorrigida_C2();

      if (Icheck >= TH_AUDIO_ON_A){
        falaC2On();                  // "Carga 2 ligada"
        lcdMsg("C2 LIGADA (OK)", 700);
      } else {
        fala(TRK_NC[1]);             // "Carga 2 sem corrente"
        lcdMsg("C2 SEM CORRENTE", 900);
      }
    } else if (wasOn && !relay2On){
      // Acabou de desligar C2
      falaC2Off();                   // "Carga 2 desligada"
      lcdMsg("Relé C2: DESLIG", 700);
    }
    lastKeyAt = millis();
  }

} else if (k == 3){
  // Tecla 3: CONSULTA ESTADO LÓGICO DA CARGA 1 (C1) via on1
  if (on1) {
    falaC1On();              // "Carga 1 ligada"
    lcdMsg("C1: LIGADA", 700);
  } else {
    falaC1Off();             // "Carga 1 desligada"
    lcdMsg("C1: DESLIGADA", 700);
  }

} else if (k == 4){
  // Tecla 4: CONSULTA ESTADO LÓGICO DA CARGA 2 (C2) via on2
  if (on2) {
    falaC2On();              // "Carga 2 ligada"
    lcdMsg("C2: LIGADA", 700);
  } else {
    falaC2Off();             // "Carga 2 desligada"
    lcdMsg("C2: DESLIGADA", 700);
  }

} else if (k == 5){
  // Tecla 5: CONSULTA CORRENTE DA CARGA 1 (C1)
  float Icheck = medirCorrenteCorrigida_C1();

  lcd.clear();
```

```
    lcd.setCursor(0,0); lcd.print("C1 Corrente:");
    lcd.setCursor(0,1); lcd.print(Icheck,3); lcd.print(" A");

    if (Icheck >= TH_AUDIO_ON_A){
      falaC1On();                    // "Carga 1 ligada"
    } else {
      fala(TRK_NC[0]);               // "Carga 1 sem corrente"
    }
    delay(900);

  } else if (k == 6){
    // Tecla 6: CONSULTA CORRENTE DA CARGA 2 (C2)
    float Icheck = medirCorrenteCorrigida_C2();

    lcd.clear();
    lcd.setCursor(0,0); lcd.print("C2 Corrente:");
    lcd.setCursor(0,1); lcd.print(Icheck,3); lcd.print(" A");

    if (Icheck >= TH_AUDIO_ON_A){
      falaC2On();                    // "Carga 2 ligada"
    } else {
      fala(TRK_NC[1]);               // "Carga 2 sem corrente"
    }
    delay(900);

  } else if (k == 9){
    // Tecla 9: INICIA fluxo de reset (precisa confirmar)
    if (canDoKey()){
      resetPending = true;

      // Mensagem no LCD pedindo confirmação
      lcd.clear();
      lcd.setCursor(0,0); lcd.print("Reset sistema?");
      lcd.setCursor(0,1); lcd.print("1=SIM  2=NAO");

      // Aviso sonoro
      fala(TRK_RESET_WARN);   // "atencao, o sistema sera reiniciado"

      lastKeyAt = millis();
    }

  } else {
    // Outras teclas: só avisa que é inválida (sem áudio)
    lcdMsg("Tecla incorreta", 900);
  }

  lastActivityAt = millis();
}

/*================ Relés ================*/
void applyRelay(uint8_t pin, bool on){
  if (REL_ACTIVE_LOW) digitalWrite(pin, on ? LOW : HIGH);
  else                digitalWrite(pin, on ? HIGH : LOW);
}

void toggleRelay1(){
  relay1On = !relay1On;
```

```
  applyRelay(RELAY1_PIN, relay1On);
}

void toggleRelay2(){
  relay2On = !relay2On;
  applyRelay(RELAY2_PIN, relay2On);
}

/*======== Medição rápida corrigida =========*/
float medirCorrenteCorrigida_C1(){
  float Iraw = medirIrms_A(PINO_ACS1, sensib1, N_RMS, DELAY_MS, offset1);
  float Icor = Iraw - calibOff1_A;
  if (Icor < 0.0f) Icor = 0.0f;
  return Icor;
}

float medirCorrenteCorrigida_C2(){
  float Iraw = medirIrms_A(PINO_ACS2, sensib2, N_RMS, DELAY_MS, offset2);
  float Icor = Iraw - calibOff2_A;
  if (Icor < 0.0f) Icor = 0.0f;
  return Icor;
}

/*======== RESET SISTEMA (tecla 9 + 1) =========*/
void resetSistema(){
  // Desliga relés e flags
  relay1On = false;
  relay2On = false;
  applyRelay(RELAY1_PIN, false);
  applyRelay(RELAY2_PIN, false);

  // Zera estado lógico de corrente
  on1 = false;
  on2 = false;
  emaInit1 = false;
  emaInit2 = false;

  // Volta para modo bloqueado (senha)
  mode = LOCKED;
  pass_pos = 0;
  lcdShowLocked();

  // Áudio de "sistema resetado" é tocado fora (em handleUnlockedKey)
}
```

Apêndice D – Procedimento para criação e gravação dos arquivos de áudio no Dfplayer mini

Este apêndice tem como objetivo descrever, de forma simples e reprodutível, o procedimento completo para a geração, nomeação e preparo dos arquivos de áudio necessários para o correto funcionamento do sistema de *feedback* de voz do projeto.

1. Geração e Preparação das Mensagens de Voz

Para a geração das mensagens de voz, foi adotado o método de síntese de voz (*Text-to-Speech – TTS*), utilizando um serviço online (`https://luvvoice.com`). O procedimento envolve acessar a plataforma, digitar a frase de *feedback* desejada (ex: "Sistema ativo," "Carga um ligada," ou "Carga dois desligada"), selecionar o Idioma Português (Brasil) e manter a mesma voz padrão em todos os arquivos para garantir a padronização e humanização do *feedback*. Após a geração, o áudio é baixado no formato MP3 e salvo temporariamente no computador, repetindo-se o processo para todas as mensagens de voz requeridas pela lógica do sistema.

2. Nomeação e Estrutura de Arquivos

A correta nomeação dos arquivos é crucial, pois o módulo DFPlayer Mini não identifica os áudios pelo nome textual, mas sim por uma numeração fixa que deve coincidir com o comando enviado pelo código do Arduino. Desta forma, o nome do arquivo deve, obrigatoriamente, conter quatro dígitos numéricos, seguidos da extensão `.mp3`. Não devem ser utilizados letras, espaços ou caracteres especiais no nome do arquivo.

Exemplos de Nomes Válidos: O áudio de "Sistema inativo" deve ser nomeado como `0001.mp3`, "Carga 1 ligada" como `0002.mp3`, e assim sucessivamente. Esta correspondência é vital: o comando `dfp.play(4);` no Arduino, por exemplo, fará o DFPlayer reproduzir o arquivo `0004.mp3`.

3. Preparação do Cartão microSD

O cartão microSD atua como a memória de armazenamento das mensagens de voz. Para garantir o funcionamento correto com o DFPlayer Mini, o cartão deve ter uma capacidade recomendada de até 32 GB e ser formatado com o Sistema de Arquivos FAT32. Após a formatação, todos os arquivos MP3 corretamente nomeados devem ser copiados diretamente para a raiz do cartão, sendo vedada a criação de subpastas. É fundamental que o cartão seja dedicado exclusivamente aos áudios do projeto. Após a verificação dos nomes e estrutura, o cartão deve ser ejetado em segurança e inserido no módulo DFPlayer Mini.

4. Boas Práticas Adotadas

Durante a implementação, foram adotadas boas práticas para otimizar o desempenho do *feedback* de voz. Todas as mensagens foram gravadas com volume semelhante, evitando

variações bruscas que pudessem prejudicar a clareza na linha telefônica. A padronização de voz e idioma foi mantida, e os áudios foram testados individualmente antes da integração ao sistema. O volume final do DFPlayer Mini foi ajustado via *software* no Arduino para evitar distorções na saída de áudio.