An IEEE 1451 Standard-based Plug-and-Play Architecture to Empower the Internet of Things

Melissa Santos Aguiar, Francisco Henrique Cerdeira Ferreira, Eduardo Pagani Julio, Bruno José Dembogurski, Gustavo Silva Semaan, and Edelberto Franco Silva, *Member, IEEE*

***Abstract*—In this paper, we introduce a new Internet of Things (IoT) solution based on a Plug-and-Play (PnP) architecture, aiming to stimulate new paths set by the concept of the Smart Environment. In this sense, the IoT ecosystem is considered, and the layers are presented from the higher level of software to the lowest level of sensors/actuators. We extend the state of art solutions since our proposal features are transparent in both hardware and software domains, supporting the identification of sensors through an easy and intuitive tool responsible for creating and managing all resources and logic programming in an IoT environment. To show the main contributions of this research, we will introduce the architecture, adding the IEEE 1451 standard, based on well-defined message protocols. The contribution of this work encompasses all PnP levels in the IoT architecture, from the platform to the user interface, which are different from all approaches currently existent in state of the art. Moreover, a proof of concept is described at the end of this article.**

***Index Terms*—IoT; Plug-and-Play; IEEE 1451; MQTT.**

I. INTRODUÇÃO

INTERNET das Coisas, ou IoT (*Internet of Things*), refere-se à interconexão de objetos físicos do cotidiano, de forma a permitir a troca de dados entre eles e seus sistemas através da rede mundial de computadores, onde os processamentos de dados do dia a dia se tornam mais inteligentes e a comunicação mais informativa [1], [2]. Tais dispositivos podem compreender desde eletrônicos domésticos comuns até equipamentos industriais mais sofisticados. Desta forma, estima-se que o impacto econômico da IoT alcançará 11 trilhões de dólares até o ano de 2025 [3].

Melissa Santos Aguiar, Department of Electrical Engineering, Universidade Federal de Juiz de Fora - UFJF, Juiz de Fora, MG, Brazil. e-mail: melissa.aguiar@engenharia.ufjf.br

Edelberto Franco Silva, Eduardo Pagani Julio and Francisco Henrique Cerdeira Ferreira, Department of Computer Science, UFJF, Juiz de Fora, MG, Brazil. e-mails: edelberto@ice.ufjf.br

Bruno José Dembogurski, Department of Computer Science, Universidade Federal Rural do Rio de Janeiro, Nova Iguaçu, RJ, Brazil.

Gustavo Silva Semaan, Departamento de Ciências Exatas Biológicas e da Terra, Universidade Federal Fluminense, Sto. Antônio de Padua, RJ, Brasil. e-mail: gustavosemaan@id.uff.br

Manuscript received May 12, 2020; revised June 30, 2020.

Os *hardwares*, em um nível mais baixo de abstração, permitem que esse ambiente funcione. Nesse nível estão compreendidos os dispositivos *transdutores*, *sensores* e os *atuadores*. O transdutor converte uma forma de energia em outra, o sensor converte um parâmetro físico em uma saída elétrica e o atuador converte um sinal elétrico em uma saída física. Portanto, uma rede IoT pode ser definida como uma rede inteligente que tem como objetivo identificar, localizar, rastrear, monitorar e gerenciar “coisas” - que, por sua vez, estão conectadas à Internet - e atua na comunicação entre dispositivos por meio de seus protocolos [1], [4].

Nesse tipo de rede é necessário o estabelecimento de um padrão de comunicação entre os objetos, já que existem uma grande diversidade em termos de fabricantes e plataformas. O padrão IEEE 1451 suporta desde a identificação por radiofrequência (RFID) a diversos sensores e tecnologias que, em conjunto, atendem a uma vasta gama de aplicações. Por isso, este trabalho irá apoiar-se nesse padrão como referência para a proposta da solução IoT.

Embora o uso de dispositivos cada vez mais inteligentes tenha aumentado substancialmente nos últimos anos, desafios ainda existem para a IoT. Com o objetivo de estimular novos caminhos definidos pelo conceito de Ambiente Inteligente, este trabalho apresenta uma nova solução baseada na arquitetura *Plug-and-Play* (PnP) do IEEE 1451. O paradigma PnP traz simplicidade no uso e integração de novos dispositivos à essa rede, de forma que, direta ou indiretamente, esta arquitetura pode ser usada para resolver diversos problemas da vida real [1], [4], [5].

Desta forma, por meio de uma investigação da literatura, foi possível constatar que os sistemas existentes apresentam carências relacionadas a: (i) *interface* gráfica do usuário (GUI) para facilitar as operações e configurações de seu ambiente de IoT; (ii) a necessidade de um método PnP transparente para plataformas de IoT respeitando o padrão IEEE 1451 e (iii) o requisito para diminuir o processamento e consumo de recursos em um nó da IoT. Com base nessas lacunas, a principal contribuição deste trabalho é a introdução de uma

arquitetura leve, flexível e escalável, na qual o usuário consiga configurar diversos sensores e atuadores como PnP de forma transparente e com o menor esforço possível. Diferentemente das soluções existentes, a arquitetura apresentada pode ser aplicada a qualquer tipo de ambiente, seja na indústria, na cidade ou no campo, em um tempo mais curto e com menor esforço do usuário. Portanto, este artigo apresenta uma proposta para preencher as exigências do padrão IEEE 1451 com aplicativos e protocolos reais de IoT. É descrita a arquitetura completa da solução, seus resultados que comprovam sua facilidade, flexibilidade e escalabilidade através de uma prova de conceito implementada em um ambiente de testes real.

O restante do artigo é organizado da seguinte maneira: a Seção II aborda o estado da arte de propostas e soluções relacionadas ao PnP; a Seção III apresenta a arquitetura proposta e seus padrões; a Seção IV apresenta a avaliação e validação da solução e, por fim, na Seção V, as conclusões e possíveis trabalhos futuros são apresentados.

II. TRABALHOS RELACIONADOS

Nesta seção são apresentados os principais trabalhos e soluções relacionadas ao contexto da presente proposta, desde soluções de *software* a *interfaces* PnP e soluções de mercado (e *black box*).

O primeiro é o trabalho de Nieves *et al.* [6], onde foi proposto um conjunto de ações, bem como um conjunto de variáveis de estado chamado uPnP (*Universal Plug and Play Smart Transducer*). Uma dessas váriaveis fornece uma lista de canais disponíveis e a outra fornece a lista de transdutores acessíveis. O protocolo uPnP visa trocar serviços entre dispositivos conectados a uma rede TCP/IP. O PnP no uPnP é baseado no controle e nas tuplas do canal responsável pela implementação do transdutor e pela identificação dos sensores conectados à rede de controle. Esta solução segue a recomendação da arquitetura IEEE 1451, tendo o NCAP (*Network Capable Application Processor*) como um *gateway* para permitir a conexão de um SM (*Sensor Manager*), ou seja, um elemento TIM (*Transducer Interface Module*) no IEEE 1451 - através de uma rede Wi-Fi. Os principais benefícios do uPnP são permitir o endereçamento, descoberta, descrição, controle e eventos para os sensores conectados à rede.

A plataforma MicroPnP [7] é uma solução comercial desenvolvida pela *VersaSense*. Os detalhes do funcionamento do seu PnP não são mostrados. O MicroPnP é composto por três sensores/atuadores simultâneos conectados à placa e os mesmos precisam fazer parte de uma lista de elementos suportados. Embora a plataforma ofereça atualmente mais de vinte e cinco periféricos que podem ser associados ao dispositivo PnP, que se ajustam a uma variedade de situações cotidianas, o dispositivo central suporta apenas três periféricos, limitando bastante a sua gama de aplicações.

Já a plataforma *IoT Plug and Play from Azure*[1], da *Microsoft*, propõe facilitar o registro de um dispositivo de IoT na plataforma *Azure*, o que torna possível a criação de suas soluções de IoT. Porém, é necessário destacar que o conceito PnP tem como foco apenas o nível de aplicação, e a incorporação das plataformas IoT já existentes à plataforma *Microsoft Azure* é de forma manual. Na mesma linha tem-se o trabalho de Martinez *et al.* [8], que propõe um *middleware* para a realização do PnP. Seguindo a mesma linha da plataforma da *Microsoft*, o PnP se dá apenas em nível de *software*, não tratando as demais camadas de *hardware* e comunicação descritas pelo padrão IEEE 1451. Em Kim *et al.* [9] é proposta uma plataforma genérica de PnP baseado em ontologia e na semântica para identificar e tratar os dados dos sensores/atuadores. É importante destacar que a proposta é vinculada a um *driver* genérico que permitiria a comunicação com qualquer tipo de sensor/atuador. Desta forma, os autores propõem uma plataforma genérica também do ponto de vista apenas de abstração de *software* e não completa como a apresentada neste artigo.

Em Yi *et al.* [10] é proposto um sensor com a função PnP baseado em uma arquitetura de sistema modular de sensor utilizando uma *interface* universal de sensor, que utiliza um microcontrolador e barramento I^2C para comunicação. Apesar de abordar o tema PnP para sensor, nosso trabalho se difere por prever o suporte à escalabilidade de conexão de sensores e menor dependência de desenvolvimento, integrando sensores já existentes de forma facilitada à nossa arquitetura. Em Engelstad *et al.* [11] tem-se a patente dos componentes integrados para sensores PnP, onde há a dependência de módulos como o microcontrolador e memórias próprias ao sensor, não levando em consideração o padrão IEEE 1451.

Por fim, tem-se o trabalho de Lim *et al.* [12], onde é apresentada uma plataforma desenvolvida no âmbito da empresa GE (*General Electric*) para suportar de forma PnP alguns sensores para monitoramento do ambiente industrial. Desta forma, o trabalho é restrito e dependente de ambiente e *hardware*, diferentemente do proposto neste trabalho.

Em contraste com os trabalhos relacionados, o presente trabalho trata-se de uma arquitetura PnP completa e aberta, respeitando os padrões IEEE 1451, de forma a criar uma integração e identificação transparentes dos sensores/atuadores à *interface* do usuário, permitindo que ele crie sua própria aplicação para monitoramento e atuação no ambiente de IoT. Outro

[1] https://azure.microsoft.com/en-us/updates/iot-plug-and-play/

diferencial é a escalabilidade, onde cada plataforma baseada em nossa solução pode suportar aproximadamente 127 sensores/atuadores simultaneamente – limitação imposta pelo barramento I^2C – e o número de plataformas que podem executar simultaneamente é superior a cem. A Tabela I mostra as características de cada uma das soluções, e comprova os benefícios da solução proposta neste trabalho, atendendo a todos os requisitos.

Tabela I: Tabela de comparação entre os trabalhos relacionados e a arquitetura proposta neste trabalho.

	Escalabilidade	**Nível de PnP**	**Apoiado no IEEE 1451**	**Padrão Aberto**
uPNP	Sim	Hardware	Sim	Sim
microPNP	Não	Hardware	Não	Não
Microsoft	N/A	Aplicação	Não	Não
Martinez *et al.*	N/A	Aplicação	Não	Não
Kim *et al.*	N/A	Aplicação	Não	N/A
Yi *et al.*	Não	Hardware	Não	N/A
Engelstad *et al.*	N/A	Hardware	Não	Não
Lim *et al.*	Não	Hardware	Não	Não
Este trabalho	Sim	Aplicação e Hardware	Sim	Sim

III. ARQUITETURA PROPOSTA

Esta seção apresenta a arquitetura proposta e todos os padrões envolvidos para sua compreensão.

A. Padrões

1) O padrão IEEE 1451: o padrão IEEE 1451 existe há quase 20 anos e, nesse período, um considerável avanço ocorreu nas tecnologias dos sistemas e sensores inteligentes, com destaque para: o controle e monitoramento em tempo real, instalações PnP, *interfaces* para aplicativos baseados em servidores, etc. O Instituto Nacional de Padrões e Tecnologia (NIST - *National Institute of Standards and Technology*) promoveu a série de padrões IEEE 1451.X para sensores inteligentes, que estabelecem uma base para o desenvolvimento de redes de sensores "inteligentes".

O padrão IEEE 1451.0 define as funções executadas por um (S)TIM (*(Smart) Transducer Interface Module*) e as características comuns para sua implementação. Já o padrão IEEE 1451.1 define um modelo comum de objeto e paradigma de programação para sistemas de transdutores inteligentes. O IEEE 1451.2 caracteriza um transdutor para *interface* NCAP e TEDS (*Transducer Electronic Data Sheets*) para uma configuração ponto a ponto [2]. O IEEE 1451.3 define uma *interface* de transdutor para NCAP e TEDS para transdutores multiponto usando o protocolo de comunicação HPNA (*Home Phoneline Networking Alliance*) [13]. O IEEE 1451.4 define uma *interface* de modo misto para transdutores analógicos com modos de operação analógico e digital.

O padrão IEEE 1451.5 define uma *interface* de transdutor para NCAP e TEDS para transdutores sem fio, que podem ser entendidos por padrões como 802.11 (Wi-Fi), 802.15.1 (*Bluetooth*), 802.15.4 (LR-WPAN ou "*ZigBee*") e 6LoWPAN. O IEEE 1451.6 define uma *interface* de transdutor para NCAP e TEDS usando uma *interface* de rede aberta de alta velocidade. E, por fim, o IEEE 1451.7 define uma *interface* e um protocolo de comunicação entre transdutores e sistemas RFID. Para o padrão IEEE 1451.1 há diversos subgrupos de protocolos específicos para trocas de mensagens, como é o caso do IEEE 1451.1.6, que suporta o protocolo de mensagens baseado em telemetria que será apresentado logo a frente neste trabalho, enquanto o IEEE 1451.1.4 suporta a troca de mensagens em XMPP (*Extensible Messaging and Presence Protocol*) e o IEEE 1451.1.2 suporta o HTTP (*Hypertext Transfer Protocol*), por exemplo.

2) MQTT: para a troca de mensagens entre o NCAP e o cliente final, ou sua *interface* de aplicação e interação, existe o MQTT [14] (*Message Queue Telemetry Transport*). O MQTT[2] é um protocolo de mensagens leve, que emprega o paradigma *publish/subscribe*, projetado especificamente para a comunicação máquina para máquina (M2M) e IoT [14]. O objetivo deste é que os dispositivos possam publicar em algum lugar (*i.e.* tópicos), para que outros possam então obter essas informações publicadas. Em [15], é apresentado como o MQTT pode ser incorporado como parte do IEEE 1451 (IEEE 1451.1.6[3]) e em [16] é proposta a incorporação da versão recente mais do MQTT, chamada MQTT-SN (MQTT para Redes de Sensores), referente ao que seria o IEEE 1451.1.7.

Este trabalho não foca a troca de mensagens segura entre as entidades, não realiza medição de impacto sobre esta característica e não indica uma solução única a ser adotada. Porém, destacamos que o protocolo MQTT suporta a autenticação por tópico através de usuário e senha. Essa autenticação é simples e não pode ser considerada segura do ponto de vista de troca de mensagens, uma vez que o tráfego da mensagem com as credenciais deve ser protegido através de implementações de algoritmos de segurança na camada de aplicação ou por certificados, *e.g.* utilizando SSL (*Secure Sockets Layer*). Neste trabalho foi utilizada a comunicação segura por usuário e senha com SSL, mas seu impacto em desempenho foi ignorado neste momento.

O MQTT entrou em um processo de padronização pela OASIS (*Organization for the Advancement of Structured Information Standards*) [17], onde teve as especificações publicadas abertamente com uma licença sem *royalties*.

No MQTT há três elementos básicos para a comunicação: o publicador (*publisher*), o usuário/apli-

[2]http://mqtt.org

[3]Atualmente já incorporado à arquitetura.

cação inscrito(a) (*subscriber*) e o *broker* (agente intermediário). O publicador é um dispositivo responsável por coletar informações e enviá-las para o *broker*. O *broker* recebe a mensagem e a envia para todos os usuários registrados, para que eles possam acessar as informações. A Figura 1 mostra a operação do protocolo.

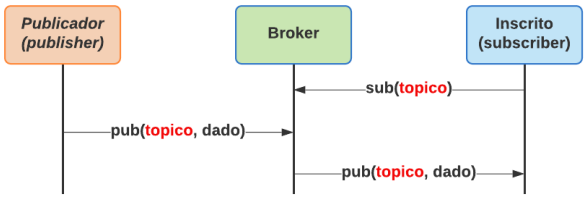

Fig. 1: Modelo de comunicação pub/sub baseado em tópicos usando o protocolo MQTT.

3) NodeRED: como ferramenta de aplicação responsável por realizar a *interface* com o cliente final (também chamada de GUI - *Graphic User Interface*) e permitir a comunicação facilitada com todo o arcabouço IEEE 1451, adotamos o NodeRED. Destacamos que quaisquer solução que implemente suporte ao MQTT pode ser utilizada no lugar do NodeRED. Para nossa demonstração utilizaremos daqui para frente o NodeRED como nossa GUI, a fim de facilitar o entendimento da arquitetura proposta e sua validação.

O NodeRED é uma ferramenta *web* de desenvolvimento de código aberto para conectar componentes de IoT, originalmente desenvolvida pela equipe de serviços de tecnologia emergente da IBM, e agora parte da *JS Foundation*[4]. Ela permite a programação visual baseada em fluxo, que descreve o comportamento de um aplicativo como uma rede de nós, o que simplifica o processo de desenvolvimento de soluções.

A forma de operação do NodeRED pode ser vista na Figura 2. Nela há a descrição dos três tipos diferentes de nós suportados pela ferramenta: nós de entrada, de processo e de saída. A caixa ao lado direito dos nós de entrada indica que estes nós apenas enviam informações. Os nós de processamento têm duas caixas, uma para entrada e uma segunda para saída do resultado do seu processo. Por fim, os nós de saída têm uma caixa no seu lado esquerdo para receber informações e enviá-las no formato correto.

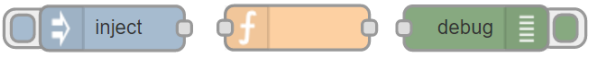

Fig. 2: Os três tipos principais de nós no NodeRED. Da esquerda para a direita: entrada, processo/função e saída.

[4]https://nodered.org

B. Arquitetura Proposta

A solução proposta nesse trabalho reconhece automaticamente mais de 100 sensores como dispositivos PnP, permitindo que a mesma seja aplicada em diversos domínios. Sensores analógicos e digitais compõem a lista de sensores apresentada na Tabela II, destacando que tal lista representa apenas os sensores utilizados na validação deste trabalho, e representa apenas uma pequena parcela dos sensores totais suportados pela solução. A comunicação entre o microcontrolador é feita por meio do protocolo de circuito integrado I^2C, que é um protocolo de comunicação serial para *interface* de dois fios que conecta dispositivos de baixa velocidade como microcontroladores, EEPROMs (*Electrically Erasable Programmable Read-Only Memory*), conversores A/D e D/A (analógico/digital e digital/analógico), *interfaces* de E/S (entrada e saída) e outros periféricos semelhantes em sistemas embarcados. O I^2C é usado por quase todos os principais fabricantes de circuitos integrados.

A arquitetura proposta e seu microcontrolador são totalmente genéricos, sendo sua principal função identificar os sensores conectados e enviar seus dados para um servidor através de um protocolo de troca de mensagens assíncronas. No protótipo utilizamos um microcontrolador ATmega328p, com 32KB de memória Flash, 16MHz de *clock* de CPU com cristal, 2kB de SRAM, 32 pinos e intervalo de voltagem de 1.8V a 5.5V. O NodeRED, por sua vez, é responsável por executar as tarefas do servidor de lógica e tomada de decisões. Nele, tem-se todo o processo de identificação de dados e, se necessário, é possível controlar os atuadores. O NodeRED é genérico para todos os microcontroladores existentes, permitindo o controle de várias caixas (ou dispositivos/funções) simultaneamente.

A arquitetura proposta é descrita, em uma visão de alto nível, pela Figura 3. Nela é possível observar todas as camadas e entidades envolvidas na família padrão IEEE 1451 *i.e.* (S)TIM, TEDS e NCAP. Pode-se destacar todas as três camadas às quais esta proposta se encaixa, com destaque maior à camada de aplicação e sua *interface* com o usuário.

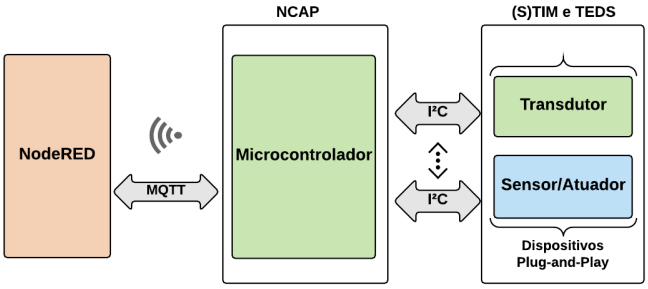

Fig. 3: Arquitetura proposta em alto nível.

Tabela II: Tabela de sensores suportados e testados pela solução.

Digital	**Analógico**
Analog/Digital Hall Sensor (49e), Switch Hall Sensor, RGB LED, Dual-color Common-Cathode LED, Shock Switch, Knock Sensor Laser Transmitter, Reed Switch, Mini Reed, Infrared Receiver, Digital Temperature Sensor, Active Buzzer, Passive Buzzer, Button Switch, Photo-interrupter, Tilt Switch, Mercury Switch, Magic Cup, DS18B20 Temperature Sensor, 7-color Auto-flash LED, Photoresistor Sensor, Obstacle Avoidance Sensor, Tracking Sensor, Metal Touch Sensor, Flame Sensor, Relay Module, Joystick PS2, Sensor ultrasônico HC-SR04, Sensor de humidade do solo higrometo, Sensor de chuva e humidade, Gás Sensor MQ-5, Sensor Touch Capacitivo 223B, Digital IR Receiver Module, MEAS Vibration Sensor, GY-65 Modulo de Pressão Atmosférica Altímetro, Modulo Potenciômetro, LDR, Sensor de Batimento Cardíaco	Analog/Digital Hall Sensor (49e), Analog Hall Sensor (44e), Analog Hall Sensor (49e), Analog Temperature Sensor

Já a Figura 4 apresenta, com mais detalhes, como a arquitetura do padrão IEEE 1451 é composta, com destaque para a comunicação entre a GUI e NCAP pelo IEEE 1451.6 (MQTT) e entre o NCAP e o (S)TIM através do IEEE 1451.X (onde 'X' representa qualquer um dos tipos suportados para a comunicação entre os componentes).

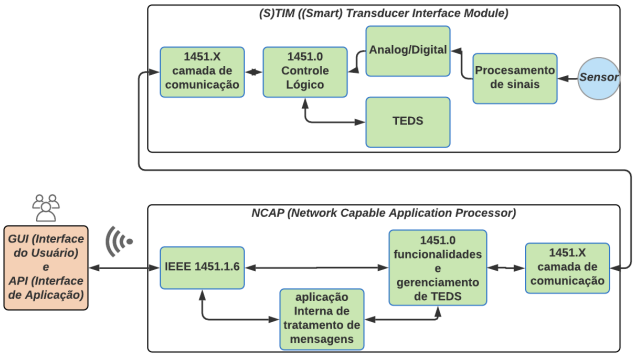

Fig. 4: Arquitetura do padrão IEEE 1451 com os protocolos utilizados para o PnP.

O reconhecimento por PnP de cada sensor na plataforma é possível pois o microcontrolador envia um ID[5] do respectivo sensor, de forma que cada dispositivo possa ser identificado ao se conectar. Esse ID é mapeado através do TED e encaminhado pela camada de comunicação IEEE 1451.X entre o (S)TIM e o NCAP. Como já citado, as entidades STIM e TEDS são responsáveis por exportar para o usuário uma *interface* de alto nível para interagir com sensores e atuadores. É nesta camada que ocorre a aquisição de dados e a funcionalidade PnP. A partir do transdutor, é possível interagir com os sensores/atuadores através de um barramento I^2C do microcontrolador. A comunicação é descrita na Figura 4, representada por uma comunicação Wi-Fi[6] que coleta e envia dados do (S)TIM, através do NCAP, para a *interface* do usuário, utilizando um protocolo de fila de mensagens suportado pelo padrão IEEE 1451, o MQTT [16], [17].

[5]Um número ID é um número de sequência exclusivo.

[6]Outras tecnologias de redes para interconexão também podem ser utilizadas no lugar do Wi-Fi.

No lado esquerdo da mesma figura é possível ver a aplicação NodeRED, uma *interface* de usuário responsável por permitir que o usuário interaja com sensores/atuadores de forma transparente, crie sua lógica de programação, obtenha a telemetria, dentre outros. Isso garante que todo o processamento que exige mais recursos computacionais ocorra no nível da aplicação.

Para concluir o detalhamento da proposta, tem-se o protótipo de placa/plataforma. Foi desenvolvido um par de placas com *interface* de conexão padrão para permitir uma conexão simples com qualquer tipo de sensor ou atuador que esteja listado na Tabela II (referente aos dispositivos já testados) ou que possua interface I^2C e tensão de operação entre 1,8V e 5,5V. Essas *interfaces* comuns podem ser qualquer tipo de *plug* serial, sendo necessário respeitar a ordem dos fios de conexão, de forma que o transdutor fica responsável por fornecer uma identificação exclusiva para cada sensor/atuador, permitindo o fluxo de dados entre o microcontrolador e o NodeRED. O formato dos dados utilizado é o JSON[7], representando o nó e seu sensor/atuador.

Juntamente com a placa principal, foi desenvolvida uma placa para os sensores, responsável por mapear qualquer pino de dispositivo para a *interface* de conexão padrão adotada na placa principal, abstraindo, assim, a conexão do dispositivo. Desta forma, a responsabilidade de tratar os dados coletados e criar todas as funções lógicas é da camada superior, *i.e.* a GUI, desonerando a plataforma de PnP dos custos de processamento advindos dessas obrigações.

IV. VALIDAÇÃO

O principal objetivo da arquitetura desenvolvida neste trabalho foi permitir que o usuário a aplicasse em muitos cenários diferentes, como em casas inteligentes, indústria, etc. Além disso, vale ressaltar que qualquer sensor ou atuador que se enquadre nos requisitos mencionados na proposta de arquitetura pode ser utilizado, definindo, no nível da aplicação, a lógica do PnP.

Essa seção demonstra a interação entre os componentes que fazem parte da arquitetura proposta. Na Figura 5, as etapas 1 a 4 são etapas de aquisição e

[7]https://www.json.org/

processamento de dados, o usuário apenas conecta os nós e cria as funções lógicas para tratá-los.

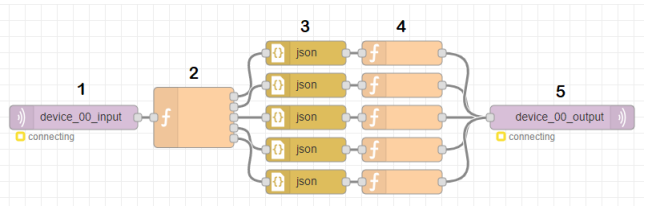

Fig. 5: Programação Baseada em Fluxo para o estudo de caso.

Como uma apresentação de baixo para cima, um sensor digital envia os dados brutos e sua identificação exclusiva para o nível da aplicação através das camadas intermediárias, ou seja, rede, NCAP e aplicação (NodeRED).

Nas etapas destacadas na Figura 5, é possível ver um bloco chamado `device_*_input` indicado por 1. Esse bloco é responsável por receber os dados brutos no formato JSON do `broker` MQTT. Todas as plataformas com o mesmo tipo/modelo de sensores/atuadores que foram identificados publicam no mesmo tópico com seu ID único. A etapa 2 (bloco chamado `f`) representa as funções responsáveis por identificar o tipo de elemento – como sensor ou atuador – e redirecionar para o fluxo correto na próxima etapa. A etapa 3 (bloco chamado `json`) é responsável por converter os dados brutos em objetos acessíveis às funções lógicas. Portanto, a etapa 4 – chamada `f` – é uma função responsável por obter os valores e tomar decisões como enviar alertas, armazenar dados em bancos de dados, responder mensagens MQTT aos atuadores, etc. Na última etapa, número 5 (chamada `device_*_output`), é possível ver todos os tópicos associados a qualquer sensor/atuador. Esta etapa é iniciada apenas se a etapa 3 desejar se comunicar com sensores/atuadores, em outras palavras, enviando uma mensagem para um tópico MQTT específico.

Portanto, o NodeMCU[8] é responsável apenas pelo envio de dados (ID exclusivo e dados coletados) de uma porta específica para um tópico específico. Toda lógica, identificação e manipulação desses dados são tratados em um nível de aplicação – no NodeRED. Dessa forma, os sensores podem ser conectados e desconectados da plataforma NodeMCU de forma transparente, a partir da arquitetura, e o usuário é o único responsável por criar a função na lógica de programação de fluxo, tratando os dados obtidos pelos sensores.

O fluxo de validação da plataforma é demonstrado na figura 6, onde há os passos dede o início (onde o usuário interage com o ambiente a partir da GUI), até a troca de mensagens por MQTT e a comunicação da

[8] A versão 3 com módulo Wi-Fi ESP8266 foi utilizado.

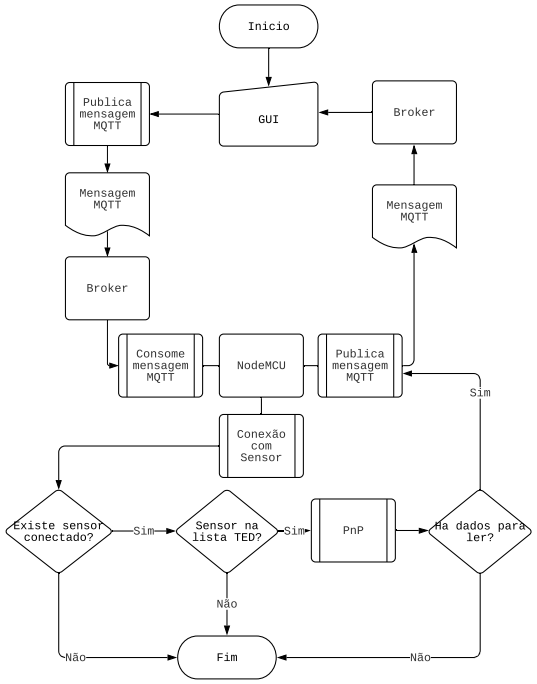

Fig. 6: Fluxo de validação da leitura de um sensor PnP pela plataforma.

plataforma para a leitura de um sensor PnP, caso ele esteja conectado à plataforma principal. Lembrando que: o NodeMCU é nossa plataforma principal de desenvolvimento, e partir dela há os processos de recebimento e envio de mensagens MQTT à GUI, e também o processo de conexão com o sensor é iniciado. A leitura do sensor só é possível caso exista um sensor conectado e o leitor esteja na TED (lista de sensores suportados pela plataforma), a partir daí é realizado o PnP através do microcontrolador acoplado ao sensor. Caso não exista sensor, ou ele não seja reconhecido na TED, ou ainda não haja dados a serem lidos, o processo simplficado demonstrado pelo fluxo pode ser finalizado.

A figura 7 mostra a plataforma de desenvolvimento e validação da solução com suas conexões físicas em uma placa de prototipação – uma forma mais simples e clara para demonstrar experimentos realizados com a solução.

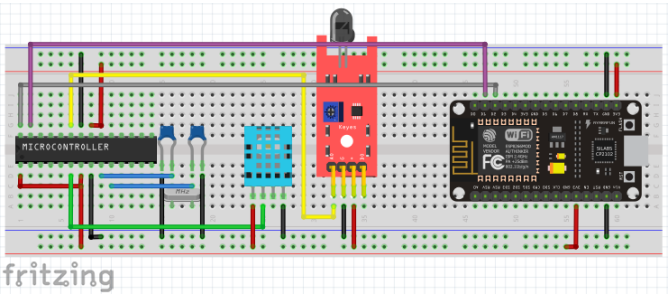

Fig. 7: Plataforma de desenvolvimento e validação com os componentes e suas conexões.

Essa mesma figura representa todas as conexões físicas em detalhes. Nela é possível ver a conexão entre os sensores utilizados no experimento, o microcontrolador e a plataforma de desenvolvimento NodeMCU. O microcontrolador é responsável por identificar, de forma PnP, os sensores conectados a ela, repassando ao NodeMCU somente os dados. Esses dados, por sua vez, serão formatados em uma mensagem JSON para envio ao NodeRED, conforme o exemplo da listagem 1.

```
{"node": {
  "id–node": "01",
  "sensor": {
    "id–sensor": "453",
    "data": "1";
}}}
```

Listagem 1: Formato JSON dos dados dos sensores/atuadores encaminhados pela plataforma.

A função lógica criada pelo usuário na GUI foi a de alertar ao usuário se a variável `flame` é `true`, se a `temperature` é maior que $40\,^{\circ}\mathrm{C}$ e se `humidity` é menor que `30%`. Mostrando a função lógica criada pelo usuário na GUI, segue o Algoritmo 1.

Neste exemplo, a lógica principal é obter os dados do sensor de chama/calor (variável `flame`), verificar se é verdade e, se for, um alerta é enviado ao usuário. Outra etapa desse processo é a verificação da possibilidade de aumento do incêndio. Isso é feito verificando as variáveis `temperature` e `humidity` pelo sensor DHT11. O algoritmo ilustra um experimento simples, porém relevante, mostrando como os usuários podem criar funções e incorporá-las no bloco `f` (etapa 4 da figura 5).

Algoritmo 1: pseudo-código para o experimento.

```
   Resultado: Alert; Max.
1  enquanto Alert ≠ true faça
2      se flame ≠ false então
3          Alert ⟵ true;
4          se temperature > 40 °C e humidity < 30% então
5              Max ⟵ true;
6          senão
7              Max ⟵ false;
8          fim
9      fim
10 fim
```

Para essa validação, uma plataforma foi conectada a um *broker* MQTT via Wi-Fi, e um sensor foi conectado e desconectado para validar a funcionalidade PnP. A *interface* do usuário foi capaz de coletar os dados e processá-los, resultando no êxito da validação. Outro caso de uso foi realizado com mais sensores e atuadores – 3 sensores e 1 atuador – sendo conectados e desconectados, validando novamente nossas expectativas em relação aos recursos de PnP e o fluxo da aplicação. Após realizados os testes, foi possível constatar que, para seu pleno funcionamento, o projeto utiliza apenas 12% do espaço de armazenamento para programas do microcontrolador, além disso, foi constatado também que as variáveis globais usam cerca de 18% da memória dinâmica do microcontrolador.

Não diretamente relacionado, mas relevante aos resultados: também foram realizados testes de alcance sobre a conexão do cabo serial para comunicação entre o (S)TIM e o NCAP. Foi concluído que sensores e atuadores podem ser usados em até 2 (dois) metros de distância da placa principal, permitindo o desenvolvimento de soluções para vários dispositivos que podem cobrir todo um espaço interno, como uma sala, ou até mesmo um pequeno espaço externo, como um poste de luz. A figura 8 mostra o resultado do teste de bancada realizado para relação distância e voltagem na conexão dos sensores com a placa principal.

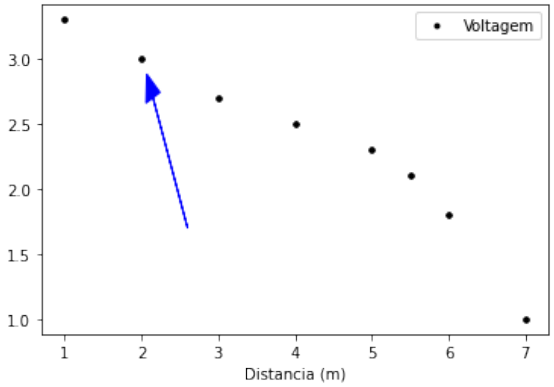

Fig. 8: Relação voltagem x distância para o cabo UTP Cat5e utilizado para conexão com os sensores.

V. CONCLUSÕES

Neste trabalho, uma nova solução para o ecossistema de IoT baseada na arquitetura *Plug-and-Play* foi proposta e validada, preenchendo as lacunas do padrão IEEE 1451 com protocolos atuais. A principal contribuição é a facilidade de comunicação e configuração dos sensores e atuadores envolvidos. Para demonstrar essas vantagens, um caso de uso para um cenário específico foi demonstrado. Assim, como o trabalho esclarece, o usuário pode alterar qualquer sensor ou atuador de forma simples, para criar novos cenários e aplicações de acordo com suas necessidades.

O paradigma PnP traz simplicidade no uso e integração de novos dispositivos à rede de IoT. Assim, a solução estimula novas maneiras de resolver diversos problemas específicos da vida real. Em trabalhos futuros, a escalabilidade de nossa proposta e a experiência do usuário serão medidas. Além disso, o consumo de energia da plataforma deve ser avaliado e serão investigadas formas de implementação de técnicas para

stand by (ou modo de espera), visando a economia de energia. Outra investigação será a avaliação da necessidade de utilizar uma zeroconf[9], a fim de possibilitar a criação de forma automática de configuração da rede IP, sem a necessidade de configuração ou servidores específicos.

Agradecimentos

Os autores agradecem à FAPEMIG, FAPERJ, CNPq, CAPES, UFJF e PROPPi-UFF pelo auxílio financeiro parcial para o desenvolvimento desta pesquisa.

Referências

[1] P. Ray, "A survey on internet of things architectures," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 3, pp. 291 – 319, 2018.

[2] A. Kumar, V. Srivastava, M. K. Singh, and G. P. Hancke, "Current status of the IEEE 1451 standard-based sensor applications," *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2505–2513, 2015.

[3] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C. Lin, "Edge of things: The big picture on the integration of edge, iot and the cloud in a distributed computing environment," *IEEE Access*, vol. 6, pp. 1706–1717, 2018.

[4] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, April 2018.

[5] B. Kang, D. Kim, and H. Choo, "Internet of everything: A large-scale autonomic iot gateway," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 3, no. 3, pp. 206–214, July 2017.

[6] A. R. Nieves, N. M. Madrid, R. Seepold, J. M. Larrauri, and B. A. Larrinaga, "A upnp service to control and manage ieee 1451 transducers in control networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 3, pp. 791–800, 2011.

[7] N. Matthys, F. Yang, W. Daniels, W. Joosen, and D. Hughes, "Demonstration of micropnp: The zero-configuration wireless sensing and actuation platform," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, June 2016, pp. 1–2.

[8] E. Martínez, D. M. Toma, S. Jirka, and J. Del Río, "Middleware for plug and play integration of heterogeneous sensor resources into the sensor web," *Sensors*, vol. 17, no. 12, p. 2923, 2017.

[9] W. Kim, H. Ko, H. Yun, J. Sung, S. Kim, and J. Nam, "A generic internet of things (iot) platform supporting plug-and-play device management based on the semantic web," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–11, 2019.

[10] W.-Y. Yi, K.-S. Leung, and Y. Leung, "A modular plug-and-play sensor system for urban air pollution monitoring: design, implementation and evaluation," *Sensors*, vol. 18, no. 1, p. 7, 2018.

[11] L. M. Engelstad, D. W. Arnston, J. H. Rud, C. E. Holmstadt, R. K. Paschke, S. V. Asmolov, and Y. M. Kuznetsov, "Sensor/transmitter plug-and-play for process instrumentation," Feb. 2 2017, uS Patent App. 15/290,757.

[12] S. Liu, J. A. Guzzo, L. Zhang, D. W. Smith, J. Lazos, and M. Grossner, "Plug-and-play sensor platform for legacy industrial machine monitoring," in *ISFA*. IEEE, 2016, pp. 432–435.

[13] A. Dutta-Roy, "Networks for homes," *IEEE spectrum*, vol. 36, no. 12, pp. 26–33, 1999.

[14] S. Quincozes, T. Emilio, and J. Kazienko, "Mqtt protocol: Fundamentals, tools and future directions," *IEEE Latin America Transactions*, vol. 17, no. 09, pp. 1439–1448, 2019.

[15] J. Velez, R. Trafford, M. Pierce, B. Thomson, E. Jastrzebski, and B. Lau, "IEEE 1451-1-6: Providing common network services over mqtt," in *SAS*, March 2018, pp. 1–6.

[16] E. F. Silva, B. J. Dembogurski, A. B. Vieira, and F. H. C. Ferreira, "IEEE P21451-1-7: Providing more efficient network services over mqtt-sn," in *2019 IEEE Sensors Applications Symposium (SAS)*. IEEE, 2019, pp. 1–5.

[17] A. Banks and R. Gupta, "MQTT version 3.1.1. OASIS standard," 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[9] *zero-configuration networking.*

MELISSA SANTOS AGUIAR é graduanda em Engenharia Elétrica com Habilitação em Sistemas Eletrônicos pela Universidade Federal de Juiz de Fora (UFJF). Atualmente é bolsista do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), atuando nas áreas de Processamento Digital de Sinais, Estimação de Energia, Sistemas Embarcados em FPGA, Internet das Coisas (IoT).

FRANCISCO HENRIQUE CERDEIRA FERREIRA é bacharel em Sistemas de Informação pelo CES/JF, mestre em Ciência da Computação pela UFJF e doutorando em Informática na UNIRIO. Atualmente ocupa o cargo de diretor do Centro de Gestão do Conhecimento Organizacional na UFJF e é professor do curso de Bacharelado em Sistemas de Informação da Faculdade Metodista Granbery. Tem experiência na área de gerenciamento e medição de redes, switching, roteamento, tecnologias WAN e VoIP.

EDUARDO PAGANI JULIO possui graduação em Ciência da Computação pela UFJF (2000), Especialização em Redes de Computadores pelo CES/JF (2001), Mestrado e Doutorado em Computação pela UFF (2007 e 2016, respectivamente). Atualmente é Professor Adjunto na UFJF. Tem experiência na área de Ciência da Computação, com ênfase em Redes de Computadores, atuando principalmente nos seguintes temas: redes sem fio, segurança em redes, IoT, sistemas embarcados, redes cognitivas e 5G.

BRUNO JOSÉ DEMBOGURSKI possui graduação em Ciência da Computação pela UFJF (2006), Mestrado e Doutorado em Computação pela UFF (2009, 2014) na área de Computação Visual. Atualmente é professor adjunto do departamento de ciência da computação da UFRRJ. Tem experiência na área de Ciência da Computação, com ênfase em Processamento Geométrico, Computação Visual, Programação em GPU, Realidade Aumentada, Inteligência Computacional e Desenvolvimento de Sistemas.

GUSTAVO SILVA SEMAAN é professor da UFF no Instituto do Noroeste Fluminense de Educação Superior (INFES) desde 2014. Doutor em Computação (Algoritmos e Otimização) e Mestre em Computação (Otimização e Inteligência Artificial) pelo IC-UFF. Bacharel em Sistemas de Informação pela Faculdade Metodista Granbery. Membro do Laboratório de Inteligência Computacional (LabIC) desde 2006. Atua com análise e desenvolvimento de sistemas desde 2001 e como professor de graduação desde 2008.

EDELBERTO FRANCO SILVA obteve os títulos de M.Sc. e D.Sc. em ciência da computação pela UFF, respectivamente em 2011 e 2016. É professor do Departamento de Ciência da Computação da UFJF. Já coordenou e participou de vários projetos de pesquisa financiados por empresas e governo. É pesquisador nos laboratórios MidiaCom e NetLab, membro da SBC e IEEE, tendo como Redes de Computadores sua principal área de pesquisa.