# TÍTULO DO ARTIGO: APLICATIVO DE CÁLCULO DIMENSIONAL COM EFEITO DO COEFICIENTE DE DILATAÇÃO TÉRMICA

# Michel Modesto Lúcio1

# Rodrigo Bernardi²

# Nome do coorientador3

Data da versão final: 26 de novembro de 2024.

# RESUMO

A ISO 1 é uma norma que estabelece a temperatura padrão para medições geométricas e dimensionais de produtos seja 20°C com tolerância de +/-1°C, pois a temperatura é um fator que influencia na comparação de medidas com instrumentos de medição, para que os fabricantes de peças e produtos atendam as especificações de engenharia, é fundamental que a variável de temperatura seja controlada ou considerada nos resultados obtidos durante o processo de fabricação. Neste sentido, o aplicativo desenvolvido teve como objetivo auxiliar no controle de dimensional em processos de fabricação em que a temperatura e umidade serão coletadas por sensor e um microcontrolador fornece os valores para o cálculo da dimensão a ser controla considerando o fator de coeficiente de dilatação térmica.

**Palavras-chave:** Flutter; ESP32; DHT22, NBR ISO 1: 1997.

# ABSTRACT

**Keywords:** Flutter; ESP32; DHT22.

# INTRODUÇÃO

Na indústria, em linhas de produção é comum encontrar células de carga instaladas em máquinas que fazem montagens de conjuntos, como exemplo uma prensa de inserção de rolamentos em alojamentos de carcaças ou pontas de eixos, onde a célula de carga é programada para monitorar o esforço da carga em relação ao deslocamento, de maneira indireta é possível perceber variações dimensionais dos conjuntos, quanto maior carga menor é a folga dimensional do conjunto e se apresentar carga baixa, maior é a folga entre o conjunto. Os conjuntos são projetados com folgas de acordo com a aplicação, que são classificadas com grau de tolerâncias, para a fabricação das peças é necessário o controle dimensional afim de cumprir com a exatidão definida pela engenharia de projetos, estes controles devem respeitar a norma NBR ISO1:1997 que define a temperatura 20°C como padrão de referência para medições industriais de comprimento, para isto foi desenvolvido um aplicativo que possa auxiliar na linha de produção a dimensão a ser controlada da peça, em ambiente com a diferença de temperatura, conforme a recomendação normativa de 20°C.

# REVISÃO DE LITERATURA

**2.1. Dilatação Térmica de Materiais**

A dilatação térmica é uma propriedade dos materiais que influencia diretamente a precisão no controle de características dimensionais em processos industriais. Segundo Callister (2019), "a expansão térmica ocorre devido ao aumento na energia vibracional das partículas de um material com a elevação da temperatura, resultando no aumento de suas dimensões".

A fórmula clássica da dilatação linear, Δ𝐿=𝐿0⋅𝛼⋅Δ𝑇, onde:

ΔL é a variação de comprimento,

L0​ é o comprimento inicial,

α é o coeficiente de dilatação térmica,

ΔT é a variação de temperatura.

Peças de materiais metálicos possuem coeficientes de dilatação que variam conforme sua composição. O controle rigoroso da temperatura durante medições e montagens é essencial para evitar erros dimensionais, especialmente em condições de medição fora da temperatura padrão.

**2.2. Norma NBR ISO 1:1997**

A NBR ISO 1:1997 define que a temperatura de referência para medições industriais deve ser 20°C, pois "essa temperatura representa um ponto médio adequado para a maioria dos materiais utilizados na engenharia" (ABNT, 1997). A norma ainda destaca a necessidade de correções em medições feitas em condições ambientais diferentes, um aspecto frequentemente negligenciado em fábricas, mas que pode comprometer a precisão das tolerâncias dimensionais especificadas.

Na prática industrial, o monitoramento da temperatura em tempo real é imprescindível para alinhar os processos às exigências normativas.

**2.3. Fabricação de Peças e Usinagem de Precisão**

A fabricação de peças por usinagem requer conformidade com especificações projetadas. Segundo Kalpakjian e Schmid (2020), "a precisão dimensional em processos de usinagem depende de diversos fatores, incluindo o controle de temperatura, qualidade das ferramentas e estabilidade da máquina".

Além disso, variações térmicas podem gerar desvios dimensionais que impactam diretamente na montagem e funcionalidades de conjuntos mecânicos. Um estudo de Smith et al. (2018) demonstrou que "erros acumulativos em medições realizadas fora das condições normativas aumentam proporcionalmente com a elevação da temperatura ambiente".

**2.4. Tolerâncias de Ajuste**

Os graus de tolerâncias definidos pelas normas, como por exemplo NBR ISO 6158/95 auxiliam na tolerâncias de folgas para fabricação de peças, para que na montagem e a funcionalidade dos conjuntos atendam as expectativas da engenharia de projetos. Como afirmam Neves e Lima (2017), "os ajustes de folga e interferência são planejados considerando variações inevitáveis na produção, mas a aplicação correta dos limites é vital para evitar falhas funcionais e reduzir retrabalho".

Por exemplo, tolerâncias como H7/g6 frequentemente utilizadas em montagens de precisão permitem "controlar os níveis aceitáveis de folga ou interferência para garantir desempenho confiável" (ISO 286-1, 2010).

**2.5. Desenvolvimento de Aplicativos com Flutter e IoT**

A utilização do Flutter para desenvolvimento de aplicativos industriais é crescente devido à sua flexibilidade e capacidade multiplataforma. Conforme explica Hammad et al. (2021), "o Flutter possibilita integração com dispositivos IoT para monitoramento e controle em tempo real, criando interfaces altamente responsivas e intuitivas".

Além disso, projetos industriais que empregam IoT frequentemente requerem aplicativos que possam se comunicar com sensores e controladores, como destaca Brown (2022): "a integração de tecnologias emergentes como IoT é essencial para otimizar processos industriais e aumentar a confiabilidade dos sistemas".

**2.6. Uso do Microcontrolador ESP32 com Sensor DHT22**

O ESP32 é um microcontrolador amplamente utilizado em aplicações IoT devido à sua alta capacidade de processamento e conectividade Wi-Fi e Bluetooth. Quando combinado com o sensor DHT22, o sistema permite medições precisas de temperatura e umidade. Segundo Espressif Systems (2020), "o ESP32 combina conectividade Wi-Fi e Bluetooth com capacidade de processamento elevada, ideal para aplicações que exigem comunicação em tempo real".

Estudos, como o de Souza et al. (2021), destacam que "a integração do ESP32 com sensores ambientais permite a implementação de sistemas de medição confiáveis e com custo acessível, otimizando processos industriais críticos".

**3 METODOLOGIA**

Na indústria existem várias hipóteses que podem influenciar nos controles dimensionais, em específico a qualidade de como os dados são coletados, dentre os maiores riscos estão no instrumento de medição e na expertise da mão de obra responsável pela análise dimensional.

Este projeto tem a finalidade de beneficiar os responsáveis por controlar características dimensionais considerando a temperatura ambiente, evitando uso de tabelas técnicas de coeficientes de dilatação, evitando também o erro de cálculo da dimensão em ambientes com temperatura diferentes da recomendação da norma ISO1:1997.

A primeira etapa foi definir um logo e telas do aplicativo sendo a tela inicial de login e tela de cálculo da dimensão com base no fator de temperatura (figura1).

Figura 1 – Telas da Aplicação

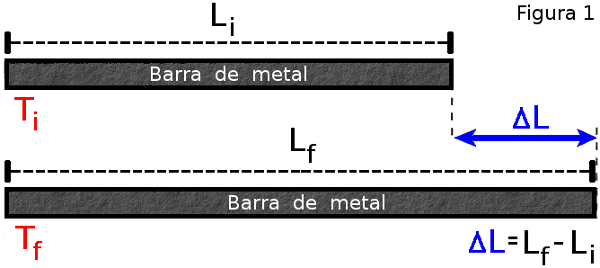
Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: autoria própria

Para fundamentar a base de cálculos de dilatação térmica a pesquisa literária foi importante para conhecer e aplicar o cálculo corretamente, pois existem três tipos de conceitos sobre dilatação térmica, sendo volumétrica, superficial e linear, e para o objetivo do projeto a base de cálculo da aplicação é dilatação térmica linear (figura 2).

Figura 2 – Dilatação Térmica Linear



Fonte: https://vamosfalardefisica.wordpress.com/2020/05/06/dilatacao-termica/

O cálculo de dilatação térmica linear é dado na (figura 3).

Figura 3 – Fórmula de Variação do comprimento

[Forma

Descrição gerada automaticamente com confiança média](https://www.algosobre.com.br/images/stories/fisica/dilatacao_04.gif)

Fonte: https://vamosfalardefisica.wordpress.com/2020/05/06/dilatacao-termica/

Onde,

ΔL:Variaçãodo comprimento  
L0: Comprimento inicial  
α: Coeficiente de dilatação linear (°C-1)  
ΔT: Variação de temperatura (°C)

Após a definição do método de cálculo de dilatação o código do aplicativo foi desenvolvido no Flutter, com o objetivo de coletar dados de um ESP32 com sensor DHT22 conectado em rede local, conforme anexo 1.

A tela inicial, tem o objetivo de identificar apenas usuário cadastrado, e o campo de IP do ESP32 deve ser inserido pelo usuário para estabelecer conexão por rede local, esta tela é denominada como tela de login (figura4).

Figura 4 – Tela inicial login

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: autoria própria

Após a autenticação pela tela de login, o aplicativo transita para segunda tela denominada com cálculo de dilatação térmica, nesta tela o usuário pode selecionar o tipo de material, pois cada material tem seu coeficiente de dilatação, por se tratar de um protótipo foi cadastrado três materiais, sendo o aço, alumínio e cobre.

Os campos de temperatura e umidade serão preenchidos pelos dados coletados pelo ESP32 do sensor DHT22, onde a variável temperatura será aplicada para o cálculo de variação de temperatura, que será a diferença da temperatura atual para a temperatura de referência de 20°C designado na norma NBR ISO 1:1997.

No campo dimensão especificada, o usuário insere o valor encontrado na medição realizada em ambientes onde a temperatura não esteja sendo controlada, para posteriormente calcular a medida teórica considerando a diferença de temperatura em ambiente controlado a 20°C.

Figura 5 – Tela cálculo de dilatação térmica

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Fonte: autoria própria

**4 RESULTADOS E DISCUSSÕES**

O desenvolvimento do aplicativo foi baseado em um contexto de problema dentro de processos de fabricação, que é realizar controle dimensional com efeito de dilatação do material em que a peça é fabricada. Com o uso do IoT nesta aplicação minimiza o erro de cálculo do operador, pois o aplicativo estará consumindo a temperatura exata no ambiente.

Com base em auxiliar o controle dimensional em ambientes de produção o aplicativo atendeu as expectativas e funcionalidades propostas, realizamos os testes de cálculo com três diferentes materiais e o resultado auxilia na interpretação do resultado, permitindo o operador melhorar a confiabilidade das dimensões obtidas nos processos de fabricação.

**5 CONCLUSÃO**

As premissas do projeto de cálculo de dimensão teórica, com base no fenômeno físico de dilatação do material foram atendidas nos testes realizados no flutter.

A aplicação tem possibilidades de melhorias e aproveitamento de envio dos dados calculados para efeito de controle estatístico de processo.

**6 REFERÊNCIAS**

* Associação Brasileira de Normas Técnicas. NBR ISO 1:1997. Temperatura padrão para medições industriais. ABNT, 1997.
* Brown, P. IoT for Industrial Applications. Springer, 2022.
* Callister, W. D. Materials Science and Engineering: An Introduction. Wiley, 2019.
* Espressif Systems. ESP32 Datasheet. Disponível em: <https://www.espressif.com>.
* Hammad, M., Khan, M., & Zafar, N. Multiplatform App Development Using Flutter. IEEE, 2021.
* Kalpakjian, S., & Schmid, S. Manufacturing Engineering and Technology. Pearson, 2020.
* Neves, J., & Lima, R. Ajustes e Tolerâncias: Fundamentos e Aplicações Industriais. LTC, 2017.
* Smith, J. et al. Thermal Effects on Precision Engineering. Journal of Advanced Manufacturing, 2018.
* Souza, F., et al. Applications of ESP32 in Smart Manufacturing. Sensors, 2021.

**ANEXO 1 – Código do aplicativo em Flutter**

import 'package:flutter/material.dart';

import 'dart:convert'; // Para processar JSON

import 'package:http/http.dart' as http; // Para requisições HTTP

import 'dart:async'; //Para usar o Timer

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

home: LoginScreen(),

);

}

}

class LoginScreen extends StatefulWidget {

@override

LoginScreenState createState() => LoginScreenState();

}

class LoginScreenState extends State<LoginScreen> {

final TextEditingController ipController = TextEditingController(); // Controller para o IP do ESP32

@override

Widget build(BuildContext context) {

return Scaffold(

backgroundColor: Colors.grey[300],

body: Center(

child: Padding(

padding: const EdgeInsets.symmetric(horizontal: 20.0),

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Image.asset(

'assets/sqa\_logo.png', // logo da SQA

height: 120,

),

SizedBox(height: 50),

TextField(

decoration: InputDecoration(

labelText: 'Login:',

border: OutlineInputBorder(

borderRadius: BorderRadius.circular(8),

),

),

),

SizedBox(height: 10),

TextField(

obscureText: true,

decoration: InputDecoration(

labelText: 'Senha:',

border: OutlineInputBorder(

borderRadius: BorderRadius.circular(8),

),

),

),

SizedBox(height: 10),

// Campo para inserir o IP do ESP32

TextField(

controller: ipController,

keyboardType: TextInputType.text,

decoration: InputDecoration(

labelText: 'IP do ESP32:',

hintText: 'Ex.: 192.168.xxx.xxx',

border: OutlineInputBorder(

borderRadius: BorderRadius.circular(8),

),

),

),

SizedBox(height: 20),

SizedBox(

width: double.infinity,

child: ElevatedButton(

onPressed: () {

if (ipController.text.isNotEmpty) {

Navigator.push(

context,

MaterialPageRoute(

builder: (context) =>

CalculationScreen(espIp: ipController.text),

),

); // Passa o IP do ESP32 para a próxima tela

} else {

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(

content: Text("Por favor, insira o IP do ESP32."),

),

);

}

},

style: ElevatedButton.styleFrom(

backgroundColor: Colors.green,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(8),

),

padding: EdgeInsets.symmetric(vertical: 16),

),

child: Text(

'ENTRAR',

style: TextStyle(

color: Colors.white,

fontSize: 18,

fontWeight: FontWeight.bold,

),

),

),

),

],

),

),

),

);

}

}

class CalculationScreen extends StatefulWidget {

final String espIp; // Recebe o IP do ESP32

CalculationScreen({required this.espIp});

@override

CalculationScreenState createState() => CalculationScreenState();

}

class CalculationScreenState extends State<CalculationScreen> {

String selectedMaterial = 'Aço 0.000012';

double temperature = 0.0;

double humidity = 0.0;

double initialDimension = 0.0;

double calculatedDimension = 0.0;

Timer? timer; // Timer para atualizações periódicas

// Constrói a URL do ESP32 com base no IP recebido

String get espUrl => "http://${widget.espIp}/sensor";

// Função para buscar os dados do ESP32

Future<void> fetchSensorData() async {

try {

final uri = Uri.parse(espUrl);

print("Conectando ao ESP32 em: $uri");

final response = await http.get(uri);

if (response.statusCode == 200) {

final data = json.decode(response.body);

setState(() {

temperature = data['temperature'] ?? 0.0; // Define valores padrão se o JSON for inválido

humidity = data['humidity'] ?? 0.0;

});

print("Dados recebidos: Temperatura=${temperature}, Umidade=${humidity}");

} else {

print("Erro ao buscar os dados: ${response.statusCode} - ${response.reasonPhrase}");

}

} catch (e) {

print("Erro na requisição ao ESP32: $e");

}

}

void calculateDimension() {

double coefficient = double.parse(selectedMaterial.split(' ')[1]);

setState(() {

double deltaTemp = temperature - 20; // Temperatura de referência: 20°C

double deltaLength = initialDimension \* coefficient \* deltaTemp;

calculatedDimension = initialDimension + deltaLength;

});

}

@override

void initState() {

super.initState();

fetchSensorData(); // Busca os dados ao abrir a tela

// Atualiza os dados do ESP32 a cada 5 segundos

timer = Timer.periodic(Duration(seconds: 5), (\_) => fetchSensorData());

}

@override

void dispose() {

timer?.cancel(); // Cancela o Timer ao sair da tela

super.dispose();

}

@override

Widget build(BuildContext context) {

return Scaffold(

backgroundColor: Colors.grey[200],

appBar: AppBar(

backgroundColor: Colors.green,

title: Text('Cálculo de Dilatação Térmica'),

),

body: SingleChildScrollView(

padding: const EdgeInsets.all(16.0),

child: Column(

crossAxisAlignment: CrossAxisAlignment.center,

children: [

Center(

child: Image.asset(

'assets/sqa\_logo.png',

height: 100,

),

),

SizedBox(height: 20),

DropdownButtonFormField<String>(

value: selectedMaterial,

items: [

'Aço 0.000012',

'Alumínio 0.000023',

'Cobre 0.000017',

].map((material) {

return DropdownMenuItem(

value: material,

child: Text(material),

);

}).toList(),

onChanged: (value) {

setState(() {

selectedMaterial = value!;

});

},

decoration: InputDecoration(

labelText: 'Dilatação Térmica Material (mm/°C/m)',

border: OutlineInputBorder(),

),

),

SizedBox(height: 15),

Row(

children: [

Expanded(

child: TextField(

readOnly: true,

controller: TextEditingController(

text: temperature.toStringAsFixed(2),

),

decoration: InputDecoration(

labelText: 'Temperatura (°C)',

border: OutlineInputBorder(),

),

),

),

SizedBox(width: 10),

Expanded(

child: TextField(

readOnly: true,

controller: TextEditingController(

text: humidity.toStringAsFixed(2),

),

decoration: InputDecoration(

labelText: 'Umidade (%)',

border: OutlineInputBorder(),

),

),

),

],

),

SizedBox(height: 15),

TextField(

keyboardType: TextInputType.number,

decoration: InputDecoration(

labelText: 'Dimensão Especificada (mm)',

border: OutlineInputBorder(),

),

onChanged: (value) {

initialDimension = double.tryParse(value) ?? 0.0;

},

),

SizedBox(height: 20),

SizedBox(

width: double.infinity,

child: ElevatedButton(

onPressed: calculateDimension,

style: ElevatedButton.styleFrom(

backgroundColor: Colors.green,

padding: EdgeInsets.symmetric(vertical: 16),

),

child: Text(

'CALCULAR',

style: TextStyle(

color: Colors.white,

fontSize: 18,

fontWeight: FontWeight.bold,

),

),

),

),

SizedBox(height: 20),

TextField(

readOnly: true,

decoration: InputDecoration(

labelText: 'Valor Teórico (mm)',

border: OutlineInputBorder(),

),

controller: TextEditingController(

text: calculatedDimension.toStringAsFixed(4),

),

),

],

),

),

);

}

}

**ANEXO 2 – Código do ESP32 com sensor DHT22**

#include <WiFi.h>

#include <DHT.h>

#include <ESPAsyncWebServer.h>

// Configurações do Wi-Fi

const char\* ssid = "Rodrigo";

const char\* password = "12345678";

// Configurações do sensor DHT

#define DHTPIN 4 // Pino do sensor DHT22

#define DHTTYPE DHT22 // Tipo de sensor: DHT22

DHT dht(DHTPIN, DHTTYPE);

// Configuração do servidor HTTP

AsyncWebServer server(80);

// Variáveis globais para armazenar os valores do sensor

float currentTemperature = 0.0;

float currentHumidity = 0.0;

void setup() {

Serial.begin(115200);

dht.begin();

// Conexão Wi-Fi

WiFi.begin(ssid, password);

Serial.print("Conectando ao Wi-Fi");

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.print(".");

}

Serial.println("\nConectado ao Wi-Fi!");

Serial.print("Endereço IP: ");

Serial.println(WiFi.localIP());

// Rota para enviar os dados do sensor

server.on("/sensor", HTTP\_GET, [](AsyncWebServerRequest \*request) {

// Coleta os dados do sensor

float temperature = dht.readTemperature();

float humidity = dht.readHumidity();

// Verifica se os dados do sensor são válidos

if (isnan(temperature) || isnan(humidity)) {

// Resposta em caso de erro

AsyncWebServerResponse\* response = request->beginResponse(500, "application/json", "{\"error\":\"Erro ao ler o sensor\"}");

response->addHeader("Access-Control-Allow-Origin", "\*"); // Adiciona o cabeçalho CORS

request->send(response);

return;

}

// Gera a resposta JSON com os dados do sensor

String json = "{\"temperature\":" + String(temperature) + ",\"humidity\":" + String(humidity) + "}";

AsyncWebServerResponse\* response = request->beginResponse(200, "application/json", json);

response->addHeader("Access-Control-Allow-Origin", "\*"); // Adiciona o cabeçalho CORS

request->send(response);

});

// Inicia o servidor HTTP

server.begin();

}

void loop() {

// Atualiza os valores do sensor a cada 5 segundos

static unsigned long lastUpdateTime = 0;

if (millis() - lastUpdateTime >= 5000) { // Verifica se passaram 5 segundos

lastUpdateTime = millis();

// Lê os valores do sensor

float temperature = dht.readTemperature();

float humidity = dht.readHumidity();

// Atualiza apenas se os valores forem válidos

if (!isnan(temperature) && !isnan(humidity)) {

currentTemperature = temperature;

currentHumidity = humidity;

Serial.print("Temperatura: ");

Serial.print(currentTemperature);

Serial.print(" °C, Umidade: ");

Serial.print(currentHumidity);

Serial.println(" %");

} else {

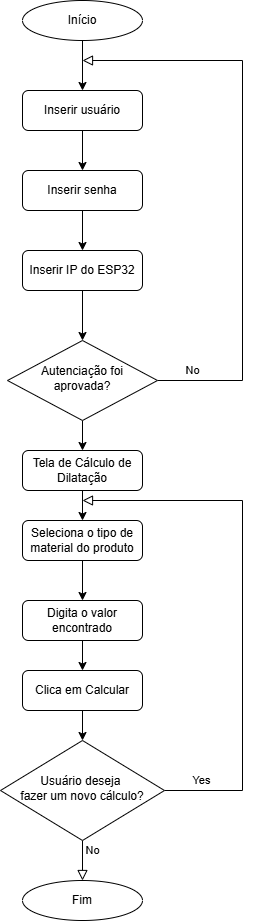
Serial.println("Erro ao ler os dados do sensor!");

}

}

}

**Anexo 3 – Fluxograma**

****