UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO TADEU MIRANDA CALABREZ

IMPLEMENTAÇÃO DE UMA ARQUITETURA IOT COM A FERRAMENTA NODE-RED

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2019

GUSTAVO TADEU MIRANDA CALABREZ

IMPLEMENTAÇÃO DE UMA ARQUITETURA IOT COM A FERRAMENTA NODE-RED

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Augusto Foronda

PONTA GROSSA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

TERMO DE APROVAÇÃO

IMPLEMENTAÇÃO DE UMA ARQUITETURA IOT COM A FERRAMENTA NODE-RED

GUSTAVO TADEU MIRANDA CALABREZ

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 31 de maio de 2019 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Augusto Foronda
Orientador

_____
Prof. Me. Geraldo Ranthum
Membro titular

_____
Prof. Dr. Diego Roberto Antunes
Membro titular

_____
Prof. Me. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. Me. Saulo Jorge Beltrão de Queiroz
Coordenador do curso

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

AGRADECIMENTOS

Agradeço principalmente a meus pais, pela oportunidade que me deram de estudar e buscar mais conhecimento.

Agradeço também as pessoas que acreditaram em mim e me deram chances, aos professores que me ajudaram dentro da faculdade, em especial o professor Augusto Foronda e a professora Sheila de Almeida.

Por fim, agradeço também a meus amigos e colegas que caminharam junto comigo desde o começo.

RESUMO

CALABREZ, Gustavo. **Implementação de uma Arquitetura IOT com a ferramenta Node-RED.** 2019. 49 f. Trabalho de Conclusão de Curso Bacharelado em Ciência da Computação - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2019.

Internet das coisas, também conhecido como *Internet of Things* ou *IoT*, é um conceito no qual diversos objetos são conectados por meio da Internet, de forma que possam se comunicar entre si. Uma arquitetura *IoT* envolve diversas áreas, tais como sensores, atuadores, sistemas embarcados, armazenamento de dados em nuvem. Quando começa a programação de um projeto desse tipo, é comum se deparar com situações onde temos muitos dados e tende a se tornar um pouco confusa a visualização de um projeto. Com a utilização da ferramenta Node-RED, temos uma IDE que permite reutilizar o código de maneira simples e mostrar através de um ambiente visual e funcional o projeto. Esse trabalho implementa uma arquitetura *IoT*, usando a ferramenta Node-RED, com o objetivo de simular um problema de segurança domiciliar.

Palavras-chave: *Internet of Things*. Node-RED. Node-MCU.

ABSTRACT

CALABREZ, Gustavo. **Implementation of IOT Architecture with Node-RED.** 2019. 49 p. Work of Conclusion Course (Graduation in Computer Science) - Federal Technology University - Paraná. Ponta Grossa, 2019.

Internet of Things is a recent concept in which we have an objective, and through a series of diferent agents like sensors, computers, softwares and microcontrolers, all of which are communicating with themselves in a network, can reach this goal. An IoT project can use all sorts of technologies in it, such as concepts of from eletronics, computer networks, automation. Its fundamental for us to understand the real benefits of a technology that has everything to keep growing, to understand how does it apply, more specifically in a Node-RED environment, explaining each step by step.

Keywords: *Internet of Things*. Node-RED. Node-MCU.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

IBM	*International Business Machines*
IDE	*Integrated Development Environment*
IOT	*Internet Of Things*
IP	*Internet Protocol*
JSON	*Javascript Object Notation*
LED	*Light Emitting Diode*
MQTT	*Message Queuing Telemetry Transport*
PIR	*Passive Infrared Sensor*
QOS	*Quality Of Service*
SSID	*Service Set Identifier*
TCP/IP	*Transmission Control Protocol / Internet Protocol*
UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO

Existe uma previsão para o crescimento de mercado para IoT de 2016 até 2020, passando dos 157 bilhões de dólares para 457 bilhões de dólares, um crescimento de 28,5%, o que mostra que é uma tecnologia recente, porém que tem um potencial de crescimento e aplicabilidade no mercado, tanto para o consumidor, como para ser utilizado pelas próprias empresas em seus maquinários (GROWTHENABLER, 2016).

A Internet das Coisas, também conhecida como *Internet of Things*, ou *IoT*, consiste em usar a Internet para a comunicação entre objetos, onde eles são capazes de interagir e de cooperar entre si para alcançar um objetivo (SINGH, 2014). Essa tecnologia pode ser aplicada em diferentes ambientes e aspectos, desde controlar um maquinário em uma fábrica até controlar a segurança de uma sala pelo celular, possibilitando diversas implementações, utilizando-se da Internet para se comunicar com objetos, enviando e recebendo dados. A ideia é ter um sistema interligado entre dois servidores desktop com ferramentas diferentes e uma placa microcontrolada, que recebe e envia informações através de uma rede para realizar uma determinada função.

Esse projeto é principalmente controlado através de uma placa microcontroladora NodeMCU, também conhecida como ESP32-F, a qual possui uma memória interna em que deve se inserir um código para que ela solucione um determinado tipo de problema. Nessa placa, além de inserir o código na memória, em determinadas aplicações, se tem a necessidade de medir algum tipo de parâmetro: temperatura, pressão, umidade. Para que isso seja captado de forma eficaz, devemos conectar os sensores de cada tipo de medida diretamente na placa e desenvolver o código para que os dados captados sejam recebidos da forma correta

Uma característica importante em ambientes de *IoT*, é que o projeto pode ser de baixo custo, fator que está diretamente ligado na capacidade de processamento de dados, armazenamento e comunicação. Por outro lado, como o custo dessas placas e sensores acaba sendo baixo, isso permite a criação de projetos grandes, com a utilização de milhares de placas e sensores, dependendo do projeto.

Em busca de uma comunicação efetiva entre esses dispositivos, os componentes da IoT devem lidar com fatores como instabilidade de comunicação, alta latência e baixa largura de banda. Para que essas placas possam se comunicar entre si, ou com diferentes usuários, é necessário um protocolo de comunicação em rede capaz de enfrentar todos os problemas citados acima. Existem diversos protocolos que podem ser utilizados. O protocolo MQTT (*Message Queuing Telemetry Transport*) é um dos principais protocolos quando se trata de Internet das Coisas, seu funcionamento é baseado em *brokers* que são um tipo centralizado de servidor, possuindo uma alta quantidade de implementações dos *brokers* em diferentes linguagens. Os *brokers* funcionam recebendo e enviando pacotes de clientes, neste caso, a placa NodeMCU e os servidores desktop. Esses clientes podem atuar como *Publisher* ou *Subscriber*, enviando ou recebendo determinados dados para/do broker. Com este protocolo é possível que diferentes placas se comuniquem entre si, ou até mesmo que se comuniquem com outros periféricos que podem ser utilizados em um projeto de Internet das Coisas, como um computador ou *smartphone*.

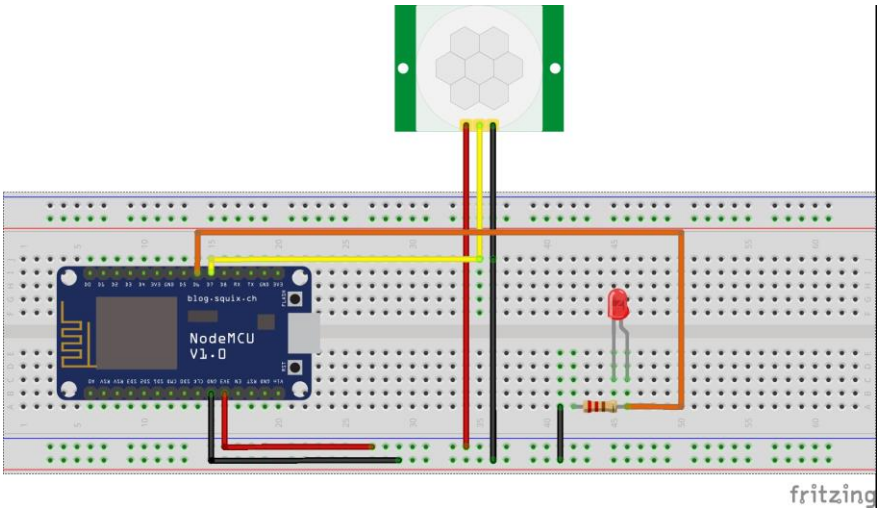
Exemplos de placas e microcomputadores que são altamente utilizadas no mercado de *IoT* são o *Raspberry Pi*, Arduino Uno e Beaglebone Black. São microcontroladores/microcomputadores que podem ser adaptados para diversas funções e projetos. Dentre muitos, a placa Node-MCU é uma que se destaca pelo seu baixo custo e bom desempenho, sendo uma placa com um ótimo custo-benefício, motivo pelo qual foi utilizada neste projeto. Para projetos que usam poucos sensores e precisam enviar pouca informação de cada vez, que é o caso deste projeto, a placa se adequa.

Levando em consideração a implementação de um projeto *IoT* que utiliza um protocolo como o *MQTT*, diversas partes do código acabam se tornando repetitivas: diversos *publish/subscribe*, salvar e carregar dados. Levando isso em consideração, desenvolvedores da IBM resolveram sanar este problema através da ferramenta Node-RED, que permite reutilização de código, em uma IDE *drag-and-drop*, o que torna mais fácil a criação de código. A ferramenta consiste em uma visualização e desenvolvimento de eventos em forma de *nodes*, que são blocos que cada um tem uma determinada função que pode ser codificada conforme requisitado. Ainda precisa codificar, mas a quantidade de código é reduzida de forma significante.

Para que se possa compreender o funcionamento da ferramenta Node-RED e o protocolo MQTT, será proposta uma solução para um problema de segurança domiciliar. Como o contexto da Internet das Coisas nada mais é do que conectividade, acaba se tornando interessante interligar diferentes aplicações. A ideia do projeto é de que quando houver uma alteração captada pela placa microcontroladora por um sensor de movimento, ela emite um sinal por meio de uma rede social, neste caso, o Twitter. O Twitter é uma rede social que permite que o usuário publique um texto de até 280 caracteres. Essa publicação será feita inteiramente pela placa microcontroladora, sem a utilização de um computador ou smartphone.

A Figura 1 mostra como foi desenvolvida a parte física do projeto, conectando o sensor e o LED a placa NodeMCU. A placa é responsável pelo envio e recebimento de pacotes os quais o fluxo é controlado pelo *broker*. O sensor PIR detecta o sinal de movimento na área e o LED é responsável apenas para demonstrar a ativação de alguma ação, como um alarme.

Figura 1 - Projeto de Segurança: Parte Física



Fonte: Autoria própria

A proposta deste trabalho é implementar uma aplicação de um projeto para a Internet das Coisas utilizando o protocolo MQTT e a plataforma de desenvolvimento Node-RED, verificando quais são suas dificuldades e vantagens

em implementação do projeto, trazendo uma solução para um problema de segurança domiciliar.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver um projeto *IoT* com o uso da ferramenta Node-RED e protocolo MQTT, voltado para a segurança domiciliar.

1.1.2 Objetivo Específico

Os objetivos específicos deste trabalho são descritos a seguir:

- Implementar o servidor MQTT;
- Implementar servidor Node-RED;
- Configurar placa NodeMCU, bem como os sensores que serão utilizados no projeto;
- Implementação de código para comunicação da placa com Node-RED e vice-versa.

1.2 JUSTIFICATIVA

Com o aumento da economia gerada pelo mercado de Internet das Coisas, tem-se uma grande diversidade de empresas desenvolvendo aplicações, ferramentas e placas para serem usadas tanto pelos desenvolvedores, quanto para o público que consome essa tecnologia (GROWTHENABLER, 2016). Dessa forma, pode-se desenvolver o mesmo projeto, com diferentes ferramentas e tecnologias.

Pelo fato de a Internet das Coisas ser uma tecnologia recente, a qual muitos estão utilizando, existem diversas implementações de suas funcionalidades, como protocolos, placas microcontroladoras e projetos. Isso mostra uma disponibilidade

para analisar como funciona uma arquitetura de um projeto *IoT*, como ela deve ser modelada e por fim, aplicada para determinada função.

1.3 ORGANIZAÇÃO DO TRABALHO

No capítulo 1, foi feita a introdução ao tema, abordando de forma breve tópicos relacionados a tecnologias e ferramentas que serão essências para o entendimento do projeto.

No capítulo 2, foi mostrada a literatura utilizada para obter o conhecimento sobre o assunto da Internet das Coisas, explicando ferramentas básicas de eletrônica (sensores e atuadores), um pouco sobre a placa microcontroladora e a IDE utilizada para programa-la. É demonstrado também o protocolo de comunicação utilizado, o MQTT, tal como a ferramenta para desenvolvimento do projeto, o Node-RED.

No capítulo 3, o desenvolvimento do projeto foi explicado, começando pela instalação do *broker* no Windows. Em seguida, a configuração do NodeMCU e a programação para que ele se comunique com o Node-RED. Por fim, é demonstrado como foi configurada a ferramenta Node-RED para que pudesse se comunicar com a placa.

No capítulo 4, conclui-se o trabalho e no capítulo 5 são apresentados trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo traz a história da *IoT*, conceitos gerais que são importantes para o entendimento do que é uma arquitetura *IoT* como: definição, componentes, utilização. Além disso, são apresentadas algumas ferramentas que foram utilizadas neste trabalho: placa microcontroladora Node-MCU, protocolo de rede MQTT e a ferramenta Node-RED.

2.1 IOT: A CRIAÇÃO DE UM CONCEITO HISTÓRICO

Recentemente a Internet das Coisas tem dominado o cenário da tecnologia. É um conceito de interligar objetos e funções, mas pouco se sabe da origem deste conceito o qual está sendo muito difundido no século XXI. O primeiro relato conhecido de um conceito de conectividade foi em 1832 pelo Pavel Schilling, o inventor do telégrafo eletrônico, o qual tinha como função enviar mensagens distantes de um ponto a outro.

Com o passar dos anos e a evolução da eletricidade, Nikola Tesla em 1926 provou a possibilidade da transmissão sem fio de eletricidade, criando o conceito *wireless*, sem a utilização de condutores sólidos para obter uma carga elétrica. Na época, Tesla percebeu o potencial que tinha esta tecnologia e sabia que isso seria utilizado no mundo todo: "Quando a tecnologia sem fio estiver aplicada perfeitamente, o mundo inteiro vai se transformar em um grande cérebro...e os instrumentos os quais farão possível essas comunicações serão extremamente simples e muito semelhantes a o nosso atual telefone. Um homem poderá carregar um em seu bolso".

Considerado o primeiro dispositivo IoT, uma torradeira criada por John Romkey, que poderia ser ligada e desligada através de um protocolo de Internet TCP/IP. Trazendo assim diferentes conceitos no projeto:

- Distância: Não precisava estar fisicamente perto da torradeira para liga-la;
- Eletricidade: Conduzir energia para a torradeira;

- Internet: Passar a informação de quando ligar/desligar a torradeira através do protocolo TCP/IP.

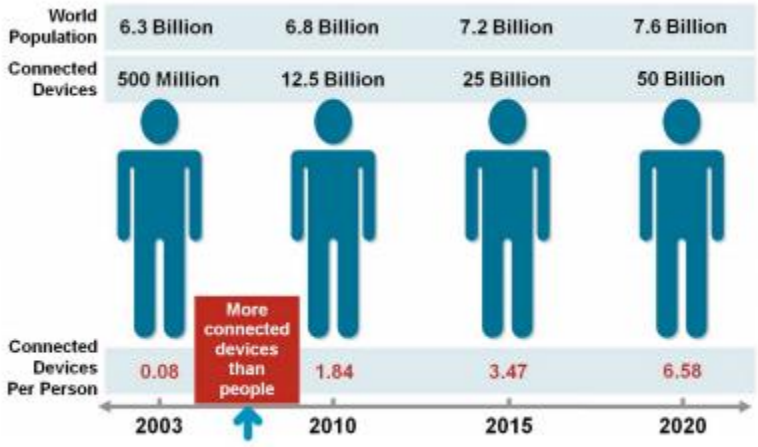
Historicamente, o conceito da Internet das Coisas existe desde 1832, mas houve um ponto em que marca o real nascimento da IoT. De acordo com um estudo feito pela *Cisco Internet Business Solutions Group*, a Internet das Coisas nasceu entre 2008 e 2009, pelo motivo de que existiam mais objetos que permitiam essa interconectividade do que pessoas propriamente ditas (CISCO IBSG, 2011).

Segundo SANTAELLA (2008), estar conectado à rede não diz respeito somente à interação entre dados e pessoas, mas sim de mais um terceiro elemento: as coisas. Poder utilizar diferentes objetos para se comunicar é o diferencial desta tecnologia.

IoT se trata de uma extensão do conceito de Internet, que é inserida nos objetos e somada aos sensores ou atuadores, permitindo que eles se comuniquem e interajam com o ambiente no qual estão inseridos. Através destes objetos é possível "detectar seu contexto, controla-lo, viabilizar troca de informações uns com os outros, acessar serviços da Internet e interagir com pessoas" (SANTOS et al, 2016).

Os dados da Figura 2 mostram que esse crescimento de objetos conectados por pessoa praticamente dobra a cada 5 anos. Atualmente, mesmo que não percebendo, cada ser humano presente na sociedade tem em média 4 dispositivos presentes em sua volta. Este número em 2010, atingiu um ponto interessante, em que o número de dispositivos acaba sendo maior do que o de pessoas no mundo inteiro.

Figura 2 - Nascimento da IOT



Fonte: (Cisco IBSG, 2011)

Finalizando a retomada histórica da *IoT*, em 2011 a plataforma Arduino e outras presentes no mercado acabaram amadurecendo suas ideias e trazendo para o mercado hardwares que tornavam a *IoT* acessível para pessoas que tinham interesse em desenvolver seus projetos e até mesmo para as grandes empresas.

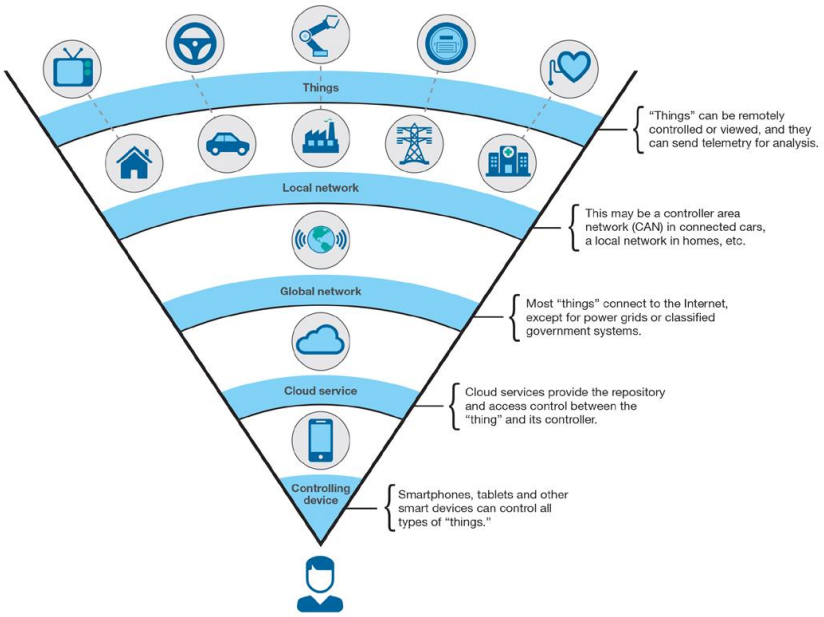
2.2 ARQUITETURA IOT

Uma arquitetura *IoT* deve ser tratada como um sistema que pode ser físico, virtual ou um híbrido dos dois, consistindo em uma coleção de inúmeros ativos físicos, sensores, atuadores, serviços na nuvem, protocolos específicos para *IoT*, camadas de comunicação, usuários, desenvolvedores e camada empresarial (Journal of King Saud University, 2018).

Cada bloco de uma arquitetura *IoT* é responsável por algo especifico, tais como verificar os sensores, identificar padrões, agir sobre determinada ação, passar informações entre dispositivos, receber informações. A Figura 3 descreve uma arquitetura *IoT* como:

- Coisas: as coisas que podem ser controladas ou visualizadas remotamente. Podem enviar dados para análise;
- Rede local: rede a qual as coisas estão interagindo entre si;
- Rede global: rede a qual as coisas conectam à Internet;
- Serviço na nuvem: repositório e local de acesso entre as coisas e o controlador;
- Controlador: objeto utilizado para controlar as coisas. Pode ser um Smartphone, tablet ou qualquer device inteligente.

Figura 3 - Arquitetura IoT



Fonte: (X-Force Research and Development, 2014)

A arquitetura de um projeto *IoT* descreve a sua solução para o problema. Não existe uma só arquitetura, existem diversas arquiteturas que podem ser utilizadas e adaptadas dependendo do seu problema (KERTÉSZ, 2016). Adotar uma arquitetura multicamadas nos permite conhecer os aspectos mais importantes do projeto antes mesmo de integrarmos todos entre si. Essa aplicação ajuda em organizar a complexidade de uma solução *IoT*.

Um exemplo dessa arquitetura, seria em um caso em que se têm diversos dados que serão analisados, ou seja, é uma aplicação movida a dados. Esse tipo de aplicação em que se envolve muita análise de dados, uma arquitetura básica dividida em três camadas consegue modelar o problema de forma eficaz.

A Figura 4 mostra as três divisões: *Cloud, Edge e Devices*. Cada divisão é responsável por um tipo de análise e uma atuação diferente:

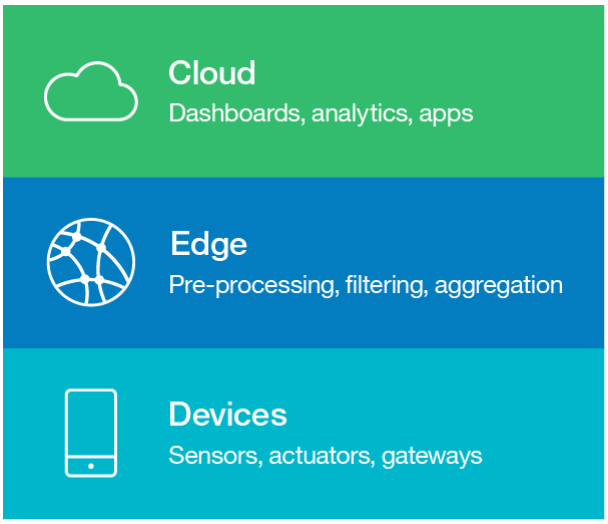
- Devices: nada mais são que os sensores e atuadores que estão presentes e conectados em nossa aplicação. Eles têm como função enviar um determinado sinal, que corresponde a algo: movimento, temperatura, humidade. Os sensores por si só, não fazem nada. Isso se deve ao fato deles não serem inteligentes, todavia eles sempre estarão conectados a algum tipo de processador, seja via wireless ou conectados. Existem alguns dispositivos no mercado que não se comunicam com um desses processadores de dados, mas sim diretamente com a nuvem, já tratando o dado em questão e

armazenando-o. Resumidamente, os dispositivos são os sensores do projeto que mostram os dados de uma forma bruta.

- Edge: A camada Edge é onde se relacionam os dados da camada de Devices com a camada Cloud. Aqui o papel é processar e analisar os dados, enviando para a nuvem apenas os dados considerados chaves para o projeto. Essa análise é feita em tempo real.

- Cloud: Depois que o dado é trabalhado, ele é enviado para a camada Cloud, onde a informação é processada, salva e pode ser utilizada em aplicações na nuvem (KERTÉSZ, 2016). As aplicações que analisam em tempo real dados, trabalham justamente com esse tipo de dado. Eles recebem a informação, tratam ela de uma maneira que possa ser utilizada, e mostram isso para o usuário por meio de um app, tendo uma visualização simplificada e de fácil entendimento.

Figura 4 - Modelo de Arquitetura IoT em 3 camadas



Fonte: (IBM Developer, 2017)

É importante ressaltar que não existe apenas este modelo para uma arquitetura *IoT*, existem diversas formas de se modelar um problema, as vezes com mais ou até mesmo menos camadas. Tudo depende do problema em questão, dos sensores e até mesmo das limitações da placa que será utilizada.

Essa é a arquitetura que foi usada no trabalho, por permitir uma divisão sólida nas camadas podendo assim controlar de uma maneira concreta a aplicação como um todo.

2.2.1 Sensores e atuadores

Os projetos para Internet das Coisas utilizam em sua grande maioria sensores e atuadores para que se possam captar e utilizar os dados. Um sensor é um dispositivo que é capaz de mensurar algum fenômeno físico como temperatura e transformar esse fenômeno em um sinal elétrico (WENDLING, 2010). É importante entender que não existe apenas um sensor para todos fenômenos, pois para um sensor conseguir captar de forma eficaz o sinal, ele deve ser sensível a esse tipo de sinal que ele está querendo medir, e não a mais nenhum outro tipo, para que não exista interferência na medição. Os sensores devem trabalhar com diversas instabilidades e fatores que devem ser levados em consideração:

- Distância: Quão distante está o sensor do fenômeno o qual ele irá analisar, isso pode afetar o seu desempenho.
- Sensibilidade: Qual deve ser a intensidade mínima para ser utilizada como parâmetro para que o sensor faça efeito. Por exemplo, um mesmo sensor de presença pode ser utilizado para medir sinais curtos e longos. É necessário configurar qual deve ser a distância mínima a qual o sensor deve ser capaz de captar. O fator distância e sensibilidade estão diretamente relacionados.

Os sensores possuem algumas classificações:

- Ativo ou Passivo: Sensores passivos sempre irão analisar algum fator do ambiente, sem necessidade de algum tipo de alimentação externa (WENDLING, 2010). Sensores ativos analisam algum tipo de dado que é gerado através de uma fonte externa que utiliza energia; Exemplos:
- Ativo: Sensor de temperatura, sensor giroscópio;
- Passivo: Sensor de movimento.
- Tipo de dado: O dado que será analisado pode ser um dado mecânico, químico. Também deve ser considerado o tipo de dado que será gerado através do sensor, podendo ser um sinal analógico, digital ou até mesmo alternando entre os dois (WENDLING, 2010).

A Figura 5 mostra o sensor foi utilizado no projeto, que é o sensor de movimento PIR (*Passive Infrared Sensor*), um sensor passivo que tem como função medir luz infravermelha que irradia dos objetos em seu campo de visão, comumente

utilizado para projetos em que se tem a necessidade de verificar presença em determinada região do projeto.

Figura 5 - Sensor de movimento PIR

Fonte: Autoria Própria

Para que os sensores possam enviar os dados para a unidade de processamento, que no nosso caso é uma placa microcontroladora, ele deve estar conectado de alguma forma a essa placa. A forma a qual os sensores interagem com um microcontrolador é descrita na sessão a seguir.

2.2.2 Placa Microcontroladora: Node-MCU

A placa microcontroladora é capaz de processar os dados capturados pelos sensores, e também configurar como eles devem fazer isso, frequência, intensidade, e o que fazer com esse sinal gerado. "O microcontrolador é um pequeno computador dentro de um chip" (MONK, 2013).

Para que os sensores possam se conectar à placa, o microcontrolador deve permitir essa conexão de alguma forma, que é disponibilizando um pino de conexão.

Cada pino tem uma função dentro do microcontrolador, alguns servem para gerar energia, outros para receber/enviar sinais, outro para aterramento. Conceitos de elétrica/eletrônica não serão aprofundados neste trabalho, porém o que deve ser dito é que existem pinos para que o sensor possa se comunicar com a placa de uma forma segura e eficaz.

A placa que será utilizada no trabalho é a Node-MCU, uma placa que possui 6 pinos analógicos e 14 pinos digitais. Para que esse pino possa enviar seu sinal

para a placa ele deve ser configurado levando em consideração o que será conectado a ele. Se for um sensor, o pino deve ser conectado para receber sinais. Se for um LED por exemplo, ele deve ser configurado para enviar sinais, já que o LED só tem como função acender. A Tabela 1 apresenta as características da placa como: alimentação, memória, voltagem, número de pinos e informações que são relevantes quando se busca uma placa para um projeto. Essas informações são interessantes pelo fato de que cada sensor/atuador necessita de um pino para poder se conectar a placa.

Tabela 1 – Informações sobre a Placa (*Data sheet*)

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20mA
DC Current for 3.3V Pin	20mA
Flash Memory	32 KB
SRAM	2KB
EEPROM	1KB
Clock Speed	16MHz
LED BUILTIN	13
Lenght	68.6mm
Width	53.4mm
Weight	25g

Fonte: (Adafruit, 2015)

2.2.3 Flash no Microcontrolador

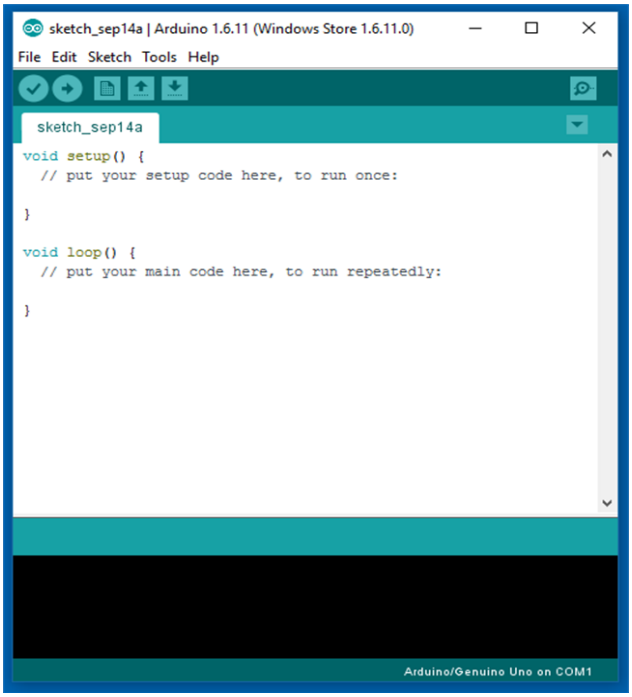
Para que se possa configurar esses pinos, a empresa Arduino disponibiliza uma IDE para que o programador desenvolva o código e em seguida transfira para a memória interna da placa para que testes possam ser efetuados. A IDE permite que assim que a placa seja conectada e o programa for inicializado *drivers* sejam instalados para que a placa possa funcionar da maneira correta, desde que os drivers estejam na pasta apropriada, facilitando assim a utilização pelos usuários.

Funções e ações podem ser desenvolvidas através do programa, que permite selecionar qual porta está conectada a placa em questão no computador para que o arquivo possa ser enviado para a memória interna. (MONK, 2013). Isso se deve ao fato de que se pode conectar diversas placas em um mesmo computador, e fazer a edição e envio de diversos programas diferentes para as placas. A IDE também sinaliza quando esta etapa de envio do código para a placa foi feita correta ou incorretamente.

Dentre diversas opções, é possível também selecionar algum programa pré-programado para ser enviado à memória para que possam ser feitos testes, economizando tempo e permitindo identificar um possível erro de forma rápida.

A Figura 6 mostra a tela inicial da IDE no momento que é inicializada pela primeira vez, mostrando os dois laços de repetição primários: *setup e loop.*

Figura 6 - IDE do Arduino



Fonte: Autoria Própria

O laço de repetição *"setup"* é a primeira iteração do programa. Nesta função devem ser colocadas as variáveis e as definições da placa, como pinos que serão utilizados como sensores ou atuadores. Este laço é executado apenas uma vez (JAUOD, 2015).

O laço de repetição *“loop”* funciona de maneira parecida, porém ele não para após a primeira iteração. Quando o “*setup”* termina de executar as instruções, o “*loop”* executa em sequência de forma continua, repetindo os códigos que estiverem dentro das chaves. Por ser uma repetição infinita, o programador deve tomar cuidado ao desenvolver o seu código para que não se tenham laços de repetição infinitos. Existe um dispositivo dentro da placa chamado *Watchdog Timer* que foi desenvolvido para impedir que esse tipo de travamento aconteça. Ele identifica quando existe esse tipo de travamento e reinicia automaticamente o microcontrolado.

É interessante informar que essa IDE não está limitada somente a placas da empresa Arduino. Ela permite a implementação de código para qualquer microcontrolado.

Para que essas placas possam se comunicar, é necessária a utilização ode algum protocolo de comunicação.

2.3 PROTOCOLO DE COMUNICAÇÃO

Uma das características fundamentais da IoT é a comunicação entre diferentes objetos. O protocolo de comunicação é o que permite interligar as coisas dentro do projeto. Ele tem como principal função criar o caminho para que se possa enviar e receber dados.

Um protocolo estabelece as regras que devem ser seguidas para se ter a comunicação entre dois pontos. Existem diferentes regras para diferentes processos, por exemplo, fazer um *broadcast* em uma rede. O que cada processo deve levar em consideração: tipo da tarefa que vai ser executada, o tipo de dado e o dispositivo que transportará esse dado.

Um único processo pode ser manipulado por mais de um protocolo diferente de forma simultânea, não dependendo apenas de uma única tecnologia. Um exemplo hipotético seria um caso em que temos um sensor que envia o sinal de recebimento para o *broker* via MQTT, e este *broker* se comunica com uma outra aplicação via protocolo *UDP*. Dois protocolos diferentes que fazem um único processo de envio de informações.

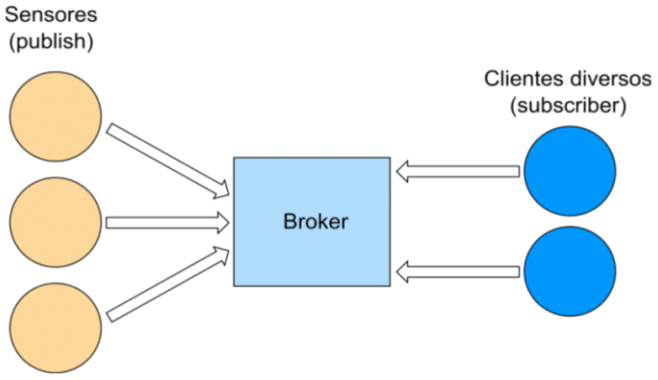
2.3.1 MQTT

O protocolo que será utilizado neste trabalho é o MQTT *(Message Queuing Telemetry Transport*). É baseado em *publish* e *subscribe*, tem seu funcionamento simples e envio de mensagens de uma forma leve (VALERA, 2014). O MQTT foi desenvolvido pensando justamente em uma rede com baixa largura de banda, alta latência e não confiável. Segundo o Dr. Andy Stanford-Clark, da IBM: "Os princípios do MQTT são minimizar a largura de banda da rede e quantidade de recursos, enquanto também tenta assegurar uma confiabilidade e um certo nível de segurança de entrega" (IBM DEVELOPER, 2017).

A ideia do MQTT é ter um *middleman*, um intermédio, entre os dados que estão sendo recebidos, e as coisas que querem esses dados. O que fica neste meio é o que será conhecido como *broker* e ele intermediará as informações que são recebidas e enviará para quem tiver este interesse (National University of Singapore, 2014). Quem enviará esses dados captados são os sensores, que para isso, devem se inscrever na rede deste *broker*, efetuando o papel de um *publisher,* pois estão enviando e não recebendo as informações. Da mesma forma, quem quer receber esses dados, deve se inscrever neste *broker*, mas por outro lado, deve assumir o papel de um *subscriber*, pois quer receber e não enviar os dados.

A Figura 7 demonstra o funcionamento do MQTT. Tem-se um broker efetuando esta passagem entre os 3 sensores em *publish* e os 2 clientes em *subscribe*.

Figura 7- Conceito de Publish/Subscribe



Fonte: Autoria Própria

A fragilidade deste modelo se dá pelo fato da concentração de informações em um único lugar, o *broker*. Esse aspecto existe, e pode ser contornado por meio de diferentes implementações, *brokers* redundantes ou a utilização de *clusters*. De qualquer forma, essa concentração permite que as partes que irão se comunicar, não estejam em um mesmo lugar, podendo estar desacopladas. Fazendo um paralelo com outros modelos de comunicação, isso não é possível, por exemplo, no modelo cliente/servidor.

A conexão de um *broker* com ambas as partes é feita via protocolo *TCP* (*Transmission Control Protoco*l), ou Protocolo de Controle de Transmissão, utilizando-se de criptografia no envio e recebimento de mensagens (KHALIL, 2018). Toda essa conexão segue uma métrica de qualidade de envio de dados, conhecida como QoS (*Quality of Service*), com diferentes níveis para diferentes desejos para a aplicação:

- QoS 0: best effort, ou seja, melhor esforço. Não se dá confirmação de entrega de mensagem, o envio não tem segurança de entrega e não permanece com a mensagem via de erro. Muito parecido com o protocolo UDP;
- QoS 1: existe uma confirmação de entrega de pelo menos uma mensagem e existe um armazenamento da mensagem por parte de quem enviou, em caso de falha no envio;
- QoS 2: tem a garantia de entrega de exatamente uma mensagem, de confirmações de recebimento, e de confirmações dessas confirmações. Quando uma mensagem não é confirmada, ela é mantida por quem está querendo enviar até que se receba uma confirmação. Similar ao protocolo TCP.

Como dito, existem diversas implementações dos brokers. A que foi utilizada neste projeto foi o mosquitto, atualmente a implementação com mais aceitação e utilização em plataformas *IoT* (GROWTHENABLER,2017).

Os conceitos sobre implementação do protocolo MQTT serão discutidos nas próximas seções nos capítulos referentes a utilização do protocolo na aplicação do projeto

2.3.2 Node-RED

O Node-RED é uma ferramenta desenvolvida pela IBM, podendo ser instalado tanto em desktops como microcomputadores, baseada no conceito de

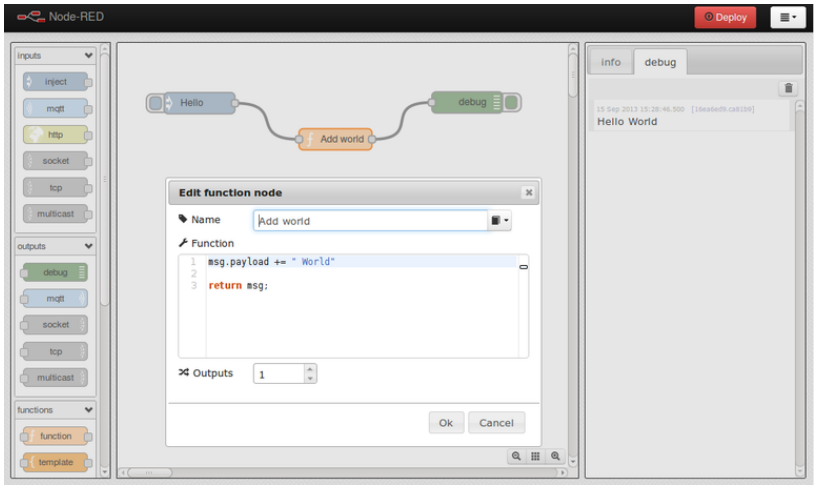
fluxo de dados, onde as mensagens são passadas por meio de caminhos pré-definidos já estruturados. Consiste em uma programação mais visual, desenvolvida para unir dispositivos de hardware, *API's* e serviços da Internet, utilizando-se dos conceitos de IoT de forma integral (AGARWAL, 2017). A ferramenta é desenvolvida através de funções em *Javascript*, com as informações sendo salvas em JSON.

Consiste na utilização de *nodes*, que são blocos de código que podem ser *inputs* ou *outputs*, ou seja, eles enviam ou recebem dados. Cada *node* é desenvolvido de uma forma genérica, para que possa reutilizar na aplicação. Um *node* recebe/envia um tipo de dado, faz a função dele, e acaba o fluxo de dados ou envia para frente.

A Figura 8 apresenta um exemplo da aplicação, de maneira simplificada, dos nodes. Cada bloco arredondado é um node, que é uma representação de um código em Javascript desenvolvido para alguma finalidade. Neste caso tem-se três nodes:

- *Hello: node responsável por enviar uma mensagem. É conhecido como Injection Node.*
- *Add world: tem o papel de unir a mensagem que está sendo recebida com a mensagem de texto "World". É um function node.*
- *Debug: node responsável por mostrar a mensagem ao usuário. É conhecido como debug node.*

Figura 8 - Função de "Hello World" no Node-RED



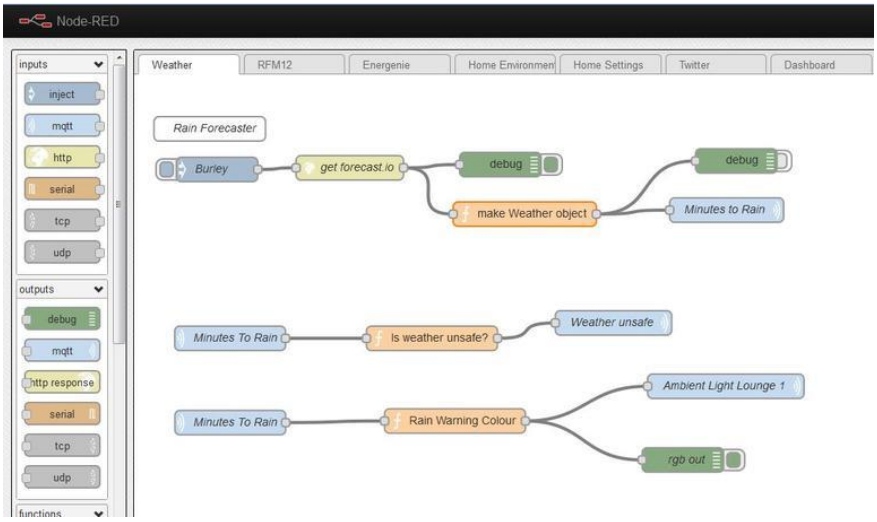
Fonte: Documentação do Node-RED

Este exemplo serve para entender que o Node-RED consegue unir diversas propostas de maneira simples. Em uma aplicação *IoT*, esses *nodes* que neste caso

enviam uma mensagem, poderiam enviar um sinal, que poderia ser transmitido para uma aplicação web. Tudo isso controlado e "colado" por meio da ferramenta Node-RED, de maneira visual e simples.

A Figura 9 mostra um exemplo desenvolvido pela IBM, em que podemos ver os dois tipos de *nodes*, *input* e *output*, sendo utilizados. A ideia deste projeto é desenvolver um medidor de clima, para dizer se é seguro ou não deixar câmeras fotográficas em exposição climática. Os dados climáticos são recebidos através de uma ferramenta meteorológica online, onde os dados são guardados em um objeto para fins de comparação. Se este objeto apresentar dados que não possibilitam a exposição da câmera, uma luz acenderá. A função que acende a luz é desenvolvida através de um comando enviado via MQTT pela placa microcontroladora

Figura 9 - Exemplo aplicação de medição do clima no Node-RED



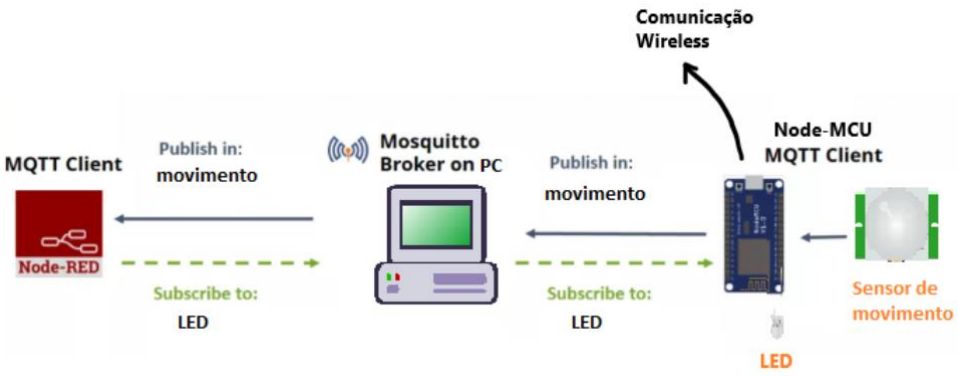
Fonte: Documentação do Node-RED

O conceito de fluxo de dados em uma aplicação IoT é muito importante pelo seguinte fato: "Operações em um fluxo de dados podem funcionar assim que o dado for recebido, não compartilham nenhum estado entre si, operações são inerentemente paralelas e podem funcionar de forma independente" (Ackerman, 1982).

3 DESENVOLVIMENTO

Neste capítulo, demonstra-se a aplicação prática do trabalho, iniciando-se na parte de configuração do *broker MQTT* em um computador com Windows. Conforme demonstra a Figura 10, o projeto consiste em um sensor de movimento acoplado na placa Node-MCU, que envia via wireless o sinal para o *broker* mosquitto que fica em um computador. Esse *broker* se comunica diretamente com o cliente *MQTT* do Node-RED, permitindo enviar e receber informações do *broker* no PC, permitindo dessa forma ligar o LED que está acoplado na placa.

Figura 10 - Topologia do Projeto



Fonte: Autoria Própria

3.1 INSTALAÇÃO DO BROKER NO WINDOWS

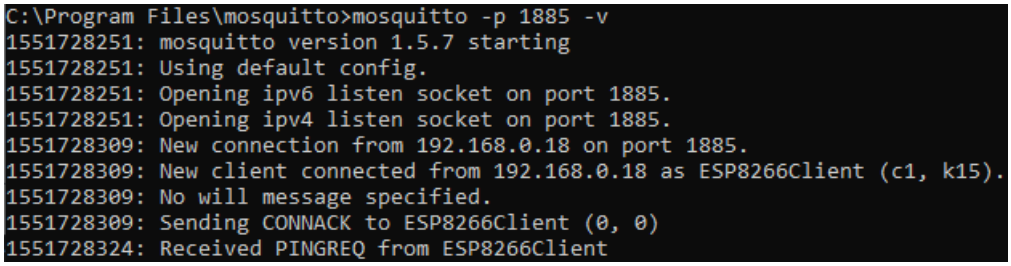
O *broker* utilizado neste projeto foi o mosquitto, um *broker* MQTT que é de fácil instalação tanto em sistemas operacionais Windows, como Linux e até mesmo em dispositivos microcomputadores, como a placa *Raspberry Pi*. O mosquitto foi utilizado neste projeto como *broker*, instalado em um computador com Windows.

O mosquitto permite com que o servidor receba mensagens de um determinado tópico, e as envie para quem precisa saber dessas informações. Todo esse controle de recebimento e envio de dados pode ser visto através da variável de sistema "Sys", que mostra o *log* de envio e recebimento em tempo real, de todos os pacotes. Pode-se filtrar por tópico, mostrando apenas o que for necessário caso a caso.

Após a instalação dos arquivos do MQTT, deve-se fazer a inicialização do servidor *broker* no computador com Windows. Existem diversas opções que podem ser alteradas na inicialização do servidor, podendo escolher até mesmo a porta a qual o servidor irá ser criado.

Neste caso, deixou-se a configuração padrão do MQTT. O comando que deve ser utilizado para a inicialização, através do *prompt* de comando é o "mosquitto". Com o comando -v (verbose), mostram-se informações como versão do *MQTT*, qual configuração está sendo utilizada e, em caso de erro, o tipo do erro. O comando -p indica qual porta quer que o servidor utilize. A Figura 11 demonstra o que deve ser exibido ao inicializar.

Figura 11 - Inicialização do Broker MQTT no Windows



Fonte: Autoria Própria

Com isso, tem-se o *broker* inicializado e pronto para enviar ou receber informações. Essas informações serão enviadas através de uma função chamada *callback,* que será vista na próxima sessão.

3.2 CONFIGURAÇÃO DO NODEMCU

Inicialmente deve-se instanciar as variáveis que serão utilizadas no projeto. Neste caso, foram instanciadas as variáveis de rede, como a rede local utilizada, a senha dessa rede, o *IP* do servidor *MQTT* e a porta deste servidor. Também foram instanciadas variáveis para os dois tópicos de mensagens *MQTT*. A Figura 12 mostra essas definições na IDE Arduino, que são enviadas para a memória da placa NodeMCU.

Figura 12 - Definições de Rede e Tópicos

```
// -- Definições de Rede e conexão MQTT / Tópicos --

const char *ssid = "Valk WIFI";
const char *password = "Valk300899";
const char *MQTTServer = "192.168.0.7";
const int   MQTTPort = 1883;
const char* ledTopic = "led";
const char* sensorTopic = "movimento";
```

Fonte: Autoria Própria

A placa NodeMCU deve atuar como um *publisher* e um *subscriber*, enviando a informação do sensor de movimento para o *broker*, que posteriormente fará a publicação na rede social a respeito do movimento suspeito, e também esperando o envio de uma mensagem MQTT enviada pelo *broker*, no tópico "LED".

O objetivo principal dessa placa é enviar e receber esses dados, portanto precisa-se de uma função de *callback*. A função de *callback* tem o propósito de enviar as mensagens para o *broker*, filtrando por tópico. O *broker* fará a divisão das mensagens por esses tópicos, enviando ou recebendo mensagens para os clientes por tópicos citados. Por exemplo, se houvesse um sensor de temperatura, ele enviaria mensagens com o tópico "temperatura" para o *broker*, atuando como um *publisher*, e um outro cliente que precisasse saber das informações de temperatura, atuaria como um *subscriber*, recebendo as informações do tópico. A Figura 13 mostra o código dessa função.

Figura 13 - Função de Callback

```
void callback(String topic, byte* message, unsigned int length) {
  Serial.print("Mensagem recebida no tópico: ");
  Serial.print(topic);
  Serial.print(". Mensagem: ");
  String messageTopic;

  for (uint8_t i = 0; i <  length; i++) {
    Serial.print((char)message[i]);
    messageTopic += (char)message[i];
  }
  Serial.println();

  if (topic == ledTopic) {
    if (messageTopic == "ON") {
      digitalWrite(pinLED, HIGH);
      Serial.print("LED - LIGADO");
    }
    else if (messageTopic == "OFF") {
      digitalWrite(pinLED, LOW);
      Serial.print("LED - DESLIGADO");
    }
  }
  Serial.println();
}
```

Fonte: Autoria Própria

No decorrer deste trabalho, essa função de *callback* foi muito utilizada para a funcionalidade deste projeto. A função de *callback* é apenas uma das funções utilizadas neste projeto. As outras que serão citadas, são:

- *connectToWifi*: Função para que seja feita a conexão na rede *wifi* local, inserindo o SSID e a senha como parâmetros;
- *connectToBrokerMQTT*: Função para que seja feita a conexão no broker MQTT;
- *reconnectToBrokerMQTT*: Função caso a conexão seja perdida;
- *sendSensorData*: Função para enviar os dados ao broker MQTT;

A Figura 14 mostra o escopo dessas funções:

Figura 14 -Funções

```
void connectToWiFi();
void connectToBrokerMQTT();
void reconnectToBrokerMQTT();
void callback(String topic, byte* message, unsigned int length);
void sendSensorData();
```

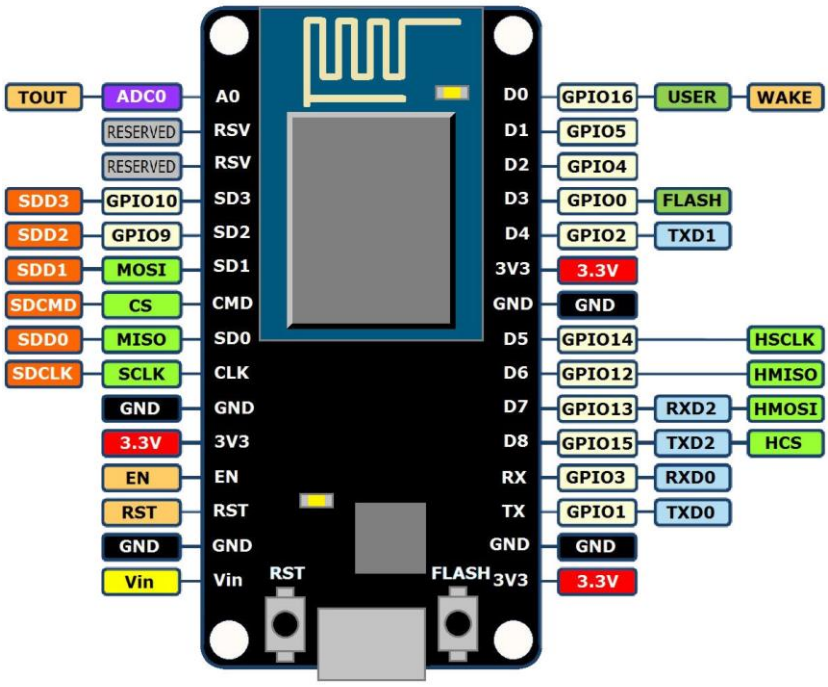
Fonte: Autoria Própria

Cada função é apresentada na sessão a seguir.

Outro ponto que é essencial para o funcionamento deste projeto, é a definição dos pinos que devem ser utilizados na placa NodeMCU. Para o sensor de movimento PIR, foram utilizados os pinos D2, e para o LED, o pino D3.

O sensor de movimento PIR precisa ser energizado para funcionar corretamente, sendo necessária uma fonte de 5V, menos que isso o sensor não funciona, e mais que isso ele pode não funcionar corretamente ou danificar o módulo. A Figura 15 mostra a pinagem da placa utilizada.

Figura 15 - Funções



Fonte: (Adafruit, 2017)

Como pode-se ver, essa placa não possui pinos com alimentação de voltagem 5V, apenas de 3.3, que é insuficiente para o sensor funcionar. A solução encontrada foi a utilização do pino Vin. Esse pino tem como função principal ser utilizado para energizar outros dispositivos com fonte de alimentação externa, como motores, porém quando ele é alimentado via USB, que foi o caso em todos os

momentos deste projeto, ele tem a voltagem de 5V. Desta forma podemos utilizar um sensor PIR neste projeto.

Finalizando este capítulo, é necessário comentar duas bibliotecas que são primordiais para o entendimento do próximo capítulo, na qual são explicadas cada função deste projeto. As duas bibliotecas importadas são a ESP8266WiFi.h e a *PubSubClient.h*. A primeira biblioteca é a que permite que a conexão a uma rede *wifi* seja feita através da placa NodeMCU, através das variáveis de definição de rede que foram explicadas neste capítulo. A segunda biblioteca é a que permite utilizarmos função como a de *callback*, permitindo a comunicação com o nosso *broker*.

3.2.1 Funções NodeMCU

As duas funções principais de qualquer projeto utilizando microcontroladores são a função de "*setup"* e "*loop"*. A função "*setup"* é a primeira que é executada, e sempre é executada apenas uma vez. Nela define-se o pino do LED como *output*, a taxa de *baud* que será utilizada, e é feita a primeira conexão na rede *wifi* e no *broker*. A taxa de *baud* é o que define a comunicação entre microcontrolador e o computador.

A função de *loop* por sua vez deve rodar de forma constante, infinita. Essa função é responsável por enviar os dados ao *broker MQTT* e permitir a reconexão ao *broker*, caso o mesmo perca a conexão. A Figura 16 mostra o fragmento de código referente ao *setup* e *loop*.

Figura 16 - Funções Setup e Loop

```
void setup() {
  Serial.begin(115200);
  pinMode(pinLED, OUTPUT);
  connectToWiFi();
  connectToBrokerMQTT();
}

void loop() {
  if (!client.connected()) reconnectToBrokerMQTT();
  sendSensorData();
  client.loop();
}
```

Fonte: Autoria Própria

Inicia-se a explicação da função *connectToWifi* primeiro, demonstrada na Figura 17, seguindo a ordem a qual as funções são executadas.

Figura 17 - Função connectToWifi

```
void connectToWiFi() {
  Serial.println("Conectando a rede: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Conectado!");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
  delay(1000);
}
```

Fonte: Autoria Própria

Essa função começa pelo processo de indentação, mostrando ao usuário qual rede ele está tentando conectar através da variável global *"ssid"*, definida previamente. Desta forma se tem um controle do que está sendo feito no projeto, se tornando mais fácil o encontro de possíveis erros. A função *Wifi.begin*, da biblioteca ESP8266WiFI.h, faz a conexão através do *"ssid"* da rede e a senha. Enquanto essa rede não for conectada, a conexão é repetida depois de 500ms. Após a conexão, é mostrado também ao usuário o IP local que foi designado a placa. Todas essas informações são demonstradas através do monitor serial.

As próximas duas funções que serão explicadas são a *connectToBrokerMQTT* e a *reconnectToBrokerMQTT*, demonstradas na Figura 18. A primeira função, importada da biblioteca *PubSubClient.h*, conecta a nossa placa como cliente em um *broker* especificado. A função de *reconnect*, reconecta o cliente caso a conexão tenha sido perdida, e atua como um *subscriber* no tópico, neste caso, do LED.

Figura 18 - Funções connect e reconnectToBrokerMQTT

```
void connectToBrokerMQTT() {
  client.setServer(MQTTServer, MQTTPort);
  client.setCallback(callback);
  Serial.println("Conectado ao broker");
  delay(500);
}


void reconnectToBrokerMQTT() {
  while (!client.connected()) {
    Serial.println("Conectando ao broker MQTT");
    if (client.connect("NodeMCU Client")) {
      client.subscribe(ledTopic);
      Serial.println("Conectado com sucesso!");
    }
    else {
      Serial.print("Falha estado: ");
      Serial.println(client.state());
      Serial.println("Tentar novamente em 5s");
      delay(5000);
    }
  }
}
```

Fonte: Autoria Própria

Por fim, a função s*endSensorData*, que define o cliente como um *publisher*, é a responsável também por enviar os dados do sensor de movimento PIR. Quando o estado do sensor for baixo (LOW), ele deve enviar um 0 ao *broker*, indicando que não tem presença alguma na região. Quando o estado do sensor for alto (HIGH), ele envia 1. A Figura 19 mostra o código da função *sendSensorData*.

A interação entre placa NodeMCU, *broker MQTT* e ferramenta Node-RED, será demonstrada na próxima sessão.

3.3 FLOW NODE-RED

O *flow* utilizado no Node-RED, demonstrado na Figura 20, mostra o funcionamento do projeto. Temos um *node MQTT* de *input*, enviando os dados do sensor, esse *node* é o primeiro, chamado de movimento. Esse *node* funciona da seguinte maneira: todos os dados que o sensor PIR recebe, ele envia ao *broker MQTT* em um servidor *windows*. No node-RED, esse *node* atua como um *subscriber* desse *broker*, e recebe todos os pacotes com o tópico de movimento, direcionando-os como *input* para o *node* de função que segue ele.

Figura 19 - Função sendSensorData

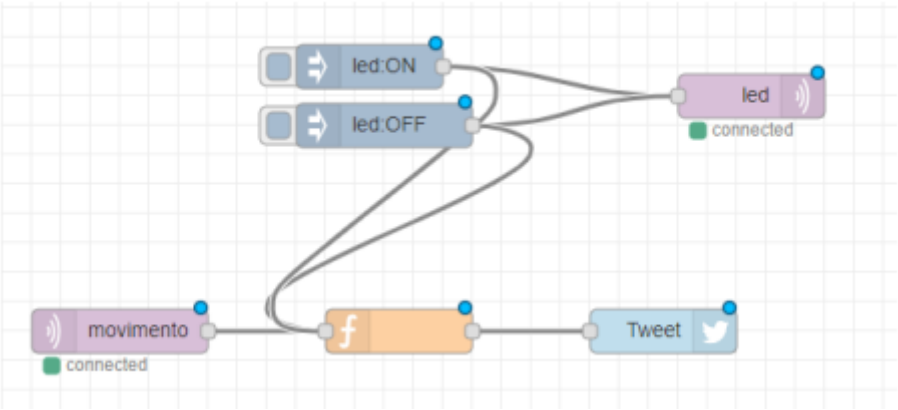
```
void sendSensorData() {

int pirVal = digitalRead(pirPin);

if(pirVal == LOW){
client.publish(sensorTopic, "0");
}
else
{
client.publish(sensorTopic, "1");
delay(5000);
}
```

Fonte: Autoria Própria

O *node* de função então irá decidir através do sinal 1 ou 0 de movimento. Se o sinal de movimento for 1, dizendo que alguém passou pelo sensor, a função enviará um sinal de ON para o LED, acionando-o, e ao mesmo tempo, enviará um *tweet* para a rede social Twitter.

Nas sessões seguintes cada *flow* será explicado um a um para melhor entendimento.

Figura 20 - Flow Node-RED



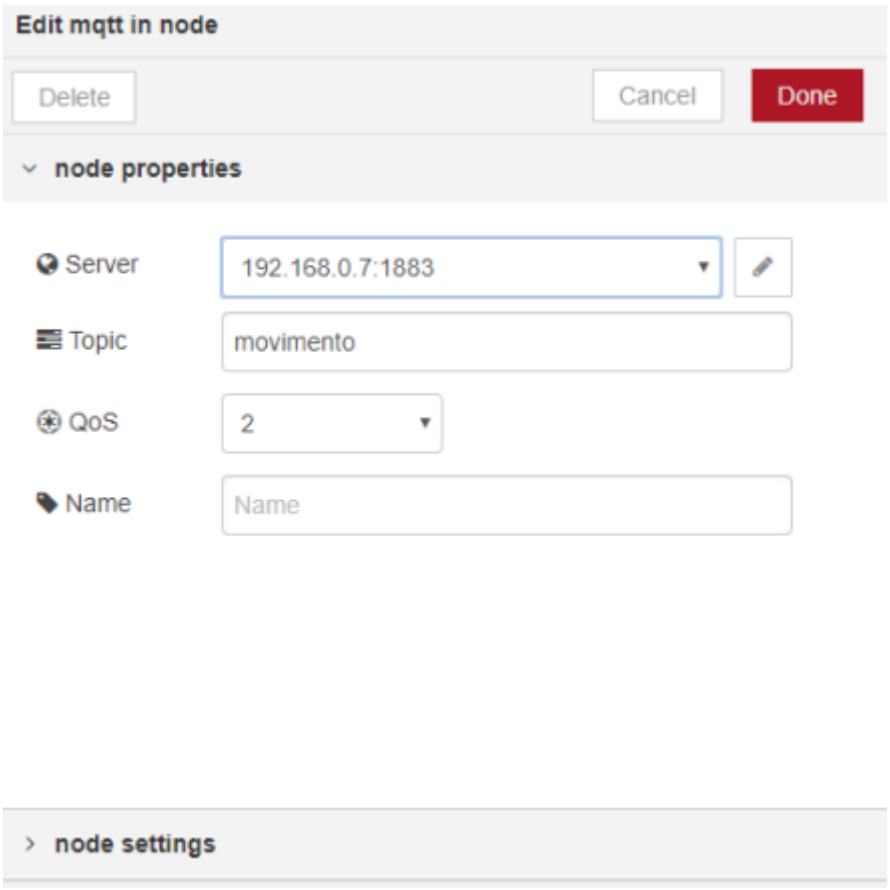
Fonte: Autoria Própria

3.3.1 Flow Node-RED: movimento input

O *flow* de movimento atua como um *node* de *input*, ou seja, tem como função enviar algum tipo de dado. Os *nodes* de *input* dependem do tipo de canal e mensagem a qual será enviada. No caso deste projeto, o node de *input* é do tipo

MQTT, e enviará a mensagem de um determinado tópico, referente ao movimento. A Figura 21 mostra as propriedades deste *node* em questão:

Figura 21- Flow Node-RED: movimento input



Fonte: Autoria Própria

Existem quatro propriedades que podem ser editadas neste tipo de *node*:

- *Server*: Qual é o endereço de IP e a porta a qual o node deve se conectar para pegar as informações do *broker*;
- *Topic*: Após conectado no *broker*, qual tópico ele deve estar inscrito para pegar as informações;
- *QoS*: Qual deve ser o indicativo de *Quality of Service*, podendo escolher entre 0, 1 ou 2.;
- *Name*: Nome que será utilizado para identificar o *node*, em caso de nulo ele escolhe o nome do tópico.

Pela lógica desenvolvida na placa *Node-MCU*, o sensor enviará sinais 0 e 1, então são esses dois valores que serão utilizados pelo node de função para que os dados possam ser utilizados. O *node* função será explicado na sessão seguinte.

3.3.2 Flow Node-RED: função

O *node* função, do tipo função, tem como objetivo utilizar o valor enviado pelo *node* anterior de movimento, podendo ser feita uma lógica própria. A Figura 22 mostra o código deste *node*.

Figura 22 - Flow Node-RED: função

```
Function
1  if (msg.payload == 1){
2      msg = today + "Sensor Ativado";
3      led:ON;
4      delay(500);
5      led:OFF;
6  }
7  return msg;
8
```

Fonte: Autoria Própria

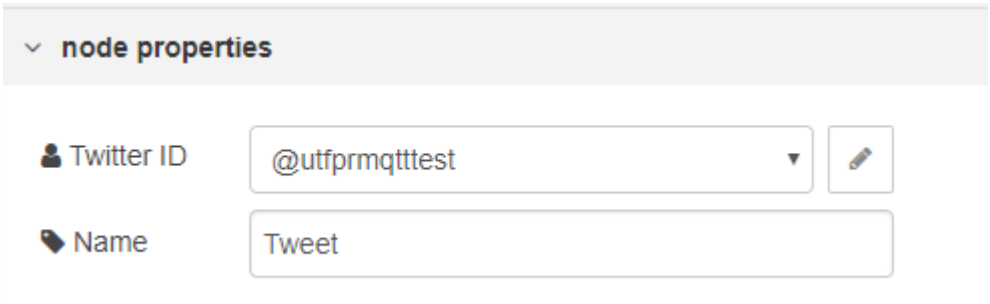
Todo valor enviado pelo *node* movimento, é passado através da variável de projeto *"msg.payload".* Neste caso, ela pode ser apenas 0 (quando não tem movimento) e 1 (quando tem movimento). Pela lógica do projeto, a função só deve funcionar quando o sensor emitir o sinal de 1.

Quando isso acontecer, deve-se pegar o momento atual desta ocorrência e salvar em uma variável "msg", para que se possa enviar para a rede social no *node* Twitter. Após esses dados serem salvos, ativa-se o *node* led:ON, que tem a função de enviar um sinal para a placa, através de outro *node*, para ligar o led. Tem-se um *delay* de 5s, e após isso chama-se o *node* led:OFF, que desliga o led. A função é finalizada retornando a variável "msg".

3.3.3 Flow Node-RED: node Twitter

O *node* de *output* Twitter, tem como função fazer uma postagem na rede social, sendo a mensagem desta postagem o *output* do *node* anterior. A Figura 23 mostra as propriedades deste *node*. Utilizou-se para este projeto uma conta especial criada apenas para este tipo de projeto.

Figura 23 - Flow Node-RED: node Twitter



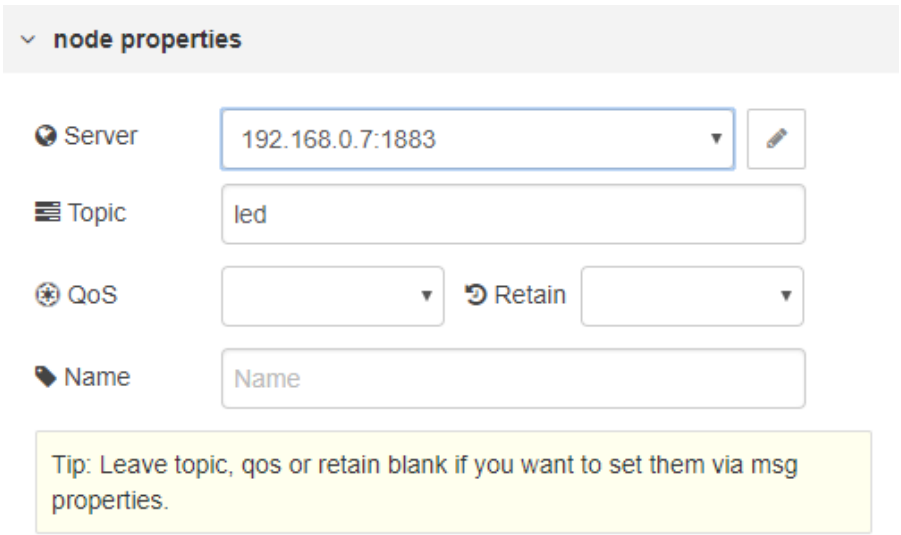
Fonte: Autoria Própria

Por fim, os *nodes* de led serão explicados na próxima sessão.

3.3.4 Flow Node-RED: *nodes* led:ON, led:OFF e led

Os *nodes* de *input*, led:ON e led:OFF tem como função enviar uma mensagem para o próximo *node* com uma mensagem, ON ou OFF. O próximo *node* é o *node* de *output MQTT* "led", esse *node* se parece com o *node* de movimento em suas propriedades, a única diferença é que em vez de receber as informações, ele envia. As configurações deste *node*, vistas na Figura 24, são as mesmas do *node* de movimento, só se altera o tópico, de movimento para led.

Figura 24 - Flow Node-RED: Node Led



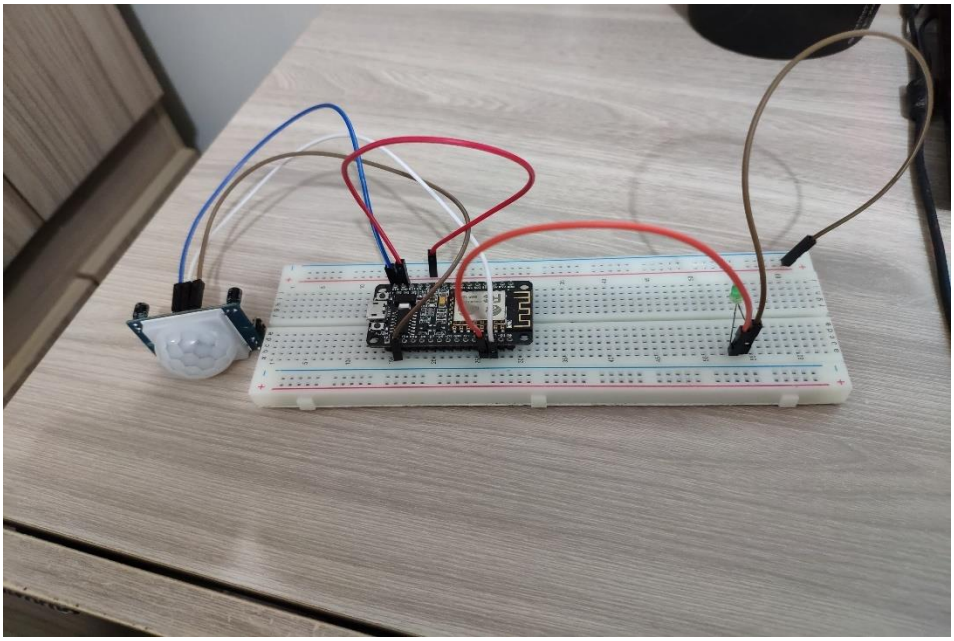
Fonte: Autoria Própria

Quando algum dos *nodes* led:ON ou led:OFF for ativado, a mensagem é passada para o *node* led, que envia a mensagem para qualquer dispositivo inteligente que esteja conectado no *broker*, e inscrito no tópico led. No caso deste projeto, o Node-MCU.

3.4 FLOW NODE-RED: FUNCIONAMENTO

Nesta sessão, será demonstrado o funcionamento do projeto. Com toda a configuração feita previamente na placa Node-MCU, montagem dos sensores na protoboard, configuração do servidor no ambiente *Windows*, e configuração do ambiente Node-RED, o projeto pode ser demonstrado. A Figura 25 mostra como foi disposto os componentes.

Figura 25 - Prototipação do projeto



Fonte: Autoria Própria

Para simular o projeto, como se fosse em um ambiente real de segurança domiciliar, o sensor precisaria captar um movimento para ativação. Neste caso, com o movimento da mão em frente ao sensor, o LED acendeu, indicando a presença, e logo a postagem na rede social foi feita, como demonstra as Figuras 26 e 27.

Figura 26 - Simulação do projeto

Fonte: Autoria Própria

Figura 27- Postagem na rede social Twitter

Fonte: Autoria Própria

Como pode-se ver, o projeto concluiu o objetivo de um simples sistema de segurança domiciliar, integrando diferentes componentes e tecnologias.

A utilização do Node-RED para esse projeto facilitou muito a identificação de falhas de erro humano, por conta da sua interface. Em um projeto de grande escala, é muito fácil se perder com o nome dado a cada placa, qual *broker* que cada placa está conectada, qual tópico ele está inscrito ou deve publicar informações. O Node-RED possibilita uma limpeza dessas informações, através dos *nodes*, fazendo com que o projeto fique mais limpo e visual para se trabalhar.

Se fosse optado por não utilizar a ferramenta, o projeto seria possível de ser desenvolvido da mesma forma, todavia, levaria muito mais tempo pelo fato de se verificar as informações das placas de uma forma mais complexa, entrando nos *logs* da própria placa e vendo o que é enviado e o que não é, o que para um projeto de digamos 100 dispositivos, é inviável.

Outra vantagem de se utilizar o Node-RED, é que se tem diversas funções que podem ser utilizadas para simular o envio de dados, o que é uma ótima ferramenta para se realizar testes, verificar se a conexão com a placa está funcionando. No decorrer dos testes, essa função foi muito utilizada para verificar a distância que a placa pode captar o sinal emitido.

Por ser de simples e rápida utilização, o Node-RED é recomendado para projetos que tem justamente essa finalidade: serem desenvolvidos de forma ágil. Ótimo para desenvolvimento de protótipos para verificar se o projeto é conciso para depois se ter um desenvolvimento mais profundo.

Uma desvantagem de se utilizar o Node-RED é de que para uma entrada de dados muito grande, com valores acima de 10.000 por segundo por exemplo, um servidor apenas não se sustenta. O que é algo relativamente compreensível pelo fato de serem dispositivos com um hardware limitado que estarão atuando como o servidor. Neste caso, como o Node-RED estava no desktop, esse não seria um problema. Mas para placas como *Raspberry Pi3*, que é muito utilizada com essa ferramenta, este pode ser um problema dependendo do projeto.

O *MQTT* teve uma taxa alta de sucesso na transmissão de pacotes para esse projeto. Sempre que a placa com o sensor estava em um local que tinha o mínimo possível de sinal *WIFI*, a placa conseguia emitir o sinal do sensor, e receber o sinal para ligar o LED. Caso estivesse fora da cobertura do roteador, não recebia, o que já é esperado.

4 CONCLUSÃO

O trabalho proporcionou a utilização de diversas áreas da computação: programação, redes de computadores e microcontroladores. Até mesmo eletrônica pôde ser utilizada. Isso se deve ao fato de que a Internet das Coisas permitir uma abrangência de áreas, o que explica claramente o crescimento que a área tem tido nos últimos anos.

Com esse projeto conseguiu-se agregar diferentes dispositivos em uma aplicação só através da utilização do protocolo *MQTT*, que permitiu essa comunicação. Como esse projeto foi um projeto considerado leve, com poucos sensores e ações, o protocolo conseguiu servir o propósito de forma eficaz, conseguindo se comunicar com o servidor sem problemas de comunicação.

Da mesma forma, a configuração da placa Node-MCU, por serem apenas um LED e um sensor de movimento, foi de fácil configuração. O código em si, onde a placa deve estar preparada para receber os pacotes enviados pelo servidor, e enviar pacotes para o mesmo, foi a parte mais trabalhosa. Percebeu-se que os pacotes devem ser enviados para o endereço e a porta corretamente especificados, caso contrário esses pacotes não são enviados de forma alguma, mesmo estando na mesma rede. Tendo a placa configurada para enviar ao endereço correto, os pacotes chegavam ao servidor sem problema algum.

O ambiente Node-RED, com sua IDE baseada em *flows* e *nodes*, permite com que o projeto se desenvolva da maneira que o desenvolvedor queira, podendo integrar *hardware*, *software, websites* e API's em um só projeto. A grande vantagem de sua utilização foi de se ter uma visualização melhor do projeto como um todo, das informações enviadas e recebidas pelas placas, o que permitiu uma implementação mais ágil e concisa. Dessa forma, o projeto conseguiu fluir de uma maneira sólida, integrando 3 diferentes objetos: placa microcontrolada, servidor em um *desktop windows* e rede social (*website*). A ferramenta concluiu a ligação necessária para que o projeto pudesse ter uma visualização limpa do que estava acontecendo, para que a programação e integração junto do protocolo *MQTT* ficasse mais fácil.

O protocolo *MQTT* foi o responsável pela comunicação entre placa, servidor e ferramenta Node-RED. A configuração feita, entre placa e Node-RED foi basicamente quando enviar o pacote, para quem esse pacote é enviado, e o sentido

inverso desse fluxo, ou seja, quem deve receber esse pacote, e o que deve ser feito quando ele for recebido (baseado na mensagem).O MQTT se trata de um protocolo utilizado para Internet das Coisas, sem a utilização dele, nenhuma comunicação poderia ter sido feita. Foi a parte mais trabalhosa do projeto, pois nesse caso, como demonstra a Figura 10, os pacotes eram enviados e recebidos em 3 sentidos diferentes:

- Placa Node-MCU: *Publish* no tópico movimento, *Subscribe* em LED, enviados para o *Broker* no *windows*;
- Servidor Windows: *Publish* no tópico movimento, *Subscribe* em LED, ambos para o Node-RED e para a placa Node-MCU;
- Node-RED: Publish no tópico movimento, *Subscribe* em LED, enviados para o Broker no *windows*;

Pode parecer que seja a mesma tarefa para os três objetos, mas por se tratar de três tecnologias diferentes, e quatro sentidos diferentes de envio e recebimento de dados, a complexidade do projeto aumentou e tornou-se um desafio.

5 TRABALHOS FUTUROS

Este projeto com toda certeza abriu portas para um pensamento diferente em como atuar utilizando a Internet das Coisas. Acredita-se que com a utilização de diferentes módulos e sensores, pode-se atingir diferentes resultados. Um possível projeto que pode utilizar essas mesmas tecnologias, com o acréscimo de diferentes conhecimentos de elétrica, seria o de uma automação de um *datacenter*. Sabe-se que existem serviços e produtos que controlam um ambiente de um datacenter, como por exemplo entrada e saída, controle de temperatura, tensão. Porém, esses serviços tem um custo alto, as vezes até uma mensalidade.

Com um projeto de baixo custo como esse que foi apresentado, podemos fazer o controle de uma sala de uma maneira simples, com a representação dos dados do sensor em uma aplicação *web*, e até mesmo a automação de ventilação e entrada e saída. Tudo isso com a adição de um módulo relé, controlando quando deve ser ativada a energia de um ar-condicionado, ou da tranca da porta de entrada e saída por exemplo.

A continuidade deste projeto pode ser tratada em diferentes aspectos:

- Porcentagem de pacotes: realizar medições através de testes referentes a perda de pacotes, utilizando o protocolo *MQTT* para analisar a distância máxima de sensores;
- *QoS*: utilizar o *QoS* do *MQTT* para analisar o meio de transmissão. Verificar resultados diferentes para cada tipo de *QoS*;
- Número de sensores: aumentar o projeto em escala de sensores, placas e atuadores, utilizando a ferramenta Node-RED para controlar o projeto e verificar desempenho;
- Armazenamento em nuvem: este projeto não possuiu qualquer armazenamento das informações dos sensores e placas. Existe a possibilidade de armazenarmos estes dados para análise futura;
- Protocolos de comunicação: utilizar protocolos diferentes do MQTT como por exemplo o http e comparar funcionalidade e desempenho.

REFERENCIAS

D. THANGAVEL, X. MA, A. VALERA, H. TAN AND C. K. TAN. "Performance evaluation of MQTT and CoAP via a common middleware," *2014* ***IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*****,** Singapore, 2014, pp. 1-6.

GROWTHENABLER. 2017 10 dez. 2017. **Roundup Of Internet Of Things Forecasts.** Disponível em: <https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/#587ff1a71480>. Acesso em: 20 jan. 2019.

HUSEYIN, Y.; ALI-ELDIN, A. M.T. A model for predicting user intention to use wearable IoT devices at the workplace, **Journal of King Saud University - Computer and Information Sciences**, 2018.

JAUOD, J. N. G. DESIGNING & DEPLOYING CONNECTED DEVICES

MIDTERM ACTIVITIES. Disponível em: <https://www.academia.edu/38851720/DESIGNING_and_DEPLOYING_CONNECTED_DEVICES>. Acesso em: 23 jun. 2018.

MOHAMED ,S.; KHALID, E.; KASEM, K. Towards Privacy Preserving IoT Environments: A Survey. **Wireless Communications and Mobile Computing.** vol. 2 2018.

MONK, S. **Programação com Arduino: Começando com Sketches**. 1. ed. Bookman, 2013.

PFLANZNER, T.; KERTÉSZ, A. A survey of IoT cloud providers. 39th International Convention on Information and Communication Technology, **Electronics and Microelectronics (MIPRO).** p. 730-735. 2016.

SANTAELLA, L. Mídias locativas: A internet móvel de lugares e coisas. **Revista Famecos.** Porto Alegre (RS), nº.35, p. 95-101, fev. 2015.

SANTOS, B. P; et al. **Livro texto minicursos.** 1. ed. Porto Alegre: SBC, 2016.

SINGH, G. **IoT - Internet of things.** Disponível em: <https://www.academia.edu/35419052/IoT_-_Internet_of_things> Acesso em: 10 fevn. 2019.

WENDLING, M. **Sensores**. 8 jan. 2016. Disponível em: <http://www2.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/4---sensores-v2.0.pdf>. Acesso em: 23 jun. 2018.