UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA

MARCOS HENRIQUE CORREA LIMA

PROTÓTIPO PARA AFERIÇÃO, MONITORAMENTO E ANÁLISE DE PARÂMETROS DO AR EM DATA CENTERS

Manaus
2022

MARCOS HENRIQUE CORREA LIMA

PROTÓTIPO PARA AFERIÇÃO, MONITORAMENTO E ANÁLISE DE PARÂMETROS DO AR EM DATA CENTERS

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Eletrônica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro em Eletrônica.

Orientação: Fábio de Sousa Cardoso, Dr.

Manaus

2022

Universidade do Estado do Amazonas – UEA

Escola Superior de Tecnologia - EST

Reitor:

André Luiz Nunes Zogahib

Vice-Reitor:

Kátia do Nascimento Couceiro

Diretor da Escola Superior de Tecnologia:

Ingrid Sammyne Gadelha Figueiredo

Coordenador do Curso de Engenharia Eletrônica:

Bruno da Gama Monteiro

Banca Avaliadora composta por:

Data da defesa: <26/10/2022>.

Prof. Fábio de Sousa Cardoso, Dr *(Orientador)*

Prof. Angilberto Muniz Ferreira Sobrinho, Dr.

Prof. Jozias Parente de Oliveira, Dr.

CIP – Catalogação na Publicação

Lima, Marcos Henrique Correa

Protótipo para aferição, monitoramento e análise de par metros do ar em data centers / Marcos Henrique Correa Lima; [orientado por] Fábio de Sousa Cardoso. – Manaus: 2022.

57 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica). Universidade do Estado do Amazonas, 2022.

1. monitoramento de temperatura e umidade. 2. Wifi. 3. Esp 32. 4. Qualidade do ar. 5. Arduino. I. Cardoso, Fábio de Sousa.

MARCOS HENRIQUE CORREA LIMA

PROTÓTIPO PARA AFERIÇÃO, MONITORAMENTO E ANÁLISE DE PAR METROS DO AR EM DATA CENTERS

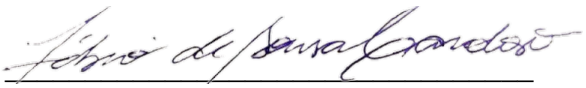
Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Eletrônica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro em Eletrônica.

Nota obtida: 9,1 (Nove virgula um)

Aprovada em 20/10/2022.

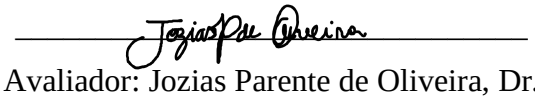
Área de concentração: Monitoramento de parâmetros do ar, sensores

BANCA EXAMINADORA



Orientador: Fábio de Sousa Cardoso, Dr.

Avaliador: Angilberto Muniz Ferreira Sobrinho, Dr.



Avaliador: Jozias Parente de Oliveira, Dr.

Manaus 2022

AGRADECIMENTOS

Primeiramente aos meus pais e meus irmãos, por sempre me apoiarem, incentivarem e sempre se dispuseram a ajudar de todas as formas. Aos meus amigos Rodrigo Alencar de Souza, Darc Pabla Sodré da Silva, Pedro Paulo Brantis de Carvalho, Denis Moraes Guimarães, Marcos Belchior Garcia, Jefferson Thiago, Arlley Gabriel Dias e Dias que sempre tornaram os dias melhores e colaboraram e contribuíram para a minha formação acadêmica.

RESUMO

O presente trabalho apresenta o desenvolvimento de um protótipo de aferição, monitoramento e análise de parâmetros do ar em *data centers*. Para isso, foi feita uma revisão da literatura dos assuntos pertinentes a essa pesquisa. Após isso, foi iniciado o desenvolvimento com a instalação das bibliotecas no Arduino IDE e calibração dos sensores utilizados. Com isso, foi feito o protótipo completo, integrando os sensores com a placa do Arduino e a placa do Esp 32, responsável pelo envio das informações coletadas para um servidor na nuvem com um Data-base MySQL, onde essas informações seriam armazenadas. Por fim, foi desenvolvido um aplicativo para exibir ao usuário os resultados coletados pelos sensores, com um gráfico para mostrar a evolução ao longo do tempo, seus máximos e mínimos e com uma notificação caso seja constatado que algum dos parâmetros não está conforme os padrões. Como resultado, foi possível desenvolver tal protótipo, que atende aos requisitos tanto nas medições dos parâmetros, quanto no envio dos dados e exibição de dados para o usuário do aplicativo.

Palavras-chave: Arduino IDE, monitoramento de temperatura e umidade, Wi-fi, Esp32, Android.

ABSTRACT

The present work presents the development of a prototype for gauging, monitoring, and analyzing air parameters in data centers. For this, a literature review of the issues relevant to this research was carried out. After that, the development started with the installation of the libraries in the Arduino IDE and calibration of the sensors used. With this, the complete prototype was made, integrating the sensors with the Arduino board and the ESP 32 board, responsible for sending the collected information to a cloud server with a MySQL database, where this information would be stored. Finally, an application was developed to display the results collected by the sensors to the user, with a graph to show the evolution over time, its maximum and minimum and with a notification if it is found that any of the parameters does not conform to the standards. As a result, it was possible to develop such a prototype, which meets the requirements both in parameter measurements, and in sending data and displaying data to the application user.

Keywords: Arduino IDE, temperature, and humidity monitoring, Wi-Fi, Esp32, Android.

SUMÁRIO

INTRODUÇÃO

Datacenter, ou Centro de Processamento de Dados, é um ambiente projetado para abrigar equipamentos de processamento, armazenamento e sistemas ativos de redes como comutadores de tráfego e roteadores (SILVA, 2018).

Eles surgiram principalmente para proporcionar um melhor gerenciamento e proteção sobre os equipamentos e os dados ali instalados. Os equipamentos de um *data center* são extremamente caros e sensíveis, além disso a perda dos dados presentes nestes equipamentos pode levar as empresas e os próprios clientes a terem prejuízos imensuráveis (RIBEIRO, 2020).

A importância do controle ambiental em *data centers* se dá, principalmente, pelas falhas ocasionadas por altas temperaturas e umidade do ar. A falha de *hardware*, por exemplo, ocorre devido a superaquecimento e é um dos principais motivos de tempos de indisponibilidade em *data centers.* E, como *datacenters* menores tendem a não usar sistemas de refrigeração de precisão devido ao custo elevado, os sistemas de monitoramento são vitais (SILVA, 2018).

Neste contexto, esse projeto tem como objetivo desenvolver um protótipo que faça a aferição e monitoramento dos parâmetros do ar em *datacenters* como umidade do ar, temperatura, ponto do orvalho e qualidade do ar. Além disso, mostrar os resultados obtidos de maneira simples e direta para o usuário por meio de um aplicativo Android.

Para teste e validação do objetivo acima, foram implementados alguns objetivos específicos como teste em um data center real e a exibição dos resultados no aplicativo Android.

Essa pesquisa se justifica por disponibilizar às empresas uma maneira de acompanhar os parâmetros do ar de maneira simples e rápida, além de um baixo custo de instalação e manutenção. Outra justificativa é que um *data center* é considerado um ambiente de missão crítica, ou seja, abrigam tecnologia que funciona de forma ininterrupta, 24 horas por dia, não podendo assim, sofrer de falhas de funcionamento por conta desses parâmetros do ar.

O presente trabalho se divide em 4 capítulos: Referencial teórico, Métodos e Materiais, implementação do projeto e resultados obtidos.

No primeiro capítulo, apresenta uma revisão bibliográfica sobre os principais

conceitos e tecnologias que serviram de base para o desenvolvimento do projeto. Neste capítulo, são explicados os parâmetros a serem medidos, além das placas usadas como Arduino e Esp32. Também são descritas as formas de comunicação utilizadas que são UART e Wi-fi e os *softwares* utilizados no desenvolvimento dos códigos.

No segundo capítulo são mostrados os métodos utilizados para se obter o resultado sendo dividido em 5 etapas para organizar o desenvolvimento. No terceiro capítulo, implementação do projeto, é abordada a execução das etapas citadas no capítulo de métodos. Mostra-se a preparação do ambiente de desenvolvimento, todos os testes executados, a confecção do protótipo, configuração do banco de dados e desenvolvimento do aplicativo.

No quarto capítulo, resultados obtidos, mostram-se os resultados dos experimentos realizados na implementação, e faz-se análises baseadas no referencial teórico e na experiência de usuário de um aplicativo Android para mostrar os resultados.

E por fim é apresentada a conclusão que retorna ao que foi proposto mostrando que o protótipo consegue realizar a aferição e o monitoramento, atendendo assim às necessidades do que foi proposto anteriormente.

1 REFERENCIAL TEÓRICO

1.1 TEMPERATURA E UMIDADE

> Temperatura é uma grandeza fenomenológica de estado que está relacionada ao “estado” de ‘quente’ ou de ‘frio’. A etimologia aponta que Temperatura vem do latim *temperare*, significando ‘misturar corretamente, regular, moderar’. Misturar, neste caso, envolve pelo menos dois sistemas com estados térmicos distintos. O foco desse estado térmico é a Temperatura. Do ponto de vista qualitativo, a Temperatura está associada ao estado de aquecimento de um sistema, enquanto, do ponto de vista quantitativo, a sua medida está relacionada ao nível desse estado de aquecimento (CORREIA, 2017).

Apesar de haver grande discussão acerca de qual a temperatura ideal para operação de um *data center*, a recomendação da *American Society of Heating, Refrigerating and Air Conditioning Engineers (*ASHRAE,2019) é que a temperatura ideal na entrada de ar dos equipamentos críticos de TI esteja entre 18° C e 27° C com umidade relativa do ar entre 40 e 55%.

> A Umidade Relativa do Ar (UR) é definida como a relação, expressa em porcentagem, entre a umidade absoluta e a umidade no seu ponto de saturação para determinada temperatura, ou ainda, a relação entre as pressões parcial de vapor e a pressão de saturação, à mesma temperatura (SEARS, 1984).

1.2 PONTO DO ORVALHO

O ponto de orvalho é aquela temperatura abaixo da qual o vapor de água em um corpo de ar não pode permanecer vapor. Quando um corpo de ar é resfriado até seu ponto de orvalho ou abaixo, alguma fração de seu vapor de água muda da fase gasosa para a líquida para formar neblina ou gotículas de nuvem. Se uma superfície lisa estiver disponível, o vapor condensa diretamente sobre ela como gotas de água (orvalho) (JRANK, 2018). O mesmo pode ser visto na figura 1.

Figura 1 – Orvalho formado em uma janela.

Fonte: Labomat, 2021

1.3 QUALIDADE DO AR

De acordo com Ministério do Meio Ambiente (2009), de uma forma geral, a qualidade do ar é produto da interação de um complexo conjunto de fatores dentre os quais destacam-se a magnitude das emissões, a topografia e as condições meteorológicas da região, favoráveis ou não à dispersão dos poluentes. A qualidade do ar num determinado local varia ao ritmo das emissões dos poluentes por suas fontes e das condições naturais, como as condições meteorológicas e a topografia da região.

Para enfrentar esse problema, o primeiro passo é o monitoramento para informar a situação da qualidade do ar no tempo e no espaço (ENERGIA E AMBIENTE, 2022).

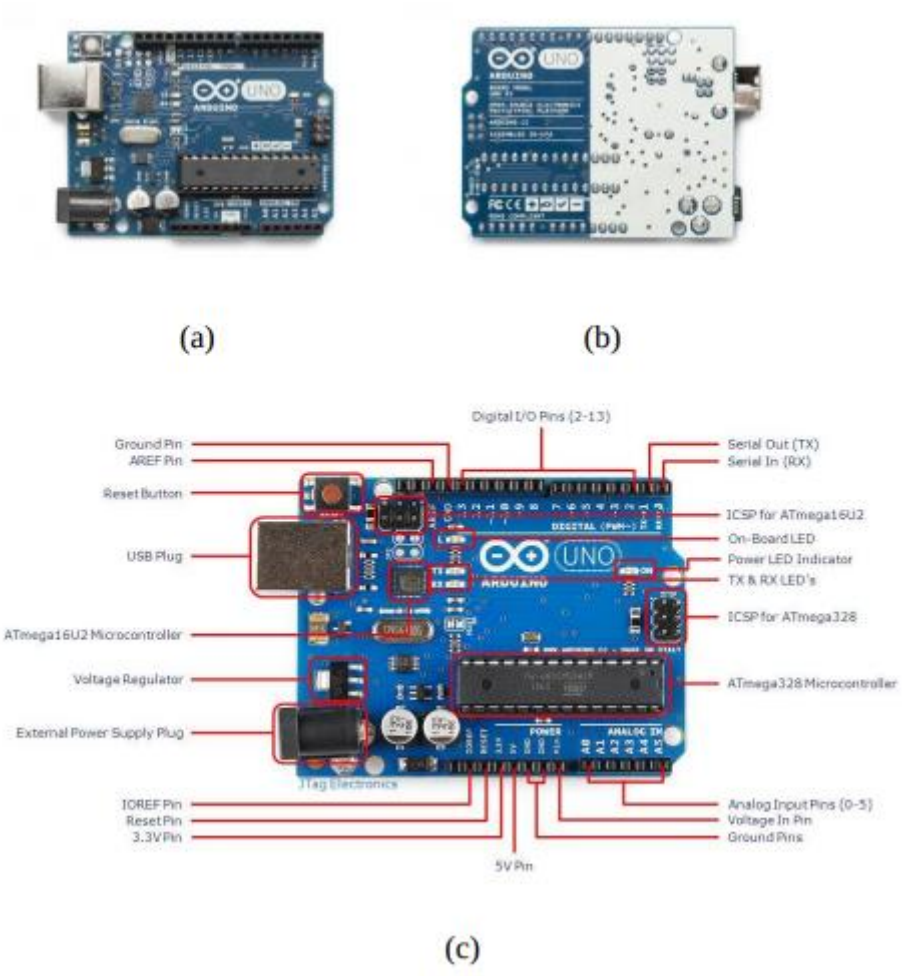
1.4 ARDUINO

O Arduino é uma placa de componentes eletrônicos criada em 2005 na Itália. A placa é composta de um microcontrolador de prototipagem eletrônica da fabricante ATMEL (microchip) e seu objetivo é o desenvolvimento para controle de sistemas interativos de baixo custo (ARDUINO, 2019).

O *hardware* é simples, mas bastante eficiente. O microcontrolador possui

entradas e saídas acopladas podendo ser conectadas a outros circuitos. Com o Arduino é possível conectar diferente tipos de periféricos, entre eles, displays, botões, sensores, LEDs, buzzers, componentes eletrônicos, além disso, também pode ser utilizado com diversas Shields (placas conectadas em cima da Printed Circuit Board Assembly-(PCBA) (FONSECA; BEPPU, 2010). Ele pode ser visto mais detalhadamente na figura 2, onde é mostrado em diferentes ângulos e são destacadas suas partes principais.

Figura 2 – Arduino UNO rev. 3: (a) Frente. (b) Verso. (c) Esquema.



Fonte: Arduino, 2019.

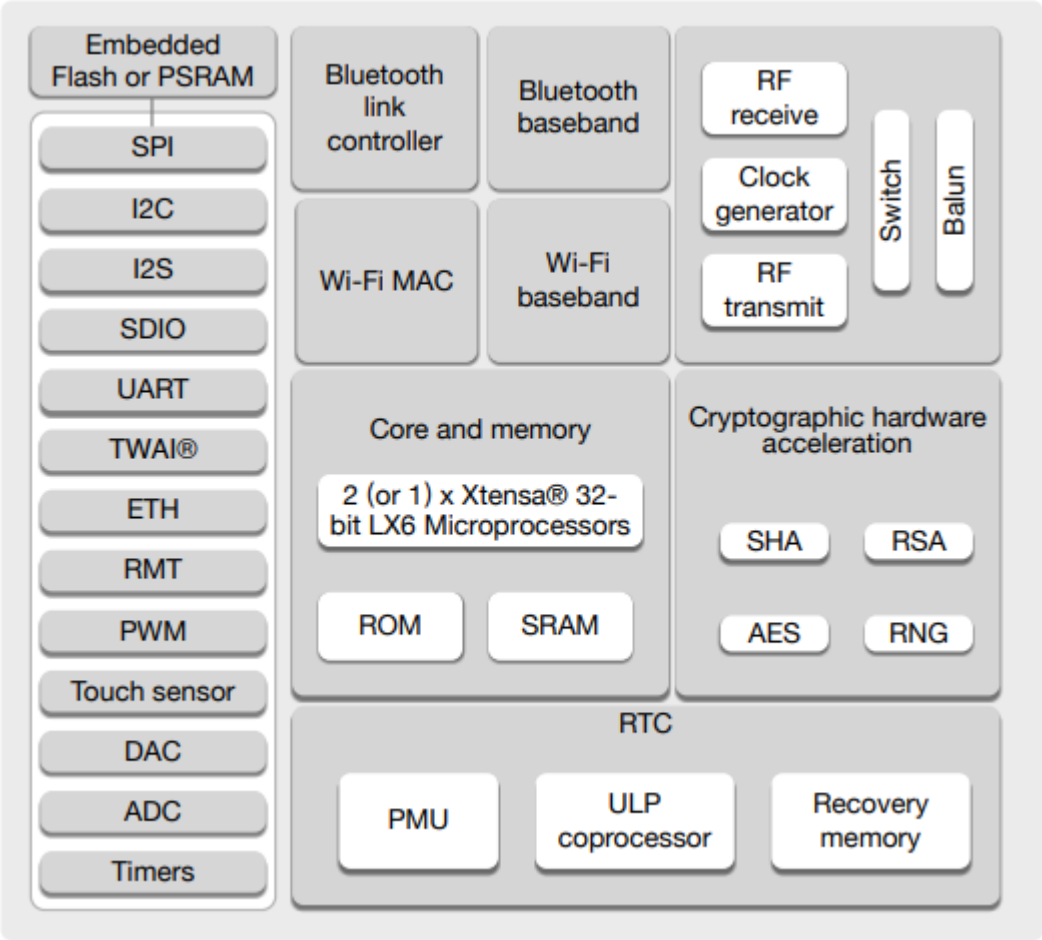
1.5 MODULO ESP32

O ESP32 é um SoC (*System on Chip* / Sistema em um Chip) que oferece conectividade (Wi-Fi, Bluetooth, com frequência de 2,4 GHz), poder computacional

(CPU + memórias), Entradas/Saídas (I/Os), Relógio em tempo real (RTC), suporte à diversas comunicações (I2C, I2S, SPI e etc.), como também suporte à operação *Low Power* e blocos de *hardware* dedicados à segurança (BERTOLETI, 2019).

O ESP32 por conter módulo Wi-Fi já embarcado no seu chip é uma opção poderosa e completa para o objetivo deste trabalho, podendo ser visto seu diagrama de blocos na figura 3 abaixo.

Figura 3 – Diagrama em blocos do ESP32.



Fonte: (ESPRESSIF SYSTEMS, 2021, p.12).

1.6 WI-FI

É uma rede de comunicação sem fio lançada em 1997 baseada nos padrões de transmissão e codificação IEEE 802.11, sendo bastante popular, pois está presente nos mais diversos lugares, fazendo parte do cotidiano de casas, escritórios, indústrias, lojas comerciais e até espaços públicos das cidades. O Wi-Fi possui algumas vantagens, como alcance de conexão e vazão, o que o torna adequado para navegação na Internet

em dispositivos móveis, como *smartphones* e *tablets*. A principal desvantagem do Wi-Fi é o maior consumo de energia, quando comparado com outras tecnologias de comunicação sem fio (SANTOS et al., 2016).

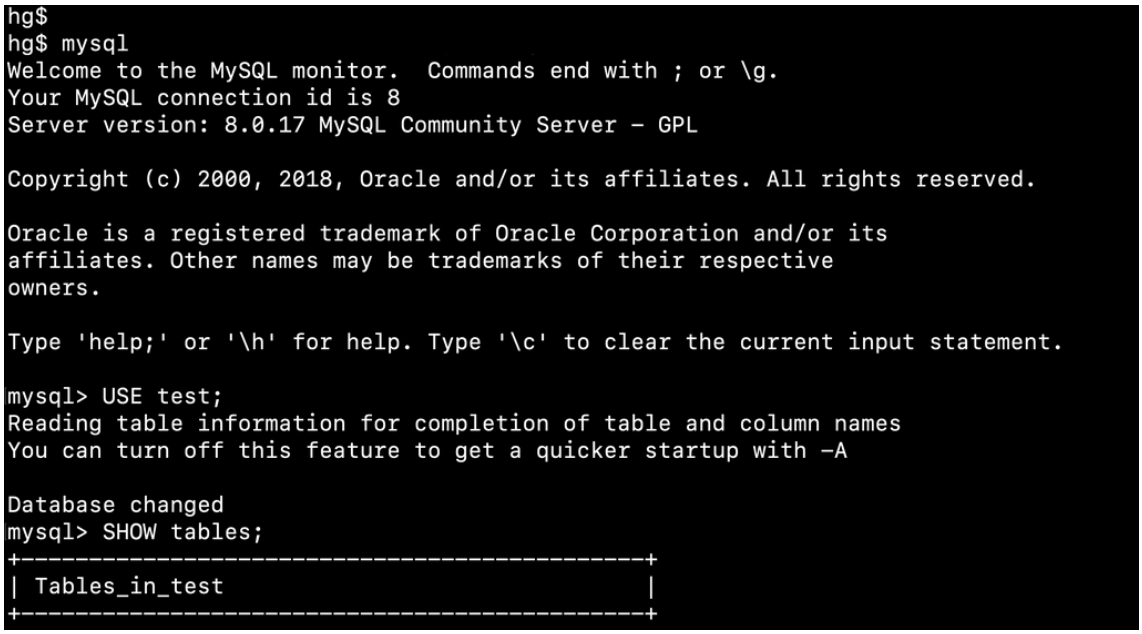
1.7 COMUNICAÇÃO SERIAL UART

Um receptor-transmissor assíncrono universal (UART) é um dispositivo de *hardware* de computador para comunicação serial assíncrona em que o formato de dados e as velocidades de transmissão são configuráveis. Ele envia bits de dados um a um, do menos significativo ao mais significativo, enquadrado por bits de início e fim para que o tempo preciso seja tratado pelo canal de comunicação. Os níveis de sinalização elétrica são controlados por um circuito driver externo à UART (GORDON, 1978)

1.8 MYSQL

MySQL é um sistema de gerenciamento de banco de dados relacional (RDBMS) de código aberto. Seu nome é uma combinação de "My", o nome da filha do cofundador Michael Widenius, My, e "SQL", a abreviação de Structured Query Language. Um banco de dados relacional organiza os dados em uma ou mais tabelas de dados nas quais os dados podem ser relacionados entre si; essas relações ajudam a estruturar os dados. SQL é uma linguagem que os programadores usam para criar, modificar e extrair dados do banco de dados relacional, bem como controlar o acesso do usuário ao banco de dados. Além de bancos de dados relacionais e SQL, um RDBMS como o MySQL trabalha com um sistema operacional para implementar um banco de dados relacional no sistema de armazenamento de um computador, gerencia usuários, permite o acesso à rede e facilita o teste de integridade do banco de dados e a criação de backups (MYSQL, 2020). Na figura 4 é possível ver um servidor MySQL mostrando as *tables* existentes.

Figura 4 – MySQL data-base.



Fonte: MySQL, 2022.

1.9 SENSORES

São responsáveis pela captação das informações do ambiente e transformação destas para sinais elétricos, as quais serão interpretadas pelas placas microcontroladas. Os sensores são elaborados para reconhecer uma determinada informação do ambiente, existindo assim diversos tipos de sensores, como por exemplo sensores de luminosidade, temperatura e umidade, entre outros (WENDLING, 2010).

1.9.1 SENSOR DE TEMPERATURA E UMIDADE DO AR

O sensor a ser utilizado no presente trabalho é o módulo medidor de temperatura e umidade DHT11, exibido na Figura 5, apresentando um sensor resistivo de umidade e componentes com coeficiente negativo de temperatura (*Negative Temperature Coeficient* - NTC). Suas principais características são baixo custo, estabilidade térmica, robustez, resposta rápida e tamanho reduzido (12x22,5x5mm) (THOMSEN, 2013).

A transmissão das grandezas físicas mensuradas é realizada de maneira digital serial por um único canal, processada por seu interno microcontrolador de 8 bits de alta performance. Sua faixa de operação de temperatura está entre 0^{o} C a 50^{0} C, com precisão de ±2^{o} C e de 20% a 90% com precisão de ±5% de umidade relativa do ar. Suas principais

aplicações são conforto térmico ambiental interior, teste e inspeção de equipamentos e reguladores de umidade (THOMSEN, 2013).

Figura 5 – Sensor DHT11.

Fonte: (THOMSEN, 2013).

1.9.2 SENSOR DE QUALIDADE DO AR

O módulo sensor MQ-135, mostrado na figura 6, possui um sensor sensível aos gases benzeno, álcool, fumaça e dióxido de carbono (CO_2). Dentre esses, no protótipo a ser desenvolvido será utilizado apenas a detecção do CO_2. O sensor possui vida útil longa, baixo custo, resposta rápida e estabilidade nas leituras (HANWEI, 201-a), sendo aquecido, internamente, por uma tensão elétrica de 5 V. Em seu manual, a faixa de medição de concentração que pode ser medida não é apresentada (STEFAN, et al., 2020).

Figura 6 – Imagem do módulo sensor MQ-135.

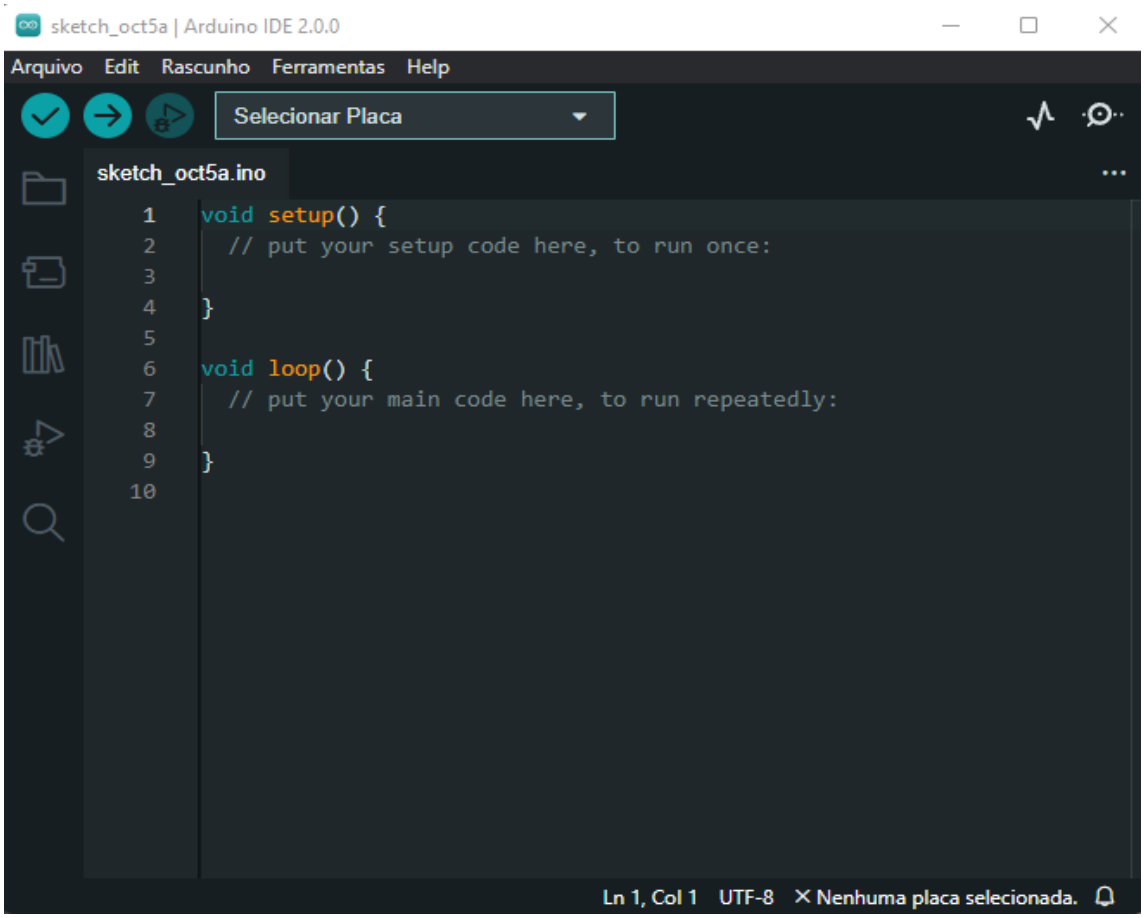
Fonte: (STEFAN, 2020).

1.10 SOFTWARES

1.10.1 ARDUINO IDE

O Arduino IDE é um Ambiente Integrado de Desenvolvimento (IDE - Integrated Development Environment) utilizado para escrever o código na placa. No Arduino a 18 linguagem de programação utilizada é baseada em C/C++ e oferece bibliotecas de programação para diversos projetos. O mais interessante é que o Arduino possibilita a transferência de *firmware* via USB. O ambiente de desenvolvimento da programação do Arduino é composto de uma IDE, que foi criada em Java, onde é baseada no *Processing* e na linguagem do próprio Arduino, que por fim é proveniente do Wiring, que é baseada em C/C++ (ARDUINO, 2020). A figura 7 representa esse ambiente, com um exemplo de código da própria biblioteca do Arduino, demonstrado também a sua linguagem de programação.

Figura 7 - Ambiente de desenvolvimento Arduino (IDE).



Fonte: autoria própria, 2022

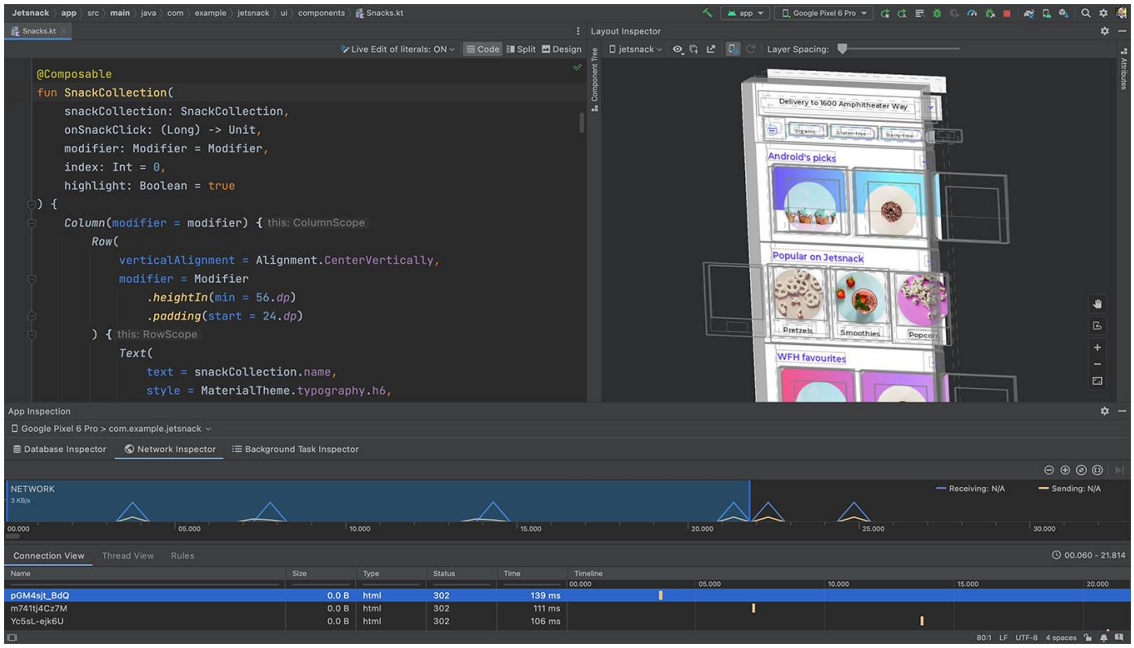
1.10.2 GOOGLE SHEETS

O *Google Sheets* é um programa de planilhas incluído como parte do pacote gratuito de Editores de Documentos Google baseado na Web oferecido pelo Google. O serviço também inclui Google Docs, Google Slides, Google Drawings, Google Forms, Google Sites e Google Keep. O Planilhas Google está disponível como aplicativo da web, aplicativo móvel para Android, iOS, Windows, BlackBerry e como aplicativo de desktop no Chrome OS do Google.

1.10.3 ANDROID STUDIO

O Android Studio é o ambiente de desenvolvimento integrado (IDE) oficial para o sistema operacional Android do Google, construído no *software* IntelliJ IDEA da JetBrains e projetado especificamente para desenvolvimento Android. Ele está disponível para *download* em sistemas operacionais baseados em Windows, macOS e Linux. É um substituto para o Eclipse Android Development Tools (E-ADT) como o IDE principal para o desenvolvimento de aplicativos Android nativos. Na figura 8, é mostrado o ambiente de trabalho do Android Studio.

Figura 8 - Ambiente de desenvolvimento Android Studio (IDE).



Fonte: autoria própria, 2022

2 MÉTODOS E MATERIAIS

Neste capítulo é mostrado como se deu o andamento do projeto desde a pesquisa e instalação das bibliotecas ESPWiFi, MySQL Connector e DHT sensor até a comunicação com o servidor MySQL e a amostragem dos dados via aplicativo Android. O sistema consiste em aferir a temperatura, umidade, ponto do orvalho e qualidade do ar em data centers e assim, enviar os dados para o servidor onde serão organizados e coletados posteriormente por um aplicativo Android que mostrará um gráfico com os dados coletados ao usuário.

O sistema foi desenvolvido nas seguintes etapas. Na primeira etapa foi realizada uma revisão bibliográfica em livros, artigos, normas técnicas e periódicos especializados para o levantamento do estado da arte das principais tecnologias aplicadas no projeto. Após essa etapa foram realizadas pesquisas na área de programação, dando ênfase na programação C++, para melhor entendimento e uso das bibliotecas do dispositivo ESPWifi, MySQL Connector e DHT sensor. Foi acessado o *site* do fabricante para *download* da biblioteca e instalação no IDE Arduino.

Na segunda etapa, se iniciou o desenvolvimento do protótipo de aferição dos parâmetros do ar. Este protótipo contém uma placa Arduino alimentada por uma fonte de 5V ligada aos 2 sensores, de temperatura e umidade e de qualidade do ar, que fará a coleta dos dados e organização em um só pacote para envio via UART para o ESP32.

Este circuito será mais detalhado posteriormente.

Na terceira etapa foram feitos testes de calibração do sensor e testes de envio dos dados. Para esse teste foi utilizado um código de teste no Arduino para coleta dos dados em condições normais do ambiente transmitindo os dados para a placa ESP32 que eram mostrados no monitor serial do software Arduino IDE.

Na quarta etapa realizou-se a comunicação do dispositivo ESP32 com um servidor MySQL onde os dados foram organizados em colunas numa tabela. Para exibir os dados foi desenvolvido um aplicativo que coleta esses dados e mostra eles em forma de um gráfico de linhas para o usuário.

2.1. MATERIAIS UTILIZADOS PARA O PROTÓTIPO

Para a realização dos testes dos sensores e rotina de calibração, foram necessários os seguintes itens:

1 Arduino Uno R3

1 esp32 devkit v1

1 protoboard para encaixe do módulo

1 Módulo DHT11 Sensor De Temperatura

1 Sensor de Qualidade do Ar MQ-135

10 jumpers

3 IMPLEMENTAÇÂO DO PROJETO

Este capítulo apresenta a realização do projeto em questão. Foi desenvolvida a programação no ambiente Arduino IDE, um protótipo de aferição de parâmetros do ar. Também se mostra a configuração do banco de dados e desenvolvimento do aplicativo para mostrar os resultados obtidos.

São apresentados neste capítulo os seguintes tópicos:

a) instalação das bibliotecas na Arduino IDE;

b) calibração dos sensores e desenvolvimento do protótipo de aferição;

c) desenvolvimento do código das placas;

d) configuração do banco de dados;

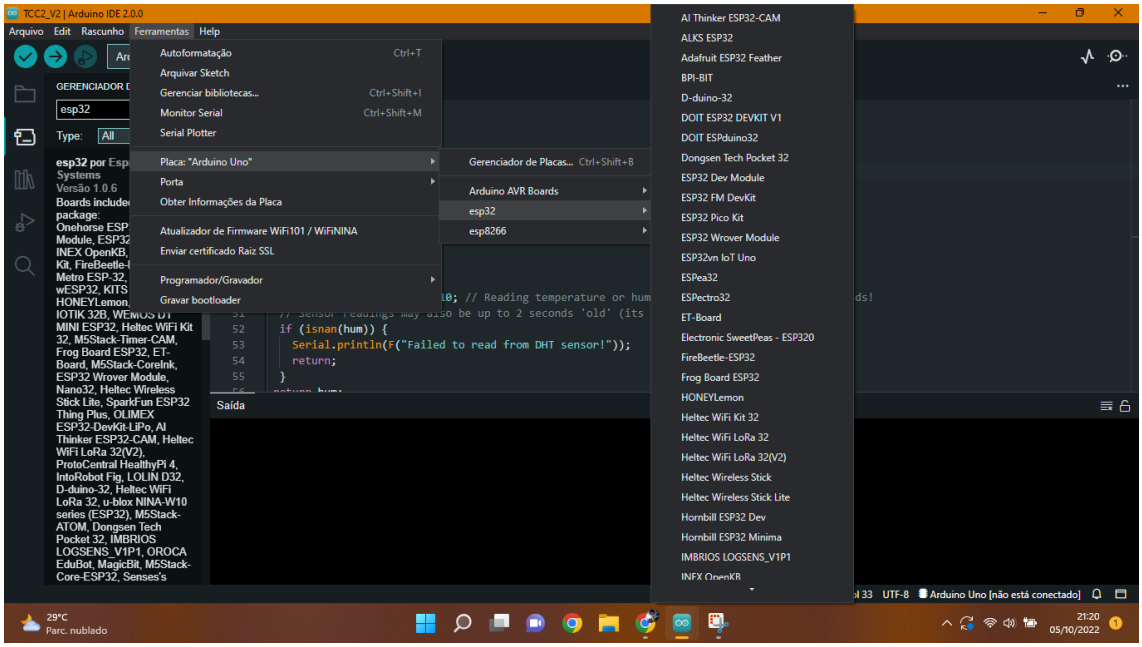
e) desenvolvimento do aplicativo Android.

3.1. INSTALAÇÃO DAS BIBLIOTECAS NA ARDUINO IDE

Para o desenvolvimento deste trabalho foram instaladas algumas bibliotecas específicas que serão usadas na programação do módulo ESP32 devkit.

Por padrão o ambiente de desenvolvimento Arduino não possui este módulo em sua biblioteca de placas. Para incluí-lo é necessário acessar a aba Arquivo/Preferências e no campo URLs Adicionais para Gerenciadores de placas incluir o endereço *https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json.* A partir daí, as placas da família ESP32 estão disponíveis para instalação digitando "esp 32" no gerenciador de placas, onde aparece a biblioteca "esp 32 by Espressif" onde, após clicar no botão instalar, fica disponível como mostrado na figura 9.

Figura 9 - Adição da biblioteca da placa Esp32



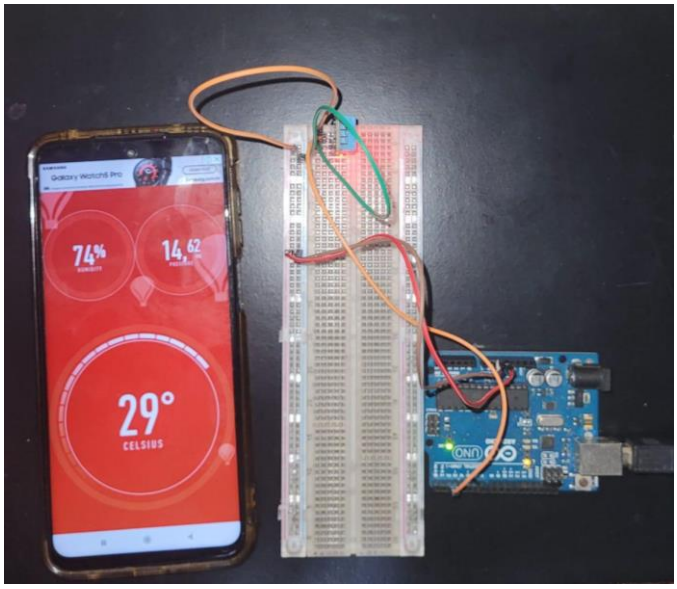
Fonte: autoria própria, 2022

3.2. CALIBRAÇÃO DOS SENSORES E DESENVOLVIMENTO DO PROTÓTIPO DE AFERIÇÃO

Nesta etapa, foram iniciados testes de calibração com os sensores presentes nesse projeto, o DHT-11 e MQ-135 e desenvolvimento do protótipo de aferição.

Utilizando somente a placa Arduino para os testes de calibração, foi conectado a ela o sensor DHT-11 para medição da temperatura ambiente e umidade ambiente próximo a um medidor para averiguação dos resultados. Na figura 10, é mostrado o teste de calibração efetuado.

Figura 10 - Teste de calibração de temperatura e umidade com comparativo de ambiente

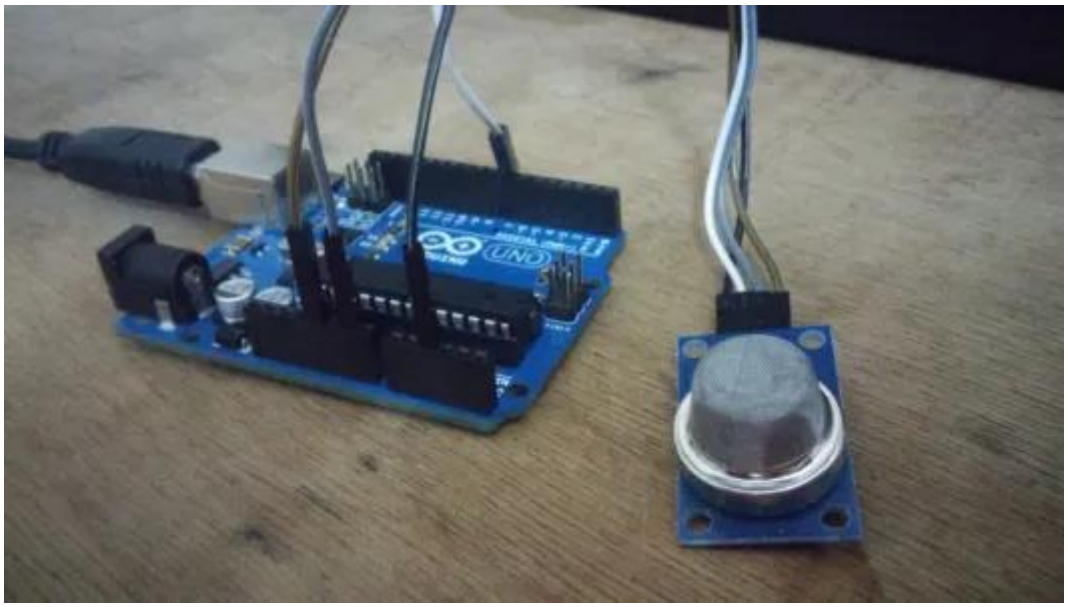


Fonte: autoria própria, 2022

Nisso foi verificado que o sensor obtinha uma resposta correta em relação a temperatura e umidade relativa.

O sensor MQ-135 também foi conectado a placa Arduino para realização de calibração, como esse sensor calcula a qualidade relativa do ar em PPM (Partes por milhão) pode-se notar que o mesmo atende as condições por mostrar que a qualidade do ar ambiente está dentro dos conformes (100-200 PPM) de acordo com o Ministério do Meio Ambiente (2009). Na figura 11, é mostrado o circuito montado para testagem.

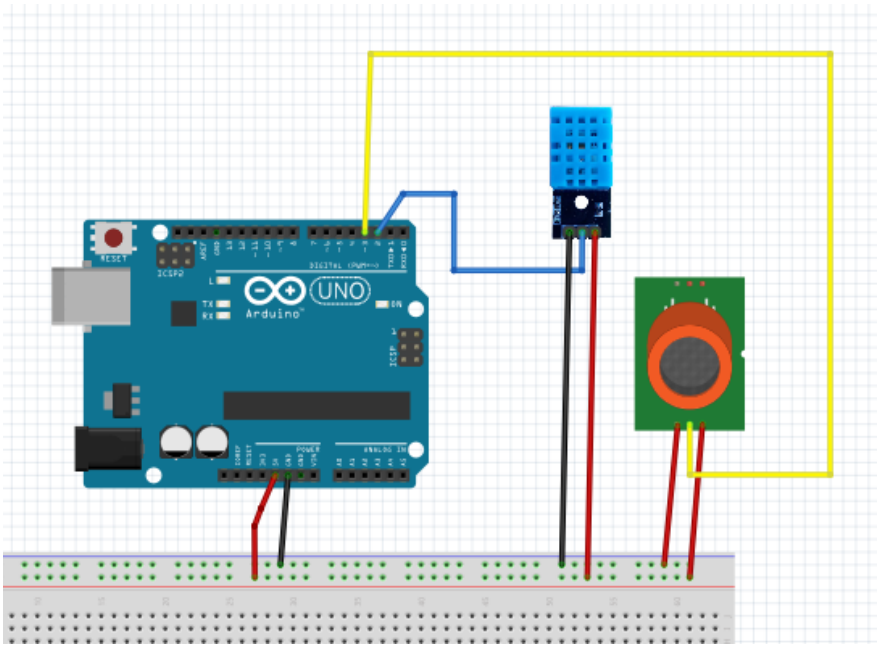
Figura 11 - Teste de calibração de qualidade do ar.



Fonte: autoria própria, 2022.

Após a calibração dos sensores, foi iniciada a montagem do protótipo conforme esquema da figura 12.

Figura 12 - Esquema do circuito dos sensores.

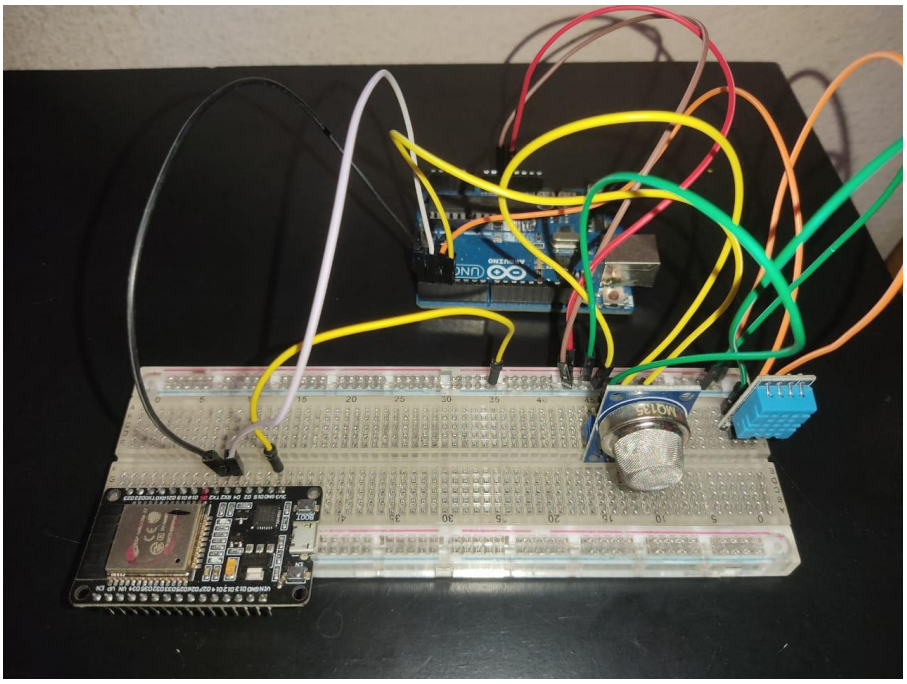


Fonte: autoria própria, 2022.

Os dois sensores são ligados ao *protoboard* que por sua vez é ligado ao Arduino, sendo na alimentação positiva, GND e entradas analógicas para a placa adquirir os dados,

e os pinos Tx e Rx são ligados na placa do Esp32 que realizará o envio para o banco de dados. Na figura 13, é mostrado como ficou o circuito.

Figura 13 - Protótipo montado.

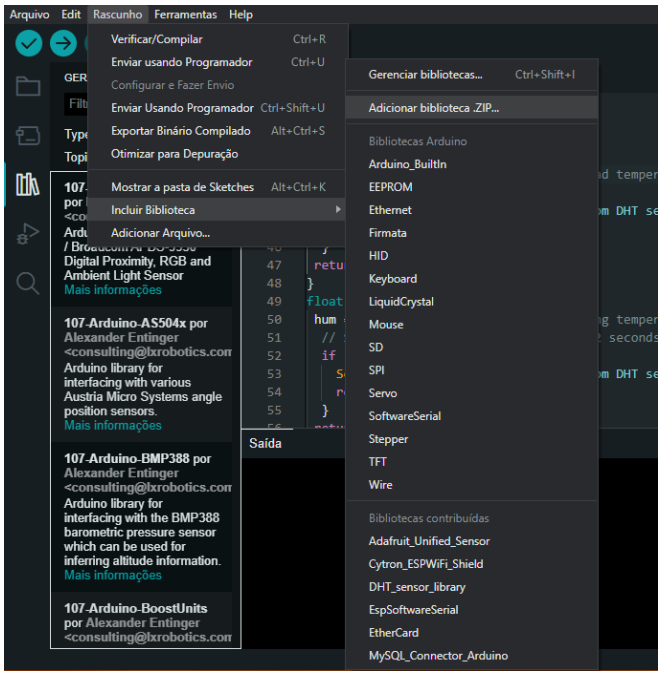


Fonte: autoria própria, 2022.

3.3. DESENVOLVIMENTO DO CODIGO DAS PLACAS

Para desenvolvimento do código de coleta de dados no Arduino, foi necessário a instalação da biblioteca do sensor DHT, já presente na plataforma do Arduino. Já no código de recebimento de dados do Arduino e envio desses dados para o servidor da placa Esp 32 devkit, foi necessário o uso de 2 bibliotecas, a Wifi.h, que já está presente no gerenciador do Arduino IDE, e a MySQL_Connection.h que, em conjunto com a MySQL_Cursor.h, fazem parte de um conjunto de bibliotecas para conexão com servidores MySQL online, pode ser obtido através de um arquivo zip no site *https://github.com/ChuckBell/MySQL_Connector_Arduino.* Ao baixar a pasta com arquivos compactados, abriu-se a IDE Arduino, no menu Sketch/incluir biblioteca, foi selecionado incluir biblioteca zip e depois direcionado para pasta onde se encontra o arquivo zip como pode ser visto na figura 14.

Figura 14 - Adição da biblioteca MySQL_Connector_Arduino no Arduino IDE.

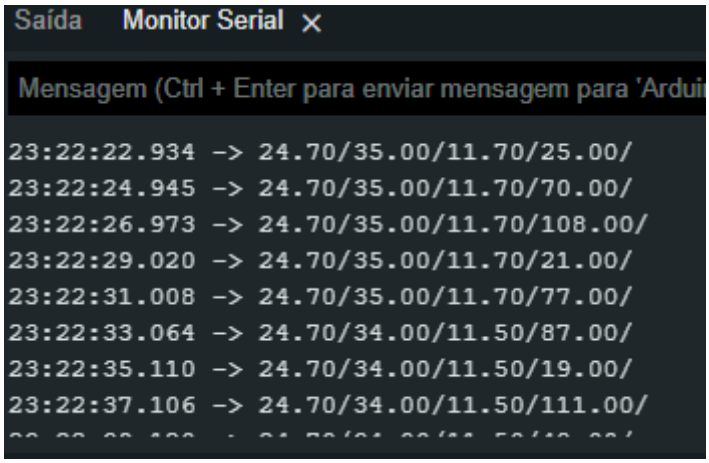


Fonte: autoria própria, 2022.

Com isso desenvolvimento pode ser iniciado, sendo primeiramente desenvolvido o código no Arduino, primeiramente definindo o pino digital em que será conectado o sensor DHT, que é representado por *#define DHTPIN 2* que por padrão vem definido como pino 2, e definido o tipo do sensor DHT utilizado, representado por *#define DHTTYPE DHT11* no caso deste trabalho o DHT11.

Para fazer a leitura dos dados, foram criados métodos para cada parâmetro do ar a ser medido, temperatura() e umidade() que utilizam a biblioteca do sensor dht com os métodos readTemperature() e readHumidity() para realizar a leitura dos parâmetros, o ponto do orvalho, por usa vez, é calculado a partir desses dois parâmetros seguindo a seguinte simplificação $T_{po} = T - \frac{100 - UR}{5}$ onde Tpo é a temperatura de ponto do orvalho, T é a temperatura atual e UR é a umidade relativa do ar, sendo utilizada no método pontoDoOrvalho(). Já a qualidade do ar é adquirida no método qualidade () que faz a leitura analógica do pino 3 do Arduino, onde a saída do sensor MQ-135 está conectada. Já no método loop (), todos esses parâmetros são lidos e transformados em string pelo método dtostrf(), para assim facilitar a concatenação em uma só string para envio via UART pelo método sprintf(). Na figura 15, é mostrada a resposta no monitor serial do Arduino.

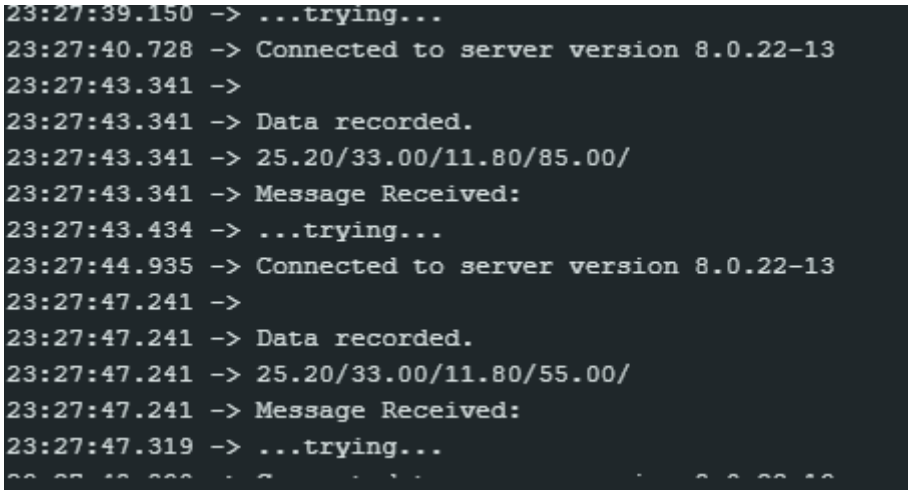
Figura 15 - Monitor serial do Arduino.



Fonte: autoria própria, 2022.

Já no código do Esp32, foi necessário definir variáveis para usuário e senha da rede ssid [] e pass[] e usuário e senha do servidor com a data-base MySQL user[] e password[] e o comando de adição na tabela respectiva em INSERT_DATA[]. Foi criado o método leitura () para realização da leitura dos dados enviados pelo Arduino via UART e o método verificaWifi() para realizar a conexão com a rede Wi-fi informada e o método enviaDados para realizar o envio dos dados para o servidor utilizando o método cur_mem->execute(query).

Figura 16 - Monitor serial do Esp32.



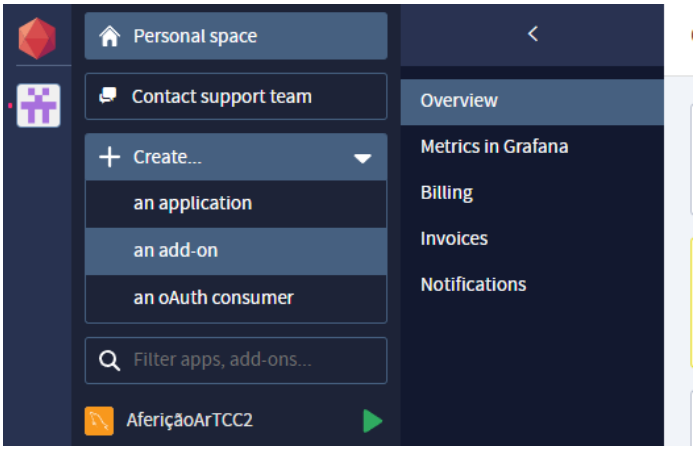
Fonte: autoria própria, 2022.

3.4. CONFIGURAÇÃO DO BANCO DE DADOS

Para utilização do banco de dados MySQL, foi utilizado o serviço de cloud

(nuvem) do Clever Cloud, que possui diversas ferramentas de armazenamento em banco de dados, além de ser gratuito. Com isso, foi iniciada a configuração do servidor onde foi criada uma conta e, após confirmada, foi criado um “add-on” que foi usado como banco de dados MySQL, conforme figura 17.

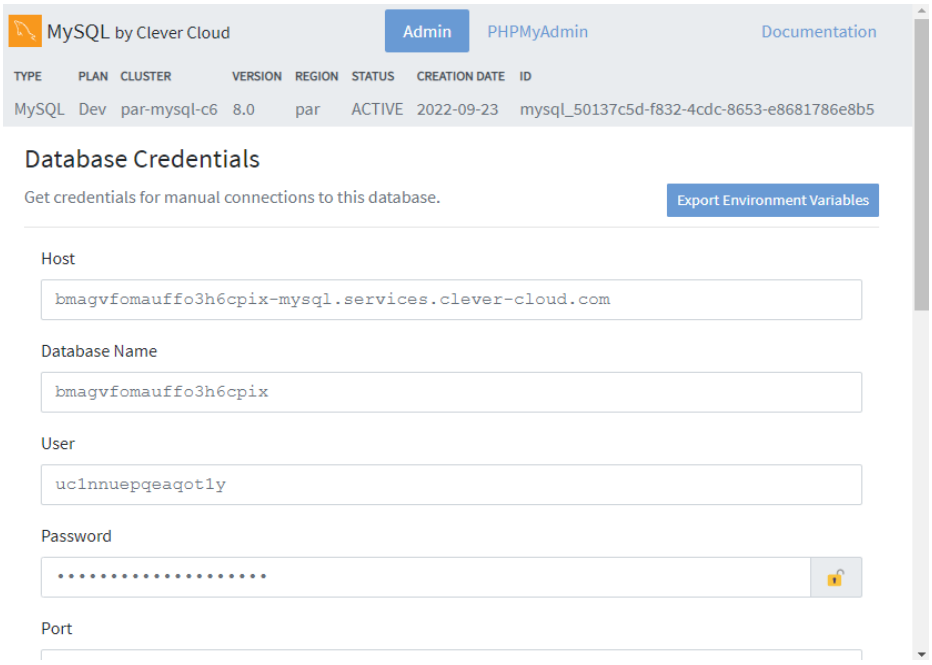
Figura 17 - Adição de um add-on no clever cloud.



Fonte: autoria própria, 2022.

Com isso, foi criado o banco de dados “AferiçãoArTCC2” para armazenamento dos dados e assim criados todos os dados como nome do host, usuário, senha, conforme figura 18.

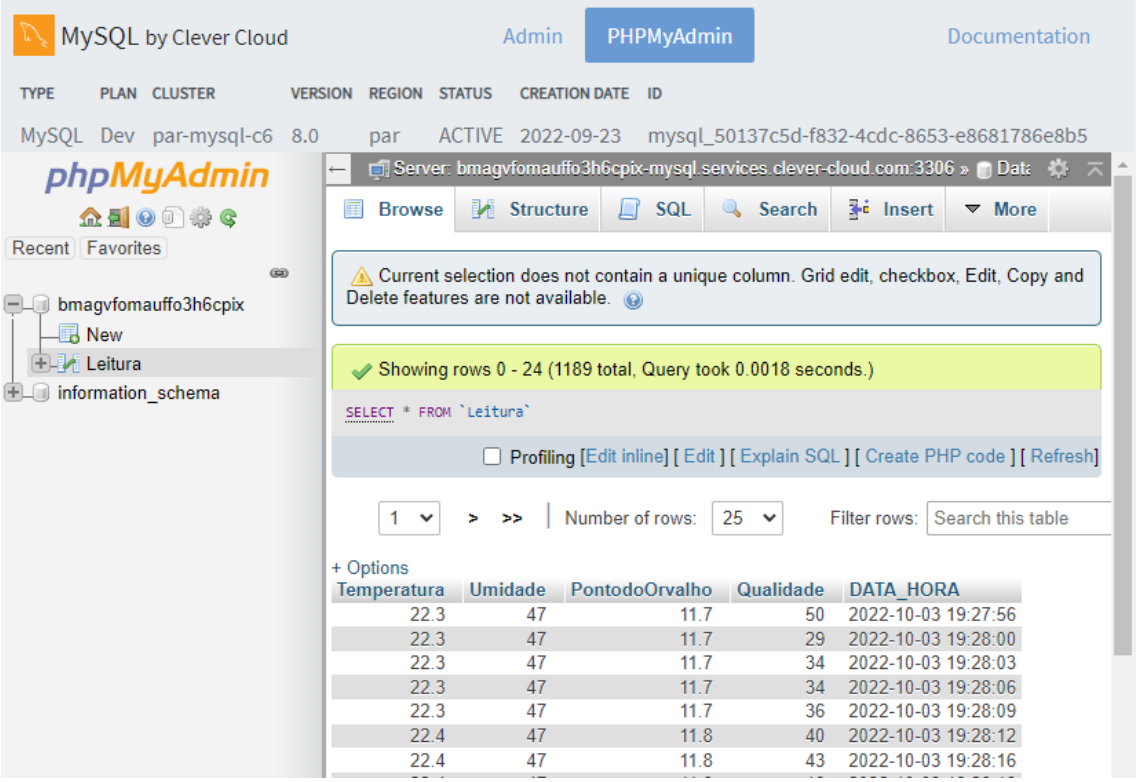
Figura 18 - Informações da data-base MySQL criada.



Fonte: autoria própria, 2022.

Após isso, houve a criação da tabela para armazenar os dados com 5 colunas, estas sendo Temperatura, Umidade, PontoDoOrvalho, Qualidade e data_hora para armazenamento dos dados enviados pelo Esp32 e a data e hora em que o dado foi armazenado, conforme figura 19.

Figura 19 - Criação da tabela para armazenar os dados.



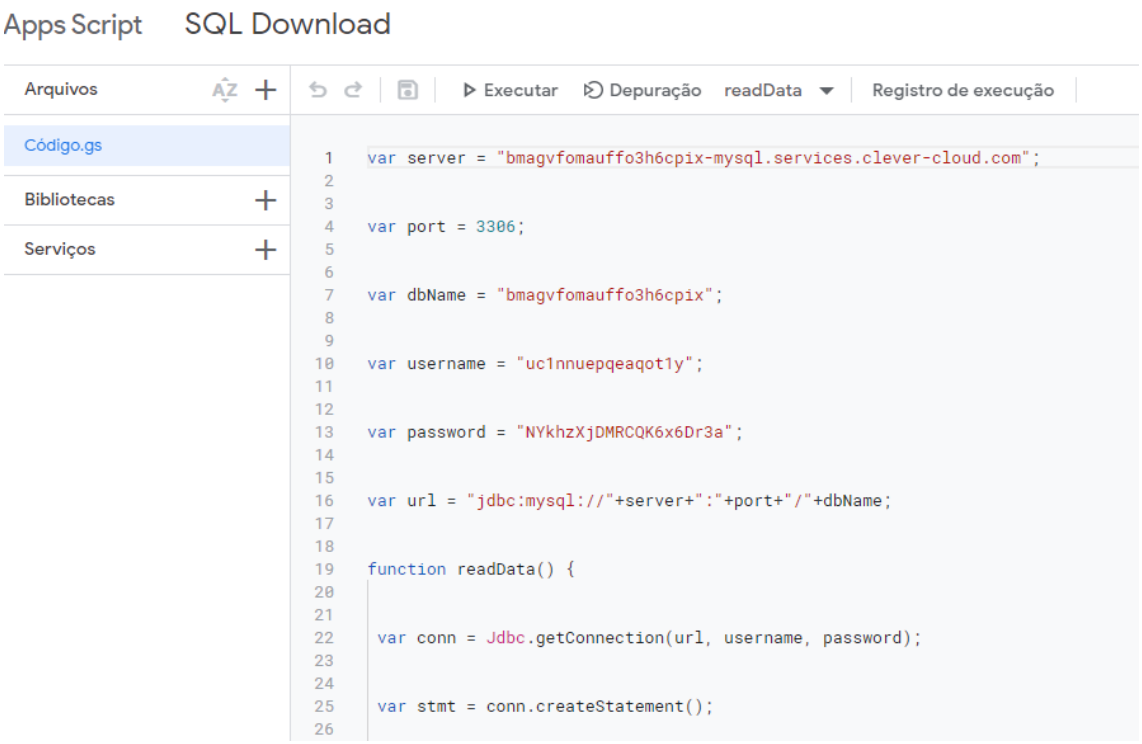
Fonte: autoria própria, 2022.

3.5. DESENVOLVIMENTO DO APLICATIVO ANDROID

O método escolhido para mostrar os resultados da leitura para o usuário foi um aplicativo Android, por conta da facilidade de acesso a um telefone celular.

Para essa etapa foi necessário o uso de uma ferramenta para reunir a informações do servidor MySQL para acesso do aplicativo logo foi criado uma planilha no Google Sheets, uma ferramenta do google para planilhas online, e nela foi desenvolvido um código para adquirir os dados do servidor utilizando a função App Script sendo criado o programa código.gs, conforme figura 20.

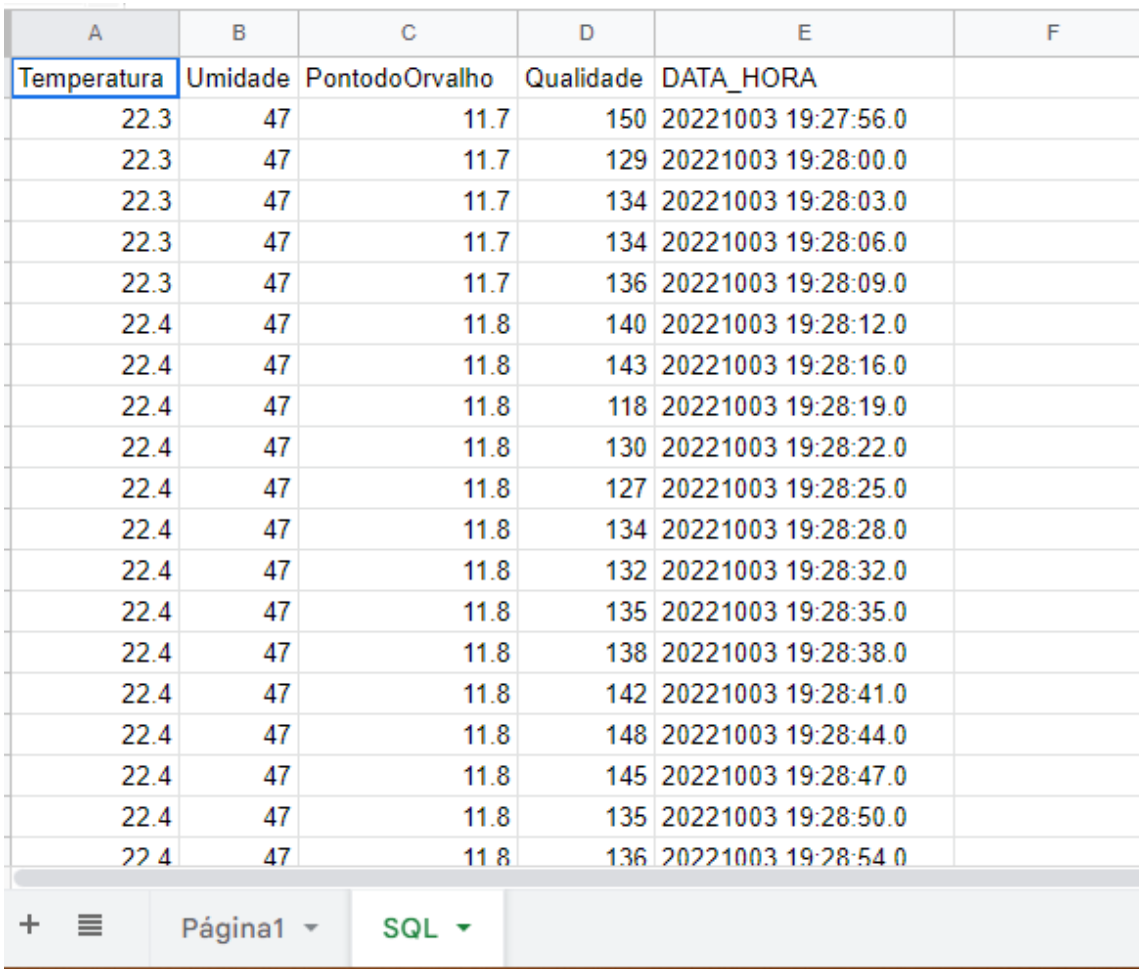
Figura 20 - Código para adquirir os dados do Clever Cloud.



Fonte: autoria própria, 2022.

Neste código foram adicionadas variáveis para todas as informações do servidor e da data-base, além da organização da planilha e um método para download automático a cada 5 minutos para manter os dados atualizados, o resultado disso pode ser observado na figura 21.

Figura 21 - Planilha com os dados adquiridos do Clever Cloud.



A	B	C	D	E	F
Temperatura	Umidade	PontodoOrvalho	Qualidade	DATA_HORA	
22.3	47	11.7	150	20221003 19:27:56.0	
22.3	47	11.7	129	20221003 19:28:00.0	
22.3	47	11.7	134	20221003 19:28:03.0	
22.3	47	11.7	134	20221003 19:28:06.0	
22.3	47	11.7	136	20221003 19:28:09.0	
22.4	47	11.8	140	20221003 19:28:12.0	
22.4	47	11.8	143	20221003 19:28:16.0	
22.4	47	11.8	118	20221003 19:28:19.0	
22.4	47	11.8	130	20221003 19:28:22.0	
22.4	47	11.8	127	20221003 19:28:25.0	
22.4	47	11.8	134	20221003 19:28:28.0	
22.4	47	11.8	132	20221003 19:28:32.0	
22.4	47	11.8	135	20221003 19:28:35.0	
22.4	47	11.8	138	20221003 19:28:38.0	
22.4	47	11.8	142	20221003 19:28:41.0	
22.4	47	11.8	148	20221003 19:28:44.0	
22.4	47	11.8	145	20221003 19:28:47.0	
22.4	47	11.8	135	20221003 19:28:50.0	
22.4	47	11.8	136	20221003 19:28:54.0	

Fonte: autoria própria, 2022.

Com isso, foi possível adquirir os dados do servidor no aplicativo com mais facilidade para construção de gráficos e amostragem de resultados. Logo foi iniciado o desenvolvimento do aplicativo, na qual foi usada uma API (Interface de Programação de Aplicação) para acesso que é a Sheets API disponibilizada pela Google. Com ela, após cadastro nas credenciais no google, foi possível fazer o *download* de toda a informação necessária para amostragem ao usuário, porém para ser uma forma mais simplificada e fácil de ser compreendida, foram montados gráficos para cada parâmetro do ar medido, podendo ser acessado diretamente na tela do aplicativo.

Figura 22 - Aplicativo mostrando gráfico a partir da planilha.

Fonte: autoria própria, 2022.

Além disso, foi desenvolvido um método para verificar os máximos e mínimos dos parâmetros para ser mostrado também na tela do celular quando fosse atualizado o gráfico.

Figura 23 - Aplicativo mostrando gráfico e máximos e mínimos da planilha.

Fonte: autoria própria, 2022.

E por fim, foi desenvolvido uma notificação que seria lançada caso um dos parâmetros saísse dos padrões normais esperados para um data center, sendo cada notificação montada para cada parâmetro e avisando o que saiu do padrão.

Figura 24 - Aplicativo mostrando notificação quando temperatura fica fora do padrão.

Fonte: autoria própria, 2022.

4 RESULTADOS OBTIDOS

Este capítulo está separado em 2 seções onde são mostrados os resultados referentes aos testes com o protótipo de aferição em um data center e verificação dos dados aferidos nesse ambiente e os resultados mostrados ao usuário pelo aplicativo.

4.1. TESTES COM PROTÓTIPO DE AFERIÇÃO EM UM DATA CENTER

Nos testes realizados em um data center, localizado no Instituto de Desenvolvimento Tecnológico (INDT), onde foi realizada a aquisição dos dados em 4 pontos do local, para comparativo de resultados em diferentes pontos do local sendo estes decididos conforme figura 25.

Figura 25 - Planta baixa do data center do INDT.

Fonte: autoria própria, 2022.

Foi medido durante 1 hora em cada ponto para comparativo dos parâmetros nesses pontos, sendo mostrado nos quadros 1 e 2, o range e a média dos parâmetros medidos.

Quadro 1 - Comparativo dos ranges dos pontos A à D.

	Ponto A	Ponto B	Ponto C	Ponto D	Unidade
Temperatura	20,2 - 21,6	21,2 - 22,1	22,1 - 23,2	22,3 - 23,3	ºC
Umidade	45 - 48	40 - 49	41 - 46	46 - 50	%
Ponto do Orvalho	7,9 - 9,2	8,9 - 10,3	10,3 - 12,3	11,7 - 13,3	ºC
Qualidade do ar	112 - 150	133 - 146	102 - 165	100 - 150	PPM

Fonte: autoria própria, 2022.

Quadro 2 - Comparativo das médias dos pontos A à D.

	Ponto A	Ponto B	Ponto C	Ponto D	Unidade
Temperatura	21,20	21,80	22,40	22,90	ºC
Umidade	47	47	43,5	48	%
Ponto do Orvalho	8,6	9,4	11,1	12,5	ºC
Qualidade do ar	120	142	148,5	135	PPM

Fonte: autoria própria, 2022

Os resultados adquiridos podem ser corroborados com o medidor de temperatura encontrado na própria sala no controle de ar, mostrado na figura 26.

Figura 26 - Controle de ar do data center.

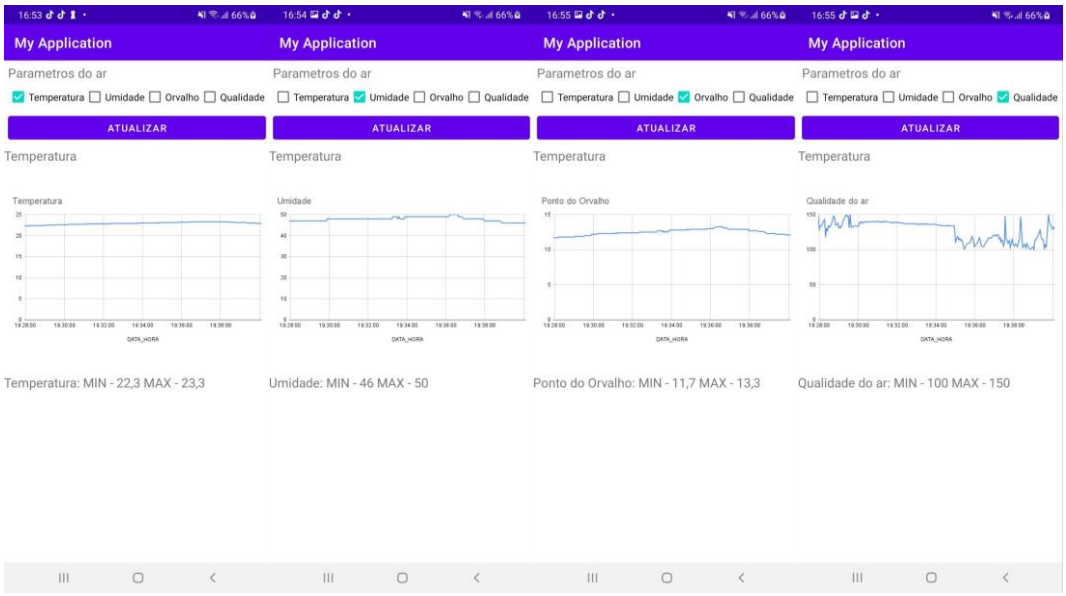
Fonte: autoria própria, 2022.

Pode-se notar que a partir dessa tabela que os parâmetros medidos são diferentes nos pontos da sala, reforçando a importância do monitoramento em vários pontos críticos de um data center, porém não saindo dos parâmetros recomendados para um data center.

4.2. RESULTADOS MOSTRADOS AO USUÁRIO PELO APLICATIVO

Os resultados podem ser observados na figura 27, onde pode ser observado o comportamento dos parâmetros do ar neste local nos pontos escolhidos para medição, além de seus maiores e menores valores.

Figura 27 - Imagens das respostas do aplicativo



Fonte: autoria própria, 2022

Esses resultados mostram correspondência com as tabelas 1 e 2 mostradas, na qual pode se afirmar que os dados mostrados ao usuário no aplicativo são verídicos e condizentes com os medidos pelos sensores.

CONCLUSÃO

Nesta pesquisa, foram efetuadas revisões bibliográficas dos assuntos abordados neste documento, como Arduino e Esp32 até Wi-fi, para que fosse possível o conhecimento do estado da arte atual quanto a essas tecnologias, e também dos próprios parâmetros a serem medidos pelos sensores para adquirir os parâmetros do ar.

No teste no *data center*, foi levado o protótipo até os quatro pontos definidos previamente para adquirir os parâmetros do ar naqueles pontos e medidos durante 1 hora cada, na qual obteve êxito em conseguir esses dados corretos visto que a sala possuía termômetro no controle de ar da parede e salvá-los no data-base MySQL.

Já no teste de exibição dos resultados medidos ao usuário, obteve-se sucesso ao adquirir os dados da data-base via google sheets e, utilizando o aplicativo desenvolvido, pode se organizar os dados e exibir ao usuário o gráfico e os máximos e mínimos dos parâmetros.

Com base nesses resultados, fica claro que o protótipo é capaz de adquirir os dados dos parâmetros do ar em data centers de maneira satisfatória, salvando-os em um data-base MySQL e podendo ser acessado e visualizado pelo usuário de maneira fácil por um aplicativo Android, tendo assim cumprido as expectativas e requisitos propostos por este trabalho.

Para trabalhos futuros é sugerido um estudo sobre o envio de dados pois, por conta da comunicação serial, pode acontecer de dados ficarem aglutinados para envio. Para isso, a recomendação seria a eliminação da placa Arduino do circuito, que, por conta dos sensores utilizados, se optou pela não retirada dele. Outra sugestão seria a extensão do trabalho para realizar além da aferição e monitoramento, realizar o controle dos parâmetros do ar com o controle dos aparelhos de controle de ar do data center, para assim, gerir as condições do ar nesse espaço.

REFERÊNCIAS

ARDUINO. **What is Arduino?** 2019. Disponível em: https://www.arduino.cc/en/Guide/Introduction. Acesso em: 1 outubro de 2022.

ASHRAE (American Society of Heating, Refrigerating and Air Conditioning Engineers), **Energy Standard for Data Centers**, Standard 90.4-2019.

BERTOLETI, Pedro**. Projetos com ESP32 e LoRa**. 1.ed. São Paulo: Instituto Newton C Braga, 2019.

CORREIA, Jornandes Jesus, Revista RBBA**: Definições de temperatura em fontes didáticas**. Vitória da Conquista: Ltc, 2017. 6v, p. 201-220.

ESPRESSIF SYSTEMS**. ESP32 Series Datasheet**. [S.l.], 2021. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em: 1 outubro de 2022.

ENERGIA E AMBIENTE, Plataforma da qualidade do ar, 2022. Disponível em: <http://energiaeambiente.org.br/qualidadedoar#:~:text=A%20qualidade%20do%20ar%20num, no%20tempo%20e%20no%20espa%C3%A7o> Acesso em: 1 de outubro de 2022

FONSECA, Erika Guimarães Pereira da; BEPPU, Fonseca Mathyan Motta. **Apostila Arduino**. Material Apresentado ao Programa de Educação Tutorial do Curso de Engenharia de Telecomunicações da Escola de Engenharia do Centro Tecnológico da Universidade Federal Fluminense. 2010. 23 p

GORDON, Bell C. Computer Engineering, **A DEC VIEW OF HARDWARE SYSTEMS DESIGN,** 1978, p. 54

HANWEI ELECTRONICS. **Technical Data MQ135** Gas Sensor. HWSENSOR. [201-a]. 3 p.

JRANK, **Dew Point,** 2018, disponível em https://science.jrank.org/pages/2038/Dew-Point.html Acesso em: 1 outubro de 2022.

LABOMAT, **What is Dew Point,** 2021. Disponível em: <https://labomat.eu/gb/corrosion-

testing-faq/708-what-is-dew-point-.html>.
Acesso em: 1 de outubro de 2022

MINISTERIO DO MEIO AMBIENTE, **Qualidade do Ar**, 2019 Disponivel em: <https://antigo.mma.gov.br/cidades-sustentaveis/qualidade-do-ar.html#:~:text=De%20uma%20forma%20geral%2C%20a,n%C3%A3o%20%C3%A0%20dispers%C3%A3o%20dos%20poluentes>. Acesso em: 1 outubro de 2022.

MYSQL, **Why MySQL?**, 2022. Disponivel em: <https://www.mysql.com/why-mysql/> Acesso em: 1 outubro de 2022.

RIBEIRO, Bruno Barros, **Monitoramento de temperatura, umidade e gases em data centers** Trabalho de Conclusão de Curso apresentado à Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, 2020

SANTOS, Bruno P.; SILVA, Lucas A. M.; CELES, Clayson S. F. S.; NETO, João B. Borges; PERES, Bruna S.; VIEIRA Marcos Augusto M.; VIEIRA, Luiz Filipe M.; GOUSSEVSKAIA, Olga N.; LOUREIRO, Antonio A. F**. Internet das Coisas: da Teoria à Prática**. Minas Gerais: Departamento de Ciência da Computação Universidade Federal de Minas Gerais (UFMG), 2016.

SEARS, F. ZEMANSKY, M. W. YOUNG, H. D**. Física 2: Mecânica dos Fluidos**, Calor, Movimento Ondulatório. 2. ed. Rio de Janeiro: Ltc, 1984. 4v, p. 385-387.

SILVA, Marco Aurélio Saminez da Aferição **e monitoramento remoto de temperatura e umidade em data center**. Trabalho de Conclusão de Curso apresentado à Universidade Federal do Maranhão, 2018.

STEFAN, Igor Alexandre et al. P**rotótipo de medidor de gases poluentes usando tecnologia de baixo custo,** revista principia, João Pessoa, ed.49 p. 31-49, 2020.

THOMSEN, Adilson. **Monitorando Temperatura e Umidade com o sensor DHT11**. 2013. Disponível em: https://www.filipeflop.com/blog/monitorando-temperaturae-umidade-com-o-sensor-dht11//. Acesso em: 1 outubro de 2022.

WENDLING, Marcelo. **Sensores V2.0**. 2010. 19 p. UNESP.

APÊNDICE A – CÓDIGO DO ARDUINO

```
/*************************************************************************
***
 UNIVERSIDADE DO ESTADO DO AMAZONAS
Projeto de conclusão de curso 2
Autor: Marcos Henrique Correa Lima
Este programa tem a finalidade de enviar os parâmetros medidos pelos sensores
a placa ESP32
*************************************************************************
*/

#include "DHT.h"

#define DHTPIN 2

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
int  qua;
float temp;
float hum;
float tpo;

void setup() {
  Serial.begin(9600);
  Serial.println(F("DHTxx test!"));

  dht.begin();
}

float temperatura() {
```

```
 temp = dht.readTemperature() -5; // Read temperature as Celsius (the default)
 if (isnan(temp)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
 return temp;
}
float umidade() {
 hum = dht.readHumidity() -10; // Reading temperature or humidity takes about 250
milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  if (isnan(hum)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
 return hum;
}

int qualidade() {
  qua  =  analogRead ( 3 ) -200;        // lê o pino 3 da entrada analógica
  return qua;
}

float pontoDoOrvalho() {
  tpo = temperatura() - ((100 -umidade())/5);
  return tpo;
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  char pacote[80];
  char str_temp [7];
```

```
  char str_umi [7];
  char str_tpo [7];
  char str_qa [7];
  char pacoteRecebido [7];
  char barra = "/";
  dtostrf(temperatura(),4,2,str_temp);
  dtostrf(umidade(),4,2,str_umi);
  dtostrf(pontoDoOrvalho(),4,2,str_tpo);
  dtostrf(qualidade(),4,2,str_qa);

sprintf(pacote,"%s%s%s%s%s%s%s%s",str_temp,"/",str_umi,"/",str_tpo,"/",str_qa,"/")
;
  Serial.println(pacote);
}
```

APÊNDICE B – CÓDIGO DO ESP32

```
/**************************************************************************
***
 UNIVERSIDADE DO ESTADO DO AMAZONAS
Projeto de conclusão de curso 2
Autor: Marcos Henrique Correa Lima
Este programa tem a finalidade de receber os dados enviados pelo arduino e
enviá-los para o servidor MySQL
**************************************************************************
*/
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>
#include <WiFi.h>

IPAddress server_addr(185, 42, 117, 115); // O IP DO SERVIDOR DA CLEVER
CLOUD
char user[] = "uc1nnuepqeaqot1y";              // Usuario MySQL
char password[] = "NYkhzXjDMRCQK6x6Dr3a";       //   Senha MySQL

char ssid[] = "LIVE TIM_0780_2G";         //  Nome de rede Wifi
char pass[] = "4mnpgu3jcf";     //           Senha Wi-Fi

char INSERT_DATA[] = "INSERT INTO bmagvfomauffo3h6cpix.Leitura
(Temperatura, Umidade, PontodoOrvalho, Qualidade) VALUES (%s,%s,%s,%s)";

WiFiClient client;
MySQL_Connection conn(&client);
MySQL_Cursor* cursor;

#define RXp2 16
#define TXp2 17
void setup() {
```

```
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, RXp2, TXp2);
  verificaWiFi();
}
void loop() {
  float temp;
  float umi;
  float tpo;
  float qa;
  leitura(&temp,&umi,&tpo,&qa);
  Serial.println("Message Received: ");
  delay(100);
  enviaDados(temp,umi,tpo,qa);
}

void verificaWiFi() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Sem conexão"); //
    WiFi.disconnect();
    delay(1000);
    WiFi.begin(ssid, pass);
    Serial.println();
    Serial.println("Conectando ao WiFi."); //
    while (WiFi.status() != WL_CONNECTED) {
      Serial.println(WiFi.status());
      delay(500);
    }
    Serial.println();Serial.println("Conectado a rede!"); //
  }
}

void enviaDados(float temp, float umi, float tpo, float qa) {
  char query[128];
  char temperatura[10];
```

```
  char umidade[10];
  char pontodoorvalho[10];
  char qualidade[10];
  verificaWiFi();
  if (conn.connect(server_addr, 3306, user, password)) {
    delay(2000);
    MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);
    dtostrf(temp, 1, 1, temperatura);
    dtostrf(umi, 1, 1, umidade);
    dtostrf(tpo, 1, 1, pontodoorvalho);
    dtostrf(qa, 1, 1, qualidade);
    // Save
    sprintf(query, INSERT_DATA, temperatura, umidade, pontodoorvalho,qualidade);
    cur_mem->execute(query);
    delete cur_mem;
    Serial.println(); Serial.println("Data recorded."); //
  }
  else
    Serial.println(); Serial.println("Connection failed."); //
  conn.close();
}

void leitura(float *temp, float *umi,float *tpo, float *qa) {
  char  mensagem[120];
  byte atual, i = 0;
  atual = 255;
  if (Serial2.available() > 0) {
    while (atual != 10) {
      if (Serial2.available() > 0) {
        atual = Serial2.read();
            Serial.print((char)atual);
        mensagem[i] = (char)atual;
        i++;
      }
```

```
    }
    i = 0;
    *temp = atof(strtok(mensagem, "/"));
    *umi = atof(strtok(NULL, "/"));
    *tpo = atof(strtok(NULL, "/"));
    *qa = atof(strtok(NULL, "/"));
  }
}
```

APÊNDICE C – CÓDIGO PARA ADQUIRIR DADOS DO DATABASE

```
var server = "bmagvfomauffo3h6cpix-mysql.services.clever-cloud.com";

var port = 3306;

var dbName = "bmagvfomauffo3h6cpix";

var username = "uc1nnuepqeaqot1y";

var password = "NYkhzXjDMRCQK6x6Dr3a";

var url = "jdbc:mysql://"+server+":"+port+"/"+dbName;

function readData() {

 var conn = Jdbc.getConnection(url, username, password);

 var stmt = conn.createStatement();

 var results = stmt.executeQuery("SELECT * FROM Leitura");

 var metaData=results.getMetaData();
```

```
var numCols = metaData.getColumnCount();

var spreadsheet = SpreadsheetApp.getActive();

var sheet = spreadsheet.getSheetByName("SQL");

sheet.clearContents();

var arr=[];

for (var col = 0; col < numCols; col++) {

  arr.push(metaData.getColumnName(col + 1));

}

sheet.appendRow(arr);

while (results.next()) {

arr=[];
```

```
 for (var col = 0; col < numCols; col++) {

   arr.push(results.getString(col + 1));

 }

 sheet.appendRow(arr);

}

results.close();

stmt.close();

sheet.autoResizeColumns(1, numCols+1);

}

ScriptApp.newTrigger(“readData”)

.timeBased()

.everyMinutes(5)
```

```
.create();
```

APÊNDICE D – CÓDIGO DO APLICATIVO

```
package com.learntodroid.androidlinecharttutorial;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.RadioGroup;
import android.widget.TextView;

import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.components.Description;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.formatter.ValueFormatter;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;
import com.google.android.material.textfield.TextInputLayout;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Date;
import java.util.List;
import java.util.Locale;
```

```
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class MainActivity extends AppCompatActivity {

    private LineChart lineChart;
    private TextView stockTickerTextInputLayout;
    private CheckBox temperaturaCheckBox, umidadeCheckBox,
pontoDoOrvalhoCheckBox, qualidadeCheckBox;

    List temperatura = new ArrayList();
    List umidade = new ArrayList();
    List pontoDoOrvalho = new ArrayList();
    List qualidade = new ArrayList();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        lineChart = findViewById(R.id.activity_main_linechart);

        stockTickerTextInputLayout = findViewById(R.id.activity_main_stockticker);
        temperaturaCheckBox = findViewById(R.id.activity_main_temperatura);
        umidadeCheckBox = findViewById(R.id.activity_main_umidade);
        pontoDoOrvalhoCheckBox =
findViewById(R.id.activity_main_pontodoorvalho);
        qualidadeCheckBox = findViewById(R.id.activity_main_qualidade);

        configureLineChart();
        findViewById(R.id.activity_main_atualizar).setOnClickListener(new
View.OnClickListener() {
            @Override
```

```
        public void onClick(View v) {
            try {
                getStockData();
                setLineChartData(temperatura, umidade, pontoDoOrvalho, qualidade);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    });
}

private void configureLineChart() {
    Description desc = new Description();
    desc.setText("Historico de Parametros do ar");
    desc.setTextSize(28);
    lineChart.setDescription(desc);

    XAxis xAxis = lineChart.getXAxis();
    xAxis.setValueFormatter(new ValueFormatter() {
        private final SimpleDateFormat mFormat = new SimpleDateFormat("dd
MMM", Locale.ENGLISH);

        @Override
        public String getFormattedValue(float value) {
            long millis = (long) value * 1000L;
            return mFormat.format(new Date(millis));
        }
    });
}

private void getStockData() throws IOException {
    temperatura = GetValues.getValues("18JOMXLrm6Xa6uj4FXXizgnb9-
Cpx5tPVmMR86hXRDyM","A2:A200");
    umidade = GetValues.getValues("18JOMXLrm6Xa6uj4FXXizgnb9-
```

```
Cpx5tPVmMR86hXRDyM","B2:B200");
        pontoDoOrvalho = GetValues.getValues("18JOMXLrm6Xa6uj4FXXizgnb9-
Cpx5tPVmMR86hXRDyM","C2:C200");
        qualidade = GetValues.getValues("18JOMXLrm6Xa6uj4FXXizgnb9-
Cpx5tPVmMR86hXRDyM","D2:D200");
    }

    private void setLineChartData(List temp, List umi, List pdo, List qual) {
        ArrayList<ILineDataSet> dataSets = new ArrayList<>();

        if (temperaturaCheckBox.isChecked()) {
            LineDataSet temperaturaDataSet = new LineDataSet(temp, "Temperatura");
            temperaturaDataSet.setDrawCircles(true);
            temperaturaDataSet.setCircleRadius(4);
            temperaturaDataSet.setDrawValues(false);
            temperaturaDataSet.setLineWidth(3);
            temperaturaDataSet.setColor(Color.GREEN);
            temperaturaDataSet.setCircleColor(Color.GREEN);
            dataSets.add(temperaturaDataSet);
        }

        if (umidadeCheckBox.isChecked()) {
            LineDataSet umidadeDataSet = new LineDataSet(umi, "Umidade");
            umidadeDataSet.setDrawCircles(true);
            umidadeDataSet.setCircleRadius(4);
            umidadeDataSet.setDrawValues(false);
            umidadeDataSet.setLineWidth(3);
            umidadeDataSet.setColor(Color.RED);
            umidadeDataSet.setCircleColor(Color.RED);
            dataSets.add(umidadeDataSet);
        }

        if (pontoDoOrvalhoCheckBox.isChecked()) {
            LineDataSet pontoDoOrvalhoDataSet = new LineDataSet(pdo, "Ponto Do
```

```
Orvalho");
            pontoDoOrvalhoDataSet.setDrawCircles(true);
            pontoDoOrvalhoDataSet.setCircleRadius(4);
            pontoDoOrvalhoDataSet.setDrawValues(false);
            pontoDoOrvalhoDataSet.setLineWidth(3);
            pontoDoOrvalhoDataSet.setColor(Color.rgb(255, 165, 0));
            pontoDoOrvalhoDataSet.setCircleColor(Color.rgb(255, 165, 0));
            dataSets.add(pontoDoOrvalhoDataSet);
        }

        if (qualidadeCheckBox.isChecked()) {
            LineDataSet qualidadeDoArDataSet = new LineDataSet(qual, "Qualidade do
Ar");
            qualidadeDoArDataSet.setDrawCircles(true);
            qualidadeDoArDataSet.setCircleRadius(4);
            qualidadeDoArDataSet.setDrawValues(false);
            qualidadeDoArDataSet.setLineWidth(3);
            qualidadeDoArDataSet.setColor(Color.rgb(255, 165, 0));
            qualidadeDoArDataSet.setCircleColor(Color.rgb(255, 165, 0));
            dataSets.add(qualidadeDoArDataSet);
        }

        LineData lineData = new LineData(dataSets);
        lineChart.setData(lineData);
        lineChart.invalidate();
    }
}
```