SMDCIC - Sistema para Monitoramento de Data Centers baseado na Internet das Coisas

Carlos Alício Andrade de Carvalho
Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Bahia, Brasil
carlosalicio.developer@gmail.com

Antônio Carlos dos Santos Souza
Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Bahia, Brasil
antoniocarlos@ifba.edu.br

RESUMO

A *Internet of Things - IoT* ou Internet das Coisas está mudando a forma com que o mundo se conecta e troca informações. Ao passo que essa transformação acontece, novas tecnologias e processos surgem, abrindo um leque de possibilidades para o seu uso. Uma aplicação possível é no ambiente de *Data Centers*, componente essencial para o desenvolvimento da computação como um todo e principalmente a área de IoT. O presente trabalho descreve os passos executados para a construção de uma solução para o monitoramento do ambiente de *data centers* valendo-se dos conceitos e tecnologias desenvolvidas nos últimos anos com base na internet das coisas.

1. INTRODUÇÃO

Com o passar do tempo, a computação e suas subáreas evoluem e a cada dia mais dispositivos estão conectados, mais serviços estão sendo disponibilizados e também mais pessoas passam a produzir e consumir informação. Nesse cenário, uma peça fundamental são os *Data Centers*, pois viabilizam o armazenamento, processamento e fluxo de dados e informação pela rede.

Um *Data Center* pode ser definido como um ambiente no qual estão equipamentos que armazenam informações essenciais para o negócio de uma organização[16]. Assim, o monitoramento de *data centers* é um fator importantíssimo a ser considerado pois a disponibilidade dos dados e da informação é fundamental para o negócio no qual estão inseridos.

No que se diz respeito ao monitoramento do ambiente de *data centers*, aspectos devem ser considerados, pois uma simples mudança de temperatura pode acarretar em danos aos equipamentos, levando a interrupções de disponibilidade do sistema, o que pode resultar em perdas para a organização.

Fatores como o alto custo e a complexidade de implantação de um sistema para monitorar o ambiente em *data centers* despertou a necessidade de se experimentar novas tecnologias a fim de solucionar essa questão.

Uma outra área da computação que vem crescendo e ganhado bastante destaque é a de IoT (*Internet of Things* ou Internet das Coisas). Com o desenvolvimento da IoT, nos últimos anos, surgiram várias tecnologias que possibilitaram a conexão e comunicação dos mais diversos tipos de dispositivos na rede (internet). Isso expandiu a gama de possíveis aplicações desenvolvidas, principalmente nas áreas de automação, monitoramento e utilização de sensores.

Tendo como base o cenário descrito acima, o decorrente trabalho descreve os passos executados para a construção da solução batizada como SMDCIC - Sistema para Monitoramento de Data Centers baseado na Internet das Coisas. O principal objetivo do SMDCIC é o uso dos novos conceitos e tecnologias da área de IoT para desenvolver uma solução que permita, de forma prática, a implantação de um sistema de monitoramento e acompanhamento do ambiente de um *data center*, atentando-se a aspectos como baixo custo, fácil escalonamento, simples implantação e acompanhamento remoto pelo usuário.

O trabalho está organizado em seções onde logo após esta introdução a Seção 2 apresenta o Referencial Teórico, posteriormente a Seção 3 apresenta a Solução Desenvolvida (SMDCIC), a Seção 4 mostra os Resultados Obtidos e, por fim, a seções 5 contém as Conclusões.

2. REFERENCIAL BIBLIOGRÁFICO

Essa seção apresenta as principais referências usadas para compreensão do problema e desenvolvimento da solução proposta. Ela está organizada da seguinte maneira: A subseção 2.1 apresenta a importância do monitoramento de *data centers*, já a subseção 2.2 traz um breve levantamento e comparação das principais plataformas para automação e por fim na subseção 2.3 são apresentados os conceitos relativos a IoT.

2.1 Monitoramento de Data Center

O tempo de indisponibilidade (*downtime* - tempo que uma aplicação ou sistema permanece indisponível para utilização) é um dos aspectos a ser levado em consideração para a qualificação de um *data center*, isso de acordo com a norma ANSI/TIA 942 (*Telecommunications Infrastructure Standard for Data Center* - Infraestrutura de Telecomunicações para *data center*[1]) que utiliza a classificação por *Tiers* desenvolvido pelo *The Uptime Institute*[25].

É nesse cenário que começam a surgir as soluções para o melhor gerenciamento dos *data centers* e sua respectivas infraestruturas.

2.1.1 Data Center Infrastructure Management

Uma abordagem que surgiu nos últimos anos é a gestão da infraestrutura de *data center* (DCIM, *Data Center Infrastructure Management*) que, de acordo com a *Data Center Knowledge*, pode ser caracterizada como um conjunto de

soluções que integram as funções tradicionais de gestão de *data centers* com a gestão dos ativos físicos e recursos da infraestrutura predial, envolvendo *softwares* especializados, *hardware* e sensores[25].

Um dos modelos mais usados para projetos DCIM é o elaborado pela Gartner, empresa de pesquisa e consultoria de tecnologia da informação[9]. Nela os vários sensores e entradas do usuário (*Other system feeds*) do sistema compreendem a entrada (*Input*) enviando dados, os quais são processados (*process*), analisados e apresentados como saída (*Output*) para o usuário em forma de um painel ou gráfico[12]. Na figura 1 segue uma imagem do modelo.

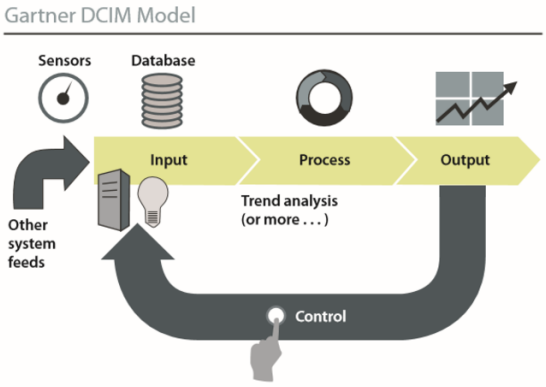

Figure 1: Modelo de arquitetura DCIM[9].

2.1.2 *Sistema Supervisório de Aquisição de Dados*

Uma solução mais tradicional, a qual já vem sendo aplicada para monitoramento e controle na indústria, é a utilização de um Sistema Supervisório de Aquisição de Dados (SCADA - *Supervisory Control and Data Aquisition*).

Para DANEELS & SALTER (1997 apud NARDI, 2007) SCADA são *softwares* que controlam e distribuem informações entre estações e também servem como interfaces homem/máquina[1]. Ao abordar o tema, Boyer (2004 apud NARDI, 2007) conceitua SCADA como um conjunto de tecnologias que permitem ao usuário coletar dados e enviar instruções de controle a uma ou mais instalações[11]. A figura 2 exemplifica o modelo da arquitetura.

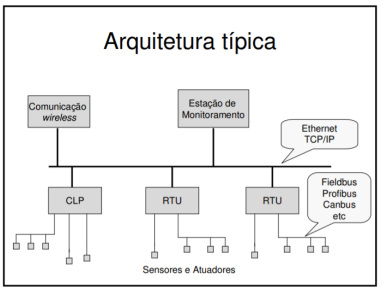

Figure 2: Modelo de arquitetura Típica SCADA[21].

Por possuir características com gerenciamento de alarmes, geração de relatórios, comunicação com sistemas externos, em alguns casos acesso via *web* e dispositivos *Mobile*[5], entre outras, as soluções SCADA também passaram a ser adotadas no monitoramento de *data centers*.

2.1.3 *Web-Based Enterprise Management - WBEM*

Para o monitoramento de aspectos lógicos, como por exemplo a rede do data center, também vem sendo empregado o padrão de gerenciamento baseado na tecnologia web WBEM *(Web-Based Enterprise Management)*. O WBEM é um projeto que começou a ser desenvolvido inicialmente por uma parceria entre as empresas Microsoft, Intel, Compaq e Cisco e que atualmente conta com a contribuição de mais de 75 empresas.

Basicamente o WBEM é um padrão que permite o gerenciamento de sistemas, englobando aspectos de *software* e *hardware*, em uma rede utilizando um *browser*. Como principais características do padrão temos a flexibilidade e simplicidade.

2.2 Microcontroladores

De forma geral, um microcontrolador pode ser entendido como um computador simples construído em um único chip, que pode ser programado e integra processador (Unidade Lógica Aritmética - ULA), memória, portas de I/O, dispositivos de comunicação serial dentre outros[19].

Devido a sua versatilidade, os microcontroladores passaram a ser largamente empregados em projetos de monitoramento e automação.

2.2.1 *Plataformas para automação*

Abaixo segue uma breve análise das principais plataformas para projetos de monitoramento e automação disponíveis no mercado.

- **Arduino**

  O Arduino é uma plataforma *open-source* de prototipagem eletrônica com *hardware* e *software* flexíveis e fáceis de usar[4]. A plataforma é composta por um hardware, que contém uma placa de prototipação e um microcontrolador ATMEGA2, e um software para programação através de um ambiente indegrado de desenvolvimento (IDE) *Cross-platform*.

  Existem diversos modelos da placa, as figuras 3 e 4 mostram os Arduinos UNO e NANO, dois dos principais modelos disponíveis no mercado.

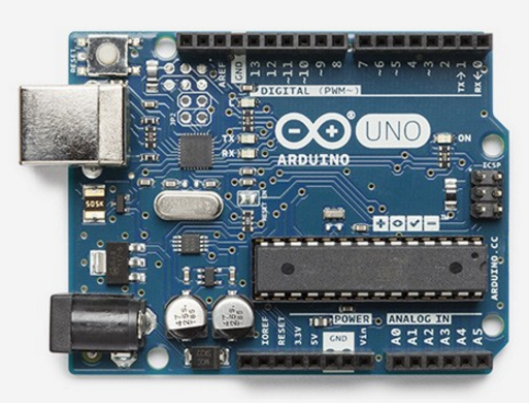

Figure 3: Arduino UNO R3[2].

Uma característica importante da plataforma é a possibilidade de utilização de *shields*, placas de expansão que ampliam as funcionalidades da placa básica. Na

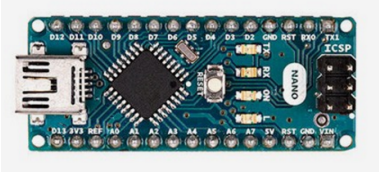

Figure 4: Arduino NANO[2].

figura 5 pode ser visto um *shields* para conexão de rede *Ethernet*.

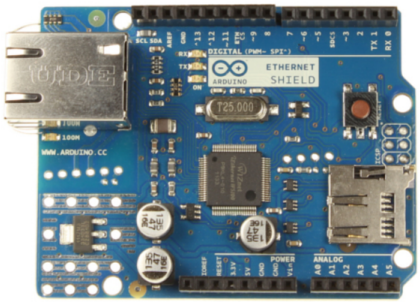

Figure 5: Arduino Ethernet Shield[2].

Além do baixo custo da plataforma, cerca de $50 versão UNO R3[4], outro aspecto interessante do Arduino é que tanto seu *hardware* quanto seu *software* são *Open source and extensible* tendo seus projetos publicados sob a *Creative Commons license*, de modo que qualquer pessoa pode construir as suas próprias versões da plataforma. Abaixo a figura 6 exibe uma imagem do Severino, uma placa construída com base nos *datasheets* disponibilizados pelo projeto Arduino.

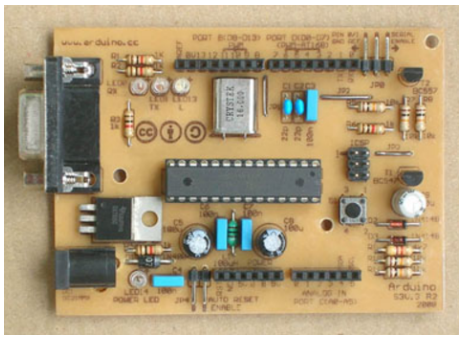

Figure 6: Severino - Placa baseada no projeto Arduino[3].

- **ESP8266**

  O ESP8266 é uma solução *WI-FI* integrada, projetada no modelo *System-on-a-Chip* (SoC)[7], produzida pela empresa Espressif Systems[10] e vem sendo amplamente utilizado em projetos de automação e monitoramento. O ESP8266, implementa a pilha TCPIP e também suporta os protocolos 802.11 b/g/n, além disso ele também integra em sua placa um microcontrolador de 32 bits, com frequência de 80 KHz e memorias RAM de 32kb e flash de 512kb. Abaixo a figura 7 exibe uma imagem da placa.

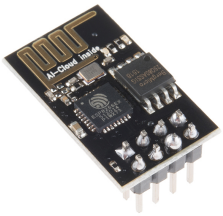

Figure 7: Módulo ESP8266[10].

Dois grandes destaques que podem ser dados ao módulo é o seu baixo custo e tamanho reduzido. A versão 01 mede cerca de 24mm x 16mm com um custo aproximado de $2,00. Existem várias versões do módulo, sendo as mais comuns no mercado às 01, 03 e 07. A figura 8 mostra os doze modelos disponíveis do módulo.

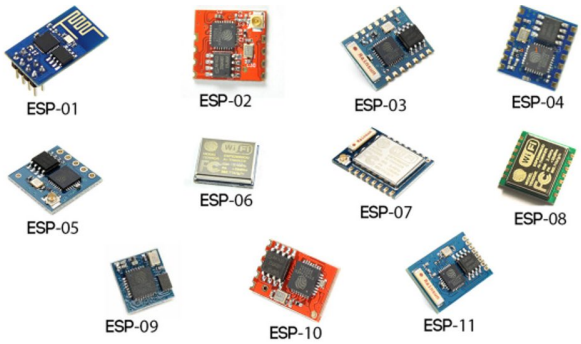

Figure 8: Versões do módulo ESP8266[14].

Algumas empresas do mercado fabricam outras placas baseadas no esp8266, as mais conhecidas são: NodeMcu, WeMos D1 mini, SparkFun Thing e Adafruit HUZZAH (figura 9). Essas placas tem como principais características a integração, em um PCB compatível com protoboard, o esp8266, um regulador de tensão de 5v para 3,3v, um conversor FTDI e um conversor USB-TTL[14].

- **Raspberry Pi**

  Apesar de não se enquadrar especificamente como um microcontrolador, o Raspberry Pi deve ser levado em consideração quando o assunto é automação e monitoramento. Isso se deve ao fato dele ser um computador completo de baixo custo tendo as dimensões aproximadas de um cartão de crédito.

  O Raspberry pi começou a ser desenvolvido em 2006 pela Fundação Raspberry Pi no Reino Unido como intuito de despertar o interesse pela computação através de um computador simplificado, flexível e que pudesse ser utilizado como uma ferramenta educacional[8]. No entanto ainda em sua fase de desenvolvimento foi percebido o potencial do pequeno equipamento para prototipação rápida e automação.

  A versão mais recente do Raspberry Pi, exibida na figura 10, vem equipada com um processador de 64-bit *quad-core* ARM *Cortex*-A53, 1 GB de memória ram LPDDR2, além de suporte a redes *Wireless* 802.11 b/g/n e *Bluetooth* 4.1[20].

Figure 9: Placas baseadas no ESP8266[14].

Figure 10: Raspberry Pi 3 Model B[20].

- **Intel Edison**

  O módulo Edison é um compacto chip de computação, com as dimensões aproximadas de um SD-card, desenvolvido pela empresa Intel[13]. O ponto forte do módulo é que ele possui um processador de 22nm rodando a 400 MHz, além de uma memória 512 DRAM, 1 GB para armazenamento e controladoras Wi-Fi e *Bluetooth* integradas.

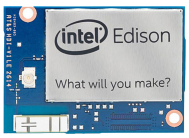

Figure 11: Módulo Intel Edison[13].

Para a prototipação rápida a Intel também criou placas de desenvolvimento que facilitam o uso do módulo Edison, como por exemplo o Intel Edison Breakout Board Kit, exibido na figura 12. Também é possível criar uma placa personalizada para o Edison.

Figure 12: Intel Edison Breakout Board Kit[13].

2.3 Internet of Things

Estar conectado deixou de ser uma característica específica de apenas alguns equipamentos e passou a englobar uma gama cada vez maior de "coisas". Isso foi possível graças a IoT, que possibilitou a comunicação entre os mais diversos tipos de objetos pela internet.

De forma mais abstrata, Atzori[6] conceitua a IoT como a presença de uma variedade de coisas ou objetos, identificados em uma rede, que interagem e cooperam entre si com um objetivos em comum . Com uma ótica mais tecnológica, McEven e Cassimally[15] sintetiza o conceito de IoT no resultado da soma entre objetos físicos, controladores, sensores, atuadores e serviço de Internet.

2.3.1 Arquitetura IoT

Atualmente existem duas principais arquiteturas que são usadas como referência para projetos de IoT.

- **Arquitetura de três camadas**

  Perception Layer (camada de percepção), camada física onde ficam os sensores e atuadores. *Network Layer* (camada de rede), responsável pela conexão entre as coisas, dispositivos de rede e servidores. *Application Layer* (camada de aplicação), responsável pela entrega de serviços específicos para o usuário[26].

- **Arquitetura de cinco camadas**

  É baseada na arquitetura de três camadas, tendo o mesmo papel para as camadas de percepção e aplicação. Em acréscimo temos há *Transport Layer* (camada de transporte), responsável por transmitir os dados da camada de percepção para a camada de processamento e vice-versa. A *Processing Layer* (camada de processamento), também conhecida como camada de *middleware*, armazena, analisa e processa os dados provenientes da camada de transporte. E a *Business Layer* (camada de negócio), responsável por gerenciar todo o sistema IoT[26].

2.3.2 Protocolos IoT

Um dos grandes desafios da IoT é garantir o bom funcionamento da rede, tendo em vista o aumento considerável do tráfego de dados devido a grande quantidade de equipamentos conectados e trocando informações. É nesse contexto que começam a surgir alguns protocolos de transferência de dados visando melhor atender as necessidades específicas deste ambiente.

- **MQTT**

  O *Messaging Queue Telemetry Transport* (MQTT) é um protocolo de troca de mensagens entre máquinas

desenvolvido por Andy Stanford-Clark (IBM), e Arlen Nipper (Arcom, atual Eurotech), em 1999. Ele segue o paradigma *publish/subscribe* e foi projetado para ser usado em ambientes com largura de banda de rede limitada e em dispositivos com *hardware* também limitado[18].

O MQTT é baseado no TCP/IP, tendo como grande vantagem em relação ao HTTP o *payload* (tamanho da mensagem) reduzido. A comunicação é feita através do envio de mensagens no formato de tópico para um *Middleware* centralizador, chamado de *broker*. O *broker* empilha e transmite a mensagem para os os equipamentos assinantes daquele tópico.

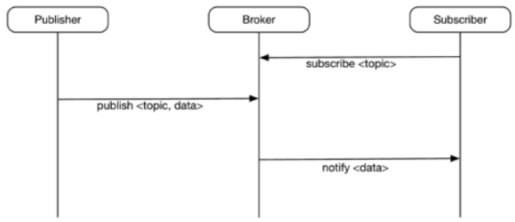

Figure 13: Comunicação do Protocolo MQTT[27].

Como pode ser visto na figura 14, o formato da mensagem é composto por um cabeçalho fixo de 2 *bytes*, sendo o primeiro responsável por armazenar o tipo da mensagem e os marcadores responsáveis que identificam o nível de qualidade do serviço (QoS, *Quality of Service*), de entregas duplicadas (DUP, *Duplicate delivery*) e retenção de mensagem (RETAIN)[17].

bit	7	6	5	4	3	2	1	0
Byte 1	Tipo da Mensagem				Flag DUP	Nível QoS		RETAIN
Byte 2	Largura restante							

Figure 14: Cabeçalho de uma mensagem MQTT[17].

Apesar de simplificado o MQTT implementa uma estrutura de níveis de qualidade do serviço. No nível 0, a mensagem é enviada no máximo uma vez, ou seja, é enviada e esquecida. No nível 1, a mensagem é enviada pelo menos uma vez, de forma que o publicador necessita da confirmação do recebimento pelo *broker*. Por fim no nível 2, a entrega da mensagem é garantida, devendo ter confirmação de ambos os lados.

2.4 Trabalhos Relacionados

Nesta seção serão apresentados alguns trabalhos acadêmicos que abordam temas como automação e monitoramento de ambientes, verificando os pontos importantes de cada proposta em comparação com o SMDCIC.

Trabalho A.

Monitoramento De Temperatura E Umidade De Data Center Utilizando o Arduino e o Sistema ZABBIX[26]

O sistema desenvolvido utiliza os conceitos de IoT para o monitoramento da umidade e temperatura de um data center. Para isso é utilizada uma placa arduino, um *Ethernet Shield* e um sensor DHT22 na coleta dos dados e também o Zabbix para o monitoramento. Os pontos fortes do sistema são o baixo custo e a integração com o protocolo SNMP.

Trabalho B.

Sistema SCADA Para Supervisão de Temperatura e Umidade[22]

O trabalho utiliza o ScadaBR para o desenvolvimento de um sistema supervisor que monitora a umidade e temperatura de ambientes. Para a realização da aquisição e transmissão dos dados foi criado um dispositivo composto por um sensor de umidade e temperatura DHT11, um microcontrolador e módulos de rádio frequência. Como pontos fortes do trabalho temos a flexibilidade de ser possível a implantação na maioria dos ambientes e o tamanho reduzido do módulo de coleta de dados.

Trabalho C.

Unificando Agentes Móveis Inteligentes com WBEM para o Gerenciamento Corporativo de Sistemas[23]

O trabalho realiza um estudo a fim de verificar a viabilidade da utilização do padrão de *WBEM* no Gerenciamento Corporativo de Sistemas (GCS). Para isso o autor desenvolveu um *framework* que realiza a interação entre a iniciativa *WBEM* e agentes moveis inteligentes. Um aspecto importante do trabalho é a aplicação da escalabilidade oferecida pelo *WBEM* no GCS.

É importante salientar que esse trabalho não tem como objetivo a realização do gerenciamento ou monitoramento de aspectos físicos do ambiente e foi escolhido pelo fato de realizar uma proposta a coleta de dados lógicos (disco e rede) dos servidores, sendo que o mesmo.

A figura 15 mostra uma tabela comparativa entres os trabalhos analisados e o SMDCIC.

Características	Trabalho A	Trabalho B	Trabalho C	SMDCIC
Escalabilidade	Baixa	Baixa	Alta	Alta
Implantação	Complexa	Complexa	Complexa	Simples
Envio de Dados	Ethernet	Rádio frequência	Ethernet	Wi-Fi
App mobile	Não	Não	Não	Sim
Notificação	E-mail	Não	Não	Push Notification

Figure 15: Comparação dos trabalhos como o SMDCIC.

3. SOLUÇÃO DESENVOLVIDA - SMDCIC

Com base no que foi apresentado, foi desenvolvido o Sistema para Monitoramento de Data Centers baseado na Internet das Coisas – SMDCIC, uma solução que mescla *software* e *hardware* a fim de disponibilizar, de forma online, informações do *data center*. Nesta seção é apresentado o processo de construção do sistema, detalhando os elementos e ferramentas utilizados.

3.1 Requisitos

O primeiro passo para a construção do sistema foi estabelecer o escopo da aplicação, definindo os pontos a serem monitorados. Como elementos físicos do ambiente foram definidos:

- **Temperatura**

  Monitorar a temperatura do ambiente é importante porque os equipamentos de um *data center*, durante

seu funcionamento, geram muito calor e dissipar o mesmo no ambiente. Um valor elevado pode causar sobreaquecimento dos servidores e consequentemente desligamento ou até mesmo danos irreversíveis ao mesmo.

- **Umidade**

  A alta umidade na sala do *data center* pode provocar condensação de água nos servidores, já a baixa umidade pode gerar curtos dentro dos servidores devido a geração de carga eletrostática.

- **Fumaça**

  A presença de fumaça no ambiente é um indicativo de incêndio, por isso também deve ser monitorada.

- **Corrente da rede elétrica**

  Monitorar a corrente elétrica se faz necessário uma vez que oscilações e interrompimento de energia podem causar o desligamento e reinicialização dos servidores.

Já na parte lógica forma definidos o espaço do disco dos servidores e disponibilidade do *link* de dados.

- **Espaço em disco rígido**

  Esse ponto é importante para saber se espaço disponível está próximo do limite e poder realizar medidas de escalonamento. Outro aspecto é prevenir possíveis perdas devido a falhas ao tentar salvar informações.

- **Status do *link* de dados**

  O monitoramento do *link* é importante para não comprometer a disponibilidade dos serviços agindo rapidamente caso o mesmo fique *offline*.

Também foram definidos os elementos que não estão diretamente ligados ao monitoramento, mas são de suma importância para o propósito do sistema (requisitos não-funcionais). Nesse sentido temos:

- **Escalabilidade**

  Deve ser possível aumentar ou diminuir o número de componentes sem causar impactos ao sistema.

- **Heterogeneidade**

  O sistema deve ser capaz de lidar com componentes que distinguem nos seguintes aspectos: Sistema operacional, *Hardware* ou linguagem de programação. Em outras palavras, o SMDCIC pode ter componentes escritos em Python rodando em um servidor Linux, componentes escritos em C# rodando em um *desktop* Windows ou mesmo componentes escritos em Java rodando em um dispositivo *mobile* Android.

A delimitação do escopo de monitoramento em conjunto com os aspectos não-funcionais do sistema, foram traduzidos em um documento descritivo de requisitos que se encontra no Apêndice A deste trabalho.

3.2 Arquitetura

Após a definição dos requisitos foi desenvolvida a arquitetura prescritiva do sistema. Para lidar com as questões de escalabilidade e heterogeneidade o SMDCIC se adotou como referência o estilo arquitetural *Event-Based*, que baseia-se em invocação implícita. O *Event-Based* consiste na comunicação independente de componentes através de um barramento de eventos. Ele é semelhante ao *publish/subscribe*, exceto pelo fato de um mesmo componente poder publicar e receber publicações. O protocolo utilizado para a comunicação interna é o MQTT, já para comunicação para a comunicação externa foi utilizado o HTTP.

Como pode ser visto na figura[16], o sistema é composto pelos seguintes elementos: Message Broker, Cliente, Banco de Dados, App *mobile*.

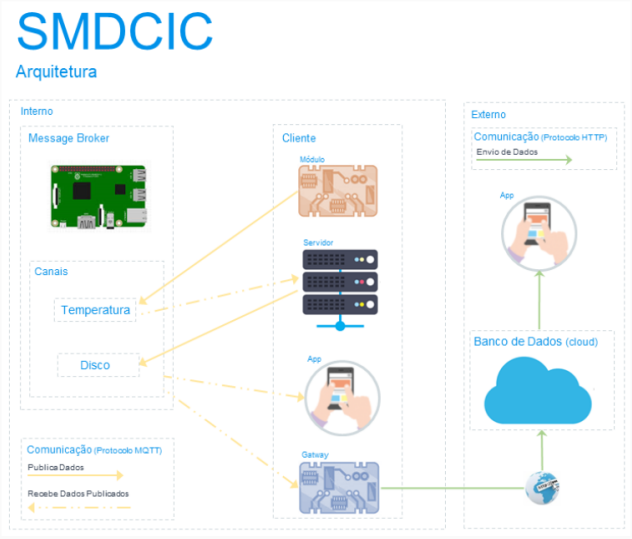

Figure 16: Arquitetura SMDCIC.

Clientes

Esse elemento da arquitetura é responsável por publicar dados, receber dados publicados ou efetuar as duas atividades concomitantemente. Publicadores enviam suas mensagens para canais (ou tópico), enquanto assinantes recebem as mensagens dos canais que estão inscritos.

Message Broker

O *Message Broker* (corretor de mensagens), é o conector responsável por receber as mensagens, criar os canais e enviar as mensagens recebidas para os clientes assinantes dos canais.

Banco de dados

Outro componente da arquitetura é o banco de dados na nuvem, que armazena todos os dados coletados e publicados e os disponibilizam a aplicações externas. A persistência dos dados é feita através de um cliente assinante de todos os canais providos pelo Message Broker.

App *Mobile* (externo)

Esse componente consulta as informações do banco de dados e as exibem em seu *dashboard*.

3.3 Modelagem

A modelagem do sistema foi feita usando a UML - Linguagem de Modelagem Unificada (do inglês, *Unified Modeling Language*), através da qual foi elaborado os diagramas de caso de uso do sistema.

As figuras 17, 18 e 19 ilustram os diagramas de caso de uso em que as ações são desempenhadas por máquinas, já a figura 20 representa as ações do usuário através dos apps *mobiles* interno e externo.

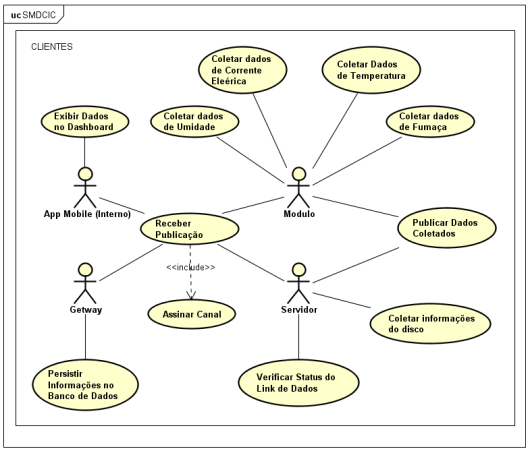

Figure 17: Caso de Uso SMDCIC (Clientes).

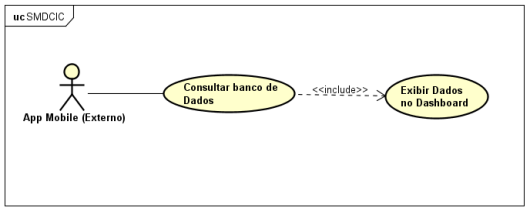

Figure 18: Caso de Uso SMDCIC (App Externo).

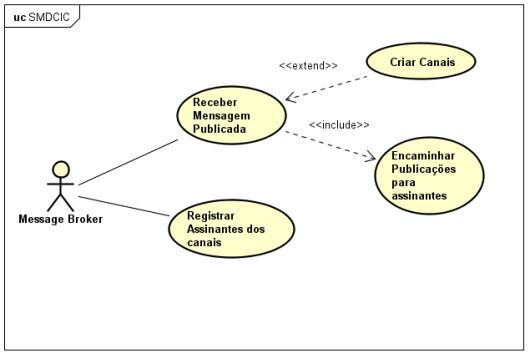

Figure 19: Caso de Uso SMDCIC (Message Broker).

3.4 Ferramentas

Essa subseção detalha as ferramentas, tanto de *hardware* quanto de *software*, utilizadas para a implementação e implantação do SMDCIC.

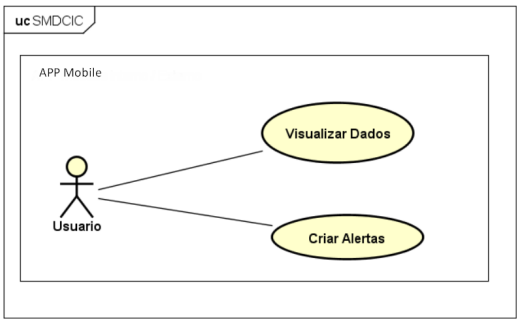

Figure 20: Caso de Uso SMDCIC (Usuário).

- ***Hardware***

  Sensor DH11 - Realiza a leitura de temperaturas entre 0 a 50 Celsius, através de um termistor do tipo NTC, e a leitura da umidade entre 20 a 90%, através de um sensor do tipo HR202. O circuito interno do DH11 captura as informações do ambiente e as enviam para um microcontrolador.

  Sensor Gás MQ-2 - Detecta a concentração no ar de fumaça e gás inflamável, tais como GLP, Metano, Propano, Butano, Hidrogênio, Álcool, Gás Natural entre outros. A sensibilidade pode ser ajustada via potenciômetro.

  Sensor SCT-013 20A - É um sensor não invasivo capaz de medir corrente até 20A. O sensor trabalho em temperaturas entre -25 a 70 °C, com um corrente de entrada entre 0-20A e sinal de saída de 1V (tensão).

  Protoboard - É uma ferramenta para prototipagem que permite a montagem de circuitos eletrônicos sem a necessidade de solda. Fabricada em plástico ABS não propaga chamas.

  Jumpers - Usado em conjunto com *protoboard*, são uma ótima ferramenta de auxílio a prototipação.

  NodeMCU - é uma placa de desenvolvimento que combina, em um mesmo PCB, o chip ESP8266, uma interface usb-serial e um regulador de tensão 3.3V. Pode ser programada usando a IDE do Arduino, ainda possui 11 pinos I/O, conversor analogio-digital, antena embutida e um conector micro-usb. O NodeMCU ainda suporta upgrade remoto de *firmware* e dimensões de 49 x 25,5 x 7 mm.

  Raspberry PI 3 - É um microcomputador completo com dimensões, aproximadas, de um cartão de crédito. A versão PI 3 conta com processador Broadcom Quad Core BCM2837 de 64 bits e *clock* de 1.2GHz, 1G de memória RAM, além de adaptadores *Wi-fi* e *Bluetooth* 4.1 integrados.

Os componentes apresentados foram utilizados para a confecção dos Clientes e *Message Broker*. A figura 21 mostra os itens utilizado na construção dos componentes.

- **Software**

  Arduino IDE - é um *software* usado para escrever e gravar programas (*firmwares*) nas placas Arduino. Uma de suas principais características é a possibilidade

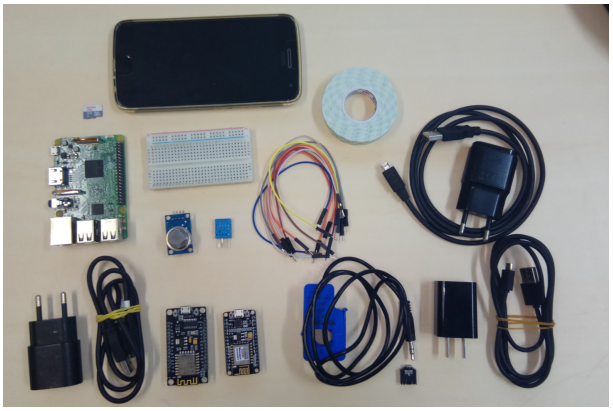

Figure 21: Componentes utilizados.

de estender funcionalidades através de bibliotecas. O Arduíno IDE foi usado para programação das placas NodeMCU, através das bibliotecas disponíveis para o esp8266.

Android Studio - é o ambiente de desenvolvimento, oficial fornecido pela Google, para dispositivos Android. A ferramenta foi usada no desenvolvimento dos APPs *mobiles*.

PyCharm - é uma ferramenta para a programação em Python e foi usada para a desenvolvimento do cliente que coleta as informações do servidor.

Eclipse Mosquitto - é um *broker* de código aberto mantido pela iot.elipse.org que implementa o protocolo MQTT nas versões 3.1 e 3.1.1. O *mosquitto* foi utilizado no gerenciamentos dos eventos de envio e recebimento de mensagens pelo *Message Broke*r.

Raspbian - é um dos sistemas operacionais disponíveis para a instalação no Raspberry PI 3. Ele é baseado no Debian e foi utilizado como servidor MQTT.

3.5 Funcionamento

Como dito anteriormente o SMDCIC integra quatro tipos de elementos, *Message Broker*, Cliente, Banco de dados e *App Mobile* externo. Segue o detalhamento do funcionamento de cada um deles.

Clientes

A função dos Clientes é coletar dados e publicá-los através do envio de mensagens, categorizadas por canais, ao *Message Broker*. Existem quatro tipos de clientes: Módulo, Servidor, *Gateway* e *App mobile*.

- **Módulos**

  O *hardware* de um módulo é composto pelo conjunto de sensor e placa NodeMCU montados em uma *protoboard* e interligados com *jumpers*. Cada placa NodeMCU de um módulo recebe o *firmware* específico responsável por ler as informações capturadas pelo sensor, conectar-se com o *Message Broker* e enviar os dados. O *firmware* também pode ser configurado, se for o caso, para assinar canais de interesse e receber as mensagens forem publicadas nesses canais. Para escrever os *firmwares* e gravá-los no NodeMCU foi utilizada a Arduino IDE.

- ***Gateway***

  O *Gateway* é composto só por uma placa NodeMCU, pois seu papel se resume em apenas persistir os dados coletados pelos demais clientes no banco de dados. Para isso, *firmware* do *Gateway* está programado para se conectar ao *Message Broker* e assinar todos os canais existentes assim todas as mensagens publicadas são recebidas por ele. Periodicamente o *Gateway* se conecta do banco de dados e faz a persistência do mesmos. Também foi utilizado o Arduino IDE para desenvolver e gravar o *firmware*.

- **Servidor**

  Como uma das funcionalidades do SMDCIC é monitorar informações lógicas do *data center*, foi desenvolvida em Python uma aplicação para os servidores que fazer a coleta periódica dos dados de disco e rede. Para aquisição dos dados foi utilizado o módulo *subprocess* do Python, que possibilita a execução de comandos *shell* através da aplicação, assim periodicamente as informações são coletadas e publicadas. Aplicação dos servidores também conta com a implementação de um cliente MQTT responsável por se conectar ao *Message Broker*, publicar e receber mensagens. A aplicação do servidor foi desenvolvida utilizando a ferramenta Pycharm.

- ***App Mobile* (interno)**

  O *App mobile* interno, consiste em um aplicativo que se conecta ao *Message Broker* e pode assinar canais pra receber publicações ou publicar mensagens. Como *App mobile* interno foi utilizado o IoT MQTT Dashboard[24], um *app* para a plataforma Android que implementa o protocolo MQTT e pode ser instalado em *tablets* ou *smatphones*.

Message Broker

O *Message Broker* é um conector que implementa um servidor MQTT para receber as mensagens publicadas e as despacham para os clientes assinantes dos canais. Vale ressaltar que o conector não armazena nenhuma mensagem, apenas verifica quais clientes estão inscritos no canal da mensagem publicada e encaminha a mesma para os respectivos assinantes. Em termos de hardware o *Message Broker* é composto de um Raspberry PI 3 rodando como sistema operacional Resbian. O servidor MQTT utilizado foi o Mosquitto.

Banco de dados

Para armazenamento das informações foi utilizado o Firebase, que é um banco de dados não relacional na nuvem (*cloud*) mantido pela empresa Google. As mensagens são armazenadas no Firebase organizadas por canal, ID do sensor, valor e data.

App Mobile (externo)

O funcionamento do *App mobile* externo é similar ao interno exceto que não possui um cliente MQTT, pois obtém os dados através de consulta ao banco de dados Firebase. Outra característica do *app* externo é a possibilidade criação de alertas pelo usuário. Para desenvolver o aplicativo foi utilizado o Android Studio e a linguagem de programação Java.

Comunicação

No ambiente do *data center* a comunicação é feita através de rede interna utilizando o protocolo MQTT. Já para a comunicação externa entre o *Gateway*, Firebase e *App mobile*, é feita pela internet através do http. Toda a comunicação é feita por *Wi-Fi*, exceto o servidor que utiliza conexão cabeada *ethernet*.

As publicações seguem o seguinte padrão: canal/ID do sensor/valor

Na figura 22 pode ser visto o esquema de funcionamento do SMDCIC.

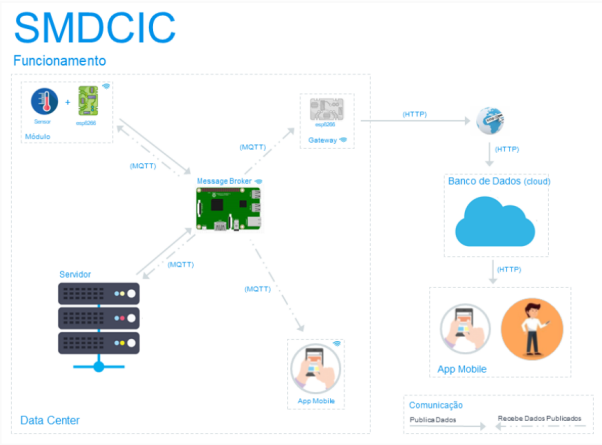

Figure 22: Funcionamento do SMDCIC.

3.6 Implantação

Nesta seção, são apresentados os resultados obtidos com o SMDCIC. Para isso o sistema foi utilizado para monitorar um dar center mantido pela empresa Computação Brasil, que também disponibilizou um servidor para testes.

Hardware

No ambiente físico do data center foram implantados três componentes sendo um *Mesage Broker*, um Módulo e um *Gateway*. Com os três componentes se comunicam através de *Wi-fi* eles podem ser posicionados em qualquer local do ambiente, porém é importante alocá-los em pontos estratégico de monitoramento. Da figura 23 à 29 são exibidas as imagens da alocação de cada componente no data center.

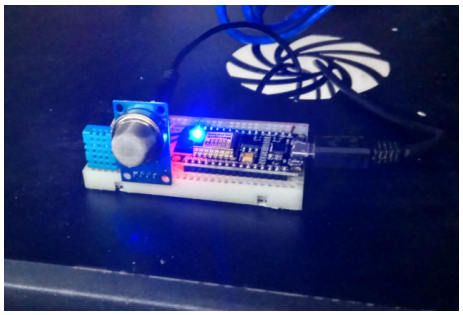

Figure 23: Posicionamento do módulo sensor no *data center*.

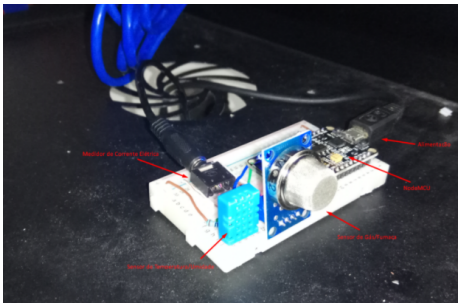

Figure 24: Posicionamento do módulo sensor no *data center*.

Figure 25: Posicionamento do sensor SCT-013 20A.

Figure 26: *Message Broker*.

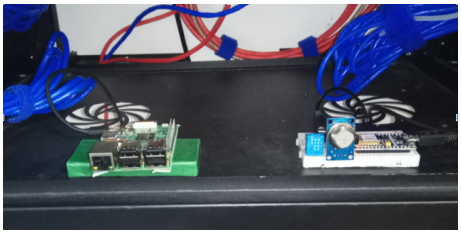

Figure 27: A esquerda *Message Broker* e a direita módulo sensor.

Figure 28: A esquerda *Message Broker*, no meio *App Mobile* (interno) e a direita módulo sensor.

Figure 29: *Gateway*.

Software

Também foi instalado um cliente no servidor para a coleta das informações de disco e status do *link* de dados, como pode ser visto na figura 30.

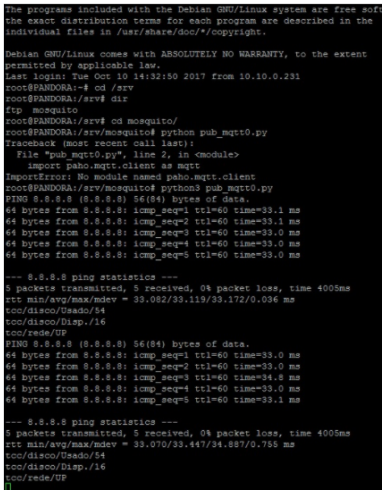

Figure 30: Cliente instalado no servidor.

Para exibição interna dos dados monitorados foi instalado o IoT MQTT Dashboard em um aparelho Android, que recebe os dados enviados diretamente pelo *Message Broker* – figura 31.

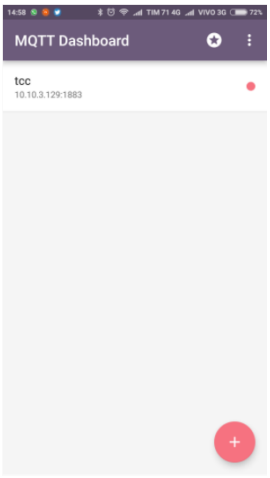

Figure 31: *App Mobile* Interno, IoT MQTT Dashboard.

Os dados persistidos no banco de dados podem ser visualizados pelo usuário através do App Mobile externo desenvolvido, como mostram as figuras 32 e 33.

Figure 32: Dados exibidos pelo *App Mobile* Externo.

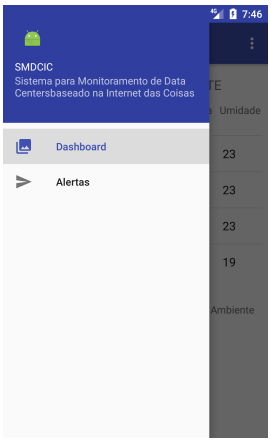

Figure 33: Dados exibidos pelo *App Mobile* Externo.

4. RESULTADOS OBTIDOS

Um dos principais pontos a ser verificado foi a escalabilidade assim a cada passo da implantação foram sendo coletado os resultados do monitoramento. Primeiramente foi instalado o cliente no servidor a fim de coletar os dados lógicos do sistema, como pode ser visto na figuras 34. Nesse momento os sensores de ambientes ainda não tinham sido colocados no data center.

Figure 34: Medição apenas dos elementos lógicos do *data center*.

Posteriormente foram incorporados os sensores de umidade, temperatura, fumaça e rede elétrica. figuras 35 e 36.

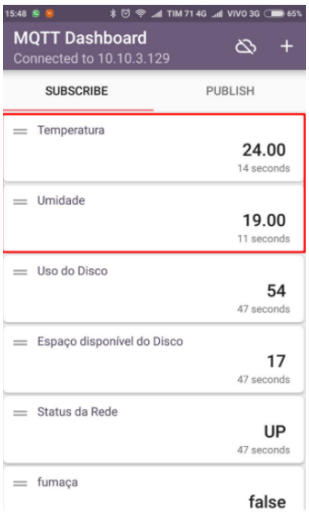

Figure 35: Inclusão das medições de temperatura e umidade.

Como esperado a adição de novos elementos ao sistema não causou impactos aos que já haviam sido implantados, assim ao incorporar mais pontos de monitoramento não foi preciso reiniciar o sistema, apenas configurar a exibição no cliente Mobile.

Durante o processo de acompanhamento do SMDCIC foram feitas algumas simulações afim de verificar os dados coletados pelo sistema.

Figure 36: Inclusão das medições de fumaça e status da rede elétrica.

Umidade / Temperatura

Foram feitas algumas alterações de temperatura através do controlador do ar condicionado e observado as leituras capturadas pelo SMDCIC. As figuras 37 e 38 mostram a variação de temperatura e umidade.

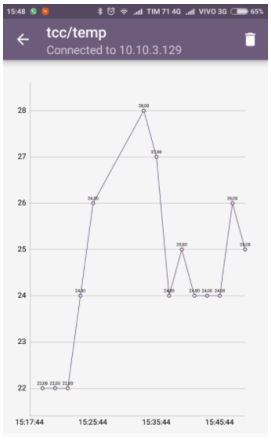

Figure 37: Dados da temperatura.

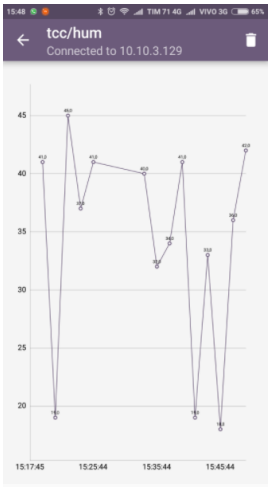

Figure 38: Dados da umidade.

Rede Elétrica

Para testes do monitoramento da rede elétrica, com o módulo devidamente instalado, o sensor SCT-013 foi periodicamente conectado e desconectado, afim de simular a interrupção da corrente elétrica. Os resultados podem ser vistos na figura 39.

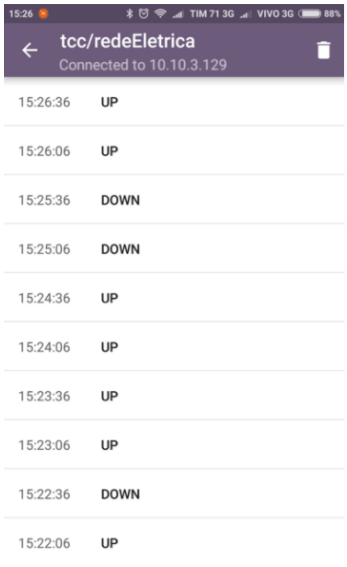

Figure 39: Dados de monitoramento da rede elétrica.

Uso do Disco

Forma copiados e removidos alguns arquivos do disco do servidor. A figura 40 representa as informações de variação do espaço usado do disco (GB) durante os testes, já figura 41 representa o espaço disponível (%).

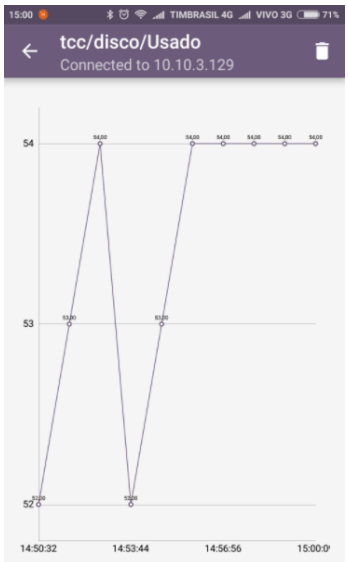

Figure 40: Dados do uso do disco (GB).

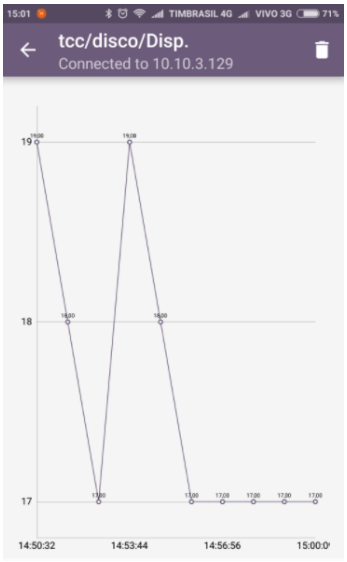

Figure 41: Dados do espaço disponível (%).

Link de Dados

O teste do *link* de dados foi feito bloqueando o acesso a rede externa pelo servidor, assim o teste de *ping* realizado pelo cliente instalado resulta em falha, publicando uma mensagem de status "*down*" da rede. Os resultados podem ser vistos na figura 42.

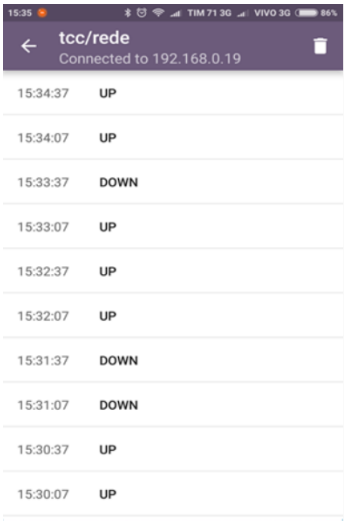

Figure 42: Dados de monitoramento do status da rede.

Também foram disparados alertas de notificação ao usuário durante os testes pelo App Mobile externo, como podem ser vistos nas figuras 43 e 44.

Figure 43: *App Mobile* externo, alerta de notificação.

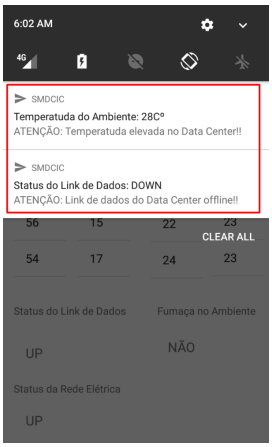

Figure 44: *App Mobile* externo, alerta de notificação.

A interface para criação dos alertas pode ser vista na figura 45.

Figure 45: *App Mobile* externo, interface de criação de alertas.

5. CONCLUSÃO

O desenvolvimento da área de IoT traz com ela novas tecnologias que podem ser aplicadas nos mais diversos fins. Um exemplo de aplicabilidade é em *data centers*, pois devido ao grau de importância para a organização no qual estão inseridos é fundamental o seu monitoramento. O presente trabalho apresentou o SMDCIC uma ferramenta composta por *hardwares* e *softwares* para o monitoramento de *data centers*. A motivação foi desenvolver uma solução que aplique os novos conceitos e tecnologias da área de IoT no contexto dos *data centers*, realizando o monitoramento do ambiente físico e dos servidores. As principais características do sistema são escalabilidade e facilidade de implantação, pois possibilita a adição de novos elementos para monitoramento sem a necessidade de parar o sistema.

- **Trabalhos futuros**

  Por se tratar de um protótipo e aplicado em um ambiente controlado, alguns aspectos não foram considerados ficando como propostas de melhorias a serem desenvolvidas e implementadas em trabalhos futuros. São elas:

  Melhoria dos componentes de hardware afim de criar uma unidade que não exponha os circuitos e suas conexões;

  Estudo mais aprofundado dos materiais utilizados tais como cabos, conectores e etc.;

  Desenvolvimento de um sistema de alimentação dos componentes independente da rede elétrica do data center;

  Implantação do protocolo SSL para comunicação entre os componentes;

  Implantação de níveis de qualidade de serviço (QoS), afim de garantir a entrega a mensagem;

6. REFERENCIAS

[1] ANSI/TIA-942. Telecommunications industry association. about data centers. https://manuais.iessanclemente.net/images/9/9f/Tia942.pdf. Acesso em 11/07/2017.

[2] ARDUINO. Arduino products. https://www.arduino.cc/en/Main/Products. Acesso em 13/07/2017.

[3] ARDUINO. Arduino single-sided serial board (version 3). https://www.arduino.cc/en/Main/ArduinoBoardSerialSingleSided3. Acesso em 13/07/2017.

[4] ARDUINO. What is arduino? https://www.arduino.cc/en/Guide/Introductionf. Acesso em 13/07/2017.

[5] ATIVA. O que é um software de supervisão scada? http://www.ativa-automacao.com.br/como-fazer-manutencao-em-clp-xxxx//. Acesso em 14/07/2017.

[6] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[7] R. J. d. Azevedo et al. Uma arquitetura para execução de codigo comprimido em sistemas dedicados. http://repositorio.unicamp.br/jspui/handle/REPOSIP/276216, 2002.

[8] R. BATISTELLO. AutomaÇÃo residencial utilizando raspberry pi e android. http://www.webartigos.com/storage/app/uploads/public/588/4ce/5f4/5884ce5f4b96e498733571.pdf, 2014. Acesso em 16/07/2017.

[9] D. COLE. Data center knowledge guide to: Data center infrastructure management (dcim). https://files.vogel.de/vogelonline/vogelonline/files/5805.pdf. Acesso em 10/07/2017.

[10] ESPRESSIF. Esp8266, overview. https://espressif.com/en/products/hardware/esp8266ex/overview. Acesso em 13/07/2017.

[11] F. Ferraz Júnior. *Arquiteturas para monitoramento e supervisão integrados de processos de usinagem em máquinas com controle numérico aberto*. PhD thesis, Universidade de São Paulo, 2007.

[12] I. GARTNER. Data center infrastructure management. http://www.gartner.com/it-glossary/data-center-infrastructure-management-dcim/. Acesso em 13/07/2017.

[13] Intel. Intel® edison development platform. https://www.intel.com/content/dam/support/us/en/documents/edison/sb/edison_pb_331179002.pdf. Acesso em 16/07/2017.

[14] L. S. IRVING. Nodemcu (esp8266) o módulo que desbanca o arduino e facilitará a internet das coisas. http://irving.com.br/esp8266/nodemcu-esp8266-o-modulo-que-desbanca-o-arduino-e-facilitara-a-internet-das-coisas/. Acesso em 10/07/2017.

[15] F. Lacerda. Arquitetura da informação pervasiva: projetos de ecossistemas de informação na internet das coisas. 2016.

[16] P. S. MARIN. *Data centers: Desvendando cada passo: conceitos, projeto, infraestrutura física e eficiência energética*. Ed. Érica, 1 edition, 2011.

[17] I. R. Martins and J. L. Zem. Estudo dos protocolos de comunicação mqtt e coap para aplicações machine-to-machine e internet das coisas. *Revista Tecnológica da Fatec Americana*, 3(1):24, 2016.

[18] MQTT. Frequently asked questions. http://mqtt.org/faq. Acesso em 13/07/2017.

[19] R. S. PENIDO, Édilus de Carvalho Castro; TRINDADE. *Microcontroladores.* e-Tec Brasil, 2013. Acesso em 14/07/2017.

[20] R. PI. Raspberry pi hardware guide. https://www.raspberrypi.org/learning/hardware-guide/. Acesso em 16/07/2017.

[21] R. Reginato. Sistemas scada e sistemas supervisórios. http://www.foz.unioeste.br/~romeu/CIP/2_Aula_scada.pdf. Acesso em 13/07/2017.

[22] W. S. d. Silva. Sistema scada para supervisão de temperatura e umidade. B.S. thesis, Universidade Tecnológica Federal do Paraná, 2016.

[23] M. A. C. SIMÕES. Unificando agentes móveis inteligentes e wbem para o gerenciamento corporativo de sistemas. 2003.

[24] N. TH. Iot mqtt dashboard. https://play.google.com/store/apps/details?id=com.thn.iotmqttdashboard&hl=pt_BR. Acesso em 13/07/2017.

[25] TIA-942. tia-942. about data centers. http://www.tia-942.org/content/162/289/About_Data_Centers. Acesso em 11/07/2017.

[26] L. M. d. VASCONCELOS. Monitoramento de temperatura e umidade de data center utilizando o arduino e o sistema zabbix. http://pt.slideshare.net/lailtonmontenegro/monitoramento-de-temperatura-e-umidade-de-data-center-utilizando-o-arduino-e-o-sistema-zabbix. Acesso em 14/07/2017.

[27] A. Vicenzi. Mqtt parte 1: O que é mqtt? http://www.butecopensource.org/mqtt-parte-1-o-que-e-mqtt/. Acesso em 13/07/2017.

APÊNDICES

A. DOCUMENTO DESCRITIVO DE REQUISITOS

RF001	**Coletar dados de Temperatura**	
Ator(es)	Módulo	
Pré-condição	Sensor de temperatura conectado	
Pós-Condição	Valor de temperatura do ambiente coletado	
Cenário de Sucesso Principal		
S001	Inicia comunicação com o sensor de temperatura	E001
S002	Ler a informação capturada pelo sensor de temperatura	E002
S003	Armazena a leitura em uma estrutura de dados interna	
Exceções		
E001	Falha na comunicação com o sensor de temperatura	F001
E002	Falha ao tentar ler as informações do com o sensor de temperatura	F001
Fluxo Alternativo		
F001	001	Continua tentando por 60 segundos
	002	Executa o [RF007] enviando uma mensagem de falha como ID do sensor

RF002	**Coletar dados de Umidade**	
Ator(es)	Módulo	
Pré-condição	Sensor de umidade conectado	
Pós-Condição	Valor de umidade do ambiente coletado	
Cenário de Sucesso Principal		
S001	Inicia comunicação com o sensor de umidade	E001
S002	Ler a informação capturada pelo sensor de umidade	E002
S003	Armazena a leitura em uma estrutura de dados interna	
Exceções		
E001	Falha na comunicação com o sensor de umidade	F001
E002	Falha ao tentar ler as informações do com o sensor de umidade	F001

Fluxo Alternativo		
F001	001	Continua tentando por 60 segundos
	002	Executa o [RF007] enviando uma mensagem de falha como ID do sensor

RF003	**Coletar dados de Fumaça**	
Ator(es)	Módulo	
Pré-condição	Sensor de fumaça conectado	
Pós-Condição	Valor de fumaça do ambiente coletado	
Cenário de Sucesso Principal		
S001	Inicia comunicação com o sensor de fumaça	E001
S002	Ler a informação capturada pelo sensor de fumaça	E002
S003	Armazena a leitura em uma estrutura de dados interna	
Exceções		
E001	Falha na comunicação com o sensor de fumaça	F001
E002	Falha ao tentar ler as informações do com o sensor de fumaça	F001

Fluxo Alternativo		
F001	001	Continua tentando por 60 segundos
	002	Executa o [RF007] enviando uma mensagem de falha como ID do sensor

RF004	**Coletar dados da Corrente Elétrica**	
Ator(es)	Módulo	
Pré-condição	Sensor de Corrente Elétrica conectado	
Pós-Condição	Valor da Corrente Elétrica coletado	
Cenário de Sucesso Principal		
S001	Inicia comunicação com o sensor de Corrente Elétrica	E001
S002	Ler a informação capturada pelo sensor de Corrente Elétrica	E002
S003	Armazena a leitura em uma estrutura de dados interna	
Exceções		

E001	Falha na comunicação com o sensor de Corrente Elétrica		F001
E002	Falha ao tentar ler as informações do com o sensor de Corrente Elétrica		F001
Fluxo Alternativo			
F001	001	Continua tentando por 60 segundos	
	002	Executa o [RF007] enviando uma mensagem de falha como ID do sensor	
RF005	**Coletar Informações do Disco**		
Ator(es)	Servidor		
Pré-condição	Módulo subprocess;		
Pós-Condição	Informações do disco coletadas		
Cenário de Sucesso Principal			
S001	Executa o shell script para captura de informações do disco		E001
S002	Armazena as informações em estrutura interna		
Exceções			
E001	Sem permissão para acessar informações do disco		F001
Fluxo Alternativo			
F001	001	Continua tentando por 60 segundos	
	002	Executa o [RF007] enviando uma mensagem de falha como ID cliente	
RF006	**Verificar Status do Link de Dados**		
Ator(es)	Servidor		
Pré-condição	Módulo subprocess;		
Pós-Condição	Status do Link de Dados Verificado		
Cenário de Sucesso Principal			
S001	Executa o shell script do utilitário ping		E001
S002	Realizar um ping para um endereço externo		
S003	Armazenar status da resposta do ping		
Exceções			

E001	Sem permissão para acessar informações da rede		F001
Fluxo Alternativo			
F001	001	Continua tentando por 60 segundos	
	002	Executa o [RF007] enviando uma mensagem de falha como ID cliente	
RF007	**Publicar Dados Coletados**		
Ator(es)	Módulo, Servidor		
Pré-condição	Cliente MQTT instalado;		
Pós-Condição	Mensagem enviada para o Message Broker ;		
Cenário de Sucesso Principal			
S001	Iniciar conexão como o Message Broker		E001
S002	Criar mensagem de envio		
S003	Enviar mensagem		
Exceções			
E001	Message Broker indisponível		F001
Fluxo Alternativo			
F001	001	Tenta novamente após 60 segundos	
RF008	**Assinar Canal**		
Ator(es)	Módulo, Servidor, Gateway, App(interno)		
Pré-condição	Cliente MQTT instalado;		
Pós-Condição	Canal assinado;		
Cenário de Sucesso Principal			
S001	Iniciar conexão como o Message Broker		E001
S002	Criar mensagem de solicitação de assinatura em de um canal		
S003	Enviar mensagem para o Message Broker		
Exceções			
E001	Message Broker indisponível		F001

Fluxo Alternativo		
F001	001	Tenta novamente após 60 segundos
RF009	**Receber Publicação**	
Ator(es)	Módulo, Servidor, Gateway, App(interno)	
Pré-condição	Cliente MQTT instalado; Ter assinado um canal;	
Pós-Condição	Mensagem recebida;	
Cenário de Sucesso Principal		
S001	Iniciar conexão como o Message Broker	E001
S002	Recebe mensagem	
S003	Armazenar mensagem recebida	
Exceções		
E001	Message Broker indisponível	F001
Fluxo Alternativo		
F001	001	Tenta novamente após 60 segundos
RF010	**Exibir Dados no Dashboard**	
Ator(es)	App Mobile (Interno); App Mobile (Externo)	
Pré-condição	Ter recebido uma publicação;	
Pós-Condição	Exibição dos dados recebidos na tela	
Cenário de Sucesso Principal		
S001	Executa o [RF009]	
S002	Organiza as mensagens por canal	
S003	Incorpora a estrutura de dados do dashboard	
Exceções		
Fluxo Alternativo		

RF011	**Persistir Informações no Banco de Dados**	
Ator(es)	Gateway	
Pré-condição	Nenhuma	
Pós-Condição	Informações salvas no banco de dados	
Cenário de Sucesso Principal		
S001	Abre conexão como banco de dados	E001
S002	Persiste os dados no banco de dados	
S003	Encerra a conexão	
Exceções		
E001	Falha na conexão com o banco de dados	F001
Fluxo Alternativo		
F001	001	Tenta novamente após 60 segundos

RF012	**Receber Mensagem Publicada**	
Ator(es)	Message Broker	
Pré-condição	Servidor MQTT instalado	
Pós-Condição	Publicação recebida	
Cenário de Sucesso Principal		
S001	Estabelecer conexão com o cliente	
S002	Recebe a publicação	
S003	Executa o [RF013]	
S004	Executa o [RF015]	
Exceções		
Fluxo Alternativo		

RF013	Criar Canais	
Ator(es)	Message Broker	
Pré-condição	Servidor MQTT instalado	
Pós-condição	Canal criado	
Cenário de Sucesso Principal		
S001	Criar canal de acordo com o especificado na mensagem	E001
Exceções		
E001	O canal já existe	F001
Fluxo Alternativo		
F001	001	Não cria o canal
RF014	**Registrar Assinantes dos canais**	
Ator(es)	Message Broker	
Pré-condição	Servidor MQTT instalado	
Pós-Condição	Publicações enviadas para os assinantes dos canais	
Cenário de Sucesso Principal		
S001	Estabelecer conexão com o cliente	
S002	Armazenar o cliente na lista de assinantes do canal informado	
Exceções		
Fluxo Alternativo		
RF015	**Encaminhar Publicações para Assinantes**	
Ator(es)	Message Broker	
Pré-condição	Servidor MQTT instalado	
Pós-Condição	Publicações enviadas para os assinantes dos canais	
Cenário de Sucesso Principal		

S001	Consultar lista de assinantes do canal	
S002	Despacha a mensagem para os assinantes	
Exceções		
Fluxo Alternativo		

RF016	**Visualizar Dados**	
Ator(es)	Usuário	
Pré-condição	App instalado em um dispositivo android	
Pós-Condição	Visualizar informações do monitoramento	
Cenário de Sucesso Principal		
S001	O usuário acessa o app	
S002	O usuário seleciona a opção dashboard	
S003	O sistema exibe as informações de monitoramento	E001
Exceções		
E001	O sistema não exibe os dados do monitoramento	F001
Fluxo Alternativo		
F001	001	O sistema exibe a mensagem “Falha ao tentar carregar as informações”
	002	O sistema tenta carregar novamente os dados de monitoramento

RF017	**Criar Alertas**	
Ator(es)	Usuário	
Pré-condição	App instalado em um dispositivo android	
Pós-Condição	Alerta para de monitoramento criado	
Cenário de Sucesso Principal		
S001	O usuário acessa o app	
S002	O usuário seleciona a opção criar alerta	

S003	O sistema disponibiliza a lista dos elementos monitorados	
S004	O usuário seleciona um elemento	
S005	O sistema exibe um campo para a inserção do valor a ser alertado	E001
S006	O usuário informa o valor e clica em confirmar	E002
S007	O sistema cria o alerta	
S008	O sistema exibe a mensagem “Alerta criado com sucesso”	
Exceções		
E001	O valor é inserido é inválido	F001
E002	Já existe um alerta com o mesmo valor para o elemento selecionado	F002
Fluxo Alternativo		
F001	001	O sistema exibe a mensagem “valor inválido”
F002	001	O sistema exibe a mensagem “já existe um alerta com o valor informado para esse elemento”