INSTITUTO FEDERAL DE SANTA CATARINA

ANDREY GONÇALVES

Sistema de monitoramento e alertas emergenciais para idosos

São José - SC

Julho/2023

SISTEMA DE MONITORAMENTO E ALERTAS EMERGENCIAIS PARA IDOSOS

Monografia apresentada ao Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Orientador: Mario de Noronha Neto

São José - SC

Julho/2023

Andrey Gonçalves

Sistema de monitoramento e alertas emergenciais para idosos

Este trabalho foi julgado adequado para obtenção do título de Engenheiro de Telecomunicações, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São José - SC, 07 de Julho de 2023:

Mario de Noronha Neto
Instituto Federal de Santa Catarina

Arliones Hoeller Jr, Dr.
Instituto Federal de Santa Catarina

Odilson Tadeu Valle, Dr.
Instituto Federal de Santa Catarina

À minha família, minha razão de viver.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus. Agradeço ao meu orientador Mario de Noronha Neto por aceitar conduzir o meu trabalho de conclusão de curso.

A todos os meus professores do curso de Engenharia de Telecomunicações da Instituto Federal de Santa Catarina (Câmpus São José) pela excelência da qualidade técnica de cada um.

Aos meus pais Gentil João Gonçalves e Claudete Maria Pereira Gonçalves que sempre estiveram ao meu lado me apoiando ao longo de toda a minha vida e trajetória acadêmica.

À minha esposa Morgana Schneider pela compreensão e paciência demonstrada durante o período do curso.

A vida é para quem topa qualquer parada. Não para quem para em qualquer topada.
Bob Marley

RESUMO

Com o envelhecimento da população, surgiram lacunas no cuidado e monitoramento de idosos. Felizmente, o avanço tecnológico possibilitou a criação de sistemas de monitoramento, preenchendo essas lacunas. Tendo em vista isso, a proposta do projeto é apresentar uma solução para auxiliar o monitoramento de pessoas idosas e seu ambiente domiciliar. A ideia principal consiste no desenvolvimento de um *gateway* Message Queuing Telemetry Transport (MQTT) utilizando uma Raspberry Pi 4 com um módulo de comunicação *Zigbee* para processar os sinais de diversos sensores *Zigbee* de mercado e uma *stack* de um cliente *Session Initiation Protocol* (SIP) para comunicação de voz em casos emergenciais ou para assistência ao contratante. As informações de monitoramento do idoso e do ambiente será exibida em um *Dashboard* na plataforma TagoIO com uma interface amigável, onde o encarregado pelo monitoramento pode verificar em tempo real os dados coletados no ambiente de instalação. Vale ressaltar que os dados coletados para desenvolvimento do sistema são de sensores de temperatura, umidade, fumaça, gás de cozinha, porta, botão de emergência, vibração para detecção de quedas, vazamento de água e presença. Além disso o produto tem duas botoeiras que podem ser acionadas para ligar para números pre-definidos, como uma central de atendimento de saúde ou algum familiar. O protótipo possui alto falante e microfone embutido e é capaz de originar e receber chamadas SIP.

Palavras-chave: Telecomunicador SIP. Monitoramento de idosos. Sensores *Zigbee*.

ABSTRACT

With the aging of the population, gaps emerged in the care and monitoring of the elderly. With technological advances, it has enabled the creation of monitoring systems. In view of this, the proposal of the project is to present a solution to assist the monitoring of elderly people and their home environment. The main idea is to develop a MQTT gateway using a Raspberry Pi 4 with a module *Zigbee* communication device to process the signals from various market *Zigbee* sensors and a *stack* from a SIP customer for voice communication in emergency cases or for assistance to the contracting party. The monitoring information of the elderly person and the environment will be displayed in a *Dashboard* on the platform TagoIO with a friendly interface, where the person in charge of monitoring can check in time real data collected in the installation environment. It is noteworthy that the data collected for system development are from temperature, humidity, smoke, cooking gas, door, emergency button, vibration sensors for detecting falls, water leakage and presence. In addition, the product has two pushbuttons that can be activated to call predefined numbers, such as a health care center or a family member. The product has a built-in speaker and microphone and is capable of making and receiving SIP calls.

Keywords: Telecom SIP. Elderly monitoring. *Zigbee* sensors.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

4G Quarta Geração de telefonia móvel.

API Application Programming Interface.

CSS Cascading Style Sheets.

GPIO General Purpose Input/Output.

GSM Sistema Global para Comunicações Móveis.

HTML Linguagem de Marcação de HiperTexto.

HTTP Hypertext Transfer Protocol.

IoT Internet of Things.

MQTT Message Queuing Telemetry Transport.

OMS Organização Mundial da Saúde.

PABX IP Private Automatic Branch Exchange over Internet Protocol.

PSTN Public Switched Telephone Network.

RF Radiofrequência.

RTP Real-time Transport Protocol.

SDP Session Description Protocol.

SIP *Session Initiation Protocol*.

VoIP Voz sobre Protocolo de Internet.

SUMÁRIO

1 INTRODUÇÃO

O envelhecimento da população está acontecendo mais rápido do que nunca. Segundo o relatório mundial de envelhecimento e saúde da Organização Mundial da Saúde (OMS) (OMS, 2015), em 2050 o número de pessoas com idade superior a 60 anos será o dobro que em 2015, esta fatia será cerca de um quinto da população do mundo, por volta de 2 bilhões idosos.

Uma grande proporção da população idosa sofre de problemas de saúde relacionado à idade, como a doença de Alzheimer, demência, doença cardiovascular, diabetes ou outras doenças crônicas. Além disso temos o declínio nas habilidades físicas e cognitivas dos idosos podendo levar a quedas. Mas com o avanço tecnológico é possível criar sistemas para monitoramento e comunicação com os idosos em sua residência.

O mercado brasileiro oferece alguns sistemas e serviços de cuidados de idosos e pessoas debilitadas, porém são soluções com tecnologias antigas, baseadas em linhas de telefonia analógica denominada Public Switched Telephone Network (PSTN), telefonia Sistema Global para Comunicações Móveis (GSM) ou Quarta Geração de telefonia móvel (4G). A solução proposta nesse artigo é baseado em tecnologia digital, com integração completa pela internet, o que traz ao protótipo maior confiabilidade e velocidade para a tomada de decisões. Agregando estas novas tecnologias, também é possível ter uma gama de cuidados e prevenções muito maiores. O protótipo foi desenvolvido com a concepção de uso para uma plataforma inovadora social de idosos, agregando sensores para detecção de riscos e rápido atendimento. O produto possuirá um intercomunicador IP que permite fazer ligações para uma central de atendimento ou algum contato pré determinado. Através dos sensores, é monitorado possíveis situações de risco no ambiente ou com o usuário do sistema e estes dados são enviados via rádio comunicação *Zigbee* para o intercomunicador que imediatamente reporta como um alarme para a central. Um *dashboard* de monitoramento para verificar os dados coletados pelos sensores será desenvolvido utilizando a plataforma TagoIO Além disso o produto será um comunicador SIP com alto falante e microfone embutido, onde é possível originar uma chamada de emergência ou até mesmo requisitar algum tipo de atendimento com uma central de monitoramento de saúde. Desta forma a solução oferece um conjunto completo de serviços e soluções de monitoramento remoto para ajudar os idosos a manter uma vida saudável e segura. Podendo ter serviços de tele assistência projetados especificamente para ajudar os idosos a manter sua independência e viver com segurança em sua casa.

1.1 Objetivo geral

Desenvolver um sistema de monitoramento e alertas emergenciais para idosos utilizando a tecnologia *Zigbee* para comunicação RF e os protocolos MQTT e SIP para a comunicação de dados entre o *gateway* e a aplicação.

1.2 Objetivos específicos

- Analisar a viabilidade técnica para o desenvolvimento do sistema
- Entender o funcionamento das tecnologias *Zigbee* e dos protocolos MQTT e SIP
- Especificar a plataforma de desenvolvimento utilizada e adquirir os componentes do sistema
- Desenvolver e testar isoladamente os componentes do sistema
- Desenvolver o *front-end* do sistema integrado
- Integrar os componentes do sistema e realizar testes com o sistema integrado

1.3 Estrutura do trabalho

O documento é estruturado da seguinte forma: No Capítulo 1 do documento encontra-se a introdução e o objetivo a ser atingido. Em seguida, no Capítulo 2 é reservada para fundamentação teórica onde é exposto as tecnologias a serem utilizadas e os produtos existentes no mercado. No Capítulo 3 é exibido a apresentação da proposta desenvolvida descrevendo os componente utilizados para criação do protótipo. O Capítulo 4 as atividades desenvolvidas . E finalmente no Capítulo 5 a conclusão da monografia.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ZigBee

Conforme aponta Stoll (2008), o *ZigBee* é uma norma técnica, baseado no padrão *IEEE 802.15.4*, para comunicações sem fio de baixa potência e curta distância. O protocolo *ZigBee* foi desenvolvido pelo grupo *ZigBee Alliance* , uma associação sem fins lucrativos com o objetivo de "construir uma norma aberta com aplicações que permitam interoperabilidade entre diversos fabricantes, apoiar a base para certificação de produtos compatíveis com *ZigBee* e promover a tecnologia" (STOLL, 2008) .

Podemos encontrar a pilha*ZigBee* em diversas aplicações, sendo algumas delas relacionadas com automação e controle predial, controle industrial, periféricos para PC, controle remoto de produtos eletrônicos, automação residencial e comercial e, saúde pessoal, conforme pode ser visualizado na Figura 1(ANDRIGHETTO, 2008).

Figura 1 – Aplicações para o *ZigBee*.

Fonte: ROGERCOM, 2010 apud (KINDERMANN; SANTOS, 2012)

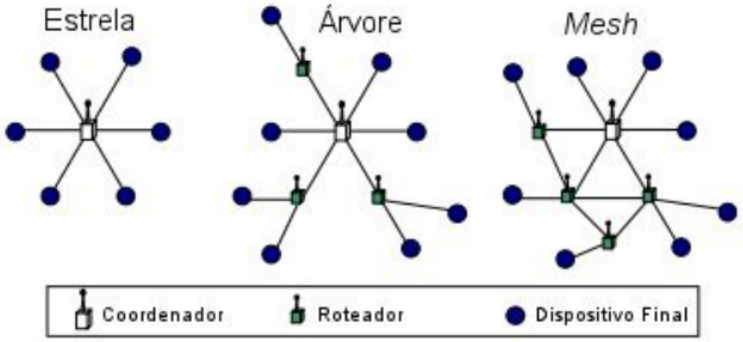
O padrão *IEEE 802.15.4* define para as redes *ZigBee* dois tipos de dispositivos, sendo eles classificados como *RFD (Reduced Function Device)* ou *FFD (Full Function Device)*. A primeira classificação é referente aos dispositivos que podem funcionar apenas como *end-pointings* da rede, pois podem apenas ter comunicação com dispositivos *FFD*, sendo assim, são dispositivos mais simples e de menor custo, pois visam um consumo de energia reduzido. Os dispositivos *FFD*, por sua vez, estão aptos a funcionar em qualquer um dos modos de configuração padrão da rede *ZigBee* (coordenador, roteador ou dispositivo final), exigindo um hardware mais potente, pois são configurados para utilizar todos os recursos da pilha protocolar, tendo um consumo maior de energia (KINDERMANN; SANTOS, 2012).

Sendo assim, os dispositivos que pertencem a uma rede *ZigBee* são classificados de acordo com a configuração padrão a que pertencem, sendo elas conforme a seguir:

- Dispositivo final *ZigBee* (DFZ) - atua como um simples dispositivo na rede;
- Roteador *ZigBee* (RZ) - possui as funcionalidades de roteamento da rede;
- Coordenador *ZigBee* (CZ) - dispositivo que gerencia a rede em que está inserido.

Além disso, a rede *ZigBee* suporta as topologias de rede em estrela, árvore e em malha, conforme pode ser observado na Figura 2 (SANTOS, 2015).

Figura 2 – Topologias de redes *ZigBee*.



Fonte: BARONTI, 2007 apud (SANTOS, 2015).

A topologia de rede em estrela é projetada com foco na comunicação de um nó para vários nós. A topologia em árvore utiliza de mecanismos hierárquicos de árvore de roteamento e, as tedes *mesh* utilizam um método de encaminhamento misto, tendo como uma das principais qualidades a redundância na rede, fazendo com que caso a comunicação entre alguns dispositivos falhe, o roteamento dos dados na rede não será afetado (SANTOS, 2015).

As aplicações *ZigBee* diferem das *Wi-Fi* e *Bluetooth Classic* em relação às implementações com distâncias mais longas e custos por dispositivos mais baixos. O protocolo foi desenvolvido para utilização em sistemas embarcados que necessitem de curto alcance de comunicação, trabalhando com faixas de distância entre 30m e 100m para ambientes internos e atingindo até 1500m em áreas abertas (KINDERMANN; SANTOS, 2012).

Kindermann e Santos (2012) trazem que o protocolo *ZigBee* opera nas faixas de frequência *ISM (Industrial, Scientific and Medical)*, sendo elas:

- 784 MHz na China
- 868 MHz na Europa;
- 915 MHz América e Austrália;

- 2,4 GHz para o resto do mundo.

Além disso, o padrão *ZigBee* não necessita de licenciamento para seu funcionamento, permitindo que a tecnologia seja usada livremente. Sua rede possui características de rede e criptografia de dados AES (Advanced Encryption Standard) de 128 bits, garantindo segurança e proteção das informações transmitidas. Oferece uma notável imunidade a ruídos, minimizando as chances de interferência entre os dispositivos configurados em redes distintas ou que utilizem da mesma faixa de frequência, porque ele é menos propenso a ser afetado por interferências devido ao seu espectro espalhado. (KINDERMANN; SANTOS, 2012).

2.2 MQTT

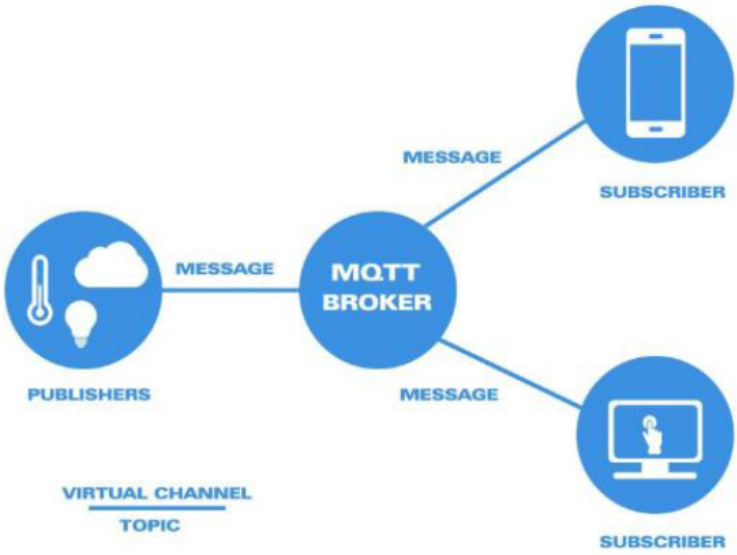
Martins e Zem (2015, p. 72) conceituam que:

> "na Internet ' tradicional' existem diversos protocolos de comunicação responsáveis por gerenciar a transferência de dados em uma rede que conecta dois ou mais computadores. Para citar alguns exemplos: HTTP, FTP e SFTP. Quando se fala em comunicação entre dois ou mais dispositivos (ou um conjunto de aplicações) conectados em uma rede também surge a necessidade de pensar em um protocolo que gerencie esta comunicação, isto é, a troca de mensagens e/ou dados entre os elementos que compõem esta rede de objetos de forma eficiente considerando as características e limitações impostas pelo ambiente" (MARTINS; ZEM, 2015, p. 72).

Dessa forma, surgem protocolos de transferência que podem ser utilizados em ambientes restritos, sendo um deles, o protocolo *MQTT (Messaging Queue Telemetry Transport)*. O *MQTT* foi criado em meados de 1999 e trata-se de um protocolo voltado para dispositivos restritos e redes inseguras, com baixa largura de banda e alta latência, sendo baseado na arquitetura *publish/subscribe* (MARTINS; ZEM, 2015).

Na arquitetura *publish/subscribe*, a qual o protocolo MQTT é baseado, o dispositivo é responsável por enviar as informações ao servidor - também chamado como *broker* - , que opera como intermediário retransmitindo as informações recebidas aos clientes interessados (CONCEIÇÃO; COSTA, 2019). A Figura 3, ilustra o modelo de comunicação utilizado pelo protocolo MQTT, onde o tópico MQTT é uma forma de categorizar e direcionar as mensagens em um sistema de comunicação baseado no protocolo MQTT. Ele permite que dispositivos IoT compartilhem informações relevantes através de uma estrutura hierárquica de tópicos. As mensagens são publicadas em tópicos específicos e os dispositivos interessados se inscrevem nesses tópicos para receber as mensagens correspondentes.

Figura 3 – Modelo de comunicação MQTT.



Fonte: (VALENZUELA et al., 2021).

Situado na camada de aplicação da arquitetura *TCP/IP*, o protocolo de comunicação MQTT pode ser aplicado sobre qualquer tecnologia, possibilitando a interconexão de dados e interações em tempo real entre sistemas, softwares e máquinas e, devido à isso desempenham um papel de alta importância em aplicações voltadas para projetos de Internet das Coisas (Internet of Things (IoT)) (VALENZUELA et al., 2021)).

Como principais características, que permitem uma boa compatibilidade do protocolo nas aplicações IoT, Valenzuela et al. (2021, p. 43410) elenca:

> "- Simples configuração: é possível fazer configurações com linhas de comando de baixa complexidade, sem a perda de padronização.
> - Comunicação bidirecional: qualquer integrante de uma rede com MQTT pode desempenhar o papel de receptor e/ou de fornecedor de informação.
> - Modelo publisher/subscriber: as informações são enviadas na estrutura de tópicos, assim o receptor não recebe informações que não sejam úteis para seu funcionamento.
> - Assíncrono: o sistema utilizado permite um funcionamento em que o fornecedor de informações não necessita de confirmações para funcionar.
> - Segurança: o protocolo MQTT permite a utilização de ferramentas para a segurança da rede, como por exemplo designar dados de login e senha para cada cliente conectado à rede. Em adicional, os brokers que são parte do protocolo podem utilizar ferramentas de criptografia.
> - Baixo consumo de energia [...]
> - Bom desempeno em áreas com problemas de transmissão."

As mensagens de comando MQTT, por sua vez, possuem um cabeçalho fixo composto de dois bytes, onde o primeiro byte identifica o tipo da mensagem e campos marcadores como entrega duplicada (*DUP*), qualidade de serviço (*QoS*) e *RETAIN* (MARTINS; ZEM, 2015). A estrutura do cabeçalho fixo das mensagens do protocolo pode ser visualizada na Figura 4.

Figura 4 – Estrutura do cabeçalho fixo da mensagem MQTT.

bit	7	6	5	4	3	2	1	0
Byte 1	Tipo da Mensagem				Flag DUP	Nível QoS		RETAIN
Byte 2	Largura restante							

Fonte: (MARTINS; ZEM, 2015).

Em relação do primeiro byte, do bit 7 ao 4 é representado o tipo da mensagem. Os quatro bits restantes do primeiro byte dividem-se em quatro campos que indicam as preferências definidas antes do envio da mensagem, sendo:

- *Duplicate delivery (DUP)*: ocupa o bit 3 e possui como função indicar a entrega duplicada quando o cliente ou servidor tentam enviar mensagens que tenham *Quality of Service (QoS)* maior que 0 e requeiram *acknowlegment (ACK)*;
- *Quality of Service (QoS)*: ocupa os bits 1 e 2 e indica o nível de garantia de entrega de uma mensagem;
- *RETAIN*: ocupa o bit 0 e possui como função a retenção da mensagem de tipo *PUBLISH* no servidor mesmo depois de ser entregue aos clientes, quando o marcador estiver ativado. No evento de uma nova subscrição a um tópico, a última mensagem retida deve ser enviada para um novo cliente, caso o marcador esteja ativo (MARTINS; ZEM, 2015).

O protocolo MQTT encontra-se atualmente na versão 5.0 e conforme apresentado pelo (IBM, 2021) possui os recursos a seguir:

- Padrão Oasis (Organization for the Advancement of Structured Information Standards - Organização para o Avanço de Padrões de Informação Estruturada);
- Aprimoramento para escalabilidade;
- Relatório de erro melhorado;
- Formulação de padrões comuns que incluem descoberta de recurso e resposta de solicitação;
- Mecanismos de extensibilidade que incluem propriedades do usuário;
- Melhorias de desempenho e suporte para clientes pequenos.

2.3 Zigbee2MQTT

O *ZigBee2MQTT* é um software de código aberto, desenvolvido por *Koen Kanters*, que provê uma interface para o usuário interagir com sistemas *ZigBee*. A função do

software é que se possa controlar múltiplos dispositivos *ZigBee* de diferentes fabricantes, através de MQTT (RUST, 2022).

2.4 VoIP

A tecnologia de Voz sobre Protocolo de Internet (VoIP) é uma abordagem inovadora que permite a transmissão de chamadas de voz por meio da Internet ou de redes IP baseadas em pacotes. Ao contrário dos sistemas tradicionais de telefonia, que usam circuitos dedicados, o VoIP utiliza pacotes de dados para enviar e receber informações de voz em tempo real.

O VoIP transformou a comunicação de voz ao proporcionar vantagens significativas, como custos mais baixos, flexibilidade, integração com outros serviços baseados na Internet e recursos avançados, como videochamadas e compartilhamento de arquivos. A tecnologia VoIP converte os sinais de áudio analógicos em dados digitais, que são comprimidos e divididos em pacotes para transmissão pela rede. No destino, os pacotes são reagrupados e convertidos novamente em sinais de áudio para reprodução (RILEY, 2005).

2.4.1 SIP

Conforme definição do *RFC3261*, o protocolo SIP *(Session Initiation Protocol)* é um protocolo de controle da camada de aplicação que pode estabelecer, modificar e encerrar sessões multimídias pela Internet (ROSENBERG et al., 2002).

O protocolo SIP atua na camada de aplicação da camada TCP/IP e oferece cinco tipos de serviços para iniciação e finalização de sessões multimídias, que são descritas abaixo, conforme apresentado por Andreis (2010, p. 16):

> "- Localização do usuário: responsável pela localização do terminal para estabelecer a conexão
> - Disponibilidade do usuário: responsável por realizar a vontade do usuário em estabelecer uma sessão de comunicação.
> - Recursos do usuário: responsável pela determinação dos meios de comunicação e os parâmetros a serem utilizados.
> - Gerência da sessão: responsável por inicializar, terminar ou colocar em espera uma sessão.
> - Modificar sessão: responsável por modificar uma sessão em andamento."

O SIP utiliza do protocolo Session Description Protocol (SDP) para estabelecer uma sessão multimídia. Este protocolo não entrega a mídia em si, mas é usado para a negociação entre os pontos finais de todas as propriedades associadas com o tipo de mídia (ANDREIS, 2010).

O transporte da mídia, por sua vez, é realizado pelo protocolo Real-time Transport Protocol (RTP), que define um formato de pacote padrão para a entrega de mídias através da Internet (ANDREIS, 2010).

Para o bom funcionamento do protocolo SIP, são definidas diversas entidades que juntas compõem a arquitetura e desempenham os seguintes papéis dentro do protocolo SIP, de acordo com Andreis (2010):

- User Agents: é a entidade que interage com o usuário, tendo a capacidade de enviar e receber requisições e dessa forma agindo tanto como cliente como servidor.
- Proxy Server: é uma entidade intermediária com a finalidade de fazer pedidos em nome de outros clientes, pode atuar tanto como servidor quanto cliente.
- Redirect Server: é um tipo de servidor SIP que ajuda a localizar entidades User Agents e fornece localizações alternativas, onde o usuário pode ser acessado.
- Register Server: é um tipo de servidor que armazena registros sobre usuários, fornecendo serviços de localização.

O protocolo SIP é baseado em texto e utiliza em sua estrutura o conjunto de caracteres *UTF-8*. Trabalha através de requisições e respostas para suas transações e suas respostas possuem um *status code*, que informa o resultado da requisição pela qual está respondendo. Conforme apresentado por Martinez e Jr (2011, p. 21), o *status code* varia de 100 a 699 e cada centena representa um tipo de resposta, tendo as seguintes classificações:

1XX - Mensagem informativa
2XX - Mensagem de sucesso
3XX - Mensagem de redirecionamento
4XX - Mensagem de erro na requisição
5XX - Mensagem de erro do servidor
6XX - mensagem de erro global

O *status code* ainda vem acompanhado de uma *reason pfrase*, que é referente à uma frase informativa ao código da mensagem. O protocolo SIP possui diversos tipos de requisições, a maior parte delas integram a parte extendida do protocolo e não se tornam importantes para o funcionamento básico. Sendo assim, as principais requisições do SIP são de 6 tipos, conforme abaixo:

INVITE - usado para convidar um usuário para uma sessão
ACK - usado para confirmar o recebimento de uma resposta final
OPTIONS - usado para perguntar ao servidor a respeito de suas funcionalidades
BYE - usado para sair de uma sessão
CANCEL - usado para cancelar transações pendentes

REGISTER - usado para que o usuário informe sua localização atual para um servidor de registro.

Em relação à segurança, o SIP utiliza diversos mecanismos adequados a diferentes aspectos e aplicações que proporcionem a autenticação de utilizadores e a privacidade dos participantes em uma sessão. Para a autenticação da identidade dos utilizadores, o protocolo SIP utiliza do método de autenticação *Digest*, utilizado pelo protocolo Hypertext Transfer Protocol (HTTP), permitindo aos utilizadores a identificação através de um nome de utilizador e de uma palavra-chave cifrada, no entanto, esse método não garante a confidencialidade e integridade das mensagens (SOUSA; CARRAPATOSO, 2003).

Embora o SIP tenha desempenhado um papel importante no desenvolvimento das aplicações de VoIP, existem outros protocolos de sinalização emergentes que também estão ganhando popularidade. Um exemplo notável é o WebRTC (Web Real-Time Communication), um conjunto de tecnologias e APIs que permite a comunicação em tempo real por meio de navegadores da web, sem a necessidade de plugins ou softwares adicionais.

O WebRTC utiliza o protocolo Session Description Protocol (SDP) para estabelecer conexões ponto a ponto e pode trabalhar em conjunto com o SIP para fornecer serviços de comunicação abrangentes. Embora o SIP ainda seja amplamente utilizado, o WebRTC tem sido adotado por muitos desenvolvedores e empresas devido à sua simplicidade de uso e capacidade de integração nativa com navegadores da web.

2.4.2 RTP

O Real-time Transport Protocol (RTP) é um protocolo utilizado no contexto de Voz sobre Protocolo de Internet (VoIP) e outras aplicações em tempo real para o transporte eficiente de áudio, vídeo e outros dados multimídia. O RTP é responsável por fornecer recursos de entrega em tempo real, sincronização e detecção de perdas durante a transmissão desses tipos de dados (SCHULZRINNE et al., 2003).

O RTP opera em conjunto com o protocolo de controle RTCP (Real-time Transport Control Protocol) para monitorar e controlar a qualidade do serviço, realizando funções como o cálculo do atraso, a perda de pacotes e a sincronização entre diferentes fluxos de mídia.

O protocolo RTP utiliza um formato de pacote padrão para encapsular os dados multimídia em pacotes que são transmitidos pela rede. Cada pacote RTP contém uma carga útil que inclui a amostra de áudio ou vídeo, bem como informações de controle, como marcação de tempo, sequência e identificação do tipo de mídia.

Os pacotes RTP são enviados em intervalos regulares, permitindo a reconstrução da mídia no destino. Além disso, o RTP possui mecanismos para lidar com perdas de pacotes, como a retransmissão seletiva e a correção de erros por meio de mecanismos de

redundância (PERKINS, 2003).

2.5 TagoIO

A TagoIO é uma plataforma web, de nuvem IoT de ponta a ponta. A finalidade da plataforma é permitir o monitoramento de ambientes via dispositivos IoT conectados à rede e é indicada para quem precisa de uma solução rápida e de fácil configuração para iniciar o sistema de monitoramento (XAVIER, 2019).

É possível a utilização em diferentes aplicações, tais como na agricultura, cadeia de mantimentos, refrigeração, automação industrial, logística, energia, entre outros. A plataforma permite que haja o gerenciamento de dispositivos, armazenamento de dados, execução de análises, integração de serviços e suporte e gerenciamento de usuários (TAGOIO, 2022).

O sistema permite que haja conexão com dispositivos MQTT, *HTTPS*, *WiFi*, *LoRaWAN*, *SIgfox*, *NBIoT*, *LTE*, *BLE*, *ZigBee* e muitos outros, além de oferecer uma vasta gama de *widgets* para construção da *dashboard* (TAGOIO, 2022).

O primeiro passo do processo de criação da solução IoT consiste na conexão com um dispositivo que realize a leitura dos ambientes e envio dos dados para a plataforma. Tendo a conexão feita, o próximo passo é a escolha do *template* para iniciar a construção da aplicação e integração das soluções, seguindo para a criação e gerenciamento dos usuários e dispositivos, escalando a aplicação de forma rápida. Por fim, tendo a configuração feita, o *dashboard* no sistema estará pronto para o monitoramento e será possível criar diversos painéis por meio dos recursos e *widgets* disponibilizados pela plataforma (XAVIER, 2019).

A plataforma também oferta para os desenvolvedores espaço para criar *scripts* que manipulam dados de qualquer dispositivo em tempo real. Então podemos fazer uma infinitividade de manipulações, para geração de relatórios e implementações de aprendizado de máquina. Para tomada de decisões é possível enviar *SMS*, *e-mails*, notificações e dados de volta para o seu dispositivo, ou até mesmo executar os *scripts* previamente desenvolvido (TAGOIO, 2022).

3 APRESENTAÇÃO DA PROPOSTA

A proposta deste projeto consiste em desenvolver um *gateway Zigbee* para processar dados de diferentes sensores RF e também embarcar uma intercomunicação SIP para chamadas de voz. Neste capítulo, será feito a descrição do sistema bem como a descrição dos componentes e sensores utilizados no mesmo.

3.1 Descrição do sistema

O sistema visa oferecer uma solução integrada para o monitoramento do idoso e de seu lar, através de sensores *ZigBee* emparelhados ao *gateway*. Através do sistema é possível receber, em tempo real, os dados dos sensores e em um *dashboard* de fácil visualização poderemos observar os eventos, como a queda do idoso (através de um colar com sensor de vibração), informações do ambiente como temperatura, umidade, sensor de porta, sensor de gás de cozinha, sensor de fumaça, botão de emergência, entre outros. Ao ser gerado um evento de emergência, imediatamente o *gateway* realizará uma chamada Voz sobre Protocolo de Internet (VoIP) utilizando a sinalização SIP para uma central de monitoramento ou outro número previamente configurado.

Além do monitoramento, o protótipo também pode ser utilizado em um serviço de teleassistência, permitindo que a central entre em contato para garantir que o idoso esteja tomando os medicamentos adequadamente e para motivar a realização de exercícios diários.

Para a configuração do protótipo foi desenvolvida uma interface de acesso à web de configuração, com usuário e senha. Um menu principal deixando a disposição submenus para certas configurações do dispositivo foi entregue. Então podemos realizar configurações de rede, configuração do ramal SIP, configurações de volume do áudio da chamada e também uma tela para pareamentos de novos sensores.

Para visualização dos dados em tempo real, foi desenvolvido um parser dentro da plataforma TagoIO e uma tela amigável para acompanhar os eventos dos sensores instalados no ambiente de operação.

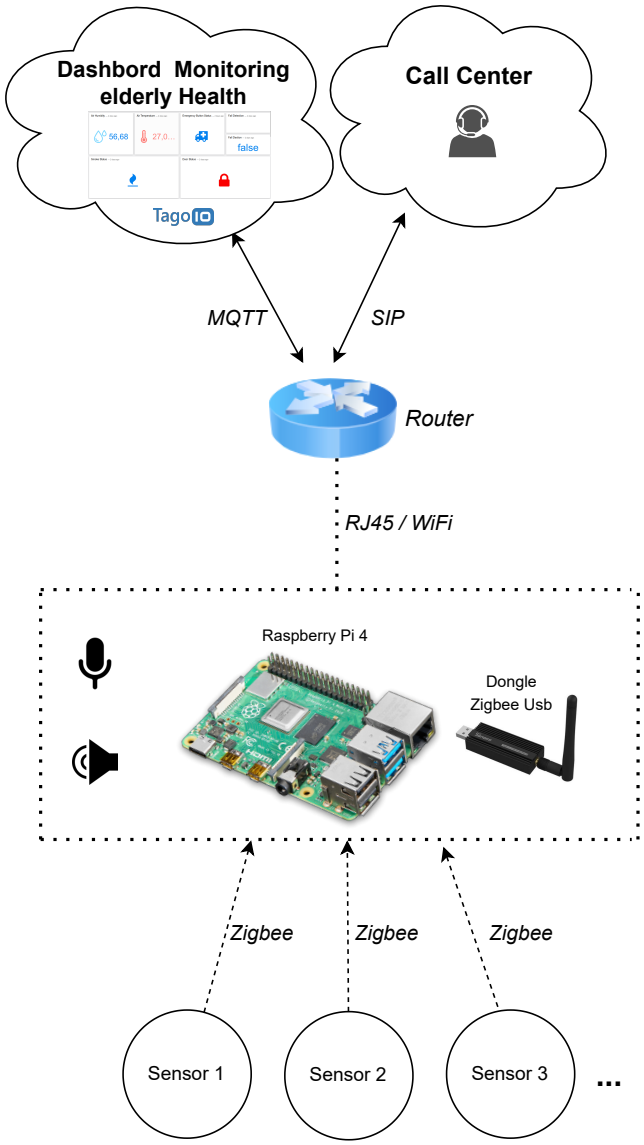
A nível de servidores, foi utilizado um serviço de nuvem para instalação de um *Asterisk* como central Private Automatic Branch Exchange over Internet Protocol (PABX IP). E para a tratativa dos dados enviados pelo *gateway* através do protocolo MQTT, foi utilizado o próprio *broker* MQTT da TagoIO.

Para salvar as configurações da interface web do protótipo e outros dados impor-

tantes, foi utilizado um banco de dados *SQLITE3* previamente instalado no *Linux* da *Raspberry Pi 4*.

A topologia do sistema pode ser visto na Figura 5.

Figura 5 – Topologia do sistema proposto



Fonte: AUTOR

3.2 Descrição dos componentes e sensores

Nesta seção será detalhada os componentes utilizados para construir o protótipo.

3.2.1 Raspberry Pi 4

Raspberry Pi é uma linha de produtos de micro computadores da *Raspberry Pi Foundation*, uma entidade de caridade do Reino Unido. Que tem o intuito de oferecer um

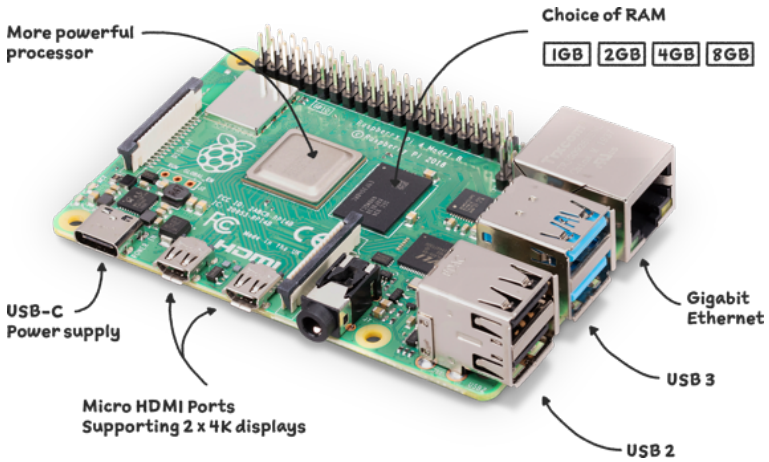
hardware de baixo custo podendo ser comparados em desempenho com computadores. (FOUNDATION, 2012).

Para nosso desenvolvimento do protótipo selecionamos o dispositivo *Raspberry Pi 4* com 4GB de memória RAM e tem como outras características técnicas:

- Rede: Porta de rede Gigabit Ethernet, WiFi, Bluetooth.
- USB: 4 portas.
- Processador: Quad core 64-bit ARM-Cortex A72.
- Clock: 1.5GHz.
- Possui um slot para SD Card.
- Alimentação: 5V.
- Vídeo: 2 portas Micro HDMI 4k.
- GPIO: 40 pinos.

A placa e seus periféricos podem ser visualizados na Figura 6:

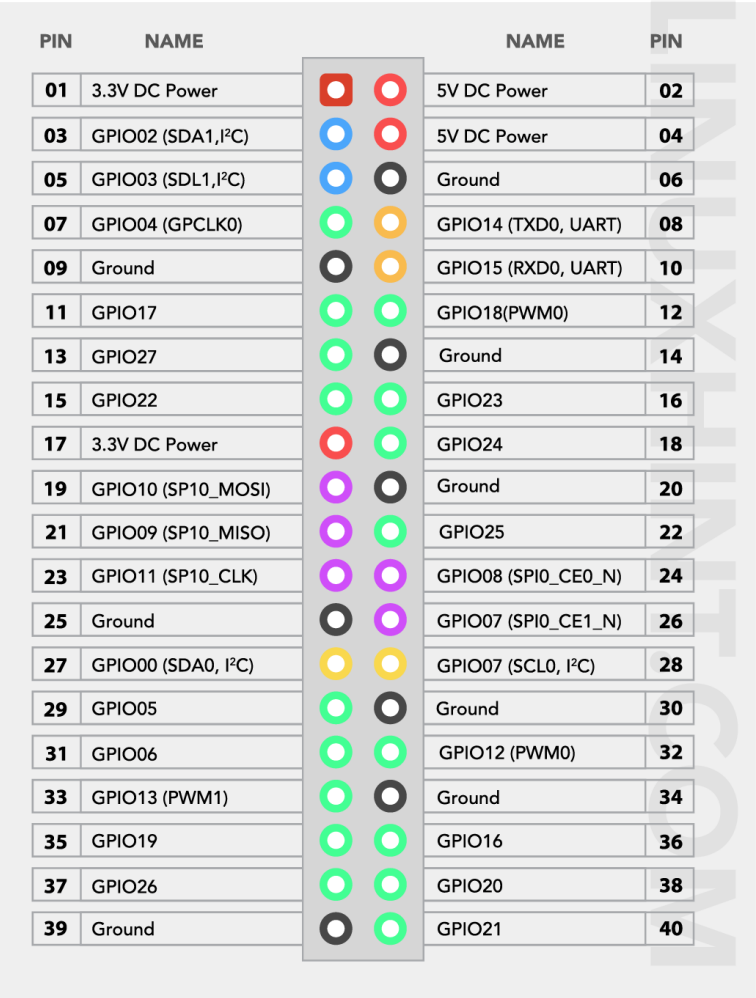
Figura 6 – Raspberry Pi 4.



Fonte: (RASPBERRY...,)

O dispositivo oferece 40 pinos, tendo pinos para alimentação, para aterramento, para comunicação e outros que podem ser configurados como entrada ou saída de dados. Os pinos General Purpose Input/Output (GPIO) entendem como alto(1) quando um nível elétrico entre 1,8V e 3V. Baixo(0) quando é menor que 1,8V. O *pinout* e suas configurações podem serem observados na Figura 7:

Figura 7 – Pinout da Raspberry Pi 4.



Fonte: (KHAN, 2022)

3.2.2 SONOFF ZigBee 3.0

SONOFF Zigbee 3.0 USB Dongle(Figura 9) é um adaptador USB Zigbee universal. Segundo a (SONOFF, 2022), ele pode ser usado como um *gateway Zigbee* universal em aplicações de *Smart Home* ou outras plataformas de código aberto via *Zigbee2MQTT* para controlar localmente todos os seus subdispositivos *Zigbee*, para que você não precise investir nos *hubs Zigbee* para diferentes marcas. Então com esse adaptador USB mais a *Raspberry Pi* conseguimos construir um *gateway Zigbee* para processar sinais de quaisquer sensores *Zigbee* de mercado.

Figura 8 – SONOFF ZigBee 3.0.

Fonte: (SONOFF, 2022)

O produto possui as seguintes características:

- CC2652P + CP2102N.
- +20dBm de ganho de saída.
- Antena externa com conector *SMA*.
- Case de alumínio.
- Frequência de operação 2,405 - 2,485 GHz.

3.2.3 Alto falante e microfone

Para a comunicação de áudio usaremos um microfone para capturar o áudio no protótipo e um alto falante com interface de áudio *P2* e alimentado através da porta USB da *Raspberry Pi 4*. Desta forma poderemos fazer a comunicação de voz com o idoso.

3.2.3.1 Alto falante

Alto falante selecionado possui saída de som P2 para fácil conexão na Raspberry Pi e tem as seguintes características:

- Fonte de alimentação: 5V USB
- Diâmetro do alto-falante: 2 polegadas
- Saída de áudio (Plugue): 3,5 mm (P2)
- Potência de saída: 2W
- Sensibilidade: -65dB
- Impedância: 40Ω

3.2.3.2 Microfone

O microfone USB utilizado é um dispositivo projetado para reprodução de som real e possui uma distância efetiva de captação de 5 metros, com alta sensibilidade. Ele apresenta recursos como cancelamento de ruído, microfone de cartão de som incorporado e blindagem automática, proporcionando um som imersivo em 360°. Segue algumas características técnicas:

- Material: ABS + Metal
- Cor: Preta
- Impedância: 2.2KΩ
- Sensibilidade: -67dBV / pBar, -47dBV / Pascal +/- 4dB
- Sensibilidade reduzida: -3dB a 1,5V
- Tensão de trabalho: 4.5V
- Resposta de frequência: 100 16kHz
- Relação sinal / ruído: mais de 67dB
- Cabo blindado de 8"com conector USB
- Tamanho do item: 2.2 * 1.8 * 0.5cm / 0.9 * 0.7 * 0.2in (L * A * P)

3.2.4 Sensores RF *ZigBee*

Para a solução proposta vamos utilizar alguns sensores *ZigBee* de mercado. Desde sensor de vibração, para processar quedas do idoso, botão/colar de emergência para o idoso iniciar um evento de emergência e outros sensores amplamente utilizado em soluções de *Smart Home* como sensor de porta, sensor de fumaça, sensor de gás de cozinha, etc.

3.2.4.1 Sensor de vibração e quedas - TuYa TS0210

O TuYa TS0210 é um sensor de vibração. Esse dispositivo pode ser instalado em um colar no pescoço do idoso para detectar quedas, por exemplo. Ele é alimentado por bateria e a única manutenção necessária é a troca da bateria, que deve ser realizada aproximadamente a cada ano, dependendo das condições de uso, como frequência de atualização e ambiente (temperatura e umidade).

- Tensão de operação: DC3V (duas pilhas alcalinas AAA)
- Rede: Zigbee

Figura 9 – TuYa TS0210.

Fonte: (TUYA,)

- Temperatura de operação: -10ºC-+50ºC
- Dimensões: 50x36,5x13mm
- Alcance: <=70 metros
- Alerta de bateria baixa

3.2.4.2 Botão de emergência - ORVIBO SE21

O sensor botão de emergência Orvibo SE21 é um dispositivo projetado para garantir a segurança e o conforto. Ele é capaz de detectar situações de emergência e enviar um sinal de alerta para a central de segurança, que pode tomar as medidas necessárias para garantir a proteção do ambiente/pessoa.

Este dispositivo tem um design discreto e elegante conforme a imagem a seguir, que se integra facilmente em qualquer ambiente. Sua principal função é servir como um botão de emergência que pode ser acionado em caso de necessidade. Quando o botão é pressionado, o sensor envia um sinal de alerta para o gateway, indicando que há uma situação de emergência em andamento.

Quanto às características técnicas, o sensor Orvibo SE21 é equipado com tecnologia sem fio *Zigbee*. Ele tem um alcance de comunicação de até 100 metros em espaço aberto e uma duração de bateria de até 1 ano, dependendo do uso. O sensor é compatível o dongle USB Zigbee da SONOFF utilizado no gateway.

Figura 10 – Botão de emergência ORVIBO SE21

Fonte: (ORVIBO,)

O dispositivo é fácil de instalar e configurar, não requerendo nenhum conhecimento técnico especializado.

Em resumo, o sensor botão de emergência Orvibo SE21 é um dispositivo de segurança completo e confiável, que pode ser usado em uma ampla gama de situações. Sua facilidade de uso, combinada com sua tecnologia avançada e características técnicas impressionantes, tornam-no uma escolha popular para aqueles que buscam garantir a segurança e o bem-estar em seus ambientes residenciais. Segue algumas características técnicas do produto:

- Tensão de operação: DC3V(1xCR2032 bateria do tipo pilha)
- Rede: Zigbee
- Temperatura de operação: -10ºC-+50ºC
- Dimensões: 57,5x34,5x13mm
- Alcance: <=70 metros
- Alerta de bateria baixa

3.2.4.3 Sensor de porta - Xiaomi MCCGQ01LM

O sensor de porta/janela MCCGQ01LM da Xiaomi é um dispositivo compacto e eficiente projetado para monitorar a abertura e o fechamento de portas e janelas. Com um design discreto e uma instalação simples, esse sensor oferece uma maneira conveniente de melhorar a segurança da sua casa ou escritório.

O sensor MCCGQ01LM utiliza tecnologia sem fio Zigbee para se comunicar com outros dispositivos inteligentes da Xiaomi, permitindo que você monitore o status das portas e janelas. Ao detectar a abertura de uma porta ou janela, o sensor envia um sinal para o gateway Zigbee, que por sua vez pode acionar alarmes, notificações ou executar outras ações programadas.

Figura 11 – Sensor de porta MCCGQ01LM

Fonte: (ZIGBEE2MQTT, 2022)

- Marca: Xiaomi
- Modelo: MCCGQ01LM
- Cor: Branco
- Rede: ZigBee
- Alcance: <=70 metros
- Comprimento: 4,08 cm
- Alimentação: 1 pilha CR2032 (3V)
- Largura: 2,08 cm
- Altura: 1,1 cm
- Peso: 13 gramas

3.2.4.4 Sensor de temperatura e umidade - ORVIBO ST20

O sensor ZigBee de temperatura e umidade ORVIBO ST20 é um dispositivo inteligente projetado para monitorar e controlar as condições climáticas em ambientes internos. Com sua capacidade de se conectar a uma rede ZigBee, ele oferece uma solução conveniente para medir e acompanhar com precisão a temperatura e umidade em sua casa, escritório ou qualquer outro espaço.

O ORVIBO ST20 possui um design compacto e discreto, permitindo que seja facilmente colocado em qualquer ambiente. Ele utiliza tecnologia avançada para fornecer leituras precisas e confiáveis de temperatura e umidade, ajudando você a criar um ambiente confortável e saudável.

Figura 12 – Sensor de temperatura e umidade - ORVIBO ST20

Fonte: (ORVIBO,)

- Tensão de operação: DC 3V (1 bateria CR2450)
- Corrente em standby: $< 10\mu$A
- Rede: Rede Zigbee ad hoc
- Distância de rede sem fio: $<$ 100m (em área aberta)
- Temperatura de operação: -15°C a +60°C
- Umidade ambiente: Máximo de 95
- Dimensões externas: 60mm x 60mm x 20.8mm

3.2.4.5 Sensor de fumaça - M415-6C

O sensor de fumaça M415-6C é um dispositivo de segurança produzido pela empresa BYUN e projetado para detectar a presença de fumaça em ambientes internos. Este dispositivo usa a tecnologia de detecção fotoelétrica, que detecta a fumaça através da análise de partículas suspensas no ar que podem ser geradas por um incêndio.

O M415-6C é compatível com o protocolo de comunicação Zigbee e pode ser integrado a um sistema de automação residencial ou empresarial. Isso permite que o sensor de fumaça se comunique com outros dispositivos Zigbee, como luzes, alarmes ou sistemas de segurança, permitindo uma resposta rápida em caso de incêndio.

O sensor de fumaça M415-6C tem uma ampla faixa de detecção de fumaça, com um alcance de até 20 metros em uma área aberta. Ele também possui um indicador de bateria fraca que alerta o usuário quando é necessário trocar a bateria. O sensor tem um design compacto e discreto, permitindo que seja instalado em qualquer lugar sem afetar a estética do ambiente.

Este dispositivo tem um alto grau de confiabilidade e precisão na detecção de fumaça, além de ser fácil de instalar e operar.

Figura 13 – Sensor de fumaça M415-6C

Fonte: (ZIGBEE2MQTT, 2022)

Segue algumas características técnicas do produto na tabela a seguir:

- Tensão de operação: DC3V(CR123a/CR17335)
- Rede: Zigbee
- Potência sonora: 120dB/1m
- Temperatura de operação: -10ºC-+50ºC

- Dimensões: 57,5x34,5x13mm
- Alcance: <=70 metros
- Alerta de bateria baixa

3.2.4.6 Sensor de gás - Xiaomi JTQJ-BF-01LM/BW

O sensor de gás Xiaomi JTQJ-BF-01LM/BW é um dispositivo projetado para detectar a presença de gases perigosos no ambiente. Ele utiliza tecnologia avançada para monitorar e alertar sobre vazamentos de gases potencialmente tóxicos, oferecendo maior segurança em residências, escritórios ou outros espaços.

O sensor de gás Xiaomi possui um design compacto e discreto, permitindo que seja facilmente instalado em diferentes locais. Ele é capaz de detectar gases como monóxido de carbono (CO), que é inodoro e altamente tóxico, além de outros gases combustíveis que podem representar riscos para a saúde e segurança das pessoas.

Com uma interface simples e intuitiva, o sensor de gás Xiaomi fornece alertas visuais e sonoros em caso de detecção de gás. Além disso, ele se integra facilmente a outros dispositivos inteligentes por meio da tecnologia de conexão sem fio Zigbee, permitindo o controle remoto e a automação de ações em resposta à detecção de gás.

Figura 14 – Sensor de gás Xiaomi JTQJ-BF-01LM/BW

Fonte: (ZIGBEE2MQTT, 2022)

- Marca: Xiaomi Gas Alarm Honeywell
- Modelo: JTQJ-BF-01LM/BW
- Material: Plástico

- Rede: Wifi/Zigbee
- Alarme: 80dB
- Cor: Branco
- Peso do produto: 0.0900 kg
- Tamanho do produto (C x L x A): 8.00 x 2.90 x 2.00 cm / 3.15 x 1.14 x 0.79 polegadas

3.2.5 Cliente SIP

Para comunicação de voz com o idoso utilizaremos o *softphone Linphone*. O mesmo oferece um *daemon* capaz de ser executado a partir de comandos do *Bash Linux*. Linphone-daemon é um programa de modo de console baseado em Liblinphone que pode registrar, enviar e receber chamadas lendo comandos de texto simples de entrada padrão ou de um soquete UNIX. Em resposta a cada comando, o linphone-daemon grava o status de execução do comando na saída padrão (ou em um soquete). Esta ferramenta pode ser facilmente usada a partir de um script de shell para executar cenários SIP simples por aqueles que não requerem todos os recursos do Liblinphone SDK (SARL, 2020).

3.2.6 Interface web de configuração

Um servidor web será desenvolvido para configuração do gateway desenvolvido, isto nos proporcionará a possibilidade de alterar configurações de rede, SIP e também para adição de novos sensores no sistema. Uma aplicação em *Python* utilizando *Flask* para criar as rotas da Application Programming Interface (API) e o uso da linguagem Linguagem de Marcação de HiperTexto (HTML) e Cascading Style Sheets (CSS) para criação do *frontend*.

4 DESENVOLVIMENTO DO SISTEMA

Nesta seção, apresentaremos as etapas realizadas para o desenvolvimento da proposta, incluindo o desenvolvimento do gateway, da interface, a integração do sistema, bem como a realização dos testes e levantamento de custos do protótipo.

4.1 Desenvolvimento do gateway

4.1.1 Instalação do sistema operacional

Para primeira parte do desenvolvimento, utilizamos o hardware *Raspberry Pi 4* para a instalação do sistema operacional *Linux*. Optamos por utilizar um Raspbian, que seria um Debian projetado para Raspberry Pi. A versão instalada foi instalado no cartão SD conectado a ao dispositivo com a versão:

Linux raspberrypi 5.10.103-armv7l GNU/Linux

4.1.2 Instalação do softphone Linphone

Via terminal instalamos o *Linhone* comentado na seção subseção 3.2.5. A função principal deste software será a intercomunicação VoIP com uma central PABX IP. Com o uso de um microfone e um alto falante na interface de áudio da placa *Rasberry Pi 4* será possível realizar e receber chamadas pela internet.

Para instalação foi necessário adicionar um repositório para obter o código, os passos para instalação estão detalhadas no Código 4.1 apresentado a seguir:

Código 4.1 – Script para instalação do Linphone

```
1 sudo add-apt-repository ppa:linphone/release
2 sudo apt-get update
3 sudo apt-get upgrade
4 sudo apt-get install linphone
```

Após a instalação, utilizaremos o *daemon* para console *Linphonec* a fim de controlar a sinalização de telefonia SIP. As configurações do *softphone* são realizadas através do arquivo *".linphonerc"*. A seguir, apresentamos um exemplo básico de configuração do *Opensource SIP*, incluindo seleção das interfaces de áudio, volume, configurações do ramal SIP, codecs de áudio, entre outros.

Código 4.2 – Script para instalação do Linphone

```
[sound]
playback_dev_id=ALSA Unknown: bcm2835 Headphones
ringer_dev_id=ALSA Unknown: bcm2835 Headphones
capture_dev_id=ALSA Unknown: USB PnP Sound Device
media_dev_id=ALSA Unknown: bcm2835 Headphones
playback_gain_db=12.000000
mic_gain_db=6.000000
echocancellation=1
remote_ring=/home/pi/linphone-sdk/build-raspberry/linphone-sdk/desktop/share/sounds
    /linphone/ringback.wav

[net]
nat_policy_ref=~SxRM6wMhArnGeU
stun_server=stun.linphone.org

[sip]
root_ca=/home/pi/linphone-sdk/build-raspberry/linphone-sdk/desktop/share/linphone/
    rootca.pem
verify_server_certs=1
verify_server_cn=1
contact=6000@192.168.110.80
media_encryption=none
default_proxy=0

[rtp]
audio_rtp_port=7078
video_rtp_port=9078
text_rtp_port=11078
audio_jitt_comp=60
video_jitt_comp=60
nortp_timeout=30

[audio_codec_0]
mime=opus
rate=48000
channels=2
enabled=1
recv_fmtp=useinbandfec=1

[audio_codec_1]
mime=speex
rate=16000
channels=1
enabled=1
recv_fmtp=vbr=on

[audio_codec_2]
```

```
mime=speex
rate=8000
channels=1
enabled=1
recv_fmtp=vbr=on

[audio_codec_3]
mime=PCMU
rate=8000
channels=1
enabled=1

[audio_codec_4]
mime=PCMA
rate=8000
channels=1
enabled=1

[auth_info_0]
username=6000
ha1=2bbf820b3d0816478c3f9ba102ff0585
realm=192.168.110.80
domain=192.168.110.80
algorithm=MD5
available_algorithms=MD5

[proxy_0]
reg_proxy=sip:192.168.110.80
reg_identity=sip:6000@192.168.110.80
quality_reporting_enabled=0
quality_reporting_interval=0
reg_expires=600
reg_sendregister=1
publish=1
avpf=-1
avpf_rr_interval=5
dial_escape_plus=0
privacy=32768
push_notification_allowed=0
idkey=proxy_config_hL4WuVarpviqEQh
publish_expires=-1
```

No contexto do arquivo de configuração mencionado, a seção [*sound*] é onde são definidas as configurações dos periféricos de áudio utilizados no protótipo, incluindo o ativamento do cancelador de eco.

O cancelador de eco desempenha um papel importante no produto, pois ajuda a

evitar a ocorrência de eco durante as chamadas. Isso é especialmente relevante devido à proximidade física do microfone e do alto-falante no design mecânico do dispositivo. O cancelador de eco é responsável por reduzir ou eliminar os reflexos sonoros indesejados, garantindo uma melhor qualidade de áudio durante as conversas.

Essa função é útil para proporcionar uma experiência de chamada mais clara e livre de interferências, melhorando a comunicação entre o idoso e a central de monitoramento.

As configurações SIP do *softphone* embarcado na placa, é realizado nos contextos *[sip]*, *[auth_info_0]* e *[proxy_0]*. Como é possível visualiza no código acima, foi configurado um ramal "6000"no endereço IP do PABX "192.168.110.80". O arquivo completo *".linphonerc"* encontra-se em apêndices.

O próximo passo consiste em configurar o gateway para iniciar a aplicação do *Linphone* automaticamente durante o boot do sistema operacional. Essa configuração é importante para garantir que a funcionalidade de telefonia esteja prontamente disponível e em execução sempre que o sistema for iniciado.

Ao adicionar a inicialização do *Linphone* no boot do sistema, você garante que a aplicação seja carregada e esteja pronta para receber chamadas ou realizar chamadas de forma contínua e sem intervenção manual. Isso é especialmente relevante para o caso do monitoramento de idosos, onde é crucial manter a comunicação ativa e contínua com a central de monitoramento. No código 4.3 é possível observar como foi realizado a configuração da inicialização e a criação de um arquivo de log para debug.

Código 4.3 – Configurar Linphone no boot do Sistema Operacional

```
1 export PATH=/home/pi/linphone-sdk/build-raspberry/linphone-sdk/desktop/bin:$PATH
2 sudo -u pi linphonecsh init -a -C -c /home/pi/.linphonerc -d 6 -l /tmp/log.txt
```

4.1.3 Monitoração da GPIO - botoeira

Para a geração de chamadas, o protótipo é equipado com um botão que permite a realização de chamadas para um número pré-configurado na interface web. Essa funcionalidade é especialmente útil em situações de emergência, permitindo ao idoso acionar rapidamente a central de monitoramento.

Para monitorar o estado dos botões presentes no protótipo, utilizamos a biblioteca *WiringPi* do *Python*. Essa biblioteca fornece uma interface para controlar os pinos da *Raspberry Pi*. No caso específico, o pino selecionado é o 18, conforme script em anexos.

Ao pressionar o botão, o protótipo utiliza o *Linphone* para originar uma chamada *SIP* para o ramal 6002 do *PABX IP*. Essa ação é realizada de forma automatizada, garantindo uma resposta rápida em situações de emergência.

A integração entre o hardware (*Raspberry Pi*) e o software (*Linphone* e *WiringPi*) permite que o protótipo ofereça uma solução eficiente e confiável para acionar chamadas de emergência. A utilização da biblioteca *WiringPi* simplifica o monitoramento dos botões e a interação com a *Raspberry Pi*.

Dessa forma, o protótipo proporciona um meio seguro e prático para o usuário entrar em contato com a central de monitoramento, garantindo assistência imediata em caso de necessidade.

4.1.4 Instalação Zigbee2MQTT

Para o tratamento dos dados recebidos pelo receptor *Zigbee USB* e seu envio para a nuvem utilizando o protocolo MQTT, utilizamos o aplicativo *Zigbee2MQTT*. Esse aplicativo estabelece a conexão entre as redes *Zigbee* e o envio dos dados via o protocolo MQTT, permitindo o transporte dos dados de forma eficiente. Além disso, o *Zigbee2MQTT* já possui integração completa com o *Dongle SONOFF ZigBee 3.0*, facilitando a configuração e o uso.

Para instalar e configurar o *Zigbee2MQTT*, é necessário primeiramente instalar um Broker MQTT no gateway, responsável por enviar os dados via MQTT pela internet. Recomenda-se o uso do *Broker Mosquitto*, que é amplamente suportado e confiável. A instalação do *Mosquitto* pode ser feita seguindo os seguintes passos:

Código 4.4 – Instalação do Broker Mosquitto. (SANTOS, 2022)

```
1 sudo apt update && sudo apt upgrade
2 sudo apt install -y mosquitto mosquitto-clients
3 sudo systemctl enable mosquitto.service
4 mosquitto -v
```

Com os passos acima, o *Mosquitto* foi instalado para ser utilizado em conjunto com o *Zigbee2MQTT*. Essa combinação permite a integração eficiente dos dados provenientes do *Zigbee USB* com a nuvem, facilitando o acesso e a análise dos dados coletados.

No próximo passo devemos configurar o *Broker Mosquitto* para permitir que tenha uma conexão externa, também ativamos autenticação para expandir a segurança do gateway. Primeiramente executamos o seguinte comando para criar um usuário:

Código 4.5 – Comando para criar usuário. (SANTOS, 2022)

```
1 sudo mosquitto_passwd -c /etc/mosquitto/passwd pi
```

Após a execução desses comandos, o usuário será criado e a senha será armazenada em um arquivo chamado *"passwd"* localizado em *"/etc/mosquitto"*. Esse arquivo é utilizado pelo *Mosquitto* para autenticar os usuários durante as conexões.

Lembre-se de configurar corretamente as permissões de acesso ao arquivo *"passwd"* para garantir a segurança das informações. Você pode utilizar o comando *chown* abaixo para definir as permissões corretas:

Código 4.6 – Comando para criar usuário. (SANTOS, 2022)

```
1 sudo chown mosquitto:mosquitto /etc/mosquitto/passwd
```

Com a autenticação de usuário configurada, apenas usuários autorizados poderão se conectar ao *Broker Mosquitto*, aumentando a segurança do sistema.

Em seguida deve-se fazer algumas configurações no arquivo de configuração */etc/-mosquitto/mosquitto.conf*, conforme exemplo do código 3.8:

Código 4.7 – Configurações a serem feitas no arquivo de configuração do *Mosquitto*. (SANTOS, 2022)

```
1  per_listener_settings true
2
3  pid_file /run/mosquitto/mosquitto.pid
4
5  persistence true
6  persistence_location /var/lib/mosquitto/
7
8  log_dest file /var/log/mosquitto/mosquitto.log
9
10 include_dir /etc/mosquitto/conf.d
11 allow_anonymous false
12 listener 1883
13 password_file /etc/mosquitto/passwd
```

Após realizar as modificações no arquivo de configuração, salve as alterações e reinicie o serviço do *Mosquitto* para que as configurações sejam aplicadas. O comando para reiniciar o serviço pode variar dependendo do sistema operacional utilizado. Para o Linux podemos utilizar o comando abaixo:

Código 4.8 – Comando para aplicar e iniciar as novas configurações do *Mosquitto*. (SANTOS, 2022)

```
1 sudo systemctl restart mosquitto
```

Na sequencia da instalação do *Mosquitto* devemos instalar e configurar o *Zigbee2MQTT*, para isso basta seguir executar os comandos listados a seguir:

Código 4.9 – Script para instalação do Zigbee2MQTT. (ZIGBEE2MQTT, 2022)

```
1 # Set up Node.js repository and install Node.js + required dependencies
```

```
# NOTE: Older i386 hardware can work with [unofficial-builds.nodejs.org](https://
    unofficial-builds.nodejs.org/download/release/v16.15.0/ e.g. Version 16.15.0
    should work.
sudo curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash -
sudo apt-get install -y nodejs git make g++ gcc

# Verify that the correct nodejs and npm (automatically installed with nodejs)
# version has been installed
node --version  # Should output v14.X, V16.x, V17.x or V18.X
npm --version  # Should output 6.X, 7.X or 8.X

# Create a directory for zigbee2mqtt and set your user as owner of it
sudo mkdir /opt/zigbee2mqtt
sudo chown -R ${USER}: /opt/zigbee2mqtt

# Clone Zigbee2MQTT repository
git clone --depth 1 https://github.com/Koenkk/zigbee2mqtt.git /opt/zigbee2mqtt

# Install dependencies (as user "pi")
cd /opt/zigbee2mqtt
npm ci
```

Após a instalação, é necessário editar o arquivo de configuração *configuration.yaml*, que contém as informações de configuração utilizadas pelo *Zigbee2MQTT*. Para abrir o arquivo, siga os passos a seguir:

Código 4.10 – Caminho do arquivo de configuração do *Zigbee2MQTT*. (ZIGBEE2MQTT, 2022)

```
nano /opt/zigbee2mqtt/data/configuration.yaml
```

As configurações abaixo foram aplicadas para que seja enviado para a conta previamente criada na plataforma da TagoIO:

Código 4.11 – Arquivo Configuration.yaml

```
homeassistant: true
permit_join: true
mqtt:
  base_topic: SISTEMAMONITORAMENTOIDOSOS
  server: mqtt://mqtt.tago.io:1883
  user: andrey
  password: dbd6ad7e-9e15-4d91-9d8f-d5720e568ded
  keepalive: 60
  reject_unauthorized: true
  version: 4
serial:
  port: /dev/ttyUSB0
```

```
frontend:
  port: 8070
advanced:
  network_keys: GENERATE
  homeassistant_legacy_entity_attributes: false
  legacy_api: false
  legacy_availability_payload: false
  log_level: debug
  report: true
device_options:
  legacy: false
```

Algumas informações do arquivo de configuração acima devem ser ressaltada, basicamente o endereço e credenciais de acesso ao broker MQTT, o tópico que o gateway irá publicar as mensagens e a porta serial que o Dongle USB Zigbee está conectado na Raspberry Pi.

Após ter realizado todas as configurações necessárias, você pode executar a aplicação *Zigbee2MQTT* utilizando os seguintes comandos:

Código 4.12 – npm start Zigbee2MQTT

```
cd /opt/zigbee2mqtt
npm start
```

A partir desse momento, o *Zigbee2MQTT* estará em execução e pronto para se comunicar com os dispositivos *Zigbee* e enviar os dados para o *Broker MQTT* configurado anteriormente.

Lembre-se de deixar o terminal aberto enquanto a aplicação estiver em execução para monitorar possíveis mensagens de erro ou informações importantes.

Para colocar a aplicação *zigbee2MQTT* na inicialização do sistema operacional da *Raspberry Pi* vamos utilizar o *systemctl*. Criar um arquivo conforme comando abaixo:

Código 4.13 – Arquivo systemctl para inicialização do Zigbee2MQTT na inicialização do *Linux*. (ZIGBEE2MQTT, 2022)

```
# Create a systemctl configuration file for Zigbee2MQTT
sudo nano /etc/systemd/system/zigbee2mqtt.service
```

Então deve-se adicionar as seguintes informações no arquivo de configuração apropriado:

Código 4.14 – Configuração do arquivo "zigbee2mqtt.service"para inicialização do Zigbee2MQTT na inicialização do *Linux*. (ZIGBEE2MQTT, 2022)

```
[Unit]
Description=zigbee2mqtt
After=network.target

[Service]
Environment=NODE_ENV=production
ExecStart=/usr/bin/npm start
WorkingDirectory=/opt/zigbee2mqtt
StandardOutput=inherit
# Or use StandardOutput=null if you don't want Zigbee2MQTT messages filling syslog,
    for more options see systemd.exec(5)
StandardError=inherit
Restart=always
RestartSec=10s
User=pi

[Install]
WantedBy=multi-user.target
```

Agora para habilitar as configurações basta executar o comando abaixo:

Código 4.15 – Comando para habilitar as configurações e a aplicação *Zigbee2MQTT*. (ZIGBEE2MQTT, 2022)

```
sudo systemctl enable zigbee2mqtt.service
```

Com os passos descritos anteriormente, o *gateway* foi configurado para coletar os sinais dos sensores *Zigbee* utilizando o *Dongle USB* e enviar esses dados através do protocolo MQTT para a plataforma TagoIO. Essa integração permite que os dados coletados sejam armazenados e visualizados de forma eficiente na plataforma, facilitando o monitoramento e análise dos dados.

Ao enviar os dados para a plataforma TagoIO, você poderá aproveitar os recursos disponíveis, como a criação de *dashboards* personalizados, a configuração de notificações e alertas, além da possibilidade de integrar esses dados com outros serviços e aplicativos.

Essa integração entre o *gateway Zigbee* e a plataforma TagoIO oferece uma solução completa para coleta, armazenamento e visualização de dados provenientes dos sensores *Zigbee*, proporcionando uma gestão mais eficiente e assertiva das informações.

4.2 Desenvolvimento da interface

4.2.1 Interface de configuração *do* protótipo

A interface de configuração do protótipo foi desenvolvida utilizando as tecnologias *Python* para o *backend* e HTML para o *frontend*. Através dessa interface, os usuários têm acesso a diversas telas de configuração que permitem ajustar diferentes aspectos do protótipo.

As telas de configuração abrangem opções como alteração das configurações SIP, configurações de rede, alteração de senha do administrador web, além de uma tela dedicada para abrir e fechar a rede de pareamento *Zigbee* e reiniciar o dispositivo.

A aplicação *frontend* consiste em seis arquivos HTML, responsáveis pela criação das páginas, e um arquivo CSS para a estilização. Além disso, os dados de configuração do protótipo são armazenados em um banco de dados utilizando o *SQLite3*.

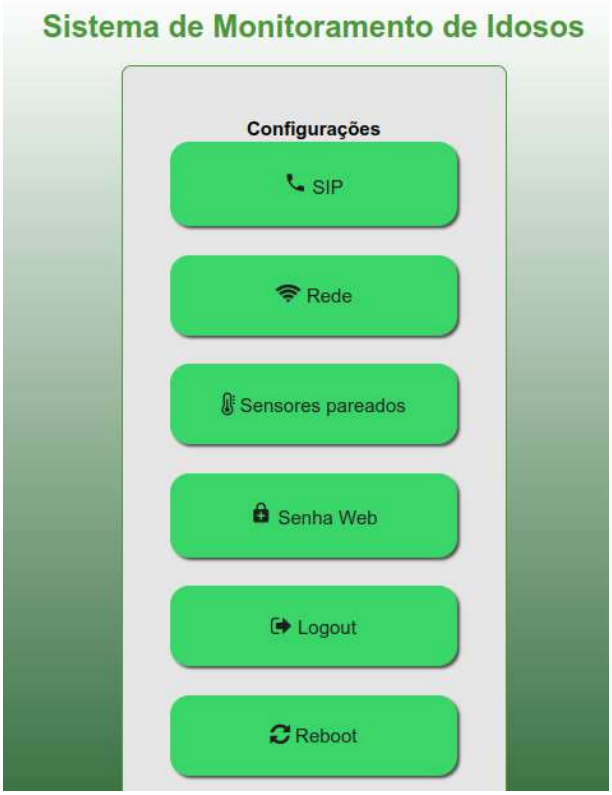
O banco de dados *SQLite3* é uma opção leve e eficiente para armazenar os dados de configuração do protótipo. Ele oferece recursos de gerenciamento de banco de dados relacionais, permitindo o armazenamento e recuperação dos dados de forma organizada e segura.

Essa interface de configuração do protótipo, juntamente com o banco de dados *SQLite3*, proporciona aos usuários uma experiência completa e flexível para personalizar as configurações do dispositivo de acordo com suas necessidades no ambiente de instalação.

4.2.1.1 Menu principal

O menu principal é representado pelo arquivo *menu.html* e serve como a página inicial onde os clientes acessarão todas as opções de configurações e visualização do sistema. Nele, estão disponíveis diversos links que permitem navegar para diferentes seções de configurações, a seguir pode ser visualizado a interface desenvolvida pelo autor:

Figura 15 – Menu principal da aplicação web.



Fonte: AUTOR

4.2.1.2 SIP

O submenu SIP foi projetado para permitir a configuração da comunicação VoIP, ou seja, o registro do protótipo no servidor *PABX IP*. Nele, são disponibilizadas opções como conta, senha, servidor e ramal de destino.

Figura 16 – Menu de configuração do VoIP.

Fonte: AUTOR

A opção "conta"refere-se às informações de autenticação do protótipo no servidor *PABX IP*, como o nome de usuário ou ID da conta. Já a opção "senha"permite definir a senha correspondente à conta configurada.

A configuração do servidor se refere ao endereço IP ou nome de domínio do servidor *PABX IP* ao qual o protótipo irá se conectar para estabelecer a comunicação.

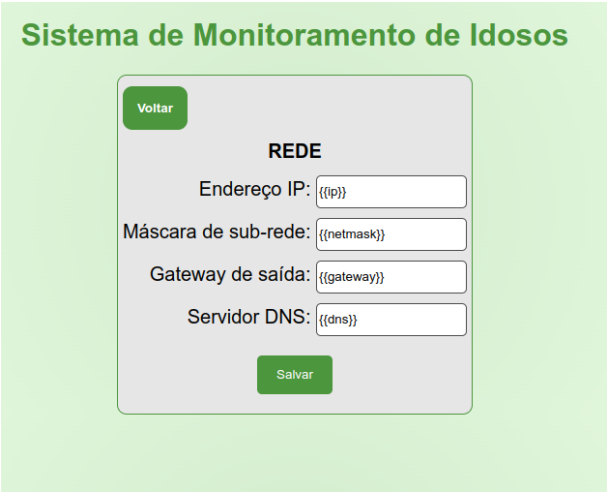
Por fim, a opção "ramal de destino"define para qual ramal o protótipo irá realizar chamadas ao receber um evento de emergência ou quando o botão físico acoplado na carcaça do dispositivo for pressionado.

Essas configurações no submenu SIP são essenciais para garantir que o protótipo esteja corretamente registrado no servidor *PABX IP* e possa realizar chamadas para o ramal de destino designado, tanto em situações de emergência quanto em casos específicos em que seja necessário estabelecer uma comunicação rápida e direta.

4.2.1.3 Rede

O submenu "Rede"(*network.html*) oferece uma interface intuitiva para a configuração das opções de rede do nosso protótipo IP. Aqui, você pode facilmente personalizar as configurações essenciais, como o endereço IP, a máscara de sub-rede, o *gateway* de saída e o servidor *DNS*. Com esse menu, é possível ajustar as configurações de rede de forma rápida e fácil, permitindo a integração perfeita do protótipo em qualquer ambiente de rede.

Figura 17 – Menu de configuração de rede.



Fonte: AUTOR

Esse submenu é especialmente útil quando é necessário adaptar o protótipo a diferentes redes ou quando ocorrem mudanças na infraestrutura de rede existente. Com as opções disponíveis, é possível configurar o protótipo para se comunicar de forma

eficiente e segura com outros dispositivos na rede, garantindo a conectividade adequada e o funcionamento correto do sistema.

4.2.1.4 Sensores Pareados

A página "Sensores Pareados"permite visualizar os sensores que estão emparelhados com o *gateway Zigbee*. Além disso, essa página oferece a funcionalidade de abrir e fechar a rede *Zigbee*.

Figura 18 – Menu para cadastrar sensores.

Fonte: AUTOR

Ao selecionar a opção "Abrir rede *Zigbee*", o protótipo permitirá novas conexões de sensores à rede *Zigbee*. Isso possibilita a inclusão de novos dispositivos no sistema, expandindo as capacidades de monitoramento e controle.

Por outro lado, ao escolher a opção "Fechar rede *Zigbee*", o protótipo interromperá a aceitação de novas conexões de sensores. Isso pode ser útil em situações em que não se deseja adicionar mais dispositivos à rede *Zigbee*, por exemplo, quando a configuração do sistema já está completa ou quando se busca limitar o acesso a sensores específicos.

Através dessa página, é possível ter controle sobre os dispositivos *Zigbee* emparelhados e gerenciar a expansão da rede conforme necessário. Isso garante flexibilidade e adaptabilidade do sistema às necessidades específicas do usuário.

4.2.1.5 Senha Web

A página "Senha Web"(*password.html*) oferece ao usuário a possibilidade de alterar sua senha de acesso à interface web do protótipo. Isso proporciona maior segurança e controle sobre as informações e configurações do sistema.

4.2.1.6 Logout

A página *"Logout"* permite ao usuário encerrar a sessão, encerrando o acesso à interface web e redirecionando-o para a página de login. Essa opção é útil quando o usuário deseja sair do sistema ou quando é necessário alternar entre diferentes contas de usuário.

4.2.1.7 Reboot

A página *"Reboot"* permite reiniciar o protótipo. Embora não haja uma página específica para essa função, a opção de reinicialização pode ser encontrada no menu principal. Essa funcionalidade é útil para reiniciar o protótipo quando necessário, seja para aplicar alterações de configuração ou solucionar problemas.

4.2.1.8 Login

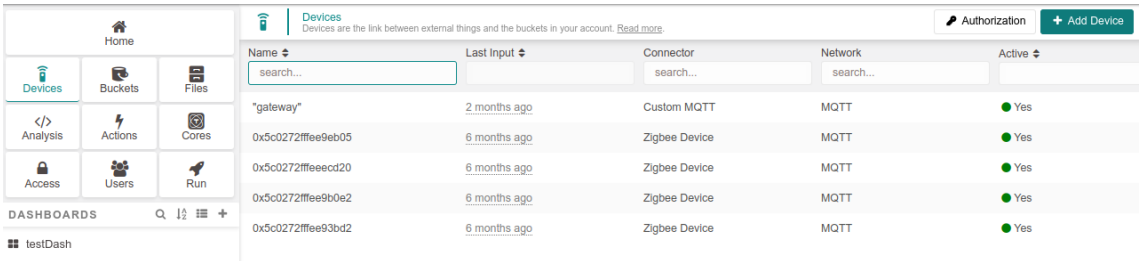
A página *"Login"* (*login.html*) é utilizada para que o usuário possa acessar sua conta no sistema, inserindo um nome de usuário e senha correspondentes registrados no banco de dados. O login é o ponto de entrada para acessar todas as funcionalidades e configurações disponíveis na interface web do protótipo.

4.2.1.9 Interface de monitoração - Dashbord

Para a visualização dos dados coletados pelos sensores em tempo real no protótipo desenvolvido, um *dashboard* foi criado na plataforma TagoIO. A mesma irá processar os dados recebidos pelo gateway e mostrar numa interface amigável os eventos dos sensores instalado na residência do idoso. Vamos usar a linguagem *Java Script* para fazer o tratamento dos dado dos sensores na plataforma.

Primeiramente deve-se fazer uma conta na plataforma, utilizamos uma versão gratuita experimental para o desenvolvimento do *dashboard*. Já conectado na plataforma, deve-se criar um *dashboard* clicando na opção "Add Dashboard", em seguida selecionando o tipo de *dashboard* como "Normal", desta forma pode-se adicionar *Widget* para cada dispositivo. Basta clicar no botão "Devices"no menu lateral esquerdo, em sequida em "+ Add Device", conforme está exposto na imagem abaixo:

Figura 19 – Adicionando um dispositivo



Fonte: AUTOR

Selecione o tipo de rede como "Custom MQTT"e de o nome ao dispositivo adicionado. Então vá no menu "Payload Parser"e desenvolva um script para transformar os dados recebidos de uma mensagem MQTT criando uma nova matriz de objetos com um formato específico que pode ser usado por outras partes do aplicativo.

Código 4.16 – Script que separa as mensagens dos dispositivos por MAC

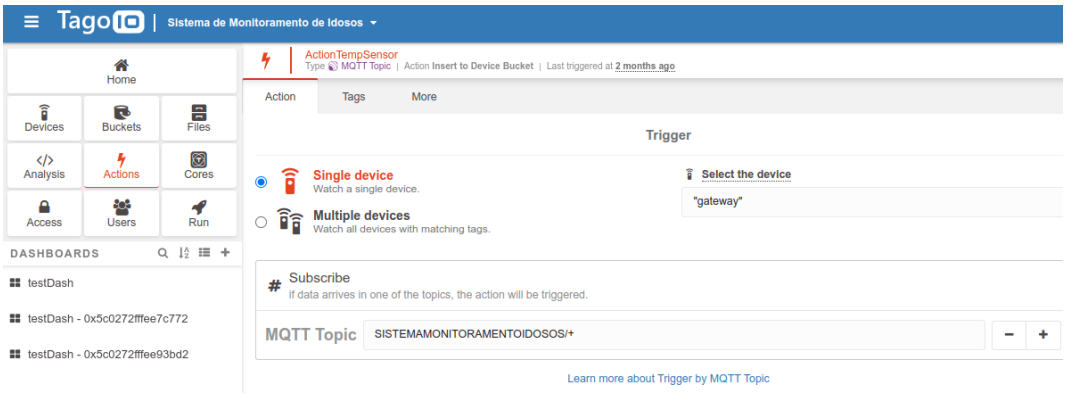
```
function transformPayload(payload) {
  const resultPayload = [];
  const topic = payload[0].metadata.mqtt_topic;
  const mac = topic.replace("TOPIC/", "");

  for (const measure in payload[0]) {
    if (measure === "metadata") {
      continue;
    }

    const variable = `${mac}_${measure}`;
    const value = String(payload[0][measure]);

    resultPayload.push({ variable, value });
  }

  return resultPayload;
}
```

Para a plataforma TagoIO receber os dados enviado pela *Raspberry Pi*, pode-se criar um *script* de análise de dados configurando o *Access Token* na variável ambiente dessa *"Analysis"*, essa funcionalidade é um tipo de serviço que permite processar dados, executar lógica personalizada e tomar ações com base nos dados recebidos. É uma forma de programação em nuvem, onde você pode criar scripts ou códigos personalizados para manipular e analisar os dados em tempo real. Esse *Access Token* gerado na plataforma é o mesmo configurado no arquivo de configuração do *Zigbee2MQTT*. Basicamente isto provê acesso a sua conta TagoIO. Foi desenvolvido um script que é responsável por ler os dados recebidos (parâmetro scope) e enviá-los para um ou mais dispositivos TagoIO específicos com base nas informações contidas em seus metadados.

Para adicionar uma "Action", configure o tópico MQTT como "SISTEMAMONITORAMENTODEIDOSOS/+, conforme apresentado na Figura 20. Com isso já conseguimos receber na TagoIO todos os dados publicados no tópico.

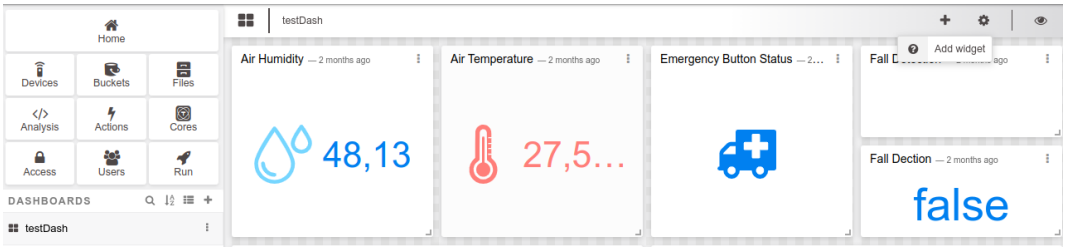
Após isso já podemos iniciar a customização da aparência do *dashboard*, crie um *widget* conforme a Figura 21:

Figura 20 – Adicionando uma action



Fonte: AUTOR

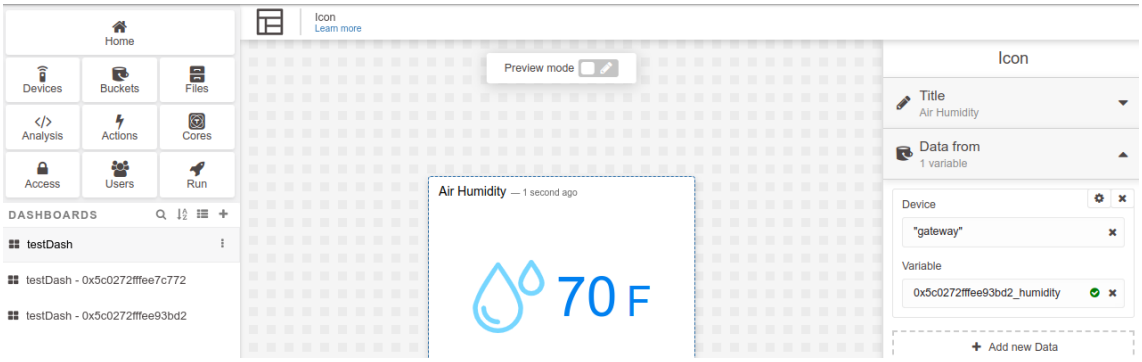
Figura 21 – Adicionando um widget



Fonte: AUTOR

Nas configurações do *widget* deve-se selecionar o dispositivo que você deseja e a variável que você deseja monitorar, conforme indica na Figura 22:

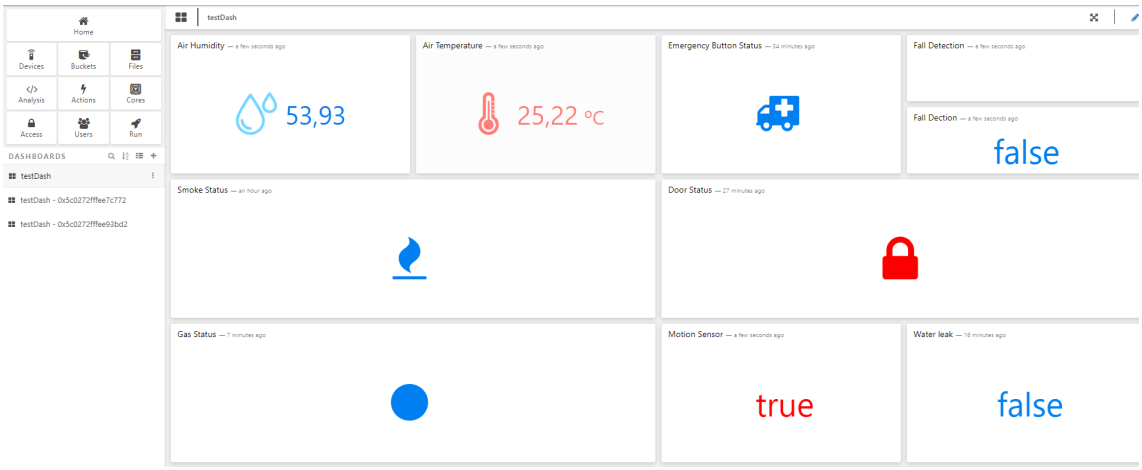
Figura 22 – Configurando o widget para mostrar o valor da variável de um sensor de umidade



Fonte: AUTOR

O resultado do *dashboard* criado para o protótipo pode ser visto na imagem a seguir, onde é possível ver a monitoração em tempo real de alguns sensores, sendo eles, sensor de umidade, sensor temperatura, botão de emergência, detector de quedas, sensor de fumaça, sensor de porta, sensor de gás de cozinha, sensor de movimento e sensor de vazamento de água.

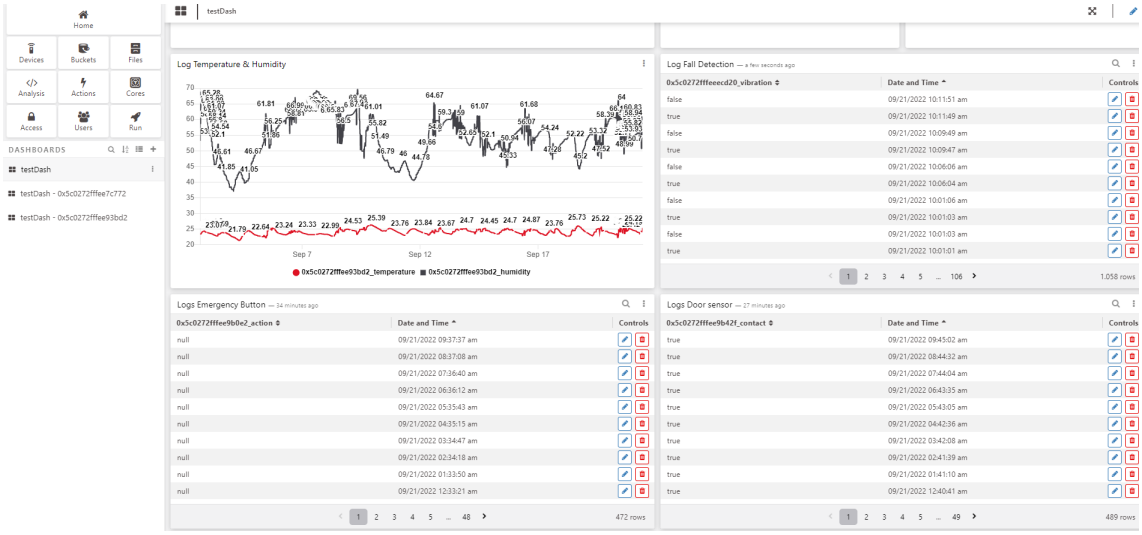
Figura 23 – Interface do dashboard criado



Fonte: AUTOR

A plataforma também oferece a criação de gráficos para os dados recebidos, como pode-se observar na imagem abaixo:

Figura 24 – Gráficos dos metadados recebidos



Fonte: AUTOR

4.3 Integração, testes e custos do protótipo

Nesta seção, serão apresentados os resultados obtidos nos testes realizados no protótipo do gateway Zigbee, bem como uma avaliação. Além disso, serão discutidas as limitações encontradas durante os testes, possíveis melhorias que podem ser implementadas no futuro, e uma análise dos custos dos protótipos, desconsiderando o tempo de desenvolvimento.

4.3.1 Integração (mecânica)

O protótipo em questão é um dispositivo projetado para oferecer um design compacto e eficiente. O case do dispositivo é feito de polímero resistente, com uma vibrante coloração vermelha, que garante uma alta visibilidade e fácil identificação.

O case foi selecionado para acomodar diversos componentes essenciais. No seu interior, encontramos a Raspberry Pi, responsável pelo processamento e controle do sistema. Além disso, há um adaptador Zigbee USB, que permite a comunicação sem fio com diversos sensores utilizando a tecnologia Zigbee.

O dispositivo também conta com um alto-falante e um microfone integrados, que permitem a transmissão e recepção de áudio em tempo real utilizando o VoIP. Essa funcionalidade é fundamental em situações de emergência, onde a comunicação rápida e clara pode ser crucial para garantir a segurança dos usuários.

O destaque do dispositivo é o botão de emergência de fácil acesso, localizado estrategicamente no case. Esse botão foi projetado para ser facilmente acionado em situações críticas, permitindo a geração imediata de uma chamada de emergência. Com um simples toque, é possível alertar rapidamente equipes de resposta a emergências, fornecendo a ajuda necessária em tempo hábil.

O case também oferece acesso para conexão de cabo de rede e alimentação, proporcionando uma fácil integração com a infraestrutura existente. Isso facilita a instalação e permite uma rápida implantação do sistema em diferentes ambientes.

Na Figura 25 é possível observar:

Figura 25 – Imagem do MVP do gateway desenvolvido.

Fonte: AUTOR

4.3.2 Apresentação dos resultados obtidos nos testes

4.3.2.1 ZigBee

A fim de avaliar o desempenho do sistema Zigbee, foram realizados testes em um ambiente controlado, onde diferentes sensores Zigbee foram simulados. Esses testes abrangeram a coleta de dados de temperatura, umidade, presença, sensor de vibração (para quedas do idoso), botão de emergência e também testes utilizando sensor de fumaça e gás.

O protótipo demonstrou sucesso ao processar e transmitir corretamente os dados dos sensores Zigbee para o dashboard em tempo real. A cobertura do sistema foi avaliada em um apartamento de 100 metros quadrados, garantindo a comunicação eficiente entre os dispositivos.

Esses resultados evidenciam a funcionalidade e confiabilidade do sistema Zigbee implementado, proporcionando uma solução efetiva para monitoramento e detecção de eventos em tempo real.

4.3.2.2 VoIP

As chamadas VoIP realizadas pelo protótipo utilizando o protocolo SIP foram bem-sucedidas, com boa qualidade de áudio. Durante os testes realizados com o protótipo utilizando o protocolo SIP, foram avaliados diferentes codecs de áudio para garantir uma boa qualidade de áudio. Os codecs de áudio são algoritmos responsáveis pela compressão e descompressão dos dados de áudio transmitidos durante as chamadas.

Foram considerados os seguintes codecs de áudio nos testes:

- G.711: É um codec de áudio sem compressão, que oferece alta qualidade de áudio, mas requer maior largura de banda. É amplamente utilizado em redes locais (LANs) devido à sua qualidade.
- G.729: É um codec de áudio com compressão, que oferece boa qualidade de áudio com menor largura de banda. É ideal para cenários de rede com restrições de largura de banda, como conexões de internet lentas.
- iLBC: O iLBC (Internet Low Bitrate Codec) é um codec de áudio projetado para comunicações em redes com largura de banda limitada e latências mais altas.

Durante os testes, foram considerados diferentes cenários para avaliar a qualidade e o desempenho dos codecs de áudio:

- PABX em nuvem com o protótipo atrás de NAT: Nesse cenário, o protótipo de chamadas VoIP está localizado atrás de um roteador NAT (Network Address Trans-

lation), conectado a um PABX em nuvem. O objetivo é verificar a interoperabilidade e a qualidade das chamadas VoIP através do NAT, considerando diferentes codecs de áudio.

- Utilizando o protótipo com um PABX em rede local: Nesse cenário, o protótipo de chamadas VoIP está conectado a um PABX local em uma rede local (LAN). O objetivo é avaliar a qualidade das chamadas VoIP na rede interna, considerando diferentes codecs de áudio.

Os testes foram realizados para verificar a qualidade de áudio, a latência, a perda de pacotes e a estabilidade das chamadas em cada cenário. Foram analisados os resultados obtidos em termos de clareza do áudio, sincronização de voz, atrasos perceptíveis e possíveis problemas de compatibilidade com os codecs utilizados.

Com base nos resultados dos testes, foram selecionados os codecs de áudio que ofereceram a melhor qualidade de áudio e desempenho adequado para cada cenário específico. Essa escolha foi feita levando em consideração os requisitos de largura de banda, as restrições da rede e a qualidade desejada para as chamadas VoIP. Para o cenário onde tem um rede com baixa latência de rede o codec G711 pode ser utilizado com uma ótima qualidade na chamada. Agora nos cenários com altas latências e banda limitada o iLBC foi selecionado.

Resultados impressionantes no que diz respeito à comunicação e processamento dos sinais dos sensores foi constatado, permitindo uma transmissão ágil e confiável das informações.

Um dos aspectos mais notáveis foi o tempo de resposta rápido entre a coleta dos dados e sua visualização em tempo real no dashboard. Esse tempo garantiu um monitoramento instantâneo, possibilitando aos usuários acessar as informações mais recentes de forma imediata. Essa característica é de suma importância em cenários onde a tomada de decisões rápidas é essencial.

Outro ponto destacado na avaliação foi a estabilidade das chamadas VoIP realizadas pelo gateway. Essas chamadas foram executadas de maneira consistente, sem apresentar problemas significativos de latência ou perda de pacotes. Essa estabilidade é crucial em comunicações de voz sobre IP, garantindo uma experiência de chamada fluida e de qualidade.

Cabe destacar que esses testes foram realizados em um cenário específico e sem formalidades estatísticas para comprovação da eficiência e qualidade do protótipo. De qualquer forma, o que observamos foi que para todos os testes realizados, foi possível obter as informações em tempo real e sem perdas significativas na comunicação.

4.3.3 Discussão sobre as limitações e possíveis melhorias

Durante os testes, identificamos uma limitação relacionada à distância de comunicação entre o gateway Zigbee e os sensores. Em algumas situações, a perda de sinal ocorreu em distâncias maiores que 30 metros. Então para melhoria do cenário foi visto que alguns sensores zigbee (normalmente os que são alimentados em 220V) podem trabalhar como um amplificador de sinal de cobertura trabalhando em rede mesh.

Além disso, a adição de recursos de segurança, como autenticação de chamadas VoIP, poderia ser uma melhoria importante para garantir a integridade e privacidade das comunicações.

Os resultados dos testes evidenciam a eficiência do protótipo do gateway Zigbee, tanto na integração e processamento dos dados dos sensores quanto nas chamadas VoIP realizadas através do protocolo SIP. Apesar das limitações identificadas, as melhorias propostas têm o potencial de aprimorar ainda mais o desempenho e a funcionalidade do gateway. É fundamental considerar esses resultados e discussões para futuros aprimoramentos e aplicações práticas do protótipo desenvolvido.

4.3.4 Custos

De acordo com a Tabela 1, que contém os preços dos sensores e peças utilizados no desenvolvimento do protótipo, é importante ressaltar que os valores apresentados não incluem o custo de desenvolvimento. É fundamental considerar que a tabela abrange apenas os componentes e dispositivos físicos necessários para construir o protótipo final.

Ao analisar os custos dos produtos listados na tabela, é possível ter uma ideia preliminar dos gastos associados à montagem e fabricação do gateway Zigbee. No entanto, é importante ressaltar que o custo de desenvolvimento, que engloba o tempo, esforço e recursos investidos na concepção, projeto, testes e implementação do sistema, não está contemplado nos valores mencionados.

A inclusão do custo de desenvolvimento é uma variável crítica a ser considerada ao calcular o preço final de um produto. Esse custo leva em conta a equipe de desenvolvimento, a pesquisa de mercado, os protótipos, as iterações e a validação do protótipo. Portanto, é necessário ponderar esse fator ao avaliar a viabilidade financeira do projeto.

Embora a tabela forneça uma referência útil para estimar os custos dos componentes físicos, é fundamental reconhecer que o valor total do protótipo final será impactado significativamente pelo custo de desenvolvimento, que pode variar dependendo do escopo e da complexidade do projeto.

Tabela 1 – Custos para o protótipo

Produto	**Preço (Reais)**
Raspberry Pi 4 model b	729,90
Alto falante USB P2	30,30
Mini Microfone USB 2.0	40
Case plástico	50
Sonoff Zigbee 3.0 USB	144
Sensor de gás de cozinha	152
Sensor de fumaça	149
Sensor de vibração	101
Sensor de temperatura e umidade	101,18
Botão de emergência	86
Sensor de porta	151
Sensor de alagamento	106
Sensor de movimento	136
TOTAL	R$ 1.976,38

Fonte: Elaborada pelo autor.

5 CONCLUSÕES

Ao finalizar este projeto, conseguimos desenvolver um protótipo operacional, projetado como um sistema de monitoramento e geração de alertas emergenciais para idosos. Esse sistema é capaz de integrar-se com uma ampla variedade de sensores Zigbee disponíveis no mercado, o que nos possibilita criar um ambiente de monitoramento personalizado e adequado para cada caso específico.

Além do monitoramento passivo, o protótipo atua como um produto IP de teleassistência, permitindo suporte remoto ao idoso, o que potencializa a segurança e a prontidão em emergências. A solução possui um painel de controle para monitoramento em tempo real e uma interface intuitiva para ajustes de rede, SIP e sensores. Essas funcionalidades permitem um acompanhamento constante e uma configuração eficaz do sistema pelos cuidadores, garantindo uma resposta rápida ao idoso e reduzindo riscos de acidentes.

Para que este protótipo se torne um produto comercialmente viável, serão necessários esforços adicionais de desenvolvimento. Uma melhoria crítica a ser implementada é uma opção de fallback para o envio de dados e comunicação, como a integração de um modem 4G/5G. Isso garantiria a continuidade do serviço em situações em que a rede Wi-Fi ou cabeada não esteja disponível.

Além disso, vemos um potencial significativo para aprimorar o sistema por meio do monitoramento das atividades diárias do usuário. Acompanhar padrões de sono, exercícios físicos, ingestão de medicamentos, entre outros, poderia proporcionar insights úteis e permitir a personalização de alertas para cada idoso.

Outro avanço a ser considerado é a implementação de técnicas avançadas de análise de dados e aprendizado de máquina. Estas ferramentas poderiam ajudar a identificar padrões, tendências e comportamentos anormais nos dados coletados, possibilitando a detecção precoce de problemas de saúde ou situações de risco. A integração com serviços de telemedicina também seria um avanço significativo, possibilitando consultas médicas remotas e o compartilhamento de informações de saúde em tempo real.

Em suma, o protótipo desenvolvido já se mostra como uma solução promissora no campo do monitoramento de idosos, proporcionando segurança e assistência. Com as melhorias sugeridas, vemos um grande potencial para transformar este protótipo em um produto comercialmente viável. Isso certamente contribuirá para o bem-estar e a qualidade de vida dos idosos, além de proporcionar tranquilidade e suporte para seus cuidadores. Portanto, o trabalho realizado representa um passo significativo na direção de um monitoramento mais seguro e eficaz para a população idosa.

REFERÊNCIAS

ANDREIS, I. Estudo comparativo entre bibliotecas de implementação para o protocolo sip. 2010. Disponível em: <https://www.lume.ufrgs.br/handle/10183/28312>. Acesso em: 1 dez 2022. 22, 23

ANDRIGHETTO, E. *Sistema de processamento de sinais biomédicos: rede wireless ZigBee com aplicação do padrão IEEE 802.15.4*. 2008. Disponível em: <https://repositorio.ufsc.br/handle/123456789/92085>. Acesso em: 23 nov 2022. 17

CONCEIÇÃO, W. N. E. da; COSTA, R. M. de R. Análise do protocolo mqtt para comunicação iot através de um cenário de comunicação. *Caderno de Estudos em Sistemas de Informação*, v. 5, n. 2, 2019. Acesso em: 29 nov 2022. 19

FOUNDATION, R. P. *Raspberry Pi Foundation – About us*. 2012. Disponível em: <https://www.raspberrypi.org/about/>. Acesso em: 15 nov 2022. 28

IBM. *Padrões e requisitos do MQTT - Documentação da IBM*. 2021. Disponível em: <https://www.ibm.com/docs/pt-br/maximo-monitor/8.2.0?topic=messaging-standards-requirements>. Acesso em: 29 nov 2022. 21

KHAN, A. *Raspberry Pi 4 GPIO Pinout*. 2022. Disponível em: <https://linuxhint.com/gpio-pinout-raspberry-pi/>. Acesso em: 21 nov 2022. 29

KINDERMANN, L. de M.; SANTOS, E. L. F. dos. *ESTUDO SOBRE O PADRÃO ZIGBEE E DESENVOLVIMENTO DE SISTEMA APLICADO EM AUTOMAÇÃO RESIDENCIAL*. Revista Ilha Digital, 2012. Disponível em: <https://ilhadigital.florianopolis.ifsc.edu.br/index.php/ilhadigital/article/view/35/34>. Acesso em: 23 nov 2022. 17, 18, 19

MARTINEZ, J. G. A.; JR, J. M. da S. Desenvolvimento de uma aplicação voip baseada no protocolo sip. Universidade do Estado do Amazonas, 2011. Disponível em: <http://tiagodemelo.info/monografias/2011/tcc-joao-martinez.pdf>. Acesso em: 1 dez 2022. 23

MARTINS, I. R.; ZEM, J. L. Estudo dos protocolos de comunicação mqtt e coap para aplicações machine-to-machine e internet das coisas. Faculdade de Tecnologia de Americana, 2015. Acesso em: 29 nov 2022. 19, 20, 21

OMS. *RELATÓRIO MUNDIAL DE ENVELHECIMENTO E SAUDE*. 2015. 1-30 p. Disponível em: <https://sbgg.org.br/wp-content/uploads/2015/10/OMS-ENVELHECIMENTO-2015-port.pdf>. Acesso em: 18 nov 2022. 15

ORVIBO. *Botão de Emergência - Orvibo*. Disponível em: <http://www.orvibobrasil.com.br/product/botao-de-emergencia/>. Acesso em: 25 fev 2023. 33, 35

PERKINS, C. Rtp: Audio and video for the internet. *Addison-Wesley Professional*, 2003. 25

RASPBERRY Pi 4. Disponível em: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>. Acesso em: 15 nov 2022. 28

RILEY, G. F. *Understanding Voice over IP Technology*. [S.l.]: Pearson Education, 2005. 22

ROSENBERG, J. et al. *SIP: session initiation protocol*. 2002. Disponível em: <https://dl.acm.org/doi/pdf/10.17487/RFC3261>. Acesso em: 1 dez 2022. 22

RUST, T. Fuzzing zigbee using z-stack. 2022. Disponível em: <http://www.cs.ru.nl/bachelors-theses/2022/Tom_Rust____1040068____Fuzzing_Zigbee_using_Z-Stack.pdf>. Acesso em: 1 dez 2022. 22

SANTOS, A. V. *UMA SOLUÇÃO SDN PARAA COMUNICAÇÃ MESH DE NÓS EM UMA REDE ZIGBEE PADRÃO 802.15.4*. Dissertação (Mestrado) — UNIVERSIDADE FEDERAL DO CEARÁ, Ceará, 2015. Disponível em: <https://repositorio.ufc.br/handle/riufc/25000>. Acesso em: 23 nov 2022. 18

SANTOS, R. *Install Mosquitto Broker Raspberry Pi | Random Nerd Tutorials*. 2022. Disponível em: <https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/>. Acesso em: 24 fev 2023. 10, 43, 44

SARL, L. B. C. 2020. Disponível em: <https://www.linphone.org/technical-corner/linphone>. Acesso em: 24 fev 2023. 38

SCHULZRINNE, H. et al. Rtp: A transport protocol for real-time applications. In: *Proc. ACM SIGCOMM*. [S.l.: s.n.], 2003. 24

SONOFF. *SONOFF Zigbee 3.0 USB Dongle Plus-P-SONOFF Official*. 2022. Disponível em: <https://sonoff.tech/product/diy-smart-switch/sonoff-zigbee-3-0-usb-dongle-plus-p/>. Acesso em: 18 nov 2022. 29, 30

SOUSA, J. P.; CARRAPATOSO, E. Uma arquitectura iptel baseada no protocolo sip. In: *6ª CONFERÊNCIA SOBRE REDES DE COMPUTADORES (CRC'2003–Bragança, Portugal)*. Universidade de Coimbra, Instituto Politécnico de Bragança, 2003. Disponível em: <https://bibliotecadigital.ipb.pt/handle/10198/7162>. Acesso em: 1 dez 2022. 24

STOLL, G. R. *O que é este tal do zigbee*. UTC Journal-Smart Utilities Networks, Special Issue, 2008. Disponível em: <http://www.mecomp.com.br/rumo/o_que_e_este_tal_do_zigbee_3_.pdf>. Acesso em: 22 nov 2022. 17

TAGOIO. 2022. Disponível em: <https://tago.io/>. Acesso em: 1 dez 2022. 25

TUYA. *Tuya*. Disponível em: <http://www.tuya.com>. Acesso em: 25 fev 2023. 32

VALENZUELA, W. A. V. et al. *Uma abordagem teórica e prática em um protocolo para IoT*. 2021. 43405–43418 p. Acesso em: 29 nov 2022. 20

XAVIER, R. C. *Conheça a TagoIO, a primeira ferramenta cloud para desenvolvimento de solução IoT homologada Khomp*. 2019. Disponível em: <https://www.khomp.com/pt/tagoio-solucao-iot/>. Acesso em: 1 dez 2022. 25

ZIGBEE2MQTT. *Linux | Zigbee2MQTT*. 2022. Disponível em: <https://www.zigbee2mqtt.io>. Acesso em: 22 nov 2022. 10, 34, 36, 37, 44, 45, 47

Apêndices

APÊNDICE A – LINPHONERC

Código A.1 – Arquivo de configuração linphonerc

```
[sound]
playback_dev_id=ALSA Unknown: bcm2835 Headphones
ringer_dev_id=ALSA Unknown: bcm2835 Headphones
capture_dev_id=ALSA Unknown: USB PnP Sound Device
media_dev_id=ALSA Unknown: bcm2835 Headphones
playback_gain_db=12.000000
mic_gain_db=6.000000
echocancellation=1
el_type=mic
el_thres=0.09
el_force=200000
el_sustain=100
el_transmit_thres=1.7
ng_floorgain=0.01
ec_delay=200
ec_tail_len=150
ec_frame_size=128
echolimiter=0
remote_ring=/home/pi/linphone-sdk/build-raspberry/linphone-sdk/desktop/share/sounds
    /linphone/ringback.wav
disable_record_on_mute=0

[net]
nat_policy_ref=~SxRM6wMhArnGeU
download_bw=0
upload_bw=0
mtu=1300
stun_server=stun.linphone.org
firewall_policy=3

[nat_policy_0]
ref=~SxRM6wMhArnGeU

[sip]
root_ca=/home/pi/linphone-sdk/build-raspberry/linphone-sdk/desktop/share/linphone/
    rootca.pem
verify_server_certs=1
verify_server_cn=1
contact=6000@192.168.110.80
media_encryption=none
guess_hostname=1
```

```
inc_timeout=30
in_call_timeout=0
delayed_timeout=4
register_only_when_network_is_up=1
register_only_when_upnp_is_ok=1
default_proxy=0

[app]
auto_download_incoming_files_max_size=-1
sender_name_hidden_in_forward_message=0

[misc]
prefer_basic_chat_room=1
uuid=fcfe0eb9-ed9b-0065-bf85-ed4dab0c16de
user_certificates_path=/root/.linphone-usr-crt

[video]
capture=0
display=0
show_local=0
size=vga

[rtp]
audio_rtp_port=7078
video_rtp_port=9078
text_rtp_port=11078
audio_jitt_comp=60
video_jitt_comp=60
nortp_timeout=30
audio_adaptive_jitt_comp_enabled=1
video_adaptive_jitt_comp_enabled=1

[audio_codec_0]
mime=opus
rate=48000
channels=2
enabled=1
recv_fmtp=useinbandfec=1

[audio_codec_1]
mime=speex
rate=16000
channels=1
enabled=1
recv_fmtp=vbr=on

[audio_codec_2]
```

```
mime=speex
rate=8000
channels=1
enabled=1
recv_fmtp=vbr=on

[audio_codec_3]
mime=PCMU
rate=8000
channels=1
enabled=1

[audio_codec_4]
mime=PCMA
rate=8000
channels=1
enabled=1

[audio_codec_5]
mime=GSM
rate=8000
channels=1
enabled=0

[audio_codec_6]
mime=G722
rate=8000
channels=1
enabled=0

[audio_codec_7]
mime=iLBC
rate=8000
channels=1
enabled=0
recv_fmtp=mode=30

[audio_codec_8]
mime=speex
rate=32000
channels=1
enabled=0
recv_fmtp=vbr=on

[audio_codec_9]
mime=L16
rate=44100
```

```
channels=2
enabled=0

[audio_codec_10]
mime=L16
rate=44100
channels=1
enabled=0

[video_codec_0]
mime=VP8
rate=90000
enabled=1

[video_codec_1]
mime=H264
rate=90000
enabled=1
recv_fmtp=profile-level-id=42801F

[auth_info_0]
username=6000
ha1=2bbf820b3d0816478c3f9ba102ff0585
realm=192.168.110.80
domain=192.168.110.80
algorithm=MD5
available_algorithms=MD5

[proxy_0]
reg_proxy=sip:192.168.110.80
reg_identity=sip:6000@192.168.110.80
quality_reporting_enabled=0
quality_reporting_interval=0
reg_expires=600
reg_sendregister=1
publish=1
avpf=-1
avpf_rr_interval=5
dial_escape_plus=0
privacy=32768
push_notification_allowed=0
idkey=proxy_config_hL4WuVarpviqEQh
publish_expires=-1
```

APÊNDICE B – SCRIPT DA BOTOEIRA

Código B.1 – Script que monitora o estado do botão e inicia uma chamada SIP

```
import os
import sys
from time import sleep
import wiringpi

wiringpi.wiringPiSetupGpio()
wiringpi.pinMode(18, 0)

while True:
    if (wiringpi.digitalRead(18)== 1 and (not os.system("linphonecsh generic 'calls
    '"))):
        sleep(1)
        os.system("linphonecsh dial 6002@192.168.110.80")
```

APÊNDICE C – SCRIPT TAGO.IO

Código C.1 – Script le os metadados recebidos e envia para os dispostivos

```
const { Analysis, Device, Account, Utils } = require("@tago-io/sdk");

async function mqtt2devices(context, scope) {
  const envVars = Utils.envToJson(context.environment);
  if (!envVars.account_token) {
    context.log("Error Access Token. Please configure.");
    return;
  }

  const account = new Account({ token: envVars.account_token });
  debug(`Scope: ${JSON.stringify(scope)}`);

  try {
    const mqttTopic = scope[0].metadata.mqtt_topic;
    const macAddress = mqttTopic.split("/")[1];
    const deviceList = await account.devices.list({
      page: 1,
      fields: ["id"],
      amount: 1,
      filter: {
        tags: [
          {
            key: "mac_address",
            value: macAddress,
          },
        ],
      },
    });


    const deviceToken = await Utils.getTokenByName(account, deviceList[0].id);
    const deviceTarget = new Device({ token: deviceToken });
    const inputDataResult = await deviceTarget.sendData(scope);
    log(`Input data to ${macAddress}: ${inputDataResult}`);
  } catch (error) {
    error(`Error: ${error}`);
  }
}

module.exports = new Analysis(mqtt2devices);
```