USO DO MICROCONTROLADOR ESP8266 PARA ACIONAMENTO DE DIODO EMISSOR DE LUZ (LED), UTILIZANDO BANCO DE DADOS DO GOOGLE (REALTIMEBASE DATABASE).

José Cícero da Silva Filho[1]
Marcos Vinicius Silva Bento[2]
Dheiver Francisco dos Santos[3]

Engenharia Mecatrônica

cadernos de graduação
ciências exatas e tecnológicas

ISSN IMPRESSO 1980-1777
ISSN ELETRÔNICO 2316-3135

RESUMO

O conceito de mobilidade da informação surgiu no início da década de 90, onde o cientista de computadores Mark Weiser demonstrou pela primeira vez as características da computação ubíqua, em sua publicação à revista *Scientic American*. Mas foi no final dos anos 90 que foram generalizados o conceito de Internet das Coisas (em inglês: *Internet of Things*, abreviadamente, IoT) com a consequente ampliação do desenvolvimento dos dispositivos, permitindo a conectividade entre dispositivos em escala. Este trabalho propõe uma arquitetura simplificada utilizando de dispositivos físicos de baixo custo e menor processamento que computadores comuns. Foi feito o uso de requisição de protocolos de dados HTTP (*Hypertext Transfer Protocol*) para comunicação e transferência de texto, o que permite estabelecer uma conexão via web ao banco de dados do google (*Firebase Realtime Database*), a transferência dados entre dispositivos eletrônicos microcontrolados, e o envio de comandos de controle. Foram apresentados também a configuração e o uso do microcontrolador NodeMCU ESP8266 como dispositivo principal para comunicação via web e a elaboração de aplicativo mobile *Android* para interface de comunicação com banco de dados em nuvem.

PALAVRAS-CHAVE:

Computação ubíqua. Internet das coisas. Protocolos. Microcontrolador. Banco de dados.

ABSTRACT

The concept of information mobility emerged in the early 1990s, where computer scientist Mark Weiser first demonstrated the characteristics of ubiquitous computing in his publication to Scientic American. But it was in the late 1990s that the Internet of Things (IoT) concept was broadened with the consequent expansion of device development, enabling connectivity between devices at scale. This work proposes a simplified architecture using low cost physical devices and less processing than ordinary computers. The use of Hypertext Transfer Protocol (HTTP) data protocols was used for communication and text transfer, which allows the establishment of a web connection to the google database (Firebase Realtime Database), the transfer of data between microcontrollable electronic devices, and sending control commands. Also presented were the configuration and use of the NodeMCU ESP8266 microcontroller as the main device for web communication and the development of mobile Android application for communication interface with cloud database.

KEYWORDS

Ubiquitous computing. Internet of things. Protocols. Microcontroller. Database.

1 INTRODUÇÃO

O termo conectividade de dispositivos e mobilidade surgiu por Weiser M. (1991), onde propôs o conceito de computação ubíqua, que representa a interseção da computação pervasiva e móvel, significando de forma simplificada, a onipresença da informação, mobilidade e portabilidade computacionais (Ferreira, 2014).

Mais tarde, o pesquisador do *Massachusetts Institute of Technology* (MIT), Kevin Ashton, usou a expressão "*Internet of Things*" (IoT) pela primeira vez em uma apresentação feita à *Procter & Gamble* (P&G) em 1999. Ashton tinha como objetivo conectar os produtos da companhia por meio de transceptores de RFID (*Radio-Frequency Identification*), (Serafim, 2014). Segundo Pereira (2017), a IoT permite a conectividade de objetos a qualquer hora, em qualquer lugar, com qualquer coisa, utilizando qualquer caminho, rede ou serviço.

O modelo de IoT proposto por Souza (2015), é constituída por camadas, onde são compostas por um conjunto de tecnologias habilitadoras, são elas: camada de comunicação, camada de *middleware*, camada de serviços e camada de aplicação. O processo de conexão das coisas por sensoriamento e redes sem fio se resume a uti-

lização dos seguintes processos: Sensoriamento; Rede de comunicação; Plataforma de comunicação (Software); Interfaceamento; Segurança dos dados.

Camada de Dispositivos: que implementa a coleta de dados, atuação no ambiente, comunicação e identificação das entidades físicas. Camada de Comunicação: que tem por função habilitar a comunicação entre as entidades físicas e camadas superiores requer protocolos de comunicação que devem acompanhar a evolução das redes de sensores e redes de sensores sem fio. Camada de Middleware: que desempenha funções como gerenciamento, segurança e contextualização de dados, entidades ou serviços, escalabilidade e facilidade de integração entre dispositivos. Camada de Serviços: desempenha funções de gerenciamento de serviços, segurança e contextualização dos serviços, e podem ser implementadas sob um middleware. Camada de Aplicação: é representada por aplicações pervasivas como, por exemplo, aplicações inteligentes em casas, automóveis, cidades, transportes e objetos. Nessa camada reside a inteligência e tomada de decisão no contexto de IoT no modelo atual. (SOUZA, 2015, p 20,21).

A camada de integração de dispositivo é formada por três componentes principais. O componente de nível mais baixo é o driver, que se comunica com os diferentes sensores, tags e atuadores de baixo nível, informações específicas do fornecedor e protocolos de comunicação. (KARZEL traduzido por LIMA, 2016).

A camada de comunicação é definida pela normativa IEEE 802.15, que específica uma variedade de WPANs (*Wireless Personal Area Network*), como Bluetooth. Segundo Momote (2016), os modelos mais utilizados de comunicação IoT são: *Bluetooth*, *Device-to-Cloud* (Dispositivo para nuvem), *Device-to-Gateway* (dispositivo para *gateway*), *back-end data sharing* (compartilhamento de dados de *back-end*).

Com a ascensão da IoT, houve a necessidade da interação de conectividade com banco de dados em nuvem para transferência de dados em tempo real. Os sistemas de banco de dados NoSQL (*Not Only Structured Query Language*) são mais utilizados por poder otimizar os aplicativos que exigem modelos de dados de grande volume de dados. Segundo Vilela (2016), o banco de dados NoSQL foi proposto com o objetivo de oferecer variedade de aplicações e de armazenamento de dados com ênfase no fator quantitativo.

Os bancos de dados NoSQL normalmente oferecem ferramentas de gerenciamento para aplicações e interface. O *firebase* da empresa Google é um exemplo de plataforma onde possui ferramenta para desenvolvimento de aplicativos e armazenamento de dados. Essa plataforma possibilita o armazenamento e sincronia dos dados com banco de dados na nuvem (*Realtime Database*) NoSQL, onde são sincronizados em tempo real e permanecem disponíveis mesmo quando o aplicativo está off-line.

Em nossa abordagem é apresentado ao leitor a elaboração e funcionamento prático via interface mobile de um sistema para monitoramento eletrônico e controle digital de um LED (*Light Emitting Diode*), utilizando como plataforma de armazenamento, o banco de dados da google (*firebase realtimebase database*). Foram utilizadas tecnologias existentes na literatura, SOUZA (2015), KARZEL (2016), MOMOTE

(2016), que são flexíveis tanto para dispositivos móveis quanto para aplicações. O processo de acionamento e segurança dos dados têm em comum a conectividade, que é realizada entre os dispositivos físicos em rede e conectividade via web e interface, fazendo uma ponte entre a interface e os dispositivos, podendo também, ser aplicado em amostragem de dados e manipulação, ou acionamento de atuadores, como por exemplo, um servomotor.

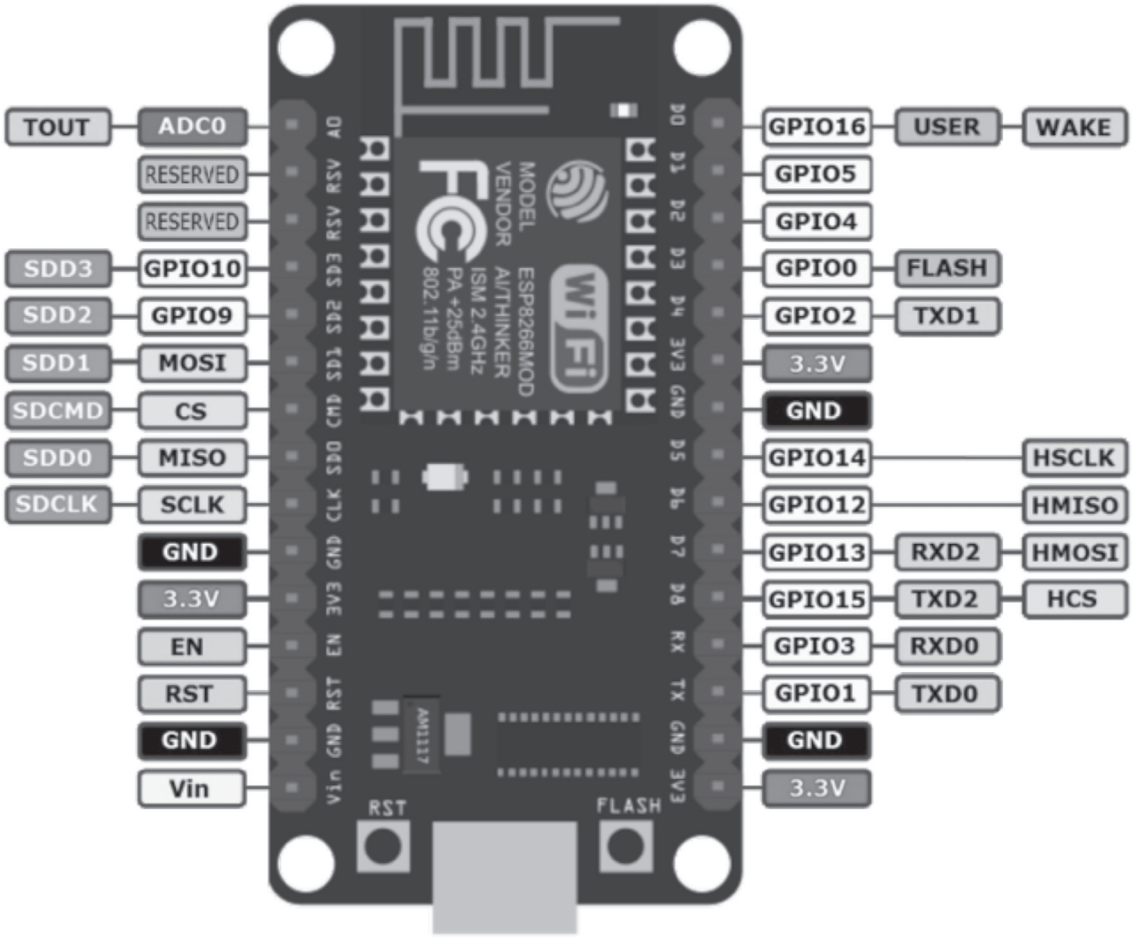
A seção 2 é apresentada a metodologia utilizada para experimentação estrutural da arquitetura IoT. A seção 3 mostra os resultados do desempenho da conectividade entre o microcontrolador com o banco de dados (*realtimebase database*) do google. A seção 4 descreve a conclusão do projeto, os problemas e as soluções que foram empregadas.

2. METODOLOGIA

O projeto proposto tem sua arquitetura voltada à orientação, à conexão e aos dispositivos, onde, além de permitir a comunicação de dispositivos com o mundo externo, possui como principal característica o uso de dispositivos mais baratos e menores, menos poderosos que computadores tradicionais e de baixo consumo de energia, que permitam utilizar a comunicação sem fio baseada em protocolos. Ao contrário de propostas como a de Ferreira (2014), Souza (2015), e Karzel (2016) que demonstram o funcionamento em camadas da IOT, este artigo demonstra uma metodologia simplificada da arquitetura, onde se divide em quatro partes principais, para maior e melhor compreensão da arquitetura e seu funcionamento. São elas: dispositivos físicos, comunicação, conectividade e interface.

O principal componente do dispositivo é o microcontrolador wifi ESP8266--NODEMCU, que foi usado pela capacidade de fazer a comunicação *wireless* proporcionada pelo chip de conexão ESP8266, que já faz parte do seu circuito. Também faz parte do circuito uma interface USB-serial e um regulador de tensão 3.3v, como mostra a Figura 1.

Figura 1: Pinagem do Microcontrolador wifi ESP8266-NODEMCU.



Curvello (2015) destaca uma breve explicação da legenda dos principais pinos do módulo, são eles:

- Vcc: Tensão de alimentação 3,3V. Módulo consome até 300 mA;
- GND: Sinal de Terra GND;
- Tx: Sinal de Tx do módulo, a ser conectado no Rx do microcontrolador (Sinal em 3,3V);
- Rx: Sinal de Rx do módulo, a ser conectado no Tx do microcontrolador (Cuidado! Sinal em 3,3V!);
- RST: Sinal de *Reset/Restart* acionado em nível baixo (GND);
- CH_PD: Sinal de habilitação do chip (*chip enable*), usado para gravação de firmware ou atualização. Deve ser mantido em nível ALTO para operação normal;
- GPIO: Pode ser controlado pelo *firmware*, e deve ser colocado em nível baixo (GND) para modo *update*, ou em nível alto para operação normal. I/O que pode ser controlada pelo firmware.

Tabela 1: Especificações do Módulo Wifi ESP8266

	ESP8266
Cores	1
Arquitetura	32 bits
Clock	80 MHz
Wifi	SIM
Bluethooth	NÂO
RAM	160KB
FLASH	16Mb
GPIO	17
Network Protocol	IPv4, TCP / UDP / FTP / HTTP

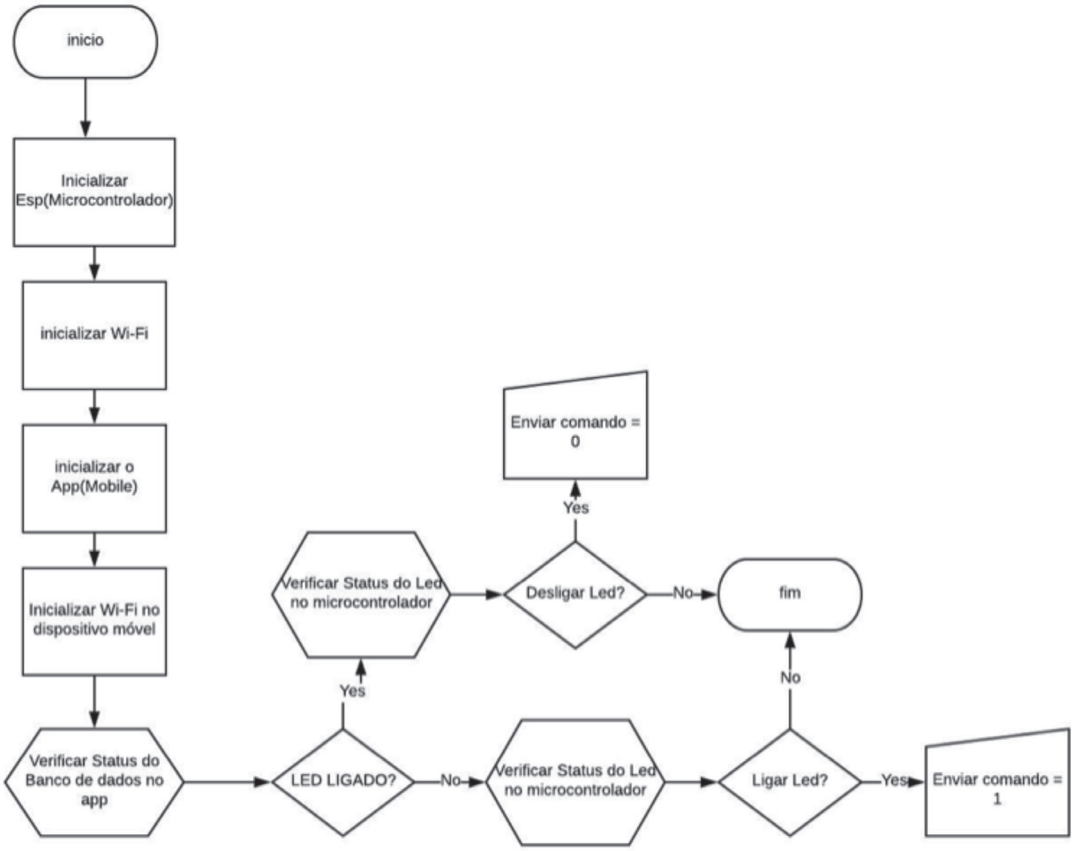
Fonte: Einstronic.com (adaptado)

Na programação do microcontrolador foi utilizada a IDE (*Integrated Development Environment),* ambiente de desenvolvimento integrado do arduino (arduino.org). A comunicação entre o microcontrolador e a API (*Application Programming Interface*), foi realizada através da biblioteca *firebase-arduino-master*. Com esta biblioteca configurada no IDE, é possível fazer uma requisição de transferência pelo protocolo HTTP (*HyperText Transfer Protocol*) com acesso ao banco de dados através

do <*secret-token*>, que é disponibilizado quando o banco de dados é criado no site do *Firebase*. Antes de realizar uma requisição HTTP. a biblioteca disponibiliza uma definição para que o microcontrolador se conecte à internet do ambiente, via rede *Wireless*, onde é definida a conexão tanto com a API do *firebase* quanto com o Wi-fi.

A conexão com o banco de dados foi configurada da maneira em que a plataforma *firebase* disponibiliza, para a gravação e leitura de dados. Através da alteração ao enviar dados para o banco de dados, o microcontrolador recebe as informações que está ocorrendo no banco e com essa requisição ele identifica se o sinal recebido é o que consta na lógica implementada no microcontrolador, executando, assim, a ação, que nesse exemplo é acender ou apagar um LED. Isso é feito através do envio de valores lógicos 1 (um) ou 0 (zero), como mostrado no fluxograma da Figura 2.

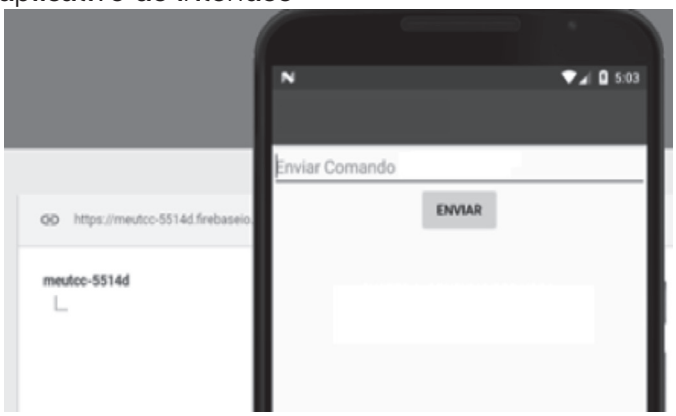
Figura 2: Fluxograma



Fonte: Autoria própria

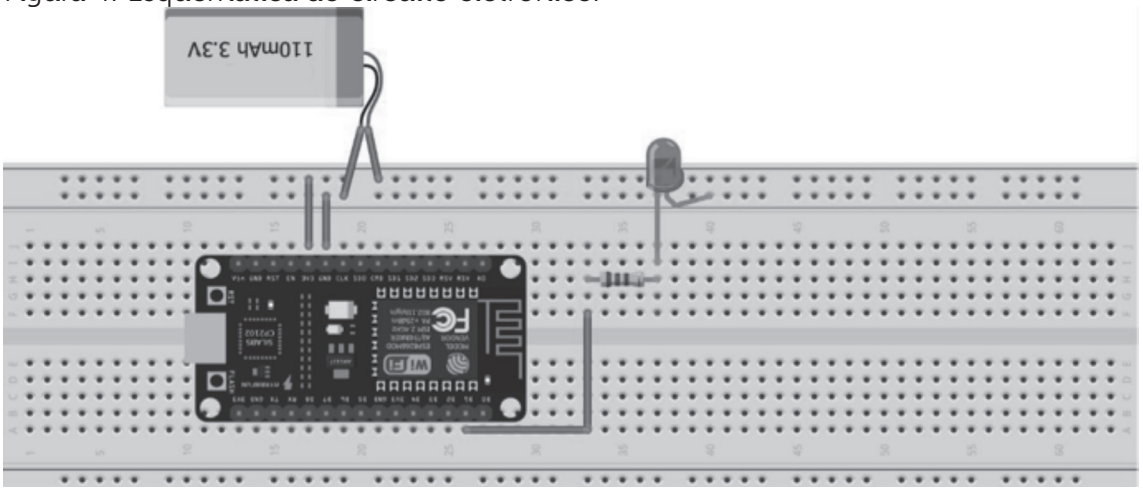
A interface de comunicação com banco de dados foi realizada através de aplicativo mobile *Android*, desenvolvido na plataforma *Android Studio*. O aplicativo tem um design simples e intuitivo, seu layout é composto por uma caixa para enviar um texto, um botão que envia o comando e um *textlabel-exibe texto*, onde exibe o status do banco de dados. Como mostra a na Figura 3.

Figura 3: Layout do aplicativo de interface



Com o projeto concluído e com acesso à internet, é possível alterar o estado do banco de dados, convertendo o valor *string* para *int*, pois o banco de dados trata de objetos do tipo string, porém o microcontrolador foi configurado para receber valores do tipo int.

Figura 4: Esquemática do circuito eletrônico.



Fonte: Autoria Própria

O cenário de teste proposto foi o acionamento de um diodo emissor de luz (LED), onde foi instalada uma infraestrutura de hardware para IoT, composta por um microcontrolador wifi ESP8266-NODEMCU, um resistor de 230Ohms, conectores do tipo jumpers e uma bateria de 3v. O esquemático do circuito eletrônico é apresentado na Figura 4.

O circuito do microcontrolador foi alimentado por uma tensão de 3 volts com origem na bateria, que por sua vez alimenta o LED pelo pino digital D0, configurada como saída. O LED está recebendo a proteção do resistor pelo catodo e o anodo está conectado ao GND (*graduated neutral density filter*), o "terra" do microcontrolador.

3. RESULTADOS

Nesse trabalho foi realizado o envio de diferentes comandos para um diodo emissor de luz (LED), utilizando para isso o microcontrolador NodeMCU ESP8266 12-E e banco de dados do google (*realtimebase database*). A abordagem apresentou a elaboração e funcionamento prático via interface mobile de um sistema para monitoramento eletrônico e controle digital (liga/desliga) do dispositivo. Foi utilizado um diodo emissor de luz (LED), um microcontrolador NODEMCU-ESP8266 12-E, um resistor de 220 Ohms, conectores do tipo jumpers e uma bateria de 3 volts.

Eletronicamente, foi possível acionar o LED diretamente pelo comando digital da porta D0 do microcontrolador. Através da definição de comando, <*pinMode(D0, OUTPUT)*>, definiu-se que a porta D0 é uma porta de saída de sinal digital, oferecendo ao dispositivo uma tensão de 3 volts. Logo, com a proteção do resistor, o LED recebeu uma corrente próxima a 0.013 Amperes. Porém, para configurar o controle para acionamento do LED foi preciso fazer uso dos comandos lógicos (liga/desliga), <*digitaWrite(D0, HIGH)*> para acender e <*digitaWrite(D0, LOW*)> para apagar o LED.

Ao conectar o microcontrolador com a web, foi possível alterar o estado do LED de acordo com o status do banco de dados, acessado pela página do Firebase. O comando foi enviado através de um smartphone com sistema operacional *Android*, e a mudança do estado do LED, de ligado para desligado, foi realizada em ms (milissegundos).

Ao começar o envio de comando via banco de dados para o microcontrolador, foi necessário fazer alguns testes no funcionamento do banco de dados, para saber se estava recebendo dados do aplicativo *Android*. Em seguida, fora realizada a análise de visibilidade de status do banco de dados e a verificação da conectividade do microcontrolador com a internet.

Tabela 2: Tempo de atraso da conexão do Módulo ESP8266 com o wi-fi

Endereço IP	Tipo	Latência
192.168.0.108	Wi-fi	184 ms

Fonte: Autoria própria

A conectividade do microcontrolador foi executada com sucesso. Usando transferência de dados através de pacote de protocolo IP, foi realizada a conexão com Wi-fi em 184 ms e um atraso de 3305 ms para a comunicação com o banco de dados, como apresentado nas Tabelas 1 e 2.

Tabela 3: Tempo de atraso da conexão do Módulo ESP8266 com o banco de dados após a conexão wi-fi.

Endereço IP	Tipo	Latência
192.168.0.108	Wi-fi	3305 ms

Fonte: Autoria própria

A transferência de pacotes de dados do módulo para o banco de dados foi feita através de protocolos de texto HTTP. Após a conexão do módulo ao banco de dados, houve um atraso de 2983 ms para rodar o primeiro log de status do LED, que permanecia em modo desligado. Ao acende o LED o status do banco é alterado de 0 (zero) para 1 (um) com um atraso de 13953 ms.

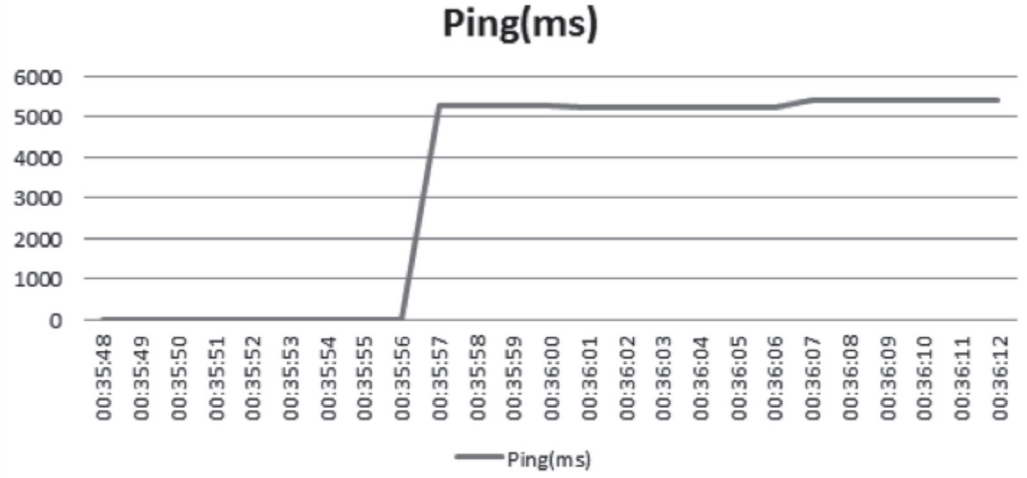
Tabela 4: Log com o banco de dados e status do LED, a partir da conexão com banco de dados.

Endereço IP	Protocolo	Status	Latência
192.168.0.108	HTTP	0	2983 ms
192.168.0.108	HTTP	1	13953 ms

Fonte: Autoria própria

A quantidade total de *payload* (quantidade de carga transmitida) foi 160 bytes, e a variação do *ping* (transmissão do pacote de dados, ida e volta) ficou entre 5246 ms - 5420 ms.

Gráfico 1: Ping X Tempo.



Fonte: Autoria própria.

4. CONCLUSÃO

Com a facilidade da implementação desta API e da biblioteca para o microcontrolador, foi possível obter resultados satisfatório. Apesar dos exemplos serem simples, a resposta da aplicação foi de acordo com o esperado, devido à própria API garantir o tempo de resposta em milissegundos. Um dos recursos que não foi explorado nesta aplicação foi a autenticação de acesso ao banco de dados, para deixar a conexão mais segura. O próprio *firebase* dispõe em sua API uma forma de autenticar o acesso, mas isso também poderia ser implementado por outras bibliotecas. Portanto, os objetivos deste trabalho foram alcançados. Com este exemplo de demonstração é possível dizer que uma aplicação IOT realiza ações em tempo real de onde o usuário estiver, basta ter conexão com a internet.

REFERÊNCIAS

CURVELLO, ANDRÉ. **Apresentando o módulo ESP8266** - (2015). Disponível em: <https://www.embarcados.com.br/modulo-esp8266/#Descricao-dos-pinos>, acessado em: 07/02/2019.

FERREIRA, HIRO G. C. **Arquitetura de Middleware para Internet das Coisas -** (2014). Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGEE.DM - 576/2014, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 125p.

KARZEL Daniel, MARGINEAN Hannelore, TRAN Tuan-si, traduzido por LIMA Tulius. **Uma Arquitetura de Referência para a Internet das Coisas - parte 1** – (2016). Disponível em: <https://www.infoq.com/br/articles/internet-of-things-reference-architecture> acesso em: 29/01/2019.

MOMOTE, Victor. **Modelos de comunicação para IoT** (2016) – disponível em:< https://www.embarcados.com.br/modelos-de-comunicacao-para-iot/> Acesso em: 01/02/2019.

PARK, Dong-Hwan. **Semantic Open IoT Service Platform Technology** - IEEE World Forum on Internet of Things (WF-IoT), 2014.

PEREIRA, C E Pinto. **A Internet das Coisas (IoT): Cenário e Perspectivas no Brasil e Aplicações Práticas.** VII SRST – Seminário de Redes e Sistemas de Telecomunicações Instituto Nacional de Telecomunicações – INATEL, 2017.

SERAFIM, Edivaldo. **Uma Estrutura de Rede Baseada em Tecnologia IoT para Atendimento Médico a Pacientes Remotos** (Dissertação de Mestrado em Ciência da Computação) - Campo Limpo Paulista, (2014).

SOUZA, A Jane. **NoSQL: Administrando Banco de Dados NoSQL com a Linguagem SQL** - Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2016.

SOUZA, A M da Costa. **Uma nova Arquitetura das Coisas com Análise e Reconhecimento de Padrões e Procedimentos com Big Data** - tese de (doutorado) – Escola Politécnica da Universidade de São Paulo, 2015.

VILELA, Flávio. **Um método de integração de dados armazenados em bancos de dados relacionais e NOSQL.** Dissertação (Mestrado) - Universidade Federal de Goiás, 2015.

WEISER, M., **The Computer for the 21st Century**, Scientific American, vol. 265, no. 3.setembro, 1991.

Data do recebimento: 30 de julho de 2018
Data da avaliação: 18 de novembro de 2018
Data de aceite: 16 de dezembro de 2018

1 Graduando em engenharia mecatrônica, Centro Universitário Tiradentes – UNIT, E-mail: cycerow@gmail.com.
2 Graduando em engenharia mecatrônica, Centro Universitário Tiradentes – UNIT, E-mail: viniciuslag@gmail.com.
3 Professor engenharia mecatrônica, Centro Universitário Tiradentes – UNIT- AL.