FACULDADE DA INDÚSTRIA SÃO JOSÉ DOS PINHAIS
CLÍSTENES GRIZAFIS BENTO
DIEGO MURILO SOUSA DA LUZ
GABRIEL RODRIGUES CARVALHO JUNIOR
LEONARDO PESTILO DOS SANTOS
LUIZ HENRIQUE PEREIRA ISBAES
OZEIAS MATEUS SANTOS THOMAZ
VITOR DE CAMPOS PODANOSKI

ESTUDO EXPLORATÓRIO SOBRE A INTERNET DAS COISAS (IOT) E SUA APLICAÇÃO NA AUTOMAÇÃO RESIDENCIAL: Como utilizar o celular para verificar se a porta de casa está trancada e destrancá-la remotamente

SÃO JOSÉ DOS PINHAIS
2020

CLÍSTENES GRIZAFIS BENTO
DIEGO MURILO SOUSA DA LUZ
GABRIEL RODRIGUES CARVALHO JUNIOR
LEONARDO PESTILO DOS SANTOS
LUIZ HENRIQUE PEREIRA ISBAES
OZEIAS MATEUS SANTOS THOMAZ
VITOR DE CAMPOS PODANOSKI

ESTUDO EXPLORATÓRIO SOBRE A INTERNET DAS COISAS (IOT) E SUA APLICAÇÃO NA AUTOMAÇÃO RESIDENCIAL: Como utilizar o celular para verificar se a porta de casa está trancada e destrancá-la remotamente

Trabalho Integrador apresentado para disciplina de Projeto Integrador, orientado pelo Professor Me. Rafael Pires Machado, do 2º período do curso de Bacharelado em Engenharia de Software, da Faculdade da Indústria São José dos Pinhais.

SÃO JOSÉ DOS PINHAIS
2020

TERMO DE APROVAÇÃO

CLÍSTENES GRIZAFIS BENTO
DIEGO MURILO SOUSA DA LUZ
GABRIEL RODRIGUES CARVALHO JUNIOR
LEONARDO PESTILO DOS SANTOS
LUIZ HENRIQUE PEREIRA ISBAES
OZEIAS MATEUS SANTOS THOMAZ
VITOR DE CAMPOS PODANOSKI

ESTUDO EXPLORATÓRIO SOBRE A INTERNET DAS COISAS (IOT) E SUA APLICAÇÃO NA AUTOMAÇÃO RESIDENCIAL: Como utilizar o celular para verificar se a porta de casa está trancada e destrancá-la remotamente

Este trabalho foi julgado e aprovado como requisito parcial para conclusão do Projeto Integrador, na Faculdade da Indústria São José dos Pinhais.

_____
Profª Me. Cassiana Fagundes da Silva
Bacharel em Engenharia do Software

Orientador(a): _____
Prof Me. Rafael Pires Machado

Banca: _____
Prof.

São José dos Pinhais, ____/_____/_____

RESUMO

Com a popularização da internet e outras inovações tecnológicas, os níveis de produção industrial desencadearam no surgimento da indústria 4.0, em que um dos pilares é a internet das coisas (IoT). Visando abordar um dos principais temas sobre a quarta revolução industrial, o presente trabalho foi elaborado com o uso de pesquisa bibliográfica, documental e de internet. Tendo em vista as diversas aplicações da IoT, esta pesquisa exploratória teve como objetivo desenvolver um sistema de automação residencial que por meio de um aplicativo para dispositivo móvel mostre ao usuário quando a porta está trancada, permitindo destrancá-la remotamente. Este estudo apresenta uma pesquisa bibliográfica sobre automação residencial, programação e prototipagem, um levantamento de requisitos para criação de protótipo, a confecção do protótipo com as informações coletadas e realização de testes de funcionamento. Após a realização das pesquisas documentais, foi realizado o esboço de um protótipo contendo programação e componentes necessários para atingir o objetivo, seguido de uma etapa de prototipagem e testes, onde chegou-se à conclusão de que a metodologia permitiu a interação entre a teoria e prática e que os resultados atingiram níveis satisfatórios, permitindo que o objetivo geral do trabalho fosse devidamente alcançado.

Palavras-chave: Indústria 4.0, Sistemas Embarcados, Domótica, Casa do Futuro, Residência Inteligente.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho é uma pesquisa exploratória realizada por estudantes do segundo período do curso de Engenharia do Software da Faculdade da Indústria, visando aproveitar o uso da tecnologia para criar uma ferramenta de automação residencial útil para o dia a dia, tendo como base os conhecimentos adquiridos por meio de pesquisas bibliográficas, documentais e de internet. Tendo em vista a relevância e atualidade do tema, poderá ser utilizado como guia para futuras pesquisas e parâmetro de avaliação para o uso de sistemas embarcados.

A pesquisa está estruturada em cinco etapas: a) introdução: que visa apresentar o foco, relevância acadêmica e prática e metodologia aplicada no trabalho, b) fundamentação teórica: apresentação de estudos publicados sobre o tema, c) desenvolvimento: relatando as experiências da confecção de um protótipo associado com a fundamentação, d) considerações finais da equipe, e) referências: do material teórico consultado que fundamentaram o trabalho.

1.1 OBJETIVOS

Os objetivos do trabalho estão divididos em geral e específicos.

1.1.1 Objetivo Geral

Desenvolver um sistema de automação residencial que por meio de aplicativo para dispositivo móvel mostre ao usuário quando a porta está trancada e destranque-a remotamente.

1.1.2 Objetivos Específicos

a) Identificar na literatura quais os conceitos sobre automação residencial, programação e prototipagem;
b) Verificar quais as ferramentas e componentes necessários para a criação de um protótipo;
c) Confeccionar protótipo de um sistema de automação residencial;
d) Avaliar funcionamento do protótipo desenvolvido.

1.2 JUSTIFICATIVA

A partir do século XXI, com a popularização da internet e outras inovações tecnológicas, os níveis de produção industrial cresceram em todo o mundo, desencadeando o surgimento da indústria 4.0, com a promessa na criação de melhores serviços e produtos para atender uma demanda cada vez maior e mais exigente, fundamentada em inovação e melhoria contínua (LIMA; PINTO, 2019). Esses avanços estão provocando alterações profundas, não só na indústria, mas também na economia, nos valores, na sociedade, na forma como nos relacionamos, nas plataformas digitais, entre outras (COELHO, 2016).

De acordo com Gonçalves (2019), a capacidade de conectar diversos dispositivos na internet ampliou o âmbito da pesquisa sobre a “Internet das Coisas” (IoT, do inglês *Internet of Things*), o que possibilitou interconectividade entre dispositivos em geral, com intuito de comunicar os equipamentos eletrônicos do dia a dia à internet através do uso de sensores e outros equipamentos, possibilitando o uso de controles remotos ou até mesmo *smartphone.*

Accardi e Dodonov (2012) apresentam a aplicação da IoT em ambientes domésticos, denominando-as de Automação Residencial (AR), sendo conhecidas também como domótica, residência inteligente ou casa do futuro. A automação residencial tem o objetivo de proporcionar conforto, produtividade, praticidade, rentabilidade, eficiência e economia, valorizando a imagem do empreendimento e de seus usuários.

A aplicação da IoT é um assunto em crescimento e pode ser aplicado em diversas áreas como agricultura, indústrias e residências. Porém, tal conhecimento não é tão disseminada fora do ramo da Tecnologia da Informação (GONÇALVES, 2019). Desta forma o presente trabalho busca apresentar um exemplo prático da aplicação da IoT no âmbito residencial.

1.5 METODOLOGIA

O presente trabalho foi desenvolvido por meio de uma pesquisa exploratória que segundo Marconi e Lakatos (2017) consiste em levantamento de dados e suas análises, como levantamentos bibliográficos, estudo de caso e levantamento de

campo e Gil (2017) acrescenta que seu propósito é proporcionar maior familiaridade com o problema, objetivando a torná-lo mais explícito ou facilitar a criação de hipóteses.

A fim de identificar os conceitos desenvolvidos neste trabalho, foi realizada uma pesquisa bibliográfica, que Gil (2017) e Marconi e Lakatos (2017) definem como aquela realizada a partir de materiais já publicados, com base em textos de livros, ensaios críticos, artigos científicos, entre outros.

Documentais que para Marconi e Lakatos (2017) tem como objetivo coleta de dados limitado à documentos, tal que Gil (2017) compara os parâmetros com a pesquisa bibliográfica, porém com fontes diferentes.

E foi realizada a pesquisa na internet, que para Batista (2012) é um local suscetível a aplicação de inúmeras fontes de pesquisa, sendo assim uma maneira de busca por dados e é um dos maiores veículos de informação, devido à quantidade de dados disponíveis (GIL 2017).

Após as pesquisas, foi desenvolvido um protótipo que segundo Ambrose e Harris (2011) oferece a oportunidade de testar uma ideia de diferentes maneiras para verificar se há êxito na prática, e Machado (2013) reforça que esta é a parte essencial do processo de desenvolvimento de um produto, pois permite que a análise de sua forma e funcionalidade seja feita antes da produção definitiva.

Na última etapa deste trabalho, há a discussão dos resultados obtidos por meio das pesquisas e do protótipo.

2 FUNDAMENTAÇÃO TEÓRICA

Com a finalidade de proporcionar melhor compreensão sobre os assuntos abordados, o presente capítulo está dividido em quatro partes: Indústria 4.0, automação residencial, programação e prototipagem.

2.1 INDÚSTRIA 4.0

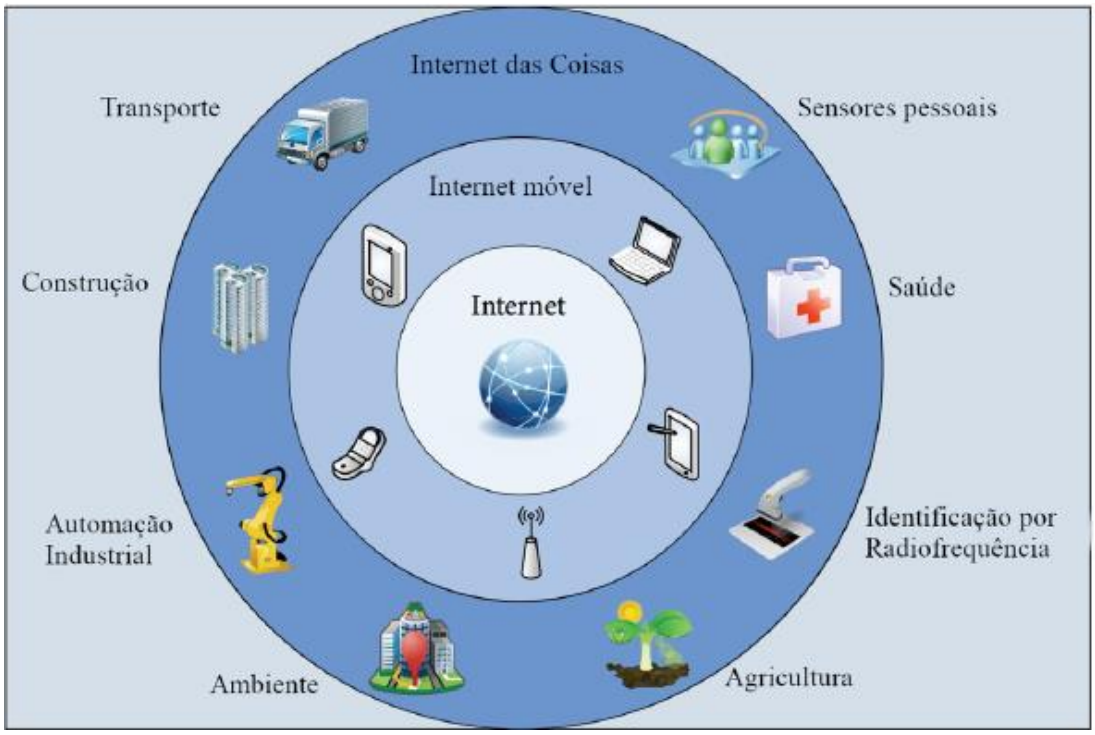
De acordo com Lima e Pinto (2019) o conceito de Indústria 4.0 surgiu em 2011 durante a feira de Hannover Messe sediada na Alemanha. Caracterizada como o início da quarta revolução industrial, que atraiu os interesses científicos, acadêmicos, políticos e empresariais, devido ao fato de ser a primeira revolução industrial a ser observada antes de se concretizar. Coelho (2016) ressalta que na visão da indústria as fábricas serão mais inteligentes, flexíveis, dinâmicas e ágeis.

Para Coelho (2016) a Indústria 4.0 está focada na melhoria contínua em relação à segurança, eficiência, produtividade e retorno de investimentos. Sendo sustentada por seis pilares principais: *Internet of Things* (IoT), computação em nuvem, *big data* e *analytics*, sistemas ciber-físicos, impressão 3D e realidade virtual e aumentada (LIMA; PINTO, 2019).

2.1.1 *Internet of Things* (IoT)

Segundo Coelho (2016) o termo internet das coisas (IoT, do inglês *Internet of Things*) refere-se a objetos físicos e virtuais ligados à internet, que tem sido impulsionado pelo aparecimento e uso generalizado de sensores, assim como o avanço dos dispositivos móveis, comunicação wireless e computação em nuvem. A figura 1 é um exemplo genérico de como a IoT pode ser aplicada.

Figura 1 – Aplicação da IoT em diferentes campos



Fonte: Wang et al. (2015) editado por Gonçalves (2019)

Lima e Pinto (2019) complementam que a IoT é uma das maiores tendências tecnológicas para o mercado, permitindo a conexão via web dos mais diversos dispositivos, como carros (autônomos), ruas, prédios, entre outros, criando assim um conceito de cidade inteligente.

2.1.2 Computação em Nuvem

A computação em nuvem surgiu com o objetivo de facilitar ao acesso a informações de maneira descentralizadas, gerando a possibilidade de decisões estratégicas. Funciona por meio de uma infraestrutura com recursos físicos compostos de computadores, servidores, redes de armazenamento, entre outros, e recursos abstratos como aplicativos, softwares e soluções integradas (LIMA; PINTO, 2019).

Na computação em nuvem os utilizadores podem acessar informações salvas na internet de qualquer lugar que desejarem (LIMA; PINTO, 2019).

2.1.3 *Big data* e *analytcs*

Big data refere-se a grande quantidade de dados armazenados resultantes da existência de milhões de sistemas atualmente ligados à rede, produzindo dados em tempo real sobre quase tudo e disponíveis em todo lado (COELHO, 2019). A medida em que esses dados são armazenados em sistemas seguros, analisados e transformados em informações relevantes, as empresas adquirem novos instrumentos de negócios, de tal forma em que o grande desafio da indústria 4.0 é colecionar todos os dados relevantes e transformá-los em conhecimento (LIMA; PINTO, 2019).

2.1.4 Sistemas ciber-físicos

De acordo com Coelho (2016) os sistemas ciber-físicos, ou *Cyber-Physical Systems* (CPS), são o resultado do processo da evolução tecnológica dos computadores, tecnologia de comunicação, e dos sensores, que evoluíram proporcionando maior capacidade de armazenamento, agilidade e preços acessíveis. Estes sistemas são integrados de computadores, redes de computadores, computadores embutidos e processos físicos interagindo entre si e influenciando-se mutuamente, como exemplo os sistemas de ABS e Airbag dos veículos.

2.1.5 Impressão 3D

A impressão 3D pode ser definida como um tipo de industrialização que se enquadra no cenário da indústria 4.0, sendo capaz de produzir protótipos que aumentam a velocidade do processo de design e produção final. As impressões 3D permitem custo cada vez mais competitivos e reduzidos em quantidades de produção. Tomando como exemplo a indústria automovia, tem-se que os protótipos são produzidos em horas ao invés de semanas como ocorriam em técnicas tradicionais (LIMA; PINTO, 2019).

2.1.6 Realidade virtual e aumentada

Para Lima e Pinto (2019) os processos são simulados e verificados de maneira virtual, onde apenas quando a versão final estiver pronta é que se inicia a produção

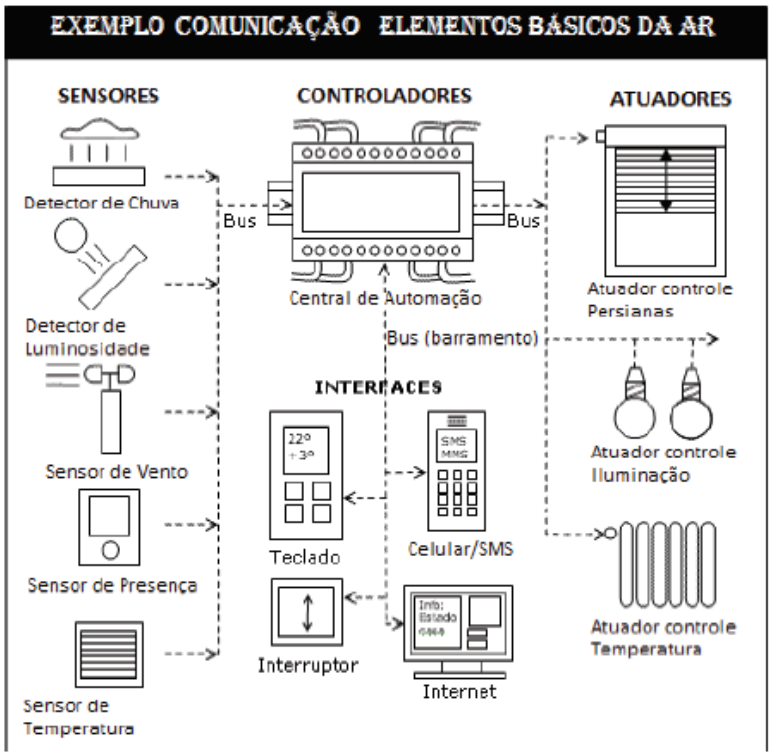
física, transferindo as informações do software de simulação para as máquinas que controlam a produção, com essa tecnologia é possível realizar diversos serviços, como por exemplo, o reparo de máquinas utilizando óculos de realidade aumentada fornecendo instruções de um reparo real.

2.2 AUTOMAÇÃO RESIDENCIAL

Segundo Gonçalves (2019), a IoT tem dominado o cenário residencial com dispositivos interconectados, proporcionando qualidade de vida, serviços e melhoria na segurança em prol das pessoas e da propriedade. A essência da IoT em casas inteligentes é a coleção de aparelhos que interagem entre si através de comando de pessoas autorizadas, onde os aparelhos são inteligentes e possivelmente autônomos.

Accardi e Dodonov (2012) definem a automação residencial como a integração entre tecnologia e serviços, cuja finalidade é tornar a residência automatizada e aumentar a segurança, o conforto e a praticidade, e apresentam como elementos básicos para a automação residencial os controladores, sensores, atuadores, barramentos e interfaces, conforme ilustra a figura 2.

Figura 2 – Exemplo da comunicação dos elementos básicos da AR



Fonte: Casadomo (2010, apud ACCARDI; DODONOV; 2012, P.157)

2.2.1 Controladores

Os controladores são elementos básicos que controlam, monitoram e ativam os dispositivos automatizados (sensores e atuadores), podendo possuir interfaces independentes, como um controle remoto ou serem sofisticadas centrais de automação (ACCARDI; DODONOV, 2012).

Gonçalves (2019) afirma que uma casa inteligente precisa dispor de um centro de controle recebendo informações e devolvendo respostas aos aparelhos, agindo como interface entre os dispositivos da residência e o servidor da internet.

2.2.2 Sensores

Sensores são dispositivos que detectam, monitoram e medem eventos e grandezas físicas, convertendo-as em um valor manipulável por sistemas de computadores (ACCARDI; DODONOV, 2012).

Para Santos et al. (2016), os sensores coletam informações sobre o contexto em que o dispositivo se encontra e em seguida armazenam/encaminham para os centros de armazenamentos, nuvem ou controladores.

2.2.3 Atuadores

Santos et al. (2016) definem atuadores como dispositivos capazes de manipular o ambiente ou reagir de acordo com os dados lidos. Accardi e Dodonov (2012) acrescentam que os atuadores recebem os comandos do sistema de automação, ativando os equipamentos automatizados.

2.2.4 Barramentos

O barramento é um meio físico pelo qual o transporte de informações ocorre, e é o locam onde a interface é conectada (ACCARDI; DODONOV, 2012; CÓRDOVA JUNIOR et al., 2018).

2.2.5 Interfaces

Interfaces são mecanismos ou dispositivos (celulares, navegador de internet, interruptores, controles remotos, painéis, entre outros) que permitem que o usuário visualize e interaja com o sistema de automação (ACCARDI; DODONOV, 2012).

Para Córdova Junior et al. (2018) as interfaces são responsáveis em fazer as traduções da linguagem em que o usuário entende para um formato em que o sistema computacional e seus dispositivos compreendam.

2.3 PROGRAMAÇÃO

Nesse subcapítulo serão abordados os conceitos e definições de programação, bem como recursos necessários para sua aplicação.

Para Souza (2016) a aplicação dos conceitos de IoT necessitam de vários conhecimentos tecnológicos, seja em relação ao uso de sensores eletrônicos ou na parte de computação para programação e transmissão de dados.

Ascenio e Campos (2012) definem programação como o ato de desenvolver um programa de para determinado processamento de dados, no entanto para que o computador entenda e execute o programa, essa programação dever ser escrita em uma linguagem que tanto o computador quanto o programador entendam. Essa linguagem é chamada de linguagem de programação, na qual existem três etapas para o desenvolvimento de um programa, sendo elas:

a) Análise: avalia o problema para definir os dados de entrada, processamento e saída;
b) Algoritmo: descreve uma sequência de passos que deve ser seguida para realizar uma tarefa, pode ser do tipo de fluxograma, descrição narrativa, ou português estruturado;
c) Codificação: onde o algoritmo é transformado em códigos para a linguagem de programação escolhida.

Manzano (2014) acrescenta que para escrever programas, é necessário utilizar no mínimo duas ferramentas: um editor de textos para produzir o código-fonte e outro para transformar o código-fonte em código executável, ou seja, um código em que o

computador compreenda. Quando essas e outras ferramentas são encontradas em um mesmo pacote, este recebe o nome de Ambiente Integrado de Desenvolvimento (IDE, do inglês *Integrated Development Environment*). Existem diversos tipos de IDEs no mercado, para os diversos tipos de linguagens existentes.

2.3 PROTOTIPAGEM RÁPIDA

De acordo com Morais et al. (2018) protótipos são versões de baixo custo de um produto, que podem ser compartilhados e testados dentro da própria equipe ou um pequeno grupo externo, sendo classificado como uma fase experimental, cujo objetivo é proporcionar a melhor solução para o problema ou produto a ser construído.

Machado (2013) ressalta a existência de uma prototipagem rápida, que ao contrário da tradicional e artesanal, está cada vez mais barata, confiável e com baixo tempo de produção. Podendo ser classificadas em tecnologias que permitem a adição e remoção de materiais, sendo aplicada nas mais diversas áreas, como por exemplo automotiva, educação e arquitetura.

O termo “prototipagem rápida” está relacionado ao conjunto de tecnologias utilizadas para criar objetos físicos a partir de fontes de dados gerados por sistemas de computadores, que fornecem informações gráficas visuais de forma em que o cliente possa ver o produto antes da realização do protótipo físico. Verificando assim se o produto atende as características solicitadas pelo cliente (MACHADO, 2013).

3 FERRAMENTAS E COMPONENTES PARA CRIAÇÃO DE PROTÓTIPO

Neste capítulo são apresentadas as ferramentas utilizadas para prototipagem e os componentes necessários para a confecção do dispositivo proposto.

3.1 COMPONENTES

Como apresentado na fundamentação teórica, a automação residencial necessita de diversos dispositivos. Para o desenvolvimento do trabalho foram utilizados dois microcontroladores, um sensor, um atuador, um computador com acesso à internet e um *smartphone*.

3.1.1 Microcontroladores

De acordo com Oliveira (2017), os microcontroladores são capazes de atender uma quantidade sem precedentes de aplicações, necessitando apenas de energia para o funcionamento. Eles possuem um barramento próprio com interfaces de entrada e saída (I/O), memória ram, memória EPROM, interfaces de comunicação e interfaces de rede (OLIVEIRA, 2017).

Os microcontroladores utilizados no presente trabalho foram o Arduino uno Rev3[1] (figura 3) e o módulo ESP-01 do ESP8266.

Figura 3 – Arduino uno Rev3

Fonte: Arduino (2020)

[1] Mais informações sobre Arduino uno Rev3: https://store.arduino.cc/usa/arduino-uno-rev3

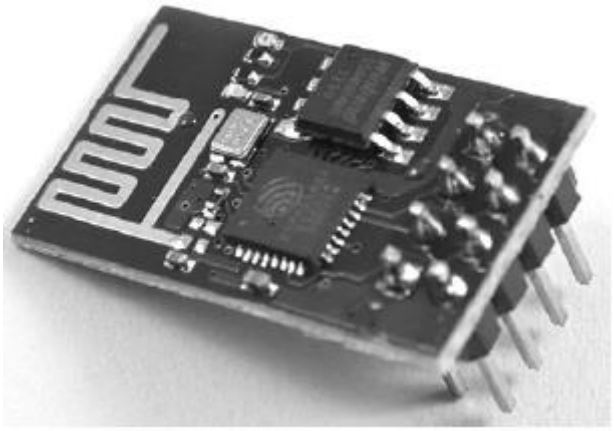
3.1.1.1 Arduino

De acordo com sua página oficial, o Arduino é uma plataforma eletrônica de código acessível baseada em hardware e software simples de utilizar, possuindo diversas versões habilitadas para ler entradas e transformá-las em saída (ARDUINO, 2020).

3.1.1.2 módulo ESP-01

O ESP8266 é um microcontrolador, produzido pela empresa chinesa Espressif, assim como o Arduino, é uma plataforma eletrônica baseada em hardware e software, tendo como diferencial um núcleo de processamento embutido na placa que permite acesso à conexão WiFi (ESPRESSIF, 2020).

Figura 4 – Módulo ESP-01



Fonte: Oliveira (2017)

A figura 4 ilustra o Módulo ESP-01 que para Oliveira (2017) é o módulo mais simples disponível pela Espressif, possui micro controlador ESP8266 e é bastante usado como módulo de comunicação WiFi para o Raspberry e o Arduino, além de permitir-se ser programado para executar algumas aplicações simples como ler um sensor liga/desliga ou acender uma lâmpada.

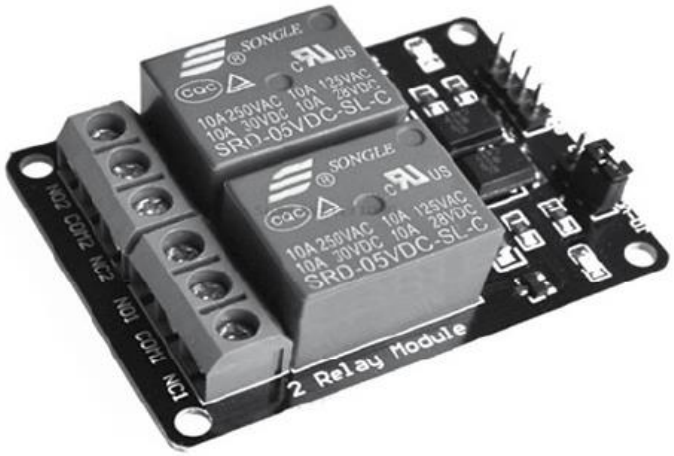
3.1.2 Sensor

O único sensor necessário para o projeto foi de liga/desliga, que transmite uma informação para o ESP-01 quanto está ligado e outro quando está desligado.

3.1.3 Atuador

Para destrancar a porta foi preciso um módulo relé de 5 volts com um canal.

Figura 5 – módulo com dois relés



Fonte: Oliveira (2017)

A figura 5 ilustra um módulo com dois relés, em que no projeto foi necessário utilizar apenas um.

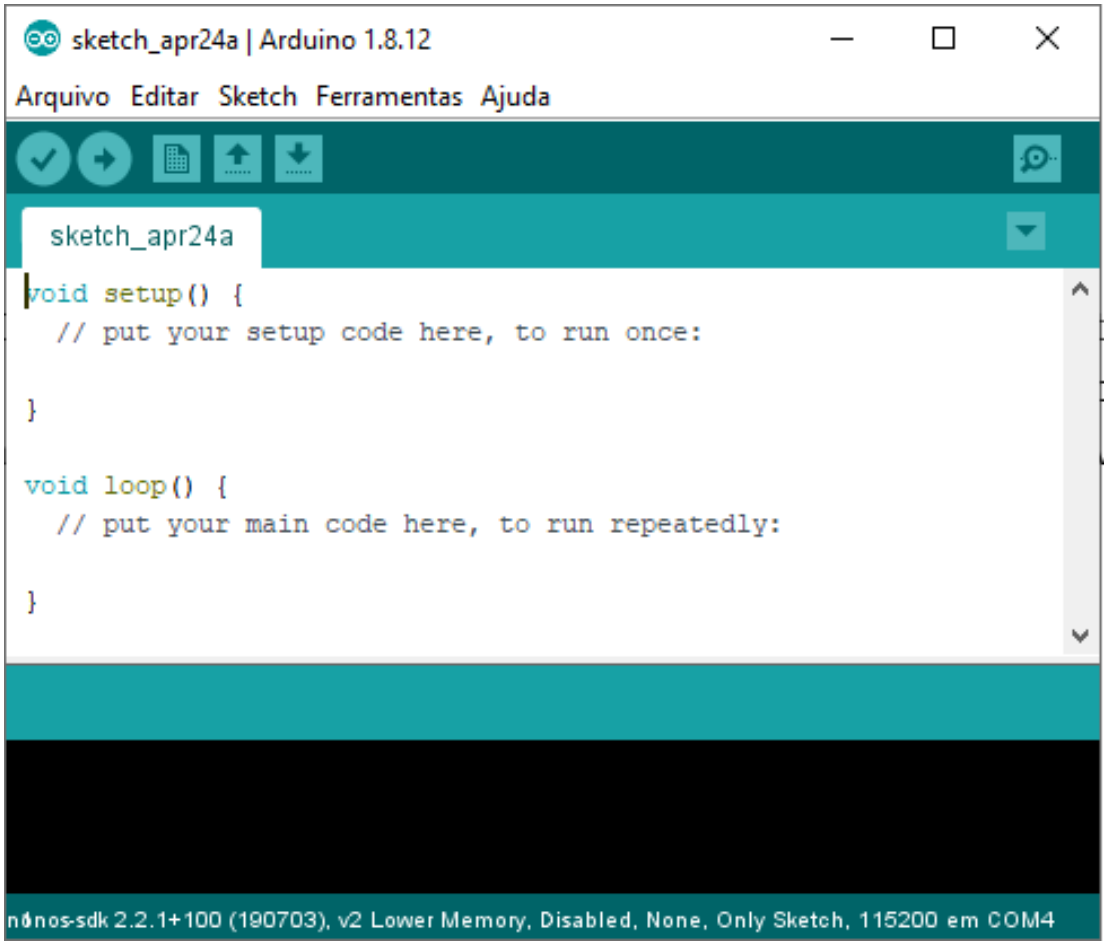
3.2 IDE PARA PROGRAMAÇÃO

Foi usada apenas um IDE para o desenvolvimento do projeto.

3.2.1 IDE Arduino

Para realizar a programação do ESP-01 foi utilizado a IDE do Arduino, a figura 6 mostra a interface do ambiente de desenvolvimento integrado utilizado, que é possível encontrar para *download* no próprio site da empresa (ARDUINO, 2020).

Figura 6 – Interface da IDE do Arduino



Fonte: Os autores (2020)

3.3 FERRAMENTAS PARA PROTOTIPAGEM RÁPIDA

Para realizar a prototipagem rápida foram utilizadas três ferramentas gráficas visuais diferentes: *SkethUp, Tinkercad, Fritzing*.

3.3.1 *SkethUp*

Gaspar (2012) define o *SketchUp* como um software para reprodução e modelagem de objetos em 3D de maneira fácil, rápida e intuitiva. O software possui uma versão gratuita que pode ser instalado em qualquer computador que atenda aos requisitos mínimos de funcionamento (SKETCHUP, 2020).

3.3.2 *Tinkercad*

De acordo com o site oficial do *Tinkercad*, esta é uma plataforma online que une diversas ferramentas de software para auxiliar as pessoas a pensarem e criarem projetos 3D, Circuitos e Blocos de códigos (TINKERCAD, 2020).

3.3.3 *Fritzing*

O *Fritzing* é um software gratuito que promove ao usuário um ecossistema criativo, permitindo a criação e documentação de protótipos sobre processamento e Arduino (FRITZING, 2020).

3.4 FERRAMENTAS PARA PROTOTIPAGEM FÍSICA

Além dos componentes supracitados foram utilizadas placas de circuito conhecidas como *protoboards*, resistores e cabos elétricos.

3.5 APLICATIVO PARA DISPOSITIVO MÓVEL

Para a comunicação entre o usuário do dispositivo móvel e o ESP-01 utilizou-se um aplicativo chamado *MQTT Dash* disponível na Google *Play*, que é uma plataforma de serviços virtuais disponibilizada pela empresa Google LLC (GOOGLE PLAY, 2020).

O Aplicativo *MQTT Dash*, de acordo com a empresa desenvolvedora, Routix Software, foi desenvolvido para o sistema operacional android e tem como objetivo criar painéis, aplicações e automação residencial IoT *Smart Home* habilitados para MQTT e M2M, proporcionando suporte para ESP8266, Arduino, Raspberry Pi, entre outros (GOOGLE PLAY,2020).

4. ESBOÇO DO PROTÓTIPO

O presente capítulo tem como objetivo apresentar ao leitor o processo de criação do protótipo para o funcionamento do dispositivo desenvolvido, com auxílio das ferramentas gráficas visuais *SketchUp, Tinkercad, Pritzing* e o aplicativo *Mqtt Dash*, desde a descrição da localização do dispositivo na porta da casa, perpassando pelas etapas de circuitos e programação, sendo concluído na simulação do uso do aplicativo.

4.1 DESCRIÇÃO DE FUNCIONAMENTO DO PROTÓTIPO

O protótipo utiliza a conexão WiFi da residência, que através do roteador envia informações transmitidas pelo dispositivo para um servidor de internet em nuvem, que reenvia através de comandos as informações para o dispositivo móvel (*Smarphone ou tablet*) e vice-versa, conforme ilustra a figura 7.

Figura 7 – Funcionamento do sistema proposto

Fonte: Os autores (2020)

Para realizar a comunicação entre o dispositivo criado, o servidor em nuvem e o dispositivo móvel, utilizou-se um protocolo de comunicação chamado *Message Queue Telemetry Transport* (MQTT).

Santos et al. (2016) definem o MQTT como um protocolo de camadas de transporte de rede da arquitetura TCP/IP[2] para dispositivos, esse protocolo utiliza uma estratégia chamada de *publish/subscribe* para transmitir mensagens, tendo como objetivo principal minimizar a largura de banda da rede e recursos dos dispositivos.

Oliveira (2017) afirma que o MQTT usa o conceito de *Broker*, que atua como um software servidor recebendo e repassando solicitações, da mesma forma que um servidor Web ou *Servidor de Gerenciamento de Banco de Dados* (SGBD), porém de uma maneira simplificada, repassando apenas dados simples.

4.2 CONFECÇÃO SIMULADA DO PROTÓTIPO

A confecção simulada do protótipo está dividida em duas partes: uma das partes está relacionada ao procedimento necessário para realizar a programação do protótipo e a outra na criação do protótipo para a aplicação.

4.2.1 Circuito para programação do ESP-01

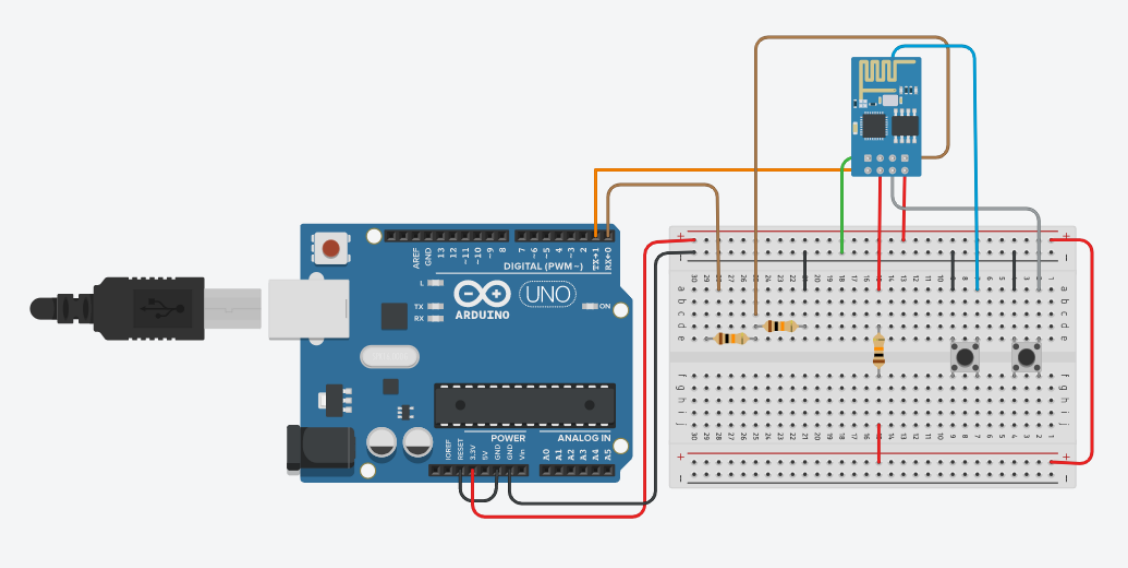
Devido à ausência de um cabo adaptador para transmitir informações do computador ao dispositivo, a fim de tornar possível programar o ESP-01, foi necessário utilizar o microcontrolador Arduino.

Na figura 8 é possível verificar os circuitos utilizados para programar o dispositivo a partir do Arduino. O procedimento é ligar o pinos *reset* e *GND* do Arduino transforma-o em um adaptador de comunicação serial com a entrada USB do computador, feito isso basta ligar as pinos *tx* e *rx* do Arduino com as mesmas portas do ESP-01, usando resistores para converter a voltagem de 5 volts do Arduino para 3.3 volts do ESP-01 e pronto. Em relação aos botões presentes na imagem, o da

[2] Para informações sobre arquitetura de rede TCP/IP e outras acesse: http://www.univasf.edu.br/~leonardo.campos/Arquivos/Disciplinas/Redes_I_2008_2/Redes_I_Aula_02.pdf

direita é o *reset* e serve para resetar a leitura do programa do ESP-01, já o da esquerda é o *boot* servindo para que o dispositivo entre no modo de programação.

Figura 8 – Usando Arduino para programar ESP-01[3]



Fonte: Os autores (2020)

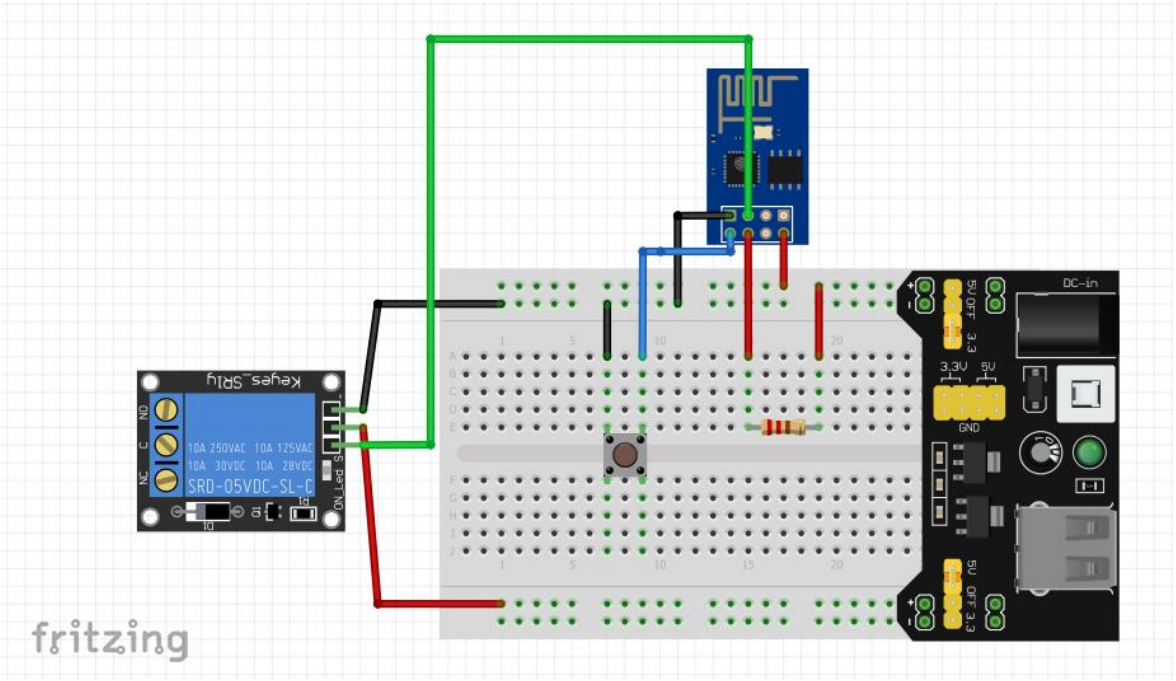
Para registrar os códigos de programação no microcontrolador ESP-01 é necessário que o dispositivo esteja no modo de programação, uma das maneiras de fazer isso é manter o botão *boot* apertado enquanto liga o cabo e energia e, a outra forma, é manter apertado o botão *boot* e em seguira apertar o botão *reset*.

4.2.2 Protótipo para aplicação

Após programado o dispositivo ESP-01 foi necessário adaptar os circuitos para a aplicação, conforme apresenta a figura 9.

[3] Para visitar no projeto no *Tinkercad* acesse: https://www.tinkercad.com/things/8KsXTajYrRB-esp-01-comunicacao-serial-arduino

Figura 9 – Circuito ESP-01 para aplicação



Fonte: Os autores (2020)

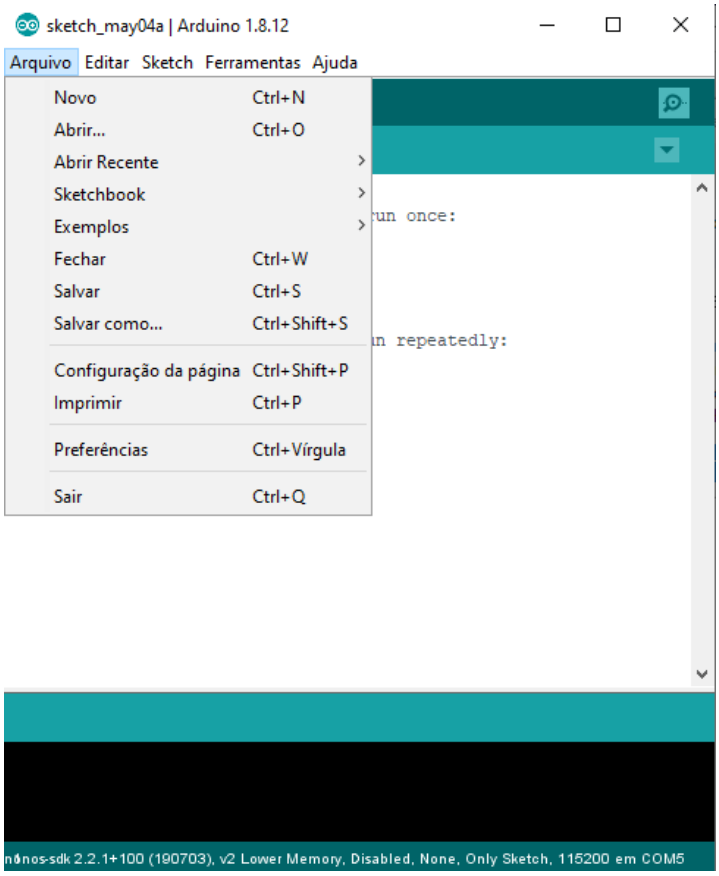
Para o funcionamento do protótipo é necessária uma fonte de alimentação com voltagem de 3.3 volts e 5 volts para ligar o ESP-01 e o relé respectivamente, um botão ligado ao pino 1 (pino tx do ESP-01) do dispositivo, que serve como sensor, identificando como “porta trancada” se for pressionado e, “porta destrancada” caso contrário, e um relé de 5v é ligado ao pino 2 (pino GPIO2 do ESP-01), necessário para destrancar a porta.

4.3 PROGRAMAÇÃO DO DISPOSITIVO MICROCONTROLADOR ESP-01

Para a programação do dispositivo foi preciso seguir os seguintes passos preliminares:

a) Após iniciar a IDE clicar na guia “arquivo” e escolher a opção “preferências”, conforme mostra a figura 10;

Figura 10 – Configuração IDE Arduino etapa a)

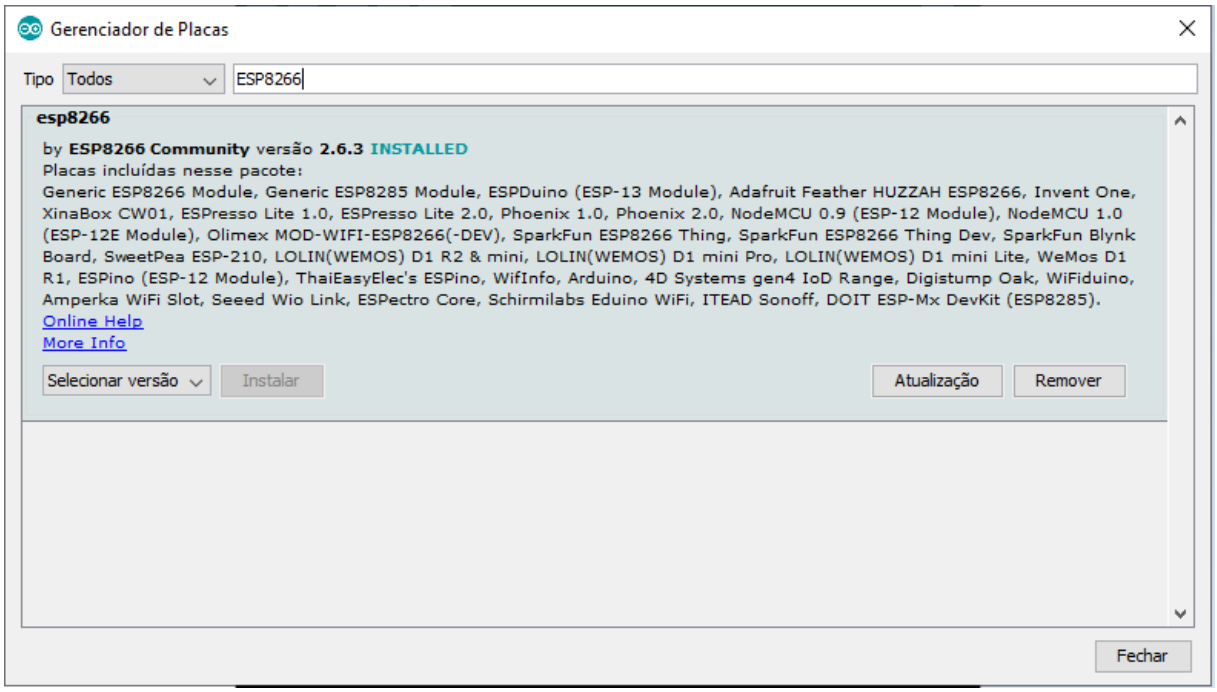


Fonte: Os autores (2020)

b) Na opção “URL Adicionais para Gerenciadores de Placas:” digitar “http://arduino.esp8266.com/stable/package_esp8266com_index.json” e clique em “OK”, isso permitirá que a IDE encontre as informações necessárias para converter os códigos do programa em linguagem de máquina para o ESP-01;
c) Selecione a guia “ferramentas”, clique em “placas” e escolha a opção “Gerenciador de placas...”;
d) Na janela do gerenciador de placas clique na guia de pesquisa e digite “ESP8266” e instale a versão atualizada do pacote, conforme mostra a figura 11;
e) Por último na guia “ferramentas”, clique em “placas” e selecione a opção “Generic ESP8266 Module”.

Feito isso já é possível programar o ESP-01 utilizando a IDE do Arduino.

Figura 11 – Configuração IDE Arduino etapa d)



Fonte: Os autores (2020)

Com a IDE pronta para a programação, inicia-se a etapa de códigos que no presente trabalho é apresentado em quatro partes, a saber: bibliotecas, definições de logins e senhas, declaração de variáveis e funções. O código completo é apresentado no Anexo A, A seguir, estão as explicações de acordo com cada parte programada.

4.3.1 Bibliotecas

As bibliotecas utilizadas foram a “ESP8266WiFi.h” que teve como objetivo utilizar comandos e funções para conectar à internet, e a “PubSubClient.h”, cuja função é enviar e receber informações do servidor MQTT.

4.3.2 Definições de Logins e Senhas

Na segunda parte da programação, foram estipulados os parâmetros de login e senha tanto para conexão ao WiFi como para conectar ao servidor MQTT, conforme mostra a figura 12.

Figura 12 – Definições de Logins e senhas

```
//WiFi
const char* ssid = "digitar o nome da rede que irá conectar"; // Rede WiFi a ser conectada
const char* senha = " digitar a senha da rede que irá conectar "; //senha da rede WiFi a ser conectada
//MQTT
const char* mqttServidor = "digitar endereço do servidor que irá conectar"; //Endereço do servidor a ser conectado
const char* mqttUsuario = "digitar nome do dispositivo"; //Nome do usuário que será conectado no servidor. Normalmente o próprio servidor fornece
const char* mqttSenha = "senha fornecida pelo servidor"; //senha do dispositivo MQTT
const int mqttPorta = 1883; //número da porta que será conectada no servidor, a padrão é 1883
const char* mqttTopicoSub1 = "tópico botao"; //endereço dos pacotes enviados, pode haver mais.
const char* mqttTopicoSub2 = "tópico rele";

WiFiClient cliente; //Cria o nome do cliente que usará o WiFi
PubSubClient client(cliente); //Cria uma Instância "cliente” para ser utilizada pelo MQTT
```

Fonte: Os autores (2020)

Os parâmetros utilizados são comandos estabelecidos pelas bibliotecas usadas, nas linhas de códigos supracitadas ao lado de cada informação tem observações separadas por duas barras (//), que servem como comentário explicado o significado do que está sendo programado.

4.3.3 Declarações de variáveis

Para atingir o objetivo do projeto foi necessário utilizar apenas uma variável chamada “botao" e outra chamada “rele”, definidas respectivamente nos pinos 2 e 1 do ESP-01.

4.3.4 Funções

Foram utilizadas oito funções na programação do dispositivo conforme descrito abaixo:

a) conectar_WiFi: função criada para seguir os procedimentos da biblioteca “ESP8266.WiFi.h”, definindo o modo de conexão, realizando login, senha e verificando o estado da conexão, conforme figura 13;

Figura 13 – Função conectar_WiFi

```
void conectar_WiFi()
{
  WiFi.mode(WIFI_STA); //Definindo o modo de conexão como Station Mode
  WiFi.begin(ssid, senha); //Chamada da conexão WiFi informando os parâmetros
  Serial.println(""); //imprimindo um espaço em branco na linha de baixo do debug
  //Comando para ficar tentando conectar na rede
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500); // cria uma pausa no programa para a comunicação
    Serial.println("Conectando no WiFi"); //O Serial.print e o Serial println servem para escrever mensagem no debug
  }
  //depois de conectado
  Serial.print("Conectado na rede: ");
  Serial.println(ssid);
  Serial.print("Com o endereço de IP: ");
  Serial.println(WiFi.localIP()); // a função WiFi.localIP() estabelece o ip local do dispositivo
}
```

Fonte: Os autores (2020)

b) conectar_MQTT: função criada para seguir os procedimentos da biblioteca “PubSubClient.h”, definindo o modo de conexão, realizando login, senha e verificando o estado da conexão, conforme figura 14;

Figura 14 – Função conectar_MQTT

```
void conectar_MQTT()
{
  client.setServer(mqttServidor, mqttPorta); //Inicia a conexão com servidor enviando os parâmetros
  client.setCallback(recebe_Valores); //Define o que acontece quando o client recebe um pacote de informações
  Serial.println(" ");

  //Comando para ficar tentando conectar ao Servidor MQTT
  while(!client.connected())
  {
    Serial.println("Conectando ao Servidor MQTT");

    if(client.connect("NOME QUE APARECERÁ NO SERVIDOR", mqttUsuario, mqttSenha))//após conectar ao servidor, é necessário sincronizar os dispositivos físicos com o virtual criado pelo servidor.
    {
      Serial.println("Conexão com servidor MQTT confirmada!");
    }
    else
    {
      Serial.print("Falha na conexão com o servidor MQTT, estado: ");
      Serial.println(client.state()); //imprime o estado de conexão
      delay(2000); //espera 2 segundos antes de dar sequência
    }
  }
  //depois de conectado
  //subescreve nos tópicos
  client.subscribe(mqttTopicoSub1);
 client.subscribe(mqttTopicoSub2);

}
```

Fonte: Os autores (2020)

c) reconectar_MQTT: função utilizada para reconectar ao servidor MQTT, caso necessário, descrita por extenso na figura 15;

Figura 15 – Função reconectar_MQTT

```
void reconectar_MQTT()
{
  if(!client.connected())
  {
    Serial.println("Conexão com o Servidor MQTT perdida, reconectando");
    conectar_MQTT();
  }
}
```

Fonte: Os autores (2020)

d) reconectar_WiFi: função utilizada para reconectar à rede WiFi, caso necessário, descrita por extenso na figura 16;

Figura 16 – Função reconectar_WiFi

```
void reconectar_WiFi()
{
  if(WiFi.status() != WL_CONNECTED)
  {
    conectar_WiFi();
  }
}
```

Fonte: Os autores (2020)

envia_Valores: função criada para enviar pacote de informações para o servidor MQTT de acordo com a biblioteca “PubSubClient.h”. As linhas descritas nesta função são para enviar ao servidor uma informação para o caso de o botão estar pressiono e outra caso contrário, conforme aponta figura 17. O efeito *boucing* apontado no código é um problema mecânico que pode acontecer nos botões que acabam enviando informações equivocadas ao programa. Para resolver esse problema, faz-se o programa esperar 30 milissegundos antes de enviar ao servidor as informações sobre o botão.

Figura 17 – Função envia_Valores

```
void envia_Valores()
{
   static bool estadoBotao = HIGH;
  static bool estadoBotaoAnt = HIGH;
  static unsigned long debounceBotao;

  estadoBotao = digitalRead(botao);
  if (  (millis() - debounceBotao) > 30 ) {  //Elimina efeito Bouncing
                                    //OBS: millis() é o comando que calcula os
                                    //milissegundos que passaram desde que o programa
iniciou
     if (!estadoBotao && estadoBotaoAnt) {

        //Botao Apertado
        client.publish(mqttTopicoSub1, "TRANCADA");

        Serial.println("Botao APERTADO. Payload enviado.");

        debounceBotao = millis();
     } else if (estadoBotao && !estadoBotaoAnt) {

        //Botao Solto
        client.publish(mqttTopicoSub1, "DESTRANCADA");
        Serial.println("Botao SOLTO. Payload enviado.");

        debounceBotao = millis();
     }

  }
  estadoBotaoAnt = estadoBotao;
}
```

Fonte: Os autores (2020)

e) recebe_Valores: função criada para receber pacote de informações enviado do servidor MQTT de acordo com a biblioteca “PubSubClient.h”, conforme mostram os códigos na figura 18.

Figura 18 – Função recebe_Valores

```
void recebe_Valores(char* topic, byte* payload, unsigned int length)
{
   String msg;

   //obtem a string do payload (pacote) recebido
   for(int i = 0; i < length; i++)
   {
     char c = (char)payload[i];
     msg += c;
   }

   //Escreve a mensagem recebida pelo servidor no debug
   Serial.print("Mensagem chegou do tópico: ");
   Serial.println(topic);
   Serial.print("Mensagem: ");
   Serial.println(msg);
   Serial.println();
   Serial.println("_____________________");

   if (msg == "0") {

   digitalWrite(rele, HIGH); //se a mensagem recebida for “0” mantem relé desativado

   }

   if (msg == "1") {

   digitalWrite(rele, LOW); //se a mensagem recebida for “1” ativa relé

   delay(2000);   //aguarda 2 segundos

   client.publish(mqttTopicoSub2, "0");  //desativa o botão apertado

   }
}
```

Fonte: Os autores (2020)

f) setup: função que define as configurações iniciais do programa, ou seja, diz os parâmetros que o microcontrolador deve seguir ao ser ligado, a figura 19 mostra como foi escrita a função setup;

Figura 19 – Função setup

```
void setup()
{
  Serial.begin(115200); //Inicia a comunicação serial com a velocidade
                        //de 115200 para comunicação serial
                        // que serve para acompanhar o funcionamento do programa
  delay(10);

  pinMode(botao,INPUT_PULLUP);
  pinMode(rele, OUTPUT);

  conectar_WiFi(); //chama a função conectar_WiFi
                   //o mesmo ocorrerá com as outras funções auxiliares criadas
  conectar_MQTT();

}
```

Fonte: Os autores (2020)

g) loop: Função que define as configurações de comandos que ficarão repetindo no programa, a figura 20 mostra as linhas de código da função loop;

Figura 20 – Função loop

```
void loop() {

  reconectar_MQTT();
  reconectar_WiFi();
  envia_Valores();

  client.loop();//comando para manter o sincronismo
                // entre o dispositivo e o servidor
                //fundamental para o funcionamento correto
                // das funções envia_Valores e recebe_Valores

}
```

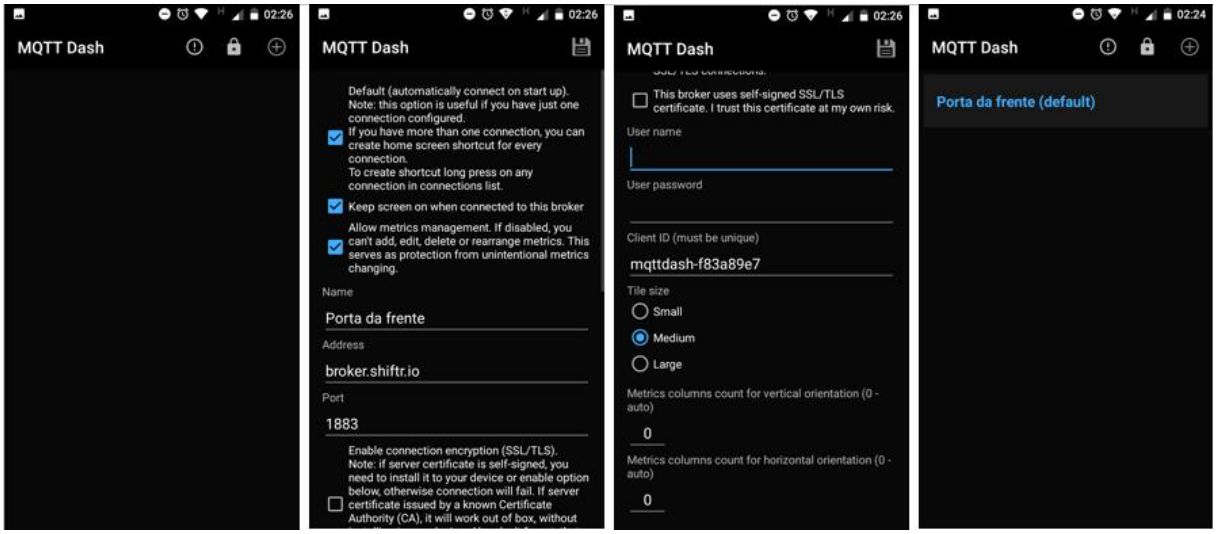
Fonte: Os autores (2020)

As funções setup e loop são funções padrões da IDE do Arduino, as outras foram desenvolvidas para atenderem as necessidades do projeto.

4.4 SIMULAÇÃO DE USO DO APLICATIVO

Para o uso do aplicativo *MQTT Dash* (citado no início do capítulo) é necessário instala-lo ao dispositivo móvel e em seguida inicia-lo. Após iniciar o aplicativo aparecerá uma tela escrito no cabeçalho “MQTT Dash”, um ponto de exclamação, um cadeado e um símbolo de mais (+), a figura 21 a mostra passo-a-passo para cadastrar um cliente ao servidor MQTT.

Figura 21 – Tela inicial *MQTT Dash*

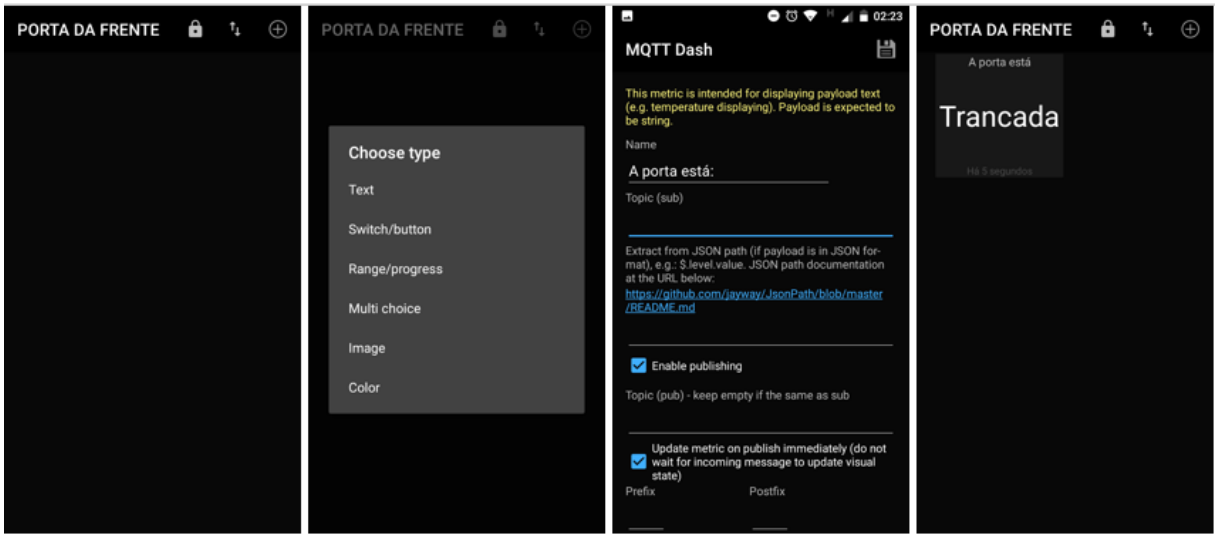


Fonte: Os autores (2020)

Para cadastrar um cliente (ambiente ou dispositivo) é necessário clicar no ícone do “mais” (+), cadastrar um nome, colocar o endereço do servidor, a porta de comunicação, o nome e a senha fornecidas pelo servidor MQTT, feito isso selecionar a opção “salvar”, representada por um disquete no canto superior direito. Ao concluir aparece o item cadastrado na tela inicial do aplicativo, basta clicar nele para dar continuidade.

Após clicar no item o aplicativo irá revelar uma nova interface, semelhante a anterior, esse novo espaço é destinado à criação de ações a serem realizadas ao novo cliente cadastrado. A figura 22 ilustra o procedimento de cadastro para recepção de mensagem do referente ao sensor do protótipo.

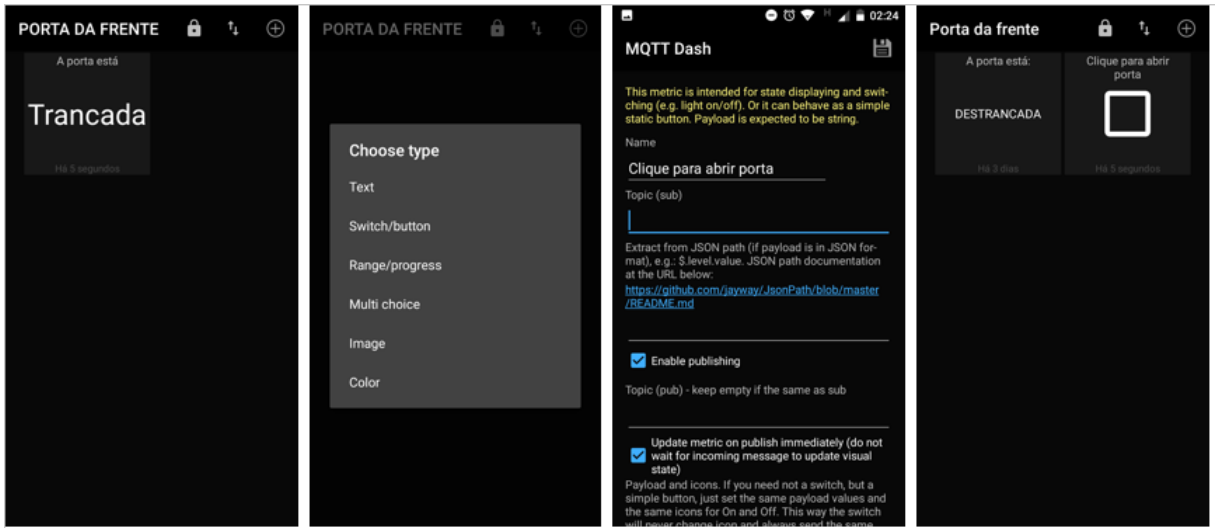
Figura 22 – Cadastro de ação para recebimento de mensagens



Fonte: Os autores (2020)

Semelhante ao cadastro do cliente, para cadastrar uma ação basta clicar no símbolo de “mais” (+), selecionar o tipo de ação que deseja, escrever o nome da ação, informar o tópico que receberá essa ação no servidor e salvar. A opção de ação escolhida para o sensor (botão) foi a “text”, pois assim o aplicativo mostra na tela o texto recebido pelo protótipo, já para o relé escolheu-se a opção “Switch/button” que envia informação de um botão presente na interface do aplicativo ao servidor, que por sua vez envia ao protótipo, a figura 23 ilustra os procedimentos de cadastro da ação para o relé.

Figura 23 – Cadastro *Switch/button* para enviar informações



Fonte: Os autores (2020)

Após realizados os cadastros, ao iniciar o aplicativo aparecerá diretamente ao usuário a última tela da figura 23, onde será possível monitorar se a porta está trancada e destrancá-la, caso necessário. Como é perceptível, o cadastro das ações realizadas no dispositivo através do aplicativo *MQTT Dash* é intuitivo e semelhante para cada tipo de ação, diferenciando apenas nas configurações de cores e layout.

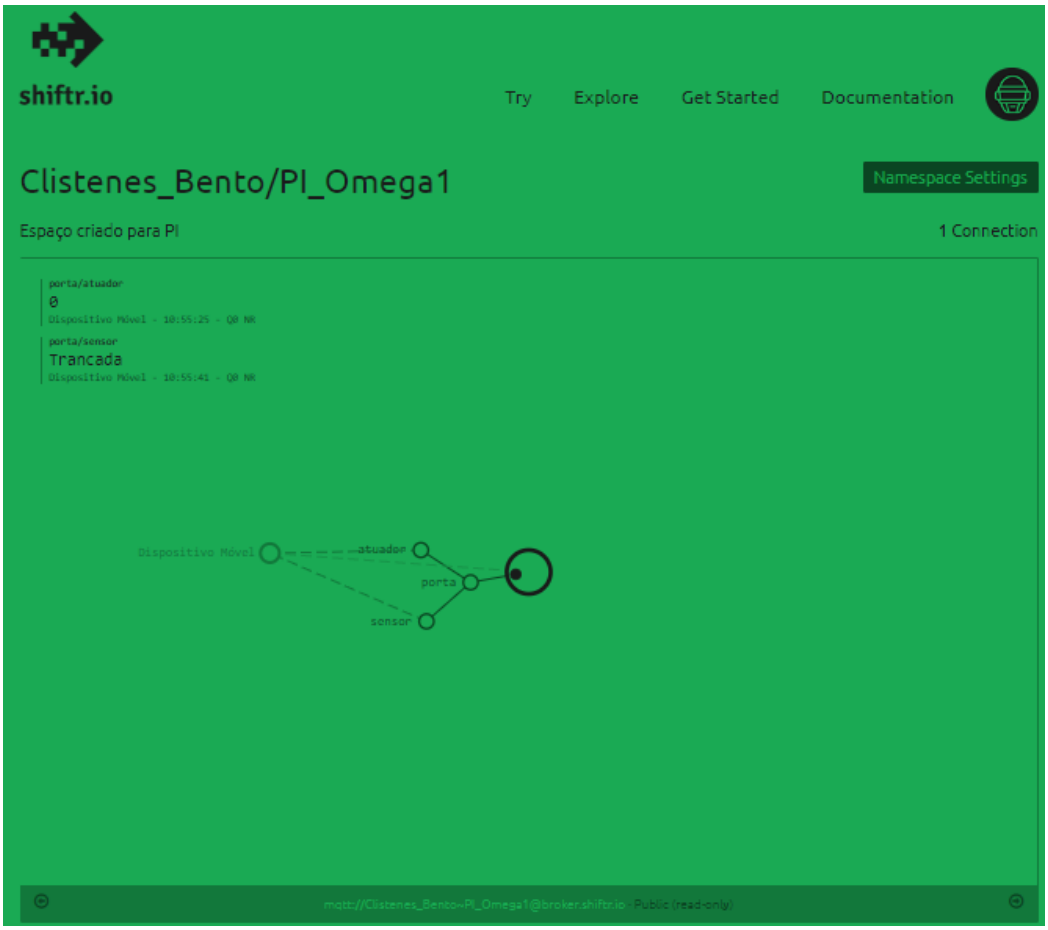
5 PROTOTIPAGEM E TESTES

Neste capítulo é apresentada a aplicação das informações presentes no capítulo anterior. O capítulo está dividido em servidor MQTT, confecção do protótipo e aplicação do protótipo. Todas as etapas do presente capítulo acompanham testes de funcionamento.

5.1 SERVIDOR MQTT

Para a comunicação entre o módulo ESP-01 e o dispositivo móvel, utilizou-se o broker da *shiftr.io*, que oferece benefícios como ser gratuito, fácil de configurar e possibilita através de interfaces gráficas visualizar o estado de cada elemento do sistema, além de identificar qual módulo está acionado e a situação identificada pelo sensor.

Figura 24 – Teste de servidor broker



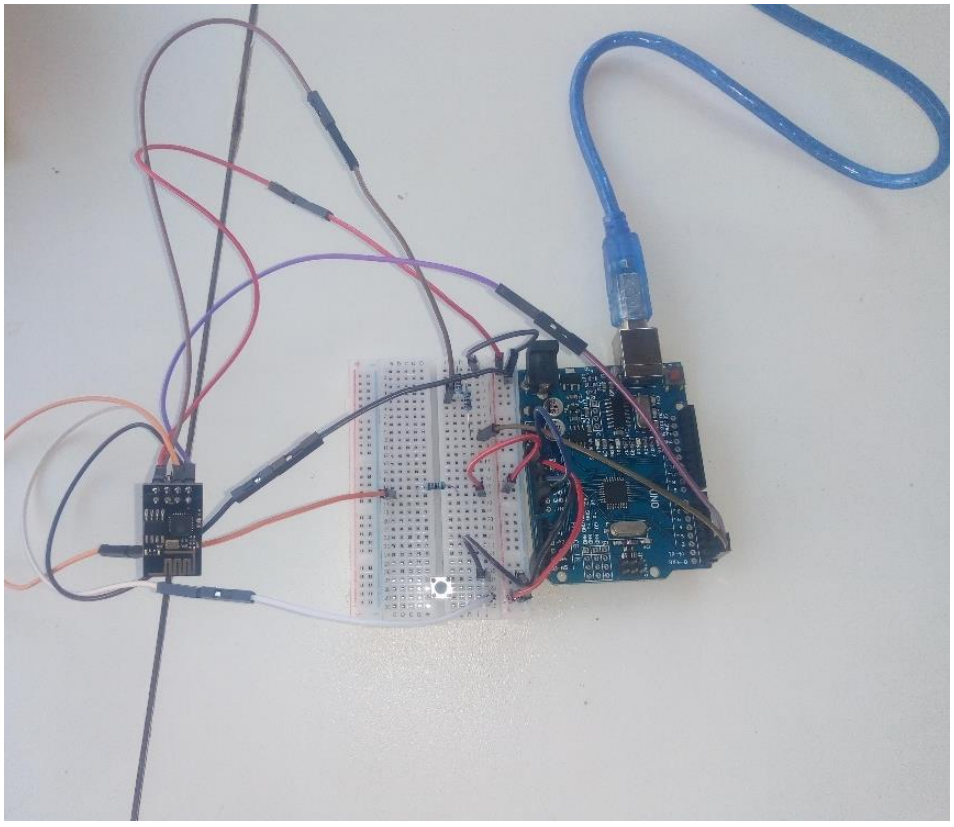
Fonte: Os autores (2020)

A fim de realizar o teste de uso do servidor broker selecionado, cadastrou-se um dispositivo móvel por meio do aplicativo *MQTT Dash*, no qual foram realizadas algumas ações registradas pelo servidor, como demonstra a figura 24, em que aparece a interface de uso em tempo real dos dispositivos utilizados pelo cliente.

5.2 CONFECÇÃO DO PROTÓTIPO

Para confeccionar o protótipo, primeiramente foi construído o circuito para programação, conforme apresentado na figura 25.

Figura 25 – Circuito para programação



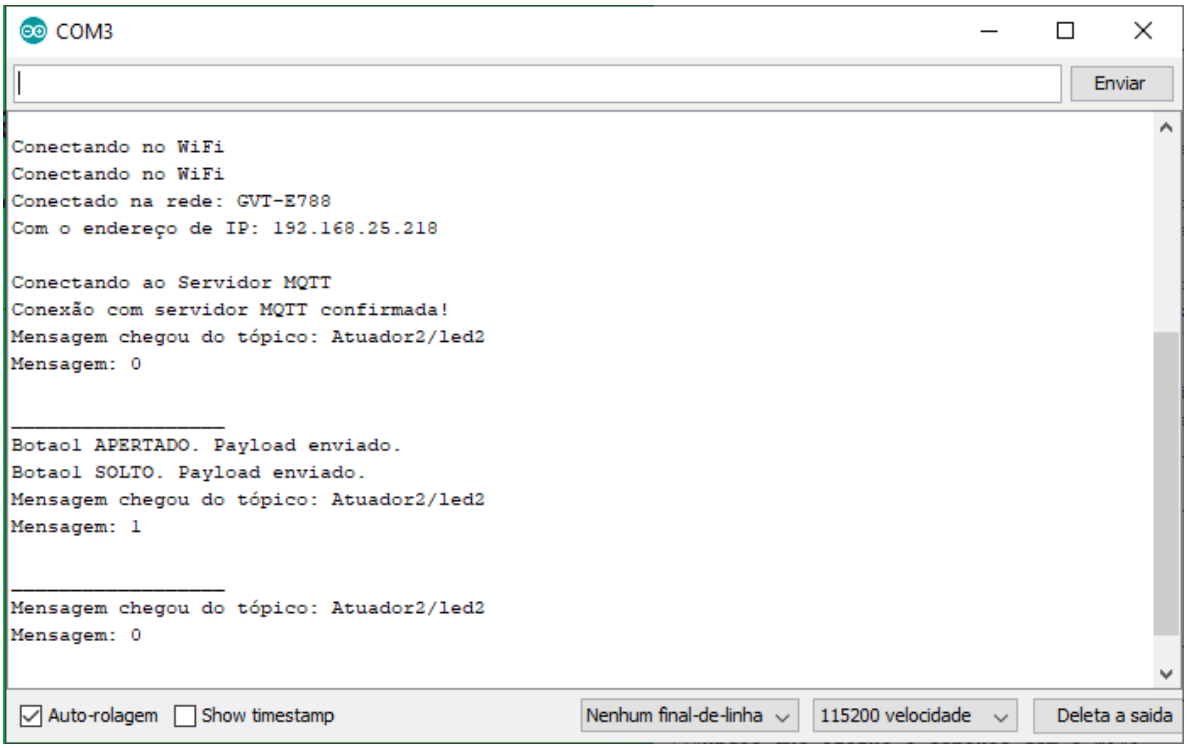
Fonte: Os autores (2020)

Nota-se que diferente da figura 8, foi utilizado apenas um botão, que serve para entrar no “modo programação” do módulo ESP-01.

Com o circuito pronto iniciou-se a etapa de programação, onde foram utilizadas as linhas de código fonte presentes no Anexo A. Após realizada a programação realizou-se os testes enviando informações para o aplicativo *MQQ Dash* por

intermédio do servidor e vice-versa. Os resultados obtidos foram verificados através da tela de comunicação serial da IDE Arduino e a interface gráfica.

Figura 26 – Tela de comunicação seria IDE Arduino



Fonte: Os autores (2020)

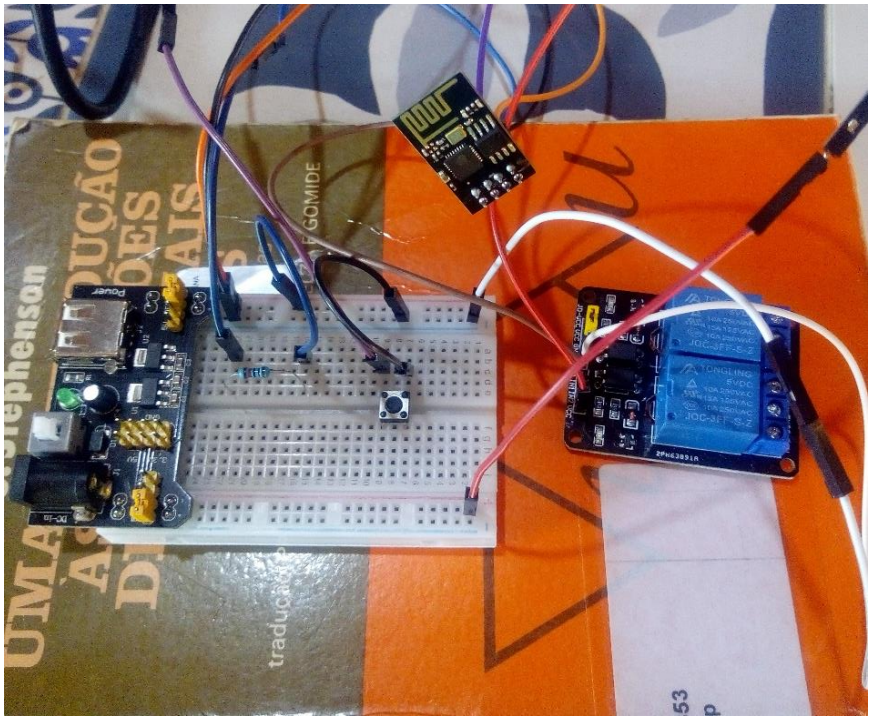
A figura 26 mostra as mensagens enviadas ao IDE pela comunicação serial. Para o teste foram utilizados dois tópicos (nome do pacote enviado/recebido via MQTT), sendo um deles chamado Atuador1/led1 e outro Atuador2/led2. O Atuador1/led1 serve para enviar informações do ESP-01 para o dispositivo móvel e o Atuador2/led2 do dispositivo móvel para o ESP-01 ambos mediados pelo servidor. Como é possível perceber na imagem todas as informações trocadas entre os dispositivos são registradas, permitindo verificar falha de comunicação.

5.3 APLICAÇÃO DO PROTÓTIPO

Depois de programado, foi necessário mudar o circuito do protótipo para o formato de aplicação e iniciou-se outro teste. Essa etapa de testes serve para verificar

o funcionamento do protótipo em seu estágio final, sem estar conectado ao computador e com o sensor e atuador ligados ao sistema. A figura 27 ilustra o circuito confeccionado para aplicação do protótipo.

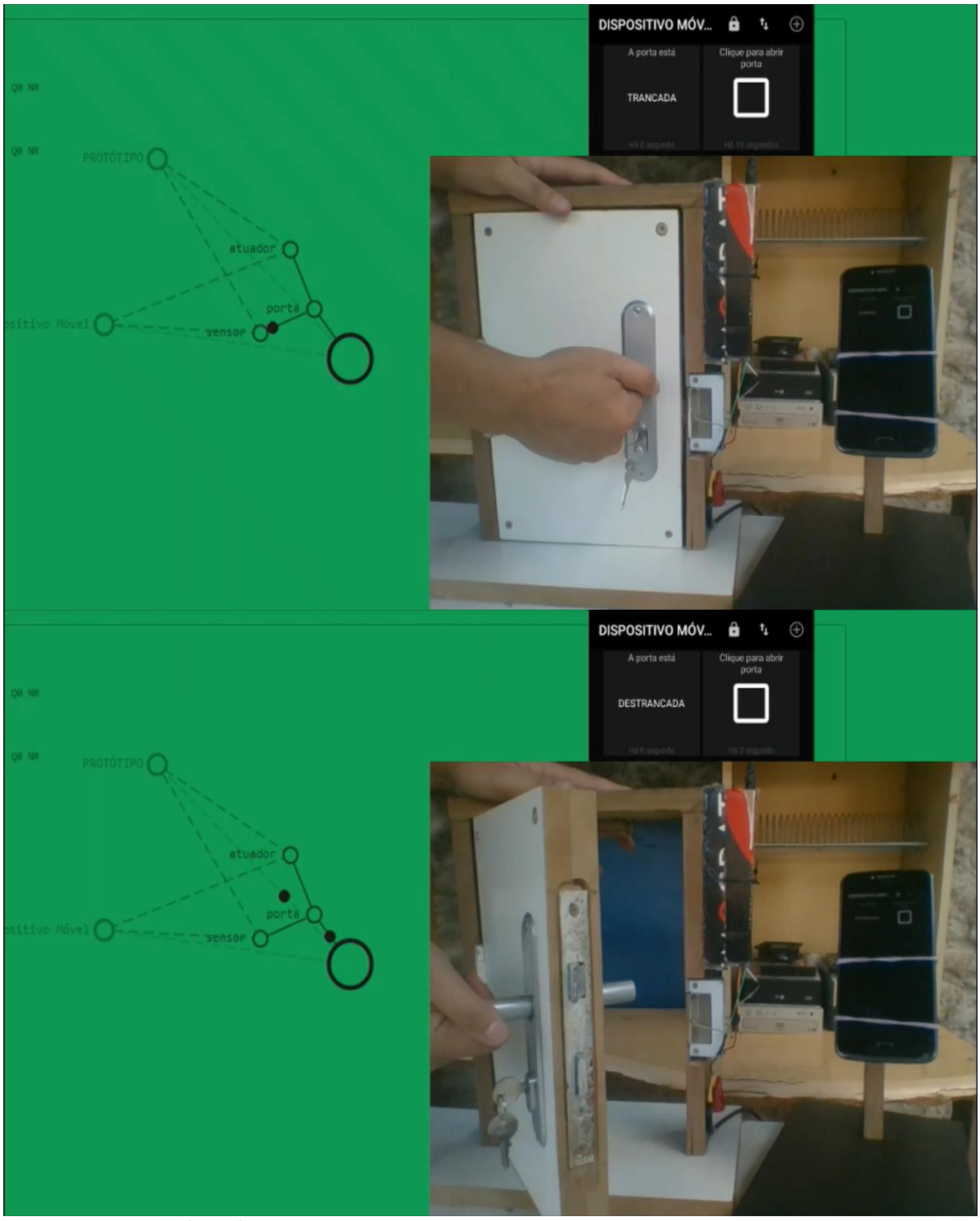
Figura 27 – Circuito para aplicação



Fonte: Os autores (2020)

Para apresentação do protótipo foi construído uma porta com batente em miniatura, dispositivo elétrico para abrir a porta (fecho elétrico) e fechadura com o protótipo criado anexo a ela, a figura 28 mostra o protótipo em sua versão final, junto com a tela do aplicativo *MQTT Dash* e o servidor em funcionamento.

Figura 28 – Versão final do protótipo[4]



Fonte: Os autores (2020)

Nota-se que na parte de cima da figura 28 a porta está trancada, a mensagem “TRANCADA” está aparecendo no aplicado do *smartphone* e Já embaixo foi realizada

[4] Para assistir vídeo do teste de funcionamento acesse: https://drive.google.com/file/d/1CJpgg7bxAmoHsN7yB6K7oeE9NZTCtBXQ/view?usp=sharing

a ação de destrancar a porta remotamente, então a fechadura elétrica abre, na tela do aplicativo aparece a mensagem “DESTRANCADA”.

6 CONSIDERAÇÕES FINAIS

As considerações finais estão divididas em quatro momentos, no primeiro momento encontram-se relatos sobre a experiência na realização do trabalho, o segundo trará informações sobre o processo de prototipagem e a relação entre teoria e prática, seguido das conclusões apresentadas pela equipe sobre a pesquisa exploratória e por fim sugestões para trabalhos futuros.

Não houve dificuldades em redigir o presente trabalho integrador, os membros da equipe estiveram presentes em todos os momentos da pesquisa. Na fundamentação teórica, tivemos facilidade em encontrar materiais recentes e confiáveis, proporcionando que o objetivo geral fosse atingido, assim como os específicos.

Encontrou-se dificuldades na realização da prototipagem simulada, pois embora existam ferramentas de visualização gráfica e simulação de circuitos, essas ferramentas não possuem suporte para todos os componentes necessários na confecção do protótipo físico, principalmente em relação à comunicação com a internet. Todavia foi possível encontrar soluções plausíveis aos empecilhos encontrados.

Com a metodologia aplicada foi possível associar os conteúdos aprendidos na sala de aula sobre programação, modelos e métodos de desenvolvimento de software e lógica aplicada. Assim como aplicar os conhecimentos adquiridos nas pesquisas do trabalho integrador dentro da sala de aula, antecipando alguns conteúdos que seriam aprendidos.

Ao terminar o presente trabalho integrador a equipe concluiu que atualmente com a indústria 4.0, a tecnologia está acessível e barata, não sendo destinada apenas a aqueles que possuem amplo conhecimento e capital, mas sim para todos que desejam aprender e conhecer novas tecnologias. E que qualquer pessoa pode construir um sistema de automação residencial em casa de baixo custo. Este sistema pode ser expandido para ambientes educacionais, coworking, ou seja, espaços em que há maior fluxo de pessoal, e seria um reforço para a segurança.

Percebeu-se também a relevância do trabalho realizado, pois devido ao fato tratar-se de um assunto da atualidade, o mesmo tem influência no desenvolvimento de novos conhecimentos e potencial para o mercado de trabalho futuro.

A presente pesquisa gerou diversas sugestões para realização de projetos futuros, com foco não apenas na automação residencial, mas também na indústria e comercio, todavia para manter mesmo conceito de sensor que avisa quando a porta está trancada, destrancando-a se necessário, surgiu a ideia de construir um sistema semelhante utilizando tecnologia M2M, para que o projeto possa ser aplicado em diversos ambientes e sem a necessidade da conexão WiFi.

REFERÊNCIAS

ARDUINO. **What is Arduino?** Disponível em: < https://www.arduino.cc/en/Guide/Introduction#>. Acesso em: 24 abr. 2020.

ACCARDI, Adonis; DODONOV, Eugeni. Automação residencial: elementos básicos, arquiteturas, setores, aplicações e protocolos. **Tecnologias, Infraestruturas e software**. São Carlos – SP, vol. 1, n. 2. p 156-166, 2012.

AMBROSE, Gavin; HARRIS, Paul. **Design thinking**. Tradução de Mariana Belloi. rev. Porto Alegre: Bookman, 2011.

ASCENCIO, Ana F. G.; CAMPOS, Edilene A. V. de. **Fundamentos da programação de computadores:** algoritmos, Pascal, C/C++ (padrão ANSI) e Java. 3 ed. São Paulo: Pearson, 2012.

BATISTA, Micheline D. G. Pesquisa na internet: considerações metodológicas. In: ENCONTRO NORTE E NORDESTE DE CIÊNCIAS SOCIAIS PRÉ-ALAS BRASIL, 15, 2012, Teresina. **Anais...** Teresina-PI: UFPI, 2012.

CÓRDOVA JUNIOR, Ramiro S.; SANTOS, Sidney C. B. dos; KISLANSKY, Pedro. **Fundamentos Computacionais**. Porto Alegre: SAGAH, 2018.

COELHO, Pedro M. N. **Rumo à Indústria 4.0**. 65 f. Dissertação (Mestrado) – Universidade de Coimbra, Coimbra, 2016.

ESPRESSIF. **ESP8266EX**. Disponível em: < https://fritzing.org/home/>. Acesso em: 24 abr. 2020.

FRITZING. **FRITZING**. Disponível em: < https://www.arduino.cc/en/Guide/Introduction#>. Acesso em: 03 mai. 2020

GASPAR, João. **Google SketchUp pro 8 passo a passo**. 1 ed. São Paulo: ProBooks, 2012.

GIL, Antonio C. **Como Elaborar Projetos de Pesquisa**. 6. ed. São Paulo: Atlas, 2017.

GONÇALVES, Rangel L. M. **Automação residencial:** um estudo de caso da aplicação da internet das coisas. 58 f. Monografia (Graduação) – Universidade do sul de Santa Catarina, Florianópolis, 2019.

GOOGLE PLAY. **MQTT Dash (IoT, Smart Home)**. Disponível em: < https://play.google.com/store/apps/details?id=net.routix.mqttdash>. Acesso em: 01 mai. 2020.

GOOGLE PLAY. **Termo de servido do Google Play**. Disponível em: <https://play.google.com/intl/pt_br/about/play-terms/index.html>. Acesso em: 01 mai. 2020.

LIMA, Alison G. de; PINTO, Giuliano S. Indústria 4.0: um novo paradigma para a indústria. **Interface Tecnológica**. Taquaritinha – SP, vol. 16, n. 2. p 299 – 311, 2019.

MACHADO, Luis C. **Modelo conceitual integrando prototipagem rápida e delineamento de experimentos na concepção de novos produtos**. 95 f. Dissertação (Mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2013.

MANZANO, José A. N. G. **Programação de computadores com C++**. 1 ed. São Paulo: Érica, 2014.

MARCONI, Marina de A.; LAKATOS, Eva M. **Fundamentos de Metodologia Científica**. 8. ed. São Paulo: Atlas, 2017.

MORAIS, Izabelly S. de *et al*. **Introdução a big data e internet das coisas (IoT)**. Porto Alegre: SAGAH, 2018.

OLIVEIRA, Sérgio de. **Internet das coisas em esp8266, Arduino e raspberry pi**. 1 ed. São Paulo: Novatec, 2017.

SANTOS, Bruno P. *et al*. Internet das coisas: da teoria à prática. In. XXXIV SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUIDOS, 2016, Porto Alegre. **Anais...** Porto Alegre: SBC, 2016. Disponível em: <http://www.sbrc2016.ufba.br/downloads/anais/MinicursosSBRC2016.pdf>. Acesso em: 10 abr. 2020.

SHIFTR.IO. **Dashboard**. Disponível em: <https://shiftr.io/>. Acesso em: 08 mai. 2020.

SKETCHUP. **O melhor caminho para o 3D, de graça**. Disponível em: < https://www.sketchup.com/pt-BR/plans-and-pricing/sketchup-free>. Acesso em: 24 abr. 2020.

SOUZA, Marcelo V. de. **Domótica de baixo custo usando princípios de IoT**. 50 f. Dissertação (Mestrado) – Universidade Federal do Rio Grande do Norte, 2016.

ANEXO A – CÓDIGOS UTILIZADOS PARA PROGRAMAÇÃO DO ESP-01

Segue abaixo código fonte utilizado no protótipo.

```
#include<ESP8266WiFi.h> //Biblioteca para conexão WiFi
#include<PubSubClient.h>
//_____Definição de LOGINS e SENHAS_____

//WiFi
const char* ssid = "xxxxxxxxx"; // Rede WiFi a ser conectada
const char* senha = "xxxxxxxxxx"; //senha da rede WiFi a ser conectada
//MQTT
const char* mqttServidor = "broker.shiftr.io"; //Endereço do servidor a ser conectado
const char* mqttUsuario = "212bfd15"; //Nome do usuário que será conectado no servidor
const char* mqttSenha = "ba8880f28f740214"; //senha do dispositivo MQTT
const int mqttPorta = 1883; //número da porta que será conectada no servidor, a padrão é 1883
const char* mqttTopicoSub1 = "porta/sensor"; //endereço dos pacotes enviados/recebidos, podem haver
mais
const char* mqttTopicoSub2 = "porta/atuador";

WiFiClient EquipeOmega; //Cria o nome do cliente que usará o WiFI
PubSubClient client(EquipeOmega); //Cria uma instância "client” para ser utilizada pelo MQTT
//_____

//_____Chamada de variáveis Globais_____

#define botao 1
#define rele 2

//_____

//_____Criação de Funções Auxiliares_____

//Função que efetua a conexão com o WiFi
void conectar_WiFi()
{
  WiFi.mode(WIFI_STA); //Definindo o modo de conexão como Station Mode
  WiFi.begin(ssid, senha); //Chamada da conexão WiFi informando os parâmetros
  Serial.println(""); //imprimindo um espaço em branco na linha de baixo do debug

  //Comando para ficar tentando conectar na rede
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500); // cria uma pausa no programa para a comunicação
    Serial.println("Conectando no WiFi"); //O Serial.print e o Serial println servem para escrever
mensagem no debug
  }

  //depois de conectado
  Serial.print("Conectado na rede: ");
  Serial.println(ssid);
  Serial.print("Com o endereço de IP: ");
  Serial.println(WiFi.localIP()); // a função WiFi.localIP() estabelece o ip local do dispositivo

}

//Função que efetua a conexão com o Servidor MQTT
void conectar_MQTT()
{
```

```
  client.setServer(mqttServidor, mqttPorta); //Inicia a conexão com servidor enviando os parâmetros
  client.setCallback(recebe_Valores); //Define o que acontece quando o client recebe um pacote de informações
  Serial.println(" ");

  //Comando para ficar tentando conectar ao Servidor MQTT
  while(!client.connected())
  {
    Serial.println("Conectando ao Servidor MQTT");

    if(client.connect("PROTÓTIPO", mqttUsuario, mqttSenha))//após conectar ao servidor, é necessário

    {
      Serial.println("Conexão com servidor MQTT confirmada!");
    }
    else
    {
      Serial.print("Falha na conexão com o servidor MQTT, estado: ");
      Serial.println(client.state());
      delay(2000);
    }
  }

  //depois de conectado

  //subescreve no tópico
  client.subscribe(mqttTopicoSub1);
  client.subscribe(mqttTopicoSub2);
}

//Função para manter conectado ao Servidor

void reconectar_MQTT()
{
  if(!client.connected())
  {
    Serial.println("Conexão com o Servidor MQTT perdida, reconectando");
    conectar_MQTT();
  }
}

// Função para manter conectado ao WiFi

void reconectar_WiFi()
{
  if(WiFi.status() != WL_CONNECTED)
  {
    conectar_WiFi();
  }
}

//Função para enviar infomações ao servidor MQTT
void envia_Valores()
{

  static bool estadoBotao = HIGH;
  static bool estadoBotaoAnt = HIGH;
  static unsigned long debounceBotao;

  estadoBotao = digitalRead(botao);
```

```
  if (  (millis() - debounceBotao) > 30 ) {  //Elimina efeito Bouncing
                                    //OBS: millis() é o comando que calcula os
                                    //milissegundos que passaram desde que o programa iniciou
    if (!estadoBotao && estadoBotaoAnt) {

       //Botao Apertado
       client.publish(mqttTopicoSub1, "TRANCADA");

       Serial.println("Botao APERTADO. Payload enviado.");

       debounceBotao = millis();
    } else if (estadoBotao && !estadoBotaoAnt) {

       //Botao Solto
       client.publish(mqttTopicoSub1, "DESTRANCADA");
       Serial.println("Botao1 SOLTO. Payload enviado.");

       debounceBotao = millis();
    }

  }
  estadoBotaoAnt = estadoBotao;
}

//Função Para receber Informações do servidor MQTT
void recebe_Valores(char* topic, byte* payload, unsigned int length)
{
   static unsigned long espera; //variavel que servira para rearmar fechadura
   String msg;

   //obtem a string do payload (pacote) recebido
   for(int i = 0; i < length; i++)
   {
      char c = (char)payload[i];
      msg += c;
   }

   //Escreve a mensagem recebida pelo servidor no debug
   Serial.print("Mensagem chegou do tópico: ");
   Serial.println(topic);
   Serial.print("Mensagem: ");
   Serial.println(msg);
   Serial.println();
   Serial.println("_____________________");

   if (msg == "0") {

   digitalWrite(rele, HIGH); //se a mensagem recebida for “0” mantem relé desativado

   }

   if (msg == "1") {

   digitalWrite(rele, LOW); //se a mensagem recebida for “1” ativa relé

   delay(2000);

   client.publish(mqttTopicoSub2, "0");
```

```
    }
}
//_____________________________________________________________________

void setup()
{
  Serial.begin(115200); //Inicia a comunicação serial com a velocidade
                        //de 115200 para comunicação serial
                        // que serve para acompanhar o funcionamento do programa
  delay(10);

  pinMode(botao,INPUT_PULLUP);
  pinMode(rele, OUTPUT);

  conectar_WiFi(); //chama a função conectar_WiFi
                   //o mesmo ocorrerá com as outras funções auxiliares criadas
  conectar_MQTT();

}

void loop() {

  reconectar_MQTT();
  reconectar_WiFi();
  envia_Valores();

  client.loop();//comando para manter o sincronismo
                // entre o dispositivo e o servidor
                //fundamental para o funcionamento correto
                // das funções envia_Valores e recebe_Valores

}
```