SILVIO RAFAEL LIMA SANTOS

DESENVOLVIMENTO DE UM SISTEMA DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO CONCEITOS E TECNOLOGIAS DE INTERNET DAS COISAS

LAVRAS – MG

2021

SILVIO RAFAEL LIMA SANTOS

DESENVOLVIMENTO DE UM SISTEMA DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO CONCEITOS E TECNOLOGIAS DE INTERNET DAS COISAS

Relatório Técnico apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. Dr. Neumar Costa Malheiros

Orientador

LAVRAS – MG

2021

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Santos, Silvio Rafael Lima

Desenvolvimento de um sistema de automação residencial utilizando conceitos e tecnologias de Internet das Coisas / Silvio Rafael Lima Santos. 2^a ed. rev., atual. e ampl. – Lavras : UFLA, 2021.

51 p. : il.

Tese(graduação)–Universidade Federal de Lavras, 2021.

Orientador: Prof. Dr. Neumar Costa Malheiros.

Bibliografia.

1. TCC. 2. Monografia. 3. Dissertação. 4. Tese. 5. Trabalho Científico – Normas. I. Universidade Federal de Lavras. II. Título.

CDD-808.066

RESUMO

Com os novos horizontes proporcionados pela evolução da tecnologia na área de Internet das Coisas (IoT - Internet of Things), é cada vez mais possível e necessária a realização da automação das atividades antes realizadas manualmente pelas pessoas. Dentro de uma residência existem inúmeras atividades que são passíveis de serem automatizadas, monitoradas e controladas através das tecnologias da Internet das Coisas. Para isso é necessário aplicações capazes de serem escaláveis, inteligentes e que atendam as demandas específicas das diferentes necessidades de automações residenciais. Este trabalho propõe uma solução para o monitoramento e a automação de equipamentos no âmbito de uma residência, utilizando tecnologias IoT. Assim, foram realizados o projeto e o desenvolvimento de um sistema de automação de um dispositivo de refrigeração e um dispositivo de irrigação residencial, através do uso de tecnologias especializadas para esse tipo de aplicação, a saber: o protocolo Message Queuing Telemetry Transport (MQTT), em seu formato *publish/subscribe*, o serviço If This Then That (IFTTT), e a integração com o aplicativo de mensagens, Telegram.

Palavras-chave: Automação Residencial. Internet das Coisas. MQTT. IFTTT.

ABSTRACT

New horizons have been opened by the technological advances in the area of the Internet of Things. Thus, it is increasingly possible and necessary to automatize manual daily activities done by humans. Many activities can be automatized in a house. These activities can also be monitored and controlled by the Internet of Things. Nevertheless, to do it, it is necessary to use intelligent and scalable applications that are ready to attend to the specific demands of residential automation. Using concepts of the Internet of Things, this undergraduate thesis proposes a solution to the problem of monitoring and automating residential activities. Therefore, an automatic system of refrigeration and irrigation was generated by this project. This system was created using specialized protocols to communicate, MQTT with its format of *publish/subscribe*, IFTTT, and its integration with message services of Telegram.

Keywords: Smart Home. Internet of Things. MQTT. IFTTT.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Um desejo comum dos indivíduos é o de ter mais conforto e comodidade nas atividades recorrentes que geralmente são feitas de forma manual e trabalhosa. Juntamente com esses motivos, a escassez de tempo na rotina das pessoas na sociedade moderna aumenta o desejo por mais facilidade na execução de tais atividades. Assim, é natural que, com a evolução da tecnologias, sejam apresentadas novas ferramentas que possibilitem mais comodidade nas tarefas repetitivas do cotidiano.

Nesse contexto, a automação está cada vez mais presente nas rotinas de residências, como uma solução para as necessidades que requerem processos facilitadores. Seja, por exemplo, através de robôs aspiradores, que varrem a residência enquanto as pessoas ficam livres para fazer outras atividades, ou portões inteligentes que fornecem a comodidade do dono da residência não ter que abrir manualmente o seu portão quando desejar entrar em sua garagem.

Tendo em vista a ampla diversidade de objetos em uma residência que têm algum tipo de ação manual, o agente humano tem que interagir com o meio para obter um resultado desejado diversas vezes. Assim, automação residencial pode ser feita em diversos elementos que compõem uma residencia, desde os objetos mais usuais como lâmpadas até mesmo os mais particulares como uma produção de cerveja artesanal.

Neste contexto, para este trabalho foram escolhidos dois equipamentos, como estudos de caso, para propor soluções de automatização: um dispositivo de fermentação de cerveja artesanal e um dispositivo de irrigação de uma horta residencial. Ambos os dispositivos possuem elementos que comumente demandam ação humana. Por exemplo, no dispositivo de fermentação, é necessário que o humano se locomova ao local em que o dispositivo se encontra e monitore os valores de temperatura, bem como manualmente programe a temperatura alvo. Assim, a pessoa se vê presa uma vez que precisa estar fisicamente presente para monitorar

e controlar a fermentação. Já no dispositivo de irrigação, o agente humano precisa manualmente irrigar as plantas, sempre levando em conta a luminosidade do momento e a umidade do solo, afim de definir o melhor momento para realizar a irrigação.

A solução proposta nesse trabalho visa o uso de conceitos e tecnologias de Internet das Coisas (IoT) (SANTOS et al., 2016) para remover a necessidade da presença física do agente humano em um ambiente residencial genérico. Dessa forma, um indivíduo poderia controlar e monitorar remotamente dispositivos, bem como definir comportamentos automáticos.

Neste contexto, foi projetado e desenvolvido[1] um sistema de monitoramento e controle do meio físico, utilizando-se o protocolo de comunicação Message Queuing Telemetry Transport (MQTT)[2]. Como estudo de caso, a solução foi aplicada ao controle de dispositivos de irrigação e fermentação. Esses dispositivos foram integrados ao serviço If This Then That (IFTTT)[3], sendo que para essa integração se fez necessário o uso de um servidor capaz de comunicação com IFTTT, utilizando o protocolo Hypertext Transfer Protocol (HTTP). Além disso, foi utilizado o aplicativo de comunicação Telegram[4] como aplicação cliente para o usuário, onde este pode controlar todo o sistema através do envio de mensagens no aplicativo.

Esse trabalho está estruturado como descrito a seguir. No Capítulo 2, aborda-se a contextualização teórica das tecnologias utilizadas, adentrando em um certo nível de profundidade sobre as características de cada uma. Já no Capítulo 3 é feito a especificação do escopo do problema abordado nesse trabalho. Em seguida, no Capítulo 4, descreve-se a forma como essas tecnologias foram relacionadas e as características da construção do projeto como um todo. E, por fim, no Capítulo 5, apresentam-se as considerações finais, bem como as dificuldades

[1] https://github.com/dcc-ufla/smart-home-mqtt
[2] https://mqtt.org
[3] https://ifttt.com
[4] https://web.telegram.org/k/

enfrentadas durante o desenvolvimento do projeto e um panorama de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo, são descritos os principais conceitos e tecnologias utilizados na implementação da solução de automação residencial, sendo essas tecnologias o protocolo HTTP, a plataforma IFTTT e por fim o protocolo MQTT.

2.1 Fundamentos do HTTP

O Hypertext Transfer Protocol (HTTP) é um protocolo de comunicação para sistemas de informação, que permite obter, através de transferências, recursos como documentos HTML (KUROSE; ROSS, 2013). Os documentos transferidos são geralmente constituídos de outros sub itens, como texto, imagens, *scripts*, entre outros.

O protocolo funciona na base de requisições-respostas em um modelo cliente-servidor. As solicitações, denominadas requisições (*requests*), são submetidas pelos clientes ao servidor. Então, o servidor processa a requisição, adicionando recursos ou quaisquer itens de interesse do cliente, e envia uma mensagem de resposta (*response*) ao cliente. Essa resposta tem em si as informações sobre o estado da requisição e pode também conter os conteúdos solicitados.

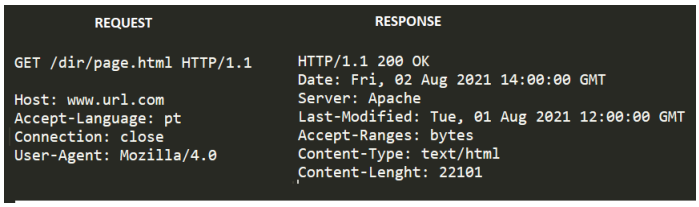
As requisições tem como destino um servidor HTTP. Entre o servidor e o cliente existem diversos outras entidades que executam operações, essas entidades ficam escondidas nas camadas da rede, estando o HTTP na camada de aplicação no topo da pilha de protocolos da *Internet*.

2.1.1 Estrutura da mensagem

Existem dois tipos de mensagens, requisições e respostas, como demonstrados na Figura 2.1, cada uma com seu próprio formato. As mensagens trocadas através desse protocolo são compostas de cabeçalho e corpo. Dentro da mensagem, o cabeçalho se faze presente, sendo possível ter uma ou mais linhas para os dados, tendo obrigatoriamente uma linha em branco que determina o fim do cabe-

çalho, separando-o das linhas que compõem o corpo da mensagem. O corpo da mensagem é opcional.

Figura 2.1 – Exemplo de resposta e requisições HTTP



Fonte: Do autor (2021).

O cabeçalho da mensagem (*header*) é utilizado para transmitir informações adicionais, existindo quatro tipos diferentes de cabeçalhos, sendo esses, *general-header*, *request-header*, *response-header* e *entity-header*. Esses cabeçalhos podem transmitir informações sobre a mensagem, configurações de clientes, os formatos que devem compor a resposta, entre outras.

O cabeçalho também pode conter informações que descrevem o tipo de informação que está sendo enviado ou o tamanho em *bytes* do conteúdo do corpo.

O corpo da mensagem de resposta pode compor os recursos que foram requisitados pelo cliente ou mensagens de erro, caso o objeto desejado não seja possível ser retornado. Já na requisição, o corpo consiste nas informações que serão enviadas do cliente para o servidor.

2.1.2 Métodos de requisição

O protocolo HTTP define um conjunto de métodos que descrevem ações a serem feitas em um recurso específico, ou seja, o método descreve ao servidor o que deve ser feito no recurso apontado na URL fornecida pela requisição. Os métodos mais utilizados nas requisições são GET e POST.

O método GET faz uma requisição de um recurso especifico. Esse método deve ser utilizado somente para recuperar informações e não deve ser utilizado para mais nada. A seguir, um exemplo de mensagem GET. Neste caso, é feita uma

requisição do recurso *index.html* no hospedeiro *www.webSite.com* considerando a versão 1.1 do HTTP.

```
GET /index.html HTTP/1.1
Host: www.WebSite.com
```

POST é o método utilizado quando se deseja enviar dados ao servidor para que o mesmo os processe. Os dados são enviados no corpo da mensagem, sendo necessário utilizar os parâmetros de cabeçalho para especificar o tipo dos dados (*Content-Type*) e a quantidade desses dados (*Content-Length*). O método fornece uma maior segurança, devido ao envio dessas informações no corpo da mensagem e não na URL como é feito com o envio de parâmetros no método GET. A seguir, um exemplo de mensagem de requisição com método POST, onde é feita uma requisição POST com o conteúdo no formato de texto padrão.

```
POST / index.html HTTP/1.0
Accept: text/html
If-modified-since: Sat, 29 Oct 1996 19:42:31 GMT
Content-Type: text/plain
Content-Length: 41

Data=DataDoEnvio&Local=Brazil&Id=36587423
```

2.1.3 Códigos de Resposta

As respostas retornadas no protocolo possuem códigos de estado e uma razão textual para aquele código. Esses códigos definem o estado da requisição feita, podendo definir informações como o sucesso ou falha da requisição. Os códigos são definidos em intervalos da seguinte forma:

- Códigos Informativos 1XX
- Códigos de sucesso 2XX

- Códigos de redirecionamento 3XX
- Códigos de erro do cliente 4XX
- Códigos de erro no servidor 5XXX

2.1.4 Segurança

A forma mais comum de aplicar segurança às comunicações feitas com HTTP é uma manter uma conexão criptografada. Para estabelecer essa conexão é amplamente utilizado o HTTPS, sendo o HTTPS um uso combinado do HTTP com o protocolo TLS.

As conexões do protocolo HTTP não são feitas utilizando criptografia, o que torna os seus dados suscetíveis a ataques, assim é possível que terceiros acessem as informações que não deveriam ser capazes de deterem.

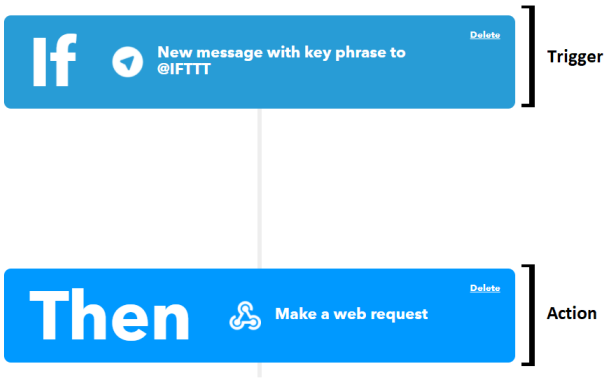
O HTTPS resolve o problema da exposição de dados por utilizar uma conexão TLS que criptografa todo o conteúdo da mensagem, bem como utiliza certificados para garantir a autenticidade, incluído o corpo e os cabeçalhos, tornando-se necessário que, quem envia os dados, encriptar e quem as recebe descriptografar.

2.2 IFTTT

A plataforma IFTTT (*If This Then That*) consiste em um serviço *web* gratuito que permite a integração e comunicação de diversos outros serviços de terceiros (MI et al., 2017). Isso é possível através da implementação de correntes de critérios condicionais, chamados de *applets*, que são disparados quando ocorrem eventos que preenchem os critérios previamente definidos. Os *applets* são constituídos de dois tipos de componentes básicos: os gatilhos e as ações decorrentes desses gatilhos. Dessa forma, pode-se integrar duas aplicações baseando-se em suas mudanças, ou seja, de acordo com os eventos relacionados às aplicações.

No diagrama ilustrado na Figura 2.2, o primeiro elemento em azul representa o gatilho, ou seja, o evento que desencadeará a ação. No segundo em azul, são especificadas as ações que serão executadas uma vez que o evento é reconhecido.

Figura 2.2 – Criação de Applet na plataforma IFTTT



Fonte: Do autor (2021).

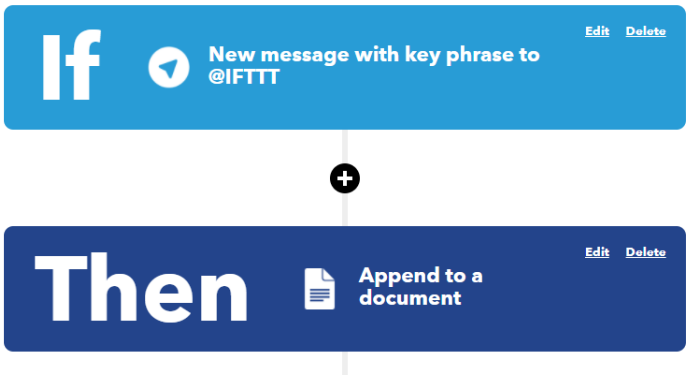
A comunicação pode ser feita através de integrações já nativas do IFTTT, uma lista de elementos já integrados está disponível aqui (INC.,). Pode também ser feito através de uma integração pelo o usuário utilizando um mecanismo denominado *webhooks*. Os Webhooks representam o uso de requisições HTTP para permitir a integração de outros serviços suportados pelo IFTTT e aplicações de terceiros.

Umas das principais características plataforma IFTTT é a possibilidade de prover integração a projetos que utilizem objetos executados em plataformas distintas, uma vez que o serviço trabalha com softwares nativos ou requisições HTTP. Assim, o seu único requisito para o uso é ter acesso à Internet e ser capaz de fazer requisições HTTP.

Um exemplo dessas integrações é ilustrada na Figura 2.3. Neste caso, é feita uma integração na qual, quando acionado o evento do recebimento de uma mensagem de texto via Telegram com um conteúdo especifico, é adicionado um

valor em uma tabela do Google Sheets, sendo essa tabela definida durante a criação do *applet*.

Figura 2.3 – Integração do serviço Telegram com Google Sheets



Fonte: Do autor (2021).

2.3 MQTT

MQTT é um protocolo de comunicação de mensagens que usa TCP como protocolo de transporte e utiliza-se do padrão Publish-Subscribe (YASSEIN et al., 2017). É um protocolo leve, simples e projetado para fácil implementação. Assim, torna-se ideal para o uso de comunicações entre máquinas (*machine to machine* - M2M) e Internet das Coisas(IoT), minimizando o uso de banda e recursos dos dispositivos.

MQTT é um protocolo de transporte para mensagens que permite ao desenvolvedor especificar aspectos relacionados a privacidade, autenticação e autorização. Sendo assim, a segurança de troca de mensagens é especificada de acordo com o contexto e é responsabilidade de quem está utilizando o protocolo implementar sistemas de segurança.

O protocolo MQTT utiliza o TCP como protocolo de transporte, assim provendo uma comunicação ordenada, sem perda e bidirecional. Também, ao utilizar o modelo Publish-Subscribe, o protocolo é capaz de suportar uma forma de

envio de mensagens um-pra-muitos, o que é desejado quando se tem um número de máquinas/sensores que irão comunicar entre si. O sistema se baseia na construção de um broker (servidor) que controla os envios das mensagens e clientes que irão enviar e receber mensagens.

Publish-Subscribe é um padrão de comunicação assíncrona de mensagens, através de eventos. O padrão funciona de forma parecida ao paradigma de fila de mensagens. Porém, uma vez que no publish-subscribe as mensagens são estruturadas por tópicos, quem envia não precisa conhecer quem recebe e vice-versa, basta os dois estarem inscritos no tópico em questão.

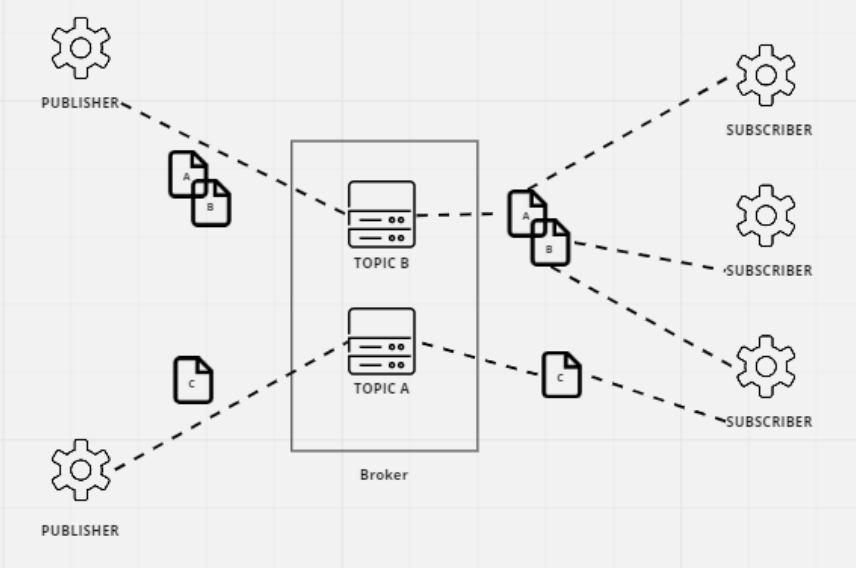
Um dispositivo pode estar inscrito em diversos tópicos, sendo capaz de publicar e receber as mensagens presentes no tópico em questão. Quando uma mensagem é publicada, é levantado um evento que indica, aos dispositivos inscritos no tópico correspondente, que uma nova mensagem foi publicada e está pronta para ser lida.

Dentro do padrão também existe o papel do broker, que é o responsável por filtrar as mensagens e saber pra qual dos tópicos as enviar. Assim, quem publica e quem se inscreve em um tópico não precisa se conhecer diretamente, somente precisam conhecer o broker e o tópico a ser inscrito.

Na Figura 2.4, mostra-se como o broker maneja o recebimento de publicações de diversos tópicos, de forma que o broker propaga as mensagens de um para muitos, sempre enviando as mensagens para todos os clientes inscritos no tópico em que a mensagem foi publicada.

Dessa forma, o padrão Publish-Subscribe fornece um ótimo potencial de escalabilidade, uma vez que, para integrar novos dispositivos a uma rede, basta os mesmos criarem novos tópicos ou subscreverem aos tópicos já estabelecidos como forma de comunicação da rede em questão. As mensagens são encaminhadas somente para os inscritos no tópico correspondente. Além disso, os dispositivos que

Figura 2.4 – Estrutura do protocolo de comunicação MQTT



Fonte: Do autor (2021).

estão *offline* no momento da publicação de uma mensagem irão recebe-la quando estiverem *online*, garantindo assim a entrega.

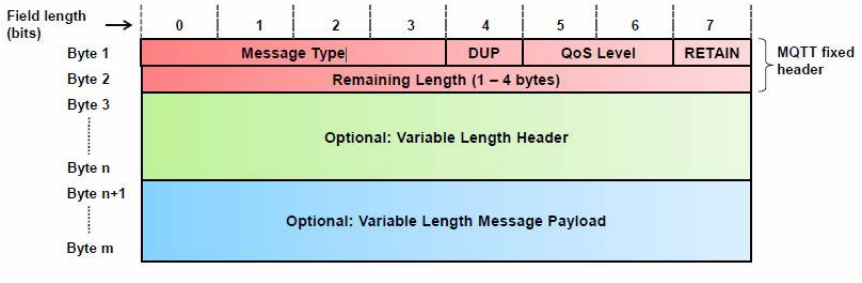
2.3.1 Estrutura da mensagem

Como ilustrado na Figura 2.5, a estrutura da mensagem trocada através do protocolo MQTT é composta de três principais campos, sendo eles: o cabeçalho fixo, que se faz presente em todos pacotes de comunicação do protocolo; o cabeçalho variável, que pode ou não estar presente nos pacotes e por fim, a carga da mensagem que pode estar ou não presente nos pacotes.

Os cabeçalhos do MQTT podem variar entre 2 a 5 *bytes*, contendo dentro de si alguns identificadores do tipo da mensagem, sendo eles: mensagem duplicada, a qualidade de serviço, retenção da mensagem e por fim um identificador que define o tamanho do resto da mensagem. Existe também a parte variável do cabeçalho, onde não existe um padrão e suas informações podem variar. Ao fim

da estrutura da mensagem exite um espaço reservado para a carga da mensagem, o *payload*.

Figura 2.5 – Formato da mensagem MQTT



Fonte: (NERI; LOMBA; BULHOES,).

2.3.1.1 Tipos de mensagem

Existem quinze tipos de mensagem no protocolo MQTT, cada um deles tem um papel diferente na comunicação, todos os tipos de mensagem estão listados na tabela 2.1.

A seguir, são descritos os três principais tipo de mensagem, do ponto de vista de quem desenvolve uma aplicação utilizando o protocolo MQTT:

Connect é um tipo de mensagem enviada pelo cliente ao servidor, que tem como objetivo tentar criar uma conexão com o broker. Assim que envia a mensagem, espera o estabelecimento da conexão.

Publish é um tipo de mensagem enviada pelo cliente ao broker, o cliente informa que o pacote em questão está transportando uma mensagem da aplicação, ou seja publicando algo.

Subscribe é um tipo de mensagem enviada pelo cliente ao servidor, criando uma ou mais inscrições. Uma inscrição faz com que o broker tenha consciência de quais clientes tem interesse em receber menagens publicadas em algum

Tabela 2.1 – Lista completa dos tipos de mensagem do protocolo MQTT

Tipos de mensagem		
Nome	Direção	Descrição
Reserved	Forbidden	Reservado
CONNECT	Cliente-Servidor	Requisição de Conexão
CONNACK	Servidor-Cliente	Reconhecimento de conexão
PUBLISH	Cliente-Servidor Servidor-Cliente	Publica uma mensagem
PUBACK	Cliente-Servidor Servidor-Cliente	Reconhecimento de publicação
PUBREC	Cliente-Servidor Servidor-Cliente	Reconhecimento de publicação
PUBREL	Cliente-Servidor Servidor-Cliente	Reconhecimento de publicação
PUBCOMP	Cliente-Servidor Servidor-Cliente	Reconhecimento de publicação
SUBSCRIBE	Cliente-Servidor	Requisição de inscrição em tópicos
SUBACK	Servidor-Cliente	Reconhecimento de inscrição
UNSUBSCRIBE	Cliente-Servidor	Requisita desinscrição de um tópico
UNSUBACK	Servidor-Cliente	Reconhecimento de desinscrição
PINGREQ	Cliente-Servidor	Requisição PING
PINGRESP	Servidor-Cliente	Resposta de um PING
DISCONNECT	Cliente-Servidor Servidor-Cliente	Requisição de desconexão
AUTH	liente-Servidor Servidor-Cliente	Troca de autenticação

tópico especifico. Assim, ele consegue repassar mensagens publicadas em um tópico a clientes inscritos no mesmo.

2.3.2 Qualidade de Serviço

MQTT prove três níveis de Qualidade de Serviço(QoS) para a entrega de mensagens, e cada conexão com o servidor pode ter um nível de qualidade diferente.

O nível de QoS 0 deixa a garantia da entrega da mensagem na responsabilidade da rede, assim, nenhuma resposta de recebimento é enviada pelo recebedor

e nenhum reenvio é feito por quem está a enviar a mensagem. Logo, a mensagem pode ou não chegar ao destinatário.

Quanto ao QoS 1, o protocolo garante que a mensagem seja devidamente entregue no mínimo uma vez ao destinatário. Após a entrega existe uma comunicação de quem recebeu informando a quem enviou o sucesso da entrega. Caso quem enviou não recebe o *feedback* positivo da entrega, continuará enviando até que tenha a resposta informando o sucesso da entrega. Ao receber a mensagem com sucesso, quem a enviou irá exclui-lá e ela somente fica armazenada no receptor.

Esse nível de qualidade tem uma possível desvantagem. Ao esperar o *feedback*, uma mensagem pode já ter sido enviada diversas vezes e ter sido processada diversas vezes, assim gerando processamento e peso desnecessário na rede.

Por fim, o nível QoS 2 garante a entrega da mensagem, assim como impede a perda e duplicação de mensagens aceitas, sendo o último e mais alto nível de qualidade de serviço.

Essas garantias são possíveis de serem feitas através de dois conjuntos de requisições-respostas. O primeiro conjunto é feito pelo envio da mensagem e a confirmação de recebimento, por fim o último conjunto é composto pelas duas partes confirmarem o recebimento de todas as mensagens trocadas entre si e a finalização da comunicação.

Dentro do cabeçalho da mensagem, há um identificador da mensagem. Porém, o identificador fica novamente disponível (pode ser reutilizado) assim que o processo de recebimento e autenticação da mensagem é terminado.

3 ESPECIFICAÇÃO DO PROBLEMA

A solução apresentada nesse trabalho é uma solução genérica, ou seja, capaz de realizar a automação de qualquer dispositivo residencial, entretanto para podermos realizar uma primeira implementação, bem como o desenvolvimento desse trabalho, foi planejado um caso de uso para se aplicar está estrutura genérica.

O caso de uso no presente relatório apresentado visa a automação de um dispositivo de refrigeração de um fermentador, bem como um dispositivo de irrigação.

Para que a automação do processo de fermentação de cerveja, bem como de um processo de irrigação, seja bem sucedido é preciso uma interface física de controle do meio, sendo o meio em questão, respectivamente, o dispositivo de fermentação e o dispositivo de irrigação.

Dentro do projeto de um dispositivo de fermentação de bebidas, existem diversos elementos a serem controlados, sendo eles: a refrigeração interna; ventilação interna e ventilação externa do dispositivo. Tipicamente, esses elementos necessitam da atuação manual de um agente humano.

Um dos processos que necessita dessa interação é a ação de manutenção da temperatura específica por meio da ativação e desativação do mecanismo de refrigeração. Sempre que a temperatura interna do dispositivo estiver maior que a o valor desejado a refrigeração deve ser ativada. Da mesma forma, quando temperatura alvo for atingida pela refrigeração, o mecanismo deve ser desativado.

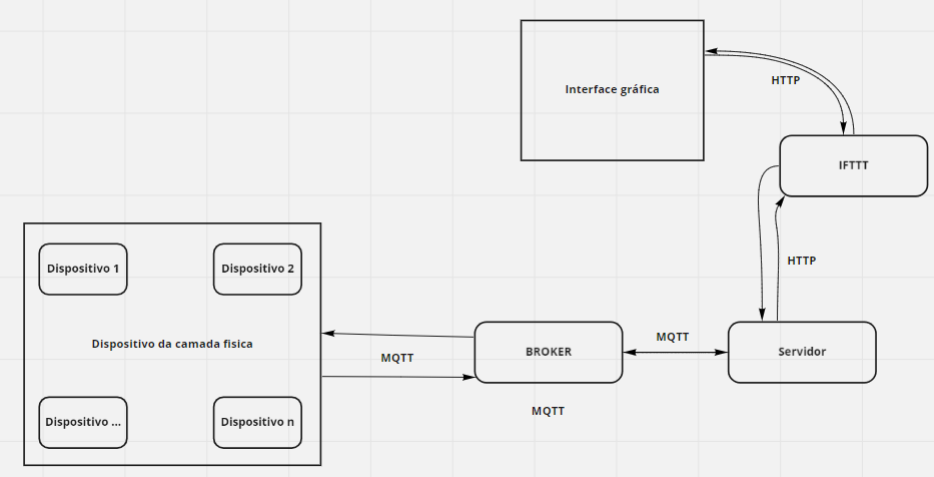
Quando abordamos o mecanismo de irrigação, vemos que este também possui elementos que necessitam de atuação física, ou seja, sua ativação e a desativação. O agente atuante deve acionar o fluxo de água, na medida do necessário, e desativar a irrigação quando atingido o nível desejado.

4 SOLUÇÃO PROPOSTA

Neste capítulo, será explicada a arquitetura e a implementação da solução proposta. Será abordado desde o planejamento da interface física de controle do meio, até a escolha da interface gráfica na qual o usuário irá manejar os detalhes da automação em questão.

Como demonstrado na Figura 4.1, essa arquitetura é construída para atuar na automação de forma genérica, assim não é relevante quais dispositivos são abordados na camada física. Foi utilizado o caso de uso do dispositivo de irrigação bem como o de refrigeração para demonstrar como foi a construção do projeto.

Figura 4.1 – Arquitetura genérica.



Fonte: Do autor (2021).

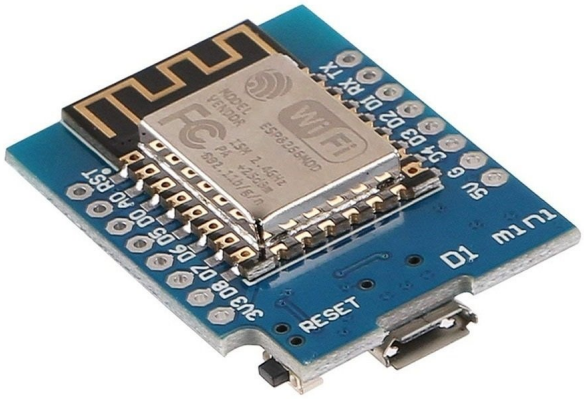
4.1 Camada Física

Para que uma automação seja feita de forma eficaz, o controle sobre o objeto deve ser feito fundamentado em informações retiradas do meio que se deseja controlar. Por exemplo, para automatizar uma lâmpada que acende à medida que o ambiente escureça é preciso retirar informações sobre a luminosidade do meio em que a mesma se encontra. Assim, pode-se controlar quando a lampada deverá

estar acesa ou apagada. O mesmo aplica-se ao caso do dispositivo de fermentação e de irrigação, sendo necessário a retirada de informação da temperatura interna do fermentador, bem como da umidade do solo e luminosidade para que as mesmas possam ser controladas e efetivamente automatizadas.

O microcontrolador ESP8266, ilustrado na Figura 4.2, é uma placa de desenvolvimento que fornece suporte à comunicação via Wi-Fi e TCP/IP. Isso facilita a instalação e uso dessa placa, uma vez que ela pode ser instalada remotamente sem a necessidade de cabos para comunicação e controle remoto. Na programação, pode ser utilizado o SDK do Arduino, assim, tem-se um ambiente de desenvolvimento bem documentado e com grande disponibilidade de recursos e bibliotecas.

Figura 4.2 – Microcontrolador ESP8266



Fonte: (RECICOMP.,)

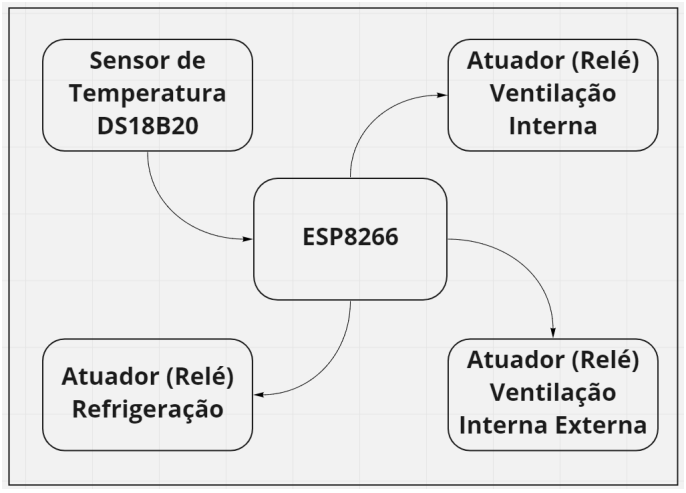
Um dos problemas a ser solucionado era o de que a automação deveria permitir ao usuário controlar e utilizar as funcionalidades dos dispositivos de forma remota. Logo, faz sentido que a manutenção e atualizações também pudessem ser

feitos desta maneira. E isso se fez possível através da capacidade da placa suportar o protocolo FOTA (Firmware Over-The-Air), ou seja, o *firmware* utilizado na placa pode ser alterado ou receber manutenções utilizando a rede Wi-Fi. Assim, torna-se extremamente prática a atualização remota dos equipamentos instalados, sendo necessária a ação humana somente em casos de falhas físicas, como uma falha elétrica de algum sensor.

Neste contexto, foi escolhida a placa ESP8266 para processamento das informações obtidas através dos sensores, bem como, utilizar essas informações para controlar o meio através de módulos de controle como relés, e, por fim, se comunicar com o servidor, dessa forma recebendo informações e instruções de controle do servidor e retornando dados sobre o meio a ser controlado.

Na figura 4.3, é demonstrada a arquitetura do dispositivo de refrigeração. O uso de um sensor se faz indispensável no dispositivo de refrigeração, uma vez que é ele quem informara o microcontrolador da temperatura do ambiente, assim é possível determinar quando a refrigeração será ligada e quando desligada.

Figura 4.3 – Arquitetura do dispositivo de refrigeração

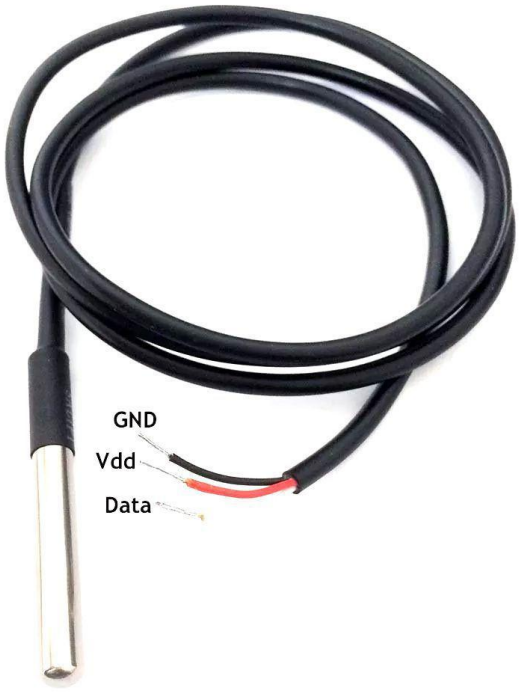


Fonte: Do Autor (2021).

Para cumprir tal papel foi utilizado o sensor digital de temperatura, DS18B20 que é exemplificado pela imagem na Figura 4.4. Através das bibliotecas, Dallas-

Temperature e OneWire, sendo elas para o ambiente do Arduino, é possível requisitar a temperatura que o sensor está medindo no ambiente.

Figura 4.4 – Sensor de temperatura DS18B20



Fonte: (ROBóTICA,)

Também é utilizado como atuador os módulos relés, exemplificado na imagem 4.5, que são dispositivos eletrônicos que fecham ou abrem um circuito elétrico, assim permitindo ou não a passagem de corrente elétrica. Dessa forma, o microcontrolador trabalha com duas instruções para os módulos relé: fechar ou abrir circuito. Assim, é possível ativar ou desativar, respectivamente, todas as ações que são dependentes do atuador, como ventilação, refrigeração e ou irrigação.

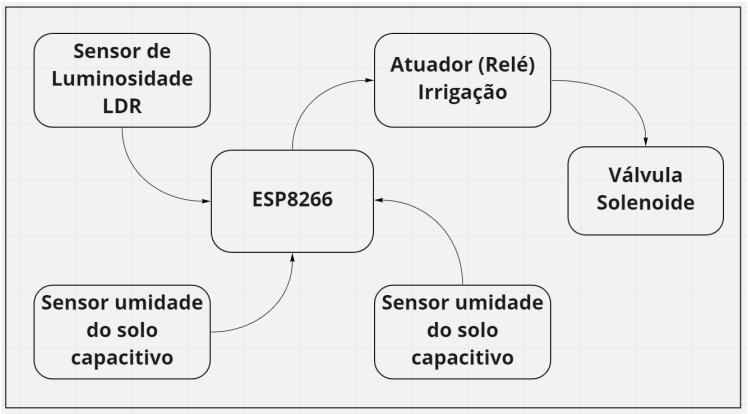
A outra arquitetura abordada nesse trabalho é a do dispositivo de irrigação, exemplificada na imagem da Figura 4.6.

Semelhante ao dispositivo de refrigeração, o dispositivo de irrigação necessita de uma forma de retirar informação do meio físico a fim de controlar o

Figura 4.5 – Modulo Relé

Fonte: (CIA,)

Figura 4.6 – Arquitetura do dispositivo de irrigação



Fonte: Do autor (2021).

ambiente. Dessa forma, foi utilizado um sensor de umidade do solo capacitivo, demonstrado na imagem da Figura 4.7. Esse sensor trabalha com um sinal digital, tendo uma representação binária que indica solo úmido e uma representação que indica solo seco, sendo possível configurar o seu limiar de classificação de seco-úmido manualmente.

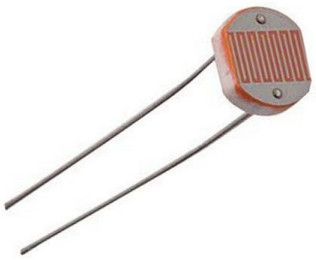
Dessa forma a placa microcontrolador consegue verificar qual sinal está sendo enviando em uma de suas entradas de dados e determinar se o solo precisa ou não ser irrigado.

Figura 4.7 – Sensor de umidade do solo

Fonte: (USINAINFO,)

Para determinar quando a irrigação será ativada, também é interessante que se extraia dados da luminosidade, pois algumas plantas podem sofrer se forem irrigadas enquanto receberem uma forte incidência do sol. Assim foi utilizado um sensor de luminosidade LDR como o da imagem 4.8.

Figura 4.8 – Sensor de luminosidade



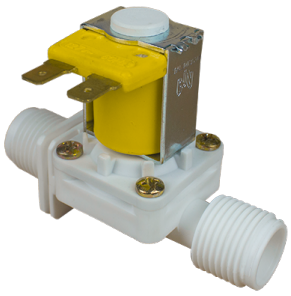
Fonte: (VERDE, a)

O sensor LDR funciona de forma analógica por alterar a resistência sobre uma corrente de acordo com a luminosidade a qual está exposto. Desta maneira, o microcontrolador consegue medir a resistência em uma de suas portas analógicas

e determinar o nível de incidência de luz sobre o sensor, baseando-se tanto na umidade quanto luminosidade para fazer a decisão de quando ativar a irrigação.

E por fim, o atuador final da irrigação, uma válvula solenoide como a da imagem 4.9. Essa válvula tem o funcionamento muito próximo do modulo relé, porém o seu objetivo não é controlar a corrente elétrica, mas sim a corrente de água. Ao receber energia em seus polos a válvula abre o seu circuito e permite que a água corra livremente, quando a corrente elétrica é interrompida a válvula volta a fechar a passagem de água.

Figura 4.9 – Válvula solenoide



Fonte: (VERDE, b)

Dessa forma, com o auxílio de um módulo relé, é possível controlar a passagem de água e, consequentemente, determinar quando a irrigação será feita.

Assim, a camada física se faz completa, pois, foram definidos mecanismos para cada necessidade: retirar as informações do meio com os sensores, processar essas informações com o microcontrolador, e controlar o meio através dos atuadores.

4.2 Camada de Comunicação

A automação proposta neste projeto tem como objetivo prover controle ao usuário de forma remota e a possibilidade de configurar ações automáticas. Dessa forma, é necessário que os dados sejam retirados do agente físico e levados ao usuário, tal como as instruções de controle do usuário devem ser levadas ao agente de controle do meio físico.

Também é comum, dentro de um sistema de automação, que os agentes automatizadores se comuniquem entre si com a finalidade de compartilhar informações. Logo, se faz essencial que uma camada de comunicação seja planejada e implementada para que esses dados possam fluir.

Como dentro de um projeto de automação residencial é muito comum que ao longo do tempo o número de elementos automatizados cresça, é essencial que uma solução proposta tenha uma boa escalabilidade ou forneça fácil integração desses novos itens aos que já são automatizados.

Com essa característica em mente, foi escolhido o protocolo MQTT para a comunicação interna na residência, uma vez que esse protocolo fornece uma fácil escalabilidade para adicionar novos elementos na camada de comunicação. Sendo que novos elementos somente precisam implementar o cliente MQTT e subscrever aos tópicos já criados. O MQTT também fornece outras características chaves, sendo uma delas a comunicação de forma Um-Para-Muitos. Isso facilita a construção da comunicação assim como impede a sobrecarga sobre a banda da rede que sustenta o sistema. Isto é possível porque cada dispositivo se conectará somente com o broker. O protocolo MQTT é uma solução leve e simples, que são características desejadas na grande maioria das aplicações focadas em Internet das Coisas, soluções essas que utilizam dispositivos que, em geral, não tem grande disponibilidade de recursos de computação e transmissão de dados.

Todos os objetos de automação, no caso os dois dispositivos, implementam em sua placa controladora o MQTT e se conectam ao broker como clientes. Cada

dispositivo tem o seu tópico dentro do broker, tópico ao qual irá estar subscrita e escutando possíveis instruções e enviando dados quando for requisitado. Para os microcontroladores, que são programados dentro da plataforma Arduino, foi utilizada a biblioteca PubSubClient[1], que fornece uma implementação de cliente para o protocolo MQTT. A biblioteca é de fácil uso e muito bem documentada, o que foi um fator determinante na hora de selecionar uma boa biblioteca para uso.

O código apresentado na Listagem 4.1 representa a implementação de um cliente MQTT, utilizando a biblioteca PubSubCliente, na plataforma do Arduino. O método apresentado na imagem faz o tratamento dos eventos de recebimento de mensagens em tópicos inscritos, e, mais especificamente na linha 20, é feita a análise (*parse*) e processamento dos dados contidos na mensagem.

```
1  // Callback para as chamadas dos topicos subiscritos
2  void _Mqtt_Callback(char* topic, byte* payload, unsigned int
       length)
3  {
4      // Variavel para conter mensagem
5      String msg;
6
7      // Atribui conteudo da mensagem a variavel
8      for (int i = 0; i < length; i++)
9      {
10         char c = (char)payload[i];
11         msg += c;
12     }
13
14     Serial.print("Conteudo do topico: ");
15
16     // Imprime conteudo da mensagem
17     Serial.println(msg);
```

[1] https://pubsubclient.knolleary.net

```
18
19      // Processa o conteudo da mensagem para poder fazer as
        acoes necessarias
20      _ProcessMsgContent(msg);
21  }
```

Listing 4.1 – Implementação de um cliente MQTT

4.2.1 Sintaxe das Mensagens

Foi definido uma estrutura geral para as mensagens de instrução trocadas dentro dos tópicos do MQTT. Toda instrução é definida de forma a se ter dois campos obrigatórios e um campo opcional.

O primeiro campo define o tipo de instrução, assim o valor utilizado pode ser GET ou SET. O valor GET é utilizado quando se quer requisitar algum tipo de informação, e o valor SET quando se está enviando uma instrução de alteração de valor para algum elemento.

O segundo campo obrigatório define um alvo para a instrução. Dessa forma, o dispositivo (de refrigeração ou irrigação) sabe qual é o alvo da ação definida no primeiro campo. Os valores para esse campo podem variar uma vez que, ao adicionar novas funcionalidades na camada física de um dispositivo, novos alvos para as instruções também serão adicionados.

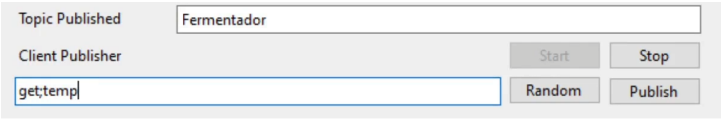
O terceiro campo, opcional, é reservado para algum tipo de valor. As instruções SET definem valores específicos para algum alvo, sendo assim é necessário que o valor seja passado. Este campo é opcional devido ao fato das instruções GET não passarem valor.

Não é utilizado um campo para a identificação do dispositivo destinatário de uma mensagem. Esse campo não é necessário, pois existem tópicos específicos para cada um dos dispositivos. Assim, o destinatário da mensagem sempre irá processar somente as instruções enviadas no tópico correspondente(no qual

ele está inscrito). Por exemplo, na Figura 4.10, é ilustrada uma instrução ao tópico "Fermentador". Dentro desse tópico, o único dispositivo é o dispositivo de fermentação, assim esse dispositivo sabe que todas as instruções no tópico são destinadas a ele.

Ao receber a mensagem, o dispositivo de fermentação executará um *parse* no texto da mensagem. Esse processo identificará o tipo e o alvo da instrução. Assim, após identificar os dados necessários, o dispositivo realiza as ações determinadas na instrução, e, a depender da necessidade, o mesmo retornará algum valor. No exemplo anterior (Figura 4.10), o retorno será um valor numérico em graus Celsius equivalente à temperatura do fermentador.

Figura 4.10 – Instrução de consulta a temperatura



Fonte: Do autor (2021).

Nota-se que a instrução exemplificada na Figura 4.10 é do tipo GET, mas poderia muito bem ser do tipo SET. A seguir um exemplo de instrução do tipo SET:

```
set;setpoint;18.6
```

Neste caso, o dispositivo vai identificar que o tipo SET irá requerer que o mesmo execute uma ação de alteração de valor no campo *setpoint*, e a temperatura alvo a ser definida deverá ser de 18,6ºC.

4.2.2 Envio de Mensagens

Na Listagem 4.2, é apresentado o código que define como a publicação de mensagens em um tópico é realizada através da biblioteca PubSubCliente para a plataforma do Arduino.

```
1  // Faz uma publicacao em um topico
2  bool _Publish(char* topic, char* content)
3  {
4      // Se estiver conectado
5      if(MQTT.connected())
6      {
7          Serial.println("Conectado para fazer publicacao");
8
9          // Publica
10         MQTT.publish(topic, content);
11
12         Serial.print("Conteudo: ");
13         Serial.println(content);
14
15         Serial.print("Publicado no topico: ");
16         Serial.println(topic);
17
18         // Publicacao feita com sucesso
19         return true;
20     }
21
22     // Falha na publicacao
23     return false;
24 }
```

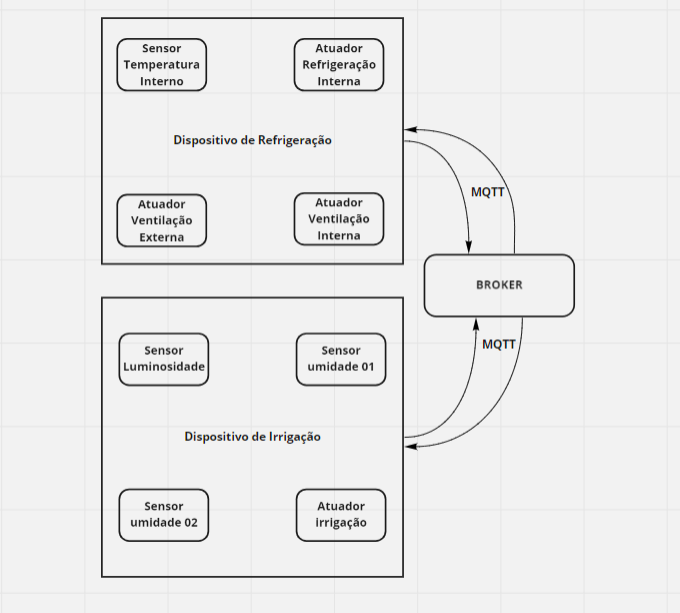
Listing 4.2 – Código de publicação feito em um dos clientes na plataforma Arduino

Nota-se que na Linha 5, é verificado se a conexão com o broker está devidamente estabelecida. Uma vez verificada com sucesso a conexão com o broker, é feita a publicação. Neste exemplo, essa publicação é executada pelo comando *publish* na linha 10.

Caso, no futuro, seja adicionado outro componente para a automação e esse componente precise de informações de outro cliente, pode ser criado um tó-

pico para os agentes da camada física que irão trocar as informações necessárias. Assim, com a adição de um broker que faz a comunicação da camada física a arquitetura do sistema fica definida como apresentado na Figura 4.11.

Figura 4.11 – Arquitetura da camada física e broker



Fonte: Do autor (2021).

Da forma como está apresentada a arquitetura, todos os dados navegam dentro de uma rede local, entre o broker e os seus cliente, mas é interessante que a automação seja feito de forma completamente remota. Dessa maneira, o usuário não se encontra preso a uma restrição de conexão para poder tanto monitorar quanto controlar o sistema. Devido a essa razão, é necessário que um cliente do broker tenha acesso a um contexto de redes externo ao local, um servidor capaz de receber requisições web (HTTP), assim trazendo e levando informações para o usuário.

Desse modo, foi implementado um servidor que utiliza HTTP para se comunicar com a interface do usuário e que também implementa o MQTT com a finalidade de poder se conectar com o broker. Esse servidor também é capaz de

atuar como cliente na camada de comunicação do MQTT. O servidor pode publicar e ler todas as publicações feitas pelos sensores nos tópicos, enviando e recebendo informações para os agentes que controlam o meio físico.

Dentro da camada do MQTT, cada um dos elementos automatizados terão um tópico no qual publicam registros de ocorrências (*logs*), retornos de dados, status da ventilação, umidade do solo, entre outros. Será através desses tópicos que o servidor, ao se conectar como cliente, conseguirá enviar para cada um dos elementos as informações necessárias (como temperatura alvo do fermentador); ou requisitar dados (como pedir que a planta de fermentação informe a temperatura atual). O servidor então deve se fazer presente em todas os tópicos, uma vez que ele será o comunicador da interface com o usuário com a rede de dispositivos que controlam o meio físico.

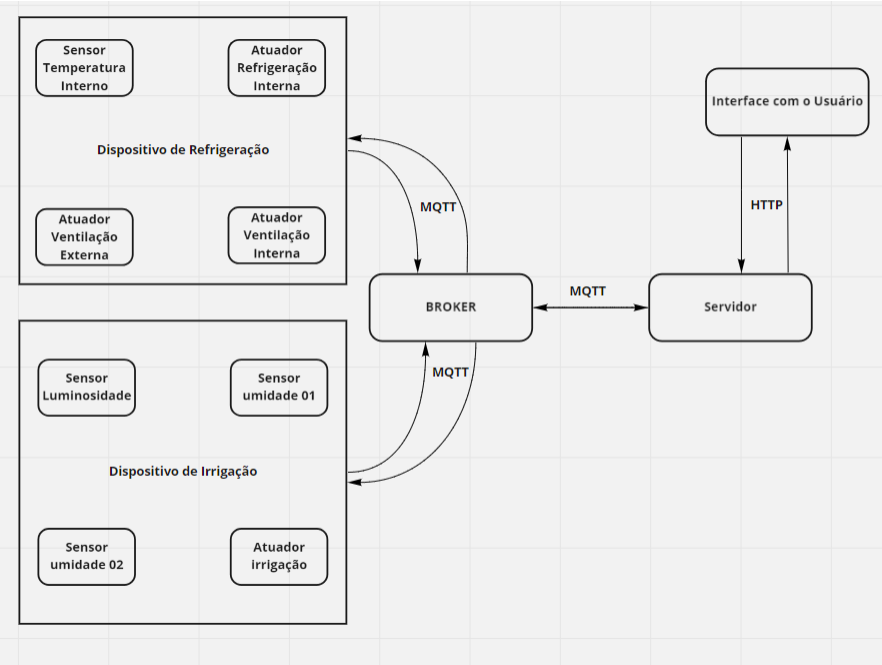
Também foi necessário pensar uma estrutura de mensagens a serem enviadas ao servidor, uma vez que o servidor precisa saber para qual tópico encaminhar as mensagens recebidas da interface com o usuário. Dessa forma, com a camada de comunicação completamente definida, a arquitetura do sistema é estruturada como demonstrado na imagem da Figura 4.12.

4.3 Interface com usuário

Durante o planejamento desse trabalho, foi decidido que seria de grande valor que a interface utilizada para o controle da aplicação fosse multiplataforma. Assim, o usuário estaria o menos impedido possível pelas plataformas que se fazem disponíveis para o mesmo. Também foi determinado que seria interessante uma interface de fácil implementação e uso. O *software* escolhido que compreendia todas essas necessidades é o aplicativo de comunicação por mensagens Telegram.

O aplicativo Telegram tem suporte para as duas principais plataformas *mobile*, Android e IOS, bem como um aplicativo para Windows e uma aplicação

Figura 4.12 – Arquitetura da camada física e de comunicação



Fonte: Do autor (2021).

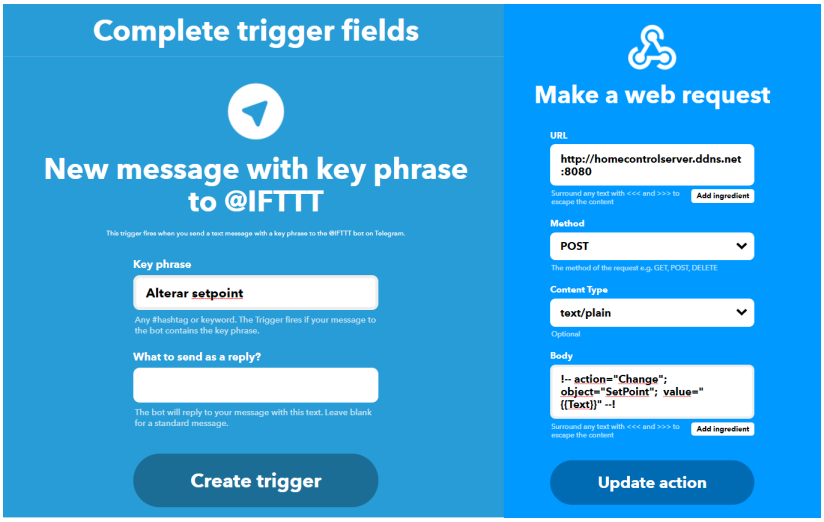
web, proporcionando grande flexibilidade para o usuário. Desta forma, o usuário pode controlar as plantas de refrigeração e irrigação tanto pelo celular quanto por uma das outras plataformas suportadas.

O uso do Telegram só se faz possível pelo fato de o mesmo ter dentro de si a funcionalidade de *bot*. Os *bots* são capazes de conversar com o usuário através de *chats*, como se fossem pessoas reais. Nesses *chats*, os *bots* podem interagir com o usuário enviando informações para o mesmo, mas as funcionalidades variam de acordo com a implementação de cada *bot*.

Um serviço conhecido que fornece *bots* e integração para o Telegram é o IFTTT(If This Then That). O IFTTT fornece um *bot* de conversação, porém esse *bot* é vazio e não faz nada até que seja configurado através da plataforma do IFTTT. Após a configuração, o serviço consegue integrar o *bot* com outros serviços suportados, sendo um deles o webhooks.

A Figura 4.13 representa a configuração dos gatilhos e ações do *bot* fornecido pelo IFTTT. No lado esquerdo da imagem, está a configuração do gatilho, sendo possível adicionar um conteúdo específico que irá desencadear uma ação e uma resposta padrão que o *bot* irá responder sempre que o gatilho for acionado. No lado direito da imagem, estão as configurações de ação, nesse exemplo, uma requisição HTTP, na qual são definidos a URL alvo, o método de requisição, o formato do *body* e o *body* em si.

Figura 4.13 – Configuração de gatilhos e ações de um bot fornecido pelo IFTTT

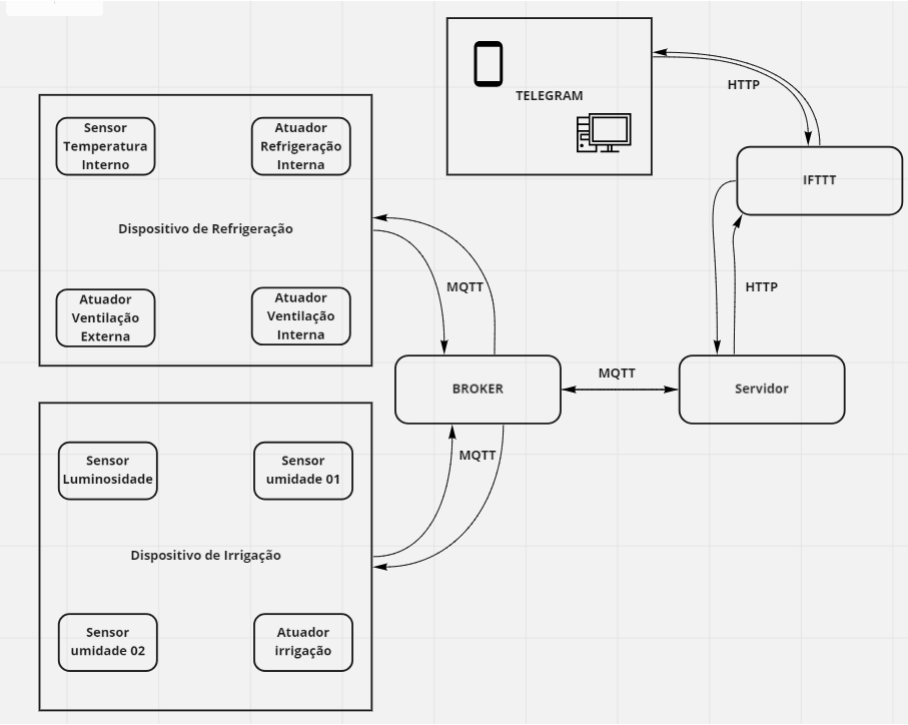


Fonte: Do autor (2021).

O Webhooks é uma forma de se fazer requisições *web* (HTTP) em endereços específicos. Neste caso, as requisições são encaminhadas para o servidor que comunica com o broker, tornando possível, por exemplo, fazer uma requisição de acordo com a mensagem recebida pelo *bot* no aplicativo de conversas.

Como apresentado na Figura 4.14, a arquitetura completa, com o modelo de comunicação, ficou definida por uma rede local com comunicação via MQTT, um servidor que transporta os dados para rede externa através de requisições HTTP que se comunicam com o serviço de Webhooks do IFTTT e, consequentemente, com o usuário.

Figura 4.14 – Arquitetura da comunicação



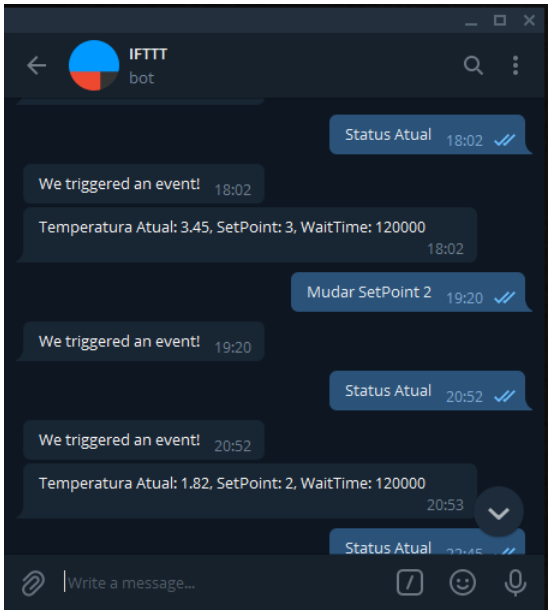
Fonte: Do autor (2021).

Sendo assim, a interface é constituída por uma estrutura de *chatbot*, na qual o usuário envia comandos ao *bot* que os repassa para o controlador no meio físico, sendo possível retornar informações ao usuário. Dessa forma, o usuário interage com uma aplicação cliente conhecidamente testada, mas com as customizações necessárias para que seja possível o controle de um sistema personalizado, implementado por terceiros, ou seja, a comunicação com o servidor através das requisições.

Na Figura 4.15, é demonstrado um exemplo da interface do *chatbot* na qual o usuário consegue controlar os agentes responsáveis pela automação dos dispositivos. Na figura, é possível ver que o usuário requisita o "Status Atual". O servidor identifica o alvo dessa ação por saber que o único sistema que informa *status* é o de refrigeração. Ele repassa a informação ao sistema de refrigeração.

Este por sua vez então identifica que é uma ação de requisição de dados e que o dado a ser requisitado é o status geral do sistema, então é retornado a informação que enfim é apresentada ao usuário.

Figura 4.15 – Interface do Telegram



Fonte: Do autor (2021).

Ainda na Figura 4.15, também é possível ver um exemplo de uma ação que requer a alteração de um dado. Neste caso, o *SetPoint* que representa o valor alvo de temperatura a ser atingido pelo sistema de refrigeração. Ao enviar a mensagem "Mudar SetPoint 2", o usuário consegue reconfigura a temperatura alvo do sistema de refrigeração.

5 CONCLUSÃO

A área da automatização residencial, principalmente utilizando conceitos de Internet das coisas, vem se tornando um ramo relevante da computação. A cada dia, mais aplicações e tecnologias com o foco em Internet das Coisas vêm ganhando espaço, tanto como produto, quanto objeto de estudos acadêmicos. Mas, para que aplicações e tecnologias se mantenham relevantes para o uso em IOT, são necessários alguns requisitos tais como: aplicações leves que demandam pouca computação e banda, fácil escalabilidade e expansão, fácil uso, entre outros.

Visando tais necessidades, foi possível criar uma aplicação que cumprisse as características necessárias para uma solução de automação residencial funcional e eficaz. Implementando assim, uma solução com uma interface de fácil uso, através da estrutura de conversa do Telegram, e uma arquitetura genérica e escalável que consegue comportar futuras integrações.

A viabilidade da solução implementada deve-se ao planejamento da comunicação feita no projeto, assim mantendo-se apto para comunicar com pouco uso de banda, bem como proporcionando fácil manutenção e uso. Além disso, foi utilizada como aplicação cliente para o usuário, uma interface de fácil implementação e uso, comum a grande maioria dos usuários, um aplicativo de mensagens instantâneas.

Apesar das ferramentas escolhidas para a implementação fornecerem diversos pontos positivos, há um problema que se fez presente, sendo ele a necessidade de implementação de medidas de segurança na camada de comunicação. Um exemplo dessa dificuldade seria uma futura integração para um portão residencial, uma vez que a automação do portão proporciona um risco de segurança para os moradores da residência, ou seja, se faz necessário um certo nível de segurança que não existe no protocolo MQTT.

5.1 Trabalhos futuros

Existem alguns pontos que trariam grandes benefícios para a solução apresentada nesse trabalho. São listadas a seguir, algumas das possíveis melhorias que podem gerar novos estudos e expansão do projeto em questão:

- Processamento de linguagem natural na comunicação com o usuário.
- Protocolos de segurança na camada de comunicação.
- Visualização e histórico dos dados monitorados em ferramentas como o Google Data Studio.[1]

[1] https://datastudio.google.com/

REFERÊNCIAS

CIA, A. e. **Modulo Relé**. Disponível em: <https://www.arduinoecia.com.br/ligando-uma-lampada-com-modulo-rele-arduino/>.

INC., I. **IFTTT All Services**. Disponível em: <https://ifttt.com/services>.

KUROSE, J.; ROSS, K. **Redes de computadores e a internet: uma abordagem top-down**. [S.l.]: Pearson, 2013.

MI, X. et al. An empirical characterization of IFTTT: Ecosystem, usage, and performance. In: **Proceedings of the 2017 Internet Measurement Conference (IMC'17)**. New York, NY, USA: ACM, 2017. p. 398–404. ISBN 9781450351188. Disponível em: <https://doi.org/10.1145/3131365.3131369>.

NERI, R.; LOMBA, M.; BULHOES, G. **MQTT**. Disponível em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/>.

RECICOMP. **ESP8266**. Disponível em: <https://www.recicomp.com.br/produtos/placa-wemos-d1-mini-esp8266/>.

ROBóTICA, C. da. **DS18B20**. Disponível em: <https://www.casadarobotica.com/sensores-e-modulos/sensores/temperatura/sensor-de-temperatura-ds18b20-a-prova-d-agua-waterproof>.

SANTOS, B. P. et al. Internet das coisas: da teoria à prática. **Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**, v. 31, 2016. Disponível em: <https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf>.

USINAINFO. **Sensor de umidade do solo**. Disponível em: <https://www.usinainfo.com.br/sensor-de-umidade-arduino/sensor-de-umidade-do-solo-arduino-hd-38-5475.html>.

VERDE, P. **Sensor de luminosidade**. Disponível em: <https://www.autocorerobotica.com.br/ldr-5mm-sensor-de-luminosidade-pacote-x10-pcs>.

VERDE, P. **Válvula Solenoide**. Disponível em: <https://www.paiolverde.com.br/valvula-solenoide-nascimetal-va-04-rosca-34-angulo-180-110v>.

YASSEIN, M. B. et al. Internet of Things: Survey and open issues of MQTT protocol. In: **2017 International Conference on Engineering MIS (ICEMIS)**. IEEE, 2017. p. 1–6. Disponível em: <https://doi.org/10.1109/ICEMIS.2017.8273112>.