Revista Tecnologia
e-ISSN: 2318-0730
DOI: 10.5020/23180730.2017.V38.1.39-54

Jardim Inteligente IoT - JIIOT

Smart Garden IoT - SMGIOT

Diego Rios Donato Marinho Marino
diegommarino@gmail.com
Unifor

Danilo Reis Vasconcelos
danilo.reis@fujitec.com.br
Unifor

Suellen Galvão Moraes
suellengalvao_m@hotmail.com
Unifor

Resumo

A internet das coisas, ou como mais conhecida do inglês, *Internet of* Things (IoT*),* é um paradigma que cresce à medida que o mundo se torna mais moderno e conectado. IoT é a técnica de se utilizar tecnologias embarcadas em objetos do dia a dia, como utensílios domésticos e equipamentos eletrônicos em geral, conectando-os à internet e trocando informações com outros objetos, usuários e servidores, com vistas a determinado objetivo. Ao passo em que esse paradigma cresce, crescem também as empresas que disponibilizam plataformas para facilitar a conexão dos objetos (ou coisas) por meio da rede. Essas plataformas podem impactar positivamente na velocidade de desenvolvimento de uma solução focada em IoT. Este trabalho apresenta a arquitetura e prototipagem de um ambiente IoT utilizando a plataforma da Amazon, AWS IoT. Tendo em vista a tendência das pessoas buscarem alimentos mais saudáveis, livres de agrotóxicos, porém não possuírem tempo para cuidar de uma horta em casa, foi proposta a ideia de desenvolver um Jardim Inteligente IoT (JIIOT), um ambiente de irrigação automático de plantas com gerenciamento inteligente de irrigação, que ocorre apenas quando as plantas necessitam de água, promovendo assim uma economia de recursos. Para tornar o ambiente possível, foram utilizadas as plataformas *Raspberry Pi* e NODEMCU, além de monitoramento com as plataformas de leitura e armazenamento de dados *plot.ly, Deewt.io, Freeboard.io* e um sensor de umidade de solo. Ao final do trabalho, foi desenvolvido um ambiente IoT eficiente para a automatização da irrigação de um jardim ou horta.

Palavras-chave: IoT. *Internet of Things*. *Amazon AWS. ThingWorx*. Jardim. Horta. Irrigação.

Abstract

The internet of things, also known as IoT, is a growing paradigm. IoT is the technique of using embedded systems in everyday objects such as fridges, trash cans and many others, and connecting these objects to the internet, so they can exchange information among other objects, users and servers, as well as be synchronized to achieve an objective, besides the fact that they can adapt to an environment. At the same time that this paradigm grows and is put into practice, there is a considerate amount of companies of which are offering platforms to help and make easier to connect these objects (or things) to a network. These platforms can help with the development speed of IoT based solutions. This work presents the architecture and prototype of an IoT environment using the Amazon AWS IoT platform. Given that nowadays most people prefer to avoid foods containing agrochemicals, yet they don't have time to take care of a garden, the idea of developing a Smart IoT Garden was proposed, an autonomous plants irrigation system that irrigates only when the plants really need water, focusing in the resources efficiency. To make the environment possible, both Raspberry Pi and NODEMCU platforms were used, besides the monitoring with the read and storage platforms Plot.ly, Dweet.io, Freeboard.io and a soil moisture sensor. At the end of this work, an efficient autonomous irrigation IoT environment for a garden was developed.

Keywords: IoT. Internet of things. Amazon AWS. ThingWorx. Garden. Irrigation. Certification. Green seal.

1 Introdução

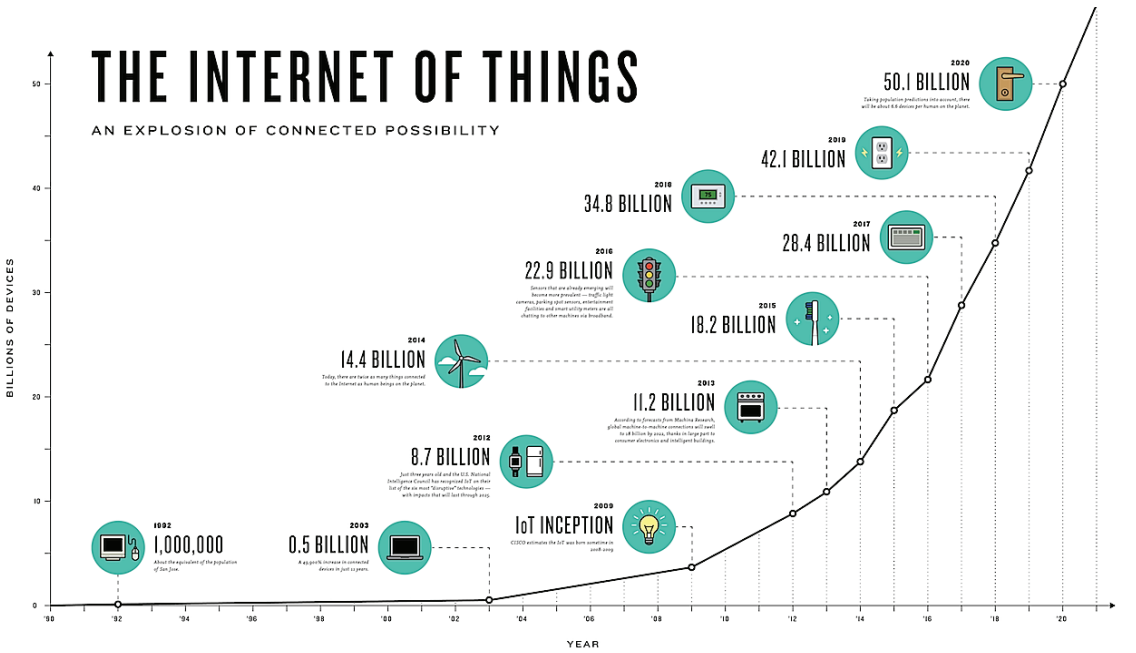
A inovação tecnológica abre caminhos para um mundo moderno e prático. Pathak (2016) relata que as inovações que possibilitaram o surgimento da Internet das Coisas (IoT) foram a computação ubíqua, a adoção de protocolos de internet, a computação em nuvem, a miniaturização de componentes eletrônicos, o avanço das técnicas de análise de dados, o aumento da velocidade de conexão com a internet e o barateamento de componentes eletrônicos.

O termo *Internet of Things* foi citado pela primeira vez por Kevin Ashton, em um artigo no *RFID Journal*, em 1999 (VASCONCELOS, 2016). Internet das coisas é um paradigma que tem como objetivo transformar objetos do cotidiano em máquinas que possam coletar informações, ou se adaptar, de acordo com determinadas condições. Essas informações são enviadas para usuários, ou outros objetos, criando um ecossistema capaz de propiciar diversos benefícios, como eficiência de custo, aumento de produtividade e processamento otimizado (HARRIS, 2016).

O conceito de IoT foi colocado em prática e vem crescendo fortemente, tentando propiciar os benefícios citados. O gráfico apresentado na Fig. 1, confeccionado pela The Connectivist (I-SCOOP, 2016 usando como base os relatórios da CISCO, mostra a evolução da quantidade de dispositivos conectados à internet e a previsão de dispositivos inteligentes que serão conectados à rede até o ano de 2020.

Observando esse cenário como oportunidade, várias empresas estão investindo fortemente em pesquisas para otimizar a ubiquidade de soluções em IoT e também em tecnologias para aumentar a velocidade em que novos produtos podem ser produzidos. Novas plataformas de hardware de sistemas embarcados estão surgindo em grande quantidade todos os anos, cada vez mais focadas em desenvolvimento utilizando IoT, como é o caso do Arduino, que recentemente lançou a plataforma Arduino Yunn, que possui um tamanho bem pequeno e um módulo wi-fi integrado.

Figura 1 – Número de dispositivos conectados.



Fonte: I-SCOOP (2016).

A Figura 1 ilustra a evolução dos equipamentos que estão e/ou serão conectados à IoT. Inicialmente, por volta do ano de 1992, apenas os computadores do tipo *desktop* estavam conectados, evoluindo para os aparelhos do tipo *notebooks* cerca de 10 anos depois e sendo expandido para diversas outros utensílios domésticos e de utilidade pública a partir de 2009. Os demais cenários são previsões em quantidade e diversidade de usos e fins. Estima-se que, até o ano de 2020, cerca de 50 bilhões de equipamentos estejam conectados à IoT.

Outra plataforma de hardware pesquisada e que vem ganhando bastante mercado, principalmente na comunidade de *makers*[1], é o NODEMCU (BRUCE, 2015). É uma plataforma de código aberto que surgiu para disputar o mercado com o Arduino Yunn, pois também possui wi-fi integrado e utiliza um microcontrolador com um significativo poder de processamento.

[1] Cultura moderna que tem em sua base a ideia de que pessoas comuns podem construir, consertar, modificar e fabricar os mais diversos tipos de objetos e projetos com suas próprias mãos.

Pode-se destacar também o *Raspberry Pi*, que vem, desde o modelo inicial de sua plataforma embarcada, aumentando o poder de processamento. A plataforma já está na terceira geração, com processadores *quadcore* e componentes wi-fi integrados, os quais são muito úteis para serem utilizados como *brokers* de comunicação, ou processamento mais intenso de dados.

A crescente quantidade de hardwares com foco em IoT fez com que empresas notassem potencial no desenvolvimento de um *middleware* com uma interface amigável ao usuário, ou desenvolvedor, para facilitar a comunicação entre esses hardwares, além de ser possível, através de um *middleware*, adicionar diversas camadas benéficas a uma aplicação. Entre essas camadas, pode-se citar uma camada de certificados e segurança.

Com essa ideia, várias empresas criaram plataformas com serviços para melhorar a segurança e a conexão com os sistemas embarcados. Além disso, algumas delas adicionaram integração do ambiente de IoT com serviços que a empresa já oferecia. As que possuem um bom destaque são as plataformas AWS IoT, pertencente à Amazon, e ThingWorkx, criada pela PTC *(Parametric Technology Corporation)*.

Juntas, as ferramentas citadas (hardwares *e middlewares*) podem, por exemplo, montar um ambiente IoT com objetos que possuem adaptação autônoma em um certo cenário e que possuem controle e monitoramento. Utilizando uma casa como exemplo de cenário IoT, seria possível aumentar a economia de energia ao verificar por RFID (Identificação por Rádio Frequência) que uma pessoa não se encontra mais em um cômodo e a luz do cômodo se apagar; ao morador sair de casa e receber um alerta de que esqueceu uma porta, ou janela, aberta, sendo possível também que seja fechada automaticamente.

As áreas onde a IoT pode ser inserida são vastas. Alguns exemplos de como a IoT pode atuar em construções é se utilizando sensores para detectar vazamentos de gás, ou água, e assim acionar mecanismos para que a chave de fornecimento seja fechada, evitando desperdícios e acidentes.

O setor de energia conseguiu um grande ganho em eficiência desenvolvendo, por exemplo, painéis solares inteligentes que sempre se posicionam de modo a absorver a maior quantidade de energia possível (SMITH, 2015).

No setor da saúde, foram desenvolvidos comprimidos que enviam dados em tempo real da saúde de um indivíduo para um médico, alertando-o quanto aos efeitos da droga no usuário e identificando, por exemplo, a efetividade do tratamento, podendo a pessoa obter um tratamento mais eficaz em um curto espaço de tempo (PROTEUS, 2016).

Acidentes de trânsito poderiam ser reduzidos se todos os carros conseguissem se comunicar, alertando, por exemplo, se um veículo à frente realizasse uma parada brusca, fazendo o carro de trás frear antes mesmo de o motorista ter notado o carro parado à sua frente. Já há empresas que adicionam sensores em veículos, tornando-os mais inteligentes, podendo o automóvel realizar frenagens de forma automática se identificar um cenário propenso a acidentes (BUDDY, 2016).

O objetivo dessa pesquisa foi elaborar a arquitetura e montar uma rede para um ambiente IoT para irrigação de um jardim de forma autônoma e adequada (quando o solo a ser irrigado atingir uma umidade pré-definida como baixa umidade de solo), bem como implementar ferramentas que possam realizar o monitoramento remoto da umidade do solo do jardim e a elaboração automática de gráficos, para que seja possível realizar a análise temporal da umidade do solo.

Este estudo se justifica pelo interesse crescente de uma parcela da população em consumir alimentos livres de agrotóxicos e com custos menores, tendo em vista a diferença de preço ser acentuada entre os alimentos orgânicos e os produzidos pelo agronegócio (com agrotóxicos). Além disso, muitos desses consumidores não têm tempo para cultivar uma horta em casa devido à necessidade constante e diária de cuidados com a irrigação. Essa tecnologia vem suprir essa demanda, já que possui um sistema de irrigação inteligente, não necessitando da presença diária do cultivador para regar a horta.

Este estudo possui cunho científico acadêmico, trata-se de uma pesquisa aplicada, pautada em estudos bibliográficos, de laboratório e de campo.

O estudo bibliográfico visou o entendimento de como montar um ambiente de IoT utilizando uma plataforma de integração e comunicação pronta. Neste caso, foi escolhida a plataforma da Amazon, AWS IoT. Para isso, foi necessário colher informações em diversas documentos, artigos e fóruns sobre qual hardware, biblioteca e plataforma seriam adequados para a montagem do ambiente.

Elaborou-se uma revisão dos trabalhos desenvolvidos sobre o tema, como pesquisas e estudos de casos de ambientes IoT desenvolvidos para tornar a irrigação de jardins autônoma, assim como áreas similares a deste trabalho, como estudos

de caso que envolvessem a análise de plataformas de software para IoT, tendo como principal foco a plataforma da Amazon, AWS IoT.

As pesquisas foram realizadas durante todo o ano de 2016 e utilizando o Google e as ferramentas[2] disponibilizadas para os alunos da Unifor. Os termos utilizados para realizar a pesquisa, seja em conjunto,seja individualmente, foram: IoT, farm, gardem, ESP8266, NODEMCU, Raspberry Pi, urban farms, MQTT, Arduino, internet das coisas, plataformas IoT, IoT platforms, crescimento IoT, IoT grown.

Esta é uma pesquisa de desenvolvimento, a qual objetiva resolver problemas concretos, com soluções imediatas e aplicações das teorias às necessidades humanas. Também chamada tecnológica, pois resulta em produtos, processos e patentes, gera novas tecnologias e conhecimentos resultantes do processo de pesquisa, sendo muito utilizada pelo curso de Ciência da Computação.

2 Resultados e discussão

De acordo com o relatório de mobilidade da Ericsson (2016), divulgado em 2016, entre 2015 e 2021, o número de dispositivos de IoT conectados crescerá 23% anualmente, somando 16 bilhões do total previsto de 28 bilhões de dispositivos conectados até 2021. Ainda no mesmo relatório, é comentado que o número de dispositivos IoT conectados, ou seja, objetos físicos que possuem uma pilha de IPs que permitem uma comunicação bilateral em uma interface de rede, irão superar a quantidade de smartphones conectados.

Esses dados nos dão uma noção do quão importante será o desenvolvimento nas áreas relacionadas à IoT, pois apontam para um mercado e áreas de pesquisa consideráveis. Mais um fato que aponta para esse crescimento é a crescente quantidade de plataformas para IoT, tanto hardwares quanto middlewares, além de IDEs para programação.

Segundo Smith e Weingarten (1997), tem-se a seguinte definição de middleware: "middleware pode ser visto como um conjunto de serviços e funções reutilizáveis e expansíveis que são comumente utilizadas por muitas aplicações para funcionarem corretamente em um ambiente conectado". O middleware, no caso deste estudo, é a plataforma da Amazon, AWS IoT. A função do middleware no estudo foi a de adicionar uma padronização no desenvolvimento de software, armazenamento de estado de máquina e adicionar camadas de segurança utilizando certificados.

A plataforma AWS é definida pela própria empresa: "A Amazon Web Services oferece um amplo conjunto de produtos de nuvem globais, como computação, armazenamento, bancos de dados, análises, redes, dispositivos móveis, ferramentas de desenvolvedor, ferramentas de gerenciamento, IoT, segurança e aplicações corporativas" (AMAZON, 2016a).

A Amazon (2016b) define sua plataforma para IoT da seguinte maneira:

> O AWS IoT é uma plataforma de nuvem gerenciada que permite que dispositivos interajam facilmente e com segurança com aplicativos de nuvem e outros dispositivos. O AWS IoT pode comportar bilhões de dispositivos e trilhões de mensagens, e pode processar e rotear essas mensagens para endpoints da AWS e para outros dispositivos confiavelmente e com segurança. Com o AWS IoT, as suas aplicações podem acompanhar todos os seus dispositivos, como também comunicar-se com eles, o tempo inteiro, mesmo quando eles não estiverem conectados.

O AWS IoT Device SDK permite que seus dispositivos estabeleçam conexão, sejam autenticados e troquem mensagens com o AWS IoT usando os protocolos MQTT, HTTP, ou WebSockets. O Device SDK é compatível com C, JavaScript e Arduino, e inclui bibliotecas do cliente, guia do desenvolvedor e guia de portabilidade para fabricantes. Ainda mostra o seu poder de integração com os outros produtos pertencentes à empresa:

> O AWS IoT facilita o uso de serviços da AWS, como AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, CloudWatch, AWS CloudTrail e Amazon Elasticsearch Service com integração embutida do Kibana para criar aplicativos da IoT que, além de intervir sobre, reúnem, processam e analisam os dados gerados pelos dispositivos conectados, sem que seja necessário gerenciar nenhuma infraestrutura" (AMAZON, 2016b).

[2] http://web.b.ebscohost.com/ e http://www.periodicos.capes.gov.br/

A integração utilizada para este trabalho foi com o AWS SNS, que é o serviço de envio de mensagens da empresa. As integrações geralmente são realizadas utilizando o mecanismo de regras da plataforma. "O mecanismo de regras avalia as mensagens recebidas publicadas no AWS IoT e as transforma e entrega para outro dispositivo ou um serviço de nuvem, com base nas regras empresariais que você definir" (AMAZON, 2016a).

Para escrever as regras, pode ser utilizado o Device SDK, ou o próprio console da AWS, que pode ser acessado via browser. "Você pode criar regras dentro do console de gerenciamento ou escrever regras usando uma sintaxe do tipo SQL" (AMAZON, 2016a).

O serviço AWS IoT ainda dispõe da funcionalidade de sombras de dispositivos. Uma sombra, ou Thing Shadow, como é chamado, é o serviço de armazenamento do estado mais recente de uma mensagem enviada ao AWS IoT, que pode conter o estado atual de um dispositivo. Toma-se o seguinte exemplo:

```
"state": {
        "desired": {
                "broker": "192.168.0.100",
                "ip": "192.168.0.15",
                "soil_moisture": "good",
                "sensor_value": "520",
                "water_pump": "off",
                "status_led": "off"
        }
}
```

Caso o dispositivo venha a desligar, ou caso falte energia onde o dispositivo está ligado, ele poderá acessar o serviço e recuperar as variáveis que definiam o último estado no qual se encontrava.

O AWS IoT ainda "é compatível com o método de autenticação da AWS (chamado SigV4), e também com o certificado X.509 com base em autenticação" (AMAZON, 2016a). Para comunicação via MQTT, faz-se necessário utilizar os certificados X.509. Os certificados são gerados pela própria plataforma para, em seguida, serem utilizados com os SDKs fornecidos para desenvolvimento.

Uma plataforma de hardware é um conjunto de hardwares compatíveis em que aplicações de software podem ser rodadas. O termo plataforma IoT se refere apenas ao fato de as plataformas definidas nesta sessão estarem ultimamente sendo mais utilizadas para soluções envolvendo IoT, geralmente devido ao pequeno tamanho dos hardwares e por já possuírem módulos de rede sem fio (wi-fi) e/ou bluetooth.

2.1 Raspberry Pi 3 modelo B

É uma plataforma de hardware desenvolvida pela fundação Rasberry Pi. A plataforma possui um grande poder de processamento, podendo ser utilizada para aplicações complexas, que talvez necessitem de um sistema operacional para serem elaboradas. As características dessa plataforma são:

- CPU 1.2GHz 64-bit quad-core ARMv8
- 802.11n *Wireless LAN*
- *Bluetooth* 4.1
- *Bluetooth Low Energy (BLE)*
- 1GB RAM
- 4 portas USB
- 40 pinos GPIO
- Porta *Full HDMI*
- Porta *Ethernet*
- Conector de áudio 3.5mm e composição de vídeo combinados
- Interface de câmera (CSI)

- Interface de *display* (DSI)
- *Slot* para cartão Micro SD
- Core de gráficos 3D VideoCore IV

2.2 NODEMCU V3

Como definido pelo fabricante, "um firmware é um kit de desenvolvimento que ajuda a prototipar seu produto IoT com poucas linhas de script Lua" (NODEMCU, 2016). A IDE, desenvolvida da empresa Arduino, também pode ser utilizada para escrever os *scripts* da plataforma em C++. Essa plataforma utiliza como CPU o módulo ESP8266, que já possui wi-fi integrado. As características da plataforma são:

- CPU 2.4GHz
- 10 GPIO, todas GPIO podem ser PWM
- I2C
- 1-*wire*
- Módulo FCC *CERTIFIED WI-FI*
- Antena PCB
- USB-TTL

2.2 DWEET.IO

Como o próprio slogan da plataforma afirma, é como um *Twitter* para máquinas. Oferece um serviço para publicar dados e também de inscrição para receber dados. Foi feito para conexões *machine-to-machine* (M2M) para a Iot (DWEET. IO, 2016). O serviço gratuito oferecido pelo *Dweet.io* torna todos os dados enviados públicos. Para torná-los privados, é necessário adquirir um plano pago, assim será possível "trancar" os seus dados, além de poder armazenar uma quantidade maior.

2.4 FREEBOARD.IO

Uma plataforma que possui integração com o *Dweet.io* e que permite utilizar os dados armazenados nele para gerar um *dashboard* de gráficos e informações em tempo real. Assim como o *Dweet.io*, é gratuito para dados públicos, mas, para torná-los privados, é necessário adquirir um plano. Como apresentado no site da empresa (DWEET.IO, 2016), as características da plataforma são:

- Fonte de dados flexível
- Desenvolvimento com widgets
- Simplicidade de drag & drop
- Acesso público ou privado
- Compartilhamento instantâneo
- Clonagem de dashboard

2.5 PLOT.LY

Uma plataforma on-line para geração de gráficos. Possui uma grande quantidade de tipos de gráficos que podem ser gerados. Por possuir APIs (*Python, R, Matlab e Javascript*) de integração, facilita bastante o envio de dados para a geração de informações em gráficos e para realizar a análise das informações obtidas. O plano gratuito da ferramenta permite um número limitado (250) de chamadas à API. As características do plano gratuito da plataforma são:

- Suporte do fórum da comunidade
- 1 GB de armazenamento
- 50/dia chamadas à API clientresp e 250/dia chamadas à API v2
- 1000 visualizações de gráficos por dia

- Upload de CSV e Excel
- Criar 1 arquivo privado de cada tipo
- Criar gráficos básicos e 3D
- Exportação nos formatos PNG e JPEG
- Paleta de cores limitada
- Regressões lineares e quadráticas

2.6 MQTT

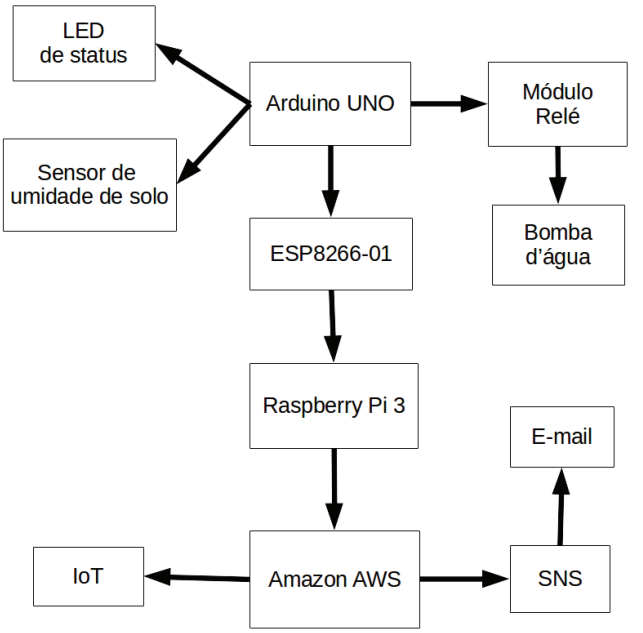
A MQTT.org apresenta a seguinte definição ao protocolo de comunicação:

> MQTT é um protocolo de conectividade machine-to-machine (M2M) /"Internet of Things". Ele foi projetado como um transporte de mensagem por inscrição/subscrição extremamente leve. Ele é útil para conexões com locais remotos onde uma pequena quantidade de códigos é requerida e/ou a largura da banda de rede é essencial (MQTT.ORG, 2016).

O protocolo já é um padrão da OASIS Standard (Advancing Open Standards for the Information Society) desde 2014. A ideia do projeto foi criar um sistema eficiente e com um bom custo benefício.

Foram pesquisadas diversas plataformas de prototipagem eletrônica de hardware, apresentadas nas seções seguintes, e módulos que possuíssem vasta documentação e comunidade, além de preço acessível para facilitar qualquer pesquisador conseguir reproduzir os resultados deste trabalho. Segue abaixo um diagrama de blocos da arquitetura do sistema para melhor entendimento do papel de cada componente (Figura. 2).

Figura 2 – Arquitetura JIIOT.



Fonte: Elaborado pelo autor (2016).

Neste estudo, optou-se por utilizar como gerenciador de sensores e atuadores para a aplicação a plataforma NODEMCU, assim adquirindo os dados de umidade de solo e acionamento de motor de irrigação. Esse componente foi escolhido por conter um módulo wi-fi integrado, facilitando sua realocação no espaço onde seria utilizado e por possuir um pino de leitura analógica, sendo possível realizar a leitura dos valores de um sensor de umidade de solo, além do baixo custo (R$ 27,00 - 45,00) quando comparado às demais opções de mercado.

O Raspberry Pi 3 modelo B foi escolhido devido a ser um dos hardwares listados na documentação da plataforma AWS IoT que suporta integração com o serviço. Outro motivo foi o fato dessa plataforma já possuir um sistema operacional

Debian compilado, facilitando o trabalho de transformar o Raspberry Pi em um broker MQTT e o desenvolvimento de scripts utilizando a linguagem Javascript.

Para adquirir o estado de umidade do solo foi utilizado um sensor de umidade de solo genérico. A medição da umidade é feita utilizando uma resistência que diminui de acordo com a quantidade de água adicionada ao solo, ou seja, fazendo com que o valor de saída do componente diminua, no caso do sensor utilizado, à medida que a umidade aumenta.

Para o acionamento da bomba d'água foi utilizado um módulo de relés e uma mini bomba d'água submersa fabricada pela empresa LITWIN, que possui regulagem de vazão que varia de 30 até 165l/h, o que foi o suficiente para o cenário experimental deste trabalho.

Antes de selecionar a plataforma escolhida (Amazon AWS IoT), foram pesquisadas algumas outras disponíveis no mercado, como ThingWorx e Temboo, mas a possibilidade de integração da plataforma AWS IoT com os outros serviços pertencentes à Amazon (envio de mensagens SMS, banco de dados, processamento e análise de dados) tornou o serviço dela o candidato mais interessante para este trabalho.

Além da integração entre todos os serviços, a AWS da Amazon proporcionou a plataforma IoT gratuita para pequenos projetos, tornando ainda mais simples começar a utilizar a ferramenta. Para a utilização da plataforma ThingWorx de forma gratuita, era necessário um professor de uma universidade realizar o cadastro de estudo da ferramenta e adicionar o aluno no projeto, possuindo ainda algum tempo até o cadastro ser aprovado. Devido ao processo burocrático para utilizar a ferramenta, ela foi deixada de lado.

2.7 Protocolo de comunicação interplataformas – MQTT

Para este trabalho, o protocolo de comunicação escolhido foi um próprio para conectividade em um ambiente de Internet das Coisas, o MQTT (Message Queuing Telemetry Transport). O MQTT é um protocolo com a vantagem de ser extremamente leve e estar se tornando um dos protocolos mais utilizados para comunicação entre plataformas e dispositivos, além de, desde 2014, ter se tornado um padrão da OASIS (Advancing Open Standards for the Information Society). Como exemplo de empresas que já o utilizam, destacam-se a Temboo, Fiorano e Amazon, que foi a plataforma utilizada neste trabalho.

2.8 Bibliotecas e configurações de core

Para realizar toda a comunicação entre as plataformas embarcadas e a plataforma on-line, foram utilizadas algumas bibliotecas para facilitar a integração entre os componentes, principalmente na parte de comunicação MQTT.

2.8.1 NODEMCU

Foi necessário utilizar a biblioteca WiFiEsp para tornar possível o acesso à rede para comunicação e troca de mensagens via wi-fi. A biblioteca PubSubClient foi utilizada para ser possível realizar a comunicação entre as plataformas NODEMCU e Raspberry Pi, utilizando o protocolo MQTT, padronizando a comunicação do projeto.

2.8.2 Raspberry Pi

Para facilitar as configurações e melhor integração com as ferramentas utilizadas no projeto, foi utilizado o sistema operacional Raspbian, que é uma versão personalizada do SO Debian compilada para Raspberry Pi.

Para a comunicação com a plataforma AWS IoT, foi necessário primeiro instalar o interpretador de código Javascript, o Node JS, para, em seguida, serem instaladas as bibliotecas que possibilitam a comunicação com a plataforma AWS IoT, a aws-iot-device-sdk, e a biblioteca para utilizar o protocolo de comunicação MQTT, a mqtt.

Foram utilizadas também as bibliotecas: pg, para armazenar dados localmente no banco de dados PostgreSQL; node-dweetio, para enviar dados para o Dweet.io e monitorar a umidade de solo; plotly, para gerar o gráfico com os dados armazenados durante o tempo de funcionamento do sistema.

2.9 Elementos do jardim

Para a montagem do ambiente do jardim foram utilizadas mudas de cebolinha, tomate cereja e coentro plantadas em copos de plástico e garrafas de leite cortadas, atingindo o volume de 700 ml. As mudas pertencem a u dos autores deste

trabalho e, antes da automatização do jardim, eram irrigadas manualmente com um regador até o ponto considerado ideal para crescerem. Todas foram irrigadas com a mesma quantidade de água e obtiveram um crescimento regular e saudável.

2.9.1 Análise de uma rede de sensores

O primeiro passo para se obter o resultado esperado foi projetar um sistema de irrigação autônomo. Para isso, foram estudados trabalhos anteriores que utilizaram técnicas de automação para irrigação de hortas e jardins, ou controle de ambiente em geral. Dentre os trabalhos pesquisados, o mais próximo ao do ambiente IoT desejado para este experimento foi o dos pesquisadores Rupali B. Mahale e Dr. S. S. Sonavane (2016).

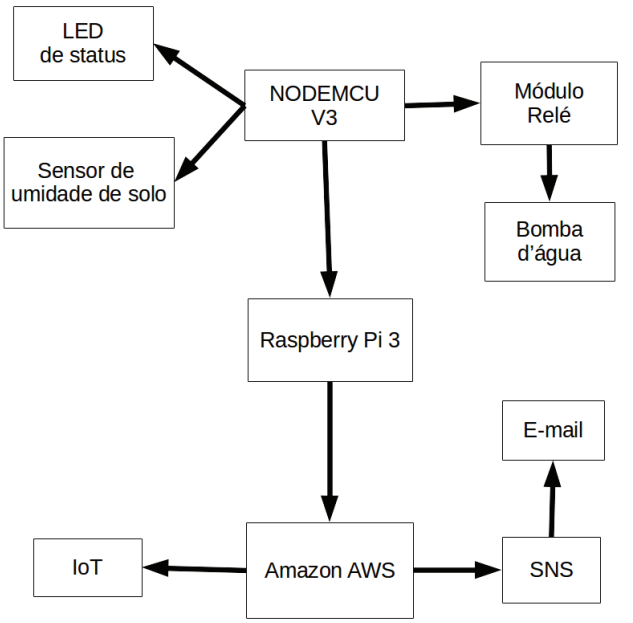
O problema do sistema apresentado pelos pesquisadores é que não há comunicação sem fio entre os cores, o que pode dificultar a escalabilidade do sistema, ou a realocação dos componentes. Outro fator que pode ser observado é que o ambiente criado não utiliza plataformas de middleware para auxiliar o envio de notificações de alerta dos sensores. Segundo os autores, essas notificações serão enviadas utilizando um script codificado em JAVA e exibidas em um aplicativo para smartphones Android, mas nenhum detalhe sobre o script, ou a aplicação, foi apresentado no trabalho.

Para suprir as deficiências de comunicação sem fio do trabalho de Mahale e Sonavane, teve-se a ideia de adicionar conectividade sem fio ao Arduino UNO. Dentre as alternativas pesquisadas, a mais viável para atingir o objetivo seria o módulo WiFi ESP8266-01, pois possui comunicação por rede e preço acessível. Ao integrar esse módulo com o Arduino tornou-se possível utilizar o protocolo MQTT para enviar mensagens ao broker MQTT.

Para broker MQTT e core de aplicação, a mesma plataforma do trabalho estudado foi escolhida, o Raspberry Pi versão 3. Através da rede, o Raspberry Pi recebe mensagens provenientes do Arduino, relatando o status do sensor de umidade de solo, da bomba d'água e do LED de alertas. Através de scripts codificados em javascript, o Raspberry faz comunicação com os serviços da Amazon para enviar alertas por e-mail quando for detectada uma baixa umidade de solo.

O serviço utilizado para os alertas por e-mail foi o serviço AWS SNS (Simple, Scalable Push Notification Service). Para a conexão com os serviços e armazenamento de status do Arduino, foi utilizado o AWS IoT. Alguns problemas na prototipagem foram enfrentados ao tentar implantar a arquitetura proposta com o módulo ESP8266, fazendo com que se optasse por usar um hardware com recurso wi-fi já integrado. O hardware escolhido foi o NODEMCU, que é um hardware de prototipagem que utiliza o microcontrolador do módulo ESP8266 como uma solução mais completa. O novo core se comporta como um Arduino, mas com wi-fi já integrado. Com essa alteração de componente, o Arduino também pôde ser removido, fazendo com que a arquitetura final do projeto fosse montada como mostrado na Figura 3.

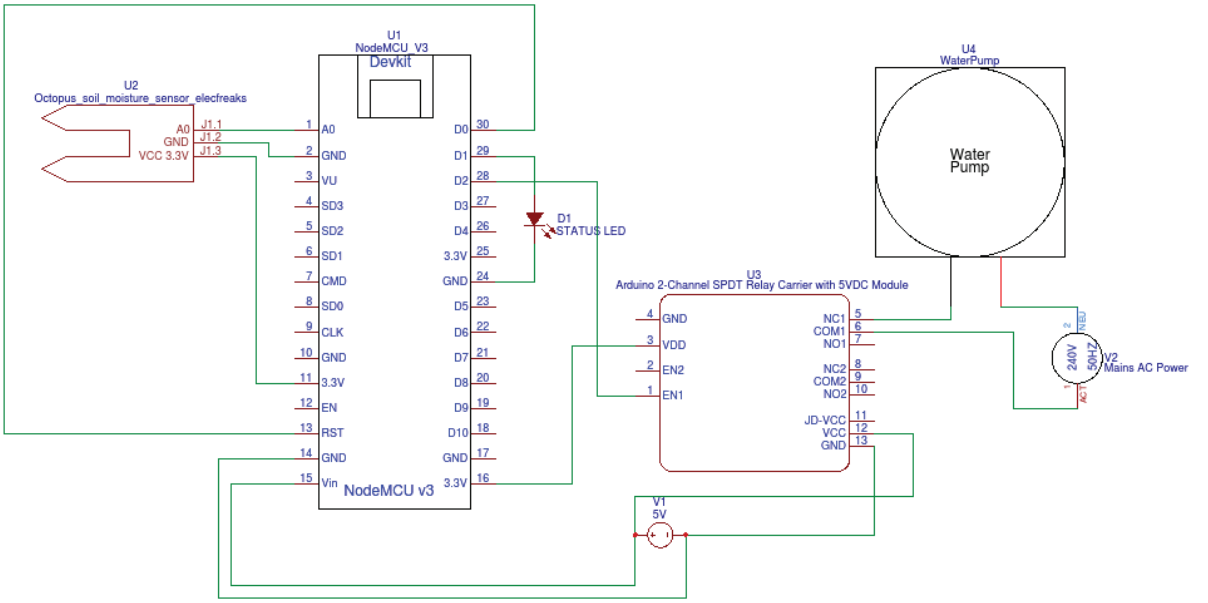
Figura 3 – Diagrama de blocos JIIOT com NODEMCU.



Fonte: Elaborado pelo autor (2016).

Após a adaptação dos componentes para o novo hardware, o esquema elétrico foi configurado como mostrado na Figura 4.

Figura 4 – Esquema elétrico (NODEMCU).



Fonte: Elaborado pelo autor (2016).

As mesmas bibliotecas foram utilizadas para atingir o objetivo de troca de mensagens com o broker MQTT. O NODEMCU mostrou estabilidade e velocidade muito boas ao conectar-se à rede, fazendo com que as trocas de mensagens por MQTT sofressem perdas mínimas, dependendo apenas da estabilidade da rede.

Para finalizar a estrutura do core de sensores, foi utilizado um recipiente de plástico para proteger os componentes de impactos e do clima, ficando o módulo de relés alocado embaixo, o NODEMCU em cima, a bomba d'água em outro recipiente e o sensor de umidade de solo fincado na terra sobre a qual se faria a leitura Figura 5. O NODEMCU e o módulo de relés foram separados no recipiente por um pedaço de esponja, para não haver contato entre pinos Figura 6.

Figuras 5 e 6 - Core de sensores e módulo relés e NODEMCU.

Fonte: Elaborado pelo autor (2016).

Para implementar um broker MQTT foi escolhida a plataforma de hardware Raspberry Pi devido às facilidades trazidas por possuir um sistema operacional embarcado próprio, o Raspian. O Raspbian é um sistema operacional Debian compilado para a plataforma. Por possuir um sistema operacional bastante difundido, a implementação do broker foi feita de forma direta, sendo necessário apenas instalar o programa que realizaria a tarefa de receber as mensagens, o Mosquitto.

Após instalado, o broker Mosquitto inicia automaticamente na porta 1833, padrão do MQTT. Para que o NODEMCU possa se comunicar com o Raspberry, os dois devem estar na mesma rede.

2.9.2 Conexão com serviços Amazon AWS

Com o Raspberry Pi já configurado para receber mensagens por MQTT do NODEMCU, resta configurar a conexão com os serviços da Amazon e implementar a lógica de quando o serviço para notificação do usuário deve ser ativado. Seguindo o guia fornecido pela Amazon de como configurar um Raspberry Pi para realizar conexão via Javascript, foi utilizado o Node JS (interpretador de códigos javascript) e a biblioteca aws-iot-device-sdk para realizar a conexão com o serviço AWS IoT.

As seguintes bibliotecas javascript também foram utilizadas para alcançar a lógica final do script e tratar as mensagens provenientes do NODEMCU: a biblioteca mqtt, para tratar as mensagens recebidas do NODEMCU via MQTT; e a biblioteca pg, que permite a utilização do banco de dados PostgreSQL para o armazenamento dos dados recebidos localmente.

Para o correto funcionamento do script, foi necessário configurar os certificados fornecidos pela Amazon no painel de configurações do serviço AWS IoT. Além das configurações de certificados, foram criadas regras de integração com o serviço AWS SNS, que disponibiliza notificações de vários tipos. O tipo escolhido para este trabalho foi a notificação por e-mail, por não ter custo financeiro.

A regra criada no AWS IoT para iniciar o serviço de notificação para os e-mails cadastrados foi a seguinte:

SELECT state.desired.soil_moisture, state.desired.ip as sensor_ip FROM '#' WHERE state.desired.soil_moisture = 'low' OR state.desired.soil_moisture = 'high' OR state.desired.soil_moisture = 'undefined'.

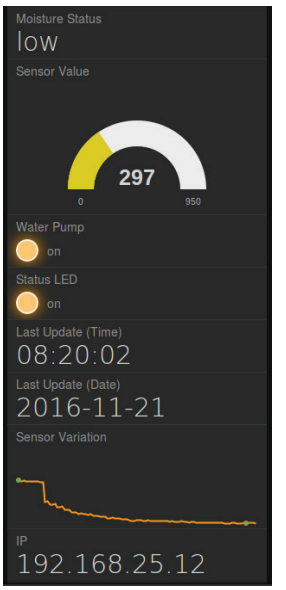
As regras são criadas em uma sintaxe semelhante ao SQL. A regra acima vai disparar e-mails caso receba um parâmetro soil_moisture contendo o valor *low, high ou undefined;* e enviará os valores dos parâmetros soil_moisture e ip como sensor_ip no corpo do e-mail.

2.10 Monitoramento de dados

Foram utilizados três serviços para tornar o monitoramento de dados possível: Plot.ly, para plotar gráficos; Dweet.io, para armazenar os dados mais recentes da aplicação; e o Freeboard.io, que transforma os dados do Dweet.io em dados apresentáveis. Tanto o Plot.ly quanto o Dweet.io possuem bibliotecas para o Node JS, fazendo com que fossem facilmente incorporados na lógica residente no Raspberry Pi. As bibliotecas se chamam plotly e node-dweetio.

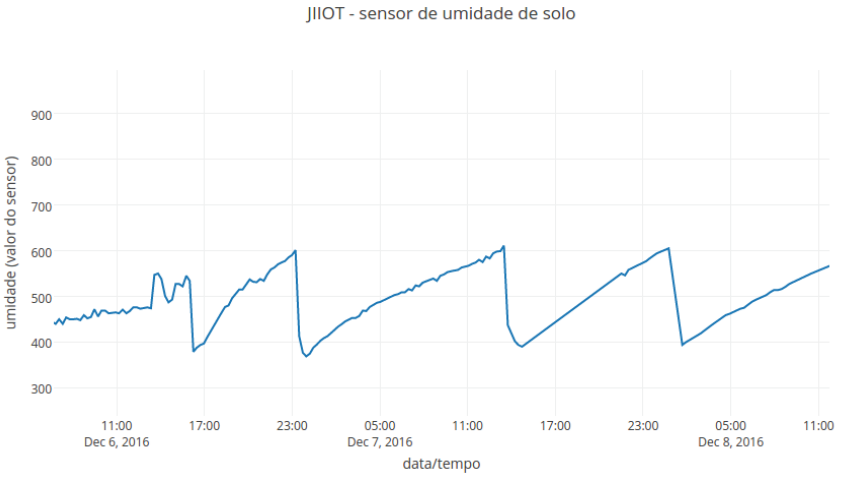
Com a integração dos dados realizada, foi possível montar um quadro no sistema Freeboard.io (Fig. 7) que monitora a umidade do solo e outros dados, e atualiza a cada envio de dados ao broker MQTT. O mesmo ocorre para a atualização do gráfico criado pelo serviço Plot.ly Figura 8. Os dois podem ser acompanhados em tempo real pelos links https://plot.ly/~diegommarino/5/ e https://freeboard.io/board/I8aa_x.

Figura 7 – Monitoramento Freeboard.io.



Fonte: Elaborado pelo autor (2016).

Figura 8 – Monitoramento Plot.ly.



Fonte: Elaborado pelo autor (2016).

2.11 Validação do Ambiente IoT

O controle de irrigação deveria seguir a seguinte lógica: tentar conectar-se à rede, tentar conectar-se ao broker MQTT, realizar a irrigação se necessária, e, por último, enviar o status atual do sistema ao broker MQTT. Para validar a lógica descrita, o procedimento a seguir foi realizado.

Para simular o cenário de solo com baixa umidade, o sensor foi removido do solo, fazendo com que a bomba d'água e o LED de status fossem ativados. O solo não foi encharcado, pois a lógica desenvolvida realiza a irrigação por 5 segundos e refaz a verificação do valor do sensor de umidade, realizando esse procedimento três vezes, e então o sistema entra em modo de repouso.

Ao inserir o sensor no solo úmido, o sistema se comportou como esperado, permanecendo com a bomba d'água e LED de status desligados, apenas realizando o procedimento de enviar o status em que o sistema se encontrava para o broker MQTT.

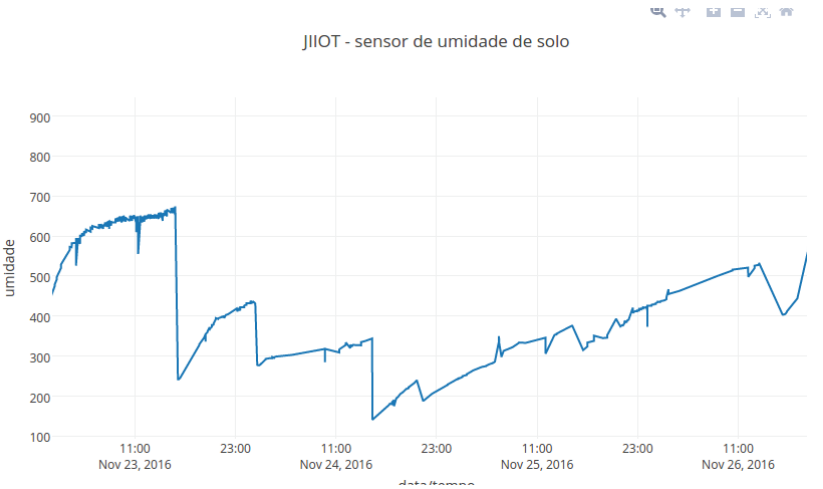
Para analisar a comunicação e a estabilidade das plataformas durante a comunicação, alguns procedimentos foram realizados. Para verificar se o sistema de irrigação continuaria a funcionar sem rede e/ou na ausência de um broker MQTT, o Raspberry Pi (broker) foi desligado e, em seguida, o roteador também. O sistema continuou a realizar a irrigação de forma autônoma, mesmo incapacitado de enviar informações ao broker, ou se conectar à rede wi-fi. O mesmo procedimento já descrito acima foi feito para realizar o teste de funcionamento da irrigação.

Quanto ao broker MQTT, a lógica implementada apenas realiza a distribuição de dados para a plataforma AWS IoT, Plot.ly e Freeboard.io. Sendo assim, no caso da falta de conexão com a internet, o sistema apenas realiza o armazenamento de dados no banco de dados local.

No caso da queda da rede wi-fi, o broker consegue realizar a reconexão poucos instantes após o restabelecimento do roteador, como foi o caso do cenário de teste desligando e ligando o roteador wi-fi.

Para validar a lógica completa do ambiente, o sistema foi deixado em operação durante vários dias para que fosse feita a coleta de dados de status do sistema. Durante esses dias, a irrigação foi realizada de forma manual, a fim de colher dados de como o sensor e o sistema deveriam se portar. A Figura 9 mostra um gráfico dos valores coletados do sensor de umidade de solo. Os vales do gráfico representam o momento em que o sistema terminou de irrigar o solo.

Figura 9 – Valores do sensor de umidade de solo.

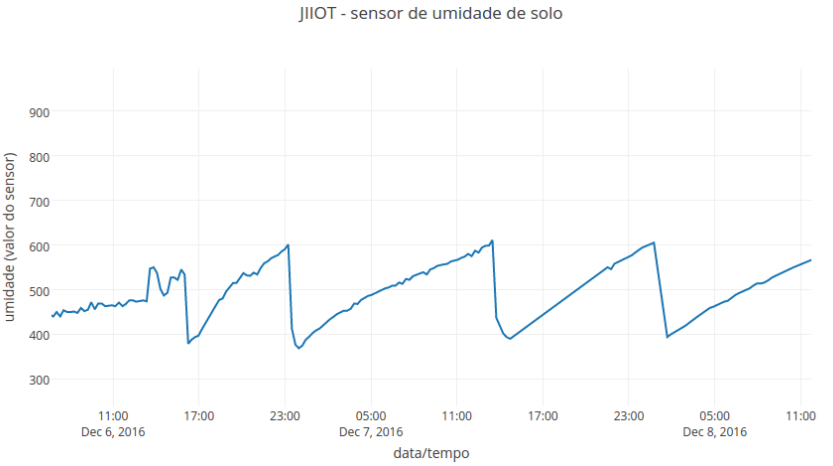


Fonte: Elaborado pelo autor (2016).

Pode-se notar pela figura que inicialmente houve uma grande variação nos valores que o sensor considerava como solo úmido ou seco. Após três dias, os valores do sensor de umidade de solo estabilizaram, indicando um valor entre 0 e 350 para solos muito úmidos, e entre 600 e 1050 para indicar quando o solo estava com baixa umidade. Sendo assim, o intervalo ideal para manter uma boa umidade do solo ficou entre 350 e 600.

Após a calibração do sistema, pôde-se notar uma boa estabilidade na leitura dos valores, dando segurança para anexar a bomba d'água ao sistema e iniciar a irrigação autônoma. Após o primeiro dia funcionando, a Fig. 10 mostra o gráfico gerado na plataforma Plot.ly.

Figura 10 – Umidade com sistema calibrado.



Fonte: Elaborado pelo autor (2016).

O sistema funcionou como esperado todas as vezes em que foi atingido o limite definido como baixa umidade de solo (600). O status do sistema foi enviado ao broker MQTT, que repassou os dados para o serviço da AWS IoT, fazendo com que o serviço detectasse que recebeu um dado de baixa umidade de solo e enviasse um alerta por e-mail ao usuário. A cada 15 minutos (intervalo de funcionamento do sistema), os dados na plataforma Freeboard.io foram atualizados, pois eram enviados pelo script presente no Raspberry Pi. A cada 10 tuplas inseridas no banco de dados, o gráfico da plataforma Plot.ly foi atualizado.

2.12 Situações adversas

O ambiente JIIOT foi concluído cumprindo com todos os objetivos, gerais e específicos. O projeto final consegue eficientemente irrigar uma área de terra de acordo com valores predefinidos de alta e baixa umidade de solo, o sensor de umidade de solo. O sistema realiza o monitoramento a cada 15 minutos e entra em estado de hibernação.

Ao realizar o monitoramento, os estados de funcionamento da bomba d'água, LED de status e valor do sensor de umidade de solo são enviados ao broker MQTT e, subsequentemente, à Amazon AWS IoT. Caso detecte que o solo possui uma baixa umidade, notifica o usuário via e-mail. Os dados também são enviados à plataforma Freeboard.io para monitoramento remoto e, a cada 10 dados, inseridos no banco de dados, sendo gerado um novo gráfico de histórico dos últimos 1000 status do sistema na plataforma Plot.ly.

Alguns problemas, os quais serão citados a seguir, ocorreram durante a execução deste projeto, mas, ao final, o ambiente e a lógica satisfizeram por completo os resultados esperados. Como o armazenamento de dados via banco de dados PostgreSQL só foi inserido no sistema próximo ao deadline de entrega, não foi possível confeccionar um gráfico com uma linha temporal maior.

Outro problema enfrentado foi uma inconsistência no sistema operacional Raspbian. Após algum tempo, variando entre duas horas e um dia, o Rasperry Pi se desconectava da rede wi-fi, não conseguindo restabelecer a conexão. Foram pesquisadas e aplicadas diversas configurações, mas sem sucesso na resolução desse problema. O problema foi contornado adicionando uma tarefa cron ao sistema, para que ele reiniciasse a cada meia hora.

No projeto inicial, o módulo WiFi ESP8266-01 foi utilizado para tornar possível a comunicação sem fio da plataforma Arduino UNO. Esse módulo possui uma documentação muito pobre do fabricante, mas uma comunidade de usuários significativa, o que ajudou bastante na sua configuração. Apesar disso, vários problemas foram enfrentados ao tentar anexar o módulo ao Arduino.

O primeiro empecilho enfrentado apareceu ao tentar alimentar o módulo. São requeridos 3.3v para o correto funcionamento do hardware, mas, ao ligá-lo à porta de saída do Arduino com a voltagem necessária, não era possível fazê-lo funcionar corretamente. Após muita pesquisa em fóruns, foi detectado que o notebook que estava sendo utilizado para alimentar o Arduino não fornecia corrente suficiente para ambos (o Arduino e o módulo). A solução foi utilizar uma fonte de 3.3v para alimentar o módulo.

Após resolver o problema de alimentação do ESP8266, outro problema foi identificado. O módulo não era capaz se comunicar via serial com o Arduino, pois não era possível configurar o Baud Rate ideal (9600 ou 115000) para comunicação entre os hardwares. Após mais pesquisas em fóruns, foi identificado que seria necessário realizar um upgrade no firmware do módulo, e só então seria possível configurar um Baud Rate fixo.

Sendo assim, após os problemas enfrentados, foi possível realizar a comunicação entre o módulo ESP8266 e o broker MQTT implementado no Raspberry Pi. Para realizar a troca de mensagens, foram utilizadas duas bibliotecas para a IDE Arduino, ESP8266WiFi e PubSubClient. A primeira simplifica o modo de conexão do módulo com a rede e a segunda implementa o protocolo MQTT para ser usado por clientes.

Mesmo obtendo sucesso na implementação do módulo, notou-se que ele possuía certa instabilidade, aleatoriamente não realizando conexão com a rede, inviabilizando a comunicação sem fio com o broker e tornando o sistema instável. As conexões de fio com o Arduino também o tornavam nada prático para fins de prototipagem inicial. Durante as pesquisas por soluções para os problemas que o ESP8266 trazia, foi encontrada outra plataforma de hardware que o utilizava como base e que estendia os pinos de seu microcontrolador para que funcionasse de forma similar a um Arduino, o NODEMCU. O NODEMCU foi a plataforma utilizada na versão final do ambiente IoT.

Com a configuração final do sistema, foi possível realizar o monitoramento da umidade do solo a cada quinze minutos e, logo em seguida, enviar os dados para a AWS IoT com uma boa frequência, dependendo apenas da estabilidade da rede em que o NODEMCU e o Raspberry Pi estavam conectados.

Como comentado anteriormente, o Arduino UNO, em conjunto com o módulo ESP8266-01, não conseguiu o desempenho necessário para o cenário proposto neste trabalho. Já o NODEMCU obteve um desempenho surpreendente em velocidade e alcance de conexão.

O serviço AWS IoT se comportou muito bem, executando suas regras todas as vezes em que lhe foram enviadas mensagens, atualizando também o serviço de sombra do dispositivo (thing). Para pessoas acostumadas com sintaxe SQL

e Javascript (ou Python, ou C++), as funções e regras do serviço podem ser compreendidas com bastante facilidade, fazendo com que a curva de aprendizado seja bem suave. Outro fato interessante é possuir integração com quase todos os outros sistemas da Amazon AWS. No caso deste estudo, a integração realizada foi com o serviço de mensagens AWS SNS.

O AWS SNS se mostrou um pouco falho quando são enviadas muitas mensagens em curto espaço de tempo. Algumas vezes as notificações por e-mail não foram entregues, ou foram entregues com atraso entre trinta minutos e duas horas. Cabe estudar esse comportamento para verificar se foi falha de codificação e/ou uma falha do serviço.

3 Conclusão

Montar um ambiente de IoT com auxílio de serviços na nuvem da *Amazon* pode ser uma solução eficiente, pois permite uma integração com a maioria dos serviços disponibilizados pela empresa, como processamento de dados, armazenamento de arquivos, sistemas de notificações e armazenamento de dados. Escalar um sistema nesse cenário pode ser algo interessante se houver um número crescente de dispositivos a serem conectados, além de possuir uma boa segurança para os dados armazenados e trocas de mensagem devido ao sistema de autenticação por certificados.

Os desafios para o JIIOT se tornar um ambiente de IoT próximo do ideal ainda são muitos. Este trabalho não abordou trocas de mensagens criptografadas, ou com autenticação na rede local, um elemento essencial em uma rede de IoT e que ainda está sendo amplamente debatido. Pensando em trabalhos futuros, seria de grande interesse realizar um estudo de mensagens via protocolo MQTT com autenticação e criptografia. Outra sugestão de trabalho seria armazenar os dados diretamente com o serviço oferecido pela *Amazon*, o *DynamoDB*, um banco de dados não relacional, integrando-o com a AWS IoT, além de tirar um maior proveito de sua ferramenta de sombra de dispositivos.

Referências

AWS IoT. Disponível em: <https://aws.amazon.com/pt/iot/>. Acesso em: 28 nov. 2016.

BETTER insights. Better treatment. Better care**.** Disponível em: <http://www.proteus.com/ discover/>. Acesso em: 31 set. 2016

BRUCE, J. **Meet the Arduino Killer:** ESP8266. 2015. Disponível em: <http://www.makeuseof. com/tag/meet-arduino-killer-esp8266/>. Acesso em: 17 set. 2016.

BUDDY. **Connected vehicles**. Disponível em: <https://buddy.com/industries/connected-vehicles/>. Acesso em: 17 set. 2016.

COMO a plataforma do AWS IoT funciona**.** Disponível em: <https://aws.amazon.com/pt/iot/how-it-works/>. Acesso em: 28 nov. 2016.

ERICSSON mobility report**:** on the pulse of the Networked Society. Disponível em: <https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf>. Acesso em: 10 jun. 2016.

HARRIS, I. **Introdução à internet das coisas e a sistemas embarcados**. Disponível em: <https://www. coursera.org/learn/iot/home/welcome>. Acesso em: 3 set. 2016.

I-SCOOP. **Internet of things:** online guide to the internet of things. Disponível em: <http://www.i-scoop.eu/internet-of-things/>. Acesso em: 20 set. 2016.

MAHALE, R. B.; SONAVANE, S. S. Smart poultry farm monitoring using IOT and wireless sensor networks. **International Journal of Advanced Research in Computer Science**, Udaipur, v. 7, n. 3, p. 187-191, May/June. 2016. Disponível em: <http://search.proquest.com/openview/7dfe615ff3604dc080d6f9743fbb8bc3/ 1?pq-origsite=gscholar&cbl=1606379>. Acesso em: 3 out. 2016.

NODEMCU connect things easy. Disponível em: <http://www.nodemcu.com/index_en. html>. Acesso em: 28 nov. 2016.

PATHAK, P. B. Internet of things: a look at paradigm shifting applications and challenges. **International Journal of Advanced Research in Computer Science***,* Pune, v. 7, n. 2, p. 49–51, Mar./Apr. 2016.

PRODUTOS em nuvem. Disponível em: <https://aws.amazon.com/pt/products/?nc2=h_ql_ny_ livestream_blu>. Acesso em: 28 nov. 2016.

RIDICULOUSLY simple messaging (and alerts) for the internet of things. Disponível em: <http://dweet.io/>. Acesso em: 21 nov. 2016.

SMITH, H. **Inventor looks to shake up cleantech with smart so- lar trackers**. Disponível em: <http://grist.org/business-technology/ inventor-looks-to-shake-up-cleantech-with-smart-solar-trackers/>. Acesso em: 23 set. 2016.

SMITH, J. E.; WEINGARTEN, F. (ed.). **Research challenges for the next generation Internet**. Washington: Computing Research Association, 1997. 78 p. Disponível em: <https://nitrd.gov/ngi/pubs/research/_research_chall.pdf>. Acesso em: 20 jun. 2016.

VASCONCELOS, D. R. D. **Survey sobre FOG computing no ambiente de internet das coisas (IoT)**. 2016. 254f. Tese (Doutorado em Ciência da Computação). Universidade Federal do Ceará, Fortaleza, 2016.

WHAT is MQTT? Disponível em: <http://mqtt.org/faq>. Acesso em: 21 nov. 2016.

Sobre os autores

Diego Rios Donato Marinho Marino
Graduado em Engenharia da Computação. Desenvolvedor de sistemas da web em Pythonda - Defensoria Geral do Estado do Ceará. Atuou com pesquisas e programação em Assembly a Ruby on Rails.

Danilo Reis Vasconcelos
Engenheiro Civil, Universidade Federal do Ceará – UFC, Mestre em Engenharia Civil, área de Estruturas, Universidade Federal do Rio de Janeiro – COPPE/UFRJ, Doutor em Engenharia Oceânica, área de Estruturas, Universidade Federal do Rio de Janeiro – COPPE/UFRJ, Pesquisador de Pós-Doutorado da Armtec (PDI/CNPq), Núcleo de Pesquisas Tecnológicas, Incubadora de Empresas, NPT/Unifor.

Suellen Galvão Moraes
Especialista em Gestão Ambiental – Unifor. Turismóloga - FIC. Estudante de Engenharia Ambiental e Sanitária, 6˚ Semestre – Unifor.