Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Sistema preditivo para otimização do consumo de energia do sistema de refrigeração em *data center* de pequeno porte

Henrique Tomé Damasio

CASCAVEL
2019

HENRIQUE TOMÉ DAMASIO

Sistema preditivo para otimização do consumo de energia do sistema de refrigeração em *data center* de pequeno porte

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Dr. Marcio Seiji Oyamada

CASCAVEL
2019

HENRIQUE TOMÉ DAMASIO

Sistema preditivo para otimização do consumo de energia do sistema de refrigeração em *data center* de pequeno porte

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. Marcio Seiji Oyamada (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Guilherme Galante
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Luiz Antonio Rodrigues
Colegiado de Ciência da Computação,
UNIOESTE

Cascavel, 18 de dezembro de 2019

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, meus irmãos e minha irmã, mas principalmente aos meus pais Aldo e Adriana pelo apoio, por acreditarem em mim, por estarem ao meu lado nos momentos felizes e principalmente nos difíceis, por financiarem esses anos de graduação para que eu tivesse uma educação de qualidade.

Também agradeço a todos os meus professores, pelo conhecimento adquirido, mas especialmente ao meu orientador Marcio Seiji Oyamada, pela sua competência, por estar sempre presente nos momentos de dúvida e por nunca medir esforços para me auxiliar. Aos membros da minha banca Guilherme Galante e Luiz Antonio Rodrigues pelo tempo gasto e pelos conhecimentos transmitidos.

A todos meus amigos, principalmente aos que fiz durante a graduação, em especial ao Gilberto Antunes Monteiro Junior e Brendo Peres Bizetto por todas as noites que passamos estudando para provas e trabalhos e pelos momentos felizes e tristes compartilhados durante todos esses anos, sem eles tenho certeza que seria muito mais difícil passar por esse anos da graduação.

Lista de Figuras

1.1 Faixas recomendadas e permitidas pela ASHRAE (ASHRAE, 2012) 2
1.2 Funcionamento do aparelho de ar condicionado dentro do *data center* 4

2.1 Microcontrolador Arduino UNO. Fonte (ARDUINO, 2019) 6
2.2 Microcontrolador Arduino Nano. Fonte (FILIPEFLOP, 2019) 7
2.3 Módulo ESP8266. Fonte (CIRCUITO, 2018) 7
2.4 Microcontrolador NodeMCU. Fonte (NODEMCU, 2019) 8
2.5 Microcontrolador Wemos D1. Fonte (FLIPEPLOP, 2019a) 8
2.6 Pinagem dos sensores DHT11 à direita e DHT22 à esquerda. Fonte (GAJERA, 2017) 10
2.7 Protocolo dos sensores DHT11 e DHT22. Fonte (GAJERA, 2017) 11
2.8 Pinagem sensor TMP36. Fonte (DEVICES, 2010) 12
2.9 Sensor SCT-013. Fonte (FLIPEPLOP, 2019b) 13
2.10 Módulo sensor de tensão ZMPT101B. Fonte (ROBóTICA, 2019) 15
2.11 Exemplo de tela da aplicação ThingSpeak. 16
2.12 Exemplo de *dashboard* na aplicação Ubidots. Fonte (PELAEZ, 2018) 17
2.13 Formação das árvores em uma *random forest*. Adaptado de (DENG, 2018) . . . 20
2.14 Exemplo de como é realizada a decisão da *random forest*. Adaptado de (DENG, 2018) 20
2.15 Exemplo de um neurônio artificial. Adaptado de (CERRI et al., 2011) 22
2.16 Exemplo de uma Multi-Layer Perceptron. Adaptado de (JANTZEN, 1998) . . . 22
2.17 Sistema em lógica fuzzy proposto por (PURWANTO; UTAMI; PRAMONO, 2018a) 25
2.18 *Data center* Keihanna. Adaptado de (TARUTANI et al., 2015) 27

3.1 Representação da sala dos servidores 32
3.2 Visão geral do sistema implementado 33
3.3 Circuito montado para a captação dos dados externos ao *data center* 34
3.4 Circuito montado para a captação dos dados internos do *data center* 34
3.5 Circuito real construido para a captação dos dados internos do *data center* . . . 35
3.6 Canal Thingspeak com os dados internos do *data center* 35
3.7 Canal Thingspeak com os dados externos ao *data center* 36
3.8 Circuito para a atuação no *data center* 39
3.9 Atuador do *data center* 40

4.1 Faixa de temperatura entre o SI e o SR 47
4.2 Temperatura dentro do *data center* durante o periodo de atuação do SR 48
4.3 Temperatura dentro do *data center* durante o periodo de atuação do SRP 49
4.4 Temperatura dentro do *data center* durante o periodo de atuação dos sistemas de atuação (SR e SRP) 50

A.1 Funcionamento do ar condicionado 56

Lista de Tabelas

2.1 Comparação entre os microcontroladores 9
2.2 Comparação entre os sensores de temperatura e umidade 13
2.3 *Datasheet* do sensor SCT-013. Fonte (YHDC, 2019) 14
2.4 Comparação entre Thingspeak e Ubidots. Adaptado de (KALAITHASAN; RADZI; ABIDIN, 2018) 18

3.1 Exemplo de tupla de dados para treino e validação do modelo de predição . . . 37
3.2 Exemplo de tupla a ser utilizada na aplicação *python* 38
3.3 Exemplo de tupla com os dados normalizados 38
3.4 Regras do sistema SR 40
3.5 Regras do sistema SRP 41

4.1 Coeficiente de correlação de cada modelo para cada iteração 44
4.2 Valores para os modelos de predição criados pela plataforam AutoML 46

A.1 Representação dos dados crus 54
A.2 Representação dos dados parcialmente tratados 55
A.3 Representação dos dados totalmente tratados 56
A.4 Resultados do treinamento dos preditores 58
A.5 Resultados do treinamento da *random forest* 59

Sumário

Lista de Figuras **v**

Lista de Tabelas **vii**

Sumário **viii**

Resumo **x**

1 Introdução **1**

2 Referencial Teórico **5**

2.1 Microcontroladores 5

2.1.1 Arduino UNO 6

2.1.2 Arduino Nano 3.0 6

2.1.3 ESP8266 7

2.2 Sensores 9

2.2.1 DHT11 10

2.2.2 DHT22 11

2.2.3 TMP36 12

2.2.4 SCT-013 13

2.2.5 ZMPT101B 14

2.3 Plataformas *cloud* IoT 15

2.4 Técnicas de *Machine Learning* 18

2.4.1 Regressão Linear 19

2.4.2 *Random Forest* 19

2.4.3 Rede Neural 21

2.4.3.1 *Multilayer Perceptron* 21

2.5 Soluções de Monitoramento de Temperatura em *Data Centers* 23

2.6 Soluções de Controle 23
2.6.1 Soluções baseadas em regras 23
2.6.2 Soluções baseadas em aprendizado de máquina 27
2.7 Considerações 29
3 Arquitetura **31**
3.1 Proposta de Sistema 32
3.1.1 Monitoramento 33
3.1.1.1 Componentes de Harware 33
3.1.1.2 Componentes de Software 34
3.1.2 Análise dos dados capturados 36
3.1.2.1 Preparação dos dados 37
3.1.2.2 Aplicação *python* 37
3.1.2.3 Plataforma AutoML 38
3.1.3 Atuação 39
3.1.3.1 Sistemas de Regras 40
4 Resultados Experimentais **43**
4.1 Modelo preditivo 43
4.2 Comparação de gasto energético 46
5 Conclusões **52**
A Estudo Preliminar **54**
A.1 Preparação do dados 54
A.2 Algoritmos utilizados 57
Referências **60**

Resumo

O gasto energético com resfriamento em geral é o que mais cresceu nos últimos anos, e quando se trata de *data centers* tal gasto pode representar mais de 50% do consumo total. O objetivo deste trabalho é o desenvolvimento de um modelo de atuação no sistema de refrigeração que busca reduzir o consumo energético. Utilizando-se técnicas de *machine learning* buscou-se identificar um modelo de predição preciso o suficiente para realizar a atuação. Três diferentes atuadores foram avaliados neste trabalho, o primeiro utilizando o aparelho de ar condicionado em 22^{o}C no modo automático, o segundo com um sistema de regras fixo, e um terceiro com um sistema de regras apoiado pelo modelo de predição. Os resultados obtidos indicaram que os sistemas de atuação desenvolvidos se sobressaem ao modo automático do condicionador de ar, contudo o sistema de atuação que se utiliza do modelo preditivo se mostra mais eficiente se comparado ao sistema de atuação de regras obtendo uma redução de aproximadamente 15% no consumo energético.

Palavras-chave: Machine Learning, Consumo Energético, Predição, Modelos de Atuação.

Capítulo 1

Introdução

O gasto energético com resfriamento é o que apresenta o crescimento mais rápido, já que sua demanda elétrica mais que triplicou entre 1990 e 2018, para cerca de 2 mil terawatts-hora (TWh) de eletricidade. Embora os condicionadores de ar com maior eficiência estejam disponíveis atualmente, a maioria das pessoas compra condicionadores de ar que são de duas a três vezes menos eficientes (DELMASTRO, 2019).

O ano de 2018 foi um ano excepcionalmente quente em muitas partes do mundo, e a energia usada para resfriamento foi estimada em 5% mais alta, no âmbito global, do que em 2017. Mais de 1,6 bilhão de unidades de ar condicionado estão agora em operação em todo o mundo, tornando o resfriamento a nova demanda elétrica no mundo (DELMASTRO, 2019).

A crescente demanda por resfriamento está afetando a geração de energia e a capacidade de distribuição, especialmente durante períodos de pico de demanda e eventos extremos de calor. O resfriamento em edifícios é responsável por 50% ou mais da demanda elétrica de pico (DELMASTRO, 2019).

A energia usada pelos equipamentos de tecnologia da informação (TI) e pelos equipamentos de suporte ao *data center* continua aumentando, à medida que a proliferação de *data centers* continua inabalável. Para conter os custos crescentes de computação, os administradores de *data centers* estão recorrendo a medidas de redução de custos, como não controlar rigidamente os níveis de temperatura e umidade e, em muitos casos, instalando economizadores com risco associado de contaminação de partículas e gases em seus *data centers* (SINGH et al., 2015).

A 1ª edição do Guia de Diretrizes Térmicas da ASHRAE (sigla para *American Society of Heating, Refrigerating and Air-Conditioning Engineers*) para Ambientes de Processamento de Dados, livro da série Datacom, publicado em 2004, definiu a faixa de temperatura-umidade

na qual os fabricantes de equipamentos de TI garantem que seus equipamentos operem com confiabilidade (ASHRAE, 2004). A segunda edição do livro, publicada em 2008, ampliou o intervalo recomendado para permitir um aumento no número de horas que os *data centers* podem operar no modo econômico do ar condicionado. Definiram a faixa de temperatura-umidade recomendada como: a faixa de temperatura de 18 à 27°C, ponto de orvalho[1] de 5,5 a 15°C e umidade relativa menor que 60% (ASHRAE, 2008).

No tocante à economia de energia do *data center*, além da faixa recomendada expandida, há faixas permitidas às quais o equipamento de TI pode ser exposto por curtos períodos de duração e frequência indefinidas. A Figura 1.1 mostra as faixas recomendadas e permitidas pela ASHRAE definidas na 3ª edição do Guia de Diretrizes Térmicas da ASHRAE para Ambientes de Processamento de Dados, da série Datacom, publicado em 2012 (ASHRAE, 2012).

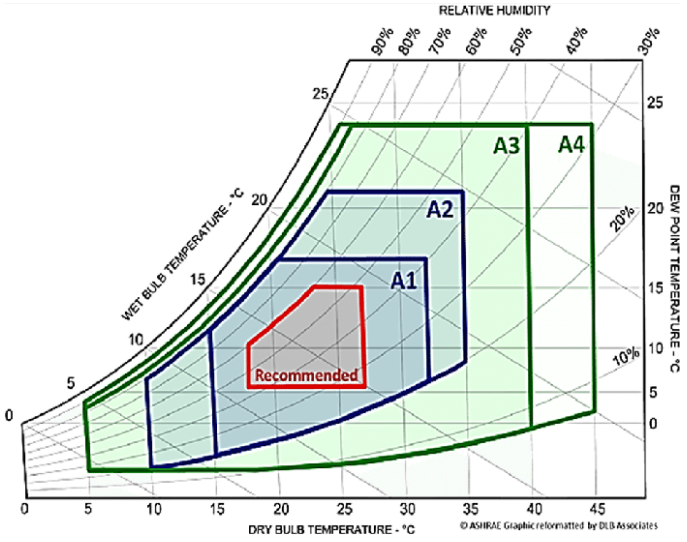

Figura 1.1: Faixas recomendadas e permitidas pela ASHRAE (ASHRAE, 2012)

A ASHRAE subdivide os equipamentos em 4 subclasses de A1 até A4, sendo que A1 representa equipamentos que podem operar em uma faixa de 15 à 32 graus Celsius com uma umidade relativa de 20% à 80%. A2 são os equipamentos que podem operar de 10 à 35 graus Celsius e com uma umidade relativa de 20% à 80%. Já os equipamentos classificados na classe A3 podem operar em temperaturas de 5 à 40 graus Celsius e com uma umidade relativa de 8% à 85%, e finalmente os equipamentos da classe A4 que podem operar em temperaturas de 5 à 45 graus

[1] Temperatura até a qual uma parcela de ar deve ser resfriada, a uma pressão constante e também uma quantidade de vapor d'água constante, para que haja saturação (SOCIETY, 2015)

Celsius e com uma umidade relativa de 8% à 90%.

Para manter essas temperaturas acaba-se gerando um grande gasto energético por parte dos sistemas de refrigeração, que em sua maior parte acaba sendo feito através de sistemas de ar-condicionado.

Este custo energético gerado pelo sistema de refrigeração acaba por ser excessivo, podendo chegar a até 50% do gasto total de energia do *data center* (DAYARATHNA; WEN; FAN, 2016). Em *data centers* menores o consumo não chega a ser tão expressivo, no entanto em alguns casos pode ultrapassar, a marca de 40% do gasto energético total (CHEUNG et al., 2014).

Para minimizar estes impactos, existem técnicas que empregam internet das coisas, do inglês, *internet of things*, para o monitoramento da temperatura da sala dos servidores através de sensores, e para o controle do funcionamento do ar-condicionado através de atuadores.

Na Figura 1.2 tem-se a representação de como é o funcionamento do aparelho de ar condicionado na sala de servidores do curso de ciência da computação. A informação em **(A)** indicam o consumo de corrente pelo aparelho de ar condicionado, enquanto que a informação apresentada em **(B)** representa a temperatura interna da sala dos servidores, e por fim a informação apresentada em **(C)** representa a temperatura externa a sala dos servidores.

É de extrema importância que os aparelhos de ar condicionado funcionem adequadamente, principalmente em *data centers* onde seu uso é constante. Problemas como o apresentados na Figura 1.2 devem ser evitados, já que para um *data center* de pequeno porte como o estudado neste trabalho o aparelho de ar condicionado não deveria estar em uso durante a madrugada, o que ocorre devido ao mesmo estar em modo automático e portanto não utilizando de toda a faixa de temperatura permitida ao *data center*, se a faixa em que o aparelho opera for flexibilizada então o mesmo não será acionado em momentos que não deveria.

Dessa forma, o objetivo principal deste trabalho é monitorar a temperatura e umidade dentro do *data center* localizado na sala de coordenação de laboratórios no bloco de Ciência da Computação da Universidade Estadual do Oeste do Paraná (UNIOESTE)- *Campus* Cascavel, e criar um modelo preditivo de consumo de energia do aparelho de ar condicionado utilizando os dados coletados com técnicas de *machine learning*. De posse do modelo preditivo, atuar no controle da temperatura e no modo de operação do aparelho de ar condicionado, de forma a reduzir o consumo energético do aparelho.

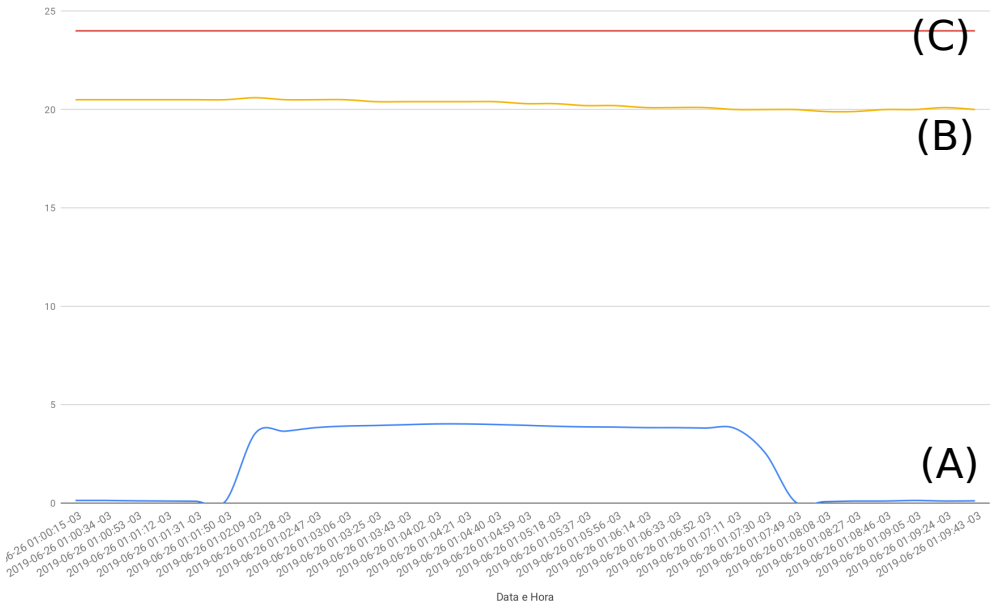

Figura 1.2: Funcionamento do aparelho de ar condicionado dentro do *data center*

O Capítulo 2 trata sobre o referencial teórico, abordando os principais componentes utilizados no monitoramento e atuação em *data centers*, também apresenta trabalhos já realizados na área.

No Capítulo 3 aborda-se o sistema já existente no *data center*, que realiza apenas o monitoramento da temperatura e o consumo de corrente do aparelho de ar condicionado. E apresenta-se a arquitetura do novo sistema de monitoramento e atuação.

No Capítulo 4 apresenta-se a construção do modelo preditivo, bem como a comparação de consumo energético entre os sistemas de atuação que foram utilizados. E por fim no Capítulo 5 uma análise do trabalho é realizada e propostas de trabalhos futuros são apresentadas.

Capítulo 2

Referencial Teórico

Este capítulo tem por objetivo apresentar os principais componentes eletrônicos, componentes de software e métodos utilizados no monitoramento e controle de *data centers*.

Inicialmente serão apresentados características dos equipamentos eletrônicos mais utilizados nesta área, tais como microcontroladores e sensores, após faremos uma análise de alguns componentes de software como as plataformas disponibilizadas na nuvem. Finalmente os trabalhos relacionados serão apresentados.

2.1 Microcontroladores

Um microcontrolador é um circuito integrado compacto projetado para controlar uma operação específica em um sistema embarcado. Um microcontrolador típico inclui processador, memória e pinos de entrada/saída (E/S) em um único chip. Geralmente são projetados para serem usados sem componentes adicionais pois podem interagir diretamente com sensores e outros componentes.

Às vezes referido como um controlador embarcado ou microcontrolador (MCU, abreviação do termo em inglês *microcontroller unit*), os microcontroladores são encontrados em vários setores e aplicações, incluindo automação residencial, manufatura, robótica, automação automotiva, iluminação, energia inteligente, automação industrial, comunicações e internet das coisas, do inglês *internet of things* (IoT).

2.1.1 Arduino UNO

A placa Arduino é um microcontrolador *open source* de grande popularidade e baixo custo. Possui uma variedade de placas de expansão (*shields*) que possibilitam a inclusão de recursos como comunicação sem fio, armazenamento de dados em cartão SD, acionamento de motores entre outros que se conectam a placa-mãe principal. Além da placa, o ecosistema Arduino é formado por uma IDE e também um conjunto extenso de bibliotecas que facilitam a utilização das placas de expansão. Por ser fácil de usar e possuir exemplos abundantes na IDE torna mais simples o processo de desenvolvimento de soluções IoT, sendo flexível e adaptável conforme a necessidade da aplicação. A placa Arduino UNO é apresentada na Figura 2.1, onde é possível verificar a presença de pinos analógicos e digitais, os detalhes dos pinos e recursos são apresentados na Tabela 2.1.

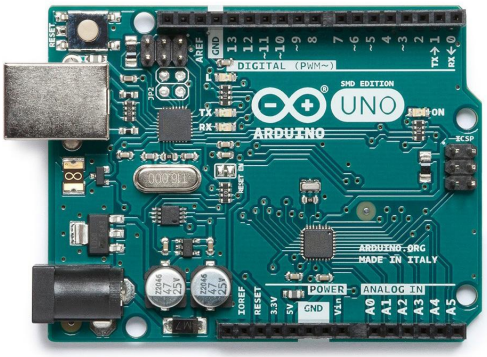

Figura 2.1: Microcontrolador Arduino UNO. Fonte (ARDUINO, 2019)

2.1.2 Arduino Nano 3.0

Esta placa pode ser considerada uma versão em miniatura do arduino UNO, abordado anteriormente, sua pinagem é muito similar a do UNO, mas sua comunicação com o computador apresenta uma diferença, uma vez que esta placa se utiliza de um USB mini-b. A principal vantagem desta placa em relação ao arduino UNO é o seu tamanho, com medidas de 1,85cm x 4,32cm. Não só a pinagem das placas é parecida, mas também as especificações de memória flash, SRAM e EEPROM. Este microcontrolador pode ser vizualizado na Figura 2.2.

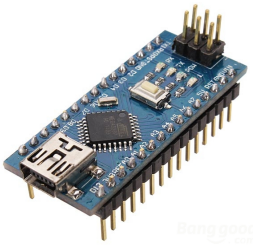

Figura 2.2: Microcontrolador Arduino Nano. Fonte (FILIPEFLOP, 2019)

2.1.3 ESP8266

O módulo ESP8266, que pode ser visualizado na Figura 2.3, é uma placa microcontroladora para desenvolvimento de soluções IoT com WIFI embutido.

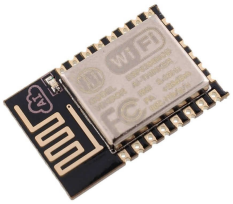

Figura 2.3: Módulo ESP8266. Fonte (CIRCUITO, 2018)

O ESP8266 é um SOC, abreviação do inglês *System on Chip*, que integra processador, placa de rede sem fio (WIFI) e memória. A programação do ESP pode ser realizada tanto na linguagem LUA quanto em Arduino, utilizando a Arduino IDE. A segunda opção facilita o desenvolvimento pois o desenvolvedor tem acesso a ampla gama de bibliotecas desenvolvidas no Arduino. O módulo WIFI pode tanto funcionar em modo cliente, quanto em modo AP (*access point*) ou *ad-hoc*. Em relação ao Arduino, tem como vantagem possuir WIFI embutido, no entanto disponibiliza apenas uma porta analógica, que em alguns tipos de projetos pode ser insuficiente.

Um diferencial deste microcontrolador é a possibilidade de fazer a programação da placa via OTA, abreviação do inglês *Over The Air*, ou seja, através do WiFi é possível enviar os códigos para a placa.

Com o sucesso alcançado pelo módulo ESP8266, os fabricantes começaram a adotá-lo como núcleo de outros microcontroladores, este é o caso do NodeMCU e do Wemos D1. No caso

do NodeMCU (Figura 2.4), o seu desenvolvimento se deu com o intuito de facilitar o uso do ESP8266, este microcontrolador adiciona a interface micro USB para programação, comunicação e alimentação.

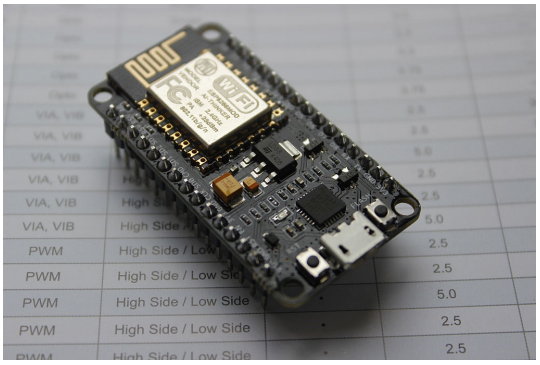

Figura 2.4: Microcontrolador NodeMCU. Fonte (NODEMCU, 2019)

Uma outra plataforma baseada no ESP8266 é o Wemos D1 (Figura 2.5), que utiliza as mesmas dimensões de uma placa Arduino Uno. No entanto, nota-se diferenças tais como um conector micro USB e o próprio módulo ESP8266, que é nativo da placa.

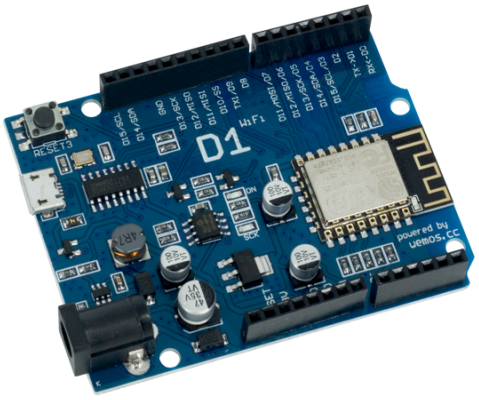

Figura 2.5: Microcontrolador Wemos D1. Fonte (FLIPEPLOP, 2019a)

A Tabela 2.1 resume as características entre os microcontroladores Arduino UNO, NodeMCU e Wemos D1, descritos nas seções anteriores. Nela podemos observar que tanto a memória RAM quanto a memória flash são muito inferiores no Arduino UNO/Nano em relação aos outros, isso acaba por limitar o microcontrolador a aplicações menos complexas que não

exija muita memória.

Por outro lado podemos notar que o mesmo possui mais pinos tanto digitais quanto analógicos em relação ao NodeMCU e ao Wemos D1, já que ambos são construídos tendo como base o ESP8266. Este número maior de pinos acaba por facilitar o uso de sensores, já que muitos acabam utilizando as entradas analógicas da placa, o que pode causar problemas para os microcontroladores baseados no ESP.

O Arduino Uno/Nano deixa a desejar quando o assunto é conexão com a internet pois este MCU não apresenta um módulo de conexão WiFi nativo. Mas esta deficiência da placa Arduino pode ser sanada pela aquisição de *shields* de conexão WiFi.

Parâmetros	Arduino UNO / NANO	NodeMCU	Wemos D1
Processador	ATMega328P	TenSilica Xtensa LX106	TenSilica Xtensa LX106
Tensão Operacional	5V	3.3V	3.3V
Clock	16MHz	80MHz	80MHz - 160MHz
Memória RAM	2KB	64KB	64KB
Memória Flash	32KB	4MB	4MB
EEPROM dedicada	1KB	-	-
Placa WiFi	-	IEEE 802.11 b/g/n	IEEE 802.11 b/g/n
Pinos Digitais de I/O	14	11	11
Pinos Analógicos	6	1	1

Tabela 2.1: Comparação entre os microcontroladores

2.2 Sensores

Para um monitoramento eficiente dos *data centers* o uso de sensores se torna indispensável, são eles que captam os dados desejados e os transmitem ao microcontrolador. Nesta seção são abordados alguns dos principais tipos de sensores utilizados no monitoramento das salas dos servidores, tais como, sensores de temperatura e umidade, sensores de corrente elétrica e sensores de tensão elétrica. Para cada tipo de sensor serão apresentados aqueles que são mais comumente utilizados e então uma comparação dos mesmos será realizada.

2.2.1 DHT11

O DHT11 é um sensor de baixo custo de temperatura e umidade que transforma sinais analógicos, do ambiente, em digitais. Seu design é uma combinação de um termístor e um sensor de umidade. Como apresentado por (GAJERA, 2017), este sensor apresenta uma pinagem bastante simples como demonstrado na Figura 2.6.

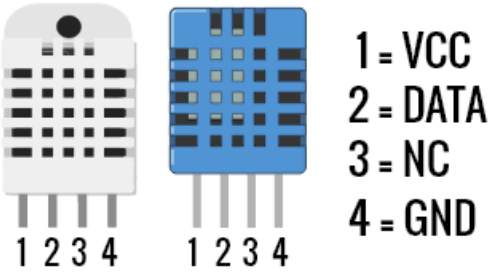

Figura 2.6: Pinagem dos sensores DHT11 à direita e DHT22 à esquerda. Fonte (GAJERA, 2017)

O pino de dados utilizado para comunicação entre o microcontrolador e o sensor DHT é mantido em alta tensão (3V à 5V varia de acordo com a placa) usando um resistor pull-up de 4.7K ou 10K. Isso é feito para colocar o barramento em um estado ocioso quando não houver comunicação. Um sinal de alta tensão contínuo na linha denota um estado inativo. O microcontrolador atua como o mestre de barramento e, portanto, é responsável por iniciar a comunicação (ou seja, leitura). O sensor de temperatura e umidade DHT sempre permanece como escravo e responde com dados quando o microcontrolador solicita. O seu protocolo é bastante simples e pode ser visto na Figura 2.7.

O processo de comunicação se dá da seguinte forma: quando a linha esta inativa o microcontrolador envia um sinal de baixa tensão (geralmente 0V) por 18ms, após isso ele envia um sinal de alta tensão por cerca de 20 a 40μs. O DHT então detecta esse sinal como um *start* do microcontrolador e responde enviando um sinal de baixa tensão por 80μs, em seguida o sensor envia um sinal de alta tensão por 80μs o que indica que ele esta pronto para enviar os dados, logo após enviará 40 bits de dados. Cada bit inicia com 50μs em baixa tensão seguido por 26-28μs em alta tensão para um "0"ou 70μs em alta tensão para um "1". Ao fim da comunicação, a linha é enviada para alta tensão pelo resistor pull-up e entra em estado inativo.

Quando o sensor envia os dados, os 40 bits são divididos em 5 bytes. Para o sensor DHT11

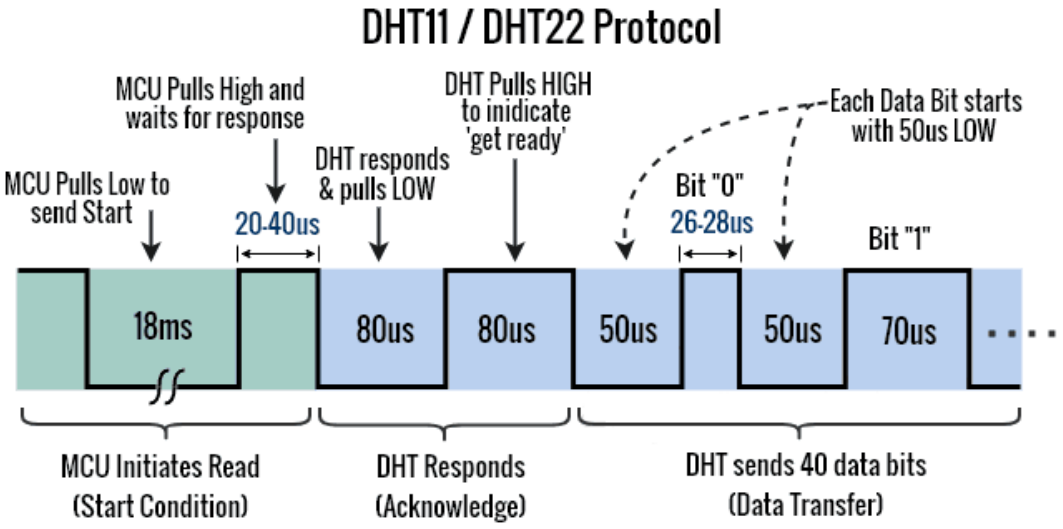

Figura 2.7: Protocolo dos sensores DHT11 e DHT22. Fonte (GAJERA, 2017)

o 2º e o 4º byte terão sempre o valor zero. O significado desses bytes é o seguinte: o 1º byte transmite os dados de umidade relativa (parte inteira), o 2º byte transmite os dados de umidade relativa (parte fracionária). O 3º byte transmite os dados de temperatura em graus Celsius (parte inteira), o 4º byte transmite os dados de temperatura (parte fracionária) e por fim o 5º byte transmite o *checksum* (últimos 8 bits de 1º byte + 2º byte + 3º byte + 4º byte)

Informações como faixa de atuação, precisão de medição, níveis de tensão e níveis de corrente podem ser encontradas na Tabela 2.2.

2.2.2 DHT22

O sensor DHT22 é na verdade uma versão um mais precisa do sensor DHT11. Este chip, como sua versão mais básica, combina um termístor e um sensor de umidade e permite a leitura dos dados utilizando sua saída serial digital.

Este sensor possui a mesma pinagem e protocolo apresentados por seu "irmão", como já apresentado nas Figuras 2.6 e 2.7. A diferença entre eles se dá pelo formato dos dados, uma vez que o DHT11 tem o 2º e 4º byte zerados, enquanto que o DHT22 utiliza esses bytes, tornando-o desta forma, mais preciso.

Outra diferença para o DHT11 é a maior faixa de temperatura e umidade que ele pode detectar e operar. Além disso, é mais preciso para medir a temperatura, apresentando um desvio de ± 0.5ºC. Além da maior precisão na temperatura, a umidade também pode ser medida com mais

precisão sendo então uma versão melhorada em comparação com o DHT11, como mostrado pela Tabela 2.2.

2.2.3 TMP36

O TMP36 é caracterizado por sua baixa tensão de operação, linearmente proporcional à temperatura em graus Celsius. Ele também não requer calibração. A Figura 2.8 mostra a pinagem do sensor.

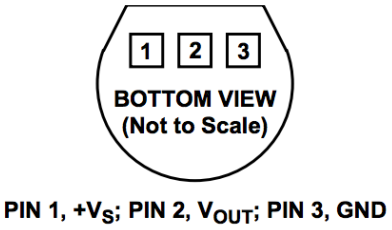

Figura 2.8: Pinagem sensor TMP36. Fonte (DEVICES, 2010)

O pino de saida deste sensor é conectado ao pino analógico da placa. Uma escala de mV para graus Celsius é fornecida para fazer a conversão da leitura em tensão para a temperatura em Celsius.

A Tabela 2.2 apresenta as especificações dos sensores de temperatura apresentados. Nela podemos analisar que, o sensor DHT11 apesar de mais básico se mostra com o melhor custo benefício, uma vez que faz leituras tanto de temperatura como de umidade.

Mas se for necessário um sensor mais acurado o DHT22 se demonstra mais interesante, além de possuir uma faixa operacional maior, a sua precissão é muito melhor se comparada a do DHT11 e do TMP36.

O TMP36 por sua vez se destaca quando não é necessário uma leitura de umidade, mas um sim um *range* maior na leitura da temperatura, já que este possui a maior faixa operacional entre os sensores analisados.

Para o monitoramento de um *data center* se faz necessário a aquisição tanto de temperatura, quanto de umidade, visto que são parâmetros que afetam diretamente a vida útil dos componentes eletrônicos, com isso em mente podemos concluir que os sensores DHT11 e DHT22 se mostram mais efetivos a esta utilização.

Parâmetros	DHT11	DHT22	TMP36
Erro máximo de medição	2ºC	0.5ºC	2ºC
Faixa operacional de temperatura	0 - +50ºC	-40ºC - +80ºC	-40ºC - +125ºC
Precisão temperatura	+/- 2ºC	+/- 0.5ºC	+/- 2ºC
Faixa operacional de umidade	20% - 80%	0% - 100%	-
Precisão umidade	5%	2% - 5%	-
Intensidade de corrente elétrica, μA	150	50	50
Nivel de tensão elétrica	3-5.5V	3.3-6V	2.7-5.5V
Média de preço em dólares	5	9	5

Tabela 2.2: Comparação entre os sensores de temperatura e umidade

2.2.4 SCT-013

O sensor de corrente SCT-013 é um sensor não invasivo e, portanto, tem como principal vantagem o fato de não precisar de contato elétrico com o circuito para medir a corrente elétrica alternada. Assim, não precisamos abrir o circuito para ligá-lo em série com a carga, basta apenas "abraçar" um dos fios ligados ao equipamento a ser monitorado. A Figura 2.9 ilustra o sensor.

Figura 2.9: Sensor SCT-013. Fonte (FLIPEPLOP, 2019b)

Com base nesse princípio de funcionamento, foram criados diferentes modelos de sensores não-invasivos,com o objetivo de atenderem os mais diversos cenários, como pode ser observado em seu *datasheet* (Tabela 2.3).

Cada modelo é caracterizado pela corrente máxima a ser medida (*Input current*) – e o tipo de saída do sensor (*Output type*).

É possível observar que somente o modelo SCT-013-000 apresenta uma variação de corrente em sua saída de 0-50mA, já os outros modelos apresentam uma variação em tensão de 0-1V, onde por meio destas variações é possível mensurar a corrente elétrica.

Model	**SCT-013-000**	***SCT-013-005***	***SCT-013-010***	***SCT-013-015***	**SCT-013-020**
Input current	0-100A	0-5A	0-10A	0-15A	0-20A
Output type	0-50mA	0-1V	0-1V	0-1V	0-1V
Model	**SCT-013-025**	**SCT-013-030**	**SCT-013-050**	**SCT-013-060**	**SCT-013-000V**
Input current	0-25A	0-30A	0-50A	0-60A	0-100A
Output type	0-1V	0-1V	0-1V	0-1V	0-1V

Tabela 2.3: *Datasheet* do sensor SCT-013. Fonte (YHDC, 2019)

Para sabermos qual será a taxa de variação tanto de corrente quanto de tensão basta dividirmos o valor máximo da saída pelo valor máximo a ser medido. Abaixo encontram-se exemplos dos cálculo para o modelo com saída em corrente e para os modelos com saída em tensão

SCT-013-000:

$$0,05/100 = 0,5mA$$

A cada um Ampere a mais ou a menos, sua saída será de 0,5mA para mais ou a menos;

SCT-013-020:

$$1/20 = 0,05V$$

A cada um Ampere a mais ou a menos, sua saída será de 0,05V para mais ou a menos;

É importante resaltar que para microcontroladores compatíveis com a Arduino IDE existe a biblioteca EmonLib que realiza a leitura do sensor. Porém se faz necessário um circuito eletrônico para a utilização deste sensor, uma vez que o mesmo realiza leitura tanto positivas como negativa, devido a corrente alternada, este circuito converte os valores para sempre serem maiores do que zero, tornando assim possível a leitura nos microcontroladores.

2.2.5 ZMPT101B

O Sensor de Tensão alternada ZMPT101B é um módulo de alta precisão que tem como finalidade detectar a existência de tensão alternada em um circuito ou fazer a medição do valor de tensão. Este pode ser observado na Figura 2.10.

Para o desenvolvimento de projetos de automação residencial o Sensor de Tensão AC ZMPT101B é de extrema importância, pois este é capaz de informar se uma lâmpada está acesa

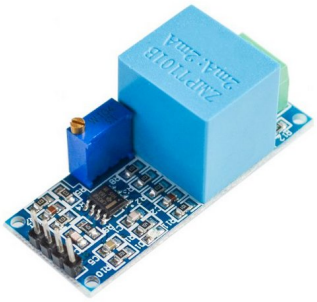

Figura 2.10: Módulo sensor de tensão ZMPT101B. Fonte (ROBóTICA, 2019)

ou apagada, se um motor está ligado ou desligado independente se o circuito está sendo controlado por uma aplicação/página web ou por interruptor/botão. Além disso, este sensor permite a implementação de um projeto que tenha como finalidade monitorar os valores de tensão na rede alternada, já que possui uma boa faixa de operação conseguindo medir tensões de 0 a 250V, e possuindo uma precisão de $\pm 0,5$V.

2.3 Plataformas *cloud* IoT

Para aplicações IoT que exijam monitoramento remoto as plataformas *cloud* são, atualmente, a melhor forma de armazenar, visualizar, processar e analisar dados. Existem diversas plataformas disponíveis com versões gratuitas, porém limitadas em relação a versão paga. Dentre as mais utilizadas encontram-se Thingspeak e Ubidots.

O Thingspeak é uma plataforma que permite coletar e armazenar dados de sensores na nuvem. Ele fornece um console para desenvolvimento de módulos em Matlab para analisar e visualizar dados via interface Web. Para enviar os dados dos sensores pode-se utilizar o Arduino, ESP8266, Raspberry Pi, Beaglebone, entre outros microcontroladores. O envio é realizado via protocolo HTTPS, e a autenticação e segurança utilizando tokens de acesso, gerados via interface Web de administração. O Thingspeak utiliza o conceito de canais. Cada canal pode conter até 8 campos e seus valores podem ser visualizados utilizando uma interface Web. Um exemplo de um canal Thinkspeak pode ser visualizado na Figura 2.11.

A versão gratuita do serviço oferece ao usuário um número limitado de 3 milhões de mensagens por ano (aproximadamente 8.200 por dia), e sua taxa de envio se limita a um envio a

cada 15 segundos. Por outro lado, a versão padrão (paga) oferece ao usuário 33 milhões de mensagens ao ano, aproximandamente 90.000 mensagens por dia, para cada unidade, e permite um envio a cada segundo.

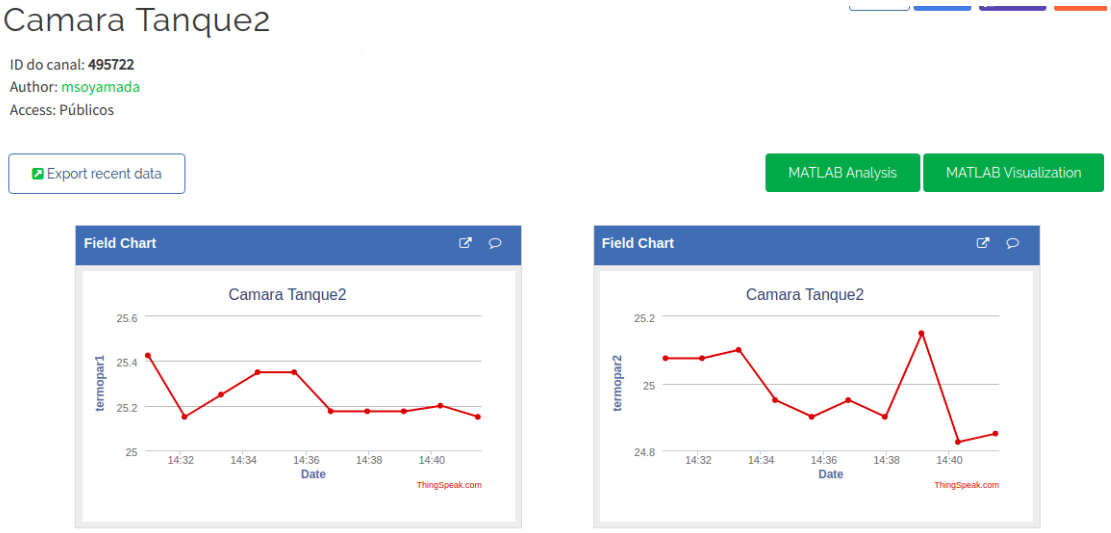

Figura 2.11: Exemplo de tela da aplicação ThingSpeak.

Ubidots é uma plataforma de IoT projetada para capacitar a criação de protótipos e projetos de IoT escaláveis. O ambiente possui uma plataforma amigável e personalizável que fornece aos usuários dados em tempo real e visualização de entradas de sensores na nuvem. A plataforma suporta a maioria das opções de hardware, protocolos de conectividade e um back-end de séries de tempo. Um exemplo da plataforma pode ser visualizado na Figura 2.12.

Na versão gratuita a plataforma proporciona a utilização de 1 até 20 dispositivos, e apresenta uma taxa de envio melhor se comparada a Thingspeak, possibilitando um envio por segundo. Já a versão paga oferece a utilização de até 100.000 dispositivos e possibilita 4 envios por segundo.

Figura 2.12: Exemplo de *dashboard* na aplicação Ubidots. Fonte (PELAEZ, 2018)

As duas plataformas mencionadas compartilham de muitas características, embora possuam algumas diferenças significativas, como apresentado na Tabela 2.3. Devido a estas disparidades, pode-se atribuir a plataforma Thingspeak aqueles usuários com necessidade mais básicas, enquanto que a Ubidots oferece uma plataforma mais robusta e com suporte a expansão.

Aspectos	Thingspeak	Ubidots
Atraso	Dependente a velocidade da rede, permite upload de dados simultâneos, portanto não possui atraso significativo .	Dependente a velocidade da rede, atraso significantes são esperados já que cada sensor possui seu próprio ID que é codificado separadamente.
Complexidade	Permite envio de dados de multiplos sensores em uma unica comunicação.	Necessário multiplos envios, a cada envio é necessário o ID do sensor e o dado do mesmo.
Características	• Biblioteca que permite o upload simultâneo de dados dos sensores relacionados e, portanto, atraso menor. • Ferramentas analíticas estão disponíveis para executar a análise em maior profundidade. • Os dados podem ser exportados para um arquivo .CSV para representação de planilhas.	• Os dados podem ser exportados para um arquivo CSV para representação de planilhas. • As ferramentas analíticas estão disponíveis com várias opções de representação gráfica. • Compatível com várias linguagens de programação
Custo do serviço	Serviço gratuito, com características adicionais pagas	Avaliação gratuita de 30 dias

Tabela 2.4: Comparação entre Thingspeak e Ubidots. Adaptado de (KALAITHASAN; RADZI; ABIDIN, 2018)

2.4 Técnicas de *Machine Learning*

Para realizar a análise dos dados captados do data center muitas vezes acaba-se por utilizar algoritmos de *machine learning*, na tentativa de prever futuros cenários e agir antes que estes ocorram, dentre as diversas técnicas existentes quatro já foram utilizadas em trabalhos relacionados para predição em data centers, estas são regressão linear, *random forest* e multi-layer

perceptron, todas descritas nas seções subsequentes.

2.4.1 Regressão Linear

Regressão linear (RL) é uma ferramenta comum na estatística para estimar o relacionamento entre duas variáveis, ou seja, o quanto uma influência a outra. A regressão linear simples trabalha com apenas 2 variáveis, estimando uma a partir do valor da outra. O modelo é definido assim pela equação $y = \alpha * x + \beta$ onde o resultado consiste de uma reta. Quando trabalhamos com mais de duas variáveis, também conhecido como regressão linear múltipla, o resultado deixa de ser representado por uma reta, passando a ser caracterizado por um hiperplano (SHALEV-SHWARTZ; BEN-DAVID, 2014).

Porém o modelo de regressão linear força a previsão a ser uma combinação linear de características, que é tanto sua maior força quanto sua maior limitação. Linearidade leva a modelos interpretáveis. Efeitos lineares são fáceis de quantificar e descrever. Eles são aditivos, por isso é fácil separar os efeitos (MOLNAR, 2018).

Outro problema do modelo de regressão linear é que este é sensível a *outliers*. Estes podem ser univariados (com base em uma variável) ou multivariados. Sendo assim o modelo irá se ajustar a esses *outliers* ocasionando um erro grande para os casos comuns (FLOM, 2018).

A regressão linear assume que cada tupla de dados é independente temporalmente de qualquer outra tupla. Porém se forem realizadas medições repetidas, como múltiplas amostras de temperatura e umidade, os dados não são independentes (MOLNAR, 2018).

2.4.2 *Random Forest*

Uma *Random Forest* (RF) consiste em um número arbitrário de árvores simples, que são usadas para determinar o resultado final. Dois tipos de aleatoriedade são construídos nas árvores. Primeiro, cada árvore é construída sobre uma amostra aleatória dos dados originais. Em segundo lugar, em cada nó da árvore, um subconjunto de caracteristicas é selecionado aleatoriamente para gerar a melhor divisão. Para problemas de classificação, o conjunto de árvores simples vota na classe mais popular. No problema de regressão, suas respostas são calculadas para obter uma estimativa da variável dependente. O uso de conjuntos de árvores pode levar a melhorias significativas na precisão da previsão (ou seja, melhor capacidade de prever novos

casos de dados) (TAN; STEINBACH; KUMAR, 2018).

Uma representação gráfica deste modelo pode ser vizualizado nas Figuras 2.13 e 2.14. Neste modelo, utilizado exclusivamente para ilustrar o funcionamento de uma *random forest*, se apresentam duas classes divididas entre classe **c1** e classe **c2**.

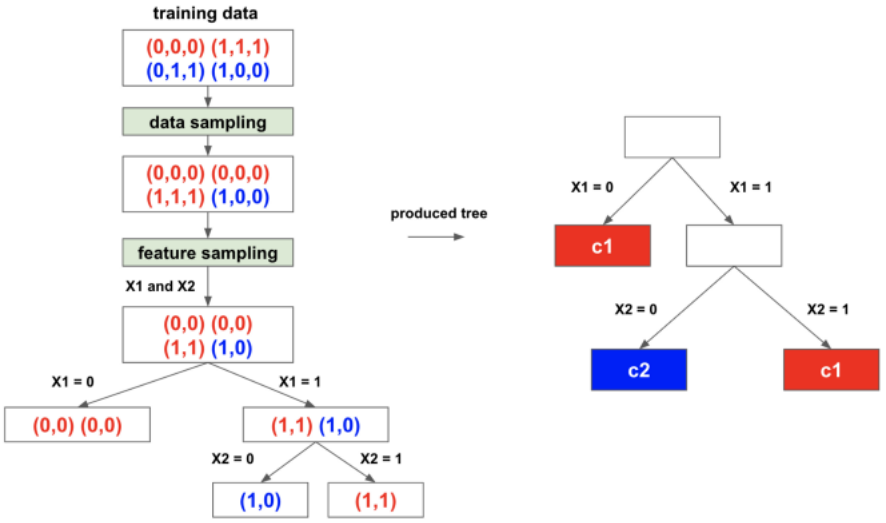

Figura 2.13: Formação das árvores em uma *random forest*. Adaptado de (DENG, 2018)

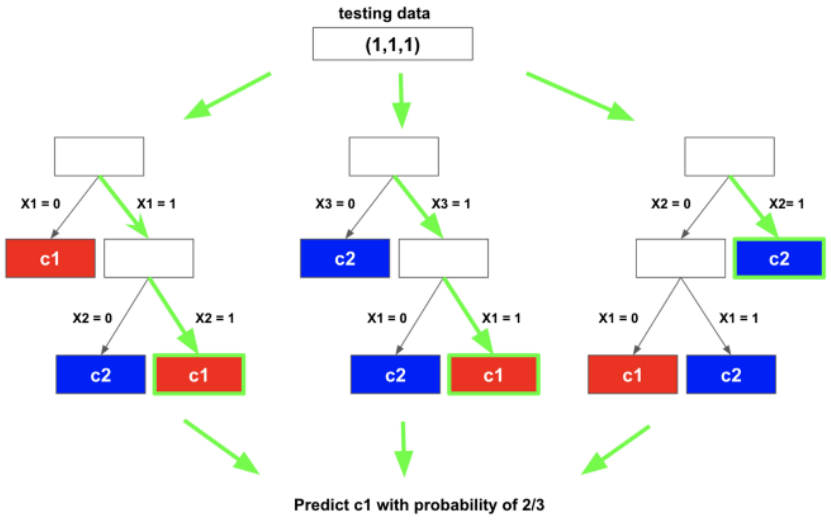

Figura 2.14: Exemplo de como é realizada a decisão da *random forest*. Adaptado de (DENG, 2018)

Na Figura 2.14 pode-se observar como a *random forest* decide a classe qual pertence a instância de entrada. Neste exemplo cada uma das árvores foi construída de forma diferente e a combinação da resposta das mesmas resulta na classificação final, onde duas árvores chegaram

a classe **c1** como resposta final, enquanto que uma árvore concluiu que a resposta seria a classe **c2**. Desta forma combina-se estas respostas obtendo uma probabilidade de 2/3 de ser a classe **c1**.

Alguns dos benefícios da *random forest* são que o seu desempenho preditivo pode competir com os melhores algoritmos de aprendizado supervisionados. Para muitos conjuntos de dados, ela produz um classificador altamente preciso. Uma desvantagem é a possibilidade do modelo apresentar *overfitting* quando existem dados com ruídos. Existem muitos parâmetros configuráveis na implementação/utilização da *random forest*, porém aqueles mais comumente utilizados são o número de árvores criadas, a função para medir a qualidade de uma divisão, geralmente gini ou entropia, e a profundidade máxima das árvores.

2.4.3 Rede Neural

Uma rede neural é um conjunto interconectado de elementos, unidades ou nós de processamento simples, a funcionalidade é vagamente baseada no neurônio. A capacidade de processamento da rede é armazenada nos pontos fortes ou pesos da conexão de unidade, obtidos por um processo de adaptação ou aprendizado de um conjunto de padrões de treinamento (GURNEY, 1997).

As redes neurais são frequentemente usadas para análise estatística e modelagem de dados, na qual seu papel é percebido como uma alternativa às técnicas de regressão não-linear ou análise de cluster. Assim, elas são normalmente usadas em problemas que podem ser expressos em termos de classificação ou previsão. Alguns exemplos incluem reconhecimento de imagem e fala, reconhecimento de caracteres textuais e domínios de conhecimento humano, como diagnóstico médico, pesquisa geológica de petróleo e previsão de indicadores do mercado financeiro (GURNEY, 1997). Um exemplo de um neurônio artificial pode ser vizualizado na Figura 2.15.

Na Figura 2.15 temos o caractere **x** representando as variáveis de entrada, o caractere **w** representando os pesos apicados a cada uma das variáveis de entrada, com os pesos aplicados tem-se o somador e a função de ativação e por fim gera-se uma saída **y**.

2.4.3.1 *Multilayer Perceptron*

O perceptron de camada única só pode classificar problemas linearmente separáveis. Para problemas não separáveis, é necessário usar mais camadas. Uma rede multicamada (*feed-*

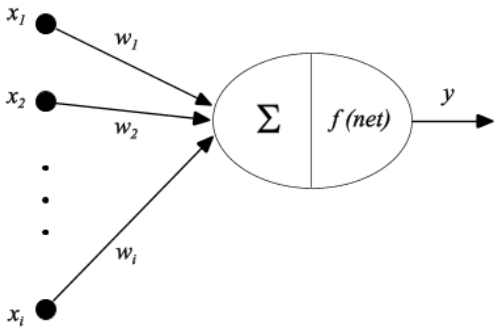

Figura 2.15: Exemplo de um neurônio artificial. Adaptado de (CERRI et al., 2011)

forward) tem uma ou mais camadas ocultas cujos neurônios são chamados de neurônios ocultos. A Figura 2.16 ilustra uma rede multicamadas com uma camada oculta. A rede é totalmente conectada, porque cada nó de uma camada é conectado a todos os nós da próxima camada. Se alguns dos links estiverem faltando, a rede está parcialmente conectada. Quando dizemos que uma rede tem n camadas, contamos apenas as camadas ocultas e a camada de saída; os nós de origem da camada de entrada não contam, porque os nós não executam cálculos. Uma rede de camada única, portanto, refere-se a uma rede com apenas uma camada de saída (JANTZEN, 1998).

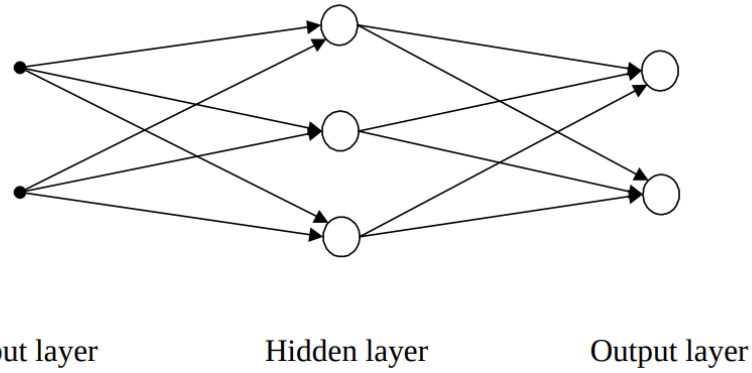

Figura 2.16: Exemplo de uma Multi-Layer Perceptron. Adaptado de (JANTZEN, 1998)

Para a *multilayer perceptron* os parâmetros mais comumente oscilados são número de camadas escondidas, bem como a quantidade de nós em cada uma dessas camadas, também é comum variar o número de iterações que serão realizadas para o treinamento e a função de ativação, os demais parâmetros são ajustados de acordo com a necessidade do modelo.

2.5 Soluções de Monitoramento de Temperatura em *Data Centers*

Em (SAHA; MAJUMDAR, 2017), os autores descrevem um sistema de monitoramento de temperatura em diferentes locais do *data center*, disponibilizando esses dados na internet utilizando dashboards em *cloud*. O trabalho se utiliza de sensores de temperatura DHT 11, uma placa ESP8266 e um roteador wireless como principais elementos de hardware. Como componentes de software utilizam a Arduino IDE e a Ubidots *cloud*. Cada *data racket* possui 2 sensores de temperatura, usados propositalmente para gerar redundância. O ESP8266 lê os dados dos sensores, que são comparados, e se qualquer divergência for encontrada, esse conjunto de dados será descartado e um próximo conjunto de dados será buscado. Caso os dados recebidos sejam de alguma forma similares, os envia para a plataforma *cloud*. Esta se encarrega de mostrar os dados em tempo real no dashboard e de realizar a notificação via SMS e email caso a temperatura suba acima das configurações de alarme.

2.6 Soluções de Controle

Entre as soluções encontradas para o problema estudado por este trabalho, existem aquelas que implementaram sistemas que controlam todo o ambiente de refrigeração de *data centers*. Entre esses sistemas existem aqueles baseados em regras e os baseados em aprendizado de máquina, ambos melhor explicados nas seções subsequentes.

2.6.1 Soluções baseadas em regras

O trabalho apresentado por (KOWSIGAN et al., 2017), busca um resfriamento automático do *data center*. Os componentes de *hardware* utilizados são um microcontrolador Arduino UNO, um módulo ESP8266 para conexão com a internet e sensores de temperatura TMP36. Adicionalmente são utilizados motores e acionadores para resfriar a sala.

O *data center* já possuia um sistema de refrigeração que não foi abordado com detalhes pelos autores, apesar de deixarem implícito a utilização de aparelhos de ar condicionado e exaustores em cima de cada *rack*, os mesmos citam as deficiências do sistema já implantado. Um dos problemas se dá pelo formato que os dados são analisados, de forma que se obtem uma tempera-

tura geral do *data center* e assim não conseguindo identificar aumentos de temperatura em *racks* isolados. Com o aquecimento dos *racks*, podem ocorrer perdas de dados, falhas de hardware, problemas de performance, entre outros.

Outro problema identificado pelos autores é que os dados coletados dos sensores eram armazenados em uma máquina física e a temperatura era controlada apenas pelo sistema, não existia nenhum *backup* de dados, e nenhuma análise era realizada a partir dos dados.

Na solução dos autores temos que os ventiladores, que são necessários para resfriar o *data center*, são mantidos ligados e se qualquer mudança na temperatura de um *data rack* ocorrer então este *data rack* em particular será resfriado a partir do acionamento do exaustor do mesmo, até retornar as condições normais. As decisões serão realizadas pelo servidor *cloud* ou manualmente.

Os componentes de software compreendem o aplicativo Arduino, que coleta a temperatura e envia para a plataforma em nuvem Thingspeak, para o armazenamento e análise dos dados. O funcionamento se dá pela leitura da temperatura da sala (sensores) pelo microcontrolador e utilizando o módulo ESP8266 envia os dados para a *cloud* Thingspeak.

Um módulo MATLAB executando na nuvem checa a temperatura do *data center* e, caso necessário, envia dados sobre a temperatura para o Arduino, que controlará a velocidade dos motores para a refrigeração do rack que está sobreaquecendo. Em suma, a maior parte do processamento e todo armazenamento de informações é realizado no *cloud server*. Dessa forma, o mesmo acaba sendo responsável por determinar as velocidades de rotação dos motores.

Neste trabalho os autores acabam por dar muito enfoque nas falhas do sistema que estava implantado no *data center* e omitem informações importantes sobre o sistema de refrigeração. Os autores também não mencionam em momento algum a utilização da análise de dados para reduzir o uso de energia, algo que poderia ser muito benéfico, mas seu foco se dá apenas pelo resfriamento de *racks* de maneira isolada para evitar seu sobreaquecimento.

(PURWANTO; UTAMI; PRAMONO, 2018a) e (PURWANTO; UTAMI; PRAMONO, 2018b) trazem uma abordagem baseada em lógica fuzzy com o intuito de monitorar temperatura e umidade, bem como atuar no controle de tais parâmetros. Para tal, utilizam de um microcontrolador Wemos D1, sensores DHT22 para temperatura e umidade, sensores ZMPT101B para tensão elétrica, um display lcd 16x2 e um módulo transmissor infravermelho. Como compo-

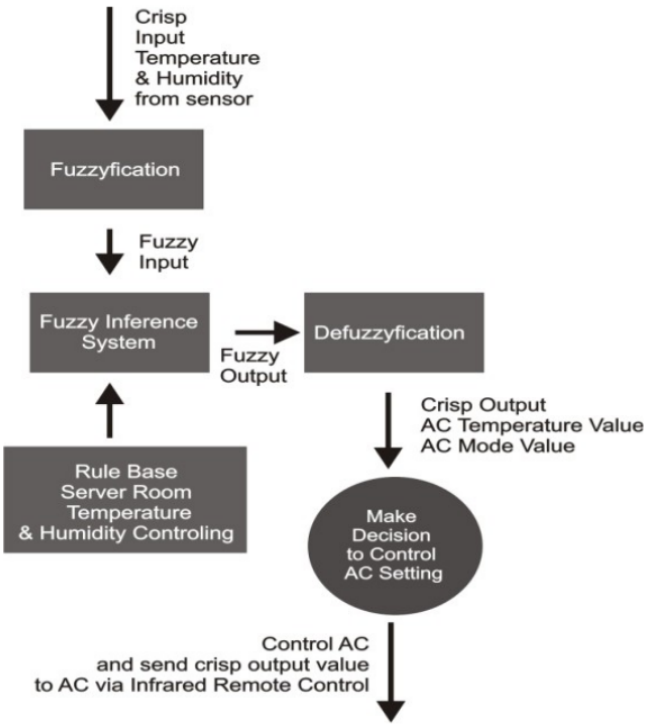

Figura 2.17: Sistema em lógica fuzzy proposto por (PURWANTO; UTAMI; PRAMONO, 2018a)

nentes de software utilizaram a Arduino IDE e Matlab.

O uso de lógica *fuzzy* se justifica pelo fato dela ser muito eficiente em controlar sistemas não lineares complexos e sistemas que são de difícil representação matemática, ou seja, se torna muito atrativa para o controle de temperatura e umidade, visto que as entradas dependem do clima da região e da utilização do *data center*.

Para fazer o controle da temperatura e umidade da sala, ocorre a leitura dos sensores e esta é usada como entrada para a fuzzificação. Em seguida, a saída da fuzificação é processada com o sistema de infêrencia das regras fuzzy. Por fim, a saída desse sistema é processada na defuzzificação e produz então uma saída para o controle das configurações de temperatura e modo do ar condicionado, como apresentado na Figura 2.17.

O processo de fuzzificação é traduzir a saída dos sensores em uma representação *fuzzy*, que fornece um valor de grau de associação entre 0 e 1. Os autores se utilizam do método Mamdani e da função de associação trapézio. Mamdani é a metodologia de inferência *fuzzy* mais empregada e consiste em quatro passos: (1) fuzzificação, (2) avaliação das regras *fuzzy*, (3) agregação das regras *fuzzy* e (4) defuzzificação. Na primeira fase se obtém o grau de pertinência com que

cada entrada pertence a cada conjunto fuzzy, neste caso se utilizando da função de associação trapézio. Para a avaliação das regras aplicam-se a entradas fuzzificadas nos antecedentes obtendo assim o valor do consequente para cada regra. Na terceira etapa são agregadas todas as funções membro dos consequente de cada regra em um único conjunto *fuzzy*. Por fim temos a defuzzificação que é utilizada para obter uma saida numérica, o método mais utilizado para tal é a técnica do centróide, que obtém o ponto onde uma linha vertical divide ao meio um conjunto agregado.

Neste sistema existem 2 variáveis de entrada, e cada uma delas possui 3 associações. Para a variável temperatura as associações são *COOL*, *NORMAL*, *HOT*, e para a variável umidade as associações são *DRY*, *NORMAL*, *WET*. Um total de 9 regras foram usadas para realizar decisões e produzir as saídas para a temperatura do ar condicionado em 3 graus de associação, que são, *LOW*, *MIDDLE*, *HIGH*, e para o modo do ar condicionado em 2 graus de associação, que são, *COOL* e *DRY*. Abaixo encontram-se 2 destas regras para exemplificação.

Algorithm 1 Funções de associação

if *Temperature COOL AND Humidity DRY* **then**
| Temperature AC HIGH AND AC Mode COOL
end
if *Temperature COOL AND Humidity WET* **then**
| Temperature AC HIGH AND AC Mode DRY
end

Para o monitoramento dos paramêtros foi desenvolvido uma aplicação usando PHP, Bootstrap, JQuery e MySQL. Essa aplicação pode armazenar e exibir em tempo real, a temperatura, umidade e tensão, dados esses gerados pelos sensores.

O objetivo deste trabalho era comparar o resultado do controlador lógico *fuzzy* usando Matlab com o controlador lógico *fuzzy* implementado no Wemos D1. Como conclusão os autores puderam observar que o controlador lógico implementado no microcontrolador obteve valores semelhantes aos resultados da implementação Matlab com um desvio médio de 0.035 para o conjunto de saída de temperatura do ar condicionado, e um desvio médio de 0.01225 para o conjunto de saída do modo do ar condicionado.

Neste primeiro trabalho os autores não apresentam dados de melhoria de desempenho no controle de temperatura e umidade da sala de servidores, mas apenas uma comparação entre

os controladores. Em um outro trabalho desenvolvido pelo mesmo grupo, é realizado uma otimização na associação do modo do ar condicionado, detalhado em (PURWANTO; UTAMI; PRAMONO, 2018b), com o intuito de separar bem os graus de associação *COOL* e *DRY*. Neste segundo trabalho, junto com essa otimização os autores relatam que a temperatura desejada no *data center* é obtida em 23 minutos, enquanto que a umidade desejada, que neste caso é de 55%, é obtida em 6 minutos, não relatam quantativamente os ganhos obtidos.

2.6.2 Soluções baseadas em aprendizado de máquina

(TARUTANI et al., 2015) aplicam uma técnica de aprendizado de máquina para prever a distribuição de temperatura em um *data center* de grande porte, o *data center* Keihanna (KDC). O mesmo consiste de duas linhas de *racks* de servidores, a linha A e a linha B, que estão organizadas em paralelo. Cada linha de *rack* consiste em seis racks, sendo A1 à A6 para a linha A e B1 à B6 para a linha B. A linha A do rack e a linha B do rack contêm 150 e 218 servidores, respectivamente. Além disso, o KDC dispõe de dois aparelhos de ar condicionado. Os condicionadores de ar inferior e superior são indicados por PAC-A e PAC-B na Figura 2.18, onde podemos analisar a estrutura do *data center*.

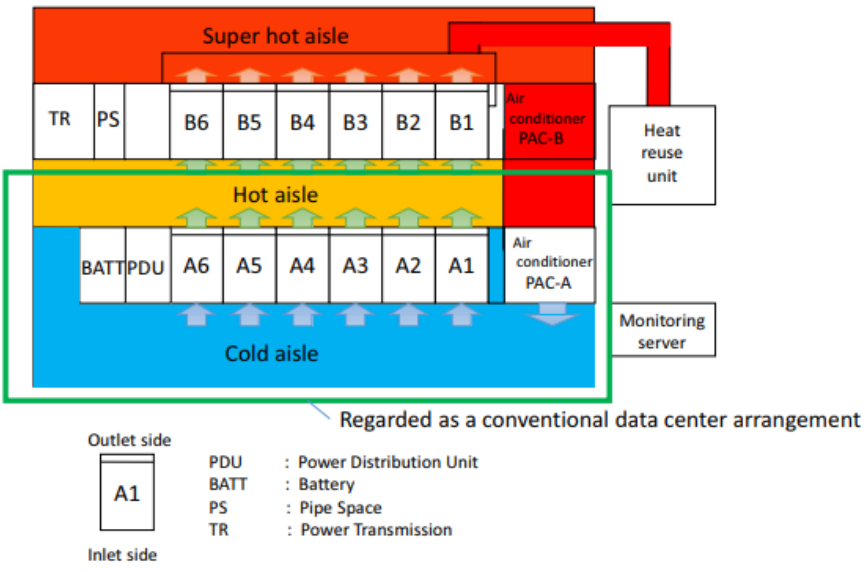

Figura 2.18: *Data center* Keihanna. Adaptado de (TARUTANI et al., 2015)

Sensores de temperatura foram instalados em vários locais no *data center* para monitorar a distribuição de temperatura. Os dados monitorados são enviados para o servidor de geren-

ciamento. Em seguida, o servidor define os parâmetros de controle, incluindo os parâmetros operacionais dos aparelhos de ar condicionado e a atribuição de tarefas nos servidores.

Como são necessários vários minutos para que as alterações nos parâmetros operacionais dos condicionadores de ar sejam refletidas na distribuição de temperatura em todo o data center, o controle proativo de acompanhamento dos condicionadores de ar é necessário para reduzir o consumo de energia.

Para prever a temperatura do *rack* A os autores se utilizam de 303 variáveis, sendo elas, temperatura de entrada de ar nos *servers* do *rack* A (150 variáveis), consumo de energia dos *servers* do *rack* A (150 variáveis), configuração da temperatura de saída do ar condicionado, configuração da velocidade (volume) do ar condicionado, e a mudança de temperatura em um passo de tempo.

Para prever a temperatura no *rack* B são utilizadas 677 variáveis, todas as utilizadas para a predição no *rack* A, e ainda, o consumo de energia dos *servers* do *rack* B (218 variáveis) e fornecimento de energia para os servidores da TR (156 variáveis).

O preditor de temperatura consiste em duas fases, chamadas fase de treinamento e fase de predição. Na fase de treinamento, os modelos de regressão (regressão linear e *random forests*) são construídos usando dados de treinamento. Na fase de predição, a distribuição de temperatura no tempo t + Δt (futuro), é prevista usando os modelos de regressão tendo como entrada o conjunto de dados atuais no tempo t.

No trabalho apresentado ambos os métodos de regressão apresentaram bons resultados, prevendo a temperatura distribuída com erros pequenos, de menos de 0,2 ^{o}C. Contudo, o método *randon forests* é superior ao método de regressão linear já que o desempenho de predição do primeiro se demonstra maior, enquanto o método de regressão linear apresenta uma pior precisão na predição.

Vale mencionar que neste trabalho os autores se utilizaram de uma máquina local para realizar a avaliação do tempo de cálculo para a fase de treinamento e previsão. Esta avaliação considera a média de dez execuções de cálculo para ambos os métodos implementados.

Em (TARUTANI et al., 2016) os autores estendem o trabalho anterior, adicionando o controle proativo dos aparelhos de ar condicionado. O servidor de gerenciamento define os parâmetros operacionais dos condicionadores de ar para maximizar a eficiência de energia dos mesmos

com base na distribuição de temperatura prevista. O parâmetro operacional **O** consiste na configuração do valor de saida da temperatura do ar condicionado $\mathrm{o'}_{temp}$ e no valor de saída do volume de ar $\mathrm{o'}_{vol}$ (velocidade do aparelho de ar condicionado). O servidor de gerenciamento determina os parâmetros operacionais do ar condicionado seguindo 4 etapas:

1. Observa os valores atuais do sensor de temperatura e os parâmetros operacionais e prevê a temperatura de cada sensor de temperatura na etapa de tempo t + Δt.

2. Calcula o valor provisório da temperatura de saída do ar condicionado.

3. Pesquisa o valor provisório do volume de ar do ar condicionado $\mathrm{o'}_{vol}$(t) para maximizar a eficiência energética do ar condicionado no caso em que a temperatura de saída é $\mathrm{o'}_{temp}$(t).

4. No caso em que a eficiência energética do ar condicionado com os valores provisórios for melhor que a eficiência atual ou a temperatura máxima prevista $\sum y_{max}$(t + Δt) for maior que o limite superior da temperatura y_{upper}, então os parâmetros operacionais são definidos aos valores provisórios. Caso contrário, os parâmetros operacionais dos condicionadores de ar não mudam.

O servidor de gerenciamento calcula o parâmetro operacional $\mathrm{o'}_{temp}$ com base na temperatura prevista na etapa 2 e pesquisa o parâmetro operacional para maximizar a eficiência de energia do ar condicionado com base na eficiência de predição de energia na etapa 3. Portanto, a distribuição de temperatura no data center é prevista usando os modelos de regressão, e a eficiência de energia dos condicionadores de ar é prevista usando os modelos de regressão vetorial de suporte.

Aplicando esta abordagem os autores apresentam uma redução de 17% do consumo médio de energia dos aparelhos de ar condicionado, podendo atingir até no máximo cerca de 30% do consumo de energia do ar condicionado. Esta redução é obtida por meio de controle proativo baseado no método de previsão de temperatura.

2.7 Considerações

Dentre os trabalhos similares encontrados não houve uma grande preocupação por parte dos autores em relatar os ganhos obtidos. Somente em 2 trabalhos fala-se de ganhos, no quesito de

redução do consumo de energia, e em apenas 1 deles há uma medida quantitativa.

Nos trabalhos (SAHA; MAJUMDAR, 2017) e (KOWSIGAN et al., 2017) os autores se mostram mais interessados em monitorar o *data center* e realizar algumas ações isoladas dentro do mesmos. Em ambos os trabalhos os autores mencionam os malefícios do não monitoramento dos servidores, mas se atém a isso não apresentando qualquer tipo de dado que demonstre uma melhora no *data center*.

(PURWANTO; UTAMI; PRAMONO, 2018a) apresentam uma proposta de controle baseado em lógica *fuzzy*, desta forma não apresentando nenhum ganho já que o sistema não havia sido implementado. Em (PURWANTO; UTAMI; PRAMONO, 2018b) os autores implementam o sistema que havia sido proposto e apresentam que a temperatura e umidade chegam aos padrões estipulados em menos de 30 minutos, mas não parece haver uma grande preocupação dos mesmos em reduzir o consumo de energia, mas sim de apenas realizar o controle do ar condicionado a partir da implementação de um sistema baseado em lógica fuzzy.

Em (TARUTANI et al., 2015) os autores demonstram uma preocupação com a predição das temperaturas dos sensores, para que a partir destas possa ser possível prever a temperatura do ar condicionado, como resultado apresentado temos uma análise entre dois algoritmos de regressão utilizados para a predição das temperaturas dos sensores. Já em (TARUTANI et al., 2016) os autores implementam um controle no ar condicionado utilizando a predição de temperatura dos sensores, apresentada por eles no trabalho anterior, e então apresentam ganhos significativos ao se utilizar um controle baseado em *machine learning* para o controle do ar condicionado, os mesmos obtiveram uma redução no consumo de energia de até 30%.

Embora a maioria dos trabalhos apresentados nesta seção se preocupem com a eficiência do aparelho de ar condicionado, buscando maneiras de manter a sala dos servidores resfriada, nenhum deles apresenta algum tipo de preocupação em relação ao consumo de energia do aparelho, desta forma este trabalho busca não só apresentar uma forma de se manter a sala dos servidores resfriada mas também se preocupa com o consumo de energia do aparelho buscando formas de minimizar o gasto energético enquanto maximiza o resfriamento da sala, de forma tal que possa manter os servidores operando em uma faixa de temperatura razoávelmente baixa e ainda assim trazer um gasto energético menor.

Capítulo 3

Arquitetura

Este capítulo tem por função descrever o *data center* onde o estudo será realizado, bem como descrever o sistema já existente no mesmo e, por fim, detalhar o sistema proposto por este trabalho, desde o monitoramento dos parâmetros do *data center* até a atuação no mesmo.

Atualmente o curso de Ciência da Computação da Universidade Estadual do Oeste do Paraná possui um *data center* de pequeno porte localizado na sala de coordenação de laboratórios. Este *data center* é composto por três máquinas e possui um aparelho de ar condicionado. Vale ressaltar que este ambiente onde o *data center* se encontra não é vedado, portanto existe uma forte influência externa no que diz respeito a temperatura e umidade da sala.

A sala de coordenação de laboratórios conta com uma janela e uma porta, estas como já dito anteriormente não são vedadas. Além disso, um fator que deve ser levado em consideração é a possibilidade de haver fluxo de pessoas na sala, algo que não pode ser ignorado, uma vez que isso pode alterar consideravelmente a temperatura dentro da sala. Uma representação da mesma encontra-se na Figura 3.1. A divisória encontrada na sala tem apenas 2 metros de altura e, portanto, não isola as máquinas do restante da sala.

O aparelho condicionador de ar presente na sala é da marca Midea 12000Btus com alimentação em 220V, possuindo uma corrente nominal de 5A, uma potência nominal de 1.234 W e uma vazão de ar de 550m^2/h.

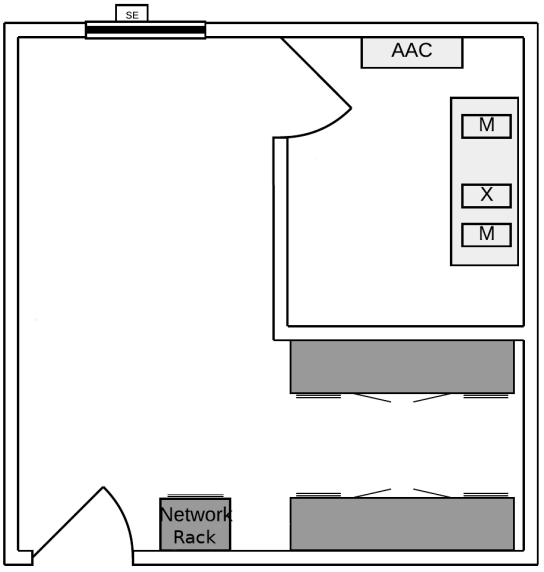

Figura 3.1: Representação da sala dos servidores

Na Figura 3.1, **AAC** representa o aparelho de ar condicionado, **M** representa as máquinas existentes no *data center*, **X** representa a máquina do *data center* onde se encontra o circuito que capta os dados e **SE** representa o sensor externo que capta temperatura e umidade fora da sala dos servidores.

O *data center* já possuia um sistema de monitoramento, implementado desde outubro de 2017, composto por um sensor de temperatura e umidade DHT22 e por um sensor de corrente SCT-013-000. O aparelho de ar condicionado é utilizado no modo automático em 22^{o}C e os dados coletados são enviados para a plataforma Thingspeak a uma taxa de 1 amostra por minuto. Os dados enviados são temperatura e umidade da sala dos servidores e corrente consumida pelo aparelho de ar condicionado.

3.1 Proposta de Sistema

O presente trabalho propõe uma alteração do sistema implementado atualmente no *data center*, com o intuito de melhorar os parâmetros observados e desta forma, ser possível, realizar uma melhor análise dos dados obtidos na tentativa de prever o consumo de corrente do condicionador de ar do *data center* nos próximos minutos. Com um modelo de predição será possível

alterar o funcionamento do modo do ar condicionado com o intuito de obter uma redução do consumo de energia e também obedecer as faixas de funcionamento adequadas de equipamentos eletrônicos. Pode-se ter uma visão geral do sistema na Figura 3.2.

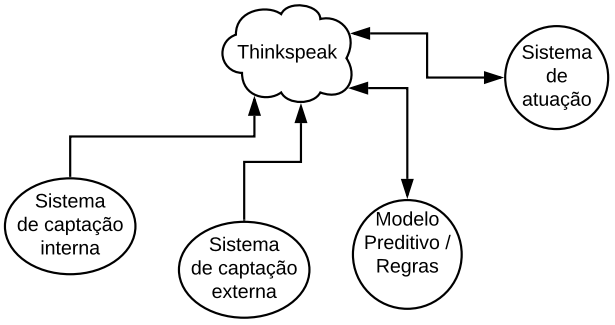

Figura 3.2: Visão geral do sistema implementado

Portanto o sistema funciona da seguinte forma, os sistemas de captação enviam dados para a plataforma online, neste caso Thingspeak, o modelo preditivo e/ou regras lê esses dados do canal, faz a predição e define a ação a ser realizada a partir de regras e salva-a em outro canal, e por fim o sistema de atuação atua de acordo com o que foi definido, e salva a temperatura que foi estipulada ao condicionador de ar para que no futuro essa informação seja utilizada para melhorar o preditor.

3.1.1 Monitoramento

No sistema proposto neste trabalho serão medidos os seguintes parâmetros: temperatura e umidade interna da sala dos servidores, temperatura e umidade externa da sala dos servidores e a corrente consumida pelo aparelho de ar condicionado.

3.1.1.1 Componentes de Harware

Para ser possível observarmos todos estes parâmentros alguns circuitos foram desenvolvidos. Para a captura da temperatura e da umidade externa a sala dos servidores é utilizado um microcontrolador NodeMCU, um sensor de temperatura DHT22, e uma fonte de 5V para a alimentação do circuito. Este pode ser visualizado na Figura 3.3.

Para a obtenção dos parâmetros internos foi desenvolvido um circuito contendo um sensor DHT22 conectado ao microcontrolador NodeMCU, um sensor SCT-013-000 conectado ao

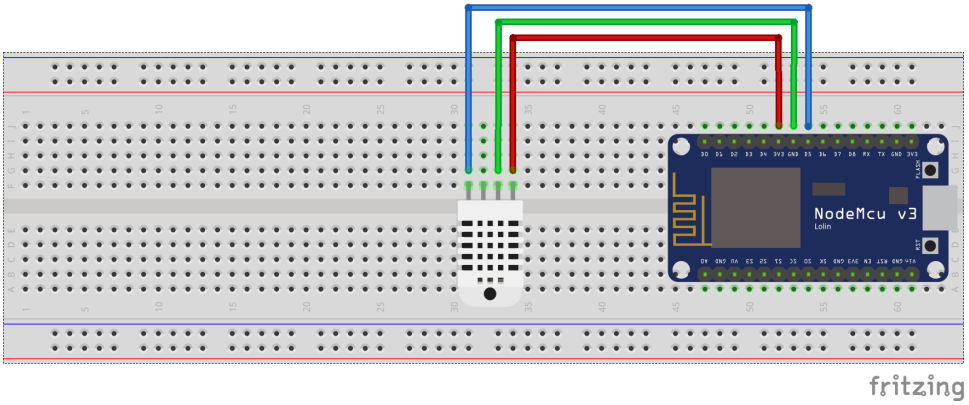

Figura 3.3: Circuito montado para a captação dos dados externos ao *data center*

MCU Arduino Nano e os dois microcontroladores conectados via comunicação serial. Aqui faz-se o uso de 2 microcontroladores devido a uma flutuação na leitura da entrada analógica do NodeMCU. Portanto, para corrigir essas leituras errôneas, optou-se por utilizar uma placa Arduino Nano e então conecta-la à NodeMCU. A alimentação do circuito é feita pela saída USB de uma das máquinas do *data center*. O circuito pode ser visualizado na Figura 3.4 e na Figura 3.5 tem-se a imagem do circuito real.

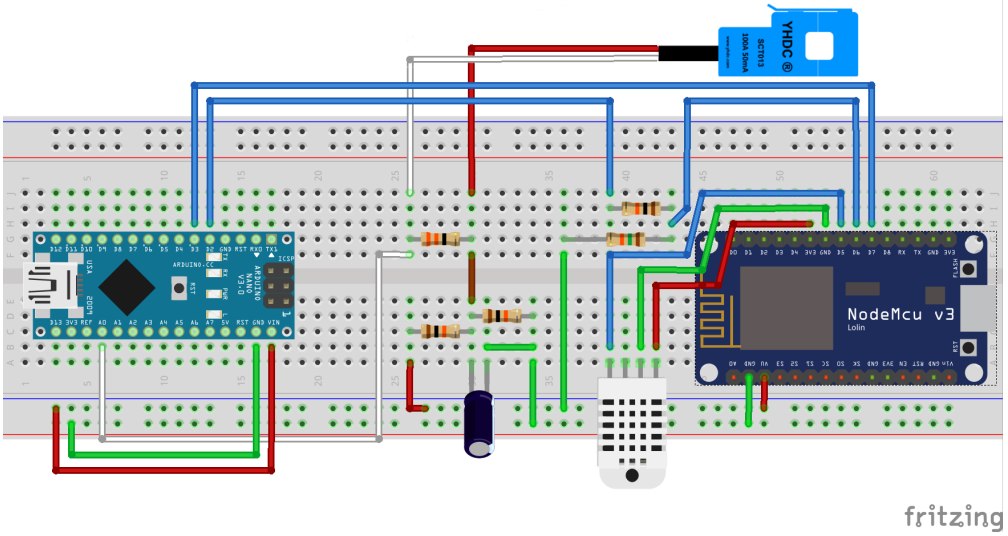

Figura 3.4: Circuito montado para a captação dos dados internos do *data center*

3.1.1.2 Componentes de Software

Para o armazenamento dos dados, bem como realizar notificações caso os mesmos não estejam sendo enviados utiliza-se a plataforma Thingspeak. Todos os parâmetros capturados são

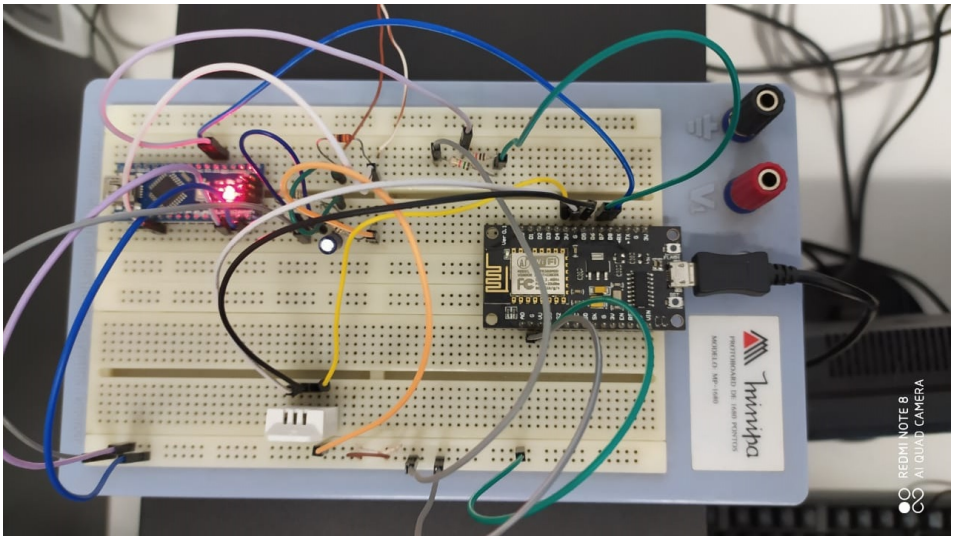

Figura 3.5: Circuito real construido para a captação dos dados internos do *data center*

enviados para a plataforma a cada 30 segundos.

Cada um dos canais configurados para receber os dados pode ser visualizados nas Figuras 3.6 e 3.7 sendo os dados captados dentro do *data center* e fora do *data center* respectivamente. Para facilitar a visualização, somente um dos parâmetros observados em cada canal é exibido.

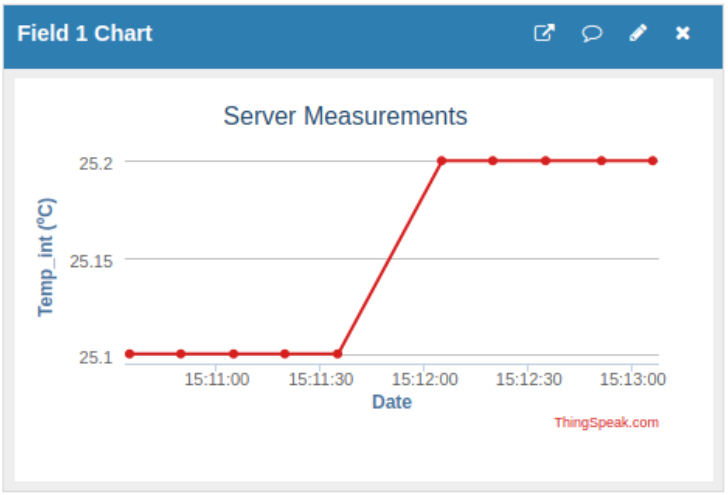

Figura 3.6: Canal Thingspeak com os dados internos do *data center*

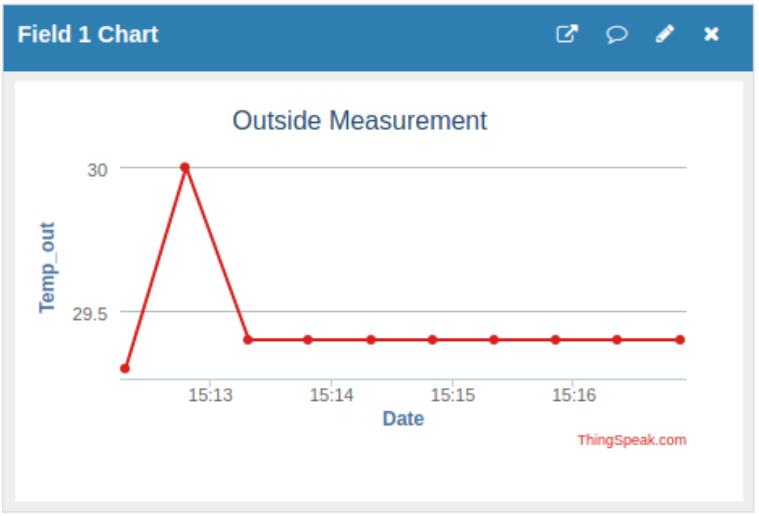

Figura 3.7: Canal Thingspeak com os dados externos ao *data center*

Para o melhor monitoramento do *data center* a cada 10 minutos um código de verificação é executado nos canais que recebem os dados e, se o mesmo identificar que algum deles não apresenta dados atualizados, dispara-se então um alerta para que possa ser possível corrigir o erro.

Com o sistema proposto visa-se poder atuar em casos onde o modo automático do ar condicionado não é eficiente, tais como restringir o funcionamento do ar condicionado em períodos noturnos, relaxando a faixa de temperatura em que o aparelho pode trabalhar, desta forma fazendo com que o mesmo seja ativado menos vezes e restringir o funcionamento do ar condicionado quando a temperatura externa está baixa o suficiente para o mesmo não seja necessário.

3.1.2 Análise dos dados capturados

Com os dados capturados pelo novo sistema de monitoramento, optou-se por realizar duas abordagens para o treinamento dos modelos de predição, a primeira sendo uma aplicação em *python* se utilizando da biblioteca sklearn e dos algoritmos de redes neurais e *random forest*. A segunda abordagem foi a utilização da plataforma AutoML do Google (GOOGLE, 2019), este é um conjunto de produtos de *machine learning* criado para que desenvolvedores treinem modelos de alta qualidade com base nas necessidades dos negócios.

Em ambas abordagens buscou-se por um preditor capaz de estimar o consumo de corrente real, diferentemente do que foi realizado no Apêndice A que visava apenas fazer a predição se o

aparelho seria acionado ou não em um determinado intervalo em minutos. No preditor proposto neste trabalho, a partir dos dados de entrada: temperatura interna e externa, umidade interna e externa e horário do dia, espera-se que o modelo possa estimar o consumo de corrente (em Amperes) acumulado nos próximos "Δ T"minutos.

3.1.2.1 Preparação dos dados

Antes de realizar o treinamento dos modelos os dados precisaram ser preparados. Isso decorre do fato de que, os sensores nem sempre operam corretamente e acabavam enviando valores nulos para o *Thingspeak*, bem como ocasionais quedas de energia ou de internet faziam com que um ou ambos os circuitos montados para a captação de dados ficassem fora do ar.

Para corrigir este problema foi implementado um algoritmo em *python* que realiza a leitura dos arquivos com os dados de captação interna e externa (uma vez que os dados estavam em canais diferentes geravam 2 arquivos csv, um para cada canal), e inicialmente elimina qualquer tupla onde um dos valores for nulo. Após essa primeira etapa, valida-se o horário da leitura dos dados, permitindo uma difereça de até um minuto entre as tuplas de dados interna e externa, e uma outra validação é realizada para garantir que existe uma instância válida no tempo t + Δ.

Se essas validações forem aceitas cria-se então uma nova tupla, esta que será salva no arquivo de instâncias para ser utilizada no treino e validação dos modelos. Tal tupla possui os campos: horário do arquivo de dados internos, temperatura interna, temperatura externa, umidade interna, umidade externa e o somatório da corrente do tempo t até t + Δ, como pode ser visualizada a Tabela 3.1.

Horário	**Temp. Ext.**	**Umidade Ext.**	**Temp. Int.**	**Umidade Int.**	**Somatório da Corrente**
2019-06-27 18:05:35 -03	17.00	95.00	20.00	68.95	0.38

Tabela 3.1: Exemplo de tupla de dados para treino e validação do modelo de predição

3.1.2.2 Aplicação *python*

Uma vez que um estudo preliminar já havia sido realizado (Apêndice A), possuíamos então a informação de quais algoritmos de predição funcionariam melhor para este tipo de dados, desta forma foram implementados os algoritmos de rede neural (MLP) e *random forest*, já que estes apresentaram melhores resultados no estudo prelimiar. Os dados foram organizados de

forma semelhante à do estudo, ou seja, a tupla era formada por horário do dia em segundos, temperatura interna, temperatura externa, umidade interna, umidade externa e por fim a coluna que deveria ser prevista, o somatório da corrente do tempo t até o tempo t + Δ, onde delta é um número inteiro que varia de 10 à 30 correspondente a janela de tempo em minutos da predição. Um exemplo da tupla pode ser visualizado na Tabela 3.2.

Horário em Segudos	**Temp. Ext.**	**Umidade Ext.**	**Temp. Int.**	**Umidade Int.**	**Somatório da Corrente**
73327	23.70	48.70	21.95	30.65	20.50

Tabela 3.2: Exemplo de tupla a ser utilizada na aplicação *python*

Com a tupla montada, se faz necessário a normalização dos dados, e para tal foi utilizada a função *MinMaxScaler* da biblioteca *preprocessing* do **scikit-learn**. Após realizado esse processo obtém-se uma tupla como a da Tabela 3.3, sendo utilizada como entrada para os modelos de predição.

Horário em Segudos	**Temp. Ext.**	**Umidade Ext.**	**Temp. Int.**	**Umidade Int.**	**Somatório da Corrente**
0.72	0.45	0.18	0.67	0.07	0.41

Tabela 3.3: Exemplo de tupla com os dados normalizados

Com o arquivo de instâncias pronto, realiza-se então a divisão do mesmo entre conjunto de treino, teste e validação, sendo 80% do conjunto instâncias para treino, 10% para teste e 10% para validação. Depois um total de 10 execuções foram realizadas, sempre alterando de forma randômica as instâncias de treino, teste e validação. No próximo capítulo, será apresentado em detalhes os resultados obtidos com esta aplicação.

3.1.2.3 Plataforma AutoML

Para não depender exclusivamente do modelo que foi gerado pela aplicação desenvolvida, decidiu-se por realizar também o treinamento de modelos de predição na plataforma AutoML do Google (GOOGLE, 2019). Diferentemente da aplicação *python*, aqui não e faz necessário o tratamento dos dados e nem a normalização dos mesmos, bem como não se fez necessário a divisão das instâncias em treino, teste e validação, a plataforma se encarrega de realizar todas essas operações.

Para treinar o modelo se fez necessário o *upload* do arquivo csv com todas as instâncias e então determinar o tamanho do conjunto de treino, teste e validação, que foram definidos em 80%-10%-10% respectivamente, bem como determinar a coluna que deve ser prevista. Após isso determina-se o tempo desejado de treinamento e então a plataforma se encarrega de encontrar e treinar o melhor modelo para os dados carregados. Ao final foi obtido um modelo de regressão que preve o consumo real de corrente.

3.1.3 Atuação

Para completar o novo sistema, foi desenvolvida a parte de atuação, responsável somente pela modificação da temperatura e modo do ar condicionado, deixando-o ligado ou desligado de acordo com a predição do consumo de corrente.

A decisão da temperatura é realizada por um sistema de regras, que será explicado em mais detalhes posteriormente. Para iniciar a atuação um sistema de regras estático foi inserido no ambiente do data center com o intuito de otimizar alguns cenários mais fáceis de serem analisados. E após a criação e treinamento do modelo preditivo este sistema de regras inicial foi substituido por outro, mais adequado para a utilização conjunta com o preditor.

Para que a atuação no aparelho de ar condicionado fosse possível foi construído um circuito que emite os comandos infravermelhos no padrão do aparelho de ar condicionado, como pode ser observado na Figura 3.8, composto por uma placa NodeMCU, alimentada por uma fonte de 5V, e um emissor infravermelho. Na Figura 3.9 pode-se observar o atuardor que foi construido.

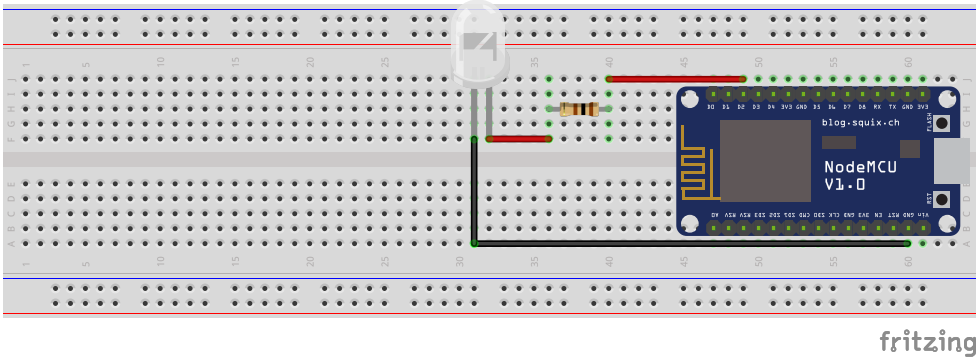

Figura 3.8: Circuito para a atuação no *data center*

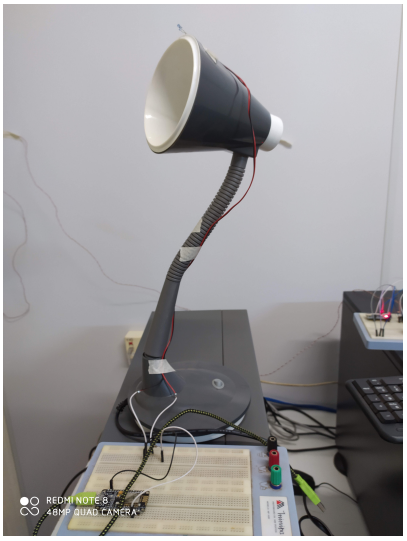

Figura 3.9: Atuador do *data center*

3.1.3.1 Sistemas de Regras

Como dito anteriormente dois sistemas de regras foram utilizados, o primeiro com regras mais gerais chamado de SR e o segundo atuando em conjunto com preditor de consumo de corrente, chamado de SRP. Para o primeiro as regras eram as descritas na Tabela 3.4, onde a coluna condição representa a regra e a coluna ação a temperatura definida para o aparelho de ar condicionado, se não adota-se nenhuma das regras para seguir definia-se então uma temperatura de 22ºC.

Numeração	Condição	Ação
1	Se sábado ou domingo	24ºC
2	Durante o dia && Temp. int. <22ºC && Temp. ext. <20ºC	24ºC
3	Durante o dia && Temp. int. >24ºC	22ºC
4	Madrugada && Temp. int. <24ºC	Desliga
5	Madrugada && Temp. int. >= 25ºC	24ºC
6	Default	22ºC

Tabela 3.4: Regras do sistema SR

As regras foram definidas de maneira empírica, ou seja, a partir da análise do consumo de

corrente do sistema de refrigeração do *data center* buscou-se por criar regras que pudessem atuar de forma a otimizar o uso do mesmo. Portanto tem-se que em finais de semana, como o uso é menos intenso, pode-se definir a temperatura para 24ºC, na intenção de fazer com que o aparelho de ar condicionado ative menos vezes.

Durante os demais dias da semana temos que, se a temperatura interna for menor do que 22ºC e a temperatura externa for inferior a 20ºC, isso indica que neste horário do dia a temperatura está amena, e desta forma pode-se subir a temperatura do aparelho de ar condicionado de forma a induzir um menor número de ativações do mesmo.

Se a temperatura interna for superior a 24ºC, define-se a temperatura do condicionador de ar em 22ºC, uma vez que a temperatura da sala começou a subir.

Durante a madrugada tem-se regras mais drásticas, uma vez que esse período tende a ser mais frio, se a temperatuta interna estiver inferior a 24ºC optou-se por desligar o aparelho de ar condicionanado e quando a temperatura atingisse a casa dos 25ºC, ou superior, definia-se então a temperatura do condicionador em 24ºC. E por fim se nenhuma das regras for atendida utiliza-se o valor padrão de 22ºC.

As regras para o segundo sistema apresentam-se na Tabela 3.5, onde a coluna condição representa a regra em si e a coluna ação representa a temperatura definida para o aparelho de ar condicioanado.

Numeração	**Condição**	**Ação**
1	Se sábado ou domingo	24ºC
2	Predição <5	Desliga
3	5 <Predição <35 && Temp. int. <23ºC	Desliga
4	5 <Predição <35 && Temp. int. >= 23ºC	22ºC
5	35 <Predição <90	22ºC
6	Predição >90	24ºC

Tabela 3.5: Regras do sistema SRP

Igualmente ao primeiro sistema de regras, as do sistema preditivo também foram definidas de forma empírica. Portanto, manteve-se a regra referente aos finais de semana, ou seja, como o uso do *data center* é menos intenso nestes dias, então pode-se definir a temperatura do aparelho de ar condicionado em 24ºC.

As demais regras baseiam-se no valor emitido pelo preditor, valores estes que podem variar de 0 à 110, e que indicam o quanto o aparelho de ar condicionado vai consumir de corrente nos próximos 30 minutos. Portanto ao decidir desligar o aparelho quando o preditor emitir um valor menor que 5, isso indica que o consumo de corrente durante esse período será ínfimo, possibilitando assim que o aparelho possa ser desligado.

Se o valor predito for entre 5 e 35, indica que o consumo será moderado-baixo e devido a isso se a temperatura interna for inferior a 23ºC pode-se desligar o aparelho de ar condicionando, uma vez que a temperatura interna durante os próximos 30 minutos (quando o sistema atuará novamente) não ultrapassará o limite estipulado pela ASHRAE. Caso a temperatura interna seja igual ou superior a 23ºC define-se a temperatura do condicionador para 22ºC.

Caso o valor da predição esteja entre 35 e 90 o consumo será moderado ou alto e assim sendo mantém-se a temperatura do aparelho de ar condicionado em 22ºC. Já se o valor da predição for superior a 90, isso indica, provavelmente, que nos próximos 30 minutos o aparelho será ativado muitas vezes, portanto para diminuir esse número de ativações define-se a temperatura do condicionador em 24ºC.

A atuação é efetivamente realizada em três etapas: predição, decisão baseada em regras e a atuação de fato. Para a primeira uma aplicação foi desenvolvida que lê os dados dos canais de monitoramento interno e externo no Thingspeak e aplica no modelo de predição. Como resposta recebe um valor real referente ao consumo de corrente estimado nos próximos Δ minutos. E a partir daí as regras da Tabela 3.5 são utilizadas para definir a temperatura do ar condicionado.

Depois de definida a temperatura do ar condicionado a mesma é salva em um outro canal Thingspeak para que possa ser lida pelo atuador. O sistema de atuação, que está embarcado na NodeMCU, lê o último valor salvo pelo software que define a temperatura e modifica o comportamento do aparelho conforme a leitura, depois disso entra em espera por 30 minutos, quando realizará uma nova leitura e configurará o aparelho de ar condicionando novamente. A temperatura que é de fato aplicada pelo sistema atuador é salva em um outro canal Thingspeak para usos em análises futuras.

Capítulo 4

Resultados Experimentais

Este capítulo apresenta os resultados obtidos durante o processo experimental, iniciando na etapa de treinamento dos modelos de *machine learning* para a predição do consumo de corrente elétrica no *data center*. Portanto, serão mostradas aqui os coeficientes de correlação[2] obtidos enquanto buscava-se um modelo mais acurado, bem como os motivos para a utilização do modelo selecionado.

E após esta fase, serão exibidas as comparações, referentes ao consumo energético entre as formas de atuação que foram realizadas, ou seja, entre o sistema inicial (sem atuação, com o aparelho de ar condicionado sempre em 22 graus Celsius no modo auomático), que será chamado de SI, o sistema de regras mais básico (SR) e entre o sistema com o modelo preditivo (SRP), sendo o principal objetivo a redução do consumo energético pelo aparelho de ar condicionado.

4.1 Modelo preditivo

Como descrito na seção anterior duas metodologias foram utilizadas, a primeira sendo o desenvolvimento de um modelo de predição utilizando a biblioteca **scikit learn**, enquanto que a segunda metodologia fez uso da plataforma online AutoML do Google.

Decidiu-se pela utilização destas duas metodologias para minimizar as chances de falhas na construção de um modelo preditivo eficiente, visto que em uma tentativa anterior de prever um número real de consumo de corrente, o resultado havia sido bastante insatisfatório, de forma que o modelo não obtinha um coeficiente de correlação superior à 0.5.

[2]R ao quadrado (R^2), também conhecido como coeficiente de determinação, é o quadrado do coeficiente de correlação de Pearson entre os rótulos e os valores previstos. Ele varia entre zero e um, em que um valor maior indica um modelo de qualidade maior.

Em relação a experiência prévia, os resultados obtidos nesta etapa foram mais precisos, atingindo um coeficiente de correlação superior a 0.75 em todas as execuções, para o melhor modelo construído utilizando a biblioteca **scikit learn**.

A Tabela 4.1 apresenta os valores de coeficiente de correlação para cada uma das execuções realizadas para cada modelo implementado, a cada execução as instâncias de treino, teste e validação eram alteradas, sendo definidas de forma aleatória. Vale ressaltar que essas execuções foram feitas em cima de dados que utilizavam Δ igual a 10, sendo que o modelo da MLP obteve um coeficiente de correlação médio de 0.44, enquanto que o modelo obtido a partir do algoritmo da *random forest* possui um coeficiente de correlação médio aproximado de 0.79, desta forma fica claro que o algoritmo de *random forest* é mais indicado para esta base de dados.

Na Tabela 4.1 **CD** representa o coeficiente de determinação, **NI** representa o número máximo de iterações, **TCE** representa o tamanho da camada escondida, **TA** representa a taxa de aprendizado, **NA** representa o número de árvores e por fim **PM** representa a profundidade máxima das árvores.

Iteração	**MLP**					***Random Forest***		
	CD	**NI**	**TCE**	**TA**	**Alpha**	**CD**	**NA**	**PM**
1	0.46	300	5	0.033	0.01	0.81	74	21
2	0.43	475	5	0.033	0.01	0.77	62	25
3	0.44	350	4	0.05	0.01	0.78	63	24
4	0.44	350	6	0.0125	0.0125	0.79	53	20
5	0.44	325	5	0.033	0.01	0.78	70	25
6	0.43	375	5	0.02	0.01	0.78	73	25
7	0.45	425	5	0.025	0.01	0.80	64	22
8	0.43	500	4	0.033	0.01	0.77	57	23
9	0.47	325	4	0.033	0.01	0.81	53	25
10	0.46	325	6	0.02	0.01	0.79	74	22
Média	0.44					0.79		
DV	0.015					0.013		

Tabela 4.1: Coeficiente de correlação de cada modelo para cada iteração

Contudo, este valor de 0.79, apesar de satisfatório, não é muito acurado, e mesmo o modelo apresentando uma taxa de acerto relativamente alta pode-se notar que os parâmetros que foram selecionados em cada iteração oscilam muito, tornando mais difícil identificar quais seriam os melhores para gerar um modelo definitivo.

Outro modelo desenvolvido foi utilizando a plataforma Google AutoML. Esta não apresenta detalhes do treinamento, como o valor do coeficiente de correlação a cada iteração, qual o algoritmo utilizado ou quais parâmetros foram utilizados no mesmo. Ainda assim algumas das informações que a plataforma disponibiliza são as seguintes: o tipo do modelo, que neste caso é de regressão e o valor final do coeficiente de correlação, que neste caso foi de 0.915, com a janela de predição Δ, igual a 10 minutos. Porém é possível se utilizar do modelo gerado de forma online através de requisições REST. No geral esta opção se demonstra mais interessante, uma vez que apresenta um coeficiente de determinação mais elevado em relação a aplicação local desenvolvida.

Como a diferença entre as precisões dos modelos foi em torno de 12.5%, decidiu-se por utilizar o modelo treinado pela plataforma AutoML. Uma vez selecionado o modelo podemos nos atentar as demais informações que a plataforma disponibiliza sobre o mesmo, tais como o erro médio absoluto, ou MAE[3], com um valor de 2.055, e ao RMSE[4], com um valor de 4.377. Esses valores nos indicam que o modelo é bastante preciso e possui um erro relativamente baixo, considerando o fato de que o mesmo está prevendo um número real.

A idéia inicial era se utilizar de 3 modelos, um treinado com dados de Δ igual a 10, outro com Δ igual a 20 e, por fim, um terceiro modelo treinado com Δ igual a 30, e então combinar a resposta dos 3 modelos para ser possível obter um resultado satisfatório para um periodo de tempo mais longo, visto que um modelo que preve o consumo elétrico em apenas 10 minutos no futuro não é muito eficaz em um ambiente de *data centers*, pois pode existir uma demanda durante um período de tempo mais longo que um modelo não conseguirá prever.

Porém, ao treinar o modelo com Δ igual a 30 na plataforma AutoML, obteve-se um resultado muito acima do esperado. Os valores podem ser visualizados na Tabela 4.2 e, embora o erro, tanto para o *MAE* quanto para o *RMSE*, sejam quantitativamente maiores em relação aos modelos com valores Δ menores, o erro percentual se mantém na mesma proporção, indicando assim que o modelo é tão acurado quanto os demais. Com isso optou-se por utilizar somente este preditor ao invés de três, como pensado inicialmente.

[3]O erro médio absoluto (MAE, na sigla em inglês): é a diferença absoluta média entre os rótulos e os valores previstos. Ele varia de zero a infinito, em que um valor menor indica um modelo de qualidade mais alto.

[4]O desvio da raiz do erro médio quadrado é uma medida muitas vezes usada das diferenças entre os valores previstos por um modelo ou um estimador e os valores observados. Ele varia de zero a infinito, em que um valor menor indica um modelo de qualidade mais alto.

		MAE		*RMSE*	
Modelo	R^2	Valor	Erro Percentual	Valor	Erro Percentual
Δ **10**	0.915	2.055	4.57%	4.377	9.73%
Δ **20**	0.936	3.413	5.25%	7.141	10.99%
Δ **30**	0.927	5.176	4.93%	10.498	10%

Tabela 4.2: Valores para os modelos de predição criados pela plataforam AutoML

4.2 Comparação de gasto energético

Os atuadores foram implantados em momentos diferentes e por períodos de tempo (quantidade de dias) diferentes, de forma que o SI operou de 27/06/2019 à 11/10/2019 totalizando 106 dias, enquanto que o SR operou de 11/10/2019 à 01/11/2019 totalizando 21 dias, enquanto que o SRP atuou de 01/11/2019 à 16/11/2019 totalizando 15 dias.

Finalmente buscou-se por realizar uma comparação entre o consumo de energia do aparelho de ar condicionado em cada uma das formas de atuação que foram utilizadas. Para tal se fez necessário identificar um período de tempo com características similares em cada modelo de atuação, de forma que a comparação fosse realizada sob as condições de temperatura e horário mais similares possíveis. Na Figura 4.1 pode-se observar a temperatura externa à sala dos servidores para o sistema inicial, portanto sem nenhhum tipo de atuação, apenas com o aparelho de ar condicionando em 22^{o}C, e para o sistema de regras, este que foi implantado antes do sistema de predição. Na Figura 4.1 o período de tempo representado é da 00:00h até as 08:00h, para o sistema inicial o dia é 2 de setembro de 2019 enquanto que para o sistema de regras o dia é 16 de outubro de 2019.

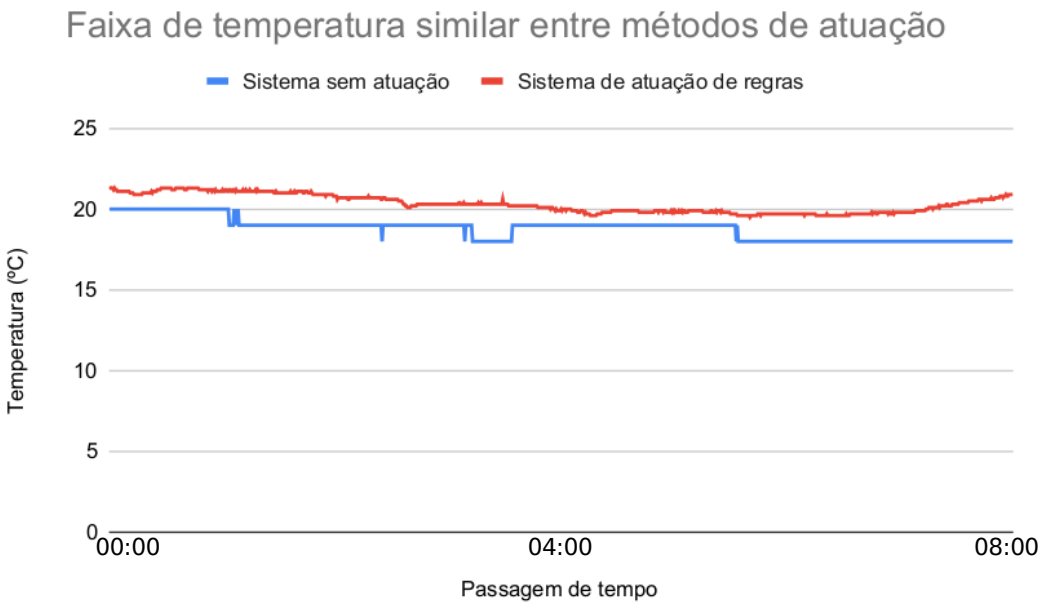

Figura 4.1: Faixa de temperatura entre o SI e o SR

Como é possível notar, as temperaturas se mantém bastante próximas, na faixa de 3ºC para mais ou para menos. Assim sendo, se torna bastante interesante analisar o consumo do ar condicionado neste período de tempo, já que as condições se mostram similares. Vale ressaltar que foram utilizadas várias faixas de tempo não contíguas para compor a análise, a fim de se obter uma média de consumo mais coerente com a realidade, e a faixa presente na Figura 4.1 é somente uma delas, utilizada para fins explicativos.

Ao todo, um total de 60 horas foram comparadas entre os dois sistemas e uma média de consumo em kWh foi gerada. O consumo do aparelho de ar condicionado configurado em 22ªC no modo automático foi de 44.71 kWh, enquanto que para o sistema de regras (SR) inicial o consumo foi de 36.94 kWh. Isso representa uma redução de aproximadamente 17.37% no consumo energético. Também é válido ressaltar que durante todo o tempo de atuação no sistema de regras as temperaturas estiveram mais elevadas em comparação ao tempo de atuação do sistema inicial, e ainda assim o sistema apresentou uma redução significativa no consumo de energia. Na Figura 4.2 pode-se notar que o sistema atuou corretamente conforme as regras estipuladas.

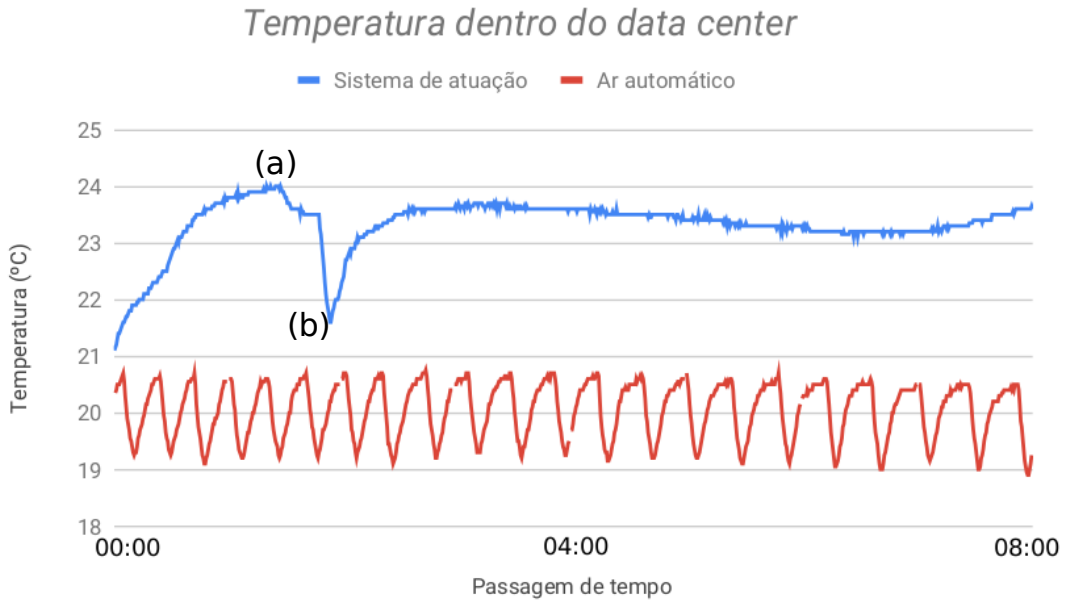

Figura 4.2: Temperatura dentro do *data center* durante o periodo de atuação do SR

Na Figura 4.2 pode-se observar que o sistema de atuação com o aparelho de ar condicionando no automático fica ativado durante toda a madrugada de forma que a temperatura fica subindo (quando o aparelho está com o compressor desligado) e descendo (quando o compressor está ligado[5]), já no sistema de atuação pode ser notado que o aparelho de ar condicionado foi desligado a partir da meia noite até o ponto indicado por **(a)** (regra 4), onde a temperatura é definida para 22°C (regra 6), e esta regra mantém-se ativa até o ponto **(b)**, onde volta a prevalecer a regra 4, esta que se mantém pelo resto do período analisado.

Para a análise de consumo elétrico do ar condicionado no sistema de atuação com o algoritmo preditivo utilizou-se do mesmo método empregado na análise anterior. Portanto, comparou-se o consumo de corrente em relação ao sistema inicial, ou seja, com o aparelho em 22°C no modo automático. Primeiramente buscou-se por períodos de tempo onde horário e as temperaturas externas fossem semelhantes para ambos os sistemas.

Uma vez encontradas essas faixas realiza-se então o cálculo de consumo energético. As faixas utilizadas na primeira análise acabaram não sendo utilizadas nesta etapa, devido a variabilidade meteorológica, que dificulta a reprodução exata das condições. A primeira diferença

[5]O aparelho de ar condicionado do *data center* é do tipo ON-OFF, de forma que o mesmo liga e desliga de forma a não sobrecarregar o compressor

desta análise para a anterior se encontra no número de horas utilizadas, neste caso houve um total de 87 horas onde as condições se mantiveram similares.

Nesta análise o consumo energético para o aparelho de ar condicionado em 22ºC no modo automático foi de 72.22 kWh, enquanto que o consumo energético para o sistema de predição foi de 44.89 kWh. Isso representa uma redução de aproximadamente 37.84% no consumo. Na Figura 4.3 pode ser observado o comportamento da temperatura interna durante o funcionamento do sistema de atuação preditivo no período de 00:00h de 09 de novembro de 2019 até as 08:00h.

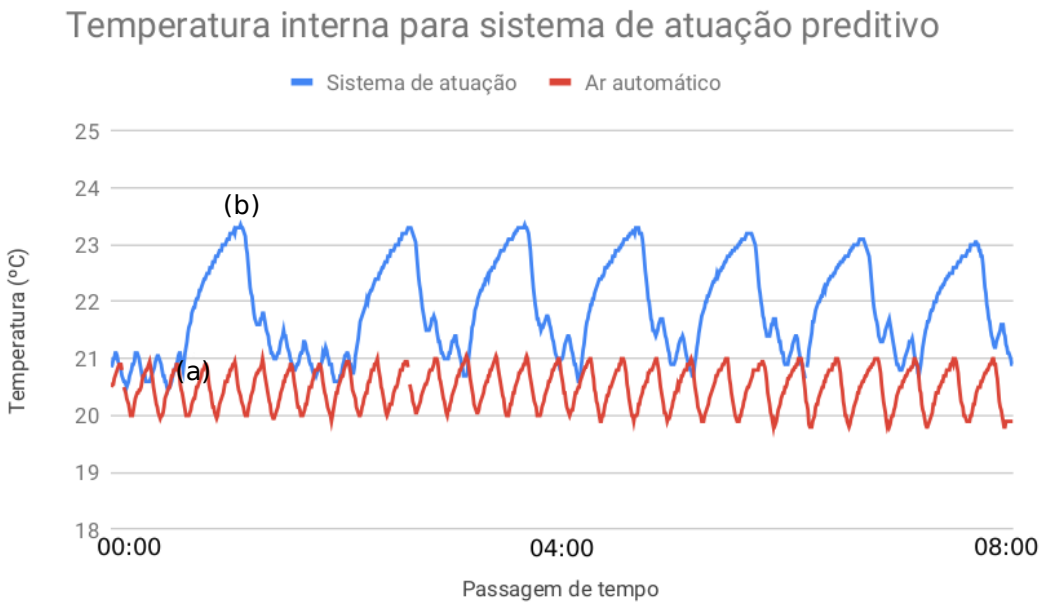

Figura 4.3: Temperatura dentro do *data center* durante o periodo de atuação do SRP

Na Figura 4.3, assim como na Figura 4.2 pode-se notar que o sistema de atuação com o aparelho de ar condicionando no automático fica ativado durante toda a madrugada, enquanto que o sistema de atuação preditivo atua de forma a utilizar as regras 3 (indicado por **(a)** na Figura) e 4 (indicado por **(b)** na Figura). A regra 3 desliga o ar condicionado se a estimativa determina que o consumo será muito baixo em um período de 30 minutos. No entanto, devido ao aquecimento da sala a temperatura interna sobe acima dos 23ºC, acionando novamente a regra 4, que comanda a temperatura de operação para 22ºC, repetindo essa oscilação de regras durante toda a madrugada. Portanto, o sistema tenta deixar o aparelho de ar condicionado desligado para reduzir o consumo, no entanto devido ao aquecimento ele atua ativamente para

evitar o aquecimento da sala dos servidores.

Por fim realiza-se uma comparação entre as duas formas de atuação baseada em regras. Neste caso um total de 37 horas foram utilizadas para a comparação. O consumo energético para o sistema de atuação sem o preditor foi de 12.56 kWh, enquanto que o consumo energético para o sistema de atuação com o preditor foi de 10.66 kWh. Representando assim uma redução de aproximadamente 15.08% no consumo, ao utilizar o sistema preditivo. Na Figura 4.4 pode ser observado a temperatura interna para ambos os sitemas de atuação. O período de tempo exibido é das 9:30h até as 12:30h.

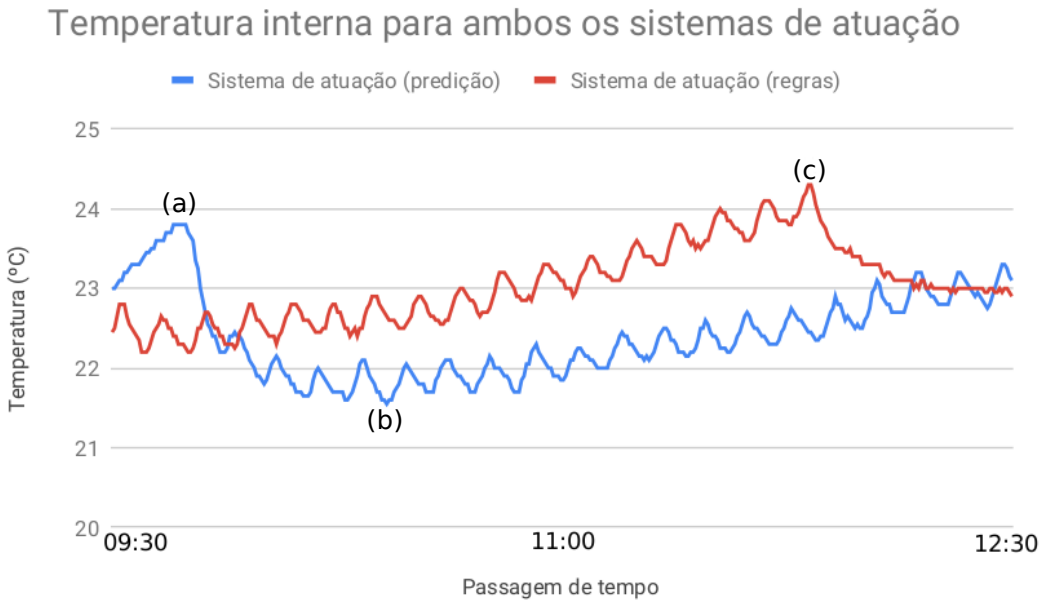

Figura 4.4: Temperatura dentro do *data center* durante o periodo de atuação dos sistemas de atuação (SR e SRP)

Na Figura 4.4 pode-se perceber que o sistema de atuação preditivo se utiliza das regras 3 (ativada em **(b)**) e 4 (ativada em **(a)**), enquanto que o sistema puramente de regras se utiliza da regra 2 (utilizada até o ponto **(c)**) e a regra 6 (acionada após o ponto **(c)**).

Todos os dados de consumo energético apresentados neste capítulo são valores estimados, uma vez que não é possível, neste caso, reproduzir o mesmo cenário para os diferentes modelos de atuação empregados, de modo que se fez necessário a utilização de faixas de temperatura similares para a realização dos cálculos. Os valores de consumo energético calculados apresentaram diferenças significantes, portanto, pode se assumir que de fato há uma economia

energética.

Capítulo 5

Conclusões

Neste trabalho realizou-se uma análise com objetivo de identificar uma forma de atuação em um aparelho de ar condicionado em um *data center* de pequeno porte, que reduzisse o consumo de energia elétrica. O sistema é baseado em um modelo preditivo capaz de prever o consumo de corrente em Δ minutos no futuro e, a partir desse modelo, atuar no sistema de refrigeração do *data center*.

Para realizar tal análise três sistemas de atuação foram utilizados, o do próprio aparelho de ar condicionado, mantendo a temperatura sempre em 22^{o}C, um sistema de regras simples e, por fim, um sistema com um algoritmo preditivo mais um sistema de regras. Para o algoritmo preditivo, presente no último sistema de atuação, realizou-se o treinamento de modelos em diversas técnicas a fim de encontrar uma predição mais acurada.

Para tanto, um estudo de técnicas de *machine learning* foi realizado com o intuito de identificar a técnica mais adequada a ser utilizada. Após identificadas as técnicas, estas foram implementadas e treinadas possibilitando avaliar o melhor modelo a ser utilizado. Por fim o modelo treinado pela plataforma AutoML do Google acabou apresentando uma melhor predição, assim sendo, este foi o utilizado durante a atuação.

Analisando os resultados alcançados, pôde-se perceber que, das três formas de atuação analisadas, aquela que utiliza do aparelho de ar condicionado sempre em 22^{o}C tem o consumo energético mais elevado em relação as outras duas estratégias. Apesar de não ser possível reproduzir exatamente o mesmo cenário nos testes, o fato de que a temperatura externa durante o uso de ambos os atuadores era mais alta que no modo automático, os resultados mostraram que os atuadores conseguiram reduzir o consumo energético. Outro ponto notado foi que o sistema que se utiliza de um preditor foi o mais eficiente apresentando uma economia de aproximada-

mente 15% maior em relação ao sistema que se utiliza apenas de regras.

Como trabalhos futuros, propõe-se a criação de um modelo preciso que possa ser utilizado localmente sem o uso da solução em nuvem AutoML, visto que a mesma gera custos para sua utilização. Adicionalmente, o sistema poderia ser melhorado com a criação de novas regras de forma dinâmica, ou possibilitando a exclusão de regras obsoletas ou ineficientes. Ou ainda possibilitar que o sistema de atuação aprenda regras conforme realiza a atuação no *data center*. Também propõe-se a realização de um estudo levando em consideração o fator umidade, que não foi considerado nesse trabalho.

Atualmente, o sistema proposto depende de conexão com a internet para funcionar. Uma proposta de trabalho futuro seria implementar uma abordagem tolerante a falhas, onde o sistema de regras mais simples atuaria em casos de desconexão da rede enquanto que o sistema preditivo atuaria quando houvesse conexão, evitando assim que o modo automático fosse empregado.

Apêndice A

Estudo Preliminar

Uma vez que já se possuía os dados que eram observados no antigo sistema de monitoramento, optamos por realizar uma análise nos mesmos para entender como o *data center* se comporta. Para a análise foram adicionadas a temperatura máxima e mínima observadas na cidade de Cascavel, disponibilizada pelo Simepar (PARANá, 2019).

A.1 Preparação do dados

Para realizar essa análise, fez-se necesário um preparo nos dados uma vez que os mesmos estavam totalmente crus. Alguns exemplos desses dados podem ser encontrados na Tabela A.1. Desta forma foi realizado o seguinte processo: Inicialmente modifica-se a representação de horário que se apresentava no formato HH:MM:SS (onde HH representa hora, MM representa minutos e SS representa segundos), para uma representação em segundos.

Horário	Temperatura Interna	Temperatura média Simepar	Corrente (A)
18:46:11	22.100000	19.975	0.127341
18:47:13	22.100000	19.975	0.128620
18:48:15	22.100000	19.975	0.128833
19:19:11	22.900000	19.325	3.233984
19:20:13	22.700001	19.325	4.150319

Tabela A.1: Representação dos dados crus

Em um segundo momento, foi necessário criar as tuplas de informação, onde cada tupla representa uma instância. Cada uma delas é composta os seguintes dados: horário do dia em segundos, o valor de temperatura medido no servidor no tempo t, o valor de temperatura média

da cidade de Cascavel-PR medido no tempo t, e por fim um somatório dos próximos 10 valores de consumo de corrente, do tempo t até o tempo t+9. Os dados após este processamento podem ser vizualizados na Tabela A.2.

Horário (s)	Temperatura interna	Temperatura média do Simepar	Somatório da Corrente (A)
69489	22.799999	19.325	22.698810
70107	22.100000	19.325	1.390598
70725	22.500000	19.325	1.412692
71344	22.799999	19.325	23.101477
71963	22.200001	19.325	1.337580

Tabela A.2: Representação dos dados parcialmente tratados

Com esse processamento realizado, fez-se então o seguinte tratamento: a coluna de somatório da corrente foi substituida por uma coluna de objetivos, onde cada elemento da coluna é representado somente por 0 (zero), para indicar que o compressor de ar não estava ligado durante os 10 períodos de tempo, ou 1 (um), que indica que o compressor de ar estava ligado durante esse período de tempo. Para determinar se o valor seria 0 ou 1 considerou-se que o aparelho de ar condicionado seria considerado como ligado quando seu consumo ultrapassasse 5A.

Esse limiar de 5A foi definido devido ao fato de que o aparelho de ar condicionado presente na sala dos servidores é do tipo ON-OFF, como pode ser visto na Figura A.1. Desta forma, o aparelho apresenta um consumo de corrente muito baixo, próximo de 0, quando o compressor de ar está desligado, e um consumo de corrente próximo de 5A quando o compressor está ligado. Em adição a isso, sempre quando ligado, o aparelho permanece com o consumo no pico por aproximadamente 6 minutos.

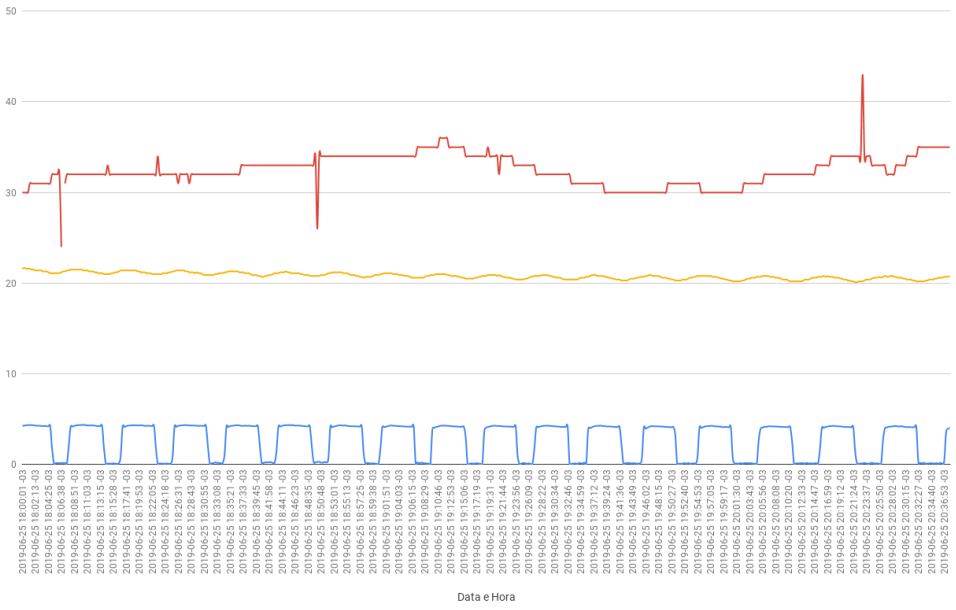

Figura A.1: Funcionamento do ar condicionado

Com todas as tuplas construídas fazia-se necessário a normalização dos dados, e para isso utiliza-se a função *MinMaxScaler* da biblioteca *preprocessing* do **scikit-learn**. Desta forma os dados finais ficam como mostrados na Tabela A.3.

Horário	Temperatura interna	Temperatura média do Simepar	Objetivo
0.82416194	0.3571429	0.60835913	1
0.90504166	0.44047624	0.51547988	1
0.11115746	0.58333328	0.17027864	0
0.33004253	0.3571429	0.44272446	0
0.78016477	0.50000006	0.92105263	1

Tabela A.3: Representação dos dados totalmente tratados

Com todos os dados preparados, separa-se então a tupla em parâmetros de entrada e parâmetro de saída (o que deve ser atingido), onde o parâmetro de saída é um valor booleano de 0 ou 1, representando se o compressor de ar ligou ou não no intervalo de 10 períodos de tempo, e os demais parâmetros fazem parte da entrada.

$$SUM_i(t+10) = pred(temp_{\text{int}}(t), temp_(smp)(t), horario(t))$$

Onde $\mathbf{temp_{int}}$ significa temperatura interna da sala dos servidores e $\mathbf{temp_{smp}}$ significa a temperatura média da cidade de Cascavel-PR medida pelo Simepar.

Após essa divisão, foi realizado um balanceamento nos dados, uma vez que possuiam muito mais instâncias onde o aparelho de ar condicionado aparecia desligado. Caso o balanceamento não fosse realizado os modelos poderiam sofrer de *overfiting*, para evitar isso limita-se o número de instâncias de cada classe para serem equivalentes a classe que contém o menor número de instâncias. Após o balanceamento outra divisão se fazia necessária, a divisão de conjuntos de treino e conjunto de validação. Para isso optou-se por utilizar uma taxa de 80/20, ou seja 80% das instâncias foram utilizadas para treino e 20% foram utilizadas para validar o modelo construído.

A.2 Algoritmos utilizados

Para obter um modelo de predição do comportamento do consumo de energia do aparelho de ar condicionado, um estudo com três algoritmos de *machine learning* foram utilizados. As técnicas utilizadas foram: MLP (*Multi Layer Perceptron*), Regressão Linear e por fim *Random Forest*. Todos os três algoritmos foram utilizados da biblioteca **scikit-learn**.

O forma utilizada para validar os modelos de predição gerados por cada um dos métodos empregados foi através do cálculo da acurácia. Desta forma sabe-se qual modelo apresenta a maior taxa de acerto ao prever se o aparelho de ar condicionado consumirá corrente no entre *t* e *t+10*. O cálculo é realizado como segue:

$$\frac{\sum_1^n (y\_real[i] = y\_pred[i])}{n}$$

Para todos os algoritmos foram realizadas 10 execuções, e em cada uma delas foram oscilados parâmetros dos algoritmos a fim de obter o melhor resultado em cada execução. Vale ressaltar que para cada iteração as instâcias utilizadas no treino e na validação são alteradas. Em cada uma das iterações realizadas existiam 1768 instâncias de treino e 444 instâncias de validação.

Para o preditor MLP oscilou-se os seguintes parâmetros, número de camadas escondidas, variando de 1 a 5 em passos de 1, número máximo de iterações, variando de 50 a 300 em passos de 25, taxa de aprendizado, variando de 0,01 a 1 em passos de 0,01 e usando o ativador *relu*.

Para o preditor *Random Forest* foram oscilados os seguintes parâmetros, o número de árvores na floresta, variando de 50 a 300 em passos de 25, a função para medir a qualidade da divisão esta podendo ser gini ou entropia e a profundidade máxima, variando de 1 a 16 em passos de 1.

Por fim para o preditor Regressão Linear o intercepto para modelo é o único parâmetro oscilado, sendo definido como verdadeiro ou falso, desta forma se definido como falso, nenhuma interceptação será usada nos cálculos.

Para cada iteração a melhor acurácia para cada classificador foi armazenada para realização de estudos posteriores. Na Tabela A.4 podemos observar a acurácia de cada classificador em cada iteração realizada.

Iteração	Regressão Linear	*Random Forest*	Melhores parâmetros (RF)	MLP	Melhores parâmetros (MLP)
1	0.790541	0.900901	300 - 15 - gini	0.786036	250 - 30 - constante
2	0.790541	0.889640	50 - 15 - gini	0.792793	200 - 6 - adaptativo
3	0.788288	0.900901	125 - 14 - entropia	0.763514	175 - 9 - invscaling
4	0.790541	0.882883	75 - 14 - gini	0.790541	150 - 27 - insvscaling
5	0.772523	0.889640	250 - 14 - entropia	0.768018	200 - 9 - constante
6	0.772523	0.876126	75 - 13 - entropia	0.772523	300 - 21 - invscaling
7	0.826577	0.918919	150 - 12 - gini	0.822072	300 - 12 - invscaling
8	0.783784	0.891892	125 - 10 - entropia	0.774775	275 - 9 - constante
9	0.815315	0.896396	50 - 10 - gini	0.840090	275 - 3 - invscaling
10	0.786036	0.912162	250 - 13 - entropia	0.779279	250 - 9 - constante
Média (DP)	0.791667 (0.016186)	0.895946 (0.012245)	-	0.788964 (0.023211)	-

Tabela A.4: Resultados do treinamento dos preditores

A tabela apresenta as acurácias obtidas por cada classificador em cada iteração e, para aqueles que os parâmetros foram oscilados, mostra-se os melhores para cada iteração. Para a *random forest* os valores representam o número de árvores, a profundidade máxima das árvores e critério de avaliação, respectivamente. Para a MLP os parâmetros exibidos são, número máximo de iterações, números de nós na camada escondida e a programação da taxa de aprendizado para atualizações dos pesos.

Analisando a Tabela A.4 pode-se notar que o algoritmo *random forest* foi bastante superior aos outros dois. Portanto, para garantir que os resultados estavam concisos, realizou-se outras 10 execuções para este algoritmo, desta vez utilizando os melhres parâmetros encontrados na excução anterior. Estes foram números de árvores definida em 125 e profundidade das árvores

em 14. A partir dessas execuções se obteve os resultados apresentados na Tabela A.5

Iteração	*Random Forest*
1	0.876126
2	0.887387
3	0.891892
4	0.914414
5	0.889640
6	0.925676
7	0.889640
8	0.896396
9	0.894144
10	0.927928
Média (DP)	0.899324 (0.016429)

Tabela A.5: Resultados do treinamento da *random forest*

Com estes resultados pode-se concluir que o algoritmo apresenta uma boa acurácia e mostra-se estável. Desta forma fica claro que o algoritmo mais indicado para este tipo de base de dados é a *random forest*. Por mais que este estudo tenha sido realizado a partir de uma base de dados diferente da base de dados que será utilizada no futuro, podemos levar estes resultados em consideração, visto que os dados que serão capturados no novo sistema de monitoramento (explicado nas seções seguintes), ainda terá muitos dos parâmetros aqui analisados.

Porém, o principal objetivo é construir um modelo capaz de estimar efetivamente o consumo de corrente, e não somente se o compressor de ar do aparelho de ar condicionado está ligado ou desligado.

Referências Bibliográficas

ARDUINO. *ARDUINO UNO REV3*. 2019. Disponível em: <https://store.arduino.cc/usa/arduino-uno-rev3>.

ASHRAE. *Thermal Guidelines for Data Processing Environments*. 1. ed. [S.l.]: Atlanta, GA, USA, 2004.

ASHRAE. *Thermal Guidelines for Data Processing Environments*. 2. ed. [S.l.]: Atlanta, GA, USA, 2008.

ASHRAE. *Thermal Guidelines for Data Processing Environments*. 3. ed. [S.l.]: Atlanta, GA, USA, 2012.

CERRI, G. et al. Automatic operator support system based on a neural network monitoring system. *Condition Monitoring and Diagnostics Engineering Management*, 2011.

CHEUNG, I. et al. Energy efficiency in small server rooms: Field surveys and findings. *2014 ACEEE Summer Study on Energy Efficiency in Buildings*, n. 12, 2014.

CIRCUITO, C. *Módulo WiFi ESP8266 - ESP-12E*. 2018. Disponível em: <https://www.curtocircuito.com.br/modulo-wifi-serial-esp-12e.html>.

DAYARATHNA, M.; WEN, Y.; FAN, R. Data center energy consumption modeling: A survey. *IEEE COMMUNICATIONS SURVEYS TUTORIALS, VOL. 18, NO. 1, FIRST QUARTER 2016*, v. 18, n. 63, p. 732–794, 2016.

DELMASTRO, C. *Cooling Tracking Clean Energy Progress*. 2019. Disponível em: <https://www.iea.org/tcep/buildings/cooling/>.

DENG, H. *An Introduction to Random Forest*. 2018. Disponível em: <https://towardsdatascience.com/random-forest-3a55c3aca46d>.

DEVICES, A. *Low Voltage Temperature Sensors TMP35/TMP36/TMP37*. [S.l.], 2010.

FILIPEFLOP. *Placa Nano V3.0 + Cabo USB para Arduino*. 2019. Disponível em: <https://www.filipeflop.com/produto/placa-nano-v3-0-cabo-usb-para-arduino/>.

FLIPEPLOP. *Placa WeMos D1 R1 Wifi ESP8266*. 2019. Disponível em: <https://www.filipeflop.com/produto/placa-wemos-d1-r1-wifi-esp8266/>.

FLIPEPLOP. *Sensor de Corrente Não Invasivo 100A SCT-013*. 2019. Disponível em: <https://www.filipeflop.com/produto/sensor-de-corrente-nao-invasivo-100a-sct-013/>.

FLOM, P. *The Disadvantages of Linear Regression*. 2018. Disponível em: <https://sciencing.com/disadvantages-linear-regression-8562780.html>.

GAJERA, U. *Basics of Interfacing DHT11/DHT22 Humidity and Temperature Sensor with MCU*. 2017. Disponível em: <http://www.ocfreaks.com/basics-interfacing-dht11-dht22-humidity-temperature-sensor-mcu/>.

GOOGLE. *Cloud AutoML*. 2019. Disponível em: <https://cloud.google.com/automl/>.

GURNEY, K. *An introduction to neural networks*. 1. ed. [S.l.]: UCL Press, 1997.

JANTZEN, J. *Introduction To Perceptron Networks*. [S.l.], 10 1998.

KALAITHASAN, K.; RADZI, N. A. M.; ABIDIN, H. Z. Internet of things application in monitoring sick building syndrome. *Indonesian Journal of Electrical Engineering and Computer Science*, v. 12, n. 2, p. 505–512, 2018. ISSN 2502-4752.

KOWSIGAN, M. et al. An enhaced automatic cooling system in cloud data center using iot. *2017 International Conference on Intelligent Computing and Control (I2C2)*, n. 5, 2017.

MOLNAR, C. *Interpretable Machine Learning*. 1. ed. [S.l.]: Leanpub, 2018.

NODEMCU. *NodeMcu Connect Things EASY*. 2019. Disponível em: <http://www.nodemcu.com/index_en.html>.

PARANá, S. M. do. *Simepar*. 2019. Disponível em: <http://simepar.br/>.

PELAEZ, A. *IoT Dashboards – Attributes, Advantages, Examples*. 2018. Disponível em: <https://ubidots.com/blog/iot-dashboards/>.

PURWANTO, F. H.; UTAMI, E.; PRAMONO, E. Design of server room temperature and humidity control system using fuzzy logic based on microcontroller. *2018 International Conference on Information and Communications Technology (ICOIACT)*, p. 390–395, 2018.

PURWANTO, F. H.; UTAMI, E.; PRAMONO, E. Implementation and optimization of server room temperature and humidity control system using fuzzy logic based on microcontroller. *Journal of Physics: Conference Series*, v. 1140, n. 012050, p. 13, 2018.

ROBóTICA, A. *Módulo Sensor de Tensão AC ZMPT101B*. 2019. Disponível em: <https://www.autocorerobotica.com.br/modulo-sensor-de-tensao-ac-zmpt101b>.

SAHA, S.; MAJUMDAR, A. Data centre temperature monitoring with esp8266 based wireless sensor network and cloud based dashboard with real time alert system. *2017 Devices for Integrated Circuit (DevIC)*, n. 4, p. 307–310, 2017.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding Machine Learning: From Theory to Algorithms*. 1. ed. [S.l.]: Cambridge University Press, 2014.

SINGH, P. et al. Effect of relative humidity, temperature and gaseous and particulate contaminations on information technology equipment reliability. In: *Proceedings of the ASME 2015 International Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Microsystems and ASME 2015 13th International Conference on Nanochannels, Microchannels, and Minichannels*. University of Texas, Arlington, Texas, USA: ASME, 2015. (NIPS'00).

SOCIETY, A. M. *dewpoint*. 2015. Disponível em: <http://glossary.ametsoc.org/wiki/Dewpoint>.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introdução ao Data Mining*. 1. ed. [S.l.]: Ciência Moderna, 2018.

TARUTANI, Y. et al. Temperature distribution prediction in data centers for decreasing power consumption by machine learning. *2015 IEEE 7th International Conference on Cloud Computing Technology and Science*, p. 635–642, 2015.

TARUTANI, Y. et al. Reducing power consumption in data center by predicting temperature distribution and air conditioner efficiency with machine learning. *2016 IEEE International Conference on Cloud Engineering*, p. 226–227, 2016.

YHDC. *Split core current transformer*. [S.l.], 2019.