UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

THIAGO MARCHETTI MICHIURA
MARCOS AURELIO VIDA DOS SANTOS

ESTUDO E DESENVOLVIMENTO DE PROTÓTIPO PARA AUTOMAÇÃO RESIDENCIAL

CAMPO MOURÃO
2021

THIAGO MARCHETTI MICHIURA

MARCOS AURELIO VIDA DOS SANTOS

ESTUDO E DESENVOLVIMENTO DE PROTÓTIPO PARA AUTOMAÇÃO RESIDENCIAL

Study and development of prototytpe for home automation

Trabalho de conclusão de curso de graduação apresentada como requisito para obtenção do título de Bacharel em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador(a): Eduardo Giometti Bertogna.

CAMPO MOURÃO

2021

THIAGO MARCHETTI MICHIURA

MARCOS AURELIO VIDA DOS SANTOS

ESTUDO E DESENVOLVIMENTO DE PROTÓTIPO PARA AUTOMAÇÃO RESIDENCIAL

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 26 de novembro de 2021.

_____

Eduardo Giometti Bertogna
Doutor em Engenharia Elétrica
Universidade Estadual de Campinas

_____

Gilson Junior Schiavon
Doutor em Engenharia Química
Universidade Estadual de Maringá

_____

Roberto Ribeiro Neli
Doutor em Engenharia Elétrica
Universidade Estadual de Campinas

CAMPO MOURÃO

2021

AGRADECIMENTOS

Agradecemos primeiramente a Deus por ter nos mantido nesta trilha certa durante este projeto de pesquisa, com saúde e forças para chegar até o final. Somos gratos a nossas famílias por todo apoio que sempre nos deram durante todas nossas vidas. Agradecemos todos os colegas que neste caminho de aprendizado contribuíram de alguma forma. Deixamos um agradecimento especial ao nosso orientador pelo incentivo, dedicação do seu escasso tempo ao nosso projeto de pesquisa.

RESUMO

A automação residencial (Domótica) está cada vez mais presente na vida das pessoas e já é uma realidade em várias residências brasileiras com soluções diferenciadas voltadas as necessidades de cada um. Desde o controle de carga, sensores que enviam notificações à smartphones até dispositivos que podem ser programados por horário, por necessidade ou outras regras. Este trabalho apresenta uma proposta de um sistema microcontrolado para automatizar uma residência. Utilizando-se de componentes de baixo custo, procurou-se demonstrar uma aplicação geral sobre a automação: controlar cargas e receber informações de sensores. O protótipo é baseado em três componentes elementares, um smartphone que controla as cargas e recebe informações, uma central de controle, e *endpoints* (cargas ou sensores) que são responsáveis por enviar e receber informações. A comunicação deste protótipo se torna real por meio da rádio frequência, ou seja, a comunicação entre central e *endpoints*. Para tanto, foram necessários ensaios de funcionamento, construção, verificação (testes) e relação de custos. O intuito deste trabalho é fomentar e retomar a discussão de sistemas relacionados a automação residencial, tanto quanto fornecer conteúdo para que mais pessoas tenham acesso a esta tecnologia e se inspirem para criar suas próprias soluções. Com este trabalho foi possível agregar conhecimento no entorno da automação residencial, tanto quanto desenvolver o protótipo, realizar os testes necessários, analisar o custo-benefício e buscar novas aplicações específicas para o estudo da automação residencial.

Palavras-chave: automação; domótica; microcontroladores; controle.

ABSTRACT

Home automation (Domotics) is increasingly present in people's lives and is already a reality in several Brazilian homes with differentiated solutions aimed at the needs of each one. From load control, sensors that send notifications to smartphones, to devices that can be programmed by time. This work presents a proposal for a micro controlled system to automate a home. Using low-cost components, we tried to demonstrate a general application on automation: controlling loads and receiving information from sensors. The prototype is based on three elementary components, a smartphone that controls load and receives information, a control center, and endpoints (loads or sensors) that are responsible for sending and receiving information. The communication of this prototype becomes real through radio frequency, that is, the communication between central and endpoints. For this purpose, functional testing, construction, verification tests, cost list were necessary. The purpose of this work is to encourage and resume the discussion of systems related to home automation, as well as provide content so that more people have access to this technology and are inspired to create their own solutions. With this work, it was possible to add knowledge in the home automation environment, as well as to develop the prototype, carry out the necessary tests, analyze the cost-benefit and seek new specific applications for the study of home automation.

Keywords: automation; domotics; microcontrollers; control.

LISTA DE ILUSTRAÇÕES

Figura 1: Ilustração da divisão de dispositivos da Automação Residencial. 19
Figura 2: Curva de operação de um sensor. 23
Figura 3: Curva de operação de um sensor digital. 23
Figura 4: Integração dos componentes da automação residencial. 26
Figura 5: Ilustração de arquitetura do sistema de Automação Residencial. 27
Figura 6: Ilustração protocolo MQTT. 29
Figura 7: Ilustração de aplicação MQTT. 29
Figura 8: Ilustração de um REST API. 31
Figura 9: O microcontrolador ATmega328p. 32
Figura 10: Arduino Pro Mini. 33
Figura 11: Relé eletromecânico visão:(a) esquemática (b) representação física. 34
Figura 12: Transceptor nRF24L01. 35
Figura 13: Raspberry Pi 3 B+. 37
Figura 14: Conversor AC-DC HLK-PM01. 37
Figura 15: Modulo de detecção de gás MQ-2. 38
Figura 16: Sensor de movimento PIR. 39
Figura 17: Sensor de temperatura DHT-11. 40
Figura 18: Sensor de Corrente. 41
Figura 19: Processo de automação e controle. 43
Figura 20: Previa do sistema implementado. 44
Figura 21: Ilustração de ligação Rpi com nRF24L01. 47
Figura 22: Ilustração de ligação do núcleo dos dispositivos endpoints. 48
Figura 23: Primeira camada de software da central. 51
Figura 24: Segunda camada de software da central. 52
Figura 25: Gerenciador de ambientes na AWS Elastic Beanstalk. 54
Figura 26: Fluxograma de aplicação em nuvem. 54
Figura 27: Sistema proposto. 57
Figura 28: Processo de comunicação nuvem e smartphone. 58
Figura 29: Diagrama didático de protótipo. 59
Figura 30: Ilustração de aplicativo e de suas configurações. 61
Figura 31: Listas de sensores e atuadores. 62

Figura 32: Protótipo final. 63

LISTA DE QUADROS

Quadro 1: Pinos Arduino Pro Mini.....31
Quadro 2: Pinos nRF24L01.....34
Quadro 3: Características HLK.....36
Quadro 4: Características MQ-2.....37
Quadro 5: Características do módulo PIR.....38
Quadro 6: Características DHT11.....39
Quadro 7: Características ACS712.....40
Quadro 8: Protótipo atuador.....42
Quadro 9: Protótipo central.....43
Quadro 10: Protótipo sensores.....43
Quadro 11: Ligações da central.....45
Quadro 12: Ligações de núcleo de hardware.....46
Quadro 13: Divisão e função de códigos de programação.....54
Quadro 14: Bibliotecas utilizadas no projeto.....55
Quadro 15: Investimentos para construção do protótipo.....62
Quadro 16: Condições ideais de funcionamento para a central.....63

LISTA DE ABREVIATURAS E SIGLAS

EEPROM	*Electrically Erasable Programmable Read-Only Memory*
I2C	*Inter-Integrated Circuit*
IDE	*Integrated development environment*
ISM	*Industrial Scientific Medical*
HDMI	*High Definition Multimedia Interface*
MQTT	*Message Queue Telemetry Transport*
PLC	*Power line carrier*
PNP	*Plug and play*
PWM	*Pulse Width Modulation*
RF	Rádio Frequência
SPI	*Serial Peripheral Interface*
SRAM	*Static Random Access Memory*
USART	*Universal Synchronous/Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO 13
2 OBJETIVOS 15
2.1 Objetivo geral 15
2.2 Objetivos específicos 15
2.3 Justificativa 16
3 DESENVOLVIMENTO 18
3.1 Fundamentação teórica 18
3.2 Componentes 21
3.2.1 Controladores 22
3.2.2 Sensores 22
3.2.3 Atuadores 24
3.2.4 Interface com o usuário 24
3.2.5 Conexão 25
3.2.6 Arquitetura do sistema 26
3.2.7 Armazenamento e processamento em nuvem 27
3.2.8 Protocolo MQTT 28
3.2.9 Interface de usuário 30
3.2.10 Rest API 30
3.2.11 Microcontrolador 32
3.2.12 Relés 34
3.2.13 Módulo de comunicação sem fio 35
3.2.14 Plataforma de desenvolvimento Raspberry Pi 36
3.2.15 Fonte de Alimentação 37
3.2.16 Sensor de gás 38
3.2.17 Sensor de movimento 39

3.2.18 Sensor de temperatura e umidade 40
3.2.19 Sensor de Corrente 41
3.3 Metodologia 42
3.3.1 Planejamento 45
3.3.1.1 Central de dispositivos de automação 46
3.3.1.2 Núcleo de dispositivos *endpoints* 47
3.3.1.3 Dispositivo *endpoint* atuador 48
3.3.1.4 Dispositivo *endpoint* sensor 49
3.3.1.5 Dispositivo *endpoint* de aplicação customizada 50
3.3.1.6 Software da central de dispositivos 50
3.3.1.7 *Broker* MQTT 53
3.3.1.8 Microsserviço hospedado na nuvem AWS 53
3.3.1.9 Interface com o usuário 55
3.3.1.10 Firmware genérico 55
3.3.1.11 Variáveis de Ambiente: arquivo dotenv 56
3.3.2 Descrição do protótipo 56
3.3.3 Descrição do processo 58
3.3.4 Resumo do sistema completo 59
3.3.5 Resultados e discussões 60
3.3.5.1 Teste de rede 63
3.3.5.2 Teste de conectividade do sistema 63
3.3.5.3 Teste de resposta do sistema 64
3.3.5.4 Levantamento de recurso financeiro 64
3.3.5.5 Considerações de funcionamento 65
4 CONCLUSÃO 67
REFERÊNCIAS 70

APÊNDICE A - Código de programação Endpoints 75

APÊNDICE B - Código de programação Camada 1 - Central.....83

APÊNDICE C - Código de programação Camada 2 - Central.....96

APÊNDICE D - Código de programação do Microsserviço.....101

1 INTRODUÇÃO

A automação tem demonstrado parte atuante em diversos segmentos de mercado, além de suas aplicações serem realizadas nas mais diversas formas, visando auxiliar e facilitar os processos do cotidiano da sociedade. Um importante segmento da automação que vem crescendo continuamente é a automação residencial, que é discutida neste trabalho.

O mercado da automação residencial, no Brasil, vem se transformando há décadas, passando de um item de luxo para realidade em muitas residências e condomínios, principalmente com o desenvolvimento de estudos e a facilidade de acessos a mecanismos de automação. Buscou-se desenvolver pesquisas e protótipo para demonstração das tecnologias no envoltório da automação residencial e demonstrar de forma simples e coesa que utilizando das mesmas é possível alcançar inúmeras aplicações.

A domótica, ou automação residencial, é todo e qualquer tipo de estudo de aplicações tecnológicas de cunho do lar, ou seja, que afetam todas as pessoas, desde a automatização de processos simples como ligação ou desligamento de cargas, até mensurar grandezas físicas e aplicações de rádio frequência para controle.

No século XXI, têm-se diversos exemplos de dispositivos capazes de realizar conexão, remetendo ao novo conceito de estudo chamado IoT (*Internet of Things* – Internet das Coisas), no qual dispositivos atuadores (relés, *drivers* de led, etc.) são conectados à internet para que possam ser controlados por interfaces de usuário (geralmente APIs desenvolvidas em nuvem ou em forma de aplicativos), possibilitando assim que os usuários finais ou clientes sejam providos de informações, controle e conforto no gerenciamento de sua residência ou indústria. Estes dispositivos, geralmente, utilizam uma frequência de 2,4 GHz ou rádio frequência, e são configurados de forma a utilizar a internet do local para se conectar, dessa forma conseguem enviar e receber informações para bancos de dados em nuvens que se conectam diretamente à aplicativos de *smartphone*.

A automação residencial representa o emprego de tecnologias ao ambiente doméstico (incluindo residências, condomínios, hotéis), com o objetivo de propiciar conforto, praticidade, produtividade, economia, eficiência e rentabilidade, com valorização da imagem do empreendimento e de seus usuários.

Nos dias atuais, as preocupações no desenvolvimento deste ramo concentram-se em torno da redução de custos dos equipamentos e de sua integração, visando ao compartilhamento de recursos. Assim, grande parte das instalações da unidade domiciliar poderá ser controlada por controle remoto e até mesmo via Internet, o que tornou muito mais fácil o acesso a novos serviços de comunicação, como alertas, voz sobre IP (Internet Protocol), intercomunicação, canais abertos, troca de mensagens entre moradores em toda parte da casa, entre outros recursos.

Algumas das partes essenciais deste trabalho incluem dispositivos atuadores, sensores, central de dispositivos e interface, ou seja, um usuário controlando e visualizando informações de uma central que por sua vez, se comunica com periféricos *endpoints* para, desta forma, oferecer conforto e praticidade no projeto proposto.

De acordo com THOMAZINI e ALBUQUERQUE (2010) os atuadores são dispositivos que alteram seu estado através de comandos elétricos recebidos. Assim é possível fazer a alteração em uma variável do ambiente. Podem ser conectados diretamente ao controlador ou possuir uma pequena interface que possibilita que outros dispositivos possam atuar no mesmo. Existem diversos tipos de atuadores como Válvulas, Relés, Cilindros, Motores, etc. Já os sensores, reagem eletricamente a alguma alteração de energia do ambiente que pode ser luminosa, térmica, cinética, entre outras. Sendo possível assim medir grandezas como temperatura, umidade, corrente elétrica, posição, velocidade e aceleração.

Desta forma, neste trabalho, procura-se elaborar uma arquitetura para implementação dessa tecnologia, utilizando um sistema de central de dispositivos (Raspberry Pi) e dispositivos *endpoint* de aplicações variadas. Buscou-se desenvolver o protótipo da forma mais didática possível para explicação de todos os componentes inerentes a sua montagem. O formato do protótipo se dá em forma de uma central de dispositivos e três dispositivos *endpoint*, sendo um deles um módulo atuador (ligando e desligando cargas), um deles um módulo sensor (recebendo informações para tomada de decisão) e um terceiro realizando as duas funções. O protótipo é feito dessa forma para que seja demonstrada a sua gama de utilidades, além de torná-lo flexível para todos os tipos de aplicação.

2 OBJETIVOS

2.1 Objetivo geral

O objetivo geral deste trabalho é estudar e desenvolver um dispositivo central responsável por automatizar processos em uma residência, controlando periféricos sensores e atuadores por meio de uma tecnologia de rede sem fio. As formas de controle serão ligar, desligar cargas, assim como monitoramento delas.

2.2 Objetivos específicos

- Estudar e destacar os conhecimentos entorno da tecnologia aplicada à automação residencial;
- Desenvolver um sistema de automação de baixo custo, que possa ser a base de desenvolvimento para aplicações de automação residencial;
- Desenvolver os periféricos atuadores capazes de controlar o estado de cargas conectadas ao sistema, além de serem capazes de verificar o estado instantâneo de cada atuador;
- Desenvolver a central, com a função *PnP*, capaz de reconhecer um novo periférico, automaticamente, associando a este um endereço para integrá-lo ao sistema;
- Elaborar a interface dispositivo-usuário, de forma que esta seja de fácil configuração e uso;
- Desenvolver um sistema, de acesso remoto ao dispositivo central, disponibilizando o acesso ao mesmo por meio de um *smartphone* com conexão à Internet;
- Desenvolver os softwares para que outras pessoas possam implementá-los em seus próprios projetos de automação de forma fácil e simplificada;
- Executar testes e verificar funcionamento e alcance dos dispositivos em relação a central.

2.3 Justificativa

Possuindo um sistema de automação residencial, é possível tornar o cotidiano mais confortável e eficiente. Ter o controle de acionamento de cargas em um único sistema muda totalmente a relação entre ser humano e dispositivo, assim, com o toque em uma tela. O desenvolvimento do dispositivo em questão facilitará o controle e o monitoramento de cargas e sensores interligadas em uma residência. De forma microcontrolada utilizando uma plataforma de desenvolvimento Raspberry Pi é possível controlar cargas e monitorar cargas de forma remota para ativação ou desligamento de dispositivos residenciais como lâmpadas, portões de garagens e afins.

Os benefícios da utilização do protótipo proposto são:

- Facilidade do controle de cargas: com a solução proposta o usuário tem uma maior facilidade de acesso remoto às cargas interligadas de sua residência. É possível, por exemplo, ligar de forma remota algum dispositivo eletrônico mesmo antes de adentrar sua residência, por meio do *smartphone*;
- Economia de energia: verificando o estado de suas cargas, o usuário poderá atuar na economia de energia de sua residência, proporcionando então maior eficiência energética com o menor custo;
- Segurança: é possível implementar sensores de segurança, alarmes sonoros e avisos de telefone, para segurança do usuário do sistema, trazendo tranquilidade e informação em tempo real sobre sua residência;
- Conforto: o conforto fornecido pela automação residencial é grande, dentre estes é possível citar o controle de todos os dispositivos e informações em tempo real, sendo possível programar e planejar o funcionamento de suas cargas assim como consultá-las para um eventual problema ocorrido na residência;
- Praticidade: a praticidade proporcionada pela aplicação e pela interface "*user friendly"* permite que com um simples toque ou com botões interativos utilizados pelo aplicativo seja possível ativar, desativar, consultar e verificar estado de suas cargas.

O aparelho de automação residencial será instalado em qualquer tipo de estabelecimento, fornecendo ao usuário controle e monitoramento sobre o dispositivo de seu interesse.

A escolha de um circuito microcontrolado, fisicamente ligado às cargas de uma residência por meio de relés, permitirá que o equipamento permaneça operacional

caso seja necessária a manutenção das interligações de cargas e, também, que se utilize o mesmo equipamento em diferentes cargas.

A proposta de projetar um sistema de automação para que possa ser replicado e implementado por outras pessoas, sem a necessidade de um conhecimento tão aprofundado no assunto, juntamente com a proposta de projetar um sistema de baixo custo, irá atrair novas pessoas para o universo da automação residencial, incentivando essas a desenvolver e aprimorar as tecnologias voltadas a essa área.

No que tange à ascendência da tecnologia na atualidade, procura-se demonstrar que a automação residencial é um estudo que abrange diversas áreas do conhecimento e que ainda existem diversos tópicos não implementados referentes à esta, como por exemplo aplicações específicas para controle de temperatura, controle de iluminação, controle de alarme, entre outros.

3 DESENVOLVIMENTO

3.1 Fundamentação teórica

A história da automação é antiga, podendo ter como ponto de partida o desenvolvimento dos primeiros dispositivos que automatizavam processos simples do cotidiano do ser humano. Existem inúmeros exemplos atrelados a esse estudo, de forma que se faz necessária a explicação do conceito da automação para o entendimento do trabalho proposto.

Segundo Teza (2002), a automação surgiu ainda nos primórdios da humanidade, sem um marco preciso em relação à data. Nesse caso, é considerado automação qualquer processo de auxilie o ser humano nas suas tarefas cotidianas, sendo elas comerciais, industriais, domésticas ou agrícolas. Um exemplo simples a ser citado é o uso da roda d'agua na automatização de processos de moagem, serrarias, ferrarias e trituração de grãos em geral.

Especificamente para a automação residencial,

> As primeiras incursões nestas tecnologias datam do final da década de 1970, quando surgiram nos Estados Unidos os primeiros módulos "inteligentes", cujos comandos eram enviados pela própria rede elétrica da residência, no conceito de PLC (*Power Line Carrier*). Tratava-se de soluções simples, praticamente não integradas e que resolviam situações pontuais, como ligar remotamente algum equipamento ou luzes. (MURATORI e BÓ, 2011, p. 71).

Ainda nessa década, o advento dos computadores pessoais e da internet, bom como a explosão da telefonia móvel passaram a ser mais acessíveis para consumidores, de forma que a aceitação das tecnologias residenciais passou a ser maior. Nesse viés, Teza (2002) reforça que a revolução que fora alavancada no século XVIII proporcionou mais do que a automação do mundo, que surgiu a partir da mecanização de processos e os quais são até hoje utilizados. A automatização, portanto, é um processo que se utiliza de dispositivos automáticos, eletrônicos e inteligentes para executar as tarefas de processos manuais.

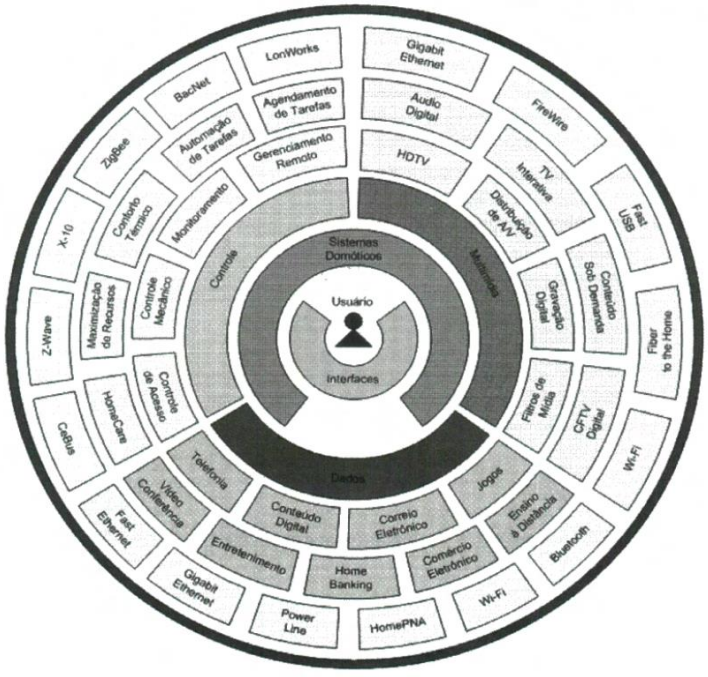
A definição da automação residencial pode ser também conhecida como Domótica, Residência Inteligente ou Casa do futuro a Automação Residencial trata da

integração de serviços e tecnologias que têm por finalidade tornar uma residência automatizada e obter aumento de economia, segurança, conforto e praticidade.

> A automação residencial pode ser definida como um conjunto de tecnologia que ajudam na gestão e execução de tarefas domésticas cotidianas. A sua utilização tem por objetivo proporcionar um maior nível de conforto, comodidade e segurança além de um menor e mais racional consumo de energia (BOLZANI, 2004 *apud* ACCARDI e DODONOVI, 2012, p. 156, 157).

Os dispositivos da automação residencial podem ser divididos de forma que podemos classificá-los de acordo com cada aplicação, a Figura 1 a seguir demonstra alguns exemplos de aplicações no mundo da automação residencial.

Figura 1: Ilustração da divisão de dispositivos da Automação Residencial.



Fonte: Adaptado de BOLZANI (2007)

De acordo com Stevan e Silva (2015), os diversos processos que têm necessidade de automação, quer no meio fabril, no meio comercial ou no meio residencial é incontestável pois diversas tarefas manuais e simples podem ser automatizadas. No cotidiano, é possível observar sistemas automatizados garantindo uma padronização, organização e melhoria no desempenho de suas funções. Como no exemplo citado por Camargo (2014), espera-se que, ao programar um micro-ondas

para funcionar por dez minutos, após esse período ele desligue, cumprindo assim o processo por si próprio, ao qual denomina-se automático.

Portanto é possível automatizar um dispositivo, equipamento, uma máquina ou um processo, definindo os parâmetros e variáveis para o controle automático, de forma a parametrizar todas as tomadas de decisões. Ao aplicar isso a um meio específico, pode-se agrupar várias automações focadas em um único seguimento, de forma que a automatização se dê em todo o ambiente. A “Domótica é o termo muitas vezes utilizado para identificar a automação residencial (*home automation*), que deriva do neologismo francês ‘*domotiq*’, que significa literalmente ‘casa automática’.” (DEMÉTRIO et al, 2016, p. 1).

Prudente (2017) afirma que a melhoria no conforto, na segurança nas unidades habitacionais e condomínios é um dos objetivos da domótica. Ele também exemplifica a ligação, o desligamento e a regulação da luminosidade de lâmpadas, instalação de aquecimento ou condicionamento de ar, comandos de controle de veneziana, dentre outras. Que juntos, podem influenciar diretamente no conforto dos transeuntes e residentes desta unidade habitacional, que pode ser um condomínio, um prédio ou uma residência.

Como Muratori e Bó (2011) explicam, o mercado da automação residencial teve como foco o desenvolvimento de projetos de novas residências, o que tem limitado o crescimento e a abrangência da outra parte do mercado (as casas já estruturadas). Esses mesmos indicadores estão variando nos últimos anos, com o surgimento de soluções mais simples e práticas, tecnologias sem fio e confiáveis. Esse tipo de solução é descrito por profissionais da área como solução *Plug and Play (Pnp)*, que possibilita a instalação em residências já consolidadas e permite a interação com o usuário comum (não especializado).

Ramos e Santos (2015) apontam que a ideia de uma casa moderna e automatizada foi durante muito tempo associada a algo caro, complicado, pouco acessível e de difícil implantação. Todos estes motivos criam uma barreira entre público da classe média e a aquisição de tecnologia para automatizar o ambiente ao seu redor.

Com o mercado economicamente ativo, a automação residencial é recente, traduzindo-se em um processo ainda em emergência. No entanto, nas economias mais desenvolvidas o cenário para as chamadas “casas inteligentes” tem evoluído de maneira muito positiva nos últimos anos. A tecnologia em ascendência tem um forte

papel nesse apoio à tecnologia de automação residencial, dispositivos como *smartphones, smartwatches,* entre outros têm ajudado a difundir os benefícios e a praticidade relacionada a tecnologia.

Percebe-se que a complexidade de comandos e a interface de usuário da máquina tem diminuído com o tempo e que a sociedade vem-se adaptando de forma positiva em relação a isso. A barreira entre usuários que não conseguem utilizar a tecnologia a seu serviço já é muito menor que há algumas décadas. Os dados sobre a situação do mercado da tecnologia de automação residencial são bem satisfatórios considerando a atualidade e o momento histórico que vivemos.

Segundo uma pesquisa da AURESIDE (2014),

> O mercado global de automação residencial teve um valor estimado de R$ 5,77 bilhões em 2014, R$ 4,41 bilhões em 2013 e deve chegar a $ 12,81 bilhões em 2020. O mercado está projetado para crescer a uma taxa composta de crescimento anual de 11,36% entre 2014 e 2020. Relatórios sobre eventos indicam que a tendência de alta de longo prazo deve continuar forte, assim como os observadores do mercado mantém esta certeza de crescimento da indústria.

Um estudo realizado pela Motorola Mobility, no Brasil, em 2012, apontou que 78% dos entrevistados têm interesse em automação residencial, mesmo que 37% desses ainda precisem entender mais sobre os benefícios dessa tecnologia. (TELETIME, 2012). Nesse mesmo estudo, a Motorola Mobility revelou que a média mundial de interesse em automação residencial é de 60%, de forma que a média brasileira se sobrepõe a ela. Ainda na questão de interesse nesse assunto, a AURESIDE (2013), em seu relatório especial sobre automação residencial informa que, a partir de pesquisas realizadas nos Estados Unidos da América, sistemas de automação residencial que contenham apelo pela sustentabilidade, economia de energia e preservação de recursos naturais estão sendo cada vez mais requisitados.

3.2 Componentes

Nesta seção do trabalho, foram dispostas as tecnologias envolvidas no desenvolvimento deste trabalho e todos os referenciais teóricos e técnicos relacionados a este. Algumas das tecnologias envolvidas são: componentes eletrônicos, microcontroladores, sensores, redes, armazenamento em núvem, protocolo MQTT e desenvolvimento de aplicativos (rest API).

Por trás do conceito e das definições da automação residencial, existem diversos elementos envolvidos na construção de um sistema inteligente de gestão residencial. De simples sensores, até complexas centrais de automação, que oferecem uma experiência ideal para as necessidades, desejos e condições de cada usuário. No presente tópico são abordados todos os elementos que podem compor um sistema de automação residencial.

3.2.1 Controladores

O controlador é uma parte essencial no projeto descrito, pois é ele que toma decisões: liga, desliga, programa. Enquanto os sensores fornecem informações sobre o ambiente, os controladores são capazes de controlar as grandezas físicas do local em que são aplicados.

Segundo Wendling (2010), os controladores são os componentes que controlam os dispositivos automatizados (sensores e atuadores). Com isso, monitora-se informações de sensores podendo enviar comandos para um atuador, que pode ativar ou desativar uma determinada carga e que irá influir diretamente no ambiente, proporcionando maior conforto e segurança. De modo geral, os controladores podem ter interfaces independentes, na forma de controle remoto ou serem sofisticadas centrais de automação.

3.2.2 Sensores

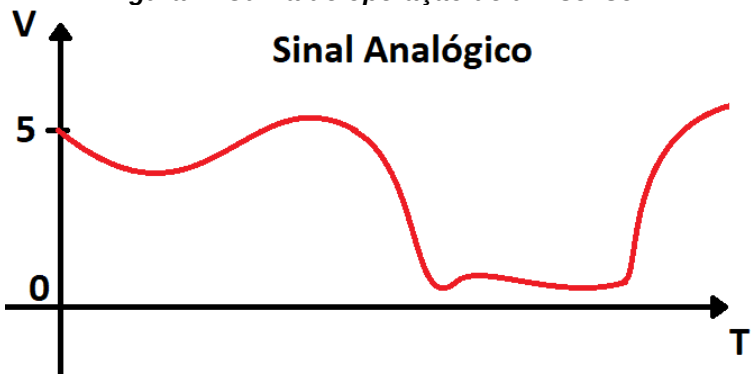
Os sensores são componentes, dispositivos ou circuitos elétricos capazes de mensurar grandezas físicas de um ambiente. São, geralmente, atrelados a uma ou mais grandezas. Esses são muito úteis nas aplicações residenciais, pois podem ser aplicados de forma a fornecer maior precisão na medição, informações sobre o sistema, entre outras variáveis essenciais para termos segurança, controle e conforto.

Sensores, portanto, são os dispositivos designados a ter sensibilidade elétrica à alguma forma de energia proveniente do ambiente, seja luminosa, térmica ou cinética e muitas outras formas de energia relacionando informações sobre grandezas físicas que precisem ser medidas, como temperatura, posição, corrente, entre outras (WENDLING, 2010).

Esses são subdivididos em duas grandes vertentes de estudo e de desenvolvimento, são elas: os sensores analógicos e sensores digitais. Os sensores

analógicos podem assumir qualquer valor no seu sinal de saída ao longo do tempo, desde que esse esteja dentro de sua faixa de operação. Essas variáveis são medidas por elementos sensíveis acoplados a circuitos eletrônicos não digitais. A Figura 2 identifica uma curva de operação de um sensor genérico.

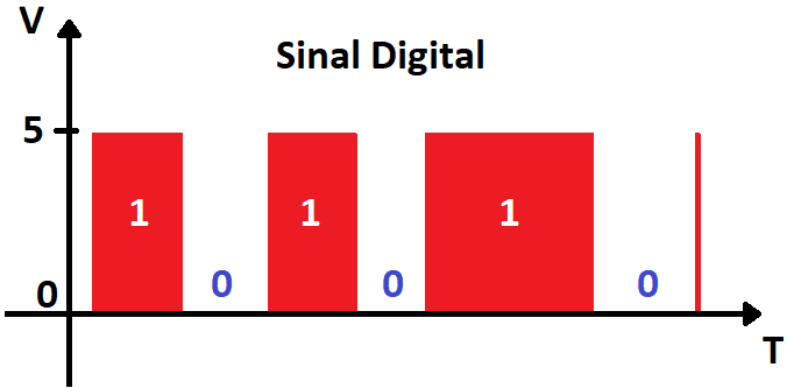
Figura 2: Curva de operação de um sensor.



Fonte: Adaptado de Embarcados (2021)

Já o sensor digital pode assumir apenas dois valores em seu sinal de saída ao longo do tempo, os quais podem ser interpretadas apenas como zero ou um, utilizando o sistema binário. Conforme Wendling (2010) aponta, eles podem ser utilizados na detecção de passagem de objetos, *encoders* na determinação de distância ou velocidade. Além disso, não existem grandezas físicas que assumam os valores disponíveis pelo sensor digital, de forma que existe a necessidade de um sistema de controle após sua conversão por um circuito eletrônico. A sua curva de operação está descrita na Figura 3.

Figura 3: Curva de operação de um sensor digital.



Fonte: Adaptado de Embarcados (2020)

3.2.3 Atuadores

Atuadores são elementos que produzem movimentos, atendendo a comandos que podem ser manuais ou automáticos. São usados em automação para entregar à planta a excitação necessária para seu funcionamento, na forma de energia adequada. Se o funcionamento da planta estiver baseado em algum movimento de uma de suas partes, serão necessários atuadores para fornecer energia mecânica para o movimento. Por exemplo, se a planta for um sistema térmico, será necessário um atuador que forneça energia térmica para atingir uma temperatura desejada.

Os atuadores são dispositivos que produz movimento. Interagem diretamente com grandezas físicas do ambiente, de forma a automatizar tomadas de decisão em relação às variáveis do ambiente em questão. Um exemplo simples seria composto por três elementos, um sensor de temperatura para análise de grandeza física, uma central de controle para tomada de decisão e um atuador qualquer, como uma resistência para aquecer o ambiente.

3.2.4 Interface com o usuário

Segundo Casadomo (2010), as interfaces com o usuário são mecanismos (*browsers*, *smartphones*, painéis, controles remotos, interruptores e etc.) que facilitam, para o usuário, a visualização das informações e interação com um sistema de automação, de forma simples e eficiente.

Nas palavras de Accardi e Dodonov “As interfaces (interruptores, celular etc.) se conectam diretamente aos controladores de forma a permitir que o usuário visualize as informações e interaja com o sistema de automação” (2012, p. 157). Elas, portanto, são partes importantíssimas em uma solução de automação residencial, já que a experiência do usuário é determinada a partir da experiência que este terá com seus componentes, sejam eles atuadores, sensores ou controladores.

3.2.5 Conexão

A interconexão é um elemento fundamental na automação residencial, já que faz a comunicação entre os dispositivos do sistema. Os protocolos de comunicação e rede são importantes para estabelecer uma ligação entre o *hardware* e o *software*, sendo possível a manipulação do sistema programado para utilização do usuário.

Existem dois tipos básicos de implementação de sistemas de automação, cada um com suas características, vantagens e desvantagens. As melhores aplicações e projetos sempre fazem análise da aplicação em questão. Os tipos são: Automação com fios e automação sem fios.

O primeiro tipo são os sistemas integrados por fios, ou seja, todos elementos básicos da aplicação são interligados de forma a criar uma rede de dispositivos do sistema. Algumas das vantagens de se utilizar tal tipo de projeto é o fato de o sistema ficar mais robusto e não depender de uma rede *wireless* para seu funcionamento. Por outro lado, a manutenção preventiva e corretiva do sistema pode ser mais complexa comparada a um sistema sem fio, já que muitas aplicações e, principalmente, na aplicação de Automação Residencial, pode ser extremamente custosa dependendo das dimensões da residência.

No caso da automação sem fio são utilizados mecanismos de conexão sem fio, por parte dos elementos básicos do sistema, para fazer a troca de informações entre todos os dispositivos conectados. Algumas vantagens de se utilizar esse tipo de solução é o fato de obter uma manutenção mais acessível, agilidade na instalação e inserção da tecnologia, controle remoto dos dispositivos e informações em tempo real. Por outro lado, por se tratar de um sistema totalmente dependente da rede de comunicação do ambiente, o sistema pode se tornar ineficiente se a rede de comunicação não estiver bem distribuída e projetada. Para estes casos, existem alguns protocolos de tratamento de erro relacionados a comunicação de dispositivos, um bom exemplo de protocolo de tratamento de erros relacionados a rede de comunicação é o MQTT.

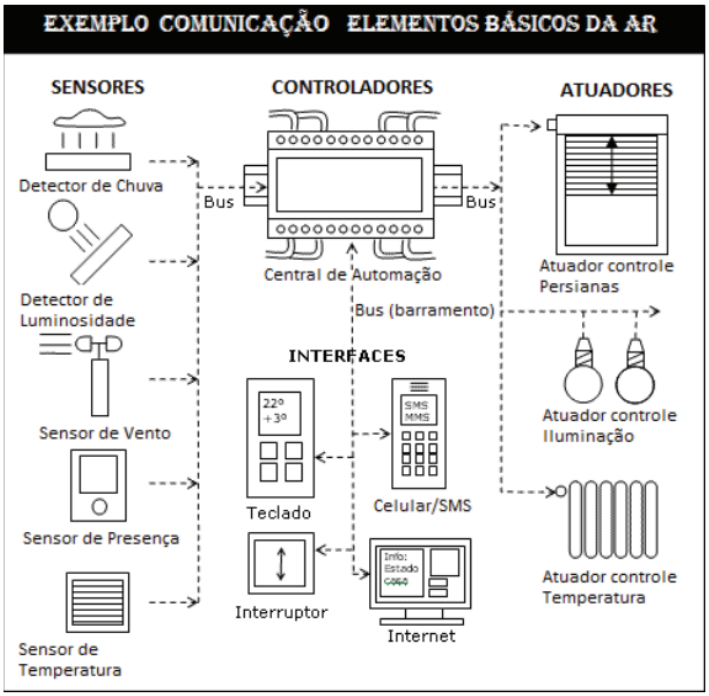
3.2.6 Arquitetura do sistema

Resumidamente, todos estes componentes combinados da forma correta podem trabalhar juntos para fornecer ao usuário a melhor experiência com a automação residencial. Combinando sensores, atuadores e controladores pode-se automatizar diversos processos de segurança, conforto, economia de energia.

Segundo Teza (2002), a metodologia do desenvolvimento de uma solução de automação residencial busca utilizar uma central de comando na qual o usuário terá acesso para controlar os diversos componentes do sistema, como sensores e atuadores.

A Figura 4 apresenta um exemplo de como é feita a comunicação dos elementos básicos de uma arquitetura centralizada de automação residencial. É possível observar, à esquerda, os sensores que são responsáveis por receber as grandezas físicas e transformá-las em eletricidade; no centro, observa-se o controlador que é responsável pela tomada de decisão e pelo controle dos atuadores e sensores, assim como as interfaces de interação com o usuário; à direita, encontra-se os atuadores que interagem direto com os dispositivos elétricos ou mecânicos para controle do ambiente (CASADOMO, 2010).

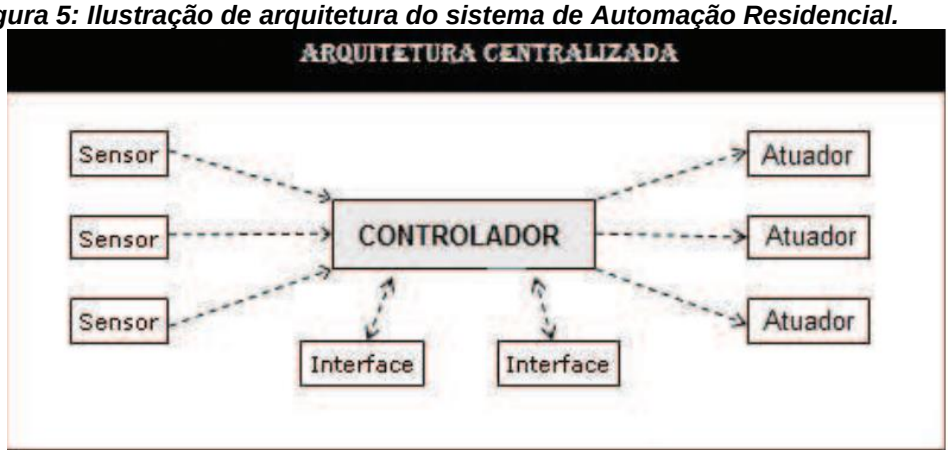
Figura 4: Integração dos componentes da automação residencial.



Fonte: Casadomo, 2010

É possível simplificar um sistema complexo de automação residencial como observa-se na Figura 5, na qual é utilizada uma arquitetura centralizada de automação residencial, em que todos os componentes do sistema estão interconectados por meio de um controlador ou central de controle.

Figura 5: Ilustração de arquitetura do sistema de Automação Residencial.



Fonte: Casadomo, 2010

3.2.7 Armazenamento e processamento em nuvem

Atualmente, é possível armazenar e processar dados utilizando a ferramenta de nuvem. Essa, por sua vez, ajuda muito nas aplicações que utilizam a internet das coisas pela sua praticidade e independência de dispositivos físicos para o armazenamento de dados. Além disso, ela também pode hospedar APIs e desenvolver aplicações diversas utilizando esse processo. O sistema de armazenamento de dados, portanto, pode ser definido como “[...] um serviço da computação em nuvem que está modificando a maneira de usuários finais e empresas utilizarem a tecnologia da informação.” (ANDRADE et al., 2015, p. 4).

Como também afirma Arutynov (2012), a internet tem impulsionado a ascensão das tecnologias de nuvem ou *cloud computing,* a qual representa uma nova forma de potencializar e flexibilizar os recursos da tecnologia da informação (TI). Com isso, é possível se ter o armazenamento de dados online ou em nuvem, oferecido no mercado de forma gratuita (para testes), ou paga por empresas que trabalham via internet. Entre as maiores vantagens dessa ferramenta, ressalta-se a disponibilidade de arquivos em qualquer dispositivo conectado, a facilidade do compartilhamento em grupos, a economia no consumo de recursos de hardware, uma menor incidência de problemas com servidores, entre outros.

Ao se utilizar a nuvem, é possível de se obter diversos benefícios em relação ao armazenamento local, para a aplicação realizada neste trabalho, tem-se como alguns pontos positivos a serem destacados: Maior segurança de dados; automação de processos de armazenamento; estrutura mais flexível; acesso de qualquer lugar.

.

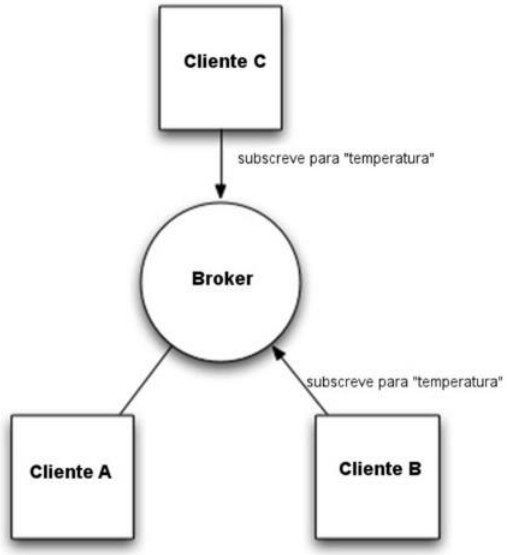
3.2.8 Protocolo MQTT

O protocolo MQTT é uma poderosa ferramenta que auxilia na implementação de novas tecnologias, com sua arquitetura *publish/subscribe,* diversos dispositivos conectados podem receber e enviar informações para uma central em nuvem, por exemplo.

> O MQTT foi criado em meados de 1999 por Andy Stanford-Clark (IBM) e Arlen Nipper (Eurotech). Trata-se de um protocolo de mensagens baseado na arquitetura *publish/subscribe*, voltado para dispositivos restritos e redes inseguras, com baixa largura de banda e alta latência. Segundo a página web do projeto 7, os princípios do design são minimizar os requisitos de recursos de dispositivo e de largura de banda tentando garantir confiabilidade e garantia de entrega. (MARTINS E ZEN, 2015, p. 72)

Segundo Jaffey (2014), o protocolo segue o modelo cliente/servidor. Os dispositivos sensores são clientes que se conectam a um servidor (chamado de *broker*) usando TCP. As mensagens a serem transmitidas são publicadas para um endereço (chamado de tópico) que, inclusive, assemelha-se a uma estrutura de diretórios em um sistema de arquivos, por exemplo, casa/quarto2/temperatura. Clientes, por sua vez, podem se subscrever para vários tópicos, tornando-se assim capazes de receber as mensagens que outros clientes publicam no tópico. Na Figura 6, vemos um broker centralizado com clientes publicando e subscrevendo a tópicos, utilizando-se do protocolo de comunicação MQTT.

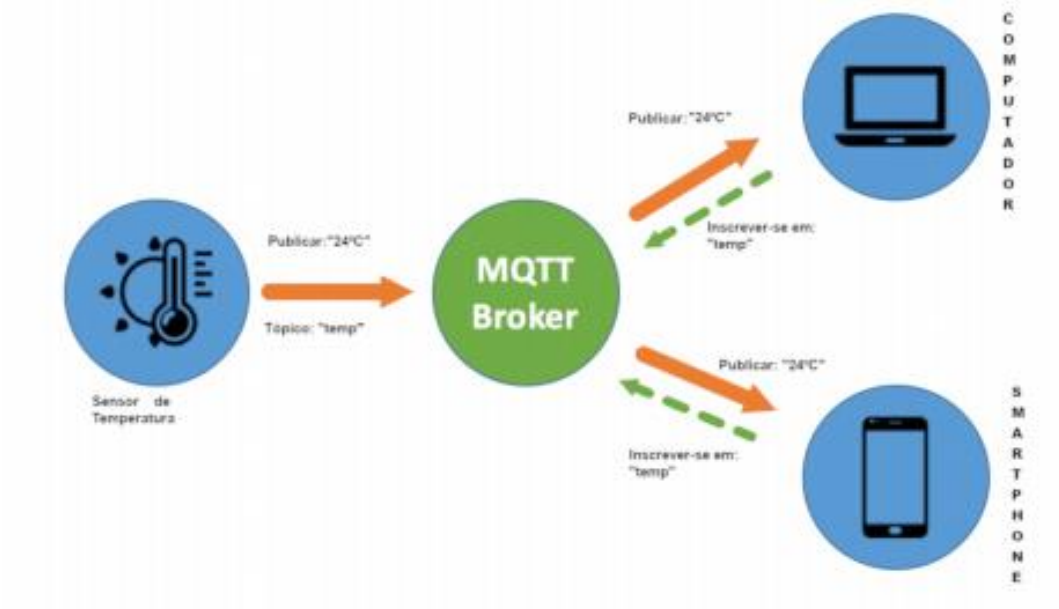
Figura 6: Ilustração protocolo MQTT.



Fonte: Jaffey (2014)

A Figura 7 ilustra o funcionamento da aplicação proposta e os papeis dos principais componentes para a realização dessa aplicação. Um objeto que contém a informação da temperatura publica a informação no broker através do protocolo MQTT. O broker, por sua vez, publica essa informação em todos os clientes que subscreveram o tópico temperatura (temp).

Figura 7: Ilustração de aplicação MQTT.



Fonte: Conceição e Costa (2018)

O protocolo MQTT permite estruturar de forma sucinta o sistema de automação residencial. Utilizando-se de sensores e atuadores para a retenção de informações e controle de cargas, de forma simultânea e proporcionando alta escalabilidade de dispositivos e alta variedade de aplicações, tanto residenciais, quanto comerciais e industriais.

3.2.9 Interface de usuário

A interface de usuário é muito importante em qualquer nova tecnologia, é a janela de comunicação com consumidores e usuários. Para este projeto é necessária uma interface de usuário (aplicativo) bem desenvolvida, para que disponha as informações de forma prática e simples.

Segundo Saccol (2011), os *smartphones* e os aplicativos tem como o objetivo principal atender o acesso das pessoas à informação e ao conhecimento, sem restrições de tempo e de espaço. Derrubando as barreiras de tempo e espaço, eles permitem, também, obter novas formas de comunicação, estas características agregam valores estratégicos para a nova sociedade da Era da informação, na qual se destaca, principalmente a utilização de aplicativos móveis entre a população mundial (BANOS et al, 2015).

3.2.10 Rest API

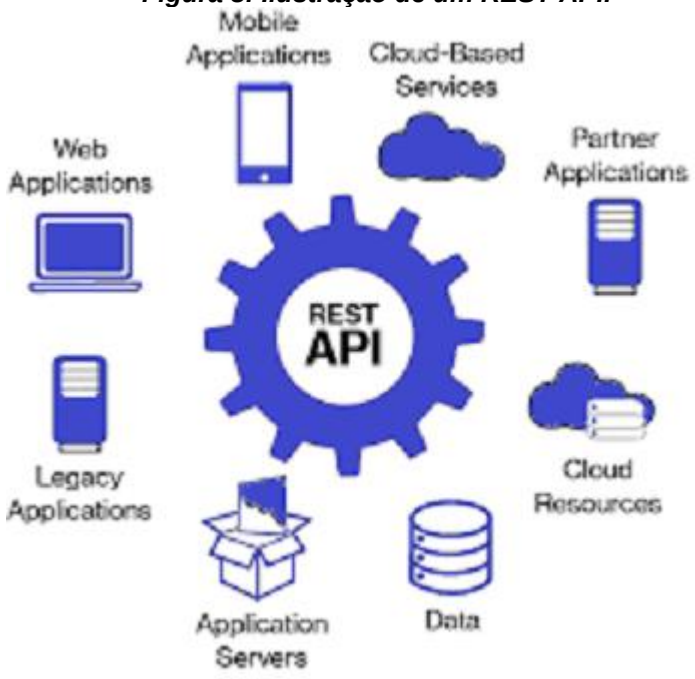
Uma API REST pode promover a distribuição de conteúdo de forma simples para as mais variadas aplicações clientes, desenvolvidas em qualquer linguagem. É uma ferramenta poderosa para novas aplicações. APIs são, em sua essência, um mediador entre os clientes e as informações, recursos e/ou serviços que eles querem obter, mantendo a segurança e o controle ao acesso desse conteúdo. REST não é um padrão ou protocolo, mas um conjunto de instruções e restrições de arquitetura, para que os desenvolvedores possam implementar a API REST de diversas maneiras. Quando o cliente faz uma requisição utilizando uma API REST, essa API entrega uma informação via HTTP utilizando um dos formatos disponíveis: JSON, HTML, XLT, XML entre outros formatos (RED HAT, 2021).

APIs REST são rápidas, leves e escaláveis, o que é ideal para a Internet das Coisas (IoT) e o desenvolvimento de aplicações *mobile* (RED HAT, 2021), aliado à possibilidade de serem implementadas em várias linguagens de programação, torna-se uma ferramenta importante na comunicação entre o servidor web e aplicação móvel desenvolvida neste projeto.

> A programação de *web services* de *Application Programming Interface* (API) que aderem ao estilo arquitetural REST, e de aplicações consumidoras deste tipo de serviços é atualmente muito popular. Por exemplo, aplicações como o Twitter, Instagram, Youtube e Uber, fornecem acesso às suas aplicações clientes através deste tipo de APIs (FERREIRA et al., 2017, p.1).

Para Puluceno (2012), o REST é uma ferramenta que permite solucionar problemas de comunicação entre sistemas, de forma rápida, eficiente e simples. Com sua versatilidade de linguagens de programação, além de tornar acessíveis os métodos para desenvolvimento de aplicações em camadas, torna as aplicações distribuídas. Utilizando-se do estilo REST de arquitetura, a criação de uma API torna-se mais simples do que a desenvolvendo com outras tecnologias como Simple Object Access Protocol (SOAP). A Figura 8 demonstra a REST API e algumas funcionalidades que podem ser implementadas utilizando-a.

Figura 8: Ilustração de um REST API.



Fonte: MACORATTI, 2021.

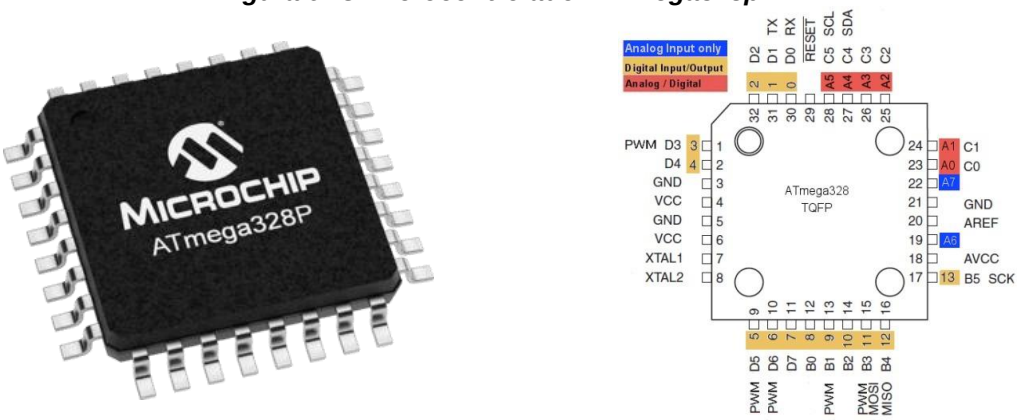
3.2.11 Microcontrolador

Os microcontroladores são utilizados para se programar um dispositivo específico para uma determinada atividade, seja ativar alguma carga, ou realizar cálculos, receber informações e afins. Para o projeto temos a necessidade de se utilizar um destes dispositivos para nossas aplicações.

Segundo Souza (2005) e Camargo (2014), um microcontrolador eletrônico é um dispositivo dotado da capacidade de ser programado, ou seja, tem a possibilidade de realização de controle de processos lógicos variados. Ele possui uma unidade de processamento, memória, interfaces de entradas e saídas digitais e/ou analógicas, temporizadores, contadores, PWM, entre outros periféricos encapsulados em um único chip. Dotado de um consumo de energia elétrica relativamente pequeno e seu design compacto o torna ideal para automação de dispositivos portáteis.

A versatilidade desse componente, faz com que ele possa ser responsável por realizar diversas tarefas, com base em sua programação, do uso de seus periféricos integrados, e da possibilidade de controlar outros elementos externos. Essa versatilidade foi responsável por colocar esse componente na maioria dos equipamentos eletrônicos de consumo utilizados atualmente. Podemos observar o microcontrolador em questão junto com seus pinos de trabalho na Figura 9.

Figura 9: O microcontrolador ATmega328p.

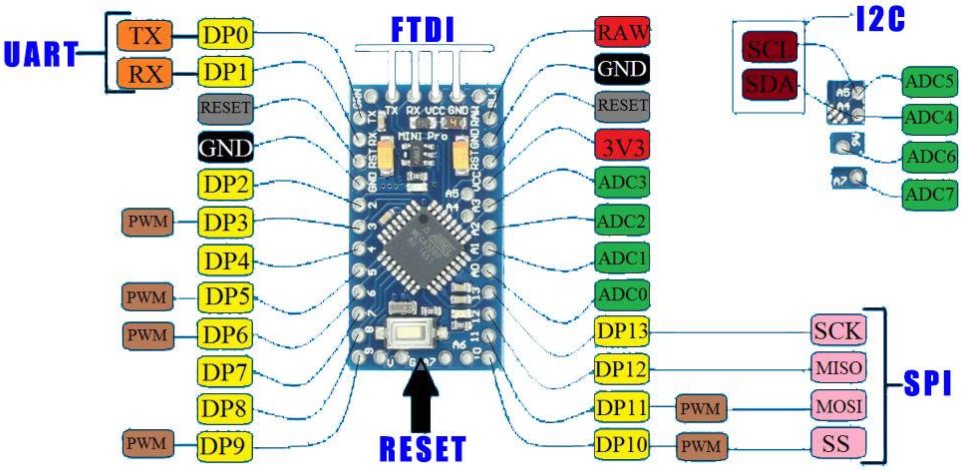


Fonte: Adaptado de Microchip Technology, 2019.

Os microcontroladores com encapsulamento SMD são mais complexos de se utilizar em fase de testes de dispositivos, por esse motivo foi escolhida, então, a placa de desenvolvimento vinculada a este. O Arduino Pro mini demonstra ser uma boa alternativa para prototipagem de dispositivos e testes relacionados a funcionamento. Além de sua flexibilidade de aplicação, mostra-se bem interessante para a o trabalho

a ser desenvolvido. Observa-se alguns aspectos construtivos na Figura 10 e, no quadro, suas respectivas especificações técnicas.

Figura 10: Arduino Pro Mini.



Fonte: electronicsmith, 2020.

Quadro 1: Pinos Arduino Pro Mini.

Função	**Pinos**	**Descrição**
Energia	VCC, GND e VIN	VCC: 5v ou 3.3v GND: fio terra VIN: Até 12V
ADC	0,1,2,3,4,5,6,7	Entradas Analógicas
PWM	DP3, DP5, DP6, DP9, DP10, DP11	Pinos de *Pulse Width Modulation*
Reset	RES	Botão Resetar
Interrupções	T0 e T1	Pinos de interrupção de *hardware*
Comparadores Analógicos	AIN0 e AIN1	Pinos de comparação

Comunicações	UART SPI I2C	UART: DP0-TX, DP1-RX SPI: SCK-DP13, MISO -DP12, MOSI-DP11, SS-DP10 I2C: SCL-ADC5, SDA-ADC4

Fonte: Adaptado de Arduino.cc (2021).

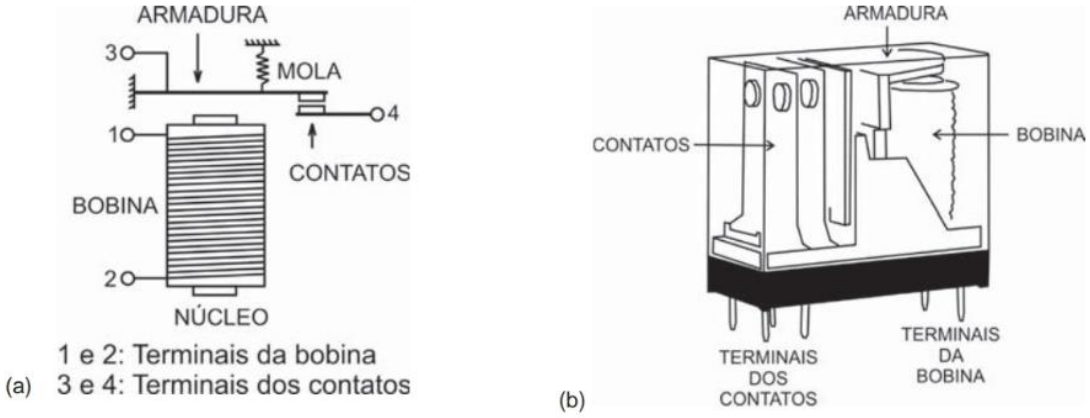
3.2.12 Relés

Segundo Dias (2014), para preservar uma boa segurança operacional, tanto para o sistema de controle, quanto para o sistema manobrado, é necessária a separação entre os circuitos por meio de dispositivos eletromecânicos, adequando valores de tensão e corrente entre o circuito de controle e potência.

Um dos componentes capazes de realizar essa tarefa é o relé eletromecânico, que será utilizado no projeto para realizar a isolação galvânica entre o microcontrolador e a rede elétrica, além de realizar a manobra de cargas.

Como podemos observar na Figura 11, a construção do relé é direcionada para que este seja um componente constituído de uma bobina, que por sua vez quando energizada, forma um campo magnético que desloca o contato móvel em direção ao contato fixo, permitindo a condução elétrica entre os terminais.

Figura 11: Relé eletromecânico visão:(a) esquemática (b) representação física.



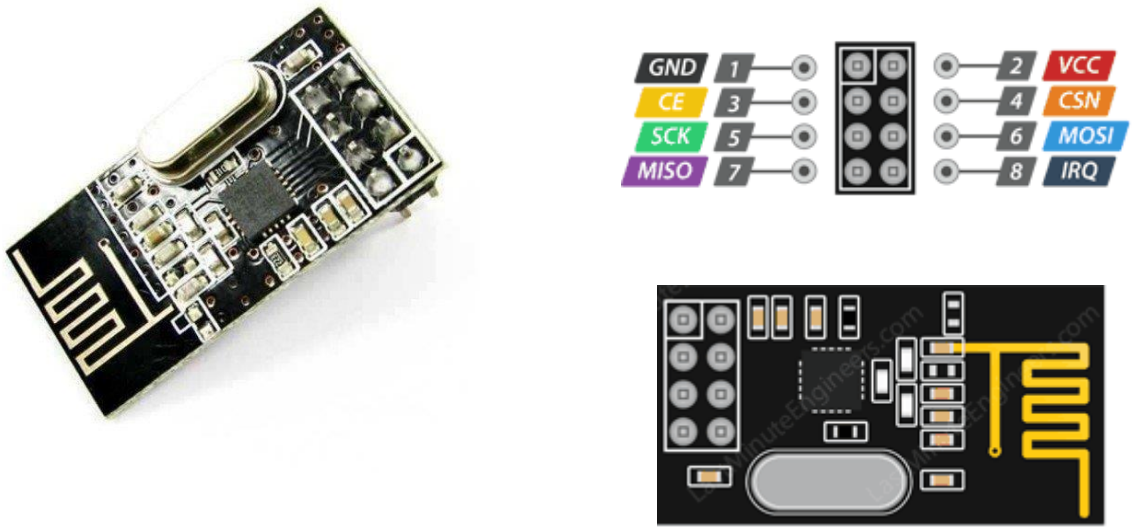
Fonte: Fernandes Filho, Dias (2014).

Ao interromper a queda de tensão, o contato móvel retorna à posição de retorno pela ação da mola, não possibilitando mais a condução elétrica.

3.2.13 Módulo de comunicação sem fio

Segundo a Nordic Semiconductor (2008), o módulo de comunicação NRFL24L01 é um transmissor e receptor de rádio frequência que opera em uma faixa de frequência ISM de 2,4 à 2,5 GHz. Este dispositivo *wireless* integra, em um único chip, um sintetizador de radiofrequências, um amplificador de potência, um modulador e um demodulador. Estes são utilizados para comunicação sem fio. A base de comunicação com microcontroladores é feita a partir de um protocolo de comunicação SPI, sendo possível alterar os registros internos e desta forma controlar a saída de potência, o canal de transmissão, a taxa de transmissão e modos de espera. Podemos observar então o transceptor e seus pinos na Figura 12.

Figura 12: Transceptor nRF24L01.



Fonte: Eletrogate (2019).

Como Thomsem (2014) afirma, o módulo utiliza a frequência de 2,4 GHz e alguns até possuem antenas embutidas, o que torna este módulo um dos mais compactos do mercado. Esses módulos possuem conectores com pinos bem próximos uns aos outros, o que limita seu uso na protoboard, nesta configuração é ideal utilização de algum tipo de conector, como jumpers.

A pinagem do NRF24L01 pode ser descrita da forma:

Quadro 2: Pinos do nRF24L01.

Pino	Nome	Função
1	GND	Terra
2	VCC	Entrada Alimentação +3,3V
3	*CE*	*Chip Enable* RX/TX
4	CSN	*SPI Chip Select*
5	SCK	SPI Clock
6	MOSI	*SPI Slave Data Input*
7	MISO	*SPI Slave Data Output*
8	IRQ	Interrupção

Fonte: Adaptado de Nordicsemi (2021).

3.2.14 Plataforma de desenvolvimento Raspberry Pi

As plataformas de desenvolvimento tem sido uma ferramenta muito útil para desenvolvedores de produtos e de protótipos na área do conhecimento da eletrônica. Os recursos oferecidos por essas são inúmeros e envolvem muitos tipos de comunicação entre dispositivos e componentes eletrônicos.

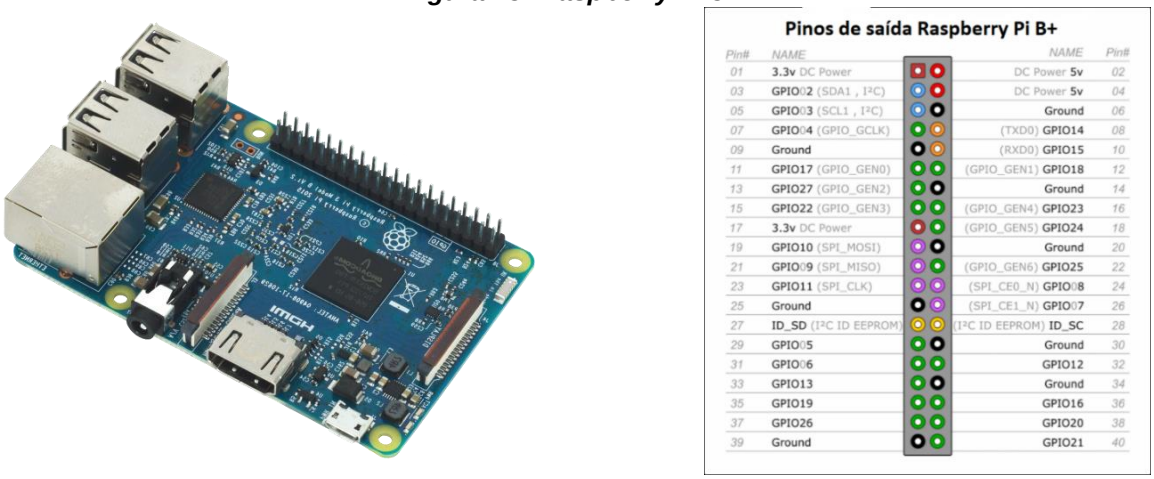
“Um computador de placa única é definido como um computador de uso geral construído em uma única placa com microprocessador, memórias, portas de entrada e saída e outros recursos de um computador funcional” (JOVANOVÍC, et al, 2013, p. 882)[1]. A placa de desenvolvimento Raspberry PI, por sua vez, é definida como um computador de placa única, que possui um microprocessador, memórias, portas de entrada e saída e outros recursos de um computador funcional (JOVANOVÍC, et al, 2013).

Conforme os dados disponibilizados pelo site FilipeFlop, revendedor oficial da Raspberry no Brasil, a placa é um equipamento extremamente versátil para os mais variados projetos. Este dispositivo que está demonstrado na Figura 13, é um minicomputador que roda distribuições Linux, Raspberry OS e Ubunto, mas também

[1] “A single-board computer can be de ned as a general purpose computer built on a single board with microprocessor(s), memories, input/output(I/O) ports and other features of a functional computer.” (JOVANOVÍC, et al, 2013, p. 882)

suporta outros sistemas operacionais como Windows 10 Iot e versões customizadas Linux.

Figura 13: Raspberry Pi 3 B+.



Fonte: Raspberry Pi Foundation (2021).

Segundo Richardson (2013) esse computador de placa única é uma ferramenta de desenvolvimento que possui entradas HDMI, placa de rede, entradas ethernet e/ou wi-fi em uma placa do tamanho de um cartão de crédito.

3.2.15 Fonte de Alimentação

O projeto proposto necessita de um conversor para que seja transformado uma corrente alternada para uma corrente contínua, e assim ser utilizada no restante dos componentes do sistema. Para isso utilizamos do conversor HLK-PM01.

Dada a necessidade de alimentar os módulos atuadores e sensores, o modulo conversor AC-DC HLK-PM01, uma compacta fonte de alimentação, com uma entrada que pode variar entre 100-240 VAC e uma saída fixa de 5V, que fornece uma potência máxima de 3W. Podemos observar as características construtivas da fonte na Figura 14.

Figura 14: Conversor AC-DC HLK-PM01.

Fonte: HI-LINK, 2021.

Quadro 3: Características do módulo HLK.

Característica	Descrição
Tensão nominal de entrada	100-240 VAC
Tensão limiar de entrada	90-264 VAC
Corrente máxima de entrada	≤200 mA
Corrente de partida	≤10 A
Tensão nominal de saída	+5±0,1 VDC
Tensão nominal de saída a plena carga	+5±0,2 VDC
Corrente máxima de saída à longo prazo	≥600 mA
Potência de saída	3 W
Proteção	Sobrecarga e curto-circuito
Eficiência	≥69 %

Fonte: Adaptado de Hlktech (2021).

3.2.16 Sensor de gás

A segurança também é um quesito em um sistema de automação residencial, para isso, uma possibilidade de uso é o sensor de gás de cozinha MQ-2.

O Módulo MQ-2 é um sensor de gases que indica presença de GLP (gás de cozinha), hidrogêneo, metano e propano, além de outros tipos de gases. A medida utilizada por ele é o ppm (partes por milhão), que varia de 30 a 10.000. A responsabilidade pela leitura e envio de informações para o microcontrolador fica com o chip LM393. A Figura 15 demonstra o sensor e seus pinos.

Figura 15: Modulo de detecção de gás MQ-2.

Fonte: Adaptado de Eprolabs (2021).

As especificações técnicas de funcionamento mais relevantes para o projeto são:

Quadro 4: Características do módulo MQ-2.

Característica	Descrição
Energia (Fonte)	5V
Sensibilidade	Gases
Tipo de Saída	Analógica ou Digital
Compatibilidade	Arduino e Arduino IDE

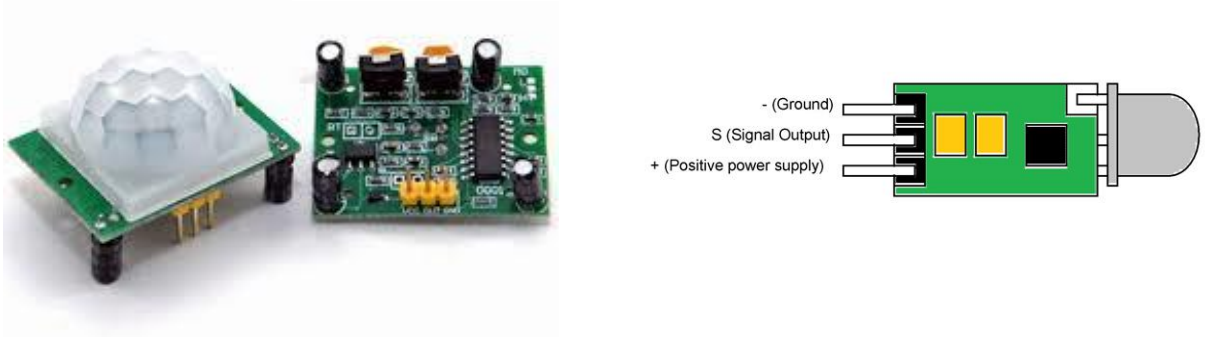
Fonte: Adaptado de Mouser (2021).

3.2.17 Sensor de movimento

O PIR (*Passive Infrared*) são sensores que permitem a detecção de movimento, quase sempre, utilizados para detectar se um ser humano se moveu para dentro ou fora do alcance do sensor. Eles são pequenos, de baixo custo, baixa potência e fáceis de se utilizar.

Os PIRs são, basicamente, feitos de um sensor piroelétrico que pode detectar níveis de radiação infravermelha. Tudo emite um pouco de radiação de baixo nível e quanto mais quente alguma coisa mais radiação é emitida. O sensor em um detector de movimento é dividido em duas metades. A razão para isso é que se está procurando detectar movimento (mudança) e não níveis médios de infravermelho. As duas metades são conectadas de modo que se cancelem. Se uma metade vê mais ou menos radiação infravermelho que a outra, a saída oscila para de nível lógico alto ou para nível lógico baixo, como podemos observar na figura 16 com suas características construtivas e pinos de trabalho.

Figura 16: Sensor de movimento PIR.



Fonte: Adaptado de FilipeFlop (2021).

As especificações técnicas relevantes para entendimento deste sensor são:

Quadro 5: Características do módulo PIR.

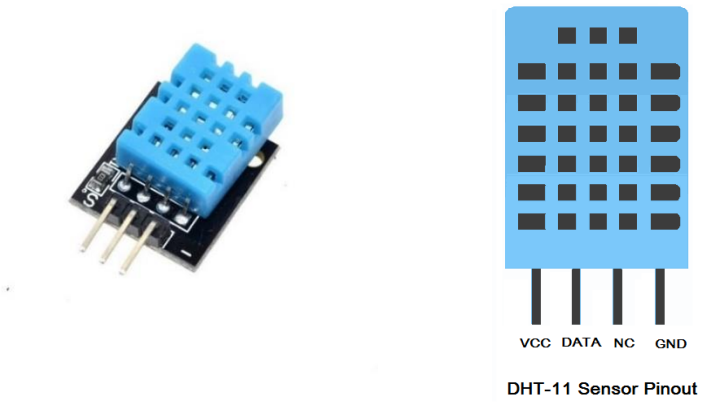
Característica	Descrição
Energia (Fonte)	5V a 12V
Sensibilidade	Movimentação
Ângulo de visão	110º x 70º
Tipo de Saída	Digital
Compatibilidade	Arduino e Arduino IDE

Fonte: Adaptado de FilipeFlop (2021).

3.2.18 Sensor de temperatura e umidade

O sensor de umidade e temperatura DHT11 tem a função de fazer a leitura de umidade e temperatura e informar a um microcontrolador. Ele possui saída de sinal digital, garantindo alta confiabilidade e estabilidade a longo prazo. O elemento sensor de temperatura é um termistor do tipo NTC e o sensor de Umidade é do tipo HR202. O circuito interno faz a leitura dos sensores e se comunica com um microcontrolador de alta performance, oferecendo excelente qualidade, baixo tempo de resposta e ótimo custo-benefício. A Figura 17 demonstra o sensor e seus pinos de trabalho.

Figura 17: Sensor de temperatura DHT-11.



Fonte: Adaptado de Robu (2021).

As especificações técnicas relevantes para o entendimento e utilização deste sensor são:

Quadro 6: Características do módulo DHT11.

Característica	Descrição
Energia (Fonte)	3V a 5V
Sensibilidade	Umidade do ar e Temperatura
Sensibilidade de temperatura	0ºC a 50ºC
Sensibilidade de umidade do ar	20% a 90%
Tipo de Saída	Saídas Analógicas
Compatibilidade	Arduino e Arduino IDE

Fonte: Adaptado de Adafruit (2021).

3.2.19 Sensor de Corrente

Este sensor usa o efeito hall para detectar o campo magnético gerado pela passagem de corrente, gerando na saída do módulo uma tensão proporcional de 66mV/A. O sensor também pode medir correntes entre -30 e +30A de maneira fácil e segura, sendo do tipo invasivo, ou seja, é preciso interromper o circuito para realizar a medição. O sensor de corrente ACS712 pode ser utilizando com corrente alternada (AC) e corrente contínua (DC). Os bornes de ligação são completamente isolados da saída para o microcontrolador. A Figura 18 demonstra as características construtivas do dispositivo.

Figura 18: Sensor de Corrente.

Fonte: Adaptado de Sparkfun (2021)

As especificações técnicas relevantes para o entendimento e utilização deste sensor são:

Quadro 7: Características do módulo ACS712.

Característica	Descrição
Energia (Fonte)	3V a 5V
Sensibilidade	66mV/A
Tipo de Saída	Saídas Analógicas
Compatibilidade	Arduino e Arduino IDE

Fonte: Adaptado de Adafruit (2021).

3.3 Metodologia

A metodologia deste trabalho é baseada em etapas, ou seja, foram divididos os trabalhos de acordo com a metodologia padrão para implementação de um protótipo. A cronologia do trabalho pode ser entendida dividida da forma:

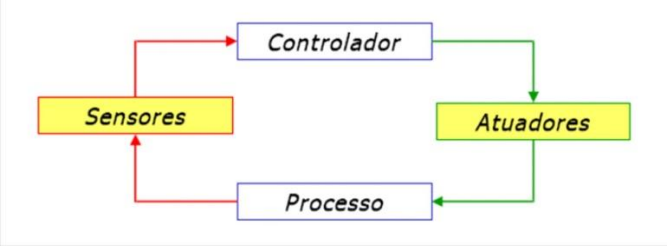
- Etapa Teórica: Estudo de tópico, levantamento bibliográfico, busca de conteúdo, pesquisa mercadológica, seleção de componentes;
- Etapa de Implementação: Compra de componentes, prototipagem por *protoboard*, verificação de comunicação, verificação de funcionamento do protótipo;
- Etapa de Prototipagem: Junção de componentes, solda e construção de *endpoints*;
- Etapa de Verificação: Verificação de conexão, verificação de funcionamento, verificação de qualidade de resposta.

Este trabalho buscou desenvolver um dispositivo que pudesse contemplar as necessidades de automação residencial de um usuário final, sem que esse tenha que realizar modificações estruturais em sua residência. A implementação de periféricos *PnP* disponibiliza um conceito de modularidade ao projeto e garantirá ao usuário que possa montar seu sistema de automação residencial conforme suas necessidades. Para maior conforto e praticidade, o sistema de automação residencial pode ser controlado e monitorado pelo usuário por um *smartphone*, atendendo não só as necessidades de controle automático, mas, também, a possibilidade de definir parâmetros e monitorar o sistema de qualquer lugar com acesso à Internet.

O projeto em questão é dividido em módulos, ou seja, são módulos específicos do sistema para cada tipo de aplicação. Cada módulo por sua parte utiliza-se de um controlador, um sensor e/ou um atuador. Em alguns casos pode se haver os três elementos em um mesmo módulo e, em outras, apenas o sensor enviando informação ou o atuador recebendo informação. Por exemplo, no caso de controle de temperatura de um ambiente, é possível mensurar a temperatura (sensor) e, ao mesmo tempo, utilizar-se de um aquecedor (atuador).

Na Figura 19, tem-se todo o processo de controle de variáveis ambientais, ou seja, todo o ciclo de recepção e controle de alguma carga específica.

Figura 19: Processo de automação e controle.

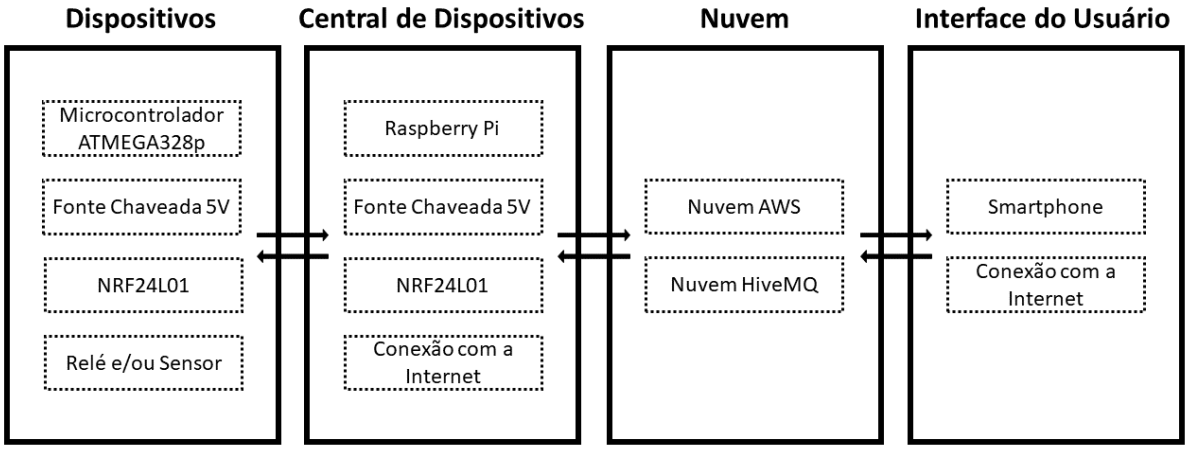


Fonte: clpredes (2010).

- Controlador: parte do módulo que controla os outros elementos básicos, recebendo ou enviando informações de acordo com a decisão do usuário ou inteligência de *software*;
- Atuadores: parte do módulo responsável por atuar diretamente no ambiente, como exemplo podemos citar a iluminação, refrigeração, aquecimento, ativação de cargas;
- Processo: o processo pode ser definido como a forma do atuador de interferir no ambiente ou na grandeza física influente do ambiente, acionando algum circuito de controle o atuador entra em um processo;
- Sensor: é a parte responsável por mensurar a grandeza física ou grandeza desejada para enviar informações ao controlador.

A Figura 20 demonstra a arquitetura citando todos os componentes de cada parte da comunicação.

Figura 20: Previa do sistema implementado.



Fonte: Os autores (2021).

Para o projeto foi escolhido o microcontrolador ATmega328p, da família AVR, produzida atualmente pela Microchip. Este microcontrolador utiliza arquitetura RISC de 8 bits, alto desempenho, ideal para aplicações de baixa e média complexidade. Ele conta com 32 kB de memória flash, 2kB de SRAM, 1kB de EEPROM, Contadores de 8 e 16 bits, 6 saídas PWM, 10 entradas analógicas, SPI, I2C, USART, entre outras diversas funcionalidades.

Durante o desenvolvimento do escopo do projeto foram realizados estudos e pesquisas sobre tecnologias de comunicação sem fio para a implementação que será responsável pela troca de informação entre a central de controle e os periféricos controlados, além de uma pesquisa sobre os modelos de microcontroladores e computadores de placa única, disponíveis no mercado para implementação dos periféricos e do dispositivo central, respectivamente. Neste estudo, a seleção de tecnologias levou em conta o fator qualidade, o custo, a facilidade do uso e, principalmente, a possibilidade de adquiri-las no mercado nacional.

A lista de componentes que utilizados no protótipo está à disposição nos quadros e dividida por partes do sistema. São elas: central, atuadores e sensores.

Quadro 8: Prévia de protótipo de atuador.

Módulo Atuador	
Componente/Módulo	**Função**
Arduino Pro Mini (Atmega328p)	Controle de Atuador
Módulo nRF24L01	Comunicação RF
Regulador de tensão	Alimentação de Atuador
Relés	Ativação de Cargas

Fonte: Os autores (2021).

Quadro 9: Prévia de protótipo de Central.

Central	
Componente/Módulo	**Função**
Raspberry Pi 3 B+	Central de controle
Módulo nRF24L01	Comunicação RF

Fonte: Os autores (2021).

Quadro 10: Prévia de protótipo dos Sensores.

Módulo Sensor	
Componente/Módulo	**Função**
Sensor de gás	Detecção de gás
Sensor de umidade/temperatura DHT11	Detecção de umidade do ar e temperatura
Sensor de presença e movimento	Detecção de presença e movimento

Fonte: Os autores (2021).

Como pode-se observar, o sistema é modular, dividido em módulos para melhor ilustração e demonstração do sistema. A metodologia adotada para o presente trabalho é organizada de forma a especificar quais os respectivos objetivos de cada parte do processo.

3.3.1 Planejamento

Inicialmente é realizado o estudo e pesquisa em relação aos componentes eletrônicos e placas de desenvolvimento que seriam utilizadas no projeto, procurando sempre buscar componentes mais acessíveis e de qualidade para o protótipo.

Em um segundo momento foram definidos os recursos necessários para elaboração do trabalho proposto, com a finalidade de pontuar questões de funcionamento e de acoplamento de tecnologias. Nesta parte, busca-se diferenciar recursos na área da eletrônica para desenvolvimento de dispositivos, tanto quanto ambientes de desenvolvimento e ferramentas a se utilizar. Para esta primeira parte de projeção foram feitas consultas com profissionais da área para especificação de tecnologias mais suscetíveis ao acoplamento.

Em seguida, foram definidos todos os componentes e placas de desenvolvimento que seriam utilizadas para o desenvolvimento do trabalho. Nesta etapa é imprescindível que as tecnologias envolvidas sejam compatíveis e que haja

um funcionamento satisfatório para o dispositivo proposto. Alguns requisitos para o desenvolvimento do protótipo são necessários para a o bom funcionamento do protótipo, portanto, é nesta etapa que será feita uma pesquisa mercadológica a fim de definir a melhor relação custo-benefício para o desenvolvimento do dispositivo.

Dada a necessidade de implementar um protótipo automação residencial, atuando para o controle de cargas, coletando dados relativos ao consumo elétrico e outras variáveis de ambiente, foram desenvolvidos equipamentos para essas finalidades. Estes também podem ser chamados dispositivos *endpoints*, que são os atuadores e os sensores que integram o protótipo, designados para atuar nas cargas e receber informações dos sensores.

O sistema é dividido em módulos, ou seja, para que seja de fácil demonstração, foram utilizados dois dispositivos padrão para o recolhimento e o envio de dados. Os dois dispositivos possuem a função de demonstrar a implementação prática de atuadores e sensores na automação residencial.

Nesta seção, são apresentados os componentes de *hardware* e *software* do sistema de automação residencial, suas devidas ligações e seu funcionamento.

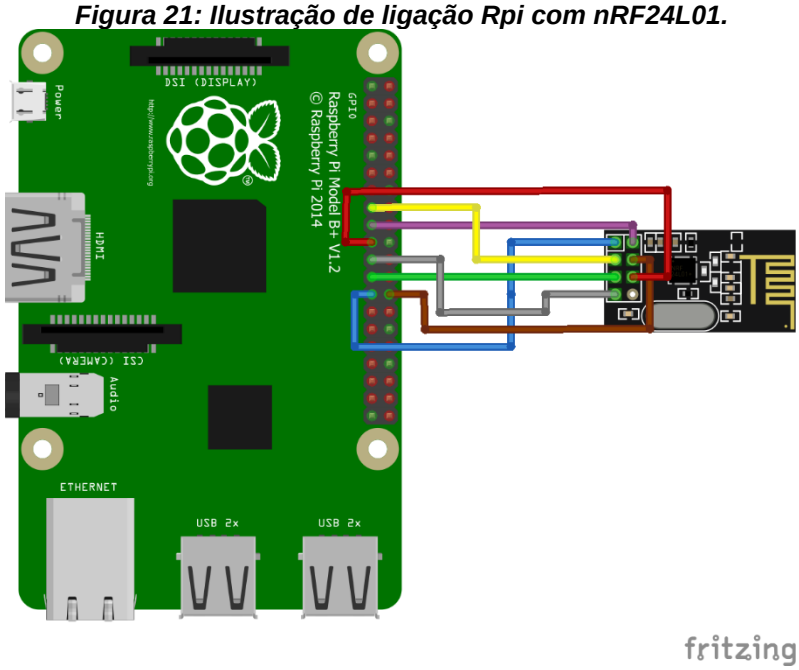
3.3.1.1 Central de dispositivos de automação

A central de dispositivos da automação residencial é responsável por interligar usuário e os dispositivos *endpoints,* ou seja, serve como ponto central dos dispositivos conectados. De acordo com cada aplicação, tem-se os diferentes tipos de dispositivos *endpoints* como, sensores e/ou atuadores. Composto por uma plataforma de desenvolvimento Raspberry PI e um módulo transceptor para comunicação sem fio, esta central estará conectada à rede de Internet para que possa enviar os dados para a nuvem.

As funções da central são: Ponte de dispositivo para internet; organização de dados recebidos; testes de resposta; u*pload* de informações.

Utilizando a ferramenta de simulação de circuitos e ligações Fritzing*,* foi possível definir o esquema de ligação da central de automação, como demonstrado na Figura 21. Nesta imagem observamos nossa central interligada com seu transceptor, que faz a comunicação com nossos *endpoints*, ainda temos a participação de uma fonte para alimentação para a central Raspberry PI 3B+.

Figura 21: Ilustração de ligação Rpi com nRF24L01.



Fonte: Os autores (2021).

Quadro 11: Ligações da central.

Pino	NRF24L01	Pinos RPi 3 B+
1	GND	25
2	VCC	17
3	CE	15
4	CSN	24
5	SCK	23
6	MOSI	19
7	MISO	21
8	IRQ	NC

Fonte: Adaptado de Nordicsemi (2021)

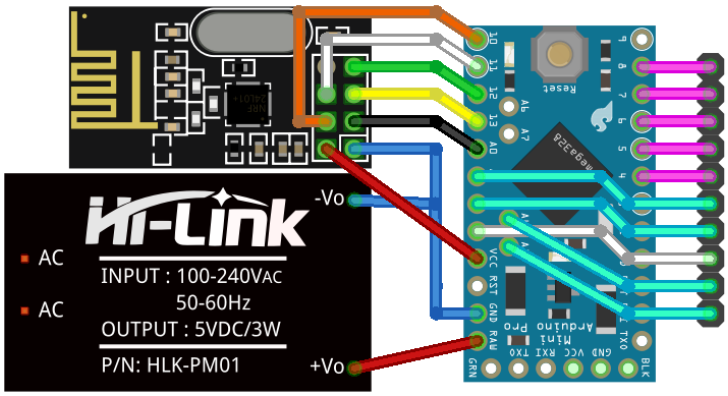
3.3.1.2 Núcleo de dispositivos *endpoints*

O núcleo dos dispositivos *endpoints* é a parte central do sistema, e a partir dela desenvolveu-se aplicações diversas, utilizando atuadores (com controle de cargas) e sensores (com recebimento de informação). Este dispositivo serve de base para que sejam agregados sensores ou atuadores a este. Composto por um conversor AC/DC

de 5V, um Arduino Pro Mini 3.3V e um transceptor nRF24l01, estes são os principais componentes para uma aplicação genérica e implementação de hardware.

Neste esquemático de ligação da Figura 22, podemos observar uma fonte para conversão de 127V AC para 5V DC mais um Arduino Pro Mini e um transceptor. Estes componentes são a base para implementação de todos tipos de *endpoints*, ou seja, utilizando esta base podemos incluir componentes como sensores (A1-A5) e atuadores (D4-D8) para termos o controle sob os mesmos.

***Figura 22: Ilustração de ligação do núcleo dos dispositivos* endpoints.**



Fonte: Os autores (2021).

Quadro 12: Ligação do núcleo dos dispositivos *endpoints*.

Pino	NRF24L01	Pinos Pro Mini
1	GND	GND
2	VCC	VCC
3	CE	A0
4	CSN	D10
5	SCK	D13
6	MOSI	D11
7	MISO	D12
8	IRQ	NC

Fonte: Adaptado de Nordicsemi (2021).

3.3.1.3 Dispositivo *endpoint* atuador

O dispositivo *endpoint* atuador é a parte do sistema que controla cargas, ou seja, ativa e desativa as cargas, assim como mensura seu gasto de energia. Com a

utilização de módulos, foi possível implementar um dispositivo completo para quaisquer tipos de aplicação.

Composto de basicamente pelo núcleo dos dispositivos *endpoints* e por relés, sendo que a quantidade de relés pode ser determinada pelo projetista do sistema, dependendo de sua aplicação. Para melhor representar o sistema é utilizado apenas um relé de carga para aquisição de dados. As funções do dispositivo *endpoint* atuador são: Ativar/Desativar cargas; enviar dados de estado; receber dados relacionados a comandos da central. O *endpoint* atuador trata-se de um núcleo de dispositivos e quatro relés ligados diretamente a saídas digitais do nosso Arduino pro mini, ou seja, são ativadas por níveis lógicos altos ou baixos de acordo com as informações recebidas pela central. A programação utilizada nestes dispositivos foi disposta conforme o apêndice A.

3.3.1.4 Dispositivo *endpoint* sensor

O dispositivo endpoint sensor é o responsável por gerir os sensores aplicáveis a funcionalidades residenciais. Para melhor didática, são implementados todos os sensores em apenas um módulo utilizando plataformas de desenvolvimento. Este dispositivo é responsável por mensurar grandezas físicas do ambiente de forma a informar o dono da residência sobre todas as situações. Diversas aplicações com sensores podem ser implementadas utilizando esse sistema.

Composto de basicamente pelo núcleo dos dispositivos *endpoints* e por sensores, sendo que a quantidade e o tipo de sensores dependem somente da aplicação proposta e pelo projetista do sistema. As funções do dispositivo *endpoint* atuador: Receber informações de grandezas físicas do ambiente residencial e enviar dados relativos aos sensores. O *endpoint* sensor trata-se de um núcleo de dispositivos com sensores ligados diretamente ao nosso Arduino pro mini, ou seja, são enviam informações para nossa central de comunicação. Os sensores utilizados neste foram: sensor de presença, sensor de umidade e temperatura e sensor de gás. A programação utilizada nestes dispositivos foi disposta conforme o apêndice A.

3.3.1.5 Dispositivo *endpoint* de aplicação customizada

É importante notar que existem possibilidades de implementação de dispositivos customizados, ou seja, com mais funcionalidades e mais recursos embutidos. Os exemplares apresentados são apenas uma perspectiva do que é possível realizar. Utilizando-se de sensores e atuadores juntos, pode-se agregar diversas aplicações residenciais, industriais e agropecuárias, a depender das grandezas físicas e cargas a serem controladas. Essas aplicações customizadas utilizarão o núcleo dos dispositivos *endpoints,* e os componentes periféricos necessários para a aplicação definida pelo projetista. O *endpoint* de aplicação customizada trata-se de um núcleo de dispositivos com um atuador e um sensor de corrente ligados diretamente ao nosso Arduino pro mini, ou seja, são enviam informações para nossa central de comunicação e também ativam cargas.

Nesta seção, são apresentados os componentes de *software* do sistema de automação residencial, suas devidas ligações e seu funcionamento.

3.3.1.6 Software da central de dispositivos

O software da central de dispositivos tem como o objetivo gerenciar as comunicações, garantido que as informações que venham dos dispositivos *endpoints* cheguem até o usuário, e que os comandos dados pelo usuário cheguem até o dispositivo *endpoint* desejado. Dado a complexidade, e necessidade de garantir a comunicação, esse software foi dividido em duas camadas de aplicação, sendo a primeira camada, responsável pela comunicação com os dispositivos *endpoints*, e a segunda camada responsável pela comunicação por meio do protocolo MQTT.

A primeira camada do software da central de dispositivos, tem a função de endereçar os dispositivos assim como requisitar informações e modificar *status* dos dispositivos a ela conectados. Alguns exemplos de funções que se pode observar na central são:

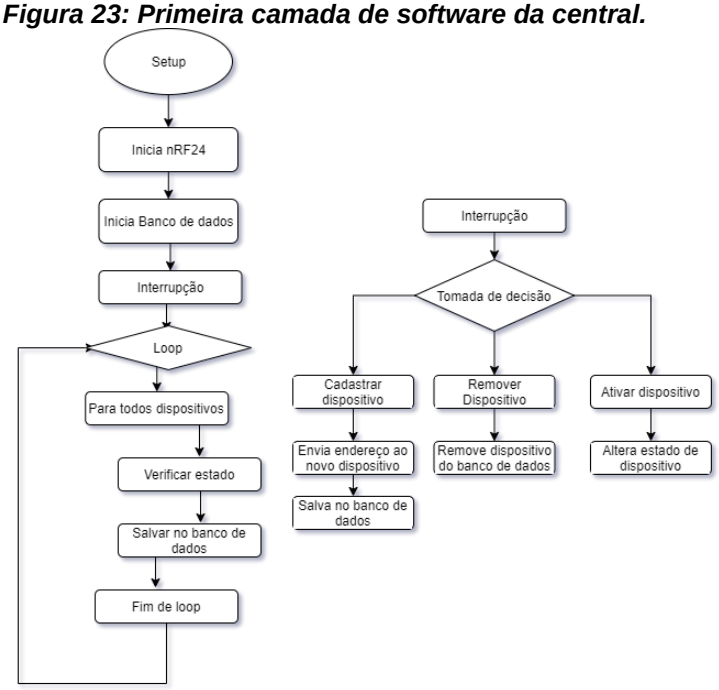
- *SetStatus*: é a função responsável por mudar o estado de um dispositivo *endpoint* atuador. A central de dispositivos utiliza um sistema de endereçamento para saber o dispositivo *endpoint* e o modelo específico a ser conectado e efetuar a troca de estado, desta forma ela envia uma

mensagem para o dispositivo *endpoint* responsável, alterando o estado atual e simultaneamente alterando seu estado no banco de dados;

- *GetStatus*: é a função responsável por receber o estado do sensor ou atuador em questão. A central de dispositivos utiliza um sistema de endereçamento para saber o dispositivo *endpoint* e o modelo específico a ser conectado e efetuado o recebimento de dados por parte deste, desta forma a central de dispositivos recebe dados do dispositivo *endpoint* conectado e altera o seu estado no banco de dados;
- *New Device; Set Adress*: é a função responsável por detectar um novo dispositivo *endpoint* a partir de um endereço padrão pré-definido e cadastrá-lo na lista do banco de dados, para que seja reconhecido posteriormente. Esse comando obtém o modelo do dispositivo a ser cadastrado que irá definir o número de atuadores e sensores que a central de dispositivos irá gerenciar.

De acordo com o fluxograma, podemos observar o funcionamento relacionado as atividades da primeira camada de software da central. A Figura 23 demonstra o fluxo de trabalho.

Figura 23: Primeira camada de software da central.



Fonte: Os autores (2021).

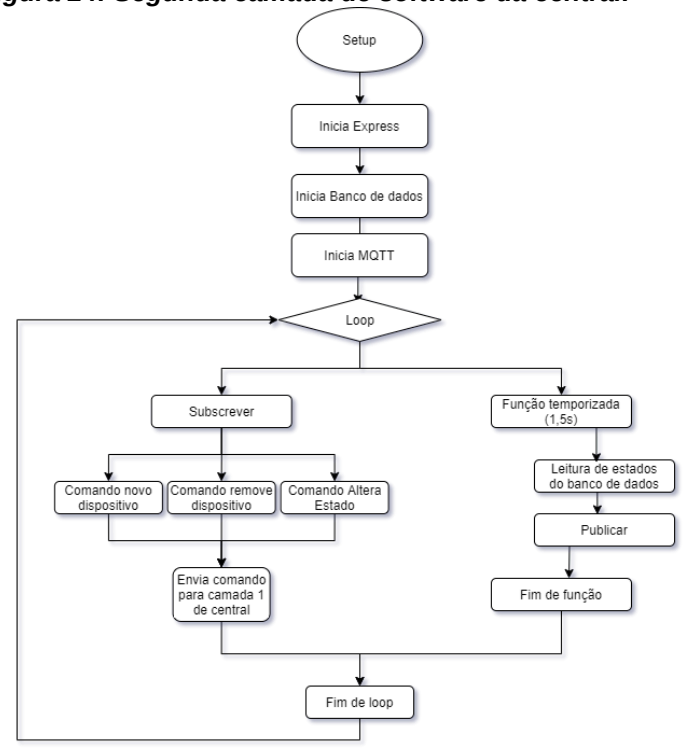
A programação disposta para esta etapa do sistema está disponível no apêndice B do trabalho. Podemos observar então que as atividades relacionadas à camada 1 de software de nossa central tem três funções elementares:

- Verificar os estados dos dispositivos;
- Atualizar o banco de dados;
- Executar comando advindo da Camada 2 por interrupções.

Esta parte da aplicação é responsável por verificar informações sobre todos os *endpoints* conectados ao sistema. Desta forma, o *looping* executado verifica os estados de todos os dispositivos conectados à central, atualizando-os periodicamente. De forma paralela as interrupções podem ser acionadas a qualquer momento durante o funcionamento da aplicação, para assim cadastrar um novo dispositivo, remover um dispositivo ou então mudar seu estado.

A programação disposta para esta etapa do sistema está disponível no apêndice C do trabalho. Na segunda camada da central, temos a outra parte da aplicação, que envolve o protocolo MQTT. Como podemos observar na Figura 24.

Figura 24: Segunda camada de software da central.



Fonte Os autores (2021).

A segunda camada da central de dispositivos realiza a comunicação entre a aplicação da primeira camada e o *broker* MQTT. Ela é responsável pela realização do *publish/subscribe* das informações no servidor MQTT. Essa camada está sempre enviando os dados dos dispositivos *endpoints* ao *broker* (*publish*) e sempre atenta aos comandos que foram emitidos (*subscribe*).

3.3.1.7 *Broker* MQTT

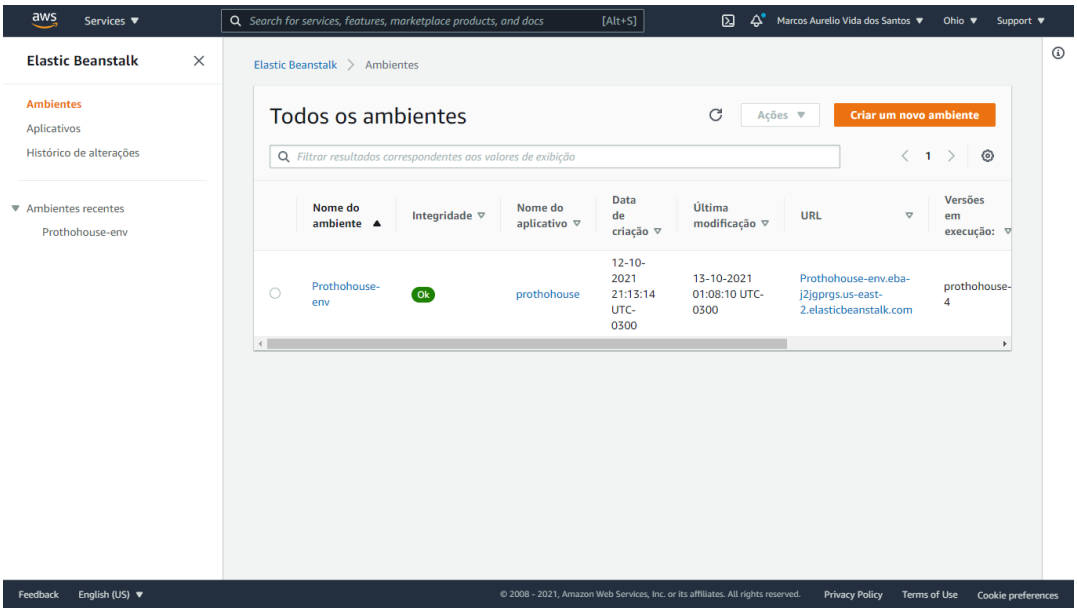
O *broker* MQTT HiveMQ é um serviço na nuvem capaz de habilitar a conexão de até 100 dispositivos gratuitamente. Esse serviço é hospedado na nuvem AWS, e fornece gratuitamente um broker. Neste serviço é possível gerenciar *Clusters*, o número de conexões ativas no momento, definir as credenciais de acesso, assim como obter exemplos que facilitam a implementação do serviço nos diversos clientes e linguagens de programação.

3.3.1.8 Microsserviço hospedado na nuvem AWS

Os serviços de nuvem AWS são de grande importância nessa aplicação. Foi utilizado o serviço AWS Elastic Beanstalk para hospedar o microsserviço responsável por traduzir as mensagens do *broker* MQTT para o protocolo HTTP. Esse microsserviço nada mais é que uma aplicação de software que executa pequenas funções visando um baixo custo computacional.

Para a programação dessa aplicação foi utilizado Node.js, um software que executa códigos JavaScript fora do navegador. Para a comunicação e gerenciamento das requisições da Rest API, foi utilizado o framework Express. A interface do serviço AWS Elastic Beanstalk é bem simplificada e permitiu a implementação do microsserviço em poucos minutos (Figura 25).

Figura 25: Gerenciador de ambientes na AWS Elastic Beanstalk.

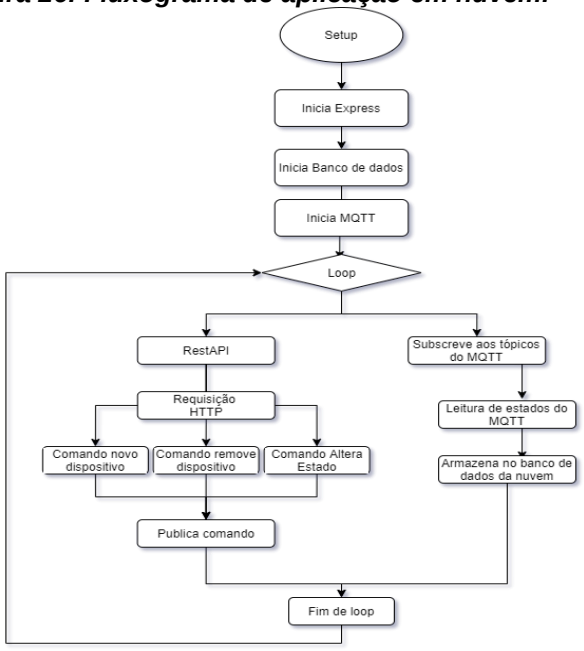


Fonte: Os autores (2021).

Há também uma parte essencial para a aplicação de microsserviço hospedada na nuvem AWS, que é o banco de dados. O serviço Amazon RDS, fornece diferentes bancos de dados para utilização e desenvolvimento de aplicações.

O ambiente em nuvem é uma parte fundamental na nossa aplicação, a partir dela podemos publicar e subscrever a tópicos da nossa central, de forma a executar comandos e receber estados dos *endpoints*. Podemos observar de forma didática como se dá o funcionamento desta parte de acordo com o fluxograma da Figura 26.

Figura 26: Fluxograma de aplicação em nuvem.



Fonte: Os autores (2021).

A programação disposta para esta etapa do sistema está disponível no apêndice D do trabalho.

3.3.1.9 Interface com o usuário

O framework *ReactNative* é uma ferramenta que proporciona uma aplicação multiplataforma, na qual podem ser desenvolvidas aplicações para *Windows*, *Android*, iOS, etc. Com ele é possível programar em JavaScript aplicações mobile nativas.

Foi escolhido pela capacidade multiplataforma, e por dispensar o uso de emuladores pesados no desenvolvimento, pois possibilita a depuração em tempo real em um dispositivo físico.

Seguindo a premissa de terceiros implementar o próprio sistema, haverá uma tela onde o usuário poderá digitar o endereço do servidor do microsserviço implementado por ele. A partir desse endereço o aplicativo desenvolvido se comunica com o microsserviço por meio de requisições HTTP utilizando Rest API. Ele recebe os estados dos dispositivos *endpoint* e envia comandos para eles com um simples toque de um botão.

3.3.1.10 Firmware genérico

O *firmware* genérico foi desenvolvido para ser facilmente manipulado para diversas aplicações. Portanto, o *firmware* genérico foi implementado de forma a funcionar corretamente em qualquer plataforma Arduino para diversas aplicações.

As funções básicas dos dispositivos são: adicionar os dispositivos a rede, checar ou definir estados, informar os valores dos sensores e comunicar-se com a central.

Desta forma, é possível que implementações futuras de dispositivos customizados sejam desenvolvidas a partir de nosso *firmware* genérico. Diversas aplicações que utilizam atuadores e sensores são possíveis com estas configurações.

3.3.1.11 Variáveis de Ambiente: arquivo dotenv

Todas as variáveis de ambiente, ou seja valores dinamicamente nomeados, como nome de servidores e serviços, assim como as credenciais de login nos servidores utilizados ficam guardados em um arquivo .env (*dotenv*). Esse arquivo fica oculto em sistemas Linux, e centraliza as informações do ambiente no qual um processo executa.

Na proposta de projetar um sistema onde terceiros possam implementar o seu próprio sistema de automação residencial, só terão a necessidade de modificar esses arquivos.

Nesses arquivos o projetista irá configurar a sua rota de comunicação MQTT e servidor de banco de dados. Já o servidor do microsserviço é configurado em uma das telas do aplicativo mobile, que faz a interface do usuário.

3.3.2 Descrição do protótipo

O *hardware* é composto por uma central de controle e um ou mais periféricos atuadores, de forma a demonstrar o funcionamento da tecnologia embutida. O desenvolvimento do *hardware* ocorreu de forma modular, onde haverá a central composta pela placa de desenvolvimento Raspberry Pi e pelo transceptor nRF24L01, já os módulos periféricos foram compostos por um microcontrolador ATmega328p, um transceptor nRF24L01, relés ou sensores.

O *software,* que é dividido como é demonstrado no quadro 13, será responsável por integrar todo o sistema e garantir que os módulos periféricos sejam utilizados na modalidade *PnP*, assegurando a praticidade e o conforto ao usuário final. A programação dos módulos periféricos foi desenvolvida em C e C++ utilizando recursos oferecidos pela Arduino IDE e algumas bibliotecas *Open Source* de acordo com o quadro 14. O dispositivo central executa um sistema operacional embarcado Raspberry OS, com rotinas programadas em python.

Quadro 13: Divisão e função de códigos de programação.

Dispositivo	Bibliotecas	Função
Raspberry PI	Python, MQTT, SQLite	Receber comandos e transmitir dados para banco de dados MQTT.

AWS Cloud	PostgreSQL, MQTT, *Rest API*, NodeJS, Express	Microsserviços para aplicativo, banco de dados, protocolo MQTT.
Endpoint (Arduino)	C e C++	Aquisição de dados, transmissão de dados e controle de cargas.

Fonte: Os autores (2021).

Quadro 14: Bibliotecas utilizadas no projeto.

Dispositivo	Biblioteca
nRF24l01	RF24 (Autor: TMR20 e Avamander. Versão 1.4.1)
DHT11	DHT *Sensor Library* (Autor: Adafruit. Versão 1.4.2)
Acs712	ACS library for Arduino (Autor: Rob Tillart. Versão 0.2.4)
Arduino	SPI (Autor: Arduino. Versão 1.2.4)
Arduino	EEPROM (Autor: Arduino)

Fonte: Os autores (2021).

Na Figura 27, observamos o processo geral juntamente com os componentes de cada parte do nosso sistema.

Figura 27: Sistema proposto.

Fonte: Os autores (2021).

O funcionamento de todo o sistema dependerá da distância em que os atuadores e sensores estiverem da central, tanto quanto questões de conexão e estabilidade da rede. Para fazer a interface da central de controle com a internet foram utilizados serviços disponibilizados pela HiveMQ. Executando micro serviços através da nuvem da AWS foi possível utilizar alguns serviços de armazenamento e interfaceamento com API.

3.3.3 Descrição do processo

Para o melhor entendimento de todo o processo foi desenvolvido um fluxograma referente à comunicação entre os serviços de nuvem e o smartphone, para que assim fosse mais bem entendido todos os componentes do sistema. Podemos observar na Figura 28 o processo de comunicação entre nuvem e smartphone.

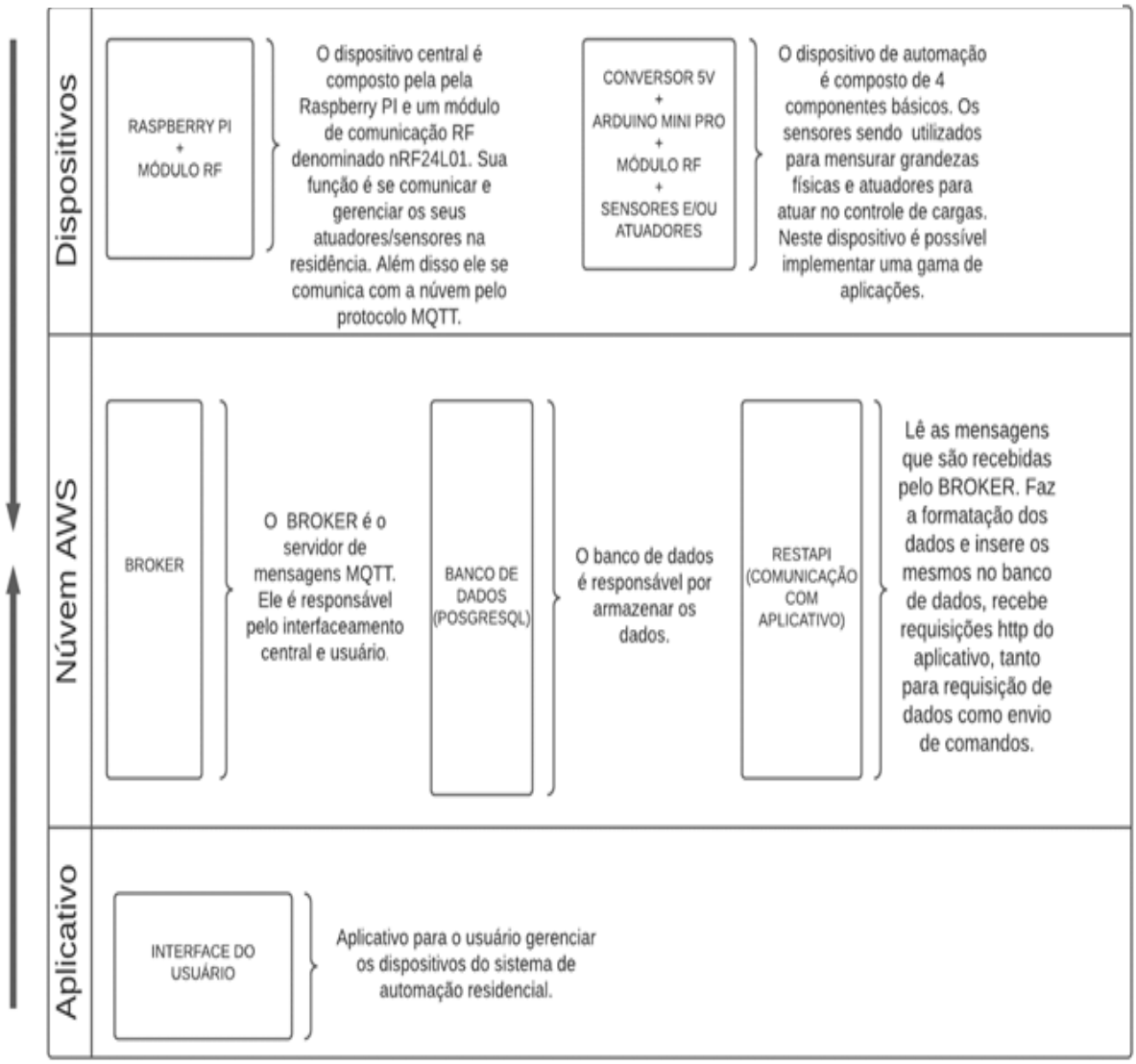
Figura 28: Processo de comunicação nuvem e smartphone.

Fonte: Os autores (2021).

3.3.4 Resumo do sistema completo

Uma outra forma de enxergar o sistema completo é utilizando um diagrama situando cada componente e descrevendo sua função no sistema completo. Para tanto foi desenvolvido um diagrama didático para explicitação de todas as partes envolvidas no processo do protótipo. A Figura 29 demonstra o diagrama desenvolvido com descrição do sistema.

Figura 29: Diagrama didático de protótipo.



Fonte: Os autores (2021).

3.3.5 Resultados e discussões

Um protótipo nada mais é do que uma amostra ou um modelo de um produto feito para testar e para simular um conceito. O objetivo do protótipo é determinar como um produto pode ser feito, qual seu design e a experiência do usuário. É uma etapa importante do projeto, pois possibilita o entendimento dos fluxos de trabalho que são fundamentais para a sua execução, usabilidade e identificação das características especiais a serem incluídas. É, também, um modelo interativo do sistema, que facilita aos usuários visualizarem a experiência de uma melhor maneira. Mesmo não sendo um sistema autônomo, o protótipo dá uma ideia clara do que seria o produto, permitindo, também, que possíveis erros sejam identificados em estágio inicial.

Para verificar o sistema completo, realizou-se uma série de testes para averiguação de erros e para a validação de funcionalidades. Os testes feitos ao protótipo proposto fora:

- Teste da rede: é testada a interconexão entre central e os dispositivos periféricos;
- Teste de conectividade sistema de automação – smartphone: verifica a responsividade do sistema aos controles do usuário;
- Teste de responsividade dos periféricos: testa as funcionalidades do controle e supervisão dos módulos atuadores;
- Teste de tempo de resposta.

Os testes propostos tiveram como objetivo principal determinar a funcionalidade do sistema, ou seja, determinar condições e limites do sistema. Para tanto, é discriminado durante a seção de resultados do trabalho, explicações técnicas e explicitando as metodologias e condições de testes que são utilizadas.

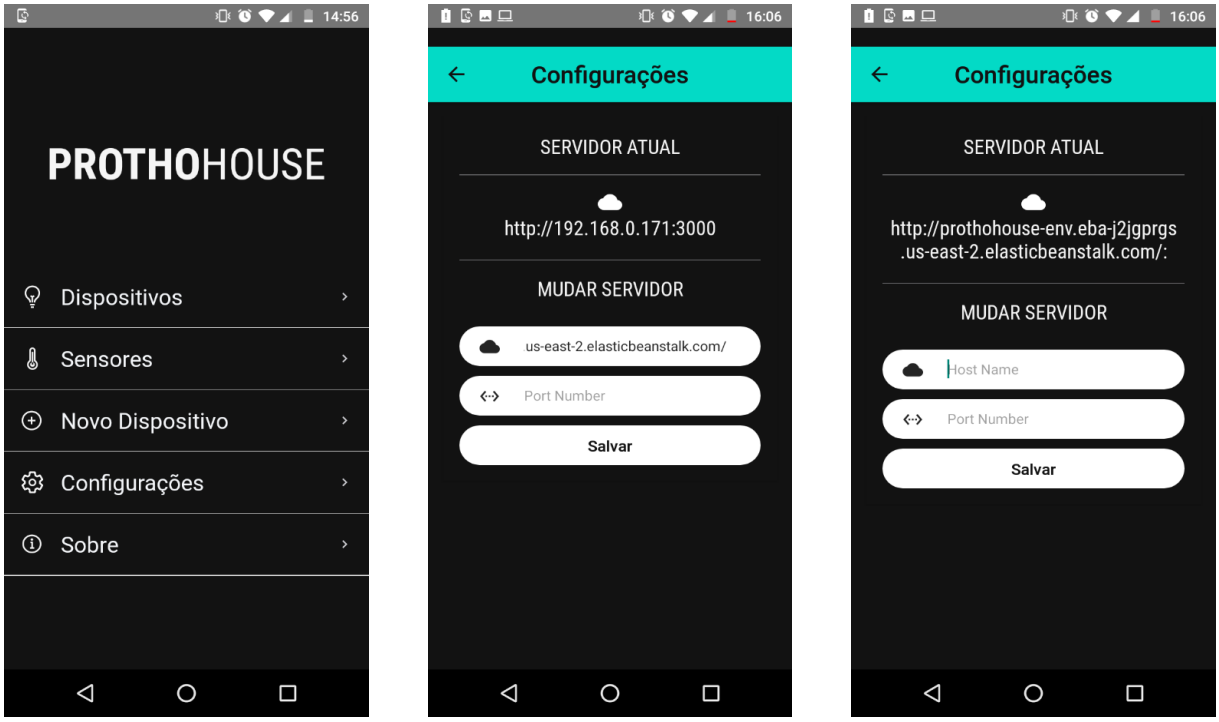
Com o resultado de cada teste pode-se ter um *feedback* de quais são os pontos fortes e discretos do dispositivo e quais as possíveis melhorias que devem ser propostas para o protótipo.

A nuvem AWS é a parte responsável pelo armazenamento de dados, central de API e *Broker MQTT*. Todos dados relacionados a dispositivos conectados ao sistema, como endereço, valor, estado atual e responsividade estão dispostos nesta parte do sistema. Ele é composto por três micro serviços que executam todas as tarefas solicitadas pelo sistema proposto, como execução de API, banco de dados e central broker MQTT.

O aplicativo tem por objetivo ser uma interface usuário-máquina de fácil acesso para que o usuário tenha controle de seus dispositivos conectados, possa cadastrar novos dispositivos e verificá-los periodicamente.

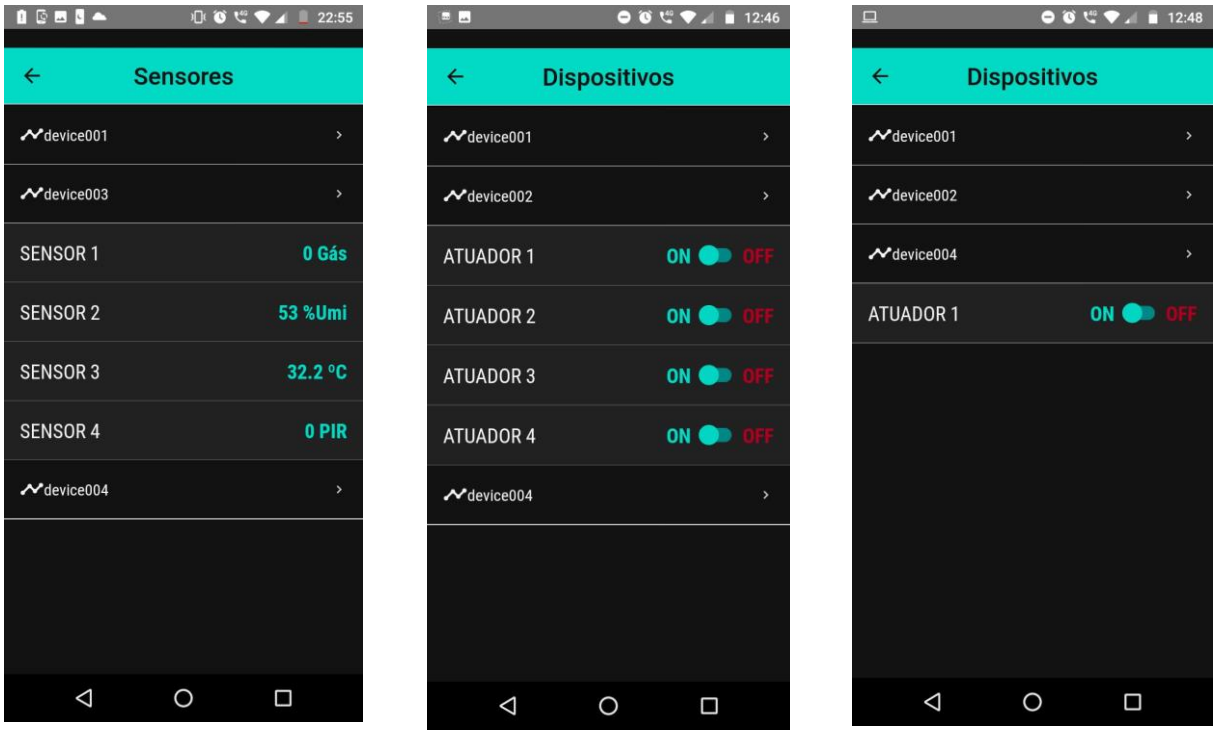
Trata-se de uma aplicação didática para demonstração do protótipo proposto, utilizando de listas, botões e medidas. Este, por sua vez, faz o controle e a verificação direto dos dispositivos a eles conectados. Nas Figuras 30 e 31 podemos observar o aplicativo e suas funcionalidades.

Figura 30: Ilustração de aplicativo e de suas configurações.



Fonte: Os autores (2021).

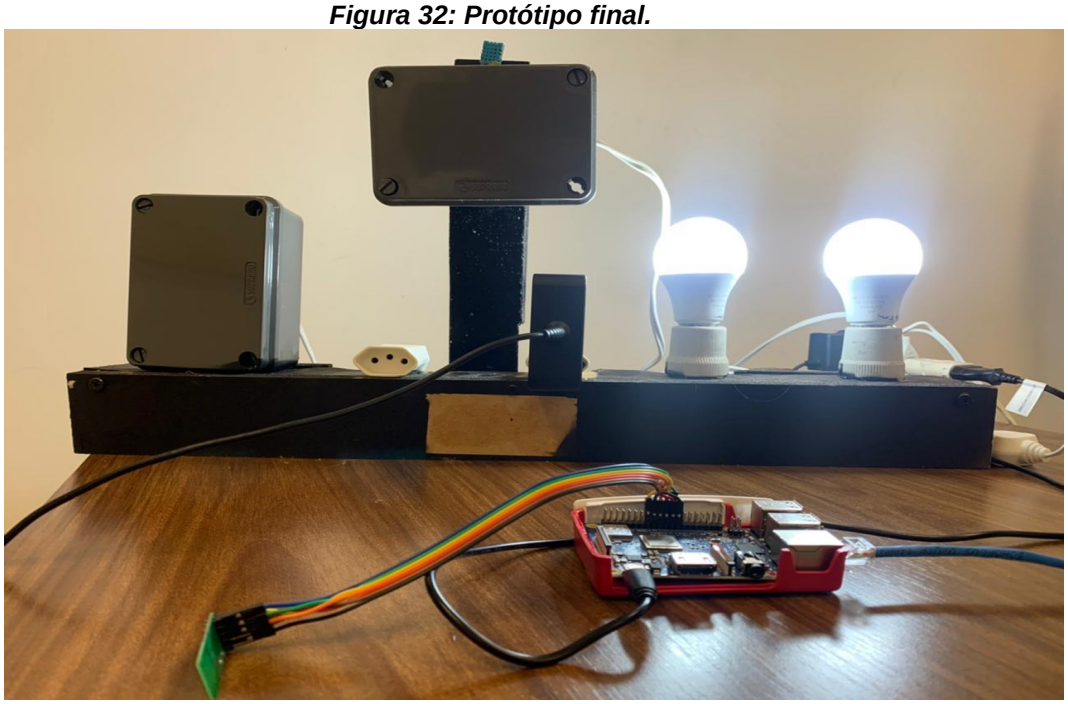
Figura 31: Listas de sensores e atuadores.



Fonte: Os autores (2021).

O protótipo foi montado com fins didáticos, ou seja, utilizamos de duas caixas herméticas separadas para atuadores e sensores, como podemos observar na figura. O posicionamento foi pensado de forma didática, para que fosse apenas verificado seu funcionamento e realizados os testes necessários para uma aplicação real. É importante lembrar-se que para uma aplicação específica de mercado, seria necessário um estudo relacionado a que tipo de carga ou sensor seria necessária para a aplicação. A Figura 32 demonstra a construção dos endpoints, no protótipo temos *endpoints* atuador e sensor, o *endpoint* localizado acima contém um núcleo *endpoint* com sensores de presença, gás, umidade e temperatura. Já o *endpoint* abaixo possui nosso núcleo tanto quanto quatro relés de ativação e desligamento de cargas. Na apresentação contamos também com sensor de corrente montado diretamente na protoboard, junto à um relé para exibir as medições necessárias.

Figura 32: Protótipo final.



Fonte: Os autores (2021).

3.3.5.1Teste de rede

A metodologia utilizada para este teste é feita de forma padrão do mercado da tecnologia *wireless,* utilizando o procedimento de reinicialização da central, desligando-a e ligando novamente, podendo assim verificar os estados de conexão com os atuadores e sensores.

Com este teste podemos observar que a *raspberry* tem uma resposta bem sólida e rápida em relação a sua inicialização, demorando apenas alguns segundos para se conectar à internet. Eventualmente se houver algum problema com o sistema é sempre pertinente utilizar-se de uma conexão *ethernet* bem localizada e centralizada na residência para evitar maiores problemas com relação a eventuais falhas de conexão.

3.3.5.2Teste de conectividade do sistema

A metodologia do teste de resposta é observar as requisições, para testar se existe a possibilidade da central ou de algum dos periféricos demonstrarem disfunções em relação ao sistema como um todo.

A metodologia para tal teste é estruturada utilizando a central do sistema e alguns periféricos, verificando suas respostas diante do aplicativo. Para tanto foi utilizada a função de ativação e desativação de cargas do protótipo. Durante este teste, foi possível verificar que apesar dos periféricos responderem rapidamente ao ligamento dos módulos, a central tem um *delay* de alguns segundos para enviar os dados corretamente para a *Rest API* e enfim para o aplicativo.

3.3.5.3 Teste de resposta do sistema

Para realizar o teste de resposta é utilizada a forma empírica em que foram efetuadas quinze requisições ao sistema à um mesmo dispositivo *endpoint*, para testar se existe a possibilidade da central ou de algum periférico apresentar atraso ou problemas na transmissão de dados. A metodologia utilizada para esse teste é feita utilizando a central do sistema e alguns periféricos, verificando suas respostas diante do aplicativo e da nuvem. Neste teste é observado que os periféricos respondem rapidamente aos comandos dados pela central.

3.3.5.4 Levantamento de recurso financeiro

O levantamento de custo dos componentes foi parte fundamental para entendermos se tal aplicação pode ou não eventualmente se tornar um produto e ser comercializado, para tanto, foi elaborado um quadro para demonstração do recurso necessário para implementação do protótipo.

Quadro 15: Investimentos para construção de protótipo.

Componente	**Componentes**	**Investimento**
Central	Raspberry PI 3 B+	R$ 436,05
	Transceptor nRF	R$ 13,21
Endpoint	Arduino Pro Mini	R$ 39,52
	Conversor AC/DC HLK	R$ 42,66
	Transceptor nRF	R$ 13,21
	Módulo Relé 4 canais	R$ 28,00
	Sensor de temperatura	R$ 13,78

	Sensor de gás	R$ 17,96
	Sensor de movimento	R$ 15,44

Fonte: baudaeletronica (2021)

Nesta consideração de investimento é feito levantamento apenas dos componentes primordiais do sistema, ou seja, foram descartados investimentos em fios, acabamento e estética do protótipo. Como é entendido que as questões de estética, ligações e quantidades de sensores e atuadores, dependem muito da aplicação específica em que o protótipo irá trabalhar.

3.3.5.5Considerações de funcionamento

Com os testes realizados foi dimensionado uma tabela de condições ideais de funcionamento para o sistema, podemos observar a seguir:

Quadro 16: Condições ideais de funcionamento para o sistema.

Condição	**Descrição**
Localização	A condição ideal de funcionamento para a central é em um local centralizado da residência, que pode ser definido pelo estudo da infraestrutura desta. Importante denotar que é necessária análise de infraestrutura.
Raio de alcance entre central e *endpoint*	A condição ideal para funcionamento dos sensores e atuadores é que estejam num raio de 15m de alcance da central (distância medida entre central e endpoint).
Infraestrutura de residência	A infraestrutura de residência ideal foi calculada com base nos valores de raio obtidos nos testes, nos resultando em uma área média de aproximadamente 94 m². Podendo ser aumentada ou reduzida de acordo com variáveis como

	número de cômodos, tubulação, estrutura elétrica, etc...
Conexão com a internet	A conexão com a internet ideal recomendada é utilizando-se cabo ethernet na conexão da central. Caso contrário deve-se analisar variáveis de ambiente para utilização de *wireless*. Recomenda-se que não haja obstruções nem barreiras entre provedor (modem) e central de controle.
Temperatura	A condição ideal de temperatura seria instalar os componentes da central numa temperatura ambiente entre 18º a 27º. Eventuais temperaturas elevadas podem afetar o processamento da *Raspberry pi,* afetando assim suas funções diante do sistema.
Umidade	A condição ideal de funcionamento da central seria umidade baixa (entre 45% e 55%)

Fonte: Os autores (2021).

4 CONCLUSÃO

O projeto proposto oferece uma solução para automatização de processos residenciais. Ele implementa de forma suscinta um equipamento de monitoramento e de controle na rede elétrica de uma residência. Por utilizar-se de recursos de relativo baixo custo e apresentar uma perspectiva de progressão da tecnologia, pode-se concluir que é um projeto viável e de interesse tecnológico-social, no entanto ainda exige muito estudo e desenvolvimento. Com o breve estudo e o desenvolvimento do projeto, foi possível concluir, também, que a demonstração do dispositivo proposto auxiliará e inspirará estudos futuros e futuras aplicabilidades de projeto no envoltório da automação de processos residenciais.

O desenvolvimento de *firmwares* específicos para novas aplicações utilizando como base o protótipo descrito neste documento pode ser um caminho para seguir o avanço tecnológico. Baseando-se em aplicações especificas que dependam apenas de controle de carga, sensores e comunicação RF podem ser desenvolvidos inúmeros dispositivos, não somente para automação residencial, mas também para qualquer ambiente, seja ele industrial, comercial ou residencial.

Finalmente, pode-se entender que o trabalho desenvolvido contribui com a área da engenharia eletrônica pois proporciona uma base de desenvolvimento para aplicações específicas residenciais, condominiais e talvez até aplicações para a universidade. Alguns trabalhos futuros que podemos designar a partir de todo o estudo desenvolvido neste trabalho são:

- Expandir tipos de comunicação entre central e *endpoints* utilizando tecnologias como *bluetooth low energy* , Wi-Fi, SubGhz;
- Implementar funções em tempo real utilizando a plataforma Firebase;
- Gerar componentes para múltiplos estados;
- Temporização de estados;
- Otimização de hardware;
- Otimização de software.

Com o estudo e levantamento realizado nesta pesquisa, foi possível designar algumas ideias para sistemas na área da eletrônica, de forma que este protótipo possa ser utilizado como base. Algumas das ideias são descritas a seguir:

- Sistema de alarme: Com este protótipo é possível que sejam realizados estudos com sensores PIR para implementação de um sistema integrado de

alarme atuando diretamente na segurança da residência. A ideia em questão é de fazer uma distribuição inteligente pela residência e desenvolver um *software* capaz de enviar dados em tempo real para o usuário, para que seja notificado sempre que for infringida algum dos acessos de sua residência. Desta forma então monitorando o acesso de sua residência de forma integrada.

- Sistema de controle de cargas: com este protótipo é possível que sejam realizados estudos de atuadores inteligentes para implementação de um sistema integrado de controle de cargas atuando diretamente na segurança da residência, na economia de energia e na comodidade do controle de ativos da residência. A ideia em questão seria de fazer uma distribuição inteligente pela residência em pontos estratégicos. Uma ideia pode ser a setorização dos cômodos da residência, separando por tipo de carga e aplicação. Deste modo é possível que o usuário final tenha, dentre outros benefícios uma medida precisa sobre seu gasto energético, obtendo-se assim de relatórios importantes para discernir onde é necessária a intervenção para eficiência energética.
- Programação de cargas: com este protótipo é possível que sejam realizados estudos de atuadores inteligentes para programação de um sistema integrado de controle de cargas atuando diretamente na segurança da residência, na economia de energia e na comodidade do controle de ativos da residência. Esta aplicação é mais específica para ambientes comuns, *lobbys* e parques. Utilizando-se da facilidade de controle de cargas que o dispositivo possui e tendo uma função específica de *software* compatível com a API designada, seria possível manter o controle tanto de gasto energético como de utilização das cargas de um ambiente.
- Monitoramento de temperatura e umidade: com este protótipo é possível que sejam realizados estudos para utilizar sensores inteligentes para programação de um sistema integrado de controle de temperatura atuando diretamente na irrigação ou refrigeração do ambiente de aplicação. Utilizando-se da facilidade de sensoriamento que o dispositivo possui e tendo uma função específica de *software* seria possível controlar tanto as aplicações de irrigação como aplicações de refrigeração. Deste modo então

atuando diretamente na temperatura de um ambiente ou até no tratamento de vegetação e afins.

- Sistema de detecção de fumaça: com este protótipo é possível que sejam realizados estudos para utilizar sensores inteligentes para programação de um sistema integrado de detecção de fumaça. Devidamente distribuídos em pontos estratégicos, podemos obter informações do mesmo em tempo real, atuando assim na segurança da residência e no conforto dos moradores. Utilizando-se da facilidade de sensoriamento que o dispositivo possui e tendo uma função específica de *software* é possível que sejam programadas notificações de usuário para acionamento de órgãos responsáveis.
- Automação de iluminação: utilizando-se de sensores fotoelétricos e acionamento de cargas (lâmpadas) é possível centralizar o controle de toda iluminação de uma residência. Viabilizando-se assim um melhor consumo, mais eficiente e ainda possibilitando um maior conforto para o usuário, já que é possível que tenha o controle da programação de horários e funções destes dispositivos.
- Detecção de excesso de som: com esta aplicação e utilizando-se de sensores acústicos é possível nivelar e detectar excessos nas ondas sonoras de uma residência ou condomínio. Esta aplicação pode ser vista apenas como uma comodidade (controle interno) ou pode ser adaptada para questões de segurança em condições extremas (idosos).
- Segurança integrada: utilizando-se de sensores magnéticos é possível detectar portas e janelas fechadas, notificando-se assim o usuário. Desta forma é possível proporcionar uma maior segurança e conforto ao usuário do sistema.
- Detecção de nível de líquidos: Para controles de caixa d’agua, piscina e outras aplicações é possível que sejam utilizados sensores capacitivos integrados ao sistema, desta forma notificando-se o usuário de alguma decisão a ser tomada ou até ativando-se uma carga automaticamente para que o problema seja resolvido.

REFERÊNCIAS

ACCARDI, Adonis. DODONOV, Eugeni. **Automação Residencial: Elementos Básicos, Arquiteturas, Setores, Aplicações e Protocolos**. n. 2, p. 156–166, 2012. Disponível em: http://professor.pucgoias.edu.br/SiteDocente/admin/arquivosUpload/17829/material/ARTIGO02.pdf. Acesso em: 3 Nov. 2020.

ADA, Lady. **PIR Motion Sensor**. Adafruit Learning System. Disponível em: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor. Acesso em: 3 Nov. 2020.

ALVES, J. A.; MOTA, J. **Casas inteligentes.** [S.l.]: Centro Atlântico, 2003.

ANDRADE, A. P. V. *et al.* **Adoção De Sistemas De Armazenamento De Dados Na Nuvem: Um Estudo Com Usuários Finais**. Review of Administration and Innovation - RAI, v. 12, n. 4, p. 04, 2015.

AURESIDE, Associação Brasileira de Automação Residencial. **Relatório Especial: O Mercado de Automação Residencial.** Edição do Autor, 2012. 41 p.

AURESIDE, Associação Brasileira de Automação Residencial. **Especial Tecnologia: O Futuro na Ponta dos Dedos**. Edição do Autor, 2012. 41 p.

ARUTYNOV, V. V. **Cloud Computing: its history of development, modern state, and future considerations**. Scientific and Technical Information Processing, 1(3), 173-178. 2012.

ARDUINO. **Arduino Board Pro Mini.** Disponível em : https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardProMini. Acesso em 30 de outubro de 2021.

BANOS, O. VILALLONGA, C. GARCIA, R. SAEZ, A. DAMAS, M. HOLGADO, J A. et al. **Design, implementation and validation of a novel open framework for agile development of mobile health applications**. Biomed Eng Online [Internet]. Disponível em: http://dx.doi.org/10.1186/1475-925X-14-S2-S6

BOLZANI, C. A. M. **Residências Inteligentes**. [S.l.]: Livraria da Física, 2004.

BOLZANI, C. A. M. **Desmistificando a domótica**. [S.l]: Revista Home Theater, 2007.

CAMARGO, Valter Luís Arlindo de. **Elementos de automação**. 1ª ed., São Paulo: Érica, 2014.

CLPREDES. **Como funciona o CLP?** Disponível em: https://clpredes.wordpress.com/2010/05/. Acesso em: 3 nov 2020.

DEMETRAS, Ezequiel. **[Tutorial] SCT-013 - Sensor de Corrente Alternada com Arduino - Portal Vida de Silício**. Portal Vida de Silício. Disponível em: https://portal.vidadesilicio.com.br/sct-013-sensor-de-corrente-alternada/#Sensor_de_corrente_nao_invasivo_SCT-013. Acesso em: 3 nov. 2020.

DEMETRIO, F. G et al. Análise e implantação da domótica em edifícios residenciais de alto padrão. In**: Congresso de Educação e Ciência para a cidadania global**. XX INIC, XVI EPG, X INIC JR. E VI INID, UNIVAP, 2016, Vale do Paraíba. Anais... Vale do Paraíba: UNIVAP, 2016, p. 1 – 6.

ELETROGATE. **Módulo Wireless Nrf24l01+ 2,4ghz Transceiver Rf**. Disponível em: https://www.eletrogate.com/modulo-wireless-nrf24l01-24ghz-transceiver-rf. Acesso em: 15 nov. 2019.

ELECTRONICSMITH. **Arduino Pro Mini Pinout, Specification and programming.** Disponível em: https://electronicsmith.com/arduino-pro-mini-pinout-specification-and-programming/. Acesso em: 3 de nov de 2020.

EMBARCADOS. **Sinal Analógico vs Sinal Digital.** Disponível em: https://www.embarcados.com.br/sinal-analogico-x-sinal-digital/. Acesso em 10 de nov de 2020.

EPROLABS. **MQ-2 Module.** Disponível em : https://www.eprolabs.com/product/mq-2-module/. Acesso em: 3 de nov 2020.

FERNANDES FILHO, Guilherme Eugênio Filippo; DIAS, Rubens Alves. **Comandos elétricos**. 1ª ed., São Paulo: Érica, 2014.

FERREIRA, F. et al. **Especificação de interfaces aplicacionais rest**. Atas do 9º Encontro Nacional de Informática, INFORUM, 2017.

FILIPEFLOP. **Mini sensor de movimento PIR.** Disponível em https://www.filipeflop.com/produto/mini-sensor-de-movimento-presenca-pir/. Acesso em 22 de agosto de 2021.

HARPER, Richard**. Inside the Smart House**. Londres: Springer, 2003. 264 p.

HLKTECH. **Ultra-small Series Power Module.** Disponível em: http://www.hlktech.net/product_detail.php?ProId=54. Acesso em 30 de outubro de 2021.

JAFFEY, Toby. **MQTT and CoAP, IoT protocols**. 2014. Disponível em: http://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php. Acesso em 16 abril de 2014.

JOVANOVIĆ, P.; MILEUSNIC, M.; PAVIĆ, B..; MIŠKOVIĆ, B. **Applications of the Single Board Computers in the Software Defined Radio Systems**. Sinteza 2014 - Impact of the Internet on Business Activities in Serbia and Worldwide, Belgrade,

Singidunum University, Serbia, 2014, pp. 882-886. doi:10.15308/sinteza-2014-882-886.

LUCAS, Andrew. **Sistema cabeado ou sem fios: qual o melhor para a casa inteligente?** Blogspot.com. Disponível em: http://aureside.blogspot.com/2016/12/sistema-cabeado-ou-sem-fios-qual-o.html. Acesso em: 3 Nov. 2020.

MACORATTI, J. C. **Guia Básico para criar APIs restful.** Disponível em: http://www.macoratti.net/19/04/aspc_gbrst1.htm. Acesso em 23 de agosto de 2021.

MARTINS, I. R.; ZEM, J. L. **Estudo dos protocolos de comunicação MQTT e COAP para aplicações machine-to-machine e internet das coisas**. Americana: FatecAM. V3. N1. p. 64–87, 2015.

MCROBERTS, Michael. **Arduino básico**. São Paulo: Novatec Editora, 2011.

MICROCHIP TECHNOLOGY. **ATmega328P DATASHEET**. San Jose, CA, USA, 2015. Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. Acesso em: 17 out. 2020.

MICROCHIP TECHNOLOGY. **ATmega328P - 8-bit AVR Microcontrollers**. Disponível em: https://www.microchip.com/wwwproducts/en/ATmega328p. Acesso em: 12 nov. 2020.

MOUSER. **MQ-2 GAS SENSOR.** Disponível em https://www.mouser.com/datasheet/2/321/605-00008-MQ-2-Datasheet-370464.pdf. Acesso em 30 de outubro de 2021.

MURATORI, José R.; BÓ, Paulo H. D. **Automação residencial: histórico, definições e conceitos**, Revista O Setor Elétrico, abr. 2011. Disponível em: https://goo.gl/F95edA. Acesso em: 12 out. 2019.

NORDIC SEMICONDUCTOR. **nRF24L01+ Product Specification**. Versão: 1.0. Noruega, 2008. Disponível em: https://www.nordicsemi.com/-/media/DocLib/Other/Product_Spec/nRF24L01PPSv10.pdf. Acesso em: 10 nov. 2019.

NORDIC SEMICONDUCTOR. **nRF24 Series – 2.4 GHz system-on-chip transceiver and audio streamer.** Disponível em https://www.nordicsemi.com/products/nrf24-series. Acesso em 30 de outubro de 2021.

OLIVEIRA, Rodolpho C. **Sinal Analógico X Sinal Digital**. Embarcados.com: 2020. Disponível em: https://www.embarcados.com.br/sinal-analogico-x-sinal-digital/. Acesso em 29 de outubro de 2020.

PALHAIS, Catarina B. C. **Prototipagem: uma abordagem ao processo de criação de um produto**, Lisboa: ULFBA, 2015. Disponível em:

https://repositorio.ul.pt/bitstream/10451/29163/2/ULFBA_TES_942.pdf. Acesso em: 17 out. 2019.

PULUCENO, T. V. **Estudo de caso sobre uma api rest em node. js**. 2012.

PRUDENTE, Francesco. **Automação Predial e Residencial - Uma Introdução**. Rio de Janeiro: LTC, 2017.

RASPBERRY PI FOUNDATION. **Raspberry Pi Zero W**. Disponível em: https://www.raspberrypi.org/products/raspberry-pi-zero-w. Acesso em: 22 out. 2020.

RICHARDSON, Matt. WALLACE, Shawn. **Primeiros passos com o Raspberry pi.** 1ª ed. São Paulo: Novatec, 2013.

RICQUEBOURG, V. et al. **The Smart Home Concept**: our immediate future. IEEE. 2006. Disponível em: ieeexplore.ieee.org/document/4152762. Acesso em: 10 out. 2019.

ROBU. **DHT-11 Digital Temperature And Humidity Sensor.** Disponível em https://robu.in/product/dht-11-digital-temperature-humidity-sensor/. Acesso em 22 de agosto de 2021.

RODRIGUES, Kevin O. KALIL, Fahad. **Tecnologia e o Futuro: internet das coisas, microcontroladores e webservices**. IMED: 2013. Disponível em: https://www.imed.edu.br/uploads/90036a9a-cc09-4b96-8c2e-74912cd41d59.pdf. Acesso em 13 nov. 2019.

SACCOL, A. SCHLEMMER, E. BARBOSA, J. **M-learning e U-learning - novas perspectivas da aprendizagem móvel e ubíqua**. São Paulo: Pearson Prentice Hall; 2011.

STEVAN JR., Sergio Luiz. SILVA, Rodrigo Adamshuk. **Automação e instrumentação industrial com arduino : teoria e projetos**. São Paulo: Érica, 2015.296p.

SOUZA, David José. **Desbravando o PIC: Ampliado e Atualizado para PIC 16F628A**. 8ª ed. São Paulo, SP, Brasil: Érica, 2005.

TEZA, Vanderlei R. **ALGUNS ASPECTOS SOBRE A AUTOMAÇÃO RESIDENCIAL -DOMÓTICA.** UFSC**:** Florianópolis, 2002. [s.l.: s.n., s.d.]. Disponível em: https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/83015/212312.pdf?sequence=1&isAllowed=y. Acesso em: 4 Nov. 2020.

THOMSEN, Adilson.**Tutorial: Comunicação wireless com Arduino e Módulo NRF24L01**. FilipeFlop: 2020. Disponível em: https://www.filipeflop.com/blog/arduino-modulo-nrf24l01-tutorial/. Acesso em: 9 Nov. 2020.

THOMAZINI, D.; ALBUQUERQUE, P. U. B. **Sensores industriais: fundamentos e aplicações**. 7ª. ed. São Paulo: Érica, 2010.

TÓFOLI, Ricardo J**. CASA INTELIGENTE -SISTEMA DE AUTOMAÇÃO RESIDENCIAL**. [s.l.: s.n.], 2014. Disponível em: https://cepein.femanet.com.br/BDigital/arqTccs/1211320586.pdf. Acesso 3 nov. 2020.

USINAINFO. **Módulo sensor de corrente.** Disponível em: https://www.usinainfo.com.br/sensor-de-corrente-arduino/sensor-de-corrente-acs712-5a-ac-dc-com-efeito-hall-2550.html. Acesso em 22 de agosto de 2021.

WENDLING, Marcelo. **UNIVERSIDADE ESTADUAL PAULISTA**. [s.l.: s.n., s.d.]. Disponível em: https://www.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/4---sensores-v2.0.pdf. Acesso em 6 de novembro de 2020.

APÊNDICE A - Código de programação *Endpoints*

CÓDIGO DE PROGRAMAÇÃO *ENDPOINTS*

```
01 #include <string.h>
02 #include <stdlib.h>
03 #include <SPI.h>
04 #include "RF24.h"
05 #include <EEPROM.h>
06
07   const   char   str[][17]={   "ADDNEWDEVICE0000",   "GETMODELDEVICE00",
"GETDATAMODEL0000", "ADDR0x0000000000","REMOVEDEVICE0000",
08                                  "SETSTATUSACT000X", "GETALLSTATUS*DEV",
"GETDATAALLSTAT00"};
09
10 uint64_t addresses = 0x0000000000LL;
11
12 char receive_payload[32+1];
13 uint8_t reset = 0;
14 uint8_t streamCounter = 0;
15
16 //MODELO DO DISPOSITIVO
17 char model[17] = "MODELDEVICE?????";
18 char modelDevice[65];
19 char allStates[97];
20 bool data_act[5] = { true, true, true, true, false };
21 float data_sen[5] = { 0, 0, 0, 0, 0};
22 char data_sens[5][13];
23
24 //GPIOs PADRÕES PARA ATUADORES E SENSORES
25 const uint8_t act_pin[4] = {4, 5, 6, 7, 8};
26 const uint8_t sen_pin[4] = {A1, A2, A3, A4, A5};
27
28
29 //INICIALIZAÇÃO DO RADIO PINOS CE E CNS
30 RF24 radio(A0, 10);
31
32 void defineModelDevice(void);
33 void floatToString(void);
34 void defineAllStates(void);
35 uint64_t stringToBinary (String hexString);
36 char * strpcpy(char* inputString,int pos);
37 void EEPROM_sys(void);
```

```
38 unsigned int EEPROM_readint(int address);
39 unsigned long EEPROM_readlong(int address);
40 uint64_t EEPROM_readlonglong(int address);
41 void EEPROM_writeint(int address, int value);
42 void EEPROM_writelong(int address, unsigned long value);
43 void EEPROM_writelonglong(int address, uint64_t value);
44 void radioConfig(void);
45
46 void setup() {
47     /* CONFIGURE OS PINOS GPIOS DOS
48      *  ATUADORES E DOS SENSORES AQUI
49      */
50     EEPROM_sys();
51     Serial.begin(9600);
52     radio.begin();
53     radio.enableDynamicPayloads();
54     radio.setPALevel(RF24_PA_HIGH);
55     radioConfig();
56 }
57
58 void loop() {
59     while (radio.available()) {
60         radio.read(receive_payload, 32);
61         receive_payload[32] = 0;
62         radio.stopListening();
63         delay(20);
64         payloadCommand();
65         radio.write(receive_payload, 32);
66         radio.startListening();
67         if (reset == 1){
68            EEPROM_writelonglong(100,addresses);
69            EEPROM.write(0,127);
70            reset = 0;
71            radioConfig();
72         }
73         else if (reset == 2) {
74            addresses = EEPROM_readlonglong(20);
75            EEPROM.write(0,63);
76            reset = 0;
77            radioConfig();
78         }
```

```
79     }
80 }
81
82 void defineModelDevice (void) {
83   uint8_t numsen = 5;//NÚMERO DE SENSORES
84   uint8_t numact = 5;//NÚMERO DE ATUADORES
85   //UNIDADES DAS VÁRIAVEIS SENSORIADAS
86   char uni001[9] = "VarX1";
87   char uni002[9] = "VarX2";
88   char uni003[9] = "VarX3";
89   char uni004[9] = "VarX4";
90   char uni005[9] = "VarX5";
91  sprintf(modelDevice,"%s S%d A%d %s %s %s %s %s *", model, numsen, numact,
uni001, uni002, uni003, uni004, uni005);
92 }
93
94
95 void floatToString(void){
96   for(int i = 0; i<5; i++){
97     int sen_int, sen_dec, sig = 1;
98     float sen_frac;
99     if (data_sen[i] < 0) {
100       sig = -1;
101     }
102     sen_int = (int)(data_sen[i]*sig);
103     sen_frac = (data_sen[i]*sig) - sen_int;
104     sen_frac *= 100;
105     sen_dec = (int)sen_frac;
106     sprintf(data_sens[i],"%d.%d",(sen_int*sig),sen_dec);
107   }
108
109 }
110
111 void defineAllStates(void){
112     floatToString();
113     sprintf(allStates,"A1%s A2%s A3%s A4%s A5%s S1%s S2%s S3%s S4%s S5%s
*", data_act[0] ? "true" : "false", data_act[1] ? "true" : "false",
114       data_act[2] ? "true" : "false", data_act[3] ? "true" : "false",
data_act[4]            ?            "true"            :            "false",
data_sens[0],data_sens[1],data_sens[2],data_sens[3],data_sens[4]);
115 }
```

```
116
117
118
119 void radioConfig(void){
120     radio.openWritingPipe(addresses+1);
121     radio.openReadingPipe(1,addresses);
122     radio.startListening();
123 }
124
125 uint64_t stringToBinary (String hexString)
126 {
127     return stringToBinary (hexString.c_str ());
128 }
129
130 uint64_t stringToBinary (const char* hexString)
131 {
132     int length = min (16, strlen (hexString));
133     uint64_t returnValue = 0;
134     for (int i = 0; i < length && isxdigit (hexString[i]); ++i) {
135         char digit = toupper (hexString [i]);
136         returnValue *= 16;
137          returnValue += (digit >= '0' && digit <= '9') ? digit - '0' :
digit - 'A' + 10;
138     }
139     return returnValue/16;
140 }
141
142 char * strpcpy(char* inputString,int pos){
143     int i = 0, j = 0;
144     char * outputString =(char*) calloc((17-pos), sizeof(char));
145     for(i = pos; i < 17; i++){
146         outputString[j] = inputString[i];
147         j++;
148     }
149     return outputString;
150 }
151
152 void EEPROM_sys(void) {
153     unsigned int isReg = word(EEPROM.read(0));
154     if(isReg != 63 && isReg != 127){
155         //apenas para produção
```

```
156         EEPROM_writelonglong(20,0x0000000000LL);
157         EEPROM.write(0,63);
158     }
159     else if(isReg == 63) {
160         addresses = EEPROM_readlonglong(20);
161     }
162     else if(isReg == 127){
163         addresses = EEPROM_readlonglong(100);
164     }
165 }
166 unsigned int EEPROM_readint(int address)
167 {
168                 unsigned   int   word   =   word(EEPROM.read(address),
EEPROM.read(address+1));
169     return word;
170 }
171
172 unsigned long EEPROM_readlong(int address)
173 {
174     unsigned long dword = EEPROM_readint(address);
175     dword = dword << 16;
176     dword = dword | EEPROM_readint(address+2);
177     return dword;
178 }
179 uint64_t EEPROM_readlonglong(int address)
180 {
181     uint64_t qword = EEPROM_readlong(address);
182     qword = qword << 32;
183     qword = qword | EEPROM_readlong(address+4);
184     return qword;
185 }
186
187 void EEPROM_writeint(int address, int value)
188 {
189     EEPROM.write(address,highByte(value));
190     EEPROM.write(address+1 ,lowByte(value));
191 }
192
193 void EEPROM_writelong(int address, unsigned long value)
194 {
195     EEPROM_writeint(address+2, word(value));
```

```
196     value = value >> 16;
197     EEPROM_writeint(address, word(value));
198 }
199 void EEPROM_writelonglong(int address, uint64_t value)
200 {
201     EEPROM_writelong(address+4, long(value));
202     value = value >> 32;
203     EEPROM_writelong(address, long(value));
204 }
205
206 void payloadCommand(void)
207 {
208     if(strncmp (str[0],receive_payload,16)==0){
209         strcpy(receive_payload, model);
210     }
211     else if(strncmp (str[1],receive_payload,16)==0){
212         strcpy(receive_payload, "STREAM MODEL DEV");
213     }
214     else if(strncmp (str[2],receive_payload,14)==0){
215         defineModelDevice();
216         int i = receive_payload[15] - '0';
217           strncpy(receive_payload, modelDevice + ((i)*32), ((i+1)*32) -
(i*32));
218     }
219
220     else if(strncmp (str[3],receive_payload,4)==0){
221         char * receive_addr;
222         receive_addr = strpcpy(receive_payload,6);
223         addresses = stringToBinary(receive_addr);
224         free(receive_addr);
225         strcpy(receive_payload, "ADDRESSSETOKAY00");
226         reset = 1;
227     }
228     else if(strncmp (str[4],receive_payload,16)==0){
229         strcpy(receive_payload, "DEVICEISREMOVED0");
230         reset = 2;
231     }
232     else if(strncmp (str[5],receive_payload,13)==0){
233         int i = receive_payload[15] - '0' -1;
234         data_act[i] = not data_act[i];
235         byte state = data_act[i] ? HIGH : LOW;
```

```
236         digitalWrite(act_pin[i],state);
237         strcpy(receive_payload, "SETSTATUSACTOKAY");
238     }
239     else if(strncmp (str[6],receive_payload,16)==0){
240         /* FUNÇÕES DE LEITURA DOS SENSORES AQUI*/
241         strcpy(receive_payload, "STREAM STATE DEV");
242     }
243     else if(strncmp (str[7],receive_payload,14)==0){
244         int i = receive_payload[15] - '0';
245         defineAllStates();
246            strncpy(receive_payload, allStates + ((i)*32), ((i+1)*32) -
(i*32));
247     }
248
249 }
```

APÊNDICE B - Código de programação Camada 1 - Central

CÓDIGO DE PROGRAMAÇÃO CAMADA 1 - CENTRAL

<main.py>

```
01 import json
02 import time
03 import asyncio
04 import requests
05 import datetime
06 from database import DataBase
07 from radio import Radio
08
09
10 DB = DataBase()
11 RD = Radio()
12
13
14 async def message_verify():
15     a_message = DB.get_message('message')
16     if a_message[0]:
17         res = requests.get('http://127.0.0.1:3000/message')
18         return res.json()
19     else:
20         return False
21
22
23 def list_devices():
24     data = DB.list_devices()
25     newdata = []
26     for i in range(len(data)):
27         newdata.append(data[i][0])
28     return newdata, len(newdata)
29
30
31 def command(message):
32     print(datetime.datetime.now().replace(microsecond=0)
33             .isoformat()+" - Comando Recebido: "+message['cmd'])
34     if message['cmd'] == 'newdevice':
35         RD.add_device()
36     elif message['cmd'] == 'setstate':
37         RD.set_state(int(message['component']),message['id'])
38     elif message['cmd'] == 'remdevice':
```

```
39          RD.rem_device(message['id'])
40
41
42  async def main():
43      print(datetime.datetime.now().replace(microsecond=0)
44              .isoformat()+" - prothoHOUSE - Versão Final")
45      print(datetime.datetime.now().replace(microsecond=0)
46          .isoformat()+" - Inicializando Gerenciamento da Central")
47      print(datetime.datetime.now().replace(microsecond=0)
48          .isoformat()+" - Loop: Verifica os estados dos ENDPOINTS")
49      while 1:
50          devices, num_devices = list_devices()
51          for i in range(1, num_devices):
52              RD.get_status(devices[i])
53              message = await message_verify()
54              if  message:
55                  command(json.loads(message['message']))
56          time.sleep(2)
57
58
59  asyncio.run(main())
60
```

<radio.py>

```
01  import time
02  import RF24
03  from RF24 import *
04  from database import DataBase
05
06  class Radio:
07
08
09      def __init__(self):
10          self.irq_gpio_pin = None
11          self.millis = lambda: int(round(time.time() * 1000))
12          self.radio = RF24(22,0)
13          self.radio.begin()
14          self.radio.enableDynamicPayloads()
15          self.radio.enableAckPayload()
16          self.radio.setRetries(5, 15)
17          self.radio.setPALevel(RF24_PA_HIGH)
```

```
18          self.DB = DataBase()
19          self.info = ()
20          self.address0 = 0
21          self.component = 0
22          self.id = 'device000'
23
24
25      def open_radio(self,address):
26          self.radio.openWritingPipe(address)
27          self.radio.openReadingPipe(1, address+1)
28
29
30      def select_radio(self,device):
31          self.id = device
32          self.address0 = self.DB.get_address(self.id)
33          self.open_radio(self.address0)
34
35
36      def flush_radio(self):
37          self.radio.openWritingPipe(0x0000000001)
38          self.radio.openReadingPipe(1,0x0000000002)
39
40
41      def radio_message(self,message,priority):
42          self.radio.stopListening()
43          self.radio.write(message)
44          time.sleep(0.002)
45          self.radio.startListening()
46          started_waiting_at = self.millis()
47          timeout = False
48          while (not self.radio.available()) and (not timeout):
49              if (self.millis() - started_waiting_at) > priority*150:
50                  timeout = True
51          if timeout:
52              return False, "Timeout"
53          else:
54              data = self.radio.read(32)
55              return True, data.decode("utf-8")
56
57      ## VALIDAÇÃO DE DADOS RECEBIDOS
58      def validate_data(self,message,data):
```

```
59          message = message.decode("utf-8")
60          data = data[0:16]
61          if message == 'ADDNEWDEVICE0000':
62              self.info = self.DB.get_devices_infos(data)
63              return True if self.info is not None else False
64          elif message[0:5] == 'ADDR0':
65              if data == 'ADDRESSSETOKAY00':
66                  return True
67              else:
68                  return False
69          elif message == 'GETMODELDEVICE00':
70              if data == 'STREAM MODEL DEV':
71                  success, data = self.data_stream(b'GETDATAMODEL000X',2)
72                  if success:
73                      data = data.split()
74                      data.pop()
75                      data[1] = data[1][1]
76                      data[2] = data[2][1]
77                      data = tuple(data)
78                      return self.DB.set_devices_infos(data)
79                  else:
80                      return False
81          elif message == 'REMOVEDEVICE0000':
82              if data == 'DEVICEISREMOVED0':
83                  self.DB.rem_device(self.id,'')
84                  return True
85              else:
86                  return False
87          elif message[0:12] == 'SETSTATUSACT':
88              if data == 'SETSTATUSACTOKAY':
89                  return True
90              else:
91                  return False
92          elif message == 'GETALLSTATUS*DEV':
93              if data == 'STREAM STATE DEV':
94                  success, data = self.data_stream(b'GETDATAALLSTAT0X',3)
95                  if success:
96                      data = data.split()
97                      data.pop()
98                      for i in range(10):
99                          data[i] = data[i][2:]
```

```
100                 data.append(self.id)
101                 data = tuple(data)
102                 return self.DB.update_status(data)
103             else:
104                 return False
105         else:
106             return  False
107 
108 
109     def data_stream(self, message, size):
110         priority = 2
111         priority_limit = priority * 2
112         big_data = ''
113         data = ''
114         for part in range (size):
115             message1 = message.replace(b'X',bytes(repr(part).encode()))
116             state, data = self.radio_message(message1,priority)
117             while not state and priority <= priority_limit:
118                 priority += 1
119                 state, data = self.radio_message(message1,priority)
120             if not state:
121                 return False, 'Falha ao obter dados'
122             else:
123                 big_data = big_data + data
124         return True, big_data
125 
126 
127     def send_message(self,message,priority):
128         priority_limit = priority * 2
129         state, data = self.radio_message(message,priority)
130         while not state and priority <= priority_limit:
131             priority += 1
132             state, data = self.radio_message(message,priority)
133         if not state:
134             return False
135         else:
136             return self.validate_data(message,data)
137 
138 
139     def get_status(self, device):
140         self.id = device
```

```
141         self.select_radio(self.id)
142         return self.send_message(b'GETALLSTATUS*DEV', 2)
143 
144 
145     def set_state(self, component, device):
146         self.id = device
147         self.select_radio(self.id)
148         self.component = component
149         message = b'SETSTATUSACT000X'
150         message =
message.replace(b'X',bytes(repr(self.component).encode()))
151         return self.send_message(message, 2)
152 
153 
154     def send_address(self,address):
155         address = f'ADDR{hex(address)}'
156         address = address.encode()
157         address = bytearray(address)
158         return self.send_message(address,2)
159 
160 
161     def add_device(self):
162         self.open_radio(1)
163         success = self.send_message(b'ADDNEWDEVICE0000', 5)
164         if success:
165             self.address0 = self.DB.add_device(self.info)
166             success = self.send_address(self.address0)
167             if success:
168                 return True
169             else:
170                 self.DB.rem_device(self.address0, 'addr')
171                 return False
172         else:
173             success = self.send_message(b'GETMODELDEVICE00',2)
174             if success:
175                 return self.add_device()
176             else:
177                 return False
178 
179 
180     def rem_device(self, device):
```

```
181         self.id = device
182         address = self.DB.get_address(self.id)
183         self.open_radio(address)
184         sucess = self.send_message(b'REMOVEDEVICE0000',5)
185         if sucess:
186             return True
187         else:
188             return False
189
190
191
192     def print(self):
193         self.radio.printDetails()
194
```

<database.py>

```
01 import psycopg2
02 import os
03 from dotenv import load_dotenv
04
05
06 class DataBase:
07
08
09     def __init__(self):
10         load_dotenv()
11
12
13     def connect_db(self):
14         dbname = os.environ.get("DB_DATABASE")
15         user = os.environ.get("DB_USERNAME")
16         host = os.environ.get("DB_HOSTNAME")
17         password = os.environ.get("DB_PASSWORD")
18         return psycopg2.connect("dbname='" + dbname +"' user='"+ user +
"' host='" + host + "' password='" + password + "'")
19
20
21     def list_devices(self):
22         try:
23             conn = self.connect_db()
24         except psycopg2.Error as e:
25             print(e.pgerror)
```

```
26              return False
27          finally:
28              if conn:
29                  cur = conn.cursor()
30                  cur.execute('SELECT id FROM devices ORDER BY id;')
31                  data = cur.fetchall()
32                  return data
33              else:
34                  return False
35
36
37      def get_address(self,data):
38          try:
39              conn = self.connect_db()
40          except psycopg2.Error as e:
41              print(e.pgerror)
42              return False
43          finally:
44              if conn:
45                  cur = conn.cursor()
46                  cur.execute('SELECT addresses FROM addresses WHERE id =
%s;',(data,))
47                  address = cur.fetchone()
48                  if address is not None:
49                      address = address[0]
50                      return address
51                  else:
52                      return None
53              else:
54                  return False
55
56
57      def add_device(self,data):
58          try:
59              conn = self.connect_db()
60          except psycopg2.Error as e:
61              print(e.pgerror)
62              return False
63          finally:
64              if conn:
65                  cur = conn.cursor()
```

```
66                 cur.execute('SELECT addresses FROM addresses WHERE
active = false ORDER BY id;')
67                 address = cur.fetchall()
68                 device = None
69                 if address != []:
70                     address = address[0]
71                     cur.execute('SELECT id FROM addresses WHERE
addresses = %s;',(address,))
72                     device = cur.fetchone()
73                     cur.execute("UPDATE addresses SET active = true
WHERE addresses = %s;",(address,))
74                     address = address[0]
75                 else:
76                     cur.execute('SELECT addresses FROM addresses WHERE
active = true ORDER BY id;')
77                     address = cur.fetchall()[::-1][0][0] + 16
78                     cur.execute('SELECT id FROM addresses WHERE active =
true ORDER BY id;')
79                     value = cur.fetchall()[::-1][0][0]
80                     value = str(int(value[6:9]) + 1)
81                     device = 'device' + value.zfill(3)
82                     cur.execute("INSERT INTO addresses VALUES ( %s,%s,
true);",(device,address,))
83                 data = ((device,) + data,)
84                 cur.execute("""INSERT INTO devices VALUES ( %s, %s, %s,
%s, 0, 0, 0, 0, 0,
85                             false, false, false, false, false,%s, %s,
%s, %s, %s);""", data[0])
86                 conn.commit()
87                 conn.close()
88                 return address
89             else:
90                 return False
91
92
93     def rem_device(self,data,addr):
94         try:
95             conn = self.connect_db()
96         except psycopg2.Error as e:
97             print(e.pgerror)
98             return False
```

```
99          finally:
100             if conn:
101                 cur = conn.cursor()
102                 if addr != '':
103                     cur.execute('SELECT id FROM addresses WHERE
addresses = %s;',(data,))
104                     data = cur.fetchone()
105                     if data is None:
106                         return False
107                     else:
108                         data = data[0]
109                 if data != 'device000':
110                     cur.execute("DELETE FROM devices WHERE id=
%s;",(data,))
111                     cur.execute("UPDATE addresses SET active = false
WHERE id = %s;",(data,))
112                     conn.commit()
113                     conn.close()
114                     return True
115                 else:
116                     return False
117             else:
118                 return False
119
120
121     def update_status(self,data):
122         try:
123             conn = self.connect_db()
124         except psycopg2.Error as e:
125             print(e.pgerror)
126             return False
127         finally:
128             if conn:
129                 cur = conn.cursor()
130                 cur.execute("""UPDATE devices SET act001 = %s, act002 =
%s, act003 = %s,
131                     act004 = %s, act005 = %s, sen001 = %s, sen002 = %s,
sen003 = %s,
132                     sen004 = %s, sen005 = %s WHERE id = %s;""",data)
133                 conn.commit()
134                 conn.close()
```

```
135                 return True
136             else:
137                 return False
138 
139 
140     def get_devices_infos(self, data):
141         try:
142             conn = self.connect_db()
143         except psycopg2.Error as e:
144             print(e.pgerror)
145             return False
146         finally:
147             if conn:
148                 cur = conn.cursor()
149                 cur.execute('SELECT * FROM models WHERE id =
%s',(data,))
150                 data = cur.fetchone()
151                 if data is None:
152                     return None
153                 else:
154                     return data
155 
156 
157     def set_devices_infos(self, data):
158         try:
159             conn = self.connect_db()
160         except psycopg2.Error as e:
161             print(e.pgerror)
162             return False
163         finally:
164             if conn:
165                 cur = conn.cursor()
166                 cur.execute('INSERT INTO models VALUES (%s, %s, %s, %s,
%s, %s, %s ,%s);',(data))
167                 conn.commit()
168                 conn.close()
169                 return True
170 
171 
172     def get_message(self, operation):
173         try:
```

```
174             conn = self.connect_db()
175         except psycopg2.Error as e:
176             print(e.pgerror)
177             return False
178         finally:
179             if conn:
180                 cur = conn.cursor()
181                 cur.execute('SELECT state FROM messages WHERE id =
%s;',(operation,))
182                 data = cur.fetchone()
183                 return data
184             else:
185                 return False
186
```

APÊNDICE C - Código de programação Camada 2 - Central

CÓDIGO DE PROGRAMAÇÃO CAMADA 2 - CENTRAL

<server.js>

```
01 require('dotenv').config();
02 const express = require('express');
03 const app = express();
04 const consign = require('consign');
05 const db = require('./config/db');
06 var mqtt = require('mqtt')
07
08 app.use(express.json());
09 app.use(express.urlencoded({ extended: true}));
10
11 consign()
12     .then('api')
13     .then('config/routes.js')
14     .into(app);
15
16 app.db = db;
17
18 const port = process.env.PORT || 3000;
19
20 app.listen(port, () => {
21     console.log(`${now()} - prothoHouse - Versão Final`)
22     console.log(`${now()} - Inicializando o Gerenciador de Mensagens da
Central`)
23
24 });
25
26 var options = require('./config/mqtt')
27
28 now = () => {
29     let data = new Date();
30     let data2 = new Date(data.valueOf() - data.getTimezoneOffset() *
60000);
31     var dataBase = data2.toISOString().replace(/\.\d{3}Z$/, '');
32     return dataBase
33 }
34
35
36 var client = mqtt.connect(options);
```

```
37
38 client.on('connect', function () {
39     console.log(`${now()} - Conectado ao Servidor MQTT`);
40     console.log(`${now()} - Mensagens MQTT sendo enviadas em
PH/PH001/data`)
41 });
42
43 client.on('error', function (error) {
44     console.log(error);
45 });
46
47 client.on('message', function (topic, message) {
48     var msg = JSON.parse(message.toString());
49     console.log(`${now()} - Comandos MQTT sendo recibidos em ${topic}:
${msg['cmd']}`)
50     app.db('messages')
51     .update({
52         state: true})
53     .where({id: 'message'})
54     .then(result => console.log(result))
55     .catch(err => console.log(err));
56     app.message = message.toString();
57 });
58
59 const runtime = () => {
60     app.db('devices')
61     .select('*')
62     .orderBy('id')
63     .then(devices => {
64             var msg = JSON.stringify(devices);
65             client.publish("PH/PH001/data", msg);})
66     .catch(err => console.log(err));
67 }
68
69 setInterval(runtime, 5000);
70
71 client.subscribe("PH/PH001/cmd");
72
73 module.exports = app;
```

<./api/commands.js>

```
01 module.exports = app =>{
02     const message = (req,res) => {
03         app.db('messages')
04         .update({
05             state: false})
06         .where({id: 'message'})
07         .then(/*result => console.log(result)*/)
08         .catch(err => console.log(err));
09         res.json({ message: app.message})
10     }
11 
12     return { message };
13 
14 };
```

<./config/db.js>

```
01 const knex = require('knex')({
02     client: 'pg',
03     version: '12.5',
04     connection: {
05       host : process.env.DB_HOSTNAME,
06       user : process.env.DB_USERNAME,
07       password : process.env.DB_PASSWORD,
08       port: process.env.DB_PORT,
09       database : process.env.DB_DATABASE
10     }
11   });
12 module.exports = knex
```

<./config/mqtt.js>

```
01 const mqtt = {
02     host: process.env.MQTT_HOSTNAME,
03     port: process.env.MQTT_PORT,
04     protocol: process.env.MQTT_PROTOCOL,
05     username: process.env.MQTT_USERNAME,
06     password: process.env.MQTT_PASSWORD
07 }
08 
09 module.exports = mqtt
```

<./config/routes.js>

```
01 module.exports = app => {
02     app.get('/message', app.api.commands.message)
03 }
```

APÊNDICE D - Código de programação do Microsserviço

CÓDIGO DE PROGRAMAÇÃO DO MICROSERVIÇO

<server.js>

```
01 require('dotenv').config();
02 const express = require('express');
03 const app = express();
04 const consign = require('consign');
05 const db = require('./config/db');
06
07 app.use(express.json());
08 app.use(express.urlencoded({ extended: true}));
09
10 consign()
11     .then('api')
12     .then('config/middlewares.js')
13     .then('config/routes.js')
14     .into(app);
15
16 app.db = db;
17
18 now = () => {
19     let data = new Date();
20     let data2 = new Date(data.valueOf() - data.getTimezoneOffset() *
60000);
21     var dataBase = data2.toISOString().replace(/\.\d{3}Z$/, '');
22     return dataBase
23 }
24
25 const port = process.env.PORT || 3000;
26 app.listen(port, () => {
27     console.log(`\n${now()} - protoHouse - Versão Final`);
28 });
29
30 app.get('/', function (req, res) {
31     res.send('Hello World');
32  })
33
34 module.exports = app;
```

<./api/commands.js>

```
01 module.exports = app =>{
```

```
02      const getstate = (req,res) => {
03          console.log(`${now()} - Requisição Recebida: /getstate`);
04          app.db('devices')
05              .select('*')
06              .orderBy('id')
07              .then(devices => res.json(devices))
08              .catch(err => res.status(400).json(err));
09      };
10      const setstate = (req,res) => {
11          function pad(num, size) {
12              var s = "0000" + req.body.id;
13              return s.substr(s.length-size);
14          }
15          const id = 'device' + pad(req.body.id,3);
16          const component =  (req.body.component).split('t')[1];
17          console.log(`${now()} - Requisição Recebida: /setstate =>
req.id=${id} req.component=${req.body.component}`);
18          const message = "{\"id\" : \""+`${id}`+"\",\"component\" :
\""+`${component}`+"\", \"cmd\": \"setstate\"}";
19          app.client.publish("PH/PH001/cmd", message);
20          return res.status(200).send('Okay');
21
22      };
23      const newdevice = (req, res) => {
24          console.log(`${now()} - Requisição Recebida: /newdevice`);
25          const message = "{\"id\" : \"device000\",\"component\" :
\"act000\", \"cmd\": \"newdevice\"}";
26          app.client.publish("PH/PH001/cmd", message);
27          return res.status(200).send('Okay');
28      };
29
30      const remdevice = (req, res) => {
31          console.log(`${now()} - Requisição Recebida: /remdevice`);
32          const message = "{\"id\" : \"device000\",\"component\" :
\"act000\", \"cmd\": \"remdevice\"}";
33          app.client.publish("PH/PH001/cmd", message);
34          return res.status(200).send('Okay');
35      };
36      return { getstate, setstate, newdevice, remdevice };
37
38  };
```

<./api/mqtt.js>

```
01 module.exports = app =>{
02     const mqtt = require('mqtt');
03     const options = require('../config/mqtt');
04     const client = mqtt.connect(options);
05     app.client = client;
06     client.on('message', function (topic, message) {
07         var msg = JSON.parse(message.toString());
08         app.db('devices')
09             .insert(msg)
10             .onConflict('id').merge()
11             .then(/*result => */)
12             .catch(err => console.log(err));
13     });
14
15     client.subscribe('PH/PH001/data');
16     client.on('connect', function () {
17         console.log(`${now()} - Conectado ao Servidor MQTT:
4569b803aeea47a086ceb3551110c7ce.s1.eu.hivemq.cloud`);
18         console.log(`${now()} - Mensagens MQTT sendo recebidas em
PH/PH001/data. Atualizando Banco de Dados Periodicamente`)
19     });
20     client.on('error', function (error) {
21         console.log(error);
22     });
23
24 }
```

<./config/db.js>

```
01 const knex = require('knex')({
02     client: 'pg',
03     version: '12.5',
04     connection: {
05       host : process.env.RDS_HOSTNAME,
06       user : process.env.RDS_USERNAME,
07       password : process.env.RDS_PASSWORD,
08       port: process.env.RDS_PORT,
09       database : process.env.RDS_DATABASE
10     }
```

```
  });
module.exports = knex
```

<./config/middlewares.js>

```
const cors = require('cors')

module.exports = app => {
    app.use(cors({
        origin: '*'
    }))
}
```

<./config/mqtt.js>

```
const mqtt = {
    host: process.env.MQTT_HOSTNAME,
    port: process.env.MQTT_PORT,
    protocol: process.env.MQTT_PROTOCOL,
    username: process.env.MQTT_USERNAME,
    password: process.env.MQTT_PASSWORD
}

module.exports = mqtt
```

<./config/routes.js>

```
module.exports = app => {
    app.post('/newdevice', app.api.commands.newdevice)
    app.post('/remdevice', app.api.commands.newdevice)
    app.post('/setstate', app.api.commands.setstate)
    app.get('/getstate', app.api.commands.getstate)
}
```