

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO**

JOÃO PAULO RIZZO LOPES

**PROPOSTA DE MONITORAMENTO DE UMA CENTRAL
DE DETECÇÃO E ALARME DE INCÊNDIO VIA
COMPUTAÇÃO EM NUVEM**

VITÓRIA
2020

JOÃO PAULO RIZZO LOPES

**PROPOSTA DE MONITORAMENTO DE UMA CENTRAL DE
DETECÇÃO E ALARME DE INCÊNDIO VIA COMPUTAÇÃO EM
NUVEM**

Parte manuscrita de Projeto de Graduação do aluno **João Paulo Rizzo Lopes**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Eng. Esp. Heliomar Guimarães Guzzo

VITÓRIA
2020

JOÃO PAULO RIZZO LOPES

**PROPOSTA DE MONITORAMENTO DE UMA CENTRAL DE
DETECÇÃO E ALARME DE INCÊNDIO VIA COMPUTAÇÃO EM
NUVEM**

Parte manuscrita de Projeto de Graduação do aluno **João Paulo Rizzo Lopes**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito para obtenção do grau de Engenheiro Eletricista.

Aprovada em 14 de dezembro de 2020.

COMISSÃO EXAMINADORA:

Prof. Esp. Heliomar Guimarães Guzzo
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. André Ferreira
Universidade Federal do Espírito Santo
Examinador

Esp. Cláudio Antunes de Oliveira
ArcelorMittal Tubarão
Examinador

AGRADECIMENTOS

Agradeço primeiramente por estar vivo. Agradeço a meu pai e minha mãe pela minha criação e pela prioridade que sempre deram à minha educação, sempre provendo as condições necessárias para isso. Agradeço ao meu orientador pelos ensinamentos, que foram extremamente valiosos, tanto para a conclusão deste trabalho quanto para minha vida profissional. Agradeço aos profissionais do setor de automação da ArcelorMittal Tubarão pelo aprendizado que tive e à gerência pela permissão de realização deste projeto.

*"Once you bid farewell to discipline, you say
goodbye to success."*

(Alex Ferguson)

RESUMO

No presente texto apresenta-se o projeto de graduação desenvolvido para integralização do curso de Engenharia Elétrica da Universidade Federal do Espírito Santo. Neste trabalho, que foi embasado por uma ampla pesquisa bibliográfica, são versados temas referentes a sistemas de detecção e alarme de incêndio, Internet das Coisas, computação em nuvem, e *Business Intelligence* (BI). Expõe-se, a princípio, a demanda concreta existente na empresa ArcelorMittal Tubarão de monitoramento de determinadas variáveis relativas às Centrais de Alarmes de Incêndio, e, em seguida, é discorrido sobre como foi proposto enfrentar este problema utilizando-se modernas tecnologias à disposição e seguindo princípios da Indústria 4.0. O enfoque deste trabalho, além do sensoriamento, é a integração de serviços de computação em nuvem com aplicações corporativas de *Business Intelligence*, desenvolvendo uma interface de visualização dos dados. Como resultado final, foi construído um protótipo designado tanto para este monitoramento quanto para integração com o *Microsoft Azure* e o *Power BI*, ferramenta utilizada para o *dashboard* de tempo real. Como desfecho da solução proposta, foram feitos os devidos testes de *software* e validação das medições, seguidos de uma análise qualitativa.

Palavras-chave: Sistema de detecção e alarme de incêndio. Internet das Coisas. Indústria 4.0. *Business intelligence*.

ABSTRACT

This text presents an academic project developed as a requirement to conclude the Electrical Engineering course at the Universidade Federal do Espírito Santo. This work, which was based on an extensive bibliographic research, themes related to Fire Detection and Alarm Systems, Internet of Things, cloud computing and Business Intelligence (BI) are presented. Initially, the actual demand from company ArcelorMittal Tubarão of monitoring certain variables related to Fire Alarm Control Panels is reported, and, after that, the proposal to how this problem could be solved using modern technologies at disposal following the principles of Industry 4.0 is discussed. The focus of this work, in addition to the sensing, is the integration of cloud computing services with Business Intelligence corporate applications, developing a data visualization interface. As a final result, a prototype was built with the purpose of both monitoring and implement integration with Microsot Azure and Power BI, a tool used for the real time dashboard. As an outcome to the proposed solution, were made proper software tests and measurements validations, followed by a qualitative analysis.

Keywords: Fire detection and alarm system. Internet of Things. Industry 4.0. Business intelligence.

LISTA DE FIGURAS

Figura 1 – Fluxo simplificado da produção do aço	11
Figura 2 – Exemplo de sistema de detecção convencional	17
Figura 3 – Exemplo de sistema de detecção endereçável.....	18
Figura 4 – Fontes de alimentação de uma central convencional	19
Figura 5 – Protocolos usuais de aplicações IoT	23
Figura 6 – Mudanças de paradigma na computação	26
Figura 7 – <i>Cloud, fog e edge computing</i>	28
Figura 8 – Quadrante da Gartner Inc. referente a plataformas de BI e <i>Analytics</i>	31
Figura 9 – Quadrante da Gartner Inc. referente a infraestrutura de nuvem.....	31
Figura 10 – Modelo de referência recomendado para aplicativo IoT no <i>Microsoft Azure</i> usando componentes PaaS.....	33
Figura 11 – <i>Raspberry Pi 3 B+</i>	34
Figura 12 – <i>Arduino Uno</i>	34
Figura 13 – Evolução e estado atual das tecnologias de comunicação	35
Figura 14 – Comparativo entre alcance e taxa de dados das tecnologias de comunicação	36
Figura 15 – Arquitetura de aplicação do projeto – modelo funcional.....	37
Figura 16 – Montagem do protótipo de testes	39
Figura 17 – Fluxograma do <i>software</i> produzido	40
Figura 18 – Interface da plataforma de nuvem para cadastro do dispositivo.....	41
Figura 19 – Interface da plataforma de nuvem com propriedades do dispositivo.....	42
Figura 20 – Interface da plataforma de nuvem para configurações de pontos de extremidade internos	42
Figura 21 – Interface da plataforma de nuvem para inserção de <i>query</i>	43
Figura 22 – Interface do <i>Power BI</i> para configuração dos dados de <i>streaming</i>	44
Figura 23 – <i>Dashboard</i> criado no <i>Power BI</i>	47

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
AMS	<i>Asset Management System</i>
AMT	ArcelorMittal Tubarão
AMQP	<i>Advanced Message Queueing Protocol</i>
BI	<i>Business Intelligence</i>
CCPM	<i>Critical Chain Project Management</i>
ERP	<i>Enterprise Resource Planning</i>
FAT	<i>Factory Acceptance Test</i>
FEL	<i>Front-End Loading</i>
HTTP	<i>Hypertext Transfer Protocol</i>
HTTPS	<i>Hypertext Transfer Protocol Secure</i>
IaaS	<i>Infrastructure as a Service</i>
IIoT	<i>Industrial Internet of Things</i>
IoT	<i>Internet of Things</i>
JSON	<i>JavaScript Object Notation</i>
LPWAN	<i>Low-Power Wide-Area Network</i>
MES	<i>Manufacturing Execution System</i>
MQTT	<i>Message Queuing Telemetry Transport</i>
NBR	Norma Brasileira
PaaS	<i>Platform as a Service</i>
PIMS	<i>Plant Information Management System</i>
SaaS	<i>Software as a Service</i>
SAT	<i>Site Acceptance Test</i>
SBC	<i>Single Board Computer</i>
SDAI	Sistema de Detecção e Alarmes de Incêndio

SUMÁRIO

1	INTRODUÇÃO.....	11
1.1	Apresentação e Metodologia.....	11
1.2	Justificativa e Motivação do Tema.....	13
1.3	Objetivos.....	15
1.3.1	Objetivo Geral.....	15
1.3.2	Objetivos Específicos.....	15
1.4	Estrutura do Trabalho.....	16
2	EMBASAMENTO TEÓRICO.....	17
2.1	Sistema de Detecção e Alarme de Incêndio.....	17
2.2	Integração de Sistemas.....	19
2.3	Internet das Coisas.....	22
2.4	Computação em Nuvem.....	26
3	DESENVOLVIMENTO.....	30
3.1	<i>Benchmarking</i> e Definições de Projeto.....	30
3.2	Implementação.....	37
4	ANÁLISES DOS TESTES E RESULTADOS.....	45
5	CONCLUSÃO E TRABALHOS FUTUROS.....	48
	REFERÊNCIAS BIBLIOGRÁFICAS.....	50
	APÊNDICE A – DIAGRAMA DO PROJETO (PROTÓTIPO IMPLEMENTADO).....	54
	APÊNDICE B – DIAGRAMA DO PROJETO (ESCOPO ORIGINAL).....	56
	APÊNDICE C – CÓDIGO ELABORADO (ARDUINO).....	58
	APÊNDICE D – CÓDIGO ELABORADO (RASPBERRY PI).....	60
	APÊNDICE E – CÓDIGO DE TESTES.....	63
	APÊNDICE F – CADERNO DE TESTES EM BANCADA.....	66

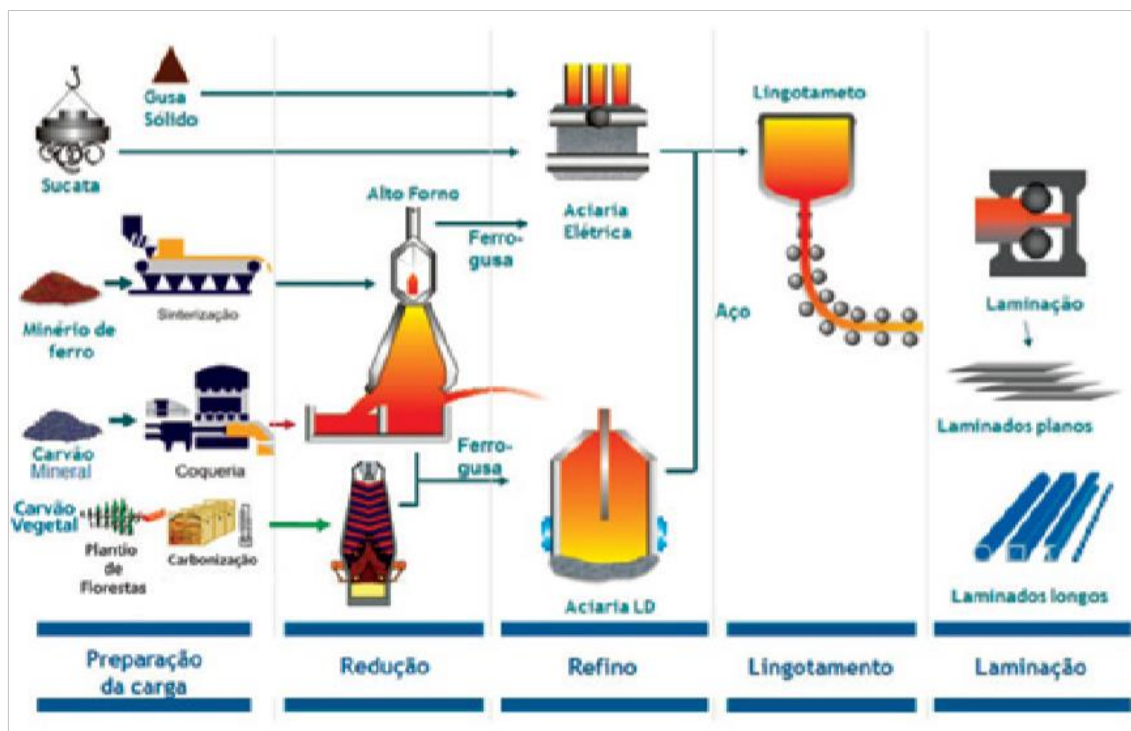
APÊNDICE G – CRONOGRAMA DO PROJETO.....	67
--	-----------

1 INTRODUÇÃO

1.1 Apresentação e Metodologia

O setor siderúrgico brasileiro é formado por empresas de grande porte, que operam diversas fases do seu processo produtivo, desde a transformação do minério de ferro em ferro-gusa até a produção de aço e coprodutos. Usinas integradas, as quais possuem plantas de coqueria, sinterização e alto-forno, têm como característica possuírem extensa dimensão devido a comportarem todas essas etapas na fabricação do aço, em contraste com as semi-integradas que compreendem apenas as três últimas etapas, ou seja, refino, lingotamento e laminação, conforme mostra a Figura 1 (CONFEDERAÇÃO NACIONAL DA INDÚSTRIA, 2012).

Figura 1 – Fluxo simplificado da produção do aço



Fonte: Confederação Nacional da Indústria (2012).

Esse tipo de usina, por via de regra, compreende instalações que são caracterizadas como ambientes de alta periculosidade, por envolver fatores que, no caso de acidentes, podem ocasionar grandes danos materiais e a vidas humanas (DAMASCENO, 2014). Um dos possíveis riscos é a decorrência de incêndios e, devido a isso, existem sistemas e centrais de detecção e

alarme de incêndios, inseridos por toda a extensão das usinas siderúrgicas (XAVIER FILHO, 2010).

Na ArcelorMittal Tubarão (AMT) isto não é diferente e a empresa tem grande apreço por padrões e medidas de segurança, visando sempre à melhoria contínua dos procedimentos e tecnologias que asseguram a perenidade da planta e o bem-estar dos seus colaboradores. Neste trabalho, pretende-se implementar uma solução para uma demanda concreta presenciada durante período de estágio realizado pelo aluno no setor de Automação da AMT.

O contexto encontrado é de que existem atualmente dois sistemas de detecção e alarme de incêndio (SDAI) funcionando na AMT, sendo um classificado como convencional e outro como endereçável, conforme a Norma Brasileira (NBR) da Associação Brasileira de Normas Técnicas (ABNT) ABNT NBR 17240:2010. A diferenciação entre os sistemas de detecção convencional e endereçável será abordada de maneira mais precisa na seção 2.

As centrais endereçáveis foram implantadas para substituir as convencionais, as quais são mais antigas. No entanto, devido ao ambiente agressivo ao qual podem estar sujeitas (RIBEIRO, 2014), e também aos custos consideráveis que a migração para o novo sistema implicaria, esta migração não foi realizada para a totalidade das centrais convencionais. Diante disso, ficou evidente que seria necessário proporcionar uma forma de acrescentar nas centrais convencionais alguns recursos hoje existentes nas endereçáveis.

Para as centrais convencionais, existe a demanda de monitoramento de alguns de seus aspectos operacionais, como: a tensão de alimentação primária, proveniente da rede elétrica principal, a tensão de alimentação secundária, que vem de baterias, e a temperatura destas, para citar apenas os imprescindíveis. Dessa maneira, uma ideia inicialmente estudada pelos profissionais do setor de Automação da AMT, e adaptada no contexto deste trabalho, se deu com base nessa premissa.

Além de agilizar a rastreabilidade dos aspectos anteriormente mencionados, outro requisito deste projeto é o monitoramento das variáveis citadas através telas de tendência de tempo real e tendência histórica, entre outras análises possíveis.

Assim sendo, vê-se que as necessidades práticas deste projeto indicavam que o empreendimento a ser realizado poderia ser implantado segundo os conceitos de Indústria 4.0 e *Internet of Things* (IoT), a Internet das Coisas. E, dessa maneira, foi feito, definindo-se no escopo do projeto que os dados de telemetria seriam pré-processados por um *Raspberry Pi* e com transmissão *wireless* para um serviço de computação em nuvem que, por sua vez, estabeleceria comunicação com o destino final, um *dashboard* de monitoramento criado com a ferramenta *Power BI*, pelo qual os dados seriam acompanhados.

Desta forma, em linhas gerais, o desenvolvimento desse projeto executado na disciplina de Projeto de Graduação, encontra-se descrito neste tópico introdutório. Observando-se adaptações de ferramentas e metodologias segundo as circunstâncias encontradas, deu-se continuidade ao projeto anteriormente em andamento na AMT, principalmente em se tratando do desenvolvimento de um sistema para monitorar as variáveis de interesse e transferir os dados através da nuvem (*Microsoft Azure*) e disponibilizá-los em um sistema de informação. Ao fim, avalia-se se os resultados finais do projeto atenderam aos requisitos operacionais e identificam-se possíveis oportunidades de melhorias para trabalhos futuros.

Este trabalho, pode ser classificado como sendo uma pesquisa aplicada, tendo em vista que se concentra em torno de problemas reais existentes nas atividades de uma empresa, e no empenho na elaboração de uma investigação em busca de soluções capazes de gerar impacto. Ao mesmo tempo, pode-se entender este projeto dentro do conceito de pesquisa exploratória, visto que a abordagem de internet das coisas direcionada a este tipo de sistema de detecção e alarme de incêndio foi minimamente explorada até o momento, e com este projeto foram validados alguns instrumentos. Por fim, como procedimentos técnicos, foram utilizados pesquisa bibliográfica para embasamento, conjuntamente com pesquisa experimental, já que a montagem de um protótipo e a realização de testes constituíram interferência ativa na situação sob estudo, e a abordagem do problema se deu de forma quantitativa.

1.2 Justificativa e Motivação do Tema

A integração de fato da gestão da Saúde e Segurança no Trabalho à gestão da produção é um dos maiores desafios que as empresas têm deparado. Segundo Haukelid (2008), dentre as principais razões que levam as organizações a promover esta integração, destacam-se a

mentalidade de que cultura de segurança não deve ser algo separado ou em adição à cultura organizacional, mas constitui uma parte integrada desta.

Os acidentes de trabalho são causados por atos inseguros ou por condições inadequadas (SILVA, 2006), dentre as quais pode-se destacar a falta de uma rotina de inspeção, que ao ser implementada promove melhorias em vários sistemas e equipamentos, e é de solução acessível. Em vista disso, quaisquer procedimentos e sistemas que proporcionem um maior grau de segurança, ou mesmo a consciência da existência de fatores de risco, e paralelamente preserve o investimento em ativos, são de grande valor para as organizações, e o monitoramento de variáveis de processo é um agente promotor dessas melhorias.

Na unidade industrial da ArcelorMittal Tubarão, existem várias fontes de riscos às quais os trabalhadores estão expostos. Os riscos são mapeados e ações preventivas são estabelecidas para minimização e/ou eliminação dos mesmos, em consonância com políticas corporativas. Dentre eles, destacam-se: ruído, equipamentos móveis, cargas suspensas, calor, aço líquido, etc. Os dois últimos itens são bons exemplos de potenciais causadores de incêndios no ambiente siderúrgico.

No que se refere à gestão do processo produtivo, duas ferramentas de suporte são igualmente importantes: o gerenciamento de ativos (AMS, do inglês *Asset Management System*) e de informação de planta (PIMS, do inglês *Plant Information Management System*). Tais ferramentas, entretanto, necessitam de que informações de campo sejam consolidadas e integradas a todo o processo produtivo.

Em linhas gerais, o gerenciamento de ativos tem por objetivo a saúde dos ativos da empresa, em particular os sensores e equipamentos de campo. Se inteligentes, podem se comunicar com o sistema de gerenciamento, dotando-o de informações que permitem a este definir performance, tempo de funcionamento e necessidade de intervenção dos elementos gerenciados, etc. Isto traz uma série de benefícios, tanto para área de manutenção quanto para a de operação.

Quanto ao gerenciamento de informações de planta, ressalta-se que se trata de uma ferramenta imprescindível nas análises de falha, de processo, pela agilidade com que fornece informações tanto aos que operam quanto àqueles que gerenciam em um nível mais operacional.

No que se refere à gestão de processos, tanto do setor industrial quanto fora dele, conceitos relativamente recentes têm direcionado esforços na efetivação de processos mais abrangentes, eficientes e multidisciplinares, derrubando fronteiras antes consolidadas. Assim, tem sido comum no jargão desta área termos como: *cloud computing*, *edge computing*, *Internet of Things*, *Information Technology/Operational convergence*, *digital transformation*, *Business Intelligence*. Tais conceitos estão reunidos em um mais generalizado denominado Indústria 4.0, sendo que alguns destes serão abordados neste trabalho.

Considerando que um sistema de detecção e alarme de incêndios tem íntima relação com as questões de segurança, se faz necessário que tenha suas variáveis principais devidamente monitoradas e gerenciadas como um ativo de grande valor estratégico. Assim, este trabalho se propõe a monitorar estas variáveis, definidas pelos profissionais da AMT, integrá-las ao seu sistema de informações de planta, usando tecnologias enquadradas no conceito do Indústria 4.0.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é promover a capacidade de monitoramento remoto em tempo real das variáveis mais importantes de uma Central de Detecção e Alarme de Incêndio, ao mesmo tempo em que se integra os valores sensorizados a uma plataforma capaz de historiar dados, realizar inferências e construir uma visualização, entre outras possibilidades, da mesma forma como é feita em um sistema de gerenciamento de informações de planta.

1.3.2 Objetivos Específicos

Os objetivos específicos a seguir foram perseguidos:

- Projetar o *hardware* necessário e a eletrônica complementar para condicionamento e monitoramento de variáveis de processo e transmissão para a nuvem;

- Desenvolver o *software* de processamento, serializar e posteriormente enviar mensagens de telemetria para a nuvem;
- Configurar a plataforma de computação em nuvem;
- Criar tela de interface com o usuário para monitoramento das variáveis;

1.4 Estrutura do Trabalho

Além do que já foi mencionado, o trabalho foi estruturado conforme a seguir:

Na seção 2, serão abordados os conceitos básicos necessários para fundamentação teórica deste projeto, com enfoque em aspectos técnicos relevantes.

Na seção 3, serão discutidos tópicos a respeito da implementação efetivamente utilizada para o desenvolvimento deste projeto visto seu uso como piloto, assim como os pormenores referentes às estratégias tecnológicas e respectivos componentes.

Já na seção 4, foi efetuada a validação do protótipo construído por meio de testes de bancada.

Por fim, na seção 5, serão discutidas as conclusões obtidas e as possíveis propostas de continuidade do trabalho.

2 EMBASAMENTO TEÓRICO

2.1 Sistema de Detecção e Alarme de Incêndio

O propósito de um SDAI, de acordo com a ABNT NBR 7240-1:2017, é “detectar o incêndio no menor tempo possível e dar um alarme de forma que uma ação apropriada possa ser tomada” (ABNT, 2017, p. 1). Trata-se de um sistema de extrema importância em um ambiente industrial, tanto para resguardo dos bens materiais quanto para prevenção de danos à saúde pessoal dos empregados (RIBEIRO, 2014). Fazem parte da composição obrigatória de um SDAI, detectores de incêndio, dispositivos sinalizadores de alarme, centrais de controle e indicação e fontes de alimentação. Outra norma de interesse deste trabalho é a ABNT NBR 17240:2010, que afirma:

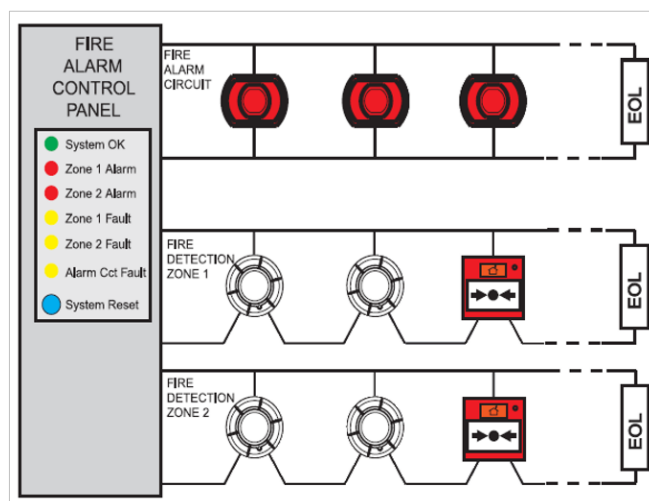
Quanto à alimentação elétrica, a central deve possuir sempre uma fonte de alimentação principal e uma de emergência, com capacidades iguais e tensão nominal de 24 Vcc. As fontes de alimentação devem ser supervisionadas e dimensionadas para o consumo máximo do sistema (ABNT, 2010, p. 33).

Também é interessante conceituar os sistemas convencional e endereçável conforme esta mesma norma, na qual consta o seguinte.

- Sistema convencional (Figura 2):

Sistema composto por um ou mais circuitos de detecção. Cada circuito de detecção é instalado em uma determinada zona ou área protegida. Quando atuado um dispositivo de detecção, a central identifica somente a área protegida pelo circuito de detecção onde o dispositivo está instalado [...] (ABNT, 2010, p. 7).

Figura 2 – Exemplo de sistema de detecção convencional

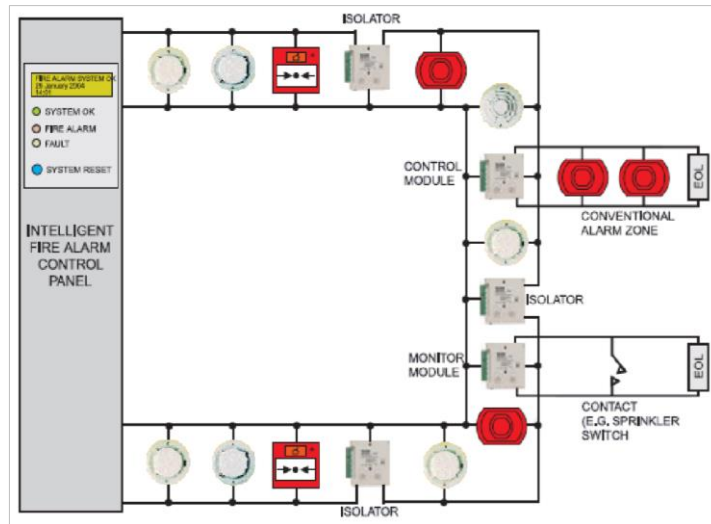


Fonte: Moreira (2010).

- Sistema endereçável (Figura 3):

Sistema composto por um ou mais circuitos de detecção. Cada dispositivo de detecção recebe um endereço que permite à central identificá-lo individualmente. Quando atuado um dispositivo de detecção, a central identifica a área protegida e o dispositivo em alarme [...] (ABNT, 2010, p. 8).

Figura 3 – Exemplo de sistema de detecção endereçável

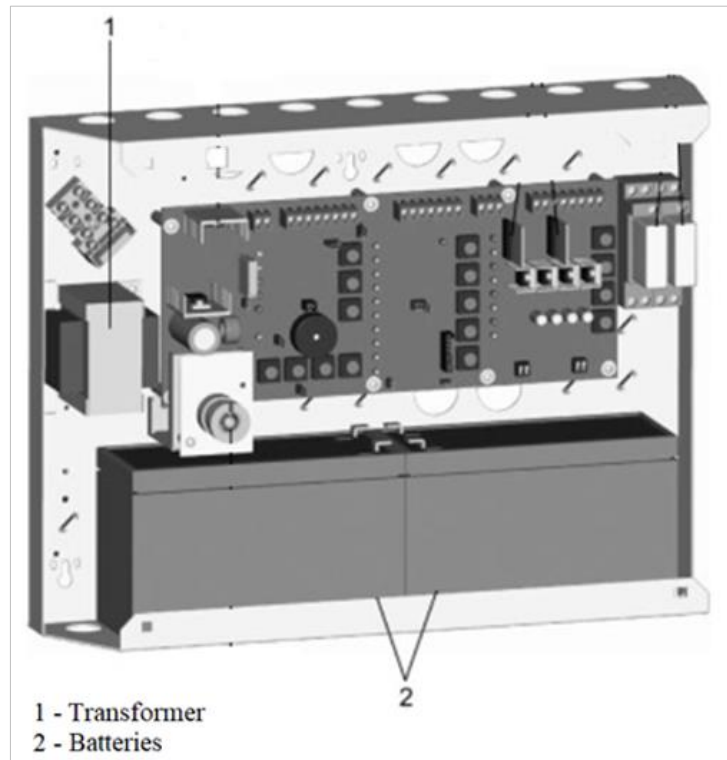


Fonte: Moreira (2010).

A principal diferença entre os dois sistemas reside, portanto, na capacidade de localização precisa de um eventual alarme pelo sistema dito endereçável, permitindo identificar cada detector individualmente, enquanto que o sistema convencional apenas estabelece uma indicação por zona. Geralmente as zonas de alarme são atribuídas a andares e/ou setores por inteiro. Os sistemas convencionais são mais simples, e, portanto, de menor custo do que os endereçáveis, e assim, em instalações de menor porte ou de menor compartimentação encontram sua maior aplicabilidade sem comprometer o grau de segurança (AGÊNCIA NACIONAL DE VIGILÂNCIA SANITÁRIA, 2014; JUSTA, 2016).

Na Figura 4 observa-se uma ilustração coletada de um catálogo do fabricante Siemens, destacando os principais componentes de uma central convencional. Os de interesse específico deste projeto sendo os identificados com os números 1 e 2, os quais são referentes à alimentação principal e secundária (baterias), respectivamente.

Figura 4 – Fontes de alimentação de uma central convencional



Fonte: Siemens (2010).
Nota: Adaptado pelo autor.

Por se tratar de um sistema em que segurança e confiabilidade são itens de fundamental importância, as interfaces com o SDAI devem ser tal que não interfiram em seu funcionamento e esta foi uma premissa deste projeto. Como exemplo, pode-se citar sistemas de Circuito Fechado de Televisão, que permite ao usuário do SDAI visualizar através de câmeras eventos relacionados com o SDAI sem interferir nele.

2.2 Integração de Sistemas

O panorama corporativo tem presenciado uma evolução tecnológica contínua com o passar dos anos. Isto se traduz na existência concomitante de tecnologias com diferentes graus de maturidade, umas antigas, outras modernas. As antigas são denominadas de sistemas legados, termo usado para designar aplicações tecnologicamente desatualizadas, mas que continuam em operação em uma organização, seja pelo seu grau de criticidade, custo de modernização, dentre outras razões (CUMMINS, 2002). Esta constatação é válida para diversos segmentos tecnológicos, desde sistemas de informação e de automação, incluindo o objeto deste trabalho,

os sistemas de detecção e alarme de incêndio. Esta evolução citada acabou por contribuir também para a fragmentação dos sistemas, na medida em que aqueles mais antigos se tornam desatualizados e de difícil integração com os mais novos. Uma dificuldade adicional acontece quando se adota a integração entre sistemas diferentes através de soluções setorizadas conforme as necessidades específicas percebidas.

Também se constata que o monitoramento de diversos sistemas e ativos muito distintos entre si de forma simultânea, frequentemente implica em uma execução rudimentar e descentralizada (BITTENCOURT; OLIVEIRA, 2017). Seja no ambiente de Tecnologia da Informação ou de Tecnologia da Automação, a vasta quantidade de dispositivos e aplicações heterogêneos, habitualmente constitui um desafio tratar a integração entre esses elementos para um melhor usufruto da informação.

Visando o constante aperfeiçoamento da eficiência operacional, existe uma demanda recente pelo acesso a informações e dados nas indústrias para que mais rapidamente se responda a problemas e se identifique oportunidades. Portanto, a integração de sistemas deve fornecer uma estrutura em que, até mesmo sistemas legados, possam continuar operando de maneira apropriada e, sobretudo, integrados às novas tecnologias e sistemas presentes nas empresas, pensada, não como uma transformação única, mas sim, como um processo de melhoria contínua, permitindo alterações incrementais nos processos, nas aplicações e na tecnologia de suporte. Urge ainda que exista uma visão gerencial voltada para o progresso tecnológico não apenas de maneira responsiva a forças externas propulsoras de alterações, mas também como resultado de uma autoanálise e uma avaliação de novas ideias para tornar a empresa mais competitiva em todos os seus níveis.

É evidente que sistemas integrados usando uma mesma tecnologia e aderentes a padrões têm precedência em relação a sistemas legados e/ou fragmentados, pois proporcionam benefícios importantes como economia de escala, produtividade no desenvolvimento, capacidade de reusabilidade das aplicações, entre outros. Há que se ressaltar, no entanto que a tomada de decisão que privilegie tais sistemas seja suportada pelo devido retorno do investimento.

Cumprе ressaltar que o sistema de informações de planta (PIMS) citado anteriormente tem papel fundamental na integração e consolidação de informações de um processo operacional,

facilitando a análise de dados, resgatando o histórico de dados, confecção de relatórios, análise de ocorrências, etc, em resumo, transformando dados em informação.

No que se refere à gestão de uma empresa é comum o uso de sistemas integrados de gestão empresarial, dentro de uma categoria de produtos denominada de *Enterprise Resource Planning* (ERP). Tais sistemas tratam da gestão de processos, vendas, finanças, contabilidade, controle de estoque, compras, produção e logística. Ocorre que a linguagem normalmente usada nessa categoria é relativamente diferente da linguagem do chão de fábrica, como por exemplo, a do PIMS. A solução encontrada foi, em linhas gerais, um terceiro sistema denominado *Manufacturing Execution System* (MES) que faz esta ligação, detalhando melhor os requisitos determinados na gestão empresarial para que sejam melhor entendidas as demandas alocadas ao processo fabril (LIMA, 2019). O MES, assim, trata de itens como: planejamento e programação da produção, controle e análise da produção, controle estatístico de processos, controle de qualidade, rastreabilidade, controle de insumos, etc.

Cumprе ressaltar, entretanto que os temas até agora citados neste item pertencem a uma arquitetura que pode ser considerada como convencional. Ocorre que uma nova arquitetura se encontra em andamento, a partir de um paradigma recente chamado Indústria 4.0 e que se traduz em uma nova visão a respeito dos processos de gerenciamento estratégico e operacional em busca de maiores benefícios quanto a redução de custos, qualidade, visibilidade, competitividade, obviamente impulsionados pela evolução tecnológica como as já citadas *cloud computing*, *Internet of Things*, *edge computing*. Dentro deste contexto emerge como um novo conceito de gestão empresarial denominado *Business Intelligence* (BI). O BI engloba metodologias, tecnologias e processos, com a finalidade de proporcionar suporte à tomada de decisão nas empresas. Extraem dados mais precisos e propiciam agilidade às operações das organizações. Têm capacidade de obter informações de diversas procedências, examinar séries temporais, e inclusive empregar métodos de verificação de falhas (LESSAK, 2018).

Atualmente o BI, é visto como uma grande tendência para resolver a falta de integração de vários sistemas no mesmo ambiente empresarial, que, pelo desacoplamento, produzem poucos *insights* de qualidade para o negócio (LESSAK, 2018; WATSON; WIXON, 2007). Essa reputação deve-se à capacidade das ferramentas de BI, que com o uso de relatórios dinâmicos e análise de tendências, e diversos outros recursos e técnicas, frequentemente são capazes de

economizar tempo, realizar uma entrega de dados mais eficiente e, conseqüentemente, fornecer informações para o nível estratégico das organizações.

2.3 Internet das Coisas

É fato de conhecimento geral que a evolução da internet em um espaço consideravelmente curto de tempo impactou de forma significativa os mais diversos setores de negócios, acabando por se tornar a base das aplicações utilizadas em muitos casos. A infraestrutura e a tecnologia da internet permitem aumentar a capacidade da automação e integração dos processos corporativos. Nos esforços de reengenharia, geralmente os sistemas computacionais são vistos como mecanismos para melhorar a produtividade dos processos das empresas.

Em se tratando de Internet das Coisas, seu significado reside na interconexão dos mais diversos objetos físicos, equipamentos e máquinas na rede internet, não apenas no ambiente industrial, mas, quando este é o caso, o termo *Industrial Internet of Things* (IIoT) é empregado (BOYES *et al.*, 2018).

Em relação ao volume de dados, aplicações em IoT têm maior abrangência e variabilidade em comparação com a IIoT. Na IIoT o objetivo essencial é realizar análise de dados para as diversas finalidades da indústria, como melhoramentos na logística e de cadeia de suprimentos, gerenciamento estratégico, melhorias operacionais, entre outras, o que também resulta em um significativo volume de dados.

Ainda não existe um grande consenso na comunidade científica e tecnológica sobre como definir a arquitetura padrão da IoT ou da IIoT. A abordagem geral têm sido a representação em multicamadas, com a diferença entre as propostas residindo na sua quantidade (SETHI; SARANGI, 2017). No entanto, algo que é intrínseco do conceito de IoT é o fato de que a primeira destas camadas serem os sensores e *edge computing*. Estes elementos são os que fundamentalmente proporcionam a ligação entre equipamentos e usuários, uma vez que o controle de processos pode ser executado pelo *edge computing*.

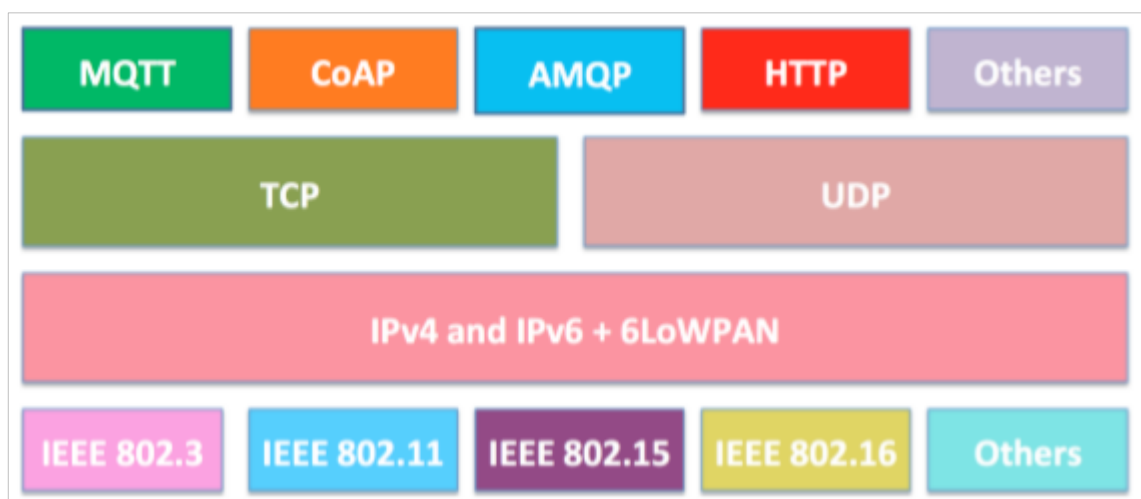
Ao cunhar o termo Internet das Coisas, expressamente estabelecendo relação com a Internet como a conhecemos, é de se imaginar que as “coisas” terão comportamentos e funcionalidades

que sejam uma evolução do que se percebe na rede de computadores atual. (SANTOS *et al.*, 2017). Entretanto, do surgimento da IoT decorreu a modificação do conceito de redes de computadores, uma vez que uma rede não se limita mais a computadores, dada a quantidade de dispositivos não convencionais que também participam da rede de comunicação.

Realmente já existe a necessidade de adotar novos protocolos e tecnologias de infraestrutura de comunicação, ou adaptar os existentes, por exemplo, para se adequar aos novos paradigmas. Para tal, entretanto, requerem-se considerações meticulosas, pois geralmente existe um propósito de integração que motiva a aplicação destes novos processos (XU; DAVID; KIM, 2018). Portanto, para um melhor entendimento do processo de elaboração deste projeto, é primordial destacar alguns conceitos que conduziram ao seu desenvolvimento.

A tecnologia de comunicação é parte essencial para o desenvolvimento de aplicações em IoT. Entretanto, a seleção do protocolo mais apropriado para tal é uma tarefa complexa, pois depende da natureza dos requisitos de cada sistema. Vários protocolos existem e são aplicados atualmente (Figura 5), mas nenhum deles é apontado como sendo superior para todas as aplicações, se tratando sempre de um *trade-off*. Devido a este dilema, é necessário compreender os prós e contras de cada possibilidade, para poder determinar com assertividade qual o mais pertinente para cada aplicação (NAIK, 2017).

Figura 5 – Protocolos usuais de aplicações IoT



Fonte: Naik (2017).

Considerando o modelo *Open System Interconnection*, que abstrai as redes de comunicação em 7 camadas, devemos observar atentamente as possibilidades dos protocolos para a IoT da camada de aplicação. Neste trabalho, convém discorrer brevemente sobre os protocolos usados pelo recurso de computação em nuvem escolhido, no caso, o *Azure IoT Hub*. Segundo a documentação disponível *online* (MICROSOFT AZURE, 2018a), estes protocolos seriam o *Hypertext Transfer Protocol* (HTTP), o *Message Queuing Telemetry Transport* (MQTT) e o *Advanced Message Queueing Protocol* (AMQP).

O MQTT é um protocolo desenvolvido nos anos 90, bastante útil para aplicações IoT. Seus princípios arquitetônicos visam minimizar o uso de banda de rede, conjuntamente com a confiabilidade e garantia da entrega das mensagens de telemetria. Estes atributos fazem com que este protocolo seja conveniente para redes sem grande confiabilidade ou com limitação de recursos, como latência, intermitência de conexão ou largura de banda (MAZZER; FRIGIERI; PARREIRA, 2015).

O fato de ser fundamentado num modelo *publish-subscribe* permite que exista o envio e recebimento de mensagens de forma assíncrona, e dessa maneira, desacoplando o emissor e o receptor da mensagem, pois ambos podem iniciar uma conexão a qualquer momento para a transferência da mensagem por meio de um elemento intermediário denominado *broker*. O *broker* é uma espécie de mediador entre os dispositivos que se comunicam, promovendo o desacoplamento entre as partes. Uma mensagem enviada, é disponibilizada no *broker* para consulta pelos dispositivos nele subscritos, independentemente de estarem disponíveis no momento de envio da mensagem.

De acordo com Naik (2017), o MQTT é mais adequado no caso de grandes redes de pequenos dispositivos que também necessitem receber comandos de controle. Além disto, o MQTT não teve como prioridade central o aspecto segurança de informação quando foi projetado (MAZZER; FRIGIERI; PARREIRA, 2015). Como exemplo pode-se citar que na sua especificação não está definido o tipo de criptografia a ser utilizado, e no caso da não utilização em um ambiente controlado, a transmissão de credenciais de autenticação pode representar um sério risco.

O AMQP é um protocolo para *middleware* orientado a mensagens, com base em filas de mensagens e modelo publica-subscribe. Sua característica de *middleware* constitui-se na camada de *software* que realiza interfaceamento entre componentes possibilitando realizar comunicação entre os dispositivos e/ou aplicações (CARMO, 2012). Pode-se dizer sobre ele que:

“[...] foi projetado para ser seguro: nenhuma mensagem pode ser revelada ou alterada por outros; confiável: há meios de garantir a entrega da mensagem; aberto e padronizado: a especificação do protocolo está disponível para todos, então é possível que diferentes implementações do AMQP conversem entre si” (MAZZER; FRIGIERI; PARREIRA, 2015, p. 4).

Assim como o MQTT, o AMQP é um protocolo que é capaz de realizar comunicação assíncrona. A utilização deste protocolo no contexto de IoT é motivada pela grande disponibilidade de implementações dos *brokers* que permitem a interoperabilidade entre diversas plataformas incluindo sistemas legados. Devido ao AMQP ter sido concebido por um grupo de grandes empresas para atuar em seus negócios, ele é por característica muito sofisticado e complexo, portanto geralmente não é tão recomendado para pequenos sistemas e dispositivos limitados.

O *Hypertext Transfer Protocol* (HTTP) é um protocolo já extremamente difundido e globalmente aceito, para sistemas de informação distribuídos, sendo fundamental para a existência da internet como se conhece nos dias de hoje. O protocolo HTTP define métodos que indicam a ação a ser realizada. Neste trabalho será usado o método *POST*, que é empregado quando o cliente deseja enviar dados para processamento ao servidor, de maneira que este aceite os dados anexados no corpo da mensagem de requisição para armazenamento.

Para fins de segurança, foi criada a implementação *Hyper Text Transfer Protocol Secure* (HTTPS), que rigorosamente dizendo, é o próprio HTTP por cima de uma camada criptografada. O principal propósito do HTTPS é fornecer segurança mesmo em uma rede insegura. Com isso, se torna possível garantir aspectos de confidencialidade, integridade e autenticação. Frequentemente, usa-se conexões HTTPS em sistemas corporativos para lidar com transações importantes ou vulneráveis.

Neste projeto, são monitoradas as variáveis e a serialização destas é feita no formato JSON (*JavaScript Object Notation*). Seu conteúdo enviado para o serviço *IoT Hub* da *Microsoft Azure* pelo protocolo HTTPS. O JSON nada mais é do que um formato de arquivo simples, com campos de atributo e valor, utilizado para troca de informações entre sistemas (NURSEITOV *et al.*, 2009). Os atributos neste caso são os nomes das variáveis e um *timestamp*.

2.4 Computação em Nuvem

Conforme constataram Voas e Zhang (2009) e reafirmaram Furht e Escalante (2010) é possível identificar na história evolutiva da computação certas mudanças de paradigma, as quais estão divididas em seis fases, e podem ser ilustradas como na Figura 6.

Figura 6 – Mudanças de paradigma na computação



Fonte: Produção do próprio autor.

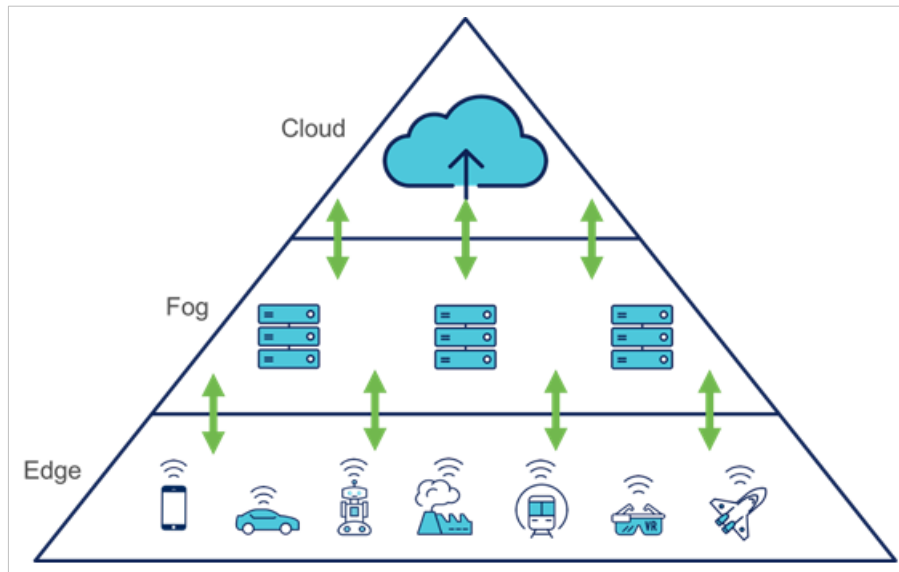
A Figura 6 dá uma boa ideia da evolução dos recursos computacionais. Na fase 1, *mainframes* concentravam todo o poder computacional, com usuários ligados através de periféricos. Na fase 2, dentro de um processo denominado de *downsizing* o poder computacional passou a ser mais distribuído migrando para computadores pessoais independentes sem a necessidade de um *mainframe*. A ideia do *downsizing* era transferir o poder computacional do *mainframe* para o computador pessoal. Já na fase 3, redes de computadores permitiram a conexão entre vários deles e assim compartilhar recursos. A fase 4, com o advento de redes locais foi possível a conexão a outras redes locais para estabelecer uma rede mais global - os usuários agora podiam se conectar à internet para utilizar aplicativos e recursos remotos. Na fase 5, já com o conceito de computação distribuída difundido, também se tornou possível que pessoas, a partir de seu computador pessoal pudessem acessar um *computational grid* orientado para aplicações. Finalmente, a fase 6, que caracteriza a computação em nuvem permite explorar todos os recursos de *hardware* e *software* na rede internet de maneira simples e escalável, que é também uma característica do *grid computation*. Uma diferença importante entre esses conceitos é que, o primeiro é mais orientado a aplicações, enquanto o segundo mais orientado a serviços (FURHT; ESCALANTE, 2010).

Fazendo parte do escopo deste projeto, a computação em nuvem está baseada na arquitetura cliente-servidor, na qual os clientes e servidores estabelecem comunicação entre si através de uma rede de computadores, utilizando protocolos de comunicação para tal. No caso desta comunicação ser via internet, o cliente normalmente está interagindo com o servidor por meio de um navegador instalado em sua máquina.

O termo computação em nuvem se refere basicamente ao fornecimento e consumo de serviços de computação via internet, englobando as aplicações constituintes destes serviços e também o *hardware* existente nos *datacenters*. Conforme este modelo, os recursos são disponibilizados de acordo com as necessidades dos clientes, permitindo grande adaptabilidade para o provedor e evitando, assim, elevados investimentos por parte dos clientes para a construção de uma infraestrutura própria, o que geralmente acontece nos modelos de *hosting* locais (SANTOS, 2019). A computação em nuvem promove a disponibilidade sob demanda de recursos, como espaço de armazenamento ou consumo de serviços, via *web*, em computadores de empresas que atuam fornecendo este tipo de serviço, denominadas provedores em nuvem. Alguns exemplos são a *Amazon*, *Google* e *Microsoft* (GONÇALVES, 2012). Outra funcionalidade oferecida que é de grande vantagem é que as informações têm garantia de disponibilidade mesmo em caso de falha, pois são replicadas em múltiplos servidores.

A computação em nuvem, portanto, constitui assim uma boa solução para múltiplas organizações, como por exemplo *startups* e empresas que não tenham a tecnologia de informação como *core business*. Também é uma tendência para negócios com diferentes tipos de objetivos e características que tenham necessidades de escalabilidade, tempo de vida incerto ou que precisem de grande capacidade de processamento de dados. Entretanto, mesmo com diversas vantagens, quando as fontes de dados começam a se tornar extremamente numerosas, a rede tende a congestionar e aplicações em IoT que necessitam imprescindivelmente de baixa latência serão afetadas. Portanto, observa-se o surgimento da *edge computing* e *fog computing*.

Figura 7 – Cloud, fog e edge computing



Fonte: Afonso (2018).

Nota: Adaptado pelo autor.

O *edge computing* é uma prática de processamento de dados próximo à borda da sua rede, onde os dados são gerados. A delegação de processamento a dispositivos na periferia da rede permite reduzir significativamente a quantidade de dados a serem enviados e como consequência descongestiona o tráfego na rede de comunicação, reduzindo os custos referentes a isto, e promovendo melhorias quanto a latência e a qualidade do serviço. Hoje, somos capazes de agregar cada vez mais poder computacional à camada de borda, uma vez que esses dispositivos, além de possuírem capacidade de processamento, consomem cada vez menos energia e podem ser adquiridos a preços acessíveis.

Já o *fog computing* surge como uma infraestrutura de computação, com objetivo de ser o meio-termo, podendo realizar tarefas como criptografia dos dados, conversão de protocolos específicos de IoT para protocolos mais usuais, e até mesmo realizar alguma etapa de cálculo ou processamento, reduzindo a sobrecarga nos objetos IoT na camada de borda (SCHENFELD, 2017).

Dessa forma, a camada de *cloud* pode atuar apenas recebendo os resultados dos processamentos realizados pelos dispositivos de borda, mas ainda com a capacidade de executar este mesmo processo, se necessário. No entanto, essa tendência não diminui a importância da camada de

nuvem, que, assim, pode dedicar seu poder computacional para garantir alta disponibilidade, envio de comandos, registro de dispositivos, gerenciamento de configurações, etc.

Outro aspecto de extrema relevância deste modelo é a intrínseca resiliência desta arquitetura, pois, por exemplo, no caso de uma falha na rede ou indisponibilidade do sistema centralizado, ou seja, a própria nuvem, os dispositivos de borda não são severamente afetados e podem continuar operando de maneira autônoma (AFONSO, 2018).

As categorias ofertadas de computação em nuvem são variadas. Existem diferentes níveis de serviço a ser oferecidos pelas empresas provedoras que podem ser devidamente categorizados de uma maneira geral como: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS) (GONÇALVES, 2012). A infraestrutura como serviço (IaaS) é a categoria mais flexível dos serviços em nuvem e tem como objetivo oferecer controle total sobre o *hardware* que executa o aplicativo (servidores, armazenamento, redes e sistemas operacionais). A plataforma como serviço (PaaS) fornece um ambiente para teste e implantação de aplicativos de *software*, sem a necessidade de gerenciar a infraestrutura subjacente. Já o *software* como serviço (SaaS) é um *software* que é hospedado e gerenciado pela nuvem para o cliente final (MICROSOFT AZURE, 2018b). O mais adequado para este projeto é o *Platform as a Service* (PaaS), que provê mecanismos que permitem aos usuários desenvolver, gerenciar e hospedar aplicativos.

O *Microsoft Azure* é a plataforma de computação em nuvem da *Microsoft* e a sua designação para ser aplicada se deu em virtude da sua adequação aos propósitos deste projeto. Além do mais, esta plataforma possui recursos que permitiriam escalamento da solução futuramente. O *Microsoft Azure* provê serviços específicos para diferentes tipos de conectividade para adequar a incorporação dos dados existentes à capacidade da computação em nuvem.

Uma das ferramentas existentes na plataforma *Microsoft Azure* é o *IoT Hub*, que é um *gateway* de nuvem que pode absorver quantidades massivas de dados e processá-los ou armazená-los para realizar ações de acordo com regras de negócio. Supre os requisitos da conexão de dispositivos de Internet das Coisas à nuvem do *Microsoft Azure*, permitindo o gerenciamento de dispositivos e até mesmo estabelecendo comunicação bidirecional.

3 DESENVOLVIMENTO

3.1 *Benchmarking* e Definições de Projeto

Anand e Kodali (2008) constatam que o próprio processo de *benchmarking* possui várias definições, mas poderia ser descrito como a avaliação sistêmica de produtos, serviços e processos das organizações que reconhecidamente possuem as melhores práticas, com intuito de promover a melhor escolha, levando em conta os padrões observados.

Para apoiar a definição de produtos e ferramentas de tecnologia, a empresa de consultoria *Gartner*, uma das mais respeitadas do ramo, realiza pesquisas e divulgação de análise de presença de mercado, efetuando comparativos entre os atuantes de variados segmentos. Sua metodologia consiste em um gráfico de quadrantes, no qual posiciona as corporações consideradas líderes, visionárias, desafiantes ou “*players* de nicho”, de acordo com sua visão do todo e habilidade para executar tal visão. A *Microsoft*, atualmente, é considerada líder tanto nos segmentos de Plataformas de *BI*, quanto de infraestrutura de nuvem, como pode ser visto nas Figuras 8 e 9, ambas as mais recentes análises divulgadas no momento de levantamento bibliográfico deste projeto.

Figura 8 – Quadrante da Gartner Inc. referente a plataformas de BI e Analytics



Fonte: Gartner Inc. (2020).

Figura 9 – Quadrante da Gartner Inc. referente a infraestrutura de nuvem



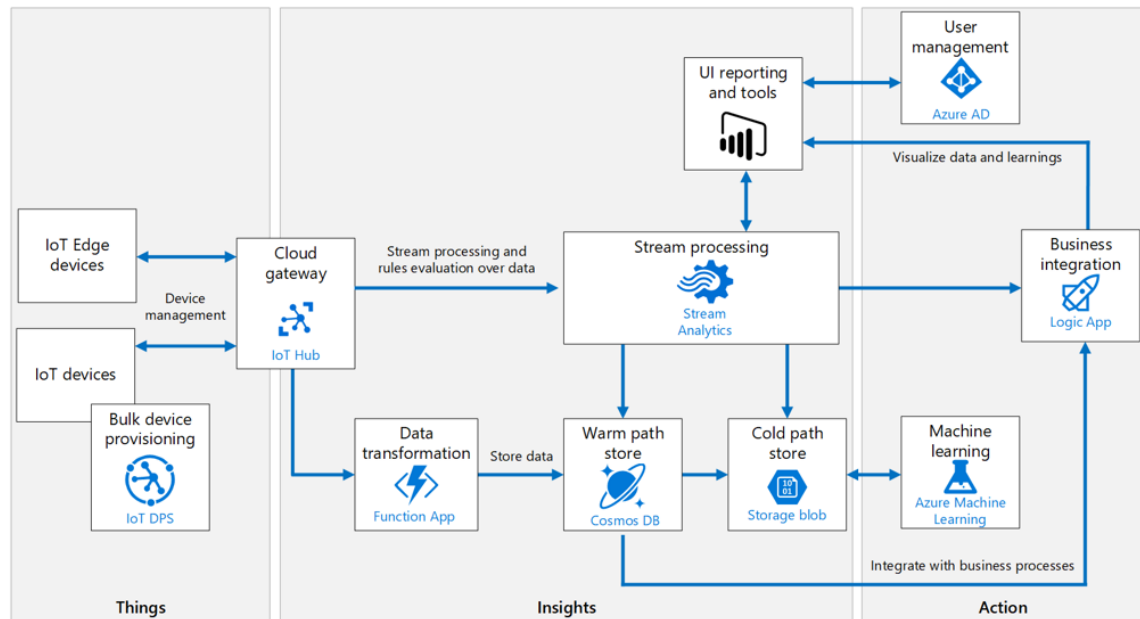
Fonte: Gartner Inc. (2019).

Em razão destas análises e também sobre o que já foi citado anteriormente identificou-se como a melhor plataforma de BI a da *Microsoft*, denominada *Power BI*. Esta ferramenta não somente se restringe a análises históricas, como também pode efetuar diagnósticos preditivos aliada a outras ferramentas como o *Azure Machine Learning* e o *Azure Stream Analytics* (BRITO; OLIVEIRA, 2017). Uma outra vantagem reside no fato de ser multiplataforma. É possível, por exemplo, criar visualizações em um *desktop*, compartilhá-las entre membros de uma organização de variadas maneiras, e até mesmo existe a possibilidade de acessar tais *dashboards* a partir de um aplicativo para *smartphones*, o *Power BI Mobile* (BRITO; OLIVEIRA, 2017), o que o torna atrativo também para o caso de inspetores que necessitem destes dados em campo com praticidade em uma indústria.

Outra grande motivação para a escolha do *Power BI*, se deu devido a este possuir recursos com os quais é possível emular o mesmo comportamento de um sistema de informações de planta (PIMS). Para viabilizar o projeto criou-se uma licença particular, usufruindo dos recursos possíveis nas categorias isentas de custos.

Considerando, conforme citado anteriormente, a *Microsoft* como líder de mercado de computação em nuvem e que seus produtos atendem adequadamente aos objetivos do projeto, optou-se pelo uso dos recursos por ela fornecidos. Desta forma decidiu-se pela plataforma *Microsoft Azure*. A Figura 10 mostra o conjunto de recursos que compõem a referida plataforma e que subsidiam esta opção.

Figura 10 – Modelo de referência recomendado para aplicativo IoT no *Microsoft Azure* usando componentes PaaS



Fonte: *Microsoft Azure* (2020).

Deste modelo de referência (Figura 10) aproveitou-se como estrutura para o projeto seguintes componentes: dispositivos IoT, *gateway* de nuvem (*IoT Hub*), processamento de fluxo (*Stream Analytics*) e as ferramentas de interface e geração de relatórios (*Power BI*). Os dispositivos IoT se registram e conectam ao *gateway* de nuvem para enviar e se necessário também receber dados. Este *gateway* por sua vez fornece gerenciamento e controle dos dispositivos, através de uma conectividade segura. O processamento de fluxo atua como uma ferramenta de tempo real para controle do fluxo de dados e podendo servir como fonte para os serviços de bancos de dados, análise de dados e *dashboards* de BI.

Para sensoriamento da Central de Detecção e Alarme de Incêndio, escolheu-se trabalhar com o microcontrolador *Arduino* e alguns de seus módulos sensores, além de alguns componentes eletrônicos que se fizeram necessários. O *Arduino* é o componente responsável por colher os dados a partir dos seus módulos sensores e tratá-los para envio, por meio de comunicação serial, para um *Raspberry Pi*. A diferença entre os dois dispositivos é que o *Arduino* é um microcontrolador e o *Raspberry Pi* é um SBC (*Single Board Computer*), o que significa que este possui um sistema operacional rodando internamente. O uso conjunto de ambos os equipamentos neste projeto se justifica pelo fato de que o *Arduino* é capaz de trabalhar com entradas analógicas de sinal, enquanto que no *Raspberry Pi* isto não é possível. Para a realização

deste projeto, foram especificados os seguintes modelos: um *Raspberry Pi 3 B+* (Figura 11), um microcontrolador *Arduino Uno* (Figura 12).

Figura 11 – *Raspberry Pi 3 B+*



Fonte: *Raspberry Pi Foundation* (2018).

Figura 12 – *Arduino Uno*



Fonte: *Arduino* (2013).

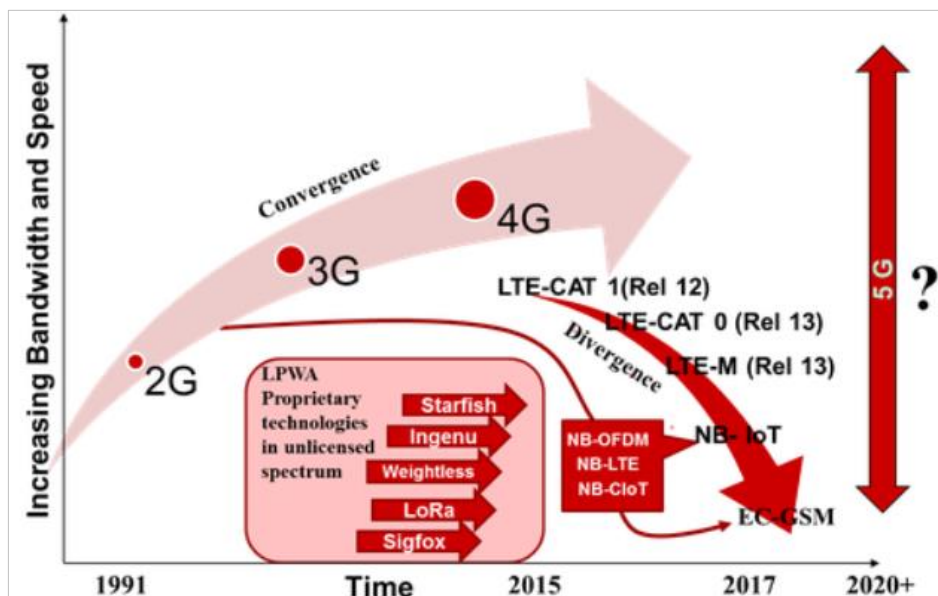
Para o projeto sob pauta, utilizou-se o sistema operacional *Raspbian*, versão *Buster*. O *Raspbian* é um sistema operacional baseado em *Debian/Linux*, concebido para o uso no *Raspberry Pi*. Foi habilitado e configurado o acesso ao *Raspberry Pi* via *Virtual Network Computing*, pensando numa possível necessidade de acessá-lo para fins de *reboot* ou possíveis ajustes, sem

a necessidade de se acoplar diretamente ao dispositivo, periféricos como teclado e monitor para comandar o *Raspberry Pi*.

No *Raspberry Pi*, é executado um programa em linguagem *Python*, o qual tem como função ler os dados de telemetria enviados pelo *Arduino* a partir da porta serial, e, com a importação das bibliotecas apropriadas, ser capaz de estabelecer conexão e enviar estes dados para o serviço de nuvem *Microsoft Azure*. Considerando que o *Raspberry Pi* possui a capacidade de se comunicar através de rede *Wi-Fi* e que por serem os testes de bancada possíveis de realizar dentro do alcance desta rede esta foi a opção escolhida.

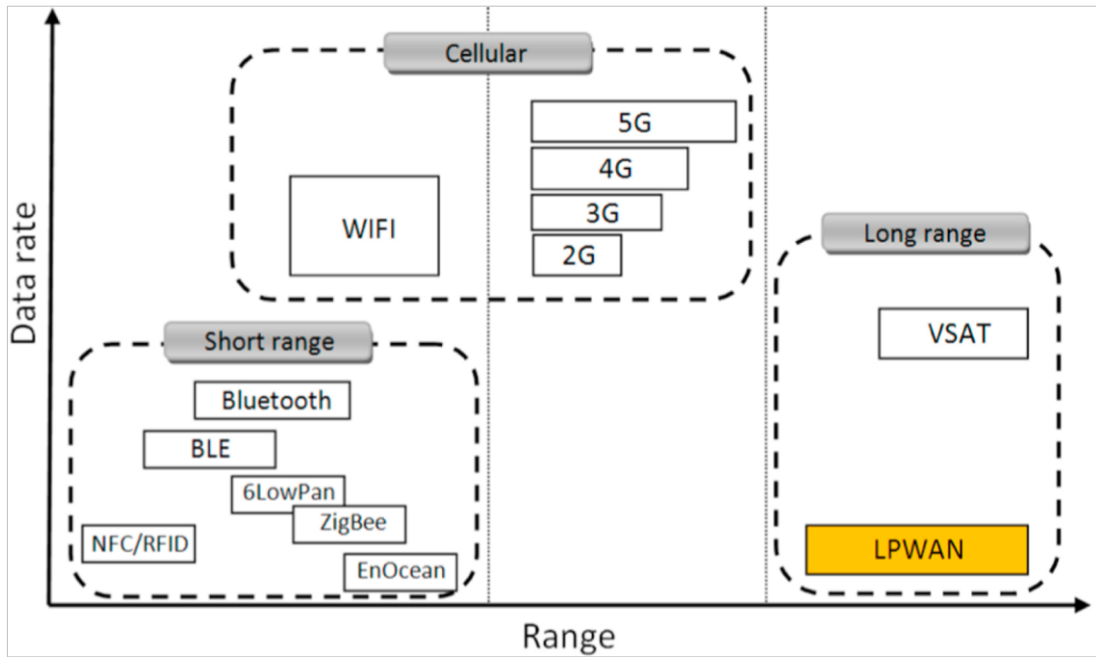
É válida a observação de que se tem a compreensão que no campo da IoT, e mais especificamente em aplicações de telemetria, o rumo que se percebe é como o que pode ser visto nas Figuras 13 e 14, devido ao fato de não necessitarem de grande largura de banda e também requererem um baixo consumo de potência.

Figura 13 – Evolução e estado atual das tecnologias de comunicação



Fonte: Bell (2016).

Figura 14 – Comparativo entre alcance e taxa de dados das tecnologias de comunicação

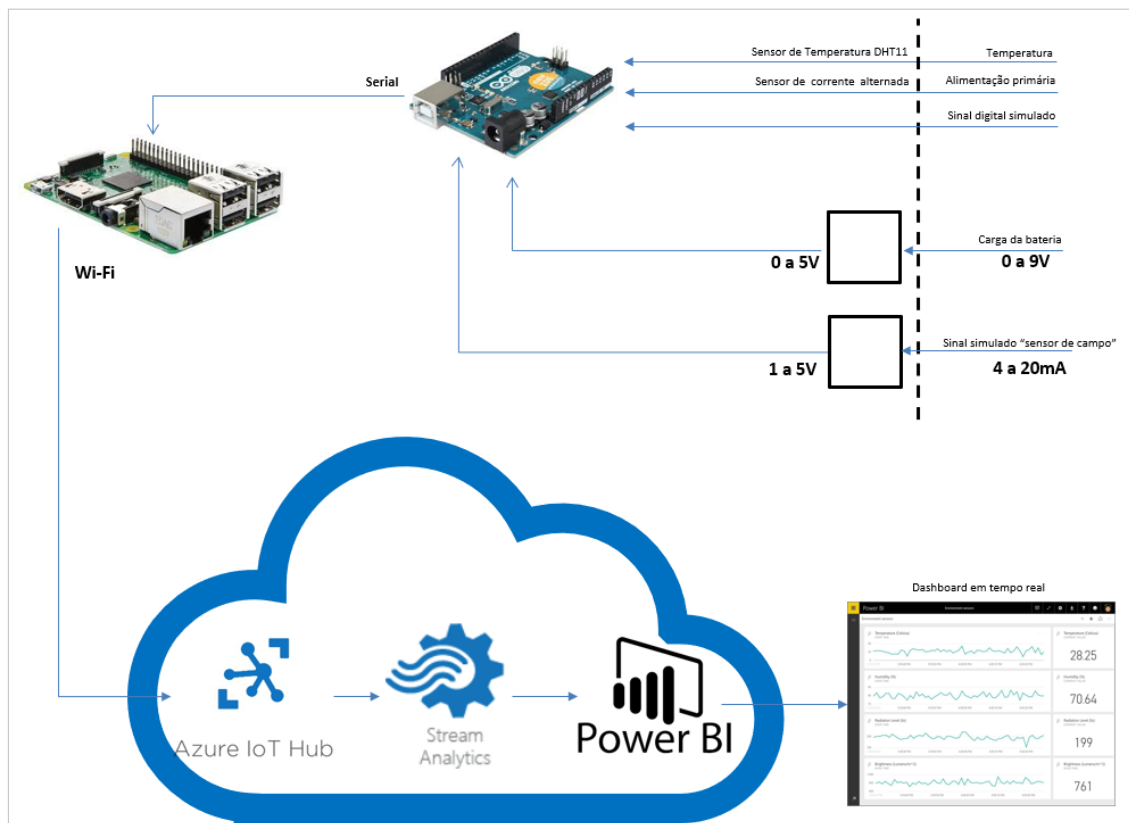


Fonte: Mekki e outros (2019).

Quanto a parte de sensoriamento de campo, foram consideradas as seguintes variáveis: alimentação primária, temperatura das baterias da alimentação secundária, tensão de alimentação secundária e status de alarme da central. Adicionalmente, incluiu-se uma variável no padrão 4 a 20 mA representando um sensor genérico. Além disso, houve também a necessidade de circuitos eletrônicos complementares para adequar as entradas de campo aos padrões de tensão do *Arduino*.

A arquitetura do projeto está mostrada na Figura 15.

Figura 15 – Arquitetura de aplicação do projeto – modelo funcional



Fonte: Produção do próprio autor.

3.2 Implementação

Após a definição da estrutura geral do projeto, procedeu-se à montagem do protótipo que consiste dos módulos microcontrolador *Arduino*, o SBC *Raspberry*, juntamente com a eletrônica necessária para adequar os sinais monitorados às entradas do *Arduino*. Os sinais monitorados foram definidos em acordo com os *stakeholders* da AMT.

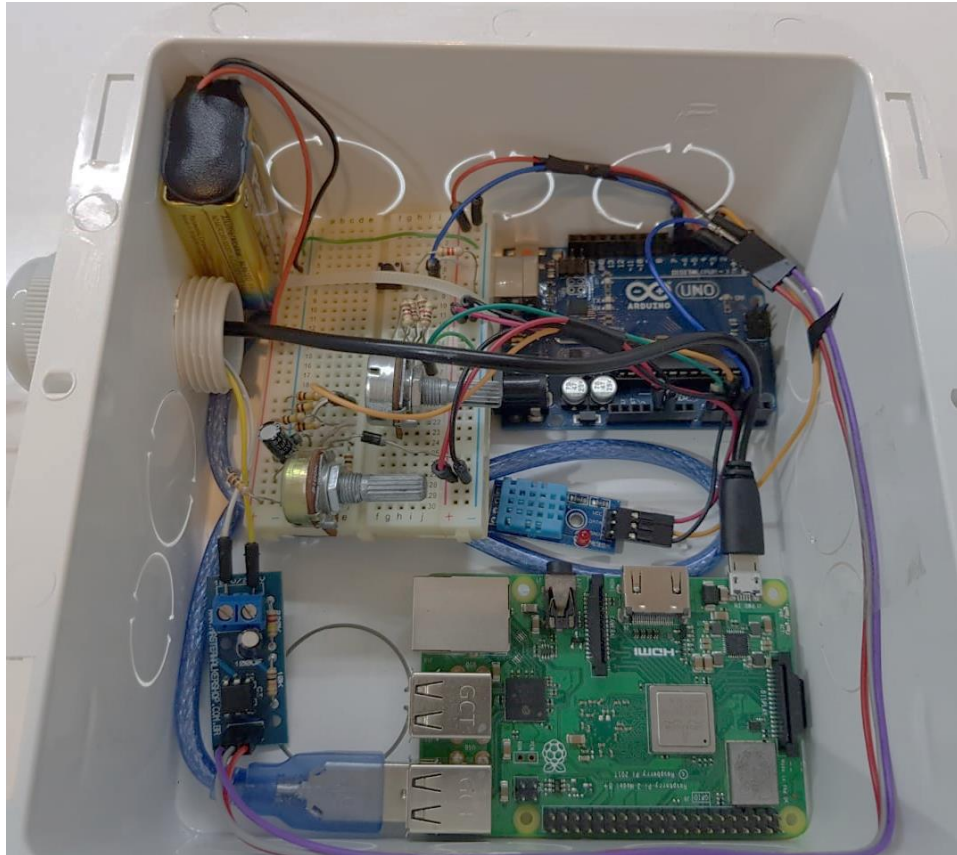
- Alimentação primária: a Central de Alarmes de Incêndio é energizada por um circuito em corrente alternada em 127 Vca, alimentação principal. O sensor de tensão CA escolhido é o modelo AC P8, de fabricação GBK e como é baseado em um acoplador óptico, fornece o devido isolamento entre a fonte de tensão e o microcontrolador, proporcionando, dessa maneira, uma boa segurança.
- Alimentação secundária: a Central de Detecção e Alarmes de Incêndio possui, além da alimentação primária citada, uma fonte de alimentação de emergência consistindo de

duas baterias em série + carregador, totalizando 24 Vcc. Para adequar a tensão da bateria (24 Vcc) ao intervalo de medição do *Arduino* (0 a 5 Vcc) um conversor de tensão foi projetado como pode ser observado no diagrama de ligação do APÊNDICE B.

- Temperatura das baterias: esta informação é bastante útil para a análise de sua situação de funcionamento e previsão de vida útil e dessa forma possibilitando prever futuras falhas. Para o seu sensoriamento escolheu-se o sensor de temperatura e umidade modelo DHT11 de fabricação *Mouser Electronics*, com resolução de 1°C, exatidão de ± 2 °C.
- Sinal digital: Trata-se do status de alarme da Central de Detecção e Alarme de Incêndio com o requisito de que o sinal da tensão digital de entrada seja compatível com a entrada digital do *Arduino* (0 a 5 Vcc). Para garantir o nível lógico zero na entrada do *Arduino* um resistor de *pull-down* foi inserido (APÊNDICE B). Este requisito de monitoramento é exclusivo das centrais convencionais sob pauta, uma vez que nas do tipo endereçada a capacidade de monitoramento remoto já existe.
- Sensor analógico padrão 4 a 20 mA: um dos padrões de sinal de instrumentação mais usados no setor industrial, existindo equipamentos detectores de incêndio que trabalham com esta padronização. Também é útil abranger esta possibilidade para tratar os mais diversos ativos que possam existir, de forma a realizar uma integração plena. A inserção de um resistor de 250 Ω em paralelo com a entrada digital do *Arduino* faz a conversão do padrão 4 a 20 mA para o padrão do *Arduino* (1 a 5 Vcc).

A Figura 16 mostra como foi feita a montagem do protótipo em sua configuração para realização dos testes de bancada. A bateria usada nesta montagem é de 9 Vcc, uma adaptação feita com enfoque na praticidade.

Figura 16 – Montagem do protótipo de testes



Fonte: Produção do próprio autor.

No APÊNDICE A encontra-se o diagrama de ligação de todo o projeto com as adequações necessárias à realização dos testes de bancada, caracterizando assim a execução da fase de projeto denominada de *Factory Acceptance Test (FAT)*.

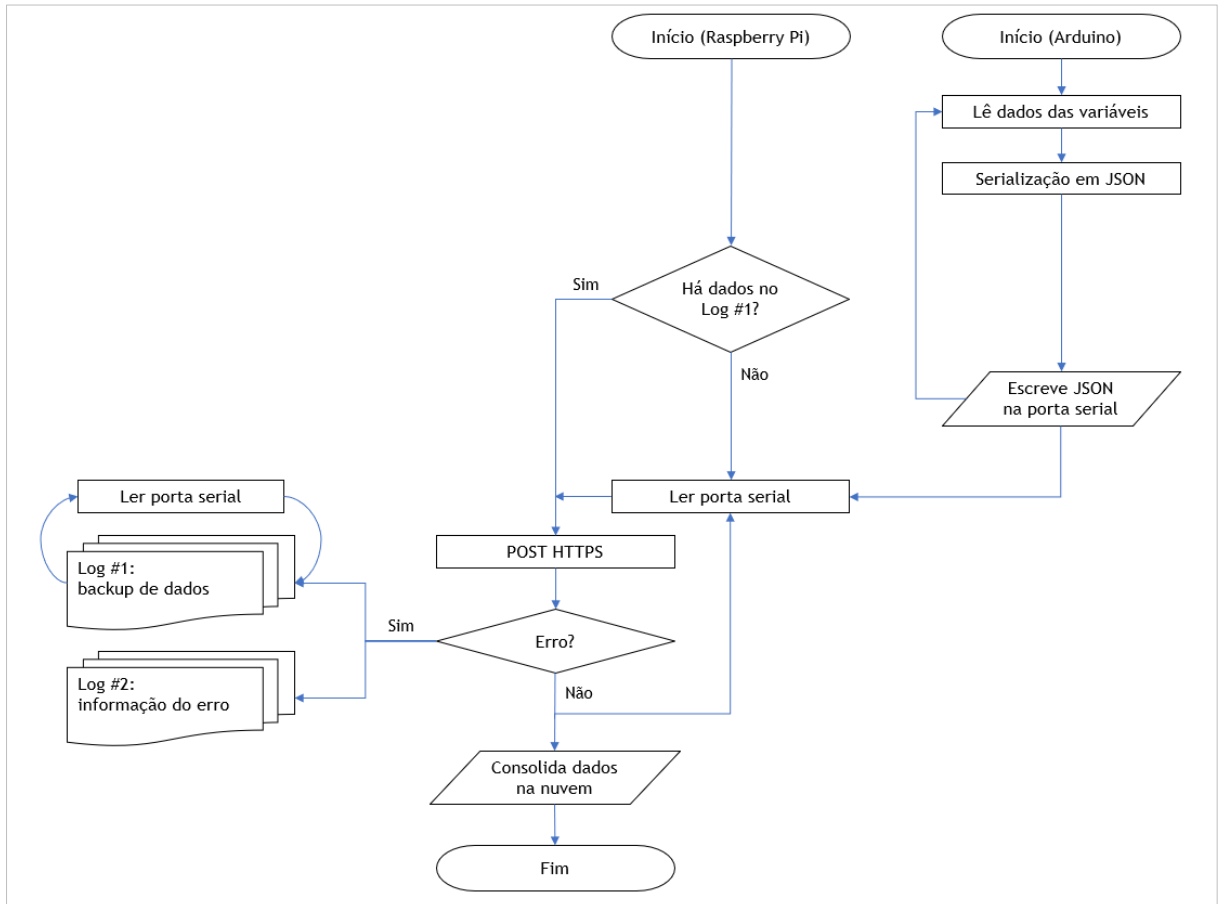
O APÊNDICE B traz o diagrama de ligação considerando o escopo gerado pela demanda da ArcelorMittal Tubarão, quando o aluno, na condição de estagiário da empresa a consultou sobre um tema de interesse desta que pudesse se transformar em um projeto de fim de curso.

Os dois diagramas de ligação têm a finalidade de mostrar que as diferenças entre o escopo original e o adaptado se situam basicamente no uso da infraestrutura de integração e consolidação das variáveis monitoradas e no valor da tensão secundária, uma vez que tais elementos só poderiam ser acessados internamente à ArcelorMittal Tubarão.

Concluída a montagem de todo o *hardware*, foi elaborado todo o código computacional necessário para o funcionamento do protótipo. Pode se ver o fluxograma básico do código

gerado na Figura 17. A estrutura principal consistiu desde a leitura de dados dos sensores pelo microcontrolador *Arduino*, passando pela serialização em formato JSON, comunicação com o *Raspberry Pi* pela porta serial, até o envio pelo protocolo HTTPS para a plataforma de nuvem.

Figura 17 – Fluxograma do *software* produzido



Fonte: Produção do próprio autor.

O *software* foi desenvolvido conforme o fluxograma da Figura 17 e descrito resumidamente a seguir.

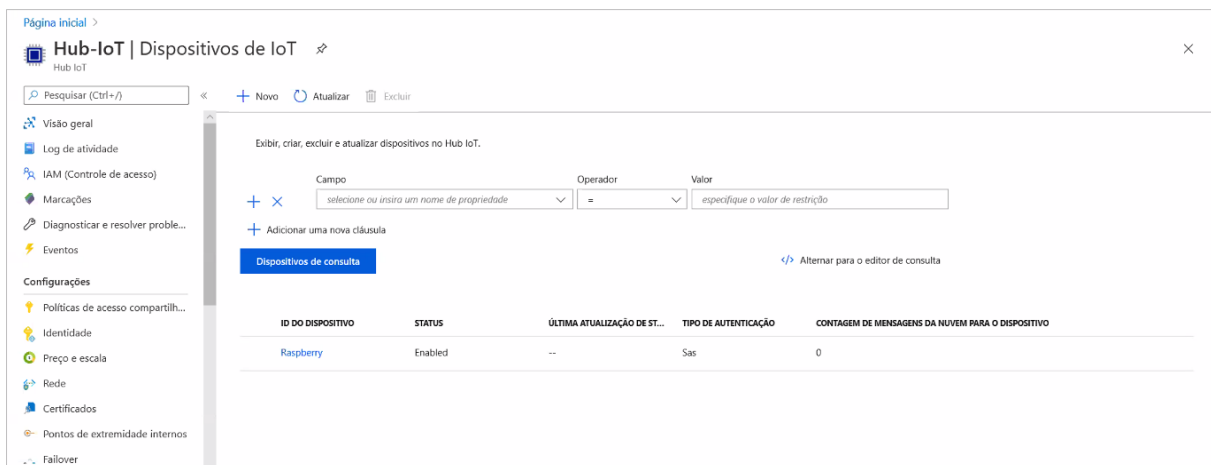
O *Arduino* captura as variáveis ligadas em suas entradas e as converte seus valores para um formato compatível para seu processamento e as serializa, enviando os dados referentes a estas variáveis para o *Raspberry* através de sua porta serial. O *software* no *Raspberry* foi desenvolvido na linguagem *Python* e realiza as seguintes rotinas: associação dos dados serializados no *Arduino* a um *timestamp*; encapsulamento desses dados em uma mensagem a ser enviada para a nuvem pelo protocolo HTTP; tratamento dos erros de comunicação conforme fluxograma da Figura 17; e gerenciamento dos arquivos de *backup*.

Obviamente, vários detalhes necessitaram de igual atenção, como as configurações do ambiente de programação, autenticação com a nuvem, como também a definição prévia do comportamento do sistema em casos de anormalidade. Em vista desses detalhes estabeleceu-se que haveriam dois arquivos de texto locais, destinados ao processo de *logging*. Em um arquivo se grava as mensagens de erro propriamente ditas, visando uma consulta por mantenedores do sistema. Um outro arquivo tem como finalidade atuar como um *backup* para os dados medidos durante o tempo em que o protótipo se encontrar sem comunicação, em virtude da ocorrência de erros. Esta estratégia foi pensada para que, assim que fosse reestabelecida a comunicação, estes dados fossem enviados para a plataforma de nuvem, contendo o *timestamp*, e, portanto, sem a ocorrência de perda de informações.

Vale também a observação de que, houve a preocupação em configurar o comportamento dos próprios arquivos de *log*, deixando como parâmetro a ser passado as seguintes características: nível de *log* (*DEBUG*, *INFO*, *WARNING*, *ERROR* ou *CRITICAL*), o tamanho máximo do arquivo em *bytes*, e o número de *backups* desejados, de forma que os dados inseridos tenham um comportamento *First In, First Out*.

A configuração da plataforma de nuvem se deu como descrito a seguir: no portal *Microsoft Azure*, primeiramente criou-se o *IoT Hub*, e a este vinculado uma representação digital do *Raspberry* (referido pela plataforma como um *device twin*), como pode ser visto na Figura 18.

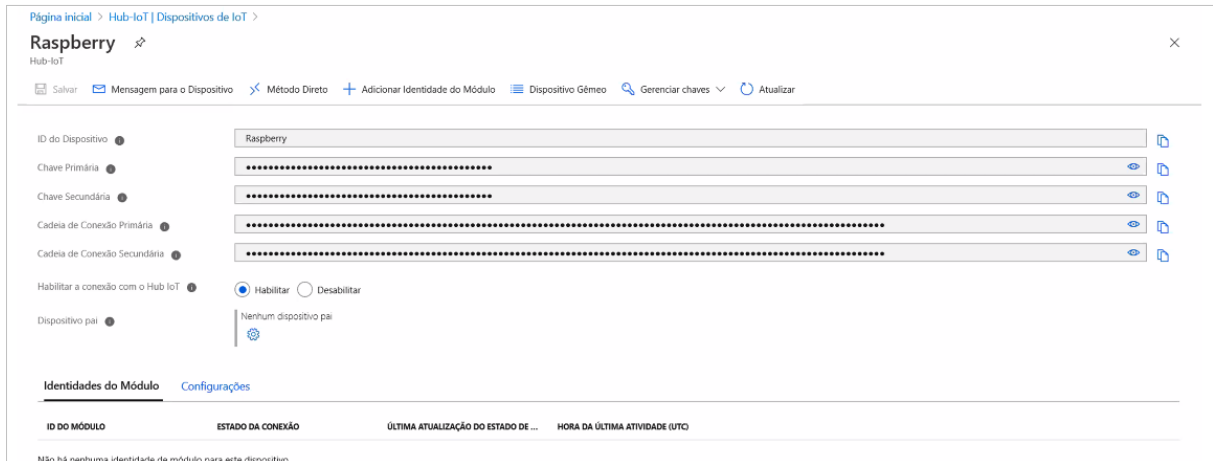
Figura 18 – Interface da plataforma de nuvem para cadastro do dispositivo



Fonte: Produção do próprio autor.

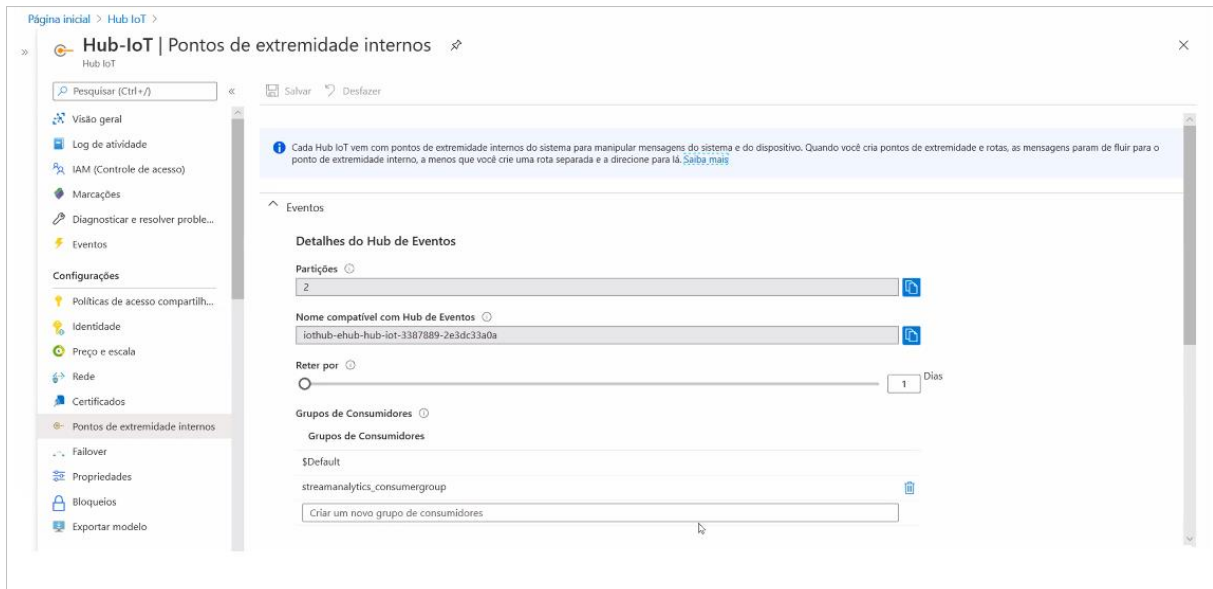
Observando as propriedades do *IoT Hub*, como na Figura 19, obtém-se as chaves de acesso (*connection strings*) que permitirão a correta conexão do protótipo com a plataforma de nuvem. Após, foi necessário criar o que é chamado de *endpoints*, ou em português, pontos de extremidade internos, como na Figura 20, pelos quais pode-se interligar os recursos da nuvem e que neste caso específico seriam o próprio *IoT Hub* e o *Stream Analytics*.

Figura 19 – Interface da plataforma de nuvem com propriedades do dispositivo



Fonte: Produção do próprio autor.

Figura 20 – Interface da plataforma de nuvem para configurações de pontos de extremidade internos



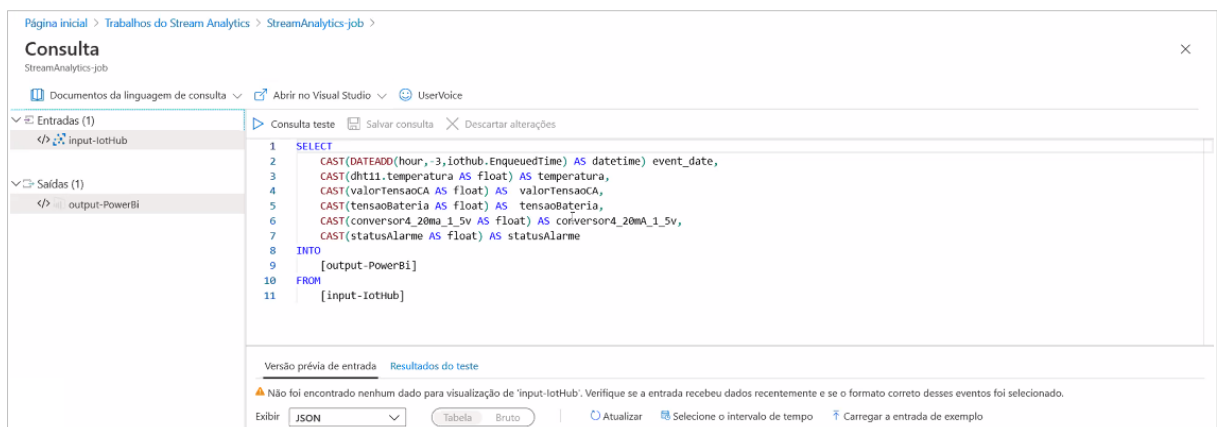
Fonte: Produção do próprio autor.

Com o *streaming* do *Power BI*, torna-se possível criar *dashboards* para exibir tais dados com atualização em tempo real. As fontes de dados, neste caso, eram provenientes do *Microsoft*

Azure com um importante detalhe: o *Power BI* armazena-os em um *cache* temporário, sendo propício, portanto, para exibir gráficos que tenham comportamento transitório. Como não há um banco de dados propriamente dito incorporado ao sistema, efetuar análises históricas de longo prazo é uma funcionalidade não implementada, porém possível.

Para possibilitar o *streaming* do *Power BI*, bastou realizar a última etapa de configuração da plataforma de nuvem, com a ferramenta *Stream Analytics*, pela qual inseriu-se uma *query*, ou consulta, tal que definisse quais dados, onde seriam lidos, e o destino destes mesmos dados. Este procedimento é encontrado na Figura 21. Concluído isto e já usufruindo do *Power BI*, efetuou-se a etapa de importação do conjunto de variáveis, exemplificada na Figura 22 para posterior criação da tela de visualização.

Figura 21 – Interface da plataforma de nuvem para inserção de *query*



Fonte: Produção do próprio autor.

Figura 22 – Interface do *Power BI* para configuração dos dados de *streaming*

Editar conjunto de dados de streaming

Crie um conjunto de dados de streaming e integre a sua API no seu dispositivo ou aplicativo para enviar dados. [Saiba mais sobre a API.](#)

* Obrigatório

Nome do conjunto de dados *

dataset1

Valores do fluxo *

event_date	DateTime	🗑️
temperatura	Número	🗑️
valorTensaoCA	Número	🗑️
tensaoBateria	Número	🗑️
conversor4_20mA_1_5v	Número	🗑️
statusAlarme	Número	🗑️

Fonte: Produção do próprio autor.

Para a realização dos testes de bancada foi preciso criar uma tela de monitoramento, empregando as possibilidades descritas do *Power BI*, que exibisse um acompanhamento em tempo real das variáveis monitoradas tal que permita fácil identificação das grandezas de interesse sensoriadas pelo protótipo.

4 ANÁLISES DOS TESTES E RESULTADOS

Tendo realizado toda a montagem e configuração e programação de todo software, conforme APÊNDICE D, prosseguiu-se para a fase de testes do protótipo, avaliando não só seu bom funcionamento, como também os resultados obtidos em condições de falhas. Para tanto, se fez necessário dar enfoque ao teste de *software*, devido à preponderância deste em relação a outras partes que compõem o protótipo, já que o *hardware* por si é o suporte para funcionamento do *software*. Considera-se que a nuvem já tem o seu funcionamento correto consolidado.

Segundo Souto (2010), “Testar um *software* significa verificar através de uma execução controlada se o seu comportamento corresponde ao especificado”. Tendo isso em vista, foi identificado na implementação que há ações que dependem de componentes externos, mais especificamente o próprio processo de envio de dados, que em seu protocolo de comunicação possui uma requisição e também uma resposta para a requisição. Desta maneira, para realizar testes nos quais se garantisse o isolamento dos testes em relação aos componentes externos, valeu-se do uso de testes unitários. Estes testes exercitam características da unidade sob teste, de preferência isolando-a, para cumprir o objetivo de encontrar falhas, que sejam independentes do sistema como um todo.

Ainda de acordo com Souto (2010), uma técnica apropriada para executar testes unitários é o uso de objetos *mock*. Estes são capazes de promover o isolamento citado, quando usados para substituir objetos pertencentes à unidade sob teste, pois eles têm a propriedade de permitir efetuar afirmações sobre o seu comportamento desejado, e assim simular os objetos originais substituídos, com igual funcionamento. Situações nas quais objetos *mock* são indicados, incluem, entre outros, o caso de a unidade sob teste ter um comportamento difícil de desencadear, como é o caso deste projeto, pois é necessário testar o funcionamento sob ocorrências de erros de comunicação. Destaca-se entre as vantagens do uso de objetos *mock* a velocidade de execução dos testes e a independência de elementos externos, como bancos de dados ou conexões de rede.

Para a realização dos testes de comunicação foi utilizada uma biblioteca de objetos destinada a este fim para programação na linguagem *Python*, denominada *unittest.mock*.

No arquivo de execução do programa principal há a função que coleta os dados de campo e os transmite para a nuvem, na qual existe um bloco de código chamado *try-except* cuja finalidade é o tratamento de erros de comunicação. Todos os comandos dentro do bloco *try-except* são executados em um *loop*, exceto no caso da ocorrência de algum erro do tipo como os citados abaixo. Os erros detectados são armazenados em um arquivo de texto que grava a mensagem, juntamente com o *timestamp*, possibilitando assim uma verificação posterior.

Esses erros podem ocorrer quando se tenta enviar as mensagens para nuvem via protocolo HTTP, através do envio de dados pelo processo conhecido como POST (a comunicação neste projeto é unidirecional – do dispositivo para a nuvem). Para poder executar os testes, um arquivo independente para este fim foi criado (*teste.py*). Este arquivo é uma cópia do trecho do programa principal que faz o tratamento de erros, substituindo o comando que executa o método POST pelo objeto *mock.Mock*. Isto permite forçar a ocorrência de um erro que se deseja testar.

Testes Executados:

- HTTP ERROR: Ocorre quando a requisição retorna um código de status mal sucedido (Ex: Código 404 – servidor não pode encontrar o recurso solicitado).
- CONNECTION ERROR: Ocorre quando há problemas de rede tais como: falha de DNS, conexão recusada, etc.
- TIMEOUT ERROR: Ocorre quando o tempo pré-determinado para a conclusão de um evento é ultrapassado.
- REQUEST EXCEPTION: Quando existe uma exceção indefinida durante uma requisição, pois, mesmo que uma requisição for feita de forma correta poderá haver um erro quando for feita a tentativa de executá-la. Os erros detectados durante a execução são chamados de exceções.

Para maiores detalhes, observar um dos programas dos testes que se encontra no APÊNDICE E, cuja estrutura base foi utilizada para os demais testes de *software*, e o caderno de testes no APÊNDICE F.

E por fim, foi realizado o teste de monitoramento das variáveis através de gráficos de tendência de tempo real, como visto na Figura 23, atendendo as necessidades estabelecidas pelo projeto.

Figura 23 – Dashboard criado no Power BI



Fonte: Produção do próprio autor.

Durante a execução destes testes foi percebido um *delay* da ordem de segundos na apresentação dos valores das variáveis. Ocorre, porém, que durante o levantamento de requisitos de projeto definiu-se que a atualização em tempo real não seria algo crucial. Isto é devido ao fato de que, na AMT, atualmente esta tarefa necessita ser realizada *in loco*, e apenas é executada poucas vezes ao ano devido ao deslocamento e mobilização de recursos para tal, visto que algumas das centrais estão em áreas consideravelmente remotas da usina. Logo, percebe-se o enorme ganho que pode ser obtido, em termos tanto de confiabilidade do sistema quanto de economia de homens-hora. Ainda é sabido que em sistemas de automação aplicações de monitoramento/supervisão são as menos afetadas por atrasos de transmissão (AKERBERG; GIDLUND; BJORKMAN, 2011).

5 CONCLUSÃO E TRABALHOS FUTUROS

Em virtude do exposto, considera-se que o projeto desenvolvido logrou resultados satisfatórios para o que se havia proposto implementar e observa-se que o protótipo desempenha aquilo do qual se tinha expectativa, efetuando aquisição de dados, conexão com serviço de computação em nuvem e monitoramento em tempo real por meio de *dashboards* para os usuários.

O embasamento teórico promoveu sólidas premissas para a concepção do protótipo, com o qual apresentou-se simulações nas quais foram percebidos os efeitos desejados. No entanto, também se ressalta que devido a contextos externos ao projeto houve impossibilidade de realizar os testes de campo (SAT, do inglês *Site Acceptance Test*). Assim, dentro do devido tempo e da oportunidade, propõe-se a realização do SAT, complementarmente a este trabalho, no futuro.

Além disso, uma ideia interessante como sugestão para uma futura empreitada seria a adequação da solução adotada às redes *Low-Power Wide-Area Network* (LPWAN), ainda não tão difundidas, porém com características mais adequadas aos princípios de IoT do que o *Wi-Fi* ou tecnologias de banda larga móvel.

Pensando em escalabilidade da solução, conforme outras Centrais de Detecção e Alarme de Incêndio fossem necessitando de monitoramento, este projeto poderia ser replicado promovendo a integração entre elas sem grandes dificuldades. Todos esses aprimoramentos proporcionariam uma aplicação ainda mais ampla da solução, bem como resultados ainda mais confiáveis.

Necessário se faz neste momento comentar a forma como este projeto foi implementado. É digno de nota que seu sucesso começa já na fase de planejamento, quando na disciplina Projeto de Graduação I se determinou em primeiro lugar as metodologias de planejamento. As escolhidas foram: *Front-End Loading* (FEL) e *Critical Chain Project Management* (CCPM). Resumidamente, a metodologia de FEL se processa em três fases, na forma de gates (um *gate* só abre quando o outro fecha), levando a um detalhamento e uma compreensão cada vez maior do que deve ser executado. O CCPM é, em linhas gerais, uma forma de gerenciar um projeto olhando para frente. A pergunta que se faz é: O que é que falta fazer? Nele se reduz o tempo originalmente estimado da tarefa em 50% e metade destes 50% é acrescentado ao final da data

de término, constituindo o que se chama de *buffer*, levando a uma redução de 25% do tempo original.

Em segundo lugar foi adoção sistemática de reuniões de performance previamente planejadas, com emissão de atas para efeito de controle de pendências.

De igual importância foi a análise de riscos feita ainda na disciplina Projeto de Graduação I para a escolha do projeto que alcançasse os melhores resultados e que fosse exequível no tempo planejado.

Tudo isto foi incorporado aqui para dizer que, ainda no cronograma de dezembro de 2019, a data de término deste projeto foi determinada em 20/07/2020, o *buffer* incluso (APÊNDICE G). A revisão final desta monografia feita com o orientador e o aluno considerada terminada no dia 23/07/2020, apesar das imprevisibilidades que ocorreram neste período letivo e que não cabe aqui mencionar.

REFERÊNCIAS BIBLIOGRÁFICAS

AGÊNCIA NACIONAL DE VIGILÂNCIA SANITÁRIA. **Segurança contra incêndio em estabelecimentos assistenciais de saúde**. 1. ed. Brasília, DF: ANVISA, 2014. Disponível em: <http://www.pncq.org.br/uploads/2017/Infostecnicas/Manual-Seguranca-Incendio-em-Estabelecimentos-Assistenciais-Saude.pdf>. Acesso em: 02 maio 2020.

AKERBERG, J.; GIDLUND, M.; BJORKMAN, M. Future research challenges in wireless sensor and actuator networks targeting industrial automation. *In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS*, 9., 2011, Lisboa. **Proceedings** [...]. [S.I.]: IEEE, 2011. p. 410-415.

AFONSO, J. **Edge Computing: learn to delegate**. 2018. Disponível em: <https://blog.octo.com/en/edge-computing-learn-to-delegate>. Acesso em: 23 jul. 2020.

ANAND, G.; KODALI, R. Benchmarking the benchmarking models. **Benchmarking: An International Journal**, v. 15, n. 3, p. 257-291, 2008.

ARDUINO. **Arduino Uno Rev3**. 2013. Disponível em: <https://store.arduino.cc/usa/arduino-uno-rev3>. Acesso em: 11 jul. 2020.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 17240**: Sistemas de detecção e alarme de incêndio – Projeto, instalação, comissionamento e manutenção de sistemas de detecção e alarme de incêndio – Requisitos. Rio de Janeiro: ABNT, 2010.

_____. **NBR 7240-1**: Sistemas de detecção e alarme de incêndio. Parte 1: Generalidades e definições. Rio de Janeiro: ABNT, 2017.

BELL, S. **How IoT Forked the Mobile Roadmap**. 2016. Disponível em: <https://www.lightreading.com/iot/iot-strategies/how-iot-forked-the-mobile-roadmap/a/d-id/720262>. Acesso em: 08 jun. 2020.

BITTENCOURT, A. A.; OLIVEIRA, C. A. Monitoramento de Aplicações no ambiente de Automação Industrial. *In: SEMINÁRIO DE AUTOMAÇÃO E TI*, 21., 2017, São Paulo. **Anais** [...]. São Paulo: ABM, 2017. p. 228-237.

BOYES, H.; HALLAQ, B.; CUNNINGHAM, J.; WATSON, T. The industrial internet of things (IIoT): An analysis framework. **Computers in Industry**, v. 101, p. 1-12, 2018.

CARMO, T. R. **Uso do Padrão AMQP para transporte de mensagens entre atores remotos**. 2012. Dissertação (Mestrado em Ciência da Computação) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2012.

CONFEDERAÇÃO NACIONAL DA INDÚSTRIA. **A Indústria do Aço no Brasil**. Brasília: CNI, 2012. Disponível em: https://bucket-gw-cni-static-cms-si.s3.amazonaws.com/media/filer_public/9d/68/9d680f8f-38e6-4077-89fb-55bf1bf63f68/20131002174604375684e.pdf. Acesso em: 14 jul. 2020.

CUMMINS, F. A. **Integração de sistemas**: arquiteturas para integração de sistemas e aplicações corporativas. 1. ed. Rio de Janeiro: Campus, 2002.

DAMASCENO, E. C. **Integração da gestão da segurança e saúde no trabalho à gestão do processo produtivo**: uma análise de abordagens adotadas no Setor Siderúrgico Brasileiro. 2014. Dissertação (Mestrado em Trabalho, Saúde e Ambiente) – Programa de Pós-Graduação em Trabalho, Saúde e Ambiente, Fundação Jorge Duprat Figueiredo de Segurança e Medicina do Trabalho, São Paulo, 2014.

FURHT, B.; ESCALANTE, A. (ed.) **Handbook of cloud computing**. Nova York: Springer, 2010.

GARTNER INC. **Magic Quadrant for Cloud Infrastructure as a Service, Worldwide**. Stamford: Gartner Inc., 2019.

_____. **Magic Quadrant for Analytics and Business Intelligence Platforms**. Stamford: Gartner Inc., 2020.

GONÇALVES, P. M. F. D. **Cloud Computing Sobre a Plataforma Windows Azure**. 2012. Dissertação (Mestrado em Engenharia Informática) – Instituto Superior de Engenharia do Porto, Politécnico do Porto, Porto, 2012.

HAUKELID, K. Theories of (safety) culture revisited - An anthropological approach. **Safety Science**, v. 46, n. 3, p. 413-426, 2008.

JUSTA, L. M. **Sistema de detecção e alarme de incêndio com supervisão SCADA**. 2016. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Centro de Tecnologia, Universidade Federal do Ceará, Fortaleza, 2016.

LESSAK, A. L. **Desenvolvimento de uma ferramenta de apoio à gestão para o polo de inovação do Instituto Federal de Santa Catarina utilizando *Business Intelligence***. 2018. Dissertação (Mestrado em Propriedade Intelectual e Transferência de Tecnologia para a Inovação) – Programa de Pós-Graduação em Propriedade Intelectual e Transferência de Tecnologia para Inovação, Centro Sócio-Econômico, Universidade Federal de Santa Catarina, Florianópolis, 2018.

LIMA, H. R. C. **Análise conceitual dos *softwares* de gerenciamento das informações industrial**. 2019. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) – Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, 2019.

MAZZER, D.; FRIGIERI, E. P.; PARREIRA, L. F. C. G. Protocolos M2M para Ambientes Limitados no Contexto do IoT: Uma Comparação de Abordagens. *In: SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES*, 33., 2015, Juiz de Fora. **Anais [...]**. Juiz de Fora: Sociedade Brasileira de Telecomunicações, 2015. p. 1-5.

MEKKI, K.; BAJICA E.; CHAXELA, F.; MEYER, F. A comparative study of LPWAN technologies for large-scale IoT deployment. **ICT Express**. v. 5, n. 1, p. 1-7, 2019.

MICROSOFT AZURE. **Arquitetura de referência de IoT do Azure**. 2020. Disponível em: <https://docs.microsoft.com/pt-br/azure/architecture/reference-architectures/iot>. Acesso em: 30 maio 2020.

_____. **Documentação do IoT Hub** – Referência – escolha um protocolo de comunicação. 2018a. Disponível em: <https://docs.microsoft.com/pt-br/azure/iot-hub/iot-hub-devguide-protocols>. Acesso em: 18 maio 2020.

_____. **Introdução para operadores de TI do Azure**. 2018b. Disponível em: <https://docs.microsoft.com/pt-br/azure/guides/operations/azure-operations-guide>. Acesso em: 10 dez. 2019.

MICROSOFT POWER BI. **Streaming em tempo real no Power BI**. 2020. Disponível em: <https://docs.microsoft.com/pt-br/power-bi/connect-data/service-real-time-streaming>. Acesso em: 29 jun. 2020.

MOREIRA, B. B. **Central de Alarmes com Interface Web**. 2010. Dissertação (Mestrado Integrado em Engenharia Electrotécnica e de Computadores) – Faculdade de Engenharia, Universidade do Porto, Porto, 2010.

NAIK, N. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. *In: IEEE INTERNATIONAL SYMPOSIUM ON SYSTEMS ENGINEERING, 3., 2017, Viena. Proceedings [...]*. Viena: IEEE, 2017. p. 12-18.

NURSEITOV, N.; PAULSON, M.; REYNOLDS, R.; IZURIETA, C. Comparison of JSON and XML Data Interchange Formats: A Case Study. *In: INTERNATIONAL CONFERENCE ON COMPUTER APPLICATIONS IN INDUSTRY AND ENGINEERING, 22., 2009, San Francisco. Proceedings [...]*. San Francisco: ISCA, 2009. p. 157-162.

RASPBERRY PI FOUNDATION. **Raspberry Pi 3 Model B+**. 2018. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus>. Acesso em: 11 jul. 2020.

RIBEIRO, D. D. **Sistema de Detecção, Alarme e Combate a Incêndio no Âmbito da Mineração**. 2014. Monografia (Especialização em Engenharia de Recursos Minerais) – Escola de Engenharia, Universidade Federal de Minas Gerais, Belo Horizonte, 2014.

SANTOS, B. P.; SILVA, L. A. M.; CELES, C. S. F. S.; BORGES NETO, J. B.; PERES, B. S.; VIEIRA, M. A. M.; VIEIRA, L. F. M.; GOUSSEVSKAIA, O. N.; LOUREIRO, A. A. F. Internet das Coisas: da Teoria à Prática. *In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, 34., 2016, Salvador. Livro de Minicursos [...]*. Porto Alegre: Sociedade Brasileira de Computação, 2016. p. 1-50.

SANTOS, Y. L. **Implementação de arquitetura baseada em IoT para leitura de tags RFID de Ultra Alta Frequência**. 2019. Dissertação (Mestrado Profissional em Computação Aplicada) – Programa de Pós-Graduação em Computação Aplicada, Instituto de Ciências Exatas, Universidade de Brasília, Brasília, DF, 2019.

SCHENFELD, M. C. **Fog e Edge Computing: Uma Arquitetura Híbrida em um Ambiente de Internet das Coisas**. 2017. Dissertação (Mestrado em Ciência da Computação)

– Programa de Pós-Graduação em Ciência da Computação, Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2017.

SETHI, P.; SARANGI, S. R. Internet of Things: Architectures, Protocols, and Applications. **Journal of Electrical and Computer Engineering**, v. 2017, p. 6-30, 2017.

SIEMENS. **FC1002, FC1004 Control Units**. Zug: Siemens, 2010. Disponível em: <https://www.downloads.siemens.com/download-center/Download.aspx?pos=download&fct=getasset&id1=A6V10301336>. Acesso em: 12 jul. 2020.

SOUTO, S. F. **Geração Automática de Testes com Objetos Mock Baseados em Interações**. 2010. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Ciência da Computação, Centro de Engenharia Elétrica e Informática, Universidade Federal de Campina Grande, Campina Grande, 2010.

SILVA, D. C. **Um sistema de gestão da segurança do trabalho alinhado à produtividade e à integridade dos colaboradores**. 2007. Trabalho de Conclusão de Curso (Graduação em Engenharia de Produção) – Faculdade de Engenharia, Universidade Federal de Juiz de Fora, Juiz de Fora, 2006.

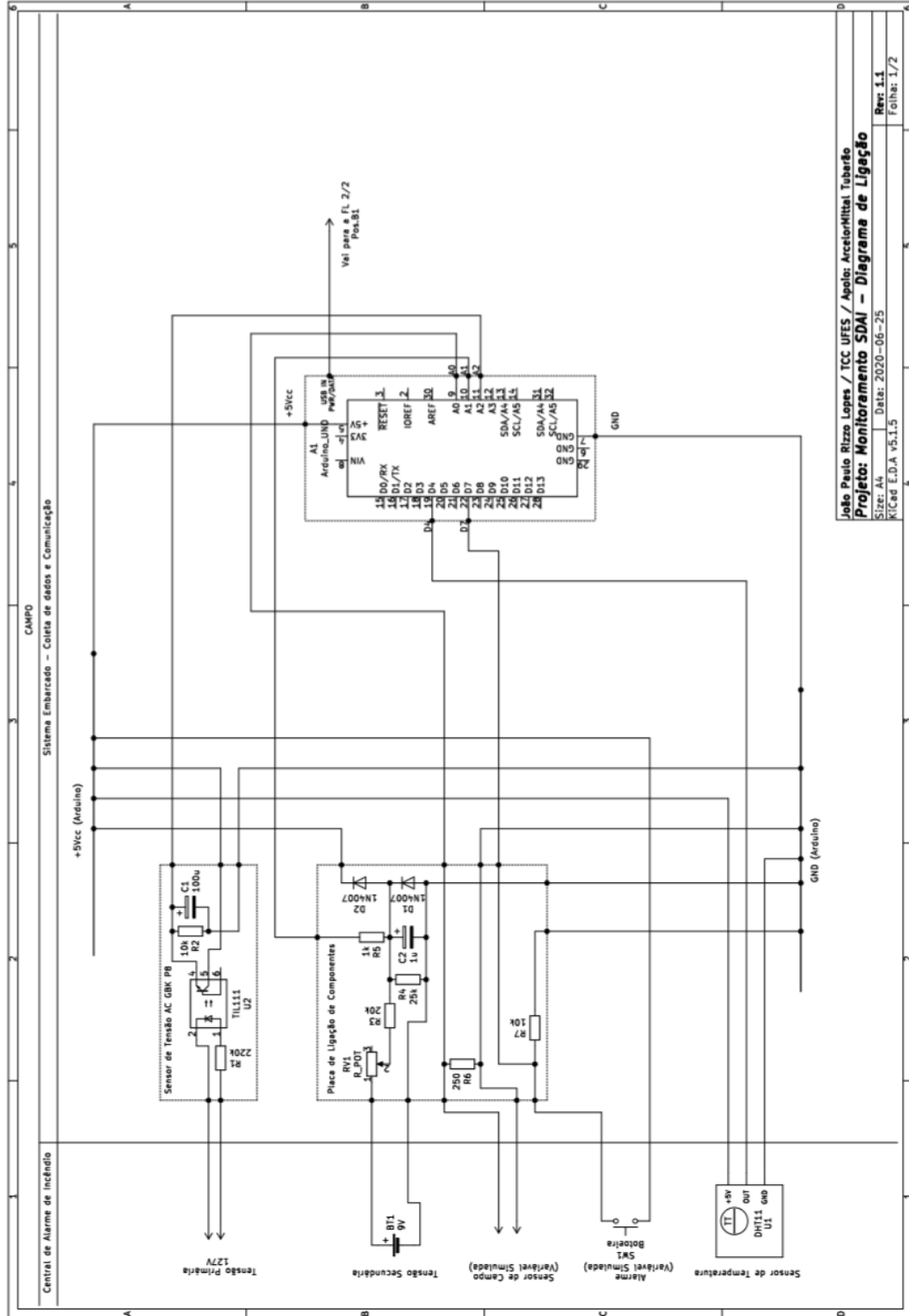
VOAS, J.; ZHANG, J. Cloud computing: New wine or just a new bottle? **IT professional**, v. 11, n. 2, p. 15-17, 2009.

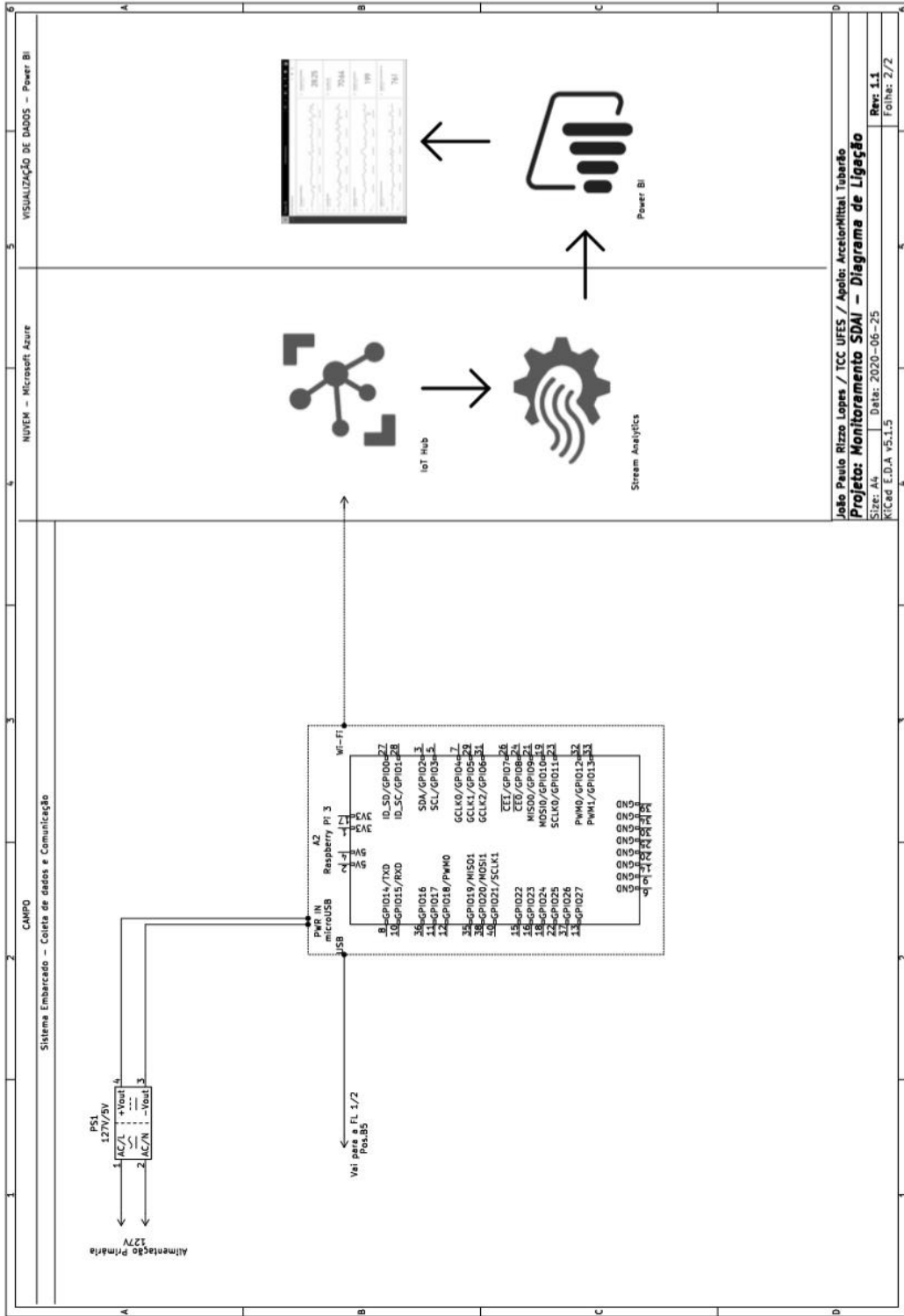
WATSON, H. J.; WIXOM, B. H. The current state of business intelligence. **Computer**, v. 40, n. 9, 2007.

XAVIER FILHO, S. **Análise dos indicadores de segurança do trabalho em reformas de altos fornos em siderúrgicas**. 2010. Dissertação (Mestrado em Engenharia Civil) – Programa de Pós-Graduação em Engenharia Civil, Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2010.

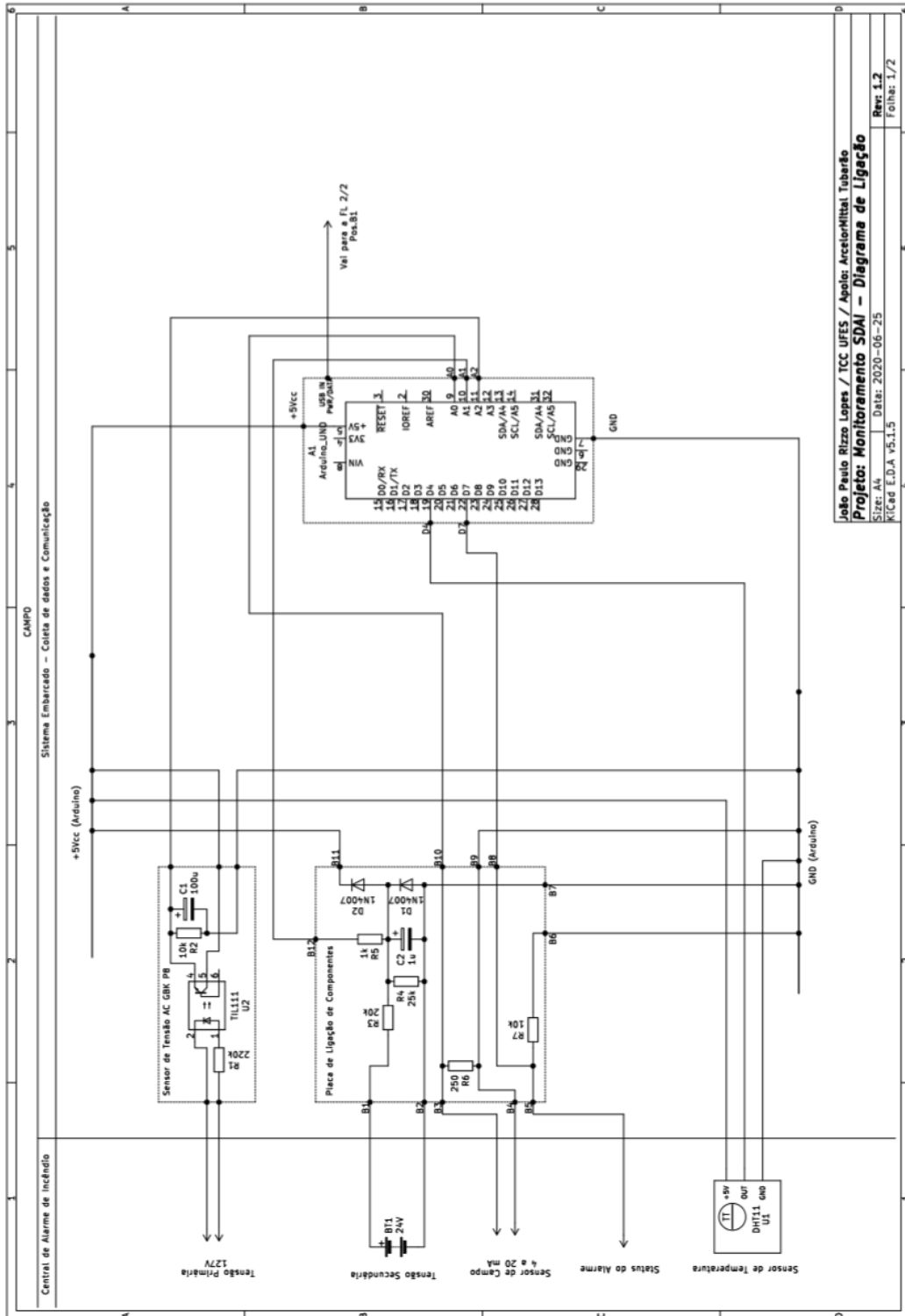
XU, M.; DAVID, J. M.; KIM, S. H. The Fourth Industrial Revolution: Opportunities and Challenges. **International Journal of Financial Research**, v. 9, n. 2, p. 90-95, 2018.

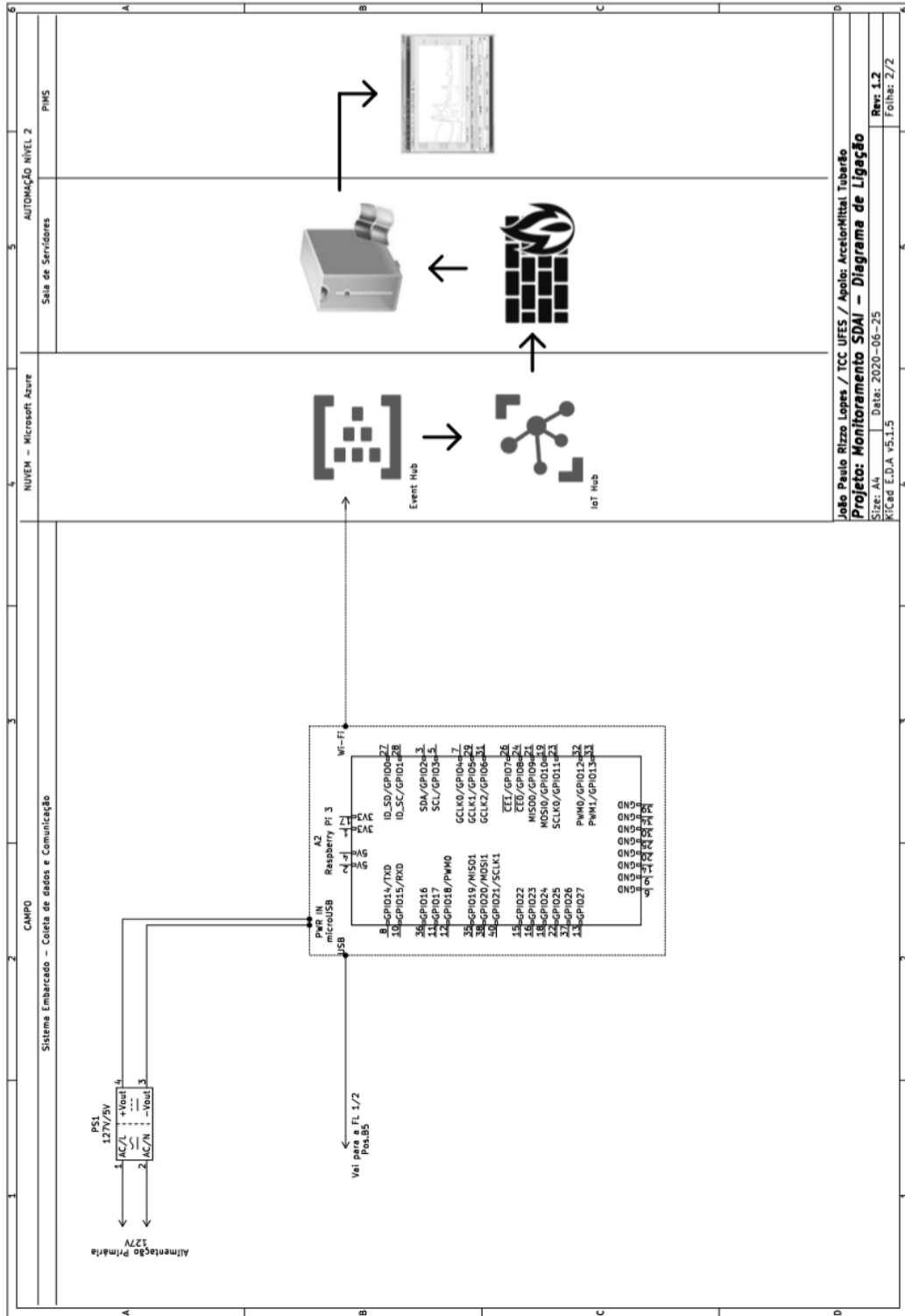
APÊNDICE A – DIAGRAMA DO PROJETO (PROTÓTIPO IMPLEMENTADO)





APÊNDICE B – DIAGRAMA DO PROJETO (ESCOPO ORIGINAL)





APÊNDICE C – CÓDIGO ELABORADO (ARDUINO)

```

1. #include <ArduinoJson.h>
2. #include <Adafruit_Sensor.h>
3. #include <DHT.h>
4. #define DHT11PIN 4
5. #define DHT11TYPE DHT11
6.
7. DHT dht11(DHT11PIN, DHT11TYPE);
8. unsigned long previousMillis = 0;
9. const long interval = 11000;
10. const int amostragemTensao = 1000;
11. float valorLidoTensao = 0;
12. float mediaTotal = 0;
13. float valorTensaoCA;
14.
15. void setup() {
16.   Serial.begin(9600);
17.   dht11.begin();
18.   pinMode(7, INPUT);
19.   pinMode(4, INPUT);
20.   pinMode(A0, INPUT);
21.   pinMode(A1, INPUT);
22.   pinMode(A2, INPUT);
23. }
24.
25. void loop() {
26.
27.   float t11 = dht11.readTemperature();
28.
29.   float statusAlarme = digitalRead(7);
30.
31.   float tensaoBateriaRead = analogRead(A1);
32.   float tensaoBateria = ((int)(10*tensaoBateriaRead*9/1024))/10.0;
33.
34.   float conversorRead = analogRead(A0);
35.   float conversor42015_volts = conversorRead*5/1024;
36.   float conversor42015_mA =
37.   ((int)(100*conversor42015_volts*20/5))/100.0;
38.   ////////// Bloco de cálculo referente ao sensor de tensão AC //////////
39.
40.   valorTensaoCA = 0;
41.   mediaTotal = 0;
42.
43.   for(int i = 0; i < amostragemTensao; i++){
44.     valorLidoTensao = analogRead(A2);
45.     mediaTotal = mediaTotal + valorLidoTensao;
46.     delay(1);
47.   }
48.   mediaTotal = mediaTotal / amostragemTensao;
49.
50.   if((mediaTotal > 100) & (mediaTotal < 483)){
51.     valorTensaoCA = ((mediaTotal*5)/1023)*70.0;
52.
53.     /// 70.0 -> valor de calibração para tensão de 127V
54.   }
55.

```

```
56. ////////////// Reserva espaço em memória para o JSON //////////////
57.
58.   StaticJsonBuffer<200> jsonBuffer;
59.
60.   ////////////// Constrói o objeto JSON //////////////
61.
62.   JsonObject& root = jsonBuffer.createObject();
63.   root["statusAlarme"] = statusAlarme;
64.   root["tensaoBateria"] = tensaoBateria;
65.   JsonObject& dht11 = root.createNestedObject("dht11");
66.   dht11.set("temperatura", t11);
67.   root["conversor4_20ma_1_5v"] = conversor42015_mA;
68.   root["valorTensaoCA"] = valorTensaoCA;
69.
70.   unsigned long currentMillis = millis();
71.   if (currentMillis - previousMillis >= interval) {
72.     previousMillis = currentMillis;
73.
74.     ////////////// Escreve a string JSON na porta Serial //////////////
75.
76.     root.printTo(Serial);
77.     Serial.println();
78.   }
79.
80. }
```

APÊNDICE D – CÓDIGO ELABORADO (RASPBERRY PI)

```

1. ##### importa as bibliotecas necessárias #####
2. import requests
3. import serial
4. import logging
5. import json
6. from logging.handlers import RotatingFileHandler
7. from unittest import mock
8. import time
9. from base64 import b64encode, b64decode
10. from hashlib import sha256
11. from urllib.parse import quote, urlencode
12. from hmac import HMAC
13. import datetime
14. from pathlib import Path
15.
16. ##### configuração de parâmetros do logger #####
17. def setupLogger(loggerName, logFileName, logLevel, formatter):
18.     l = logging.getLogger(loggerName)
19.     filehandler = RotatingFileHandler(logFileName, maxBytes = 1024,
    backupCount = 1)
20.     filehandler.setFormatter(formatter)
21.     streamhandler = logging.StreamHandler()
22.     streamhandler.setFormatter(formatter)
23.     l.setLevel(logLevel)
24.     l.addHandler(filehandler)
25.     l.addHandler(streamhandler)
26.
27. ### Cria classe para contagem do número de exceções disparadas ###
28. class CallCounted:
29.     def __init__(self,method):
30.         self.method=method
31.         self.counter=0
32.
33.     def __call__(self,*args,**kwargs):
34.         self.counter+=1
35.         return self.method(*args,**kwargs)
36.
37. ##### Setup do logger #####
38. setupLogger('log1', 'Log_SDAI.txt', logging.ERROR,
    logging.Formatter('%(levelname)s: %(asctime)s %(message)s', "%d/%m/%Y
    %I:%M:%S"))
39. logger1 = logging.getLogger('log1')
40. logger1.exception = CallCounted(logger1.exception)
41. NumMaxErros = 3
42. setupLogger('log2', 'LogDados_SDAI.txt', logging.DEBUG,
    logging.Formatter('%(message)s'))
43. logger2 = logging.getLogger('log2')
44.
45. ##### Checa o número de erros #####
46. def comparaNumeroErros():
47.     print("Quantidade de erros: ", logger1.exception.counter)
48.     if (logger1.exception.counter > NumMaxErros):
49.         logger1.error("Ultrapassagem da quantidade máxima de erros
    de comunicação permitida")
50.     return 0

```

```

51.         else:
52.             return 1
53.
54. ##### Instancia porta serial #####
55. ser = serial.Serial('/dev/ttyACM0',9600)
56.
57. ##### Parâmetros do Azure IoT Hub #####
58. URI = ### divulgação suprimida para segurança da cloud
59. KEY = ### divulgação suprimida para segurança da cloud
60. IOT_DEVICE_ID = ### divulgação suprimida para segurança da cloud
61. POLICY = ### divulgação suprimida para segurança da cloud
62.
63. ##### Gera token de conexão #####
64. def generate_sas_token():
65.     expiry=3600
66.     ttl = time.time() + expiry
67.     sign_key = "%s\n%d" % ((quote(URI)), int(ttl))
68.     signature = b64encode(HMAC(b64decode(KEY),
69.     sign_key.encode('utf-8'), sha256).digest())
70.     rawtoken = {
71.         'sr' : URI,
72.         'sig': signature,
73.         'se' : str(int(ttl))
74.     }
75.     rawtoken['skn'] = POLICY
76.     return 'SharedAccessSignature ' + urlencode(rawtoken)
77. ##### Envia mensagem para a nuvem #####
78. def send_message(token, message):
79.     url = 'https://{0}/devices/{1}/messages/events?api-
80.     version=2016-11-14'.format(URI, IOT_DEVICE_ID)
81.     headers = {
82.         "Content-Type": "application/json",
83.         "Authorization": token
84.     }
85.     response = requests.post(url, data=message, headers=headers)
86. ##### Lê o JSON na porta Serial #####
87. def lerDadosPortaSerial():
88.     read_serial = ser.readline().decode('utf-8').replace('\n',
89.     '').replace('\r', '')
90.     dados = json.loads(read_serial)
91.     now = datetime.datetime.now()
92.     timestamp = str(now.strftime("%H:%M:%S %d/%m/%Y"))
93.     dados["timestamp"] = timestamp
94.     return json.dumps(dados)
95. ##### Escreve no arquivo buffer .txt #####
96. def send_messagesBuffer():
97.     with open('LogDados_SDAI.txt') as json_file:
98.         for line in json_file:
99.             dados = json.loads(line)
100.            dados = json.dumps(dados)
101.            send_message(token,dados)
102.
103. ##### Função principal #####
104. if __name__ == '__main__':
105.     try:
106.         token = generate_sas_token()
107.         tamanho = Path('LogDados_SDAI.txt').stat().st_size

```

```
108.         if (tamanho > 0):
109.             send_messagesBuffer()
110.             print('buffer enviado')
111.         while True:
112.             message = lerDadosPortaSerial()
113.             send_message(token, message)
114.     except requests.exceptions.HTTPError as errh:
115.         logging.exception('HTTP Error')
116.     except requests.exceptions.ConnectionError as errc:
117.         logging.exception("Connection Error")
118.     except requests.exceptions.Timeout as errt:
119.         logging.exception("Timeout Error")
120.     except requests.exceptions.RequestException as err:
121.         logger1.exception("Other Error")
122.         continuar = comparaNumeroErros()
123.         if (continuar == 1):
124.             while True:
125.                 message = lerDadosPortaSerial()
126.                 logger2.debug(message)
```

APÊNDICE E – CÓDIGO DE TESTES

```

1. ##### importa as bibliotecas necessárias #####
2. import requests
3. import serial
4. import logging
5. import json
6. from logging.handlers import RotatingFileHandler
7. from unittest import mock
8. import time
9. from base64 import b64encode, b64decode
10. from hashlib import sha256
11. from urllib.parse import quote, urlencode
12. from hmac import HMAC
13. import datetime
14. from pathlib import Path
15.
16. ##### configuração de parâmetros do logger #####
17. def setupLogger(loggerName, logFileName, logLevel, formatter):
18.     l = logging.getLogger(loggerName)
19.     filehandler = RotatingFileHandler(logFileName, maxBytes = 1024,
    backupCount = 1)
20.     filehandler.setFormatter(formatter)
21.     streamhandler = logging.StreamHandler()
22.     streamhandler.setFormatter(formatter)
23.     l.setLevel(logLevel)
24.     l.addHandler(filehandler)
25.     l.addHandler(streamhandler)
26.
27. ### Cria classe para contagem do número de exceções disparadas ###
28. class CallCounted:
29.     def __init__(self,method):
30.         self.method=method
31.         self.counter=0
32.
33.     def __call__(self,*args,**kwargs):
34.         self.counter+=1
35.         return self.method(*args,**kwargs)
36.
37. ##### Setup do logger #####
38. setupLogger('log1', 'Log_SDAI.txt', logging.ERROR,
    logging.Formatter('%(levelname)s: %(asctime)s %(message)s', "%d/%m/%Y
    %I:%M:%S"))
39. logger1 = logging.getLogger('log1')
40. logger1.exception = CallCounted(logger1.exception)
41. NumMaxErros = 3
42. setupLogger('log2', 'LogDados_SDAI.txt', logging.DEBUG,
    logging.Formatter('%(message)s'))
43. logger2 = logging.getLogger('log2')
44.
45. ##### Checa o número de erros #####
46. def comparaNumeroErros():
47.     print("Quantidade de erros: ", logger1.exception.counter)
48.     if (logger1.exception.counter > NumMaxErros):
49.         logger1.error("Ultrapassagem da quantidade máxima de erros
    de comunicação permitida")
50.         return 0
51.     else:

```



```

52.         return 1
53.
54.
55. ##### Instancia porta serial #####
56. ser = serial.Serial('/dev/ttyACM0',9600)
57.
58. ##### Objetos Mock para o teste unitário de falha de comunicação ###
59. ##### O objeto substitui a requisição POST #####
60. ##### Primeiro teste: HTTP Error #####
61. mock_http = mock.Mock(side_effect=requests.exceptions.HTTPError)
62. ##### Segundo teste: Connection Error #####
63. mock_conn =
    mock.Mock(side_effect=requests.exceptions.ConnectionError)
64. ##### Terceiro teste: Timeout Error #####
65. mock_timeout = mock.Mock(side_effect=requests.exceptions.Timeout)
66. ##### Quarto teste: Request Exception #####
67. mock_reqexc =
    mock.Mock(side_effect=requests.exceptions.RequestException)
68.
69. ##### Parâmetros do Azure IoT Hub #####
70. URI = ### divulgação suprimida para segurança da cloud
71. KEY = ### divulgação suprimida para segurança da cloud
72. IOT_DEVICE_ID = ### divulgação suprimida para segurança da cloud
73. POLICY = ### divulgação suprimida para segurança da cloud
74.
75. ##### Gera token de conexão #####
76. def generate_sas_token():
77.     expiry=3600
78.     ttl = time.time() + expiry
79.     sign_key = "%s\n%d" % ((quote(URI)), int(ttl))
80.     signature = b64encode(HMAC(b64decode(KEY),
    sign_key.encode('utf-8'), sha256).digest())
81.     rawtoken = {
82.         'sr' : URI,
83.         'sig': signature,
84.         'se' : str(int(ttl))
85.     }
86.     rawtoken['skn'] = POLICY
87.     return 'SharedAccessSignature ' + urlencode(rawtoken)
88.
89. ##### Envia mensagem para a nuvem #####
90. def send_message(token, message):
91.     url = 'https://{0}/devices/{1}/messages/events?api-
    version=2016-11-14'.format(URI, IOT_DEVICE_ID)
92.     headers = {
93.         "Content-Type": "application/json",
94.         "Authorization": token
95.     }
96.     response = requests.post(url, data=message, headers=headers)
97.
98. ##### Lê o JSON na porta Serial #####
99. def lerDadosPortaSerial():
100.    read_serial = ser.readline().decode('utf-8').replace('\n',
    '').replace('\r', '')
101.    dados = json.loads(read_serial)
102.    now = datetime.datetime.now()
103.    timestamp = str(now.strftime("%H:%M:%S %d/%m/%Y"))
104.    dados["timestamp"] = timestamp
105.    return json.dumps(dados)
106.

```

```
107. ##### Escreve no arquivo buffer .txt #####
108. def send_messagesBuffer():
109.     with open('LogDados_SDAI.txt') as json_file:
110.         for line in json_file:
111.             dados = json.loads(line)
112.             dados = json.dumps(dados)
113.             print(dados)#send_message(token,dados)
114.
115. ##### Função principal #####
116. if __name__ == '__main__':
117.     try:
118.         token = generate_sas_token()
119.         tamanho = Path('LogDados_SDAI.txt').stat().st_size
120.         if (tamanho > 0):
121.             mock_http() ##### comentar para testar os outros
122.             ##### mock_conn() ##### descomentar para testar
123.             ##### mock_timeout() ##### descomentar para testar
124.             ##### mock_reqexc() ##### descomentar para testar
125.             print('buffer enviado')
126.             while True:
127.                 message = lerDadosPortaSerial()
128.                 mock_http() ##### comentar para testar os outros
129.                 ##### mock_conn() ##### descomentar para testar
130.                 ##### mock_timeout() ##### descomentar para testar
131.                 ##### mock_reqexc() ##### descomentar para testar
132.             except requests.exceptions.HTTPError as errh:
133.                 logging.exception('HTTP Error')
134.             except requests.exceptions.ConnectionError as errc:
135.                 logging.exception("Connection Error")
136.             except requests.exceptions.Timeout as errt:
137.                 logging.exception("Timeout Error")
138.             except requests.exceptions.RequestException as err:
139.                 logger1.exception("Other Error")
140.                 continuar = comparaNumeroErros()
141.                 if (continuar == 1):
142.                     while True:
143.                         message = lerDadosPortaSerial()
144.                         logger2.debug(message)
145.
```

APÊNDICE F – CADERNO DE TESTES EM BANCADA

CADERNO DE TESTES Nº 001

Projeto	Monitoramento de uma Central de Detecção e Alarme de Incêndio via Computação em Nuvem – Protótipo
Etapa	Teste em Bancada
Ferramentas	Multímetro, <i>Notebook</i>
Responsável	João Paulo

ITENS SOB TESTE:

Item 1	Circuito eletrônico em <i>protoboard</i>
Item 2	Circuito eletrônico integrado ao <i>Arduino</i> (Bloco de Sensoriamento)
Item 3	Bloco de Sensoriamento integrado ao <i>Raspberry Pi</i>

Circuito eletrônico em *protoboard*:

DESCRIÇÃO	RESULTADO	DIA
Inspeção visual	OK	14/05/2020
Testes de continuidade	OK	14/05/2020

Bloco de Sensoriamento:

DESCRIÇÃO	RESULTADO	DIA
Calibração de sensores	OK	19/05/2020
Tratamento dos sinais dos sensores por <i>software</i>	OK	19/05/2020

Bloco de Sensoriamento integrado ao *Raspberry Pi*:

DESCRIÇÃO	MENSAGEM	RESULTADO	DIA
Teste de <i>software</i> #1 – <i>Http Error</i>	ERROR: 03/07/2020 11:34:54 HTTP Error	OK	03/07/2020
Teste de <i>software</i> #2 – <i>Connection Error</i>	ERROR: 03/07/2020 11:38:35 Connection Error	OK	03/07/2020
Teste de <i>software</i> #3 – <i>Timeout Error</i>	ERROR: 03/07/2020 11:42:13 Timeout Error	OK	03/07/2020
Teste de <i>software</i> #4 – <i>Request Exception</i>	ERROR: 03/07/2020 11:45:00 Other Error	OK	03/07/2020
Teste funcional – envio de dados para plataforma de nuvem	N/A	OK	04/07/2020
Teste funcional – integração com <i>dashboard</i> do <i>Power BI</i>	N/A	OK	04/07/2020

APÊNDICE G – CRONOGRAMA DO PROJETO

