SISTEMA DE MEDIÇÃO DE TEMPERATURA E UMIDADE DO AR CONTROLADO POR UMA ESP32 E MONITORADO EM TEMPO REAL POR UM APLICATIVO

Bruna Oliveira dos Santos[1]

RESUMO

Este artigo apresenta o desenvolvimento de um sistema de monitoramento de temperatura e umidade do ar utilizando o microcontrolador ESP32, o sensor DHT22 e a plataforma Firebase, integrados a um aplicativo móvel para visualização dos dados em tempo real. O objetivo é propor melhorias que otimizem a precisão dos dados, a usabilidade do aplicativo, a integração dos componentes e a escalabilidade do sistema, com foco em aplicações práticas como o controle ambiental em ambientes refrigerados. A pesquisa busca oferecer uma solução eficiente e inovadora para monitoramento ambiental, comparando seu desempenho com métodos tradicionais.

Palavras-chave: Esp32; DHT22; Temperatura; Umidade; IOT.

ABSTRACT

This article presents the development of an air temperature and humidity monitoring system using the ESP32 microcontroller, the DHT22 sensor and the Firebase platform, integrated into a mobile application to visualize data in real time. The objective is to propose improvements that optimize data accuracy, application usability, component integration and system scalability, with a focus on practical applications such as environmental control in refrigerated environments. The research seeks to offer an efficient and innovative solution for environmental monitoring, comparing its performance with traditional methods.

Keywords: Esp32; DHT22; Temperature; Humidity; IOT.

1 INTRODUÇÃO

A presente pesquisa tem como tema o desenvolvimento e aprimoramento de um sistema de monitoramento de temperatura e umidade do ar, com foco na implementação de melhorias que permitam a visualização dos dados em tempo real através de um aplicativo móvel. Diante da crescente demanda por sistemas de controle ambiental, especialmente em ambientes onde a temperatura e umidade influenciam diretamente no funcionamento do setor, como por exemplo, o refrigeramento de equipamentos de TI, ou mesmo armazenamento de produtos refrigerados, o projeto visa solucionar questões relacionadas à precisão dos dados, usabilidade do aplicativo, integração eficiente entre componentes, alertas em tempo real.

---

[1] Discente do Curso de Engenharia da Computação da Universidade La Salle- Unilasalle, matriculada na disciplina de Trabalho de Conclusão de Curso II. E-mail: bruna.santos0138@unilasalle.edu.br, sob a orientação Prof. Dr. Mozart Lemos de Siqueira. E-mail: mozart.siqueira@unilasalle.edu.br. Data de entrega: 02 de Dezembro de 2024.

O problema central que norteia o estudo é: como podem ser implementadas melhorias em um sistema de monitoramento de temperatura e umidade para otimizar sua precisão, usabilidade, integração e escalabilidade? Para responder a essa questão, busca-se investigar, dentre outros aspectos, a integração do microcontrolador ESP32 com o sensor DHT22 e a plataforma Firebase, a fim de garantir maior confiabilidade e precisão nos dados coletados. Além disso, o desenvolvimento de um aplicativo intuitivo e eficiente para o monitoramento em tempo real será explorado como uma solução essencial para maximizar a utilidade prática do sistema.

O objetivo geral do trabalho é desenvolver um sistema integrado de monitoramento utilizando o ESP32, o sensor DHT22 e a plataforma Firebase, aliado a um aplicativo móvel para visualização em tempo real, aplicável a contextos como o controle ambiental de ambientes refrigerados. A justificativa para o desenvolvimento desse projeto está enraizada no interesse pessoal da pesquisadora pela área de tecnologia, que, desde a infância, despertou sua curiosidade e motivação para buscar soluções inovadoras na engenharia e computação.

2 REFERENCIAL TEÓRICO

2.1 Sensor Temperatura e Umidade DHT11 e DHT22

O DHT11 é um sensor de temperatura e umidade que permite fazer leituras de temperaturas entre 0 a 50 Celsius e umidade entre 20 a 90%, muito usado para projetos com Arduino. [7]

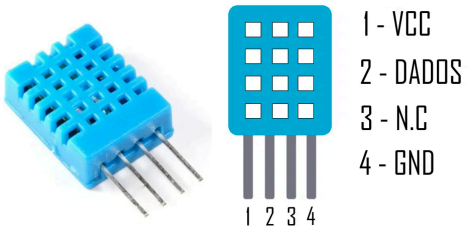

Figura 1- Sensor de Temperatura e Umidade - DHT11. [7]

O sensor DHT22 é uma ferramenta versátil para medir temperatura e umidade, capaz de operar em uma ampla faixa de -40 a +80 ºC para temperatura e de 0 a 100% para umidade relativa. [8]

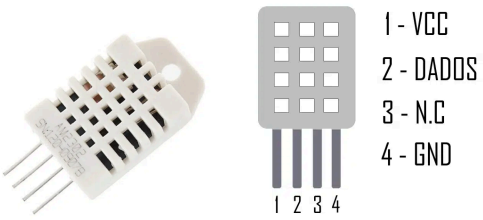

Figura 2- Sensor de Temperatura e Umidade - DHT22.[8]

A principal diferença do Sensor de umidade temperatura DHT11 para o DHT22 é o que o segundo (DHT22) apresenta maior margem de trabalho, alcançando maiores temperaturas e maior nível de umidade, além de possuir maior precisão. [4]

2.2 ESP32

A Esp32 é uma série de microcontroladores de baixo consumo de energia e custo, ela apresenta o clássico módulo de comunicação Wi-Fi e um sistema com processador Dual Core, Bluetooth híbrido e múltiplos sensores embutidos, tornando assim, qualquer construção de sistema envolvendo IOT mais simples e sugestivo[11]. Neste projeto utilizei a Doit Esp32 Devkit V1.

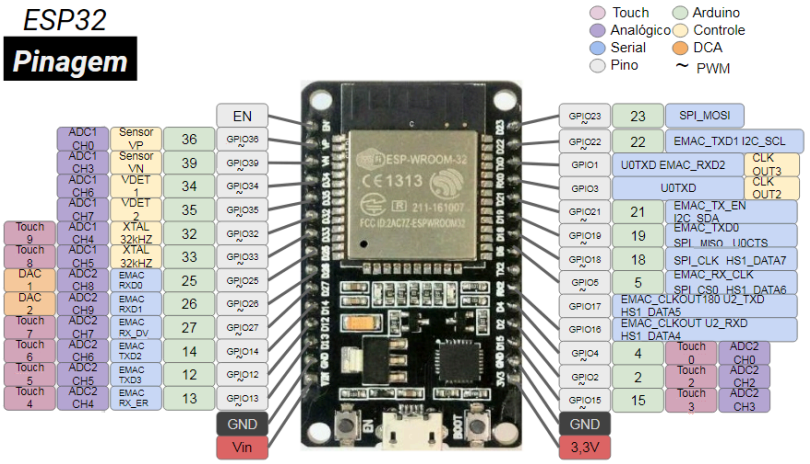

Figura 3 – ESP32 DEV KIT V1. [3]

2.3 Display LCD 16x2 com I2C

O Display Lcd 16×2 é uma pequena tela com fundo azul empregada basicamente para visualização de dados e/ou exibição de mensagens locais em desenvolvimento de projetos robóticos e de automação residencial, é compatível com um grande número de sistemas microcontroladores, entre eles, Arduino, ESP32, PIC, Atmel, etc..

Já o módulo I2C tem como principal função simplificar as conexões e facilitar a utilização de displays LCD ao microcontrolador. Ela simplifica as conexões em apenas 4 com a placa utilizada, das quais dois pinos são para alimentação (VCC e GND), e os outros dois pinos são da interface I2C (SDA e SCL). Outras características são um potenciômetro que fica na placa e serve para ajuste do contraste do display, e um jumper na lateral oposta permite que a luz de fundo (backlight) seja controlada pelo programa ou permaneça apagada.

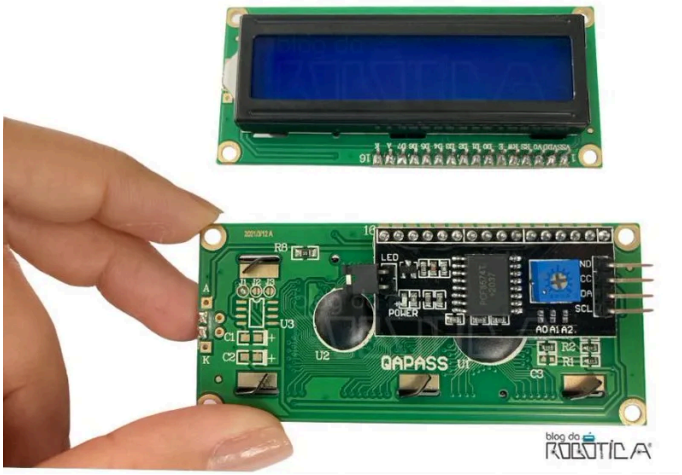

Figura 4: Display 16×2 com módulo I2C. [1]

2.4 Internet das Coisas

A internet das coisas, ou Internet of Things (IoT), é a conexão entre objetos e equipamentos, capaz de armazenar e transmitir dados. Atualmente essa conexão vem trazendo comodidade e facilidade para o cotidiano das pessoas. Uma vez que a encontramos facilmente no nosso dia a dia, seja em um sistema residencial utilizando uma Alexa ou até mesmo em grandes empresas.

3 DESENVOLVIMENTO DO PROJETO

A seguir serão apresentados os processos de montagem do circuito, programação da ESP32, configuração do firebase e programação do aplicativo.

3.1 Circuito eletrônico utilizando ESP32, DHT22 e Display LCD I2C

Os componentes utilizados na montagem do circuito foram: uma ESP32, duas protoboards de 400 pontos, um lcd 16x2 com I2C, o sensor de temperatura e umidade DHT22, um resistor de 10k Ohms e jumpers para conexão.

A função da I2C é reduzir as conexões do LCD com o restante do circuito em apenas 4, enquanto o resistor de 10KΩ entre o VCC e a linha de dados serve pull-up para mantê-lo ALTO para a comunicação adequada entre o sensor e o MCU

A conexão foi realizada conforme imagem abaixo:

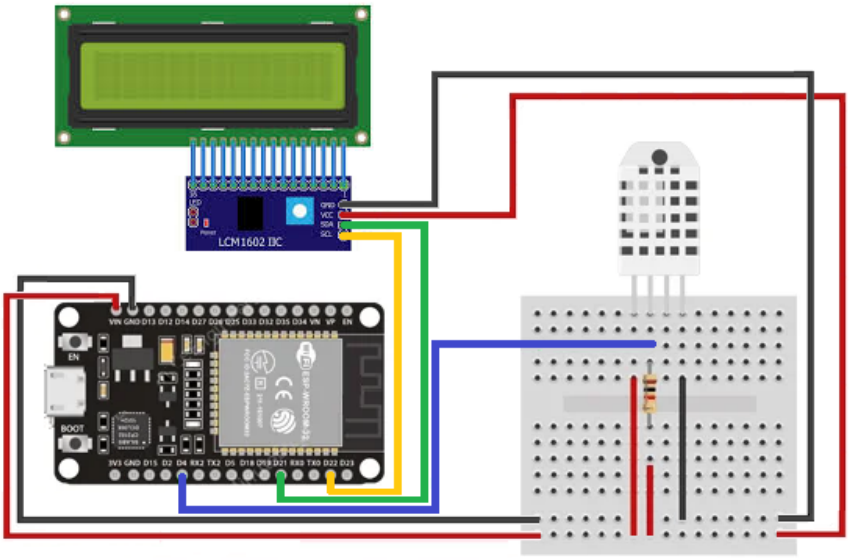

Figura 5: Edição do autor(a)

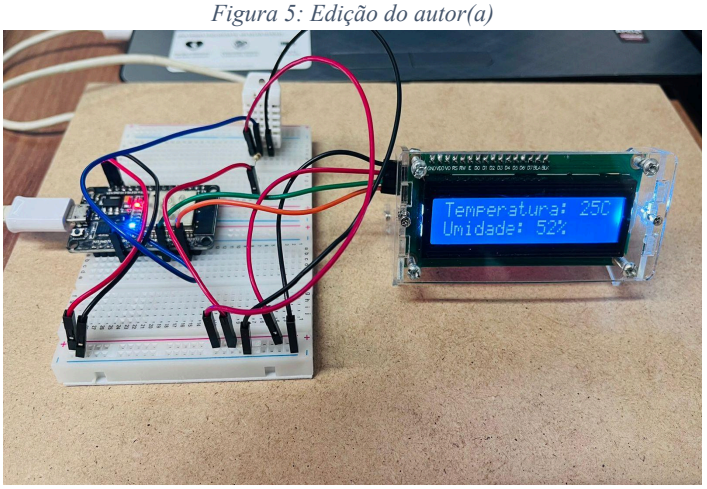

Figura 6: Autor(a)

3.2 Programação ESP32

A programação da ESP32 encontrasse no apêndice A, e através dela configuramos a integração com o firebase, enviando os caminhos em que os dados serão salvos

3.3 Configuração do Firebase

Nesta etapa do projeto, o tutorial do site Eletrofun[5] foi essencial para o avanço nas configurações do firebase. Nele é explicado desde como criar o projeto no site, configurar e mostra até um exemplo de aplicação. Tornando mais fácil a integração do Firebase com a ESP32 e o aplicativo. A imagem abaixo, mostra como os dados ficam no Realtime database do Firebase. Nela é possível ver onde o alerta e sua condição são salvos, e onde também o histórico e os valores em tempo real ficam armazenados.

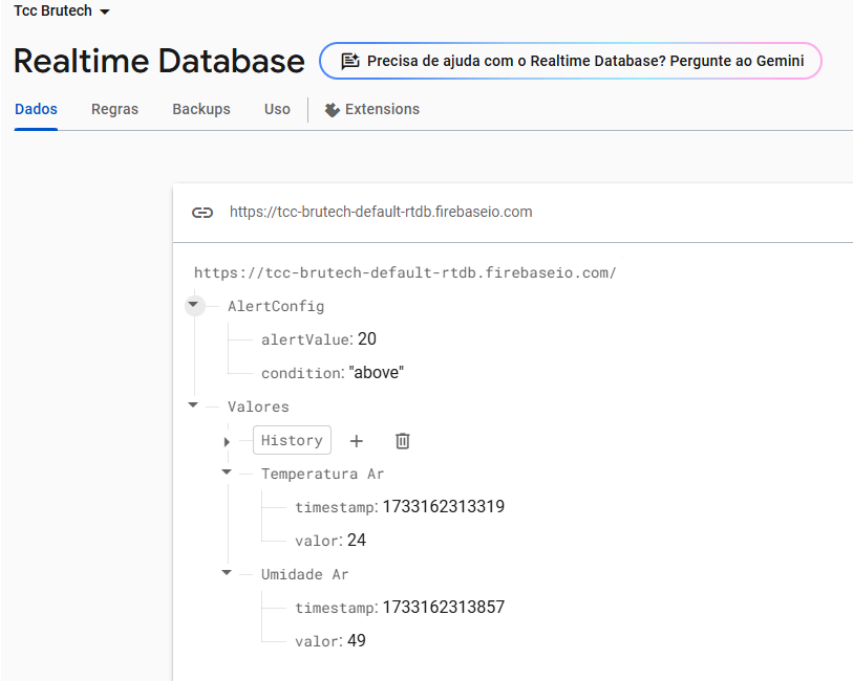

Figura 7: Autor(a)

3.4 Programação Aplicativo

A programação do aplicativo foi realizada utilizando o VS Code e o Expo, ferramentas que facilitam significativamente os testes da aplicação. Basta atualizar o código, salvá-lo, e o Expo permite testá-lo facilmente em um emulador próprio.

Foram criadas duas telas: HomeScreen e HistoryScreen, além de um arquivo chamado App.js, que realiza a navegação e a integração entre as telas. A tela HomeScreen apresenta os dados de temperatura e umidade em tempo real e permite configurar o valor do alerta. Nela, há um botão chamado "Exibir Histórico", que leva à próxima tela, onde está disponível o histórico de todas as leituras do sensor em intervalos de 10 minutos entre cada leitura.

Figura 8: Tela HomeScreen

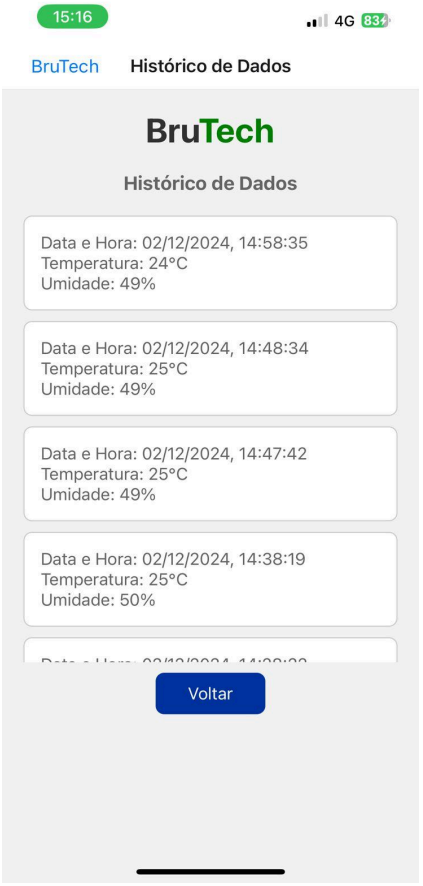

Figura 9: Tela HistoryScreen

3.5 Testes

Os testes foram realizados em diferentes ambientes, como na própria sala do servidor da empresa em que trabalho, na minha casa e no escritório da empresa. Onde, ao meu ver, o melhor desempenho foi na sala do escritório. Creio eu que pelo sinal da rede wifi ser melhor, uma vez que a sala do servidor era um cofre antigamente e o sinal das operadoras são péssimos.

Figura 10: Sala dos Servidores da empresa

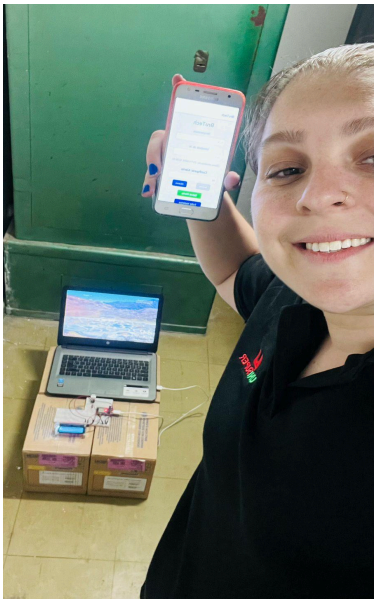

Figura 11: Sala dos Servidores da empresa

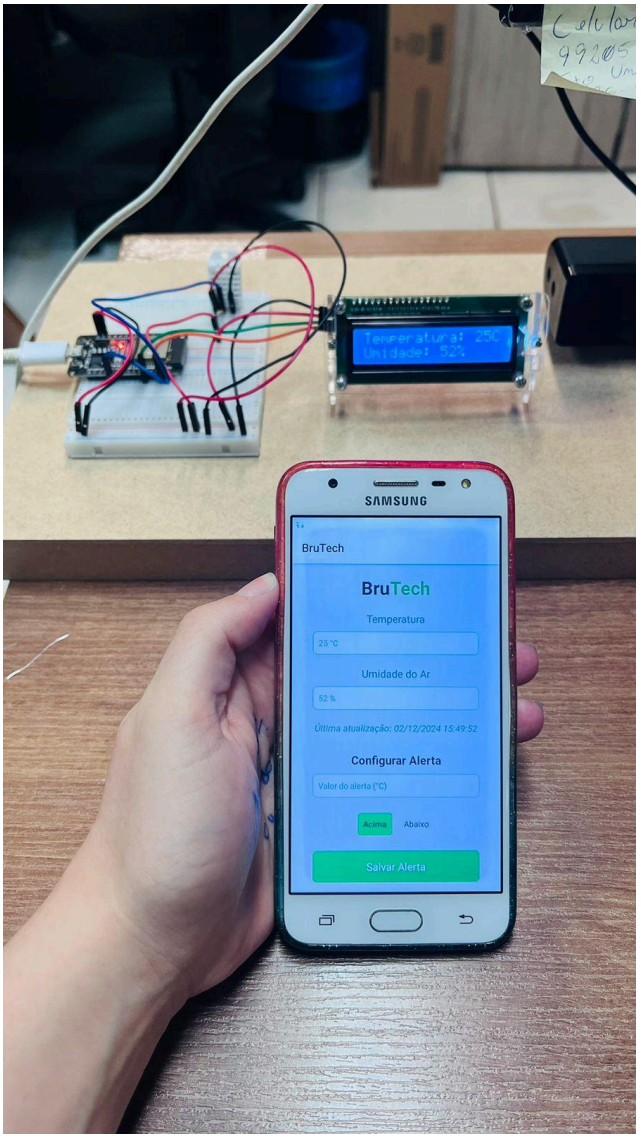

Figura 12: Sala do escritório da empresa

4 CONSIDERAÇÕES FINAIS

Analisando o trabalho até aqui apresentado, o maior desafio foi realmente o desenvolvimento do aplicativo, por ter pouca prática nesta área. Os objetivos foram atingidos, desenvolvendo um sistema de monitoramento integrado ao firebase e exibindo os dados em tempo real e histórico em um aplicativo.

Vale ressaltar que ao fim do desenvolvimento, a função de notificação dos alertas foi inserida e até o momento da apresentação estava demonstrando erro, não atualizando os dados em tempo real, neste momento verifiquei que o causador do erro foi um resistor de 1k Ohm colocado no lugar do resistor de 10k Ohms por engano, após a troca aplicativo voltou a funcionar corretamente.

Melhorias para o futuro são uma configuração melhor das notificações e um filtro para busca no histórico. Através deste projeto, foi possível reunir o hardware e o software fazendo jus de que a engenharia da computação é mesmo a ciência que projeta, desenvolve e gerencia sistemas computacionais.

5 APÊNDICE A

```
#include <Adafruit_I2CDevice.h>
```

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <Arduino.h>
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"

// Definição de variáveis
#define WIFI_SSID "iPhone de Bruna"
#define WIFI_PASSWORD "brutech123"
#define API_KEY "AIzaSyCMGXYGpik9fbcgZT8vyIiB1AD--i4AAuc"
#define DATABASE_URL "https://tcc-brutech-default-rtdb.firebaseio.com"

// Definição de pinos
#define pinoSensorTemp 4  // Conectar o sensor de temperatura DHT22 ao pino 4

LiquidCrystal_I2C lcd(0x27, 16, 2);
DHT dht(pinoSensorTemp, DHT22);
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

// Variáveis necessárias
int temperaturaAr, umidadeAr;
char simbolograu = (char)223;
unsigned long sendDataPrevMillis = 0;
bool signupOK = false;

void setup() {
  Serial.begin(115200);

  // Ligação à Rede Wi-Fi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("A ligar à rede Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
```

```
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Conexão bem sucedida. IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  // Configuração do Firebase
  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;
  if (Firebase.signUp(&config, &auth, "", "")) {
    Serial.println("Acesso à base de dados concedido com sucesso!");
    signupOK = true;
  } else {
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
  }
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  // Inicialização do sensor e LCD
  dht.begin();
  lcd.init();
  lcd.backlight();
}

void loop() {
  if (millis() - sendDataPrevMillis > 600000 || sendDataPrevMillis == 0) { // 10 minutos
    sendDataPrevMillis = millis();

    // Ler os valores do sensor
    temperaturaAr = dht.readTemperature();
    umidadeAr = dht.readHumidity();

    // Atualizar o LCD
    atualizarInformacoes();
```

```
    // Enviar os dados para o Firebase
    atualizarDatabase();
  }
}

void atualizarInformacoes() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temperatura: ");
  lcd.print(temperaturaAr);
  lcd.print("C");

  lcd.setCursor(0, 1);
  lcd.print("Umidade: ");
  lcd.print(umidadeAr);
  lcd.print("%");
}

void atualizarDatabase() {
  if (Firebase.ready() && signupOK) {
    // Enviar dados de temperatura e umidade
    if (!Firebase.RTDB.setInt(&fbdo, "Valores/Temperatura Ar/valor", temperaturaAr)) {
      Serial.print("Erro ao enviar temperatura: ");
      Serial.println(fbdo.errorReason());
    }
    if (!Firebase.RTDB.setInt(&fbdo, "Valores/Umidade Ar/valor", umidadeAr)) {
      Serial.print("Erro ao enviar umidade: ");
      Serial.println(fbdo.errorReason());
    }

    // Adicionar timestamps às leituras mais recentes
    if (!Firebase.RTDB.setTimestamp(&fbdo, "Valores/Temperatura Ar/timestamp")) {
      Serial.print("Erro ao enviar timestamp de temperatura: ");
      Serial.println(fbdo.errorReason());
    }
    if (!Firebase.RTDB.setTimestamp(&fbdo, "Valores/Umidade Ar/timestamp")) {
      Serial.print("Erro ao enviar timestamp de umidade: ");
```

```
    Serial.println(fbdo.errorReason());
  }

  // Adicionar ao histórico com o timestamp do servidor
  String timestamp = String(millis());
  String historyPath = "Valores/History/" + timestamp;

  if (!Firebase.RTDB.setInt(&fbdo, historyPath + "/temperature", temperaturaAr)) {
    Serial.print("Erro ao enviar histórico de temperatura: ");
    Serial.println(fbdo.errorReason());
  }
  if (!Firebase.RTDB.setInt(&fbdo, historyPath + "/humidity", umidadeAr)) {
    Serial.print("Erro ao enviar histórico de umidade: ");
    Serial.println(fbdo.errorReason());
  }
  if (!Firebase.RTDB.setTimestamp(&fbdo, historyPath + "/timestamp")) {
    Serial.print("Erro ao enviar timestamp do histórico: ");
    Serial.println(fbdo.errorReason());
  }

  Serial.println("Dados e timestamps enviados para o Firebase com sucesso!");
 }
}
```

6 APÊNDICE B

6.1 App.js

```
import React, { useEffect, useState } from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import { Platform, Alert } from 'react-native';
import * as Notifications from 'expo-notifications';
import HomeScreen from './HomeScreen'; // Tela principal
import HistoryScreen from './HistoryScreen'; // Tela de histórico

const Stack = createStackNavigator();

// Configuração global do NotificationHandler para ambas as plataformas
Notifications.setNotificationHandler({
  handleNotification: async () => ({
    shouldShowAlert: true, // Exibe notificações no formato de alerta
    shouldPlaySound: true, // Reproduz som de notificação
    shouldSetBadge: false, // Não altera o badge do ícone do app
```

```
  }),
});

export default function App() {
  const [pushToken, setPushToken] = useState(null);

  useEffect(() => {
    async function requestPermissions() {
      const { status } = await Notifications.requestPermissionsAsync();
      if (status !== 'granted') {
        alert('Você precisa dar permissão para receber notificações!');
      } else {
        const token = await Notifications.getExpoPushTokenAsync();
        setPushToken(token.data);
        console.log('Token de notificação:', token.data);
      }
    }

    requestPermissions();

    const backgroundSubscription =
Notifications.addNotificationReceivedListener((notification) => {
      console.log('Notificação recebida no segundo plano:', notification);
      Alert.alert('Notificação recebida', notification.request.content.body);
    });

    const responseSubscription =
Notifications.addNotificationResponseReceivedListener((response) => {
      console.log('Resposta à notificação:', response);
      const screen = response.notification.request.content.data.screen;
      if (screen) {
        navigationRef.current?.navigate(screen);
      }
    });

    return () => {
      backgroundSubscription.remove();
      responseSubscription.remove();
    };
  }, []);

  const navigationRef = React.createRef();

  return (
    <NavigationContainer ref={navigationRef}>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen} options={{ title:
'BruTech' }} />
        <Stack.Screen name="History" component={HistoryScreen} options={{ title:
'Histórico de Dados' }} />
```

```
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

6.2 HomeScreen.js

```
import React, { useEffect, useState } from 'react';
import { StyleSheet, Text, View, TextInput, TouchableOpacity, Image, Alert,
KeyboardAvoidingView, Platform, ScrollView } from 'react-native';
import { initializeApp } from 'firebase/app';
import { getDatabase, ref, onValue, set, get } from 'firebase/database';
import * as Notifications from 'expo-notifications';

// Configuração do Firebase
const firebaseConfig = {
  apiKey: "AIzaSyCMGXYGpik9fbcgZT8vyIiB1AD--i4AAuc",
  authDomain: "tcc-brutech.firebaseapp.com",
  databaseURL: "https://tcc-brutech-default-rtdb.firebaseio.com",
  projectId: "tcc-brutech",
  storageBucket: "tcc-brutech.firebasestorage.app",
  messagingSenderId: "38224103025",
  appId: "1:38224103025:web:412ce9ad1e957066f83dd0",
};

// Inicialização do Firebase
const app = initializeApp(firebaseConfig);
const db = getDatabase(app);

export default function HomeScreen({ navigation }) {
  const [temperature, setTemperature] = useState('');
  const [humidity, setHumidity] = useState('');
  const [lastUpdate, setLastUpdate] = useState('');
  const [alertValue, setAlertValue] = useState('');
  const [alertCondition, setAlertCondition] = useState('above');
  const [alertConfig, setAlertConfig] = useState(null);
  const [lastNotificationTime, setLastNotificationTime] = useState(null);

  const formatTimestamp = (timestamp) => {
    const date = new Date(timestamp);
    return `${date.toLocaleDateString()} ${date.toLocaleTimeString()}`;
  };

  const fetchInitialData = async () => {
    try {
      const tempSnapshot = await get(ref(db, 'Valores/Temperatura Ar'));
      const humSnapshot = await get(ref(db, 'Valores/Umidade Ar'));
      const alertSnapshot = await get(ref(db, 'AlertConfig'));

      if (tempSnapshot.exists()) {
        const data = tempSnapshot.val();
```

```
      setTemperature(data.valor);
      if (data.timestamp) setLastUpdate(formatTimestamp(data.timestamp));
    }

    if (humSnapshot.exists()) {
      const data = humSnapshot.val();
      setHumidity(data.valor);
    }

    if (alertSnapshot.exists()) {
      setAlertConfig(alertSnapshot.val());
    }
  } catch (error) {
    console.error('Erro ao buscar dados iniciais:', error);
  }
};

useEffect(() => {
  fetchInitialData();

  const tempRef = ref(db, 'Valores/Temperatura Ar');
  const humRef = ref(db, 'Valores/Umidade Ar');
  const alertRef = ref(db, 'AlertConfig');

  const tempListener = onValue(tempRef, (snapshot) => {
    const data = snapshot.val();
    if (data) {
      setTemperature(data.valor);
      if (data.timestamp) setLastUpdate(formatTimestamp(data.timestamp));
    }
  });

  const humListener = onValue(humRef, (snapshot) => {
    const data = snapshot.val();
    if (data) {
      setHumidity(data.valor);
    }
  });

  const alertListener = onValue(alertRef, (snapshot) => {
    if (snapshot.exists()) {
      setAlertConfig(snapshot.val());
    }
  });

  return () => {
    tempListener();
    humListener();
    alertListener();
  };
```

```
}, []);

const sendNotification = async (message) => {
  await Notifications.scheduleNotificationAsync({
    content: {
      title: 'Alerta de Temperatura',
      body: message,
    },
    trigger: null, // Notificação imediata
  });
};

useEffect(() => {
  const checkAndNotify = async () => {
    if (temperature && alertConfig) {
      const tempValue = parseFloat(temperature);
      const { alertValue, condition } = alertConfig;

      let shouldSendNotification = false;
      let message = '';

      if (condition === 'above' && tempValue > alertValue) {
        message = `A temperatura ultrapassou o limite de ${alertValue}°C!`;
        shouldSendNotification = true;
      } else if (condition === 'below' && tempValue < alertValue) {
        message = `A temperatura caiu abaixo do limite de ${alertValue}°C!`;
        shouldSendNotification = true;
      }

      if (shouldSendNotification) {
        const currentTime = new Date().getTime();

        if (!lastNotificationTime || currentTime - lastNotificationTime > 10 * 60 * 1000) {
          await Notifications.cancelAllScheduledNotificationsAsync();
          await sendNotification(message);

          await Notifications.scheduleNotificationAsync({
            content: { title: 'Alerta de Temperatura', body: message },
            trigger: { seconds: 600, repeats: true }, // 10 minutos
          });

          setLastNotificationTime(currentTime);
        }
      } else {
        await Notifications.cancelAllScheduledNotificationsAsync();
      }
    }
  };

  checkAndNotify();
```

```
  }, [temperature, alertConfig, lastNotificationTime]);

  useEffect(() => {
    const requestPermissions = async () => {
      const { status } = await Notifications.requestPermissionsAsync();
      if (status !== 'granted') {
        alert('Permissão para notificações não concedida.');
      }
    };

    requestPermissions();
  }, []);

  const saveAlertConfig = () => {
    if (!alertValue || isNaN(alertValue)) {
      alert('Por favor, insira um valor válido para o alerta.');
      return;
    }

    const config = {
      alertValue: parseFloat(alertValue),
      condition: alertCondition,
    };

    set(ref(db, 'AlertConfig'), config)
      .then(() => {
        setAlertConfig(config);
        alert(`Configuração salva: Alerta para ${alertCondition === 'above' ? 'acima' :
'abaixo'} de ${alertValue}°C`);
      })
      .catch((error) => {
        alert('Erro ao salvar configuração: ' + error.message);
      });
  };

  return (
    <KeyboardAvoidingView style={styles.container} behavior={Platform.OS === 'ios' ?
'padding' : undefined}>
      <ScrollView contentContainerStyle={styles.scrollContainer}
keyboardShouldPersistTaps="handled">
        <Text style={styles.title}>Bru<Text style={styles.tech}>Tech</Text></Text>
        <Text style={styles.label}>Temperatura</Text>
        <TextInput
          style={styles.input}
          value={temperature !== null ? `${temperature} °C` : 'Carregando...'}
          editable={false}
        />
        <Text style={styles.label}>Umidade do Ar</Text>
        <TextInput
          style={styles.input}
```

```
          value={humidity !== null ? `${humidity} %` : 'Carregando...'}
          editable={false}
        />
        <Text style={styles.lastUpdate}>Última atualização: {lastUpdate ||
'Carregando...'}</Text>

        <Text style={styles.sectionTitle}>Configurar Alerta</Text>
        <TextInput
          style={styles.input}
          keyboardType="numeric"
          placeholder="Valor do alerta (°C)"
          value={alertValue}
          onChangeText={setAlertValue}
        />
        <View style={styles.buttonGroup}>
          <TouchableOpacity
            style={[styles.conditionButton, alertCondition === 'above' &&
styles.conditionButtonActive]}
            onPress={() => setAlertCondition('above')}
          >
            <Text style={styles.conditionButtonText}>Acima</Text>
          </TouchableOpacity>
          <TouchableOpacity
            style={[styles.conditionButton, alertCondition === 'below' &&
styles.conditionButtonActive]}
            onPress={() => setAlertCondition('below')}
          >
            <Text style={styles.conditionButtonText}>Abaixo</Text>
          </TouchableOpacity>
        </View>
        <TouchableOpacity style={styles.saveButton} onPress={saveAlertConfig}>
          <Text style={styles.buttonText}>Salvar Alerta</Text>
        </TouchableOpacity>

        <TouchableOpacity style={styles.historyButton} onPress={() =>
navigation.navigate('History')}>
          <Text style={styles.historyButtonText}>Exibir histórico</Text>
        </TouchableOpacity>

        <Image source={require('./assets/universidade-logo.png')}
style={styles.footerImage} resizeMode="contain" />
      </ScrollView>
    </KeyboardAvoidingView>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#f2f2f2',
```

```
},
scrollContainer: {
  paddingVertical: 20,
  alignItems: 'center',
},
title: {
  fontSize: 32,
  fontWeight: 'bold',
  color: '#333',
},
tech: {
  color: 'green',
},
label: {
  marginTop: 20,
  fontSize: 18,
  color: '#666',
},
input: {
  height: 40,
  width: '80%',
  borderColor: '#ccc',
  borderWidth: 1,
  borderRadius: 8,
  paddingHorizontal: 10,
  marginTop: 10,
  backgroundColor: '#fff',
},
lastUpdate: {
  marginTop: 20,
  fontSize: 16,
  color: '#888',
  fontStyle: 'italic',
},
sectionTitle: {
  marginTop: 30,
  fontSize: 20,
  fontWeight: 'bold',
  color: '#333',
},
buttonGroup: {
  flexDirection: 'row',
  marginTop: 20,
},
conditionButton: {
  padding: 10,
  margin: 5,
  backgroundColor: '#f0f0f0',
  borderRadius: 5,
},
```

```
  conditionButtonActive: {
    backgroundColor: '#4CAF50',
  },
  conditionButtonText: {
    color: '#333',
  },
  saveButton: {
    backgroundColor: '#4CAF50',
    padding: 15,
    borderRadius: 8,
    marginTop: 20,
    width: '80%',
  },
  buttonText: {
    color: '#fff',
    textAlign: 'center',
    fontSize: 18,
  },
  footerImage: {
    width: 150,
    height: 150,
    marginTop: 30,
  },
  historyButton: {
    backgroundColor: '#007bff',
    padding: 15,
    borderRadius: 8,
    marginTop: 20,
    width: '80%',
  },
  historyButtonText: {
    color: '#fff',
    textAlign: 'center',
    fontSize: 18,
  },
});
```

6.3 HistoryScreen.js

```
import React, { useEffect, useState } from 'react';
import { StyleSheet, Text, View, TouchableOpacity, ScrollView, Image } from
'react-native';
import { getDatabase, ref, onValue } from 'firebase/database';

const db = getDatabase();

export default function HistoryScreen({ navigation }) {
  const [history, setHistory] = useState([]);

  useEffect(() => {
    const historyRef = ref(db, 'Valores/History'); // Ajuste o caminho para o histórico
```

```
    onValue(historyRef, (snapshot) => {
      const data = snapshot.val();
      if (data) {
        const processedData = Object.entries(data)
          .map(([key, values]) => ({
            timestamp: values.timestamp, // Obtém o timestamp diretamente do banco de
dados
            temperature: values.temperature,
            humidity: values.humidity,
          }))
          .sort((a, b) => b.timestamp - a.timestamp); // Ordena em ordem decrescente
de timestamp

        setHistory(processedData);
      }
    });
  }, []);

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Bru<Text style={styles.tech}>Tech</Text></Text>
      <Text style={styles.label}><Text style={{ fontWeight: 'bold' }}>Histórico de
Dados</Text></Text>

      <ScrollView style={styles.historyContainer}>
        {history.map((item, index) => (
          <View key={index} style={styles.historyItem}>
            <Text style={styles.historyText}>Data e Hora: {new
Date(item.timestamp).toLocaleString()}</Text>
            <Text style={styles.historyText}>Temperatura: {item.temperature}°C</Text>
            <Text style={styles.historyText}>Umidade: {item.humidity}%</Text>
          </View>
        ))}
      </ScrollView>

      {/* Botão para voltar */}
      <TouchableOpacity style={styles.button} onPress={() => navigation.goBack()}>
        <Text style={styles.buttonText}>Voltar</Text>
      </TouchableOpacity>

      {/* Substituindo o texto pela imagem */}
      <Image source={require('./assets/universidade-logo.png')}
style={styles.footerImage} resizeMode="contain" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
```

```
  flex: 1,
  backgroundColor: '#f2f2f2',
  alignItems: 'center',
  padding: 20,
},
title: {
  fontSize: 32,
  fontWeight: 'bold',
  color: '#333',
},
tech: {
  color: 'green',
},
label: {
  marginTop: 20,
  fontSize: 18,
  color: '#666',
},
historyContainer: {
  width: '100%',
  marginTop: 20,
},
historyItem: {
  backgroundColor: '#fff',
  padding: 15,
  borderRadius: 8,
  borderColor: '#ccc',
  borderWidth: 1,
  marginBottom: 10,
},
historyText: {
  fontSize: 16,
  color: '#666',
},
button: {
  backgroundColor: '#0033a0',
  paddingVertical: 10,
  paddingHorizontal: 30,
  borderRadius: 8,
  marginTop: 10, // Ajuste para que os botões tenham um pequeno espaço entre si
},
buttonText: {
  color: '#fff',
  fontSize: 16,
},
footerImage: {
  marginTop: 40,
  width: 100, // Ajuste a largura da imagem
  height: 100, // Ajuste a altura da imagem
},
```

});

REFERÊNCIAS

[1] BLOG DA ROBOTICA. Como utilizar o display LCD 16×02 com módulo I2C no Arduino. Disponível em: <https://www.blogdarobotica.com/2022/05/02/como-utilizar-o-display-lcd-16x02-com-modulo-i2c-no-arduino/>
Acesso em: 20 ago. 2024.

[2] CAPSISTEMA. Interface DHT11 DHT22 com ESP32 e valores de exibição usando servidor Web. Disponível em: <https://capsistema.com.br/index.php/2020/12/03/interface-dht11-dht22-com-esp32-e-valores-de-exibicao-usando-servidor-web/>
Acesso em: 27 set. 2024.

[3] CURTO CIRCUITO. Conhecendo o Esp32. Disponível em:<https://www.curtocircuito.com.br/blog/Categoria%20IoT/conhec endo-esp32> .
Acesso em: 10 ago. 2024.

[4] CURTO CIRCUITO. Diferenças entre os Sensores: AHT10, DHT11 e DHT22. Disponível em: <https://curtocircuito.com.br/blog/Categoria%20Arduino/diferencas-entre-os-sensores-aht10-dht11-e-dht22?srsltid=AfmBOopFKhAUYkHPDDuMug31xA4Xua-vNZRlJoGI-KmEdNS4Wu6pVwUY .
Acesso em: 21 out. 2024.

[5] ELECTRO FUN. Como Utilizar a Firebase para Visualizar Dados de um Arduino (ESP32).Disponível em:<https://www.electrofun.pt/blog/como-utilizar-a-firebase-para-visualizar-dados-de-um-arduino-esp32/>. Acesso em: 28 set. 2024.

[6] MAKER HERO. Mostrando a temperatura no LCD 16×2 com o sensor DHT11. Disponível em: <https://www.makerhero.com/blog/mostrando-informacoes-de-temperatura-no-lcd-16x2-com-o-sensor-dht11/>. Acesso em: 24 out. 2024.

[7] MAKER HERO. Sensor de Umidade e Temperatura DHT11. Disponível em:<https://www.makerhero.com/produto/sensor-de-umidade-e-temperatura-dht11/>. Acesso em: 21 out. 2024.

[8] MAKER HERO. Sensor de Umidade e Temperatura DHT22. Disponível em:<https://www.makerhero.com/produto/sensor-de-umidade-e-temperatura-am2302-dht22/>. Acesso em: 21 out. 2024.

[9] MEIO E MENSAGEM. Internet das Coisas: o que é, como funciona e como é utilizada. Disponível em:<https://www.meioemensagem.com.br/proxxima/internet-das-coisas?gad_source=1&gclid=CjwKCAiA3ZC6BhBaEiwAeqfvyioCfjvl2Zp4NbAeXcj27blU3gm6DtwrolsHws8xGOCR8OTqgRKApRoC10kQAvD_BwE>. Acesso em: 20 nov. 2024.

[10] WIKIPEDIA. Arduino IDE. Disponível em: <https://pt.wikipedia.org/wiki/Arduino_IDE/>. Acesso em: 26 ago. 2024.

[11] WIKIPEDIA. Esp32. Disponível em: <https://pt.wikipedia.org/wiki/ESP32> Acesso em: 26 ago. 2024.