INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS

CAMPUS MANAUS DISTRITO INDUSTRIAL

BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

FRANCISCO FERREIRA DA SILVA NETO

SISTEMA REMOTO DE AUTOMAÇÃO RESIDENCIAL COM RECURSOS DE COMPUTAÇÃO EM NUVEM

MANAUS – AM

2022

FRANCISCO FERREIRA DA SILVA NETO

SISTEMA REMOTO DE AUTOMAÇÃO RESIDENCIAL COM RECURSOS DE COMPUTAÇÃO EM NUVEM

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia do Amazonas como requisito parcial para a obtenção do grau de bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Vitor Bremgartner da Frota

MANAUS-AM

2022

Dados Internacionais de Catalogação na Publicação (CIP)

S586s Silva Neto, Francisco Ferreira da.

Sistema remoto de automação residencial com recursos de computação em nuvem. / Francisco Ferreira da Silva Neto. – Manaus, 2022.
59 f. : il. color

TCC (Graduação em engenharia de controle e automação) – Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Distrito Industrial, 2022.
Orientador: Prof. Dr. Vitor Bremgartner da Frota

1. Automação residencial. 2. Computação em nuvem. 3. Aplicativo android. 4. Internet das coisas. 5. MQTT. I. Frota, Vitor Bremgartner da (orient.) II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 629.8

Elabora por Fcª. Amélia Frota, registro n.858 (CRB11)

Ministério da Educação
Instituto Federal de Educação, Ciência e Tecnologia do Amazonas
Campus Manaus Distrito Industrial
Departamento de Automação e Produção
Engenharia de Controle e Automação

TERMO DE APROVAÇÃO

SISTEMA REMOTO DE AUTOMAÇÃO RESIDENCIAL COM RECURSOS DE COMPUTAÇÃO EM NUVEM

por

FRANCISCO FERREIRA DA SILVA NETO

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 13 de maio de 2022 como requisito parcial para a obtenção do título de Bacharel em Engenharia de Controle e Automação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Vitor Bremgartner da Frota
Prof.(a) Orientador(a)

Profa. Me. Priscila Silva Fernandes
Membro titular

Prof. Dr. Alyson de Jesus dos Santos
Membro titular

AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de minha vida. Portando, desde já peço desculpas àquelas que não estão presente entre essas palavras, mas elas podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Agradeço primeiramente a Deus, por me proporcionar todo o necessário para finalização desse trabalho, principalmente pela minha saúde, inteligência e força de vontade.

A minha família, em especial, minha esposa Anne Paloma e aos meus filhos Helena e Guilherme, pois eles são a minha maior e melhor fonte de inspiração para seguir em frente, inclusive para terminar esse trabalho em um momento nada fácil da minha vida.

Aos meus parentes, principalmente a minha falecida bisavó, que considero como mãe, dona Julieta Valente, que me apoiou e me forneceu tudo o que eu precisava desde o começo da minha vida. Foram de grande importância em minha caminhada também meu pai e minha mãe, juntamente com minhas tias Ana e Nádia.

Ao meu orientador Vitor Bremgartner, que foi bastante paciente e com sua sabedoria me deu todo o norte para o correto desenvolvimento da pesquisa.

Aos colegas de curso que compartilharam conhecimentos, diversão e momento felizes em nossos grupos de estudos.

A Secretaria do curso pelo apoio e cooperação para que pudesse me formar, mesmo com o prazo apertado. Principalmente ao Sr. Edevaldo.

Aos sócios e fundadores da empresa iotech Automação, que me acolheram e me ajudaram desenvolveram como profissional.

Enfim, a todos os que de alguma forma contribuíram para a realização desta pesquisa.

“Quem nunca errou nunca experimentou nada novo”

Albert Einstein.

RESUMO

DA SILVA NETO, F. F. **Sistema de automação residencial com recursos de computação em nuvem**. 2022.1. 58. Trabalho de Conclusão de Curso Bacharelado em Engenharia de Controle e Automação – Instituto Federal de Educação, Ciência e Tecnologia do Amazonas – Campus Manaus Distrito Industrial. Manaus, Amazonas, 2022.

Computação em nuvem (CN) é um recurso muito usado na área da Tecnologia da Informação (TI), sua implementação em ambiente residencial utilizando o conceito de Internet das Coisas (IdC) pode proporcionar toda a segurança, processamento e estabilidade dos grandes serviços de CN atuais, além de escalabilidade e integração com outros serviços, como banco de dados e aplicativos de dispositivos móveis. Este trabalho tem como objetivo proporcionar ao usuário o controle de dispositivos domésticos de forma remota através de um aplicativo móvel. Para alcançar tal objetivo foi necessário utilizar recursos de *harware* e *software* integrados em uma arquitetura de IdC horizontal. Especificamente foi necessário utilizar a AWS IoT como principal serviço de nuvem, a IDE Android Studio, o *framework* Zerynth e a placa de desenvolvimento ESP32, além de alguns sensores e atuadores. Ao fim se obteve uma arquitetura proposta completamente implementada e com boa estabilidade na troca de informações, possibilitando assim o comando de forma remota e *online* dos dispositivos eletrônicos no protótipo obtido.

Palavras-chave*:* Automação Residencial. Computação em Nuvem. Aplicativo Android. Internet das Coisas. MQTT,

.

ABSTRACT

DA SILVA NETO, F. F. **Home automation system with cloud computing resources.** 2022.1. 58. Bachelor's Degree Completion Work in Control and Automation Engineering – Federal Institute of Education, Science and Technology of Amazon – Campus Manaus Distrito Industrial. Manaus, Amazonas 2022.

Cloud Computing (CC) is a resource widely used in the Information Technology (IT) area, its implementation in a residential environment using the Internet of Things (IoT) concept can provide all the security, processing and stability of large CC services, as well as scalability and integration with other services such as database and mobile applications. This work aims to provide the user with remote control of home devices through a mobile application. To achieve this goal, it was necessary to use hardware and software resources integrated in a horizontal IoT architecture. Specifically, it was necessary to use AWS IoT as the main cloud service, the Android Studio IDE, the Zerynth framework and the development module ESP32, in addition to some sensors and actuators. In the end, a proposed architecture was fully implemented and with good stability in the exchange of information, thus enabling remote and online command in the electronics at result prototype.

Keyword*s:* Residential Automation. Cloud Computing. Android application. Internet of Things. MQTT.

LISTA DE ILUSTRAÇÕES

Figura 1 - Modelos de arquiteturas vertical e horizontal....19
Figura 2 - Exemplos de arquiteturas horizontal e vertical....19
Figura 3 - Modelo OSI e TCP/IP....21
Figura 4 - Modelo de publicação/assinatura de exemplo....23
Figura 5 - Diagrama de conectividade dos serviços do AWS IoT Core....29
Figura 6 - Circuito de iluminação com Led....33
Figura 7 - sensor de movimento/presença....34
Figura 8 - Circuito de Acionamento do Cooler de Resfriamento....34
Figura 9 - Placa de desenvolvimento NodeMCU-32S....35
Figura 10 - Conexão elétrica do microcontrolador em *protoboard*....36
Figura 11 - Maquete escolhida para montagem dos dispositivos....37
Figura 12 - Itens de decoração da maquete....37
Figura 13 - Interface de programação do Android Studio....39
Figura 14 - Interface padrão do aplicativo exemplo.....40
Figura 15 - Comparação de exemplo e a aplicação modificada....41
Figura 16 - Arquitetura proposta....42
Figura 17 - Tela de configuração do AWS Cognito....44
Figura 18 - Tela de gerenciamento e listagens de coisas criadas na AWS IoT....45
Figura 19 - Criação de itens ou coisas na AWS IoT....46
Figura 20 - Editor do estado do device shadow.....51
Figura 21 - Sombra da coisa obtida.....52
Figura 22 - Aplicativo Android Obtido....53
Figura 23 - Ícone do aplicativo na tela do smartphone....54
Figura 24 - Montagem final da maquete....55

LISTA DE ABREVIATURAS E SIGLAS

IFAM	Instituto Federal de Educação, Ciência e Tecnologia do Amazonas
CN	Computação em Nuvem
TI	Tecnologia da Informação
AR	Automação Residencial
IHC	Interação Humano-Computador
IoT	*Internet of Things*
MIT	*Massachusetts Institute of Technology*
B2B	*Business to Business*
B2C	*Business to Commerce*
MCU	*Microcontroller Unit*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
OSI	*Open Systems Interconnection*
HTTP	*Hipertex Transfer Protocol*
MQTT	*Message Queue Telemetry Transport*
TELNET	*Terminal Virtual*
Wi-Fi	*WirelessFidelity*
LLC	*Logic Link Control*
MAC	*Media Access Control*
TLS	*Transport Layer Security*
SDK	*software development kit*
APK	*Android Application Pack*
RAM	*random access memory*
GPIO	*General Purpose Input/Output*
ADC	*Analog Digital Converter*
A/D	Analógico/Digital
D/A	Digital / Analógico
I2C	*Inter-Integrated Circuit*
I2S	*Inter-IC Sound*
SPI	*Serial Peripheral Interface*
CAN	*Controller Area Network*
AWS	*Amazon Web Service*
IDE	*Integrated Development Environment*
LED	*Light Emitting Diode*
PIR	*Passive Infrared*
PWM	*Pulse width modulation*
API	*Application Programming Interface*
JSON	*JavaScript Object Notation*
ID	*identity*
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO 12

1.1 OBJETIVOS 13

1.1.1 Objetivo Geral 13
1.2.2 Objetivo Específico 14
1.3 JUSTIFICATIVA 14

2 REFERENCIAL TEÓRICO 15

2.1 INTERNET DAS COISAS 15

2.1.2 Internet das Coisas e Computação em Nuvem 16
2.2 ARQUITETURAS IOT 17

2.3 PILHAS DE PROTOCOLOS TCP/IP 20

2.3.2 MQTT 22
2.4 SISTEMA ANDROID 24

2.5 HARDWARE 25

2.5.1 Framework Zerynth 27
2.6 SOLUÇÕES, SERVIÇOS E RECURSOS DA AWS 27

2.6.1 AWS IoT 29
2.6.2 AWS SDK Para Sistema Android 30
2.7 TRABALHOS CORRELATOS 30

3 MATERIAIS E MÉTODOS 32

3.1 DESENVOLVIMENTO DE HARDWARE 32

3.1.1 Iluminação 32
3.1.2 Alarme 33
3.1.2 Ar condicionado 34
3.1.3 Placa de Controle 34
3.1.4 Maquete 36
3.2 INTERFACE NO SMARTPHONE 38

3.3 ARQUITETURA PROPOSTA 42

3.4 CONFIGURAÇÕES DA PLATAFORMA EM NUVEM 43

3.3.1 AWS Cognito - Configuração Para Conexão do Aplicativo 43
3.3.2 AWS IoT –Configuração para conexão com o hardware 45
3.5 DESENVOLVIMENTO DE FIRMWARE 46

3.5.3 Autenticação na AWS IoT 47
3.5.4 Comunicação por MQTT 48
3.5.5 Tratamento de Mensagens da Sombra da Coisa 48

4 RESULTADOS E DISCUSSÕES 51

5 CONCLUSÃO 57

REFERÊNCIAS 58

Capítulo 1

1 INTRODUÇÃO

O termo Computação em Nuvem (CN), mais popularmente em inglês, "*Cloud Computing*" constitui um dos temas mais abordados na área da Tecnologia da Informação (TI) na atualidade (RAMALHO, 2012). A computação já é uma utilidade e, uma vez mais, as regras econômicas que determinam o modo de trabalhar e viver das pessoas estão sendo reescritas, à semelhança do que ocorreu no passado, com a eletricidade e a telefonia (CHAVES, 2011). A provisão de serviços utilizando CN estabelece que recursos computacionais possam ser contratados conforme demanda, aprimorando a eficiência operacional e reduzindo custos para a organização que adotar esse paradigma (ZHOU, 2010).

Nesse cenário, na busca da melhor relação custo/benefício, as empresas estão adotando a CN para oferecer serviços de confiabilidade e qualidade aos seus clientes, mesclando serviços de provimento próprio com outros adquiridos de terceiros. Isso abre um leque de novas oportunidades e o setor de Automação Residencial (AR) pode ser beneficiado com essa tendência, além de ser bastante atrativo nessa era em que a tecnologia e a Interação Humano-Computador (IHC) está em constante evolução. A AR, também conhecida como domótica, é a tecnologia que visa melhorar a vida das pessoas, seja em tarefas rotineiras ou para tornar o ambiente mais seguro e confortável.

Um sistema de AR é, geralmente, baseado em Internet das Coisas, do inglês *Internet of Things* (IoT) que consiste em conectar o ambiente virtual ao físico por meio da *internet*. A IoT é capaz de proporcionar à objetos do dia a dia, com capacidade computacional e de comunicação, se conectarem à *internet*, tornando-os objetos inteligentes (MANCINI, 2018).

A nova lei do futuro será "Qualquer coisa que pode ser conectada, será conectada". Não é difícil ver como e porque a IoT é um assunto tão importante hoje em dia, certamente abre portas para muitas oportunidades, mas também para muitos desafios (MORGAN, 2014).

Existem dois aspectos bastante atrativos acerca da IoT que é o monitoramento em tempo real e a possibilidade de processamento das informações em nuvem utilizando a CN. Nesse sentido, o trabalho apresentado visa contribuir com a melhor percepção de um sistema

remoto de monitoramento de objetos físicos do dia a dia, como a iluminação de uma residência, um estado de um alarme e até mesmo a climatização do ambiente. No que se trata de processamento das informações, o projeto utiliza de técnicas de processamento na nuvem de dados reais dos dispositivos físicos. Além disso, o sistema proposto terá a capacidade de interação por meio de interface de gerenciamento e interface de usuário.

O decorrer do trabalho busca apresentar, em capítulos, o desenvolvimento do sistema proposto e está dividido da seguinte maneira:

No capítulo 2, Fundamentação Teórica, será apresentado os principais temas estudados para o desenvolvimento do trabalho. É composto pelas principais tecnologias para conectar objetos, uma abordagem sobre o hardware usado, ferramentas para desenvolvimento do aplicativo e trabalhos correlatos.

No capítulo 3, Materiais e Métodos, será discorrido o processo de desenvolvimento do sistema de automação residencial, explicando as principais implementações, decisões de projeto de hardware e software, bem como a arquitetura do sistema.

No capítulo 4, Resultados, será apresentado o protótipo físico da solução, a interface de gerência do dispositivo e a aplicação para dispositivo móvel, assim como os testes realizados e a análise dos dados.

No capítulo 5, Conclusão, será feita uma abordagem crítica do desenvolvimento do trabalho e dos resultados obtidos. Serão propostos aprimoramentos e melhorias ao sistema, a fim de incentivar uma possível evolução do protótipo.

1.1 OBJETIVOS

Nesta seção pode-se esclarecer o objetivo geral do trabalho e os objetivos específicos para se obter o resultado esperado.

1.1.1 Objetivo Geral

O objetivo deste trabalho de conclusão de curso é dispor para o usuário final um sistema de automação residencial capaz de entregar de maneira remota o estado real dos dispositivos eletrônicos monitorados e ainda proporcionar o usuário o controle desses dispositivos.

1.2.2 Objetivo Específico

Neste sentido foram considerados os seguintes objetivos:

- Manter o usuário final informado sobre o estado de sua residência.
- Prover recursos de baixo custo para automação residencial
- Tornar seguras as informações a respeito de automação residencial com serviços em nuvem
- Utilizar plataforma em nuvem capaz de entregar os dados em tempo real.

1.3 JUSTIFICATIVA

Este trabalho justifica-se, no momento, sob o enfoque principal de abordar um cenário completo de IoT, que vai desde de a aquisição dos dados, passa pelo processamento em nuvem e, por fim, a disponibilização para o usuário final.

O projeto proposto aborda um assunto bastante atual e relevante, a IoT, que já é realidade e faz o mundo passar por um processo vertiginoso de hiperconectividade, pois o seu conceito não se restringe apenas a objetos, mas na verdade, se refere a interações inteligentes e quando objetos se conectam entre si, tentem a mudar tudo, incluindo o comportamento humano e cotidiano (OLIVEIRA, 2020). Nesse contexto, o Brasil vem incentivando o avanço do uso da tecnologia através do Plano Nacional da Internet das Coisas que tem como objetivo viabilizar a implementação da tecnologia no país.

O projeto tende a mostrar como um sistema de AR que usufrui de recursos de nuvem é capaz de simplificar as aplicações dos usuários com transparência e segurança, além de manter um sistema altamente escalável no mercado de alta competitividade.

O trabalho também proporciona uma relevante aplicabilidade com uso de uma plataforma relativamente simples, pois o projeto oferece uma intuitiva e prática integração dos sistemas de iluminação, climatização e segurança de uma residência. Uma vez que o aplicativo desenvolvido neste trabalho foi instalado no *smartphone* do usuário, o mesmo pode ter o controle dos dispositivos através da internet.

Capítulo 2

2 REFERENCIAL TEÓRICO

Nesta seção serão abordados os tópicos principais para entendimento técnico das ferramentas usadas no desenvolvimento do trabalho. Grandes autores foram consultados por meio da literatura para que o trabalho fosse desenvolvido da melhor forma possível, pois se tratando de várias áreas envolvidas, o claro entendimento se mostra necessário.

2.1 INTERNET DAS COISAS

A limitação de tempo e a rotina cada vez mais atarefada, aliadas à evolução dos dispositivos móveis e à popularização da "internet móvel" vem criando espaços para que o acesso à internet se dê por maneiras até então pouco exploradas. Como consequência da maior conectividade entre pessoas e dispositivos, demandas por serviços outrora pouco explorados, esse conceito vem aumentando em grande escala (FREITAS, 2016).

O conceito de IoT não é muito atual, pois já se discutia o assunto desde o início do século XXI, onde professores do *Massachusetts Institute of Technology* (MIT) descreveram um mundo onde “coisas” (dispositivos ou sensores) são conectados e capazes de compartilhar dados (BOSWARTICK, 2012). A partir daí até atualmente, esse conceito vem cada vez mais ganhando força e popularidade devido aos diversos tipos de aplicações como em casas inteligentes, cidades inteligentes, indústria 4.0, negócios, etc. Nos últimos anos a humanidade vem promovendo ainda mais a conexão dos dispositivos. Previsões de especialistas indicam que mais de 20 bilhões de dispositivos estarão conectados ao final de 2020, além disso, esse número é esperado que cresça para 75 bilhões em 2025 (Review 42, 2020).

Graças a essa tecnologia é possível imaginar que no ambiente que nos cerca há diversos sensores conectados e enviando dados para um servidor que faz a análise do local em que, como exemplo, um carro se encontra, e por uma série de configurações, interpreta que a pessoa está chegando em casa e aciona o sistema de ar-condicionado, acende a luz do jardim, abre a porta da garagem e manda uma mensagem aos familiares para avisar que a pessoa está chegando. Situações como essas já estão disponíveis hoje e estão cada dia mais acessíveis para fazer parte da realidade.

2.1.2 Internet das Coisas e Computação em Nuvem

Aliado a essa grande tecnologia está a CN, não exatamente como é conhecida hoje, mas em um ponto de sua evolução no qual ela consegue dar suporte à conexão e ao processamento de bilhões de dispositivos e ao tráfego de *terabytes* de dados.

O acelerado desenvolvimento tecnológico pelo qual a IoT é responsável, tem gerado um volume astronômico de dados que devem ser capturados, armazenados e transformados em conhecimento para a humanidade. Nesse cenário, a CN é vista como uma parceira inseparável da IoT, como um fator facilitador desse processo.

Nos últimos anos, houve uma mudança em direção à computação em nuvem. Cerca de 78% das empresas do Reino Unido já possuem alguns de seus serviços de Tecnologia da Informação (TI) na nuvem e 63% têm planos de transferir todas as suas operações para a computação em nuvem antes do final da década (HOSTING, 2016). O uso de recursos em nuvem se torna necessário à medida que a IoT vai se tornando cada vez mais presente na realidade das pessoas. Assim, a CN se mostra ideal para armazenar, processar e garantir a segurança dos dados em tempo real.

É nítida a complementabilidade das duas tecnologias, e esta é a principal razão pela qual muitos pesquisadores têm apostado nessa integração (BALDISSERA, 2017). De acordo com a literatura pesquisada, a CN é o principal recurso disponível de complementação da IoT. Uma comparação é feita no quadro 1 a seguir.

Quadro 1 - Comparação entre IoT e CN.

	Internet das Coisas	**Nuvem**
Deslocamento	Infiltrado	Centralizada
Acessibilidade	Limitado	Ubíquo
Componentes	Coisas do mundo real	Recursos Virtuais
Capacidades computacionais	Limitado	Virtualmente ilimitado
Armazenamento	Limitado ou nenhum	Virtualmente Ilimitado
Papel da Internet	Ponto de convergência	Prestação de Serviços
Big Data	Fonte de informações	Recursos para gerenciar

Fonte: Julia Baldissera

Dentre os benefícios da CN aliada à IoT está a escalabilidade, característica de grande importância em um sistema complexo que está em plena evolução. É importante, pois um sistema de AR, por exemplo, pode nascer simples e ir ganhando mais funcionalidades de acordo com a necessidade do usuário que o adquirir, pois a cada momento surgem novos produtos nesse mercado.

2.2 ARQUITETURAS IOT

Uma parte importante para realização da IoT é a sua arquitetura, que se refere às infraestruturas de alto nível para a integração com a CN e os componentes que definem suas respectivas funções. Tratando-se de IoT e CN, vale ressaltar que não há meios padronizados para apoiar o design arquitetônico de soluções que envolva essas duas tecnologias (BALDISSERA, 2017), o que existe é uma gama de soluções prontas no mercado à disposição das empresas e dos usuários.

Na pesquisa realizada por Ray (2017), que apresenta 26 plataformas de nuvem IoT de acordo com seus domínios e aplicações específicas, em seu trabalho, o autor testa cada uma na realidade, o qual elaborou alguns pontos fortes e fracos para cada uma delas. Conforme afirma, existem inúmeras plataformas de IoT no mercado, mas, mesmo assim, o número de plataformas apresentadas na pesquisa é razoável. Embora as 26 plataformas tenham sido estudadas, nenhuma delas é perfeita em todos os termos e aspectos para os desenvolvedores.

Com diversas plataformas disponíveis e as soluções de IoT evoluindo a cada dia de maneira mais veloz, muitos desafios relacionados sobre o tema acabam surgindo. Segundo Cavalcante et al. (2016), a literatura ainda carece de uma visão abrangente sobre o que foi investigado a respeito de IoT e CN. Mesmo assim, esses desafios vêm sendo tema de muitas pesquisas no mundo científico para aprimorar esse cenário. Dentre os principais desafios estão:

- Segurança e privacidade: O sistema em questão é exposto a possíveis ataques e os dados dos usuários podem estar vulneráveis.
- Heterogeneidade: A integração dos dispositivos ainda não é tão simples para algumas plataformas, pois os serviços em nuvem possuem, em boa parte, interface prioritária.

- Desempenho: Os serviços deveriam ocorrer com alta reatividade, mas a evolução da banda larga não tem acompanhado a evolução da IoT e CN.
- Confiabilidade: Em aplicações críticas, por exemplo, em veículos inteligentes, existe sempre uma incerteza relacionado a falhas dos dispositivos (BOTTA, 2016).
- Big Data: Com muitos dispositivos conectados, a manipulação, análise e integridade dos dados são importantes fatores para uma boa solução de IoT e CN.

As arquiteturas existentes são bastante variadas, com diferentes funcionalidades e número de camadas. Do ponto de vista geral, existem dois tipos básicos de arquiteturas: Uma vertical e outra horizontal.

Nas estruturas verticais, aplicadas atualmente na maioria dos sistemas de IoT, os dispositivos são tipicamente nós de sensores e atuadores conectados à *internet* diretamente ou por meio de um *gateway* local, onde o *gateway* é o equipamento responsável por fazer a ligação de duas redes distintas. Elas têm fácil implementação e reduzem problemas de compatibilidade, pois estão vinculadas a serviços prioritários, geralmente de um único provedor. Uma desvantagem dos sistemas verticais é que o usuário final depende do fornecedor para melhorias, aprimoramento e atualizações. Este modelo é o de mais fácil comercialização, de modo que foi adotado por várias *startups* de IoT focadas em *Business-to-Customer* (B2C), ou seja, com foco em fornecer soluções para o consumidor final.

Por outro lado, a estrutura horizontal é um conceito mais moderno e pode impulsionar o crescimento e a inovação disruptiva, garantindo alta modularidade, escalabilidade e personalização. Por essa razão, a abordagem horizontal é preferível em casos *Business-to-Business* (B2B) onde as soluções IoT são apenas a infraestrutura técnica por trás de um produto inteligente projetado por uma empresa (MAZZEI et al., 2016).

A figura 1 mostra o diagrama dos dois tipos de arquitetura IoT existentes. Na arquitetura vertical, é possível notar que apenas um provedor é responsável de ponta a ponta da solução, o que facilita a integração dos *hardwares*. Um exemplo dessa arquitetura é o modelo da empresa Solar.io mostrado na figura 2.

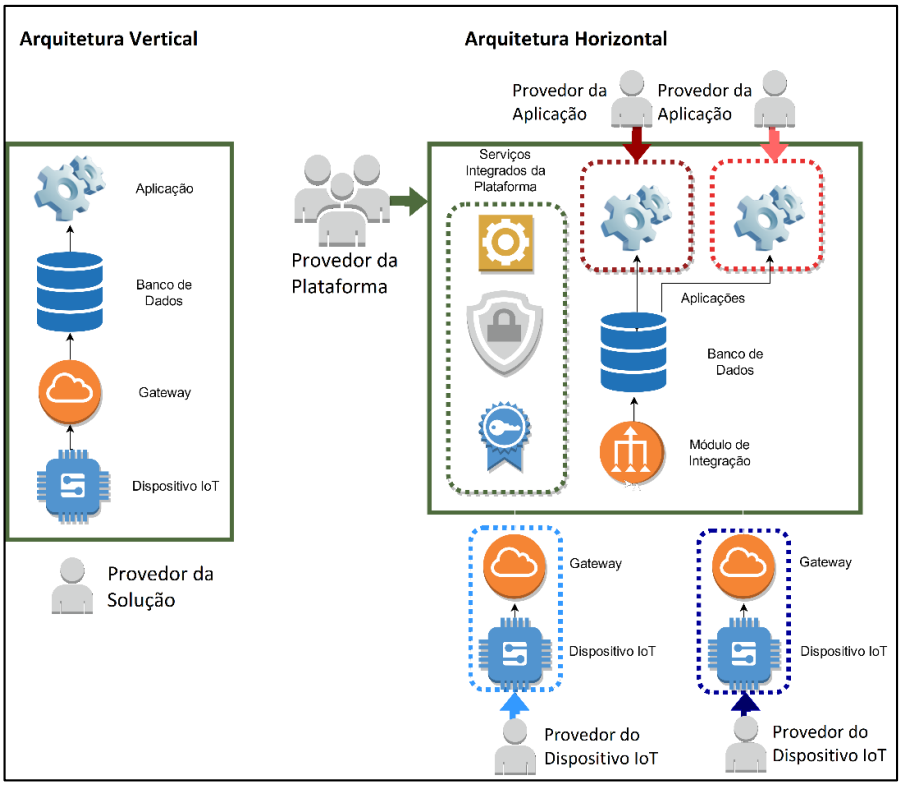

Figura 1 - Modelos de arquiteturas vertical e horizontal

Fonte: (Guimarães, 2017)

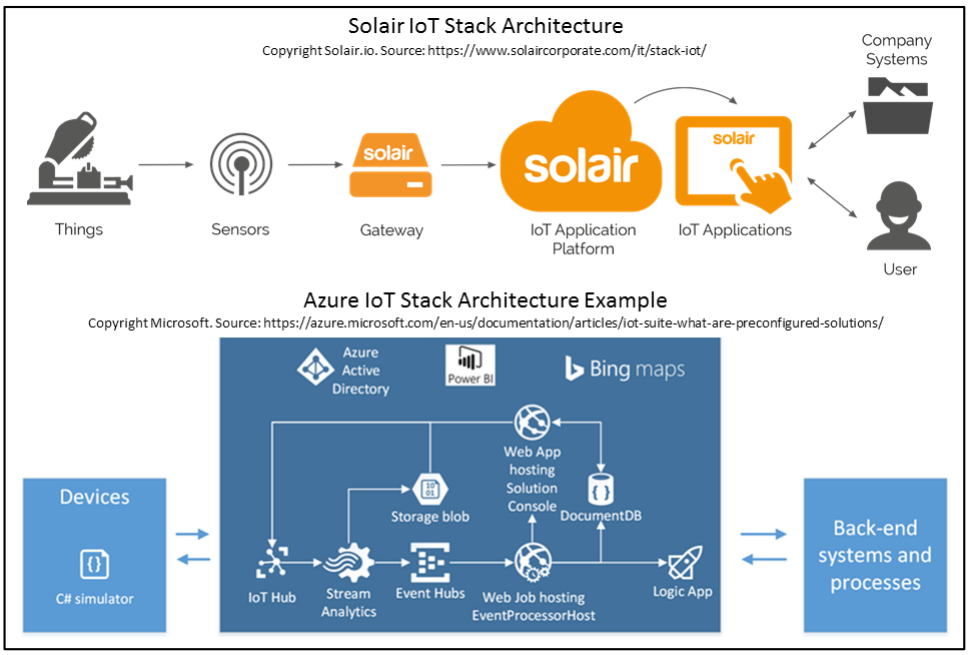

Figura 2 - Exemplos de arquiteturas horizontal e vertical

Fonte: www.fuiasjdija.com.br

O diagrama da arquitetura horizontal também é mostrado na figura 1 e pode-se notar que a solução final é algo amplo e que pode abranger diversos tipos de provedores. A abordagem desse tipo de arquitetura é em direção a uma solução totalmente baseada em nuvem para que o controle dos dispositivos remotamente seja de forma simples (MAZZEI, et al, 2016). Um exemplo desse modelo de arquitetura é mostrado também na Figura 2, a plataforma da Microsoft Azure.

Mazzei (2016) afirma que as soluções do mercado podem ser agrupadas em duas categorias, de acordo com a abordagem: soluções rodando em um hardware prioritário ou rodando em hardware genérico. Soluções sob medida para hardware proprietário são, por exemplo, Particle e Onion. O Particle fornece uma família de placas baseadas em microcontroladores (MCU) que estão prontas para serem conectadas a uma nuvem proprietária, permitindo um rápido desenvolvimento de protótipos de produtos inteligentes usando uma linguagem C similar ao Arduino. Onion produz uma placa baseada em microprocessador (MPU) com conexão Wi-Fi, onde uma versão minúscula do sistema operacional Linux é executável, permitindo o desenvolvimento de soluções de IoT usando programação de alto nível como Python, Ruby on Rails e Java.

Solução baseada em software capaz de executar em várias plataformas de *hardware* são, por outro lado, Micropython e Zerynth. Micropython é um interpretador da linguagem de programação Python executável por MCUs que permite um desenvolvimento rápido e fácil do aplicativo incorporado na linguagem Python. Os próprios autores desenvolveram em 2014 o VIPER que é uma Máquina Virtual multiplataforma para MCUs que permite a programação Python de aplicativos executando em tempo real em dispositivos MCU e com um baixo uso de memória.

2.3 PILHAS DE PROTOCOLOS TCP/IP

A pilha de protocolos TCP/IP é o conjunto de protocolos usados para envio e recebimento de dados pela internet. Integram esse conjunto o protocolo TCP, do inglês, *Transmission Control Protocol* (TCP) e o protocolo IP, do inglês,*Internet Protocol* (IP). A pilha de protocolo TCP/IP é composta por quatro camadas: aplicação, transporte, *internet* e interface de rede. Em comparação com um modelo equivalente, podemos citar o modelo

OSI(em inglês, *Open Systems Interconnection*), que possui sete camadas. A Figura 3 apresenta a relação das camadas que compõem os dois modelos.

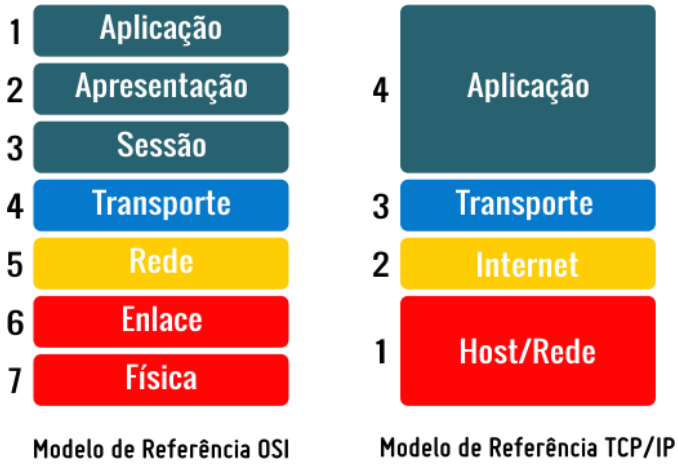

Figura 3 - Modelo OSI e TCP/IP

Fonte: (SANTOS, 2017)

Cada camada é responsável por executar tarefas distintas para garantir a integridade dos dados trafegados na rede, fornecendo um conjunto de serviços para o protocolo da camada superior. As camadas mais altas estão mais próximas do usuário e lidam com dados passados por camadas de nível mais baixo.

A camada de Aplicação é o nível mais alto da pilha, é a camada que convivemos no dia a dia quando lidamos com redes, ela é responsável por tratar as requisições dos programas invocados pelos usuários para executar tarefas na rede. Contém protocolos como *Hipertex Transfer Protocol* (HTTP), *Message Queue Telemetry Transport* (MQTT), *Terminal Virtual*(TELNET), etc.

A camada de transporte é equivalente para os dois modelos comparados anteriormente, tem a função de realizar a troca de informações entre programa, essa comunicação é conhecida também como comunicação de ponta a ponta. Os protocolos nessa camada podem resolver problemas de confiabilidade, assim, garantindo que os dados cheguem ao seu destino com integridade. O protocolo TCP é o mais usado nessa camada, ele pode dividir o fluxo de dados em partes menores, chamadas de pacotes, e reagrupá-los. A cada pacote é associado um cabeçalho, que contém a porta de origem e a de destino, número de sequência, entre outras informações. As portas de origem e destino contêm os números de porta TCP, cuja função é identificar a aplicação (JUNIOR, 2017). Por exemplo, o HTTP

utiliza a porta 80 e o MQTT utiliza a porta 1883. Essas características definem o protocolo TCP como um protocolo orientado a conexão.

A camada de internet é equivalente a camada de rede do modelo OSI, também conhecida pelo nome de interredes, tem como principal protocolo o IP. Este protocolo é responsável por permitir que *hosts* enviem pacotes a qualquer rede e garantam que os mesmos trafeguem até o seu destino, definindo caminhos entre roteadores e *hosts* e processos para os pacotes. Ele adiciona nos pacotes que vêm da camada de transporte um cabeçalho, conhecido como endereço IP. Isso possibilita a comunicação de um *host* com outro em qualquer rede do mundo.

Finalmente a camada de *host*/rede, também conhecida como camada de acesso à rede, corresponde às camadas de enlace e física do modelo OSI, camadas que trabalham os aspectos físicos da comunicação. O que define essa camada é o tipo de rede física que está sendo usada, normalmente é usado a rede *Ethernet* ou *WiFi*. O protocolo não é definido e varia de rede para rede e de *host* para *host*, a única exigência é que o *host* se conecte à rede usando algum protocolo capaz de enviar pacotes IP. A *Ethernet*, por exemplo, é composta por três camadas, *Logic Link Control* (LLC), *Media Access Control* (MAC) e a Física. A camada LLC tem a função de direcionar os pacotes recebidos da rede para os protocolos da camada Internet. A camada MAC monta o quadro que será transmitido pela rede, este quadro contém os endereços MAC da fonte e do destino. Tal endereço MAC está associado à placa de rede de cada computador ou ao módulo de rede em casos de MCUs com suporte a esse tipo de conexão. A última camada, a Física, é responsável por converter o quadro gerado na camada MAC em sinais elétricos (FALL, 2012).

2.3.2 MQTT

Para os dispositivos de Internet das Coisas (IoT), a conexão com a Internet é um requisito. É necessário que haja uma padronização de comunicação, o *Message Queue Telemetry Transport* (MQTT) tornou-se o padrão para comunicações de IoT e é o protocolo que será utilizado neste trabalho.

O MQTT foi inventado e desenvolvido inicialmente pela IBM no final dos anos 90. Sua aplicação original era vincular sensores em *pipelines* de petróleo a satélites. Como seu

nome sugere, ele é um protocolo de mensagens com suporte para a comunicação assíncrona entre as partes. Um protocolo de sistema de mensagens assíncrono desacopla o emissor e o receptor da mensagem tanto no espaço quanto no tempo e, portanto, é escalável em ambientes de rede que não são confiáveis. Apesar de seu nome, ele não tem nada a ver com filas de mensagens, na verdade, ele usa um modelo de publicação e assinatura, do inglês, *publish* e *subscribe*. No final de 2014, ele se tornou oficialmente um padrão aberto OASIS, com suporte nas linguagens de programação populares, usando diversas implementações de software livre (YUAN, 2017).

O MQTT opera em modelo cliente/servidor, em que cada dispositivo IoT é um cliente e se conecta a um servidor, conhecido como *broker*, via protocolo TCP. O *broker* recebe todas as mensagens dos clientes e depois roteia essas mensagens para os endereços relevantes conhecidos como tópicos. Um cliente pode ser qualquer coisa que pode interagir com o *broker*, dessa forma os clientes podem assinar os tópicos e receber qualquer mensagem publicada nesse tópico, além do mais, cada cliente pode se assinar em múltiplos tópicos. A Figura a seguir mostra o funcionamento de um sistema de configuração de um sensor.

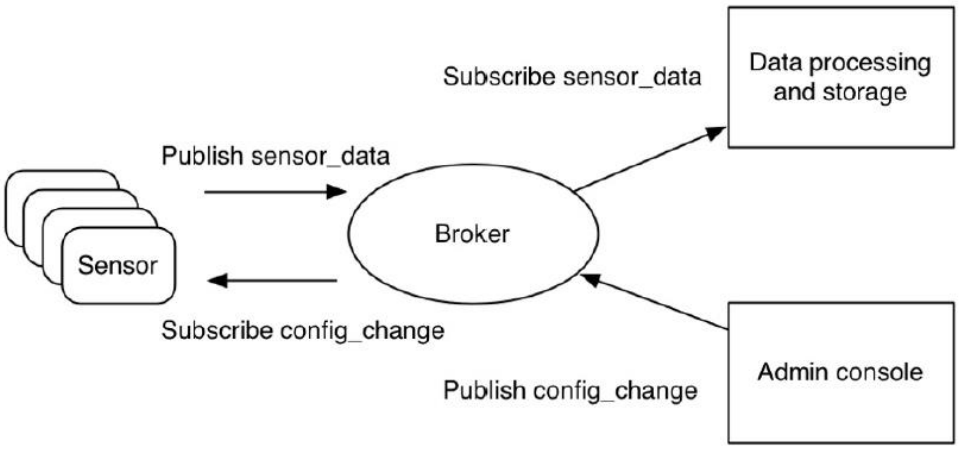

Figura 4 - Modelo de publicação/assinatura de exemplo

Fonte: (YUAN, 2017)

O fluxo de comunicação pode ser visualizado em três pontos.

1) O cliente se conecta ao *broker* e pode assinar qualquer tópico de mensagem no *broker*. Essa conexão pode ser via TCP/IP simples ou via TLS com criptografia.

2) O cliente publica as mensagens em um tópico, enviando a mensagem e o tópico ao *broker*.
3) Em seguida, o *broker* encaminha a mensagem a todos os clientes que assinaram esse tópico.

Como as mensagens do MQTT são organizadas por tópicos, o desenvolvedor de aplicativos tem a flexibilidade de especificar que determinados clientes somente podem interagir com determinadas mensagens. Por exemplo, na figura 4, os sensores publicarão suas leituras no tópico "*sensor_data*" e assinarão o tópico "*config_change*". Os aplicativos de processamento de dados que salvam os dados do sensor em um banco de dados de *backend*, assinarão o tópico "*sensor_data*". Um aplicativo de console administrativo poderia receber comandos do administrador do sistema para ajustar as configurações dos sensores, como a sensibilidade e a frequência de amostragem, e publicar essas mudanças no tópico "*config_change*".

2.4 SISTEMA ANDROID

O *Android* é uma plataforma de desenvolvimento para aplicativos móveis como *smartphones* e *tablets* que conta com uma interface rica e repleta de diversas aplicações já instaladas e ainda um ambiente de desenvolvimento bastante poderoso, inovador e flexível (LECHETA, 2013).

O sistema operacional *Android* é um sistema *linux* multiusuário em que cada aplicativo é um usuário diferente. Este sistema operacional é responsável por gerenciar todos os processos e memória do dispositivo móvel, assim como os arquivos e recursos como rede e *drivers* (ANDROID, 2019).

Os aplicativos que rodam em *Android* geralmente são programados em Java. O kit de desenvolvimento, do inglês, *software development kit* (SDK) é um conjunto de ferramentas necessárias para criar aplicações para determinado pacote de *software*, *framework*, plataforma de *hardware*, sistema operacional, etc. Normalmente essas ferramentas são fornecidas por empresas ou projetos de plataforma aberta, do inglês, *open source* para que programadores externos possam desenvolver aplicações integrando o software proposto (BURNETTE, 2008).

As ferramentas SDK compilam o código em um *Android Application Pack* (APK), que é um arquivo de sufixo ".apk". O APK é formado do conteúdo da aplicação e são usados para instalar nos dispositivos com sistema *Android*. É possível emular os dispositivos *android* durante o desenvolvimento da aplicação. Existem diversos ambientes de desenvolvimento integrado (IDE) para o desenvolvimento de aplicativos *android*, como o *android studio*, *eclipse*, etc.

Dentre os conceitos já citados aqui, está o conceito de *Framework* que é uma estrutura de suporte definida em que um projeto de *software* pode ser organizado e desenvolvido (SODRE, 2011). Um *framework* pode incluir diversos recursos para agilizar no desenvolvimento do *software* para aplicativos e até *firmware* para microcontroladores.

Um *framework* pode incluir programas de suporte, bibliotecas de código, linguagens de *script* e outros *softwares* para ajudar no desenvolvimento (SODRE, 2011). Uma vantagem dos *frameworks* é que eles proporcionam uma gama de possibilidades de integrar diferentes linguagens à diferentes plataformas de *hardware* e *software*.

2.5 HARDWARE

O item do referencial teórico sobre o hardware trará conhecimento das tecnologias utilizadas para desenvolver o hardware de controle dos dispositivos na maquete da residência que tem como principal componente a placa de desenvolvimento NodeMCU-32S mostrada na Figura 5, esta placa é baseada no microcontrolador ESP32.

Figura 5 - Placa de desenvolvimento NodeMCU-32S

Fonte: (www.filipeflop.com/)

O ESP32 é um microcontrolador fabricado pela *Espressif Systems*. De acordo com a documentação (ESPPRESSIF, 2018) é um microcontrolador de baixa potência *dual core Tensilica Xtensa 32-bit* LX6 com suporte embutido à rede *WiFi*, *bluetooth* versão 4.2 e é largamente utilizado em aplicações que envolvem IoT. Além disso, possui outros recursos como processador *dual core* de 240MHz, memória RAM de 512kB, memória *flash* integrada de 4MB, 32 pinos de entrada e saída, mais conhecido como GPIO, do inglês, *General Purpose Input Output* (GPIO), conversor digital analógico (ADC) de 12 bits. Isso faz do ESP32 um microcontrolador altamente potente e portátil, considerando que todos esses recursos se encontram em um pequeno módulo, o ESP-WROM-32 mostrado na Figura 6.

Figura 6 - Módulo ESP-WROOM-32

Fonte: https://uploads.filipeflop.com

O microcontrolador possui 24 GPIO livres, 18 entradas A/D e duas saídas D/A, como mostra a Figura 7 com a descrição dos pinos da placa de desenvolvimento. Por possuir memória *flash* externa, alguns pinos são exclusivos para uso desta. O módulo também possui pinos para comunicação serial, I2C, I2S, SPI, CAN 2.0, sensor Hall, etc.

Para que o microcontrolador execute suas principais funções em um projeto é necessário que seja gravado o *firmware* que é um programa de *software* escrito para interpretar e controlar recursos de *hardware*. Geralmente, para se escrever o *firmware* e até mesmo gravá-lo em um microcontrolador, utiliza-se de um *framework*.

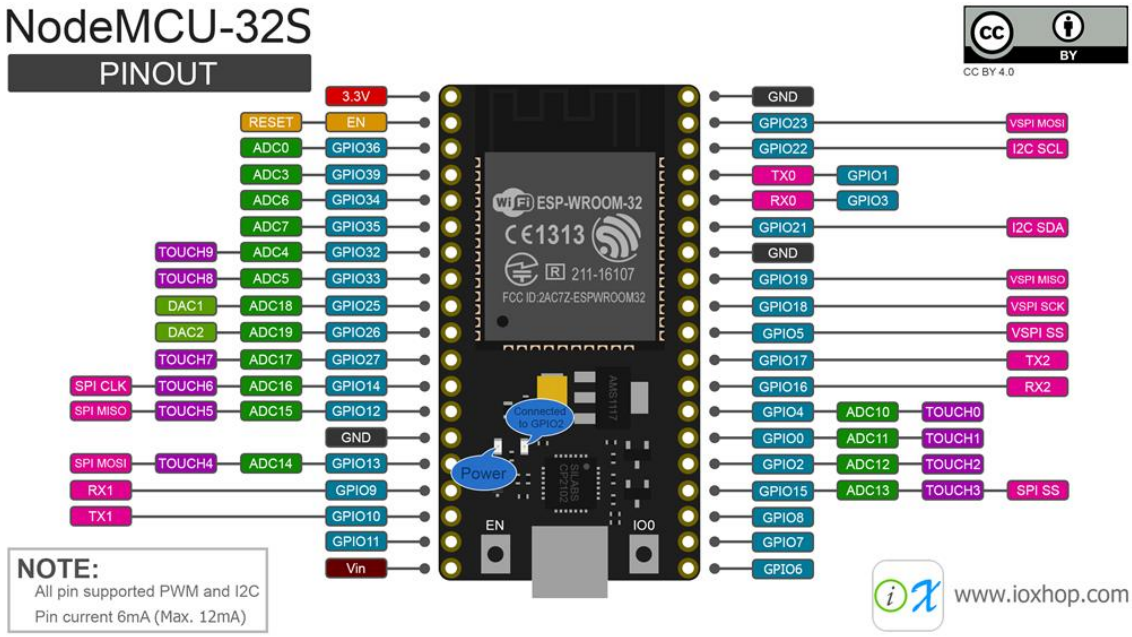

Figura 7 – Descrição dos pinos da placa de desenvolvimento NodeMCU-32S

Fonte: https://www.ioxhop.com

2.5.1 Framework Zerynth

Zerynth é uma plataforma de desenvolvimento de aplicações para interação com objetos de internet das coisas. É composto por um conjunto de ferramentas de código aberto que permitem interagir com microcontroladores e soluções de nuvem em poucos cliques. Seus componentes podem ser usados em sistema Windows, Linux e MacOS.

Seus principais componentes são: Zerynth OS, que é um sistema operacional que roda em tempo real; Zerynth SDK que é um kit de ferramentas da plataforma que inclui o Zerynth Studio e Zerynth SDK. Através do Zerynth Studio pode-se desenvolver a programação do *firmware* em *python* e compilar aplicações para executar no microcontrolador com Zerynth OS.

2.6 SOLUÇÕES, SERVIÇOS E RECURSOS DA AWS

A Amazon Web Services (AWS) é a plataforma de soluções em nuvem da Amazon. Tem como características principais: ser segura, abrangente em todo o mundo e possuir

diversos recursos em seus serviços. Podemos encontrar soluções em banco de dados, *machine learning*, inteligência artificial, análise e internet das coisas (AMAZON, 2020).

Projetada para desenvolvedores criar e gerir qualquer tipo de aplicação, a AWS hoje possui mais de 175 serviços disponíveis. Para o melhor entendimento, uma solução pode ser definida por um conjunto de serviços que por sua vez possui um conjunto de recursos com características distintas entre si, mas com funcionalidades que se complementam. Na figura 8é mostrado os serviços para Internet das Coisas, o AWS IoT.

Como exemplo, temos a solução para internet das coisas chamado AWS IoT que é composta por serviços como o AWS IoT Core, AWS Iot Analytics, AWS IoT Device Management, etc. Dos quais têm finalidades de conexão centralizada dos dispositivos IoT, análise dos dados e gerenciamento dos dispositivos, respectivamente.

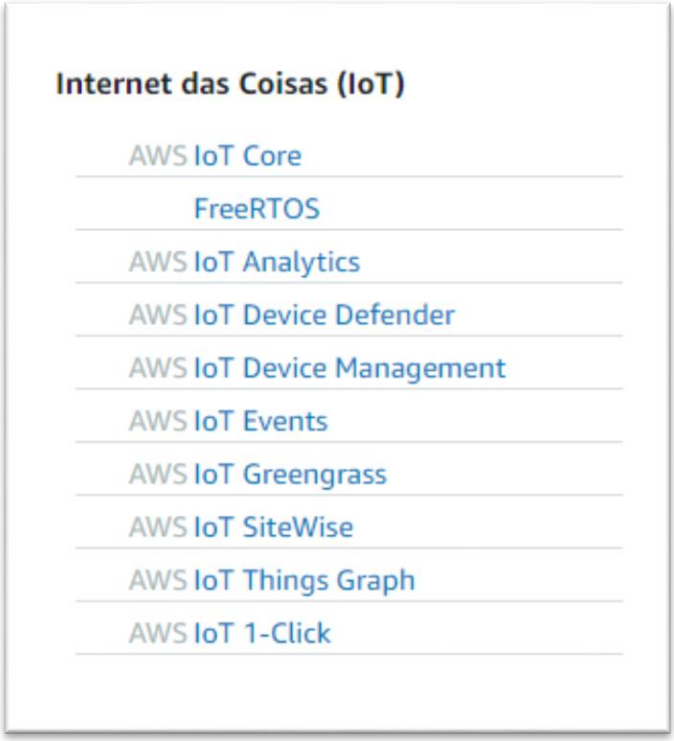

Figura 8 - AWS: Serviço de Internet das coisas e seus recursos

Fonte: https://docs.aws.amazon.com

2.6.1 AWS IoT

O serviço AWS IoT é responsável por possuir os recursos necessários para conectar os dispositivos de internet das coisas a outros dispositivos ou à serviços da AWS como ilustrado no diagrama da figura 6.

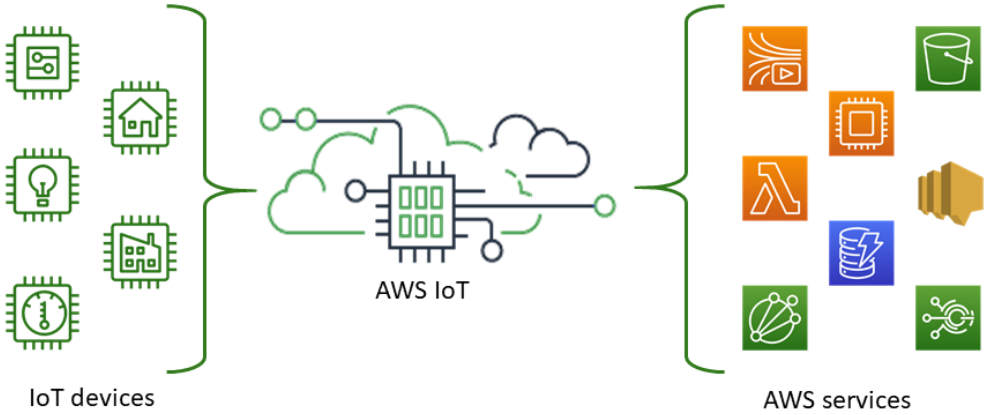

Figura 9 - Diagrama de Conexão do AWS IoT

Fonte: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot

Um dos principais serviços de internet das coisas é o AWS IoT Core que fornece os recursos que possibilitam a interação das aplicações com os dispositivos conectados à internet. O diagrama desse serviço é mostrado na figura 9. É possível notar que os dispositivos se conectam diretamente à um *broker* que por sua vez está conectado a outros serviços da nuvem. O *broker* entrega as mensagens de um dispositivo para todos os outros dispositivos e serviços que estejam subscritos nos tópicos correspondentes de forma segura.

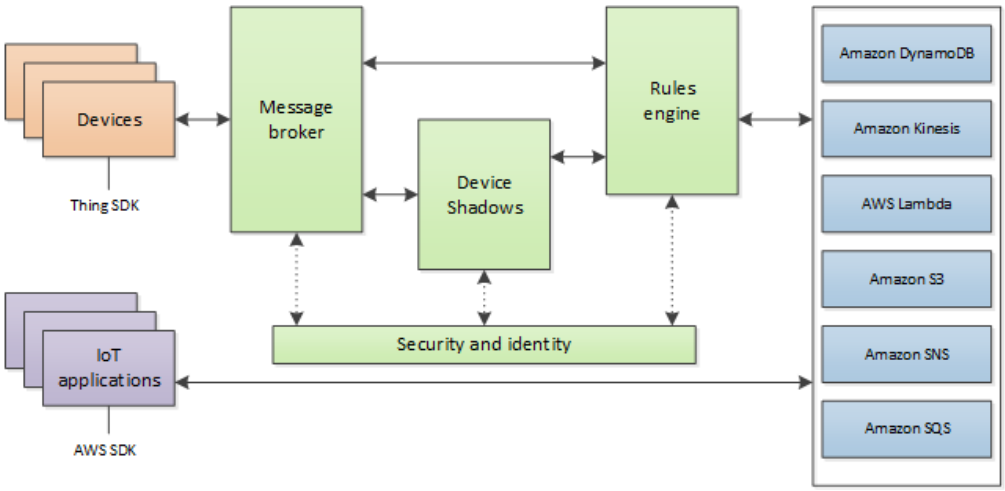

Figura 5 - Diagrama de conectividade dos serviços do AWS IoT Core

Fonte: https://docs.aws.amazon.com/pt_br/iot/latest/developerguide/aws-iot-how-it-works

Outro serviço não menos importante é o Sombra da Coisa, do inglês, *Device Shadows* que se trata de uma forma de disponibilizar na nuvem o estado de um dispositivo para outros aplicativos e serviços, independentemente de o dispositivo estar conectado ao AWS IoT ou não.

2.6.2 AWS SDK Para Sistema Android

O AWS SDK para Android fornece uma biblioteca e documentação para desenvolvedores criarem aplicativos móveis conectados usando o AWS. Essa biblioteca é composta por diversos recursos de programação que auxiliam o desenvolvedor (AMAZON, 2020).

Os recursos com suporte pela biblioteca são autenticação, *analytics*, armazenamento, notificação por *push*, *pubsub*, entre outros. Diversos exemplos são disponibilizados pela AWS para uso do desenvolvedor, por exemplo, aplicações para transmitir vídeo ou fazer *download* e *upload* de arquivos ou até mesmo assinar e publicar em tópicos do serviço AWS IoT.

Para instalar e usar um aplicativo de exemplo é necessário instalar ou possuir uma IDE, baixar a SDK da AWS, depois importar o projeto exemplo, preencher alguns campos e gerar um arquivo ".apk". A aplicação gerada deve promover a interação dos dispositivos móveis com os recursos de CN da AWS.

2.7 TRABALHOS CORRELATOS

Atualmente existem vários trabalhos sobre automação residencial, muitos deles focados na parte de monitoramento e controle e, assim como este, usando hardware de baixo custo. Isso evidencia uma viabilização para um sistema de automação residencial com esses aspectos. Alguns destes trabalhos são citados a seguir.

O trabalho de Junior (2017) utiliza um sistema de monitoramento residencial de baixo custo utilizando o módulo WiFi ESP8266 e usa como protocolo de comunicação o MQTT. Ele apresenta um trabalho com foco em segurança, onde monitora sensores de temperatura e *reed switches* nas portas e janelas da residência. Diferentemente do modelo dele que utilizou o servidor MQTT localmente, neste trabalho foi utilizado o servidor MQTT na nuvem. Assim como no trabalho dele que foi utilizado o *smartphone*, aqui também foi usado para o controle

e visualização dos dados. Por fim, ele enfatiza o baixo custo do sistema e aponta o potencial do módulo WiFi ESP8266.

Em outra proposta, Andrade (2018) apresenta um adaptador de tomada inteligente aplicado para automação residencial, no qual ele também usa o módulo ESP8266 com WiFi conectando através do AWS IoT, assim como este trabalho. Ele também usa uma conexão segura, fazendo uso de autenticação na plataforma e certificados como também foi usado na solução aqui apresentada.

Abaixo é possível visualizar uma tabela resumida comprando o trabalho aqui apresentado com o trabalho de outros dois autores.

Tabela 01 – Comparação com trabalhos de Junior e Andrade.

ITEM	**JUNIOR**	**ANDRADE**	**FRANCISCO**
CUSTO	Baixo	Baixo	Baixo
MICROCONTROLADOR	ESP8266	ESP8266	ESP32
PROTOCOLO DE COMUNICAÇÃO	MQTT	MQTT	MQTT
ÁREA	Segurança	Consumo elétrico	Controle por aplicativo
HOSPEDAGEM SERVIDOR (BROKER)	Local	Nuvem	Nuvem
SMARTPHONE	Sim	Não	Sim
APLICAÇÃO MQTT	*Mosquitto*	AWS IoT	AWS IoT
SEGURANÇA (AUTENTICAÇÃO)	Não	Sim	Sim

Capítulo 3

3 MATERIAIS E MÉTODOS

Neste capítulo será apresentada detalhadamente a metodologia de desenvolvimento do sistema de automação residencial proposto neste trabalho de conclusão de curso. Na seção 3.1 será explicado o desenvolvimento de *hardware*, assim como seus esquemas elétricos para controle dos dispositivos. Em seguida, na seção 3.2 será abordado o desenvolvimento da interface em aplicação *Android*, detalhando os pontos importantes. Após detalhar os itens de bordas, será apresentado na seção 3.3 a arquitetura proposta. Na seção 3.4 será explicado a configuração da plataforma em nuvem, bem como todas as configurações no Cognito e AWS IoT. A seção 3.5 ou última seção aborda os principais pontos do desenvolvimento do *firmware* para a placa de controle.

3.1 DESENVOLVIMENTO DE HARDWARE

Os dispositivos explicados a seguir têm o intuito de dar capacidade ao sistema de simular controle e monitoramento de iluminação, alarme e liga e desliga de ar condicionado. Cada um desses dispositivo é descrito nas seções seguintes.

Sendo assim, o projeto de desenvolvimento do *hardware* foi dividido em quatro partes:

- Iluminação
- Alarme
- Ar condicionado
- Placa de Controle

3.1.1 Iluminação

Para simular uma carga de iluminação foi usado diodos emissores de luz, do inglês, *Light Emitting Diode*, mais conhecido simplesmente como LED. Esse componente apresenta o comportamento similar a uma lâmpada convencional usada em residências, por esta razão foi utilizado no trabalho.

Em cada circuito de iluminação existe um ponto de atuação e leitura para controle e monitoramento do estado do LED, respectivamente. O LED 1 simula a luz do térreo, o LED 2 simula a luz do teto, o LED 3 simula a luz da fachada e o LED 4, a luz da mesa. Todos podem ser observados na Figura 6.

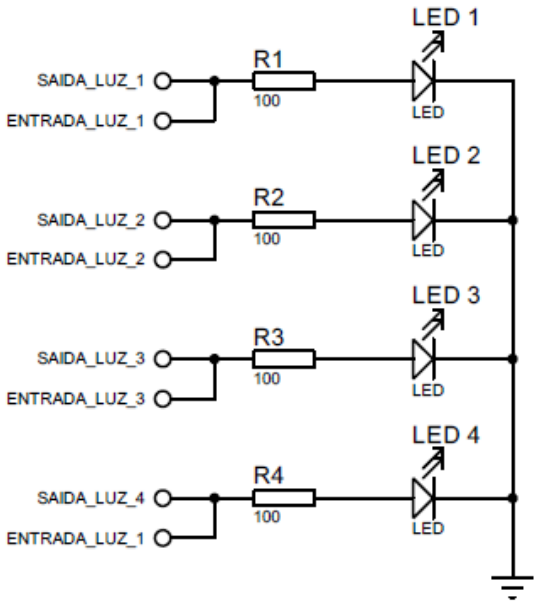

Figura 6 - Circuito de iluminação com Led

Fonte: Autor

3.1.2 Alarme

Os dispositivos usados para simular um alarme foram um LED na cor vermelha, um *buzzer*, um botão do tipo *push* e um sensor infravermelho passivo, também conhecido como sensor PIR, do inglês, *passive infrared*. Através do botão é possível armar e desarmar o alarme além de desligar caso esteja disparado. O modelo do sensor PIR utilizado nesse projeto foi o DYP-ME003 que permite ajuste de sensibilidade e tempo de ativação do sensor. O PIR serve para detectar movimento na entrada lateral da residência simulada no trabalho, disparando assim o alarme que faz o sinal sonoro ativar e o LED vermelho piscar. A Figura 7 apresenta o sensor PIR utilizado. O *buzzer* por sua vez é do tipo DC, que aciona com nível lógico alto e não com ondas PWM.

Figura 7 - sensor de movimento/presença

Fonte: (FILIPEFLOP, 2018)

3.1.2 Ar condicionado

O sistema de ar condicionado é simulado por um *cooler* de resfriamento. No circuito acrescentou-se um transistor Q1 para chavear a energia para o *cooler*, visto que a GPIO do microcontrolador não seria suficiente para fornecer a corrente para o fazer girar. Foi utilizado o transistor de modelo BC546. No intuito de evitar possíveis correntes reversas no transistor, o diodo D1 foi implementado em paralelo com o motor como mostra a Figura 8. Também como no circuito do LED, terá um ponto para controle e monitoramento do seu estado.

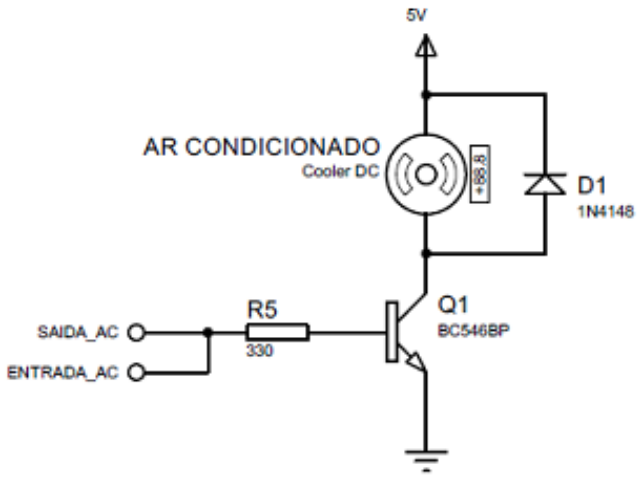

Figura 8 - Circuito de Acionamento do Cooler de Resfriamento

Fonte: Autor

3.1.3 Placa de Controle

Como componente central para controle dos dispositivos foi utilizado a placa de desenvolvimento NodeMCU-32S, que é compatível com o Arduino, ela foi escolhida por ser

ideal para projetos de automação que utilizam a internet das coisas, pois possui suporte embutido à rede *Wi-Fi*.

A figura 9 mostra a placa NodeMCU-32S escolhida que foi utilizada basicamente para conectar o *hardware* com a nuvem, ela se conecta pela internet aos serviços em nuvem da Amazon AWS através da biblioteca padrão que o *framework* chamado Zerynth possui para a plataforma AWS IoT e por meio disso controlar o *hardware* do projeto através da nuvem.

Figura 9 - Placa de desenvolvimento NodeMCU-32S

Fonte: Autor

O NodeMcu-32S foi escolhido entre diversas placas do seu tipo por ser voltado para aplicações de IoT e possuir os recursos necessários para o projeto proposto. Um diferencial do microcontrolador é que ele possui várias GPIO e grande capacidade de processamento. É compatível com diversos *frameworks* do mercado e por ser um microcontrolador de 32 bits, pode ser programado em linguagens de alto nível no intuito de agilizar o processo de conectividade com a nuvem.

A programação foi feita em linguagem Python em uma IDE do *software* chamado "Zerynth Studio" que faz parte de um *framework* que permite a programação e gravação de *firmware* feito em linguagem c e/ou Python e possibilita uma fácil conexão com a nuvem. A escolha dessa linguagem de programação e do *framework* foi feita com vista a facilitar a conectividade do hardware com a plataforma em nuvem.

Mas especificamente, a linguagem Python foi escolhida para o *firmware*, pois seu uso se mostra muito promissor para aplicação em internet das coisas. Com ela pode-se programar sistemas mais complexos em menos linhas de código e sua gerência de *hardware* é mais eficiente em comparação a linguagem C para microcontroladores (SOUZA, 2012).

A placa foi conectada aos periféricos por meio de cabos elétricos e conexões de *protoboard*, como mostra a Figura 10.

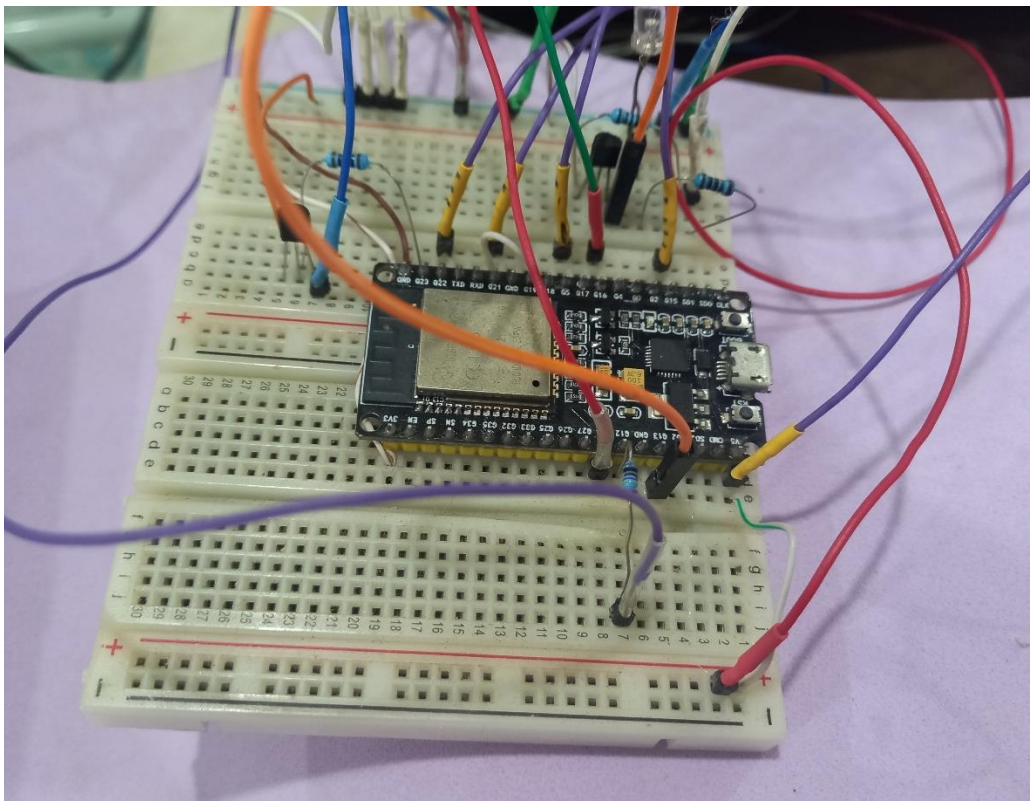

Figura 10 - Conexão elétrica do microcontrolador em *protoboard*

Fonte: Autor

3.1.4 Maquete

Além do *software* e do *hardware* envolvidos no protótipo, foi adquirida e montada uma maquete artesanal que simula uma residência. A maquete foi utilizada para distribuir os diversos sensores e atuadores com o propósito de tornar mais realístico o ambiente automatizado. A Figura 11 mostra a estrutura da maquete e seus respectivos ambientes como o andar superior, térreo e entrada lateral. No térreo ficou um ponto de iluminação e na parte interna foi colocado o sensor de movimento para o alarme. No andar superior ficou os três pontos de iluminação, o *buzzer* e a ventoinha do ar condicionado.

Figura 11 - Maquete escolhida para montagem dos dispositivos

Fonte: Autor

A Figura 12 mostra também os componentes internos simulando móveis de decoração. Ainda na imagem, é possível visualizar as mesas e cadeiras, o balcão de doces e a estante, todas elas para montar internamente à maquete.

Figura 12 - Itens de decoração da maquete

Fonte: Autor

3.2 INTERFACE NO SMARTPHONE

O *smartphone* é utilizado como interface de usuário para interação com o sistema por meio de conexão com a nuvem, enviando e recebendo dados para comandos e visualização dos dados, respectivamente. Para isso, é necessário se fazer um aplicativo que, no caso deste trabalho, foi elaborado usando a IDE Android Studio no intuito de se gerar um arquivo de formato ".apk" para então instalar em um *smartphone*.

Para o desenvolvimento da aplicação foram necessários os seguintes requisitos:

- Android Studio instalado no computador;
- API Android de exemplo para publicação e inscrição no AWS IoT;
- AWS IoT configurada para tal conexão.

A aplicação foi desenvolvida com base na SDK da Amazon para programação Android usando os recursos de programação para comunicação com o AWS IoT via MQTT, recursos de autenticação e uso de políticas de interação com a plataforma em nuvem. O aplicativo contou com informações de usuário e segurança como parte de seu código fonte para devido acesso e autenticação no sistema. Com esses dados preenchidos corretamente no código e estando tudo dentro dos padrões de segurança estabelecidos o aplicativo poderá ter uma conexão bem-sucedida com a nuvem.

Os arquivos de exemplo usados como base para o desenvolvimento do aplicativo foram os disponibilizados pela Amazon em seu *github*, mais especificamente, no repositório do exemplo "*PubSub Android with WebSockets Sample*", ou em português, exemplo do AndroidPubSub com *Websockets*. Esse exemplo demonstra o uso da API do AWS IoT para publicar e assinar tópicos MQTT através de *Websocket*, que é uma tecnologia avançada que permite abrir uma conexão entre cliente e servidor. A autenticação da conexão é feita pelo Amazon Cognito que é um serviço de nuvem responsável pelo controle e regras de acesso nas plataformas da Amazon.

Através da configuração da plataforma em nuvem, que será abordada mais detalhadamente no item 3.4, para a elaboração desse aplicativo de exemplo bastou-se preencher alguns campos no decorrer do código fonte. Utilizou-se o programa Android Studio para abrir a pasta do código e visualizar os campos a serem alterados como é possível verificar na Figura 13.

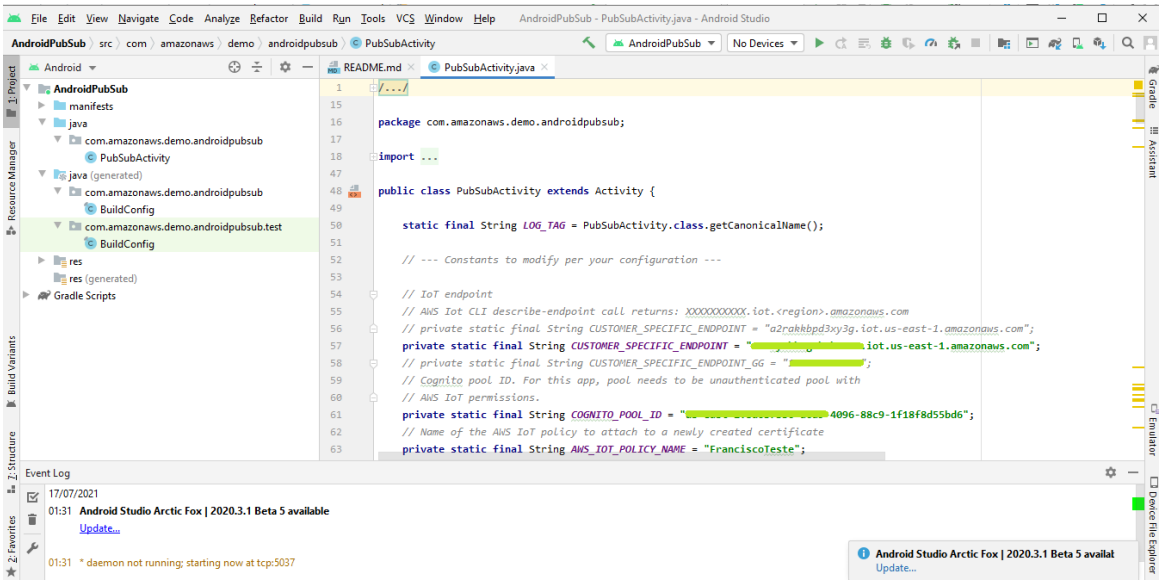

Figura 13- Interface de programação do Android Studio

Fonte: Autor

O funcionamento do aplicativo considerando o código de exemplo baixado é da seguinte maneira: quando o aplicativo é iniciado, ele procura por uma chave local com um certificado do AWS IoT. Se a chave não existir localmente, o aplicativo de exemplo obterá credenciais temporárias usando o Cognito e fará uma chamada para o AWS IoT Core para criar um certificado. Este certificado é então armazenado localmente e usado para conectar-se ao AWS IoT usando o MQTT. Nesse caso, o aplicativo também faz uma chamada para anexar uma AWS IoT *policy* a esse novo certificado. As *policies* são objetos no formato JSON que são criadas pelo usuário na plataforma com o intuito de adicionar ou retirar permissões para cada tipo de acesso. O fluxo neste exemplo pressupõe que já tenha sido criado essa *policy* e fornecido o nome por meio do campo “AWS_IOT_POLICY_NAME” no código.

Esse exemplo de aplicativo pressupõe que o usuário esteja se conectando a uma região com o AWS IoT e o Cognito (como us-east-1, ap-northeast-1 ou eu-west-1) e que o certificado tenha sido criado nesta região.

Assim, o campo “MY_REGION” no código é usado na criação do provedor de credenciais no Cognito, bem como no cliente AWS IoT.

O aplicativo foi elaborado seguindo os passos abaixo:

1. Importar o projeto de exemplo *AndroidPubSubWithWebSockets* para o programa Android Studio, clicando no menu superior esquerdo da janela e depois em "*Import Project*";
2. Importar as bibliotecas necessárias para o projeto automaticamente;
3. Configurar a plataforma em nuvem como explicado na seção seguinte;
4. Obter o endereço do *broker* MQTT na nuvem;
5. Obter as credencias para conexão;
6. Anexar uma política de interação com a nuvem;
7. Preencher os campos: "PoolId": "REPLACE_ME" e "Region": "REPLACE_ME" no arquivo "awsconfiguration.json" com os dados obtidos;
8. Preencher corretamente os campos: "CUSTOMER_SPECIFIC_ENDPOINT = "CHANGE_ME";
9. Fazer o build, instalar e usar o aplicativo;

Depois que a conexão for estabelecida, o aplicativo apresenta uma interface simples para publicar e assinar tópicos na AWS IoT como mostra a Figura 14.

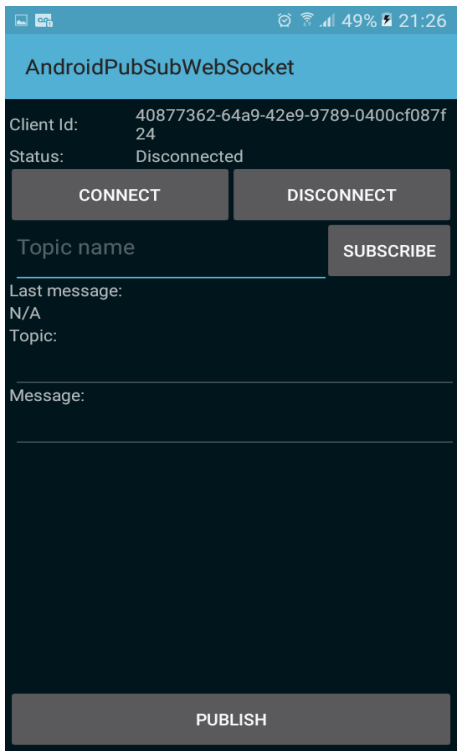

Figura 14- Interface padrão do aplicativo exemplo.

Fonte: Autor

Com os passos acimas executados, a plataforma da AWS configurada, os campos “PoolId”, “Region” e “CUSTOMER_SPECIFIC_ENDPOINT” devidamente preenchidos, pode-se alterar o código fonte em relação ao tratamento de mensagens que serão trocadas entre o aplicativo, a nuvem e o hardware. Com as mudanças no código fonte também foi possível alterar o design do aplicativo.

Para fazer um aplicativo de interface amigável foi utilizado um pouco de conhecimento de linguagem de programação Java Script e o padrão de mensagens no formato JSON. A Figura 15 mostra uma comparação entre o aplicativo modificado e o aplicativo de exemplo que teve seu código alterado para se alcançar o objetivo do trabalho. Na esquerda se tem o aplicativo de exemplo com código nativo disponibilizado pela AWS. Na direita é mostrada a tela da aplicação com algumas alterações no design e em algumas funcionalidades.

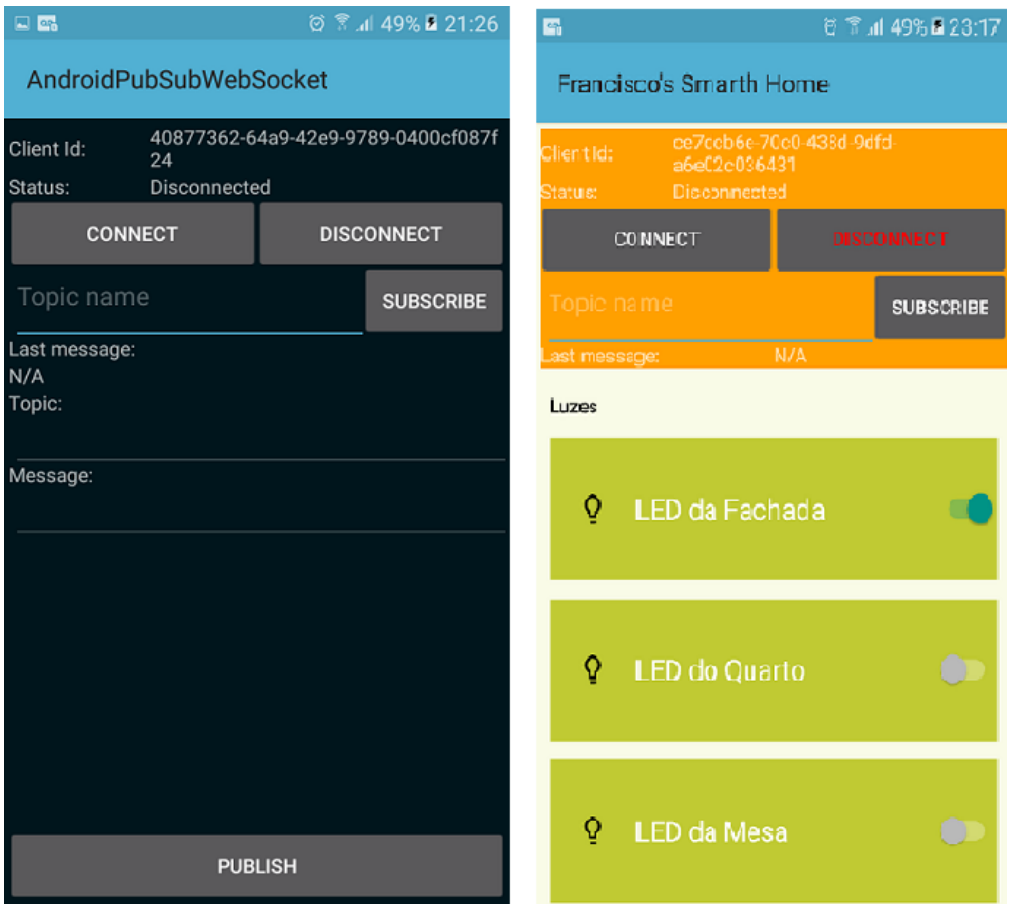

Figura 15 - Comparação de exemplo e a aplicação modificada

Fonte: Autor

Ainda na Figura 15, é importante notar no aplicativo modificado que na mesma tela pode-se visualizar os dados de autenticação na plataforma como *client id*, *status* de conexão e, os botões para comandos dos dispositivos como ligar e desligar. Nota-se também que as cores e informações de respostas foram incluídas e/ou alteradas na interface resultante.

Por questões de organização do trabalho, tratou-se até aqui apenas as principais partes do que foi desenvolvido para a elaboração do aplicativo.

3.3 ARQUITETURA PROPOSTA

Após a definição dos elementos de bordas, ou seja, o que será monitorado e por onde será o monitoramento, pode-se então visualizar o modelo de arquitetura proposta. A Figura 16 demonstra o diagrama de funcionamento e fluxograma de dados permitindo ao usuário o total controle dos dispositivos através do aplicativo.

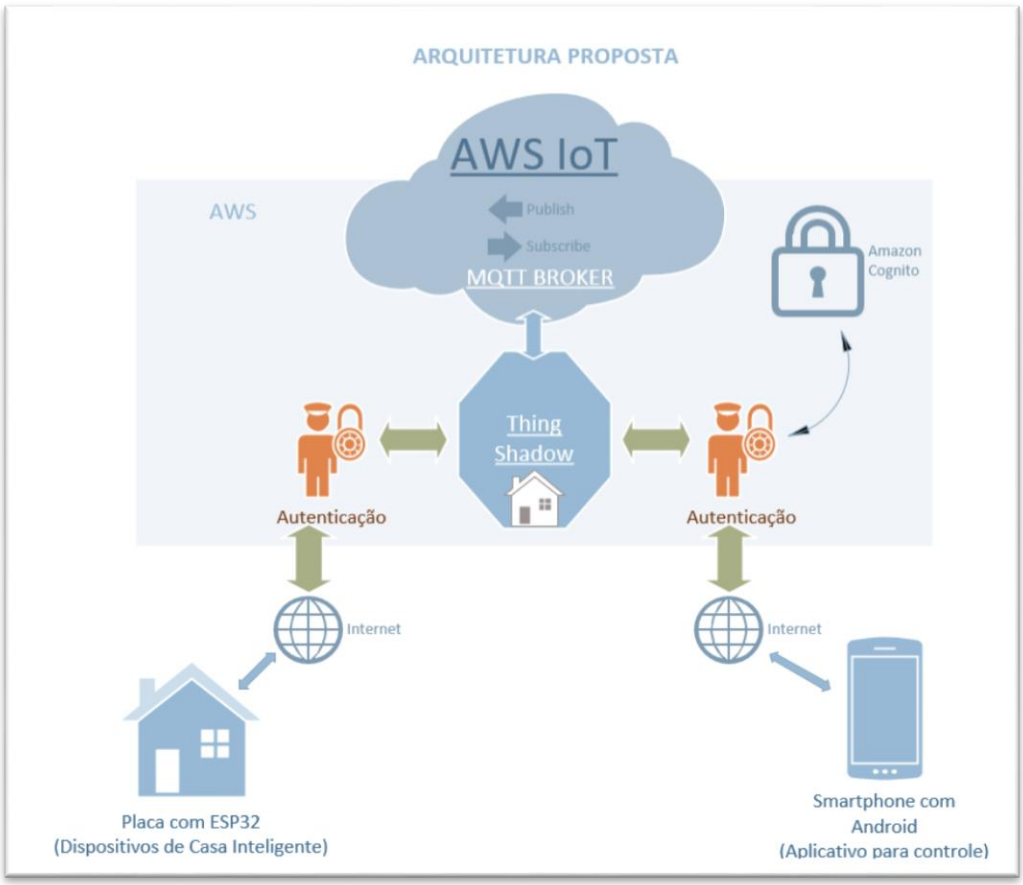

Figura 16- Arquitetura proposta

Fonte: Autor

Nota-se que para a comunicação com a nuvem da AWS é necessária autenticação dos dois lados, o que reforça a segurança do sistema. Além de se autenticar na plataforma, os clientes aplicativos e placa têm que ter sincronização em relação aos tópicos específicos para troca de mensagens. Objetiva-se usar os serviços AWS IoT e Cognito para troca de mensagens e autenticação respectivamente. O recurso sombra da coisa, do Inglês, *Thing Shadow* também é usado para sincronização na nuvem dos estados de funcionamento dos dispositivos.

3.4 CONFIGURAÇÕES DA PLATAFORMA EM NUVEM

Para o seguinte projeto, a solução da AWS IoT foi configurada para prover a conexão do hardware e do aplicativo para controle dos equipamentos. Por isso, a configuração é feita em duas partes, uma para configurar a conexão do aplicativo e outra para configurar a conexão do hardware.

Inicialmente é necessário ter uma conta de acesso na AWS ou criar uma para efetuar o *login* e começar a usar os recursos de nuvem. Para isso, basta acessar a página https://aws.amazon.com/ e clicar em fazer *login*, colocar usuário, senha e selecionar o serviço a ser configurado. Visando alcançar o objetivo do projeto, precisou-se configurar os serviços de autenticação do AWS Cognito e a solução AWS IoT de maneira explicada a seguir.

3.3.1 AWS Cognito - Configuração Para Conexão do Aplicativo

Para a implementação do aplicativo, é necessário obter os devidos campos de código diretamente da plataforma após executar os seguintes passos:

1. Baixar os arquivos de projeto do aplicativo pelo github;
2. Importar o projeto AndroidPubSub no Android Studio;
3. Importar as bibliotecas;
4. Acessar o Amazon Cognito, clicar em criar e depois em nomear um grupo de identidades;
5. Criar e selecionar a função “Cognito_IoTSampleUnauth_Role”;
6. Depois que mudar de tela, salvar o “ID do grupo de identidades”, que é um código tipo “us-east-1:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx”;

7. Anexar a política referente à função criada no passo 5 através do botão “Anexar Política”;

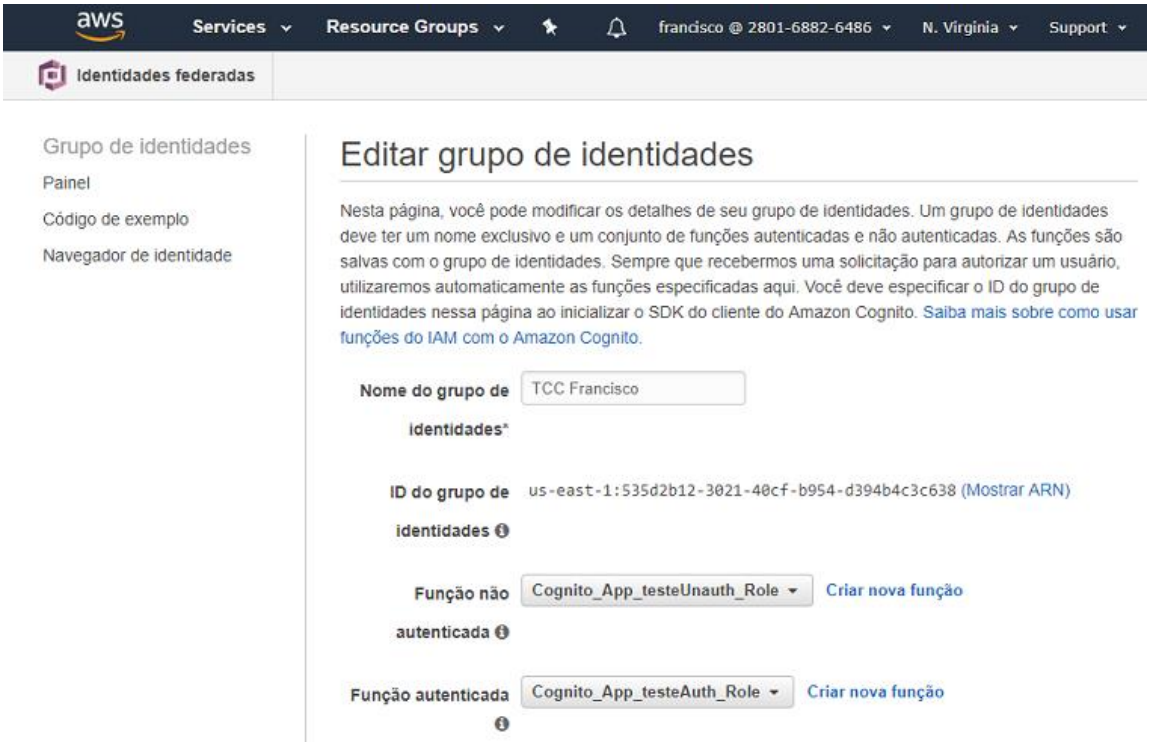

Figura 17 - Tela de configuração do AWS Cognito

Fonte: Autor

A Figura 17 mostra a tela de configuração do grupo de identidades, nota-se que é de simples configuração, visando a segurança e praticidade na adição de novos grupos de usuários.

Com os passos anteriores executados corretamente, uma janela com uma parte de código na linguagem Java Script vai aparecer. Deve-se copiar esse trecho e colar no código do Android Studio aberto anteriormente. Esse simples trecho de código chama a função de conexão ao Cognito, o que possibilita a autenticação e geração de um certificado que será armazenado na memória interna do smartphone.

```
// Inicializar o provedor de credenciais do Amazon Cognito
CognitoCachingCredentialsProvider credentialsProvider = new CognitoCachi
ngCredentialsProvider(
    getApplicationContext(),
```

```
"us-east-1:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx", // ID do grupo de iden
tidades
    Regions.US_EAST_1 // Região
);
```

Após seguido esses passos, a plataforma está pronta para autenticar aplicativos que tentarem conexão com o serviço da AWS IoT bastando terem as devidas credenciais em seu código fonte.

3.3.2 AWS IoT –Configuração para conexão com o hardware

Como anteriormente, para a devida configuração, basta ter feito *login* na plataforma, selecionar o serviço AWS IoT e depois a opção “coisas”, na tela resultante terá a opção de criar uma nova coisa. Avançando nos passos seguintes, foi preciso dar um nome para a coisa e gerar um certificado o qual é baixado e guardado para se usar embarcado no ESP32 e assim realizar a conexão e troca de mensagens com o AWS IoT.

A Figura 18 mostra a tela de listagem das coisas criadas na plataforma e no canto superior direito a opção de “criar itens”. É possível observar que todas têm um nome único, mas que podem fazer parte do mesmo tipo.

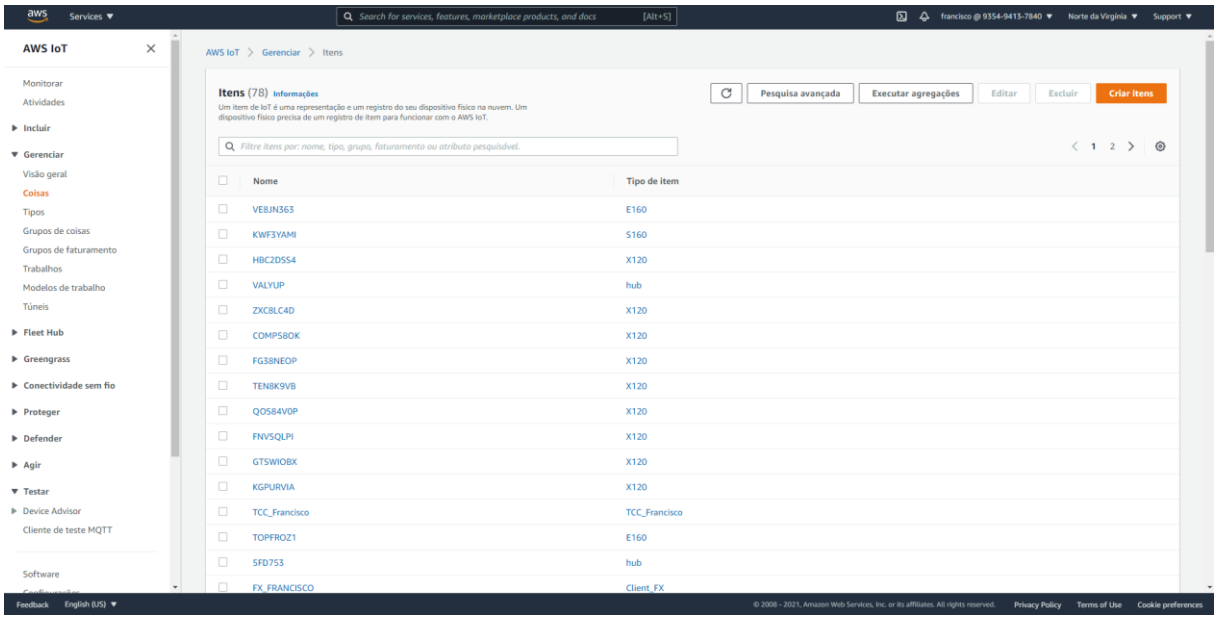

Figura 18- Tela de gerenciamento e listagens de coisas criadas na AWS IoT

Fonte: Autor

Como resultado se tem a coisa criada com o certificado ativado e pronta para ser usada na plataforma. A Figura 19 mostra a tela após a criação do “item” ou “coisa” com alguns parâmetros colocado na hora de criar.

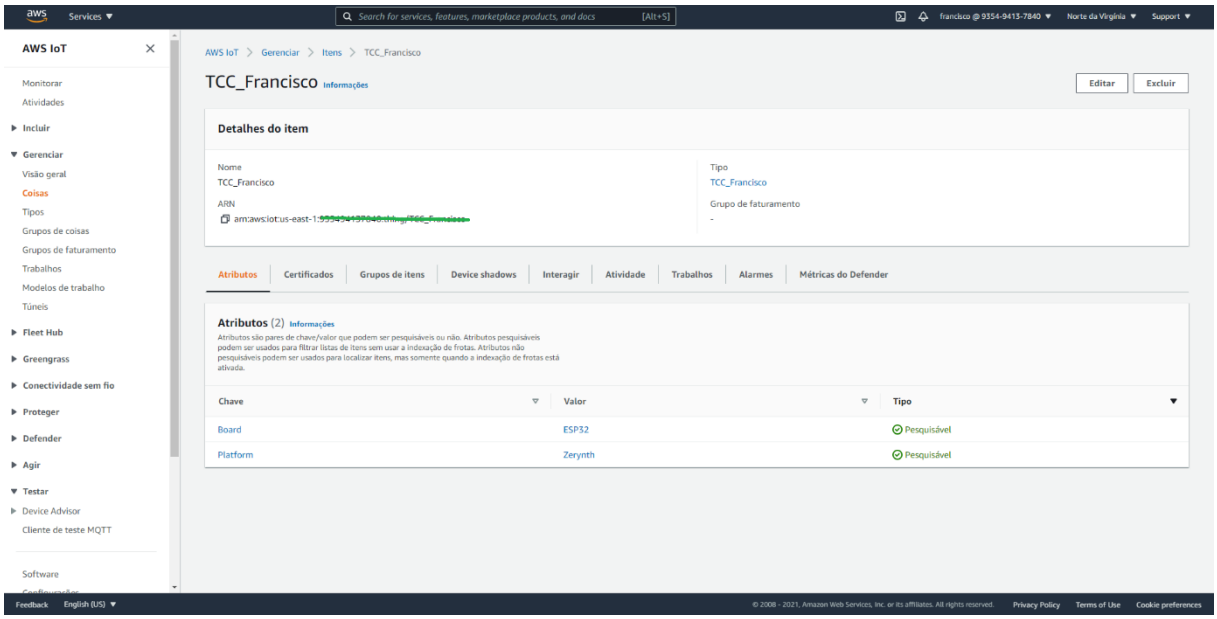

Figura 19- Criação de itens ou coisas na AWS IoT

Fonte: Autor

3.5 DESENVOLVIMENTO DE FIRMWARE

O firmware da placa ESP32 foi desenvolvido usando linguagem Python na plataforma do Zerynth Studio. Foi escolhido essa plataforma devido a sua facilidade de conexão com o WiFi e a comunicação com o *broker* em nuvem; a linguagem de programação Python foi escolhida por ser de alto nível para microcontroladores, o que facilita o entendimento e, por possuir uma forma simples de criar e executar *threads*. Destaca-se no *firmware* as seguintes etapas:

Autenticação na AWS IoT;

Comunicação por MQTT;

Tratamento de mensagens da sombra da coisa.

Cada uma dessas etapas é explicada mais detalhadamente a seguir.

3.5.3 Autenticação na AWS IoT

Através da *policy* criada para acesso neste trabalho, do certificado individual juntamente com a chave privada foi possível fazer a comunicação da placa ESP32 na AWS IoT. Essas informações foram incluídas na estrutura de código de exemplo disponível pelo Zerynth para conexão com a AWS IoT, o que facilita o desenvolvimento do *firmware*. De forma simples e com poucas linhas de código faz-se possível fazer a comunicação da placa ESP32 com a nuvem. Os dados incluídos no código do Zerynth são os seguintes:

```
{
  "cert_arn": "",
  "endpoint": "",
  "mqttid": "my_aws_thing",
  "policy_name": "my_aws_thing_policy",
"thingname": "my_aws_thing"
}
```

A política padrão usada para os acessos é mostrada a seguir, nela todos os acessos ao AWS IoT são permitidos. Contudo, os acessos aos outros serviços como banco de dados, por exemplo, não são permitidos.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
"Resource": "*"
}
  ],
  "Version": "2012-10-17"

}
```

Os certificados são incluídos dentro da pasta de projeto onde consta o arquivo main.py e são carregados automaticamente ao corpo do código para realizar a autenticação.

3.5.4 Comunicação por MQTT

Uma vez autenticado na AWS IoT a placa ESP32 já pode se comunicar com o *broker* MQTT. O *firmware* foi configurado para enviar informações relativas aos estados dos dispositivos através de mensagens usando o formato JSON. O *broker* que o *firmware* tem que se comunicar encontra-se no *endpoint* da AWS IoT que possui o seguinte formato:

```
mqtts://<endpoint>.iot.us-east-1.amazonaws.com:8883
```

Os tópicos para troca de mensagens serão da sombra da coisa, que têm formatos exclusivos e únicos para cada coisa cadastrada na AWS IoT. Esses tópicos são configurados automaticamente pelo Zerynth uma vez que o usuário ou desenvolvedor preenche corretamente os dados nos campos do código. As informações são enviadas para um tópico que tem o seguinte formato:

```
$aws/things/TCC_Francisco/shadow/update
```

As informações são enviadas a cada 10 segundos ou assim que o estado de algum dispositivo for alterado, desta forma, espera-se que os dados fiquem sempre atualizados na nuvem.

3.5.5 Tratamento de Mensagens da Sombra da Coisa

O serviço *device shadow* adiciona sombras a objetos de coisas do AWS IoT (AMAZON, 2021). Trata-se de um documento no formato JSON criado especificamente para cada coisa registrada. Nele é possível se ter os estados desejados para uma coisa e o estado reportado pela própria coisa. O uso desse recurso no trabalho foi com o intuito de se ter sempre o estado de cada dispositivo disponível na nuvem, independente do ESP32 estar conectado ou não, além de possibilitar todo o controle simplesmente alterando o *device shadow*.

No trabalho em questão, a AWS IoT garante a tentativa de envio do estado desejado até o recebimento pelo ESP32, nesse cenário, o firmware tem a função de receber a solicitação do novo estado desejado, modificar o seu estado de forma correspondente e publicar uma mensagem com o estado atualizado através de tópicos reservados. A sombra da coisa armazena e disponibiliza esses estados reportados e toda vez que o estado desejado

muda, o firmware recebe isso automaticamente e executa o fluxo de atualização explicado nesta seção.

A função do *firmware* responsável por receber a atualização do estado desejado do *device shadow* é a shadow_callback(requested), onde o parâmetro *requested* é no formato JSON do tipo:

```
{
  "desired":
  {
      "publish_period": 10000,
      "state_LED1": "on",
"state_Coller": "off"
  }
}
```

A declaração da função que faz o tratamento da mensagem a cima, por exemplo é do tipo:

```
def shadow_callback(requested):
    global publish_period
    global state_LED1
    global state_COOLER
    for k,v in requested.items():
        if k == "publish_period":
            print('requested publish period:', v)
            publish_period = v
            print('return  ', k, v)
            return {'publish_period': publish_period}

        if k == "state_LED1":
            state_LED1 = v
            if state_LED1 == "on":
                digitalWrite(LED1, HIGH)
                return{'state_LED1': state_LED1}
```

```
            elif state_LED1 == "off":
                digitalWrite(LED1, LOW)
                return{'state_LED1': state_LED1}

        if k == "state_Cooler":
            state_COOLER = v
            if state_COOLER == "on":
                digitalWrite(COOLER, HIGH)
                return{'state_Cooler': state_Cooler}
            elif state_COOLER == "off":
                digitalWrite(COOLER, LOW)
               return{'state_Coller': state_Cooler}
```

A mensagem em formato JSON que o ESP32 recebe contém todos os parâmetros que o aplicativo publica como estado desejado, após isso e as devidas alterações, os estados desejados, do inglês, *desired*, se igualam ao estado reportado, do inglês, *reported*. Com isso a sombra do dispositivo deve ser algo do tipo:

```
{
  "state": {
    "desired": {
      "publish_period": 10000,
      "state_LED1": "on",
      "state_Cooler": "off"
    },
    "reported": {
      "publish_period": 10000,
      "state_LED1": "on",
      "state_Cooler": "off"
}
  }
}
```

Capítulo 4

4 Resultados e Discussões

A fim de iniciar os trabalhos com o microcontrolador, testes iniciais foram feitos usando a placa ESP32 efetuando a conexão ao WiFi e se comunicando via MQTT com a nuvem. Para isso, a interface do console da AWS foi utilizada, onde pode-se criar os certificados e a *policy* com as regras de conexão. Os certificados, assim como a *policy*, são baixados do AWS IoT e são arquivos em texto para se colocar no código fonte do firmware do microcontrolador. Após o *firmware* de teste ser carregado no ESP32 juntamente com os certificados, o desenvolvimento foi feito com base em mensagens de testes diretamente no editor de estado do *device shadow* da AWS IoT visto na Figura 20, principalmente alterando-se o campo "*desired*" na sombra da coisa e observando o tratamento e resposta da placa ESP32. Ainda na Figura 20 é possível observar o resultado do documento de device shadow para o protótipo resultante.

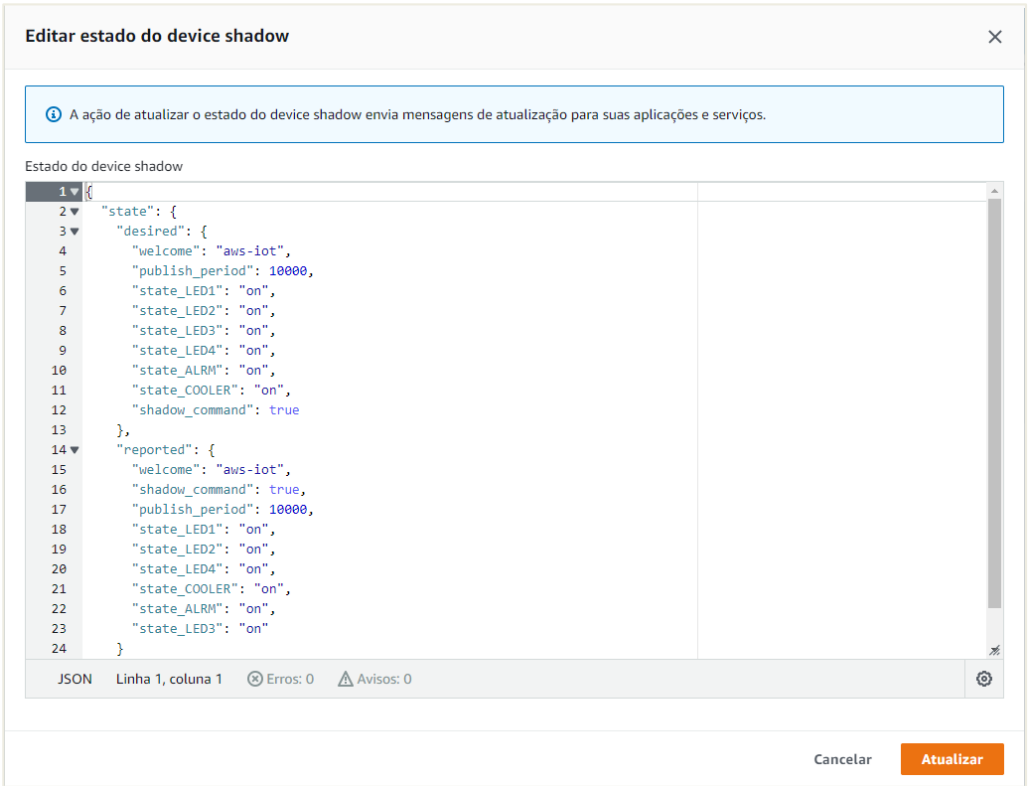

Figura 20- Editor do estado do device shadow.

Fonte: Autor

Por meio do editor foi possível realizar todas as mudanças no campo *desired* e pôde-se verificar que a placa ESP32 recebia a mensagem do campo alterado, tratava e logo reportava seu estado atual no campo *reported*. Quando havia falha ou atraso no comando, a sombra criava um objeto json chamado "*delta*" composto por todos os campos que diferenciava nos objetos *desired* e *reported*.

Como explicado anteriormente, o campo *desired* reflete a intensão de mudança de estado do dispositivo, sendo assim, o dispositivo retornaria a mudança realizada dentro do campo *reporte* caso o comando fosse executado com sucesso. Com base nisso, pôde-se observar o comportamento esperado do fluxo de comunicação entre o ESP32 e o *broker* na AWS IoT, realizados com sucesso. Além disso, considerando os dispositivos e as funções implementadas no firmware, a sombra resultante do trabalho é mostrada na Figura 21.

Estado do device shadow


```
{
  "state": {
    "desired": {
      "welcome": "aws-iot",
      "publish_period": 10000,
      "state_LED1": "off",
      "state_LED2": "off",
      "state_LED3": "off",
      "state_LED4": "off",
      "state_ALRM": "on",
      "state_COOLER": "off",
      "shadow_command": true
    },
    "reported": {
      "welcome": "aws-iot",
      "shadow_command": true,
      "publish_period": 10000,
      "state_LED1": "off",
      "state_LED2": "off",
      "state_LED4": "off",
      "state_COOLER": "off",
      "state_ALRM": "on",
      "state_LED3": "off"
    }
  }
}
```


Figura 21 - Sombra da coisa obtida.

Fonte: Autor

Posteriormente, para começar os testes de software foi carregado o código fonte do exemplo da SDK para Android disponibilizado pela Amazon para comunicação com a AWS IoT. O arquivo gerado no Android Studio de extensão ".apk" foi transferido via USB para o smartphone e instalado usando a interface de configuração, neste ponto é preciso ativar a permissão de instalação de aplicativos que não estão na loja de aplicativos oficial.

Antes de chegar na versão usada no trabalho, vários testes foram feitos com a aplicação instalada. Um dos mais importantes foi o teste de instabilidade de conexão, do qual se tratou de fazer várias tentativas de conexão com a internet tanto através de dados móveis como por Wi-Fi. Os resultados para estes testes foram satisfatórios, mostrando que a aplicação era estável em relação a esse requisito. Outro teste não menos importante foi o de cliques de comando praticamente simultâneos, do qual se tratava de testar se todos os cliques nos botões resultariam em um comando no servidor. Para esses testes o aplicativo se mostrou bem estável e o desenvolvimento pode seguir.

O aplicativo resultante é observado na Figura 22 onde pode-se observar os mesmos campos da tela do exemplo citado na Figura 15, como os botões "*connect*" e os campos "*Client Id*". Também se observa os campos da tela obtidos pela alteração do código fonte como os botões de "LED", "ALARME" e "Ar Condicionado".

Esse aplicativo em seu modo de exemplo foi de simples configuração e instalação, porém o entendimento do seu funcionamento foi difícil de encontrar nas documentações. Mas uma vez funcionando e a documentação entendida, foi simples alterar seu código fonte para elaboração da tela proposta. Para esse desenvolvimento, um conhecimento básico em programação em Java Script foi necessário, contudo obteve-se sucesso.

Figura 22 - Aplicativo Android Obtido

Fonte: Autor

Com o aplicativo com seus botões configurados, foi possível testá-lo fazendo uma requisição para alterar o campo “desired”, assim como foi feito manualmente enquanto se testava o *hardware*. Os botões virtuais do aplicativo têm o intuito de possibilitar ao usuário interagir com o sistema automatizado na maquete e ter como retorno os estados dos dispositivos controlados. Foi empregado para o funcionamento o fluxo de comunicação onde ao clicar em um botão o aplicativo envia uma mensagem MQTT para o servidor, esse servidor altera a *device shadow* na AWS e entrega a mensagem para o microcontrolador, o mesmo atua sobre os dispositivos e retorna o novo estado. Essa interação de hardware, software e servidor foi observado com sucesso. A Figura 23 demonstra como ficou o ícone do aplicativo na tela inicial do smartphone.

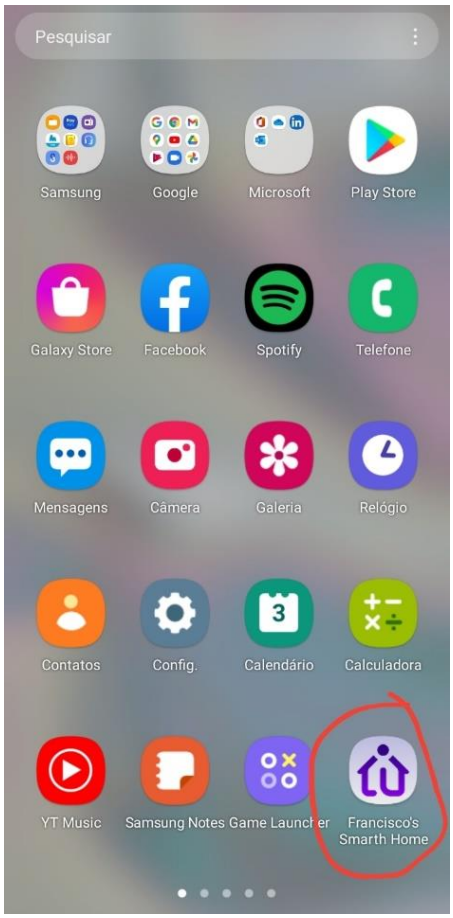

Figura 23- Ícone do aplicativo na tela do smartphone

Fonte: Autor

Pensando na experiência do usuário, durante o decorrer do trabalho houve tentativas de desenvolver uma resposta do botão de comando no aplicativo que refletisse o estado reportado no *device shadow*, principalmente para o estado do alarme, contudo não foi possível

realizar com os conhecimentos básicos em Java Script que possuía naquele momento. Como solução de contorno um *buzzer* foi incrementado na parte do *hardware* acrescentando uma resposta sonora ao estado real do alarme.

Foi possível observar que seguindo a metodologia de desenvolvimento escolhido é possível implementar facilmente outras funções como, por exemplo, controlar mais lâmpadas, controle de motor, etc. Isso se dá pelo fato de o código do aplicativo ser construído por meios de blocos de códigos, de mesmo modo o *firmware* do microcontrolador ser desenvolvido em *threads* e funções.

Se tratando do hardware, uma maquete completa foi escolhida para abrigar os dispositivos controlados e uma base de madeira foi acrescentada ao projeto para armazenar o microcontrolador e as conexões dos cabos elétricos, além de servir como suporte para a maquete. A Figura 24ilustra o resultado final da montagem da maquete com os dispositivos e o ESP32 devidamente conectado em *protoboard* dentro da base.

Figura 24- Montagem final da maquete

Fonte: Autor

Ainda se tratando de *hardware*, usou-se os dispositivos nas seguintes GPIO conforme tabela 02. Nota-se que apenas GPIO digitais foram usadas e que ainda assim, pôde-se obter o sucesso no funcionamento.

Tabela 02 – Relação de GPIO usadas e suas funções.

GPIO	TIPO	FUNÇÃO
5	Saída Digital	Ativação LED1 (Luz do térreo)
21	Saída Digital	Ativação LED2 (Luz da mesa)
22	Saída Digital	Ativação LED3 (Luz da fachada)
15	Saída Digital	Ativação LED4 (Luz do teto superior)
23	Saída Digital	Ativação Cooler (Ar Condicionado)
12	Saída Digital	Ativação LED do Alarme (LED do alarme)
16	Saída Digital	Ativação BUZZER (Sinal sonoro do alarme)
17	Entrada Digital	Leitura de estado do LED1
18	Entrada Digital	Leitura de estado do LED2
19	Entrada Digital	Leitura de estado do LED3
4	Entrada Digital	Leitura de estado do LED4
28	Entrada Digital	Leitura de estado do Cooler
14	Entrada Digital	Leitura de estado do Botão do alarme

O *firmware* do ESP32 foi desenvolvido usando a linguagem Python e o uso dos *threads* se mostrou bastante eficiente e prático. As principais funções, *threads* e interrupções usadas no código estão relacionadas na tabela 03 com suas respectivas funcionalidades.

Tabela 03 – Relação das funções, threads e interrupções declaradas no código do ESP32.

DECLARAÇÃO	TIPO	FUNÇÃO
SHADOW_CALLBACK	Interrupção	Tratar a mensagem recebida por MQTT
ON_TOUCH_BTN	Interrupção	Tratar a leitura do botão de alarme
ON_PIR_DETECTION	Interrupção	Tratar a leitura do estado do sensor PIR
BLINKLEDALRM	Thread	Piscar o LED de alarme e tocar o buzzer
PUBLISH_THING_STATES	Função	Publica o estado dos dispositivos no MQTT

Capítulo 5

5 Conclusão

O presente trabalho validou a possibilidade de desenvolvimento de um sistema de controle residencial por meio da internet utilizando três componentes básicos: microcontrolador, servidor em nuvem e aplicativo de *smartphone* android.

O projeto apresentou soluções em nuvem como o AWS IoT, que é uma plataforma onde pode se destacar sua estabilidade e segurança de conexão. Além de apresentar o desenvolvimento de firmware em uma plataforma completa como o Zerynth, onde se é possível programar em python para microcontroladores. O uso do Android Studio sem dúvidas se mostrou muito promissor, visto que a grande maioria dos smartphones usam o sistema operacional android.

Sobre os testes realizados, foi possível obter as informações em tempo real dos dispositivos na maquete como o estado dos LED, presença detectada pelo sensor, estado do alarme e do ar condicionado. Para obter-se tais informações, foi necessária uma troca segura de dados *online* e de uma conexão estável dos dispositivos o que se mostrou funcionar perfeitamente devido a todos os comandos serem executados pelo sistema.

Evidentemente o protótipo não está pronto para o mercado. É necessário considerar aprimoramentos na aplicação das variáveis de *hardware* e *software* para se ter um cenário mais realista. Contudo o desenvolvimento em uma maquete proporcionou validar diversas funcionalidades de uso no dia a dia e a interação entre o mundo real e o virtual. Pode-se considerar que a arquitetura e os dispositivos são adequados para um cenário de controle residencial. Os únicos custos que foram em *hardware* e plataforma em nuvem se mostrou irrisório quanto aos benefícios, podendo diminuir mais ainda ao ser produzido em larga escala, o que torna mais atrativo comercialmente.

Em geral o resultado final do sistema foi satisfatório. A arquitetura utilizada proporcionou uma troca de informações muito simples e estável por meio de uma conexão segura. A facilidade e simplicidade do aplicativo deixou o sistema muito intuitivo e fácil de ser operado pelo usuário final. Esses fatos demonstram que os benefícios da IoT tendem a melhorar a confiabilidade da troca de informações entre o ambiente virtual e o físico.

REFERÊNCIAS

AMAZON, AWS. **Device Shadows**. Disponível em: <https://docs.aws.amazon.com/pt_br/iot/latest/developerguide/iot-device-shadows.html> Acessado em 14 de fevereiro de 2020.

ANDROID. **Fundamentals Android**. 2018. Disponível em: <https://developer.android.com/guide/components/fundamentals_android>. Acessado em: 12 de julho de 2019.

BALDISSERA, Julia. **Integração da Computação na Nuvem e Internet das Coisas** / TCC do Curso de Bacharelado em Sistema da Informação. Universidade Federal de Santa Catarina. 2017

BARBOSA R. **Decreto que cria Plano Nacional de Internet das Coisas pode ser assinado ainda nesta semana**. Disponível em: <https://www.tudocelular.com/tech/noticias/n135672/decreto-iot-br-pode-ser-assinado-nesta-semana.html#comments>. Acessado em 28 de dezembro de 2018

BOSWARTHICK, David; ELLOUMI, Omar; HERSENT, Oliver. **M2M Communications: A System Approach**. 1 ed.[S.L.]: Wiley, 2012.

BOTTA, Alessio. Integration of cloud computing and internet of things: a Survey. **Future Generation Computer systems**. Napoli Federico, Italy. 2015

CHAVES, Sidney. **A questão dos riscos ambientais de computação em nuvem**. Dissertação (Mestrado) - Departamento de Administração, Faculdade de Economia, Administração e Contabilidade da Universidade de São Paulo, p. 7. 2011

ESPRESSIF. **ESP32 Datasheet** 2018b. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em 10 de julho de 2018.

FALL, Kevin R. TCP/IP illustrated.—2nd ed. / Kevin R. Fall, W. Richard Stevens. ddison-Wesley Professional. 1995

FREITAS, Thiago Augusto de. **Internet das Coisas: Uma Análise sobre o Impácto da Tecnologia nos Cuidados com Animais Domésticos**. Dissertação de Mestrado em Sistemas de Informação. Universidade FUMEC. 2016

HOSTING, Web. **The Future of Cloud Computing: 7 Predications**. 2016. Disponível em:<https://www.webhosting.uk.com/blog/the-future-of-cloud-computing-7-predictions>. Acesso em 03/10/2018.

JUNIOR, Olair Ricardo. **Sistema de Monitoramento Residencial Baseado em Internet das Coisas** / TCC do curso de Engenharia Elétrica. Universidade Estadual de Londrina. 2017

MANCINI, M. **Internet das Coisas: História, Conceitos, Aplicações e Desafios**. Artigo na Revista Design Management, 16-22. 2018

MAZZEI, D. et al. **A full stack for quick prototyping of iot solutions**. IEEE, 2016.

MORGAN, J. **A Simple Explanation Of 'The Internet of Things'.** 2014. Disponivel em: <https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#497a94d31d09>. Acesso em: 19/10/2018.

OLIVEIRA, Bruno Bastos de; PISSOLATO, Solange Teresinha Carvalho. **Direito e Tecnologia de Hiperconectividade: aspectos jurídicos da internet das coisas**. Artigo na revista Unicuritiba. 2-3. 2020

PITON, Otávio Henrique Gotardo. **Automação residencial utilizando a plataforma em nuvem IBM Bluemix** / TCC de Engenharia Elétrica. Escola de Engenharia de São Carlos da Universidade de São Paulo. 2017

RAMALHO, Neilson Carlos Leite. **Um estudo sobre a adoção da computação em nuvem no Brasil**. Dissertação (Mestrado) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidade da Universidade de São Paulo, 2012.

RAY, Partha Pratim. ***A survey of IoT cloud platforms***. Department of Computer Applications, Sikkim University, 6th Mile, PO Tadong, Gangtok, Sikkim 737102, India, 2017.

SANTOS, André H. O. **Arquitetura de Rede TCP/IP**. 2017. Disponível em: <https://www.uniaogeek.com.br/arquitetura-de-redes-tcpip/>. Acesso em 05/11/2018.

SODRE, Eduardo. **Desenvolvimento do Framework Java - Fácil** / TCC de graduação em engenharia da computação. Fundação Educacional do Município de Assis – FEMA – Assis, 2011.

SOUZA, Jefferson; NASCIMENTO, Luís; FILHO, Paulo. **Arduino and Python: Do it yourself**. Universidade Estadual do Piauí (UESPI), Parnaíba, 2011.

YUAN, Michael. ***Getting to know MQTT***. 2017. Disponível em: <https://www.ibm.com/developerworks/library/iot-mqtt-why-good-for-iot/index.html>. Acesso em: 03 de dezembro de 2018.

ZHOU, M. et al. ***Services in the cloud computing era: A survey***. Em: IEEE. *Universal Communication Symposium (IUCS), 2010 4th International* p. 40. 2010.

Ed Burnette, Hello, Android: Introducing Google's Mobile Development Platform. Pragmatic Bookshelf, 2008.