UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

CAIO VINICIUS DOMINGUES KATO

Desenvolvimento de um Sistema de Automação Residencial com Microcontrolador ESP32 e Protocolo MQTT

Sorocaba
2024

CAIO VINICIUS DOMINGUES KATO

Desenvolvimento de um Sistema de Automação Residencial com Microcontrolador ESP32 e Protocolo MQTT

Projeto Final de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP), como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Everson Martins

Sorocaba
2024

K19d

Kato, Caio Vinicius Domingues
Desenvolvimento de um sistema de automação residencial com microcontrolador ESP32 e protocolo MQTT / Caio Vinicius Domingues Kato. -- Sorocaba, 2024
62 p. : il., tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Everson Martins

1. Automação residencial. 2. Internet das coisas. 3. Microcontroladores. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

Desenvolvimento de um Sistema de Automação Residencial com Microcontrolador ESP32 e Protocolo MQTT

CAIO VINICIUS DOMINGUES KATO

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO GRAU DE **BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

Prof. Dr. Everson Martins
Coordenador

BANCA EXAMINADORA*:*

Prof. Dr. Everson Martins

Orientador/UNESP-Campus de Sorocaba

Prof. Dr. Galdenoro Botura Junior

UNESP-Campus de Sorocaba

Prof. Dr. Eduardo Verri Liberado

UNESP-Campus de Sorocaba

Abril de 2024

AGRADECIMENTOS

Agradeço a toda minha família, em especial a meus pais Terezinha e Paulo, minha tia Cida e minha avó Kazuyo que sempre estiveram ao meu lado me apoiando, que sempre ressaltaram a importância dos estudos e que fizeram muitos sacrifícios para prestar todo o suporte que precisei. Agradeço também a todos os colegas de curso que conviveram diariamente comigo durante essa trajetória. Agradeço a todos os professores pelos ensinamentos, em especial ao Prof. Dr. Everson Martins que me orientou durante o desenvolvimento deste trabalho. Agradeço também a todos os técnicos dos laboratórios, equipe de limpeza, funcionários da secretaria e da biblioteca por todo o suporte prestado. Por fim, um agradecimento especial ao meu cachorro Shoi, que esteve presente comigo durante essa jornada, me alegrando e me dando forças para superar todos os desafios.

KATO, C. V. D. **Desenvolvimento de um sistema de automação residencial com microcontrolador ESP32 e protocolo MQTT**. 2024. 62 p. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação) – Instituto de Ciência e Tecnologia, Universidade Estadual Paulista, Sorocaba, 2024.

RESUMO

Nos últimos anos devido aos avanços tecnológicos a Internet das Coisas vem ganhando cada vez mais importância e há potencial para que esse cenário continue se expandindo gradativamente nos próximos anos. O conceito de Internet das Coisas que está atrelado à possibilidade da comunicação de diferentes tipos de dispositivos por meio da internet garante uma infinidade de aplicações nas mais variadas áreas. Nesse contexto, surge a Internet das Coisas aplicada no âmbito residencial, ou Automação Residencial como é mais comumente conhecida, que têm se tornado cada vez mais popular, devido às vantagens proporcionadas, como conforto, segurança, praticidade, economia, conveniência e controle. Dentre as principais aplicações destacam-se a automatização do sistema de iluminação para controlar que as lâmpadas apaguem ou acendam, do sistema de som para controlar caixas de som instaladas em qualquer cômodo da casa, do sistema de segurança através do acesso por biometria e acionamento de alarmes, etc. O custo para implementação dessas aplicações depende de alguns fatores como tamanho da residência, quantidade de dispositivos, tipo da automação desejada, consequentemente, o custo acaba sendo geralmente elevado. Contudo, existem soluções alternativas com custo mais acessível. Dessa forma, neste trabalho foi desenvolvido um sistema de automação residencial que pode ser acessado remotamente via computador ou smartphone, permitindo o controle e monitoramento do sistema de iluminação, temperatura, umidade, segurança e acesso de uma residência. Para isso foi utilizado a placa ESP32 que irá se comunicar com sensores por meio da aplicação do protocolo MQTT e de uma interface desenvolvida no Node-RED.

Palavras-chave: ESP32; MQTT; automação residencial; internet das coisas; Node-RED.

KATO, C. V. D. **Development of a home automation system with ESP32 microcontroller and MQTT protocol**. 2024. 62 p. Final Paper (Bachelor's degree in Control and Automation Engineering) – Institute of Science and Technology, São Paulo State University, Sorocaba, 2024.

ABSTRACT

In recent years, due to technological advances, the Internet of Things has become increasingly important and there is potential for this scenario to continue to gradually expand in the coming years. The concept of Internet of Things, which is linked to the possibility of communicating different types of devices via the internet, guarantees a multitude of applications in the most varied areas. In this context, Internet of Things applied in the residential environment, or Home Automation as it is more commonly known, appears, which has become increasingly popular, due to the advantages provided, such as comfort, security, practicality, economy, convenience, and control. Among the main applications, we highlight the automation of the lighting system to control that lamps turn off or on, the sound system to control speakers installed in any room in the house, the security system through biometric access and activation of alarms, etc. The cost to implement these applications depends on some factors such as size of the home, number of devices, type of automation desired, consequently, the cost generally ends up being high. However, there are alternative solutions at a more affordable cost. Therefore, in this work, a home automation system was developed that can be accessed remotely via computer or smartphone, allowing the control and monitoring of a home's lighting, temperature, humidity, security, and access system. For this, the ESP32 board was used, which will communicate with sensors through the application of the MQTT protocol and an interface developed in Node-RED.

Keywords: ESP32; MQTT; home automation; internet of things; Node-RED.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Esta seção irá apresentar as justificativas acerca do tema escolhido e os objetivos que devem ser alcançados com o desenvolvimento deste trabalho. Também será detalhado como foi realizada a divisão dos conteúdos, indicando o assunto tratado em cada um dos tópicos.

1.1 Justificativa

O termo *Internet of Things* (IoT), que traduzido para o português significa Internet das Coisas, surgiu no ano de 1999 pelo pesquisador britânico Kevin Ashton do *Massachusetts Institute of Technology* (MIT) em uma pesquisa sobre identificadores de rádio frequência, onde ele apresentou sua ideia para melhorar a logística de produção, que consistia em colocar identificadores e conectividade sem fio nos objetos, de modo a possibilitar o gerenciamento pelo computador (MAGRANI, 2018).

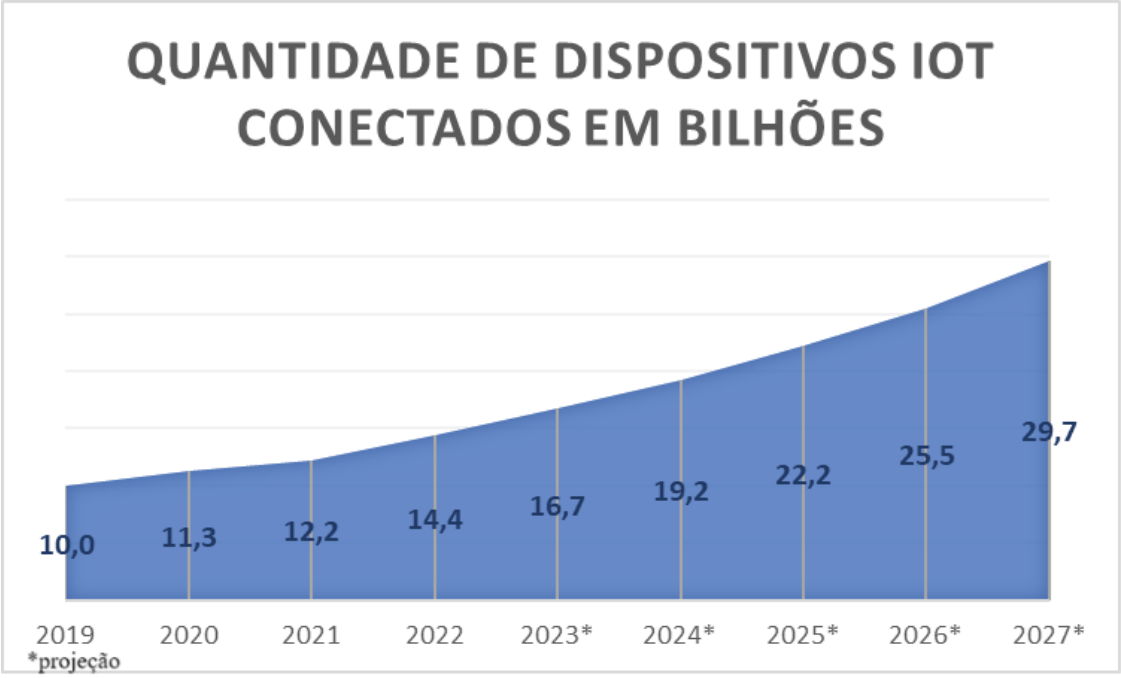
Segundo Cleber Paradela, professor de futurismo da ESPM, o crescimento da IoT começou quando os computadores deixaram de ser utilizados somente como uma máquina de escrever e passaram a ter inteligência.

> A internet das coisas é mais uma nova fase desse processo: pegar objetos que até então funcionavam isoladamente, de forma analógica, e fazer com que eles gerem dados e informações que possam transformar o mundo à nossa volta. Portanto, IoT é a capacidade de você identificar, observar e controlar o mundo físico por meio de sensores (ESPM, 2021).

Com o passar dos anos e evolução da tecnologia, o conceito de IoT foi ganhando forma e conquistando seu espaço nos mais diversos segmentos da sociedade, destacando-se inicialmente a automação nas indústrias. Segundo relatório divulgado em maio de 2023 pela IoT Analytics, conforme mostra a Figura 1 o número de aparelhos ativos em 2022 foi igual a 14,3 bilhões representando um crescimento de 18% em relação ao ano de 2021, a projeção para os anos seguintes é de que em 2023 chegue em 16,7 bilhões o número de aparelhos ativos e até 2027 atinja o número de 29 bilhões de conexões IoT (IOT ANALYTICS, 2023).

Além dessa popularização, o desenvolvimento tecnológico acelerado aliado com o barateamento do custo, fizeram com que a automação começasse a ser também aplicada nas residências.

Figura 1 *- Gráfico com a quantidade de dispositivos IoT conectados.*



Fonte: Adaptado de IoT Analytics (2023).

A IoT aplicada no âmbito residencial, conhecida como automação residencial ou domótica proporciona uma série de vantagens, como conforto, segurança, praticidade, economia, conveniência, controle. Sendo assim, progressivamente a presença da tecnologia vêm se tornando cada vez mais essencial no cotidiano das pessoas, e nesse cenário percebe-se que a automação residencial vem cada vez mais ganhando espaço.

Segundo um estudo realizado pelo IDC Brasil, este mercado de automação residencial tinha previsão de aumentar em 21% no ano de 2021 e a expectativa de crescimento para os próximos anos é de aproximadamente 30%. Ou seja, espera-se que aumente cada vez mais a quantidade de residência automatizadas no Brasil (IDC, 2021).

Nesse contexto, a proposta deste projeto é o desenvolvimento de um sistema de automação residencial que pode ser acessado remotamente por dispositivos como celulares ou computadores, permitindo o controle e monitoramento dos sistemas de iluminação,

temperatura, umidade, segurança e acesso de uma residência, utilizando como componente principal o microcontrolador ESP32.

1.2 Objetivo Geral

Desenvolver um sistema de automação residencial para controlar e monitorar os sistemas de iluminação, temperatura, umidade, segurança e acesso de uma residência, através de uma interface com o usuário que poderá ser acessada remotamente pelo celular ou pelo computador.

1.3 Objetivos Específicos

- Definição do circuito elétrico do projeto;
- Definição do circuito eletrônico do projeto;
- Desenvolvimento da programação do microcontrolador;
- Desenvolvimento da página web para interação com o usuário;
- Análise dos resultados.

1.4 Organização do trabalho

O conteúdo deste trabalho está organizado em 6 capítulos. No primeiro capítulo são apresentadas contextualizações e justificativas sobre os objetivos do desenvolvimento do sistema de automação residencial. No capítulo seguinte são explorados conceitos teóricos acerca do tema. Em seguida é detalhado a proposta de desenvolvimento, bem como os materiais e métodos utilizados. Nos capítulos 4 e 5, são respectivamente apresentados, o desenvolvimento e discussão sobre resultados. Por fim, no capítulo 6 é apresentada as conclusões sobre o desenvolvimento do trabalho e dos resultados obtidos.

2 REFERENCIAL TEÓRICO

Esta seção irá apresentar um embasamento teórico sobre Internet das Coisas, automação residencial e protocolo MQTT. O entendimento destes conceitos é importante para posteriormente compreender o desenvolvimento do sistema de automação residencial.

2.1 Estado da Arte

Santos e Junior (2019) desenvolveram um sistema de automação residencial utilizando o microcontrolador ESP32 visando um baixo custo. Os elementos que foram controlados via smartphone foram LEDs, portão elétrico e um sistema de alarme.

Silva (2021) propôs um sistema de monitoramento e controle de energia utilizando o microcontrolador ESP32. Para a medição do consumo de energia elétrica, foi utilizado sensor de corrente, onde os valores medidos eram armazenados em um banco de dados.

Costa (2020) desenvolveu um sistema de automação residencial utilizando ESP32. Os dispositivos que foram automatizados foram: controle de iluminação, monitoramento do nível de água com acionamento automático da bomba, monitoramento da temperatura e sistema de segurança. Para desenvolver o sistema responsável por controlar esses dispositivos, foi utilizado o Apache Cordova.

Em 2019 Martins desenvolveu um sistema de automação residencial a partir do protocolo MQTT, node-red, Mosquitto broker e os microcontroladores ESP32 e ESP8266. A automação foi destinada para a cozinha e o escritório de uma residência. O sistema desenvolvido tinha a função de monitorar os dados de temperatura, umidade e alarme.

2.2 Internet das Coisas (IoT)

Segundo Bassi e Horn (2008) Internet das Coisas é definido como “coisas que tem identidades e personalidades virtuais operando em espaços inteligentes usando interfaces inteligentes para conectar e comunicar dentro de contextos sociais, ambientais e de usuário”, onde a expressão Internet das Coisas pode ter o conceito obtido a partir da interpretação das

palavras “internet” e “coisas”: internet sendo definida como a rede de computadores mundialmente interconectadas baseada em um protocolo de comunicação e coisas sendo definida como um objeto não precisamente identificável, ou seja, Internet das Coisas significa uma rede mundial de objetos interconectados, baseado em protocolos de comunicação.

Para Sacomano (2018) esses objetos não se limitam apenas aos celulares, notebooks e computadores, que são os mais comuns, mas também englobam uma infinidade de outros tipos de dispositivos, conforme ilustra a Figura 2, como por exemplo, máquinas, veículos, aparelhos eletrodomésticos, sistemas de iluminação, segurança e som etc.

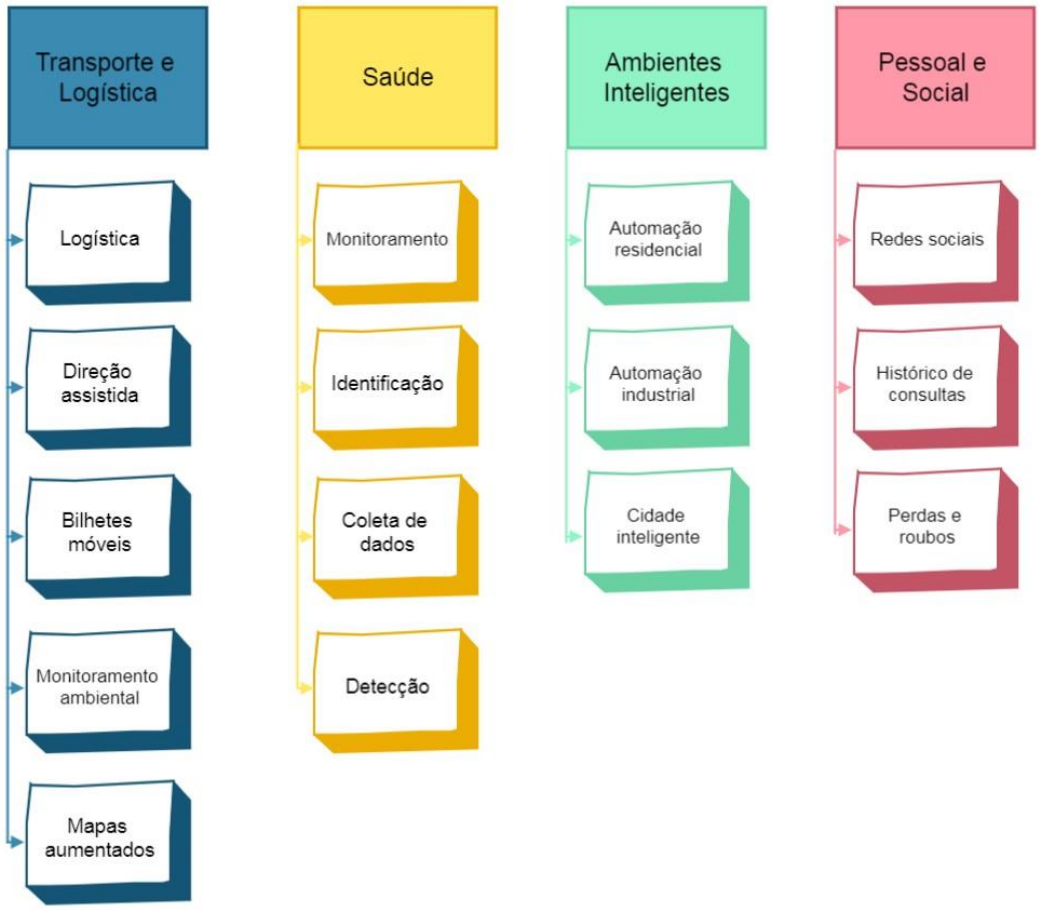
Figura 2 *- Dispositivos conectados pela Internet das Coisas.*

Fonte: Alecrim (2016).

Devido a essa grande diversidade de dispositivos, evidencia-se um grande potencial da Internet das Coisas que possibilita o desenvolvimento de aplicações nas mais variadas áreas, dentre as quais se destacam: transporte e logística, saúde, ambientes inteligentes, domínio pessoal e social, conforme exemplifica o diagrama de blocos da Figura 3.

Na área de transporte e logística as principais aplicações são: monitorar em tempo real todas as etapas da logística; obter informações importantes para uma direção em segurança; ter acesso à determinadas informações sobre serviços ou visualizar o mapa da região ao apontar o celular para uma etiqueta *Near-Field Communication* (NFC); monitoras as variáveis como temperatura e umidade durante o trajeto do local de produção até o local de consumo (ATZORI; IERA; MORABITO, 2010).

Figura 3 *- Diagrama de Blocos das principais áreas das aplicações com Internet das Coisas.*



Fonte: Atzori, Iera e Morabito (2010).

Na área da saúde os principais benefícios são: monitoramento da localização do paciente em tempo real e do fluxo do trânsito decorrente do trajeto entre o hospital e a casa do paciente; identificação do paciente para evitar incidentes como medicamento/procedimentos errados; coleta e transferência automática de dados visando a redução do tempo do preenchimento de formulários; detectar o diagnóstico a partir do monitoramento dos indicadores de saúde do paciente (ATZORI; IERA; MORABITO, 2010).

Na área de ambientes inteligentes os principais benefícios são: controlar a iluminação, a temperatura e o sistema de alarme de uma residência; execução de determinadas ações a partir a leitura de etiquetas *Radio Frequency Identification* (RFID) que enviam notificações para uma máquina e/ou robô; controle da iluminação pública (ATZORI; IERA; MORABITO, 2010).

Na área pessoal e social os principais benefícios são: atualização automática de informações em redes sociais; histórico de consultas das atividades para observar a existência

de alguma tendência; mecanismo de busca para objetos perdidos ou roubados (ATZORI; IERA; MORABITO, 2010).

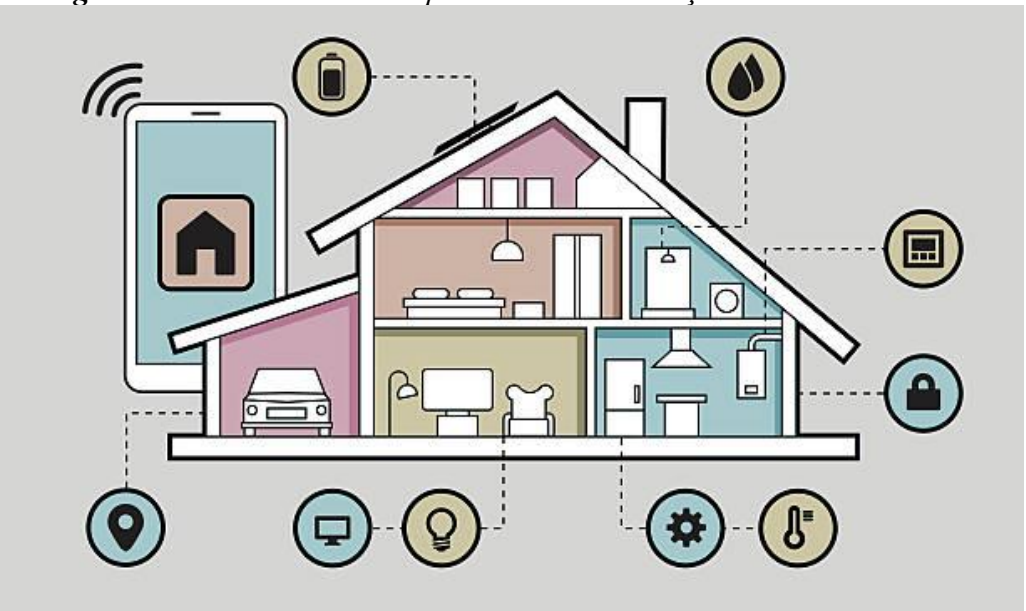
2.3 Automação Residencial

Automação residencial ou domótica é o nome dado para a junção de tecnologias que possibilitam o controle automatizado de uma residência, simplificando as ações cotidianas das pessoas. A denominação domótica é derivada do termo francês *domotique*, que se baseia na junção de duas palavras: domus que significa casa e robótica que significa controle automatizado de algo. Essa denominação foi utilizada pela primeira vez na França na década de 80, época em que houve os primeiros relatos de edifícios com funções de controlar iluminação, segurança e a climatização (BARROS, 2010).

Para Abreu (2013) a domótica é uma ciência que tem como alicerce quatro fatores: eficiência energética, segurança, comunicação e conforto. Dessa forma, para alcançar a integração desses fatores, é necessário um conjunto de dispositivos distribuídos na residência, de acordo com as necessidades dos proprietários. Esses dispositivos são sensores, atuadores, controladores e interfaces.

Barros (2010) afirma que a domótica é uma opção imprescindível com soluções originais e que se destacam por sua diversidade, conforme ilustra a Figura 4, onde é possível observar diferentes tipos de dispositivos espalhados por todos os cômodos de uma residência. Os objetivos para a utilização desses dispositivos podem ser inúmeros, por exemplo, controlar a iluminação dos cômodos, evitando a necessidade de a pessoa precisar ir até o interruptor para ligar/desligar a lâmpada, ou para controlar a temperatura, umidade e segurança dos cômodos.

***Figura 4** - Diversidade de dispositivos na automação de uma residência.*



Fonte: Gettyimages (2016).

2.4 Protocolo de Comunicação MQTT

MQTT é a sigla para *Message Queuing Telemetry Transport*, segundo sua definição oficial trata-se de um protocolo de transporte de mensagens, sendo que seu modo de operação é baseado em *Publish/Subscribe*, permitindo que dispositivos publiquem mensagens nos tópicos desejados, ao mesmo tempo que possibilita que dispositivos possam receber as mensagens publicadas nos tópicos. O protocolo MQTT é leve, simples e projetado para ser fácil de implementar, dessa forma torna-se ideal para aplicações de Internet das Coisas (HIVEMQ, 2015b).

A origem do protocolo MQTT se deu no ano de 1999, quando Andy Stanford-Clark da IBM e Arlen Nipper da Cirrus Link necessitavam monitorar oleodutos via satélite, para isso precisavam que o protocolo tivesse implementação simples, qualidade de serviço na entrega de dados, leve e largura de banda eficiente, dessa forma, a partir desses requisitos desenvolveram a primeira versão do MQTT. Ao decorrer dos anos, novas versões foram sendo desenvolvidas, de modo que o foco passou a ser em aplicações de Internet das Coisas (HIVEMQ, 2015b).

Como mencionado anteriormente o MQTT se baseia no modelo de Publish/Subscribe e seu funcionamento necessita da pilha TCP/IP, para isso existem três componentes principais: cliente, *broker* e tópicos (HIVEMQ, 2015b).

O cliente pode ser qualquer dispositivo, desde um microcontrolador até um servidor, que execute uma biblioteca MQTT e se conecte em um *broker* através de uma rede. O cliente pode ser denominado como *publisher* quando o dispositivo é quem publica as mensagens, enquanto o *subscriber* é o dispositivo que recebe as mensagens, porém, existem casos em que o dispositivo é ao mesmo tempo um *publisher* e um *subscriber* (HIVEMQ, 2019).

O *broker* é um servidor intermediário que é responsável por receber as mensagens publicadas pelos dispositivos *publishers* e distribuir essas mesmas mensagens para os dispositivos *subscribers* (HIVEMQ, 2019).

Por fim, tem-se os tópicos que são *strings* onde as mensagens são publicadas e assinadas. O cliente deve realizar inscrição nos tópicos em que possui interesse em receber as mensagens publicadas, consequentemente, o *broker* irá acompanhar as assinaturas e encaminhará as mensagens publicadas em cada um dos tópicos para os respectivos assinantes. Cada tópico permite a assinatura de vários clientes ao mesmo tempo, da mesma forma que um mesmo cliente pode assinar diferentes tópicos ao mesmo tempo. A estrutura de um tópico é hierárquica, podendo ter vários níveis separados pelo caractere "/", conforme mostra o exemplo da Figura 5, onde observa-se quatro tópicos de exemplo, onde todos estão atribuídos no primeiro nível "casa", no segundo nível estão separados pelos cômodos e no terceiro nível estão separados pelo tipo de finalidade do dispositivo (HIVEMQ, 2019).

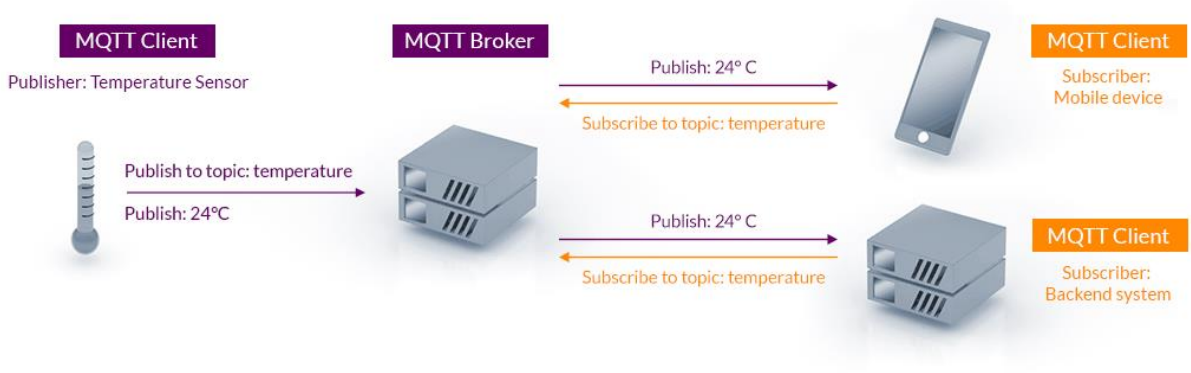
Figura 5 *- Exemplo da nomenclatura de tópicos.*

Tópicos:
casa/cozinha/luz
casa/cozinha/temperatura
casa/sala/luz
casa/quarto/luz

Fonte: Autoria própria.

A Figura 6 apresenta um exemplo de estrutura do protocolo MQTT. No exemplo existem três dispositivos clientes e um *broker*. O sensor de temperatura está publicando o valor obtido no tópico "temperature", o *broker* por sua vez, está recebendo essa mensagem e está encaminhando-a para os outros dois dispositivos que realizaram a assinatura deste tópico.

Figura 6 *- Estrutura do protocolo MQTT.*



Fonte: MQTT (2022).

Os clientes não conseguem realizar a conexão entre si, somente com o *broker*. Uma vez que a conexão entre cliente e *broker* seja estabelecida, ficará ativa até que o cliente solicite desconexão ou que ocorra uma interrupção da conexão (HIVEMQ, 2015c).

O protocolo MQTT possui o recurso de Qualidade de Serviço, traduzido do termo em inglês *Quality of Service* (QoS) que fornece três níveis diferentes de garantia da mensagem ser entregue para o destinatário: QoS 0, QoS 1 e QoS 2. A Figura 7 ilustra as diferenças entre os níveis (HIVEMQ, 2015d).

No nível QoS 0 a mensagem é enviada apenas uma vez e sem garantia que a mensagem será entregue ao receptor. Dessa forma, garante mais rapidez na transmissão da mensagem, mas existe a possibilidade que mensagem seja perdida. Sua utilização é indicada quando há uma conexão completa e estável entre o emissor e o receptor, ou em casos em que não tem problema se a mensagem for perdida (HIVEMQ, 2015d).

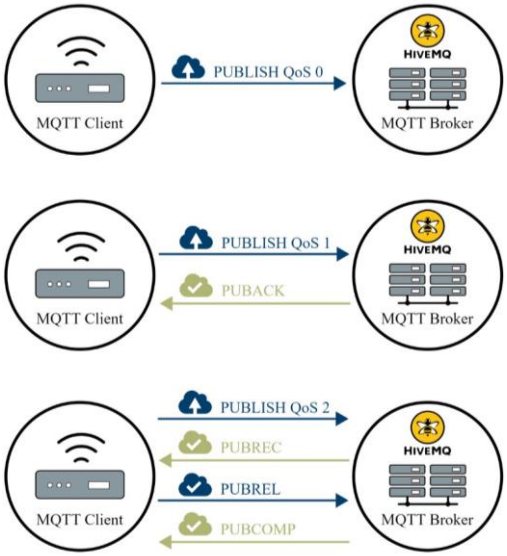
No nível QoS 1 existe garantia que a mensagem será entregue pelo menos uma vez para o receptor. Quando a mensagem é publicada, o emissor mantém uma cópia da mensagem até que o receptor envie uma confirmação do recebimento da mensagem, essa confirmação é chamada de “PUBACK”, caso após algum tempo essa confirmação não seja recebida, o emissor reenvia a mensagem. Dessa forma, existe a possibilidade de a mensagem ser enviada várias vezes e o emissor processar ela várias vezes (HIVEMQ, 2015d). Sua utilização é indicada quando é necessário garantia que a mensagem seja entregue e que não tenha problema caso a mesma mensagem seja recebida múltiplas vezes.

No nível QoS 2 a mensagem é enviada apenas uma vez e existe a garantia que será entregue exatamente uma vez para o receptor. Quando a mensagem é publicada, o receptor

processa a mensagem recebida e envia como resposta uma mensagem de confirmação, chamada de “PUBREC”. Enquanto o emissor não receber essa mensagem, ele irá enviar novamente a mensagem publicada. Quando o emissor receber a mensagem de confirmação, ele apaga a mensagem inicial, armazena a mensagem “PUBREC” e envia uma mensagem “PUBREL”. Quando o receptor recebe a mensagem “PUBREL”, ele processa a mensagem encaminhando-a para os assinantes e apaga o que estiver armazenado. Para finalizar, envia como resposta a mensagem “PUBCOMP”. Assim, nesse nível é garantido que tanto o receptor quanto o emissor tenham certeza de que a mensagem foi entregue, porém, o processo de transmissão da mensagem é mais lento que os outros dois níveis, portanto, é indicado que seja utilizado somente em situações em que seja necessário que a mensagem seja entregue somente uma única vez, sem a possibilidade de ser perdida ou duplicada (HIVEMQ, 2015d).

Os níveis de QoS definidos pelo emissor e pelo receptor podem ser diferentes, neste caso, o *broker* irá receber a mensagem do emissor com a QoS definida por ele, e irá encaminhar a mensagem com a QoS definida pelo receptor.

Figura 7 *- Níveis de Qualidade de Serviço do Protocolo MQTT.*



Fonte: HIVEMQ (2015d).

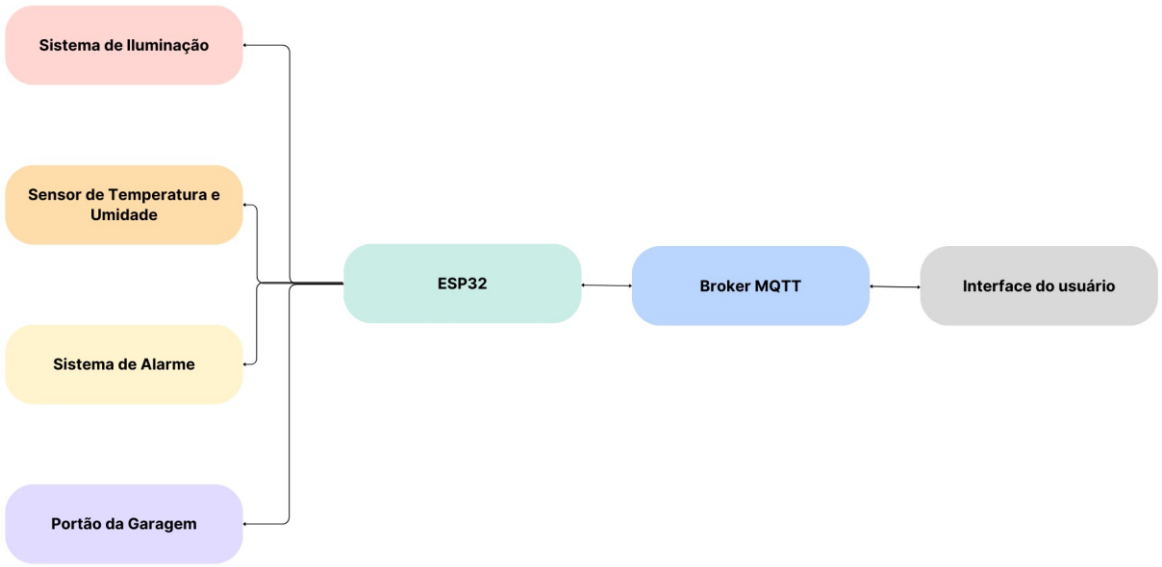
3 SISTEMA DE AUTOMAÇÃO RESIDENCIAL

Esta seção irá apresentar o sistema de automação residencial desenvolvido, listando todos os materiais e ferramentas utilizadas e descrevendo a metodologia aplicada.

3.1 Proposta do Trabalho

A proposta deste trabalho é o desenvolvimento de um sistema de automação residencial que permite o controle e monitoramento do sistema de iluminação, temperatura, umidade, segurança e acesso de uma residência, através de uma interface que pode ser acessada por computadores e smartphones. Para isso, será utilizado o protocolo de comunicação MQTT e o microcontrolador ESP32 que irá gerenciar a comunicação entre o *broker* e os dispositivos da residência, como pode ser visto na Figura 8.

Figura 8 *- Arquitetura do sistema de automação residencial.*



Fonte: Autoria própria.

O sistema de iluminação será composto por todas as lâmpadas da residência, dessa forma será possível acender ou apagar cada uma delas e consultar o respectivo estado delas. O sensor de temperatura e umidade ficará responsável por coletar os dados referentes à temperatura e umidade do ambiente e enviá-los ao ESP32. O sistema de alarme terá o funcionamento baseado em um sensor de movimento, a partir da detecção de algum movimento, um sinal será enviado para o ESP32 que consequentemente deverá acionar um alarme sonoro proveniente de um módulo buzzer. O portão da garagem terá sua abertura/fechamento controlado por um micro servo motor. Ademais, tudo poderá ser controlado e monitorado por uma interface desenvolvida utilizando a ferramenta Node-RED. Para melhor visualização desse sistema, será montada uma maquete de uma casa e todos os componentes serão fixados e conectados nela.

A ideia inicial é que o sistema proposto seja capaz de cumprir os requisitos abaixo:

- O intervalo de tempo entre o comando ser dado na interface e a ação ser executada, deve ser no máximo 1 segundo;
- O som emitido pelo sistema de alarme deve ser audível de qualquer lugar da residência;
- O sistema consiga ser controlado e monitorado mesmo que a pessoa esteja longe da residência.

3.2 Materiais e métodos

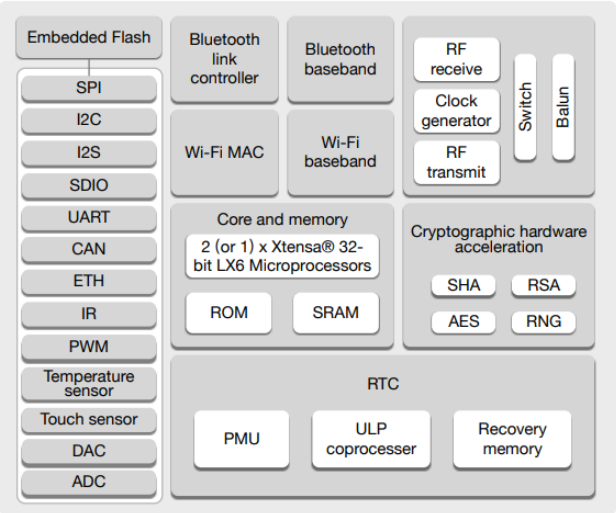
3.2.1 ESP32

O ESP32 é um microcontrolador desenvolvido pela empresa chinesa Espressif Systems, baseado em um processador dual-core Tensilica Xtensa 32-bit LX6 e suporte embutido para rede WiFi, Bluetooth v4.2 e memória flash integrada. A arquitetura do ESP32 está apresentada na Figura 9.

Suas principais características são: baixo consumo de energia, amplificador de baixo ruído, alto desempenho de potência, robustez, versatilidade e confiabilidade. Para a programação do ESP32, há suporte para diversas linguagens de programação, sendo que as

mais comumente utilizadas são C, C++ e Python. Todas as características tornam o ESP32 ideal para diversas aplicações em projetos de IoT e robótica.

Figura 9 *- Diagrama de Blocos do ESP32.*



Fonte: Espressif Systems (2017).

O ESP32 apresenta-se como sucessor de outro microcontrolador da Espressif Systems, o ESP8266, que se tornou muito utilizado desde seu lançamento em 2015. Porém, o ESP32, como foi desenvolvido somo sucessor, apresenta requisitos de hardware superiores e com recursos adicionais, conforme mostrado na comparação da Tabela 1.

Tabela 1 - Comparação entre ESP32 e ESP8266.

Descrição	**ESP32**	**ESP8266**
WiFi	Sim	Sim
Bluetooth	Bluetooth Low Energy v4.2 (BLE)	Não Possui
Conversores ADC	18 ADC com 12-bit de resolução	1 ADC com 10-bit de resolução
Conversores DAC	2 DAC com 8-bit de resolução	Nenhum
Corrente de Consumo	Média de 80 mA	Média de 80 mA
Interfaces de Módulos	SPI, SDIO, LED PWM, Motor PWM, I2S e IR	SPI, SDIO, LED PWM, I2S e I2C.
Memória FLASH	16 MB	16 MB
Memória RAM/SRAM	520 kB	36 kB
Pinos de I/O	36	17

Fonte: Oliveira (2019).

As principais vantagens do ESP32 em relação ao ESP8266 são: quantidade de pinos com conversor analógico para digital, pino com conversor digital para analógico e o suporte para Bluetooth. Em comparação com outros microcontroladores como, por exemplo, o Arduino Uno, o ESP32 se apresenta como uma melhor opção para ser utilizado neste trabalho, devido principalmente ao suporte para conectividade WiFi. Para simplificar a utilização do ESP32, utilizou-se o NodeMCU-32S, que está exibido na Figura 10.

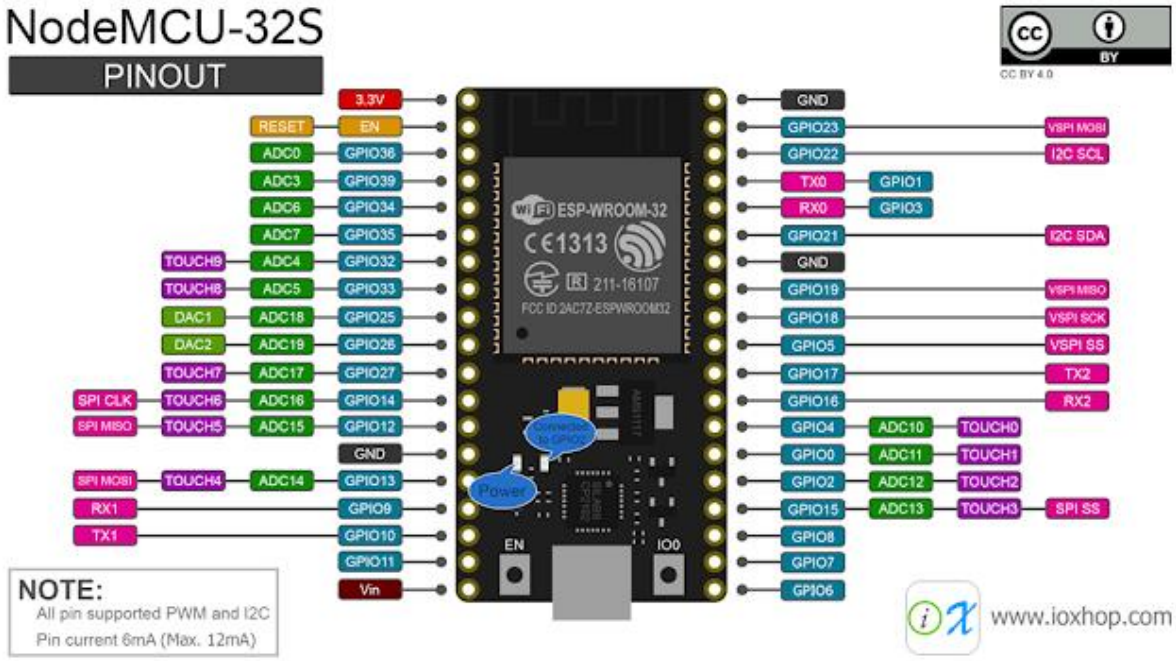
O NodeMCU 32S é uma placa de desenvolvimento baseada no ESP32 e que possui o módulo ESP-WROOM-32, antena embutida e porta micro USB para alimentação e programação. A faixa de tensão de operação varia entre 2,2 V e 3,6 V enquanto a corrente de operação é de 80 mA.

Figura 10 *- NodeMCU 32S.*

Fonte: UsinaInfo ([ca. 2020a]).

Na Figura 11 está exibido a pinagem do NodeMCU 32S, observa-se que ele é composto por 38 pinos, dois LEDs internos, um porta micro USB e os dois botões: *reset* (EN) e *boot* (IO0). O botão de reset quando pressionado envia um pulso de nível lógico baixo no pino EN, fazendo com que ocorra o *reset* do ESP32. O botão de *boot* quando pressionado envia um pulso de nível lógico baixo no pino IO0 permitindo a gravação do programa no ESP32.

Figura 11 - Pinagem do NodeMCU 32S.



Fonte: Koyanagi (2018).

Dentre os pinos do NodeMCU 32S, existem 2 destinados a alimentação: o pino 3.3V e o pino Vin. O pino 3.3V é a saída do regulador de tensão 3,3V enquanto no pino Vin pode ser conectado uma alimentação regulada de 5 V. Também conta com 3 pinos de *ground*.

Possui 32 pinos *General Purpose Input/Output* (GPIO) que são as portas responsáveis pela comunicação de entrada e saída de sinais digitais. Estes pinos requerem nível lógico de 3,3 V, não tolerando 5 V. Todos os pinos de GPIO podem ser configurados para gerar ondas com modulação por largura de pulso, traduzido do termo em inglês *Pulse Width Modulation* (PWM), para acionamento de motores digitais e LEDs.

3.2.2 Sensores

Os sensores são dispositivos que possuem a função de detectar alguma entrada do ambiente físico e a partir disso transformar em dados para que possa ser interpretado por um ser humano ou por máquinas. Os sensores possuem um papel importante na IoT, onde esses dispositivos coletam os dados de um determinado ambiente e os transmitem através de uma rede, dessa forma os dados desse ambiente podem ser monitorados e controlados de uma forma

mais eficiente. Existem diversos tipos de sensores que dependendo de sua aplicação diferem no tipo de entrada detectado, como por exemplo, temperatura, umidade, pressão, movimento, luz, fumaça, calor etc. Devido a essa diversidade de tipos de sensores, hoje em dia são utilizados nas mais variadas áreas da sociedade.

No setor industrial os sensores podem ser utilizados para medir e controlar a direção, a velocidade, a temperatura, a posição, e o nível dos materiais, garantindo uma maior segurança aos funcionários e uma maior eficiência do processo da linha de produção. Na agricultura os sensores podem ser utilizados para verificar a qualidade do solo, as condições climáticas, monitoramento de pragas e doenças, dessa forma possibilita aumentar a produtividade e otimizar o uso de recursos humanos e insumos. Na automação residencial os sensores podem ser utilizados para se prevenir contra roubos e incêndio, para medir a temperatura e umidade do ambiente, para monitorar a abertura de janelas e portas.

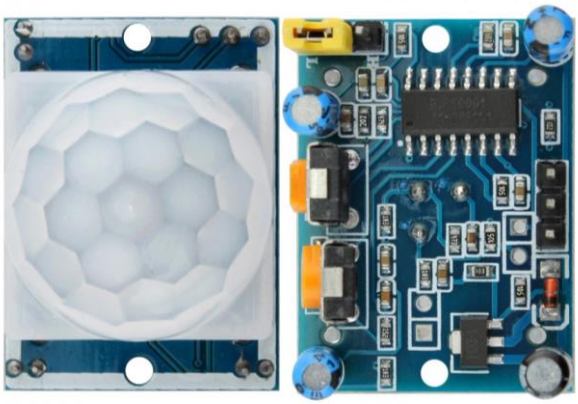
Para o desenvolvimento deste trabalho foram utilizados dois sensores: sensor de movimento e sensor de temperatura e umidade.

3.2.3 Sensor de Movimento

O sensor de movimento infravermelho passivo ou *Passive Infrared* (PIR) que foi utilizado é do modelo HC-SR501, que pode ser visto na Figura 12. Este dispositivo possui dois componentes principais: um sensor piroelétrico e uma lente especial chamada Fresnel.

O sensor piroelétrico é constituído de um par de eletrodos de sinais opostos e por uma janela com duas ranhuras retangulares que permite a passagem da radiação infravermelha. Os eletrodos são conectados de uma forma que se cancelem e não produzam nenhum sinal enquanto não há movimento detectado. Quando um movimento é detectado, o sensor emite um sinal de 3,3 V. A lente Fresnel é utilizada com o objetivo de aumentar o alcance e o campo de visão do sensor.

Figura 12 - Sensor PIR de Movimento HC-SR501.



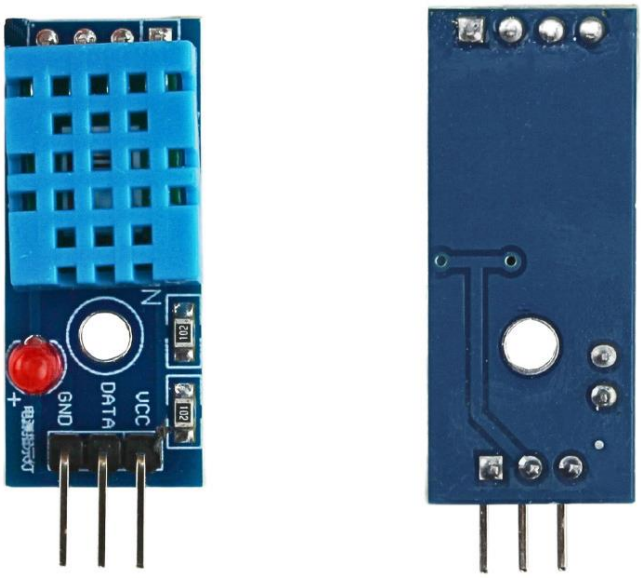
Fonte: UsinaInfo ([ca. 2022]).

Este modelo é capaz de detectar movimentos até a distância de 7 metros e trabalhando com ângulo de até 100º. Além disso, na Figura 12 é possível notar que ele possui dois potenciômetros em sua parte traseira: um para definir a distância da detecção e outro para definir o tempo que a saída permanecerá em nível lógico alto após a detecção do movimento. Possui três pinos: um pino Vcc, um pino *ground* e um pino de dados.

3.2.4 Sensor de Temperatura e Umidade

O sensor de temperatura e umidade que foi utilizado é do modelo DHT11, que pode ser visto na Figura 13. Este dispositivo é composto por um sensor capacitivo para medir a umidade e um termistor para medir a temperatura. O sensor capacitivo é formado por dois eletrodos com um substrato que retém a umidade entre eles, assim, conforme a umidade do ambiente sofre alteração, a resistência entre os eletrodos varia, podendo ser medida para estimar a umidade relativa do ambiente. O termistor é do tipo *Negative Temperature Coeficient* (NTC), portanto o valor de sua resistência diminui conforme a temperatura aumenta.

Figura 13 - Sensor de temperatura e umidade DHT11.



Fonte: UsinaInfo ([ca. 2014]).

Este sensor mede temperaturas entre o intervalo de 0ºC e 50ºC com precisão de ±2°C e mede a umidade entre 20% e 90% com precisão de ±5%. Possui três pinos: um pino Vcc, um pino *ground* e um pino de dados.

3.2.5 Micro Servo Motor

Para controlar a abertura e fechamento do portão da residência foi utilizado um Micro Servo Motor SG90 que está ilustrado na Figura 14. Servo motores são pequenos dispositivos leves, mas que apresentam alto desempenho. A partir de um sinal de controle que define um ângulo específico, controla com precisão o giro e a posição. Este modelo utilizado possibilita a rotação de 180º, sendo ideal para diversas aplicações de robótica e eletrônica. Possui três pinos: um pino Vcc, um pino *ground* e um pino de dados.

Figura 14 - Micro Servo Motor SG90.

Fonte: UsinaInfo ([ca. 2016]).

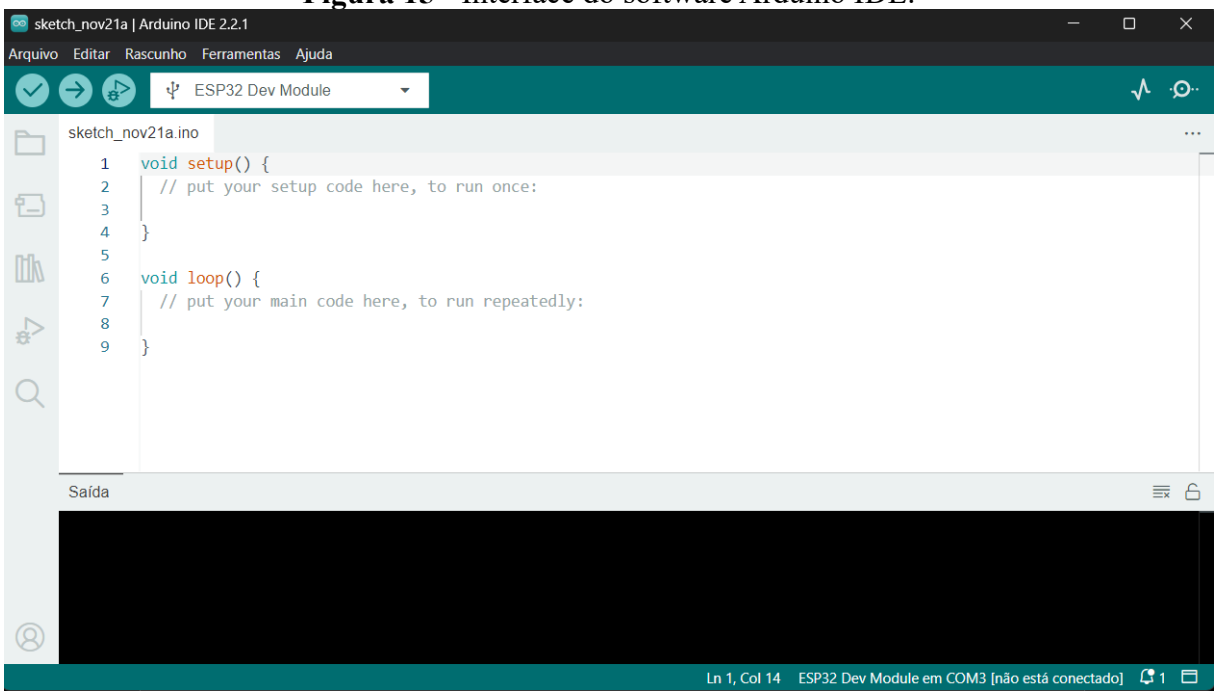
3.2.6 Arduino IDE

Para programar o ESP32 foi utilizado o Arduino IDE, onde o termo IDE significa *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, cuja interface está apresentada na Figura 15. Trata-se de um software de código aberto que permite desenvolver o código de programação utilizando as linguagens de programação C e C++ e então fazer upload desse código na placa. Embora o software tenha sido projetado para placas Arduino, é possível utilizar outras placas como por exemplo o ESP32.

Para o desenvolvimento do código de programação utilizado neste projeto foi necessário a inclusão das seguintes bibliotecas: WiFi.h, PubSubClient.h, DHT.h, DHT_U.h e ESP32Servo.

A biblioteca WiFi.h é utilizada para possibilitar que o ESP32 se conecte a uma rede WiFi. Com a finalidade de utilizar a comunicação MQTT, faz-se necessário a utilização da biblioteca PubSubClient, que contém funções que possibilitam o envio e recebimento de mensagens utilizando o protocolo MQTT. Já as bibliotecas DHT.h e DHT_U.h são utilizadas para que o sensor DHT11 consiga realizar a leitura dos valores da temperatura e da umidade do ambiente. Por último temos a biblioteca ESP32Servo.h que possibilita que o ESP32 controle servo motores.

Figura 15 - Interface do software Arduino IDE.



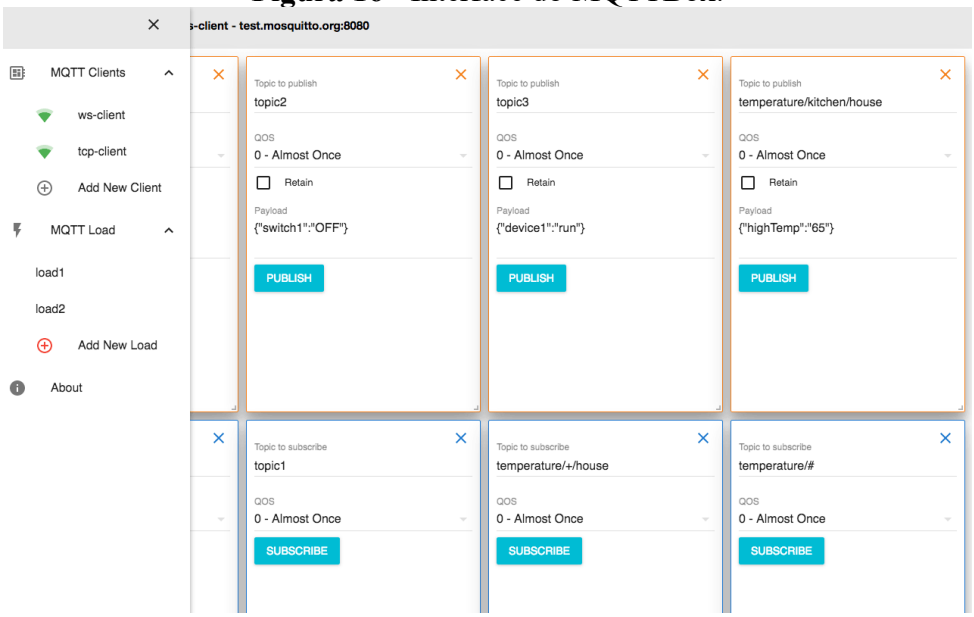
Fonte: Autoria própria.

3.2.7 MQTTBox

O MQTTBox é um software utilizado para facilitar o monitoramento da comunicação MQTT. Após criar um cliente e conectá-lo ao *broker*, permite a inscrição em tópicos, com isso é possível monitorar todo o histórico das mensagens publicadas nestes tópicos, publicar mensagens etc.

Conforme mostra a Figura 16, o MQTTBox permite a criação de vários clientes e a inscrição em vários tópicos ao mesmo tempo. Também permite selecionar o QoS da comunicação.

Figura 16 - Interface do MQTTBox.

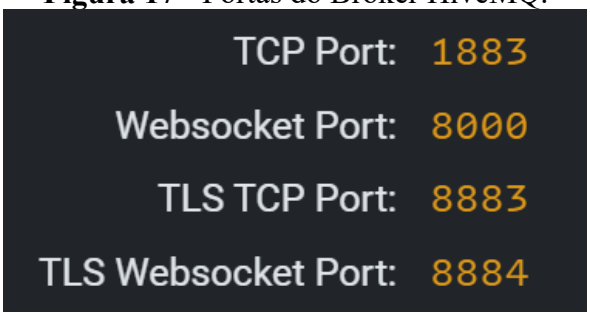


Fonte: HIVEMQ (2016).

3.2.8 Broker HiveMQ

Como visto anteriormente, para utilização do protocolo MQTT, faz-se necessário de um *broker*, portanto, neste trabalho será utilizado um *broker* gratuito disponibilizado pela HiveMQ. Este *broker* pode ser utilizado pelo endereço "broker.mqtt-dashboard.com", que possui as portas conforme mostra a Figura 17.

Figura 17 - Portas do Broker HiveMQ.

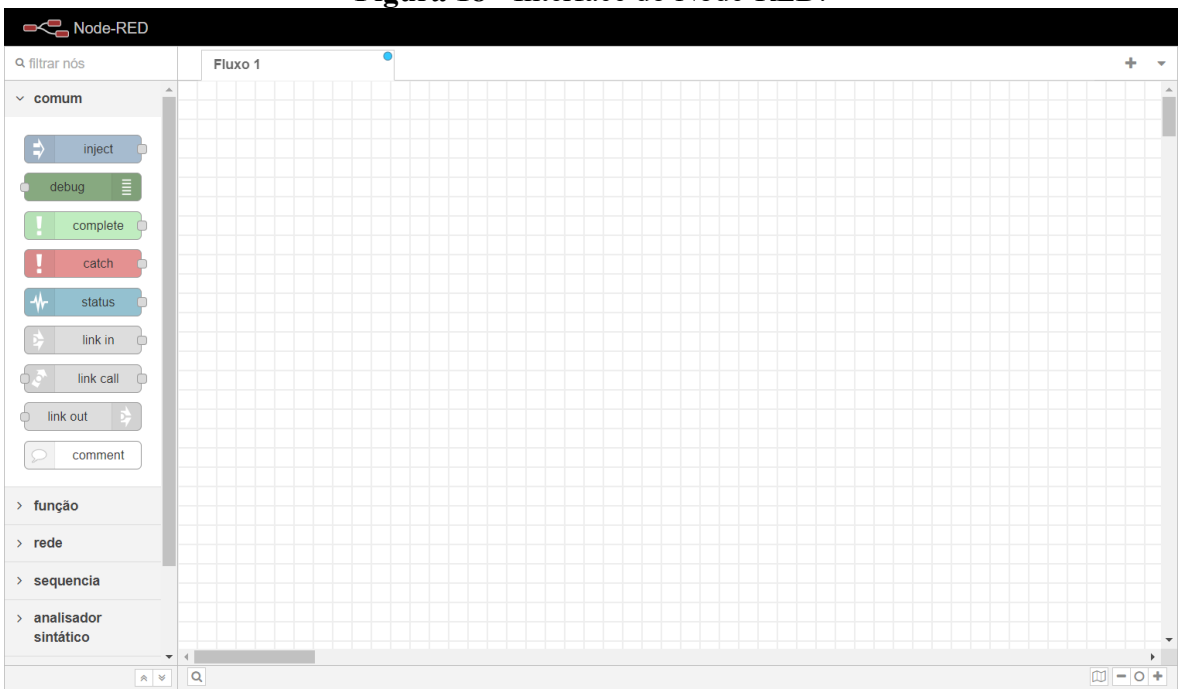


Fonte: Autoria própria.

3.2.9 Node-RED

Node-RED é uma ferramenta de programação que foi desenvolvida para conectar dispositivos de hardware, APIs e serviços online. Ele é baseado na ligação de nós que são blocos de códigos predefinidos, ou seja, sua utilização simplifica o desenvolvimento, já que não é necessário digitar nenhuma linha de código, basta apenas utilizar os nós para a criação de um fluxo. A edição do fluxo de nós pode ser realizada pelo próprio navegador e é bastante intuitiva, basta arrastar os nós desejados e realizar a conexão entre eles, conforme mostra a Figura 18.

Figura 18 - Interface do Node-RED.

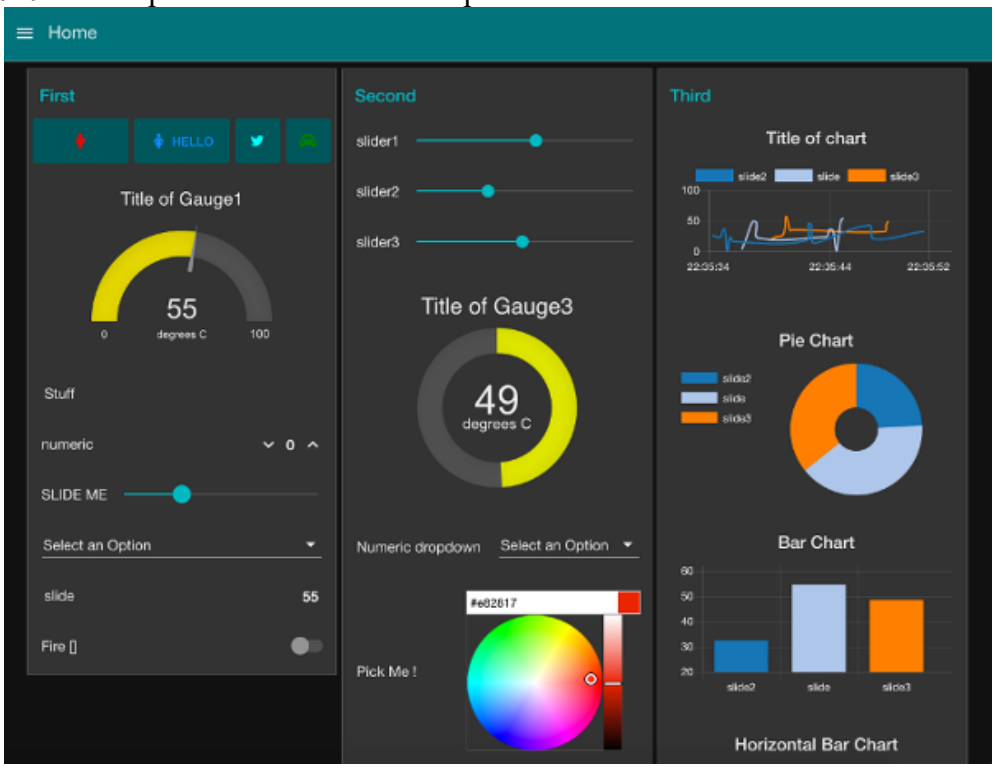


Fonte: Autoria própria.

Na Figura 18 é possível notar no lado esquerdo as diversas opções de nós que podem ser adicionadas ao fluxo, porém é possível incluir outros nós a partir da inclusão de bibliotecas. Para o desenvolvimento de uma interface de monitoramento do sistema de automação residencial, foi necessário incluir as bibliotecas node-red-dashboard, node-red-contrib-ui-led e node-red-contrib-remote. A biblioteca node-red-dashboard adiciona novos nós que permitem a criação de uma interface para monitoramento em tempo real, através de gráficos, textos e botões, ilustrados no exemplo da Figura 19. Já a biblioteca node-red-contrib-ui-led adiciona novos ícones indicadores de status LED para utilizar na interface. E a biblioteca node-red-

contrib-remote é responsável por incluir nós que possibilitam que a interface seja acessada pelo aplicativo Remote-RED no smartphone.

Figura 19 - Exemplo de interface criada a partir dos nós da biblioteca node-red-dashboard.



Fonte: Node-RED (2016).

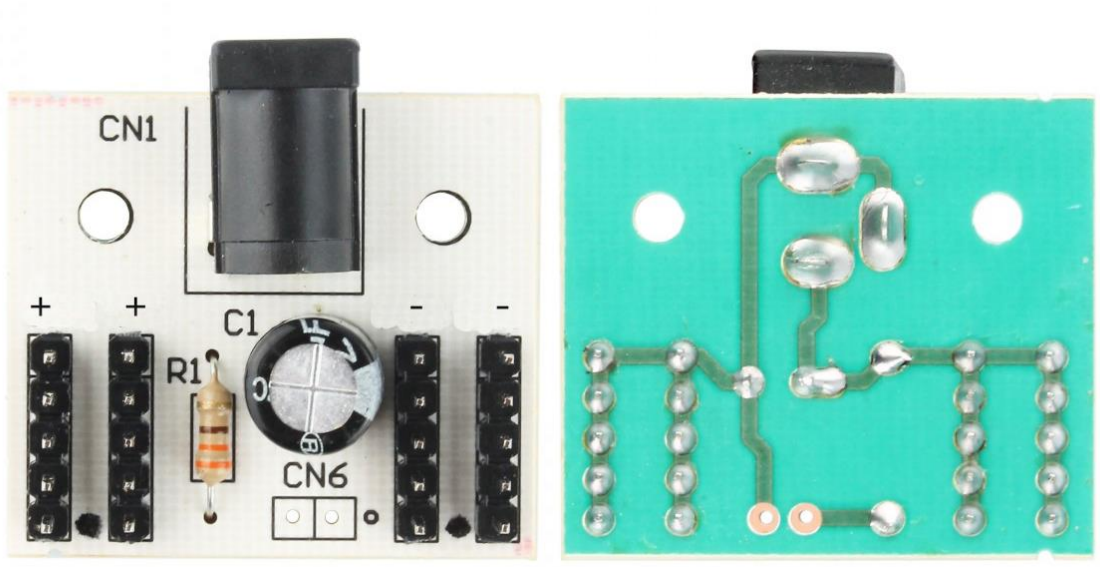
4 DESENVOLVIMENTO

O desenvolvimento do sistema de automação residencial pode ser dividido em três partes: circuito, código de programação e a interface do usuário. Adiante serão detalhados cada uma dessas partes.

4.1 Circuito

Como o ESP32 possui uma quantidade limitada de portas de alimentação, foi utilizado um módulo extensor de portas, cuja estrutura é composta por dez pinos Vcc e dez pinos *ground*, conforme mostra a Figura 20.

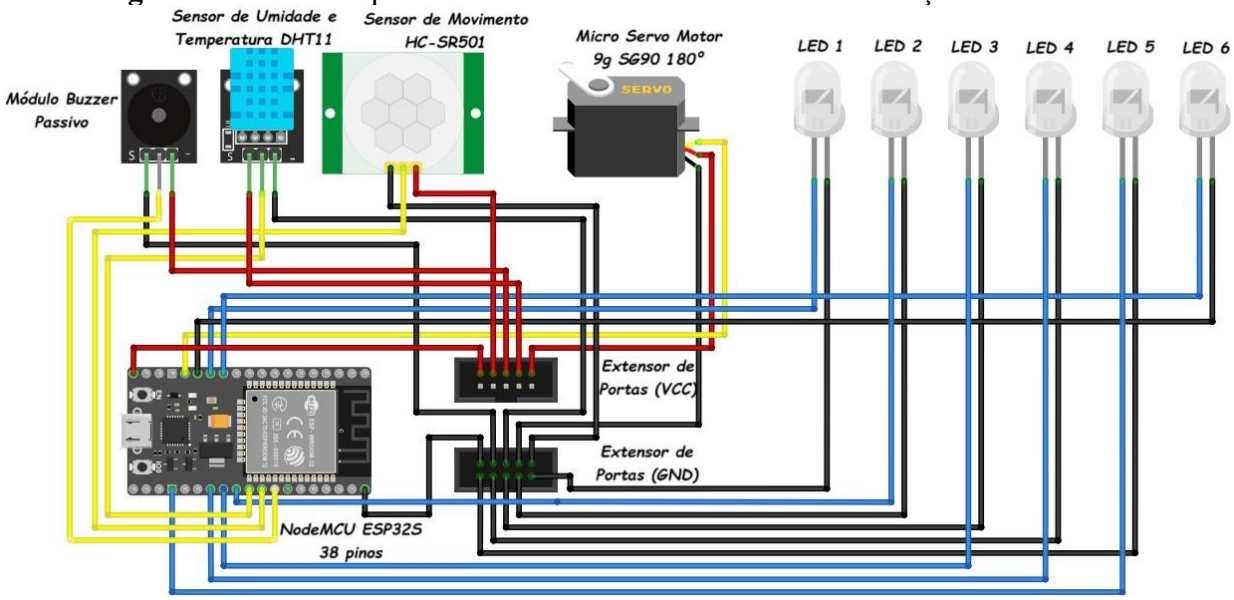
Figura 20 - Módulo extensor de portas.



Fonte: UsinaInfo ([ca. 2020b]).

Com o problema da quantidade de portas de alimentação solucionado, o circuito para desenvolvimento do sistema de automação residencial da Figura 21 foi esboçado. Ele é composto pelos seguintes componentes: microcontrolador ESP32 38 pinos, sensor de movimento HC-SR501, sensor de umidade e temperatura DHT11, micro servo motor SG90, módulo buzzer passivo, seis LEDs e o módulo extensor de portas da Figura 20.

Figura 21 - Circuito para o desenvolvimento sistema de automação residencial.



Fonte: Autoria própria.

Os seis LEDs representam cada uma das lâmpadas dos cômodos da residência e foram conectados da seguinte forma: os terminais positivos foram conectados diretamente em pinos GPIO do ESP32 e os terminais negativos foram conectados nas portas *ground* do extensor de portas, com exceção do terminal negativo do LED 6 da Figura 21 que foi conectado no pino *ground* do ESP32. Os LEDs utilizados são de cor branca e operam com uma tensão de 3 a 3,3 V, sendo assim, como os pinos digitais operam com nível lógico de 3,3 V, não foi necessário a utilização de resistores para impedir que os LEDs fossem danificados.

O sensor de temperatura e umidade, sensor de movimento, micro servo motor e o módulo buzzer passivo são todos compostos por três pinos, dessa forma, tiveram seus pinos Vcc conectado em terminais Vcc do extensor de portas, os pinos *ground* conectados em terminais *ground* do extensor de portas e os pinos de dados conectados em pinos do ESP32.

Por fim, para alimentação do ESP32, o pino Vin foi conectado em um dos terminais Vcc do extensor de portas e seu pino *ground* conectado em um dos terminais *ground* do extensor de portas. Conforme nota-se na Figura 20, o módulo extensor de portas é composto por uma entrada de conector modelo Jack P4, consequentemente, uma fonte de alimentação chaveada de 5V será conectada nesse componente, que irá realizar a alimentação de todo o circuito.

Todos os componentes foram fixados na maquete da residência, esta que será dividida entre os seguintes cômodos: garagem, cozinha, varanda, quarto, sala e sótão. Todos os seis

cômodos irão conter um LED, representando o sistema de iluminação da residência. O micro servo motor ficará na garagem para controlar a abertura e fechamento do portão. Na cozinha ficará instalado o sensor de movimento e o buzzer, que serão responsáveis por detectar movimento provenientes da varanda e acionar um alarme sonoro caso alarme esteja acionado. O sensor de umidade e temperatura realizará leituras da temperatura e umidade da cozinha.

4.2 Código de Programação

Para o desenvolvimento do código de programação no Arduino IDE, foi utilizado como base um código exemplo da biblioteca PubSubClient que apresenta linhas de instruções para estabelecer conexão entre o microcontrolador ESP8266 e o *broker*. Assim, partindo-se desse exemplo, foram realizadas alterações no código visando o sistema de automação residencial proposto. Na Apêndice A o código de programação desenvolvido pode ser visualizado na íntegra.

Figura 22 - Definições para conexão com a rede WiFi e com o broker.

```
uint32_t delayMS;
const char* ssid = "NOME_DA_REDE";
const char* password = "SENHA_DA_REDE";
const char* mqtt_server = "broker.mqtt-dashboard.com";
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
int value = 0;
```

Fonte: Autoria própria.

A Figura 22 exibe o trecho do código responsável por definir as configurações para estabelecer conexão com o *broker* e com a rede WiFi, portanto, é preciso inserir o nome e a senha da rede WiFi que o ESP32 irá se conectar, bem como o endereço do servidor MQTT. Como mostrado na Figura 23, o endereço do servidor MQTT será passado como parâmetro da função “client.setServer” que é responsável por estabelecer conexão com o servidor, além dele outro parâmetro também é passado, que é o número da porta que será utilizada, como visto anteriormente na Figura 17, o número da porta TCP do *broker* utilizado é a 1883. Na Figura 23

ainda se observa a inicialização da função “setup_wifi”, cujo finalidade é estabelecer a conexão entre o ESP32 e a rede WiFi. A função completa está apresentada na Figura 24.

Figura 23 - Inicialização da conexão com o servidor MQTT.

```
setup_wifi();
client.setServer(mqtt_server, 1883);
client.setCallback(callback);
```

Fonte: Autoria própria.

Utilizando a função “WiFi.begin” que recebe como parâmetros o nome e a senha da rede WiFi que foram definidas anteriormente no código da Figura 22, estabelece-se a conexão entre a placa e a rede WiFi. Quando a conexão for estabelecida com sucesso, é impresso um aviso no monitor serial.

Figura 24 - Inicialização da conexão com a rede WiFi.

```
void setup_wifi() {
  delay(10);
  Serial.println("");
  Serial.print("Conectando com ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

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

  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("IP: ");
  Serial.println(WiFi.localIP());
}
```

Fonte: Autoria própria.

Caso a conexão não seja estabelecida com sucesso, há a função “reconnect”, exibida na Figura 25, que será executada repetidamente enquanto a conexão não é estabelecida, durante esse processo um aviso será sempre impresso no monitor serial. No momento que a conexão é

finalmente estabelecida com sucesso, é realizada a inscrição no tópico “casa/publisher" e publicada uma mensagem no monitor serial avisando que a conexão foi estabelecida com sucesso.

Figura 25 - Função de reconnect.

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Aguardando conexão...");
    String clientId = "ESP32Client";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("Conectado");
      client.subscribe("casa/publisher");
    } else {
      Serial.print("Falhou, rc=");
      Serial.print(client.state());
      Serial.println("Tente novamente em 5s");
      delay(5000);
    }
  }
}
```

Fonte: Autoria própria.

Na Figura 23 também é inicializada a função “callback”, que é responsável por monitorar o recebimento de mensagens e a partir disso define a ação que será realizada. Uma parte dessa função está apresentada na Figura 26. Neste trecho da Figura 26 está definido que caso a mensagem recebida seja o caractere “S”, o pino designado para o LED do cômodo sala receberá nível lógico alto e será publicada uma mensagem no tópico “casa/sala” informando a ação realizada. Porém, caso o caractere seja “s”, o pino designado para o cômodo sala irá receber nível lógico baixo e será publicada uma mensagem no tópico “casa/sala”. Na sequência da função “callback”, que não está exibida na Figura 26, há a definição para os demais cômodos da residência.

Figura 26 - Trecho da função de callback.

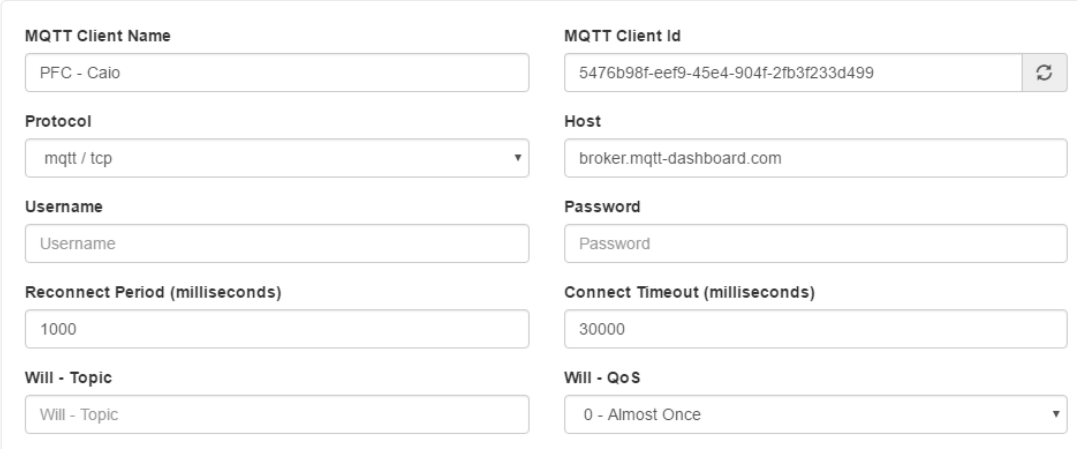
```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Mensagem recebida [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println("");
  if ((char)payload[0] == 'S') {
    digitalWrite(Sala, HIGH);
    snprintf (msg, MSG_BUFFER_SIZE, "A luz da sala está ligada");
    Serial.print("Publica mensagem: ");
    Serial.println(msg);
    client.publish("casa/sala", msg);
  }
  Serial.println("");
  if ((char)payload[0] == 's') {
    digitalWrite(Sala, LOW);
    snprintf (msg, MSG_BUFFER_SIZE, "A luz da sala está apagada");
    Serial.print("Publica mensagem: ");
    Serial.println(msg);
    client.publish("casa/sala", msg);
  }
```

Fonte: Autoria própria.

4.3 Configuração do MQTTBox

Utilizando o software MQTTBox, um cliente foi criado a partir das configurações da Figura 27. Foi atribuído um nome para o cliente, já o ID de cliente é um código identificador único para cada cliente, portanto o ID gerado pelo software não foi alterado. Em seguida foi definido o tipo de protocolo MQTT / TCP para se comunicar com o *broker*. Os campos de usuário e senha foram deixados em branco, pois como o *broker* utilizado é público, não havendo necessidade de identificação. O período de reconexão e o tempo limite de conexão foram mantidos no valor padrão do software. Para a configuração da mensagem de *Last Will* caso ocorra falha na conexão, foi definido um tópico padrão. Enquanto para a configuração de QoS foi definido o QoS 0, pois a prioridade é a rapidez na transmissão da mensagem e não há problema caso a mensagem seja perdida. As demais configurações para criação do cliente não foram alteradas, manteve-se no padrão.

Figura 27 - Configuração para criação de um cliente.



Fonte: Autoria própria.

Com o cliente já criado, foi realizada a inscrição nos seguintes tópicos:

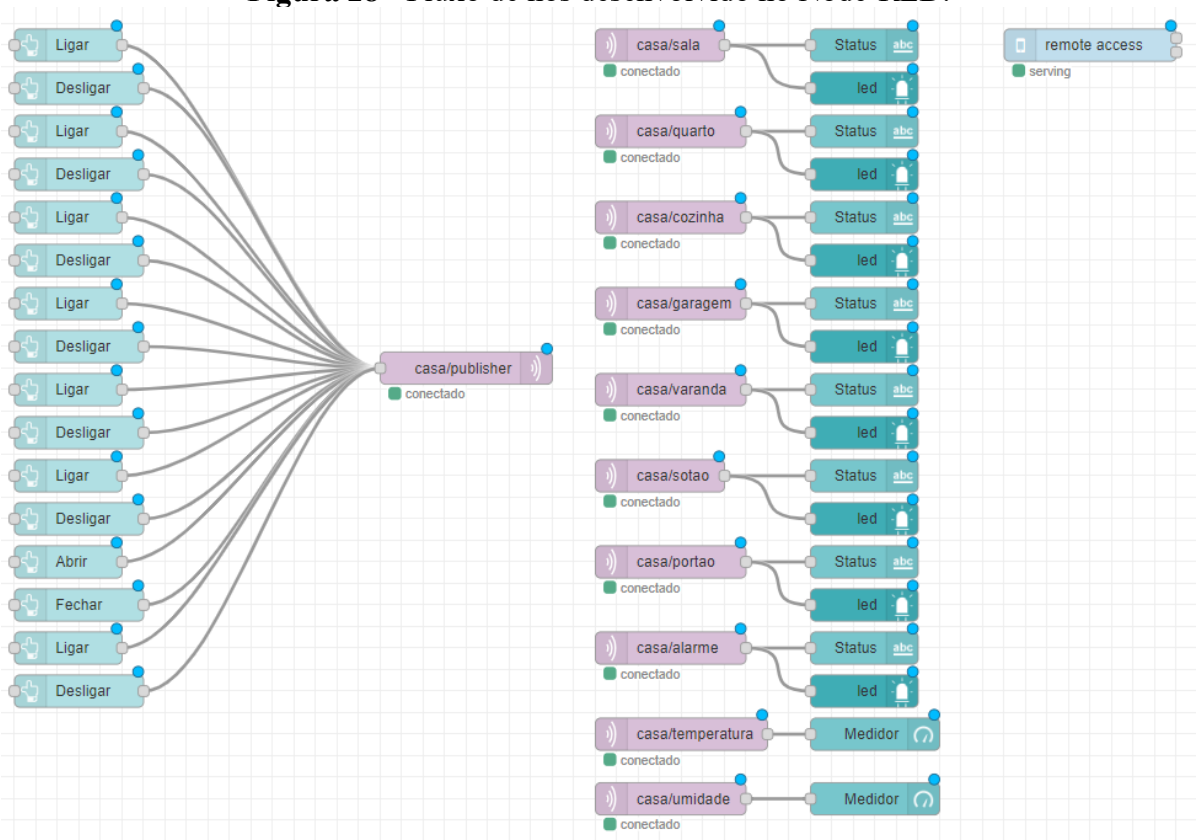
- casa/sala - tópico contendo mensagens sobre o estado do LED da sala;
- casa/quarto - tópico contendo mensagens sobre o estado do LED do quarto;
- casa/cozinha - tópico contendo mensagens sobre o estado do LED da cozinha;
- casa/garagem - tópico contendo mensagens sobre o estado do LED da garagem;
- casa/varanda - tópico contendo mensagens sobre o estado do LED da varanda;
- casa/sotao - tópico contendo mensagens sobre o estado do LED do sótão;
- casa/portao - tópico contendo mensagens sobre o estado do portão da garagem;
- casa/temperatura - tópico contendo mensagens com os valores da temperatura;
- casa/umidade - tópico contendo mensagens com os valores da umidade;
- casa/movimento - tópico contendo mensagens quando um movimento é detectado pelo sensor;
- casa/alarme - tópico contendo mensagens sobre o estado de acionamento do alarme;
- casa/publisher – tópico para o cliente MQTTBox publicar mensagem.

Portanto, “casa/publisher" é o tópico no qual o ESP32 estará inscrito para receber as mensagens e a partir do caractere recebido como mensagem, irá realizar uma ação como acender/apagar um LED, ou abrir/fechar o portão e adicionalmente irá publicar no devido tópico uma mensagem descrevendo a ação realizada.

4.4 Configuração do Node-RED

Para desenvolvimento da interface do sistema de automação residencial foi construído na ferramenta Node-RED o fluxo exibido na Figura 28. O fluxo pode ser dividido em três partes: a primeira parte composta pelo conjunto de nós localizados no lado esquerdo da Figura 28, a segunda parte composta pelo conjunto de nós localizados no meio da Figura 28, e a terceira parte composta somente pelo nó “remote access”.

Figura 28 - Fluxo de nós desenvolvido no Node-RED.

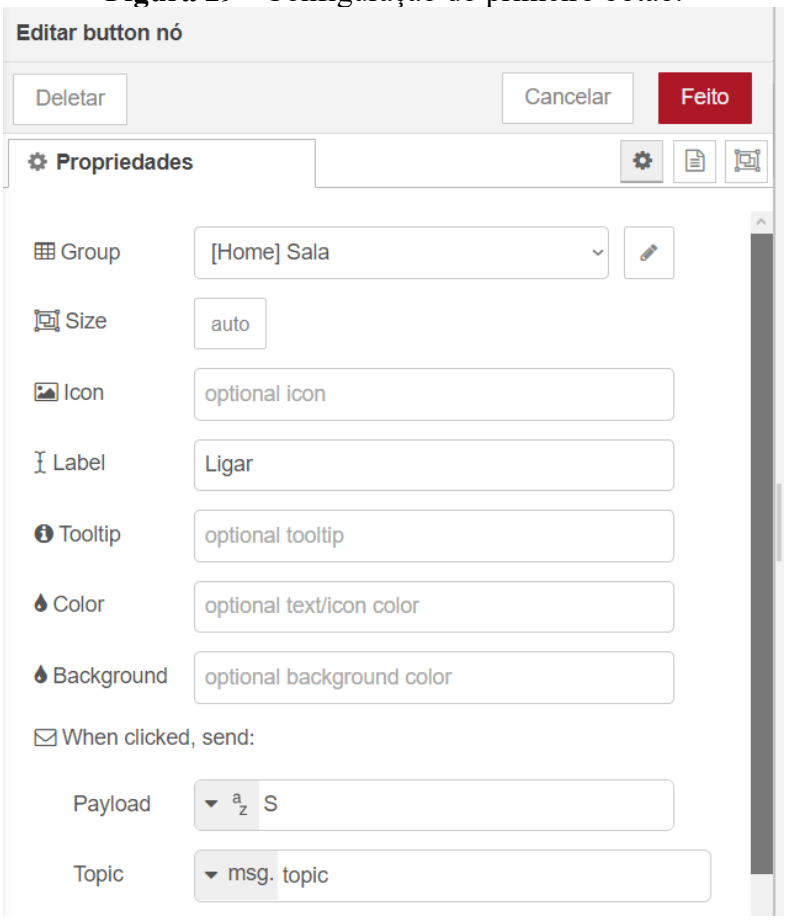


Fonte: Autoria própria.

A primeira parte do fluxo é responsável por exibir na interface os botões para ligar e desligar as lâmpadas e o alarme ou para abrir e fechar o portão da garagem, para isso, foram utilizados 16 nós do tipo “button” da biblioteca node-red-dashboard. A configuração aplicada para o primeiro botão está mostrada na Figura 29, onde inicialmente é criado um grupo, que neste primeiro caso foi “Sala”, pois este será o botão responsável por ligar a lâmpada da sala. Em seguida, na opção *label* foi digitado o texto que irá aparecer dentro do botão na interface. Na opção de *payload*, clicando no ícone de setas aparece uma lista de opções do tipo de

mensagem que será enviada quando esse botão ser pressionado, dentre as quais foi selecionado a opção cadeia de caracteres e foi digitado o caractere “S”. Por último, na lista de opções de *topic*, foi selecionado o tipo da mensagem como “msg” e foi digitado “topic”. A mesma configuração foi realizada para os outros 15 botões, alterando o nome do grupo para o nome do respectivo cômodo ou nome da função (no caso do portão e do alarme), alterando o texto do botão e alterando o caractere que será enviado quando o botão ser pressionado.

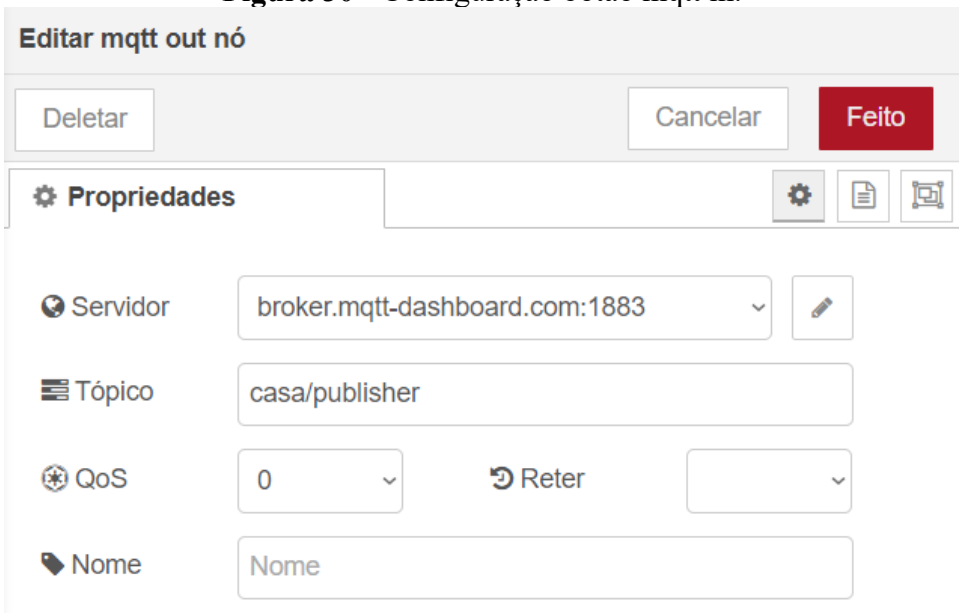
Figura 29 - Configuração do primeiro botão.



Fonte: Autoria própria.

Na Figura 28 observa-se que os 16 botões estão conectados no nó do tipo “mqtt in”, com a configuração exibida pela Figura 30. Na opção Servidor foi digitado o endereço do *broker* utilizado e a numeração da porta TCP. Em seguida na opção Tópico foi colocado o nome do tópico no qual serão publicados os caracteres quando os botões forem pressionados, que neste caso é o tópico “casa/publisher”, e na opção de QoS foi selecionado 0.

Figura 30 - Configuração botão mqtt in.

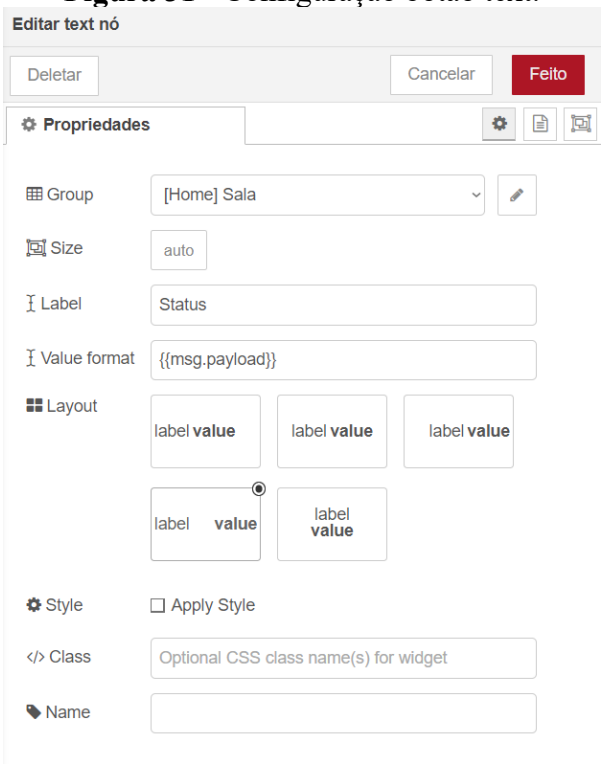


Fonte: Autoria própria.

A segunda parte do fluxo da Figura 28 é composta pelos nós que irão exibir na interface o status das lâmpadas, alarme, portão, e os valores da temperatura e umidade. Para isso, foram utilizados 10 nós do tipo “mqtt in”, que foram configurados da mesma forma que mostra Figura 29, somente com alteração do nome do tópico. Para os nós referentes aos tópicos dos cômodos, portão e alarme, cada um foi conectado em um nó do tipo “text” da biblioteca node-red-dashboard e um nó do tipo “LED” da biblioteca node-red-contrib-ui-led. Enquanto para os nós dos tópicos de temperatura e umidade foram conectados cada um em um nó do tipo “gauge” da biblioteca node-red-dashboard.

A Figura 31 exibe a configuração de um dos nós do tipo “text”. Inicialmente foi selecionado o grupo que já tinha sido criado anteriormente. Na opção “label” foi digitado “Status” para que seja exibido na dashboard e em “Value format” foi digitado {{msg.payload}}, para que seja exibido a mensagem que foi publicada no respectivo tópico conectado. Por último na opção “Layout” foi selecionado a maneira desejada para que sejam exibidas as informações. Para os demais nós do tipo “text” a configuração foi realizada dessa mesma forma, apenas com alteração do grupo.

Figura 31 - Configuração botão text.

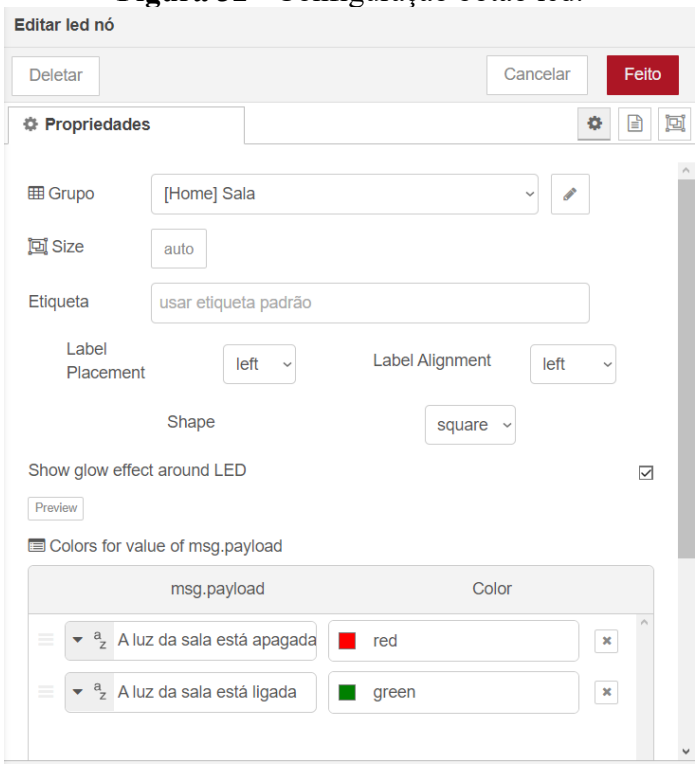


Fonte: Autoria própria.

Os nós do tipo “LED” foram configurados conforme mostra a Figura 32. Inicialmente foi selecionado o respectivo grupo e em seguida as opções de posicionamento e alinhamento do texto na interface. O formato do ícone do LED foi selecionado como quadrado e com a opção de exibir brilho ao redor do ícone. Logo abaixo foram definidas as cores que o ícone irá indicar com base na mensagem publicada no respectivo tópico conectado. Para os outros nós do tipo “LED” as configurações foram as mesmas, com alterações do grupo e do conteúdo da mensagem.

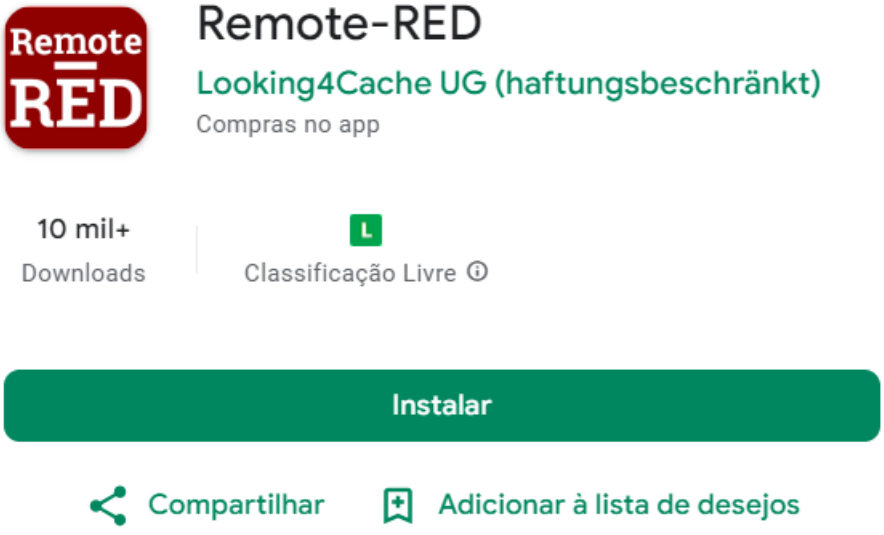
Na terceira parte do fluxo da Figura 28 temos somente o nó do tipo “remote-access” da biblioteca node-red-contrib-remote, que é utilizado para possibilitar que a interface desenvolvida possa ser acessada também pelo smartphone. Para isso é necessário que o aplicativo Remote-RED exibido na Figura 33 esteja instalado no smartphone. Acessando as configurações desse nó, existe uma opção que exibe um código de resposta rápida ou *Quick Response* (QR) que deve ser escaneado pelo smartphone ao abrir o aplicativo, dessa forma a conexão entre o Node-RED e o aplicativo será estabelecida e a interface poderá ser acessada utilizando esse aplicativo.

Figura 32 - Configuração botão led.



Fonte: Autoria própria.

Figura 33 - Aplicativo Remote-RED.



Fonte: Autoria própria.

5 RESULTADOS

Após montagem do circuito da Figura 21 na maquete da residência, a estrutura ficou da forma apresentada na Figura 34. Os cômodos ficaram divididos da seguinte forma: no andar térreo temos a garagem, a cozinha e a varanda, o primeiro andar é composto pelo quarto e pela sala, no último andar tem-se o sótão.

Figura 34 - Maquete da residência com todos os dispositivos instalados.

Fonte: Autoria própria.

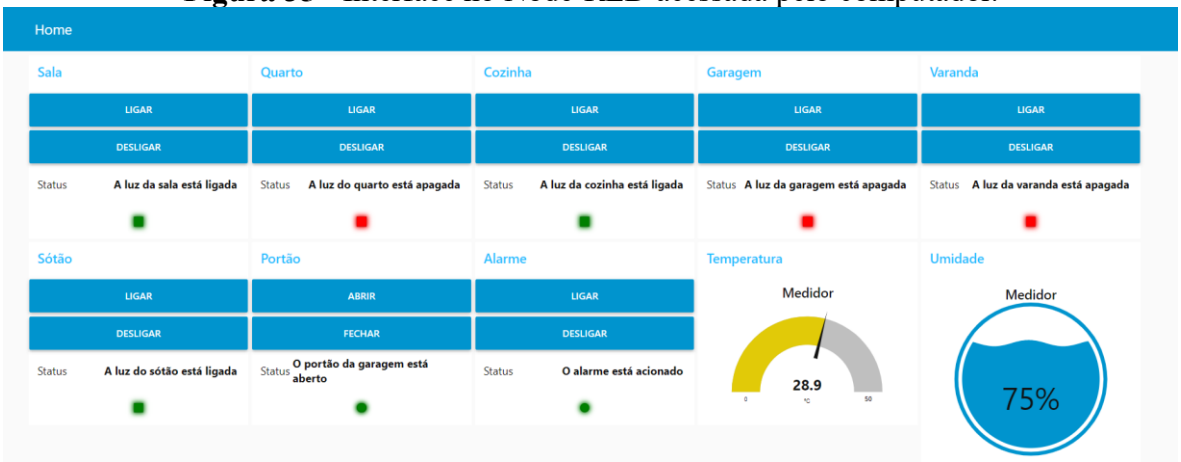
Com o código de programação compilado e carregado no ESP32, a fonte chaveada foi conectada na tomada para que o circuito entrasse em funcionamento. Foi realizado testes para acender e apagar todas as lâmpadas, abertura e fechamento do portão da garagem, acionamento do alarme e leitura da temperatura e umidade. O sistema apresentou funcional para todos os testes realizados, tanto para mensagens publicadas nos tópicos utilizando o MQTTBox, quanto para comandos enviados pela interface do Node-RED.

O nível de Qualidade de Serviço selecionado para todas as conexões entre cliente e broker foram o nível QoS 0, pois a prioridade neste caso é de que o processo de transmissão

das mensagens seja o mais rápido possível, sendo assim, mediu-se o intervalo de tempo entre o botão ser pressionado na interface e a respectiva ação ser executada e o tempo resultante foi de aproximadamente 0,13 segundos. Em relação ao portão da garagem, o processo de abertura/fechamento durou cerca de 7,5 segundos. Utilizando a interface pelo aplicativo no celular, é possível monitorar e controlar o sistema mesmo que o dispositivo esteja longe da residência, é necessário somente que o fluxo de nós do Node-RED esteja sendo executado. Para que fosse possível escutar o som do alarme de qualquer lugar da residência, foi necessário alterar a frequência do som emitido pelo módulo buzzer por meio do código de programação. Dessa forma, todos os requisitos funcionais do sistema citados anteriormente foram cumpridos.

Na Figura 35 é possível observar a interface de controle e monitoramento do Node-RED acessada pelo computador e a Figura 36 quando acessada pelo aplicativo Remote-Red no smartphone.

Figura 35 - Interface no Node-RED acessada pelo computador.



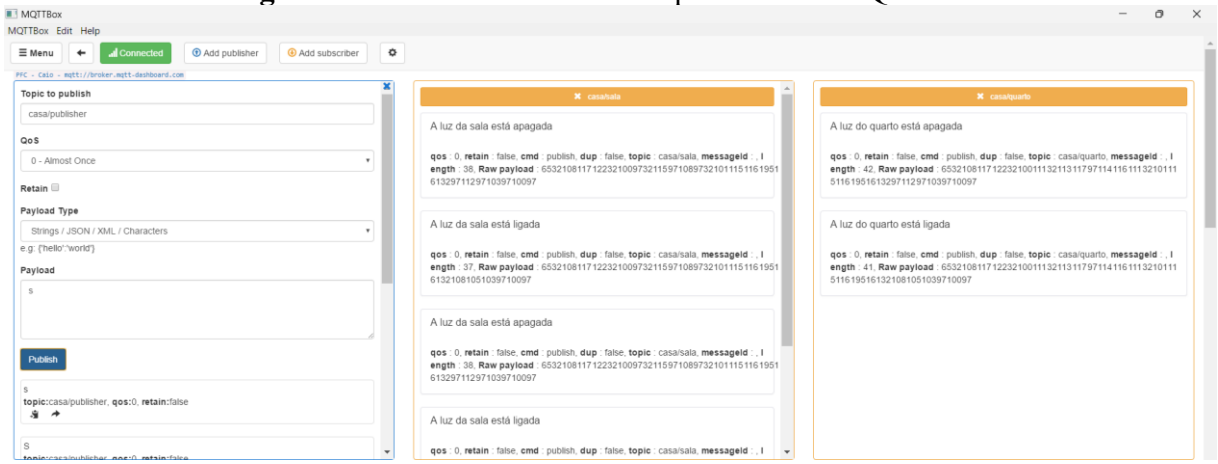
Fonte: Autoria própria.

Figura 36 - Interface no Node-RED acessada pelo celular.

Fonte: Autoria própria.

Na Figura 37 é mostrado o monitoramento das postagens nos tópicos utilizando o MQTTBox. No canto esquerdo a parte destinada para publicação de mensagens no tópico “casa/publisher", enquanto no centro e no canto direito são mostrados o monitoramento das mensagens publicadas nos tópicos, respectivamente, “casa/sala” e “casa/quarto”.

Figura 37 - Monitoramento dos tópicos com o MQTTBox.



Fonte: Autoria própria

6 CONCLUSÃO

Com a realização deste trabalho, foi possível a criação de um sistema de automação residencial que permite o controle e monitoramento de diferentes dispositivos instalados na residência. Geralmente os custos para implementação de automação residencial são altos e não são acessíveis para a maior parte das pessoas, porém neste trabalho, utilizou-se materiais com custos mais acessíveis. Mesmo utilizando materiais que não possuem custos elevados, o sistema desenvolvido atendeu todos os requisitos e apresentou uma boa performance para o tempo de acionamento dos dispositivos.

A maior dificuldade apresentada durante o desenvolvimento deste trabalho, foi a elaboração do código de programação, que necessitou um estudo aprofundado sobre o protocolo MQTT e sobre como utilizar as bibliotecas dos sensores e do micro servo motor.

Para trabalhos futuros há a possibilidade da utilização de mais tipos de sensores e dispositivos conectados na residência, de tal forma que o sistema de automação residencial conte com mais recursos. Além disso, seria interessante a utilização de um servidor na nuvem para armazenamento das mensagens publicadas e implementação do recurso de reconhecimento de comando pela voz.

REFERÊNCIAS

ABREU, T. M. B. **Edifícios Inteligentes – Soluções para gestão de climatização em instalação de domótica KNX Estudo de um caso**. 2013. Tese (Mestrado em Engenharia Industrial) – Instituto Politécnico de Bragança, Escola Superior de Tecnologia e Gestão, Bragança, 2013.

ALECRIM, Emerson. **O que é Internet das Coisas (IoT)**. [S. l.]: Infowester, 07 mar. 2016. Disponível em: https://www.infowester.com/iot.php. Acesso em: 31 de out de 2023.

ARDUINO. **DHT sensor library**. [S. l.]: Arduino, 01 mar. 2010. Disponível em: https://www.arduino.cc/reference/en/libraries/dht-sensor-library/. Acesso em: 21 de nov de 2023.

ARDUINO. **ESP32Servo**. [S. l.]: Arduino, 03 out. 2017. Disponível em: https://www.arduino.cc/reference/en/libraries/esp32servo/. Acesso em: 21 de nov de 2023.

ARDUINO. **PubSubClient**. [S. l.]: Arduino, 10 set. 2015a. Disponível em: https://www.arduino.cc/reference/en/libraries/pubsubclient/. Acesso em: 21 de nov de 2023.

ARDUINO. **WiFi**. [S. l.]: Arduino, 25 jun. 2015b. Disponível em: https://www.arduino.cc/reference/en/libraries/wifi/. Acesso em: 21 de nov de 2023.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The Internet of Things: A Survey. **Computer Networks Journal**, Cagliari, v. 54, n. 15, p. 2787-2805, 2010. Disponível em: https://www.researchgate.net/publication/222571757_The_Internet_of_Things_A_Survey. Acesso em: 31 de out de 2023.

BARROS, A. L. D. **Edifícios Inteligentes e a Domótica**: Proposta de um Projeto de Automação Residencial utilizando o protocolo X-10. 2010. Trabalho de Conclusão de Curso (Licenciatura em Engenharia de Construção Civil), Campus Universitário da Cidade da Praia, Universidade Jean Piaget de Cabo Verde, Praia, 2010.

BASSI, Alessandro. HORN, Geir. **Internet of Things in 2020: A roadmap for the future**. 2008. Disponível em: https://docbox.etsi.org/erm/Open/CERP%2020080609-10/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v1-1.pdf. Acesso em: 31 de out de 2023.

COSTA, Alex. **Desenvolvimento de sistema de automação residencial utilizando microcontrolador**. 2020. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica), Departamento de Engenharias e Ciência da Computação, Universidade Regional Integrada do Alto Uruguai e das Missões, Erechim, 2020.

ESPM. **Internet das coisas: a origem e o futuro da tecnologia que está mudando o mundo**. [S. l.]: Propmark, 25 ago. 2021. Disponível em: https://propmark.com.br/internet-das-coisas-a-origem-e-o-futuro-da-tecnologia-que-esta-mudando-o-mundo/. Acesso em: 30 de out de 2023.

ESPRESSIF SYSTEMS. **ESP32 Datasheet**. [S. l.]: Espressif Systems, 26 out. 2017. Disponível em: https://shopofthings.ch/wp-content/uploads/2017/11/esp32_datasheet_en.pdf. Acesso em: 04 de fev de 2024.

FERNANDES, Gisele. **Automação e muita tecnologia: novas tendências dos condomínios oferecem praticidade, segurança e sustentabilidade**. [S. l.]: Tiinside, 11 out. 2022. Disponível em: https://tiinside.com.br/11/10/2022/automacao-e-muita-tecnologia-novas-tendencias-dos-condominios-oferecem-praticidade-seguranca-e-sustentabilidade/. Acesso em: 30 de out de 2023.

GOYAL, Akshay. **Difference between ESP8266 and ESP32**. Nova Deli: Hnhcart, 11 jan. 2022. Disponível em: https://www.hnhcart.com/blogs/iot-rf/difference-between-esp8266-and-esp32. Acesso em: 14 de nov de 2023.

GETTYIMAGES. **Smart casa conectado**. 2016. 1 imagem digital, color. 37.3 Kb. Formato JPEG. Imagem creditada ao usuário Comomolas. Disponível em: https://media.gettyimages.com/id/623205612/pt/vetorial/elegante-home-ligado.jpg?s=612x612&w=0&k=20&c=5UMGOECp3MPZ4o2SHtYt9gcHn5pjyj6_eaQM9DOP53o=. Acesso em: 06 nov. de 2023.

HIVEMQ. **HiveMQ Public Broker MQTT Dashboard**. [S. l.]: Hivemq, 31 mar. 2015a. Disponível em: https://broker.mqtt-dashboard.com/. Acesso em: 21 de nov de 2023.

HIVEMQ. **Introducing the MQTT Protocol – MQTT Essentials: Part 1**. [S. l.]: Hivemq, 12 jan. 2015b. Disponível em: https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/. Acesso em: 08 de nov de 2023.

HIVEMQ. **MQTT Client, MQTT Broker, and MQTT Server Connection Establishment Explained – MQTT Essentials: Part 3**. [S. l.]: Hivemq, 17 jul. 2019. Disponível em: https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/. Acesso em: 08 de nov de 2023.

HIVEMQ. **MQTT Publish/Subscribe Architecture (Pub/Sub) – MQTT Essentials: Part 2**. [S. l.]: Hivemq, 19 jan. 2015c. Disponível em: https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/. Acesso em: 08 de nov de 2023.

HIVEMQ. **MQTTBox - MQTT Toolbox by HiveMQ**. [S. l.]: Hivemq, 08 ago. 2016. Disponível em: https://www.hivemq.com/article/mqtt-toolbox-mqttbox/. Acesso em: 21 de nov de 2023.

HIVEMQ. **What is MQTT Quality of Service (QoS) 0,1, & 2? – MQTT Essentials: Part 6**. [S. l.]: Hivemq, 16 fev. 2015d. Disponível em: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. Acesso em: 08 de nov de 2023.

IDC. **Predictions Brazil 2021**. [S. l.]: IDC, [2021]. Disponível em: https://www.idclatin.com/2021/events/02_04_br/ppt.pdf. Acesso em: 30 de out de 2023.

IOT ANALYTICS. **Insights Release: State of IoT 2023: number of connected IoT devices growing 16% to 16.7 billion globally**. Hamburgo: IoT Analytics, 24 mai. 2023. Disponível em: https://iot-analytics.com/wp/wp-content/uploads/2023/05/Insights-Release-State-of-IoT-2023-Number-of-connected-IoT-devices-growing-16-to-16.0-billion-globally.pdf. Acesso em: 31 de out de 2023.

JACTO. **Sensores na agricultura: 6 formas de adotar na sua propriedade**. [S. l.]: Jacto, 24 out. 2022. Disponível em: https://blog.jacto.com.br/sensores-na-agricultura/. Acesso em: 15 de nov de 2023.

JUNIOR, Fabiano; MINOTTI, Cristiano. Monitoramento do consumo de energia elétrica em tempo real. **RECIMA21 - Revista Científica Multidisciplinar - ISSN 2675-6218**, [S. l.], v. 1, n. 1, p. e210828, 2021. DOI: 10.47820/recima21.v1i1.828. Disponível em: https://recima21.com.br/index.php/recima21/article/view/828. Acesso em: 30 de out de 2023.

KOYANAGI, Fernando. **ESP32: Detalhes internos e pinagem**. [S. l.]: FernandoK, 06 mar. 2018. Disponível em: https://www.fernandok.com/2018/03/esp32-detalhes-internos-e-pinagem.html. Acesso em: 14 de nov de 2023.

LAST MINUTE ENGINEERS. **How HC-SR501 PIR Sensor Works & Interface It With Arduino**. [S. l.]: Last Minute Engineers, 26 jul. 2022a. Disponível em: https://lastminuteengineers.com/pir-sensor-arduino-tutorial/. Acesso em: 15 de nov de 2023.

LAST MINUTE ENGINEERS. **Interfacing DHT11 and DHT22 Sensors with Arduino**. [S. l.]: Last Minute Engineers, 25 nov. 2022b. Disponível em: https://lastminuteengineers.com/dht11-dht22-arduino-tutorial/. Acesso em: 15 de nov de 2023.

MAGRANI, Eduardo. **A Internet das Coisas**. 1. ed. Rio de Janeiro: FGV Editora, 2018.

MARTINS, Victor. **Automação residencial usando protocolo MQTT, Node-RED e mosquitto broker com ESP32 e ESP8266**. 2019. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação), Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia, 2019.

NODE-RED. **Node-RED**. [S. l.]: Node-RED, 25 ago. 2015. Disponível em: https://nodered.org/. Acesso em: 21 de nov de 2023.

NODE-RED. **node-red-contrib-remote**. [S. l.]: Node-RED, 26 jan. 2021. Disponível em: https://flows.nodered.org/node/node-red-contrib-remote. Acesso em: 21 de nov de 2023.

NODE-RED. **node-red-contrib-ui-led**. [S. l.]: Node-RED, 31 dez. 2018. Disponível em: https://flows.nodered.org/node/node-red-contrib-ui-led. Acesso em: 21 de nov de 2023.

NODE-RED. **node-red-dashboard**. [S. l.]: Node-RED, 25 jul. 2016. Disponível em: https://flows.nodered.org/node/node-red-dashboard. Acesso em: 21 de nov de 2023.

OLIVEIRA, Euler. **Conhecendo o NodeMCU-32S ESP32**. [S. l.]: Blog MasterWalker Shop, 30 mai. 2017. Disponível em: https://blogmasterwalkershop.com.br/embarcados/esp32/conhecendo-o-nodemcu-32s-esp32. Acesso em: 14 de nov de 2023.

OLIVEIRA, Jailson. **Arduino, ESP32 e ESP8266 – Comparação**. [S. l.]: XProjetos, 01 ago. 2019. Disponível em: https://xprojetos.net/arduino-esp32-e-esp8266-comparacao/. Acesso em: 14 de nov de 2023.

SACOMANO, João; et al. **Indústria 4.0: conceitos e fundamentos**. 1. ed. São Paulo: Blucher, 2018.

SAMORA, Helio. **Como surgiu e qual a utilidade da IoT**. [S. l.]: Indústria 4.0, 17 ago. 2020. Disponível em: https://www.industria40.ind.br/artigo/20246-como-surgiu-e-qual-a-utilidade-da-iot. Acesso em: 30 de out de 2023.

SANTOS, Jean. JUNIOR, Renato. **Sistema de automatização residencial de baixo custo controlado pelo microcontrolador ESP32 e Monitorado via smartphone**. 2019. Trabalho de Conclusão de Curso (Especialização em Automação Industrial), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2019.

SHELDON, Robert. **What is a sensor**. [S. l.]: TechTarget, 16 ago. 2022. Disponível em: https://www.techtarget.com/whatis/definition/sensor. Acesso em: 15 de nov de 2023.

SILVA, Saymon. **Sistema de monitoramento e controle de energia utilizando o microcontrolador ESP32**. 2021. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica), Universidade Federal Tecnológica Federal do Paraná, Ponta Grossa, 2021.

USINAINFO. **ESP32S NodeMCU Iot com WiFi e Bluetooth - 38 Pinos**. Santo Ângelo: UsinaInfo, [ca. 2020a]. Disponível em: https://www.usinainfo.com.br/nodemcu/esp32s-nodemcu-iot-com-wifi-e-bluetooth-38-pinos-5346.html. Acesso em: 14 de nov de 2023.

USINAINFO. **Extensor de Portas 0 a 6V 10 Saídas com Jack P4 - EPX10**. Santo Ângelo: UsinaInfo, [ca. 2020b]. Disponível em: https://www.usinainfo.com.br/regulador-de-tensao/extensor-de-portas-0-a-6v-10-saidas-com-jack-p4-epx10-6004.html. Acesso em: 14 de nov de 2023.

USINAINFO. **Micro Servo Motor 9g SG90 180° 1.6Kgf.cm de Posição**. Santo Ângelo: UsinaInfo, [ca. 2016]. Disponível em: https://www.usinainfo.com.br/servo-motores/micro-servo-motor-9g-sg90-180-16kgfcm-de-posicao-2299.html. Acesso em: 15 de nov de 2023.

USINAINFO. **Módulo Sensor de Umidade e Temperatura DHT11 + Jumpers**. Santo Ângelo: UsinaInfo, [ca. 2014]. Disponível em: https://www.usinainfo.com.br/sensor-de-umidade-arduino/modulo-sensor-de-umidade-e-temperatura-dht11-jumpers-2307.html. Acesso em: 15 de nov de 2023.

USINAINFO. **Sensor PIR de Movimento HC-SR501 100°**. Santo Ângelo: UsinaInfo, [ca. 2022]. Disponível em: https://www.usinainfo.com.br/sensor-de-movimento/sensor-pir-de-movimento-hc-sr501-100-2634.html. Acesso em: 15 de nov de 2023.

APÊNDICE A – CÓDIGO DE PROGRAMAÇÃO

```
// PFC - Caio Vinicius Domingues Kato

// Inclusão das bibliotecas
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <DHT_U.h>
#include <ESP32Servo.h>

//Definições dos pinos e das variáveis
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
#define Som 19
#define Alarme 2
#define DHTPIN 5
#define DHTTYPE DHT11
DHT_Unified dht(DHTPIN, DHTTYPE);
ESP32Servo motorgaragem;
#define pir 18
int pinStateCurrent = LOW;
int pinStatePrevious = LOW;
int alarmeState = LOW;
char Sala = 4;
char Quarto = 16;
char Cozinha = 15;
char Garagem = 14;
char Varanda = 12;
char Sotao = 17;

//Configurações para estabelecer conexão com o broker
uint32_t delayMS;
const char* ssid = "NOME_DA_REDE";
const char* password = "SENHA_DA_REDE";
const char* mqtt_server = "broker.mqtt-dashboard.com";
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
int value = 0;

// Laço de inicialização
void setup() {
  Serial.begin(115200);
  sensor_t sensor;
  motorgaragem.attach(13);
  dht.begin();

  delayMS = sensor.min_delay / 1000;

  pinMode(Sala, OUTPUT);
  pinMode(Quarto, OUTPUT);
  pinMode(Cozinha, OUTPUT);
  pinMode(Garagem, OUTPUT);
  pinMode(Varanda, OUTPUT);
  pinMode(Sotao, OUTPUT);
  pinMode(Som, OUTPUT);
  pinMode(Alarme, OUTPUT);
  pinMode(pir, INPUT);
```

```
  digitalWrite(Sala, LOW);
  digitalWrite(Quarto, LOW);
  digitalWrite(Cozinha, LOW);
  digitalWrite(Garagem, LOW);
  digitalWrite(Varanda, LOW);
  digitalWrite(Sotao, LOW);
  digitalWrite(Som, LOW);
  digitalWrite(Alarme, LOW);

  motorgaragem.write(117, 60);

  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

// Laço de configuração WiFi
void setup_wifi() {
  delay(10);
  Serial.println("");
  Serial.print("Conectando com ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

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

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

// Laço de configuração callback
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Mensagem recebida [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println("");
  if ((char)payload[0] == 'S')
    digitalWrite(Sala, HIGH);
    snprintf (msg, MSG_BUFFER_SIZE, "A luz da sala está ligada");
    Serial.print("Publica mensagem: ");
    Serial.println(msg);
    client.publish("casa/sala", msg);
  }
  Serial.println("");
  if ((char)payload[0] == 's') {
    digitalWrite(Sala, LOW);
    snprintf (msg, MSG_BUFFER_SIZE, "A luz da sala está apagada");
    Serial.print("Publica mensagem: ");
    Serial.println(msg);
    client.publish("casa/sala", msg);
  }
```

```
Serial.println("");
if ((char)payload[0] == 'Q') {
  digitalWrite(Quarto, HIGH);
  snprintf (msg, MSG_BUFFER_SIZE, "A luz do quarto está ligada");
  Serial.print("Publica mensagem: ");
  Serial.println(msg);
  client.publish("casa/quarto", msg);
}
Serial.println("");
if ((char)payload[0] == 'q') {
  digitalWrite(Quarto, LOW);
  snprintf (msg, MSG_BUFFER_SIZE, "A luz do quarto está apagada");
  Serial.print("Publica mensagem: ");
  Serial.println(msg);
  client.publish("casa/quarto", msg);
}
Serial.println("");
if ((char)payload[0] == 'C') {
  digitalWrite(Cozinha, HIGH);
  snprintf (msg, MSG_BUFFER_SIZE, "A luz da cozinha está ligada");
  Serial.print("Publica mensagem: ");
  Serial.println(msg);
  client.publish("casa/cozinha", msg);
}
Serial.println("");
if ((char)payload[0] == 'c') {
  digitalWrite(Cozinha, LOW);
  snprintf (msg, MSG_BUFFER_SIZE, "A luz da cozinha está apagada");
  Serial.print("Publica mensagem: ");
  Serial.println(msg);
  client.publish("casa/cozinha", msg);
}
Serial.println("");
if ((char)payload[0] == 'G') {
  digitalWrite(Garagem, HIGH);
  snprintf (msg, MSG_BUFFER_SIZE, "A luz da garagem está ligada");
  Serial.print("Publica mensagem: ");
  Serial.println(msg);
  client.publish("casa/garagem", msg);
}
Serial.println("");
if ((char)payload[0] == 'g') {
  digitalWrite(Garagem, LOW);
  snprintf (msg, MSG_BUFFER_SIZE, "A luz da garagem está apagada");
  Serial.print("Publica mensagem: ");
  Serial.println(msg);
  client.publish("casa/garagem", msg);
}
Serial.println("");
if ((char)payload[0] == 'V') {
  digitalWrite(Varanda, HIGH);
  snprintf (msg, MSG_BUFFER_SIZE, "A luz da varanda está ligada");
  Serial.print("Publica mensagem: ");
  Serial.println(msg);
  client.publish("casa/varanda", msg);
}
Serial.println("");
if ((char)payload[0] == 'v') {
  digitalWrite(Varanda, LOW);
  snprintf (msg, MSG_BUFFER_SIZE, "A luz da varanda está apagada");
  Serial.print("Publica mensagem: ");
```

```
    Serial.println(msg);
    client.publish("casa/varanda", msg);
  }
  Serial.println("");
  if ((char)payload[0] == 'P') {
    digitalWrite(Sotao, HIGH);
    snprintf (msg, MSG_BUFFER_SIZE, "A luz do sótão está ligada");
    Serial.print("Publica mensagem: ");
    Serial.println(msg);
    client.publish("casa/sotao", msg);
  }
  Serial.println("");
  if ((char)payload[0] == 'p') {
    digitalWrite(Sotao, LOW);
    snprintf (msg, MSG_BUFFER_SIZE, "A luz do sótão está apagada");
    Serial.print("Publica mensagem: ");
    Serial.println(msg);
    client.publish("casa/sotao", msg);
  }
  Serial.println("");
  if ((char)payload[0] == 'X') {
    motorgaragem.write(0, 60);
    snprintf (msg, MSG_BUFFER_SIZE, "O portão da garagem está aberto");
    Serial.print("Publica mensagem: ");
    Serial.println(msg);
    client.publish("casa/portao", msg);
  }
  Serial.println("");
  if ((char)payload[0] == 'x') {
    motorgaragem.write(117, 60);
    snprintf (msg, MSG_BUFFER_SIZE, "O portão da garagem está fechado");
    Serial.print("Publica mensagem: ");
    Serial.println(msg);
    client.publish("casa/portao", msg);
  }
  Serial.println("");
  if ((char)payload[0] == 'Z') {
    digitalWrite(Alarme, HIGH);
    snprintf (msg, MSG_BUFFER_SIZE, "O alarme está acionado");
    Serial.print("Publica mensagem: ");
    Serial.println(msg);
    client.publish("casa/alarme", msg);
  }
  Serial.println("");
  if ((char)payload[0] == 'z') {
    digitalWrite(Alarme, LOW);
    snprintf (msg, MSG_BUFFER_SIZE, "O alarme não está acionado");
    Serial.print("Publica mensagem: ");
    Serial.println(msg);
    client.publish("casa/alarme", msg);
  }
  Serial.println("");
}

// Função para configurar o som emitido pelo buzzer
void myTone(byte pin, uint16_t frequency, uint16_t duration) {
  unsigned long startTime=millis();
  unsigned long halfPeriod= 1000000L/frequency/2;
  pinMode(pin,OUTPUT);
  while (millis()-startTime< duration)
  {
```

```
    digitalWrite(pin,HIGH);
    delayMicroseconds(halfPeriod);
    digitalWrite(pin,LOW);
    delayMicroseconds(halfPeriod);
  }
  pinMode(pin,INPUT);
}

// Laço de reconexão
void reconnect() {
  while (!client.connected()) {
    Serial.print("Aguardando conexão...");
    String clientId = "ESP32Client";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("Conectado");
      client.subscribe("casa/publisher");
    } else {
      Serial.print("Falhou, rc=");
      Serial.print(client.state());
      Serial.println("Tente novamente em 5s");
      delay(5000);
    }
  }
}

// Laço de repetição
void loop() {

  // Condição para que o alarme seja acionado
  pinStatePrevious = pinStateCurrent;
  pinStateCurrent = digitalRead(pir);
  alarmeState = digitalRead(Alarme);

  if (pinStatePrevious == LOW && pinStateCurrent == HIGH && alarmeState ==
HIGH){
    client.publish("casa/movimento", "Movimento detectado");
    Serial.println("Movimento detectado");
    myTone(19,1000, 2000);
    delay(5000);
  }
  else {
    client.publish("casa/movimento", "Movimento não detectado");
  }

  // Leitura e impressão da temperatura e umidade
  delay(delayMS);
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (isnan(event.temperature)) {
    Serial.println(F("Erro na leitura da temperatura!"));
  }
  else {
    Serial.print(F("Temperatura: "));
    Serial.print(event.temperature);
    Serial.println(F("°C"));
    sprintf(msg,"%f",event.temperature);
    client.publish("casa/temperatura", msg);
  }
  dht.humidity().getEvent(&event);
  if (isnan(event.relative_humidity)) {
```

```
    Serial.println(F("Erro na leitura da umidade!"));
  }
  else {
    Serial.print(F("Umidade: "));
    Serial.print(event.relative_humidity);
    Serial.println(F("%"));
    sprintf(msg,"%f",event.relative_humidity);
    client.publish("casa/umidade", msg);
  }
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```