TRƯỜNG ĐẠI HỌC KHOA HỌC

KHOA CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI: " Hệ thống giám sát độ rung với ESP32"

Dùng cảm biến gia tốc MPU6050 để phát hiện rung động, ứng dụng trong bảo vệ thiết bị.

HỌC PHẦN : Phát triển ứng dụng IoT

Giảng viên hướng dẫn : Võ Việt Dũng

Chuyên ngành : Công nghệ phần mềm

Khóa học : K45

HUẾ, THÁNG 4 NĂM 2025

MỤC LỤC

1. Giới thiệu

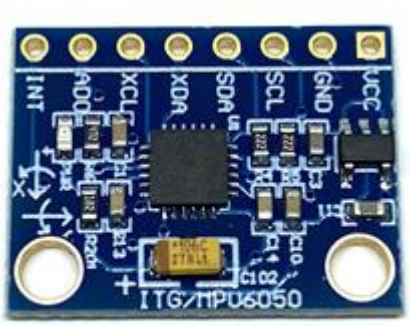
Trong thời đại công nghiệp hóa và tự động hóa ngày càng phát triển, việc giám sát độ rung của thiết bị đóng vai trò quan trọng trong bảo trì và đảm bảo an toàn vận hành. Độ rung bất thường có thể là dấu hiệu cảnh báo sớm về các sự cố kỹ thuật như lệch trục, hao mòn linh kiện hoặc thậm chí hỏng hóc nghiêm trọng. Việc giám sát và phân tích rung động giúp các nhà máy, xí nghiệp và công trình xây dựng có thể đưa ra biện pháp bảo trì dự đoán, giảm thiểu rủi ro và tăng hiệu suất hoạt động của thiết bị.

Hệ thống giám sát độ rung sử dụng vi điều khiển ESP32 kết hợp với cảm biến MPU6050 là một giải pháp thông minh, chi phí thấp nhưng mang lại hiệu quả cao. MPU6050 là một cảm biến gia tốc và con quay hồi chuyển có thể đo lường rung động theo ba trục X, Y, Z, giúp cung cấp dữ liệu chi tiết về trạng thái chuyển động của thiết bị. ESP32 với khả năng xử lý mạnh mẽ và kết nối không dây giúp truyền dữ liệu đo lường đến hệ thống giám sát từ xa, hỗ trợ phân tích và cảnh báo nhanh chóng.

Bài tiểu luận này sẽ trình bày chi tiết về nguyên lý hoạt động của cảm biến MPU6050, vai trò của vi điều khiển ESP32 trong hệ thống, cũng như phương pháp đo và xử lý dữ liệu rung động. Ngoài ra, bài viết cũng sẽ đề cập đến thiết kế phần cứng, lập trình hệ thống và các ứng dụng thực tế trong công nghiệp và đời sống.

2. Tổng quan về cảm biến MPU6050

MPU6050 là một trong những cảm biến đo lường quán tính (IMU - Inertial Measurement Unit) phổ biến nhất hiện nay. Đây là một module cảm biến 6 trục, bao gồm một gia tốc kế ba trục và một con quay hồi chuyển ba trục, cho phép đo lường chuyển động và rung động một cách chính xác.



cảm biến MPU6050

2.1 Cấu tạo và nguyên lý hoạt động

Cảm biến MPU6050 bao gồm:

- Gia tốc kế 3 trục: Giúp đo gia tốc của vật thể theo ba phương X, Y, Z.
- Con quay hồi chuyển 3 trục: Đo vận tốc góc của vật thể theo ba phương.
- Bộ xử lý tín hiệu số (DMP - Digital Motion Processor)**:** Hỗ trợ xử lý và lọc nhiễu tín hiệu để cung cấp dữ liệu chính xác.
- Giao tiếp I2C/SPI: MPU6050 chủ yếu sử dụng giao tiếp I2C để kết nối với vi điều khiển.

Nguyên lý hoạt động của MPU6050 dựa trên việc đo gia tốc và vận tốc góc của vật thể. Gia tốc kế xác định mức gia tốc của thiết bị khi có sự thay đổi về vị trí hoặc độ nghiêng. Con quay hồi chuyển đo tốc độ quay quanh các trục, từ đó có thể xác định mức độ rung động hoặc sự thay đổi góc quay.

2.2 Đặc điểm kỹ thuật chính

- Điện áp hoạt động: 3.3V - 5V
- Giao tiếp: I2C (địa chỉ 0x68 hoặc 0x69)
- Phạm vi đo gia tốc: ±2g, ±4g, ±8g, ±16g
- Phạm vi đo vận tốc góc: ±250, ±500, ±1000, ±2000 độ/giây

- Tích hợp bộ lọc số DMP giúp giảm nhiễu
- Tích hợp cảm biến nhiệt độ

2.3 Ưu điểm của MPU6050

- Kết hợp cả gia tốc kế và con quay hồi chuyển trong một module nhỏ gọn, giúp tiết kiệm không gian.
- Độ nhạy cao, có thể đo được những rung động nhỏ.
- Có thể kết hợp với các cảm biến khác để đo lường chính xác hơn (ví dụ: cảm biến từ trường để đo góc quay tuyệt đối).
- Chi phí thấp, dễ dàng tích hợp vào các hệ thống nhúng.

2.4 Ứng dụng của MPU6050 trong giám sát độ rung

- Giám sát máy móc công nghiệp: Phát hiện các rung động bất thường có thể gây hỏng hóc thiết bị.
- Hệ thống cảnh báo động đất: MPU6050 có thể đo các dao động nhỏ, giúp cảnh báo trước các trận động đất.
- Kiểm soát ổn định thiết bị: Dùng trong các hệ thống cân bằng tự động.
- Ứng dụng trong y tế: Theo dõi chuyển động của bệnh nhân để phân tích tư thế và phát hiện rối loạn vận động.

3. Vi điều khiển ESP32

3.1 Giới thiệu về ESP32

ESP32 là một vi điều khiển mạnh mẽ được phát triển bởi Espressif Systems, nổi bật với khả năng kết nối không dây và hiệu suất xử lý cao. Nó được sử dụng rộng rãi trong các ứng dụng IoT, tự động hóa, và các hệ thống nhúng nhờ vào khả năng hỗ trợ Wi-Fi và Bluetooth.

Vi điều khiển ESP32

3.2 Cấu trúc và đặc điểm kỹ thuật

ESP32 có các đặc điểm chính sau:

- Bộ vi xử lý: CPU lõi kép Xtensa LX6, xung nhịp lên đến 240MHz.
- Bộ nhớ: RAM 520KB, hỗ trợ thêm bộ nhớ ngoài.
- Kết nối: Wi-Fi 802.11 b/g/n, Bluetooth 4.2 BLE.
- Giao tiếp: Hỗ trợ nhiều giao thức như I2C, SPI, UART, ADC, PWM, DAC.
- Điện áp hoạt động: 2.2V - 3.6V.
- Tích hợp cảm biến nhiệt độ và bộ quản lý năng lượng giúp tiết kiệm điện.

3.3 Ưu điểm của ESP32

- Hiệu suất cao: Với CPU lõi kép mạnh mẽ, ESP32 có thể xử lý các tác vụ phức tạp nhanh chóng.
- Kết nối không dây: Hỗ trợ Wi-Fi và Bluetooth giúp dễ dàng tích hợp vào hệ thống IoT.
- Đa dạng cổng giao tiếp: Cho phép kết nối với nhiều loại cảm biến và thiết bị ngoại vi.
- Tiết kiệm năng lượng: Có nhiều chế độ ngủ giúp giảm tiêu thụ điện năng, phù hợp cho các ứng dụng chạy bằng pin.
- Chi phí thấp: So với các vi điều khiển có cùng tính năng, ESP32 có giá thành hợp lý, dễ tiếp cận.

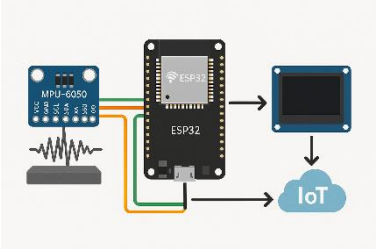
3.4 Ứng dụng của ESP32 trong giám sát độ rung

ESP32 đóng vai trò trung tâm trong hệ thống giám sát độ rung, giúp:

- Thu thập dữ liệu từ MPU6050 thông qua giao tiếp I2C.
- Xử lý dữ liệu rung động để phát hiện bất thường.
- Gửi dữ liệu lên máy chủ hoặc nền tảng IoT để giám sát từ xa.
- Kích hoạt cảnh báo khi phát hiện rung động vượt mức cho phép.

ESP32 có thể được lập trình bằng Arduino IDE, PlatformIO hoặc ESP-IDF, giúp lập trình viên dễ dàng phát triển các ứng dụng.

4. Phương pháp đo và phân tích độ rung



Hệ thống sử dụng MPU6050 để thu thập dữ liệu rung động. Dữ liệu được gửi về ESP32 để xử lý, sau đó hiển thị lên màn hình hoặc gửi lên nền tảng IoT để giám sát từ xa.

4.1 Các bước đo rung động

1. Đọc dữ liệu từ cảm biến MPU6050: Dữ liệu gia tốc và con quay hồi chuyển được lấy theo ba trục X, Y, Z.
2. Tiền xử lý dữ liệu: Lọc nhiễu sử dụng thuật toán Kalman hoặc trung bình động (Moving Average) để loại bỏ các tín hiệu không mong muốn.
3. Tính toán biên độ và tần số rung động: Áp dụng biến đổi Fourier (FFT) để phân tích phổ tần số của tín hiệu rung.
4. Đánh giá mức độ rung: So sánh giá trị đo được với ngưỡng rung định trước để xác định tình trạng hoạt động của thiết bị.
5. Lưu trữ hoặc truyền dữ liệu: Ghi dữ liệu vào bộ nhớ hoặc gửi qua Wi-Fi/Bluetooth để phục vụ phân tích và giám sát từ xa.

4.2 Phân tích phổ rung động

Một trong những kỹ thuật quan trọng để phân tích độ rung là biến đổi Fourier (FFT - Fast Fourier Transform). Kỹ thuật này giúp chuyển tín hiệu rung từ miền thời gian sang miền tần số, từ đó xác định được các thành phần dao động chính, tần số cộng hưởng hoặc các dao động bất thường.

- Tần số dao động thấp có thể liên quan đến mất cân bằng cơ học.
- Tần số cao có thể là dấu hiệu của mòn vòng bi hoặc các chi tiết nhỏ bị lỏng lẻo.

4.3 Xử lý bất thường và cảnh báo

Sau khi phân tích tín hiệu rung động, nếu giá trị vượt ngưỡng an toàn, hệ thống sẽ:

- Kích hoạt cảnh báo tại chỗ (đèn LED, còi báo).
- Gửi thông báo đến máy chủ, điện thoại hoặc email.
- Ghi nhận sự kiện để phục vụ truy vết và bảo trì sau này.

4.4 Độ chính xác và hiệu quả

- Việc lựa chọn thuật toán lọc và phân tích phù hợp sẽ quyết định đến độ chính xác của hệ thống.
- Việc hiệu chuẩn cảm biến MPU6050 định kỳ cũng rất cần thiết để duy trì tính ổn định của dữ liệu đo lường.

Thông qua quy trình đo và phân tích nêu trên, hệ thống không chỉ theo dõi rung động theo thời gian thực mà còn giúp phát hiện sớm các dấu hiệu hỏng hóc tiềm ẩn.

5. Thiết kế hệ thống

Hệ thống giám sát độ rung sử dụng ESP32 được thiết kế với mục tiêu đơn giản, hiệu quả và tiết kiệm chi phí. Thiết kế bao gồm cả phần cứng và phần mềm, đảm bảo khả năng thu thập, xử lý và truyền dữ liệu rung động một cách chính xác và thời gian thực

5.1 Kiến trúc tổng thể

Hệ thống bao gồm các thành phần chính như sau:

- Cảm biến MPU6050: Đo rung động thông qua dữ liệu gia tốc và con quay hồi chuyển.
- Vi điều khiển ESP32: Thu thập dữ liệu từ cảm biến, xử lý và truyền thông tin.

- Màn hình hiển thị (OLED/LCD)**:** Hiển thị trực tiếp dữ liệu rung hoặc thông báo cảnh báo.
- Đèn LED/Còi báo: Cảnh báo người dùng khi phát hiện rung vượt ngưỡng an toàn.
- Nguồn cấp điện: Pin Li-ion, adapter hoặc nguồn từ USB.
- Kết nối Wi-Fi hoặc Bluetooth: Gửi dữ liệu đến máy chủ giám sát hoặc ứng dụng di động.

5.2 Sơ đồ kết nối phần cứng

- MPU6050 kết nối với ESP32 qua giao thức I2C:
  - VCC → 3.3V ESP32
  - GND → GND ESP32
  - SDA → GPIO21 (SDA)
  - SCL → GPIO22 (SCL)
- Màn hình OLED (nếu sử dụng) cũng sử dụng cùng giao tiếp I2C.
- Đèn LED kết nối với một chân GPIO bất kỳ (ví dụ GPIO 4).
- Còi buzzer (nếu có) cũng nối với một chân GPIO.

5.3 Sơ đồ khối hệ thống

Hệ thống bao gồm các thành phần chính như sau:

- ESP32**:** Vi điều khiển trung tâm xử lý và truyền dữ liệu.
- MPU6050**:** Đo độ rung, chuyển động theo trục X, Y, Z.
- OLED 0.96 inch (tùy chọn): Hiển thị dữ liệu rung đo được, trạng thái cảnh báo,...
- Đèn LED màu đỏ : Hiển thị cảnh báo rung vượt ngưỡng.

5.4 Yêu cầu phần mềm

- Arduino IDE hoặc PlatformIO: Để lập trình ESP32.
- ESP32 Board Support Package: gói phần mềm giúp Arduino IDE hiểu cách biên dịch và upload code cho ESP32**.**

- Thư viện MPU6050.h: Đọc và xử lý dữ liệu từ cảm biến.
- Thư viện Adafruit GFX: Được dùng kèm với SSD1306 để điều khiển hiển thị trên OLED.
- Thư viện Adafruit_SSD1306 (nếu dùng OLED): Điều khiển màn hình OLED SSD1306 thông qua I2C.

5.5 Sơ đồ nguyên lý kết nối phần cứng

Thiết bị	Chân kết nối MPU6050	ESP32 GPIO
MPU6050	VCC	3.3V
	GND	GND
	SDA	GPIO 21
	SCL	GPIO 22
OLED	SDA	GPIO 21
	SCL	GPIO 22
LED	-	GPIO 2 (hoặc bất kỳ chân digital nào còn trống)

5.6 Nguyên lý hoạt động

1. Khởi động hệ thống: ESP32 khởi tạo kết nối với cảm biến MPU6050, màn hình OLED.
2. Thu thập dữ liệu: ESP32 đọc giá trị gia tốc theo 3 trục từ MPU6050 theo chu kỳ thời gian.
3. Tính toán rung động: Từ dữ liệu thu thập, hệ thống tính toán độ rung tổng quát bằng công thức:

   Vibration = sqrt(ax^2 + ay^2 + az^2)

4. Hiển thị dữ liệu: Dữ liệu được hiển thị trên màn hình OLED.

5. Cảnh báo: Nếu độ rung vượt ngưỡng định sẵn, hệ thống kích hoạt LED.

6. Lập trình hệ thống

Hệ thống được lập trình bằng ngôn ngữ Arduino C++ sử dụng thư viện hỗ trợ MPU6050, SSD1306 và kết nối Wi-Fi của ESP32. Phần mềm thực hiện nhiều tác vụ đồng thời như thu thập dữ liệu từ cảm biến, hiển thị thông tin, phân tích độ rung và gửi cảnh báo qua internet.

6.1 Khởi tạo phần cứng và thư viện

- Sử dụng thư viện Wire.h để giao tiếp I2C với cảm biến MPU6050 và màn hình OLED.
- Thư viện MPU6050.h giúp dễ dàng truy xuất dữ liệu từ cảm biến.
- Thư viện Adafruit_SSD1306.h và Adafruit_GFX.h hỗ trợ hiển thị OLED.
- WiFi.h và HTTPClient.h dùng để truyền dữ liệu đến máy chủ qua giao thức HTTP.

6.2 Cấu hình ngưỡng cảnh báo

Ngưỡng rung động được thiết lập ở mức 1.5g (đơn vị gia tốc trọng trường). Khi độ rung đo được vượt qua ngưỡng này, hệ thống sẽ bật LED và gửi dữ liệu đến server.

6.3 Kết nối Wi-Fi

Trong hàm setup(), ESP32 sẽ tự động kết nối Wi-Fi bằng tên mạng và mật khẩu được khai báo. Kết nối này cần thiết để truyền dữ liệu lên internet.

6.4 Đọc dữ liệu từ MPU6050

Sử dụng hàm getAcceleration() để đọc giá trị gia tốc theo ba trục x, y, z. Sau đó chuyển đổi dữ liệu từ đơn vị số nguyên sang gia tốc (g).

6.5 Tính toán độ rung tổng

Giá trị độ rung tổng được tính bằng công thức:

float vibration = sqrt(accX * accX + accY * accY + accZ * accZ);

Giá trị này phản ánh mức độ dao động tổng hợp của thiết bị.

6.6 Hiển thị và cảnh báo

OLED hiển thị giá trị rung động liên tục. Nếu rung động vượt ngưỡng, LED sẽ được bật. Đồng thời, hàm sendData() sẽ gửi dữ liệu độ rung lên server**.**

6.7 Mã chương trình hoàn chỉnh và sơ đồ wokwi

* Dưới đây là đoạn mã đầy đủ cho hệ thống giám sát độ rung sử dụng ESP32 và MPU6050:

```
#include <Wire.h>
#include <MPU6050.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <math.h>

// MPU6050
MPU6050 mpu;

// OLED display
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

// LED cảnh báo
const int ledPin = 13;

// Ngưỡng rung (g)
const float THRESHOLD = 0.2;

// Calibration
float baseline = 1.0;
const int CALIB_SAMPLES = 50;

// Moving Average
#define MA_WINDOW 5
float maBuffer[MA_WINDOW] = {0};
int maIndex = 0;

// Non-blocking timing
unsigned long lastMs = 0;
const unsigned long INTERVAL = 200; // ms

// Hàm calibrate để lấy baseline khi khởi động
void calibrate() {
  Serial.println("Calibrating...");
  long sum = 0;
  for (int i = 0; i < CALIB_SAMPLES; i++) {
    int16_t ax, ay, az;
    mpu.getAcceleration(&ax, &ay, &az);
    float total = sqrt(sq(ax) + sq(ay) + sq(az)) / 16384.0;
```

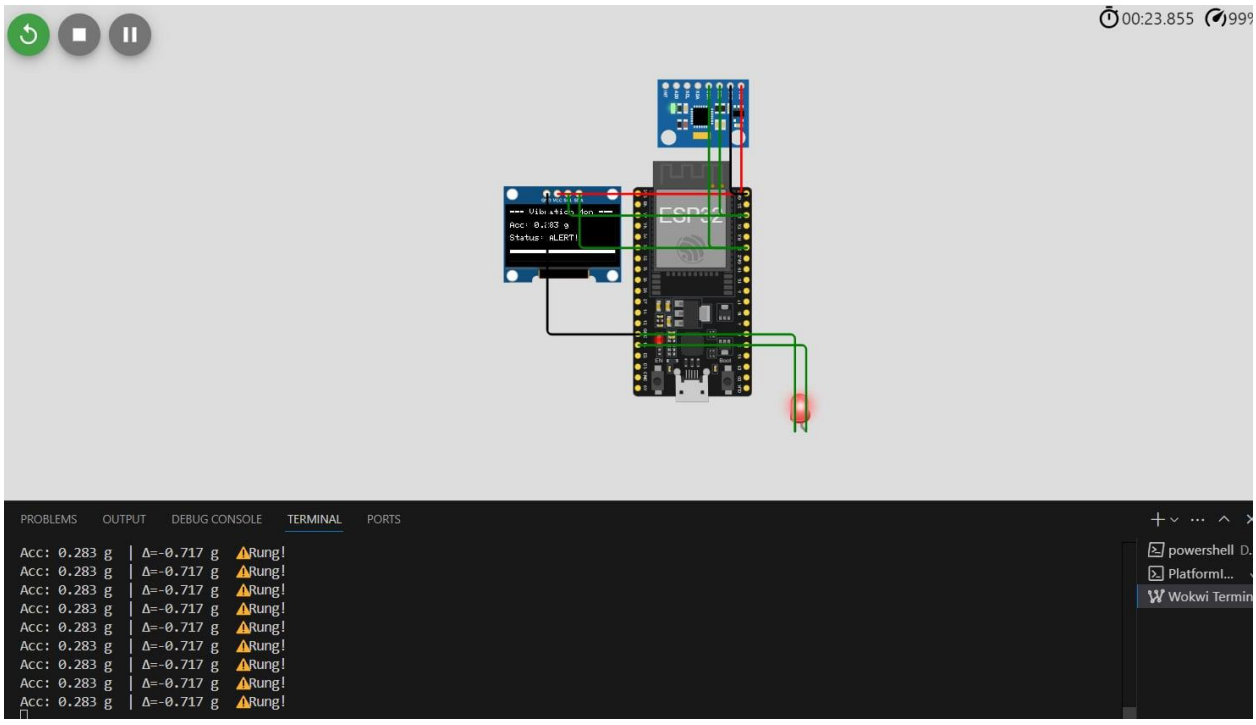
```
    sum += total * 1000;    // nhân để giữ phần thập phân
    delay(20);
  }
  baseline = sum / (CALIB_SAMPLES * 1000.0);
  Serial.print("Baseline = ");
  Serial.print(baseline, 3);
  Serial.println(" g");
}

// Hàm moving average
float movingAverage(float newVal) {
  maBuffer[maIndex] = newVal;
  maIndex = (maIndex + 1) % MA_WINDOW;
  float sum = 0;
  for (int i = 0; i < MA_WINDOW; i++) sum += maBuffer[i];
  return sum / MA_WINDOW;
}

void setup() {
  Serial.begin(115200);
  // Khởi động I2C trên pins custom (SDA=14, SCL=12)
  Wire.begin(21,22);

  // Khởi động MPU6050
  mpu.initialize();
  if (!mpu.testConnection()) {
    Serial.println("❌ Không tìm thấy MPU6050!");
    while (1);
  }
```

```
  // Calibrate baseline
  calibrate();

  // Khởi động OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("❌ OLED không khởi động được!"));
    while (1);
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println("=== Vibration Mon ===");
  display.display();

  // LED cảnh báo
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
}

void loop() {
  // Non-blocking delay
  if (millis() - lastMs < INTERVAL) return;
  lastMs = millis();

  // Đọc gia tốc
  int16_t ax, ay, az;
  mpu.getAcceleration(&ax, &ay, &az);
  float fx = ax / 16384.0;
  float fy = ay / 16384.0;
```

```
float fz = az / 16384.0;

// Tính raw tổng gia tốc
float rawAcc = sqrt(fx*fx + fy*fy + fz*fz);
// Lọc nhiễu bằng moving average
float totalAcc = movingAverage(rawAcc);

// Phát hiện rung
bool isVibrating = abs(totalAcc - baseline) > THRESHOLD;

// Hiển thị lên Serial
Serial.print("Acc: ");
Serial.print(totalAcc, 3);
Serial.print(" g");
Serial.print("  | Δ=");
Serial.print(totalAcc - baseline, 3);
Serial.print(" g");
Serial.println(isVibrating ? "  ⚠ Rung!" : "  OK");

// --- Cập nhật OLED ---
// 1. Giá trị Acc
display.fillRect(0, 16, SCREEN_WIDTH, 16, SSD1306_BLACK);
display.setCursor(0, 16);
display.printf("Acc: %.3f g", totalAcc);

// 2. Trạng thái
display.fillRect(0, 32, SCREEN_WIDTH, 16, SSD1306_BLACK);
display.setCursor(0, 32);
display.print("Status: ");
display.println(isVibrating ? "ALERT!" : "Normal");
```

```
  // 3. Thanh đồ thị biểu diễn mức rung (max 0.5g tương ứng full width)
  float delta = abs(totalAcc - baseline);
  int barLen = map(constrain((int)(delta * 1000), 0, 500), 0, 500, 0, SCREEN_WIDTH);
  display.fillRect(0, 50, SCREEN_WIDTH, 4, SSD1306_BLACK);
  display.fillRect(0, 50, barLen, 4, SSD1306_WHITE);
  display.drawRect(0, 50, SCREEN_WIDTH, 4, SSD1306_WHITE);

  display.display();

  // LED cảnh báo
  digitalWrite(ledPin, isVibrating ? HIGH : LOW);
}
```

*** Sơ đồ wokwi :**



7. Ứng dụng và triển vọng

7.1 Ứng dụng trong công nghiệp

Hệ thống giám sát độ rung với ESP32 và MPU6050 có thể ứng dụng hiệu quả trong các lĩnh vực sau:

- **Giám sát máy móc công nghiệp**: Phát hiện sớm các rung động bất thường của động cơ, máy bơm, quạt công nghiệp để ngăn ngừa hư hỏng và giảm thời gian dừng máy.
- **Giám sát cầu đường, kết cấu**: Theo dõi độ rung của các kết cấu lớn như cầu, tòa nhà, đặc biệt là sau động đất hoặc va chạm.
- **Thiết bị y tế và nghiên cứu**: Đo rung động trong các thiết bị y tế hoặc mô phỏng thí nghiệm vật lý.
- **Bảo vệ thiết bị điện tử**: Tự động ngắt nguồn hoặc báo động khi có rung mạnh để bảo vệ dữ liệu hoặc phần cứng.

7.2 Triển vọng phát triển

- Kết nối IoT: Dữ liệu độ rung có thể được đồng bộ hóa lên nền tảng IoT như Blynk, Firebase, ThingsBoard, hoặc gửi qua MQTT để giám sát từ xa.
- Ứng dụng trí tuệ nhân tạo (AI): Kết hợp với mô hình học máy để nhận diện kiểu rung đặc trưng của từng thiết bị, giúp dự đoán hỏng hóc.
- Tích hợp hệ thống cảnh báo thông minh: Gửi thông báo qua SMS, email hoặc app khi phát hiện rung vượt ngưỡng.
- Thiết kế module nhỏ gọn: Có thể gắn trực tiếp vào các thiết bị hoặc kết cấu cần giám sát mà không cần dây dẫn phức tạp.

8. Kết luận

Trong đề tài này, chúng ta đã xây dựng thành công hệ thống giám sát độ rung sử dụng ESP32 và cảm biến MPU6050. Hệ thống không chỉ đo lường và hiển thị dữ liệu rung động một cách trực quan mà còn có khả năng phát cảnh báo và gửi dữ liệu lên server để theo dõi từ xa.

Qua việc thiết kế phần cứng đơn giản, chi phí thấp, cùng với lập trình linh hoạt, hệ thống có thể mở rộng và ứng dụng trong nhiều lĩnh vực khác nhau, từ công nghiệp cho đến nghiên cứu khoa học. Đây là một bước tiếp cận thực tế với công nghệ IoT trong việc giám sát và bảo vệ thiết bị.

Trong tương lai, hệ thống có thể nâng cấp với các tính năng như học máy để nhận dạng mẫu rung bất thường, hoặc sử dụng năng lượng mặt trời để hoạt động độc lập lâu dài. Điều này mở ra cơ hội lớn cho việc triển khai giám sát tự động, thông minh trong nhiều môi trường khác nhau.

9. Tài liệu tham khảo

9.1 Tài liệu tiếng Việt

[8] Nguyễn Văn Hiệp (2020), "Ứng dụng ESP32 trong giám sát và điều khiển thiết bị", NXB Khoa học và Kỹ thuật.

[9] Trần Quốc Huy (2019), "Cảm biến và ứng dụng", Đại học Bách khoa TP.HCM.

[7] Tài liệu học Arduino cơ bản và nâng cao – www.arduino.vn

9.2 Tài liệu tiếng Anh

[1] InvenSense, "MPU6050 Product Specification", https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/

[2] Espressif Systems, "ESP32 Technical Reference Manual", https://www.espressif.com/en/support/documents/technical-documents

[3] Adafruit Industries, "Adafruit SSD1306 OLED Library", https://github.com/adafruit/Adafruit_SSD1306

[4] Arduino.cc, "Wire Library Documentation", https://www.arduino.cc/en/Reference/Wire

[5] Electronic Cats, "MPU6050 Arduino Library",
https://github.com/electroniccats/mpu6050

[6] Random Nerd Tutorials, "ESP32 with Arduino IDE",
https://randomnerdtutorials.com

[10] IBM, "Internet of Things (IoT) – A Beginner's Guide",
https://www.ibm.com/blogs/internet-of-things/iot-devices-beginners-guide/

.