



*Serviço Nacional de Aprendizagem Industrial*

**PELO FUTURO DO TRABALHO**

# Safe Vibration

## Sensor de Vibração

Junho/2023  
**Florianópolis/SC**



## SUMÁRIO

1. INTRODUÇÃO.....	5
1.1. POR QUE UTILIZAR?.....	5
1.2. COMO A VIBRAÇÃO PODE AFETAR A ESTRUTURA?.....	5
2. REQUISITOS FUNCIONAIS.....	6
2.1. COLETA DE DADOS DE VIBRAÇÃO EM TEMPO REAL.....	6
2.2. CÁLCULO DE OUTLIERS (PONTOS FORA DA CURVA) .....	6
2.3. EXIBIÇÃO DE DADOS COLETADOS.....	6
2.4. PERMISSÃO DO ENVIO DE DADOS COLETADOS PARA SERVIDOR EXTERNO.....	7
3. REQUISITOS NÃO FUNCIONAIS.....	7
3.1. RESISTENTE A INTERFERÊNCIAS EXTERNAS.....	7
3.2. BAIXO CONSUMO DE ENERGIA.....	7
4. OBJETIVOS.....	8
5. DESENVOLVIMENTO.....	9
5.1 HISTÓRICO DE ALTERAÇÕES.....	9
5.2. SENSORES.....	9
5.2.1. MPU-6050.....	9
5.3. ATUADORES.....	10
5.3.1. BUZZER PCI 12MM.....	10
5.3.2 LED.....	10
5.4. INSTALAÇÃO DO SISTEMA.....	10

---

5.5. PROGRAMAÇÃO DO SISTEMA.....	11
5.6. PROTOCOLO E PLATAFORMA USADOS.....	26
5.7. INTEGRAÇÃO.....	27
5.8. DIAGRAMAS.....	28
5.9. DASHBOARD.....	30
5.9.1.....	30
5.9.2.....	31
5.9.3.....	31
6.CONCLUSÃO.....	32
7.REFERÊNCIAS .....	33

## **1. INTRODUÇÃO**

A Safe Vibration é um projeto criado para desempenhar a função de realizar um monitoramento da vibração de uma estrutura a longo e curto prazo através da leitura da mesma pela utilização de um sensor que nos traz os dados necessários para tal realização.

A ideia é fazer este monitoramento da forma mais precisa o possível, porém com um baixo custo de produção, pois a questão é que esse tipo de monitoramento já pode ser realizado por outros aparelhos, entretanto eles são de alto custo, com alguns modelos podendo chegar a mais de R\$9.000,00 e são dispositivos que estão fadados a uma possível destruição, pois podem sofrer perante um desabamento de alguma estrutura ou deslizamento de algum morro.

### **1.1 POR QUE UTILIZAR?**

Um sensor de vibração é uma ferramenta importante para prevenir danos estruturais em edifícios. Ele oferece monitoramento contínuo, identificação de fontes de vibração, prevenção de danos, economia de custos a longo prazo e ajuda a cumprir normas de segurança. Utilizar sensores de vibração permite identificar e lidar com problemas antes que se tornem graves, economizando tempo além de dinheiro, garantindo a segurança dos ocupantes e cumprindo as regulamentações aplicáveis.

### **1.2 COMO A VIBRAÇÃO PODE AFETAR A ESTRUTURA?**

Já sabemos que a vibração afeta a estrutura, a questão agora é como ela faz isso. Em poucas palavras todo o objeto possui uma vibração natural, que podem ser atingidos por uma vibração externa. que pode vir de diversas formas, como os geradores citados no desastre acima, pela ação humana ou ainda pela ação da própria natureza através de terremotos, dentre outros, e quando a frequência externa se aproxima da frequência natural do objeto a frequência do objeto amplifica, causando assim uma ressonância, o'que pode danificá-lo. Um exemplo mais conhecido por nós sobre esse evento é quando uma pessoa consegue chegar em um tom agudo sobre um cálice de cristal, o mesmo se quebrar por causa da ressonância .

## 2. REQUISITOS FUNCIONAIS

### 2.1. COLETA DE DADOS DE VIBRAÇÃO EM TEMPO REAL

**Prioridade:** ☒ **Essencial** ☐ **Importante** ☐ **Desejável**

O sistema deve captar os dados de vibração do sensor MPU 6050, que estará conectado à placa ESP32. Sendo os dados, especificamente, da aceleração nos eixos X,Y e Z.

### 2.2. CÁLCULO DE OUTLIERS (PONTOS FORA DA CURVA)

**Prioridade:** ☒ **Essencial** ☐ **Importante** ☐ **Desejável**

O sistema deve captar os dados de aceleração dos eixos do sensor MPU 6050, que estará conectado à placa ESP32. Com os valores de aceleração nos eixos (X, Y e Z), e a fórmula do Z Score, o sistema detecta qualquer alteração na aceleração e, em caso de alteração extrema, emite sinalização luminosa e sonora para alertar.

### 2.3. EXIBIÇÃO DE DADOS COLETADOS

**Prioridade:** ☒ **Essencial** ☐ **Importante** ☐ **Desejável**

A placa tem conexão com a internet, o que possibilita a comunicação com um software através de requisições pré definidas. O software Tago.io é onde acontece a criação e exibição do dashboard com os dados captados pelo sensor MPU 6050.

## **2.4. PERMISSÃO DE ENVIO DE DADOS COLETADOS PARA SERVIDOR EXTERNO**

**Prioridade:**      ☒ **Essencial**      ☐ **Importante**      ☐ **Desejável**

Como dito anteriormente, a placa ESP32 tem conexão com a internet, o que nos possibilita fazer conexões com servidores externos, e através de requisições de tipo “get” nós pegamos dados da placa para o servidor.

## **3. REQUISITOS NÃO FUNCIONAIS**

### **3.1. RESISTENTE A INTERFERÊNCIAS EXTERNAS**

A ESP32 possui um recurso exclusivo em termos de segurança: um acelerador de hardware que oferece suporte a algoritmos de criptografia, como AES, SHA, RSA e ECC. Essa funcionalidade permite a criação de aplicações seguras e a implementação de um sistema de atualização de firmware e software da ESP32 no modo OTA (Over-The-Air), garantindo a segurança dos dados durante o processo.

### **3.2. BAIXO CONSUMO DE ENERGIA**

O ESP32 tem um baixo consumo de energia devido a recursos como modos de suspensão, um processador de baixa potência, gerenciamento inteligente de energia, aceleradores de hardware e opções de conectividade otimizadas. Esses recursos permitem que o chip ajuste dinamicamente o consumo de energia, desligue partes não utilizadas, execute tarefas de forma eficiente e economize energia durante a comunicação sem fio.

#### **4. OBJETIVOS**

Quando falamos em vibração, não podemos subestimar seu poder de atuar sobre as estrutura, um exemplo que poderia ter sido evitado caso houvesse aparelhos monitorando o local, onde muitas mortes poderiam ter sido poupadas ou o próprio desastre em si, foi em 2013, um prédio em Bangladesh que abrigava várias fábricas de tecidos desabou devido a vibração de geradores que foram instalados no teto do edifício, assim acabou causando cerca de 500 mortes.

Outro exemplo é de um edifício nos Estados Unidos em que um prédio desabou após a construção de um mega condomínio ao lado dele causou danos estruturais devido às vibrações da obra e além da falta de manutenção na estrutura culminou na tragédia que tirou a vida de 98 pessoas e poderia ser facilmente evitada.

Estes exemplos nos mostram o quão importante é haver um monitoramento das estruturas em relação à exposição de uma vibração externa, e é isso que estamos buscando aqui, fazer uma aplicação eficaz e de baixo custo.



## 5. DESENVOLVIMENTO

### 5.1 HISTÓRICO DE ALTERAÇÕES

Data	Versão	Descrição	Autor
05/06/2023	1.1	Criação deste documento	Ricardo
23/05/2023	1.2	Início do código	Giulia
29/05/2023	1.3	Atuadores recebendo sinais do Tago.io	Guilherme
31/05/2023	1.4	Configuração do buzzer	Guilherme
07/06/2023	1.5	Organização do código e adição de alguns módulos além de alterações gerais	Caio
10/06/2023	1.6	Adição dos módulos do índice 5	Giulia
11/06/2023	1.7	Formatações gerais	Ricardo
12/06/2023	1.8	Nova fórmula usada para o sensor MPU6050	Guilherme
13/06/2023	2.0	Nova fórmula usada para o sensor MPU6050 e integração de um botão	Caio
15/06/2023	3.0	Nova fórmula usada para o sensor MPU6050	Guilherme
20/06/2023	3.1	Ajustes finais	Caio,Guilherme e Giulia

### 5.2. SENSORES

#### 5.2.1. MPU-6050

O sensor MPU-6050 é um dispositivo eletrônico que contém em um único chip um acelerômetro e um giroscópio tipo MEMS. Com 3 eixos para o acelerômetro e 3 eixos para o giroscópio, o sensor oferece 6 graus de liberdade (6DOF). Além disso, a placa inclui um sensor de temperatura embutido no CI MPU6050, o que permite medições de temperatura entre -40 e +85 °C. Com um conversor analógico digital de 16-bits para cada canal, o MPU-6050 oferece alta precisão, capturando os canais X, Y e Z ao mesmo tempo. O sensor

é frequentemente utilizado em aplicações de controle de movimento, como em drones, robôs e jogos eletrônicos, onde a precisão é essencial para o bom desempenho das tarefas.

### **5.3. ATUADORES**

#### **5.3.1. BUZZER PCI 12MM**

O buzzer pci 12mm é um alto-falante compacto feito com o objetivo de emanar sinais sonoros a partir de um fornecimento de energia para o módulo sem varia a frequência da emissão.

É um sistema de transmissão eletrônico básico feito de transdutores eletrônicos que é usado amplamente em sistemas de alerta, computadores, projetos robóticos e etc.

O objetivo principal é emitir sinais sonoros como um aviso para deixar o operador saber que algo está acontecendo. O buzzer é ativado pelo painel equipado com um micro controlador, que deve ser programado para suprir energia para o buzzer em resposta para o evento específico para que ele possa oferecer um sinal de alerta para o operador.

O módulo é compatível com a maioria de sistemas de micro controlador, incluindo o Arduino, AVR, Rasberry PI e muitos outros.

#### **5.3.2 LED**

O led é um tipo de diodo que quando passa uma corrente elétrica por ele, ocorre a emissão de luz, o diodo é um componente eletrônico que conduz corrente elétrica em apenas uma polarização.

### **5.4. INSTALAÇÃO DO SISTEMA**

A construção deste dispositivo foi realizada sob uma placa protoboard. A placa ESP32 foi instalada na linha g, com suas conexões instaladas nos pinos 11 ao 30 da protoboard.

Conectados aos pinos g12 e g14 existem resistores de  $330\mu\Omega$ , cada resistor está conectado ao pino positivo de um dos Leds respectivos a sua entrada digital, o pino negativo do dos Leds estão conectados a linha de conexões negativas da protoboard, que por sua vez tem uma conexão ao pino ground da placa Esp32.

Sobre a conexão do sensor MPU6050 com o microcontrolador, seu pino de alimentação está ligado no VCC da placa, que emite 5v de energia para o sensor, a conexão gnd do sensor, está conectado a linhas de conexões negativas da protoboard, e suas saídas digitais estão conectadas nos pinos digitais 32 e 33 da placa Esp.

O buzzer tem duas conexões simples na protoboard, sendo uma digital conectado ao pino g26 da placa, já em sua conexão negativa, está conectado ao através de um fio, a linha negativa da placa protoboard.

## 5.5. PROGRAMAÇÃO DO SISTEMA

```
//Envio de Dados para tagoIO via MQTT

#include <ArduinoJson.h>
#include "EspMQTTClient.h"
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <math.h>
#include <Wire.h>

//pinos de entrada e saída
Adafruit_MPU6050 mpu;

//declaração das variáveis do sensor mpu6050
const uint8_t scl = 32;
const uint8_t sda = 33;

//coisas do X
float mediaX, mediaY, mediaZ = 0;
float desvioPadraoX, desvioPadraoY, desvioPadraoZ = 0;
```

```
float zScoreX, zScoreY, zScoreZ = 0;

int j = 0;

//declaração do buzzer
const int buzzer = 26;

//declaração de vetor armazenamento de dados
float ArrayEixoX[10], ArrayEixoY[10], ArrayEixoZ[10];
;
//declaração dos leds
const int ledBranco = 27;
const int ledAmarelo = 14;
const int ledVermelho = 12;

//variáveis para Json
char json_ax[100];
char json_ay[100];
char json_az[100];

char json_zScoreX[100];
char json_zScoreY[100];
char json_zScoreZ[100];

//variáveis internas
const uint8_t MPU6050SlaveAddress = 0x68;

const float AccelScaleFactor = 4096;

const uint16_t GyroScaleFactor = 131;

const uint8_t MPU6050_REGISTER_SMPLRT_DIV = 0x19;

const uint8_t MPU6050_REGISTER_USER_CTRL = 0x6A;
```

```
const uint8_t MPU6050_REGISTER_PWR_MGMT_1 = 0x6B;

const uint8_t MPU6050_REGISTER_PWR_MGMT_2 = 0x6C;

const uint8_t MPU6050_REGISTER_CONFIG = 0x1A;

const uint8_t MPU6050_REGISTER_GYRO_CONFIG = 0x1B;

const uint8_t MPU6050_REGISTER_ACCEL_CONFIG = 0x1C;

const uint8_t MPU6050_REGISTER_FIFO_EN = 0x23;

const uint8_t MPU6050_REGISTER_INT_ENABLE = 0x38;

const uint8_t MPU6050_REGISTER_ACCEL_XOUT_H = 0x3B;

const uint8_t MPU6050_REGISTER_SIGNAL_PATH_RESET = 0x68;

float AccelX, AccelY, AccelZ, GyroX, GyroY, GyroZ;

uint8_t Temperature;

int i, y, z = 0;

float Ax, Ay, Az, T, Gx, Gy, Gz;

//gap para ter um intervalo entre os sons do buzzer
int gap=1000;

const int SAMPLING_FREQ = 100;

//configurações da conexão MQTT
EspMQTTClient client
```

```
(  
  "FIESC_IOT", //nome da sua rede Wi-Fi  
  "C6qnM4ag81", //senha da sua rede Wi-Fi  
  "mqtt.tago.io", // MQTT Broker server ip padrão da tago  
  "Token", // username  
  "b3ed8e3e-f9e7-4945-8eae-a475f3a5bc17", // Código do Token  
  "SafeVibration", // Client name that uniquely identify your device  
  1883 // The MQTT port, default to 1883. this line can be omitted  
);  
  
//conectar o ESP32 a internet  
void I2C_Write(uint8_t deviceAddress, uint8_t regAddress, uint8_t data){  
  
  Wire.beginTransmission(deviceAddress);  
  
  Wire.write(regAddress);  
  
  Wire.write(data);  
  
  Wire.endTransmission();  
  
}  
  
//configuração dos pinos  
void setup()  
{  
  Serial.begin(115200);  
  
  Wire.begin(sda, scl);  
  
  MPU6050_Init();  
  
  pinMode(27, OUTPUT);  
  
  pinMode(14,OUTPUT);
```

```
pinMode(12,OUTPUT);

pinMode(buzzer, OUTPUT);

mpu.setAccelerometerRange(MPU6050_RANGE_8_G);

// mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
}

//loop do programa
void loop()
{

    leitura_sinais();

    converte_json();
    envia_msg();

    delay(1000);
    if(ArrayEixoZ[9] != NULL){
        emiteAlerta();
    }

    client.loop();
}

//inicializa o mpu6050, são protocolos de comunicação entre a ESP32 e o mpu6050
void MPU6050_Init(){

    delay(150);

    I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_SMPLRT_DIV, 0x07);
```

```
I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_PWR_MGMT_1,
0x01);

I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_PWR_MGMT_2,
0x00);

I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_CONFIG, 0x00);

I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_GYRO_CONFIG,
0x00);

I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_ACCEL_CONFIG,
0x00);

I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_FIFO_EN, 0x00);

I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_INT_ENABLE,
0x01);

I2C_Write(MPU6050SlaveAddress,
MPU6050_REGISTER_SIGNAL_PATH_RESET, 0x00);

I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_USER_CTRL, 0x00);

}

//lendo os valores do mpu6050
void Read_RawValue(uint8_t deviceAddress, uint8_t regAddress){

Wire.beginTransmission(deviceAddress);

Wire.write(regAddress);

Wire.endTransmission();
```



```
Wire.requestFrom(deviceAddress, (uint8_t)14);

AccelX = (((int16_t)Wire.read()<<8) | Wire.read());

AccelY = (((int16_t)Wire.read()<<8) | Wire.read());

AccelZ = (((int16_t)Wire.read()<<8) | Wire.read());

}

float calculaMedia(float Array[10]){
    float acumulador = 0;
    float media = 0;

    for(int count = 0; count <=9; count++){
        acumulador += Array[count];
    }

    media = acumulador/10;
    return(media);
}

float calculaDesvioPadrao(float Array[10], float media){
    float diff = 0;
    float somaDiff = 0;
    float desvioPadrao = 0;

    for(int k = 0; k <=9; k++){
        diff = Array[k] - media;
        somaDiff += diff * diff;
    }

    desvioPadrao = sqrt(somaDiff / 10);
    return(desvioPadrao);
}

float calculaZscore(float valor, float media, float desvioPadrao){
    float zScore = 0;
```

```
        zScore = (valor - media) / desvioPadrao;
        return(zScore);
    }

    //tratando os dados do mpu6050 e mostrando eles no serial
    void leitura_sinais()
    {
        Read_RawValue(MPU6050SlaveAddress, MPU6050_REGISTER_ACCEL_XOUT_H);

        Ax = ((double)AccelX/AccelScaleFactor);
        Ay = ((double)AccelY/AccelScaleFactor);
        Az = ((double)AccelZ/AccelScaleFactor);

        if(i <= 9){
            ArrayEixoX[i] = Ax;
            i++;
        }
        else{
            i = 0;
        }

        if(y <= 9){
            ArrayEixoY[y] = Ay;
            y++;
        }
        else{
            y = 0;
        }

        if(z <= 9){
            ArrayEixoZ[z] = Az;
            z++;
        }
        else{
            z = 0;
        }
    }
}
```

```
}

if(ArrayEixoX[9] != NULL){
    mediaX = calculaMedia(ArrayEixoX);
    desvioPadraoX = calculaDesvioPadrao(ArrayEixoX ,mediaX);
    zScoreX = calculaZscore(Ax, mediaX, desvioPadraoX);
    Serial.print("zScoreX = ");
    Serial.print(" ");
    Serial.println(zScoreX);

}

if(ArrayEixoY[9] != NULL){
    mediaY = calculaMedia(ArrayEixoY);
    desvioPadraoY = calculaDesvioPadrao(ArrayEixoY ,mediaY);
    zScoreY = calculaZscore(Ay, mediaY, desvioPadraoY);
    Serial.print("zScoreY = ");
    Serial.print(" ");
    Serial.println(zScoreY);

}

if(ArrayEixoZ[9] != NULL){
    mediaZ = calculaMedia(ArrayEixoZ);
    desvioPadraoZ = calculaDesvioPadrao(ArrayEixoZ ,mediaZ);
    zScoreZ = calculaZscore(Az, mediaZ, desvioPadraoZ);
    Serial.print("zScoreZ = ");
    Serial.print(" ");
    Serial.println(zScoreZ);
}

delay(1000 / SAMPLING_FREQ);

Serial.print("Ax: "); Serial.print(Ax);
Serial.print(" Ay: "); Serial.print(Ay);
Serial.print(" Az: "); Serial.println(Az);
```

```
}

// funções zap1, zap2, risefall, fall, rise, twotone, são referentes ao som do buzzer

//-----
void zap1()
{
    for (float f=3000;f>40;f=f*0.93){
        tone(buzzer,f);
        delay(10);
    }
}

void zap2()
{
    for (float f=3000;f>10;f=f*0.85){
        tone(buzzer,2*f);
        delay(5);
        tone(buzzer,f);
        delay(5);
    }
}

void risefall()
{
    float rise_fall_time=180;
    int steps=50;
    float f_max=2600;
    float f_min=1000;
    float delay_time=rise_fall_time/steps;
    float step_size=(f_max-f_min)/steps;
    for (float f=f_min;f<f_max;f+=step_size){
        tone(buzzer,f);
        delay(delay_time);
    }
}
```

```
    for (float f=f_max;f>f_min;f-=step_size){
        tone(buzzer,f);
        delay(delay_time);
    }
}

void fall(float rise_fall_time)
{
    int steps=50;
    float f_max=2000;
    float f_min=500;
    float delay_time=rise_fall_time/steps;
    float step_size=0.97;
    for (float f=f_max;f>f_min;f*=step_size){
        tone(buzzer,f);
        delay(delay_time);
    }
}

void rise()
{
    float rise_fall_time=2000;
    int steps=100;
    float f_max=1500;
    float f_min=500;
    float delay_time=rise_fall_time/steps;
    float step_size=1.012;
    for (float f=f_min;f<f_max;f*=step_size){
        tone(buzzer,f);
        delay(delay_time);
    }
    noTone(buzzer);
    delay(100);
}

void twotone()
```

```
{
    float f_max=1500;
    float f_min=1000;
    float delay_time=800;
    tone(buzzer,f_max);
    delay(delay_time);
    tone(buzzer,f_min);
    delay(delay_time);

}

//-----

// converte os dados de leitura_sinais para JSON
void converte_json()
{
    StaticJsonDocument<300> sjson_ax;
    StaticJsonDocument<300> sjson_ay;
    StaticJsonDocument<300> sjson_az;
    StaticJsonDocument<300> sjson_zScoreX;
    StaticJsonDocument<300> sjson_zScoreY;
    StaticJsonDocument<300> sjson_zScoreZ;

    sjson_ax["variable"] = "ax";
    sjson_ax["value"] = Ax;
    serializeJson(sjson_ax, json_ax);

    sjson_ay["variable"] = "ay";
    sjson_ay["value"] = Ay;
    serializeJson(sjson_ay, json_ay);

    sjson_az["variable"] = "az";
    sjson_az["value"] = Az;
    serializeJson(sjson_az, json_az);
```

```
    sjson_zScoreX["variable"] = "zScoreX";
    sjson_zScoreX["value"] = zScoreX;
    serializeJson(sjson_zScoreX, json_zScoreX);

    sjson_zScoreY["variable"] = "zScoreY";
    sjson_zScoreY["value"] = zScoreY;
    serializeJson(sjson_zScoreY, json_zScoreY);

    sjson_zScoreZ["variable"] = "zScoreZ";
    sjson_zScoreZ["value"] = zScoreZ;
    serializeJson(sjson_zScoreZ, json_zScoreZ);

}

void emiteAlerta() {
    if(zScoreX >= 3 || zScoreX < -3 || zScoreY >= 3 || zScoreY < -3 || zScoreZ >= 3 ||
zScoreZ < -3){
        digitalWrite(ledVermelho, HIGH);
        digitalWrite(buzzer, HIGH);
        digitalWrite(ledAmarelo, LOW);
        digitalWrite(ledBranco, LOW);
        for (int count=1;count<=10;count++)
        {
            risefall();
        }
        noTone(buzzer);
        delay(gap);
        for (int count=1;count<=10;count++)
        {
            fall(300);
        }
        noTone(buzzer);
        delay(gap);
        for (int count=1;count<=5;count++)
        {
```

```
        fall(600);
    }
    noTone(buzzer);
    delay(gap);
    for (int count=1;count<5;count++)
    {
        rise();
    }
    noTone(buzzer);
    delay(gap);
    for (int count=1;count<5;count++)
    {
        twotone();
    }
    noTone(buzzer);
    delay(gap);
    for (int count=1;count<10;count++)
    {
        zap1();
    }
    noTone(buzzer);
    delay(gap);
    for (int count=1;count<10;count++)
    {
        zap2();
    }
    noTone(buzzer);
    digitalWrite(ledBranco, LOW);
    digitalWrite(ledAmarelo, LOW);
    delay(gap);

}
```



```
    else{
        digitalWrite(ledBranco, HIGH);
        digitalWrite(ledVermelho, LOW);
        digitalWrite(buzzer, LOW);
        digitalWrite(ledAmarelo, LOW);
    }
}

//enviando os dados JSON do mpu6050 para o TAGO.IO
void envia_msg()
{
    client.publish("node/acelerometro", json_ax);
    client.publish("node/acelerometro", json_ay);
    client.publish("node/acelerometro", json_az);
    client.publish("node/acelerometro", json_zScoreX);
    client.publish("node/acelerometro", json_zScoreY);
    client.publish("node/acelerometro", json_zScoreZ);

}

//conexão do ESP32 com o tópico actuators do TAGO.IO
void onConnectionEstablished()
{
    client.subscribe("node/actuators", [] (const String &payload) {
        Serial.println(payload);
        processa_msg(payload);
    });
}

//traz os dados do TAGO.IO e faz as condições para ligar os leds e o buzzer
void processa_msg(const String payload)
{
    StaticJsonDocument<300> msg;
```

```
DeserializationError err = deserializeJson(msg, payload);
if (err) {
  Serial.print(F("deserializeJson() failed with code "));
  Serial.println(err.f_str());
}
Serial.print("var:");
String var = msg["variable"];
Serial.println(var);

if(var == "ax")
{
  Serial.print("value:");
  String val = msg["value"];
  float valVib = val.toFloat();
  Serial.println(valVib);
}
}
```

## 5.6. PROTOCOLO E PLATAFORMA USADOS

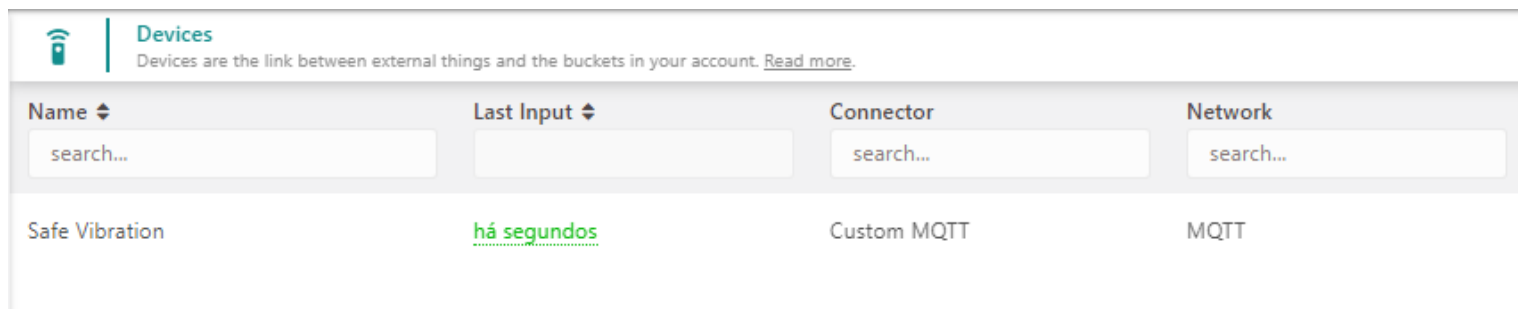
A plataforma utilizada para receber e armazenar os dados, gerar dashboards e enviar mensagens para o ESP32 é o Tago.io, sendo ele a primeira plataforma de cloud criada para o monitoramento de ambiente iot. O Tago a partir de dados recebidos com a utilização do protocolo mqtt pode gerar em tempo real, uma dashboard que exibirá todas as informações através de diferentes widgets

O protocolo Mqtt é baseado em mensagens que serão emitidas entre computadores, muito utilizado por dispositivos IOT, geralmente suas conexões com rede possuem grandes limitações de recursos e uma pequena disponibilidade de banda larga. Seu funcionamento é a partir de um modelo de publicação e assinaturas.

Na comunicação de redes, normalmente existe um cliente, que efetua uma requisição e um servidor, que devolve ao cliente uma resposta, no caso do protocolo Mqtt existe um terceiro dispositivo que tem a função de filtrar as mensagens recebidas pelo servidor assim distribuindo corretamente para os clientes além de servir como um desacoplador, esse dispositivo no protocolo Mqtt é conhecido como Broker, isso garante a segurança para o cliente protegendo o seu ip de ser adquirido pelo assinante.

## 5.7. INTEGRAÇÃO

Para realizar a integração do dispositivo ESP32 com o Tago.io, fora utilizado uma função da biblioteca “**EspMQTTClient**”, que pode conectar o dispositivo ao ip do Tago.io sendo uma conexão ao Mqtt server broker do próprio Tago.io, ademais utilizando a mesma função, foi conectado um token da plataforma Tago, que tem a utilidade de reconhecer o dispositivo Esp como um dispositivo específico criado na plataforma Tago.



Name	Last Input	Connector	Network
Safe Vibration	há segundos	Custom MQTT	MQTT

O envio de mensagens JSON, dentro de um tópico mqtt (node/acelerometro), que é enviado utilizando a biblioteca “EspMQTTClient” com a função “client.publish” que ao serem recebidas pelo tago são imediatamente armazenadas dentro do bucket, através de uma action (sistema de ações do próprio site)

⚡

Receber Valores  
Type MQTT Topic | Action Insert to Device Bucket | Last triggered at há 16 horas

Active ✓

Action

Tags

More

Trigger

•

Single device

Watch a single device.

○

Multiple devices

Watch all devices with matching tags.

#

Subscribe

if data arrives in one of the topics, the action will be triggered.

MQTT Topic

node/acelerometro

-

+

[Learn more about Trigger by MQTT Topic](#)

Select the device

Safe Vibration

×

Action

✎

Name

Receber Valores

⚡

Type of action

[Learn more about this Action type](#)

Insert to Device Bucket

×

ⓘ

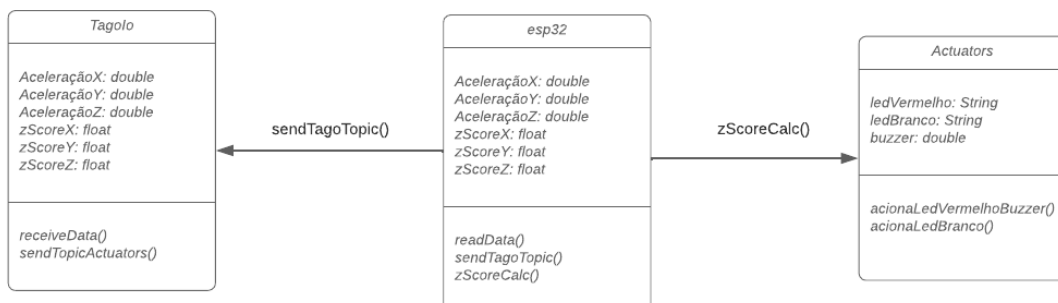
How will the data be inserted?

The data will be inserted as raw format, if it's not in TagoIO format you must use our [Payload Parser](#).  
You can debug your data by using the Device Inspector.

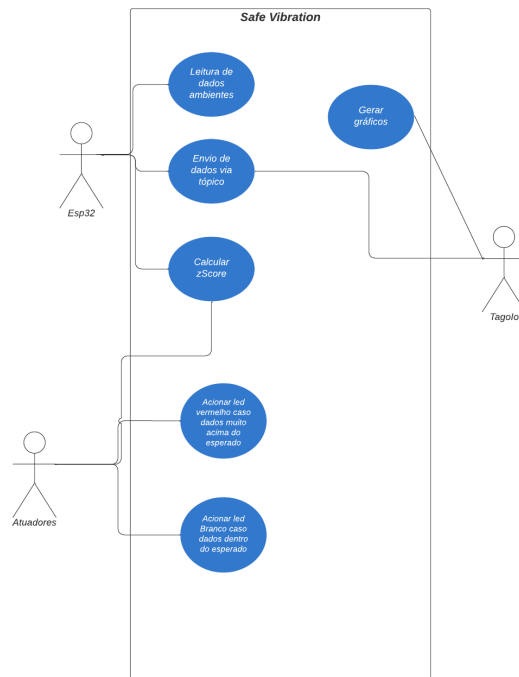
A partir desses dados agora armazenados no bucket, é gerada uma dashboard para acompanhar os dados em tempo real (5.12).

## 5.8. DIAGRAMAS

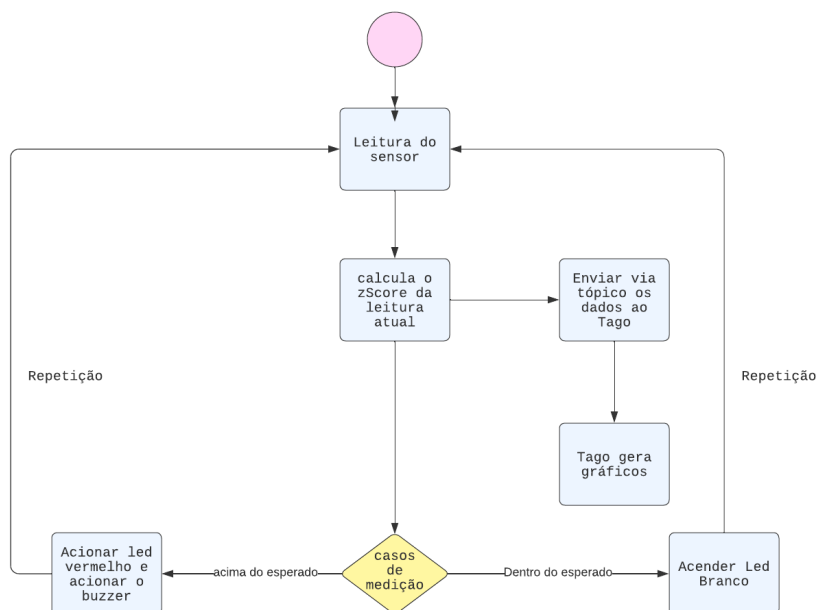
- Diagrama de classes



- Diagrama de casos de usos



- Diagrama de atividades

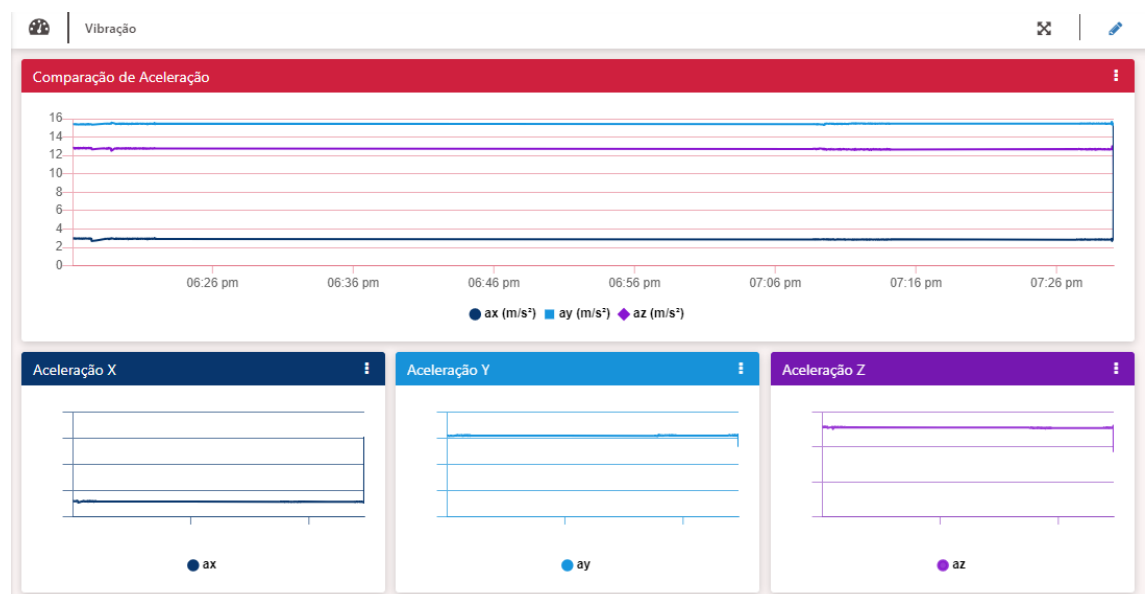


## 5.9. DASHBOARD

O Dashboard é nosso painel de informações, onde fica fácil de visualizar e monitorar a aceleração de qualquer eixo. É possível analisar os eixos tanto em conjunto quanto separadamente, fazendo com que tenhamos mais atenção aos detalhes das acelerações captadas. Além de análise de aceleração, o sistema também tem o Dashboard de pontos fora da curva, nossos outliers, que são vibrações fora do esperado e que são tidas como o nosso foco, tendo em vista que o objetivo do projeto é monitorar as vibrações e alertar quando ficam perigosas. Esses Outliers são calculados diretamente no código do nosso sensor, e exibidos no dashboard instantaneamente, e nos trazem uma perspectiva ainda mais aproximada da aceleração nos eixos, assim como esperado.

### 5.9.1 DASHBOARD VIBRAÇÃO

A dashboard Vibração é onde monitoramos a aceleração dos eixos em  $m/s^2$ . Representados em gráficos de linhas, em cores distintas, exibimos os eixos X, Y e Z.



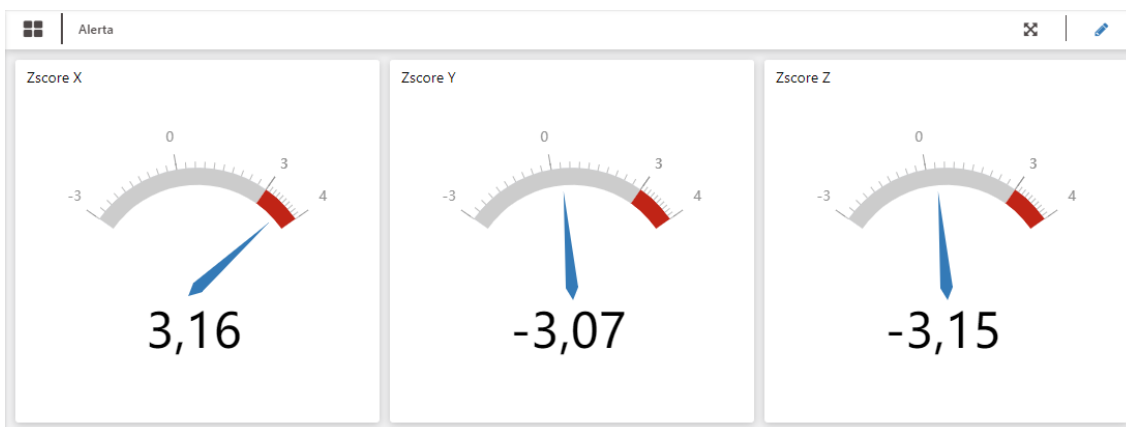
### 5.9.2 DASHBOARD ZSCORE

O dashboard Zscore é onde monitoramos os nossos outliers, que são as vibrações atípicas que o sensor MPU 6050 capta. Os outliers são o foco do nosso projeto, o que torna esse dashboard o mais importante, já que o cálculo é a principal funcionalidade.



### 5.9.3 DASHBOARD ALERTA

O dashboard Alerta também monitora os outliers, porém exibe em forma de velocímetro a fim de facilitar a visualização e entendimento.



---

## **6. CONCLUSÃO**

Com esse projeto conseguimos adquirir o conhecimento de como utilizar o sensor MPU 6050 de uma forma prática, onde utilizamos seus valores de aceleração, calculamos e detectamos suas alterações, assim permitindo monitorar as vibrações de uma estrutura determinada. Com isso, é possível obter informações importantes sobre as condições da estrutura, identificar possíveis anomalias e tomar as medidas necessárias para garantir a segurança das pessoas e a preservação do patrimônio. O uso de um microcontrolador como a placa ESP32WROOM e do sensor MPU 6050 torna o sistema de monitoramento de baixo custo e de fácil integração. Além disso, a integração do sistema com a página web permite a visualização dos dados de forma clara e acessível. Desta forma, o projeto pode ser útil em diversos contextos, mas durante o desenvolvimento do projeto, mantivemos o foco em estruturas de edifícios, contribuindo para a melhoria das condições de segurança e qualidade no ambiente para os cidadãos.



## 7.REFERÊNCIAS

Título- [Queda de prédio em Bangladesh é atribuída à vibração de geradores - Jornal O Globo](#)

Autor- O Globo e Com agências internacionais

Acessado- 05/06

Título- [ENGENHARIA CIVIL E A VIBRAÇÃO NATURAL DAS ESTRUTURAS](#)

Autor- Causos Concretos

Acessado- 06/06

Título- [Champlain Towers: Desabamento em Surfside ainda não elucidado](#)

Autor- AMG International Realty

Acessado- 07/06

Link- <https://www.mundodaeletrica.com.br/o-que-e-um-led/>

Autor- Mundo da Elétrica

Acessado- 20/06

Link- <https://embarcados.com.br/mqtt-protocolos-para-iot/>

Autor- Marcelo Barros

Acessado- 09/05/2023

Link-

<https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/>.

Autor-HiveMQ

Acessado- 09/05/2023

Link-

<https://aws.amazon.com/pt/what-is/mqtt/#:~:text=O%20protocolo%20MQTT%20foi%20inventado,para%20monitorar%20oleodutos%20via%20sat%C3%A9lite.>

Autor- Aws

Acessado- 08/05/2023

---

Link- <https://engprocess.com.br/mqtt-broker/>

Autor- EngProcess

Acessado- 08/05/2023