UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATHEUS COELHO SOLHA

CONSTRUÇÃO DE MÓDULOS BASEADOS EM INTERNET DAS COISAS PARA COLETA DE DADOS E GERENCIAMENTO DE INFORMAÇÕES

BAURU
2018

MATHEUS COELHO SOLHA

CONSTRUÇÃO DE MÓDULOS BASEADOS EM INTERNET DAS COISAS PARA COLETA DE DADOS E GERENCIAMENTO DE INFORMAÇÕES

Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação apresentado ao Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho" – UNESP, Câmpus de Bauru.
Orientadora: Profª. Associada Roberta Spolon

BAURU
2018

MATHEUS COELHO SOLHA

CONSTRUÇÃO DE MÓDULOS BASEADOS EM INTERNET DAS COISAS PARA COLETA DE DADOS E GERENCIAMENTO DE INFORMAÇÕES

Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação apresentado ao Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho" – UNESP, Câmpus de Bauru.

Aprovado em 13/11/2018.

BANCA EXAMINADORA

Orientadora: Profª. Associada Roberta Spolon
Departamento de Computação
Faculdade de Ciências
UNESP - BAURU

Profª. Drª. Simone das Graças Domingues Prado
Departamento de Computação
Faculdade de Ciências
UNESP - BAURU

Profª. Drª. Andréa Carla Gonçalves Vianna
Departamento de Computação
Faculdade de Ciências
UNESP - BAURU

AGRADECIMENTOS

Agradeço a minha família por todo apoio, seja ele financeiro ou emocional. Durante toda minha jornada acadêmica sempre estiveram apoiando minhas decisões, comemorando minhas vitórias e oferecendo consolo nos momentos de tristeza.

Agradeço a Barbara Carvalho Silva por ter me acompanhado a todo momento durante todo o período da graduação, sendo paciente e sempre estando disposta a me ajudar independente de qual fosse a situação e sempre compartilhando sabedoria adquirida ao longo da vida na faculdade.

A Mariana Almeida Pereira Dias que caminhou ao meu lado durante o desenvolvimento de projetos importantes de minha formação acadêmica e também de momentos pessoais tão importantes quanto.

A Gabriel Luiz Bastos de Oliveira por ter se disposto a compartilhar todos seu conhecimento na área de computação e me guiado quando houve o surgimento de dúvidas.

A Vítor Henrique Tonsig Alves pela paciência e disposição de me acompanhar na busca pelos dispositivos utilizados no desenvolvimento desse trabalho e as boas risadas oferecidas em momentos tensos de preocupação.

A Profª. Associada Roberta Spolon por ser uma grande professora e pesquisadora que marcou meu período acadêmico de forma positiva e me deu a honra de realizar um trabalho como seu orientando.

A todos os professores que já fizeram parte de minha educação e àqueles que ainda o farão.

A todos os meus colegas de trabalho que se dispuseram de diversas formas para me oferecer apoio desde meu momento de ingresso na empresa.

A todos meus amigos, tanto os que ainda se fazem presente em minha vida quanto os que já não mantenho contato independente do motivo, por terem vivenciado momentos memoráveis de minha vida.

"A ciência atua na fronteira entre o conhecimento e a ignorância sem medo de admitir que não sabemos. Não há nenhuma vergonha nisso. A única vergonha é fingir que temos todas as respostas."
(Neil deGrasse Tyson)

RESUMO

Com a expansão das pesquisas e implementações na área de Internet das Coisas, diversos desenvolvedores e estudiosos criaram ferramentas que permitiram a interpretação dos dados gerados pelo ambiente ao nosso redor e a disponibilização dessas informações em uma rede de internet. Com esse avanço tecnológico diversos dispositivos foram desenvolvidos e melhoraram a utilização do conceito de Internet das Coisas, como por exemplo o Raspberry Pi e o NodeMCU. Através da junção de microcontroladores, sensores e programação, um módulo baseado nesse conceito pode ser criado, o que possibilita um melhor gerenciamento sobre determinado ambiente ou região. Baseado nessas ideias, esse projeto visou a criação de três módulos capazes de coletar dados sobre temperatura, umidade, luminosidade e o estado de abertura de portas, e então disponibilizá-los na internet através da utilização do protocolo *Message Queuing Telemetry Transport* (MQTT). Mediante o estudo dos mais diferentes componentes eletrônicos, do planejamento de código modular para cada sensor e do desenvolvimento, pôde-se construir os três módulos de baixo custo possibilitando um possível desenvolvimento em larga escala, permitindo a utilização nas mais diversas áreas.

Palavras-chave: Raspberry Pi, NodeMCU, Internet das Coisas.

ABSTRACT

Given the expansion of research and implementations in the Internet of Things area, several developers and scholars created tools that allowed the interpretation of the data generated by the environment around us and the availability of this information in an Internet network. With this technological advancement several devices were developed and improved the use of the Internet of Things concept, such as RaspberryPi and NodeMCU. Through the combination of microcontrollers, sensors and programming, a module based on this concept can be created, which allows better management of a particular environment or region. Based on these ideas, this project aimed at creating three modules capable of collecting data on temperature, humidity, luminosity and the opening of doors state, and then making them available on the Internet through the use of the Message Queuing Telemetry Transport (MQTT) protocol. Through the study of the most diverse electronic components, modular code planning for each sensor and development, we were able to construct the three low-cost modules, enabling a large scale development, allowing them to be used in many different areas.

Keywords: Raspberry Pi, NodeMCU Internet of Things.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CLI	*Command-Line Interface*
DIY	*Do It Yourself*
GPIO	*General Input and Output*
IBM	*International Business Machines*
IoMT	*Internet of Medical Things*
IoT	*Internet of Things*
MQTT	*Message Queuing Telemetry Transport*
OASIS	*Organization for the Advancement of Structured Information Standards*
REST API	*Application Programming Interface based on Representational State Transfer*
RFID sem chip	Chipless Radio Frequency Identification
SO	Sistema Operacional
WLAN	*Wireless Local Area Network*

SUMÁRIO

1 INTRODUÇÃO

Devido ao grande desmembramento da atenção dos seres humanos em diversas tarefas, o tempo ser limitado para cada uma delas e não existir uma grande acurácia na realização das mesmas, existe uma grande perda de dados já que antes da existência da Internet das Coisas, também conhecida como *Internet of Things* (IoT), muitos dados acabavam sendo deixados para trás, no entanto, com o imenso avanço tecnológico diversos utensílios vem ganhando a capacidade de se conectar com a internet e o conceito de IoT se torna cada vez mais presente. A partir do momento em que um item é capaz de coletar os dados de seus arredores e transmiti-los para que sejam utilizados de forma inteligente por outros membros da rede a ideia de dispositivo IoT é colocada em prática, dessa forma, além dos aparelhos usuais como computadores, dispositivos móveis e consoles de *videogame*, itens comuns passam a ter a capacidade de se comunicar com a internet (ASHTON, 2009).

Com o passar do tempo o custo de processamento e de acesso a internet tem caído e consequentemente cada vez mais dispositivos tem se conectado a essa imensa rede (GOLDMAN SACHS, 2014). Devido a esse barateamento e o surgimento de componentes eletrônicos de baixo custo como o Raspberry Pi e o NodeMCU, a quantidade de dispositivos IoT vem aumentando em grande escala, o quê gera um grande impacto no número de aparelhos que estão conectados a internet. De acordo com os dados divulgados por Lueth (2018), esse número já excede o valor de 17 bilhões de aparelhos, onde 7 bilhões são apenas dispositivos IoT (Figura 1).

Figura 1 – Número total de conexões de dispositivos ativos em todo o mundo

Fonte: Lueth (2018, tradução nossa)

As mais diversas áreas podem usufruir das vantagens fornecidas pela Internet das Coisas, desde a área urbana, com o desenvolvimento de cidades inteligentes onde os elementos de trânsito podem comunicar-se entre si (GOLDMAN SACHS, 2014), até a área de monitoramento de ambientes de desenvolvimento, já que de acordo com Roelofsen (2002), o aprimoramento dos espaços de trabalho reduz o número de reclamações e ociosidade entre os trabalhadores. Assim pode-se realizar um controle de temperatura, umidade e luminosidade, e também indicar a existência de portas abertas no local, para que possa ser feito o aprimoramento.

Para a captação de dados do meio externo são utilizados sensores, os quais são definidos por Wilson (2004) como sendo dispositivos capazes de captar e converter um fenômeno físico em um sinal elétrico. Assim, através do acoplamento de um ou mais sensores com os pinos gerais de entrada e saída, *General Input and Output* (GPIO) presentes em dispositivos como o NodeMCU, pode-se realizar a leitura dos dados captados e transmiti-los através da internet, já que o mesmo possui um microcontrolador capaz de realizar comunicação sem fio, conhecida como *Wireless Local Area Network*, (WLAN). Com a finalidade de realizar a troca de informações de maneira leve e rápida, o protocolo mais utilizado na comunicação dentro das redes é conhecido como *Message Queuing Telemetry Transport* (MQTT) ou pelo termo MQTT *broker*.

Dado o baixo custo financeiro dos componentes eletrônicos básicos para compor um módulo IoT e a fácil implementação do protocolo MQTT em diversas linguagens de programação, a construção dos mesmos foi popularizada através da cultura *maker*, que é fortemente baseada na ideia de a aprendizagem ser mais eficaz através do fazer (CRAIG, 2015), consequentemente, o conceito *Do It Yourself* (DIY), também conhecido como Faça Você Mesmo, é muito empregado na área de Internet das Coisas já que modelos básicos podem ser facilmente encontrados na internet e adaptados para usos específicos.

Por meio de um simples servidor que colete as informações transmitidas através do protocolo já citado, uma interface de programação de aplicativos baseada em transferência de estado representacional, popularmente conhecida como *Application Programming Interface based on Representational State Transfer* (REST API), pode ser desenvolvida, permitindo que as informações transmitidas por módulos IoT que utilizem o mesmo padrão de protocolo possam ser consumidas por outras aplicações através de requisições utilizando o *Hypertext Transfer Protocol* (HTTP), as quais são de extrema popularidade no desenvolvimento de softwares (MULESOFT, 2016). Pode-se utilizar como exemplo a plataforma de automatização residencial chamada Home Assistant, que permite a comunicação com um servidor MQTT e disponibiliza os dados através de uma *REST API* ou até mesmo por meio de uma interface amigável ao usuário.

1.1 Objetivos

O objetivo geral e os objetivos específicos que deverão ser atingidos com o desenvolvimento deste projeto estão listados a seguir.

1.1.1 Objetivos Gerais

O objetivo geral desse projeto é construir três módulos IoT que sejam capazes de coletar dados de sensores e transmiti-los de forma adequada para serem consumidos por outros sistemas, como por exemplo, um sistema de *dashboard* para controle de projetos.

1.1.2 Objetivos Específicos

a) Estudar diferenças entre diversos sensores de luminosidade;

b) Estudar diferenças entre diversos sensores de abertura eletromagnéticos;

c) Estudar diferenças entre diversos sensores de temperatura;

d) Estudar qual melhor microcontrolador para cada sensor;

e) Criar um servidor MQTT para realizar a troca de dados;

f) Integrar cada sensor com seu respectivo microcontrolador;

g) Realizar a programação de cada microcontrolador para envio das informações para a Internet;

h) Receber através de um software as informações que foram transmitidas na rede por cada módulo.

1.2 Organização da Monografia

O presente trabalho divide-se em seções, sendo esta a primeira (Introdução). As demais seções estão dispostas na seguinte ordem:

a) **Seção 2 - Fundamentação Teórica:** apresentação dos conceitos teóricos envolvidos no trabalho.

b) **Seção 3 - Metodologia:** descrição do método de pesquisa, bem como as ferramentas e materiais utilizadas.

c) **Seção 4 - Desenvolvimento:** descrição de toda a etapa de desenvolvimento do projeto.

d) **Seção 5 - Conclusão:** considerações finais sobre o trabalho.

e) **Seção 6 - Trabalhos Futuros:** possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

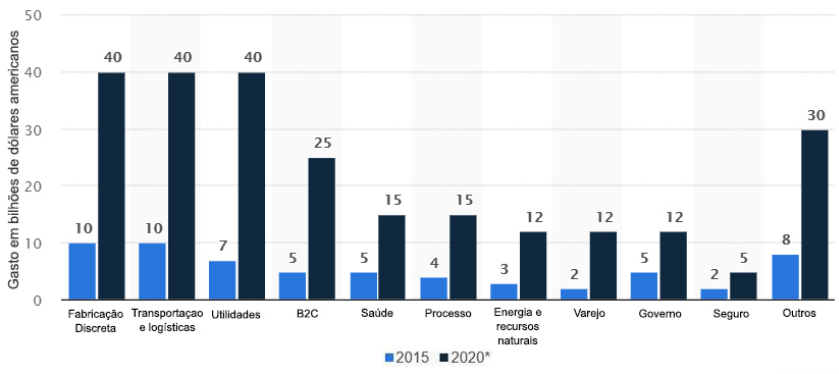
Neste capítulo são descritos conceitos abordados ao longo do trabalho, pertencentes ao domínio de Internet das Coisas, *Message Queuing Telemetry Transport*, Raspberry Pi, Arduino, NodeMCU e diversos tipos de sensores.

2.1 Internet das Coisas

O crescimento na área de IoT tem alavancado um imenso investimento nesse mercado, o que tem despertado cada vez mais o interesse de desenvolvedores por essa área. De acordo com Thibodeau (2015), em 2015 um mapeamento da indústria de desenvolvimento de software estimou que 19% de um universo com 19 milhões de desenvolvedores estariam trabalhando com projetos relacionados a essa área e que padrões e protocolos ainda eram assuntos obscuros dentro do ramo, no entanto pode-se destacar facilmente um avanço na área uma vez que os desenvolvedores tem demonstrado maior interesse em protocolos específicos de comunicação e plataformas para o desenvolvimento como apontado por Bjorlin (2018).

De acordo com projeções realizadas por Statista (2018), o gasto até 2020 em diversos setores para o desenvolvimento do ramo de IoT, incluindo o setor da saúde, superará o valor de 15 bilhões de dólares americanos, indicado por meio da Figura 2. Dado esse crescimento no setor da saúde, a IoT ganhou uma nova ramificação que foi popularizada como *Internet of Medical Things* (IoMT), a qual visa o estudo e desenvolvimento de técnicas e dispositivos que auxiliem no monitoramento da saúde da população e que sejam capazes de evitar o aumento do custo de acesso a essa área (MARR, 2018).

Figura 2 – Gastos com Internet das Coisas ao redor do mundo em 2015 e 2020



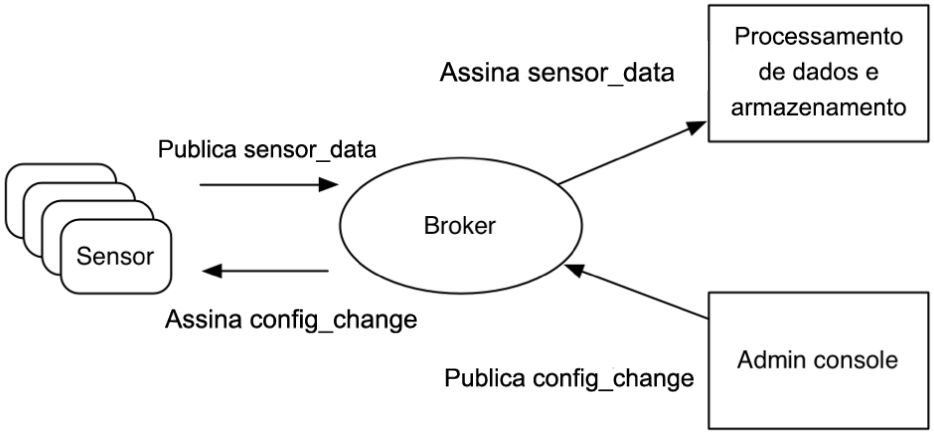
Fonte: Statista (2018, tradução nossa)

2.2 Message Queuing Telemetry Transport

Inventado e desenvolvido pela *International Business Machines* (IBM) no final da década de 90 com o intuito de vincular sensores em *pipelines* de petróleo a satélites através da utilização de um modelo de publicação e assinatura, o protocolo foi ganhando cada vez mais popularidade, assim, no final de 2014 acabou tornando-se oficialmente um padrão aberto da *Organization for the Advancement of Structured Information Standards* (OASIS), com suporte nas mais diversas linguagens de programação, usando implementações de software livre (YUAN, 2017).

O MQTT atua sobre a camada *TCP/IP*, a qual trata-se de um conjunto de protocolos de comunicação de dados (HUNT, 1998). O modelo utilizado pelo protocolo possui dois tipos de entidades para o transporte de mensagens, sendo elas o cliente, que pode publicar ou assinar determinados tópicos, e o *broker*, que é o servidor responsável por receber todas as mensagens publicadas através dos clientes e de encaminha-las para todos aqueles que tiverem assinado o tópico que recebeu a mensagem. Esse conceito é exemplificado através da Figura 3, a qual possui um elemento atuando como *broker* e um sensor fazendo o papel do cliente através da assinatura do tópico *config_change*, que permite receber parâmetros de configuração, e a publicação de mensagens com os dados coletados em outro chamado *sensor_data*. Dessa forma outros dois clientes são criados, onde um deles assina o tópico *sensor_data*, assim recebe as mensagens que são repassadas pelo *broker* e o outro publica mensagens no tópico *config_change*, permitindo que o sensor seja configurado remotamente (YUAN, 2017).

Figura 3 – Modelo MQTT de publicação e assinatura para sensores IoT



Fonte: Yuan (2017, tradução nossa)

2.3 Raspberry Pi

O Raspberry Pi é um computador de baixo custo que possui o tamanho de um cartão de crédito, apresenta entradas para periféricos como monitor, teclado, mouse e até mesmo fones de ouvido. Contém também pinos de entrada e saída digitais para que possam ser conectados sensores e outros dispositivos. A capacidade de processamento e quantidade de funcionalidades de hardware varia entra cada modelo. O componente foi desenvolvido com a intenção de ser amplamente utilizado por crianças em escolas no aprendizado da programação e sobre o funcionamento de computadores (RASPBERRY PI, 2014). Na Figura 4 são mostrados três modelos diferentes.

Figura 4 – 3 Modelo B+ (a), 3 Modelo B (b) e 2 Modelo B (c)

(a) (b) (c)

Fonte: elaborado pelo autor

A Tabela 1 apresenta uma comparação entre os três modelos apresentados anteriormente.

Tabela 1 – Comparativo entre os modelos de Raspberry Pi

Versão	2 Modelo B	3 Modelo B	3 Modelo B+
Memória RAM	1 GB	1 GB	1 GB
Processador	ARMv7 quad-core	ARMv8 quad-core	ARMv8 quad-core
Velocidade CPU	900 MHz quad-core single-core	1.2 GHz MHz quad-core	1.4 GHz quad-core
Portas USB	4	4	4
Ethernet	Sim	Sim	Sim
WLAN	Não	Sim	Sim

Fonte: elaborado pelo autor

O sistema operacional (SO) recomendado para ser utilizado no Raspberry Pi é o *Raspbian*, que é baseado na distribuição *Linux* chamada *Debian* para que possa ser executado em processadores de arquitetura ARM (RASPBIAN, 2012).

2.4 Arduino

O Arduino é uma plataforma eletrônica *open-source* composta por uma placa de hardware, a qual pode ser facilmente programada através da linguagem de programação C++, e um software para desenvolvimento. Com finalidade de compilar o código do software de forma correta para cada tipo de microcontrolador diferente, é utilizado uma variação do *framework* chamado *Wiring* feito especificamente para a plataforma. As placas Arduino são de baixo custo e podem ser facilmente integradas com os mais diversos tipos de sensores e dispositivos eletrônicos capazes de fornecer dados. Existem diversos modelos onde cada um deles possui uma grande variedade de aplicações. A Tabela 2 apresenta alguns modelos e as principais diferenças entre eles.

Tabela 2 – Comparativo entre diferentes modelos de placas Arduino

Versão	Memória	Pinos Digitais	Pinos Analógicos	Clock
UNO	32Kb	14	6	16Mhz
MEGA 2560	256KB	54	16	16Mhz
NANO	16 ou 32Kb	14	8	12Mhz

Versão	Tensão de Operação
UNO	5V
MEGA 2560	5V
NANO	5V

Fonte: elaborado pelo autor

Como citado por Arduino (2012) nenhum dos modelos possui acesso à internet. Para que as placas possam se conectar a rede, é necessário que integrem componentes como o *Wifi Shield CC3000* (Figura 5) ou o *ESP8266 ESP-01* à placa Arduino (THOMSEN, 2014; THOMSEN, 2015).

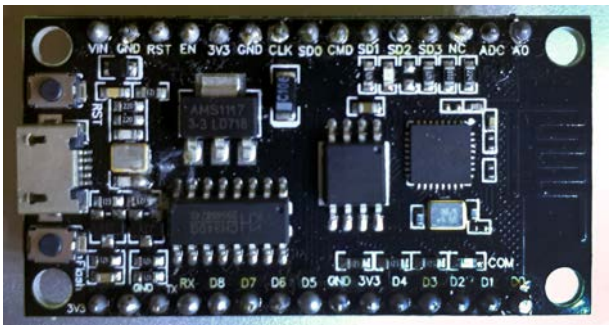
Figura 5 – Módulo *Wireless ESP8266 ESP-01*

Fonte: Thomsen (2015)

2.5 NodeMCU

O NodeMCU (Figura 6) é uma placa de desenvolvimento que possui um módulo ESP8266 12-E já integrado, o mesmo permite a comunicação em uma rede WLAN de internet. A placa possui todos os componentes necessários para a utilização do módulo *wireless*, dessa forma sua programação se torna simples e bastante similar aos diversos modelos de placa Arduino (MURTA, 2018).

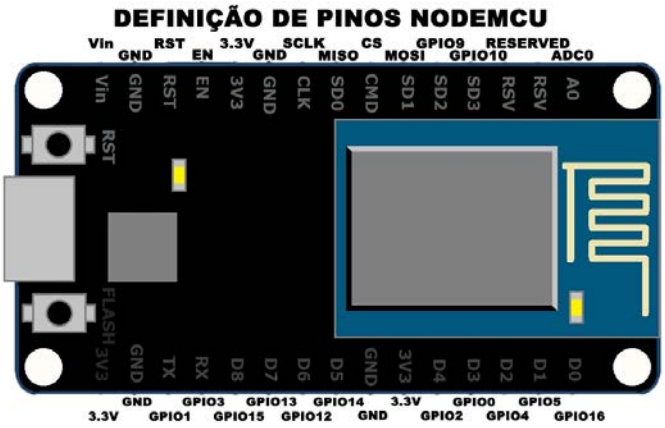
Figura 6 – Placa de desenvolvimento NodeMCU



Fonte: elaborado pelo autor

A placa dispõe de 10 pinos GPIO (Figura 7), uma velocidade de clock de até 160 megahertz e uma memória interna de até 4 megabytes, onde velocidade de clock pode ser entendida como a velocidade que um microprocessador executa as instruções (ROUSE, 2005).

Figura 7 – Definição de pinos NodeMCU



Fonte: elaborado pelo autor

2.6 Sensores

As seguintes subseções tem como propósito apresentar os diversos sensores que foram base de estudo para alcançar os objetivos propostos nesse trabalho.

2.6.1 Temperatura

Esse tipo de sensor é utilizado para realizar a medição de temperatura. Existem diversos modelos (Figura 8) para os mais variantes casos de uso, onde cada modelo possui a precisão, a amplitude de alcance e o preço variados. Através da Tabela 3 pode-se observar as principais diferenças entre os modelos LM35, DS18B20, DHT11 (TEXAS INSTRUMENTS, 1999; MAXIM INTEGRATED, 2007; ADAFRUIT, 2012).

Figura 8 – LM35 (a), DS18B20 (b) e DHT11 (c)

(a) (b) (c)

Fonte: elaborado pelo autor

Tabela 3 – Comparativo entre sensores de temperatura

Modelo	Amplitude de Alcance	Precisão	Preço
LM35	-55ºC à 150ºC	0.5ºC para mais ou menos	R$9,90[1] à R$10,78[2]
DS18B20	-55ºC à 125ºC	0.5ºC para mais ou menos	R$7,80[3] à R$39,90[4]
DHT11	0ºC à 50ºC	2ºC para mais ou menos	R$6,80[5] à R$25,00[6]

Fonte: elaborado pelo autor

Notas:

1 Disponível em: <http://bit.ly/lm35vidasilicio>. Acesso em: 12 out. 2018

2 Disponível em: <http://bit.ly/lm35sensormoduleblack>. Acesso em: 12 out. 2018

3 Disponível em: <http://bit.ly/DS18B20-lightinthebox>. Acesso em: 12 out. 2018

4 Disponível em: <http://bit.ly/flipeflopDs18b20>. Acesso em: 12 out. 2018

5 Disponível em: <http://bit.ly/dht11mekanus-ml>. Acesso em: 12 out. 2018

6 Disponível em: <http://bit.ly/vanguardeletronicsdht11>. Acesso em: 12 out. 2018

2.6.2 Umidade

Com o intuito de realizar medições de umidade relativa, os sensores HS1101, HTU21D e DHT11 são listados na Tabela 4 com suas principais diferenças e preços, onde os dois primeiros podem ser vistos na Figura 9 (HUMIREL, 2002; TE CONNECTIVITY, 2012; ADAFRUIT, 2012).

Figura 9 – HS1101 (a) e HTU21D (b)

(a) (b)

Fonte: elaborado pelo autor

Tabela 4 – Comparativo entre sensores de umidade

Modelo	Amplitude de Alcance	Precisão	Preço
HS1101	1% à 99%	2% para mais ou menos	Indisponível
HTU21D	0% à 100%	2% para mais ou menos	R$16,46[1] à R$39,63[2]
DHT11	20% à 80%	5% para mais ou menos	R$6,80[3] à R$25,00[4]

Fonte: elaborado pelo autor

Notas:

1 Disponível em: <http://bit.ly/banggoodhumid>. Acesso em: 12 out. 2018

2 Disponível em: <http://bit.ly/dxhumid>. Acesso em: 12 out. 2018

3 Disponível em: <http://bit.ly/dht11mekanus-ml>. Acesso em: 12 out. 2018

4 Disponível em: <http://bit.ly/vanguardeletronicsdht11>. Acesso em: 12 out. 2018

2.6.3 Estado de abertura de porta

Este tipo de sensor é utilizado para realizar a leitura do estado de abertura de portas. O modelo sem fio Intruder F105547, envia seus dados sem a necessidade de comunicação física com outro componente através de rádio frequência de 433 megahertz (ECP, 20–). Assim, torna-se necessário que para leitura dos dados, seja implementado ou integrado um receptor de rádio frequência também de 433 megahertz (THOMSEN, 2013).

Já um modelo genérico conhecido como reed switch magnético de porta, consiste na combinação de duas lâminas compostas por material ferro magnético que não se encostam, sendo envoltas por um recipiente de vidro. Ao receber a influência de um campo magnético no eixo das lâminas, as duas se atraem devido a natureza ferro magnética de sua composição, permitindo a passagem de corrente elétrica (PICKERINGRELAY, 2018).

A Figura 10 mostra fisicamente cada um dos modelos.

Figura 10 – Intruder F105547 (a) e reed switch magnético de porta (b)

(a) (b)

Fonte: elaborado pelo autor

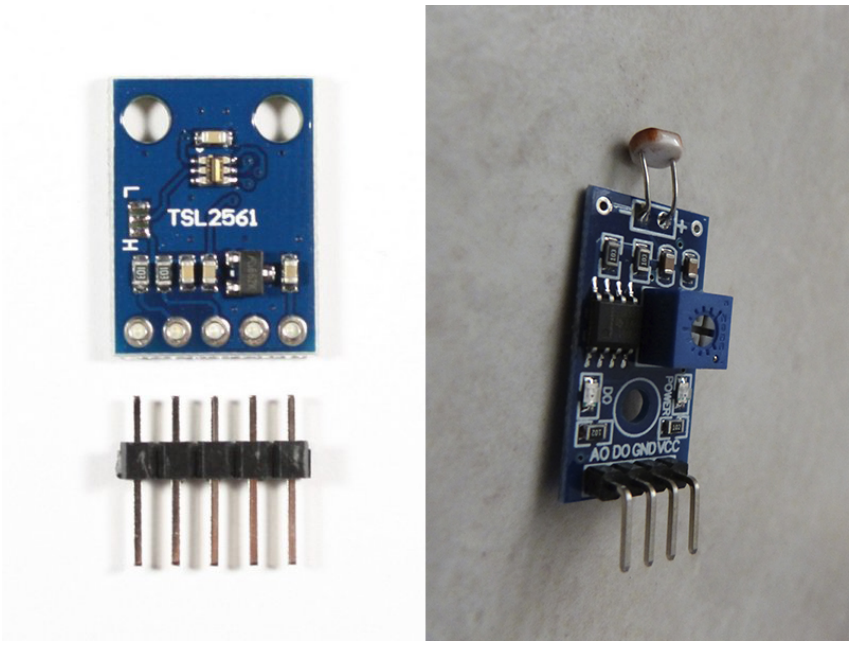
2.6.4 Luminosidade

Através da utilização de um fotodiodo capaz de captar a frequência emitida por luz visível aos olhos humanos e outro capaz de captar raios infravermelhos, o sensor TSL2561 realiza a leitura desses dados e os transforma em sinal digital de saída, permitindo uma fácil leitura já que a informação é dada em medida de luminosidade entre 1 e 40.000 lux (TEXAS ADVANCED OPTOELECTRONIC SOLUTIONS INC., 2009). Possui um custo elevado devido sua robustez e precisão.

Em contrapartida, o módulo sensor de luminosidade fotossensitivo, composto por um Resistor Dependente de Luz (LDR), tem o valor de sua resistência alterada conforme a quantidade incidente de luz (GIBBS, 2009), assim em sua saída analógica pode-se obter o valor atual dessa resistência ou, através da saída digital, pode-se verificar se o ambiente está iluminado ou não (USINAINFO, s.d).

A Figura 11 apresenta os dois modelos de sensores de luminosidade estudados para esse trabalho.

Figura 11 – TSL2561 (a) e módulo sensor de luminosidade fotossensitivo (b)



(a) (b)

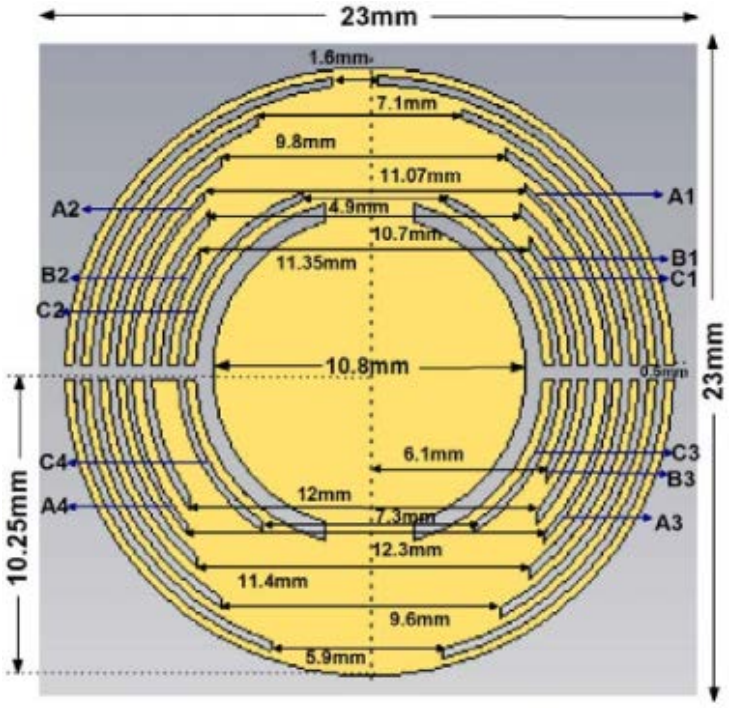
Fonte: elaborado pelo autor

2.7 Trabalhos Correlatos

O projeto apresentado por Satti et al. (2018) utiliza uma tecnologia recente conhecida como *Chipless Radio Frequency Identification* (RFID sem chip), a qual faz uso de um

radiador de formato circular de cobre com pequenos sulcos que são preenchidos com sensores químicos capazes de captar a temperatura e a umidade para formar uma *tag* (Figura 12) apta a ser estimulada através da emissão de rádio frequência sendo suficiente para que a mesma possa emitir de volta os dados coletados. Assim, os dados podem ser coletados por um dispositivo receptor desse tipo de frequência e emitidos através da internet para monitorar a temperatura e umidade.

Figura 12 – Design da *tag* RFID sem chip capaz de medir temperatura e umidade



Fonte: Satti et al. (2018)

3 METODOLOGIA

Neste capítulo são apresentadas as etapas em que o projeto foi dividido e os materiais utilizados para o desenvolvimento.

3.1 Métodos e Etapas

O presente trabalho foi organizado nas seguintes etapas:

a) busca bibliográfica de todos os assuntos abordados durante o trabalho;

b) estudo sobre o funcionamento dos sensores e placas de hardware possíveis;

c) planejamento de hardware;

d) planejamento de software;

e) estudo sobre o protocolo e o desenvolvimento de um servidor MQTT;

f) escolha e implementação do software responsável por captar os dados dos módulos;

g) montagem do hardware em paralelo com a programação do NodeMCU;

h) testes de captação dos dados por meio do software responsável;

i) apresentação dos dados em um dashboard para controle de projetos[1].

3.2 Materiais Utilizados

As subseções posteriores apresentam os materiais escolhidos para serem utilizados no processo de desenvolvimento do projeto.

3.2.1 Placa de hardware de desenvolvimento

Diante das possibilidades apresentadas na seção 2.4 e na seção 2.5, deve-se salientar o fato de que a placa de desenvolvimento NodeMCU já possui um módulo de comunicação sem fio com a internet, diferente da placa Arduino que necessita a utilização de componentes externos para poder realizar essa comunicação, logo, foi escolhida a placa NodeMCU. Outro fator importante que levou a essa decisão é a disponibilidade no mercado com um preço relativamente baixo.

[1] Trabalho de conclusão de curso de Barbara Carvalho Silva realizado na Universidade Estadual Paulista - Câmpus de Bauru para o curso de Bacharelado em Ciência da Computação que visa o desenvolvimento de um dashboard para controle de projetos

Também foi necessário escolher um modelo de microcomputador, dessa forma, para este trabalho pôde-se notar que qualquer modelo apresentado anteriormente na seção 2.3 seria aplicável, assim, por uma questão de disponibilidade o Raspberry Pi 2 modelo B (Figura 13) foi escolhido para o desenvolvimento de um servidor MQTT e execução do software de captura de dados. O sistema operacional que é executado é o Raspbian Stretch que foi desenvolvido para poder funcionar em processadores de arquitetura ARM, o qual permite a compilação e execução de programas criados para sistemas operacionais que utilizam o Kernel Linux.

Figura 13 – Raspberry Pi 2 modelo B

Fonte: elaborado pelo autor

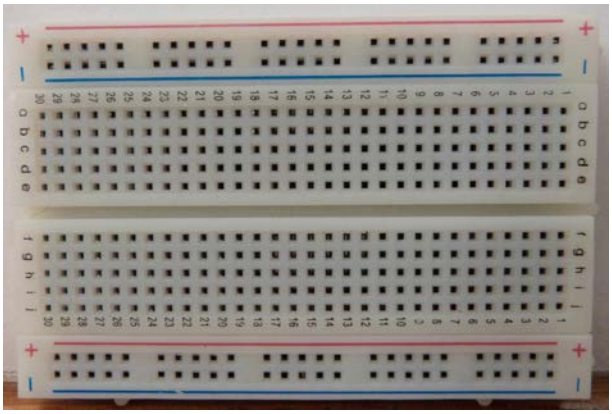
3.2.2 Elementos de montagem do hardware

Para conectar o NodeMCU com os sensores foi utilizada uma placa de ensaio, mais conhecida como *protoboard*, de 400 pontos (Figura 14). A conexão entre eles foi realizada utilizando *jumpers* Figura 15, que são fios metálicos capazes de conduzir eletricidade.

3.2.3 Sensores

Ao efetuar a análise dos comparativos realizados na seção 2.6, foi constatado que os sensores mais adequados para o atual projeto são o DHT11, para medição de temperatura e umidade, o *reed switch* magnético de porta, capaz de detectar o estado de abertura das mesmas e o módulo sensor de luminosidade fotossensitivo, que fornece a informação se o ambiente está iluminado ou não. O principal fator decisivo na escolha de todos os sensores foi o fácil acesso aos produtos e a disponibilidade, uma vez que o projeto não visa a construção de módulos de extrema precisão. Os mesmos podem ser vistos na Figura 16.

Figura 14 – Placa de ensaio de 400 pontos

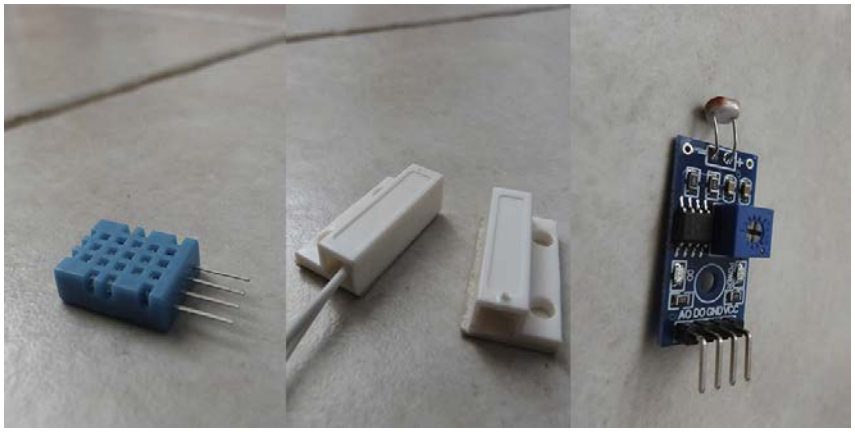


Fonte: elaborado pelo autor

Figura 15 – Fios metálicos para conexão entre pontos do *protoboard*

Fonte: elaborado pelo autor

Figura 16 – DHT11 (a), Reed Switch (b) e LDR (c)



(a) (b) (c)

Fonte: elaborado pelo autor

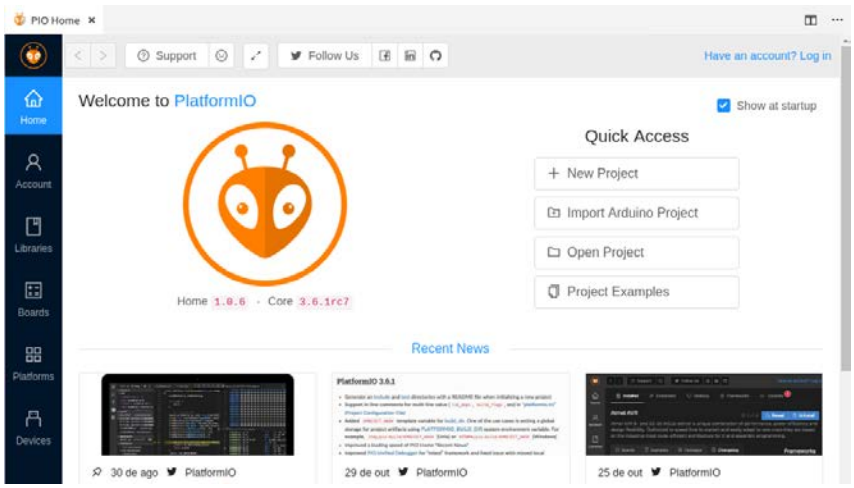
3.3 Tecnologias e Ferramentas Utilizadas

Essa seção tem como objetivo listar e explicar as tecnologias e ferramentas utilizadas durante o desenvolvimento do projeto.

3.3.1 PlatformIO

O PlatformIO é um Ambiente de Desenvolvimento Integrado para programação de diversos microcontroladores através da linguagem C++, sendo citado como um ecossistema de desenvolvimento IoT. Seu único requisito para poder rodar em qualquer sistema operacional é a existência do interpretador de linguagem Python. Por ser uma aplicação de console, pode ser integrada a diversos editores de códigos já existentes como, por exemplo, o Visual Studio Code[2] (Figura 17) e o Atom[3] (PLATFORMIO, 2014).

Figura 17 – Tela principal do PlatformIO integrado ao Visual Studio Code



Fonte: elaborado pelo autor

Seu funcionamento, listado de forma simplificada por PlatformIO (2014), é dado da seguinte maneira:

a) o usuário escolhe o microcontrolador que será programado através de um arquivo de configuração de projeto chamado *platformio.ini* (Figura 18);

b) baseado em uma lista de placas, o PlatformIO realiza o *download* das bibliotecas e ferramentas necessárias instalando-as automaticamente;

c) assim o desenvolvedor pode escrever o código e a IDE garante que será compilado, preparado e carregado à placa de destino.

[2] <https://code.visualstudio.com/>
[3] <https://atom.io/>

Figura 18 – Exemplo de arquivo platformio.ini para programar o NodeMCU

```
You, 23 days ago | 1 author (You)
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:nodemcuv2]
platform = espressif8266
board = nodemcuv2
framework = arduino

monitor_speed = 115200
build_flags = -DSSID_NAME=HELLO -DSSID_PASWORD=WORLD

```

Fonte: elaborado pelo autor

A partir do ambiente pode-se também gerenciar e instalar bibliotecas de código disponibilizadas por outros usuários.

3.3.2 Docker

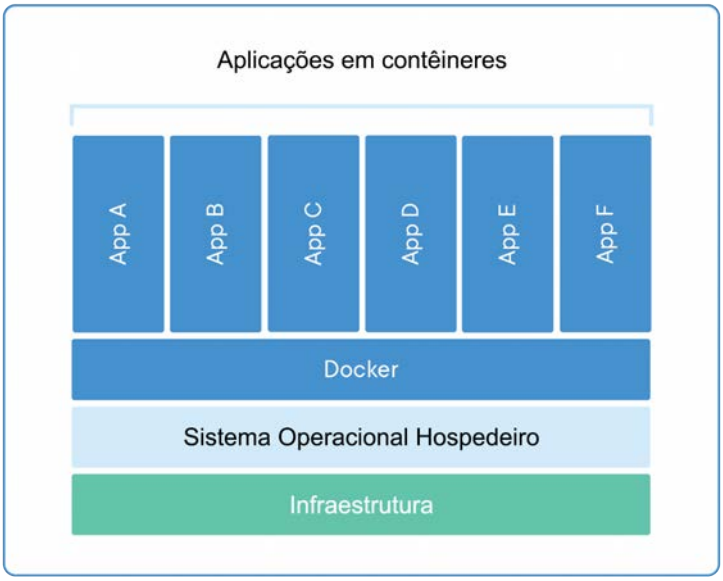
O Docker, de acordo com Docker (2018a), é uma tecnologia recente, mas que tem ganhado um grande espaço entre desenvolvedores, já que permite a fácil entrega de aplicações para clientes.

> O Docker é uma plataforma para desenvolvedores e administradores de sistemas desenvolverem, implementarem e executarem aplicativos com contêineres. O uso de contêineres do Linux para implementar aplicativos é chamado de conteinerização. Os quais não são novos, diferente de sua utilização para implantação fácil de aplicações. (DOCKER, 2018a, tradução nossa).

A inicialização de um contêiner é dada através da utilização de imagens, as quais são pacotes executáveis que já incluem todas as bibliotecas e outras dependências para que a aplicação seja executada. Eles podem ser facilmente associados a máquinas virtuais (VM), uma vez que compartilham o núcleo do sistema operacional linux que estiver rodando para criar um ambiente isolado (Figura 19). Apesar da associação realizada, existe uma diferença muito grande entre os dois em sua arquitetura e maneira de funcionamento, essa dessemelhança é que ao invés de compartilhar o núcleo, as VMs possuem uma abstração do hardware físico chamado Hypervisor, o qual permite que cada máquina virtual possua a cópia completa de um sistema operacional (Figura 20), assim tornam-se mais pesadas e demoradas para inicializar, já os contêineres são abstrações na camada de

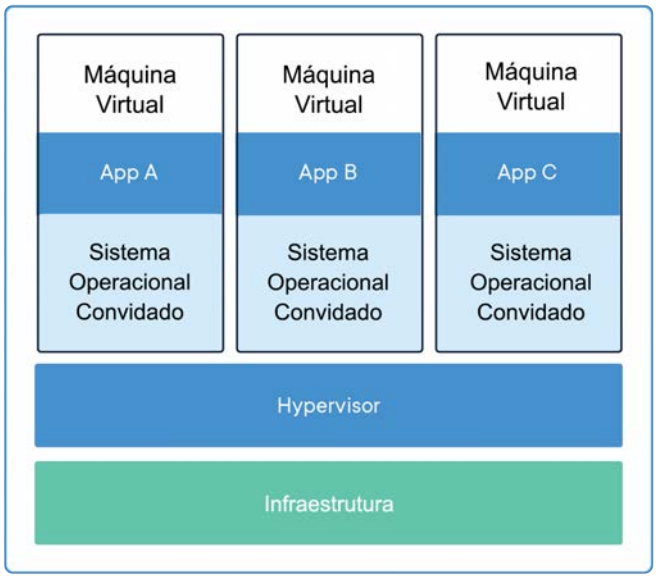
aplicação que empacotam o código e suas dependências, e a utilização do mesmo núcleo do sistema operacional hospedeiro. Dada essa forma de execução, conclui-se que os contêineres ocupam menos memória e possuem melhor desempenho comparado com as VMs. (DOCKER, 2018b).

Figura 19 – Arquitetura de execução de contêineres



Fonte: Docker (2018b, tradução nossa)

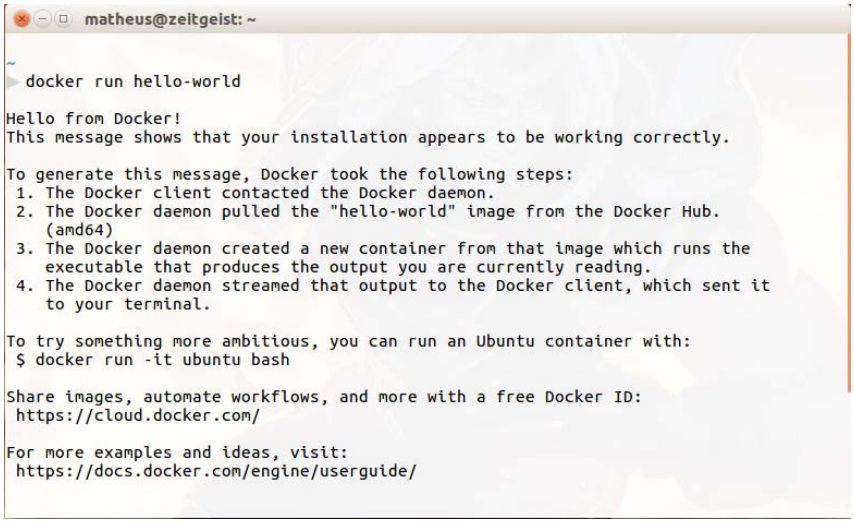
Figura 20 – Arquitetura de execução de máquinas virtuais



Fonte: Docker (2018b, tradução nossa)

A execução das imagens é feita através da utilização de uma interface de linha de comandos, conhecida *command-line interface* (CLI), a Figura 21 demonstra uma execução simples.

Figura 21 – Resultado de execução da imagem *hello-world*



Fonte: elaborado pelo autor

A criação de imagens elaboradas requerem a digitação de extensos comandos para serem executados, sendo que pode existir dependências e comunicações a serem realizadas entre contêineres. Dessa forma, foi desenvolvida uma ferramenta chamada *Docker Compose*, a qual é utilizada para definir e rodar aplicações Docker de múltiplos contêineres. Através de um único arquivo de serialização de dados de extensão YAML[4] chamado *docker-compose.yml*, são configurados todos os serviços a serem executados, a Figura 22 demonstra um exemplo.

[4] <http://yaml.org/>

Figura 22 – Exemplo de arquivo docker-compose.yml

Fonte: elaborado pelo autor

3.3.3 Node.js e Yarn

A linguagem de programação JavaScript foi originalmente desenvolvida para utilização em navegadores *web* permitindo que *scripts* fossem executados no lado cliente, os quais são definidos por Duarte (2013) como uma série de instruções a serem executadas pela máquina. Porém com a criação da plataforma Node.js a utilização da linguagem foi ampliada para escrita de aplicações que possam ser executadas no lado servidor, já que é baseada no motor de JavaScript utilizado pelo software Google Chrome. Através do uso de um modelo de aplicação voltado a eventos, a plataforma possui uma execução de código não bloqueante, ou seja, ela não é interrompida devido a elementos de entrada e saída, cálculos, etc. Assim o seu uso para criação de aplicações *web* cresceu rapidamente, uma vez que são facilmente escaláveis (NODEBR, 2016).

O Node.js permite a criação de pacotes de códigos para que possam ser compartilhados. Para serem utilizados é necessário usar uma ferramenta de gerenciamento. Nesse projeto foi utilizado o gerenciador chamado Yarn[5] devido a familiarização com a ferramenta.

Durante o desenvolvimento foram utilizados os seguintes pacotes:

a) **axios**[6]**:** permite a realização de requisições HTTP;

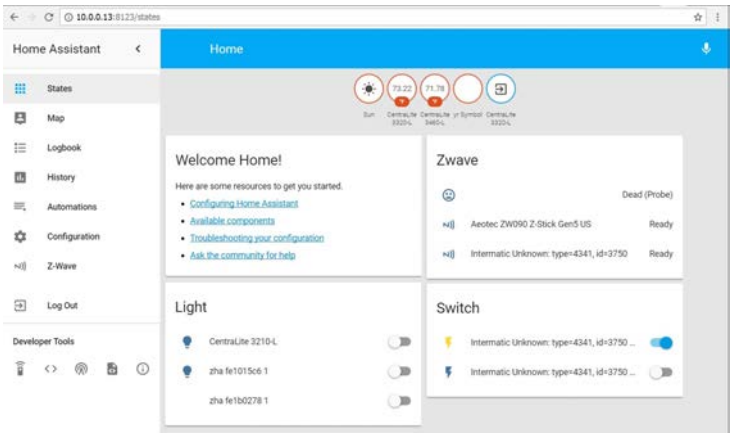
b) **express**[7]**:** *framework* para criação de aplicações *web* com Node.js.

[5] <https://yarnpkg.com/pt-BR/>
[6] <https://github.com/axios/axios>
[7] <https://expressjs.com/pt-br/>

3.3.4 Home Assistant

Home Assistant é uma plataforma *open source* para automação residencial que pode ser executada em diversos sistemas que possuam Python 3[8], como por exemplo, um Raspberry Pi (BAKER, 2017). O software foi desenvolvido com conceito de modularização, portanto possui suporte para diversos dispositivos IoT já existentes, por exemplo o equipamento Alexa[9], que permite a utilização de comandos de voz para se comunicar com outros aparelhos conectados ao Home Assistant. Também é fornecida uma interface de usuário através de um navegador *web* para exibição dos componentes (Figura 23).

Figura 23 – Interface de usuário disponibilizada pelo Home Assistant

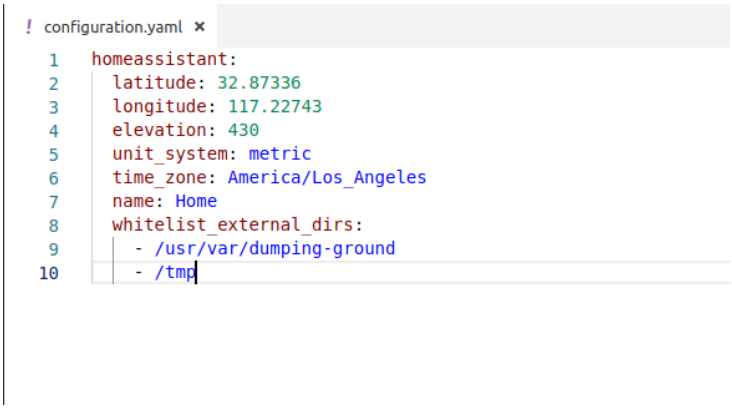


Fonte: Artik (2018)

As integrações com o sistema principal da plataforma são chamadas de componentes, os quais são facilmente configurados através do arquivo principal de configuração chamado *configuration.yaml* que possui extensão YAML[10] (Figura 24). Devido a utilização do protocolo MQTT durante o desenvolvimento do projeto, um componente que recebe o mesmo nome do protocolo será utilizado para que o sistema possa assinar tópicos e se torne capaz de receber os dados transmitidos através dos módulos IoT, assim eles podem ser exibidos na interface do usuário. A principal vantagem da utilização dessa plataforma é a existência de um componente que fornece uma REST API, para que as informações possam ser requisitadas por qualquer outra aplicação, por exemplo, um dashboard para controle de projetos.

[8] <https://www.python.org/download/releases/3.0/>
[9] Aparelho de som e assistente de voz desenvolvido pela empresa Amazon.
[10] <https://www.home-assistant.io/docs/configuration/yaml/>

Figura 24 – Arquivo de configuração principal do Home Assistant

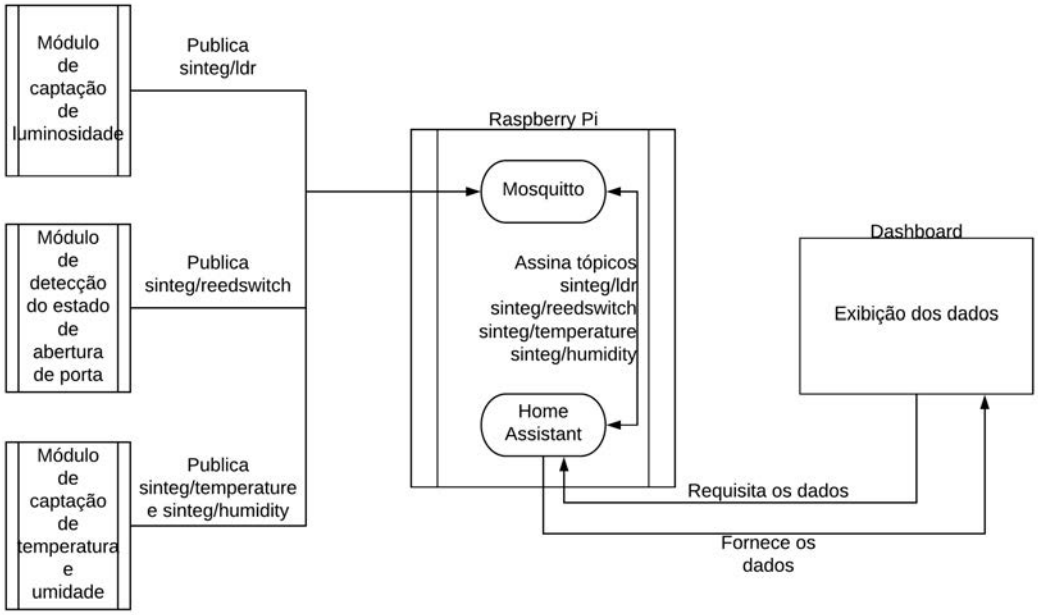


Fonte: elaborado pelo autor

4 DESENVOLVIMENTO

Esse capítulo irá abordar todo o processo de planejamento e desenvolvimento do trabalho, representado pela Figura 25, desde a criação da arquitetura dos módulos até o funcionamento dos mesmos.

Figura 25 – Arquitetura de funcionamento geral



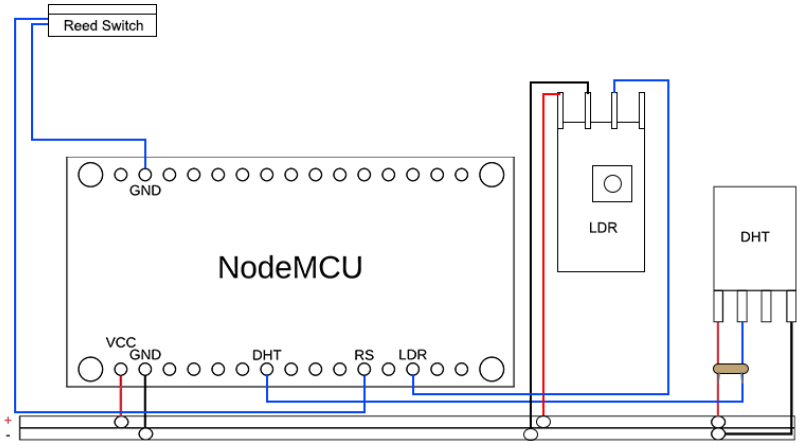
Fonte: elaborado pelo autor

4.1 Planejamento de Hardware

Para criação dos módulos IoT propostos, foi necessário elaborar um diagrama para representar a conexão entre cada componente eletrônico. Criou-se um primeiro esboço a mão para pensar na melhor distribuição dos pinos disponíveis na placa NodeMCU para que os sensores pudessem ser conectados, a maneira de disposição e conexão dos elementos permite que todos os sensores utilizem uma única placa ou sejam unificados em uma só (Figura 26).

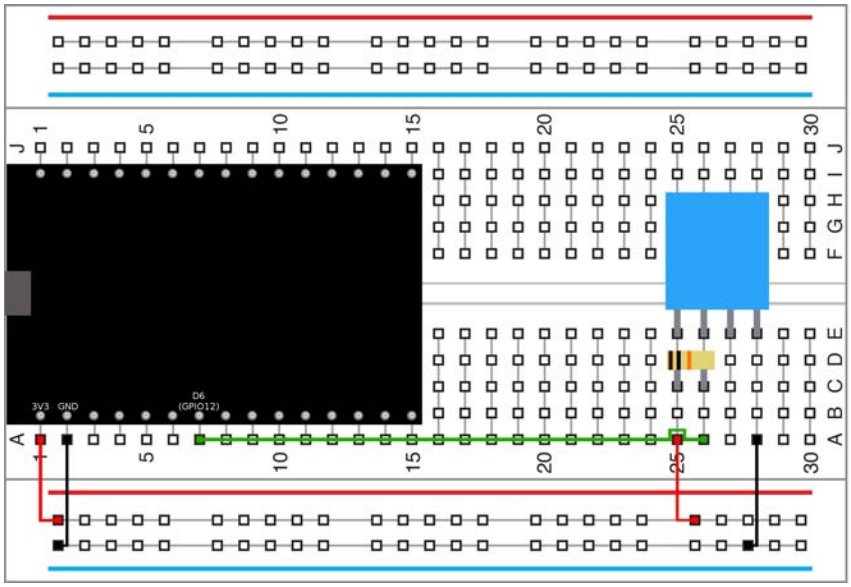
As figuras apresentam os diagramas finais digitais para os módulos de temperatura e umidade (Figura 27), luminosidade (Figura 28) e estado de abertura de porta (Figura 29).

Figura 26 – Esboço de conexão dos sensores com o NodeMCU



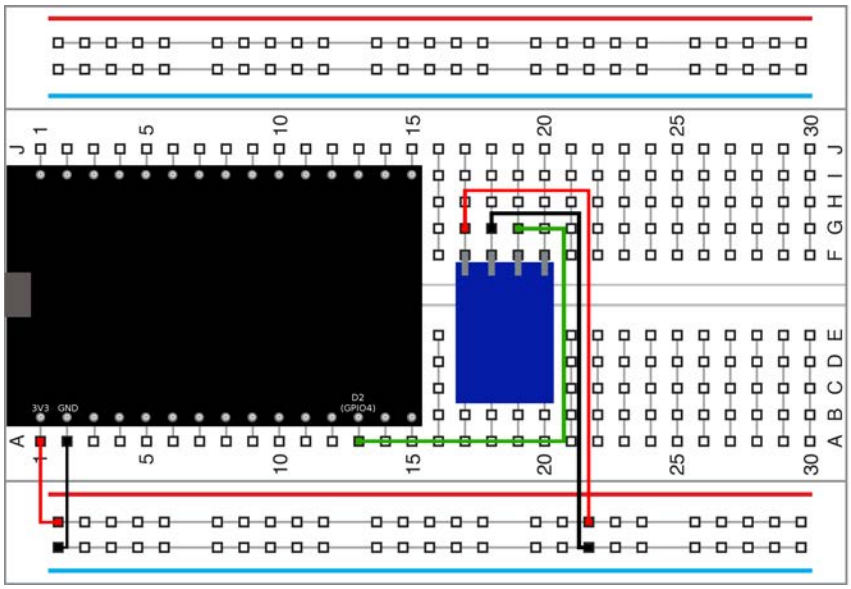
Fonte: elaborado pelo autor

Figura 27 – Diagrama de conexão do sensor DHT11



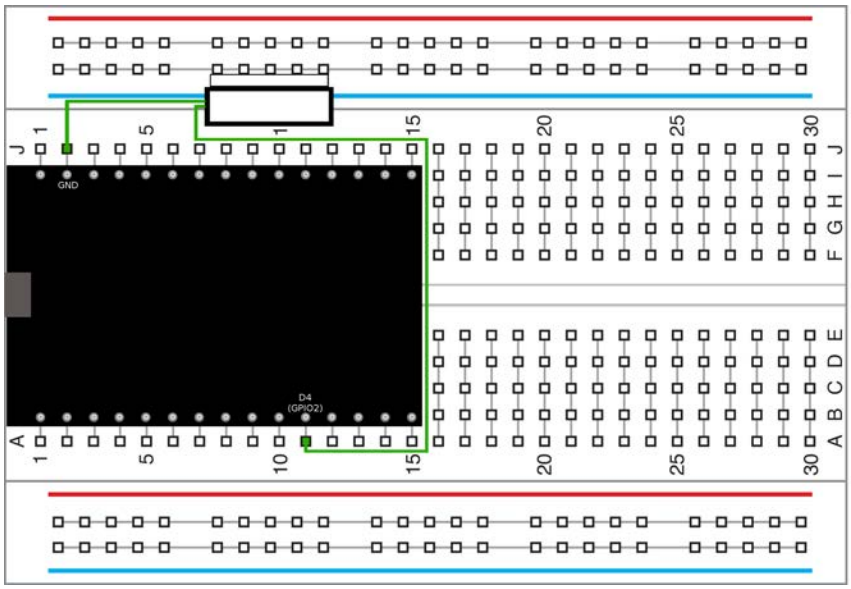
Fonte: elaborado pelo autor

Figura 28 – Diagrama de conexão do sensor de luminosidade



Fonte: elaborado pelo autor

Figura 29 – Diagrama de conexão do sensor para captação do estado de abertura de porta

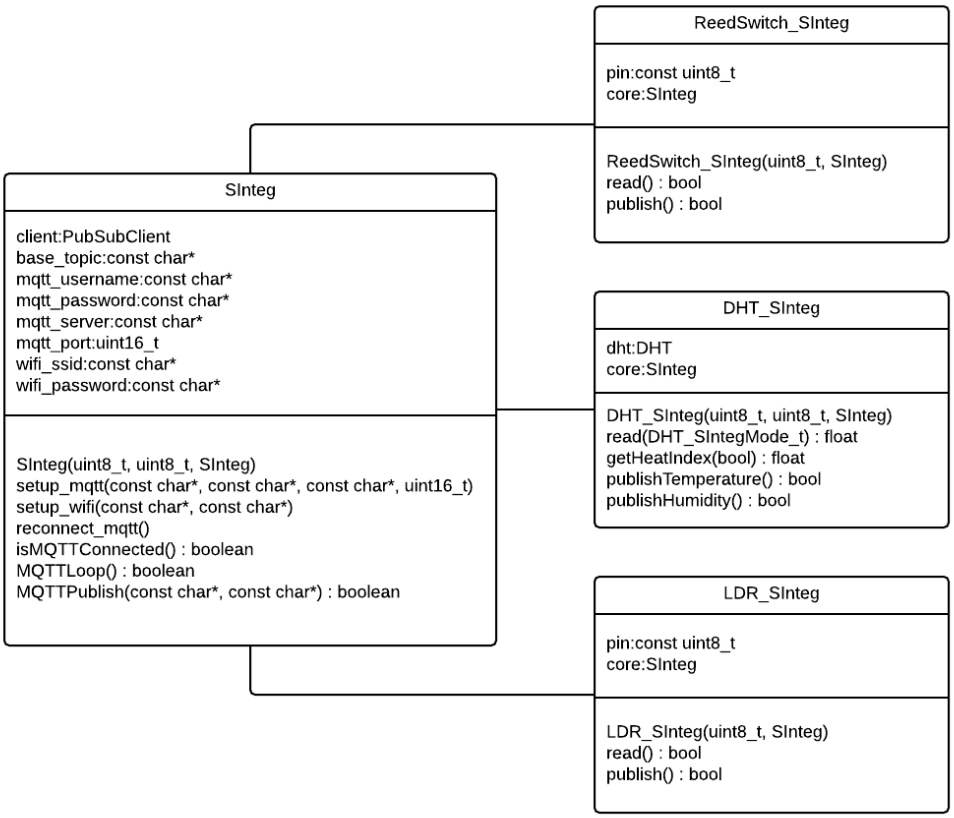


Fonte: elaborado pelo autor

4.2 Planejamento de Software

A programação da placa NodeMCU é realizada utilizando a linguagem C++. Por ser orientada a objetos, pode-se criar classes, as quais são consideradas formas para criação de objetos, onde cada uma contém funcionalidades específicas. Utilizando esse raciocínio foi desenvolvida uma classe para cada sensor, de forma que cada classe possua um algoritmo diferente para leitura dos dados, e uma classe que sirva como núcleo contendo as configurações e funções básicas do microcontrolador (Figura 30).

Figura 30 – Diagrama de classes do projeto



Fonte: elaborado pelo autor

Para poder simular um ambiente similar ao disponibilizado pelo sistema operacional do Raspberry Pi, o desenvolvimento de todas as aplicações que seriam utilizadas nesse computador foram definidas em um único arquivo *docker-compose.yaml* para que cada serviço rode como um contêiner diferente, assim garantindo a não interferência de qualquer que seja a origem. A fim de melhor organização, foi definida uma hierarquia de pastas que contém todos os arquivos necessários para a execução das imagens (Figura 31).

Figura 31 – Hierarquia de pastas

Fonte: elaborado pelo autor

4.3 Configuração do Raspberry Pi

Essa etapa consistiu na preparação do Raspberry Pi, sendo separada nas seguintes categorias:

a) instalação do sistema operacional;

b) desenvolvimento do servidor MQTT;

c) instalação e configuração do Home Assistant (subseção 3.3.4)

4.3.1 Instalação do Sistema Operacional

A memória do Raspberry Pi 2 modelo B consiste em um cartão do tipo microSD de 8 gigabytes (Figura 32), assim, para realizar a instalação do sistema operacional Raspbian o cartão foi inserido no computador para processo a gravação dos dados necessários.

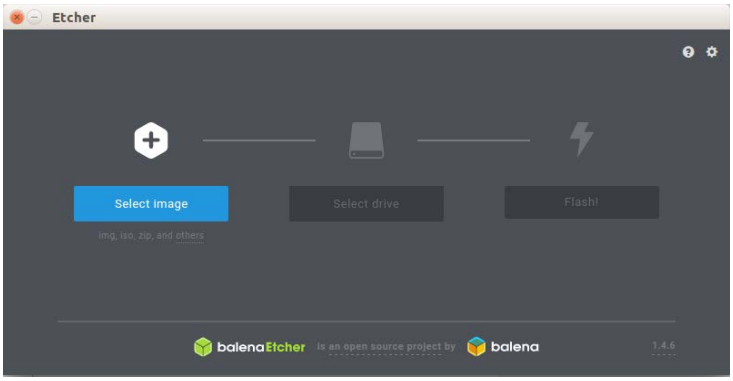
Primeiramente foi descarregada através da internet uma imagem do sistema opera-

Figura 32 – Cartão microSD utilizado no Raspberry Pi

Fonte: elaborado pelo autor

cional e um software específico, conhecido como Etcher[1] (Figura 33), para gravação do SO no cartão. O processo de instalação ocorreu ao iniciar a gravação, uma vez que a imagem a ser gravada e em qual cartão isso iria ocorrer foram selecionados.

Figura 33 – Tela principal do software Etcher

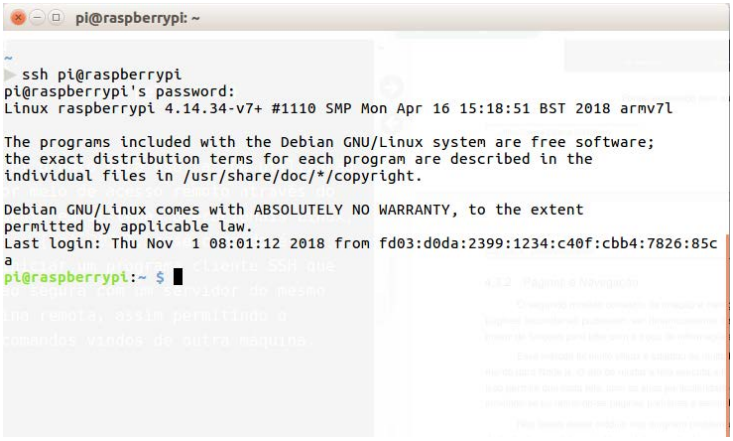


Fonte: elaborado pelo autor

Com o Raspberry Pi pronto para uso, a sua utilização foi realizada por meio de acesso remoto através do comando ssh (Figura 34) presente em sistemas operacionais Linux. De acordo com (SSH, 2018), esse comando é utilizado para iniciar um programa cliente SSH que permite a conexão segura com um servidor do mesmo tipo em um máquina remota, assim permitindo o recebimento de comandos vindos de outra máquina.

[1] <https://www.balena.io/etcher/>

Figura 34 – Demonstração do comando ssh



Fonte: elaborado pelo autor

4.3.2 Desenvolvimento do Servidor MQTT

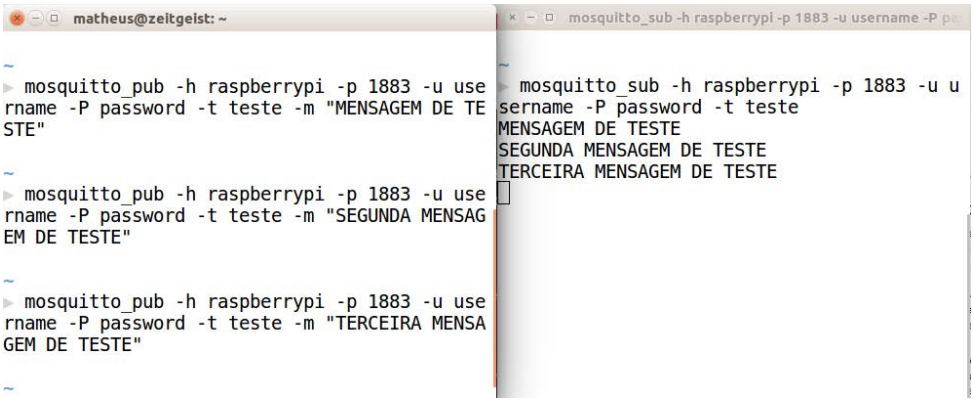
O desenvolvimento do servidor MQTT teve como base o *broker* chamado Mosquitto já disponibilizado pela fundação Eclipse. Primeiramente o programa foi instalado em um computador pessoal que utiliza um sistema operacional Linux, para verificar se suas funcionalidades requeridas são executadas de maneira estável e eficiente. O *broker* foi configurado através de um arquivo chamado *mosquitto.conf* para utilizar a porta 1883 com a finalidade de receber conexões de clientes para assinarem ou publicaram mensagens em tópicos. Com o intuito de criar um baixo nível de segurança de acesso ao servidor, foi definido um usuário e senha para que conexões fossem realizadas.

Em seguida, criou-se uma pasta principal, chamada *raspdock*, dentro da memória do Raspberry Pi, seguindo a hierarquia da Figura 31, contendo um arquivo *docker-compose.yaml* e outra pasta chamada mosquitto que detém os arquivos de configuração do contêiner correspondente ao servidor MQTT. No arquivo YAML foi definido um novo serviço em que todas as variáveis necessárias para criação correta do contêiner foram estabelecidas.

4.3.2.1 Teste de Funcionamento

Para verificar o funcionamento do servidor MQTT na porta 1883, em um computador pessoal foi utilizado os comandos *mosquitto_pub*, para publicar mensagens em um tópico chamado teste, e *mosquitto_sub* para assinar o mesmo tópico e poder receber as mensagens. Pode-se observar através da Figura 35 que as mensagens foram publicadas e entregues corretamente.

Figura 35 – Teste de funcionamento do servidor MQTT



Fonte: elaborado pelo autor

4.3.3 Instalação do Home Assistant

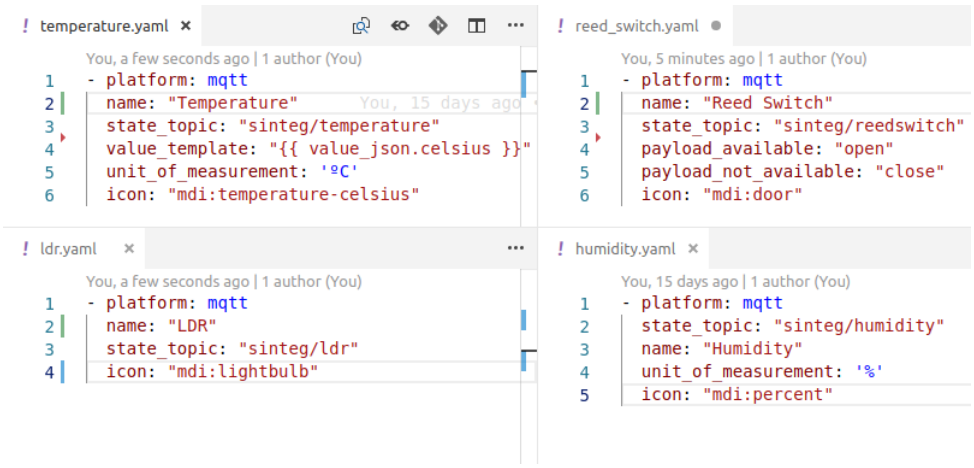
Para utilização da plataforma Home Assistant, criou-se uma pasta com o nome homeassistant contendo um arquivo base de configuração para o software. Nele foi adicionado o conteúdo da Figura 36 para indicar que os componentes api, mqtt e *sensor* seriam utilizados. A entrada das configurações do último componente foram separadas em quatro arquivos diferentes para melhor organização (Figura 37).

Figura 36 – Configuração dos componentes no Home Assistant

```
70  sensor: !include_dir_merge_list sensors/
71
72  mqtt:
73    broker: localhost
74    port: 1883
75    username: username
76    password: password
77
78  api:
79
```

Fonte: elaborado pelo autor

Figura 37 – Configuração dos sensores no Home Assistant



Fonte: elaborado pelo autor

4.4 Programação e Montagem dos Módulos

Para a programação da placa NodeMCU utilizou-se a biblioteca *Arduino* que fornece as funcionalidades básicas, sendo elas a comunicação serial para monitorar o funcionamento da placa, tipos diferentes para declaração de variáveis, dentre outros. Foram adicionadas as seguintes bibliotecas através do gerenciador disponibilizado pela IDE PlatformIO:

a) **PubSubClient**[2]**:** realiza a conexão com o servidor MQTT, permitindo a publicação de mensagens e a assinatura de tópicos;

b) **DHT *sensor library***[3]**:** efetua a leitura de sensores do tipo DHT11;

c) **Adafruit *Unified Sensor***[4]**:** biblioteca necessária para o funcionamento da DHT *sensor library*;

d) **ArduinoJson**[5]**:** utilizada para serialização e deserialização de dados do tipo JSON.

Uma vez que todas as dependências foram incluídas pôde-se desenvolver uma classe base contendo as funções responsáveis por configurar os dados para conexão com a internet e com o servidor MQTT. Utilizando os modificadores de acesso disponíveis na linguagem C++, alguns dados foram protegidos. Para que as classes dos sensores pudessem acessar esses dados, foi necessária a definição delas como amigas[6] da classe base.

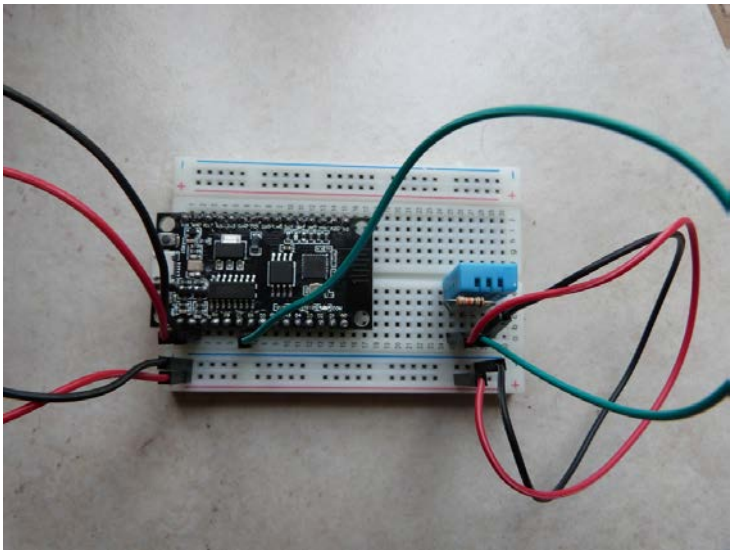
[2] <https://github.com/knolleary/pubsubclient>
[3] <https://github.com/adafruit/DHT-sensor-library>
[4] <https://github.com/adafruit/Adafruit_Sensor>
[5] <https://github.com/bblanchon/ArduinoJson>
[6] <http://www.cplusplus.com/doc/tutorial/inheritance/>

4.4.1 Temperatura e Umidade

Utilizando a biblioteca DHT *sensor library* pôde-se programar a classe do sensor de temperatura e umidade, sendo elaboradas funções para leitura dos dados e publicação de cada um deles nos tópicos *sinteg/temperature* e *sinteg/humidity* utilizando a biblioteca PubSubClient.

Em seguida, o módulo foi montado utilizando o protoboard, a placa NodeMCU, o sensor, os *jumpers* e um resistor de 10000Ω, seguindo o diagrama da Figura 27. O módulo produzido é apresentado na Figura 38.

Figura 38 – Montagem do módulo para captação de temperatura e umidade



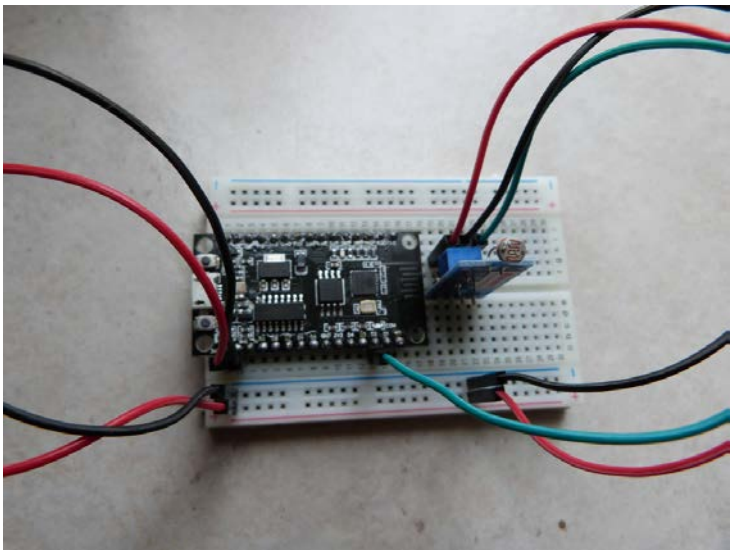
Fonte: elaborado pelo autor

4.4.2 Luminosidade

Utilizando a função para leitura de dados digitais disponibilizada pela biblioteca Arduino, programou-se uma classe com funções responsáveis por ler os dados provenientes do pino GPIO4 e pública-los em um servidor MQTT específico.

Depois da compilação correta do código para funcionamento da leitura de luminosidade utilizando o respectivo sensor, foi realizado o processo de montagem do módulo conforme apresentado na Figura 39, seguindo o diagrama apresentado na Figura 28.

Figura 39 – Montagem do módulo de captação de luminosidade



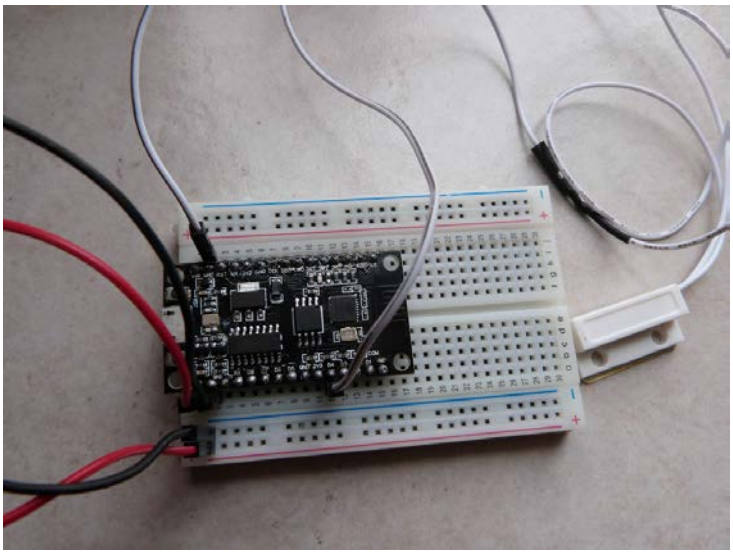
Fonte: elaborado pelo autor

4.4.3 Estado de Abertura de Porta

Com a utilização de funcionalidades disponibilizadas pela biblioteca Arduino, pode-se implementar em uma classe específica para o sensor de detecção de estado de abertura de porta, uma função encarregada de realizar a leitura do estado através do pino GPIO2, o qual foi configurado como INPUT_PULLUP que permite a leitura de sensores do tipo interruptor.

Devido a diferença de formato entre os fios do sensor e as entradas do protoboard, foi necessária a utilização de tubo termo retrátil para unir a ponta de cada sensor a um *jumper*, dessa forma pôde ser conectado ao protoboard. Seguindo o diagrama da Figura 29, o módulo montado é apresentado na Figura 40.

Figura 40 – Montagem do módulo de detecção do estado de porta



Fonte: elaborado pelo autor

4.5 Captação dos dados no Home Assistant

Para realizar o teste de funcionamento dos módulos, foi utilizado o software implementado na subseção 4.3.3. Através da utilização do comando SSH, os contêineres definidos no arquivo *docker-compose.yaml* foram iniciados e no processo de inicialização a plataforma Home Assistant se conecta ao servidor MQTT como cliente assinando os tópicos *sinteg/ldr*, *sinteg/temperature*, *sinteg/humidity* e *sinteg/reedswitch*.

Em seguida os módulos foram conectados a uma fonte e logo que a conexão com a internet e com o servidor foram estabelecidas, o envio de dados através dos tópicos se iniciou, assim a plataforma Home Assistant recebeu as mensagens publicadas com os dados e pôde disponibiliza-las na interface visual através de três cartões, um para o módulo de luminosidade (Figura 41), outro para o módulo de detecção do estado de porta (Figura 42) e outro para o módulo de temperatura e umidade (Figura 43).

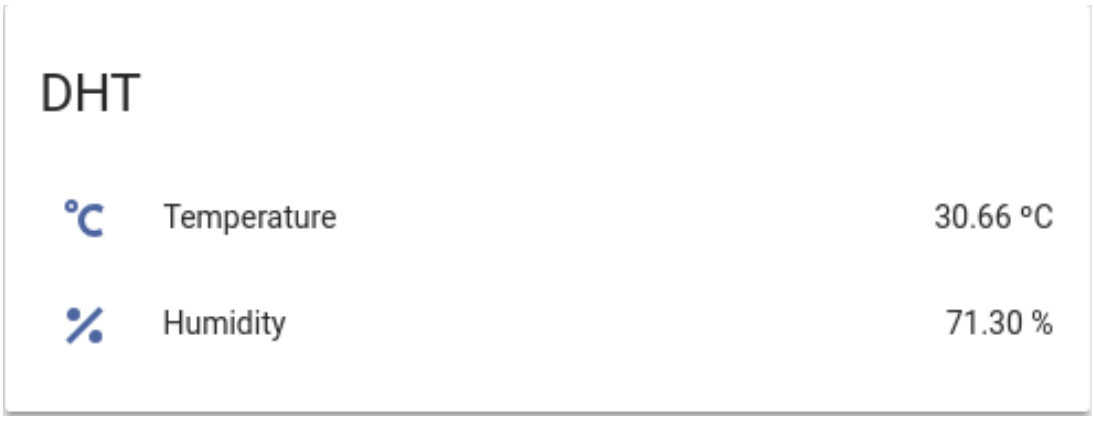
Figura 41 – Informações sobre luminosidade

Fonte: elaborado pelo autor

Figura 42 – Informações sobre estado da porta

Fonte: elaborado pelo autor

Figura 43 – Informações sobre temperatura e umidade



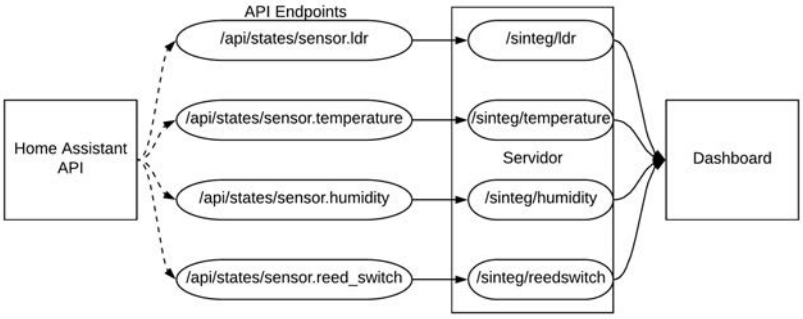
Fonte: elaborado pelo autor

4.6 Integração com Dashboard para Controle de Projetos

Para realizar a integração com um Dashboard para Controle de Projetos, foi elaborado um servidor utilizando a ferramenta Node.js e as bibliotecas axios e express (Figura 44). A construção do servidor foi realizada através da abertura da porta 4392 e a disponibilização de quatro rotas que ao receber uma requisição HTTP do tipo *GET*, é retornada uma resposta contendo os dados obtidos por meio da API disponibilizada pela aplicação Home Assistant.

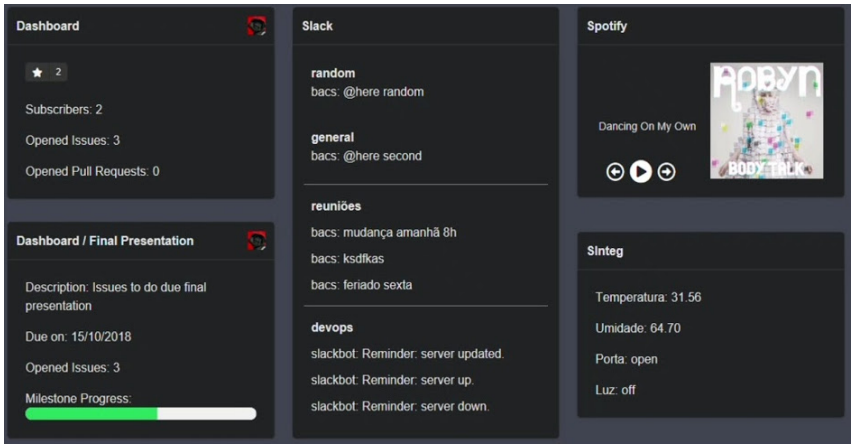
A preferência pela implementação de um servidor como intermediário entre a comunicação do Dashboard com a API do Home Assistant, foi motivada pela possível implementação de novas rotas que disponibilizem informações à aplicação final. O resultado final mostrado pelo software de controle de projeto pode ser visualizado na Figura 45.

Figura 44 – Diagrama de integração com Dashboard



Fonte: elaborado pelo autor

Figura 45 – Apresentação dos dados dos sensores no Dashboard



Fonte: elaborado pelo autor

5 CONCLUSÃO

Com o desenvolvimento desse trabalho foi realizado o aprofundamento e materialização do conceito de Internet das Coisas e a integração de diversas tecnologias recentes, como a plataforma Docker e o protocolo MQTT, resultando em três módulos IoT totalmente funcionais, atingindo os requisitos definidos na sessão dos objetivos.

A etapa de fundamentação teórica foi crucial para entender o funcionamento de cada ferramenta utilizada e também para realizar comparações entre os diversos componentes eletrônicos que seriam necessários, assim pôde-se escolher os elementos mais adequados ao projeto, sendo aqueles que possuíam maior disponibilidade e preços acessíveis para que a produção em larga escala de outros módulos possa vir a ser realizada.

A elaboração de diagramas para diversas etapas do projeto permitiu um desenvolvimento fluído e de fácil implementação, já que os requisitos foram bem definidos através das pesquisas realizadas e dos desenhos digitais. Eles também podem servir como base para a criação de projetos mais robustos e de sistemas maiores.

Uma breve comparação com o trabalho correlato citado na fundamentação teórica permite concluir que o trabalho desenvolvido não possui uma grande precisão em suas medições, porém, deve-se salientar que esse requisito não se faz necessário para os casos em que os módulos desenvolvidos se aplicam. Uma vez que o dispositivo desenvolvido no trabalho correlato apresenta a necessidade de utilização de outro aparelho para que seja feita a emissão dos dados e a dessas informação via internet, concluí-se que os módulos aqui desenvolvidos são de fácil elaboração e não necessitam um conhecimento prévio para que sejam colocados em prática.

6 TRABALHOS FUTUROS

Por apresentar a possibilidade de implementação em larga escala e sua montagem ser acessível a diversos públicos, os seguintes pontos podem ser aprimorados em trabalhos futuros:

a) automatizações integradas com outros componentes existentes no Home Assistant;
b) elaboração de um sistema de segurança para acesso as informações do servidor;
c) implementação de uma interface para o NodeMCU que permita a conexão com outra rede sem fio sem ter que reprogramá-lo;
d) elaboração de outros módulos utilizando o mesmo planejamento de software e hardware.

REFERÊNCIAS

ADAFRUIT. *DHT11, DHT22 and AM2302 Sensors*. 2012. Disponível em: <https://learn.adafruit.com/dht/overview>. Acesso em: 12 out. 2018.

ARDUINO. *Compare board specs*. 2012. Disponível em: <https://www.arduino.cc/en/products.compare>. Acesso em: 11 out. 2018.

ARTIK. *Home Assistant*. 2018. Disponível em: <https://developer.artik.io/documentation/developer-guide/wireless-iot/hass.html>. Acesso em: 13 out. 2018.

ASHTON, K. *That 'Internet of Things' Thing*. 2009. RFID Journal. Disponível em: <https://www.rfidjournal.com/articles/view?4986>. Acesso em: 10 out. 2018.

BAKER, J. *6 open source home automation tools*. 2017. Opensource. Disponível em: <https://opensource.com/tools/home-automation>. Acesso em: 13 out. 2018.

BJORLIN, C. *IoT Developers: Security, Data Collection Are Top Concerns*. 2018. IoT World Today. Disponível em: <https://www.iotworldtoday.com/2018/05/02/iot-developers-security-data-collection-are-top-concerns/>. Acesso em: 11 out. 2018.

CRAIG, W. *What Is 'Maker Culture,' And How Can You Put It To Work?* 2015. Forbes. Disponível em: <https://www.forbes.com/sites/williamcraig/2015/02/27/what-is-maker-culture-and-how-can-you-put-it-to-work/#11fa8c04540b>. Acesso em: 10 out. 2018.

DOCKER. *Get Started, Part 1: Orientation and setup*. 2018a. Disponível em: <https://docs.docker.com/get-started/>. Acesso em: 13 out. 2018.

DOCKER. *What is a Container*. 2018b. Disponível em: <https://www.docker.com/resources/what-container>. Acesso em: 13 out. 2018.

DUARTE, H. *O que são scripts; entenda para o que servem*. 2013. TechTudo. Disponível em: <https://www.techtudo.com.br/noticias/noticia/2013/12/o-que-sao-scripts-entenda-para-o-que-servem.html>. Acesso em: 13 out. 2018.

ECP. *Catálogo Segurança Eletrônica*. 20–. Disponível em: <http://www.ecp.com.br/download?file_name=\%2Fuploads\%2Fcatalogo\%2Ftranslation\%2Farquivo\%2F15\%2FCATALOGO\discretionary{-}{}{}SEG\discretionary{-}{}{}CFTV\discretionary{-}{}{}ECP_280617.pdf&locale=pt\discretionary{-}{}{}BR>. Acesso em: 12 out. 2018.

GIBBS, K. *Light dependent resistor (LDR) and thermistor*. 2009. Schoolphysics. Disponível em: <http://www.schoolphysics.co.uk/age14-16/glance/Electricity\%20and\%20magnetism/LDR_and_thermistor/index.html>. Acesso em: 12 out. 2018.

GOLDMAN SACHS. *What is the Internet of Things?* 2014. Disponível em: <http://www.goldmansachs.com/our-thinking/outlook/iot-infographic.html>. Acesso em: 10 out. 2018.

HUMIREL. *RELATIVE HUMIDITY SENSOR*. 2002. Disponível em: <https://www.parallax.com/sites/default/files/downloads/27920-Humidity-Sensor-Datasheet.pdf>. Acesso em: 12 out. 2018.

HUNT, C. Overview of tcp/ip. In: LOUKIDES, M.; CAMERON, D. (Ed.). *TCP/IP Network Administration*. 3. ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly & Associates, Inc., 1998. cap. 1, p. 1.

LUETH, K. L. *State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating*. 2018. IoT Analytics. Disponível em: <https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>. Acesso em: 10 out. 2018.

MARR, B. *Why The Internet Of Medical Things (IoMT) Will Start To Transform Healthcare In 2018*. 2018. Forbes. Disponível em: <https://www.forbes.com/sites/bernardmarr/2018/01/25/why-the-internet-of-medical-things-iomt-will-start-to-transform-healthcare-in-2018/#16eb416c4a3c>. Acesso em: 11 out. 2018.

MAXIM INTEGRATED. *DS18B20 - Programmable Resolution 1-Wire Digital Thermometer*. 2007. Disponível em: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>. Acesso em: 12 out. 2018.

MULESOFT. *What is a REST API?* 2016. Disponível em: <https://www.mulesoft.com/resources/api/what-is-rest-api-design>. Acesso em: 10 out. 2018.

MURTA, G. *Guia completo do NodeMCU – ESP12 – Introdução (1)*. 2018. Eletrogate Blog. Disponível em: <http://blog.eletrogate.com/nodemcu-esp12-introducao-1/>. Acesso em: 11 out. 2018.

NODEBR. *O que é Node.js?* 2016. Disponível em: <http://nodebr.com/o-que-e-node-js/>. Acesso em: 13 out. 2018.

PICKERINGRELAY. *Reed Relay Basics*. 2018. Disponível em: <https://www.pickeringrelay.com/introduction-reed-relay-basics-part-1/>. Acesso em: 12 out. 2018.

PLATFORMIO. *What is PlatformIO?* 2014. Disponível em: <https://docs.platformio.org/en/latest/what-is-platformio.html>. Acesso em: 13 out. 2018.

RASPBERRY PI. *What is a Raspberry Pi?* 2014. Disponível em: <https://www.raspberrypi.org/help/what-\%20is-a-raspberry-pi/>. Acesso em: 11 out. 2018.

RASPBIAN. *About Raspbian*. 2012. Disponível em: <https://www.raspbian.org/RaspbianAbout>. Acesso em: 11 out. 2018.

ROELOFSEN, P. The impact of office environments on employee performance: The design of the workplace as a strategy for productivity enhancement. *Journal of Facilities Management*, The Netherlands, n. 3, p. 247–264, 2002.

ROUSE, M. *clock speed*. 2005. What is. Disponível em: <https://whatis.techtarget.com/definition/clock-speed>. Acesso em: 11 out. 2018.

SATTI, J. A. et al. Miniaturized humidity and temperature sensing rfid enabled tags. *International Journal of Rf and Microwave Computer-Aided Engineering*, Pakistan, n. 1, p. n.p., 2018.

SSH. *SSH COMMAND*. 2018. Disponível em: <https://www.ssh.com/ssh/command/>. Acesso em: 15 out. 2018.

STATISTA. *Spending on Internet of Things worldwide by vertical in 2015 and 2020 (in billion U.S. dollars)*. 2018. Disponível em: <https://www.statista.com/statistics/666864/iot-spending-by-vertical-worldwide/>. Acesso em: 11 out. 2018.

TE CONNECTIVITY. *HTU21D(F) RH/T SENSOR IC*. 2012. Disponível em: <https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet\%7FHPC199_6\%7FA6\%7Fpdf\%7FEnglish\%7FENG_DS_HPC199_6_A6.pdf\%7FCAT-HSC0004>. Acesso em: 12 out. 2018.

TEXAS ADVANCED OPTOELECTRONIC SOLUTIONS INC. *TSL2560, TSL2561 LIGHT-TO-DIGITAL CONVERTER*. 2009. Disponível em: <https://cdn-shop.adafruit.com/datasheets/TSL2561.pdf>. Acesso em: 12 out. 2018.

TEXAS INSTRUMENTS. *LM35 Precision Centigrade Temperature Sensors*. 1999. Disponível em: <http://www.ti.com/lit/ds/symlink/lm35.pdf>. Acesso em: 12 out. 2018.

THIBODEAU, P. *Um em cada cinco desenvolvedores já trabalha em projetos de IoT*. 2015. ComputerWorld. Disponível em: <https://computerworld.com.br/2015/05/18/um-em-cada-cinco-desenvolvedores-ja-trabalham-em-projetos-de-iot/>. Acesso em: 11 out. 2018.

THOMSEN, A. *Comunicação Wireless com Módulo RF 433MHz*. 2013. FilipeFlop. Disponível em: <https://www.filipeflop.com/blog/modulo-rf-transmissor-receptor-433mhz-arduino/>. Acesso em: 12 out. 2018.

THOMSEN, A. *Como conectar o Arduino a uma rede Wi-Fi*. 2014. FilipeFlop. Disponível em: <https://www.filipeflop.com/blog/arduino-wifi-shield-cc3000/>. Acesso em: 11 out. 2018.

THOMSEN, A. *Tutorial Módulo Wireless ESP8266 com Arduino*. 2015. FilipeFlop. Disponível em: <https://www.filipeflop.com/blog/esp8266-arduino-tutorial/>. Acesso em: 11 out. 2018.

USINAINFO. *Módulo Sensor de Luminosidade Fotossensitivo LDR*. s.d. Disponível em: <https://www.usinainfo.com.br/sensor-de-luminosidade-arduino/modulo-sensor-de-luminosidade-fotossensitivo-ldr-2539.html>. Acesso em: 12 out. 2018.

WILSON, J. *Sensor Technology Handbook*. Elsevier Science, 2004. ISBN 9780080480848. Disponível em: <https://books.google.com.br/books?id=5UE6YCjDG-MC>. Acesso em: 10 out. 2018.

YUAN, M. *Getting to know MQTT*. 2017. IBM Developer. Disponível em: <https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/>. Acesso em: 11 out. 2018.