REVISTA DE TECNOLOGIA APLICADA (RTA)
v.10, n.3, Set-Dez 2021, p. 33-47.
ISSN: 2237-3713

http://dx.doi.org/10.48005/2237-3713rta2021v10n3p3347

Macoffee: sistema de monitoramento IoT para dispositivos over-the-air*

Macoffee: IoT monitoring system for over-the-air devices

João Paulo Lemos Escola
Instituto Federal de São Paulo - IFSP
jpescola@ifsp.edu.br

Uender Barbosa de Souza
Instituto Federal de Goiás - IFG
uender.souza@ifg.edu.br

Rodrigo Capobianco Guido
Universidade Estadual Paulista - Unesp
guido@ieee.org

Ivan Nunes da Silva
Universidade de São Paulo - USP
insilva@sc.usp.br

Regiane Denise Solgon
Centro Universitário do Norte Paulista - Unorp
regiane.solgon@gmail.com

Jovander da Silva Freitas
Instituto Federal de São Paulo - IFSP
jovander@ifsp.edu.br

Resumo
O recurso *Over-The-Air* (OTA) é uma solução para gerenciar as atualizações de software em dispositivos IoT instalados em locais de difícil acesso, como caldeiras, piscinas e torres. Essa facilidade pode gerar um problema: caso o dispositivo esteja programado para executar tarefas em segundo plano ou hibernar alternadamente, como saber em que momentos do dia ele fica ativo? O presente trabalho apresenta uma solução para os desenvolvedores que utilizam OTA em dispositivos IoT, permitindo-lhes monitorar a atividade do dispositivo em tempo real, além de poder ser empregado como ferramenta para monitoramento manual, facilitando a verificação de estado em locais remotos e de difícil acesso.

Palavras-chave: ESP8266, IoT, Monitoramento, OTA

Abstract
The Over-The-Air (OTA) feature is a solution for managing software updates on IoT devices installed in locations that are difficult to access, such as boilers, pools and towers. This facility can create a problem: if the device is programmed to perform tasks in background or hibernate alternately, how do you know at what times of the day it is active? This work presents a solution for developers who use OTA on IoT devices, allowing them to monitor device activity in real

* Received 14 September 2021; accepted in 31 March 2022; published online 13 April 2022.

time, in addition to being able to be used as a tool for manual monitoring, facilitating status verification in remote and difficult locations access.

Keywords: ESP8266, IoT, Monitoring, OTA

1. Introdução

A Internet das Coisas (*Internet of Things* - IoT) é um paradigma que existe desde os anos 2000 com o advento da internet e vem se popularizando atualmente com a redução do custo de fabricação dos microcontroladores, difusão das conexões de banda larga e seu substancial aumento de velocidade aliados à sua abrangência territorial e redução do preço ao consumidor (OLIVEIRA, 2017).

Essa redução dos preços unida à popularização da tecnologia tem impulsionado aplicações cada vez mais ousadas, como Smart Farm (ESCOLA et al., 2020a; GUPTA et al., 2020), Smart Cities (CUNHA et al., 2016; SHAFIQUE et al., 2020), Smart Houses (TESLYUK et al., 2021), Smart Homes (STOJKOSKA; TRIVODALIEV, 2017) e inúmeras outras. Dependendo do tipo de solução, os dispositivos são acondicionados em locais de difícil acesso, o que pode dificultar eventuais manutenções de código como correções e atualizações.

A inclusão de um recurso para atualização remota OTA (HE et al., 2019) pode facilitar a vida do administrador do sistema, pois não será necessário se deslocar ao ponto de instalação do dispositivo para realizar sua manutenção, caso esta esteja restrita a mudanças ou adequações de código-fonte. Além disso, para fazer uma atualização manual pode ser necessário realizar outras tarefas trabalhosas como remover o dispositivo de um local alto, como uma torre, ou de perigo iminente, como uma caldeira. Essas tarefas, além do risco explícito, podem demandar tempo e dinheiro.

Com a tecnologia OTA é possível realizar a devida manutenção de código, como correções e atualizações sem a necessidade de conexão física ao microcontrolador. Esse recurso funciona, no microcontrolador, como uma rotina que busca por um determinado ponto de acesso previamente configurado pelo desenvolvedor. Por meio dessa conexão sem fio, é possível submeter uma nova compilação de código e atualizar ou corrigir o sistema de forma facilitada (SHEN et al., 2018).

O monitoramento da atividade do dispositivo de forma remota é importante pois permite ao administrador do sistema conhecer o chamado *uptime* do equipamento. Além disso, para sistemas que geram registro em nuvem, a inexistência de atividade *online* pode estar relacionada com algum problema de conexão, mesmo estando ativo e consumindo energia.

Neste trabalho é proposto um sistema de monitoramento para dispositivos IoT que utilizam a tecnologia OTA, permitindo ao administrador conhecer o estado atual de um equipamento ou mesmo os horários do dia em que os dispositivos estiveram ativos.

O presente artigo está organizado da seguinte forma: na próxima seção apresenta-se uma breve revisão da literatura, em seguida a metodologia utilizada para construção do protótipo do sistema proposto. A seção seguinte refere-se aos resultados obtidos no presente trabalho e, em seguida, as conclusões e trabalhos futuros.

2. Revisão da Literatura

2.1 Internet das Coisas (IoT)

De acordo com Atzori et al. (2010) a internet das coisas (IoT) é um paradigma em ascensão no cenário da computação e das telecomunicações. Além disso, graças à evolução da indústria dos semicondutores, os componentes necessários para a construção de um computador agora cabem em um chip de tamanho reduzido e baixo custo.

Com isso, as tecnologias disponíveis, como os microcontroladores, módulos e componentes eletrônicos possibilitam o desenvolvimento de protótipos físicos configurados por meio de uma linguagem de programação com funções inteligentes como coleta de dados, interligação, monitoramento e ativação/desativação de equipamentos externos (OLIVEIRA, 2017), graças também aos módulos adicionais disponíveis como temperatura, umidade, chuva, luminosidade, relê (ativação/desativação de equipamentos), som, música, *display*, led, sensor de presença, sensor de distância, umidade do solo, fogo, comunicação rádio, RFID, infravermelho etc.

O tamanho reduzido dos componentes, seu poder de processamento, baixo custo e consumo reduzido de energia permitem que sistemas que antes dispendiam equipamentos maiores como servidores Web, em muitos casos, sejam substituídos por protótipos funcionais. Além disso, novas soluções vão gradualmente sendo criadas, com a possibilidade de implantação em espaços restritos e reduzidos.

As diversas formas de comunicação como rede sem fio (wi-fi), *bluetooth*, rádio, dentre outras possíveis com a inclusão de módulos externos, permitem a criação de sistemas distribuídos com trocas de dados e conectividade antes não imaginados.

De acordo com Santos et al. (2016) IoT é uma extensão da própria Internet, uma tecnologia que proporciona conexão à rede mundial de computadores a objetos do cotidiano com capacidade computacional, permitindo inicialmente controlar remotamente tais componentes e em um segundo momento, que os próprios forneçam serviços aos usuários. Ainda segundo Santos et al. (2016), o conceito de "Rede de Computadores" está evoluindo, saindo de uma interconexão de computadores para uma rede de diversos equipamentos heterogêneos como consoles de jogos, *smartphones* e TVs.

2.2 ESP8266

Para desenvolver soluções IoT, primeiramente precisamos decidir qual será o microcontrolador responsável por gerenciar o processo, ao qual será configurado via código-fonte por meio de uma linguagem de programação. Essa escolha passa pela análise da necessidade de processamento do projeto, quantidade de pinos GPIO e pela quantidade de memória RAM necessária para viabilizá-lo. A disponibilidade de bibliotecas e documentação também é fator importante na escolha do microcontrolador.

O microcontrolador mais comum e barato disponível atualmente no mercado é o Arduino (OLIVEIRA, 2017), com ele é possível produzir diversos tipos de soluções de automação. Existem diversos modelos no mercado e cada um provê uma quantidade de portas lógicas analógicas e digitais que permitem ampla gama de soluções de automação. Com a inclusão de módulos adicionais é possível extender os recursos do microcontrolador e permitir soluções adicionais como ligar/desligar lâmpadas e medir temperatura/umidade.

O ESP8266 (SCHWARTZ, 2016; OLIVEIRA, 2017) é um microcontrolador de baixo custo com CPU de 80MHz e memória RAM de 32KB, que provê as mesmas funcionalidades do Arduino com a vantagem da conexão sem fio embutida, permitindo maior facilidade ao desenvolvedor na produção de soluções que utilizem internet ou comunicação entre múltiplos dispositivos.

Para programar um ESP8266 é necessário instalar em um microcomputador o conjunto de softwares de desenvolvimento do Arduino, que provê diversas bibliotecas de código aberto para várias funcionalidades, como DNS, cliente/servidor HTTP e integração com módulos externos. Em seguida, é possível criar um novo projeto, compilar e gravar no microcontrolador por meio de uma interface USB.

No presente trabalho o ESP8266 foi utilizado como microcontrolador devido ao recurso de rede sem fio embutido, com bibliotecas personalizadas. Além disso, tem configuração de *clock* e memória ligeiramente superior ao Arduino, se mostrando uma boa opção para projetos mais robustos.

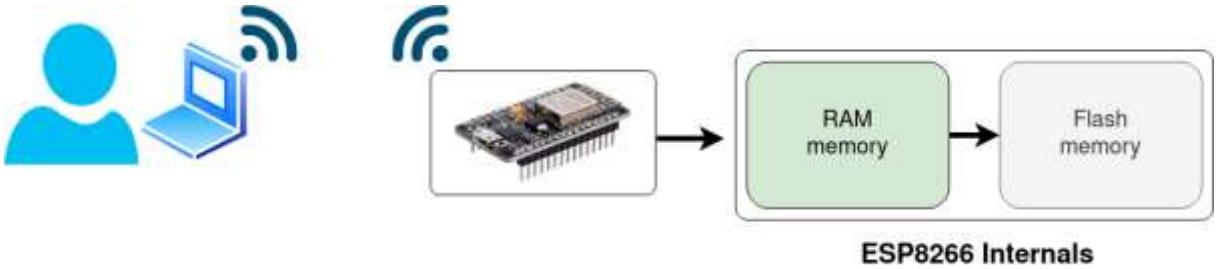
2.3 Over-The-Air (OTA)

Para desenvolver soluções utilizando dispositivos IoT, como Arduino ou ESP8266, é necessário estar conectado fisicamente ao dispositivo. Durante o processo de programação, o código desenvolvido é compilado no microcomputador e transmitido via cabo para o dispositivo IoT. Dessa forma, caso se trate de uma atualização ou correção e o dispositivo já esteja implantado na localidade final, é necessário que o desenvolvedor se desloque, levando consigo o equipamento e o cabeamento para refazer o processo de atualização, o que pode resultar em prejuízos e transtornos.

A tecnologia OTA permite que um dispositivo IoT seja atualizado pela rede sem fio, ou seja, sem a necessidade de conexão manual de um cabo em um computador (SHEN et al., 2018). Para isso é necessário instalar a biblioteca OTA antes da implantação, configurando um nome de ponto de acesso (*Access Point* - AP), usuário e senha. Dessa forma, o dispositivo vai se conectar ao AP sempre que esse estiver disponível, mesmo que nenhuma compilação seja submetida a ele, ficando em estado de espera para novas compilações, mas executando sua tarefa principal sem interferências.

O processo de *upload* da nova compilação com OTA acontece conforme ilustrado na Figura 1. Uma vez que o dispositivo está conectado ao AP e, consequentemente aguardando requisições, o desenvolvedor pode submeter o novo código ao dispositivo, bastando para isso um laptop ou outro aparato conectado à mesma rede sem fio. A nova compilação é enviada ao dispositivo e armazenada em uma memória temporária. Ao final do *upload*, a nova compilação é substituída na memória principal do dispositivo e este é reiniciado com a nova programação e as devidas correções submetidas pelo desenvolvedor.

Figura 1 - Processo de substituição da compilação usando OTA



Fonte: Os autores (2021)

O alcance da rede sem fio vai ditar a necessidade de deslocamento do desenvolvedor no processo, assim, se o dispositivo está conectado na mesma rede empresarial ou residencial, o desenvolvedor poderá realizar o processo sem a necessidade de locomoção, entretanto, nos casos de dispositivos implantados em localidades afastadas o OTA elimina, ao menos, a

necessidade de contato com o dispositivo, que pode estar em local de difícil acesso, como uma torre alta ou uma caixa hermética.

2.4 Web Services

Os serviços de internet ou *Web Services* (WS) (RICHARDSON; RUBY, 2008) são soluções que permitem integração entre sistemas, que podem ser de diferentes tipos e tecnologias (MARTINS, 2006; PERACCI; BESSA, 2015; VIGO et al., 2015), recebendo e respondendo requisições de rede TCP (TANENBAUM, 2003), como HTTP e HTTPS. Exemplos de soluções possíveis de serem providas por um Web Service podem ser uma URL de coleta (caso A) e outra de relatório (caso B):

- No caso A, ao receber requisições, os dados do pacote são armazenados em discos ou em um banco de dados e uma resposta positiva é fornecida em caso de sucesso ou uma resposta negativa em caso de uma possível falha na leitura ou armazenamento dos dados;
- No caso B, ao receber uma requisição, o WS realiza a leitura dos dados em disco e retorna em resposta os pacotes necessários com os dados solicitados na requisição.

Um WS é um software executado em um servidor de redes remoto, normalmente disponível na Internet. Sua função é aguardar requisições e respondê-las, retornando dados quando apropriado. No presente trabalho essa solução foi utilizada para permitir o armazenamento dos dados coletados pelo protótipo e serem disponibilizados em tempo real para o administrador por meio de uma interface Web, conforme apresentado adiante.

2.5 Trabalhos Correlatos

No trabalho de Cantanhede e Silva (2014) é feito o monitoramento do ambiente hospitalar por meio de módulos sensores de temperatura DHT11 e DS18B20, além de um sensor de CO2.

Na pesquisa de Silva et al. (2016) também são utilizados sensores de temperatura/umidade e dispositivos IoT como Raspberry Pi e ZigBee, além de um WS para persistência dos dados.

No trabalho de Dornelas e Campello (2017), um exemplo de implementação prática de IoT em *Smart Home*, são utilizados módulos de medição de fluxo de água conectados a um Arduino para monitorar consumo de água residencial.

No artigo de Hossain et al. (2019) é apresentado um sistema para monitoramento de sub-estações de energia utilizando ESP8266 e sensores como ultrassônico e de nível de óleo, utilizando WS para armazenamento de dados, sob protocolo MQTT.

O improvimento de segurança no processo de atualização de código-fonte IoT em OTA é foco do trabalho de He et al. (2019), onde a tecnologia *blockchain* é empregada incorporando criptografia no referido processo, permitindo monitorar suas atualizações. Os autores utilizaram um conjunto de microcontroladores ESP8266 e efetuaram testes de escalabilidade e vulnerabilidade de redes de computadores.

Na publicação de Košt'ál et al. (2019) temos um sistema de monitoramento para dispositivos IoT utilizando a tecnologia *blockchain*. Seu foco está no improvimento da segurança do processo de atualização dos dispositivos ao mesmo tempo que mantém o controle de suas atualizações.

O artigo de Vázquez et al. (2020) apresenta uma ferramenta de monitoramento de integridade de *datacenter* utilizando IoT. Nesse caso, um ESP8266 com sensores de

temperatura/humidade, além de um sensor de presença, integrado a um Raspberry Pi, envia dados a um servidor Zabbix a fim de aprimorar seu monitoramento padrão.

O monitoramento do ambiente de *datacenter* também é foco do trabalho de Ramphela et al. (2020), onde um dispositivo IoT é desenvolvido com múltiplos sensores, como temperatura, fumaça e variação de tensão, para monitorar a estabilidade do ambiente de acondicionamento dos servidores e demais equipamentos de redes de computadores.

Após a revisão da literatura buscando soluções para monitoramento IoT, verificou-se que, na maioria dos trabalhos, onde se encontra o termo "monitoramento", este se refere à aferição de sensores acoplados ao dispositivo IoT e não à averiguação de estado de dispositivos que realizam tais tarefas, como é o objetivo do presente trabalho. Entretanto, os poucos trabalhos encontrados onde se busca aplicar o monitoramento de dispositivos IoT confirmam a importância do presente estudo para a área de pesquisa.

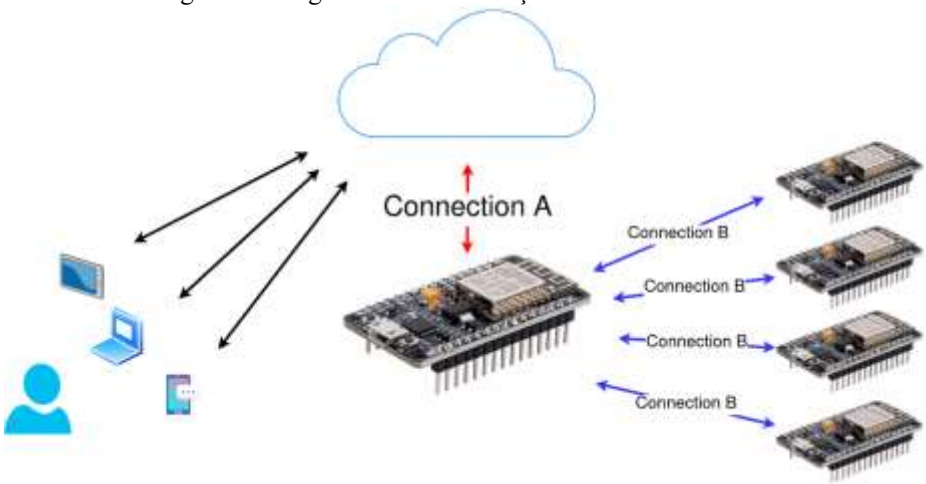
Por serem considerados *hosts* nas redes em que estão conectados, as soluções de monitoramento de redes de computadores como Olups (2010) e Olups et al. (2017) podem se mostrar adaptáveis para monitoramento de dispositivos IoT, apesar se não contemplarem todos os objetivos do protótipo proposto, conforme veremos adiante.

3. Metodologia

3.1 Protótipo Proposto

O sistema de monitoramento proposto no presente trabalho (Figura 2) é composto por microcontrolador ESP8266 alimentado por um cabo micro-USB (Modo A) ou uma bateria de 6V 2.8Ah (Modo B) e um Web Service desenvolvido em linguagem Web PHP, detalhados a seguir:

Figura 2 - Diagrama de comunicação com o Web Service



Fonte: Os autores (2021)

- Modo A: Dispositivo implantado em ambiente local (residência ou empresa), dentro do alcance da rede sem fio com acesso à Internet, com alimentação pela rede elétrica, permitindo captura (Connection B) e atualização dos registros na nuvem em tempo real (Connection A);

- Modo B: Dispositivo manuseado por usuário habilitado ou instalado em veículo (trator, carro elétrico, drone etc), fora do alcance de redes sem fio com acesso à Internet. O processo de captura é realizado em uma primeira etapa (Connection B) e os dados são enviados à nuvem em uma etapa posterior, quando o acesso à Internet estiver disponível (Connection A).

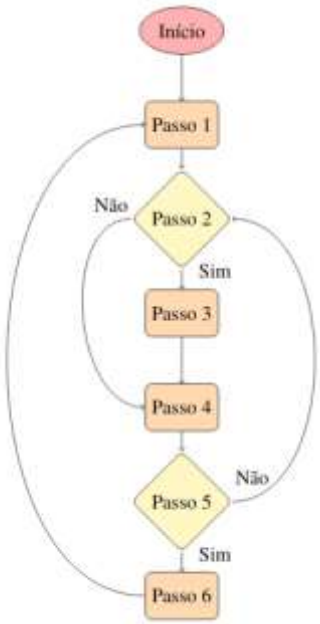
Em ambos os modos, ao final ou durante o processo, o administrador pode acompanhar os resultados por uma página Web, podendo haver atraso na atualização do relatório quando houver dificuldade de acesso à internet (Modo B).

3.2 Algoritmo de Monitoramento

O microcontrolador ESP8266 atua como um ponto de acesso sem fio e seu algoritmo é responsável por monitorar e registrar na nuvem as requisições de conexão, conforme ilustrado na Figura 3 e descrito a seguir:

- Passo 1: O sistema entra em modo AP aguardando requisições;
- Passo 2: Verifica se algum cliente se conectou;
- Passo 3: Para cada cliente conectado, cria um novo objeto *Reg*, composto dos atributos *mac* (Endereço MAC) e *time* (horário atual), armazenando em um vetor de objetos *logs*;
- Passo 4: Aguarda 60 segundos e atualiza o número de iterações;
- Passo 5: Verifica se o limiar de iterações *max_loops* foi atingido;
- Passo 6: Executa a rotina *cloudRegister()*, responsável por desconectar do modo AP e entrar no modo *Client*, conectar à Internet e submeter o vetor *logs* ao WS.

Figura 3 - Fluxograma do algoritmo embarcado no microcontrolador



Fonte: Os autores (2021)

É importante salientar que, no passo 6, na rotina *cloudRegister()*, para cada item do vetor *logs*, faz uso da sub-rotina *encrypt()* responsável por gerar uma URL codificada e criptografada, realizando uma requisição ao WS. Ao final do conjunto de requisições, a rotina

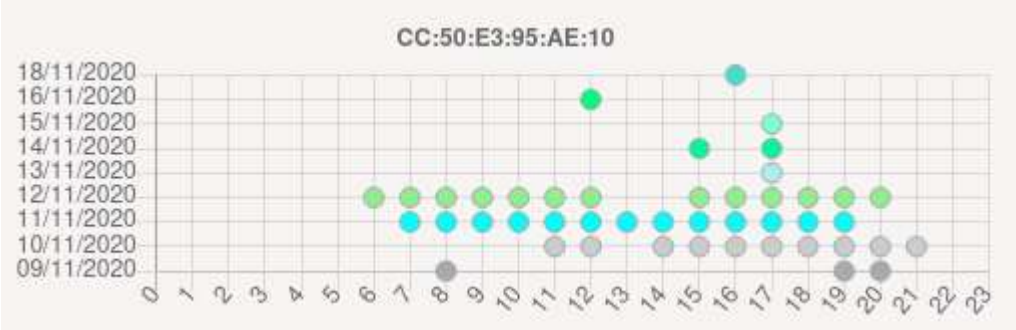
limpa o vetor de objetos *logs*. Caso não seja possível conectar à Internet, esse vetor permanece na memória RAM do microcontrolador até que seja possível realizar o procedimento.

Para que o dia/horário do *uptime* seja corretamente registrado posteriormente, o WS executa o cálculo do tempo (em minutos) decorrido entre o registro e o envio dos dados à nuvem. Com isso, o cálculo do horário real do registro é baseado na hora certa mantida no servidor por meio do protocolo NTP, não sendo necessário sincronizar os horários entre as plataformas, o que poderia causar falha de integridade.

3.3 Web Service

O WS desenvolvido é responsável por receber, via requisição HTTPS, e armazenar em disco os registros de monitoramento do dispositivo. Além disso, provê uma interface amigável de monitoramento ao usuário (Figura 4).

Figura 4 - Exemplo de gráfico de monitoramento de dispositivo no relatório Web



Fonte: Os autores (2021)

Desenvolvido em linguagem de programação PHP (CONVERSE et al., 2004), utilizando tecnologias como Bootstrap (COCHRAN, 2012; SPURLOCK, 2013) e ChartJS (DA ROCHA, 2019), sob o paradigma de orientação à objetos (COX, 1986; AZIS; KOM, 2005), o algoritmo inclui uma rotina de decriptação *decrypt()*, responsável por decodificar e descriptografar os dados submetidos, utilizando o método AES-128-CBC (KROMANN, 2018), uma vez que os dados são submetidos encriptados pelo protótipo físico. O WS também inclui a rotina *save()* que tem função de armazenamento do registro em disco.

A interface de monitoramento provida pelo WS permite visualizar os dias e horários que o dispositivo esteve ativo (*uptime*). Conforme observa-se na Figura 4, o eixo *x* representa as horas do dia e o eixo *y* os dias do mês. Cada círculo no gráfico representa a hora do dia em que o dispositivo esteve ativo, assim, uma rotina verifica se o dispositivo esteve ativo em ao menos um momento dentro de determinada hora do dia, em caso positivo, o círculo é incluído no gráfico e a próxima hora do dia é analisada. Esse gráfico permite, portanto, conhecer o período do dia em que o dispositivo esteve ativo. Esse tipo de informação é importante tanto para dispositivos alimentados por baterias e energia solar (DEZOTTI et al., 2019; ESCOLA et al., 2020a; ESCOLA et al., 2020b) quanto para quaisquer sistemas de nível crítico ou tempo real.

3.3.1 Algoritmo do Web Service

O algoritmo do WS é composto de duas rotinas principais:

- Rotina 1: Responsável por recebimento de requisições de registro. Ao receber uma requisição HTTPS/GET com o parâmetro *s*, a rotina decodifica e descriptografa a *String s*, obtendo os valores de *mac* e *elapsed_time*, em seguida calcula a data e hora do registro subtraindo da data/hora atual *time()* o valor de *elapsed_time*, conforme a Equação 1, salvaguardando os dados em disco;
- Rotina 2: Responsável por gerar o relatório visual para o administrador do sistema. Ao receber uma requisição HTTPS/GET sem parâmetros, a rotina realiza a leitura dos dados em disco, filtrando-os por MAC, dia e hora. Em seguida gera um gráfico para cada MAC armazenado, apresentando o resultado (Figura 4).

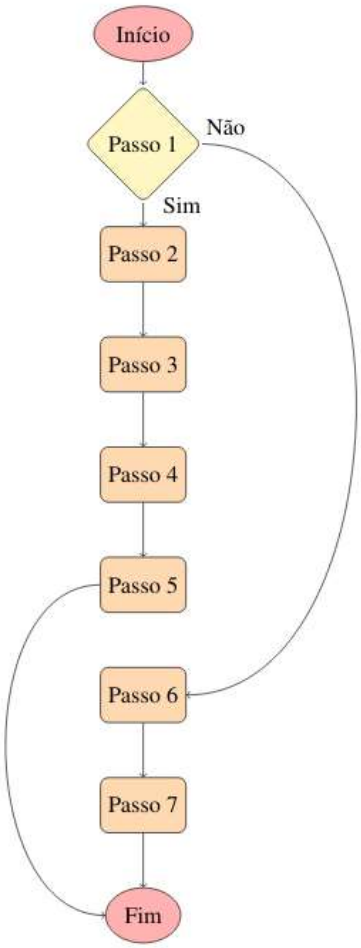
$$datetime = time() - elapsed\_time \quad (1)$$

Conforme mencionado, a Figura 2 ilustra o funcionamento do WS no escopo do presente trabalho. Os dados coletados pelo dispositivo são armazenados em sua memória e transmitidos para o WS sempre que uma conexão à internet estiver disponível, para isso o dispositivo alterna sistematicamente a conexão sem fio entre as redes A (internet) e B (rede OTA).

O algoritmo ilustrado no fluxograma da Figura 5 apresenta os passos executados pelo WS de acordo com o parâmetro recebido. A rotina 1, composta dos passos 2 a 5, e a rotina 2, composta dos passos 6 e 7, executam as tarefas necessárias para o funcionamento do sistema proposto, sendo que, passo 1 é responsável por direcionar o fluxo de processamento entre as rotinas, da seguinte forma:

- Passo 1: Verifica a existência do parâmetro *s* na requisição;
- Passo 2: Descriptografa *s*;
- Passo 3: Decodifica *s*;
- Passo 4: Extrai o endereço *mac* e calcula *time* de acordo com a Equação 1;
- Passo 5: Armazena os dados em disco;
- Passo 6: Recupera os dados em disco;
- Passo 7: Apresenta a página Web com os resultados.

Figura 5 - Fluxograma do algoritmo WS



Fonte: Os autores (2021)

4. Resultados

Os testes propostos foram divididos em duas etapas, sendo a Etapa 1, constituída de testes em laboratório e os testes da Etapa 2 em ambiente de *Smart Farm*, onde os dispositivos se encontram em posições afastadas sem acesso à Internet.

4.1 Etapa 1 (E1)

Os testes dessa etapa foram realizados em laboratório onde o dispositivo monitorado (DM) implementa o recurso OTA e o protótipo monitor (PM) encontra-se no raio de alcance de uma rede sem fios com acesso à Internet.

O teste durou 10 dias, no período de 9/11/2020 a 19/11/2020 e possibilitou comprovar o funcionamento apropriado do sistema, que realizou todas as tarefas esperadas: registro, envio, armazenamento e apresentação dos dados (Figura 4). A acurácia do sistema nessa etapa foi de 100%, considerando que em todo o tempo que DM ficou ligado foi devidamente registrado por PM.

4.2 Etapa 2 (E2)

Os testes em E2 consideram DM em um ambiente afastado, sem acesso à Internet. Apesar disso, com o recurso OTA habilitado, há sondagem pelo ponto de acesso configurado, por isso, ao receber o sinal de PM, deve buscar se conectar, culminando no devido registro em PM.

Segue na Tabela 1 possíveis cenários, comparando seus tempos de resposta de acordo com a forma de deslocamento escolhida. Nessa tabela as variáveis *v1*, *v2* e *v3* correspondem às velocidades do drone, trator e de uma pessoa à pé respectivamente, *n* é o número de DMs. A variável *m* representa o tempo de manuseio, que pode ser o tempo necessário para realizar uma ou mais das tarefas a seguir:

Tabela 1 - Análise comparativa de custo de tempo em E2

Cenário	PM	Deslocamento	Desmontagem	Tempo
A	não	N/A	sim	$n\left(\frac{d}{v}\right)m$
B	sim	Drone	não	$n\left(\frac{d-A}{v1}\right)$
C	sim	Trator	não	$n\left(\frac{d-A}{v2}\right)$
D	sim	A pé	não	$n\left(\frac{d-A}{v3}\right)$

Fonte: Os autores (2021)

- Desmontagem e montagem de DM;
- Subida e descida caso instalado em uma torre ou poste;
- Mergulho ou esvaziamento, caso instalado em um poço ou tanque.

No cenário A, não há utilização do sistema proposto, assim, é necessário que o administrador se desloque até a posição em que DM está instalado para averiguar seu estado. Um led externo poderia ser uma solução que evitaria a necessidade de alcançar o dispositivo, subindo em torres, escadas ou desmontando o suporte, entretanto, o led pode chamar a atenção durante a noite, até durante o dia, causando vandalismo e atraindo curiosos.

No cenário B, um drone é utilizado para monitorar a área. Nesse caso, PM está embarcado no drone e uma distância A (alcance) é suficiente para capturar o estado do dispositivo, economizando bateria do drone, entretanto, o custo inicial para se adquirir um drone é alto atualmente e é necessário treinamento técnico inicial para manuseá-lo. A alimentação pode ser realizada por bateria, considerando a Equação 2 (TIPLER; MOSCA, 2000; YOUNG; FREEDMAN, 2008; HALLIDAY et al., 2017) ou integrada à bateria do drone por meio de um cabo adaptado.

$$autonomia(h) = \frac{capacidade(mAh)}{consumo(mA)} * aproveitamento \quad (2)$$

No cenário C, um trator é empregado para movimentar PM na lavoura. De acordo com Almeida et al. (2010), a velocidade de um trator pode variar entre 4 Km/h e 10 Km/h,

aumentando a agilidade do processo. Uma alternativa viável pode ser aproveitar a atividade corriqueira do motorista, assim, acoplando PM no trator é possível obter um conjunto razoável de dados, dependendo da frequência de passagem do trator. A exemplo do cenário anterior, PM pode ser alimentado por bateria ou cabo.

No cenário D, o trabalhador da lavoura leva consigo o PM para realizar a varredura dos pontos DM. Apesar de ser recomendável manter PM desligado durante o trajeto, ativando-o quando houver proximidade mínima de DM, a Equação 2 também pode ser considerada aqui. A aproximação também se restringe a A, entretanto a frequência de análise diminui, uma vez que a velocidade do trabalhador é a mais baixa dos casos estudados. Além disso, caso o trabalhador não tenha outra atividade a realizar, seu pró-labore será exclusivo para essa atividade, inviabilizando o processo.

Em todos os cenários, a resposta do sistema é dependente da frequência de aplicação do processo, assim, para cobrir todos os horários do dia seria necessário aplicar cada método ao menos uma vez por hora, aumentando o custo e diminuindo a confiança no sistema pois nada garante que no momento da aplicação o dispositivo estava ativo ou que não poderia estar ativo 1 minuto depois, contabilizando-o como inativo naquela hora nesse caso.

5. Conclusões

Este artigo apresentou uma proposta de sistema para monitoramento de dispositivos IoT que utilizam OTA, composto por um microcontrolador e um Web Service. Os resultados mostraram a viabilidade do sistema tanto para monitoramento automatizado quanto para auxílio na conferência manual ou individual de estado de um ou mais dispositivos.

Uma alternativa para o sistema proposto poderia ser a utilização de um software de monitoramento, como o Zabbix (OLUPS, 2010; HORST et al., 2015), entretanto, isso seria possível somente em E1, pois é necessário que o alcance da internet seja possível para comunicação entre ele e DM. Além disso, poderia ser necessário um treinamento prévio já que o software possui diversas funcionalidades, não exclusiva ao monitoramento de dispositivos.

Uma questão positiva para adoção de PM está na segurança, visto que funciona como um *Firewall* para a rede de dispositivos DM, por ser o único da rede a estar conectado à Internet, diferente do que acontece com softwares como Zabbix.

Apesar do presente trabalho focar no monitoramento IoT, o dispositivo proposto pode ser utilizado para monitorar qualquer dispositivo que possua interface wi-fi como computadores, celulares, notebooks, tablets, smart TVs etc. Entretanto, como trata-se de um protótipo com função exclusiva de monitoramento, esses dispositivos não teriam acesso à Internet por meio da interface monitorada, sendo necessário uma segunda interface para isso.

Para os testes em E2, mesmo considerando a confiabilidade inferior nos casos de frequência reduzida de coletas, todos os cenários em que PM é utilizado (cenários B,C e D) têm maior rapidez e facilidade na análise da situação do dispositivo, mostrando-se melhor alternativa em relação a realizar a verificação manualmente (cenário A), mesmo existindo um led ou outra forma externa de analisar o funcionamento do dispositivo, pois no caso de PM esse procedimento pode ser feito à considerável distância.

Não foram encontrados relatos na literatura científica de trabalhos semelhantes ao proposto, colocando este protótipo, denominado Macoffee, como promissor para monitoramento de dispositivos IoT com OTA contribuindo efetivamente para a área de pesquisa.

Para trabalhos futuros desejamos aprimorar a interface Web incluindo filtros que possibilitem ajustes de visualização de acordo com as necessidades do usuário. Principalmente para o caso de E1, seria possível analisar a atividade de DM em determinada hora do dia ou em determinado dia da semana no mês.

Referências

ALMEIDA, R. A. S. de; TAVARES-SILVA, C. A.; SILVA, S. de L. **Desempenho energético de um conjunto trator-semeadora em função do escalonamento de marchas e rotações do motor**. Agrarian, v. 3, n. 7, p. 63–70, 2010.

ATZORI, L.; IERA, A.; MORABITO, G. **The internet of things: A survey. Computer networks**, Elsevier, v. 54, n. 15, p. 2787–2805, 2010.

AZIS, I. M. F.; KOM, M. **Object oriented programming php 5.** Elex Media Komputindo, 2005.

CANTANHEDE, R. F.; SILVA, C. E. da. **Uma proposta de sistema de iot para monitoramento de ambiente hospitalar**. Anais da VII Escola de Computação e suas Aplicações-EPOCA, 2014.

COCHRAN, D. **Twitter bootstrap web development how-to**. Packt Pub., 2012.

CONVERSE, T.; PARK, J.; MORGAN, C. **PHP5 and MySQL bible**. John Wiley & Sons, 2004. v. 147.

COX, B. J. **Object-oriented programming: an evolutionary approach**. CUMINCAD, 1986.

CUNHA, M. A. et al. **Smart cities: transformação digital de cidades**. Programa Gestão Pública e Cidadania, 2016.

DEZOTTI, A. K. et al. **Dispositivo de monitoramento de densidade populacional de insetos a partir de sinais acústicos emitidos**. Brazilian Journal of Animal and Environmental Research, v. 2, n. 5, p. 1781–1785, 2019.

DORNELAS, E.; CAMPELLO, S. **Monitoramento de consumo doméstico de água utilizando uma meta-plataforma de iot**. Revista de Engenharia e Pesquisa Aplicada, v. 2, n. 2, 2017.

ESCOLA, J. et al. **Automated acoustic detection of a cicadid pest in coffee plantations**. Computers and Electronics in Agriculture, Elsevier, v. 169, p. 105215, 2020.

ESCOLA, J. P. L. et al. **A case study of wavelets and svm application in coffee agriculture: Detecting cicadas based on their acoustic and image patterns**. In: SILVA, I. N. da; FLAUZINO, R. A. (Ed.). Application of Expert Systems. Rijeka: IntechOpen, 2020. cap. 1. Disponível em: <https://doi.org/10.5772/intechopen.90156>.

GUPTA, M. et al. **Security and privacy in smart farming: Challenges and opportunities**. IEEE Access, IEEE, v. 8, p. 34564–34584, 2020.

HALLIDAY, D.; RESNICK, R.; WALKER, J. **Halliday Physik**. John Wiley & Sons, 2017.

HE, Xinchi et al. Securing Over-The-Air IoT Firmware Updates using Blockchain. In: **Proceedings of the International Conference on Omni-Layer Intelligent Systems**. 2019. p. 164-171.

HORST, A. S.; PIRES, A. dos S.; DÉO, A. L. B. **De A a Zabbix: Aprenda a monitorar e gerenciar aplicações e equipamentos de redes com o Zabbix**. Novatec Editora, 2015.

HOSSAIN, M. S. et al. **A smart iot based system for monitoring and controlling the sub-station equipment**. Internet of Things, Elsevier, v. 7, p. 100085, 2019.

KOŠT'ÁL, K. et al. **Management and monitoring of iot devices using blockchain**. Sensors, Multidisciplinary Digital Publishing Institute, v. 19, n. 4, p. 856, 2019.

KROMANN, F. M. **Secure php programming**. In: Beginning PHP and MySQL. Springer, 2018. p. 461–478.

MARTINS, V. M. M. **Integração de Sistemas de Informação: Perspectivas, normas e abordagens**. Tese (Doutorado), 2006.

OLIVEIRA, S. de. **Internet das coisas com ESP8266, Arduino e Raspberry PI**. Novatec Editora, 2017.

OLUPS, R. **Zabbix 1.8 network monitoring**. Packt Publishing Ltd, 2010.

OLUPS, R.; VACCHE, A. D.; UYTTERHOEVEN, P. **Zabbix: Enterprise Network Monitoring Made Easy**. Packt Publishing Ltd, 2017.

PERACCI, R. F.; BESSA, G. M. A. **Utilizando web service para integração de sistemas um estudo de caso: Prefeitura de juiz de fora**. Caderno de Estudos em Sistemas de Informação, v. 1, n. 1, 2015.

RAMPHELA, M. K. J. et al. **Internet of things (iot) integrated data center infrastructure monitoring system**. In: 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD). [S.l.: s.n.], 2020. p. 1–6.

RICHARDSON, L.; RUBY, S. **RESTful web services.** "O'Reilly Media, Inc.", 2008.

ROCHA, H. D. Learn **Chart. js: Create interactive visualizations for the web with chart. js** 2. Packt Publishing Ltd, 2019.

SANTOS, B. P. et al. **Internet das coisas: da teoria à prática**. Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos, v. 31, 2016.

SCHWARTZ, M. **Internet of Things with ESP8266**. Packt Publishing Ltd, 2016.

SHAFIQUE, K. et al. **Internet of things (iot) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5g-iot scenarios**. Ieee Access, IEEE, v. 8, p. 23022–23040, 2020.

SHEN, P. et al. **Ota measurement for iot wireless device performance evaluation: Challenges and solutions**. IEEE Internet of Things Journal, IEEE, v. 6, n. 1, p. 1223–1237, 2018.

SILVA, M. da et al. **Implementação da iot para o monitoramento das variáveis meteorológicas num AAL**. In: SBC. Anais do XVI Workshop de Informática Médica. [S.l.], 2016. p. 117–126.

SPURLOCK, J. **Bootstrap: Responsive Web Development**. "O'Reilly Media, Inc.", 2013.Referências 17 STOJKOSKA, B. L. R.;

TRIVODALIEV, K. V. **A review of internet of things for smart home: Challenges and solutions**. Journal of Cleaner Production, Elsevier, v. 140, p. 1454–1464, 2017.

TANENBAUM, A. S. **Redes de computadores**. Pearson education, 2003.

TESLYUK, V. et al. **Optimal artificial neural network type selection method for usage in smart house systems**. Sensors, Multidisciplinary Digital Publishing Institute, v. 21, n. 1, p. 47, 2021.

TIPLER, P. A.; MOSCA, G. **Física–volume 1–mecânica, oscilações e ondas, termodinâmica**. Rio de Janeiro: Livros Técnicos e, 2000.

VÁZQUEZ, A. M.; DAFONTE, C.; GÓMEZ, Á. **Open source monitoring system for it infrastructures incorporating iot-based sensors**. In: Multidisciplinary Digital Publishing Institute Proceedings. [S.l.: s.n.], 2020. v. 54, n. 1, p. 56.

VIGO, L. P. et al. **Web services: Integração e padronização de serviços.** RETEC-Revista de Tecnologias, v. 7, n. 1, 2015.

YOUNG, H. D.; FREEDMAN, R. A. **Física I: mecânica.** Young e Freedman.[Colaborador A. Lewis Ford]. Tradução de Sonia Midori Yamamoto. Revisão técnica de Adir Moysés Luiz, v. 12, 2008.