IoT - A Internet das coisas

Willian Marques Freire e Munif Gebara Junior

***Resumo*—The abstract goes here.**

***Index Terms*—IEEE, IEEEtran, journal, LaTeX, paper, template.**

I. Introdução

IOT ou Internet das coisas, algumas vezes referida como a Internet dos objetos, está gerando uma revolução tecnológica em diversas áreas e inclusive na humanidade. Considerando que o IoT representa a próxima evolução da Internet, dando um salto na 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].

Segundo o Artigo Internet of Things volume III, na empresa Dzone (empresa que faz publicações online sobre tecnologia) encontram-se 797 profissionais que são responsáveis pela área de IoT, e uma das maiores preocupações da mesma é este âmbito. Em um inquérito realizado pela mesma, foi perguntado 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 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].

Em 1999 foi fundado um grupo chamado MIT (Massachusetts Institute of Technology), que tem trabalhado no campo de identificação de frequência de rádio, em rede (RFID) e tecnologias de sensores emergentes. Em 2003, haviam aproximadamente 6,3 bilhões de pessoas vivendo no mundo, e aproximadamente 500 milhões de dispositivos conectados à internet. O crescimento mastodôntico de smartphones e tablets, elevou o número de dispositivos conectados a Internet para aproximadamente 12,5 bilhões em 2010 [1, p. 3].

Em janeiro de 2009, uma equipe de pesquisadores chinesa, finalizou um estudo sobre os dados de roteamento da internet em intervalos de seis meses, entre dezembro de 2001 e dezembro de 2006. 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.

Faculdade de Filosofia Ciências e Letras de Mandaguari é uma fundação situada em Mandaguari no Paraná região sul brasileira, na rua Rene Taccola, 152 - Centro Site: (see http://www.fafiman.br/index.html).

Artigo realizado em 2017.

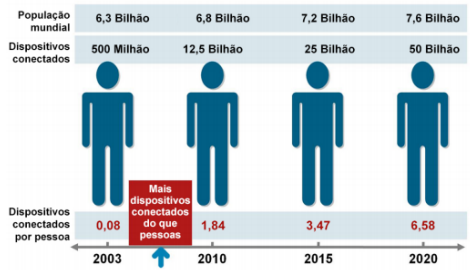

Figura 1. "Nascimento"do IoT entre 2008 e 2009 [1].

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, compartilharem informações na Internet para 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, Twitter se tornaram famosas e profícuas, ao permitir que pessoas se comuniquem e compartilhem informações [1, p. 6].

IoT Tem mudado o modelo de hardware computacional, que nasceu a aproximadamente 40 anos atrás. Considerando todas as fases diferentes de modelos de hardware que existiram, dentre elas micro-computadores pessoais, e hoje computadores conectados na nuvem expandindo o modelo cliente-servidor para novos paradigmas, comprova que a cada revolução IoT surge a necessidade de modificar este modelo [3, p. 6]. Devido a diversos estudos visando melhor desenvolvimento e aproveitamento do modelo cliente-servidor, surgiram vários protocolos para transmissão de dados. 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 (IInternet Message Access Protocol) para envio de Mensagens, entre outros.

Aproveitando-se destes protocolos, também surgiu especificações e arquiteturas, para que aplicações possam se 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 restições arquiteturais aplicadas a componentes, conectores e elementos de dados

dentro de um sistema de hipermídia distribuído, entretanto, é um assunto para o próximo artigo que será falado sobre Micro-serviços.

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.

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 entrevistadors diversas pessoas, para obter um percentual de preferência entre os dispositivos. Dentre os mesmos, 53% preferem Rapberry 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 Bluetooh LE em ambientes que não são de produção. No total, 24% já havia utilizado Wifi-Direct e 23% Bluetookh 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), 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. Além destes aspectos, complexos sistemas sociotécnicos cercam a humanidade. 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. O ultimo desafio para área de IoT, são as identificações e dignosticação de falhas. Tomar ações rapidamente para resolução de problemas, é um aspecto fundamental no desenvolvimento de dispositivos IoT. Técnicas para integração continua são utilizados para resolver estes problemas, 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 lança-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 de processamento, armazenamentos, memória e dispositivos de rede com baixo custo. Atualmente, dispositivos físicos estão sendo desenvolvidos com mais capacidade compitacional, e interligados atrabés da internet de maneira efetiva. A adoção generalizada das tecnologias IoT enriquecem a idéia de computação ubíqua, um conceito que surgiu no final dos anos 80. Segundo Mark Weiser, criador do conceito citado, 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, entretanto, o utilizem. 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].

Em um artigo feito pelo professor Michael Wooldridge, chefe do Departamento de Ciência da Computação da Universidade de Oxford, o mesmo definiu um termo utilizado no meio tecnológico, que são os agentes. Sendo 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 respota 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].

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 entrvista, ele disse qua inclusive os ser humano seria capaz de se comunicar uns com os outros de imediato, independente da distância [12].

2) NodeMCU: Neste trabalho, será utilizado o NodeMCU para desenvolvimento das aplicaçoes IoT, pelo fato do mesmo conter um módulo WiFi, o que facilitará na comunicação via interface de rede com os micro-serviços. NodeMCU é um firmware baseado em eLua - uma 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 Shangai, 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 Ierusalimschy, 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 WiFi. 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 nao 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 p 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 203 e é totalmente compatível com o 802.11b, e possui como atrativo a possibilidade de trabalhar com taxas de transmiss ao 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 WiFi, 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 trasmiss ao de dados. Resumidamente, Wi-Fi é um conjunto de especificaçoes para redes locais sem fio baseada no padrão IEEE 802.11, uma abreviatura para Ẅireless Fidelity.̈ [4]

4) Segurança Wi-Fi: Apesar de todas as facilidades que se encontram ao utilizam tecnologias sem fio como Wi-Fi, assim como todo sistemas, medidas de segurança devem ser utilizadas para previnir 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 prativamente todos os dispositivos Wi-Fi disponível no mercado. Este padrão se torna mais inseguro à medida que o poder de processamento dos computadores aumenta. Pelo fato de conter um numero 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 encriptação 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 especialistras, o riscos de intrusos em redes domésticos é quase zero. Isto se deve a 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 encriptação que protege os dados que passam pela rede. Devido a complexidade do mesmo, muitos dispositivos, mesmos 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 utilizado uma ferramenta chamada ESPlorer. ELa é utlizada para facilitar o desenvolvimento de aplicações LUA para dispostivos 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.

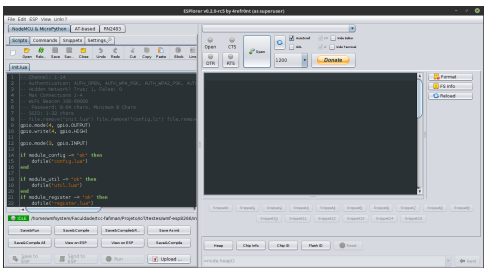

Figura 2. Esplorer [7]

7) Protocolos TCP e UDP: O protocolo TCP ou Protocolo de controle de transmissão, é um dos protolocos 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 feito verificação 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 encriptação 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 o meio físico de tansmissão 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].

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 I-B1, pois o mesmo é distribuido 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.

```
#define LUA\_USE_MODULES\_MQTT
// #define LUA\_USE\_MODULES_COAP
// #define LUA\_USE\_MODULES\_U8G
```

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 so 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 I-B2. 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 Ahlbeg, 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.

```
docker pull marcelstoer/nodemcu-build

sudo docker run --rm -ti -v `pwd`
:/opt/nodemcu-firmware
marcelstoer/nodemcu-build
```

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 I-B2, utilizando a ferramenta esptool. Estes comandos serão utilizados para gravar o firmware no ESP8266.

Os comandos acima estão divididos em 3 partes. Primeiramente é removido todo o firmware do dispositivo, posteriormente é gravado nos endereçamentos 0x00000 e 0x10000

```
sudo python esptool.py
  --port="/dev/ttyUSB1"
  erase_flash

sudo python ./esptool.py
  --port /dev/ttyUSB0
  write_flash
  0x00000
    ../nodemcu-firmware/bin/0x00000.bin
  0x10000
    ../nodemcu-firmware/bin/0x10000.bin

sudo python ./esptool.py
  --port="/dev/ttyUSB0"
  write_flash -fm=dio
  -fs=16m 0x00000
  ./../nodemcu-firmware/bin
  /nodemcu_integer__20170510-0106.bin
```

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 necessario 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 I-B3. Neste trabalho será criado inicialmente um arquivo chamado init.lua, onde se encontrará o código fonte que será executado ao iniciar o dispositivo.

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

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 I-B3, para que quando alguém se conectar no dispositivo em uma determinada porta padrão, esteja disponível a central de configuração caso o dispositivo não esteja conectado em nenhum roteador Wireless.

```
srv = net.createServer(net.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.lc conforme a figura I-B3, 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.

```
if file.exists("config.lc") == true then
  print("open configuragion...")
  if (file.open("config,lc")) then
    local corte = split(file.read(), " ")
    user = corte[1]:gsub("%s+", "")
    password = corte[2]:gsub("%s+", "")

    print("Connecting in ip with user: "
        .. user .. " and password: "
        .. password)
    wifi.sta.config(user, password)
    wifi.sta.connect()
  end
else
  wifi.ap.config({
    ssid = "NodeMcuEsp8266"
        ..node.chipid(),
        pwd = nil, auth = AUTH_OPEN,
        channel = 6, hidden = 0,
        max = 4, beacon = 100})
    wifi.ap.setip({ip = "192.168.10.1",
        netmask = "255.255.255.0",
        gateway = "192.168.10.1"})
    wifi.ap.dhcp.config(
        {start = "192.168.10.2"})

    tmr.alarm(
        1,
        1000,
        1,
        function()
          if wifi.ap.getip() == nil then
              print("Connecting...")
          else
            print("Connected in "
                  .. wifi.ap.getip())
            configureWifi()
          end
          tmr.stop(1)
        end
    )
end
```

Para configurar um Access Point, é necessário algumas configurações como: ssid (Nome da conexão), pwd (Senha da conexao), auth (Tipo de autenticação), channel (Canal que será liberado no dispositivo Wi-Fi), hidden (indica se é visível), max (máximo de conexões simutâneas) e beacon (intervalo de tentativas de conexão do dispositivo). Quando é encontrado o arquivo, é feito uma leitura do arquivo, e neste trabalho, está sendo salvo no padrão usuário e senha separados por um espaço em branco, por isto está sendo separado em duas palavras ao ler a primeira linha do arquivo. 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, deve-se fazer conforme a figura I-B3.

```
dofile("config.lua")
dofile("util.lua")
dofile("register.lua")
```

II. Conclusion

The conclusion goes here.

Apêndice A
Código fonte Configuracional: config.lua

```
function configureWifi()
  local ap = {}
  wifi.sta.getap(
    function(t)
        i = 0
        for ssid, v in pairs(t) do
            ap[i] = ssid
            i = i + 1
        end
    end
  )
  srv:listen(
    80,
    function(conn)
        conn:on(
          "receive",
          function(conn, payload)
          listaAp =
            "SSID: <select name='ssid'>"
          print("Request Configuration")
          for i = 0, tablelength(ap), 1 do
              if ap[i] ~= nil then
                  listaAp = listaAp ..
                  "<option value='"
                  .. ap[i] .. "'>"
                  .. ap[i] .. "</option>"
              end
          end

          listaAp =
            listaAp .. "</select><br/>"
```

```
        local _GET
          = getParamsUrl(payload)

        if (_GET ~= nil) then
          if (_GET.ssid ~= nil
          and _GET.password ~= nil) then
            print("Connecting to wifi
              creating configuration...")
            print("SSID: "
              .. _GET.ssid)
            print("Password: "
              .. _GET.password)

            print("log1")
            for i = 0,
              tablelength(ap), 1 do
                if ap[i] ~= nil then
                if (string.find(
                ap[i], _GET.ssid)) then
                  user = ap[i]
                end
                end
            end

            password = _GET.password

            wifi.sta
              .config(user, password)
            wifi.sta.connect()

            if file
              .open(
                "config.lc", "w") then
                file.writeline(user ..
                    " " .. password)
                file.close()
            end

            node.restart()
            node.chipid()
            end
          end

          conn:send([[
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="utf8"/>
<title>NodeMcu Configuration</title>
</head>
<body>
<form action="#">
]] .. listaAp .. [[
<p>Password: </p><input type="password"
      name="password"/></p><br/>
<p><input type="submit"
      value="Conectar"/><br/>
</form>
  </body>
  </html>
              ]])
          end
        )
        conn:on(
            "sent",
            function(conn)
              conn:close()
              collectgarbage();
            end
        )
      end
    )
  end
```

APÊNDICE B
UTILITÁRIOS: UTIL.LUA

```
function split(str, pat)
    local t = {}
    local fpat = "(.-)" .. pat
    local last_end = 1
    local s, e,
      cap = str:find(fpat, 1)
    while s do
        if s ~= 1
          or cap ~= "" then
            table.insert(t, cap)
        end
        last_end = e + 1
        s, e, cap =
          str:find(fpat, last_end)
    end
    if last_end <= #str then
        cap = str:sub(last_end)
        table.insert(t, cap)
    end
    return t
end

local unescape = function(s)
    s = string.gsub(s, "+", " ")
    s = string.gsub(
        s,
        "%%(%x%x)",
        function(h)
          return string
            .char(tonumber(h, 16))
        end
    )
    return s
end

function tablelength(T)
    local count = 0
    for _ in pairs(T) do
```

```
        count = count + 1
    end
    return count
end

function getParamsUrl(request)
    local buf = ""
    local _, _, method,
      path, vars = string.find(request,
        "([A-Z]+) (.+)?(.+) HTTP")
    if (method == nil) then
        _, _, method, path =
          string.find(
            request, "([A-Z]+) (.+) HTTP")
    end
    local _GET = {}
    if (vars ~= nil) then
        for k, v in string
          .gmatch(vars, "(%w+)=(%w+)&*") do
            _GET[k] = unescape(v)
        end
    end
    return _GET
end
```

ACKNOWLEDGMENT

The authors would like to thank...

REFERÊNCIAS

[1] Dave Evans. *T*he 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. *M*icro-serviços, vol 1, pp. 1-3, 2017
[3] John Esposito. *T*he 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. *D*ocker 2017 [Online] Disponível: https://www.docker.com. [Acesso: 11-Jun-2017]
[6] Peter Andersson. *S*PIFFS (SPI Flash File System) 2013 [Online] Disponível: https://github.com/pellepl/spiffs. [Acesso: 11-Jun-2017]
[7] ESPlorer. *E*SPlorer Integrated Development Environment (IDE) for ESP8266 developers 2013 [Online] Disponível: https://github.com/4refr0nt/ESPlorer. [Acesso: 11-Jun-2017]
[8] Felipe Demartini. *W*EP, WPA, WPA2: o que as siglas significam para o seu WiFi? 2013 [Online] Disponível: https://www.tecmundo.com.br/wi-fi/42024-wep-wpa-wpa2-o-que-as-siglas-significam-para-o-seu-wifi-.htm. [Acesso: 03-Jun-2017]
[9] Mundo Max. *Q*ual 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-roteador-wireless-e-um-access-point/. [Acesso: 03-Jun-2017]
[10] Irving Souza Lima. *N*odeMCU (ESP8266) o módulo que desbanca o Arduino e facilitará a Internet das Coisas. 2016 [Online] Disponível http://irving.com.br/esp8266/nodemcu-esp8266-o-modulo-que-desbanca-o-arduino-e-facilitara-a-internet-das-coisas/ [Acesso: 05-Jun-2017]
[11] Mark Weiser. *T*he Computer for the 21st Century, vol 1, pp. 1-2, 1991
[12] John B. Kennedy. *W*HEN 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. *I*ntelligent agents: theory and practice, vol 1, pp. 1-3, 1995
[14] Danilo Sato. *C*anaryRelease. 2014 [Online] Disponível: https://martinfowler.com/bliki/CanaryRelease.html. [Acesso: 10-Abr-2017]
[15] NodeMCU. *N*odeMCU 2.0.0. [Online] Disponível: https://github.com/nodemcu/nodemcu-firmware. [Acesso: 10-Dez-2017]
[16] ELua. *W*hat is eLua. [Online] Disponível: http://www.eluaproject.net/overview. [Acesso: 19-Nov-2017]
[17] Systems. *E*xpressif 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. *A*uthors. [Online] Disponível: http://www.lua.org/authors.html. [Acesso: 5-Fev-2017]
[20] Roy Thomas Fielding. *R*epresentational State Transfer (REST. [Online] Disponível: http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm. [Acesso: 29-Mai-2017]
[21] Cate Lawrence. *I*nternet of Things Applications, Protocols, and Best Practices, vol 4, pp. 1-10, 2017
[22] Filipeflop. *C*ontrole 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. *R*edes – 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

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.