UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MILA STEFANNY CANTANHEDE TEIXEIRA

UM SISTEMA BASEADO EM *IoT* PARA APLICAÇÃO EM DATA CENTER COM MONITORAMENTO DE TEMPERATURA, UMIDADE E PRESSÃO ALIADO A UM CONTROLE TÉRMICO AUTOMÁTICO

SÃO LUÍS
2019

MILA STEFANNY CANTANHEDE TEIXEIRA

UM SISTEMA BASEADO EM *IoT* PARA APLICAÇÃO EM DATA CENTER COM MONITORAMENTO DE TEMPERATURA, UMIDADE E PRESSÃO ALIADO A UM CONTROLE TÉRMICO AUTOMÁTICO

Monografia apresentada ao Curso de Engenharia Elétrica da Universidade Federal do Maranhão como parte dos requisitos para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Denivaldo Cícero Pavão Lopes

SÃO LUÍS

2019

MILA STEFANNY CANTANHEDE TEIXEIRA

UM SISTEMA BASEADO EM *IoT* PARA APLICAÇÃO EM DATA CENTER COM MONITORAMENTO DE TEMPERATURA, UMIDADE E PRESSÃO ALIADO A UM CONTROLE TÉRMICO AUTOMÁTICO

Monografia apresentada ao curso de Engenharia Elétrica da Universidade Federal do Maranhão como parte dos requisitos para obtenção do título de Bacharel em Engenharia Elétrica.

Aprovada em: 07/01/2019

BANCA EXAMINADORA

Prof. Dr. Denivaldo Cícero Pavão Lopes (Orientador)
(Departamento de Engenharia Elétrica- UFMA)

Prof. Dr. Aristófanes Correa Silva
(Departamento de Engenharia Elétrica- UFMA)

Prof. Dr. Luciano Buonocore
(Departamento de Engenharia Elétrica- UFMA)

A Deus, ao meu pai Cláudio Teixeira (*in memoriam*), à minha família e aos meus amigos.

AGRADECIMENTOS

A Deus, em primeiro lugar sempre, pelo dom da vida, por todas as graças e dádivas que me foram contempladas. Sem O Senhor, meu Pai de infinita misericórdia, jamais conseguiria chegar até aqui. A Nossa Senhora da Boa Viagem, minha mãezinha querida, que me guia, ampara e auxilia.

Aos meus pais, em especial, a minha mãe Maria Lúcia por todo esforço para me educar e sustentar, pela força, companheirismo, carinho, inspiração, ensinamentos e incentivo a nunca desistir dos meus sonhos.

A todos os meus familiares pelo apoio, em especial, ao meu irmão Lúcio Flávio pela parceria, preocupação, carinho, motivação e aos meus queridos avós, pelo exemplo de pessoas que são, Maria Raimunda e José Cantanhede.

A todos os meus professores pelo ensino, pela dedicação e esforço para nos orientar bem, de modo particular, agradeço ao meu orientador, doutor Denivaldo Cícero Lopes, sempre disposto a contribuir e a somar esforços neste projeto.

A todos que de algum modo nos momentos serenos e/ou apreensivos fizeram ou fazem parte da minha vida. Em especial, a Thiago Leal, pelo amor, carinho, compreensão, paciência e apoio.

Aos meus amigos e colegas do curso de Engenharia Elétrica com os quais partilhei e/ou partilho experiências no estudo e na vida: Raissa, Juliana, Deyvison, Luana, Ana, Camila, Iranildo, Abmael, Noely, Brenda, Bianca, Patrícia, Matheus, Éder e Felipe.

Aos meus colegas de trabalho pelo apoio e ensinamentos compartilhados. Em especial: Claudinha, Lanise, Nelson, Argemiro, Thompson, Serginho, Altimar, Lucas, Fábio, Yves, Willam, Arnon, Regis, Claudionor e Marcos.

Enfim, a todos que durante essa jornada me fizeram acreditar que esse era um sonho possível. Muito obrigada!

"Por vezes sentimos que aquilo que fazemos não é senão uma gota de água no mar. Mas o mar seria menor se lhe faltasse uma gota".

(Madre Teresa de Calcutá)

RESUMO

Este projeto surge em uma problemática constante nos Centros de Processamento de Dados (*Data Center*), que é a segurança da informação, a qual é definida por um conjunto de métodos adotados estrategicamente para gerenciar e prevenir os riscos de danos aos dados, redes e servidores. Além disso, evitando falhas, intermitências operacionais e garantindo a disponibilidade das informações. Entretanto, para que isso ocorra é necessário implantar uma série de medidas preventivas, sendo uma das consideradas cruciais a garantia de temperatura em níveis adequados. Caso contrário, é possível que ocorram sérios problemas com os equipamentos e, consequentemente, na disponibilidade dos dados pelo *Data Center*. Nesse contexto, propõe-se uma solução de baixo custo, portátil e baseada na tecnologia Internet das Coisas com o intuito de monitorar as variáveis temperatura, umidade e pressão, além de automatizar o controle térmico. O sistema de monitoramento consiste basicamente em posicionar, de forma estratégica, os sensores DHT11 e BMP180 conectados ao microcontrolador ESP8266 que envia os dados, em intervalos regulares, para a nuvem do servidor *ThingSpeak,* através do protocolo HTTP (*HyperText Transfer Protocol*). Em conjunto, esses itens compõem o sistema de monitoramento via Internet, tornando possível visualizar as aferições *on-line*, bem como alertar os técnicos em caso de valores inadequados. Adicionado a isso, o sistema ainda consta com LEDs infravermelhos posicionados próximo ao ar-condicionado para que, caso a temperatura ultrapasse a faixa permitida, seja feita a alteração automaticamente. Os testes de medição assinalaram que o protótipo consegue aferir os valores de temperatura, umidade e pressão; e encaminhar os dados à nuvem. Além disso, realizar o controle térmico automático. Para potencializar o entendimento da proposta, os resultados obtidos desses testes foram analisados com o auxílio do *software* interativo de alta performance, *MATLAB® analytics,* presente dentro do *ThingSpeak*.

Palavras-chave: *Data Center*. Segurança da Informação. Controle de temperatura. Microcontrolador (ESP8266). IoT

ABSTRACT

This project arises in a constant problematic in the Data Center: information security. Which is defined by a set of methods strategically adopted to manage and prevent the risks of data, network and server damage. Besides that, to avoid failures, operational intermittency and ensure system availability. However, for this to occur, a number of precautionary measures must be implemented and one of the crucial measures is to ensure adequate temperature, otherwise serious problems with the equipment and, consequently, the availability of the data may occur. In this context, a low-cost portable solution based on the technology Internet of Things is proposed with the goal of monitoring the variables: temperature, humidity and pressure and also automate the thermal control. The monitoring system basically consists of strategically positioning the DHT11 and BMP180 sensors connected to the ESP8266 microcontroller which sends the data at regular intervals to the cloud of the ThingSpeak server through the Message Queue Telemetry Transport (MQTT) protocol. Together, these items make up the monitoring system via the Internet, making it possible to visualize the online measurements and, also, alert the technicians in case of inappropriate values. Added to this, the system still consists of infrared LEDs positioned next to the air-conditioning so that if the temperature exceeds the allowed range, the change is made automatically. Measurement tests indicated that the prototype can measure temperature, humidity and pressure values; and forward the data to the cloud. In addition, perform automatic thermal control. To enhance the understanding of the proposal, the results obtained from these tests were analyzed with the help of the high-performance interactive software *MATLAB*® analytics within ThingSpeak.

Keywords: Data Center. Information security. Temperature control. Microcontroller (ESP8266). IoT

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ASHRAE	*American Society of Heating, Refrigerating and Air-Conditioning Engineers*
ARP	*Adress Resolution Protocol*
DC	*Data Center*
DHCP	*Dynamic Host Configuration Protocol*
IBM	*International Business Machines*
ICMP	*Internet Control Message Protocol*
IDE	*Integrated Development Environment*
IGMP	*Internet Group Management Protocol*
IoT	*Internet of Things*
I2C	*Inter-Integrated Circuit*
HTTP	*HyperText Transfer Protocol*
HVAC	*Heating, Ventilating and Air Conditioning*
LED	*Light Emitting Diode*
MATLAB	*MATrix LABoratory*
MIT	*Massachusetts Institute of Technology*
MQTT	*Message Queue Telemetry Transport*
NodeMcu	*Node MicroController Unit*
NBR	Norma Brasileira
NTI	Núcleo de Tecnologia da Informação
PoP	Ponto de Presença
PTT	Ponto de Troca de Tráfego
RFID	*Radio frequency identification*
RNP	Rede Nacional de Ensino e Pesquisa
SCL	*Serial Clock Line*
DAS	*Serial Data Line*
SoC	*System on Chip*
SPI	*Serial Peripheral Interface*
SUB	*Subscriber*
TI	Tecnologia da Informação
TIC	Tecnologia da Informação e Comunicação

TSL	*Transport Layer Security*
UART	*Universal Asynchronous Receiver/Transmitter*
UDP	*User Datagram Protocol*
UFMA	Universidade Federal do Maranhão
USB	*Universal Serial Bus*
Wi-Fi	*Wireless Fidelity*

SUMÁRIO

1. INTRODUÇÃO

A Tecnologia da Informação (TI) é uma área que tem transformado o modo como as empresas gerenciam os seus negócios. Aliado a isso, tem-se a crescente necessidade de armazenamento de dados e os diversos benefícios proporcionados pela TI: aumento de produtividade, segurança na produção, aumento da lucratividade, melhoria nos resultados, etc. Tudo isso contribui para evidenciar a importância e o espaço que a mesma vem ganhando na atualidade.

Todavia, o investimento em infraestrutura de TI e a manutenção da mesma costumam ser bem onerosos, e, um dos principais responsáveis pelo custo elevado dessa área é o *Data Center* (DC), o qual exige cuidados constantes e redobrados com a segurança. Segundo Marin (2016), DCs são considerados ambientes críticos, ou seja, não podem parar a sua operação. São dedicados a hospedar servidores, equipamentos de armazenamento de dados, *hardware* de redes e demais equipamentos computacionais responsáveis pelo processamento da TI. São provedores de serviço, cujo objetivo principal é a entrega ininterrupta de serviços computacionais.

Dessa forma, este projeto sugere uma estação de monitoramento de temperatura, umidade e pressão através de sensores e o SoC (*System On Chip*) ESP8266. O SoC é um sistema altamente integrado que contém todos os componentes necessários para um projeto de *hardware* determinado, tais como processador, memória e periféricos (GANSSLE; BARR, 2003). Isto é, um sistema embarcado em um único chip, o qual apresenta, nesse caso, módulo para comunicação sem fio, microcontroladores integrados, memória adicional para armazenamento de dados, etc.

Embora no mercado já existam dispositivos capazes de realizar o mesmo monitoramento, este projeto prioriza um protótipo economicamente viável, visando a aplicabilidade em micro, pequenas e médias empresas. Não obstante, por se tratar de um protótipo embarcado e de baixo custo, algumas questões podem e serão levantadas, a título de exemplo: o sistema de medição é seguro? Qual o tempo de duração do mesmo? Qual a precisão dos sensores? Esses e outros questionamentos serão analisados no decorrer do projeto.

1.1 Contexto tecnológico

Atualmente, um dos conceitos mais promissores e pervasivos denomina-se internet das coisas ou IoT (*Internet of Things)*. Essa tecnologia com grande poder de ascensão e importância, vai muito além dos equipamentos de uso comum, como *smartphones*, *tablets* e computadores. Ela tem modificado o modo como as pessoas interagem com o meio, haja vista que o surgimento da mesma, aliada às novas tecnologias, abriu um leque de possibilidades que podem ajudar a solucionar, de maneira prática, problemas reais da sociedade.

O que hoje é chamado de IoT trata-se de um conjunto de tecnologias e protocolos associados que permitem que objetos se conectem a uma rede de comunicações e são identificados e controlados através desta conexão de rede (CAVALLI, 2016). Essa tecnologia tem sido objeto de atenção prioritária de governos e da iniciativa privada pelo mundo inteiro. Pois, trata-se da progressiva automatização de setores inteiros da economia e da vida social com base na comunicação máquina-máquina: logística, agricultura, transporte de pessoas, saúde, produção industrial. (MAGRANI, 2018)

Além do que, a IoT constitui uma das tecnologias mais importantes, com potencial para afetar profundamente nosso modo de vida (ABDMEZIEM; TANDJAOUI, 2014). E seus efeitos já estão aparecendo e afetando a vida das pessoas, tanto de forma positiva como negativa, embora o percentual de cada uma ainda seja impreciso.

Analisando pelo lado positivo, ela tem possibilitado diversas aplicações, tornando possível desenvolver projetos econômicos e aplicáveis nas mais variadas situações. A título de exemplo, o trabalho feito por Kaur e Jasuja (2017) mostra um sistema de monitoramento de sinais vitais do corpo humano. Esse sistema com baixo custo e baixo consumo de energia é capaz de monitorar batimentos cardíacos e temperatura corporal. Para isso os autores utilizam o Arduino UNO, sensores e um *Raspberry* PI para enviar os dados da leitura dos sensores pela internet, o protocolo MQTT é utilizado para a comunicação entre o sistema de monitoramento e o serviço de nuvem IBM *Bluemix*.

Entretanto, pelo lado negativo existem muitas questões a serem consideradas e uma delas trata-se de segurança. Segundo um estudo da Cisco, em 2020 haverá cerca de 50 bilhões de dispositivos conectados. Como garantir a privacidade dos indivíduos em uma sociedade

hiperconectada? Todos esses dispositivos coletarão uma quantidade estupenda de dados pessoais. E o que será feito com esses dados?

Segundo Magrani (2018) as falhas de segurança abrem espaço para ataques, visando ao acesso às informações geradas pelos próprios dispositivos. Além disso, os aparelhos inteligentes, quando invadidos, podem acarretar problemas não só para o aparelho em si, mas também interferindo na própria infraestrutura da rede. Essas são algumas questões que precisam ser analisadas com cautela quando se trata de projetos que visem segurança da informação.

1.2 Trabalhos Correlatos

É crescente o número de aplicações baseadas na tecnologia IoT, a título de exemplo, tem-se dois projetos práticos dos autores indianos Saha e Majumdar (2017). O primeiro projeto visa o monitoramento ambiental em hospitais e lares de idosos para a manutenção da temperatura, umidade e qualidade do ar.

A proposta se justifica, pois na Índia existem muitos casos de incêndio nos hospitais, sendo que os pacientes são as principais vítimas nesses casos. Na maioria das vezes, constata-se que a falta de uma indicação precoce é a causa dos principais danos. Dessa forma, eles propõem um sistema com sensores para temperatura (DHT11) e para verificar a qualidade do ar (MQ135) em conjunto com o ESP8266 e o servidor *Ubidots*.

No segundo projeto, tem-se por objetivo monitorar a temperatura de um centro de processamento de dados e notificar, através de mensagens ou e-mails, os funcionários, caso a temperatura ultrapasse um limite pré-estabelecido. O sistema consiste em sensores de temperatura (DHT11) conectados ao NodeMCU Devkit 1.0 que envia os dados, em intervalos regulares, para a nuvem do *broker Ubidots*. Em termos de software, foi utilizada a IDE (*Integrated Development Environment*) do Arduino e a linguagem de programação C.

1.3 Problemática

O Núcleo de Tecnologia da Informação (NTI) da Universidade Federal do Maranhão (UFMA) é um órgão técnico que tem por finalidade, de acordo com a resolução nº 126-CONSAD, assessorar a administração superior e apoiar os demais órgãos da UFMA em assuntos relativos à área de Tecnologia da Informação e Comunicação (TIC). E umas das áreas que compõem a sua estrutura organizacional é o Departamento de Redes, o qual gerencia a infraestrutura de rede e mantém seus serviços disponíveis à comunidade acadêmica e administrativa. Para manter esses serviços funcionando, esse departamento é subdividido em duas áreas: Divisão de Telefonia, responsável pela gestão da rede de voz, seja ela a telefonia analógica ou digital; e a Divisão de Infraestrutura de Rede, incumbido da gestão física e lógica do DC, e da rede UFMA.

Percebe-se, pela descrição acima, a importância do NTI para a universidade, e, sendo essa uma área de TI que utiliza muitos dados, não poderia deixar de ter um DC, o qual exige monitoramento e cuidados com o arrefecimento constante. No entanto, não há um sistema de monitoramento climático independente, ou seja, os técnicos fazem o acompanhamento da medição *in loco*, o que demanda tempo e recurso. Embora já existam diversos equipamentos no mercado que realizam as aferições, os preços ainda são elevados e dependentes da variação cambial, pois muitos componentes são produzidos fora do Brasil. Isso acaba dificultando a aquisição por parte de pesquisas científica e empresas de pequeno e médio porte, como é o caso do NTI. Então, o que fazer para solucionar esse problema, de forma econômica e prática?

1.4 Solução Proposta

A solução proposta neste projeto trata-se de um protótipo alicerçado em dois pilares: *hardware* e *software*. Em termos de *hardware*, propõe-se um circuito base, que será apresentado com mais detalhes no Capítulo 4 através de diagramas de interligação e fluxogramas, baseado no módulo ESP8266. Esse dispositivo IoT de baixo custo consiste em um processador ARM de 32 bits com suporte embutido à rede *Wi-Fi* e memória *flash* integrada, ou seja, uma arquitetura independente que permite ser programada sem a necessidade de interação com outras placas microcontroladas.

Para integrar o circuito, têm-se os sensores de temperatura, umidade e pressão, além dos atuadores. Neste projeto optou-se pelos sensores DHT11 e BMP180, e os atuadores infravermelho TL1838 e LED 5mm, responsáveis por fazer a comunicação do circuito com o ar-condicionado.

Já na parte de software, utilizou-se a IDE do Arduino e a linguagem C++. Embora a IDE do Arduino não tenha suporte para a placa de desenvolvimento NodeMCU ESP-12E, pode-se adicionar e instalar o pacote ESP8266. Dessa forma, possibilita-se a inclusão das bibliotecas do ESP8266 e a programação desejada. Sendo a IDE um ambiente de desenvolvimento integrado e de uso acessível, torna-se favorável para implementar os códigos utilizados no projeto aliado a uma linguagem de alto nível.

Resumidamente, a logística pensada para o sistema consiste em um circuito baseado no módulo ESP8266 que envia os dados, captados pelos sensores, em intervalos regulares, para o serviço de nuvem *ThingSpeak* através do protocolo HTTP (*HyperText Transfer Protocol*). Esses itens, em conjunto, compõem o sistema de monitoramento via *web* capaz de alertar possíveis anormalidades nas medições no DC. O sistema também utiliza os LEDs infravermelhos, baseado nos valores aferidos, para atuarem na parte de controle do processo.

1.5 Justificativa

Automatizar processos, economizar tempo e dinheiro é o ideal para qualquer investidor. A instalação de sistemas de monitoramento é uma forma de automatizar diversas tarefas, dentre elas o monitoramento do DC, onde ficam contidas as informações e sistemas da organização produtiva, comercial ou de serviços. Os equipamentos nesse local necessitam de um ambiente específico e uma das variáveis cruciais é a temperatura, ou seja, há necessidade do uso de condicionadores de ar, e uma monitoração de temperatura precisa.

No entanto, ter um profissional no ambiente, monitorando vinte e quatro horas por dia, é inviável. Ademais, fazer aferições esporadicamente aumentam as chances de ocorrer sérios problemas com os equipamentos, além da possível indisponibilidade parcial ou total dos serviços e, até mesmo, perdas irreparáveis.

Uma solução seria comprar um sistema robusto de monitoramento. Entretanto, pode aumentar os gastos além do estabelecido para manter os próprios equipamentos, o que seria inviável para algumas empresas e pesquisas. Nesse cenário, uma proposta alternativa seria um protótipo de um sistema de monitoramento de temperatura para DC com uso de redes de sensores sem fios, com custo benefício favorável, torando essa uma proposta alternativa, prática e econômica.

1.6 Objetivos

Os objetivos do desenvolvimento desta de monografia estão organizados em geral e específicos como segue.

1.5.1 Objetivo geral:

Propor uma solução de baixo custo para monitoramento de temperatura, umidade e pressão em um *Data Center* utilizando a tecnologia IoT de forma a realizar o controle automático da temperatura.

1.5.2 Objetivos específicos

Para atingir o objetivo geral, as seguintes etapas foram seguidas:

- Escolher entre os diversos dispositivos os que mais se adequam à aplicação: sensores e atuadores;
- Validar a proposta do sistema de baixo custo em diferentes cenários climáticos;
- Lidimar a interação entre as etapas de monitoração e controle do sistema desenvolvido;
- Ajustar parâmetros para medição e realizar testes em laboratório para verificação de precisão dos componentes;
- Analisar os resultados.

1.7 Metodologia

A metodologia utilizada na execução desta monografia consiste em uma pesquisa bibliográfica sobre o tema, ressaltando as características e funções de cada componente utilizado no projeto e demonstrar como cada componente funciona e a função que exerce no sistema final. O protótipo desenvolvido utilizará como base a plataforma de desenvolvimento NodeMCU ESP8266 que possui um módulo Wi-Fi próprio e apresenta um custo benefício favorável. Os principais conceitos de IoT, microcontrolador ESP8266 e DC serão aplicados em uma abordagem quantitativa e empírica através da criação de uma estação de monitoramento e das aferições feitas com a mesma. Os resultados dos testes são utilizados ainda para potencializar o entendimento em conjunto com uma análise crítica em relação aos prós e contras relativos ao protótipo proposto.

1.8 Estrutura da Monografia

Este trabalho está organizado em 06 (seis) capítulos, para propiciar o melhor entendimento do mesmo:

a) Sendo que o capítulo 1 tem por objetivo contextualizar o leitor, fornecendo informações pertinentes sobre a ideia base do trabalho;
b) O capítulo 2 possui uma revisão bibliográfica, apresentando as informações referentes aos protocolos, equipamentos utilizados neste projeto: sensores, o módulo Wi-Fi NodeMcu ESP8266, os LEDs emissores e receptores;
c) O capítulo 3 apresenta as principais características de um DC, os sistemas de arrefecimento, eficiência, as consequências das varáveis umidade, temperatura e pressão para os equipamentos do DC;
d) O capítulo 4 apresenta uma descrição sobre os procedimentos para a realização da implantação da estação de medição e controle;
e) No capítulo 5 são expostos todos os resultados obtidos por meio dos testes e estudo de caso elaborado durante o projeto;
f) Por fim, o capítulo 6 analisa os prós e os contras do protótipo, apresenta propostas futuras que poderão melhorar o tema e as considerações finais do projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma revisão bibliográfica e o embasamento teórico sobre as principais tecnologias empregadas no desenvolvimento do projeto, bem como, os principais conceitos e elementos utilizados para subsidiá-lo com o intuito de fundamentar a metodologia proposta.

Serão vistos conceitos sobre a tecnologia IoT, suas aplicações; conceitos de DC; a relação da umidade, temperatura e pressão em um DC; os principais conceitos dos protocolos usados no projeto; o funcionamento dos dispositivos empregados no protótipo e uma análise geral do sistema.

2.1 Internet das coisas

O avanço da tecnologia IoT tem modificado o modo como as pessoas interagem com o meio. E esse progresso vai muito além de receber uma mensagem quando algum item está em falta na geladeira. Em geral, objetivam-se soluções práticas para problemas reais na sociedade.

A título de exemplos tem-se: o estacionamento inteligente localizado no município de Águas de São Pedro, região central do estado de São Paulo, com sensores instalados em suas vias públicas, permitindo checar em tempo real a disponibilidade de vagas nos estacionamentos por meio de um aplicativo para *smartphones* ou *tablets*. Esse é um investimento que oferece praticidade aos moradores, turistas e diminui os transtornos com congestionamentos (MEIRELLES, 2014).

Apesar de o acrônimo IoT soar novo para muitos, ele já existe há quase vinte anos. Sua origem é atribuída a Kevin Ashton, pesquisador britânico do MIT (*Massachusetts Institute of Technology*), que em 1999 em uma apresentação para executivos da Procter & Gamble (P&G) utilizou a expressão pela primeira vez.

Em 2009, referenciou-se a apresentação de Ashton num artigo através do RFID *Journal* que cita o que é tido por muitos como a definição de IoT:

> […] Se tivéssemos computadores que soubessem tudo o que há para saber sobre coisas, usando dados que foram colhidos, sem qualquer interação humana, seríamos capazes de monitorar e mensurar tudo, reduzindo o desperdício, as perdas e o custo. Gostaríamos de saber quando as coisas precisarão de substituição, reparação ou atualização, e se eles estão na vanguarda ou se tornaram obsoletas, (ASHTON, 1999).

Não há um conceito único que defina a tecnologia IoT que possa ser considerado pacífico ou unânime. De maneira geral, pode ser entendido como um ambiente de objetos físicos interconectados com a internet por meio de sensores pequenos e embutidos, criando um ecossistema de computação onipresente (ubíqua), voltado para a facilitação do cotidiano das pessoas, introduzindo soluções funcionais nos processos do dia a dia. O que todas as definições de IoT têm em comum é que elas se concentram em como computadores, sensores e objetos interagem uns com os outros e processam informações/dados em um contexto de hiperconectividade. (MAGRANI, 2018)

Ainda segundo Magrani, professor e pesquisador do Centro de Tecnologia e Sociedade da Fundação Getúlio Vargas (FGV), a estimativa de impacto econômico global vinculado ao cenário de IoT corresponde a mais de US$ 11 trilhões em 2025. Um exemplo desse tipo de investimento é o realizado pela empresa IBM, que não só empregou US$ 3 bilhões em seu negócio de IoT, como também fez uma parceria com a AT&T para fornecer soluções IoT industriais em uma série de questões, desde a eficiência energética até serviços de saúde.

Essas novas frentes de investimento em IoT decorrem das perspectivas de lucro positivo do setor. Somente a título de exemplo, cabe ressaltar a pesquisa recente realizada pela Cisco que estima que a IoT pode adicionar 352 bilhões de dólares à economia brasileira até o final de 2022. Por conta de previsões como essa, diversas empresas estão investindo mais em tecnologias que envolvem o tema, desenvolvendo iniciativas concretas em diversas áreas.[1]

---

[1] Entrevista disponível no site: https://computerworld.com.br/2017/05/04/iot-vai-injetar-us-11-trilhoes-na-economia-mundial-em-ate-2025-diz-professor-da-fgv/

Muitas empresas estão investindo nessa ferramenta e a tendência é só aumentar o número de dispositivos conectados à Internet. E se viver em um ambiente hiperconectado monitorado e controlado via internet era algo, praticamente, impensável até a década de 1990. Essa já é uma realidade atual que vem sendo largamente discutida, estudada, aprofundada e melhorada, pois assim como qualquer tecnologia, existem muitas vantagens, mas também há muitos pontos que precisam melhorar.

2.2 TCP/IP

O protocolo TCP (*Transmission Control Protocol*) e IP (*Internet Protocol*) integram o TCP/IP, conjunto de protocolos hierárquicos, compostos por módulos interativos, cada um dos quais provendo funcionalidades específicas. O termo hierárquico significa que cada protocolo de nível superior é suportado por um ou mais protocolos de nível inferior (FOROUZAN, 2008).

Para facilitar o processo de padronização e obter interconectividade entre máquinas de diferentes fabricantes, a Organização Internacional de Padronização (ISO- *International Standards Organization)* aprovou, no início dos anos 80, um modelo de referência para permitir a comunicação entre máquinas heterogêneas, denominado OSI (*Open Systems Interconnection*). Esse modelo serve de base para qualquer tipo de rede, seja de curta, média ou longa distância[2].

Diferentemente do modelo OSI, o modelo de comunicação TCP/IP emprega apenas quatro camadas para seu funcionamento, são elas: camada de aplicação, camada de transporte, camada de internet e camada de rede. Dessa forma, cada uma das camadas do modelo TCP/IP pode corresponder a uma ou mais camadas do modelo OSI[3].

A camada de aplicação no modelo TCP/IP corresponde às camadas de aplicação, apresentação e sessão do modelo OSI. É a responsável por iniciar o processo de transmissão de

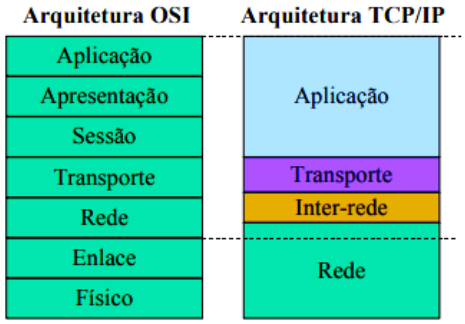
---

[2] Disponível em: <http://professor.unisinos.br/linds/introcomp/redesintro.pdf>. Acesso em novembro de 2018.
[3] Disponível em:<https://www.juliobattisti.com.br/tutoriais/paulocfarias/redesbasico018.asp>. Acesso em novembro de 2018.

dados. Os protocolos da camada de aplicação mais conhecidos permitem a troca de informações entre usuários.

As camadas de transporte dos modelos TCP/IP e a camada de transporte do modelo OSI possuem correspondência direta, sendo ela responsável por garantir a comunicação entre *hosts* e levar os dados da camada de aplicação para a da internet. Conforme ilustra a Figura 1, o modelo TCP/IP é formado por quatro camadas, são elas: aplicação, transporte, internet e interface de Rede. Outro modelo equivalente ao TCP/IP é o Modelo OSI, que possui sete camadas.

Figura 1– Modelos OSI e TCP/IP



Fonte: Internet e Arquitetura TCP/IP - PUC RIO/CCE[4].

A camada de rede do modelo OSI é a responsável pelo endereçamento e roteamento dos pacotes transmitidos. Sendo que a mesma também apresenta alguns protocolos auxiliares que ajudam o IP nas tarefas de entrega e roteamento. O ICMP (*Internet Control Message Protocol*, ou protocolo de Mensagens de Controle da Internet) ajuda o IP a relatar alguns problemas durante o roteamento de pacotes. O IGMP (*Internet Group Management Protocol*, ou Protocolo de Gerenciamento de Grupos Internet) é outro protocolo que ajuda o IP, neste em tarefas de *multicast*. O DHCP (*Dynamic Host Configuration Protocol*, ou Protocolo de Configuração Dinâmica de *Host*) ajuda o IP a obter o endereço de camada de rede para um *host*. O ARP (*Adress Resolution Protocol*, ou Protocolo de Resolução de Endereços) é um protocolo que ajuda o IP a localizar o endereço da camada de enlace de um *host* ou de roteador quando o endereço de camada é dado (FOROUZAN; MOSHARRAF, 2013).

4 Disponível em: <ftp://ftp.univap.br/pub/manutencao/Drivers/CD_Rec_HDs_e_Outros/Cursos%20e%20Apostilas/Redes%20-%20Arquitetura%20TCP-IP%20Parte%201.pdf>. Acesso em nov.2018.

As camadas mais baixas do protocolo TCP/IP, correspondem à camada de enlace e à camada física do modelo OSI, são responsáveis por determinar a forma como o dispositivo se conecta à rede. É nesse nível que ocorre o envio de datagramas produzidos na camada de internet e que estão todas as especificações e protocolos de acesso referentes à transmissão de dados em uma rede física, também estão presentes os protocolos LAN (*Local Area Network*) de enlace da rede local de computadores, sendo que sua aplicação geográfica apresenta "cobertura limitada" quanto à extensão que pode atuar. Além da WAN (*Wide Area Network*) ou rede de longa distância, corresponde a uma rede de computadores que abrange uma grande área geográfica (FRANCISCATTO et al, 2014).

O Wi-Fi é uma tecnologia de WLAN (*Wireless Local Area Network*) que segue o padrão IEEE 802.11 e está incluído na última camada do modelo TCP/IP, ou nas camadas 1 e 2 do modelo OSI. Esse padrão estabelece normas para criação de redes sem fio. A comunicação é feita por radiofrequência e com o alcance de algumas centenas de metros. O padrão IEEE 802.11 foi apenas o início das redes sem fio, apresentando uma taxa de dados muito baixa, próximo a 2Mbps. Atualmente, o padrão 802.11g é o mais utilizado e trabalha na faixa de frequência de 2.4GHz e possui uma taxa de transferência de dados de até 54 Mbps. Existem outros padrões com taxa de transferência maior o 802.11n e o 802.11ac. Entretanto, ainda apresentam custo elevado e possuem poucos dispositivos compatíveis no mercado (OLAIR, 2017).

2.3 HTTP

HTTP (*HyperText Transfer Protocol* ou Protocolo de Transferência de Hipertexto) é o protocolo mais utilizado para enviar e receber dados pela Internet. Um navegador Web pode ser chamado de cliente HTTP, enquanto um servidor Web seria um servidor HTTP. O funcionamento geral do HTTP é o seguinte: um cliente HTTP abre uma conexão e envia uma solicitação e o servidor HTTP correspondente recebe a mensagem, gera outra mensagem com a resposta adequada (MINERA, 2010).

Esse padrão de solicitação e resposta tem algumas limitações graves:

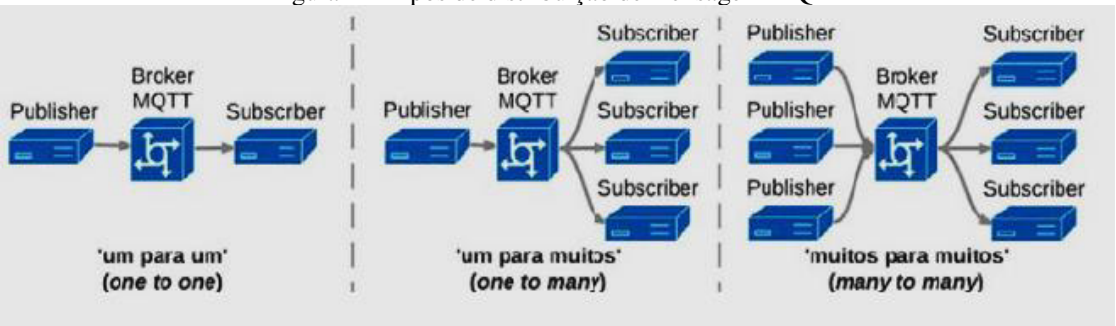
a) O HTTP é um protocolo síncrono. O cliente espera que o servidor responda. Os navegadores da web têm esse requisito, mas o custo é a baixa escalabilidade. No mundo da IoT, a comunicação síncrona tem sido um problema devido ao grande número de dispositivos e à rede, muitas vezes não confiável e de alta latência. Um protocolo de mensagem assíncrono é muito mais adequado para aplicativos de IoT. Os sensores podem enviar leituras e permitir que a rede descubra o caminho e a sincronização ideais para entregar aos dispositivos e serviços de destino;
b) HTTP é unidirecional. O cliente precisa iniciar a conexão. Em um aplicativo de IoT, os dispositivos e sensores geralmente são clientes, o que significa que eles não podem receber comandos da rede passivamente;
c) HTTP é um protocolo de um para um. O cliente faz uma solicitação e o servidor responde. É difícil e caro transmitir uma mensagem a todos os dispositivos na rede, o que é um caso de uso comum em aplicativos de IoT;
d) HTTP é um protocolo pesado com muitos cabeçalhos e regras. Ele não é adequado para redes restringidas. (YUAN, 2017)

Um diferencial dos protocolos para tecnologia IoT é o MQTT (*Message Queue Telemetry Transport*) é um protocolo aberto de mensagens projetado para comunicação M2M (*Machine-to-Machine*). É um protocolo leve e requer pouca largura de banda, além disso, permite vários clientes conectados simultaneamente a um único servidor.

O MQTT foi criado para ser um protocolo de mensagem usado sobre o protocolo TCP/IP e baseia-se em um esquema de *publisher/ subscriber* (publicador/assinante) (YUAN, 2017), onde todos os dados são enviados para um intermediário, denominado *broker*, ou seja, servidor que faz a intermediação entre o *publisher* e *subscriber*, ele recebe e organiza as mensagens e se encarrega de enviar as mensagens ao destinatário. Uma característica importante do MQTT é que um c*lient* pode ser *publisher* e *subscriber* ao mesmo tempo. MQTT (2017)

Uma qualidade da arquitetura MQTT é que o protocolo permite desacoplar o produtor do cliente, assim, apenas o endereço do *broker* precisa ser conhecido, possibilitando a comunicação, conforme Figura 2: de um para um (one-to-one), um para muitos (one-to-many) ou muitos para muitos (many-to-many).

Figura 2 – Tipos de distribuição de mensagem MQTT



Fonte: TORRES (2016).

Em termos de segurança, até a versão 3.1.1, o protocolo não possuía qualquer tipo de criptografia. O método de autenticação atual é por usuário e senha. Entretanto, isso não impede que o protocolo TLS (*Transport Layer Security*) de segurança e criptografia possa ser utilizado de forma independente.

2.4 I2C

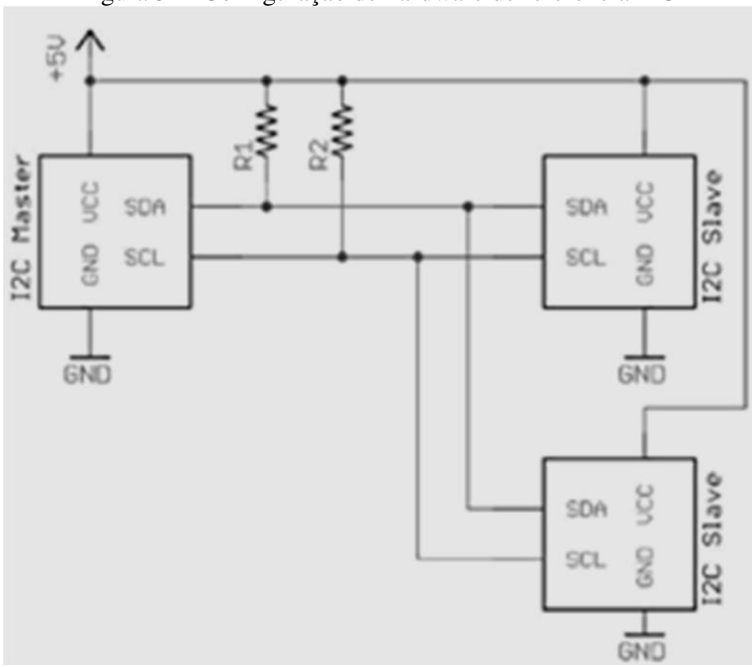
O protocolo I2C foi desenvolvido pela Philips no início da década de 1980 e padronizado na década de 1990. Tendo por objetivo: permitir a comunicação de maneira simples e eficiente entre componentes eletrônicos de uma mesma placa, sendo muito utilizado para conectar periféricos de baixa velocidade a microcontroladores e afins. Esse protocolo foi desenvolvido para suportar uma taxa de comunicação de 100kbps (100.000 bits por segundo). Em sua última versão, 4.0 de 2012, permite que os componentes ajustem taxas de até 5Mbps. (ALMEIDA et al, 2016)

A utilização do barramento I2C é muito popular e poderosa para fazer comunicação entre um dispositivo mestre e vários outros dispositivos escravos, (VALDEZ, 2015). Muitos sensores e conversores A/D utilizam a comunicação I2C. O barramento I2C é controlado por um dispositivo mestre (normalmente um microcontrolador), e contém um ou mais dispositivos escravos que recebem informações do mestre.

No protocolo I2C vários dispositivos compartilham as mesmas linhas de comunicação: um sinal de clock, SCL (*Serial Clock*) que serve para temporização entre os dispositivos, de modo que a comunicação possa ter confiabilidade; e uma linha bidirecional de comunicação

usada para troca de informações entre um mestre e seus escravos SDA (*Serial Data*). A Figura 3 mostra uma configuração de referência comum para um sistema de comunicação I2C.

Figura 3 – Configuração de hardware de referência I2C



Fonte: BLUM (2016).

O ESP8266EX possui um I2C, que é realizado via programação de software, usado para conectar outros microcontroladores e outros equipamentos periféricos, como sensores. A configuração de pino é mostrada na tabela 1.

Tabela 1 – Definições de pino I2C

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	I2C_SCL
GPIO2	14	IO2	I2C_SDA

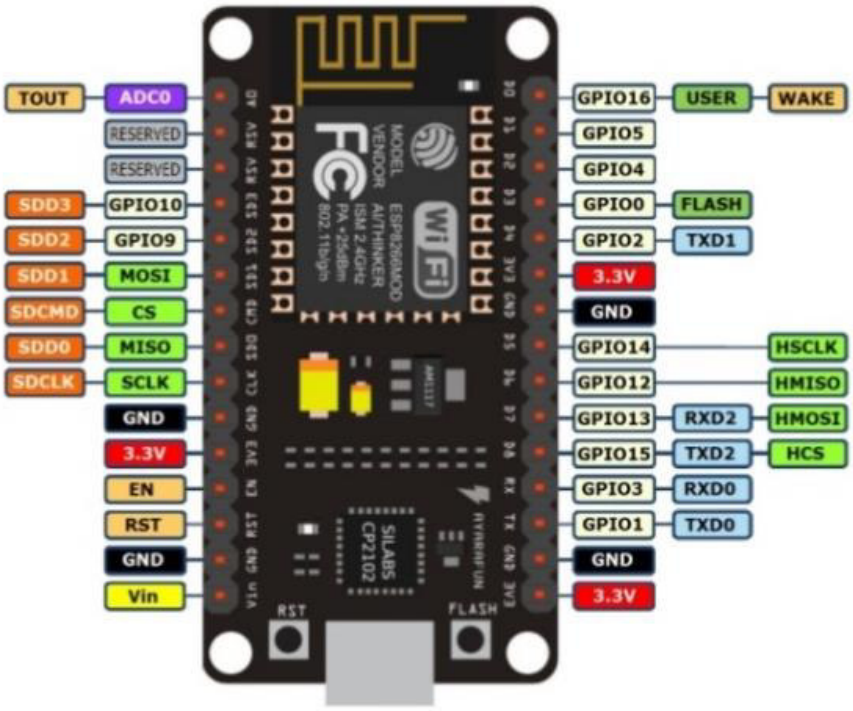
Fonte: Datasheet- ESP8266EX

2.5 NodeMCU

A NodeMCU (*Node MicroController Unit*) é um ambiente de desenvolvimento de software e hardware livre criado em torno de um SoC (*System-on-a-Chip*) titularizado ESP8266, conforme mostra a Figura 4. Em 2014, a empresa chinesa *Espressif Systems* lançou as primeiras versões do microcontrolador ESP, o qual apresenta algumas vantagens em relação

aos outros módulos, pois o chip permite fazer a integração do microprocessador a uma rede sob o padrão IEEE 802.11 b/g/n (Wi-Fi) sem a necessidade de um componente adicional, como é o caso do Arduino, o que o torna mais atrativo em termos práticos e financeiros.

Figura 4 – Pinos ESP8266



Fonte: MURTA (2018)

O circuito do ESP8266 é integrado, com interfaces I/O (*input* e *output*) digitais e analógicas, ao todo são 16 portas GPIO, comunicação I2C, UART, SPI, um módulo ADC, uma CPU da *Tensilica* L106 de 32 bits com a arquitetura *Xtensa, clock* de 80MHz podendo chegar a 160MHz, (ESPRESSIF, 2017).

Além disso, contém os principais elementos de um computador moderno: CPU, RAM, rede (Wi-Fi) e até mesmo um moderno sistema operacional e SDK e ainda apresentar um baixo custo para a aquisição desse módulo, em torno de 2 dólares. O que o torna uma excelente opção para projetos de IoT de todos os tipos.

As principais características estão resumidas na Tabela 2.

Tabela 2 – Especificações ESP8266

Categoria	Itens	Parâmetros
Wi-Fi	Certificação	Wi-Fi Alliance
	Protocolos	802.11b/g/n
	Range de Frequência	2.4GHz a 2.5GHz
	Antena	PCB Trace, External, IPEX Connector, Ceramic Chip
	CPU	Tensilica L106 32-bit processor
	Periféricos	UART/HSPI/I2C/I2S/Ir Remote Control/GPIO/PWM
	Tensão de operação	2.5V a 3.6V
	Corrente de operação	80Ma
	Range temperatura de operação	–40°C a 125°C
Software	*Wi-Fi Mode*	Station/SoftAP/SoftAP+Station
	Segurança	WPA/WPA2
	Encriptação	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Protocolos de rede	IPv4, TCP/UDP/HTTP
	Configuração do usuário	AT Instruction Set, Cloud Server, Android/iOS App

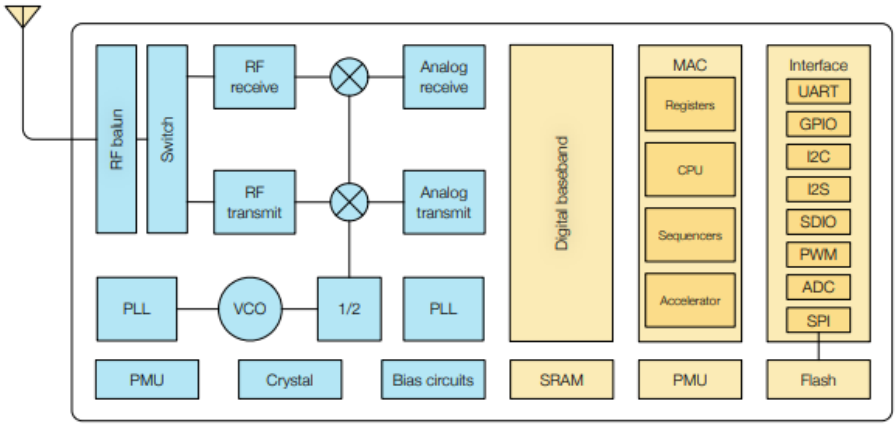
Fonte: Tabela baseada nos dados do Datasheet- ESP8266EX (2018).

O ESP8266EX oferece uma solução de rede Wi-Fi completa e autônoma. Ele pode ser usado para hospedar o aplicativo ou para acessar as funções de rede Wi-Fi de outro processador de aplicativos. Quando o ESP8266EX hospeda o aplicativo, ele é inicializado diretamente de um *flash* externo.

ESP8266EX Wi-Fi SoC integra controlador de memória e unidades de memória, incluindo SRAM e ROM. O MCU pode acessar as unidades de memória por meio de interfaces iBus, dBus e AHB. Todas as unidades de memória podem ser acessadas mediante solicitação, enquanto um árbitro de memória decidirá sequência de execução de acordo com o momento em que esses pedidos são recebidos pelo processador.

O diagrama de blocos do ESP8266EX é demonstrado na Figura 5. Em suma, o microcontrolador integra os interruptores de antenas, RF balun (dispositivo que casa impedâncias), amplificador de potência, amplificador de recepção de baixo ruído, filtros e módulos de gestão de energia.

Figura 5 – ESP8266EX Diagrama de Blocos



Fonte: Data Sheet ESP8266EX (2018).

A parte de RF (Rádio Frequência) do microcontrolador é formada pelos seguintes blocos:

a. Receptor: converte o sinal RF para um sinal em quadratura e em banda base, em seguida digitaliza-o por 2 conversores A/D (Analógico/Digital) de altas precisão e velocidade;
b. Transmissor: converte o sinal em quadratura para um sinal em banda passante de 2,4GHz, utilizando um amplificador CMOS para ligar a antena, sendo que todos os componentes utilizados pelo gerador de clock estão integrados no microcontrolador, incluindo o indutor, o varicap (diodo que possui uma capacitância variável em função da tensão aplicada) e o PLL (*Phase Locked Loop*) que é um oscilador que fornece o pulso de leitura para o conversor A/D na frequência de amostragem utilizada;
c. Gerador de clock de alta precisão: gera sinais em quadratura de 2,4GHz para serem utilizados pelo receptor e pelo transmissor;
d. Relógio de tempo real, reguladores e gerenciador de energia; e
e. Filtros RF: utilizados para contornar os problemas gerados pelas condições adversas do canal de comunicação, integra-se a ele o controle automático de ganho (AGC) e circuitos que cancelam o *offset* do nível DC, através de calibrações.

Em termos de aplicações, o ESP8266 pode rodar em projetos que usam Wi-Fi, pois contém todas as ferramentas para se estabelecer essa comunicação: conectores para antenas ou

antena integrada, RF balun, amplificadores, filtros. Além disso, ele também pode servir de adaptador Wi-Fi a qualquer projeto baseado em microcontroladores com conectividade simples, ou seja, interface SPI / SDIO ou I2C / UART.

2.6 Sensores

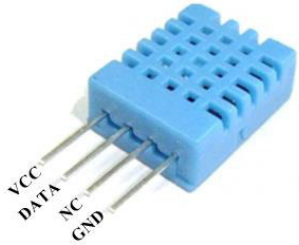
Neste projeto utilizou- se os sensores DHT11 responsável pela aferição da temperatura e umidade e o BMP180 pela pressão.

2.6.1 DHT11

O DHT11, representado na Figura 6, é um sensor digital capaz de obter a temperatura e umidade do ambiente. Ele é indicado para as mais variadas aplicações, por exemplo: Aquecimento, ventilação e ar condicionado ou simplesmente HVAC (*Heating, Ventilating and Air Conditioning*), desumidificador, equipamento de teste e inspeção, bens de consumo, automotivo, automático controle, registradores de dados, estações meteorológicas, eletrodomésticos, regulador de umidade e tantas outras.

O DHT11 não é um sensor de alta precisão, ele trabalha com a tensão elétrica entre 3.3V e 5.5V com a corrente máxima de 2.5mA, com a faixa de umidade entre 20 e 90% e faixa de temperatura entre 0 e 50 °C. A precisão do sensor é de 2 °C e 5% de umidade. Das características do DHT11, pode ser destacado: o baixo consumo de energia, sinal digital calibrado, o fato de não necessitar de mais componentes para poder usar e as bibliotecas disponíveis para suporte. (SUNROM, 2012).

Figura 6 – DHT11



Fonte: *Datasheet*, AOSONG (2014).

A Tabela 3 mostra as principais especificações técnicas do DHT11.

Tabela 3 – Especificações técnicas DHT11

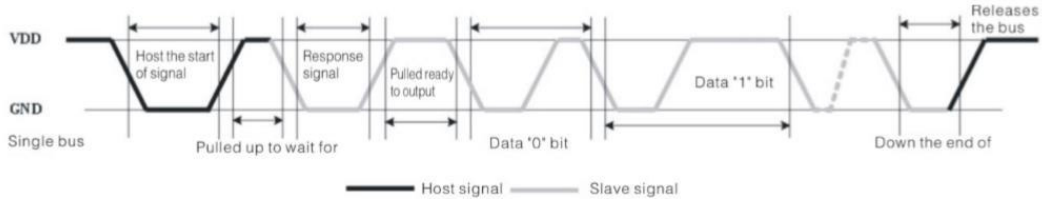
Especificações técnicas	
Alimentação	3 a 5VDC
Corrente	200uA a 500mA
Faixa de medição de temperatura	0° a 50°C
Faixa de medição de umidade	20 a 90% UR
Precisão de medição de temperatura	± 2.0 °C
Precisão de umidade de medição	± 5,0% UR

Fonte: *Datasheet*, AOSONG (2014).

O DHT11 é composto por um circuito integrado que une um sensor capacitivo para medição de umidade relativa do ar e um termistor para medição de temperatura ambiente. Ele se comunica com o ESP8266 através de uma interface serial, que funciona da seguinte forma: inicialmente o ESP8266 envia um degrau negativo de cerca de 18ms e em seguida é enviado em degrau positivo de curta duração (20µs a 40 µs), após isto o sensor envia os dados para o módulo.

O sensor envia 5 pacotes de 8 bits. O primeiro byte informa a parte inteira da umidade, o segundo transmite a parte decimal da umidade, o terceiro pacote envia o valor inteiro da temperatura, o quarto byte informa os valores decimais da temperatura e, por fim, é enviado um byte de verificação de envio que é retratado na Figura 7.

Figura 7 – Interface serial DHT11



Fonte: *Datasheet*, AOSONG (2014).

Devido a este processo cada ciclo de leitura tem duração de s, tempo aceitável tornando a medição praticamente em tempo real. Apesar do DHT22 ser preciso, conforme Tabela 4, utilizou-se neste projeto o DHT11, sendo que, em laboratório as medições atenderam ao esperado e o tempo de resposta entre as medições é menor.

Tabela 4 – Comparação entre DHT11 e DHT22

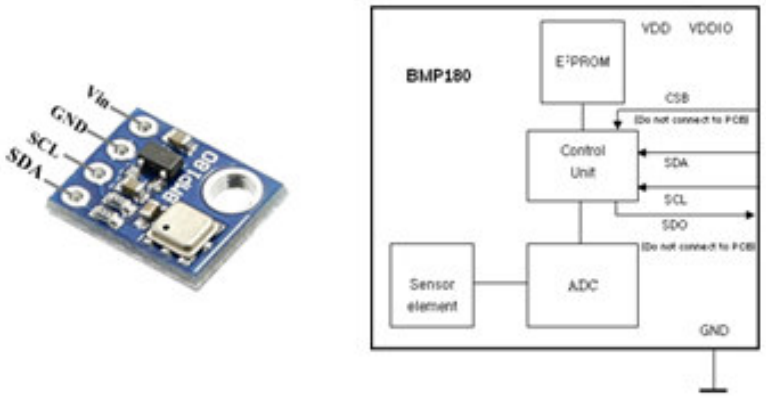
	DHT11	DHT22
		
Tensão de Alimentação	3V - 5.5V	3.3V - 6V
Corrente Máxima(Captura Dados)	2.5mA	1.5mA
Faixa de Leitura - Umidade	20 - 80%	0 - 100%
Precisão - Umidade	5%	2%
Faixa de Leitura - Temperatura	0 - 50°C	- 40°C - 125°C
Precisão - Temperatura	+/- 2°C	+/- 0.5°C

Fonte: LIMA (2018).

2.6.2 BMP180

O Sensor de Pressão BMP180 ou também conhecido como Barômetro (Figura 8) é uma versão melhorada do chip BMP085. O BMP180 é um módulo digital de alta capacidade e resolução baseado no chip BMP180 desenvolvido pela Bosch, apresenta baixo consumo de energia e é utilizado nas mais diversas aplicações, como: projetos como *drones*, estações meteorológicas, dispositivos com GPS, relógios, etc.

Figura 8 – BMP180 e arquitetura



Fonte: *Datasheet*- BOSCH (2013).

O sensor BMP 180 tem quatro importantes periféricos internos: Sensor elemento, ADC (conversor analógico para digital), unidade de controle e a memória não volátil EEPROM (*Electrically-Erasable Programmable Read-Only Memory*) com capacidade de 176 bits. O elemento sensor fornece um sinal transduzido e amostrado para o ADC, então o ADC converte o sinal em dados digitais, a unidade de controle se comunica com os periféricos externos através

da interface serial I2C, o que vem a aumentar a praticidade de sua utilização. Ele suporta alimentação de 1.8 a 3.6V. A Tabela 5 resume as principais especiações técnicas.

Tabela 5 – Especificações técnicas BMP180

Especificações técnicas	
Tensão de operação	1.8 a 3.6VDC
Consumo de corrente	0.5µA
Faixa de leitura de pressão	300 à 1100hPa (+9000 a -500m)
Conexão	I2C
Sensor de temperatura embutido	-40 à +85°C
Calibração	De fábrica

Fonte: *Datasheet*, Bosch.

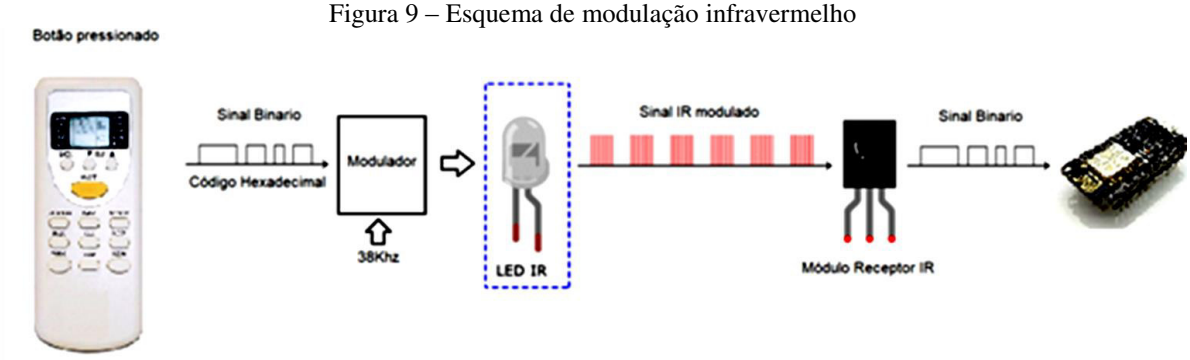
2.6.3 LED infravermelho

As frequências das ondas infravermelhas variam entre 300 GHz a 300THz e a radiação infravermelha são ondas de comprimento de 1mm até 700nm, e, portanto, não visíveis para o olho humano. Por isso, é uma radiação considerada não ionizante. No espectro de luz, está localizado depois da luz vermelha, daí surgiu seu nome.

Apesar de não poder ser vista, a radiação infravermelha pode ser notada no corpo em forma de calor: terminações nervosas, chamados termorreceptores, conseguem captar essa radiação.

Sua descoberta vem do ano de 1880. Willian Herschel, astrônomo inglês, estava fazendo estudos relacionados à temperatura das cores. Em suma o sensor e o atuador operam conforme a Figura 9. O LED emissor emite um trem de pulsos que pode ser capturado por um LED receptor.

Figura 9 – Esquema de modulação infravermelho



Fonte: Adaptado- MADEIRA (2018)

2.6.3.1 Receptor

No caso deste projeto, utilizou-se o modelo TL1838 VS1838B 38Hz (Figura 10) que foi programado a partir de uma biblioteca própria (*IRremoteESP8266.h*) para ser utilizada com o ESP8266. Tal biblioteca recebe e analisa os dados recebidos pelo sensor, e realiza o reconhecimento da marca do ar condicionado. Assim, uma vez reconhecido torna-se possível efetuar o controle através do LED de infravermelho que será o responsável por enviar as frequências que correspondem a comandos para o ar condicionado.

Figura 10 – LED receptor

Fonte: Próprio autor (2018)

O modelo em questão apresenta um sensor VS1838. Este sensor, além de possuir um fotodiodo infravermelho, contém alguns componentes internos para fazer a correta leitura de pulsos de 38kHz.

Os receptores consistem de um circuito eletrônico com um fotorreceptor sensível à luz infravermelha (invisível para o ser humano), que capta o sinal emitido pelo emissor e converte as variações do feixe de luz em variações de corrente elétrica. Quanto menor a distância entre o conjunto sensor e o obstáculo, maior será a quantidade de luz infravermelha, por conseguinte, maior será a corrente que atingirá o sensor.

2.6.3.2 Emissor

Utilizou-se o LED emissor 5mm, conforme Figura 11. Ele é constituído de um circuito eletrônico com um LED infravermelho que quando acionado por um sinal de controle emite luz na forma de um trem de pulsos com uma determinada frequência.

Figura 11 – LED emissor 5mm

Fonte: Próprio autor (2018)

Existe um problema na transmissão de dados via luz infravermelha, que é a interferência da luz infravermelha do ambiente. Para contornar esse obstáculo, surgiram diversos protocolos para padronizar a comunicação da luz infravermelha de forma que o ambiente não fizesse interferência. Por padrão, os protocolos definem que a comunicação é feita com pulsos em uma frequência de 38kHz. E é a partir do trem de pulsos que um ar condicionado pode diferenciar a luz que de um controle remoto ou de algum outro objeto.

2.7 ThingSpeak

O *ThingSpeak* (Figura 12) é uma plataforma de *IoT* que permite agregar, visualizar e analisar fluxo de dados em tempo real na nuvem. Ele é um serviço gratuito que permite comunicação através de um protocolo baseado em HTTP. Para enviar dados para o servidor, deve-se criar uma conta no site thingspeak.com e criar um canal para receber as informações dos sensores. Os canais criados são personalizáveis e podem armazenar até 8 campos com informações.

Figura 12 – Servidor

Fonte: Próprio autor (2018).

O ThingSpeak é uma plataforma IoT gratuita de código aberto. Mas apresenta limitação de criação de canais, 8. Acima disso, é necessário pagar uma determinada taxa. Esse API permite o upload de dados numéricos de sensores, tais dados ficam armazenados na nuvem, onde então é permitida a visualização dos mesmos por meio de gráficos através do software interativo de alta performance, MATLAB, presente dentro do *ThingSpeak*. Outra limitação, trata-se do tempo entre upload de dados, o qual deve ser de, no mínimo, 15 segundos.

Para enviar dados ao *ThingSpeak*, faz-se uma requisição HTTP ao servidor do *ThingSpeak*. Uma requisição HTTP nada mais é que uma *string* (que contém as informações da requisição HTTP) enviada via socket TCP client (quem deseja fazer a requisição) a um socket TCP server (servidor que receberá a requisição, no caso o servidor do *ThingSpeak*) através da porta 80. Outra informação importante é que, na requisição HTTP, constará uma chave de escrita para o canal *ThingSpeak*. Trata-se de uma *string* (única para cada canal), necessária para o *ThingSpeak* "saber" para qual canal direcionar os dados enviados (BERTOLETI, 2016).

2.8 Síntese do Capítulo 2

A IoT está presente em distintas aplicações, entre elas: automação residencial, meio ambiente, cidades inteligentes, agricultura, saúde, indústria. A integração de sensores e atuadores nos mais diversos equipamentos utilizados no quotidiano humano visa à solução de problemas de forma prática. Aliando o embasamento teórico de climatização do *DC*, os protocolos de comunicação HTTP, TCP/IP, a versatilidade do microcontrolador ESP8266 e a simplicidade dos sensores DHT11 e BMP180 à problemática do monitoramento *in loco* no DC do NTI, percebeu-se a possibilidade e viabilidade do desenvolvimento e execução deste projeto de pesquisa empírica.

3 DATA CENTER

De acordo com Veras (2010), a definição de Data Center é um conjunto integrado de componentes de alta tecnologia que permitem fornecer serviços de infraestrutura de TI de valor agregado. Para Marin (2011) o DC é conhecido como ambiente de missão crítica, responsável por armazenar diversas estruturas e equipamentos, destinados ao processamento, armazenamento e proteção das informações vitais para a sequência dos negócios e, consequentemente, da continuidade das operações de uma organização.

De acordo com Lange (2014) os DCs, que possuem como principal negócio o processamento de dados, é notória a necessidade de climatizar com precisão. Como a climatização corresponde a um percentual em torno de 40% do consumo de energia do DC, muitos gerentes de infraestrutura têm percebido que investir nesta área desde o projeto inicial pode gerar benefícios econômicos significativos durante a operação.

O *DC* necessita de constante refrigeração para manter o bom funcionamento de seus ativos, sendo que algumas áreas necessitam de mais refrigeração do que outras. Os locais onde existe uma alta concentração de *racks* com diversos ativos, possivelmente, haverá mais emissão de calor. Segundo Marin (2016), alguns técnicos utilizam o termo ar-condicionado para *Data Center*, mas o correto seria climatização. Enquanto ar-condicionado normalmente está associado apenas à temperatura, a climatização tem um significado mais amplo, ou seja, envolve temperatura e umidade relativa do ar. Pois os equipamentos de climatização de precisão utilizados em DCs, devem controlar tanto a temperatura quanto a umidade relativa do ar.

A climatização nos DCs são feitas essencialmente por dois tipos de equipamentos de ar-condicionado:

a. CRAH (*Camputer Room Air Handling*): essas unidades utilizam o condicionamento de ar através de uma unidade de resfriamento de água, conhecida como *chiller*. A refrigeração é obtida por um fluxo de ar que sopra por uma serpentina, um *fan coil* com água gelada para transferir o calor do refrigerante (água) para o ambiente a ser climatizado;

b. CRAC (*Computer Room Air Conditioner*): segundo Marin (2016), a unidade CRAC é parecida com os equipamentos de ar-condicionado que utilizamos em

residências, isso significa que o compressor e o evaporador utilizados no ciclo de climatização são montados dentro da unidade, o fluido refrigerante que entra frio na unidade CRAC absorve o calor e o transporta do ambiente interno (sala de computadores) para o ambiente externo, onde fica a condensadora. O calor retirado é dissipado no ambiente externo.

3.1 Sistema de Refrigeração

A temperatura e a umidade no ambiente de um DC devem ser constantemente controladas para de assegurar o bom desempenho e a integridade operacional dos sistemas internos. Empresas que operam DCs devem atentar a todos os detalhes de infraestrutura para garantir a energia e a conexão com a internet, além de evitar paralisações e atenuar riscos, pois geralmente, grandes incidentes são resultado de pequenas falhas.

A NBR 14565 (2013) e TIA 942-A (2012) indicam que a temperatura na entrada de ar dos equipamentos críticos de TI deve estar entre 18°C e 27°C, com uma umidade relativa do ar de 60%. Segundo o *Data Center Knowledge* a Microsoft conseguiu economizar 250 mil dólares anuais elevando de 2 a 4 graus a temperatura de sua instalação no vale do silício.

Algumas instituições, preocupadas com a sustentabilidade e rendimento dos DC, desenvolvem recomendações para a refrigeração que ajudam no aumento da disponibilidade das operações e não violam as especificações de garantia dos equipamentos instalados. Uma delas é a ASHRAE (*American Society of Heating, Refrigerating and Air-Conditioning Engineers*), que, em 2008, alterou a faixa de temperatura de 20°C a 25°C para 18°C a 27°C, sem modificar os requerimentos de garantia. Temperaturas mais elevadas em DCs correspondem a gastos menores em sistemas de refrigeração e consequente economia de energia.

A Classe 1 da ASHRAE (*American Society of Heating, Refrigerating and Air-Conditioning Engineers*), uma entidade norte-americana internacional na área de padronização para climatização, demonstra na Tabela 6 as informações permitidas e recomendadas de temperatura para a entrada de ar nos equipamentos e também a umidade relativa.

Tabela 6 – Especificações de temperatura e umidade relativa

ESPECIFICAÇÕES DO AMBIENTE				
Classe	Temperatura permitida (°C)	Temperatura recomendada (°C)	% umidade relativa permitida	% umidade relativa recomendada
1	15 até 32.2	20 até 25	20-80	40-55

Fonte: ASHRAE (2018).

Segundo Veras (2009), uma alternativa para melhorar a refrigeração do DC é utilizar fileiras de frente para outra fileira. O ar frio será fornecido pela frente do *rack* através de aberturas no piso elevado. O corredor ventilado é conhecido como corredor frio. O ar frio é atraído através dos *racks* pelas ventoinhas dos servidores e expulso de volta para o corredor quente. O ar quente ascendente a partir deste corredor encontra o seu caminho de volta para a unidade de ar condicionado a ser refrigerado e, em seguida, repetir o ciclo.

3.2 Eficiência energética

Um aspecto relevante em projetos é sempre considerar a potência real consumida nos projetos para cálculo da carga de TI. Um servidor com 1600W de fonte de alimentação não consome esse valor. A plena carga consumirá em torno de 67% do valor nominal da fonte. Sem esse ajuste, pode-se incorrer no erro de projetar um datacenter pelo valor nominal da fonte, superdimensionando os equipamentos de infraestrutura (VERAS, 2015).

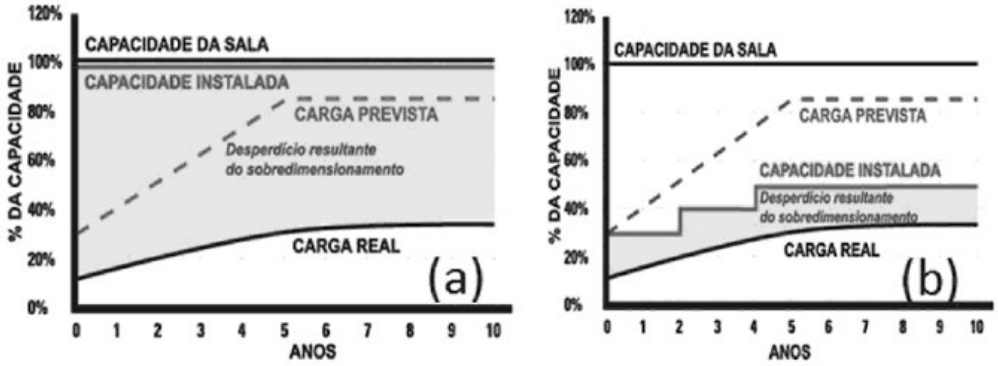
O artigo publicado na revista Dell *Power Solutions* (2007), apresenta um estudo, feito pela empresa Dell, sobre o consumo efetivo em DC projetados há alguns anos. Verificou-se que, de 100% da energia injetada no DC, só 41% efetivamente foi utilizada pela carga de TI, 28% foi consumida pelos equipamentos que transformam e adequam a energia que chega à carga de TI e 31% foi consumida pelos equipamentos de suporte para o arrefecimento.

O artigo reforça a ideia discrepante que há em alguns DC construídos em torno do máximo da capacidade nominal, que não é consumido pela carga de TI. Ou seja, são DC superdimensionados que estão consumindo mais, gastando mais energia, gerando custos de capital e manutenção em excesso. Caso fossem dimensionados de forma coerente poderiam apresentar um percentual alto de economia.

A maioria das ações de redução do consumo de energia passa por medidas na gestão da TI, tais como: remoção de servidores inativos, aumento da temperatura e a utilização de um sistema de monitoramento mais abrangente. Tais ações poderiam ocasionar impacto significativo na eficiência energética e sem precisar de muito investimento (AHLERT, 2017).

A APC by Schneider Electric, fornecedor líder em soluções de infraestrutura física para todo o DC, sugere como boa prática: a construção do DC de forma escalonável. A Figura 13 ilustra a diferença entre os projetos. No caso (a), mostra o jeito antigo de fazer projeto, no qual a capacidade instalada é a própria capacidade da sala. Já no caso (b), o objetivo é que a capacidade instalada aumente em incrementos de acordo com o aumento da demanda de TI (VERAS, 2011).

Figura 13 – Design da capacidade de energia típico (a) e sugerido (b)



Fonte: VERAS (2011)- APC.

Ainda segundo a Veras (2011), uma arquitetura ideal, do ponto de vista da eficiência energética, deveria utilizar os seguintes princípios:

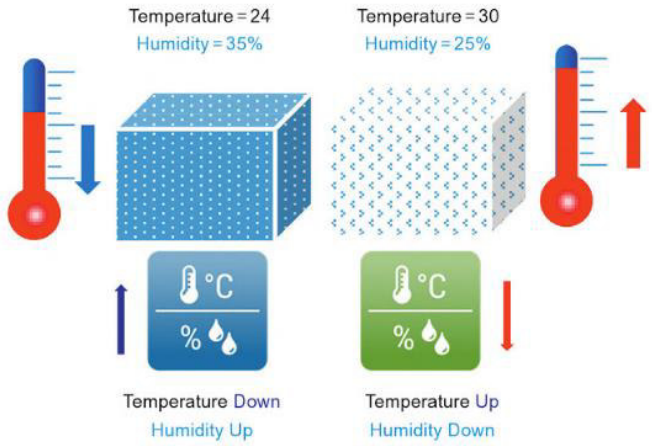
a) O superdimensionamento deve ser reduzido sempre que possível para que os equipamentos possam operar o mais próximo possível da curva ideal de eficiência;
b) Equipamentos de energia e refrigeração que não são necessários no momento, não devem ser energizados;
c) Equipamentos de energia, refrigeração e iluminação devem aproveitar as novas tecnologias para reduzir o consumo de energia;

d) Subsistemas que devem ser utilizados abaixo da capacidade nominal (devido à redundância) devem ser otimizados para aquela fração de carga e não para a eficiência com carga plena.

3.3 Relação entre umidade e temperatura em um DC

Quando o ar do DC fica mais quente, a quantidade de umidade que é retirada no ar é maior. Dessa forma a umidade relativa cairá. Ou seja, para manter a mesma quantidade de umidade, tem-se que adicionar água ao ar caso a temperatura aumente. Essa relação entre umidade e temperatura pode ser observada na Figura 14.

Figura 14 – Relação entre umidade e temperatura



Fonte: WU (2015)

3.3.1 Umidade

A umidade é uma medida da quantidade relativa de vapor d'água presente no ar ou em um gás.

3.3.1.1 Umidade relativa

De acordo com a Equação (1), a umidade relativa corresponde à porcentagem de vapor d'agua por peso presente em um determinado volume de ar ou gás em relação ao peso do vapor d'água existente no mesmo volume de ar ou de gás saturado com vapor d'água, considerando as mesmas temperatura e pressão.

$$Umidade\ Relativa(\phi) = \frac{\substack{quantidade\ de\ vapor\ d'água existente\ em \\ um\ dado\ volume\ de\ ar\ ou\ gás}}{\substack{máxima\ quantidade\ de\ vapor\ d'água ou\ solúvel \\ no\ mesmo\ volume\ de\ ar\ ou\ gás}} \times 100 \qquad (1)$$

3.3.2 Ponto de Orvalho

Ponto de orvalho (*Dew-Point*) é o valor de temperatura onde o ar não pode manter toda a umidade no ar, ou seja, designa a temperatura à qual o vapor de água presente no ar ambiente passa ao estado líquido na forma de pequenas gotas por via da condensação, o chamado orvalho.

Existem duas maneiras de alcançar o ponto de orvalho:

a) Diminuir a temperatura enquanto a umidade ou a umidade relativa é mantida a mesma;
b) Aumentar a umidade do ar enquanto a temperatura é mantida a mesma.

Em regiões mais úmidas ou que estejam mais afetadas pela umidade, o regime de chuvas tende a ser maior, como é o caso da aplicação do projeto, pois a saturação do ar que provoca a condensação é mais frequente.

A Figura 15 mostra que para uma temperatura de 30°C e umidade relativa do ar de 60%, tem-se que o ponto de orvalho é em 20,5°C.

Figura 15 – Ponto de Orvalho

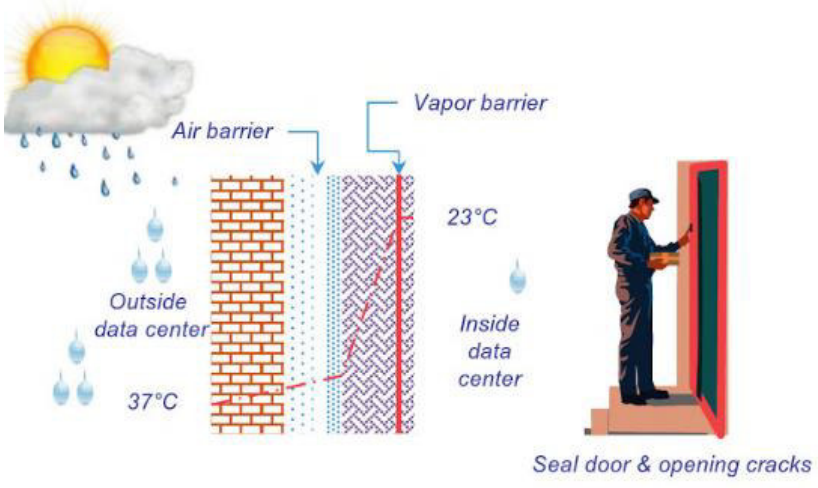
Umidade relativa do ar (%) \ Temperatura do ar (°C)	-5	0	5	10	15	20	25	30	35	40
90	-6,5	-1,0	-3,5	8,5	13,5	18,5	23,5	28,0	33,0	38,5
85	-7,5	-2,0	-2,5	7,5	12,5	17,5	22,5	27,0	32,0	37,5
80	-8,0	-3,0	2,0	6,5	11,5	16,5	21,0	26,0	31,0	36,0
75	-8,5	-3,5	1,0	5,5	10,5	15,5	20,0	25,0	30,0	35,0
70	-9,5	-4,5	0,0	4,5	9,0	14,5	19,0	23,5	28,0	33,5
65	10,0	-5,5	-1,0	3,0	8,0	13,0	17,5	22,0	27,0	32,0
60	-11,0	-6,5	-2,0	2,0	7,0	12,0	16,5	20,5	25,5	30,5
55	-11,5	-7,5	-3,0	1,0	5,5	10,5	15,0	19,5	24,0	29,0
50	-13,0	-8,5	-4,5	-0,5	4,0	9,0	13,5	18,0	22,5	27,0
45	-14,5	-9,5	-6,0	-1,5	2,5	7,0	12,0	16,0	20,5	25,5
40	-16,0	-11,0	-7,5	-3,5	1,0	5,5	9,5	14,0	18,0	23,0
35	-18,0	-12,0	-8,5	-5,0	-1,0	3,0	7,5	12,0	16,5	21,0
30	-19,0	-14,5	-10,5	-7,0	-3,0	1,5	5,5	9,5	13,5	18,0

Fonte: PERFORTEX (2015)

3.4 Controle de temperatura e umidade

Fora do DC, a temperatura e umidade oscilam com as diferentes estações do ano, hora do dia e padrões climáticos. Uma maneira eficaz de controlar a umidade e temperatura externa e interna é isolar o impacto externo nas condições internas do DC e adotar barreiras de vapor para controlar o ganho e a perda de umidade, conforme Figura 16.

Figura 16 – Medidas para evitar interferências do meio externo no DC



Fonte: WU (2015).

Dentro do DC a umidade relativa deve ser restrita dentro da chamada “banda morta”, que está no intervalo de 35% a 45%. O benefício de ter umidade operacional nessa faixa é que

o umidificador não precisa funcionar. Caso a umidade esteja acima de 45% ou abaixo de 35%, haverá desperdício de energia e de água.

Os efeitos da umidade sobre o clima são sentidos tanto nas temperaturas quanto no regime de chuvas. A água, em razão de seu calor específico, tende a conservar por mais tempo as temperaturas, fazendo com que haja uma menor variação delas, ou seja, a amplitude térmica (diferença entre a maior e a menor temperatura) é menor quanto maior for a umidade do ar.

3.5 Rede Nacional de Ensino e Pesquisa

A Rede Nacional de Ensino e Pesquisa (RNP), criada em 1989, é uma organização social ligada ao Ministério de Ciência e Tecnologia do Governo Federal brasileiro, responsável pelo *backbone*[5] da rede acadêmica brasileira.

3.5.1 Rede Ipê

Operada pela RNP, a rede Ipê é uma infraestrutura de rede Internet dedicada à comunidade brasileira de ensino superior e pesquisa, que interconecta universidades e seus hospitais, institutos de pesquisa e instituições culturais.

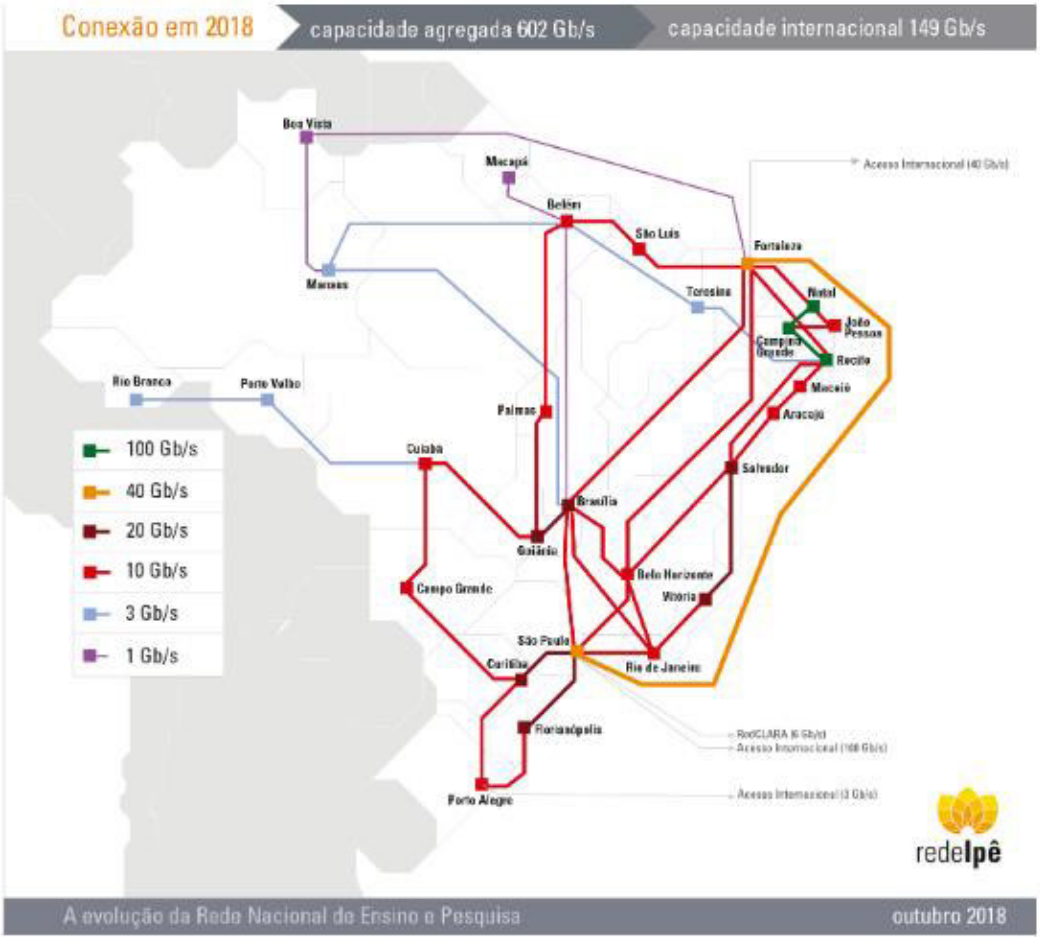
A infraestrutura da rede Ipê engloba 27 Pontos de Presença (PoPs), um em cada unidade da federação, além de ramificações para atender **1522** campi e unidades de instituições de ensino, pesquisa e saúde em todo o país, beneficiando mais de **3,5 milhões de usuários.**

Inaugurada em 2005, como evolução da rede inicialmente implantada em 1992, foi a primeira rede óptica nacional acadêmica a entrar em operação na América Latina.

[5] Backbone significa “espinha dorsal”, e é o termo utilizado para identificar a rede principal pela qual os dados de todos os clientes da Internet passam. É a espinha dorsal da Internet. Esta rede também é a responsável por enviar e receber dados entre as cidades brasileiras ou para países de fora. (Fonte: Site TecMundo)

A Figura 17 ilustra o *backbone* da RNP atualizado.

Figura 17 – Rede Ipê



Fonte: Site da RNP (2018)

Seu *backbone* é projetado para garantir não só a velocidade necessária ao tráfego de internet de aplicações básicas (navegação web, correio eletrônico e transferência de arquivos), mas também ao tráfego de serviços, aplicações avançadas e projetos científicos, e à experimentação de novas tecnologias, serviços e aplicações.

Em 2010, a rede Ipê passou por um grande salto qualitativo, atingindo a capacidade agregada de 233,2 Gbps, um aumento de 280% em relação à capacidade agregada anterior. Nesta nova rede, que é a sexta geração do *backbone* operado pela RNP, as velocidades multigigabits (acima de 1 Gbps) estão disponíveis para 24 dos 27 PoPs.

A ampliação foi resultado de acordo de cooperação com a empresa de telecomunicações Oi, que proverá à NP infraestrutura de transmissão em fibras ópticas para uso não comercial e participará de projetos de Pesquisa & Desenvolvimento (P&D) de interesse comum.[6]

3.6 Data Center NTI- UFMA

O NTI exerce um papel fundamental na instituição DC da UFMA exerce um papel fundamental na instituição: promover soluções criativas e inovadoras em TIC visando ao desenvolvimento da Universidade Federal do Maranhão.

Recentemente, o DC do NTI- UFMA abrigou o Ponto de Presença do Maranhão (PoP-MA), que tem por missão prover integração global e colaboração apoiada em tecnologias de informação e comunicação para a geração do conhecimento e a excelência da educação e da pesquisa.[7]

Algumas instituições no Maranhão já integram a rede primária: A Universidade Federal do Maranhão; os Institutos Federais do Maranhão, Campus: São Luís e Codó; e instituição secundária, tem-se a Universidade Estadual do Maranhão.

O DC do NTI da UFMA apresenta uma estrutura, relativamente, simples; se comparado a DCs mais sofisticados. Utilizam-se três equipamentos de ar-condicionado para realizar o arrefecimento do ambiente, não há desumidificador no local, o que ainda representa um problema. O que agrava o problema é o fato de localizar-se em uma área que apresenta altos níveis de umidade.

A Figura 18 mostra a sala de equipamentos do DC do NTI- UFMA, que é o espaço físico onde estão alocados equipamentos de rede, como: *switches*, roteadores, unidades de *backup*, controladora *wireless*, *storage*, *core switch*, servidores *blade* e de *rack*.

---

[6] Site da RPN: https://memoria.rnp.br/backbone/
[7] Portal UFMA- *http://portais.ufma.br/PortalUfma/paginas/noticias/noticia.jsf?id=49056*

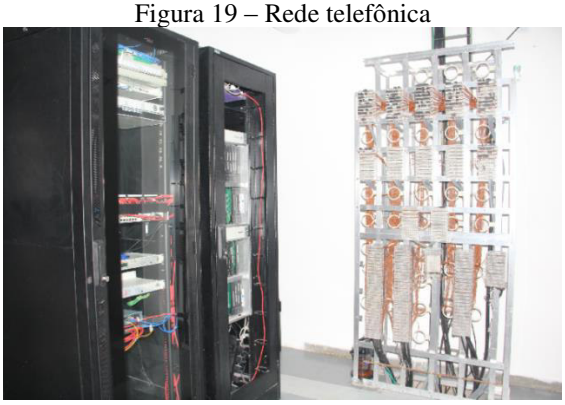
Serviços fundamentais para o bom funcionamento da instituição dependem desse ambiente para seu perfeito funcionamento. Alguns desses serviços como: sistemas acadêmicos, sistemas administrativos, e-mail, além de outras aplicações, são processados nesse sistema.

Figura 18 – Servidores

Fonte: Portal UFMA NTI, (2018).

A comunicação é vital para o funcionando da Universidade. Desse modo, a UFMA possui um sistema de telefonia e rede em fibra ótica a fim de facilitar a troca de informações dentro e fora da comunidade acadêmica, possibilitando assim, aos alunos, professores e funcionários a se conectarem através da banda larga ou Wi-Fi em qualquer lugar do Campus da Universidade. A Figura 19 mostra a rede telefônica da UFMA.

Figura 19 – Rede telefônica



Fonte: Portal UFMA NTI, (2018).

A sala de equipamentos ainda consta com dois quadros elétricos, onde estão ligados os circuitos elétricos que atendem ao sistema de refrigeração e aos *racks* de telecomunicações.

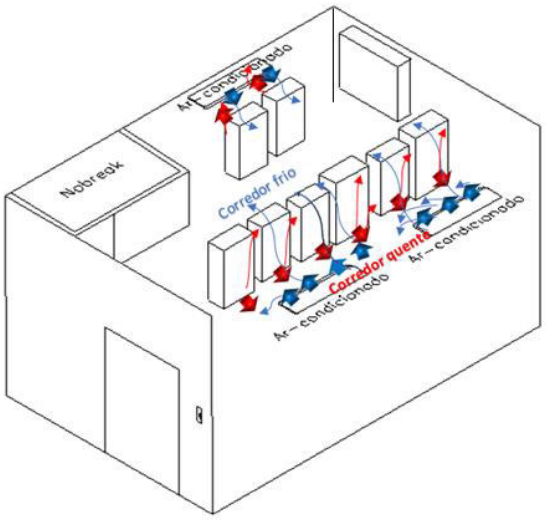
3.6.2 Sistema de refrigeração

Atualmente, os servidores necessitam, incondicionalmente, de um sistema de climatização. Pois, atingem temperaturas elevadas quando submetidos ao uso contínuo, o que ocasiona um aquecimento excessivo de seus componentes internos e a consequente desativação, por um sistema de proteção. (MOSSÉ et al., 2009).

O DC do NTI-UFMA possui um sistema de climatização denominado de sala controlada. O qual tem por objetivo climatizar todo o ambiente que se encontra. Para isso, faz-se uso de três ares-condicionados, sendo dois de 36 mil BTUs e um de 30 mil BTUs do tipo piso/teto.

Percebe-se pela Figura 20 que existem dois corredores térmicos no DC: corredor quente e corredor frio. No corredor quente concentra-se o ar quente que sai dos equipamentos e o corredor frio é composto pelo ar frio que é cedido pelo ar-condicionado. Sendo o ar quente mais leve, ele acaba subindo e como o ar frio é mais denso, ele acaba descendo. Dessa forma é feita, basicamente, a troca de calor no DC.

Figura 20 – Representação do Data Center do NTI- UFMA

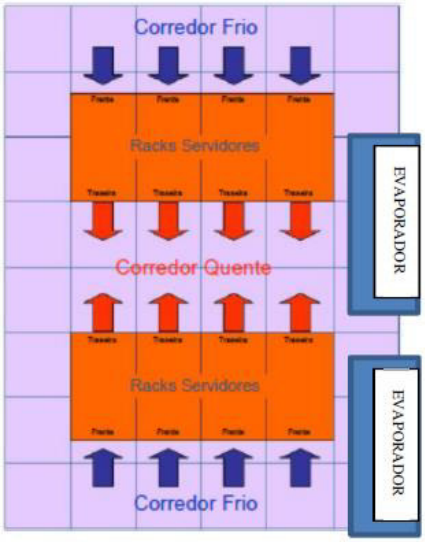


Fonte: Próprio autor (2018).

O princípio básico de climatização por sala controlada é a utilização de corredores quentes e frios, que são criados pelo fluxo de ar de alta e de baixa temperatura. O corredor frio é criado pelo insuflamento do ar, pelo piso ou pelo teto, que vem da umidade evaporada. Já o

corredor quente é criado pelo ar que atravessa os servidores e se esquenta. Esse ar aquecido é succionado pela umidade evaporadora, que faz novamente o seu tratamento (SANTOS, 2011). A Figura 21 ilustra o layout da sala controlada.

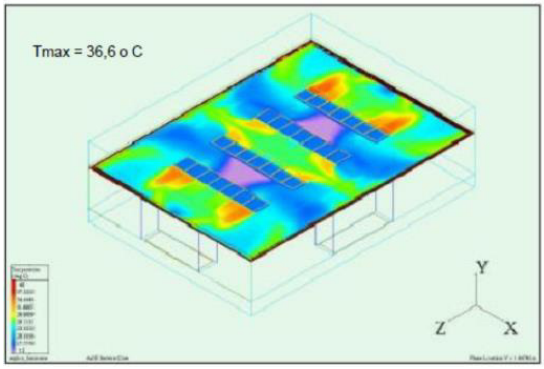
Figura 21 – Layout de climatização da sala controlada



Fonte: MOSSÉ et al (2009).

Ainda segundo Santos (2011). Esse tipo de solução, mesmo quando utilizada em sobrecapacidade, não consegue suprir as altas temperaturas localizadas quando os equipamentos são submetidos a uma alta demanda. Muitas vezes os fluxos irregulares de ar frio acabam resfriando menos uma área que outra, gerando *hot spots* nas salas de Data Center, conforme ilustrado na Figura 22.

Figura 22 – Hot Spots em Data Center- Climatização por sala controlada



Fonte: SHARMA (2003).

A Figura 23 mostra um dos ares-condicionados utilizados no NTI, conforme se pode observar, são modelos comuns; com consumo de energia tipo B, com potência em torno de 24000 a 36000 BTUs.

Figura 23 – Ar-condicionado do Data Center

Fonte: Próprio autor (2018).

3.6.3 Sala de energia ou sala de nobreak

Nessa sala estão alocadas as UPS que atendem o DC. Esse é o ambiente responsável em gerar a energia estabilizada para todos os equipamentos utilizados no DC.

3.7 Síntese do Capítulo 3

É vital para qualquer DC que seus níveis de temperatura e umidade estão na faixa permitida, isso pode economizar milhares ou mesmo milhões por ano, dependendo do tamanho do DC. Uma recente matéria do The Green Grid (*Updated Air-Side Free Cooling Maps: The Impact of ASHRAE 2011 Allowable Ranges*) discute a nova ASHRAE e as faixas recomendadas e permissíveis no contexto da refrigeração livre.

Manter a faixa de temperatura entre 20 ° a 24 ° C é ideal para a confiabilidade do sistema. Esta faixa de temperatura fornece um colchão seguro para o equipamento operar em caso de falha do ar condicionado ou equipamento HVAC, enquanto é mais fácil manter um nível de umidade relativa seguro. Em geral, ETI não deve operar em um DC, onde a temperatura ambiente excedeu 30 ° C. Manter os níveis de umidade relativa entre 45% e 55% é recomendado.

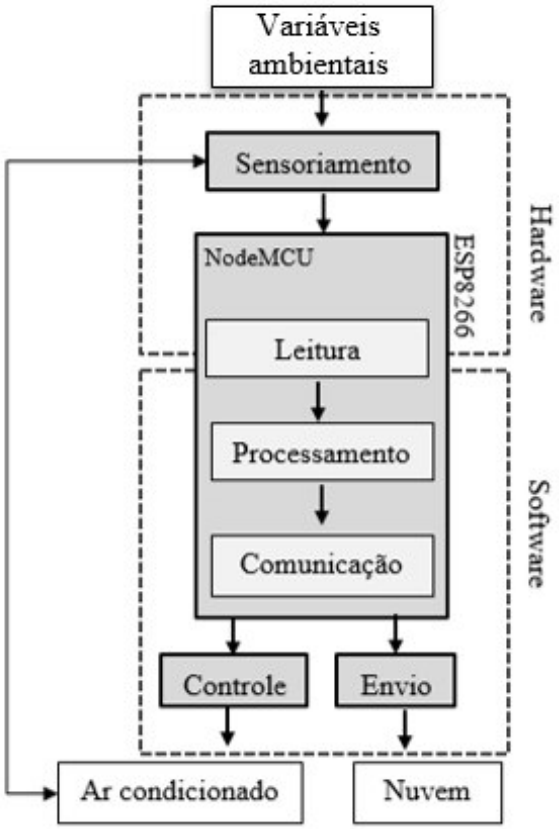
4 IMPLEMENTAÇÃO DO PROJETO

Este projeto propõe um protótipo de baixo custo baseado em *IoT* para a aferição climática das variáveis temperatura, umidade e pressão em conjunto com um controle de temperatura automatizado. A modelagem de *hardware* consiste em um circuito para aquisição dos sinais analógicos e a de *software*, no processamento digital dos sinais e envio das informações via módulo *Wi-Fi*.

4.1 Arquitetura proposta

A arquitetura do protótipo proposto e implementado neste trabalho de monografia engloba duas partes: hardware e software, conforme ilustra o diagrama em bloco do sistema mostrado na Figura 24. Na primeira parte (destacada na parte de cima do diagrama) tem- se a sensibilização do ambiente através dos sensores DHT11 e BMP180, responsáveis por monitorar as variáveis temperatura, umidade e pressão.

Figura 24 – Arquitetura do protótipo



Fonte: Próprio autor (2018).

Conforme pode ser observado na Figura 24, a arquitetura é composta por várias partes:

a) Variáveis ambientais: são todas as variáveis ambientais: temperatura, umidade, pressão, etc.;
b) Sensoriamento: esta etapa é formada pelos sensores DHT11 e BMP180;
c) A plataforma de desenvolvimento NodeMcu apresenta parte de hardware e software. Essa plataforma, em conjunto com a IDE do Arduino, é responsável pela leitura, processamento e comunicação com o servidor;
d) O protótipo é dividido em termos de controle e monitoramento. A parte de monitoramento envia os dados para o servidor *ThingSpeak*. Já a parte de controle, baseado nos dados aferidos, envia comando ao ar-condicionado do DC para aumentar ou diminuir a temperatura dependendo da faixa em que se encontre.

4.2 Orçamento

A Tabela 7 mostra os componentes que foram necessários para a montagem do protótipo de monitoramento e controle. Com exceção do DHT11, jumpers e resistor, todos os componentes foram adquiridos via Ali Express[8] e Mercado Livre[9].

Tabela 7 – Orçamento do protótipo

Componentes	Quantidade	Preço
ESP8266	1	16,88
DHT11	1	13,00
BMP180	1	14,00
Jumpers	10	2,02
LED IR emissor	1	0,19
Resistores	3	0,50
TL1838	1	1,90
Total		R$ 48,49

Fonte: Dados baseados nos preços da AliExpress e Mercado Livre, (dez. 2018)

[8] Disponível em: <https://pt.aliexpress.com/> . Acesso em nov. 2018.
[9] Disponível em: <https://www.mercadolivre.com.br/>. Acesso nov. de 2018.

Percebe-se pela Tabela 7 que os preços dos materiais utilizados no protótipo apresentam um preço acessível. Sendo que alguns microcontroladores existentes no mercado custam bem mais caro que o protótipo final proposto neste projeto.

Um diferencial da Plataforma de desenvolvimento NodeMCU é o módulo Wi-Fi que já é embutido no SoC, a capacidade da arquitetura, 32bits, a frequência de 80MHz, que, consequentemente, isso acarreta um custo benefício favorável em relação aos outros microcontroladores, conforme Tabela 8.

Tabela 8 – Comparação entre alguns microcontroladores

	ESP32	ESP8266	ARDUINO UNO R3
Cores	2	1	1
Arquitetura	32 bits	32 bits	8 bits
Clock	160MHz	80MHz	16MHz
WiFi	Sim	Sim	Não
Bluetooth	Sim	Não	Não
RAM	512KB	160KB	2KB
FLASH	16Mb	16Mb	32KB
GPIO	36	17	14
Interfaces	SPI / I2C / UART / I2S / CAN	SPI / I2C / UART / I2S	SPI / I2C / UART
ADC	18	1	6
DAC	2	0	0

Fonte: Fernando K. Tecnologia (2018) [10].

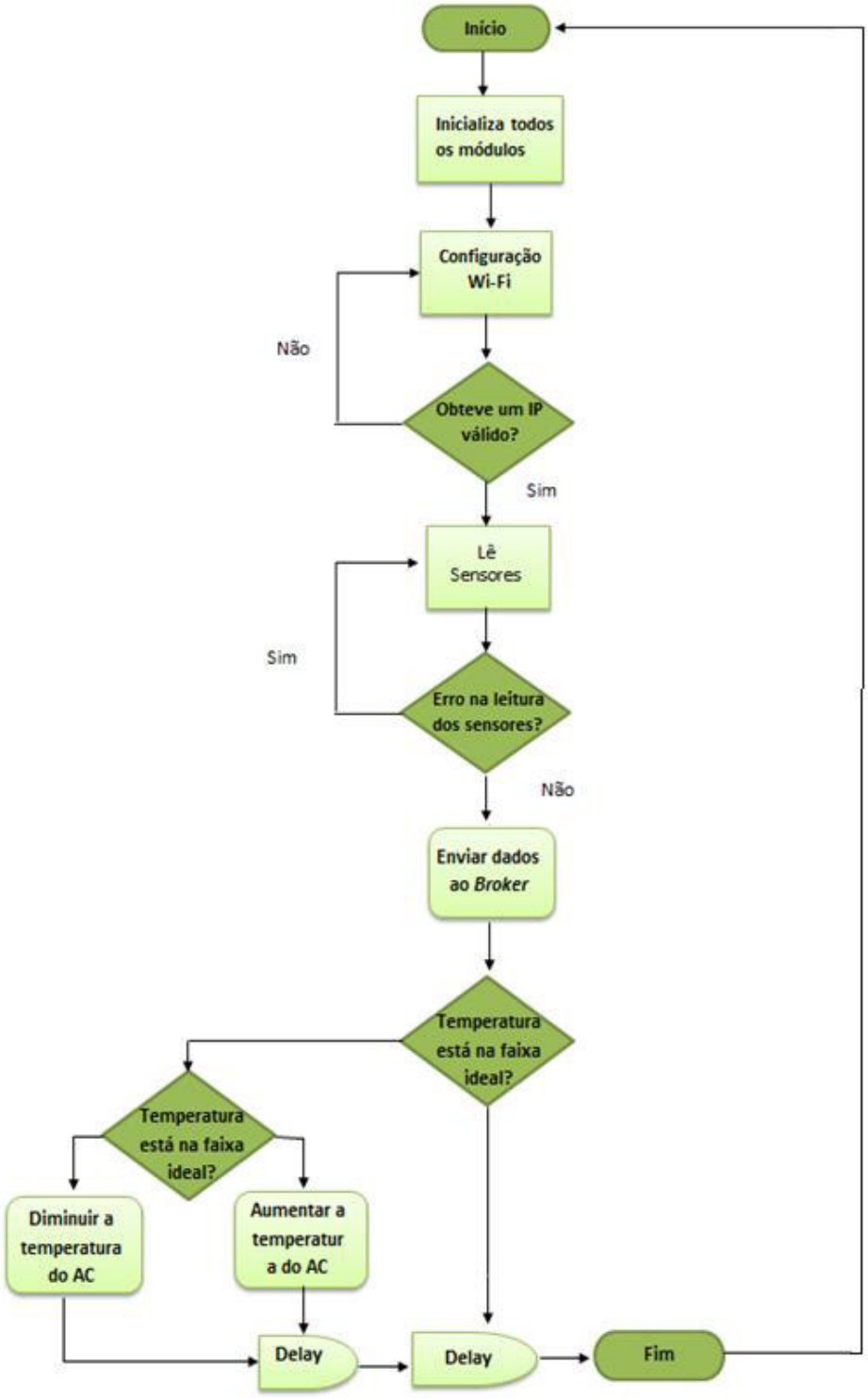
4.3 Montagem do modelo proposto

O sistema proposto foi construído em etapas: Primeiramente inicializaram-se os módulos, fez-se o teste de conexão via Wi-Fi. Em seguida, coletaram-se os dados dos sensores, que após os testes de aquisição de temperatura, umidade e pressão, foram enviados ao servidor via protocolo HTTP para transmiti-los à Internet. Nesta etapa de desenvolvimento do projeto foram utilizados apenas dois tipos de sensores, o DHT11 e o BMP180.

[10] Disponível em:< https://www.fernandok.com/2017/11/introducao-ao-esp32.html>. Acesso em out. 2018.

A Figura 25 mostra o diagrama de blocos do sistema proposto.

Figura 25– Diagrama de Blocos do modelo proposto

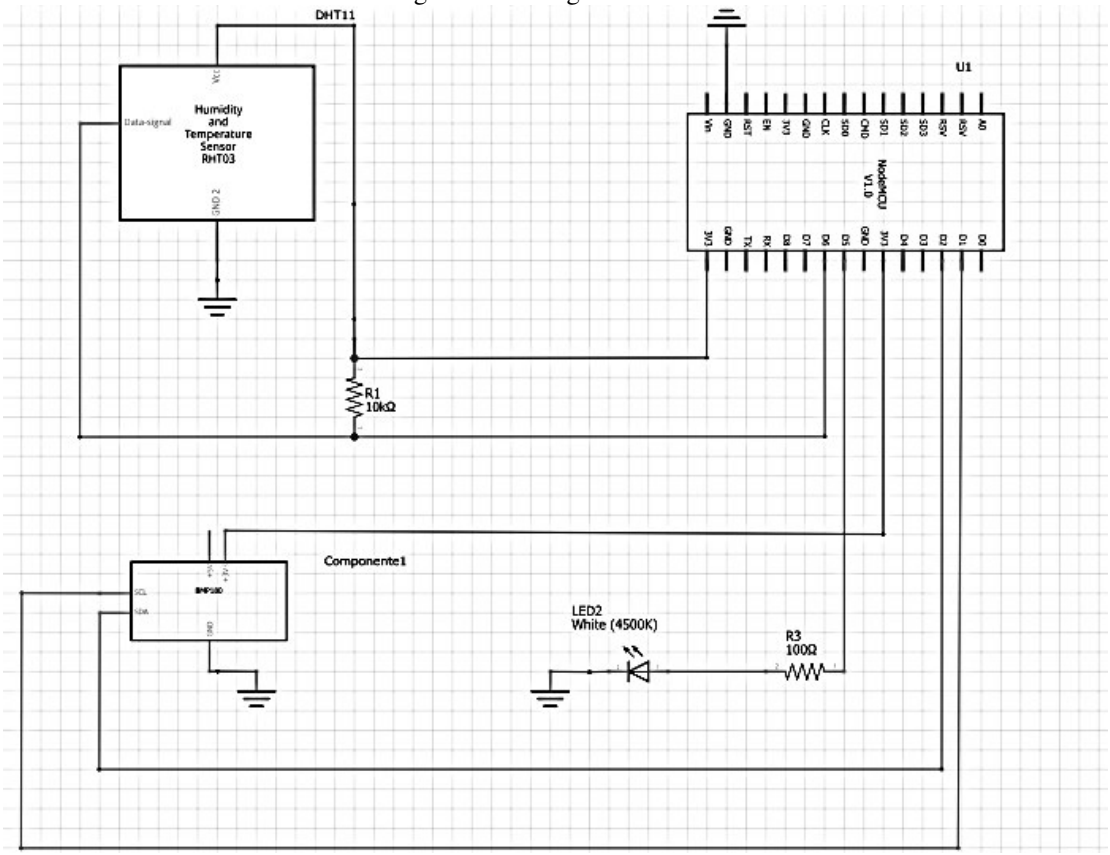


Fonte: Próprio autor (2018).

Primeiramente foi montado o circuito que vai clonar o botão *On/Off* do AC e os botões das temperaturas. Esse circuito é composto pelo receptor infravermelho e o NodeMcu. Para emissão do sinal foi utilizado o emissor infravermelho, em que o pino de saída do receptor foi ligado ao pino digital (D5) do NodeMcu, o pino GND foi conectado ao GND do NodeMcu já o pino VCC do receptor foi conectado ao 3V3 do NodeMcu.

O circuito do protótipo é mostrado na Figura 26.

Figura 26 – Diagrama elétrico



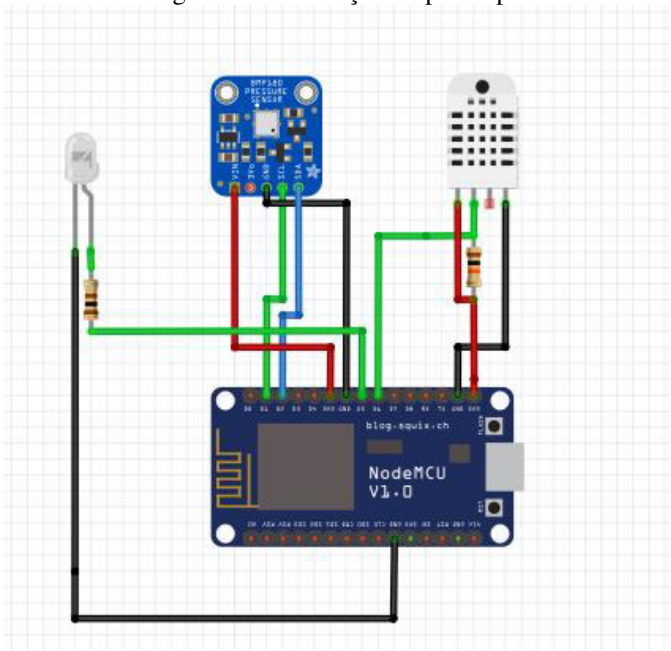
Fonte: Próprio autor (2018).

Todo o trabalho foi feito tendo a placa de desenvolvimento NodeMCU como ferramenta principal. A placa de desenvolvimento conta com um regulador de tensão e um adaptador serial UART-USB. Adicionou-se ao sistema sensores de temperatura umidade e pressão.

O sensor de temperatura e umidade, DHT11, foi alimentado em 3.3V. Colocou-se entre os pinos de dados e alimentação um resistor de 10kΩ para que a corrente não ultrapassasse o limite do sensor. O sensor de pressão e temperatura, BMP180, apresenta dois pinos de dados que foram ligados aos pinos D1 e D2 do NodeMCU. O circuito ilustrado na

A Figura 27 mostra, com a mais clareza, a ligação dos dispositivos ao NodeMcu.

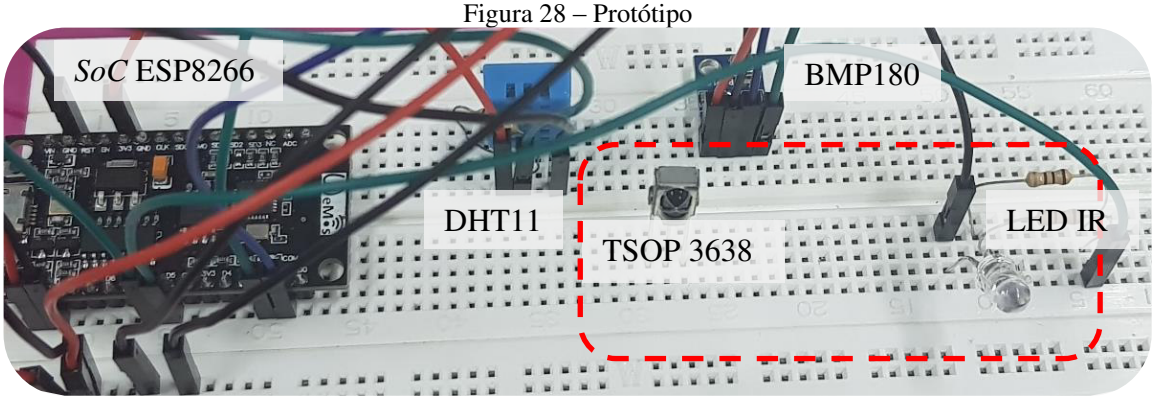
Figura 27 – Ilustração do protótipo



Fonte: Próprio autor (2018).

O processo de montagem foi elaborado em duas etapas, conforme ilustra Figura 28. A primeira consistiu no monitoramento das variáveis: temperatura, umidade e pressão. Os componentes necessários para essa parte foram: o microcontrolador ESP8266, um resistor de 10KΩ os sensores DHT11 e BMP180.

Figura 28 – Protótipo



Fonte: Próprio autor (2018).

A segunda parte, área tracejada na Figura 28, consistiu em controlar a variável temperatura baseado nos dados dos sensores. Para isso foi feita a captação das frequências

emitidas pelo controle remoto através do LED receptor infravermelho, depois essas frequências foram utilizadas no código principal (Apêndice A), responsável pelo controle automatizado propriamente dito.

4.4 Conexão com o ThingSpeak

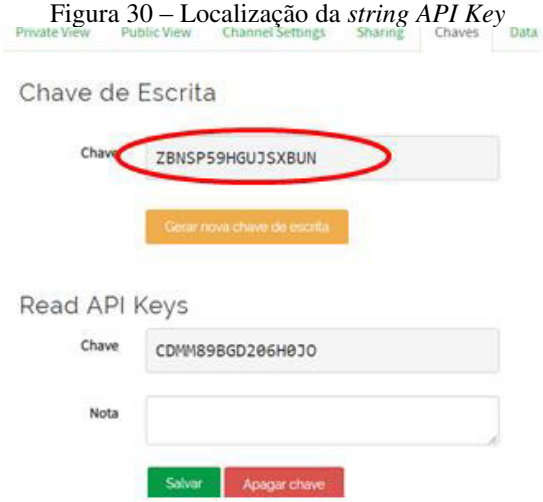
Para enviar dados à nuvem do ThingSpeak, foi necessário criar um canal para receber e exibir os dados coletados pelos sensores DHT11 e BMP180. A Figura 29 mostra a parte do código em que há conexão com o servidor ThingSpeak.

Figura 29 – Criação de um canal *ThingSpeak*

Fonte: ThingSpeak (2018)

A Figura 30 mostra como obter o código a ser utilizado pelo programa para realizar *upload* informações.

Figura 30 – Localização da *string API Key*



Fonte: ThingSpeak (2018)

4.5 Síntese do Capítulo 4

O protótipo proposto para um sistema de medição climática e controle térmico utilizando o ESP8266 foi montado em uma *protoboard*, com sensores DHT11, BMP180 e LEDs infravermelho. O sistema de monitoramento consiste em sensores conectados ao microcontrolador ESP8266 que envia os dados, em intervalos regulares, para a nuvem do servidor *Thingspeak,* através do protocolo HTTP.

Em conjunto, o sistema ainda consta com LEDs infravermelho posicionados próximo ao ar-condicionado, para que caso a temperatura ultrapasse a faixa permitida, seja feita a alteração automaticamente.

O protótipo apresenta um custo razoável. Em relação a outros microcontroladores ele apresenta algumas vantagens. E uma das principais é o módulo Wi-Fi, a arquitetura de 32bits, um processador de 80MHz, podendo chegar até 160MHz.

Com a tecnologia IoT ganhando cada vez mais espaço no mercado atual, diversas empresas tem desenvolvido ferramentas capazes de lidar com essa tecnologia. Uma dessas ferramentas trata-se do aplicativo e API IoT de código aberto para armazenar e recuperar dados de coisas. Neste projeto, optou-se pelo *ThingSpeak* que permite a criação, gratuita, de aplicativos de registro de sensores, conjugada a uma análise matemática com o auxílio de um software interativo de alta performance, *MATLAB*.

5 RESULTADOS E ANÁLISE

Este capítulo tem por objetivo mostrar os testes e uma análise crítica dos resultados alcançados no decorrer do desenvolvimento, averiguando assim se os objetivos foram alcançados.

5.1 Testes

Monitoramento

A Figura 31 mostra a declaração das bibliotecas utilizadas no código. Elas são necessárias para que o programa reconheça as funções a serem utilizadas.

Figura 31 – Declaração das bibliotecas

```
#include <IRremoteESP8266.h>
#include <IRsend.h>
#include <UIPEthernet.h>
#include <Wire.h>
#include <DHT.h>
#include <Adafruit_BMP085.h>
#include <ESP8266WiFi.h>

Adafruit_BMP085 bmp;

String apiKey = "ZBNSP59HGUJSXBUN";
```

Fonte: Próprio autor (2018).

A Figura 32 apresenta a sub-rotina de leitura. A qual irá receber as informações através da porta serial virtual *mySerial*.

Figura 32 – Sub-rotina de leitura

```
void loop() {

      float u = dht.readHumidity();
      float t1 = dht.readTemperature();
      float p = bmp.readPressure();
      float t2 = bmp.readTemperature();
      float temp_base= 23;
      float temp_min= 20;
      float temp_max= 27;
      float umid_min= 40;
      float umid_max= 55;
      float temp=t1+temp_erro;

  //MONITORAMENTO DAS VARIÁVEIS TEMPERATURA, UMIDADE E PRESSÃO
            if (isnan(u) || isnan(t1))
               {
                   Serial.println("Falha na leitura do sensor DHT11!");
                   return;
               }
            if (isnan(p)||isnan(t2))
               {
                   Serial.println("Falha na leitura do sensor BMP180!");
                   return;
               }



                      if (client.connect(server,80))   //
```

Fonte: Próprio autor (2018).

A Figura 33 ilustra a comunicação com o *ThingSpeak* através da Sub-rotina *send Data*- dados enviados ao bando de dados online.

Figura 33 – Conexão com ThingSpeak

```
if (client.connect(server,80))   //
{

    String postStr = apiKey;
    postStr +="&field1=";
    postStr += String(t1);
    postStr +="&field2=";
    postStr += String(u);
    postStr +="&field3=";
    postStr += String(p);
    postStr +="&field4=";
    postStr += String(t2);
    postStr += "\r\n\r\n";

    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(postStr.length());
    client.print("\n\n");
    client.print(postStr);
```

Fonte: Próprio autor (2018).

Controle

Primeiramente, captaram-se as frequências correspondentes a cada temperatura, através do código apresentado no Apêndice A que é um código exemplo da própria biblioteca do ESP8266. Essas frequências foram utilizadas para o circuito de controle, onde foram carregadas no NodeMcu. Um exemplo do teste é mostrado na Figura 34, conforme Apêndice B.

Figura 34 – Frequências captadas pelo LED receptor

Desligar

uint16_t rawData [227] =

{3024, 1642, 438, 1146, 436, 1146, 438, 356, 456, 368, 480, 354, 464, 1146, 436, 354, 464,
406, 436, 1148, 436, 1122, 462, 354, 464, 1146, 438, 352, 464, 356, 432, 1174, 438, 1146,
432, 370, 480, 1122, 462, 1120, 462, 354, 464, 354, 464, 1146, 436, 356, 432, 410, 464,
1144, 436, 354, 464, 356, 488, 378, 438, 354, 464, 356, 486, 340, 478, 354, 434, 392, 482,
354, 464, 354, 432, 392, 484, 352, 464, 354, 434, 390, 484, 352, 464, 354, 492, 380, 436,
354, 464, 356, 456, 368, 480, 1122, 460, 354, 464, 358, 430, 1176, 406, 1176, 406, 390,
484, 354, 464, 354, 432, 390, 484, 354, 464, 356, 430, 1176, 406, 386, 488, 352, 464, 1146,
436, 354, 464, 358, 486, 354, 462, 356, 432, 1172, 410, 390, 482, 1120, 462, 380, 438, 354,
464, 358, 486, 380, 438, 354, 434, 392, 482, 354, 464, 352, 434, 390, 484, 354, 464, 352,
434, 390, 484, 354, 464, 358, 486, 380, 438, 354, 464, 356, 458, 368, 478, 354, 464, 354,
456, 386, 464, 354, 432, 392, 482, 354, 464, 354, 434, 388, 484, 354, 464, 354, 464, 406,
436, 354, 464, 356, 458, 368, 480, 354, 464, 354, 432, 392, 482, 354, 462, 360, 484, 352,
464, 1146, 436, 1146, 436, 354, 464, 354, 434, 392, 482, 1120, 464, 352, 464};

Fonte: Próprio autor (2018)

A Figura 35 ilustra a parte do código responsável por captar as frequências correspondentes a cada comando do botão do controle remoto do ar-condicionado. A Figura 34 mostra o comando desliga e a respectiva frequência.

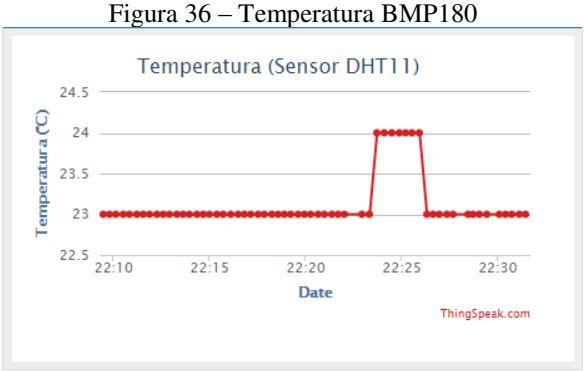
Figura 35 – Código para captar as frequências de comando do controle remoto

```
void loop() {
  // Check if the IR code has been received.
  if (irrecv.decode(&results)) {
    // Display a crude timestamp.
    uint32_t now = millis();
    Serial.printf("Timestamp : %06u.%03u\n", now / 1000, now % 1000);
    if (results.overflow)
      Serial.printf(
          "WARNING: IR code is too big for buffer (>= %d). "
          "This result shouldn't be trusted until this is resolved. "
          "Edit & increase kCaptureBufferSize.\n",
          kCaptureBufferSize);
    // Display the basic output of what we found.
    Serial.print(resultToHumanReadableBasic(&results));
    dumpACInfo(&results);  // Display any extra A/C info if we have it.
    yield();  // Feed the WDT as the text output can take a while to print

    // Display the library version the message was captured with.
    Serial.print("Library   : v");
    Serial.println(_IRREMOTEESP8266_VERSION_);
    Serial.println();

    // Output RAW timing info of the result.
    Serial.println(resultToTimingInfo(&results));
    yield();  // Feed the WDT (again)

    // Output the results as source code
    Serial.println(resultToSourceCode(&results));
    Serial.println("");  // Blank line between entries
    yield();             // Feed the WDT (again)
  }
```

Fonte: Exemplo IRremote- IRrecvDumpV2.

Em seguida, foi implementado o circuito de monitoramento. Os dados foram enviados para o *ThingSpeak* que gerou os gráficos correspondentes a cada aferição feita pelos sensores. A Figura 36 e a Figura 37 mostram o registro de monitoramento de temperatura durante certo período.

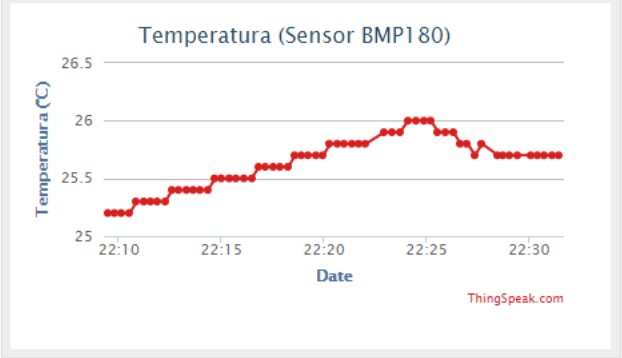
Figura 36 – Temperatura BMP180



Fonte: Próprio autor (2018).

O gráfico é temperatura no tempo onde as mudanças de temperatura são atualizadas após um intervalo de 15 segundos, sendo o primeiro gráfico baseado nos dados coletados pelo sensor DHT11 e o segundo pelo BMP180.

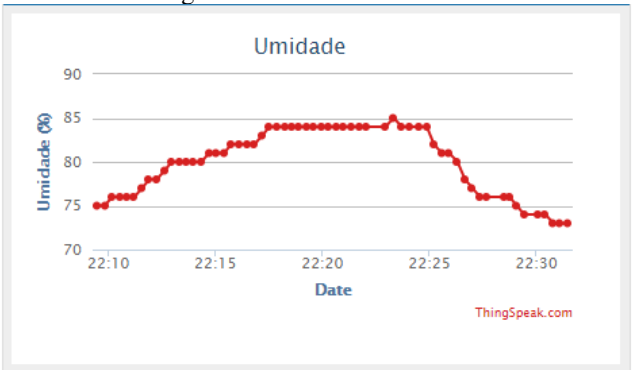
Figura 37 – Temperatura BMP180



Fonte: Próprio autor (2018).

A Figura 38 mostra o registro de monitoramento de umidade em período de tempo. O gráfico é umidade no tempo em que as mudanças na umidade são atualizadas após um intervalo de 10 segundos, de forma semelhante ao monitoramento de temperatura.

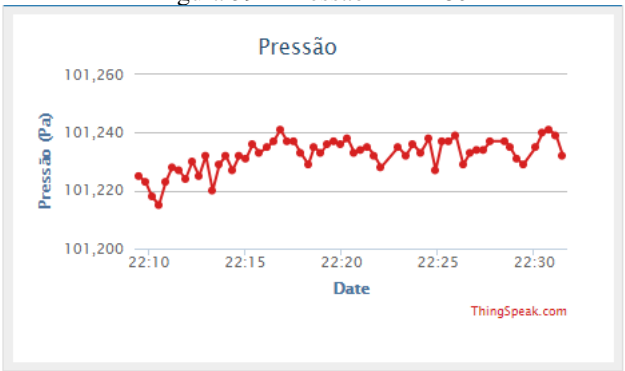
Figura 38 – Umidade DHT11



Fonte: Próprio autor (2018).

A Figura 39 mostra o registro de monitoramento de pressão em período de tempo. O gráfico é pressão no tempo em que as mudanças na umidade são atualizadas após um intervalo de 10 segundos.

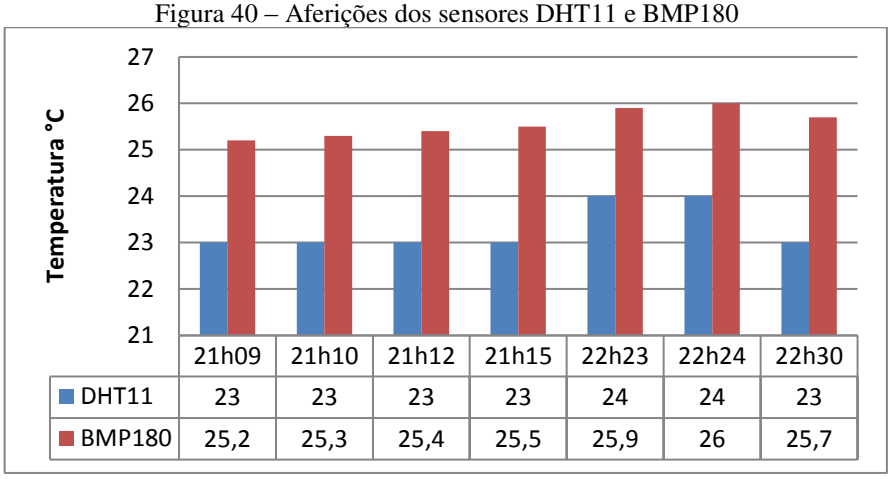
Figura 39 – Pressão BMP180



Fonte: Próprio autor (2018).

As aferições de temperatura dos dois sensores BMP180 e DHT11 foram comparadas, conforme Figura 40. Os valores estão próximos do esperado até por que os sensores têm uma margem de precisão diferenciada. O DHT11 está com uma diferença de aproximadamente 2°C em relação ao BMP180, o que ainda é muito para aplicações mais precisas.

Figura 40 – Aferições dos sensores DHT11 e BMP180



	21h09	21h10	21h12	21h15	22h23	22h24	22h30
DHT11	23	23	23	23	24	24	23
BMP180	25,2	25,3	25,4	25,5	25,9	26	25,7

Fonte: Próprio autor (2018).

Para aplicação especificamente no DC este projeto ainda precisa de melhorias. Entretanto, os resultados em laboratório foram satisfatórios. A questão da segurança é algo que precisa ser mais detalhadamente analisado.

Conforme visto no API *ThingSpeak* não há ainda uma segurança criptográfica, mas nada impede que seja acrescentada, isso acarretaria em melhoria para o protótipo, o que o tornaria mais seguro e também mais viável para uma aplicação envolvendo uma área tão delicada que é a TI.

O controle funcionou de forma satisfatória, mas também exigem melhorias, o ideal seria que a umidade e pressão também fossem variáveis controladas, ainda mais que a área de aplicação apresenta um índice de umidade extremamente elevado. Então, o desumidificador precisa ser acrescentado e, também, controlado para que o protótipo atenda de forma satisfatória a aplicação.

5.2 Dificuldades encontradas

Primeiramente, foi feito o circuito de controle e depois de monitoramento. Ambos funcionaram separadamente. Entretanto, quando os códigos foram unidos, o programa não funcionou por completo nas primeiras tentativas. É como se o ESP8266 compilasse apenas a parte do código de controle ou de monitoramento. A solução foi colocar tempos de varredura diferenciados para cada parte do código.

Em termos de bibliografia, o material referente ao ESP8266 ainda é limitado. O que dificulta a aquisição de certas informações. Outras dificuldades encontradas foram os próprios testes, que foram feitos em laboratório. Às vezes, o LED infravermelho não estava bem posicionado o que, consequentemente, inviabilizava a chegada do sinal ao sensor do ar-condicionado.

O objetivo inicial do protótipo era correlacionar a temperatura e umidade e fazer a programação do código de controle. No entanto, os resultados não foram satisfatórios. O código não funcionou da forma esperada. Então, optou-se por deixar apenas o controle térmico. O ideal seria fazer o controle de temperatura e umidade. No entanto, não foi possível, até por que a aplicação que o projeto está voltado não há desumidificador. Há apenas 3 equipamentos de ar-condicionado, mas isso não impede que este item se torne uma melhoria futura na aplicação.

6 CONCLUSÕES E TRABALHOS FUTUROS

Propôs-se um protótipo de baixo custo utilizando o microcontrolador ESP8266 para o monitoramento de um *Data Center* em tempo real e de forma remota, via Internet, aliado a um sistema de controle térmico para modificar a temperatura em caso de avaria nas medições em relação ao *ranger* pré-estabelecido. Este sistema não apenas agrega valor ao sistema de combate a incêndios existente no local, mas também proporciona maior praticidade e economia, pois não é necessário se deslocar até o local para fazer as aferições.

Os dados podem ser monitorados de qualquer lugar, em qualquer momento, pelo painel *on-line*. O sistema de alerta baseado em SMS acrescenta uma camada adicional de monitoramento mais rigoroso dos parâmetros vitais, auxiliando assim a tomar medidas imediatas se qualquer situação alarmante for relatada. Adicionado a isso, o sistema consta com um controle automatizado da temperatura.

Os testes de medição assinalaram que o protótipo consegue aferir os valores de temperatura, umidade e pressão; e encaminhar os dados à nuvem. Além disso, realizar o controle térmico automático. Para os experimentos iniciais em laboratório, eles apresentaram resultados razoáveis. O código de controle reagiu da forma esperada aos comandos. O que são pontos positivos para o projeto. E analisando por esse ângulo, a ideia do protótipo pode servir como base para trabalhos futuros, podendo até mesmo ser aplicado em um DC. Claro que ainda há muito a ser melhorado, até mesmo modificado, mas já é um começo.

Um diferencial deste projeto, em relação aos trabalhos correlatos, trata-se do uso integral da plataforma de desenvolvimento NodeMcu. Pois, em geral, nos projetos analisados utilizam-se outros microcontroladores como o Arduino, Raspberry e apenas o módulo Wi-Fi do ESP.

Outro fator que diferencia a ideia deste projeto, aos demais, é o controle térmico. No caso da aplicação específica, o controle é essencial. À noite, por exemplo, não há colaboradores, técnicos no local. Em caso de alguma avaria em relação à faixa estabelecida, o controle atuaria. O que pode evitar diversos problemas.

6.1 Sugestões para trabalhos futuros

Para aplicações, especialmente, aqui em São Luís é imprescindível que seja levado em consideração a variável umidade. Pois, localiza-se em uma das regiões mais úmidas do país. Então, torna-se crucial implementar o controle aliado à umidade. Nesse caso, além do ar condicionado, o desumidificador também seria controlado.

O sistema de combate a incêndio do DC é uma questão relevante, pois no NTI-UFMA há apenas alguns extintores. Ou seja, caso haja um princípio de incêndio ter sensores acoplados ao sistema torná-lo-ia mais eficaz, visto que dessa forma seriam emitidos alertas com antecedência, evitando maiores danos.

Realizar um estudo mais aprofundado sobre o microcontrolador ESP8266 e estudar alternativas de implementar um protótipo mais robusto, com maior quantidade de sensores, que consigam sensibilizar o ambiente como um todo, por conseguinte, adquirindo resultados mais precisos.

Em termos de segurança, pode-se adicionar ao protótipo um sistema de criptografia independente. No *ThingSpeak* os dados não são criptografados. A única restrição de acesso aos dados é o *login* e a senha. Mas nada impede que haja um sistema de segurança adicional, fazendo com que o protótipo torne-se mais seguro e viável para uma aplicação em um DC.

REFERÊNCIAS

ABDMEZIEM, Riad; TANDJAOUI, Djamel. Internet of Things: Concept, Building blocks, Applications and Challenges. arXiv preprint arXiv:1401.6877, 2014.

ABNT. Norma ABNT NBR 14565: Cabeamento estruturado para edifícios comerciais e datacenters. 2013.

AHLERT, Edson. Data Centers e Eficiência Energética - Estudo de Caso Univates. Artigo apresentado como Trabalho de Conclusão do Curso de Especialização em Datacenter, 2017. Disponível em: <https://riuni.unisul.br/bitstream/handle/12345/4626/Datacenter%20Unisul%20-%20Edson%20Ahler.pdf?sequence=1&isAllowed=y>. Acesso em novembro de 2018.

ALMEIDA, Rodrigo; MORAES, Carlos; SERAPHIM, Thatyana. Programação de sistemas embarcados: desenvolvendo software para microcontroladores em linguagem C. 1. Ed.- Rio de Janeiro: Elsevier, 2016.

ANSI/TIA-942. Telecommunications Infrastructure- Standard for Data Center, 2005.

AOSONG. Temperature and humidity module: DHT11 Product Manual. DataSheet. China, 2014. Disponível em: <https://img.filipeflop.com/files/download/Datasheet_DHT11.pdf>. Acesso em novembro de 2018.

ASHTON, Kevin. That 'Internet of Things' Thing. RFID Journal, 2009. Disponível em: <https://www.rfidjournal.com/articles/view?4986>. Acesso em outubro de 2018.

BOSCH Invented for life. Data Sheet BMP180 Digital pressure sensor, 2013. Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/514264/TI1/BMP180.html>. Acesso em novembro de 2018

CISCO. Visão Geral do TCP/IP, Atualizado em 2005. Disponível em: < https://www.cisco.com/c/pt_br/support/docs/ip/routing-information-protocol-rip/13769-5.pdf>. Acesso em novembro de 2018.

COMPUTER WORLD. Inovação: IoT vai injetar US$ 11 trilhões na economia mundial em até 2025, 2017. Disponível em: <https://computerworld.com.br/ 2017/05/04/iot-vai-injetar-us-11-trilhoes-na-economia-mundial-em-ate-2025-diz-professor-da-fgv/>. Acesso em dezembro de 2018.

DELL Power Solutions. Data Center Efficiency in the Scalable Enterprise. Dell Inc., 2007. Disponível em: <https://www.dell.com/downloads/global/power/ps1q07-20070210-CoverStory.pdf>. Acesso em dezembro de 2018.

DHT11 GREENGARD, Samuel. The internet of things. Cambridge, MA: The MIT Press, 2015. p. 188-189.

ESP8266EX, Data Sheet. Espressif Systems, 2018- versão 6.0. Disponível em: < https://www.espressif.com/sites/default/files/documentation/0a-sp8266ex_datasheet_en.pdf>. Acesso em novembro de 2018

FRANCISCATTO, Roberto; CRISTO, Fernando de; PERLIN, Tiago. Redes de Computadores. Universidade Federal de Santa Maria, Colégio Agrícola de Frederico Westphalen, 2014.

FOROUZAN, Behrouz. Comunicação de dados e redes de computadores. AMGH Editora S.A. 2008, 4ª edição.

FOROUZAN, Behrouz; MOSHARRAF, Firouz. Redes de Computadores- uma abordagem TOP-DOWN. AMGH Editora S.A. 2003.

HOWER, Mike. As "internet of things" grows, so do e-waste concerns. Sustainable Brands, 2014.

KAUR, A.; JASUJA, A. Health monitoring based on iot using raspberry pi. International Conference on Computing, Communication and Automation, 2017.

KUMAR, C. Kishore; IBRAHIM, Mohammed; SRIKANTH, Neerumalla, ASWIN, S.; PEEYUSH, K.. Internet of things based approach for open precision farming. 2017 International Conference on Advances in Computing, Communications And Informatics (icacci), p.1-6, set. 2017. IEEE.

LANGE, Milena. Entrevista: Milena Lange, especialista em climatização em data centers. Portal Web Ar Condicionado, jun. 2014. Disponível em: <http://www.webarcondicionado.com.br/entrevista-milena-lange-especialista-em-climatizacao-de-data-centers>. Acesso em outubro de 2018.

LIMA, Izabelle. Diferenças entre DHT11 e DHT22. AutoCore Robótica 2018. Disponível em: <http://autocorerobotica.blog.br/diferencas-entre-os-sensores-dht11-e-dht22-2/>. Acesso em dezembro de 2018.

MADEIRA, Daniel. Controle remoto e módulo receptor infravermelho com Arduino.Vida de Silicio, 2018. Disponível em: < https://portal.vidadesilicio.com.br/controle-remoto-e-modulo-receptor-infravermelho/>. Acesso em dezembro de 2018.

MAGRANI, Eduardo. A internet das coisas. Rio de Janeiro: FGV Editora, 2018.

MURTA, Gustavo. O que é o Módulo NodeMCU-ESP12 ? Eletrogate Blog, 2018. Disponível em: <http://blog.eletrogate.com/nodemcu-esp12-introducao-1/>. Acesso em outubro de 2018.

MARIN, Paulo Sérgio. Data Centers – Engenharia: Infraestrutura Física. São Paulo: PM Books, 2016.

MARTINS, Elaine. O que é Backbone? TecMundo, 2009. Disponível em: <https://www.tecmundo.com.br/conexao/1713-o-que-e-backbone->. Acesso em novembro de 2018.

MEIRELLES, Alessandro. 'Estacionamento inteligente' mostra vagas desocupadas por aplicativo. Atualizado novembro de 2014. Disponível em: <http://g1.globo.com/sp/piracicaba-regiao/noticia/2014/11/estacionamento-inteligente-mostra-vagas-desocupadas-por-aplicativo-guasdesaopedro.html>. Acesso em dezembro de 2018.

MINATEL, Pedro. Sistemas Embarcados e Internet das Coisas. Disponível em: <http://pedrominatel.com.br/pt/esp8266/nodemcu-utilizando-pwm/>. Acesso em dezembro de 2018.

MINERA, Francisco. Guia Profissional de Criação de Sites. São Paulo: Digerati Books, 2010. 128 p.- (v.2).

MOSSÉ, D.; Leite, J. C. B.; Barros, A. G. B. Introdução aos clusters verdes de servidores, 2009.

OLAIR, Ricardo Junior. Sistema de monitoramento residencial baseado em Internet das Coisas. 2017. 67 p. Trabalho de Conclusão de Curso em Engenharia Elétrica - Universidade Estadual de Londrina, Londrina.

PORTA, Leonardo dalla. Sensor de Temperatura e Umidade com Jumper – DHT11. Usinainfo, 2014. Disponível em: <http://blog.usinainfo.com.br/sensor-de-temperatura-e-umidade-com-jumper-dht11/>. Acesso em setembro 2018.

PORTAL UFMA. Núcleo de Tecnologia da Informação- NTI. Disponível em: <http://portais.ufma.br/PortalUnidade/nti/paginas/pagina_estatica.jsf?id=906>. Acesso em novembro 2018.

RAY, Partha Pratim. Internet of Things cloud enabled MISSENARD index measurement for indoor occupants. Measurement, p.157-165, out. 2016. Elsevier BV

RNP- Rede Nacional de Ensino e Pesquisa. Rede ipê, 2018. Disponível em: <https://www.rnp.br/servicos/conectividade/rede-ipe>. Acesso em novembro de 2018.

SUNROM. DHT11: Humidity and Temperature Sensor. [S.l.], 2012. Disponível em: <http://robocraft.ru/files/datasheet/DHT11.pdf>. Acesso em outubro de 2018.

SANTOS, J. Andrade. Controle dinâmico de capacidade de refrigeração de gabinetes climatizados em Data Centers. Brasília, nov. 2011.

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

SAHA, S.; MAJUMDAR, A. Temperature and Air Quality Monitoring in Hospitals Helping Quick Evacuation of Patients during fire- A solution with ESP8266 Based Wireless Sensor Network and Cloud Based Dashboard with Real Time Alert System, 2017.

SHARMA. R., Patel C., Bash, C., and Beitelmal, A. ”Thermal Considerations in Cooling Large Scale Compute Density Data Centers”, Itherm Conference, San Diego, California, June 2002, pp. 767 – 776.

SISTO, Tiago M.; CABRAL, Eduardo L.. Sistema de captura de movimento com sensores infravermelho. Escola de Engenharia Mauá, 2014 (EEM/CEUN IMT).

THOMSEN, Adilson. Qual módulo ESP8266 comprar? FILIPEFLOP, 2016. Disponível em: <https://www.filipeflop.com/blog/qual-modulo-esp8266-comprar/>. Acesso em novembro de 2018

TORRES, Andrei B., ROCHA, Atslands R., SOUZA, José N. de. Análise de Desempenho de Brokers MQTTem Sistemas de Baixo Custo. Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)- Universidade Federal do Ceara (UFC), 2016.

UNIVASF. Protocolo I2C. I2C Bus, 2013. Disponível em: <http://www.univasf.edu.br/~romulo.camara/novo/wp-content/uploads/2013/11/Barramento-e-Protocolo-I2C>.pdf. Acesso em dezembro 2018.

VALDEZ, J., BECK J. Understanding the I2C Bus. 2015. Disponível em: <http://www.ti.com/lit/an/slva704/slva704.pdf>. Acesso em: out. 2018.

VERAS, Manoel. Datacenter: Componente central da infraestrutura de TI. Rio de Janeiro: Brasport, 2009.

VERAS, Manoel. Virtualização: Componente Central do Datacenter. Rio de Janeiro: Brasport, 2011.

VERAS, Manoel. Computação em nuvem: Nova Arquitetura de TI. Rio de Janeiro: Brasport, 2015.

WU, Caesar. RAJKUMAR, Buyya. Cloud Data Centers and Cost Modeling: A Complete Guide To Planning, Designing and Building a Cloud Data Center. 2015,1ª edição.

YUAN, Michael. Conhecendo o MQTT- Por que o MQTT é um dos melhores protocolos de rede para a Internet das Coisas? Ibm Developer, 2017. Disponível em: <https://www.ibm.com/developerworks/br/ library/iot-mqtt-why-good-for-iot/index.html>. Acesso em dezembro de 2018.

APÊNDICE A

```
//CÓDIGO EXEMPLO: IRrecvDumpV2

#ifndef UNIT_TEST
#include <Arduino.h>
#endif
#include <IRrecv.h>
#include <IRremoteESP8266.h>
#include <IRutils.h>
/*************** Descodificação prolongada de mensagem*****************/
#include <ir_Coolix.h>
#include <ir_Daikin.h>
#include <ir_Fujitsu.h>
#include <ir_Gree.h>
#include <ir_Haier.h>
#include <ir_Hitachi.h>
#include <ir_Kelvinator.h>
#include <ir_Midea.h>
#include <ir_Mitsubishi.h>
#include <ir_Panasonic.h>
#include <ir_Samsung.h>
#include <ir_Toshiba.h>
#include <ir_Whirlpool.h>

const uint16_t kRecvPin = 14;
const uint32_t kBaudRate = 115200;
const uint16_t kCaptureBufferSize = 1024;


        #if DECODE_AC
        const uint8_t kTimeout = 50;

        #else
        const uint8_t kTimeout = 15;

        #endif  // DECODE_AC
        const uint16_t kMinUnknownSize = 12;

        IRrecv irrecv(kRecvPin, kCaptureBufferSize, kTimeout, true);
        decode_results results;
        void dumpACInfo(decode_results *results)
        {
          String description = "";

        #if DECODE_DAIKIN
          if (results->decode_type == DAIKIN) {
            IRDaikinESP ac(0);
            ac.setRaw(results->state);
            description = ac.toString();
          }
        #endif
        #if DECODE_FUJITSU_AC
```

```
  if (results->decode_type == FUJITSU_AC) {
    IRFujitsuAC ac(0);
    ac.setRaw(results->state, results->bits / 8);

    description = ac.toString();
  }
#endif
#if DECODE_KELVINATOR
  if (results->decode_type == KELVINATOR) {
    IRKelvinatorAC ac(0);
    ac.setRaw(results->state);
    description = ac.toString();
  }

#endif
#if DECODE_MITSUBISHI_AC
  if (results->decode_type == MITSUBISHI_AC) {
    IRMitsubishiAC ac(0);
    ac.setRaw(results->state);
    description = ac.toString();
  }

#endif
#if DECODE_TOSHIBA_AC
  if (results->decode_type == TOSHIBA_AC) {
    IRToshibaAC ac(0);
    ac.setRaw(results->state);
    description = ac.toString();
  }

#endif
#if DECODE_GREE
  if (results->decode_type == GREE) {
    IRGreeAC ac(0);
    ac.setRaw(results->state);
    description = ac.toString();
  }

#endif
#if DECODE_MIDEA
  if (results->decode_type == MIDEA) {
    IRMideaAC ac(0);
    ac.setRaw(results->value);  // Midea uses value instead of state.
    description = ac.toString();
  }

#endif
#if DECODE_HAIER_AC
  if (results->decode_type == HAIER_AC) {
    IRHaierAC ac(0);
    ac.setRaw(results->state);
    description = ac.toString();
  }
```

```
  #endif
  #if DECODE_HAIER_AC_YRW02
    if (results->decode_type == HAIER_AC_YRW02) {
      IRHaierACYRW02 ac(0);
      ac.setRaw(results->state);
      description = ac.toString();
    }

  #endif
  #if DECODE_SAMSUNG_AC
    if (results->decode_type == SAMSUNG_AC) {
      IRSamsungAc ac(0);
      ac.setRaw(results->state);
      description = ac.toString();
    }

  #endif
  #if DECODE_COOLIX
    if (results->decode_type == COOLIX) {
      IRCoolixAC ac(0);
      ac.setRaw(results->value);
      description = ac.toString();
    }

  #endif
  #if DECODE_PANASONIC_AC
    if (results->decode_type == PANASONIC_AC &&
        results->bits > kPanasonicAcShortBits) {
      IRPanasonicAc ac(0);
      ac.setRaw(results->state);
      description = ac.toString();
    }

  #endif
  #if DECODE_HITACHI_AC
    if (results->decode_type == HITACHI_AC) {
      IRHitachiAc ac(0);
      ac.setRaw(results->state);
      description = ac.toString();
    }

  #endif
  #if DECODE_WHIRLPOOL_AC
    if (results->decode_type == WHIRLPOOL_AC) {
      IRWhirlpoolAc ac(0);
      ac.setRaw(results->state);
      description = ac.toString();
    }

  #endif
    if (description != "") Serial.println("Mesg Desc.: " + description);
}
```

```
void setup() {
  Serial.begin(kBaudRate, SERIAL_8N1, SERIAL_TX_ONLY);
  while (!Serial)
    delay(50);
  Serial.println();
  Serial.print("IRrecvDumpV2 is now running and waiting for IR input on Pin ");
  Serial.println(kRecvPin);
#if DECODE_HASH
  irrecv.setUnknownThreshold(kMinUnknownSize);
#endif
  irrecv.enableIRIn();
}

void loop()

{  if (irrecv.decode(&results)){
    uint32_t now = millis();
    Serial.printf("Timestamp : %06u.%03u\n", now / 1000, now % 1000);
    if (results.overflow)
      Serial.printf(  "WARNING: código IR é muito grande para o buffer (>= %d). "
          "Este resultado não deve ser confiável até que isso seja resolvido. "
          "Edit & increase kCaptureBufferSize.\n",
          kCaptureBufferSize);
       Serial.print(resultToHumanReadableBasic(&results));
    dumpACInfo(&results);  .
    yield();
    Serial.print("Library  : v");
    Serial.println(_IRREMOTEESP8266_VERSION_);
    Serial.println();
    Serial.println(resultToTimingInfo(&results));
    yield();
    Serial.println(resultToSourceCode(&results));
    Serial.println("");
    yield();
   }

}
```

APÊNDICE B

//CÓDIGO FONTE PROTÓTIPO

//SISTEMA DE MONITORAMENTO CLIMÁTICO E CONTROLE TÉRMICO AUTOMATIZADO

```
#include <IRremoteESP8266.h>
#include <IRsend.h>
#include <UIPEthernet.h>
#include <Wire.h>
#include <DHT.h>
#include <Adafruit_BMP085.h>
#include <ESP8266WiFi.h>


Adafruit_BMP085 bmp;
String apiKey = "ZBNSP59HGUJSXBUN";
const char *ssid =  "XXXXXXXX";
const char *pass =  "XXXXXXXXXXXX
const char* server = "api.thingspeak.com";
#define DHTPIN 12          //D6
DHT dht(DHTPIN, DHT11);


IRsend irsend(14);

uint16_t t_off[277] = {3024, 1642,  438, 1146,  436, 1146,  438, 356,  456, 368,  480, 354,  464,
1146,  436, 354,  464, 406,  436, 1148,  436, 1122,  462, 354,  464, 1146,  438, 352,  464, 356,
432, 1174,  438, 1146,  432, 370,  480, 1122,  462, 1120,  462, 354,  464, 354,  464, 1146,  436,
356,  432, 410,  464, 1144,  436, 354,  464, 356,  488, 378,  438, 354,  464, 356,  486, 340,  478,
354,  434, 392,  482, 354,  464, 354,  432, 392,  484, 352,  464, 354,  434, 390,  484, 352,  464,
354,  492, 380,  436, 354,  464, 356,  456, 368,  480, 1122,  460, 354,  464, 358,  430, 1176,
406, 1176,  406, 390,  484, 354,  464, 354,  432, 390,  484, 354,  464, 356,  430, 1176,  406,
386,  488, 352,  464, 1146,  436, 354,  464, 358,  486, 354,  462, 356,  432, 1172,  410, 390,
482, 1120,  462, 380,  438, 354,  464, 358,  486, 380,  438, 354,  434, 392,  482, 354,  464, 352,
434, 390,  484, 354,  464, 352,  434, 390,  484, 354,  464, 358,  486, 380,  438, 354,  464, 356,
458, 368,  478, 354,  464, 354,  456, 386,  464, 354,  432, 392,  482, 354,  464, 354,  434, 388,
484, 354,  464, 354,  464, 406,  436, 354,  464, 356,  458, 368,  480, 354,  464, 354,  432, 392,
482, 354,  462, 360,  484, 352,  464, 1146,  436, 1146,  436, 354,  464, 354,  434, 392,  482,
1120,  464, 352,  464};

uint16_t t_on[277] ={2974, 1696,  412, 1172,  410, 1148,  436, 404,  412, 382,  434, 390,  428,
1174,  410, 408,  436, 384,  432, 1172,  412, 1172,  410, 386,  430, 1172,  410, 392,  452, 382,
436, 1172,  410, 1172,  410, 384,  434, 1172,  410, 1174,  410, 386,  432, 406,  436, 1148,  434,
382,  436, 388,  430, 1172,  410, 410,  432, 380,  436, 384,  432, 412,  432, 380,  436, 384,  434,
408,  436, 382,  436, 388,  456, 382,  436, 382,  436, 388,  430, 408,  436, 382,  434, 412,  432,
380,  438, 384,  434, 1172,  410, 388,  454, 382,  436, 1172,  410, 382,  436, 392,  426, 1172,
410, 1172,  410, 406,  436, 382,  436, 384,  432, 412,  432, 380,  436, 390,  428, 1174,  410,
406,  436, 380,  438, 1172,  410, 382,  434, 392,  452, 380,  436, 388,  430, 1172,  410, 408,
434, 1150,  434, 380,  436, 382,  436, 388,  456, 380,  436, 386,  430, 408,  436, 380,  436, 386,
432, 406,  438, 382,  434, 386,  432, 412,  432, 382,  436, 390,  454, 382,  436, 382,  436, 390,
454, 382,  436, 380,  438, 386,  430, 406,  434, 410,  410, 430,  412, 382,  436, 384,  434, 410,
432, 380,  436, 384,  434, 408,  436, 380,  436, 388,  430, 406,  432, 408,  412, 388,  430, 408,
```

436, 380, 436, 412, 432, 380, 438, 1172, 410, 380, 436, 1172, 412, 386, 430, 408, 436, 1148, 434, 380, 436};

uint16_t temp16[277] = {2972, 1698, 412, 1174, 408, 1174, 408, 382, 436, 388, 454, 382, 436, 1150, 432, 382, 436, 390, 426, 1174, 408, 1174, 410, 412, 430, 1150, 434, 382, 436, 382, 434, 1174, 410, 1176, 406, 390, 428, 1174, 410, 1172, 410, 392, 452, 382, 436, 1172, 410, 380, 436, 408, 434, 1150, 434, 380, 436, 382, 434, 390, 454, 382, 434, 382, 436, 390, 454, 382, 438, 386, 430, 408, 434, 382, 434, 386, 432, 434, 410, 382, 434, 386, 432, 408, 436, 382, 436, 390, 428, 1174, 410, 408, 434, 382, 436, 1172, 410, 384, 434, 408, 436, 1172, 410, 1150, 434, 382, 436, 382, 434, 390, 454, 380, 436, 382, 434, 408, 436, 1150, 432, 1150, 408, 1172, 410, 1170, 412, 412, 432, 382, 436, 382, 434, 408, 436, 1172, 410, 380, 436, 1172, 410, 382, 434, 392, 454, 380, 436, 382, 436, 408, 436, 380, 436, 386, 430, 408, 436, 382, 436, 386, 432, 406, 436, 380, 436, 392, 452, 380, 436, 384, 434, 390, 452, 382, 436, 382, 436, 390, 452, 382, 436, 386, 430, 408, 436, 382, 436, 386, 432, 406, 436, 380, 436, 386, 434, 410, 432, 382, 436, 390, 454, 382, 436, 380, 438, 388, 428, 408, 436, 380, 436, 388, 430, 406, 436, 386, 432, 410, 432, 380, 436, 384, 434, 1172, 410, 388, 428, 1174, 410, 406, 436};

uint16_t temp17[277] = {3004, 1668, 438, 1120, 462, 1118, 464, 352, 464, 352, 468, 356, 430, 1174, 408, 392, 484, 350, 466, 1144, 440, 1144, 438, 352, 466, 1144, 408, 388, 486, 350, 466, 1144, 438, 1056, 526, 352, 466, 1144, 438, 1144, 440, 350, 434, 390, 484, 1120, 464, 350, 466, 354, 432, 1176, 406, 390, 484, 350, 468, 350, 434, 390, 486, 352, 466, 348, 436, 434, 440, 352, 466, 354, 456, 412, 438, 350, 468, 354, 432, 392, 482, 350, 466, 358, 486, 350, 466, 352, 466, 1144, 438, 354, 488, 378, 440, 1122, 462, 350, 466, 358, 428, 1174, 408, 1174, 408, 390, 484, 352, 466, 352, 434, 390, 484, 350, 466, 356, 488, 352, 464, 1146, 438, 1120, 460, 1122, 462, 378, 440, 352, 464, 356, 488, 354, 464, 1144, 440, 352, 464, 1144, 438, 354, 456, 370, 480, 352, 464, 356, 432, 408, 464, 354, 434, 390, 484, 352, 464, 352, 434, 390, 486, 352, 464, 352, 464, 406, 438, 354, 462, 356, 456, 366, 482, 354, 464, 354, 432, 394, 482, 352, 464, 360, 484, 352, 464, 354, 464, 358, 486, 354, 464, 354, 464, 358, 486, 354, 464, 356, 456, 370, 480, 354, 462, 354, 432, 394, 480, 354, 464, 352, 434, 392, 482, 354, 464, 1148, 434, 1148, 436, 1146, 436, 1148, 434, 356, 462, 356, 432, 1176, 406, 392, 480};

uint16_t temp18[277] = {3026, 1642, 440, 1144, 438, 1144, 440, 352, 434, 390, 484, 352, 464, 1124, 460, 352, 464, 362, 482, 1120, 432, 1150, 462, 360, 460, 1120, 464, 352, 466, 352, 464, 1146, 438, 1144, 408, 390, 484, 1118, 408, 1174, 408, 392, 482, 352, 466, 1142, 440, 352, 434, 410, 466, 1120, 462, 352, 466, 352, 434, 392, 482, 352, 466, 352, 434, 392, 484, 352, 466, 358, 486, 352, 466, 352, 466, 356, 486, 380, 438, 354, 464, 356, 488, 350, 468, 354, 432, 392, 482, 1120, 464, 352, 464, 352, 466, 1144, 440, 354, 458, 384, 464, 1146, 436, 1122, 460, 354, 466, 354, 432, 392, 480, 354, 464, 354, 432, 408, 466, 1124, 460, 354, 464, 1144, 436, 1146, 438, 354, 464, 358, 484, 354, 464, 356, 430, 1176, 406, 392, 482, 1120, 462, 354, 462, 354, 464, 358, 486, 380, 436, 356, 456, 368, 480, 354, 464, 354, 434, 392, 480, 356, 462, 354, 434, 390, 484, 354, 462, 360, 484, 380, 436, 354, 462, 358, 486, 382, 436, 354, 462, 358, 486, 356, 462, 356, 432, 392, 480, 354, 464, 354, 432, 392, 482, 356, 462, 356, 462, 408, 436, 354, 432, 388, 486, 340, 448, 384, 462, 358, 458, 368, 478, 356, 462, 362, 480, 356, 462, 1150, 434, 1150, 432, 1124, 460, 354, 462, 356, 462, 1148, 406, 386, 484};

uint16_t temp19[277] = {3030, 1668, 470, 1112, 470, 1090, 462, 378, 472, 344, 408, 412, 406, 1178, 404, 408, 500, 346, 408, 1174, 470, 978, 608, 372, 380, 1176, 408, 414, 490, 348, 440, 1142, 472, 1084, 470, 350, 466, 1142, 440, 1114, 468, 378, 406, 436, 440, 1120, 492, 348, 442, 378, 404, 1178, 406, 436, 440, 376, 474, 344, 382, 438, 464, 348, 470, 376, 408, 410, 464, 352, 432, 414, 462, 350, 498, 346, 440, 358, 482, 352, 472, 348, 466, 402,

440, 376, 470, 348, 436, 1120, 408, 440, 462, 350, 470, 1140, 472, 318, 468, 382, 460, 1122, 402, 1180, 458, 382, 440, 348, 472, 376, 406, 438, 440, 350, 468, 380, 494, 322, 466, 376, 442, 1116, 466, 1118, 466, 378, 408, 412, 464, 352, 494, 350, 492, 324, 466, 1116, 468, 374, 442, 352, 432, 436, 438, 380, 440, 348, 434, 436, 440, 376, 408, 390, 486, 376, 440, 378, 438, 382, 460, 354, 468, 376, 440, 404, 438, 352, 466, 352, 434, 412, 464, 348, 468, 378, 408, 436, 438, 376, 470, 350, 460, 356, 494, 348, 438, 382, 460, 378, 444, 350, 466, 378, 406, 408, 498, 348, 406, 412, 464, 378, 440, 378, 408, 412, 464, 378, 440, 352, 432, 416, 490, 348, 474, 348, 404, 1178, 406, 436, 438, 1124, 460, 378, 440, 378, 440, 1144, 440, 356, 402};

uint16_t temp20[277] = {3050, 1622, 500, 1078, 508, 1076, 502, 342, 476, 344, 460, 382, 442, 1118, 496, 344, 476, 344, 456, 1098, 432, 1174, 408, 390, 484, 1098, 514, 348, 476, 342, 474, 1106, 446, 1110, 472, 376, 406, 1150, 432, 1174, 410, 412, 466, 376, 444, 1108, 474, 374, 476, 348, 458, 1122, 464, 376, 444, 374, 408, 388, 514, 322, 470, 374, 474, 346, 462, 352, 472, 374, 406, 436, 474, 342, 472, 320, 482, 340, 516, 346, 476, 342, 408, 410, 500, 318, 468, 356, 428, 1150, 434, 436, 442, 374, 444, 1138, 446, 372, 408, 410, 468, 1140, 440, 1118, 466, 348, 498, 346, 410, 412, 462, 378, 444, 372, 440, 404, 472, 1088, 460, 1122, 462, 350, 472, 1140, 474, 342, 442, 350, 432, 436, 442, 350, 432, 412, 460, 1122, 460, 354, 498, 346, 444, 376, 462, 380, 474, 344, 444, 402, 442, 374, 444, 374, 406, 416, 462, 348, 472, 374, 408, 412, 464, 376, 472, 326, 484, 354, 496, 348, 440, 378, 406, 436, 442, 374, 442, 354, 430, 438, 440, 348, 436, 416, 460, 378, 442, 374, 408, 414, 460, 378, 442, 350, 466, 380, 460, 354, 466, 378, 406, 412, 464, 378, 440, 378, 408, 438, 440, 376, 442, 352, 432, 436, 442, 1114, 468, 376, 470, 350, 404, 1152, 432, 414, 462, 350, 470, 1142, 440, 378, 440};

uint16_t temp21[277] = {2998, 1672, 504, 1106, 478, 1106, 476, 342, 474, 344, 458, 360, 494, 1088, 498, 344, 476, 344, 404, 1152, 430, 1176, 408, 412, 460, 1100, 486, 352, 504, 342, 472, 1082, 504, 1104, 478, 344, 406, 1148, 468, 1140, 408, 412, 466, 376, 446, 1138, 476, 342, 474, 346, 460, 1120, 464, 376, 444, 372, 476, 344, 460, 354, 502, 340, 442, 376, 460, 354, 472, 374, 406, 412, 468, 372, 476, 342, 406, 412, 466, 374, 474, 344, 406, 410, 472, 372, 474, 346, 404, 1152, 432, 412, 468, 374, 472, 1108, 476, 342, 408, 434, 444, 1118, 460, 1120, 464, 376, 444, 374, 440, 380, 464, 378, 444, 374, 474, 370, 444, 372, 442, 1112, 470, 350, 498, 1110, 442, 354, 486, 350, 472, 346, 472, 402, 442, 346, 470, 1138, 444, 372, 442, 380, 462, 350, 502, 344, 472, 348, 462, 378, 444, 376, 406, 414, 464, 374, 442, 376, 408, 414, 464, 348, 498, 348, 406, 410, 496, 348, 440, 380, 460, 354, 466, 376, 442, 378, 406, 410, 466, 376, 440, 380, 462, 352, 464, 354, 430, 414, 462, 354, 464, 378, 406, 412, 462, 380, 438, 380, 404, 410, 466, 380, 440, 380, 406, 412, 462, 378, 440, 354, 430, 414, 460, 352, 434, 414, 460, 352, 464, 380, 438, 1120, 464, 356, 426, 438, 436, 1124, 460, 354, 464};

uint16_t temp22[277] = {2976, 1672, 410, 1172, 410, 1172, 410, 390, 428, 432, 412, 404, 412, 1172, 410, 406, 410, 430, 414, 1172, 410, 1172, 410, 404, 412, 1172, 410, 406, 412, 390, 428, 1172, 410, 1174, 434, 404, 412, 1172, 410, 1148, 434, 404, 414, 406, 412, 1172, 410, 388, 428, 430, 412, 1172, 410, 406, 410, 392, 452, 404, 412, 404, 412, 390, 452, 404, 412, 408, 410, 430, 412, 404, 412, 408, 410, 430, 412, 404, 412, 408, 410, 430, 412, 406, 412, 392, 452, 404, 412, 1172, 410, 406, 410, 410, 410, 1172, 410, 410, 432, 404, 412, 1172, 410, 1172, 410, 406, 410, 392, 452, 404, 412, 382, 436, 392, 452, 404, 412, 1172, 410, 408, 410, 410, 432, 1150, 434, 404, 412, 404, 412, 390, 454, 406, 412, 1174, 410, 406, 412, 1172, 410, 390, 428, 430, 412, 404, 412, 388, 430, 432, 412, 406, 412, 410, 434, 404, 414, 406, 412, 412, 432, 404, 412, 406, 412, 432, 412, 404, 412, 388, 430, 430, 412, 406, 412, 388, 430, 430, 414, 380, 436, 412, 432, 382, 436, 406, 412, 392, 452, 404, 412, 406, 410, 392, 452, 406, 412, 408, 410, 408, 434, 404, 412, 388, 430, 408, 436, 380, 436,

386, 430, 432, 412, 406, 412, 390, 428, 1172, 410, 430, 412, 1172, 410, 404, 412, 386, 432, 1172, 410, 390, 452};

uint16_t temp23[277] = {3006, 1666, 442, 1118, 464, 1118, 466, 350, 468, 350, 436, 388, 430, 1172, 410, 412, 464, 350, 466, 1140, 440, 1144, 440, 352, 432, 1174, 406, 414, 462, 376, 444, 1140, 442, 1142, 440, 350, 436, 1172, 410, 1174, 408, 384, 434, 392, 452, 1148, 434, 382, 436, 388, 430, 1174, 410, 390, 454, 382, 436, 382, 436, 390, 454, 382, 436, 380, 436, 408, 436, 382, 436, 388, 430, 408, 436, 380, 436, 386, 430, 412, 432, 382, 436, 392, 452, 380, 436, 382, 436, 1172, 410, 386, 430, 408, 436, 1150, 432, 382, 436, 388, 428, 1174, 410, 1172, 410, 390, 452, 380, 438, 380, 436, 390, 454, 382, 436, 388, 430, 406, 436, 380, 436, 386, 430, 1172, 410, 390, 452, 382, 436, 380, 436, 408, 436, 1172, 410, 380, 438, 1174, 410, 382, 436, 390, 454, 382, 436, 380, 436, 408, 436, 380, 438, 386, 432, 406, 436, 380, 436, 386, 432, 410, 434, 380, 436, 392, 452, 380, 436, 382, 436, 390, 454, 382, 436, 380, 436, 388, 430, 406, 436, 386, 432, 408, 436, 382, 436, 384, 432, 410, 434, 380, 436, 384, 434, 392, 452, 380, 436, 390, 454, 382, 436, 380, 436, 388, 430, 406, 436, 380, 436, 388, 430, 406, 436, 1174, 410, 380, 436, 390, 428, 1174, 410, 406, 436, 380, 436, 1172, 410, 382, 434};

uint16_t temp24[277] = {2972, 1696, 444, 1140, 442, 1140, 442, 348, 470, 352, 432, 436, 442, 1118, 464, 350, 468, 356, 430, 1174, 408, 1174, 408, 390, 484, 1118, 464, 350, 468, 350, 466, 1144, 440, 1144, 438, 358, 428, 1176, 406, 1176, 406, 392, 484, 352, 464, 1144, 438, 352, 466, 362, 482, 1120, 462, 354, 464, 354, 434, 390, 484, 354, 464, 352, 464, 358, 486, 354, 464, 356, 456, 368, 480, 352, 464, 356, 432, 394, 480, 354, 464, 354, 434, 410, 464, 354, 462, 360, 452, 1152, 462, 340, 478, 356, 460, 1148, 436, 354, 432, 410, 462, 1148, 434, 1124, 460, 354, 462, 354, 434, 390, 484, 354, 462, 356, 462, 408, 436, 1138, 414, 1152, 460, 1122, 460, 382, 436, 356, 462, 358, 486, 338, 478, 354, 434, 1178, 436, 358, 452, 1152, 432, 368, 448, 386, 432, 384, 458, 368, 478, 356, 434, 390, 452, 384, 462, 354, 464, 358, 484, 354, 462, 356, 462, 358, 486, 354, 464, 354, 432, 394, 482, 352, 464, 354, 434, 392, 484, 352, 466, 352, 434, 410, 468, 350, 468, 354, 488, 352, 468, 350, 468, 354, 430, 414, 464, 348, 468, 358, 486, 350, 468, 348, 436, 412, 432, 382, 436, 382, 436, 388, 454, 384, 436, 386, 432, 412, 432, 382, 436, 384, 434, 1172, 410, 390, 454, 382, 436, 1174, 410, 380, 436};

uint16_t temp25[277] = {2998, 1626, 482, 1122, 406, 1176, 406, 392, 480, 354, 434, 384, 434, 1118, 490, 386, 460, 356, 462, 1146, 406, 1176, 434, 356, 462, 1146, 406, 386, 458, 368, 480, 1122, 460, 1150, 434, 356, 432, 1176, 434, 1148, 406, 384, 462, 358, 430, 1178, 406, 392, 480, 354, 462, 1148, 434, 358, 486, 338, 478, 356, 462, 356, 458, 368, 480, 354, 462, 362, 482, 356, 462, 354, 462, 360, 484, 354, 462, 356, 462, 360, 484, 354, 462, 356, 458, 368, 478, 354, 462, 1146, 436, 354, 434, 390, 482, 1124, 460, 380, 436, 356, 462, 1146, 436, 1148, 436, 356, 486, 340, 478, 354, 462, 356, 458, 368, 480, 354, 434, 390, 482, 1120, 406, 1176, 432, 368, 480, 354, 462, 356, 432, 392, 480, 354, 462, 1146, 436, 358, 430, 1176, 406, 392, 482, 354, 462, 354, 434, 390, 482, 354, 464, 358, 486, 378, 438, 354, 464, 358, 486, 380, 438, 354, 464, 356, 458, 384, 464, 354, 434, 390, 484, 354, 462, 354, 432, 390, 482, 354, 464, 354, 464, 406, 436, 354, 464, 354, 488, 340, 478, 354, 464, 356, 458, 368, 480, 354, 464, 360, 484, 354, 464, 354, 434, 390, 484, 354, 462, 354, 464, 358, 486, 354, 464, 356, 432, 1176, 436, 1146, 438, 1146, 436, 356, 458, 368, 480, 354, 464, 1146, 438, 352, 434};

uint16_t temp26[277] = {3064, 1576, 502, 1108, 480, 1078, 502, 344, 460, 382, 476, 342, 476, 1080, 472, 372, 408, 406, 504, 1086, 462, 1118, 468, 374, 472, 1084, 472, 372, 474, 346, 406, 1174, 444, 1118, 488, 352, 470, 1096, 484, 1118, 466, 350, 470, 372, 446, 1136, 446, 376, 460, 354, 472, 1136, 476, 342, 444, 376, 464, 350, 470, 372, 476, 344, 462, 350, 502, 342, 408, 412, 466, 372, 446, 372, 410, 412, 494, 346, 448, 368, 410, 410, 494, 318,

```
502, 346,  462, 352,  470, 1138,  444, 372,  444, 372,  444, 1112,  436, 388,  490, 374,  478,
1080,  470, 1136,  446, 346,  436, 388,  488, 348,  502, 342,  476, 344,  464, 346,  474, 1136,
446, 346,  472, 1136,  444, 352,  486, 350,  472, 372,  446, 372,  408, 406,  502, 344,  474, 1108,
410, 388,  488, 346,  474, 344,  472, 374,  466, 348,  472, 374,  408, 412,  466, 372,  444, 372,
410, 390,  488, 346,  500, 344,  408, 390,  488, 348,  470, 352,  488, 350,  472, 344,  474, 352,
488, 348,  472, 346,  472, 350,  486, 350,  474, 372,  410, 390,  488, 346,  472, 346,  436, 388,
490, 346,  472, 372,  410, 408,  470, 348,  470, 350,  490, 350,  472, 348,  502, 344,  406, 434,
446, 370,  446, 376,  462, 1118,  406, 1174,  464, 380,  444, 346,  472, 348,  434, 392,  484,
1096,  488, 348,  472};

int AC_Temp;
char temp_erro = 2;
int ref;
boolean AC = false;
WiFiClient client;
     void setup()
 {
       irsend.begin();
       Serial.begin(115200);
       delay(10);
       dht.begin();
       Wire.begin (4, 5);
       Wire.setClock(400000); //Temporização entre dispositivos

    if (!bmp.begin())
       {
         Serial.println("Sensor não encontrado!");
         while (1) { }
        }
      Serial.println("Conectando");
      Serial.println(ssid);
        WiFi.begin(ssid, pass);

      while (WiFi.status() != WL_CONNECTED)
    {
          delay(500);
          Serial.print(".");
    }
     Serial.println("");
     Serial.println("WiFi conectado");}


void loop() {

     float u = dht.readHumidity();
     float t1 = dht.readTemperature();
     float p = bmp.readPressure();
     float t2 = bmp.readTemperature();
     float temp_base= 23;
     float temp_min= 20;
     float temp_max= 27;
     float umid_min= 40;
     float umid_max= 55;
```

```
    float temp=t1+temp_erro;


  //MONITORAMENTO DAS VARIÁVEIS TEMPERATURA, UMIDADE E PRESSÃO
          if (isnan(u) || isnan(t1))
            { Serial.println("Falha na leitura do sensor DHT11!");
              return;}
          if (isnan(p)||isnan(t2))
            { Serial.println("Falha na leitura do sensor BMP180!");
              return; }
           if (client.connect(server,80))   //
                {     String postStr = apiKey;
                      postStr +="&field1=";
                      postStr += String(t1);
                      postStr +="&field2=";
                      postStr += String(u);
                      postStr +="&field3=";
                      postStr += String(p);
                      postStr +="&field4=";
                      postStr += String(t2);
                      postStr += "\r\n\r\n";

                      client.print("POST /update HTTP/1.1\n");
                      client.print("Host: api.thingspeak.com\n");
                      client.print("Connection: close\n");
                      client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
                      client.print("Content-Type: application/x-www-form-urlencoded\n");
                      client.print("Content-Length: ");
                      client.print(postStr.length());
                      client.print("\n\n");
                      client.print(postStr);

                      Serial.print("Temperatura DHT11: ");
                      Serial.print(t1);
                      Serial.print(" graus Celsius\nTemperatura BMP180: ");
                      Serial.print(t2);
                      Serial.print(" graus Celsius\nUmidade: ");
                      Serial.print(u);
                      Serial.print(" %\nPressão: ");
                      Serial.print(p/100);
                      Serial.println(" Hectopascal.\nEnviando para Thingspeak.");
                  }
          client.stop();
          Serial.println("Esperando...");

//CONTROLE TÉRMICO AUTOMATIZADO

  if ((temp >= (temp_base+4)) && AC == true)
      {     irsend.sendRaw(t_on,277,38);  delay(2000);
            AC_Temp = 0; AC =false;
       }
   if ( temp != ref)
       {
```

```
                if (temp == temp_base+3)
                { irsend.sendRaw(temp20,277,38);  delay(2000);
                  AC_Temp = 20;
                 }

                 if (temp == temp_base+2)
                 { irsend.sendRaw(temp21,277,38);  delay(2000);
                   AC_Temp = 21;
                  }

                 if (temp == temp_base+1)
                   { irsend.sendRaw(temp22,277,38);  delay(2000);
                     AC_Temp =22 ;
                   }

                 if (temp == temp_base-3)
                   { irsend.sendRaw(temp26,277,38);  delay(2000);
                     AC_Temp = 26;  }

               if (temp == temp_base-2)
                   { irsend.sendRaw(temp25,277,38);  delay(2000);
                     AC_Temp = 0; }

                 if (temp == temp_base-1)
                   { irsend.sendRaw(temp24,277,38);  delay(2000);
                     AC_Temp = 24; AC =false;}

                if (temp == temp_base)
                   { irsend.sendRaw(temp23,277,38);  delay(2000);
                     AC_Temp = 23;   }
    }
        ref = temp;
 delay(10000);
}
```