

CENTRO UNIVERSITÁRIO INTERNACIONAL UNINTER
BACHARELADO EM ENGENHARIA ELÉTRICA – HABILITAÇÃO
ELETRÔNICA

VINÍCIUS PICCHI CAETANO

AUTOMAÇÃO RESIDENCIAL UTILIZANDO RASPBERRY PI 3 B

Curitiba

2021

VINÍCIUS PICCHI CAETANO

AUTOMAÇÃO RESIDENCIAL UTILIZANDO RASPBERRY PI 3 B

Trabalho de conclusão de curso
apresentado ao Centro Universitário
Internacional Uninter, como parte dos
requisitos para obtenção do título de Bacharel
em Engenharia Elétrica.

Orientador: Prof. Dr. Felipe de Aguiar Neves

Curitiba

2021

Agradecimentos

Meu primeiro agradecimento, vai para o meu Senhor Deus, pois sem toda a sua ajuda, eu não teria concluído com êxito minha graduação.

Agradeço a todos os professores do Centro Universitário Internacional Uninter, em especial, do Centro Politécnico, que me capacitaram com todo o conhecimento necessário para concluir minha graduação, todos, foram peças fundamentais para a conclusão desta importante etapa da minha vida.

Sou grato ao meu orientador, professor Felipe, que me deu suporte em todas as etapas deste projeto, me instruindo e ajudando a desenvolver o mesmo.

Agradeço a minha família, em especial meus pais, por serem minha base em todos os sentidos e por terem sido essenciais ao longo da minha graduação.

Agradeço a todos os meus amigos e colegas, onde, sempre que precisei, me ajudaram de alguma forma para concluir essa trajetória.

Agradeço a todos, que de alguma maneira, me ajudaram a concluir esta importante etapa de minha vida.

RESUMO

De acordo com dados divulgados pela Associação Brasileira de Automação Residencial (AURESIDE) estudos mostraram que, o mercado de automação residencial teve uma projeção de crescimento anual de 11,35% entre 2014 e 2020, globalmente. Tal crescimento, se dá ao fato de que as pessoas ao redor do mundo têm percebido um grande crescimento quanto ao consumo de energia, que consequentemente traz um custo financeiro ainda maior. A vontade de ter uma residência automatizada com o intuito de trazer comodidade, segurança e economia, fez com que pessoas no mundo todo tenham interesse em deixar suas casas mais inteligentes. Partindo deste pressuposto, questiona-se: como aplicar uma automação dentro de uma residência com o objetivo de trazer segurança, comodidade e economia? Sabe-se que para a implementação de qualquer processo de automação, um usuário sem conhecimento técnico prévio, pode fazer isso, desde que, siga os tutoriais de instalação junto ao fabricante, que normalmente vem com os produtos adquiridos. Então, esta monografia, tem a finalidade de aplicar uma automação residencial no qual o usuário terá o controle de iluminação de uma residência em qualquer lugar do mundo, irá obter dados como: intensidade da luminosidade do ambiente e umidade e temperatura, para que o usuário possa controlar ações que possam agregar em sua vida. Além da implementação, a presente pesquisa visa ajudar a qualquer pessoa que, tenha interesse em automatizar sua residência.

Palavras-chave: Automação; Casas Inteligentes; Internet das coisas; Interface Humano Computador

ABSTRACT

According to data released by the Brazilian Association of Home Automation (AURESIDE) advanced studies that, the home automation market had an annual growth projection of 11.35% between 2014 and 2020, globally. Such growth is due to the fact that people around the world have noticed a great increase in energy consumption, which consequently brings an even greater financial cost. The desire to have an automated residence in order to bring safety, security and savings, has made people around the world interested in making their homes smarter. Assuming this, the following question arises: how to apply automation within a home with the objective of bringing security, convenience and savings? It is known that for the implementation of any automation process, the user must have a slight technical knowledge for this. So, this monograph, has a standard of applying a home automation without which the user will have the lighting control of a residence anywhere in the world, will obtain data such as: intensity of the ambient light and humidity and temperature, so that the user you can control the actions you can add in your life. In addition to the implementation, this research aims to help anyone who is interested in automating their residence.

Keyword: Automation; Smart Homes; Internet of things; Human Computer Interface

SUMÁRIO

1.	INTRODUÇÃO	9
2.	FUNDAMENTAÇÃO TEÓRICA	12
2.1.	Interação Humano-Computador	12
2.2.	Sistemas Embarcados	13
2.3.	Microcontroladores	13
2.4.	Organização de um software computacional de modo geral.....	15
2.5.	Interconexão de um sistema e estrutura da unidade de processamento	16
2.6.	RASPBERRY PI 3	18
2.7.	Sensores	21
2.7.1.	Sensor de umidade e temperatura DHT11	22
2.7.2.	Acionando o Circuito Interno.....	30
2.7.3.	Módulo Relé utilizado no projeto	31
2.8.	PROTOCOLO MQTT.....	32
2.8.1.	PORQUE ESCOLHER O MQTT?	32
2.8.2.	O modelo de publicação e assinatura	33
2.9.	Adafruit IO	35
2.9.1.	Adafruit IO, como funciona?	35
2.10.	Linguagem de programação Python.....	37
2.10.1.	Quais são os tipos de aplicações que podem utilizar Python?	38
2.11.	Tecnologias Semelhantes no Mercado Atual.....	39
3.	METODOLOGIA	44
3.1.	Requisitos da maquete e <i>Hardware</i>	46
3.2.	Desenvolvimento do <i>Software</i>	53
3.2.1.	Premissas de Funcionalidade	53
3.2.2.	Premissas não funcionais	55
3.2.3.	Caso de Uso	55
3.2.4.	Interface WEB	56
3.2.5.	Utilização do sensor DHT11.....	60
3.2.6.	Utilização do sensor LDR	61
3.2.7.	Utilização do sensor PIR	63
3.2.8.	Controle de Iluminação.....	64
4.	RESULTADOS E DISCUSSÕES	66
5.1.1.	Dificuldades encontradas	72
5.1.2.	Melhorias futuras	73

LISTA DE FIGURAS

Figura 1 - Ciclos de instrução	14
Figura 2 - Principais componentes de um sistema computacional.	Erro! Indicador não definido.
Figura 3 - Célula de Memória.....	16
Figura 4 - Arquitetura geral de um sistema.	Erro! Indicador não definido.
Figura 5 - Raspberry pi 3.	Erro! Indicador não definido.
Figura 6 - SoC BCM2837.....	Erro! Indicador não definido.
Figura 7 - Processos internos DHT11.	Erro! Indicador não definido.
Figura 8 - Sensor DHT11.....	Erro! Indicador não definido.
Figura 9 - Sensor DHT11.....	25
Figura 10 – LDR e seus aspectos mais comuns.	Erro! Indicador não definido.
Figura 11 - Fotorresistor (LDR)	Erro! Indicador não definido.
Figura 12 - Relé Telefônico em 1939.....	Erro! Indicador não definido.
Figura 13 - Ilustração dos cálculos binários de um computador na década de 1940.....	Erro! Indicador não definido.
Figura 14 - Módulo Relé e suas conexões.	Erro! Indicador não definido.
Figura 15 - Circuito de acionamento do relé de maneira geral.	Erro! Indicador não definido.
Figura 16 - Módulo relé de 4 canais, com circuito de acionamento integrado a placa.	Erro! Indicador não definido.
Figura 17 - Módulo relé eletromagnético.....	31
Figura 18 - Modelo de publicação e assinatura do MQTT para sensores de IoT.....	Erro! Indicador não definido.
Figura 19 - Criando uma instância de serviço da plataforma Internet das Coisas da IBM Cloud.	Erro! Indicador não definido.
Figura 20 - Exemplo de código Python.	Erro! Indicador não definido.
Figura 21 - Sonoff básico.	Erro! Indicador não definido.
Figura 22 – Aplicativo eWeLink Sonoff	Erro! Indicador não definido.
Figura 23 - Kit Smart Home Xiaomi.....	Erro! Indicador não definido.
Figura 24 - Diagrama de blocos - Controle de Iluminação	44
Figura 25 - Diagrama de Blocos - Sensores de Dados	44
Figura 26 - Diagrama de Blocos - Sensor PIR	45
Figura 27 - Diagrama Unifilar - Maquete Residencial	47
Figura 28 – Esquemático do circuito do módulo Relé	50
Figura 29 - Esquemático do circuito de GPIO do Raspberry	50
Figura 30 - Esquemático de conexão do circuito do projeto	51
Figura 31 - Conexão módulo Relé e planta elétrica.	52
Figura 32 - Diagrama de caso de uso do software	56
Figura 33 - Interface WEB do projeto – Lâmpadas desligadas.....	57
Figura 34 - Interface WEB do projeto – Lâmpadas ligadas	58
Figura 35 - Feeds utilizados no projeto	58
Figura 36 - Tela de configuração dos Feeds.....	59
Figura 37 - Fluxograma de funcionamento do algoritmo do DHT11	60
Figura 38 - Banco de Dados DHT11 - SQLITE	61
Figura 39 - Banco de Dados LDR - SQLITE	62
Figura 40 - Fluxograma de funcionamento de algoritmo do LDR	62
Figura 41 - Fluxograma de funcionamento do algoritmo do PIR	63
Figura 42 - Fluxograma de funcionamento do algoritmo do controle de iluminação	64
Figura 43 - Interface WEB final do projeto	66
Figura 44 - Maquete do projeto finalizada	67
Figura 45 - Central para controle de iluminação do projeto	67
Figura 46 - Circuito de sensores implementados do projeto.....	68
Figura 47 - Gráfico de valores coletados do DHT11	69

Figura 48 - Dados coletados do DHT11 ao longo de um dia	70
Figura 49 - Gráfico de luminosidade do LDR	71

LISTA DE QUADROS

Quadro 1 - Dados técnicos Raspberry Pi 2 e Pi3 Modelo B.	21
Quadro 2 - Especificações técnicas Sensor de Umidade e Temperatura DHT11	22
Quadro 3 - Especificações técnicas de Alimentação do DHT11	25
Quadro 4 - Especificações técnicas DHT22.....	26
Quadro 5 - Especificações técnicas Sonoff	40
Quadro 6 - Especificações técnicas Mibo Home Intelbras	41
Quadro 7 - Especificações técnicas Smart Home Xiaomi	43
Quadro 8 - Lista de Materiais Maquete Residencial	46
Quadro 9 - Demanda de Iluminação.	48
Quadro 10 - Quadro de cargas da iluminação.....	48
Quadro 11 - Relação dos terminais de GPIO e módulo Relé.	49
Quadro 12 - Relação de conexão dos sensores e Raspberry.	53
Quadro 13 - Envio dos comandos ao Relé.....	53
Quadro 14 - Ligando e Desligando as Lâmpadas	54
Quadro 15 - Interação do usuário	54
Quadro 16 - Interação do usuário com o Sensor DHT11	54
Quadro 17 - Interação do usuário com o Sensor LDR.....	54
Quadro 18 - Interação do usuário com o Sensor PIR.....	55
Quadro 19 - Biblioteca de configuração das GPIOs	55
Quadro 20 - Biblioteca de configuração Python 3.....	55
Quadro 21 - Link de atualização dos repositórios mais recentes do Adafruit	55
Quadro 22 - Biblioteca para implementar o sensor DHT11	55
Quadro 23 - Quadro de informações técnicas do sistema de sensores	68

1. INTRODUÇÃO

O ser humano em geral, sempre buscou melhorar seu estilo de vida, com consideráveis melhorias. A criação da roda, a descoberta do fogo, dentre outras invenções, sempre tivera o objetivo de facilitar o trabalho manual, deixando o mesmo fácil e prático. Com surgimento da automação, o homem conseguiu desenvolver técnicas e dispositivos que impulsionaram seus processos se tratando de qualidade e capacidade, evoluindo a qualidade de suas produções (LIMA, 2003).

Entretanto, é importante ressaltar que, ao longo do processo evolutivo da automação residencial, outras tecnologias foram criadas de maneira contundente que se relacionaram de um jeito um tanto quanto efetivo, desta forma, em conjunto, estas tecnologias tiveram uma evolução em sua história, tecnologias como: telefonia, rádio e televisão, videogames, computadores, iluminação dentre outras (PEREIRA, 2007).

A automação residencial tem como fundamento trazer inovação e desenvolvimento tecnológico em suas aplicações. Além disso, os usuários deste segmento prezam pela comodidade e conforto que também pode ser chamado de domótica (OLHAR DIGITAL, 2020).

Domótica é a tradução da palavra "Domotique", originada do francês, da qual a junção Domus derivada de casa e Imotique significa automático, ou seja, o controle automático de algo em casa. Este termo é utilizado no prisma da robótica, na qual utiliza esses mecanismos e tecnologias com a finalidade de trazer praticidade para as pessoas, simplificando afazeres manuais e humanizados. Desta maneira, sua utilização faz com que o usuário possa ter controle e segurança, com maior abrangência e minuciosidade, satisfazendo tais necessidades através de meios comunicativos avançados (BOLZANI, 2004).

1.1 OBJETIVOS

Neste capítulo, serão abordados os objetivos geral e específico do trabalho como num todo.

1.1.1 Objetivo Geral

Implementar o controle de processos residenciais. Desenvolver aplicações para aquisição de informações como: umidade, temperatura e presença de pessoas no local, além de controlar o sistema de iluminação. Realizar o controle e visualizar os dados obtidos através de uma página WEB, através de um smartphone, computador, etc. Implementar os controles e sensores de aquisição utilizando um computador de placa única, chamado Raspberry pi 3.

1.1.2 Objetivos Específicos

- Verificar e controlar os dispositivos atuadores dentro de uma residência;
- Realizar a programação para comunicação entre a página na WEB e o Raspberry;
- Implementar um sistema automatizado para a iluminação interna de uma residência, representada por uma maquete;
- Monitorar a umidade e temperatura com o sensor para obtenção de dados referentes a temperatura ambiente da residência;
- Verificar a presença de pessoas dentro da residência com o sensor de presença, para que o software apresente movimentações internas na casa.
- Acompanhar a intensidade da iluminação no local com o sensor de luminosidade, para que o programa indique se está claro, escuro ou sombreado.

1.1.3 JUSTIFICATIVA

De acordo com (AURESIDE, 2016), a Associação Brasileira de Automação Residencial e Predial, em um estudo levantado pela empresa *Markets and Markets*, o crescimento da automação residencial e seu mercado, é dado por vários fatores. Dentre eles estão:

- A grande evolução do mercado de *Internet Of Things (IOT)*;
- A redução de custos e desperdícios que esta tecnologia traz ao usuário;
- O fato de inúmeros fabricantes estarem expandindo seu leque de produtos;
- E do importante crescimento quanto ao monitoramento remoto de residências.

Mundialmente, o mercado de automação residencial está em crescimento em um valor de U\$ 32 bilhões desde 2015 para U\$ 78 bilhões até 2022, tendo uma taxa crescente de 12,5% (AURESIDE, 2016).

O comércio voltado ao controle de iluminação em implementações de automação doméstica, deve crescer exponencialmente durante o período de previsão citado anteriormente, levando em conta que, promove uma considerável redução no consumo de energia nas residências, utilizando, por exemplo, sensores de luminosidade com o objetivo de ajustar a luz artificial conforme a intensidade da luz natural.

Sistemas de iluminação, estão em grupo de custos que, consomem mais energia elétrica em uma residência, em todo o mundo. O consumo de energia elétrica em residências, é um dos maiores motivos pelo qual o crescimento da automação residencial vem sendo exponencial. Por outro lado, com o passar dos anos, as progressivas preocupações ambientais têm sido de suma importância para espalhar a consciência sobre o quanto é importante a utilização de medidas de automação doméstica e do gerenciamento de energia em seu lar (AURESIDE, 2016).

As principais empresas que, participaram deste importante estudo de mercado, ao redor do mundo, voltado a automação residencial foram: *Honeywell International Inc.* (EUA), *Legrand* (França), *Ingersoll-Rand plc.* (Irlanda), *Johnson Controls Inc.* (EUA), *Schneider Electric SE* (França), *Siemens AG* (Alemanha), *ABB Ltd.* (Suíça), *Acuity Brands, Inc.* (EUA), *United Technologies Corporation* (EUA), *Samsung Electronics Co., Ltd.* (Coréia do Sul), *Nest Labs, Inc.* (EUA) e *Crestron Electronics, Inc.* (EUA) (AURESIDE, 2016).

De acordo com os estudos levantados, a implementação deste projeto trará benefícios econômicos e segurança ao usuário. Sobretudo, visando sempre a comodidade e conforto quanto a utilização e controle de seus processos residenciais que normalmente são em sua maioria manuais.

2. FUNDAMENTAÇÃO TEÓRICA

Esse capítulo traz a fundamentação sobre os principais assuntos que relacionam o mundo da automação, os principais elementos práticos e teóricos para a projeção e implementação deste processo.

2.1. Interação Humano-Computador

Para os desenvolvedores de *softwares* existem duas características relevantes para a sua criação, sendo a Usabilidade de *Software* e a Interação Humano-Computador (IHC) (PEREIRA, 2007).

De acordo com a norma ISO 9241 usabilidade é a medida pela qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com efetividade, eficiência e satisfação em um contexto de uso específico. Quando se fala de medida nesse contexto, deve-se levar em consideração os valores resultantes de algum tipo de medição, e quais foram os processos empregados pelo desenvolvedor para chegar naquele resultado.

A efetividade trata-se da permissão que o usuário terá para alcançar determinadas etapas de interação. Neste caso, a efetividade deve ser qualificada de acordo com a finalização de uma determinada tarefa durante o uso pelo usuário, e também, pela qualidade com base no resultado final daquele desenvolvimento.

Para a eficiência, o desenvolvedor deve se atentar a quantidade de esforço e maneiras que o usuário terá que utilizar para chegar até seu objetivo, dependendo do caminho, ou até mesmo do tempo que o usuário leva até chegar ao seu objetivo, isso pode ser levado em consideração ao avaliar um determinado software criado por um desenvolvedor, incluindo *bugs* e erros.

Quando se fala de satisfação, esta definição pode ser algo discutível, pois depende muito do que o usuário quer, e como o mesmo se sente confortável ao estar utilizando determinado software, isso está ligado ao objetivo do usuário ou sua aceitação de acordo com a interface em questão.

A interação humano-computador se caracteriza como uma disciplina que visa, se aprofundar no projeto, avaliação e implementação de sistemas de computação, interativos para

uso humano e todos os fenômenos ao seu redor (HEWETT ET ALL., 1992).

Além disso, a Interação Homem-Computador, visa o estudo de mecanismo e lado humano em conjunto, entretanto, com um foco maior voltado a classe de dispositivos. Uma grande gama de tecnologias e dispositivos estão sendo desenvolvidos para permitir e facilitar a interação entre pessoas e computadores. Os dispositivos de entrada e saída, são a origem de meios físicos responsáveis por intervir nas relações físicas entre pessoas e softwares. Estas relações ocorrem em conformidade com técnicas de diálogo, selecionar opções em um menu interativo utilizando um mouse ou até mesmo preencher uma planilha em Excel utilizando um teclado, por exemplo (BARBOSA, E SILVA 2010).

2.2. Sistemas Embarcados

Sistemas embarcados, podem ser definidos como processos computacionais. Estes processos, contém recursos com um poder de memória elevado e, um processamento otimizado, podendo assim, ter usos especiais e específicos para uma determinada aplicação. Para poder diferenciar um sistema comum de uso e um sistema embarcado, pode-se dizer que, um computador comum é utilizado de diversas maneiras para diversos fins, porém, um sistema embarcado é utilizado com o objetivo de executar apenas uma única tarefa (MEHL, 2011).

Inúmeros sistemas operacionais são produzidos todo o ano, todos de maneira geral e aplicados em diversos segmentos da eletrônica e da tecnologia (STALLINGS, 2017). Alguns exemplos destes segmentos são:

- Automotivo: Computadores de bordo, freios ABS, interface de vídeo e voz;
- Industrial: Sistemas de controle, automação e robótica;
- Aparelhos de comunicação: Equipamentos de rede como switch, roteadores e telefones;

2.3. Microcontroladores

Computadores compactos, também, podem ser chamados de microcontroladores ou computadores de chip único. São dispositivos constituídos por um processador dedicado com o objetivo de executar uma aplicação especial, distintivamente de computadores dedicados ao uso geral (TANENBAUM, 2013). Estes computadores, são ligeiramente parecidos com computadores de uso geral, porém, demonstram uma menor quantidade de elementos em seu hardware e software, concentrados em apenas um chip (HILL, 2018).

Segundo (STALLINGS, 2017), o sistema embarcado é caracterizado por um produto que precisa de um hardware e software. Ainda, de acordo com (BARR, 2006) o sistema embarcado tem o conceito de desempenhar uma tarefa específica, em uma combinação de hardware e software.

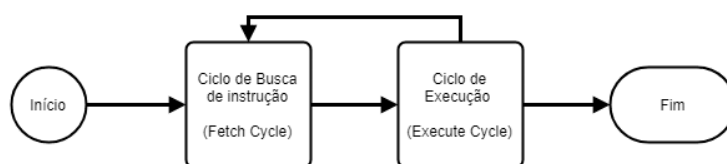
Os microcontroladores são utilizados para executar apenas uma função em um projeto eletrônico, ou em um compilado de aplicações, formando apenas um sistema (PATTERSON, e col., 2013). Ou seja, em seu *software*, existe um determinado sistema adaptado que interage com seu exterior. A partir dessas definições, este sistema pode ser chamado de sistema embarcado.

Em um projeto voltado a sistemas eletrônicos embarcados, o mesmo abrange áreas de diversos segmentos, por conta do avanço tecnológico, a união entre *hardware* e *software* aos poucos se torna ligeiramente frágil (Li, 2003). Seguindo esta ideia, o programador, especialmente de microcontroladores, é fundamental que conheça a estrutura interna do dispositivo, pois o programa se comunica diretamente com o hardware e seus elementos. Levando isso em consideração, é importante que o programador ou desenvolvedor conheça os principais fundamentos de arquitetura e organização de sistemas computacionais.

Para se ter uma ampla visão da organização de um sistema computacional, é importante saber que um computador digital é uma junção de equipamentos e circuitos digitais. Estes se caracterizam por realizar uma sequência programada de procedimentos (TOCCI, 2011). A sequência destes procedimentos se denomina como programa ou *software* (WEBER, 2012).

Basicamente, um sistema operacional se caracteriza por ser um conjunto de instruções. A Figura 1, ilustra este conjunto de instruções codificadas e dados a serem processados de maneira que atinjam o objetivo computacional do programador, desta forma, os dois são armazenados na memória (TOCCI, 2011). Portanto, o sistema operacional tem a função de acessar a memória para adquirir e executar os procedimentos indicados. Contudo, um determinado programa computacional possui uma junção de estruturas que possibilitam esses procedimentos a se tornarem executáveis.

Figura 1 - Ciclos de instrução



Fonte: Fernando Deluno Garcia, 2018.

2.4. Organização de um software computacional de modo geral

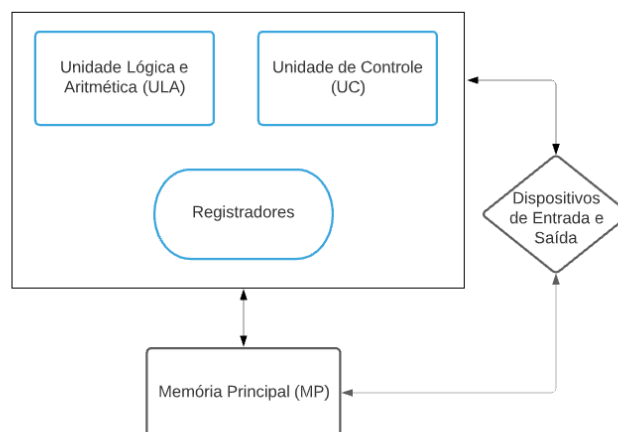
Internamente em um computador, um dos principais componentes para o seu funcionamento é o processador (HILL e col., 2018). Sua função é muito importante, pois o mesmo tem a capacidade de realizar cálculos, processar e enviar recursos para periféricos e executar muito rapidamente processos determinantes para o funcionamento de um sistema operacional (PATTERSON e col., 2013).

Mesmo sendo o principal componente de um computador, o processador obrigatoriamente deve interagir com todos os periféricos acoplados no sistema computacional, como memórias, dispositivos de entrada e saída, conversores de sinais, entre outras importantes partes de um computador.

Ainda que os softwares de computadores tenham suas especificidades, a estrutura desenvolvida internamente, tem a possibilidade de ser generalizada em algumas unidades em particular.

Sendo assim, um software computacional se representa por cinco blocos combinados entre si, são eles: a unidade lógica e aritmética (ULA), a unidade de memória, a unidade de controle (UC), e as unidades de entrada e saída (TOCCI e col., 2011). A seguir, na Figura 2, são ilustrados os 5 blocos de um software computacional, como citados anteriormente.

Figura 2 - Principais componentes de um sistema computacional.

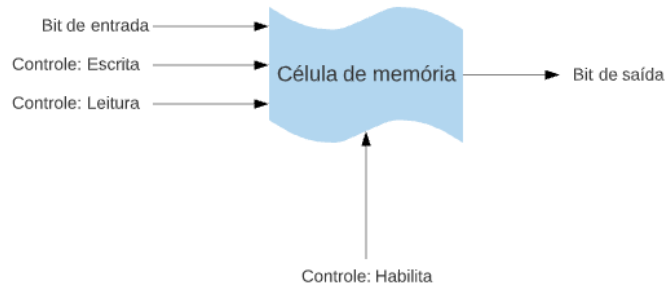


Fonte: O Autor

Em relação a memória, a mesma define-se como: um grupo de elementos que, sua função, é armazenar informação. Se tratando de uma memória semicondutora, o elemento básico se denomina como célula de memória (STALLINGS, 2017). Esta célula (Figura 3) tem a

capacidade de armazenar informações de 1 bit, ou seja, demonstra dois estados estáveis sendo 0 ou 1. Contudo, deve viabilizar a informação de maneira que seja gravada e verificada, sendo lida após sua gravação.

Figura 2 - Célula de Memória



Fonte: O Autor

Para realizar a formação de uma palavra, é necessário entender que se precisa de um conjunto de células de memória. Portanto, essas palavras, constituem-se de um agregado de palavras sendo uma por uma identificadas de um jeito homogêneo por um determinado endereço. Esta palavra, geralmente tem uma característica relevante que é, sua aptidão em armazenar informação, ou seja, a quantidade de *bytes* ou até mesmo *bits* que refletido da palavra.

2.5. Interconexão de um sistema e estrutura da unidade de processamento

Os módulos descritos anteriormente são conectados através de um sistema de interconexão. Este sistema de interconexão tem como principal função a execução destes módulos, também é conhecido como barramento (HILL e col., 2018). Através do barramento os módulos deste sistema de interconexão se comunicam, em outras palavras, trocam informações. Contudo, para realizar estas trocas de informações, deve-se levar em consideração os seguintes tipos de transferências (STALLINGS, 2017):

- Memória para processador;
- Processador para memória;
- E/S para processador;
- Processador para E/S;
- E/S de ou para a memória.

De modo geral, um barramento é estabelecido a partir de um conjunto de atalhos e conexões físicas onde cada atalho tem a possibilidade de transmitir sinais representados por códigos binários, sendo 1 ou 0. Um sistema é constituído por 3 tipos de barramentos diferentes, listados a seguir:

Barramento de dados:

- Trajeto de movimentação dos dados entre os módulos;
- Largura destes dados, sendo fundamental para seu desempenho, podendo ser em: 8,16 e 32 bits.

Barramento de endereço:

- Estes, tem a função de determinar qual a origem e pra onde estes dados serão destinados;
- Largura estabelece qual a maior capacidade de memória.

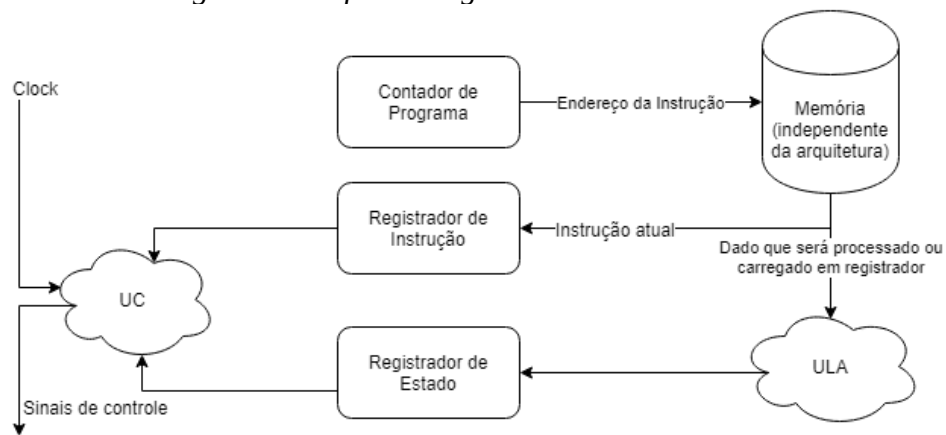
Barramento de controle:

- Contém informações de tempo e controle dos dados;
- Sinais de leitura e escrita de memória, recursos de interrupção.

Em sua arquitetura interna o processador, também chamado de unidade central de processamento (*Central Process Unit* - CPU), tem sua divisão conceituada em duas etapas, sendo cada uma com finalidades particulares. Entretanto, sua composição é constituída por uma ULA e uma UC, de modo geral. Assim, seu principal objetivo como hardware, é a execução e a busca de funções (STALLINGS, 2017).

Com uma grande capacidade de operar em dados binários, a ULA é capaz de executar operações relativamente simples contendo um ou mais operandos (WEBER, 2012). Contudo, a Unidade de Controle (UC) tem a função de coordenar as operações entre as demais unidades, desta forma, estipulando o fluxo operacional do sistema, bem como, ilustrado na Figura 4.

Figura 4 - Arquitetura geral de um sistema.



Fonte: Fernando Deluno Garcia, 2018

Internamente em certas arquiteturas existe uma parte do software muito importante, o registrador acumulador. A função do acumulador é de um operando, armazenando os resultados das operações realizadas pela ULA. Sobretudo, registradores especiais executam funções únicas dentro da unidade de controle (UC) (TOCCI e col., 2011) estas funções:

- Contador de programa: A função deste registrador é indicar o endereço de memória da instrução seguinte, com o intuito de buscar e executar essa instrução;
- Registrador de instrução: Tem como objetivo, armazenar uma instrução que foi codificada, realizando a leitura da memória no endereço apontador através do contador do programa;
- Registrador de estado: Inúmeras operações são executadas pelo processador, algumas informações tem seu armazenamento realizado neste registrador, por exemplo: uma equação matemática resultante em zero, pode ser um indicativo de algo desligando ou ligando.

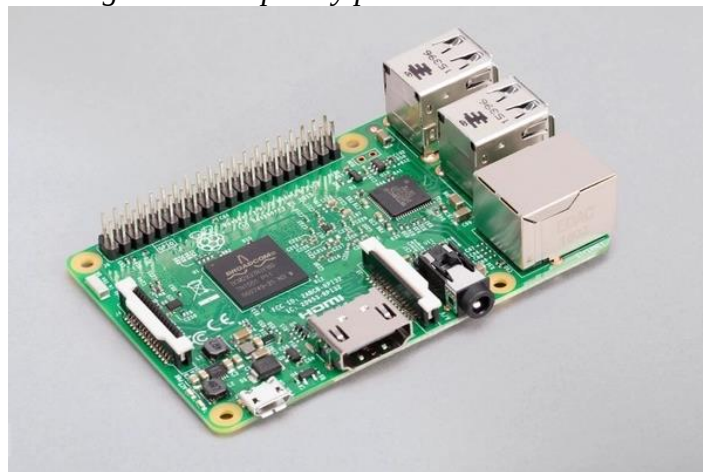
2.6. RASPBERRY PI 3

Em 2006 na universidade de Cambridge (UK) surgiram os primeiros conceitos do Raspberry Pi, baseados no microcontrolador ATmega644. Nesse tempo seu layout e esquemas de placa de circuito impresso (*Printed Circuit Board* – PCB), foram disponibilizadas ao público com o intuito de motivar a criatividade das crianças a desenvolver algo.

Nos dias atuais, várias pessoas tem a vontade de construir, consertar, modificar ou criar suas próprias coisas e, devido a isso, os desenvolvedores amam esta placa conhecida por seu pequeno tamanho e leveza, podendo ser levada até em um bolso ou pelo fato de ocupar pequenos espaços, capaz de executar tarefas que um computador comum pode fazer (FLOP, 2018).

Neste trabalho será utilizado o Raspberry Pi 3 modelo B (Figura 5) para o projeto de automação residencial representado por uma maquete.

Figura 5 - Raspberry pi 3 - modelo B



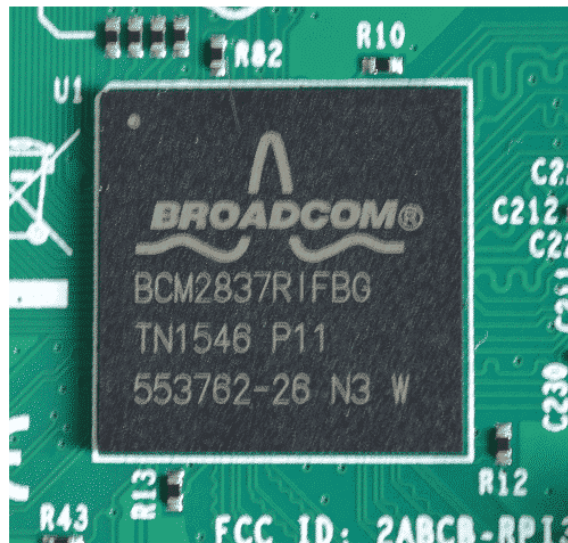
Fonte: <https://www.raspberrypi.org/>

Em relação a modelos anteriores, o Raspberry pi 3 Model B, apresentou as seguintes melhorias, justificando a sua escolha:

- Antena Wi-Fi e conexão bluetooth integrada, não sendo necessário a conexão com uma antena externa
- Novo SoC;
- Broadcom BCM2837 com: 4 processadores de 64 bits ARM Cortex a 1.2GHz;
- GPU igual ao seu antecessor, no entanto, com um clock aumentado para 400MHz.

A seguir, pode ser na Figura 6 o novo SoC BCM28737 na placa.

Figura 6 - SoC BCM28737



Fonte: <https://www.embarcados.com.br/hardware-da-raspberry-pi-3/>

Uma das partes mais importantes desta placa de chip único são as GPIOs, estas, não tiveram mudanças e, todos os seus 40 pinos foram mantidos. Uma ligeira mudança quanto a comunicação serial do dispositivo a (UART), foi implementada nesse modelo, trocaram para a segunda UART pelo fato da primeira ter sido utilizada na comunicação com o BCM43438. Neste caso, o sistema operacional é o responsável internamente por essa mudança, não afetando diretamente o funcionamento dos hardwares acoplados nos pinos das GPIOs (RASPBerryPI.ORG, 2016).

Um último *overview* trouxe um importante detalhe quanto ao seu funcionamento. Devido ao fato de o *clock* ter aumentado em relação a versão anterior e também outros circuitos terem sido implementados na mesma, logicamente, mais corrente elétrica seria necessária para sua alimentação. Portanto, o recomendado para alimentar a Raspberry, é uma fonte que tenha em torno de 2,5 A de corrente (RASPBerryPI.ORG, 2016). Em resumo, os novos recursos encontrados na Raspberry pi3 modelo B, são encontrados no Quadro 1 a seguir, que, ilustra uma comparação entre o modelo utilizado neste projeto e seu antecessor.

Quadro 1 - Dados técnicos Raspberry Pi 2 e Pi3 Modelo B.

Placa	Raspberry Pi 2 Model B	Raspberry Pi 3 Model B
Processador	Broadcom BCM2836	Broadcom BCM2837
CPU	Quadcore ARM Cortex-A7, 32Bit	Quadcore ARM Cortex-A53, 64Bit
Clock	900 MHz	1.2GHz (cerca de 50% mais rápido que a Pi 2)
RAM	1 GB	1 GB
GPU	250 MHz VideoCore IV®	400 MHz VideoCore IV®
Conectividade de rede	1 x 10 / 100 Ethernet (RJ45 Port)	1 x 10 / 100 Ethernet (RJ45 Port)
Conectividade sem fio	–	802.11n Wireless LAN (Wi-Fi) e Bluetooth 4.1, BLE
Portas USB	4 x USB 2.0	4 x USB 2.0
GPIOs	40 pinos	40 pinos
Interface com câmera	15-pin MIPI	15-pin MIPI
Interface com display	DSI 15 Pin / HDMI Out / Composite RCA	DSI 15 Pin / HDMI Out / Composite RCA
Fonte (Corrente)	1.8 A	2.5 A

Fonte: <https://www.embarcados.com.br/hardware-da-raspberry-pi-3/>

2.7. Sensores

De acordo com (BALBINOT, A. E COL. 2011) quando se fala do assunto automação, seja ela industrial ou até mesmo residencial, cujo no momento é o foco deste projeto, os sensores são peças fundamentais para todo o processo e seu funcionamento. Presentes em nosso dia a dia, os sensores são utilizados em diversos ramos e setores industriais e residenciais. Os sensores são largamente utilizados na medicina, agricultura, indústria, robótica e etc. Tudo isso com o intuito de trazerem informações importantes, como de processos físicos, químicos, biológicos, entre outros.

Dito isso, é extremamente importante diferenciar um sensor de um transdutor, mesmo que os dois sejam semelhantes. Um sensor por exemplo, identifica as alterações físicas que são feitas em sua volta, enquanto o transdutor, transforma a quantidade física ou não-elétrica em outro sinal ou sinal elétrico. (AGUIRRE, L.A. 2014).

Para ilustrar de maneira, ainda mais simplificada o que é um sensor, um termômetro,

pode ser usado como exemplo, já que este objeto é conhecido por qualquer pessoa e utilizado por muitos no mundo todo, principalmente por profissionais da saúde. O termômetro, tem como sensor o elemento químico mercúrio, que, de acordo com a temperatura ambiente pode dilatar ou diminuir.

Dito isso, o termômetro pode ser caracterizado como um sensor com a finalidade de medir a temperatura corporal, com o objetivo principal de saber se a pessoa está com sua temperatura elevada ou abaixo do normal (AGUIRRE, L.A. 2014).

2.7.1. Sensor de umidade e temperatura DHT11

O sensor escolhido para este projeto foi o, DHT11, é um sensor especificamente para controlar umidade e temperatura, é considerado um dispositivo ligeiramente complexo e possui uma saída digital. O DHT11, é um sensor com uma boa durabilidade e pode ser usado de maneira constante sem perder seu rendimento (LASTMINUTEENGINEERS.COM, 2020).

Este sensor foi escolhido, pelo fato de que, internamente em sua arquitetura, é um componente de medição resistivo tendo como padrão um coeficiente de variação de resistência negativo chamado de *Negative Temperature Coefficient* (NTC), oferecendo uma resposta e resultados eficientes (LASTMINUTEENGINEERS.COM, 2020).

O sensor DHT11 é produzido em laboratório com a mais alta qualidade, sendo extremamente preciso em suas funções e resultados. O mesmo utiliza o tipo de memória para armazenar seus coeficientes de calibração do tipo *One Time Programmable* (OTP), podendo ser programado apenas uma vez para executar tal tarefa (LASTMINUTEENGINEERS.COM, 2020). No Quadro 2, é possível verificar algumas especificações técnicas do DHT11.

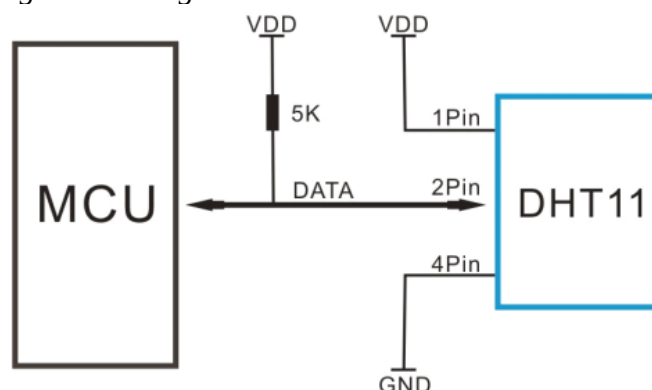
Quadro 2 - Especificações técnicas Sensor de Umidade e Temperatura DHT11

Item	Medição de Alcance	Prec. de Umidade	Prec. De Temp.	Resolução	Pacote
DTH	20-90%RH	±5%RH	±2°C	1	4 Pinos em
11	0-50 °C	X	X	X	Linha única

Fonte: Datasheet

A Figura 7 apresenta um diagrama da conexão entre o DHT11 e o microcontrolador.

Figura 7 - Diagrama de conexões internas do DHT11



Fonte: Datasheet

Caso seja necessário utilizar um cabo com uma extensão menor do que 20 metros, é recomendado que ele utilize um resistor de 5K como mostrado na Figura 7 (Datasheet). No entanto, para cabos maiores do que 20 metros, deve-se adotar um resistor pull-up para esta tarefa (LASTMINUTEENGINEERS.COM, 2020).

A fonte de alimentação para o sensor DHT11 é de 3,5V a 5V em corrente contínua, e geralmente o processo de comunicação entre um computador ou microcontrolador e o DHT11, ocorre em um intervalo de tempo de 4 ms (MOUSER.COM, 2020).

Quando se fala em formato de comunicação de dados entre o microcontrolador e o DHT11, o barramento faz a sincronização dos dados e a comunicação entre o sensor e a MCU, geralmente, esta comunicação dura em torno de 4 ms. Os dados são formados por partes decimais e integrais, para realizar a transmissão de dados completa, a transmissão é feita em torno de 40 bits, com o sensor encaminhando o bit de nível mais alto primeiro (MOUSER.COM, 2020).

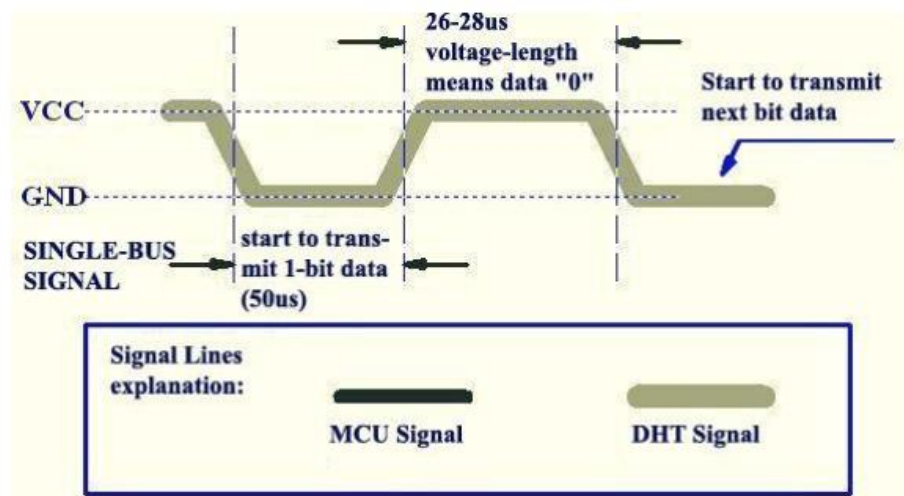
Para iniciar o processo, o DHT11 deve receber um sinal de início, sinal este para começar todo o processo juntamente com a MCU, quando este sinal chega o DHT11 envia outro sinal de resposta em nível baixo de tensão, que irá durar eventualmente em torno de 80 μ s. Logo após isso, a programação interna que envolve o DHT11 deve definir toda a parte de tensão presentes no barramento de dados, que normalmente é baixa nesse caso para uma tensão elevada, mantendo por 80 μ s para assim então, o DHT11 enviar os dados (MOUSER.COM, 2020).

Quando se observa que, o barramento único de dados está em um nível baixo de tensão, isso nos dá a informação de que, o DHT11 está enviando a resposta sinal quanto ao sinal recebido e, logo após o DHT11 envia o sinal de resposta em um intervalo mínimo de tempo de 80 μ s, faz a preparação dos dados e então envia ou transmite os mesmos (MOUSER.COM,

2020).

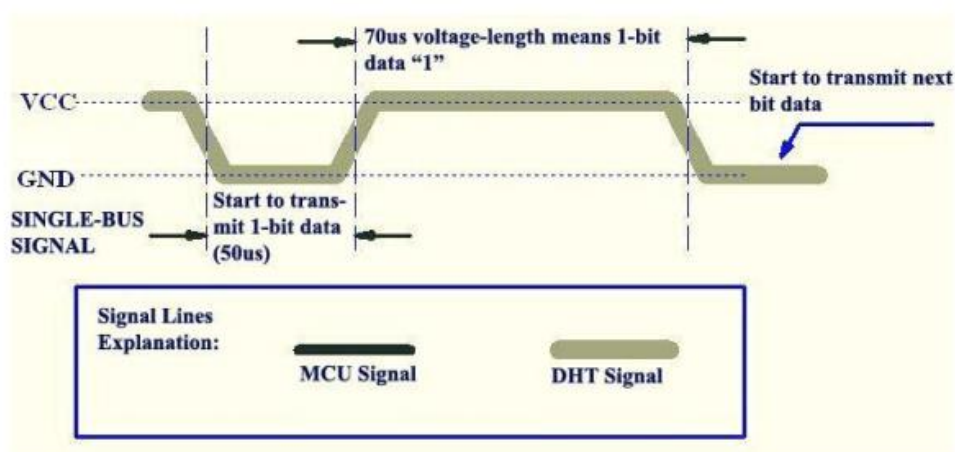
Toda vez que o DHT11 está processando o envio dos dados, todos os níveis lógicos iniciam com uma tensão ligeiramente reduzida em torno de 50 μ s e, o comprimento do sinal em relação a tensão determina se o bit de dados é “0” ou “1”. A seguir de maneira ilustrada, a Figura 8, mostra como é este processo (MOUSER.COM, 2020).

Figura 8 - Processos internos DHT11.



Fonte: Datasheet

Figura 9 - Processos internos DHT11.



Fonte: Datasheet

Pode-se ver que, na Figura 3 o processo de dados é indicado pelo BIT “0” e na Figura 9 a indicação é feita pelo BIT “1”.

Caso seja observado que o sinal está constantemente em um nível elevado de tensão, dá a entender que, o DHT11 pode não estar funcionando corretamente. Neste caso, o ideal é verificar a conexão para corrigir este problema (MOUSER.COM, 2020).

Portanto, toda vez que o último BIT de dados é transmitido, o DHT11 retornar seu nível de tensão para 50 μ s e mantém o mesmo neste nível. Depois disso, a tensão de barramento única é puxada pelo resistor de maneira que o DHT11 é restaurado retornando ao seu status livre (MOUSER.COM, 2020).

Quadro 3 - Especificações técnicas de Alimentação do DHT11

Detalhes	Condições	Mínimo	Típico	Máximo
Fonte de energia	CC	3V	5V	5.5V
Fornecimento atual	Medindo	0.5mA		2.5mA
	Média	0.2mA		1mA
	Espera	100uA		150uA
Período de Amostragem	Segundo	1		

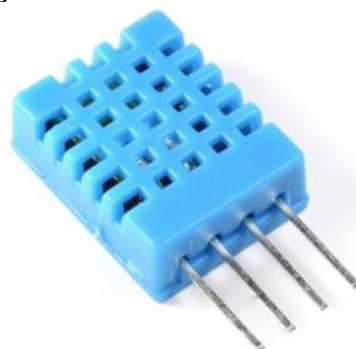
Fonte: Datasheet

Em relação ao Quadro 3 e suas informações, vale ressaltar que em todo o período de amostragem, o valor não deve ser menor que 1.

Uma boa observação a ser feita em relação ao DHT11, é que se possível o mesmo deve ser mantido longe de temperaturas elevadas pois, é recomendável que ele trabalhe sempre em condições de temperatura iguais ou semelhantes para não haver maiores danos, recomenda-se temperaturas entre 10° e 40° graus celsius para manter a integridade do dispositivo.

Em seguida, a Figura 10, ilustra como é um sensor DHT11.

Figura 3 - Sensor DHT11.



Fonte: Filipeflop (2020)

2.1 Sensor de umidade e temperatura DHT22

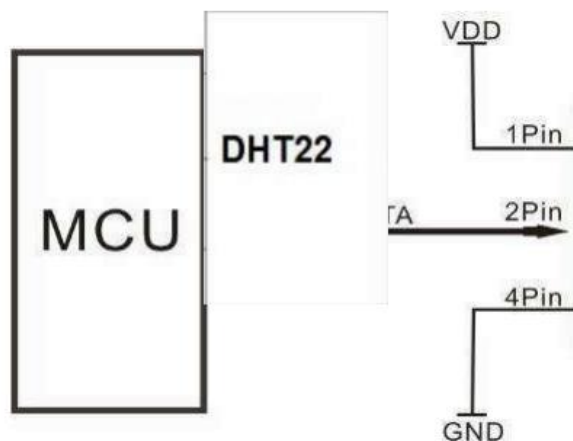
Assim como o sensor DTH11, o DTH22 também é um sensor utilizado com o intuito de monitorar a umidade e temperatura. Ambos contêm especificações semelhantes, mudam em poucos aspectos, então, um breve resumo em relação ao DTH22 será descrito a seguir, o Quadro 4 apresenta as informações mais relevantes para o mesmo ser comparado com seu antecessor o DTH11.

Quadro 4 - Especificações técnicas DHT22.

Modelo	DHT22
Fonte de energia	3.3-6V CC
Sinal de saída	Sinal digital via barramento único
Deteção	Capacitor de polímero
Limites Operacionais	Umidade 0-100% RH; temperatura -40 ~ 80Celsius
Precisão	Umidade + -2% de umidade relativa (Max + -5% de umidade relativa); temperatura <+ - 0.5Celsius
Repetibilidade	Umidade + -1% de umidade relativa; temperatura + -0,2 graus Celsius
Período de detecção	Média: 2s
Dimensões	Tamanho pequeno 14 * 18 * 5.5mm; tamanho grande 22 * 28 * 5mm

Fonte: Datasheet

Figura 11 - Diagrama de conexões DHT22.



Fonte: Datasheet

Assim como ilustra a Figura 11, o primeiro terminal do DHT22 é o terminal de alimentação, sendo conectado ao potencial positivo da fonte de alimentação. O terminal 2 é denominado Data, ou terminal de dados, esse terminal é conectado ao microcontrolador, no caso deste projeto estaria conectado, ao Raspberry pi 3 B. O pino de dados, tem a função de receber os dados coletados do ambiente e, repassar ao dispositivo onde neste caso, será

manipulado pela Raspberry pi 3 B. O terceiro terminal é nomeado como N/A (*not applicable*) e não possui uma aplicação, podendo ficar desconectado do circuito.

O quarto terminal é terra ou GND (Ground), devendo ser conectado ao menor potencial da fonte de alimentação, conforme Figura 11 (CDN-SHOP.ADAFRUIT.COM, 2020).

Contudo, um dos principais aspectos em uma escolha de um dispositivo se tratando de um projeto como este, geralmente, se faz em relação ao custo benefício e, analisando as informações de ambos, é notável que, o DHT22, se torna ligeiramente mais caro.

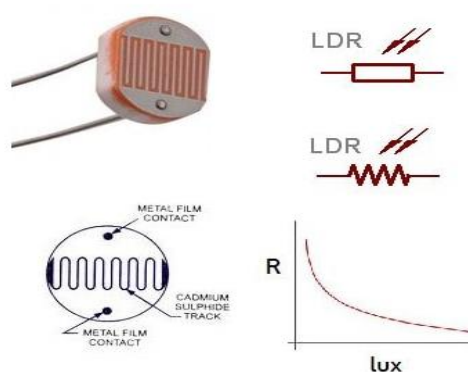
O sensor DTH11, tem o valor em torno de R\$18,90 e, já o sensor DTH220, pode ser encontrado por, R\$49,90 podendo ter variação assim como o, DHT11. Portanto, o sensor escolhido para este projeto, se tornou vantajoso em relação ao custo benefício, satisfazendo o propósito do projeto (FILIPEFLOP.COM, 2020).

2.2 Sensor LDR

Os *Light Dependent Resistors* (LDRs), são componentes sensíveis à luz, quer dizer que, são dispositivos eletrônicos que funcionam de acordo com a intensidade da luz incidente em sua superfície (MUNDO PROJETADO, 2017).

Na Figura 12 pode-se ver o símbolo utilizado para representar um LDR e seus aspectos mais comuns a serem encontrados a cerca destes componentes.

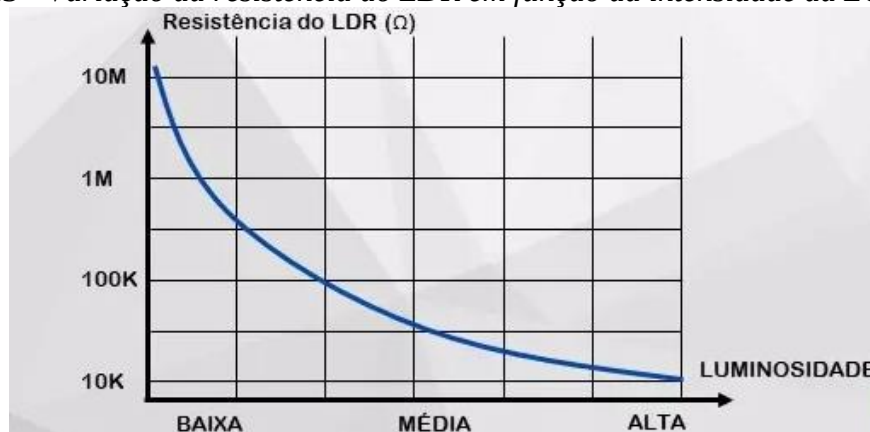
Figura 12 - LDR e seus aspectos mais comuns.



Fonte: ScienceProg (2020)

Da Figura 13 observa-se a curva característica do LDR, onde a resistência elétrica deste componente é maior no escuro ou quando não há luminosidade na ordem de Mega Ohms (MΩ). Em contrapartida, reduzindo a resistência quando o LDR é iluminado diretamente.

Figura 13 - Variação da resistência do LDR em função da intensidade da LUZ

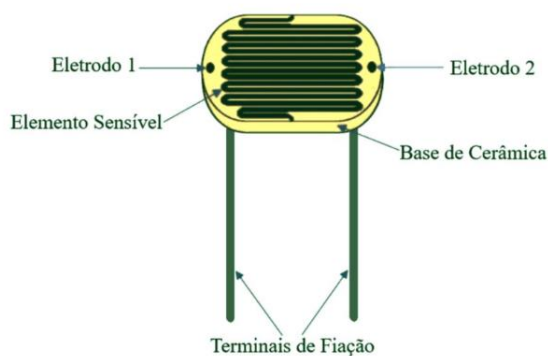


Fonte: Manual da Eletrônica (2020)

A superfície do LDR é constituída por um material semicondutor, como ilustra a Figura 14 chamado Arseneto de Gálio (GaAs), o qual possibilita a mudança da sua resistência de acordo com a intensidade da luz incidente sobre sua superfície.

O seu princípio de funcionamento baseia-se nos fótons ou partículas de luz incidentes no material, que liberam elétrons do semicondutor diminuindo sua resistência (BRAGA, 2013). Portanto, é um resistor onde, sua resistência, altera de acordo com a intensidade da luz, ou iluminação do ambiente que incide sobre ele.

Figura 14 - Foto resistor (LDR)



Fonte: Scielo (2020)

Para o projeto, o LDR será utilizado para o monitoramento dos ambientes residenciais, realizando a medição da luminosidade internamente à residência. Como parâmetro, representado em uma maquete, fora utilizado para este controle, apenas um cômodo em uma residência de tamanho real.

2.3 Módulo Relé

Componentes eletrônicos como o relé, são muito comuns em inúmeras aplicações na eletrônica. O relé, é muito utilizado para acionamento de dispositivos com uma elevada potência, mas, que normalmente, são dispositivos com baixo sinal de corrente e tensão, como por exemplo um ar-condicionado.

O relé, do termo em inglês *Relay*, é um componente com característica eletromecânica, onde, o mesmo adequa-se como um interruptor advindo de um sinal, este sinal juntamente com o interruptor é isolado entre si. O interruptor se encontra na parte interna do relé, e quase sempre suporta uma alta capacidade de tensão e corrente, não importando o seu tamanho (ATHOS ELECTRONIC, 2016).

Figura 15 – Módulo Relé utilizado no projeto



Fonte: Filipeflop.com (2020)

Segundo ATHOS ELECTRONIC (2016), o módulo relé conforme representado na Figura 15, é utilizado em diversas aplicações de automação residencial. Por isso, se tornou um importante aliado para o desenvolvimento deste projeto. Este componente pode ser utilizado em automação de portões, janelas eletrônicas, acionamento de lâmpadas como nesse projeto e etc. Este pequeno componente pode ser acionado através de um pequeno computador como o Raspberry pi 3, polarizando sua entrada com 5 V ou 12 V.

Em sua parte interna, o relé contém uma bobina e um contato fixado em uma mola de rearme, conectando estes dois elementos em dois terminais, o normalmente Aberto (NA) e o normalmente Fechado (NF). Quando o contato se encontra em seu estado de repouso, o mesmo se mantém na posição NF. Quando a bobina recebe energia, ao se redor, cria-se um campo eletromagnético, funcionando como um ímã, fazendo com o que o contato se mova, atraindo e

deslocando o mesmo. Portanto, o relé se desconecta do contato centralizado, passando a se conectar no contato NA.

Para ATHOS ELECTRONIC (2016), uma grande vantagem nos relés utilizados na eletrônica, são que os contatos suportam geralmente, correntes de 10 amperes ou maiores em 250 volts, mesmo a bobina sendo acionada com sinais pequenos de 5 volts. Sendo assim, é possível acionar cargas elevadas de dispositivos e equipamentos com uma alta potência, mas com um sinal de baixa tensão, que pode até mesmo ter origem de um Raspberry pi3 por exemplo.

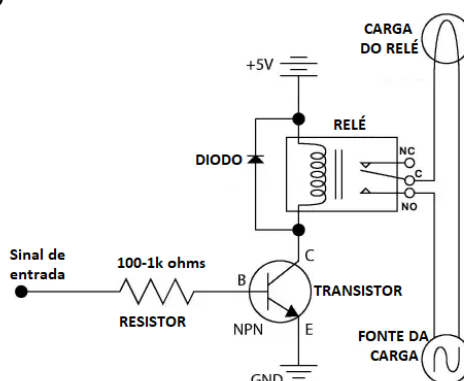
2.7.2. Acionando o Circuito Interno

A bobina do relé deve ser polarizada, e para isso, é essencial a utilização de um circuito de proteção, conectando o mesmo a bobina. Isso deve ser feito, pois, quando a corrente elétrica se desloca pela bobina, o campo eletromagnético ao seu redor pode variar, gerando correntes reversas em sua volta, que podem causar um determinado dano ao circuito. Circuito esse, que está enviando comandos a partir de sinais para o relé.

Por conta disso, aplica-se um diodo paralelo a bobina, e uma alimentação externa dedicada a alimentação da bobina. O chaveamento é constituído por um transistor NPN ou PNP, que herdará em sua base o sinal de comando para executar ou não o acionamento da bobina (ATHOS ELECTRONIC, 2016).

Lembrando que este circuito não é obrigatório, mas acionar um módulo relé sem este circuito pode danificar o circuito principal de comando, como por um exemplo de um microcontrolador ou até mesmo de um Raspberry pi3. A seguir, pode-se ver uma ilustração deste circuito na Figura 16.

Figura 16 – Circuito módulo relé



Fonte: Athoselectronics (2016)

2.7.3. Módulo Relé utilizado no projeto

De acordo com ATHOS ELECTRONIC (2016), existem módulos que em sua composição estão inclusos o circuito de acionamento prontos para utilização. Neste projeto o módulo relé escolhido contém este circuito de acionamento integrado e soldado na placa. O modelo utilizado, pode ser visualizado na Figura 17.

Figura 4 - Módulo relé eletromagnético



Fonte: O Autor

O módulo relé utilizado neste projeto, contém 4 canais para utilização, conforme Figura 19. O componente dispõe de 4 entradas denominadas (IN), um VCC para alimentação e um GND para o aterramento do módulo.

Para o projeto, o módulo relé utilizado, teve o objetivo de controlar a iluminação, realizando a função de substituir os interruptores, ligando e desligando as lâmpadas da maquete. Os quatro relés funcionam como chaves eletrônicas, fechando o contato (NA), e acendendo as lâmpadas, assim que pressionado o botão no Adafruit IO.

O controle de cada relé é feito de acordo com a escolha da GPIO no Raspberry. Todas as portas são definidas como saídas e alternam os estados LOW e HIGH. No caso deste projeto, o estado baixo (LOW) é que faz o acionamento do relé.

2.8. PROTOCOLO MQTT

Quando se trata de equipamentos voltados para IoT, ter conectividade com a internet é algo imprescindível. Esta conexão, concede aos dispositivos que trabalham entre si, a possibilidade de serviços com *backend*. O protocolo de rede contido na Web, é o TCP/IP, estando entre os mais utilizados e mais populares. Seu desenvolvimento, foi realizado na estrutura da pilha TCP/IP, sendo assim, o *Message Queue Telemetry Transport* (MQTT) se tornou o padrão em comunicações de IoT (BASÍLIO, 2020).

Inicialmente, o MQTT, teve sua criação e seu desenvolvimento feito pela IBM no final da década de 90. De início, sua principal aplicação era, realizar conexão de sensores em pipelines de petróleo, ligando-os com satélites. Mesmo podendo confundir o usuário, por conta de seu nome, o MQTT define-se como um protocolo de mensagem que oferece um suporte para que as partes envolvidas tenham comunicação assíncrona (BASÍLIO, 2020).

Contendo como uma de suas principais características, o protocolo, é composto por mensagens assíncronas e, por conta disso, é muito utilizado sistemas de redes vulneráveis. Ainda que seu nome sugira, o MQTT não tem relação com filas de mensagens, sendo assim, o mesmo utiliza um sistema de publicação e assinatura para, realizar a comunicação entre os envolvidos. Ao fim de 2014, o protocolo tornou-se de maneira oficial um padrão aberto OASIS, suportando estruturas para linguagens de programação conhecidas, utilizando inúmeras implementações de *software* livre (BASÍLIO, 2020).

2.8.1. PORQUE ESCOLHER O MQTT?

O MQTT, se tornou um padrão em aplicações de IoT, devido à dois fatores: ser leve e flexível (YUAN, 2017). Estes aspectos, entregam o conjunto ideal para os desenvolvedores de IoT. Logo:

- O termo leve, é que, o protocolo permite implementações em *hardwares* com dispositivos bastante restritos com largura de banda limitada e de latência elevada.
- Sua flexibilidade, viabiliza sua utilização em variados cenários de aplicativos e equipamentos com serviços de IoT.

Boa parte dos desenvolvedores, já se familiarizou com os serviços web utilizando o protocolo HTTP. Devido a isso, surge o questionamento, por qual motivo os dispositivos IoT não se

conectam aos serviços da web?

- O protocolo HTTP, é do tipo síncrono. Desta forma, o servidor aguarda a solicitação do cliente. Sendo assim, se tratando de IoT, normalmente, muitos dispositivos são utilizados ao mesmo tempo, e provavelmente, uma rede com muitas lacunas e sem um grau de confiabilidade maior, deixa a comunicação síncrona com problemas.
- O HTTP é unilateral, ou seja, o cliente é quem inicia a conexão. Se tratando de um aplicativo ou de uma interface WEB para IoT, os equipamentos ou sensores são geralmente os clientes, isso significa que, eles não podem receber comandos da rede passivos, pelo fato de, não interferirem nas informações em deslocamento.
- O HTTP é um protocolo 1-1, sendo assim, o cliente solicita uma ação e o servidor o responde. Com o protocolo HTTP, a transmissão de dados para vários dispositivos ao mesmo tempo se torna ligeiramente mais difícil e mais cara, onde, em IoT, isso é algo normalmente mais comum.

O protocolo MQTT, se tornou extremamente acertado para desenvolvedores de IoT e, por conta disso, foi utilizado neste projeto como protocolo de comunicação.

2.8.2. O modelo de publicação e assinatura

O protocolo MQTT, tem como base, dois elementos em sua rede: um *message broker* e inúmeros clientes. O broker, é denominado como um servidor, sua principal atribuição, é receber qualquer mensagem que tenha como origem os clientes e, depois disso, o mesmo roteia essas mensagens para os clientes designados, onde estes, são considerados os principais clientes. Normalmente, um cliente é, um sensor ou algum tipo de serviço, basicamente é algo que, possa se comunicar com o broker recebendo suas mensagens (BASÍLIO, 2020).

Um sensor em IoT, é considerado um cliente, no caso deste projeto por exemplo, tem-se como cliente: DHT11, o LDR e o PIR, sensores de umidade e temperatura, luminosidade e movimento. Sobretudo em campo, ou, até mesmo em um sistema junto de um data center que, realiza a implementação de dados de IoT. Para contextualizar o Broker MQTT, pode-se ver a seguir alguns passos para o entendimento de seu funcionamento.

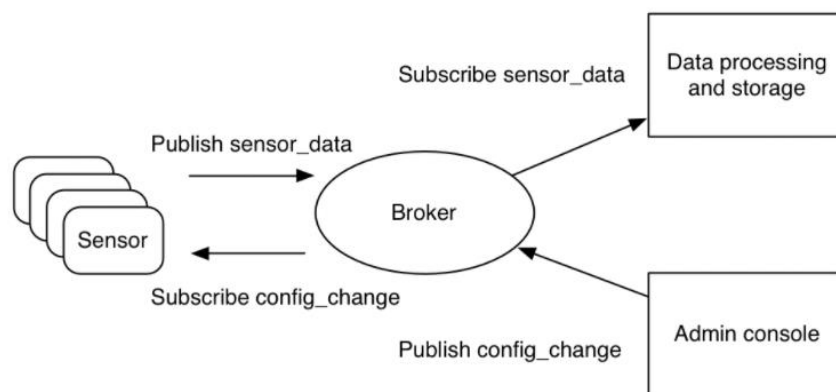
1. O cliente, realiza sua conexão ao broker. O mesmo tem a possibilidade de assinar qualquer “tópico” de mensagem atrelado ao broker. Tal conexão, é possível ter uma conectividade em TCP/IP comum, ou então, uma conexão TLS com criptografia, em mensagens com maior importância.
2. O cliente, efetua a publicação das mensagens em um tópico, destinando-as em conjunto com o tópico, ao servidor broker.
3. Posteriormente, o broker conduz a mensagem para todos os clientes envolvidos na assinatura desse tópico.

Pelo fato das mensagens do MQTT serem segregadas em tópicos, o criador de softwares dispõe de uma flexibilidade para classificar determinados clientes, e estes, podem interagir exclusivamente com certos tipos de mensagens. Em resumo, os clientes ou, nesse caso, os sensores deste projeto, irão publicar suas decodificações no tópico “*sensor_data*” e assinarão o tópico “*config_change*”.

Falando sobre programas de processamento de dados que, guardam os dados dos sensores, o armazenamento é feito internamente dentro de um banco de dados em *backend*, e em seguida, assinam o tópico “*sensor_data*” (BASÍLIO, 2020).

Um aplicativo ou uma interface WEB como o Adafruit IO por exemplo, utilizado neste projeto, tem a possibilidade de gerenciar esses tópicos como um console administrativo, consegue obter comandos do gestor de dados do software com o intuito de regular os parâmetros dos clientes, como por exemplo, a intensidade da luz sob a superfície do LDR, trazendo estes dados em uma *dashboard*. Estes dados, são publicados com suas alterações no tópico “*config_change*” conforme Figura 18 (BASÍLIO, 2020).

Figura 18 - Modelo de publicação e assinatura do MQTT para sensores de IoT



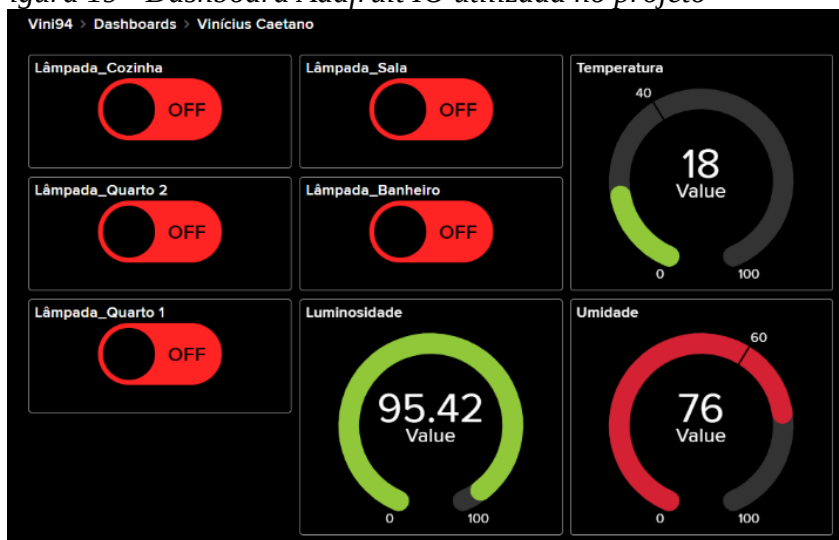
Simultaneamente, o MQTT é leve. O protocolo, possui um cabeçalho descomplicado que, determina qual mensagem, um tópico textual e, em sequência, uma carga proveitosa binária arbitrária. O software, consegue utilizar, diferentes dados para a carga útil como JSON, XML, binário criptografado ou Base64, contudo, os clientes que irão receber estes dados, precisam analisar a carga útil (YUAN, 2017).

2.9. Adafruit IO

O Adafruit.IO é um serviço em nuvem, ou seja, o usuário precisa se preocupar apenas em executar suas ferramentas dentro dele, todo o gerenciamento é feito pela própria plataforma do Adafruit IO. A conexão com o serviço é totalmente feita pela internet, podendo ser por computador, tablet, notebook, celular, etc. A principal funcionalidade da ferramenta, é armazenar e recuperar dados, entretanto o mesmo disponibiliza muito mais do que somente isso.

2.9.1. Adafruit IO, como funciona?

Figura 19 - Dashboard Adafruit IO utilizada no projeto



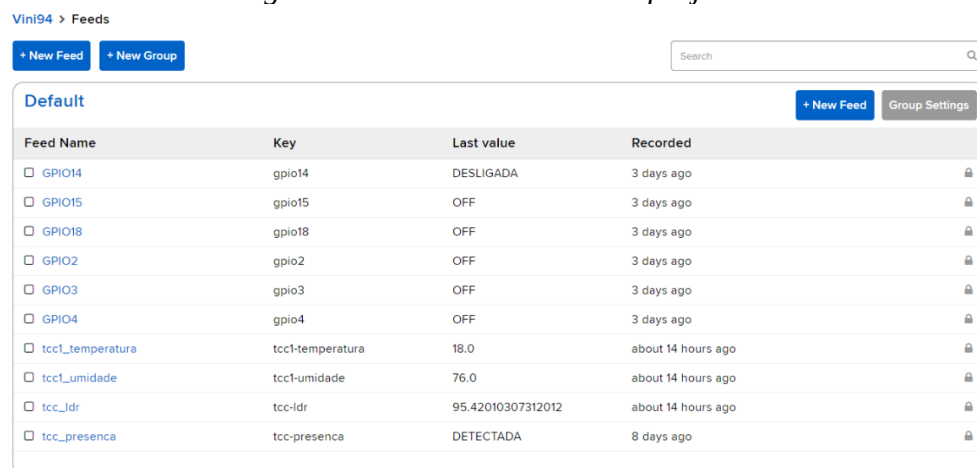
Fonte: O Autor

Neste projeto, o Adafruit foi utilizado para o desenvolvimento da Interface WEB do mesmo conforme apresentado na Figura 19, com o objetivo de mostrar os valores obtidos de: umidade e temperatura através do sensor DHT11, intensidade da luz por meio do sensor LDR, dar ao usuário a possibilidade de verificar a movimentação em sua casa, através do sensor de

movimento PIR e, controlar a iluminação de sua residência, contando com o módulo relé como interruptor.

O Adafruit IO, trabalha com mecanismos chamados “feeds” que, são os núcleos do Adafruit IO. Pode-se dizer que, os feeds, são responsáveis por guardar os dados que foram carregados e os metadados sobre os dados que seus sensores DHT11, LDR e PIR e o módulo relé, enviam para o Adafruit IO, conforme Figura 19.

Figura 20 - Feeds utilizados no projeto.



Feed Name	Key	Last value	Recorded
☐ GPIO14	gpio14	DESLIGADA	3 days ago
☐ GPIO15	gpio15	OFF	3 days ago
☐ GPIO18	gpio18	OFF	3 days ago
☐ GPIO2	gpio2	OFF	3 days ago
☐ GPIO3	gpio3	OFF	3 days ago
☐ GPIO4	gpio4	OFF	3 days ago
☐ tcc1_temperatura	tcc1-temperatura	18.0	about 14 hours ago
☐ tcc1_umidade	tcc1-umidade	76.0	about 14 hours ago
☐ tcc_ldr	tcc-ldr	95.42010307312012	about 14 hours ago
☐ tcc_presenca	tcc-presenca	DETECTADA	8 days ago

Fonte: O Autor.

Ainda falando sobre o *dashboard*, esses painéis são um recurso integrado ao Adafruit IO que, dão a permissão ao usuário para controlar através de botões, ou widgets, o controle de iluminação de sua residência, por exemplo, como foi feito neste projeto, conforme Figura 20. O usuário pode visualizar seus painéis e, realizar este controle de qualquer lugar do planeta, através de um celular, computador ou qualquer dispositivo conectado com a internet.

Depois de toda essa introdução deste bom serviço em nuvem, inclusive utilizado neste projeto, pode surgir a pergunta. Porque não utilizar ferramentas mais completas como IBM CLOUD este citado no capítulo 2.10.4 desta monografia, e Aws iot da AMAZON? A resposta, se torna um tanto quanto simples.

O serviço em nuvem da Adafruit é uma alternativa simples, fácil de escalar, além de ser mais vantajoso financeiramente, podendo utilizar gratuitamente com algumas restrições, entretanto, que cumprem muito bem com as funções solicitadas deste projeto.

Pode-se dizer que, se o desenvolvedor precisar de uma ferramenta rápida e prática para utilizar em sua automação residencial, o mesmo pode contar com uma tecnologia que irá atendê-lo bem e com bons resultados.

Sobretudo, se integrar o serviço com por exemplo, o Raspberry PI 3, incluindo a linguagem de programação Python para integrar o microcontrolador com o Adafruit IO, o usuário terá sucesso em sua aplicação.

2.10. Linguagem de programação Python

Criada no final da década de 80 por Guido Van Rossum, Python é uma linguagem muito utilizada atualmente. Van Rossum, foi considerado um importante programador e pesquisador do Instituto Nacional de Matemática e Ciência da Holanda. Python, surgiu por conta da necessidade de aprimorar outro software de programação, conhecido como ABC. (COUTINHO, 2019).

Este processo de melhoria, tinha o objetivo de torná-lo mais ágil e otimizar a sua capacidade relação com um software de interesse na época, reduzindo o tempo de execução dos programas garantindo maior eficiência do conjunto (COUTINHO, 2019).

As linguagens de programação desempenham um papel de extrema importância dentro da internet na qual conhecemos hoje. Esse papel, não se faz presente apenas no desenvolvimento de sites e sistemas operacionais que, são suas aplicações mais notórias, mas também em áreas mais complicadas como ciência de dados, interações entre humanos e máquinas através de inteligência artificial e coleta de informações, entre outros tipos de soluções (ROVEDA, 2020).

No entanto, ao contrário do que parece, para o público desconhecedor deste tema, quanto mais complexas são as funções executadas em um sistema, mais intuitiva deve ser sua utilização. Esta é uma das principais razões pelas quais fazem com que, as linguagens de programação de entendimento mais fáceis têm ganhado espaço nos últimos anos e claro que Python está entre as mais amadas e utilizadas (ROVEDA, 2020).

A filosofia por trás da criação da linguagem Python, visa valorizar o esforço humano sobre o computacional. Python, é uma linguagem de programação acessível e bastante popular em departamentos provenientes da indústria de tecnologia (ROVEDA, 2020). Caso o desenvolvedor queira saber mais sobre Python, seja como programador ou apenas um entusiasta, a seguir neste capítulo, serão abordadas, algumas aplicações com Python, no que o Python foi utilizado nesse projeto e qual sua função no mesmo.

2.10.1. Quais são os tipos de aplicações que podem utilizar Python?

Python pode ser utilizada em inúmeras aplicações, por conta da facilidade de compreender a linguagem, ter seu código aberto e utilização gratuita. A seguir, serão destacadas algumas aplicações da linguagem em questão.

- **Desenvolvimento WEB:** A linguagem de programação Python é demasiadamente utilizada no *backend* de sites e sistemas Web. Sites famosos como: Instagram, Spotify e Reddit, são exemplos de desenvolvimento web bem-sucedidos utilizando Python (FERNANDES, 2020).
- **Ciência de Dados:** Python é muito utilizada em pesquisas científicas e computação, pelo fato de conter inúmeras bibliotecas científicas e específicas para esse tipo uso, incluindo:
 1. atropy para astronomia;
 2. biopython para biologia e bioinformática;
 3. numpy um recurso de dados fundamental para computação científica com Python;
 4. scipy completa o módulo numpy, com foco na matemática, ciências e engenharia;
 5. pandas concede estruturas de dados de alta performance com fácil utilização e incluindo ferramentas de análises de dados (FERNANDES, 2020).
- **Machine Learning:** O *Machine Learning*, traz a ideia de que os computadores tem a possibilidade de aprender ativamente ou, de operar somente com o que foi codificado.

Essa tecnologia oferece um procedimento totalmente novo para resolver problemas e na dianteira do aprendizado de máquina está o Python (FERNANDES, 2020).

- **Internet das Coisas – *Internet of Things*:** A Internet das Coisas varia por diferentes definições, aqui se trata, sobre objetos físicos, no caso deste projeto como: um sensor de umidade e temperatura por exemplo, conectado a um

software embarcado que os conecta à internet.

Os projetos de IoT normalmente abrangem análises e processos em tempo real, equivalente aos sensores utilizados neste projeto como o LDR e DHT11 onde, a linguagem de programação tem de ser leve, performática e escalável, sendo ideal para o Python (FERNANDES, 2020).

No presente projeto, Python entrou como uma das ferramentas bases para o seu desenvolvimento. A Linguagem, teve a função de realizar a interface do Adafruit IO citado anteriormente, com o Raspberry PI 3. Isso foi feito, em conjunto com o método de publicação e assinatura do protocolo MQTT.

2.11. Tecnologias Semelhantes no Mercado Atual

- **Sonoff Relé Wifi**

Este dispositivo, conhecido como um bom mecanismo para automações residenciais de baixo custo, controla o acionamento de cargas AC por meio de um smartphone ou tablet, utilizando conexão WiFi.

É um dos projetos IoT mais procurados atualmente, por conta da facilidade com que o usuário tem de automatizar. Quando o assunto é automação residencial, uma das primeiras coisas que vem como ideia é: acionamento de lâmpadas ou controle de iluminação através de WiFi. Dito isso, se o desenvolvedor busca uma solução simples, prática e com um bom custo benefício, onde queira controlar remotamente a iluminação da sua casa, o Sonoff Básico é a opção ideal (OLIVEIRA, 2017).

Figura 21 - Sonoff básico.



Fonte: MasterWalker Eletrônico Shop

O Relé WiFi Interruptor Inteligente Sonoff, Figura 21, é um equipamento IoT, que possibilita o

acionamento de cargas AC remotamente através de conexão WiFi. O Sonoff Básico é composto basicamente por um relé, um ESP8285 (versões antigas do módulo utilizam o ESP8266), um regulador de tensão AC DC que permite alimentar a placa com tensão AC na faixa de 90 a 250V e memória flash. Além disso, o mesmo possui acabamento em uma pequena caixa em ABS.

A constituição do Sonoff Básico, se baseia no ESP8285, este módulo contém um firmware que faz com que o módulo se conecte com um servidor nuvem através de uma rede WiFi, deste forma permitindo que todo o controle acerca da placa seja feito remotamente. O aplicativo que acompanha o Sonoff, é o eWeLink, quando utilizado em seu smartphone ou tablet, o software torna a solução e o controle do Sonoff Básico, ligeiramente mais simples (OLIVEIRA, 2017).

Este Sonoff Básico, tem a possibilidade de ser integrado ao Google Home e a Amazon Alexa, logo, você pode fazer o uso destes assistentes pessoais para o acionamento do módulo utilizando comandos de voz.

Vale ressaltar que, o Sonoff Básico não faz apenas aplicações de controle de lâmpadas através de redes sem fio. O usuário pode realizar o acionamento de outros tipos de cargas AC, desta maneira, deixando sua residência como uma casa inteligente. A seguir, pode-se ver no Quadro 5, as especificações técnicas do Sonoff básico (OLIVEIRA, 2017).

Quadro 5 - Especificações técnicas Sonoff

Sonoff Básico - Especificações técnicas	
Base	Módulo ESP8285
Alimentação	Entre 90 a 250VAC
Corrente Máxima	10A
Potência Máxima	de 1270 W em 127V/ 2200W em 220V
Conexão	Redes Wifi padrão 802.11 b/g/n;
Segurança do Wifi	WPA/WPA2
Faixa de Operação Wifi	2.4GHz
Sinal	Antena Integrada
Temperatura em Operação	0 a 40º C
Humidade em Operação	5 a 90%
Interatividade	Aplicativo próprio para Android e iOS

Fonte: blogmasterwalkershop.com.br

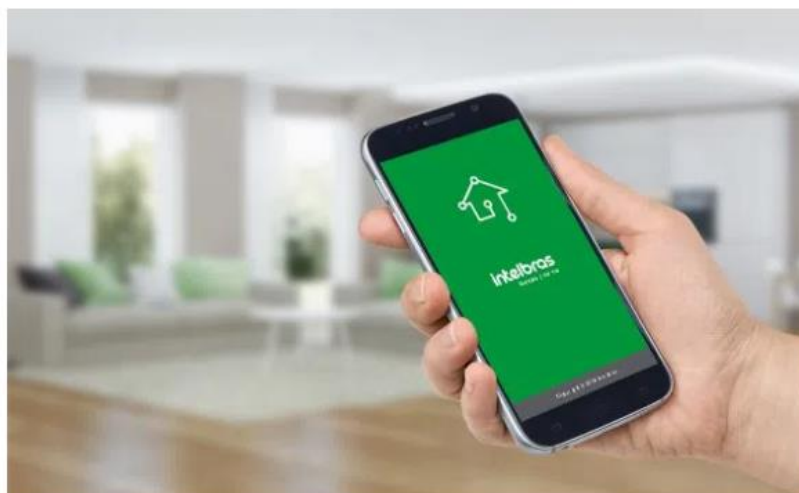
- **Intelbras Mibo Home**

Assim como em seu concorrente, o Sonoff, o Mibo Home vem forte no mundo da internet das coisas, promete transformar sua casa em uma casa inteligente, de maneira simples e objetiva. O Sistema de Automação Mibo Home traz ao usuário, facilidade em automatizar tarefas cotidianas, simplificando e deixando sua vida mais inteligente (INTELBRAS, 2020).

O Mibo Home, contém um aplicativo próprio, onde o mesmo pode monitorar o status de diversas

aplicações como umidade e temperatura, ligar e desligar lâmpadas, ligar e desligar outros dispositivos como ar-condicionado, entre outros (INTELBRAS, 2020). Na sequência, é possível visualizar o software de Intelbras na Figura 22.

Figura 22 - Aplicativo Intelbras



Fonte: <https://loja.intelbras.com.br/sistema-automacao-residencial-mibo-home>

De acordo com o site do fabricante, as especificações técnicas do mesmo são as seguintes, citadas no Quadro 6:

Quadro 6 - Especificações técnicas Mibo Home Intelbras

Hub 3.0 - Especificações técnicas	
OS	Linux
Alimentação	Entre 90 a 250VAC
Corrente Máxima	10A
Tensão:	100 ~ 240V, 50/60Hz
Zigbee	Versão Zigbee 3.0
Comandos	Envio e controle por Infra Vermelho
Conexão	Redes Wifi padrão 802.11 b/g/n;
Segurança do Wifi	WPA/WPA2
Faixa de Operação Wifi	2.4GHz
Sinal	Antena Integrada
Temperatura em Operação	0 a 40° C
Humidade em Operação	5 a 90%
Interatividade	Aplicativo próprio para Android e iOS

Fonte: <https://loja.intelbras.com.br/sistema-automacao-residencial-mibo-home>
O mesmo utiliza dois sensores, e um mini interruptor. São eles:

- Sensor de movimento
Zigbee, IEEE 802.15.4

Alimentação a bateria: CR2450

- Sensor de abertura
Zigbee, IEEE 802.15.4
Alimentação a bateria: CR1632
- Mini interruptor
Zigbee, IEEE 802.15.4
Alimentação a bateria: CR2032

- **Smart Home Xiaomi**

Já imaginou configurar os sensores inteligentes espalhados pela sua casa através de uma única central? Como seria se as luzes se acendessem automaticamente pelo seu caminho? E se o umidificador fosse desligado quando as janelas estivessem abertas? A Xiaomi torna isso possível!

O kit de sensores inteligentes da Xiaomi, conhecido como *Smart Home* Inteligente, contém em sua composição, dois sensores de movimento, um interruptor sem fio, dois sensores de janela e porta e um hub controlador, na Figura 23, é possível visualizar o kit completo (MIBRASIL, 2020).

Figura 23 - Kit Smart Home Xiaomi.



Fonte: <https://www.mibrasil.com.br/kit-smart-home-com-sensores-inteligentes-x351-p1968>

O hub controlador contém a possibilidade de se conectar e se comunicar com todos os sensores e o interruptor, no entanto, o mesmo traz a funcionalidade de ser uma campainha e alarme, pelo fato de ter um alto falante integrado. Para o usuário acioná-lo, basta inseri-lo a uma tomada e pareá-lo ou conectá-lo ao aplicativo Mi Home. Os sensores de movimento, atuam conforme disparadores automáticos de luz, além disso, como mecanismos de segurança, avisando o usuário sobre qualquer movimentação em sua casa ou trabalho.

Os sensores de janelas e portas, alertam se as mesmas estão abertas e preparam a iluminação para sua chegada. O usuário também pode configurar os horários de iluminação automática, com base em sua rotina. A seguir, no Quadro 7, pode-se ver as especificações técnicas do *Smart Home* Xiaomi.

Quadro 7 - Especificações técnicas Smart Home Xiaomi

Smart Home Xiaomi - Especificações técnicas	
Alimentação	100-240V AC, 50Hz/60Hz
Conexão	Redes Wifi padrão 802.11 b/g/n;
Segurança do Wifi	WPA/WPA2
Faixa de Operação Wifi	2.4GHz
Interatividade	Aplicativo Mi Home para Android e iOS

Fonte: <https://www.mibrasil.com.br/kit-smart-home-com-sensores-inteligentes-x351-p1968>

O kit de controle de automação da Xiaomi, promete ao usuário controle, comodidade e segurança para o seu dia a dia, para quem busca uma implementação simples e fácil de instalar, esse kit da Xiaomi pode facilitar a vida do usuário (MIBRASIL, 2020).

3. METODOLOGIA

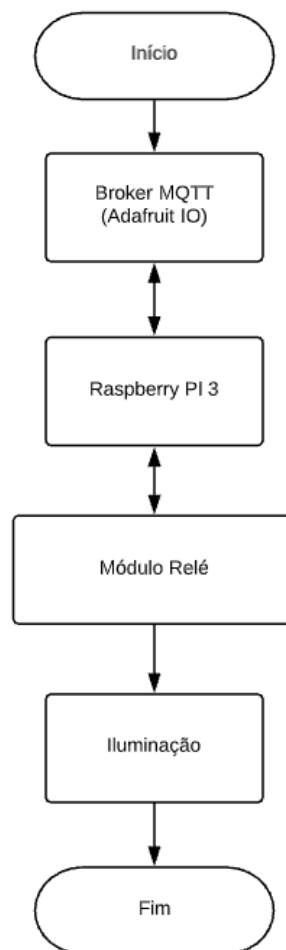
O sistema desenvolvido, foi representado através de uma maquete e, representa uma residência automatizada, com a aplicação no dia a dia em uma residência comum.

Após a breve introdução sobre o tema, destacando o progresso evolutivo da automação ao longo dos anos e, toda a importância da mesma para o futuro da humanidade, todos os componentes deste projeto foram estudados um a um, com o objetivo de juntá-los e caracterizá-los como um projeto.

Os estudos realizados e citados anteriormente, oferecem o conhecimento necessário para tornar este projeto ~~de automação residencial~~, possível.

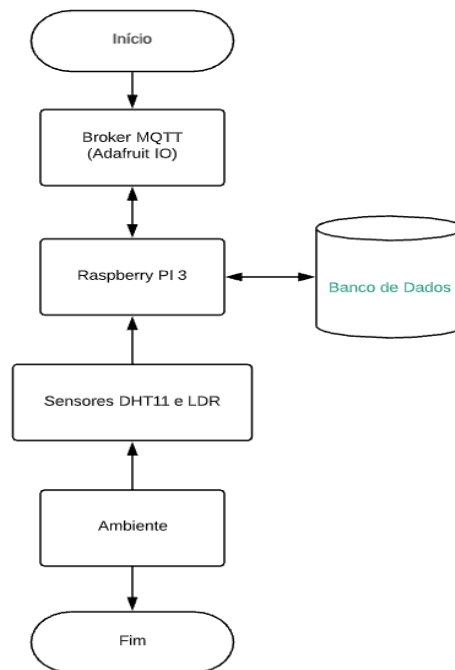
O presente documento, apresenta todo o sistema de *hardware* e *software*, representados por diagramas e esquemáticos e, toda sua implementação. A seguir, nas Figuras 24, 25 e 26, pode-se observar a visão geral dos sistemas desenvolvidos.

Figura 5 – Diagrama de Fluxo de Dados - Controle de Iluminação



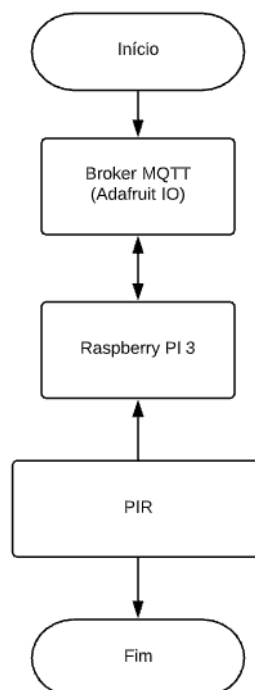
Fonte: O Autor

Figura 6 - Diagrama de Fluxo de Dados - Sensores de Dados



Fonte: O Autor

Figura 7 - Diagrama de Fluxo de Dados - Sensor PIR



Fonte: O Autor

- O sensor DHT11 recebe os dados de umidade e temperatura e envia para a Raspberry;

- O sensor LDR, assim como o DHT, recebe os dados de luminosidade do ambiente, e os envia para o Raspberry;
- O Relé, responsável por apagar e acender as luzes da maquete, recebe os comandos através da Interface WEB por intermédio da Raspberry, e executa a tarefa;
- O Raspberry recebe e processa os dados captados pelos sensores DHT e LDR, e então os envia para a INTERFACE WEB utilizando o protocolo de comunicação MQTT. Estes dados, são armazenados em um banco de dados SQLIT para a coleta de informações;
- O RASPBERRY recebe e processa os dados captados pelo módulo relé, e então os envia para a INTERFACE WEB utilizando o protocolo de comunicação MQTT. Também, recebe comandos vindos da INTERFACE WEB e de acordo com esses comandos, as luzes da maquete acendem e apagam;
- A INTERFACE WEB tem o papel de comunicar os comandos feitos, através do protocolo MQTT entre o usuário e o módulo relé, também, tem a atribuição de transmitir para o RASPBERRY o comando selecionado pelo usuário, por meio dos botões criados na *dashboard* do ADAFRUIT IO, ON e OFF;

3.1. Requisitos da maquete e *Hardware*

Para dar início ao projeto, antes de tudo, se fez necessário escolher os materiais a serem utilizados, os quais estão listados no Quadro 8.

Quadro 8 - Lista de Materiais Maquete Residencial

Maquete Residencial		
Componentes	Tipo	Quantidade
Disjuntor	Monofásico	1
Soquete	E27	5
Lâmpada	LED	5
Fio	10mm	2
Fio	1,5mm	1

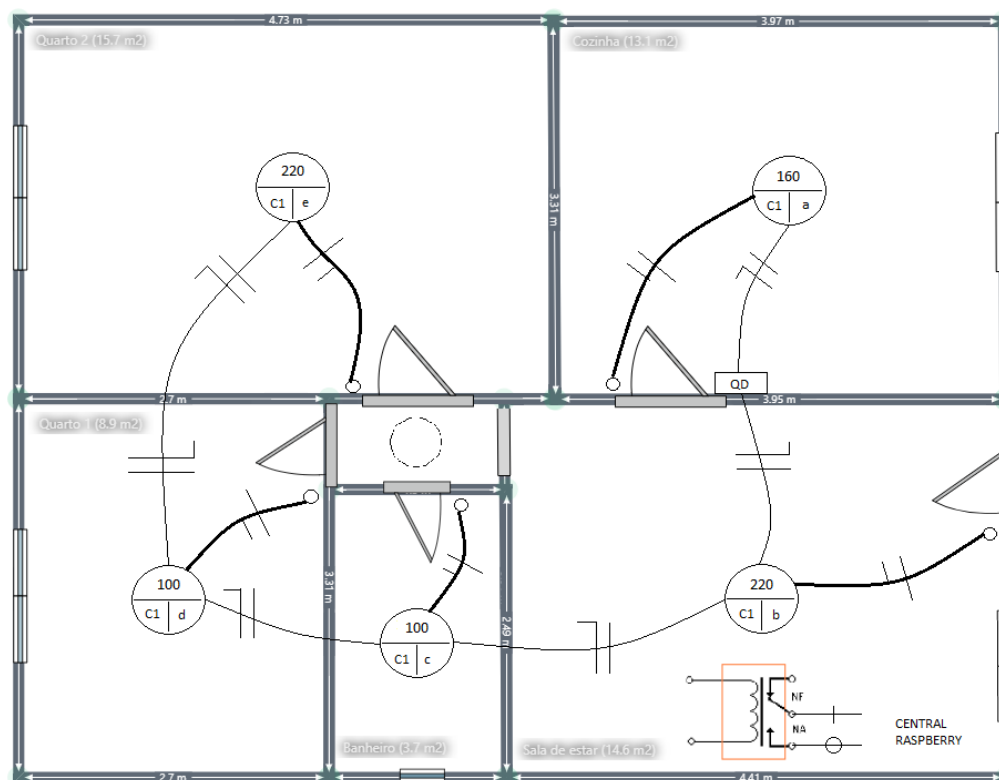
Interruptor	Simples	5
Relé	Eletromagnético	2
Jumpers	Fêmea Fêmea	9

Fonte: O Autor

A Norma Brasileira-5410 (NBR-5410), estabelece as condições ideais para o correto funcionamento de instalações elétricas de baixa tensão, sobretudo com segurança, até 1000V em tensão alternada e 1500V em tensão contínua (MUNDODAELETRICA, 2020).

Portanto, para seguir com o padrão da NBR-5410 e, dar sequência de maneira correta ao projeto, um diagrama unifilar teve de ser desenvolvido para o entendimento técnico necessário da instalação realizada na maquete. O diagrama unifilar, é ilustrado na Figura 27.

Figura 8 - Diagrama Unifilar - Maquete Residencial



Fonte: O Autor

Antes de desenvolver o diagrama unifilar do projeto, é importante que o desenvolvedor calcule a demanda de iluminação do projeto. Em resumo, o cálculo pode ser feito da seguinte forma:

- A cada 6 metros quadrados em um cômodo, deve-se contabilizar 100VA de potência, mais 60VA para os próximos 4 metros quadrados, até que as medidas totais do cômodo sejam cumpridas.

Sendo assim, no Quadro 9, é possível visualizar a demanda de iluminação do projeto. Separado por cômodo, e o total de demanda da iluminação.

Quadro 9 - Demanda de Iluminação.

Cálculo da demanda de iluminação	
Dependência	Demanda (VA)
Banheiro	100VA
Cozinha	160VA
Quarto 1	100VA
Quarto 2	220VA
Sala	220VA
Total	800VA

Fonte: O Autor

As medidas de cada cômodo, apresentadas na Figura 27, foram feitas com base em dimensões de um apartamento real, devido ao fato de que, a maquete, não representa o tamanho real de uma residência. Outro importante aspecto que o desenvolvedor deve observar, é a distribuição do quadro de cargas. Isto, porque caso haja alguma manutenção na rede elétrica, por exemplo, o eletricitista irá, com antecedência saber quais são as cargas do sistema. No presente projeto, o Quadro 10 ilustra os valores do quadro de cargas.

Quadro 10 - Quadro de cargas da iluminação.

Circuito	Descrição	Esquema	Método de Inst.	Tensão (V)	Pot. Total (VA)	Pot. Total (W)
C1	Iluminação	F+N	B1	127V	800VA	800VA

Fonte: O Autor

A iluminação residencial, representada pela maquete, contém os componentes citados no Quadro 8. Cada componente tem sua função sendo elas:

- Disjuntor – Como um protetor da rede elétrica, o mesmo protege a rede em caso de sobrecarga ou curto-circuito, se autodesarmando para evitar danos maiores;
- Soquete – Encaixe para a lâmpada, a mesma fica presa nesse equipamento para receber a tensão adequada;
- Lâmpada de LED – Peça fundamental na iluminação do ambiente;
- Fio 10mm – Utilizado para a rede elétrica da iluminação por conta do seu diâmetro, o mesmo é recomendado utilizar para correntes ligeiramente elevadas;
- Fio 1,5mm – Utilizado para a distribuição elétrica nos interruptores, igualmente ao fio de 10mm, é recomendado o uso em interruptores por conta de seu diâmetro;
- Interruptor —Acionamento das lâmpadas;

- Jumpers – Conectar a central Raspberry com o relé;
- Relé - Interruptor eletromecânico que irá fazer o acionamento ou interrupção do circuito por meio da unidade controladora (Raspberry), ele ligar.

Para que o módulo relé consiga se comunicar com a Raspberry, o relé deve ser conectado com os pinos GPIO do mesmo, conforme Figura 30. Estes pinos, são programados para enviar ou receber dados por meio de um código de programação e, no caso deste projeto, o código utilizado é o Python. Sobretudo, neste caso, estes pinos são utilizados para enviar os comandos ao módulo relé.

A seguir, no Quadro 11, pode-se observar as conexões feitas entre o Raspberry e o módulo relé, para realizar o controle da iluminação da maquete.

Quadro 11 – Relação dos terminais de GPIO e módulo Relé.

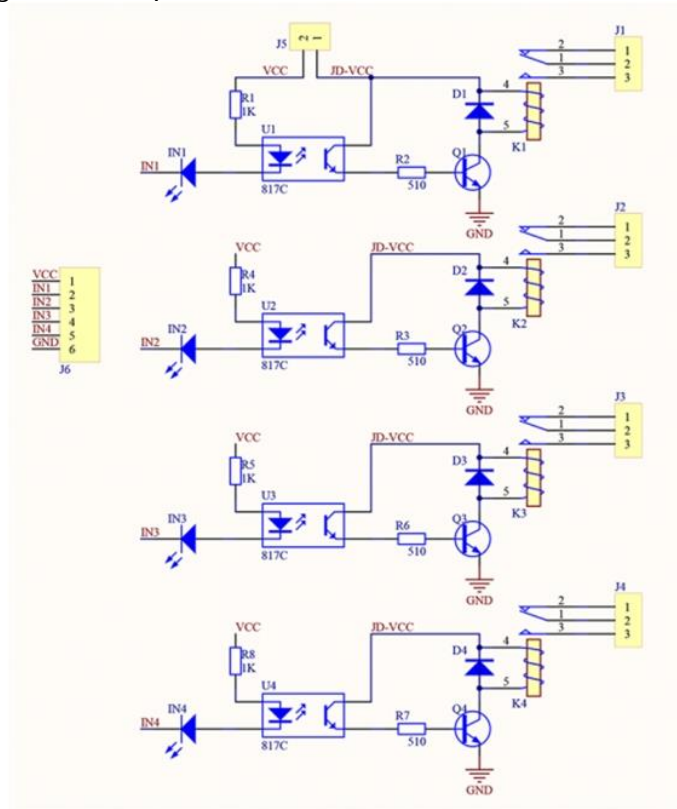
NÚMERO DO GPIO	RASPBERRY (TERMINAL)	MÓDULOS RELÉ (TERMINAL)
4	7	IN1
15	10	IN2
2	3	IN3
3	5	IN4
18	12	IN1
GND	6	GND
5V	2	VCC
5V	4	VCC
GND	14	GND

Fonte: O Autor

Ambos os dispositivos que são o módulo relé e o Raspberry pi 3, possuem circuitos internos interessantes. O módulo relé por exemplo, possui um diodo de proteção em paralelo com uma bobina, conectado a um transistor com a função de um transistor amplificador em emissor comum. Este transistor é conectado a um opto acoplador que, possui a função de proteger algumas partes do circuito de possíveis descargas elétricas, visto que, o módulo relé, opera com baixa corrente, no entanto em conexão com equipamentos de alta tensão.

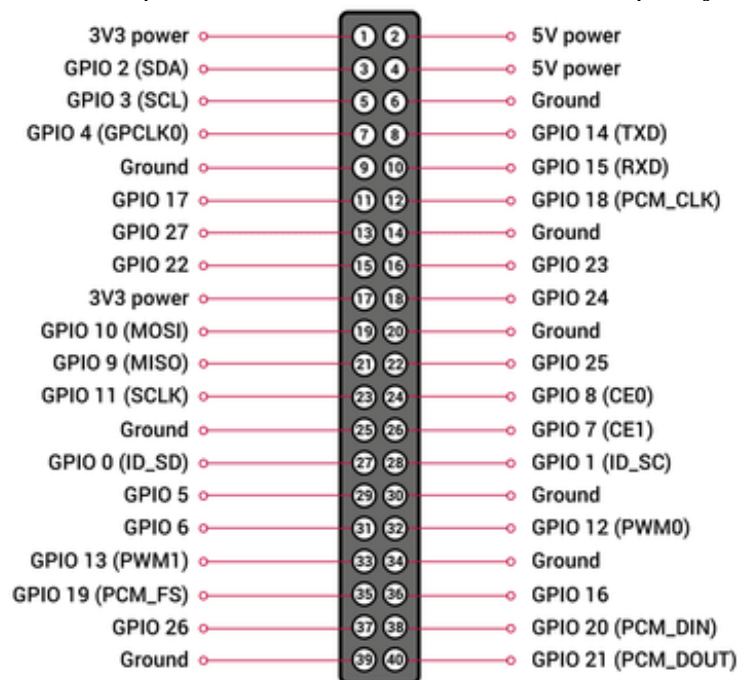
Conforme a seguir, nas Figuras 28 e 29, pode-se observar os circuitos internos do Raspberry pi 3 e do módulo relé de 4 canais utilizado neste projeto.

Figura 9 – Esquemático do circuito do módulo Relé



Fonte: <https://www.robtronics.com.br/eaduino/2020/08/4-relay-module/>

Figura 10 - Esquemático do circuito de GPIO do Raspberry



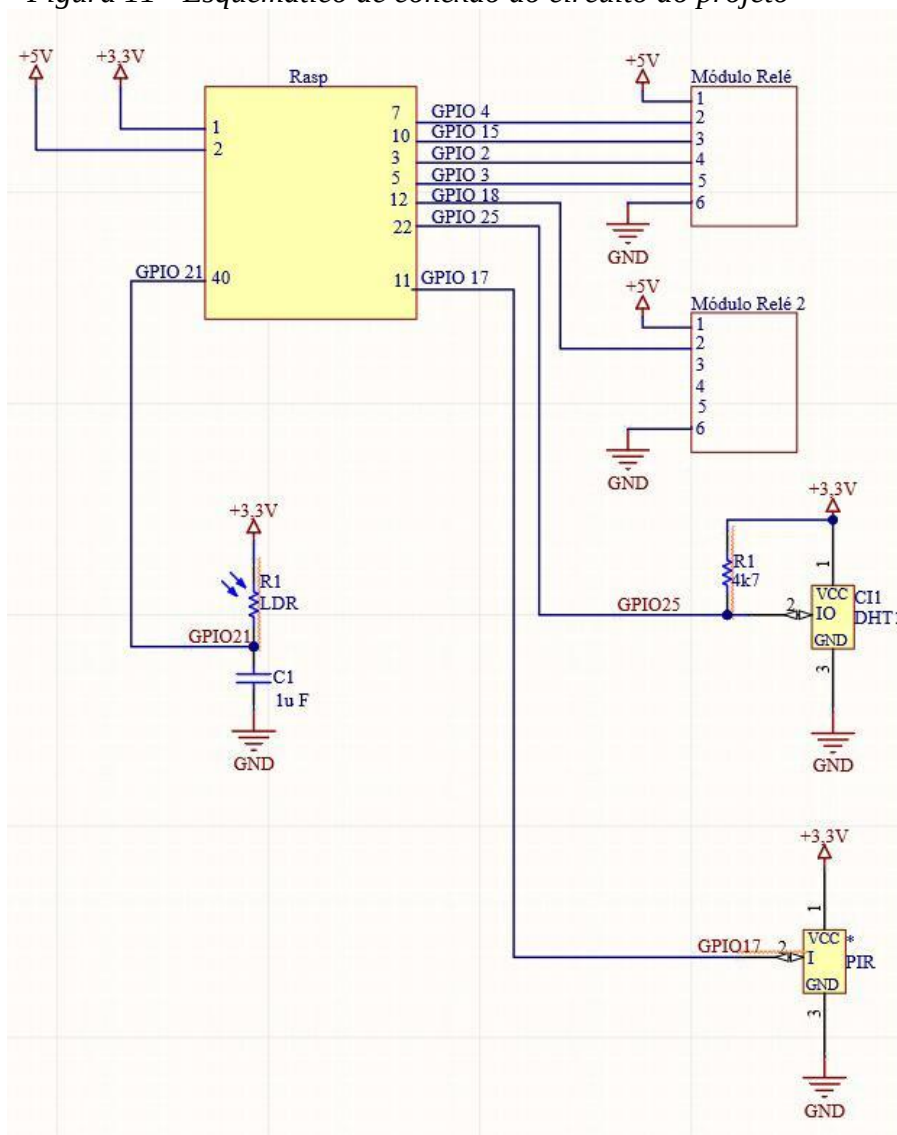
Fonte: <https://www.dz-techs.com/pt/beginners-guide-to-raspberry-pi-breadboarding-with-simon>

Conforme a Figura 30, o módulo relé 1 é alimentado através da conexão feita com a GPIO(VCC) do Raspberry, no terminal (VCC) do módulo relé. Em seguida, conecta-se a GPIO(GND) do Raspberry no terminal (GND) do módulo relé para a proteção do circuito.

Para ativar cada canal do módulo relé 1, composto por 4 canais, deve-se conectar os terminais (IN1, IN2, IN3 e IN4) nas respectivas GPIO definidas, no algoritmo desenvolvido, presente internamente no Raspberry. Cada GPIO escolhida, representa uma lâmpada a ser controlada pelos canais do módulo relé.

Ainda, em relação à Figura 30, representando o esquemático de conexão do módulo relé 2 com o Raspberry, pode-se considerar a mesma explicação em relação as conexões, referente às conexões com o módulo relé 1 e, como ferramenta de auxílio o Quadro 11.

Figura 11 - Esquemático de conexão do circuito do projeto

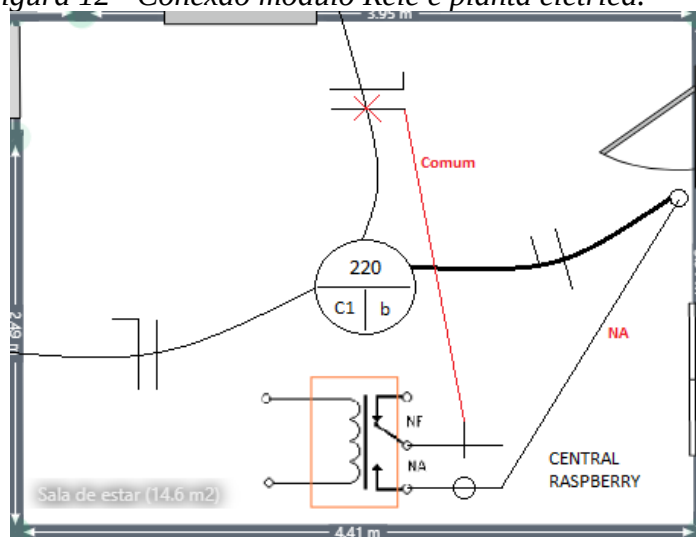


Fonte: O Autor

Deve-se lembrar que, a corrente induzida para ativar as lâmpadas da maquete, ocorre sob uma tensão de 127 V, então, o condutor fase ligado diretamente no interruptor simples, conforme Figura 27 é interrompido, e ligado no terminal comum do relé como ilustra a Figura 31.

Sendo assim, o espaço que, logicamente ficou vago no interruptor após a interrupção do condutor fase até o mesmo (Figura 32), é também ligado ao terminal normalmente aberto (NA) do relé, de modo que, quando o relé for acionado, estes dois trechos sejam interligados.

Figura 12 - Conexão módulo Relé e planta elétrica.



Fonte: O Autor

O próximo passo, foi realizar o monitoramento de umidade e temperatura do ambiente, monitorar a intensidade da iluminação e também, monitorar a movimentação de pessoas dentro da residência. Para isso, se fez necessário criar um circuito em uma *Printed Circuit Board* (PCB), com o intuito de realizar as tarefas citadas no parágrafo anterior.

De acordo com a ilustração da Figura 30, os sensores se encontram no mesmo circuito e estão alimentados pela mesma fonte, os pinos de DADOS e GND são diferentes. No Quadro 18, é possível verificar as conexões entre os terminais da PCB e do Raspberry. Contudo, o terminal de DADOS do DHT11 é conectado a GPIO 25 do Raspberry, e, o terminal de DADOS do LDR é conectado a GPIO 40 do Raspberry.

Os circuitos implementados com todos os sensores, podem ser vistos na Figura 46, na seção de resultados e discussões contendo, o sensor PIR que está na mesma PCB, entretanto, separado do outro circuito com os sensores DHT11 e LDR. O componente se torna visível na placa PCB, conforme Figura 46.

Para um melhor entendimento de como é feita a conexão dos sensores, pode-se observar e utilizar o Quadro 12 como ferramenta de auxílio.

Quadro 12 - Relação de conexão dos sensores e Raspberry.

DHT11	NºGPIO	RASPBERRY (TERMINAL)	LDR	NºGPIO	RASPBERRY (TERMINAL)	PIR	NºGPIO	RASPBERRY (TERMINAL)
VCC	3.3V	1	VCC	3.3V	1	VCC	5V	2
GND	GND	6	GND	GND	9	GND	GND	14
DADOS	25	22	DADOS	40	21	DADOS	17	11

Fonte: O Autor

3.2. Desenvolvimento do Software

Neste subcapítulo, são abordados os temas com relação aos *softwares* implementados no projeto, itens que ilustram como foi feito o controle de iluminação, o processamento dos dados e as premissas funcionais e não funcionais para o funcionamento do mesmo.

3.2.1. Premissas de Funcionalidade

Nesta seção, são encontradas as premissas de funcionalidade do projeto. Sobretudo, pontos importantes para o correto funcionamento do mesmo e, de como o usuário pode interagir com o projeto como num todo.

Dos Quadros 13 ao 18, é possível verificar as funcionalidades quanto aos comandos de funcionamento interno do sistema, como o usuário pode interagir com o sistema de automação desenvolvido e também, como o software utilizado reage aos pedidos de solicitação do usuário.

Falando dos Quadros 19 ao 22, é possível verificar as premissas não funcionais, onde pode-se observar as principais bibliotecas e links utilizados no desenvolvimento do projeto.

Quadro 13 - Envio dos comandos ao Relé

Quadro 15 – Exemplo dos comandos do RUC		
Descrição	O usuário seleciona se vai ligar ou desligar a lâmpada	
Definições de utilização		
UC01	Título	Ligar ou Desligar Lâmpada de algum cômodo
	Fluxo Principal	O usuário seleciona por meio dos botões (ON e OFF) se quer a lâmpada ligada ou desligada. ON - Liga a Lâmpada; OFF - Desliga a Lâmpada.
	Fluxo Alternativo	Não aciona os botões.

Fonte: O Autor

Quadro 14 - Ligando e Desligando as Lâmpadas

Quadro 14 – Ligando e Desligando as Lâmpadas		
Descrição	Envio dos comandos para os módulos Relé	
Definições de utilização		
UC02	Título	Envio dos comandos para os módulos Relé
	Fluxo Principal	Software envia os comandos para ligar ou desligar as lâmpadas de acordo com a escolha do usuário
	Fluxo Alternativo	Não envia os comandos.

Fonte: O Autor

Quadro 15 - Interação do usuário

Quadro 15 – Interação do usuário		
Descrição	Interação do usuário com a maquete	
Definições		
UC03	Título	Ligando e Desligando as Lâmpadas
	Fluxo Principal	Opções de interação com a maquete; ON; OFF;
	Fluxo Alternativo	Não ligar e não desligar as Lâmpadas.

Fonte: O Autor

Quadro 16 - Interação do usuário com o Sensor DHT11

Quadro 16 – Interação do usuário com o Sensor DHT11		
Descrição	Interação do usuário com o DHT11	
Definições		
UC04	Título	Recepção de Dados do DHT11
	Fluxo Principal	Usuário visualiza os dados coletados pelo DHT11, no <i>dashboard</i> .
	Fluxo Alternativo	Não coleta os dados.

Fonte: O Autor

Quadro 17 - Interação do usuário com o Sensor LDR

Quadro 17 – Interação do usuário com o Sensor LDR		
Descrição	Interação do usuário com o LDR	
Definições		
UC05	Título	Recepção de Dados do LDR
	Fluxo Principal	Usuário visualiza os dados coletados pelo LDR, no <i>dashboard</i> .
	Fluxo Alternativo	Não coleta os dados.

Fonte: O Autor

Quadro 18 - Interação do usuário com o Sensor PIR

Quadro 18 – Interação do usuário com o Sensor PIR		
Descrição	Interação do usuário com o PIR	
Definições		
UC06	Título	Recepção de Dados do PIR
	Fluxo Principal	Usuário visualiza as movimentações dentro da residência de acordo com os dados coletados pelo PIR, no <i>dashboard</i> .
	Fluxo Alternativo	Não coleta os dados.

Fonte: O Autor

3.2.2. Premissas não funcionais

Quadro 19 - Biblioteca de configuração das GPIOs

Título	Biblioteca para utilização da GPIO
Descrição	(pip3 install RPI.GPIO)

Fonte: O Autor

Quadro 20 - Biblioteca de configuração Python 3

Título	Biblioteca de cliente Adafruit IO Python
Descrição	(pip3 install adafruit-io)

Fonte: O Autor

Quadro 21 - Link de atualização dos repositórios mais recentes do Adafruit

Título	Repositório Adafruit / io-client-python
Descrição	(git clone https://github.com/adafruit/io-client-python.git)

Fonte: O Autor

Quadro 22 - Biblioteca para implementar o sensor DHT11

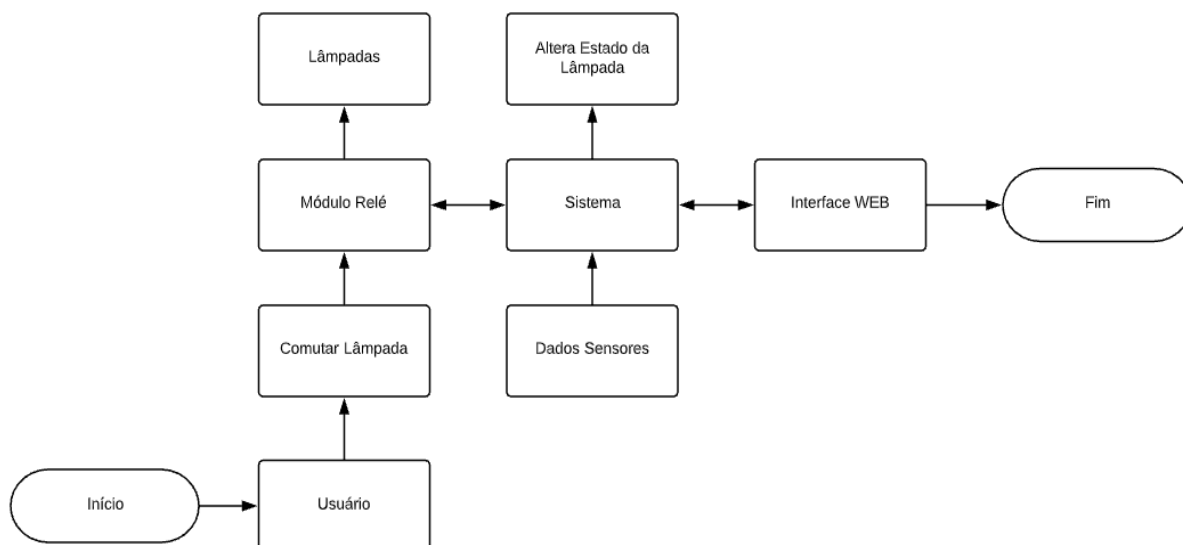
Título	Biblioteca para utilizar o DHT11
Descrição	(https://github.com/adafruit/Adafruit_Python_DHT.git)

Fonte: O Autor

3.2.3. Caso de Uso

Na Figura 32, é apresentado o fluxograma de dados para ilustrar a interação do usuário com a maquete e com os circuitos dos sensores.

Figura 13 – Diagrama de Fluxo de Dados



Fonte: O Autor

3.2.4. Interface WEB

A interface WEB, foi desenvolvida através do serviço em nuvem da *ADAFRUIT IO*. Para criar o *dashboard*, primeiramente foi necessário criar uma conta no (io.adafruit.com) e se cadastrar na plataforma.

Depois de criar uma conta na plataforma *ADAFRUIT IO*, alguns passos foram seguidos para criar o *dashboard*:

- Clicar no botão “Actions”
- Clicar no botão “Create a new dashboard”
- Definir o Nome da *dashboard*.

Depois de realizar os passos acima, foram escolhidos os *Widgets* para compor o *dashboard*. É possível visualizar, por meio da Figura 33, que os blocos demonstram que as lâmpadas estão desligadas e, se pressionar qualquer um dos botões, as lâmpadas do cômodo desejado ligam conforme Figura 34.

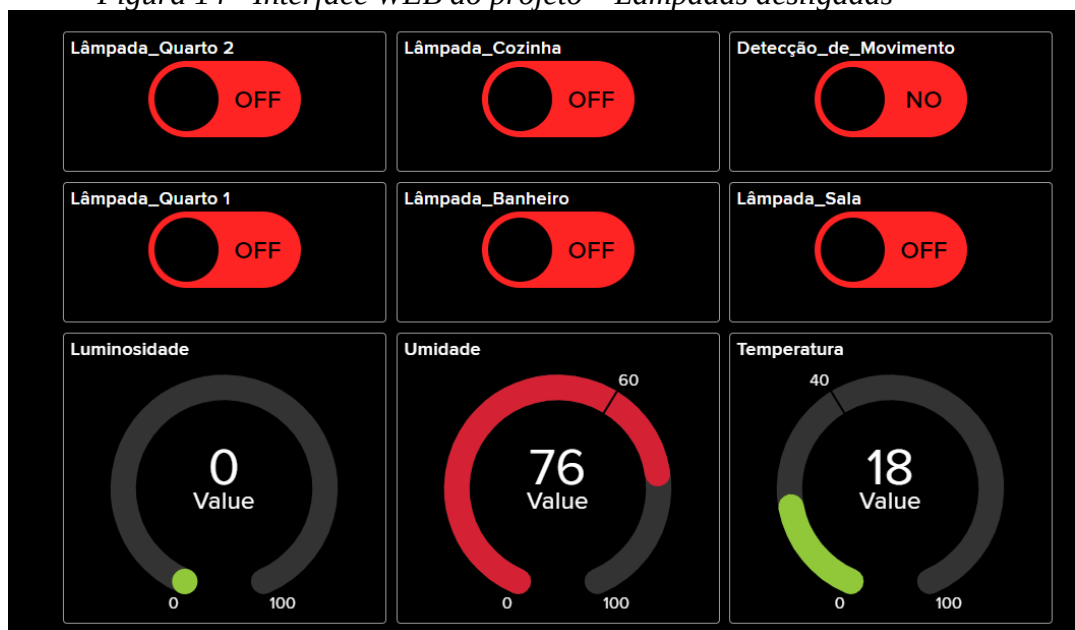
Ainda falando das Figuras 33 e 34, é notável mais 4 *Widgets* que são eles:

- *Deteção_de_Movimento*: A Figura 33 ilustra um momento sem movimentação e a Figura 34, ilustra um momento com movimentação. Entretanto, esse botão se movimenta automaticamente, de acordo com a leitura do sensor PIR. Como é feita essa

leitura, ficará mais claro em um fluxograma mais adiante;

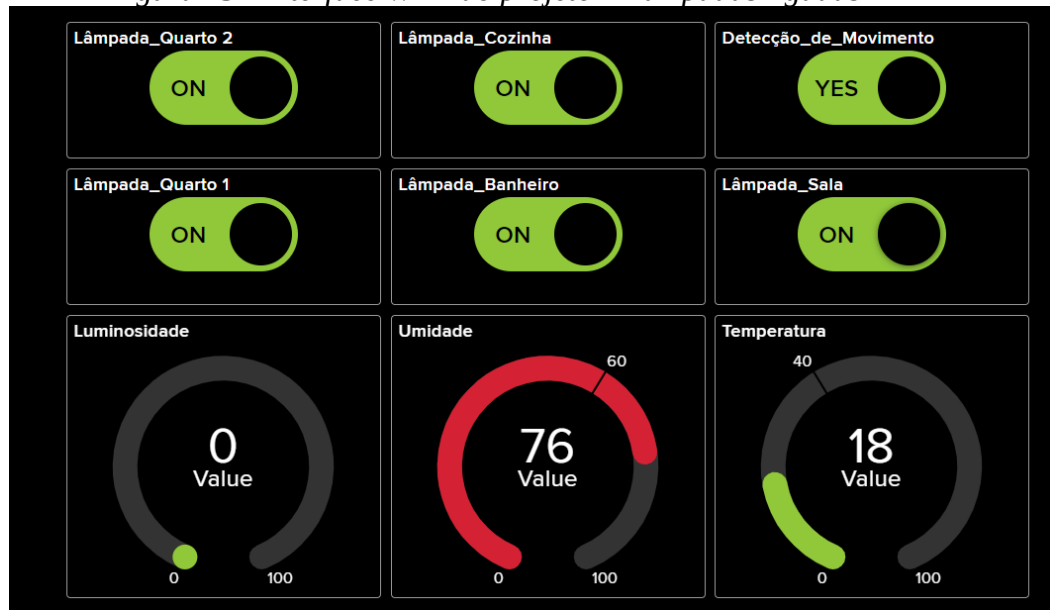
- Luminosidade: Demonstra a intensidade da Luz, através do sensor LDR. Em ambas as Figuras 33 e 34, a luminosidade apresentada é igual a zero, demonstrando que não há luz nenhuma no ambiente. Será possível verificar, através de um banco de dados, os momentos em que houveram variações de luminosidade;
- Umidade: Semelhante ao sensor de luminosidade, o mesmo altera de acordo com a umidade do ambiente, também, altera os valores na *dashboard* automaticamente, de acordo com a coleta de dados do DHT11;
- Temperatura: Em conjunto com os dados de umidade, o mesmo altera de acordo com a temperatura do ambiente, assim como o anterior, altera os valores na *dashboard* automaticamente, de acordo com a coleta de dados do DHT11.

Figura 14 - Interface WEB do projeto – Lâmpadas desligadas



Fonte: O Autor

Figura 15 - Interface WEB do projeto – Lâmpadas ligadas



Fonte: O Autor

Para cada lâmpada da maquete e para cada sensor do projeto, um *feed* teve de ser criado conforme mostra a Figura 35.

Figura 16 - Feeds utilizados no projeto

Default			
Feed Name	Last value	Recorded	
☐ GPIO15	ON	about 4 hours	
☐ GPIO18	ON	about 4 hours	
☐ GPIO2	ON	about 4 hours	
☐ GPIO3	ON	about 4 hours	
☐ GPIO4	ON	about 4 hours	
☐ tcc1_temperatura	18.0	5 days	
☐ tcc1_umidade	76.0	5 days	
☐ tcc_ldr	0	about 2 hours	
☒ tcc_presenca	YES	about 4 hours	
Enter new feed name Create			

Fonte: O Autor

Após a criação e configuração do *feed*, é só clicar em “Create Block” e o *feed* estará criado, conforme Figura 38. Desta forma, o *dashboard* foi finalizado, conforme as Figuras 35 e 36.

Figura 17 - Tela de configuração dos Feeds

Block settings ✕

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title (optional)

Block Preview



Button On Text

Button Off Text

Toggle A toggle button is useful if you have an ON or OFF type of state. You can configure what values are sent on press and release.

Test Value

Published Value

0 bytes

Fonte: O Autor

Para que a Interface WEB desenvolvida tenha, a possibilidade de exibir os dados dos sensores e, comunicar ao sistema o ato de ligar ou desligar uma das lâmpadas, determinados passos devem ser seguidos.

Depois de criar a conta no Adafruit IO, uma chave chamada “AIO” é gerada em paralelo, no perfil de desenvolvimento do *dashboard*. Ao executar o script de cada código responsável pelas tarefas, essa chave é inserida nos códigos através de um *feed*, bem como os demais componentes do projeto.

Essas chaves são exclusivas de cada desenvolvedor, portanto, a chave utilizada não pode ser compartilhada. Sobretudo, essas chaves contêm um usuário e senha, que; são criadas para o Adafruit ter acesso a sua central de controle, onde, nesse caso, é o Raspberry.

As chaves foram inseridas nos algoritmos utilizados no projeto, criados na linguagem de programação Python. As chaves são inseridas no mesmo, da seguinte forma:

- `ADAFRUIT_IO_KEY = 'CHAVE_AIO_KEY'`
- `ADAFRUIT_IO_USERNAME = 'USUÁRIO_AIO_USERNAME'`

Como pode ser visto acima, a chave criada no *Adafruit IO* para a Interface WEB ter acesso ao Raspberry, é inserida no início e no meio do algoritmo quando foi criado um *feed* específico para tal e

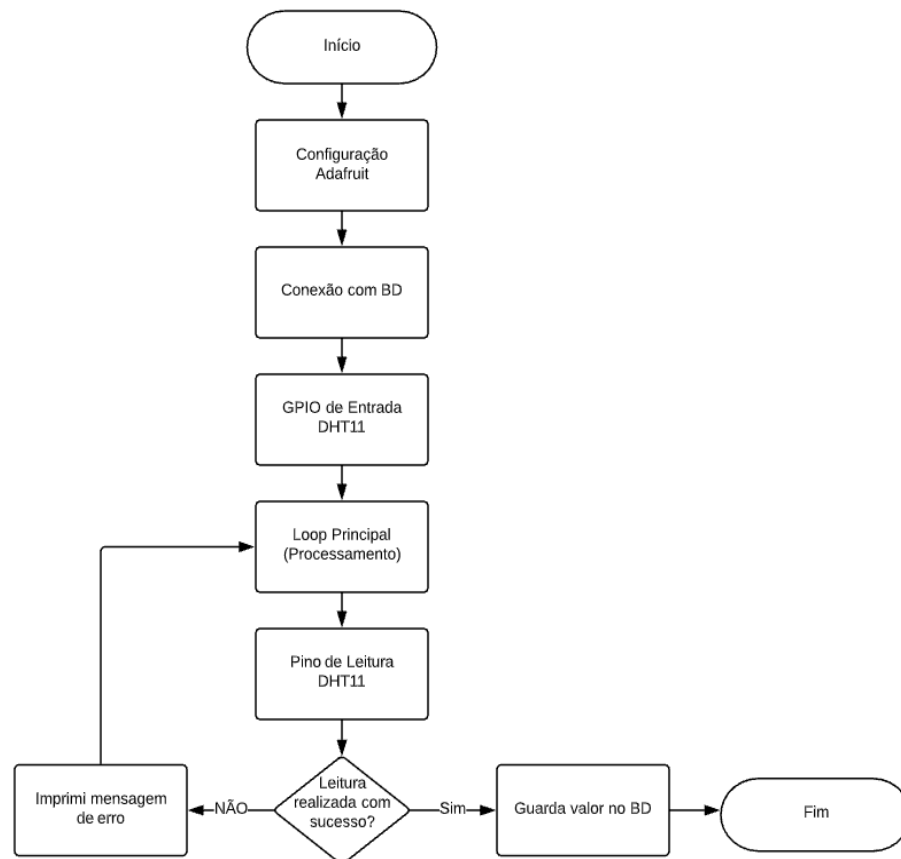
também, quando a instância do protocolo MQTT é inserida. Desta maneira, a Interface WEB teve acesso ao Raspberry e ao algoritmo desenvolvido.

3.2.5. Utilização do sensor DHT11

Para utilizar o DHT11 no projeto, primeiramente foi necessário implementar o mesmo em uma protoboard para testar o componente.

Depois de realizar os testes e verificar o correto funcionamento do mesmo, o sensor foi implantado em uma PCB. Após a implantação do DHT11 na PCB, o script teve de ser executado internamente na Raspberry, através do código criado em Python.

Figura 18 - Fluxograma de funcionamento do algoritmo do DHT11



Fonte: O Autor

Depois de implantar o DHT11 na PCB, um algoritmo foi desenvolvido para realizar a leitura dos dados coletados pelo DHT11: umidade e temperatura. O código criado, inicia com a configuração do Adafruit e sincronizando a Interface WEB desenvolvida, em conjunto com o protocolo MQTT. Agora, o código permite a conexão do DHT11 com o banco de dados criado na versão SQLITE para

armazenar os dados coletados pelo sensor. Por fim, o código realiza as configurações das GPIOs de entrada do DHT11 para a leitura do mesmo pelo Raspberry, conforme ilustra a Figura 37.

Daí em diante, o loop principal do código inicia o processamento de cada linha do mesmo e realiza a leitura do DHT11. Contudo, se a leitura for um sucesso, o mesmo guarda as informações coletadas no SQLITE (Figura 38), se não, volta ao loop principal para um novo processamento, até a leitura ser realizada corretamente. Para maiores detalhes, o código comentado linha a linha, estará disponível nos anexos.

Figura 19 - Banco de Dados DHT11 - SQLITE



Tabela: tb_dht							Novo registro	Deletar registro
	id	Temperatura	Umidade	Horario				
	Filtro	Filtro	Filtro	Filtro				
1	1	19.0	78.0	09/06/2021 19:41:56				
2	2	19.0	76.0	09/06/2021 21:41:57				
3	3	19.0	80.0	09/06/2021 23:41:57				
4	4	18.0	88.0	10/06/2021 01:43:32				
5	5	18.0	93.0	10/06/2021 03:48:57				
6	6	18.0	92.0	10/06/2021 05:52:52				
7	7	18.0	91.0	10/06/2021 07:56:39				

Fonte: O Autor

3.2.6. Utilização do sensor LDR

Assim como o sensor DHT11, primeiramente foi necessário implementar o mesmo em uma protoboard para testar o componente e em seguida implementá-lo em uma PCB. Após a implantação do LDR na PCB, o script teve de ser executado internamente na Raspberry, através do código criado em Python, de acordo com a Figura 39.

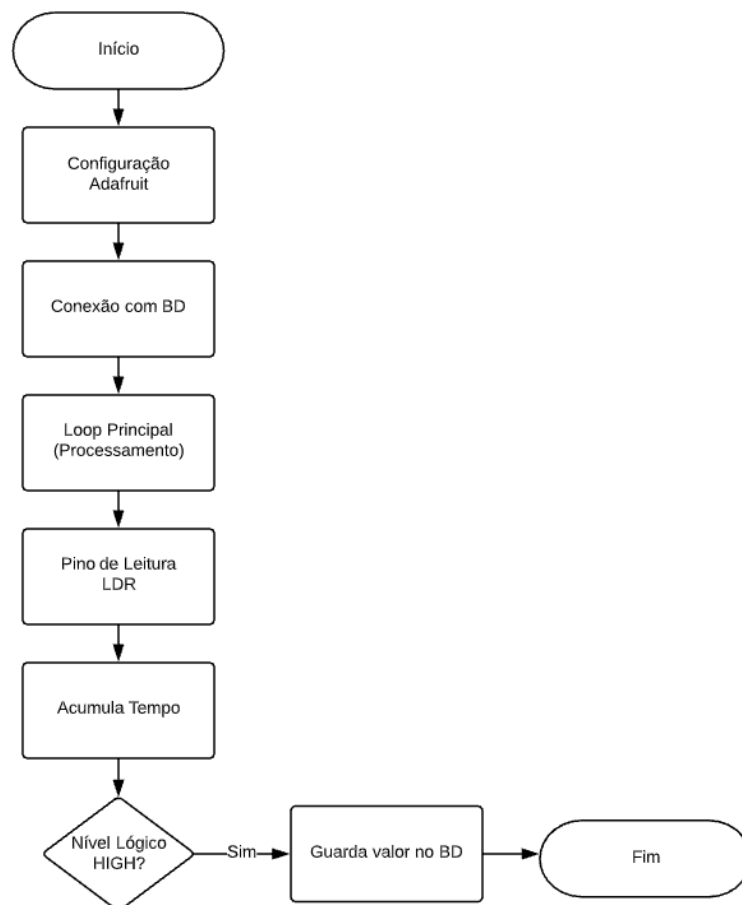
Figura 20 - Banco de Dados LDR - SQLITE

Tabela: tb_ldr Novo registro Deletar registro

	id	Valor	Horario
		Filtro	Filtro
22		99.8356008529663	20/06/2021 12:22:15
23		99.8684787750244	20/06/2021 10:22:14
24		99.7119522094727	20/06/2021 08:22:12
25		0.0	20/06/2021 06:22:05
26		0.0	20/06/2021 04:21:55
27		0.0	20/06/2021 02:21:48
28		99.6849679946899	20/06/2021 00:21:46

Fonte: O Autor

Figura 21 - Fluxograma de funcionamento de algoritmo do LDR



Fonte: O Autor

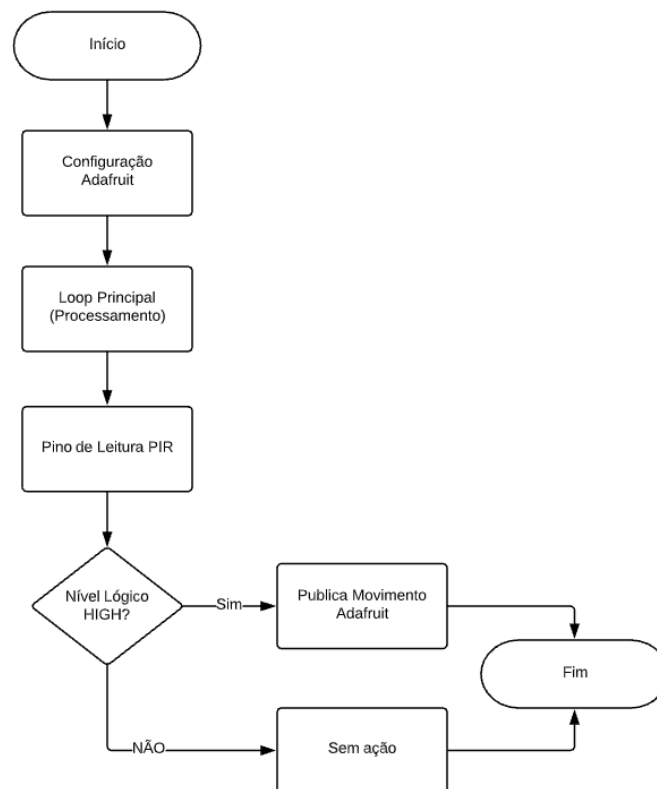
O código desenvolvido, inicia com a configuração do Adafruit e sincronizando a Interface WEB desenvolvida, em conjunto com o protocolo MQTT. Agora, o código permite a conexão do LDR com o banco de dados criado na versão SQLITE para que os dados sejam guardados, como mostram as Figuras 39 e 40. Por fim, o código realiza as configurações das GPIOs de entrada do LDR para a leitura do mesmo pelo Raspberry.

Daí em diante, o loop principal do código inicia o processamento de cada linha do mesmo e realiza a leitura do LDR. O LDR realiza a leitura da luminosidade do ambiente, o mesmo, acumula o tempo em que o LDR levou para medir determinada luminosidade. Portanto, se o valor lógico do pino for *HIGH*, a leitura sai do loop principal e guarda o valor coletado no banco de dados. Para maiores detalhes, o código estará disponível nos anexos, comentado linha a linha.

3.2.7. Utilização do sensor PIR

Assim como os sensores anteriores, foi necessária a implementação do mesmo em uma protoboard para testar o componente, e em seguida, implementá-lo em uma PCB. O script teve de ser executado internamente na Raspberry, através do código criado em Python, para o funcionamento do mesmo, conforme Figura 41.

Figura 22 - Fluxograma de funcionamento do algoritmo do PIR



Fonte: O Autor

O algoritmo desenvolvido para o sensor PIR, inicia com a configuração do Adafruit e sincronizando a Interface WEB desenvolvida, em conjunto com o protocolo MQTT. Depois, o código realiza as configurações das GPIOs de entrada do LDR para a leitura do mesmo pelo Raspberry.

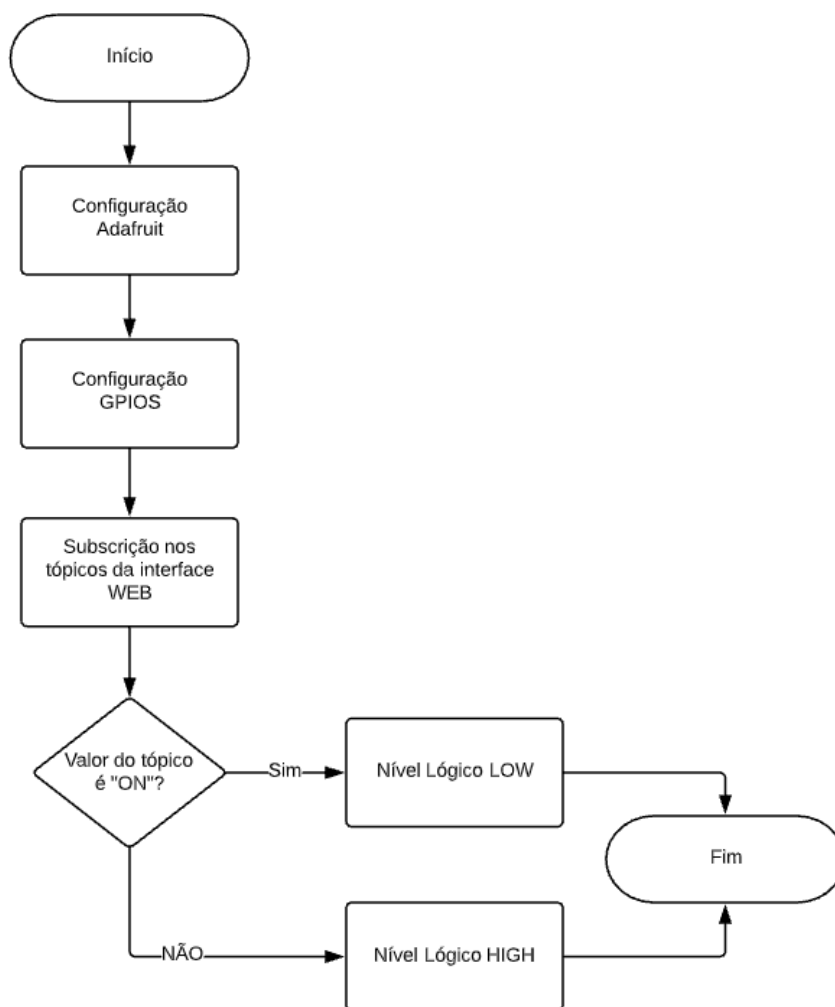
Por fim, o loop principal do código inicia o processamento de cada linha do mesmo e realiza a leitura do PIR. O pino de leitura do PIR, inicia o monitoramento do ambiente. Portanto, se o valor lógico do pino for *HIGH*, a ação “Movimento” será publicada na Interface WEB, se não, a mensagem publicada será, “Sem movimento”.

Para um maior entendimento, o código estará disponível nos anexos, comentado linha por linha.

3.2.8. Controle de Iluminação

Para executar o controle de iluminação da maquete, foi desenvolvido um código em Python, utilizando os níveis lógicos: HIGH (Alto) e LOW (Baixo) para ligar e desligar as lâmpadas.

Figura 23 - Fluxograma de funcionamento do algoritmo do controle de iluminação



Fonte: O Autor

O algoritmo criado, inicia com a configuração do Adafruit e sincronizando a Interface WEB desenvolvida, em conjunto com o protocolo MQTT. Depois, o código realiza as configurações das GPIOs de saída do relé para a leitura do mesmo pelo Raspberry. O protocolo MQTT, realiza a subscrição dos comandos em tópicos na Interface WEB, em seguida o tópico é recebido pelo Raspberry e enviado para o módulo relé.

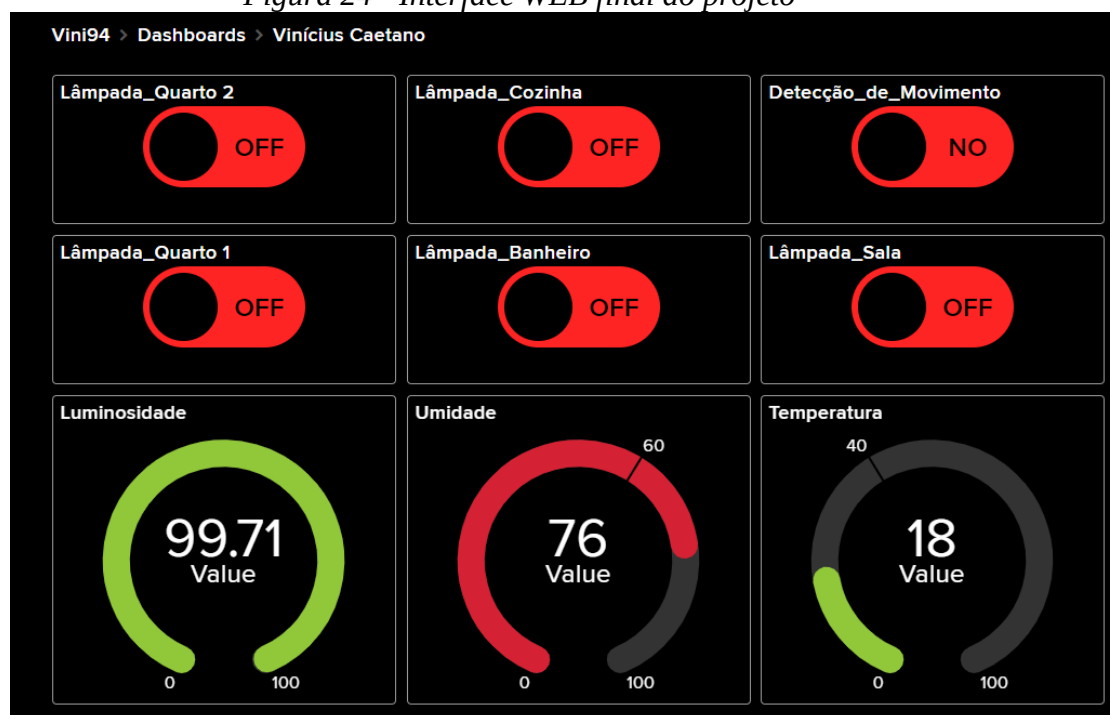
Os pinos de leitura do módulo relé, são definidos em dois estados: *HIGH* e *LOW* e, inicia o controle da iluminação da seguinte forma: Se o valor lógico do pino for *LOW*, a ação será acender a Lâmpada, se for *HIGH* a ação será, apagar a lâmpada. Na interface WEB aparecerá ao usuário como: OFF ou ON, assim como, a Figura 42 ilustra.

Para maiores detalhes, o código estará disponível nos anexos, comentado linha por linha.

4. RESULTADOS E DISCUSSÕES

Para realizar o controle de iluminação do projeto e verificar as tarefas dos sensores, foi desenvolvida uma Interface WEB por meio do serviço em nuvem da Adafruit IO, conforme Figura 43.

Figura 24 - Interface WEB final do projeto



Fonte: O Autor

Bem como citado na metodologia desta monografia, a comunicação da Interface WEB é feita por meio do protocolo MQTT. A execução do Broker MQTT, é realizada pelo próprio serviço de MQTT do Adafruit.

O Raspberry utiliza os métodos *Publish* e *Subscriber* da biblioteca MQTT, disponibilizada em Python, pela própria Adafruit.

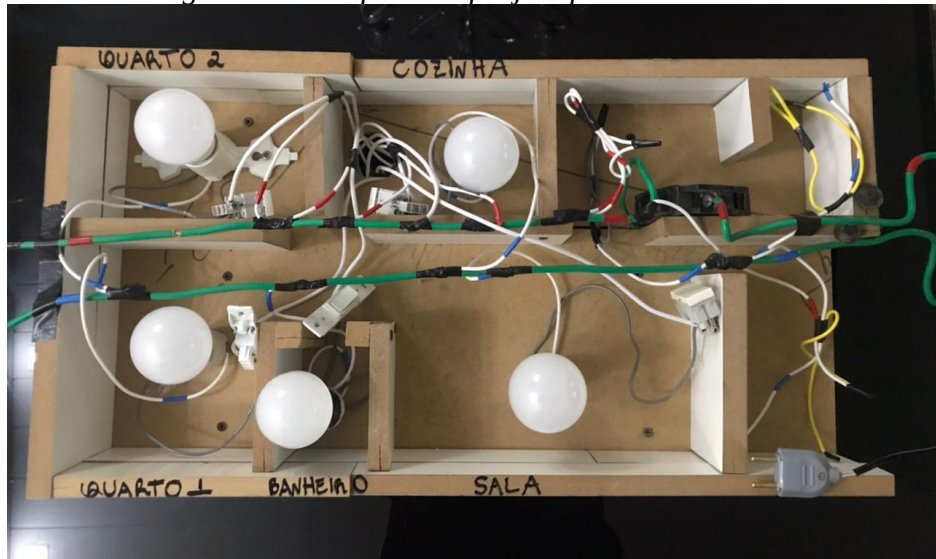
Para realizar o acionamento das lâmpadas da maquete, o módulo de acionamento da Raspberry efetua o *Subscribe* nos *feeds* criados na Interface WEB e, aguarda que uma ação seja realizada na mesma.

A interface WEB, também, realiza a comunicação com o Broker MQTT do Adafruit, nos *feeds* que foram criados na mesma.

Ao receber uma mudança de estado de um dos *feeds*, o módulo muda o estado da sua correspondente GPIO para executar o acionamento de um item da maquete.

Com o objetivo de desenvolver o projeto com um baixo custo, a maquete foi criada para representar uma residência comum. Esta maquete, contém todos os materiais citados nos Quadros 8 e, 5 cômodos ativos com lâmpadas em funcionamento, conforme Figura 44.

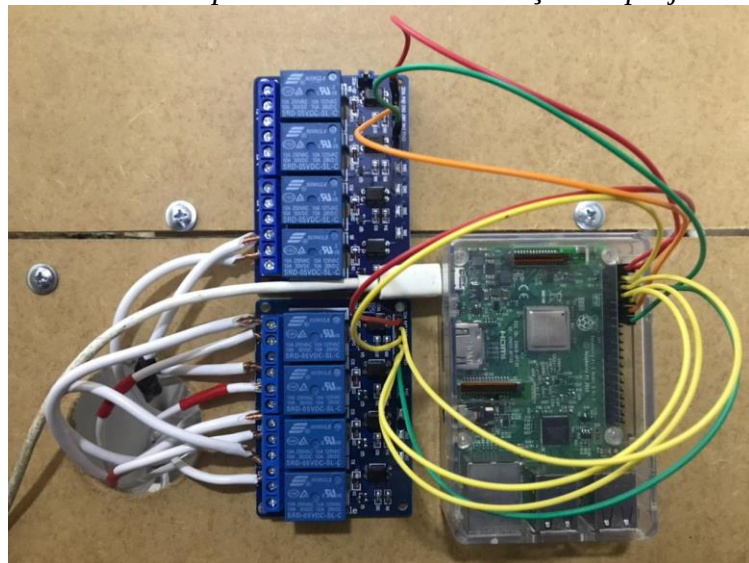
Figura 25 - Maquete do projeto finalizada



Fonte: O Autor

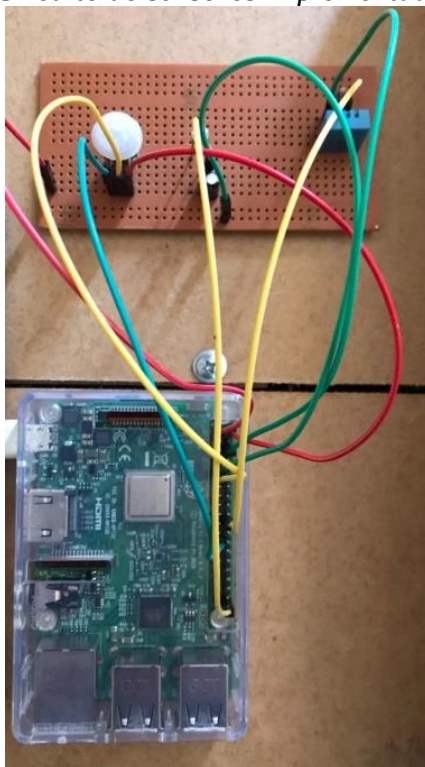
Depois de montar a maquete, foi desenvolvida a central de controle da iluminação (Figura 45). A central; tem como base o minicomputador Raspberry Pi 3 modelo B e 2 módulos relés de 4 canais cada. Portanto, conforme citado anteriormente, o Raspberry recebe os comandos para ligar e desligar as lâmpadas por meio da Interface WEB que, codifica essas informações através do código implementado na Raspberry e, as transmite para os módulos relés, onde, cada módulo de acordo com a escolha do usuário, irá desligar ou ligar uma lâmpada.

Figura 26 - Central para controle de iluminação do projeto



Fonte: O Autor

Figura 27 - Circuito de sensores implementados do projeto



Fonte: O Autor

Após concluir o desenvolvimento da maquete para representar o controle de iluminação, todo o sistema de sensores foi implementado, como mostra a Figura 46.

O sistema de sensores ilustrado na Figura 46, é o sistema final do projeto. O mesmo dispõe dos seguintes componentes, ilustrados no Quadro 23:

Quadro 23 - Quadro de informações técnicas do sistema de sensores

Sensores	Alimentação	Dados de Coleta
DHT11	3.3V	Umidade e Temperatura
LDR	3.3V	Intensidade da Luz
PIR	5V	Movimentações
PCB	N/A	N/A
RASPBERRY PI 3 - B	5V	Dados do projeto implementado

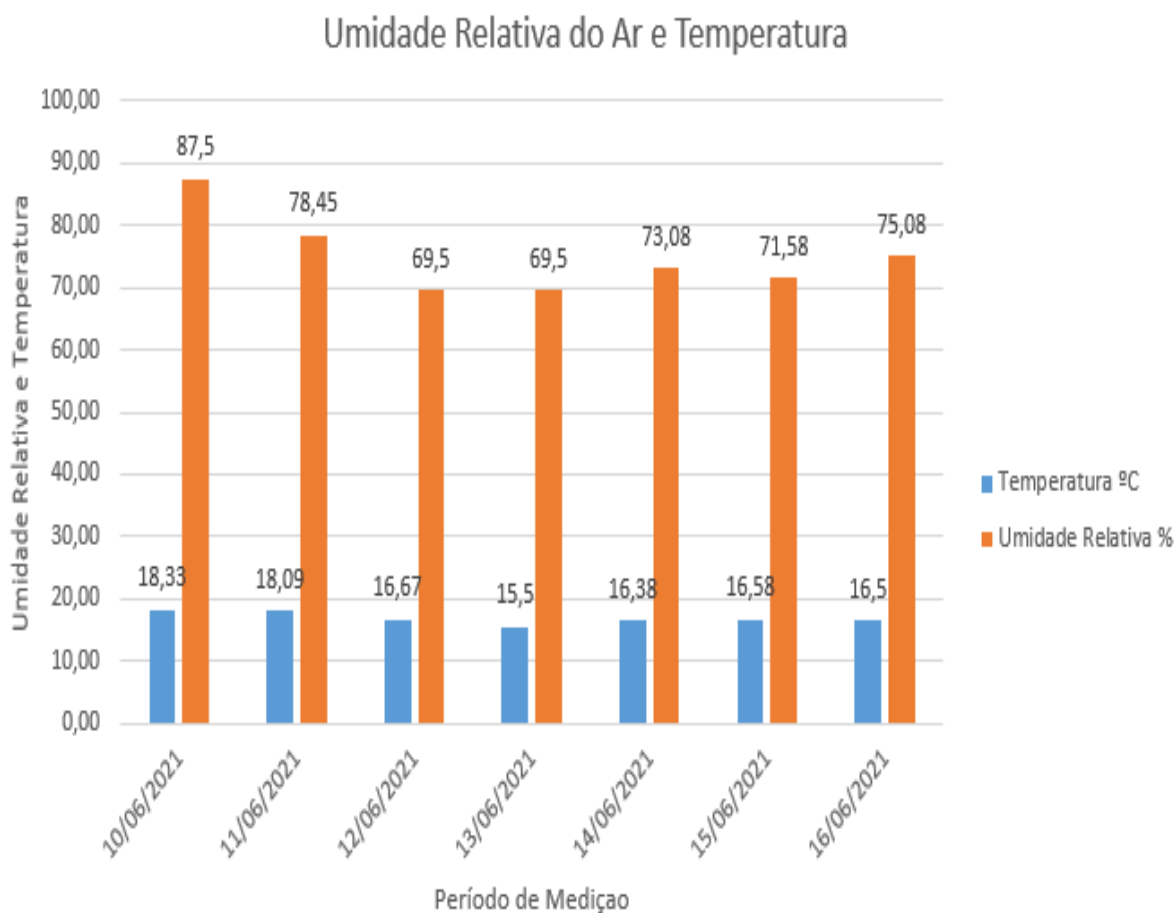
Fonte: O Autor

Depois de implementar os sensores: DHT11 e LDR, seus dados foram coletados e guardados em um banco de dados SQLITE. A tabela criada no SQLITE, foi convertida em formato de Excel para que pudesse ser ilustrada através de um gráfico.

Para visualizar os dados de umidade e temperatura coletados, pode-se observar na Figura 47, os valores obtidos. Os valores que aparecem no gráfico, são uma média tirada por dia, durante 7 dias.

A decisão de ilustrar os dados coletados por média, se deu pelo fato de que, as medições foram feitas de 2 em 2 horas. Sendo assim, para não poluir o gráfico com muitos valores, foi decidido tirar a média dos dados durante o dia, para se ter uma base dos valores coletados pelo DHT11 ao longo do experimento.

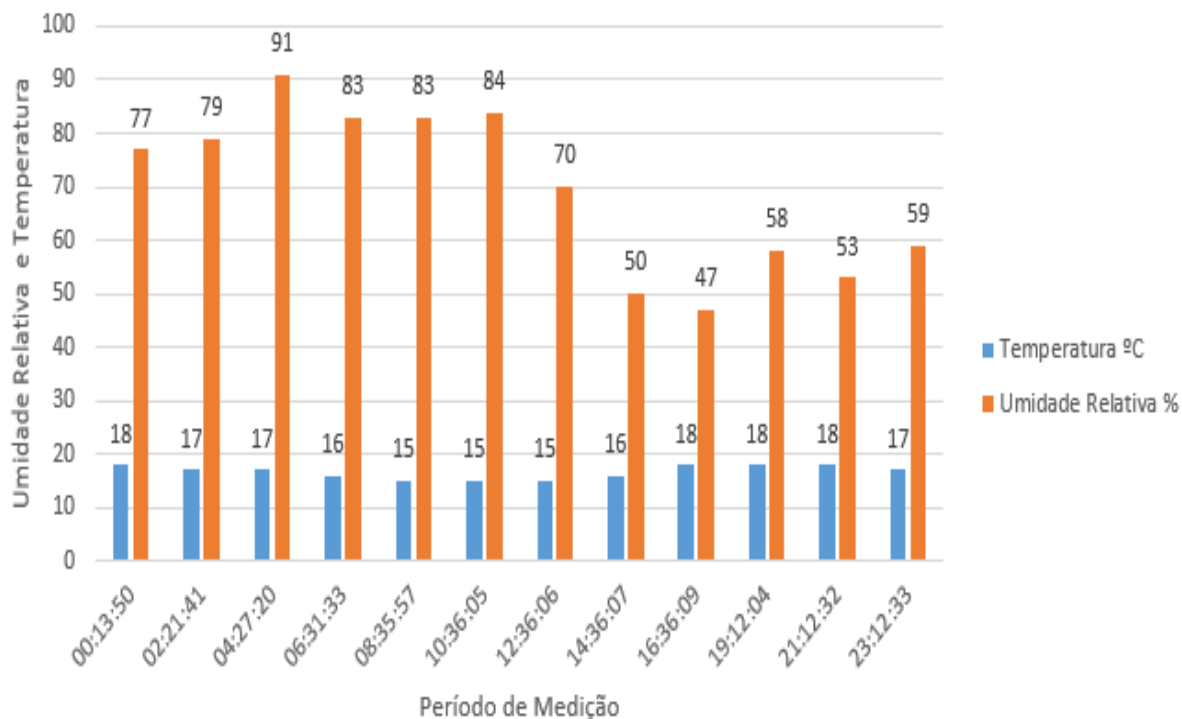
Figura 28 - Gráfico de valores coletados do DHT11



Fonte: O Autor

Para visualizar com um maior detalhe, a Figura 48, ilustra um gráfico de medições de um dia, de 2 em 2 horas e, é possível notar a variação da umidade e temperatura do ambiente, de acordo com os horários ao longo do dia.

Figura 29 - Dados coletados do DHT11 ao longo de um dia
Umidade Relativa do Ar e Temperatura - 12/06/2021



Fonte: O Autor

Com o sensor LDR, o experimento se assemelha ao DHT11. Entretanto, pelo fato de o mesmo medir a luminosidade de um ambiente, com seus valores coletados, pode-se ver de acordo com o gráfico na Figura 49, quando está escuro ou sem luz sobre o LDR, e quando está com pouca luz sobre o LDR, de acordo com os valores obtidos, conforme explicação a seguir.

Devido ao fato de o LDR realizar uma medição analógica e os pinos da Raspberry receberem dados digitais, um circuito RC foi desenvolvido para que a medição fosse feita. Com esse circuito, pode-se ver quando está claro, quando está escuro ou, sombreado. Como isso foi feito? Com o circuito do LDR devidamente montado, percebeu-se que era preciso realizar a calibragem dos dispositivos de aquisição de sinal. Uma vez que, a função final desse circuito era de fornecer informações a respeito da luminosidade do ambiente.

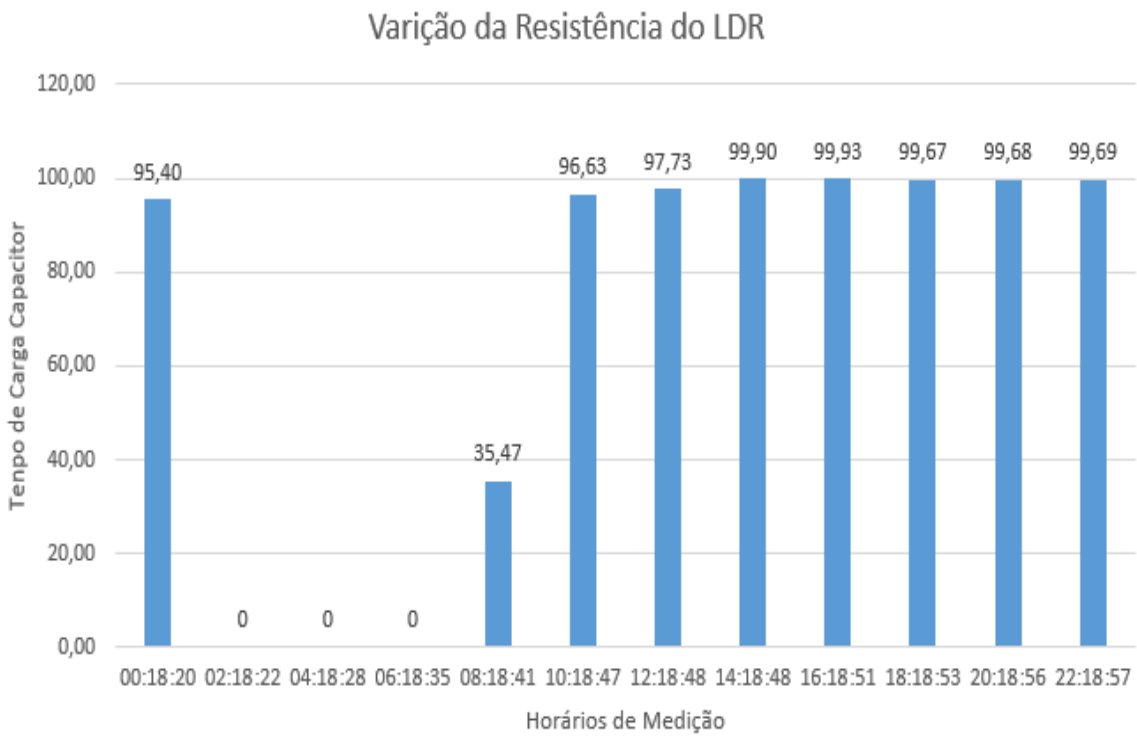
Pôde-se controlar as variáveis do local em que esse dispositivo foi montado, ajustando a exposição dos fótons em mínima e máxima. Com isso, foram realizadas medidas de calibragem e definidos os limites entre zero e cinco segundos. Em uma máxima exposição à luz, pode-se perceber que o tempo de acionamento se mostrou sempre próximo de zero.

Em seguida, por se tratar de uma medida de tempo (o tempo em que o capacitor demorava a ser carregado, devido à variação de resistência do LDR), percebeu-se que, ao privar o circuito de qualquer fonte emissora de luz, o capacitor levava, no mínimo, 5 segundos para realizar o disparo e sensibilizar

a GPIO do Raspberry PI. Fornecendo então, os subsídios necessários para se definir os limites mínimo e máximo a serem inseridos no algoritmo de conversão de tempo em luminosidade. Dada pela fórmula: $Luminosidade = (5(\text{limite máximo}) - (\text{valor de tempo lido})) * 20$ (constante multiplicadora utilizada para converter proporcionalmente o resultado da leitura em valores de 0 a 100).

O decremento do valor lido do limite máximo foi feito a partir da natureza da leitura. Buscou-se transformar o valor de tempo lido do circuito de medição em medida de luminosidade. Como quanto menor a exposição do circuito à luminosidade, maior a resistência no LDR, foi preciso inverter o dado lido para que se pudesse apresentar um dado de luminosidade e não um valor transposto de resistência.

Figura 30 - Gráfico de luminosidade do LDR



Fonte: O Autor

5. CONSIDERAÇÕES FINAIS

Neste projeto, foram apresentados conceitos de IoT e o desenvolvimento de um sistema de automação para residências. O sistema em questão, contém um Raspberry PI 3 Modelo B, onde, esse dispositivo é a central do projeto.

O Raspberry é conectado por meio de suas GPIOs a dois módulos relés de 4 canais cada e a 3 sensores com objetivos diferentes: sensor de umidade e temperatura, sensor de luminosidade e sensor de movimento. Para efetuar o controle de iluminação da maquete desenvolvida, o usuário pode fazer isso, por meio de uma interface WEB. Se o mesmo quiser verificar os dados dos sensores, pode o fazer, através da mesma interface.

Os resultados foram os esperados e demasiadamente satisfatórios. Visto que, é possível controlar o ato de apagar e ascender as luzes de uma residência representada por uma maquete, por meio de uma interface WEB. Pode-se observar em tempo real a umidade e temperatura do ambiente e em quais momentos do dia o ambiente tem mais ou menos intensidade de luz. Também, é possível verificar a presença ou movimentação de pessoas no ambiente, através do sensor de movimento.

Este projeto, foi extremamente importante para o meu desenvolvimento pessoal, acadêmico e profissional do autor. Os conhecimentos ministrados pelos professores ao longo da graduação, sendo eles, práticos e teóricos, foram fundamentais para o desenvolvimento deste projeto.

5.1.1. Dificuldades encontradas

- Desenvolvimento da maquete:
 - A criação da maquete, foi algo ligeiramente mais demorado, pelo fato da falta de experiência com eletrotécnica. Primeiro foi necessário entender alguns conceitos de eletrotécnica, depois, separar os fios corretamente para cada um estar no lugar certo e distribuir toda a fiação elétrica ao longo da maquete.
- Implementação dos Sensores
 - Uma ligeira dificuldade encontrada ao longo do projeto, foi quanto a implementação dos sensores, devido à dificuldade com programação. As aplicações ao longo do projeto, tomaram um certo tempo durante a implementação e, algumas etapas foram mais complicadas como: entender como funciona um código e como aproveitá-lo no projeto. Por exemplo, no início do

projeto, foi sugerida a não utilização de Arduino, logo, foi necessário criar um circuito RC para realizar a leitura de luminosidade do LDR, por meio do tempo de carga no capacitor e, em seguida, converter este tempo em luminosidade dentro de uma lógica criada no algoritmo. Foi algo trabalhoso pois, o Raspberry só aceita leituras digitais e não analógicas.

- Implementação do controle de iluminação
 - A dificuldade encontrada consistiu em realizar a comunicação entre a Interface WEB e o Raspberry, mas, graças ao método de *feeds* do Adafruit IO e comunicação por tópicos do broker MQTT, depois de compreender como funcionava este protocolo, foi ligeiramente mais fácil o desenvolvimento.

5.1.2. Melhorias futuras

Utilizar a aplicação do controle de umidade e temperatura para controlar um sistema de irrigação de plantas ou um ar condicionado, automaticamente.

Aprimorar o ato de coletar os dados de luminosidade do ambiente, por meio do protocolo I2C, assim, os valores de resistências coletados, seriam mais precisos.

Utilizar o sensor PIR para abertura e fechamento de portas e janelas, automaticamente, quando o sensor detectar um movimento humano.

Implementar a assistente virtual Alexa, com o objetivo de utilizar comando de voz, para controlar o sistema de iluminação da residência.

6. REFERÊNCIAS

AGUIRRE, L.A. **Fundamentos de Instrumentação**. 1ª ed. Pearson, 2014.

ALMEIDA, André Duarte. **Conhecendo o Raspberry Pi. Possibilidades de uso em contextos educacionais e profissionais**. Disponível em:<<https://docente.ifrn.edu.br/andrealmeida/palestras/ensol-conhecendo-o-raspberry-pi>> Acesso em: 10/12/2019

AURESIDE. **Previsões para o mercado global de automação residencial**. Disponível em:<<http://www.aureside.org.br/noticias/previsoes-para-o-mercado-global-de-aut-residencial>> Acesso em: 05/05/2021.

BRAGA, N. **Relés Circuitos e Aplicações**. INSTITUTO NEWTON C. BRAGA 1ª. ed. 2012.

BOLZANI, M.A.C. **Residências Inteligentes**. 1ª ed. Livraria da Física, 2004.

BRAGA, N. **LDR(ALM332)** Disponível em:<<https://www.newtoncbraga.com.br/index.php/almanaque-tecnologico/201-l/7547-ldr-alm332.html>> Acesso em: 05/05/2021>.

BALBINOT, A.; Brusamarello, V.J. **Instrumentação e Fundamentos de Medidas 2 ed**. São Paulo: LTC, 2011.

BARBOSA, S.D.J.; SILVA, B.S. **Interação Humano-Computador**. Série SBC, Editora Campus-Elsevier, 2010.

BARR, M.; MASSA, A. **Programming Embedded Systems**. O'REILLY. 2006

BASÍLIO, S. **Conhecendo o Protocolo MQTT**. Disponível em:<<https://blogmasterwalkershop.com.br/outros/conhecendo-o-protocolo-mqtt>> Acesso em: 28/05/2021.

COUTINHO, T. **O que é Python e pra que serve?** Disponível em:<<https://www.voitto.com.br/blog/artigo/python>> Acesso em: 01/06/2021.

CDN-SHOP.ADAFRUIT.COM. **datasheet**. Disponível em:<<https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf>> Acesso em: 20/10/2020.

FILIPPE.F **Guia Raspberry PI para iniciantes**. Disponível em:<https://www.if.ufrj.br/~gpenello/Lab2_2018-1/Guia_Raspberry_Pi_para_Iniciantes_-_FilipeFlop.pdf> Acesso em: 28/10/2019>.

FERNANDES, M. H. **O que é Python e pra que serve?** Disponível em:<<https://marquesfernandes.com/desenvolvimento/o-que-e-python-e-pra-que-serve/>> Acesso em: 31/05/2021.

HEWETT, BAECKER, CARD, CAREY, GASEN, MANTEI, PERLMAN, STRONG E VERPLANK. **ACM SIGCHI Curricula for Human-Computer Interaction**. ACM, NY.

Disponível em: <<http://old.sigchi.org/cdg>, 1992>.

HILL, W.; HOROWITZ, P. A Arte da Eletrônica. 3ª ed. 2018.

INTELBAS. **Sistema de Automação Residencial Intelbras Mibo Home** Disponível em: <<https://loja.intelbras.com.br/sistema-automacao-residencial-mibo-home>> Acesso em: 02/06/2021.

ISO (1997). ISO 9241-11: **Ergonomic requirements for office work with visual display terminals (VDTs). Part 11 — Guidelines for specifying and measuring usability**. Genève: International Organization for Standardization.

IBM Cloud. **Internet of Things Platform**. Disponível em: https://cloud.ibm.com/catalog/services/internet-of-things-platform?_ga=2.24111930.1802952936.1622500773-229868618.1621555551> Acesso em: 31/05/2021.

LASTMINUTEENGINEERS.COM. **datasheet**. Disponível em: <<https://lastminuteengineers.com/datasheets/dht11-datasheet.pdf>> Acesso em: 19/10/2020.

LI, Q. **Real-time Concepts for Embedded Systems**. 2003.

LIMA, S, F. **A AUTOMAÇÃO E SUA EVOLUÇÃO**. Disponível em: <https://www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1_16.pdf> Acesso em: 15/06/2021.

MOUSER.COM. **DHT11 Humidity & Temperature Sensor**. Disponível em: <<https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>> Acesso em: 20/10/2020.

MUNDO PROJETADO. **LDR- O que é e como funciona**. Disponível em: <http://mundoprojetado.com.br/ldr-o-que-e-e-como-funciona/>> Acesso em: 03/05/2021.

MUNDO DA ELETRICA. **O que é NBR-5410?** Disponível em: <<https://www.mundodaeletrica.com.br/o-que-e-nbr-5410/>> Acesso em: 18/06/2021.

MOSQUITTO.ORG. **Documentation**. Disponível em: <<https://mosquitto.org/documentation/>> Acesso em 28/05/2021.

MELO, DIEGO. **O que é Python? [Guia para iniciantes]** disponível em: <<https://tecnoblog.net/405640/o-que-e-python-guia-para-iniciantes/>> Acesso em: 31/05/2021.

MIBRASIL, **Kit com Sensores Inteligentes Smart Home** Disponível em: <<https://www.mibrasil.com.br/kit-smart-home-com-sensores-inteligentes-x351-p1968>> Acesso em: 04/06/2021.

OLIVEIRA, E. **Conhecendo o Sonoff Relé Wifi para Automação Residencial** <Disponível

em:<https://blogmasterwalkershop.com.br/automacao/conhecendo-o-sonoff-rele-wifi-para-automacao-residencial>> Acesso em: 01/06/2021.

PATTERSON, D. A.; HENNESSY, J. L. **Computer Organization and Design: The Hardware/Software Interface 5th Edition**. Morgan Kaufmann-Elsevier. 2013.

PEREIRA, L.A. **Automação Residencial: rumo a um futuro pleno de novas soluções**. Disponível em:<<http://www.luizantoniopereira.com.br/downloads/publicacoes/AutomacaoResidencial-ISA2007.pdf>>

ROVEDA, U. **O que é Python, para que serve e por que que aprender?** Disponível em: <<https://kenzie.com.br/blog/o-que-e-python/>>Acesso em: 01/06/2021.

SEM AUTOR: Raspberrypi.org - **FAQs**. Disponível em:<<https://www.raspberrypi.org/documentation/faqs/>>. Acesso em: 05, maio de 2021.

STALLINGS, W. Arquitetura e organização de computadores. Pearson Prentice-Hall, 10ª ed., São Paulo. 2017.

SEM AUTOR: Relé - **O que é e como funciona**. Athos Electronics, 2016. Disponível em: <<https://athoselectronics.com/rele/>>. Acesso em: 23, fevereiro de 2021.

TANENBAUM, A. Organização Estruturada de Computadores. 6ª Edição, 2013.

TOCCI, R. J; WIDMER, N.S; MOSS, G. L. **Sistemas Digitais - Princípios e Aplicações**. Prentice-Hall, 11ª ed. São Paulo. 2011.

WEBER, R. F. Fundamentos de arquitetura de computadores. 4ª ed. Bookman. Porto Alegre. 2012.

YUAN, M. **Conhecendo o MQTT**. Disponível em: <https://developer.ibm.com/br/technologies/iot/articles/iot-mqtt-why-good-for-iot/>. Acesso em:27/05/2021.

7. ANEXOS E APÊNDICES

Algoritmo do controle de iluminação

```
#Hello IO World
#Posting Raspberry Pi "data" to Adafruit IO network
#by Vini Caetano: ViniCaetanoTCC
#TCC 2021

# Importa o client do Adafruit

from Adafruit_IO import MQTTClient
import time
import sys

# Importa GPIO para realizar os acionamentos
import RPi.GPIO as GPIO

# Meu Username criado no Adafruit IO
# Chave gerada para meu Username no Adafruit IO
USER_ADAFRUIT = #'MEU_USER_NAME'
USER_ADAFRUIT_KEY = #'MY_KEY'

# Escolha FEED predentido. Exemplo, LÂMPADA_QUARTO 1
IO_FEED1 = 'GPIO2'
IO_FEED2 = 'GPIO3'
IO_FEED3 = 'GPIO4'
IO_FEED4 = 'GPIO15'
IO_FEED6 = 'GPIO18'
# Meu Username para o meu feed - geralmente o mesmo de cima
IO_FEED_USERNAME = 'MEU_USERNAME'

# configurações de GPIO
GPIO.setmode(GPIO.BCM) #Indica que utilizaremos o endereçamento dos GPIOs que
aparecem no PINOUT e não o número do PINO
GPIO.setup(2, GPIO.OUT) #Quero que meu pino GPIO02 seja output
GPIO.setup(3, GPIO.OUT) #Quero que meu pino GPIO03 seja output
GPIO.setup(4, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)

##### Métodos criados pra usar o mqtt client e ficar ouvindo u
m tópico
def connected(client):
```

```

# quando conectado, ele subscrive num tópico
client.subscribe(IO_FEED1, IO_FEED_USERNAME)
client.subscribe(IO_FEED2, IO_FEED_USERNAME)
client.subscribe(IO_FEED3, IO_FEED_USERNAME)
client.subscribe(IO_FEED4, IO_FEED_USERNAME)
client.subscribe(IO_FEED6, IO_FEED_USERNAME)

def disconnected(client):
    print('conexao fechou. fecha o programa')
    sys.exit(1)

def message(client, feed_id, payload):
    # quando recebe a informação do tópico
    # Apresentamos na sysout os dados recebidos
    print('Feed {0} novo valor recebido: {1}'.format(feed_id, payload))

    if feed_id == IO_FEED1:
        # Aciona ou não a GPIO
        # Manda o resultado da comparação "not (payload == 'ON')" para a GPIO em q
        uestão
        GPIO.output(2, not (payload == 'ON')) # Quando usamos Ligado x Desliga
        do no botao configurado no dashboard do adafruit, caso contrario, usar ON/OFF
    if feed_id == IO_FEED2:
        # Aciona ou não a GPIO
        GPIO.output(3, not (payload == 'ON')) # Quando usamos Ligado x Desliga
        do no botao configurado no dashboard do adafruit, caso contrario, usar ON/OFF
    if feed_id == IO_FEED3:
        # Aciona ou não a GPIO
        GPIO.output(4, not (payload == 'ON')) # Quando usamos Ligado x Desliga
        do no botao configurado no dashboard do adafruit, caso contrario, usar ON/OFF
    if feed_id == IO_FEED4:
        GPIO.output(15, not (payload == 'ON'))
    if feed_id == IO_FEED6:
        GPIO.output(18, not (payload == 'ON'))
#----- Area de instancias
# Cria instancia mqtt.
client = MQTTClient(USER_ADAFRUIT, USER_ADAFRUIT_KEY)

# seta os metodos de callback para cair nos metodos criados anteriormente,
# a partir de cada evento.
client.on_connect = connected
client.on_disconnect = disconnected
client.on_message = message

# tenta conectar
client.connect()

# abre um thread pra ficar ouvindo o tópico em background
client.loop_background()

```

```

#Inicializa GPIOs
GPIO.output(2, True)
GPIO.output(3, True)
GPIO.output(4, True)
GPIO.output(15, True)
GPIO.output(18, True)

# Segura o script ligado, apenas ouvindo
while True:
    time.sleep(5)
    print ('Estado do GPIO 2 -> ' + str(GPIO.input(2)))
    print ('Estado do GPIO 3 -> ' + str(GPIO.input(3)))
    print ('Estado do GPIO 4 -> ' + str(GPIO.input(4)))
    print ('Estado do GPIO 15-> ' + str(GPIO.input(15)))
    print ('Estado do GPIO 18-> ' + str(GPIO.input(18)))

```

Algoritmo do DHT11 e LDR

```

from Adafruit_IO import MQTTClient #mqtt client do adafruit Dashboard
import RPi.GPIO as GPIO
import Adafruit_DHT
import time
import sqlite3
import datetime

USER_ADAFRUIT = #'MEU_USER_NAME'
USER_ADAFRUIT_KEY = #'MY_KEY'

IO_TEMP = 'tcc1_temperatura'
IO_UMID = 'tcc1_umidade'

IO_TCC_LDR = 'tcc_ldr'

IO_FEED_USERNAME = 'MEU_USERNAME'

sensor = Adafruit_DHT.DHT11

pino = 25 # GPIO 25
resistorPin = 21 #GPIO 21 - BOARD 40

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

#conecta ou cria o banco dht.db na pasta db
con = sqlite3.connect("db/dados.db")
#cria um cursor do sqlite3

```

```

cur = con.cursor()
#monta string de criação da tabela caso não exista uma
query = "CREATE TABLE IF NOT EXISTS tb_dht (id INTEGER PRIMARY KEY, Temperatur
a REAL, Umidade REAL, Horario TEXT)"
#executa a query que foi montada
cur.execute(query)

#monta string de criação da tabela caso não exista uma
query = "CREATE TABLE IF NOT EXISTS tb_ldr (id INTEGER PRIMARY KEY, Valor REAL
, Horario TEXT)"
cur.execute(query)

#####3
# AREA ADAFRUIT DASHBOARD
def connected(client):
    # Quando conectado, ele subscreve num tópico
    print('Conectado...')

def disconnected(client):
    print('conexao fechou. fecha o programa')
    sys.exit(1)

def message(client, feed_id, payload):
    # Quando recebe a informação do tópico
    # Apresentamos na sysout os dados recebidos
    print('Feed {0} novo valor recebido: {1}'.format(feed_id, payload))

client = MQTTClient(USER_ADAFRUIT, USER_ADAFRUIT_KEY)

client.on_connect = connected
client.on_disconnect = disconnected
client.on_message = message

client.connect()

client.loop_background()

#####

while True:

    print ('***** INICIO DOS SENSORES *****')
    #####
    # AREA LDR
    #
    GPIO.setup(resistorPin, GPIO.OUT)
    GPIO.output(resistorPin, GPIO.LOW)

```

```

time.sleep(0.1)

now = datetime.datetime.now()
now_str = now.strftime("%d/%m/%Y %H:%M:%S")

GPIO.setup(resistorPin, GPIO.IN)
currentTime = time.time()
diff = 0

while(GPIO.input(resistorPin) == GPIO.LOW):
    diff = time.time() - currentTime
    # Quando o tempo de diff fica acima do limite
    # parametrizado nos testes de medida,
    # indica que o LDR já se encontra há muito tempo
    # exposto a baixa luminosidade e não há necessidade
    # de se contar mais tempo. Já chegamos no limite
    # e o LDR encontra-se em alta impedância
    if diff >= 5:
        diff = 5
        break

valor = diff
print ('***** Leitura do LDR')
print('bruto=' + str(valor))
valor = abs((5-valor)*20)
print ('luminosidade='+str(valor))
client.publish(IO_TCC_LDR, valor)
# Cria a query de inclusão de dados na tabela ldr
query = "insert into tb_ldr (Valor, Horário) values (?, ?)"

cur.execute(query , (valor, now_str))
con.commit()

#####
# AREA DHT
print ('***** Leitura do LDR INICIO')
umid, temp = Adafruit_DHT.read_retry(sensor,pino)
print ('***** Leitura do LDR FIM')

if umid is not None and temp is not None:
    print("Temperatura = " + str(temp))
    print("Umididade = " + str(umid))

# Cria a query de inclusão de dados na tabela dht
query = "insert into tb_dht (Temperatura, Umididade, Horário) values (?,
?, ?)"

```

```

        # Executa a query de inclusão montada, inserindo valores na tabela
        cur.execute(query , (temp, umid, now_str))
        # Grava as alterações efetuadas no arquivo do banco
        con.commit()
        client.publish(IO_TEMP, temp)
        client.publish(IO_UMID, umid)
    else:
        print("Erro,falha em obter dados")

    print("Espera 2 horas para próxima leitura")
    #time.sleep((60*60)*2) # 60 minutos * 2
    time.sleep(10) # 10 segundos

con.close()

```

Algoritmo Sensor PIR

```

from Adafruit_IO import MQTTClient #mqtt client do adafruit Dashboard
import RPi.GPIO as GPIO
import time

# Meu username criado no adafruit IO
# Chave gerada para o Meu Username no adafruit IO
USER_ADAFRUIT = #'MEU_USER_NAME'
USER_ADAFRUIT_KEY = #'MY_KEY'

IO_PRESENCA = 'tcc_presenca'

# Username dono do feed - geralmente o mesmo de cima
IO_FEED_USERNAME = 'MEU_USERNAME'

def connected(client):
    # quando conectado, ele subscrive num topico
    print('Conectado...')

def disconnected(client):
    print('conexao fechou. fecha o programa')
    sys.exit(1)

def message(client, feed_id, payload):
    # quando recebe a informacao do topico
    # Apresentamos na sysout os dados recebidos
    print('Feed {0} novo valor recebido: {1}'.format(feed_id, payload))

```

```

GPIO.setmode(GPIO.BCM)
pino = 11
GPIO.setup(pino, GPIO.OUT)
GPIO.output(pino, GPIO.HIGH)
time.sleep(5)
GPIO.setup(pino, GPIO.IN)
print('Primeira leitura -> ' + str(GPIO.input(pino)))

#----- Area de instancias
# Cria instancia mqtt.
client = MQTTClient(USER_ADAFRUIT, USER_ADAFRUIT_KEY)

# seta os metodos de callback para cair nos metodos criados anteriormente,
# a partir de cada evento.
client.on_connect      = connected
client.on_disconnect   = disconnected
client.on_message      = message

# tenta conectar
client.connect()

# abre uma thread pra ficar ouvindo o topico em background
#client.loop_background()

client.publish(IO_PRESENCA, 'NAO_DETECTADA')

anter = GPIO.HIGH
while True:
    time.sleep(0.1)

    atual = GPIO.input(pino)

    if atual == GPIO.LOW:
        print('movimento....')
        client.publish(IO_PRESENCA, 'DETECTADA')
        time.sleep(3)
        GPIO.setup(pino, GPIO.OUT)
        GPIO.output(pino, GPIO.HIGH)
        time.sleep(5)
        GPIO.setup(pino, GPIO.IN, pull_up_down=GPIO.PUD_UP)

    else:
        if anter != atual:
            client.publish(IO_PRESENCA, 'NAO_DETECTADA')
            print('SEM movimento...')

        anter = atual

```