UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA
DE CONTROLE E AUTOMAÇÃO

MATHEUS ATANASIO DE MORAIS MELO

SISTEMA EMBARCADO E SUPERVISÓRIO PARA UMA APLICAÇÃO IOT

Recife
2023

MATHEUS ATANASIO DE MORAIS MELO

SISTEMA EMBARCADO E SUPERVISÓRIO PARA UMA APLICAÇÃO IOT

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador(a): Prof. Dr. Douglas Contente Pimentel Barbosa

Recife
2023

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Melo, Matheus Atanasio de Morais.
Sistema Embarcado e Supervisório para uma Aplicação IoT / Matheus Atanasio de Morais Melo. - Recife, 2023.
56 : il., tab.

Orientador(a): Douglas Contente Pimentel Barbosa
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia de Controle e Automação - Bacharelado, 2023.
Inclui referências, apêndices, anexos.

1. Automação. 2. ESP32. 3. Sistemas Embarcados. 4. Sistemas Supervisórios. I. Barbosa, Douglas Contente Pimentel. (Orientação). II. Título.

620 CDD (22.ed.)

MATHEUS ATANASIO DE MORAIS MELO

SISTEMA EMBARCADO E SUPERVISÓRIO PARA UMA APLICAÇÃO IOT

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Controle e Automação

Aprovado em: 26/09/2023

BANCA EXAMINADORA

_____
Prof. Dr. Douglas Contente Pimentel Barbosa
(Orientador)
Universidade Federal de Pernambuco

_____
Prof. Dr. Eduardo José Barbosa (Examinador Interno)
Universidade Federal de Pernambuco

_____
Prof. Msc. Eduardo Augusto Oliveira Barbosa
(Examinador Interno)
Universidade Federal de Pernambuco

RESUMO

A Internet das Coisas (IoT) é uma tendência tecnológica cujo objetivo é a integração de dispositivos à internet, permitindo troca de informações e automação de processos, compreendidos desde acionamentos manuais até rotinas personalizadas. Nesse contexto, este trabalho apresenta uma solução baseada na ESP32, um microcontrolador que possui características ideias para a implementação de projetos IoT. Assim, a aplicação desenvolvida combina o poder da ESP32 no controle e acionamento de dispositivos por meio de um sistema supervisório na web que fornece uma interface visual simples e intuitiva para o usuário, permitindo o controle e monitoramento desses dispositivos conectados à rede. O sistema criado para este trabalho controla aparelhos de ar-condicionado, televisão e iluminação, como LEDs e lâmpadas, e foi instalado, testado e validado em um estabelecimento comercial. Portanto, será demonstrado como o sistema funciona, as interfaces desenvolvidas e suas funcionalidades, além da construção do sistema embarcado para o controle e acionamento dos dispositivos então trabalhados.

Palavras-chave: Automação; ESP32; IoT; Sistemas Embarcados; Sistemas Supervisórios

ABSTRACT

The Internet of Things (IoT) is a technological tendency whose objective is the integration of everyday devices to the internet, allowing the exchange of information and the automation of these processes, from manual activation to personalized routines. In this context, this work presents a solution based on the ESP32, a microcontroller that has ideal features for implementing IoT projects. The developed application combines the power of the ESP32 for controlling and activating devices with a supervisory system on the web that provides a simple and intuitive visual interface for the user, allowing them to control and monitor the devices connected to the network. The system created for this work controls AC units, televisions, and lighting devices, such as LEDs and light bulbs, and was installed, evaluated, and validated in a commercial establishment. It will be demonstrated how the system works, the developed interfaces and their functionalities, in addition to the construction of the embedded system for controlling and activating the devices.

Keywords: Automation; Embedded Systems; ESP32; IoT; Supervisory Systems

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AC	*Alternating Current*
AP	*Access Point*
ADC	*Analog to Digital Converter*
ALU	*Arithmetic Logic Unit*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
CI	*Circuito Integrado*
DBMS	*Database Management System*
DC	*Direct Current*
GPIO	*General Purpose Input/Output*
HTTP	*Hypertext Transfer Protocol*
Hz	*Hertz*
I2C	*Inter-Integrated Circuit*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
IR	*Infrared*
JSON	*JavaScript Object Notation*
LED	*Light Emitting Diode*
MCU	*Microcontroller Unit*
MIME	*Multipurpose Internet Mail Extensions*
NC	*Normally Open*
NO	*Normally Closed*
PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*
RGB	*Red, Green Blue*
SDK	*Software Development Kit*
SPI	*Serial Peripheral Interface*
SQL	*Structured Query Language*
SSE	*Server-Sent Event*
SSID	*Service Set Identifier*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UART	*Universal Asynchronous Receiver/Transmiter*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
WiFi	*Wireless Fidelity*
XML	*Extensible Markup Language*

SUMÁRIO

1. INTRODUÇÃO

IoT, do inglês *Internet of Things* (internet das coisas), descreve a rede de objetos físicos incorporados a sensores, softwares e outras tecnologias com o objetivo de conectar e trocar dados com outros dispositivos e sistemas pela internet. Atualmente, esses dispositivos vão desde objetos mais comuns como lâmpadas que podem ser ligadas pelo celular até ferramentas industriais mais sofisticadas, e é uma tendência tecnológica cada vez mais presente no cotidiano.

A internet das coisas pode ser aplicada em soluções das mais diversas áreas, como automação residencial, setor hoteleiro, saúde, cidades inteligentes, entre outros.

1.1 Justificativa

O uso de automação tanto em residências quanto em ambientes empresariais aumentou expressivamente nos últimos anos e, em razão dessa popularização, tecnologias da área estão deixando de ser um item de luxo, ou seja, passando a abranger uma maior gama de usuários. Nesse contexto, destacam-se aplicações desenvolvidas com tecnologias de baixo custo, há diversos exemplos de produtos disponíveis no mercado que são comercializados por valores acessíveis.

Um sistema IoT deve ser preferencialmente simples e intuitivo para o usuário, de modo que ele possa ser utilizado por qualquer pessoa, de qualquer idade e dispensando qualquer conhecimento prévio. Essas características podem ser alcançadas por meio de interface simples e de fácil entendimento projetadas em um sistema supervisório, compreendendo todas as funcionalidades de um dado sistema em um dispositivo ao alcance do usuário, seja um celular ou computador.

Portanto, diante da crescente adoção do IoT para soluções comerciais e das relevâncias de suas aplicações, este trabalho busca contribuir para o avanço dessa tecnologia, destacando o papel do ESP32 como uma solução acessível e eficaz para o desenvolvimento de sistemas inteligentes e conectados. Ademais, também é mostrado a relevância da boa construção de sistemas supervisórios para esse fim.

1.2 Objetivo

O objetivo deste trabalho é propor uma solução de automação, inserida no contexto IoT, baseada em ESP32 que se comunica com banco de dados na nuvem para controle de dispositivos em quartos de hotel, inclusive oferecendo interface web

para usuários finais, ou seja, cada um dos hospedes individualmente.

O sistema visa proporcionar aos usuários maior comodidade e controle sobre o ambiente, permitindo que eles possam facilmente ajustar iluminação, ar-condicionado e controlar a televisão de acordo com suas preferências.

Ademais, pretende-se demonstrar o potencial da tecnologia IoT aplicada ao setor hoteleiro, destacando os benefícios de automação e personalização oferecidas por essa solução.

Todas as finalidades mencionadas são alcançadas com o desenvolvimento de um sistema embarcado e supervisório, ambos se comunicando com banco de dados *Firebase* e trocando informações entre si, permitindo total controle e monitoramento daquele ambiente onde o sistema se encontra instalado.

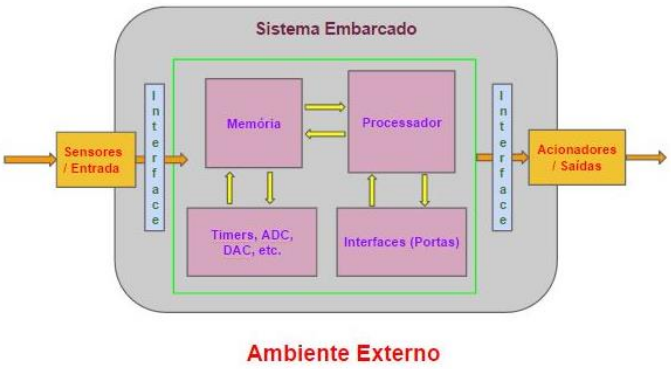
2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os referenciais teóricos que são base para a realização deste trabalho. É apresentado o conceito de microcontroladores, o dispositivo escolhido para o desenvolvimento da solução (ESP32), bem como as demais tecnologias utilizadas, como banco de dados, comunicação na web e sistemas supervisórios.

2.1 Sistemas Embarcados

Um sistema embarcado é um conjunto de hardware e software, projetado para executar uma tarefa específica em um sistema maior. Geralmente compostos por um limitado conjunto de componentes, como microcontroladores, sensores e atuadores. Esses sistemas são integrados em outras soluções ou equipamentos, visando controlar e/ou monitorar determinadas e variadas funções (REIS, 2015). A Figura 1 ilustra os principais elementos citados.

Figura 1 - Elementos de um sistema embarcado



Fonte: (REIS, 2015)

Os sistemas embarcados são geralmente projetados para serem simples, de baixo custo, compactos ou consumir o mínimo de energia possível. Outra característica é a limitação em termos de espaço e recursos, exemplo de memória e processamento reduzidos (REIS, 2015).Existem sistemas com memória de 30 kb e processadores com único núcleo de 8 bits.

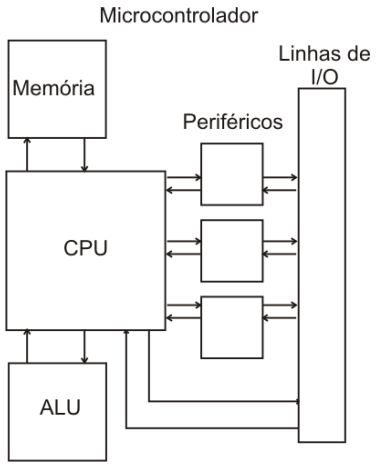
Podendo ser encontrados em uma variedade de aplicações, incluindo

automóveis, indústrias e aparelhos eletrônicos, os sistemas embarcados desempenham um papel crucial na vida cotidiana, estando presente em dispositivos de uso pessoal até sistemas de infraestrutura crítica (GARCIA, 2018).

2.1.1 Microcontroladores

Um microcontrolador, também chamado de MCU, é um componente eletrônico programável de pequeno porte e baixo custo, integrado em um único circuito, que contêm processador, unidade de lógica e aritmética (ALU), memória e periféricos, como GPIOs (entradas e saídas), módulos de comunicação serial e conversores ADC (SOARES, 2002). Com o auxílio de todos esses componentes e do programa gravado em sua memória, ele é projetado para executar tarefas específicas em sistemas eletrônicos, como operar atuadores, receber e processar dados de sensores, entre outras funções. A Figura 2 mostra de maneira simplificada as partes integrantes de um microcontrolador genérico.

Figura 2 - Blocos básicos de um MCU



Fonte: (SOARES, 2002)

Para além de sua capacidade de integração com sensores e atuadores, destaca-se também a possibilidade de comunicação desses dispositivos (SOARES, 2002). É possível trocar informações entre dois microcontroladores ou entre um microcontrolador e um página na web, por exemplo. Isso pode ser feito de diversas maneiras e com diferentes protocolos de comunicação. O que vem se destacando no transcorrer dos últimos anos é realizar esse tipo de comunicação via WiFi, ou seja,

para aplicações sem fio.

É justamente com essa comunicação via WiFi que a IoT vem se popularizando. Torna-se mais simples para o usuário comandar e monitorar suas tarefas quando se tem um sistema capaz de transmitir informações e receber comandos de uma interface na web ou aplicativo no celular, por exemplo.

Existem no mercado diversos fabricantes e diferentes modelos de microcontroladores, cada um com sua especificação própria, como tensão de alimentação e de saída, memória RAM, número de GPIOs, dentre outros. Cabe ao projetista escolher o modelo que melhor se encaixa para sua aplicação, levando em consideração todas as características que cada MCU oferece, isto em termos de necessidades especificas, relação custo-benefício, entre outras que venham a ser pertinentes para a especificidade de cada projeto.

2.1.1.1 ESP32

Na Figura 3 está ilustrado o *chip* de um dos microcontroladores mais populares da atualidade, a ESP32. Ela é um poderoso MCU desenvolvido pela *Espressif Systems.* É uma plataforma integrada que combina um processador de alto desempenho e conectividade WiFi e Bluetooth, tornando-o uma solução ideal para projetos IoT.

Figura 3 - ESP32

Fonte: (DIGIKEY, 2023)

Existem diferentes modelos de ESP32, cada um com sua própria especificação. Eles variam desde modelos *single-core* a *dual-core*, com diferentes capacidades de memória RAM, armazenamento, além de outras características que mudam de modelo para modelo (KOYANAGI, 2017).

Existem em torno de 50 modelos diferentes de ESP32, divididos em 7

categorias distintas. Cada uma tem sua particularidade em termos de periféricos, número de entradas e saídas, capacidade de processamento, dentre outras especificações técnicas. Esse é um ponto interessante, pois dá ao projetista a opção de escolher um modelo que melhor se adeque ao sistema que ele irá construir.

A ESP32 também conta com uma ampla gama de periféricos integrados, como UART (*Universal Asynchronous receiver/transmitter)*, SPI (*Serial Peripheral Interface)* e I2C (*Inter-Integrated Circuit)* para comunicação serial. Além disso, possui diversas entradas com capacidade PWM e *touch.*

A programação da ESP32 pode ser realizada em diferentes linguagens, como C, C++ ou python, por meio de diferentes IDEs, como Arduino IDE, VsCode e Espressif SDK (BATISTA DA CRUZ; DE CARVALHO RUTZ DA SILVA; LUIZ RUTZ DA SILVA, 2022). Para auxiliar o desenvolvimento de projetos, existem diversas bibliotecas e *frameworks* disponíveis que facilitam a sua execução.

Em resumo, a ESP32 é um microcontrolador versátil e poderoso, com recursos de conectividade sem fio, periféricos integrados e baixo e consumo de energia. Essa combinação a torna uma popular escolha para projetos de IoT, permitindo a criação de dispositivos inteligentes e conectados de forma eficiente.

2.2 Comunicação Infravermelha

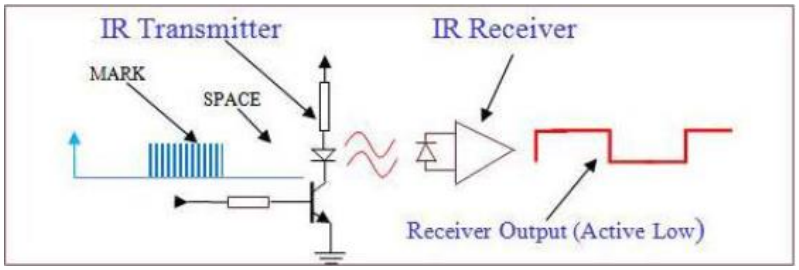
A comunicação infravermelha está entre os métodos de comunicação sem fio mais simples, e serve como maneira econômica de transmitir alguns bits de dados dessa forma. Muitos equipamentos modernos, como aparelhos de ar-condicionado e televisores, utilizam essa tecnologia e nesses dois casos, tem-se um controle remoto com um LED infravermelho que funciona como emissor e, no equipamento que irá receber o sinal, um fototransistor que funciona como receptor.

Tipicamente, para a transmissão de dados infravermelho não há métodos de autenticação, autorização ou *handshake* entre o transmissor e o receptor. Por exemplo, uma televisão pode ser controlada por qualquer controle que “fale” a sua mesma “linguagem”. Em comunicações, linguagem está se referindo a um protocolo de comunicação, que nada mais é que uma sequência de regras entre o transmissor e o receptor para que alguma mensagem seja passada entre um e outro.

Um método simples de transmitir valores binários com um LED infravermelho seria ligando-o e desligando-o por um certo período. O transmissor poderia fazer isso

até que todos os bits tenham sido transmitidos. Na realidade, isso não funcionaria porque pode haver muitas outras fontes de radiação infravermelha e o receptor não saberia distinguir os sinais desejados dos indesejados. Para superar esse obstáculo, o transmissor precisa ligar e desligar o LED muito rapidamente, modulando o sinal. Nessa técnica, é inserida a informação transmitida dentro de outra frequência, chamada de frequência portadora, que costuma ser de 38 kHz (AMEBAIOT, 2023). Na Figura 4 está ilustrado de forma mais clara como é o sinal modulado pelo transmissor e como ele é interpretado pelo receptor.

Figura 4 - Modulação de sinal IR



Fonte: (AMEBAIOT, 2023)

2.2.1 Protocolo NEC

A técnica de piscar o LED IR não é um protocolo de comunicação, mas sim uma modulação de sinal. O protocolo vai além disso, ele descreve como os dados devem ser disponibilizados para que os dois dispositivos se comunicando entendam a mensagem.

Um dos protocolos mais populares para transmissão de dados via sinais IR é o NEC (*Nippon Electric Company,* empresa que desenvolveu o protocolo). Por meio de uma técnica de modulação chamada *Pulse Distance Encoding,* o dispositivo receptor consegue distinguir entre os dois estados lógicos e saber qual radiação infravermelha é o sinal que ele procura (AMEBAIOT, 2023).

Nesse esquema de codificação, as durações de espaços (digital 0) e marcas (digital 1) codificam os valores lógicos. O receptor reconhece os sinais com base em suas durações específicas e reconstrói o comando original para executar a função desejada no dispositivo de destino (AMEBAIOT, 2023).

O protocolo NEC tem três características principais:

- 8 bits de endereço e 8 bits de comando;

- Endereço e comando transmitidos duas vezes para fins de confiabilidade da transmissão;
- Frequência portadora de 38 kHz.

2.2.1.1 Funcionamento do Protocolo NEC

O funcionamento do protocolo NEC pode ser dividido em etapas bem definidas, cada uma com sua respectiva função. Abaixo, está descrito a função de cada etapa (ALTIUM, 2017), além de uma amostra de um comando NEC na Figura 5.

- *Burst Pair*: O protocolo NEC inicia com um par de “rajadas”, que consiste em um pulso positivo e negativo da frequência portadora. Este par de rajadas está presente no início de cada transmissão e atua como um sinal de sincronização;
- Bit de partida: Após a rajada inicial, um bit de partida é transmitido. Esse bit consiste em um espaço de duração de 9 ms;
- Bits de endereço: O bit de endereço é utilizado para especificar o dispositivo de destino ou a função a ser controlada. O protocolo NEC utiliza 8 bits de endereço, cada um transmitido por um espaço ou marca por uma duração de 560 µs;
- Bits de endereço barrado: Para prover detecção de erro, o protocolo NEC também envia o valor invertido dos 8 bits de endereço. Essa redundância permite que o receptor verifique a precisão dos dados recebidos;
- Bits de comando: Uma vez transmitidos os bits de endereço e seus respectivos valores invertidos, seguem-se os bits de comando. Os bits de comando também têm um tamanho de 8 bits e representam a ação ou função específica a ser executada. Tal qual os bits de endereço, os bits de comando são transmitidos usando espaços e marcas. Analogamente, seu inverso também é transmitido com a finalidade de detecção de erros;
- Bit de parada: para marcar o fim do sinal, um bit de parada é transmitido. Esse bit é um espaço de duração de 560 µs.

Figura 5 - Protocolo NEC

Fonte: (AMEBAIOT, 2023)

2.2.1.2 Protocolo NEC Raw

Uma forma de visualizar o protocolo NEC é convertendo seu número de bits e seus tempos de espaços e marcas no formato de uma lista de números. Levando em consideração um sinal de 32 bits junto com os bits iniciais e finais, é possível separar os tempos, em microssegundos, e dispô-los em formato de lista como da Figura 6. Cada número representa tempos em microssegundos de espaços e marcas. É importante observar que os números são dispostos em pares de espaços e marcas. Esse é o chamado *raw NEC data* (AMEBAIOT, 2023). Esse formato é muito utilizado para gravar programas em microcontroladores que irão controlar dispositivos infravermelho.

Figura 6 - Amostra de dados Raw NEC

```
9000, 4500,                                                                   // starting bit
560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560, 560, 560,       // address
560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, 560, 1690, // ~address
560, 560, 560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560,       // data
560, 1690, 560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, //~data
560                                                                           // stoping bit
```

Fonte: (AMEBAIOT, 2023)

2.3 Transferência de Dados e Comunicação na Web

A transferência de dados via internet é o processo de enviar e receber informações entre computadores ou dispositivos conectados à internet. Isso pode ser feito através de diferentes protocolos de comunicação, como TCP/IP, UDP, HTTP, entre outros.

A escolha do protocolo é determinada pela finalidade da aplicação. Por exemplo, pode ser que seja necessário a verificação se todos os dados estão sendo enviados, comprometendo a velocidade de transmissão, ou então que a velocidade seja o principal em detrimento da perda de bits durante a comunicação.

2.3.1 HTTP

O HTTP é um protocolo de comunicação utilizado para transferir dados pela internet. Ele define a forma como os dados devem ser solicitados e transmitidos entre um cliente e servidor.

Ele opera no modelo cliente-servidor, onde o cliente envia uma solicitação para o servidor e aguarda uma resposta. A solicitação é composta por um método, um caminho de recurso (URL) e parâmetros opcionais. Após essa solicitação, a informação é processada pelo servidor, que retorna uma resposta, que geralmente contém um código de status e os dados solicitados (MOZILLA, 2023).

Uma vantagem do HTTP é, por meio do uso de diferentes tipos de mídia, permitir a transferência de diversos conteúdos, como *strings*, inteiros, booleanos, imagens ou vídeos. Também há disponibilidade de outros recursos como autenticação, *cache* de conteúdo e segurança, por exemplo.

2.3.1.1 Media-Types/Content Type

Os recursos pode ser um de diversos elementos encontrados em servidores web, como imagens, arquivos HTML, XML, JSON, dentre outros. Para que um servidor possa servir um recurso e para que o cliente possa recebê-lo adequadamente, ambos têm de ser específicos e precisos quanto ao tipo. Por exemplo, não faria sentido um servidor processar inteiros como *strings* (FERREIRA, 2022).

Quando um servidor responde uma requisição HTTP, ele devolve o recurso e o seu tipo, chamado *Media Type* ou *Content Type.* Para especificar tipos de recurso, o HTTP usa outro protocolo, que originalmente foi desenvolvido para comunicação por meio de e-mail, chamado MIME (FERREIRA, 2022).

O *Media-Type* tem duas partes: tipo e subtipo. Por exemplo, um servidor pode devolver uma imagem no formato png. O *Media-Type* da resposta viria no formato tipo/subtipo, nesse caso, como *image/png* (FERREIRA, 2022).

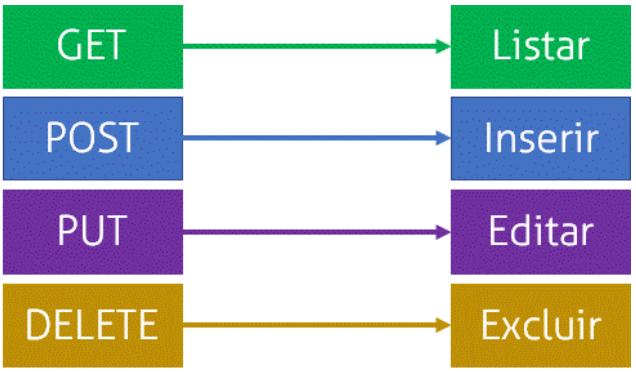
2.3.1.2 Verbos

O protocolo HTTP possui alguns métodos, os chamados verbos. Os verbos HTTP definem qual ação deve ser realizada e, a depender do verbo, o servidor pode dar uma resposta diferente (DEVMEDIA, 2023).

A utilização dos verbos é mais uma forma de padronizar a comunicação entre

um cliente e um servidor dentro do protocolo, permitindo que diferentes operações sejam realizadas em recursos da web. Existem diversos verbos, sendo os principais destacados na Figura 7.

Figura 7 - Principais Verbos HTTP



Fonte: (DEVMEDIA, 2023)

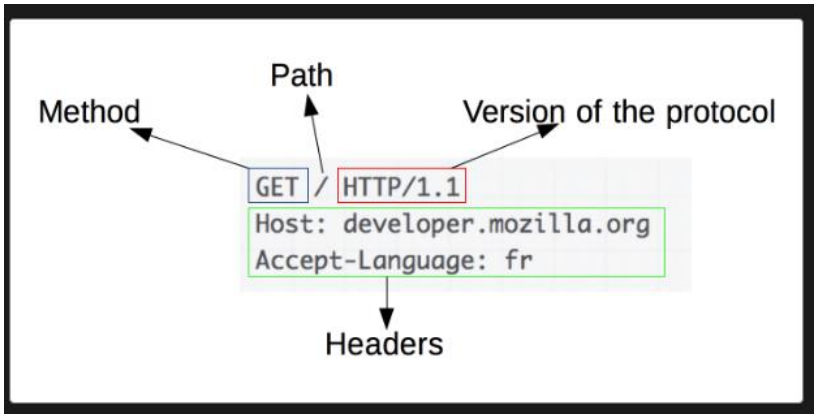
2.3.1.3 Requisição e Resposta

Em suma, o HTTP funciona da seguinte forma: quando um cliente precisa de um recurso que está em outra máquina, ele faz uma requisição para um servidor utilizando uma linguagem e vocabulários específicos que o servidor possa entender. Uma vez que o servidor compreenda a requisição e tenha o recurso disponível, ele irá enviar uma resposta. Por outro lado, se a requisição for entendida e o recurso não estiver disponível, o servidor pode informar que não possui o item solicitado. Ainda, se a requisição não for compreendida, é possível que não haja resposta do servidor (MOZILLA, 2023).

Dentro da requisição e resposta, existem dois elementos principais: o cabeçalho e o corpo. O cabeçalho permite que o cliente e o servidor passem informações adicionais, como autenticação e data, por exemplo (MOZILLA, 2023).

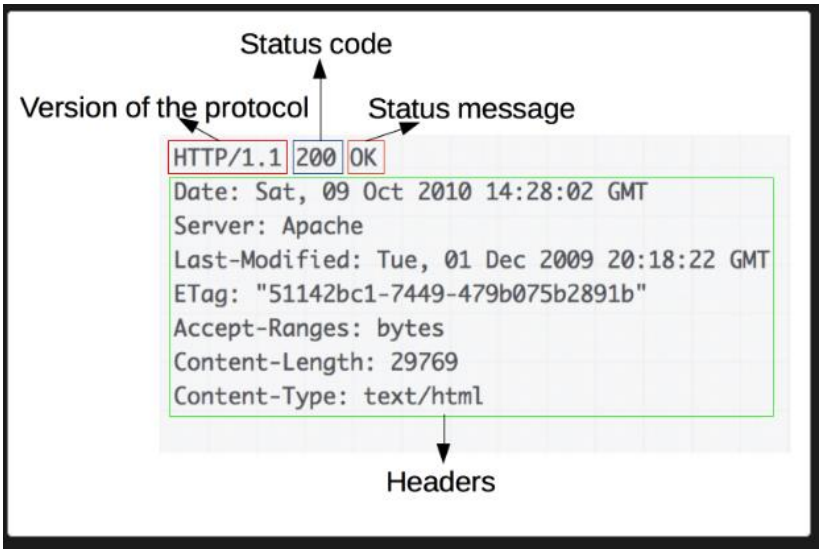
No contexto do HTTP, requisição e resposta são dois tipos de mensagens diferentes. A especificação HTTP define claramente o que pode ser incluído em cada tipo de mensagem, garantindo assim que todas as partes envolvidas na comunicação consigam trocar informações de maneira correta. Na Figura 8**Erro! Fonte de referência não encontrada.** é mostrado um formato típico de uma mensagem de requisição HTTP, enquanto a Figura 9 ilustra um exemplo de resposta HTTP.

Figura 8 - Exemplo de requisição HTTP



Fonte: (MOZILLA, 2023)

Figura 9 - Exemplo de Resposta HTTP



Fonte: (MOZILLA, 2023)

2.3.2 Tecnologia Push

Tecnologia *Push*, também conhecida como *server push*, refere-se a uma técnica de comunicação que permite que um servidor envie dados ou atualizações para um cliente automaticamente sem que ele solicite ou pesquise explicitamente (operação de *polling*). Nesse cenário, o servidor toma a iniciativa de enviar os dados ao cliente, “empurrando” as informações (ROUSE, 2012).

Tradicionalmente, as tecnologias web adotam um modelo *pull*, onde o cliente envia uma solicitação ao servidor para recuperar dados ou atualizações. Com a tecnologia *push*, o servidor pode enviar atualizações proativamente ao cliente,

reduzindo a necessidade de *polling* contínuo e permitindo a entrega de dados em tempo real ou muito próximo disso.

2.3.3 Eventos Server-Sent

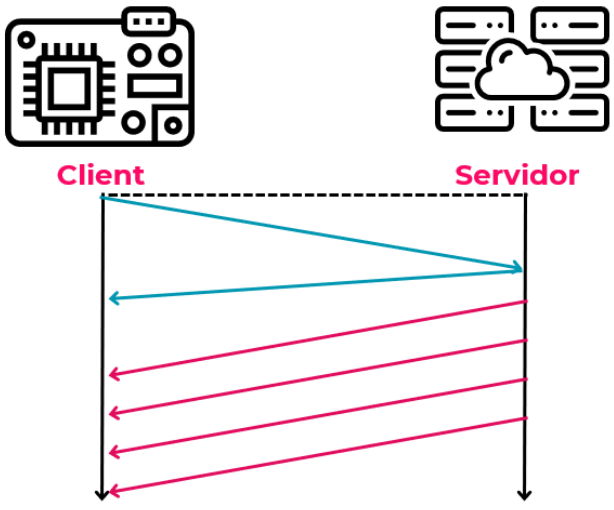
Um evento *Server-Sent* (SSE) é uma tecnologia *push* que permite que um servidor envie atualizações para um cliente em tempo real por meio de uma única conexão HTTP de longa duração. Ele permite que uma página web assine um fluxo de atualizações ou eventos de um servidor, e este pode enviar dados ao cliente sempre que novas informações estiverem disponíveis (DIAS, 2023).

O SSE costuma ser aplicado em cenários em que atualizações em tempo real precisam ser exibidas ou manipuladas. É uma alternativa mais simples a tecnologias que utilizam comunicação bidirecional entre o servidor e o cliente.

No geral, o SSE oferece uma maneira leve e eficiente de implementar atualizações em tempo real em aplicativos da web sem a necessidade de protocolos de comunicação bidirecionais mais complexos.

A Figura 10 ilustra o funcionamento do SSE em comparação ao HTTP padrão. As setas azuis destacam como é necessário haver uma requisição para então receber uma resposta. Em contrapartida, não é necessário que haja requisições contínuas para respostas no modelo SSE.

Figura 10 - Comparação entre HTTP padrão e SSE



Fonte: Autor

2.3.3.1 HTTP Streaming

HTTP *Streaming* é uma técnica de transferência de dados de estilo *push* que permite que um servidor da web envie dados continuamente a um cliente por meio de uma única conexão HTTP que permanece aberta indefinidamente.

Com o HTTP *Streaming*, o servidor é configurado para manter uma solicitação específica de um cliente e manter a resposta aberta para que possa enviar dados por meio dela. Quando atualizações pertencentes à solicitação estão disponíveis no lado do servidor, ele envia uma resposta por meio do canal de solicitação-resposta e só fecha a conexão quando é explicitamente instruído a fazê-lo. Dessa maneira, um cliente pode “ouvir” atualizações do servidor e recebê-las instantaneamente sem nenhuma sobrecarga associada a cabeçalhos HTTP e abertura/fechamento de conexões (PUBNUB, 2023).

Os dados podem ser transmitidos por meio do método SSE, para o qual existe uma API HTML5 padronizada chamada *EventSource*. Com o SSE, os dados são codificados como *text/event-stream* no cabeçalho (W3SCHOOLS, 2023).

2.4 JSON

O JSON é um formato de dados leve e de fácil leitura utilizado para troca de informações entre sistemas computacionais. Ele é frequentemente usado para transmitir dados entre um servidor e um cliente em aplicações web e móveis, embora também seja utilizado em diversos outros contextos.

Ele é amplamente utilizado na web para representar dados estruturados de forma legível tanto para humanos quanto para máquinas. O JSON é em formato texto, conforme ilustrado na Figura 11, e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C, C++, Java, JavaScrpit, python e muitas outras. Essas propriedades fazem com que JSON seja um formato ideal de troca de dados (JSON, 2023).

JSON está construído em duas estruturas: uma coleção de pares nome/valor e uma lista ordenada de valores. Estas são estruturas de dados universais. Virtualmente todas as linguagens de programação modernas as suportam de uma forma ou de outra. É aceitável que um formato de troca de dados que seja independente de linguagem de programação se baseie nestas estruturas (JSON, 2023).

Em resumo, o JSON é uma forma popular de representar dados estruturados e

transferi-los entre diferentes sistemas.

Figura 11 - Estrutura do JSON

```
{
   "cliente": {
       "id": 2020,
       "nome": "Maria Aparecida"
   },
   "pagamentos": [
       {
           "id": 123,
           "descricacao": "Compra do livro Cangaceiro JavaScript",
           "valor": 50.5
       },
       {
           "id": 124,
           "descricacao": "Mensalidade escolar",
           "valor": 1500
       }
   ]
}
```

Fonte: (LEITE, 2023)

2.5 Computação em Nuvem

A computação em nuvem é a entrega de recursos de TI, como servidores, armazenamento e banco de dados, sob demanda por meio da internet. Normalmente, o usuário que deseja utilizar serviços de computação em nuvem só paga por aqueles que efetivamente irá utilizar, ajudando a reduzir os custos operacionais, a executar sua infraestrutura com mais eficiência e a escalonar conforme as suas necessidades (AMAZON).

2.5.1 AWS

A AWS é a plataforma de computação em nuvem da Amazon. Ela entrega um conjunto de soluções para armazenamento de dados, infraestrutura de redes e servidores, virtualização, computação e hospedagem de aplicações, além de diversas outras ferramentas.

2.5.1.1 Hospedagem na Web

Um dos serviços da AWS é o de hospedagem. A hospedagem na web nada mais é que um serviço que armazena sites ou aplicações na web para torná-los

facilmente acessíveis em diferentes dispositivos como desktops, celulares e tablets.

As aplicações ou sites da web normalmente contém muitos arquivos, como imagens, vídeos, textos e códigos, que precisam ser armazenados em servidores. O provedor de serviços de hospedagem na web mantém, configura e executa servidores físicos que podem ser alugados para as finalidades de cada usuário (AMAZON).

2.5.2 Armazenamento na Nuvem

O armazenamento em nuvem é um modelo de computação em nuvem que permite armazenar dados e arquivos na internet por meio de um provedor de computação em nuvem que é acessado utilizando a internet pública ou uma conexão privada dedicada (AMAZON).

O provedor armazena, gerencia e mantém com segurança os servidores de armazenamento, a infraestrutura e a rede para garantir que o usuário tenha acesso aos dados quando precisar, em escala praticamente ilimitada e com capacidade elástica.

O armazenamento em nuvem elimina a necessidade de comprar equipamentos físicos e gerencia sua própria infraestrutura de armazenamento de dados, oferecendo agilidade, escalabilidade e durabilidade com acesso aos dados a qualquer hora e em qualquer lugar.

2.6 Banco de Dados

Um banco de dados é uma coleção organizada de informações – ou dados -, normalmente armazenadas eletronicamente em um sistema de computador.

Um banco de dados é geralmente controlado por um sistema de gerenciamento de banco de dados (DBMS). Os dados e o DBMS, juntamente com os aplicativos associados a eles, são chamados de sistema de banco de dados, geralmente abreviados para apenas banco de dados (ORACLE).

Os dados nos tipos mais comuns de banco de dados em operação atualmente são modelados em linhas e colunas em uma série de tabelas para tornar o processamento e a consulta de dados eficientes. Os dados podem ser facilmente acessados, gerenciados, modificados, atualizados, controlados e organizados. A maioria dos bancos de dados usam uma linguagem de consulta estruturada (SQL) para escrever e consultar dados (ORACLE).

Existem dois tipos principais de bancos de dados: bancos de dados relacionais e os não relacionais, também chamados de bancos de dados SQL e NoSQL, respectivamente.

2.6.1 Banco de Dados Relacionais

Os bancos de dados relacionais são um tipo de banco de dados que armazena e organiza pontos de dados com relacionamentos definidos para acesso rápido. Com um banco de dados relacional, os dados são organizados em tabelas, como mostrado na Figura 12, que são compostas por linhas e colunas (ORACLE).

Relacional significa indicar ou construir relação. No contexto de banco de dados, a forma como essa definição é feita se aplica principalmente aos próprios dados. Os conjuntos de dados que são relacionais tem relacionamentos predefinidos entre eles. Por exemplo, um banco de dados que inclui informações de clientes de uma empresa também pode incluir dados de transações individuais anexados a cada conta (ORACLE).

Figura 12 - Tabela de um bando de dados SQL

customerNumber	customerName	sales
146	Saveley & Henriot, Co.	130305.35
321	Corporate Gift Ideas Co.	132340.78
276	Anna's Decorations, Ltd	137034.22
187	AV Stores, Co.	148410.09
323	Down Under Souveniers, Inc	154622.08
148	Dragon Souveniers, Ltd.	156251.03
151	Muscle Machine Inc	177913.95
114	Australian Collectors, Co.	180585.07
124	Mini Gifts Distributors Ltd.	584188.24
141	Euro+ Shopping Channel	715738.98

Fonte: (MYSQLTUTORIAL)

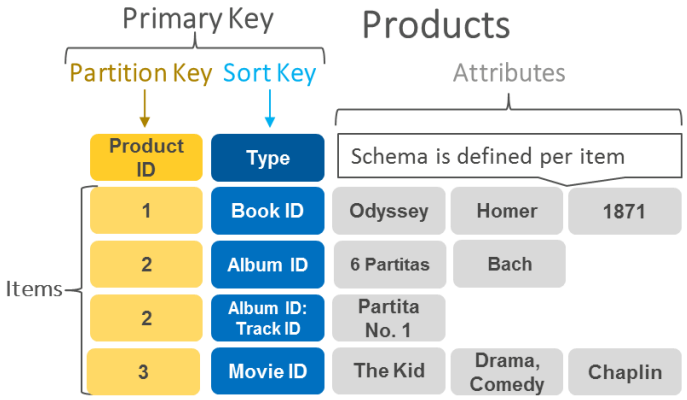
2.6.2 Banco de Dados NoSQL

Os bancos de dados NoSQL são um tipo de banco de dados que não seguem o modelo relacional mais tradicional. Eles foram desenvolvidos para lidar com problemas de escalabilidade, flexibilidade e velocidade de acesso aos dados que são desafiadores para os bancos de dados relacionais (ORACLE).

Um exemplo de banco de dados NoSQL bastante comum é o de chave-valor,

visto na Figura 13. Um banco de dados de chave-valor armazena os dados como um conjunto de pares chave-valor em que uma chave funciona como um identificador exclusivo. A chave e os valores podem ser de praticamente qualquer formato, desde objetos simples até objetos compostos complexos. Eles são altamente particionáveis e permitem escalabilidade horizontal que outros tipos de bancos de dados não conseguem alcançar (AMAZON).

Figura 13 - Estrutura de um banco de dados do tipo chave-valor



Fonte: (AMAZON)

2.6.2.1 Google Firebase

O *Google Firebase* é uma plataforma de desenvolvimento de aplicativos móveis e web, que fornece uma variedade de serviços e ferramentas para ajudar os desenvolvedores a criar, melhorar e dimensionar aplicativos. Ele oferece recursos como autenticação de usuário, armazenamento em nuvem, mensagem em tempo real, entre outros.

Ele possui um banco de dados em tempo real chamado *Realtime Database*, que funciona como um banco de dados NoSQL. Nesse caso, os dados são estruturas numa árvore JSON (GOOGLE). Essa estruturação em formato JSON é interna do *Realtime Database,* porém na interface para o usuário, a manipulação é feita de forma mais simples, como visto na Figura 14.

Todos os dados do *Firebase Realtime Database* são armazenados com objetos JSON. Ao contrário de um banco de dados SQL, não há tabelas nem registros. Quando dados são adicionados à árvore JSON, eles se tornam um nó na estrutura JSON com uma chave associada (GOOGLE).

O banco de dados usa uma árvore JSON, mas os dados armazenados nele

podem ser representados com tipos nativos que correspondem a tipos JSON disponíveis, como texto, inteiro e booleano, por exemplo.

Figura 14 - Estrutura criada no Firebase Realtime Database

Fonte: Autor

2.7 Sistemas Supervisórios

Os sistemas supervisórios desempenham um importante papel no contexto da Internet das Coisas, pois são responsáveis por monitorar e controlar os dispositivos conectados à rede. Eles coletam todos os dados relevantes e fornecem essa informação em tempo real. A Figura 15 mostra um simples sistema supervisório voltado para uma aplicação residencial.

Figura 15 - Exemplo de um simples sistema supervisório de automação residencial

Fonte: Autor

2.7.1 Características de um Sistema Supervisório

Um sistema supervisório precisa retirar as informações que ele disponibiliza de algum dispositivo. Para tal, ele tem a capacidade de se conectar com uma ampla gama de dispositivos, independente do protocolo de comunicação utilizado. Essa flexibilidade também permite que ele seja facilmente integrado a outros sistemas existentes.

Um ponto importante é que ele tenha uma interface amigável e intuitiva, permitindo ao usuário monitorar e controlar os dispositivos com facilidade. Um sistema supervisório pode ser aplicado no contexto de automação residencial, por exemplo. Nesse cenário, os usuários podem ser desde uma criança até uma pessoa idosa, e de qualquer formação. Logo, é crucial que a interface seja a mais simples possível.

Por fim, outra característica importante é a sua escalabilidade. Um sistema supervisório pode ser facilmente escalável, podendo lidar com diversos dispositivos conectados e adaptando-se às necessidades dos usuários.

2.7.2 Vantagens de um Sistema Supervisório

Uma das principais vantagens dos sistemas supervisórios é a capacidade de monitorar e controlar remotamente uma ampla variedade de dispositivos independente de sua localização geográfica. Isso proporciona maior flexibilidade e conveniência aos usuários, permitindo acesso e interação com os dispositivos conectados a qualquer momento e em qualquer lugar.

Também é interessante destacar o monitoramento de dados em tempo real, permitindo que os usuários acompanhem de maneira eficiente o estado de cada dispositivo.

3. DESENVOLVIMENTO DO TRABALHO

Neste capítulo é apresentando o projeto desenvolvido no trabalho. Primeiramente, é comentado sobre a planta onde ele foi instalado e as premissas das suas funcionalidades. Posteriormente, é analisado no detalhe como se dá esse funcionamento e os resultados então observados são discutidos.

Neste trabalho, o sistema desenvolvido teve como ponto de partida uma demanda real do mercado. Uma exigência do proprietário do local é que a solução não deveria ser composta por nenhuma das soluções mostrados na Figura 16 e que os clientes de alguma forma deveriam acessar um controle remoto virtual através da internet.

Portanto, a ESP32 foi escolhida como o microcontrolador de base para o projeto por ter especificações adequadas de memória, processamento e conectividade, e o sistema supervisório como interface de controle a fim de evitar o uso de aplicativos.

Figura 16 - Elementos que não poderiam compor a solução

Fonte: Autor

3.1 Planta

A planta onde o trabalho foi desenvolvido diz respeito a um estabelecimento comercial localizado no aeroporto do Recife. São cabines que funcionam como quartos de hotel que são normalmente utilizadas por passageiros que precisam aguardar horas até o horário do seu voo.

No total são dez cabines e cada uma delas tem três equipamentos que são o foco desse trabalho: ar-condicionado, televisão e iluminação. No momento em que o cliente aluga uma cabine, ele recebe em seu e-mail uma URL para acessar o sistema supervisório da respectiva cabine alugada. Esse sistema de e-mail, já existente, foi aproveitado para disponibilizar o *link* da cabine de forma fácil. Em cada cabine foi

instalada uma placa de controle (o sistema embarcado), que por sua vez se comunica com o respectivo supervisório.

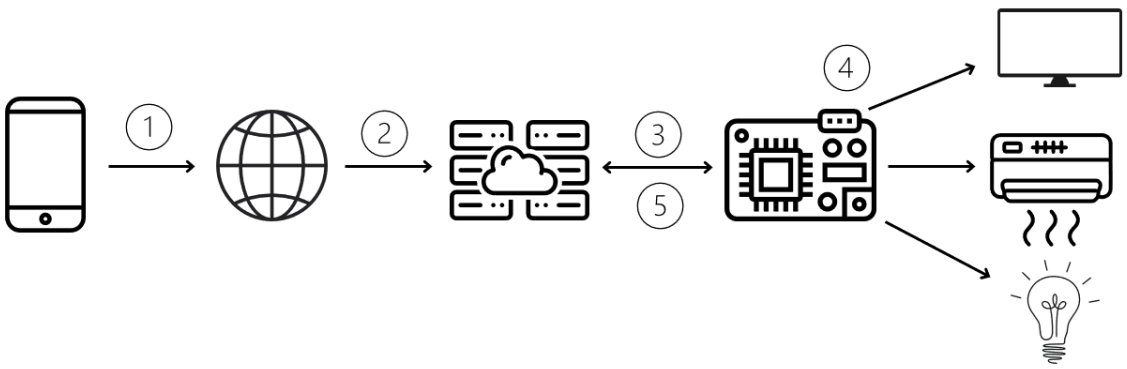
É importante ressaltar que o supervisório não está armazenado na placa, ou seja, não está armazenado na ESP32. O supervisório é uma página na web hospedado em servidores na nuvem. Pode-se comparar com qualquer site da internet que é possível acessar por algum navegador, que possui suas diversas páginas, conteúdos e funcionalidades.

Ao acessar essa URL, os usuários são direcionados para o controle da cabine. Nele, é possível controlar os três dispositivos da cabine. Para a iluminação, há um botão de ligar e desligar. Para o ar-condicionado, é possível ligar, desligar e mudar a temperatura. Para a televisão, é possível ligar, desligar, alterar volume, ativer o *mute*, alterar canal, retornar ao canal anterior e escolher um canal na lista de favoritos.

Também foi criado uma interface para o gestor do local, onde ele tem acesso ao controle de todas as cabines e opções de personalização, como alterar as cores do supervisório, alterar a lista de canais favoritos, entre outros.

Todo o funcionamento da solução depende de conexão com a internet. Essa escolha foi feita por ser a mais apropriada para o estabelecimento, já que não era possível a instalação de servidor local, por exemplo. Além do mais, se tinha como proposta que cada cliente realizasse o acesso utilizando os dados móveis do seu próprio celular ou mesmo de alguma outra rede disponível. Para tal, foi necessário deixar o supervisório em um local onde todos pudessem acessar, bastando ter conexão com a internet. O procedimento do sistema está ilustrado na Figura 17.

Figura 17 - Fluxograma geral de funcionamento do sistema proposto



Fonte: Autor

Em suma, os passos para o funcionamento são os seguintes:

1. Ao alugar uma cabine, o usuário de posse de um aparelho celular, por exemplo, conecta-se à internet;
2. Uma vez conectado à internet, é possível acessar a URL disponibilizada via e-mail e conectar-se ao supervisório da cabine;
3. Após acessar o sistema supervisório, é possível controlar os dispositivos da cabine, pressionando o botão da funcionalidade desejada. Nesse momento, o sistema supervisório envia um código, referente ao botão pressionado, para o banco de dados;
4. No momento que a escrita é realizada no banco de dados, o sistema embarcado recebe o código e o interpreta. Por exemplo, se código escrito for referente a acender a luz, a ESP32 aciona a iluminação;
5. Por fim, o sistema supervisório reescreve, no mesmo campo do banco de dados que em que houve a escrita pelo supervisório, que o comando foi executado.

Também vale ressaltar que o acesso ao supervisório pode ser realizado por qualquer dispositivo conectada à internet e que possua um navegador instalado, como computadores ou tabletes.

3.2 Supervisório Desenvolvido

O sistema supervisório desenvolvido é uma página na web hospedada na AWS. A escolha de elaborar o supervisório dessa forma foi feita para evitar a necessidade de cada cliente que alugar a cabine ter que baixar algum aplicativo ou que cada cabine tivesse algum sistema instalado para acessar o supervisório, seja um tablet ou computador. A interface é totalmente responsiva e consegue se adequar a telas de computadores, celulares e tablets.

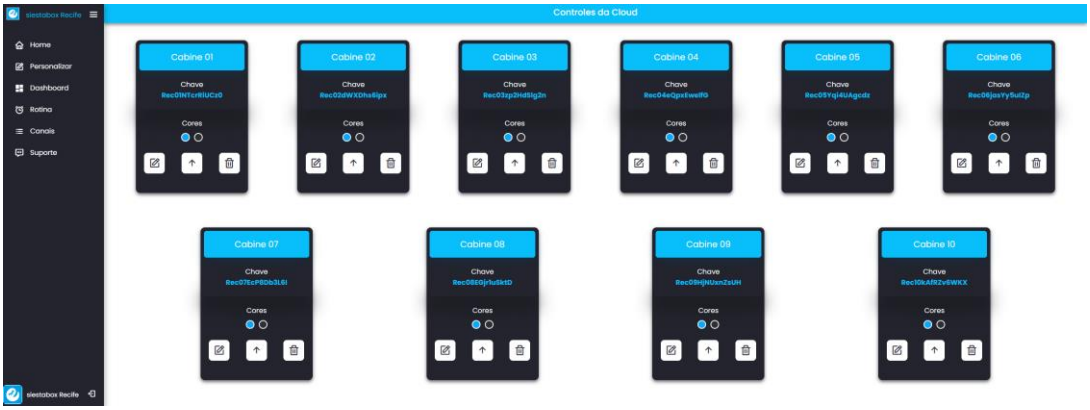
3.2.1 Layout do Supervisório

Como mencionado, existem as interfaces do gestor e do cliente. A Interface do gestor tem um menu principal onde é possível visualizar todas as cabines do estabelecimento. Como é possível visualizar na Figura 18, cada cabine está representada como uma caixa que informa o seu número, a sua chave (identificação),

a paleta de cores e os botões para personalizar, deletar e acessar a página correspondente de cada cabine individualmente.

Também existe uma barra lateral para acessar demais configurações do supervisório, como personalização geral e escolha de canais.

Figura 18 - Layout do supervisório



Fonte: Autor

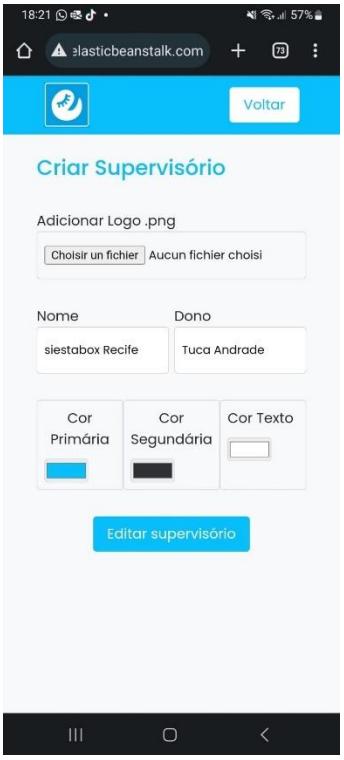
3.2.2 Interface do Gestor

A interface do gestor funciona como uma tela mestra do sistema supervisório. Naturalmente, cada cliente tem acesso à interface da respectiva cabine que foi alugada, ao passo que a interface do gestor tem acesso a todas as interfaces e diferentes configurações.

3.2.2.1 Personalização

É possível personalizar tanto o menu principal do gestor quanto cada um dos controles de cada cabine. Como visto na Figura 19, para o menu principal, há opções de adicionar logotipo, alterar o nome do estabelecimento, o nome do dono e a paleta de cores.

Figura 19 - Menu de personalização do supervisório



Fonte: Autor

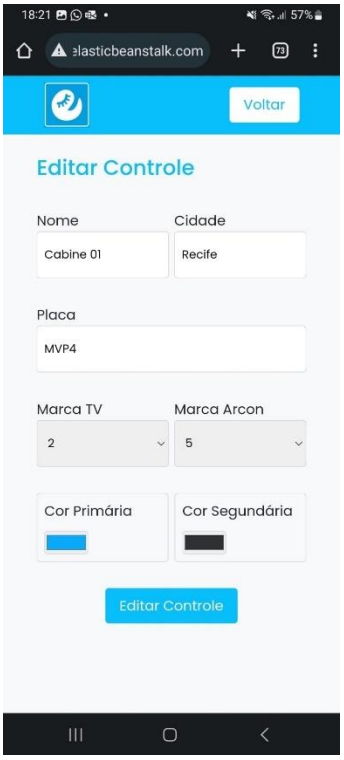
Outra opção disponibilizada para o gestor é a criação de uma lista de canais que aparece no controle das cabines. Basta preencher o nome do canal e o seu respectivo número que essa lista será disponibilizada em todos os controles. É possível adicionar e deletar os canais de forma indefinida. Esse menu está ilustrado na Figura 20.

Figura 20 - Menu de adicionar canais

Fonte: Autor

Por fim, em cada caixa que representa as cabines, há um botão para editar o controle que leve ao menu ilustrado na Figura 21. Existem as opções de alterar o nome da cabine, a cidade que ela está instalada, o nome da placa, o jogo de cores e alterar a marca da televisão e do ar-condicionado. Essa última funcionalidade será explicada com detalhes mais adiante.

Figura 21 - Menu de edição de controles



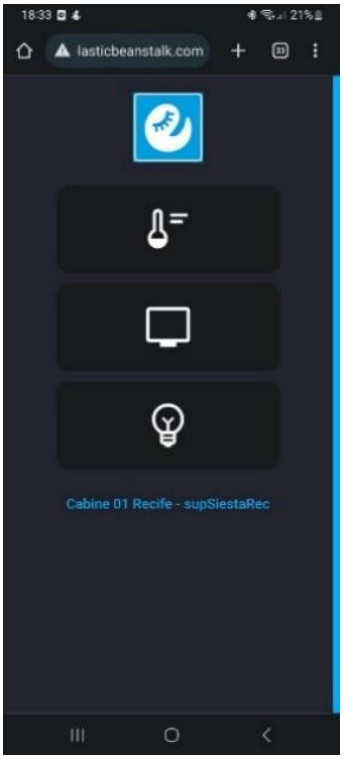
Fonte: Autor

3.2.3 Interface do Usuário

A interface do usuário nada mais é do que o controle da cabine. Existem três botões principais, referentes a cada equipamento que pode ser controlado. As cores, logo e legendas são definidas pelo gestor nos menus de configuração mostrados anteriormente. Na Figura 22 é possível visualizar a interface do usuário acessada por um aparelho celular.

Ao selecionar cada dos botões principais (ar-condicionado, televisão e iluminação), é aberto um menu de opções para que o usuário possa executar as ações do controle. A Figura 23 mostra o menu do ar-condicionado, enquanto a Figura 24 mostra o menu da televisão e a Figura 25 o menu da iluminação.

Figura 22 - Interface do usuário



Fonte: Autor

Figura 23 - Menu do ar-condicionado

Fonte: Autor

Figura 24 - Menu da televisão

Fonte: Autor

Figura 25 - Menu da iluminação

Fonte: Autor

3.3 Troca de Dados entre Supervisório e Sistema Embarcado

Existe uma árvore no *Firebase Realtime Database* para o sistema supervisório. Essa árvore pode ser entendida como diretórios aninhados. Nesse caso, o diretório principal é “dbCloud” e logo abaixo dele estão os diretórios secundários. Cada um desses diretórios pode ter seus subdiretórios e assim sucessivamente. Vale ressaltar que a estrutura específica criada no *Firebase Realtime Database* pode variar de acordo com o usuário que está montando, bem como os nomes dos diretórios e subdiretórios. Neste caso, a estrutura e nomeações são as que aparecem na Figura 26.

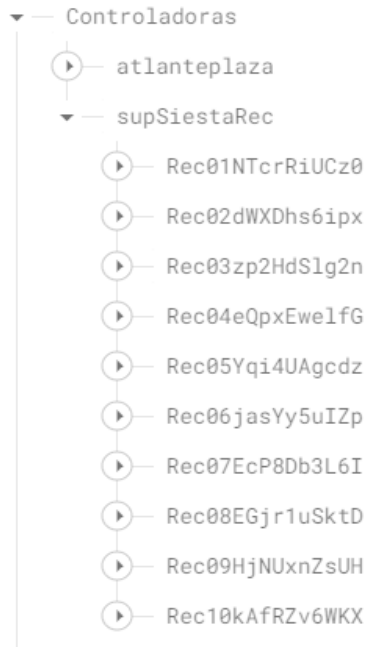
Figura 26 - Árvore geral do *Realtime Database*

Fonte: Autor

Em particular, para o funcionamento dos comandos do supervisório, o diretório mais importante é o “Controladoras”, ilustrado na Figura 26. As demais são para configurações em geral. Todas as configurações realizadas no supervisório são armazenadas nessa árvore, como a escolha das cores, número de cabines, nome do estabelecimento e assim por diante.

Dentro da chave “Controladoras”, está listado todas as cabines com seus respectivos códigos de identificação, como mostrado na Figura 27.

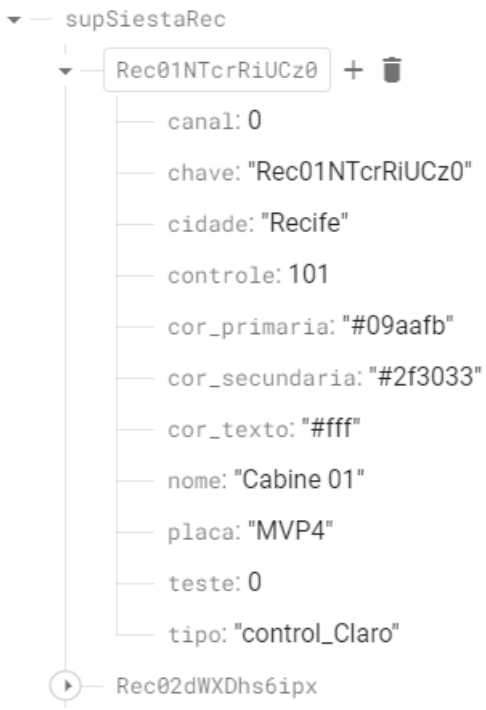
Figura 27 - Diretório das controladoras



Fonte: Autor

Mais adiante, dentro de cada subdiretório de controladoras há uma série de informações, como a identificação da cabine, a cidade, código das cores primária e secundária e nome da cabine. Esses detalhes são vistos na Figura 28 abaixo.

Figura 28 - Diretório da cabine 1



Fonte: Autor

É nessa parte do campo de dados onde é realizada a interface entre o sistema supervisório e o sistema embarcado de cada cabine. Em particular, isso é feito pelos campos “controle” e “canal”. Todos os demais campos são para armazenamento das configurações da interface do controle.

Sempre que o usuário pressiona um botão na sua tela, é enviado um valor para o campo “controle”. Por exemplo, se for pressionado o botão para ligar a lâmpada, é enviado o valor “101” para o campo “controle”. Uma vez que esse valor for escrito, a ESP32 o recebe e executa uma ação associada a ele que, neste caso, é o comando de ligar/desligar a iluminação. Maiores detalhes desse funcionamento são explorados na próxima sessão.

O campo “canal” recebe o número dos canais. Por exemplo, se o usuário desejar enviar o comando para ir até o canal 513, o campo “canal” recebe o valor “513” e, analogamente ao funcionamento do campo “controle”, a ESP32 recebe esse valor e executa a ação de enviar o comando, neste caso, mudando para o canal 513.

3.4 Sistema Embarcado Desenvolvido

Como mencionado, o sistema embarcado desenvolvido para este projeto tem o objetivo de comunicar-se com o banco de dados e controlar os dispositivos de campo, tendo como base para essas tarefas a ESP32. Apesar do projeto ter um foco de aplicação bem definido no escopo deste trabalho, todo o sistema embarcado foi construído para ser versátil, permitindo que ele possa ser aplicado em outras plantas com pequenas modificações.

3.4.1 Hardware

O *hardware* nada mais é do que a parte física do sistema embarcado. Compreende todos os componentes tangíveis, como o microcontrolador, sensores e demais elementos da placa, como capacitores e indutores. O *hardware* é responsável por permitir o funcionamento e a interação do usuário com o *software* do dispositivo.

3.4.1.1 ESP32-DevKitC V2

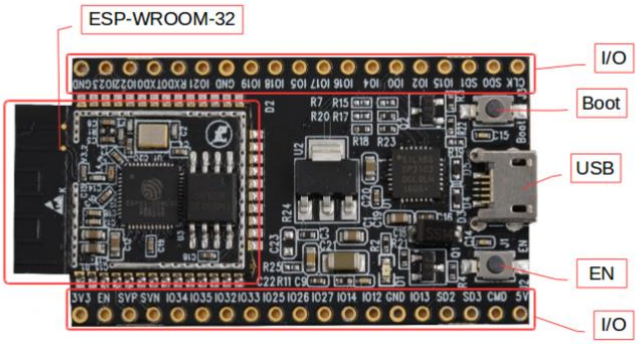
A placa que serviu de base para a construção do sistema final foi a ESP32-DevKitC V2, vista na Figura 29 . Ela é uma pequena placa produzida pela *Espressif Systems* baseada no módulo ESP-WROOM-32. Ela foi utilizada em razão da sua

robustez e por já possuir elementos necessários para o funcionamento da ESP32, como conversor USB-Serial para descarregar o programa, botões para *boot* e *reset* e demais componentes para o funcionamento do módulo que se encontra integrado na placa.

A ESP-WROOM-32 foi o módulo escolhido por suas especificações. Trata-se de um MCU bastante poderoso que possui integrado compatibilidade para comunicação WiFi, arquitetura de 32 bits, grande capacidade de armazenamento e processamento, além de expressivo número de GPIOs, totalizados em 36.

O *firmware* do sistema requer bom poder de processamento, visto que ele está a todo momento conectado à rede WiFi e comunicando com o *Firebase*, além da aplicação exigir um bom espaço de armazenamento. A Figura 30 abaixo mostra um comparativo entre a ESP32 e outros MCUs disponíveis no mercado e sua superioridade em relação a eles.

Figura 29 - ESP32 DevKitC V2



Fonte: (ESPRESSIF SYSTEMS)

Figura 30 - Comparativo de diferentes microcontroladores

	ESP32	ESP8266	ARDUINO UNO R3
Cores	2	1	1
Arquitetura	32 bits	32 bits	8 bits
Clock	160MHz	80MHz	16MHz
WiFi	Sim	Sim	Não
Bluetooth	Sim	Não	Não
RAM	512KB	160KB	2KB
FLASH	16Mb	16Mb	32KB
GPIO	36	17	14
Interfaces	SPI / I2C / UART / I2S / CAN	SPI / I2C / UART / I2S	SPI / I2C / UART
ADC	18	1	6
DAC	2	0	0

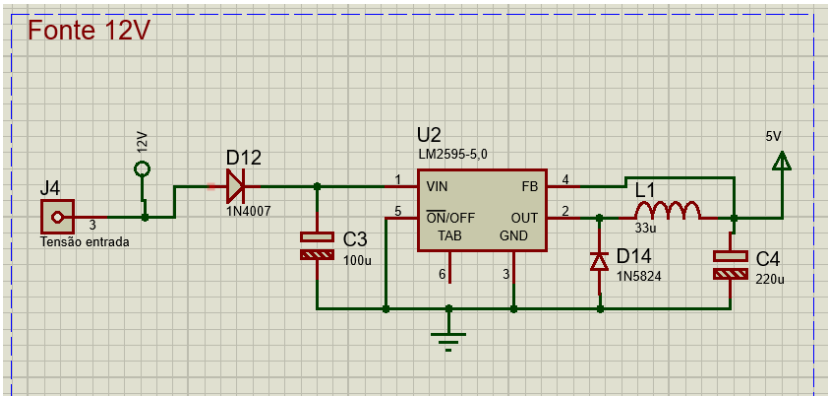
Fonte: (KOYANAGI, 2017)

3.4.1.2 Alimentação

O módulo escolhido da ESP32 utilizado na placa é normalmente alimentado pela interface USB quando este está conectado a um computador, por exemplo. Como a ideia da solução é ser instalada em quartos onde, normalmente, só há disponibilidade de tomadas ligadas à rede de 220 V AC, foi preciso desenvolver um circuito que pudesse alimentar a placa por meio de uma fonte típica de 12 V comumente encontrada no comércio local.

Esse circuito, visto na Figura 31, é baseado no CI LM2595 da *Texas Instruments*, que nada mais é do que um regulador de tensão baseado em um conversor *step-down* (*buck*). Conversores *step-down* são utilizados para reduzir a tensão de entrada DC. No caso desse projeto, ele foi construído para reduzir a tensão de entrada de 12 V, oriunda da fonte, para 5 V, com o intuito de alimentar a placa sempre que a fonte for conectada à rede.

Figura 31 - Circuito de alimentação



Fonte: Autor

3.4.1.3 Transmissor Infravermelho

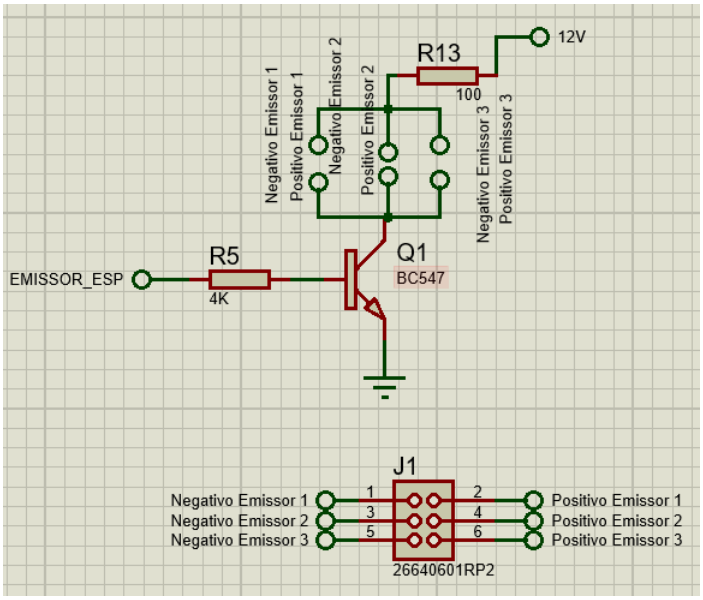
Para realizar o controle do ar-condicionado e televisão, é necessário o uso dos transmissores infravermelhos. O transmissor é um LED que emite sua luz na faixa do infravermelho. Ele é conectado em uma das GPIOs do microcontrolador e, uma vez que o MCU receber o comando de disparar um código infravermelho, o LED emite os comandos conforme protocolo descrito anteriormente.

Como os GPIOs da ESP32 fornecem uma tensão de 3,3 V, foi montado um

circuito para amplificar a corrente que passa pelo LED infravermelho. A configuração escolhida foi um circuito emissor comum com um transistor BC547. Outro artifício utilizado para aumentar a potência do LED foi alimentá-lo à fonte de 12 V. Normalmente, a própria placa fornece tanto a tensão de alimentação quanto o sinal de comando para esses dispositivos.

O intuito desse circuito é fazer com que o LED tenha um brilho forte, de modo que ele possa ser instalado a certa distância do equipamento e este conseguir receber e interpretar corretamente o sinal emitido. O circuito para pode ser visto na Figura 32.

Figura 32 - Circuito do transmissor infravermelho



Fonte: Autor

3.4.1.4 Módulo Relé

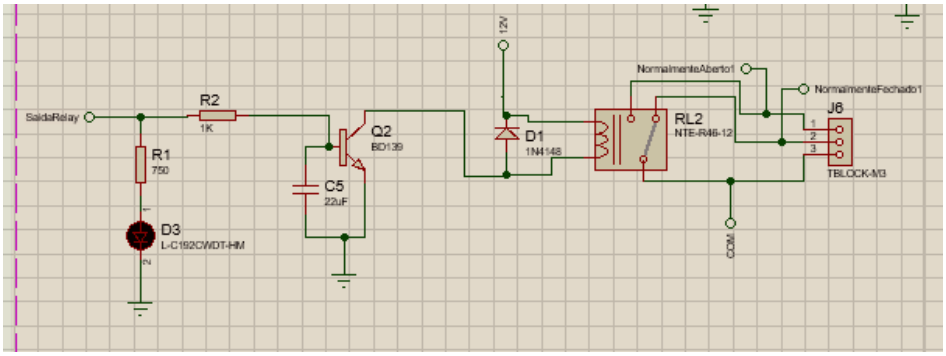
Relé é um dispositivo eletromecânico utilizado para controle de cargas. Internamente, ele possui dois contatos, NO e NC, e um eletroímã. Quando uma tensão é aplicada nos terminais do eletroímã, ele atua como um indutor e gera um campo eletromagnético, atraindo a haste metálica dos contatos para a sua proximidade. A depender da ligação realizada, o contato abre ou fecha.

As GPIOs da ESP32 fornecem no máximo uma corrente de 40 mA. Então, aqui também é interessante a utilização de um transistor para amplificar a corrente, visto que o relé possui uma bobina que é acionada por uma corrente relativamente alta. Ao utilizar um transistor para controlar o relé, é possível também isolar eletricamente o

circuito de controle do circuito de acionamento, protegendo o MCU de surtos de tensão, por exemplo.

Ao aplicar um sinal de comando na base do transistor, o relé é acionado. Aplicando novamente, ele é desligado. Essa mudança de nível lógico alto para nível lógico baixo gera uma indutância que tende ao infinito, ou seja, o campo magnético gera uma alta tensão inversa que pode resultar em surtos ao circuito. Também com o intuito de proteger os dispositivos de controle e acionamento, é adicionado um diodo em paralelo entre os terminais da bobina, que terá uma queda de tensão direta no momento que a bobina é desenergizada. O circuito para acionamento do relé está ilustrado na Figura 33.

Figura 33 - Circuito do módulo relé

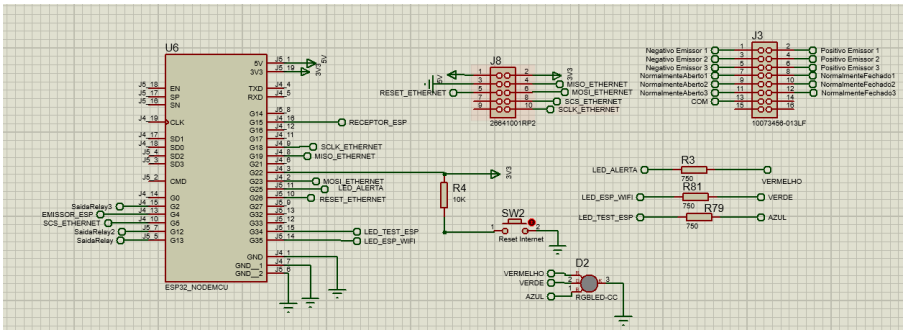


Fonte: Autor

3.4.1.5 Funcionamento Geral da Placa

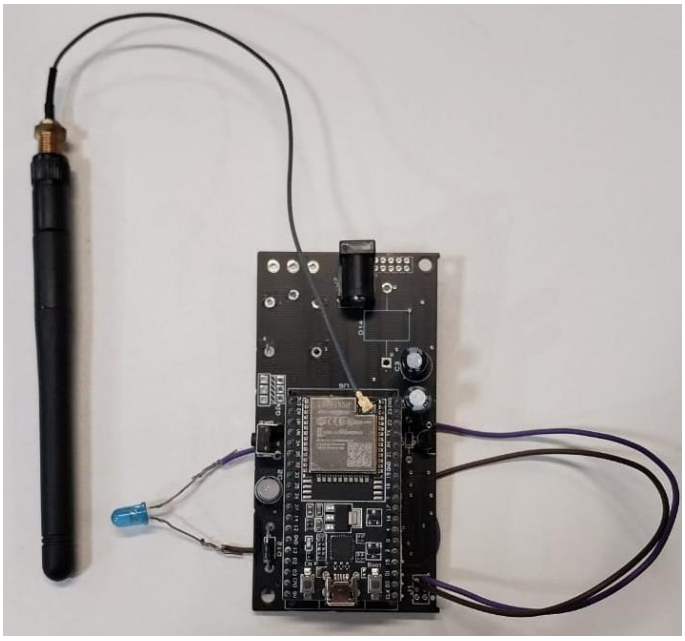
Como já mencionado, o módulo ESP32 irá controlador os dois circuitos demonstrados anteriormente. Cada um deles está ligado a uma GPIO da placa, como visto na Figura 34. Além desses circuitos, também existem alguns LEDs indicativos, que servem como auxílio visual para tratativas de problemas que venham a surgir. Por exemplo, há um led RGB que muda de cor a depender se a placa está ou não conectada à internet e comunicando com o *Firebase*, além de outro LED para indicar se o módulo relé está acionado. Finalmente, a placa já produzida em circuito impresso e com alguns periféricos conectados (antena para conexão na internet e LED infravermelho) pode ser vista na Figura 35.

Figura 34 - Circuito geral da placa



Fonte: Autor

Figura 35 - Sistema embarcado pronto



Fonte: Autor

3.4.2 Firmware

O *firmware* é o *software* no dispositivo de hardware que executa todas as funções da placa, desde as tarefas mais básicas como acionar uma saída até as mais complexas como conexão WiFi e comunicação com o banco de dados.

3.4.2.1 Conexão WiFi

A conexão com a internet só pode ser feita via WiFi e isso acontece sempre que a placa é ligada. Para tal, a ESP32 funciona como uma estação WiFi (STA), sendo possível a conexão com outras redes, nesse caso, a internet disponibilizada pelo roteador. Nesse cenário, o roteador designa um IP à placa e então é possível

comunicar-se com outros dispositivos conectados nessa mesma rede ou requisitar e publicar informações na internet, que é o ponto mais importante para esse trabalho.

Para o projeto, como só havia necessidade de conexão com uma única rede, as credenciais (SSID e senha) foram escritas diretamente no código. No momento que essa conexão for estabelecida, em seguida é possível comunicar com o *Firebase* e trocar informações com o *Realtime Database.*

3.4.2.2 Conexão e Comunicação com Firebase

Após ser estabelecida uma conexão com a internet, para a conexão com o *Firebase*, é necessário a chave API que está disponibilizada nessa aplicação criada, a URL do *Realtime Database* e credenciais de acesso para fins de autenticação. De maneira análoga a fazer *login* em algum site, o programa na ESP32 requisitará acesso ao banco de dados especificado na URL e se as credenciais estiverem corretas, o *Firebase* libera o acesso.

É utilizado HTTP *streaming* para realizar a conexão com o banco de dados. Essa conexão pode ser feita em campos específicos do banco de dados. Por exemplo, a placa da cabine 1 só precisa estar conectada e “ouvir” as mudanças feitas na cabine 1, o mesmo é verdade para a cabine 2 e assim sucessivamente.

Como mencionado na sessão anterior, quando é pressionado qualquer botão da interface web, um valor é enviado para o campo “controle” ou “canal” e no momento que isso é feito, quase que instantaneamente, pelo HTTP *streaming*, a ESP32 recebe esse valor. Como pode ser visto na Figura 36, e lembrando que a estrutura do *Firebase* é uma árvore JSON, é justamente um JSON que é recebido e interpretado pelo *firmware.*

Figura 36 - Informações no monitor serial da ESP32 quando um comando é enviado

```
JSON enviado do supervisorio:
{
        "controle": 101
}
Iterando JSON:

Valor inteiro: 101
Nome do Campo: controle
```

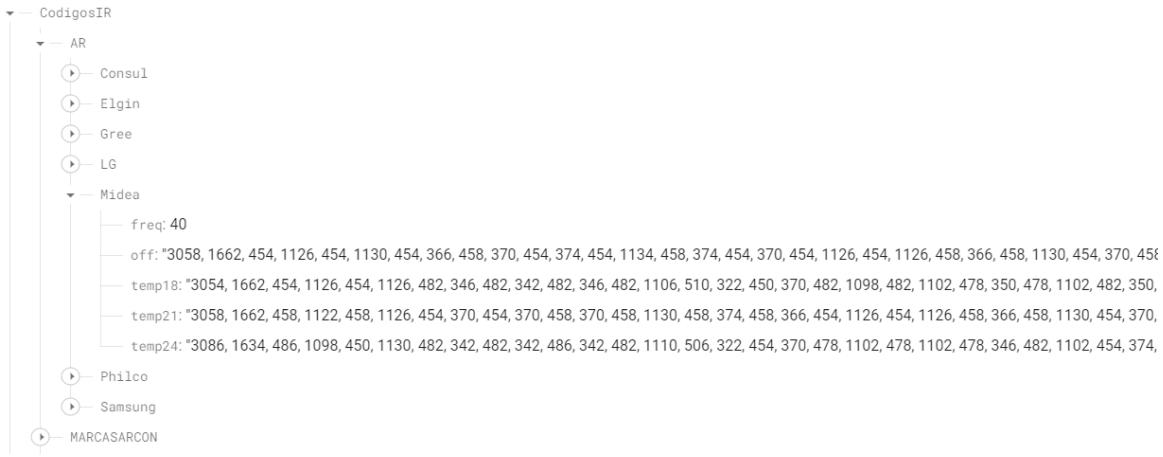
Fonte: Autor

3.4.2.3 Envio de Códigos Infravermelho

Como mencionado anteriormente, os códigos IR podem ser entendidos como sequências de espaços e marcas de diferentes durações e podem ser escritos como uma lista de números que representam essas durações em µs. No código do *firmware*, esses códigos IR são declarados como vetores de inteiros.

Foi observado que os aparelhos da cabine não seguiam o protocolo NEC, contudo, comunicavam-se com um protocolo bastante semelhante. A diferença básica era a duração das marcas e espaços para os bits de início, comando, endereço e parada, além do número de bits, mas o princípio de funcionamento era o mesmo. Então, no *Firebase*, foi criada uma árvore, que armazena códigos IR de diferentes modelos de aparelhos de ar-condicionado e televisão. A árvore construída pode ser vista na Figura 37.

Figura 37 - Base de dados de códigos IR



Fonte: Autor

A ideia de criar essa base de dados de códigos é de eliminar a necessidade de deixar fixo no *firmware* o código referente a cada comando. Isso é importante porque é plausível supor que eventualmente os aparelhos podem ser trocados e o novo modelo pode ser de uma marca diferente, o que significaria que o código gravado na placa não serviria para aquele aparelho. Nesse caso, seria necessário alguém ter que descarregar um código novo sempre que houvesse a mudança de algum equipamento.

Portanto, foi criada uma função que recebe o JSON referente a cada comando específico e converte a sequência de números e salva em um vetor de inteiros

inicialmente declarado vazio. Pelo par nome-valor, ele saberá de qual comando se trata. O valor recebido está no formato de *string*, mas como se trata de números, é possível converter cada um deles para um inteiro e armazená-los nos vetores.

Por fim, quando o usuário apertar em algum comando do supervisório e escrever no banco de dados o valor referente àquela operação, o programa envia o código desejado. Está gravada no código o que cada valor enviado pelo supervisório significa em termos de ação a ser executada.

3.4.2.4 Acionamento do Relé

O acionamento do relé funciona de forma muito mais simples. No momento em que é recebido o comando de ligar/desligar a iluminação, a placa altera o nível lógico da saída em que está conectado o módulo relé.

4. CONCLUSÕES E PROPOSTAS DE CONTINUIDADE

O uso da automação está crescendo nos últimos anos com a popularização de tecnologias IoT e a redução nos preços de dispositivos e ferramentas dedicadas a esse fim. Atualmente, a automação já é vista nos mais diversos lares e estabelecimentos, não sendo mais categorizada como um item de luxo. Contudo, percebe-se que nem sempre é fácil instalar esse tipo de sistema, seja por dificuldades em sua compreensão, utilização e/ou instalação.

Nesse contexto, foi apresentado um sistema supervisório simples e intuitivo destinado ao uso com algum microcontrolador que tenha capacidade de comunicação com a internet, no caso, a ESP32. Esse sistema visa gerenciar e monitorar diversas soluções que possam ser implementadas em automação, tanto em residências quanto em estabelecimentos comerciais. Todo o sistema foi construído de forma que ele seja versátil. Ainda, o sistema supervisório visa facilitar o controle de diversos dispositivos a partir de um só lugar.

Com mínimas modificações em *hardware,* é possível adicionar mais dispositivos para serem controlados, como mais pontos de iluminação ou pequenos motores, por exemplo. A Figura 38 também ilustra a versatilidade do sistema supervisório. Ela é a interface de controle da televisão para outro estabelecimento. Vale notar que as cores estão diferentes e possui menos botões do que a interface mostrada na Figura 24. E para adicionar na ESP32 o controle desses novos dispositivos, são poucas linhas de código a mais. Seria o caso de apenas habilitar mais entradas/saídas.

Figura 38 - Sistema supervisório de outro estabelecimento

Fonte: Autor

A criação de um sistema supervisório a partir do zero em detrimento de algum produto já disponível no mercado, se mostrou acertada. A liberdade para adaptá-lo à medida que o trabalho foi sendo desenvolvido, propiciou atender com exatidão as demandas do gestor do negócio de forma mais precisa. Não seriam todos os produtos que teriam todos os recursos desejados para esse projeto, da mesma forma que a solução final não tem todas os recursos também. Porém, tudo que era necessário foi possível ser incorporado e, caso surja a necessidade de incorporar mais funcionalidades, é possível fazê-lo, o que nem sempre é o caso quando se trabalha com uma ferramenta já pronta.

Um ponto que era de suma importância a ser alcançado dizia respeito à facilidade do uso do sistema supervisório. Todos os usuários que utilizaram e utilizam o sistema não sentem maiores dificuldades em compreender seu funcionamento.

Visto que a solução implementada utiliza tecnologias de uso geral, ela se provou ser adaptável e escalável. É possível adicionar outros atuadores e sensores sem ser necessária a alteração do *hardware* e com poucas adaptações de *firmware*, do sistema supervisório e banco de dados.

Também vale destacar o diferencial de mercado que esse tipo de solução

oferece para o gestor de um estabelecimento que a possui. Durante diversos testes em campo foi possível observar o contentamento dos clientes com o sistema desenvolvido e admiração com sua facilidade de uso e conforto oferecido.

Dessa maneira, os objetivos que pretendiam ser alcançados foram atingidos. O sistema desenvolvido é funcional e escalável. Os resultados obtidos mostram que, de fato, as soluções de automação com foco no IoT são uma tendencia e que nos próximos anos será possível ver essas tecnologias mais presentes no cotidiano.

4.1 Futuro da Solução

A solução desenvolvida atingiu um resultado satisfatório, tanto pelo lado do supervisório quanto pelo lado do sistema embarcado. No entanto, há espaço para diversas melhorias dessas duas principais ferramentas.

No que tange ao sistema supervisório, é possível explorar outras possibilidades que esse sistema oferece para o gestor, como parcerias locais e opções de suporte. Por exemplo, é possível disponibilizar o cardápio de algum restaurante, ou mesmo de vários, que se tenha firmado parceria. Também é possível acrescentar uma opção de suporte ao cliente, onde ele possa entrar em contato com a recepção, seja por chat ou por ligação. Nesse sentido, é possível expandir a interface do cliente para muito além do controle, onde esse seria apenas uma das opções de um menu principal.

A nível de *firmware*, as possibilidades também são grandes. É muito comum haver a necessidade de alteração do código. À medida que o sistema for sendo utilizado por mais tempo e por mais usuários, pontos de melhoria vão aparecendo e outras funcionalidades vão sendo incorporadas ao produto. Em um cenário onde existem vários clientes em diversas localidades geográficas, fica impraticável ter que se deslocar até cada uma delas para atualizar o *firmware*. Logo, fica nítida a necessidade de haver um sistema de atualização remota.

Outro ponto de melhoria interessante é a maneira como a placa se conecta na internet. No caso do trabalho desenvolvido, a placa só consegue conectar com uma única rede e as credenciais estão escritas no código. Se manterem o SSID e alterarem a senha, por exemplo, já não é mais possível estabelecer conexão. Para qualquer cenário que aquelas credenciais não sejam mais válidas, seria necessário descarregar outro código na placa para que ela tenha acesso a rede. Isso também torna a solução inviável para larga escala. Uma solução é, sempre que ligar a placa, a ESP32 atuar

como AP e buscar redes por perto. Então, seria possível escolher a rede desejada e estabelecer a conexão, passando a ESP32 para o modo estação novamente.

No sistema desenvolvido, é essencial ter uma conexão WiFi com a internet. Sem isso, ele não funciona. Para outras aplicações, nem sempre será interessante trabalhar desta forma. Em outros casos, pode ser que seja mais atrativo ter um servidor local, por exemplo, hospedado numa *raspberry pi*. Por outro lado, é possível que ainda seja o ideal conexão com a internet, entretanto, a melhor opção seria conexão cabeada. Assim, também é pertinente explorar essas possibilidades.

Ainda é possível estudar com mais profundidade como a solução poderia ser adaptada para o mercado de automação residencial. Apesar de ser de fácil instalação, foi observado que muitas residências não poderiam ter o aparelho instalado sem algumas modificações. O ideal é ter um produto que seja de fácil manuseio, ou seja, qualquer pessoa que adquiri-lo pode instalar em sua casa sem que seja necessária qualquer reforma. Uma das opções seria modular a placa. Ao invés de ter uma única placa e construir tudo a partir dela, ter pequenos módulos escravos que recebem comandos do módulo mestre.

Tanto para aplicações no setor hoteleiro, onde foi desenvolvido o trabalho, quanto para aplicações residenciais ou em outros setores, também é válido a implementação de rotinas personalizadas. Por exemplo, acender ou apagar uma lâmpada em determinado horário do dia, ligar ar-condicionado e até criar perfis de usuário com suas preferências, como lista de canais mais assistidos e temperatura média que costuma permanecer o ar-condicionado.

Fica evidente que existem inúmeras possibilidades para expandir e melhorar a solução, deixando-a mais completa para o setor hoteleiro e tornando-a mais atrativa para o mercado residencial.

REFERÊNCIAS

ALTIUM. **NEC Infrared Transmission Protocol**, 2017. Disponivel em: https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol. Acesso em: ago. 2023.

AMAZON. **O que é hospedagem Web?** Disponivel em: https://aws.amazon.com/pt/what-is/web-hosting/. Acesso em: ago. 2023.

AMAZON. **O que é a computação em nuvem?** Disponivel em: https://aws.amazon.com/pt/what-is-cloud-computing/. Acesso em: ago. 2023.

AMAZON. **O que é armazenamento na nuvem?** Disponivel em: https://aws.amazon.com/pt/what-is/cloud-storage/. Acesso em: 2023.

AMAZON. **O que é um banco de dados de chave-valor?** Disponivel em: https://aws.amazon.com/pt/nosql/key-value/#:~:text=Um%20banco%20de%20dados%20de%20chave-valor%20é%20um%20tipo,funciona%20como%20um%20identificador%20exclusivo. Acesso em: ago. 2023.

AMEBAIOT. **IR - Transmit IR NEC Raw Data and Decode**, 2023. Disponivel em: https://www.amebaiot.com/en/amebad-arduino-irdevice/. Acesso em: ago. 2023.

BATISTA DA CRUZ, H.; DE CARVALHO RUTZ DA SILVA, S.; LUIZ RUTZ DA SILVA, S. **Guia de Programação ESP32**, 2022. Disponivel em: https://repositorio.utfpr.edu.br/jspui/bitstream/1/31330/4/roboticaaprendizagemprojetos_produto.pdf. Acesso em: ago. 2023.

DEVMEDIA. **ASP.NET Web API Criando um web service com várias entidades**, 2023. Disponivel em: https://www.devmedia.com.br/exemplo/asp-net-web-api-criando-um-web-service-com-varias-entidades/48. Acesso em: ago. 2023.

DIAS, M. **SERVER SENT EVENTS**, 2023. Disponivel em: https://marcdias.com.br/server-sent-events/. Acesso em: ago. 2023.

DIGIKEY. **ESP32-WROOM-32U-N8**, 2023. Disponivel em: digikey.com.br/pt/products/detail/espressif-systems/ESP32-WROOM-32U-N8/9381720. Acesso em: ago. 2023.

ESPRESSIF SYSTEMS. **ESP32-DevKitC V2 Getting Started Guide**. Disponivel em: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc-v2.html. Acesso em: ago. 2023.

FERREIRA, G. **HTTP Desmistificando o protocolo da Web**, 2022. Disponivel em: https://www.alura.com.br/artigos/desmistificando-o-protocolo-http-parte-1. Acesso em: ago. 2023.

GARCIA, F. D. **Introdução aos sistemas embarcados e microcontroladores**, 2018. Disponivel em: https://embarcados.com.br/sistemas-embarcados-e-microcontroladores/. Acesso em: ago. 2023.

GOOGLE. **Firebase Realtime Database**. Disponivel em: https://firebase.google.com/docs/database?hl=pt-br. Acesso em: ago. 2023.

JSON. **Introdução ao JSON**, 2023. Disponivel em: https://www.json.org/json-pt.html. Acesso em: ago. 2023.

KOYANAGI, F. **Introdução ao ESP32**, 2017. Disponivel em: https://www.fernandok.com/2017/11/introducao-ao-esp32.html. Acesso em: ago. 2023.

LEITE, G. **Saiba o que é JSON e como utilizar**, 2023. Disponivel em: https://www.alura.com.br/artigos/o-que-e-json. Acesso em: ago. 2023.

MOZILLA. **Uma visão geral do HTTP**, 2023. Disponivel em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview#fluxo_http. Acesso em: ago. 2023.

MYSQLTUTORIAL. **MySQL Temporary Table**. Disponivel em: https://www.mysqltutorial.org/mysql-temporary-table/. Acesso em: ago. 2023.

ORACLE. **O que é um Banco de Dados?** Disponivel em: https://www.oracle.com/br/database/what-is-database/. Acesso em: ago. 2023.

ORACLE. **O que é um banco de dados relacional (RDBMS)?** Disponivel em:

https://www.oracle.com/br/database/what-is-a-relational-database/. Acesso em: ago. 2023.
ORACLE. **O que é NoSQL?** Disponivel em: https://www.oracle.com/br/database/nosql/what-is-nosql/. Acesso em: ago. 2023.
PUBNUB. **What is HTTP Streaming?**, 2023. Disponivel em: https://www.pubnub.com/guides/what-is-http-streaming/. Acesso em: ago. 2023.
REIS, F. D. **Introdução aos Sistemas Embarcados**, 2015. Disponivel em: http://www.bosontreinamentos.com.br/eletronica/eletronica-geral/introducao-aos-sistemas-embarcados/. Acesso em: ago. 2023.
ROUSE, M. **Push Technology**, 2012. Disponivel em: https://www.techopedia.com/definition/5732/push-technology. Acesso em: ago. 2023.
SOARES, M. J. **MICROCONTROLADORES PIC - Teoria - Parte 1**, 2002. Disponivel em: http://www.arnerobotics.com.br/eletronica/Microcontrolador_PIC_teoria_1.htm. Acesso em: ago. 2023.
W3SCHOOLS. **HTML SSE API**, 2023. Disponivel em: https://www.w3schools.com/html/html5_serversentevents.asp#:~:text=A%20server-sent%20event%20is,events%2C%20the%20updates%20come%20automatically. Acesso em: ago. 2023.