

IoT - A Internet das coisas

Willian Marques Freire e Munif Gebara Junior

Resumo—The abstract goes here.

Index Terms—IoT, Internet, Coisas.

I. INTRODUÇÃO

A Internet têm transcorrido por diversas etapas evolucionárias distintas. A primeira fase existente foi quando a Web foi chamada de ARPANET (Advanced Research Projects Agency Network). A segunda fase da Web foi caracterizada pela concentração de todas as empresas, para compartilharem informações na Internet com intuito de divulgação de produtos e serviços. Seguido por uma terceira evolução, que mudou a Web, de um estágio com informações estáticas, para informações transacionais, nas quais produtos e serviços são comercializados totalmente online. Após todas estas evoluções, surge a quarta etapa, onde é criado o conceito de Web social e experiência do usuário, na qual empresas como Facebook e Twitter se tornaram famosas e profícuas, ao permitir que pessoas se comuniquem e compartilhem informações [1, p. 6].

Considerando estas evoluções tecnológicas, IoT ou Internet das coisas, algumas vezes referida como a Internet dos objetos, também têm revolucionado, mudando o modelo de *hardware* computacional, que nasceu a aproximadamente 40 anos atrás. Dentre todas as fases diferentes de modelos de *hardware* que existiram, comprova-se que a cada revolução IoT, surge a necessidade de modificar este modelo [3, p. 6]. Devido a diversos estudos visando o melhor desenvolvimento e aproveitamento do modelo cliente-servidor, surgiram vários protocolos para transmissão de dados, e o protocolo principal utilizado atualmente é o HTTP (Hypertext Transfer Protocol), encontrando-se por padrão em praticamente quase todos os navegadores atuais. Existem ainda outros protocolos como FTP (File Transfer Protocol) para transferência de arquivos, IMAP (Internet Message Access Protocol) para envio de Mensagens, entre outros.

Aproveitando-se destes protocolos, também surgiram especificações e arquiteturas, para que aplicações possam ser disponibilizadas como serviços. Dentre elas encontra-se o REST (Representational State Transfer) ou Transferência de Estado representacional, que segundo Fielding [20], é uma abstração da arquitetura World Wide Web, um estilo arquitetural, que consiste em um conjunto coordenado de restrições aplicadas a componentes, conectores e elementos de dados dentro de um sistema de hipermídia distribuído, entretanto, é um assunto para um próximo trabalho.

Considerando que o IoT representa a próxima evolução da Internet, melhorando a capacidade de coletar, analisar e

distribuir dados, faz-se com que os mesmos possam ser transformados em informação. Atualmente existem projetos IoT em desenvolvimento, que prometem fechar a lacuna entre ricos e pobres, melhorando a distribuição dos recursos mundiais para aqueles que precisam deles, e ajudando a entender a sociedade atual para que possa ser mais proativa e menos reativa [1, p. 2]. Em 2003, haviam aproximadamente 6.3 bilhões de pessoas vivendo no mundo, e aproximadamente 500 milhões de dispositivos conectados à internet. O crescimento de smartphones e tablets, elevou o número de dispositivos conectados a Internet para aproximadamente 12,5 bilhões em 2010 [1, p. 3].

Anteriormente, em janeiro de 2009, uma equipe de pesquisadores chinesa, finalizou um estudo sobre os dados de roteamento da internet em intervalos de seis meses, e foi descoberto pelos mesmos, que a Internet dobra de tamanho a cada 5,32 anos (EVERS, 2003). Na figura 1, é possível observar o refinamento desta pesquisa. Devido a este crescimento no número de dispositivos conectados a internet, no artigo Internet of Things volume III, publicado pela empresa Dzone (empresa que faz publicações online sobre tecnologia) no ano de 2017, foi feito uma pesquisa interna na área de IoT. Na mesma, encontram-se 797 profissionais que são responsáveis por esta área, e foi questionado sobre as maiores preocupações quando se trata de IoT. O centro de preocupação estava pela segurança e privacidade, pois os mesmos, estavam interessados em IoT para o contexto empresarial de suas empresas. A falta de padrões, era um terço na coluna "muito preocupada" chegando a 34% dos envolvidos. Dentre eles, 25% dos entrevistados também desconfiam da conectividade e do baixo custo de energia, sendo que 24% se preocupam com a manutenção de *hardware* e *software*, e 14% estão apreensivos com o desenvolvimento imaturo e paradigmas de redes [1, p. 4].

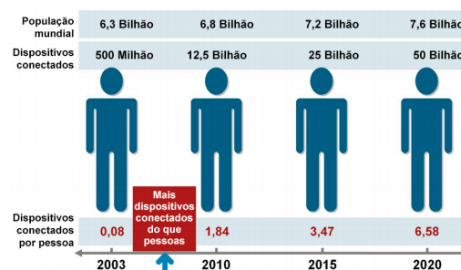


Figura 1. "Nascimento" do IoT entre 2008 e 2009 [1].

O objetivo deste trabalho, é exemplificar o processo de criação de uma estrutura IoT, e demonstrar a resolução da lacuna de configuração de um dispositivo IoT de forma simples. Será desenvolvido durante este trabalho uma estrutura IoT flexível,

que seja capaz de se comunicar através do protocolo HTTP com serviços WEB, que ficarão encarregados de compartilhar as informações recebidas dos dispositivos IoT pela Internet. Este artigo está organizado da seguinte forma: primeiramente será feito uma revisão bibliográfica sobre os assuntos que serão tratados, posteriormente será desenvolvido uma estrutura IoT que vise a fácil configuração da mesma, será feito também uma apresentação de resultados e por fim a conclusão deste artigo.

13 de Maio de 2017

A. Revisão Bibliográfica

1) *IoT*: Atualmente, existem diversos tipos e marcas de placas com circuito integrado e computadores, para desenvolvimento IoT. Em uma nova pesquisa realizada pelo Dzone, foram entrevistados diversas pessoas, para obter um percentual de preferência entre dispositivos. Dentre os mesmos, 53% preferem Raspberry Pi, 28% preferem o Arduino, e 19% não apresentam nenhuma preferência. Nesta pesquisa, foram relatados protocolos mais utilizados dentro do ramo IoT. Dentre os entrevistados, 14% disseram ter preferência por Wifi-Direct em produção, 8% preferem utilizar Bluetooth LE em ambientes que não são de produção. No total, 24% já havia utilizado Wifi-Direct e 23% Bluetooth LE. Um protocolo também citado, é o MQTT (Message Queue Telemetry Transport), que segundo o site FilipeFlop [22], é um protocolo de mensagens leve, criado para comunicação M2M (Machine to Machine), que obteve 33% de preferência de utilização em produção [21, p. 4].

Assim como aplicações web ou móveis, dispositivos físicos também precisam ser consensuais. No entanto, existem desafios adicionais para o IoT. As arquiteturas atuais ainda deixam a desejar em questão de terminais e configuração. Uma segunda preocupação na área de IoT é o tempo necessário para atualização de software para o dispositivo. Dentre as complicações que poder existir, a vulnerabilidade de um dispositivo, pode ser alvo para ataques e invasões. Uma das maneiras de gerenciar a testabilidade do software no dispositivo, é testar isoladamente. Reproduzir o ecossistema em torno do dispositivo e como os usuários interagem garantem uma boa testabilidade. Estudos avançados reconhecem que não se podem identificar, e muito menos testar cada possível cenário de falha. Por este fato, estes estudos concentram cada vez mais na confiabilidade e segurança IoT. O último desafio para esta área, são as identificações e diagnóstico de falhas. Tomar ações rapidamente para resolução de problemas, é um aspecto fundamental no desenvolvimento de dispositivos IoT. Técnicas para integração contínua são utilizados para resolver estes problemas, e uma delas é a Canary Release, que Segundo Danilo Sato [14], é uma técnica para reduzir o risco de introduzir uma nova versão de software na produção, lançando mudanças lentamente para um pequeno subconjunto de usuários, antes de lançá-la em toda a infra-estrutura e torná-la disponível para todos [21, p. 9].

Nas últimas duas décadas, houveram avanços tecnológicos na área computacional, surgindo processadores com mais capacidade, armazenamentos, memória e dispositivos de rede com baixo custo. Atualmente, dispositivos físicos estão sendo

desenvolvidos com mais capacidade computacional, e interligados através da internet de maneira efetiva. A adoção generalizada das tecnologias IoT enriquecem a ideia de computação ubíqua, um conceito que surgiu no final dos anos 80. Mark Weiser, criador deste conceito, escreveu em seu artigo *The Computer for the 21st Century*, que o computador se integra a vida das pessoas de modo que elas não o percebam, todavia, o utilizam. Motivado por esta convicção, Weiser percebeu que em sua época não haviam tecnologias necessárias para que a Computação Ubíqua se tornasse realidade, fazendo com que assim, dedicasse esforços para desenvolver estes meios. [11].

Após alguns anos, no início de 1926, uma revista chamada *Collier* publicou uma entrevista com Nikola Tesla, no qual ele falou sobre suas previsões para as próximas décadas. Entre elas, ele falava de um mundo com máquinas voadoras, energia sem fio e superioridade feminina. Segundo palavras de Tesla, quando a tecnologia sem fio estivesse perfeitamente estabelecida em todo o mundo, o planeta se tornaria um enorme cérebro. Nesta entrevista, ele disse que inclusive os seres humanos seriam capazes de se comunicar uns com os outros de imediato, independente da distância [12].

Além disto, em um artigo feito pelo professor Michael Wooldridge (chefe do Departamento de Ciência da Computação da Universidade de Oxford), definiu um termo utilizado no meio tecnológico, que são os agentes. Segundo ele, um agente é um computador que está situado em algum ambiente, capaz de realizar ações autônomas sobre este, para cumprir seus objetivos delegados. O mesmo torna-se inteligente com as seguintes propriedades: Reatividade (a percepção de seu ambiente, e a capacidade de resposta em tempo hábil sobre mudanças que ocorrem), proatividade (antecipação de problemas futuros), habilidade social (capacidade de interação entre agentes para satisfazer seus objetivos de design) [13].

2) *NodeMCU*: Neste trabalho, será utilizado o NodeMCU para desenvolvimento das aplicações IoT, pelo fato do mesmo conter um módulo WiFi, o que facilitará na comunicação via interface de rede com outros dispositivos. NodeMCU é um firmware baseado em eLua (implementação completa utilizando programação Lua para sistemas embarcados) [16]. O mesmo foi projetado para ser integrado com o Chip WiFi ESP8266 desenvolvido pela empresa Espressif, situada em Shanghai, especializada no ramo de IoT [17]. O NodeMCU utiliza sistema de arquivos SPIFFS (SPI Flash File System) e seu repositório no Github consiste em 98.1% de código na linguagem C - criada em 1972 por Dennis Ritchie [18] e o demais existente em código escrito na linguagem Lua - criada em 1993 por Roberto Ierusalimsky, Luiz Henrique de Figueiredo e Waldemar Celes [19].

ESP8266 é um chip com arquitetura 32 bits, e o seu tamanho está em 5mm x 5mm. Existem diversos módulos parecidos. Dentre eles estão, o ESP-01 que contém 8 conectores, e surgiu para ser utilizado como um módulo para o Arduino, o ESP-07 que contém 16 pinos, antena, cerâmica e conector para antena externa, e o ESP-12E, que conta com 22 pinos, que possibilita a ligação de diversos módulos ao ESP como displays, cartões SD, dentre outros. Um ponto importante nestes módulos, é que eles utilizam 3,3V. Comparado ao Arduino UNO, o NodeMCU se destaca por ter um processador Tensilica LX106, que pode

atingir até 160MHz, e possui uma memória RAM de 20KB e uma memória flash de 4MB. Já o Arduino UNO possui um micro controlador de 16MHz, possui uma memória RAM de 2KB e uma memória flash de 32KB. Outra questão a ser levado em conta, é o custo benefício, pois atualmente é possível encontrar o ESP8266 por até US\$1,70, e um Arduino UNO ultrapassa os 20 dólares[10].

3) *Módulo Wi-Fi*: Devido ao elevado número de cabos necessários para interconectar computadores e dispositivos, surgiu o Wi-Fi. É muito utilizado cabos, entretanto possuem algumas limitações. Um exemplo é o deslocamento dos mesmos, é trabalhoso pelo fato de possuir limite de alcance, e em ambientes com muitos computadores, são necessárias apropriações estruturais. O uso do Wi-Fi tem se tornando comuns não somente em residências, mas também em ambientes corporativos e públicos. Dentre os padrões de Wi-Fi existentes, estão o 802.11b que tem como possibilidade de estabelecimento de conexões nas seguintes velocidades de transmissão: 1 Mb/s, 2Mb/s, 5,5 Mb/s e 11 Mb/s, o 802.11g que surgiu em 2003 e é totalmente compatível com o 802.11b, e possui como atrativo a possibilidade de trabalhar com taxas de transmissão de até 54 Mb/s, o 802.11n que tem como principal característica a utilização de um esquema chamado MIMO ou Multiple-Input Multiple-Output, que é capaz de atingir taxas de transmissão de até 600 Mb/s, e finalmente o 802.11ac, também chamado 5G Wi-Fi, que sua principal vantagem está em sua velocidade, estimada em até 433 Mb/s no modo mais simples, fazendo com que, é possível fazer a rede suportar 6 Gb/s de velocidade na transmissão de dados. Resumidamente, Wi-Fi é um conjunto de especificações para redes locais sem fio baseada no padrão IEEE 802.11, uma abreviatura para "Wireless Fidelity". [4]

4) *Segurança Wi-Fi*: Apesar de todas as facilidades que se encontram ao utilizar tecnologias sem fio como Wi-Fi, assim como todos os sistemas, medidas de segurança devem ser utilizadas para prevenir ataques e invasões. Existem protocolos de segurança que auxiliam na segurança de conexões Wi-Fi, dentre eles se encontram: WEP, WPA, e o WPA2. Wired Equivalent Privacy (WEP) é um algoritmo de segurança que foi criado em 1999 e é compatível com praticamente todos os dispositivos Wi-Fi disponíveis no mercado. Este padrão se torna mais inseguro à medida que o poder de processamento dos computadores aumenta. Pelo fato de conter um número máximo de combinações de senha totalizando 128 bits, é possível descobrir a palavra-passe em poucos minutos por meio de softwares de ataques. Wi-Fi Protected Access ou WPA, surgiu quando o WEP saiu de circulação, e entrou como protocolo-padrão industrial. Adotado em 2003, trazia como novidade a criptografia 256 bits e como segurança adicional, fazia análise de pacotes, para verificar alterações e invasões. Atualmente é utilizado o Wi-Fi Protected Access II ou WPA2, pelo fato de ser o mais seguro. Foi implementado pela Wi-Fi Alliance em 2006, e possui como diferencial a maneira como lida com senhas e algoritmos, excluindo totalmente a possibilidade de um ataque de força bruta. Segundo especialistas, o risco de intrusos em redes domésticas é quase zero. Isto se deve à utilização de duas tecnologias que são utilizadas neste algoritmo, o AES (Advanced Encryption Standard) que é um novo padrão para segurança das informações, e o CCMP (Counter Cipher

Mode), um mecanismo de criptografia que protege os dados que passam pela rede. Devido à complexidade do mesmo, muitos dispositivos, mesmo recentes, não são compatíveis com ele. [8]

5) *Sistema de arquivos*: O NodeMCU utiliza o sistema de arquivos SPIFFS, um sistema de arquivos destinado a dispositivos flash embarcados. SPIFFS é projetado para projetos que são pequenos e memória sem pilha [6]. O projeto pode ser clonado do repositório do mesmo no github.

6) *Ferramenta para desenvolvimento*: Neste projeto será utilizada uma ferramenta chamada ESPlorer. Ela é utilizada para facilitar o desenvolvimento de aplicações LUA para dispositivos ESP8266. Basicamente, é uma IDE (Ambiente de desenvolvimento integrado), que possui diversos facilitadores para melhor visualização do código e envio do código fonte para o dispositivo. Esta ferramenta possui um terminal para debug, e acessos rápidos para executar comandos. Nela também é possível executar comandos de forma rápida. Todos estes fatores beneficiaram na escolha de utilização desta ferramenta para o desenvolvimento do trabalho [7]. Na figura 2 é possível observar estas características.

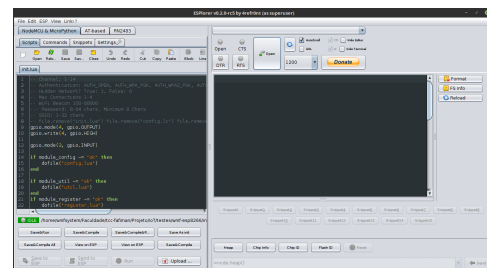


Figura 2. Esplorer [7]

7) *Protocolos TCP e UDP*: O protocolo TCP ou Protocolo de controle de transmissão, é um dos protocolos mais utilizados. Ele é complementado pelo IP (Protocolo da Internet), sendo normalmente chamado de TCP/IP. A versatilidade e robustez do mesmo tornou-o mais adequado para utilização em redes globais, já que são feitas verificações de dados para confirmação de recebimento e entrega sem erros. O TCP é um protocolo que se encontra no nível da camada 4 ou camada de transporte do modelo OSI (sistema de interconexão aberto), que é um modelo de referência da ISO (Organização Internacional para Padronização) dividido em camadas de funções, que são: Aplicação que fornece serviços às aplicações do cliente, Apresentação que fornece criptografia e compressão de dados, além de garantir a compatibilidade entre camadas de aplicação de sistemas diferentes, Sessão que controla as sessões entre aplicações, transporte que controla o fluxo de informação e controle de erros, rede que encaminha pacotes e possui esquema de endereçamentos, dados que controla o acesso ao meio físico de transmissão e erros da camada física, e finalmente a camada física, que define as características do meio físico de transmissão da rede, conectores, interfaces, codificação ou modulação de sinais [23][24].

Outro protocolo que também será utilizado, é o UDP (User Datagram Protocol). Antes de entender a diferença do UDP para o TCP é necessário saber o que é um datagrama. Segundo

a RFC 1594 [25] é uma entidade de dados completa e independente que contém informações suficientes para ser roteada da origem ao destino sem precisar confiar em trocas anteriores entre essa fonte, a máquina de destino e a rede de transporte. O UDP permite que a aplicação envie um datagrama encapsulado em um Pacote IPv4 (ou IPv6, e então enviado ao destino. Ao contrário de TCP, no UDP não há qualquer tipo de garantia que o pacote será entregue.

B. Desenvolvimento

1) *Configurações iniciais no NodeMCU ESP-8266*: Primeiramente, para se utilizar este dispositivo, é necessário fazer o build (compilação) do firmware, conforme a figura 3, pois o mesmo é distribuído sem nenhum sistema operacional. Atualmente, pode-se fazer o build do mesmo utilizando o site <https://nodemcu-build.com> ou clonar o repositório do projeto do github. No caso deste projeto, o build do projeto será feito manualmente. Para incluir bibliotecas neste módulo, é necessário editar o arquivo que se encontra no diretório `app/include/user_modules.h` do projeto e retirar os comentários das bibliotecas que será utilizado.

```
1 #define LUA_USE_MODULES\MQTT
2 // #define LUA_USE_MODULES_COAP
3 // #define LUA_USE_MODULES_USG
```

Figura 3. Compilação do firmware

Duas possibilidades disponível pelo módulo NodeMCU, são ativar o suporte TLS e o debug (depurador) quando em execução. Para utilizar o suporte TLS, é necessário descomentar a opção `CLIENT_SSL_ENABLE` que se encontra dentro do arquivo `user_config.h` no diretório `app/include`, e para ativar o modo debug em tempo de execução, e descomentar a linha em que se encontra `#define DEVELOP_VERSION`.

Por padrão, o build do firmware é gerado com suporte a variáveis com ponto flutuante, entretanto, isto ocupa mais memória, e para reduzir a quantidade de memória utilizada, deve-se comentar em que é definido a constante `LUA_NUMBER_INTEGRAL` dentro do arquivo `user_config.h`.

2) *Gerando e gravando o firmware*: Após ajustar as devidas configurações inicial, é necessário compilar e gravar o firmware no dispositivo, conforme a figura 4. Neste trabalho, será utilizado a ferramenta `esptool` para gravar o mesmo. Esta ferramenta pode ser encontrada no repositório da Espressif (empresa que desenvolveu o NodeMCU) no github, e foi iniciada por Fredrik Ahlberg, um dos envolvidos no projeto, e atualmente é mantido por Angus Gratton e pela comunidade. E para compilar os códigos fontes, deve-se primeiramente executar os seguintes comandos dentro do diretório do projeto.

Como este projeto utiliza o docker, uma infraestrutura independente de plataforma [5], primeiramente é atualizado o container e posteriormente é executado um comando para compilar o projeto. Após compilado, deve ser executado os comandos conforme a figura 5, utilizando a ferramenta `esptool`. Estes comandos serão utilizados para gravar o firmware no ESP8266.

```
1 $ docker pull marcelstoer/nodemcu-build
2 $ sudo docker run --rm -ti -v 'pwd':/opt
3 /nodemcu-firmware
4 marcelstoer/nodemcu-build
```

Figura 4. Compilação do firmware

```
1 $ sudo python esptool.py
2   --port="/dev/ttyUSB1"
3   erase_flash
4
5 $ sudo python ./esptool.py
6   --port /dev/ttyUSB0
7   write_flash
8   0x00000
9   ../nodemcu-firmware/bin/0x00000.bin
10  0x10000
11  ../nodemcu-firmware/bin/0x10000.bin
12
13 $ sudo python ./esptool.py
14   --port="/dev/ttyUSB0"
15   write_flash -fm=dio
16   -fs=16m 0x00000
17   ./../nodemcu-firmware/bin
18   /nodemcu_integer__20170510-0106.bin
```

Figura 5. Compilação do firmware

Os comandos da figura 5 estão divididos em 3 partes. Primeiramente é removido todo o firmware do dispositivo, posteriormente é gravado nos endereçamentos `0x00000` e `0x10000` os arquivos com bibliotecas padrões utilizadas no firmware. Após isto é gravado o firmware com as bibliotecas extras selecionadas para inclusão no dispositivo.

3) *Access Point para configuração*: Assim como todo dispositivo, neste trabalho será desenvolvido uma central de configuração. Como este dispositivo foi desenvolvido com foco na utilização e conexão Wi-fi, será criado uma central para configuração sobre qual dispositivo Wireless o mesmo se conectará, e para isto, como a maioria destes dispositivos possuem regras de segurança e senhas, deverá conter também a opção para que seja configurado a senha do dispositivo.

Um dos recursos que o ESP8266 provê, é o desenvolvimento de um AP (Access Point ou ponto de acesso), que permite interligar duas ou mais redes sem fio. Dentre as diversas funções de um AP estão: repetir um sinal e transformar um sinal de um cabo em sinal sem fio [9]. Entretanto, neste trabalho será utilizado somente como central para configuração do dispositivo.

Primeiramente é necessário fazer o NodeMCU trabalhar com o padrão Access Point. Ele provê um facilitador para que o dispositivo trabalhe como Access Point e como cliente. Para isto é necessário fazer conforme a figura 6. Neste trabalho será criado inicialmente um arquivo chamado `init.lua`, onde se encontrará o código fonte que será executado ao iniciar o dispositivo.

Uma outra possibilidade ao utilizar o NodeMCU, é a criação de um pequeno servidor em uma de suas portas. Neste caso, será criado um servidor TCP conforme a figura 7, para que quando alguém se conectar no dispositivo em uma determinada porta padrão, esteja disponível a central de configuração

```
1 wifi.setmode(wifi.STATIONAP)
```

Figura 6. Configurando modo Access Point.

caso o dispositivo não esteja conectado em nenhum roteador Wireless.

```
1 srv = net.createServer(net.TCP)
```

Figura 7. Criando Servidor TCP.

Uma das questões encontradas neste trabalho, foi a forma de como manter estas configurações em funcionamento, pois se ocorrer algo que faça com que o dispositivo desligue, seria interessante que o mesmo mantivesse as configurações salvas, para que não seja necessário a reconfiguração do Wi-Fi. Para isto, será utilizado o sistema de arquivos do ESP8266, para criação de um arquivo `config.lua`, conforme a figura 11, e sempre que o dispositivo iniciar, será feito a verificação da existência do arquivo de configurações para conexão do Wi-Fi, e se caso não existir este arquivo, será iniciado a central de configuração do dispositivo.

Para configurar um Access Point, é necessário algumas considerações como: `ssid` (Nome da conexão), `pwd` (Senha da conexão), `auth` (Tipo de autenticação), `channel` (Canal que será liberado no dispositivo Wi-Fi), `hidden` (indica se é visível), `max` (máximo de conexões simultâneas) e `beacon` (intervalo de tentativas de conexão do dispositivo). Quando é encontrado o arquivo, é feito uma leitura do mesmo, e neste trabalho, está sendo salvo no padrão usuário e senha separados por um espaço em branco. Os demais códigos desenvolvidos, podem ser encontrados no apêndice deste trabalho. No arquivo `init.lua` deverá ser feito a importação dos mesmos, pois serão divididos da seguinte forma: `init.lua` (arquivo inicial), `config.lua` (arquivo que terá o código de configuração) e `util.lua` (arquivo que terá funções utilitárias). Para importar os demais módulos, foi feito conforme a figura 8.

```
1 dofile("config.lua")
2 dofile("util.lua")
3 dofile("register.lua")
```

Figura 8. Importação dos módulos

II. RESULTADOS

O NodeMCU ESP8266 possui um módulo Wi-Fi, o que possibilitou a conexão à rede de internet. O mesmo possui um sistema de arquivos, o que permitiu as configurações serem salvas no dispositivo, para que se caso ocorra alguma falha, ou até mesmo o dispositivo seja desligado, não seja necessário fazer novamente as configurações iniciais. Foi implementado uma regra no dispositivo, que faz a verificação de conexão a internet dentro de 20 segundos, e se ocorrer alguma falha na conexão, será deletado o arquivo de configurações e reiniciado o dispositivo, liberando assim, o acesso à configuração de rede.

Como este dispositivo tem suporte a protocolos TCP e UDP, é aberto possibilidades para integração com outros dispositivos ou sistemas que utilizam este protocolo. Foram realizados diversos testes com o mesmo, dentre eles utilizando o GPIO [26] para utilização de pinos digitais, SJSON para serialização de Objetos utilizando textos no padrão JSON [27] e o Timer para executar funções em determinado tempo [28].

III. CONCLUSÕES E TRABALHOS FUTUROS

Utilizando-se destes recursos disponíveis no NodeMCU ESP8266, é possível montar uma estrutura IoT flexível e de simples configuração. Atualmente existem outros dispositivos embarcados semelhantes que são acessíveis. Entretanto, neste trabalho foi utilizado este, pelo fato de ter um custo-benefício acessível, e de estar sendo utilizado pela comunidade. Foi utilizado a linguagem LUA para desenvolver neste dispositivo, entretanto, existem alguns projetos que possibilitam o desenvolvimento em outras linguagens. Como neste trabalho foi desenvolvido uma estrutura IoT de fácil configuração e com tecnologias como TCP e UDP, foi aberto possibilidades para o desenvolvimento de sistemas que se integrem com o mesmo, utilizando a rede de internet, e trabalhos futuros podem visar esta possibilidade. Sistemas complexos de automação residencial ou industrial podem ser resolvidos acoplando estas tecnologias, integrando o dispositivo com a rede de internet de forma fácil, foi resolvida.

APÊNDICE A
CÓDIGO FONTE CONFIGURACIONAL: CONFIG.LUA

```

1 function configureWifi()
2   local ap = {}
3   wifi.sta.getap(
4     function(t)
5       i = 0
6       for ssid, v in pairs(t) do
7         ap[i] = ssid
8         i = i + 1
9       end
10    end)
11  srv:listen(
12    80,
13    function(conn)
14      conn:on(
15        "receive",
16        function(conn, payload)
17          listaAp = "SSID: <select name='ssid'>"
18          print("Request Configuration")
19          for i = 0, tablelength(ap), 1 do
20            if ap[i] ~= nil then
21              listaAp = listaAp
22                .. "<option value='"
23                .. ap[i] .. "'>"
24                .. ap[i] .. "</option>"
25            end
26          end
27          listaAp = listaAp .. "</select><br/>"
28
29          local _GET = getParamsUrl(payload)
30
31          if (_GET ~= nil) then
32            if (_GET.ssid ~= nil
33              and _GET.password ~= nil) then
34              print("Connecting to wifi
35                and creating configuration...")
36              print("SSID: "
37                .. _GET.ssid)
38              print("Password: "
39                .. _GET.password)
40
41              for i = 0, tablelength(ap), 1 do
42                if ap[i] ~= nil then
43                  if (string
44                    .find(ap[i], _GET.ssid)) then
45                    user = ap[i]
46                  end
47                end
48              end
49
50              password = _GET.password
51              wifi.sta.config(user, password)
52              wifi.sta.connect()
53
54              if file.open("config.lc", "w") then
55                file.writeline(user
56                  .. " " .. password)
57                file.close()
58              end
59
60              node.restart()
61              node.chipid()
62            end
63          end
64          conn:send([[... HTML Whatever...]])
65          conn:on("sent", function(conn)
66            conn:close()
67            collectgarbage()
68          end)
69        end)
70    end

```

Figura 9. Módulo Utilitário

APÊNDICE B
CÓDIGO FONTE UTILITÁRIO: UTIL.LUA

```

1 function split(str, pat)
2   local t = {}
3   -- NOTE: use {n = 0} in Lua-5.0
4   local fpat = "(.-)" .. pat
5   local last_end = 1
6   local s,
7     e,
8     cap = str:find(fpat, 1)
9   while s do
10     if s ~= 1 or cap ~= "" then
11       table.insert(t, cap)
12     end
13     last_end = e + 1
14     s,
15     e,
16     cap = str:find(fpat, last_end)
17   end
18   if last_end <= #str then
19     cap = str:sub(last_end)
20     table.insert(t, cap)
21   end
22   return t
23 end
24
25 local unescape = function(s)
26   s = string.gsub(s, "+", " ")
27   s = string.gsub(
28     s,
29     "%%(%%x%%x)",
30     function(h)
31       return string.char(tonumber(h, 16))
32     end
33   )
34   return s
35 end
36
37 function tablelength(T)
38   local count = 0
39   for _ in pairs(T) do
40     count = count + 1
41   end
42   return count
43 end
44
45 function getParamsUrl(request)
46   local buf = ""
47   local
48     -, -,
49     method,
50     path,
51     vars = string
52     .find(request, "([A-Z]+) (.)?(.) HTTP")
53   if (method == nil) then
54     -, -,
55     method,
56     path = string.find(request, "([A-Z]+) (.)?
57     HTTP")
58   end
59   local _GET = {}
60   if (vars ~= nil) then
61     for k, v in string
62       .gmatch(vars, "(%w+)=(%w+)&*" ) do
63       _GET[k] = unescape(v)
64     end
65   end
66   return _GET
67 end

```

Figura 10. Módulo Configuracional

APÊNDICE C

CÓDIGO FONTE INICIAL: INIT.LUA

```

1  srv = net.createServer(net.TCP)
2
3  if file.exists("config.lua") == true then
4
5      print("Open configuration ...")
6
7      if file.open("config.lua") then
8
9          local corte =
10             split(file.read(), " ")
11             user = corte[1]:gsub("%s+", "")
12             password = corte[2]:gsub("%s+", "")
13
14             print("Connecting in ip with user: "
15                 .. user
16                 .. " and password: "
17                 .. password)
18             wifi.sta.config(user, password)
19             wifi.sta.connect()
20
21         end
22     else
23
24         wifi.ap.config({
25             ssid = "NodeMcuEsp8266"
26             .. node.chipid(),
27             pwd = nil,
28             auth = AUTH_OPEN,
29             channel = 6,
30             hidden = 0,
31             max = 4,
32             beacon = 100
33         })
34
35         wifi.ap.setip({
36             ip = "192.168.10.1",
37             netmask = "255.255.255.0",
38             gateway = "192.168.10.1"
39         })
40
41         wifi.ap.dhcp.config(
42             { start = "192.168.10.2" }
43         )
44
45         tmr.alarm(
46             1,
47             1000,
48             1,
49             function()
50                 if wifi.ap.getip() == nil then
51                     print("Connecting ...")
52                 else
53                     print("Connected in "
54                         .. wifi.ap.getip())
55                     configureWifi()
56                 end
57             end
58         )
59         tmr.stop(1)
60     end
61 end
62
63 end

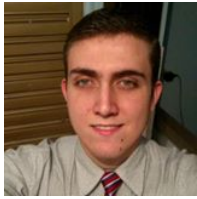
```

Figura 11. Central de Configuração

REFERÊNCIAS

- [1] Dave Evans. *The Internet of Things How the Next Evolution of the Internet Is Changing Everything*, vol 1, pp. 2-4, 2011

- [2] Willian Marques Freire e Munif Gebara Júnior. *Micro-serviços*, vol 1, pp. 1-3, 2017
- [3] John Esposito. *The Dzone Guide to Internet of Things*, vol 3, pp. 1-6, 2016
- [4] Emerson Alecrim. *O que é Wi-Fi (IEEE 802.11)?* 2013 [Online] Disponível: <https://www.infowester.com/wifi.php#80211>. [Acesso: 20-Mai-2017]
- [5] Docker. *Docker 2017* [Online] Disponível: <https://www.docker.com>. [Acesso: 11-Jun-2017]
- [6] Peter Andersson. *SPIFFS (SPI Flash File System)* 2013 [Online] Disponível: <https://github.com/pellepl/spiffs>. [Acesso: 11-Jun-2017]
- [7] ESPlorer. *ESPlorer Integrated Development Environment (IDE) for ESP8266 developers* 2013 [Online] Disponível: <https://github.com/4refr0nt/ESPlorer>. [Acesso: 11-Jun-2017]
- [8] Felipe Demartini. *WEP, WPA, WPA2: o que as siglas significam para o seu Wi-Fi?* 2013 [Online] Disponível: <https://www.tecmundo.com.br/wifi/42024-wep-wpa-wpa2-o-que-as-siglas-significam-para-o-seu-wifi-hm>. [Acesso: 03-Jun-2017]
- [9] Mundo Max. *Qual a diferença entre um Roteador Wireless e um Access Point?* 2011 [Online] Disponível: <http://www.mundomax.com.br/blog/informatica/qual-a-diferenca-entre-um-roterador-wireless-e-um-access-point/>. [Acesso: 03-Jun-2017]
- [10] Irving Souza Lima. *NodeMCU (ESP8266) o módulo que desbancou o Arduino e facilitará a Internet das Coisas*. 2016 [Online] Disponível: <http://irving.com.br/esp8266/nodemcu-esp8266-o-modulo-que-desbancou-o-arduino-e-facilitara-a-internet-das-coisas/> [Acesso: 05-Jun-2017]
- [11] Mark Weiser. *The Computer for the 21st Century*, vol 1, pp. 1-2, 1991
- [12] John B. Kennedy. *WHEN WOMAN IS BOSS*, 1926 [Online] Disponível: <http://www.tfcbooks.com/tesla/1926-01-30.htm>. [Acesso: 10-Mai-2017]
- [13] Michael Wooldrige e Nicholas R. Jennings. *Intelligent agents: theory and practice*, vol 1, pp. 1-3, 1995
- [14] Danilo Sato. *CanaryRelease*. 2014 [Online] Disponível: <https://martinfowler.com/bliki/CanaryRelease.html>. [Acesso: 10-Abr-2017]
- [15] NodeMCU. *NodeMCU 2.0.0*. [Online] Disponível: <https://github.com/nodemcu/nodemcu-firmware>. [Acesso: 10-Dez-2017]
- [16] ELua. *What is eLua*. [Online] Disponível: <http://www.eluaproject.net/overview>. [Acesso: 19-Nov-2017]
- [17] Systems. *Expressif Systems*. [Online] Disponível: <http://espressif.com/company/contact/pre-sale-questions>. [Acesso: 11-Nov-2016]
- [18] William Stewart. *C Programming Language History*. [Online] Disponível: http://www.livinginternet.com/i/iw_unix_c.htm. [Acesso: 15-Mai-2017]
- [19] Lua. *Authors*. [Online] Disponível: <http://www.lua.org/authors.html>. [Acesso: 5-Fev-2017]
- [20] Roy Thomas Fielding. *Representational State Transfer (REST)*. [Online] Disponível: http://www.ics.uci.edu/~fielding/pubs/dissertation/-rest_arch_style.htm. [Acesso: 29-Mai-2017]
- [21] Cate Lawrence. *Internet of Things Applications, Protocols, and Best Practices*, vol 4, pp. 1-10, 2017
- [22] Filipeflop. *Controle e monitoramento IoT com NodeMCU e MQTT*. 2016 [Online] Disponível: <http://blog.filipeflop.com/wireless/controle-monitoramento-iot-nodemcu-e-mqtt.html>. [Acesso: 10-Abr-2017]
- [23] Pedro Pinto. *Redes – Sabe o que é o modelo OSI?* 2010 [Online] Disponível: <https://pplware.sapo.pt/tutoriais/networking/redes-sabe-o-que-e-o-modelo-osi>. [Acesso: 17-Jun-2017]
- [24] Vinton G. Cerf, Robert E. Kahn. *A Protocol for Packet Network Intercommunication*, IEEE Transactions on Communications, vol. 22, pp. 637-648, 1974
- [25] April N. Marine et Al. *RFC 1594*, 1994 [Online] Disponível: <https://tools.ietf.org/html/rfc1594>. [Acesso: 17-Jun-2017]
- [26] NodeMCU. *GPIO Module?* 2014 [Online] Disponível: <https://nodemcu.readthedocs.io/en/master/en/modules/gpio/> [Acesso: 17-Jun-2017]
- [27] NodeMCU. *SJSON Module* 2017 [Online] Disponível: <https://nodemcu.readthedocs.io/en/master/en/modules/sjson/> [Acesso: 17-Jun-2017]
- [28] NodeMCU. *Timer Module* 2014 [Online] Disponível: <https://nodemcu.readthedocs.io/en/master/en/modules/tmr/> [Acesso: 17-Jun-2017]



Willian Marques Freire Possui Ensino Médio completo pelo Colégio Estadual Rosa Delúcia Calsavara (2013). Atualmente é Desenvolvedor de Software da Gumga Tecnologia da Informação S/A. Tem experiência na área de Ciência da Computação.



Munif Gebara Júnior Possui graduação em Ciência da Computação pela Universidade Estadual de Maringá (1997) e mestrado em Engenharia Elétrica e Informática Industrial pela Universidade Tecnológica Federal do Paraná (2001). Atualmente é professor da Fundação Faculdade de Filosofia Ciências e Letras de Mandaguari e professor de ensino superior da Faculdade de Tecnologia e Ciências do Norte do Paraná Ltda e desenvolvedor.