UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

JOÃO VÍTOR BUSCAINO MERGENER - 00265016

PROPOSTA DE UM *FRAMEWORK* PARA AUTOMAÇÃO RESIDENCIAL VISANDO UM AMBIENTE RESTAURADOR

Porto Alegre
2024

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

JOÃO VÍTOR BUSCAINO MERGENER - 00265016

PROPOSTA DE UM *FRAMEWORK* PARA AUTOMAÇÃO RESIDENCIAL VISANDO UM AMBIENTE RESTAURADOR

Trabalho de Conclusão de Curso (TCC-CCA) apresentado à COMGRAD-CCA da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de *Bacharel em Eng. de Controle e Automação* .

ORIENTADOR:
Prof. Dr. Fabiano Disconzi Wildner

Porto Alegre
2024

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

JOÃO VÍTOR BUSCAINO MERGENER - 00265016

PROPOSTA DE UM *FRAMEWORK* PARA AUTOMAÇÃO RESIDENCIAL VISANDO UM AMBIENTE RESTAURADOR

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção dos créditos da Disciplina de TCC do curso *Eng. de Controle e Automação* e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Fabiano Disconzi Wildner, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Banca Examinadora:

Prof. Dr. Fabiano Disconzi Wildner, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Marcelo Götz, UFRGS
Doutor pela Universität Paderborn – Paderborn, Alemanha

Prof. Dr. Rafael Antônio Comparsi Laranja, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

_____
Alceu Heinke Frigeri
Coordenador de Curso
Eng. de Controle e Automação

Porto Alegre, Fevereiro 2024

AGRADECIMENTOS

Agradeço o apoio da minha família, em especial dos meus pais, que sempre me incentivaram na minha busca pelo conhecimento, dando todo apoio necessário. Agradeço aos meus amigos, por compartilharem comigo os momentos de alegria e também de tristeza. Um agradecimento especial ao meu amigo Pedro Henrique Rettore, por auxiliar na implementação da interface visual deste projeto. Agradeço à minha namorada, por ser minha companheira de vida e proporcionar o apoio emocional necessário. Agradeço ao meu orientador, pelo auxílio na elaboração da proposta deste trabalho e por me manter engajado na realização deste projeto.

RESUMO

Este trabalho apresenta o projeto completo de um sistema de automação residencial que utiliza como dado de entrada a leitura da frequência cardíaca de um indivíduo e utiliza esses dados para controle de dispositivos com o intuito de promover um ambiente restaurador, reduzir o estresse e melhorar a qualidade de vida do usuário. O trabalho contempla os passos realizados para obtenção da leitura da frequência cardíaca a partir de uma pulseira inteligente disponível no mercado. Contempla a implementação de uma plataforma web para que o usuário possa interagir com as variáveis que definem o sistema e também visualizar o histórico do monitoramento de sua frequência cardíaca. Há também a escolha e implementação do protocolo de comunicação para integração de todos os módulos do sistema. Neste trabalho há o projeto de dispositivos com o intuito de gerar um ambiente restaurador, sendo eles: um dispositivo para envio de comandos infravermelhos para interação com um condicionador de ar e controle da temperatura do ambiente, um dispositivo para controle da cor e intensidade da iluminação do ambiente, e um dispositivo para atuar como uma tomada inteligente. O projeto desses dispositivos contempla a escolha de um microcontrolador para servir como base, o princípio de funcionamento e diagramas esquemáticos contendo o circuito implementado de cada dispositivo. Todos os dispositivos foram então instalados no quarto da residência do autor, que configurou o sistema para suas preferências e utilizou-o ao decorrer de um dia para poder avaliar a efetividade do ambiente restaurador gerado e a lógica de ativação do sistema em relação ao estado emocional do usuário. Os resultados dessa avaliação foram julgados como satisfatórios, mas apontaram a necessidade da utilização de mais sinais vitais como entrada do sistema para eliminação de falsos positivos ao caracterizar um estado de estresse do usuário.

Palavras-chave: Engenharia de Controle e Automação, MQTT, Automação Residencial, Ambiente Restaurador, Estresse, IoT, Internet das Coisas, Esp32.

ABSTRACT

This work presents the complete project of a home automation system that uses an individual heart rate frequency as input data to control devices aimed at fostering a restorative environment, reducing stress, and enhancing the user's quality of life. The work contemplates the steps taken to obtain a heart rate reading from a smart band available on the market. It includes the implementation of a web platform allowing the user to interact with the system's defining variables and to visualize the history of their heart rate monitoring. Additionally, it involves the selection and implementation of a communication protocol to integrate all system modules. Within this work, there is the design of devices intended to create a restorative environment, including: a device for sending infrared commands to interact with an air conditioner and control room temperature, a device for controlling ambient lighting color and intensity, and a device acting as a smart socket. The design of these devices involves the selection of a microcontroller as a base, the working principle, and schematic diagrams containing the implemented circuit of each device. All devices were then installed in the author's bedroom, who configured the system to his preferences and used it for one day to assess the effectiveness of the generated restorative environment and the system activation logic concerning the user's emotional state. The results of this evaluation were deemed satisfactory, but indicated the need for incorporating additional vital signs as system inputs to eliminate false positives when characterizing an user's stress state.

Keywords: Control and Automation Engineering, MQTT, Home Automation, Restorative Environment, Stress, IoT, Internet of Things, Esp32.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE LISTAGENS

SUMÁRIO

1 INTRODUÇÃO

A ansiedade é uma emoção normal que representa um "sinal de alerta" ao se deparar com situações de perigo ou ameaça real ou imaginária. Originalmente essa emoção serve para proteção, porém quando ela ocorre de forma exacerbada ou desproporcional, ela pode ser classificada como um transtorno de ansiedade, passível de tratamento (BRASIL, 2023).

A ansiedade pode se manifestar com sintomas corporais, como batimentos cardíacos acelerados e aumento da frequência respiratória. Esses sinais podem vir em conjunto com pensamentos de dúvida, antecipação de problemas ou pensamentos catastróficos (BRASIL, 2023). Essa emoção é caracterizada como um transtorno mental quando se torna uma emoção desconfortável e perturbadora sem uma razão clara ou ameaça externa que justifique esta reação no indivíduo. Nessas situações, a intensidade, duração, ou frequência da ansiedade são maiores do que o que seria esperado, resultando em angústia para o indivíduo. Essa ansiedade excessiva também está ligada a dificuldades no funcionamento social ou profissional de um ser humano, prejudicando sua qualidade de vida (BRASIL, 2023).

A partir de um levantamento sobre transtornos de ansiedade realizado em 2015 pela Organização Mundial da Saúde (OMS) (OMS, 2017), pode-se concluir que o Brasil é o país que contém a população que mais sofre deste transtorno no mundo. O Brasil lidera este ranking tendo uma estimativa de que aproximadamente 9,3% de sua população está sofrendo com ansiedade patológica. Em seguida no ranking aparece o Paraguai, com 7,6%, Noruega, com 7,4%, Nova Zelândia, com 7,3% e Austrália com 7%. A pesquisa demonstra uma grande diferença entre os percentuais do Brasil e os outros países, reforçando a dimensão do problema no país. Outro dado importante, advém de um resumo científico divulgado pela OMS em 2022 (OMS, 2022). O resumo conclui que a pandemia de COVID-19 aumentou em 25% a prevalência global de ansiedade e depressão, agravando o problema citado anteriormente.

O estresse pode ser um gatilho para os transtornos de ansiedade (SONSIN, 2018). Quando uma pessoa é submetida ao estresse, hormônios como adrenalina, noradrenalina e cortisol são liberados resultando no aumento da frequência cardíaca (BAUER, 2002). Isso possibilita a identificação do estresse através da leitura dos batimentos cardíacos.

Diante desse cenário, surge a ideia de implementar um sistema de automação residencial com o foco no bem estar de um indivíduo, realizando o controle das condições ambientais da residência promovendo um ambiente restaurador, visando reduzir os níveis de estresse do usuário e os gatilhos para o transtorno de ansiedade.

1.1 Objetivos do trabalho

O trabalho tem como objetivo o desenvolvimento de um sistema para auxílio ao bem estar pessoal, onde através da modificação das condições ambientais de uma residência

é promovida uma sensação de relaxamento ao usuário. O sistema consiste na leitura da frequência de batimentos cardíacos a partir de uma pulseira inteligente. A frequência lida serve como base para a definição do estado emocional desse indivíduo. Esses dados são armazenados possibilitando o monitoramento da saúde pelo próprio indivíduo e servem como entrada para um sistema de controle de variáveis como iluminação, temperatura, aromas, entre outros.

A partir de uma interface gráfica o usuário pode definir o modo de funcionamento, seja ele automático (definindo faixas de operação através da frequência cardíaca do usuário) ou manual (mandando comandos diretamente para os dispositivos integrados no sistema).

A seguir estão listados de forma mais objetiva os intentos do trabalho.

- Realização da leitura de batimentos cardíacos de um indivíduo utilizando como dados de entrada em um sistema de automação residencial
- Armazenamento desses dados de batimentos cardíacos para monitoramento da saúde do indivíduo
- Criação de um dispositivo que realiza o acionamento de condicionadores de ar a partir de sinais infravermelhos
- Produção de um aparelho que atua como tomada inteligente para realização de acionamentos remotos
- Elaboração de um instrumento que possibilita o controle da intensidade e cor da iluminação de um ambiente
- Implementação de um sistema que realiza a integração e programação de todos estes dispositivos
- Desenvolvimento, para o sistema, de uma interface de usuário simples para que ela seja utilizada sem conhecimento prévio
- A partir de estímulos para o relaxamento, tornar a vida do usuário mais tranquila

1.2 Organização

A estrutura da proposta de trabalho está dividida em quatro capítulos, além da Introdução: Revisão da Literatura, Metodologia, Resultados e Conclusões. No Capítulo de Revisão da Literatura são apresentadas soluções já implementadas para projetos de automação residencial, bem como uma revisão da utilização de um ambiente restaurador para atenuar os efeitos do estresse. Além disso, são apresentados conceitos técnicos de tecnologias utilizadas na área de automação residencial para uma melhor compreensão desta proposta. No Capítulo de Metodologia é apresentado o projeto da implementação do sistema discutido na Seção 1.1. No Capítulo de Resultados é apresentada a montagem prática dos dispositivos, bem como os testes de funcionamento do sistema. Por fim, no Capítulo de Conclusões são apresentadas as considerações finais do projeto.

2 REVISÃO DA LITERATURA

Neste Capítulo são analisadas algumas soluções já implementadas na área de automação residencial, a relação entre ambientes restauradores e ansiedade, bem como o funcionamento de algumas tecnologias utilizadas nestas soluções para a melhor compreensão do projeto apresentado neste documento.

2.1 Soluções já existentes

Primeiramente, fez-se uma pesquisa para procurar soluções já existentes de projetos de automação residencial, a seguir são apresentados três exemplos:

1. **Projeto de monitoramento domiciliar de pacientes que utilizam marcapasso realizado pela IBM** (IBM, 2023): Dados sobre o paciente e o marcapasso são transferidos para um outro aparelho através de radiofrequência, tipicamente esta transmissão ocorre durante a noite para este aparelho que fica localizado ao lado da cama. Esse dispositivo após receber os dados encaminha-os para o hospital para que profissionais da saúde possam fazer a análise. Esse sistema reduz o número de visitas ao hospital que um paciente deve realizar, pois ele detecta quando o paciente ou o dispositivo realmente necessitam de atenção. Em caso de emergência, o sistema ainda pode alertar um médico em tempo real.

2. **Projeto de um sistema de segurança residencial inteligente realizado pela Bevywise** (BEVYWISE, 2020): O sistema consiste na integração de vários sensores convencionais de segurança residencial, como detectores de fumaça, detectores de monóxido de carbono, sensores de abertura de portas, sensores de temperatura, entre outros. Os dados destes dispositivos são enviados através do protocolo de comunicação MQTT para uma central que os armazena. Através de uma aplicação que pode ser acessada remotamente, um painel de controle é implementado para visualização destes dados e configuração de alarmes. Esses alarmes notificam o usuário caso qualquer irregularidade seja detectada, possibilitando ao usuário tomar medidas imediatas e proativas para prevenir potenciais maiores problemas.

3. **Projeto de automação residencial utilizando o protocolo MQTT controlado por Android/iOS por Dreher (2019)**: Esse trabalho teve como objetivo desenvolver dispositivos para automação residencial de baixo custo e instalação fácil, visando aumentar o acesso das pessoas à dispositivos inteligentes. Nesse projeto foram desenvolvidos cinco dispositivos: fonte chaveada, interruptor inteligente, tomada inteligente, termostato inteligente e uma tranca inteligente. Utilizou-se o protocolo MQTT para realizar a integração dos dispositivos e para a interface gráfica do sistema, foi desenvolvido um aplicativo para Android/iOS.

2.2 Utilização de ambientes restauradores para redução da ansiedade

Segundo Ulrich (1984), trata-se de um ambiente restaurador aquele com potencial de restaurar recursos e capacidades emocionais e funcionais comprometidas pelo estresse ou demandas cotidianas. É possível promover um ambiente restaurador através da emissão de sons, controle da iluminação, controle da temperatura, entre outras variáveis. Em um estudo realizado por Kjellgren e Buhrkall (2010), 18 participantes foram expostos a 30 minutos em um ambiente natural e em outro com uma simulação desse mesmo ambiente natural. Foi constatado pelos autores que ambos os ambientes facilitaram a redução do estresse, demonstrando que é possível criar um ambiente restaurador simulado. Outro estudo realizado por Ratcliffe, Gatersleben e Sowden (2013), concluiu por meio de entrevistas com residentes em contextos urbanos, que estímulos auditivos naturais, como som de pássaros, são potencialmente benéficos à redução do estresse.

2.3 Internet das Coisas

Segundo Dictionary.com (2023), a definição de Internet das Coisas (IoT) é "uma rede de dispositivos cotidianos, eletrodomésticos e outros objetos equipados com chips de computador e sensores que podem coletar e transmitir dados pela internet".

O conceito de um sistema IoT consiste na habilidade dos componentes envolvidos na constituição deste sistema de se conectar à internet (gillis, 2023). Isso permite que os aparelhos troquem informações entre si e também com a nuvem. Esse conceito serve como base para muitos sistemas de automação residencial, transformando objetos comuns em objetos “inteligentes”. Diante desse contexto, tornou-se essencial a aplicação do conceito de IoT no desenvolvimento deste trabalho.

Atualmente já existem modelos comerciais para dispositivos cotidianos que aplicam o conceito de IoT. A seguir são apresentados alguns módulos presentes no mercado. Outros módulos com menor relevância para o desenvolvimento deste trabalho são discutidos no Apêndice A.

2.3.1 Pulseira inteligente

A pulseira inteligente, também conhecida como *smartband*, é um dispositivo vestível que está ganhando destaque na vida moderna. Este dispositivo se conecta a um dispositivo móvel através de Bluetooth e possui funcionalidades como medição da frequência cardíaca, monitoramento de atividades físicas, monitoramento da qualidade do sono, envio de notificações de mensagens, relógio, contador de passos, controle de reprodução de músicas, entre outras. Geralmente este aparelho é equipado com sensores como acelerômetros, frequencímetro cardíaco e até GPS.

Os dados gerados pela pulseira são sincronizados com o dispositivo móvel para visualização permitindo que os usuários analisem seus hábitos e metas de saúde. Um modelo comercial de pulseira inteligente fabricado pela Xiaomi pode ser visto na Figura 1.

Figura 1: *Pulseira inteligente fabricada pela Xiaomi.*

Fonte: Xiaomi (2023).

2.3.2 Tomada inteligente

A tomada inteligente é um dispositivo que torna a automação residencial simples e acessível. O dispositivo é ligado diretamente em uma tomada convencional. Com ela é possível programar horários e controlar remotamente, através de Wi-Fi, aparelhos elétricos conectados a esta tomada, com comandos simples de liga/desliga. Alguns modelos ainda permitem realizar o monitoramento do consumo de energia elétrica. Uma grande variedade de modelos suporta uma corrente de até 10 A e podem operar na faixa de tensão de 100 V a 240 V de corrente alternada (PPA, 2023). Um exemplo comercial como este da marca PPA pode ser visto na Figura 2.

Figura 2: *Tomada inteligente fabricada pela PPA.*

Fonte: PPA (2023).

Com esse dispositivo é possível gerenciar o funcionamento de um difusor de aromas, tornando o ambiente mais agradável. Além disso, ele pode ser utilizado para controlar o acionamento de uma fonte de água zen, promovendo a criação de sons naturais relaxantes.

2.3.3 Lâmpada RGB inteligente

Lâmpadas inteligentes são dispositivos que permitem ao usuário ajustar a intensidade e mudar as cores da iluminação. Isso é realizado através do sistema Vermelho, Verde e Azul (RGB), um sistema utilizado na computação que cria diferentes cores combinando as componentes de vermelho, verde e azul em diferentes proporções (W3SCHOOLS, 2024). Com as lâmpadas inteligentes também é possível programar horários e rotinas de controle do uso do dispositivo através de uma comunicação Wi-Fi. Além disso, a eficiência energética é aprimorada, pois é possível monitorar e ajustar o consumo de energia das lâmpadas. Alguns dos modelos possuem o formato de um bulbo que podem ser ligados em diversos

Figura 3: *Lampada inteligente fabricada pela Intelbras.*

Fonte: Intelbras (2023).

soquetes, como de um abajur por exemplo. Um modelo comercial fabricado pela Intelbras pode ser visto na Figura 3.

Uma pesquisa da Universidade de Granada (MINGUILLON et al., 2017), na Espanha, sugere que a luz azul desempenha um papel benéfico no relaxamento após situações de estresse, portanto é possível utilizar a lâmpada RGB para emitir uma luz azul, criando um ambiente propício ao relaxamento.

2.3.4 Central de controle Infravermelho

Em uma residência, existe uma ampla gama de dispositivos controlados por infravermelho, como por exemplo: televisores, sistemas de ar condicionado e aparelhos de som. Normalmente, cada um destes dispositivos possui seu próprio controle remoto. A central de controle infravermelho serve para unificar e reproduzir os sinais produzidos por estes controles remotos, a partir de um único dispositivo, assim como permitir comandos e a programação de rotinas de controle através de uma comunicação Wi-Fi. Um modelo de central de controle infravermelho comercial fabricado pela Geonav pode ser visto na Figura 4.

Figura 4: *Central de controle infravermelho fabricada pela Geonav.*

Fonte: Geonav (2023).

2.4 Problema de soluções estanques citadas anteriormente

O maior problema da utilização dos módulos comerciais é que cada fabricante exige a utilização de seu próprio aplicativo para controle de seus dispositivos, não disponibilizando para o público geral nenhum kit de desenvolvimento de software (SDK) ou interface de programação de aplicações (API). Impossibilitando o desenvolvimento de interfaces

customizadas para controlar estes dispositivos. Para obter estas informações é necessário realizar uma parceria com estas empresas.

Portanto, a solução encontrada é a reprodução das funcionalidades desses dispositivos através de criações próprias, estabelecendo uma comunicação entre os dispositivos e a interface de controle desenvolvida.

2.5 Protocolos de comunicação

Como o projeto proposto requer a implementação de mais de um sistema comercial existente, é necessário estabelecer um protocolo de comunicação para garantir a troca de informações. Nesta Seção são discutidos alguns protocolos de comunicação que cumprem essa tarefa.

2.5.1 TCP/IP

Segundo (lopez, 2003) o protocolo de comunicação TCP/IP (*Transmission Control Protocol / Internet Protocol*) é um conjunto de dois dos mais importantes protocolos que constitui a pilha de protocolos usados na Internet, pois eles servem de base para a comunicação entre redes de computadores e a própria Internet.

Com o objetivo de estabelecer uma forma consistente de comunicação em redes de computadores, o protocolo TCP/IP possui uma arquitetura de encapsulamento dos dados em diferentes camadas (lopez, 2003). As camadas mais importantes são: Camada de Enlace, Camada de Rede, Camada de Transporte e Camada de Aplicação. As camadas estão ordenadas de forma crescente em nível de abstração, sendo a primeira a que está logicamente mais perto da transmissão física do dado e a última está logicamente mais perto do usuário.

2.5.2 MQTT

O MQTT (*Message Queueing Telemetry Transport*) é um protocolo de comunicação com foco no conceito de IoT, foi criado nos anos 90 pela IBM e surgiu de uma necessidade de estabelecer uma comunicação entre várias máquinas, incluindo *hardwares* com baixa capacidade de processamento como microcontroladores. É um protocolo que utiliza como camada de transporte a comunicação cliente-servidor estabelecida pelo protocolo TCP/IP (zolett; ramirez, 2020).

O princípio do funcionamento do protocolo se baseia no conceito de *publisher/subscriber* onde um *publisher* (que pode ser interpretado como "publicador") é responsável por publicar mensagens dentro de tópicos específicos na rede, enquanto um *subscriber* (ou "assinante") é aquele que consome as mensagens que foram publicadas em tópicos de seu interesse. É possível que um único dispositivo desempenhe um papel de *publisher* e *subscriber* ao mesmo tempo (zolett; ramirez, 2020).

Para assegurar a entrega das mensagens publicadas em um determinado tópico aos dispositivos inscritos existe um elemento intermediário chamado de *broker*. O *broker* opera como uma aplicação hospedada em um servidor dedicado. Para um dispositivo participar da comunicação é necessário se conectar apenas ao *broker* através do endereço

desse servidor, para então poder publicar mensagens e realizar a inscrição em tópicos relevantes, garantindo uma gestão eficiente das mensagens e uma distribuição confiável das informações. Por conta desta necessidade de se conectar apenas a um dispositivo para participar da rede, torna-se viável utilizar este protocolo em hardwares com baixa capacidade de processamento (ZOLETT; RAMIREZ, 2020).

Outra característica do protocolo é a possibilidade de trabalhar com níveis diferentes de QoS (*Quality of Service*, ou qualidade de serviço) no manejo das mensagens. Ao trabalhar com QoS 0, denominado "No máximo uma vez", é garantido apenas o envio da mensagem, sem se preocupar com a garantia do recebimento. Ao trabalhar com QoS 1, denominado "Pelo menos uma vez", é exigido uma confirmação do recebimento da mensagem através de um *feedback* do destinatário, garantindo que a mensagem tenha sido recebida pelo menos uma vez, mas não impedindo múltiplos recebimentos. Ao trabalhar com QoS 2, denominado "Exatamente uma vez", é garantido, através de mais de um *feedback* do destinatário, o recebimento da mensagem exatamente uma vez, impedindo múltiplos recebimentos (ZOLETT; RAMIREZ, 2020).

2.5.3 Comunicação por infravermelho

O protocolo de comunicação por infravermelho utiliza luz infravermelha para transmitir dados entre aparelhos eletrônicos. A faixa do infravermelho é definida por todas as ondas eletromagnéticas que possuem comprimento de onda de 700 nm até 1 mm (OLIVEIRA, 2015). Essa faixa do espectro está localizada abaixo do espectro visível da luz, logo abaixo do vermelho.

A transmissão por infravermelho funciona através da emissão de pulsos de luz que são detectados por sensores infravermelhos em dispositivos receptores, limitando a transmissão de dados a um curto alcance.

Para evitar interferências de fontes de radiação infravermelha comuns, como luz ambiente e calor, o sinal infravermelho é modulado utilizando uma frequência pouco convencional. Um exemplo de valor de frequência utilizado para modulação é de 38 kHz (OLIVEIRA, 2015). Esse processo de modulação é realizado de modo que o sinal se configura em intervalos definidos e intercalados entre dois tipos de blocos: um bloco ativo e um bloco inativo. No bloco ativo, ocorre uma oscilação da luz infravermelha na frequência de modulação, enquanto no bloco inativo, a luz infravermelha permanece ligada de maneira estática (OLIVEIRA, 2015). Por conta dessa modulação, é possível codificar um sinal gravando apenas valores numéricos que indicam os intervalos, em microssegundos, da alternância entre os blocos ativos e inativos. Um exemplo de um sinal infravermelho obtido pelo autor a partir de um controle remoto de um condicionador de ar fabricado pela LG pode ser visualizado na Listagem 1.

Listagem 1: *Exemplo de um sinal infravermelho obtido pelo autor a partir de um controle remoto fabricado pela LG.*

```
const uint16_t rawSignal[] = { 3348, 9836, 488, 1592, 504, 556, 492, 540, 492,
   540, 488, 1576, 488, 552, 488, 524, 492, 572, 492, 540, 488, 544, 488, 520,
    512, 548, 492, 520, 508, 1576, 464, 548, 492, 540, 508, 1584, 492, 1608,
   464, 1592, 508, 1564, 492, 532, 508, 1572, 488, 540, 492, 540, 492, 540,
   492, 1560, 508, 1572, 492, 1572, 492 }
```

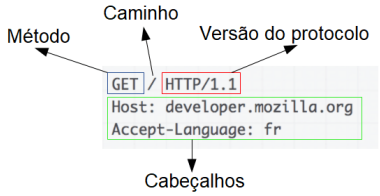
Fonte: Do Autor.

2.5.4 HTTP

O protocolo de comunicação HTTP (*HyperText Transfer Protocol*) é a base da troca de dados através da internet. Ele funciona seguindo o modelo de requisição e resposta, em que o cliente envia uma solicitação HTTP para o servidor, e o servidor responde com uma mensagem contendo os dados solicitados (MOZILLA, 2023). A base para essa comunicação acontece através de uma conexão TCP entre o cliente e o servidor (MOZILLA, 2023).

Uma requisição HTTP consiste em um método, que indica a ação a ser realizada (por exemplo, GET para obter dados, POST para enviar dados), seguido de um caminho descrito por um URI (*Uniform Resource Identifier*) que identifica o servidor e o recurso a ser utilizado do mesmo. A requisição também pode conter cabeçalhos, que fornecem informações adicionais sobre a requisição, como o tipo de conteúdo aceito pelo cliente (MOZILLA, 2023). Um exemplo de uma requisição HTTP pode ser visto na Figura 5.

Figura 5: *Exemplo de uma requisição HTTP.*



Fonte: Adaptado de Mozilla (2023).

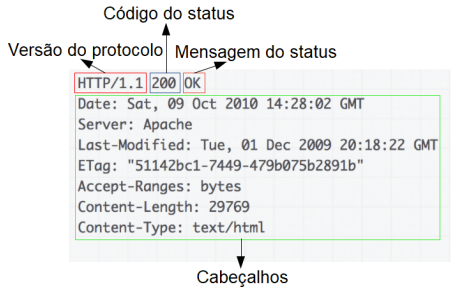
Ao receber uma requisição o servidor responde com uma mensagem parecida, contendo um código e uma mensagem do status da requisição, um cabeçalho identificando várias características da resposta e o conteúdo (MOZILLA, 2023). Um exemplo de resposta HTTP pode ser visto na Figura 6.

2.5.5 REST API

Uma REST API tem seu funcionamento baseado no protocolo de comunicação HTTP para padronizar a comunicação entre dois componentes de software. Sua principal característica é a ausência de um estado, fazendo com que cada solicitação seja tratada da mesma maneira independente do momento que ela seja realizada (SOUZA, 2020).

Uma REST API se baseia nos métodos HTTP sendo os seguintes os mais importantes:

Figura 6: *Exemplo de resposta HTTP.*

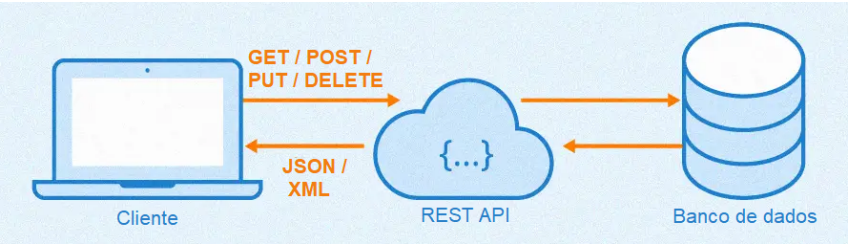


Fonte: Adaptado de Mozilla (2023).

- GET - Método utilizado para realizar uma leitura de dados do servidor
- POST - Método utilizado para enviar dados ao servidor
- DELETE - Método utilizado para excluir alguma informação no servidor
- PUT - Método utilizado para atualizações de registros no servidor

Além disso, as REST APIs geralmente utilizam formatos de dados como JSON (*JavaScript Object Notation*) ou XML (*Extensible Markup Language*) para representar e trocar informações entre os sistemas, por serem compatíveis com muitas linguagens de programação e comumente utilizados por aplicações (AMAZON, 2023b). Uma representação esquemática do funcionamento de uma REST API pode ser visto na Figura 7.

Figura 7: *Representação esquemática do funcionamento de uma REST API.*



Fonte: Adaptado de Souza (2020).

3 METODOLOGIA

Neste Capítulo são apresentados os passos seguidos para a implementação dos objetivos descritos na Seção 1.1.

3.1 Arquitetura da solução

Para implementação do sistema desenvolveu-se uma aplicação web, permitindo assim que o usuário pudesse acessar esta aplicação em qualquer aparelho que possua um navegador de internet, como por exemplo: utilizando-se de um computador ou aparelho celular. Esta aplicação conta com um banco de dados para armazenamento das configurações do usuário, assim como os dados da frequência cardíaca. Para obtenção destes dados utilizou-se uma pulseira inteligente ligada ao pulso do usuário, e através de um aplicativo denominado Pulsoid e uma REST API disponibilizada pelo mesmo, estabeleceu-se uma comunicação com a aplicação web.

Para integrar o sistema implementou-se o protocolo de comunicação MQTT, pelo seu foco em sistemas que aplicam o conceito de IoT e por conta do autor deste trabalho já possuir afinidade com este protocolo de comunicação, visto que já implementou sistemas anteriormente utilizando-o. Portanto um *broker* MQTT serviu de ligação entre os dispositivos de automação residencial desenvolvidos e a aplicação web.

A implementação dos dispositivos discutidos na Seção 1.1 tem como base um microcontrolador que possui acesso à rede Wi-Fi. Levou-se em conta, a disponibilidade de uma biblioteca para realização do acesso do microcontrolador ao *broker* MQTT. Os elementos específicos de cada dispositivo estão listados à seguir:

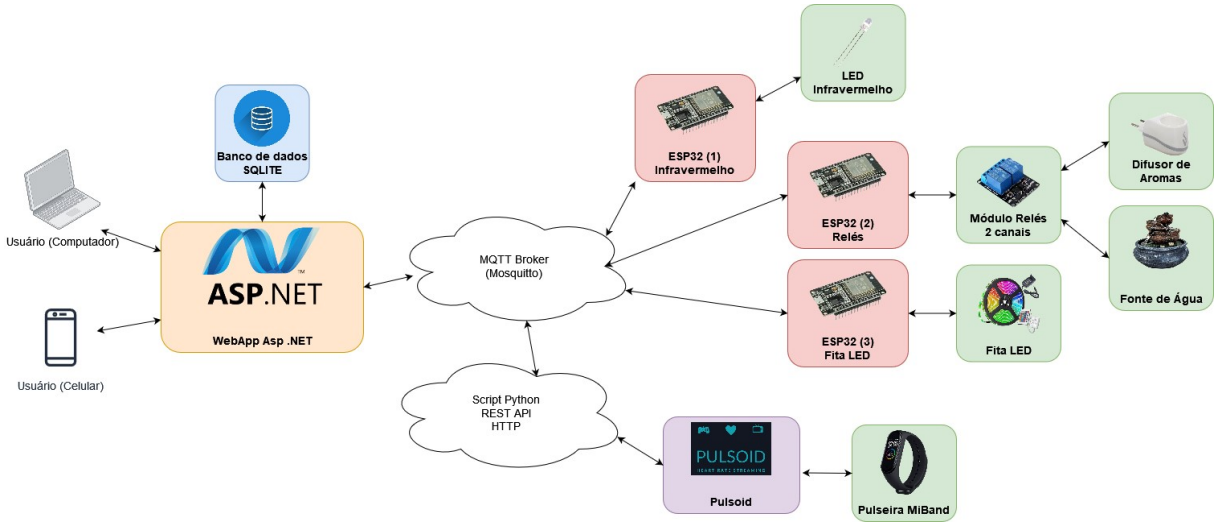
- A implementação da tomada inteligente foi feita a partir de 2 canais independentes de relés. Com isto é possível realizar o controle de liga/desliga de dispositivos que se ligam diretamente em uma tomada, como por exemplo, um difusor de aromas e uma fonte artificial de água.

- Para o controle da cor e intensidade da iluminação utilizou-se uma fita LED RGB.

- Os sinais utilizados para controlar o condicionador de ar são emitidos a partir de um LED infravermelho.

A arquitetura da solução pode ser vista na Figura 8.

Figura 8: *Arquitetura da solução.*



Fonte: Do Autor.

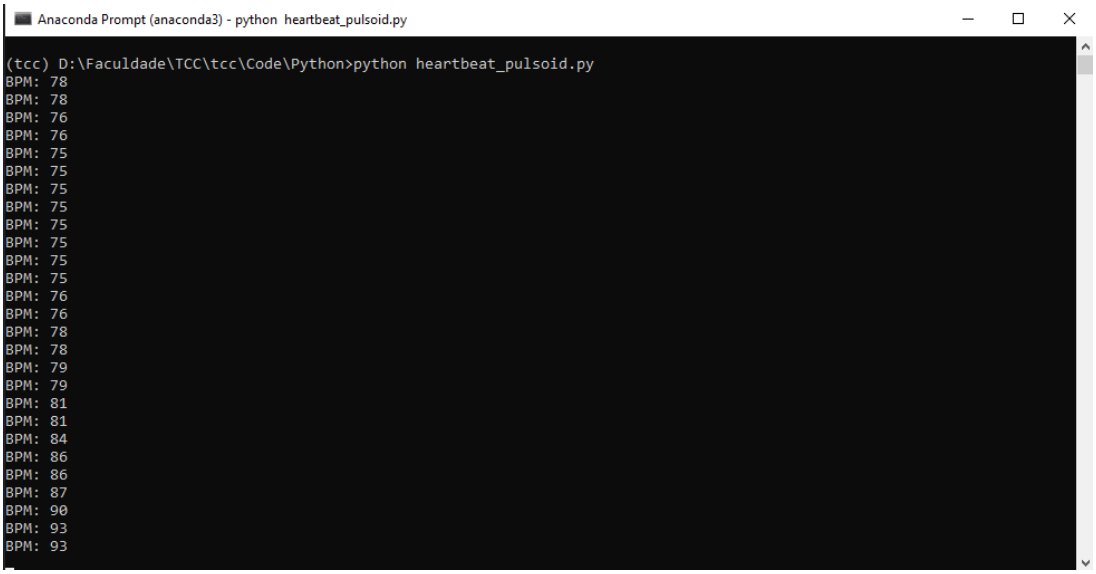
3.2 Obtenção dos batimentos cardíacos

Como o objetivo deste trabalho é aproveitar pulseiras inteligentes comerciais para obtenção da frequência de batimentos cardíacos por minuto (BPM). Portanto, utilizou-se um modelo de pulseira inteligente que o autor já possuia, a Xiaomi MiBand 4. Durante o desenvolvimento deste trabalho, por conta de um desejo pessoal do autor em atualizar o modelo da pulseira, e por representar uma melhora no monitoramento dos BPM, utilizou-se também a pulseira Xiaomi MiBand 6.

Como primeiro passo para estabelecer uma comunicação entre a aplicação principal e os dados de BPM advindos da pulseira utilizou-se um aplicativo de celular chamado Pulsoid que obtém os dados através de uma conexão Bluetooth com a pulseira. Para isso, foi essencial ajustar as configurações da pulseira no seu aplicativo oficial, Zepp Life. Isso envolveu ativar o modo de detecção por aplicativos de terceiros e permitir o compartilhamento de dados de frequência cardíaca com esses aplicativos. Também fez-se necessário ativar o modo de exercício da pulseira, para habilitar a obtenção da frequência cardíaca em tempo real. O Pulsoid então armazena estes dados em seus servidores a os atrela à uma conta de usuário, permitindo o acesso e monitoramento destes dados em um *dashboard* na web.

O Pulsoid também oferece um serviço de REST API para obtenção de informações armazenadas em seus servidores. Para autenticação de usuário é disponibilizado um *token* API obtido ao acessar a conta pela plataforma web. Para realizar a ligação entre o protocolo de comunicação MQTT e a REST API desenvolveu-se um script em python utilizando as bibliotecas "requests" (Reitz, 2023) para requisições HTTP e "paho.mqtt" (Light, 2021) para conexão com o *broker*. A cada requisição feita pela aplicação principal, o script captura o resultado da última leitura dos BPM realizada pelo Pulsoid e o publica no *broker* MQTT. O script python de obtenção dos BPM em execução pode ser visto na Figura 9.

Figura 9: *Script em python de obtenção dos BPM em execução.*



Fonte: Do Autor.

3.3 Funcionamento do sistema

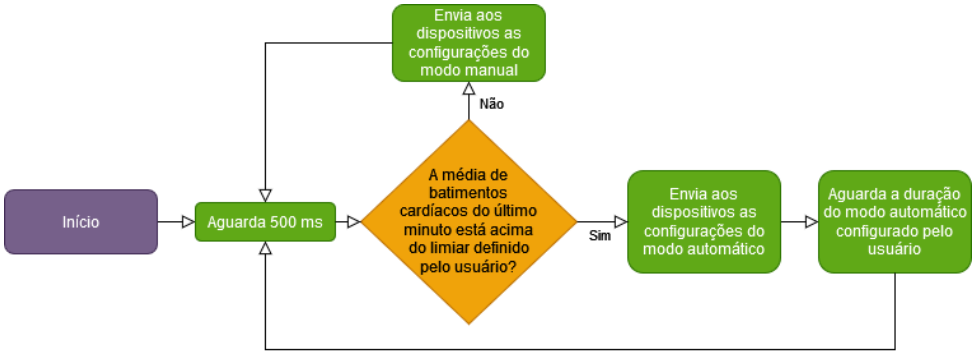
Para monitoramento do estado do indivíduo, implementou-se um serviço integrado à aplicação web operando em paralelo, solicitando a leitura dos BPM a cada segundo. Para atenuar o impacto de leituras atípicas, calcula-se uma média móvel do último minuto das leituras realizadas. Essa média é utilizada para caracterizar o estado atual do indivíduo.

O usuário pode interagir com os dispositivos de sua residência em dois modos diferentes: modo manual, no qual o usuário envia comandos assincronamente aos dispositivos, e o modo automático, no qual o usuário configura comandos a serem enviados aos dispositivos de forma síncrona associados à ativação do sistema.

Ao utilizar o modo automático, a ativação do sistema ocorre quando a média das leituras do último minuto ultrapassa um valor de limiar pré determinado pelo usuário. Uma vez ativado, o sistema permanece ativo durante um intervalo de tempo, para permitir que as novas condições do ambiente tranquilizarem o usuário. Após esse intervalo, é realizada uma nova verificação da média dos batimentos cardíacos em relação ao limiar, caso esteja abaixo do valor estipulado, o sistema então é desativado e as configurações do modo manual são enviadas para os dispositivos, permitindo que voltem ao estado anterior à ativação do sistema. Um fluxograma mostrando o funcionamento simplificado pode ser visto na Figura 10. O funcionamento real do sistema é projetado para evitar redundâncias, como o reenvio das configurações do modo automático quando o sistema já estiver ativo, entre outras possíveis duplicações de operação.

Para proporcionar uma adaptação mais personalizada, é permitido que o próprio usuário defina o limiar de ativação do sistema e a duração do seu funcionamento em segundos. Além disso, o usuário tem a opção de desativar o modo automático, evitando que ocorram ativações indesejadas do sistema, como durante a prática de exercícios, em que a frequência cardíaca pode se elevar mesmo não estando em uma situação caracterizada como estressante.

Figura 10: *Fluxograma do funcionamento simplificado do modo automático.*



Fonte: Do Autor.

3.4 Banco de dados

Para armazenamento em um banco de dados escolheu-se a ferramenta SQLite que é uma biblioteca, construída com a linguagem de programação C, de uso aberto que implementa um banco de dados relacional em um arquivo, eliminando a necessidade de instalação de um serviço SQL a parte.

De acordo com o site da ferramenta (SQLITE, 2023), a biblioteca é uma implementação de baixo consumo de memória com uma boa performance, além de possuir uma alta confiabilidade. Portanto, é indicado o uso dessa solução quando não há necessidade de suportar um grande número de usuários realizando acessos simultâneos e intensivos à base de dados, e quando não é necessário manter uma base de dados extensa. Concluiu-se que a ferramenta é adequada para esse projeto, pois faz-se necessário o uso de uma base de dados local sem a necessidade de acessos simultâneos frequentes.

Para estruturação do banco de dados, definiu-se três tabelas:

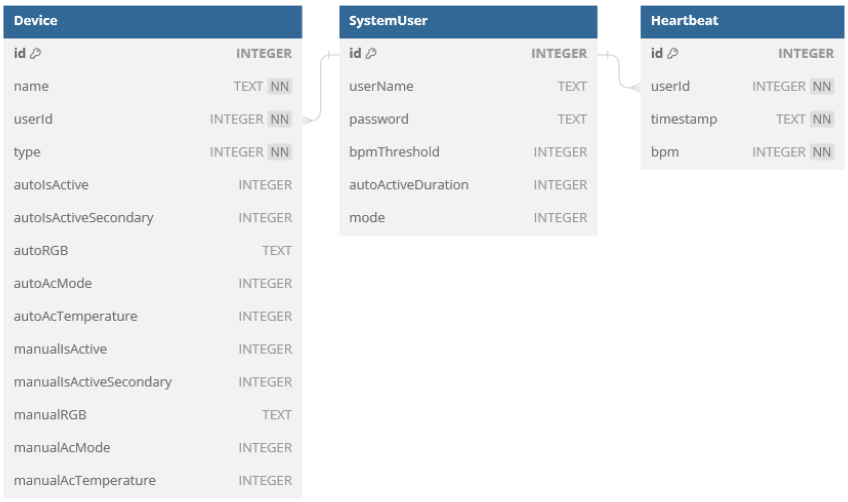
1. *SystemUser*: Destinada a armazenar informações relacionadas ao usuário do sistema, como as configurações do limiar dos BPM para ativação do modo automático e o tempo de permanência do sistema ativo.

2. *Heartbeat*: Projetada para armazenar os valores das leituras dos BPM juntamente com a data e hora que cada leitura ocorreu.

3. *Device*: Modelada para armazenar as informações básicas dos dispositivos, bem como configurações específicas do modo manual e do modo automático de cada dispositivo.

O diagrama entidade relacionamento completo pode ser visto na Figura 11. Essa modelagem foi elaborada considerando-se a possibilidade de expansão futura do sistema, facilitando a inclusão de novos dispositivos e usuários para uma eventual implementação de acessos simultâneos.

Figura 11: *Diagrama entidade relacionamento do banco de dados.*



Fonte: Do Autor.

3.5 Aplicação Web

As funções da aplicação web são fornecer uma interface visual através da qual o usuário possa interagir e efetuar modificações no sistema, implementar o funcionamento do sistema descrito na Seção 3.3 e a integração de todos os dispositivos.

Para implementação da aplicação web, optou-se pelo uso de um *framework*, que consiste no conjunto de ferramentas prontas para resolver problemas genéricos de uma aplicação web como autenticação, definição de rotas, disponibilização do sistema na rede, entre outros. Utilizou-se dois requisitos como base, o primeiro requisito foi a disponibilização gratuita do *framework*, o que já restringiu algumas opções. O segundo requisito, julgado importante para a escolha, foi a linguagem de programação base utilizada. Na Tabela 1 são apresentados alguns *frameworks* de implementação gratuita populares.

Tabela 1: *Frameworks de implementação gratuita populares.*

Nome do *framework*	Linguagem de programação base
React	Javascript
Django	Python
Laravel	PHP
ASP .NET	C#

Fonte: Clark (2022).

Por questões de experiência e afinidade com a linguagem de programação C#, escolheu-se o *framework* ASP .NET para o desenvolvimento da aplicação web. Esse *framework* possibilita o registro de serviços, que são classes em C# que encapsulam funções importantes para o sistema, o que poderia ser caracterizado como o *back-end* da aplicação. Registrar um serviço é facilitar o acesso de uma instância única dessa classe por toda a aplicação através do conceito de *Dependency Injection*. Para modularizar as

funcionalidades e facilitar a organização do projeto, nessa aplicação implementou-se 4 importantes serviços:

1. SQLiteService: Classe responsável por conter a conexão com o banco de dados SQLite através do pacote nuget *System.Data.SqlClient* (MICROSOFT, 2024), bem como conter métodos para criar, ler, atualizar e apagar (CRUD) dados das tabelas do banco de dados.

2. MqttService: Serviço responsável por gerenciar a ligação com o *broker* MQTT através do pacote nuget *MQTTnet* (MQTTNET, 2023), assim como um método para publicar uma mensagem no *broker* e outro para registrar uma *callback* que será chamada ao receber uma mensagem de um tópico inscrito.

3. HeartbeatService: Responsável por disparar uma *task* que realiza a requisição das leituras dos BPM a cada segundo, bem como conter a *callback* que registra as leituras no banco de dados.

4. SystemManagerService: Classe que dispara a *task* que realiza o monitoramento da ativação do sistema, ou seja, implementa a lógica do modo automático descrita na Seção 3.3.

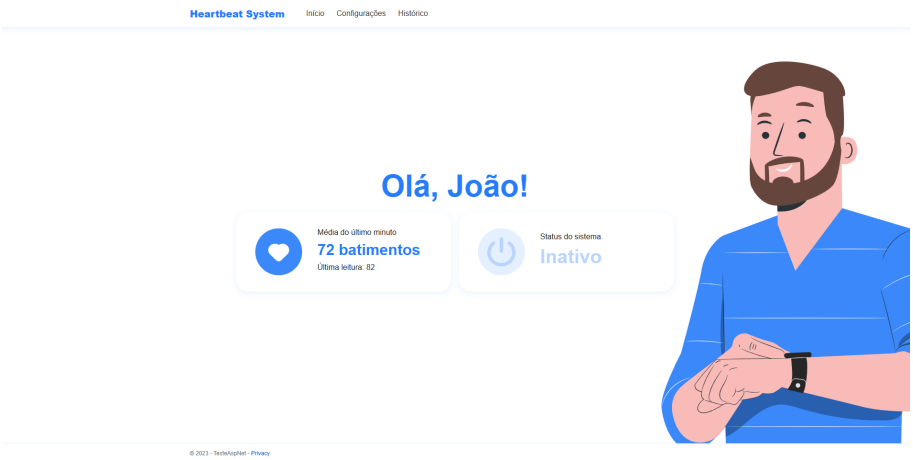
Com o desejo de alcançar a meta de criar uma aplicação intuitiva para os usuários, desenvolveu-se a interface visual em colaboração com Pedro Henrique Rettore, um colega do curso de Design Visual da Universidade Federal do Rio Grande do Sul. Dividiu-se essa interface em 3 páginas separadas: a página inicial exibe o estado atual do sistema, juntamente com informações da última leitura dos BPM e a média móvel do último minuto; a tela de configurações permite ao usuário definir variáveis gerais do sistema, ajustar as configurações do modo automático para cada dispositivo e enviar comandos aos dispositivos no modo manual; a última página possibilita ao usuário selecionar um período para gerar um gráfico, utilizando a biblioteca *ApexCharts* (DANGÅRDEN, 2023), das leituras dos BPM durante esse intervalo. Todas as páginas possuem um menu para navegação do aplicativo na parte superior.

Na Figura 12 está representada a tela inicial, contendo dois cartões ao centro da tela. O cartão da esquerda possui em azul a média móvel das leituras de BPM do último minuto e o valor da última leitura realizada, o cartão à direita possui o status do sistema, podendo ser ativo ou inativo.

Na Figura 13 está representada a tela de configurações do usuário no modo manual, possibilitando o envio de comandos assíncronos aos dispositivos. Na parte superior da tela está situada os campos para definição do modo atual do sistema, o limiar dos batimentos cardíacos para ativação do sistema e o tempo de duração do modo automático. Na parte inferior da tela estão situados três cartões, representando os três dispositivos do sistema. Para o dispositivo de tomadas, à esquerda, é possível ativar/desativar os canais dos relés individualmente. O dispositivo de comandos infravermelhos, situado no centro, possui três campos, um para ativar/desativar o condicionador de ar, um para definir a temperatura alvo e outro para definir o modo de funcionamento do condicionador de ar. À direita, está situado o cartão para controle do dispositivo de iluminação, possuindo um campo para ativar/desativar o dispositivo e outro para definir a cor da iluminação.

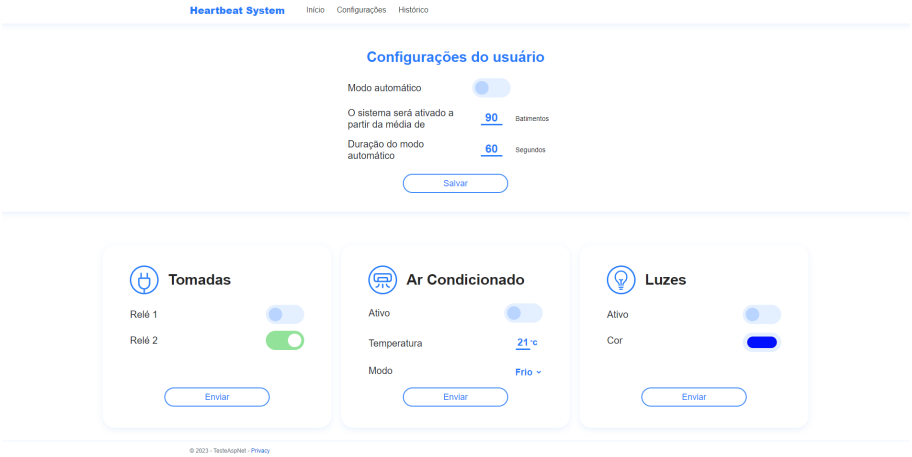
Na Figura 14 está representada a tela de configurações do usuário no modo automático, possibilitando a definição do estado que os dispositivos recebem em uma ativação do

Figura 12: *Interface visual da página inicial da aplicação web.*



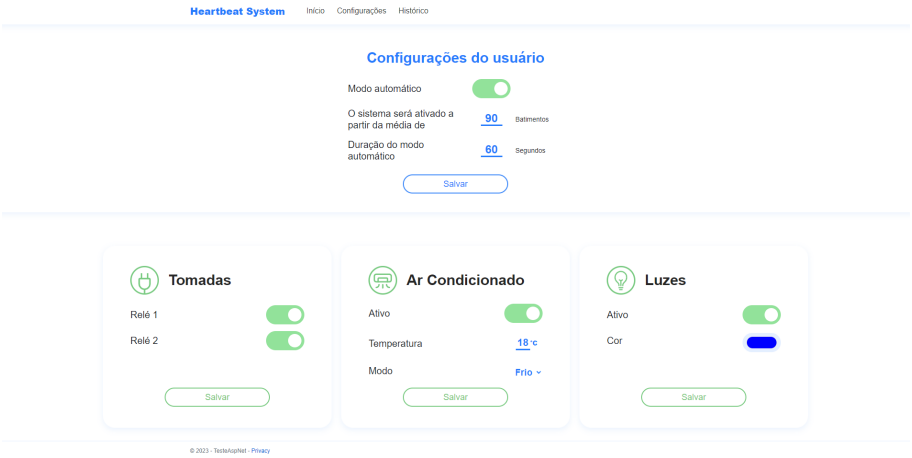
Fonte: Do Autor.

Figura 13: *Interface visual da página de configurações manuais da aplicação web.*



Fonte: Do Autor.

Figura 14: *Interface visual da página de configurações automáticas da aplicação web.*

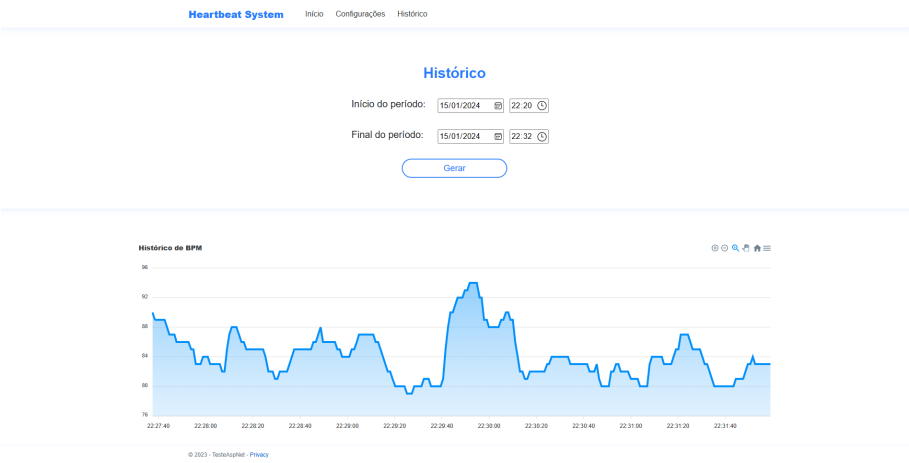


Fonte: Do Autor.

modo automático. Essa tela possui a mesma estrutura da tela anterior, apenas modifica-se a cor dos cartões dos dispositivos para melhor entendimento do usuário.

Na Figura 15 está representada a tela para geração de gráficos contendo o histórico de batimentos cardíacos. Na parte superior da tela existem campos para a definição do período que será realizada a consulta para obtenção do histórico dos dados de BPM. Na parte inferior da tela está situado o gráfico resultante da consulta do histórico de BPM.

Figura 15: *Interface visual da página de histórico de BPM da aplicação web.*



Fonte: Do Autor.

3.6 *Broker* MQTT

Para o *broker* MQTT foi escolhido o Eclipse Mosquitto que é um *broker* de mensagens MQTT de uso livre que implementa versões 5.0, 3.1.1 e 3.1 do protocolo MQTT. Segundo o site da ferramenta (ECLIPSE, 2023), ele provê um método leve para a implementação do gerenciamento de mensagens do modelo de publisher/subscriber. O Mosquitto também possui compatibilidade com muitas implementações de clientes MQTT e é de fácil instalação em diversos sistemas operacionais, como no Windows por exemplo. Uma razão adicional para essa escolha é a experiência que o autor desta proposta tem com a ferramenta, uma vez que já a utilizou anteriormente para implementar outros sistemas. Para configurar o *broker* localmente fez-se necessário definir um endereço IP estático à máquina principal e adicionar um arquivo de configuração com a propriedade "*allow_anonymous*"definida em verdadeiro, permitindo a conexão de clientes sem autenticação. Assim, basta inserir esse endereço IP nos clientes para estabelecer a conexão.

3.7 Microcontrolador

Um critério na escolha do microcontrolador que utilizou-se como base para os módulos desenvolvidos neste trabalho é a capacidade deste se conectar a uma rede através de uma conexão sem fio Wi-Fi. Como planejou-se um microcontrolador individual para cada módulo, uma conexão via cabo torna-se inviável pois comprometeria a mobilidade de instalação destes módulos.

Portanto, fez-se uma comparação entre quatro modelos de módulos de desenvolvimento de microcontroladores presentes no mercado que possuem capacidade de conexão via Wi-Fi. Na Tabela 2 há uma comparação entre o controlador utilizado, número de portas I/O e possibilidade de estabelecer uma comunicação Bluetooth. Na Tabela 3 há uma comparação entre o número de portas analógicas (ADC), a resolução do conversor analógico digital e o preço encontrado no mercado.

Tabela 2: *Comparação dos módulos entre controlador, portas I/O e Bluetooth.*

Módulo	Controlador	Portas I/O	Bluetooth
NodeMCU 8266	Esp8266	17	Não
ESP32 Devkit	Esp32	32	Sim
Arduino Nano IoT	Arm® Cortex®-M0 SAMD21	14	Sim
Raspberry Pi Pico W	RP2040	26	Não

Fonte: Espressif (2015), Espressif (2017), Arduino (2023) e Pi (2023).

Tabela 3: *Comparação dos módulos entre portas ADC, resolução do ADC e preço.*

Módulo	Portas ADC	Resolução do ADC	Preço
NodeMCU 8266	1	10 bit	R$ 32,90
ESP32 Devkit	6	12 bit	R$ 44,09
Arduino Nano IoT	8	12 bit	Importado US$ 25,50
Raspberry Pi Pico W	3	12 bit	R$ 129,90

Fonte: Espressif (2015), Espressif (2017), Arduino (2023) e Pi (2023).

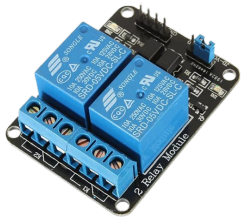
Todos os microcontroladores listados atendem os requisitos de portas necessárias para a implementação individual dos dispositivos discutidos na Seção 1.1, tornando o preço um fator primordial para tomar a decisão. Diante desses critérios, uma opção poderia ser o microcontrolador Esp8266, visto que é o microcontrolador mais econômico com base na pesquisa de mercado. Porém, visando a possibilidade de implementar projetos mais complexos no futuro, incluindo projetos que necessitam de uma conexão Bluetooth, e também buscando a possibilidade de reaproveitar algum microcontrolador, optou-se por investir um pouco mais e escolher o Esp32 como o microcontrolador base para o desenvolvimento dos dispositivos.

Para desenvolvimento do código de todos os dispositivos utilizou-se o *software* Arduino IDE. Definiu-se uma estrutura base para todos os dispositivos, contendo a conexão WiFi através de uma biblioteca nativa do Esp32, a conexão com o *broker* MQTT através da biblioteca PubSubClient (O'LEARY, 2020) e uma função *callback* para a interrupção gerada ao receber uma mensagem MQTT.

3.8 Dispositivo de Tomada Inteligente

Este dispositivo possui como base um módulo de relés de 2 canais, para fazer o chaveamento individual de duas tomadas inteligentes. Utilizou-se um módulo comum encontrado no mercado, representado na Figura 16.

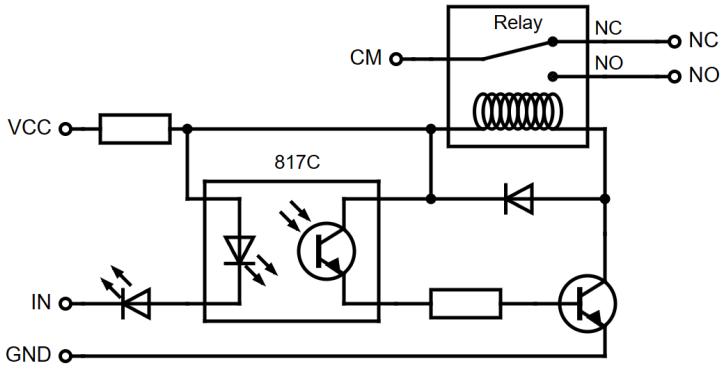
Figura 16: *Módulo relé de 2 canais.*



Fonte: Eletrogate (2024)

Esse módulo (ELETROGATE, 2024) é alimentado por uma tensão de 5 V e possui dois relés, cada um com uma tensão alternada máxima de 250 V e corrente máxima de 10 A, suficiente para as aplicações desse projeto. Ele pode ser controlado por uma tensão de 3,3 V, que é a tensão utilizada pelas portas digitais do microcontrolador Esp32, utilizando um acoplador óptico 817C (SHARP, 2024). Um diagrama esquemático do circuito representando o módulo com 1 canal pode ser visto na Figura 17, para o módulo com 2 canais o circuito é apenas duplicado.

Figura 17: *Diagrama esquemático do circuito do módulo de relé com 1 canal.*



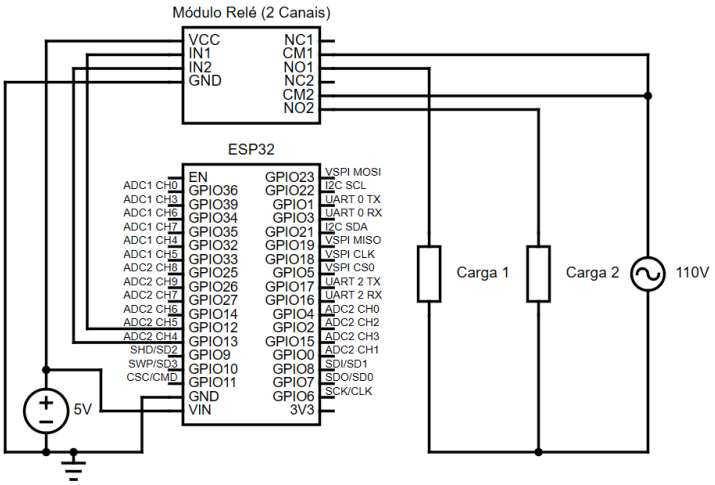
Fonte: Do Autor.

Esse circuito resulta em uma lógica de ativação do relé inversa, ou seja: ao chavear a entrada IN em um nível lógico baixo, ocorre uma diferença de tensão no LED do acoplador óptico, ativando o transistor óptico e, por consequência, o relé. Ao chavear em um nível lógico alto, a diferença de tensão no LED acoplador não é mais suficiente para ativar o transistor óptico, desativando o relé.

Para integrar o módulo dos relés com o microcontrolador Esp32, utilizou-se, respectivamente, as saídas digitais GPIO12 e GPIO13 para controlar o canal 1 e 2 do módulo. Para isso, basta escrever os valores digitais utilizando a lógica inversa da ativação dos relés. O diagrama esquemático do circuito completo, incluindo a alimentação da rede elétrica residencial, pode ser visto na Figura 18.

Definiu-se a comunicação do dispositivo através da inscrição do microcontrolador no tópico MQTT "esp32/relay", para receber mensagens publicadas pela aplicação principal nesse tópico. Modelou-se a mensagem com o seguinte formato: "channel<canal>-<estado>", substituindo o espaço reservado <canal> pelo número inteiro do canal chaveado e o espaço <estado> por "on"(para ligar) ou "off"(para desligar).

Figura 18: *Diagrama esquemático do circuito completo do dispositivo de tomada inteligente.*



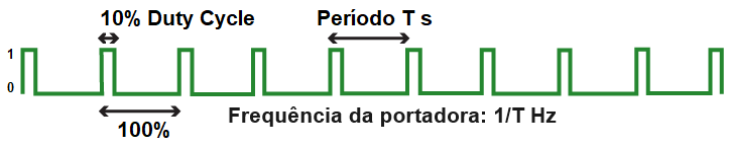
Fonte: Do Autor.

3.9 Dispositivo de Iluminação

A finalidade deste dispositivo é controlar a cor e a intensidade da iluminação do ambiente. Para isso, utilizou-se uma fita com segmentos em série de LEDs verdes, vermelhos e azuis (RGB). Como o foco reside em obter uma iluminação homogênea, o controle independente de cada LED na fita não se faz necessário. Portanto uma fita LED 5050 RGB (amazon, 2024) atende as necessidades da aplicação. Algumas características do modelo adquirido: necessita uma alimentação de 12 V. Possui 5 metros de comprimento. Possui 4 pinos, 1 para entrada da alimentação e os outros 3 para saída dos canais (vermelho, verde e azul) em paralelo, possibilitando o controle independente de cada canal.

Através da mistura de diferentes intensidades de cada canal do RGB, que são as cores primárias para a luz, é possível gerar qualquer cor do espectro visível (w3schools, 2024). O controle da intensidade de um LED pode ser realizado através de uma modulação por largura de pulso (PWM). Um sinal PWM consiste em alternar um sinal entre um intervalo ativo e um intervalo inativo em uma base de tempo fixo. A proporção entre a duração do intervalo ativo e inativo é denominada de *Duty Cycle*, e a frequência da portadora é o inverso da soma desses intervalos. Recomenda-se o uso de uma frequência da portadora alta em relação à frequência da visão humana, evitando assim a percepção do chaveamento pelo olho humano. Um exemplo de sinal PWM com *Duty Cycle* de 10% pode ser visto na Figura 19. Portanto, para controlar a intensidade de um LED, altera-se o valor do *Duty Cycle* do sinal PWM, o qual um valor de 100% corresponde à intensidade máxima do LED, enquanto um valor de 0% resulta no desligamento do LED.

Figura 19: *Exemplo de sinal PWM com Duty Cycle de 10%.*

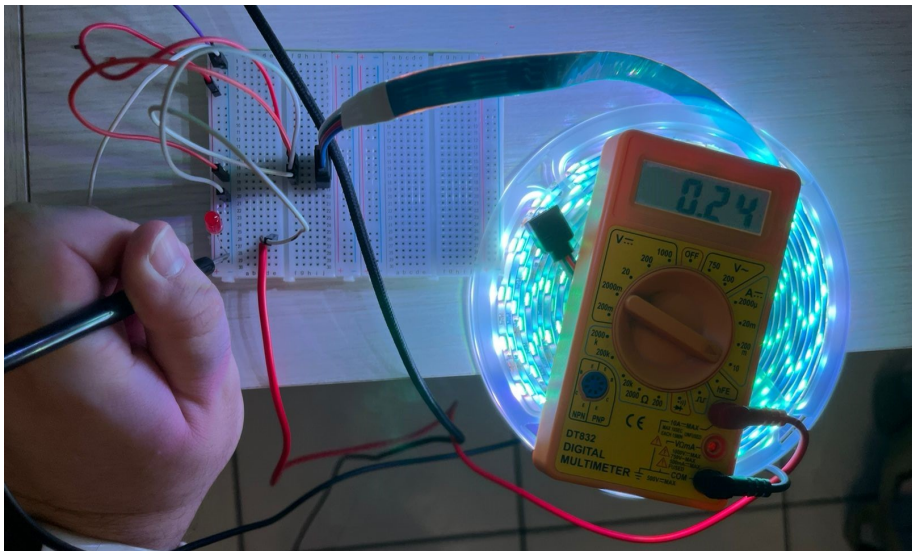


Fonte: Adaptado de GeeksForGeeks (2023).

Para gerar três sinais PWM distintos, utilizou-se as portas GPIO13, GPIO12 e

GPIO14 do Esp32, correspondendo, respectivamente, à intensidade da cor vermelha, verde e azul. Definiu-se arbitrariamente uma frequência da portadora de 5 kHz, considerada suficiente para ser imperceptível ao olho humano. Como o Esp32 trabalha em uma tensão de $3,3$ V, fez-se necessário o uso de um transistor de junção bipolar NPN para controlar o sinal da alimentação de 12 V de cada canal. Para dimensionar o transistor, mediu-se a corrente de um dos canais de LED em brilho máximo, que resultou em $0,24$ A, como pode ser visto na Figura 20.

Figura 20: *Medição da corrente de um dos canais RGB em brilho máximo.*



Fonte: Do Autor.

Esta corrente representa a corrente de coletor máxima que será atingida nesta aplicação. Portanto escolheu-se o transistor BC337 (SEMICONDUCTOR, 2005), por possuir uma corrente de coletor máxima de 800 mA e pela disponibilidade em uma loja próxima à residência do autor. Para o cálculo da resistência de polarização do transistor, primeiramente calculou-se a corrente de base necessária para polarização, utilizando a equação de ganho das correntes no transistor representada na Equação 1.

$$I_b = \frac{I_c}{hFE} \tag{1}$$

Ao consultar o *datasheet* do BC337, encontrou-se $hFE \approx 70$ para $I_c \approx 240$ mA, substituindo os valores, obtém-se $I_b \approx 3,43$ mA. Após, utilizou-se a equação da malha formada pela saída do Esp32, resistência da base do transistor e a tensão V_{BE} do transistor, representada na Equação 2. E a lei de Ohm, representada na Equação 3.

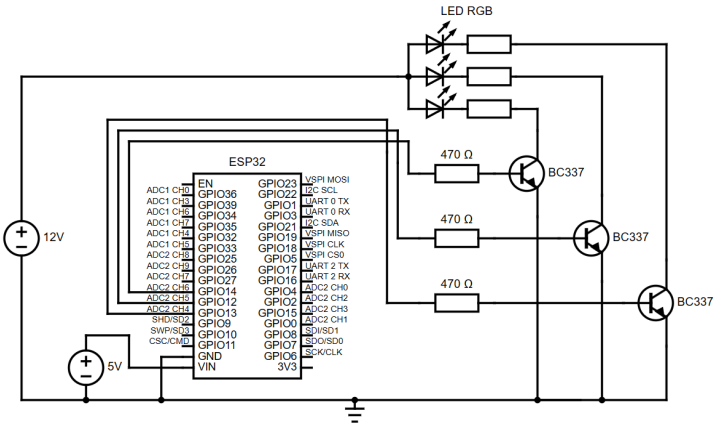
$$V_{Esp32} = V_R + V_{BE} \tag{2}$$

$$V_R = R \cdot I_R \tag{3}$$

Substituindo 3 em 2, e utilizando $I_R = I_b$, $V_{Esp32} = 3,3$ V e o valor retirado do *datasheet* $V_{BE} = 1,2$ V, obtém-se $R \approx 612,5\ \Omega$. Para utilizar um valor de resistência comercial com alguma folga, definiu-se $R = 470\ \Omega$. Com os componentes dimensionados, montou-se um diagrama esquemático do circuito do dispositivo de iluminação, que pode

ser visto na Figura 21. Para simplificar a visualização, representou-se a fita LED RGB por um LED em série com um resistor para cada canal RGB.

Figura 21: *Diagrama esquemático do circuito do dispositivo de iluminação.*



Fonte: Do Autor.

Para definir o *Duty Cycle* de cada canal que resulta em uma cor específica, utilizou-se a convenção digital de cores RGB, a qual define um valor entre 0 e 255 para cada cor, sendo 0 a ausência e 255 a intensidade máxima da cor. Essa convenção está implementada em um componente da interface visual discutida na Seção 3.5.

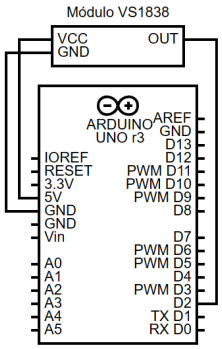
A comunicação do dispositivo realizou-se através da inscrição do microcontrolador no tópico MQTT "esp32/led-strip", para receber mensagens publicadas pela aplicação principal nesse tópico. Definiu-se que a mensagem para ligar a fita LED em uma cor específica teria o seguinte formato: "led-on-r<R>g<G>b<B>&", substituindo o espaço reservado <R> pelo valor entre 0 a 255 do canal vermelho, o espaço <G> pelo valor do canal verde e o espaço <B> pelo valor do canal azul. Para desligar a fita LED utilizou-se a seguinte mensagem "led-off".

3.10 Dispositivo de Comandos Infravermelhos

Esse dispositivo tem como objetivo o acionamento de um condicionador de ar através de comandos infravermelhos. Para isso, primeiramente, realizou-se testes utilizando o controle remoto do condicionador de ar para compreender seu funcionamento. Concluiu-se que o controle remoto envia o estado final completo ao condicionador de ar a cada botão pressionado. Por exemplo, ao pressionar o botão de incrementar a temperatura em 1 ºC, o comando enviado não contém uma mensagem de incremento de temperatura, mas uma mensagem contendo a temperatura que será atingida junto das outras variáveis que caracterizam o estado de funcionamento do condicionador de ar. Portanto, cada comando novo recebido é independente do estado em que estava o condicionador de ar antes do recebimento desse comando.

Com o teste anterior, definiu-se que para controlar o condicionador de ar seria necessário obter um comando para cada estado atingível. Para obter esses comandos a partir do controle remoto, utilizou-se um Arduino Uno (ARDUINO, 2024) e um módulo receptor de infravermelho baseado no componente VS1838 (ALLDATASHEET, 2024), o diagrama esquemático desse circuito pode ser visto na Figura 22.

Figura 22: *Diagrama esquemático do circuito de obtenção dos comandos infravermelhos.*



Fonte: Do Autor.

O Arduino Uno é alimentado pela porta USB do computador e utiliza-se as saídas de 5 V e GND do Arduino para alimentar o módulo receptor de infravermelho. Ao apertar um botão do controle remoto, o receptor captura esse sinal e replica a lógica através de um sinal de tensão em sua saída que está ligada na entrada D2 do Arduino. Esse sinal é codificado pelo Arduino através da estrutura discutida na Subseção 2.5.3 e enviado ao computador via serial USB. Para mapear os estados disponíveis no condicionador de ar, foram obtidos 42 sinais codificados: 13 sinais representando as temperaturas de 18 ºC até 30 ºC no modo frio; 15 sinais representando as temperaturas de 16 ºC até 30 ºC no modo quente; 13 sinais representando as temperaturas de 18 ºC até 30 ºC no modo automático; e 1 sinal para desligar o condicionador de ar. Dois desses sinais estão representados na Listagem 2 e Listagem 3.

Listagem 2: *Codificação do sinal representando 18 ºC no modo frio do condicionador de ar, obtido de um controle remoto fabricado pela LG.*

```
const uint16_t rawSignal[] = { 3224, 9868, 488, 1588, 516, 548, 488, 524, 516,
   544, 464, 1584, 484, 544, 488, 544, 488, 556, 484, 552, 488, 544, 484, 548,
    484, 544, 488, 544, 512, 520, 488, 544, 488, 548, 492, 540, 488, 544, 488,
    1600, 464, 1596, 464, 1588, 508, 524, 492, 544, 488, 1604, 468, 1584, 508,
    1572, 516, 504, 512, 540, 488 }
```

Fonte: Do Autor.

Listagem 3: *Codificação do sinal de desligar do condicionador de ar, obtido de um controle remoto fabricado pela LG.*

```
const uint16_t rawSignal[] = { 3140, 9868, 536, 1584, 488, 524, 516, 544, 468,
   564, 484, 1580, 484, 528, 516, 540, 488, 544, 488, 1572, 488, 1576, 488,
   544, 472, 560, 464, 568, 488, 540, 492, 540, 492, 540, 488, 544, 488, 544,
   488, 552, 488, 544, 480, 548, 488, 1576, 488, 544, 488, 1576, 488, 524, 516,
    540, 488, 544, 488, 1592, 488}
```

Fonte: Do Autor.

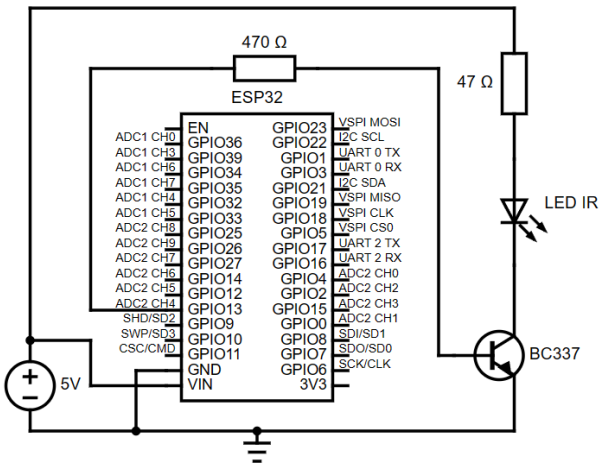
Após a obtenção de todos os sinais, planejou-se a estrutura do dispositivo para sua emissão. O comando seria emitido através de um LED IR (ROITHNER, 2011) e

controlado pela porta GPIO13 do Esp32 através da utilização de uma biblioteca chamada IRremote (JOACHIMSMEYER, 2023). Ao consultar o *datasheet* do LED IR, encontrou-se uma corrente de funcionamento de 100 mA, o que resultaria em uma sobrecarga da saída do Esp32. Portanto, implementou-se uma estratégia parecida com a utilizada para controle de um dos canais RGB discutidos na Seção 3.9, na qual empregou-se um transistor de junção bipolar NPN para chaveamento do LED. O circuito com o LED IR resultaria em uma $I_c = 100$ mA, abaixo da corrente de coletor da Seção 3.9, portanto empregou-se os mesmos componentes, o transistor BC337 (SEMICONDUCTOR, 2005) e a resistência de polarização de 470 Ω. Restando assim, o calculo da resistência para limitação da corrente no LED. Para isso, utilizou-se a equação da malha formada pela alimentação de 5 V, a resistência de limitação, o LED IR e o transistor representada na Equação 4; e a lei de Ohm, representada na Equação 3.

$$V_{CC} = V_R + V_{LED} + V_{CE} \tag{4}$$

Substituindo 3 em 4, e utilizando $I_R = I_c$, $V_{CC} = 5$ V, o valor retirado do *datasheet* $V_{LED} = 1,2$ V e desprezando a queda de tensão V_{CE}, obtém-se $R \approx 38$ Ω. Para utilizar um valor de resistência comercial, definiu-se $R = 47$ Ω. Após o dimensionamento dos componentes, montou-se um diagrama esquemático do dispositivo de comandos infravermelhos, representado na Figura 23.

Figura 23: *Diagrama esquemático do circuito do dispositivo de comandos infravermelhos.*



Fonte: Do Autor.

A comunicação do dispositivo realizou-se através da inscrição do microcontrolador no tópico MQTT "esp32/ir", para receber mensagens publicadas pela aplicação principal nesse tópico. Definiu-se que a mensagem para ligar o condicionador de ar em um estado específico teria o seguinte formato: "air-on-m<modo>t<temperatura>&", substituindo o espaço reservado <modo> pela representação em inteiro de um dos 3 modos de operação e o espaço <temperatura> pelo valor em graus Celsius da temperatura desejada. Para desligar o condicionador de ar utilizou-se a seguinte mensagem "air-off".

4 RESULTADOS

Neste Capítulo são apresentados testes individuais, testes de integração do sistema e um teste de uso do sistema ao decorrer de um dia.

Para facilitar os testes individuais dos dispositivos, implementou-se uma aplicação de console em C# para conectar-se como cliente no *broker* MQTT e publicar mensagens em tópicos personalizados manualmente.

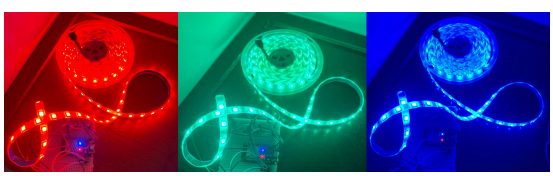
4.1 Teste do dispositivo de tomada inteligente

Para este teste, ligou-se um abajur no canal 1 e um difusor de aromas no canal 2. Utilizando a aplicação de testes enviou-se no tópico "esp32/relay"as mensagens "channel1-on"em seguida de "channel1-off"e obteve-se uma resposta do abajur ligando e desligando. Após, enviou-se as mensagens "channel2-on"em seguida de "channel2-off"e obteve-se uma resposta do difusor de aromas ligando e desligando. Confirmando assim, o funcionamento do dispositivo.

4.2 Teste do dispositivo de iluminação

Neste teste, utilizou-se a aplicação de testes para publicar mensagens no tópico "esp32/led-strip". Alguns exemplos de mensagens utilizadas: "led-on-r255g255b255&"(branco), "led-on-r255g0b0&"(vermelho), "led-on-r0g255b0&"(verde), "led-on-r0g0b255&"(azul), "led-on-r204g0b153&"(roxo), "led-on-r255g255b0&"(amarelo) e "led-on-r255g102b0&"(laranja). Para cores compostas de apenas um canal (vermelhos, verde e azul) a fita LED apresentou um bom desempenho, com as cores bem definidas, como visualizado na Figura 24. Cores como branco e roxo, a fita LED apresentou um desempenho aceitável, permitindo a definição das cores, podendo ser visualizado na Figura 25. E para as cores amarelo e laranja, a fita LED não apresentou um bom desempenho, não possibilitando a distinção das cores, como visto na Figura 26. Ao final do teste, enviou-se a mensagem "led-off", fazendo com que a fita LED desligasse.

Figura 24: *Resultado da fita LED nas cores contendo apenas um canal.*

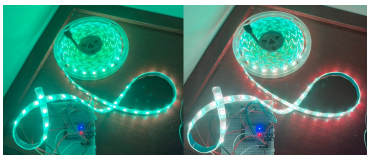


Fonte: Do Autor.

Figura 25: *Resultado da fita LED nas cores branca e roxa.*

Fonte: Do Autor.

Figura 26: *Resultado da fita LED nas cores amarela e laranja.*



Fonte: Do Autor.

4.3 Teste do dispositivo de comandos infravermelhos

Para execução deste teste, utilizou-se a aplicação de testes para publicar mensagens no tópico "esp32/ir". Testou-se o envio de mensagens contendo as 41 combinações possíveis entre modo de funcionamento e temperatura, obtendo uma resposta do condicionador de ar correspondente a cada uma dessas combinações. Ao final, enviou-se a mensagem "air-off"e o condicionador de ar desligou.

4.4 Teste de integração dos sistemas

Para execução deste teste, realizou-se os seguintes passos: primeiro, conectou-se todos os dispositivos à rede elétrica. Em seguida, executou-se o *broker* MQTT, a aplicação web e o *script* Python de obtenção da frequência cardíaca da pulseira. Após isso, configurou-se o aplicativo Pulsoid no celular e ajustou-se a pulseira no autor no modo de exercícios. Através da aplicação web ajustou-se o limiar de ativação do sistema em 90 BPM, a duração do modo automático em 60 segundos e habilitou-se o modo automático. Enviou-se comandos manuais para desativar todos os dispositivos e ajustou-se o modo automático conforme a Figura 27.

Figura 27: *Configurações do modo automático utilizadas para o teste de integração.*

Fonte: Do Autor.

No início do teste, obteve-se uma média do último minuto da frequência cardíaca de 80 BPM. Após a realização de exercícios físicos pelo autor, obteve-se uma média acima de

90 BPM, cruzando o limiar definido para o teste e ativando o sistema, por consequência, causando o envio das configurações automáticas, confirmando a lógica de funcionamento. Logo após esta confirmação, o autor manteve-se em repouso, fazendo a média retornar aos 80 BPM do início do teste. Após aguardar a duração do modo automático definida para o teste, o sistema enviou aos dispositivos o estado em que eles se encontravam antes da ativação do modo automático, resultando na desativação de todos. Os dispositivos ativados e desativados podem ser visualizados na Figura 28.

Figura 28: *Dispositivos ativados à esquerda e dispositivos desativados à direita.*

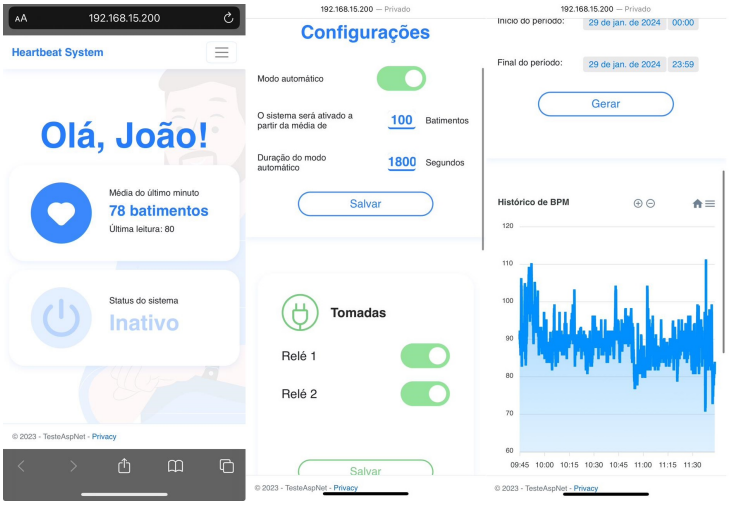
Fonte: Do Autor.

4.5 Publicação da aplicação web na rede local

Para confirmar o funcionamento da aplicação web em dispositivos além do computador principal, publicou-se a aplicação no serviço de informações da internet (IIS) do *Windows* utilizando o gerenciador do IIS. Definiu-se a porta TCP/IP de acesso como 8090 e habilitou-se regras de entrada no *Firewall* do *Windows* para possibilitar o acesso à essa porta por clientes da rede local. Através do navegador de internet *Safari* de um aparelho celular, realizou-se o acesso à aplicação web, o resultado da interface visual pode ser visualizado na Figura 29.

Implementou-se através do acesso da aplicação web via celular a mesma rotina de teste descrita na Seção 4.4, obtendo os mesmos resultados e confirmando o funcionamento em dispositivos alternativos. Realizou-se também o acesso da aplicação web através de dispositivos não convencionais, como uma Smart TV por exemplo.

Figura 29: *Resultado da interface visual em um aparelho celular.*



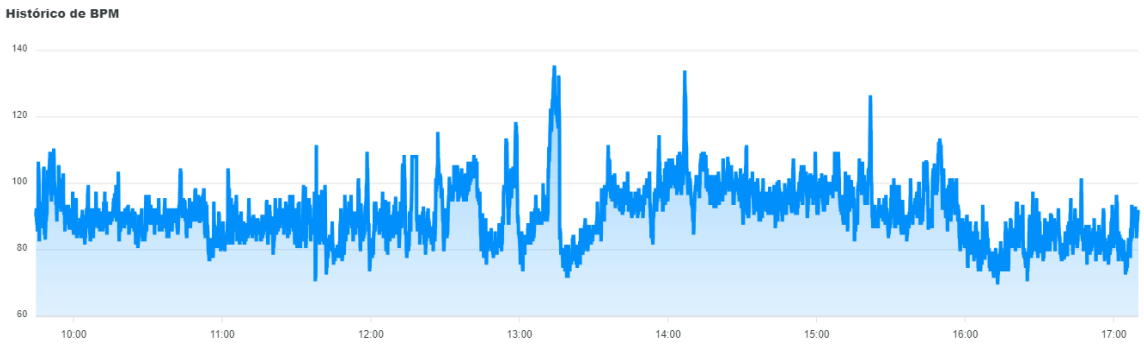
Fonte: Do Autor.

4.6 Avaliação do uso ao decorrer de um dia

Com o intuito de avaliar o resultado final propôs-se a utilização da pulseira pelo autor ao decorrer de um dia. Instalou-se todos os dispositivos em um quarto da residência do autor. Ajustou-se o limiar de ativação do sistema em 100 BPM e a permanência do modo ativo em 1800 segundos. No dispositivo de tomada conectou-se um difusor de aromas para tornar o ambiente mais agradável.

Na Figura 30 pode ser visualizado o histórico dos batimentos cardíacos obtidos ao decorrer da avaliação.

Figura 30: *Histórico do BPM entre às 9:50 e 17:10 do dia da avaliação.*



Fonte: Do Autor.

As leituras ficaram situadas entre 100 BPM e 80 BPM na maior parte do teste, próximas à 90 BPM. No decorrer do teste houveram picos na leitura acima do limiar de 100 BPM que foram corretamente diluídos através do cálculo da média móvel.

Quanto às ativações do sistema, houveram seis ativações. Todas as situações reais de estresse por conta de eventos relacionados ao emprego do autor foram corretamente identificadas, contabilizando três ativações. As outras três ativações foram consideradas indesejadas. As ativações indesejadas ocorreram por conta de alguma tarefa não relacionada com estresse, mas relacionadas à algum esforço físico, como descer escadas por exemplo. Demonstrando a necessidade do monitoramento de mais sinais vitais para caracterização do

estado do indivíduo, como por exemplo: pressão arterial, frequência respiratória, saturação do oxigênio e temperatura corporal.

Todas as ativações permaneceram por 1800 segundos, ou seja, a frequência cardíaca ficou abaixo do limiar nas segundas checagens. O ambiente gerado pelas ativações do modo automático julgou-se como agradável, cumprindo o seu propósito.

5 CONCLUSÕES

A leitura dos batimentos cardíacos através de uma pulseira comercial foi realizada e armazenada para monitoramento da saúde do usuário. Através dessas leituras foi possível identificar um estado de estresse no usuário e implementar uma lógica para um funcionamento automático do sistema. Foram projetados e implementados: um dispositivo que atua como tomada inteligente, um dispositivo para controle da cor e intensidade da iluminação de um ambiente e um dispositivo para acionamento de um condicionador de ar através de comandos infravermelhos. Através de um teste de avaliação, o ambiente gerado pelos dispositivos criados foi julgado como relaxante, indicando a criação de um ambiente restaurador. Através de um teste de integração foi confirmado o correto funcionamento das interações entre os dispositivos por meio do protocolo de comunicação. Foi implementada uma aplicação com uma interface gráfica de simples manuseio, permitindo o usuário interagir com o sistema e calibra-lo de acordo com suas preferências para uma melhor experiência. Foi confirmado o funcionamento dessa aplicação em mais de um tipo de dispositivo que possua acesso à um navegador de internet. Portanto, concluiu-se que os objetivos do trabalho foram alcançados.

Através da avaliação do sistema realizada na Seção 4.6 concluiu-se que apenas a utilização da frequência cardíaca como dado de entrada para caracterização do estado emocional do usuário mostrou-se insuficiente, pois houveram ativações do sistema que foram causadas por realização de algum exercício físico e não em decorrência de uma crise de estresse, sendo caracterizadas como falsos positivos. Para resolver esse problema e realizar a expansão do conceito empregado neste trabalho, aconselha-se a utilização de mais sinais vitais para dados de entrada, como:

- Pressão arterial
- Saturação do oxigênio
- Frequência respiratória
- Temperatura corporal

Para trabalhos futuros, existe a possibilidade de implementação de um controle de usuário através de autenticação, para que o mesmo servidor pudesse ser utilizado por mais usuários simultâneos. Porém, isso implicaria em realizar uma melhoria da segurança desse projeto, por se tratar da manipulação de dados sensíveis inerentes ao usuário.

REFERÊNCIAS

ALLDATASHEET. *VS1838 Datasheet*. [S.l.], 2024. Disponível em: <https://html.alldatasheet.com/html-pdf/1132466/ETC2/VS1838/110/1/VS1838.html>. Acesso em: 22 jan. 2024.

AMAZON. *Echo Dot 5ª geração*. [S.l.], 2023a. Disponível em: <https://www.amazon.com.br/Echo-Dot-5%C2%AA-gera%C3%A7%C3%A3o-Cor-Preta/dp/B09B8VGCR8?pf_rd_r=2A1NQM1Z3YXPFRPNSTB9&pf_rd_t=PageFrameworkApplication&pf_rd_i=19877613011&pf_rd_p=4e91d5ba-eb24-4d35-9805-0746690c0f60&pf_rd_s=merchandised-search-4&ref=dlx_19877_gd_dcl_tlt_3_792c6bbc_dt_mese4_60>. Acesso em: 19 jul. 2023.

AMAZON. *Fita LED RGB 5050*. [S.l.], 2024. Disponível em: <https://www.amazon.com.br/LED-metros-Fonte-Controle-Remoto/dp/B08WFYRQVG>. Acesso em: 22 jan. 2024.

AMAZON. *Qual é a diferença entre JSON e XML?* [S.l.], 2023b. Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-json-xml/>. Acesso em: 19 jul. 2023.

ARDUINO. *Arduino Nano 33 Iot Datasheet*. [S.l.], 2023. Disponível em: <https://docs.arduino.cc/resources/datasheets/ABX00027-datasheet.pdf>. Acesso em: 9 ago. 2023.

ARDUINO. *Arduino UNO R3 Datasheet*. [S.l.], 2024. Disponível em: <https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>. Acesso em: 22 jan. 2024.

BAUER, M. E. Estresse. *Ciência hoje*, v. 30, n. 179, p. 20–25, 2002.

BEVYWISE. *The eFon Technology's Smart Home security system trusts Bevywise MQTT Solution*. [S.l.], 2020. Disponível em: <https://www.bevywise.com/blog/iot-home-security-system-mqtt-broker/>. Acesso em: 9 ago. 2023.

BRASIL. *Definição - Transtornos de Ansiedade no adulto*. [S.l.], 2023. Disponível em: <https://linhasdecuidado.saude.gov.br/portal/ansiedade/definicao/>. Acesso em: 7 ago. 2023.

CLARK, M. Os 10 principais frameworks do lado do servidor. *Back4App*, 2022. Disponível em: <https://blog.back4app.com/pt/top-10-frameworks-para-desenvolvimento-web/>. Acesso em: 29 jul. 2023.

DANGÅRDEN, J. *Blazor-ApexCharts*. [S.l.], 2023. Disponível em: <https://github.com/apexcharts/Blazor-ApexCharts>. Acesso em: 22 jan. 2024.

DICTIONARY.COM. *Internet of Things*. [S.l.: s.n.], 2023. Disponível em: <https://www.dictionary.com/browse/internet-of-things>. Acesso em: 9 ago. 2023.

DREHER, H. S. *Automação residencial utilizando o protocolo MQTT controlado por app Android/iOS*. 2019. Trabalho de conclusão de curso – Universidade Tecnológica Federal do Paraná, Campo Mourão. Disponível em: <https://repositorio.utfpr.edu.br/jspui/bitstream/1/24369/1/automacaoresidencialprotocolomqtt.pdf>. Acesso em: 10 ago. 2023.

ECLIPSE. *Mosquitto version 2.0.15*. [S.l.], 2023. Disponível em: <https://mosquitto.org/>. Acesso em: 9 ago. 2023.

EKAZA. *Cortina Inteligente*. [S.l.], 2023. Disponível em: <https://www.ekaza.com.br/cortina-inteligente/>. Acesso em: 19 jul. 2023.

ELETROGATE. *Modulo Relé 2 Canais*. [S.l.], 2024. Disponível em: <https://www.eletrogate.com/modulo-rele-2-canais-5v>. Acesso em: 22 jan. 2024.

EMTECO. *Motor 6 N.m. PLUG*. [S.l.], 2023. Disponível em: <https://emtecomotores.com.br/motor-6nm/>. Acesso em: 19 jul. 2023.

ESPRESSIF. *ESP-WROOM-32 Datasheet*. [S.l.], 2017. Disponível em: <https://blogmasterwalkershop.com.br/arquivos/datasheet/Datasheet%20ESP-WROOM-32.pdf>. Acesso em: 9 ago. 2023.

ESPRESSIF. *ESP8266EX Datasheet*. [S.l.], 2015. Disponível em: <https://components101.com/sites/default/files/component_datasheet/ESP8266-NodeMCU-Datasheet.pdf>. Acesso em: 9 ago. 2023.

GEEKSFORGEEKS. *Duty Cycle*. [S.l.], 2023. Disponível em: <https://www.geeksforgeeks.org/duty-cycle/>. Acesso em: 22 jan. 2024.

GEONAV. *Central de Controle Infravermelho Universal Wi-Fi*. [S.l.], 2023. Disponível em: <https://www.geonav.com.br/produtos/central-de-controle-infravermelho-universal-wi-fi/>. Acesso em: 19 jul. 2023.

GILLIS, A. S. Internet of things (IoT). *TechTarget*, 2023. Disponível em: <https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT>. Acesso em: 22 jan. 2024.

IBM. *Telemetry use case: Home patient monitoring*. [S.l.], 2023. Disponível em: <https://www.ibm.com/docs/en/ibm-mq/8.0?topic=cases-telemetry-use-case-home-patient-monitoring>. Acesso em: 9 ago. 2023.

INTELBRAS. *Lampada LED Smart Wi-Fi EWS 410*. [S.l.], 2023. Disponível em: <https://www.intelbras.com/pt-br/lampada-led-smart-wi-fi-ews-410>. Acesso em: 19 jul. 2023.

JOACHIMSMEYER, A. *Arduino IRremote*. [S.l.], 2023. Disponível em: <https://github.com/Arduino-IRremote/Arduino-IRremote>. Acesso em: 26 jul. 2023.

KJELLGREN, A.; BUHRKALL, H. A comparison of the restorative effect of a natural environment with that of a simulated natural environment. *Journal of Environmental Psychology*, v. 30, p. 464–472, 2010.

LG. *LG DUAL Inverter VOICE 12.000 Quente/Frio 220V*. [S.l.], 2023. Disponível em: <https://www.lg.com/br/ar-condicionado-residencial/lg-s4-w12ja31b>. Acesso em: 19 jul. 2023.

LIGHT, R. *Eclipse Paho™ MQTT Python Client*. [S.l.], 2021. Disponível em: <https://github.com/eclipse/paho.mqtt.python>. Acesso em: 22 jan. 2024.

LOPEZ, N. G. *IP Security*. 2003. Seminário da Disciplina de Redes de Computadores I – Universidade Federal do Rio de Janeiro, Rio de Janeiro. Disponível em: <https://www.gta.ufrj.br/grad/03_1/ip-security/paginas/introducao.html>. Acesso em: 8 ago. 2023.

MICROSOFT. *System.Data.SqlClient*. [S.l.], 2024. Disponível em: <https://www.nuget.org/packages/System.Data.SqlClient/4.8.6?_src=template>. Acesso em: 22 jan. 2024.

MINGUILLON, J. et al. Blue lighting accelerates post-stress relaxation: Results of a preliminary study. *PLoS ONE*, 2017. Disponível em: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0186399>. Acesso em: 8 ago. 2023.

MOZILLA. *Uma visão geral do HTTP*. [S.l.], 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Acesso em: 19 jul. 2023.

MQTTNET. *MQTTnet*. [S.l.], 2023. Disponível em: <https://github.com/dotnet/MQTTnet>. Acesso em: 22 jan. 2024.

O'LEARY, N. *Arduino Client for MQTT*. [S.l.], 2020. Disponível em: <https://github.com/knolleary/pubsubclient/tree/v2.8>. Acesso em: 26 jul. 2023.

OLIVEIRA, C. M. DE. *Sistema para Mapeamento de Códigos Infravermelho e Criação de Controle Remotos Virtuais*. 2015. Trabalho de conclusão de curso – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, Rio de Janeiro. Disponível em: <https://eic.cefet-rj.br/portal/wp-content/uploads/Monografia-001.pdf>. Acesso em: 8 ago. 2023.

OMS. *Depression and Other Common Mental Disorders - Global Health Estimates*. [S.l.], 2017. Disponível em: <https://apps.who.int/iris/bitstream/handle/10665/254610/WHO-MSD-MER-2017.2-eng.pdf>. Acesso em: 29 jul. 2023.

OMS. *Mental Health and COVID-19: Early evidence of the pandemic's impact*. [S.l.], 2022. Disponível em: <https://www.who.int/publications/i/item/WHO-2019-nCoV-Sci_Brief-Mental_health-2022.1>. Acesso em: 29 jul. 2023.

PI, R. *Raspberry Pi Pico W Datasheet*. [S.l.], 2023. Disponível em: <https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>. Acesso em: 9 ago. 2023.

PPA. *Tomada 10A Smart ON*. [S.l.], 2023. Disponível em: <https://www.ppa.com.br/brasil/products/ppa-on/produto/tomada-10a-smart-on>. Acesso em: 19 jul. 2023.

PRIMEBRAS. *Fechadura Digital Eletronica Biometrica Primebras Wi-fi Rio*. [S.l.], 2023. Disponível em: <https://www.primebras.com.br/MLB-2005929743-fechadura-digital-eletronica-biometrica-primebras-wi-fi-rio-_JM>. Acesso em: 19 jul. 2023.

RATCLIFFE, E.; GATERSLEBEN, B.; SOWDEN, P. T. Bird sounds and their contributions to perceived attention restoration and stress recovery. *Journal of Environmental Psychology*, v. 36, p. 221–228, 2013.

REITZ, K. *Requests*. [S.l.], 2023. Disponível em: <https://github.com/psf/requests>. Acesso em: 22 jan. 2024.

RIBEIRO, E. Alexa, Google e Siri: como funcionam os assistentes virtuais. *Estadão*, 2024. Disponível em: <https://www.estadao.com.br/recomenda/tech/eletronicos/alexa-google-assistente-siri-assistentes-virtuais/>. Acesso em: 22 jan. 2024.

ROITHNER. *BC337 Datasheet*. [S.l.], 2011. Disponível em: <https://html.alldatasheet.com/html-pdf/709991/ROITHNER/LED1200/384/1/LED1200.html>. Acesso em: 22 jan. 2024.

SEMICONDUCTOR, O. *BC337 Datasheet*. [S.l.], 2005. Disponível em: <https://html.alldatasheet.com/html-pdf/171972/ONSEMI/BC337/654/3/BC337.html>. Acesso em: 22 jan. 2024.

SHARP. *817C Datasheet*. [S.l.], 2024. Disponível em: <https://html.alldatasheet.com/html-pdf/43371/SHARP/PC817/253/2/PC817.html>. Acesso em: 22 jan. 2024.

SONSIN, J. A diferença entre estresse e ansiedade. *Telavita*, 2018. Disponível em: <https://www.telavita.com.br/blog/diferenca-de-estresse-e-ansiedade/>. Acesso em: 22 jan. 2024.

SOUZA, I. DE. Entenda o que é Rest API e a importância dele para o site da sua empresa. *RockContent*, 2020. Disponível em: <https://rockcontent.com/br/blog/rest-api/>. Acesso em: 19 jul. 2023.

SQLITE. *SQLite version 3.42.0*. [S.l.], 2023. Disponível em: <https://www.sqlite.org/>. Acesso em: 9 ago. 2023.

SWITCHBOT. *SwitchBot Curtain Rod2*. [S.l.], 2023. Disponível em: <https://us.switchbot.com/pages/switchbot-curtain-rod2>. Acesso em: 19 jul. 2023.

ULRICH, R. S. View Through a Window May Influence Recovery from Surgery. *Science*, v. 224, p. 420–421, 1984. Disponível em: <https://www.science.org/doi/10.1126/science.6143402>. Acesso em: 22 jan. 2024.

W3SCHOOLS. *Colors RGB and RGBA*. [S.l.], 2024. Disponível em: <https://www.w3schools.com/colors/colors_rgb.asp>. Acesso em: 22 jan. 2024.

XIAOMI. *Pulseira Inteligente Mi Smart Band 5*. [S.l.], 2023. Disponível em: <https://www.mibrasil.com.br/pulseira-inteligente-xiaomi-mi-band-5-x539-p1841>. Acesso em: 19 jul. 2023.

ZEMISMART. *Zemismart M515EGWT*. [S.l.], 2023. Disponível em: <https://www.zemismart.com/products/m515egwt>. Acesso em: 19 jul. 2023.

ZOLETT, D.; RAMIREZ, A. R. G. Desenvolvimento de uma Interface de Monitoração Remota para o Sistema Robótico OBIX, Integrando o Protocolo MQTT e oROS. In: ANAIS do 11 Computer on the Beach. Baln. Camboriú, SC: [s.n.], 2020. P. 405–412.

Apêndices

Apêndice A - OUTROS DISPOSITIVOS COMERCIAIS REVISADOS

Neste Apêndice são apresentados outros dispositivos comercias revisados para a elaboração deste trabalho.

A.1 Condicionadores de ar

No presente momento, existem modelos de condicionadores de ar que podem se conectar à uma rede Wi-Fi e possuem integração com a internet, a partir de aplicativos de celular fornecidos pelos fabricantes é possível enviar comandos de liga/desliga, mudar o modo de funcionamento, ajustar a temperatura, e até programar horários para envio de comandos ao condicionador de ar. Um modelo comercial fabricado pela LG pode ser visto na Figura 31.

Figura 31: *Condicionador de ar fabricado pela LG.*

Fonte: LG (2023).

Além do aplicativo de celular, é possível enviar comandos para o condicionador de ar através de uma REST API bem documentada e fornecida pela LG chamada de "LG Thinq Connect API". Porém esta API utiliza um modo de autenticação através de um "*Service Id*" e uma "*Service Key*" que são fornecidos apenas para desenvolvedores que possuem uma parceria com a LG.

A.2 Assistente virtual

Uma assistente virtual é um dispositivo que, através de inteligência artificial, se comunica com o usuário de maneira intuitiva utilizando comandos de voz. É possível interagir com estes assistentes através de aparelhos celulares, alto-falantes inteligentes, e alguns outros dispositivos. Alguns exemplos de serviços oferecidos: definir alarmes e lembretes, realizar pesquisas na internet, reproduzir músicas (ribeiro, 2024). Alguns assistentes conseguem se comunicar com aparelhos domésticos inteligentes, como os descritos nesta seção.

Um exemplo comercial é a assistente virtual Alexa, desenvolvida pela Amazon, que pode ser vista na Figura 32. Essa assistente possui integração com outros aparelhos através das *Alexa Skills*, aplicativos desenvolvidos especificamente para a Alexa, para configurar funcionalidades extras através de comandos de voz. Alguns destes aplicativos são desenvolvidos pelas fornecedoras de dispositivos inteligentes para realizar o controle de seus produtos através da Alexa. Um exemplo de *Alexa Skill* é o "LG Thinq", aplicativo desenvolvido pela marca LG para controle de seus produtos, como condicionadores de ar, refrigeradores, ventiladores de teto, máquinas de lavar, entre outros.

Figura 32: *Alto falante com Alexa produzido pela Amazon.*

Fonte: Amazon (2023a).

A.3 Cortina inteligente

Os aparelhos comerciais para automatização de cortinas seguem o mesmo padrão dos outros aparelhos discutidos nesta seção, comunicam-se através de uma rede Wi-Fi e recebem comandos através de um aplicativo de celular, no qual podem ser criadas rotinas programadas para utilização do produto.

A seguir, serão discutidos alguns exemplos de modelos de aparelhos utilizados para realizar a automatização da abertura e fechamento de cortinas.

- Neste modelo, o próprio trilho da cortina se move a partir de uma correia dentada controlada por uma engrenagem acoplada a um motor elétrico (ekaza, 2023).

- Neste modelo é acoplado um motor inteligente no interior de uma persiana de tubo. É possível calibrar uma faixa de operação deste motor, definindo uma posição como ponto máximo ao enrolar a persiana e uma posição como ponto mínimo para desenrolar a persiana (emteco, 2023).

- Para persianas enroláveis manualmente por uma corda, é possível acoplar um motor inteligente para atuar diretamente nesta corda. O acoplamento é realizado utilizando uma polia dentada (zemismart, 2023).

- Para automatizar cortinas presas em uma haste por argolas, existe um robô que pode ser acoplado diretamente na haste entre a primeira e a segunda argola da cortina. Este robô se movimenta linearmente na haste, empurrando a cortina para realizar a abertura e o fechamento (switchbot, 2023).

A utilização de uma persiana inteligente pode melhorar as condições do ambiente residencial, uma vez que a persiana poderia fechar automaticamente reduzindo o ruído sonoro externo, como os provenientes de carros, por exemplo. Além disso, a persiana pode ser acionada em conjunto com a lâmpada inteligente garantindo um ambiente com baixa iluminação natural para que a luz produzida pela lâmpada tenha um impacto mais eficaz.

A.4 Fechadura inteligente

A fechadura inteligente combina a funcionalidade de uma fechadura normal com a possibilidade de se conectar a uma rede Wi-Fi para se comunicar através da internet. A partir de um aplicativo de celular é possível monitorar remotamente o estado da fechadura, assim como acionar sua abertura ou fechamento. Através do aplicativo também podem ser enviadas notificações de abertura da fechadura, alertando possíveis invasões. A abertura da fechadura também pode ser realizada através de tags eletrônicas, chaves convencionais, leitura biométrica e códigos numéricos (temporários ou não).

Outras funcionalidades de uma fechadura inteligente incluem um modo não perturbe, que impede a abertura da fechadura durante a ativação do modo, e o travamento automático que aumenta a segurança evitando que a fechadura permaneça aberta de maneira indesejável. A fechadura produzida pela fabricante Primebras pode ser vista na Figura 33.

Figura 33: *Modelo de fechadura inteligente produzido pela Primebras.*

Fonte: Primebras (2023).

Apêndice B - MONTAGEM DOS DISPOSITIVOS

Neste Apêndice são apresentados os passos para a montagem física dos dispositivos criados neste trabalho.

Para a montagem dos dispositivos, adquiriu-se componentes comuns à todos eles, estes componentes estão listados à seguir:

- Três kits de desenvolvimento Esp32.
- Três fontes de alimentação bivolt com saída de 5 V e 2 A, para alimentação externa dos microcontroladores.
- Diversas *protoboards* para facilitação da implementação dos circuitos.
- Kits de cabos *jumper* macho e fêmea para realização das conexões.

B.1 Montagem do dispositivo de tomada inteligente

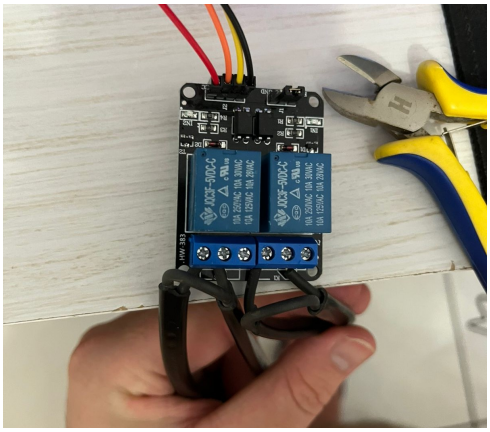
Os componentes adquiridos para realização do dispositivo de tomadas inteligentes estão listados à seguir:

- Dois cabos de força bipolar
- Dois plugs de tomada fêmea
- Módulo de relés com 2 canais

Para a montagem do dispositivo, realizou-se a confecção de dois cabos de extensão a partir dos dois cabos de força e dos plugs de tomada fêmea. Após, desencapou-se uma pequena seção de cabo aproximadamente no centro da extensão confeccionada. Posteriormente, cortou-se um dos cabos revelados e conectou-se suas duas extremidades no módulo de relés, uma extremidade no contato comum do relé e outra no contato normalmente aberto. Esse procedimento foi repetido para a outra extensão confeccionada. O resultado da montagem no módulo de relés pode ser visto na Figura 34.

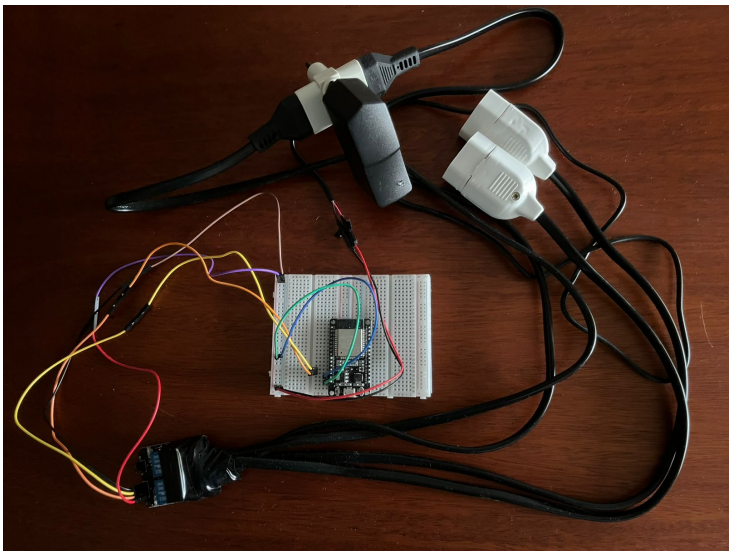
Em seguida, para evitar acidentes, isolou-se a área do módulo de relés que apresentava riscos de exposição do usuário à alta tensão quando o dispositivo estivesse conectado à rede elétrica. A seguir, realizou-se a programação do Esp32 com a lógica do dispositivo através do computador principal e um cabo USB, e implementou-se na *protoboard* o circuito ilustrado na Figura 18. O resultado pode ser visualizado na Figura 35.

Figura 34: *Conexão das extensões confeccionadas no módulo de relés de 2 canais.*



Fonte: Do Autor.

Figura 35: *Dispositivo de tomadas inteligentes montado na protoboard.*



Fonte: Do Autor.

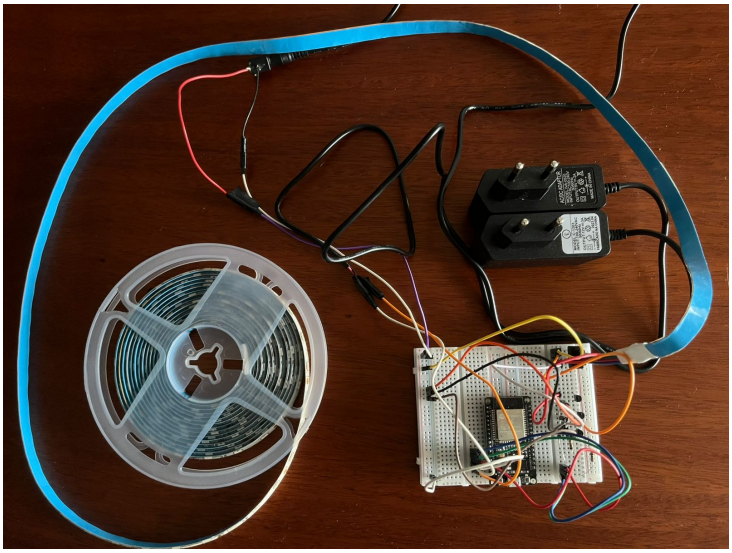
B.2 Montagem do dispositivo de iluminação

Os componentes adquiridos para realização do dispositivo de tomadas inteligentes estão listados à seguir:

- Kit contendo uma fita LED RGB 5050 de 5 m de comprimento e uma fonte de alimentação bivolt com saída de 12 V e 3 A.
- Três transistores BC337.
- Três resistores de 470 Ω.

Para a montagem, realizou-se a programação do Esp32 com a lógica do dispositivo através do computador principal e um cabo USB. Em seguida, implementou-se o circuito na *protoboard* seguindo o diagrama esquemático da Figura 21. O resultado da montagem pode ser visualizado na Figura 36.

Figura 36: *Dispositivo de iluminação montado na protoboard.*



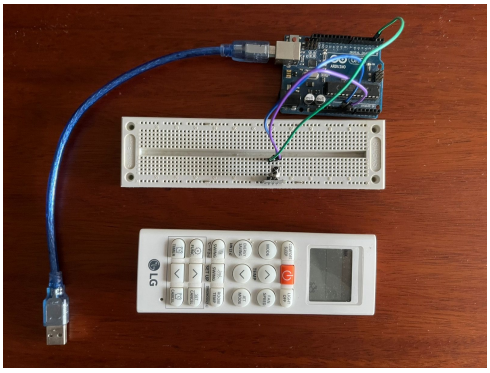
Fonte: Do Autor.

B.3 Montagem do dispositivo de comandos infravermelhos

Os componentes adquiridos para realização do dispositivo de tomadas inteligentes estão listados à seguir:

- Kit contendo um módulo receptor e um LED emissor de luz infravermelha.
- Um transistor BC337.
- Dois resistores, um de 470 Ω e outro de 47 Ω.

Figura 37: *Dispositivo de obtenção dos sinais infravermelhos montado na protoboard.*

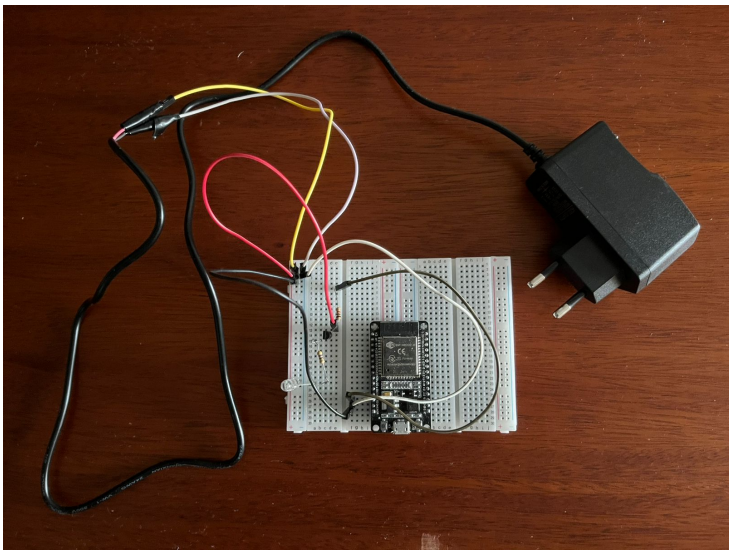


Fonte: Do Autor.

Utilizando um Arduino Uno e um módulo receptor de luz infravermelha, montou-se na *protoboard* o circuito de obtenção dos sinais infravermelhos conforme o diagrama esquemático representado na Figura 22. O resultado pode ser visualizado na Figura 37.

Após a obtenção de todos os sinais necessários emitidos pelo controle remoto do condicionador de ar LG, realizou-se a programação do Esp32 com a lógica do dispositivo através do computador principal e um cabo USB. Em seguida, implementou-se o circuito na *protoboard* seguindo o diagrama esquemático da Figura 23. O resultado da montagem pode ser visualizado na Figura 38.

Figura 38: *Dispositivo de comandos infravermelhos montado na protoboard.*



Fonte: Do Autor.

Apêndice C - CÓDIGOS FONTE DOS *SOFTWARES* DO PROJETO

Neste Apêndice são apresentados os códigos fontes dos *softwares* utilizados neste trabalho.

C.1 Código fonte do dispositivo de tomada inteligente

```
// Definicao do pino dos reles
#define RELAY_CHANNEL_1 12
#define RELAY_CHANNEL_2 13

#include <IRremote.h>
#include <WiFi.h>
#include <PubSubClient.h>

// Configuracoes da Wifi
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";

// Endereco IP do Broker MQTT:
const char* mqtt_server = "192.168.15.200";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

// LED Pin
const int internalLedPin = 2;

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  pinMode(internalLedPin, OUTPUT);
```

```
  // Configuracao dos pinos dos reles
  Serial.print(F("Relay channel 1 at pin "));
  Serial.println(RELAY_CHANNEL_1);
  pinMode(RELAY_CHANNEL_1, OUTPUT);
  digitalWrite(RELAY_CHANNEL_1, HIGH);
  Serial.print(F("Relay channel 2 at pin "));
  Serial.println(RELAY_CHANNEL_2);
  pinMode(RELAY_CHANNEL_2, OUTPUT);
  digitalWrite(RELAY_CHANNEL_2, HIGH);
}

// Conexao com a Wifi
void setup_wifi() {
  delay(10);

  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

// Callback de mensagem recebida via MQTT
void callback(char* topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageString;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageString += (char)message[i];
  }
  Serial.println();


  // Tratamento das mensagens
  if (String(topic) == "esp32/relay") {
    if(messageString == "channel1-on"){
      Serial.println("Turning channel 1 relay on");
      digitalWrite(RELAY_CHANNEL_1, LOW);
```

```
    }
    else if(messageString == "channel1-off"){
      Serial.println("Turning channel 1 relay off");
      digitalWrite(RELAY_CHANNEL_1, HIGH);
    }
    else if(messageString == "channel2-on"){
      Serial.println("Turning channel 2 relay on");
      digitalWrite(RELAY_CHANNEL_2, LOW);
    }
    else if(messageString == "channel2-off"){
      Serial.println("Turning channel 2 relay off");
      digitalWrite(RELAY_CHANNEL_2, HIGH);
    }
  }
}

void reconnect() {
  // Loop ate conseguir conectar no broker MQTT
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Tentativa de conexao com o broker
    if (client.connect("ESP32-Relay")) {
      Serial.println("connected");
      // Inscricao no topico MQTT
      client.subscribe("esp32/relay");
      digitalWrite(internalLedPin, HIGH);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      delay(5000);
    }
  }
}
void loop() {
  if (!client.connected()) {
    digitalWrite(internalLedPin, LOW);
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 5000) {
    lastMsg = now;
  }
}
```

C.2 Código fonte do dispositivo de iluminação

```
// Definicao dos pinos dos canais
#define RED_PIN 13
#define RED_CHANNEL 0
#define GREEN_PIN 12
#define GREEN_CHANNEL 1
#define BLUE_PIN 14
#define BLUE_CHANNEL 2
// Frequencia de modulacao do sinal 5kHz
#define FREQUENCY 5000
// ResoluÃ§Ã£o em bits do sinal PWM - 8: 2^8 = 256 valores
#define RESOLUTION  8

#include <IRremote.h>
#include <WiFi.h>
#include <PubSubClient.h>

// Configuracoes da Wifi
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";

// Endereco IP do Broker MQTT:
const char* mqtt_server = "192.168.15.200";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

// LED Pin
const int internalLedPin = 2;

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  pinMode(internalLedPin, OUTPUT);

  // ConfiguraÃ§Ã£o dos pinos RGB em PWM
  Serial.print(F("Red PWM channel at pin "));
  Serial.println(RED_PIN);
  ledcSetup(RED_CHANNEL, FREQUENCY, RESOLUTION);
  ledcAttachPin(RED_PIN, RED_CHANNEL);

  Serial.print(F("Green PWM channel at pin "));
  Serial.println(GREEN_PIN);
  ledcSetup(GREEN_CHANNEL, FREQUENCY, RESOLUTION);
  ledcAttachPin(GREEN_PIN, GREEN_CHANNEL);
```

```
  Serial.print(F("Blue PWM channel at pin "));
  Serial.println(BLUE_PIN);
  ledcSetup(BLUE_CHANNEL, FREQUENCY, RESOLUTION);
  ledcAttachPin(BLUE_PIN, BLUE_CHANNEL);

  ledcWrite(RED_CHANNEL, 0);
  ledcWrite(GREEN_CHANNEL, 0);
  ledcWrite(BLUE_CHANNEL, 0);
}

// Conexao com a Wifi
void setup_wifi() {
  delay(10);

  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

// Callback de mensagem recebida via MQTT
void callback(char* topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageString;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageString += (char)message[i];
  }
  Serial.println();

  // Tratamento das mensagens
  if (String(topic) == "esp32/led-strip") {
    if(messageString == "led-off"){
      Serial.println("Turning LED strip off");
      ledcWrite(RED_CHANNEL, 0);
      ledcWrite(GREEN_CHANNEL, 0);
```

```
      ledcWrite(BLUE_CHANNEL, 0);
    }
    else if(messageString.indexOf("led-on") > -1){ // Checa se a mensagem
   contem a substring "led-on"
      // Decomposicao da mensagem
      int pos1 = messageString.indexOf('r');
      int pos2 = messageString.indexOf('g');
      int pos3 = messageString.indexOf('b');
      int pos4 = messageString.indexOf('&');

      String redString = messageString.substring(pos1+1, pos2);
      String greenString = messageString.substring(pos2+1, pos3);
      String blueString = messageString.substring(pos3+1, pos4);

      Serial.print("Red value: ");
      Serial.println(redString);
      Serial.print("Green value: ");
      Serial.println(greenString);
      Serial.print("Blue value: ");
      Serial.println(blueString);

      // Definicao do PWM
      ledcWrite(RED_CHANNEL, redString.toInt());
      ledcWrite(GREEN_CHANNEL, greenString.toInt());
      ledcWrite(BLUE_CHANNEL, blueString.toInt());
    }
  }
}

void reconnect() {
  // Loop ate conseguir conectar no broker MQTT
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Tentativa de conexao com o broker
    if (client.connect("ESP32-LED-Strip")) {
      Serial.println("connected");
      // Inscricao no topico MQTT
      client.subscribe("esp32/led-strip");
      digitalWrite(internalLedPin, HIGH);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      delay(5000);
    }
  }
}
void loop() {
  if (!client.connected()) {
    digitalWrite(internalLedPin, LOW);
```

```
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 5000) {
    lastMsg = now;
  }
}
```

C.3 Código fonte do dispositivo de captura dos sinais infravermelhos

```
#define LEDPIN 13
// Tamanho maximo do buffer
#define maxLen 650

volatile unsigned int irBuffer[maxLen]; // Buffer que armazena os valores dos
   intervalos
volatile unsigned int x = 0; // Ponteiro do buffer

void setup() {
  Serial.begin(115200);
  attachInterrupt(0, rxIR_Interrupt_Handler, CHANGE);// Setup da interrupcao
   para captura do sinal IR
}

void loop() {

  Serial.println(F("Press the button on the remote now - once only"));
  delay(5000); // Pausa de 5 segundos para captura do sinal
  if (x) { // Se um sinal foi capturado
    digitalWrite(LEDPIN, HIGH);//Indicador visual de que o sinal foi recebido e
     esta sendo enviado via serial
    Serial.println();
    Serial.print(F("Raw: ("));
    Serial.print((x - 1));
    Serial.print(F(") "));
    detachInterrupt(0);// Desabilita as interrupcoes e as capturas ate o final
   da escrita serial
    for (int i = 1; i < x; i++) { // Envia os intervalos da codificacao do
   sinal via serial
      Serial.print(irBuffer[i] - irBuffer[i - 1]);
      Serial.print(F(", "));
    }
    x = 0;
    Serial.println();
    Serial.println();
    digitalWrite(LEDPIN, LOW);// Indicador visual
```

```
    attachInterrupt(0, rxIR_Interrupt_Handler, CHANGE);// Habilita a
    interrupcao novamente para capturar novo sinal
  }

}

void rxIR_Interrupt_Handler() {
  if (x > maxLen) return; // Ignora se o buffer esta cheio
  irBuffer[x++] = micros(); // Alimenta o buffer com os valores de intervalos
    capturados das transicoes do sinal

}
```

C.4 Código fonte do dispositivo de comandos infravermelhos

```
// Definicao do pino de envio do sinal infravermelho
#define IR_SEND_PIN 13

#include <IRremote.h>
#include <WiFi.h>
#include <PubSubClient.h>

// Configuracoes da Wifi
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";

// Endereco IP do Broker MQTT:
const char* mqtt_server = "192.168.15.200";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

// LED Pin
const int internalLedPin = 2;

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  pinMode(internalLedPin, OUTPUT);

  // ConfiguraÃ§Ã£o do IR
  Serial.println(F("START " __FILE__ " from " __DATE__ "\r\nUsing library
    version " VERSION_IRREMOTE));
```

```
  Serial.print(F("Send IR signals at pin "));
  Serial.println(IR_SEND_PIN);
  IrSender.begin(); // Utiliza o IR_SEND_PIN
}

// Conexao com a Wifi
void setup_wifi() {
  delay(10);

  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

// Callback de mensagem recebida via MQTT
void callback(char* topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageString;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageString += (char)message[i];
  }
  Serial.println();

  // Tratamento das mensagens
  if (String(topic) == "esp32/ir") {
    if(messageString == "air-off") {
      Serial.print("Sending AC off signal");
      // Definicao dos intervalos do comando infravermelho
      const uint16_t rawData[] = { 3140, 9868, 536, 1584, 488, 524, 516, 544,
    468, 564, 484, 1580, 484, 528, 516, 540, 488, 544, 488, 1572, 488, 1576,
    488, 544, 472, 560, 464, 568, 488, 540, 492, 540, 492, 540, 488, 544, 488,
    544, 488, 552, 488, 544, 480, 548, 488, 1576, 488, 544, 488, 1576, 488, 524,
     516, 540, 488, 544, 488, 1592, 488};
      // Envio do comando pela GPIO13 na frequencida de modulacao NEC - 38kHz
      IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]), NEC_KHZ);
```

```
  delay(1000); // Delay para garantir que dois sinais seguidos nao sejam
interpretados como um longo sinal
 }
 else if(messageString.indexOf("air-on") > -1){ // Checa se a mensagem
contem a substring "air-on"
  // Decomposicao da mensagem
  int pos1 = messageString.indexOf('m');
  int pos2 = messageString.indexOf('t');
  int pos3 = messageString.indexOf('&');

  String modeString = messageString.substring(pos1+1, pos2);
  String temperatureString = messageString.substring(pos2+1, pos3);

  Serial.println("Sending AC on signal");
  Serial.print("Mode: ");
  Serial.println(modeString);
  Serial.print("Temperature: ");
  Serial.println(temperatureString);

  switch(modeString.toInt()) {
    case 0:
      Serial.println("Frio");
      if(temperatureString == "18") {
          const uint16_t rawData[] = { 3224, 9868, 488, 1588, 516, 548,
488, 524, 516, 544, 464, 1584, 484, 544, 488, 544, 488, 556, 484, 552, 488,
 544, 484, 548, 484, 544, 488, 544, 512, 520, 488, 544, 488, 548, 492, 540,
 488, 544, 488, 1600, 464, 1596, 464, 1588, 508, 524, 492, 544, 488, 1604,
468, 1584, 508, 1572, 516, 504, 512, 540, 488 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "19") {
          const uint16_t rawData[] = { 3248, 9828, 488, 1604, 492, 568,
488, 544, 460, 548, 508, 1556, 516, 524, 492, 540, 492, 540, 488, 544, 488,
 544, 488, 540, 492, 548, 492, 540, 516, 516, 488, 540, 492, 544, 488, 540,
 516, 1548, 488, 544, 512, 516, 516, 1556, 492, 548, 492, 548, 512, 1552,
488, 1564, 508, 1580, 488, 540, 516, 1548, 496 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "20") {
          const uint16_t rawData[] = { 3248, 9820, 488, 1616, 512, 552,
484, 552, 468, 564, 488, 1572, 492, 540, 488, 544, 488, 544, 468, 564, 488,
 544, 488, 544, 488, 524, 516, 540, 488, 544, 468, 564, 488, 544, 488, 540,
 488, 1580, 484, 524, 516, 1584, 464, 1596, 488, 552, 488, 524, 516, 1564,
516, 1576, 484, 1584, 488, 1580, 488, 552, 468 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "21") {
          const uint16_t rawData[] = { 3196, 9876, 504, 1596, 492, 548,
516, 504, 512, 540, 488, 1572, 516, 516, 488, 544, 488, 540, 516, 516, 492,
 540, 492, 540, 492, 548, 516, 516, 488, 544, 488, 540, 492, 540, 492, 540,
 492, 1572, 488, 1580, 492, 540, 492, 1580, 488, 548, 516, 524, 492, 1580,
```

```
492, 1572, 512, 1540, 508, 1572, 492, 1580, 488 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "22") {
          const uint16_t rawData[] = { 3268, 9816, 488, 1596, 512, 544,
468, 564, 488, 544, 488, 1576, 488, 520, 516, 544, 488, 544, 464, 568, 468,
 564, 488, 540, 488, 544, 488, 544, 464, 568, 488, 544, 464, 548, 516, 544,
 460, 1600, 464, 1608, 488, 1592, 484, 1568, 512, 524, 516, 544, 484, 1576,
 488, 544, 464, 544, 512, 524, 492, 568, 484 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "23") {
          const uint16_t rawData[] = { 3224, 9844, 488, 1612, 496, 544,
488, 548, 516, 516, 516, 1552, 488, 540, 488, 540, 516, 516, 492, 540, 492,
 540, 492, 540, 492, 528, 512, 540, 488, 540, 516, 516, 492, 540, 492, 1572,
 488, 540, 516, 508, 532, 516, 492, 1580, 492, 528, 532, 516, 516, 1548,
504, 516, 536, 516, 516, 512, 516, 1548, 492 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "24") {
          const uint16_t rawData[] = { 3196, 9892, 488, 1596, 516, 508,
532, 152, 884, 520, 488, 1572, 492, 540, 492, 540, 492, 536, 492, 540, 492,
 544, 512, 516, 516, 524, 492, 540, 492, 536, 492, 540, 516, 516, 492, 1572,
 492, 528, 512, 548, 488, 1572, 492, 1580, 492, 548, 512, 516, 492, 1580,
492, 540, 492, 540, 488, 1580, 492, 540, 512 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "25") {
          const uint16_t rawData[] = { 3252, 9812, 484, 1592, 516, 528,
516, 528, 488, 568, 464, 1612, 464, 548, 516, 520, 488, 544, 512, 520, 508,
 520, 492, 540, 512, 524, 516, 516, 508, 576, 464, 540, 512, 528, 488, 1600,
 464, 544, 488, 1576, 488, 540, 488, 1600, 464, 532, 508, 544, 512, 1552,
484, 544, 488, 540, 500, 1596, 492, 1560, 484 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "26") {
          const uint16_t rawData[] = { 3352, 9840, 544, 1584, 484, 508,
532, 544, 464, 568, 488, 1584, 464, 544, 484, 544, 512, 524, 492, 556, 516,
 524, 516, 532, 492, 540, 512, 520, 488, 544, 488, 544, 488, 540, 496, 1576,
 488, 540, 532, 1552, 488, 1572, 488, 1588, 496, 536, 492, 540, 516, 1556,
488, 540, 532, 1556, 516, 504, 532, 520, 488 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "27") {
          const uint16_t rawData[] = { 3328, 9868, 516, 1584, 488, 532,
532, 516, 496, 536, 488, 1592, 488, 548, 492, 548, 492, 536, 536, 516, 488,
 152, 880, 540, 516, 504, 512, 540, 488, 540, 516, 520, 488, 532, 532, 1552,
 484, 1564, 532, 524, 492, 528, 512, 1580, 488, 544, 488, 540, 516, 1548,
492, 528, 536, 1548, 488, 532, 532, 1536, 512 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
```

```
      } else if(temperatureString == "28") {
          const uint16_t rawData[] = { 3352, 9840, 488, 1588, 492, 548,
512, 520, 488, 540, 492, 1576, 488, 548, 492, 540, 488, 540, 492, 540, 492,
 540, 492, 540, 492, 528, 536, 516, 516, 516, 488, 540, 492, 540, 492, 1572,
 492, 1576, 492, 540, 492, 1580, 492, 1572, 488, 544, 512, 516, 492, 1572,
492, 548, 488, 1572, 492, 1588, 492, 528, 536 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "29") {
          const uint16_t rawData[] = { 3332, 9864, 520, 1608, 464, 528,
512, 540, 512, 520, 488, 1592, 484, 552, 488, 552, 488, 536, 512, 540, 492,
 540, 516, 516, 492, 528, 508, 544, 512, 516, 516, 516, 492, 528, 536, 1548,
 488, 1560, 536, 1564, 492, 528, 532, 1556, 492, 528, 536, 516, 516, 1548,
508, 520, 516, 1556, 512, 1548, 516, 1564, 492 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "30") {
          const uint16_t rawData[] = { 3328, 9844, 488, 1588, 492, 552,
488, 540, 492, 540, 488, 1576, 488, 552, 488, 528, 536, 524, 492, 540, 488,
 544, 488, 544, 488, 548, 516, 516, 492, 540, 492, 540, 512, 516, 536, 1544,
 508, 1560, 532, 1556, 492, 1580, 516, 1544, 492, 540, 516, 524, 492, 1572,
 508, 1568, 508, 540, 516, 516, 516, 516, 516 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      }
      break;
    case 1:
      Serial.println("Quente");
      if(temperatureString == "16") {
          const uint16_t rawData[] = { 3240, 9860, 484, 1612, 492, 548,
492, 548, 492, 540, 492, 1572, 488, 544, 488, 540, 492, 540, 492, 540, 492,
 540, 512, 520, 488, 532, 532, 516, 516, 1548, 492, 540, 488, 540, 492, 540,
 492, 540, 492, 528, 536, 1556, 488, 544, 488, 1584, 488, 528, 536, 512,
536, 1544, 516, 516, 492, 540, 492, 1572, 492 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "17") {
          const uint16_t rawData[] = { 3224, 9856, 488, 1612, 516, 524,
516, 524, 512, 520, 488, 1576, 488, 528, 500, 544, 492, 536, 492, 544, 488,
 540, 492, 540, 508, 516, 532, 516, 488, 1572, 492, 540, 492, 540, 516, 516,
 492, 540, 516, 1544, 532, 516, 492, 540, 516, 1556, 488, 532, 532, 512,
536, 1548, 516, 516, 512, 1548, 492, 540, 516 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "18") {
          const uint16_t rawData[] = { 3248, 9828, 488, 1596, 516, 540,
488, 544, 488, 544, 488, 1576, 488, 524, 516, 540, 488, 544, 468, 564, 488,
 544, 488, 540, 492, 540, 488, 544, 488, 1576, 488, 532, 516, 544, 464, 564,
 488, 544, 488, 1584, 488, 1556, 516, 524, 512, 1584, 488, 524, 516, 544,
484, 1576, 488, 544, 488, 1556, 516, 1572, 488 };
```

```
            IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
       } else if(temperatureString == "19") {
            const uint16_t rawData[] = { 3248, 9856, 488, 1588, 516, 548,
488, 524, 516, 544, 488, 1580, 492, 544, 484, 548, 484, 552, 488, 532, 484,
 544, 488, 568, 468, 564, 460, 568, 488, 1576, 488, 532, 516, 540, 480, 552,
 488, 1576, 488, 544, 488, 544, 464, 544, 484, 1600, 488, 524, 516, 544,
488, 1580, 488, 1584, 464, 544, 512, 544, 488 };
            IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
       } else if(temperatureString == "20") {
            const uint16_t rawData[] = { 3248, 9836, 484, 1616, 512, 552,
488, 552, 488, 544, 464, 1596, 488, 544, 488, 544, 488, 544, 488, 540, 468,
 564, 488, 544, 464, 548, 516, 544, 488, 1572, 488, 548, 460, 568, 484, 548,
 488, 1592, 484, 544, 464, 1600, 488, 552, 488, 1564, 516, 524, 516, 520,
484, 1600, 488, 1584, 484, 532, 484, 1600, 488 };
            IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
       } else if(temperatureString == "21") {
            const uint16_t rawData[] = { 3248, 9856, 488, 1588, 516, 548,
488, 524, 516, 544, 464, 1604, 488, 544, 488, 544, 488, 552, 488, 552, 464,
 568, 484, 548, 484, 544, 488, 544, 464, 1600, 488, 532, 516, 540, 468, 564,
 488, 1576, 464, 1596, 488, 548, 464, 544, 516, 1584, 476, 556, 488, 544,
484, 1584, 488, 1572, 516, 1556, 516, 540, 464 };
            IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
       } else if(temperatureString == "22") {
            const uint16_t rawData[] = { 3248, 9824, 488, 1588, 516, 532,
512, 552, 488, 544, 464, 1616, 488, 524, 512, 548, 484, 544, 488, 544, 488,
 544, 488, 544, 488, 524, 516, 528, 516, 1584, 488, 552, 472, 560, 484, 548,
 484, 1576, 468, 1596, 488, 1580, 464, 552, 516, 1580, 488, 544, 464, 564,
492, 1580, 488, 1576, 488, 1580, 488, 1568, 512 };
            IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
       } else if(temperatureString == "23") {
            const uint16_t rawData[] = { 3220, 9868, 484, 1608, 516, 520,
516, 544, 488, 540, 488, 1584, 488, 544, 488, 544, 488, 544, 488, 540, 488,
 544, 488, 544, 488, 544, 468, 564, 488, 1572, 488, 552, 488, 544, 488,
1576, 488, 540, 492, 540, 488, 544, 488, 544, 488, 1576, 488, 544, 484, 544,
 488, 544, 488, 544, 488, 544, 488, 544, 464 };
            IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
       } else if(temperatureString == "24") {
            const uint16_t rawData[] = { 3248, 9836, 484, 1612, 516, 548,
488, 552, 488, 544, 488, 1576, 468, 564, 460, 568, 488, 544, 464, 568, 488,
 544, 488, 544, 472, 540, 512, 524, 484, 1600, 488, 544, 464, 568, 488,
1576, 460, 556, 516, 544, 488, 1576, 464, 576, 484, 1568, 512, 524, 516,
520, 512, 520, 508, 520, 488, 544, 488, 1600, 488 };
            IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
       } else if(temperatureString == "25") {
```

```
          const uint16_t rawData[] = { 3248, 9836, 484, 1616, 512, 552,
488, 552, 488, 544, 484, 1576, 488, 544, 488, 544, 488, 544, 488, 544, 484,
 548, 484, 548, 484, 524, 516, 544, 488, 1572, 488, 544, 464, 568, 488,
1576, 488, 532, 512, 1576, 488, 552, 488, 552, 464, 1588, 516, 540, 464,
548, 484, 548, 508, 520, 508, 1576, 468, 544, 484 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "26") {
          const uint16_t rawData[] = { 3328, 9868, 508, 1588, 516, 548,
464, 548, 516, 544, 464, 1608, 488, 544, 464, 564, 492, 548, 472, 568, 488,
 544, 488, 544, 484, 544, 488, 544, 464, 1600, 488, 532, 516, 520, 504,
1580, 488, 544, 488, 1576, 488, 1588, 488, 520, 512, 1576, 464, 544, 488,
540, 512, 520, 488, 548, 508, 1584, 464, 1608, 484 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "27") {
          const uint16_t rawData[] = { 3352, 9852, 488, 1604, 516, 540,
488, 544, 464, 568, 488, 1584, 484, 552, 488, 552, 488, 524, 516, 544, 488,
 544, 464, 568, 488, 548, 488, 544, 488, 1576, 488, 544, 488, 544, 488,
1572, 488, 1576, 488, 524, 516, 540, 492, 540, 488, 1584, 488, 544, 488,
520, 512, 544, 484, 1584, 488, 544, 488, 544, 464 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "28") {
          const uint16_t rawData[] = { 3352, 9856, 540, 1584, 488, 524,
516, 544, 488, 520, 508, 1584, 488, 520, 512, 544, 488, 524, 516, 532, 492,
 544, 516, 532, 516, 544, 464, 568, 484, 1576, 464, 568, 464, 544, 508,
1576, 488, 1576, 464, 552, 512, 1576, 464, 548, 492, 1596, 464, 544, 512,
520, 488, 540, 492, 1576, 508, 520, 492, 1576, 484 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "29") {
          const uint16_t rawData[] = { 3324, 9884, 488, 1592, 512, 552,
472, 556, 488, 544, 488, 1576, 488, 552, 484, 544, 488, 544, 464, 568, 464,
 568, 488, 544, 488, 524, 512, 548, 464, 1596, 488, 524, 516, 544, 484,
1576, 464, 1608, 488, 1576, 484, 548, 464, 544, 484, 1600, 468, 572, 480,
528, 488, 544, 484, 1600, 464, 1616, 464, 548, 516 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "30") {
          const uint16_t rawData[] = { 3376, 9816, 488, 1596, 512, 544,
488, 544, 464, 568, 464, 1624, 460, 548, 516, 544, 488, 544, 464, 544, 488,
 568, 460, 548, 508, 528, 488, 568, 464, 1604, 468, 572, 488, 528, 512,
1576, 464, 1588, 516, 1580, 488, 1584, 488, 508, 508, 1596, 488, 524, 516,
516, 532, 552, 464, 1596, 464, 1608, 464, 1588, 516 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      }
      break;
    case 2:
      Serial.println("Auto");
```

```
      if(temperatureString == "18") {
          const uint16_t rawData[] = { 3228, 9868, 488, 1588, 512, 524,
516, 516, 492, 540, 492, 1572, 500, 540, 516, 516, 516, 512, 532, 516, 516,
 516, 516, 516, 516, 504, 532, 516, 516, 516, 516, 1556, 516, 1544, 532,
516, 520, 504, 508, 1572, 516, 1544, 532, 1544, 536, 516, 516, 504, 536,
1560, 532, 1552, 512, 1572, 512, 1564, 516, 1548, 516 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "19") {
          const uint16_t rawData[] = { 3248, 9848, 484, 1596, 492, 548,
516, 504, 536, 512, 496, 1572, 488, 540, 492, 540, 492, 540, 492, 540, 488,
 544, 488, 540, 516, 524, 492, 540, 492, 540, 492, 1572, 492, 1576, 492,
540, 492, 1580, 488, 532, 508, 540, 492, 1580, 492, 528, 512, 536, 492,
1572, 492, 528, 536, 516, 492, 540, 492, 536, 492 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "20") {
          const uint16_t rawData[] = { 3224, 9848, 484, 1592, 516, 524,
488, 540, 492, 540, 492, 1600, 460, 576, 488, 524, 516, 524, 492, 540, 492,
 540, 488, 544, 488, 552, 488, 544, 488, 568, 488, 1580, 488, 1584, 512,
524, 516, 1584, 484, 508, 508, 1608, 464, 1584, 488, 528, 536, 516, 488,
1600, 464, 540, 492, 528, 536, 516, 488, 1600, 488 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "21") {
          const uint16_t rawData[] = { 3248, 9824, 488, 1596, 500, 532,
516, 516, 488, 544, 512, 1552, 488, 528, 536, 516, 492, 152, 876, 544, 512,
 516, 492, 540, 492, 540, 492, 540, 488, 544, 512, 1560, 488, 1588, 512,
520, 492, 1576, 492, 1588, 512, 520, 492, 1584, 516, 516, 492, 532, 532,
1560, 532, 508, 532, 508, 532, 1556, 488, 544, 512 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "22") {
          const uint16_t rawData[] = { 3220, 9852, 536, 1600, 464, 548,
516, 528, 516, 528, 488, 1616, 464, 528, 536, 516, 488, 540, 492, 544, 488,
 540, 492, 540, 492, 532, 532, 512, 532, 528, 488, 1572, 492, 1580, 492,
548, 488, 1576, 488, 1576, 512, 1560, 516, 1548, 492, 540, 516, 516, 488,
1576, 488, 528, 528, 524, 516, 1556, 488, 1588, 516 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "23") {
          const uint16_t rawData[] = { 3248, 9820, 488, 1592, 512, 552,
464, 568, 464, 568, 488, 1572, 464, 552, 488, 548, 516, 524, 492, 544, 484,
 544, 488, 544, 512, 528, 488, 544, 488, 544, 488, 1604, 464, 1604, 516,
1556, 516, 528, 508, 508, 532, 520, 488, 1608, 464, 540, 488, 544, 512,
1584, 488, 516, 492, 1592, 508, 520, 488, 544, 488 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "24") {
          const uint16_t rawData[] = { 3248, 9844, 520, 1608, 464, 528,
512, 536, 492, 540, 492, 1588, 492, 548, 492, 548, 488, 540, 512, 536, 492,
```

```
 540, 492, 540, 492, 528, 508, 540, 516, 516, 492, 1580, 492, 1568, 532,
1556, 492, 528, 536, 512, 536, 1560, 516, 1556, 492, 540, 516, 524, 512,
1548, 508, 532, 508, 1568, 532, 520, 504, 1572, 492 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "25") {
          const uint16_t rawData[] = { 3228, 9860, 520, 1608, 464, 548,
492, 540, 492, 540, 488, 1588, 516, 504, 512, 540, 492, 548, 492, 536, 532,
 516, 492, 540, 516, 508, 528, 520, 512, 520, 492, 1584, 492, 1580, 492,
1568, 536, 512, 492, 1580, 492, 528, 512, 1560, 512, 528, 508, 540, 492,
1580, 492, 540, 488, 1572, 492, 1572, 492, 528, 536 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "26") {
          const uint16_t rawData[] = { 3324, 9856, 488, 1596, 492, 540,
512, 520, 488, 544, 488, 1596, 492, 528, 536, 516, 488, 544, 488, 540, 492,
 540, 492, 540, 516, 524, 516, 516, 488, 540, 492, 1572, 492, 1572, 492,
1568, 508, 548, 492, 1588, 492, 1568, 508, 1564, 508, 532, 508, 536, 512,
1580, 488, 544, 512, 1556, 492, 1588, 488, 1560, 536 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "27") {
          const uint16_t rawData[] = { 3352, 9852, 488, 1588, 492, 576,
488, 544, 484, 516, 492, 1600, 464, 552, 488, 568, 488, 532, 516, 516, 512,
 516, 492, 544, 488, 548, 492, 540, 512, 524, 484, 1608, 488, 1564, 492,
1604, 492, 1560, 516, 508, 508, 540, 488, 1584, 488, 540, 492, 532, 532,
1580, 492, 1592, 492, 540, 492, 540, 492, 540, 492 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "28") {
          const uint16_t rawData[] = { 3328, 9884, 488, 1584, 492, 548,
492, 528, 536, 516, 512, 1560, 512, 516, 516, 516, 492, 548, 492, 548, 512,
 520, 492, 536, 492, 540, 492, 540, 516, 516, 516, 1556, 488, 1580, 492,
1580, 488, 1572, 492, 540, 516, 1536, 512, 1572, 492, 548, 512, 516, 516,
1548, 516, 1552, 516, 524, 516, 516, 516, 1544, 512 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "29") {
          const uint16_t rawData[] = { 3356, 9824, 488, 1584, 520, 532,
516, 524, 512, 520, 512, 1568, 488, 528, 536, 520, 484, 544, 488, 540, 492,
 540, 492, 540, 492, 532, 532, 512, 532, 524, 516, 1552, 512, 1556, 492,
1604, 464, 1572, 492, 1588, 488, 540, 492, 1576, 488, 540, 492, 540, 516,
1548, 492, 1568, 508, 540, 492, 1588, 488, 532, 532 };
          IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
NEC_KHZ);
      } else if(temperatureString == "30") {
          const uint16_t rawData[] = { 3324, 9876, 484, 1608, 512, 544,
464, 544, 484, 548, 484, 1584, 512, 528, 488, 568, 464, 540, 492, 540, 512,
 544, 464, 544, 488, 548, 492, 540, 492, 540, 516, 1572, 464, 1580, 484,
1600, 464, 1576, 508, 1584, 464, 1576, 508, 1556, 484, 552, 488, 544, 488,
1580, 488, 1600, 464, 552, 488, 1588, 492, 1596, 512 };
```

```
              IrSender.sendRaw(rawData, sizeof(rawData) / sizeof(rawData[0]),
    NEC_KHZ);
          }
          break;
      }

      delay(1000);
    }
  }
}

void reconnect() {
  // Loop ate conseguir conectar no broker MQTT
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Tentativa de conexao com o broker
    if (client.connect("ESP32-IR")) {
      Serial.println("connected");
      // Inscricao no topico MQTT
      client.subscribe("esp32/ir");
      client.subscribe("test");
      digitalWrite(internalLedPin, HIGH);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      delay(5000);
    }
  }
}
void loop() {
  if (!client.connected()) {
    digitalWrite(internalLedPin, LOW);
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 5000) {
    lastMsg = now;
  }
}
```

C.5 Código fonte do script de obtenção dos batimentos cardíacos

```
import requests
from time import sleep
```

```
import paho.mqtt.client as mqtt

# Endpoint da API
url = "https://dev.pulsoid.net/api/v1/data/heart_rate/latest?response_mode=
   text_plain_only_heart_rate"

api_call_headers = {'Authorization': 'Bearer ' + '<token API>'}

def on_message(client, userdata, msg):
    if (msg.topic == "heartbeat/get"):
        if (msg.payload.decode('utf-8') == "get_point"):
            # GET request para a API
            response = requests.get(url, headers=api_call_headers)

            response_json = response.json()

            # Publicacao da resposta no broker MQTT
            client.publish("heartbeat/return", response_json, 1)

broker_address="localhost"
client = mqtt.Client("heartbeat-script")
client.on_message = on_message
client.connect(broker_address, 1883) # Conexao com o broker
client.subscribe("heartbeat/get", 1)
client.loop_start()

stop = ""
while(stop != "q"):
  stop = input()
```

C.6 Código fonte do serviço de obtenção dos batimentos cardíacos

```
ï»¿using ControlPanelWebApp.Models;
using MQTTnet.Client;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ControlPanelWebApp.Services
{
    public class HeartbeatService
    {
        SQLiteService SQLiteService;
        MqttService MqttService;
```

```
        public HeartbeatService(SQLiteService sqliteService, MqttService
    mqttService)
        {
            SQLiteService = sqliteService;
            MqttService = mqttService;
            MqttService.RegisterOnMessageReceivedCallback(
    HeartbeatOnMessageReceivedCallback);

            Task.Run(GetHeartbeatTask);
        }

        private Task GetHeartbeatTask()
        {
            while (true)
            {
                MqttService.PublishMessage("heartbeat/get", "get_point").Wait()
    ;
                Task.Delay(1000).Wait();
            }
        }

        private Task HeartbeatOnMessageReceivedCallback(
    MqttApplicationMessageReceivedEventArgs e)
        {
            if(e.ApplicationMessage.Topic == "heartbeat/return")
            {
                Heartbeat heartbeat = new Heartbeat()
                {
                    Timestamp = DateTime.Now,
                    Bpm = Convert.ToInt32(Encoding.UTF8.GetString(e.
    ApplicationMessage.PayloadSegment))
                };

                SQLiteService.InsertToDb(heartbeat);
            }
            return Task.CompletedTask;
        }
    }
}
```

C.7 Código fonte do serviço que contém a lógica de ativação do sistema

```
ï»¿using ControlPanelWebApp.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```
namespace ControlPanelWebApp.Services
{
    public class SystemManagerService
    {
        private SQLiteService SQLiteService { get; set; }
        private LoggedUserService LoggedUserService { get; set; }
        private MqttService MqttService { get; set; }
        public SystemState State { get; set; }


        public SystemManagerService(SQLiteService sqliteService,
   LoggedUserService loggedUserService, MqttService mqttService)
        {
            SQLiteService = sqliteService;
            LoggedUserService = loggedUserService;
            MqttService = mqttService;
            State = SystemState.INACTIVE;

            Task.Run(MainLoop);
        }

        private Task MainLoop()
        {
            MqttService.PublishMessage("esp32/led", "off").Wait();
            while (true)
            {
                SystemUser systemUser = SQLiteService.GetAllFromDb<SystemUser
   >().First(u => u.Id == LoggedUserService.LoggedUser.Id);
                if(systemUser.Mode == SystemMode.Auto)
                {
                    switch (State)
                    {
                        case SystemState.INACTIVE:
                            if (SQLiteService.GetLastMinuteHeartbeatAverage()
   >= systemUser.BpmThreshold) /// Verificacao do limiar definido pelo usuario
                            {
                                State = SystemState.ACTIVE;
                                // Envia as configuracoes do modo automatico
   para os microcontroladores
                                foreach(Device device in SQLiteService.
   GetAllFromDb<Device>())
                                {
                                    switch (device.Type)
                                    {
                                        case DeviceType.Relay:
                                            string channel1 = device.
   AutoIsActive ? "channel1-on" : "channel1-off";
                                            string channel2 = device.
   AutoIsActiveSecondary ? "channel2-on" : "channel2-off";
                                            Task.Run(() => MqttService.
   PublishMessage("esp32/relay", channel1));
```

```
                                        Task.Run(() => MqttService.
PublishMessage("esp32/relay", channel2));
                                        break;
                                    case DeviceType.Light:
                                        if (device.AutoIsActive)
                                        {
                                            Task.Run(() => MqttService.
PublishMessage("esp32/led-strip", "led-on-" + SQLiteService.HexToRGB(device.
AutoRGB)));
                                        }
                                        else
                                        {
                                            Task.Run(() => MqttService.
PublishMessage("esp32/led-strip", "led-off"));
                                        }
                                        break;

                                    case DeviceType.AC:
                                        if (device.AutoIsActive)
                                        {
                                            Task.Run(() => MqttService.
PublishMessage("esp32/ir", $"air-on-m{(int)device.AutoAcMode}t{device.
AutoAcTemperature}&"));
                                        }
                                        else
                                        {
                                            Task.Run(() => MqttService.
PublishMessage("esp32/ir", "air-off"));
                                        }
                                        break;
                                    default:
                                        break;
                                }
                            }
                        }
                        break;
                    case SystemState.ACTIVE:
                        while (SQLiteService.GetLastMinuteHeartbeatAverage
() >= systemUser.BpmThreshold) /// Verificacao do limiar definido pelo
usuario
                        {
                            Task.Delay(systemUser.AutoActiveDuration *
1000).Wait();
                            systemUser = SQLiteService.GetAllFromDb<
SystemUser>().First(u => u.Id == LoggedUserService.LoggedUser.Id);
                        }
                        State = SystemState.INACTIVE;

                        // Envia as configuracoes do modo manual para os
microcontroladores
```

```
                        foreach (Device device in SQLiteService.
GetAllFromDb<Device>())
                        {
                            switch (device.Type)
                            {
                                case DeviceType.Relay:
                                    string channel1 = device.ManualIsActive
 ? "channel1-on" : "channel1-off";
                                    string channel2 = device.
ManualIsActiveSecondary ? "channel2-on" : "channel2-off";
                                    Task.Run(() => MqttService.
PublishMessage("esp32/relay", channel1));
                                    Task.Run(() => MqttService.
PublishMessage("esp32/relay", channel2));
                                    break;
                                case DeviceType.Light:
                                    if (device.ManualIsActive)
                                    {
                                        Task.Run(() => MqttService.
PublishMessage("esp32/led-strip", "led-on-" + SQLiteService.HexToRGB(device.
ManualRGB)));
                                    }
                                    else
                                    {
                                        Task.Run(() => MqttService.
PublishMessage("esp32/led-strip", "led-off"));
                                    }
                                    break;
                                case DeviceType.AC:
                                    if (device.ManualIsActive)
                                    {
                                        Task.Run(() => MqttService.
PublishMessage("esp32/ir", $"air-on-m{(int)device.ManualAcMode}t{device.
ManualAcTemperature}&"));
                                    }
                                    else
                                    {
                                        Task.Run(() => MqttService.
PublishMessage("esp32/ir", "air-off"));
                                    }
                                    break;
                                default:
                                    break;
                            }
                        }
                        break;
                    }
                }

                Task.Delay(500).Wait();
            }
```

```
        }

    }

    public enum SystemState
    {
        INACTIVE,
        ACTIVE
    }
}
```