### **1º Objetivo: Pesquisa sobre Linguagens Utilizadas (Histórico e Manual)**

#### **Linguagens Identificadas**

* **C/C++**
  + **Histórico da Linguagem:**
    - **Ano de Criação:** C foi desenvolvido em 1972 por Dennis Ritchie na Bell Labs, e C++ foi criado em 1983 por Bjarne Stroustrup.
    - **Objetivo Inicial:** C foi projetado para desenvolver sistemas operacionais e software de sistema. C++ foi criado como uma extensão de C para incluir programação orientada a objetos.
    - **Principais Características:** C é uma linguagem de programação de baixo nível, eficiente e próxima do hardware, enquanto C++ adiciona recursos de orientação a objetos, como classes e herança.
  + **Documentação Oficial:**
    - C Programming Language
    - C++ Programming Language
  + **Justificativa de Uso:** C/C++ são amplamente utilizados para programação de microcontroladores, como Arduino e ESP32, devido à sua eficiência, controle de hardware e suporte em tempo real. A facilidade de integração com bibliotecas específicas para sensores e dispositivos torna essas linguagens ideais para o projeto.
* **HTML/JavaScript**
  + **Histórico da Linguagem:**
    - **Ano de Criação:** HTML foi criado por Tim Berners-Lee em 1991. JavaScript foi desenvolvido por Brendan Eich em 1995.
    - **Objetivo Inicial:** HTML foi projetado para estruturar conteúdo na web, enquanto JavaScript foi criado para adicionar interatividade e dinamicidade às páginas web.
    - **Principais Características:** HTML é uma linguagem de marcação que define a estrutura de uma página web. JavaScript é uma linguagem de programação que permite manipular o DOM, responder a eventos e realizar operações assíncronas.
  + **Documentação Oficial:**
    - HTML Documentation
    - JavaScript Documentation
  + **Justificativa de Uso:** HTML e JavaScript são essenciais para criar a interface web do projeto, permitindo a visualização e interação com os dados coletados pelos sensores. A compatibilidade com navegadores e a facilidade de uso tornam essas linguagens ideais para o desenvolvimento front-end.

#### **Relatório Estruturado**

* **Título:** Pesquisa sobre Linguagens Utilizadas no Projeto
* **Introdução:** Descrição do projeto e a importância das linguagens escolhidas.
* **Linguagens Utilizadas:** Seção detalhando cada linguagem, incluindo histórico, documentação e justificativa de uso.
* **Conclusão:** Resumo das escolhas feitas e a relevância das linguagens para o sucesso do projeto.

### **2º Objetivo: Primeira Funcionalidade Programada**

#### **Funcionalidade Inicial: Detecção de Vibração com SW-420**

**Código para a Funcionalidade:**

cpp

VerifyOpen In EditorEditCopy code

1const int vibrationPin = 2; // Pino do sensor de vibração  
2int vibrationState = 0; // Variável para armazenar o estado do sensor  
3  
4void setup() {  
5 Serial.begin(115200); // Inicializa a comunicação serial  
6 pinMode(vibrationPin, INPUT); // Define o pino do sensor como entrada  
7}  
8  
9void loop() {  
10 // Lê o estado do sensor de vibração  
11 vibrationState = digitalRead(vibrationPin);  
12   
13 // Envia o estado do sensor para o monitor serial  
14 if (vibrationState == HIGH) {  
15 Serial.println("Vibração detectada!"); // Mensagem quando a vibração é detectada  
16 } else {  
17 Serial.println("Sem vibração."); // Mensagem quando não há vibração  
18 }  
19  
20 delay(500); // Aguarda meio segundo antes da próxima leitura  
21}

#### **Organização do Código**

Salve o código na pasta **SRC** do projeto como **vibration\_detection.ino**.

O código está comentado para facilitar a compreensão de cada parte.

### **3º Objetivo: Função Principal Programada**

#### **Identificação da Função Principal**

**Função Principal:** Detectar impactos usando o MPU-6050 e transmitir dados para o monitor serial.

**Código para a Função Principal:**

cpp

VerifyOpen In EditorEditCopy code

1#include <Wire.h>  
2#include <MPU6050.h>  
3  
4MPU6050 mpu; // Cria um objeto MPU6050  
5int16\_t ax, ay, az; // Variáveis para armazenar os valores de aceleração  
6  
7void setup() {  
8 Serial.begin(115200); // Inicializa a comunicação serial  
9 Wire.begin(); // Inicia a comunicação I2C  
10 mpu.initialize(); // Inicializa o MPU-6050  
11}  
12  
13void loop() {  
14 // Lê os dados de aceleração do MPU-6050  
15 mpu.getAcceleration(&ax, &ay, &az);  
16   
17 // Envia os dados de aceleração para o monitor serial  
18 Serial.print("Aceleração X: ");  
19 Serial.print(ax);  
20 Serial.print(" | Aceleração Y: ");  
21 Serial.print(ay);  
22 Serial.print(" | Aceleração Z: ");  
23 Serial.println(az);  
24  
25 delay(500); // Aguarda meio segundo antes da próxima leitura  
26}

#### **Organização do Código**

Salve o código na pasta **SRC** do projeto como **impact\_detection.ino**.

O código está comentado para facilitar a compreensão de cada parte.

### **4º Objetivo: Funções Totalmente Programadas**

#### **Programação de Todas as Funções do Projeto**

Finalize a implementação de todas as funcionalidades planejadas, incluindo sensores, comunicação entre dispositivos e transmissão de dados para a aplicação web.

**Código Exemplo para Funções Completas:**

cpp

VerifyOpen In EditorEditCopy code

1#include <Wire.h>  
2#include <MPU6050.h>  
3  
4MPU6050 mpu; // Cria um objeto MPU6050  
5const int vibrationPin = 2; // Pino do sensor de vibração  
6int vibrationState = 0; // Variável para armazenar o estado do sensor  
7int16\_t ax, ay, az; // Variáveis para armazenar os valores de aceleração  
8  
9void setup() {  
10 Serial.begin(115200); // Inicializa a comunicação serial  
11 Wire.begin(); // Inicia a comunicação I2C  
12 mpu.initialize(); // Inicializa o MPU-6050  
13 pinMode(vibrationPin, INPUT); // Define o pino do sensor de vibração como entrada  
14}  
15  
16void loop() {  
17 // Lê o estado do sensor de vibração  
18 vibrationState = digitalRead(vibrationPin);  
19   
20 // Envia o estado do sensor de vibração para o monitor serial  
21 if (vibrationState == HIGH) {  
22 Serial.println("Vibração detectada!"); // Mensagem quando a vibração é detectada  
23 } else {  
24 Serial.println("Sem vibração."); // Mensagem quando não há vibração  
25 }  
26  
27 // Lê os dados de aceleração do MPU-6050  
28 mpu.getAcceleration(&ax, &ay, &az);  
29   
30 // Envia os dados de aceleração para o monitor serial  
31 Serial.print("Aceleração X: ");  
32 Serial.print(ax);  
33 Serial.print(" | Aceleração Y: ");  
34 Serial.print(ay);  
35 Serial.print(" | Aceleração Z: ");  
36 Serial.println(az);  
37  
38 delay(500); // Aguarda meio segundo antes da próxima leitura

39)

#### **Comentar e Otimizar**

* Certifique-se de que todos os trechos do código estão comentados e revisados para eficiência.
* Salve todos os arquivos na pasta **SRC**, organizados por função, com explicações claras e sem redundâncias.