INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS – *CAMPUS* FORMIGA
BACHARELADO EM ENGENHARIA ELÉTRICA

Guilherme Eduardo Fonseca

DESENVOLVIMENTO DE SISTEMA PARA AUTOMAÇÃO RESIDENCIAL BASEADO EM IOT COM PROTOCOLO DE MENSAGENS MQTT

FORMIGA – MG
2022

GUILHERME EDUARDO FONSECA

DESENVOLVIMENTO DE SISTEMA PARA AUTOMAÇÃO RESIDENCIAL BASEADO EM IOT COM PROTOCOLO DE MENSAGENS MQTT

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga como requisito para obtenção do título de bacharel em Engenharia Elétrica.

Orientador: Prof. Me. Marco Antônio Silva Pereira

FORMIGA – MG

2022

Fonseca, Guilherme Eduardo
F676d Desenvolvimento de Sistema para Automação Residencial baseado em IOT com protocolo de mensagens MQTT
/ Guilherme Eduardo Fonseca -- Formiga : IFMG, 2022.
68p. : il.

Orientador: Prof. Me. Marco Antonio Silva Pereira
Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Internet das coisas. 2. Microcontroladores. 3. MQTT. 4. Domótica. 5. ESP32. I. Pereira, Marco Antonio Silva. II. Título.

CDD 621.3

Ficha catalográfica elaborada pela Bibliotecária Msc. Simoni Júlia da Silveira

GUILHERME EDUARDO FONSECA

DESENVOLVIMENTO DE SISTEMA PARA AUTOMAÇÃO RESIDENCIAL BASEADO EM IOT COM PROTOCOLO DE MENSAGENS MQTT

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga como requisito para obtenção do título de bacharel em Engenharia Elétrica.

Orientador: Prof. Me. Marco Antônio Silva Pereira

Avaliado em 18 de janeiro de 2022.

Nota: 98,0

BANCA EXAMINADORA

Marco Antônio Silva Pereira

Prof. Me. Marco Antônio Silva Pereira (Orientador)

Prof. Dr. Gustavo Lobato Campos

Patrick Santos de Oliveira

Prof. Dr. Patrick Santos de Oliveira

AGRADECIMENTOS

Primeiramente gostaria de agradecer os meus pais Eduardo e Gislaine por todo o apoio durante esta caminhada e por todo amor e carinho.

Agradeço também a minha irmã, Maria Luiza, por sempre estar presente e por todo o carinho.

Ao meu orientador, Prof. Marco, pela disponibilidade, por ajudar sempre que foi preciso, pelas orientações e conselhos.

Aos professores e funcionários do IFMG *Campus* Formiga pelo ótimo trabalho, tornando o instituto um lugar tão excepcional.

A todos os amigos que estiveram juntos durante a graduação, em especial a todos que sempre ajudaram quando foi preciso.

Por fim, agradeço a todos os outros que tiveram alguma contribuição ao longo dessa jornada e na realização deste trabalho.

“Se eu vi mais longe, foi por estar sobre ombros de gigantes.” (Isaac Newton)

RESUMO

Existem no mercado, atualmente, inúmeros dispositivos com capacidade de processamento de dados, inclusive com recursos de comunicação com a rede de internet, os quais representam os alicerces para o conceito de Internet das Coisas (IoT). A IoT possibilita várias aplicações que podem ter grande utilidade no cotidiano das pessoas, como é o caso da Domótica que utiliza os conceitos de IoT para a gestão de recursos habitacionais, podendo tornar uma residência automatizada. Neste contexto, o presente trabalho tem como objetivo apresentar o desenvolvimento de um protótipo para um sistema de automação residencial, com base nos conceitos de Internet das Coisas, verificando-se a viabilidade técnica e realizando uma análise de custos do projeto. Este sistema proposto apresenta, dentre outros, recursos para o acionamento de luzes, controle de temperatura e gerenciamento de segurança. Sendo assim, há uma integração dos dispositivos da residência, visando um melhor controle do usuário sobre o sistema. Para o desenvolvimento do projeto foram utilizados dois microcontroladores, os quais são compostos, basicamente, por processador, memória e módulos de entradas/saídas, assim como os sensores, atuadores e sinalizadores específicos para cada recurso a ser implementado. Foi utilizado o microcontrolador ESP32 e o ESP32-CAM, pelo baixo custo de aquisição e baixo consumo energético, além de que ambos possuem módulo Wi-Fi integrado, o que o torna uma das melhores opções disponíveis no mercado para este tipo de aplicação, além disso o ESP32-CAM é uma versão que ainda possui um módulo de câmera que possibilita criar um sistema de vigilância. A comunicação entre os dispositivos é baseada principalmente no protocolo de mensagens *Message Queuing Telemetry Transport* (MQTT), que foi projetado para ter um baixo consumo de banda e apresenta suporte para comunicação assíncrona, já que uma comunicação síncrona poderia gerar problemas devido a grande quantidade de dispositivos conectados na rede. Além deste protocolo também foi utilizado *o Hypertext Transfer Protocol* (HTTP) para o *streaming* de vídeo utilizado no sistema de vigilância, sendo este necessário visto o maior consumo de banda da aplicação. Por fim, os resultados provenientes das implementações realizadas são apresentados, com detalhamento da operação de cada recurso proposto.

Palavras Chave: Internet das Coisas, Microcontrolador, MQTT, Domótica, ESP32.

ABSTRACT

Currently, there are numerous devices on the market with data processing capacity, including communication resources with the internet network, which represent the foundations for the concept of the Internet of Things (IoT). IoT enables several applications that can be very useful in people's daily lives, such as Domotics, which uses IoT concepts for the management of housing resources, making it an automated home. In this context, this work aims to present the development of a prototype for a home automation system, based on the concepts of Internet of Things, verifying the technical feasibility and performing a cost analysis of the project. This proposed system features, among others, features for activating lights, temperature control and security management. Thus, there is an integration of home devices, aiming at better user control over the system. For the development of the project, two microcontrollers were used, which are basically composed of processor, memory and input/output modules, as well as specific sensors, actuators and flags for each resource to be implemented. The ESP32 and ESP32-CAM microcontrollers were used, due to their low acquisition cost and low energy consumption, in addition to both having an integrated Wi-Fi module, which makes them one of the best options available on the market for this type of application. Furthermore, the ESP32-CAM is a version that also has a camera module that makes it possible to create a surveillance system. Communication between devices is mainly based on the Message Queuing Telemetry Transport (MQTT) messaging protocol, which was designed to have a low bandwidth consumption and supports asynchronous communication, as a synchronous communication could cause problems due to the large amount of devices connected to the network. In addition to this protocol, the Hypertext Transfer Protocol (HTTP) was also used for video streaming used in the surveillance system, which is necessary given the application's higher bandwidth consumption. Finally, the results from the implementations carried out are presented, detailing the operation of each proposed resource.

Keyword: Internet of Things, Microcontroller, MQTT, Domotics, ESP32.

LISTA DE ILUSTRAÇÕES

Figura 1: Blocos da Internet das coisas. 18
Figura 2: Exemplos de automação no ambiente residencial. 19
Figura 3: Modelo de publicação e assinatura do protocolo MQTT. 21
Figura 4: Sensor DHT11. 22
Figura 5: Sensor HC-SR501. 23
Figura 6: Sensor MQ-2. 23
Figura 7: Buzzer Ativo BP09. 24
Figura 8: Sensor de luminosidade. 25
Figura 9: Sensor de umidade do solo. 25
Figura 10: Micro Servo TowerPro 9g SG90. 26
Figura 11: USB-Serial Ttl PL2303. 27
Figura 12: Módulo relé de oito canais. 27
Figura 13: Válvula Solenóide. 28
Figura 14: Arduino IDE. 29
Figura 15: Pinout ESP32. 30
Figura 16: Pinout ESP32-CAM. 31
Figura 17: ESP32-CAM conectado ao módulo FTDI. 31
Figura 18: Arquitetura de comunicação do projeto. 32
Figura 19: Fluxograma do funcionamento do sistema de iluminação automática. 34
Figura 20: Fluxograma do funcionamento do sistema de irrigação automática. 35
Figura 21: Fluxograma do funcionamento do controle da persiana. 36
Figura 22: Fluxograma do funcionamento do sistema de alarme. 37
Figura 23: Fluxograma do funcionamento do sistema de detecção de gás. 38
Figura 24: Fluxograma do funcionamento do sistema de climatização. 39
Figura 25: Protótipo final. 40
Figura 26: Ligações da implementação do protótipo. 41
Figura 27: Comunicação entre os dispositivos. 42
Figura 28: Gravação do código no ESP32-CAM. 43
Figura 29: Saída serial da Arduino IDE com ESP32-CAM. 43
Figura 30: Streaming de imagem do ESP32-CAM. 44
Figura 31: Dashboard no MQTT DASH. 45
Figura 32: Ícone para adicionar novo elemento a dashboard. 46
Figura 33: Opções de elementos no MQTT DASH. 47
Figura 34: Adicionando o elemento de temperatura do ambiente. 48
Figura 35: Fluxograma da lógica do protocolo nos microcontroladores. 49

LISTA DE TABELAS

Tabela 1: Comparação dos Microcontroladores. 20
Tabela 2: Significado da leitura dos sensores. 33
Tabela 3: Pinos utilizados para conectar os dispositivos. 42
Tabela 4: Publicações MQTT dos microcontroladores. 45
Tabela 5: Inscrições MQTT dos microcontroladores. 46
Tabela 6: Custo dos materiais utilizados no protótipo. 50

LISTA DE SIGLAS E ABREVIATURAS

ADC – Conversão de Analógico para Digital.

CC – Corrente Contínua.

CPU – Central Processing Unit.

HTTP – Hypertext Transfer Protocol

IDE – Integrated Development Environment.

IoT – Internet of Things.

IP – Internet Protocol.

LDR – Light Dependent Resistor.

M2M – Machine-to-Machine.

MQTT – Message Queue Telemetry Transport.

NTC – Negative Temperature Coefficient.

PWM – Pulse Width Modulation.

RX – Receive Data.

TX – Data Transfer.

SUMÁRIO

FICHA CATALOGRÁFICA **3**
AGRADECIMENTOS **5**
RESUMO **7**
ABSTRACT **8**
LISTA DE ILUSTRAÇÕES **9**
LISTA DE TABELAS **10**
LISTA DE SIGLAS E ABREVIATURAS **11**
SUMÁRIO **12**
1. INTRODUÇÃO **14**
1.1. Motivação **15**
1.2. Objetivos gerais e específicos **16**
2. REFERENCIAL TEÓRICO **17**
2.1. Internet das Coisas **17**
2.1.1. Domótica ***18***
2.2. Microcontroladores **19**
2.3. Protocolos de comunicação **20**
2.3.1. Message Queue Telemetry Transport (MQTT) ***21***
2.4. Sensores e dispositivos eletrônicos **22**
2.5. Arduino IDE **28**
3. METODOLOGIA **30**
3.1. ESP32 ..... **30**
3.2. ESP32-CAM **30**
3.3. Arquitetura do projeto **32**
3.4. Operação dos recursos de automação residencial **33**
3.4.1. Sistema de iluminação automático ***33***
3.4.2. Sistema de irrigação automático ***34***
3.4.3. Controle de persianas ***35***
3.4.4. Sistema de alarme ***36***
3.4.5. Sistema de detecção de vazamento de gás ***37***
3.4.6. Sistema de climatização automático ***38***
4. RESULTADOS E DISCUSSÕES **40**

4.1. Ligação e comunicação entre sensores e microcontroladores 40
4.2. Sistema de vigilância 43
4.3. Dashboard no MQTT DASH 44
4.4. Análise de custos e viabilidade técnica 49
5. CONCLUSÕES 52
REFERÊNCIAS BIBLIOGRÁFICAS 53
ANEXOS 56
Anexo A – Códigos ESP32 56
Anexo B – Códigos ESP32-CAM 63

1. INTRODUÇÃO

O termo Internet das coisas (do inglês *Internet of Things* – IoT) representa uma área do conhecimento em expansão na atualidade. Para entender seus fundamentos, é importante definir que a internet é um ambiente virtual repleto de informações provenientes de diversos dispositivos a ela conectados, sendo que não existe a necessidade de estarem no mesmo local para se comunicarem. Assim, uma grande quantidade de dados pode ser reunida e disponibilizada para acesso virtual, dando origem a uma "nuvem" de dados. Uma vez que os dispositivos estão ligados à rede de internet, é possível identificá-los por um endereço IP (do inglês *Internet Protocol*, ou simplesmente protocolo de rede) e, portanto, pode-se estabelecer a interação entre estes, obedecendo regras para padronização do formato das informações. Em termos gerais, qualquer "coisa" é capaz de prover informações ao ambiente e se comunicar com qualquer outra "coisa" que também esteja conectada na rede. Esta é uma das mais amplas definições de internet das coisas (STEVAN JR, 2018) e este conceito pode ser utilizado nas mais variadas aplicações, sendo que as projeções apontam para um crescimento potencialmente significativo da IoT em um futuro próximo (BORBA, 2018).

Uma das aplicações dos fundamentos de IoT está relacionada com a área de Automação Residencial, também conhecida como Domótica. A Automação Residencial pode proporcionar maior conforto e conveniência aos usuários, otimizando tarefas do residente e, inclusive, podendo exercer um excelente fator de economia, assim como foi a evolução da telefonia celular na década de 90 (TEZA, 2002). Neste contexto, é necessário empregar dispositivos específicos para cada tipo de recurso desejado, estabelecendo um ou mais protocolos de comunicação entre eles. Como os dispositivos utilizados na domótica normalmente possuem capacidade limitada, os protocolos devem ser adequados para lidar com a baixa largura de banda, latência e instabilidades de comunicação (BORBA, 2018).

O protocolo *Message Queuing Telemetry Transport* (MQTT) é voltado para aplicações Máquina para Maquina (do inglês *Machine-to-Machine* – M2M), além disso utiliza o paradigma *publish/subscribe* para realizar a troca de mensagens. Este protocolo foi criado pensando em atender a taxas de conexão de rede e *hardwares* mais modestos, porém possibilitando em certo grau uma garantia de entrega das mensagens. Isso torna o MQTT um dos principais candidatos para a comunicação quando se trata de aplicações em IoT (MELO, 2018).

> O MQTT foi inventado e desenvolvido inicialmente pela IBM no final dos anos 90. Sua aplicação original era vincular sensores em pipelines de petróleo a satélites. Como seu nome sugere, ele é um protocolo de mensagem com suporte para a comunicação assíncrona entre as partes. Um protocolo de sistema de mensagens assíncrono desacopla o emissor e o receptor da mensagem tanto no espaço quanto no tempo e, portanto, é escalável em ambientes de rede que não são confiáveis. Apesar de seu nome, ele não tem nada a ver com filas de mensagens, na verdade, ele usa um modelo de publicação e assinatura. No final de 2014, ele se tornou oficialmente um padrão aberto OASIS, com suporte nas linguagens de programação populares, usando diversas implementações de software livre (IBM, 2017).

A implementação de sistemas de automação residencial pode ser obtida com o emprego de microcontroladores devido, principalmente, às características integradoras de suas entradas e saídas, o que os tornam práticos para executar variadas tarefas. A possibilidade de conexão com a internet também é algo fundamental, apresentando-se como uma opção para controlar e monitorar ambientes podendo, ainda, servir de host para uma página *web*, o que permite que o usuário mantenha a supervisão e gestão dos equipamentos integrados ao microcontrolador de forma remota. (SANTOS; JUNIOR, 2019). O ESP32 é um chip microcontrolador produzido pela empresa Espressif para trabalhar com dispositivos móveis e aplicações em IoT, sobretudo com protocolo MQTT. O módulo contém ainda Wi-Fi e Bluetooth já integrados e conta com um processador de 32 bits (MARTINS, 2019).

1.1. Motivação

Segundo a Aureside (2021) o uso de dispositivos de automação nas residências deve crescer 20% até 2023. Pretende-se mostrar nesse trabalho a possibilidade de uma automação de baixo custo e fácil aplicabilidade. A ideia é demonstrar as vantagens de se instalar um sistema de automação residencial, assim como apresentar as principais formas de implementação, as quais podem proporcionar um aumento no conforto e na segurança da casa. É importante destacar que a escolha adequada de protocolos de comunicação, microcontroladores, sensores e demais dispositivos eletrônicos serão abordados, já que influenciam diretamente na funcionalidade do sistema, assim como na redução dos custos envolvidos.

Para destacar o crescimento da IoT, foi feita uma análise de dados na plataforma Web of Science a respeito da quantidade de publicações com o termo "*Internet of Things"*. É possível ver no Gráfico 1 a quantidade de publicações no período entre o ano 2000 e 2021. Percebe-se que há um aumento significativo de publicações relacionadas a IoT nos últimos anos, o que aponta a grande relevância do tema na atualidade.

Gráfico 1: Publicações relacionadas a *Internet of Things* (2000-2021)

Fonte: Adaptado de (Web of Science).

1.2. Objetivos gerais e específicos

Esta monografia tem como objetivo apresentar um protótipo de automação residencial utilizando conceitos de IoT. Serão utilizados os microcontroladores ESP-32 e ESP-32 CAM. Além disso, também serão utilizados os protocolos de mensagens *Message Queuing Telemetry Transport* (MQTT) e o *Hypertext Transfer Protocol* (HTTP) para a comunicação entre os dispositivos, sendo que o objetivo geral consiste em apresentar, principalmente, a utilidade do MQTT em aplicações de domótica. Especificamente, foram escolhidos os seguintes recursos de automação residencial a serem implementados neste trabalho:

- Sistema de vigilância por câmera;
- Sistema de alarme por detecção de presença;
- Detecção de vazamento de gás;
- Vigilância da umidade e temperatura do ambiente;
- Sistema de climatização;
- Irrigação automática;
- Controle de iluminação;
- Controle automatizado de persiana;
- Iluminação externa automatizada.

2. REFERENCIAL TEÓRICO

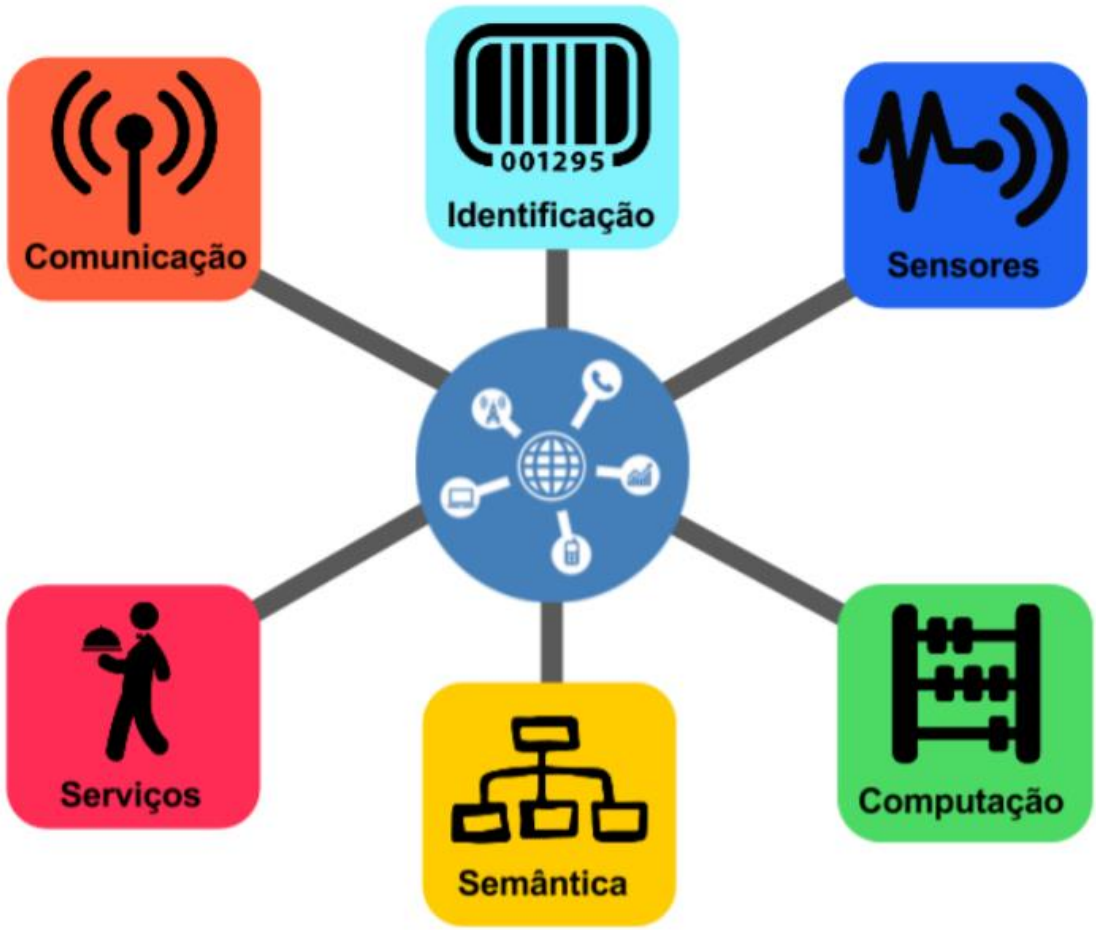
O referencial teórico tem como objetivo apresentar os principais conceitos e o embasamento utilizado no desenvolvimento do trabalho, apresentando assim os protocolos de mensagens, os sensores, dispositivos e microcontroladores utilizados.

2.1. Internet das Coisas

Segundo Santos (2016) a Internet das Coisas surgiu dos avanços de diversas áreas como a microeletrônica, sistemas embarcados, comunicação e sensoriamento. A IoT nada mais é do que uma extensão da internet atual, onde há a possibilidade de conectar diversos dispositivos. Atualmente, além dos computadores outros dispositivos também estão conectados à rede de internet, como por exemplo as TVs, smartphones, automóveis e diversos outros. Além desses, uma nova gama de aplicações deve ser considerada, como as cidades inteligentes, aplicações para saúde e casas automatizadas.

Desta forma, a IoT pode ser vista como uma combinação de várias tecnologias, as quais são complementares para viabilizar a integração dos objetos no ambiente físico ao mundo virtual. A construção de um sistema baseado em IoT pode ser dividida em alguns blocos, sendo eles o de identificação cujo objetivo é o de identificar objetos unicamente para conectá-los à internet. Em seguida a utilização de sensores permite que informações sobre o ambiente sejam coletadas, podendo manipulá-lo com o auxílio de dispositivos atuadores e indicadores. O bloco da comunicação representa as técnicas para conectar e comunicar os objetos inteligentes. Há, ainda, o bloco da computação que inclui a unidade de processamento como, por exemplo, os microcontroladores; e o bloco de serviço que possibilita a tomada de decisões do sistema para reagir de modo adequado a um determinado cenário. Por fim, existe o bloco da semântica que se refere à habilidade de extração de conhecimento dos objetos na IoT (SANTOS, 2016), conforme diagrama de blocos apresentado pela Figura 1.

Figura 1: Blocos da Internet das coisas.

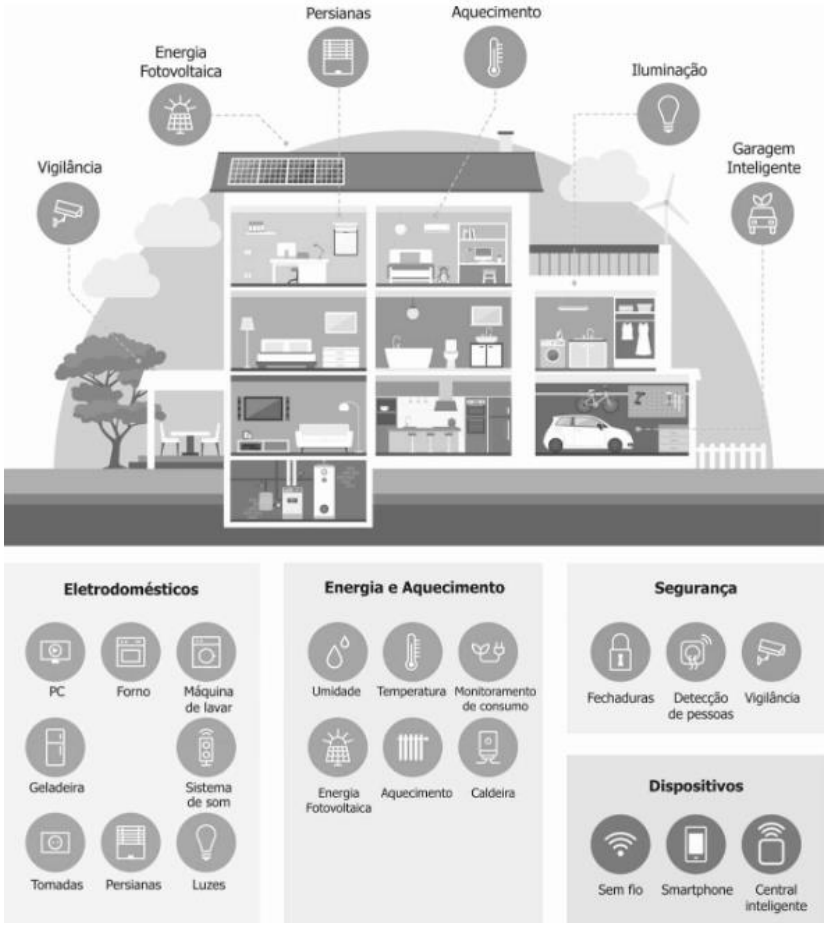


Fonte: Extraída de (SANTOS, 2016).

2.1.1. Domótica

O termo domótica é o resultado da junção da palavra romana *domus*, que significa casa, e a palavra robótica, que se refere a realização de controle automatizado de algo por robôs e que também pode ser simplificado pelo termo automatização. Então tem-se que a domótica resulta na automatização do ambiente residencial. O processo de automação residencial se relaciona diretamente com a Internet das coisas, objetivando-se a criação de casas mais inteligentes (JUNIOR; FARINELLI, 2018). Alguns recursos que podem ser automatizados em uma residência são, por exemplo, o gerenciamento de dispositivos de segurança, o controle da temperatura ambiente e o acionamento de eletrodomésticos de modo geral, como mostra a Figura 2.

Figura 2: Exemplos de automação no ambiente residencial.



Fonte: Extraída de (JUNIOR; FARINELLI, 2018).

2.2. Microcontroladores

Segundo Borges et al (2006) em sistemas de Automação, são raros os exemplos que não necessitam do uso de alguma unidade de processamento. A não ser que o sistema possua uma lógica extremamente simples, o uso de uma ou mais CPUs (*Central Processing Unit*) se faz necessário. No geral os microcontroladores podem ser definidos como processadores encapsulados com memória, interface de entrada/saída de dados e dispositivos periféricos como por exemplo os conversores ADC, temporizadores, interface para comunicação serial, etc. Após definida a necessidade de empregar um processador no sistema, ainda é necessário definir qual plataforma utilizar. A escolha do "cérebro" do sistema deve-se partir das necessidades do projeto e as características apresentadas como custo, capacidade de processamento, memória entre diversos outros aspectos.

Tendo isso em vista foram comparados alguns dos microcontroladores mais populares para definir qual adequa-se melhor ao projeto proposto. Como visto na Tabela 1, os

microcontroladores ESP32 e ESP32-CAM foram os escolhidos para este projeto pois possuem maior capacidade de processamento e memória se comparado a plataformas Arduino. Além disso, ambos os microcontroladores possuem Wi-Fi integrado, o que os tornam convenientes para a utilização de comunicação via internet. Além do Wi-Fi integrado, o ESP32-CAM também possui o módulo de câmera incluso, o que se mostra mais adequado para a aplicação de vigilância proposta anteriormente.

Tabela 1: Comparação dos Microcontroladores.

Plataforma microcontrolada	**Arduino UNO**	**Arduino MEGA 2560**	**ESP32**	**ESP32 CAM**
Processador	AVR® 8-bit RISC	ATmega2560 RISC com até 16 MIPS	Xtensa® Dual-Core 32-bit LX6	Xtensa® Dual-Core 32-bit LX6
Frequência de operação	0 ~ 16 MHz	0 ~ 16 MHz	80MHz ~ 240MHz	80MHz ~ 240MHz
Memória FLASH	32KB	256 KB	4MB	4MB
Memória RAM/SRAM	2KB	8 KB	520KB	520KB
Memória ROM/EEPROM	4KB	4KB	448KB	448KB
Pinos I/O	23 com 6 PWM	54 com 14 PWM	34 com com 16 PWM	11 com 11 PWM
Wi-Fi integrado	Não	Não	Sim	Sim
Bluetooth integrado	Não	Não	Sim	Sim
Câmera	Não	Não	Não	Sim

Fonte: Adaptado de (OLIVEIRA, 2019) e (ROBOCORE, 2021).

2.3. Protocolos de comunicação

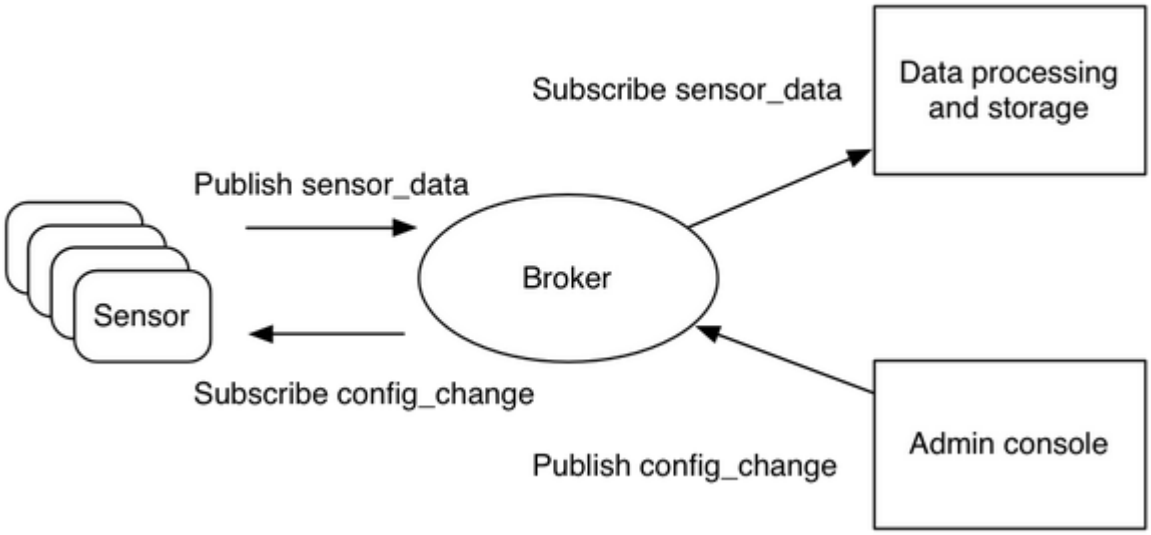
Os protocolos são conjuntos de normas que permitem que uma máquina conectada à internet consiga se comunicar com outra também conectada à rede. É dessa maneira que um usuário consegue enviar e receber mensagens instantâneas, fazer *download* e *upload* de dados/arquivos, etc. Os protocolos de internet funcionam como uma língua universal entre computadores, de forma que independente do fabricante e do sistema operacional da máquina, essa linguagem pode ser interpretada por todas as máquinas igualmente. Dessa forma não é necessário um *software* adicional para que um computador possa entender os protocolos de rede (WEBLINK, 2020).

2.3.1. *Message Queue Telemetry Transport* (MQTT)

O protocolo *Message Queue Telemetry Transport* (MQTT) foi criado em 1999 pela IBM e é um protocolo de comunicação otimizado para rodar em redes TCP/IP de pouca confiabilidade, alta latência e baixa banda. Este protocolo foi desenvolvido para projetos que necessitam de poucos recursos de *hardware*, ou seja, para rodar em dispositivos embarcados mais simples. Um sistema que utiliza microcontroladores para obter informações a partir de sensores pode publicar esses dados via protocolo MQTT em um *Broker*. O *Broker* é um dispositivo responsável por distribuir as mensagens para os clientes, então estes dados podem ser coletados por diversos dispositivos já que o MQTT é um protocolo "*publish-subscribe*" (BORBA, 2018).

No funcionamento do protocolo, o cliente conecta-se ao *Broker* e ele pode assinar qualquer tópico de mensagem no *Broker*. O cliente também publica as mensagens em um tópico, enviando assim esta mensagem e o tópico ao *Broker*, em seguida o *Broker* encaminha a mensagem aos clientes que assinam este tópico (IBM, 2017). O modelo de publicação e assinatura pode ser melhor entendido através da Figura 3.

Figura 3: Modelo de publicação e assinatura do protocolo MQTT.



Fonte: Extraída de (IBM, 2017).

Segundo Martins (2019), o MQTT provém 3 níveis de mensagens. O nível 0 é o mais simples onde não há garantias de entrega da mensagem e também não existe retransmissão ou confirmação da entrega da mensagem. No nível 1 as mensagens são transmitidas até serem confirmadas pelo receptor, porém há a chance de a mensagem chegar múltiplas vezes. Por fim

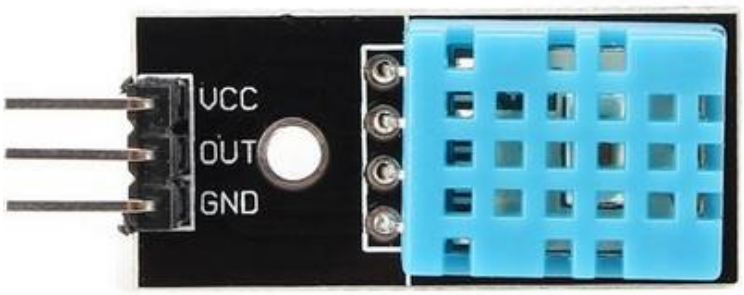
tem-se o nível 2 que, além de garantir o recebimento da mensagem, também garante que elas serão entregues apenas uma vez ao destinatário.

2.4. Sensores e dispositivos eletrônicos

Existem inúmeros sensores, os quais são responsáveis pela medição de uma grandeza física. Além disso, esses sensores podem estar associados a atuadores e outros dispositivos eletrônicos com o objetivo de executar alguma ação específica. Neste contexto, esta seção apresenta informações básicas sobre os principais elementos utilizados nas implementações propostas. O DHT11, por exemplo, é um sensor de temperatura que utiliza um termistor NTC (*Negative Temperature Coeficient*).

Este sensor também permite a medição da umidade ambiente pelo acoplamento à um sensor do tipo HR202. A faixa de operação de temperatura é de 0 a 50°C com uma precisão de $\pm$2°C e, para a umidade a faixa é entre 20% a 90% de umidade relativa e possui uma precisão de $\pm$5%. O tempo de resposta para as medições é na ordem de 5s, sendo sua operação em frequência de até 1Hz. A alimentação deve ser feita entre 3V e 5V e o consumo é de 200 μA a 500 mA, conforme indicação mostrada na Figura 4. Um dispositivo alternativo é o DHT22 que possui medições mais precisas em comparação ao DHT11 (STEVAN JR, 2018).

Figura 4: Sensor DHT11.



Fonte: Extraída de (FILIPEFLOP, 2021e).

O sensor de presença HC-SR501 emite sinais de saída quando movimentos são identificados. Sua aplicação está voltada para a segurança, em alarmes por exemplo, porém também pode ser utilizado para acender lâmpadas ou abrir portas de forma automática. A sensibilidade mínima é 3 metros e a máxima é de 7 metros, já o tempo em que sua saída é

sensibilizada por algum movimento pode variar de 3 segundos a 5 minutos, sendo que a sensibilidade e o tempo podem ser ajustados. O sensor deve ser alimentado com tensão entre 5 V a 20 V. A saída é de 3,3 V para sinal alto e 0 V para baixo. O HC-SR501 pode ser visto na Figura 5 (OLIVEIRA, 2017).

Figura 5: Sensor HC-SR501.

Fonte: Extraída de (FILIPEFLOP, 2021i).

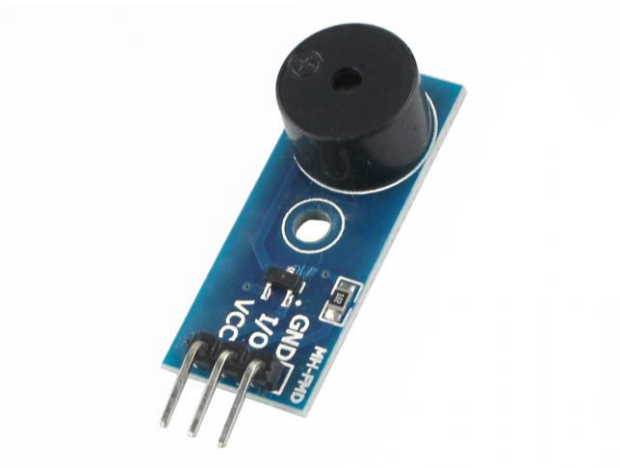
Um sensor de gases inflamáveis pode informar se a residência está em risco de explosões. O MQ-2 possui uma saída digital que indica se há gases inflamáveis ou não, assim como uma saída analógica que indica a quantidade de presença de partículas de gás no ar, indo de 300 a 10000 partes por milhão. A saída digital é representada por uma tensão que varia de 0 a 3,3 V (OLIVEIRA, 2017). O MQ-2 pode ser visto na Figura 6.

Figura 6: Sensor MQ-2.

Fonte: Extraída de (OLIVEIRA, 2017).

O buzzer ativo BP09 é um indicador sonoro que reproduz sinais com intensidade acima de 85 dB, ao ser alimentado com tensão entre 3,3V e 5V. Também foi projetado para operar em temperaturas entre -20°C a 45°C, o que significa que funcionará para a temperatura ambiente adequadamente. Possui aplicações na área de segurança, podendo funcionar como um alarme (USINAINFO, 2021). Este buzzer pode ser visto na Figura 7.

Figura 7: Buzzer Ativo BP09.



Fonte: Extraída de (USINAINFO, 2021).

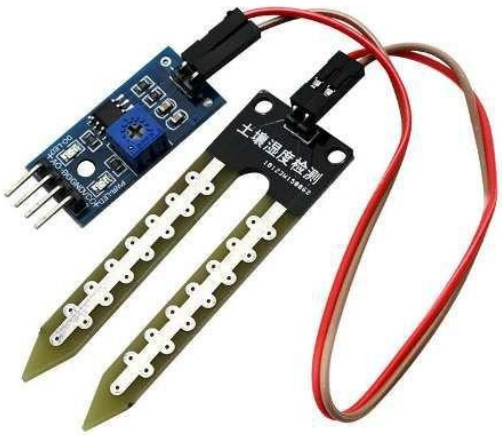
O módulo sensor de luminosidade é um pequeno módulo equipado com um fotoresistor cuja resistência é determinada de acordo com a intensidade de luz que nele é aplicada. É muito utilizado em aplicações que necessitem de uma iluminação automatizada. É equipado com o comparador LM393 e sua tensão de operação é de 3,3V a 5V em corrente contínua. Possui três pinos sendo um VCC, um GND e uma saída digital, sua temperatura de operação é entre -30°C e 70°C (SARAVATI,2021). Este módulo pode ser visto na Figura 8.

Figura 8: Sensor de luminosidade.

Fonte: Extraída de (SARAVATI, 2021).

O sensor de umidade de solo tem a finalidade de detectar as variações de umidade no solo, quando o solo está seco a saída digital fica em estado alto, quando o solo está úmido a saída muda para estado baixo. Este módulo pode ser calibrado via potenciômetro sendo um comparador LM393 utilizado para comutar os estados de sua saída. O módulo possui quatro pinos sendo eles um VCC (3,3V a 5V), um GND, uma saída digital e uma saída analógica (ELETROGATE, 2021). Neste projeto o sensor de umidade, o qual pode ser visualizado pela Figura 9, foi utilizado para um sistema de irrigação automático.

Figura 9: Sensor de umidade do solo.



Fonte: Extraída de (ELETROGATE, 2021).

O servomotor é um motor que possibilita o controle de posição. O Micro Servo TowerPro 9g SG90 é um servo excelente para projetos mecatrônicos, sendo alimentado por uma tensão de operação entre 3V e 7,2V. Este motor apresenta velocidade de 0,12s/60 graus ao ser alimentado a 4,8V e estando sem carga. Seu torque pode ser de 1,2 kg.cm a 1,6 kg.cm, dependendo da tensão de alimentação, e sua temperatura de operação pode variar entre -30°C e 60°C (BAUERMEISTER, 2018). Este servo motor pode ser visualizado pela Figura 10 e é utilizado neste trabalho de conclusão de curso como representação da abertura de persianas.

Figura 10: Micro Servo TowerPro 9g SG90.

Fonte: Extraída de (BAUERMEISTER, 2018).

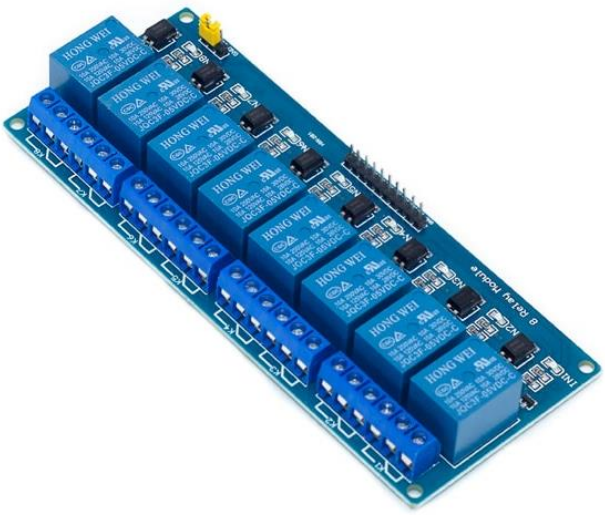
O conversor USB-Serial TTL PL2303 é um adaptador para converter sinais usb em sinais TTL RS232, o que o torna útil para comunicação com microcontroladores no geral. Ele possui em uma extremidade um conector usb para ser conectado ao computador e na outra extremidade uma barra de pinos para uma fácil ligação com a placa cuja comunicação deve ser estabelecida. Este conversor possui tensões de 3,3V e 5V, o que lhe permite trabalhar com uma quantidade maior de microcontroladores (CURTOCIRCUITO, 2021). O conversor é mostrado na Figura 11 e é utilizado neste trabalho para realizar a gravação do código no ESP32-CAM.

Figura 11: USB-Serial Ttl PL2303.

Fonte: Extraída de (CURTOCIRCUITO, 2021).

O módulo relé 5V de oito canais apresentado na Figura 12 trata-se de uma interface de potência, cuja finalidade é acionar dispositivos a serem alimentados por uma fonte de energia compatível, o acionamento é feito com um sinal de baixa corrente proveniente do microcontrolador. Com este módulo é possível acionar até oito cargas com tensões de até 250V em corrente alternada ou com uma tensão de até 30V corrente contínua, suportando uma corrente de até 10A em ambos os casos. O tempo de resposta do dispositivo é de 5ms a 10ms e sua corrente típica de operação é entre 15mA e 20mA (FILIPEFLOP, 2021d).

Figura 12: Módulo relé de oito canais.



Fonte: Extraída de (FILIPEFLOP, 2021d).

A válvula solenóide é um equipamento que pode ser utilizado em diversas aplicações, funcionando como um sistema de abertura e fechamento, que permite ou não a passagem de água (JEFFERSON, 2021). Um exemplo de válvula solenóide pode ser visto na Figura 13. Neste trabalho a válvula não foi utilizada, é apresentada para demonstrar qual ferramenta seria utilizada em uma aplicação real do sistema de irrigação.

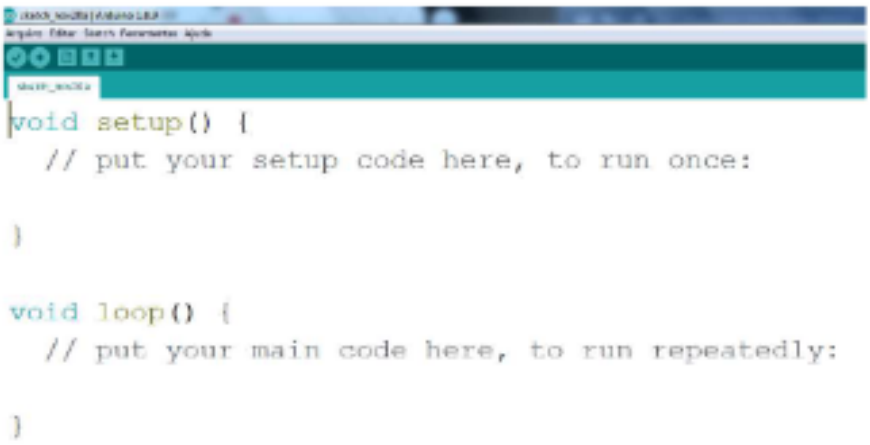
Figura 13: Válvula Solenóide.

Fonte: Extraída de (JEFFERSON, 2021).

2.5. Arduino IDE

Segundo Martins (2019) a Arduino IDE é uma ferramenta utilizada para programar instruções (códigos) a serem executadas por microcontroladores da plataforma Arduino. Apesar do nome, essa ferramenta possui compatibilidade com vários tipos de placas, inclusive o ESP32, desde que seja feito o download de bibliotecas que dão suporte a placa. O código a ser compilado deve se encaixar nas funções pré-definidas pela Arduino IDE, essas funções são chamadas de *setup* e *loop*. A função *setup* é executada uma única vez, ou seja, serve para as configurações iniciais, enquanto a função *loop* é executada repetidas vezes para garantir o funcionamento contínuo da aplicação implementada. A imagem da tela inicial da Arduino IDE é mostrada na Figura 14.

Figura 14: Arduino IDE.



Fonte: Extraída de (MARTINS, 2019).

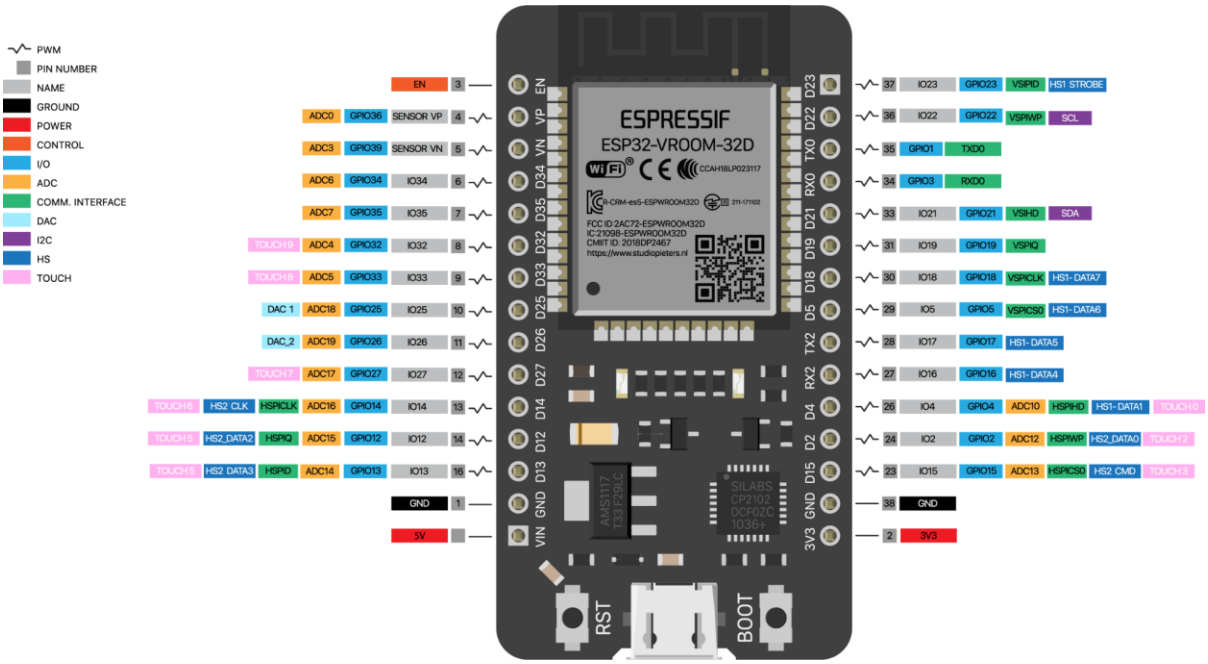
3. METODOLOGIA

Neste tópico será abordado a metodologia utilizada para desenvolver o protótipo proposto. Inicialmente foram escolhidos os microcontroladores utilizados para a aplicação, logo em seguida é apresentado o princípio de funcionamento do sistema, baseando-se em seu diagrama esquemático e, por fim, é exposto o protótipo construído.

3.1. ESP32

Foi utilizado o ESP32 no protótipo de automação residencial pelos fatores já citados anteriormente. Neste microcontrolador foram conectados os seguintes sensores DHT11, sensor de umidade do solo, servo motor, sensor de luminosidade e também o módulo relé de oito canais. Na Figura 15 é possível visualizar a distribuição de pinos (*pinout*) em um ESP32.

Figura 15: Pinout ESP32.



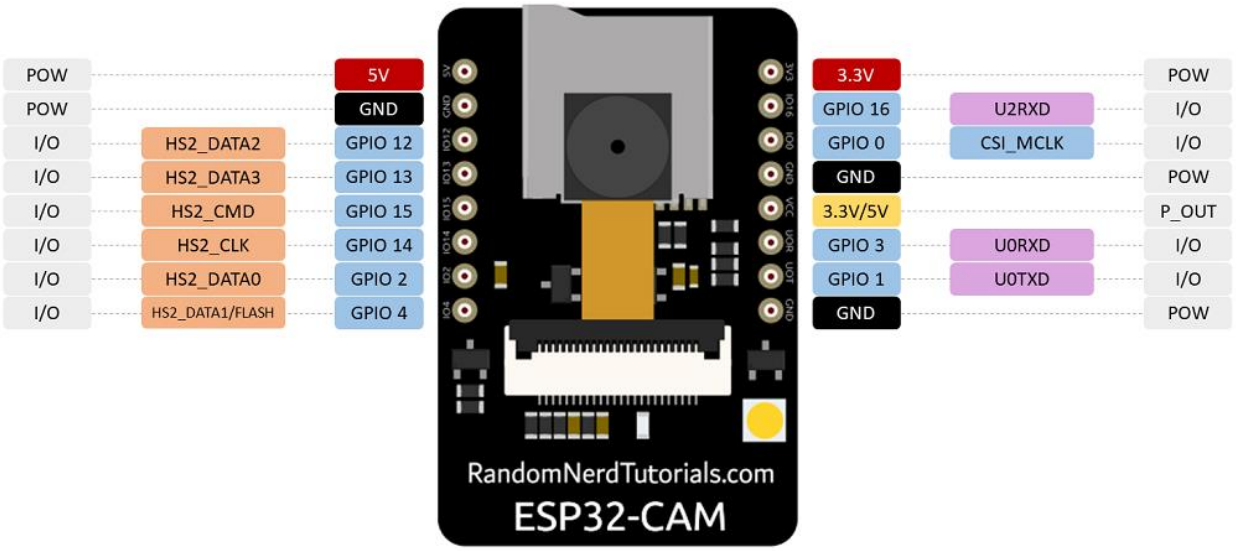
Fonte: Extraída de (STUDIOPIETERS, 2020).

3.2. ESP32-CAM

Outro microcontrolador utilizado foi o ESP32-CAM. Nesta placa foram conectados o sensor de presença, o sensor de gás e, ainda, o buzzer ativo. Outra aplicação deste

microcontrolador é o sistema de vigilância, pois este módulo possui uma câmera de 2 MP integrada. O *pinout* do ESP32-CAM pode ser encontrado na Figura 16.

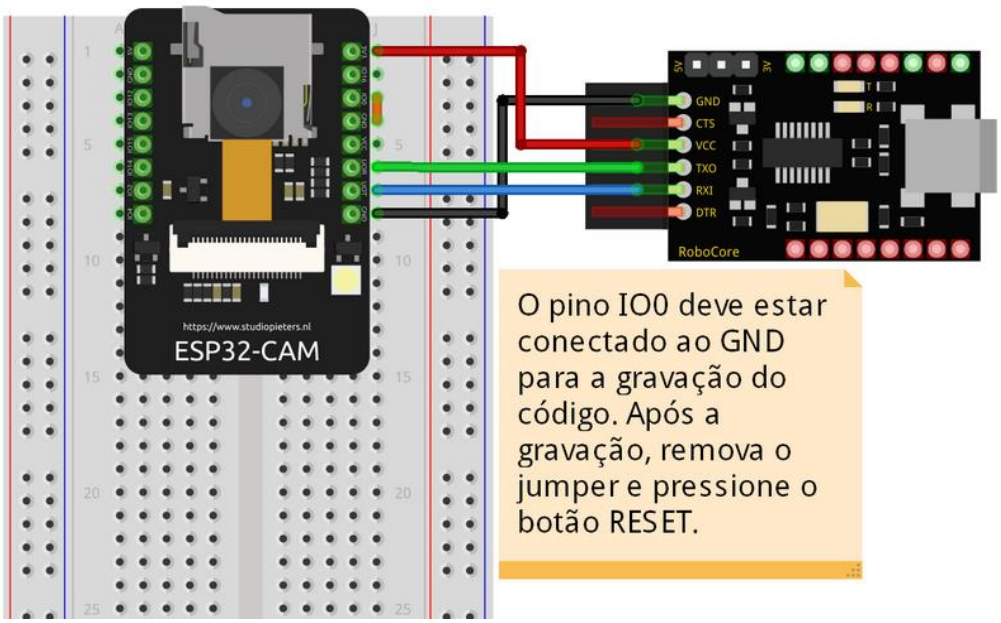
Figura 16: Pinout ESP32-CAM.



Fonte: Extraída de (RANDOMNERDTUTORIALS, 2020).

É importante destacar que, para gravar o código no ESP32-CAM é necessário utilizar um conversor, já que a placa não possui entrada USB. Para a comunicação com o computador é utilizado um módulo FTDI como mostrado na Figura 17.

Figura 17: ESP32-CAM conectado ao módulo FTDI.

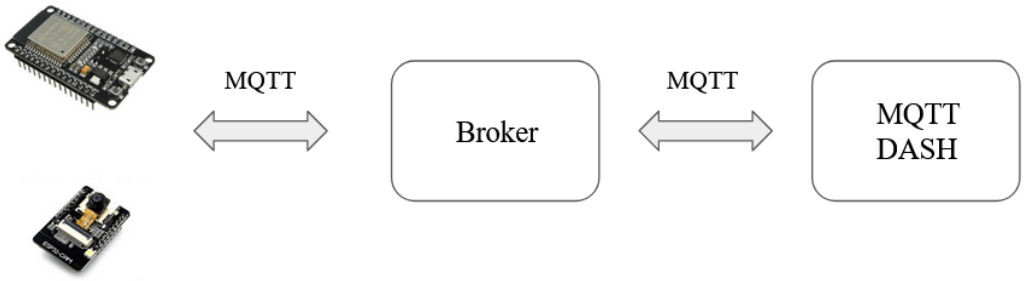


Fonte: Extraída de (ROBOCORE, 2021).

3.3. Arquitetura do projeto

A proposta de automação residencial contempla a implementação do controle de iluminação, climatização automática, sistema de alerta para vazamentos de gás, irrigação automática, sistema de alarme, controle a distância de persianas e sistema de vigilância via câmera. Para a comunicação entre os dispositivos conectados ao ESP32 e ao ESP32-CAM foi utilizado o protocolo de mensagens MQTT com o uso das seguintes ferramentas: aplicativo gratuito para smartphones *Android MQTT DASH* e o *EMQX Broker*, que é um *Broker* gratuito e *open-source*. Esta arquitetura de comunicação do projeto pode ser vista na Figura 18. É importante destacar que toda a programação dos microcontroladores foi desenvolvida na Arduino IDE e que, além do protocolo MQTT, foi utilizado o HTTP para realizar o *streaming* de vídeo com o módulo de câmera do ESP32-CAM.

Figura 18: Arquitetura de comunicação do projeto.



Fonte: Elaborado pelo autor, 2021.

Para a comunicação via protocolo MQTT foi escolhido o nível QoS 2 para as mensagens, visto que por não ser uma rede residencial, não há um nível de confiabilidade muito grande. Sendo assim, com este nível escolhido é possível garantir que as mensagens vão ser entregues de maneira correta e apenas uma única vez.

3.4. Operação dos recursos de automação residencial

Nesta seção é abordado o funcionamento do protótipo de maneira mais detalhada, de forma a mostrar como foram programados os microcontroladores para executar os recursos propostos para automação residencial. As cargas acionadas pelo módulo relé de oito canais neste protótipo foram simuladas por lâmpadas CC de 12V. Todas as leituras dos sensores, além de executar uma ação, também retornam um valor para a *dashboard*, que é uma interface gráfica onde o usuário pode visualizar e comandar todos os processos. O significado dessas leituras pode ser visto na Tabela 2.

Tabela 2: Significado da leitura dos sensores.

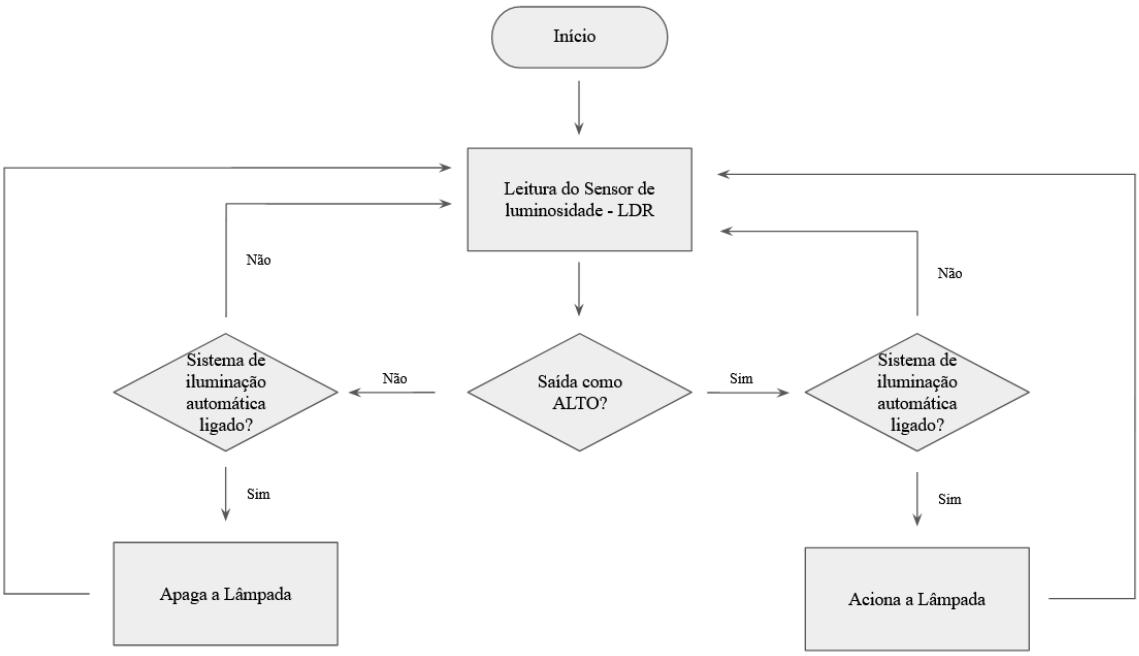
Sensor	Saída	Significado
DHT 11	Sinal Digital	Valor da temperatura em °C
	Sinal Digital	Porcentagem de umidade do ar
HC-SR501	Nível ALTO	Presença detectada
	Nível BAIXO	Nenhuma presença detectada
Sensor de luminosidade	Nível ALTO	Baixa luminosidade
	Nível BAIXO	Alta luminosidade
Sensor de umidade do solo	Nível ALTO	Solo seco
	Nível BAIXO	Solo úmido
MQ-2	Nível ALTO	Gás detectado
	Nível BAIXO	Gás não detectado

Fonte: Elaborado pelo autor, 2021.

3.4.1. Sistema de iluminação automático

Para o sistema de iluminação automático, foi utilizado o sensor de luminosidade para identificar a necessidade de acionar as lâmpadas, caso o usuário habilite esta função. Sempre que o sensor retornar o nível alto o microcontrolador envia um sinal para o relé acionar a carga responsável pela iluminação, simulando por exemplo uma iluminação externa que é acionada automaticamente ao anoitecer. O fluxograma desta aplicação é apresentado na Figura 19.

Figura 19: Fluxograma do funcionamento do sistema de iluminação automática.



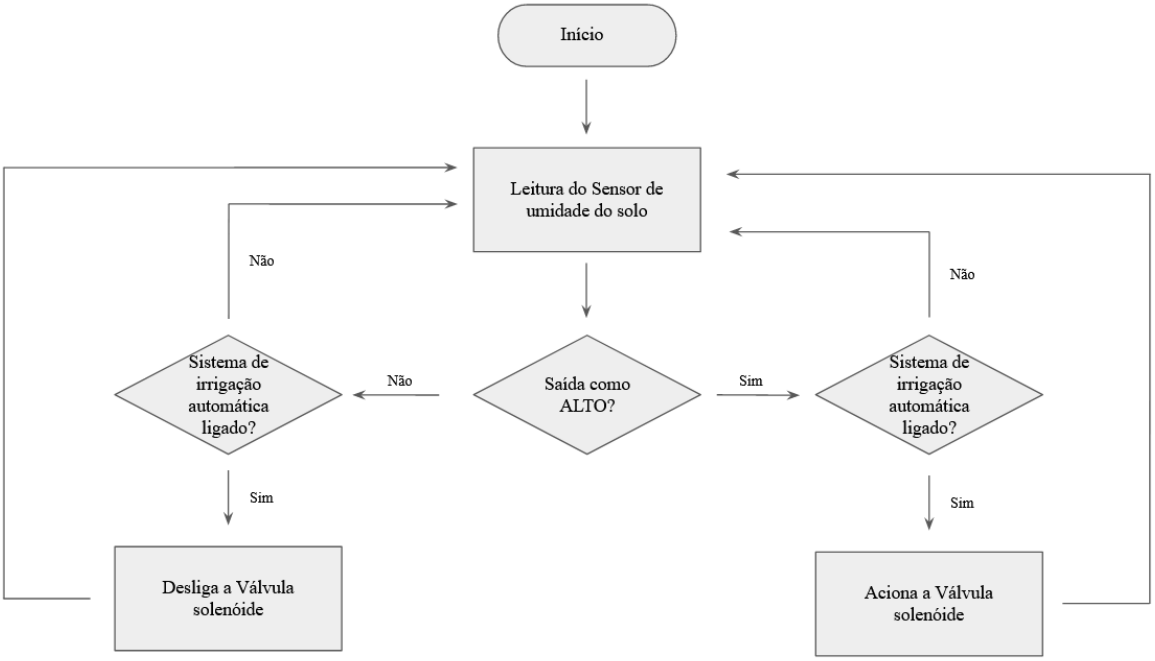
Fonte: Elaborado pelo autor, 2021.

Além deste sistema, também foi criado a opção onde o usuário pode acender uma lâmpada independente de algum sensor, dependendo apenas do comando de ligar ou desligar através do dashboard. Para este caso foi adicionado um interruptor em paralelo com o relé para possibilitar que além do comando via smartphone, também possa ser feito o acionamento manual por interruptor.

3.4.2. Sistema de irrigação automático

O sistema de irrigação utiliza o sensor de umidade do solo para definir quando é necessário que a carga ligada ao relé responsável pela irrigação seja acionada. No protótipo a carga para realizar a simulação foi uma lâmpada de corrente contínua, porém em uma aplicação real espera-se utilizar uma válvula solenoide. O fluxograma desta aplicação pode ser visto na Figura 20.

Figura 20: Fluxograma do funcionamento do sistema de irrigação automática.

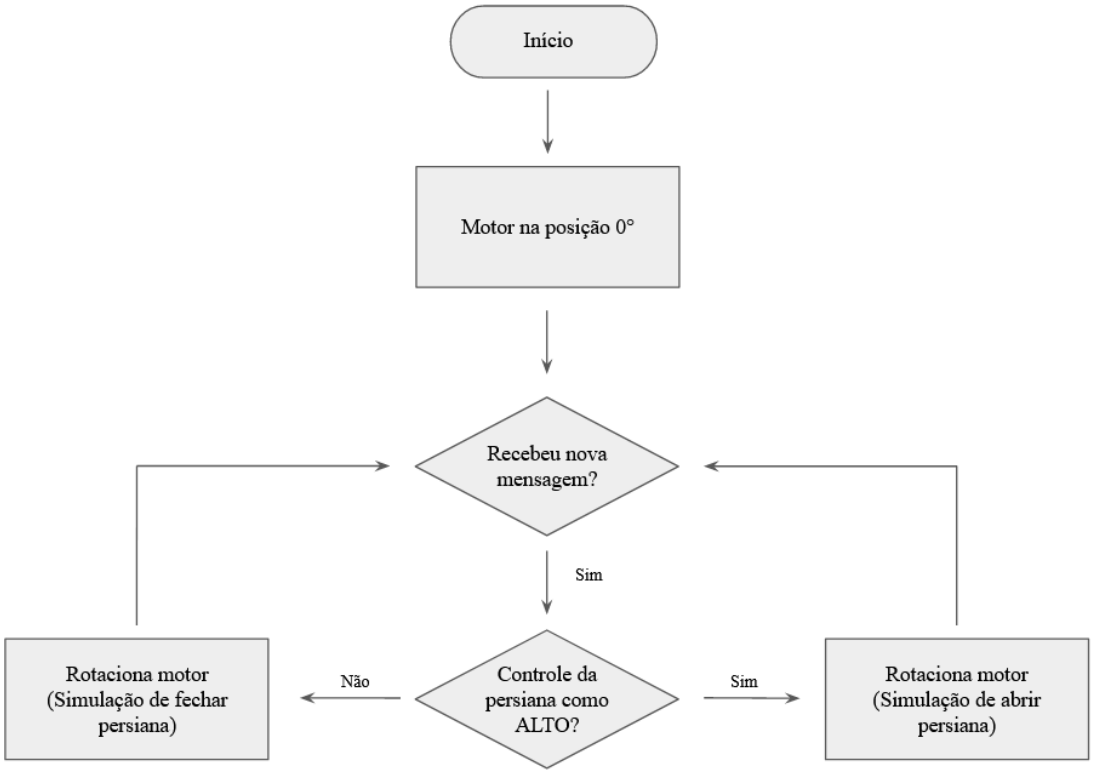


Fonte: Elaborado pelo autor, 2021.

3.4.3. Controle de persianas

Para simbolizar o controle de uma persiana foi utilizado, neste protótipo, um servo motor. Esse motor inicialmente é colocado em uma posição de zero graus e, em seguida, o motor é rotacionado em determinada direção caso o microcontrolador receba a mensagem que a persiana deve ser aberta. Caso mensagem recebida seja para fechar a persiana o motor será rotacionado na direção contrária, fechando a persiana se a mesma estiver aberta, como pode ser visto no fluxograma da Figura 21.

Figura 21: Fluxograma do funcionamento do controle da persiana.

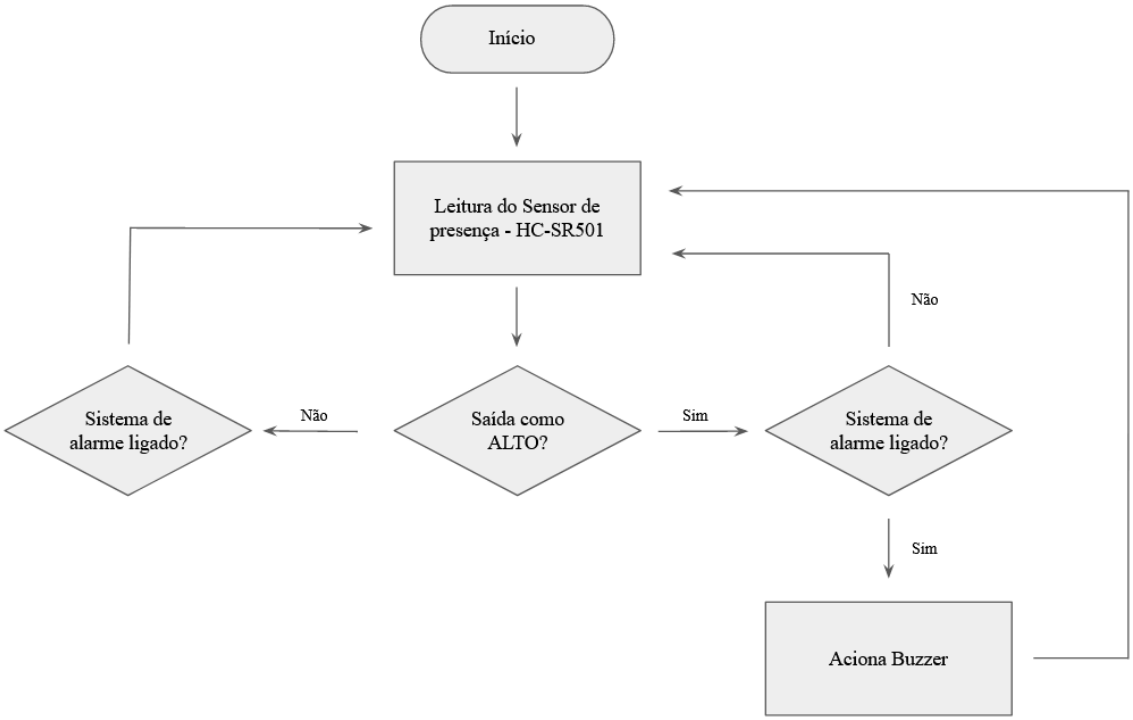


Fonte: Elaborado pelo autor, 2021.

3.4.4. Sistema de alarme

Para este sistema foi utilizado o sensor de presença HC-SR501, caso o usuário ligue o sistema de alarme através da *dashboard* e a leitura do sensor retornar valor com nível alto, o buzzer ativo é acionado por alguns segundos. Caso o sensor retorne ao valor alto, porém o usuário não ligar o alarme, o buzzer não é acionado, sendo possível, no entanto, verificar se houve alguma movimentação local pela *dashboard*. O fluxograma deste sistema de alarme está na Figura 22.

Figura 22: Fluxograma do funcionamento do sistema de alarme.

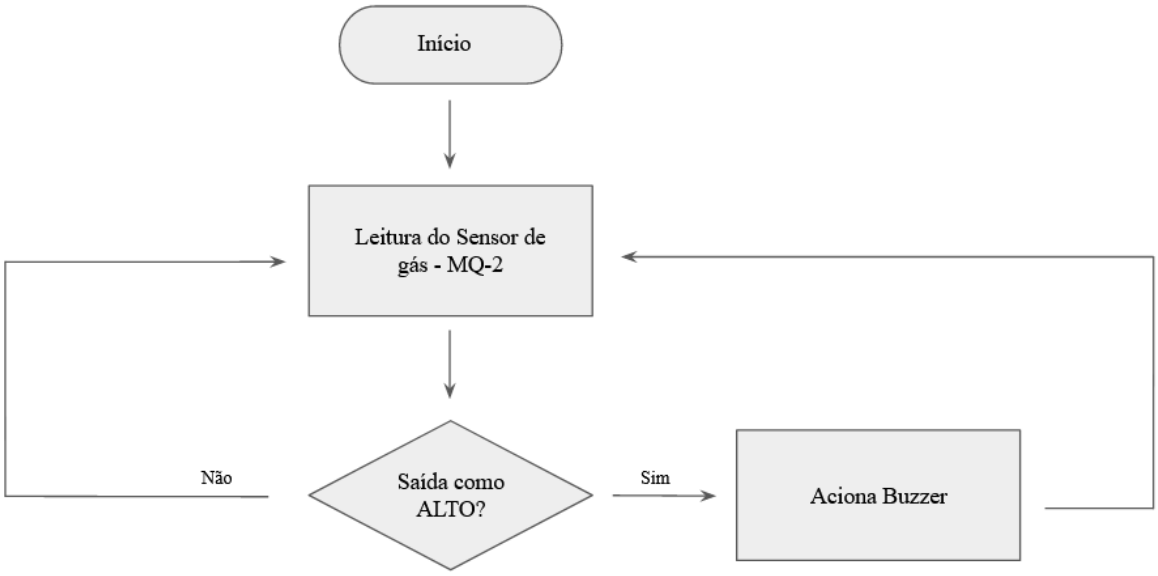


Fonte: Elaborado pelo autor, 2021.

3.4.5. Sistema de detecção de vazamento de gás

Utilizando o sensor de gás MQ-2 foi construído um sistema para detectar vazamento de gás que emite um sinal sonoro caso o sensor retorne em nível alto. Como se trata de um sistema de extrema importância para a segurança do usuário, não há a necessidade de acionar o mesmo. Um local interessante de posicionar o sensor em uma aplicação real seria próximo a região onde está instalado o sistema de gás, como por exemplo na cozinha. Na Figura 23 pode-se ver melhor o funcionamento através do fluxograma.

Figura 23: Fluxograma do funcionamento do sistema de detecção de gás.

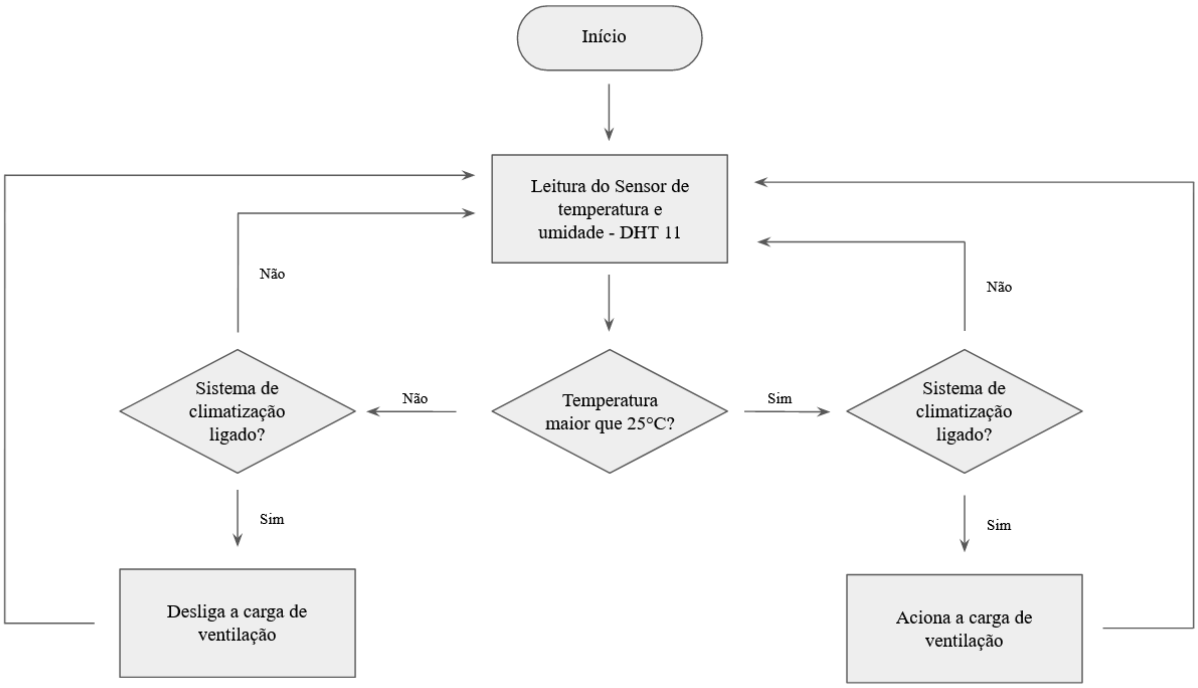


Fonte: Elaborado pelo autor, 2021.

3.4.6. Sistema de climatização automático

Para esta aplicação foi utilizado o sensor de temperatura e umidade do ar DHT11, caso o usuário deixe o sistema ativado e a leitura do sensor apontar uma temperatura superior a 25°C. Então o relé é acionado, no exemplo foi utilizado uma lâmpada para simular uma carga, porém em uma aplicação real seria acionado um sistema de ventilação ou, dependendo do caso, seria emitido um sinal infravermelho para acionamento de ar-condicionado. Caso o sensor faça a leitura que a temperatura esteja menor que 25°C, então o microcontrolador irá atuar e desligar a carga. O funcionamento está mais bem ilustrado na Figura 24.

Figura 24: Fluxograma do funcionamento do sistema de climatização.

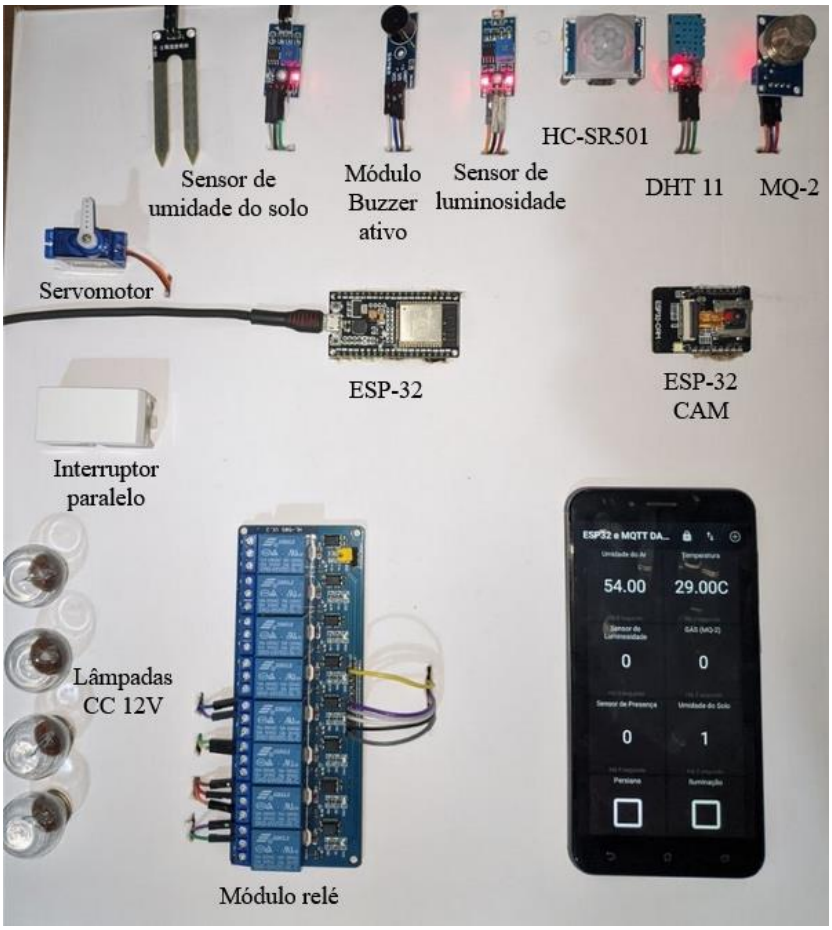


Fonte: Elaborado pelo autor, 2021.

4. RESULTADOS E DISCUSSÕES

Nesta seção do trabalho, serão apresentados os resultados obtidos com a construção do protótipo proposto, o qual pode ser visualizado pela Figura 25. Nesta figura, é possível observar como foram organizados os sensores e os microcontroladores. Sendo que o funcionamento geral do circuito e dos *softwares* foram satisfatórios, ou seja, o protótipo operou conforme previsto, viabilizando sua utilização em uma possível aplicação real de domótica com os mesmos princípios deste sistema.

Figura 25: Protótipo final.

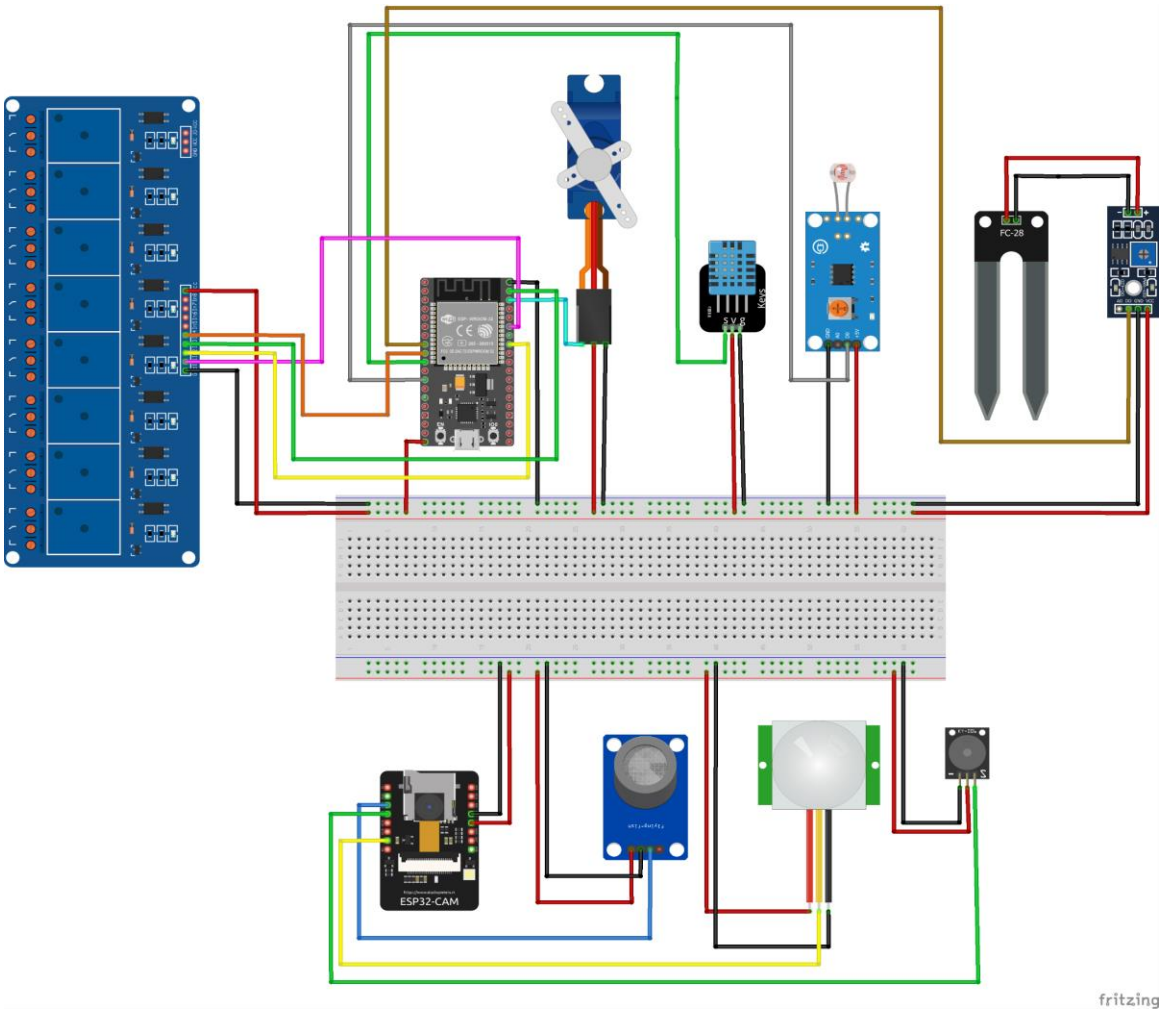


Fonte: Elaborado pelo autor, 2021.

4.1. Ligação e comunicação entre sensores e microcontroladores

A conexão dos elementos constituintes do circuito foi realizada conforme esquemático desenvolvido no *software* Fritzing, o qual está ilustrado pela Figura 26. Nesta figura, é possível verificar os detalhes das conexões que foram realizadas fisicamente no protótipo.

Figura 26: Ligações da implementação do protótipo.



Fonte: Elaborado pelo autor, 2021.

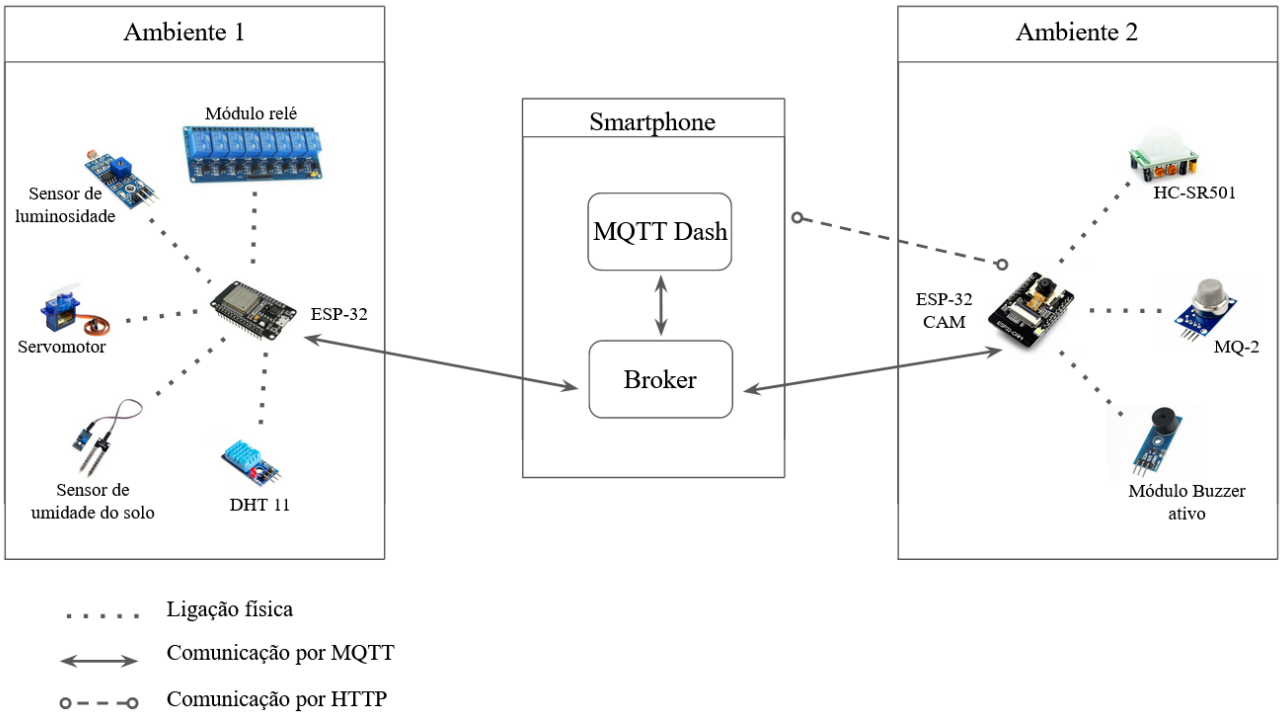
Além das conexões físicas também é necessário estabelecer a comunicação dos dispositivos via internet, utilizando os protocolos já citados anteriormente. Para isso foi elaborado o diagrama de comunicação com seus respectivos protocolos na Figura 27, em que é possível verificar os tipos de comunicação utilizados em trecho da aplicação no protótipo final. Na Tabela 3 está disposto como foi realizada a ligação dos dispositivos nas respectivas plataformas microcontroladas.

Tabela 3: Pinos utilizados para conectar os dispositivos.

Plataforma Microcontrolada	Dispositivo	Pinos
ESP32	Módulo relé	19; 21; 23; 25
	Sensor de luminosidade	14
	Sensor de umidade do solo	33
	DHT 11	26
	Servomotor	22
ESP32-CAM	HC-SR501	2
	MQ-2	12
	Módulo Buzzer ativo	13

Fonte: Elaborado pelo autor, 2021.

Figura 27: Comunicação entre os dispositivos.



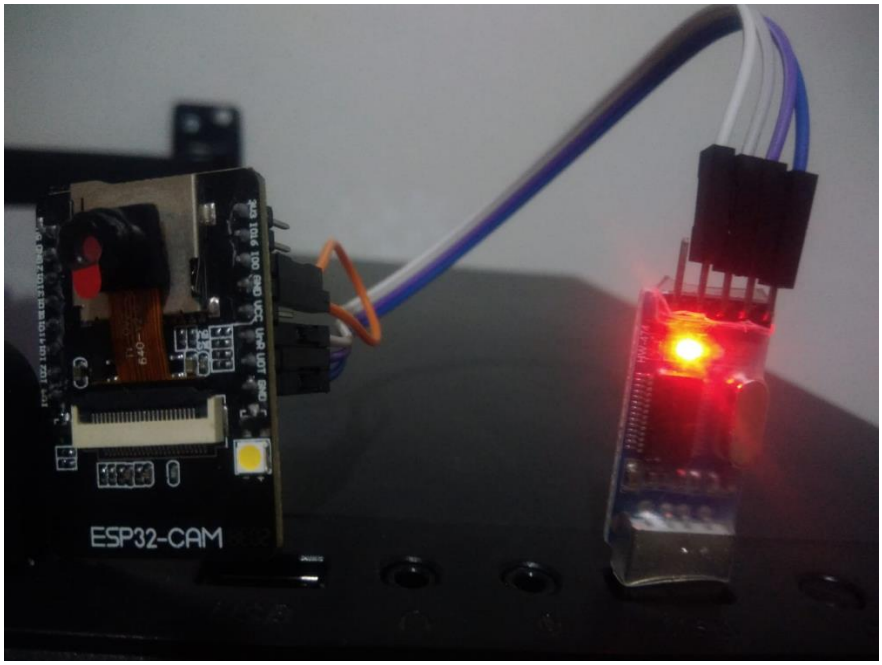
Fonte: Elaborado pelo autor, 2021.

É importante destacar que, para o funcionamento adequado do circuito, o ESP32 foi programado com o código apresentado no Anexo A deste trabalho, enquanto que o código para o ESP32-CAM está disponível no Anexo B.

4.2. Sistema de vigilância

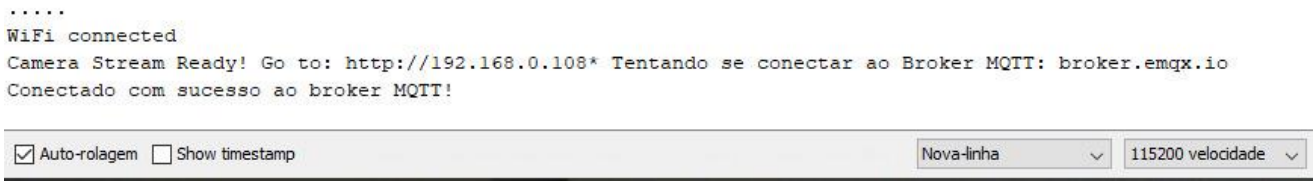
Para o sistema de vigilância foi utilizado o ESP32-CAM, cuja gravação do código foi obtida com o auxílio de um módulo FTDI, seguindo o diagrama mostrado na Figura 17, conforme montagem física mostrada na Figura 28. Após isso, na saída serial da Arduino IDE é possível visualizar o endereço IP para acessar o *streaming* de vídeo, como é apresentado na Figura 29.

Figura 28: Gravação do código no ESP32-CAM.



Fonte: Elaborado pelo autor, 2021.

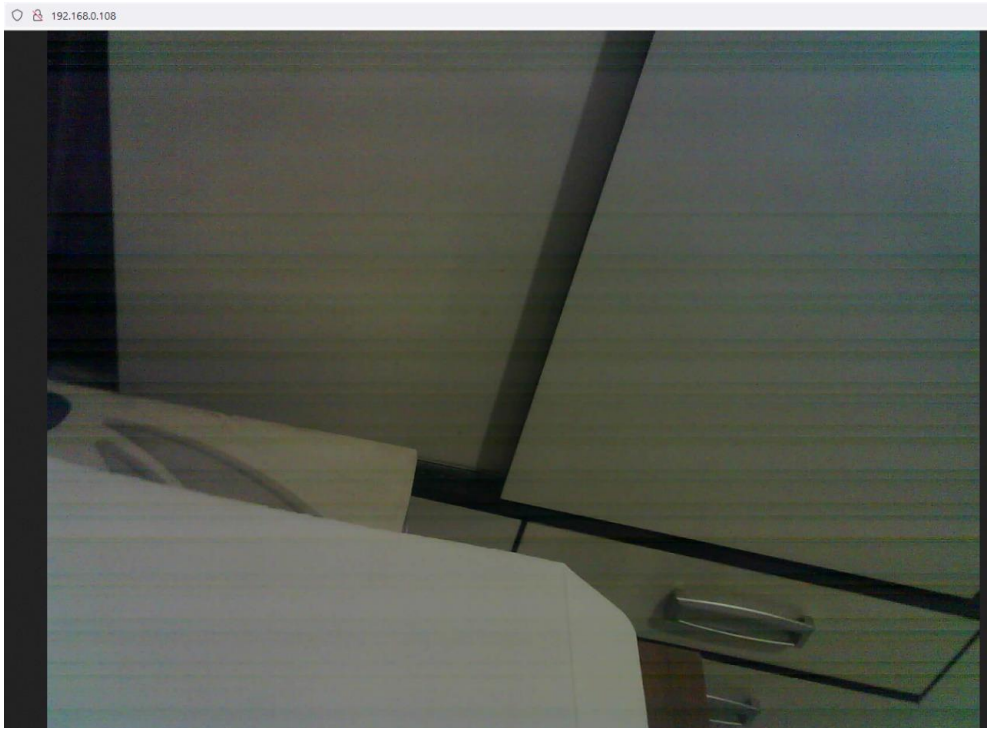
Figura 29: Saída serial da Arduino IDE com ESP32-CAM.



Fonte: Elaborado pelo autor, 2021.

Ao acessar o IP disponibilizado em algum navegador com dispositivo conectado à mesma rede que o microcontrolador, é possível ver as imagens capturadas pela câmera, como visto na Figura 30, a qual foi obtida pelo navegador Google Chrome, possibilitando o acesso ao *streaming* realizado pelo ESP32-CAM.

Figura 30: Streaming de imagem do ESP32-CAM.

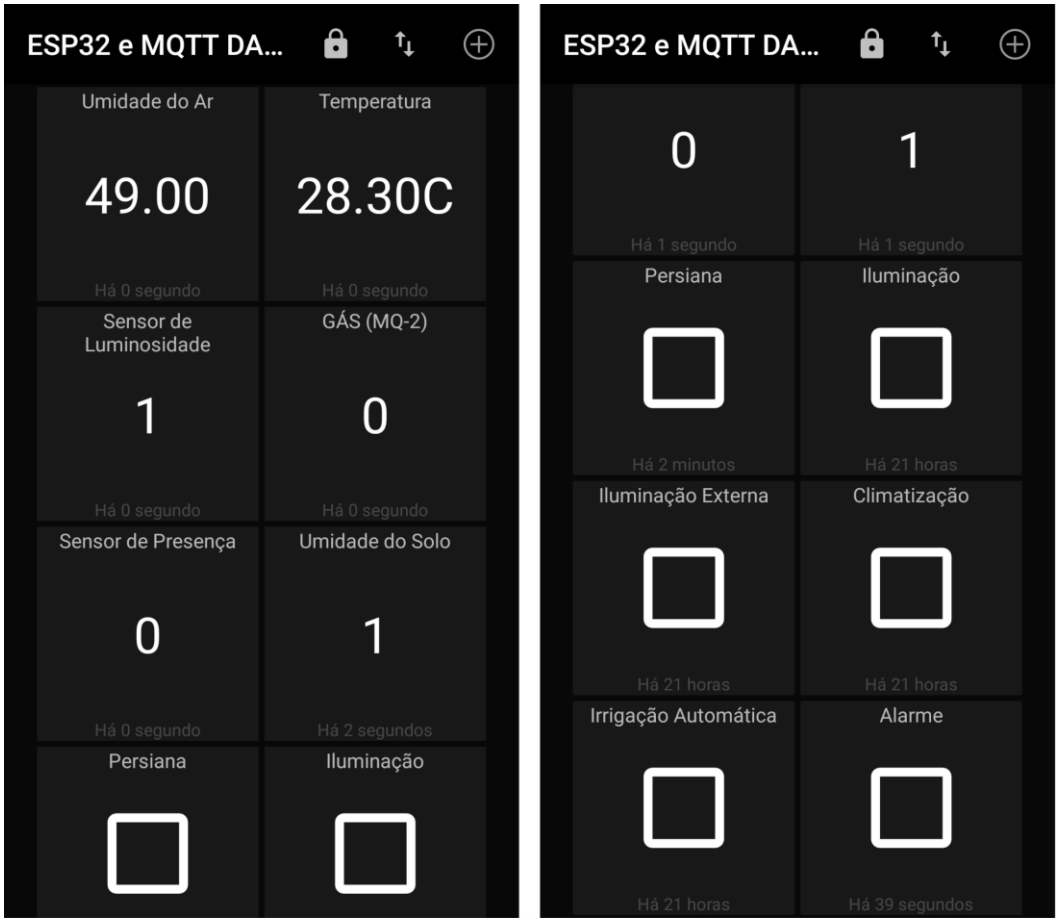


Fonte: Elaborado pelo autor, 2021.

4.3. Dashboard no MQTT DASH

Para criar a interface com o usuário foi utilizado o aplicativo para smartphones Android MQTT DASH. Ao adicionar os tópicos referentes a cada sensor e comandos foi possível realizar todo o controle e monitoramento através da *dashboard* que pode ser vista na Figura 31. Todos os tópicos utilizados estão disponíveis nas Tabelas 4 e 5.

Figura 31: Dashboard no MQTT DASH.



Fonte: Elaborado pelo autor, 2021.

Tabela 4: Publicações MQTT dos microcontroladores.

Microcontrolador	Publicações	Significado
ESP32	TOPICO_PUBLISH_TEMPERATURA_TCC	Valor de saída de temperatura do DHT11
	TOPICO_PUBLISH_UMIDADE_TCC	Valor de saída de umidade do DHT11
	TOPICO_PUBLISH_LDR_TCC	Saída do sensor de luminosidade
	TOPICO_PUBLISH_UMI_TCC	Saída do sensor de umidade do solo
ESP32-CAM	TOPICO_PUBLISH_GAS_TCC	Saída do sensor de gás MQ-2
	TOPICO_PUBLISH_PIR_TCC	Saída do sensor de presença HC-SR501

Fonte: Elaborado pelo autor, 2021.

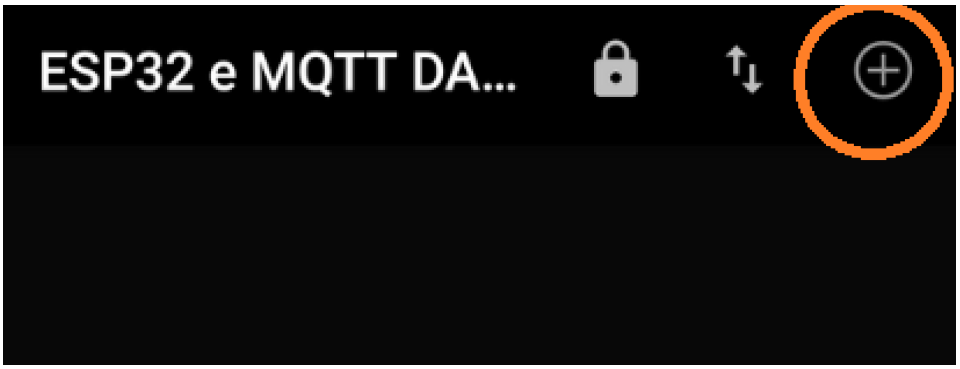
Tabela 5: Inscrições MQTT dos microcontroladores.

Microcontrolador	Inscrições	Significado
ESP32	TOPICO_SUBSCRIBE_SERVO	Comando para controlar o motor
	TOPICO_SUBSCRIBE_ILUM	Comando para controlar lâmpada
	TOPICO_SUBSCRIBE_LDR	Comando para acionar iluminação automática
	TOPICO_SUBSCRIBE_CLIMA	Comando para acionar irrigação automática
ESP32-CAM	TOPICO_SUBSCRIBE_ALARME	Comando para acionar o alarme

Fonte: Elaborado pelo autor, 2021.

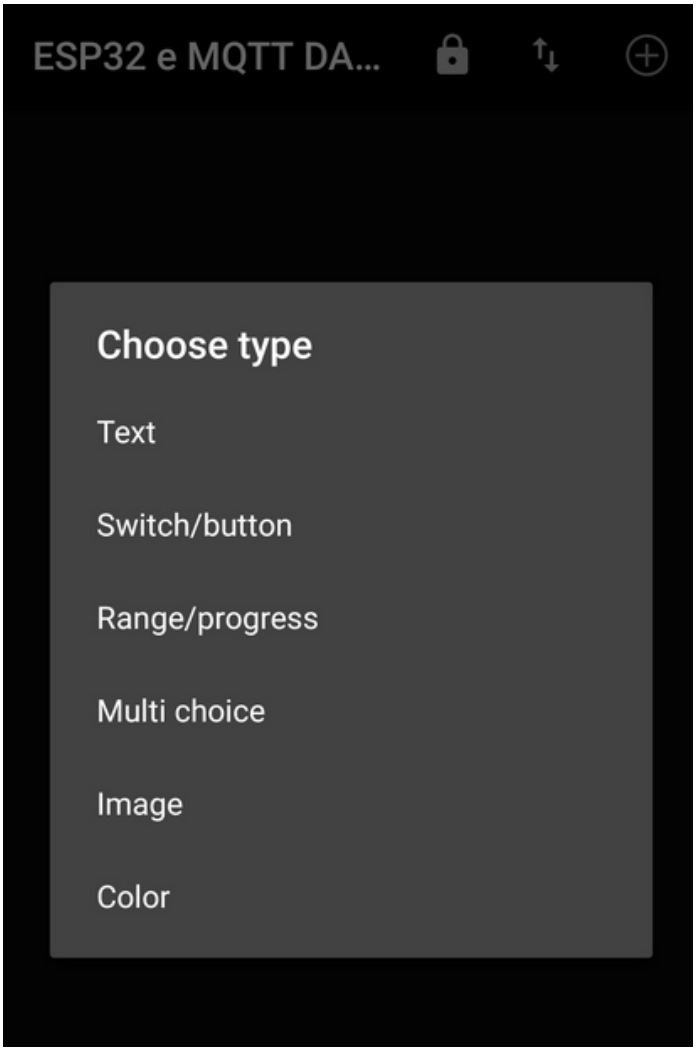
Para adicionar um novo elemento a *dashboard*, basta clicar no símbolo indicado na Figura 32 e, em seguida, escolher uma das opções listadas na Figura 33. Para exemplificar, na Figura 34 é apresentado como se realiza a adição de um elemento na *dashboard* relacionado ao sensor de temperatura do ambiente. Foi selecionado o tipo *Text* e adicionado o respectivo tópico. Também foi selecionado o nível da mensagem utilizando o protocolo MQTT, como já mencionado anteriormente, todas as aplicações foram utilizadas no nível 2.

Figura 32: Ícone para adicionar novo elemento a dashboard.



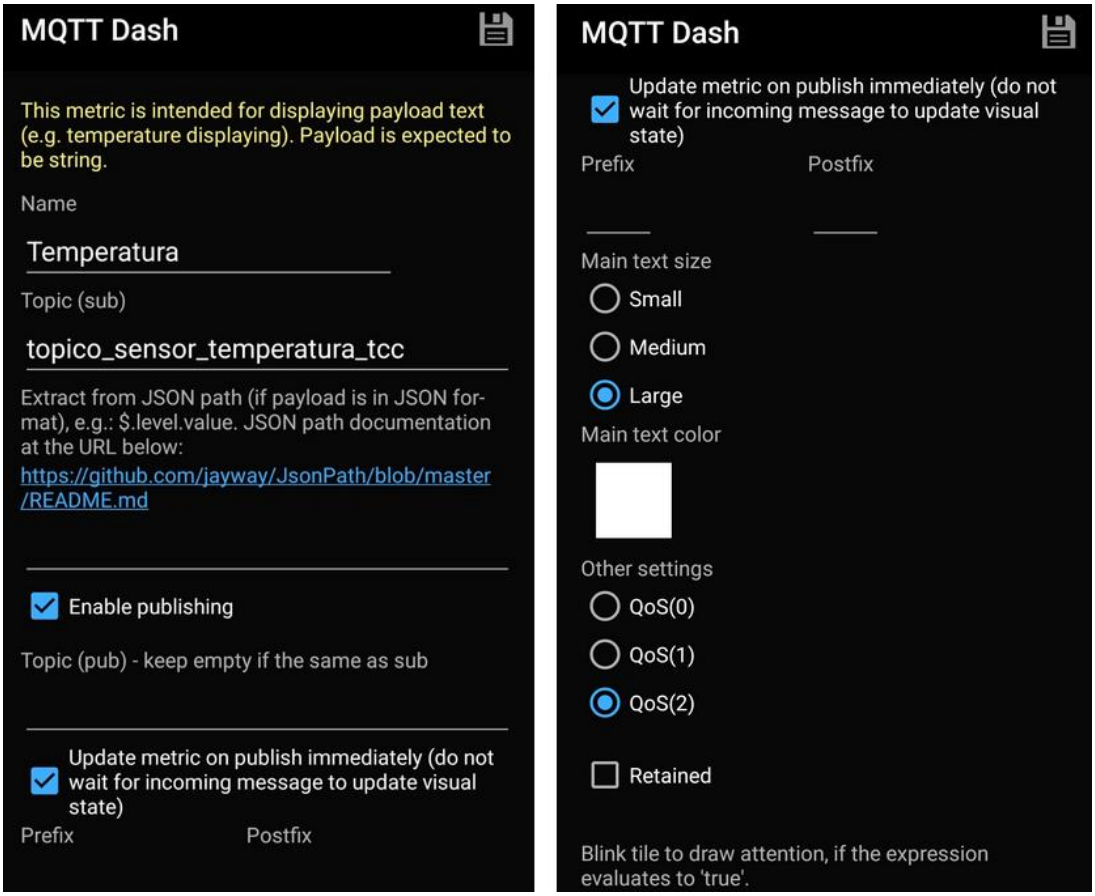
Fonte: Elaborado pelo autor, 2021.

Figura 33: Opções de elementos no MQTT DASH.



Fonte: Elaborado pelo autor, 2021.

Figura 34: Adicionando o elemento de temperatura do ambiente.



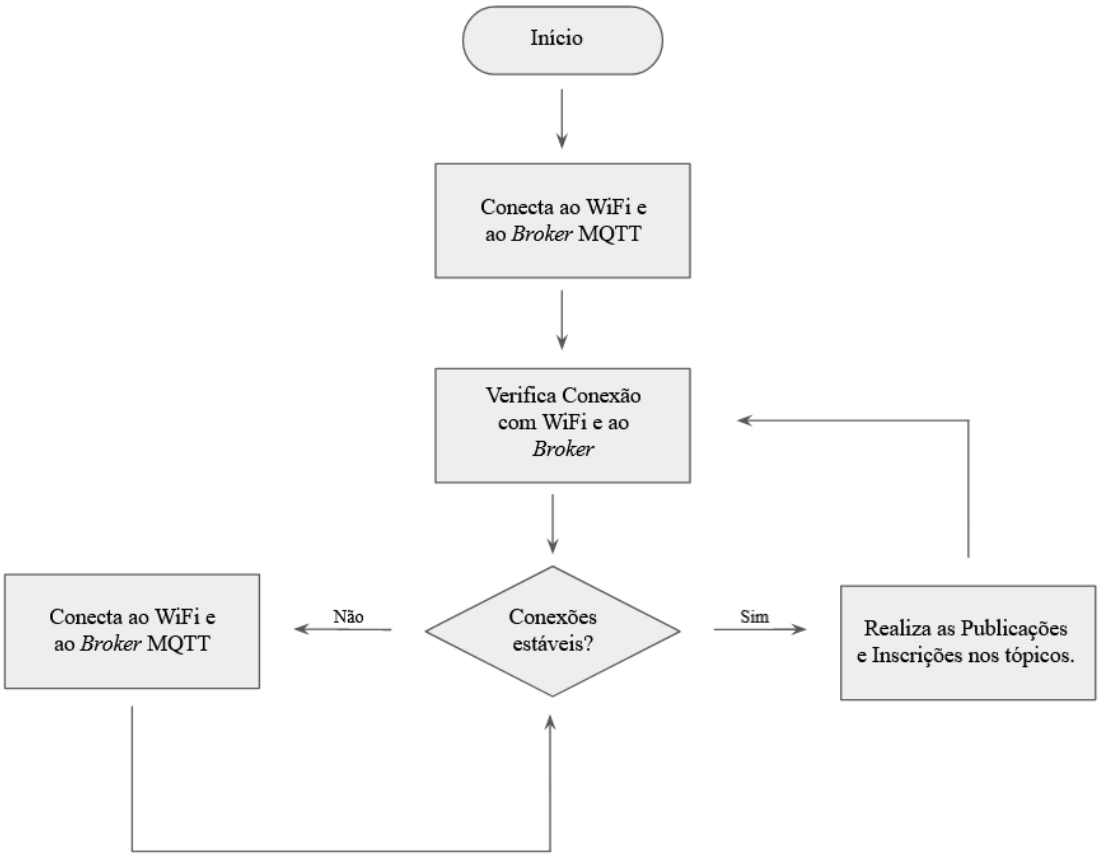
Fonte: Elaborado pelo autor, 2021.

Se tratando da implementação do MQTT nos códigos dos microcontroladores, foi utilizada a biblioteca *PubSubClient*. Primeiramente são definidos os tópicos que serão utilizados para publicação e para a inscrição, os tópicos usados neste trabalho são os referentes às Tabelas 4 e 5. Em seguida também é definido o ID MQTT, este é responsável por identificar o cliente na sessão. Quanto às funções utilizadas nos Anexos A e B relacionadas ao protocolo, primeiramente é utilizada a função "initMQTT" onde é definido os parâmetros de endereço e porta para a conexão ao *Broker*, um detalhe importante é que esta deve estar dentro da função principal *setup*.

Além disto, também foram criadas as funções "mqtt_callback", "VerificaConexoesWiFiEMQTT" e "reconnectMQTT" onde suas atribuições são respectivamente de ler alguma possível mensagem através dos tópicos de inscrição, verificar se a conexão com a internet e com o *Broker* está funcionando corretamente e reconectar ao *Broker* em caso de perda de conexão. Por fim, uma função da biblioteca de extrema importância utilizada dentro do *loop* é a MQTT.publish(), cuja finalidade é escrever nos tópicos de

publicação como, por exemplo, os valores obtidos nos sensores. Todo este processo está melhor descrito na Figura 35.

Figura 35: Fluxograma da lógica do protocolo nos microcontroladores.



Fonte: Elaborado pelo autor, 2021.

4.4. Análise de custos e viabilidade técnica

Para o estudo proposto, os custos com os materiais e equipamentos utilizados no protótipo desenvolvido foram de R$392,69, conforme mostrado na Tabela 6. Para realizar uma aplicação real de domótica os custos seriam mais elevados, considerando-se os custos adicionais referentes à quantidade/potência dos dispositivos elétricos presentes em uma residência e, sobretudo, aos recursos implementados em cada caso.

Tabela 6: Custo dos materiais utilizados no protótipo.

Número	Materiais	Quant.	Custo Unitário	Custo total	Fonte
1	ESP32	1	R$ 73,90	R$ 73,90	FILIPEFLOP (2021)
2	ESP32-CAM	1	R$ 89,00	R$ 89,00	ROBOCORE (2021)
3	Conversor USB / Serial - UART - PL2303	1	R$ 10,50	R$ 10,50	CURTOCIRCUITO (2021)
4	Sensor De Umidade De Solo	1	R$ 9,90	R$ 9,90	ELETROGATE (2021)
5	Módulo Sensor de Luminosidade Luz LDR	1	R$ 8,90	R$ 8,90	SARAVATI (2021)
6	Módulo Buzzer Ativo 3,3V a 5V - BP19	1	R$ 6,89	R$ 6,89	USINAINFO (2021)
7	Módulo Relé 5V 8 Canais	1	R$ 58,90	R$ 58,90	FILIPEFLOP (2021)
8	Sensor de Gás Inflamável MQ-2	1	R$ 19,90	R$ 19,90	FILIPEFLOP (2021)
9	Sensor de Movimento PIR	1	R$ 12,90	R$ 12,90	FILIPEFLOP (2021)
10	Módulo Sensor de Umidade e Temperatura DHT11	1	R$ 22,90	R$ 22,90	FILIPEFLOP (2021)
11	Micro Servo 9g SG90 TowerPro	1	R$ 21,90	R$ 21,90	FILIPEFLOP (2021)
12	Lâmpada 12V	4	R$ 0,85	R$ 03,40	Autor (2021)
13	Kit com 40 Jumpers Fêmea-Fêmea	1	R$ 9,90	R$ 9,90	FILIPEFLOP (2021)
14	Kit com 40 Jumpers Macho-Fêmea	1	R$ 9,90	R$ 9,90	FILIPEFLOP (2021)
15	Protoboard 830 pontos	1	R$ 33,90	R$ 33,90	FILIPEFLOP (2021)
	Total do orçamento			R$ 392,69	

Fonte: Elaborado pelo autor, 2021.

Ao considerar uma aplicação real, mesmo que o protótipo apresente um bom funcionamento, é necessário avaliar se é viável utilizá-lo em cada situação. No mercado, já existem dispositivos de fácil instalação e com aplicativos proprietários que funcionam de maneira *plug and play*, onde basta conectá-los a uma fonte de alimentação e fazer uma breve configuração, como por exemplo os interruptores e plugues inteligentes com Wi-Fi. Considerando que a proposta apresentada neste trabalho exige um conhecimento prévio de eletrônica e programação de microcontroladores para realizar a montagem e configuração da automação, existe maior dificuldade em sua implementação. Porém, o projeto se mostrou de baixo custo e é uma opção viável na automação residencial devido às funcionalidades apresentadas e, sobretudo, pela capacidade de customização dos recursos de acordo com as necessidades de cada usuário. É importante ressaltar que, para obter uma análise de viabilidade econômica seria necessário realizar um estudo de caso envolvendo as características de cada equipamento elétrico de uma residência específica e respectivo dimensionamento dos dispositivos atuadores do sistema de automação. Desta forma, seria possível realizar uma comparação efetiva com os dispositivos do tipo *plug and play* disponíveis no mercado para atender aos mesmos recursos propostos.

5. CONCLUSÕES

Através deste trabalho foi possível criar um protótipo de automação residencial utilizando o protocolo MQTT e o protocolo HTTP, com o auxílio do aplicativo MQTT DASH para a criação da *dashboard* de diversos sensores e os microcontroladores ESP32 e ESP32-CAM.

Foram alcançados os objetivos propostos de controle de climatização, sistema de vigilância por câmera, sistema de alarme, sistema de detecção de gás, monitoramento da umidade e temperatura do ambiente, sistema de irrigação automática, controle de iluminação interna, automatização de persianas e iluminação automatizada através de sensor de luminosidade.

Destaca-se a aplicação do protocolo MQTT na área de domótica, tendo em vista seu funcionamento em redes de baixa confiabilidade, alta latência e pouca banda disponível. Sendo que o nível de mensagem se mostra mais confiável com o QoS(2), pois há garantia do recebimento da mensagem e garantia que a mensagem não chegará múltiplas vezes.

O protótipo desenvolvido neste trabalho teve um custo total de 392,69 reais, porém para escalar o projeto de domótica a um nível de uma residência os custos seriam significativamente mais elevados, pois haveria uma adição e redimensionamento dos atuadores para serem capazes de acionar os equipamentos de uma residência, além de aumentar a quantidade de materiais. Tendo em vista alguns equipamentos de automação residencial disponíveis no mercado que funcionam de maneira *plug and play*, onde não é necessário nenhum conhecimento prévio em programação, estes equipamentos podem ser mais interessantes do ponto de facilidade de implementação.

Para trabalhos futuros podem ser adicionados novos sistemas de automação residencial e aplicar em uma residência, realizando uma análise de viabilidade econômica com o dimensionamento do sistema elétrico. Além disso, para o sistema de vigilância utilizando o ESP32-CAM poderia, por exemplo, ser utilizado um túnel para que o acesso às imagens também possa ser feito de fora da rede local.

REFERÊNCIAS BIBLIOGRÁFICAS

AURESIDE. **Casa ‘inteligente’ é cada vez mais realidade**. 2021. Acesso em: 08 out. 2021. Disponível em: <http://www.aureside.org.br/noticias/casa--inteligente--e-cada-vez-mais-realidade>.

BAUERMEISTER, G. **Como usar Servo Motor com Arduino**. 2018. Acesso em: 02 out. 2021. Disponível em: <https://blog.fazedores.com/como-usar-servo-motor-com-arduino/>.

BORGES, Geovany A. et al. **Desenvolvimento com microcontroladores Atmel AVR**. 2006.

CURTOCIRCUITO. **Conversor USB / Serial - UART - PL2303**. 2021. Acesso em: 15 nov. 2021. Disponível em: <https://www.curtocircuito.com.br/conversor-usb-serial-uart-pl2303.html/>.

ELETROGATE. **Sensor De Umidade De Solo**. 2021. Acesso em: 02 out. 2021. Disponível em: <https://www.eletrogate.com/modulo-sensor-de-umidade-de-solo>.

FILIPEFLOP. **ESP32 e MQTT DASH: controle e monitoramento através de um dashboard MQTT para Android**. 2021. Acesso em: 05 out. 2021. Disponível em: < https://www.filipeflop.com/blog/esp32-e-mqtt-dashboard-android/>.

FILIPEFLOP. **Jumpers Fêmea-Fêmea x40 Unidades**. 2021a. Acesso em: 20 nov. 2021. Disponível em: <https://www.filipeflop.com/produto/jumpers-femea-femea-x40-unidades/>.

FILIPEFLOP. **Jumpers Macho-Fêmea x40 Unidades**. 2021b. Acesso em: 20 nov. 2021. Disponível em: <https://www.filipeflop.com/produto/jumpers-macho-femea-x40-unidades/>.

FILIPEFLOP. **Micro Servo 9g SG90 TowerPro**. 2021c. Acesso em: 20 nov. 2021. Disponível em: <https://www.filipeflop.com/produto/micro-servo-9g-sg90-towerpro/>.

FILIPEFLOP. **Módulo Relé 5V 8 Canais**. 2021d. Acesso em: 20 nov. 2021. Disponível em: <https://www.filipeflop.com/produto/modulo-rele-5v-8-canais/>.

FILIPEFLOP. **Módulo Sensor de Umidade e Temperatura DHT11 KY-015.** 2021e. Acesso em: 20 nov. 2021. Disponível em: <https://www.filipeflop.com/produto/modulo-sensor-de-umidade-e-temperatura-dht11-ky-015/>.

FILIPEFLOP. **Módulo WiFi ESP32 Bluetooth**. 2021f. Acesso em: 20 nov. 2021. Disponível em: <https://www.filipeflop.com/produto/modulo-wifi-esp32-bluetooth/>.

FILIPEFLOP. **Protoboard 830 Pontos**. 2021g. Acesso em: 20 nov. 2021. Disponível em: <https://www.filipeflop.com/produto/protoboard-830-pontos/>.

FILIPEFLOP. **Sensor de Gás Inflamável MQ-2**. 2021h. Acesso em: 20 nov. 2021. Disponível em: <https://www.filipeflop.com/produto/sensor-de-gas-mq-2-inflamavel-e-fumaca/>.

FILIPEFLOP. **Sensor de Movimento Presença PIR**. 2021i. Acesso em: 20 nov. 2021. Disponível em: <https://www.filipeflop.com/produto/sensor-de-movimento-presenca-pir/>.

JEFFERSON. **Válvula Solenoide**. 2021. Acesso em: 20 nov. 2021. Disponível em: <https://www.jefferson.ind.br/conteudo/valvula-solenoide.html>.

JUNIOR, Sergio Luiz Stevan; FARINELLI, Felipe Adalberto. **DOMÓTICA-Automação Residencial e Casas Inteligentes com Arduíno e ESP8266**. Saraiva Educação SA, 2018.
MARTINS, Victor Ferreira et al. **Automação residencial usando protocolo MQTT, Node-RED e Mosquitto Broker com ESP32 e ESP8266**. 2019.

MELO, Rodrigo Cassiano da Silva et al. **Sistema de monitoramento de consumo de água utilizando o protocolo de comunicação MQTT**. 2018.

OLIVEIRA, Jailson. **Arduino, ESP32 e ESP8266 – Comparação**. XProjetos, 2019. Acesso em: 08 out. 2021. Disponível em: <https://xprojetos.net/arduino-esp32-e-esp8266-comparacao/>.

RANDOMNERDTUTORIALS. **ESP32-CAM AI-Thinker Pinout Guide: GPIOs Usage Explained**. 2021. Acesso em: 15 nov. 2021. Disponível em: <https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/>.

ROBOCORE. **ESP32-CAM - ESP32 com Câmera**. 2021. Acesso em: 08 out. 2021. Disponível em: <https://www.robocore.net/wifi/esp32-cam-esp32-com-camera>.

SANTOS, Bruno P. et al. **Internet das coisas: da teoria à prática**. 2016.

SANTOS, Jean Willian; LARA JUNIOR, Renato Capelin de. **Sistema de automatização residencial de baixo custo controlado pelo microcontrolador ESP32 e monitorado via smartphone**. 2019. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná.

SARAVATI. **Módulo Sensor de Luminosidade Luz LDR**. 2021. Acesso em: 02 out. 2021. Disponível em: <https://www.saravati.com.br/modulo-sensor-de-luminosidade-luz-ldr>.
STUDIOPIETERS. **ESP32 – PinOut**. 2021. Acesso em: 15 nov. 2021. Disponível em: <https://www.studiopieters.nl/esp32-pinout/>.

TEZA, Vanderlei Rabelo et al. **Alguns aspectos sobre a automação residencial: domótica**. 2002.

USINAINFO. **Módulo Buzzer Ativo 3,3V a 5V - BP19**. 2021. Acesso em: 02 out. 2021. Disponível em: <https://www.usinainfo.com.br/buzzer-arduino/modulo-buzzer-ativo-33v-a-5v-bp19-5831.html>.

WEBLINK. **Protocolos de Rede: O Que São, Como Funcionam e Tipos de Protocolos de Internet**. 2020. Acesso em: 15 nov. 2021. Disponível em: <https://www.weblink.com.br/blog/tecnologia/conheca-os-principais-protocolos-de-internet/>

ANEXOS

Anexo A – Códigos ESP32

```
#include <Servo.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <ETH.h>
#include <WiFi.h>
#include <WiFiAP.h>
#include <WiFiClient.h>
#include <WiFiGeneric.h>
#include <WiFiMulti.h>
#include <WiFiScan.h>
#include <WiFiServer.h>
#include <WiFiSTA.h>
#include <WiFiType.h>
#include <WiFiUdp.h>
#include <PubSubClient.h>

//Sensores
#define DHTPIN 26     //GPIO que está ligado o pino de dados do sensor DHT11
#define PIN_LDR 14   //GPIO que está ligado o pino de dados do sensor de luminosidade
#define UMI 33     //GPIO que está ligado o pino de dados do sensor umidade do solo
#define DHTTYPE DHT11   //sensor em utilização: DHT11
#define SERVO 22     //GPIO que está ligado o servo motor
#define ILUMI 23     //GPIO que está ligado o rele da iluminação
#define RELE_LDR 21   //GPIO que está ligado o rele da iluminação controlado por ldr
#define RELE_CLIMA 25     //GPIO que está ligado o rele de climatização
#define RELE_IRRI 19     //GPIO que está ligado o rele de irrigacao

/* Defines do MQTT */

#define TOPICO_PUBLISH_TEMPERATURA_TCC   "topico_sensor_temperatura_tcc"
#define TOPICO_PUBLISH_UMIDADE_TCC       "topico_sensor_umidade_tcc"
#define TOPICO_PUBLISH_LDR_TCC       "topico_sensor_ldr_tcc"
#define TOPICO_PUBLISH_UMI_TCC       "topico_sensor_umi_tcc"

#define TOPICO_SUBSCRIBE_SERVO         "topico_controla_servo"
#define TOPICO_SUBSCRIBE_ILUM         "topico_controla_luz"
#define TOPICO_SUBSCRIBE_LDR         "topico_controla_ldr"
#define TOPICO_SUBSCRIBE_CLIMA         "topico_controla_clima"
#define TOPICO_SUBSCRIBE_IRRIGACAO         "topico_controla_irrigacao"

//ID MQTT
#define ID_MQTT  "esp32_mqtt"

/* Variaveis, constantes e objetos globais */
DHT dht(DHTPIN, DHTTYPE);

const char* SSID = "*******"; // SSID / nome da rede WI-FI que deseja se conectar
const char* PASSWORD = "*******"; // Senha da rede WI-FI que deseja se conectar

const char* BROKER_MQTT = "broker.emqx.io";//URL do broker MQTT a se deseja utilizar
int BROKER_PORT = 1883; // Porta do Broker MQTT

Servo s; // Variável Servo
int pos; // Posição Servo

int luz_ldr = 0;
int luz = 0;
int clima = 0;
int irrigacao = 0;

//Variáveis e objetos globais
WiFiClient espClient; // Cria o objeto espClient
PubSubClient MQTT(espClient); // Instancia o Cliente MQTT passando o objeto espClient

```

```
66  //Protótipos de funções
67  float faz_leitura_temperatura(void);
68  float faz_leitura_umidade(void);
69  int faz_leitura_ldr(void);
70  int faz_leitura_umi(void);
71  void abre_persiana(void);
72  void fecha_persiana(void);
73  void luz_controlada_ldr(void);
74  void clima_controlada_dht(void);
75  void irrigacao_controlada(void);
76  void initWiFi(void);
77  void initMQTT(void);
78  void mqtt_callback(char* topic, byte* payload, unsigned int length);
79  void reconnectMQTT(void);
80  void reconnectWiFi(void);
81  void VerificaConexoesWiFIEMQTT(void);
82
83  //Funções de automação
84
85  //função de irrigação automatica
86  void irrigacao_controlada(void){
87       if((irrigacao == 1) && (digitalRead(UMI) == HIGH)){
88         digitalWrite(RELE_IRRI, LOW);
89       }
90       else{
91         digitalWrite(RELE_IRRI, HIGH);
92       }
93  }
94
95  //função para ativa rele de climatização
96  void clima_controlada_dht(void){
97       if((clima == 1) && (dht.readTemperature() >= 25)){
98         digitalWrite(RELE_CLIMA, LOW);
99       }
100      else{
101        digitalWrite(RELE_CLIMA, HIGH);
102      }
103 }
104
105 //função para ativar rele da iluminação controlada por sensor de luminosidade
106 void luz_controlada_ldr(void){
107      if((luz_ldr == 1) && (digitalRead(PIN_LDR) == HIGH)){
108        digitalWrite(RELE_LDR, LOW);
109      }
110      else{
111        digitalWrite(RELE_LDR, HIGH);
112      }
113 }
114
115 //função para abrir persiana
116 void abre_persiana(void)
117 {
118      for(pos = 0; pos < 90; pos++)
119             {
120             s.write(pos);
121             delay(15);
122             }
123 }
124
125 //função para fechar persiana
126 void fecha_persiana(void)
127 {
128      for(pos = 90; pos >= 0; pos--)
129             {
130             s.write(pos);
131             delay(15);
```

```
            }
}

//função para ler temperatura do ar
float faz_leitura_temperatura(void)
{
    float t = dht.readTemperature();
    float result;

    if (! (isnan(t)) )
        result = t;
    else
        result = -99.99;

    return result;
}

//função para ler umidade do solo
int faz_leitura_umi(void)
{

    int valor;
  if(digitalRead(UMI) == HIGH){ //SE LEITURA DO PINO FOR IGUAL A 1 (HIGH), FAZ
    valor = 1;
  }
  else{ //SENÃO, FAZ
    valor = 0;
  }
    return valor;
}

//função para ler sensor de luminosidade
int faz_leitura_ldr(void)
{
  int valor;
  if(digitalRead(PIN_LDR) == HIGH){ //SE LEITURA DO PINO FOR IGUAL A 1 (HIGH), FAZ
    valor = 1;
  }
  else{ //SENÃO, FAZ
    valor = 0;
  }
    return valor;
}

//função para ler umidade do ar
float faz_leitura_umidade(void)
{
    float h = dht.readHumidity();
    float result;

    if (! (isnan(h)) )
        result = h;
    else
        result = -99.99;

    return result;
}

//Função: inicializa e conecta-se na rede WI-FI desejada
void initWiFi(void)
{
    delay(10);
    Serial.println("------Conexao WI-FI------");
    Serial.print("Conectando-se na rede: ");
    Serial.println(SSID);
    Serial.println("Aguarde");
```

```
198
199     reconnectWiFi();
200 }
201
202 //Função: inicializa parâmetros de conexão MQTT
203 void initMQTT(void)
204 {
205     MQTT.setServer(BROKER_MQTT, BROKER_PORT);
206     //informa qual broker e porta deve ser conectado
207     MQTT.setCallback(mqtt_callback);
208     //atribui função de callback
209 }
210
211 // Função: função de callback
212 void mqtt_callback(char* topic, byte* payload, unsigned int length)
213 {
214
215     //servo
216     String messageTemp;
217
218     for (int i = 0; i < length; i++) {
219     Serial.print((char)payload[i]);
220     messageTemp += (char)payload[i];
221     }
222     Serial.println();
223
224     //controlar motor da persiana
225     if (String(topic) == "topico_controla_servo") {
226          Serial.print("Changing output to ");
227           if (messageTemp == "1") {
228             Serial.println("on");
229             abre_persiana();
230            }
231           else if (messageTemp == "0") {
232             Serial.println("off");
233             fecha_persiana();
234           }
235      }
236
237     //aciona iluminação
238     if (String(topic) == "topico_controla_luz") {
239          Serial.print("Changing output to ");
240           if (messageTemp == "1") {
241             Serial.println("on");
242             digitalWrite(ILUMI, LOW);
243            }
244           else if (messageTemp == "0") {
245             Serial.println("off");
246             digitalWrite(ILUMI, HIGH);
247           }
248      }
249
250     //aciona iluminação controlada por ldr
251     if (String(topic) == "topico_controla_ldr") {
252          Serial.print("Changing output to ");
253           if (messageTemp == "1") {
254             Serial.println("on");
255             luz_ldr = 1;
256            }
257           else if (messageTemp == "0") {
258             Serial.println("off");
259             luz_ldr = 0;
260           }
261      }
262
263     //aciona climatização automatica
```

```
264     if (String(topic) == "topico_controla_clima") {
265          Serial.print("Changing output to ");
266           if (messageTemp == "1") {
267             Serial.println("on");
268             clima = 1;
269            }
270           else if (messageTemp == "0") {
271             Serial.println("off");
272             clima = 0;
273           }
274      }
275
276     //aciona irrigação automatica
277     if (String(topic) == "topico_controla_irrigacao") {
278          Serial.print("Changing output to ");
279           if (messageTemp == "1") {
280             Serial.println("on");
281             irrigacao = 1;
282            }
283           else if (messageTemp == "0") {
284             Serial.println("off");
285             irrigacao = 0;
286           }
287      }
288
289
290 }
291
292 // Função: reconecta-se ao broker MQTT
293 void reconnectMQTT(void)
294 {
295     while (!MQTT.connected())
296     {
297         Serial.print("* Tentando se conectar ao Broker MQTT: ");
298         Serial.println(BROKER_MQTT);
299         if (MQTT.connect(ID_MQTT))
300         {
301             Serial.println("Conectado com sucesso ao broker MQTT!");
302             MQTT.subscribe(TOPICO_SUBSCRIBE_SERVO);
303             MQTT.subscribe(TOPICO_SUBSCRIBE_ILUM);
304             MQTT.subscribe(TOPICO_SUBSCRIBE_LDR);
305             MQTT.subscribe(TOPICO_SUBSCRIBE_CLIMA);
306             MQTT.subscribe(TOPICO_SUBSCRIBE_IRRIGACAO);
307         }
308         else
309         {
310             Serial.println("Falha ao reconectar no broker.");
311             Serial.println("Havera nova tentatica de conexao em 2s");
312             delay(2000);
313         }
314     }
315 }
316
317 // Função: verificar o funcionamento do wifi e mqtt
318 void VerificaConexoesWiFIEMQTT(void)
319 {
320     if (!MQTT.connected())
321         reconnectMQTT(); //se não há conexão com o Broker, a conexão é refeita
322
323      reconnectWiFi(); //se não há conexão com o WiFI, a conexão é refeita
324 }
325
326 // Função: reconectar ao wifi
327 void reconnectWiFi(void)
328 {
329     //se já está conectado a rede WI-FI, nada é feito.
```

```
330     //Caso contrário, são efetuadas tentativas de conexão
331     if (WiFi.status() == WL_CONNECTED)
332         return;
333
334     WiFi.begin(SSID, PASSWORD); // Conecta na rede WI-FI
335
336     while (WiFi.status() != WL_CONNECTED)
337     {
338         delay(100);
339         Serial.print(".");
340     }
341
342     Serial.println();
343     Serial.print("Conectado com sucesso na rede ");
344     Serial.print(SSID);
345     Serial.println("IP obtido: ");
346     Serial.println(WiFi.localIP());
347 }
348
349 /* Função de setup */
350 void setup()
351 {
352     Serial.begin(115200);
353
354     //definir os pinos
355     digitalWrite(ILUMI,HIGH);
356     pinMode(PIN_LDR, INPUT);
357     pinMode(UMI, INPUT);
358     pinMode(SERVO, OUTPUT);
359     pinMode(RELE_CLIMA, OUTPUT);
360     pinMode(RELE_LDR, OUTPUT);
361     pinMode(RELE_IRRI, OUTPUT);
362     pinMode(ILUMI, OUTPUT);
363
364     s.attach(SERVO);
365     s.write(0); // Inicia motor posição zero
366
367     /* Inicializacao do sensor de temperatura */
368     dht.begin();
369
370     /* Inicializa a conexao wi-fi */
371     initWiFi();
372
373     /* Inicializa a conexao ao broker MQTT */
374     initMQTT();
375 }
376
377 /* Loop principal */
378 void loop()
379 {
380     char temperatura_str[10] = {0};
381     char umidade_str[10]     = {0};
382     char ldr_str[10] = {0};
383     char umi_str[10] = {0};
384
385     VerificaConexoesWiFIEMQTT();
386
387     // strings que serão mandadas para o dashboard
388     sprintf(temperatura_str,"%.2fC", faz_leitura_temperatura());
389     sprintf(umidade_str,"%.2f", faz_leitura_umidade());
390     sprintf(ldr_str,"%d", faz_leitura_ldr());
391     sprintf(umi_str,"%d", faz_leitura_umi());
392
393     //publicações
394     MQTT.publish(TOPICO_PUBLISH_TEMPERATURA_TCC, temperatura_str);
395     MQTT.publish(TOPICO_PUBLISH_UMIDADE_TCC, umidade_str);
```

```
395     MQTT.publish(TOPICO_PUBLISH_UMIDADE_TCC, umidade_str);
396     MQTT.publish(TOPICO_PUBLISH_LDR_TCC, ldr_str);
397     MQTT.publish(TOPICO_PUBLISH_UMI_TCC, umi_str);
398
399     clima_controlada_dht();
400     luz_controlada_ldr();
401     irrigacao_controlada();
402     // keep-alive
403     MQTT.loop();
404     // Refaz o ciclo após 2 segundos
405     delay(2000);
406 }
407
```

Anexo B – Códigos ESP32-CAM

```
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_http_server.h"
#include <PubSubClient.h>

//sensores
#define MQ_GAS 12     //GPIO que está ligado o pino de dados do sensor MQ-2
#define PIR 2     //GPIO que está ligado o pino de dados do sensor PIR
#define BUZZER 13     //GPIO que está ligado o buzzer

//Configuração da rede WiFi
const char* ssid = "*********";
const char* password = "*********";

#define PART_BOUNDARY "123456789000000000000987654321"

/* Defines do MQTT */
#define ID_MQTT  "esp32_cam_mqtt"     //ID MQTT

#define TOPICO_PUBLISH_GAS_TCC       "topico_sensor_gas_tcc"
#define TOPICO_PUBLISH_PIR_TCC       "topico_sensor_pir_tcc"

#define TOPICO_SUBSCRIBE_ALARME       "topico_controla_alarme"

const char* BROKER_MQTT = "broker.emqx.io";//URL do broker MQTT que se deseja utilizar
int BROKER_PORT = 1883; // Porta do Broker MQTT

WiFiClient espClient; // Cria o objeto espClient
PubSubClient MQTT(espClient); //Instancia o Cliente MQTT passando o objeto espClient

int alarme = 0;

//prototipos das funções
void initMQTT(void);
void mqtt_callback(char* topic, byte* payload, unsigned int length);
void reconnectMQTT(void);
void reconnectWiFi(void);
void VerificaConexoesWiFIEMQTT(void);
void aciona_buzzer(void);
int faz_leitura_pir(void);
int faz_leitura_gas(void);

// Configuração do modelo de câmera (CAMERA_MODEL_AI_THINKER)
  #define PWDN_GPIO_NUM     32
  #define RESET_GPIO_NUM    -1
  #define XCLK_GPIO_NUM      0
  #define SIOD_GPIO_NUM     26
  #define SIOC_GPIO_NUM     27
  #define Y9_GPIO_NUM       35
  #define Y8_GPIO_NUM       34
  #define Y7_GPIO_NUM       39
  #define Y6_GPIO_NUM       36
  #define Y5_GPIO_NUM       21
  #define Y4_GPIO_NUM       19
  #define Y3_GPIO_NUM       18
  #define Y2_GPIO_NUM        5
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     23
  #define PCLK_GPIO_NUM     22

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";

httpd_handle_t stream_httpd = NULL;

static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char * part_buf[64];

  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
  if(res != ESP_OK){
    return res;
  }

```

```
  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      res = ESP_FAIL;
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
          _jpg_buf = fb->buf;
        }
      }
    }
    if(res == ESP_OK){
      size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if(_jpg_buf){
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;

  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };

  //Serial.printf("Starting web server on port: '%d'\n", config.server_port);
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &index_uri);
  }
}
```

```
void aciona_buzzer(void)
{
     if(((alarme == 1) && (digitalRead(PIR) == HIGH))||(digitalRead(MQ_GAS) == LOW)){
       digitalWrite(BUZZER, LOW);
       delay(2000);                // Espera 2 segundos
       digitalWrite(BUZZER, HIGH);
     }else{
       digitalWrite(BUZZER, HIGH);
     }
}
int faz_leitura_gas(void)
{

    int valor;
  if(digitalRead(MQ_GAS) == HIGH){ //SE LEITURA DO PINO FOR IGUAL A 1 (HIGH), FAZ
    valor = 0;
  }
  else{ //SENÃO, FAZ
    valor = 1;
  }
    return valor;
}

int faz_leitura_pir(void)
{

  int valor;
  if(digitalRead(PIR) == HIGH){ //SE LEITURA DO PINO FOR IGUAL A 1 (HIGH), FAZ
    valor = 1;
  }
  else{ //SENÃO, FAZ
    valor = 0;
  }
    return valor;
}

void initMQTT(void)
{
    MQTT.setServer(BROKER_MQTT, BROKER_PORT); //informa broker e porta
    MQTT.setCallback(mqtt_callback);     //atribui função de callback
}

void mqtt_callback(char* topic, byte* payload, unsigned int length)
{

    //servo
    String messageTemp;

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

    //aciona alarme
    if (String(topic) == "topico_controla_alarme") {
         Serial.print("Changing output to ");
          if (messageTemp == "1") {
            Serial.println("on");
            alarme = 1;
           }
          else if (messageTemp == "0") {
            Serial.println("off");
            alarme = 0;
          }
```

```
     }

}

void reconnectMQTT(void)
{
    while (!MQTT.connected())
    {
        Serial.print("* Tentando se conectar ao Broker MQTT: ");
        Serial.println(BROKER_MQTT);
        if (MQTT.connect(ID_MQTT))
        {
            Serial.println("Conectado com sucesso ao broker MQTT!");
            MQTT.subscribe(TOPICO_SUBSCRIBE_ALARME);
        }
        else
        {
            Serial.println("Falha ao reconectar no broker.");
            Serial.println("Havera nova tentatica de conexao em 2s");
            delay(2000);
        }
    }
}
void VerificaConexoesWiFIEMQTT(void)
{
    if (!MQTT.connected())
        reconnectMQTT(); //se não há conexão com o Broker, a conexão é refeita

     reconnectWiFi(); //se não há conexão com o WiFI, a conexão é refeita
}

void reconnectWiFi(void)
{
    //se já está conectado a rede WI-FI, nada é feito.
    //Caso contrário, são efetuadas tentativas de conexão
    if (WiFi.status() == WL_CONNECTED)
        return;

    WiFi.begin(ssid, password); // Conecta na rede WI-FI

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(100);
        Serial.print(".");
    }

    Serial.println();
    Serial.print("Conectado com sucesso na rede ");
    Serial.print(ssid);
    Serial.println("IP obtido: ");
    Serial.println(WiFi.localIP());
}



void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);

  Serial.begin(115200);
  Serial.setDebugOutput(false);

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
```

```
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_UXGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;

  // Iniciação da câmera
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  // Conexão WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Início da transmissão no servidor Web
  startCameraServer();
  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.print(WiFi.localIP());

  /* Inicializa a conexao ao broker MQTT */
  initMQTT();

  pinMode(BUZZER, OUTPUT);
  digitalWrite(BUZZER,HIGH);
  pinMode(MQ_GAS, INPUT);
  pinMode(PIR, INPUT);
}

void loop() {

  char gas_str[10] = {0};
  char pir_str[10] = {0};

  VerificaConexoesWiFIEMQTT();

  sprintf(gas_str,"%d", faz_leitura_gas());
  sprintf(pir_str,"%d", faz_leitura_pir());

  MQTT.publish(TOPICO_PUBLISH_GAS_TCC, gas_str);
  MQTT.publish(TOPICO_PUBLISH_PIR_TCC, pir_str);

  aciona_buzzer();

```

```
344
345   /* keep-alive da comunicação com broker MQTT */
346   MQTT.loop();
347
348   delay(2000);
349 }
350
351
```