Uma solução de IoT baseada no FIWARE para gerenciamento de recursos energéticos e serviços acadêmicos em um campus universitário

Antonio D. T. Amurim[1], **João I. M. da Silva**[1], **Marcos D. Ortiz**[1], **Paulo A. L. Rego**[2] **e José N. de Souza**[2]

[1]Universidade Federal do Ceará (UFC)
Quixadá – CE – Brazil

[2]Departamento de Computação
Universidade Federal do Ceará
Fortaleza, Ceará, Brasil

{davidamurim7,joaoigormatos123}@alu.ufc.br, {mdo,pauloalr,neuman}@ufc.br

Resumo. *A Internet das Coisas (IoT) abrange as mais diversas áreas, desde a saúde às cidades e prédios inteligentes. Em ambientes com grande fluxo de pessoas, como os campus universitários, a IoT pode ser útil na gerência dos recursos, bem como no fornecimento de novos serviços. Neste trabalho, é apresentada uma solução para controle e monitoramento do consumo de energia e serviços acadêmicos automatizados, tais como a frequência de alunos por reconhecimento facial. O middleware FIWARE foi integrado à arquitetura da solução para lidar com a comunicação heterogênea, o armazenamento e processamento dos dados de contexto. Ademais, testes foram realizados para avaliar a solução proposta, e os resultados são promissores, indicando, por exemplo, que a solução escala bem com o aumento da quantidade de dispositivos conectados.*

1. Introdução

Uma cidade inteligente é um projeto de cidade que inclui uma infraestrutura e soluções tecnológicas para os problemas dos cidadãos [Özcan et al. 2017]. Nesse contexto, um campus inteligente pode ser visto como uma iniciativa que emprega soluções através da tecnologia para os problemas enfrentados pelos alunos e funcionários de um campus universitário. Dito isso, pode-se identificar diversas complicações para o funcionamento ideal do prédio como acessibilidade, automatização de equipamentos elétricos, controle do consumo de energia e água, entre outros.

Os problemas encontrados podem ser listados dos mais simples aos mais severos. Contudo, vale ressaltar que os edifícios comerciais e residenciais representam cerca de 60% do consumo mundial de eletricidade [Rocha et al. 2019]. Nesse viés, o mau gerenciamento do consumo de energia elétrica se destaca como um dos mais urgentes, pois afeta bruscamente o planejamento econômico do campus além de contrariar as boas práticas de sustentabilidade. Ademais, isso também inclui o controle de acionamento dos equipamentos elétricos contidos nas salas, pois o funcionamento em horários indevidos resulta em desperdício de energia elétrica.

Em resposta a esses problemas, este trabalho apresenta uma solução composta de dispositivos e *middleware* IoT que fazem o controle de forma automática dos equipamentos elétricos como lâmpadas, projetores e aparelhos de ar-condicionado, evitando o

desperdício de energia com o funcionamento fora do período de aula. Além desses, foi notada a relevância do uso de dispositivos que monitoram fatores do ambiente, como a temperatura e a umidade do ar, a fim de que esses dados sirvam de parâmetros para o controle do ar-condicionado. A solução faz também o monitoramento da corrente consumida pelos equipamentos para identificar se um determinado dispositivo está em funcionamento, o consumo em potência, além possibilitar a capacidade de analisar padrões no sinal elétrico para prever quedas de energia, mau funcionamento e identificar assinaturas de equipamentos. Por fim, também é suportado o uso de câmeras, que podem ser utilizadas para detectar a presença de pessoas em um ambiente, bem como utilizar analíticos de reconhecimento facial, o que pode facilitar o controle da frequência dos alunos.

Os dispositivos inteligentes recebem comandos e enviam dados por meio do FIWARE[1], um serviço de *middleware* que controla a troca de mensagens entre os dispositivos IoT e as aplicações de gerência. Além disso, o FIWARE disponibiliza diversos GEs (*Generic Enabler*) que são usados para armazenar e processar as informações de contexto e envio de alertas.

Este trabalho está organizado da seguinte forma: na Seção 2, são apresentados os trabalhos relacionados, que foram comparados com este trabalho; na Seção 3, é apresentada a solução IoT para gerenciamento de recursos e serviços em um campus universitário; a Seção 4 apresenta resultados de alguns experimentos voltados para a avaliação da solução e, por fim, a Seção 5 apresentada as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Atualmente, podem ser encontrados diversos artigos recentes associados à implementação de dispositivos inteligentes em campi, casas e cidades, especialmente, quando envolve o uso da IoT. Nesta seção, são listados e discutidos trabalhos que apresentam projetos de construções inteligentes que visam, principalmente, a diminuição do desperdício de energia elétrica e automatização de tarefas.

Em [Rocha et al. 2019], é proposto o projeto *Smart Place*, cuja finalidade é criar um campus inteligente na Universidade Federal do Rio Grande do Norte, na cidade de Natal. O principal propósito do trabalho é controlar os aparelhos de ar-condicionado levando em conta o número de pessoas presentes no ambiente monitorado, reduzindo assim o desperdício de energia. Além disso, variáveis como temperatura e umidade são capturadas, ainda, é usado um sensor de presença e uma câmera para obter a presença e o número de pessoas no ambiente. Ademais, o projeto utiliza o microcomputador Raspberry Pi como *hardware* principal da solução proposta, além de placas microcontroladoras ESP NodeMCU como dispositivos auxiliares. Por fim, o *middleware* FIWARE é empregado para intermediar a comunicação entre os dispositivos inteligentes e a aplicação.

O trabalho [Ferreira et al. 2018] promove o desenvolvimento e a implementação de um sistema para gerenciar a energia elétrica de uma construção pública inteligente. O projeto visa reduzir o consumo de energia por meio do controle de equipamentos elétricos, como lâmpadas e aquecedores. Além disso, é usado o conceito de gamificação que objetiva conseguir mudanças no comportamento dos usuários rumo à redução do desperdício. O projeto utiliza a placa microcontroladora Arduino Uno, além do microcomputador

[1]https://www.fiware.org/

Raspberry Pi para receber dados de contexto dos sensores e enviá-los para a camada de aplicação via rede LoRaWAN, a fim de coletar informações para tomadas de decisões do controle dos equipamentos elétricos e disponibilizar o consumo de energia do edifício.

Em [França et al. 2020], é relatado o projeto SIGOc, que tem como intuito promover um campus inteligente voltado à segurança do ambiente universitário. O projeto possui uma aplicação móvel, onde é possível registrar ocorrências de crimes que ocorreram próximo ao campus universitário. O SIGOc conta também com outra aplicação onde os seguranças podem visualizar os dados registrados da ocorrência como o local, o número de criminosos, a descrição e etc. Além disso, o sistema agrega o monitoramento de variáveis das salas de aula, como temperatura e número de alunos. Por fim, o projeto também utiliza o *middleware* FIWARE e o microcomputador Raspberry Pi como *hardware* principal para obter dados dos sensores.

O trabalho [Agate et al. 2018] propõe um projeto de campus inteligente que implementa os conceitos de *Fog Computing* para compor um sistema que agrega vários serviços tecnológicos. O trabalho faz uso de sensores para capturar variáveis do ambiente, também utiliza câmeras inteligentes, além de interagir com os dispositivos móveis dos usuários como *smartphones* e *tablets* para coletar informações disponibilizadas pelos alunos. Por fim, os dados capturados são processados e disponibilizados a diferentes aplicações, como monitoramento do consumo de energia, vigilância e suporte ao aluno.

Em [Salhofer et al. 2019], é descrito um projeto de cidade inteligente no qual sensores são espalhados em pontos estratégicos da cidade, onde podem se conectar a uma fonte de energia, como postes e semáforos. Além desses, o projeto também traz a ideia de sensores móveis que são colocados em ônibus e bondes públicos. Esses sensores têm como propósito informar variáveis como temperatura, umidade e concentração de partículas finas (PM2.5 e PM10), a fim de analisar as condições de saúde da população. Os autores realizaram testes com protótipos de sensores utilizando o microcomputador Raspberry Pi. Para a comunicação e análise dos dados de contexto, foram utilizados vários GEs do *middleware* FIWARE para armazenamento e análise de dados históricos e, principalmente, na segurança da rede de comunicação.

Neste trabalho, foram desenvolvidos dispositivos de baixo custo, com componentes facilmente encontrados no mercado. Além disso, o uso de um *middleware* voltado para IoT, como o FIWARE, possibilita a integração entre dispositivos IoT heterogêneos e aplicações. A solução aqui proposta está sendo implantada no campus da UFC, na cidade de Quixadá, inicialmente como um estudo de caso. Posteriormente, a plataforma será aberta para os demais professores e alunos adicionarem seus dispositivos e aplicações com o propósito de fomentar o ecossistema de campus universitário inteligente.

A Tabela 1 apresenta um comparativo dos trabalhos mencionados. Nela, são apresentadas as principais tecnologias utilizadas, como o hardware principal juntamente com o custo, obtido em [LCSC 2021], e o modelo de comunicação. Além disso, são apresentados o objetivo principal de cada trabalho, bem como vantagens dos mesmos. Como pode ser observado na coluna tecnologia, o microcomputador Raspberry Pi apresenta um custo elevado para ser usado em escala no campus universitário. Nesse contexto, a placa ESP-01 foi escolhida para ser o núcleo de processamento principal dos dispositivos IoT projetados neste trabalho, principalmente pelo seu baixo custo e tamanho reduzido. Em

menor escala, foram utilizadas placas ESP32 integradas com câmera para reconhecimento facial. Por fim, apenas uma unidade por bloco do *Raspberry Pi* foi utilizada no campus, para oferecer serviços de *Fog Computing*.

Tabela 1. Comparação entre os trabalhos de construções inteligentes

Referência	Tecnologia	Aplicação	Vantagens
[Rocha et al. 2019]	Raspberry Pi (US$ 69.44), ESP NodeMCU (US$ 5.16) e FIWARE	Reduzir o desperdício de energia dos aparelhos de ar-condicionado	Boa comunicação entre os dispositivos com a arquitetura *master e slave*
[Ferreira et al. 2018]	Raspberry Pi (US$ 69.44), Arduino Uno (US$ 15.41) e LoRaWAN	Reduzir o consumo de energia de aparelhos elétricos	Uso de rede de longo alcance e conceitos de gameficação
[França et al. 2020]	Raspberry Pi (US$ 69.44) e FIWARE	Proporcionar mais segurança ao ambiente universitário	Uso de aplicações mobile para interação com os alunos
[Agate et al. 2018]	Computação em nuvem e *fog computing*	Promover um campus inteligente em diferentes aspectos	Interação com os alunos e uso de *fog computing*
[Salhofer et al. 2019]	Raspberry Pi (US$ 69.44) e FIWARE	Analisar condições de saúde em uma cidade inteligente	Uso de serviços de armazenamento e segurança
Este trabalho	ESP-01 (US$ 2.32), ESP-CAM (US$ 6.76) Raspberry Pi (US$ 69.44) e FIWARE	Reduzir o desperdício de energia de um campus universitário	Dispositivos de baixo custo facilmente escaláveis

3. Solução Proposta

Para atender às necessidades do campus universitário inteligente, alguns requisitos foram definidos para a solução proposta. A solução deve permitir:

- controlar lâmpadas, projetores e aparelhos de ar-condicionado;
- monitorar a corrente consumida pelos equipamentos;
- monitorar fatores do ambiente, tais como temperatura e umidade do ar; e
- monitorar o *stream* de câmeras com analíticos de vídeo.

Isto posto, esta seção apresenta a arquitetura da solução, abordando os principais componentes divididos em 3 camadas: Aplicação, *Middleware* e Dispositivos IoT. A Figura 1 apresenta uma visão geral da solução, onde os principais componentes são localizados nas três camadas mencionadas. As subseções seguintes abordam, respectivamente, (i) o *middleware* IoT FIWARE, responsável por padronizar a comunicação, armazenamento e processamento dos dados de contexto; (ii) os dispositivos IoT que foram desenvolvidos para gerenciar e monitorar os equipamentos do campus inteligente e (iii) a aplicação de gerência para visualização dos dados e controle dos recursos e serviços.

3.1. FIWARE

O FIWARE é uma plataforma de código aberto que se destaca dentre outros *middlewares* pela sua compatibilidade com os principais protocolos utilizados em ecossistemas IoT [Oliveira et al. 2018]. O objetivo do FIWARE, iniciado em 2011, é fornecer uma plataforma padrão para o desenvolvimento de aplicações inteligentes [Detzner and Salhofer 2020].

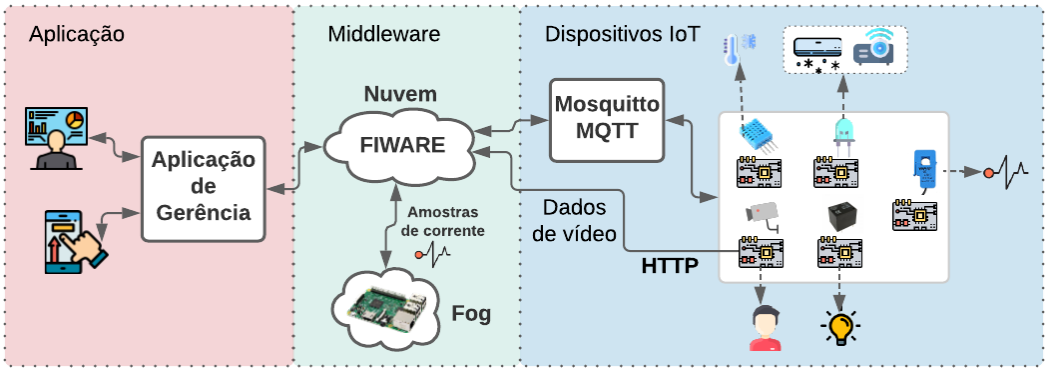


Figura 1. Visão geral da arquitetura da solução

O FIWARE dispõe de vários componentes, denominados GEs (do inglês, *Generic Enabler*), que implementam funções essenciais para atender os requisitos da solução proposta. Dessa forma, o FIWARE foi escolhido para gerenciar a troca de mensagens entre os dispositivos IoT e a aplicação de gerência. Como pode ser visto na Figura 1, o *middleware* é executado em uma arquitetura hierárquica típica de uma solução IoT, onde alguns componentes são executados na Nuvem e outros na *Fog*.

Na Nuvem, estão os serviços de armazenamento e processamento dos dados de contexto capturados pelos sensores e câmeras. Na *Fog*, é executada, especificamente, a análise das amostras capturadas pelos sensores de corrente para possibilitar, por exemplo, o monitoramento do consumo energético e qualidade do sinal, bem como a identificação das assinaturas de equipamentos ligados à rede elétrica. Uma vez processados, os dados são enviados e armazenados na Nuvem. Correlações podem ser feitas entre os dados dos sensores de corrente e temperatura, por exemplo, para identificação de quais salas consomem mais energia. A Figura 2 ilustra a arquitetura desta solução integrada aos GEs do FIWARE, bem como a interação entre os próprios GEs, que são explicados a seguir:

- **Orion:** implementa um *broker Publish/Subscribe* para execução das seguintes operações: consultar e atualizar informações de contexto, receber notificações quando mudanças no contexto ocorrem, cadastrar aplicações que produzem contexto. Ele foi utilizado para cadastrar os dispositivos IoT e a Aplicação de Gerência, permitindo a troca de informações de contexto e comandos entre eles.
- **KeyRock:** atua como Provedor de Identidade (IdP) no fornecimento de serviços de autenticação para diferentes Provedores de Serviços (SP), atuando como partes confiáveis por meio de padrões abertos, como os protocolos *OASIS SAML (Security Assertion Markup Language)* e OAuth2 [Celesti et al. 2019].
- **Wilma:** implementa um proxy PEP *(Policy Enforcement Point)* que combinado ao *Keyrock* controla o acesso dos usuários aos serviços da solução.
- **Cygnus:** persiste dados de contexto em bancos de dados e utiliza o Apache Flume[2] para a criação de uma visão histórica do contexto.
- **STH-Comet:** o *Short Time Historic* (STH) é responsável por persistir e recuperar dados de contexto em formato temporal. Neste trabalho, o STH é utilizado em conjunto com o Cygnus e é consumido pelo serviço de *back-end*.

[2]https://flume.apache.org/

- **IDAS:** traduz os protocolos IoT (por exemplo, ULtralight/MQTT, LoRaWan, M2M) para o protocolo NGSI-v2 internamente usado pelos GEs do FIWARE.
- **Kurento:** é um servidor de mídia WebRTC que fornece um conjunto de APIs para simplificar o desenvolvimento de aplicativos de vídeo avançados para a web e *smartphone* [Garcia et al. 2017].
- **FogFlow:** fornece um *framework* para orquestrar automaticamente o processamento dos fluxos de dados de contexto em nós *fog* atendendo requisitos de tempo de resposta e consumo de largura de banda [Cheng et al. 2017].

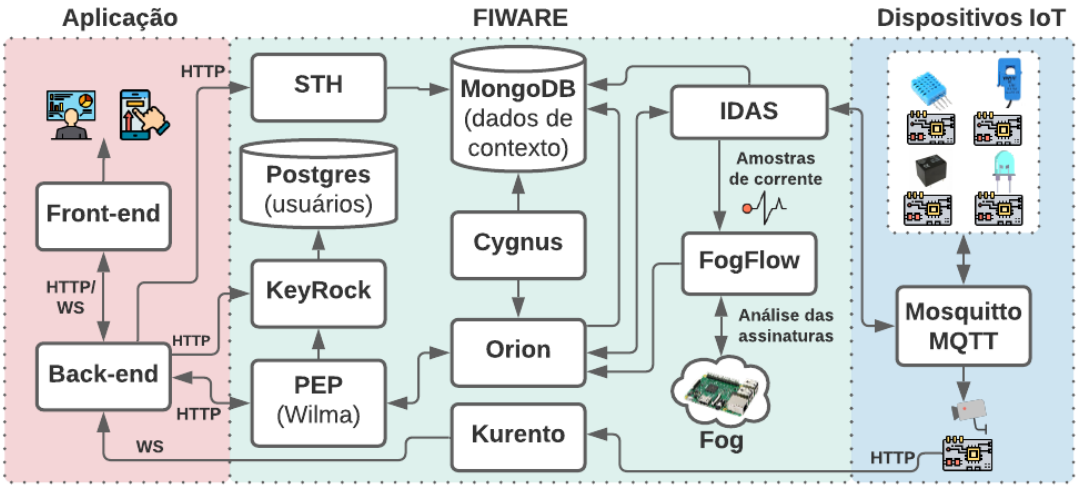


Figura 2. Arquitetura da solução estendida

3.2. Dispositivos IoT

Este trabalho utiliza a arquitetura ESP como microcontrolador principal devido ao seu baixo custo, facilidade de programação e conectividade sem fio. A Tabela 2 lista os dispositivos IoT desenvolvidos neste trabalho com seus respectivos custos (os valores foram obtidos com base em [LCSC 2021]). Estão listados também o *hardware*, protocolos de comunicação e com quais componentes do *middleware* estão associados. Para a implantação do campus inteligente, cada sala/laboratório conterá um exemplar de cada dispositivo IoT, todavia, é necessário apenas um nó de *Fog* por bloco no campus, reduzindo assim a necessidade de aquisição de várias placas *Raspberry Pi*.

Tabela 2. Comparação entre os dispositivos IoT

Dispositivo	Custo	Hardware	Protocolos	GEs
Sensor de Temperatura	US$ 3.68	ESP-01	MQTT	Orion e IDAS
Sensor de Corrente	US$ 8.48	ESP-01	MQTT	Orion, IDAS e FogFlow
Controle Infravermelho	US$ 2.69	ESP-01	MQTT	Orion e IDAS
Dispositivo Relé	US$ 6.63	ESP-01	MQTT	Orion e IDAS
Dispositivo Câmera	US$ 10.74	ESP32-CAM	MQTT e HTTP	Orion, IDAS e Kurento
Fog	US$ 69.44	Raspberry Pi 3	NGSI-v2	FogFlow

A conexão sem fio da arquitetura ESP foi utilizada para a rápida implantação da comunicação através de protocolos HTTP e MQTT, sem modificação da infraestrutura existente no campus universitário. Todavia, o protocolo MQTT se destaca para aplicações

IoT por ser mais leve e possuir um cabeçalho menor, também por consumir menos energia na transferência dos dados em comparação ao HTTP [Kodali and Soratkal 2016]. Dessa forma, o MQTT foi utilizado para a comunicação com a maioria dos dispositivos criados, enquanto o HTTP foi utilizado exclusivamente para as câmeras, que demandam *payload* maior e consomem, portanto, mais largura de banda e bateria por precisarem capturar e enviar imagens.

O MQTT é um protocolo do tipo publicação/assinatura que utiliza um *broker*. Neste trabalho, foi escolhido o *broker* Mosquitto[3], como ilustrado nas Figuras 1 e 2. Além do MQTT, utilizado para o transporte de dados, foi utilizado o protocolo Ultralight 2.0 para a representação de dados, por ser simples e de texto plano. Assim, tanto o MQTT quanto o Ultralight 2.0 foram utilizados em conjunto para a comunicação entre os dispositivos e o GE IDAS, que traduz o MQTT/Ultralight para o protocolo NGSI-v2[4], utilizado pelo FIWARE e vice-versa. Com isso, é possível a comunicação com o GE Orion, responsável por armazenar e gerenciar os dados de contexto vindos dos sensores, além de enviar comandos aos atuadores quando requisitado.

Ademais, o GE IDAS envia as amostras de corrente ao GE FogFlow para que seja feito o processamento do sinal no nó de *fog*. Para a análise do sinal de corrente, é necessário que o sensor de corrente capture e envie várias amostras do sinal para que seja feito o processamento a fim de permitir a identificação de assinaturas de equipamentos ligados à rede elétrica, anomalias e padrões de qualidade, semelhante ao que foi feito no trabalho [Barker et al. 2014]. Esse processamento requer um serviço mais dedicado para que os resultados da análise do sinal possam ser obtidos com menor tempo de resposta, além de não sobrecarregar a rede com os dados das amostras.

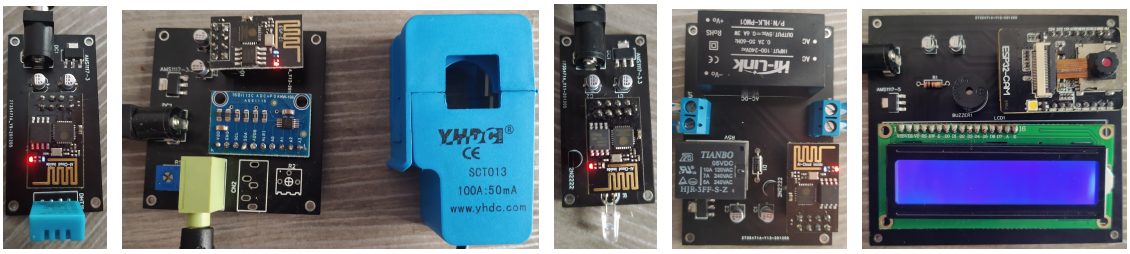
Diante da necessidade de tratar imagens provenientes das câmeras, o GE Kurento foi escolhido para receber os *streams* de vídeo, executar analíticos para extrair eventos e informações de contexto e disponibilizá-los para a camada de aplicação. Dessa forma, tal recurso foi utilizado para permitir, por exemplo, a aplicação de filtros de reconhecimento facial, identificação do uso de máscara e identificação de placas de veículos.

Inicialmente, protótipos dos circuitos foram testados em uma sala do campus. Em seguida, foram modeladas e encomendadas PCBs (Placa de Circuito Impresso) para cada dispositivo sensor e atuador. A Figura 3 ilustra os dispositivos desenvolvidos neste trabalho. Além dos microcontroladores ESP, é possível observar outros componentes importantes utilizados em cada dispositivo IoT, como os sensores e atuadores de cada placa.

Na Figura 3(a), é ilustrado o dispositivo para sensoreamento da temperatura e umidade, que tem o propósito de auxiliar o controle do ar-condicionado, mantendo o ambiente na temperatura desejada. Este dispositivo utiliza o sensor DHT11, que captura tanto a temperatura do ambiente quanto a umidade. A Figura 3(b) apresenta o dispositivo criado para monitorar a corrente consumida por equipamentos elétricos. Ele captura amostras do sinal de corrente de um fio utilizando o sensor não intrusivo SCT-013-000 e as envia ao *middleware*. Ambos dispositivos utilizam a placa ESP-01 como componente principal.

[3]https://mosquitto.org/

[4]https://fiware.github.io/specifications/ngsiv2/stable/



(a) (b) (c) (d) (e)

Figura 3. Dispositivos IoT desenvolvidos

A Figura 3(c) mostra o controle infravermelho responsável por ligar e desligar o ar-condicionado, além de aumentar e diminuir a temperatura através de comandos enviados por um LED infravermelho (IR). Com esse dispositivo, também é possível enviar comandos IR para outros equipamentos como o projetor e, assim, controlá-lo. Ademais, para o controle das lâmpadas foi desenvolvido um dispositivo contendo um relé, que recebe os comandos de ligar e desligar, exibido na Figura 3(d). Ambos dispositivos também utilizam a ESP-01 como componente principal. Por fim, também foi desenvolvido um dispositivo para fazer o reconhecimento facial dos alunos, visto na Figura 3(e), a fim de auxiliar nos registros de frequência. Este recurso pode ser utilizado ou não, mediante autorização dos alunos. O dispositivo utilizado foi o ESP32-CAM por possuir uma câmera integrada. Além disso, o dispositivo possui um *display* e um *buzzer* para sinalizar ao aluno a confirmação do reconhecimento.

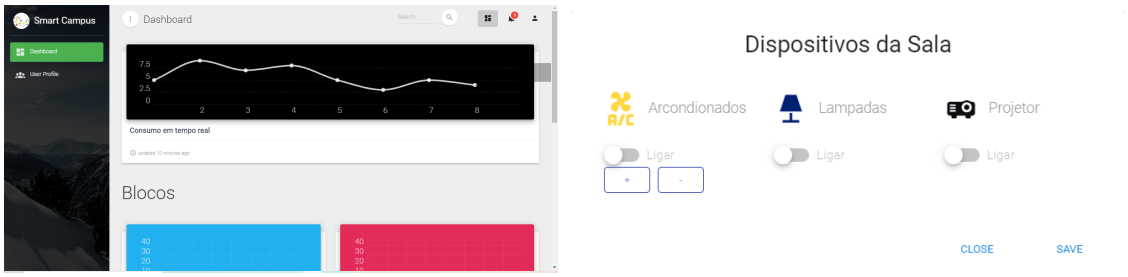
3.3. Aplicação de Gerência

Visando atender os requisitos definidos para a solução, a aplicação de gerência foi desenvolvida com foco em prover visualização dos dados de consumo energético e controle dos dispositivos IoT. Ela foi dividida em dois módulos principais: um Front-end com Dashboard responsivo (Web e *Mobile*) e um servidor *Back-end* para tratamento das requisições e interação com o FIWARE.

A aplicação Front-end consiste em um *Dashboard*, implementado usando o *framework* VueJS, que possui como funcionalidades básicas monitoramento, controle e análise do consumo de energia, controle dos ar-condicionados, lâmpadas e projetores, contagem do números de pessoas e verificação do uso de máscara. Os dados de consumo energético podem ser apresentados em nível de sala, bloco e campus universitário. A Figura 4(a) apresenta a visualização a nível de campus, onde os indicadores principais são exibidos: consumo do bloco, a quantidade de dispositivos ativos e a quantidade de salas ativas que cada bloco possui. A Figura 4(b) mostra o controle dos dispositivos de uma determinada sala, onde é possível realizar operações de ligar/desligar os aparelhos e mudar a temperatura dos ar-condicionados. O Dashboard se comunica com o *Back-end* para ter acesso aos dados através de serviços RESTful e *WebSocket* (WS).

O servidor *Back-end* foi implementado usando NodeJS e o *framework* Express. A comunicação com os demais componentes da arquitetura, como ilustrado na Figura 2, usa os protocolos HTTP e WS. Os serviços providos pelo *Back-end*, através do protocolo WS, são notificações das atualizações em tempo real do consumo de energia, temperatura

e umidade. Por outro lado, via HTTP, são providos serviços RESTful para consulta ao consumo energético de um ou mais blocos e consumo de uma ou mais salas, além do envio de comandos para ligar e desligar os aparelhos de ar-condicionado, lâmpadas e projetores. Ainda via HTTP, são providos serviços para mudar a temperatura dos ar-condicionados.



(a) Dashboard a nível de controle do campus (b) Dashboard a nível de controle da sala

Figura 4. Algumas telas do *Front-end* da aplicação de gerência

Para a funcionalidade de frequência automatizada, o *Back-end* recebe do Kurento, via WS, os dados do aluno reconhecido. Em seguida, ele envia os comandos de confirmação de reconhecimento para a placa da câmera, que os apresenta ao aluno por meio do *display* e do buzzer. Para acessar os serviços RESTful e de notificação providos pelo *Back-end*, o cliente deve estar cadastrado e autenticado no GE KeyRock, uma vez que todo o fluxo de autorização e autenticação ocorre utilizando o protocolo OAuth2.

4. Avaliação

A avaliação da solução proposta analisou o consumo de recursos de *hardware* (CPU e memória) de alguns componentes da arquitetura, bem como o tempo decorrido para efetivação das operações de sensoriamento e autuação em um ambiente de campus universitário inteligente.

Essas métricas foram escolhidas para análise da carga de trabalho que a solução proposta suporta, sem sofrer esgotamento dos recursos de hardware e rede [Silva et al. 2020]. Para os experimentos, foram utilizadas duas ferramentas: o SenSE - *Sensor Simulation Environment* [Zyrianoff et al. 2017] e o IMAIoT [Heideker et al. 2019]. O SenSE foi utilizado para gerar o tráfego de sensores IoT distribuídos em um campus universitário, enquanto o IMAIoT retorna o consumo de CPU e memória dos principais componentes da solução. A Figura 5 ilustra o cenário dos testes realizados com o SenSE e o IMAIoT no campus da UFC na cidade de Quixadá-CE.

O SenSE foi configurado para enviar dados via protocolo MQTT, no entanto, foram feitas alterações no seu código-fonte para suportar o envio de dados representados em *Ultralight* 2.0. Desta forma, foi possível analisar o tempo decorrido desde a criação da mensagem pelo nó sensor, o envio e processamento via FIWARE, até o recebimento pela camada de aplicação. Os experimentos variaram o número de nós sensores e a frequência de envio. A Tabela 3 apresenta um resumo das configurações dos experimentos.

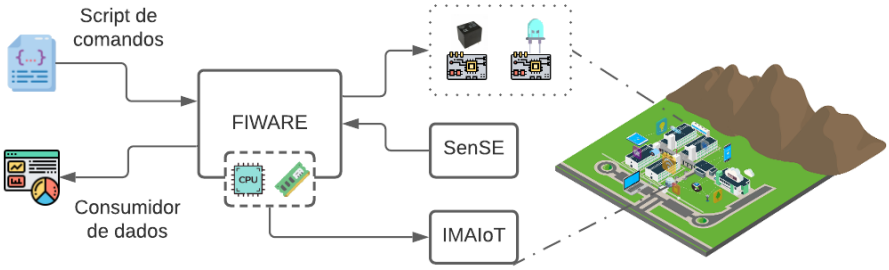


Figura 5. Arquitetura dos testes com o SenSE

Para o cenário do campus da UFC em Quixadá, estimou-se que cada um dos quatro blocos do campus demandaria um total de 50 sensores. Dessa forma, foi avaliado o desempenho da arquitetura com a ativação de cada bloco. Os dados foram enviados pelos sensores com frequências que variam a cada 1, 3 e 5 segundos. Durante o envio de tráfego pelo SenSE, foram enviados também comandos para os atuadores reais que estão em uma sala do campus, a fim de termos resultados também no fluxo inverso (i.e., da aplicação para os dispositivos IoT).

O IMAIoT foi instalado na máquina virtual que hospeda o *middleware* e a aplicação. Dado que cada GE e componentes da aplicação executam em *containers* Docker diferentes, foi possível capturar informações de uso de CPU e memória de cada *container* individualmente. Desta forma, em cada teste, foram armazenadas essas métricas para se obter médias e intervalos de confiança de cada combinação de configuração de teste, no que diz respeito ao número de sensores e periodicidade.

Tabela 3. Configurações dos experimentos

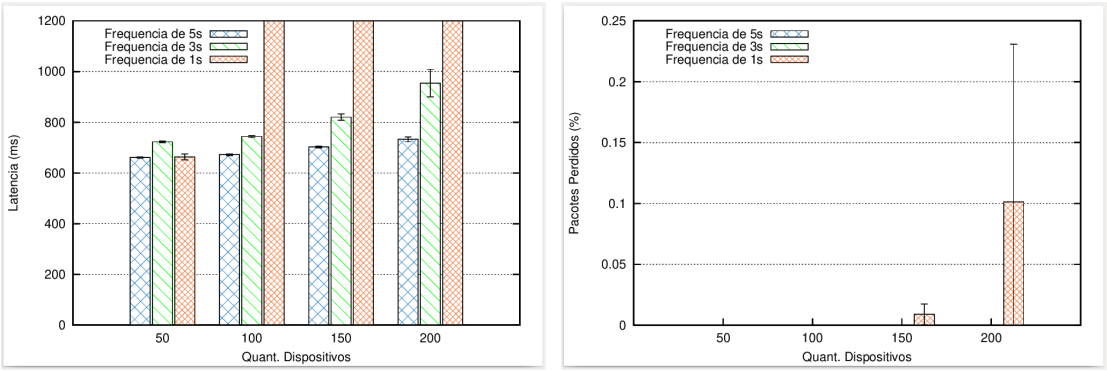
Métricas	Tempo decorrido, Descarte, Consumo de CPU/MEM
Número de nós	50, 100, 150 e 200
Frequência de envio	1s, 3s e 5s
Número de rodadas	10
Duração/rodada	2 minutos
Intervalo de Confiança	95%
VMs Back-end e FIWARAE	Ubuntu 14.04/64bits, 2.4GHz, 2GB RAM, 40GB disco
PC SenSE	Windows 8/64bits, 3.2GHz, 8GB RAM
GE - Containers	Docker

4.1. Resultados - Latência e descarte de pacotes

Na Figura 6(a), é apresentado um gráfico de latência (eixo y) com variação do número de dispositivos (eixo x) e da frequência com que eles enviam as mensagens, enquanto o gráfico da Figura 6(b) apresenta a taxa de descarte de pacotes. Como pode ser observado, com uma quantidade de 50 sensores, a frequência de envio não tem grande influência sobre a latência. Para 100 sensores, as frequências de 3 e 5 segundos causaram apenas um leve aumento na latência. No entanto, ao utilizar a frequência de envio de 1 segundo, o FIWARE se tornou um gargalo, degradando rapidamente o tempo de resposta e passou a descartar pacotes a partir de 150 sensores.

O gargalo na frequência de envio de 1 segundo se intensificou com o aumento no número de sensores para 150 e 200 unidades, onde a latência atingiu médias de aproximadamente 16,5 e 29,5 segundos, respectivamente. Foi observado um descarte de aproximadamente 7 e 97 pacotes, respectivamente, cerca de 0.1% do total de pacotes enviados.

No entanto, para as frequências de 3 e 5 segundos, ocorreu apenas um leve aumento na latência sem perdas de pacotes. Portanto, para o cenário de campus universitário avaliado, a frequência de envio entre 3 e 5 segundos é a recomendada, pois não provoca degradação excessiva na latência, nem perdas de pacotes, e ainda permite o acompanhamento do estado dos sensores em tempo real. Caso a solução deseja adotada em diversos campi da UFC, uma infraestrutura mais robusta precisa ser alocada na Nuvem e avaliada.

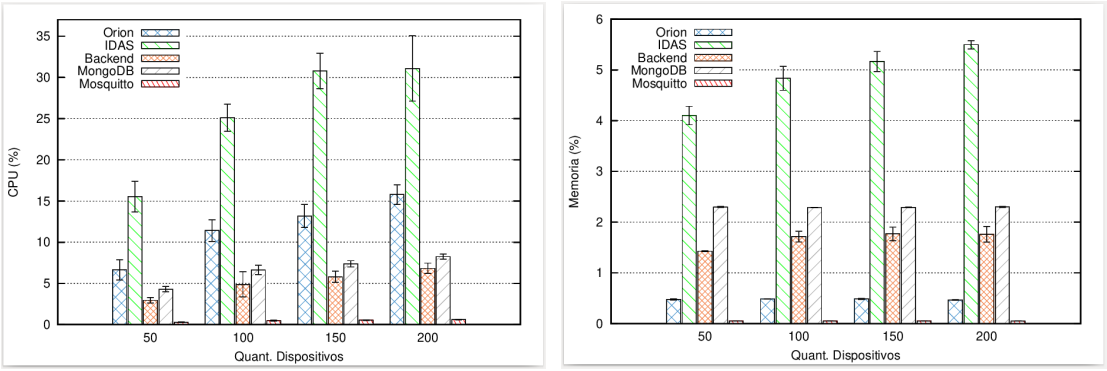


(a) Latência (b) Descarte de Pacotes

Figura 6. Envio do Estado do Sensor

4.2. Resultados - Consumo de CPU e Memória

A Figura 7(a) apresenta o consumo de CPU dos principais componentes da arquitetura, enquanto a Figura 7(b) apresenta o consumo de memória. Ambos os gráficos foram gerados para a frequência de envio de 5 segundos, definida como suficiente para acompanhar as mudanças de estado dos sensores em um cenário de campus universitário inteligente. Como esperado, o consumo de CPU e Memória aumenta à medida que mais dispositivos são ativados. Entretanto, o uso dos recursos ficou aquém da capacidade alocada na Nuvem. Logo, pode-se concluir que a frequência de envio de 5 segundos apresentou um bom desempenho também no consumo de recursos de *hardware*.



(a) Consumo de CPU (b) Consumo de Memória

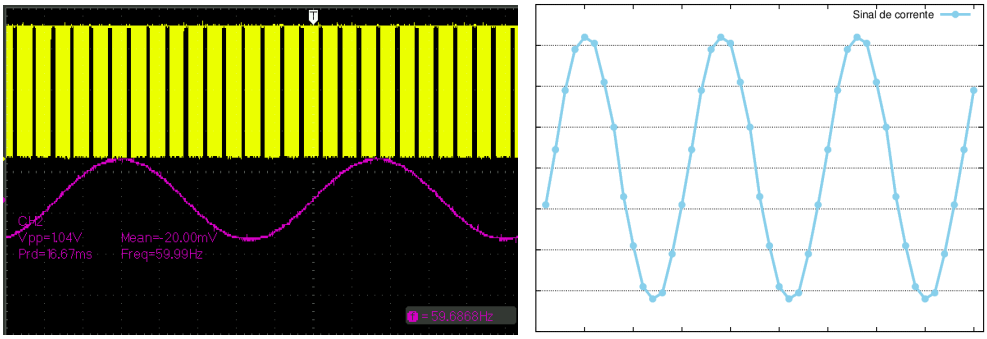
Figura 7. Taxa de uso dos recursos do *hardware* com frequência de 5s

4.3. Resultados - Eficácia dos Dispositivos IoT

Como mencionado anteriormente, os cinco dispositivos desenvolvidos neste trabalho estão implantados em uma sala do Campus da UFC em Quixadá. O FIWARE e o servidor de aplicação estão em execução em máquinas virtuais na nuvem do Campus. Dessa forma, foi possível realizar testes reais da arquitetura proposta. Neste contexto, foram enviados comandos de controle do ar-condicionado e das luzes a partir da camada de aplicação e observou-se que essas ações eram feitas corretamente.

A leitura dos dados de corrente foi verificada com um multímetro e um osciloscópio, como ilustra a Figura 8(a), onde são mostradas, na cor amarela, as várias capturas feitas pelo sensor no sinal de corrente ilustrado na cor rosa. Já a Figura 8(b) exibe a onda formada com as amostras capturadas pelo sensor, onde foram obtidas 14 amostras por período do sinal de corrente, que é semelhante ao obtido em [Pacheco[1] et al. 2016], que capturou 15 amostras por período, mas com o uso do microcomputador BeagleBone Black (que é um dispositivo mais caro do que os utilizados neste trabalho).

A leitura dos sensores de temperatura e umidade correspondeu com as variações de temperatura à medida que o ar-condicionado era ligado ou desligado. Por fim, foi testado o envio de capturas da câmera via protocolo HTTP, a recepção de comandos de *feedback* por MQTT (texto no *display* e códigos para emissão de sons), bem como a aplicação dos filtros de reconhecimento facial e identificação de máscaras, que por limitações de espaço não são apresentadas aqui.



(a) Sinal capturado pelo osciloscópio (b) Sinal gerado pelo sensor

Figura 8. Teste de captura de amostras de um sinal

Em comparação com [Rocha et al. 2019] e [Ferreira et al. 2018], que usam o microcomputador *Raspberry Pi* como componente principal, este trabalho adota, de forma mais ampla, placas microcontroladoras de baixo custo, tornando a solução menos onerosa, além do menor consumo de energia/bateria e redução de tamanho que facilita sua implantação. Apenas em nível de *Fog* (uma por bloco) são utilizados microcomputadores *Raspberry Pi*. Ademais, em relação ao trabalho [Ferreira et al. 2018], este trabalho utiliza conversores analógicos/digitais mais precisos para a aquisição de dados de corrente, tornando o monitoramento de consumo mais confiável.

5. Considerações Finais

O presente trabalho apresentou uma solução IoT para um campus universitário, cujo foco principal é controlar equipamentos elétricos, como aparelhos de ar-condicionado,

lâmpadas e projetores, visando a redução do consumo de energia da sala de aula. Esse controle é feito por meio de dispositivos inteligentes, desenvolvidos neste trabalho, que se comunicam com a aplicação de gerência por intermédio do *middleware* FIWARE. Também foram desenvolvidos dispositivos de monitoramento de temperatura, umidade, corrente elétrica e imagens de câmeras, que podem ser utilizados não só para reduzir o consumo de energia, mas também para melhorar alguns serviços acadêmicos em um campus universitário. Como exemplos de uso, podemos citar o registro de frequência dos alunos e a identificação do uso de máscara, visando o retorno das atividades presenciais em tempo de pandemia de COVID-19.

Outrossim, a solução foi avaliada por meio de testes de carga que mediram latência e taxa de descarte de pacotes, bem como consumo de CPU e memória dos principais componentes utilizados. Os resultados são promissores e indicam que a solução proposta escala bem com o aumento da quantidade de dispositivos conectados. É importante ressaltar que este trabalho visa contribuir para a formação de um ecossistema de campi inteligente, em que pesquisadores e alunos possam participar e criar soluções inovadores. Portanto, todos os artefatos[5] gerados no desenvolvimento deste trabalho estão disponíveis para a comunidade: a modelagem das PCBs e códigos-fonte dos dispositivos e componentes da aplicação de gerência.

Como trabalhos futuros, serão adicionados novos dispositivos IoT para rastrear o ônibus universitário, monitorar o consumo de água e controlar o acesso com fechaduras inteligentes. Outros serviços do FIWARE também serão incorporados, como o de análise de *Big Data* por meio do GE Cosmos, integrado ao Apache Flink/Spark, além do processamento de eventos complexos e geração de alertas com o uso do GE Perseo. Por fim, será implantado o modelo de federação para habilitar cenários IoT descentralizados com a integração dos demais campi da UFC à visão da aplicação de gerência.

Agradecimentos

Os autores agradecem à FUNCAP (processo no. 6945087/2019) e à FAPESP (processo no. 2015/24144-7) pelo suporte financeiro.

Referências

Agate, V., Concone, F., and Ferraro, P. (2018). Wip: Smart services for an augmented campus. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 276–278. IEEE.

Barker, S., Musthag, M., Irwin, D., and Shenoy, P. (2014). Non-intrusive load identification for smart outlets. In *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 548–553. IEEE.

Celesti, A., Fazio, M., Galán Márquez, F., Glikson, A., Mauwa, H., Bagula, A., Celesti, F., and Villari, M. (2019). How to develop iot cloud e-health systems based on fiware: a lesson learnt. *Journal of Sensor and Actuator Networks*, 8(1):7.

Cheng, B., Solmaz, G., Cirillo, F., Kovacs, E., Terasawa, K., and Kitazawa, A. (2017). Fogflow: Easy programming of iot services over cloud and edges for smart cities. *IEEE Internet of Things Journal*, 5(2):696–707.

[5]https://github.com/iotsmartcampus/Smart-Campus

Detzner, P. and Salhofer, P. (2020). Analysing fiwares platform-potential improvements. In *53rd Hawaii International Conference on System Sciences - HICSS*.

Ferreira, J. C., Afonso, J. A., Monteiro, V., and Afonso, J. L. (2018). An energy management platform for public buildings. *Electronics*, 7(11):294.

França, Í., Araújo, N., Gomes, A., Cacho, N., Lopes, F., Lima, J., and Adachi, E. (2020). Sigoc: A smart campus platform to improve public safety. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6. IEEE.

Garcia, B., Lopez-Fernandez, L., Gallego, M., and Gortazar, F. (2017). Kurento: the swiss army knife of webrtc media servers. *IEEE Communications Standards Magazine*, 1(2):44–51.

Heideker, A., Silva, D. O., Zyrianoff, I., Kleinschmidt, J. H., and Kamienski, C. A. (2019). Imaiot infrastructure monitoring agent for iot: Um agente monitor de infraestruturas para ambientes de iot. In *Anais Estendidos do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 9–16. SBC.

Kodali, R. K. and Soratkal, S. (2016). Mqtt based home automation system using esp8266. In *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, pages 1–5. IEEE.

LCSC (2021). Lcsc eletronics. `https://lcsc.com/`. Acessado: 19 Mar. 2021.

Oliveira, C. T., Moreira, R., de Oliveira Silva, F., Miani, R. S., and Rosa, P. F. (2018). Improving security on iot applications based on the fiware platform. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pages 686–693. IEEE.

Özcan, U., Arslan, A., İlkyaz, M., and Karaarslan, E. (2017). An augmented reality application for smart campus urbanization: Msku campus prototype. In *2017 5th International Istanbul Smart Grid and Cities Congress and Fair (ICSG)*, pages 100–104.

Pacheco[1], Á. C., de Souza, A. A., Martins, B. D., Neves, D. P., Silva, M. F., and de Paula Silva, S. F. (2016). Projeto de um sistema de medição, monitoramento e acionamento remoto de uma carga elétrica.

Rocha, F., Dantas, L. C., Santos, L. F., Ferreira, S., Soares, B., Fernandes, A., Cavalcante, E., and Batista, T. (2019). Energy efficiency in smart buildings: An iot-based air conditioning control system. In *IFIP International Internet of Things Conference*, pages 21–35. Springer.

Salhofer, P., Buchsbaum, J., and Janusch, M. (2019). Building a fiware smart city platform. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*.

Silva, D. O., Zyrianoff, I. D., Heideker, A. H., Kleinschmidt, J. H., and Kamienski, C. A. (2020). Desempenho e escalabilidade de plataformas livres de iot. In *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC.

Zyrianoff, I., Borelli, F., and Kamienski, C. (2017). Sense–sensor simulation environment: Uma ferramenta para geração de tráfego iot em larga escala. *Salão de Ferramentas-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos–SBRC*.