UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM ENGENHARIA DE SOFTWARE

DAVI MAGALHÃES LIMA

UM SISTEMA DE AUTOMAÇÃO RESIDENCIAL MODULAR SOB INTERNET DAS COISAS E COMPUTAÇÃO UBÍQUA

QUIXADÁ

2018

DAVI MAGALHÃES LIMA

UM SISTEMA DE AUTOMAÇÃO RESIDENCIAL MODULAR SOB INTERNET DAS COISAS E COMPUTAÇÃO UBÍQUA

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software.
Área de concentração: Computação.

Orientador: Me. Luis Rodolfo Rebouças Coutinho

QUIXADÁ
2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L697s Lima, Davi Magalhães.
Um sistema de automação residencial modular sob Internet das Coisas e Computação Ubíqua / Davi Magalhães Lima. – 2018.
67 f.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2018.
Orientação: Prof. Me. Luis Rodolfo Rebouças Coutinho.

1. Automação Residencial. 2. Internet das Coisas. 3. Computação Ubíqua. I. Título.

CDD 005.1

DAVI MAGALHÃES LIMA

UM SISTEMA DE AUTOMAÇÃO RESIDENCIAL MODULAR SOB INTERNET DAS COISAS E COMPUTAÇÃO UBÍQUA

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software.
Área de concentração: Computação.

Aprovado em: ___/___/___

BANCA EXAMINADORA

Me. Luis Rodolfo Rebouças Coutinho (Orientador)
Universidade Federal do Ceará (UFC)

Ma. Antonia Diana Braga Nogueira
Universidade Federal do Ceará (UFC)

Ma. Jéssyka Flavyanne Ferreira Vilela
Universidade Federal do Ceará (UFC)

Dr. Marcio Espíndola Freire Maia
Universidade Federal do Ceará (UFC)

À minha mãe, irmãos e à memória de meu pai.

AGRADECIMENTOS

Agradeço a minha família, em especial à minha mãe, pela dedicação e esforço em me possibilitar a oportunidade de me graduar, e aos meus irmãos, pelo apoio moral e financeiro. Agradeço ainda por me incentivarem em meus planos.

À minha namorada que com resiliência esteve comigo quando precisei.

Aos poucos bons colegas.

E aos interioranos de São Paulo que, supostamente, no começo do século passado, criaram por acidente a caipirinha e, graças a esta, família e namorada, pude manter minha sanidade mental neste fim de faculdade.

“Cada um terá a vista da montanha que subir.”

(Icaro Fonseca)

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema de automação residencial modular que utiliza dos paradigmas Internet das Coisas e Computação Ubíqua em seu atuador de automação. Atuador este que é capaz de controlar a iluminação e tomadas prediais de modo pervasivo no ambiente, com uma interface por meio de uma aplicação móvel. Este sistema foi proposto e desenvolvido para agregar ao catálogo de sistemas de automação com o diferencial de proporcionar facilidade de instalação, configuração e manutenção, tudo sem a necessidade de mão de obra especializada para tanto. O seu desenvolvimento ocorreu por meio de um processo de software que melhor se adequava ao projeto, o qual possuía fases para a elucidação da proposta até a realimentação sobre o produzido, o que por meio dos resultados foi possível constatar o sucesso do desenvolvimento desta pesquisa. O trabalho finalizou por apresentar um sistema de automação residencial com atuadores de automação embutidos no ambiente, capazes de automatizar a iluminação e tomadas residenciais, por meio de uma aplicação móvel e da internet, de modo facilitado.

Palavras-chave: Automação Residencial. Internet das Coisas. Computação Ubíqua.

ABSTRACT

This monograph presents the development of a modular home automation system that uses the paradigm Internet of Things and Ubiquitous Computing in its automation actuator. This actuator is able to control the lighting and building plugs in a pervasive way in the environment, with an interface through a mobile application. This system was proposed and developed to add to the catalog of automation systems with the differential of providing ease of installation, configuration and maintenance, all without the need of specialized labor for both. Its development took place through a software process that was better suited to the project, which had phases for the elucidation of the proposal until the feedback on the produced, which through the results it was possible to verify the success of the development of this research. This monograph ended up presenting a residential automation system with automation actuators built into the environment, capable of automating lighting and residential sockets, through a mobile application and the internet, in a facilitated way.

Keywords: Home Automation. Internet of Things. Ubiquitous Computing.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A tecnologia evolui em prol de solucionar problemas, trazer comodidade, praticidade e segurança ao dia a dia de seus usuários. No contexto habitacional, aumenta cada vez mais a procura pela antes surreal "casa do futuro", casa automatizada ou *Smart Home* e, consequentemente, surgem cada vez mais pesquisas e soluções de mercado com propostas para atender essa demanda. Porém, esses produtos ainda se encontram pouco flexíveis, muitos dependendo de projetos específicos para implantação (BRUSH *et al.*, 2011).

Nesse contexto, pesquisas em Internet das Coisas, um paradigma que permite controlar equipamentos eletrônicos por meio da internet (XIA *et al.*, 2012), visam agregar ao catálogo de soluções de automação residencial ao implantar interfaces à internet em aparelhos residenciais, a fim de torná-los controláveis de modo móvel e amigável (RICQUEBOURG *et al.*, 2006). Além disso, trabalhos em Computação Ubíqua, o paradigma que aplica a computação de modo onipresente no ambiente (WEISER, 1991), propõem embutir sistemas de computação ao ambiente residencial a fim de torná-los pervasivos e, assim, imperceptíveis (ARAUJO, 2003).

Diante disso, este trabalho possui como objetivo principal desenvolver uma solução para flexibilização de implantações de sistemas, por meio de um sistema modular de automação residencial que embarca os seus atuadores em caixas da rede elétrica predial, como tomadas e interruptores de iluminação, utilizando conceitos como Internet das Coisas e Computação Ubíqua, aprofundados no Capítulo 3. Tal sistema busca proporcionar facilidade de implantação, manutenção e expansão, além de possuir interações com os usuários por meio de um aplicativo para dispositivos móveis.

Ademais, este trabalho possui como objetivos específicos:

a) desenvolver um protótipo de atuador de automação residencial capaz de ligar e desligar interruptores de iluminação da rede elétrica e que seja compacto o suficiente para ser embutido a esses interruptores tradicionais;
b) desenvolver um protótipo de atuador de automação residencial capaz de ligar e desligar tomadas da rede elétrica e que seja compacto o suficiente para ser embutido a essas tomadas tradicionais;
c) desenvolver uma aplicação para dispositivos móveis em prol da interface entre o sistema de automação e os usuários, e que permita a instalação e utilização dos atuadores de automação por meio da internet;
d) configurar um servidor a fim de protocolar a comunicação do aplicativo móvel

de interface com os usuários, comentado na alínea “c”, por meio da internet, com os atuadores do sistema de automação residencial, mencionados nas alíneas “a” e “b”.

Este projeto visa solucionar os principais problemas pertinentes à maioria das soluções de automação residencial existentes no estado da prática atualmente, dentre os quais cita-se (BRUSH *et al.*, 2011):

1. dependência de reforma elétrica predial para implantação de atuadores;
2. necessidade de mão de obra especializada para instalação, manutenção e expansão;
3. produtos dependentes de fornecedores dedicados – não havendo extenso catálogo de sistemas de prateleira;
4. alto custo para instalação, manutenção e expansão.

A proposta procura atender a quaisquer tipos de usuários residentes em moradia predial que possua rede elétrica, e tenha idade segura para utilização de aparelhos móveis e elétricos, além de ser destinado a usuários de qualquer classe social.

O projeto contribui para a crescente pesquisa por soluções tecnológicas sob o paradigma da Internet das Coisas (XIA *et al.*, 2012), explicado no Capítulo 3, a qual inclui a automação residencial – controle de tomadas, luzes e aparelhos domésticos, por meio da internet – que visa proporcionar conforto, economia e segurança aos usuários.

No próximo capítulo são comentados os trabalhos relacionados dos estados da arte e da prática. No Capítulo 3 são explanados os conceitos e paradigmas chaves para entendimento e execução deste trabalho, dentre os quais: Automação Residencial, Internet das Coisas, Computação Ubíqua, Computação Móvel e Computação em Nuvem – Capítulo 3. No Capítulo 4 são apresentadas todas as etapas e atividades para desenvolvimento do sistema de automação proposto neste trabalho. O capítulo Resultados e Discussão, de número 5, detalha o relato de experiência do desenvolvimento deste trabalho e seus principais resultados. No Capítulo 6, Considerações Finais, são apresentadas as conclusões e as sugestões para trabalhos futuros.

2 TRABALHOS RELACIONADOS

A patente “Expandable Home Automation System” (LAUNEY *et al.*, 1992) descreve um sistema de automação residencial capaz de controlar incontáveis aparelhos domésticos, bem como aparelhos externos à casa, sob múltiplas linguagens de comunicação suportadas. O sistema oferece a possibilidade do controle de sistemas de iluminação, segurança, além da conexão com diversos sensores. Possui, ainda, diversos meios de interface com o usuário, destacando as telas sensíveis ao toque (*touchscreens*) e sistemas de reconhecimento de voz. No entanto, o sistema registrado depende de mão de obra especializada para instalação e manutenção, além de depender de fios para comunicação entre os seus módulos, tornando o sistema dispendioso em tempo e recursos financeiros.

O registro de patente “Energy Management and Home Automation System” (JR; ROMANOWIZ; STAPLES, 1996) volta-se ao controle elétrico de sistemas de controle de temperatura, aquecimento e arrefecimento, acoplados a controladores do sistema de automação. Estes são responsáveis por executar comandos recebidos por sinais digitais de uma central que estabelece um cronograma de eventos para operar cada dispositivo acoplado, conforme programado pelos usuários. Como descrito, o sistema registrado nessa patente volta-se ao controle da rede elétrica responsável por alimentar sistemas de controle de temperatura dependentes de cronograma, não possuindo foco em automação residencial pertinente ao conforto e comodidade de moradores de uma residência inteligente.

Na patente “Home Automation System”, de Humphries *et al.* (1997), um sistema de automação residencial modular é proposto, o qual possui módulos capazes de efetuar o controle de iluminação, segurança e aparelhos de entretenimento. Apesar dessas características, a patente possui diferentes painéis próprios para interface com os usuários, além de sugerir a necessidade de mão de obra especializada para instalação, manutenção e treinamento para utilização do sistema.

No artigo de Gill *et al.* (2009) é proposto um sistema modular a fim de solucionar problemas de falta de flexibilidade de arquiteturas, comunicando seus dispositivos sob rede sem fio, por meio do protocolo *ZigBee* (ZIGBEE, 2006) e comunicando-se com a internet por Wi-Fi, utilizando um *gateway* para interoperabilidade e interface com usuário por meio de uma aplicação móvel. Para prova de conceito, foi implementado e avaliado um sistema composto de quatro módulos, sendo um para controle de iluminação, outros dois para válvula de radiador e sistema de segurança e, por último, um módulo de controle remoto. Porém, como no artigo

citado anteriormente, este não provê facilidade de instalação e utilização por parte de usuários finais leigos.

O artigo “Design and Implementation of Smart Home Energy Management Systems based on ZigBee” (HAN; LIM, 2010) propõe um sistema modular sob rede sem fio, que utiliza dos protocolos IEEE 802.15.4 e *ZigBee* para comunicação entre seus módulos. Esse sistema é capaz de controlar luzes, válvulas de gás, cortinas, televisores e ar-condicionados, além de interagir com diversos dispositivos móveis, como PDAs e telefones celulares. Entretanto, diferencia-se do proposto neste trabalho por não se voltar a usuários que não possuam conhecimento técnico para instalação e manutenção.

O projeto descrito no artigo “Home Automation System”, de Petrov, Seru e Petrov (2011), é o que possui mais características que se aproximam da proposta diferenciada deste trabalho. Voltado à automação de interruptores de luz e portas para pessoas com deficiência física, acopla os módulos de funcionamento a interruptores de luz, facilitando, desse modo, a instalação. Possui, como interface com os usuários, computadores e dispositivos móveis. Os módulos citados comunicam-se com uma central de processamento via radiofrequência segura. Porém, o artigo descreve que a central de processamento e as interfaces com o usuário comunicam-se via *bluetooth*, o que limita o alcance para utilização e não permite o acesso ao sistema de automação por uma rede externa à residência, como a internet.

Outra patente, a “Home Automation Project” (KUNJUMON; PINTO, 2016), registra um sistema de automação residencial sob a proposta de um sistema de casa inteligente que possibilita o controle de luzes, ventiladores, aparelhos de ar-condicionado e eletrodomésticos com interface infravermelha, como TVs e DVDs. O sistema também conta com inteligência capaz de ligar e desligar aparelhos, na medida em que identifica, por meio de sensores, a presença de moradores na residência.

A interface entre os usuários e o sistema, porém, é feita por meio de um *website* hospedado em um módulo para a plataforma de prototipação embarcada Arduino, que pode ser acessado por navegadores em computadores e aparelhos móveis. Por não ser um sistema modular que permita uma fácil instalação e expansão e, por utilizar de plataforma de prototipação para implementação, esse sistema é limitado pela necessidade de mão de obra especializada.

Em resumo, os principais pontos sobre os trabalhos relacionados e o proposto neste trabalho são destacados no Quadro 1 a seguir.

Quadro 1 – Trabalhos relacionados.

Autores:	**Capaz de controlar:**	**Interface(s) com usuários:**	**Comunicação por meio de:**	**Controle externo à residência:**	**Mão de obra especializada:**
Launey *et al.* 1992.	Incontáveis aparelhos.	Telas *touchscreen*; Reconhecimento de voz.	Fios.	Não.	Sim.
JR *et al.* 1996.	Voltados a temperatura.	Painel.	*Wireless*.	Não.	Sim.
Humphries *et al.* 1997.	Iluminação, segurança e entretenimento.	Diversos painéis.	Fios.	Não.	Sim.
Gill *et al.* 2009.	Iluminação, segurança e um controle remoto.	*Smartphones*.	Zigbee e Wi-Fi.	Sim.	Sim.
Han e Lim de 2010.	Iluminação, válvulas, cortinas, televisores, etc.	PDAs e dispositivos móveis.	Zigbee.	Não.	Sim.
Petrov, Seru e Petrov de 2011.	Iluminação e portas.	Computadores e dispositivos móveis.	Radiofrequência e *bluetooth*.	Não.	Não.
Kunjumon e Pinto de 2016.	Iluminação, eletrodomésticos, afins.	*Website* em plataforma de prototipação Arduino.	Fios.	Sim.	Sim.
Este trabalho.	Iluminação e tomadas.	Dispositivos móveis.	Wi-Fi.	Sim.	Não.

Fonte – Próprio autor.

3 FUNDAMENTAÇÃO TEÓRICA

Alguns conceitos e paradigmas precisam ser definidos a fim de explanar os pilares que fundamentam e direcionam o desenvolvimento deste trabalho. Desses, iniciamos pelo alvo deste projeto: Automação Residencial.

3.1 Automação Residencial

Automação residencial, residência automatizada ou inteligente, domótica ou, em inglês, *smart home*, são expressões utilizadas para definir dispositivos inteligentes instalados em moradias residenciais. Esses dispositivos visam proporcionar comodidade, conforto e segurança ao ambiente do lar, sendo capazes de coletar, transportar e disponibilizar informações pertinentes às interações entre os residentes da edificação (RICQUEBOURG *et al.*, 2006).

Um projeto de sistema de automação residencial busca conectar equipamentos eletroeletrônicos residenciais a atuadores, sensores, centrais de processamento e *gateways* de interface com a internet, a fim de serem configurados, pelos moradores da residência, podendo ainda funcionar de modo autônomo, baseados no comportamento dos residentes, tudo voltado ao estabelecimento de um maior conforto e segurança do ambiente (RICQUEBOURG *et al.*, 2006).

Essa conexão entre eletroeletrônicos residenciais e dispositivos computacionais, para suas utilizações, ocorre popularmente, na atualidade, por meio da Internet das Coisas.

3.2 Internet das Coisas

A Internet das Coisas (do inglês, *Internet of Things*) é uma revolução tecnológica que define o acesso, por meio da internet, a quaisquer equipamentos elétricos ou eletrônicos, possibilitando controlá-los e programá-los para a realização de suas funções e tarefas (XIA *et al.*, 2012).

Para melhor entendimento, um exemplo no contexto residencial: o controle de aparelhos de televisão ou som, de qualquer cômodo da casa, mesmo o usuário não estando presente nesse cômodo, e o acionamento de um ar-condicionado pouco antes da chegada dos moradores à residência – adequando a temperatura ambiente ao bem-estar dos residentes.

No que se refere à instalação de um sistema de automação no ambiente residencial, tem-se que pode ocorrer de diversas maneiras, numa especial de modo a encontrar-se imperceptível aos moradores, ao utilizar do paradigma Computação Ubíqua, oferecendo, assim,

maior conforto.

3.3 Computação Ubíqua

Computação Ubíqua é o termo usado para descrever o paradigma que aplica a computação de modo onipresente no ambiente. Foi descrita pela primeira vez pelo pesquisador Weiser (1991), o qual previu que a próxima grande era da computação será a de computadores não visíveis aos olhos, os quais intercederão pelos humanos sem que estes tenham de expressar qualquer comando explícito (WEISER, 1991).

A Computação Ubíqua pode ser aplicada de várias formas ao ambiente, podendo estar sob sistemas computacionais invisíveis que aprendem sobre o comportamento dos utilizadores, a fim de adiantar funcionalidades pertinentes a padrões de interação dos usuários com o ambiente, ou ainda, encontrar-se sob uma de suas principais características: a pervasividade (ARAUJO, 2003).

A pervasividade na Computação Ubíqua, também chamada de Computação Pervasiva, define sistemas computacionais embutidos a objetos comuns do dia a dia, com o objetivo de se tornarem imperceptíveis no ambiente (ARAUJO, 2003). O intento deste trabalho é utilizar sistemas computacionais embutidos a interruptores de iluminação e tomadas residenciais a fim de torná-los discretos à percepção.

Ademais, para a interação entre os usuários e o sistema de automação devemos utilizar de uma aplicação em dispositivo móvel, sobre o conceito Computação Móvel.

3.4 Computação Móvel

Segundo o livro de Johnson e Maltz (1996), a Computação Móvel é definida como o uso de dispositivos móveis (transportáveis) capazes de se comunicarem por meios sem fio. Isso se evidencia na atualidade por meio de aparelhos celulares/*smartphones*, que são dispositivos computacionais transportáveis e com conexão sem fio, para efetuação de chamadas, acesso à internet e outros fins.

Na última década, dispositivos de computação móvel populares tornaram-se computadores pessoais portáteis que possuem sistemas operacionais complexos para gerência de seus recursos. Tais recursos podem ser tanto físicos, como câmeras fotográficas, quanto lógicos, por meios de aplicativos como calculadoras. Existem diversos meios de implementação desses

aplicativos para dispositivos móveis e um deles é o desenvolvimento híbrido.

3.4.1 Desenvolvimento Híbrido

Desenvolvimento híbrido é a implementação de sistema que combina diferentes técnicas e linguagens de programação voltadas para diferentes fins, com o objetivo de portabilizar o produto gerado, por exemplo, no intento deste trabalho: a combinação de desenvolvimento WEB (linguagens HTML5, CSS3 e Javascript 7) com o empacotamento nativo de plataformas de desenvolvimento móvel como Android (GOOGLE, 2017) e iOS (APPLE, 2018) (CAMPAGNOLI, 2016). Isso tudo permite escrever, uma única vez, um código-fonte e deste gerar a mesma aplicação para diferentes plataformas.

Para que seja possível uma comunicação entre sistemas computacionais por meio da internet pode-se fazer necessário um terceiro sistema para intermediar a interação, o qual pode ser implantado num ambiente que ofereça melhor controle sobre os recursos computacionais necessários, por meio do paradigma Computação em Nuvem.

3.5 Computação em Nuvem

Computação em Nuvem é um modelo de acesso, sob demanda, a uma rede compartilhada de recursos de computação, como servidores, armazenamento e serviços, que pode ser facilmente gerenciado (MELL; GRANCE *et al.*, 2011). Segundo estes autores, o modelo é composto de cinco características essenciais:

a) *auto-serviço sob demanda:* um usuário pode solicitar mais recursos de computação e ser provido, sem necessidade de interação humana;
b) *acesso via rede:* os recursos são oferecidos via rede e acessados por meio de mecanismos padrões das plataformas dos clientes;
c) *agrupamento de recursos:* múltiplos recursos físicos e virtuais são fornecidos sobre uma mesma interface e atribuídos e reatribuídos dinamicamente de acordo com a demanda;
d) *elasticidade rápida:* os recursos podem ser aumentados e diminuídos, dinamicamente, sob demanda;
e) *medição de serviço:* as medidas de uso são abstraídas, monitoradas e reportadas aos clientes de modo amigável.

Por fim, para o desenvolvimento do sistema deste trabalho devemos definir Processo de Desenvolvimento de Software para que, por meio deste, possamos melhor realizar a etapa de desenvolvimento da presente pesquisa.

3.6 Processo de Desenvolvimento de Software

Segundo Pressman (2011), Processo de Desenvolvimento de Software é a metodologia que compreende as atividades, ações e tarefas necessárias para desenvolver um produto de software com qualidade.

Existem diversos modelos de processos de desenvolvimento de software, dos quais podemos citar o modelo cascata, que sugere uma abordagem sequencial e sistemática ao desenvolvimento, o modelo incremental, o qual gera incrementos entregáveis desenvolvidos de forma escalonada, e o modelo evolucionário, que é um modelo iterativo que permite desenvolver versões cada vez mais completas do software (PRESSMAN, 2011).

O intento deste trabalho é utilizar um processo do modelo evolucionário: o Processo Evolucionário.

3.6.1 Processo Evolucionário

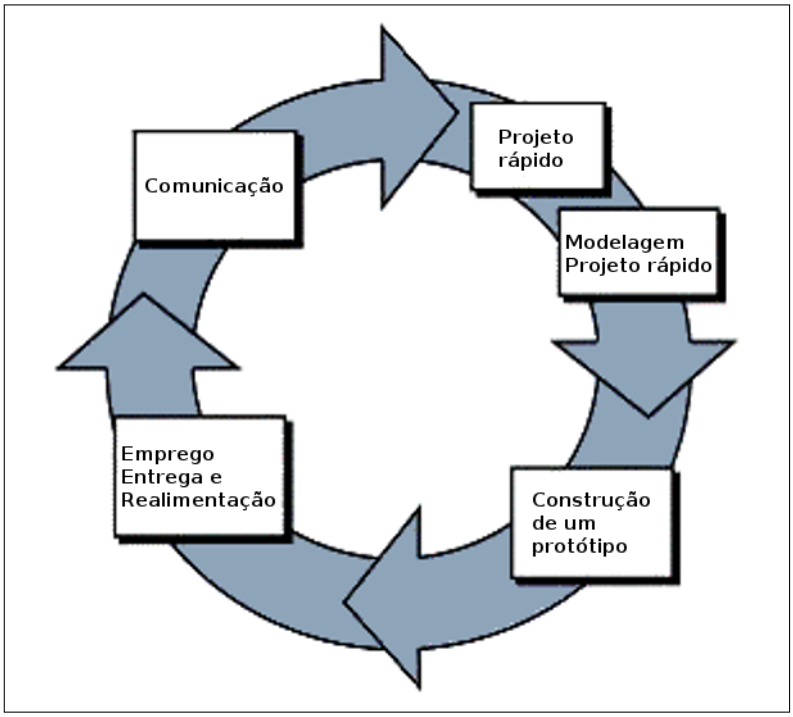
O Processo Evolucionário é um processo que utiliza o paradigma da prototipação de modo iterativo o que possibilita evoluir o produto de software, ao longo do tempo, devido o cliente não conseguir definir bem seus requisitos ou o desenvolvedor encontrar-se inseguro para implementar (PRESSMAN, 2011).

O Paradigma da Prototipação é fortemente utilizado no Processo Evolucionário para evitar/mitigar problemas quanto a má elucidação de requisitos, insegurança de desenvolvedores, raso conhecimento de ambiente de implantação e tecnologias necessárias, dentre outros motivos (PRESSMAN, 2011).

O Paradigma possui fases, como ilustradas na Figura 1 a seguir, as quais iniciam com a Comunicação, por meio de reunião com interessados no projeto a fim de identificar objetivos e requisitos bem conhecidos e, com isso, identificar conhecimentos superficiais acerca de outras necessidades. Após a comunicação ocorre a modelagem, em forma de “Projeto Rápido”, que evolui para a Construção do protótipo, o qual, ao fim, é avaliado pelos interessados por meio da Realimentação, com a finalidade de amadurecer o projeto e evoluí-lo. Por isso o Paradigma da

Prototipação é iterativo e integrante do Processo Evolucionário (PRESSMAN, 2011).

Figura 1 – Fases do Paradigma da Prototipação em seu ciclo iterativo.



Fonte – Pressman (2011).

Por fim, na modelagem da fase “Projeto Rápido”, ocorre a definição do padrão arquitetura escolhido para o sistema.

3.7 Padrão Arquitetural de Software

Um padrão arquitetural de software é um conhecimento arquitetônico consolidado acerca da organização de um sistema de software (PRESSMAN, 2011). A reutilização de um padrão oferece melhores chances de sucesso na definição de uma arquitetura de software diante da complexidade disso (PRESSMAN, 2011).

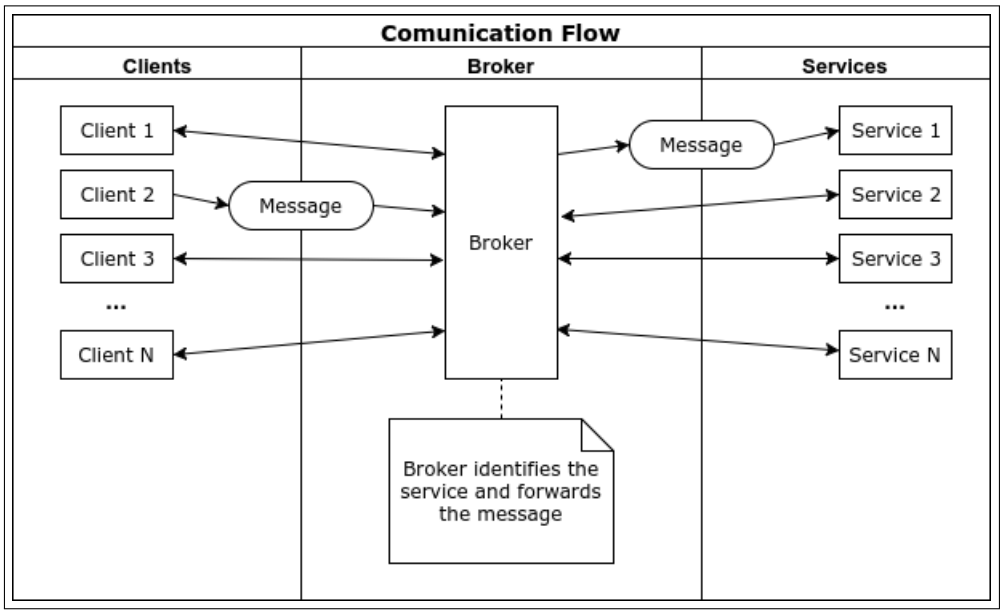
Há diversos padrões arquiteturais, entre os quais podemos citar o padrão modular, o qual volta-se ao desenvolvimento de um sistema em partes (módulos), o padrão componente e conector, voltado a implementação de uma coleção de serviços distribuídos em vários servidores, bem como o padrão arquitetural modelo visão e controlador, o qual volta-se a sistemas com alta modificabilidade da interface visual (PRESSMAN, 2011), e o padrão utilizado no presente trabalho, o padrão Broker, explicado a seguir.

3.7.1 Padrão Broker

O padrão arquitetural Broker volta-se ao desenvolvimento de sistemas que requerem uma comunicação entre uma coleção de serviços com múltiplos clientes. A implementação desses sistemas é complexa devido à preocupação com a forma como as partes se encontrarão, se conectarão e trocarão mensagens (BASS, 2007).

Diante disso, o padrão Broker permite que quando um cliente precise de um serviço, possa por meio de um *middleware* em *servidor* que intermedia as comunicações, estabelecer comunicação com o serviço desejado, conforme ilustra o diagrama da Figura 2.

Figura 2 – Diagrama: comunicação por *broker*.



Fonte – Próprio autor.

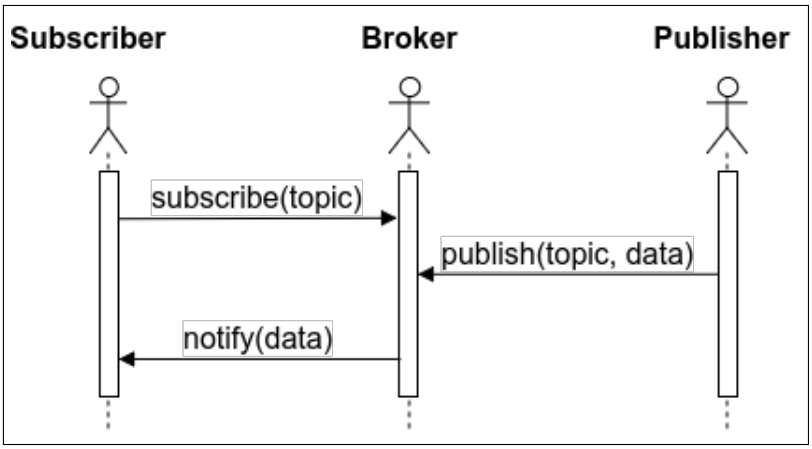
A Figura 2 utiliza o diagrama de atividades (BEZERRA, 2017) adaptado para ilustrar uma comunicação por meio do padrão arquitetural Broker. Na raia *Clients* há múltiplos clientes que se comunicam com múltiplos serviços na raia *Services*, porém a comunicação é intermediada pelo objeto *Broker* da raia de mesmo nome, que identifica o serviço para o qual a mensagem do *Client 2* está endereçada e encaminha adequadamente para o *Service 1*.

O encaminhamento da mensagem ao destinatário correto pode ocorrer por diversas lógicas, numa delas utiliza-se o paradigma Publish/Subscribe.

3.8 Paradigma Publish/Subscribe

O paradigma Publish/Subscribe é responsável por receber, enfileirar e enviar mensagens recebidas dos *publishers* para os *subscribers*, por meio de um *middleware* ou *broker*, conforme modela o diagrama de sequência (BEZERRA, 2017) da Figura 3.

Figura 3 – Diagrama: fluxo de comunicação sob *publish/subscribe*.



Fonte – Próprio autor.

No diagrama da Figura 3, sistemas distribuídos são representados pelos atores. O *Subscriber* inicialmente se inscreve (*subscribe*) num tópico para receber qualquer mensagem que o *Broker* receber no mesmo tópico comunicado, sob uma publicação (*publish*). Toda mensagem que o *Broker* recebe de publicadores (*Publisher*), ele confere os inscritos no tópico e dispara notificações (*notify*) (HUNKELER; TRUONG; STANFORD-CLARK, 2008).

Finalmente, no próximo capítulo, apresentamos as fases para o projeto e implementação do sistema de automação proposto neste trabalho.

4 DESENVOLVIMENTO

Para o desenvolvimento do sistema de automação residencial modular proposto neste trabalho, num primeiro momento, faz-se necessário definir um processo de Engenharia de Software que melhor acolha as necessidades pertinentes ao projeto. Para tanto, o processo definido é o Processo Evolucionário.

O Processo Evolucionário foi selecionado diante de suas características acolhedoras a projetos em que o desenvolvedor encontra-se inseguro quanto à concepção e definição do projeto, ver Subseção 3.6.1, e consequentemente quanto à arquitetura de projetos de software.

Nas Subseções 4.1 a 4.4 a seguir, estão documentadas as fases de desenvolvimento do sistema deste trabalho sob o processo definido, numa única iteração.

4.1 Comunicação

Nesta primeira fase, Comunicação, identificamos requisitos de sistema e negócio fundamentais para a proposta, descrevendo quais as principais funções e recursos desejados.

A seguir definimos os requisitos elucidados pelo autor, Quadros 3 a 10, em forma de estórias de usuário (PRESSMAN, 2011), seguindo o modelo do Quadro 2.

Quadro 2 – Modelo de estória de usuário: Estória de usuário *<identificador> - <título da estória>*.

<identificador> - <título>.	
Como um	<o ator>
Eu quero	<a funcionalidade>
De modo que	<o valor agregado>
Critério de aceitação	<o critério de aceitação para o teste de validação>

Fonte – Próprio autor.

Quadro 3 – Estória de usuário EU001 - Acender a luz.

EU001 - Acender a luz	
Como um	usuário
Eu quero	acender a luz
De modo que	eu tenha comodidade para não me locomover até o interruptor
Critério de aceitação	a luz pertinente deve ser acesa

Fonte – Próprio autor.

Quadro 4 – Estória de usuário EU002 - Apagar a luz.

EU002 - Apagar a luz	
Como um	usuário
Eu quero	apagar a luz
De modo que	eu tenha comodidade para não me locomover até o interruptor
Critério de aceitação	a luz pertinente deve ser apagada

Fonte – Próprio autor.

Quadro 5 – Estória de usuário EU003 - Verificar estado da luz.

EU003 - Verificar estado da luz	
Como um	usuário
Eu quero	saber o estado da luz (acesa ou apagada)
De modo que	eu tenha noção de sua situação
Critério de aceitação	ao alterar o estado da luz seu estado na interface deve corresponder

Fonte – Próprio autor.

Quadro 6 – Estória de usuário EU004 - Ligar circuito de tomada.

EU004 - Ligar circuito de tomada	
Como um	usuário
Eu quero	ligar a tomada
De modo que	eu tenha comodidade para não me locomover até a tomada
Critério de aceitação	a tomada pertinente deve ser ligada

Fonte – Próprio autor.

Quadro 7 – Estória de usuário EU005 - Desligar circuito de tomada.

EU005 - Desligar circuito de tomada	
Como um	usuário
Eu quero	desligar a tomada
De modo que	eu tenha comodidade para não me locomover até a tomada
Critério de aceitação	a tomada pertinente deve ser desligada

Fonte – Próprio autor.

Quadro 8 – Estória de usuário EU006 - Tamanho compacto dos atuadores.

EU006 - Tamanho compacto dos atuadores	
Como um	usuário
Eu quero	que os atuadores sejam embutidos aos seus equipamentos residenciais pertinentes (interruptores e tomadas)
De modo que	eu não os veja em minha residência
Critério de aceitação	os atuadores estarão embutidos e em pleno funcionamento

Fonte – Próprio autor.

Quadro 9 – Estória de usuário EU007 - Configurar um novo atuador.

EU007 - Configurar um novo atuador	
Como um	usuário
Eu quero	configurar um novo atuador
De modo que	eu não precise de um técnico para isso
Critério de aceitação	o atuador pertinente deve estar configurado corretamente

Fonte – Próprio autor.

Quadro 10 – Estória de usuário EU008 - Utilização externa à residência.

EU008 - Utilização externa à residência	
Como um	usuário
Eu quero	utilizar as funcionalidades do sistema estando fora da residência
De modo que	eu possa usufruir de suas funcionalidades de qualquer lugar
Critério de aceitação	atuadores executam suas funções por comandos de usuário de fora da residência de instalação

Fonte – Próprio autor.

Em seguida definimos os módulos desenvolvíveis do sistema de automação, a fim de atender as funcionalidades e especificidades dos requisitos definidos nas estórias. São eles:

a) um atuador de automação residencial capaz de acionar e desacionar circuitos de iluminação da rede elétrica predial, por meio de comandos pela internet, e que seja compacto o suficiente para ser embutido aos interruptores tradicionais;
b) um atuador de automação residencial capaz de acionar e desacionar circuitos de tomadas da rede elétrica predial, por meio de comandos pela internet, e que seja compacto o suficiente para ser embutido a tomadas tradicionais;
c) uma aplicação para dispositivos móveis que sirva como interface para os atuadores, tanto para execução de suas funcionalidades como para suas instalações;
d) um servidor para protocolar a comunicação da aplicação móvel de interface com os usuários, comentada na alínea “c”, por meio da internet, com os atuadores do sistema de automação residencial mencionados nas alíneas “a” e “b”.

Definidos as requisitos do sistema total e seus módulos necessários, prosseguimos para a próxima fase: o Projeto Rápido.

4.2 Projeto Rápido

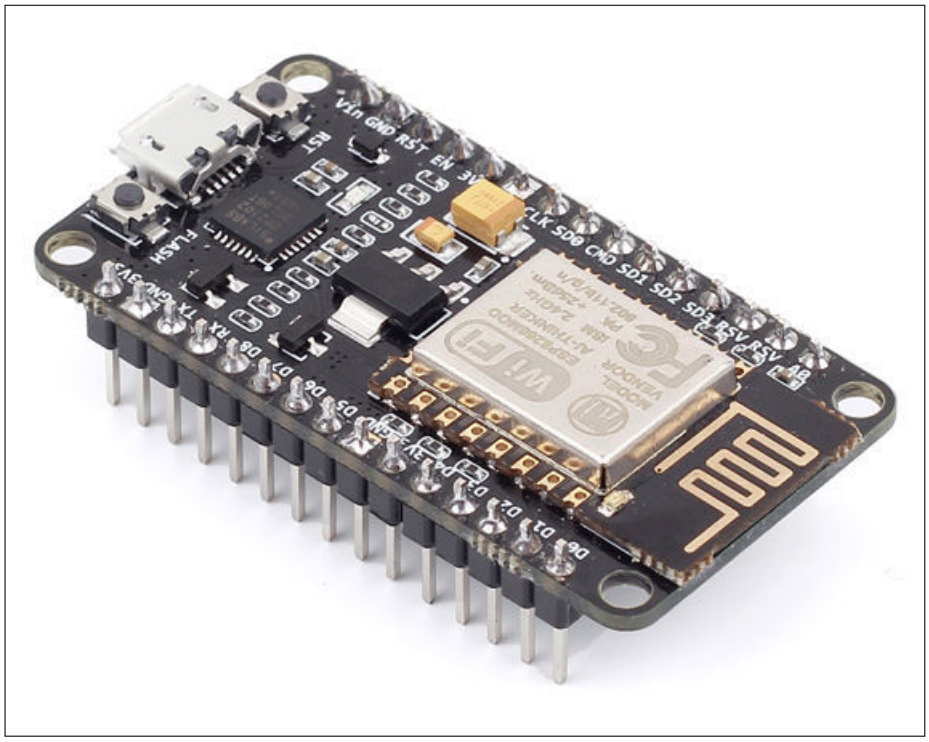
A fase de Projeto Rápido volta-se à avaliação dos requisitos e, graças a isso, à definição do que será implementado e como será implementado nesta primeira iteração. Por opção do autor, todos os requisitos definidos na fase anterior, Seção 4.1, serão implementados nesta iteração do processo definido.

Os *hardwares* escolhidos, bem como a arquitetura e os projetos de software possuem como fim o desenvolvimento ágil da proposta e o que é de mais importante sob a perspectiva do autor: os valores agregados oferecidos aos usuários finais.

4.2.1 Os hardwares

A plataforma de prototipagem escolhida para controlar os atuadores do sistema de automação é a NodeMCU ESP12-E, exibida na Figura 4. Esta foi criada colaborativamente, sob software e *hardware* abertos, com foco para aplicações de Internet das Coisas. Utiliza o microcontrolador ESP8266, da chinesa Espressif, que já possui capacidade de comunicação Wi-Fi IEEE 802.11 b/g/n embutida (NODEMCU, 2017), e por isso escolhida para este projeto.

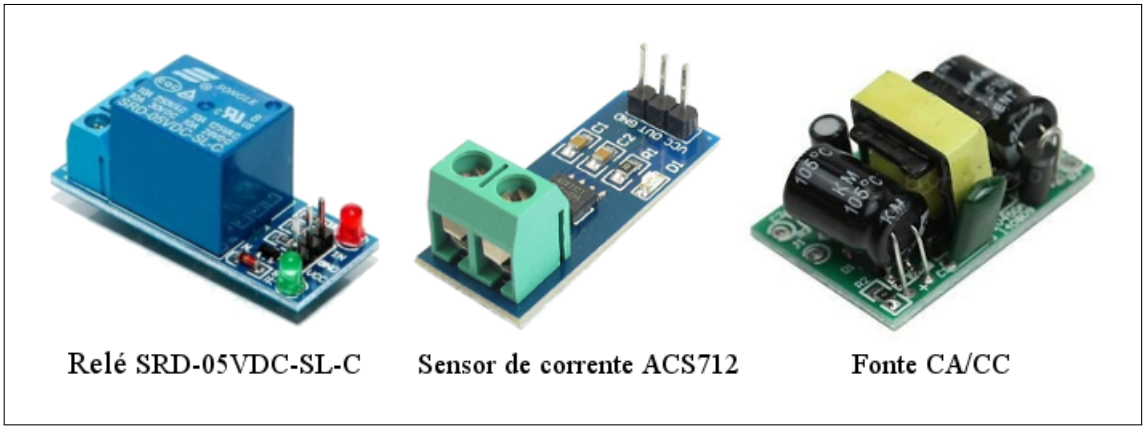
Figura 4 – Plataforma NodeMCU ESP-12E.



Fonte – Instructables (2016).

Além disso, fazem-se necessários componentes eletrônicos agregados à plataforma, a fim desta poder acionar e desacionar os circuitos elétricos, além de obter o estado da corrente elétrica a fim de ser capaz de afirmar se estão ou não acionados e, por fim, alimentar todo o atuador. Para tanto, foram escolhidos o relé Songle SRD-05VDC-SL-C, o sensor de corrente ACS712 e uma fonte universal CA/CC 220/127Vac - 5Vdc @ 0,5A, respectivamente. Estes escolhidos são exibidos na Figura 5 a seguir.

Figura 5 – Componentes eletrônicos dos atuadores.

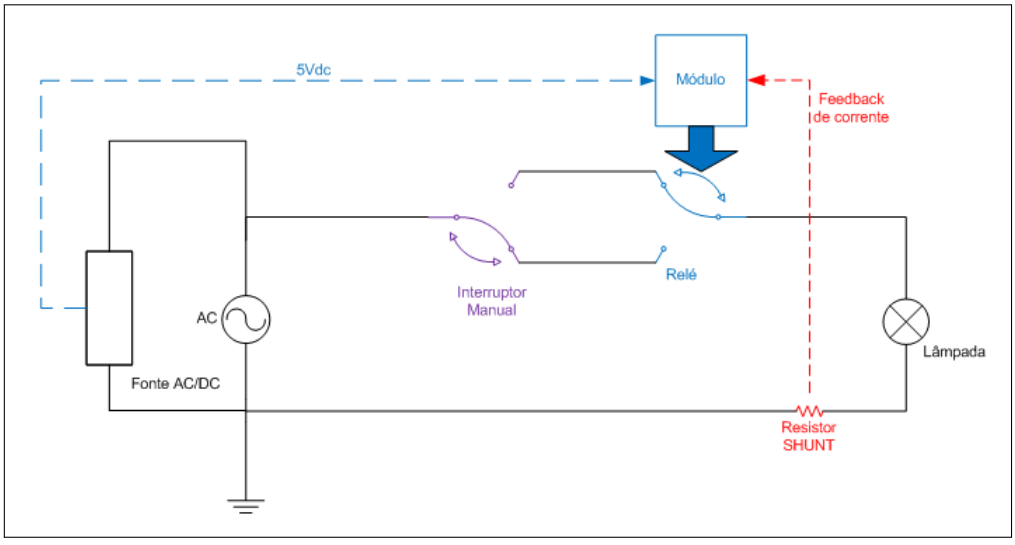


Fonte – Próprio autor.

Importante: os atuadores de automação do sistema, tanto para interruptores de iluminação quanto para tomadas, possuem o mesmo papel de chaveamento de corrente elétrica, diferenciando-se fundamentalmente pelas ligações elétricas em suas instalações. Diante disso, uma mesma implementação de atuador satisfaz tantos os requisitos voltados a interruptores de iluminação quanto os requisitos voltados a tomadas. Portanto, daqui em diante, falar-se-á somente da implementação de um único atuador, devendo este somente ser instalado de modo coerente ao seu contexto em ambiente real, como ilustrado nas Figuras 6 e 7.

O circuito elétrico da Figura 6 a seguir diagrama a instalação do atuador de automação em um interruptor de iluminação residencial.

Figura 6 – Circuito para interruptor de iluminação.

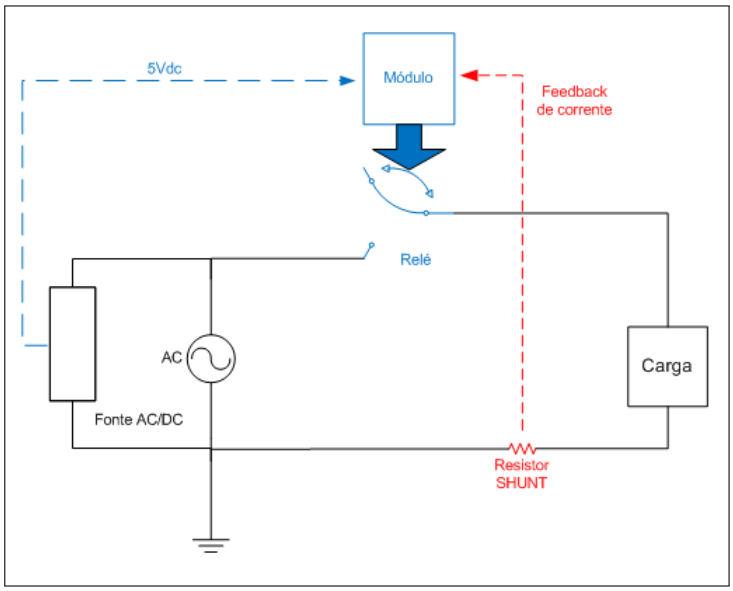


Fonte – Próprio autor.

No diagrama da Figura 6 pode-se notar o transformador *Fonte AC/DC* de corrente alternada em corrente contínua a esquerda da figura, o qual alimenta o *Módulo* atuador acima. Este está ligado em paralelo com o *Interruptor Manual* de iluminação residencial ao centro, para a lâmpada que está a direita. Mais abaixo o *Resistor SHUNT* faz o papel do sensor de corrente ligado ao *Módulo* atuador (SEMIG, 2012).

Semelhante a instalação do atuador para interruptor de iluminação, a Figura 7 a seguir diagrama a instalação do atuador numa tomada residencial.

Figura 7 – Circuito para tomada.



Fonte – Próprio autor.

Retornando aos *hardwares* necessários, é indispensável um servidor para intermediar a comunicação entre os módulos constituintes do sistema: atuadores e *smartphones* de interface com usuários. E para aplicação em ambiente real, é necessário que o usuário possua ao menos um *smartphone* para hospedagem da aplicação de interface com usuários e um roteador de rede Wi-Fi doméstico, ambos ilustrados no diagrama de arquitetura de software da Figura 10.

Na próxima seção será definido o projeto do sistema.

4.2.2 Projeto do sistema

Após análise concisa dos requisitos, do implementável atuador, dos *hardwares* e da aplicação móvel de interface definida chegou-se a problemática para a definição da arquitetura de software: a comunicação entre os sistemas constituintes.

A problemática em torno da arquitetura de software deste projeto refere-se aos problemas pertinentes a sistemas distribuídos em que múltiplos serviços (os atuadores) são oferecidos simultaneamente a múltiplos clientes (o aplicativo de interface). Os principais problemas são:

1. Como a aplicação de interface encontrará o atuador?
2. Como o atuador encontrará a aplicação de interface?
3. Como atuador e aplicação se conectarão?
4. Como atuador e aplicação se comunicarão?

O padrão arquitetural escolhido que melhor atende a problemática, segundo definições de Bass (2007), é o padrão Broker, ver Subseção 3.7.1.

Já a troca de mensagens, intermediada pelo *broker* do padrão arquitetural, utiliza do paradigma Publish/Subscribe para gerenciamento da comunicação, ver Subseção 3.8, do protocolo MQTT – *Message Queue Telemetry Transport* (HUNKELER; TRUONG; STANFORD-CLARK, 2008).

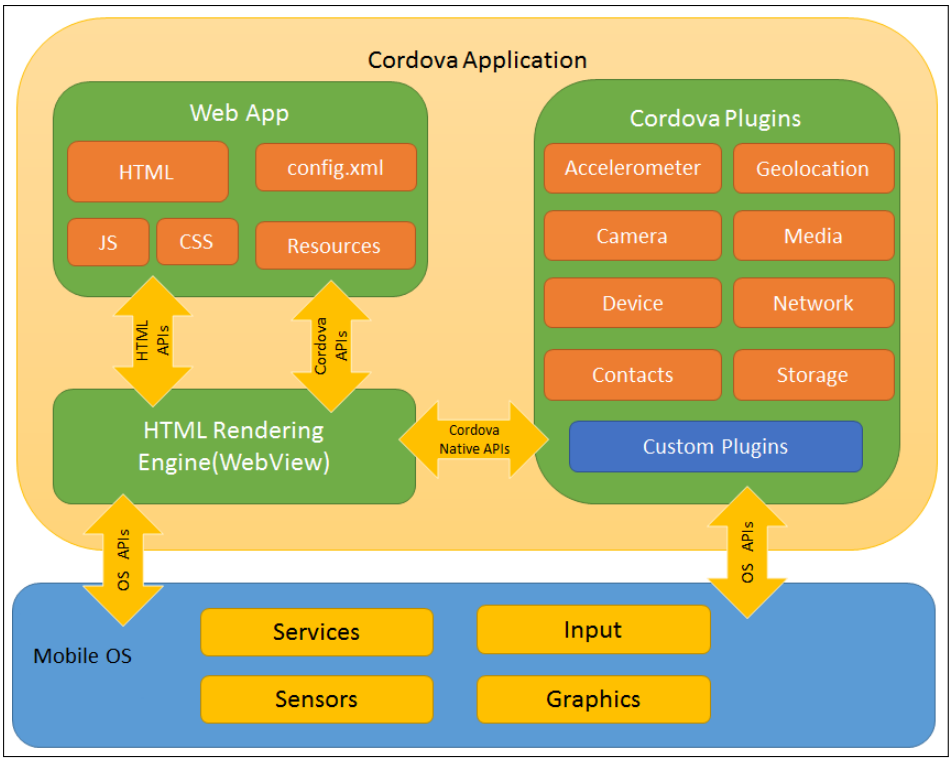
O protocolo MQTT, desenvolvido pelas IBM e Eurotech, é baseado no TCP/IP (FOROUZAN; FEGAN, 2002) e possui uma estrutura organizada de tópicos que permite ao *broker* identificar o destinatário, ou vários, de uma mensagem recebida. Um tópico é uma cadeia de caracteres que descreve um ou mais níveis, em que cada nível é separado por uma barra inclinada, por exemplo: "**país/estado/cidade/bairro**". Além disso, caracteres coringas para níveis permitem a *subscribers* se inscreverem em mais de um tópico por vez. Num exemplo com base no anterior, o coringa "+" no tópico "**país/estado/+/bairro**" permite a um *subscriber* receber qualquer mensagem para o país e estado definidos, desde que a cidade possua o bairro especificado.

Sobre a segurança da informação, o protocolo MQTT utiliza os protocolos de segurança TLS, *Transport Layer Security*, e SSL, *Security Sockets Layer*, que fornecem um canal seguro entre clientes e servidores. Em seus núcleos, utilizam um mecanismo de *handshake* para negociar parâmetros para estabelecer conexão segura e comunicação criptografada entre os *hosts*, a fim de impedir a leitura e modificação das mensagens (HIVEMQ, 2017).

Para a aplicação em dispositivos móveis de interface com os usuários, o meio escolhido para desenvolvimento é o Ionic Framework (IONIC, 2017). O Ionic Framework é um SDK (*Software Development Kit*) de código aberto que permite a desenvolvedores criarem aplicativos móveis de alta qualidade, usando linguagens WEB populares (IONIC, 2017), como

HTML5, CSS3 e Javascript 7. O Ionic acrescenta uma camada superior a outro *framework* chamado Cordova (CORDOVA, 2017), organização ilustrada na Figura 8, a qual oferece componentes de interface para as plataformas móveis compatíveis que imitam o comportamento de componentes de interface nativos (IONIC, 2017).

Figura 8 – Visão abstrata da arquitetura de aplicativos Cordova.



Fonte – Cordova (2017).

Sobre os *frameworks* Ionic e Cordova, a Figura 8 anterior ilustra a arquitetura de aplicativos Cordova em que é possível verificar o módulo onde o Ionic acrescenta uma camada superior de interface gráfica, o módulo “Web App”.

Ainda sobre a Figura 8, o Cordova Framework encapsula um *web app* (*website* que se comporta como aplicativo móvel) numa *web view* nativa da plataforma e oferece suporte a serviços nativos por meio de *plugins*, como acesso a câmera, internet e dados de sensores.

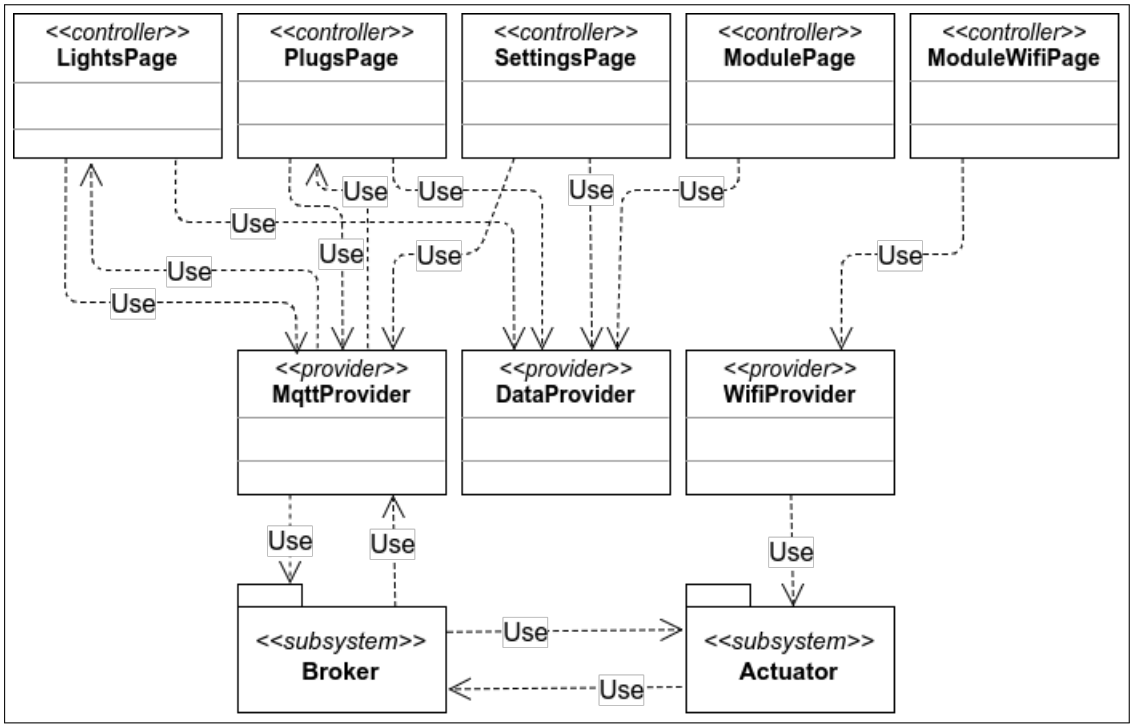
Ao desenvolvimento de aplicação que combina elementos nativos da plataforma alvo com meios não nativos, como, por exemplo, utilizando os *frameworks* Ionic e Cordova, denomina-se desenvolvimento híbrido, fundamentado na Subseção 3.4.1.

No tocante ao servidor para execução do *broker* de intermediação da comunicação entre os sistemas, a hospedagem escolhida deve utilizar o conceito de Computação em Nuvem, explicado na Subseção 3.5.

Finalmente, no que se refere ao projeto do sistema, a seguir se encontra uma visão modular da arquitetura de software, Figura 9, definida para o sistema de automação deste projeto.

Uma visão modular de arquitetura de software destaca as principais unidades de implementação de um sistema. A escolha das unidades e a visão desse conjunto facilita determinar como mudanças em uma parte do sistema afetarão outras partes e, por conseguinte, averiguar a modificabilidade, portabilidade e capacidade de reutilização (CLEMENTS *et al.*, 2003).

Figura 9 – Visão modular, tipo uso, da arquitetura de software.



Fonte – Próprio autor.

A visão modular da Figura 9 é do tipo uso que permite identificar relações de dependência funcional entre os módulos (CLEMENTS *et al.*, 2003), indicadas pelas relações "*Use*". Esses módulos foram estereotipados como *controller*, *provider* ou *subsystem*, os quais significam:

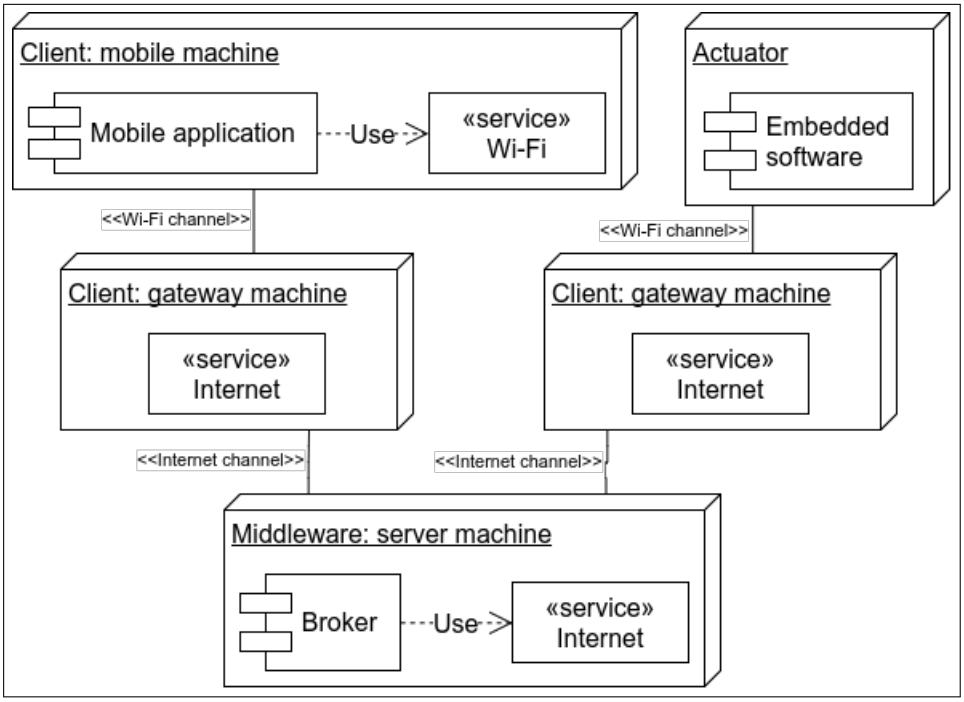
a) *Controller*: classe controladora de página do *web app* do framework Cordova. Funciona semelhante às Activitys da plataforma Android (DEVELOPERS, 2018);

b) *Provider*: classe provedora de serviço de dados, de qualquer tipo, à aplicação (MORONY, 2018);

c) *Subsystem*: é um subsistema que compõe a arquitetura.

Em seguida, na Figura 10, definimos mais uma visão voltada à abstração dos recursos necessários para implantação do sistema em ambiente real, a visão de alocação tipo implantação (CLEMENTS *et al.*, 2003).

Figura 10 – Visão de alocação, tipo implantação, da arquitetura de software.



Fonte – Próprio autor.

Uma visão de alocação de arquitetura de software foca na relação entre a arquitetura e o ambiente de implantação, haja vista um software não existir por conta própria (CLEMENTS *et al.*, 2003).

A visão de alocação da Figura 10 é do tipo implantação que busca definir quais os recursos de ambiente necessários para implantação em ambiente real do domínio do sistema, o que pode incluir tanto recursos físicos, como memórias, quanto recursos lógicos, como serviços de banco de dados (CLEMENTS *et al.*, 2003). Nessa visão da Figura 10 são definidos os ambientes, os componentes do sistema e seus principais recursos necessários, que são:

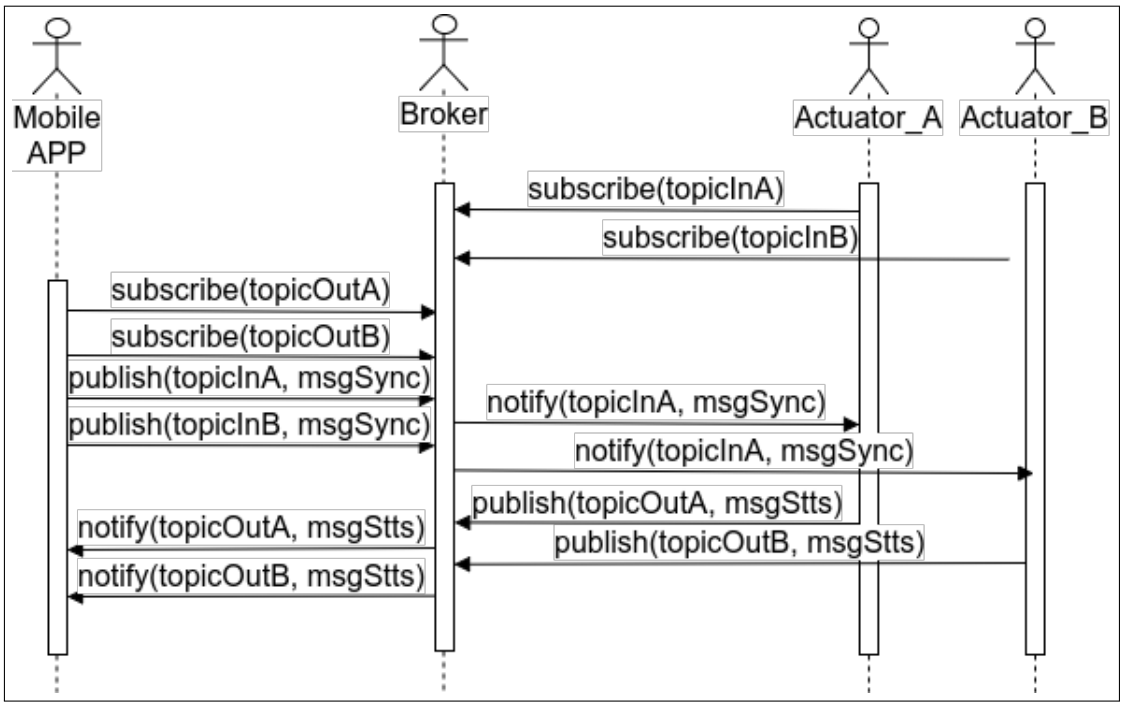
a) ambiente *Client: mobile machine* é o dispositivo móvel (*smartphone*) necessário para hospedagem do componente aplicação móvel de interface com usuários (*Mobile application*) o qual necessita do recurso: serviço Wi-Fi do dispositivo;
b) ambientes *Client: gateway machine* são os roteadores de internet residenciais que necessitam do recurso: acesso a internet;

c) ambiente *Middleware: server machine* é o servidor que executa o componente *Broker* que intermedia a comunicação entre a aplicação móvel e o atuador, e que necessita do recurso: acesso à internet;

d) ambiente *Actuator* é o atuador do sistema de automação que executa o componente *Embedded software* e não necessita explicitamente de recursos.

Resta saber como dar-se-á a comunicação entre os sistemas distribuídos do projeto deste trabalho, a qual é abstraída por meio dos diagramas de sequência das Figuras 11 e 12 adiante.

Figura 11 – Diagrama: início da comunicação entre os módulos do sistema.



Fonte – Próprio autor.

No diagrama de sequência da Figura 11 vemos uma abstração sobre a comunicação entre dois atuadores, *Actuator_A* e *Actuator_B*, o *Broker* e a aplicação móvel de interface, *Mobile APP*, a qual é explicada sob a ordem das mensagens:

1. inicialmente um atuador se inscreve em seu tópico para recebimento de instruções. No diagrama isso ocorre para ambos os atuadores: *Actuator_A* e *Actuator_B*;
2. ao iniciar a aplicação *Mobile APP*, esta se inscreve nos tópicos para recebimento de mensagens dos seus atuadores. Em seguida, envia instruções a fim de receber os estados atuais de seus atuadores para sincronia de informações;
3. o *Broker*, ao receber as mensagens de instruções, envia notificações aos

destinatários pertinentes, identificados pelos tópicos;

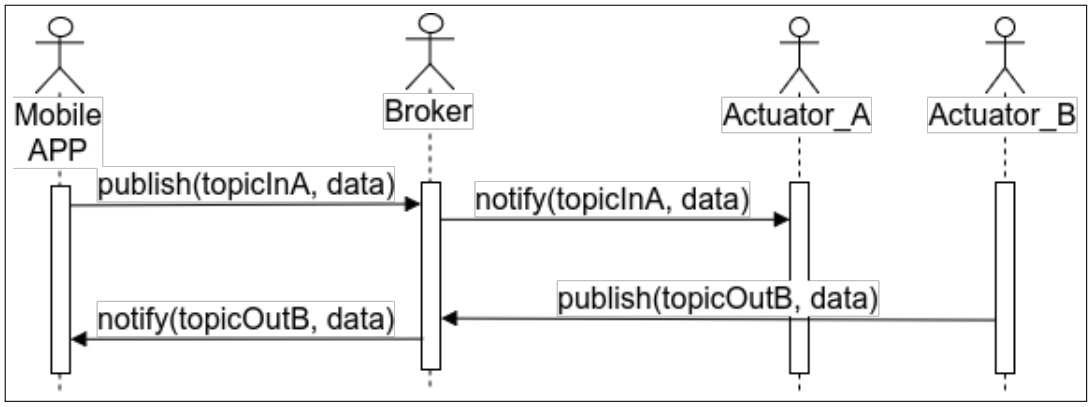
4. os atuadores, ao serem notificados sobre a instrução para sincronia, enviam seus estados atuais à aplicação de interface *Mobile APP*.

Na troca de mensagens ilustrada no diagrama da Figura 11 seus parâmetros significam o seguinte:

a) *topicInA* e *topicInB*: tópicos para recebimento de mensagens pelos atuadores *Actuator_A* e *Actuator_B*, respectivamente;
b) *topicOutA* e *topicOutB*: tópicos para envio de mensagens dos atuadores *Actuator_A* e *Actuator_B*, respectivamente, para a aplicação móvel *Mobile APP*;
c) *msgSync*: mensagem de instrução da aplicação móvel *Mobile APP* para recebimento do estado atual do atuador para sincronia de informações;
d) *msgStts*: mensagem contendo o estado atual do atuador para sincronia com a aplicação móvel.

Por fim, a comunicação entre a aplicação móvel e os atuadores, com o objetivo de executar instruções e receber atualizações, acontece conforme o diagrama de sequência da Figura 12 em seguida.

Figura 12 – Diagrama: fundamental comunicação entre os módulos do sistema.



Fonte – Próprio autor.

O diagrama da Figura 12 exemplifica o envio de uma instrução para uma função do *Actuator_A*, por parte do cliente em *Mobile APP*, e em seguida o envio de uma atualização de estado por parte do *Actuator_B* ao cliente em *Mobile APP*.

Finalmente definido o projeto de sistema, partimos para os projetos de software.

4.2.3 Projetos de software

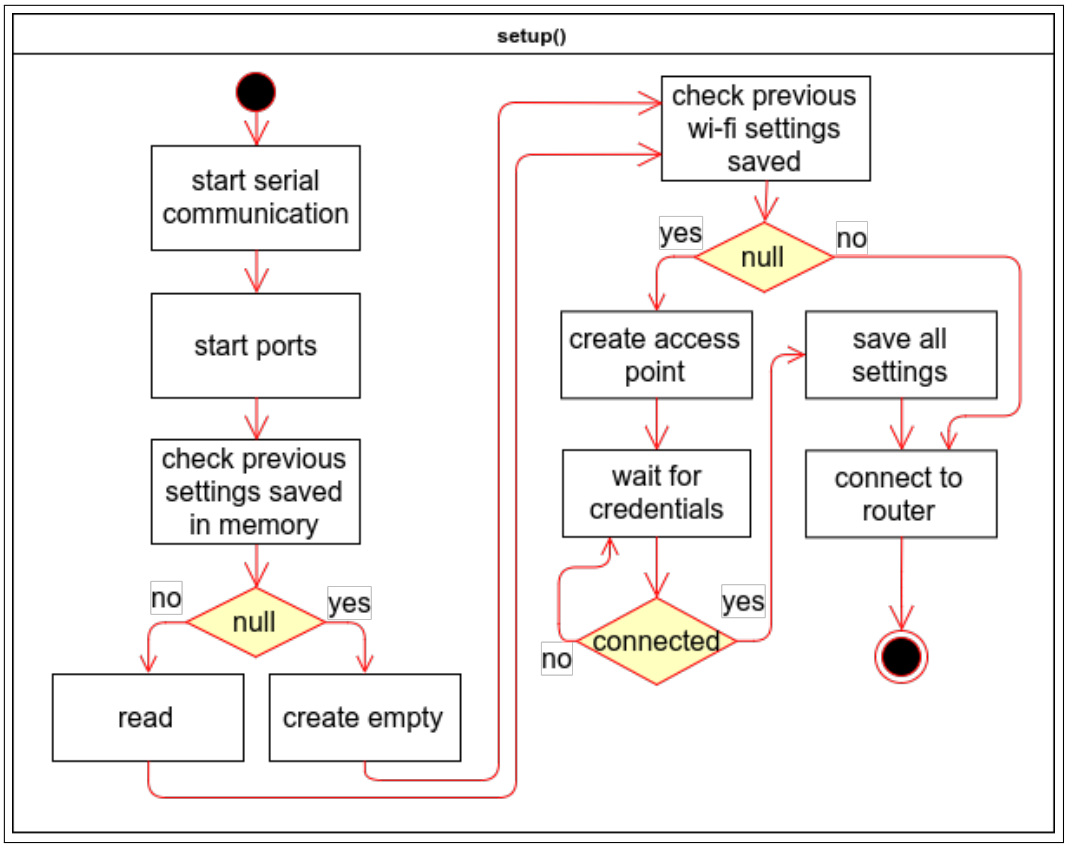
Em projetos de software iniciamos pelo projeto do software embarcado para o atuador único de automação para interruptores de iluminação e tomadas.

O desenvolvimento e a compilação do software embarcado utiliza a plataforma Arduino (ARDUINO, 2018) a qual estrutura seus projetos em duas funções principais, que são:

a) função *setup()*: é chamada logo no início da execução do software embarcado e somente uma vez, sempre que for iniciada a controladora;
b) função *loop()*: é a função de rotina do software embarcado. É executada logo após a função *setup()* e fica em *loop* infinito enquanto a controladora estiver ligada.

O diagrama de atividades (DUMAS; HOFSTEDE, 2001) da Figura 13 na sequência deve-se à natureza do software procedural e modela, abstraidamente, a função de início do software embarcado.

Figura 13 – Diagrama: fluxo de atividades da função *setup()* do software atuador.



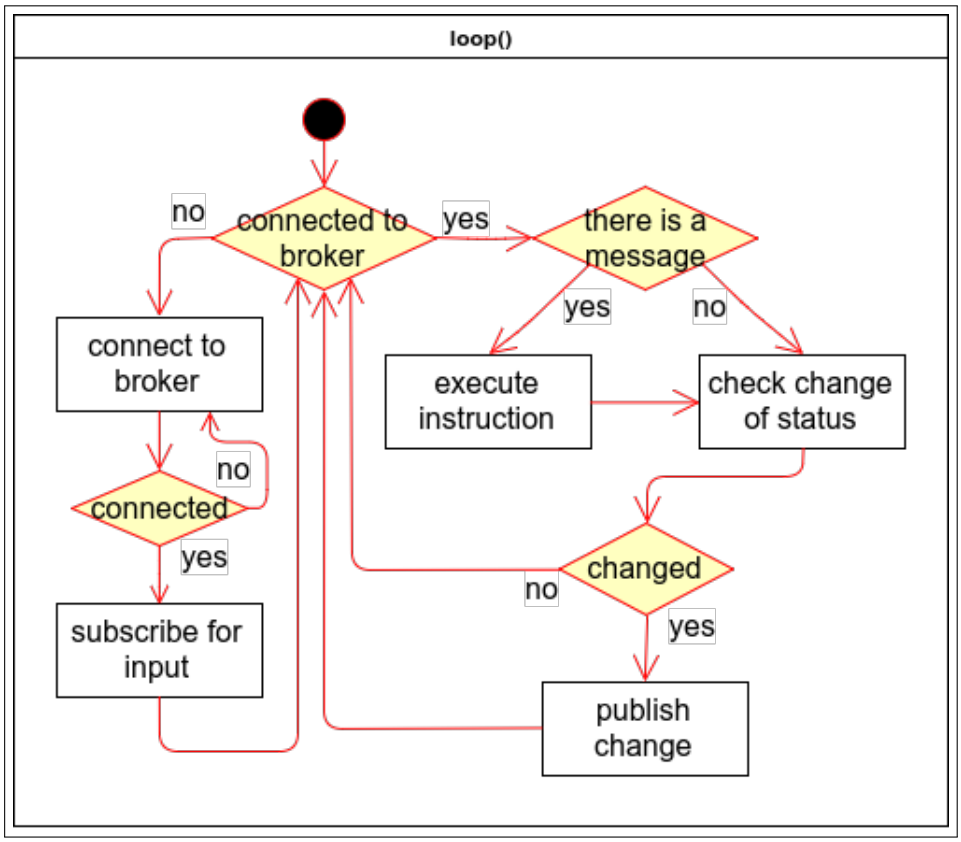
Fonte – Próprio autor.

A função *setup()* da Figura 13 possui como atividades:

1. iniciar uma comunicação serial utilizada para depuração, em *start serial communication*;
2. iniciar as portas utilizadas para o controle do relé e leitura do sensor de corrente, da Figura 5, em *start ports*;
3. verificar se há configurações prévias para conexão ao *broker*. Se houver, faz a leitura, senão, cria configurações vazias;
4. verificar se há configurações prévias para se conectar ao roteador Wi-Fi doméstico, em *check previous wi-fi settings saved*. Se sim, conecta-se ao roteador, senão, cria um ponto de acesso e aguarda pelas credenciais, por meio de um *webserver* criado, em *wait for credentials*. Ao fim, salva as credenciais recebidas na memória e conecta-se ao roteador.

O próximo diagrama de atividades, na Figura 14 a seguir, destaca os papéis da função de rotina do software embarcado.

Figura 14 – Diagrama: fluxo de atividades da função *loop()* do software do atuador.



Fonte – Próprio autor.

A função *loop()* da Figura 14 realiza os seguintes tratamentos:

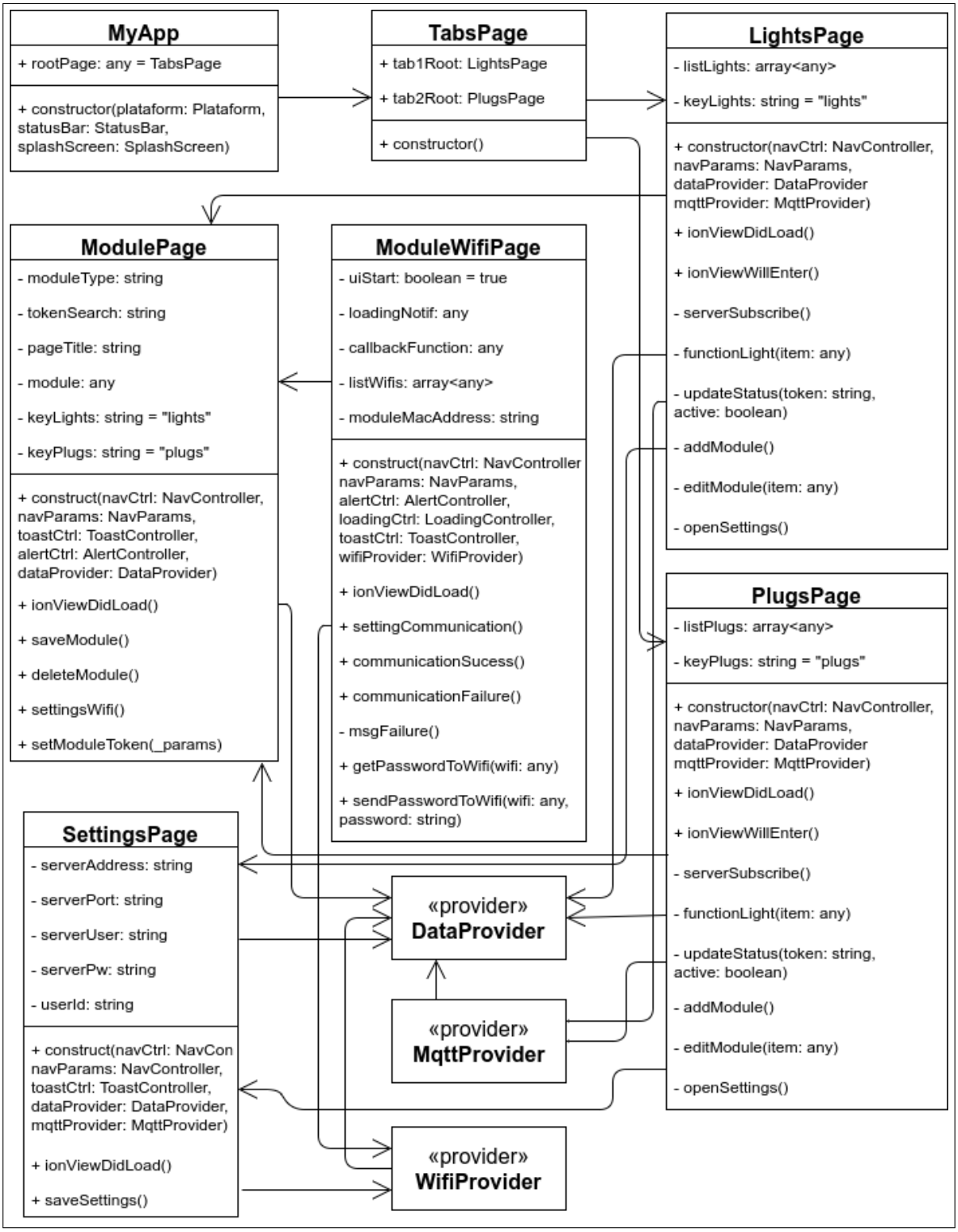
a) verifica se há conexão com o *broker*, em *connected to broker*: se não, conecta-se ao *broker* e se inscreve no tópico necessário para a comunicação, se sim, continua seus procedimentos na condição *there is a message*;
b) caso esteja conectado ao *broker*, a função *loop()* verifica se há mensagem recebida, caso haja, executa a instrução da mensagem, em *execute instruction*, caso não, verifica se há alterações no estado atual do atuador;
c) caso haja alteração no estado atual do atuador, envia mensagem para a aplicação móvel de interface em *publish change*, caso não, volta ao início da rotina em *connected to broker*.

Em *execute instruction* no diagrama da Figura 14, as instruções recebidas podem ser para dois fins: sincronia e função.

A instrução de sincronia é recebida no início da comunicação da aplicação móvel que possui acesso ao atuador, a fim de sincronizar as informações do atuador com a aplicação, ver diagrama da Figura 11. Já a instrução de função é a responsável por alterar o estado do relé conforme necessário (devido uma possível ligação *three-way* em instalação em interruptor de iluminação) para ligar ou desligar correntes de energia, ver diagrama da Figura 12.

Agora, parte-se para o projeto do software da aplicação móvel, o qual inicia pela definição das classes constituintes, por meio dos diagramas das Figura 15 e 16.

Figura 15 – Diagrama: classes da aplicação móvel de interface com os usuários.



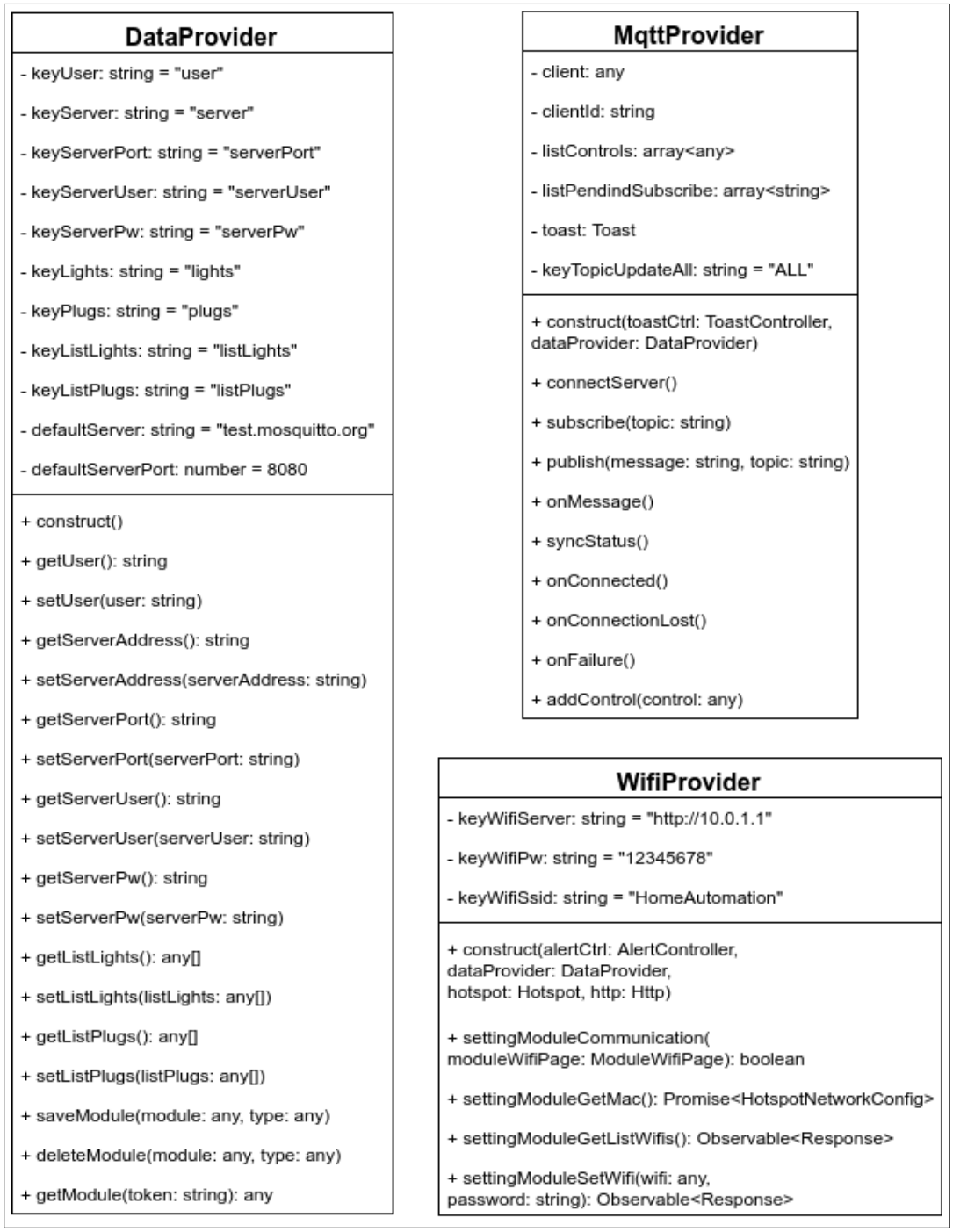
Fonte – Próprio autor.

Diante do diagrama de classes (PURCHASE *et al.*, 2001) da Figura 15, devemos esclarecer os seguintes aspectos:

a) a princípio, deve-se destacar que a maioria dos relacionamentos são unidirecionais, com exceção dos relacionamentos entre as classes *LightsPage* e *PlugsPage* com a classe *MqttProvider*, que é bidirecional 1 para 1 devido a ambas as classes necessitarem estarem cientes do relacionamento;
b) a classe *MyApp* é a classe principal da aplicação. É gerada automaticamente pelo *framework* Ionic, assim como a classe *TabsPage*, e é nela que é definida a tela inicial da aplicação por meio do atributo *rootPage*;
c) a classe *TabsPage* é necessária devido a interface gráfica inicial escolhida pelo autor: interface em abas (IONIC, 2018). As controladoras de suas abas são definidas por meio de seus atributos *tab1Root* e *tab2Root*;
d) as classes *LightsPage* e *PlugsPage* são semelhantes devido à pertencerem a classe *TabsPage* para exercer os papéis de controladoras das abas. São as classes que comportarão as listas de atuadores a fim de estarem disponíveis para suas utilizações;
e) já as classes *ModulePage* e *ModuleWifiPage* são as controladoras das duas páginas necessárias para instalação e edição de atuador. A primeira, para salvar as informações pertinentes e a segunda, para configurar o estabelecimento da primeira comunicação entre a aplicação e o atuador, a fim do reconhecimento;
f) a classe *SettingsPage* é a controladora da página necessária para definição de configurações comuns à aplicações com interface à rede, como determinar o endereço do servidor do *broker*, a porta e as credenciais;
g) deve-se destacar as três classes com o estereótipo *«provider»*, por fazerem parte de uma espécie de pacote especial, apesar de um projeto do *framework* Ionic não ser organizado de tal maneira, mas, para fins didáticos, será aludido assim, pois são classes de interface com serviços nativos da plataforma móvel, como banco de dados, internet e Wi-Fi. Essas classes serão abstraídas no diagrama da Figura 16.

Como resta definir, as classes estereotipadas com *«provider»* do diagrama da Figura 15 são abstraídas no esboço de diagrama da Figura 16 a seguir.

Figura 16 – Classes *«provider»* da aplicação móvel.



Fonte – Próprio autor.

Como os relacionamentos das classes *«provider»* já foram ilustrados no diagrama da Figura 15, foram suprimidos na Figura 16. Além disso, dissertando sobre essas classes, tem-se:

a) a classe *DataProvider* é a classe de serviço responsável pela persistência de todos os dados necessários para a aplicação na plataforma móvel;
b) a classe *MqttProvider* é responsável pela interface de comunicação com o *broker*.

É por meio dela que é possível inscrever-se num tópico, método *subscribe(topic: string)*, e realizar publicações, método *publish(message: string, topic: string)*, além de receber e encaminhar as mensagens recebidas por meio do método *onMessage()*;

c) a classe *WifiProvider* é a classe responsável pela interface com o recurso Wi-Fi nativo da plataforma. É por meio dela que é possível comunicar-se com o ponto de acesso do atuador no momento de sua instalação, gerenciando o Wi-Fi do dispositivo móvel, por meio de uma comunicação inicial com um *webserver* no atuador, por HTTP (FIELDING *et al.*, 1999), para troca de credenciais.

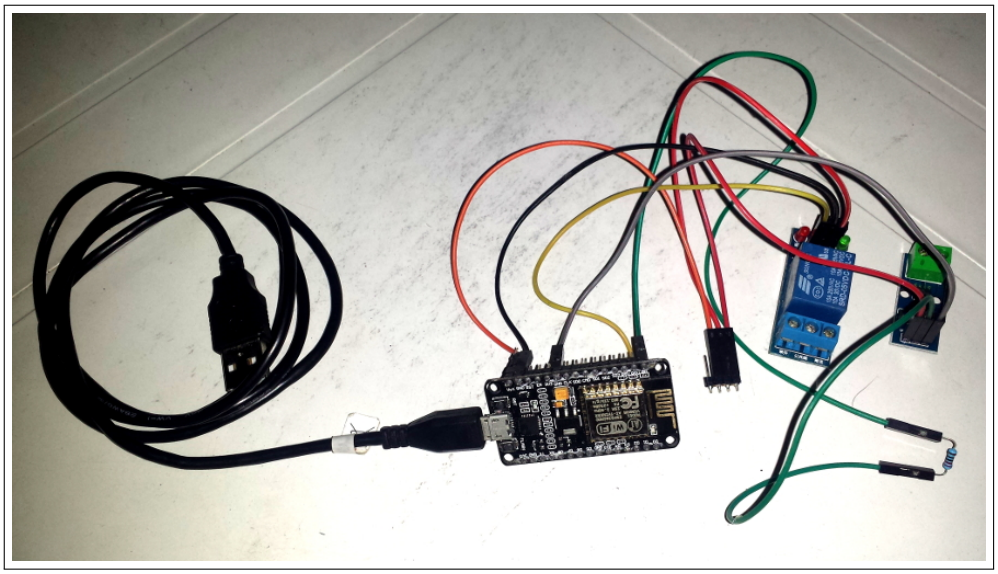
Tendo sido definidos os requisitos de sistema, o *hardware* necessário ao projeto, a arquitetura e os projetos de software pertinentes, parte-se então para a fase Construção do sistema, segundo o processo evolucionário.

4.3 Construção

Passada a fase de Projeto Rápido do sistema, a fase Construção é, enfim, a fase de implementação do sistema, fase esta que põe em prática definições da fase anterior.

A implementação do software embarcado do atuador de automação ocorre após a montagem do protótipo de *hardware*, que consiste na conexão entre os componentes eletrônicos ilustrados na Figura 5 com a plataforma da Figura 4, conforme ilustra a Figura 17.

Figura 17 – Protótipo embarcado do atuador de automação.



Fonte – Próprio autor.

Há ressalva para a não utilização do componente de alimentação fonte universal CA/CC 220/127Vac - 5Vdc @ 0,5A da Figura 5, na construção desse protótipo ilustrado na Figura 17. Isso ocorre devido ao risco de choque elétrico, ao caráter exploratório da primeira iteração no Paradigma da Prototipação e à sua utilização ser somente para alimentação do atuador. Diante disso, para alimentação e gravação da plataforma é utilizado um cabo USB-MicroUSB Samsung ligado a uma fonte 5 *volts* da mesma marca.

As ligações entre a plataforma e os componentes foram realizadas por meio dos pinos definidos no Quadros 11 e 12 e utilizando cabos Dupont Fêmea-Macho de secção 24 AWG.

Quadro 11 – Ligação entre os pinos da NodeMCU e do relé.

NodeMCU ESP12-E	**Relé Songle**	**Descrição**
VIN	VCC	Alimentação do relé.
GND	GND	Aterramento do relé.
D1	IN	Sinal para relé.

Fonte – Próprio autor.

Quadro 12 – Ligação entre os pinos da NodeMCU e do sensor de corrente.

NodeMCU ESP12-E	**Sensor de Corrente**	**Descrição**
VIN	VCC	Alimentação do sensor.
GND	GND	Aterramento do sensor.
A0	OUT	Sinal para a plataforma.

Fonte – Próprio autor.

Deve-se destacar que na ligação entre a plataforma NodeMCU e o sensor de corrente, no Quadro 12, entre os pinos A0 e OUT para o sinal para a plataforma, é necessário reduzir a tensão devido o sensor trabalhar com 5 *volts* enquanto a plataforma trabalha com 3.3 *volts*.

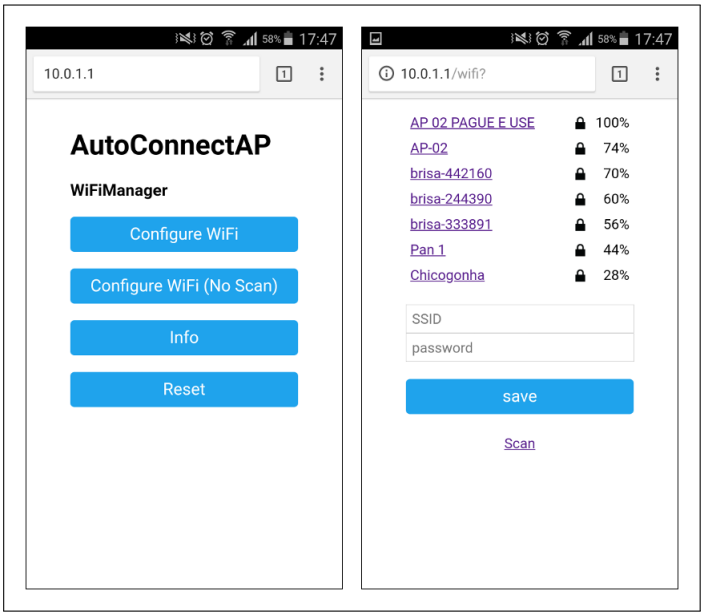
Para que seja possível a implementação e gravação do software na plataforma, é necessária a configuração do ambiente, e mesmo a plataforma escolhida oferecendo a possibilidade nativa de programação por meio da linguagem Lua (PUC-RIO, 2018), o ambiente escolhido para escrita e gravação do software deste trabalho é o da plataforma Arduino, como comentado na Subseção 4.2.3, por oferecer suporte para a NodeMCU ESP12-E, as linguagens C/C++ e diversas bibliotecas disponíveis. Diante disso, a Arduino IDE usada neste trabalho é a de versão 1.8.5 e o pacote de suporte à plataforma NodeMCU ESP12-E é o de versão 2.3.0. A

princípio, caso este pacote não esteja disponível na IDE Arduino, é possível instalá-lo por meio do gerenciador de placas, no menu Ferramentas, Placas e Gerenciador de Placas.

A implementação do software do atuador utiliza seis bibliotecas, três nativas de suporte à plataforma Arduino e três inseridas, uma para gerenciar o acesso amigável da plataforma a redes domésticas Wi-Fi, a segunda para gerenciar comunicação sob o protocolo MQTT e a última para gerenciar persistência de dados na memória, respectivamente a WifiManager (TZAPU, 2017), a PubSubClient (KNOLLEARY, 2017) e a File System (ESP8266, 2017). O software foi escrito em 250 linhas de códigos, possui seu fluxo lógico diagramado nas Figuras 13 e 14, e encontra-se no repositório online do autor Magalhães (2018a).

Para implementação da atividade *wait for credentials* do diagrama de atividades da Figura 13, a qual volta-se ao recebimento do identificador da rede Wi-Fi e da senha para conexão do atuador a rede doméstica, além do recebimento do endereço do servidor MQTT, foi necessário incrementar a biblioteca WifiManager (TZAPU, 2017) a qual reserva-se a aguardar credenciais por meio de uma interface amigável em *webserver* e recebê-las via HTTP por método GET (FIELDING *et al.*, 1999), conforme mostra a Figura 18. Isso foi possível devido à licença permissiva MIT (MIT, 2018) da biblioteca.

Figura 18 – Telas da interface amigável da biblioteca WifiManager.



Fonte – Próprio autor.

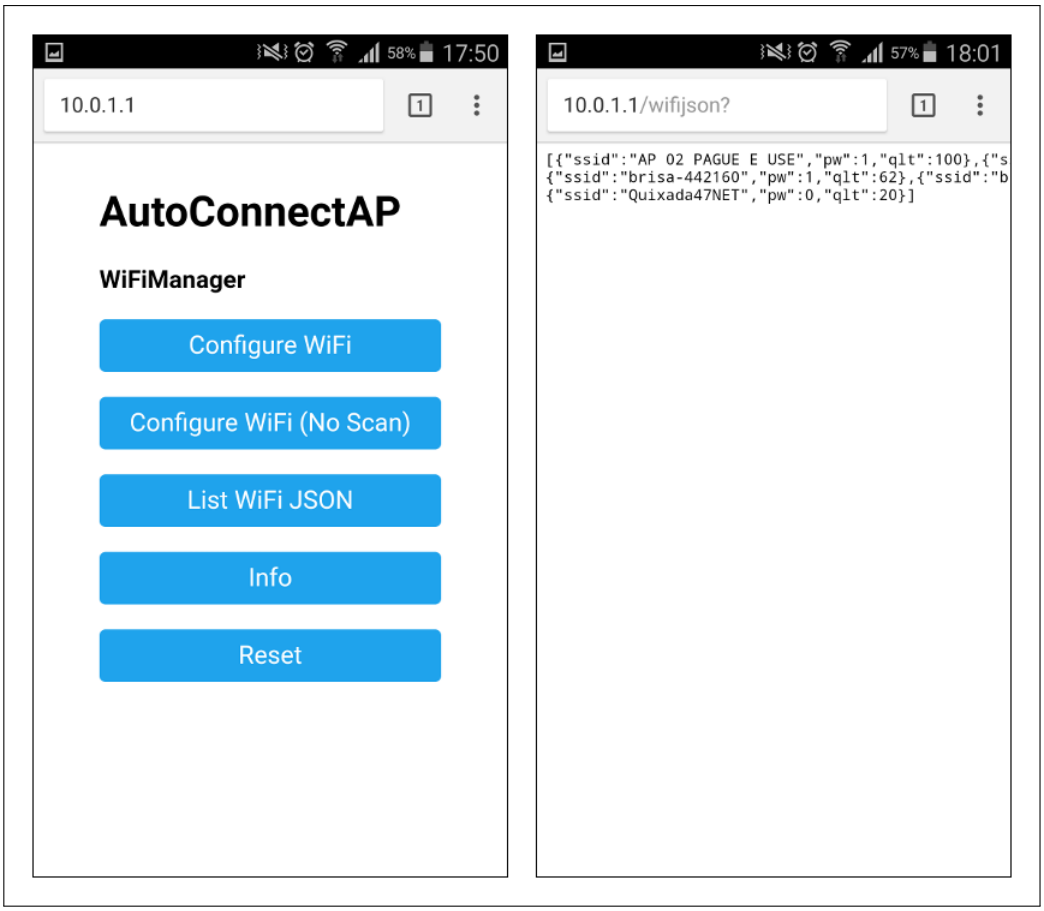
Sobre a Figura 18 anterior, as duas telas referem-se ao gerenciamento amigável da

conexão do atuador a uma rede Wi-Fi doméstica pela biblioteca WifiManager. A tela à esquerda refere-se à página inicial do *webserver*, que ao clicar no primeiro botão, de título *Configure WiFi*, é redirecionada para a tela à direita que lista as redes alcançáveis à plataforma NodeMCU, a fim de permitir que o usuário escolha qual rede que a plataforma deve se conectar.

O incremento consisti da disponibilização da lista de redes Wi-Fi alcançáveis pela plataforma por meio de um objeto lista JSON (JAVASCRIPT, 2018), mediante requisição GET, para a leitura por parte da aplicação móvel para que toda a configuração ocorra por meio desta, e pode ser conferido no repositório online do autor Magalhães (2018c).

A Figura 19 a seguir ilustra o incremento.

Figura 19 – Telas da interface da biblioteca WifiManager incrementada.



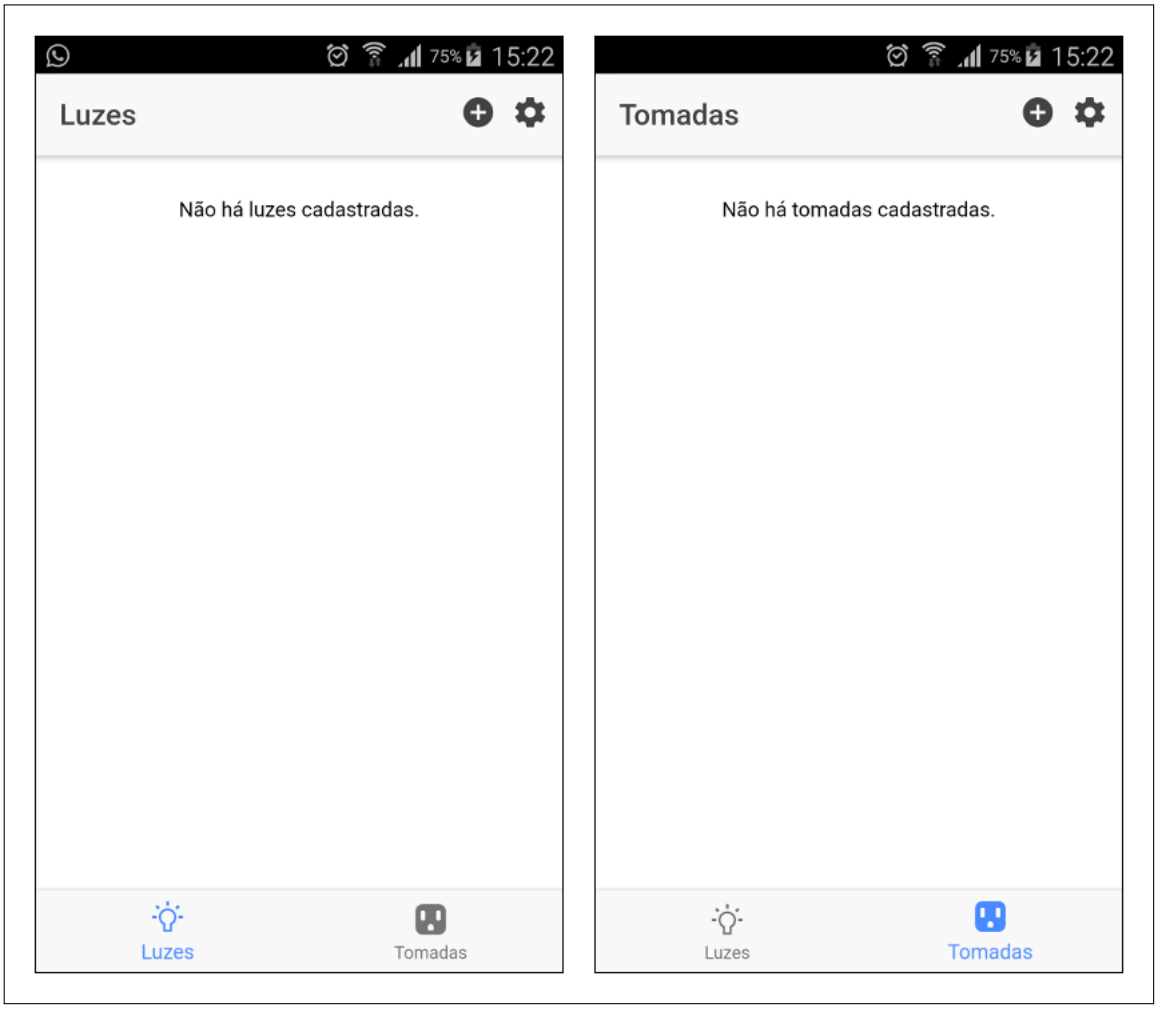
Fonte – Próprio autor.

Na tela à esquerda, da Figura 19, foi acrescentado o terceiro botão, de título *List WiFi JSON*, que ao ser "clicado" resulta na tela à direita, com o objeto JSON lista de objetos com as informações sobre as redes disponíveis.

Sobre a implementação da aplicação móvel de interface para usuários, parte-se do

projeto de software sob as classes controladoras diagramadas nas Figuras 15 e 16 para construção das telas visuais e programação. Essas podem ser conferidas nas Figuras 20 a 24 adiante.

Figura 20 – Telas da aplicação móvel: principais.

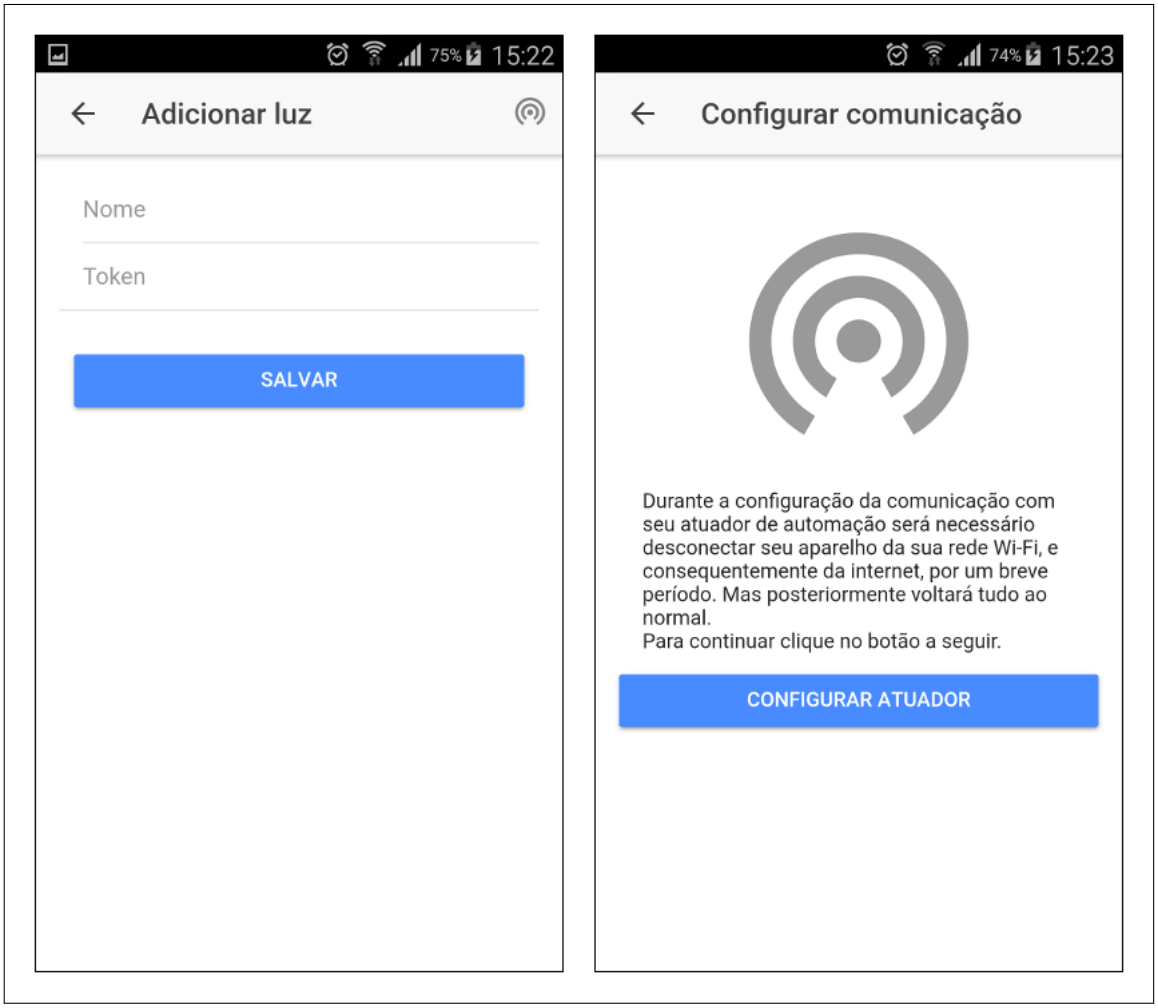


Fonte – Próprio autor.

As telas da interface da aplicação móvel da Figura 20 são as telas principais. Organizadas em abas, são as responsáveis por listar os atuadores disponíveis para controle de interruptores de luz, como ilustra a Figura 23, e tomadas, respectivamente a tela à esquerda e à direita. Suas classes controladoras são as *LightsPage* e *PlugsPage* do diagrama de classes da Figura 15. A tela inicial é a de título "Luzes", à esquerda.

Ao clicar no ícone "**+**" à direita na barra de navegação superior, uma tela responsável por configurar um novo atuador é aberta, para luzes ou tomadas, dependendo da aba que estiver, conforme ilustra a tela à esquerda da Figura 21. E ao clicar no ícone *affordance* engrenagem, a tela da direita da Figura 24 é aberta para configurações da comunicação com o servidor.

Figura 21 – Telas da aplicação móvel: configuração de atuador.



Fonte – Próprio autor.

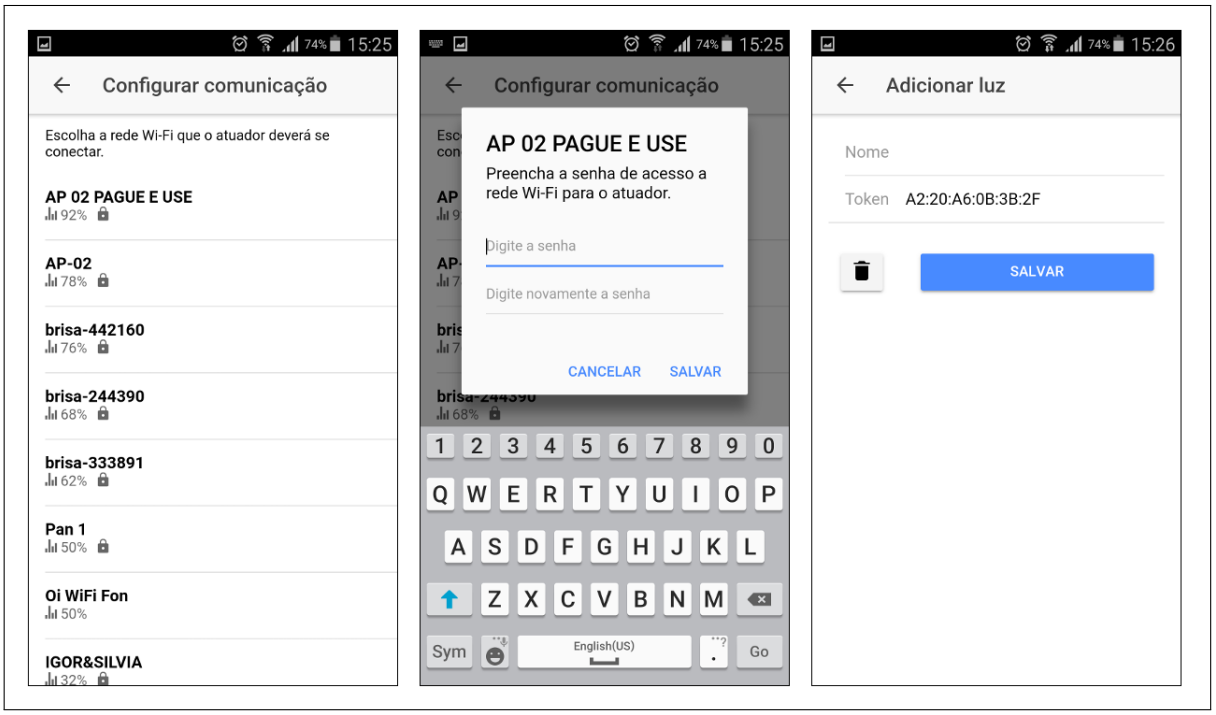
Sobre a Figura 21 anterior, a tela à esquerda é responsável por adicionar um novo atuador. Os campos “Nome” e “Token” são responsáveis pela identificação do atuador, o primeiro para o usuário, conforme o nome “Sala” na Figura 23, e o segundo para o sistema, sendo este utilizado no tópico para troca de mensagens no paradigma Publish/Subscribe do protocolo MQTT comentado na Subseção 4.2.2.

Porém, o “Token” não necessita ser inserido pelo usuário, pois numa instalação do atuador, a aplicação recebe esta informação e a preenche no formulário, como exemplifica a Figura 22 em seguida, necessitando o usuário apenas definir um nome para identificação do atuador. As suas classes controladoras são as classes *ModulePage* e *ModuleWifiPage*, respectivamente, do diagrama de classes da Figura 15.

Já a tela à direita, da Figura 21, é a tela aberta após TAP no ícone *affordance* de sinal Wi-Fi à direita na barra de navegação, que fica em destaque sob efeito *blink*, para estabelecer

o primeiro reconhecimento e comunicação com o atuador. Isto ocorre ao "clicar" no botão "CONFIGURAR ATUADOR" e, ao fim, surge a nova tela à esquerda da Figura 22 abaixo.

Figura 22 – Telas da aplicação móvel: instalando novo atuador.



Fonte – Próprio autor.

Na Figura 22, a tela à esquerda lista as redes Wi-Fi domésticas identificadas pelo atuador e capazes deste se conectar. Além disso, a lista informa a potência relativa do sinal e se a rede é ou não protegida por senha. Ao clicar numa rede, por exemplo, a rede "AP 02 PAGUE E USE", a aplicação solicita a senha para que o atuador possa conectar-se à mesma, conforme ilustra a tela ao centro da Figura. Após salva a senha, esta é enviada ao atuador e a aplicação recebe seu endereço MAC único utilizado para reconhecimento, o qual é automaticamente inserido no formulário para adição de novo atuador, conforme a tela à direita da Figura 22.

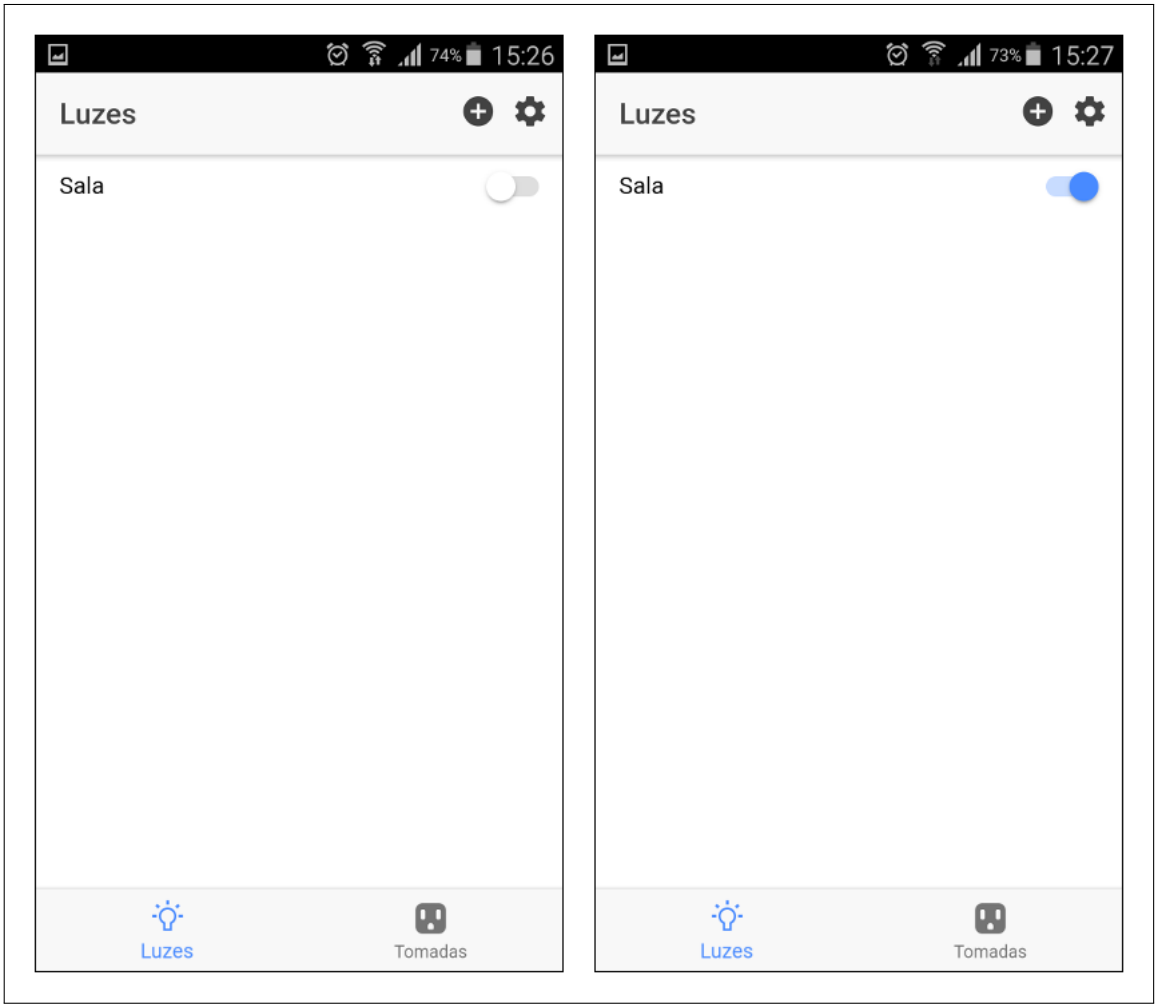
Feita a configuração da comunicação com o atuador, basta o usuário inserir um nome para identificação do atuador no campo "Nome" e salvar.

As classes controladoras da primeira e segunda telas da Figura 22, bem como da terceira são, respectivamente, as classes *ModuleWifiPage* e *ModulePage*, do diagrama da Figura 15.

A Figura 23 a seguir ilustra a página "Luzes" com o novo atuador já configurado, chamado "Sala".

O atuador "Sala" da Figura 23, recém configurado conforme Figuras 21 e 22, já

Figura 23 – Telas da aplicação móvel: atuador para iluminação desligada e ligada.



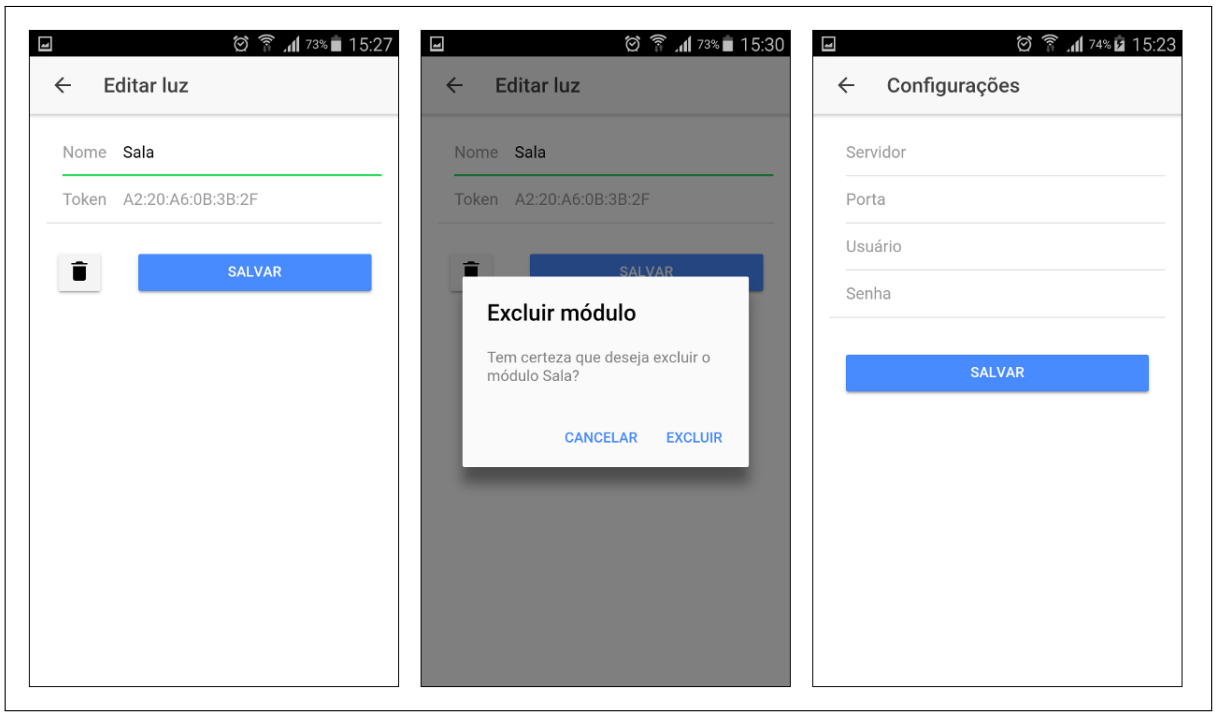
Fonte – Próprio autor.

encontra-se em pleno funcionamento, podendo ser “ligado” ou “desligado”, segundo ilustra o estado do botão *toggle* em ambas as telas.

Deve-se destacar que, na aba “Tomadas”, a instalação e utilização de um atuador ocorre de modo idêntico ao descrito aqui para um atuador para “Luzes”. Ainda, o botão *toogle*, que referencia o estado da luz, altera seu estado em concordância com um novo estado da luz definido pela mudança da chave mecânica do interruptor doméstico, conforme a aplicação recebe notificação de alteração pelo *broker*, enviada pelo atuador.

Por fim, a Figura 24 a seguir ilustra as demais telas da aplicação móvel.

Figura 24 – Telas da aplicação móvel: demais telas.



Fonte – Próprio autor.

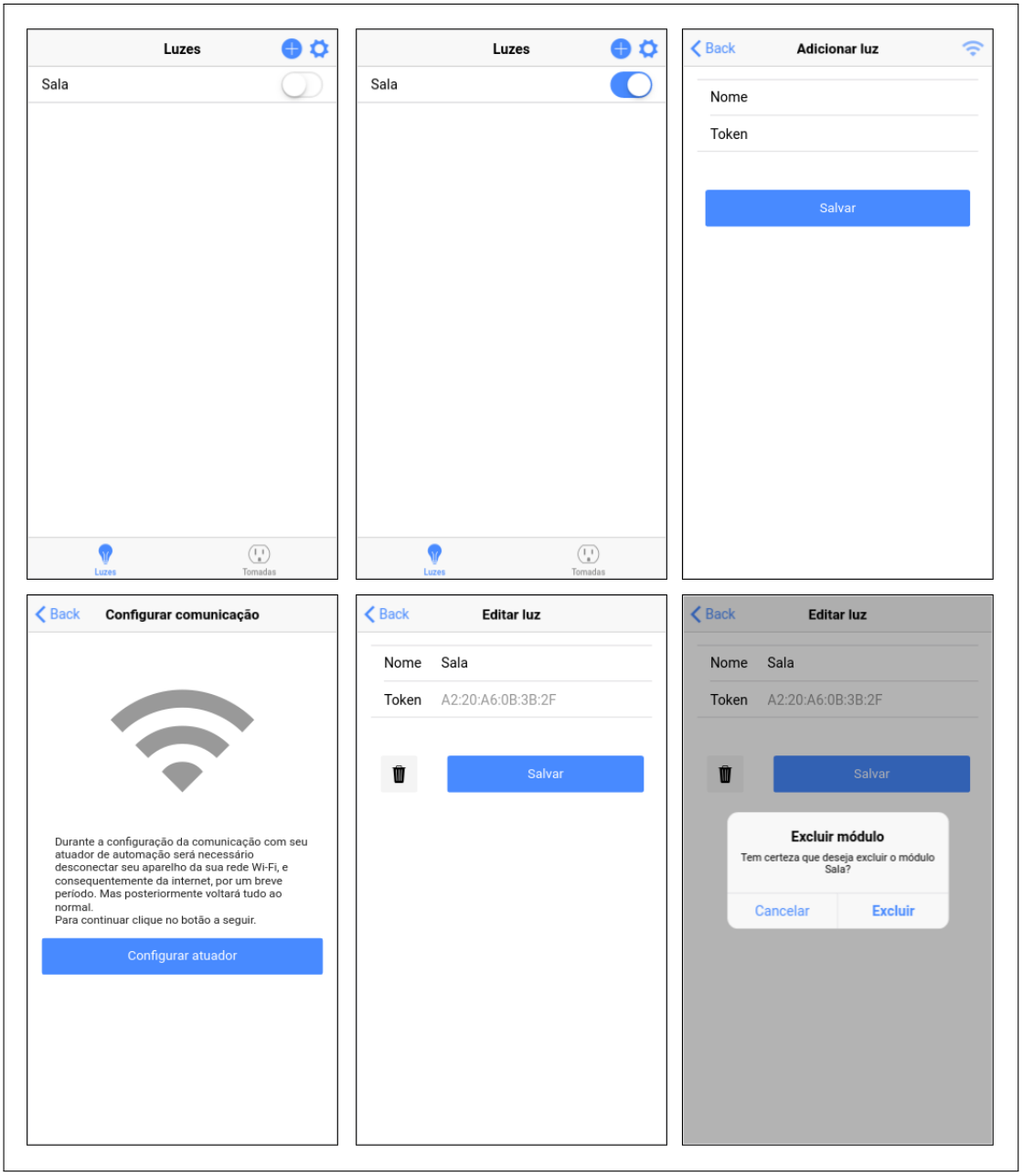
A tela à esquerda da Figura 24 é a responsável pela edição ou exclusão de um atuador. É aberta ao "clicar" e segurar no item pertinente da lista de atuadores, conforme Figura 23. Caso o usuário queria alterar o nome do atuador, basta alterar a informação do campo "Nome" e salvar, porém, caso queira excluir o atuador, basta clicar no primeiro botão *affordance* lixeira, que uma confirmação é exibida, conforme a tela do centro da Figura.

Já a tela à direita da Figura 24 refere-se à definição das informações pertinentes a comunicação entre a aplicação e o servidor do *broker*.

Quanto as classes controladoras das telas primeira e segunda da Figura 24, bem como da terceira, são respectivamente as classes *ModulePage* e *SettingsPage*, do diagrama da Figura 15.

Todos os componentes visuais utilizados nas telas apresentadas nas Figuras 20 a 24 são réplicas dos componentes padrões do Material Design (GOOGLE, 2018) pertinente ao sistema operacional móvel Android (GOOGLE, 2017). Ademais, graças aos componentes visuais do Ionic Framework serem reutilizáveis para diversas plataformas, a mesma interface pode ser gerada para o sistema operacional iOS (APPLE, 2018), de *smartphones* iPhone, utilizando seus padrões, conforme ilustra a Figura 25 a seguir.

Figura 25 – Telas da aplicação móvel sobre padrões iOS.



Fonte – Próprio autor.

Para finalizar, sobre a aplicação móvel, seu código fonte encontra-se no repositório online do autor Magalhães (2018b).

No que se refere ao servidor para execução do *broker* que intermedia a comunicação entre a aplicação móvel e os atuadores, utiliza-se o serviço em nuvem, esta fundamentada na Seção 3.5, chamado CloudMQTT (CLOUDMQTT, 2018).

O serviço CloudMQTT gerencia servidores MQTT na nuvem da Amazon Web Services (AWS, 2018) por meio de uma interface WEB amigável (CLOUDMQTT, 2018). Possui

diversos planos para clientes, o mais básico oferece até dez conexões simultâneas a 10 Kbit/s, gratuitamente, e por isto escolhido para este trabalho.

Finalmente concluída esta fase Construção, resta partir para a próxima e última fase do Paradigma da Prototipação no Processo Evolucionário: a fase Realimentação.

4.4 Realimentação

Nesta última fase, Realimentação, é realizada a validação para que se possa concluir se foi implementado o definido na primeira fase Comunicação, Seção 4.1. Isso ocorre por meio de testes de validação.

Testes de validação, segundo Pressman (2011), são os testes que determinam, por meio de *feedback* dos *stackholders* do projeto, se o sistema em desenvolvimento está sendo implementado da maneira correta e se está razoavelmente de acordo com o esperado por seus requisitos. Para tanto, determina-se o conjunto de testes de validação necessários para a aprovação desta primeira iteração.

Nos Quadro 13 e 14 adiante encontra-se o plano de testes definido, os quais possuem como critérios de aceitação os critérios definidos nas estórias de usuário, Quadros 3 a 10.

Sobre os casos de teste dos Quadros 13 e 14, deve-se explanar:

a) casos CT003 e CT004: os estados são verificados por meio dos estados dos botões *toggle* correspondentes. À princípio, os estados devem corresponder ao inverso do qual se deseja verificar a mudança, ou seja, para o CT003, inicialmente, o estado da luz deve corresponder à luz apagada (botão *toggle* desativado) para verificação da mudança para luz acesa ao alterar a chave mecânica do interruptor para acendimento manual da luz;
b) casos CT005 e CT006: deve-se ligar algum aparelho elétrico à tomada pertinente para plena verificação do acionamento e desacionamento do circuito;
c) casos CT007 e CT008: as caixas padrões de rede elétrica são as caixas retangulares embutidas em paredes prediais onde são instalados interruptores de iluminação e/ou tomadas. Além disso, os testes devem ser executados tomando as devidas precauções de segurança;
d) casos CT009 e CT010: ao fim dos passos deve-se executar os casos de teste CT001 e CT001 a CT004, respectivamente, para plena aprovação do testes.

Quadro 13 – Plano de testes de validação do sistema.

Caso de teste	Estória de usuário	Descrição	Passos	Resultado esperado
CT001	EU001	Submeter comando para acender a luz.	1 - Na tela "Luzes"; 2 - Apertar botão toggle para acender uma luz.	A luz pertinente acende.
CT002	EU002	Submeter comando para apagar a luz.	1 - Na tela “Luzes”; 2 - Apertar botão toggle para apagar uma luz.	A luz pertinente apaga.
CT003	EU003	Acender luz pelo interruptor.	1 - Na tela “Luzes”; 2 - Acender a luz pelo interruptor; 3 - Verificar estado na aplicação.	O estado corresponde a luz acesa.
CT004	EU003	Apagar luz pelo interruptor.	1 - Na tela “Luzes”; 2 - Apagar a luz pelo interruptor; 3 - Verificar estado na aplicação.	O estado corresponde a luz apagada.
CT005	EU004	Submeter comando para ativar tomada.	1 - Na tela “Tomadas”; 2 - Apertar botão toggle para ativar uma tomada.	A tomada pertinente é ativada.
CT006	EU005	Submeter comando para desativar tomada.	1 - Na tela “Tomadas”; 2 - Apertar botão toggle para desativar uma tomada.	A tomada pertinente é desativada.
CT007	EU006	Instalar atuador em caixa padrão de rede elétrica predial de interruptor de luz.	1 - Remover interruptor de luz da caixa de energia; 2 - Ligar fio fase no interruptor para three-way; 3 - Realizar ligação three-way entre o relé e o interruptor; 4 - Ligar saída do relé na entrada do sensor; 5 - Ligar saída do sensor para a luz; 6 - Embutir tudo na caixa de energia.	O atuador está instalado embutido e em pleno funcionamento.
CT008	EU006	Instalar atuador em caixa padrão de rede elétrica predial de tomada.	1 - Remover tomada da caixa de energia; 2 - Ligar fio fase no relé; 3 - Ligar saída do relé (normalmente ligada) a entrada do sensor; 4 - Ligar saída do sensor à tomada; 5 - Embutir tudo na caixa de energia.	O atuador está instalado embutido e em pleno funcionamento.

Fonte – Próprio autor.

Quadro 14 – Continuação do plano de testes de validação do sistema.

Caso de teste	Estória de usuário	Descrição	Passos	Resultado esperado
CT009	EU007	Configurar um novo atuador para plena comunicação com a aplicação móvel.	1 - Na tela “Luzes”; 2 - Apertar botão “**+**” na barra superior; 3 - Na tela “Adicionar luz”; 4 - Apertar botão “Wi-Fi” na barra superior; 5 - Na tela “Configurar comunicação”; 6 - Apertar botão “CONFIGURAR ATUADOR”; 7 - Na tela para seleção de rede Wi-Fi, escolher uma rede e salvar senha; 8 - Retornar para tela “Adicionar luz” com campo “Token” preenchido. 9 - Adicionar o nome “Teste” ao campo “Nome”; 10 - Apertar botão “SALVAR”; 11 - Realizar teste CT001.	O atuador está em plena comunicação com a aplicação móvel.
CT010	EU008	Utilizar funções do sistema de fora da residência de implantação.	1 - Estando fora da residência e conectado a uma rede Wi-Fi distinta, da instalação do atuador; 2 - Realizar testes CT001 a CT004.	Os casos de teste CT001 a CT004 foram aprovados.

Fonte – Próprio autor.

Para execução dos testes é necessário adaptar o protótipo de atuador da Figura 17, para instalação em cenário satisfatoriamente próximo do real. Para isso, o protótipo é instalado embutido em caixas de rede elétrica para interruptores de iluminação e tomada, Figura 26 a seguir.

Conforme ressalvado na fase Construção, Seção 4.3, não foi utilizada a fonte universal da Figura 5 por riscos à segurança, ficando sua utilização para iterações futuras.

Definido o plano de testes e preparado o ambiente, segue-se para a execução dos testes de validação.

Figura 26 – Protótipo embutido numa caixa de rede elétrica.

Fonte – Próprio autor.

4.4.1 Execução dos testes de validação

Nos Quadros 15 e 16 a seguir, encontra-se o resumo sobre as saídas e resultados dos testes de validação desta iteração definidos nos Quadros 13 e 14.

Quadro 15 – Resumo da execução dos testes de validação.

Caso de teste	**Descrição**	**Resultado esperado**	**Saída**	**Resumo**
CT001	Submeter comando para acender a luz.	A luz pertinente acende.	A luz pertinente está acesa.	Aprovado.
CT002	Submeter comando para apagar a luz.	A luz pertinente apaga.	Luz pertinente está apagada.	Aprovado.
CT003	Acender luz pelo interruptor.	O estado corresponde a luz acesa.	Estado correspondendo a luz acesa.	Aprovado.
CT004	Apagar luz pelo interruptor.	O estado corresponde a luz apagada.	Estado correspondendo a luz apagada.	Aprovado.
CT005	Submeter comando para ativar tomada.	A tomada pertinente é ativada.	A tomada pertinente está ativada.	Aprovado.
CT006	Submeter comando para desativar tomada.	A tomada pertinente é desativada.	A tomada pertinente está desativada.	Aprovado.
CT007	Instalar atuador em caixa padrão de rede elétrica predial de interruptor de luz.	O atuador está instalado embutido e em pleno funcionamento.	O atuador instalado está funcionando adequadamente.	Aprovado.

Fonte – Próprio autor.

Quadro 16 – Continuação do resumo da execução dos testes de validação.

Caso de teste	**Descrição**	**Resultado esperado**	**Saída**	**Resumo**
CT008	Instalar atuador em caixa padrão de rede elétrica predial de tomada.	O atuador está instalado embutido e em pleno funcionamento.	O atuador instalado está funcionando adequadamente.	Aprovado.
CT009	Configurar um novo atuador para plena comunicação com a aplicação móvel.	O atuador está em plena comunicação com a aplicação móvel.	A comunicação está ocorrendo adequadamente. CT001 aprovado.	Aprovado.
CT010	Utilizar funções do sistema de fora da residência de implantação.	Os casos de teste CT001 a CT004 foram aprovados.	Casos CT001 a CT004 aprovados conforme especificado.	Aprovado.

Fonte – Próprio autor.

Diante dos excelentes resultados obtidos para todos os casos de teste, pode-se concluir que esta primeira iteração do desenvolvimento do sistema de automação residencial modular, por meio do Processo Evolucionário, está finalizada, podendo ser dado início a uma nova iteração do ciclo de desenvolvimento evolucionário, conforme definido no início deste capítulo e ilustrado na Figura 1.

Segue-se, portanto, para os Resultados e Discussão a cerca do produzido por este trabalho.

5 RESULTADOS E DISCUSSÃO

Como comentado na Introdução, Capítulo 1, este trabalho possui como objetivo principal desenvolver um sistema de automação residencial modular com foco na flexibilização da instalação, embarcando os seus atuadores em caixas da rede elétrica predial. Tal solução busca proporcionar facilidade de implantação, manutenção e expansão, além de possuir interações com os usuários por meio de um aplicativo para dispositivos móveis.

Diante do produzido em Desenvolvimento, Capítulo 4, e, especialmente, dos resultados obtidos nas fases do desenvolvimento Construção e Realimentação, Seções 4.3 e 4.4, pode-se concluir que se atingiu o objetivo principal, por meio da implementação de um atuador de automação o qual é instalado de modo embarcado à caixas de rede elétrica residencial de interruptores de iluminação e tomadas, como mostra a Figura 26, que não necessita de mão de obra especializada em automação ou tecnologia da informação, podendo ser instalado por um eletricista predial ou outro profissional da área, o que deve ainda reduzir significativamente os custos de instalação, manutenção e expansão.

Além disso, diante do fato do atuador ser controlado por meio de aplicação móvel e da internet, bem como do caráter pervasivo da instalação do atuador, ao ser embutido em objeto comum do cotidiano, o mesmo se sustenta sobre os principais pilares deste trabalho: Internet das Coisas e Computação Ubíqua.

Outros problemas comuns a sistemas de automação residenciais comentados na Introdução deste trabalho, suas soluções e seus resultados aqui desenvolvidos são comentados adiante.

1. dependência de reforma elétrica predial para implantação de atuadores;
2. necessidade de mão de obra especializada para instalação, manutenção e expansão;
3. produtos dependentes de fornecedores dedicados – não havendo extenso catálogo de sistemas de prateleira;
4. alto custo para instalação, manutenção e expansão.

Sobre as alíneas 1 e 4 da enumeração anterior, o atuador implementado neste trabalho não necessita de reforma elétrica predial para automação de tomadas ou interruptores, bastando apenas sua instalação ocorrer de modo embutido na caixa da rede elétrica pertinente, por um eletricista ou outro profissional capacitado.

Acerca das alíneas 2 e 3 da enumeração anterior, o atuador e a aplicação móvel deste

trabalho buscam ser uma solução simples: fácil montagem, instalação e configuração. Diante disso, a replicação dos resultados produzidos por esta pesquisa, sob caráter comercial, deve agregar às opções de sistemas de prateleira. Ademais, tornando-se um produto final de prateleira, após novas iterações sob o Processo Evolucionário (passando assim por refinamento de suas principais características, implementação e melhorando suas respostas aos seus objetivos), por meio de uma produção em massa e de um modelo de negócio adequado, o sistema de automação residencial deste trabalho deve tornar-se uma opção de peso para o catálogo de produtos voltados à automação residencial.

Por fim, não houve dificuldades sobre as pesquisas para fundamentação e definição dos estados da arte e da prática deste trabalho e para a implementação da proposta e sua documentação, por parte do autor.

Em Considerações Finais, a seguir, são comentados os resultados obtidos e possíveis incrementos para trabalhos futuros.

6 CONSIDERAÇÕES FINAIS

Ao longo deste trabalho foram realizadas pesquisas sobre o estado da arte e, principalmente, da prática, como documentado em Trabalhos Relacionados, Capítulo 2. A pesquisa sobre o estado da prática foi um destaque, haja vista a busca por soluções de mercado sobre automação permitir elucidar as melhores práticas e inferir melhores soluções, não se restringindo a somente o que há na literatura acadêmica.

A Fundamentação Teórica, Capítulo 3, buscou explanar paradigmas em evidência no momento utilizados como pilares deste trabalho, como Computação Ubíqua, Internet das Coisas e Computação em Nuvem.

Em Desenvolvimento, Capítulo 4, em princípio, definiu-se um processo para melhor direcionar a implementação da parte prática da pesquisa deste trabalho e, assim, garantir a obtenção dos melhores resultados possíveis e, com o processo definido, buscamos seguir as fases do mesmo, elucidando, projetando, implementando e testando, sob um típico processo de Engenharia de Software.

O capítulo Resultados e Discussão, de número 5, voltou-se a resumir o produzido por este trabalho conectando o obtido ao projeto e as fundamentações definidas. Além disso, deve-se destacar como ponto positivo a ideia de embutir o atuador de automação a objeto comum do dia a dia de qualquer edificação, o que permitiu solucionar problemas pertinentes a sistemas de automação voltados ao público final.

Ademais, ressalta-se a capacidade empreendedora deste trabalho, sob uma perspectiva do autor, diante do potencial escalonável de produção de um sistema de automação sobre atuadores modulares de pequenas dimensões voltados a facilidade de instalação e manutenção.

Possíveis melhorias, para trabalhos futuros, no sistema desenvolvido neste trabalho são comentadas na próxima seção.

6.1 Trabalhos Futuros

Para finalizar, diante do desenvolvido, deve-se destacar a necessidade de amadurecimento do produto deste trabalho, possuindo o autor sugestões de melhorias para possíveis próximas iterações para evolução em trabalhos futuros, são elas os requisitos funcionais definidos nos Quadros 17 a 20 e incrementos no *hardware* comentados em seguida.

Quadro 17 – Estória de usuário EU009 - Ligar TV.

EU009 - Ligar TV	
Como um	usuário
Eu quero	ligar um aparelho de televisão
De modo que	eu não necessite do controle remoto
Critério de aceitação	a televisão pertinente deve ligar

Fonte – Próprio autor.

Quadro 18 – Estória de usuário EU010 - Desligar TV.

EU010 - Desligar TV	
Como um	usuário
Eu quero	desligar um aparelho de televisão
De modo que	eu não necessite do controle remoto
Critério de aceitação	a televisão pertinente deve desligar

Fonte – Próprio autor.

Quadro 19 – Estória de usuário EU011 - Aumentar volume da TV.

EU011 - Aumentar volume da TV	
Como um	usuário
Eu quero	aumentar o volume do som da televisão
De modo que	eu não necessite do controle remoto
Critério de aceitação	a televisão pertinente deve aumentar o volume

Fonte – Próprio autor.

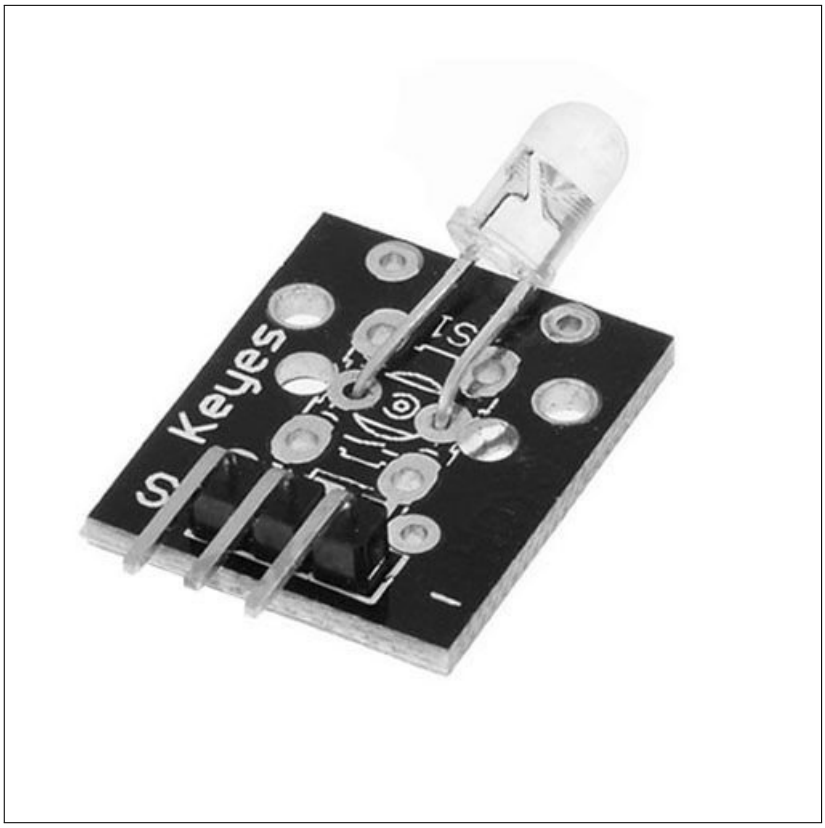
Quadro 20 – Estória de usuário EU012 - Diminuir volume da TV.

EU012 - Diminuir volume da TV	
Como um	usuário
Eu quero	diminuir o volume do som da televisão
De modo que	eu não necessite do controle remoto
Critério de aceitação	a televisão pertinente deve diminuir o volume

Fonte – Próprio autor.

As estórias de usuário sugeridas para trabalhos futuros nos Quadros 17 a 20 referem-se à interação do sistema de automação com aparelhos de televisão, a qual deve acontecer por meio de interface à comunicação por infravermelho. Para isto, um componente LED emissor infravermelho, ilustrado na Figura 27, deve ser acrescido aos componentes da Figura 5 para permitir essa comunicação.

Figura 27 – Emissor infravermelho KY-005.



Fonte – Autocore (2018).

Ademais, para trabalhos futuros, sugere-se uma validação por implantação em ambiente real seguindo o modelo ilustrado na Figura 28 a seguir, a fim de obter maior fidelidade aos resultados de testes.

A Figura 28 ilustra uma implantação do sistema de automação deste trabalho num ambiente residencial real, uma casa composta de cinco cômodos, com instalação de oito atuadores, sendo cinco para interruptores de iluminação e três para tomadas. Além disso, ilustra o roteador Wi-Fi doméstico necessário e dois *smartphones* para a aplicação móvel, um dentro da residencia e outro fora.

Figura 28 – Ilustração de implantação para validação em ambiente real.

Fonte – Adaptado de Mariz (2014).

REFERÊNCIAS

APPLE. **iOS**. 2018. Disponível em: <https://www.apple.com/br/ios/>. Acesso em: 31 jan. 2018.

ARAUJO, R. B. de. Computação ubíqua: Princípios, tecnologias e desafios. In: **XXI Simpósio Brasileiro de Redes de Computadores**. Natal, CE: [s.n.], 2003. v. 8, p. 11–13.

ARDUINO. **Arduino - Home**. 2018. Disponível em: <https://www.arduino.cc/>. Acesso em: 05 fev. 2018.

AUTOCORE. **Módulo Emissor Infravermelho KY-005**. 2018. Disponível em: <https://www.autocorerobotica.com.br/modulo-emissor-infravermelho-ky-005>. Acesso em: 16 fev. 2018.

AWS. **Amazon Web Services (AWS) - Serviços de compução em nuvem**. 2018. Disponível em: <https://aws.amazon.com/pt/>. Acesso em: 09 fev. 2018.

BASS, L. **Software architecture in practice**. [S.l.]: Pearson Education India, 2007.

BEZERRA, E. **Princípios de Análise e Projeto de Sistema com UML**. [S.l.]: Elsevier Brasil, 2017. v. 3.

BRUSH, A.; LEE, B.; MAHAJAN, R.; AGARWAL, S.; SAROIU, S.; DIXON, C. Home automation in the wild: challenges and opportunities. In: ACM. **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**. [S.l.], 2011. p. 2115–2124.

CAMPAGNOLI, J. de A. **PhoneGap e Cordova**: como criar aplicativos mobile híbridos e offline. 2016. Disponível em: <https://www.devmedia.com.br/phonegap-e-cordova-como-criar-aplicativos-mobile-hibridos-e-offline/32361>. Acesso em: 31 jan. 2018.

CLEMENTS, P.; GARLAN, D.; LITTLE, R.; NORD, R.; STAFFORD, J. Documenting software architectures: views and beyond. In: IEEE COMPUTER SOCIETY. **Proceedings of the 25th International Conference on Software Engineering**. [S.l.], 2003. p. 740–741.

CLOUDMQTT. **Hosted message broker for the Internet of Things**. 2018. Disponível em: <https://www.cloudmqtt.com>. Acesso em: 09 fev. 2018.

CORDOVA. **Overview**. 2017. Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>. Acesso em: 01 fev. 2018.

DEVELOPERS, A. **Introdução ao Android**. 2018. Disponível em: <https://developer.android.com/guide/index.html>. Acesso em: 02 fev. 2018.

DUMAS, M.; HOFSTEDE, A. H. T. Uml activity diagrams as a workflow specification language. In: SPRINGER. **International conference on the unified modeling language**. [S.l.], 2001. p. 76–90.

ESP8266. **File System - ESP8266 Arduino Core**. 2017. Disponível em: <http://esp8266.github.io/Arduino/versions/2.0.0/doc/filesystem.html>. Acesso em: 06 fev. 2018.

FIELDING, R.; GETTYS, J.; MOGUL, J.; FRYSTYK, H.; MASINTER, L.; LEACH, P.; BERNERS-LEE, T. **Hypertext transfer protocol–HTTP/1.1**. [S.l.], 1999.

FOROUZAN, B. A.; FEGAN, S. C. **TCP/IP protocol suite**. [S.l.]: McGraw-Hill Higher Education, 2002.

GILL, K.; YANG, S.-H.; YAO, F.; LU, X. A zigbee-based home automation system. **IEEE Transactions on Consumer Electronics**, IEEE, v. 55, n. 2, 2009.

GOOGLE. **Android**. 2017. Disponível em: <https://www.android.com/>. Acesso em: 12 nov. 2017.

GOOGLE. **Material Design**. 2018. Disponível em: <https://material.io/>. Acesso em: 08 fev. 2018.

HAN, D.-M.; LIM, J.-H. Design and implementation of smart home energy management systems based on zigbee. **IEEE Transactions on Consumer Electronics**, IEEE, v. 56, n. 3, 2010.

HIVEMQ. **MQTT Security Fundamentals: TLS / SSL**. 2017. Disponível em: <https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl>. Acesso em: 06 fev. 2018.

HUMPHRIES, L. S.; RASMUSSEN, G.; VOITA, D. L.; PRITCHETT, J. D. **Home automation system**. [S.l.]: Google Patents, 1997. US Patent 5,621,662.

HUNKELER, U.; TRUONG, H. L.; STANFORD-CLARK, A. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In: IEEE. **Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on**. [S.l.], 2008. p. 791–798.

INSTRUCTABLES. **NODEMCU ONBOARD LED ACCESS VIA WEB SERVER**. 2016. Disponível em: <http://www.instructables.com/id/NodeMCU-Onboard-LED-Access-Via-Web-Server/>. Acesso em: 01 fev. 2018.

IONIC. **Core Concepts**. 2017. Disponível em: <https://ionicframework.com/docs/intro/concepts/>. Acesso em: 01 fev. 2018.

IONIC. **Tabs - Ionic API Documentation**. 2018. Disponível em: <https://ionicframework.com/docs/api/components/tabs/Tabs/>. Acesso em: 05 fev. 2018.

JAVASCRIPT. **Introducing JSON**. 2018. Disponível em: <https://www.json.org/>. Acesso em: 05 fev. 2018.

JOHNSON, D. B.; MALTZ, D. **Mobile computing**. [S.l.]: Kluwer academic publishers Dordrecht, 1996.

JR, R. J. B.; ROMANOWIZ, J. D.; STAPLES, C. W. **Energy management and home automation system**. [S.l.]: Google Patents, 1996. US Patent 5,544,036.

KNOLLEARY. **Arduino Client for MQTT - Library**. 2017. Disponível em: <https://github.com/knolleary/pubsubclient>. Acesso em: 16 abr. 2017.

KUNJUMON, S.; PINTO, K. **Home Automation System**. 2016.

LAUNEY, R. O.; GRENDLER, P. A.; PACKHAM, D. L.; BATTAGLIA, J. M.; LEVINE, H. E. **Expandable home automation system**. [S.l.]: Google Patents, 1992. US Patent 5,086,385.

MAGALHãES, D. **Automation Actuator**. 2018. Disponível em: <https://github.com/DaviMagalhaes/AutomationActuator>. Acesso em: 06 fev. 2018.

MAGALHãES, D. **Home Automation**. 2018. Disponível em: <https://github.com/DaviMagalhaes/HomeAutomation>. Acesso em: 08 fev. 2018.

MAGALHãES, D. **WifiManager**. 2018. Disponível em: <https://github.com/DaviMagalhaes/wifimanager>. Acesso em: 06 fev. 2018.

MARIZ. **PLANTAS DE CASAS SIMPLES E BARATAS: MODELOS GRÁTIS**. 2014. Disponível em: <http://www.euamodecoracao.com/plantas-de-casas-simples-e-baratas-modelos-gratis/>. Acesso em: 02 mar. 2018.

MELL, P.; GRANCE, T. *et al.* **The NIST definition of cloud computing**. [S.l.]: Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011.

MIT. **The MIT License**. 2018. Disponível em: <https://opensource.org/licenses/MIT>. Acesso em: 05 fev. 2018.

MORONY, J. **An In-Depth Explanation of Providers in Ionic 2**. 2018. Disponível em: <https://www.joshmorony.com/an-in-depth-explanation-of-providers-in-ionic-2/>. Acesso em: 02 fev. 2018.

NODEMCU. **NodeMCU - An open-source firmware based on ESP8266 wifi-soc**. 2017. Disponível em: <http://nodemcu.com>. Acesso em: 25 maio 2017.

PETROV, I.; SERU, S.; PETROV, S. Home automation system. **School of Engineering Science**, 2011.

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 7ª Edição. 2011.

PUC-RIO. **The Programming Language Lua**. 2018. Disponível em: <https://www.lua.org/>. Acesso em: 06 fev. 2018.

PURCHASE, H. C.; COLPOYS, L.; MCGILL, M.; CARRINGTON, D.; BRITTON, C. Uml class diagram syntax: an empirical study of comprehension. In: AUSTRALIAN COMPUTER SOCIETY, INC. **Proceedings of the 2001 Asia-Pacific symposium on Information visualisation-Volume 9**. [S.l.], 2001. p. 113–120.

RICQUEBOURG, V.; MENGA, D.; DURAND, D.; MARHIC, B.; DELAHOCHE, L.; LOGE, C. The smart home concept: our immediate future. In: IEEE. **E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference on**. [S.l.], 2006. p. 23–28.

SEMIG, P. **A Current Sensing Tutorial** - Part 1: Fundamentals. 2012. Disponível em: <https://www.eetimes.com/document.asp?doc_id=1279404>. Acesso em: 30 abr. 2018.

TZAPU. **WiFiManager - Library**. 2017. Disponível em: <https://github.com/tzapu/WiFiManager>. Acesso em: 16 abr. 2017.

WEISER, M. The computer for the 21st century. **Scientific american**, Nature Publishing Group, v. 265, n. 3, p. 94–104, 1991.

XIA, F.; YANG, L. T.; WANG, L.; VINEL, A. Internet of things. **International Journal of Communication Systems**, v. 25, n. 9, p. 1101, 2012.

ZIGBEE, A. Zigbee specification. **ZigBee document 053474r13**, 2006.